

# Notes Of DBMS (SQL)

- By KITI KUMARI

# Module 1 (ER model)

## Introduction to DBMS

Data - It could be any fact that can be recorded or stored.

Eg - text, number, images, video, speech

Database - Collection of related data. (Only for text and number right now)

Text and number

Traditional database

images + videos

Multimedia database

NASA

Supermarket  
(selling some goods)

Geographical interface DB

Real time database

Data warehouse - The data is huge and historical  
It is a kind of database

Database management system - Set of programs used to define datatypes structure & constraint (place the data in the hardrive) data & manipulate it.

### Types of DB

- 1) Traditional Database
- 2) Multimedia DB
- 3) Geographical interface DB.
- 4) Real time DB.

# DB models in Database.

Modelling level  
level 1

High level view or  
Conceptual view

(E-R model)

level 2

Representational or  
implementation.

~~tables or  
views~~  
~~Relational~~

level 3.

Low level or physical (how to store,  
view.  
datatype etc)

Introduction of E-R model

entity      relationship

Entity  
A person

Attributes  
phone, address, name

Relationship  
owns a

Things

properties of  
entities

Association among  
entities

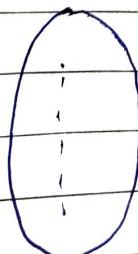
Person

name, age, phone no

works for

Entity type - Entity → (26, Raj, Bks)  
↓ (Schema)

PERSON (Age, Name, add)  
extension

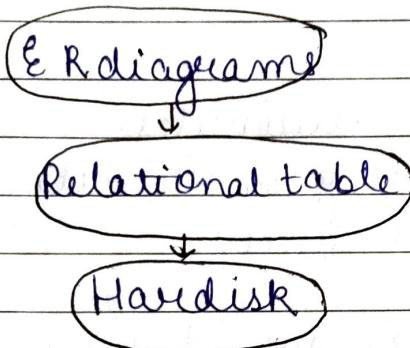


state of  
database

Schema - heading (how do you represent)

ER diagrams → Entity type  
 ↓

for a communication establishment.



DB attributes (to describe something)  
 ↓

the more attributes the better information

Person → Name, age, address, phone no.  
 (entity) (attribute)

Types of attributes

i) Simple and composite attribute

↓  
 can't be divided further      can be divided further (composed of many attributes)

Name  
 F. Name      M. Name      Last Name

NULL - the value doesn't exist or not applicable.  
eg - Middle name

Date \_\_\_\_\_ / \_\_\_\_\_ / \_\_\_\_\_



## 2) Single valued vs multivalued attribute

Only one  
value

eg - age.

More than

one value.

eg - address

(permanent or  
residential address)

## 3) Stored vs derived attributes

We store  
these values

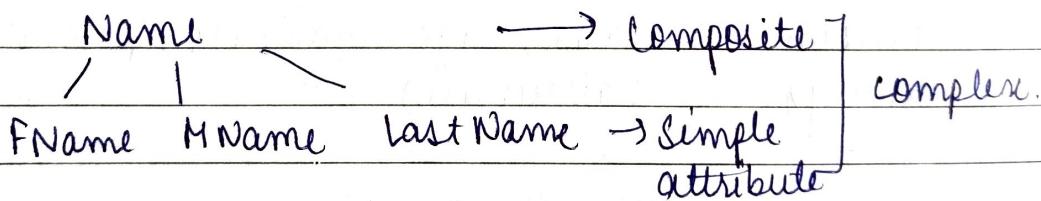
Depending on the data  
we derive

eg - DOB.

eg - Age (deriving from DOB)

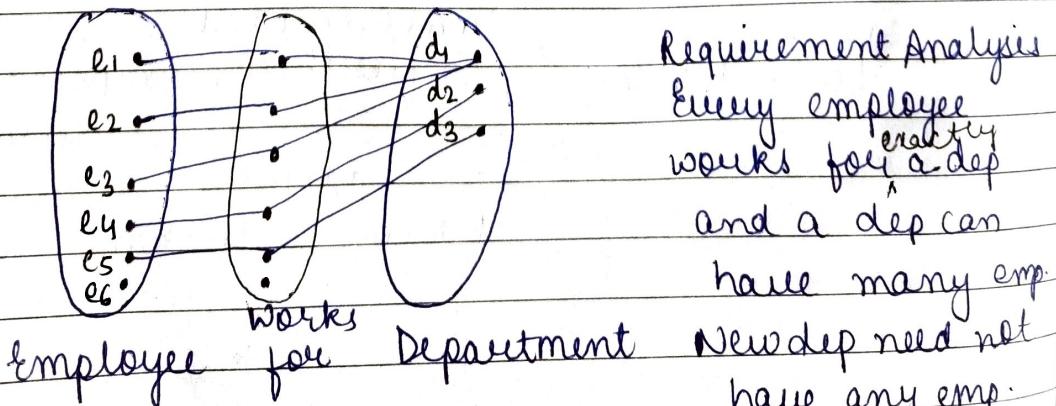
## 4) Complex attributes

Collectn of attributes or combn of attributes



DB Relationships

1 TO Many



Requirement Analysis  
Every employee works for exactly one dep  
and a dep can have many emp.  
New dep need not have any emp.



~~structured  
constraints~~ Degree - (how many entity set is participating)

Degree - 2 (Employee, department)

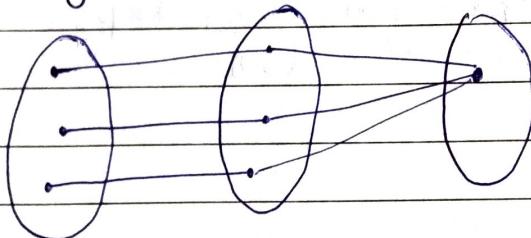
Cardinality Ratio - what is the max<sup>m</sup> no of relationships in which an entity can participate.

Cardinality 1

Employee

N

Department



Participation or existence (min<sup>m</sup> cardinality)

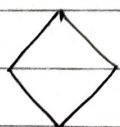
Min<sup>m</sup> relationship in which it can participate.

In case of Employee participation is 1.

New dep need not have any employee. Therefore participation is 0 in this case of dep.

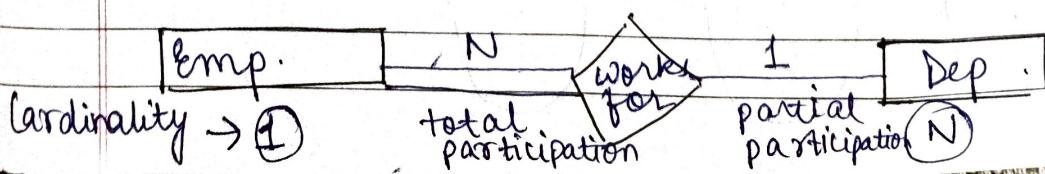


Entity



Relationship

1 dep can have many emp.



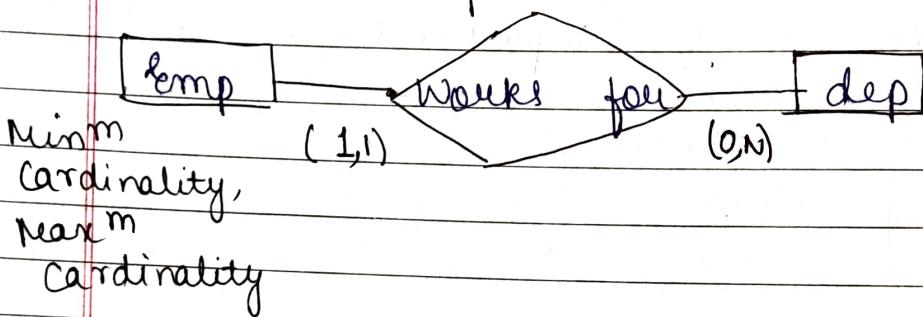


total participation  
All entities are  
participating

partial participation  
If some entities  
are participating

Cardinality double line  
ratio

Min<sup>m</sup> max representation



Min<sup>m</sup>-max<sup>m</sup> relation or representation  
elaborates more information

Role  
name

Employee

Employer

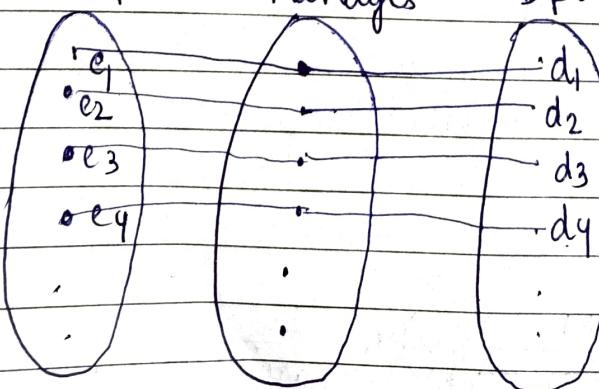
role  
name DB relationship  
(one to one)

(Employee,  
manager)

Emp.

manages

Dept



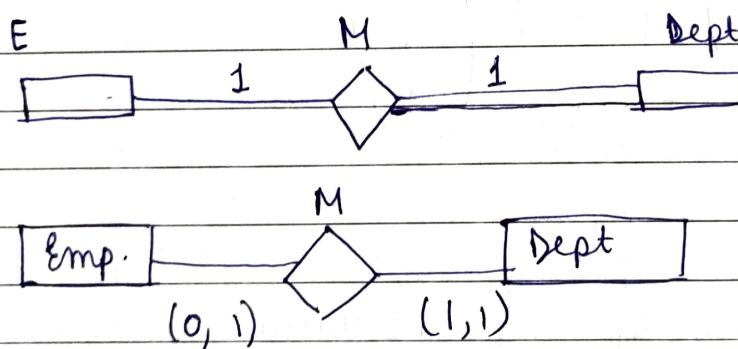
Every Dept. should have a manager and only one employee manages a dept.

Any employee can manage only one department

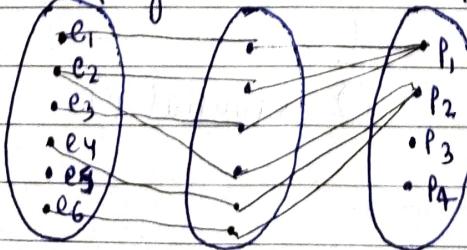
Degree  $\rightarrow 2$

Cardinality  $\rightarrow \begin{cases} 1 & (\text{max}^m \text{ no of relationship one entity can manage}) \\ 1 & \end{cases}$

Participation  $\rightarrow$  Employee (P) 0 ( $\min^m$  no of rel<sup>n</sup>)  
Dept. (D) 1 (every dept should have a manager)



DB Relationship (Many to many)  
Employee works Project



Req. Analysis - Every employee can work on atleast project & an employee can work on many projects

Degree 2

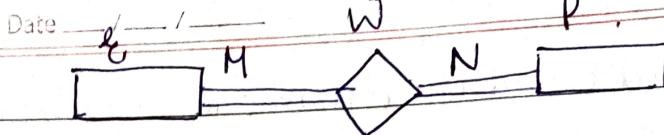
Cardinality N

Participation 1

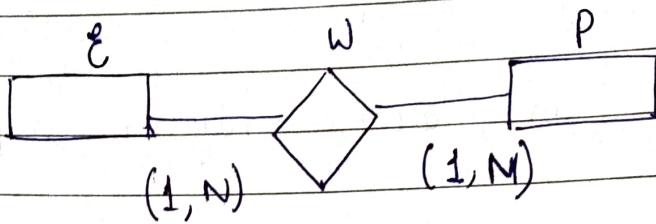
N  
1



Single L-  
Double  
line



Min-max  
reprsn'



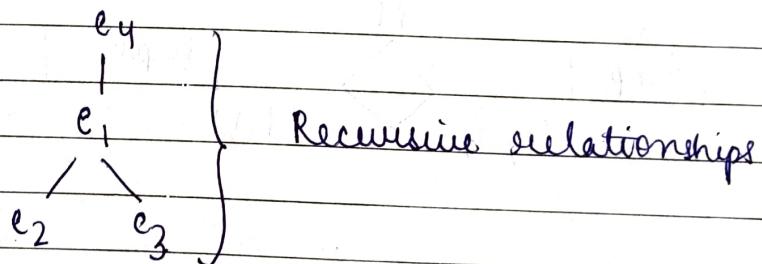
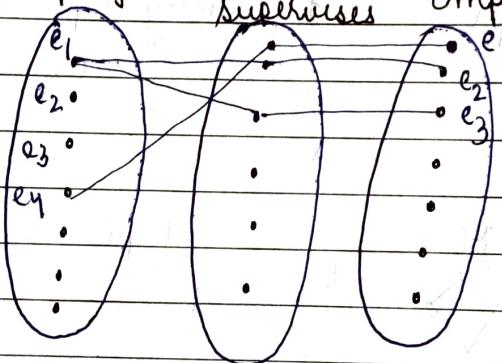
### DB recursive relationships

even though its recursive its binary (2 entities only)

An employee managing other employee  
(Boss) (Secretary)

as actuals.

Employee (a.s, supervisor)  
supervises Employee (a.s, supervisee)



Degree : 2 entities

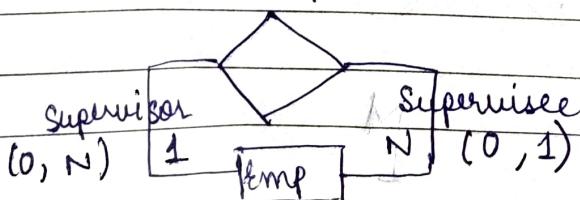
Cardinality : N

1

Participation : O

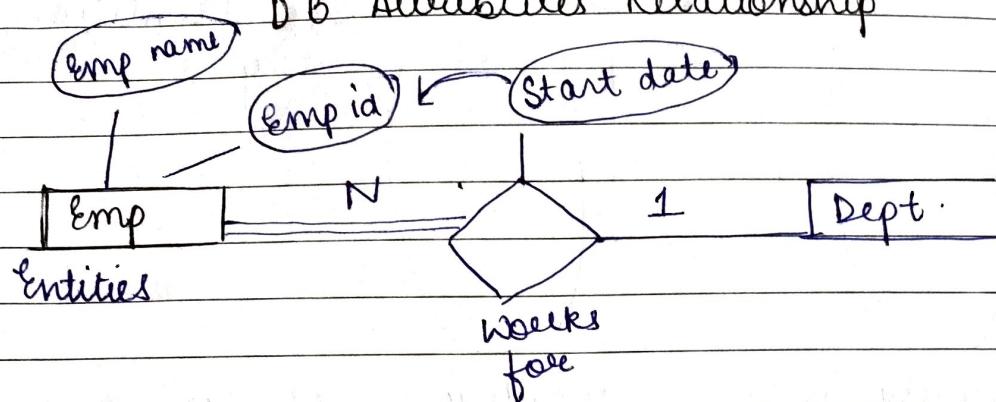
O (Partial P)

Supervises



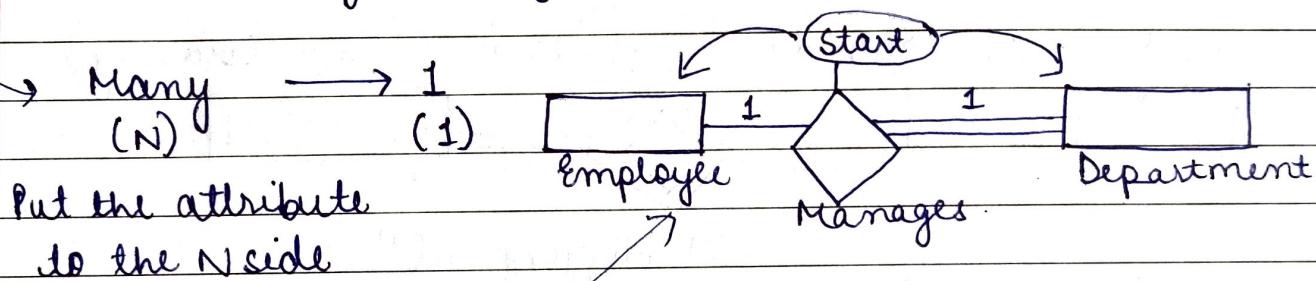


## DB Attributes Relationship



We want to have as less as attribute to the relationship

So it's better to give the attribute at the  $n$  side or many side as every employee can have a start date.



1 → 1

Any side we can move the attribute

M → N

Putting the attribute at any side is not feasible

hours

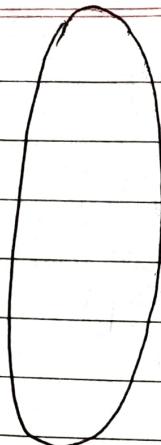
M      N  
Employee      Project  
week on

DB weak entity

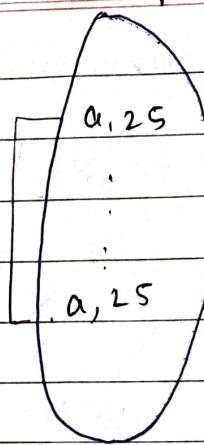
Date \_\_\_\_\_ Employee. -(Emp Id) dependents



Strong entity



weak entity



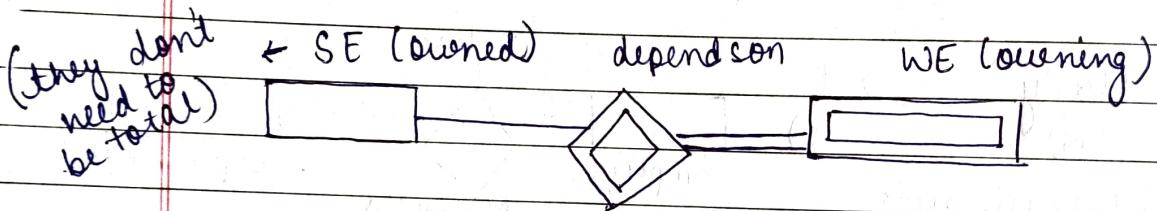
Identifying relatn.

if entity is having primary key

if entity is not having primary key.

Soln.

Every week entity should be related to strong entity.



Identifying reln.

\* The participation should always be total in identifying reln of weak entity.

(eid + Name, age)

Total participation  $\not\Rightarrow$  Weak entity  
Weak entity  $\Rightarrow$  Total participation



## DB ER diagram notations

Derived attribute



attribute



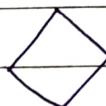
entity



Weak entity



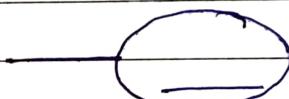
Relationship



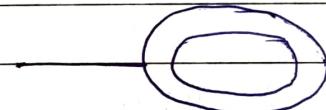
identifying rel<sup>n</sup>



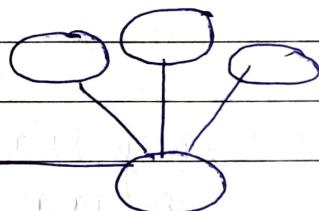
key attribute



multivalued  
attribute



Composite  
attribute



Total participation  
of E<sub>2</sub> in R



Cardinality ratio

$$E_1 : E_2 = 1 : N$$



(RDBMS) SQL - SEQUEL - Structured English query language  
↓  
simple

Date \_\_\_\_\_



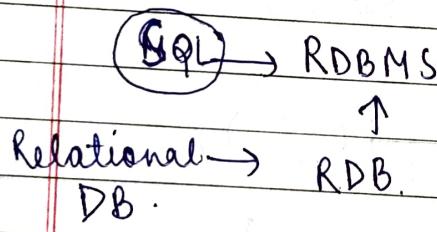
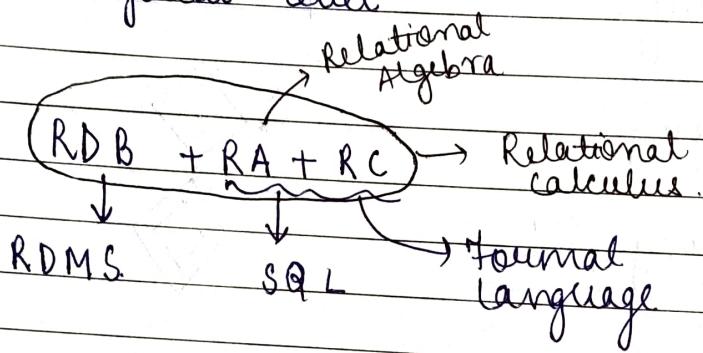
## Relational Database Model

DB - Introduction to relational model.

Conceptual level → ER diagrams

(Database model) Representation level → Relational model

Physical level



Data warehouse → huge data  
Databases. → less data

Set theory → relations → relational DB models.

SQL runs on RDBMS.

Query → converted to relational algebra.



## DB - Terminology of Relational database

1) Relation (table)


2) Tuple (row) → An entire entity


3) Attribute (column)

attribute →

0		

4) Domain Set of names/values eg - set of integers  
It is finite for assumption

int

1			

5) Relation schema (heading of the table)

Attributes R( A<sub>1</sub>, A<sub>2</sub>, A<sub>3</sub>, A<sub>4</sub>, A<sub>5</sub>)      R

↓      ↓      ↓      ↓      ↓  
Domains    D<sub>1</sub>    D<sub>2</sub>    D<sub>3</sub>    D<sub>4</sub>    D<sub>5</sub>.

A <sub>1</sub>	A <sub>2</sub>	A <sub>3</sub>	A <sub>4</sub>	A <sub>5</sub>

$\delta(R) = D_1 \times D_2 \times D_3 \times D_4 \times D_5 = \text{an ordered}$

pair containing all values from  
D<sub>1</sub>, D<sub>2</sub>, D<sub>3</sub>, D<sub>4</sub>, D<sub>5</sub>.



The relation is a subset of cartesian product.

$$\gamma(R) \subseteq D_1 \times D_2 \times D_3 \times D_4 \times D_5.$$

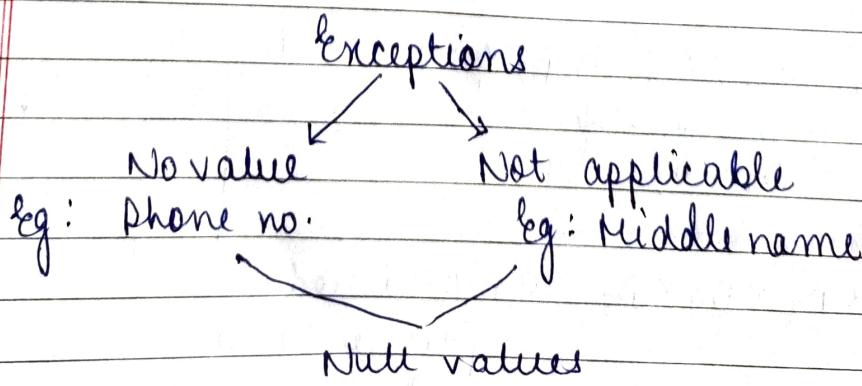
Cardinality:  $|D_1|$

- 6) Current relation state - The no. of elements present in the table at current state.
- 7) Degree of relation  $\rightarrow$  The no. of attributes in table.
- 8) intension  $\rightarrow$  relation schema (heading).
- 9) extension  $\rightarrow$  table itself.

DB tuples, tuple values  
& Null

Relational model - A relation is a set so ordering is something we give.

No 2-tuples would have same value.  
No duplicates are allowed.





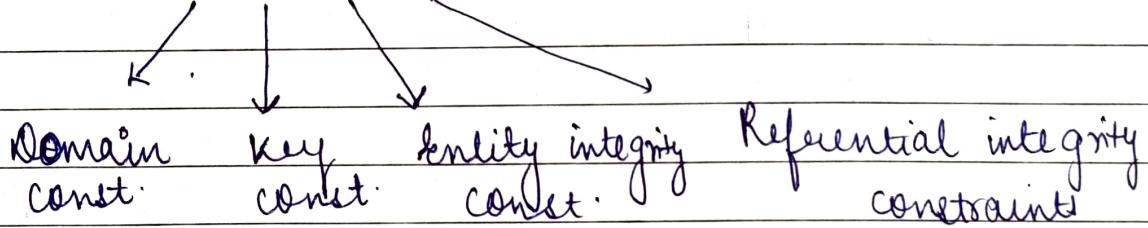
## DB constraints on Relational DB Schema.

### Domain constraints

In the design level we could put up some restriction on data.

Eg: The value should be etc.

### Relational constraints ( Restrictions )



### Domain constraints

SNo	FNO	MNa.	LNa.	Name	→ Not allowed
					✓ Every value should be atomic not divisible.

### Relation - Flat file

No composite or multivalue attributes are allowed into the domain? You can break it into another table.

Entire schema should be atomic.

### Key constraints

No 2 tuples should have the same values.



$t_1 \rightarrow$			
$t_2 \rightarrow$			

$t_1 \neq t_2$ .

Some attributes can have the same value but not all the tuples.

S No	SName	marks.	$\rightarrow$ Superkey
1	Ravi	100	
2	Sakshi	50	
3	Ritu	40	

Super key  $\rightarrow$  A subset of attributes which uniquely identifies a table.

Even after deleting SName or marks we can uniquely identify a table.

Key - minimal superkey is a key.

If in the worst set of all the attributes in a super key is called a key.

\* Any superset of a key is a superkey.

(SNo + SName)  
 $\downarrow$        $\downarrow$   
 Key      Super  
 Key

SK  $\rightarrow$  all the attributes  
 key  $\rightarrow$  minimize the SK

Key  $\rightarrow$  A1

Attributes  $\rightarrow$  A1, A2, A3, A4

$$2 \times 2 \times 2 \times 2 = 8 = 2^3$$

minimal - del till no keys are present



Date: \_\_\_\_\_

How many keys can be there for a relation?  
More than one key.

For ex.

Candidate keys

RN + EN

ON	color	Price	RN	EN

Candidate keys - If we have more than one minimal superkey for a table then it is called a candidate key.

Primary key - One of the key in candidate key which uniquely identifies a table.

$A_1, A_2, A_3, A_4 \rightarrow$  Superkey (No 2 tuples can have the same value)

$(A_2, A_4, A_3) \rightarrow SK$

$(A_3, A_4) \rightarrow SK$  and key (minimal superkey)

$A_1, A_2, A_3, A_4 \xrightarrow{X} SK$

$(A_1, A_2, A_4) \rightarrow SK$  and key primary key

Candidate key =  $((A_3, A_4), (A_1, A_2, A_4))$  choose candidate key with less no of attrb



## DB Entity and referential integrity constraint

### i) Entity integrity

Referential integrity

ENO

↳

There should be any null value.

→ No attribute should have a null value.

### ii) Referential integrity (Btw n<sup>o</sup> 2 or more tables)

Employee FK

Department

ENO	ENa	Dept No	Dept No
1	a	1	1
2	a	x	2
			3
			4

If he  
newly  
joined

Referencing

Referred or base

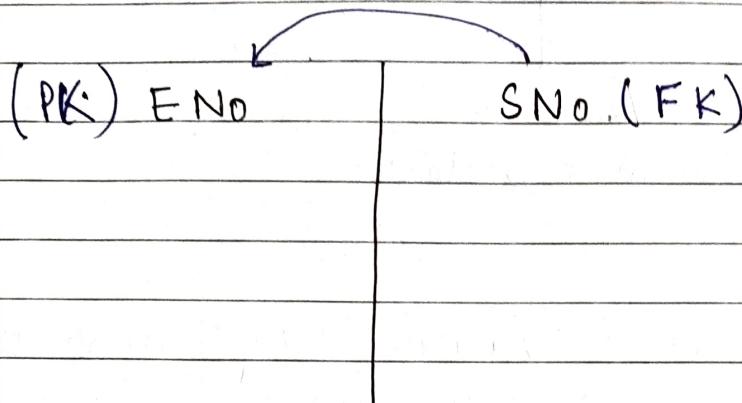
Foreign key → Primary key on another table.

1) Null values are allowed.

2) We can have more than one foreign key



In case of recursive relationships referential integrity exists in the same table.



Primary key

Foreign key

- |   |   |
|---|---|
| <ol style="list-style-type: none"> <li>1) A key which uniquely identifies tuples in a table.</li> <li>2) Null values not allowed</li> <li>3) Only one primary key exists</li> </ol> | <ol style="list-style-type: none"> <li>1) Primary key on another table is called foreign key for the given table.</li> <li>2) Null values allowed</li> <li>3) More than one foreign key can be there</li> </ol> |
|---|---|

DB Action upon constraints violations.

When we modify (insertion, updation & deletion) constraints are violated:

- 1) Insertion



Ename	Eno	Dep.	Supervisor

Domain constraint → violated

key constraint → violated (duplicate values)

~~Entity~~ Entity " " → null values can be inserted.

Referential constraint → value present <sup>not</sup> can't be some value. violated

Can be performed	Domain	key	entity	Referential
Insertion	x	x	x	x
Updation	✓	✓	✓	x
Deletion	✓	✓	✓	x 1) (Reject) 2) (Cascade)

Delete the tuple & the entire tuple from other table

3) (Set Null)  
or some other value



## Counting the number of possible key

1)  $R(A_1, A_2, A_3 \dots A_n)$

$CK = \{A_1\}$ . (smallest superkey possible)

✓

Superset of

$CK$  is a

SK.

$A_1$	$A_2$
1	a
2	a
3	b
4	b

2 2  
1 1

$R(A_1, A_2, A_3)$

$X$   
 $CK = \{A_1\}$

$SK = A_1, (A_1 A_2), (A_1 A_3), (A_1 A_2 A_3)$

$R(\underbrace{A_1}, \underbrace{A_2, A_3 \dots A_n})$

$SK = 2^{n-1}$

$n-2$ .

2)  $R(A_1, A_2, A_3 \dots \underbrace{A_n})$

$CK = \{A_1, A_2\}$

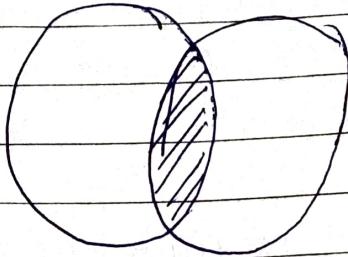
$CK = \{A_1, A_2\} 2^{n-2}$

$CK = \{A_1, A_2 A_3\} = 2^{n-3}$ .

$CK = \{A_1, A_2\}$

Date \_\_\_\_\_ / \_\_\_\_\_ / \_\_\_\_\_

$$CK = A_1 \cup A_2$$



$$SK = SK(A_1) + SK(A_2) - SK(A_1 A_2)$$

$$= 2^{n-1} + 2^{n-1} - 2^{n-2}$$

$$= 2 \times 2^{n-1} - 2^{n-2}$$

3)  $R(A_1, A_2, A_3 \dots A_n)$

$$CK = \{A_1, A_2 A_3\}$$

$$SK(A_1) + SK(A_2 A_3) - SK(A_1 A_2 A_3)$$

$$2^{n-1} + 2^{n-2} - 2^{n-3}$$

$$CK = \{A_1, A_1 A_2\} \rightarrow \text{not possible}$$

$\downarrow$   
 $(SK)$

$$CK = \{A_1, A_2, A_3 A_4\}$$

$$SK(A_1 A_2) + SK(A_3 A_4) - SK(A_1 A_2 A_3 A_4)$$
$$= 2^{n-2} + 2^{n-2} - 2^{n-4}$$



Date \_\_\_\_\_

4)  $R(A_1 A_2 \dots A_n)$

$C_K = \{A_1 A_2, A_1 A_3\}$ .  $\rightarrow$  these is possible because they are not the superset of each other.

$$SK(A_1 A_2) + SK(A_1 A_3) - SK(A_1 A_2 A_3)$$

$$2^{n-2} + 2^{n-2} - 2^{n-3}.$$

5)  $R(A_1 A_2 \dots A_n)$

$$C_K = \{A_1, A_2, A_3\}.$$

$$SK(A_1) + SK(A_2) + SK(A_3) - SK(A_1 A_2) - SK(A_2 A_3) - SK(A_1 A_3) + SK(A_1 A_2 A_3)$$

$$2^{n-1} + 2^{n-1} + 2^{n-1} - 2^{n-2} - 2^{n-2} - 2^{n-2} + 2^{n-3}.$$

6)  $R(A B C D)$

$$C_K(A, BC)$$

find SK ?

$$SK(A) + SK(BC) - S(A BC)$$

$$= 2^{n-1} + 2^{n-2} - 2^{n-3}.$$

$$n=4$$

$$= 2^{4-1} + 2^{4-2} - 2^{4-3}$$

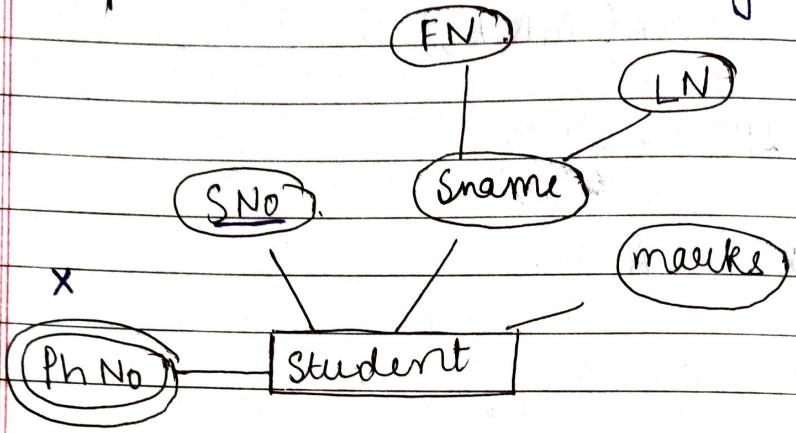
$$= 6 + 4 - 2.$$

$$= 10 \text{ Keys}$$

## Module 3 .

## Conversion of ER model to Relational model :

Step 1: Convert the entity into relations

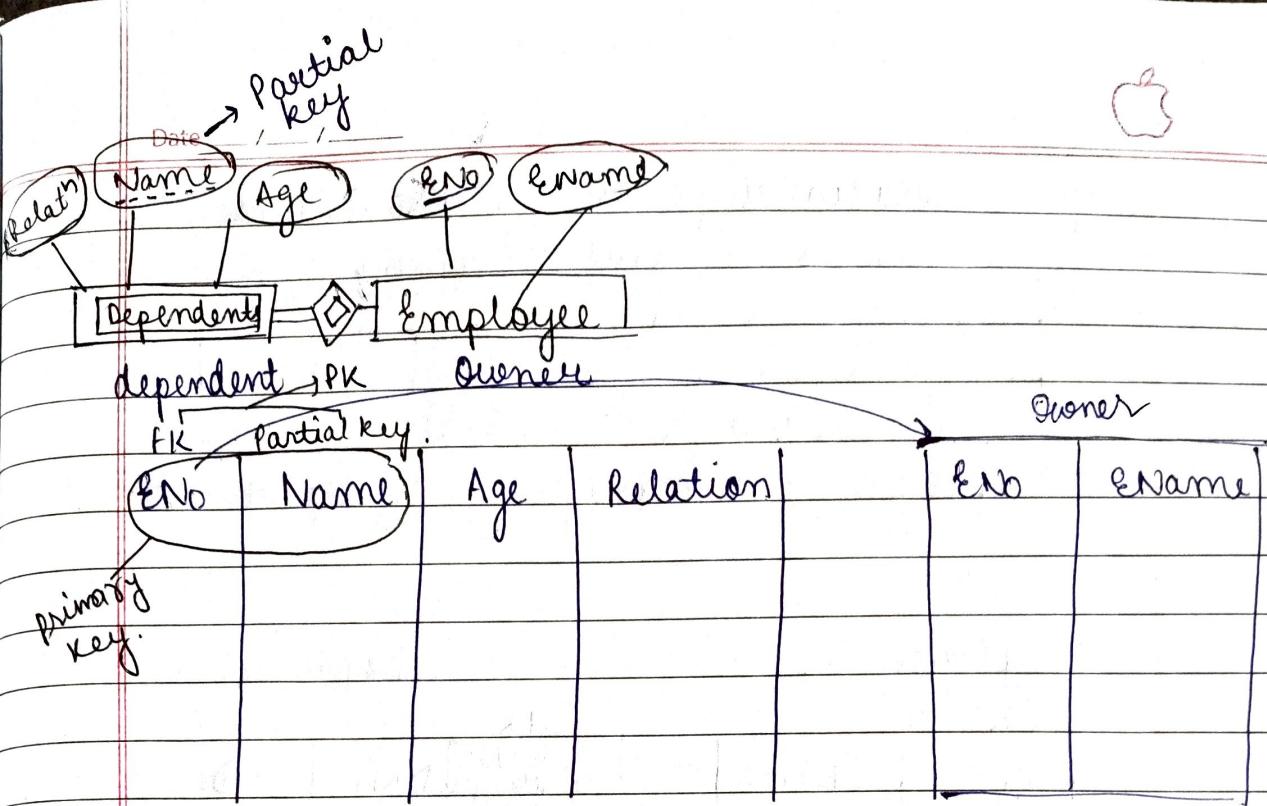


ignore the  
multivalue  
attribute  
in table

Student

- a) Add the simple attribute
  - b) for composite add the simple attribute in the table
  - c) Don't add the multivalue attribute

Step 2:



Once you create the relation add all the primary key of owner entity to dependent with full contribution.

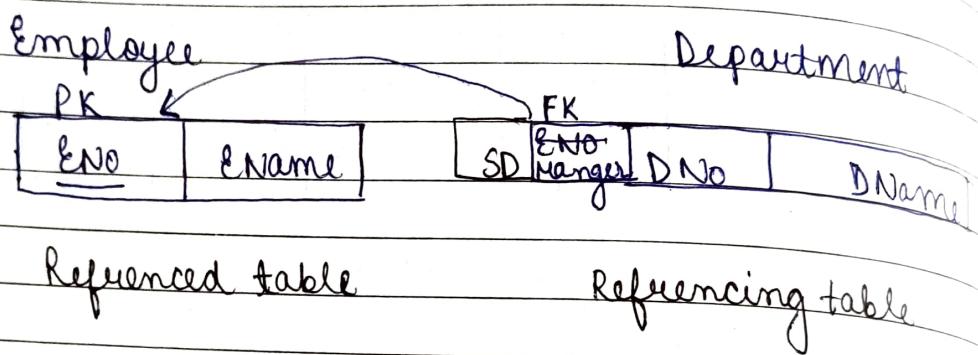
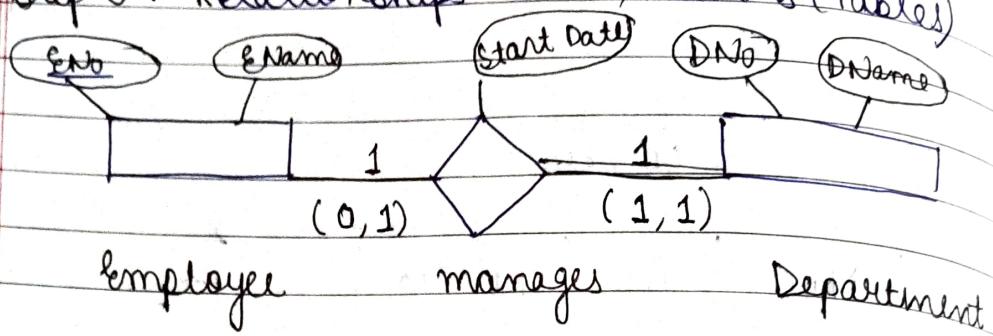
**Partial key** - You will not identify the tuple uniquely but can use for part of a table.

Need not to take care of the identifying rel<sup>n</sup>

Foreign key → weak entity + identifying.

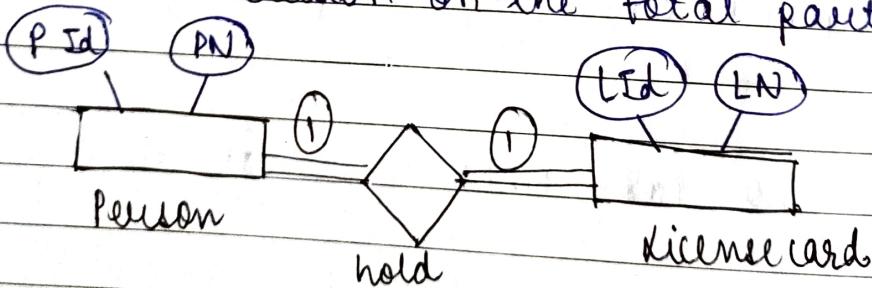
On del or update on the owner side there should be cascading (i.e. del on both the sides) for weak entity.

## Step 3: Relationships to relations (tables)



Take the PK on either side or the total participating side and include it on this side as FK.

\* Do the inclusion on the total participating side.



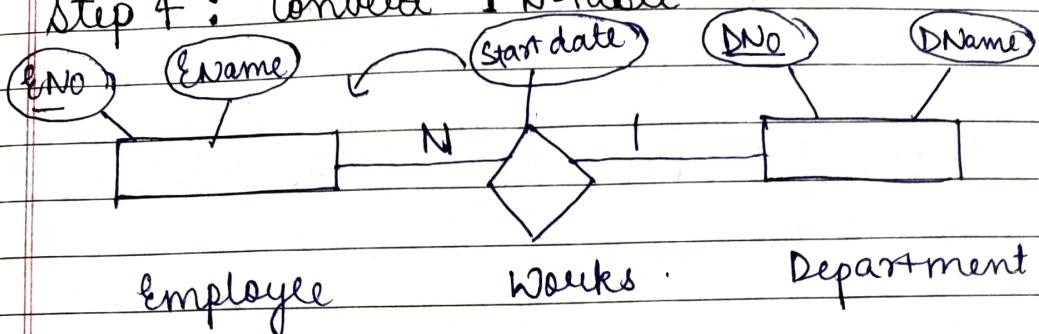
no of entities (person) = no of license card (entity)

Combine both the table.

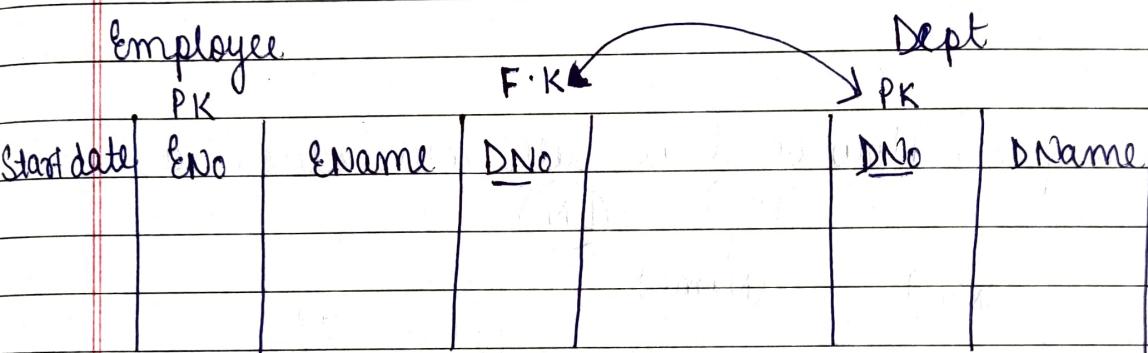
Person	license card		
Pid	PN	Lid	LN



Step 4 : Convert 1 N-table

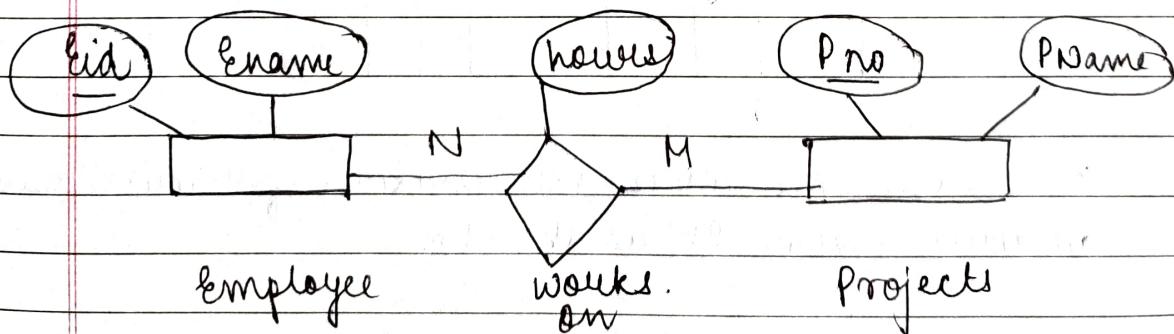


A Dept. can have many employees.

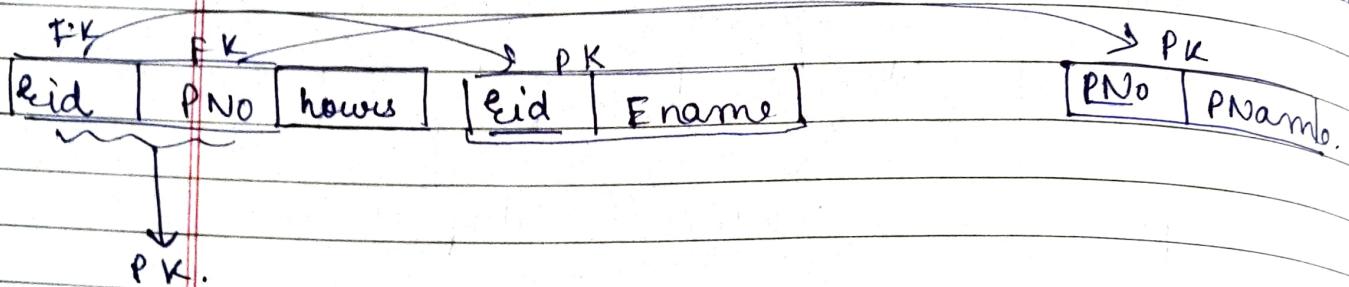


Add the attribute on the one side.

Step 5: Convert Many Many to table



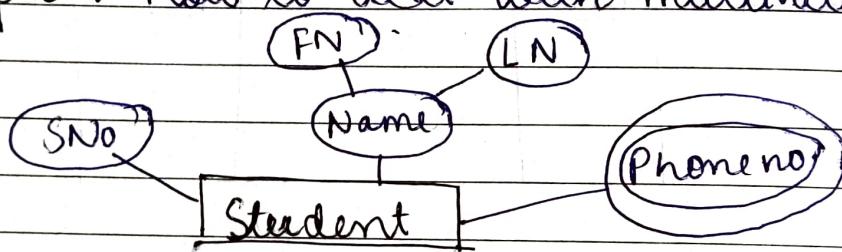
We can't use the foreign key on diffn. tabl.



In case of M to N create a new table.

1 - 1

Step 6 : How to deal with multivalued attribute

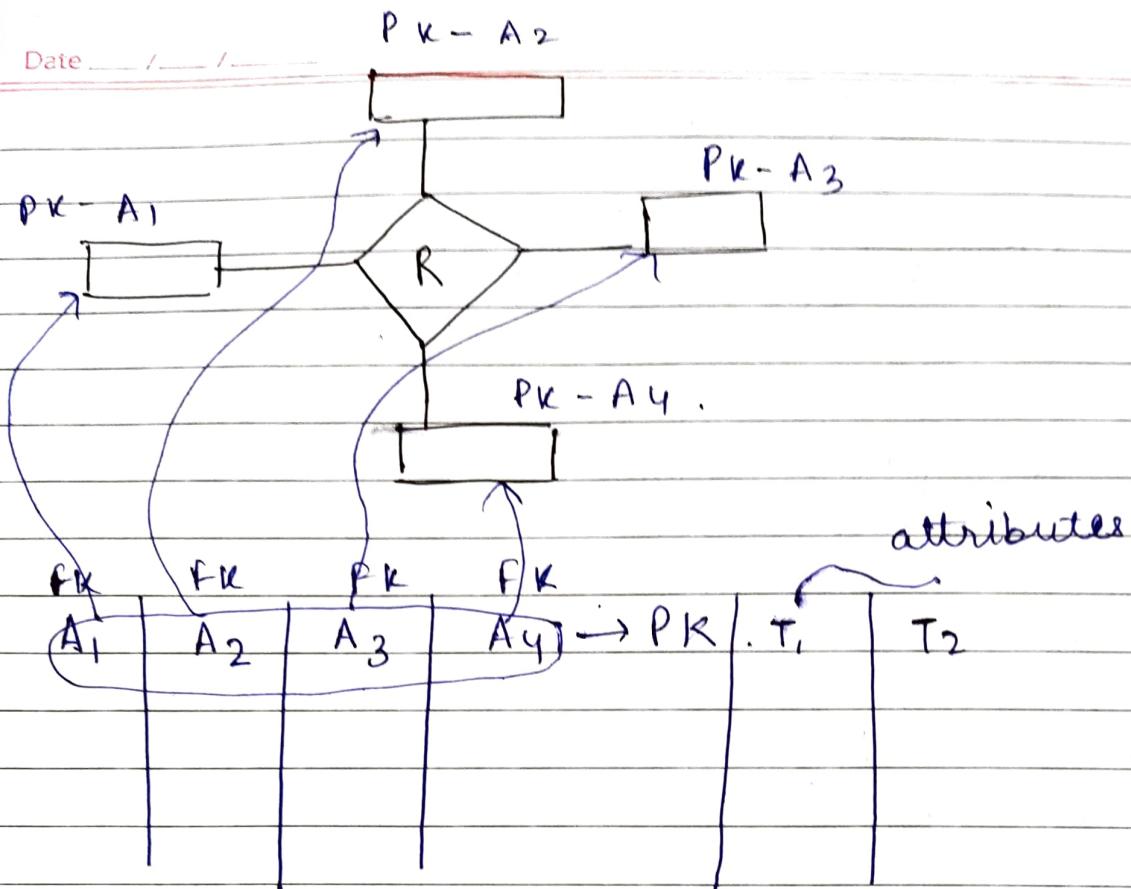


SNo	FN	LN	FK
			comb^n of both attr

SNo	Phoneno	Pk
1	P <sub>1</sub>	
1	P <sub>2</sub>	
1	P <sub>3</sub>	

We create one more new relation for multivalued attribute with PK as the FK.

Step 7 :



ER model

Entity type

1:1 see 1:N

M:N

Nary relationship type

Simple attribute

Composite "

multivalued "

value set

key attribute

Relational model

entity relation

foreign key (or rel<sup>n</sup>)

relationship (rel<sup>n</sup>) + 2 F Ks.

relationship (rel<sup>n</sup>) + n F Ks.

Attribute

Set of simple component  
relation and foreign key

domain

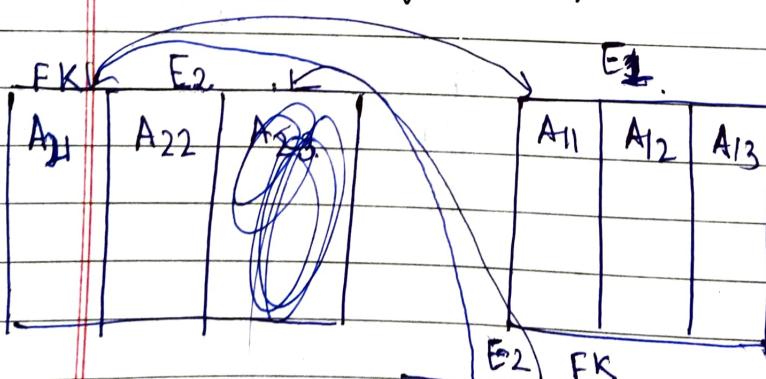
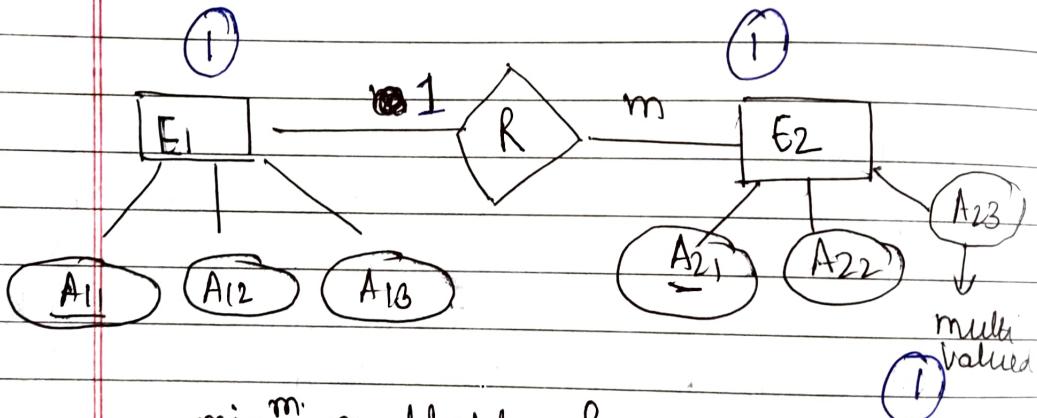
primary key

# Questions



Rent & payment should be added as an attribute to

- a) none
- b) Hotel
- c) Person
- d) Lodging



3 tables

A <sub>21</sub>	A <sub>23</sub>
-----------------	-----------------



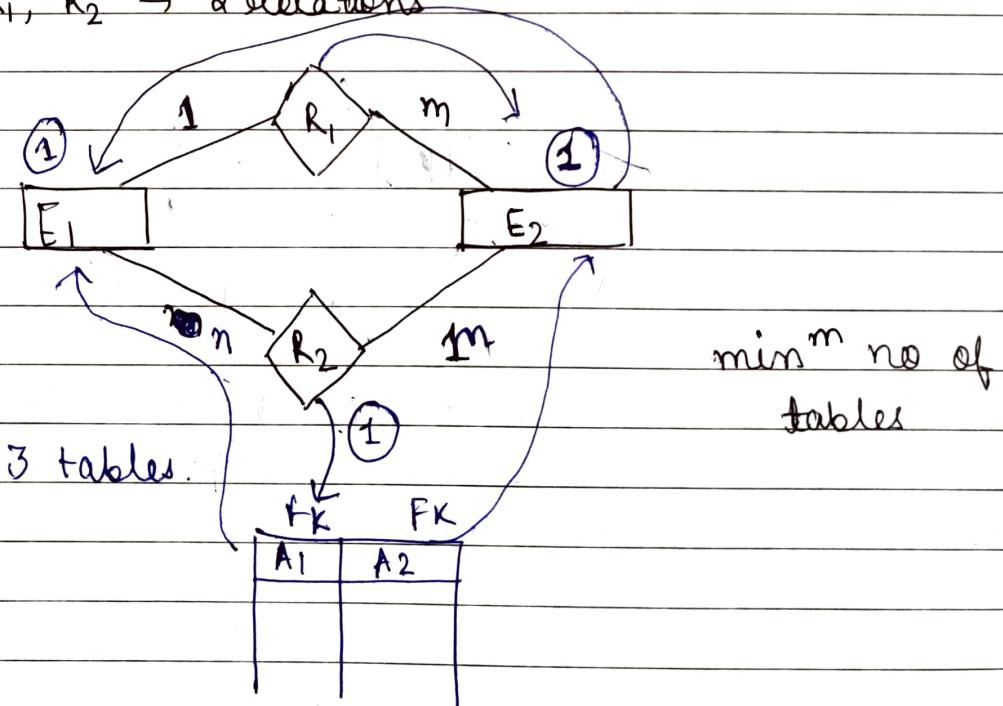
A	C	
2*	4*	(2, 4)
3	4	
4	3	(2, 5)
5*	2*	(2, 7)
7*	2*	(2, 9)
9*	5*	(5, 1)
6	4	

Which tuples should be deleted when (2, 4) is deleted:

(5, 2) (7, 2) & (9, 5)

Q.  $E_1, E_2 \rightarrow 2$  entities

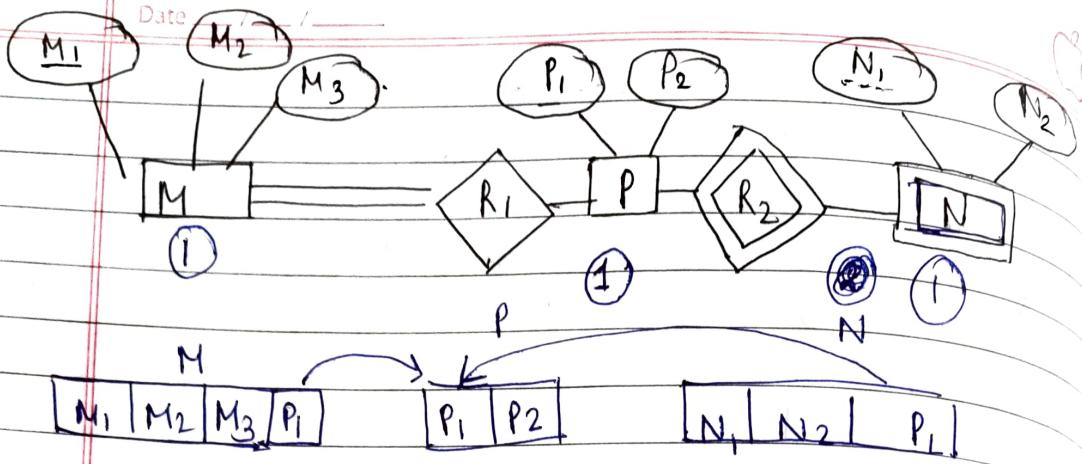
$R_1, R_2 \rightarrow 2$  relations



Here we need a table for  $E_1$ , one for  $E_2$

$R_1$  is represented as FK in  $E_2$

for  $R_2$  we need one table have PK $E_1$  & PK $E_2$ .



3 tables

Q:

Let  $R(a, b, c)$  and  $S(d, e, f)$  be 2 relations "d" is foreign key of  $S$  that refers to the primary key of  $R$ .

'R' & 'S'

R		S		
(PK) a	b	c	(FK)d	e
			f	

which can cause violation of referential integrity

Insert into S <sup>might</sup> cause  
Delete from R



## Normalisation

### Introduction

Procedure of dividing the table into small tables.

Eid	EN	Did	Did	Dname
1	a	1	1	C.S.

Putting everything we have in one big (universal) table:

1) Redundancy is possible

Eid	Did	DN
1	1	C.S
2	1	C.S
3.	1	C.S.

2) anomalies (problems)

1) Insert (same information at many places leads to inconsistency)

2) Deletion

3) Updation or modification

To solve take every table and do splitting of tables called normalisation.

## Introduction to Functional dependencies

A	B	C
1		
(2)	a	b
3		
4		
(2)	a	b

$A \rightarrow BC$

If we knew value of A then we can find values of B & C.

$$t_1(A) = t_2(A)$$

then

$$t_1(BC) = t_2(BC)$$

A	B	
a	1	
a	1	$A \rightarrow B$ .
b	2	A determines B.
b.	2.	B is functionally dependent on A

Create a new table

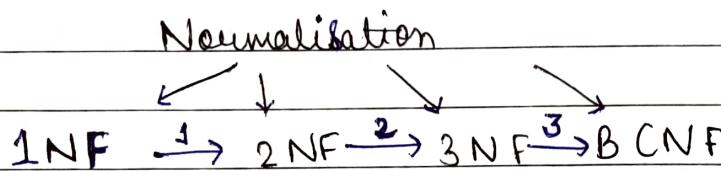
A	B
a	1
b	2

Area to be minimized further.

a	1
a	1
a	1
a	1
a	1

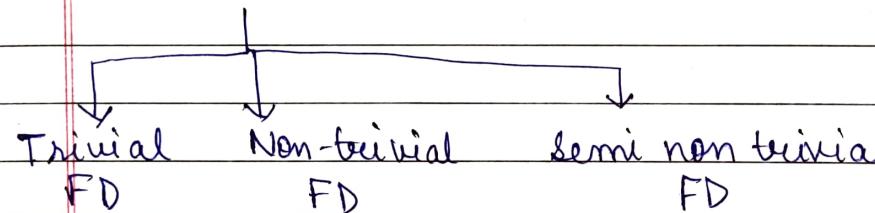


$A \rightarrow B$   
 $\downarrow$   
PK  
for the  
table  
(left)



$X \rightarrow Y$   
 $X$  determines  $Y$ ,  $Y$  is functionally dependent on  $X$

$ABC \rightarrow DEF$



$A \rightarrow A$	$X \cap Y = \emptyset$	$XY \rightarrow YZ$
$AB \rightarrow A$	$A \rightarrow B$	$AB \rightarrow BC$
	$A \rightarrow BC$	

$X \supseteq Y$        $AB \rightarrow CD$       something in common  
 $X \cap Y \neq \emptyset$

Present on the right hand side  
is present on the left hand side  
not present on either side

$$X \cap Y \neq \emptyset$$

Rule out the FD based on the table

Eid	Ename	A	B
1	a	1	1
2	b	1	2
3	b	2	2

(PK)

$Eid \rightarrow Ename \checkmark$   
 $Ename \rightarrow Eid \times$

$A \rightarrow B \times$   
 $B \rightarrow A \times$

$A \rightarrow B$  for a given value of A B should be unique

A	B	C	$A \rightarrow B \times$	$I \rightarrow 1 \times$
1	1	4	$B \rightarrow C \times$	$I \rightarrow 2 \times$
1	2	4	$B \rightarrow A \times$	$I \rightarrow 3 \times$
2	2	3	$C \rightarrow B \times$	
2	4	3	$C \rightarrow A \checkmark$	
			$A \rightarrow C \checkmark$	

PK A B C

1	2	3	$A \rightarrow B \checkmark$
4	2	3	$BC \rightarrow A \times$
5.	3	3.	$B \rightarrow C \checkmark$
			$AC \rightarrow B \checkmark$

A	B	C
1	1	1
1	1	0
2	3	2
2	3	2

3	2(PK)	2	$XZ \rightarrow X \checkmark$	$Y \rightarrow Z$
X	Y	Z	$XY \rightarrow Z \checkmark$	$XZ \rightarrow Y \checkmark$
1	4	3		
1	5	3.	$Z \rightarrow Y \times$	
4	6	3.		



X	Y	Z
1	4	2
1	5	3
1	6	3
2	2	2

- a)  $XY \rightarrow Z$  &  $Z \rightarrow Y$  X
- b)  $YZ \rightarrow X$  &  $Y \rightarrow Z$  ✓
- c)  $YZ \rightarrow X$  &  $X \rightarrow Z$  X
- d)  $XZ \rightarrow Y$  &  $Y \rightarrow X$  X

RCA, B, C)	A	B	C
1.	1	1	1
1.	1	1	0
2.	3	3	2
2.	3	3	2

- 1)  $A \rightarrow B$  &  $B \rightarrow C$  X { we can be sure by what doesn't hold but
- 2)  $A \rightarrow B$  &  $B \not\rightarrow C$  X not by what holds
- 3)  $B \not\rightarrow C$  ✓ from the req. Analysis
- 4) A  $\not\rightarrow B$  and  $B \not\rightarrow C$  X only we can define functional dependency

Formal definition of  
Functional dependency

	A	B	$A \rightarrow B$
$t_1$			
$t_2$			

If  $t_1$  &  $t_2$  agree then they must agree here

If  $t_1$  &  $t_2$  disagree here they may agree or disagree here



	A	B
1	a a	
2		a
3		b

For a given value of A value of B is unique

A	B
1	a

### Various usage of FD

- i) identifying the add'n. FD
- ii) identify the keys
- iii) identifying equivalences of FD
- w) finding minimal FD set

Two methods

inference  
rules

closure set  
of attributes

reflexive:  $A \rightarrow B$  if  $B \subseteq A$

$A \rightarrow B \& B \rightarrow C$

transitive: then  $\bullet A \rightarrow C$

Decompositional  $A \rightarrow BC$  then  
 $A \rightarrow B$  and  $A \rightarrow C$ .

Augmentation:  $A \rightarrow B$  then  
 $A C \rightarrow BC$ .



### Union

If  $A \rightarrow B$  and  $A \rightarrow C$  then  
 $A \rightarrow BC$

### Composition.

If  $A \rightarrow B$  and  $C \rightarrow D$   
then  $AC \rightarrow BD$

Closure set of rules.

Set of attributes that can be functionally determined

FD:  $A \rightarrow B$      $B \rightarrow D$      $C \rightarrow DE$      $CD \rightarrow AB$ . determined from it.

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

In the table having attributes A, B, D

$$\begin{aligned} A^+ &= \{B, D, A\} \\ B^+ &= \{D, B\}. \end{aligned}$$

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

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

$$E^+ = \{E\}.$$

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

$$(AD)^+ = \{A, D, B\} \rightarrow$$



$$A \rightarrow B$$

$$B \rightarrow D$$

$$C \rightarrow DE$$

$$CD \rightarrow AB.$$

$$AB \rightarrow CD$$

$$AF \rightarrow D$$

$$DE \rightarrow F$$

$$C \rightarrow G_1$$

$$F \rightarrow E$$

$$G_1 \rightarrow A$$

$$(CF)^+ = \{ C, F, D, E \} \\ G_1, A$$

$$(BG_1)^+ = \{ B, G_1, D, A, E \}$$

$$(AF)^+ = \{ A, F, B, E, D, E, G_1 \} \quad \{ A, F, D, E \}$$

$$(AB)^+ = \{ A, B, D, C, G_1 \}$$

Determining the candidate key  
 $R(ABCDA)$

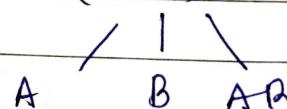
$$A \rightarrow B$$

$$B \rightarrow C$$

$$C \rightarrow D$$

$$D \rightarrow A$$

$$(A B)$$



$$(A B C)$$

A, B, C, AB, BC, CA, ABC

(")

Date

$(A_1, A_2, \dots, A_n)$

Candidate key =  $2^{n-1}$  candidate key

$$R(ABCD) \Rightarrow 2^4 - 1 = 15$$

If no candidate key exists then set of all attributes is candidate key.

ABCD.	①	BCD, ABC, ACD, ABD.	④	
AB,	BC,	CD,	DA, AC, BD	⑥
A,	B,	C,	D	④

$A \rightarrow B$
$B \rightarrow C$
$C \rightarrow D$
$D \rightarrow A$

$$\begin{aligned}
 A^+ &= \{A, B, C, D\} && A \text{ is a candidate key} \\
 B^+ &= \{B, C, D\} \\
 C^+ &= \{A, B, C, D\} \\
 D^+ &= \{A, B, C, D\}.
 \end{aligned}$$

$R = (A, B, C, D, E, F)$

PDS:  $C \rightarrow F$     $E \rightarrow A$     $EC \rightarrow D$     $Key = ?$   
 $A \rightarrow B$

$$CD^+ = \{C, D, F\} \quad (CE)^+ = (\overbrace{AB}^C \overbrace{CD}^E \overbrace{EF}^F)$$

key  $\checkmark$   $EC^+ = \{E, C, D, A, F, B\}$ .

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

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

R = {E, F, G, H, I, J, K, L, M, N}

FD:  $\begin{cases} \{E, F\} \rightarrow \{G\} \\ \{F\} \rightarrow \{J, I\} \\ \{K\} \rightarrow \{M\} \end{cases}$

$\{E, H\} \rightarrow \{K, L\}$

$\{K\} \rightarrow \{M\}$

$\{L\} \rightarrow \{N\}$

$EF \rightarrow G$

$F \rightarrow J, I$

$K \rightarrow M$

$EH \rightarrow K, L$

$K \rightarrow M$

$L \rightarrow N$ .

$(EFH)^+ = (\overline{E}\overline{F}\overline{G}\overline{H})^+ IJKLMN$

2<sup>7</sup> Super Key.  
1 candidate key

R = {A, B, C, D, E, M}

FD:

$A \rightarrow B$        $E \rightarrow C$   
 $B \subset D$        $D \rightarrow A$



$$(EH)^+ = (\check{A} \check{B} \check{C} \check{D} EH)$$

d) AEH, BEH, DEH

$$(EH)^+ = \{C, E, H\}$$

No of SK =

$$b - (AEH)^+ = \{A E H C B D\} \cup (ck)$$

$$b - (BEH)^+ = \{B E H C D A\} \cup (ck)$$

$$(CEH)^+ = \{C \cdot E H\} \times$$

$$b - (DEH)^+ = \{D E H A B D\} \cup (ck)$$

$$SK(k_1 k_2 k_3) = k_1 + k_2 + k_3 - k_1 k_2 - k_2 k_3 - k_1 k_3 + k_1 k_2 k_3$$

Q:

$$R = ABCDEF G_1 H$$

$$F = \{CH \rightarrow G_1, A \rightarrow BC, B \rightarrow CFH, \\ E \rightarrow A, F \rightarrow EG_1\}.$$

$$(\underline{-D})^+ = (\check{A} \check{B} \check{C} \check{D} \check{E} \check{F} G_1 H)$$

$$\times \\ D^+ = \{D\}$$

$$\checkmark (AD)^+ = \{A, D, B, C, E, H, E, G_1\}.$$

$$\checkmark (BD)^+ = \{B, D, C, F, H, E, G_1, A\}. \quad \text{4 keys}$$

$$X(CD)^+ = \{C, D\}$$

$$\checkmark (ED)^+ = \{E, D, A, B, C, F, H, E\}$$

$$\checkmark (FD)^+ = \{F, D, E, G_1, A, B, C, H\}$$

$$X(G_1 D)^+ = \{G_1, D\}$$

$$X(HD)^+ = \{H, D, G_1\}$$

hot CK      CD ,      G<sub>1</sub> D ,      HD.

$$\begin{array}{c} (\text{C } \text{G}_1 \text{ H}) \\ \text{---} \\ (\text{C } \text{G}_1 \text{ D}) \end{array}$$

$$\begin{array}{c} (\text{G}_1 \text{ D } \text{ C})^+ \\ \text{---} \\ (\text{G}_1 \text{ D } \text{ H})^+ \end{array}$$

$$\begin{array}{c} (\text{H D } \text{ C})^+ \\ \text{---} \\ (\text{H D } \text{ G}_1)^+ \end{array}$$

$$\begin{array}{c} (\text{C } \text{D } \text{ G}_1)^+ \\ \text{---} \\ (\text{C } \text{D } \text{ H})^+ \end{array}$$

$$(\text{C } \text{D } \text{ G}_1 \text{ H})^+ = \{ \text{C}, \text{G}_1, \text{H}, \text{D} \}$$

4 keys possible.

Q      R = { A, B, C, D, E }

$$\text{AB} \rightarrow \text{C}$$

$$\text{C} \rightarrow \text{D}$$

$$\text{B} \rightarrow \text{E}$$

$$(\text{A } \text{B} \cdot )^+ = \text{ABCDE}$$

$$\text{CK} \rightarrow (\text{A } \text{B})^+ = \{ \text{A}, \text{B}, \text{E}, \text{C}, \text{D} \}.$$

$$R = \{ \text{A}, \text{B}, \text{C}, \text{D} \}$$

$$\text{FD} = \{ \text{A } \text{B} \rightarrow \text{C } \text{D}, \text{C} \rightarrow \text{A}, \text{D} \rightarrow \text{B} \}$$

$$\text{A}^+ = \text{A}$$

$$\text{B}^+ = \text{B}$$

$$\text{C}^+ = \text{CA}$$

$$\text{D}^+ = \text{DB}$$



$$\checkmark AB^+ = A B C D$$

$$\times AC^+ = A C$$

$$\checkmark AD^+ = A D B \cdot C$$

$$\checkmark BC^+ = B C A \cdot D$$

$$\times BD^+ = BD$$

$$\checkmark CD^+ = C D B A$$

$$\begin{array}{c} \checkmark \\ \times \\ B \\ \downarrow \\ A \end{array} \quad \begin{array}{c} \times \\ \times \\ D \\ \downarrow \\ C \end{array} \quad ( \begin{array}{c} \checkmark \\ \times \\ A \\ \times \\ B \\ \times \\ C \\ \times \\ D \end{array} ) \quad \begin{array}{c} \times \\ \times \\ A \\ \downarrow \\ C \end{array}$$

BD

Keys are AB, AD, BC, CD

$$R = ABCDEF$$

$$FD = \{ AB \rightarrow C, C \rightarrow D, D \rightarrow E, E \rightarrow F \\ F \rightarrow A \}$$

$$(B \cdot)^T = \overbrace{ABCDEF}^{\text{. . . . .}}$$

$$B^+ = \{ B \}.$$

$$\checkmark AB^+ = \{ A, B, C, D, E, F \}.$$

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

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

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

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

5 keys.

Q. R (AB CDEF)

$A \rightarrow BCDEF$

$BC \rightarrow ADEF$

$DEF \rightarrow ABC$

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

$$B^+ = \{x\}$$

$$C^+ = \{x\}$$

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

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

3 possible candidate keys.

Q20 R = (A B C DE)

$A \rightarrow BC$

$CD \rightarrow E$

$B \rightarrow D$

$E \rightarrow A$

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

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

$$XC^+ = \{C\}$$

$$XD^+ = \{D\}$$

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

$$(A, C, D, E) \quad (A, B, D, E)$$

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

4 Keys.



21)  $R(ABCD)$   $AB \rightarrow CD, D \rightarrow A$

what are the CK of sub reln  $R, (BCD)$

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

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

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

$$\times BC^+ = \{B, C\}$$

$$\times CD^+ = \{C, D, A\}$$

$$\checkmark BD^+ = \{D, B, A, C\} \text{ CK}$$

$$BCD^+ = \{\text{Superkey}\}.$$

22)  $R(ABCDEF)$

$$AB \rightarrow C$$

Find CK for  $R, (DEF)$

$$B \rightarrow D$$

$$AD \rightarrow F$$

$$\checkmark D^+ = \{E, D, F\}$$

$$C \rightarrow D$$

$$\checkmark E^+ = \{E, F, D\}$$

$$D \rightarrow E$$

$$\times F^+ = \{F\}.$$

$$E \rightarrow F$$

$$E \rightarrow D$$

$$\times EF^+ = \{E, F, D\} \rightarrow SK$$

$$(DE)^+ \rightarrow SK$$

$$(DF)^+ \rightarrow SK$$

$$(DEF)^+ \rightarrow SK$$

$$CK = D, E \blacksquare$$



## Equivalence of FD.

$F = \{ A \rightarrow C, AC \rightarrow D, E \rightarrow AD, E \rightarrow H \}$ .  
 $G_1 = \{ A \rightarrow CD, E \rightarrow AH \}$ .

$F \supseteq G_1 ?$        $G_1 \supseteq F ?$

If every dependency in  $G_1$  is already applied in  $F$

$$\text{im } F(A^+) = \text{im } G_1(A^+)$$

"                          ↓

$$\{ C, A, D \} = \{ C, D, A \}$$

$F \supseteq G_1$

$$\text{im } F(E^+) = \text{im } G_1(E^+)$$

↓                          ↓

$$\{ E, A, H, C, D \} \quad \{ A, D, E, C, H \}$$

$G_1 \supseteq F$

$$\text{im } G_1(A^+) = \text{im } F(A^+)$$

$\{ C, D \}$                            $\{ C, D \}$

$$\text{im } G_1(AC^+) = \text{im } F(AC^+)$$

$$\text{im } G_1(E^+) = \text{im } F(E^+)$$

$\{ A, H, D \}$                            $\{ AH, D \}$

∴  $G_1 \supseteq F$

Both are equivalent

## Example on equivalence of FD.

$F = \{ A \rightarrow B, B \rightarrow C, C \rightarrow D \}$        $G_1 = \{ A \rightarrow BC, C \rightarrow D \}$

$G_1 \subseteq F$

~~$F \subseteq G_1$~~

~~$G_1 \subseteq G_2$~~

in  $G_1$

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

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

in  $F$

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

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

$G_1 \supseteq F$

~~$F \supseteq G_1$~~

~~$C \supseteq G_1$~~

Each production  
in  $G_1$  in  $F$

in  $F$

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

$$B^+ = \{C\}$$

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

in  $G_1$

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

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

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

$F \supseteq G_1$  we will see g me vo ~~no~~ values hai  
jo  $F$  ko cover kar sakte hai

like

$F \supseteq G_1$

$G_1 \supseteq F$

$$A^+ = \{ABC\}$$

$$C^+ = \{CD\}$$

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

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

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

20.

Check equivalent FD.



1)  $F: \{ A \rightarrow B, AB \rightarrow C, D \rightarrow AC, D \rightarrow E \}$ .  
 $G_1: \{ A \rightarrow BC, D \rightarrow AB \}$ .

2)  $F: \{ A \rightarrow B, B \rightarrow C, C \rightarrow A \}$ .  
 $G_1: \{ A \rightarrow BC, B \rightarrow A, C \rightarrow A \}$ .

For I.

$F \supseteq G_1$

$$A^+ = \{ B, A, C \} \quad \checkmark$$

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

$G_1 \supseteq F \quad \times$

$$A^+ = \{ B, C, A \} \quad \checkmark$$

$$AB^+ = \{ A, B, C \} \quad \checkmark$$

$$D^+ = \{ D, A, B \} \quad \times \text{ no } E$$

not equivalence.

$F \supseteq G_1$

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

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

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

$G_1 \supseteq F$

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

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

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

Equivalent.

## DB - Minimal cover.

Procedure to find minimal set

- 1) Split the FDS such that RHS contain single attribute.

$$A \rightarrow BC \Rightarrow A \rightarrow B \text{ and } A \rightarrow C$$

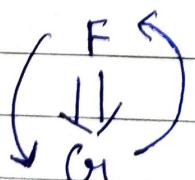
- 2) find the redundant FDS and delete them from the set

Ex :  $\{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$  If closure of  
 $\Rightarrow \{A \rightarrow B, B \rightarrow C\}$   $B \rightarrow D$   $B^+ = \{D\}$  add

- 3) find the redundant attributes on LHS and delete them.

Ex:  $A B \rightarrow C$   $A \rightarrow$  can be deleted if  
 $B^+$  contains 'C'. 'A'.

Minimize  
 $\{A \rightarrow C, AC \rightarrow D, E \rightarrow AD, E \rightarrow H\}$



minimal - you can't del further

- 1)
 
$$\begin{aligned} A &\rightarrow C \\ AC &\rightarrow D \\ E &\rightarrow AD \\ E &\rightarrow H \end{aligned}$$

- $$\begin{aligned} A &\rightarrow C \\ AC &\rightarrow D \\ E &\rightarrow A \\ E &\rightarrow D \\ E &\rightarrow H \end{aligned}$$



2)  $A^+ = \{A\}$   
 $AC^+ = \{A, C\}$   
 $E^+ = \{D, E, H\}$ .

$E \rightarrow D$      $E^+ = \{A, H, E\}$ . ✓

$$\begin{array}{l} A \rightarrow C \\ AC \rightarrow D \end{array}$$

$$\begin{array}{l} E \rightarrow A \\ E \rightarrow D \cdot x \\ E \rightarrow H \end{array}$$

$E \rightarrow H$      $E^+ = \{A, C, E, H\}$ .

D\*

$$A \rightarrow C \quad A^+ = \{A\}.$$

$$AC \rightarrow D \quad AC^+ = \{A, C\}$$

$$E \rightarrow A \quad E^+ = \{D, H, E\}$$

2) Delete  $E \rightarrow D$

$$E \rightarrow D \quad E^+ = \{A, H, C, D, E\} \checkmark$$

$$E \rightarrow H \quad E^+ = \{A, D, C, E\}$$

$A \rightarrow C$ $E \rightarrow AD$ $E \rightarrow H$ .	$\boxed{AC \rightarrow D}$ $C^+ = \{C\}$ $A^+ = \{A, C\}$ . $A^+ = \{C, A\}$ . $C^+ = \{C\}$
--	--

So we will delete 'C'

$$A \rightarrow C$$

$$A \rightarrow CD$$

$$A \rightarrow D$$

$$E \rightarrow AH$$

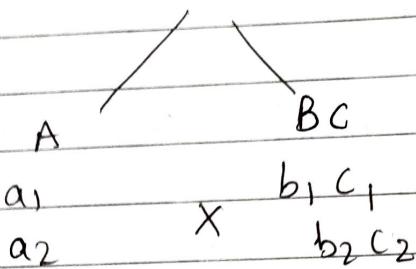
$$E \rightarrow A$$

~~A → AD~~

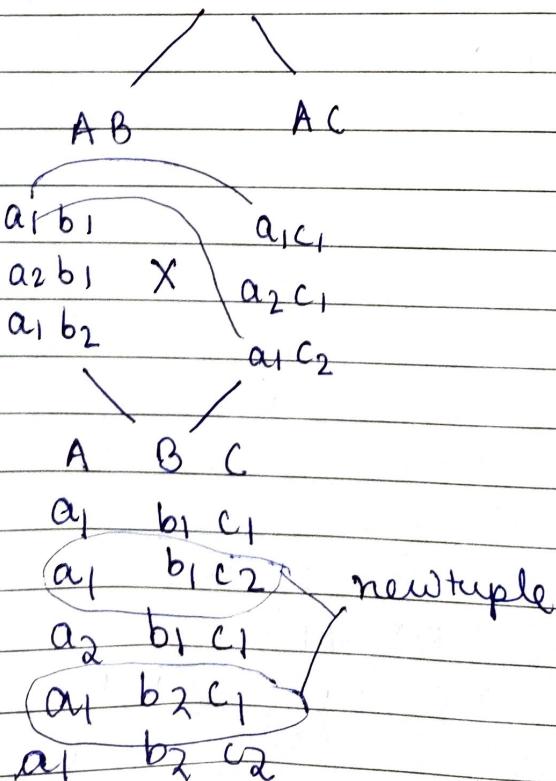
$$E \rightarrow H$$

# Lossless decomposition

	A	B	C
a <sub>1</sub>	b <sub>1</sub>	c <sub>1</sub>	
a <sub>2</sub>	b <sub>1</sub>	c <sub>1</sub>	
a <sub>1</sub>	b <sub>2</sub>	c <sub>2</sub>	



	A	B	C
a <sub>1</sub>	b <sub>1</sub>	c <sub>1</sub>	
a <sub>1</sub>	b <sub>2</sub>	c <sub>2</sub>	
a <sub>2</sub>	b <sub>1</sub>	c <sub>1</sub>	
a <sub>2</sub>	b <sub>2</sub>	c <sub>2</sub>	new tuple





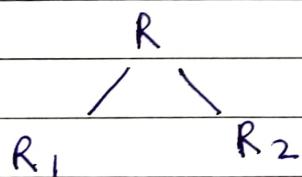
Some times extra information is added  
is called lossy decomposition

$A \cup B$	$B \cup C$
$a_1 b_1$	$b_1 c_1$
$a_2 b_1$	$b_2 c_2$
$a_1 b_2$	

extra tuple  $\rightarrow$  data is lost

Whenever we combine the table we put one or more attribute common to avoid extra data.  
So we decompose it along the CK.

If the common attribute is CK then the decomposition is not lossy.



lossless : while decomposing when a common attribute is CK or key attribute

$$(R_1 \cap R_2) \rightarrow (R_1 \text{ or } R_2 - R_2)$$

$$(R_1 \cap R_2) \rightarrow (R_2 \text{ or } R_2 - R_1)$$

should  
be a key

FD preserving

$$A \Rightarrow G$$

$$R \xrightarrow{(A \cup B \cup C \cup D)} A_1, A_2, A_3 \dots A_n$$

$$F \rightarrow F^+ \text{ (set of all FD that can be applied on } R)$$

$$F_1 \subseteq F^+$$

$$R_1 \quad (A, B)$$

$$R_2 \quad F_2 \subseteq F^+$$

$$(F_1 \cup F_2)^+ = F^+ \text{ dependency preservation}$$

- i) Original fun. D = later FD
- ii) There should be no data loss.

### DB decomposition example

$R(A, BC)$       FD :  $\{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$

$R_1(AB)$        $R_2(BC)$

$R_1$	$R_2$	
<u>A</u> <u>B</u>	<u>B</u> <u>C</u>	lossless +
✓ $A \rightarrow B$	✓ $B \rightarrow C$	preserved
✓ $B \rightarrow A$	✓ $C \rightarrow B$	
$\begin{cases} A & B \rightarrow B \\ A & B \rightarrow A \\ A & B \rightarrow AB \\ AB \rightarrow \emptyset \end{cases}$ not imp.	$B^+ = \{C, A, B\}$ $C^+ = \{A, B, C\}$	✓

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

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

$$R_1 \cup R_2 = R$$

Q.  $R(ABCD)$

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

$$D = (AB, BC, CD)$$



```

graph TD
    AB[AB] --> A[A]
    AB --> B[B]
    B --> BC[BC]
    B --> C[C]
    C --> CD[CD]
    C --> D[D]
    D --> CD[CD]
    D --> ABCD[ABCD]
    BCD[BCD]

```

It is lossless as  $C^T = \{D, A, B\}^{AB, CD}$

It is lossless

lossless - one key in common  
preserving - FD tuples all derived are present in set

$$R(A \rightarrow B \rightarrow D) \\ F = \{ AB \rightarrow CD, D \rightarrow A \} \\ D = \{ AD, B \rightarrow CD \}$$

$AD$ $A \rightarrow D$ $D \rightarrow A$	$BCD$ $B \rightarrow C \rightarrow D$
--	--

$$\begin{aligned} A^+ &= \{A\} \\ D^+ &= \{D, A\} \\ B^+ &= \{B\} \\ C^+ &= \{C\} \\ D^+ &= \{D, A\} \end{aligned}$$

$$AB^T = \{A, B, C, D\}$$

A (D) B C (D)

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

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

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

$$D^+ = \{ D, A \}.$$

$$Bc^+ = \{ B c \}$$

$$CDT = \{ CDA \}$$

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

$AD$ $A \rightarrow D \times$ $D \rightarrow A \checkmark$  <span style="border: 1px solid black; border-radius: 50%; padding: 5px;"><math>\rightarrow A</math></span>	$BCD$ $B \rightarrow B \times$ <del><math>B \rightarrow D</math></del> $C \rightarrow A$ $C \rightarrow C \times$ $D \rightarrow A \times$ $D \rightarrow D \times$ $B \leftarrow C \leftarrow C \times$
$\downarrow$ not preserving	$\downarrow$ <span style="border: 1px solid black; border-radius: 50%; padding: 5px;"><math>BD \rightarrow</math></span>

$D \rightarrow A'$

It is not preserving

Q. R (A B C D E G<sub>1</sub>)

F: A B → C, A C → B, AD → E,  
B → D, BC → A, E → G<sub>1</sub>.

D: (ABC, ABDE, EG<sub>1</sub>)

- 1) First see all the attributes are present on the decomposed table or not.

Attribute = A B C D E G<sub>1</sub> ✓ } Getting the  
D = A B C D E G<sub>1</sub> ✓ } designatable

- 2) Lossless → find the candidate key.

(ABC), (AB) DE, EG<sub>1</sub>

If AB<sup>+</sup> contains C or DE it's the PK for the table.

$$AB^+ = \{A, B, D, E, G_1, C\}$$

AB can be the candidate key if one of them

ABC	ABDE	EG <sub>1</sub>
ABCDEF	EG <sub>1</sub>	



$$E^+ = \{ ., E, G_1 \}$$

$E$  can be the candidate key for one of the tables.

∴ It is lossless decomposition.

### 3) Function preserving

Only non-trivial dependencies should be taken into account.

	ABC	A BDE	EG <sub>1</sub>
rivial	$A \rightarrow A$ $B \rightarrow D$ $C \rightarrow C$	<del><math>A \rightarrow A</math></del> $B \rightarrow D$ $E \rightarrow G_1$ $AB \rightarrow DE$ $AD \rightarrow$	$G_1$
1 → no dependencies found			
	$AC^+ = \{ AC, B, D \}$ $BC^+ = \{ BC, DA \}$ $A^+ = \{ A \}$ $B^+ = \{ B, D \}$ $C^+ = \{ C \}$ $AB^+ = \{ ABC, DEG_1 \}$	$EG_1$ $B \rightarrow D$ $E \rightarrow G_1$ $AB \rightarrow DE$ $AD \rightarrow$	
2 → 3 dependencies		$AD^+ = \{ ADE, G_1 \}$	

DB IN Form. (all the attribute should be atomic)

- 1) We will go for normalisation in order to remove redundancy
- 2) In INF attribute values in a table should be atomic, i.e neither multivalued attributes nor composite attributes is allowed.
- 3) Every table is in INF since the formal definition of a reln guarantee that the attribute values should be atomic (not division).

### Multivalued

			approach <sup>1</sup>	ENo	Ename	Phone
ENo	Ename	Phone		1	a	1234
1	a	1234		1	a	5676
		5678		2	b	1212
				2	b	1342

Redundant data

			approach <sup>2</sup>	ENo	Ename	Phone
ENo	Ename	Phone		1	a	1234
2	b	1212		2	b	1342
		1842		.	.	.

ENo	Ename	Phone
1	a	1234
2	b	1342



## Composite attribute

ENo	Ename		ENo	FN <sub>1</sub>	FN <sub>2</sub>
	FN <sub>1</sub>	FN <sub>2</sub>	=)		

nested attributes  $\rightarrow$  composite + multivalued.

To reduce redundancy  $\rightarrow$  normalization

DB - Second normal  
form

- 1) In 2NF, we don't allow any partial dependencies.
- 2) A relation schema R is in 2NF, if every non prime attribute "A" in R is not partially dependent on any key of R.
- 3) A functional dependency  $X \rightarrow Y$  is a partial dependency, if some attribute "A"  $\in X$  can be removed from X and the dependency still holds.

CK.

A	B	C	$AB \rightarrow C$	$AB^+ = ABC$	$ABC$
a <sub>1</sub>	b <sub>1</sub>	c <sub>1</sub>	$\{B \rightarrow C\}$	Key = AB.	b <sub>1</sub> c <sub>1</sub>
a <sub>2</sub>	b <sub>2</sub>	c <sub>2</sub>	part of $B \rightarrow C$	NK = C.	b <sub>2</sub> c <sub>2</sub>
a <sub>3</sub>	b <sub>3</sub>	c <sub>3</sub>	Redundancy		b <sub>3</sub> c <sub>3</sub>
a <sub>4</sub>	b <sub>3</sub>	c <sub>3</sub>	b <sub>3</sub> c <sub>3</sub> .		a <sub>4</sub> b <sub>3</sub>
a <sub>5</sub>	b <sub>3</sub>	c <sub>3</sub>			a <sub>5</sub> b <sub>2</sub>
a <sub>6</sub>	b <sub>3</sub>	c <sub>3</sub>			a <sub>6</sub> b <sub>3</sub>

## Third normal form

Even after making the table in 2NF it contains some dependencies or redundancies which need to be removed.

No non key attribute should define other non key attribute.

It is preserving as well as lossless decom.

A	B	C	$A \rightarrow B$	$A^+ = ABC$
1	b	x	$B \rightarrow C$	
2	b	x		
3	b	x		
4	c	y		K NK
5	c	y		$A \rightarrow B$
6	c	y		NK NK
				$B \rightarrow C$

Yearly  
dependence

B	C
b	x
c	y

$$\begin{array}{c} A B C \\ \diagdown \quad \diagup \\ A B \quad B C \\ A \rightarrow B \quad B \rightarrow C \\ B \rightarrow A \end{array}$$

$$ABA BC = f$$

it is less

it is more

Ex1: R(A B C D E), F: {AB → C, B → D, D → E}

Examine whether in 2NF or not

i) Find CK.

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

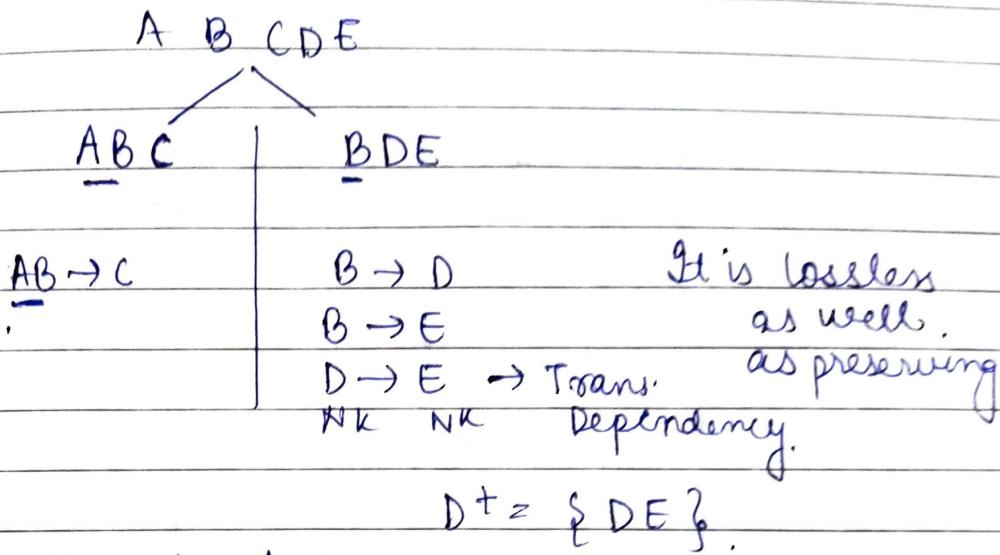
$$CK = AB.$$

$$\begin{array}{l} AB \rightarrow C \\ FD \rightarrow NK \end{array}$$

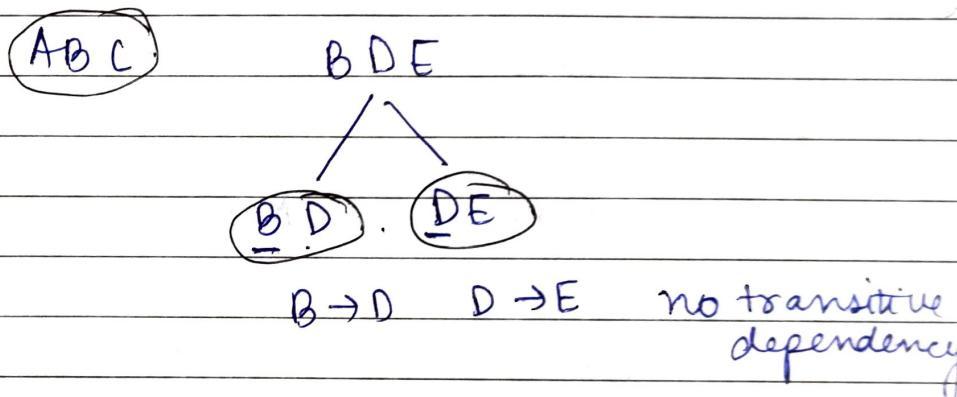
$$B \rightarrow D$$

$$\begin{array}{l} D \rightarrow E \\ PD \rightarrow NK : NK \quad NK \end{array}$$

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



For 3<sup>rd</sup> Normal form.



So, ABC, BD and DE are the tables.

Q. R(ABC)      f: {AB → C, C → A}

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

✓  $AB^+ = \{A, B, C\}$ .       $BC^+ = \{B, C, A\}$ .  
 ✗  $AC^+ = \{A, C\}$

$BC^+$  &  $AB^+$  is the candidate key

Date / /

BC as well as AB.

$$AB \rightarrow C$$

~~CK~~ ~~PD~~

$$C \rightarrow A$$

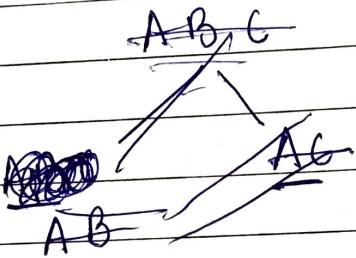
$$PD \rightarrow \cancel{PD}$$

not violating  
2NF

$$AB^+ = \{A, B, C\}.$$

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

since all A B C are  
prime attributes.  
so it is in 2NF



$$A \rightarrow B \quad A \rightarrow C$$

no NK = NK exists

CK

It is in 3NF

(B) — NK.

Q. R(A B C D E F G H I J)

$$AB \rightarrow C$$

$$BD \rightarrow EF$$

$$AD \rightarrow GH$$

$$A \rightarrow I$$

$$H \rightarrow J.$$

$$(ABD)^+ = \{A, B, D, C, E, F, G, H, I, J\}$$



ABD is the CK.

$$\begin{array}{l} AB \rightarrow C \\ PD \quad NK \end{array}$$

$$\begin{array}{l} BD \rightarrow EF \\ PD \quad NK \end{array}$$

$$\begin{array}{l} AD \rightarrow GH \\ PD \quad NK \end{array}$$

$$\begin{array}{l} A \rightarrow I \\ PD \quad NK \end{array} \quad \begin{array}{l} H \rightarrow J \\ NK \quad NK \end{array}$$

$$H^+ = \{ H, J \}$$

$$AB^+ = \{ A, B, C, I \}$$

$$BD^+ = \{ B, D, E, F \}$$

$$AD^+ = \{ A, D, G, H, J, I \}$$

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

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

$$AB \quad C \quad D \quad E \quad F \quad G \quad H \quad I \quad J$$

$$\underline{AB} \quad CI$$

$$\underline{BD} \quad EF$$

$$\underline{AD} \quad GH \quad JI$$

$$\underline{AI} \quad \underline{ABD}$$

$$AB \xrightarrow{NK} C$$

$$\underline{BD} \rightarrow EF$$

$$AD \rightarrow GH \quad A \rightarrow I$$

$$AB \xrightarrow{NK} I$$

$$PD \quad NK$$

$$H \rightarrow J \quad NK - NK$$

$$\underline{AI}$$

$$\underline{ABC}$$

$$A \rightarrow I$$

$$\underline{AB} \rightarrow C$$

$$\cancel{\underline{BD}} \quad \cancel{\underline{DEF}}$$

$$\cancel{B \rightarrow S}$$

$$D \rightarrow EF$$

$$A \rightarrow I$$

$$PD$$

$$HJ$$

$$\underline{ADGI}$$

$$AD \rightarrow GI$$

$$A \rightarrow I$$

$$AI, ABC, \cancel{B \rightarrow S}, HJ, \cancel{ADGI}, ABD$$

$$\begin{array}{l} \cancel{A \rightarrow I} \quad \cancel{D \rightarrow G} \\ \hline \end{array}$$

Q. Consider the relation for published book.

Bookl  $b_1^1$ ,  $b_n^2$ ,  $B_{t_y}^3$ ,  $\{P\}^4$ ,  $A_a^5$ ,  $P^6$ .

$b_t \rightarrow P, B_{t_y}$

$B_{t_y} \rightarrow LP$ .

$A_n \rightarrow A_a$ .

R:  $(A_1, A_2, A_3, A_4, A_5, A_6)$

PD:  $A_1 \rightarrow A_6, A_3$ .

$A_3 \rightarrow A_4$ .

$A_2 \rightarrow A_5$ .

$(A_1 A_2)^+ = \{A_1, A_2, A_6, A_3, A_4, A_5\}$

$C_K = A_1 A_2$ .

$A_1 \rightarrow A_6, A_3$ .  
Pd.

$A_3 \rightarrow A_4$ .  
NK

$A_2 \rightarrow A_5$   
Pd

$A_1^+ = \{A_1, A_6, A_3, A_4\}$

$A_2^+ = \{A_2, A_5\}$ .



A<sub>1</sub> A<sub>2</sub> A<sub>3</sub> A<sub>4</sub> A<sub>5</sub> A<sub>6</sub>.

A<sub>1</sub> A<sub>6</sub> A<sub>3</sub> A<sub>4</sub>

A<sub>2</sub> A<sub>5</sub>

A<sub>1</sub> A<sub>2</sub>

A<sub>1</sub> → A<sub>6</sub> A<sub>3</sub>

A<sub>2</sub> → A<sub>5</sub>

A<sub>1</sub> A<sub>2</sub>

A<sub>3</sub> → A<sub>4</sub>

K      NK

A<sub>1</sub> → A<sub>6</sub> A<sub>3</sub>

K      NK

A<sub>3</sub> → A<sub>4</sub>

NK    NK

A<sub>3</sub> + = { A<sub>3</sub>, A<sub>4</sub> }

(P)    NP ✓  
 NP (P) ✓  
 P      P ✓  
 NL    NP

A<sub>1</sub> A<sub>6</sub> A<sub>3</sub> A<sub>4</sub>

A<sub>1</sub> A<sub>6</sub> A<sub>3</sub>    A<sub>3</sub> A<sub>4</sub>

A<sub>1</sub> → A<sub>6</sub> A<sub>3</sub>.    A<sub>3</sub> → A<sub>4</sub>

A<sub>1</sub> A<sub>6</sub> A<sub>3</sub>, A<sub>3</sub> A<sub>4</sub>, A<sub>2</sub> A<sub>5</sub>, A<sub>1</sub> A<sub>2</sub>.

3NF formal definition (directly in 3NF)

A relational schema 'R' is in 3NF only if in every non trivial FD  $X \rightarrow Y$  either.

1) X is a superkey (or)

2) Y is a prime attribute

## DB - BCNF introduction

A relational schema 'R' is in BCNF<sub>if</sub> whenever a non-trivial FD  $X \rightarrow A$  holds in R, then X is a superkey of R i.e. determinants of all FD must be a superkey.

Ex R(ABC) + {AB  $\rightarrow$  C, C  $\rightarrow$  B}.

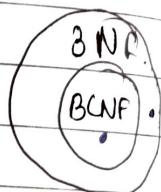
A	B	C
0	1	1
1	1	2
1	0	3
0	0	4
4	1	1
5	1	1
6	1	1
7	1	1

In 3NF  $X \rightarrow A$ .

- ① X is a superkey of R
- ② A is prime w.r.t X

All BCNF  $\rightarrow$  3NF

All 3NF  $\nrightarrow$  BCNF



In BCNF

AB  $\rightarrow$  C  
SK

(C  $\rightarrow$  B)  
PD

Not in BCNF  
as C has PD.

- ① X is a superkey  
only one condition shall  
be satisfied

AC & AB CK  
 $C^+ = \{CB\}$

ABC

AC

CB

C  $\rightarrow$  B

CK

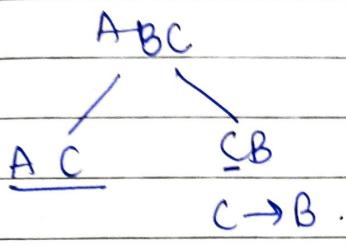
C is the CK

0	1
1	2
1	3

1	1
2	1
3	0
4	0



Since FD.



we will never get FD  $AB \rightarrow C$  so we will lose the FD. fun. Dependency is not preserved.

BCNF

- 1) It is lossless
- 2) It may or may not be FD preserving.

Q. Given  $R(A B C D E F G H I J)$  and  $F: \{AB \rightarrow C, A \rightarrow DE, B \rightarrow F, F \rightarrow GH, D \rightarrow IJ\}$   
decompose in BCNF.

$$AB^+ = \{A, B, C, D, E, F, G, H, I, J\}.$$

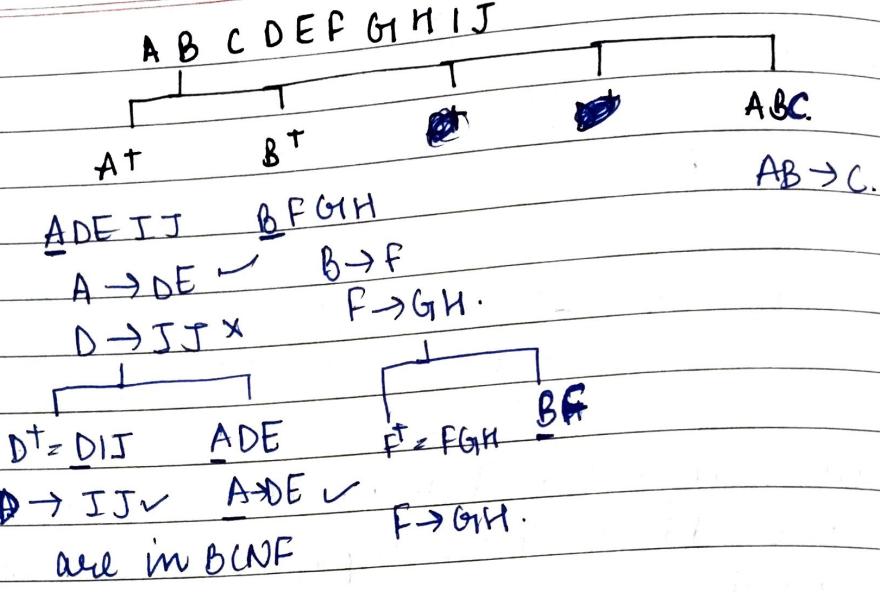
So, AB is the CK.

$AB \rightarrow C$	$A \rightarrow DE$	$B \rightarrow F$	$F \rightarrow GH$	$D \rightarrow IJ$
CK	PD	PD	NP	NP.

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

$$B^+ = \{ B, F, G, H \}$$

Date \_\_\_\_\_ / \_\_\_\_\_ / \_\_\_\_\_



DIJ, ADE

Q. R (ABCDEF<sup>GHIJ</sup>) and F: {AB → C, B → D, D → EF  
A → GH, H → IJ}

$$AB^+ = \{ A, B, C, D, E, F, G, H, I, J \}$$

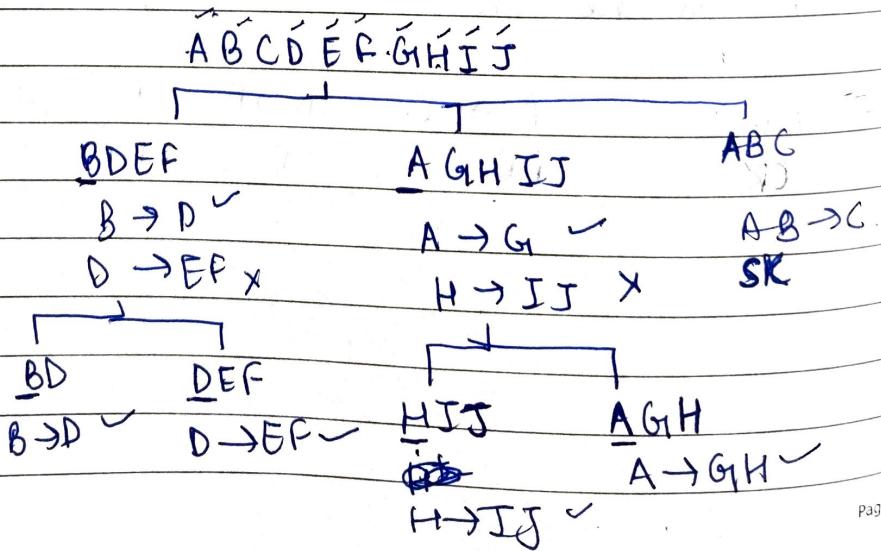
$$\begin{array}{lllll} AB \rightarrow C & B \rightarrow D & D \rightarrow EF & A \rightarrow GH & H \rightarrow IJ \\ CK & PD & NP & PD & NP \end{array}$$

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

$$D^+ = \{ D, E, F \}$$

$$H^+ = \{ H, I, J \}$$

$$A^+ = \{ A, G, H, I, J \}$$





BD, DEF, HIJ, AGH, ABC

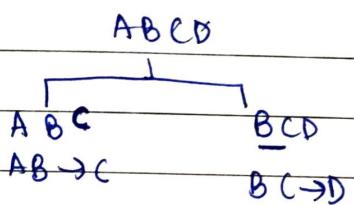
Q. R(ABCD) F: { AB → C, BC → D }.  
Decompose it to BCNF.

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

AB is the CR.

$$\begin{array}{ll} AB \rightarrow C & BC \rightarrow D \\ CK & NP \end{array}$$

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



So, tables are ABC and BCD

Q. R(ABDLPT) and F: { B → PT, T → L, A → D }.  
Decompose R into BCNF

$$AB^+ = \{ A, B, D, P, T, L \}$$

$$CK = AB$$

$$\begin{array}{l} B \rightarrow PT \\ PD \end{array}$$

$$\begin{array}{l} T \rightarrow L \\ NP \end{array}$$

$$\begin{array}{l} A \rightarrow D \\ PD \end{array}$$

A B D L P T A

B<sup>+</sup> = PTBL

A B D

B<sup>+</sup> = BPTL

B → PT ✓

T → L X

TL

BPT

T → L

B → PT

A → D

AD AB.

A → D ✓

Tables are TL, BPT, AD, AB.

A B D L P T A

B<sup>+</sup> = PTBL

A B D

~~don't care~~

B<sup>+</sup> = BPTL

B → PT ✓

T → L X

A → D

AD    AB  
A → D ✓

TL

BPT

T → L

B → PT

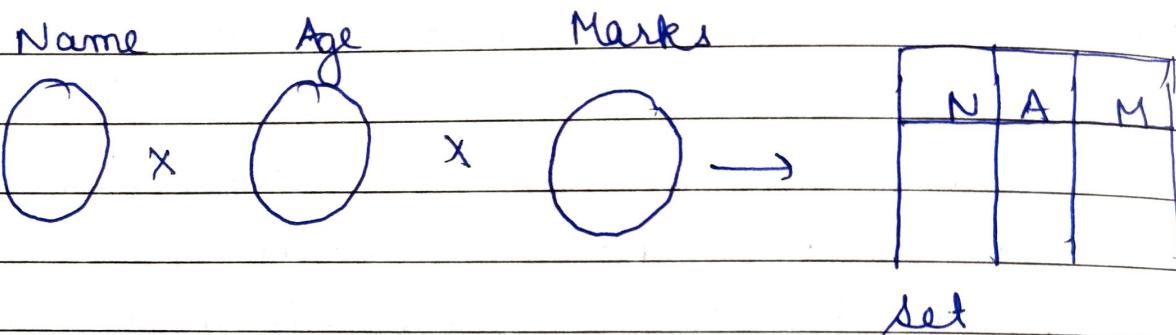
tables are TL, BPT, AD, AB.

~~Notes~~

# Relational Algebra .

implement "RDBMS - SQL

↑ At conceptual level we look at every table as a subset of cross product of all its attributes.



"Algo relational model: basic for RDBMS

Relational + Relational  
Algebra      calculus.

what we  
want & how  
we want

what we  
want

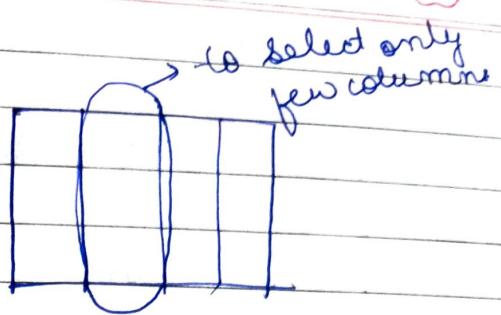
Date / /



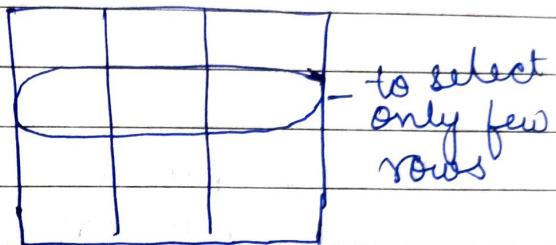
basic

## operations (unary)

$\pi$  - projection  
(attributes)

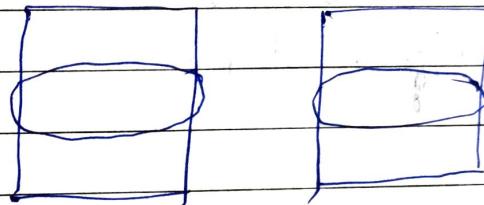


$\sigma$  - selection  
(tuples)



$\times$  - cross product

Combining 2 tables of  
same or diff type



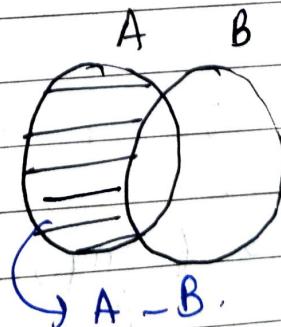
to compare tuples

$\cup$  - union

Combining 2 tables of  
same type.

- → minus

In A but not in B.



9 → rename

to rename a table



Sequence of oper<sup>n</sup>:

∩ : Intersection ( $A - (A - B)$ )

 : Join ( $\alpha x$ )

/ : Division ( $\pi, \times, -$ )

On cross product every tuple combines with every other table.

Join is an extension to cross product for selecting meaningful tuples.

### DB Selection Operation (a) (Horizontal partitioning)

ENo	Ename	Dno	Salary
1	a	1	10K
2	b	1	11K
3	c	2	12K
4	d	2	14K
5	e	3	15K
6	f	3	16K

⑨ Dno = 3 f Emp)

↑ Relation or

relational algebra query  
whose result is a reln.



All the employee with Dno = 1 having salaries > 10k.  
 $\alpha_{Dno=1} (\alpha_{Salary > 10k} (Emp))$  or  $\alpha_{Salary > 10} (\alpha_{Dno=1} (Emp))$

Nested expression

- \* Selection is commutative
- \* Degree remains same.

$$\alpha_A (\alpha_B (E)) \cong \alpha_B (\alpha_A (E))$$

$(\alpha_{Salary > 10} \text{ AND } \alpha_{Dno=1} (Emp))$

↑

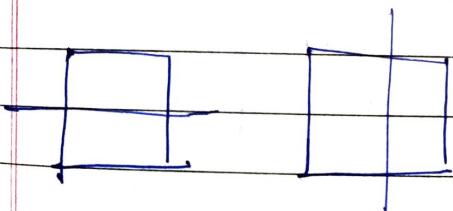
Boolean connectivity

$|\alpha_C(R)|$

min 0      max.  $|R|$

$$0 \leq |\alpha_C(R)| \leq |R|$$

DB-projection operation ( $\pi$ ) (vertical partitioning)



$\pi_{x,y} (Emp)$

Emp	x	y	z
1	a	c	
1	b	c	
2	a	c	
2	b	c	

	x	y
1	a	
1	b	
2	a	
2	b	

% of any RA expression is a relation.

Degree may be variant in the result

cardinality is not always same.

$$\pi_Z(\text{emp}) = \boxed{\begin{array}{c} Z \\ C \end{array}}$$

RA is derived from relational model so  
duplicate are removed.

$$\pi_X(\text{emp})$$

X
1
2

$$\pi_Y(\text{emp}) =$$

Y
a
b

Syntax :

$\pi_{\text{attribute list}} \text{ Reln}$

As long as you project superkey on attribute list you might get all the tuples.

\* It is not commutative.

$$\pi_X(\pi_{xy}(\text{emp})) \neq \pi_{xy}(\pi_X(\text{emp}))$$

(Duplicates SELECT  $\cong$  Projection. (Duplicates are allowed) (in SQL)  $\cong$  Projectn.  
SELECT distinct  $\cong$  Projectn. (Duplicates are not allowed))



Q: which of the following query transformations  
 (i.e. replacing LHS by the RHS expression)  
 is incorrect? R<sub>1</sub> & R<sub>2</sub> are reln's. C<sub>1</sub> and  
 C<sub>2</sub> are selection conditions and A<sub>1</sub> & A<sub>2</sub>  
 are attributes of R<sub>1</sub>.

a)  $\sigma_{C_1}(\sigma_{C_2}(R_1)) \rightarrow \sigma_{C_2}(\sigma_{C_1}(R_1))$

b)  $\sigma_{C_1}(\pi_{A_1}(R_1)) \rightarrow \pi_{A_1}(\sigma_{C_1}(R_1))$

c)  $\sigma_{C_1}(R_1 \cup R_2) \rightarrow \sigma_{C_1}(R_1) \vee \sigma_{C_1}(R_2)$

d)  $\pi_{A_1}(\sigma_{C_1}(R_1)) \rightarrow \sigma_{C_1}(\pi_{A_1}(R_1))$

cond'n may be applied  
on all or any attribute

cond'n to imposed on A<sub>1</sub>  
only.

b)

A <sub>1</sub>	A <sub>1</sub>	A <sub>2</sub>	A <sub>3</sub>
a <sub>1</sub>	a <sub>1</sub>		
b <sub>1</sub>	b <sub>1</sub>		
c <sub>1</sub>	c <sub>1</sub>		

c) R<sub>1</sub> ∪ R<sub>2</sub>

R<sub>1</sub> ∪ R<sub>2</sub>

d)

A <sub>1</sub>	A <sub>2</sub>	A <sub>3</sub>

A <sub>1</sub>	A <sub>2</sub>	A <sub>1'</sub>	A <sub>2'</sub>

A <sub>1</sub>	A <sub>2</sub>	A <sub>3</sub>

Date / /

B

Q. Suppose  $R_1(A, B)$  and  $R_2(C, D)$  are 2 relations. Let  $r_1$  &  $r_2$  be the corresponding reln. instances. B is a foreign key that refers to C in  $R_2$ . If data in  $r_1$  &  $r_2$  satisfy referential integrity constraints. Which of the following is always TRUE?

- a)  $\pi_B(r_1) - \pi_C(r_2) = \emptyset$  T
- b)  $\pi_C(r_2) - \pi_B(r_1) = \emptyset$  F
- c)  $\pi_B(r_1) = \pi_C(r_2) = \emptyset$  F
- d)  $\pi_B(r_1) - \pi_C(r_2) \neq \emptyset$

$r_1$	$r_2$ (instance name of table)
$R_1$	
A   B	C   D
1	1
2	2
	3

$$\{1, 2\} - \{1, 2, 3\} = \emptyset$$

$$\{1, 2, 3\} - \{1, 2\} \neq \emptyset$$

$$\{1, 2\} \neq \{1, 2, 3\}$$

$$\{1, 2\} - \{1, 2, 3\} \neq \emptyset$$

DB scenario operations (P)

Emp

A	B	C

$\pi_{A,B}(r_E(Emp))$



A	B

$P_x(c, d) \uparrow_{A, B} (a_c \text{ (emp)})$

renaming table emp as X and

renaming attributes A, B as C, D.

A ~~join~~ A  
join

$P_x \text{ (emp)}$

Rename employee as  
 $X$

Only attributes  $P_x(c, d, e) \text{ (emp)}$

$\downarrow P(c, d, e) \text{ (emp)}$

for few attributes  $\rightarrow P(x, y, c) \text{ (emp)}$

$P_x(s, t, c) \text{ (emp)}$

DB\_set operations  
 $\cup \cap -$  (binary)

RUS  $\rightarrow$  R and S should be union all type compatible.

same domain

$R(r_1, r_2, r_3)$

$S(s_1, s_2, s_3)$

same degree



$\text{emp\_c}_1$  (ename, eid)

$\text{emp\_c}_2$  (ename, ei)

union will never contain duplicate values.

$R \cup S = S \cup R$  (commutative)

$R \cup (S \cup T) = (R \cup S) \cup T$  (associative)

1

$R \cap S$  both the tables should be union compatible (same attributes & domain)

$R \cap S = S \cap R$  (commutative)

-  $R \cap (S \cap T) = (R \cap S) \cap T$  (associative)

$(R - S)$  both the tables should be  $\setminus$  compatible.

Result

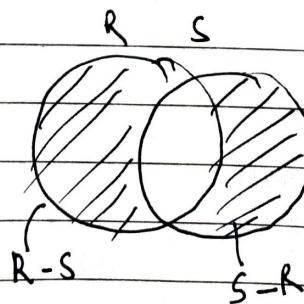
would be named

	R		S	
as R :	a	b	c	d
1	2	x	1	2
3	4	✓	7	8
5	6	✓	9	10

R	S
3	4
5	6

$R - S \neq S - R$  (not commutative)

$R - (S - T) = (R - S) - T$  (associative)



$$R \Delta S = ((R \cup S) - (R \cap S)) - (S - R)$$

$$R \Delta S = R - (R - S)$$



## DB - Cartesian Product

Cartesian product ( $\times$ )  $\rightarrow$  we will get ordered pairs.

$R(\text{emp})$

A	B	C
1	a	b
2	c	d
3	e	f

degree = 3 (k)

$S(\text{dep})$

D	E
1	c
2	d

X

2.

degree = 2 (l)

$R \times S$       Empid

A	B	C	D	E
1	a	b	1	c
1	a	b	2	d
2	c	d	1	c
2	c	d	2	d
3	e	f	1	c
3	e	f	2	d

degree = k + l

Empid

$3 \times 2 = 6$  tuples

$|R| = m \quad |S| = n$

$|R \times S| = (m \times n)$

To examine or compare 2 tuples from 2 diff. n. tables. We can get the name of employee & name of dependent.

$\sigma_{A=1} (R \times S)$

Emp dep

A	B	C	D	E
1	a	b	1	c
2	c	d	2	d

In general we use Cartesian product with selection to get some meaningful data.

subset of cartesian product

## DB-Join ( $\bowtie$ )

(extension for cartesian product)

$X, a \bowtie$

R	A	B	C
S	D	E	

$$\sigma_{A=D} (R \times S) \cong R \bowtie S \quad \langle A=D \rangle$$

(i)  $R \bowtie S$   
 (ii)  $\leftarrow$  join cond'n)

cond<sup>n</sup>1 AND cond<sup>n</sup>2

$$A=D \text{ AND } B=E$$

$$R \bowtie S \quad (\kappa + \lambda \text{ attributes})$$

$\langle A=D \text{ AND } B=E \rangle$

Join = only comparing 2 attributes  $(A=10) \times$   
 not join

$$0 < R \bowtie S < mxn.$$

DB-Natural join ( $\bowtie$ )

↓  
 (2 tables, common attribute  
 same name)

Emp	empid				X	Dep	Eid	

 $R * S$  $\cong$  $R \bowtie_{\{A=A\}} S \cong \sigma_{\{A=A\}}(R \times S)$ 

(A)	B	C	(A)	D
-----	---	---	-----	---

(should have the same)

if

reverse

(A)	B	C	(E)	D
-----	---	---	-----	---

 $R * \rho_{\{A,D\}}^{(S)}$ 

A	B	C	A	D
1	a	b	1	d
2	c	d	3	e

A	B	C	D
1	a	b	d

Common attributes  
needs not be written  
2 times

A	B	C	A	B
1	a	b	1	a
2	c	d	3	e

A	B	C
1	a	b

- 1) attributes might change
  - 2) can be applied on any no. of tables
- $0 < R * S < m * n$



When names of attributes are not same then  
natural join is equal to the cartesian product

	$A_1$	$B$	$C$	$D$	$E$
1	a	b		1	a
2	c	d		3	e.

	$A_1 \bowtie A_2$	$A$	$B$	$C$	$D$	$E$
1		a	b		1	a
		a	b		1	a
2		c	d		3	e
2		c	d		3	e

## DB - Division operation ( $R \div S$ )

Implementation of division with basic operations

1.  $T_1 \leftarrow \pi_{(R-S)}(R)$
2.  $T_2 \leftarrow \pi_{(R-S)}((S \times T_1) - R)$
3.  $T \leftarrow T_1 - T_2$

$R \div S$

$R$	$S$								
<table border="1"> <thead> <tr> <th><math>A</math></th> <th><math>B</math></th> </tr> </thead> <tbody> <tr> <td><math>a_1</math></td> <td><math>b_1</math></td> </tr> <tr> <td><math>a_2</math></td> <td><math>b_1</math></td> </tr> </tbody> </table>	$A$	$B$	$a_1$	$b_1$	$a_2$	$b_1$	<table border="1"> <thead> <tr> <th><math>A</math></th> </tr> </thead> <tbody> <tr> <td></td> </tr> </tbody> </table>	$A$	
$A$	$B$								
$a_1$	$b_1$								
$a_2$	$b_1$								
$A$									
	<table border="1"> <thead> <tr> <th><math>B</math></th> </tr> </thead> <tbody> <tr> <td></td> </tr> </tbody> </table>	$B$							
$B$									

$$\{A, B\} - \{A\} = \{B\}$$

Date / /

R		A		B	
		a <sub>1</sub>	a <sub>2</sub>	b <sub>1</sub>	b <sub>2</sub>
a <sub>1</sub>				b <sub>1</sub>	
a <sub>2</sub>				b <sub>1</sub>	
a <sub>1</sub>				b <sub>2</sub>	
a <sub>2</sub>				b <sub>2</sub>	
a <sub>1</sub>				b <sub>3</sub>	

b<sub>1</sub> & b<sub>2</sub> is  
present in  
comb<sup>n</sup> with  
both a<sub>1</sub> & a<sub>2</sub>

$$1) T_1 \leftarrow \pi_{(R-S)}^{(R)}$$

$$\pi_B(R) = \begin{array}{|c|} \hline B \\ \hline b_1 \\ b_2 \\ b_3 \\ \hline \end{array} T_1$$

$$2) T_2 \leftarrow \pi_{(R-S)}^{((S \times T_1) - R)}$$

$$S \times T_1 = \begin{array}{|c|c|} \hline A & B \\ \hline a_1 & b_1 \\ a_1 & b_2 \\ a_1 & b_3 \\ a_2 & b_1 \\ a_2 & b_2 \\ a_2 & b_3 \\ \hline \end{array}$$

$$(S \times T_1) - R$$

$$\begin{array}{|c|c|} \hline A & B \\ \hline a_1 & b_1 \\ \cancel{a_1} & b_2 \\ \cancel{a_2} & b_3 \\ \cancel{a_2} & b_1 \\ \cancel{a_2} & b_2 \\ \cancel{a_2} & b_3 \\ \hline \end{array}$$

$$T_2 = \begin{bmatrix} B \\ b_3 \end{bmatrix}$$

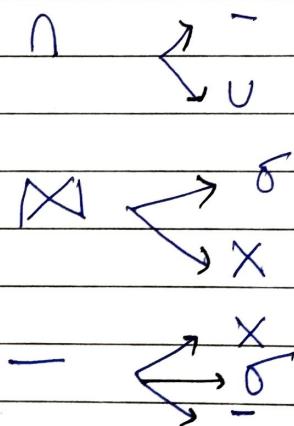


$$3. T \leftarrow T_1 - T_2$$

B
b <sub>1</sub>
b <sub>2</sub>

### DB - Complete set of RA operations

We can say that the following set of RA operations  $\{ \cap, \pi, \cup, -, \times \}$  is a complete set that is any of the other RA oper<sup>n</sup>. can be expressed as a sequence of oper<sup>n</sup>. from this set -



### DB - Types of Join

left ~~inner~~ outer join

Right outer join

Full outer join

R			S	
	A	B	C	D
1	a		1	b
2	c		3	d

Date \_\_\_\_\_

A = D

A	B	C	D
1	a	1	b
Null	a	1	b



R ~~X~~ S  
A = C

(R ~~X~~ S)  
A = C

A	B	C	D
1	a	1	b

A	B	C	D
1	a	1	b
2	c	Null	Null

left outer R ~~X~~ S.  
join  
A = C.

whatever present on the left side & not included in the table should be present there.

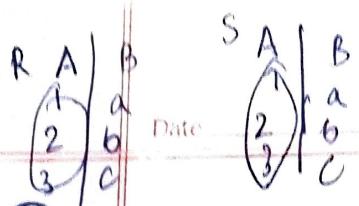
right outer R ~~X~~ S.  
join

whatever present on the right side & not selected, should be present in the o/p

R ~~X~~ S (present on both the sides)

A	B	C	D
1	a	1	b
2	c	N	N
N	N	3	d

X ~~X~~ ~~X~~ ~~X~~ ~~X~~



$\downarrow$   
 $m$

④ ~~remove tuple.~~

	min	max
$\times$	$mn$	$m n$
$\bowtie$	$0$	$mn$
$\bowtie$	$'m'$	$mn$
$\bowtie$	$'n'$	$mn$
$\bowtie$	<del><math>mn</math></del>	$mn$
	$mn(m,n)$	$mn$ (when all the tuples are matching with all the tuples of other side)

(if nothing is mentioned)

#### 14. DB - Extended RA oper'n's

Generalized projection : Can use arithmetic oper'n such as  $+$ ,  $-$ ,  $*$ ,  $\div$  on numeric attributes, numeric constants and on an expression that generate numeric result. Also permits oper'n' on other datatypes such as concatenation of strings.

eg:  $\pi_{ID, name, dept\_name, \text{salary} \div 12}$  (\$instructor)

Aggregate function: (SUM, AVERAGE, MAX, MIN, COUNT, COUNT-DISTINCT)

$f_{\text{function list}}(R)$

$f_{\text{(Employee)}}$

COUNT SSN,  
AVERAGE SALARY

SSN  $\rightarrow$  Social Security number.

Aggregate functions with GROUP BY.

$f_{(DNO)}$  COUNT SSN, AVERAGE SALARY

P

(DNO, NO, AVG(SAL))

(Emp)



Q Consider the join of a relation R with a relation S. If R has m tuples and S has n tuples, then the max<sup>m</sup> and min<sup>m</sup> sizes of joins are respectively?

- a)  $m+n, 0$
- b)  $mn, 0$
- c)  $m+n \mid m-n \mid$
- d)  $mn, m+n$

R

S

$\downarrow$   
m

$\downarrow$   
n

R  $\bowtie$  S

$$\min^m = mn, 0$$

Q The relational algebra expression equivalent to the following tuple condition calculus expression

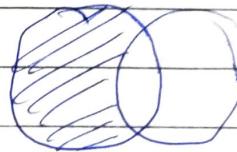
$$\{ t / t \in \pi \wedge (t[A] = 10 \wedge t[B] = 20) \}$$

a)  $\sigma_{(A=10 \vee B=20)} y$

$A=10 \quad B=20$

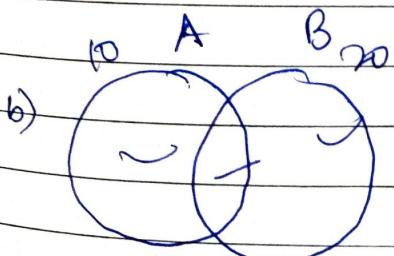
b)  $\sigma_{A=10} (\bar{x}) \cup \sigma_{(B=20)} (y)$

d)

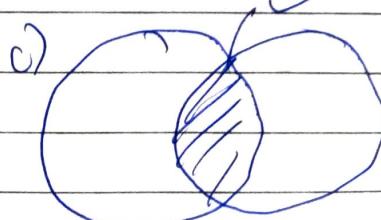


c)  $\sigma_{A=10} (\bar{x}) \cap \sigma_{(B=20)} (y)$

d)  $\sigma_{A=10} (y) - \sigma_{(B=20)} (y)$

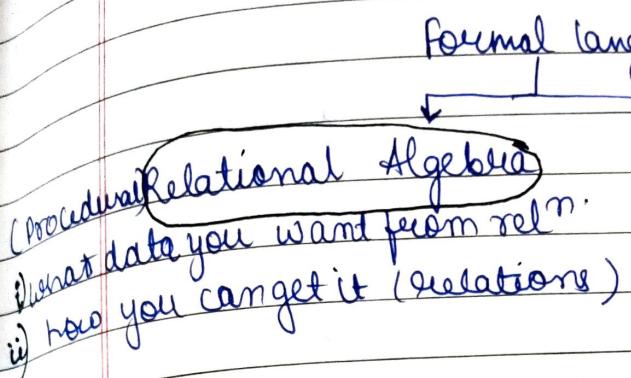


$A=10 \quad B=20$





## Introduction to relational Calculus.



i) what we want

→ (declarative lang)  
Relat<sup>n</sup> calculus

Tuple R.C

Domain R.C

They will be helpful to implement any query pr.

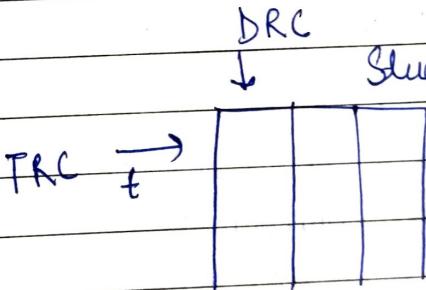
\* If a language is able to express everything a RA can then it is called as relationally complete.

we generally range over the column.

TRC, DRC, RA → relationally complete

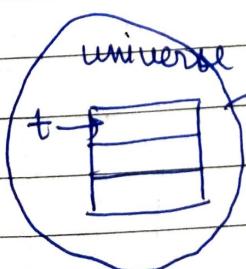


take each  
tuple at a time  
& examine



Unsafe operat<sup>n</sup>. Student ( $t$ )

$\sim t \rightarrow$  all the tuples not  
in student table.  
(infinite).



$\sim t$  RA and RC are  
of same power but  
due to unsafe operat<sup>n</sup>.  
RA is more powerful  
than RC (Relational  
calculus)



## DB TRC Syntax

Student

FN	LN	marks
a	b	100
b	c	90
c	f	60

FN of student  
having marks  $> 50$

$\{ t \cdot FN \mid \text{student}(t) \text{ AND } t \cdot M > 50 \}$   
 set of tables AND condition  
 all attributes in final QP

$\{ t_1 \cdot A_1, t_2 \cdot A_2, \dots \mid T_1(t_1) \text{ AND } T_2(t_2) \dots \text{ AND } t_i \cdot A_i \}$

DB - FREE AND BOUNDED variables :

$\{ t_1 \cdot A_1, t_2 \cdot A_2 \}$

Bounded variables

expression

atomic (simple) or  
composite (complex)

Emp

t	A
	21
	20

$\rightarrow \text{emp}(t)$

$A_1 \wedge A_2$  AND

$\rightarrow t \cdot A > 20$

$A_1 \vee A_2$  OR

$\rightarrow t_1 \cdot A_1 > t_2 \cdot A_2$

$\neg A_1$  NOT

$t_1 \rightarrow A_1$

$t_2 \rightarrow A_2$

$\exists t ( )$  Quantifier  
for all

$\forall t ( )$



~~Unit for processing  $\rightarrow$  Entire table~~  
Bounded variables

One tuple,  
free variable

there exists  
t

$\exists t ( \quad )$  } having quantifiers  
 $\forall t ( \quad )$  } quantifiers

no quantifiers

$\exists t (t(A) > 50)$  false

there exists atleast one tuple  
such that a A value is 50.

A	B
10	1
20	2
30	3
50	4

$\nexists t (t(B) < 10)$

True

processing the entire table

processing step by step.

In bounded variable we have to check the entire table i.e if  $t(B) < 10$  then only it will return true or false but in case of free variable only one tuple needs to checked each time.

$\neg \exists t (t(A) > 50)$   
doesn't exist

$\exists t (t(A) > 50)$   
there exists

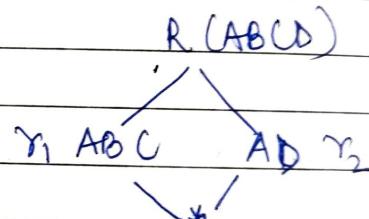
Q19. Let  $r$  be a relation instance with schema  $R = (A, B, C, D)$ . We define  $r_1 = \pi_{A, B, C}(r)$  and  $r_2 = \pi_{A, D}(r)$ . Let  $s = r_1 * r_2$ , where  $*$  denotes natural join. Given that the decomposition of  $r$  into  $r_1$  &  $r_2$  is lossy. Which one is true?

- i)  $S \subset r$
- ii)  $r \cup S = r$ .
- iii)  $r \subset S$
- d)  $r * S = S$ .

$r_1$			$r_2$		$r$			
A	B	C	A	D	A	B	C	D
1	$b_1$	$c_1$	1	$d_1$	1	$b_1$	$c_1$	$d_1$
1	$b_2$	$c_2$	1	$d_2$	1	$b_2$	$c_2$	$d_2$
2	$b_3$	$c_3$	2	$d_3$	2	$b_3$	$c_3$	$d_3$
2	$b_4$	$c_4$	2	$d_4$	2	$b_4$	$c_4$	$d_4$

$S = r_1 * r_2$

A	B	C	D



lossy - after joining we get more tuples.

$r \subset S$

final table must be a superset of initial table

1	$b_1$	$c_1$	$d_1$	$\circlearrowleft$
1	$b_1$	$c_1$	$d_2$	$\circlearrowleft$
1	$b_2$	$c_2$	$d_1$	$\circlearrowleft$
1	$b_2$	$c_2$	$d_2$	$\circlearrowleft$

Date / /



## Introduction to DRC

SQ L - most prac. lang.

$$RA + TRC = SQ L$$

$$DRC = QBE$$

R TRC

A	B	C
1		
	1	
		1
1	1	1

t:A t:B

t:C

a	b	c
1		
	1	
		1
1	1	1

R ↓ DRC ↓ ↓

A | B | C

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

# SQL

E.F Codd - A relational model of data for large shared data banks.

RAYMOND and DONALD: SEQUEL



structured English query language

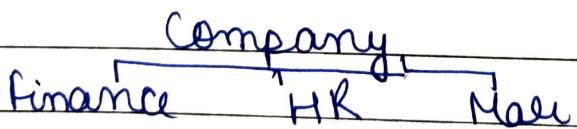
SQL

- 1) It is based on RA and TRC
- 2) First implemented on System R by IBM
- 3) Oracle, Microsoft SQL Server, MS Access, MySQL etc
- 4) Portable
- 5) Basics and extensions

## Creation of Schema

Schema: Groups logical objects together

For privacy concerns we group objects together



Syntax: CREATE SCHEMA Schema\_name [AUTHORIZATION owner\_name]

CREATE SCHEMA FINANCE AUTHORIZATION RAVI;  
GRANT SELECT ON FINANCE \*.\* TO user;



## Create Table

Syntax: CREATE TABLE <tablename> (<col1>datatype (width),  
                   <col2> datatype (width); ...);

Eg: CREATE TABLE Customer ( Name varchar(20),  
                   cust\_id Number,  
                   Address varchar(50));

Datatypes: We have numerous datatypes in SQL

Numeric	Character	Date	Boolean
NUMBER	CHAR	DATE	BOOLEAN
FLOAT	VARCHAR2	TIME	
INT	NCHAR	TIMESTAMP	
REAL	NVARCHAR2	INTERVAL	
DECIMAL	LONG		
BINARY	RAW		
FLOAT	LONG RAW		

Constraints: We have 7 constraints in SQL

- 1) NOTNULL (one attribute shouldn't contain null value)
- 2) UNIQUE (no two tuples can have same value)
- 3) Primary key (should be unique + NOTNULL)
- 4) Foreign key (referential integrity (attribute in table referring to attribute of other table))
- 5) CHECK (sometimes we need to check any attribute)
- 6) DEFAULT (default value is set when there is no value)
- 7) REF constraints (



Our requirement is:

Dept table with dep\_id(PK) dept\_name (UNIQUE)  
location.

CREATE TABLE DEPARTMENT(

deptid NUMBER PRIMARYKEY,  
dept name varchar(30),  
location varchar(100),  
UNIQUE (dept - name) );

Constraints could be defined at 2 levels.

a) column level  
↓

dept - id NUMBER PRIMARYKEY

Specifying the constraints  
along with the column  
name.

b) table level

dept - name varchar(30)  
UNIQUE (dept - name));

Specifying the constraint  
at the last of the  
columns.

- \* NOT NULL should be specified at column level
- \* composite key should be specified at table level.  
(more than one attribute as key)

Emp: emp\_id (PK) emp\_name (NOT NULL)  
job , dept\_id (REF <sup>referring</sup> dept\_id of Dept)  
mgu , salary (> 5000)  
comm (if not given default 100)  
email ,  
phone unique .



CREATE TABLE EMPLOYEE (

- emp\_id NUMBER PRIMARY KEY,
- emp\_name varchar(30) NOT NULL
- dept\_id NUMBER REFERENCES Dept(dept\_id),
- job varchar(2(30)), mgr varchar(30)
- salary NUMBER CHECK (salary >= 5000),
- comm NUMBER DEFAULT 100,
- email varchar(30),
- phone varchar(12),
- UNIQUE (email, phone));

### DB-INSERT.

Insert: It is used to insert data.

Syntax: In that order

INSERT INTO <tablename>

values (attr. value), attr. value 2... )

Ex: INSERT INTO Dept

values (1, 'CSE', 'Hyderabad')

Syntax 2: In other order.

INSERT INTO <tablename> (attr1, attr2, attr3...)

values (attr-value1, attr-value2...)



## DB - delete and Update

Delete : Delete the data from existing table.

Syntax :

`DELETE FROM <table name>  
WHERE < condition >;`

Delete \* from Emp;  
Emp.  
(table  
exists)

`DROP TABLE Emp;`  
(table &  
data deleted)

Dept			Emp		
dept_id	dept_name	locn	emp_id	dept_id	ename
2	ece	kgp	2	2	def
3	ecc	hyd.	3	1	ghi

Delete from Dept

where dept\_id = 1; →

Referential integrity  
gets violated

Drop : deletes both the data & table

Syntax : `DROP TABLE <table name>`

We can't use insert after drop.



Update : Used to update the data.

Syntax: UPDATE <table name>  
SET <attr> = <Value>  
WHERE <condn>;

Eg:

```
Update Emp.  
SET dept_id = 1  
WHERE emp_id = 2;
```

### DB - Referential triggered actions

Referential integrity gets hindered when you try to update or delete the data. So to maintain the referential integrity

dept_id	dept_name.	emp_id	dep_id
1	CSE	1	3
2	ECE	2	2
3	EEE	3	4
4	IT	4	6
5	CE	5	5
6	ME	6	1

On delete

on update

- ① Set Null
- ② Set default
- ③ Set cascade



It allows us to further describe the relationship between the ref. columns and the object it references by attaching a 'referential triggered action' to the foreign key.

3 actions are

- 1) SET Null
- 2) SET Default
- 3) SET cascade

### SET NULL

```
CREATE TABLE Emp (emp_id NUMBER,
Dept_id NUMBER REFERENCES Dept(dept_id)
ON DELETE SET NULL ON UPDATE SET NULL)
```

Ex: DELETE FROM Dept  
WHERE dept\_id = 1;  
UPDATE Dept  
SET dept\_id = 7  
WHERE dept\_id = 6;

SET Default (default value is never be deleted  
to maintain the integrity constant)

### SET DEFAULT

Ex: dept\_id NUMBER DEFAULT 3 REFERENCES Dept(id)  
ON DELETE SET DEFAULT ON UPDATE SET DEFAV

DELETE FROM Dept WHERE dept\_id = 2;  
UPDATE Dept SET dept\_id = 8 WHERE dept\_id

## SET CASCADE



dept\_id NUMBER REFERENCES Dept(dept\_id)  
ON DELETE SET CASCADE ON UPDATE SET CASCADE  
  
DELETE FROM Dept WHERE dept\_id = 4;  
UPDATE Dept SET dept\_id = 9 WHERE dept\_id = 5;

## DB Alter

Alter: It is used to add, delete or modify columns in an existing table.

Also used to add and drop various constraints on an existing table.

Create table Emp( emp\_id NUMBER, emp\_name CHAR(10)  
UNIQUE, Salary NUMBER  
check(salary >= 5000),  
Phone VARCHAR(13),  
pincode VARCHAR(6));

ALTER TABLE Emp  
ADD PRIMARY KEY(emp\_id);

ALTER TABLE Emp  
DROP UNIQUE (emp\_name);

ALTER TABLE Emp  
MODIFY CHECK (Salary > 1000);



for attributes

→ ALTER table Employee  
 ↳ ① ADD Add varchar(30);  
 ↳ ② MODIFY phone varchar(20);  
 ↳ ③ DROP COLUMN pincode;

### DB - Select

Used to retrieve data from database.

Syntax:    SELECT <attr\_list> π  
 FROM <table\_list> ×  
 WHERE <conditions>; a

attr\_list : list of attr whose values are to be retrieved by the query.

table\_list : list of tables from which we retrieve the data.

Condition: Conditional [Boolean] expressions that identifies the rows retrieved by query.

Dept		Emp		
dept_id	deptname	emp_id	dept	emp_name
1	CSE	1	5	Sindhu. 9
2	IT	2	2	Serinivas. 6
3	ECE	3	1	Santosh 9
4	EEE	4	2	Saumya 6
5.	ME	5	4	Sindhu 6
		6	3	Saumya 6

Q. Retrieve the names of all employee who work for dept 1.



Select emp\_name  
from Employee  
where dept = 1;

Q. Retrieve the names ,emp\_id of the employees who work for IT dept.

Select emp\_name, emp\_id  
from Employee, Dept  
where dept = dept\_id  
AND dept\_name = 'IT';

Q. Retrieve the jobs of all employee.

Select Job from Emp; {a,b,a,a,b,c}

Select distinct Job from Emp; {a,b,c}.

DB\_view.

dept_id	dept_name	stud_id	stu_name	dept
1	CSE	1	Sindhu	2
2	ECE	2	Sowmya	1
3	ME	3	Seinivas	3
4	EEE	4	Santosh	2
5	CE	5	Subbu	4
		6	girnarani	2
		7	satish	2



Create table Student\_ece as

( Select stu\_name, stu\_id

from student , Dept

where dept = dept\_id AND .

dept\_name = 'ece' );

Same data at multiple places  $\rightarrow$  coherence

Dynamic table - view

View : It is a virtual table based on the result set of a SQL statement.

Syntax : CREATE VIEW <view name> AS  
 (Select <columns> FROM <table name>  
 where <condition>

### DB-Aliasing

SQL aliases are used to nickname a table or a column in a table.

- Q. For each employee retrieve employee's id, name, salary & the name of his/her immediate supervisor.

Dept_id	Dept_name
1	Accounting
2	Sales
3	Research
4	Finance

## Emp



emp_id	dept_id	sup_id	Sal	emp_name
1	1	7	11000	Sindhu
2	3	4	2200	Seumya
3	1	7	2100	Sentoch
4	4	7	3000	Satish
5	2	4	2000	Lincharni
6	1	7	5000	Murjali
7	1	-	1500	Ravi

Select E.emp\_id AS "Employee\_id", Employee\_id | Salary | employee\_name  
 E.Sal AS "Salary",  
 E.emp\_name AS "Employee\_name",  
 S.emp\_name AS "Supervisor\_name"  
 FROM EMP AS E, EMP AS S  
 WHERE E.<sup>sup</sup>emp\_id = S.emp\_id ;

Retrieve the ~~name~~ <sup>id</sup> of employee and their corresponding name of dep.

Select E.emp\_id AS "Employee Id",  
 D.dept\_name AS "Department\_name"  
 FROM Emp E , Dept D  
 WHERE E.dept\_id = D.dept\_id ;

## DB\_Pattern Matching

LIKE : It is used to search for a specific pattern in a column.

We used a wild card

$\%.$  → represents any sequence of 0 or more char

$_{-}$  → used to replace a single character,

a  $\%.$  → first character in the name is a.

$\%.$  a → second character in the name is  $\overset{a}{\underset{\text{under}}{|}}$

$\%.$  a  $-$  → ending with a.

$\%.$  a  $-$  → last one char is a.

Q. Retrieve all the students whose name starts with R.

Select \* from student  
WHERE name like 'R%';

Q. Display the names of students who secured 4 digit rank.

Select name from student  
WHERE rank like '\_\_\_\_';

$\%.$  as the second symbol use escape sequence  
to search for data field as a  $\%.$ .

Select name from student where marks like '90%'  
escape '\' and email like 'abc\\_\%.\@%\\_.%\'  
escape '\'



Date

NOT LIKE

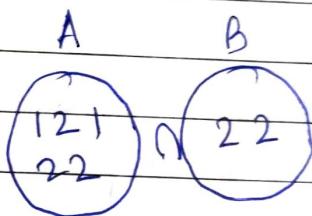
where name NOT LIKE 'R%';

name of students whose name  
didn't start with R.

## DB Set operations.

Union      ( $\cup$ )  
 Intersect    ( $\cap$ ) }      }      duplicates are not  
 Except      (-)      }      allowed.

Union all    }  
 Intersect all }      }      duplicates allowed.  
 Except all    }



Retrieve the list of students who got either grade 'A' or grade 'B'. (with duplicates)

Select studId from Enroll  
where grade = 'A' ;

$$\text{Union} = (1, 2)$$

Union All .

$$\text{Union all} = (1, 2, 1,$$

$$2, 2)$$

Select studId from Enroll.  
where grade = 'B' ;

Both grade 'A' and grade 'B'.

Select StudId from Enroll where grade = 'A' ;  
Intersect All.

Select StudId from Enroll where grade = 'B' ;

$$\text{Intersect} = (2)$$

$$\text{Intersect All} = (2, 2)$$



Date \_\_\_\_\_

who got ~~not~~ grade 'A' but not grade 'B'.

Select StudId from enroll where grade='A';

Except

Select StudId from enroll where grade='B';

Except = { 1 }.

Except All = { 1, 1, 2 }.

## DB Arithmetic operators

SQL allows use of arithmetic operators in queries on numeric domain.

- Q. Increase the salary of an employee by 10%. and display the salary and employee name.

Select EmpName, Salary \* 1.1 From Employee;

- Q. CONCATENATE - For string datatype the concatenate operator '||' can be used in a query to append 2 string values.

Select Fname || Lname AS "Full Name"  
from Employee;

BETWEEN - It is a comparison operator on numeric datatype

- Q. Retrieve all the employees whose salary is between 30,000 and 50,000.



Date

Select \* from Employee  
where salary Between 30000 AND 50000;

NOT BETWEEN - vice versa on between

between can be used on text & dates .

Select \* from Employee where Emp-Name  
Between 'A' and 'D';

%P A , AA , B , Ba , D , Dax

↓  
greater  
than  
D not  
displayed.

In case of strings

Between = [A, D] Inclusive

NOT Between = (A, D) Exclusive . (B, C, D)

Select \* from Employee where .

hire date Between '1-JAN-2010' AND '1-JUL-2010';

DB- ORDER BY .

Order by is used to order our SQL query  
in ascending or descending order .

Syntax : Select <column list> FROM  
<table list> ORDER BY  
<column 1> ASC / DESC, <column> ASC /  
DESC ;



A	B	C
1	2	3
1	2	1
2	1	3
2	1	1
3	5	4
3	4	3

Select A,B,C from R ORDER BY A,B,C ;

In Asc order 1 2 1 ← O/P.

1 2 3.

2 1 1

2 1 3.

3 4 3.

3 5 4

Select A,B,C from R ORDER BY A DESC, B,C DESC

O/P → 3 4 3.

3 5 4 .

2 1 3

2 1 3 1

1 2 3

1 2 3 1

1 2 3 1

Order by Example

Q Retrieve the list of employee names , their dept & project names they are working on ordered by dept name & within each dept. orders



alphabetically by last name, fname.

Select d.dname, e.lname, e.fname, p.name from  
 EMPLOYEE e, works\_on w, PROJECT p, Department  
 where d.number = e.Dno AND  
 e.ssn = w.essn AND  
 w.pno = p.pnumber;

Order by d.dname, e.lname, e.fname;

	Dname	Lname	Fname	Pname
1	a	a	b	p <sub>1</sub>
3	b	a	c	p <sub>2</sub>
2	a	b	c	p <sub>3</sub>
5	c	c	b	p <sub>3</sub>
4	b	c.	b.	p <sub>2</sub>

### DB null values.

Dealing with null values at diff<sup>n</sup> context.

- Arithmetic expressions (+, -, \*, /)
- Logical expressions

1)	A	A+1	Null + arithmetic op = NULL
	5 + 1	6	
	4 + 1	5	
	Null + 1	NULL	
	1 + 1	2	

2)  $1 < \text{NULL}$  unknown  
 $(1 \stackrel{\text{UN}}{<} \text{NULL}) \text{ AND } (1 \stackrel{T}{<} 2)$



Date: / /

A	B	A AND B	A OR B
T	T	T	T
T	F	F	T
F	T	F	T
F	F	F	F
T	UK	UK	T*
F	UK	F*	UK
UK	UK	UK	UK

for null values.

$$\text{NOT}(T) = F$$

$$\text{NOT}(F) = T$$

$$\text{NOT}(UK) = UK$$

### DB Null values in various contexts

R

A	B
a	1
b	2
c	NULL
d	NULL

Select A

from R

where B = NULL X

where B IS NULL ✓

o/p c, d.

Where B IS NOT NULL

o/p → a, b

e		a	
N		N	
a	X	a	
b		b	
c	X X	c	
NULL	X	NULL	
NULL	X X	NULL	

we don't get NULL  
value in join  
selection

using DISTINCT we can consider every NULL as same



R

A	B.
a	1
b	2
c	NULL
d	NULL

Select B from R  
1, 2, N, N

Select DISTINCT B from R.  
1, 2, N.

(1, 2, N, N) UNION (1, 2, N, N)

INTERSECT

(1, 2, 3, N) → (1, N)

## IN Operator

The IN operator allows you to specify ~~multiple~~ multiple values in a where clause:

Syntax:

Select column\_name(s) FROM Table name  
WHERE Column-name IN (val1, val2...);

Ex: Retrieve the names of employees who works for the dept. 2, 3, 4, 5.

Select fname

From Employee

Where dept IN (2, 3, 4, 5);

- Q. Find the FName, Address of employees who work for the department located in 'New York'.

Select FName, Address

From Employees

WHERE DNo IN

Subquery → (Select DNo from Dept Location

*correlated | non correlated*      where location = 'NEWYORK');

↓  
Non correlated

Inner table could be executed first

- Q. Retrieve the Fname, Lname of all managers.

Select FName, Lname

From Employee e

WHERE SSN IN

(Select MgrSSN from Department d);

- Q. Retrieve the Fname of each employee who has a dependent with the Fname & same Sex as the employee

*Correlated* Select (e.FName) FROM EMPLOYEE e  
 WHERE <sup>e.SSN</sup> IN (Select essn from ~~Employee~~ department d  
 WHERE (e.Fname = d.dependent name AND e.sex = d.sex)



Date - / /

- Q. find out all the employee id's who worked on same (or) any project on which employee 10 has worked for same number of hours.

Select distinct essn from WORKS\_ON  
 WHERE (Pno, hours) IN  
 (Select pno, hours from WORKS\_ON  
 WHERE essn = 10);

essn	Pno	Hours
10	20	100
20	20	100
30	10	10
40	20	100
10	20	20

20	100
30	20

$$10, 20; 40; 10 \Rightarrow 10, 20, 40$$

DB- Any/ SOME , ALL ( $>$ ,  $<$ ,  $\geq$ ,  $\leq$ ,  $=$ )

- Q. find out names of all the employees whose salary is greater than all employees in Dno = 5.

Select Fname from Employee

WHERE salary > ALL( Select salary from employee where Dno = 5);

ALL

a, 11K

(10K, 20K, 30K) ✗

SOME/ANY

a, 11K

(10K, 20K, 30K) ✓

? any / = some  $\Rightarrow$  = in



= any      b, 10K (10K, 20K, 30K) ✓

\* in IN we can compare more than 2 values

only one value can be compared

< > (NOT Equal to)

<> some  $\hat{=}$  <> Any  $\rightarrow$  not equal to any  
 $\hat{=}$  NOT IN

### DB - Exists / NOT Exists

The SQL exists condition is used in comb<sup>n</sup> with a subquery and is considered to be met if the subquery returns atleast 1 row.

Q. Retrieve the fnames of employees who have dependents with same fname, sex as that of the employee.

essn	Sex	Dept
10	m	a
30	f	b

ex) 

SSN	Fname	Sex
10	(a)	M
30	b	F

Select fname from Employee e  
WHERE EXISTS (Select \* from Dependent d WHERE e.ssn=d.essn  
AND e.sex=d.sex AND e.fname=d.dept)

Q. Retrieve the fnames of the employee who have no dependent.

Select fname from Employee e

WHERE NOT EXISTS (Select \* from Dependent  
where SSN  $\neq$  e.ssn)



DB - Examples on EXISTS & NOT EXISTS

list the Fname of managers who have atleast one dependent.

Select Fname from Employee e.

WHERE EXISTS (Select \* from

dependent d where e.ss = d.essn

AND EXISTS (Select \* from Department

Dept where e.ssn = Dept.Mgrssn);

Employee

Department

Dependent

Frame	SSN		Deptno	Mgrssn	essn
a	(10)				
b	20		(10) →	(10)	20

① He is manager or not

② He has dependent or not

Q. Retrieve the Fname of each employee who works on all the projects controlled by dept number 5.

Select Fname from Employee e

WHERE exists (Select Pnum from WORKSON W WHERE Dnum = 5)

EXCEPT (Select Pno from WORKSON W WHERE Dnum = 5)

(Select Pno from WORKSON W WHERE Dnum = 5)

A-B

Date \_\_\_\_\_



## Non correlated query

employee	Project	Works on			
Fname	SSN	Pno	dNo	P no	eSSN
a	(1)	(10)	5 ✓	10	(1)
		20	1	15	1
b	(2)	(30)	5 ✓	25	1
		(25)	5 ✓	30	1
				(40)	2

Select Fname from Employee e where NOT EXISTS

$$(10, 15, 25) \setminus (10, 15, 25) \quad \text{returns True}$$

$$- \quad (10, 15, 25, 30) \quad \text{returns false.}$$

O/p → a.

## DB Aggregate Function

Set



noduplicates

multiset



duplicates

Aggregate functions are functions that take a collection (a set or multiset) of values as input & return a single value.

?) AVG : Average

MIN : Minimum

MAX : Maximum

~~SUM~~ SUM : Total

COUNT : Count



Date \_\_\_\_\_ / \_\_\_\_\_ / \_\_\_\_\_  
 Select AVG(salary)  
 from emp;

Select COUNT(salary) → how many salary  
 from emp;

Avg(MARKS\_A + MARKS\_B) ✓

WHERE SUM X not allowed where with  
 Aggregate function

Dealing with NULL values in Aggregate func<sup>n</sup>.

All aggregate func<sup>n</sup> except count(\*) ignore  
 NULL values in their input collection.

\* The count of an empty collection is defined to  
 be zero(0) & all other aggregate opern's return  
 a value of NULL when applied on an  
 empty set.

A	B	Count(*) = 4	Sum(A) = 6
1	N	Count(A) = 3	Avg(A) = 2
2	N	Count(B) = 0	Max(A) = 3
N	N		Min(A) = 1
3	N		

$$\begin{aligned} \text{Sum}(B) &= N & N \rightarrow \text{null} \\ \text{Avg}(B) &= N \\ \text{Max}(B) &= N \\ \text{Min}(B) &= N \end{aligned}$$



## DB - Group By clause.

- Q. For each department that has more than 5 employees retrieve the department name & the number of its employees who are making more than 40,000.

Select Dname, count(\*)

FROM Department, Employee WHERE

Dnumber=Dno and Salary > 4000 AND

Dno IN ( Select Dno from Employee  
 GROUP BY Dno Having count(\*)>5 )  
 GROUP BY Dname; !

- Q. Find the avg salary in each dept.

Select Dno, Avg(salary)

FROM Employee

GROUP BY DNo ;

Employee

SSN	Dno	Sal				
10	1	100	1	100	1	250
20	2	200	1	400	2	350
30	3	300	2	200	3	450
40	1	400	2	500		
50	2	500	3	300		
60	3	600	3	600		



All attributes used in GROUP BY clause need to appear in the Select clause. Any attribute that is not present in GROUP BY clause must appear only inside the aggregate in the select clause.

→ for filtering group HAVING clause.

S F W (G) (H)

Only Group by is there, then there is Having.

Ex: list the dno with Avg salary > 20000.

Select Dno , Avg(salary)  
FROM employee  
GROUP BY DNo.

HAVING AVG(salary) > 20000;

DB - Create 2012.

Consider the following reln. A, B, C.

A:	Id	Name	Age
	12	Arun	60
	15	Shrey	24
	99	Rohit	11

B:	Id	Name	Age
	15	Shrey	24
	25	Hari	40
	98	Rohit	20
	99	Rohit	11

C:	Id	Phone	Age
	10	2200	02
	99	2100	01

How many tuples does the result of the following SQL query contain?

Select A.Id from A WHERE A.Age > ALL (select B.Age FROM B WHERE B.Name = 'Arun');



B

age

ALL (empty set)

	A
12	
15	
99	

Ans: 3

### DB\_ With Clause

The SQL "with" clause allows you to give a sub-query block a name, a process also called sub query factoring which can be referenced in several places within the main SQL query.

The name assigned to sub-query is treated as though it was a view or table. It is basically a drop in replacement to the normal sub query.

Ex: Select emp\_id from Employee;

Using with : WITH Emp\_tab AS( Select emp\_id  
from Employee )

Select \* from Emp\_tab ;

Multiple WITH clauses:

WITH Dept\_temp AS( Select dept\_id from Department  
WHERE d\_name='CSE'),

Emp\_name AS( Select emp\_name from employee E,  
Dept\_temp D WHERE E.deptid = D.dept\_id )

Date / /

Q. For each employee, display the number of employees working in their respective department.

Select e.ename AS emp-name, DC.dept\_count AS emp-dept\_count FROM emp e,  
(Select dept\_no, count(\*) AS dept\_count  
FROM emp GROUP BY dept\_no) DC  
WHERE e.dept\_no = DC.dept\_no;

WITH:

WITH dept\_count AS (Select deptno, count(\*) AS dept\_count  
from emp GROUP BY deptno)

Select e.ename AS emp-name, DC.dept\_count AS emp-dept\_count  
FROM emp e, dept\_count DC  
WHERE e.dept\_no = DC.dept\_no;

R

Joins

i) Normal Join

Select \* from (R JOIN S ON A=C)

A	B
1	
2	
3	
4	
5	
6	

S

C	D



Normal Join takes place when we provide any condition.

R		S	
A	B	C	D
1	a	1	g
2	b	2	h
3	c	3	i
4	d	7	j
5	e	8	k
6	f	9	l

Select \* from ( R JOIN S ON A = C )

A	B	C	D
1	a	1	g
2	b	2	h
3	c	3	i

- 2) Natural Join : Natural join doesn't have condition. So natural join only takes place when there is an ~~condition~~ matched attribute

Rename C as A

A	B	D
1	a	g
2	b	h
3	c	i



3. left outer Join: Join 2 tables add whatever is in left table.

A	B	C	D
1	a	1	g
2	b	2	h
3	c	3	p
4	d	N	N
5	e	N	N
6	f	N	N

4. Right Outer Join: Join 2 tables add whatever is right table.

A	B	C	D
1	a	1	g
2	b	2	h
3	c	3	p i
N	N	7	j
N	N	8	k
N	N	9	l

5. full outer Join: Add whatever present on left as well as right side.

## Difference between Alter & update

Alter

- 1) It is a DDL Command
- 2) It works on table structure

Update

- 1) It is DML command.
- 2) It works on table data

update Emp

Set salary = salary \* 2  
where name = "A";

Emp				
id	name	Salary	Email	
1	A	10 000		
2	B	20 000		
3	C	30 000		

20 000  
40 000  
60 000

Difference between deleted  
drop, truncate

			(logs are there)	(no logs are there)
Rollback ✓	rollback ✗			
Delete	Drop	Truncate		
DML	DDL	DDL		
Delete from Student; Every row is deleted)	Drop table student (whole structure/ schema is deleted)	Truncate Student; (It deletes every row)		
Delete from Student Where Id=1;		(At once in go delete every row)		

Student	
id	name
1	A
2	B
3	C

Rollback Student X

Delete has an option to apply condition  
so its slow but truncate is fast

Constraints in SQL

(Condition or restrictions on DB)

Conditions are put upon columns.

① Unique (no duplicate values)

- 2) NOT NULL  $\Rightarrow$  User can't leave the column empty
- 3) (unique + Not Null) Primary key - The column should be unique as well as not null
- 4) Check - Before entering the value it checks the value.
- 5) Foreign key - Used to maintain referential integrity.
- 6) Default  $\Rightarrow$  In case no data, there is any value.

## SQL queries & Sub queries

Q1. Write a SQL Query to display max<sup>m.</sup> salary from Emp.

Q2. write a SQL query to display Employee name who is taking maximum salary .

i) Select MAX(salary) from Employee;

ii) Select Ename from Employee

where salary = (Select MAX(salary)  
from Emp);

Q3. Write a SQL to display the second highest salary from the table:

Select Max(Salary) from Employee

where salary <> (Select Max(Salary)  
from Employee);

Q4) Write a SQL query to display Employee name who is taking 2nd highest salary.

Select Ename from Employee

Where salary = (Select Max(Salary) from Employee where salary <> (Select Max(Salary)  
from Employee));

= → Only 1 value       $2 = 2$   
IN → for multiple values       $2 \in \{1, 2, 3, 4, 5\}$

## Group By (Solving by groups)

Q. Write a query to display all dept names along with no of emp working in that?

%P →

HR	2
MRKT	2
IT	1

group by  
↓  
aggregate  
func.

Select dept, count(\*) from Emp.  
Group by dept;

Having (It works as where in group by since where works on whole table)

Q. Write a query to display all the dept names where no. of employees are less than 2.

Select dept ~~no.~~ from Emp  
group by dept having count(\*) < 2;

%P → IT

Q. Write a query to display highest salary department wise name of emp who is taking that salary ..

Select E-name from Emp  
where Salary IN

(Select max(Salary)  
from Emp  
group by department);

## IN & NOTIN

Q

Emp			Project			
Eid	Ename	Address	b Eid	Pid	Fname	Location
1	Ravi	Chd	1	P1	IOT	Banglore
2	Varen	Delhi	5	P2	Big Data	Delhi
3	Nitin	Pune	3	P3	Retail	Mumbai
4	Robin	Banglore	4	P4	Android	Hyderabad
5	Ammy	Chd				

Find the detail of Emp whose address  
is either Delhi or chd or Pune;

Select \* from Emp

WHERE Address IN ('Delhi', 'Chd', 'Pune');

Not from Delhi & Pune

Project	SLP Project
DHL	

Select \* from Emp

where Address NOT IN ('DELHI', 'PUNE');

Q. find the name of Emp who are working on a project.

Select ename from Emp  
WHERE Eid IN

(Select eid from project)  
<sup>distinct</sup>

O/P  
Ravi  
Nitin  
Robin  
Ammy

EXIST AND NOT EXIST

Nested  
Correlated Queries

Nested Query  
Outer IN, NOT IN, ANY ALL

Inner

Correlated Query  
Outer exists, Not exists  
Inner

find the detail of emp who is working on atleast one project

Select \* from Emp

where Eid exists (Select Eid from Project  
where Emp.Eid = Project.Eid)

Top down approach

Row from outer  $\rightarrow$  In inner one

Inner query repeats every time till outer query.

SQL Aggregate functions  
(SUM, AVG(n), COUNT, MIN, MAX)

Emp				
Eid	Ename	Dept	Salary	
1	Ram	HR	10000	
2	Amit	MRKT	20000	
3	Ravi	HR	30000	
4	Nitin	MRKT	30000	
5	Varun	IT	50000	
6	Sandy	Testing	Null	

- ① Select MAX(salary) from Emp; O/p 50000
- ② Select MIN(salary) from Emp; O/p 10000
- ③ Select count (\*) from Emp;  
O/p: total no of tuples O/p  $\rightarrow$  6
- ④ Select count(salary) from Emp;  
O/p  $\rightarrow$  5 (doesn't counts null)
- ⑤ Select distinct (count(salary)) from Emp;  
O/p  $\rightarrow$  4
- ⑥ Select SUM(salary) from Emp;  
O/p  $\rightarrow$  140000
- ⑦ Select Distinct SUM(salary) from Emp;  
O/p  $\rightarrow$  100000
- ⑧ Select Avg (salary) from Emp;  
O/p  $= 140000/5$

$$\text{Distinct Avg} = \frac{\text{Distinct SUM}}{\text{Distinct Count}}$$

Q. Find the details of all employee who works in department.

Joins	Nested Subquery	Correlated Subquery
takes less time cross product + condition	takes more time Bottom up	takes more time Top down approach
attributes <pre>Select * from Emp</pre> dept where WHERE eid IN emp.eid = dept.eid	<pre>Select * from Emp</pre> (Select eid from dept)	<pre>Select * from emp</pre> where EXISTS (Select eid from dept WHERE emp.eid = dept.eid);

Emp

Eid	Name
1	A
2	B
3	C
4	D
5	E

Dept

Dept no	Name	Eid
D1	IT	1
D2	HR	2
D3	MRKT	3

find the N<sup>th</sup> highest salary using SQL.

Select id, salary from Emp e1  
where  $N-1 = (\text{Select count(Distinct salary)}$   
from Emp e2  
where e2.salary > e1.salary);

## Basic PL/SQL Execution Part 1

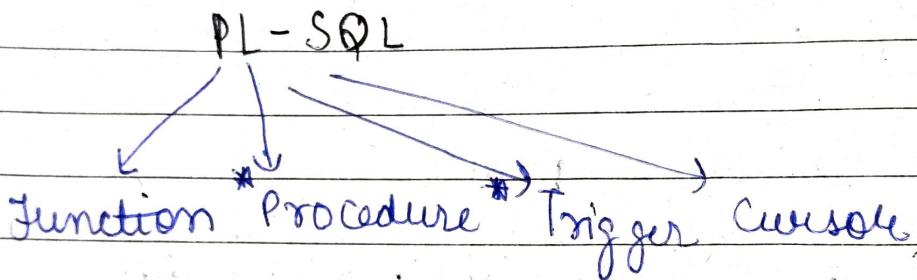
Program 1: find the sum of 2 numbers

```
declare
    a int;
    b int;
    c int;
begin
    a := &a;    J live server doesn't work but
    b := &b;    on system server does)
    c := a + b;
    dbms_output.put_line ('sum of a and b=' || c);
end;
```

Program 2: Greatest of 2 numbers.

```
declare
    a int;
    b int;
begin
    a := &a;
    b := &b;
    if (a > b)
        then
            dbms_output.put_line ('a is greater' || a);
        else
            dbms_output.put_line ('b is greater' || b);
        end if;
    end;
```

Emp e1		Emp e2	
Id	Salary	Id	Salary.
1	10	1	10
2	20	2	20
3	20	3	20
4	30	4	30
5	40	5	40
6	50	6	50



SQL → declarative (what to do)

PL SQL → procedural (what to do + How to do)

Block      Declaration      a int

Executable      begin  
code.      a := 10;  
                  end;

Exception  
handling (error)

## PL-SQL (while, for loop)

for loop

Declare

```
a number(2)
begin
for a in 0..10
loop
dbms_output.put_line(a);
end loop;
end;
```

while loop

declare

```
a int;
b int;
begin
a:=0;
b:=2b;
while(a < b)
loop
a:=a+1;
dbms_output.put_line(a);
end loop;
end;
```

Single row & Multi row  
functions in SQL

↓  
single row  
(single row  
single o/p)

↓  
Multi row  
(multiple row  
single o/p)

## single Row function.

### A) Number func<sup>n</sup>.

- 1) Round
- 2) Truncate
- 3) Mode.

### B) Character functions.

#### Case manipulation

- 1) lower
- 2) upper
- 3) init cap

#### Character manipulation

- 1) Concat
- 2) Substr
- 3) length
- 4) Instr
- 5) LPAD/RPAD
- 6) TRIM
- 7) REPLACE

### C) Date function

- 1) Month between
- 2) Add months
- 3) NEXT\_DAY
- 4) LAST\_DAY
- 5) ROUND
- 6) TRUNC

### D) General functions

- 1) NVL
- 2) NVL2
- 3) NULLIF
- 4) COALESCE

5) CASE

6) DECODE

5) Datatype conversion

- 1) TO\_CHAR
- 2) TO\_NUMBER
- 3) TO\_DATE

Multirow function

A) Aggregate

- 1) SUM
- 2) MAX
- 3) MIN
- 4) COUNT
- 5) AVG

Various Char functions

Case function manipulation.

1) lower

Select lower(name) from emp;

2) Upper

Select upper(name) from emp;

3) Init cap.

Select initcap(name) from emp;

Algorithm	QUESTION
Date	

## ⑥ character manipulation.

1) Concat ('Varun' 'Singla') Varun Singla.

Select concat(id, name) from emp;

2) Substr ('Varun', 2, 4) aeu.

Select substr(name, 2, 5) from emp;

3) Instr ('Varun', 'u') 4th.

Select instr(name, 'T') from emp;

4) Length ('Varun') 5.

Select length(name) from emp;

5) Lpad ('Varun', 10, '\*') \*\*\*\*\* Varun

Select lpad(name, 15, '\*') from emp;

6) Rpad ('Varun', 10, '\*') Varun \*\*\*\*\*

Select rpad(name, 15, '\*') from emp;

7) Trim ('V' from 'Varun')

Select trim('V' from name) from emp;

8) Replace ('Varun', 'V', 'T')

## Transaction And Concurrency Control:

Transaction: Transaction ~~of~~ is a collection of related Read & Write oper<sup>n</sup>s used to perform some related task.

R(A)

$$A = A + 500$$

W(A)

R(B)

$$B = B - 500$$

W(B)

Commit

R(A) - Read oper<sup>n</sup>

W(A) - write oper<sup>n</sup>.

## ACID

Atomicity - (All or nothing)  $\rightarrow$  Either execute all the oper<sup>n</sup>'s of the transaction or none of them

Component of Database Responsible for it:

- 1) Recovery Management Component  $\rightarrow$  It takes care that either the atomicity is applied else rollback is done in case of error arises before commit.  
How?
- 2) It maintains a log file till commit.

2) Consistency  $\rightarrow$  (No violation of ~~data~~ integrity constants)

Rule  $\rightarrow$  "If the Database was consistent before a particular transaction, then it should also be consistent after that execution of the transaction."

Who is responsible?

$\rightarrow$  It is user's / programmer's responsibility

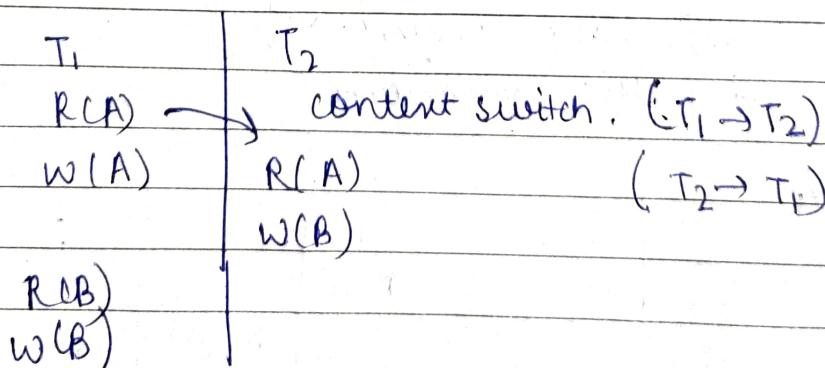
How does it do?

By setting various integrity.

3) Isolation

Concurrent changes are invisible

Rule  $\rightarrow$  "Concurrent execution of 2 or more transactions should not cause any inconsistency. It should be as if the transaction executed independent of others."



4) Durability - (the committed update persists)  
 The effect of a completed or committed transaction  
 should be persist even after a crash.

It means once a transaction commits, the system must guarantee that the result will never be lost.

How?

- 1) By ensuring Recoverability
- 2) By using RAID → It keeps multiple copies of information in various databases.

R → Redundant

A → Array of

I → Independent

D → Disk

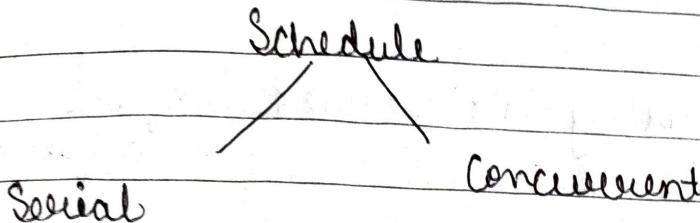
## Serializability : Basics & Classification

A Time order sequence of 2 or more transactions

Ex1: S1

T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>	T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>
R(A)			R(A)		
	R(A)			R(A)	
		R(A)			R(A)
w(B)			w(B)		
				w(B)	
					w(B)
			commit	commit	commit

$S_2 \rightarrow$  complete schedule (commit operation)



If only after the commit of one transaction, the other transaction begins then the schedule is said to be serial.

A schedule consisting of interleaved execution of 2 or more transaction simultaneously is called concurrent schedule.

It satisfies all ACID prop.

$T_1 (R(A))$

$T_1 (W(A))$

Commit  $T_1$

$T_2 (R(B))$

$T_2 (W(B))$

Commit  $T_2$

$T_1$ $R(A)$	$T_2$ $R(A)$	$T_3$ $R(A)$
	$w(B)$	
		$w(B)$
		Commit

### Advantages

→ No inconsistency

### Disadvantages

→ Poor throughput

→ Poor resource utilization

### Advantages

- Better throughput
- Better resource utilization

### Disadvantages

→ Inconsistency

soln to the serialization?  
 → serializability.

Serializability : Procedure & Conflict  
 serializability

The problem of Inconsistency may arise  
 in case of concurrent schedule.

soln → serializability.

Convert the concurrent schedule into an equivalent schedule which has a "serial order execution of its constituent transactions".

Concurrent schedule → equivalent serializable  
 Schedule  
 ↓  
 is consistent

How to convert concurrent schedule to consistent concurrent schedule

- 1) conflict serializable
- 2) view serializable.

Conflict serializable.

A schedule is said to be "conflict serializable" if one of its conflict equivalent schedule is serializable.

Page No. \_\_\_\_\_  
Date \_\_\_\_\_

Conflict Equivalent schedule  $\Rightarrow$  If  $s'$  is a schedule formed after swapping the non conflict pairs in a given schedule  $s$ , then  $s$  &  $s'$  are called equivalent schedule.

### Conflict pairs

R(CN)	W(CA)	W(CA)	R(A)	W(A)	W(CA)
-------	-------	-------	------	------	-------

O/P

A pair which ~~can't~~ change if the order of the execution of these conflict pair is inverted.

### Non conflict pairs

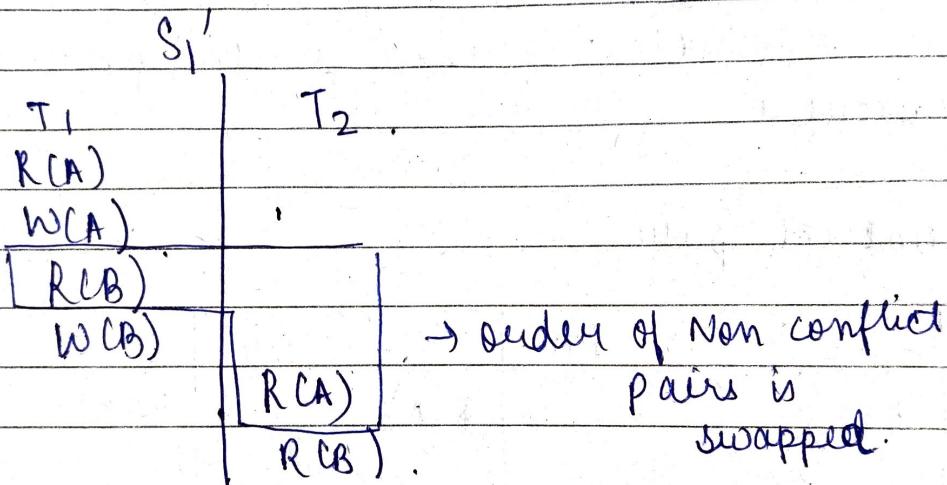
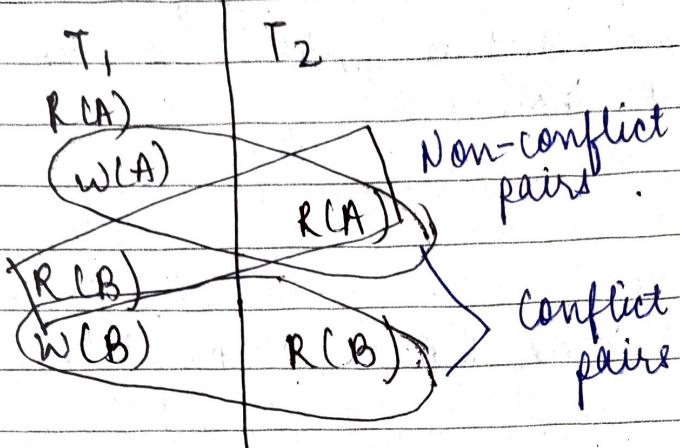
R(A)	R(A)	W(A)	W(CB)
R(CB)	R(A)		

R(CA)	W(CB)
-------	-------

We can never swap the order of conflict pair. As by doing so, the final result changes & the schedules no more remain equivalent.

# Page No. \_\_\_\_\_ Date \_\_\_\_\_

## finding a conflict equivalent schedule



↙ Order:  $T_1 \rightarrow T_2$ .

Conflict Equivalent Schedule.

Precedence Graph.

Example 1:  $R_1(x) R_1(y) R_2(x) R_2(y) W_2(y) W_1(x)$

$R_1(y) \rightarrow w_2(y)$



$R_2(x) \rightarrow w_1(x)$

Cyclic

Hence

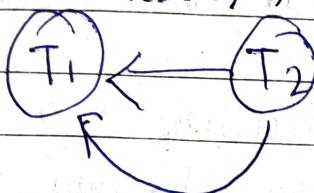
non-conflict

serializable.

Ex-2

$R_1(x) \quad R_2(x) \quad w_2(y) \quad R_1(y) \quad w_1(x)$

$R_2(x) \rightarrow w_1(x)$



$w_2(y) \rightarrow R_1(y)$

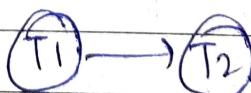
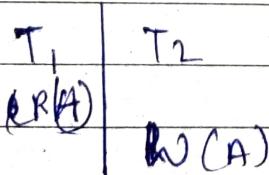
Acyclic : conflict serializable

Order:  $T_2 : T_1$

Precedence graph method.

node  $\rightarrow$  transactions

edges  $\rightarrow$  various conflicts



Acyclic : conflict serializable

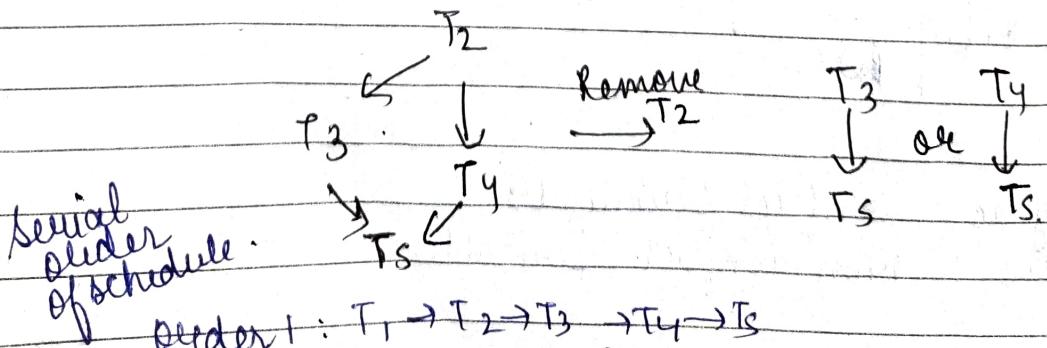
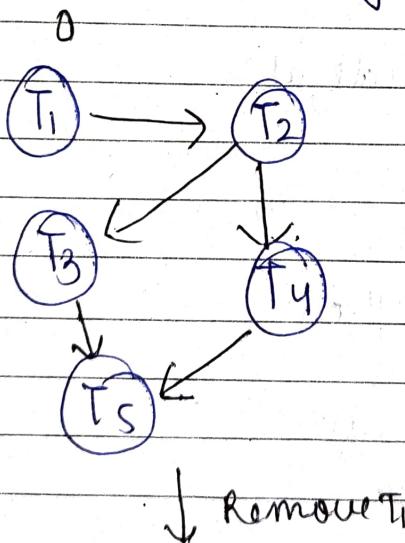
Cyclic : Non-conflict serializable

In case the graph is Acyclic

- 1) It is conflict serializable

We find the serial execution Order as follows.

- 1) Remove the node with an Indegree 0 & also remove all the edges corresponding to it
- 2) Repeat Step 1 till all the Nodes are covered.
- 3) The final order of execution of transactions is the order in which the nodes are removed from the graph.



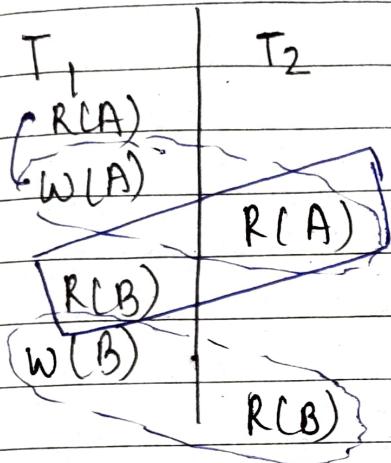
order 1 :  $T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4 \rightarrow T_5$

order 2 :  $T_1 \rightarrow T_2 \rightarrow T_4 \rightarrow T_3 \rightarrow T_5$

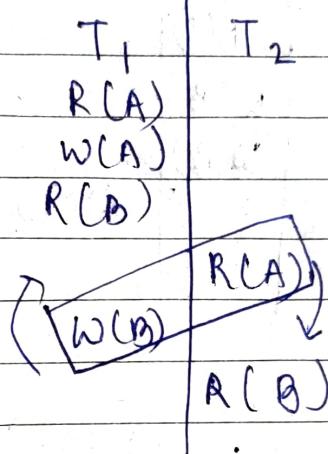
## A Tabular form Example

Schedule

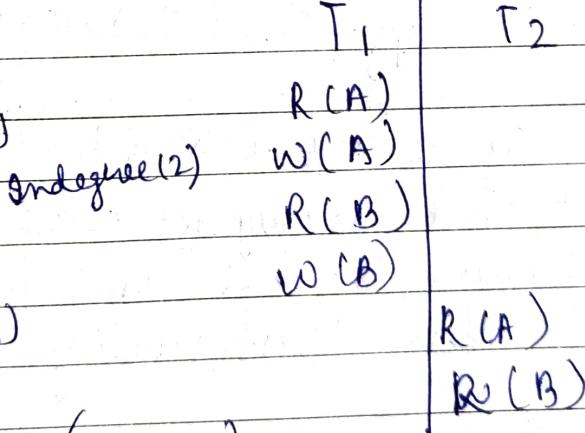
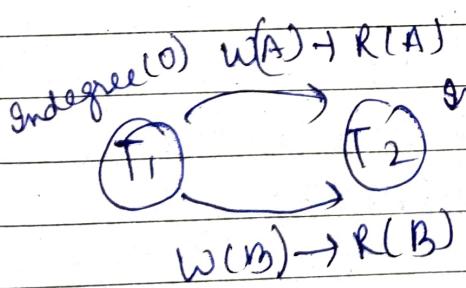
$S_1$



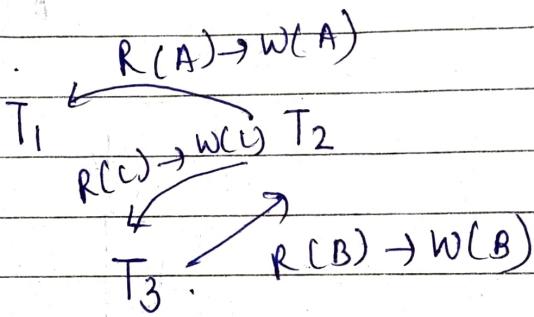
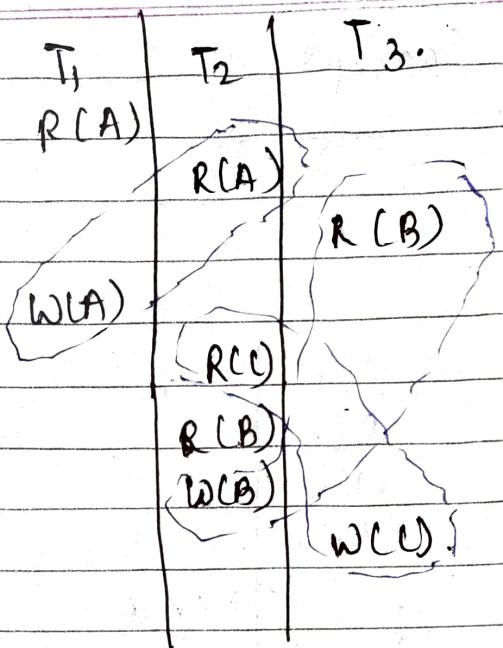
conflict pair



Order  $T_1 \rightarrow T_2$



Acyclic  $\therefore$  Order  $(T_1 \rightarrow T_2)$

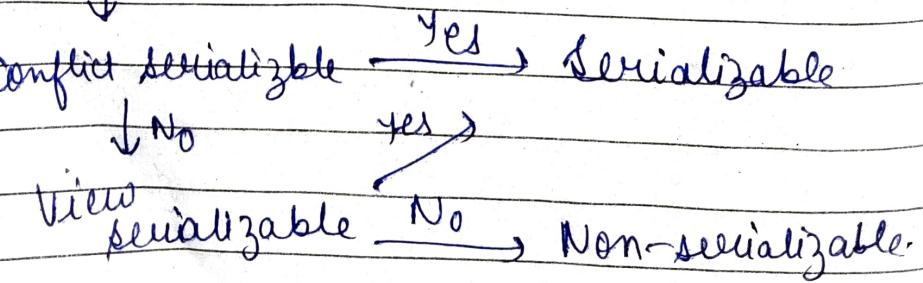
S<sub>1</sub>

Cycle  $\rightarrow$  Non serializable.

Through Conflict we only get to know it is non conflict serializable not non serializable.

View Serializability

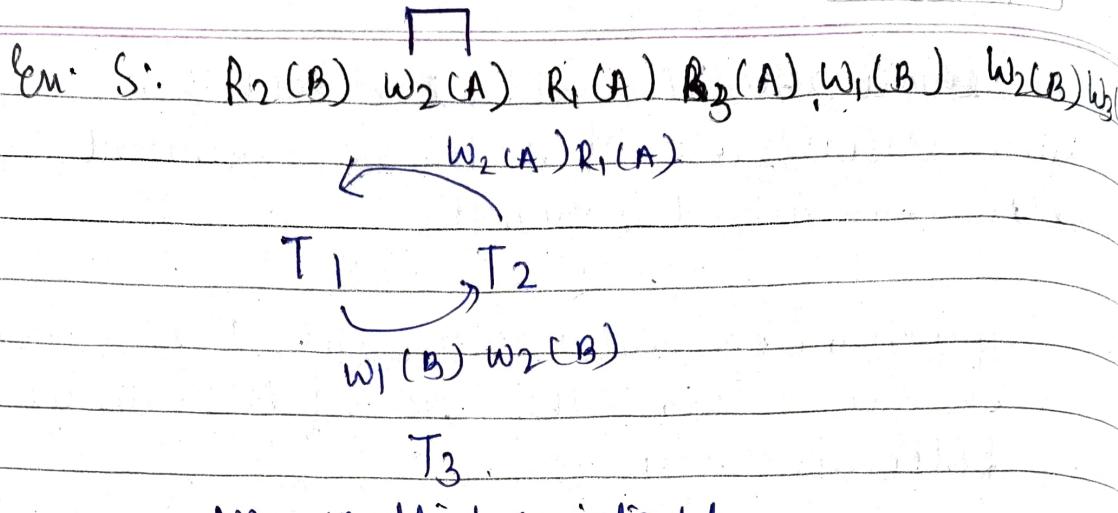
Serializability



**View serializable :** A schedule is said to be view serializable, if it has an equivalent view equivalent schedule.

**View equivalent schedule:** A schedule  $S'$  is said to be view serializable so if it satisfies the following 3 cond'n.

- (1) **Initial Read**  $\rightarrow$  If  $T_1, T_2, T_3 \dots T_j$  transaction perform initial read (i.e. Read the values of data items directly from the database before any modification (write operation) on the data items A, B, C ... J respectively in a schedule  $S$  then none of these Initial Reads should be violate in its view equivalent schedule  $S'$ .
- (2) **Read write sequence** - The Read write conflict pairs should be in the same sequence in both the view equivalent schedules.
- (3) **Final update**  $\rightarrow$  If  $T_i$  is a transaction that updates a data item (X) finally (at the end) in schedule  $S$ , then  $T_i$  should only update finally the data item X in the view equivalent schedule  $S$ .



Non conflict serializable

there is  
a write first

	A	B	
Initial Read	X	$T_2$	$T_2 \rightarrow T_3$
Final update	$\bullet T_2$	$T_3$	$T_2 \rightarrow T_1 \rightarrow T_3$
Update	$T_2$	$T_1, T_2, T_3$	

Order  $T_2 \rightarrow T_1 \rightarrow T_3$

View Serializability : A tabular form

$T_1$ $R(A)$	$T_2$	$T_3$
	$R(A)$	$R(B)$
$W(A)$		
	$R(L)$ $R(B)$ $W(B)$	$W(L)$

$R_1(A)$   $R_2(A)$   $R_3(B)$   $w_1(A)$   $R_2(C)$   $R_2(B)$   $w_2(B)$   
 $w_3(C)$

	A	B	C
Initial	$T_1, T_2$	$T_3, T_2$	$T_2$
read			
<del>update</del>	$T_1$	$T_2$	$T_3$
final update	$T_1$	$T_2$	$T_3$

$$A : T_2 \rightarrow T_1$$

$$B : T_3 \rightarrow T_2$$

$$C : T_2 \rightarrow T_3$$

conflicting

So, no order can be there. So it is non-serializable.

$R_1(A)$   $R_2(A)$   $R_3(B)$   $w_1(A)$

$R_2(C)$   $R_2(B)$   $w_2(B)$   $w_1(C)$

$T_1$ $R(A)$	$T_2$ $R(A)$	$T_3$ $R(B)$	Initial Read	A	B	C
$w(A)$			Update	$T_1, T_2$	$T_3, T_2$	$T_2$
				$T_1$	$T_2$	$T_1$
			Final Update	$T_1$	$T_2$	$T_1$
$w(C)$						

$$A : T_1 \rightarrow T_2$$

$$B : T_3 \rightarrow T_2$$

$$C : T_2 \rightarrow T_3$$

Combining all orders

$$T_3 \rightarrow T_2 \rightarrow T_1 \quad T_3 : T_2 : T_1$$

Grade 2012 (Serializability)

$T_1$  : Read ( $P$ )

Read ( $Q$ )

If  $P = 0$  then  $Q := Q + 1$ ;

Write ( $Q$ )

$T_2$  : Read ( $Q$ )

Read ( $P$ )

If  $Q = 0$  then  $P := P + 1$ ;

Write ( $P$ );

Any non-serial interleaving of  $T_1$  &  $T_2$ ?

$T_1$

$R_1(P)$

~~$R_1(Q)$~~

$W_1(Q)$

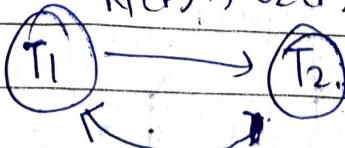
$T_2$

$R_2(Q)$

~~$R_2(P)$~~

$W_2(P)$

$R_1(P) \rightarrow W_2(P)$



loop is occurring

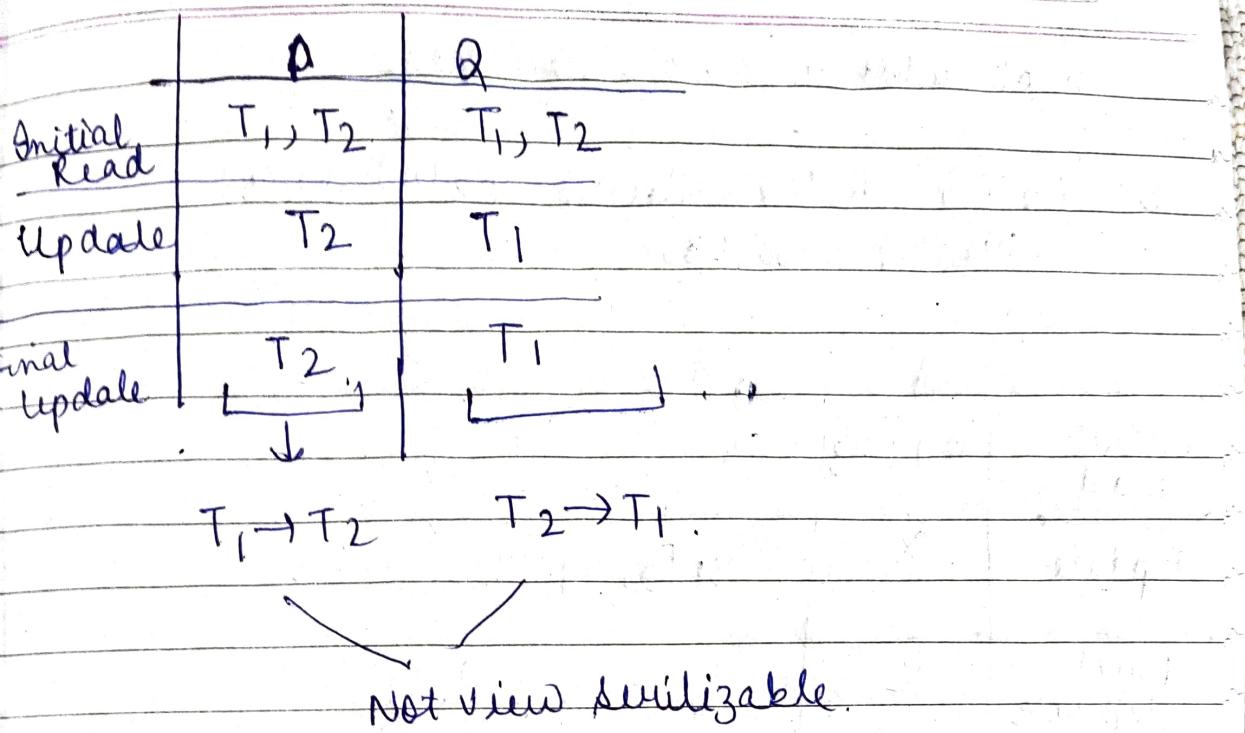
$W_1(Q) \rightarrow R_2(Q)$

non conflict

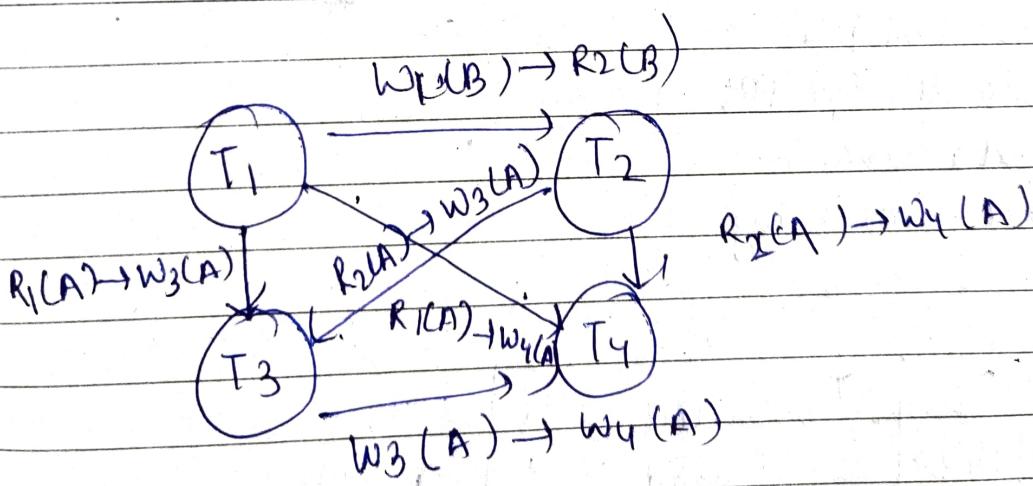
serializable.

$R_1(P) R_4(Q)$

Page No.	5/10
Date	10/10/2023



$Q:$   $(R_2(A), R_1(A), W_1(L), R_3(C), W_1(B), R_4(B))$   
 $(W_3(A), R_4(C), W_2(D), R_2(B), W_4(A))$   
 $W_4(B)$



Acyclic graph  $\rightarrow$  Serializable

order:  $T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4$ .

Check whether view serializable.

S:  $R_2(B), R_2(A) \quad R_1(A) \quad R_3(A) \quad W_1(B)$   
 $W_2(B) \quad W_3(B)$

	A	B	
Initial Read	$T_2, T_1, T_3$	$T_2$	→ ①
update	X	$T_3, T_2, T_1$	→ ②
final update	X	$T_3$	→ ③

since no transaction  
is performing any  
write operatn.

By ① & ② & ③

on Data item "A"  
so we can just

$T_2 \rightarrow T_1 \rightarrow T_3$

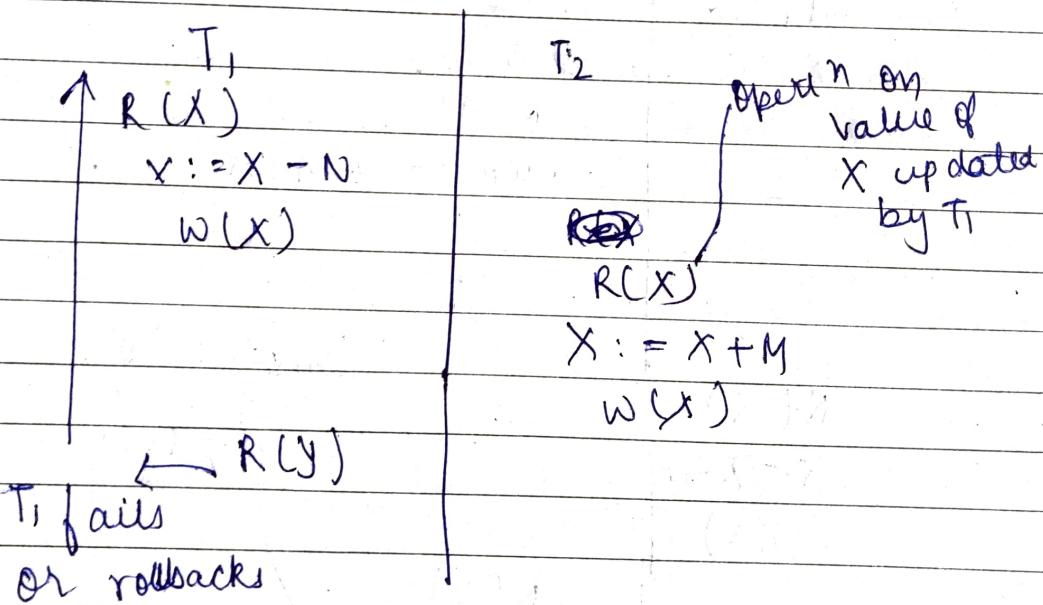
ignore the operatn.  
On data item A  
for the time being

S		S'	
$T_1$	$T_2$ $R_2(B)$ $R_2(A)$	$T_3$ Initial read.	$T_1$
$R_1(A)$	$R_3(A)$	$R_1(A)$ $W_1(B)$	$T_2$ $R_2(B)$ $R_2(A)$ $W_2(B)$

Q. Check whether the schedule given is view serializable or not.

### Dirty Read Problem (W-R problem)

This problem occurs when one transaction updates a database item & then the transaction fails for some reason. Meanwhile, the update item is accessed (read) by another transaction before it is changed back (or roll back) to its original value.



So, update by  $T_1$  was temporary ( $W(x)$ ). If the opern's performed by  $T_2$  on this temporary value should now be rollbacked or reversed back.

## The Incorrect Summary Problem.

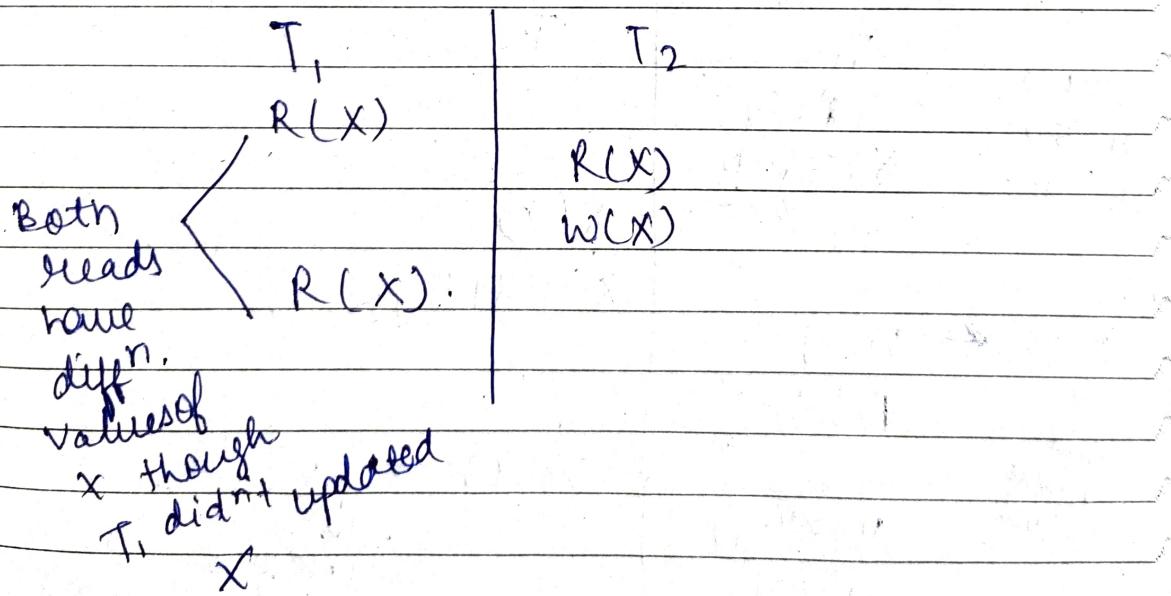
If one transaction is calculating an aggregate summary function on a number of database items while other transactions are updating some of these items, the aggregate function may calculate some values before they updated & others after they are updated. So, the aggregate sum in such a case would contain a wrong value.

$T_1$	$T_2$
	Sum := 0
$R(A)$	
	Sum := Sum + A
	:
$R(X)$	
$X := X - N$	
$W(X)$	
	Computes sum using value updated by $T_1$
	)
$R(Y)$	
$Y := Y - N$	
$W(Y)$	
	$R(X)$
	Sum := Sum + X
	$R(Y)$
	Sum := Sum + Y
	Old
	$\downarrow$
	New value $\rightarrow$ Sum has incorrect value.

## The Unrepeatable Read Problem

Let's suppose a transaction T reads the same item twice & the item is changed by the another Transaction T' between the two reads. Hence T receives diff<sup>n</sup> value for its 2 reads of the same item.

Example: During an airline reservation system, transaction a customer enquires about seat availability on a particular flight, the transaction then reads the no. of seats on that flight a second time before completing a the reservation & it may end up showing a diff<sup>n</sup> value.



## Problems due to concurrent Transaction

- 1) lost update (WW)
- 2) Unrepeatable Read (RW)
- 3) The incorrect summary. (due to opertn of aggregate func)
- 4) Dirty Read (WR)  
Temporary

Lost update Problem (Some update is lost)  
(write)

This problem occurs when 2 transactions that access the same database items have their operatn's interleaved in a way that makes the value of some database item in correct

$T_1$	$T_2$
$X := 100$	
$N := 50$	
$M := 20$	
$R(X)$	
$X := X - N$	
$N := 50$	
$X = 50$	
$\leftarrow W(X)$	
$J_1$	
$R(Y)$	
<i>This value is lost</i>	
$Y := Y + N$	
$W(Y)$	
	$R(X)$
	$X := X + M \Rightarrow X = 120$
	$W(X) \rightarrow 120$
	$J_2$
	item $X$ has an incorrect value.

Page No.	51
Date	

## Recoverability

recoverable



Change can be

undo



Previous state

can be restored

irrecoverable



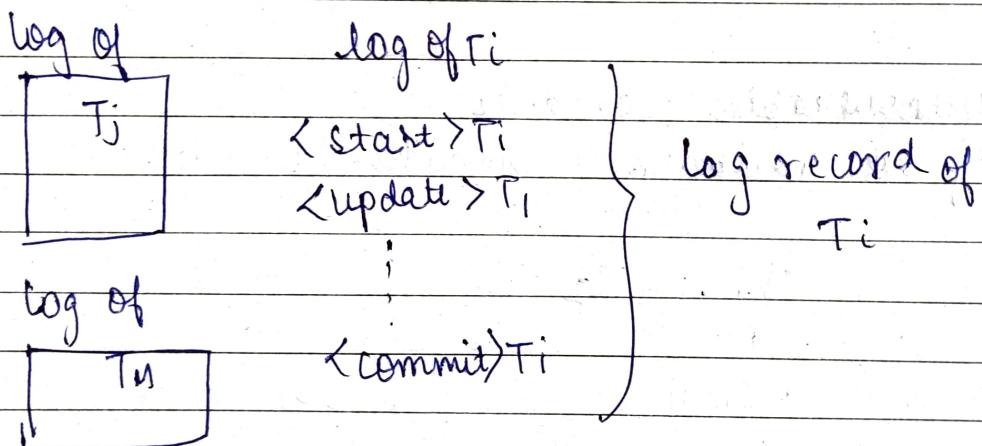
Changes can't be undone



previous state

can't be restored

Procedures: When a transaction (say  $T_i$ ) executes, there is always a transaction log created corresponding to it.



Update Example

$\langle T_0, X_j, v_i, v_j \rangle$

$\langle T_0, A, 100, 200 \rangle$

when Roll backs previous value of Ti  
is stored.

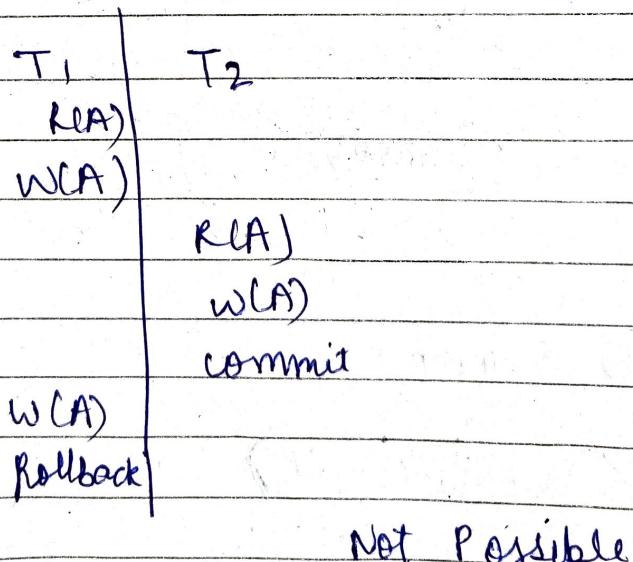
After a commit, there won't be rollback.  
It makes the schedule recoverable as  
the log is deleted after schedule commit.

### Schedule On the basis of Recoverability

- 1) Irrecoverable Schedule.
- 2) Cascading Rollback (Cascading Recoverable)
- 3) Strict Recoverability (Third level of recoverability).

#### Irrecoverable Schedule

If we rollback a committed transaction



Not Possible

Cascading Rollback  $\rightarrow$  It is a type of Rollback when, because of the Rollback of 1 transaction, rollback of many other transactions is caused.

T <sub>1</sub> W(A)	T <sub>2</sub> R(A) W(A)	T <sub>3</sub> R(A) W(A)	T <sub>4</sub> R(A) W(A)
Rollback.			

Rollback of T<sub>1</sub> causes Rollback of all T<sub>2</sub>, T<sub>3</sub>, T<sub>4</sub>.

Sol<sup>n</sup>  $\rightarrow$  Transaction which completes first should commit first.

T <sub>1</sub> W(A) commit	T <sub>2</sub> R(A) W(A) commit	T <sub>3</sub> R(A) W(A) commit	T <sub>4</sub> R(A) W(A) commit

Disadvantage  $\rightarrow$  unnecessary wastage of CPU time.

Problem of (WR) has been removed

Remaining problem

$\rightarrow R\ W$   
 $\rightarrow W\ W$ .

Shortcut  $\rightarrow$  To check if a scheduling is Cascading Recoverable.

1) Check for WR cond<sup>n</sup>, If no WR problem exists, it is cascading recoverable schedule.

3) Strict Recoverability.

The dependent transaction should <sup>read</sup> or write only after the commit of the independent Transaction  $T_1$ , here.

Independent  
transaction  
 $\rightarrow T_1$   
 $R(A)$   
 $W(A)$   
Commit

$T_2$  & Dependent  
transaction  
 $R(A)/W(A)$



So  $T_2$  show R/W only after  $T_1$

Advantages

- WW also remove as well as WR.
- R-W still left

### Types of Schedule

Recoverable

Cascading recoverable (level 0)

Cascadeless recoverable (level 1)

Strict recoverable (level 2)

Irrecoverable

Cascading Rollback → When Rollback of 1 transaction causes the Rollback of some other transactions, then the phenomena is called as Cascading Rollback.

Imp Note → All transactions haven't performed their rollback.

→ The phenomena of Cascading Rollback occurs in some recoverable schedules where an uncommitted transaction has to rolled back because it Read an item from a transaction that failed.

### Cascadeless Schedule

A schedule is said to be cascadeless, if every transaction in the schedule reads only item that were written by committing transactions.

En.	T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>	T <sub>4</sub>
	W(A)			
		R(A)		
		W(A)		
			R(A)	
			W(A)	
				R(A)
				W(A)
	T <sub>1</sub> Abort			

T <sub>1</sub>	T <sub>2</sub>
R(A)	
W(A)	
Commit	
Rollback	R(A)

The problem of (RW) Dirty Read is removed.

Check if ~~Rea~~ cascadeless Recoverable/Irrecoverable.

(WR)

$R_1(x) R_2(z) R_1(z) R_3(x) R_3(y) W_1(x) W_3(y) \boxed{R_2(y)}$   
 $W_2(z)$

Step 1

Step 1 → Checking if Recoverable/ ~~Irrecoverable~~ recoverable.  
Transactions are committing in the same order as they are completing their work.

Step 2: → Check if RW problem is there.

$W_3(y) R_2(y)$ , therefore it is not cascadeless recoverable.

Strict schedule → cascadeless

Cascadeless schedule → recoverable.

Check for strict Recoverability

$R_1(x) R_2(z) R_3(x) R_1(z) R_2(y) R_3(y) W_1(x)$   
~~W\_2(z)~~  $W_3(y) W_2(y) C_3 C_2$

I) Level of completion | commit there

$C_1$

$C_3$

$C_2$

$C_1$

$C_3$

$C_2$

Recoverable