

## 1. ER-MODEL

### 1. INTRODUCTION TO DBMS

Data: Any fact that could be Recorded and stored (Text, Numbers, Image)

Database: collection of Related data.

⇒ The database that contains text and Numbers is called Traditional Database.

⇒ Real-time Databases: Supermarkets, Firms.

⇒ "Data warehouse" contains large amount of Data, the data is going to be historical.

⇒ Now a days the Databases are computerised and there must be some software that defines, constructs, Manipulate the Databases.

Now, the software that performs above operations on the Database is called "Database Management Systems".

⇒ DB + DBMS = DATA BASE SYSTEMS.

### 2. MODELS IN DBMS

⇒ The various models that are used when designing the database is

1. High Level or Conceptual Models ⇒ NAIVE USERS ⇒ Diagrams  
↓  
ER-model.

2. Representational / Implementation Model ⇒ used by programmers.  
(Tables).

3. Logical Level / physical Data models ⇒ Structure, Datatype.

### 3. INTRODUCTION TO ER-MODEL

ER Model = Entity - Relationship Model. (Entity, Attributes, Relationships)

ENTITY: Any object in our Database.

ATTRIBUTES: The things that describe the Entities are called Attributes.  
(Properties that are used to describe Entities better).

RELATIONSHIPS: Association among entities.

⇒ Entity type = Schema = Heading = PERSON(age, Name, Ad<sub>d</sub>)

⇒ Entity = (26, Raju, ..) = Extension

⇒ In the ER Model we use Entity types but not the Entities.

#### 4. ATTRIBUTES

⇒ Attributes are useful in order to describe the entities better.

The Attributes are mainly classified into

- 1) Simple Attributes (vs) Composite Attributes.
- 2) Single valued (vs) Multi valued Attributes.
- 3) Stored (vs) Derived Attributes
- 4) Complex Attributes.

PERSON  
Name = SurName, FirstName, MiddleName, LastName (Composite Attribute)

Age = Single valued Attribute (also called Scalar Attribute)

PNO = Multi valued Attribute

DOB = Stored Attribute (Derives)

Age = Derived Attribute  
Address = Complex Attribute

Composite Attribute

Multi valued Attribute.

#### 5. RELATIONSHIPS (4-M)

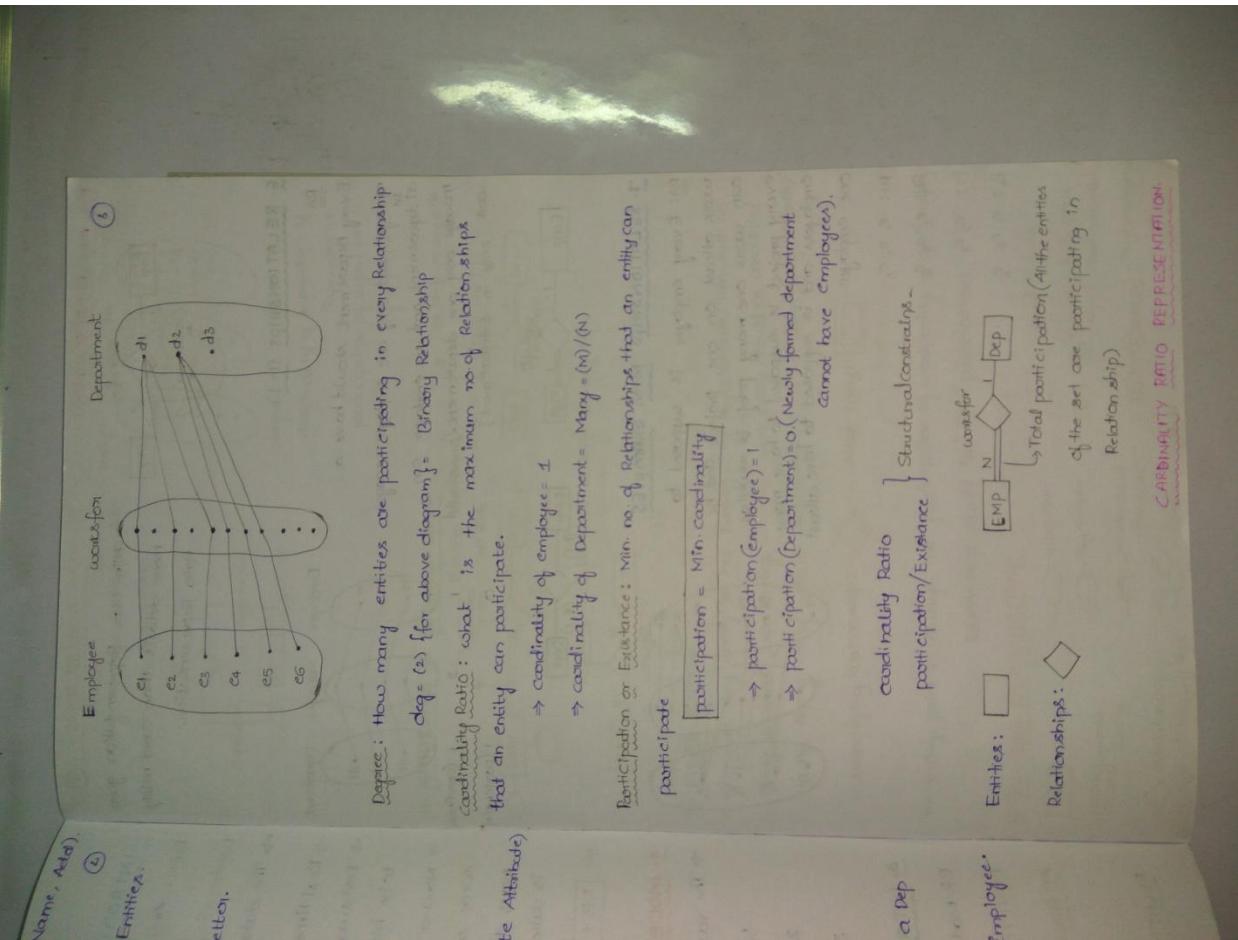
⇒ Relationship is nothing but Association among entities.

Requirement Analysis : 1. Every employee works for a dep and a dep exactly can have many employees

2. New department need not have any employee.

Entities

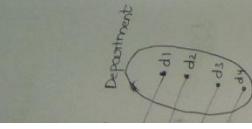
Relationships



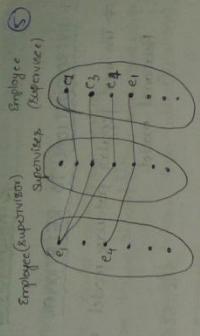


#### ④ relation gives cardinality

RA: Every employee is supposed to have exactly one supervisor.



#### ⑤ RECURSIVE RELATIONSHIPS



Degrec = 2

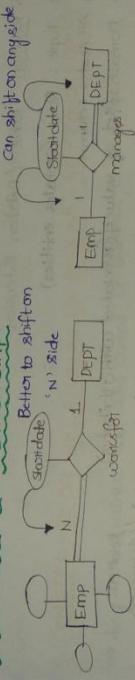
Cardinality of supervisor = N

Supervisee = 1

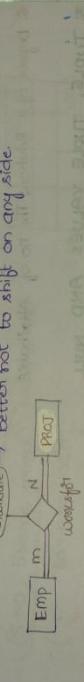
participation of supervisor = 0

of supervisor = 0

#### ⑥ ATTRIBUTES TO RELATIONSHIPS



Bottom not to shift on any side



Bottom to shift on any side

Can shift on any side

Bottom to shift on any side

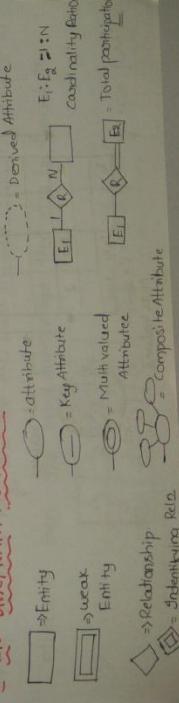
Bottom to shift on any side

⇒ whenever any entity is not having any key attribute then such an entity is called weak entity, so the weak entity should be associated to the strong entity with a relationship called as "Identifying Relationship".

Identifying Relationship: Weak entity:

⇒ v. imp point: Always the participation of weak entity in an identifying relationship should be "total participation": on both sides of relationship

#### ⑦ ER-DIAGRAM NOTATIONS



= Derived Attribute

= Attribute

= Entity

= Weak Entity

= Entity

= Relationship

= Multi-valued Attribute

= Composite Attribute

= Identifying Relation

= Total participation

## 2. RELATIONAL DB MODEL

## 4. CONSTRAINT

1. INTRODUCTION TO RELATIONAL DATABASE
  - ⇒ The most popular database model used at representational level is Relational model.

ORACLE, IBM → ER DIAGRAM  
SQL, Sybase → SQL

- ⇒ We can view

## 3. TERMINOLOGY OF RELATIONAL DATABASE

1. Relation: Table / Relation Extension.
2. Tuple: Row (Contains entities)
3. Attribute: Column
4. Domain: Set of values (Associated with attributes)
5. Relational schema: Heading of the table = R(A<sub>1</sub>, A<sub>2</sub>, A<sub>3</sub>, A<sub>4</sub>, A<sub>5</sub>) / Relation
6. Degree of a Relation: The no of Attributes

## 3. TUPLE, TUPLE VALUES

- ⇒ whenever we store a relation in a memory then it is stored in particular order.
- ⇒ In a Relation no two tuples can have the same values in all the attributes. (No duplicate values).
- ⇒ some values of the attributes are not specified/present then we use "null" in that field.
- ⇒ For Example: Name Comprises of first name, middle name, last name and now a person might not have middle name so middle name=NULL

a	1	2013	2
a	1	2013	2
b	2	2014	NULL

Null values.

## 5. CONSTRAINT

⇒ key - constraint  
(SNO, SNAME, RA, RNAME)

- ⇒ SUPER KEY
- ⇒ Any Min. value
- ⇒ Every Rec
- ⇒ If All is attribute

⑥

#### 4. CONSTRAINTS ON RELATIONAL DB SCHEMA - DOMAIN CONSTRAINTS

- ⇒ Domain constraints ⇒ Entire schema should be Atomic.
- ⇒ Key constraints ⇒ No two tuples should have same value.
- ⇒ Entity Integrity constraints ⇒ Entire tuple should follow some constraints.

⇒ Referential Integrity constraints ⇒ Applied between two tables(Relations)

- ⇒ We can view a Relation as "Flat file structure".
- | X | S.NO | Name X |
|---|------|--------|
| ✓ | SNO  | FN     |
|   |      | MN     |
|   |      | LN     |
- Should be Atomic.

Composite, Multivalued attributes are not allowed in Relations.
---

#### 5. CONSTRAINTS ON RELATIONAL DB SCHEMA - KEY CONSTRAINTS

- ⇒ Key-constraints are also called "Oneness constraints".

(S.NO, Sname, marks)	→ SUPER KEY
(1, Ravindra, 100)	

Student Marks

t <sub>1</sub> →	100
t <sub>2</sub> →	100
	100

- ⇒ [SUPER KEY ⊆ Attributes] for which no two tuples have the same values in all the attributes.

KEY = MINIMAL SUPERKEY

- ⇒ Any minimal superkey is a "Key". (SNO) because
- ⇒ Any minimal superkey is a "Key". (SNO, Sname) because
- ⇒ Every Relation is going to have a superkey by default and that superkey is "set of all Attributes".
- ⇒ Any superset of a key is a superkey.
- ⇒ If A<sub>1</sub> is a key, and the set/table/Relation contains (A<sub>1</sub>, A<sub>2</sub>, A<sub>3</sub>, A<sub>4</sub>) attributes, then the no. of superkeys that can be formed is

1	2	3	4
①	②	③	④

$1 \times 2 \times 3 \times 4 = 8$  Superkeys are possible

→ If we are going to have two keys for a Relation then they are

going to be candidate keys. (or) If we have two minimal super keys for a Relation then they are called "Candidate Keys".

→ one of the keys of candidate keys is chosen and it is going to play some important role while we insert some numbers and that key is called "Primary Key".

$(A_1 A_2 A_3 A_4)$  - SK  
 $(A_1 A_2 A_3 A_4)$  - SK and key  
 $(A_2 A_3 A_4)$  - SK and key  
 $(A_3 A_4)$  - SK and key

$(A_1 A_2 A_3 A_4)$  -  
 $(A_1 A_2 A_3 A_4)$  -  
 $(A_1 A_2 A_3 A_4)$  - SK and key

$(A_1 A_2 A_3 A_4)$  -  
 $(A_1 A_2 A_3 A_4)$  -  
 $(A_1 A_2 A_3 A_4)$  - SK and key

$(A_1 A_2 A_3 A_4)$  -  
 $(A_1 A_2 A_3 A_4)$  -  
 $(A_1 A_2 A_3 A_4)$  - SK and key

"primary key" doesn't allow "NULL" values.

## 6. ENTITY AND REFERENCE INTEGRITY CONSEQUENCES

⇒ Entity Integrity says that no prime attribute should have null value

Employee → Employee KEY  
Department → Department KEY

ENO	ENAME	DNO
1	a	4
2	b	1
3	c	2

ENO	ENAME	DNO
1	a	4
2	b	1
3	c	2

⇒ Foreign Key

1. ACTIONS UP

→ The Actions

+ If the also to select a  
⇒ while "delet"  
and the a

1. Minim superkeys
2. COUNTIN
3. Given a

Given a

Minim superkeys  
COUNTIN  
Given a

Given a

Minim superkeys  
COUNTIN  
Given a

• they are → Foreign key can have "NULL" values unlike primary key

Super key

⑧

going to  
and that key

⑨

### ACTIONS UPON CONSTRAINT VIOLATIONS

→ The actions that are performed on the database are

- i) Insertion
  - ii) Deletion
  - iii) Update
- on performing these actions we should be aware that the constraints are not violated (Domain, key Entity, Referential constraints).

→ If the above actions violate the constraints the default action is to reject such actions which are conflicting in violation.

→ While "deleting", the constraint that get violated is "Referential Integrity" and the actions that must be taken are:

1. Ignore it (reject the action)
2. cascade (Delete the tuple and also delete the tuples which are being referenced by the above tuple & they should be deleted).
3. Set NULL or some other value

use null value

→ Given a Relation R(A<sub>1</sub> A<sub>2</sub> A<sub>3</sub>... A<sub>n</sub>)  
candidate key = {A<sub>1</sub>}

Then the no. of possible super keys are = 2<sup>n-1</sup>

usually will

primary key

department

Employee

① ② ③ ④

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

↓ ↓ ↓ ↓ ↓  
(n-1) elements

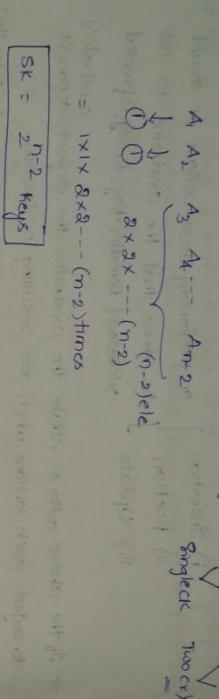
∴ Total no. of keys = 1 × 2 × 2 × 2 × ... (n-1) times

∴ Total no. of SK's = 2<sup>n-1</sup>

### 9. COUNTING THE NO. OF SKS POSSIBLE - EXAMPLE 2

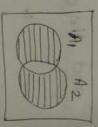
Relation  $R = (A_1, A_2, A_3, \dots, A_n)$

Candidate keys =  $\{A_1, A_2\}$  Candidate key is  $A_1, A_2$  not  $A_1, A_2$



$$SK = \frac{n-2}{2} \text{ keys}$$

No. of candidate keys =  $\{A_1, A_2\}$   
The no. of super keys =  $SK(A_1) + SK(A_2) + SK(A_1, A_2)$



$$SK = 2^0 + 2^{n-2}$$

### 10. COUNTING THE NO. OF SKS POSSIBLE - EXAMPLE 3

Relation  $R = (A_1, A_2, A_3, \dots, A_n)$

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

No. of SK's =  $SK(A_1) + SK(A_2, A_3) - SK(A_1, A_2, A_3)$

$$SK = \frac{n-1}{2} + \frac{2^{n-2}}{2} - \frac{2^{n-3}}{2}$$

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

No. of SK's =  $SK(A_1, A_2) + SK(A_3, A_4) - SK(A_1, A_2, A_3, A_4)$

$$SK = \frac{n-2}{2} + \frac{2^{n-2}}{2} - \frac{2^{n-4}}{2}$$

CK =  $\{A_1, A_2, A_1, A_3\} \Rightarrow$  No. of SK's =  $SK(A_1, A_2) + SK(A_1, A_3) - SK(A_1, A_2, A_3)$

$$SK = \frac{n-2}{2} + \frac{2^{n-2}}{2} - \frac{2^{n-3}}{2}$$

### 11. COUNTING

Relation  $R$

CK

$$\Rightarrow R = (A_1, A_2)$$

$$CK = (A_1, A_2)$$

### 11. COUNTING THE NO. OF SK'S POSSIBLE - Example - 4

Relation  $R = (A_1, A_2, A_3, \dots, A_n)$

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

$$\begin{aligned} \text{No. of } SK's &= SK(A_1) + SK(A_2) + SK(A_3) - SK(A_1, A_2) - SK(A_1, A_3) \\ &\quad + SK(A_2, A_3) \end{aligned}$$

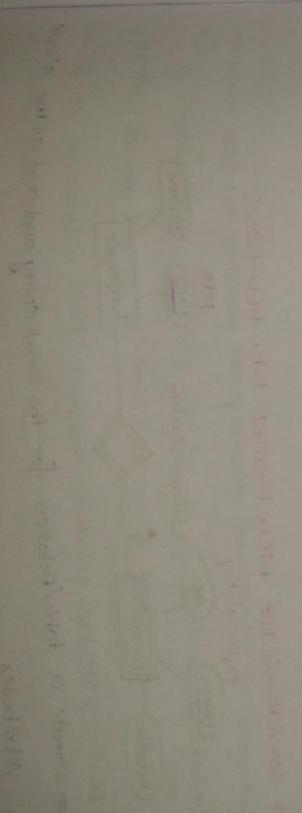
$$SK = 2^{n-1} + 2^{n-1} + 2^{n-1} - 2^{n-2} - 2^{n-2} + 2^{n-3}$$

$$\Rightarrow R = (A, B, C, D)$$

$$CK = (A_1, BC) \quad \left\{ \begin{array}{l} \text{No. of } SK's = SK(A) + SK(BC) - SK(ABC) \\ \qquad \qquad \qquad = 2^3 + 2^2 - 2^1 \end{array} \right.$$

$$SK = 8 + 4 - 2 = 10$$

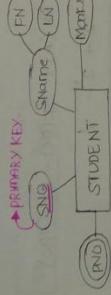
Below is a diagram showing all possible subsets of  $A = \{a, b, c, d\}$ . It shows all non-empty subsets of  $A$ , including the empty set and the full set  $A$ .



### 3. CONVERSION OF ER MODEL TO RELATIONAL MODEL

#### 1. STEP 1

- ⇒ There are 4 steps to convert ER model (which is designed at conceptual level) to Relational model and RDBMS can be applied appropriately.
- ⇒ The delete operation will be handled if you do not want a particular record.
- ⇒ FOR EVERY ENTITY IN ER-model, we HAVE TO come up with A RELATION IN RELATIONAL model.



Now the Relation for this  
ER-diagram will be

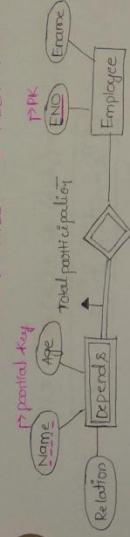
Student [SNO] [LN] [FN] [ENR]

#### 3. STEP-3

- ⇒ Every simple attribute is represented in the table.
- ⇒ Composite attribute is further divided and atomic parts are represented in the table.
- ⇒ Multi-valued entities are not represented in the Relation.
- ⇒ Represent the primary key in the ER-model in the Relation also (underlined).

#### 2. STEP-2

##### ⇒ CONVERTING THE WEAK ENTITIES INTO RELATIONS.

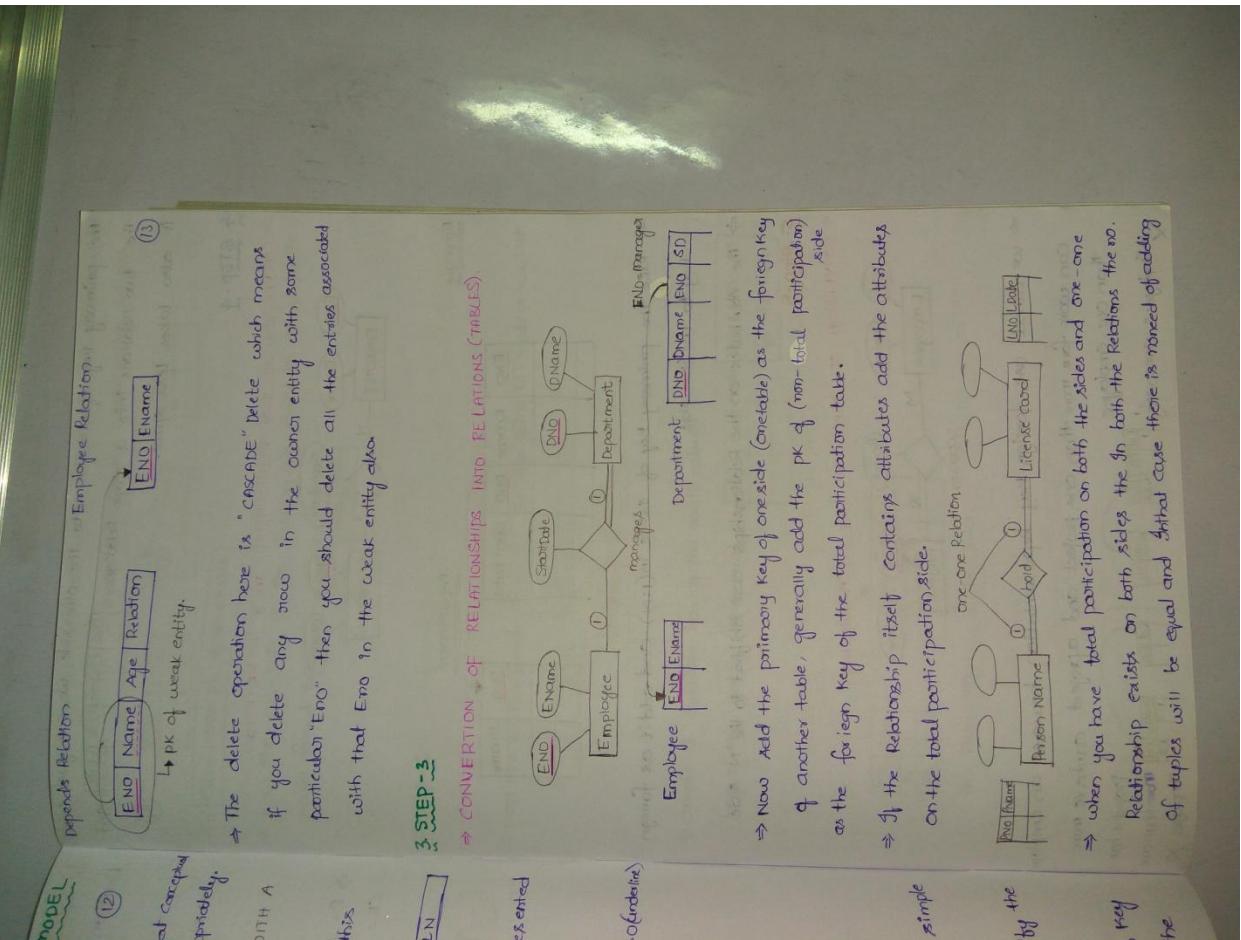


#### 4. STEP-4

- ⇒ Now Add primary key of the dependent (ENO) as the foreign key of the weak entity, and make primary key of weak entity and the pk of strong entity as the pk of weak entity.
- ⇒ If the weak entity has attributes, then add them as attributes of the strong entity.

- ⇒ Create a table/Relation for the weak entity and add all the simple attributes.
- ⇒ Identify the primary key in the weak entity and also identify the primary key in the owner entity (strong entity).

- ⇒ Now Add primary key of the owner entity (ENO) as the foreign key of the weak entity, and make primary key of weak entity and the pk of strong entity as the pk of weak entity.
- ⇒ When you add foreign key, then add it with the name of the table.

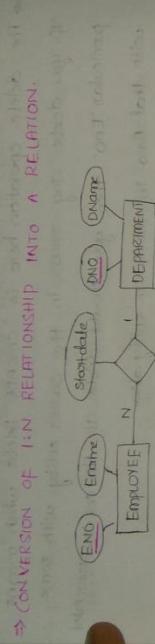


the primary key of one side onto the other side we can just combine the relation to the two relations into a single relation.

Proj	Phone	Proj	Phone
1	1	2	2

It also takes less space.

#### 4. STEP-4



Employee

ENO	Ename	DNO	Ssn

Department

⇒ Now the attributes  
are ready for use.

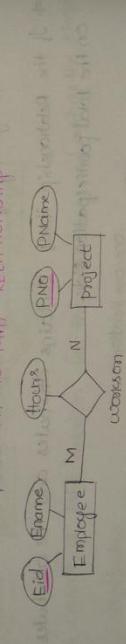
⇒ Take the primary key of 1 side i.e. N and add it as foreign key to the N's side.

⇒ Create a new table.

⇒ The attributes on the relationships are shifted to the N side.

#### 5. Step-5

⇒ CONVERSION OF MANY TO MANY RELATIONSHIP INTO A RELATION.



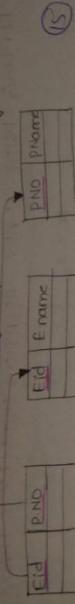
∴ PK = ProjID

⇒ we cannot use foreign key representation because an employee can work for more than one project and a project can have more than one employee.

Project 1 has  
Employee 1 works on  
Project 2 has  
Employee 2 works on  
X

ProjID	Proj	ProjID	Proj
1	1	2	2
1	1	2	2

combine  $\Rightarrow$  The solution to the above problem is create a new table having and attributes as the primary keys of the participating entities.



Eid + PNO = PK of Newtable

$\Rightarrow$  Now the attribute on the Relationship should be added to the newly formed table.



Step - 6

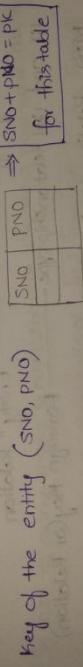
$\Rightarrow$  DEALING WITH MULTIVALUED ATTRIBUTES

$\Rightarrow$  For the multivalued attribute we are going to create a new table



Foreign

$\Rightarrow$  Create a Relation having the multi valued attribute and primary side.



for this table

SNO + PNO = PK

SNO	PNO	LN
1	P1	a
1	P2	b

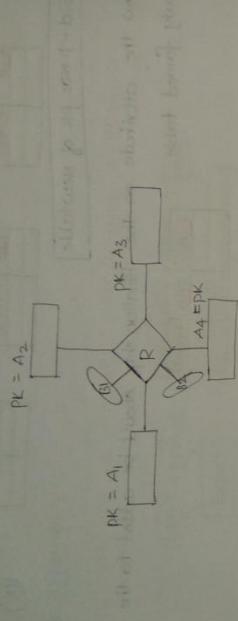
$$\therefore PK = SNO + PNO$$

PK = SNO  
PK = PNO  
PK = SNO + PNO

in employee  
one more

## 7. STEP-7

$\Rightarrow$  DEALING WITH N-ARY RELATION SHIPS (MORE THAN 4 ENTITIES)



$\Rightarrow$  Now create a newtable having the attributes as primary keys of all the entities.

	$A_1$	$A_2$	$A_3$	$A_4$	$B_1$	$B_2$
Attribute values of room						

$\Rightarrow$  Let us say

## 8. SUMMARY OF ER TO RDG CONVERSION

ER-Model	Relational Model
Entity type	"Entity" Relation
1:1, and 1:N Relationship type	Foreign Key (or Relation)
M:N Relationship type	Relationship "relation" + 2 PKs
N-ary Relationship type	Relationship "relation" + n PKs
Simple Attribute	Attribute
Composite Attribute	Set of simple component Attributes
Multi-valued Attribute	Relation and Foreign Key
Value set	Domain
Key Attribute	Prominary Key

9. GATE 12

Given the box  
Inconvertible

$\Rightarrow$  An attribute  
 $\Rightarrow$  An attribute  
 $\checkmark$  A row

#### 4. ENTITIES

#### 9. GATE II 2005 QUESTION ON ER-DIAGRAMS



Lodging is many-many Relationship. Rent, payment to be made, by person(s) occupying different hotel rooms should be added as an attribute to  
 a) Hotel b) Lodging c) person d) None.

Sol: Let us say Hotel Room connects HNo, HName, and if we add

HNo	HName	Rent
1	a	
2	b	

Many people can reside and we cannot put all the Rents in single tuple  $\Rightarrow$  option A X

$\Rightarrow$  Let us say the person table has pid, pname, Rent

Pid	Pname	Rent
1	a	
2	b	

$\Rightarrow$  He may stay at many hotels and each hotel has its own Rent, so we cannot parallelise Rents here  $\Rightarrow$  option C X

$\Rightarrow$  In case of many-many Relationship we create a new table having

The pk of participating entities.

" + 2FK's  $\Rightarrow$  Table of

" + n FK's.

$\therefore$  The Attribute Rent should be on Lodging.

#### Attributes

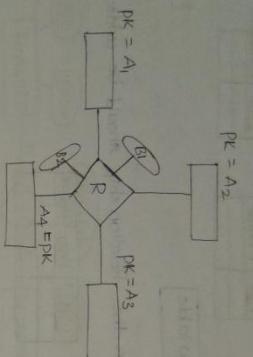
#### 10. GATE II 2005 QUESTION ON CONVERTING ER TO RDB

Given the basic ER and Relational models, which of the following is incorrect

- a) An attribute of an entity can have more than one value
  - b) An attribute of an entity can be composite.
- $\checkmark$  A row of Relational table an attribute can have more than one value as a row of Relational table, an attribute can have exactly one value or null

## 1. STEP - 1

⇒ DEFINING WITH N-ARY RELATIONSHIPS (MORE THAN 1 ENTITIES)



→ Now create a new table having the attributes as primary keys of all the entities.

A1	A2	A3	A4	B1	B2

## 2. SUMMARY OF ER TO RDB CONVERSION

ER-Model

Relational model

Entity type

"Entity" Relation

1:N and :N Relationship type → Foreign Key (or Relation)

MIN Relationship type → Relationship "relation" + 2FK's

N:M Relationship type → Relationship "relation" + nFK's

Simple Attribute → Attribute

Composite Attribute → set of simple component Attributes

Multivalued Attribute → Relation and Foreign Key

Value set → Domain

Key Attribute → primary key.

## 3. GATE IT :

Hold
Room

Lodging is a

by person (A)

attribute to

of Hotel

Sol: Let us

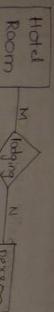
Rent

⇒ Let us say

⇒ Let us say

GATE IT 2005 QUESTION ON ER-DIAGRAMS

ER Diagram



Lodging is many-many Relationship. Rent, payment to be made.

by person(s) occupying different hotel rooms should be added as an attribute to

- Hotel
- Lodging
- Person
- Name

Q1: Let us say Hotel Room contains HNO, HName, and if we add

Primary Keys

Rent

HNO	HName	Rent
1	a	
2	b	

→ Many people can reside and  
we cannot put all the Rents  
in single tuple → option A X

⇒ Let us say the person table has pid, name, Rent.

Pid	Name	Rent
1	a	
2	b	

→ It may stay at many hotels  
and each hotel has its own  
Rents. So we cannot parallel the  
Rents here, ⇒ option C X

⇒ Increase of many-many Relationship we create a new table having

Pid	HNO	Rent
1	1	1000
2	2	2000

↳ Table of  
the Relationship  
Lodging

⇒ The attribute Rent should be on Lodging.

Tables

### Q12. GATE 12 QUESTION ON CONVERTING ER TO RDB

Given the basic ER and Relational models, which of the following is correct

- An attribute of an entity can have more than one value.
- An attribute of an entity can be composite.
- In a row of relational table an attribute can have exactly a value or null
- In a row of relational table, an attribute can have exactly a value or null

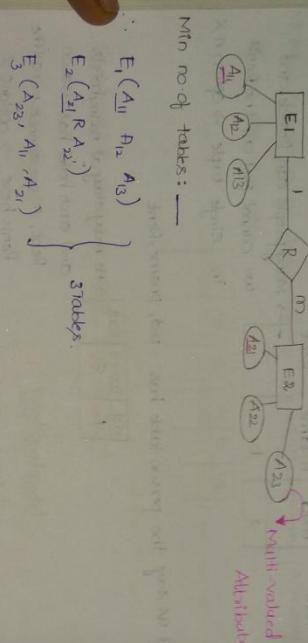
- a) option a talks about ER-model and multi-valued attributes in ER-model = CORRECT

b) In ER-model Attributes can be composite = CORRECT

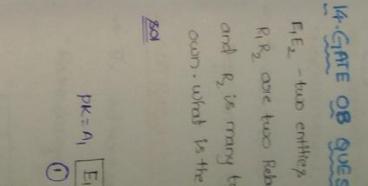
c) option c talks about Relational model and Relational model doesn't allow multi-valued attributes and composite attributes = INCORRECT

d) option d is correct since there should be exactly one value in each field and can have null values = CORRECT.

### 11. STATE IT OR CONVERSION OF ER TO RDB



$$\left. \begin{array}{l} E_1(A_1, A_2, A_3) \\ E_2(A_{21}, A_{22}) \\ E_3(A_{23}, A_1, A_2) \end{array} \right\} \text{3 tables}$$



### 14. STATE OR QUES

R<sub>1</sub>, R<sub>2</sub> are two relations and R<sub>2</sub> is many to one. what is the reason?

$$PK = A_1$$

①

### 12. STATE OR ON CASCADE DELETE IN CASE OF FOREIGN KEYS

The following table has two attributes A and C where A is the primary key referencing with an delete cascade.

The set of all tuples that must be additionally deleted to preserve referential integrity when the tuple (2,4) is deleted is:

- (a) (3,4) and (6,4) (b) (5,2)(7,2) (c) (5,2)(7,2)(9,5) (d) (3,4)(4,3)(6,4)

### 15. STATE IT OR

Let R(a,b,c) and S(b,c,d) that refers to R and S.

- (a) Insert into R

which of the following is true?

- (a) None of a, b, c  
 (b) All of a, b, c  
 (c) Both a, b  
 (d) Both b, c

Table:

id	A	C
1	1	1
2	2	4
3	3	4
4	4	3
5	5	2
6	7	2
7	9	5
8	6	4

R
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

R
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

(8)

(9)

(10)

(11)

(12)

(13)

(14)

(15)

(16)

(17)

(18)

(19)

(20)

(21)

(22)

(23)

(24)

(25)

(26)

(27)

(28)

(29)

(30)

(31)

(32)

(33)

(34)

(35)

(36)

(37)

(38)

(39)

(40)

(41)

(42)

(43)

(44)

(45)

(46)

(47)

(48)

(49)

(50)

(51)

(52)

(53)

(54)

(55)

(56)

(57)

(58)

(59)

(60)

(61)

(62)

(63)

(64)

(65)

(66)

(67)

(68)

(69)

(70)

(71)

(72)

(73)

(74)

(75)

(76)

(77)

(78)

(79)

(80)

(81)

(82)

(83)

(84)

(85)

(86)

(87)

(88)

(89)

(90)

(91)

(92)

(93)

(94)

(95)

(96)

(97)

(98)

(99)

(100)

(101)

(102)

(103)

(104)

(105)

(106)

(107)

(108)

(109)

(110)

(111)

(112)

(113)

(114)

(115)

(116)

(117)

(118)

(119)

(120)

(121)

(122)

(123)

(124)

(125)

(126)

(127)

(128)

(129)

(130)

(131)

(132)

(133)

(134)

(135)

(136)

(137)

(138)

(139)

(140)

(141)

(142)

(143)

(144)

(145)

(146)

(147)

(148)

(149)

(150)

(151)

(152)

(153)

(154)

(155)

(156)

(157)

(158)

(159)

(160)

(161)

(162)

(163)

(164)

(165)

(166)

(167)

(168)

(169)

(170)

(171)

(172)

(173)

(174)

(175)

(176)

(177)

(178)

(179)

(180)

(181)

(182)

(183)

(184)

(185)

(186)

(187)

(188)

(189)

(190)

(191)

(192)

(193)

(194)

(195)

(196)

(197)

(198)

(199)

(200)

(201)

(202)

(203)

(204)

(205)

(206)

(207)

(208)

(209)

(210)

(211)

(212)

(213)

(214)

(215)

(216)

(217)

(218)

(219)

(220)

(221)

(222)

(223)

(224)

(225)

(226)

(227)

(228)

(229)

(230)

(231)

(232)

(233)

(234)

(235)

(236)

(237)

(238)

(239)

(240)

(241)

(242)

(243)

(244)

(245)

(246)

(247)

(248)

(249)

(250)

(251)

(252)

(253)

(254)

(255)

(256)

V.V. Imp						
R	a	b	c	d	e	f
all rows	1	1	1	1	1	1
all columns	1	2	3	2	3	2
row 1	1	2	3	2	3	2
row 2	2	3	4	3	4	3
row 3	3	4	5	4	5	4
row 4	4	5	6	5	6	5
row 5	5	6	7	6	7	6
row 6	6	7	8	7	8	7
row 7	7	8	9	8	9	8
row 8	8	9	10	9	10	9
row 9	9	10	11	10	11	10
row 10	10	11	12	11	12	11

S

V.V. Imp  $\Rightarrow$  d is depending on a not a is depending on b.

1. INT  $\Rightarrow$  f is depending on c.

2. INT  $\Rightarrow$  f is depending on d.

Violation (Inserting into S).  $\Rightarrow$  Les

$\Rightarrow$  Inserting into "S" and Deletion from R' are going to cause violations. (may cause we are not sure about it).

$\Rightarrow$  splitting not

2. INT  $\Rightarrow$  The "Recursion" words.

## 4. NORMALISATION

### INTRODUCTION TO NORMALISATION

- ⇒ If we hold the entire data in a single table it will take more space.
- ⇒ To cause less Redundancy
- ⇒ Various Anomalies will occur
  - Insert Anomalies
  - Update Anomalies
  - Deletion Anomalies

(2)

- ⇒ splitting the tables into small tables such that our design will not contain all the above anomalies and Redundancy is called "NORMALISATION".
- ⇒ In order to do Normalisation we use the concept of Functional Dependencies (FDs) and the concept of candidate keys.

### INTRODUCTION TO FUNCTIONAL DEPENDENCIES

- ⇒ The Advantage of the Functional dependency is it Reduces Redundancy.

Records	A	B	C
$t_1$	1	a	b
$t_1$	2	3	c
$t_2$	2	a	b

Here the functional dependency is  $A \rightarrow BC$   
 ⇒ for a value of A you can get / derive the values of B and C uniquely

$$A \rightarrow BC$$

$$2 \rightarrow ab.$$

$$\boxed{\begin{array}{l} \text{if } (a=2) \text{ then } \\ \quad (ab) \end{array}}$$

- (a) if  $t_1(A) = t_2(A)$  then  
 $t_1(BC) = t_2(BC)$

Normal form.

- ⇒ Initially we see that the table is in 1st Normal form.
- ⇒ Next 2nd NF
- ⇒ Next 3NF
- ⇒ Next BCNF

ie possible.

we find  $G_1 \supseteq F$  then

Now find  $\mathfrak{I} \equiv F$  then,

$$F: \{A \rightarrow C, AC \rightarrow D, E \rightarrow AD, E \rightarrow$$

✓ Check whether these are derivable

61

of FDs 27

lowing P.O's is

2

Both the F and G are Equivalent.

## EQUIVALENCE OF EDs Example-1

$F = \{A \Rightarrow B, B \Rightarrow C, C \Rightarrow D\}$  check whether these two FDs are Equivalent or not?

26  $\Rightarrow$  Take Each FDoG and check whether it is derivative from;

$\Rightarrow$  Now take A from F and check B is present in A.

$$\Rightarrow C \rightarrow D \Rightarrow C^+ = \{D, C\} \Rightarrow \text{holds} \quad \therefore F \sqsupseteq G$$

res Equivalents

$\vdash \text{take}$   
 $\text{ingn: } A^+ = \{A, B, C\} - \{A \rightarrow B \text{ holds}\}$

180

it is already

Both the functional dependencies are not equivalent.

TRUE

### 30. EQUIVALENCE OF TWO FDs EXAMPLE - 2

①  $F = \{A \rightarrow B, AB \rightarrow C, D \rightarrow AC, D \rightarrow E\}$  - ②  $F = \{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$   
 $G_1 = \{A \rightarrow BC, D \rightarrow AB\}$        $G_1 = \{A \rightarrow BC, B \rightarrow A, C \rightarrow A\}$

$\underline{\underline{G_1 \sqsupseteq F}}$        $\Rightarrow A^+ \text{ in } F = \{A, B, C\} \quad F \sqsupseteq G_1$

$\Rightarrow D^+ \text{ in } F = \{D, E, A, C\}$

$\Rightarrow A^+ \text{ in } G_1 = \{A, B, C\} \quad A \rightarrow B \quad F \sqsupseteq G_1$

$\Rightarrow (AB)^+ = \{A, B, C\} \quad AB \rightarrow C \quad F \sqsupseteq G_1$

$\Rightarrow D^+ = \{A, B, C\} \quad D \rightarrow AC \quad F \sqsupseteq G_1$

$\Rightarrow D^+ = \{A, B, C, D\} \quad D \rightarrow EX \quad F \sqsupseteq G_1$

∴ These two FDs are not equivalent.

②  $A^+ \text{ in } G_1 = \{A, B, C\} \quad A \rightarrow BC \quad F \sqsupseteq G_1$

$B^+ \text{ in } F = \{B, C, A\} \quad B \rightarrow A \text{ holds} \quad F \sqsupseteq G_1$

$\Rightarrow C^+ \text{ in } F = \{A, B, C\} \quad C \rightarrow A \text{ holds} \quad F \sqsupseteq G_1$

$\Rightarrow G_1 \sqsupseteq F \Rightarrow \text{Now, } A^+ \text{ in } G_1 = \{A, B, C\} \quad A \rightarrow B \quad F \sqsupseteq G_1$

$\Rightarrow \text{Now, } B^+ \text{ in } G_1 = \{B, A, C\} \quad B \rightarrow C \quad F \sqsupseteq G_1$

$\Rightarrow \text{Now, } C^+ \text{ in } G_1 = \{C, A, B\} \quad C \rightarrow A \text{ holds} \quad F \sqsupseteq G_1$

∴ Both the functional dependencies are equivalent.

### 31. MINIMAL COVER

If we have a set of functional dependencies 'F' and we could minimise it to other set of functional dependencies 'G' such that 'G' covers 'F' and 'F' covers 'G' and 'G' is minimal, then 'G' is called Minimal cover of 'F'.

### 32. MINIMA

Minimize  $\{A$

①  $A \rightarrow B$

Now,  $G_1$

PRACTICE TO  
1. split the F  
Ex:  $A \rightarrow BC$ ,

2. find the Red

Ex:  $AB \rightarrow C$

3. Find the Red

Ex:  $AB \rightarrow C$

4) Minimize  $\{A \Rightarrow$

①  $A \Rightarrow C$ ,  $A$

③ PROCEDURE TO FIND MINIMAL SET

$\Rightarrow A^+$

$C \Rightarrow A^+$

$B \Rightarrow A^+$

$A \Rightarrow C$

1. split the FDs such that LHS contain single attribute.
- Ex:  $A \Rightarrow BC$ ,  $A \Rightarrow B \wedge A \Rightarrow C$
2. Find the Redundant FDs and delete them from the set.

Ex:  $\{A \Rightarrow B, B \Rightarrow C, A \Rightarrow C\} \Rightarrow \{A \Rightarrow B, B \Rightarrow C\}$

3. Find the Redundant attributes on RHS and delete them.

- i) minimize  $\{A \Rightarrow C, AC \Rightarrow D, E \Rightarrow AD, E \Rightarrow H\}$
- ii)  $A \Rightarrow C, AC \Rightarrow D, E \Rightarrow A, E \Rightarrow D$

- iii)  $E \Rightarrow H$

This production is useless (remove/delete)

so, this production and try to find  $E^+$  in the Remaining FDs.

If it contains 'B' then ' $E \Rightarrow D$ ' is Redundant FD.

$$E^+ = \{A, E, H, C, D\} \Rightarrow E \Rightarrow D \text{ is Redundant.}$$

$\Rightarrow \{A\}$

$\boxed{\{C, D, F\}}$

- ②  $A \Rightarrow C$ ,  $AC \Rightarrow D$ ,  $E \Rightarrow AD$ ,  $E \Rightarrow H$

$\downarrow$  This will be Redundant if  $C^+$  contains A

$C^+ = \{C\}$   $\downarrow$  This is Redundant

$\downarrow$  This will be Redundant if A<sup>+</sup> contains C then,

$$A^+ = \{A, C\} \Rightarrow AC \Rightarrow D \text{ becomes } A \Rightarrow D$$

- ③  $A \Rightarrow C$ ,  $A \Rightarrow D$ ,  $E \Rightarrow A$ ,  $E \Rightarrow D$

$\downarrow$  This will be Redundant if A<sup>+</sup> contains D then,

$\Rightarrow A \Rightarrow CD$ ,  $E \Rightarrow AH$

### 32. MINIMAL COVER Example - 1

Minimize  $\{A \Rightarrow B, C \Rightarrow B, D \Rightarrow ABC, AC \Rightarrow D\}$

- ①  $A \Rightarrow B$ ,  $C \Rightarrow B$ ,  $D \Rightarrow A$

$\cancel{D \Rightarrow B}$

$D \Rightarrow C$

$AC \Rightarrow D$

Redundant  $\Rightarrow D^+ = \{D, A, B\}$   $D \Rightarrow B$  is derivable

in G'

Now,  $(AC)^+ = \{ACB\}$   $AC \Rightarrow D$  is not Redundant

$$\textcircled{2} \quad A \rightarrow B, C \rightarrow B, D \rightarrow A, D \rightarrow C, AC \rightarrow D$$

Now  $\Delta C \rightarrow D$  can be deleted if  $A^+$  contains  $(62)$  or  $C$  contains  $A$ .

Now,  $A^t = \{A, B\}$   
 $C^t = \{C, D\}$   $\therefore A \leftarrow D$  cannot be deleted.

卷之三

$\overrightarrow{ABC}$

### 33. GATE-2013 ON MINIMAL COVER

$\Rightarrow \{AB \rightarrow C, D \rightarrow E, E \rightarrow C\}$  is the minimal cover of  $\{AB \rightarrow C, D \rightarrow E, E \rightarrow C\}$

Step-1:  $AB \rightarrow C$        $D \rightarrow E$

$\overbrace{AB \rightarrow E}$        $E \rightarrow C$

$\therefore \{AB \rightarrow C, D \rightarrow E, E \rightarrow C\}$  is not the minimal cover, The minimal cover should be  $\{D \rightarrow E, AB \rightarrow E, E \rightarrow C\}$

$AB^+$  =  $\{A, B, C\}$ . This cannot be deleted and must be in the minimal set

⇒ Mainly Normalisation is about splitting the tables i.e. decomposing the tables, so while decomposing we should see some properties. One such property is "Lossless decomposition".

⇒ when we decompose a Relation we should check that there is a common attribute in both of them, if not check what happens in below example.

Now if I choose a common Alphabet Randomly.

$ABC$

$\overline{AB}$

$\overline{AC}$

$\overline{BC}$

$a_1$	$b_1$	$c_1$
$a_2$	$b_1$	$c_1$
$a_1$	$b_2$	$c_1$
$a_1$	$b_1$	$c_2$
$a_2$	$b_2$	$c_1$
$a_1$	$b_2$	$c_2$

$E, A \rightarrow E$

Spurious Tuples.

the

deletions

the

deletions

the

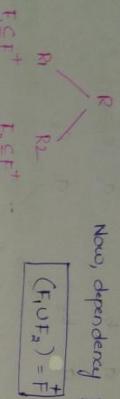
<div data

### 35. FD PRESERVING

$R(A_1 A_2 A_3 \dots A_n)$

$FD - F^+$  {set of all functional dependencies that are applicable on R}

Now, dependency preserving means



⇒ The dependency preserving is not a mandatory thing.

### 36. DECOMPOSITION - EXAMPLE 1

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

$R_1(AB)$

$R_2(BC)$

⇒ Now we need to check the lossless decomposition and dependency preservation.

⇒  $R_1(AB) R_2(BC) \Rightarrow$  This decomposition is lossless because the common variable in  $R_1 R_2 = B$  and 'B' is a key attribute in  $R_2(BC) \{ \vdash B \rightarrow C \}$ .

Now,  $R(AB)$

$\begin{array}{l} \checkmark A \rightarrow B \\ \checkmark B \rightarrow A \\ \checkmark C \rightarrow B \end{array}$  are the non-trivial dependencies.

Non-Trivial Dependencies

$\begin{array}{l} B^+ = \{BC\} \\ C^+ = \{CA\} \quad (C \rightarrow B) \end{array}$

$F_1 \cup F_2 = \begin{array}{l} A \rightarrow B \\ B \rightarrow A \\ C \rightarrow B \end{array}$

$= \begin{array}{l} A \rightarrow B \\ B \rightarrow A \\ A \rightarrow C \\ C \rightarrow B \end{array}$

Redundant

∴ The decomposition is lossless and dependency preserving.

### 37. DECOMPOSING

$R(ABCD)$

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

$D = \{AB, BC, CD\}$

Given  $D =$

LOSSLESS  
DECOMPOSITION

Now coming

$R_1(AB) \quad | \quad R(BC)$

$A \rightarrow B$

$B \not\rightarrow A$

$C \rightarrow B$

$\checkmark B \rightarrow A$

$C \not\rightarrow A$

$D \not\rightarrow A$

$C \not\rightarrow A$

$B \not\rightarrow C$

$C \not\rightarrow B$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

$C \not\rightarrow D$

$D \not\rightarrow C$

$B \not\rightarrow D$

$D \not\rightarrow B$

## DECOMPOSITION EXAMPLE 2

de con R<sub>j</sub>  
R(ABCD)  
 $f = (A \Rightarrow B, B \Rightarrow C, C \Rightarrow D, D \Rightarrow A)$   
D: (AB, BC, CD)  
R<sub>i</sub><sup>T</sup> = I<sub>D</sub>

$$\begin{aligned} & \text{Given } D = \left( \frac{AB}{BC}, CD \right) \\ & \text{Given } D = \left( \frac{B^2 + \{ABC\}}{BC}, CD \right) \\ & \text{Given } D = \left( \frac{B^2 + \{ABC\}}{BC}, CD \right) \\ & \text{Given } D = \left( \frac{B^2 + \{ABC\}}{BC}, CD \right) \end{aligned}$$

decomposition

(BCD)

(CD)

(AB)

(BCD)

(CD)

(AB)

(ABCD)

decomposition

LOSE less

comm'mg to Sanc:

$R_1(AB)$	$R_1(BC)$
$A \rightarrow B$	$B \rightarrow C$
$\downarrow B \rightarrow A$	$C \rightarrow B$
$B = \{ABC\}$	$D = \{ABC\}$
$\downarrow$	$\downarrow$
$C^t = \{ABC\}$	$D^t = \{ABC\}$

Now,  $F_1 \cup F_2 \cup F_3 = A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A$  [indirectly covered]  
 $\boxed{E_1 \cup E_2 \cup E_3 = E}$   
 Ex:  $E_1$  is homogeneous (PULL)

### 3. DECOMPOSITION

encigas.

$$D = \{ \underline{AD}, \underline{BCD} \}$$

Q1 The common attribute is 'D' now 'D' should be a key in any one of

### the Relationships $R_{\text{CAD}}$ $R_2(\text{BCD})$

卷之三

∴ The decomposition is lossless decomposition

sen ving.

Now,

$R_1(AD)$

$A \rightarrow D \times$

$D \rightarrow A \checkmark$

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

$R_2(BCD)$

$B^+ = (B) \times$

$C^+ = (C) \times$

$D^+ = (D, A) \times$

$\therefore BD \rightarrow C$

$(BC)^+ = (BC) \times$

$(BD)^+ = (BDA) \checkmark$

$(CD)^+ = (CDA) \times$

40. DECOMPO

$R(ABCDE)$

$F: (A \rightarrow BC, C \rightarrow D, D \rightarrow E)$

$R_1(ABCD), R_2(E)$

Now, Given

$\therefore ABCD$

$A \rightarrow BCD \checkmark$

$B \rightarrow BC \times$

$C \rightarrow D \checkmark$

$D \rightarrow DX$

$(BC) \rightarrow D.$

Now  $A^+ = AB$

$B^+ = B$

$C^+ = CD$

$D^+ = D,$

$(BC)^+ =$

$(BD)^+ =$

$(CD)^+ =$

$(AB)^+ =$

$(AC)^+ =$

$(AD)^+ =$

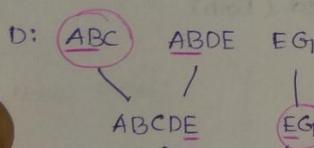
$\therefore$  The decomposition is lossless and non-FD preserving.

### 39. DECOMPOSITION EXAMPLE 4

$R(ABCDEF)$

$F: \{AB \rightarrow C, AC \rightarrow B, AD \rightarrow E, B \rightarrow D, BC \rightarrow A, E \rightarrow G\}$

$D: (ABC, ABDE, EG)$



$\therefore AB \rightarrow C$  is possible  $\Rightarrow$

The decomposition is lossless

Now

$R_1(ABC)$

$A \rightarrow A \times$

$B \rightarrow D$  (D notion)

$C \rightarrow C \times$  (ABC)

$(AB)^+ = ABCDEF \Rightarrow AB \rightarrow C$  (BE)  $\rightarrow D$

$BC \rightarrow A$

$AC \rightarrow B$

$R_2(ABDE)$

$B \rightarrow D$ .

$AB \rightarrow DE$

$AD \rightarrow E$

$ABD \rightarrow E$

$ABE \rightarrow D$

$ABE \times \Rightarrow$

$R_3(EG)$

$E \rightarrow G$

$A^+ = A$

$B^+ = BD$

$C^+ = C$

$(AB)^+ = ABCDEF$

$(BC)^+ = BCDAEG$

$(AC)^+ = ACBDEF$

(42) The decomposition is lossless and dependency preserving.

#### 4. DECOMPOSITION EXAMPLE 5

$R(ABCDE)$

$f: (A \rightarrow BC, C \rightarrow DE, D \rightarrow E)$

$R_1: (ABCD) R_2: (DE)$ .

(43)

NOW Given

$(ABCD)$

$(DE)$

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

$= (AB) \Rightarrow \therefore$

$AB \rightarrow CD$  is not  
preserved)

$\therefore ABCD$

$A \rightarrow BCD \checkmark$

$B \rightarrow BX$

$C \rightarrow D \checkmark$

$D \rightarrow DX$

$(BC) \rightarrow D$ .

Now  $A^+ = ABCDE$

$B^+ = B$

$C^+ = CDE$

$D^+ = D, E$

$(BC)^+ = BCDE$

$(BD)^+ = BDE$

$(CD)^+ = CDE$

$(AB)^+$

$(AC)^+$

$(AD)^+$

$R_2 (DE)$

$D \rightarrow E \checkmark$

$E \rightarrow D X$

$E^+ = \{E\}$

$$\begin{aligned} \therefore FD's &= A \rightarrow BCD \\ &\quad C \rightarrow D \\ &\quad BC \rightarrow D \\ &\quad D \rightarrow E \end{aligned} \left. \right\} = F_1 \cup F_2 = F$$

and they will become SK's.

$\therefore$  The decomposition is lossless and dependency preserving.

A

BD

C

ABCDEG

BCDAEG

ABDEG

#### 4.1. DECOMPOSITION Example - 6

R (ABCDEF)

F: {AB → C, AC → B, AD → E, BC → A, E → G}

D: (AB, BC, ABDE, EG)

$$D: \begin{array}{c} AB \\ \equiv \\ BC \\ / \\ ABC \end{array} \quad \begin{array}{c} ADE \\ / \\ E(G) \end{array}$$

Now  $B^+ = \{B, D\}$  B is not the key of AC

∴ The decomposition is lossy decomposition.

#### 4.2. FIRST NORMAL FORM

⇒ The process of Removing the Redundancy is Normalisation

A	B	C	D	E	F
1	2	3	4	5	6
7	8	9	10	11	12

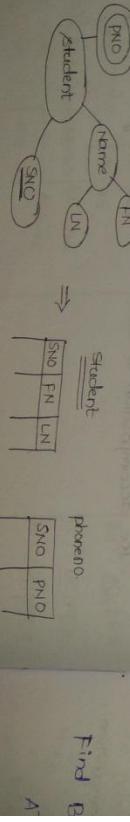
D	E	F	A
1	2	3	4
5	6	7	8

⇒ Our ultimate aim is to Reduce a table to BCNF, it is known that by the time you convert the table into BCNF there won't be any Redundancy (Redundancy = 0%)

⇒ INF → 2NF → 3NF → BCNF

⇒ INF says that the table has to be flat

⇒ By default every relational database is in First Normal form



⇒ NO

43. SECOND

⇒ Let us do  
the func

⇒ Now,

$(AB)^+ = C$ .

Now,

A
1
2
3

A
1
2
3

A
1
2
3

A
1
2
3

A
1
2
3

(4)

→ No multivalued and composite values are allowed in 1NF.

### SECOND NORMAL FORM INTRODUCTION

Let us assume we have a table having attributes ABC and the functional dependencies that are allowed are  $AB \rightarrow C$ ,  $B \rightarrow C$ .

$$\text{Now } AB \rightarrow C \quad \left\{ \begin{array}{l} A^+ = A \\ B^+ = BC \\ C^+ = C \end{array} \right\} \text{ with one attribute key is not possible}$$

... Try with 2 attributes.

$(AB)^+ = (ABC) \therefore (AB)$  has the capacity to become Candidate Key.

Now,  $AB \rightarrow C \nrightarrow$  says that AB (combination) can determine C.  $B \rightarrow C \nrightarrow$  says that BC (itself) can determine C uniquely.

A	B	C
1	a	c <sub>1</sub>
2	a	c <sub>1</sub>
1	b	c <sub>2</sub>
2	b	c <sub>2</sub>
1	c	c <sub>3</sub>
2	c	c <sub>3</sub>
3	c	c <sub>3</sub>
4	c	c <sub>3</sub>
5	c	c <sub>3</sub>

⇒ The 2NF says that if your candidate key is containing more than one attribute then a part of the key should not determine anything else.

⇒ 2NF is based on Full Functional Dependency		
→ partial dependency is called (A part of the key determines something)		

This table is not in 2NF because there is a partial dependency  $(B \rightarrow C)$  in the candidate key  $(AB \rightarrow C)$ . So we should eliminate partial dependency.

1 form

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

$\cancel{\text{No}} \quad \cancel{\text{PNO}}$

$$A^+ = \{A\} \quad \xrightarrow{X_A \in A} \quad \begin{array}{|c|c|} \hline AB & BC \\ \hline \end{array} \rightarrow B^+ = BC$$

lossless decomposition.  
dependency.  
 $\times AB \rightarrow ABC$  Not FD preserving.

#### 44. 2NF Example 1

$R(ABCD)$

FD:  $\{AB \rightarrow C, B \rightarrow D\}$  what is the highest Normal Form satisfied?

⇒ By default Every Relation will be in 1NF

⇒ Now find all the candidate keys

$$A^+ = A$$

$$B^+ = BD \quad \therefore (AB) \text{ is the candidate key}$$

$$AB^+ = ABCD$$

Now the three attribute candidate keys are possible and they should not contain  $(AB)$  if they include then it will become superkey.

$$(ACD)^+ = (ACD)$$

$$(BCD)^+ = (BCD)$$

$$\therefore (AB) \rightarrow ABCD$$

$(B \rightarrow D) \rightarrow$  partial Dependency

Now, find  $A^+ = A$ .

$$B^+ = BD$$

$R(ABCD)$

$R(ACB)$

$R(BD)$

Remaining we inserted so that we should have some common part

3	8	A
19	0	I
10	10	2
33	d	3
33	d	2
22	3	1
22	3	1
22	3	1

$R(ABCD)$

$R(ACB)$

$R(BD)$

A	C	B
B	D	

$$AB \rightarrow C$$

$$B \rightarrow D$$

FD preserving too.

Lossless decomposition.

#### 45. 2NF E

$R(ABCDE)$

$$AB \rightarrow C$$

$$BD \rightarrow EF$$

$$AD \rightarrow GH$$

$$A \rightarrow I$$

$$H \rightarrow J$$

Now, find

of left ho

all the o

$$(AB)^+ =$$

$$R(ABCDEF)$$

A	B	C	D	E	F

$$A^+ = (A, I)$$

A	I

$$A \rightarrow I$$

#### 46. 2NF E

$R(ABCDE)$

$$F: \{A \rightarrow B, B \rightarrow C\}$$

The pair

Now,

$$A \rightarrow RC$$

### 45. 2NF EXAMPLE 2

fixed?

$R(ABCDEFGHIJ)$

$AB \rightarrow C$

$BD \rightarrow EF$

$AD \rightarrow GH$

$A \rightarrow I$

$H \rightarrow J$

Candidate Key = ABD.

$(ABD) \rightarrow (ABCDEFGHIJ)$

$BD \rightarrow EF$

$AD \rightarrow GH$

$A \rightarrow I$

$AB \rightarrow C$

} partial Dependencies

and  
some

Now, find all the closures of all partial dependencies (find closures of left hand side) and try to create a new table by taking away all the attributes which are determined by such dependencies

$$(AB)^+ = \underline{ABC}$$

$R(ABCDEFGHIJ)$

AB	$\rightarrow$	A	B	C
----	---------------	---	---	---

$$(BD)^+ = \underline{BDEF}$$

$R(ABCDEFGHIJ)$

BD	$\rightarrow$	B	D	E	F
----	---------------	---	---	---	---

$$(AD)^+ = \underline{ADGHJ}$$

$R(ABCDEFGHIJ)$

AD	$\rightarrow$	A	D	G	H	J
----	---------------	---	---	---	---	---

$$A^+ = (A, I)$$

A	I
---	---

$$A \rightarrow I$$

$$ABD$$

A	B	D
---	---	---

$R(BD)$

B	D
---	---

$$B \rightarrow D$$

vring too.  
decomposition

### 46. 2NF EXAMPLE 3

$R(ABCDE)$

Candidate Key = (AC),

F:  $\{A \rightarrow B, B \rightarrow E, C \rightarrow D\}$

The partial dependencies are  $A \rightarrow B$   
 $C \rightarrow D$

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

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

$A \rightarrow B$        $R(ABCDEF)$

A	B	E
---	---	---

$R(ACDEF)$

C	D
---	---

AC

A	C
---	---

Now, if the candidate key for a table is a single attribute and that is the only candidate key then the relation is in 2nd NF.

### 5.1. THIRD NORMAL FORM INTRODUCTION

3NF: No Transitive Dependencies

Transitive Dependency: Non prime attribute transitively depending on the key.

R(ABC)

FD: {A → B, B → C}

CK = {A} ⇒ No partial dependencies.

R(ABC)

FD: {AB → C, C → D}

CK = {B}

Transitive dependency : The attribute which is a part of key.

Non-prime attribute : The attribute which is not a part of the key.

A → B  
B → C

Transitive dependency

Now, B<sup>t</sup> = {B, C} }  
R(A B C)

Lossless Decomposition

$\begin{array}{|c|c|} \hline A & B \\ \hline \end{array}$        $\begin{array}{|c|c|} \hline B & C \\ \hline \end{array}$

FD preserving

A → B  
B → C

### 48. 3NF Example 1

R(ABCDE)

FD = {AB → C, B → D, D → E}

CK = AB

No. of partial dependencies are B → D

B<sup>t</sup> = {B, D}      R(A B C D E F)

AB → C

2NF

$\begin{array}{|c|c|c|} \hline B & D & E \\ \hline \end{array}$        $\begin{array}{|c|c|c|} \hline A & B & C \\ \hline \end{array}$

B → D  
D → E

Ex

RC

CK

A -

A →

A →

B -

Table Q and Rule  $D \rightarrow E$  is the Transitive dependency. The rule  $(B \rightarrow D) \rightarrow E$  is not in 3NF.

Q

$D \rightarrow E$  is the Transitive Dependency present in table RQE

$\Rightarrow$  Now DT = {D, E}

R (B D E)

	D	E	A	B	C
B					
	D	E	AB	BC	

3NF

spending on

### 4. 3NF EXAMPLE 2

R(ABC)

FD: {AB  $\rightarrow$  C, C  $\rightarrow$  A}

↓ Key.  
↓ the key.

some candidate

some candidate

$$\begin{aligned} \text{Now, } B^+ &= \{B\} \\ (AB)^+ &= \{ABC\} \quad \vdash \text{Candidate Keys} \\ (BC)^+ &= \{BCA\} \quad \vdash \\ &= (AB) \oplus C \end{aligned}$$

Allude prime attributes.

VVVV Imp.

partial dependency = part of key  $\rightarrow$  Non-prime attribute  
 $\Rightarrow$  prime attribute  $\rightarrow$  Non-prime attribute

Here  $C \rightarrow A$  is not partial dependency because  $C \rightarrow$  prime attribute but  $(A)$

$\therefore C \rightarrow A$  is not partial dependency.

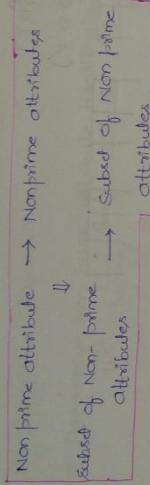
Ex: R(ABCDEF)

$$\begin{aligned} CK: ABC \\ A \rightarrow D = \text{partial dependency.} \\ A \rightarrow B \\ A \rightarrow C \\ B \rightarrow C \end{aligned}$$

3NF

In the above table there are no partial dependencies. The Dependencies are in 2NF

Transitive dependency



$$R(ABCDEF) \Rightarrow CK = ABC$$

$$\begin{cases} D \rightarrow E \\ E \rightarrow F \end{cases}$$

Transitive Dependencies.

$$B \rightarrow C$$

$$D \rightarrow C$$

Not Transitive dependency

The above relation does not contain Transitive dependency

$\Leftrightarrow$  The Relation is in 3NF

### 50. Formal Definition of 3NF

A Relational schema R is in 3NF if every Non-trivial FD  $X \rightarrow Y$

- a)  $X$  is a superkey
- or
- b)  $X$  is a prime attribute

Which of the following is allowed in 3NF?

- proper subset of CK  $\rightarrow$  Non-prime (partial dependency)
- Non-prime  $\rightarrow$  Non-prime (transitive dependency)
- proper subset of CK + Non-prime  $\rightarrow$  Non-prime (transitive dependency)
- proper subset of CK  $\rightarrow$  proper subset of other CK (not a pp, TD) ✓

### 51. 3NF E

R(ABCDEF)  
Now, consider

Now, A<sup>+</sup> =

$\{A, B, C\}$

Transitive Dependencies

$C \rightarrow D$

$A \rightarrow C$

$B \rightarrow C$

$F$

$\Rightarrow$  Candidate

Now, C

$\{C, D\}$

$E$

$\{E\}$

$\{F\}$

$\{G\}$

Now,

e.g.: The Relation  
is in 2NF

### 3NF Example 3

R(ABCDEF) F: {A → FC, C → D, B → E}

⑤

Now, candidate key =  $(AB)^+$  =  $(ABCE\bar{F})$

Now,  $(AB)^+ = (ABFCDE)$  ∴ (AB) is the candidate key

Now,  $A \rightarrow FC$ ,  $B \rightarrow E$  } are the practical dependencies.

⇒

Now,  $A^+ = \{A\}_{FD}$   $B^+ = \{B, E\}$

dependency

$\begin{array}{|c|c|c|c|} \hline A & F & C & D \\ \hline \end{array}$   $\begin{array}{|c|c|} \hline B & E \\ \hline \end{array}$   $\begin{array}{|c|} \hline F \\ \hline \end{array}$  ∵ The relation is in 2NF

Transitive dependency

Relation is in 3NF

Non-trivial FDs are:  
 $A \rightarrow FC$ ,  $C \rightarrow D$ ,  $B \rightarrow E$

### 52. BCNF INTRODUCTION

Relational schema R is in BCNF if whenever a non-trivial functional dependency  $x \rightarrow A$  holds in R, then 'x' is a superkey of R. i.e. determinants of all functional dependencies should be superkeys.

R(ABC)  
F: {A → B, B → C, C → A}  
⇒ candidate keys = {A, BC} ✓  
so pp, PD ✓

$\begin{array}{|c|c|c|c|} \hline A & B & C & A \\ \hline \end{array}$

BCNF







(5)

- ⇒ If a table contains only 2 attributes then the Relation will definitely be in BCNF.

### BCNF Example - 6

$R(ABDPLT) \quad FD\{B \rightarrow PT, T \rightarrow L, A \rightarrow D\}$

Candidate key =  $(AB)^+ = (ABDPLT) \Rightarrow (AB)^+ = (ABDPLT) //$

$\boxed{CK = AB}$

Now, partial Dependencies =  $B \rightarrow PT \quad \left\{ \begin{array}{l} B^+ = (B, PT) \\ A \rightarrow D \end{array} \right. \quad (GAP) \quad (GAP)$

$A^+ = (A, D)$

$R(ABDPLT) \quad \left\{ \begin{array}{l} R_1(A, D) \\ R_2(B, PL) \\ R_3(A, B) \end{array} \right. \quad \overline{BCNF}$

$A \rightarrow D \quad \left\{ \begin{array}{l} B \rightarrow PT \\ T \rightarrow L \end{array} \right. \quad 2NF$

$\left\{ \begin{array}{l} SK \\ SK \end{array} \right. \quad \text{The tuples are}$

$R(ABDPLT)$

$R_1(A, D)$

$R_2(B, PL)$

$R_3(A, B)$

$T_L \quad \left\{ \begin{array}{l} B \rightarrow PT \\ T \rightarrow L \end{array} \right. \quad BCNF$

$PT \quad \left\{ \begin{array}{l} B \rightarrow PT \\ SK \end{array} \right. \quad BCNF$

$L \quad \left\{ \begin{array}{l} T \rightarrow L \\ SK \end{array} \right. \quad BCNF$

$[D, TL, PT, AB]$

### Super Key

### BCNF Example - 7

$R(ABCDE) \quad FD\{A \rightarrow BC, CD \rightarrow E, B \rightarrow D, E \rightarrow A\}$  candidate key = {AE, CD}

All the attributes are prime

⇒ 3NF

Now, To check whether the Relation is in BCNF check the functional dependencies and find whether the LHS is a super key or not

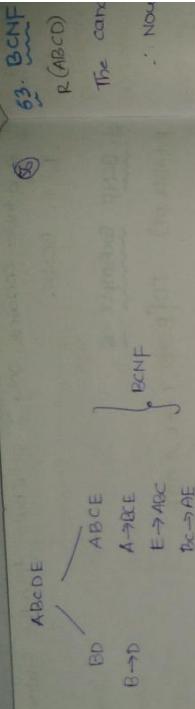
$A \rightarrow BC \Rightarrow A^+ = (ABCDE) \Rightarrow A$  is SK

$CD \rightarrow E \Rightarrow (CD)^+ = (CDEAB) \Rightarrow CD$  is SK

$B \rightarrow D \Rightarrow (B)^+ = (BD) \Rightarrow$  Not SK

$E \rightarrow A \Rightarrow (E)^+ = (E, A, BC, D) = E$  is SK

$AC \rightarrow D \quad \left\{ \begin{array}{l} BD \\ ABCE \\ A \rightarrow BC \\ E \rightarrow A \end{array} \right. \quad BCNF$



### 61. BCNF Example 8 - PART 1 AND 62 - PART 2

- $R(ABCD)$
- $FD\{AB \rightarrow CD, D \rightarrow A\}$
- $\Rightarrow$  candidate key =  $\{(AB)\} \{DC\}$  :  $(AB)$  is DB candidate key.
- $\Rightarrow$  partial dependencies are absent. Therefore the relation is in 2NF
- $\Rightarrow$  Transitive dependencies are not there and  $D$  is not superkey but the R is prime attr.
- $\Rightarrow$  For BCNF the LHS should always be a super key, and here  $D$  is not superkey in  $D \rightarrow A$

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

Lossless Decomposition	
$ABcD$	$/$
$D$	$AB$
$A$	$BC$
$D \Rightarrow A$	$B^+ = \{B\}$
$BCNF(2)$	$(BC)^+ = \{BC\}$
Attributes	$C^+ = \{C\}$
$D^+ = \{D, A\}$	$(BDAC) \Rightarrow BD \rightarrow C \checkmark$
	$(CD)^+ = \{CD\}$

Now, find if  $AB \rightarrow CD$  is preserved using  $D \rightarrow A$  and  $BD \rightarrow C$

$\Rightarrow (AB)^+ = \{AB\}$  ...  $AB \rightarrow CD$  is not preserved.

$\therefore$  Dependency preserving failed.

(B)

### BCNF Example 9

R(ABCD) {A  $\rightarrow$  B, B  $\rightarrow$  C, C  $\rightarrow$  D}

The candidate Key = ( )<sup>+</sup> = (ABC)

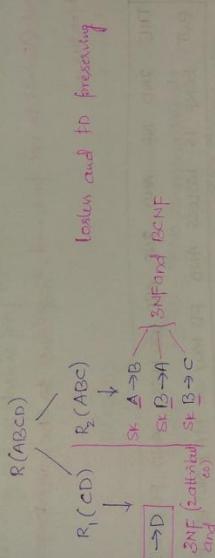
$$\begin{aligned} \text{Now, } A^+ &= \{A, BC\} \cup \\ B^+ &= \{B, AC, D\} \cup \end{aligned}$$

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

Now, A, B are prime attributes  $\Rightarrow$  Now, partial dependencies are not present  
(Only Single attribute CK's).

Now for 3NF check the this. This must be a super key or  
the RHS should be a prime attribute in  $C \rightarrow D \Rightarrow$  violating  
3NF  
Not SK Not prime attribute

Now, C<sup>+</sup> = {C, D}



BD  $\rightarrow$  C ✓  
and BD  $\rightarrow$  C

### 64. BCNF - Example 10 PART 1 AND 65 PART 2

R(ABCDE) FD: {AB  $\rightarrow$  C, C  $\rightarrow$  D, D  $\rightarrow$  E, E  $\rightarrow$  A}

Candidate Key = ( B )<sup>+</sup> = (ABCDE)

$$\left. \begin{aligned} B^+ &= \{B\} \quad \text{Now } (AB)^+ = \{A, B, C, D, E\} \\ (CB)^+ &= \{B, C, D, E, A\} \quad \left\{ \begin{array}{l} \text{All the attributes} \\ \text{are prime attributes} \end{array} \right. \\ (PB)^+ &= \{B, D, E, A, C\} \\ (EB)^+ &= \{B, E, A, C, D\} \end{aligned} \right\} \Rightarrow R' \text{ is in 3NF}$$

GATE-98  
which N  
database

a) 2NF

GATE-99  
Let R =

$C \rightarrow F, E$

$RABC$

For BCNF, every this of the functional dependency should be a superkey

( $\oplus$ )

$AB \rightarrow C \Rightarrow AB$  (SK)

$C \rightarrow D$

$D \rightarrow E$   
C, D, E are not SK's

$ABCD$

$(C, DEA)$

$E \rightarrow A$

$C \rightarrow F$

$RABC$

GATE-01  
consider

$C \rightarrow F$

$RABC$

$B \rightarrow C$

$A \rightarrow B$

$C \rightarrow D$

$D \rightarrow E$

$E \rightarrow A$

$F \rightarrow P$

$P \rightarrow B$

d) not

lossless

partial

total

BCNF

$A^+ = (A)$

$B^+ = (B)$

$C^+ = (C)$

$D^+ = (D)$

$E^+ = (E)$

$F^+ = (F)$

$G^+ = (G)$

$H^+ = (H)$

$I^+ = (I)$

$J^+ = (J)$

$K^+ = (K)$

$L^+ = (L)$

$M^+ = (M)$

$N^+ = (N)$

$O^+ = (O)$

$P^+ = (P)$

$Q^+ = (Q)$

$R^+ = (R)$

$S^+ = (S)$

$T^+ = (T)$

$U^+ = (U)$

$V^+ = (V)$

$W^+ = (W)$

$X^+ = (X)$

$Y^+ = (Y)$

$Z^+ = (Z)$

Now,  $D^+ = DEA$

Now,  $C^+ = CDE$

Now,  $B^+ = ABC$

Now,  $A^+ = A$

Now,  $F^+ = F$

Now,  $G^+ = G$

Now,  $H^+ = H$

Now,  $I^+ = I$

Now,  $J^+ = J$

Now,  $K^+ = K$

Now,  $L^+ = L$

Now,  $M^+ = M$

Now,  $N^+ = N$

Now,  $O^+ = O$

Now,  $P^+ = P$

Now,  $Q^+ = Q$

Now,  $R^+ = R$

Now,  $S^+ = S$

Now,  $T^+ = T$

Now,  $U^+ = U$

Now,  $V^+ = V$

Now,  $W^+ = W$

Now,  $X^+ = X$

Now,  $Y^+ = Y$

Now,  $Z^+ = Z$

Now,  $D^+ = DAE$

Now,  $C^+ = CDE$

Now,  $B^+ = ABC$

Now,  $A^+ = A$

Now,  $F^+ = F$

Now,  $G^+ = G$

Now,  $H^+ = H$

Now,  $I^+ = I$

Now,  $J^+ = J$

Now,  $K^+ = K$

Now,  $L^+ = L$

Now,  $M^+ = M$

Now,  $N^+ = N$

Now,  $O^+ = O$

Now,  $P^+ = P$

Now,  $Q^+ = Q$

Now,  $R^+ = R$

Now,  $S^+ = S$

Now,  $T^+ = T$

Now,  $U^+ = U$

Now,  $V^+ = V$

Now,  $W^+ = W$

Now,  $X^+ = X$

Now,  $Y^+ = Y$

Now,  $Z^+ = Z$

Now,  $D^+ = DAE$

Now,  $C^+ = CDE$

Now,  $B^+ = ABC$

Now,  $A^+ = A$

Now,  $F^+ = F$

Now,  $G^+ = G$

Now,  $H^+ = H$

Now,  $I^+ = I$

Now,  $J^+ = J$

Now,  $K^+ = K$

Now,  $L^+ = L$

Now,  $M^+ = M$

Now,  $N^+ = N$

Now,  $O^+ = O$

Now,  $P^+ = P$

Now,  $Q^+ = Q$

Now,  $R^+ = R$

Now,  $S^+ = S$

Now,  $T^+ = T$

Now,  $U^+ = U$

Now,  $V^+ = V$

Now,  $W^+ = W$

Now,  $X^+ = X$

Now,  $Y^+ = Y$

Now,  $Z^+ = Z$

Now,  $D^+ = DAE$

Now,  $C^+ = CDE$

Now,  $B^+ = ABC$

Now,  $A^+ = A$

Now,  $F^+ = F$

Now,  $G^+ = G$

Now,  $H^+ = H$

Now,  $I^+ = I$

Now,  $J^+ = J$

Now,  $K^+ = K$

Now,  $L^+ = L$

Now,  $M^+ = M$

Now,  $N^+ = N$

Now,  $O^+ = O$

Now,  $P^+ = P$

Now,  $Q^+ = Q$

Now,  $R^+ = R$

Now,  $S^+ = S$

Now,  $T^+ = T$

Now,  $U^+ = U$

Now,  $V^+ = V$

Now,  $W^+ = W$

Now,  $X^+ = X$

Now,  $Y^+ = Y$

Now,  $Z^+ = Z$

Now,  $D^+ = DAE$

Now,  $C^+ = CDE$

Now,  $B^+ = ABC$

Now,  $A^+ = A$

Now,  $F^+ = F$

Now,  $G^+ = G$

Now,  $H^+ = H$

Now,  $I^+ = I$

Now,  $J^+ = J$

Now,  $K^+ = K$

Now,  $L^+ = L$

Now,  $M^+ = M$

Now,  $N^+ = N$

Now,  $O^+ = O$

Now,  $P^+ = P$

Now,  $Q^+ = Q$

Now,  $R^+ = R$

Now,  $S^+ = S$

Now,  $T^+ = T$

Now,  $U^+ = U$

Now,  $V^+ = V$

Now,  $W^+ = W$

Now,  $X^+ = X$

Now,  $Y^+ = Y$

Now,  $Z^+ = Z$

Now,  $D^+ = DAE$

Now,  $C^+ = CDE$

Now,  $B^+ = ABC$

Now,  $A^+ = A$

Now,  $F^+ = F$

Now,  $G^+ = G$

Now,  $H^+ = H$

Now,  $I^+ = I$

Now,  $J^+ = J$

Now,  $K^+ = K$

Now,  $L^+ = L$

Now,  $M^+ = M$

Now,  $N^+ = N$

Now,  $O^+ = O$

Now,  $P^+ = P$

Now,  $Q^+ = Q$

Now,  $R^+ = R$

Now,  $S^+ = S$

Now,  $T^+ = T$

Now,  $U^+ = U$

Now,  $V^+ = V$

Now,  $W^+ = W$

Now,  $X^+ = X$

Now,  $Y^+ = Y$

Now,  $Z^+ = Z$

Now,  $D^+ = DAE$

Now,  $C^+ = CDE$

Now,  $B^+ = ABC$

Now,  $A^+ = A$

Now,  $F^+ = F$

Now,  $G^+ = G$

Now,  $H^+ = H$

Now,  $I^+ = I$

Now,  $J^+ = J$

Now,  $K^+ = K$

Now,  $L^+ = L$

Now,  $M^+ = M$

Now,  $N^+ = N$

Now,  $O^+ = O$

Now,  $P^+ = P$

Now,  $Q^+ = Q$

Now,  $R^+ = R$

Now,  $S^+ = S$

Now,  $T^+ = T$

Now,  $U^+ = U$

Now,  $V^+ = V$

Now,  $W^+ = W$

Now,  $X^+ = X$

Now,  $Y^+ = Y$

Now,  $Z^+ = Z$

Now,  $D^+ = DAE$

Now,  $C^+ = CDE$

Now,  $B^+ = ABC$

Now,  $A^+ = A$

Now,  $F^+ = F$

Now,  $G^+ = G$

Now,  $H^+ = H$

Now,  $I^+ = I$

Now,  $J^+ = J$

Now,  $K^+ = K$

Now,  $L^+ = L$

Now,  $M^+ = M$

Now,  $N^+ = N$

Now,  $O^+ = O$

Now,  $P^+ = P$

Now,  $Q^+ = Q$

Now,  $R^+ = R$

Now,  $S^+ = S$

Now,  $T^+ = T$

<p><u>Q1</u> What would be a suitable database design?</p> <p>a) 2NF      b) 3NF      c) 4NF      d) 5NF      e) 3NF (Actual ans is BCNF)</p>
<p><u>ANSWER</u></p> <p>Let <math>R = (ABCDEF)</math> be a relation scheme with the following dependency <math>C \rightarrow F</math>, <math>E \rightarrow A</math>, <math>EC \rightarrow D</math>, <math>A \rightarrow B</math>. what is the key of <math>R</math>?</p>
$R(ABCDEF) \Rightarrow (Ec)^+ = \overline{(ABCDEF)} \Rightarrow (Ec)^+ = \{Ec, A, B, EF\}$
<p><u>ANSWER</u></p> <p>Consider the schema <math>R(ABCD)</math> and functional dependencies <math>A \rightarrow B</math> and <math>C \rightarrow D</math>. Then the decomposition of <math>R'</math> into <math>R_1(AB)</math> and <math>R_2(CD)</math> is</p> <ul style="list-style-type: none"> <li>a) FD preserving and lossless join</li> <li>b) lossless but FD preserving fails</li> <li>c) FD preserving but not lossless join</li> <li>d) Not FD preserving and not lossless join</li> </ul>
<p><u>ANSWER</u></p> <p>Given <math>R(ABCD)</math></p> <p>Now, <math>R_1(AB) \quad R_2(CD)</math></p> <p><math>R_1(AB) \quad R_2(CD) \Rightarrow</math> No common attribute</p> <p><math>\therefore</math> lossy decomposition.</p>

### GATE 02

Relation  $R$  with an associated set of FDs  $F$  is decomposed into BCNF. The Redundancy (causing out of functional dependency) in the resulting set of relations is:

- Zero BNF = 0% Redundancy)
- More than Zero but less than 3NF decomposition
- proportional to the size of  $F^+$
- Indecomposable

### GATE QUESTION ON NORMALISATION 2

#### GATE-05

which one of the following statements about Normal forms is false?

- BCNF is stronger than 3NF (left hand side should be a SK in BCNF)  
by go son
- losses, FD preserving into 3NF is always possible (TRUE)
- loses, " " " BCNF " " " (WE CAN NOT GET RID OF)
- Any Relation with 2 attributes is in BCNF. (TRUE) LHS = Key (SK) consider RHS =

#### GATE-II-OS

A table has fields  $F_1 F_2 F_3 F_4 F_5$  with the following functional dependencies  $F_1 \rightarrow F_3, F_2 \not\rightarrow F_4, F_1 F_2 \rightarrow F_5$  what is the NF of the relation?

- 1NF
- 2NF
- 3NF
- None

Now, Candidate key  $(F_1 F_2)^+ = (F_1 F_2 F_3 F_4 F_5)$

$(F_1 F_2)^+ = (F_1 F_2 F_3 F_4 F_5) \therefore F_1 F_2$  is candidate key.

Now,  $\not\rightarrow$  partial dependences are present  $\therefore$  2NF  $\times \begin{cases} F_1 \rightarrow F_3 \\ F_2 \rightarrow F_5 \end{cases}$

Now, LHS should be super key for 3NF  $\Rightarrow F_1 F_2 \left\{ \begin{array}{l} \text{one SK} \\ F_1 F_2 \end{array} \right\} = 3NF$

No.

Ques-12

which of the following is True?

- a) Every relation in 3NF is also in BCNF (Not always)
- b) A Relation 'R' is in 3NF if every non-prime attribute of 'R' is fully functionally dependent on every key of R
- c) Every relation in BCNF is in 3NF Some Key
- d) No Relation can be in both BCNF and 3NF (FALSE)

(61)

### 6.8 GATE QUESTION ON NORMALISATION 3

GATE-14

A prime attribute of a relation scheme R is an attribute that appears

- a) In all candidate keys of R  
~~if~~ some candidate key of R
- b) In a foreign key of R
- c) Only in the primary key of R

GATE-95

Consider  $R(ABC)$  FD:  $\{AB \rightarrow C, C \rightarrow A\}$  Show that R is in 3NF but not BCNF

$\therefore$  Now, candidate key =  $(B)^{\pm} = (ABC)$   
functional  
FD of the

$$\begin{aligned} (AB)^+ &= (ABC) \\ (CB)^+ &= (ABC) \end{aligned} \quad \left. \begin{array}{l} \text{All are prime attributes} \\ \text{if the relation is in 3NF} \end{array} \right.$$

$$\begin{aligned} AB \rightarrow C &\in \text{SK} \\ C \rightarrow A &\in \text{NOT SK} \end{aligned} \quad \left. \begin{array}{l} \text{not in BCNF} \end{array} \right.$$

GATE-97

$R(a,b,c,d)$ , abcd contains atomic values (No composite and multivalue)

- a) 3NF but not 2NF  
b) 2NF but not 3NF  
c) IN 3 NF  
d) None of the above  
 $\Rightarrow a \rightarrow c$   $b \rightarrow d$   
 $\Rightarrow a \rightarrow c$   $b \rightarrow d$  are partial dependency.

GATE-14

one SK  
 $= 3NF$

### 69. STATE QUESTIONS ON NORMALISATION 4

Q1 ANS - q9

R(S,T,U,V,W) ; FD{S→T, T→U, U→V, V→S} and let (R<sub>1</sub> and R<sub>2</sub>) = R

be a decomposition such that R<sub>1</sub>∩R<sub>2</sub> ≠ ∅ . The decomposition is:

- a) Not in 2NF
- b) In 2NF but not in 3NF
- c) In 3NF but not in 2NF
- d) Both 2NF and 3NF

$$\begin{aligned} C^k &= (S)^t = (STUV) \\ (T)^t &= (TUWS) \\ U^t &= (UVT) \end{aligned} \quad \left\{ \begin{array}{l} \text{All are prime} \\ \Rightarrow NF = 3NF \end{array} \right.$$

### 70. STATE 2001 QUESTION ON BCNF

R(A,B,C,D) is a relation. which of the following does not have a lossless-join dependency preserving BCNF decomposition?

- a) A→B, B→CD
- b) A⇒B, B→C, C⇒D
- c) AB→C, C→AD
- d) A→BCD

option A: A→B, B→CD      R(A,B,C,D)

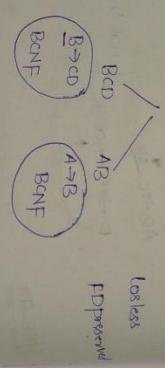
C<sup>k</sup> = (A)<sup>t</sup> = (ABCD)      No partial dependencies = 2NF

Transitive dependency = B→CD

B<sup>t</sup> = (BCD)

A<sup>t</sup> = (AB)

C<sup>t</sup> = (CD)



option B: A→B, B→C, C→D

C<sup>k</sup> = A      (A)<sup>t</sup> = (ABCD) → Transitive dependencies

A<sup>t</sup> = AB      (AB)<sup>t</sup> = (ABCD) → No partial dependencies

B<sup>t</sup> = BC      (BC)<sup>t</sup> = (BCD)

C<sup>t</sup> = CD      (CD)<sup>t</sup> = (BCD)

lossless and FD preserving

The highest n

④ 2NF

### 71. STATE 2001 QUESTION

option C: AB→C

C→AD is

C<sup>t</sup> = {C, A, D}

R is decomposed into R<sub>1</sub> and R<sub>2</sub>. The Relation R<sub>1</sub> is in BCNF.

To make query

should be 0

- a) Dependency-
- b) Lossless join
- c) BCNF definition
- d) 3NF definition

### 72. STATE Q4

The Relation R has the FDs

present

lossless

dependency

dependency

(Q2)

option c:  $AB \rightarrow C$      $C \rightarrow AD$      $CK = (A,B)(C,B)$  one candidate key

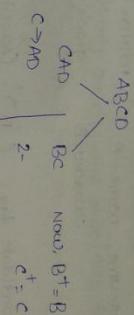
$$R_2) = R$$

ion is:

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

All are prime

$\Rightarrow NF = 3NF$



The dependency  $[AB \rightarrow C]$  is lost

### 11. GATE 2002 QUESTION

Relation 'R' is decomposed using a set of FD's F and Relation 'S' have a  
is decomposed using another set of FD's G. One decomposition  
is in BCNF and other is in 3NF, but is not known which which.  
To make guaranteed identification, which one of the following tests  
should be used on the decompositions?

- Dependency preservation  $\Rightarrow$  BCNF may not have dependency preserving
- Lossless join  $\Rightarrow$  Both have lossless but we cannot differentiate them.
- BCNF definition  $\Rightarrow$  Enough to prove BCNF table and other automatically
- 3NF definition  $\Rightarrow$  Not enough BCNF cannot be identified.

### 12. GATE 94 QUESTION ON NORMALISATION

The Relation scheme student performance (name, courseNo, rollNo, grade)  
has the FD's

lost  
FD preserved

Name, courseNo  $\rightarrow$  grade

rollNo, courseNo  $\rightarrow$  grade

Name  $\rightarrow$  rollNo

rollNo  $\rightarrow$  name

The highest normal form of this Relation scheme is:  
a) 2NF    b) 3NF    c) BCNF    d) 4NF

lossless join

candidate key =  $(CNO)^+ = (\text{name}, \text{rollno}, \text{courseNo}, \text{Grade})$

(6)

$(CNO)^+ = (CNO)$

Let us assume, name = A

courseNo = B

rollNo = C

Grade = D

FD's

$AB \rightarrow D$

$CB \rightarrow D$

$A \rightarrow C$

$C \rightarrow A$

Given

Also, f given terms

Now, candidate key =  $(B)^+ = (ABC)$

$B^+ = B$

$(AB)^+ = (ABC)$

$(CB)^+ = (ABC)$

$(BC)^+ = (ABC)$

$(AB)^+ = (ABC)$

are CK's.  $\Rightarrow (AB), (CB) \Rightarrow$  prime attributes are A, B, C

Now, partial dependencies are absent  $\therefore 2NF \checkmark$

Now, 3NF check

AB = SK

$CB = SK$

$A \rightarrow C \Rightarrow LHS$  is not SK but RHS is prime attribute

$C \rightarrow A \Rightarrow LHS$  is not SK but RHS is prime attribute

$AB = SK$

Now, BCNF check

$\Rightarrow LHS$  should be superkey  $\Rightarrow$  courseNo to find

$\Rightarrow AB = SK$

$CB = SK$

$\left. \begin{array}{l} A \rightarrow C \\ C \rightarrow A \end{array} \right\} BCNF \times$

$\therefore$  Now, BCNF

Ans = 3NF

13. 5A

A Rel

name,

also, f

given

terms

a) 1NF

Let on

n

Given

Now,

13. GATE IT- 95 QUESTION ON NORMALISATION

३

A relation Empdt1 is defined with attributes emp code (unique), name, street, city, state and pincode; if there is only one city and state. Also, for any pincode, there is only one city and state. Also, for any given street, city and state, there is just one pincode. In Normalization

From Empdt<sub>2</sub> is a relation in.

```

Let empcode = A    Street = C    state = E
      name = B    city = D    pincode = F
                                         R(ABCDEF)

```

Given  $\text{empcode}$ := the candidate key  $\Rightarrow A \rightarrow BCDEF$

Now, 'A' is the candidate key  $\Rightarrow$  No partial dependencies are CDE  $\rightarrow$  F

⇒ 2NF ✓

3NF ✓  $\Rightarrow$  Now, BNF check, [LHS = SK or RHS = prime attribute]

$F \rightarrow DE$   
 Not s/c      Not prime      3NF X

2NP

4. GRATE OF QUESTION

Which one of the following statements is false?

- b) Any relation with 2 attributes is in BCNF  
 c) Any relation with every Key having only one attribute is in 2NF  
 d) A prime attribute can be transitively dependent on a key in 3NF

15. GATE 2008

QUESTION ON NORMALISATION

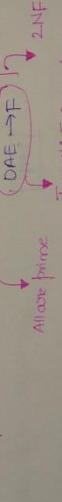
(6)

- Consider the relation scheme of Library DB.
- Book (Title, Author, catalog\_no, publication\_year, price)  $\xrightarrow{F}$  (Title, Author, catalog\_no, publication\_year, price)
- collection (Title, Author, catalog\_no) with the dependencies
- Author  $\rightarrow$  catalog\_no
  - catalog\_no  $\rightarrow$  title, Author, publisher, year
  - publisher, year title  $\rightarrow$  price
- Assume (Author, Title) is the key for both schemas • which of the following statements are true?
- Both Book and collection are in BCNF
  - Both Book and collection are in 3NF only
  - Book is in 2NF and collection is in 3NF
  - Both Book and collection are in 2NF only

Sol: Book (ABCEF)

collection (ABC)

FDS	AB $\rightarrow$ C	collection	Book
	C $\rightarrow$ ABDE	NOW, (ABC)	(ABCDEF)
		AB $\rightarrow$ C	AB $\rightarrow$ C
	DAE $\rightarrow$ F	C $\rightarrow$ ABDE	C $\rightarrow$ AB



= 3NF

Book = 2NF
Collection = 3NF

16. GATE IT-08

(6)

Let R(ABCDEF) be a functional dependency

$P \supseteq C$  and  $B \supseteq G$

a) BCNF b) 3NF

Now, Carddard

Now,

Now, the func

now

.. Not in

Functional dependency

transitive

dependency

2NF

3NF

BCNF

transitive

dependency

2NF

3NF

BCNF

transitive

dependency

2NF

3NF

BCNF

(66)

### 16. GATE IT-08 AND 2013 QUESTION ON NORMALISATION

Let  $R(ABCDEF\{g\})$  be a Relational schema in which the following functional dependencies are known to hold  $AB \rightarrow CD, DE \rightarrow P, C \rightarrow E$ ,  $P \rightarrow C$  and  $B \rightarrow g$ . The Relation schema 'R' is in

- a) BCNF  
b) 3NF but not in BCNF  
c) 2NF but not in 3NF

Now Candidate Key =  $(AB)^+ = (ABCDEF^*)$

Now,  $(AB)^+ = (ABCDEF\{g\})$  :  $(AB)^+$  is the candidate key

The

Now the functional dependences are

$$\begin{array}{l} AB \rightarrow CD \\ DE \rightarrow P \\ C \rightarrow E \\ P \rightarrow C \\ \boxed{B \rightarrow g} \end{array}$$

Transitive dependency  
Partial dependency

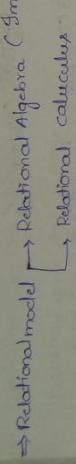
$\therefore$  Not in 2NF

Normal forms  
 1. 1NF  
 2. 2NF  
 3. 3NF  
 4. BCNF  
 5. 4NF  
 6. 5NF  
 7. 6NF  
 8. 7NF  
 9. 8NF  
 10. 9NF  
 11. 10NF  
 12. 11NF  
 13. 12NF  
 14. 13NF  
 15. 14NF  
 16. 15NF  
 17. 16NF  
 18. 17NF  
 19. 18NF  
 20. 19NF  
 21. 20NF  
 22. 21NF  
 23. 22NF  
 24. 23NF  
 25. 24NF  
 26. 25NF  
 27. 26NF  
 28. 27NF  
 29. 28NF  
 30. 29NF  
 31. 30NF  
 32. 31NF  
 33. 32NF  
 34. 33NF  
 35. 34NF  
 36. 35NF  
 37. 36NF  
 38. 37NF  
 39. 38NF  
 40. 39NF  
 41. 40NF  
 42. 41NF  
 43. 42NF  
 44. 43NF  
 45. 44NF  
 46. 45NF  
 47. 46NF  
 48. 47NF  
 49. 48NF  
 50. 49NF  
 51. 50NF  
 52. 51NF  
 53. 52NF  
 54. 53NF  
 55. 54NF  
 56. 55NF  
 57. 56NF  
 58. 57NF  
 59. 58NF  
 60. 59NF  
 61. 60NF  
 62. 61NF  
 63. 62NF  
 64. 63NF  
 65. 64NF  
 66. 65NF  
 67. 66NF  
 68. 67NF  
 69. 68NF  
 70. 69NF  
 71. 70NF  
 72. 71NF  
 73. 72NF  
 74. 73NF  
 75. 74NF  
 76. 75NF  
 77. 76NF  
 78. 77NF  
 79. 78NF  
 80. 79NF  
 81. 80NF  
 82. 81NF  
 83. 82NF  
 84. 83NF  
 85. 84NF  
 86. 85NF  
 87. 86NF  
 88. 87NF  
 89. 88NF  
 90. 89NF  
 91. 90NF  
 92. 91NF  
 93. 92NF  
 94. 93NF  
 95. 94NF  
 96. 95NF  
 97. 96NF  
 98. 97NF  
 99. 98NF  
 100. 99NF  
 101. 100NF  
 102. 101NF  
 103. 102NF  
 104. 103NF  
 105. 104NF  
 106. 105NF  
 107. 106NF  
 108. 107NF  
 109. 108NF  
 110. 109NF  
 111. 110NF  
 112. 111NF  
 113. 112NF  
 114. 113NF  
 115. 114NF  
 116. 115NF  
 117. 116NF  
 118. 117NF  
 119. 118NF  
 120. 119NF  
 121. 120NF  
 122. 121NF  
 123. 122NF  
 124. 123NF  
 125. 124NF  
 126. 125NF  
 127. 126NF  
 128. 127NF  
 129. 128NF  
 130. 129NF  
 131. 130NF  
 132. 131NF  
 133. 132NF  
 134. 133NF  
 135. 134NF  
 136. 135NF  
 137. 136NF  
 138. 137NF  
 139. 138NF  
 140. 139NF  
 141. 140NF  
 142. 141NF  
 143. 142NF  
 144. 143NF  
 145. 144NF  
 146. 145NF  
 147. 146NF  
 148. 147NF  
 149. 148NF  
 150. 149NF  
 151. 150NF  
 152. 151NF  
 153. 152NF  
 154. 153NF  
 155. 154NF  
 156. 155NF  
 157. 156NF  
 158. 157NF  
 159. 158NF  
 160. 159NF  
 161. 160NF  
 162. 161NF  
 163. 162NF  
 164. 163NF  
 165. 164NF  
 166. 165NF  
 167. 166NF  
 168. 167NF  
 169. 168NF  
 170. 169NF  
 171. 170NF  
 172. 171NF  
 173. 172NF  
 174. 173NF  
 175. 174NF  
 176. 175NF  
 177. 176NF  
 178. 177NF  
 179. 178NF  
 180. 179NF  
 181. 180NF  
 182. 181NF  
 183. 182NF  
 184. 183NF  
 185. 184NF  
 186. 185NF  
 187. 186NF  
 188. 187NF  
 189. 188NF  
 190. 189NF  
 191. 190NF  
 192. 191NF  
 193. 192NF  
 194. 193NF  
 195. 194NF  
 196. 195NF  
 197. 196NF  
 198. 197NF  
 199. 198NF  
 200. 199NF  
 201. 200NF  
 202. 201NF  
 203. 202NF  
 204. 203NF  
 205. 204NF  
 206. 205NF  
 207. 206NF  
 208. 207NF  
 209. 208NF  
 210. 209NF  
 211. 210NF  
 212. 211NF  
 213. 212NF  
 214. 213NF  
 215. 214NF  
 216. 215NF  
 217. 216NF  
 218. 217NF  
 219. 218NF  
 220. 219NF  
 221. 220NF  
 222. 221NF  
 223. 222NF  
 224. 223NF  
 225. 224NF  
 226. 225NF  
 227. 226NF  
 228. 227NF  
 229. 228NF  
 230. 229NF  
 231. 230NF  
 232. 231NF  
 233. 232NF  
 234. 233NF  
 235. 234NF  
 236. 235NF  
 237. 236NF  
 238. 237NF  
 239. 238NF  
 240. 239NF  
 241. 240NF  
 242. 241NF  
 243. 242NF  
 244. 243NF  
 245. 244NF  
 246. 245NF  
 247. 246NF  
 248. 247NF  
 249. 248NF  
 250. 249NF  
 251. 250NF  
 252. 251NF  
 253. 252NF  
 254. 253NF  
 255. 254NF  
 256. 255NF  
 257. 256NF  
 258. 257NF  
 259. 258NF  
 260. 259NF  
 261. 260NF  
 262. 261NF  
 263. 262NF  
 264. 263NF  
 265. 264NF  
 266. 265NF  
 267. 266NF  
 268. 267NF  
 269. 268NF  
 270. 269NF  
 271. 270NF  
 272. 271NF  
 273. 272NF  
 274. 273NF  
 275. 274NF  
 276. 275NF  
 277. 276NF  
 278. 277NF  
 279. 278NF  
 280. 279NF  
 281. 280NF  
 282. 281NF  
 283. 282NF  
 284. 283NF  
 285. 284NF  
 286. 285NF  
 287. 286NF  
 288. 287NF  
 289. 288NF  
 290. 289NF  
 291. 290NF  
 292. 291NF  
 293. 292NF  
 294. 293NF  
 295. 294NF  
 296. 295NF  
 297. 296NF  
 298. 297NF  
 299. 298NF  
 300. 299NF  
 301. 300NF  
 302. 301NF  
 303. 302NF  
 304. 303NF  
 305. 304NF  
 306. 305NF  
 307. 306NF  
 308. 307NF  
 309. 308NF  
 310. 309NF  
 311. 310NF  
 312. 311NF  
 313. 312NF  
 314. 313NF  
 315. 314NF  
 316. 315NF  
 317. 316NF  
 318. 317NF  
 319. 318NF  
 320. 319NF  
 321. 320NF  
 322. 321NF  
 323. 322NF  
 324. 323NF  
 325. 324NF  
 326. 325NF  
 327. 326NF  
 328. 327NF  
 329. 328NF  
 330. 329NF  
 331. 330NF  
 332. 331NF  
 333. 332NF  
 334. 333NF  
 335. 334NF  
 336. 335NF  
 337. 336NF  
 338. 337NF  
 339. 338NF  
 340. 339NF  
 341. 340NF  
 342. 341NF  
 343. 342NF  
 344. 343NF  
 345. 344NF  
 346. 345NF  
 347. 346NF  
 348. 347NF  
 349. 348NF  
 350. 349NF  
 351. 350NF  
 352. 351NF  
 353. 352NF  
 354. 353NF  
 355. 354NF  
 356. 355NF  
 357. 356NF  
 358. 357NF  
 359. 358NF  
 360. 359NF  
 361. 360NF  
 362. 361NF  
 363. 362NF  
 364. 363NF  
 365. 364NF  
 366. 365NF  
 367. 366NF  
 368. 367NF  
 369. 368NF  
 370. 369NF  
 371. 370NF  
 372. 371NF  
 373. 372NF  
 374. 373NF  
 375. 374NF  
 376. 375NF  
 377. 376NF  
 378. 377NF  
 379. 378NF  
 380. 379NF  
 381. 380NF  
 382. 381NF  
 383. 382NF  
 384. 383NF  
 385. 384NF  
 386. 385NF  
 387. 386NF  
 388. 387NF  
 389. 388NF  
 390. 389NF  
 391. 390NF  
 392. 391NF  
 393. 392NF  
 394. 393NF  
 395. 394NF  
 396. 395NF  
 397. 396NF  
 398. 397NF  
 399. 398NF  
 400. 399NF  
 401. 400NF  
 402. 401NF  
 403. 402NF  
 404. 403NF  
 405. 404NF  
 406. 405NF  
 407. 406NF  
 408. 407NF  
 409. 408NF  
 410. 409NF  
 411. 410NF  
 412. 411NF  
 413. 412NF  
 414. 413NF  
 415. 414NF  
 416. 415NF  
 417. 416NF  
 418. 417NF  
 419. 418NF  
 420. 419NF  
 421. 420NF  
 422. 421NF  
 423. 422NF  
 424. 423NF  
 425. 424NF  
 426. 425NF  
 427. 426NF  
 428. 427NF  
 429. 428NF  
 430. 429NF  
 431. 430NF  
 432. 431NF  
 433. 432NF  
 434. 433NF  
 435. 434NF  
 436. 435NF  
 437. 436NF  
 438. 437NF  
 439. 438NF  
 440. 439NF  
 441. 440NF  
 442. 441NF  
 443. 442NF  
 444. 443NF  
 445. 444NF  
 446. 445NF  
 447. 446NF  
 448. 447NF  
 449. 448NF  
 450. 449NF  
 451. 450NF  
 452. 451NF  
 453. 452NF  
 454. 453NF  
 455. 454NF  
 456. 455NF  
 457. 456NF  
 458. 457NF  
 459. 458NF  
 460. 459NF  
 461. 460NF  
 462. 461NF  
 463. 462NF  
 464. 463NF  
 465. 464NF  
 466. 465NF  
 467. 466NF  
 468. 467NF  
 469. 468NF  
 470. 469NF  
 471. 470NF  
 472. 471NF  
 473. 472NF  
 474. 473NF  
 475. 474NF  
 476. 475NF  
 477. 476NF  
 478. 477NF  
 479. 478NF  
 480. 479NF  
 481. 480NF  
 482. 481NF  
 483. 482NF  
 484. 483NF  
 485. 484NF  
 486. 485NF  
 487. 486NF  
 488. 487NF  
 489. 488NF  
 490. 489NF  
 491. 490NF  
 492. 491NF  
 493. 492NF  
 494. 493NF  
 495. 494NF  
 496. 495NF  
 497. 496NF  
 498. 497NF  
 499. 498NF  
 500. 499NF  
 501. 500NF  
 502. 501NF  
 503. 502NF  
 504. 503NF  
 505. 504NF  
 506. 505NF  
 507. 506NF  
 508. 507NF  
 509. 508NF  
 510. 509NF  
 511. 510NF  
 512. 511NF  
 513. 512NF  
 514. 513NF  
 515. 514NF  
 516. 515NF  
 517. 516NF  
 518. 517NF  
 519. 518NF  
 520. 519NF  
 521. 520NF  
 522. 521NF  
 523. 522NF  
 524. 523NF  
 525. 524NF  
 526. 525NF  
 527. 526NF  
 528. 527NF  
 529. 528NF  
 530. 529NF  
 531. 530NF  
 532. 531NF  
 533. 532NF  
 534. 533NF  
 535. 534NF  
 536. 535NF  
 537. 536NF  
 538. 537NF  
 539. 538NF  
 540. 539NF  
 541. 540NF  
 542. 541NF  
 543. 542NF  
 544. 543NF  
 545. 544NF  
 546. 545NF  
 547. 546NF  
 548. 547NF  
 549. 548NF  
 550. 549NF  
 551. 550NF  
 552. 551NF  
 553. 552NF  
 554. 553NF  
 555. 554NF  
 556. 555NF  
 557. 556NF  
 558. 557NF  
 559. 558NF  
 560. 559NF  
 561. 560NF  
 562. 561NF  
 563. 562NF  
 564. 563NF  
 565. 564NF  
 566. 565NF  
 567. 566NF  
 568. 567NF  
 569. 568NF  
 570. 569NF  
 571. 570NF  
 572. 571NF  
 573. 572NF  
 574. 573NF  
 575. 574NF  
 576. 575NF  
 577. 576NF  
 578. 577NF  
 579. 578NF  
 580. 579NF  
 581. 580NF  
 582. 581NF  
 583. 582NF  
 584. 583NF  
 585. 584NF  
 586. 585NF  
 587. 586NF  
 588. 587NF  
 589. 588NF  
 590. 589NF  
 591. 590NF  
 592. 591NF  
 593. 592NF  
 594. 593NF  
 595. 594NF  
 596. 595NF  
 597. 596NF  
 598. 597NF  
 599. 598NF  
 600. 599NF  
 601. 600NF  
 602. 601NF  
 603. 602NF  
 604. 603NF  
 605. 604NF  
 606. 605NF  
 607. 606NF  
 608. 607NF  
 609. 608NF  
 610. 609NF  
 611. 610NF  
 612. 611NF  
 613. 612NF  
 614. 613NF  
 615. 614NF  
 616. 615NF  
 617. 616NF  
 618. 617NF  
 619. 618NF  
 620. 619NF  
 621. 620NF  
 622. 621NF  
 623. 622NF  
 624. 623NF  
 625. 624NF  
 626. 625NF  
 627. 626NF  
 628. 627NF  
 629. 628NF  
 630. 629NF  
 631. 630NF  
 632. 631NF  
 633. 632NF  
 634. 633NF  
 635. 634NF  
 636. 635NF  
 637. 636NF  
 638. 637NF  
 639. 638NF  
 640. 639NF  
 641. 640NF  
 642. 641NF  
 643. 642NF  
 644. 643NF  
 645. 644NF  
 646. 645NF  
 647. 646NF  
 648. 647NF  
 649. 648NF  
 650. 649NF  
 651. 650NF  
 652. 651NF  
 653. 652NF  
 654. 653NF  
 655. 654NF  
 656. 655NF  
 657. 656NF  
 658. 657NF  
 659. 658NF  
 660. 659NF  
 661. 660NF  
 662. 661NF  
 663. 662NF  
 664. 663NF  
 665. 664NF  
 666. 665NF  
 667. 666NF  
 668. 667NF  
 669. 668NF  
 670. 669NF  
 671. 670NF  
 672. 671NF  
 673. 672NF  
 674. 673NF  
 675. 674NF  
 676. 675NF  
 677. 676NF  
 678. 677NF  
 679. 678NF  
 680. 679NF  
 681. 680NF  
 682. 681NF  
 683. 682NF  
 684. 683NF  
 685. 684NF  
 686. 685NF  
 687. 686NF  
 688. 687NF  
 689. 688NF  
 690. 689NF  
 691. 690NF  
 692. 691NF  
 693. 692NF  
 694. 693NF  
 695. 694NF  
 696. 695NF  
 697. 696NF  
 698. 697NF  
 699. 698NF  
 700. 699NF  
 701. 700NF  
 702. 701NF  
 703. 702NF  
 704. 703NF  
 705. 704NF  
 706. 705NF  
 707. 706NF  
 708. 707NF  
 709. 708NF  
 710. 709NF  
 711. 710NF  
 712. 711NF  
 713. 712NF  
 714. 713NF  
 715. 714NF  
 716. 715NF  
 717. 716NF  
 718. 717NF  
 719. 718NF  
 720. 719NF  
 721. 720NF  
 722. 721NF  
 723. 722NF  
 724. 723NF  
 725. 724NF  
 726. 725NF  
 727. 726NF  
 728. 727NF  
 729. 728NF  
 730. 729NF  
 731. 730NF  
 732. 731NF  
 733. 732NF  
 734. 733NF  
 735. 734NF  
 736. 735NF  
 737. 736NF  
 738. 737NF  
 739. 738NF  
 740. 739NF  
 741. 740NF  
 742. 741NF  
 743. 742NF  
 744. 743NF  
 745. 744NF  
 746. 745NF  
 747. 746NF  
 748. 747NF  
 749. 748NF  
 750. 749NF  
 751. 750NF  
 752. 751NF  
 753. 752NF  
 754. 753NF  
 755. 754NF  
 756. 755NF  
 757. 756NF  
 758. 757NF  
 759. 758NF  
 760. 759NF  
 761. 760NF  
 762. 761NF  
 763. 762NF  
 764. 763NF  
 765. 764NF  
 766. 765NF  
 767. 766NF  
 768. 767NF  
 769. 768NF  
 770. 769NF  
 771. 770NF  
 772. 771NF  
 773. 772NF  
 774. 773NF  
 775. 774NF  
 776. 775NF  
 777. 776NF  
 778. 777NF  
 779. 778NF  
 780. 779NF  
 781. 780NF  
 782. 781NF  
 783. 782NF  
 784. 783NF  
 785. 784NF  
 786. 785NF  
 787. 786NF  
 788. 787NF  
 789. 788NF  
 790. 789NF  
 791. 790NF  
 792. 791NF  
 793. 792NF  
 794. 793NF  
 795. 794NF  
 796. 795NF  
 797. 796NF  
 798. 797NF  
 799. 798NF  
 800. 799NF  
 801. 800NF  
 802. 801NF  
 803. 802NF  
 804. 803NF  
 805. 804NF  
 806. 805NF  
 807. 806NF  
 808. 807NF  
 809. 808NF  
 810. 809NF  
 811. 810NF  
 812. 811NF  
 813. 812NF  
 814. 813NF  
 815. 814NF  
 816. 815NF  
 817. 816NF  
 818. 817NF  
 819. 818NF  
 820. 819NF  
 821. 820NF  
 822. 821NF  
 823. 822NF  
 824. 823NF  
 825. 824NF  
 826. 825NF  
 827. 826NF  
 828. 827NF  
 829. 828NF  
 830. 829NF  
 831. 830NF  
 832. 831NF  
 833. 832NF  
 834. 833NF  
 835. 834NF  
 836. 835NF  
 837. 836NF  
 838. 837NF  
 839. 838NF  
 840. 839NF  
 841. 840NF  
 842. 841NF  
 843. 842NF  
 844. 843NF

## 5. RELATIONAL ALGEBRA

### INTRODUCTION TO RELATIONAL ALGEBRA

- ⇒ Every data model is supposed to provide a way to manipulate the data.
- ⇒ Relational model → Relational Algebra (Temp for STATE)



- ⇒ Relational Algebra is a procedural language (what we want & how to get it)  
Relational calculus is a declarative language (what we want)
- ⇒ Relational Algebra → Operations (smallest unit of work that we can perform on any Relation)

$$\Rightarrow RA = RC + Set\ operations$$

### Operations

$\pi$  : Projection (Selecting columns)

$\sigma$  : Selection (Selecting Rows)

$\times$  : Cross product (Between two tables) (combine 2 tables)

(By using cross product we can work on more tuples simultaneously.)

$\cup$  : Union (Same as union of sets)

$-$  : Minus (Set difference)

$\circ$  : Rename (change name of the attribute(s))

### Relational Algebra Expressions

⇒ Sequence of operations

$\Rightarrow \cap$  : Intersection  $A \cap B = A - (A - B)$

$\Rightarrow \bowtie$  : Join ( $\alpha \times \beta$ )

$\Rightarrow /$  : Division ( $\pi, x, -$ )

### The Syntax



## SELECTION OPERATION ( $\sigma$ ):

(6)

- ⇒ This operation is used to choose a subset of tuples (Rows).
- ⇒ Selection / Horizontal partition manipulation.

Emp	ENO	Employee Name	DNO	Salary
1	1000	John	1	10000
2	1001	Mike	2	12000
3	1002	David	3	15000
4	1003	Sarah	4	18000

$$\Rightarrow \sigma_{DNO=4} (Emp) \Rightarrow \text{gives list of all employees who belong to DNO=4}$$

whose salary is > 10,000.

$$\Rightarrow \sigma_{Sal > 10000} (Emp) \Rightarrow \text{gives list of all employees whose salary is > 10,000 and who belong to DNO=4}$$

$$\Rightarrow \sigma_{Sal > 10000} (\sigma_{DNO=4} (Emp)) \Rightarrow$$

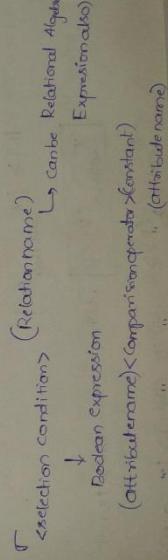
$$= \sigma_{Sal > 10000 \text{ AND } DNO=4} (Emp) \Rightarrow$$

⇒ Select operation works on only one tuple at a time.

- ⇒ Let us consider a Relation 'R' and it represents "no. of tuples in the Relation 'R' then

$$|R| \geq \left[ \left( \sigma_c |R| \right) \right]_{\max |R|, \min \sigma^* \text{ tuples}}$$

The syntax of selection operation is



### 3. PROJECTION OPERATION ( $\pi$ ) :

⇒ Used to choose a subset of columns, and the output will always be  $\pi_{ABC}(R)$

a relation.

⇒ projection/vertical partitioning

⇒ Degree of a table = No. of attributes in that table

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

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

⇒ General criteria of projection operation is

$\pi_{\text{Attribute list}}(R)$  , can be Relational Algebra Expression

↓  
subset of attributes present in R.

⇒  $\pi^n$  operation eliminates duplicates ( Duplicate elimination )

(AB) = Candidate key

C	C
1	a
2	b

$\pi_A(R)$  :

A
1
2

Representation -

$\pi_{ABC}(R)$ : Entire table

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

⇒ Consider a Relation  $R'$  and the tuples be represented by  $|R'|$   
then,

$|R| \geq |\pi_{\text{Attribs}}(R)|$

$\pi_{\text{Attribs}}(R) < R$ , where attributes > is not a superkey

⇒ sometimes if  
then the  $\pi_{\text{Attribs}}(R) = R$ , when attribute is a superkey  
⇒ Renaming o

⇒ sometimes +  
attribute of

⇒ sometimes if  
then the  $\pi_{\text{Attribs}}(R) = R$ , when attribute is a superkey  
⇒ Renaming o

⑥ The projection operation is not "commutative."

always  $\pi_A(\pi_B(R)) \neq \pi_B(\pi_A(R))$

Now,  $\pi_A(\pi_{AB}(R)) =$

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

Now,  $\pi_{AB}(\pi_A(R)) =$

A
1
2
3

AB not possible.

Let R be a Relation, then  $\pi_{A_2}(\pi_{A_1}(R))$  is valid only when  $A_1 \subseteq A_2$ .

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

whereas condition satisfy

then we can directly write as  $\pi_{A_2}(R)$ .

⇒ The "projection" operation is same as "select" operation in SQL with distinct.

#### 4. RENAME OPERATION (ε)

Now, consider a Relation R(ABC)

R

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

Now, if need to get the AB columns satisfying a condition x then the query is  $\pi_{AB}(\tau_x(R))$ , what some people do is instead of writing in single line (shown) representation they represent in different table.

use

Temp  $\leftarrow \tau_x(R)$   
Answer  $\leftarrow \pi_{AB}(\text{Temp})$

⇒ sometimes there might be a need that you want to Rename the attributes of a table, then the syntax is  $\epsilon_{S(R)}^{\text{NewName}}$  and table name from R' to's' [ R'(ABC)  $\rightarrow$  S'(AS, VS) ]

- ⑦ Sometimes you might want to Rename the table not the attributes then the syntax is  $\epsilon_S(R)$
- ⑧ Renaming operator in SQL is "AS".

### 5. GATE '98 QUESTION ON SELECTION AND PROJECTION

Which of the query transformations (by replacing the RHS expression by the RHS conjunction) is incorrect?  $R_1$  and  $R_2$  are relations  $C_1, C_2$  are selection conditions and  $A_1, A_2$  are attributes of  $R_1$

- $\sigma_{C_1}(R_1) \rightarrow \sigma_{C_2}(\pi_{A_2}(R_1))$  - False
- $\pi_{A_1}(\sigma_{C_1}(R_1)) \rightarrow \pi_{A_1}(\sigma_{C_1}(R_1)) = \text{True}$
- $\pi_{C_1}(R_1 \cup R_2) \rightarrow \pi_{C_1}(R_1) \cup \pi_{C_2}(R_2) \Rightarrow \text{TRUE}$
- $\pi_{A_2}(\pi_{C_1}(R_1)) \rightarrow \pi_{A_2}(\pi_{C_1}(R_1)) \Rightarrow \text{FALSE}$

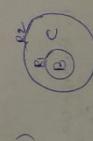
If  $C_1$  condition is in such a way that it contains  $A_1, A_2$  values in columns, we can't calculate  $A_2$ .

### 6. GATE 2012 QUESTION ON PROJECTION

Suppose  $R_1(AB)$  and  $R_2(CD)$  are two relation schemas. Let  $r_1$  and  $r_2$  be the corresponding relations. 'B' is the Foreign Key that occurs in  $R_2$ . If data in  $R_1$  and  $R_2$  satisfy Referential Integrity Constraints, what the following is always True?

- $\pi_B(r_1) - \pi_C(r_2) = \phi$
- $\pi_B(r_1) - \pi_B(r_2) = \phi$
- $\pi_B(r_1) - \pi_C(r_2) \neq \phi$

Sol: 'B' is the foreign key of 'C' in  $R_2 \Rightarrow$  Every value of 'B' will be in 'C'.



B)



$\pi_B(r_1) - \pi_B(r_2) \neq \phi$   $\Rightarrow$  Bis the tab of condition of option, if every value of 'B' will be present in 'C'.

### 7. SET OPERATIONS

$\Rightarrow$  The next  $\Rightarrow$  when we should

$$R(A_1, A_2, \dots, A_n, B_1, B_2, \dots, B_m)$$

Now, (ROS)

$\therefore$  The Union  $\therefore$  The Difference

$\Rightarrow$  Intersection  $\Rightarrow$  Join

### 8. CATEGORISATION

i) Binary operation  
ii) Relations in

$\therefore$  Emp

$\Rightarrow$  If the tab root attri  $\Rightarrow$  If  $R_1$  has

$$D) \pi_B(r_1) - \pi_C(r_2) \neq \phi$$

of condition of option,

## SET OPERATIONS

session by  
the adie

#3  
 The next set of operations are union ( $\cup$ ), intersection ( $\cap$ ), Min<sub>(l)</sub>  
 when we apply the above operations on Relations then they  
 should be "union compatible" or "Type compatible".

$R(A_1 A_2 \dots A_n)$       i) have some degree  
 $S(B_1 B_2 \dots B_n)$       ii) covers bonding elements

Now,  $(R \cup S) = R(A_1, A_2, \dots, A_n)$

we use the this set name of ROS = R( $\beta_1, \beta_2, \dots, \beta_n$ )

The Union operator is commutative =  $\text{RUS} = \text{SUR}$  [No duplicates]

values

all difference operators & not commutative =  $(R-S) \neq (S-R)$  [No duplicates]

which of

8. CARTESIAN

### ① Binary operation

① Relations need not be Union compatible

5

<u>Dependent</u>	D	E
B	G	C

- ⇒ if the table  $R_1$  contains  $n_1$  attributes and table  $R_2$  contains  $n_2$  attributes  
then  $R_1 \times R_2$  contains  $(n_1 \times n_2)$  attributes (columns)
- if  $R_1$  has  $m_1$  tuples and  $R_2$  has  $m_2$  tuples then  $R_1 \times R_2$  has  $(m_1 \times m_2)$  tuples

Ex:

R(ABC)

A	B	C	D	E
1	b1	c1	1	a1
2	b2	c2	1	a2
3	b3	c3	2	a3

(m) rows = 3      (n) columns = 2      (n<sub>1</sub>) columns = 2      (n<sub>2</sub>) columns = 2

⇒ using a cross product alone is meaningful, using it with some condition

⇒  $\sigma_{A=D} (Employee \times Dependent) \Rightarrow$

A	B	C	D	E
1	b1	c1	1	a1
2	b2	c2	2	a2

i. A new operation called join  $\bowtie$  is developed specifying the join condition with a condition.

ii. Join is a combination of ( $\times$  and  $\sigma$ ).

### 9. JOIN AND NATURAL JOIN

⇒ "Join" operation is used to combine two operations / Relations into a single larger tuple.

Ex:

R

A	B	C	D	E
1	b1	c1	1	a1
2	b2	c2	2	a2

S

A	B	C	D	E
3	b3	c3	1	a3
4	b4	c4	2	a4

(R  $\bowtie$  S)

A	B	C	D	E
1	b1	c1	1	a1
2	b2	c2	2	a2
3	b3	c3	1	a3
4	b4	c4	2	a4

Related tuples from R and S are combined based on common column values.

Ex:

R

A	B	C	D	E
1	b1	c1	1	a1
2	b2	c2	2	a2

S

A	B	C	D	E
3	b3	c3	1	a3
4	b4	c4	2	a4

(R  $\bowtie$  S)

A	B	C	D	E
1	b1	c1	1	a1
2	b2	c2	2	a2
3	b3	c3	1	a3
4	b4	c4	2	a4

⇒ Some of the questions that will be asked in GATE are:

join two tables what are the no. of tuples in the resulting new relation.

Ex:

R

A	B	C	D	E
1	b1	c1	1	a1
2	b2	c2	2	a2

S

A	B	C	D	E
3	b3	c3	1	a3
4	b4	c4	2	a4

(R  $\bowtie$  S)

A	B	C	D	E
1	b1	c1	1	a1
2	b2	c2	2	a2
3	b3	c3	1	a3
4	b4	c4	2	a4

⇒ Some of the questions that will be asked in GATE are:

join two tables what are the no. of tuples in the resulting new relation.

1. JOIN

Ex:  $R(ABC)$        $S(CDE)$        $(R \bowtie S)_{A=B}$

E	A	B	C
a <sub>1</sub>	1	a	j
a <sub>2</sub>	2	b	e
a <sub>1</sub>	1	a	f
a <sub>2</sub>	2	b	c
a <sub>1</sub>	2	c	d

D	E
1	a
2	b
2	b
2	b

The syntax of Join is  $[R \bowtie S] <condition>$

$\downarrow$   $\langle condition \rangle$  AND  $\langle condition \rangle$

$\downarrow$   $R_1$  AND  $R_2$   $\Rightarrow$  This join is called "Theta" join (no comparison against with values, comparison only between attributes)

Now, some conditions are frequently used while performing join operation and so we don't specify the condition and they are:

- Automatically derived from the Relations such kind of join is called natural join denoted by (\*).
- The join operation is automatically performed on matching names.

2. DIVISION

Condition:  $R$  is divided into two relations  $R_1$  and  $R_2$  such that  $R = R_1 \cup R_2$  and  $R_1 \cap R_2 = \emptyset$ .

Ex:  $R(ABC)$        $S(A)$

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

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

$R \div S$

$R \div S \Rightarrow [R : S]$   $\downarrow$   $R : S \Leftrightarrow \begin{cases} A = A \\ B = B \end{cases}$   $\downarrow$   $S$

"Adjustable width"

3. DIVISION OPERATION ( $\div$ )

$R \div S(R-S)$

A	B
a <sub>1</sub>	
a <sub>2</sub>	

A
b <sub>1</sub>

If you are doing now

$R \div S \Rightarrow$  what even attributes we have in 'S' you subtract them from 'R' and write what is remaining in 'R'.  
 $\Rightarrow$  The attributes of  $(R-S)$  will be in  $R \div S$ .

Ex:

R      S      R÷S

A	
a1	b1
a2	b1
a3	b1
a4	b2

This table contains the values of 'B' that are associated with the corresponding values of 'S'.

⇒ Here 'b1' is associated with all the attributes (a1, a2) of S so b1 will be present in (R÷S)

Ex:

R      S      R÷S

A	
a1	b1
a2	b1
a3	b1
a4	b1
a1	b2
a3	b2
a2	b3
a3	b3
a4	b3
a1	b4
a1	b4
a3	b4

$\Rightarrow A$   
Attribute  
m

$\Rightarrow R$   
Relation  
associated with  $a_2, a_3$

Attributes of S:

These are the values associated with  $a_2, a_3$

$\Rightarrow S$   
Relation

U. Answer

The Agree

COUNT

$\Rightarrow$  Another a mainly p perform are miss join any

B	
b1	
b2	

$S \times B - R$

B	
b1	a1
b2	a1

$S \times B - R$

B	
b1	a2
b2	a2

$S \times B - R$

B	
b1	a3
b2	a3

$S \times B - R$

B	
b1	a4
b2	a4

$S \times B - R$

B	
b1	a1
b2	a1

$S \times B - R$

B	
b1	a2
b2	a2

$S \times B - R$

B	
b1	a3
b2	a3

$S \times B - R$

B	
b1	a4
b2	a4

$S \times B - R$

B	
b1	a1
b2	a1

$S \times B - R$

B	
b1	a2
b2	a2

$S \times B - R$

B	
b1	a3
b2	a3

$S \times B - R$

B	
b1	a4
b2	a4

$S \times B - R$

B	
b1	a1
b2	a1

$S \times B - R$

B	
b1	a2
b2	a2

$S \times B - R$

B	
b1	a3
b2	a3

$S \times B - R$

B	
b1	a4
b2	a4

$S \times B - R$

B	
b1	a1
b2	a1

$S \times B - R$

B	
b1	a2
b2	a2

$S \times B - R$

B	
b1	a3
b2	a3

$S \times B - R$

B	
b1	a4
b2	a4

$S \times B - R$

B	
b1	a1
b2	a1

$S \times B - R$

B	
b1	a2
b2	a2

$S \times B - R$

B	
b1	a3
b2	a3

$S \times B - R$

B	
b1	a4
b2	a4

$S \times B - R$

B	
b1	a1
b2	a1

$S \times B - R$

B	
b1	a2
b2	a2

$S \times B - R$

B	
b1	a3
b2	a3

$S \times B - R$

B	
b1	a4
b2	a4

$S \times B - R$

B	
b1	a1
b2	a1

$S \times B - R$

B	
b1	a2
b2	a2

$S \times B - R$

B	
b1	a3
b2	a3

$S \times B - R$

B	
b1	a4
b2	a4

$S \times B - R$

B	
b1	a1
b2	a1

$S \times B - R$

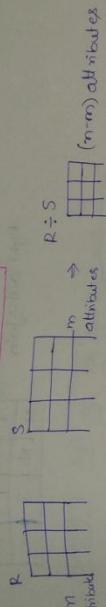
B	
b1	a2
b2	a2

$S$

76. The implementation of division using the basic operations is

$$\begin{aligned} T_1 &\leftarrow \pi_{(R)}(R) \\ T_2 &\leftarrow \pi_{(R)}[(S \times T_1) - R] \\ T &\leftarrow T_1 \setminus T_2 \end{aligned}$$

In the values associated with  $T_2$  values of  $S$ ,  
of  $S \times T_1$  will



Attributes in  $S \subseteq$  Attributes in  $R$

$\Rightarrow$  If  $R'$  contains  $X$ -attributes,  $S$  contains  $Y$  attributes then the

Relation  $Z = (R \div S)$  contains  $X-Y$  attributes

$$\begin{aligned} Z &= (X) - (Y) \\ \Rightarrow X &= Z \cup Y \end{aligned}$$

and the values  
d with  $a_{12}, a_{23} =$   
 $\lambda$  of  $S$ :

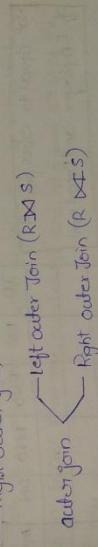
then there will be  
values in  $R$  which  
are not present in  $S$ .

## II. AGGREGATE FUNCTIONS AND OUTER JOINS

The Aggregate functions are SUM, AVERAGE, MAXIMUM, MINIMUM,  
COUNT.

$\Rightarrow$  Another additional function provided is outer join. outerjoin is mainly introduced because we may miss some tuples when we perform Natural join (tuples which are not satisfying a condition are missed). The outer join is of 2 types they are left outer join and Right outer join

T <sub>2</sub>	B
a	b1
a	b2
b	b1
b	b2



$\Rightarrow$  By performing left outer join the output contains all the matching tuples in  $R$  and  $S$  and also the non matching tuples in the left side Relation.

<p><u>Ex:</u></p> <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>R</td><td>A</td><td>C</td></tr> <tr><td>S</td><td>B</td><td>D</td></tr> <tr><td>1</td><td>-</td><td>-</td></tr> <tr><td>2</td><td>-</td><td>-</td></tr> <tr><td>3</td><td>-</td><td>-</td></tr> </table>	R	A	C	S	B	D	1	-	-	2	-	-	3	-	-	<p>Now, <math>(R \Delta S)</math></p> <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>A</td><td>C</td><td>B</td><td>D</td></tr> <tr><td>1</td><td>-</td><td>-</td><td>-</td></tr> <tr><td>2</td><td>-</td><td>N</td><td>N</td></tr> <tr><td>3</td><td>-</td><td>-</td><td>-</td></tr> </table> <p><math>(R \Delta S) =</math></p>	A	C	B	D	1	-	-	-	2	-	N	N	3	-	-	-
R	A	C																														
S	B	D																														
1	-	-																														
2	-	-																														
3	-	-																														
A	C	B	D																													
1	-	-	-																													
2	-	N	N																													
3	-	-	-																													
<p><u>Ex:</u></p> <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>ID</td><td>EN</td><td>ENO</td></tr> <tr><td>ID</td><td>DN</td><td>MID</td></tr> <tr><td>1</td><td>-</td><td>-</td></tr> <tr><td>2</td><td>-</td><td>-</td></tr> <tr><td>3</td><td>-</td><td>-</td></tr> </table>	ID	EN	ENO	ID	DN	MID	1	-	-	2	-	-	3	-	-	<p>Right Outer Join</p> <p><u>Ex:</u></p> <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>ID</td><td>EN</td><td>ENO</td></tr> <tr><td>ID</td><td>DN</td><td>MID</td></tr> <tr><td>1</td><td>-</td><td>-</td></tr> <tr><td>2</td><td>-</td><td>-</td></tr> <tr><td>3</td><td>-</td><td>-</td></tr> </table> <p>Department</p>	ID	EN	ENO	ID	DN	MID	1	-	-	2	-	-	3	-	-	
ID	EN	ENO																														
ID	DN	MID																														
1	-	-																														
2	-	-																														
3	-	-																														
ID	EN	ENO																														
ID	DN	MID																														
1	-	-																														
2	-	-																														
3	-	-																														
<p><u>Ex:</u></p> <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>ID</td><td>EN</td><td>ENO</td></tr> <tr><td>ID</td><td>DN</td><td>MID</td></tr> <tr><td>1</td><td>-</td><td>-</td></tr> <tr><td>2</td><td>-</td><td>-</td></tr> <tr><td>3</td><td>-</td><td>-</td></tr> </table>	ID	EN	ENO	ID	DN	MID	1	-	-	2	-	-	3	-	-	<p>Manager ID.</p> <p>Now,</p> <p>Tuples where MID matches with EID</p> <p>which does not match with MID but present in Emp table.</p>																
ID	EN	ENO																														
ID	DN	MID																														
1	-	-																														
2	-	-																														
3	-	-																														
<p><u>Ex:</u></p> <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>ID</td><td>EN</td><td>ENO</td></tr> <tr><td>ID</td><td>DN</td><td>MID</td></tr> <tr><td>1</td><td>-</td><td>-</td></tr> <tr><td>2</td><td>-</td><td>-</td></tr> <tr><td>3</td><td>-</td><td>-</td></tr> </table>	ID	EN	ENO	ID	DN	MID	1	-	-	2	-	-	3	-	-	<p>Emp <math>\Delta</math> Department =</p> <p><math>\langle m=ID \rangle</math></p> <p>Not matching tuples from Department table.</p> <p>Emp <math>\Delta</math> Department = Full Outer Join.</p>																
ID	EN	ENO																														
ID	DN	MID																														
1	-	-																														
2	-	-																														
3	-	-																														
<p><u>Ex:</u></p> <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>ID</td><td>EN</td><td>ENO</td><td>DN</td><td>MID</td></tr> <tr><td>ID</td><td>DN</td><td>MID</td><td></td><td></td></tr> <tr><td>1</td><td>-</td><td>-</td><td>-</td><td>-</td></tr> <tr><td>2</td><td>-</td><td>-</td><td>-</td><td>-</td></tr> <tr><td>3</td><td>-</td><td>-</td><td>-</td><td>-</td></tr> </table>	ID	EN	ENO	DN	MID	ID	DN	MID			1	-	-	-	-	2	-	-	-	-	3	-	-	-	-	<p>Given a Relational Algebra expression from <math>R \Delta S = [R \cup S] - [R \cap S]</math></p> <p>Department table</p> <p>Not Equal to Cross product of Employee and Department</p> <p>Not matching tuples from Employee table.</p> <p>Not matching tuples from Department table.</p>						
ID	EN	ENO	DN	MID																												
ID	DN	MID																														
1	-	-	-	-																												
2	-	-	-	-																												
3	-	-	-	-																												

78

	Min	Max
$\times$	$(m \times n)$	$(m \times n)$
$\bowtie$	$(0)$	$(m \times n)$
$(R \bowtie S)$	$n$	$(m \times n)$
$\bowtie^*$	$m$	$(m \times n)$
$\bowtie^{\text{max}}$	$\max(m, n)$	$(m \times n)$

79

### 12. COMPLETE SET OF RA OPERATIONS

The operations  $\{\pi, \sigma, \cup, \cap, -, \times\}$  = Complete set.

$\{\bowtie, \bowtie^*, \bowtie^{\text{max}}, \bowtie^{\text{min}}, \bowtie^{\text{not}}$   $\}$   $\Rightarrow$  can be derived from complete set.

Now,

	Min	Max
$\cup$	$\max(m, n)$	$(m+n)$
$\cap$	$0$	$\min(m, n)$
$-$	$0$	$m$

but present

GATE-94

Given a Relational Algebra Expression using only the minimum no of tuples matching with EID

or matching

$$\begin{aligned} \& (R \cap S) = R - (R - S) \\ \& [(R \cup S) - (R - S)] - (S - R) = (R \cap S) \end{aligned}$$

particular

GATE-99

Consider the join of a relation R with relation S - If 'R' has m tuples

and 'S' has 'n' tuples, then the maximum and minimum sizes of the

join respectively are

- 1)  $m \times n$  and  $m + n$
- 2)  $m \times n$  and  $m + n$
- 3)  $m \times n$  and  $m + n$
- 4)  $m \times n$  and  $m + n$

gate -99

The Relational Algebra expression equivalent to the following tuple calculus expression  $\{t / \text{term}(t[\alpha_1] = 10 \wedge t[\beta_1] = 20)\}$  is:

a)  $\sigma_{\{\alpha_1=10 \wedge \beta_1=20\}}(r)$  OR but we need AND

b)  $\sigma_{\{\alpha_1=10\}}(r) \cup \sigma_{\{\beta_1=20\}}(r)$

c)  $\sigma_{\{\alpha_1=10\}}(r) \cap \sigma_{\{\beta_1=20\}}(r)$

d)  $\sigma_{\{\alpha_1=10\}}(r) - \sigma_{\{\beta_1=20\}}(r)$

or

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

Given the Relations

Employee (Name, salary, deptno) and

Department (deptno, dname, address)

which of the following queries cannot be expressed using the basic Relational Algebra operations  $(\pi, \sigma, \times, \bowtie, \cup, \cap, -)$ ?

IS-GATE 9

Let  $R_1(ABC)$

are shown

Suppose the

In the Co

following

and emp

or  $\pi_D(x_2) -$

say

#### 14. GATE 2003 QUESTION ON CONVERTING SQL TO RA

Consider the following SQL query to calculate sum of all employees' salaries.

Select distinct  $\alpha_1, \alpha_2, \dots, \alpha_n$

From  $r_1, r_2, \dots, r_m$

where P

For an arbitrary predicate 'P' this query is equivalent to

which of the following Relational Algebra expressions

⑥

$$\text{proj}_{a_1, a_2, \dots, a_n} \pi_p(r_1 \times r_2 \times \dots \times r_m)$$

$$b) \overline{\prod}_{a_1, a_2, \dots, a_n} \pi_p(r_1 \Delta r_2 \Delta \dots \Delta r_n)$$

Select distinct = projection

where = Selection.

$$\begin{aligned} ① \quad & [R_1] \times [R_2] \times [R_3] \xrightarrow{\text{From clause}} \prod_{a_1, a_2, \dots, a_n} \pi_p(r_1 \times r_2 \times \dots \times r_n) \\ & \Downarrow \text{F} = \text{where} \\ & \boxed{\begin{array}{|c|c|c|} \hline R_1 & \times & R_2 \times R_3 \\ \hline \end{array}} \quad R_1 \times R_2 \times R_3 \\ & \Downarrow \pi = \text{Selected distinct} \end{aligned}$$

### 15. GATE QUESTIONS ON FOREIGN KEY AND RELATIONAL ALGEBRA

Let  $R_1(ABC)$  and  $R_2(CDE)$  be two Relation schema, where the primary keys are shown underlined, and let  $C$  be a foreign key in  $R_2$  referring to  $R_1$ .

Suppose there is no violation of the above referential integrity constraint. In the corresponding relation instances  $r_1$  and  $r_2$ , which are of the following Relational Algebra expressions would necessarily produce

and empty relation?

corresponding  
an dept number  
on dept number  
 $\oplus \pi_D(r_1) - \pi_C(r_1) - \pi_D(r_2) \quad \oplus \pi_D(r_1) \Delta r_2 \quad \oplus \pi_C(r_1) \Delta r_2$

$\Rightarrow$   $R_1(\underline{A}\underline{B}\underline{C}) \quad R_2(\underline{C}\underline{D}\underline{E})$   $\xrightarrow{\text{FK}}$  The values present in the 'C' will definitely be present in 'D'.

$$\begin{array}{|c|c|c|} \hline A & B & C \\ \hline 1 & 2 & 1 \\ \hline 2 & 1 & 1 \\ \hline \end{array} \quad \begin{array}{|c|c|c|} \hline D & E \\ \hline 1 & 2 \\ \hline 2 & 3 \\ \hline \end{array}$$

$\{1, 2\} - \{1, 2, 3, 4\} = \emptyset.$

$\therefore \pi_C(r_1) - \pi_D(r_2) = \emptyset$

### 16. GATE 2015 QUESTION ON DECOMPOSITION AND JOIN

Let  $\gamma'$  be a relation instance with schema  $R(ABCD)$ . we define  $S = \pi_{ABC}(\gamma')$  and  $\gamma_2 = \pi_{A,D}(\gamma')$ . Let  $S = \gamma_1 * \gamma_2$  where  $*$  denotes natural join. Given that the decomposition of  $\gamma'$  into  $\gamma_1$  and  $\gamma_2$  is lossy, which one of the following is True?

- a)  $S \subset \gamma'$       b)  $\gamma_1 \cup S = \gamma'$       c)  $\gamma_1 \cap S = S$       d)  $\gamma_1 * S = S$ .

Given the decomposition of  $\gamma'$  into  $\gamma_1$  and  $\gamma_2$  is lossy which means we get spurious tuples when we perform natural join them.  $\therefore S$  will have additional tuples (also called spurious tuples) that are not present in  $\gamma'$ .

$$\therefore \boxed{\gamma_1 \cap S = S}$$
 or  $\boxed{S \supset \gamma'}$

### 18. GATE 2015 Question

Consider the Relation  $S = \pi_{name}(\sigma_{Sex = 'M'}(G))$ . Which of the following is true?

- a) Names of girls  
b) Names of girls &  
c) Names of girls &  
d) Names of girls

$$\text{Sol} \quad \text{Let } A = \pi_{name}(G)$$

### 17. GATE 2014 QUESTION ON CASCADE PROJECTION AND JOIN

What is the optimised version of the Relation algebra expression  $\pi_{A_1}(\pi_{A_2}(\sigma_{F_1 \wedge F_2}(\gamma)))$ , where  $A_1$  and  $A_2$  are sets of attributes in  $\gamma'$  with  $A_1 \subset A_2$  and  $F_1, F_2$  are Boolean expressions based on the attribute in  $R$ ?

$$\textcircled{Q} \pi_{A_1}(\sigma_{F_1 \wedge F_2}(\gamma)) \quad \textcircled{R} \pi_{A_1}(\sigma_{F_1 \vee F_2}(\gamma)) \quad \textcircled{S} \pi_{A_2}(\sigma_{F_1 \wedge F_2}(\gamma))$$

$\Rightarrow$  Selection operation is commutative and we can cascade it.

$$\therefore \pi_{A_1}(\sigma_{F_1}(\sigma_{F_2}(\gamma))) \equiv \sigma_{F_1 \wedge F_2}(\gamma)$$

$$\Rightarrow \text{Now projection operation} \Rightarrow \boxed{\pi_A(\pi_B(\gamma)) : \downarrow A \subset B \text{ then we write } \pi_A(\gamma)}$$

Here Given  $A_1 \subset A_2$   $\pi_{A_1}(\pi_{A_2}(\sigma_{F_1 \wedge F_2}(\gamma)))$

$$= \pi_{A_1}(\pi_{F_1 \wedge F_2}(\gamma))$$

\* option a.

21

(82)

i. we define  
dies natural  
is lossy which

which means  
join them  
of tuples) that

of tuples) that

### 18. GATE 04 QUESTION ON INTERPRETING AN RA EXPRESSION

Consider the Relation Student (name, Sex, marks) where the primary key is shown underlined, pertaining to students in a class that has at least one boy and one girl, what does the following relational algebra expression produce? (Note:  $\sigma$  is the Rename operator).

$$\Pi_{\text{name}}(\sigma_{\text{Sex}=\text{female}}(\text{Student})) - \Pi_{\text{name}}(\text{Student} \Delta \rho(\text{Student}))$$

$\Delta$ : Names of girls students with highest marks  
 $\rho$ : Names of girls students with more marks than some boy student.

- a) Names of girl students with highest marks
- b) Names of girl students with more marks than some boy student.
- c) Names of girl students with marks not less than some boy student
- d) Names of girl students with more marks than all the boy students

So let  $A = \Pi_{\text{name}}(\sigma_{\text{Sex}=\text{female}}(\text{Student}))$  = {get the names of all girls (sex=female)}

### AND JOIN

expression  
of attributes  
based on the  
constraints

$$\pi_{A_1}(\sigma_{F_1 \vee F_2}(s))$$

Now  $A - B =$

$\{ \text{Names of all girls} \} - \{ \text{Names of girls whose marks are less than all boys} \}$

add it

$\Rightarrow \{ \text{Names of all girls whose marks are more than all the boy students} \}$

Then we  
 $\pi_A(r)$

\* OPTION D



### SESSION 6:

Q. 20. GATE Q 11 on optimisation of RA Expression

relation

1. D gives

b-schema =  $(b\_name, b\_city, assets)$

e that student.

a-schema =  $(a\_num, b\_name, bal)$

d-schema =  $(c\_name, a\_number)$

Let Branch, account and depositor be prospective instances of above schema. Assume that account and depositor relations are much bigger than the branch relation.

consider the following query  $\Pi_{c\_name, b\_city} (\sigma_{b\_city = "Agra"} \Delta (branch \bowtie (account \bowtie depositor)))$

involved

which one of the following queries is the most efficient version of above query?

$\sigma \Pi_{c\_name} (\sigma_{b\_city < 0} (\sigma_{b\_city = "Agra"} (branch \bowtie account) \bowtie depositor))$

$\sigma \Pi_{c\_name} (\sigma_{b\_city = "Agra"} (branch) \bowtie (\sigma_{bal > 0} (account \bowtie depositor)))$

$\sigma \Pi_{c\_name} ((\sigma_{b\_city = "Agra"} (branch) \bowtie (\sigma_{bal > 0} (account \bowtie depositor))) \bowtie (\sigma_{b\_city < 0} (branch \bowtie account) \bowtie depositor))$

$\sigma \Pi_{c\_name} ((\sigma_{b\_city = "Agra"} (branch) \bowtie (\sigma_{bal > 0} (account \bowtie depositor))) \bowtie (\sigma_{b\_city < 0} (branch \bowtie account) \bowtie (\sigma_{bal > 0} (depositor))))$

$\sigma \Pi_{c\_name} ((\sigma_{b\_city = "Agra"} (branch) \bowtie (\sigma_{bal > 0} (account \bowtie depositor))) \bowtie (\sigma_{b\_city < 0} (branch \bowtie account) \bowtie (\sigma_{bal > 0} (depositor))) \bowtie (\sigma_{b\_city < 0} (branch \bowtie account) \bowtie (\sigma_{bal > 0} (depositor))))$

$\Rightarrow$  If you perform cross product first, and then the selection operation it wont be efficient because cross product generates large more root tuples.

$\Rightarrow$  So first apply the selection condition and then apply the cross product from the obtained two smaller tables.

1st select Agra city from b-schema

2nd select the tuples with bal > 0 from a-schema

Now cross product the account with depositor

Now, cross product the result with b-schema

$\Pi_{c\_name} ((\sigma_{b\_city = "Agra"} (branch) \bowtie (account \bowtie depositor)))$

= option B.

{Subset  
of female  
make  
registered  
Registers}

## QUESTION ON NATURAL JOIN

Let  $R$  and  $S$  be two Relations with the following schema

$R(PQ, RP_2, R_3)$  and  $S(PQ, S_1, S_2)$  where  $\{PQ\}$  is the key for both schemas. Which of the following queries are equivalent?  $\{PQ\}$  is the key for both schemas.

1.  $\nabla_p(R \otimes S)$     2.  $\nabla_p^{(R)} \bowtie \nabla_p^{(S)}$   
   3.  $\nabla_p(\nabla_{p_A(R)} \nabla_{p_B(S)})$     not  $p_A$  alone  
   4.  $\nabla_p(\nabla_{p_A(R)} - \nabla_{p_B(S)})$

(a) only I and II      (b) only I and III      (c) only I, II and III      (d) only I, III, and IV

R $\otimes S$		P	Q	R
		1	2	3
P	1	2	3	Q
P	1	2	3	Q
Q	1	2	3	R

$$\begin{array}{l} \text{R M S} \\ \boxed{\begin{array}{c} P \\ - \\ 1 \\ - \\ 2 \\ - \\ 3 \end{array}} \\ (1) \\ \boxed{\begin{array}{c} P \\ - \\ 1 \\ - \\ 2 \\ - \\ 3 \end{array}} \\ S \\ \boxed{\begin{array}{c} P \\ - \\ 1 \\ - \\ 2 \\ - \\ 3 \end{array}} \\ X \\ \boxed{\begin{array}{c} P \\ - \\ 1 \\ - \\ 2 \\ - \\ 3 \end{array}} \\ \text{I. R.} \end{array}$$

$$\text{Ans} \quad \begin{array}{|c|c|c|} \hline P & Q & R \\ \hline 1 & 0 & 1 \\ \hline 2 & 1 & 0 \\ \hline 3 & 0 & 1 \\ \hline \end{array} \quad \Rightarrow \quad \begin{array}{|c|c|c|} \hline P & Q & R \\ \hline 1 & 0 & 1 \\ \hline 2 & 0 & 0 \\ \hline 3 & 0 & 0 \\ \hline \end{array}$$

now we know that  $A \cap B = A - (A - B)$   
 Let  $A = \pi_{P/Q}(R)$        $B = \pi_{P/Q}(S)$

option I (III) are equivalent.

How many types  
expression can  
as that of A.

2) 7  
6) 4

**2. GATE 2010 QUESTION ON NATURAL JOIN ON PRIMARY KEY**

(Q)

for both

Q is the key  
of P & also  
of R

The following FDs hold for Relations R(A,B,C) S(C,D,E) {B  $\rightarrow$  A, A  $\rightarrow$  C}

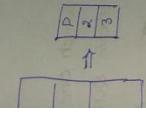
The relation 'R' contains 200 tuples and 'S' contains 100 tuples, what  
is the max no. of tuples possible in natural join of R,S i.e. R  $\bowtie$  S?

- a) 100    b) 200    d) 300    e) 2000

Now,  $B \rightarrow A$ ,  $A \rightarrow C \Rightarrow B^+ = \{B, A, C\} \Rightarrow B$  is CK in  $R(ABC) \Rightarrow B$   
contains distinct values in  $R(ABC)$

R			S		
A	B	C	B	D	E
1	1		1		
2	2		2		
3	3		3		
				100	
					100

R			S		
A	B	C	B	D	E
1	1		1		
2	2		2		
3	3		3		
				100	
					100



**3. GATE 12 QUESTIONS ON UNION AND NATURAL JOIN**

consider the following relations A,B,C

A			B			C		
ID	Name	Age	ID	Name	Age	ID	Name	Age
12	Arun	60	15	Shreya	24	10	22	50
15	Shreya	24	25	Homi	40	20	20	52
99	Rohit	11	98	Rohit	20	99	Rohit	11

How many tuples does the result of the following relational Algebra  
Expression contain? Assume that the schema of A,B,C is the same  
as that of A.  $(A \cup B) \bowtie C$

a) 4    b) 5    c) 6    d) 9.

~~24~~ b) 4    c) 5    d) 9.

ID	Name	Age
12	Arun	60
15	Shreya	24
99	Rohit	11
25	Hari	40
98	Rohit	20

C

ID	Phone	Area
10	2200	02
99	2100	01

(AUB)  $\bowtie$  C

ID	Name	Age	ID	Phone	Area
12	Arun	60	10	2200	02
12	Arun	60	99	2100	01
15	Shreya	24	10	2200	02
15	Shreya	24	99	2100	01
99	Rohit	11	10	2200	02
99	Rohit	11	99	2100	01
25	Hari	40	10	2200	02
25	Hari	40	99	2100	01
98	Rohit	20	10	2200	02
98	Rohit	20	99	2100	01

Now, pick the tuples where  $A.id > 40 \vee C.id < 15$

ID	Name	Age	ID	Phone	Area
12	Arun	60	10	2200	02
15	Shreya	24	10	2200	02
99	Rohit	11	10	2200	02
25	Hari	40	10	2200	02
98	Rohit	20	10	2200	02
98	Rohit	20	99	2100	01
99	Rohit	11	99	2100	01

$C.id < 15 = 5$  tuples

$A.id > 40 = 2$  tuples

= 7 tuples

Refer 547 page  
3.25(a) in  
madeeasy  
book

The nested loop  
we are going to

$r(R)$

↓  
20 tuples

↓  
2 blocks.

In this case the  
No. of Block access  
are

$1 + 10 \times 10$  - for  
 $1 + 10 \times 10$  - for  
202

..  $\square$

#### 24. GATE 14 QUESTION ON NESTED EVALUATION OF JOIN

Consider a join between relations  $r(R)$  and  $s(S)$  using the nested loop method. There are 3 buffers each of size equal to disk block size, out of which one buffer is reserved for intermediate results. Assuming size ( $r(R) < \text{size } s(S)$ ), the join will have fewer no of disk blocks access if

- a) relation  $r(R)$  is in the outerloop
- b) relation  $s(S)$  is in the outerloop
- c) join selection factor between  $r(R)$  and  $s(S)$  is more than 0.5
- d) join selection factor between  $r(R)$  and  $s(S)$  is less than 0.5.

#### 25. GATE 201

for the  
in the dependent  
dependent (d)  
deletional AF

ID	phone	Area
10	2100	02
90	2100	01
10	2200	02
99	2100	01
10	2200	02
99	2100	01
10	2200	02
99	2100	01
10	2200	02
99	2100	01

The nested loop method means for every tuple in one relation we are going to take <sup>other</sup> entire table and we are joining them. (8)

$r(R)$

↓  
20 tuples

↓  
2 Blocks.

C point

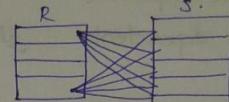
$s(S)$

↓  
100 tuples

↓  
10 Blocks

Assuming,

{ size of each block = 10 tuples  
and  
nested looping means }



In this case the  
No. of Block access  
are

$1 + 2 * 10$  — for 1st blk { Take every row of R and  
 $1 + 2 * 10$  — for 2nd blk [ combine with 'S' ] }

$\frac{1 + 10 \times 10}{10}$  — for 1st blk  
 $\frac{1 + 2 * 10}{10}$  — for 2nd blk  
10 times → for 10th blk

10 Block Access

$1 + 10 \times 10$  — for 1st block

$1 + 10 \times 10$  — for 2nd block

202

∴ put the smaller size table in outer loop.

Refer 547 page  
3.25(a) in  
made easy  
book

→ How did we write this is for the 1st block, we have to access

it so (1+) and for every block we have to link with all the

10 blocks in 'S' (so  $1 + 10 \times 10$ )

↓

1st block Accessing all the blocks in

'S' = 10 blocks and size of each block = 10 tuples

$= 10 \times 10 = 100$  tuples.

nested Loop  
block size,  
ulte. Assuming

sk blocks

### 5. GATE 2014 QUESTION ON INTERPRETATION OF RA EXPRESSION

Consider the Relational schema given below, where "eID" of the relation dependent is a foreign key referring to "empID" of the relation "employee".

Assume that every employee has atleast one associated dependent in the dependent relation. Employee (empID, empName, empAge)  
dependent (depID, eID, depName, depAge) consider the following  
Relational Algebra Query.

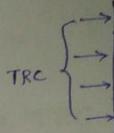
$$\Pi_{\text{empid}} (\text{employee}) - \Pi_{\text{empid}} (\text{employee} \bowtie (\text{dependent}))$$

$\downarrow$

$$\langle \text{empid} = \text{eId} \rangle \wedge (\text{empAge} \leq \text{depAge})$$

Q. A language

express



The above query evaluates to the set of empId's of employees whose age is greater than that of

- a) some dependent      c) some his/her dependents
- b) All dependents.      d) All of his/her dependents.

Employee

ID	Age
1	40
2	30
3	20

Dependent

did	Age
1	30
1	20
2	40
3	40

27. TRC

The most

{t/c}

↓  
Tuple variable

Ex:-

student

FN

Now,  $\Pi_{\text{empid}} (\text{employee} \bowtie (\text{dependent}))$   $\Rightarrow$  It returns the empids of employees whose Age is less than or equal to their dependents.

Now, from employee table, subtract the above relation then we will get

$\Rightarrow$  Empid's of employees, whose age is greater than his/her dependents.

∴ Option D.

## 26. INTRODUCTION TO RELATIONAL CALCULUS

On the Relational model two formal languages are proposed they are

1) Relational Algebra

2) Relational calculus

Tuple Relation Calculus (TRC)

Domain Relation Calculus (DRC)

The general

{t; A<sub>j</sub>}

28. FREE

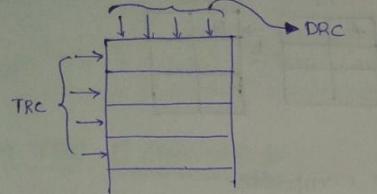
General  
in the

'1' R(t)

90

A language is said to be Relationally complete if it has the capacity to express every query of Relational Algebra.

Bye's



91

### 27. TRC SYNTAX

The most general form of TRC is

$$\{t / \text{cond}(t)\}$$

↓      ↓  
Tuple      condition  
variable

Ex:-

Student

FN	LN	Marks

Now if I want to find the student names whose marks are greater than 50.

$$\{t / \text{student}(t) \text{ AND } t.\text{marks} > 50\}$$

↓  
Tuple variable to range on table  
Name of the table

$$\{t.FN / \text{student}(t) \text{ AND } t.\text{marks} > 50\}$$

that the tuple variable should give FN of students whose marks > 50.

The general form is

$$\{t_1.A_1, t_2.A_2, \dots, t_n.A_n / \text{cond}(t_1, \dots, t_n, \dots, t_{n+m})\}$$

### 28. FREE AND BOUNDED VARIABLES

Generally we use atomic expression while representing conditions

in the TRC. The basic Atomic expressions will be

(1)  $R(t)$      $\rightarrow$

(2)  $t_i.A \theta \text{constant}$   
 $t_i.\text{marks} > 50$

(3)  $t_1.A < t_2.B$



Now,  $\neg \exists t \{t < 0\} \rightarrow$  There does not exist atleast one value of  $t'$  such that  
 $t' < 0 \therefore \text{TRUE}$

Now,  $\forall t \{t > 10\} = \text{TRUE}$  (All values are greater than 10).

### 29. TRC Example 4

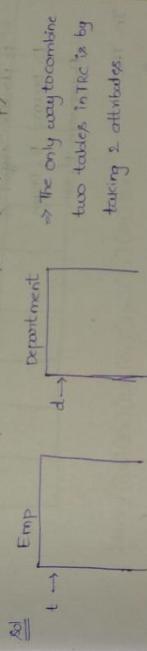
List the names and Address of all employees who work for the "Research" department. The database schema is:

Employee (Name, Minit, Name, SSN, Date, Address, Sex, Salary, Super-SSN,  
 Department (Name, Dnumber, Manager, Mgr-Ssn, Proj)  
 Dept-Locations (Dnumber, Location)

Project (Pname, Pnumber, Place)

Works-on (Esn, Proj, Hours)

Dependent (Esn, Dependent-name, Sex, Birthdate, Relationship)



$\Rightarrow$  Now, after choosing the variables we are going to make one of the variables free. In general we free the variable whose values should be printed. Now, [Name, Address] are the values that should be printed and they are present in Emp-table  $\Rightarrow$  free the variable pointing to Employee table  $\Rightarrow$  free the variable  $t$ .  
 $\Rightarrow$  The cross product in Relational Algebra is equal to  $\exists$  in the values of TRC

$\{t \cdot \text{Name}, t \cdot \text{Address} / \text{Employee}(t) \text{ AND }$   
 $(\exists d) (\text{Department}(d) \text{ AND } d.\text{Name} = \text{"Research"})$   
 $\text{AND } d.\text{Dnumber} = t.\text{DNO}\}$

### 30. TRC Example 2

For every project located in "Shafford" list the project number, the controlling department number, and the department manager's lastname, birthdate, and Address. The database schema is same as in the previous problem.

First of all we want to find the project location from "Project table".

Project	location	pro	Deptnum
P	→	d	m

In Relational Algebra

$$\delta_{(P \times D \times E)}$$

P.location = "Shafford"  $\wedge$

P.deptnum = D.deptnum

D.manager = E.lastname

In TRC

$\{P.projnum, P.deptnum, m.lastname, m.birthdate, m.Address / PROJECT(p) \text{ AND }$

$SUPERVISOR(Em) \text{ AND } P.location = "Shafford" \text{ AND }$

$(\exists d)(DEPARTMENT(d) \text{ AND } P.deptnum = d.deptnum \text{ AND } d.manager = m.lastname)$

$\exists d (d.deptnum = P.deptnum \text{ AND } d.manager = m.lastname)$

### 38. DOMAIN RELATIONAL CALCULUS INTRODUCTION

The main difference between the TRC and DRC is the way in which we use the variables

$\Rightarrow$  In case of domain relational calculus we need more variables than the TRC

### 39. DRC - E

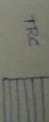
List the

"JOHN B.

Now, the  
present in

Employee

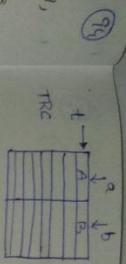
present in



$\Rightarrow SQL =$  to

$\Rightarrow Item = A$

③



R
$a = 10 \text{ in DRC}$

Ques

⇒ SQL = based on (TRC and RA)

⇒ IBM = QBE (Query By Example) - based in DRC.

able!

List the Birth date and the address of the employee whose name is

'JOHN B SMITH'

Now, the details like Birth date, Address of John-B-Smith are present in employee table.

Employee	v	r	s	t	u	v	w	x	y	z
FN	MN	LN	SSN	Bdate	Adr	Sex	sal	Supervisor	Dno	

{ $v, r / (v, r, s, t, u, v, w, x, y, z)$  }

Ques

(EMPLOYEE( $\gamma_{\text{name}} = q_1$ ) AND

$q_1 = 'JOHN'$  AND  $r = 'B'$  AND  $s = 'SMITH'$ )}

TRC- DRC- Not much needed for GATE only
Simple Questions will be asked

10

## QUESTION

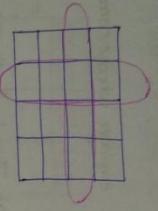
R → open/close connection  
frame

### 1. INTRODUCTION TO SQL

SQL is based on Relation Algebra, TRC

SEQUEL = prior version = Sequential English Query language

Table - Row - column



### 2. CREATING A TABLE AND CONSTRAINTS ON IT

CREATE SCHEMA COMPANY AUTHORIZATION;

Employee

#### FOREIGN

→ NO, INSTEAD  
they are

⇒ The way  
of SET NULL

when we try

Employee

};

primary key allows NOTNULL by default

Foreign key allows NULL by default.

The various datatypes that are used are NUMERIC, CHAR(m), VARCHAR

$\overbrace{\text{INT} \quad \text{SMALLINT} \quad \text{TINYINT} \quad \text{REAL}}$

⇒ If i delete

⇒ It is

⇒ It SET

⇒ spaced

⇒ If then

⇒ If Related

### 3. REFERENCES

⇒ whenever  
the default

OPEN

CLOSE

(8)

write is

see in result

## 5. ALIASING

for each employee, retrieve the employee's first and lastname and the first and lastname of his immediate supervisor.

(9)

Employee (Fname, Minit, Lname, Ssn, Bdate, Address, sex, salary, super-ssn,

Employee

Dno)

Department (Dname, Dnumber, Mgr-ssn, Mgr-start date)

Supervisor

Dept-locations (Dnumber, Dlocation)

Project (Pname, Pnumber, Plocation)

works-on (Esn, Pno, Hours)

Dependent (Esn, Dependent-name, sex, Bdate, Relationship).

Aliasing Employee as E

E.Dno

(Employee)  
(ssn=10)

Select:

Employee

ESSN	FN	LN	SSN
1	A	B	10
X			
SSSN	FN	LN	ESSN
10	C	D	

(we are doing  
cross product  
on the same table  
Rename it as S)

work for

SELECT E.FN, E.LN, S.Fname, S.Lname

FROM Employee AS E, Employee AS S

WHERE E.ESSN = S.SSN;

## 6. DUPLICATE TUPLES AND SET OPERATIONS

→ we can use the keyword DISTINCT to eliminate the duplicates.

SELECT DISTINCT Fname  
From Employee  
WHERE Dno='4';

⇒ All the Fnames of employee  
who work for dept no '4' (No  
duplications)

<p>Now,</p> <p><u>(SELECT DISTINCT From Employee WHERE DNO='4') UNION (SELECT DISTINCT From Employee WHERE DNO='5').</u></p> <p>At End where DNO=4/5 No Repetitions.</p> <p>UNION = Eliminates duplicates + SELECT DISTINCT</p> <p>UNION ALL = Does not Eliminates duplicates + SELECT ALL</p>	<p>Now if :</p> <p><math>\Rightarrow R \cup S</math></p> <p><math>R \cup S = \{ \text{SELECT * FROM Employee} \}</math></p> <p><math>R \cup S = \{ \text{SELECT * FROM Employee} \}</math></p> <p><math>R \cup S = \{ \text{SELECT * FROM Employee} \}</math></p>																	
<p><u>R              S</u></p> <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>a1</td></tr> <tr><td>a2</td></tr> <tr><td>a3</td></tr> <tr><td>a4</td></tr> <tr><td>a5</td></tr> </table> <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>a1</td></tr> <tr><td>a2</td></tr> <tr><td>a3</td></tr> <tr><td>a1</td></tr> <tr><td>a3</td></tr> <tr><td>a5</td></tr> </table> <p><u>R UNION S</u></p> <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>a1</td></tr> <tr><td>a2</td></tr> <tr><td>a3</td></tr> <tr><td>a1</td></tr> <tr><td>a3</td></tr> <tr><td>a5</td></tr> </table>	a1	a2	a3	a4	a5	a1	a2	a3	a1	a3	a5	a1	a2	a3	a1	a3	a5	<p><math>\Rightarrow \text{Now if :}</math></p> <p><math>\left\{ \begin{array}{l} \text{SELECT *} \\ \text{FROM} \\ \text{Employee} \end{array} \right\}</math></p> <p><math>\Rightarrow \text{Now if :}</math></p> <p><math>\left\{ \begin{array}{l} \text{SELECT *} \\ \text{FROM} \\ \text{Employee} \end{array} \right\}</math></p>
a1																		
a2																		
a3																		
a4																		
a5																		
a1																		
a2																		
a3																		
a1																		
a3																		
a5																		
a1																		
a2																		
a3																		
a1																		
a3																		
a5																		

### 7. PATTERN MATCHING AND LIKE OPERATORS

The 'LIKE' is used for string comparing / comparing

Now, if I want to find all the employees whose names contains "Ravi"

$\Rightarrow$  The 2 operators used are

% = no of 'o' or more character in between  
'\_,' = Single character

SELECT Frame ZEROFILL Test, Data ZEROFILL  
From Employee  
WHERE Frame LIKE '%Ravi%'

$\Rightarrow$  Before Ravi there can be any no of characters  
 $\Rightarrow$  After Ravi there can be any no of characters

$\left\{ \begin{array}{l} A \\ B \\ A \\ C \\ B \end{array} \right\}$

$\Rightarrow$  ORDER BY > 0  
Order By clause  
Select Query

⑩ ⇒ Now if I want employees whose third letter in frame is 'v' then  
 → Frame LIKE  $\underline{\underline{v}}$  After that there can be anything  
 ↓  
 1st character  
 2nd character  
 3rd character

⇒ Frame || Name ⇒ Outputs Concatenated Names.

⇒ Now if I want to increase the salary of all the employees by 10% then

```
SELECT Frame, 1.1 * SALARY
FROM EMPLOYEE;
BETWEEN ⇩
SELECT *
FROM Employee
WHERE SALARY BETWEEN 10000 AND 20000;
OR 10,000 ≤ SALARY ≤ 20,000.
```

### 3 ORDER BY

Retrieve a list of employees and the projects they are working on, ordered by department and within each department ordered alphabetically by last name, then first name.

⇒ Suppose if I have a Relation like

$\begin{cases} A & b \\ B & a \\ A & c \\ B & b \\ C & c \\ B & b \end{cases}$	First order by dept name, if they are same then order by last name and then by first name.
--	--

Output should be

$\begin{cases} A & b \\ B & a \\ A & c \\ B & b \\ C & c \\ B & b \end{cases}$	$\begin{cases} A & b \\ B & a \\ A & c \\ B & b \\ C & c \\ B & b \end{cases}$
--	--

\* ORDER BY → used to order the output  
 → "OrderBy" clause can be applied on the attributes appearing in select query.  
 can be done



10. INSERT

The command used to insert a tuple in the relation is INSERT

```
INSERT INTO Employee VALUES ('Ravi', 'Ravah', 1234, 4);
```

```
→ INSERT INTO EMPLOYEE ('Name', 'IN_ID', 'SSN', 'DNO) VALUES
```

('Ravi', 'Ravada', '1234', '4'));

⇒ Now, if I want to insert the values in a table in which one the

R  
  
S  


BY ABC

Now, if I want to delete all the employees whose last name is "Ravi" then,

```
→ DELETE FROM Employee WHERE LN = 'Kavir'
```

SETI ॥ ३

330  
updates the Salaries of all Employees who possess  
more than

### DEFINING HUMAN VALUES

## 12 DEALING WITH NEGATIVE VALUES

→ The problem with the NULL values is we don't know how to interpret them.

it when we are comparing especially  
null values can be TRUE / FALSE

Now if i have to do (x^n)

→ In SQL every NULL is considered to be distinct, so just to deal with them two new operators "IS" AND "ISNOT" are introduced

### 13. IN

I want to find out all the Frame, Addresses of employees who work for dept = {2,3,4,5}

```
SELECT frame, address  
FROM employee  
WHERE Dno IN (1,2,3,4);  
(Or)
```

Dno=1 OR Dno=2 OR Dno=3 OR Dno=4

Find Frame, Address of employees who work for department location in "Stafford"

```
SELECT frame, address  
FROM employee  
WHERE Dno IN (SELECT Dno  
FROM DEPT_LOCATIONS  
WHERE location='Stafford');
```

Frame	Ad
A	1
B	2
C	2
D	3
E	3
F	3

location="Stafford";

### 14. ANY, ALL, SOME

```
SELECT DISTINCT E.ssn  
FROM WORKS_ON  
WHERE (Pno, hours) IN (SELECT Pno, hours  
FROM WORKS_ON  
WHERE E.ssn='10');  
  
→ Both and query.
```

E	
1	
2	
3	
10	

(10)

ON S  
ON

Find Frame

of all emplo

```
SELECT F  
FROM E  
WHERE
```

RETRIEVE

### 15. NESTED

RETRIEVE  
the same

SELECTED

From Employee

WHERE

Both and query.

introduced

(Ex)

E	P	H
1	10	10
2	20	10
1	30	10
10	1	10
10	20	10
10	30	10

⇒ The given query will give,  
(Ex)  
1. The outer query produces  
2. The inner query finds all the employees  
who have worked on some project  
on which employee no. 10 has worked for  
the same no. of hours.

Find frame of all employee whose salary is greater than the salary  
of all employee in department no. 5

```
SELECT F.Name
FROM Employee E
WHERE SALARY > ALL (SELECT SALARY
                     FROM Employee
                     WHERE DNO = 5);
```

### 15. NESTED CORRELATED EXPRESSIONS

RETRIEVE the frame of each employee who has a dependent with  
the same frame and is the same sex as the employee

```
SELECTED (E.Frame),
       FROM Employee AS E
       WHERE E.SSN IN (SELECT D Dependent AS D
                       WHERE (E.Frame) = D.Dependent_name
                         AND E.Sex = D.Sex);
```

If the same attribute is present  
on both outer and inner loops then that query is called co-related  
query.

17. Exist

List the products produced by employee whose SSN is 501.

```

SELECT
  FROM
    WHERE
      AND
      EXISTS
    SELECT
      FROM
        UNION
        AND E
      Emp
      FN
  
```

Employee

Frame	SSN	Sex
A	501	F
B	502	M
C	503	M

Dependent

Dependent	Name	Sex	SSN
E200	A	F	501
S01	B	M	502
502	F	M	501

OuterQuery produces

	A	B
Employee	1	0
Dependent	0	1

18. Exists AND NOT EXISTS

Select employees who have no dependents.

```

SELECT Frame
  FROM EMPLOYEE AS E
 WHERE EXISTS (SELECT *
    FROM DEPENDENT AS D
   WHERE E.SSN = D.SSN AND E.Sex = D.Sex AND
     D.Frame = D.Department_name)
  
```

⇒ If the Internal query returns some value then exists will return TRUE and if the internal query doesn't return anything the exists will return FALSE.

Retrieve the frames of employees who have no dependents.

```

SELECT Frame
  FROM EMPLOYEE
 WHERE NOT EXISTS (SELECT * FROM DEPENDENT WHERE SSN = E.SSN);
  
```

→ If this query returns something then exists will return TRUE.

→ If this query returns nothing then exists will return FALSE.

Employee

Frame	SSN	Sex
1	501	F
2	502	M
3	503	M
4	504	F

Dependent

Dependent	Name	Sex	SSN
E200	A	F	501
S01	B	M	502
502	F	M	501

### 11. EXISTS Example 1

24

2

**Q1** List the frames of managers who have atleast one dependent

```
FROM Employee  
WHERE EXISTS (SELECT * FROM department WHERE state = 'CA')  
AND  
EXISTS (SELECT * FROM department WHERE state = 'NY' OR state = 'IL');
```

SINGAPORE

```
FROM EMPLOYEE  
WHERE EXISTS (SELECT * FROM DEPARTMENT WHERE ESSN = D.MGR-ESSN)  
AND EXISTS (SELECT * FROM DEPARTMENT WHERE ESSN = MGR-ESSN)
```

$$\begin{array}{l} \text{Dependent} \\ \text{for } C = 1 - \text{ton} \\ \text{for } F^o = 1 - \text{ton} \\ f_{C\bar{O}} = \text{ton} \end{array}$$

18. EXISTS EXAMPLE 2 -  
Retrieve the name of each employee who works on all the projects  
controlled by department number 5.

From Employee  
 WHERE NOT EXISTS (SELECT Number  
 FROM Project  
 WHERE Drum=5) EXCEPT (SELECT Proj FROM  
 Works-on WHERE Sn = Essn);  
 RETURNS TRUE

Project		From	To	PNO	ESD
Emp	Task	S/N	F/N	Duration	1
	Revise	1		5	10
	Revise	2		5	1
	Revise	3		5	2
	Revise	4		1	3
	Revise	5		2	1
	Revise	6		2	2
	Revise	7		2	2
	Revise	8		2	2

Ravi will be print

## 19. JOINS

SELECT Name, LName, Address FROM (EMPLOYEE JOIN DEPARTMENT  
ON Dno= Dnumber)

WHERE Dname = 'Research';

SELECT Name, Lname, Address FROM (Employee NATURAL JOIN  
AS DEPT (Dname, Dno, Mgr, Msdate))  
WHERE Dname = 'Research';

SELECT E.Lname AS Employee\_name, S.LName AS Supervisor\_name

FROM (Employee AS E LEFT OUTER JOIN  
Employee AS ON E.Super\_SSN = S.SSN);

P  $\rightarrow$  A, for natural join

R  $\times$  S (Join And C)  
A  $\times$  C

	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	

$\Rightarrow$  Select S

(R  $\times$  S) (Right Outer Join)

A  $=$  C

	A	B	C	D
1	a	1	g	
2	b	2	h	
3	c	3	i	
4	d	N	N	
5	e	N	N	
6	f	N	N	

$\Rightarrow$  Select S

## 20. Aggregate

The Aggregate

Employee

Eno	Emp
1	A
2	B
3	C
4	D
5	E
6	F

$\Rightarrow$  while C

3) SELECT

R  $\times$  S (Natural Join)

A  $\times$  C

	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	

$\Rightarrow$  Select S

(R  $\times$  S) (Left Outer Join)

S  $=$  C

	A	B	C	D
1	a	1	g	
2	b	2	h	
3	c	3	i	
4	d	N	N	
5	e	N	N	
6	f	N	N	

$\Rightarrow$  Select S

(R  $\times$  S) (Right Outer Join)

R  $=$  C

	A	B	C	D
1	a	1	g	
2	b	2	h	
3	c	3	i	
4	d	N	N	
5	e	N	N	
6	f	N	N	

$\Rightarrow$  Select S

108

## 10. AGGREGATE FUNCTIONS 1

The Aggregate functions are Avg, min, max, sum, count

Employee

Eno	Ename	Salary	Hra
1	A	1	2
2	B	2	4
3	C	4	6
4	D	4	8
5	E	4	Null
6	F	Null	6

① SELECT sum(Salary), avg(Salary)

FROM Employee

Sum	Avg
15	3

② SELECT sum(Salary) AS total , avg(Salary)

as Average From Employee

Total	Average
15	3

⇒ while calculating the Average Null values are not considered

3) SELECT (max(Salary)-min(Salary)) as diff from Employee :

Diff
3

4) SELECT COUNT(\*) as Total From Employee :

Total
6

→ min, max are applied on Numbers, String, Dates

⇒ SELECT max(name) From Employee :

Name
E

C. i) ⇒ Select sum(name) From Employee X not valid  
⇒ Avg, Sum are applied only on Numerical Data not on any other data types.

⇒ SELECT COUNT(Salary) FROM Employee

5
5

⇒ Select COUNT(Salary) FROM Employee

4
---

⇒ Select COUNT(Salary + Hra) FROM Employee

10
----

⇒ Select sum(Salary + Hra) From Employee

31
----

(3+6+10)

⇒ Select sum(Salary + Hra) From Employee

10
----

(3+6+10)

## All Aggregate Functions 2

$$\text{COUNT}(\text{Salary}) = 5$$

$$\text{COUNT(DISTINCT Salary)} = 3$$

$$\text{SUM(Salary)} = 15$$

$$\text{SUM(DISTINCT Salary)} = 7$$

$$\text{AVG(Salary)} = \frac{\text{SUM}(Salary)}{\text{COUNT}(Salary)} = \frac{15}{5} = 3$$

$$\min(\text{Salary}) = 1$$

$$\min(\text{DISTINCT Salary}) = 1$$

$$\max(\text{Salary}) = 4$$

$$\max(\text{DISTINCT Salary}) = 4$$

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

2) Group By

R  $\rightarrow$  Select \* From R Group By A.

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

3) Select \* From R Group By A,B

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

4) Select A,B COUNT(\*) From R Group By A,B

Group By A,B whenever you have some attributes in

A	B	No of rows
1	2	2
2	3	2

→ No of 2's appear in select clause}

Transaction

1) Atomicity

2) Consistency

3) Isolation

4) Durability

⇒ Every Null value will be treated as a separate group. (null group)

From Employee

Group By DNO

↳ Durability

1. TRANSACTION MANAGEMENT AND CONCURRENCY CONTROL 11  
 1) Single Transaction:  
 A Transaction is a collection of operations that forms a single logical unit of work.

- 2) Concurrent Transaction:  
 A Transaction is a collection of operations that forms a single logical unit of work.

- 3) Atomicity: All operations done by a Transaction must become permanent. - Re-Owning Manager takes care

### Transferring money

$$A_{\text{Initial}} = 1000$$

$$S(A_{\text{Initial}}) = 1$$

$$R(A) = T_1$$

$$A = A - 500; -900$$

$$W(A) = T_2$$

$$B = B + 500$$

$$W(B) = T_3$$

$$B = B + 500$$

$$W(B) = T_4$$

$$B = B + 500$$

$$W(B) = T_5$$

$$B = B + 500$$

$$W(B) = T_6$$

$$B = B + 500$$

$$W(B) = T_7$$

$$B = B + 500$$

$$W(B) = T_8$$

$$B = B + 500$$

$$W(B) = T_9$$

$$B = B + 500$$

$$W(B) = T_{10}$$

$$B = B + 500$$

$$W(B) = T_{11}$$

$$B = B + 500$$

$$W(B) = T_{12}$$

$$B = B + 500$$

$$W(B) = T_{13}$$

$$B = B + 500$$

$$W(B) = T_{14}$$

$$B = B + 500$$

$$W(B) = T_{15}$$

$$B = B + 500$$

$$W(B) = T_{16}$$

$$B = B + 500$$

$$W(B) = T_{17}$$

$$B = B + 500$$

$$W(B) = T_{18}$$

$$B = B + 500$$

$$W(B) = T_{19}$$

$$B = B + 500$$

$$W(B) = T_{20}$$

$$B = B + 500$$

$$W(B) = T_{21}$$

$$B = B + 500$$

$$W(B) = T_{22}$$

$$B = B + 500$$

$$W(B) = T_{23}$$

$$B = B + 500$$

$$W(B) = T_{24}$$

$$B = B + 500$$

$$W(B) = T_{25}$$

$$B = B + 500$$

$$W(B) = T_{26}$$

$$B = B + 500$$

$$W(B) = T_{27}$$

$$B = B + 500$$

$$W(B) = T_{28}$$

$$B = B + 500$$

$$W(B) = T_{29}$$

$$B = B + 500$$

$$W(B) = T_{30}$$

$$B = B + 500$$

$$W(B) = T_{31}$$

$$B = B + 500$$

$$W(B) = T_{32}$$

$$B = B + 500$$

$$W(B) = T_{33}$$

$$B = B + 500$$

$$W(B) = T_{34}$$

$$B = B + 500$$

$$W(B) = T_{35}$$

$$B = B + 500$$

$$W(B) = T_{36}$$

$$B = B + 500$$

$$W(B) = T_{37}$$

$$B = B + 500$$

$$W(B) = T_{38}$$

$$B = B + 500$$

$$W(B) = T_{39}$$

$$B = B + 500$$

$$W(B) = T_{40}$$

$$B = B + 500$$

$$W(B) = T_{41}$$

$$B = B + 500$$

$$W(B) = T_{42}$$

$$B = B + 500$$

$$W(B) = T_{43}$$

$$B = B + 500$$

$$W(B) = T_{44}$$

$$B = B + 500$$

$$W(B) = T_{45}$$

$$B = B + 500$$

$$W(B) = T_{46}$$

$$B = B + 500$$

$$W(B) = T_{47}$$

$$B = B + 500$$

$$W(B) = T_{48}$$

$$B = B + 500$$

$$W(B) = T_{49}$$

$$B = B + 500$$

$$W(B) = T_{50}$$

$$B = B + 500$$

$$W(B) = T_{51}$$

$$B = B + 500$$

$$W(B) = T_{52}$$

$$B = B + 500$$

$$W(B) = T_{53}$$

$$B = B + 500$$

$$W(B) = T_{54}$$

$$B = B + 500$$

$$W(B) = T_{55}$$

$$B = B + 500$$

$$W(B) = T_{56}$$

$$B = B + 500$$

$$W(B) = T_{57}$$

$$B = B + 500$$

$$W(B) = T_{58}$$

$$B = B + 500$$

$$W(B) = T_{59}$$

$$B = B + 500$$

$$W(B) = T_{60}$$

$$B = B + 500$$

$$W(B) = T_{61}$$

$$B = B + 500$$

$$W(B) = T_{62}$$

$$B = B + 500$$

$$W(B) = T_{63}$$

$$B = B + 500$$

$$W(B) = T_{64}$$

$$B = B + 500$$

$$W(B) = T_{65}$$

$$B = B + 500$$

$$W(B) = T_{66}$$

$$B = B + 500$$

$$W(B) = T_{67}$$

$$B = B + 500$$

$$W(B) = T_{68}$$

$$B = B + 500$$

$$W(B) = T_{69}$$

$$B = B + 500$$

$$W(B) = T_{70}$$

$$B = B + 500$$

$$W(B) = T_{71}$$

$$B = B + 500$$

$$W(B) = T_{72}$$

$$B = B + 500$$

$$W(B) = T_{73}$$

$$B = B + 500$$

$$W(B) = T_{74}$$

$$B = B + 500$$

$$W(B) = T_{75}$$

$$B = B + 500$$

$$W(B) = T_{76}$$

$$B = B + 500$$

$$W(B) = T_{77}$$

$$B = B + 500$$

$$W(B) = T_{78}$$

$$B = B + 500$$

$$W(B) = T_{79}$$

$$B = B + 500$$

$$W(B) = T_{80}$$

$$B = B + 500$$

$$W(B) = T_{81}$$

$$B = B + 500$$

$$W(B) = T_{82}$$

$$B = B + 500$$

$$W(B) = T_{83}$$

$$B = B + 500$$

$$W(B) = T_{84}$$

$$B = B + 500$$

$$W(B) = T_{85}$$

$$B = B + 500$$

$$W(B) = T_{86}$$

$$B = B + 500$$

$$W(B) = T_{87}$$

$$B = B + 500$$

$$W(B) = T_{88}$$

$$B = B + 500$$

$$W(B) = T_{89}$$

$$B = B + 500$$

$$W(B) = T_{90}$$

$$B = B + 500$$

$$W(B) = T_{91}$$

$$B = B + 500$$

$$W(B) = T_{92}$$

$$B = B + 500$$

$$W(B) = T_{93}$$

$$B = B + 500$$

$$W(B) = T_{94}$$

$$B = B + 500$$

$$W(B) = T_{95}$$

$$B = B + 500$$

$$W(B) = T_{96}$$

$$B = B + 500$$

$$W(B) = T_{97}$$

$$B = B + 500$$

$$W(B) = T_{98}$$

$$B = B + 500$$

$$W(B) = T_{99}$$

$$B = B + 500$$

$$W(B) = T_{100}$$

$$B = B + 500$$

$$W(B) = T_{101}$$

$$B = B + 500$$

$$W(B) = T_{102}$$

$$B = B + 500$$

$$W(B) = T_{103}$$

$$B = B + 500$$

$$W(B) = T_{104}$$

$$B = B + 500$$

$$W(B) = T_{105}$$

$$B = B + 500$$

$$W(B) = T_{106}$$

$$B = B + 500$$

$$W(B) = T_{107}$$

$$B = B + 500$$

$$W(B) = T_{108}$$

$$B = B + 500$$

$$W(B) = T_{109}$$

$$B = B + 500$$

$$W(B) = T_{110}$$

$$B = B + 500$$

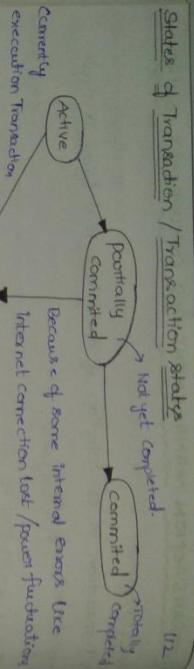
$$W(B) = T_{111}$$

$$B = B + 500$$

$$W(B) = T_{112}$$

$$B = B + 500$$

## States of Transaction / Transaction State



## dirty Read problem

T <sub>1</sub>	A=100
	Read(A)
	A=A-50
	Write(A)

T <sub>1</sub>	A=50
	Roll Back
	Because of Roll back
	$\Rightarrow A=100$ but the transaction is not considered

## Unrepeatable Read

T <sub>1</sub>	A=100
	Read(A)
	A=A-100
	Write(A)

- The Reason why we need concurrent execution is:
- > For Avoiding long waiting time
  - > Transaction consists of multiple steps. Some involve I/O activities and other involve CPU activities. And Computer System CPU and I/O operations can be done in parallel. Therefore, I/O activities can be done in parallel with processing at the CPU.
  - > The processor and disk utilization increases.
- Schedule:  
It represents the order in which instructions of a transaction are executed.

## loss update problem: (w-w conflict)

Initial value of A=100	
A → B	A → n
A = 100	
Read(A)	
A = 50	
A: A = 50	
Read(A) → A = 100	
X → 0.04A → A = <del>0.04 * 100</del>	
A = A + X → A = 104	
A = 104	
write(A)	
A = 104	
write(A) → A = 104	
A = 50 × $\frac{1}{100}^2 = 2.5$	
A = B + 50	
B = B + 2.5	

Test: Two write operations of diff transactions and in them there is no Read then 2nd write operation overwrites 1st write operation

## Phantom Type C

T <sub>1</sub>	
1	Employee
2	A
3	50
4	C
5	40
6	Select * from Employee
7	salary > 3000

Roll

Fail

Because

of Roll back

$\Rightarrow A=100$  but the transaction is not considered

11.2

Dirty Read problem (Write conflict) (write-read conflict)

11.3

Totally completed	
$A = 100$	$T_1$
$A = 50$	$T_2$
$A = 50$	Write(A)
	Roll Back
	Read(A)
	$x = 0$ (initial) $x = 2$
	$A = 52$
	Write(A) $\rightarrow A = 52$
	(Temporarily saved this transaction is not committed)
	Fail
	$B = B + 50$
	Because
	of Roll back
	$\rightarrow A = 100$ but the modifications done by $T_2$ will not be saved and not considered

Unrepeatable Read problem (R-R) Non-repeatable Read Problem (R-w)

⇒ when a transaction tries to Read the value of data item twice and another transaction updates the same data item in between the two Read Operations of the 1st transaction, acc. result the 1st Transaction Reads varied values of some data item during its execution, this is called un-repeatable reads.

Phantom Tuple (phantom phenomenon)

		$T_1$	$T_2$
	<u>eno</u>	<u>ename</u>	<u>salary</u>
1	1	A	5000
3	3	C	4000
Actual ans		Select * from Emp where salary > 3000	
0	0		Insert into Emp values (4,0,3500);

After

 $A = 52$ 

$$50 \times \frac{50}{100} = 25$$

$$+ 25 = 50 + 25 = 52$$

After

 $A = 52$ 

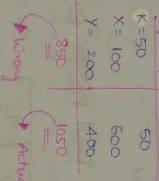
$$50 \times \frac{52}{100} = 26$$

$$+ 26 = 50 + 26 = 52$$

### Incorrect summary problem

T <sub>1</sub>	T <sub>2</sub>
Sum=0	$\rightarrow$ Sum=0
Read(x)	$\rightarrow$ Let K=50
Sum=Sum+K; $\rightarrow$ Sum=50	
X=100 $\leftarrow$ Read(x)	
X=600 $\leftarrow$ X+500	
X=600 $\leftarrow$ write(x)	
Read(x)	$\rightarrow$ X=600
Sum=Sum+x; $\rightarrow$ Sum=650	
Read(y)	$\rightarrow$ Let y=200
Sum=Sum+y; $\rightarrow$ Sum=850	
Read(y)	
y=y+200	
write(y)	

Because of this operation  
'y' will be changed and  
and as a result sum must  
be changed but if we retrieve  
sum it gives 850.



### 3. No. of Schedules possible

Let us say there are two operations associated with a particular Transaction and there are two transactions T<sub>1</sub> and T<sub>2</sub>. Now what are the diff ways to combine these two together.

T <sub>1</sub>	T <sub>2</sub>
a <sub>1</sub>	b <sub>1</sub>
a <sub>2</sub>	b <sub>2</sub>
b <sub>1</sub>	a <sub>1</sub>
b <sub>2</sub>	a <sub>2</sub>

In these schedules the order should be condition  
which means a<sub>1</sub> should be executed only before a<sub>2</sub>  
and a<sub>2</sub> should not be executed before a<sub>1</sub>.

$\Rightarrow$  All these schedules will not give us correct results (not consistent)  
 $\therefore$  we need to find the correct schedule that guarantees the consistency

### of database

#### 4. TYPES OF

##### Serial schedule

##### Transactions T<sub>1</sub> $\rightarrow$ T<sub>2</sub>

T<sub>1</sub>  $\rightarrow$  T<sub>2</sub>

R(A) R(A)

A=100 A=100

w(A) w(A)

R(B) R(B)

A=100 A=100

w(B) w(B)

R(C) R(C)

A=100 A=100

w(C) w(C)

T <sub>1</sub>	T <sub>2</sub>
R(A)	R(A)
A=100	A=100
w(A)	w(A)
R(B)	R(B)
A=100	A=100
w(B)	w(B)
R(C)	R(C)
A=100	A=100
w(C)	w(C)

Operations of

$\Rightarrow$  serial sched

$\Rightarrow$  when Transac-

state

$\Rightarrow$  If there are

Complete sched

$\Rightarrow$  At the end

present then

T <sub>1</sub>	T <sub>2</sub>
R(A)	R(A)
A=100	A=100
w(A)	w(A)
Commit	Commit
A=100	A=100
w(A)	w(A)
Abort	Abort
R(B)	R(B)
A=100	A=100
w(B)	w(B)
Commit	Commit
A=100	A=100
w(B)	w(B)
Abort	Abort
R(C)	R(C)
A=100	A=100
w(C)	w(C)
Commit	Commit
A=100	A=100
w(C)	w(C)

Recoverable as

No -

Recov -

Recov

No -

Recov

## 5 TYPES OF SCHEDULES

Serial schedule: serial schedule means one after the other. If we have two transactions  $T_1$  and  $T_2$ , then  $T_1, T_2$  and  $T_2, T_1$  are called serial schedules.

$T_1 \rightarrow T_2$	$T_2 \rightarrow T_1$
$R(A)$	$R(A)$
$A = A - 50$	$R(A)$
$w(A)$	$R(A)$

$T_1$	$T_2$
$R(A)$	$R(A)$
$A = A - 50$	$R(A)$
$w(A)$	$R(A)$

$T_1$	$T_2$
$R(A)$	$R(A)$
$A = A - 50$	$R(A)$
$w(A)$	$R(A)$

memory

operations

- Serial schedules are the ones that don't interleave the actions of operations of different transaction.
- When transactions are executing serially then they ensure a consistent state.

If there are  $n$  transactions then there are ' $n!$ ' serial schedules.

Complete schedule

- At the end of every transaction in the lines commit and after one present then the schedule is called complete schedule.

Non-serial

$T_1$	$T_2$
$R(A)$	$R(A)$
$A = A - 50$	$R(A)$
$w(A)$	$R(A)$

$T_1$	$T_2$
$R(A)$	$R(A)$
$A = A - 50$	$R(A)$
$w(A)$	$R(A)$

Roll Back  
because it is committed

$T_1$	$T_2$
$R(A)$	$R(A)$
$A = A - 50$	$R(A)$
$w(A)$	$R(A)$

$T_1$	$T_2$
$R(A)$	$R(A)$
$A = A - 50$	$R(A)$
$w(A)$	$R(A)$

Roll Back  
because it is committed

$T_1$	$T_2$
$R(A)$	$R(A)$
$A = A - 50$	$R(A)$
$w(A)$	$R(A)$

$T_1$	$T_2$
$R(A)$	$R(A)$
$A = A - 50$	$R(A)$
$w(A)$	$R(A)$

Roll Back  
because it is committed

$T_1$	$T_2$
$R(A)$	$R(A)$
$A = A - 50$	$R(A)$
$w(A)$	$R(A)$

Roll Back  
because it is committed

$T_1$	$T_2$
$R(A)$	$R(A)$
$A = A - 50$	$R(A)$
$w(A)$	$R(A)$

Roll Back  
because it is committed

Non-serial

Recoverable schedule

$T_1$	$T_2$
$R(A)$	$R(A)$
$A = A - 50$	$R(A)$
$w(A)$	$R(A)$

$T_1$	$T_2$
$R(A)$	$R(A)$
$A = A - 50$	$R(A)$
$w(A)$	$R(A)$

Roll Back  
because it is committed

$T_1$	$T_2$
$R(A)$	$R(A)$
$A = A - 50$	$R(A)$
$w(A)$	$R(A)$

Roll Back  
because it is committed

$T_1$	$T_2$
$R(A)$	$R(A)$
$A = A - 50$	$R(A)$
$w(A)$	$R(A)$

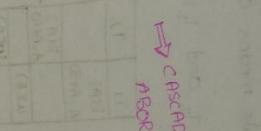
Roll Back  
because it is committed

$T_1$	$T_2$
$R(A)$	$R(A)$
$A = A - 50$	$R(A)$
$w(A)$	$R(A)$

## CASCADING SCHEDULE

116

T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>
R(A) w(A)		
	R(A) w(A)	
		R(B) w(B)
		w(B)



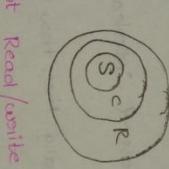
T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>
R(A) w(A)		
	R(A) w(A)	
		R(B) w(B)
		w(B)

## STRICT SCHEDULE

S = STRICT SCHEDULE

C = CASCADELESS SCHEDULE

R = RECOVERABLE SCHEDULE



strict schedule

strict schedule has no cascading writes for two or more transactions held at snapshot.

T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>
R(A) w(A)		
	R(A) w(A)	
		R(B) w(B)
		w(B)

→ g absent Read / write until the other one Commit.

strict schedule

T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>
R(A) w(A)		
	R(A) w(A)	
		R(B) w(B)
		w(B)

T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>
R(A) w(A)		
	R(A) w(A)	
		R(B) w(B)
		w(B)

T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>
R(A) w(A)		
	R(A) w(A)	
		R(B) w(B)
		w(B)

T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>
R(A) w(A)		
	R(A) w(A)	
		R(B) w(B)
		w(B)

T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>
R(A) w(A)		
	R(A) w(A)	
		R(B) w(B)
		w(B)

T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>
R(A) w(A)		
	R(A) w(A)	
		R(B) w(B)
		w(B)

T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>
R(A) w(A)		
	R(A) w(A)	
		R(B) w(B)
		w(B)

T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>
R(A) w(A)		
	R(A) w(A)	
		R(B) w(B)
		w(B)

T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>
R(A) w(A)		
	R(A) w(A)	
		R(B) w(B)
		w(B)

T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>
R(A) w(A)		
	R(A) w(A)	
		R(B) w(B)
		w(B)

T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>
R(A) w(A)		
	R(A) w(A)	
		R(B) w(B)
		w(B)

T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>
R(A) w(A)		
	R(A) w(A)	
		R(B) w(B)
		w(B)

T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>
R(A) w(A)		
	R(A) w(A)	
		R(B) w(B)
		w(B)

T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>
R(A) w(A)		
	R(A) w(A)	
		R(B) w(B)
		w(B)

T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>
R(A) w(A)		
	R(A) w(A)	
		R(B) w(B)
		w(B)

T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>
R(A) w(A)		
	R(A) w(A)	
		R(B) w(B)
		w(B)

T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>
R(A) w(A)		
	R(A) w(A)	
		R(B) w(B)
		w(B)

T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>
R(A) w(A)		
	R(A) w(A)	
		R(B) w(B)
		w(B)

T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>
R(A) w(A)		
	R(A) w(A)	
		R(B) w(B)
		w(B)

T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>
R(A) w(A)		
	R(A) w(A)	
		R(B) w(B)
		w(B)

T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>
R(A) w(A)		
	R(A) w(A)	
		R(B) w(B)
		w(B)

T

of database.

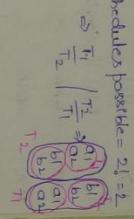
Now let us say there are Transactions  $T_1, T_2, T_3, \dots, T_m$  and no. of operations in each transaction is  $n_1, n_2, n_3, \dots, n_m$  then the no. of schedules that are possible are

$$\frac{(n_1 + n_2 + n_3 + \dots + n_m)!}{(n_1!) (n_2!) (n_3!) \dots (n_m!)}$$

$$\text{No. of schedules possible} = \frac{(n_1 + n_2 + n_3 + \dots + n_m)!}{(n_1!) (n_2!) (n_3!) \dots (n_m!)}$$

→ Now if there is no interleaving (No Transaction is supposed to enter in middle when a particular transaction is executing) then the no. of serial schedules possible are  $\underline{\underline{m!}}$

In the above example the no. of serial schedules possible =  $2! = 2$



→ All serial schedules are always consistent.

### 5. RESULT EQUIVALENT

Two schedules are said to be Equivalent if they follow some rules.

they are

1) Two schedules are said to be Result Equivalent if they produce same final database state for a given initial state of database.

	S1	S2
1	R(A)	R(A)
2	A = 100	A = 100
3	A = 100	A = 100
4	A * 10	A * 10
5	W(A)	W(A)
6	A = 100	A = 100

• consistent  
ly before a  
transaction

⇒ check whether these two schedules are equivalent / not

18

	<u>S1</u>	<u>S2</u>
	T1	T2
R(x)		x, y
x = x+5		1, 5
w(x)		2, 5
		9, 10
		11, 10
R(x)		R(x)
x = x*3		x = x*3
w(x)		w(x)
R(y)		R(y)
y = y+5		y = y+5
w(y)		w(y)

No - Conf

1) Test which

S1: R1(A) R2

S2: R2(C)

S1

S2

T1

R(A)

w(A)

T2

R(C)

w(C)

No - Conf

2)

S1:

R(A)

w(A)

T1

R(A)

w(A)

T2

R(A)

w(A)

R(B)

w(B)

T1

R(A)

w(A)

T2

R(B)

w(B)

T1

R(A)

w(A)

T2

R(B)

w(B)

T1

R(A)

w(A)

T2

R(B)

w(B)

## 5. CONFLICT EQUIVALENT AND CONFLICT SERIALIZABLE

Conflict operations

T1

T2

R(A)

w(A)

R(A)

w(A)

R(A)

w(A)

R(A)

w(A)

R(A)

w(A)

⇒ Both the transactions are accessing the same data items and one of the operation is write then it is a conflict operation.

Non-Conflict operations

Now, Two schedules are said to be conflict Equivalent if all the conflicting operations in both the schedules must be executed in the same order.

w1

w2

T1

R(A)

w(A)

T2

R(A)

w(A)

T1

R(A)

w(A)

T2

R(A)

w(A)

T1

R(A)

w(A)

T2

R(A)

w(A)

Conflict

A schedule

Equivalent

Conflict

Conflict</p



### PROCEDURE

- ① Construct a transaction operation.
- ② If the Dine is not conflict
- ③ If the graph conflict set

Ex

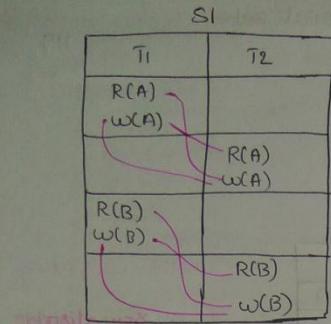
T1	T2
R(A) w(A)	R(A) w(A)
R(B) w(B)	R(B) w(B)

Ex :

T1	T2
R(A) w(A)	R(A) w(A)
R(B) w(B)	R(B) w(B)

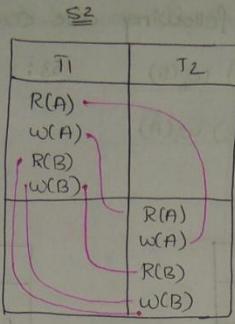
Ex :

T1	T2
	R(A)
w(A)	R(B)
	w(B)



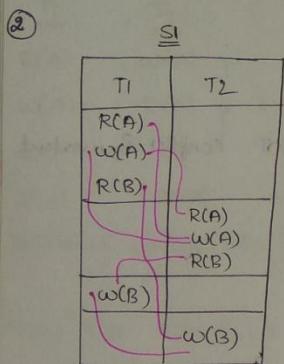
$$\begin{aligned} R_1(A) &\rightarrow w_2(A) \\ w_1(A) &\rightarrow R_2(A) \\ w_1(A) &\rightarrow w_2(A) \end{aligned}$$

$$\begin{aligned} R_1(B) &\rightarrow w_2(B) \\ w_1(B) &\rightarrow R_2(B) \\ w_1(B) &\rightarrow w_2(B) \end{aligned}$$

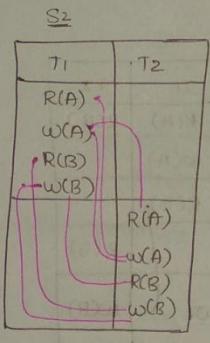


$$\begin{aligned} R_1(A) &\rightarrow w_2(A) \\ w_1(A) &\rightarrow R_2(A) \\ w_1(A) &\rightarrow w_2(A) \\ R_1(B) &\rightarrow w_2(B) \\ w_1(B) &\rightarrow R_2(B) \\ w_1(B) &\rightarrow w_2(B) \end{aligned}$$

Conflict  
Serializable



$$\begin{aligned} R_1(A) &\rightarrow w_2(A) \\ w_1(A) &\rightarrow R_2(A) \\ w_1(A) &\rightarrow w_2(A) \\ R_1(B) &\rightarrow w_2(B) \\ w_1(B) &\rightarrow w_2(B) \end{aligned}$$



$$\begin{aligned} R_1(A) &\rightarrow w_2(A) \\ w_1(A) &\rightarrow R_2(A) \\ R_1(B) &\rightarrow w_2(B) \\ w_1(B) &\rightarrow R_2(B) \\ w_1(B) &\rightarrow w_2(B) \end{aligned}$$

Not Conflict Equivalent.

∴ The Relation S1 is not equivalent to any of the serializable schedule.

∴ S1 is not conflict serializable.

### PROCEDURE FOR CONFLICT SERIALIZABILITY USING PRECEDENCE GRAPH

- ① construct a directed graph where each vertex corresponds to a transaction and each directed edge represents a conflicting operation. (Read→write / write-write/write→Read)
- ② If the Directed Graph contains cycles then the concurrent schedule is not conflict serialisable.
- ③ If the graph contains no cycles then the schedule is called conflict serializable.

conflict  
serializable

Ex

T1	T2
R(A)	
w(A)	
	R(A)
	w(A)
R(B)	
w(CB)	
	R(B)
	w(B)



⇒ In the precedence Graph there is no cycle and therefore this schedule is conflict serializable

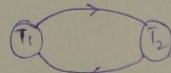
⇒ The serial schedule for which this schedule is equivalent to is Topological order of the Graph = T<sub>1</sub>T<sub>2</sub>.

(A)  $\xrightarrow{\text{NO}}$   
(B) F

Equivalent.

Ex :

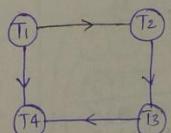
T1	T2
R(A)	
w(A)	
R(B)	
	R(A)
	w(A)
	R(C)
w(CB)	
	w(B)



⇒ In the precedence Graph there is a cycle and hence the schedule is not conflict serializable

Ex :

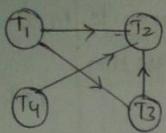
T1	T2	T3	T4
	R(X)		
		w(X)	
w(Y)			
	R(Y)		
		w(Z)	
			R(X)
			R(Y)



There is no cycle and therefore the schedule is conflict serializable

⇒ The serial schedule to which the given schedule is T<sub>1</sub>T<sub>2</sub>T<sub>3</sub>T<sub>4</sub> (Topological sort)

Ex



④ Not conflict serializable

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

⇒ No cycles in Graph ⇒ Conflict

⑥ T<sub>1</sub> T<sub>4</sub> T<sub>3</sub> T<sub>2</sub>

Serializable

⑦ T<sub>2</sub> T<sub>3</sub> T<sub>1</sub> T<sub>4</sub>

Conflict

TRICK (By me) (not By RBR) : Start the sequence which does not have any incoming edge and proceed further.

1	2	3	4
R <sub>1</sub> (A)			
w <sub>1</sub> (A)			

### 8. NO. OF CONFLICT SERIALIZABLE SCHEDULES

T<sub>1</sub>: R<sub>1</sub>(A) W<sub>1</sub>(A) R<sub>1</sub>(B) W<sub>1</sub>(B)

T<sub>2</sub>: R<sub>2</sub>(A) W<sub>2</sub>(A) R<sub>2</sub>(B) W<sub>2</sub>(B)

Find the total no of conflict serializable schedules that can be formed by T<sub>1</sub> and T<sub>2</sub>?

Now, try to find the schedules that are equivalent to (T<sub>1</sub> → T<sub>2</sub>) or (T<sub>2</sub> → T<sub>1</sub>). (The schedules equivalent to T<sub>1</sub> → T<sub>2</sub> / T<sub>2</sub> → T<sub>1</sub> are called conflict serializable)

Now, try to find out the schedules that are conflict equivalent to the serial schedule (T<sub>1</sub> → T<sub>2</sub>)

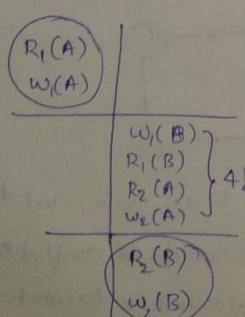
Now, Among {R<sub>1</sub>(A) W<sub>1</sub>(A) R<sub>1</sub>(B) W<sub>1</sub>(B)} {R<sub>2</sub>(A) W<sub>2</sub>(A) R<sub>2</sub>(B) W<sub>2</sub>(B)}

R<sub>1</sub>(A) should come first.

cannot write R<sub>2</sub>(A) 1st because

if I write it get T<sub>2</sub> → T<sub>1</sub> (but my aim is T<sub>1</sub> → T<sub>2</sub>)

∴ R<sub>1</sub>(A)  $\xrightarrow{T_1 \rightarrow T_2}$  and R<sub>2</sub>(B)  $\xrightarrow{w_2(B)}$  should come in end



4! ways but R<sub>1</sub>W<sub>1</sub> and R<sub>2</sub>W<sub>2</sub> should be executed as group

∴ No. of conflict serializable schedules

$$= \frac{4!}{2!2!} = 6$$

### g. VIEW E

Two sched  
following 3

1> For each  
in sched  
read th

⇒ If Trans  
produced  
s' also s

3> For each  
of 'A' of  
in s.

⇒ In summar  
same order

Ex:

T <sub>1</sub>			
R(A)			
W(A)			

T <sub>2</sub>			
R(B)			
W(B)			

12.2

Now  $(T_2 \rightarrow T_1)$ 

$\Rightarrow$  conflict  
serializable

s. not have

$R_2(A)$	
$w_2(A)$	
$R_1(B)$	$R_1(B)$
$w_2(B)$	$w_1(B)$
$R_1(A)$	
$w_1(A)$	

12.3

### 9. VIEW EQUIVALENT AND SERIALIZABLE

Two schedules 'S' and 'S'' are said to be view Equivalent if the following 3-Conditions are met for each data item (say A)

- 1) For each data item A if Transaction  $T_i$  reads the initial value of 'A' in schedule 'S' then transaction  $T_i$  must schedule 'S'' also read this initial value of 'A'.
- 2) If Transaction  $T_i$  executes Read - R(A) in schedule 'S' and that was produced by Transaction  $T_j$  (if any) then the transaction must in schedule 'S'' also read the value of 'A' that was produced by  $T_j$ .
- 3) For each data item the transaction (if any) that performs final write of 'A' operation in 'S' must also perform the final write of 'A' operation in 'S'.

$\Rightarrow$  In summary All Read write sequences to be maintained in the same order in both S and S'

Ex:

S1		S2	
$T_1$	$T_2$	$T_1$	$T_2$
R(A)		R(A)	
w(A)		w(A)	
	R(A)		R(B)
	w(A)		w(B)
R(B)			R(A)
w(B)			w(A)
	R(B)		R(B)
	w(B)		w(B)

1) Are S1 and S2 view

Equivalent

Initial Read on 'B' in 'S1' is by  $T_1$

Initial Read on 'B' in 'S2' is by  $T_2$

First write on 'B' in 'S1' is performed by  $T_1$ .

Initial Read on 'A' in 'S2' is performed by  $T_2$ .

Final write on 'A' in 'S' is done by  $T_2$ .

one first. I  
(A) 1st because

 $T_2 \rightarrow T_1$  (but my

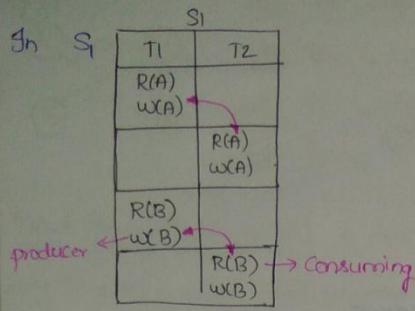
acted as group

Serializable schedules

Now, check for producers and consumers. (Write and Read)

124

S <sub>1</sub>	
T <sub>1</sub>	T <sub>2</sub>
R(A) W(A)	
	R(A) W(A)
R(B) W(B)	
	R(B) W(B)



S <sub>2</sub>	
T <sub>1</sub>	T <sub>2</sub>
R(A) W(A)	
R(B) W(B)	
	R(A) W(A)
	R(B) W(B)

producer ←

Ex

T <sub>1</sub>
R(A)
W(A)

Ex

T <sub>1</sub>
R(A)
W(A)

∴ These two schedules are view equivalent.

Ex:

S <sub>1</sub>	T <sub>1</sub>	T <sub>2</sub>
	R(A)	
	W(A)	
		W(A)

Final write here is by T<sub>1</sub>

S <sub>2</sub>	T <sub>1</sub>	T <sub>2</sub>
	R(A)	
	W(A)	

Final write here is by T<sub>2</sub>

S <sub>3</sub>	T <sub>1</sub>	T <sub>2</sub>
	producer	W(A)
	R(A)	W(A)

producer ← W(A)

∴ T<sub>1</sub> ≠ T<sub>2</sub> (Not view Equivalent)

S<sub>1</sub> ≠ S<sub>2</sub>

S<sub>2</sub> ≠ S<sub>3</sub>

(final write of A's conflict)

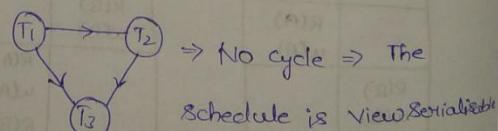
S<sub>1</sub> ≠ S<sub>3</sub>

(No producer consumer in S<sub>1</sub>)

## 10. VIEW SERIALISABLE EXAMPLES

The method that we use to check view serializability is by using "poly graphs."

T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>
R(A)		
	W(A)	
R(A)		
		W(A)



⇒ Here T<sub>1</sub> should start first which means T<sub>2</sub> & T<sub>3</sub> should be executed only after T<sub>1</sub>, so T<sub>1</sub> → T<sub>2</sub> → T<sub>3</sub>

⇒ Now coming to T<sub>2</sub> and T<sub>3</sub>, T<sub>2</sub> should start first so T<sub>1</sub> → T<sub>2</sub> → T<sub>3</sub>

Blind write:

⇒ If a schedule has blind writes then there is

## 11. SERIALIZABLE

⇒ A schedule

⇒ View se

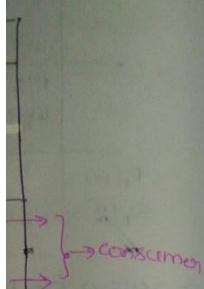
⇒ Now, consi

d)

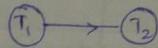
124

Ex

T1	T2
R(A) W(A)	
	R(A) W(A)
R(B) W(B)	
	R(B) W(B)



125



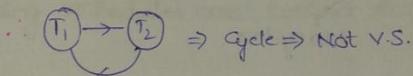
∴ The Graph contains NO cycle and so the schedule is view serializable and conflict serializable

Ex

T1	T2
R(A)	
	R(A)
W(A)	

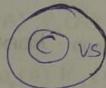
Says that T1 should happen first and then T2

This Says that T1 should occur last which means T2 and then T1



∴  $(T_1 \rightarrow T_2) \Rightarrow \text{cycle} \Rightarrow \text{Not V.S.}$

⇒ If a schedule is conflict serializable then it is view serializable but a view serializable schedule need not be conflict serializable



Blind write: Write without Read is called Blind write

⇒ If a schedule should be view serializable but not conflict serializable then there should be atleast one blind operation.

## II. SERIALIZABLE SCHEDULES

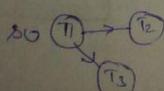
⇒ A schedule is view serializable if it is either conflict serializable or view serializable

ability is by

e ⇒ The

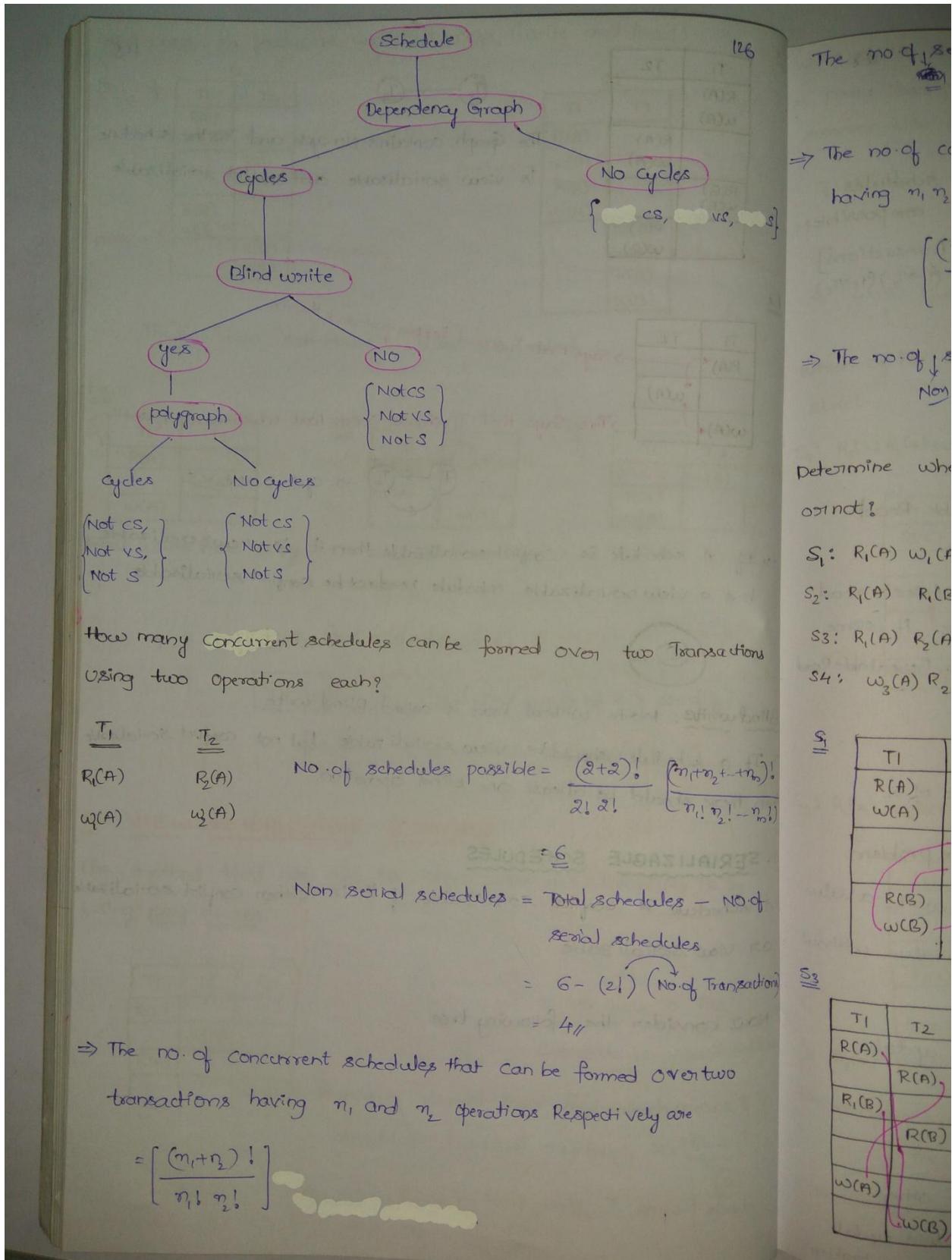
is View Serializable

which means  $T_2, T_3$



should start

⇒ Now, consider the following tree



The no. of serial schedules possible are  $\frac{(n_1+n_2)!}{n_1! n_2!} - 2$

⇒ The no. of concurrent schedules that can be formed over m-transactions having  $n_1, n_2, \dots, n_m$  operations respectively are

$$\left[ \frac{(n_1+n_2+n_3+\dots+n_m)!}{n_1! n_2! n_3! \dots n_m!} \right]$$

⇒ The no. of serial schedules are  $\left[ \frac{(n_1+n_2+\dots+n_m)!}{n_1! n_2! n_3! \dots n_m!} \right] - (m!)$

Determine whether the following schedules are conflict serializable or not.

$S_1: R_1(A) W_1(A) R_2(B) W_2(B) R_1(B) W_1(B)$

$S_2: R_1(A) R_1(B) W_2(A) R_3(A) W_1(B) W_3(A) R_2(B) W_2(B)$

$S_3: R_1(A) R_2(A) R_1(B) R_3(B) R_3(A) W_1(A) W_2(B)$

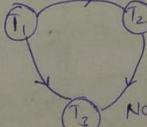
$S_4: W_3(A) R_2(A) W_1(B) R_2(B) W_2(C) R_3(C)$

$S_1$	
T1	T2
R(A)	
W(A)	
	R(B)
	W(B)
	R(B)
	W(B)



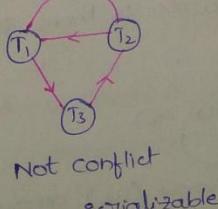
∴ conflict serializable

$S_2$		
T1	T2	T3
R(A)		
R(B)		
	W(A)	R(A)
	W(B)	
		R(B)
		W(A)
		W(B)

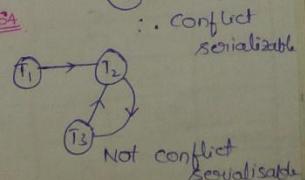


No cycles  
∴ conflict serializable

$S_3$		
T1	T2	T3
R(A)		
R(A)		
R(B)		R(B)
	R(B)	
W(A)		
	W(B)	
		R(B)



Not conflict  
serializable



No cycles  
∴ conflict serializable

Not conflict  
serializable

## 12. Examples on Types of Schedules

128

Two Transactions  $T_1$  and  $T_2$  are given as follows

$T_1: R_1(A) W_1(A) R_1(B) W_1(B)$

$T_2: R_2(B) W_2(B) R_2(A) W_2(A)$  Now How many serial schedules are possible?

$$\Rightarrow \text{No. of serial schedules} = m! = 2! = 2 \quad [m = \text{No. of Transactions}]$$

$(T_1 \rightarrow T_2) (T_2 \rightarrow T_1)$

$\Rightarrow$  Consider the following schedules

$S_1: R_2(x) R_2(y) R_1(x) R_1(y) W_1(x) \underbrace{R_2(x)}_{x=x}$  ✓

$S_2: R_2(x) R_2(y) W_2(x) R_1(x) R_1(y)$

$S_3: R_2(x) R_2(y) R_3(x) R_2(y) W_2(x) W_2(y)$

which of the above schedules are having un-repeatable Read problem?

$\Rightarrow$  Un Repeatable Read problem means if a transaction tries to read the same data two times and in b/w this two reads if some other transaction changes the value then it is called Un-Repeatable Read.

$\Rightarrow$  Now  $S_1: R_2(x) \dots W_1(x) R_2(x)$

$x=x$

$x = \text{NOT } x$

modified the value of  $x$

which of the above schedules is having lost update problem.

$\Rightarrow$  Lost update problem means one transaction will write a value and immediately another transaction writes a value without reading it (one writes over

$S_1: \text{Only one write is there so no loss update}$

$S_2: \dots \dots \dots \dots \dots \dots \dots$

$S_3: W_1(x) W_2(x)$

one write ( $W_2$ ) overwrites the other

Write: Lost update problem

which of the  
⇒ Dirty Read  
someone else

$S_2: R_2(x)$

$S_1: W_1(x)$

which of the  
strict

$S_1: R_1(x) R_2(x)$

T
RC
WC
COM

$S_2: R_1(x) W_2(x)$

T
R
W
R
C
O

$S_3: R_3(x) R_1(x)$

Recoverable: ✓

+

Cascade less

128

which of the above schedules is having dirty Read problem.

129

⇒ Dirty Read problem is if someone will write it and before writing someone else will read it this is called Dirty Read problem.

schedules  
one possible?

reactions  
 $T_2(T_1 T_2)$

$S_2: R_1(x) \dots w_2(x)(R_1(x))$

$T_2$  is waiting the value of  $x$  and  $R_1$  is reading it before it got committed. so it is dirty Read.

$S_1: w_1(x) R_2(x)$  ← Dirty Read problem.

Which of the following schedules are Cascadeless, Recoverable and strict.

$S_1: R_1(x) R_2(x) w_1(x) w_2(x) \text{ commit}(c_2) \text{ commit}(c_1)$

Read

some  
repeatable Read

$T_1$	$T_2$
$R(x)$	
	$R(x)$
$w(x)$	
	$w(x)$ Commit
commit	

⇒ Recoverable : consumer should not commit before the producer commit

⇒ There are no producer and consumer ..

The schedule is Recoverable, cascadeless

⇒ Strict says if some one writes a value you should not Read/write it until the previous one commits ∴ Not strict schedule

5c

$S_2: R_1(x) w_2(x) w_1(x) R_1(x) \text{ Commit}(c_2) \text{ Commit}(c_1)$

problem.

write a value

without

update

$T_1$	$T_2$
$R(x)$	
	$w(x)$
$w(x)$	
$R(x)$	
	Commit
Commit	

⇒ No producer consumer ⇒ cascadeless,

⇒ Not strict schedule

$T_1$	$T_2$	$T_3$
		$R_3(x)$
$R_1(x)$		
	$R_2(x)$	
$w(y)$		Producer - Consumer
	$R_2(y)$	
	$R_2(x)$	
Commit		Commit

$S_3: R_3(x) R_1(x) R_2(x) w_1(y) R_2(y) w_2(z) c_3 c_2$

Recoverable : says that producer should commit before the consumer ∴ Recoverable ( $T_1$  commits before  $T_2$ )

Cascadeless : says that producer should commit first then only you have to read it. NOT CASCADELESS ∴ NOT STRICT

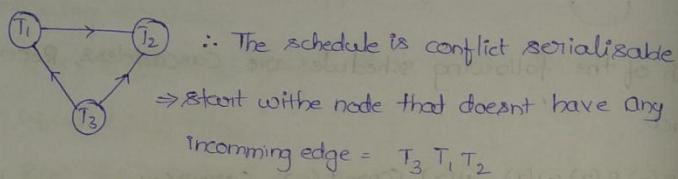
### 13. EXAMPLES ON CONFLICT SERIALIZABLES

130

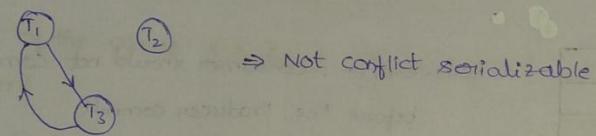
• which of the following schedules is conflict serialisable? For each serializable schedule, determine the equivalent serial schedules?

- $r_1(x), r_3(x), w_1(x), r_2(x), w_2(x)$
- $r_1(x), r_3(x), w_2(x), w_1(x), r_2(x)$
- $r_3(x), r_2(x), w_3(x), r_1(x), w_1(x)$
- $r_3(x), r_2(x), r_1(x), w_3(x), w_1(x)$

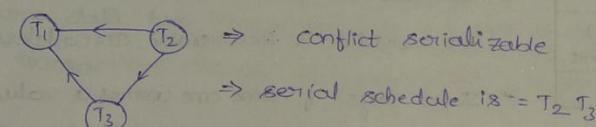
Now, i)



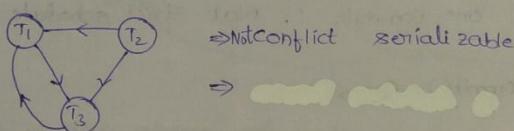
ii)



iii)



iv)



### 14. EXAMPLES ON CONFLICTS

Given

$T_1: R(x) \ R(y) \ W(x)$

$T_2: R(x) \ R(y) \ W(x) \ W(y)$

} Now form the schedules that result in  
WR-conflict, RW-conflict, WW-conflict

WR

$T_1$	$T_2$
$R(x)$	
$R(y)$	
$w(x)$	
	$R(x)$
	$R(y)$
	$w(x)$
	$w(y)$

RW

$T_1$	$T_2$
$R(x)$	
	$R(x)$
	$R(y)$
	$w(x)$
	$w(y)$

WW

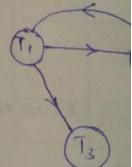
$T_1$	$T_2$
$R(x)$	
$R(y)$	
$w(x)$	
	$w(x)$
	$w(y)$
	$R(x)$
	$R(y)$

### 15. POLY GR

$T_K$	$T_1$		
		$w(x)$	$R(x)$

$S: r_1(A) \ w_2$

NOTE: If  
is view &  
→ If  
write  
not



Now, Insert two  
trans actions  $T_0$

$T_0$	$T_1$	$T_2$	$T_3$	
$w(x)$				
	$R(x)$			
		$w(x)$		
			$w(x)$	
				$R(x)$
				$w(y)$

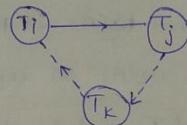
## 15. POLYGRAPHS FOR VIEW SERIALIZABLE -- EXAMPLE 1

each

des?

TK	T <sub>i</sub>	T <sub>j</sub>
	w(A)	
	R(A)	

⇒ Suppose there are two transactions T<sub>i</sub> and T<sub>j</sub> at some point of time T<sub>i</sub> performs write operation and then T<sub>j</sub> reads it. Now another Transaction T<sub>k</sub> executes write operation the it should be before T<sub>i</sub> and before T<sub>j</sub> so the polygraph will be

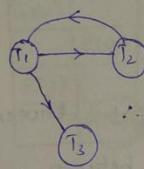


S : r<sub>1</sub>(A) w<sub>2</sub>(A) w<sub>1</sub>(A) w<sub>3</sub>(A)

NOTE: ⇒ If a schedule is conflict serializable then the schedule is view serializable, so the 1st check for VS should be CS.

⇒ If the schedule is not CS, now check if there are Blind writes or not, in case if there are Blind writes and it is not CS then only we should check for VS.

CS X	CSX	CSV	CS = Conflict serializable
BW X	BW ✓	VS ✓	BW = Blind writes
VS X	VS ✓		VS = View serializable



∴ Not CS X ... Now check for Blind writes (Write without Read)  
Transaction 2 is writing without Reading (·BW ✓)

Now, Insert two dummy transactions T<sub>b</sub> and T<sub>f</sub>

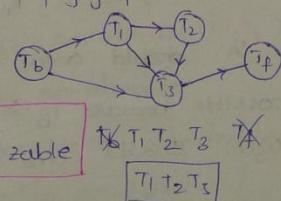
T <sub>b</sub>	T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>	T <sub>f</sub>
w(A)				
R(A)				
w(A)				
	w(A)			
		w(A)		
			R(A)	

CS X  
BW ✓  
VS ✓

No cycles in poly graph

The schedule is view serializable

Now draw poly graph



↑ T<sub>1</sub> T<sub>2</sub> T<sub>3</sub> ↑  
T<sub>b</sub> T<sub>f</sub>

We Assume

T<sub>b</sub> = writes data items initially  
T<sub>f</sub> = Read all data items

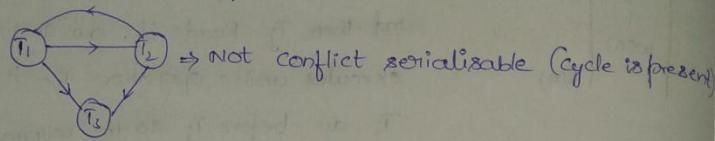
T<sub>b</sub>, T<sub>f</sub> are dummy transactions

## 16. POLY GRAPHS FOR VIEW SERIALISABLE EXAMPLE 2

152

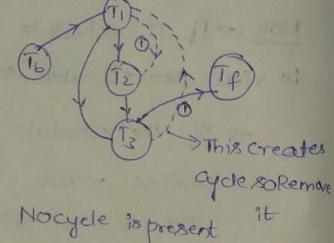
$S = T_1(A) \quad W_2(A) \quad T_3(A) \quad W_1(A) \quad W_3(A)$   $\Rightarrow$  find if it is view serialisable / not

$\Rightarrow$  First check whether it is conflict serialisable / not



$\Rightarrow$  Now, check for Blind writes.  $W_2(A)$  is the Blind write because it is not reading 'A' before  $W_2(A)$  so  $W_2(A)$  is Blind write  $\Rightarrow$  Now check for VS by Adding ( $T_b$  and  $T_f$  as dummy Transactions)

$T_b$	$T_1$	$T_2$	$T_3$	$T_f$
$w(A)$				
	$R(A)$			
		$w(A)$		
			$R(A)$	
			$w(A)$	
				$R(A)$



$\therefore$  The schedule is view serializable and the serial schedule is

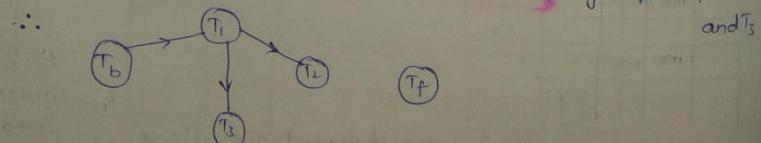
$$\cancel{T_1 T_2 T_3} = T_1 T_2 T_3$$

$\Rightarrow$  Procedure to draw the above poly graph

i) check for write and Read 's of the same object between two different transactions. The first WR is between

$\Rightarrow w_b(A) \quad R_1(A)$  Now, all the writes of the data item

'A' should occur before  $T_b$  or after  $T_1$ , now before  $T_b$  is not possible because  $T_b$  is the 1st transaction  $\therefore$  The write  $w_2(A)$ ,  $w_3(A)$  should come after  $T_1$ , and therefore there will be edges from  $T_1$  to  $T_2$  and  $T_3$



## 17. Example

$T_1$
$R(x)$
$w(x)$

$T_1$
$w(x)$
$R(y)$

$T_1$
$R(x)$

152

ble (not

e is present)

use it

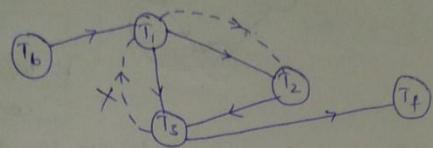
$\Rightarrow$  Now

actions)

(R/W : 2

$\Rightarrow$  Now again check for write Reads we find WR in  $w_2(A), R_3(A)$  so all the writes of 'A' should occur before  $T_2$  and after  $T_3$ .

153



$\Rightarrow$  Now, one write is before  $T_2$  i.e.  $w_1(A)$  so there exists an edge from  $T_1$  to  $T_2$  and  $w_1(A)$  occurs between  $T_3$ .  $\therefore$  there exists an edge from  $T_3$  to  $T_1$ . Since it forms cycle Remove it

$\Rightarrow$  Next WR is between  $T_3$  and  $T_4$  so there exists an edge from  $T_3$  to  $T_4$

## II. EXAMPLES ON VERIFYING THE SCHEDULES - I

T1	T2
R(x)	
	R(x)
w(x)	
	w(x)

This creates cycle so Remove it

Schedule is

$\Rightarrow T_1 \rightarrow T_2 \Rightarrow$  Not conflict serializable  
 $\Rightarrow$  No blind write (Not vs)  
 $\Rightarrow$  Recoverable ✓  
 $\Rightarrow$  Cascading schedule  
 $\Rightarrow$  Not strict  
 $\Rightarrow$  No (producer-consumer) (Read after writes)  
 $\Rightarrow w_2(x)$  should be done after  $w_1(x)$  commits

T1	T2
w(x)	
	R(y)
R(y)	
	R(x)

between

data items

where  $T_2$  is not write  $w_2(A)$ , from  $T_1$  to  $T_2$  and  $T_3$

$\Rightarrow T_1 \rightarrow T_2 \Rightarrow$  Conflict serializable  
 $\Rightarrow$  Recoverable cannot be said (No commit)  $\Rightarrow$  Strict  
 $\Rightarrow$  cascading schedule

Dirty Read.

T1	T2	T3
R(x)		
	R(y)	
R(y)		
	R(x)	

producer-consumer

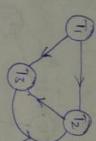
$\Rightarrow$  conflict serializable ( $T_1 T_2$ )  
 $\Rightarrow$  view serializable  
 $\Rightarrow$  Recoverable cannot be decided  
 $\Rightarrow$  cascading schedule  
 $\Rightarrow$  Not strict.

### 18. EXAMPLES ON VERIFYING THE SCHEDES - 2

45

	$T_1$	$T_2$	$T_3$
$R(x)$			
$R(y)$			
$w(x)$			
$R(q)$			
$(w(x))$			

Now check if it is vs by drawing polygons



Not conflict  
serially

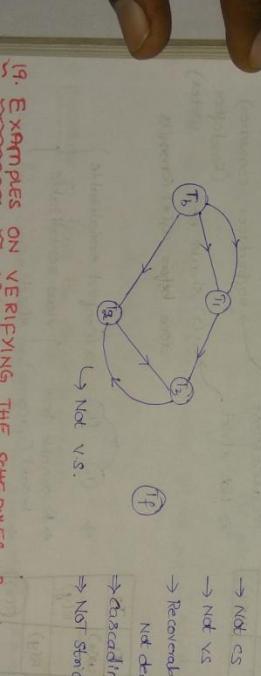
$\overline{I_b}$	$T_1$	$T_2$	$T_3$	$T_f$
$w(x)$				
$w(y)$				
$R(x)$				
$R(y)$				
$w(x)$				
$R(y)$				
$R(x)$				
$R(y)$				

$\omega(x)$	$T'$

20. EXPECTS

	$T_1$
$\omega(x)$	$\omega(x)$
$(\omega x)$	$\omega(\omega x)$
$\omega$	$\omega \omega$

→ Recoverability  
Not deductible



## 19. EXAMPLES ON VERIFYING THE STATEMENTS - 3

$\Rightarrow$  when a Transaction ends

that means : should Roll back ~~most~~ most then

$\Rightarrow$  when a transaction says that then

there is only one transaction and hence it

⇒ Revocable (No consideration)

$\Rightarrow$  Cascades (Independent  
 Not Strict  $\omega_2(x)$  has not Read  $x$ )  
 (i.e. Commit on  $\{x\}$ )

$T_1$	$T_2$
$R(x)$	
	$w(x)$
$w(x)$	
Abort	
	<u>Roll Back</u>

⇒ when a Transaction says Abort  
that means : should Roll back  
⇒ when a transaction says Abort  
there is only one transaction  
hence it will be CS

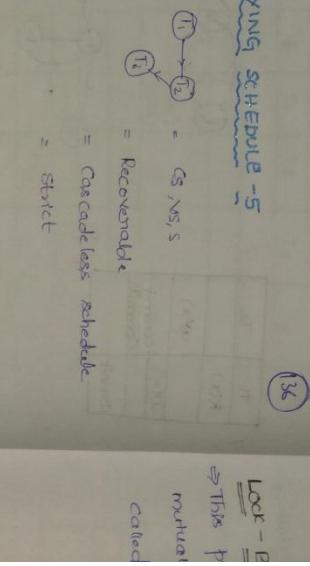
before T  
another  
 $T_b$  edge  
will be preser-



	$T_1$	$T_2$
$w(\alpha)$		
$w(\gamma)$		
$R(\alpha)$		
comment		
Abort		

- ⇒ when  $T_1$  aborts there will be Roll back and there will be only one transaction left.
- ∴ The schedule is conflict serializable
- ⇒ VS
- <sup>(1)</sup> Recoverable - X
- <sup>(2)</sup> cascading - ✓ (NOT)

## 2. EXAMPLES ON VERIFYING SCHEDULE -5

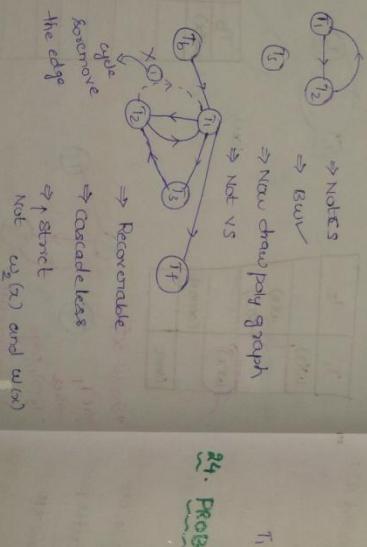


Cascadeless scheduled

Lock - B  
⇒ This p  
method  
called

$\therefore$  cascading - ✓ (NOT  
cascadeless)

23. INTI  
The C



24.

Q3. INTRODUCTION TO LOCKING

(13+)

The concurrency manager present in the OS make sure that the transactions that are being executed are serializable, by using some rules they are:

- ⇒ lock-based protocol
- ⇒ Time stamp based protocols
- ⇒ Graph based protocols.

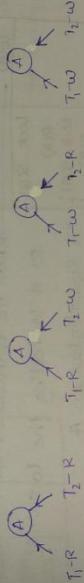
### Lock-Based Protocol

This protocol says that all the data items should be accessed in a mutually exclusive manner, to achieve this we use two locks called:

- ⇒ Shared lock (Only Read) [lock-S]
- 2> Exclusive lock (Both Read and write) [lock-X]

$\begin{array}{|c|c|} \hline S & X \\ \hline S & S \\ \hline X & X \\ \hline \end{array}$

⇒ If a transaction has shared lock on particular data item then another transaction may also acquire the shared lock on the same data item & share on share is allowed.



### PROBLEMS WITH LOCKING

T<sub>1</sub>: A 50 → B

T<sub>2</sub>: Display (B+n)

Read(A)  
Lock-SCB  
Read(B)  
A = A - 50  
Write(A)  
Unlock(A)  
Lock(X)(B)  
Read(A)  
Unlock(A)  
Display(B+n)

Now if it is only reading  
then after shared lock  
→ Now if the Transaction  
are interleaved (concurrent  
execution) then there  
occur some conflicts.

Read(B)  
B = B + 50  
Write(B)  
Unlock(B)

13

$T_1$	$T_2$
locked X(B)	
Read(A)	
A = A - 50	
write(A)	
unlock(A)	
lock - S(C)	
Read(C)	
unlock(C)	
lock - S(A)	
Read(A)	
unlock(A)	
Display(C+A)	

$$\frac{100}{50} + \frac{200}{200} = \frac{100}{100} = 1$$

Because of concurrent execution it should be 200.

$\Rightarrow$  Therefore Simple locking protocol doesn't give appropriate Result.

$T_1$	$T_2$
lock - X(A)	
B = B + 50	
write(B)	
unlock(C)	

$T_1$	$T_2$
	lock - X(B)
	lock - S(A)
	lock - S(C)
	read(C)
	unlock(A)

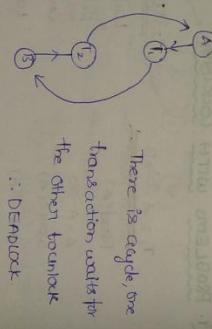
$T_1$	$T_2$
lock - X(A)	
R(A)	
A = A - 50	
W(A)	

→ Here the transaction has not unlocked A.  
So it is holding the lock on A.

→ The 2-phase readiness serial

$T_1$	$T_2$
L(X(A))	
L(S(C))	
L(S(A))	
lock - S(A)	
lock - S(C)	
lock - X(B)	
read(C)	
unlock(A)	

$T_1$	$T_2$
lock - X(B)	
lock - S(A)	
lock - S(C)	
lock - X(A)	
read(A)	
commit	
unlock(A)	
unlock(B)	



- Simple locking might produce Deadlock
- Simple locking might not guarantee Serializability

Which requires node locks  
and commits. (a)

LX  
WB  
COM  
UCX

14

25. Two Phase

Since the 2-phase lock!

Growing phase,  
locking ph

Shrinking ph

⇒ The point w

⇒ with the help

$T_1$	$T_2$
L(X(A))	
L(S(C))	
L(S(A))	
lock - S(A)	
lock - S(C)	
lock - X(B)	
read(C)	
unlock(A)	

$$\frac{100}{50} + \frac{200}{200} = \frac{100}{100} = 1$$

Because of concurrent execution it should be 200.

$\Rightarrow$  Therefore Simple locking protocol doesn't give appropriate Result.

$T_1$	$T_2$
lock - X(A)	
B = B + 50	
write(B)	
unlock(C)	

$T_1$	$T_2$
lock - X(B)	
lock - S(A)	
lock - S(C)	
lock - X(A)	
read(C)	
unlock(A)	

$T_1$	$T_2$
lock - X(A)	
R(A)	
A = A - 50	
W(A)	

$T_1$	$T_2$
lock - X(B)	
lock - S(A)	
lock - S(C)	
lock - X(A)	
read(C)	
unlock(A)	

$T_1$	$T_2$
lock - X(B)	
lock - S(A)	
lock - S(C)	
lock - X(A)	
read(C)	
unlock(A)	

$T_1$	$T_2$
lock - X(B)	
lock - S(A)	
lock - S(C)	
lock - X(A)	
read(C)	
unlock(A)	

$T_1$	$T_2$
lock - X(B)	
lock - S(A)	
lock - S(C)	
lock - X(A)	
read(C)	
unlock(A)	

$T_1$	$T_2$
lock - X(B)	
lock - S(A)	
lock - S(C)	
lock - X(A)	
read(C)	
unlock(A)	

$T_1$	$T_2$
lock - X(B)	
lock - S(A)	
lock - S(C)	
lock - X(A)	
read(C)	
unlock(A)	

$T_1$	$T_2$
lock - X(B)	
lock - S(A)	
lock - S(C)	
lock - X(A)	
read(C)	
unlock(A)	

$T_1$	$T_2$
lock - X(B)	
lock - S(A)	
lock - S(C)	
lock - X(A)	
read(C)	
unlock(A)	

$T_1$	$T_2$
lock - X(B)	
lock - S(A)	
lock - S(C)	
lock - X(A)	
read(C)	
unlock(A)	

$T_1$	$T_2$
lock - X(B)	
lock - S(A)	
lock - S(C)	
lock - X(A)	
read(C)	
unlock(A)	

$T_1$	$T_2$
lock - X(B)	
lock - S(A)	
lock - S(C)	
lock - X(A)	
read(C)	
unlock(A)	

$T_1$	$T_2$
lock - X(B)	
lock - S(A)	
lock - S(C)	
lock - X(A)	
read(C)	
unlock(A)	

$T_1$	$T_2$
lock - X(B)	
lock - S(A)	
lock - S(C)	
lock - X(A)	
read(C)	
unlock(A)	

$T_1$	$T_2$
lock - X(B)	
lock - S(A)	
lock - S(C)	
lock - X(A)	
read(C)	
unlock(A)	

$T_1$	$T_2$
lock - X(B)	
lock - S(A)	
lock - S(C)	
lock - X(A)	
read(C)	
unlock(A)	

$T_1$	$T_2$
lock - X(B)	
lock - S(A)	
lock - S(C)	
lock - X(A)	
read(C)	
unlock(A)	

$T_1$	$T_2$
lock - X(B)	
lock - S(A)	
lock - S(C)	
lock - X(A)	
read(C)	
unlock(A)	

$T_1$	$T_2$





<tbl\_r

## 25. TWO PHASE LOCKING PROTOCOL

since the simple transaction has some disadvantages we come for 2-phase locking protocol. There are 2 phases in 2PL they are Growing phase, Shrinking phase

⇒ Growing phase - Obtain the locks } No one can come in middle  
 ⇒ Shrinking phase - Release all the locks. } and solve serializability in 2PL

⇒ The point where the transaction has the **final lock** is called **locking point**.

⇒ with the help of locking points we can find serial schedule

T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>
L·S(A)		
	L·S(A)	
L·X(B) U(A)		
	L·X(C)	
	U(B)	
L·S(B)		
		U(A) U(C)
L·X(A) U(A) U(B)		

→ locking point for T<sub>2</sub>

→ locking point for T<sub>3</sub> (point where the final lock appears)

∴ Serial schedule = [T<sub>2</sub> T<sub>3</sub> T<sub>1</sub>] = order in which we get locking points

sometimes

⇒ The 2-phase locking has [Cascading Rollbacks and Deadlocks] but it assures serializability.

## 26. STRICT RIGOROUS AND CONSERVATIVE 2PL

Some modifications are made for the above 2PL so that it can prevent Cascading Rollbacks

### Strict 2PL

Which requires that in addition to locking being 2PL all exclusive mode locks taken by a transaction must be held until the transaction commits. (unlock only after a particular transaction commits).

L·X-(A)

W(A)

Commit

U(X) → unlock only after transaction commits.

Strict 2PL are

Cascadeless,

Recoverable,

Deadlock possible

Rigorous 2PL:

which requires that in addition to locking being 2-phase all locks must be held until the transaction commits.

⇒ can avoid cascading rollbacks.

⇒ deadlock cannot be avoided (deadlocks are possible because of hold and wait)

Conservative 2PL

⇒ which requires the transaction to update all the locks before it starts and release all the locks after it commits.

⇒ avoids cascading rollbacks and deadlocks.

## 27. Examples on 2PL

Which of the following schedules (Transactions) are in Strict 2PL?

a) Lock - SCA

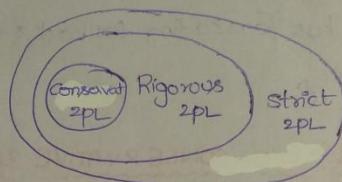
R(A)  
Lock - X(B)  
R(B)  
unlock(A)  
w(B)  
unlock(B)

⇒ There is growing phase and shrinking phase  
therefore it is 2PL

⇒ Now consider the exclusive locks, Lock - X(B)  
this should be unlocked only after 'B' commits  
But it is unlocked before 'B' committed ∵ NOT

got

STRICT 2PL



b) Lock - SCA

R(A)  
Lock - X(B)  
unlock(A)  
RC(B)  
w(B)  
Commit  
unlock(B)

⇒ There is growth and shrink ⇒ 2PL ✓

⇒ Consider Exclusive locks lock X-(B) it

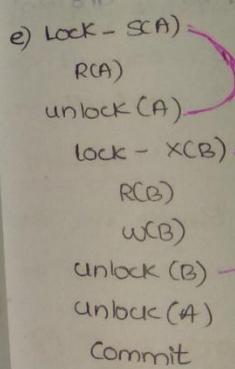
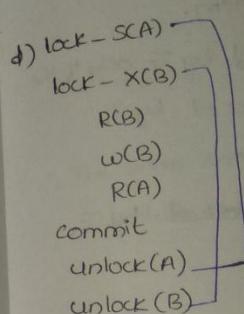
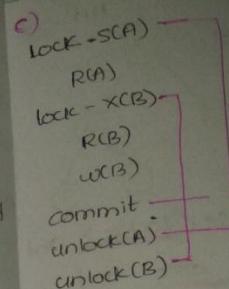
should be unlocked only after 'B' commits

lock done ∵ Strict 2PL

before Commit ⇒ NOT Rigorous

⇒ Rigorous says that either it is shared lock or exclusive lock they should be unlocked only after commit, here shared lock is released before commit

∴ NOT RIGOROUS 2PL



⇒ In the G

⇒ In the

140  
all locks

increase of hold

before it

strict 2PL?

phase

lock - X(CB)

B' commits  
ted ∴ NOT

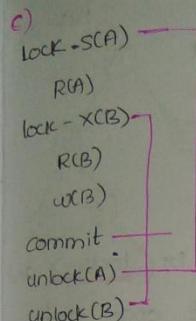
STRICT 2PL

2PL ✓

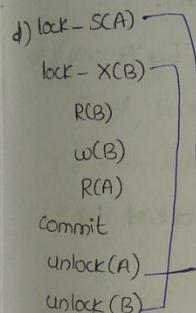
(B) it

B' commits

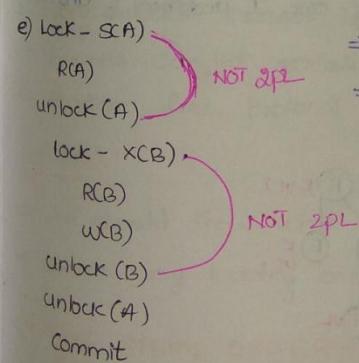
it is shared  
should be  
shared lock



⇒ Growth and shrink ⇒ 2PL  
⇒ unlock(B) is done after commit ∴ Strict 2PL  
⇒ unlock(A) is done after commit ∴ Rigorous 2PL  
⇒ Conservative 2PL says that you have to acquire the locks before starting any operation  
∴ The given schedule started R(A) before acquiring the lock on B  
∴ NOT conservative 2PL



⇒ Growth and shrink ⇒ 2PL  
⇒ unlock(B) done after commit = Strict 2PL  
⇒ Both A,B are unlocked after commit = Rigorous 2PL  
⇒ All the locks are acquired before proceeding the operations ⇒ conservative 2PL



⇒ 2PL X  
⇒ not strict 2PL ∵ unlock(B) done before commit

⇒ In the Growing phase locks are acquired and they can be upgraded  
⇒ shared lock → upgraded to Exclusive lock  
⇒ In the Shrinking phase locks are released and locks can be degraded  
⇒ Exclusive lock → degraded to shared lock

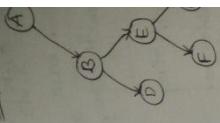
## 28. GRAPH BASED protocol

- ⇒ One of the alternative to 2PL is Graph Based protocol
- ⇒ we can avoid Deadlock

### Rules

⇒ In tree protocol the only lock instruction allowed is lock-X<sub>0j</sub>, each transaction  $T_i$  can lock a data item almost once and must observe the following rules.

- i) The 1st lock by  $T_i$  may be on any data item.
- ii) Subsequently a data item can be locked by  $T_i$  only if the parent of the data item is currently locked by  $T_i$ .
- iii) Data items may be unlocked at any time.
- iv) A data item that has been locked and unlocked by  $T_i$  cannot subsequently be locked by  $T_i$ .



### Advantages

⇒ Now consider the transactions one-by-one  $T_1$  and then  $T_2$  and then  $T_3$  ⇒ Early unlock

⇒ Ensures deadlock  
( $T_1$  →  $T_2$ ) →  $T_3$ )

⇒ Deadlock  
( $T_2$  →  $T_3$ ) →  $T_1$ )

⇒ You stuck  
( $T_3$  →  $T_1$ ) →  $T_2$ )

⇒ Unnecessary  
( $T_1$  →  $T_3$ ) →  $T_2$ )

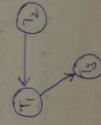
⇒ Time S

⇒ Time share  
order, replace all the locks with write statements}

⇒ If Req. in advance

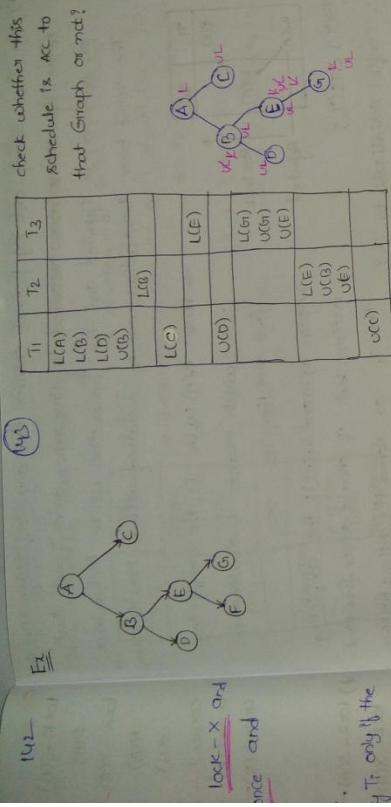
⇒ Each Transaction → If any → Then

$T_1$	$T_2$	$T_3$	⇒ Now consider the transactions one-by-one $T_1$ and then $T_2$ and then $T_3$ ⇒ Early unlock
lock-X(B)			lock-X(C) (lock-X(H)) unlock(D)
	lock-X(D)		lock-X(E) (lock-X(B)) unlock(C)
	unlock(B)		lock-X(E) (lock-X(D)) unlock(C)
	unlock(E)		lock-X(B) lock-X(E)
		unlock(A)	Now if you want to see serializable ⇒ Time share order, replace all the locks with write statements
		unlock(D)	⇒ If Req. in advance



152

三



check whether this  
schedule is acc to  
that Graph or not?

$T_1$	$T_2$	$T_3$
$L(A)$		
$L(B)$		
$L(D)$		
$U(E)$		
	$L(G)$	
		$L(F)$
$L(C)$		
	$L(F)$	
		$L(G)$
$U(D)$		
	$U(G)$	
		$U(E)$
	$L(F)$	
	$U(B)$	
		$U(E)$
$U(C)$		

The given schedule is according to the Graph given and general

Schedule is

T  
R

$$T_3 - T_1 - T_2$$

pick first

Emergency Manager to provide

Rules come always Serialised

than reaction is determined

time stamp denoted by

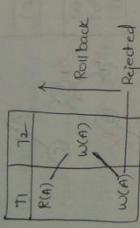
that the product

**Schedule must be** equivalent to a general schedule  $T_1 \rightarrow T_j$   
**Their  $\uparrow$  stamps** be  $TS(T_1) < TS(T_j)$  which means  $T_1$   $\uparrow$

### 29. TIME STAMP ORDERINGS PROTOCOL

- Time stamp protocol is used by the concurrency manager to provide concurrency and sees that the resulting schedules are always Serializable
  - At Requires that ordering among the transaction is determined in advance based on their time stamps.
  - Each Transaction is given unique fixed timestamp denoted by  $TS(T_i)$
  - If any transaction  $T_j$  that enters after  $T_i$  the relation between their timestamps be  $TS(T_1) < TS(T_j)$  which means that the producing schedule must be equivalent to a serial schedule  $T_1 \rightarrow T_j$

- 1) In Time Stamp ordering protocol ensures that any conflicting read or write operations are executed in time stamp order if not such an operation is rejected and the transaction will be rolled back. The rolled back transaction will be restarted with a new timestamp.



$\Rightarrow$  Here  $w_2(A)$  is the conflicting action which means  $T_1$  should occur first and then  $T_2$ .

$\Rightarrow$  Again  $w_2(A)$  and  $w_1(A)$  is the conflicting action but  $T_1$  should occurs first and  $T_2 < TS(T_1)$

$\therefore$  Conflict Serializable to  $T_1-T_2$  so Reject the action & Rollback

### 3. Thomas WRITE Rule

Thomas write Rule says that obsolete writes can be ignored. obsolete write means If the writes are late and before that some other writes has happened which is supposed to happen after it then you can clearly ignore the write and make the transaction play there before rolling back.

		Not Allowed		Allowed	
		$TS(T_2) < TS(T_1)$		$TS(T_2) \geq TS(T_1)$	
		$T_2 \rightarrow T_1$		$T_1 \rightarrow T_2$	
Time Stamp ordering protocol		$R_1(A)$ $w_2(A)$	$R_2(A)$ $R_1(A)$ $w_1(A)$	$R_2(A)$ $R_1(A)$ $w_2(A)$ $w_1(A)$	$R_1(A)$ $R_2(A)$ $w_1(A)$
Thomas write rule		$R_1(A)$ $w_2(A)$	$R_2(A)$ $R_1(A)$ $w_1(A)$	$R_2(A)$ $R_1(A)$ $w_2(A)$ $w_1(A)$	$R_1(A)$ $R_2(A)$ $w_1(A)$

$\Rightarrow$  In the above  $T_2 \rightarrow T_1$  then  $R_2(A)$  must be present  $\Rightarrow$  Hence  $w_2(A)$

$\therefore$  If  $T_1 \rightarrow T_2$  is the Time stamp order and there is  $R_1(A)$  is present

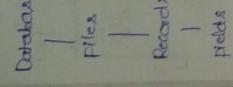
$\therefore$  Now, Thom has the cor

144

3. Examples on Time Stamp Ordering Protocol									
<p>read and such check which of the following schedules can appear under TOP.</p> <p>timestamping conflict occur conflicting first and Rollback</p> <p>affair transaction</p>									
1)	<table border="1"> <thead> <tr> <th>T<sub>1</sub></th> <th>T<sub>2</sub></th> </tr> </thead> <tbody> <tr> <td>R(B) B-B-BS W(B)</td> <td>R(B) B-B-BS W(B)</td> </tr> </tbody> </table> <p>⇒ Here T<sub>1</sub> is starting first <math>\Rightarrow TS(T_1) &lt; TS(T_2)</math></p> <p>⇒ There are 2 conflicts <math>\Rightarrow T_1</math> should come first acc to Time stamp.</p> <p>⇒ In Both the conflicts T<sub>1</sub> has occurred first before T<sub>2</sub> <math>\therefore</math> The schedule is according to top</p>	T <sub>1</sub>	T <sub>2</sub>	R(B) B-B-BS W(B)	R(B) B-B-BS W(B)				
T <sub>1</sub>	T <sub>2</sub>								
R(B) B-B-BS W(B)	R(B) B-B-BS W(B)								
2)	<table border="1"> <thead> <tr> <th>T<sub>1</sub></th> <th>T<sub>2</sub></th> </tr> </thead> <tbody> <tr> <td>R(A) A=A-50 W(A)</td> <td>R(A) A=A-50 W(A)</td> </tr> </tbody> </table> <p>⇒ TS(T<sub>1</sub>) &lt; TS(T<sub>2</sub>)</p> <p>⇒ ∴ NOT according to Top</p>	T <sub>1</sub>	T <sub>2</sub>	R(A) A=A-50 W(A)	R(A) A=A-50 W(A)				
T <sub>1</sub>	T <sub>2</sub>								
R(A) A=A-50 W(A)	R(A) A=A-50 W(A)								
3)	<table border="1"> <thead> <tr> <th>T<sub>1</sub></th> <th>T<sub>2</sub></th> <th>T<sub>3</sub></th> <th>T<sub>4</sub></th> </tr> </thead> <tbody> <tr> <td>W(A) A=A-50 W(A)</td> <td>W(X) X=X-10 W(X)</td> <td>W(X) X=X-10 W(X)</td> <td>R(X)</td> </tr> </tbody> </table> <p>SI: The above schedule is possible under Top</p> <p>Se: The above schedule is possible under Thomas Write Rule</p> <p>which of the above statements is true about the schedule given?</p> <p>⇒ In the above schedule T<sub>1</sub> arrived first and then followed by T<sub>2</sub>, T<sub>3</sub>, T<sub>4</sub>. <math>\therefore</math> The order in which we should serialise them is 1-2-3-4.</p> <p>⇒ Here W(A) R(X) <math>\Rightarrow</math> '3' should occur first and then '4' (Conflict present)</p> <p>This is not possible acc to Top</p> <p>Now, Thomas write rule saves us in write-write conflict but here the conflict is (W-R) so Not acc to Top</p>	T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>	T <sub>4</sub>	W(A) A=A-50 W(A)	W(X) X=X-10 W(X)	W(X) X=X-10 W(X)	R(X)
T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>	T <sub>4</sub>						
W(A) A=A-50 W(A)	W(X) X=X-10 W(X)	W(X) X=X-10 W(X)	R(X)						

## 8. FILE STRUCTURES

### 1. FILE ORGANISATION



Bloking Factor : It is the average no. of records per block

Strategies for storing file of records into block

① Spanned strategy : It allows partial part of Record Can be stored in a block

Adv: No wastage of memory / suitable for variable length record

Disadv: Block Access increases.

② Unspanned strategy

No Record can be stored in more than 1 block

Disadv: wastage of memory

Adv: Block Access reduced, suitable for fixed length records

Implementation of database system will use

③ Organisation of Records in a file : In this organisation all records are stored in sequential manner

i) ordered file organisation

All files of Records are ordered based on some search key value

Searching: Binary search R-blocks to access a record

Adv no of block access =  $\lceil \log_2(B) \rceil$  blocks

Adv: Searching is efficient

Dis: Insertion is expensive

### 2. INDEXING

↳ unordered file  
All the file  
usually at the

Searching: L

Arg. no. of b

Adv: Insertion

Disadv: Search

### 3. STRUCTURE

↳ Index

→ Indexes are also

→ Index is a tree

→ Index is an

Searching: D

Unordered file organisation

All the file records are inserted at where ever the place is available usually at the end of the file.

Searching: Linear search

$$\text{Avg. no of block access} = \log_2 \text{Blocks}$$

Adv: Insertion is easy

Disadv: Searching is inefficient.

## 2. INDEXING

Datafile

1	B1
61	10
20	
50	
12	70
80	
90	
63	100
400	

Index file

1	B1
10	B1
20	B1
50	
12	70
80	
90	
63	100
400	

1	B1
10	B1
20	B1
50	
12	70
80	
90	
63	100
400	

For searching in index file

$$\text{Avg. Block access} = (\log_2 B_1 + 1)$$

$B_1 = \text{size of Index file}$

(Binary search)

⇒ For every record in the datafile if we have a record/entry in the index file then that index is called Dense index

⇒ For every block in the datafile if we have a entry in the index file then that index is called Sparse index.

Ques

## 3. STRUCTURE OF INDEX FILES

Index:

⇒ Indexes are used to improve the search efficiency of data.

⇒ Index is a record that consists of two fields.

⇒ Index is an ordered file

⇒ Searching: Binary Search

Key | Block pointer



Ex. Example on Primary Indexing

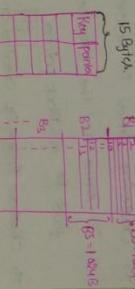
(Ans)

Suppose that we have an ordered file of 30,000 records on a disk with block size 1024 Bytes. File records are of fixed length and are un-spaced of size 1024 Bytes and suppose that we have created a primary index on the key field of the file of size 9 bytes and a block pointer of size 6 Bytes. Then find the avg no of blocks to search for a selected using with and without index?

Without index  
With index

$$\text{No. of Records} = 30,000$$

$$\text{Block size} = 1024 \text{ Bytes}$$



$$\text{Record size} = 1024 \text{ Bytes}$$

$$\text{Key + pointer} = 6 + 9 = 15 \text{ Bytes}$$

$$\Rightarrow \text{Data records that can fit in 1 Block} = \text{Block factor} = \frac{1024}{15} = [68]$$

$$= 10$$

: 10 records can be placed in one block

$$\Rightarrow \text{Total no of blocks} =$$

$$1 \text{ Block} \rightarrow 10 \text{ Records}$$

$$\propto \text{Blocks} = 30,000 \text{ Records}$$

$$\Rightarrow \frac{x}{10} = \frac{30,000}{10} = \boxed{x = 3000 \text{ blocks}}$$

where

$$\Rightarrow \text{No. of block access} = \lceil \log_2(3000) \rceil = 12 \text{ Block access}$$

With index  
Without index

$$\Rightarrow \text{Size of index Record} = 15 \text{ Bytes}$$

$$\text{No. of index Records per block} = \left\lceil \frac{1024}{15} \right\rceil = 68 \text{ Index Records.}$$

(a)

$$\Rightarrow \text{No. of Block Access} = \lceil \log_2(45) + 1 \rceil = \lceil \log_2(2^{5.5}) + 1 \rceil$$

$$= \lceil 5.5 \rceil = 7 \text{ blocks}$$

For file i go

Search in them

o Go to 80

o Now (97) 80

o Now (97) 80

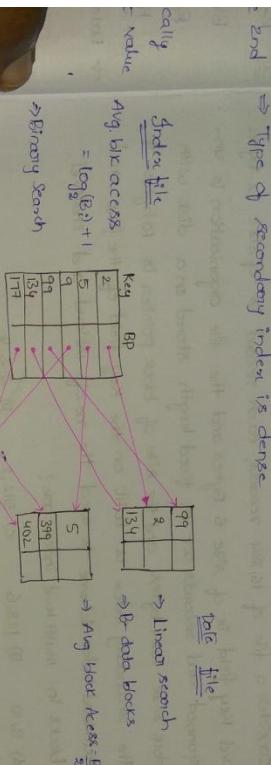
2



### 8. SECONDARY INDEXING ( $Nk/\log_k + \text{unordered}$ )

151

- ⇒ Index file should always be an ordered file
- ⇒ Secondary Index provides a secondary means of accessing file for which some primary access already exists.
- ⇒ Secondary Index maybe a key or non-key
- ⇒ Index entry is created for every record of datafile
- No. of index entries = No. of records.
- Type of secondary index is dense.



⇒ Binary Search

1029
399
402
99

Consider a secondary index is built on the key field of the file of previous question. The find. no. avg no. of block access to access a record using with and without index

- ⇒ R = 30,000 (unsorted)
- ⇒ No. of data Blocks =  $300,000/15 = 20,000$
- ⇒ No. of block access required without indexing =  $B/2 = 10,000$
- ⇒ Size of index record = 15 Bytes.
- ⇒ No. of index records =  $300,000/15 = 20,000$
- ⇒ No. of data records we can place in 1 block =  $\lceil \frac{1029}{15} \rceil = 68$
- ⇒ No. of blocks required =  $300,000/68 = 4416$
- ⇒ No. of blocks required =  $B/2 = \lceil \frac{\log_2(4416)}{15} \rceil + 1 = 10$
- ⇒ 4416 blocks =  $4416/10 = 441.6$  blocks
- ⇒ 441.6 blocks =  $441.6/15 = 29.44$  records
- ⇒ 29.44 records =  $29.44/15 = 1.96$  pages
- ⇒ 1.96 pages =  $1.96/15 = 0.13$  blocks
- ⇒ 0.13 blocks =  $0.13/15 = 0.008$  records
- ⇒ 0.008 records = 1 record

### Q. EXAMPLE ON MULTILEVEL INDEXING

17

As single level index is an ordered file we can create a primary index itself. In this case the original file is called 1st level index and the index to index is called 2nd level index.

⇒ If there are  $n$  levels in multilevel index then no. of block access to search for a record =  $n+1$  (One blk at each level and 1 blk for final)

Consider a file of 16,384 records, each record is of size 32-bytes and key field is of size 6 Bytes and the file organization is unspanned and records are of fixed length stored on a disk with block size 1024 bytes and the size of block pointers is 10 bytes. If the secondary index is built on the key field of the file, and multilevel index scheme is used the no. of 1st level and 2nd level blocks in multilevel index are?

- A) 8,10      B) 128,6      C) 512,2      D) 256,4

$$\text{Data records} = 16,384 = 2^{14}$$

$$\text{Record size} = 32B \sim 2^5 B$$

$$\text{Block size} = 1024B = 2^{10} B$$

$$\text{Key field} = 6B, \text{ bp} = 10B$$

$$= \text{Index Record size} = \text{Key} + \text{Block pointers} = 16B$$

⇒ 1st level Index

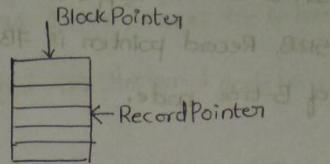
$$\text{No. of Index Records} = \text{No. of Data Records} / \text{No. of index Records}$$

$$\therefore \text{No. of blocks} = \frac{2^{14}}{2^6} = 2^8 = 256$$



## 10. INTRODUCTION TO BTREES AND B<sup>+</sup> TREES

- ⇒ BTree and B<sup>+</sup> Trees are Dynamically balanced.
- ⇒ Node pointer and Block pointer
- ⇒ Record pointer
- ⇒ All the databases today use B<sup>+</sup> Trees.
- ⇒ The advantage of B<sup>+</sup> Trees is they grow vertically

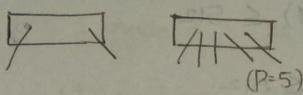


## II. PROPERTIES OF BTREES

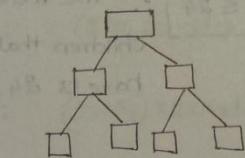
1) Root :- Between 2 and 'P' children where ( $P = \text{Order of the Tree}$ )

(Order is the max amount of children that can present to a particular node.)

particular node.



2) Internal nodes:

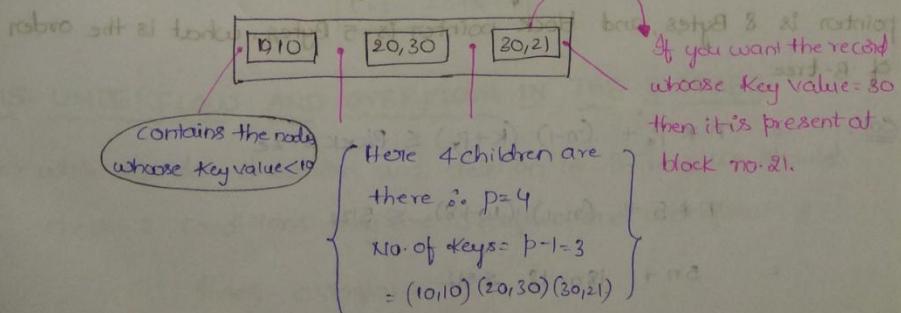


⇒ Internal nodes can store upto

$(P-1)$  keys.

⇒ Have between  $\lceil \frac{P}{2} \rceil$  and  $P$  children

The general representation is  $\langle P, \langle K_1, P_1 \rangle, P_2 \langle K_2, P_2 \rangle, \dots, P_q \langle K_q, P_q \rangle \rangle$



3) Leaf node : - stores between  $\lceil \frac{(P-1)}{2} \rceil$  and  $P-1$  keys

- All nodes have same depth.  
leaf

## 12. EXAMPLE ON BTREE

In a BTree, suppose search key is 9 Bytes long, disk block size is 512B, Record pointer is 7B, Block pointer is 6B, then calculate the order of B-tree node.

⇒ Every node in BTree is a disk block

$$P(k, p_r) P(k, p_r) \dots P$$

⇒ The formula that every node should satisfy is

$$= m * p + (m-1)(k+p_r) \leq \text{Block size}$$

$m$  = Order of the tree

$$= m * 6 + (m-1)(9+7) \leq 512$$

$p$  = Block pointer size

$$= 6m + 16m - 16 \leq 512$$

$p_r$  = Record pointer

$$= 22m \leq 528 \Rightarrow [m \leq 24] \therefore \text{The maximum amount of children that this tree can have is } 24.$$

At level 1

At level 2

At level 3

At level 4

## 13. EXAMPLE ON BTREE - 2

Consider a B-tree with Key size 10 Bytes, Block size 512B, data pointer is 8 Bytes, and block pointer is 5 Bytes. what is the order of B-tree

$$\text{Sol: } = m * p + (m-1)(k+p_r) \leq \text{Block size}$$

$$= m * 5 + (m-1)(10+8) \leq 512$$

$$= 5m + 18m - 18 \leq 512$$

$$= 23m \leq 540$$

$$= m \leq [23.478]$$

$$= m = 23$$

## 15. UNDER

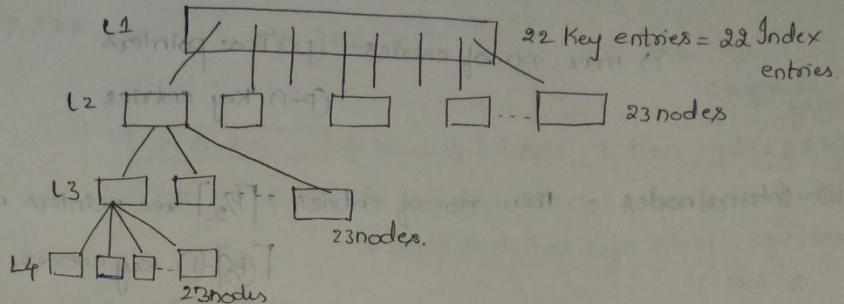
⇒ while w

check :

#### 14. EXAMPLE IN BTREE - 3

Suppose that the order of Btree is 23. Then how many index records will be stored in 4 level (including root as 1 level) across the B-tree.

$\Rightarrow n = 23$  (max amount of children that an Internal node have)



At level 1 : 1 node 22 Index Records 23 disk/node pointers

At level 2 : (23) nodes  $23 \times 22$  Index Records  $23 \times 23$  node pts.

At level 3 :  $(23) \times 23$  nodes  $(23 \times 23) \times 22$  IR  $(23 \times 23) \times 23$  NP

At level 4 :  $(23 \times 23) \times 23$  nodes  $(23 \times 23 \times 23) \times 22$  IR X(leaf node)

$$\text{No. of IR} = 23 \times 23 \times 23 \times 22$$

$$\boxed{\text{IRs} = 267674}$$

#### 15. UNDERFLOW AND OVERFLOW IN THE B-TREES

$\Rightarrow$  while we do insertion and deletion in B-Trees we should check 2 conditions they are overflow and underflow

Insert - overflow

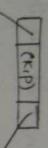
Delete - Underflow.

Order of B-Tree = p then,

Min:

Max: 4

Keys: 2

i) Root node  $\Rightarrow$  min no. of entries = 2 Tree pointers  
1 Key entry  


ii) Internal nodes = min no. of entries =  $\lceil \frac{p}{2} \rceil$  Tree pointers  
and  
(p-1) Key entries

iii) Leaf nodes = min no. of key entries =  $\lceil \frac{p}{2} \rceil - 1$  Key entries  
= max no. of key entries =  $p$  Tree entries  
(p-1) Key entries

iv) Leaf nodes = min no. of key entries =  $\lceil \frac{p}{2} \rceil - 1$  Key entries  
max no. of key entries = (p-1).

p=5  
i) Root = min: 2TP  
1KE  
max: 5TP  
4KE

ii) Internal nodes = min:  $\lceil \frac{p}{2} \rceil$  TP =  $\lceil \frac{5}{2} \rceil$  = 3TP  
2KE  
max: 5TP  
4KE

iii) Leaf nodes = min:  $\lceil \frac{p}{2} \rceil - 1$ , max: 4 (which is 5 when p=5)  
2KE  
max: 4 KE

$\therefore$  Final

### 6. INSERTION INTO BTREES -- Example -1

Insert the following Keys in B-tree of order - 4.

Keys: 2, 5, 10, 1, 6, 9, 4, 3, 12, 18, 20, 25.

Min: 2TP, 1KE

2TP, 1KE

2TP, 4KE

4TP, 3KE

6TP, 3KE

8TP, 4KE

10TP, 5KE

12TP, 6KE

14TP, 7KE

16TP, 8KE

18TP, 9KE

20TP, 10KE

22TP, 11KE

24TP, 12KE

26TP, 13KE

28TP, 14KE

30TP, 15KE

32TP, 16KE

34TP, 17KE

36TP, 18KE

38TP, 19KE

40TP, 20KE

42TP, 21KE

44TP, 22KE

46TP, 23KE

48TP, 24KE

50TP, 25KE

52TP, 26KE

54TP, 27KE

56TP, 28KE

58TP, 29KE

60TP, 30KE

62TP, 31KE

64TP, 32KE

66TP, 33KE

68TP, 34KE

70TP, 35KE

72TP, 36KE

74TP, 37KE

76TP, 38KE

78TP, 39KE

80TP, 40KE

82TP, 41KE

84TP, 42KE

86TP, 43KE

88TP, 44KE

90TP, 45KE

92TP, 46KE

94TP, 47KE

96TP, 48KE

98TP, 49KE

100TP, 50KE

102TP, 51KE

104TP, 52KE

106TP, 53KE

108TP, 54KE

110TP, 55KE

112TP, 56KE

114TP, 57KE

116TP, 58KE

118TP, 59KE

120TP, 60KE

122TP, 61KE

124TP, 62KE

126TP, 63KE

128TP, 64KE

130TP, 65KE

132TP, 66KE

134TP, 67KE

136TP, 68KE

138TP, 69KE

140TP, 70KE

142TP, 71KE

144TP, 72KE

146TP, 73KE

148TP, 74KE

150TP, 75KE

152TP, 76KE

154TP, 77KE

156TP, 78KE

158TP, 79KE

160TP, 80KE

162TP, 81KE

164TP, 82KE

166TP, 83KE

168TP, 84KE

170TP, 85KE

172TP, 86KE

174TP, 87KE

176TP, 88KE

178TP, 89KE

180TP, 90KE

182TP, 91KE

184TP, 92KE

186TP, 93KE

188TP, 94KE

190TP, 95KE

192TP, 96KE

194TP, 97KE

196TP, 98KE

198TP, 99KE

200TP, 100KE

202TP, 101KE

204TP, 102KE

206TP, 103KE

208TP, 104KE

210TP, 105KE

212TP, 106KE

214TP, 107KE

216TP, 108KE

218TP, 109KE

220TP, 110KE

222TP, 111KE

224TP, 112KE

226TP, 113KE

228TP, 114KE

230TP, 115KE

232TP, 116KE

234TP, 117KE

236TP, 118KE

238TP, 119KE

240TP, 120KE

242TP, 121KE

244TP, 122KE

246TP, 123KE

248TP, 124KE

250TP, 125KE

252TP, 126KE

254TP, 127KE

256TP, 128KE

258TP, 129KE

260TP, 130KE

262TP, 131KE

264TP, 132KE

266TP, 133KE

268TP, 134KE

270TP, 135KE

272TP, 136KE

274TP, 137KE

276TP, 138KE

278TP, 139KE

280TP, 140KE

282TP, 141KE

284TP, 142KE

286TP, 143KE

288TP, 144KE

290TP, 145KE

292TP, 146KE

294TP, 147KE

296TP, 148KE

298TP, 149KE

300TP, 150KE

302TP, 151KE

304TP, 152KE

306TP, 153KE

308TP, 154KE

310TP, 155KE

312TP, 156KE

314TP, 157KE

316TP, 158KE

318TP, 159KE

320TP, 160KE

322TP, 161KE

324TP, 162KE

326TP, 163KE

328TP, 164KE

330TP, 165KE

332TP, 166KE

334TP, 167KE

336TP, 168KE

338TP, 169KE

340TP, 170KE

342TP, 171KE

344TP, 172KE

346TP, 173KE

348TP, 174KE

350TP, 175KE

352TP, 176KE

354TP, 177KE

356TP, 178KE

358TP, 179KE

360TP, 180KE

362TP, 181KE

364TP, 182KE

366TP, 183KE

368TP, 184KE

370TP, 185KE

372TP, 186KE

374TP, 187KE

376TP, 188KE

378TP, 189KE

380TP, 190KE

382TP, 191KE

384TP, 192KE

386TP, 193KE

388TP, 194KE

390TP, 195KE

392TP, 196KE

394TP, 197KE

396TP, 198KE

398TP, 199KE

400TP, 200KE

402TP, 201KE

404TP, 202KE

406TP, 203KE

408TP, 204KE

410TP, 205KE

412TP, 206KE

414TP, 207KE

416TP, 208KE

418TP, 209KE

420TP, 210KE

422TP, 211KE

424TP, 212KE

426TP, 213KE

428TP, 214KE

430TP, 215KE

432TP, 216KE

434TP, 217KE

436TP, 218KE

438TP, 219KE

440TP, 220KE

442TP, 221KE

444TP, 222KE

446TP, 223KE

448TP, 224KE

450TP, 225KE

452TP, 226KE

454TP, 227KE

456TP, 228KE

458TP, 229KE

460TP, 230KE

462TP, 231KE

464TP, 232KE

466TP, 233KE

468TP, 234KE

470TP, 235KE

472TP, 236KE

474TP, 237KE

476TP, 238KE

478TP, 239KE

480TP, 240KE

482TP, 241KE

484TP, 242KE

486TP, 243KE

488TP, 244KE

490TP, 245KE

492TP, 246KE

494TP, 247KE

496TP, 248KE

498TP, 249KE

500TP, 250KE

502TP, 251KE

504TP, 252KE

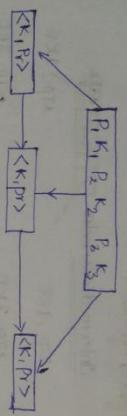
506TP, 253KE

508TP, 254KE

510TP, 255KE

## 2. B<sup>+</sup>-TREES STRUCTURE OF LEAF AND INTERNAL NODES

A records of length  $\frac{P}{B}$  are stored in each individual slot having size of  $\frac{P}{B} \times B$ . If each field  $P_i$  has size  $\frac{P}{B}$ .



⇒ The information needed to access the data is present in the leaves.

⇒ The intermediate nodes do not contain the pointers that points to data instead they guide to reach the leaves for which it points to the required data block.

⇒ The structure of all the Internal nodes and leaf nodes are different.

Internalnode :  $\langle P_1, K_1, P_2, K_2, \dots, K_{q-1}, P_q \rangle$

leaf node :  $\langle \langle k_1, p_1 \rangle, \langle k_2, p_2 \rangle, \dots, \langle k_{q-1}, p_{q-1} \rangle, P_{next} \rangle$

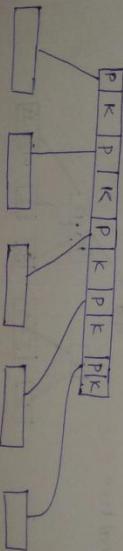
Let  $o$  be the order of leaf node

⇒ The min.no of children that an Internal node can have is  $\lceil \frac{o}{2} \rceil$  to  $o$  pointers.

⇒ min.no of key value pairs =  $\frac{o}{2}$ , max no. of key value pairs =  $o$ .

### 2.2. FINDING THE ORDER OF B<sup>+</sup> TREES

Search Key field is  $v = q_B$ , Blk size =  $512B$ , Record size =  $P_0 = 4B$ , Blk per =  $P = 6B$ . what is the order of internal node and leaf node?



⇒ At leaf1

⇒ leaf node

⇒ At Root

### 2.3. FIND

In a B<sup>+</sup>,

node is

many need

'0' and '1'

⇒ Given

ODES

854

$$= 0(6) + (6-1)q \quad \wedge \quad 5|2$$

P = Block pointer

卷之三

$$= 15(6) - 9 \leq 512$$

卷之三

leaf node)

$$S \leq \text{Bd}((k+p_r) + p) \leq \text{Bd}(k+p_r)$$

$$f_{\text{leaf}} \equiv (x+y) + 6 \leq 10$$

### FINDING THE CAPACITY OF A BTREE

In a B<sup>+</sup> tree, order of internal nodes is 34 and order of the leaf node is 31. If all the nodes of the tree are 60% full then how many nodes are there in the tree?

many records will be present in 4-level BT tree with root at level 'o' and leaf 'o' and leaves at level-3.

$$\Rightarrow \text{Inter node} = 69\% \text{ full} = \text{The no of children} = \frac{69}{100} \times 24 = 16.56 \approx 17 \text{ keys}$$

$\Rightarrow$  At Root  
 $1 \text{ node} = 82 \text{ KE} = 83 \text{ pointers}$ .  
 $\Rightarrow$  At level 1  
 $83 \times 82 = 506 \text{ KE}$  | At L2: 529 nodes  
 $83 \times 83 = 539 \text{ pointers}$ . |  $(529 \times 2) \text{ KE} = 1058 \text{ KE}$   
No. of ptns =  $539 \times 23 = 12401$

Q: 12,167 nodes

$12,167 \times 21 = 255,507$  (index records that are here

is able to store) (160)

### Temp points

▷ Secondary index is built on non-ordering field. Hence it will consists of attributes then the no. of secondary indices is  $2^{n-1}$

⇒ clustered Index = Nonkey + ordered  $\Rightarrow$  clustering index is used when we have clusters of similar database entries for particular attribute.

so indexing is done by ordering all the instances of attributes and clustering the same values, so index entry for only the first item is needed only.

This attribute has Repeated values so it is non-key.

Non-trivial  
FD =  $X \rightarrow Y$

trivial (2)

(2)

(3)

RELATIONSHIPS

are not  
because  
might violate

B X  
C X  
A X  
B X  
A ? Holds but  
C { Not sure

$\rightarrow X$  ✓  
 $\rightarrow Z$  X  
 $\rightarrow Y$  X  
 $\rightarrow Z$  ✓  
 $\rightarrow Y$  X

#### 4. GATE 2000 QUESTION ON FDs

Given the following Relation Instance

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

which of the following functional dependencies are satisfied by the instance?

- (a)  $XY \rightarrow Z$  and  $Z \rightarrow Y$  ✓  
(b)  $Y2 \rightarrow X$  and  $Y \rightarrow Z$   
(c)  $Y2 \rightarrow X$  and  $X \rightarrow Z$  (d)  $XZ \rightarrow Y$  and  $Y \rightarrow X$

a)  $XY \rightarrow Z$  and  $Z \rightarrow Y$   
↓  
Holds

Not hold  
{3→5}  
{3→6}

b)  $YZ \rightarrow X$  and  $Y \rightarrow Z$   
↓  
Holds

Holds

c)  $Y2 \rightarrow X$  and  $X \rightarrow Z$   
Holds

Not hold  
{1-2}  
{1-3}

d)  $XZ \rightarrow Y$  and  $Y \rightarrow X$   
Not hold  
{1,3→5}  
{1,3→6}

#### 5. GATE 2002 QUESTION ON FDs

From the following instance of a relation schema  $R(ABC)$ , we can conclude that

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

- a) A functionally determines B and B functionally determines C  
⇒  $A \rightarrow B$  and  $B \rightarrow C$   
✓  
or  $B \rightarrow C$   
d)  $A \not\rightarrow B$  and  $B \not\rightarrow C$

①  $A \rightarrow B$       B  $\rightarrow C$   
↓  
holds      Not holds.

(b)  $A \rightarrow B$       B  $\not\rightarrow C$   
↓  
holds      holds.  
for this instance  
but fail when  
insertion.⇒  
NOT holds.

(c)  $A \not\rightarrow B$       B  $\rightarrow C$   
↓  
Not      holds

## 6. FORMAL DEFINITION OF FUNCTIONAL DEPENDENCY

(24)

	A	B
$t_1$		
$t_2$		
	If $t_1$ and $t_2$ then they must agree here	Agree here
	If $t_1$ and $t_2$ then they may disagree here	Agree or may not Agree here

	A	B
Agree	1 1	a a
DisAgree	2 3	b b

## 8. CLOSURE

Functional

Here,  $A^+$  =  
 $B^+$  =

(cont)

## 7. VARIOUS USAGES OF FD's

⇒ Identify Additional Functional Dependencies

⇒ Identifying Keys

⇒ Identifying Equivalences of FD's

⇒ finding minimal FD set.

In order to perform all the above activities we have 2 methods

1) Inference Rules ⇒ Error prone

2) Closure set of Attributes ⇒ Easy and less Error prone

### Inference Rules

a) Reflexive :  $A \rightarrow A$  if  $B \subseteq A$

b) Transitive :  $A \rightarrow B$  and  $B \rightarrow C$  then  $A \rightarrow C$

c) Decomposition:  $A \rightarrow BC$  then  $A \rightarrow B$ ,  $A \rightarrow C$

d) Augmentation:  $A \rightarrow B$  then  $AC \rightarrow BC$

e) Union :  $A \rightarrow B$  and  $A \rightarrow C$ , then  $A \rightarrow BC$

f) Composition:  $A \rightarrow B$  and  $C \rightarrow D$ , then  $AC \rightarrow BD$ .

## 9. GATE

The following

$AB \rightarrow CD$

$AF \rightarrow D$

$DE \rightarrow F$

$C \rightarrow G$

$F \rightarrow E$

$G \rightarrow A$

So) A) {C

B) {

C) {

D) {

## 10. DE

Given

The no

⇒ For

### 8. CLOSURE SET OF ATTRIBUTES

functional Dependencies:  $A \rightarrow B$

$B \rightarrow D$

$C \rightarrow DE$

$CD \rightarrow AB$

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

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

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

$D^+ = \{D\}$

$E^+ = \{E\}$

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

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

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

The closure set of a set 'A' is

denoted by  $A^+$  and closure set says

that what are the other attributes

that you can uniquely identify

using A

### 9. GATE 2006 QUESTION ON CLOSURE SET OF ATTRIBUTES

The following functional dependencies are given:

$AB \rightarrow CD$

which one of the following options is false?

$AF \rightarrow D$

$(AF)^+ = \{A, C, D, E, F, G\}$  YES  $(AF)^+ = \{A, C, D, E, F, G\}$

$DE \rightarrow FG$

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

$C \rightarrow G$

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

$F \rightarrow E$

$G \rightarrow A$

$$\text{Sol: } A) \{CF\}^+ = \{C, F, G, A, E, D\} = \{A, C, D, E, F, G\} = \text{TRUE}$$

$$B) \{BG\}^+ = \{B, G, A, C, D\} = \{A, B, C, D, G\} = \text{TRUE}$$

$$C) \{AF\}^+ = \{A, F, E, D\} = \text{FALSE}$$

$$D) \{AB\}^+ = \{A, B, C, D, G\} = \text{TRUE}$$

### 10. DETERMINING CANDIDATE KEYS

Given Relation R(A,B,C,D) and the FDs are  $A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A$

The no. of candidate keys that can be formed =  $2^{n-1}$  ( $n$  = no. of attributes in the relation)

$$\therefore \text{No. of CK's} = 2^{n-1}$$

$\Rightarrow$  For the above example No. of candidate keys that should be examined is  $2^{4-1} = 15$ , at maximum.

Now,  $R(ABCD)$  the CK of length 4 =  $(ABCD) = \textcircled{1}$

26

CK of length 3 =  $(BCD)(ABC)(ACD)(ABD) = \textcircled{4}$

CK of length 2 =  $(AB)(BC)(CD)(DA)(AC)(BD) = \textcircled{6}$

CK of length 1 =  $(A)(B)(C)(D) = \textcircled{4}$

15

Now, perform Bottom to Top Approach

Now, let's check if 'A' is candidate key  $\Rightarrow A^+ = \{A, B, C, D\}$

$\therefore A$  is candidate key

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

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

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

$\Rightarrow \{A, B, C, D\}$  are candidate keys

$\therefore \boxed{\text{No. of CK's} = 4}$

Note : If 'A' is a candidate key then anything that contains A like  $(AB)(CA)(AD)(ABC)(ABD)$  will not be CK's they will be SK's.

### II. GATE 1999 QUESTION ON CANDIDATE KEY

GATE - 99

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

Functional dependencies are  $(C \rightarrow F, F \rightarrow A, EC \rightarrow D, A \rightarrow B)$  then CK = ?

- A) CD    B) EC    C) AE    D) AC

Sol

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

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

$$\boxed{CK = EC}$$

$\{A, B, D, F$  are derivable from the RHS but there is no way to determine C, E so the only way to derive them is by having  $(CE)^+ = \{C, E\}$

$\therefore CE$

or check by opvm

12. GATE 20

Consider the functions  
 $\{K\} \rightarrow \{M\}$ ,

①  $\{E, F\}$

Sol Given F

Now,

Now, (E)

13. GATE

Consider a functional candidate

a) AE, BE

Sol The f

20) ④

26)

$\{XBD\} \rightarrow \{G\}$

27)

### 12. GATE 2014 QUESTION ON CANDIDATE KEYS

Consider the Relation Scheme  $R = (E, F, G, H, I, J, K, L, M, N)$  and the set of functional dependencies  $\{E, F\} \rightarrow \{G\}$ ,  $\{F\} \rightarrow \{I, J\}$ ,  $\{E, H\} \rightarrow \{K, L\} \rightarrow \{M\}$ ,  $\{K\} \rightarrow \{M\}$ ,  $\{L\} \rightarrow \{N\}$  on R. What is the key for R?

- ①  $\{E, F\}$  ②  $\{E, F, H\}$  ③  $\{E, F, H, I, K, L\}$  ④  $\{E\}$

Sol Given FDs =  $\{E, F\} \rightarrow \{G\} \Rightarrow (EF) \rightarrow (G)$

$\{F\} \rightarrow \{I, J\} \Rightarrow (F) \rightarrow (IJ)$

$\{E, H\} \rightarrow \{K, L\} \rightarrow \{M\} \Rightarrow (EH) \rightarrow (KL) \rightarrow (M)$

$\{K\} \rightarrow \{M\} \Rightarrow (K) \rightarrow (M)$

$\{L\} \rightarrow \{N\} \Rightarrow (L) \rightarrow (N)$

contains A

will be SKs.

$(E, F, H)^+ = (E, F, G, H, I, J, K, L, M, N)$  The attributes that

can be derived from this are

(G, I, J, K, L, M, N) and there is

noway i could get  $\underline{(EFH)}$

Now,  $(EFH)^+ = \{E, F, H, G, I, J, K, L, M, N\}$

$\therefore \{EFH\}$  is the candidate key. and  $\{F, H, K, L\}$  is

CK?

derivable from  
there is noway  
C, E so the  
derive them?  
 $(CE)^+ = \{F\}$

### 13. GATE 05 QUESTION ON CANDIDATE KEY

Consider a Relation Scheme  $R = (A, B, C, D, E, H)$  on which the following functional dependencies hold:  $\{A \rightarrow B, BC \rightarrow D, E \rightarrow C, D \rightarrow A\}$ . What are the candidate keys of R?

- ① AE, BE ② AE, BE, DE ③ AEH, BEH, BCH

$\therefore AEH, BEH, DEH$

Sol The functional dependencies =  $A \rightarrow B$   
 $BC \rightarrow D$   
 $E \rightarrow C$   
 $D \rightarrow A$  Candidate keys should

contain EH.

$\therefore (EH)^+ = (A, B, C, D, E, H)$  Now,  $(EH)^+ = (E, H, C)$

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

$\therefore (A^H)^+ = \{A, B, C, D, E, H\} \leftarrow$  8 SK's [Total 6 attributes already included ( $A^H$ ): Total keys 8]

$(B^H)^+ = \{A, B, C, D, E, H\} \leftarrow$  8 SK's

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

$(D^H)^+ = \{A, B, C, D, E, H\} \leftarrow$  8 SK's.

#### 14. Ques 2013 QUESTION ON CANDIDATE KEY

Relation R has 8 attributes ABCDEGH. Fields of R' contain only atomic values.  $F = \{CH \rightarrow G, A \rightarrow BC, B \rightarrow CFH, E \rightarrow H, F \rightarrow EG\}$  is set of functional dependencies so that  $F^+$  is exactly the set of FDs that hold for R. How many CKs does R have?

sol

CH  $\rightarrow$  G

A  $\rightarrow$  BC

B  $\rightarrow$  CFH

E  $\rightarrow$  H

F  $\rightarrow$  EG

$\Rightarrow$  Now,  $D^+ = \{D\}$

$\Rightarrow (AD)^+ = (ABCD EFGH) \leftarrow$   $\Rightarrow (FD)^+ = (ABCDEF GH)$

$\Rightarrow (BD)^+ = (BCDE FGH) \leftarrow$   $\Rightarrow (GD)^+ = (GHD)$

$\Rightarrow (CD)^+ = (CD) \times$   $\Rightarrow (HD)^+ = (HD) \times$

$\Rightarrow (ED)^+ = (ABCDE FGH) \leftarrow$

$\therefore$  No. of CK's possible are =  $4^8 = (AD)(BD)(CD)(ED)$

#### 15. EXAMPLES ON CANDIDATE KEYS - I

i)  $R = (ABCDE)$

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

$\Rightarrow (AB)^+ = (ABCDEF)$

Now,  $(AB)^+ = \{AB, C, D, E\}$  "AB" is the candidate key

②  $R(ABC)$

③  $R(ABC)$

Now (B)

Now (A)

Now (C)

Now (D)

Now (E)

Now (F)

Now (G)

Now (H)

Now (I)

Now (J)

Now (K)

Now (L)

Now (M)

Now (N)

Now (O)

Now (P)

Now (Q)

Now (R)

Now (S)

Now (T)

Now (U)

Now (V)

④  $R(AB)$

⑤  $R(BC)$

Now (B)

Now (C)

Now (A)

Now (D)

Now (E)

Now (F)

Now (G)

Now (H)

Now (I)

Now (J)

Now (K)

Now (L)

Now (M)

Now (N)

Now (O)

Now (P)

Now (Q)

Now (R)

Now (S)

Now (T)

Now (U)

Now (V)

⑥ Ex A

R = (ABC)

FD = {AB}

Now, A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

(2)  $R(ABCDE)$  FD =  $\{AB \rightarrow C, C \rightarrow D, B \rightarrow EA\}$   
 $\therefore$  keys of  $R$  are  $AB, AD, BC, CD$

Given already  $B \rightarrow EA$   
 $\Rightarrow (B)^+ = (ABCE)$

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

$\therefore$  No of superkeys =  $2^4 = 16$

(3)  $R(ABCDEF)$  FD =  $\{A \rightarrow B, C \rightarrow D, E \rightarrow F\}$   
 $\therefore (ACE)^+ = (ABCDEF)$

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

$\therefore$  "ACE" is the ck and  $2^3 = 8$  its subsets

(4)  $R = ABCD$ , FD =  $\{AB \rightarrow CD, D \rightarrow A\}$

$\Rightarrow (B)^+ = (ABCD) \Rightarrow \text{Now } (B)^+ = \{B\}$

Now  $(AB)^+ = (A, B, C, D)$

$(CB)^+ = (C, B)$   $\leftarrow$  (CB) and (DB) are candidate keys

$\therefore (DB)^+ = (D, B, A, C)$

$\therefore$  DB is the ck

5. Examples on Candidate Key - 2

$R = (ABCD)$

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

Now,  $A^+ = \{A\}$  Two attribute keys:  $(AB)^+ = (ABCD) \leftarrow$

$(AC)^+ = (AC)$

$(AD)^+ = (ADBC) \leftarrow$

$(BC)^+ = (BCD) \leftarrow$

$(BD)^+ = (BD) \leftarrow AB, AD,$

$(CD)^+ = (CD) \leftarrow BC, CD$  are

$D^+ = \{D\}$  CK's.

Ck's are: AB, AD, BC, CD

### EXAMPLES ON CANDIDATE KEYS - 3

$R = \overbrace{ABCDEF}$

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

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

$$\Rightarrow (AB)^+ = (ABCDEF)$$

$$\Rightarrow (CB)^+ = (ABCDEF)$$

$$\Rightarrow (DB)^+ = (ABCDEF)$$

$$\Rightarrow (EB)^+ = (ABCDEF)$$

$$\Rightarrow (FB)^+ = (ABCDEF)$$

$$\therefore CKS = (AB)(CB)(DB)(EB)(FB)$$

$$= 5CKS$$

### 18. CANDIDATE KEY - EXAMPLE - 4

$R = \overbrace{ABCDEF}$

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

Now,

$$A^+ = A$$

$$B^+ = B$$

$$C^+ = (CDEBFA) \checkmark$$

$$D^+ = (DEBFAC) \checkmark$$

$$E^+ = EFA$$

$$F^+ = FA$$

Now,

$$(AB)^+ = (ABCDEF)$$

$$\text{[Redacted]}$$

$$(AE)^+ = (AEF)$$

$$(AF)^+ = (AF)$$

$$\text{[Redacted]}$$

$$(EF)^+ = EFA$$

$$(BE)^+ = (BEFACD) \checkmark$$

$$(BF)^+ = (BFACDE) \checkmark$$

Now,

$$(BEF)^+ = (ABEF) \times$$

$$(AEF)^+ = (AEF) \times$$

$$(ABF)^+ \times$$

$$(ABE)^+ \times$$

$$\text{[Redacted]}$$

### 19. EXAMPLE

$R = \overbrace{ABCDEF}$

$$A \rightarrow BCDEF$$

$$BC \rightarrow ADEF$$

$$DEF \rightarrow ABC$$

Now,

$$A^+ = (ABCDE)$$

$$B^+ = (B) \times$$

$$C^+ = X$$

$$D^+ = X$$

$$E^+ = X$$

$$F^+ = X$$

### 20. EXAMPLE

$R = \overbrace{ABCDEF}$

$$FD = \{A \rightarrow B\}$$

Now,

$$(E)^+ = \{E\}$$

### 21. EXAMPLE

$R = \overbrace{ABCDEF}$

$$A \rightarrow BC$$

$$CD \rightarrow E$$

$$B \rightarrow D$$

$$E \rightarrow A$$

$\therefore C$  and  $D$  are candidate keys

AB and BE, BF are candidate keys.

19. EXAMPLES ON CANDIDATE KEYS - 5

$R(\overline{ABCDEF})$

$A \rightarrow BCDEF$

$BC \rightarrow ADEF$

$DEF \rightarrow ABC$

Now,

$$A^+ = (ABCDEF) \checkmark$$

$$B^+ = (B) \times$$

$$C^+ = X$$

$$D^+ = X$$

$$E^+ = X$$

$$F^+ = X$$

Now,

$$(BC)^+ = ABCDEF$$

Now,

$$(DEF)^+ = (DEFABC)$$

$$\therefore \text{Candidate keys} = A, BC, DEF$$

20. EXAMPLE ON CANDIDATE KEY - 6

$R(\overline{ABCDE})$

Now,

$$(ABEF) \times$$

Now,

$$(E)^+ = \{E\}$$

$$(AE)^+ = (AEBCD) \checkmark$$

$$(BE)^+ = (BECDA) \checkmark$$

$$(CE)^+ = (CEDAB) \checkmark$$

$$(DE)^+ = (DEABC) \checkmark$$

The CK's are : AE, BE, CE, DE

No. of CK's = 4

21. EXAMPLE ON CANDIDATE KEY - 1

$R(\overline{ABCDE})$

Now,

$A \rightarrow BC$

$$A^+ = (ABCDE) \checkmark$$

$$(BC)^+ = (ABCDE) \checkmark$$

$CD \rightarrow E$

$$B^+ = BD$$

$$(BD)^+ = (BD)$$

$B \rightarrow D$

$$C^+ = C$$

$$(CD)^+ = (ABCDE) \checkmark$$

$E \rightarrow A$

$$D^+ = D$$

$$E^+ = (EABC) \checkmark$$

∴ CK's are A, E, BC, CD

## 22. CANDIDATE KEY FOR SUB RELATION - EXAMPLE - 1

$R(ABCD)$

$\{AB \rightarrow CD, D \rightarrow A\}$  what are the candidate keys of subrelation  $R_1(BCD)$ ?

Now, Given Relation  $R_1(BCD)$

$$\begin{array}{ll} B^+ = B & (BC)^+ = BC \\ C^+ = C & (BD)^+ = \overbrace{BCDAB} \\ D^+ = DA & (CD)^+ = CDA \end{array} \quad \therefore \text{candidate key} = \underline{\underline{BD}}$$

## 23. CANDIDATE KEY FOR SUB RELATION EXAMPLE 2

$R(ABCDEF)$

FD's :  $\{AB \rightarrow C, B \rightarrow D, AD \rightarrow F, C \rightarrow D, D \rightarrow E, E \rightarrow F, E \rightarrow D\}$

find candidate keys for  $R_1(DEF)$ .

Given Sub Relation  $R_1(DEF)$

$$\begin{array}{ll} D^+ = \{DEF\} & \therefore \text{The candidate keys are } DE \\ E^+ = \{EDF\} & \\ F^+ = \{F\} & \end{array}$$

## 24. CANDIDATE KEYS FOR SUB RELATION -- EXAMPLE 3

$R(ABCDE)$

FD's  $\{A \rightarrow BC, CD \rightarrow E, B \rightarrow D, E \rightarrow A\}$

what are the candidate keys of  $R_1(ABCE)$

Given Sub Relation  $R_1(ABCE)$

$$A^+ = \overbrace{ABCDE}$$

$$B^+ = BD$$

$$C^+ = C$$

$$E^+ = \overbrace{EABCD}$$

NOW,

$$BC = \overbrace{BCDEA}$$

$BD \times$  (not intable)

$CD \times$  (not intable)

$\therefore \text{candidate keys are,}$

$$A, E, BC$$

## 25. CHECKING

Given the be  
functional Dep  
Inference Rule

$R(AB)$  and the

$$\begin{array}{ll} \underline{\underline{X}} \rightarrow \underline{\underline{Y}} \\ \emptyset \quad \emptyset \\ A \quad A \quad \Rightarrow \\ B \quad B \\ AB \quad AB \end{array}$$

The no. of Add

$$\begin{array}{ll} \emptyset \rightarrow \emptyset & \checkmark \\ X \emptyset \rightarrow A & \checkmark \\ X \emptyset \rightarrow B & \checkmark \\ X \emptyset \rightarrow AB & \checkmark \end{array}$$

The no. of FD's

## 26. CHECKING

$R(ABC)$

FD's :  $\{A \rightarrow B, B \rightarrow$

$$\begin{array}{ll} \underline{\underline{X}} \rightarrow \underline{\underline{Y}} \\ \downarrow \quad \downarrow \\ 2^3 \quad \times \quad 2^3 = 8 \times \end{array}$$

$\emptyset \rightarrow \emptyset$  (only

$A \rightarrow \{ABC\} =$

$B \rightarrow \{BC\} =$

$C \rightarrow \{C\} =$

$$A, B \rightarrow \{ABC\} = \{A, B, C\}$$



∴ In General the no. of functional dependencies that are possible over relation R containing 'n' attributes is  $2^{2^n}$

Now find  $G_1$

$$F: \{A \rightarrow C, A \rightarrow D\}$$

$G_1:$

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

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

$$(ED)^+ = \{E, A, H\}$$

### 27. GATE IT OS QUESTION ON ADDITIONAL FD's

In a schema with attributes A,B,C,D and E, following set of FD's are given  $A \rightarrow B$ ,  $A \rightarrow C$ ,  $CD \rightarrow E$ ,  $B \rightarrow D$ ,  $E \rightarrow A$  which of the following FD's is NOT implied by above set?

- a)  $CD \rightarrow AC$
- b)  $BD \rightarrow CD$
- c)  $BC \rightarrow CD$
- d)  $AC \rightarrow BC$

Now,  $(CD)^+ = \{CDEABD\} \Rightarrow (CD)^+ = (AC)$  ✓ possible

$(BD)^+ = \{BD\}$  = NOT POSSIBLE

$(BC)^+ = \{BCDEA\} = (CD)$  is possible

$(AC)^+ = \{ACBDE\} = (BC)$  is possible

### 28 EQUIVALENCE OF FD's

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

$G_1: \{A \rightarrow CD, E \rightarrow AH\}$  are both the functional dependencies Equivalent

Sol First check if 'F' covers  $G_1$  i.e  $F \supseteq G_1$  how to check is take

each FD in the set ' $G_1$ ' and find whether it is derivable from 'F' or not

And now, check  $G_1 \supseteq F$ , take each FD of 'F' and check if it is already implied in ' $G_1$ ' or not.

Now,  $F \supseteq G_1$  Now,  $G_1 = \{A \rightarrow CD, E \rightarrow AH\}$

$$\begin{array}{c} F: \\ \downarrow \quad \downarrow \\ A^+ = \{A, C, D\} \quad E^+ = \{EADCH\} \end{array}$$

Covered by  $\downarrow$  Covered by  $\downarrow$   $F \supseteq G_1$  is TRUE

$$G_1 \supseteq F \Rightarrow \text{TRUE}$$

in  $G_1$ :

### 29. EQUIVALENT

$$F: \{A \rightarrow B, B \rightarrow C\}$$

$$G_1: \{A \rightarrow BC, C \rightarrow A\}$$

Sol

$$F \supseteq G_1 \Rightarrow \text{TRUE}$$

$$\Rightarrow A$$

$$\Rightarrow B$$

$$\Rightarrow C$$