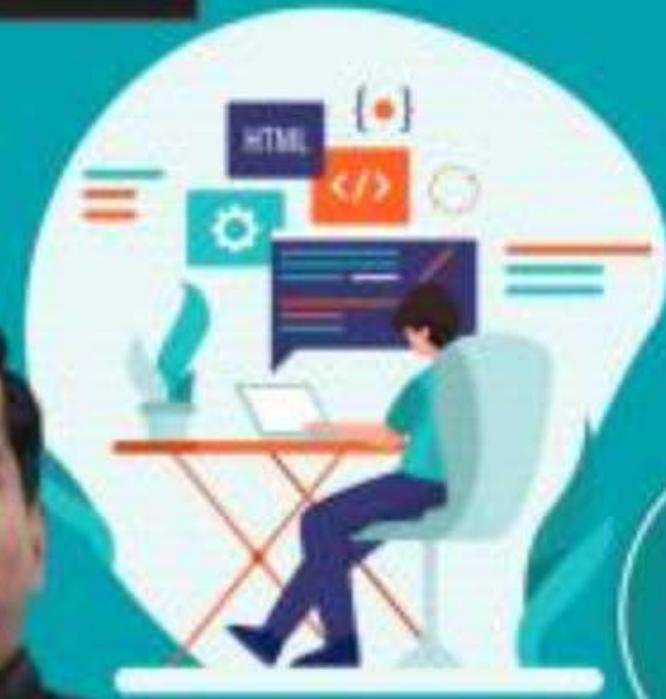




COMPUTER SCIENCE

DATABASES

MAHA REVISION



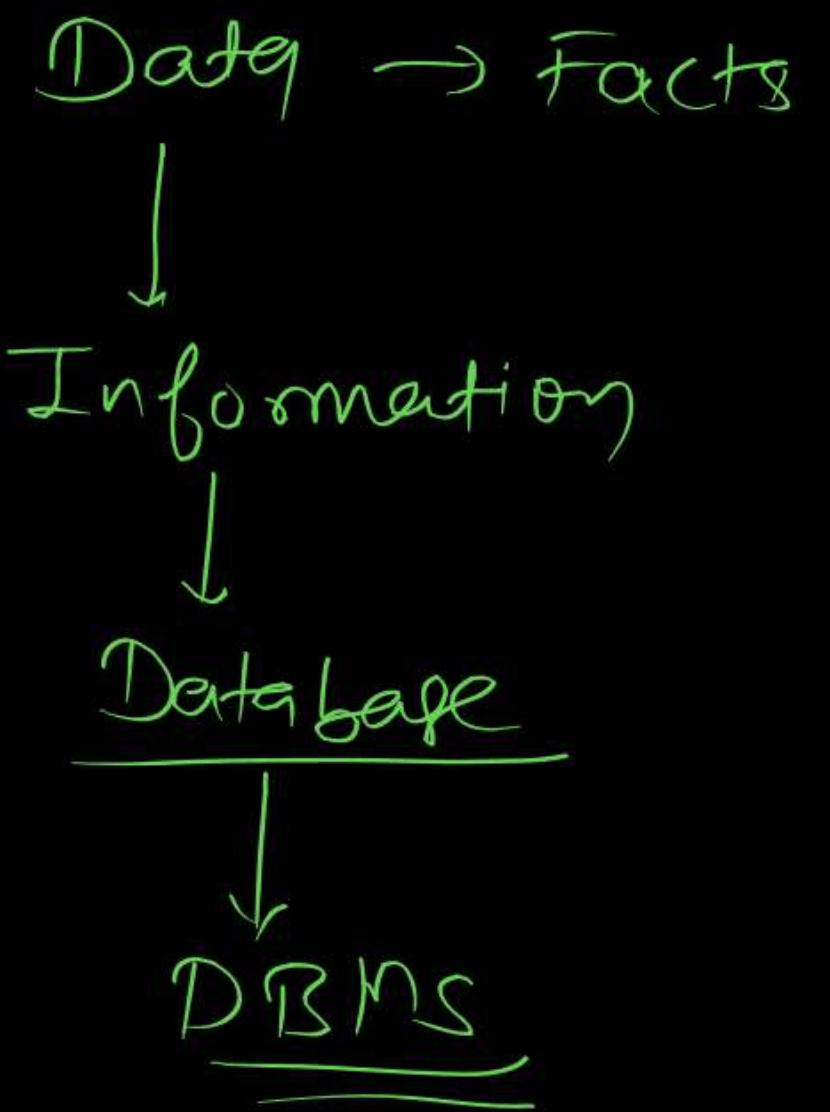
Vijay Agarwal sir

1. DATABASE DESIGN

Introduction



A database is an organized collection of structured data or related data.



Limitations of File System

- The physical details of the data to access the database are managed by the user.
- File system can be used to manage small database.
- When database is large, the operating system fails to control the concurrency.
- Only single user can access the whole data of the file system at a time.

Integrity Constraints Of RDBMS

- ❑ According to codd, data in database file must be stored in tabular format (set of row's and column's).
- ❑ According to codd, no two row's/records of the table should be equal.

Relational Schema : Structure / Definition of Table
Heading | Table Abstraction

STUDENT (Sid Sname Branch)

Relational Instance : Set of Records

Keys

Sub keys

foreign key

(eg) AB
Composite key
FN LN

Surrogate key
S.N.

minimal Candidate key (Assume 6 C.K)

Simple key
(eg) A
Roll No.

L.C.K Selected → P.K (unique + Not Null)

Remaining 5 C.K → AK/SK

Arity / Degree

Number of attributes of database table.

Cardinality

Number of records of database table.



Row → Tables / Record

Column → field / Record

<u>RDBMS</u>		
<u>STUDENT</u>	<u>①</u>	<u>②</u>
	<u>Sid</u>	<u>Sname</u>
<u>S₁</u>	A	CS
<u>S₂</u>	B	IT
<u>S₃</u>	C	CS
<u>S₄</u>	D	IT

Arity : ③

Cardinality : 4

Simple key → Roll No, A, id

Composite key : AB, FNLN, Sname Mobile No



Keys in database

- ❑ Candidate Key: A minimal set of attributes that differentiate records/row's of DB table uniquely.
- ❑ Primary Key : One of the candidate key whose field value cannot be null.
- ❑ Simple Candidate Key: A candidate key with only one attribute.
- ❑ Composite Candidate Key: A candidate key with atleast two attributes.
- ❑ Overlapped Candidate Key: Two or more candidate key with some common attribute.

AB, DB

$R(ABCDE)$

Prime Attribute

Candidate key = $\{A\}$

key / Prime Attribute = $\{A\}$

Non key / Non prime Attribute = $\{B, C, D, E\}$

Candidate key = $\{AB\}$

Prime Attribute = $\{A, B\}$

Non Prime Attribute = $\{C, D, E\}$

$R(A B C D E F G H)$

Prime Attribute = (\underline{A}, C, F, H)

Candidate = (A, H, F, C)

$R(A B C D E F)$

$A, B C, D E$

P.F
 $[A, B, C, D, E]$

Keys in database

- Prime attribute: The attribute that belongs to some/Any candidate keys of a relation.
- Non-prime attribute: The attribute that does not belongs to any of the candidate keys of the relation.

Example

Consider a relation R (ABCDE)

1. Assume candidate key : {AB, BC}
2. The above candidate keys are composite and overlapped.
3. Prime attribute {A, B, C}
4. Non-prime attribute : {D, E}

Difference between Primary key and Alternative key



Primary Key	Alternative Key
<ol style="list-style-type: none">1. Any one candidate key whose field value can not be null.2. Atmost one primary key allowed for any DB table. <i>O@ At most 1</i>	<ol style="list-style-type: none">1. All candidate key of relation except primary key, whose field value can be null.2. Many (zero or more) alternative keys are allowed for DB table.

NOTE:



For any RDBMS table there must be atleast one candidate key, whose
field value can not be null.

All (Unique + Not Null) \neq Primary key

At most 1 PK per Table (Relation)

Super key

The set of attributes which can differentiate records/tuples uniquely or

- super set of candidate key.

Example 1: R (ABCD) with CK = {A}

∴ Super key = CK ⊕ [Any subset of other attributes (BCD)]

= A ⊕ 2^3 = 8 Super keys.

Any Subset of Super key is also Super key.

Continued.

$R(ABCDE)$ $\Sigma A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow E$

$$[A]^+ = [ABCD E]$$

$$[B]^+ = [BCDE]$$

$$[C]^+ = [CDE]$$

$$[D]^+ = [DE]$$

$$[E]^+ = [E]$$

A is super key

Any subset of super key is also super key.

A is super key.

AB

AC

AD

... . . .

ABCDE : Super key

Candidate key : Minimal of Super key.

$R(ABCDE)$ $\{AB \rightarrow C, C \rightarrow D, B \rightarrow E\}$

$$(AB)^+ = [ABCDE]$$

AB is Super key

$$(A)^f = [A]$$

$$(B)^f = [BE]$$

AB is Candidate key

AB is Super key

ABC
ABD
ABE
:
ABCDE

Subkey

(Note) Every key is a Subkey
Candidate

But Every Subkey is Not Candidate Key.

Super key



Example 2 : Let R be the relational schema with n-attributes, R (A₁, A₂, A_n) then number of superkeys possible:

With (A₁) as candidate key : 2^{n-1}

With {A₁, A₂} as candidate key : $2^{n-1} - 2^{n-2} + 2^{n-1}$

With {A₁A₂, A₃A₄} as candidate key: $2^{n-2} - 2^{n-4} + 2^{n-2}$

The maximum number of super keys possible when each attribute of R is candidate

key : $2^{n-1} \cdot 2^n - 1$

Q8) R(A B C D E) [AB, CD]

$$2^5 - 2 + 2^5 - 2 - 2^5 - 4$$

$$2^3 + 2^3 - 2^1$$

$$8 + 8 - 2 = 14 \text{ SIC}$$

The minimum number of super key possible when all the attributes combined form single candidate key:

1 {A₁A₂ A_n : candidate key}

$$\text{Max \# Candidate key} = n_c \left\lfloor \frac{n}{2} \right\rfloor$$

$${}^6C_{\left\lfloor \frac{6}{2} \right\rfloor} = {}^6C_3$$

6 Attribute , Max # C.K

Single Attribute

$$GC_1 = 6$$

2 Attribute

$$GC_2 = 15$$

3 Attribute

$$GC_3 = 20 \quad \underline{\text{Ans}}$$

4 Attribute

$$GC_4 = 15$$

5 Attribute

$$GC_5 = 6$$

6 Attribute

$$GC_6 = 0$$

$R(ABCD)$

Max # Subkey = $2^n - 1$

$$\Rightarrow 2^4 - 1$$

- IS Subkey

$R(ABCD)$

A BCD = $2^3 = 8 \Rightarrow A, AB, AC, AD, ABC, ACD,$

B CD $\Rightarrow 2^2 = 4 \Rightarrow B, BC, BD, BCD$

C D $\Rightarrow 2^1 = 2 \Rightarrow C, CD$

$$- | \quad | \quad D$$

IS Subkey

$ABD, ABCD$

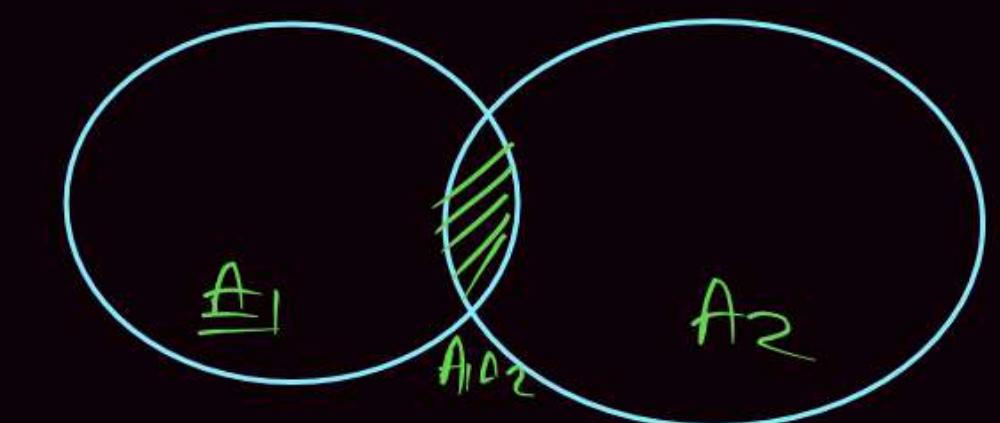
$$R(A \cap B \cap C \cap D \cap E) \subset k = (A, B)$$

$$\#S \cdot k = \frac{5-1}{2} + \frac{5-1}{2} - \frac{5-2}{2}$$

$$\Rightarrow 2^4 + 2^4 - 2^3$$

$$= 16 + 16 - 8$$

~~- (24 SIC)~~



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

$R(ABCD)$ CK: A

$$\# \text{Subkeys} = 2^{n-1}$$

n: #Attributes

$$\Rightarrow 2^{4-1}$$

$$= 2^3$$

= 8 Subkey

$R(ABCD)$

A BCD

$$2^3 = 8 \text{ Subkey}$$

$R(ABCD)$

A BCD

$$\{ 000 \rightarrow A$$

$$001 \rightarrow AD$$

$$010 \rightarrow AC$$

$$011 \rightarrow ACD$$

$$100 \rightarrow AB$$

$$101 \rightarrow ABCD$$

$$110 \rightarrow ABC$$

$$111 \rightarrow \underline{ABCD}$$

Referential Integrity Constraints



Foreign key

Foreign key is a set of attributes that references primary key or
alternative key of the same relation or other relation.

Referential Integrity Constraints

Foreign key

Student
Referential
Key must be
(PK or AK)

Sid	Sname	Age

Referenced Relation

Enroll
Sid is foreign
Key in Enroll

Sid	Cid	Fee

Referencing Relation

Foreign Key

Foreign Key: is a set of Attribute reference to the primary key or alternative key of the same table or same other table.



It is used to relate or relation (table) with other or same relation (table)

Referencing Relation: Table which contain the foreign key is known as Referencing Relation [CHILD Relation].

Referenced Relation: Table which is referenced by foreign key is referenced relation.

Foreign Key Constraint

[Referential Integrity Constraint]

STUDENT		
<u>Sid</u>	Sname	Login
S ₁	A	-P
S ₂	A	-P
S ₃	B	-P
S ₄	C ₁	-P

[Sid: Primary Key]
Referenced Relation (Parent)

Enrolled		
<u>Sid</u>	<u>Cid</u>	Fed
S ₁	C ₁	5K
S ₁	C ₂	6K
S ₂	C ₁	7K
S ₃	C ₂	8K

[Sid, Cid: Primary Key]
Referencing Relation (CHILD)

Sid of Enrolled table is the foreign key referencing to the primary key of student table.

CREATE TABLE ENROLLED

Sid Varchar (10)

Cid Varchar (10)

Fees Integer (11)

Primary key (Sid Cid)

Foreign Key (Sid) Reference Student

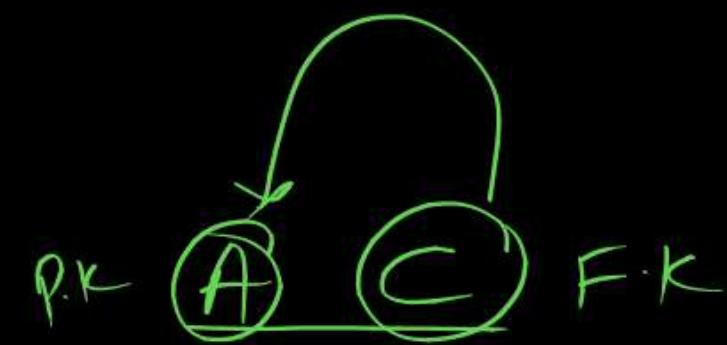
→ By Default foreign key
Reference to Primary key.

When Sid is the primary key of Student

Let login is primary key & Sid is alternative key then

Foreign Key (Sid) Reference Student (Sid)

→ When Sid is not primary key.



Foreign Key Constraint

[Referential Integrity Constraint]

P →

STUDENT		
<u>Roll No</u>	Name	Branch
1	A	CSE
2	B	IT
3	C	CSE

Registration		
<u>CNo</u>	Cname	<u>Roll No</u>
101	DBMS	1
102	OS	1
103	CD	3
104	TOC	-

Referenced Relation
(Parent)

Referencing Relation
(CHILD)

Duplicate @ NULL

Referenced

✓ Insert

✗ Delete

Referencing ^{Relation}

✗ Insert

✓ Delete

Note: The value present in Foreign key must be Present in Primary key of Referenced relation

Foreign key may contain duplicate & NULL values.

<u>Parent table</u> <u>(Referenced table)</u>	<u>CHILD table</u> <u>(Referencing Relation)</u>
✓ Insert < 4 D ECE>	✗ Insert < 105 DSA 67
✗ Delete < 1 A CSE>	✓ Delete < 103 CD 3>

Note: Deletion from the Referenced Relation and Insertion into Referencing Relation may violate Foreign key constraint.

Note: A Relation can Act as Parent & CHILD i.e. Relation may contain a primary key & a Foreign key that Refer to the same Relation.

Referential Integrity Constraint

(1) Referenced Relation

- (i) Insertion: No Violation
- (ii) Deletion: May cause Violation if Primary key is used by referencing relation

I. ON DELETE NO ACTION. → Default ⇒ Restrict the Deletion

II. ON DELETE CASCADE.

III. ON DELETE SET NULL.



Q.

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

P
W

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) and (7, 2)
- C (5, 2), (7, 2) and (9, 5)
- D 1

PK FK

A	C
2	4
3	4
4	3
5	2
7	2
9	5
6	4

(5,2) (7,2)
↓
(9,5)

Q.

Referenced *Referencing* P
Let R (a, b, c) and s(d, e, f) be two relation in which d is the foreign key of S that refers to the primary key of R. Consider the following four operations on R and S.

R
referenced (i) Insert into R

Delete ~~(iii)~~ Delete from R

S: Referencing
(ii) Insert into S

Insert X
(iv) Delete from S

Which of the following is true about the referential integrity constraint above?

A None of (i), (ii), (iii), or (iv) can cause its violation

B All of (i), (ii), (iii), and (iv) can cause its violation

C Both (i) and (iv) can cause its violation

D Both (ii) and (iii) can cause its violation

Ans (D)

Referenced Relation

1. Insertion : No violation
2. Deletion : [May cause violation]
 - (a) On delete no action : Means if it cause problem on delete then deletion is not allowed on table.
 - (b) On delete cascade : If we want to delete primary key value from referenced table then it will delete that value from referencing table also.
 - (c) On delete set null : If we want to delete primary key value from referenced table then it will try to set the null values in place of that value in referencing table.

Foreign key

3. Updation : [May cause violation]

- (a) On update no action
- (b) On update cascade
- (c) On update set null

Referencing Relation → Foreign key

- 1. Insertion : [May cause violation]
- 2. Deletion : No violation
- 3. Updation : [May cause violation]

Example

P.K	F.K
A	B
2	4
3	4
4	5
5	4
6	2

B is foreign key
Referencing A,
Delete (2, 4) and
on delete cascade

A	B
3	4
4	5
5	4

Result

So, If we delete (2, 4) then PK "2". gets deleted from the table and all the tuples in which B is referencing PK.2" also gets deleted.

Database Design Goals



1. 0% redundancy
2. Loss-less join
3. Dependency preservation

Functional Dependency

Let R be the relational schema and x, y be the non-empty set of attributes.

Consider t_1, t_2 are any tuples of relation then $x \rightarrow y$ (y functionally determined by x):

If $\forall t_1, t_2 t_1.x = t_2.x$ then $t_1.y = t_2.y$

$$\boxed{x \rightarrow y}$$

If $t_1.x = t_2.x$ then $t_1.y = t_2.y$ must be same

Whenever X Value Repeat Corresponding Y Value

Must be Same

Types of Functional Dependency

1. Trivial Functional Dependency

Always Valid

Consider relational schema R(ABCD)

A FD $x \rightarrow y$ is trivial FD only if $x \supseteq y$.

① Trivial FD

$x \supseteq y$

Example:

$$\underline{AB} \rightarrow \underline{A}$$

$$x \supseteq y$$

$$\underline{A} \rightarrow \underline{A}$$

$$\underline{AB} \rightarrow A$$

$$\underline{B} \rightarrow \underline{B}$$

$$A \rightarrow A$$

$$\underline{B} \rightarrow \underline{B}$$

② Non Trivial

③ Semi Non Trivial

Types of Functional Dependency

2. Non-trivial Functional Dependency

Consider relational schema R(ABCD)

A FD $x \rightarrow y$ is non-trivial only if

$x \cap y = \phi$ means no common attributes between x and y attribute sets.

Example :

$$\begin{array}{l} \cancel{A \rightarrow B} \\ \cancel{AB \rightarrow CD} \end{array}$$

FD Definition

If $t_1x = t_2x$ then $t_1y = t_2y$ must be same

Types of Functional Dependency



3. Semi-Non-Trivial Functional Dependency

Example :

$$A \rightarrow AB = \{A \rightarrow A, A \rightarrow B\}$$

$$\underline{\underline{A \rightarrow AB}}$$

$$X \not\supseteq Y \text{ & } X \cap Y \neq \emptyset$$

$$X \not\supseteq Y \text{ & } X \cap Y = \emptyset$$

x	y	z
-	-	-
-	-	-
-	-	-
-	-	-

$x \rightarrow y$

$x \rightarrow z$

$x \rightarrow yz$

$y \rightarrow x$

$y \rightarrow z$

$y \rightarrow xz$

$z \rightarrow x$

$z \rightarrow y$

$z \rightarrow yx$

$xy \rightarrow z$

$yz \rightarrow x$

$xz \rightarrow y$

Q.

Given the following relation instance.

[2000: 2 Marks]

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 $Z \rightarrow Y$ $\text{Arg}(R)$

B

 $YZ \rightarrow X$ and $Y \rightarrow Z$

C

 $YZ \rightarrow X$ and $X \rightarrow Z$

D

 $XZ \rightarrow Y$ and $Y \rightarrow X$

Q.

Consider the relation $X(P, Q, R, S, T, U)$ with the following set of functional dependencies

[2015: 1 Marks]

$$\begin{aligned} F = \{ \\ \{P, R\} \rightarrow \{S, T\} \\ \{P, S, U\} \rightarrow \{Q, R\} \\ \} \end{aligned}$$

Which of the following is the trivial functional dependency in F^+ is closure of F ?

- A $\{P, R\} \rightarrow \{S, T\}$
 - B $\{P, R\} \rightarrow \{R, T\}$
 - C $\{P, S\} \rightarrow \{S\}$
 - D $\{P, S, U\} \rightarrow \{Q\}$
- Ans(C)*

Attribute Closure (X^+)

The set of all possible attributes determined by x .

Example :

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

$$\therefore \underline{(A)^+} = [ABCDEF]$$

$$(B)^+ = [BCDEF]$$

$$(C)^+ = [CDEF]$$

$$(D)^+ = [DE]$$

$$(E)^+ = [E]$$

A B C D E
 $2^4 = 16$ superkey

Subkey

A is subkey

Note

Any super set of subkey is also subkey

A

AB

AC

ABC

;

ABCDE

Armstrong's Axioms

1. Reflexive : If $x \sqsupseteq y$ then $x \rightarrow y$ or $x \rightarrow x$

2. Transitivity : If $x \rightarrow y$ and $y \rightarrow z$ then $x \rightarrow z$

$$A \rightarrow BC \Rightarrow A \rightarrow B, A \rightarrow C$$

3. Augmentation : If $x \rightarrow y$ then $xz \rightarrow yz$

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

4. Splitting : If $x \rightarrow yz$ then $x \rightarrow y, x \rightarrow z$

$$\begin{aligned} A \rightarrow C \\ B \rightarrow C \end{aligned} \Bigg) \times$$

5. Union : If $x \rightarrow y$ and $x \rightarrow z$ then $x \rightarrow yz$

6. Pseudo transitivity : If $x \rightarrow y, yw \rightarrow z$ then $xw \rightarrow z$

Candidate key

↳ Minimal of Subkey



Finding Candidate Key [Minimal Super Key]

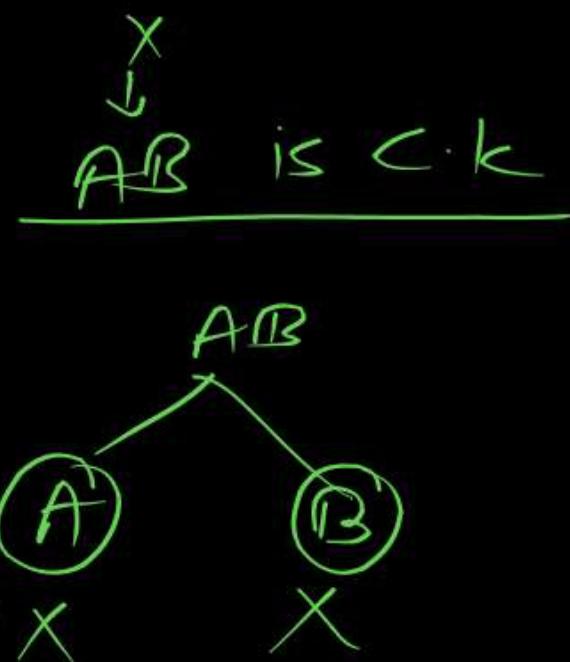
X is candidate key of R If and only if

1. x must be super key of relation R

$(x)^+ = \{\text{determine all attributes of } R\}$

2. No proper subset of 'x' is super key of relation R.

$\forall y \subset x : (y)^+ = \{\text{not determine all attributes of } R\}$



- $x \rightarrow y$ is a non-trivial FD in R with y is a prime attribute then relation R has atleast two candidate key.

$(x \rightarrow y)$: Non - trivial FD

↓
Prime-attribute

Finding Multiple Candidate key

First Find Only One CK then that Attribute is Prime/Key Attribute

If $X_{\text{Attribute}} \rightarrow \{\text{Prime Attribute}\}$

Q.

Consider the following relational schema $R(ABCDEF)$ with functional dependency $\{AB \rightarrow C, C \rightarrow D, D \rightarrow E, E \rightarrow F, F \rightarrow B\}$.
The number of candidate keys for relation R ?

P
W

Q.

$R(ABCDE) \{A \rightarrow BC, CD \rightarrow E, \underline{B \rightarrow D}, E \rightarrow A\}$

Prime Attribute = $\{A, E, C, D, B\}$

Find candidate keys for the relation R ?

$$(A)^+ = \{ABCDE\}$$

A is candidate key —①

If

X Attribute \rightarrow [Prime Attribute]

$$CD \rightarrow E$$

$$(CD)^+ = \{ABCDE\}$$

CD is super key

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

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

UCK

A

E

CD

BC

Ans

$$\overline{E \rightarrow A}$$

$$(E)^+ = \{FABCD\}$$

E is candidate key

—②

CD is candidate key

—③

C

B

$$\overline{B \rightarrow D}$$

$$(CB)^+ = \{ABCDE\}$$

$$(B)^+ = \{BD\}$$

BC is CK

—④

Q.

Relation R has eight attributes ABCDEFGH. Fields of R contain only atomic values.

$F = \{CH \rightarrow G, A \rightarrow BC, B \rightarrow CFH, E \rightarrow A, F \rightarrow EG\}$ is a set of functional dependencies (FDs) so that F is exactly the set of FDs that hold for R.

How many candidate keys does the relation R have? [2013: 2 Marks]

A

3

B

4

C

5

D

6

Ans

$$(AD)^{\perp} = \{ABC\overline{EFGH}\}$$

$$(AD)^+ = \{ABCDEF\overline{GH}\}$$

D

Q.

Consider the following relational schema $R(ABCDEF)$ with functional dependency $\{AB \rightarrow C, C \rightarrow D, D \rightarrow E, E \rightarrow F, F \rightarrow B\}$
The number of candidate keys for relation R ?

P
W

$$P.A = \{A, B, F, E, D, C\}$$

$$\overset{A}{\supset} (AB)^+ = [ABCDEF]$$

$$\overset{A}{\supset} (A)^+ = [A]$$

$$\overset{B}{\supset} (B)^+ = [B]$$

$$\boxed{AB \text{ is CK}} - ①$$

$\chi_{\text{Attribute}} \rightarrow (\text{Prime Attribute})$

$$\overset{F}{\supset} (F)^+ = [FB]$$

$$(AF)^+ = [ABCDEF]$$

$$(F)^+ = [FB]$$

$$\boxed{AF \text{ is CK}} - ②$$

$$\underline{E \rightarrow F}$$

$$\overset{E}{\supset} (AE)^+ = [AEFRCD]$$

$$(E)^+ = [EFRB]$$

$$\boxed{AE \text{ is CK}} - ③$$

$$\overset{C}{\supset} \underline{C \rightarrow D}$$

$$\overset{C}{\supset} (AC)^+ = [ABCDEF]$$

$$(C)^+ = \underline{[CDEFB]}$$

$$\boxed{AC \text{ is CK}} - ④$$

$$\underline{D \rightarrow E}$$

$$\overset{D}{\supset} (AD)^+ = [ABCDEF]$$

$$(D)^+ = [DEFB]$$

$$\boxed{AD \text{ is CK}}$$

5CK

AB
AF
AE
AD
AC

Q.

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

[2014: 1 Mark]

- A In all candidate keys of R.
- B In some candidate key of R.
- C In a foreign key of R.
- D Only in the primary key of R.

Ans (B)

Membership of FD

A FD $x \rightarrow y$ is member (logically implied) of FD set F if and only if $(x)^+$ should determine 'y' in FD set F.

Example : Given FD Set F : { }



$x \rightarrow y$ FD belongs to F or not?



x^+ = { ... y ... }

then $x \rightarrow y$ is member of F.

$R(xyz)$ $\{x \rightarrow y, y \rightarrow z\}$

$x \rightarrow z$ is Member

$x \rightarrow z$ logically implied

$(x)^+ = [x \dots \underline{y} \dots z]$

$x \rightarrow z$ is implied | Member

Q.

In a schema with attributes A, B, C, D and E following set of functional dependencies are given

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

$$CD \rightarrow E$$

$$B \rightarrow D$$

$$E \rightarrow A$$

$$(CD)^+ = [CDEAB]$$

$$(BC)^+ = [BCD \dots]$$

$$(AC)^+ = [ABC \dots]$$

Ans (A)

Which of the following functional dependencies is NOT implied by the above set

[MCQ: GATE - 2M]

C ~~Implied~~ $CD \rightarrow AC$

D ~~Implied~~ $BC \rightarrow CD$

A BD \rightarrow CD
B AC \rightarrow BC

$$(BD)^+ = [BD]$$

Equality between FD Set

$F: [$

$] \quad G: [$

$]$

$$\boxed{F = G}$$

If $F \text{ Cover } G$: True

$G \text{ Cover } F$: True

$F \text{ Cover } G$: F Cover All the FD's of G .

All G FD's ^{OR} logically implied in F FD Set

$G \text{ Cover } F$: G Cover All the FD's of F

All F FD ^{OR} logically implied in G FD Set.

Testing of Two FD Sets whether they are equal or not

F and G FD sets equality test:

F and G FD sets logically equal if and only

1. F cover's G : All FD's of G set must be member's of F set.
2. G cover's F : All FD's of F set must be member's of G set.

$$F \supseteq G$$

$$F \supseteq G$$

F Cover G	G Cover F	Result
<u>True</u>	<u>True</u>	$F \equiv G$
<u>True</u>	False	$F \supset G$
<u>False</u>	<u>True</u>	$F \subset G$
False	False	<u>F & G are not comparable</u>

Q.

Consider relation schema R(A C D E H) with two set of FD's

F : [A → C, AC → D, E → AD, E → H]

[MSQ]

G : [A → CD, E → AH]

Which of the following is correct?

- A F Cover G
- B G Cover F
- C F and G are equivalent
- D None of these

G Cover F $\checkmark A \rightarrow C$ $(A)^+ = [ACD]$ $\checkmark AC \rightarrow D$ $(AC)^+ = [ACD]$ F Cover G $\checkmark A \rightarrow CD$ $(A)^+ = [ACD]$ $\cancel{E \rightarrow AD}$ $(E)^+ = [EAHCD]$ $\checkmark E \rightarrow AH$ $(E)^+ = [EAHCD]$ TrueTrue

$F = G$

Minimal Cover or Canonical Cover

- Minimal Cover of given FD Set (F) is minimum possible FD's (F_m), which is logically equal to ' F ' : ($F = F_m$)
- Remove extraneous attribute (useless attribute) from each determinant of FD set F .

$$wxy \rightarrow z$$


Extraneous Attribute $\cancel{\{wxy \rightarrow z, w \rightarrow x\}} \equiv \underline{\{wy \rightarrow z, w \rightarrow x\}}$

- Every FD must be simple (RHS of any FD should have single attribute).

Minimal Cover : Elimination of R.F.D

F: $[A \rightarrow B, B \rightarrow C, A \rightarrow C]$

$A \rightarrow C$ is R.F.D

G: $[A \rightarrow B, B \rightarrow C]$

$(A)^+ [AB \subseteq]$

$\therefore A \rightarrow C \underline{\underline{R.F.D}}$

Minimal Cover = $[A \rightarrow B, B \rightarrow C]$

$$\begin{array}{c} \text{AB} \\ \hline \text{A} \end{array} \rightarrow C \quad A \rightarrow B$$

A is extra if $\underline{[B]}^+ = [\dots \underline{A}]$

B is extra if $[A]^+ = [\dots \underline{B}]$

$F \Rightarrow$
 $Xy \geq \omega$
 $y \rightarrow z$
 $x \rightarrow y$

~~$Xy \geq \omega$~~
 $y \rightarrow z$

z is extra

$F \equiv G$

$F \text{ Cover } G$

~~$X \rightarrow \omega$~~ $(x)^+ = (ny_2\omega)$

~~$y \rightarrow z$~~ $(y)^+ = (y_2)$

~~$X \rightarrow y$~~ $(x)^+ = (ny_2\omega)$

True

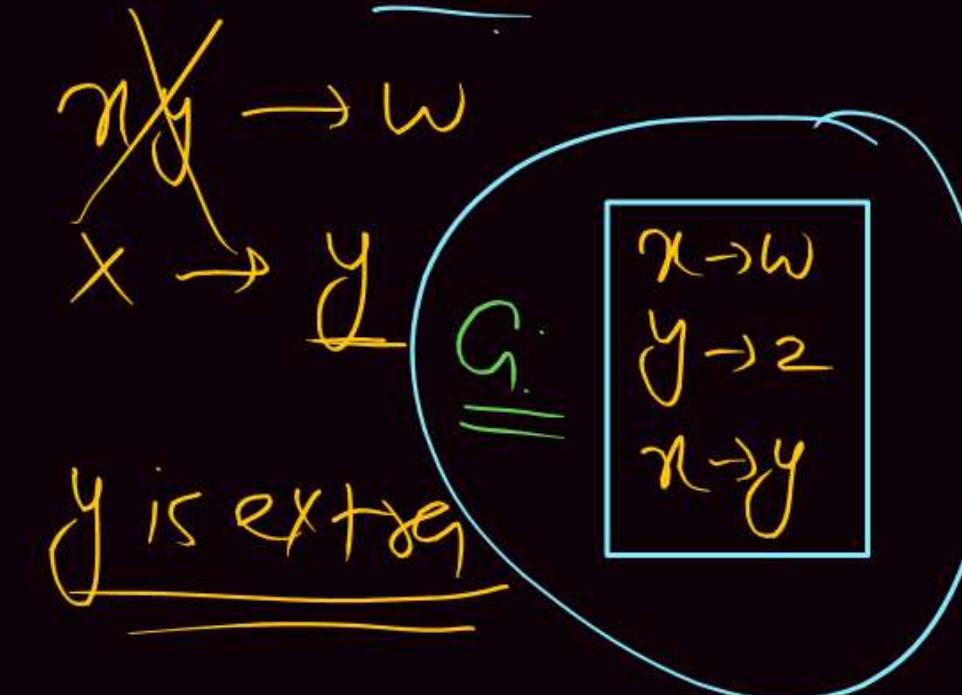
$G \text{ Cover } F$

~~$Xy \geq \omega$~~ $(ny_2)^+ = (ny_2\omega)$

~~$y \rightarrow z$~~ $(y)^+ = (yz)$

~~$X \rightarrow y$~~ $(x)^+ = (ny_2\omega)$

True



$F \equiv G$

Minimal Cover or Canonical Cover

Example : $F = \{A \rightarrow CD\}$ then $\{A \rightarrow C, A \rightarrow D\}$

- FD set must be non-redundant FD. (eliminating unnecessary FD's)

Example : $F = \{A \rightarrow B, B \rightarrow C, \underline{A \rightarrow C}\}$ then

$F = \{A \rightarrow B, B \rightarrow C\}$ because $A \rightarrow C$ is redundant.

NOTE:

Minimal Cover of FD Set (F) may not be unique, but all minimal cover's are logically equal.

$$\therefore Fm_1 \equiv Fm_2 = F$$

Step 1 : Split the FD such that R.H.S contain single attribute

Step 2 : L.H.S find Extra Attribute

Step 3 : find Redundant FD

Procedure to find minimal set

Step (1)

Split the FD such that RHS contain single Attribute.

Ex. $A \rightarrow BC$, \Rightarrow $A \rightarrow B$ and $A \rightarrow C$

Step (2)

Find the redundant attribute on L.H.S and delete them.

Ex. $AB \rightarrow C$,

A - Can be deleted $[B]^+ = [A]$ B^+ Contains 'A'

OR

B can be delete if A^+ contain 'B' $[A]^+ = \dots B$

Step

(3)

R.F.D

Find the redundant FD and delete them from the set

Ex. $\{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$

$\{A \rightarrow B, B \rightarrow C\}$ $[A]^+ = [A B \overline{C}]$

$[A]^+ = [AC]$ $[B]^+ = [B]$

Example1:

[AB → CD, A → E, E → C]

Step1 (R.H.S) $AB \rightarrow C$, $AB \rightarrow D$ $A \rightarrow E$, $E \rightarrow C$

Step2 L.H.S
 Ext^{ng} Attribute

AB → C A is ext^{ng} if $(B)^+ = C$. . . A
 $(A)^+ = [AEC]$ B is ext^{ng} if $(A)^+ = C$. . . B
 $(B)^+ = [B]$

Step3

① ~~AB → C~~ ② ~~AB → D~~ ③ $\checkmark A \rightarrow E$, ④ $E \rightarrow C$

$(AB)^+ = [ABDEC]$ $(AB)^+ = [ABEC]$ $(A)^+ = [A]$ $(E)^+ = [E]$

$AB \rightarrow D$
$A \rightarrow E$
$E \rightarrow C$

Ans

Example 2:

$[A \rightarrow C, AC \rightarrow D, E \rightarrow AD, E \rightarrow H]$

Step 1 $A \rightarrow C, AC \rightarrow D, E \rightarrow A, E \rightarrow D, E \rightarrow H$

Step 2 $A \cancel{\rightarrow} D$ $(C)^t = [C]$
 $(A)^t = [AC] \Rightarrow C \text{ is extra}$

① $\checkmark A \rightarrow C$ ② $\checkmark A \rightarrow D$ ③ $\checkmark E \rightarrow A$ ④ $\cancel{E \rightarrow D}$ ⑤ $\checkmark E \rightarrow H$
 $(A)^t = [AD]$ $(A)^t = [AC]$ $(E)^t = [EDH]$ $(E)^t = [EAHC \cancel{D}] \Rightarrow$ $(E)^t = [EAHD]$

$A \rightarrow C, A \rightarrow D, E \rightarrow A, E \rightarrow H$	⊗	$A \rightarrow CD, E \rightarrow AH$
--	---	--------------------------------------

Ans

Normalization



Normalization is used to eliminate/reduce redundancy in DB relations.

The normalization is especially meant to eliminate the following anomalies:

- Insertion anomaly
- Deletion anomaly
- Update anomaly
- Join anomaly

Normalization of DB table

Decompose relation into two or more sub-relations in-order to reduce or eliminate redundancy and DB anomalies.

Properties of Decomposition

- Loss-less Join decomposition: Relational Schema R decomposed into $R_1, R_2, R_3, \dots, R_n$ sub-relations.
- In general $[R_1 \bowtie R_2 \bowtie R_3 \dots \bowtie R_n] \supseteq R$
 1. If $[R_1 \bowtie R_2 \bowtie R_3 \dots \bowtie R_n] \equiv R$ then loss-less join decomposition.
 2. If $[R_1 \bowtie R_2 \bowtie R_3 \dots \bowtie R_n] \supset R$ then lossy join decomposition.

Natural Join ($R \bowtie S$)

Step 1 Cross Product of R & S

Step 2: Select the tuple which satisfy

Equality Condition on All

Common Attribute of $R \bowtie S$ [From $R \times S$]

Step 3:
Projection of Distinct Attributes

\underline{R}	\underline{S}
n_1 Tuple	n_2 Tuple
C_1 Attribute	C_2 Attribute
$R \times S = n_1 \times n_2$ Tuple	
$C_1 + C_2$ Attribute	

Q.

R(ABC)

A	B	C
1	5	5
2	5	8
3	8	8

Step 1
Wisey Join

Decomposed into

Q.1 $R_1(AB) \text{ & } R_2(BC)$

Q.2 $R_1(AB) \text{ & } R_2(AC)$

Step 3

A	B	C
1	5	5
1	5	8
2	5	5
2	5	8
3	8	8

R_1	R_2
A B	B C
1 5	5 5
2 5	5 8
3 8	8 8

3 Table 3 Table
2 Attribute 2 Attribute

$R_1 \times R_2 = 3 \times 3 = 9$ Table

$2+2=4$ Attributes

Step 1

$R_1 A$	$R_1 B$	$R_2 B$	$R_2 C$
1	5	5	5
1	5	5	8
2	5	5	5
2	5	5	8
2	5	8	8
3	8	5	5
3	8	5	8
3	8	8	8

P
W

$R_1 B = R_2 B$

Step 2

$R_1 A$	$R_1 B$	$R_2 B$	$R_2 C$
1	5	5	5
1	5	5	8
2	5	5	5
2	5	5	8
3	8	8	8

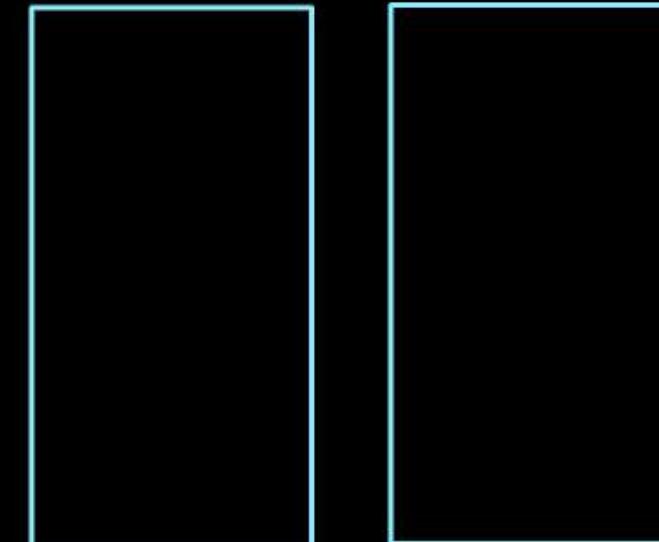
Q.

R(ABC)

 $A \rightarrow B$

A	B	C
1	5	5
2	5	8
3	8	8

 ~~$B \rightarrow A$~~

 ~~$B \rightarrow C$~~


(Q.1)

$R_1(AB)$	$R_2(BC)$
-----------	-----------

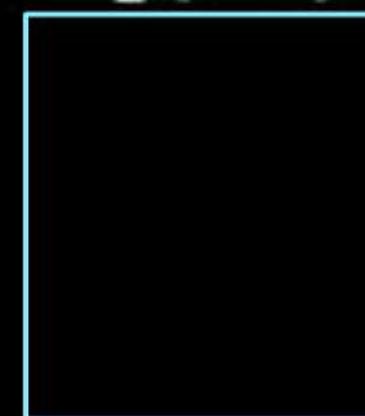
$$\textcircled{1} R_1(AB) \cup R_2(BC) = R(ABC)$$

$$\textcircled{2} R_1(AB) \cap R_2(BC) = \emptyset$$

$$(R)^t = [B]$$

Lossy Join

Decomposed into

Q.1 $R_1(AB) \& R_2(BC)$ Q.2 $R_1(AB) \& R_2(AC)$ 
 $\textcircled{1} R_1(AB) \& R_2(AC)$
 $\textcircled{2} R_1(AB) \cup R_2(AC) \Rightarrow ABC$

$$R_1(AB) \cap R_2(AC) = [A]$$

 $(A)^t = (AB)$ Subkey of R_1
Lossless Join
P
W

Q.

 $R(ABC)$

A	B	C
1	5	5
2	5	8
3	8	8

A	B
1	5
2	5
3	8

A	C
1	5
2	8
3	8

Step1

$R_1.A$	$R_1.B$	$R_2.A$	$R_2.B$
1	5	1	5
1	5	2	8
1	5	3	8
2	5	1	5
2	5	2	8
2	5	3	8
3	8	1	5
3	8	2	8
3	8	3	8

Decomposed into

Q.1 $R_1(AB)$ & $R_2(BC)$ Q.2 $R_1(AB)$ & $R_2(AC)$

Step 2
 \downarrow
 Lossless Join

A	B	C
1	5	5
2	5	8
3	8	8

Lossless Join

Step2

A	B	A	C
1	5	1	5
2	5	2	8
3	8	3	8

P
W

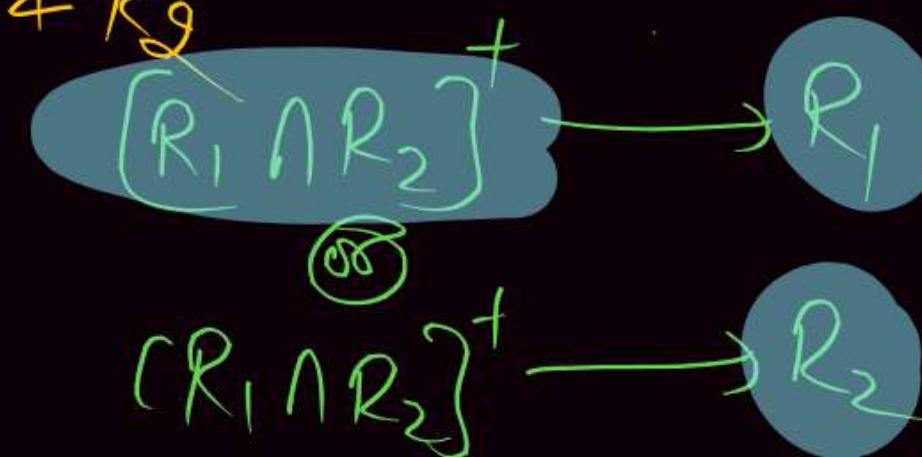
Lossless Join

Let R be the Relational Schema is decomposed into 2 sub Relation R_1 & R_2

$R_1 \bowtie R_2$ is lossless iff

① $R_1 \cup R_2 = R$

→ ② If Common Attribute of R_1 & R_2 either sub key of R_1 or sub key of R_2 .



Lossy Join

① If Common Attribute of R_1 & R_2

Neither a Subkey of R_1 $(R_1 \cap R_2) \nrightarrow R_1$

(nor)

Subkey of R_2

(or)

$(R_1 \cap R_2) \nrightarrow R_2$

$R_1(ABC) \cup R_2(CDE)$
 $R_1(ABC) \cup R_2(BCD)$
F' missing
Lossy

②

$R(ABCDEF)$

$R_1(ABC)$ $R_2(DEF)$

Lossy BC₂ No Common Attribute

③

$R(ABCDEF)$

$R_1(ABC)$ $R_2(CDE)$

Lossy BC₂ Attribute 'F' Missing

Example:

Let R be the relational schema decomposed into R1 and R2.

Given decomposition is loss-less join if-

1. $\underline{R_1 \cup R_2 = R}$ (all attributes covers)
2. $R_1 \cap R_2 \neq \emptyset$
3. $R_1 \cap R_2 \rightarrow \underline{R_1}$ or $\underline{R_1 \cap R_2 \rightarrow R_2}$
 $\underbrace{R_1 \cap R_2 \text{ is SK of } R_1}$ $\underbrace{R_1 \cap R_2 \text{ is SK of } R_2}$

Q) $R(ABCDEFG)$ $\{AB \rightarrow C, C \rightarrow D, D \rightarrow E, E \rightarrow FG\}$

$R_1(ABCD) \not\models R_2(DEFG)$

Soln (i) $R_1(ABCD) \cup R_2(DEFG) \Rightarrow ABCDEFG$

(ii) $R_1(ABCD) \cap R_2(DEFG) \Rightarrow [D]$

$[D]^- = \{DEFG\}$ Superkey of R_2

$R_{12}(ABCPFG)$
Lossless Join

Q.

R(ABCDEG) {AB → C, AC → B, AD → E, B → D, BC → A, E → CG}

P
W

1. Decomposed into R₁(AB) R₂(BC) R₃(ABDE) and R₄(EG)

Q. 2. Decomposed into R₁(AB) R₂(BC) R₃(ABDE) and R₄(ECG)

Solⁿ² R₁(AB) ∩ R₂(BC) ⇒ (B) ⇒ (B)⁺ = (BD) Not subkey of R₁ Not R₂

R₁(AB) ∩ R₃(ABDE) ⇒ AB ⇒ (AB)⁺ = (AB, CDEG) Subkey of R₁ ⊕ R₃

R₁₃(ABDE) ∩ R₄(ECG) ⇒ E ⇒ (E)⁺ = (ECG) Subkey of R₄

R₁₃₄(ABCDEFG) ∩ R₂(BC) ⇒ BC ⇒ (BC)⁺ = (BC, DAEG) Subkey of R₂
or R₁₃₄

Solⁿ² R₁₂₃₄(ABCDEFG) lossless

Q.

$R(ABCDEFG) \{AB \rightarrow C, AC \rightarrow B, AD \rightarrow E, B \rightarrow D, BC \rightarrow A, E \rightarrow CG\}$

P
W

① Decomposed into $R_1(AB)$ $R_2(BC)$ $R_3(ABDE)$ and $R_4(EG)$

2. Decomposed into $R_1(AB)$ $R_2(BC)$ $R_3(ABDE)$ and $R_4(ECG)$

Lossy

$$R_1(AB) \cap R_2(BC) \Rightarrow [B] \Rightarrow [B]^+ = [BD]$$

$$R_1(AB) \cap R_3(ABDE) \Rightarrow [AB] \Rightarrow [AB]^+ = [AB.CDEG.] \text{ Subkey of } R_1$$

R_3

$$R_1(ABDE) \cap R_4(EG) \Rightarrow [E] \Rightarrow [E]^+ = [\underline{EGC}] \text{ Subkey of } R_4$$

$$R_{134}(ABDEG) \cap R_2(BC) \Rightarrow [B] \Rightarrow [B]^+ = [BD] \text{ Not a subkey of } R_{134}$$

No
 R_2

Lossy

Dependency Preserving Decomposition



Relational Schema R with FD set F decomposed into R_1, R_2, \dots, R_n sub relations

Assume F_1, F_2, \dots, F_n FD sets of sub relations respectively.

In general

$$[F_1 \cup F_2 \cup \dots \cup F_n] \subseteq F$$

1. If $[F_1 \cup F_2 \cup \dots \cup F_n] = F$ then dependency preserving decomposition.
2. If $[F_1 \cup F_2 \cup \dots \cup F_n] \subset F$ then not dependency preserving decomposition.

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

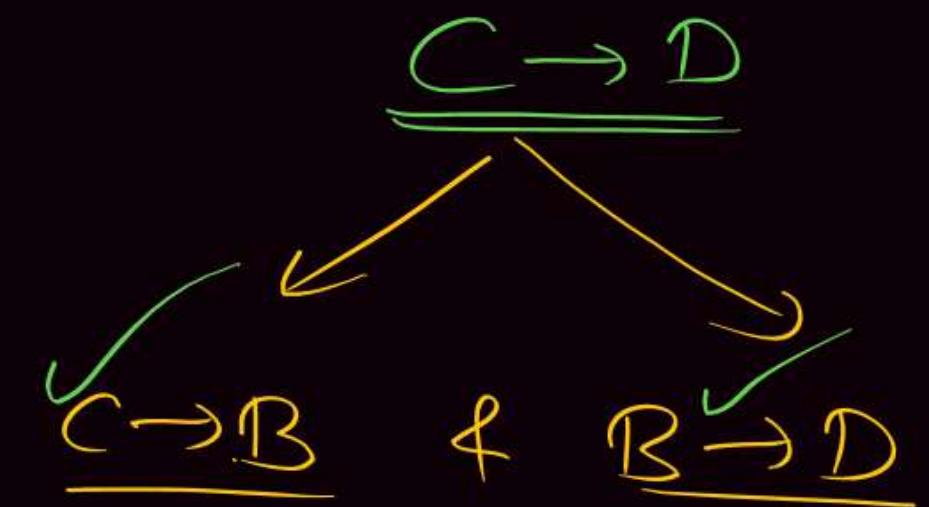
$C \rightarrow B$

$(C)^+ = [CD\underline{B}]$

$(B)^+ = [B\underline{C}D]$



Via



Dependency Preserving

$(C)^+ = [CD\underline{B}]$

$(B)^+ = [B\underline{C}D]$

$C \rightarrow D$

$C \rightarrow B$ in R_2

$B \rightarrow D$ in R_3

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

$R_1(AB) \neq R_2(CD)$

$R_1(AB)$	$R_2(CD)$
$\checkmark A \rightarrow B$	$C \rightarrow D \checkmark$
(F_1)	(F_2)
$[A]^+ = [AB]$	
$[B]^+ = [B]$	
$[C]^+ = [CD]$	
$[D]^+ = [D]$	

$$F_1 \cup F_2 = F$$

Dependency
Preserving

① find the Non Trivial FD of
each Sub Relation
(Domain)
& take a closure of each
Attribute

$$F_1 \cup F_2 \cup F_3 \dots \cup F_n = F$$

Dependency
Preserving

Let $R(A, B, C, D, E)$ be a relational schema with the following function dependencies:

$A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$ and $D \rightarrow BE$.

Decomposed into $R_1(AB)$ $R_2(BC)$ $R_3(CD)$ and $R_4(DE)$

$R_1(AB)$	$R_2(BC)$	$R_3(CD)$	$R_4(\underline{DE})$
$\checkmark A \rightarrow B$	$\checkmark B \rightarrow C$ $C \rightarrow B$	$C \rightarrow D$ $D \rightarrow C$	$D \rightarrow F$
$(A)^t = (AB \subset DE)$			

$(B)^t = (BC \subset DE)$

$(C)^t = (CD \subset DE)$

$(D)^t = (DBE \subset C)$

$(E)^t = [E]$

Dependency
Preferring

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

$D \rightarrow BE$

$D \rightarrow F$

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

$D \rightarrow F$, $D \rightarrow B$

$D \rightarrow BE$

$D \rightarrow BE$



Let $R(A, B, C, D)$ be a relational schema with the following function dependencies:

$A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$ and $D \rightarrow B$.

The decomposition of R into (A, B) , (B, C) , (B, D)

[MCQ: 2M]

$$(R)^+=\{BCD\}$$

Gives a lossless join, and is dependency preserving

$$R_1(ABC) \cap R_3(BD) \Rightarrow (R)^+=\{BCD\} \text{ Subrelay of } R_3$$

Gives a lossless join, but is not dependency preserving

Does not give a lossless join, but is dependency preserving

$$R_{123}(ABCD)$$

Lossless Join

Does not give a lossless join and is not dependency preserving

Q.

Let $R(A, B, C, D)$ be a relational schema with the following function dependencies:

$A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$ and $D \rightarrow B$.

The decomposition of R into (A, B) , (B, C) , (B, D)

[MCQ: 2M]



A Gives a lossless join, and is dependency preserving



B Gives a lossless join, but is not dependency preserving



C Does not give a lossless join, but is dependency preserving



D Does not give a lossless join and is not dependency preserving

Normal Forms

Used to find degree of redundancy

- ❑ BCNF relations have 0% redundancy over FD's whereas 4NF relations have 0% redundancy over FD and MVD.
- ❑ To Eliminate redundancy over Non-Trivial FD, relation should decompose into BCNF but BCNF may not avoid the redundancy due to MVD.
- ❑ To eliminate redundancy over non-trivial MVD, relation should decompose into 4NF (4th Normal Form).

Single Value FD $[x \rightarrow y]$

If $t_1.x = t_2.x$ then $t_1.y = t_2.y$ must be same

Multivalued FD $[x \rightarrow \rightarrow y]$

	\otimes	\otimes	\otimes
	Sid	Sname	Course
	t_1	A/B	C/JAVA
t_2	t_1	A	C
t_3	t_1	A	JAVA
t_4	t_1	B	C
	t_5	B	JAVA

If $t_1.x = t_2.x = t_3.x = t_4.x$

&&

$t_1.y = t_2.y$ && $t_3.y = t_4.y$

&&

$t_1.z = t_3.z$ && $t_2.z = t_4.z$

Normal Forms

Used to find degree of redundancy

Redundancy in relation because of



Non-Trivial FD

$$X \rightarrow Y$$

(Single Value Dependency)

Non-Trivial MVD

$$X \rightarrow \rightarrow Y$$

(Multivalued Dependency)

Normal Forms

- ❑ BCNF relations have 0% redundancy over FD's whereas 4NF relations have 0% redundancy over FD and MVD.
- ❑ To Eliminate redundancy over Non-Trivial FD, relation should decompose into BCNF but BCNF may not avoid the redundancy due to MVD.
- ❑ To eliminate redundancy over non-trivial MVD, relation should decompose into 4NF (4th Normal Form).

Normal Form

1NF → Atomic

2NF

3NF

BCNF

Every Higher NF Contain the lower NF.

First Normal Form

- Default NF of RDBMS relations.
- Relation (DB Table) R is in 1 NF if and only if no multivalued attributes in R. [Every attribute of R must be atomic/single valued].

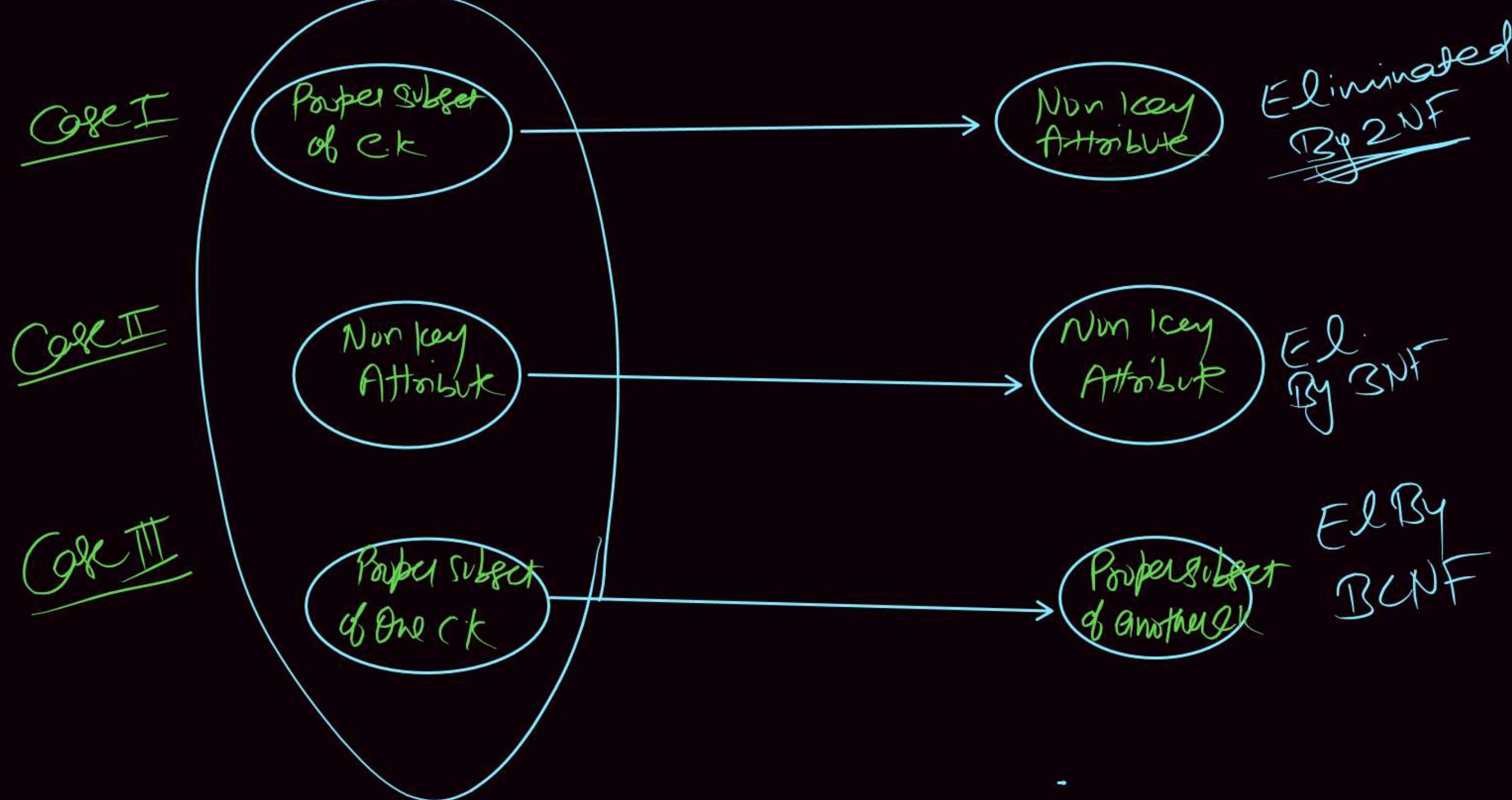
NOTE:

Degree of redundancy is very high in 1NF relation.

Level

1NF > 2NF > 3NF > BCNF

Possible Non Trivial FD which Create Redundancy



Case I

Proper subset
of $C \cup C$

Non key
attribute

$R(ABCDE)$ $\{AB \rightarrow C, C \rightarrow D, B \rightarrow E\}$

Candidate key = (AB)

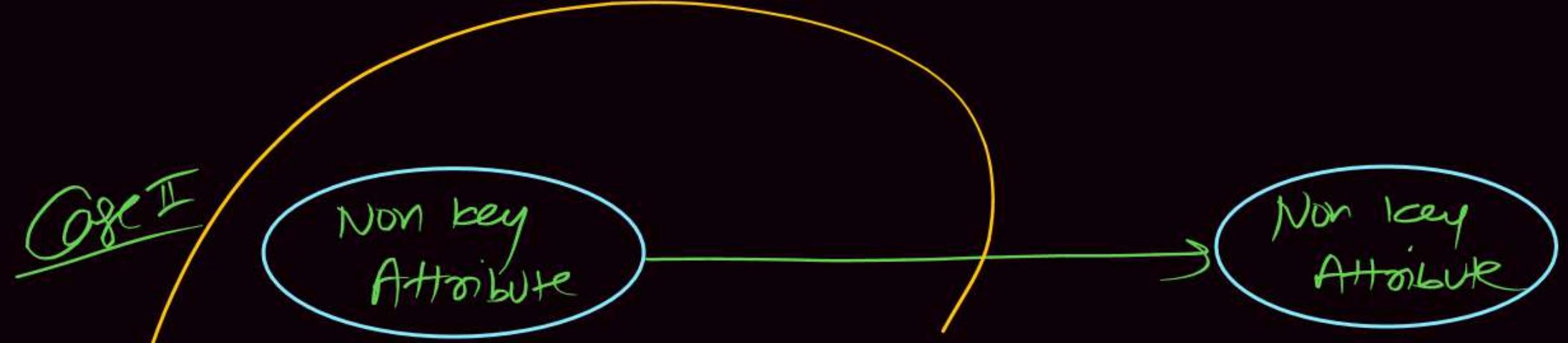
Non key Attribute = (C, D, E)

$B \rightarrow E$

Partial Dependency

Not in

2NF



$R(ABC) \quad (A \rightarrow B, \quad B \rightarrow C)$

Candidate Key = [A]

Non Prime / Non Key Attribute = [B, C]

$$B \rightarrow C$$

Transitive FD

Case 3

Proper subset
of one CK

Proper subset
of another CK

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

Candidate key = (AB, DB)

$D \rightarrow A$

First Normal Form

Important Point 1

- $x \rightarrow y$ FD forms redundancy in relational schema R if and only if -
 - (i) Non-Trivial FD
and
 - (ii) X is not super key.
- $x \rightarrow y$ FD not forms any redundancy in relation R if and only if-
 - (i) Trivial FD [$x \supseteq y$] or Semi-trivial
or
 - (ii) x : super key

Second Normal Form (2NF)

Relational Schema R is in 2 NF iff

- R should be 1NF
- R should not contain any partial dependency

Partial Dependency

Let R be the relational schema and x, y, z are non-empty set of attributes.

X : Any candidate key of R.

Y : Non-prime attribute

Z : Proper subset of candidate key

∴ ~~$z \rightarrow y$~~ is said to be partial dependency iff-

- z is proper subset of candidate key
- y should be non-prime attribute.



Second Normal Form (2NF)



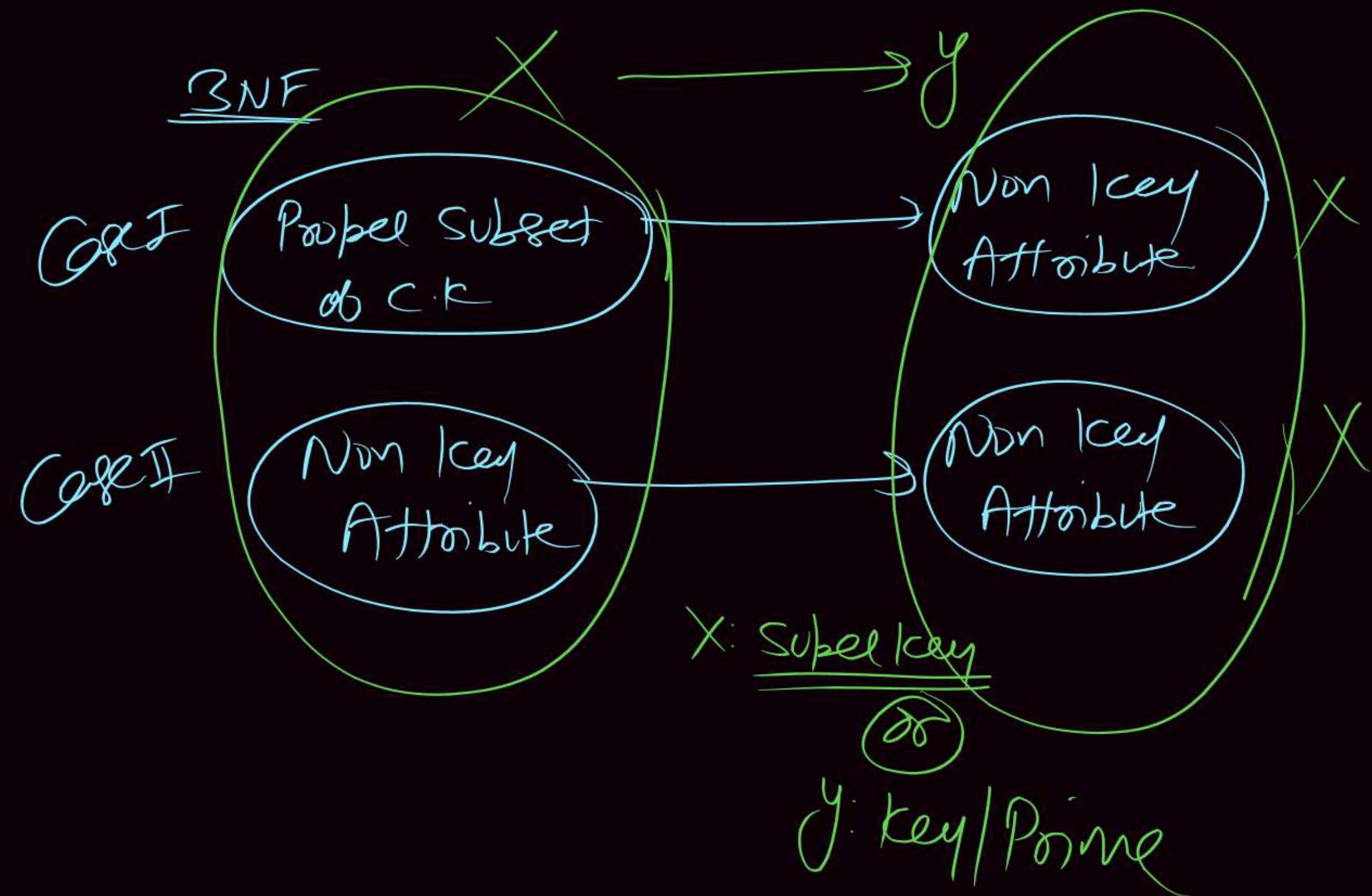
Important Point 2

[Proper Subset of Candidate Key] → [Non-prime attribute]



Non super key

The above FD forms redundancy in R.



Third Normal Form (3NF)

Relational schema R is in 3NF iff every non-trivial FD $x \rightarrow y$ in R with-

(i) x must be super key (SK)

or

(ii) y must be prime attribute.

$\therefore \{ \text{SK} \rightarrow \text{Prime/Non-Prime}, (\text{Not SK}) \rightarrow \text{Prime attribute} \}$

Important Point 3

3NF allow FD set:

[Proper subset of candidate key] \rightarrow [Proper subset of other candidate key]

which forms redundancy in R.

$R(ABCD) \quad \overbrace{\{AB \rightarrow CD, D \rightarrow A\}}$

Candidate key = (AB, DB)

$\overbrace{(AB)} \rightarrow CD$

$D \rightarrow A$; But A is key / Prime Attribute

AB super key

Dis not super key

Boyce Codd Normal Form (BCNF)

Relational schema R is in BCNF iff every non-trivial FD “ $x \rightarrow y$ ” with x must be super key.

∴ Prime/ Non-prime attributes must be determined by super key.

Important Point 4

If R is in 3NF but not BCNF then

[Proper subset of candidate key] \rightarrow [Proper subset of other candidate key]
must exist in R.

3NF

$X \rightarrow Y$: Non Trivial FD
every

X : Super key

or

Y : key / Prime AttribUR

BCNF

every Non Trivial FD

$X \rightarrow Y$

X : Super key

Boyce Codd Normal Form (BCNF)



Important Point 5

If relational schema are with only simple candidate key then R always in:

- I. 2NF
- II. May or may not 3NF/BCNF

Reason : [Proper subset of candidate key] \rightarrow [Non-prime attribute]

From the above statement, we can conclude that “partial dependency” not possible if all CK’s are simple candidate key.

$R(AB\text{CD}E)$ $[A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow E, C \rightarrow A]$

~~2NF BKT~~
~~Not in 3NF~~
Candidate Key = (A, B, C)
~~N.K~~ ~~$D \rightarrow E$~~ ~~N.K~~

Boyce Codd Normal Form (BCNF)

Important Point 6

If relational schema R is with only prime attribute (No non-prime attribute in R) then R always in:

- I. 3NF
- II. May or may not BCNF

Important Point 7

If relational schema R with no non-trivial FD's then R always in BCNF.

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

$$\text{Candidate Key} = (A D, B D, C D)$$

$$\text{Prime Attribute} = (A, B, C, D)$$

R is 3NF But Not in BCNF

$$(X) \rightarrow Y$$

X: Not Subkey

Q

R(ABCDEFGHIJ) {AB → C, A → DE, B → F, F → GH, D → IJ}

P
W

Candidate key = (AB)

Nonkey Attribute = (C, D, E, F, G, H, I, J)

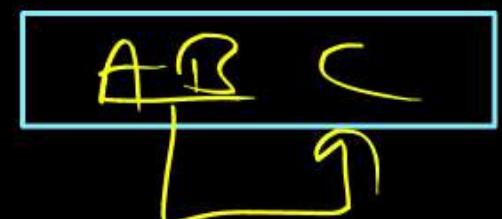
Check 2NF?

$\overbrace{A \rightarrow DE, B \rightarrow F}^{\text{PD}}$ Not in 2NF.

2NF Decomposition

(A)^t - (ADEIJ)

(B)^t - (BFGH)



Q

R (~~ABCDEFHIJ~~) {AB→C, BD→EF, AD→GH, A→I, H→J}

Candidate key = (ABD)

Check 2NF ? AB→C . BD→EF . AD→GH . A→I

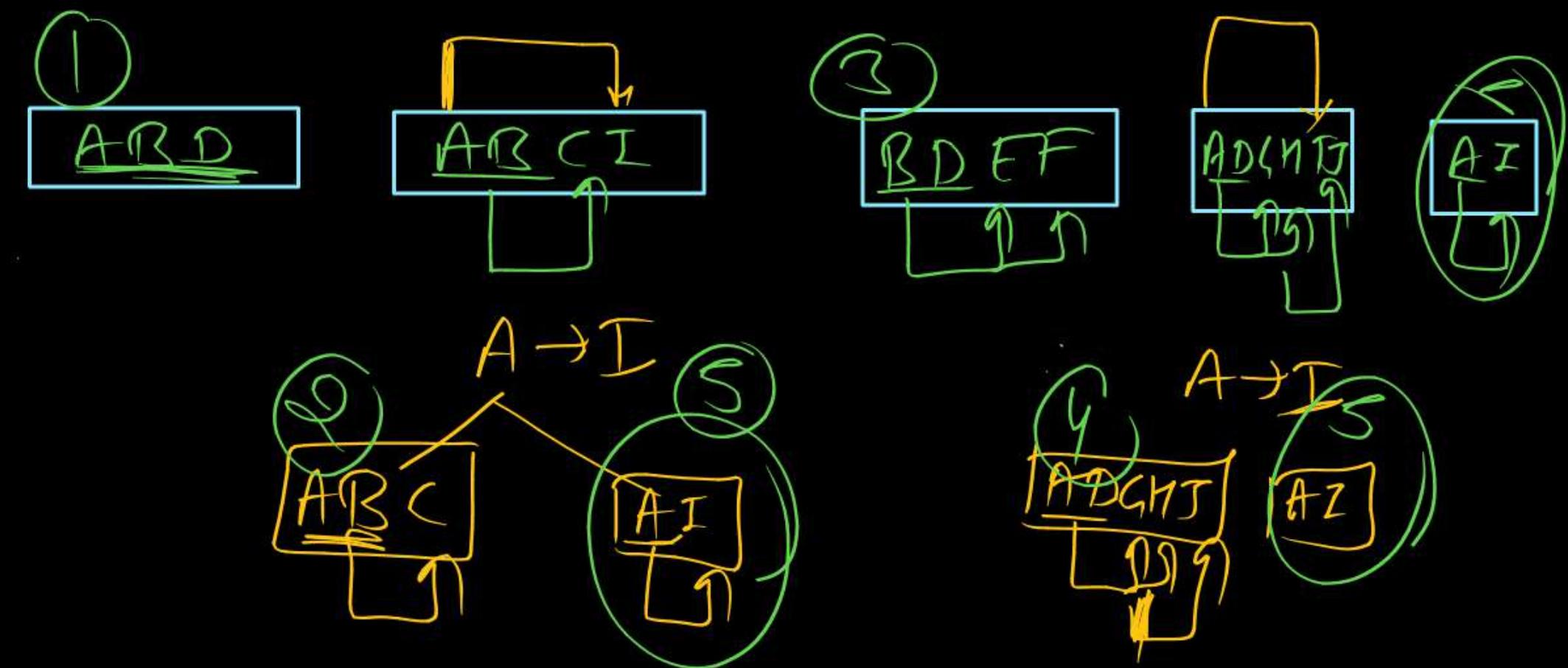
2NF Decomposition

(AB)^t = (ABCDF)

(BD)^t = (BDEF)

(AD)^t = (ADGHI)

(A)^t = (AI)



2NF Decomposition

3NF Decomposition

N.IC \rightarrow N.IC

BCNF Decomposition

$R(ABCDE)$ ($A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$, $D \rightarrow E$)

Candidate key = $\underline{[A]}$

BCNF Decomposition

$R(ABC \not\perp \not\perp DE)$

① $B \rightarrow C$

$C \rightarrow D$

② $D \rightarrow E$

$R_2(BC)$

$R_3(DE)$

$R_1(ABD)$
 $R_2(BC)$
 $R_3(DE)$

② $B \rightarrow C$

① $C \rightarrow D$

$D \rightarrow E$

$R_1(A \not\perp \not\perp E)$

$R_2(CD)$

$R_3(BC)$

BCNF Decomposition

$R_1(\overline{ABE})$
 $R_2(\overline{CD})$
 $R_3(\overline{BC})$

③ $B \rightarrow C$

② $C \rightarrow D$

$R_1(AB) R_2(BI)$
 \oplus
 $R_3(C) R_4(DF)$

$R_1(A \not\perp \not\perp E)$

$R_2(DE)$

$R_3(CD)$

$R_4(BC)$



R (ABCDEFGHIJ) {AB→C, BD→EF, AD→GH, A→I, H→J}

Database design Table



DB design goal	1NF	2NF	3NF	BCNF
1. Loss-less join decomposition	Yes ✓	Yes ✓	Yes ✓	Yes ✓
2. Dependency preserving decomposition	Yes ✓	Yes ✓	Yes ✓	May not
3. 0% redundancy	No	No	No	Yes [Over MVFD]

Database design Table

- Every relation possible to decompose into 1NF, 2NF, 3NF, BCNF with loss-less join decomposition.
- Every relation possible to decompose into 1NF, 2NF, 3NF with Dependency preserving decomposition.
- Not every relation can decompose into BCNF and 4NF with dependency preserving decomposition.
- Most accurate normal form to design simple database is 3NF because every relation is possible to decompose into 3NF with loss-less join and dependency preserving.

2. ER MODEL

Introduction



Entity relationship diagram used to represent diagrammatic design
(High level design) of DB.

Main components in ER Diagram

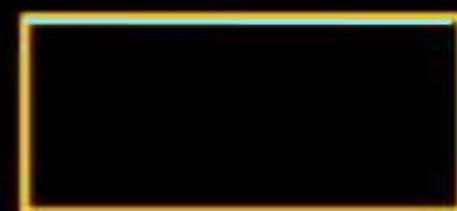
- (i) Attributes
- (ii) Entity sets
- (iii) Relationship sets

Main components in ER Diagram

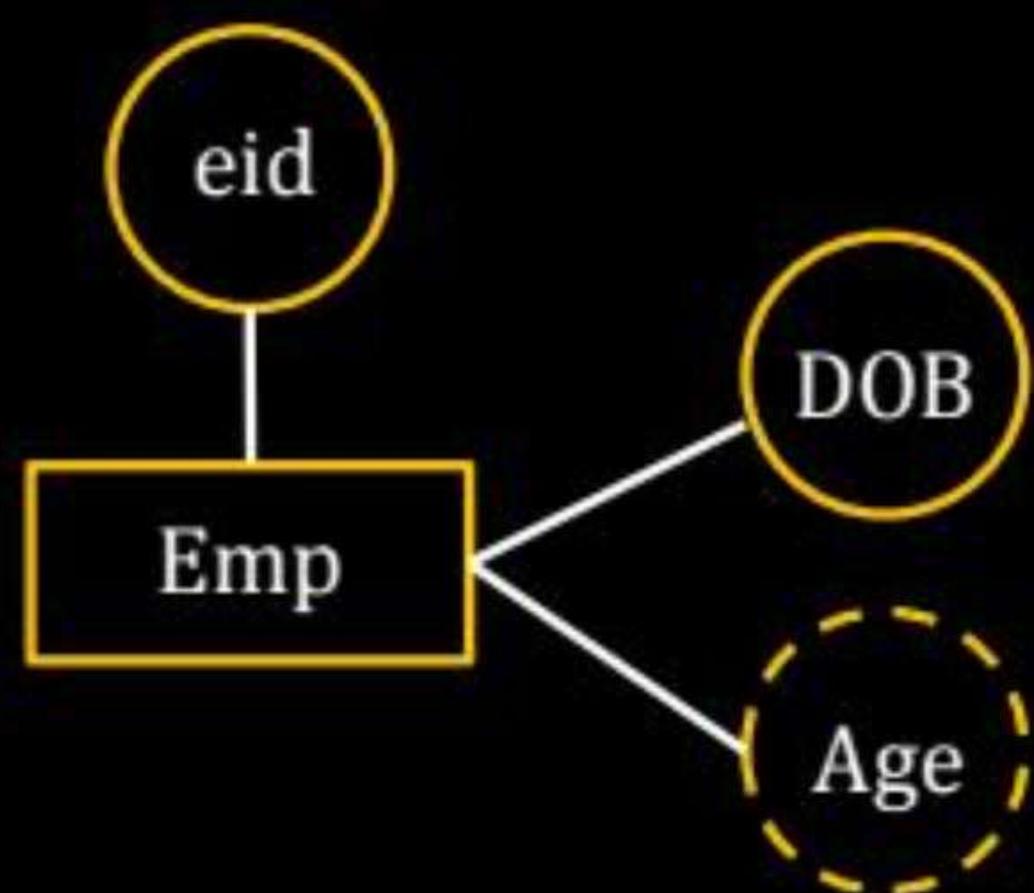
(a) Entity sets

It is a set of entities of the same type denoted by a rectangular box in ER diagram. Entity can be identified by a list of attributes which are placed in ovals.

Represented By:



Example:



Main components in ER Diagram

(b) Attributes

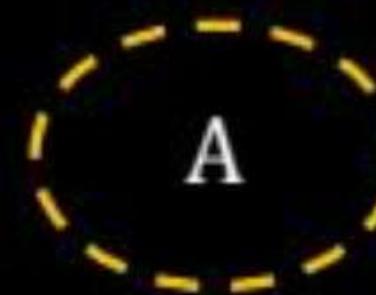
(i) Attribute



(ii) Key attribute

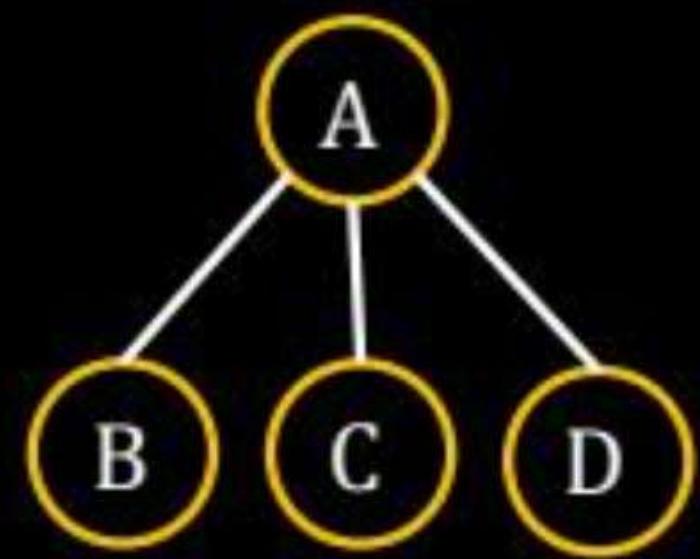


(iii) Derived attribute



D.O.B Age

(iv) Composite Attribute: Attribute
which can be represented as
two or more attributes.



(v) Multivalued Attribute:

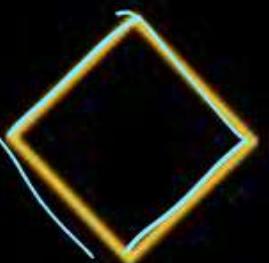


Main components in ER Diagram

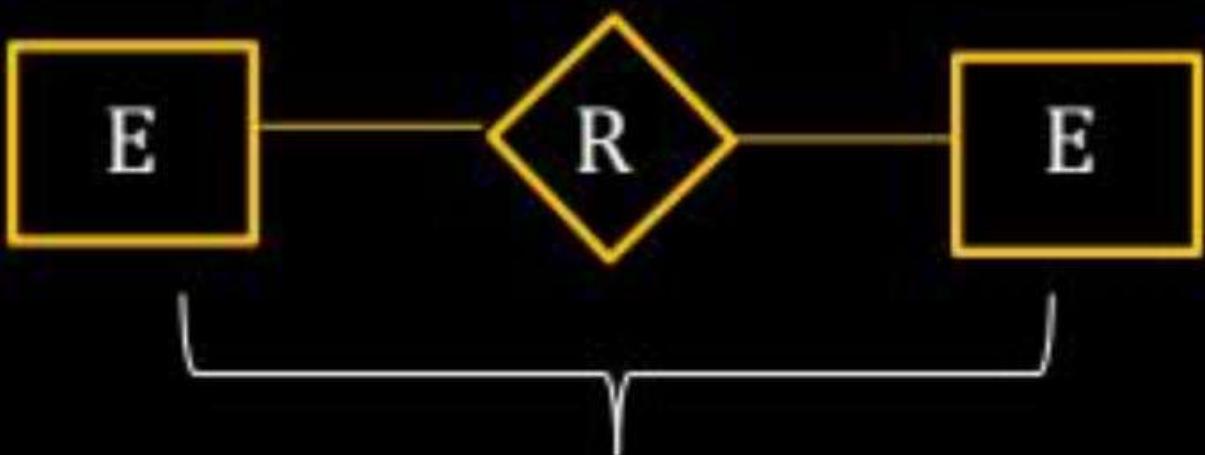
(c) Relationship Set:

It is used to relate two or more entity set.

Represented by:



Example:

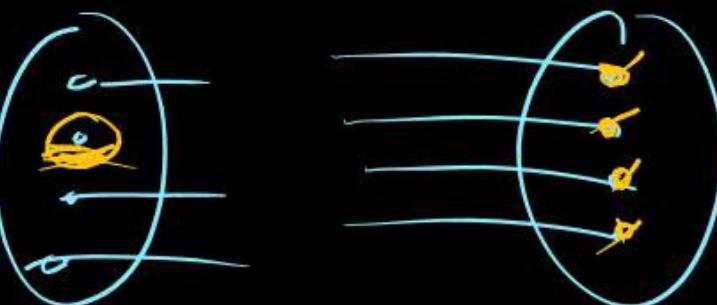


Binary Relationship

Participation

- If every entities of entity set are participated with relationship set then it is total participation (100% participation) otherwise it will be partial participation (< 100% participation)

Example : Consider Emp and Dept entity set.



Manages relationship set such that each dept must have manager.

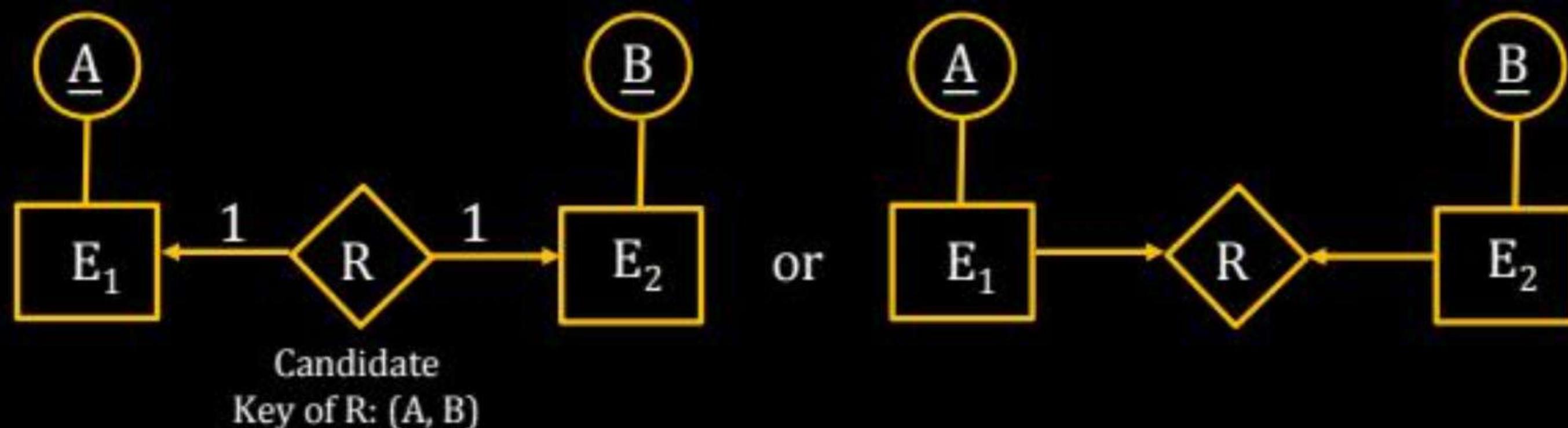


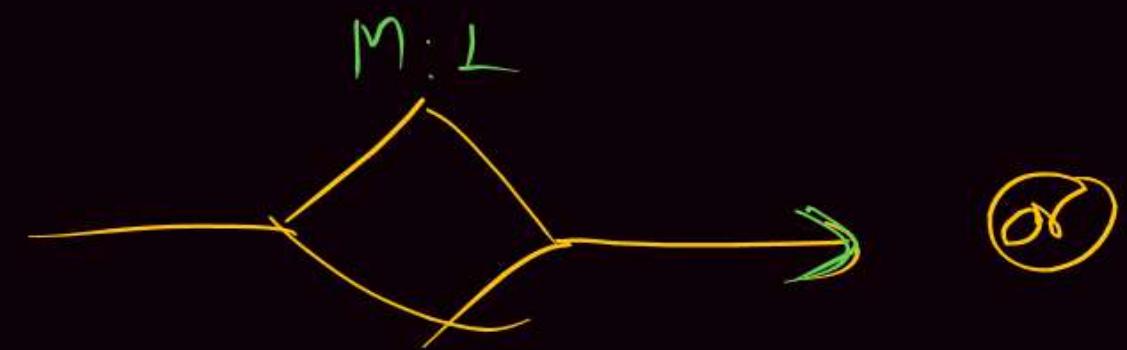
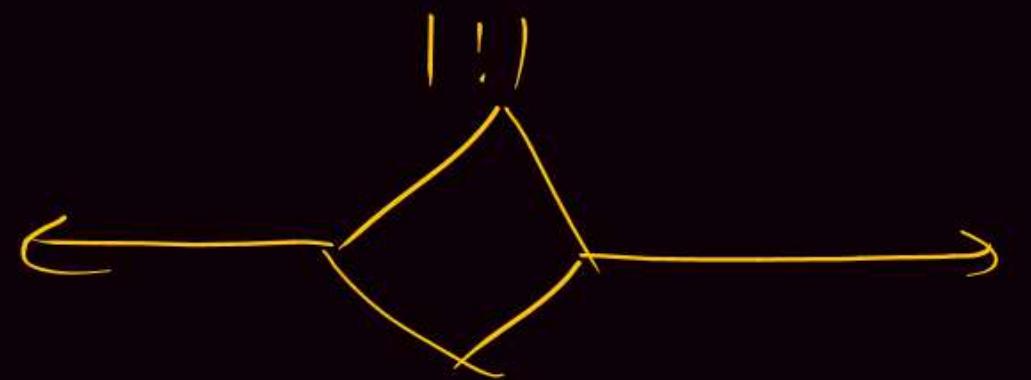
Mapping [Cardinality constraints of relationship set]

One mapping : At most one (0 or 1)

Many mapping : 0 or more (0 *)

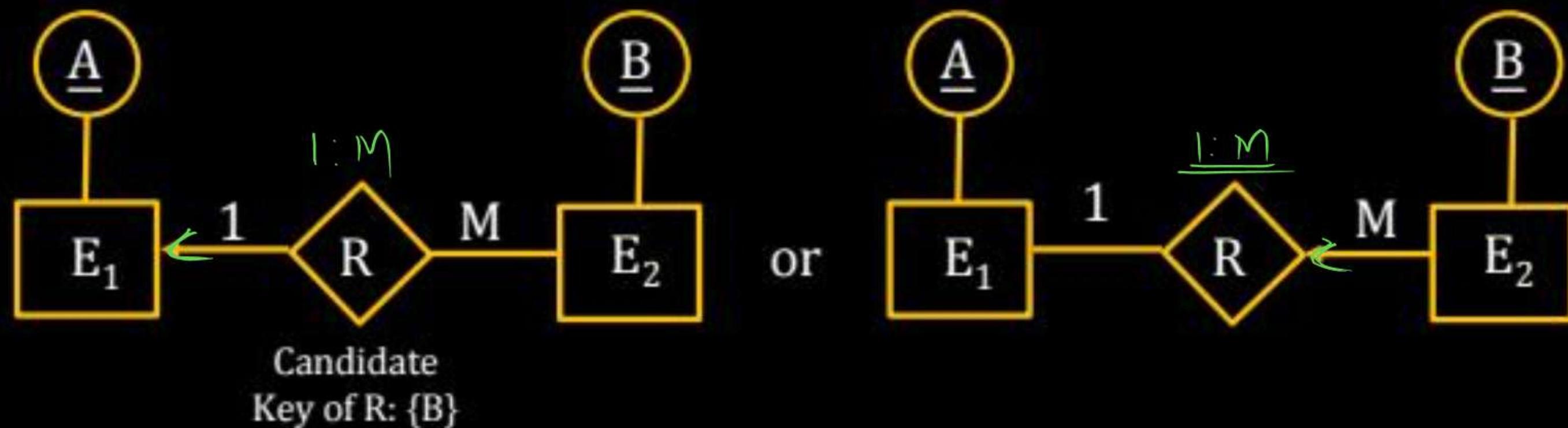
Binary Relationship Mapping (One : One)





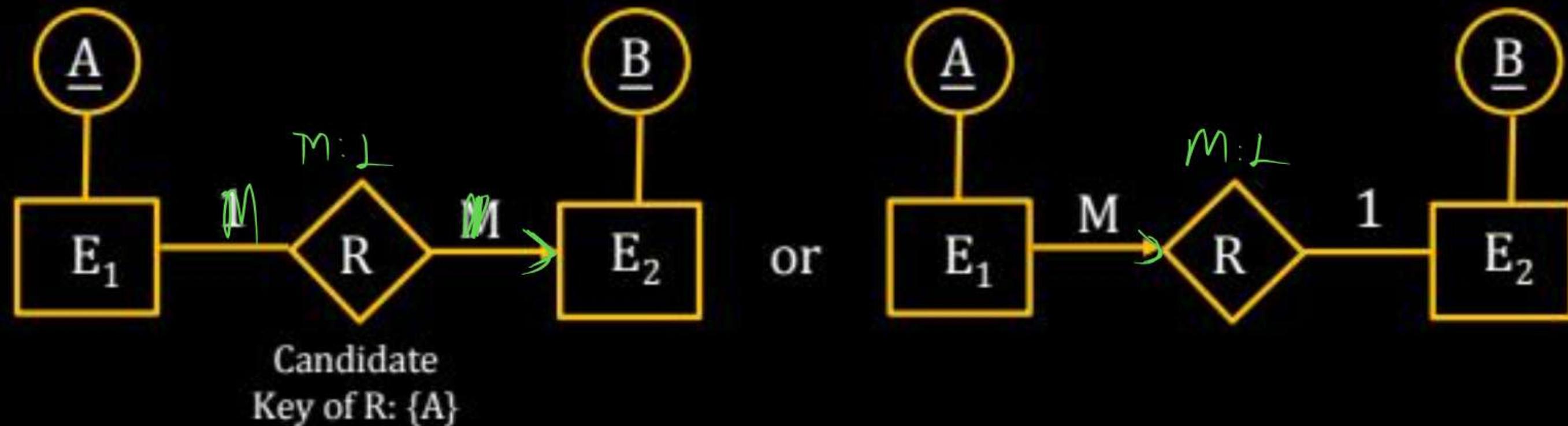
Mapping [Cardinality constraints of relationship set]

Binary Relationship Mapping (One : Many)



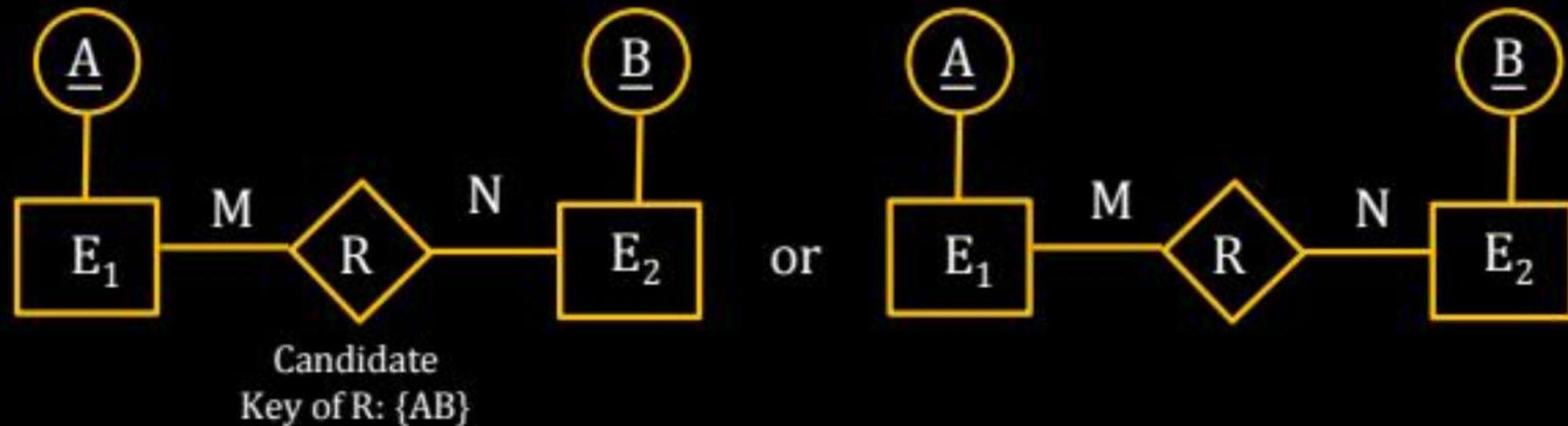
Mapping [Cardinality constraints of relationship set]

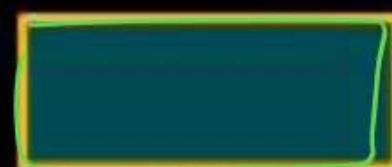
Binary Relationship Mapping (Many to One)



Mapping [Cardinality constraints of relationship set]

Binary Relationship Mapping (Many to Many)



Symbol**Meaning**

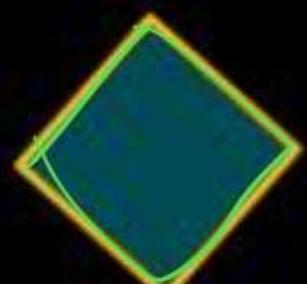
Entity



Multivalued
Attribute



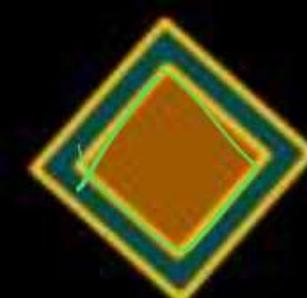
Weak Entity



Relationship



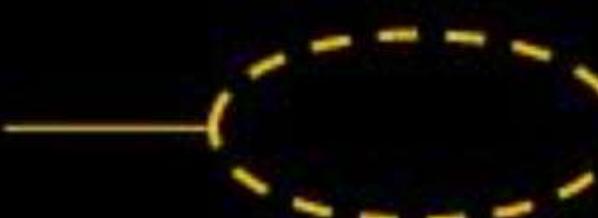
Composite
Attribute



Identifying Relationship



Attribute



Derived
Attribute



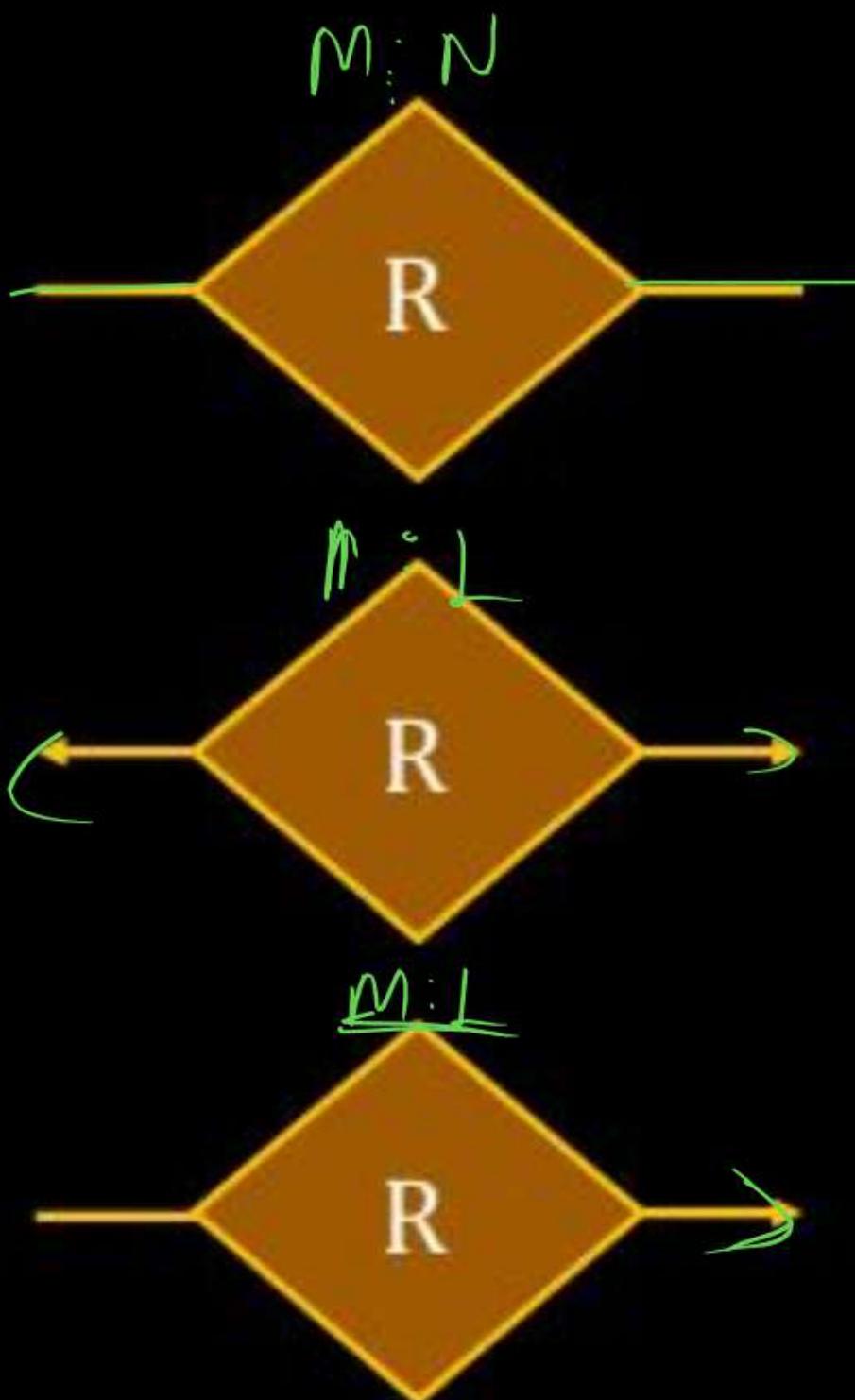
Key Attribute

Symbol**Meaning**

Total Participation of E₂ in R



Cardinality Ratio 1:N for E₁:E₂ in R



Many - to- Many relationship

One - to - One relationship

Many - to - One relationship

ER to Relational Model Conversion

Entity Set → Relation (Table)

Composite → Single Attribute

Multivalued Attribute → Separate Table
(Key + All Multivalued Attributes)

Q.

P
W

The term in list A have been mapped to list B so that is corresponds to the mapping process of ER MODEL into relational. Which of the following represent the mapping process?

[MCQ]

List-A	List-B
A. <u>Entity type</u>	1. Primary key (or alternate key)
B. Key attributes	2. Child table
C. <u>Composite attribute</u>	3. <u>Set of simple component attributes</u>
D. Multivalued attribute	4. Relation

A

A-3, B-1, C-4, D-2

C

A-3, B-2, C-2, D-4

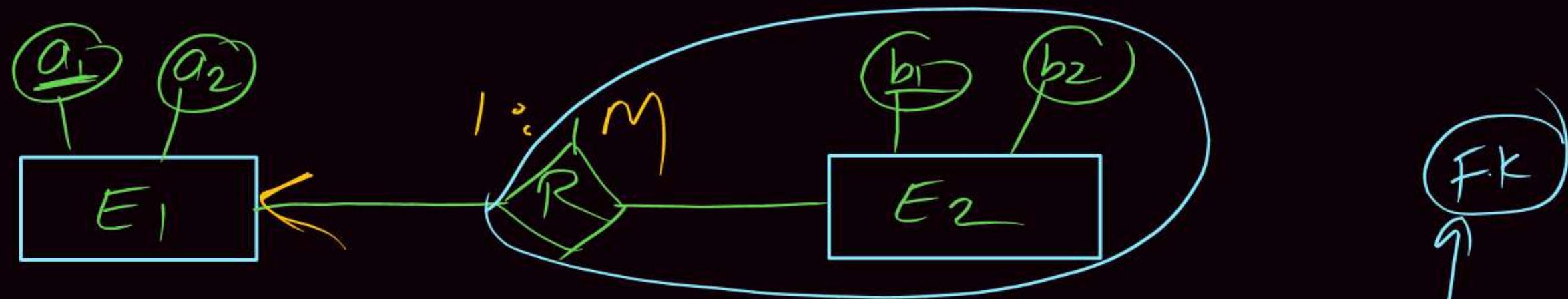
B

A-4, B-1, C-3, D-2

D

A-4, B-1, C-2, D-3

①

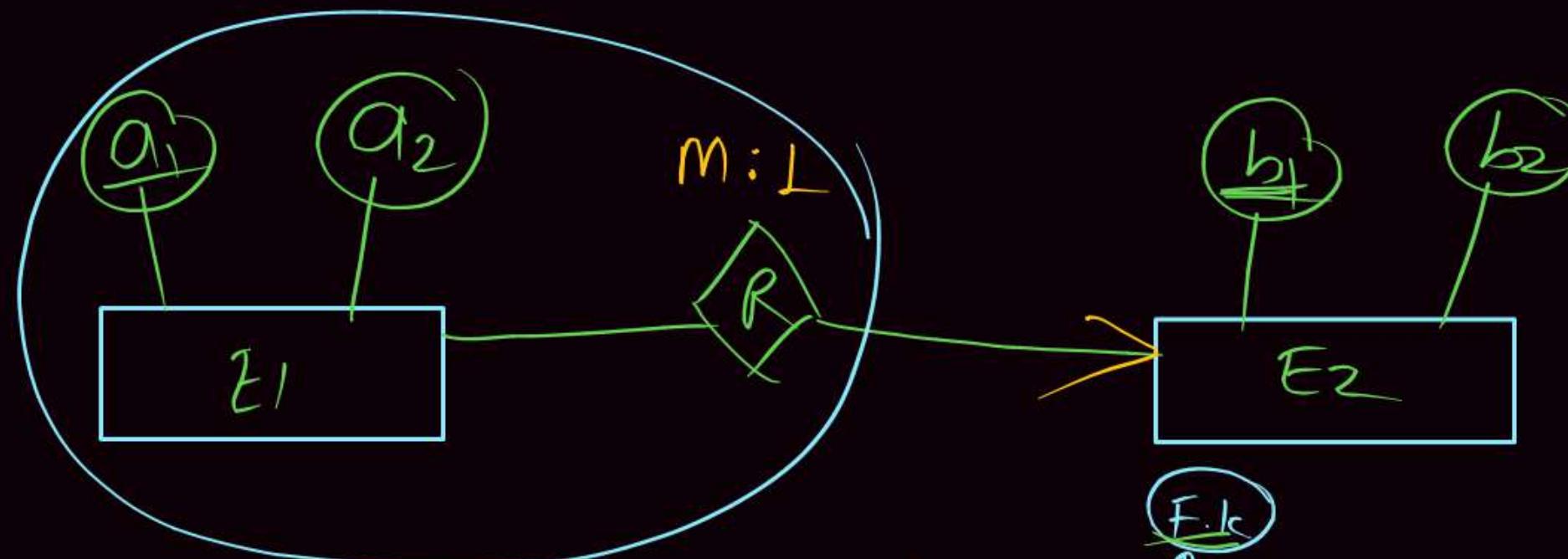


2 Table

$E_1(a_1, a_2)$

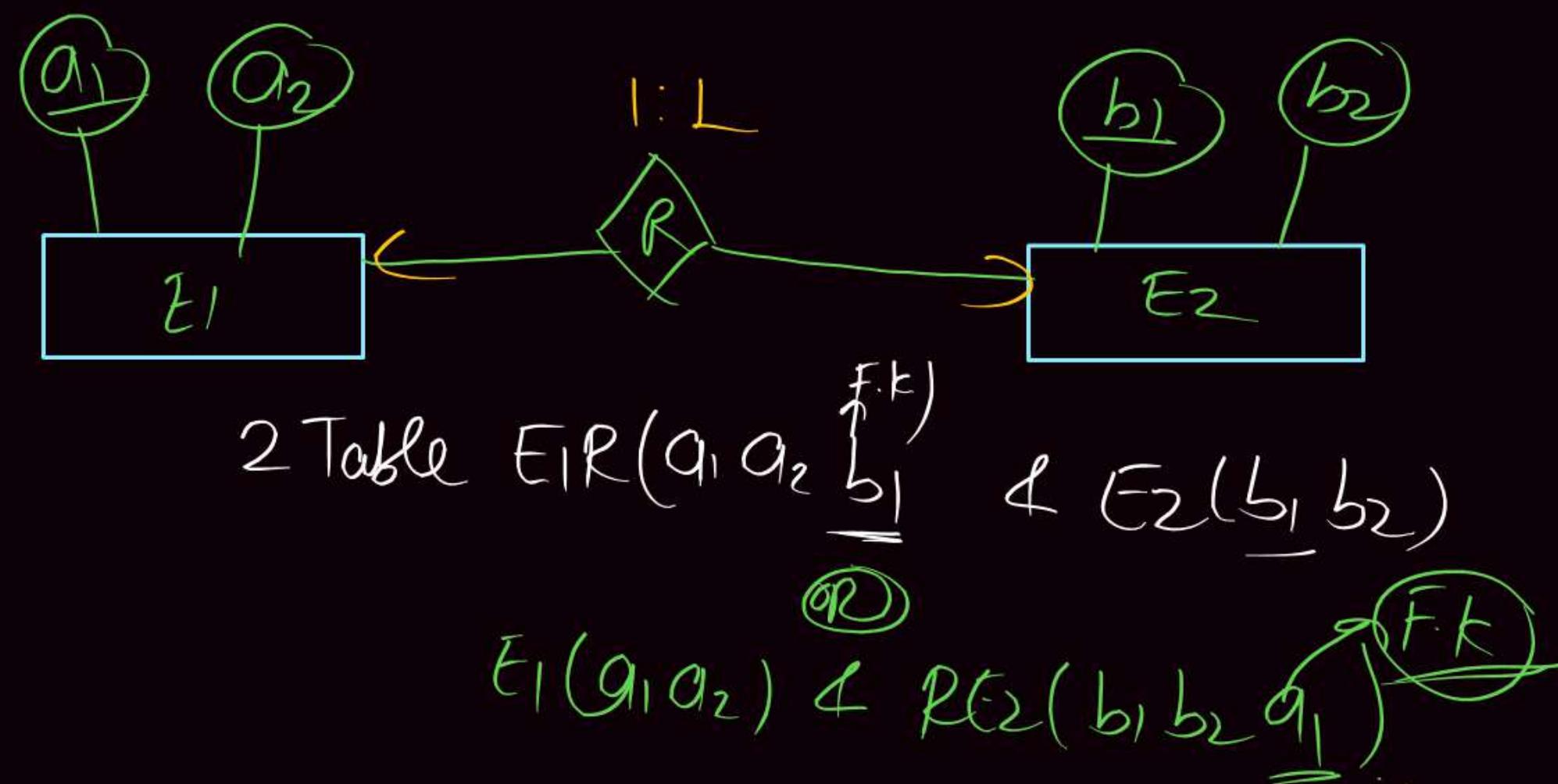
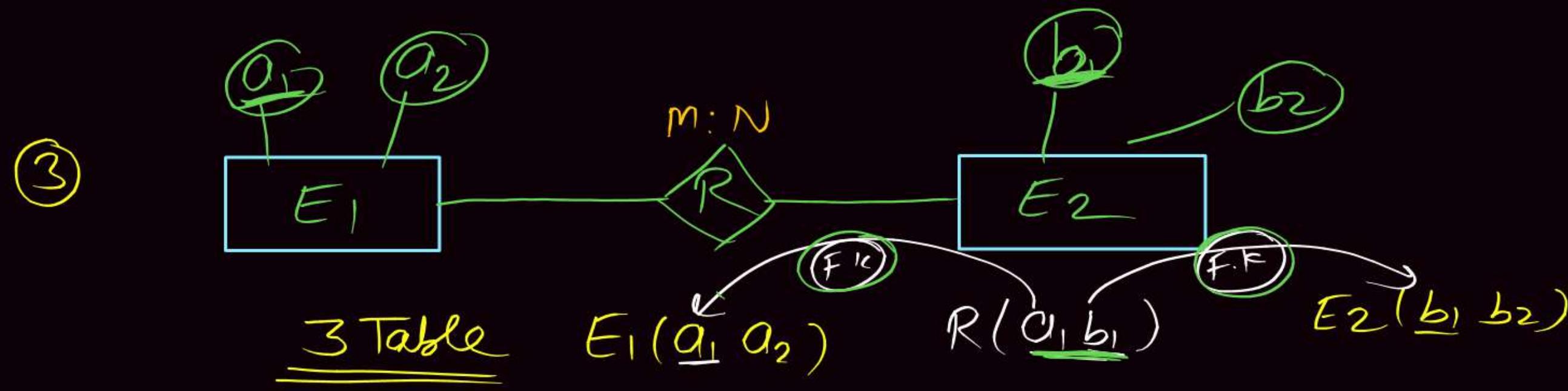
$RE_2(b_1, b_2, a_1)$

②



2 Table

$E_1R(a_1, a_2, \underline{b}_1) \& E_2(\underline{b}_1, b_2)$



Mapping [Cardinality constraints of relationship set]

(For binary relationship)

Partial participation on both side of binary relationship

- One to Many : Merge relationship set towards many side. So, 2 relational tables.
- Many to one : Merge relationship set towards many side. So, 2 relational tables.
- One to one : Merge relationship set any one side. So, 2 relational tables.
- Many to Many : Separate table for each entity set and relationship set. so, 3 relational tables.

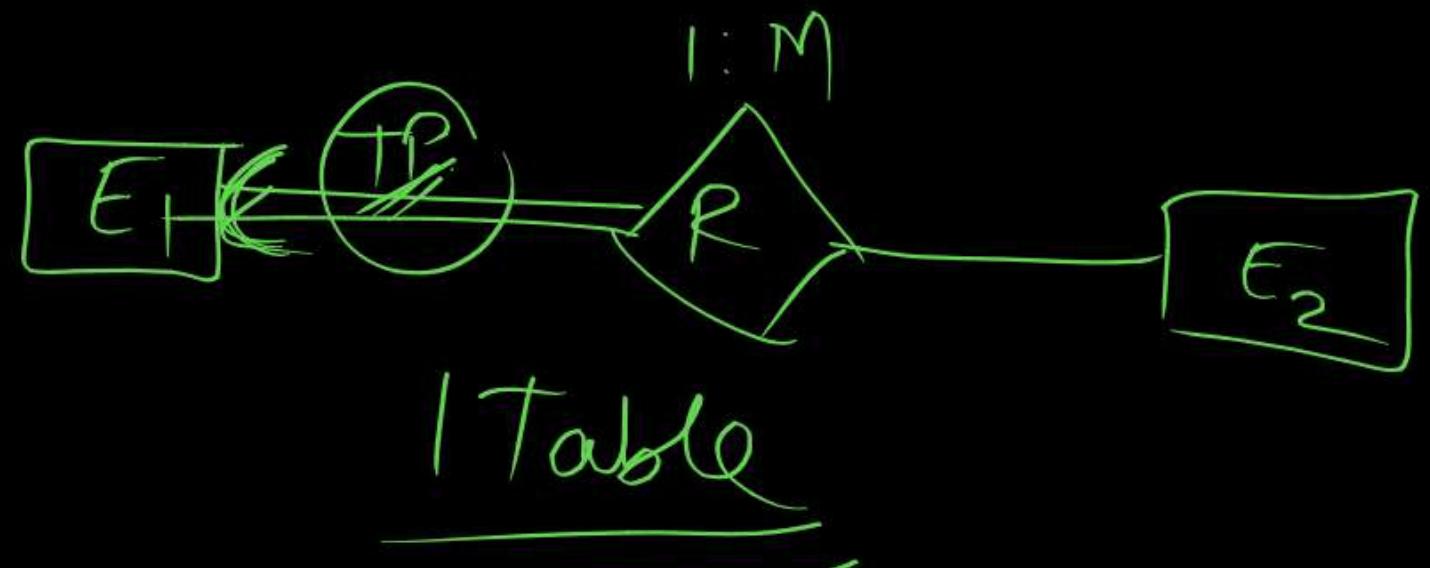
Mapping [Cardinality constraints of relationship set]

(For binary relationship)

Full / Total Participation

Full participation on “one” side of many to one relationship

Merge the entities and relationship set into single relational table. So, 1 table.



Mapping [Cardinality constraints of relationship set]

(For binary relationship)

Full participation on “Many” side of Many-to-one relationship

Merge relationship set towards many side. So, 2 relational tables.

Mapping [Cardinality constraints of relationship set]

(For binary relationship)

Full participation on any “one” side in one-to-one relationship

Merge the entity sets and relationship set into single table. So, 1 table.



Mapping [Cardinality constraints of relationship set]

(For binary relationship)

Full participation on any “Many” side of Many-to-Many relationship

Merge relationship set towards any “Many” side of relationship. So, 2 table.

Mapping [Cardinality constraints of relationship set]

(For binary relationship)

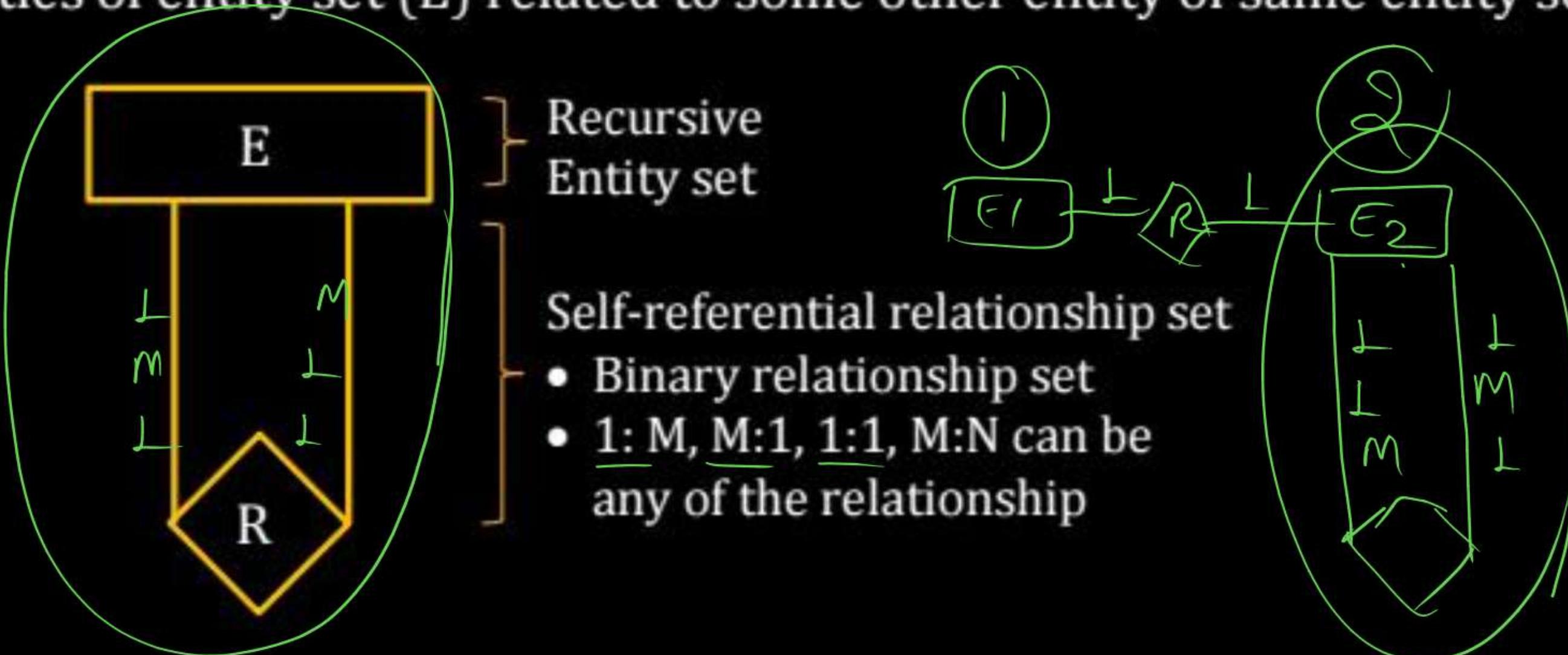
Full participation on both side of relationship

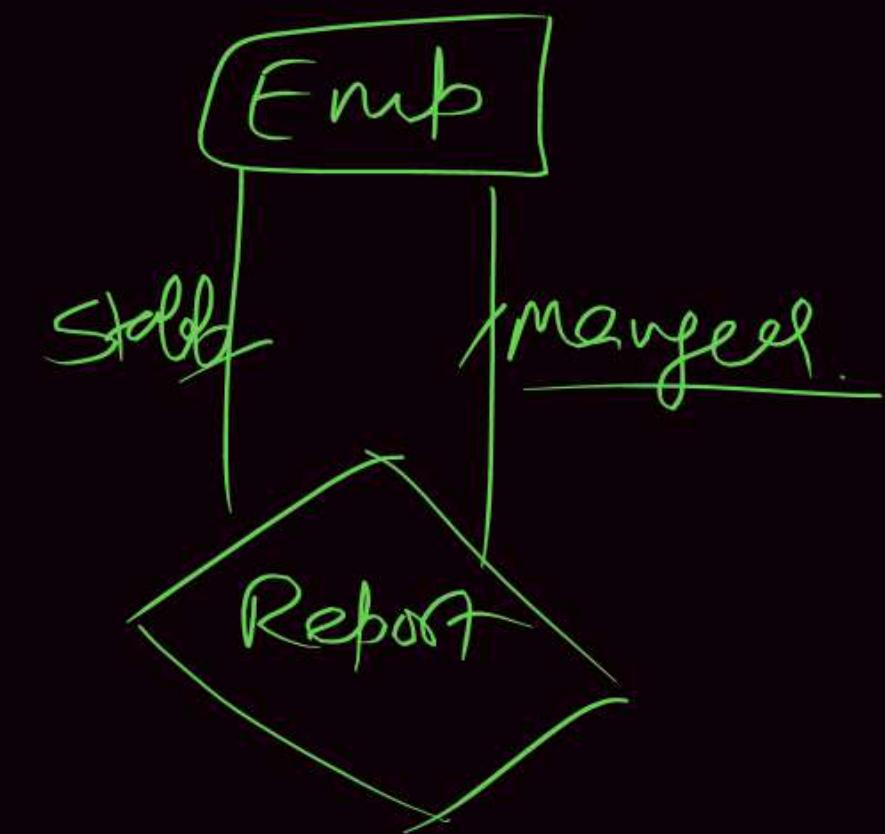
<u>1 : 1</u>	Merge the entity sets and Relationship into single Relational <u>table</u> so, <u>1 relational table</u> .
<u>1 : M</u>	
<u>M : 1</u>	
<u>M : N</u>	

Self-Referential Relationship Set

(Recursive entity set)

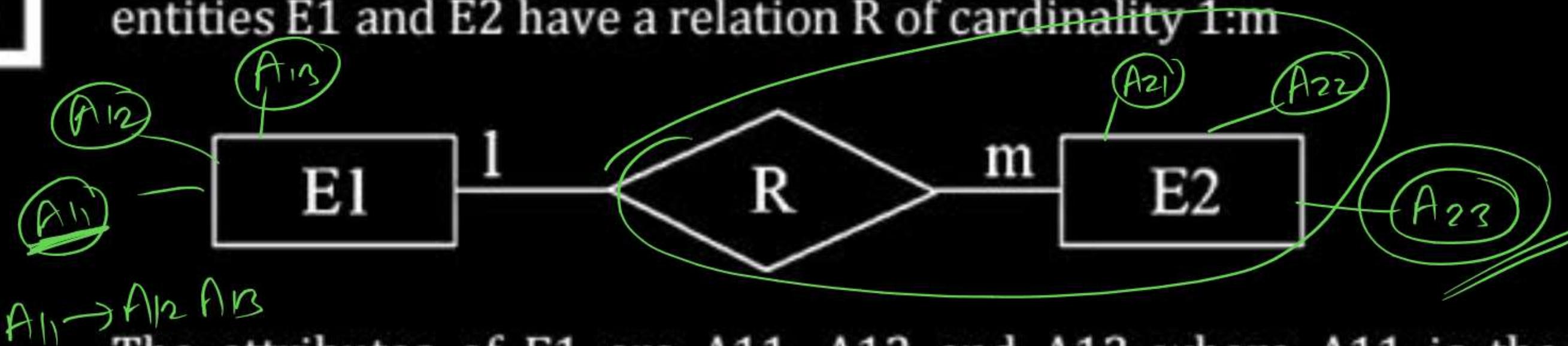
Entities of entity set (E) related to some other entity of same entity set (E).





Q.

Consider the following entity relationship diagram(ERD), where two entities E1 and E2 have a relation R of cardinality 1:m



$$A_{11} \rightarrow A_{12} A_{13}$$

The attributes of E1 are A11, A12 and A13 where A11 is the key attribute. The attributes of E2 are A21, A22, A23 where A21 is the key attribute and A23 is a multi-valued attribute. Relation R does not have any attribute. A relational database containing minimum number of tables with each tables satisfying the requirements of the third normal form (3NF) is designed from the above ERD. The number of tables in the database is

[GATE-2004 : 2 Marks]

$E_1 (A_{11} A_{12} A_{13})$

$R E_2 (A_{21} A_{22} A_{11})$

$E_2' (A_{21} A_{23})$

A

2

B

3

C

5

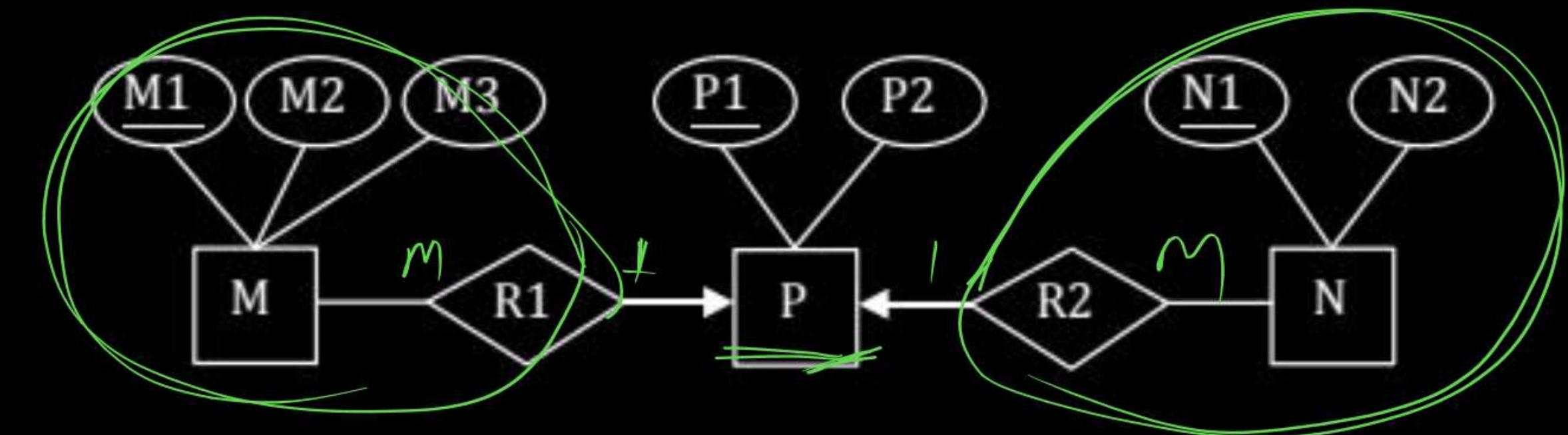
D

4

Q.

Common Data for Question

Consider the following ER Diagram



- (i) The minimum number of tables needed to represent M, N
P, R1, R2 is [GATE-2008 : 2 Marks]

A

2

C

4

B

3

D

5

$MR_1(M, M_1, M_2, M_3, P_1)$
 $P(P_1, P_2)$
 $NR_2(N, N_1, N_2, P_1)$

(ii) Which of the following is a correct attribute set for one of the table for the correct answer to the above question?

GATE-2008 : 2 Marks]

- A {M1, M2, M3, P1}
- B {M1, P1, N1, N2}
- C {M1, P1, N1}
- D {M1, P1}

Weak Entity Set and Weak Relationship Set

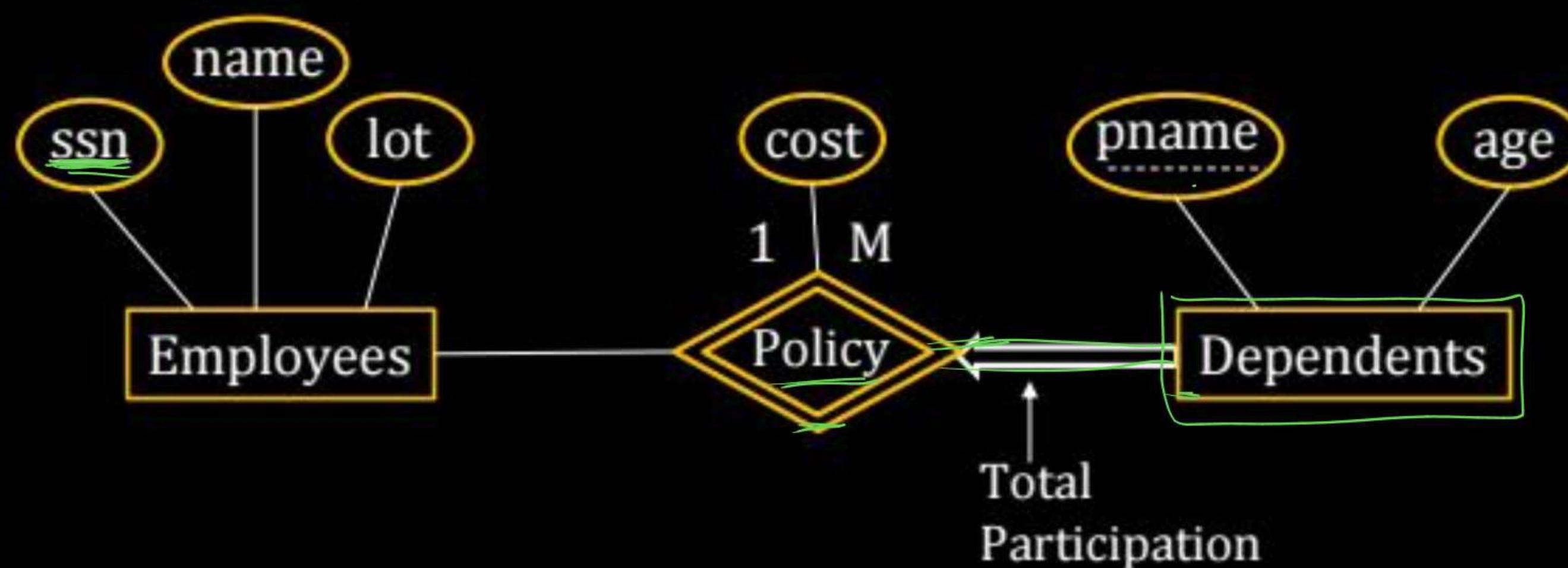
The entity set with no key. (Attributes of weak entity sets are not sufficient to differentiate entities uniquely).



Points:

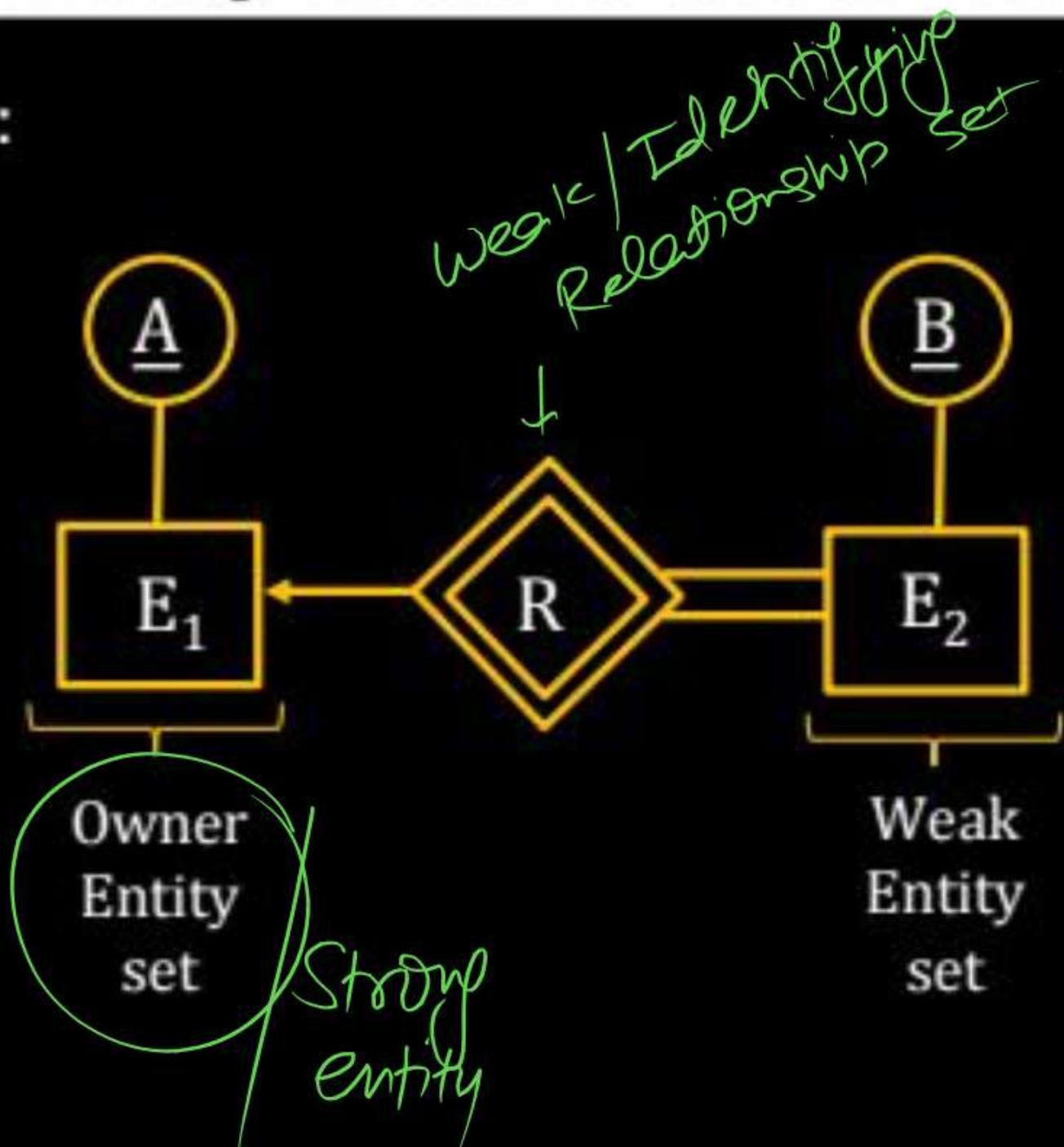
- (a) For each weak entity set there must be owner entity set, which is strong entity set.
- (b) Relationship set between weak entity set and identifier entity set is also "weak relationship set".
- (c) The participation towards weak entity set end must be "total participation".
- (d) The mapping between identifier entity set and weak entity set must be one : many (1 : M)

- ❑ A weak entity can be identified uniquely only by considering the primary key of another (owner) entity.
- ❖ Owner entity set and weak entity set must participate in a one-to-many relationship set (one owner, many weak entities).
- ❖ Weak entity set must have total participation in this identifying relationship set.



Weak Entity Set and Weak Relationship Set

Example:



NOTE:

Weak entity set and multivalued attributes allowed to represent in ER
diagram but not allowed in RDBMS table.

3. RELATION MODEL AND SQL

Procedural Query Language and Non-procedural Query Language

<u>Procedural Query Language</u>	<u>Non-procedural Query Language</u>
<p>Formulation of <u>how</u> to access data from the database table and what data required to retrieve from DB tables.</p> <p>"Relational Algebra"</p>	<p>Formulation of <u>what data retrieve</u> from DB tables.</p> <p><u>SQL &</u> "Relational Calculus"</p>

Relational Algebra



(Always generate distinct tuples)

Relational algebra refers to a procedural query language that takes relation
instances as input and returns relation instances as output

Basic operators

π : Projection operator

σ : Selection operator

\times : Cross-product operator

\cup : Union

$-$: Set difference

ρ : Rename operator

Relational Algebra



Derived operators

\cap : Intersection {using $-$ }

\bowtie : Join {using X, σ }

/ or \div : Division {using $\pi, X, -$ }

$$R \cap S = \{3, 4, 5\}$$

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

$$R = \{1, 2, 3, 4, 5\}$$

$$S = \{3, 4, 5, 6\}$$

$$R - S = \{1, 2\}$$

$$R - (R - S) \rightarrow \{1, 2, 3, 4, 5\} - \{1, 2\}$$

$$= \underline{\{3, 4, 5\}} = \underline{\underline{R \cap S}}$$

Basic operators

$\pi_{\text{Attribute List}} (\text{Relation})$

I. π : Projection

- $\pi_{\text{Attribute name}} (R)$: It is used to project required attribute from relation R.
- $\sigma_{\text{Condition}(P)} (R)$: It is used to select records from relation R, those satisfied the condition (P).

$$\overline{\sigma_1(\sigma_{c_2}(\sigma_{c_3}(R)))} \equiv \sigma_{c_3}(\sigma_{c_1}(\sigma_{c_2}(R)))$$

Example:

R	A	B	C
	8	4	5
X	2	4	5
	7	4	6
X	3	5	5

 $\pi_{B,C}(R):$

B	C
4	5
4	6
5	5

 $\sigma_{A>C}(R):$

A	B	C
8	4	5
7	4	6

Relational Algebra



Basic operators

II. Cross product (\times):

- $R \times S$: It result all attributes of R followed by all attributes of S, and each record of R paired with every record of S.
- Degree $(R \times S)$ = $\text{Degree}(R) + \text{Degree}(S)$
- $|R \times S| \geq |R| \times |S|$

$$R \times S = \begin{array}{c} n_1 \text{ Tuple} \\ \text{---} \\ C_1 \text{ Attribute} \end{array} \quad \begin{array}{c} n_2 \text{ Tuple} \\ \text{---} \\ C_2 \text{ Attribute} \end{array}$$
$$R \times S = \begin{array}{c} n_1 \times n_2 \text{ Tuple} \\ \text{---} \\ C_1 + C_2 \text{ Attribute} \end{array}$$

NOTE:

- Relation R with n tuples and
- Relation S with 0 tuples then
- number of tuples in $R \times S = 0$ tuples

Join (\bowtie)

I. Natural join (\bowtie)

$\overline{\bowtie}_{\text{Distinct Attrib}} \left[\begin{array}{l} \sigma_{\text{equality cond. } (R \times S)} \\ \text{On All Common} \end{array} \right]$

$$R \bowtie S \equiv \pi_{\text{distinct attributes}} (\sigma_{\text{equality between common attributes of } R \text{ and } S} (R \times S))$$

Example:

$T_1 (ABC)$ and $T_2 (BCDE)$

$$T_1.B = T_2.B \wedge T_1.C = T_2.C$$

$$\therefore T_1 \bowtie T_2 = \pi_{\underline{\underline{ABCDE}}}^{\underline{\underline{3}}} \left(\begin{array}{l} \sigma_{T_1 \cdot B} = T_2 \cdot B (T_1 \times T_2) \\ \wedge T_1 \cdot C = T_2 \cdot C \end{array} \right)$$

$T_1 (AB)$ and $T_2 (CD)$

$$\therefore T_1 \bowtie T_2 \equiv T_1 \times T_2 = \pi_{\underline{\underline{ABCD}}}^{\underline{\underline{4}}} (T_1 \times T_2)$$

NOTE:

Natural join equal to cross-product if join condition is empty.

Join (\bowtie)**II. Conditional Join (\bowtie_c)**

$$\square R \bowtie_c S \equiv \sigma_c (R \times S)$$

Join (\bowtie)

III. Outer Joins:

(a) LEFT OUTER JOIN

$R \bowtie S$: It produces

$(R \bowtie S) \cup \{ \text{Records of } R \text{ those are failed join condition with remaining attributes null} \}$

(b) RIGHT OUTER JOIN (\bowtie)

$R \bowtie S$: It produces

$(R \bowtie S) \cup \{ \text{Records of } S \text{ those are failed join condition with remaining attributes null} \}$

(c) FULL OUTER JOIN (\bowtie)

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

$$R \bowtie S = R \bowtie S \& \text{ left side Relation } \underline{\underline{R}}$$

$R \bowtie S \& \text{ Right Side Relation } \underline{\underline{S}}$ which failed to satisfy join condition

Natural Join

$$R.B = S.B \wedge R.C = S.C$$

R

A	B	C
1	2	4
3	2	6

S

B	C	D
2	4	8
2	7	4

R

2 Tuple

3 Attribute

S

2 Tuple

3 Attribute

$R \times S =$

R.A	R.B	R.C	S.B	S.C	S.D
1	2	4	2	4	8
1	2	4	2	7	4
3	2	6	2	4	8
3	2	6	2	7	4

A	B	C	D
1	2	4	8

$$R \times S = 2 \times 2 = 4 \text{ Tuple}$$

$$3+3 = 6 \text{ Attribute}$$

$$R \bowtie S = \pi_{ABCD} \left\{ \begin{array}{l} \sigma_{RB} = S.B \Lambda^{(R \times S)} \\ R.C = S.C \end{array} \right\}$$

$$R \bowtie S = \begin{array}{|c|c|c|c|} \hline & A & B & C & D \\ \hline 1 & 2 & 4 & 8 & \\ \hline \end{array}$$

Left Outer Join [\bowtie]

$(R \bowtie S)$

P
W

R

A	B	C
1	2	4
3	2	6

S

B	C	D
2	4	8
2	7	4

$(R \bowtie S) =$

A	B	C	D
1	2	4	8

fail

Fail

$R \bowtie S =$

A	B	C	D
1	2	4	8
3	2	6	Null

$R \bowtie S & \text{ Table From left side}$
 Relation R
 which failed
 Join Condition

Right Outer Join [\bowtie]

$$R \bowtie S =$$

A	B	C	D
1	2	4	8
Null	2	7	4

Full Outer Join [\bowtie]

Full outer join = Left outer join Union Right outer join

$$R \bowtie S = R \bowtie S \cup R \bowtie S$$

A	B	C	D
1	2	4	8
3	2	6	Null

U

A	B	C	D
1	2	4	8
Null	2	7	4

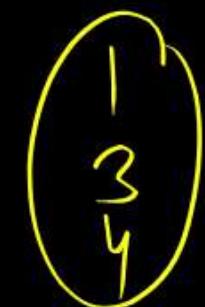
A	B	C	D
1	2	4	8
3	2	6	Null
Null	2	7	4

Q.

Let R and S be two relations with the following schema

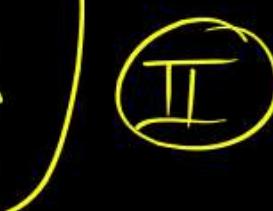
$R(P, Q, R1, R2, R3)$

$S(P, Q, S1, S2)$



1	2
3	5
1	7
4	9

1	2
3	5
1	9
4	2



Where $\{P, Q\}$ is the key for both schemas. Which of the following queries are equivalent?

- I. $\pi_P(R \bowtie S) \rightarrow R.P = S.P \wedge R.Q = S.Q$
- II. $\pi_P(R) \bowtie \pi_P(S)$
- III. $\pi_P(\pi_{P,Q}(R) \cap \pi_{P,Q}(S))$
- IV. $\pi_P(\pi_{P,Q}(R) - (\pi_{P,Q}(R) - \pi_{P,Q}(S)))$

[GATE & NIC 2020]

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

A

Only I and II

B

Only I and III

C

Only I, II and III

D

Only I, III and IV

Rename operator (ρ)

 ρ

It is used to rename table name and attribute names for query processing.

Example:

(I) Stud (Sid, Sname, age)

ρ (Temp, Stud) : Temp (Sid, Sname, age)

(II) $\rho_{I, N, A}$ (Stud) : Stud (I, N, A)

All attributes renaming

(III) ρ $sid \rightarrow I$ (Stud) : Stud (I, Sname, A)

$age \rightarrow A$

Some attribute renaming

Division

- It is used to retrieve attribute value of R which has paired with every attribute value of other relation S.
- $\pi_{AB}(R)/\pi_B(S)$: It will retrieve values of attribute 'A' from R for which there must be pairing 'B' value for every 'B' of S.

$$\pi_{AB}(R) / \pi_B(S)$$

Expansion of '/' by using basic operator

- Example: Retrieve sid's who enrolled every course.
- Result:

$$\boxed{\pi_{\text{sidcid}}(\text{Enroll}) / \pi_{\text{cid}}(\text{Course})}$$

Step 1: Sid's not enrolled every course of course relation.

(Sid's enrolled proper subset of course)

$$\underline{\pi_{\text{sid}}}((\pi_{\text{sid}}(\text{Enroll}) \times \pi_{\text{cid}}(\text{course})) - \pi_{\text{sidcid}}(\text{Enroll}))$$

- Step 2:

[sid's enrolled every course] = [sid's enrolled some course] - [sid's not enrolled every course]

$$\therefore \pi_{\text{sidcid}}(E) / \pi_{\text{cid}}(c) = \pi_{\text{sid}}(E) - \pi_{\text{sid}}((\pi_{\text{sid}}(E) \times \pi_{\text{cid}}(C)) - \pi_{\text{sidcid}}(E))$$

Division

Q.

Retrieve all student who are Enrolled **Some course** or **Any course** or at least one course?

Solution

$\Pi_{\text{Sid}} (\text{Enrolled})$

Enrolled	
Sid	Cid
S ₁	C ₁
S ₁	C ₂
S ₁	C ₃
S ₂	C ₁
S ₂	C ₃
S ₃	C ₁

Course
Cid
C ₁
C ₁
C ₃

Division

Q.

Retrieve all student who are Enrolled every course?

Solution

$\Pi_{\text{Sid}, \text{Cid}}(\text{Enrolled}) / \Pi_{\text{Cid}}(\text{Course})$

Find

2nd attribute must be same.

Enrolled	
Sid	Cid
S ₁	C ₁
S ₁	C ₂
S ₁	C ₃
S ₂	C ₁
S ₂	C ₃
S ₃	C ₁

Course
Cid
C ₁
C ₁
C ₁
C ₃

Division

P
W

~~$\Pi_{Sid, Cid} (\text{Enrolled})$~~
 ~~$\Pi_{Cid} (\text{Course})$~~

~~$\Pi_A(R) = \Pi_A [\Pi_A(R) \times \Pi_B(S) - R]$~~

$\Pi_{Sid} (\text{Enrolled}) - \Pi_{Sid} [\Pi_{Sid} (\text{Enrolled}) \times \Pi_{Cid} (\text{Course}) - \text{Enrolled}]$

S_1
 S_2 — S_2
 S_3 S_3

$\frac{Sid}{S_1}$ \cong
Avg

$$\begin{bmatrix} Sid \\ S_1 \\ S_2 \\ S_3 \end{bmatrix} \times \begin{bmatrix} Cid \\ C_1 \\ C_2 \\ C_3 \end{bmatrix} \Rightarrow$$

Sid	Cid
S_1	C_1
S_1	C_2
S_1	C_3
S_2	C_1
S_2	C_2
S_2	C_3
S_3	C_1
S_3	C_2
S_3	C_3

all student
enrolled every
course

Sid	Cid
S_1	C_1
S_1	C_2
S_1	C_3
S_2	C_1
S_2	C_3
S_3	C_1

Sid	Cid
S_2	C_2
S_3	C_2
S_3	C_3

These student
Not enrolled
their course

Tsid
S_2
S_3

Division

$$\overline{\Pi_B}(R) = \overline{\Pi_A} \left(\overline{\Pi_A}(R) \times \overline{\Pi_B}(S) - R \right)$$

$$\Pi_{AB}(R) / \Pi_B(S) = \Pi_A(R) - \Pi_A[\Pi_A(R) \times \Pi_B(S) - R]$$

Find Connection

$$\Pi_{ABCD}(R) / \Pi_{CD}(S) \Rightarrow \underbrace{\Pi_{AB}(R)}_{\uparrow} - \underbrace{\Pi_{AB}}_{\uparrow} \left[\underbrace{\Pi_{AB}(R)}_{\uparrow} \times \underbrace{\Pi_{CD}(S)}_{\uparrow} - R \right]$$

Q.

Consider the following three relations in a relational database:

Employee (eId, Name), Brand (bId, bName), Own(eId, bId)

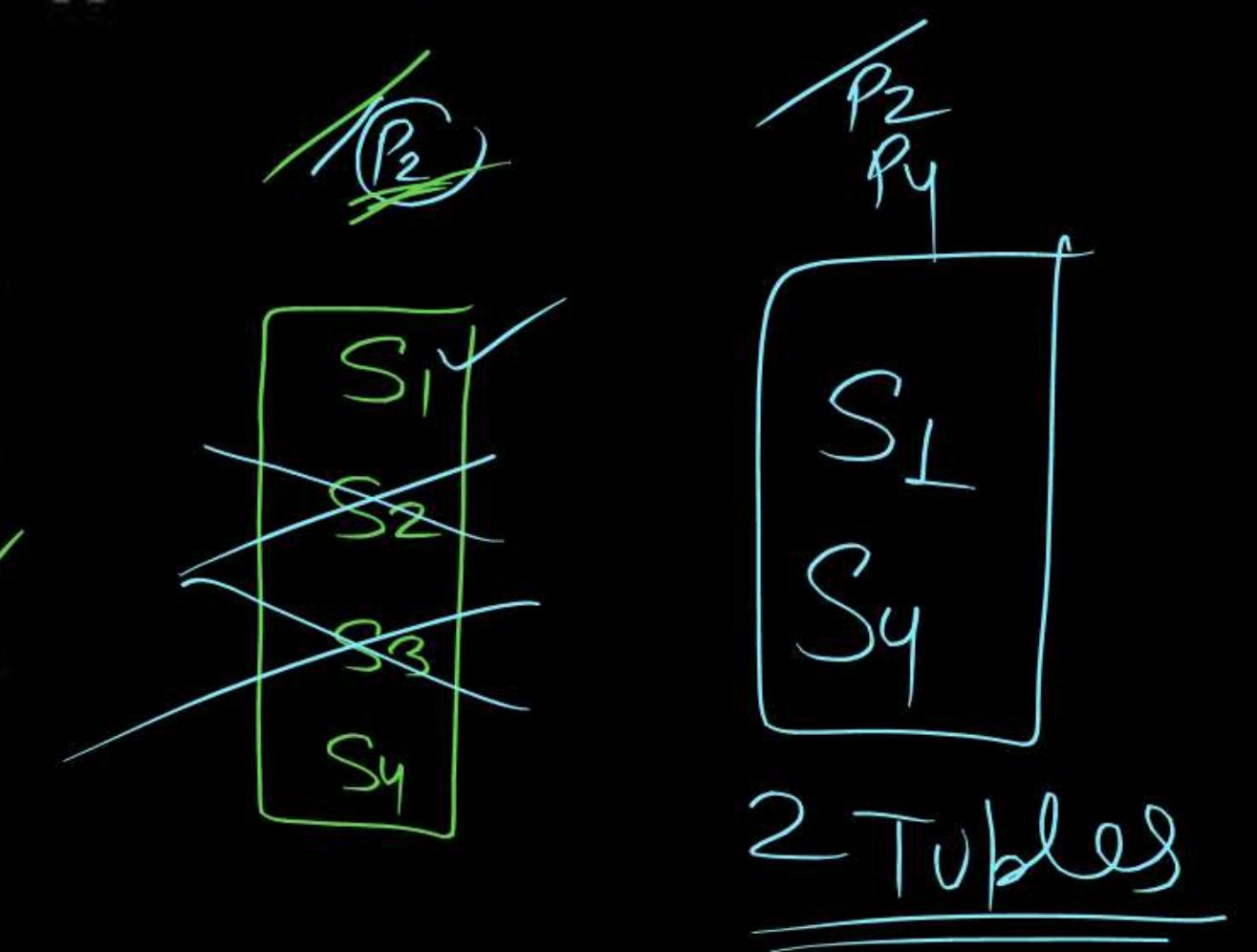
P
W

Which of the following relational algebra expressions return the set of eIds who own all the brands? [GATE: 2022]

- A $\pi_{eId} (\pi_{eId, bId} (Own / \overline{\pi_{bId} (Brand)})$
 $\overline{\pi_{eId} (Own)} = \overline{\pi_{eId} ((\pi_{eId} (Own) \times \pi_{bId} (Brand)) - \pi_{eId, bId} (Own))}$
 $\overline{\pi_{eId} (Own)} = \overline{\pi_{eId} (Own)} - \overline{\pi_{eId} ((\pi_{eId} (Own) \times \pi_{bId} (Brand)) - \pi_{eId, bId} (Own))}$
 $\overline{\pi_{eId} (Own)} = \overline{\pi_{eId} (Own)} - \overline{\pi_{eId} ((\pi_{eId} (Own) \times \pi_{bId} (Brand)) - \pi_{eId, bId} (Own))}$
 $\overline{\pi_{eId} (Own)} = \overline{\pi_{eId} (Own)} - \overline{\pi_{eId} ((\pi_{eId} (Own) \times \pi_{bId} (Brand)) - \pi_{eId, bId} (Own))}$
- B $\pi_{eId} (Own) - \pi_{eId} ((\pi_{eId} (Own) \times \pi_{bId} (Brand)) - \pi_{eId, bId} (Own))$
- C $\pi_{eId} (\pi_{eId, bId} (Own) / \pi_{bId} (Own))$
- D $\pi_{eId} ((\pi_{eId} (Own) \times \pi_{bId} (Own)) / \pi_{bId} (Brand))$

Consider the two relation Suppliers and Parts are given below.

Suppliers		Parts
S _{no}	P _{no}	P _{no}
S ₁	P ₁	
S ₁	P ₂	
S ₁	P ₃	
S ₁	P ₄	
S ₂	P ₁	
S ₂	P ₂	
S ₃	P ₂	
S ₄	P ₂	
S ₄	P ₄	



$\pi_{S_{no} P_{no}}$ (Suppliers) / $\pi_{P_{no}}$ (Parts)

The number of tuples are there in the result when the above relational algebra query executes is ____.

Set operator

U: Union operator

- : Except or minus

\cap : Intersection operator

- To apply set operations relations must be union compatible.
- R and S relations are union compatible
- If and only if-
 - (i) Arity of R equal to Arity of S and
 - (ii) Domain of attributes of R must be same as domain of attributes of S respectively.

Example

Example 1:



{Arity not same so, set operation not allowed}

Example 2:



{Arity same but Sname domain is different from marks so, not allowed}

Example

$$\pi_{\underline{S \text{ id } Sname}}(\dots \dots \dots) \cap \pi_{\underline{S \text{ id } ID, Stud name}}(\dots \dots \dots)$$

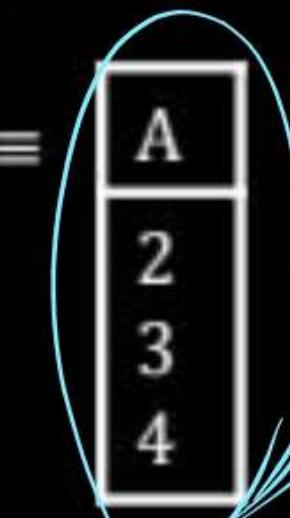
{Arity and domains are same so, allowed for set operation}

1. Set operation on relation:

R	A
	2
	2
	2
	3

S	B
	2
	2
	2
	4

$$\underline{R \cup S : \{x / x \in R \vee x \in S\} = }$$



A
2
3
4

$$R - S : \{x / x \in R \wedge x \notin S\} = \cancel{\text{ }}$$



A
3

$$R \cap S : \{x / x \in R \wedge x \in S\} = \cancel{\text{ }}$$



A
2

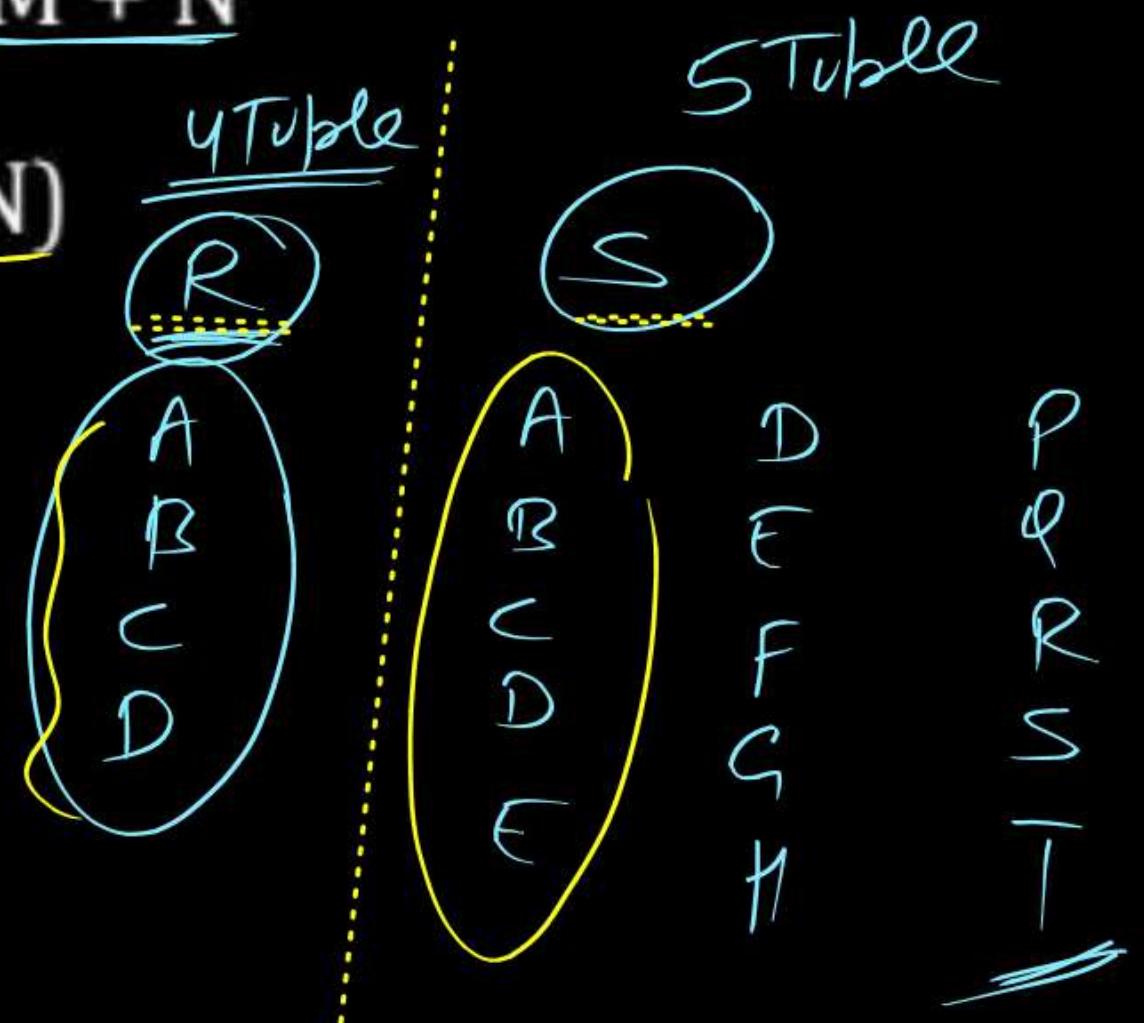
Assume Relation R & Relation S consist M & N Tuple Respectively

(1) Range of tuples in $\underline{R \cup S} = \underline{\max(M, N)}$ to $\underline{M + N}$

(2) Range of tuples in $\underline{R \cap S} = \underline{\phi}$ to $\underline{\min(M, N)}$

(3) Range of tuples in $R - S = \underline{\phi}$ to \underline{M}

(4) Range of tuples in $S - R = \underline{\phi}$ to \underline{N}



Sub-language of SQL

Data definition language (DDL)
⇒ used to define modify structure & definition of DB table and allowed Modify 'IC'[PK/AK/FK]
Create Table
DROP table
Alter Table

- Add/remove attributes
- Used to modify IC

Data manipulation language (DML)
⇒ Used to modify data records And used to access data records From set of tables.
Insert into T name
Delete from T name
Update T name Set
Select from table, to access data

Data Control Language

↓

Data control for Transaction control [To avoid inconsistency] Lock (T name) unlock (T name commit: Roll back; etc)
↓
Data control for Role based access (security)
⇒ Grant access of DB table
⇒ Revoke access of DB table

- SQL : SELECT [DISTINCT] A₁, A₂, A_n =
FROM R₁, R₂, R_n
WHERE P;

Equivalent RA : $\pi_{A_1, A_2, \dots, A_n}(\sigma_p(R_1 \times R_2 \times \dots \times R_n))$

- SELECT projection of RA (π)
FROM Cross-product (x)
WHERE Selection operator of RA (σ)

- SELECT [DISTINCT] A₁, A₂, A_n
 - FROM R₁, R₂, R_n
- [WHERE condition]
- [GROUP BY (attributes)]
- [HAVING condition]
- [ORDER BY (attributes) (DESC)]
- [] ← optional

FROM \Rightarrow cross-product of relations



WHERE \Rightarrow Selection operator (σ) to apply condition for each record



GROUP BY



HAVING



SELECT



DISTINCT



ORDER BY



- ❑ It is used to group records data based on specific attribute.

- ❑ HAVING clause must be followed by GROUP BY clause.
- ❑ HAVING clause used to select groups those are satisfied having clause condition.

HAVING: Fourth executable clause (if used in query).

It is used to select the group which satisfy the condition (condition is for each group).

STUDENT

Sid	Branch	Marks
S ₁	CS	60
S ₂	IT	70
S ₃	CS	90
S ₄	IT	60
S ₅	EC	55
S ₆	EC	NULL

(GROUP By
Branch)



Sid	Branch	Marks
S ₁	CS	60
S ₃	CS	90
S ₂	IT	70
S ₄	IT	60
S ₅	EC	55
S ₆	EC	NULL

Select * FROM STUDENT
GROUP By (Branch)
HAVING AVG(Marks) > 61

Sid	Branch	Marks
S ₁	CS	60
S ₃	CS	90
S ₂	IT	70
S ₄	IT	60

Q.

Select min(marks), Branch
FROM Student

55

CS

IT

CS

IT

EC

EC

COUNT

SUM

AUG

MIN

MAX



Such Syntax is not allowed in SQL

NOTE:

When aggregate operator & other Attribute used in select clause is
Allowed only if other attribute must be in Group of Clause.

Select min (marks) Branch

FROM Student

GROUP By (Branch)

Example:

1. SELECT A
FROM R
GROUP BY A
HAVING AVG (c) > 60;

2. SELECT A, AVG (c)
FROM R
GROUP BY A
HAVING some (c) > 50;

3. SELECT A
FROM R
GROUP BY A
HAVING C > 60;

↓

Not allowed

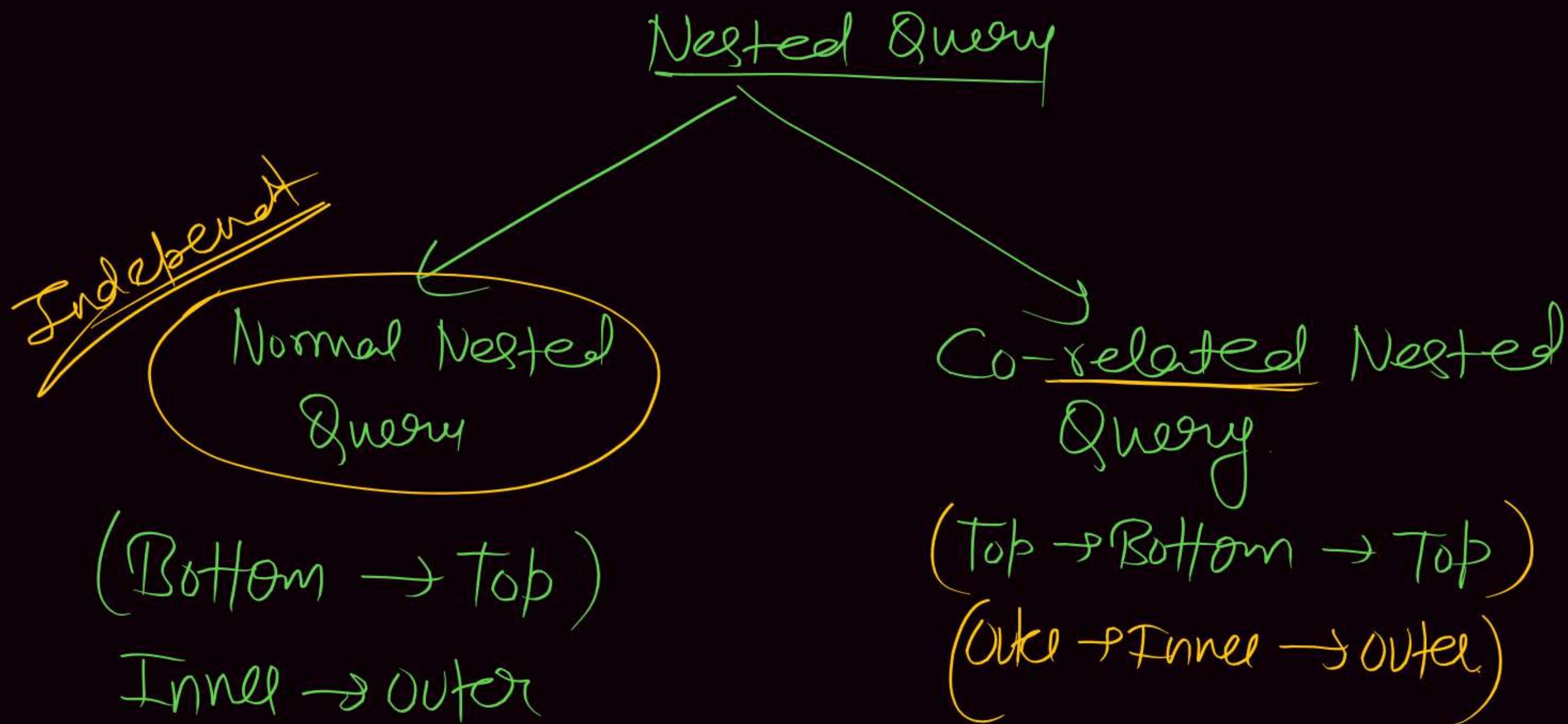
NOTE:

WHERE clause condition tested for each record but HAVING clause condition tested for each GROUP.

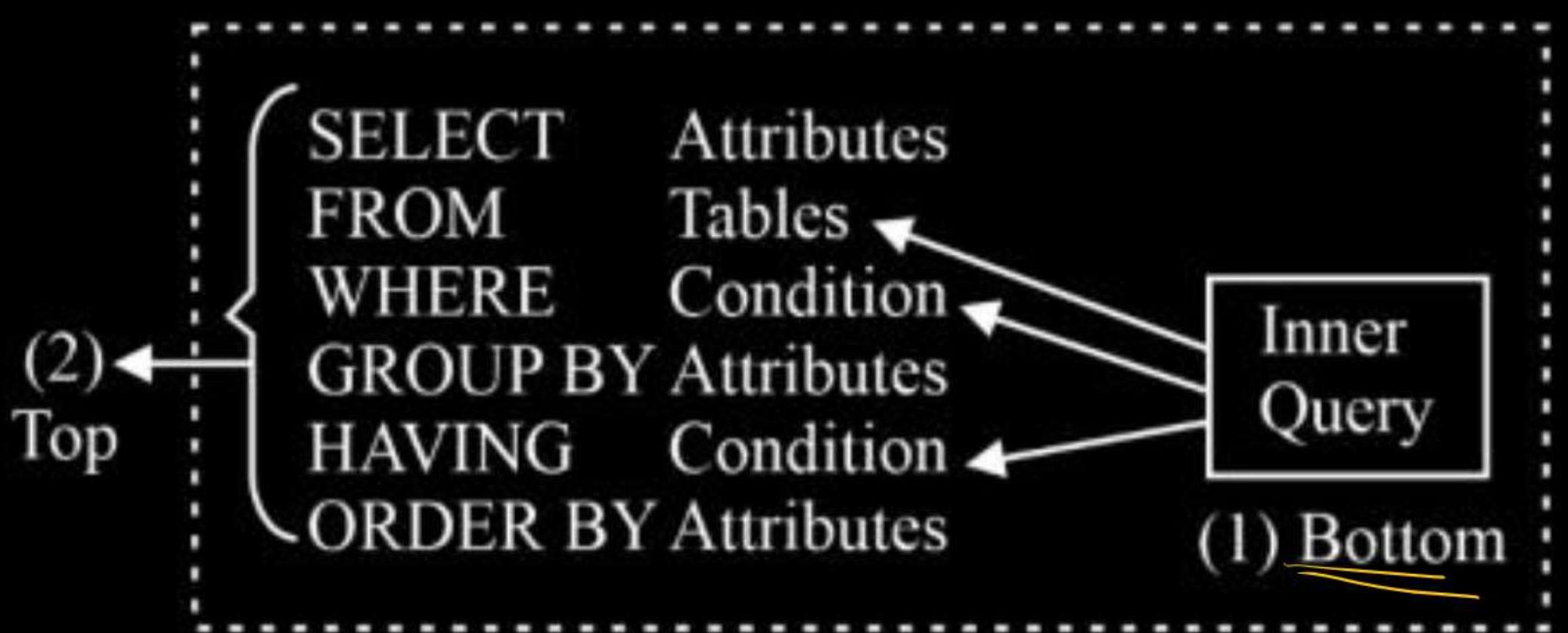
- WITH clause is used to create sub-query result that can be re-used many times in query processing.

Example:

```
WITH Temp (A1, A2) as
  (SELECT T1.A1, T2.A2
   FROM Emp T1, Emp T2
   WHERE T1.A1 < T2.A1)
```



- Inner query independent of outer query.



- Execution flow must be Bottom to Top mean first bottom (Inner query) evaluated then outer query executed.

- In Nested Co-related query inner query uses attributes from outer query tables.
- In Co-related Nested query inner query allowed in WHERE, HAVING clause of outer query.

Example: SELECT

```
FROM R
WHERE (SELECT count(*)
FROM S
WHERE S.B < R.A) < 5;
```

Attribute defined in
the outer query.

EXISTS: (Checks): Return True if Inner Query Result

EXISTS: (Checks): Return True if Inner Query Result Not Empty

NOT EXIST: Return True if Inner Result Empty

Correlated Nested Query: Inner Query Using attribute defined in Outer Query

Select C.Sid
FROM Catalog C
WHERE EXISTS

(Select *
FROM Part P
WHERE P.Pid = C.Pid)

Corelated Nested Query

```
Select C.sid  
FROM Catalog  
WHERE EXISTS
```

Inner Query using
Attributes defined in
the Outer Query

```
Select*  
FROM Parts P  
WHERE P.Pid = C.Pid  
AND Color = Red
```

Catalog (Sid, Pid, Cost)
Parts (Pid, Pname, Color)

Nested Queries

(Independent)
Normal Nest Query

Inner → Outer

Bottom → Top

Corelated Nested
Query

Outer → Inner → Outer

Top → Bottom → Top

```
for(i = 1; i <= n; i++)  
  for(j = 1; j <= m, j++)
```

P
W

i = 1
((|))
J = 1...m

i = 2
((|))
J = 1...m

i = 3
((|))
J = 1...m

.....

i = n
((|))
J = 1...m

i=1
Outer Inner
j=1...m

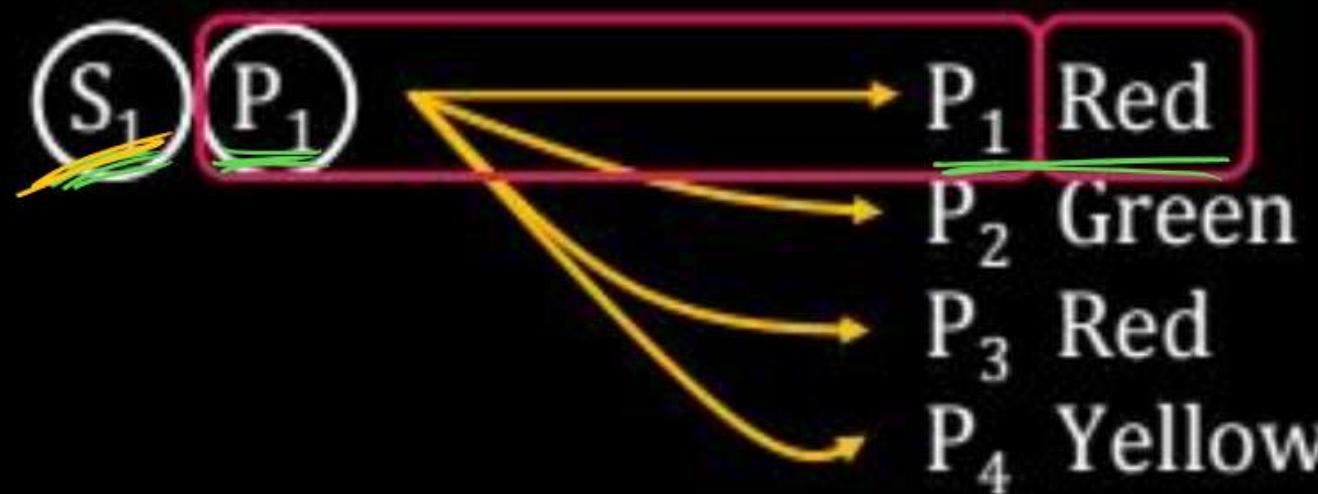
i=2

Outer

P
W

1st Iteration:

Pid Match & Color = Red



[Pid Match & color = Red]

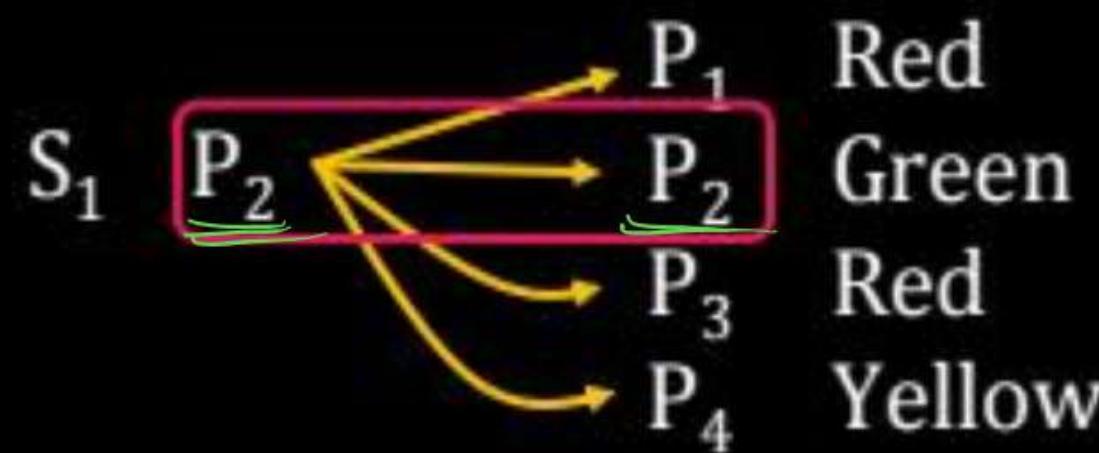
1 Tuple Return

S_r

Catalog Parts

Sid	Pid	Pid	Color
S ₁	P ₁	P ₁	Red
S ₁	P ₂	P ₂	Green
S ₂	P ₃	P ₃	Red
S ₄	P ₄	P ₄	Yellow

IInd Iteration:

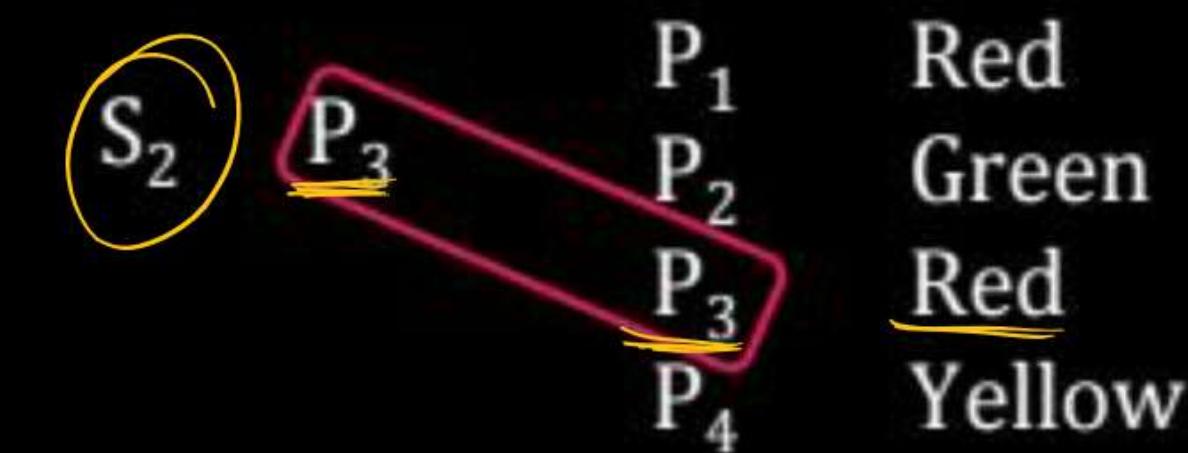


[Pid Match but color not Red]

0 Tuple Return



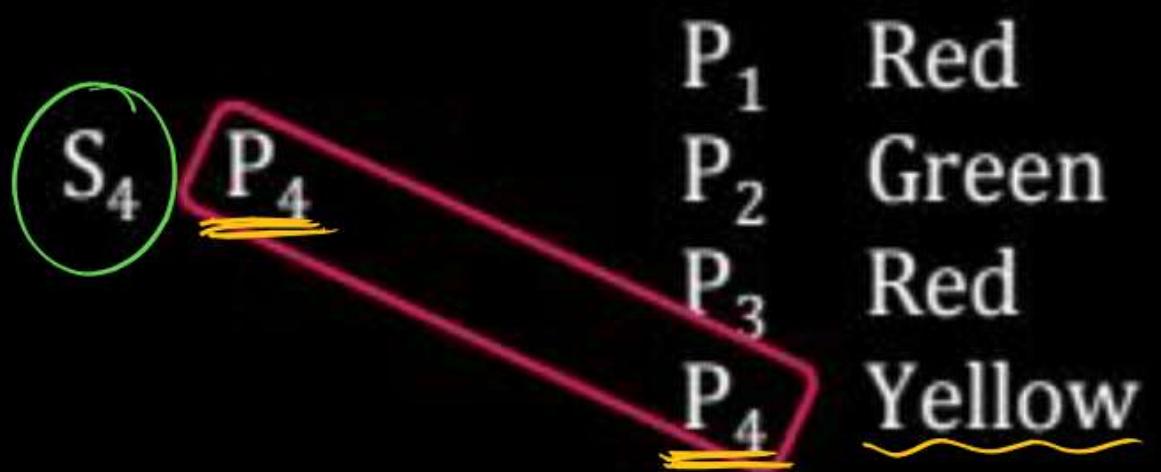
IIIrd Iteration:



[Pid Match & Color Red]

1 Tuple Return



IVth Iteration:

[Pid Match but color not Red]

0 Tuple Return

NOT Exist 

EXISTS

o/p

Sid
S_1
S_2

If NOT EXIST then output

Sid
S_1
S_4

Before EXIST & NOT EXISTS No Attribute is required.

Before IN & NOT IN Attribute is Required.

Q.

Given Relative Schema

Emp(Eid, Ename, Salary)

Department(Eid, dname, code)

Retrieve Employee ID who have no Department?

Query I: Select Eid

FROM Emp E, Dep D

WHERE E.Eid <> D.Eid

Which is true?

A) Q₁ ✓ Q₂ ✗

B) Q₂ ✓ Q₁ ✗

C) Q₁ ✓ Q₂ ✓

D) Q₁ ✗ Q₂ ✗

P
W

Eid	Ename
E ₁	A
E ₂	B
E ₃	A
E ₄	

Eid	Dname
E ₁	A
E ₁	B

Query II: Select Eid
 FROM Emp E
 WHERE NOT EXISTS

Empty

(Select *
 FROM Dep D
 WHERE E.Eid = D.Eid)

Query I:

$E_1 \neq E_1 \rightarrow F$

$E_1 \neq E_1 \rightarrow F$

$E_1 \neq E_2 \rightarrow T$

$E_2 \neq E_1 \rightarrow T$

$E_2 \neq E_1 \rightarrow T$

$E_2 \neq E_2 \rightarrow F$

$E_3 \neq E_1 \rightarrow T$

$E_3 \neq E_1 \rightarrow T$

$E_3 \neq E_2 \rightarrow T$

$E_4 \neq E_1 \rightarrow T$

$E_4 \neq E_1 \rightarrow T$

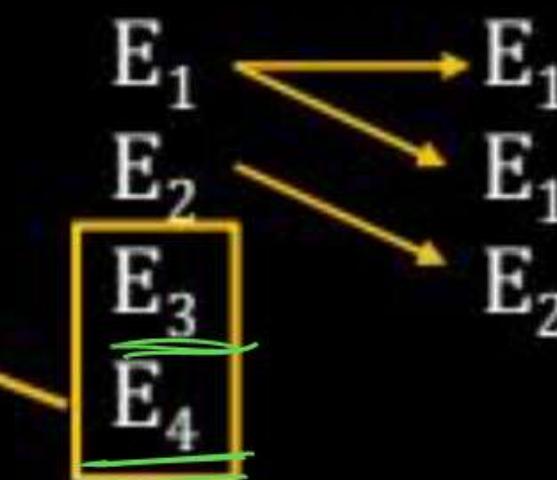
$E_4 \neq E_2 \rightarrow T$

Output of Query I

Eid
E ₁
E ₂
E ₂
E ₃
E ₃
E ₃
E ₃
E ₄
E ₄
E ₄
E ₄

Output of Query II

E ₃
E ₄



Ans

- ❑ If co-relation in WHERE clause then inner query re-computes for each record of outer query From clause.
- ❑ If correlation in HAVING clause then inner query re-computes for each group of outer query.

Function used for nested Queries

1. IN/NOT IN
2. ANY
3. ALL
4. EXISTS/NOT EXISTS

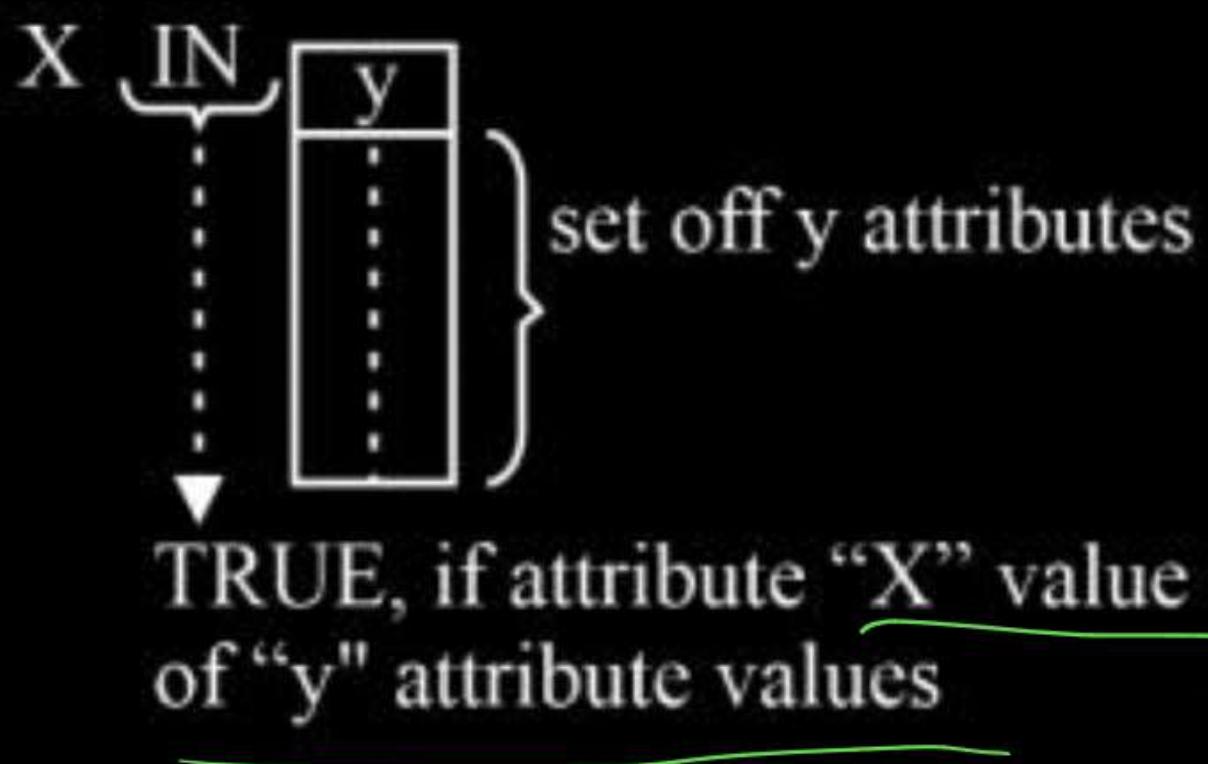
Best suitable for
Non-co-related query

Best suitable for
co-related query ✓

Function used for nested Queries

1. IN Function

It is used for membership testing.

**NOTE:**

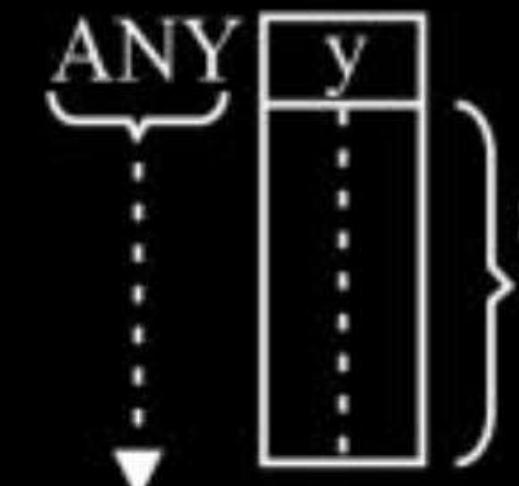
Queries of "equi Join" can be implemented by IN function

$(R \bowtie S)$

Function used for nested Queries

2. ANY Function (operator "<, ≤, >, ≥, =, <>")

X operator



set of y attributes

TRUE, Only if attribute 'x' value satisfied for given comparison (operator) with some (atleast one) 'y' value of 'y' set.

$x > \text{ANY} (10, 20, 30)$

$(n > 10) \textcircled{OR} (n > 20) \textcircled{OR} (n > 30)$

11, 12, 13,

Function used for nested Queries

3. ALL Function

X operator

ALL



set of 'y' attributes

$(x > 10) \text{ AND } (x > 20) \text{ AND } (x > 30)$

~~g AND~~

$x > \text{ALL } (10, 20, 30)$

O/P

31, 32, 33, 34

TRUE, Only if attribute 'x' value satisfied for given comparison (operator) with every (all) 'y' value of 'y' set.

NOTE:

ANY function can be used as queries of conditional join query.

Example

$\pi_A(R \bowtie S)$ = SELECT A
FROM R
WHERE R.A > ANY (SELECT B FROM S)

Important point 2

- IN function equal to '= ANY' function $X \text{ IN } \begin{array}{|c|} \hline Y \\ \hline \vdots \\ \hline \end{array} \equiv X = \text{ANY } \begin{array}{|c|} \hline Y \\ \hline \vdots \\ \hline \end{array}$
- IN function not equal to '= ANY' if more than one attribute used for IN comparison.

$$(A,B) \text{ IN } \begin{array}{|c|c|} \hline C & D \\ \hline \vdots & \vdots \\ \hline \end{array} \neq (A,B) = \underbrace{\text{ANY}}_{\text{Not allowed}} \begin{array}{|c|c|} \hline C & D \\ \hline \vdots & \vdots \\ \hline \end{array}$$

- NOT IN function equal to "<> ALL". $X \text{ NOT IN } \begin{array}{|c|} \hline Y \\ \hline \vdots \\ \hline \end{array} \equiv X <> \text{ALL } \begin{array}{|c|} \hline Y \\ \hline \vdots \\ \hline \end{array}$

4. EXISTS clause

- It is used to test result of inner query is empty or not empty.
- EXISTS(Query)



TRUE, if result of inner query is
non empty that is at least one
record in result of inner query

Example:

```
SELECT R.A  
FROM R  
WHERE EXISTS (SELECT *  
               FROM S  
               WHERE R.A > S.B);
```

At least one record of S

relation satisfy $R.A > S.B$

Emp (eid, sal ,dno., Gen) Retrieve eids of female's whose salary more than (some) male emp.

Emp (eid Sal dno . Gender)

SQL
[JOIN Query]

```
SELECT DISTINCT T1. eid
FROM EMP T1, EMP T2
WHERE
T1.gen = Female and
T2.gen = Male and
T1.sal > T2.sal    > Male
```

SQL
[Nested Query]

```
SELECT Eid
FROM Emp
WHERE gen = Female
and sal > ANY(SELECT sal
                FROM Emp
                WHERE gen = male);
```

SQL**[Co-related Nested Query]**

```
SELECT Eid
FROM Emp T1
WHERE T1.gen = Female and
EXISTS (SELECT *
        FROM Emp T2
        WHERE T2.gen = male and T1.Sal > T2.sal);
```

4. TRANSACTIONS AND CONCURRENCY CONTROL

Transaction



A set of logically related operation to perform unit of work.

Degree of concurrency

The number of users (Transactions) using data bases simultaneously (concurrently).

Flat File System

In flat file system 1 file is a resource so, only one user allowed at a time.

DBMS File System

- In DBMS file system 1 record is a resource so, it allows as many users as the number of records.
- More degree of concurrency.

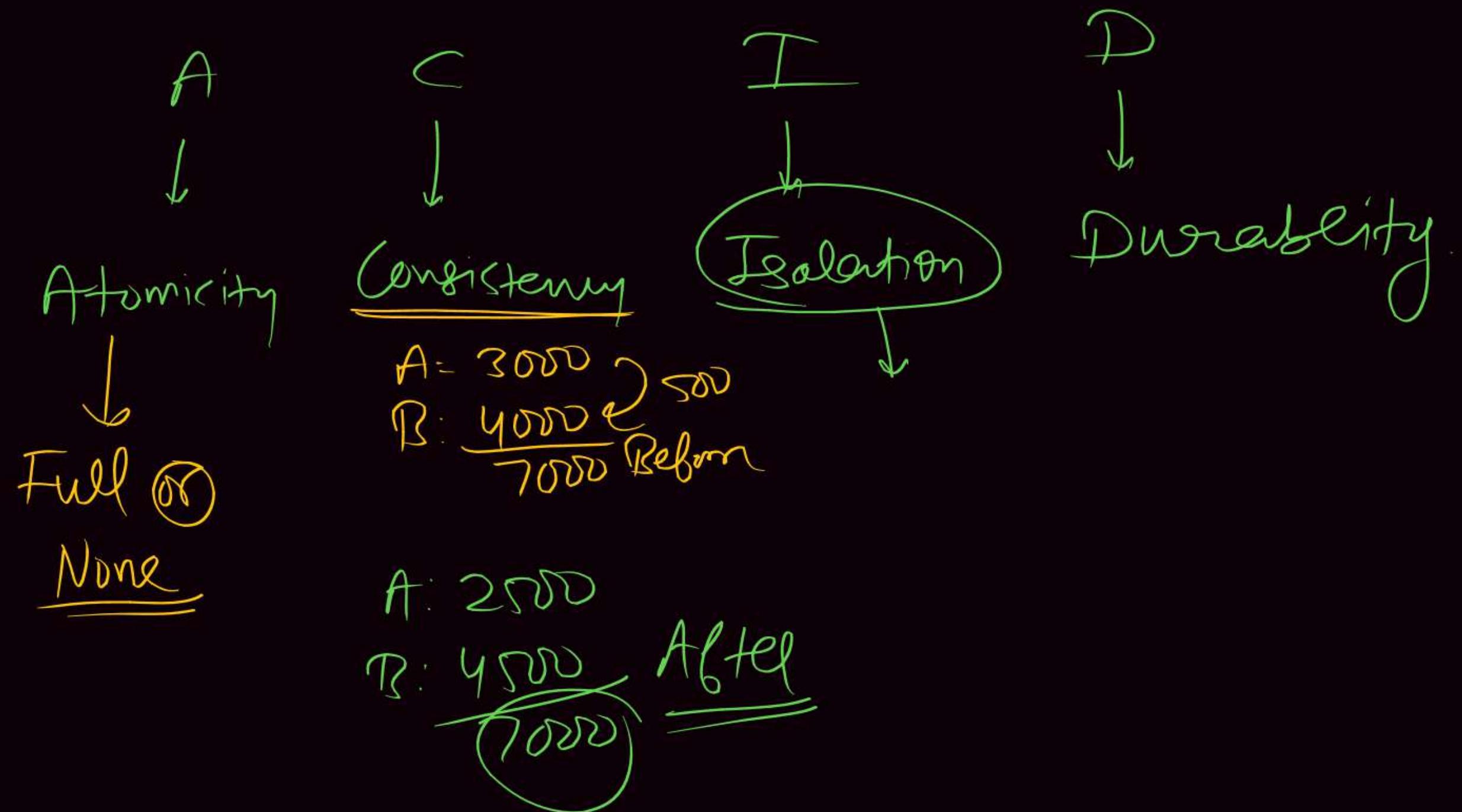
Main Operations in Transactions

Read (A)

Access the data item 'A' from DB file 'Disk' to programmed variable (main memory) in order to use current value of 'A' in transaction logic.

Write (A)

Modification of data item 'A' in DB file.



ACID Properties

To preserve integrity (correctness) each transaction must satisfy ACID properties

DBMS Software	<p>A:Atomicity } D:Durability }</p> <p>Recovery management component of DBMS software take cares of atomicity and durability</p>
	<p>I : Isolation} Concurrency control component of DBMS</p>
DBA User	<p>software is responsible for isolation.</p> <p>C : Consistency} User (DBA or DB developer) is responsible for consistency.</p>

Atomicity

Execute all operations of transaction including commit or execute none of the operation of transaction by the time of transaction termination.

Recovery management component should roll back, if transaction fails anywhere before commit.

Redo Operation → New Value

- It performs all the modification of database file because of transaction commit.
- Clean all the log entries of committed transaction.
- Redo is not performed after every commit but after every check point.

Undo Operation → OLD Value

- Undone all the write operation performed by the transaction.
- Convert dirty block (updated block) into clean block.

Transaction Number	Data Item	OLD Value	New Value
(T ₁ , X, S, ?)		Top	
(T ₁ , X, ?, LL)		UNDO	Top
X = 5	<u>UNDO</u>	X ≠ 5 / 5	X = LL

T_i : Commit

Check point

T_j: Commit

Check point

□ Check point issued by DBMS software in regular interval.

□ If checkpoint issued

(I) Performs Redo operation on all the committed transaction until previous checkpoint.

Example :

9 : 00 AM

:

9 : 05 AM

:

9 : 10 AM

:

9 : 15 AM

DB Started

1st checkpoint

2nd checkpoint

Commit

System Crash

9:05
T₁ T₂ T₃ T₄ T₅

T₃, T₅ Commit ✓
Checkpoint 9:10 AM

T₄ Commit

Redo : T₄
Undo : T₁, T₂

System Crash

If System crash/failure happen, required operation to recover are

1. All committed transaction until previous checkpoint will perform Redo.
2. All uncommitted transaction in entire system will perform undo.
3. Clean all log entries.

Roll back (Abort)

Undo modification of database file which are done by failure transaction.

Durability

- Durability maintained by Recovery management component of DBMS Software.
- The transaction should able to recover under any case of failure.
- Transaction fails because of
 - 1. Power failure
 - 2. Software crash
 - OS restarted
 - DBMS restarted
 - 3. OS/DBMS concurrency controller may kill transaction.
 - 4. Hardware Crash (Disk failure)
RAID architecture of disk design is used to overcome the problem.

Consistency

- ❑ User responsible for consistency.
- ❑ Database should be consistent before and after the execution of the transaction.
- ❑ DB operations requested by user (SQL queries/transaction operation) must be logically correct.

Isolation



- ❑ Isolation maintained by concurrency control component.
- ❑ Isolation means concurrent execution of 2 or more transaction result must be equal to result of some serial schedule.



Consistency

Isolation

Schedules



Time order execution sequence of two or more transactions.

Example :

$S : R_2(A) R_1(A) W_3(A)$

Types of schedule

Serial
Schedule

Concurrent
Schedule

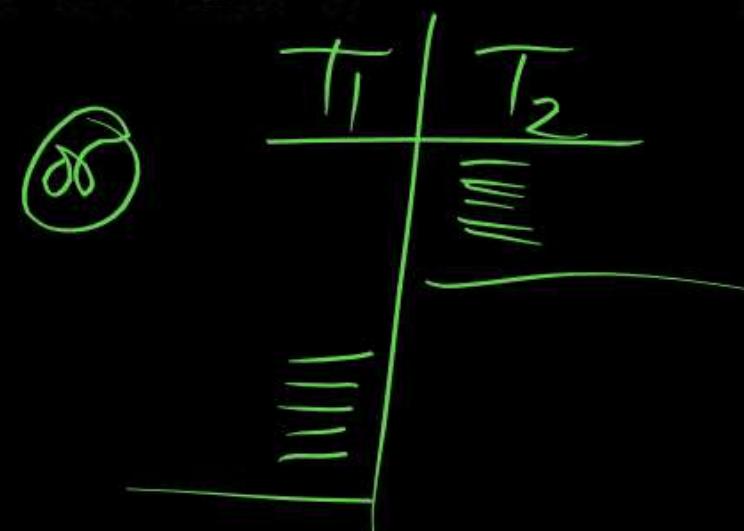
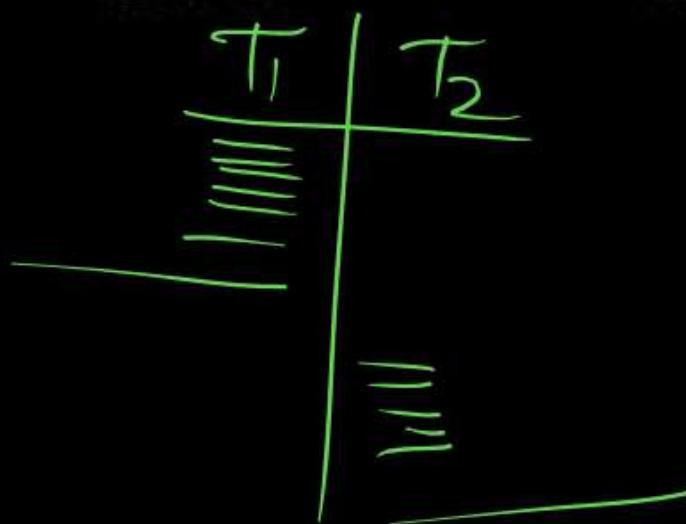
Serial Schedule

- ❑ After Commit of one transaction, begins (Start) another transaction.
- ❑ Number of possible serial Schedules with 'n' transactions is "n!"
- ❑ The execution sequence of Serial Schedule always generates consistent result.

$n!$ Serial Schedule

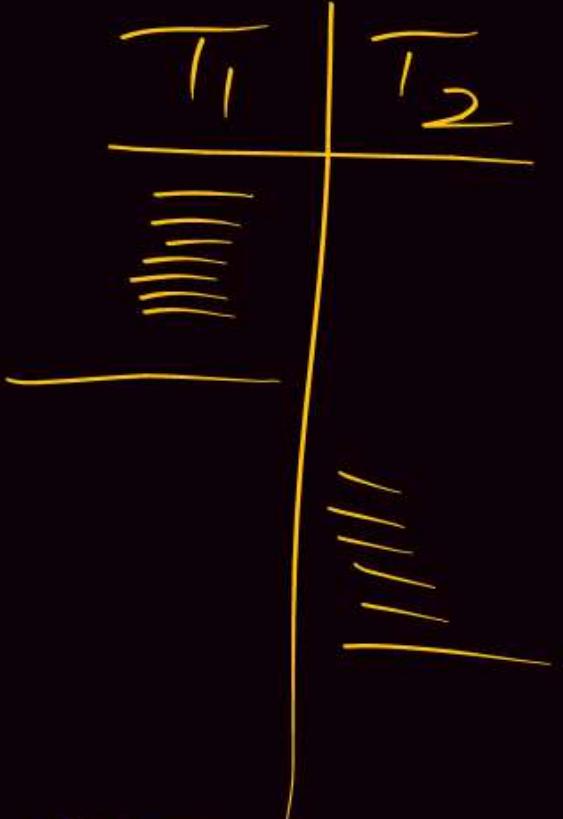
Example

S : R₁(A) W₁(A) Commit (T₁) R₂(A) W₂(A) commit (T₂).



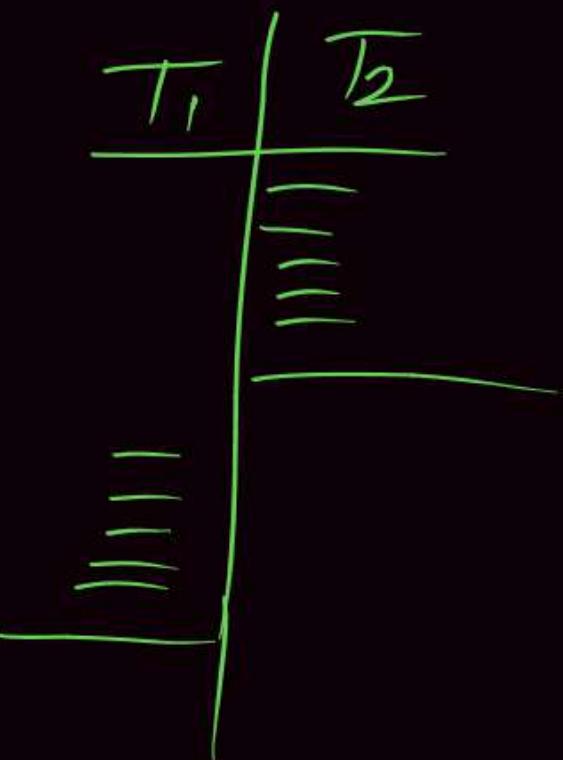
2 Transaction

$2 \rightarrow 2$ serial
Schedule



$\langle T_1 \quad T_2 \rangle$

T_1 followed by T_2



$\langle T_2 \quad T_1 \rangle$

T_2 followed by T_1

3 Transaction

3) Serial

6 Serial
Schedule

$\langle T_1 \ T_2 \ T_3 \rangle$

$\langle T_1 \ T_3 \ T_2 \rangle$

$\langle T_2 \ T_1 \ T_3 \rangle$

$\langle T_2 \ T_3 \ T_1 \rangle$

$\langle T_3 \ T_1 \ T_2 \rangle$

$\langle T_3 \ T_2 \ T_1 \rangle$

Advantage

- Serial Schedule always produce correct result (integrity guaranteed) as no resource sharing.

Disadvantage

- Less degree of concurrency.
- Through put of system is low.
- It allows transactions to execute one after another.

Concurrent Schedule

Transactions can execute concurrently or simultaneously

- May result inconsistency.
- Better throughput and less response time.
- To maintain consistency transaction should satisfy ACID property.
- If T_1 and T_2 transaction with 'n' and 'm' operations each, then

$$\text{No. of concurrent Schedule} = \frac{(n+m)!}{n! m!}$$

$$\begin{aligned} T_1 &\rightarrow n \quad (2) \\ T_2 &\rightarrow m \quad (4) \end{aligned} \rightarrow \frac{(2+4)!}{2! 4!}$$

Schedule Classification



Schedule Classification



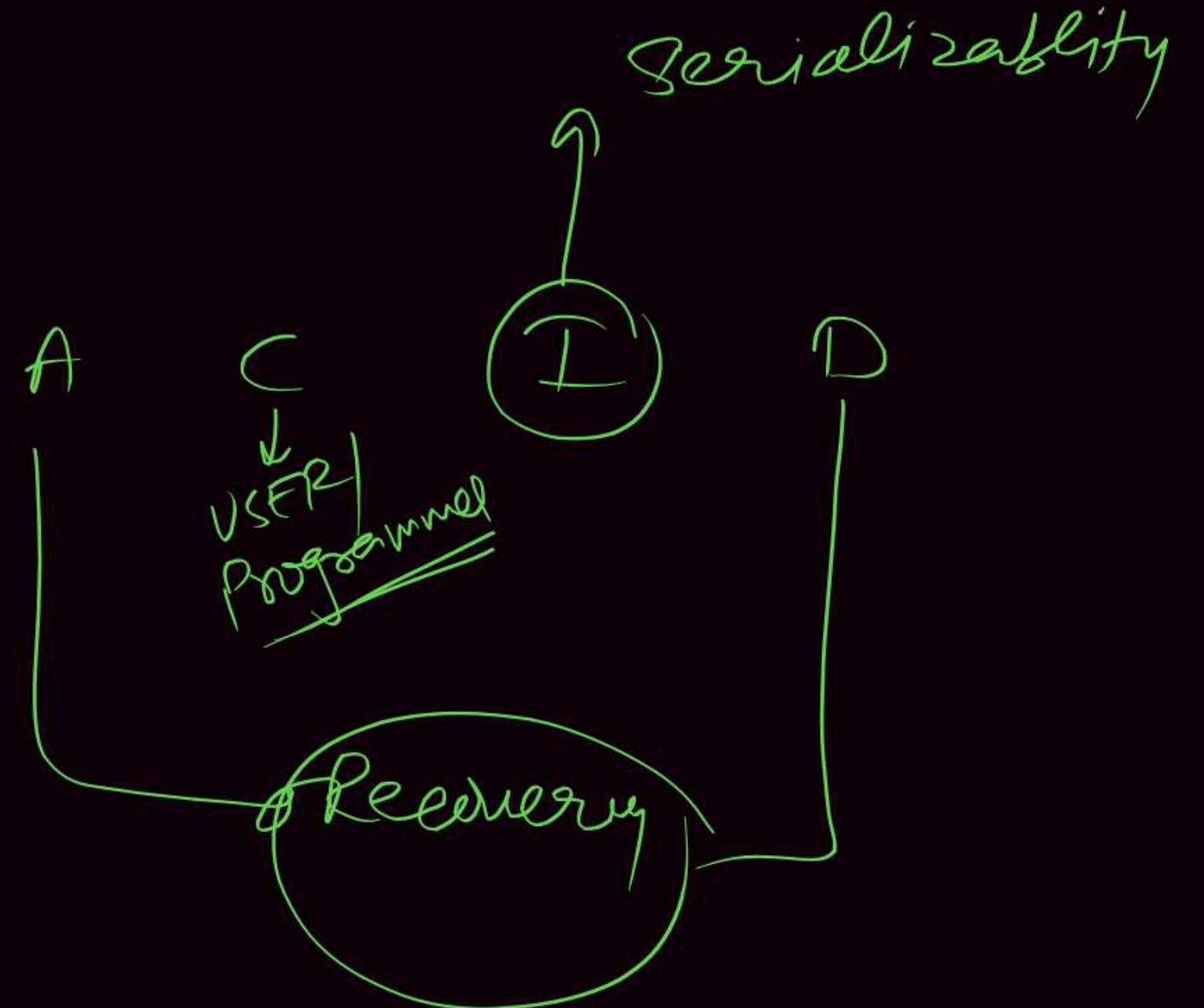
Serializability

(Isolation must satisfy)

Recoverable Schedule

(Atomicity and

Durability must satisfy)



Let T_1 transfer 100 Rs from A to B, and T_2 transfer 10% of the balance from A to B.

Schedule 1

T_1	T_2
<pre> read (A) $A = 2000$ A := A - 100 write (A) $A = 1900$ read (B) $B = 3000$ B := B + 100 write (B) $B = 3100$ commit $A = 1900$ $A = 1710$ </pre>	<pre> A: 1710 B: 3290 $\frac{5000}{5000}$ </pre>

$S_1 < T_1 \quad T_2 >$ S_1

Serial schedule in which T_1 is followed by T_2 :

Schedule 2

T_1	T_2
<pre> A: 1700 B: 3300 $\frac{5000}{5000}$ </pre>	<pre> read (A) $A = 2000$ temp := A * 0.1 A := A - temp $temp = 200$ write (A) read (B) B := B + temp write (B) Commit $B = 3200$ </pre>

serial schedule where T_2 is followed by T_1

~~A = 2000~~
~~B = 3000~~
~~5000~~

Schedule 3

T ₁	T ₂
read (A) $A = 2000$	
$A := A - 100$	
write (A) $\textcircled{A = 1900}$	
read (B) $B = 3000$	
$B := B + 100$	
write (B)	
commit $\textcircled{B = 3100}$	
$B = 3100$	
$\textcircled{B = 3290}$	
read (B)	
$B := B + \text{temp}$	
write (B)	
Commit	
	C ₁

~~A = 2000~~
~~B = 3000~~
~~5000~~

Schedule 4

T ₁	T ₂
read (A) $A = 2000$	
$A := A - 100$	
	+ $\boxed{A: 1900}$
	$\boxed{B: 3300}$
	$\boxed{5200}$
	Inconsistent
	$\textcircled{A = 1900}$
write (A)	
read (B) $\rightarrow B = 3000$	
$B := B + 100$	
write (B) $\textcircled{B = 3100}$	
commit	
	$\textcircled{B = 3100}$
	$\textcircled{B = 3290}$
	$\textcircled{B = 3300}$
	$\textcircled{B = 3300}$
	$\textcircled{B = 3300}$
	read (B) $\rightarrow B = 3100$
	$B := B + \text{temp}$
	write (B)
	Commit
	$\textcircled{B = 3300}$
	C ₂

m : # Transaction

(Note)

Serial Schedule (All Serial Schedule m!)

are always Consistent

(Note)

Non Serial Schedule May \otimes May Not be Consistent

But we do Concurrent execution

Serializable Schedule

→ Conflict Serializable
→ View Serializable

Serializability

→ Consistent

- Basic Assumption: Each transaction preserves database consistency.
- Thus, serial execution of a set of transactions preserves database consistency.
- A (possibly concurrent) schedule is serializable if it is equivalent to a serial schedule. Different forms of schedule equivalence give rise to the notions of:
 1. Conflict serializability
 2. view serializability

Conflict Instruction

Same Data Item [Q]

T_i T_j

$R(Q) \rightarrow W(Q)$

$W(Q) \rightarrow R(Q)$

$W(Q) \rightarrow W(Q)$

Conflicting Instructions

 $\underline{T_i}$ $\underline{T_j}$

- Instructions l_i and l_j of transactions T_i and T_j respectively, conflict if and only if there exists some item Q accessed by both l_i and l_j , and at least one of these instructions wrote Q.
 1. $l_i = \text{read}(Q), l_j = \text{read}(Q)$. l_i and l_j don't conflict.
 2. $l_i = \text{read}(Q) l_j = \text{write}(Q)$. They conflict.
 3. $l_i = \text{write}(Q) l_j = \text{read}(Q)$. They conflict
 4. $l_i = \text{write}(Q) l_j = \text{write}(Q)$. They conflict
- Intuitively, a conflict between l_i and l_j forces a (logical) temporal order between them.
 - ❖ If l_i and l_j are consecutive in a schedule and they do not conflict, their results would remain the same even if they had been interchanged in the schedule.

Conflict Serializability

- If a schedule S can be transformed into a schedule S' by a series of swaps of non-conflicting instructions, we say that S and S' are conflict equivalent.
- We say that a schedule S is conflict serializable if it is conflict equivalent to a serial schedule.

- ① Concept
- Conflict Scheduling
Serial Testing Method Precedence Graph Method [CNC] Cycle Not Conflict
- ② Testing method
- ③ Conflict Equivalent to Any Serial Schedule

II

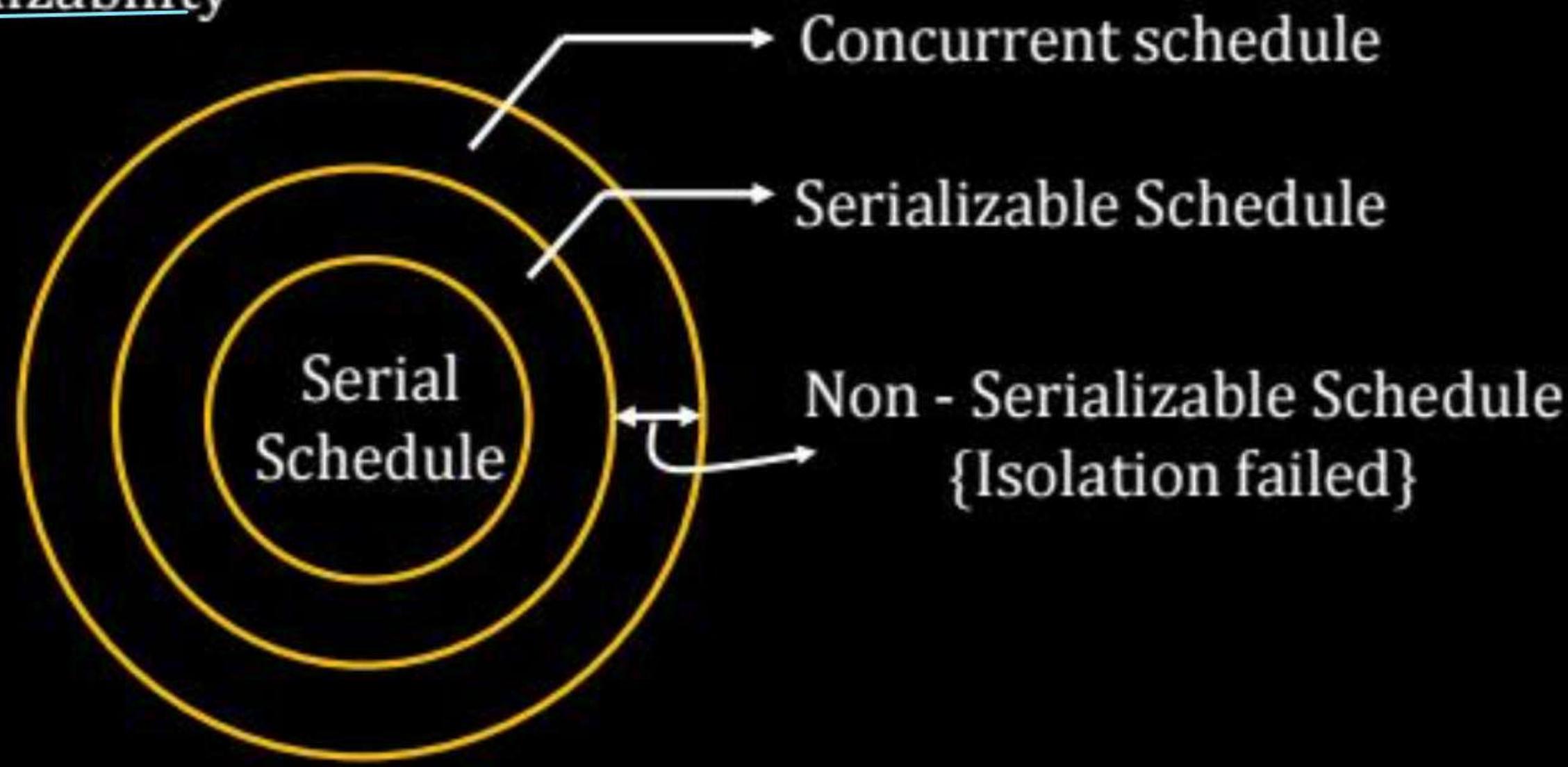
Conflict Equivalence

Serializable Schedule

A Schedule is serializable Schedule if it is equivalent to a Serial Schedule.

(i) Conflict Serializability

(ii) View Serializability



Topological order

Graph traversal algorithm for directed graph

- (i) Visit vertex (v), whose indegree is '0' and delete 'V' from the graph.
- (ii) Repeat (i) for all the vertices of graph.

Conflict Pairs

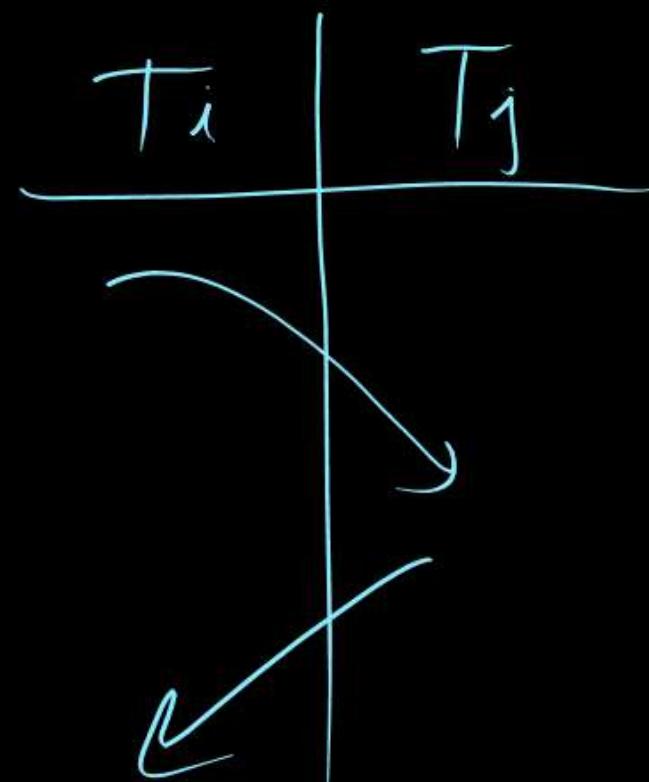
Two operations from Schedule (s) are conflict pair if and only if

- (i) Atleast one write operation.
- (ii) Operation on same data item.
- (iii) Operations are from different transactions.

S: r_i(A).....w_j(A)
S: w_i(A).....r_j(A)
S: w_i(A).....w_j(A)

Conflict Pairs

DATA Item A



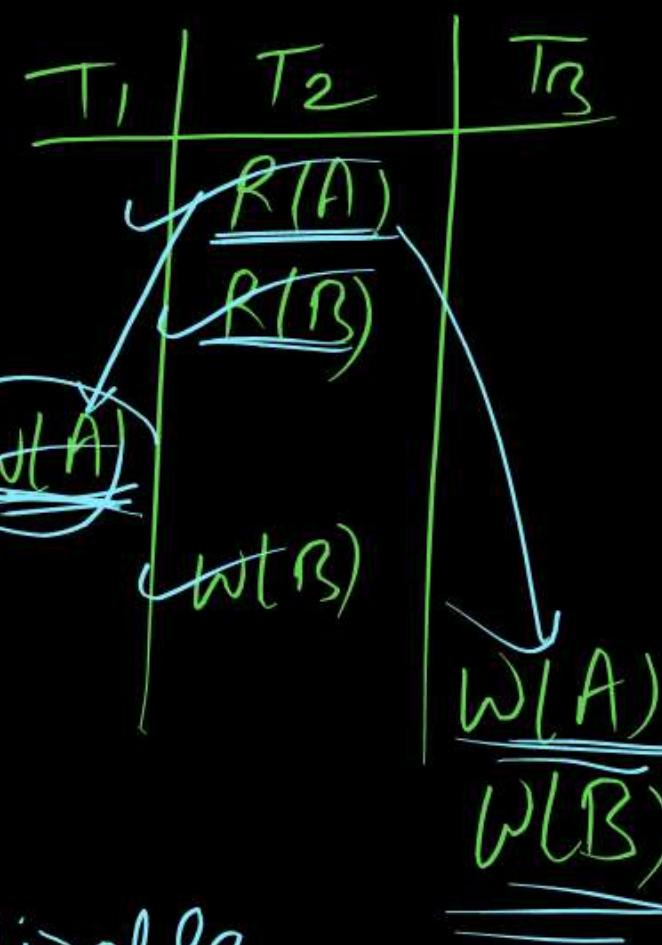
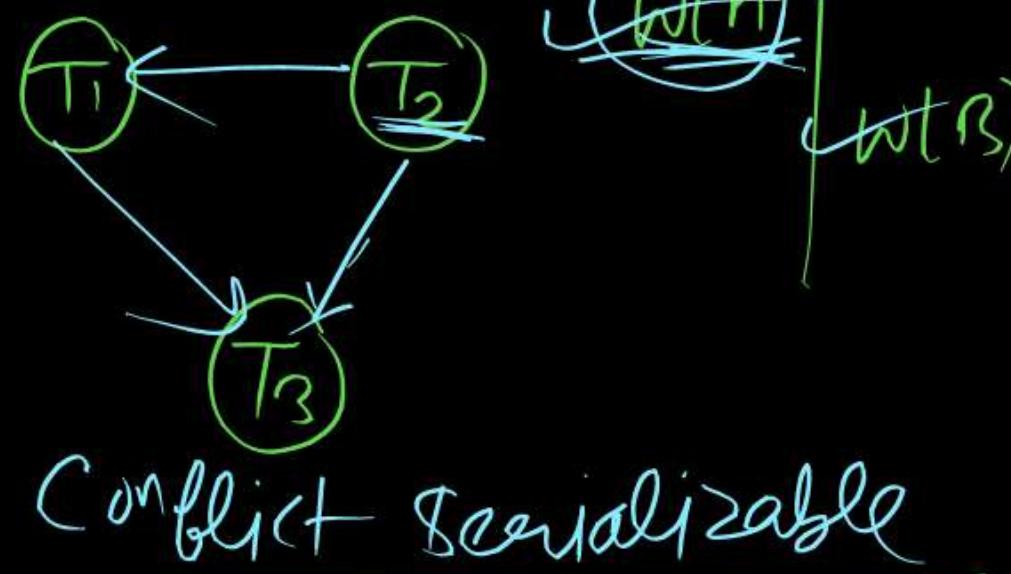
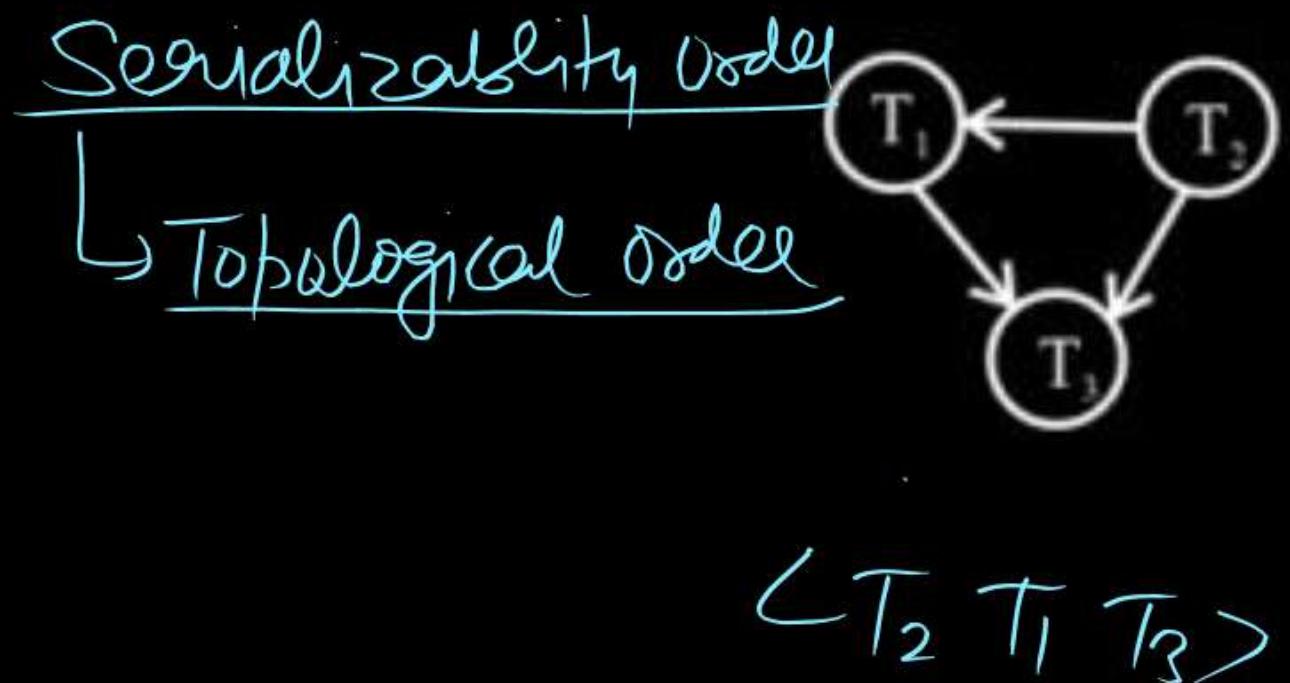
Precedence Graph

P
W

A graph G in which vertex (v) represent the transaction of Schedule and edges (E) represent conflict pair precedence's.

Example:

S : R₂(A) R₂(B) W₁(A) W₂(B) W₃(A) W₃(B)



Conflict Equal Schedule

Schedule S_1 and S_2 are conflict equal Schedule if and only if Schedule S_2 can derived by inter changing consecutive non-conflict pairs of Schedule S_1 .

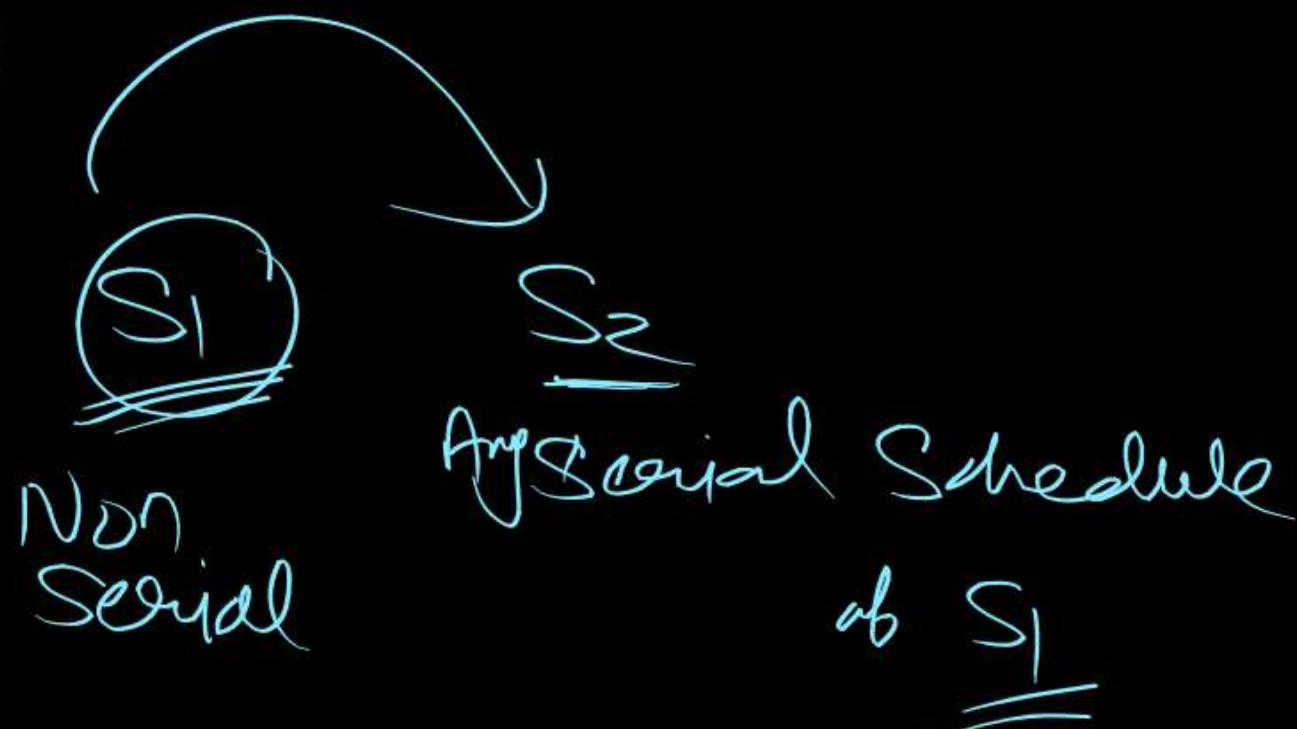


Note:

If Schedule $\underline{S_1}$ and $\underline{S_2}$ have

- (i) Same set of transactions, and
- (ii) Same precedence graph and
- (iii) Acyclic precedence graph then

$\underline{S_1}$ and $\underline{S_2}$ are conflict equal Schedule.

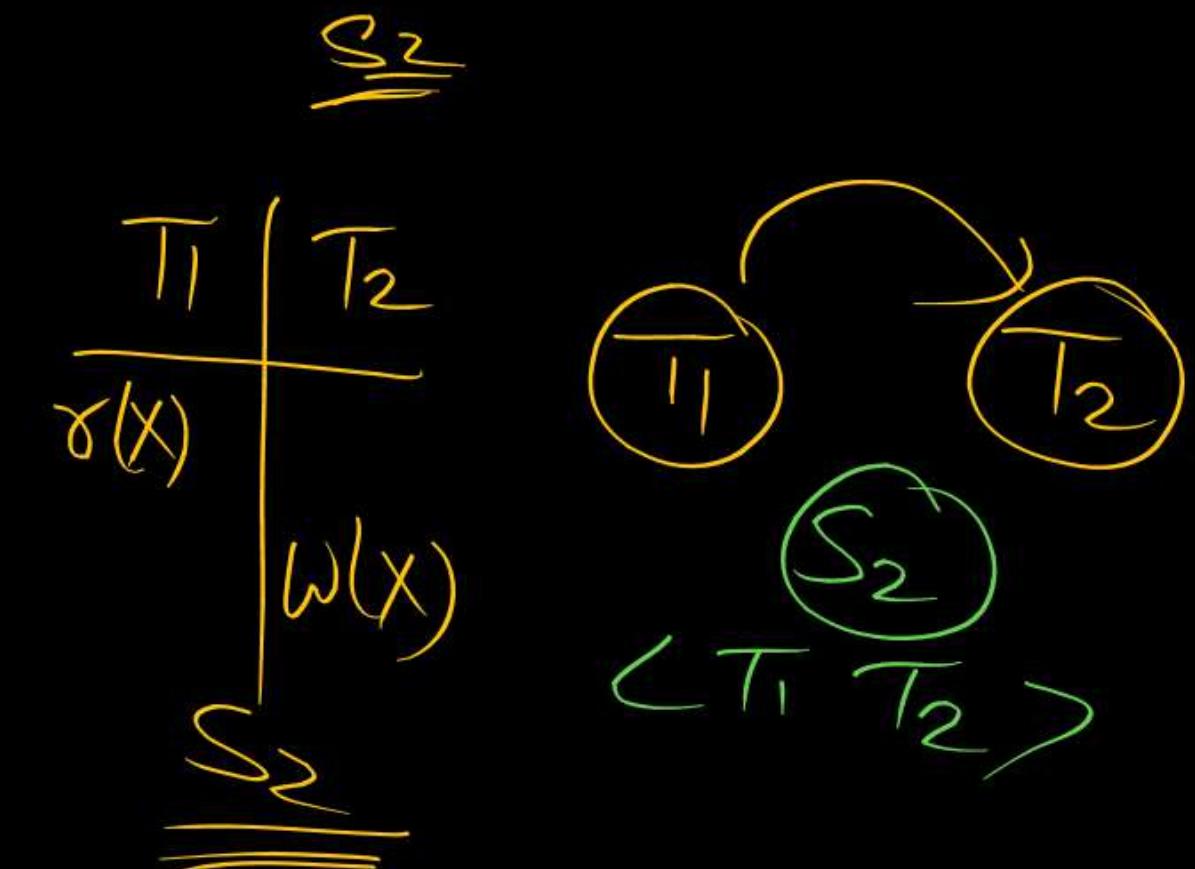
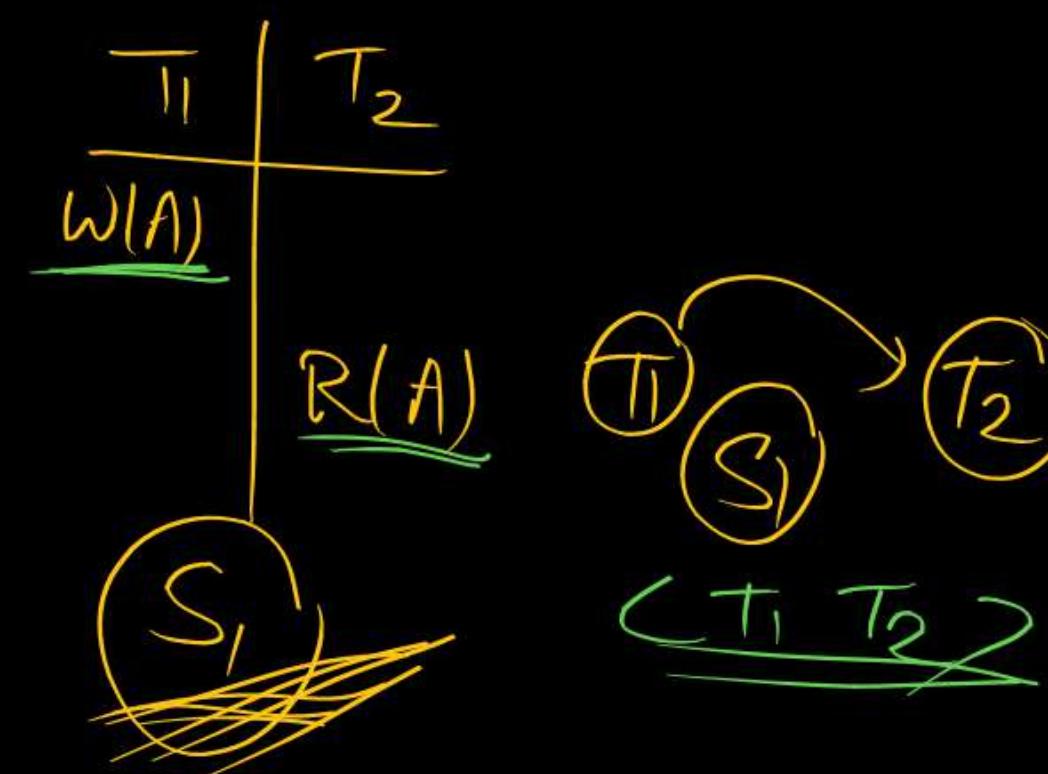


Important Point 1:

1. If S_1, S_2 Schedule are conflict equal then precedence graph of S_1 and S_2 must be same.
2. If S_1 and S_2 have same precedence graph then S_1 and S_2 may or may not conflict equal.

Conflict equal

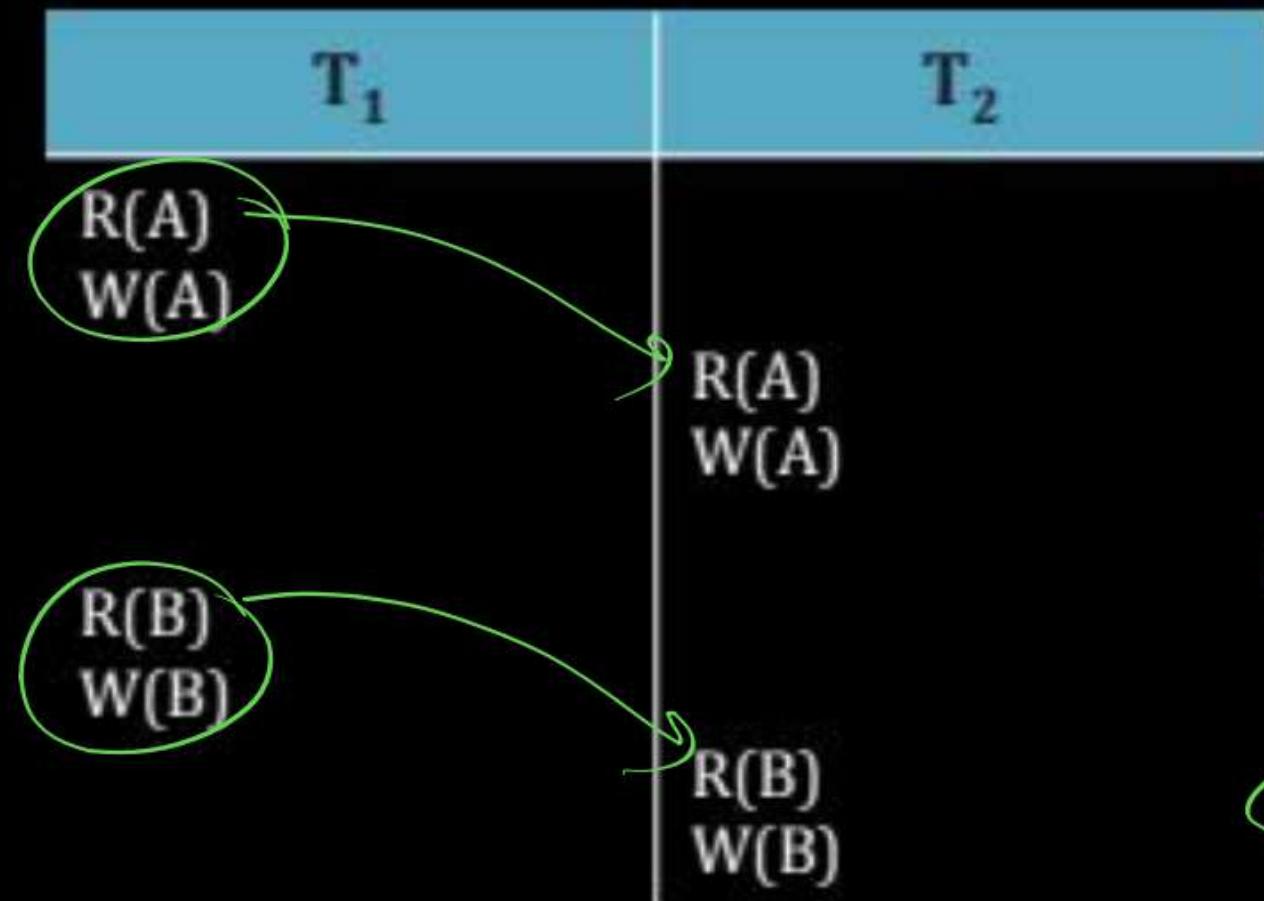
\equiv



Q.

S: R₁(A) W₁(A) R₂(A) W₂(A) R₁(B) W₁(B) R₂(B) W₂(B)

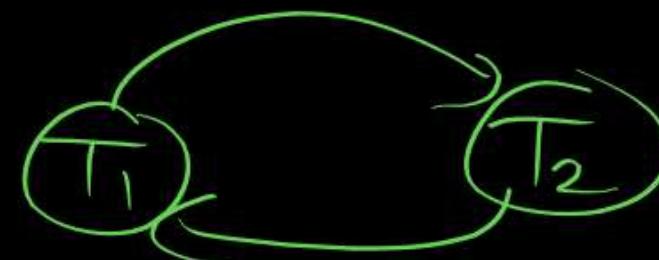
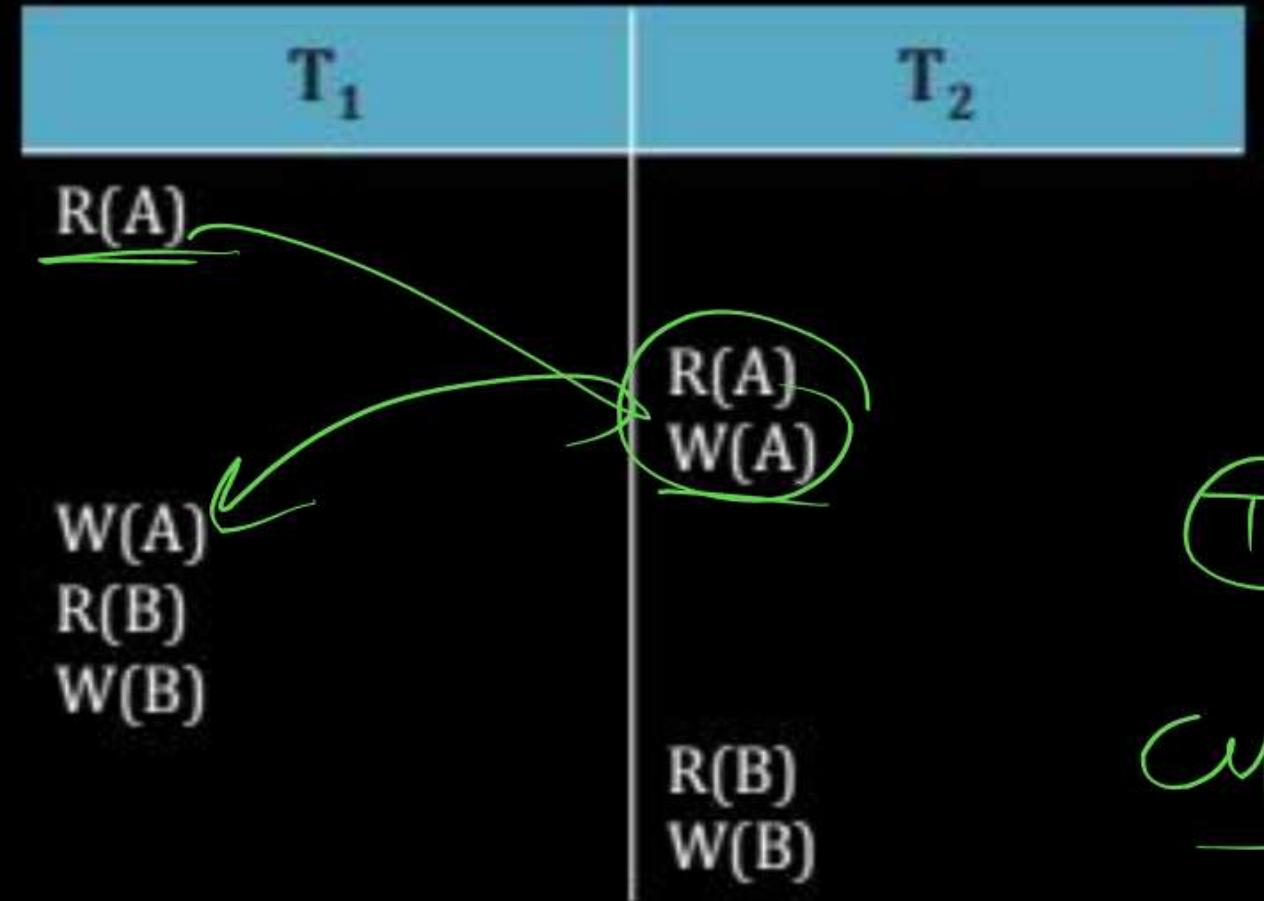
P
W



Conflict serializable

$\langle T_1, T_2 \rangle$

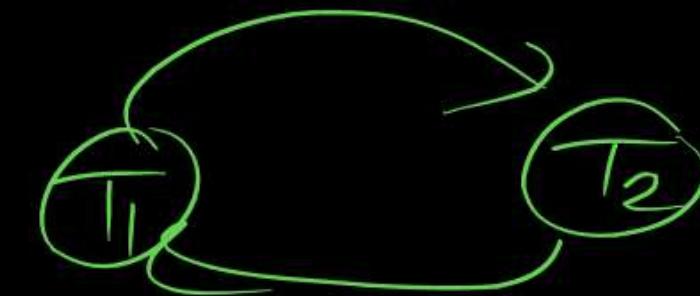
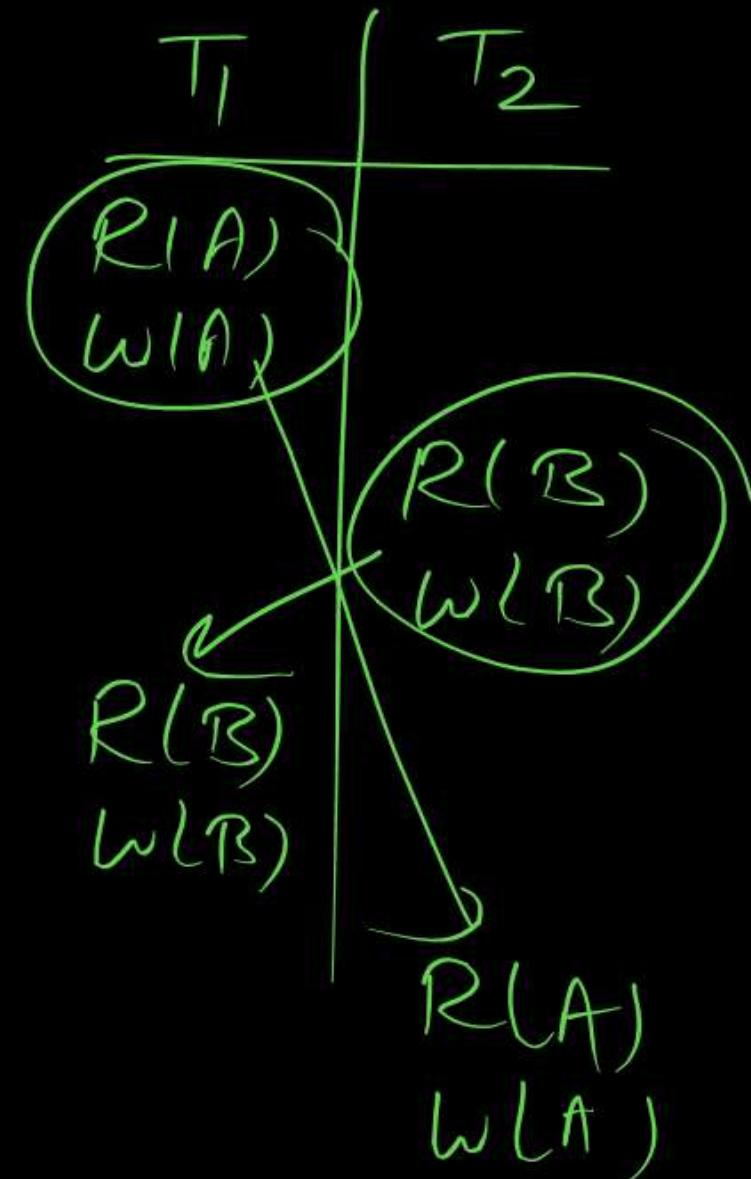
Q.

 $R_1(A) R_2(A) W_2(A) W_1(A) R_1(B) W_1(B) R_2(B) W_2(B)$ P
WCNCWill Not Conflict

Q.

$R_1(A) W_1(A)$ $R_2(B) W_2(B)$ $R_1(B) W_1(B)$ $R_2(A) W_2(A)$

P
W

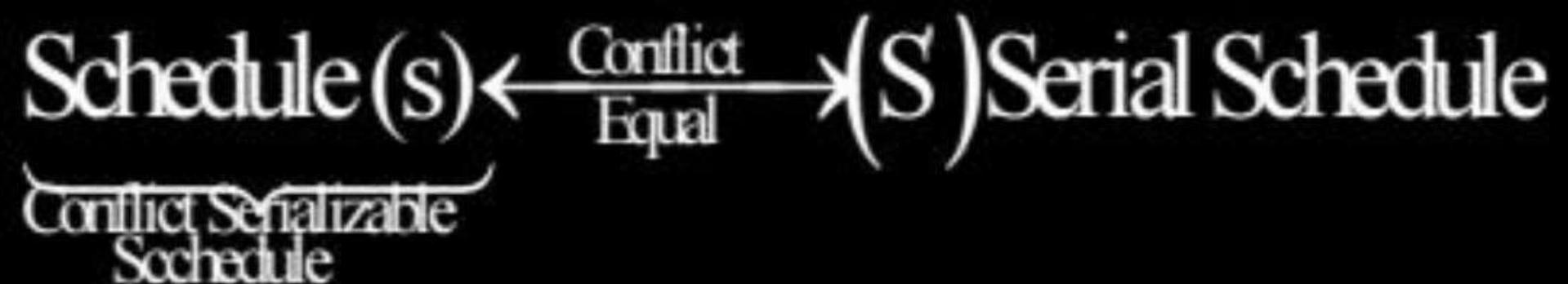


Cycle Not Conflict

Conflict Serializable Schedule



Schedule (s) is conflict serializable Schedule if and only if some serial Schedule (S) must be conflict equal to Schedule (s).



Important Point 2:

Schedule (s) is conflict serializable Schedule if and only if

1. Precedence graph is acyclic and
2. Conflict equal serial schedule are to apological order of precedence graph.

Note:

[No. of serial schedule conflict equal to schedule s] = [No. of topological order of schedules precedence graph]

Conflict equivalent

s_1 & s_2

All Conflict operation
Pair same order

Conflict Serializable

P
W

A schedule is said to be conflict serializable if it is conflict equivalent to a serial schedule.

Same conflicting operation order in C_1 & S_1

$R_1(A) - W_2(A)$
 $W_1(A) - R_2(A)$
 $W_1(A) - W_2(A)$
 $\cancel{R_1(B) - W_2(B)}$
 $W_1(B) - \cancel{R_2(B)}$
 $W_1(B) - W_2(B)$

T_1	T_2
read(A)	
write(A)	
	read(A)
	write(A)
	read(B)
	write(B)
	read(B)
	write(B)

T_1	T_2
read(A)	
write(A)	
read(B)	
write(B)	
read(A)	
write(A)	
read(B)	
write(B)	

S_L

$\cancel{R_1(A) - W_2(A)}$
 $\cancel{W_1(A) - W_2(A)}$
 $W_1(A) - \cancel{R_2(A)}$
 $W_1(A) - \cancel{W_2(A)}$

Q.

Consider the following four schedules due to three transactions (indicated by the subscript) using read and write on a data item x , denoted by $r(x)$ and $w(x)$ respectively. Which one of them is conflict serializable?

[2014(Set-1): 2 Marks]

- A $r_1(x); r_2(x); w_1(x); r_3(x); w_2(x)$
- B $r_2(x); r_1(x); w_2(x); r_3(x); w_1(x)$
- C $r_3(x); r_2(x); r_1(x); w_2(x); w_1(x)$
- D $r_2(x); w_2(x); r_3(x); r_1(x); w_1(x)$

Conflict Equivalent Schedule



Two schedule are said to be conflict equivalent, if all conflicting operations in both the schedules must be executed in the same order.

Q.

Consider a schedule of transactions T1 and T2;

T1	RA			RC	WD	WB	commit	
T2		RB	WB	RD	WC		commit	

$$\begin{aligned}
 & R_2(B) - W_1(B) \\
 & W_2(B) - W_1(B) \\
 & R_1(C) - W_2(C) \\
 & R_2(D) - W_1(D)
 \end{aligned}$$

Here, RX stands for “Read(X)” and WX stands for “Write(X)”. Which one of the following schedules is conflict equivalent to the above schedule?

[MCQ:2020-2M]

A

T1				RA	RC	WD	WB	Commit	
T2	RB	WB	RD				WC	Commit	

$$R_2(B) - W_1(B)$$

B

T1	RA	RC	WD	WB			Commit		
T2				RB	WB	RD	WC	Commit	

$$W_2(B) - W_1(B)$$

C

T1	RA	RC	WD			WB		Commit	
T2				RB	WB	RD	WC	Commit	

$$R_1(C) - W_2(C)$$

D

T1				RA	RC	WD	WB	Commit	
T2	RB	WB	RD	WC				Commit	

$$R_2(D) - W_1(D)$$

P
W

View Serializability

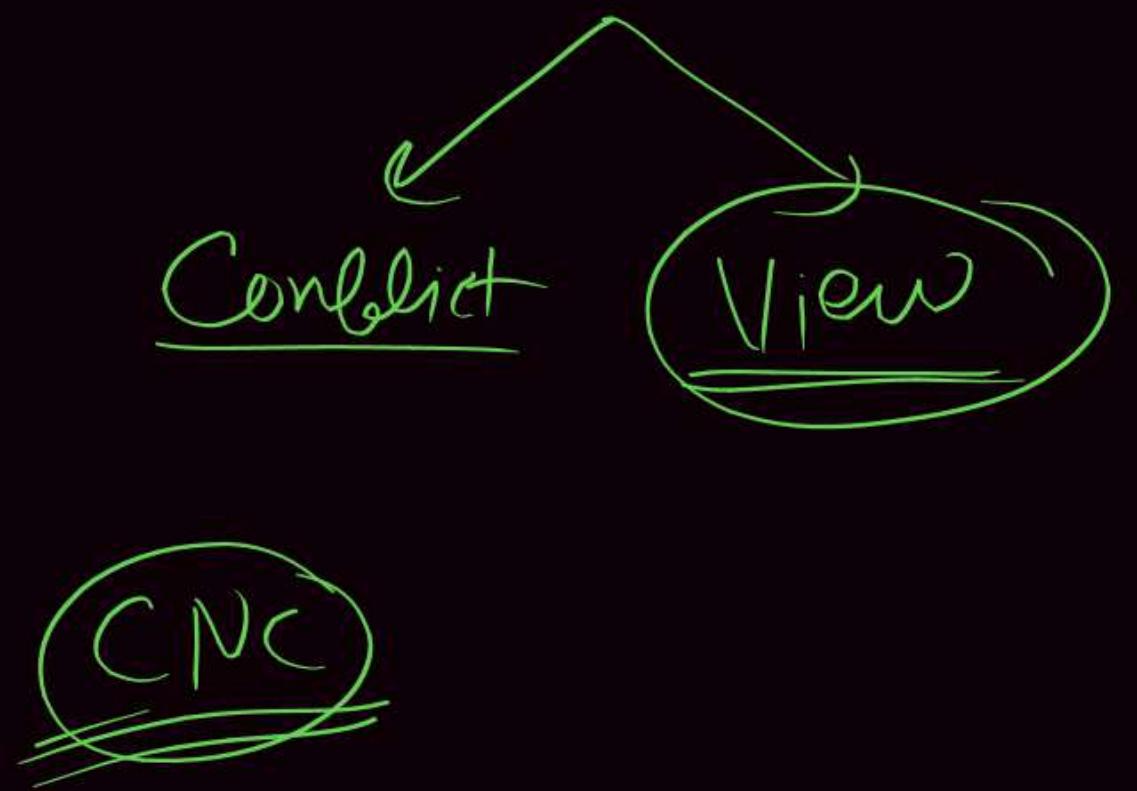
S

(S')

Let S and S' be two schedules with the same set of transactions. S and S' are **view equivalent** if the following three conditions are met, for each data item Q

1. If in schedule S, transaction T_i reads the initial value of Q then in schedule S' also transaction T_i must read the initial value of Q.
2. If in schedule S transaction T_i executes read(Q), and that value was produced by transaction T_j (if any), then in schedule S' also transaction T_i must read the value of Q that was produced by the same write(Q) operation of transaction T_j .
3. The transaction (if any) that performs the final write(Q) operation in schedule S must also perform the final write(Q) operation in schedule S'.

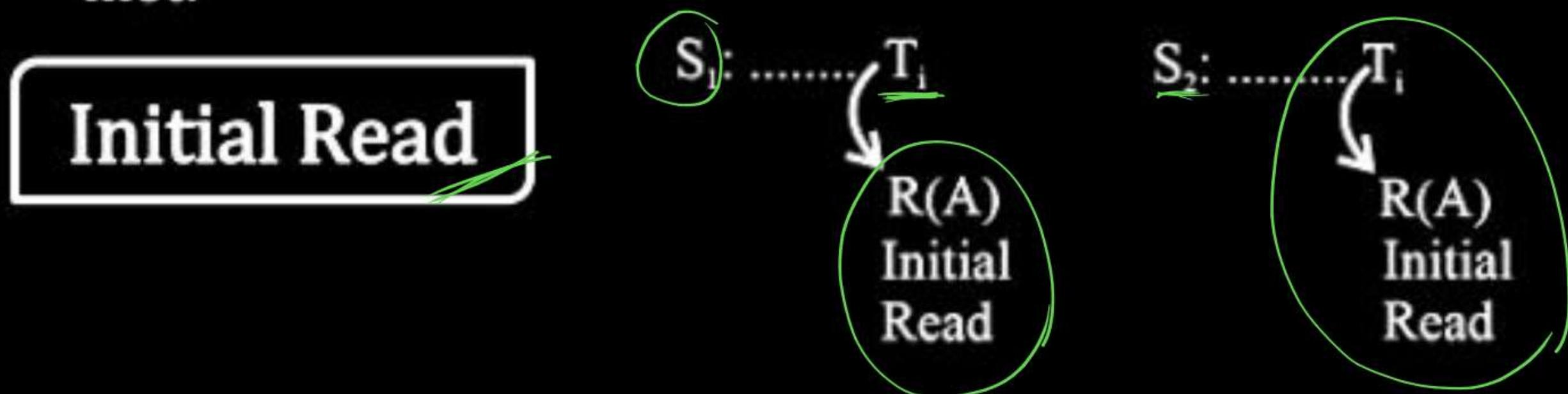
On each Data Item



- ① Initial Read
- ② Final Write
- ③ Write - Read
Sequence

View Serializable Schedule

- Schedule (s) is view serializable if and only if some serial schedule (s') view equal to schedule (s).
- Let S_1 and S_2 be two schedules with the same set of transactions, S_1 and S_2 are view equal if the following three conditions are met:



For each data item A , if transaction T_i reads the initial value of A in Schedule S_1 , then transaction T_i must, in Schedule S_2 , also read the initial value of A .

View Serializability

- **View Serializable Schedule:** View equivalent serial schedule.

- **View Equivalent:** S_1 and S_2 said to be view equivalent.

Only if

- [1] initial reads of S_1 and S_2 should be same.

Example

Tolakhu

$S_1 \neq S_2$

S_1

S_2

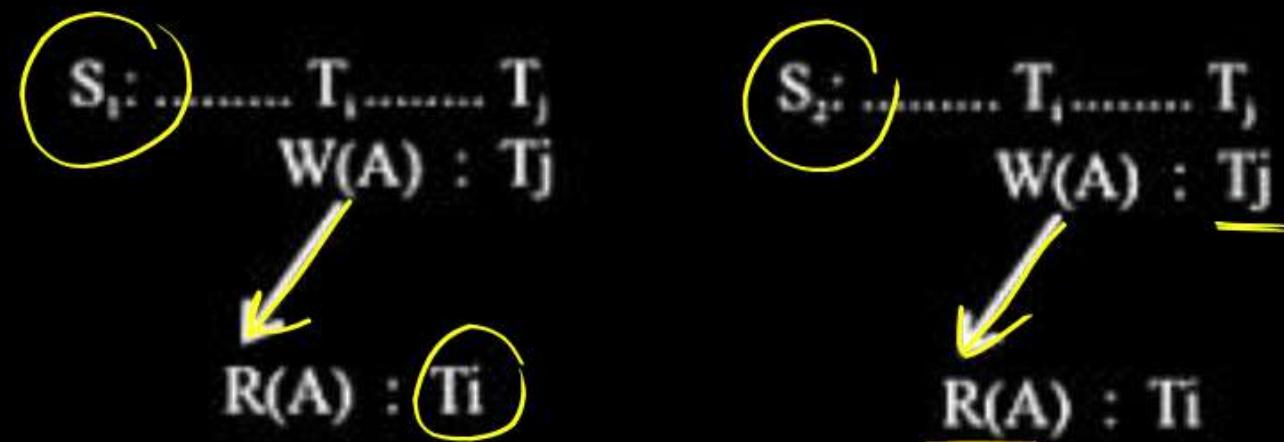
T ₁	T ₂	T ₃	T ₁	T ₂	T ₃
$A=102$ $R(A)$			$A=104$ $R(A)$		
$B=102$ $R(B)$		$W(B)$	$B=500$ $R(B)$		$W(B)$

Initial Read on A $\rightarrow T_1$

Initial Read on B $\rightarrow T_1$

~~B \neq Not T₁~~

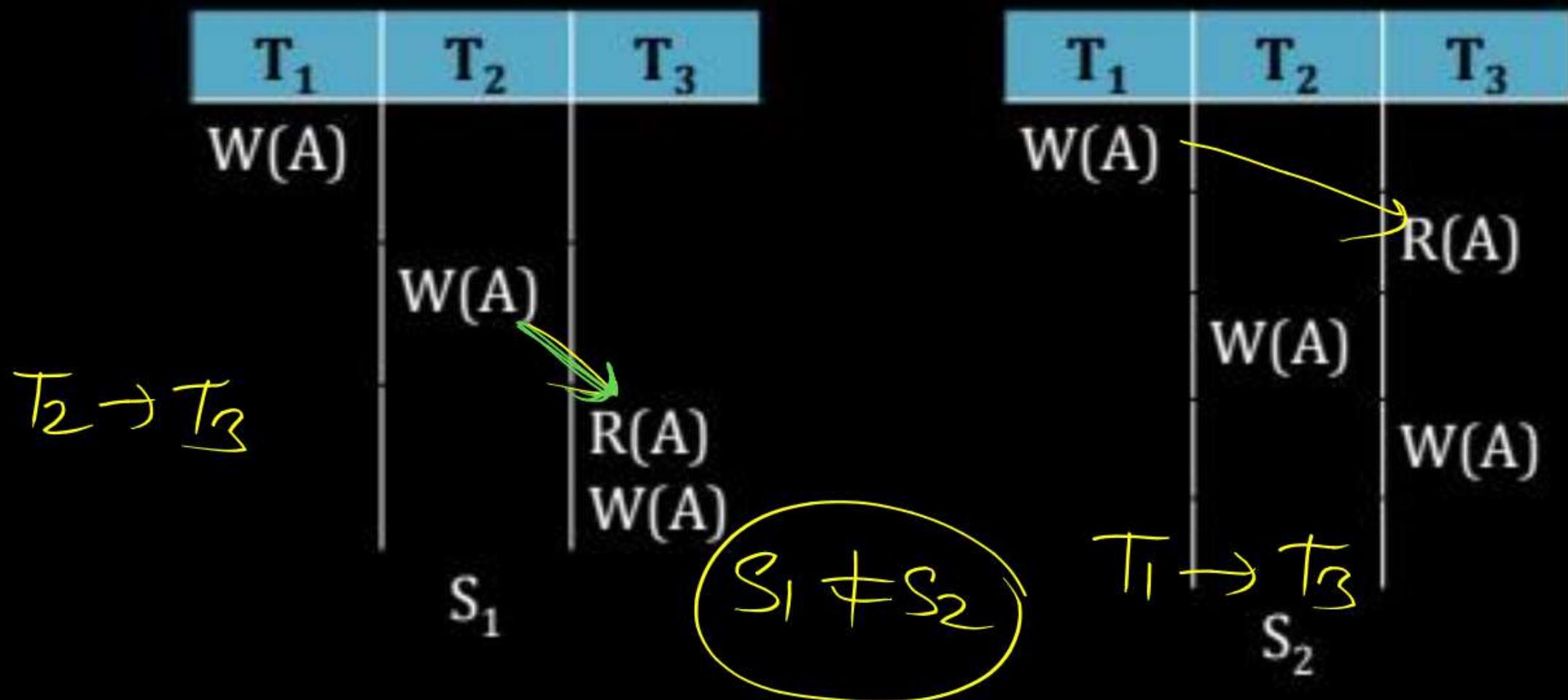
Updated Read



For each data item A if transaction T_i perform Read (A) in Schedule S_1 and that value was produced by transaction T_j , then transaction T_i must in Schedule S_2 also read the value of A that is produced by the transaction T_j .

View Serializability

Write-Read sequence should also be equal. (Updated Reads should be same)



Final Write



$S_1 : \dots \circled{T_i} \dots$

Final write of A : $w(A)$
Last write

$S_2 : \dots \circled{T_i} \dots$

Final write of A : $w(A)$
Last write

For each data item A, the transaction that perform the final write (A) operation in schedule (S_1) must perform the final write (A) operation in schedule S_2 .

View Serializability

Final updations for every data item should be same in S_1 and S_2

T ₁	T ₂	T ₃
W(A)		
W(B)	W(A)	W(A)

$R \rightarrow T_3$

$A \rightarrow T_3$

S_1

~~$R \rightarrow T_3$~~

~~$A \rightarrow T_3$~~

$W(A)$

$W(B)$

$S_1 \neq S_2$

$R \rightarrow T_3$

$W(A)$

$W(B)$

S_2

$W(A)$

$W(B)$

$A \rightarrow T_2$

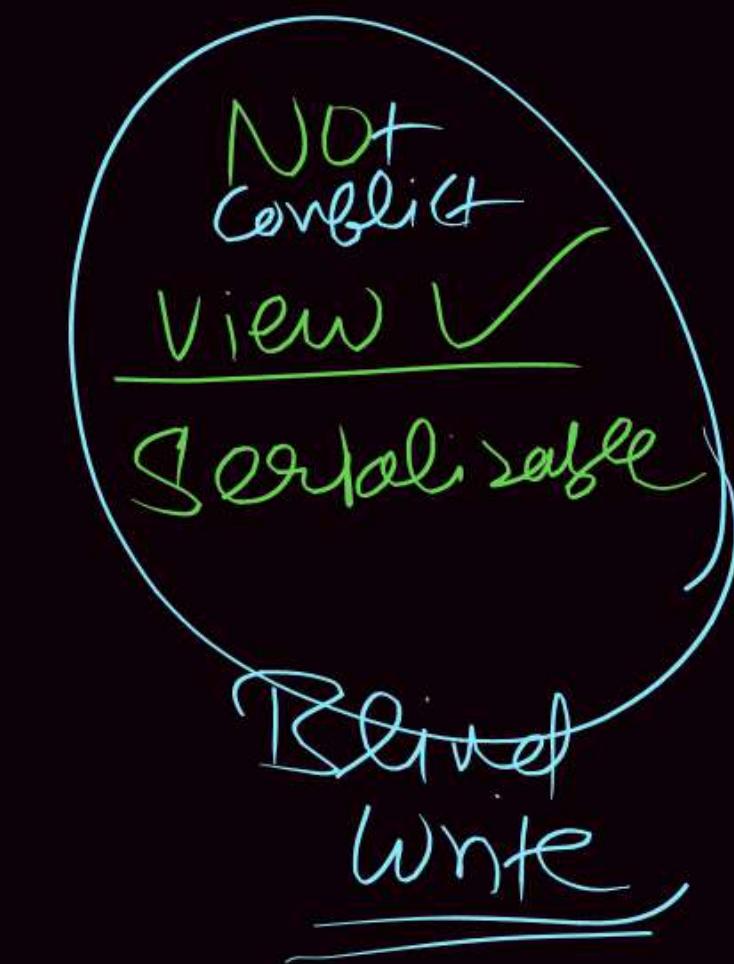
\times



Conflict ✓

Views ✓ No Need
to check

Conflict serializable



NO

NO

Not Serializable

Important Point 3

- If schedule S is conflict serializable schedule, then S is also view serializable schedule.
- If schedule s is not conflict serializable then it may or may not view serializable
- Every view serializable schedule that is not conflict serializable.

View Serializability(Blind Write)

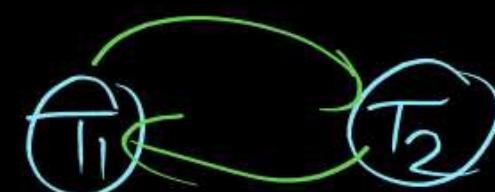
P
W

- A schedule S is view serializable if it is view equivalent to a serial schedule.
- Every conflict serializable schedule is also view serializable.
- Below is a schedule which is view-serializable but not conflict serializable.

Note:

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

T_{27}	T_{28}	T_{29}
read(A)		
	write(Q)	
	write(Q)	write(Q)



Cycle NOT Conflict



View
 $\langle T_1 \ T_2 \ T_3 \rangle$

Q.

Consider the following schedule S of transactions T_1 and T_2 :

Which of the following is TRUE about the schedule S? [2004: 2 Marks]

P
W

- A S is serializable only as T_1, T_2
- B S is serializable only as T_2, T_1
- C S is serializable both as T_1, T_2 and T_2, T_1
- D S is not serializable either as T_1 or as T_2

T_1	T_2
<u>Read(A)</u> $A = A - 10$ <u>Write(A)</u> Read(B) B = B + 10 Write(B)	 <u>Read(A)</u> $Temp = 0.2 * A$ <u>Write(A)</u> Read(B) B = B + Temp Write(B)

Classification based on Recoverability

The concurrent execution may leads:

1. Irrecoverable schedule
2. Cascading rollback problem
3. Lost update problem

The above problems can occur even if the schedule is serializable (Integrity satisfied)

Dependency

T_1	T_2
<u>W(A)</u>	
:	
Commit	<u>R(A)</u>

YES

T_1	T_2
<u>W(A)</u>	
commit	
	<u>R(A)</u>

No Deb.

T_1	T_2
<u>W(A)</u>	
undo	<u>A=20</u>
rollback	<u>R(A)</u>

NO

T_1	T_2
<u>W(A)</u>	
	<u>w(A)</u>

No Deb

T_1	T_2
<u>W(A)</u>	
	<u>W(A)</u>

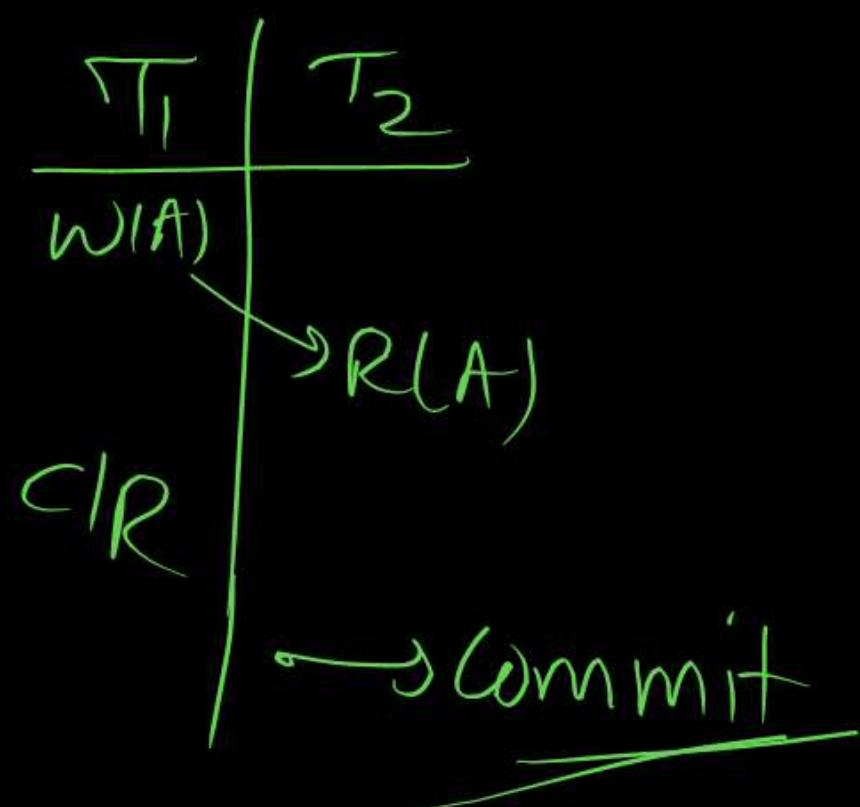
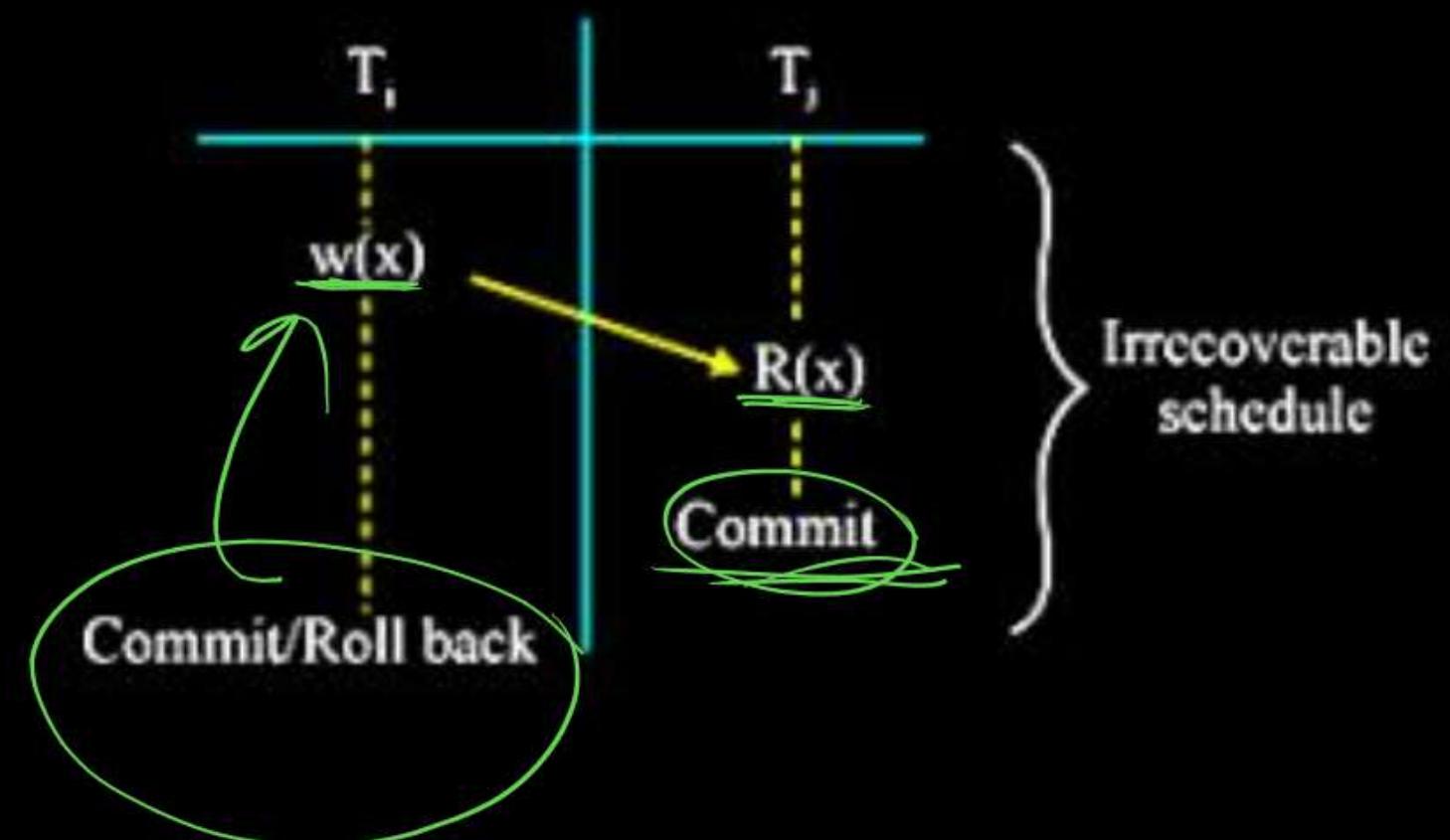
R(A) ↗

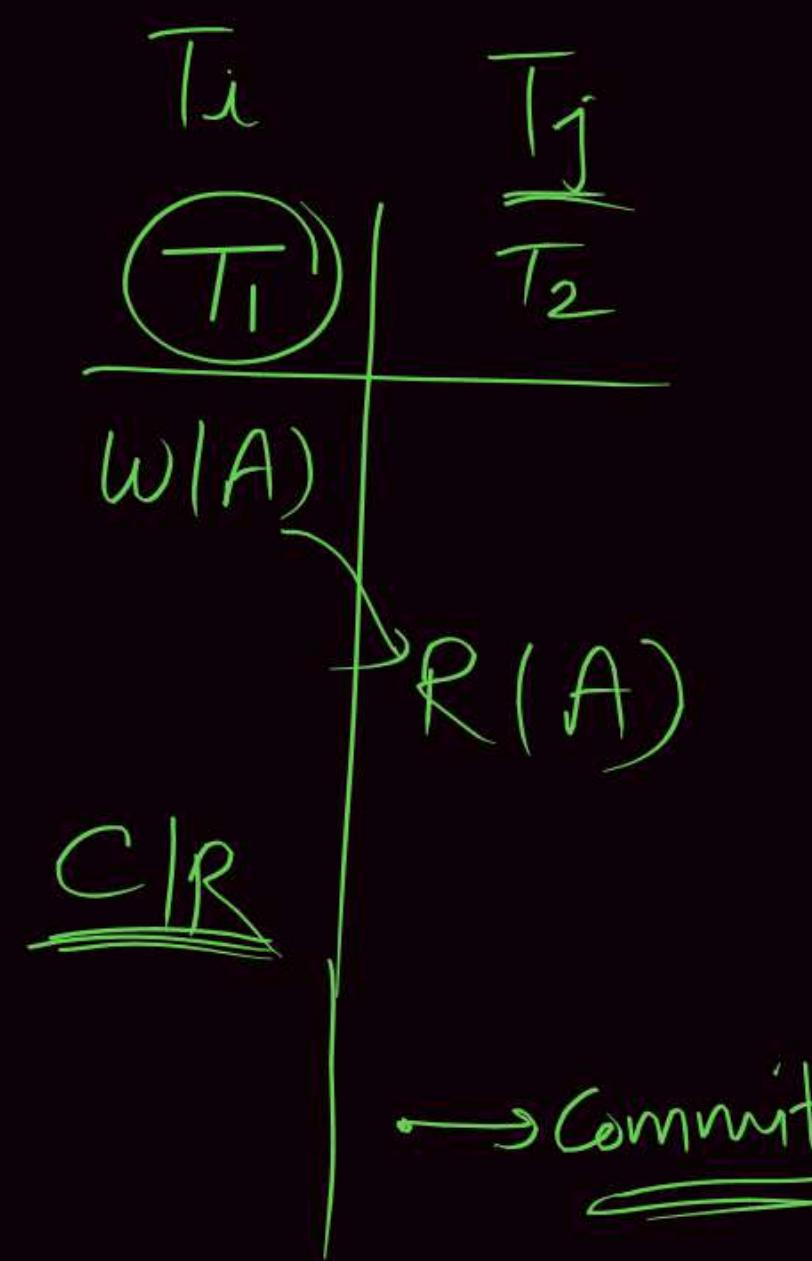
T_1	T_2
<u>W(A)</u>	
R(B) ↗	<u>W(B)</u>

T_2 Deb on T_1
 T_1 Dep. on T_2

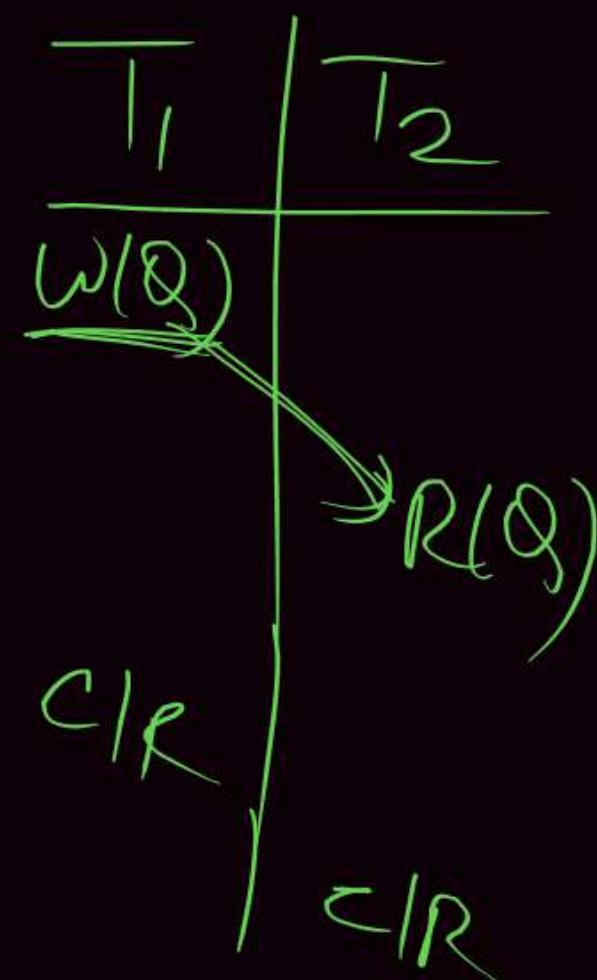
Irrecoverable Schedule

- A schedule(s) is irrecoverable if and only if transaction T_j reads data item 'x', which is updated by T_i and commit of T_j before commit/rollback of transaction T_i .
- Irrecoverable means unable to recover or rollback.





WR Problem
 Uncommitted | Dirty Read



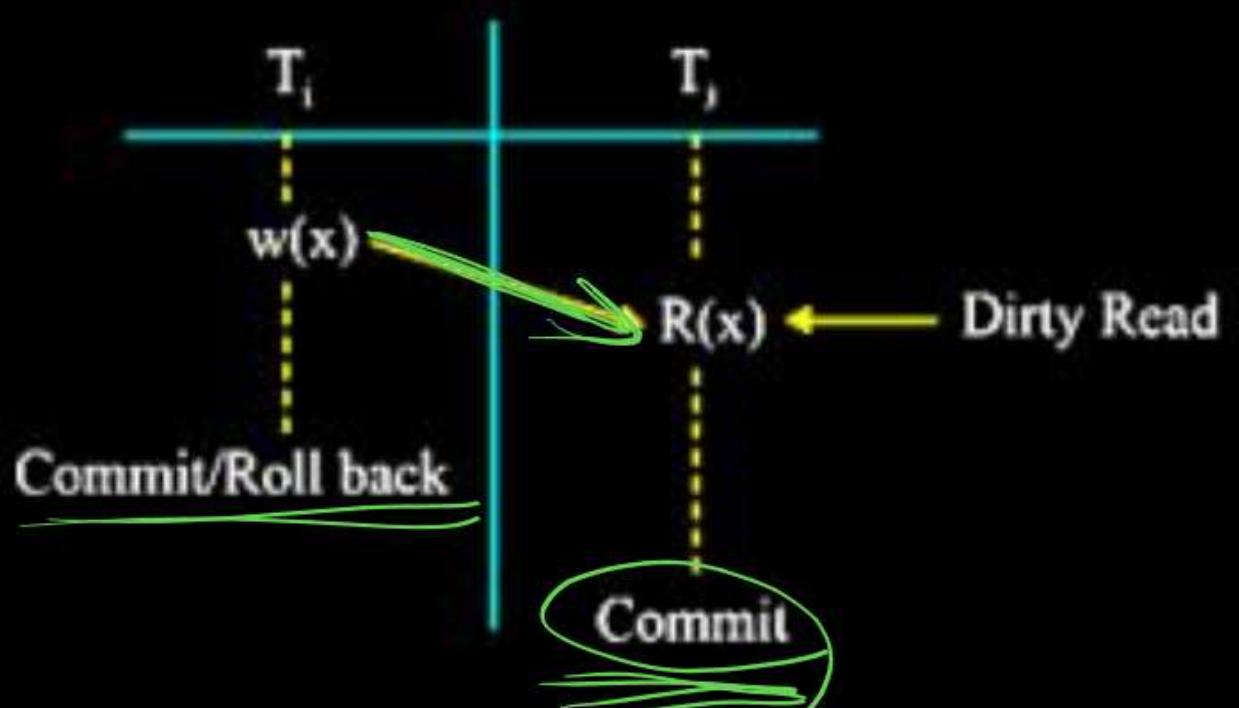
Recoverable Schedule

A schedule(s) is recoverable if and only if

- (I) No uncommitted read (no dirty ready in schedule(s)).

Or

- (II) If transaction T_j reads data item ' x ' which is updated by transaction ' T_i ', then commit of T_j must be delayed until commit/rollback of T_i .



Examples

(1)

T_1	T_2
w(A)	R ₁ (A) C ₂ (commit)
<u>Rollback</u>	

T_2 Recoverable

(3)

T_1	T_2
w(A) C R	R(A)
<u>✓</u>	

(5)

T_1	T_2
w(A)	R A rollback
C R	<u>✓</u> <u>Rel</u>

(2)

T_1	T_2
w(A)	R(A) C ₂
<u>C₁</u>	

Non Recoverable

(4)

T_1	T_2
w(A)	w(A) R(A) commit
<u>Commit/ Rollback</u>	<u>Recovered</u>

(6)

T_1	T_2
w(A)	R(A)
C R	<u>✓</u> <u>Rel</u> C R

NOTE: Recoverable schedule may or may not be free from

- WR problem / uncommitted Read
- RW Problem
- WW Problem

T ₁	T ₂
R(A)	W(A)
Commit	Commit

Recoverable
But RW Problem

T ₁	T ₂
W(A)	W(A)
Commit	Commit

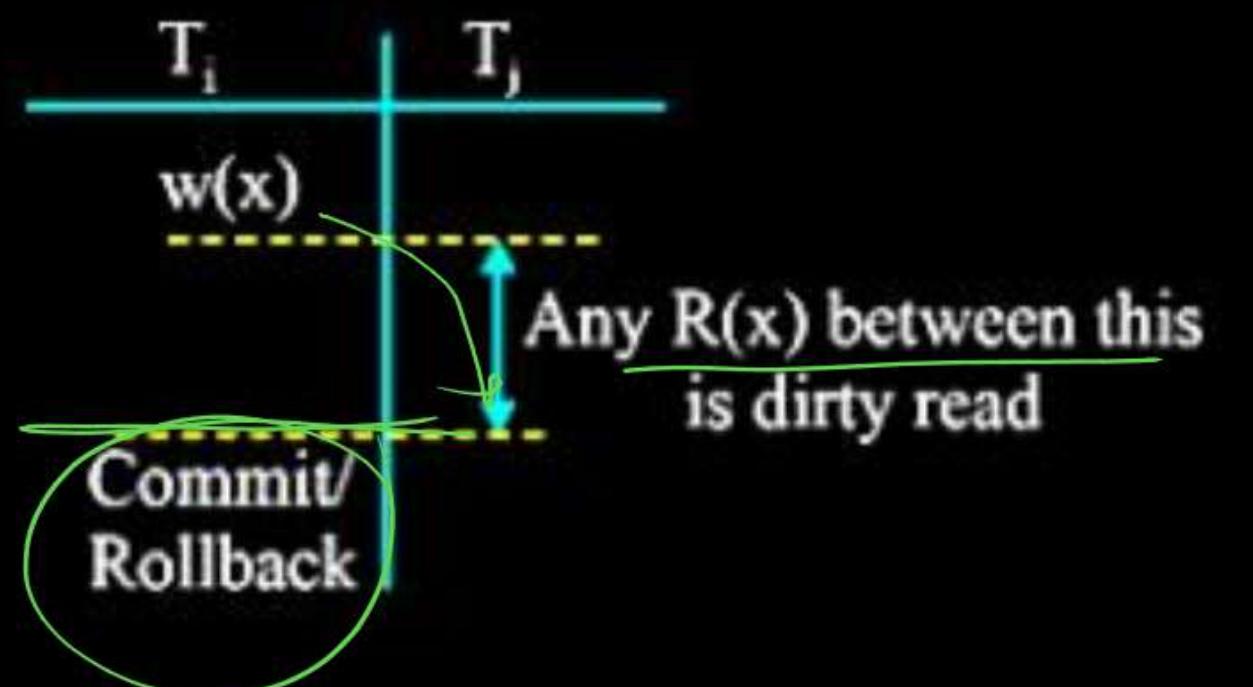
Recoverable
But WW Problem

T ₁	T ₂
R A	R(A)
W(A)	R(B)
W(B)	Commit
Commit	Commit

✓ Recoverable
But WR Problem

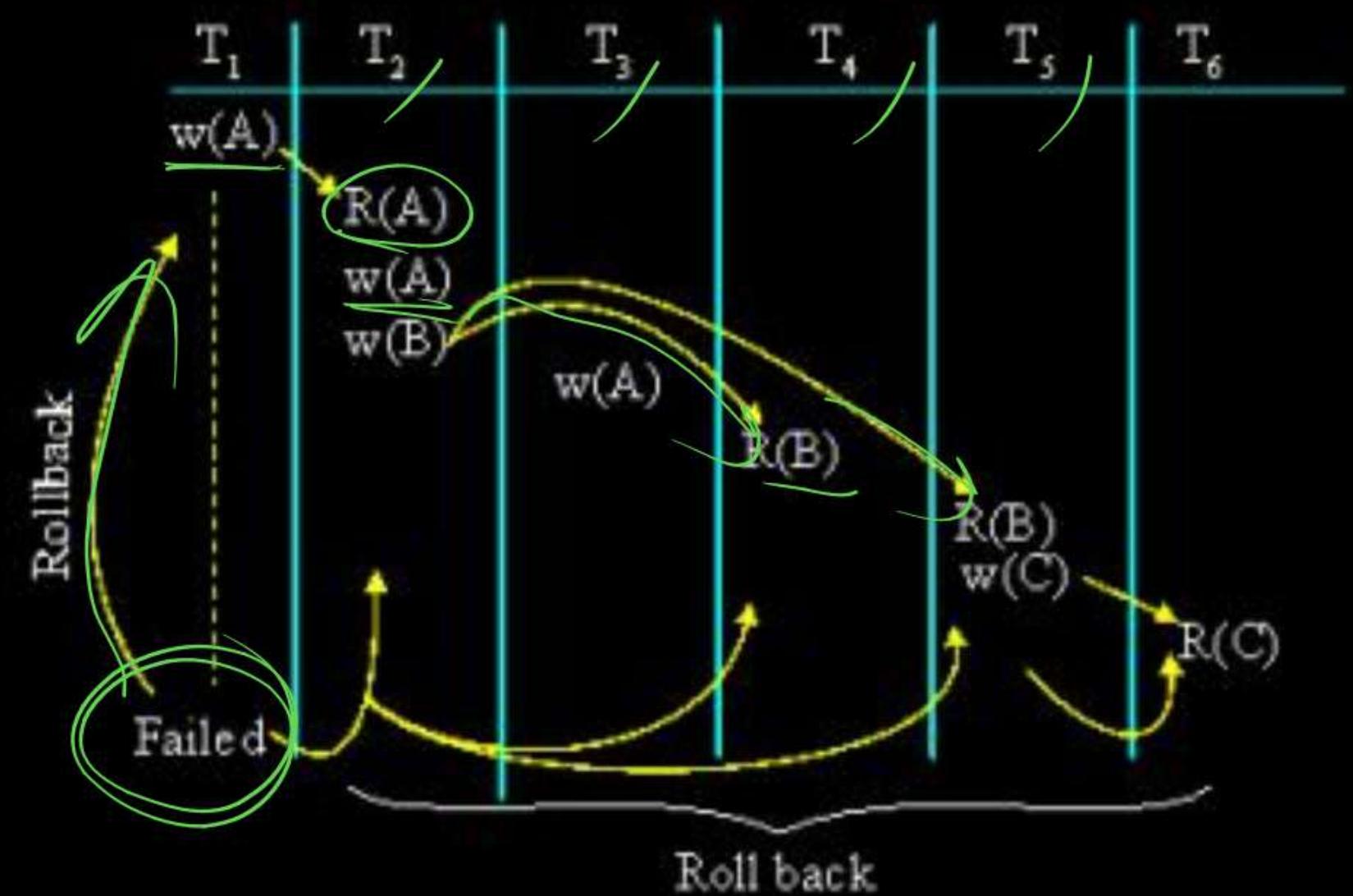
Uncommitted Read (Dirty Read)

Transaction T_j reads data item 'x' which is updated by uncommitted transaction T_i .



Cascading Rollback Schedule (Problem)

When some transaction reads data items which is updated by some other transaction then because of failure of the transaction that updated the data item may rollback all the dependent transaction.



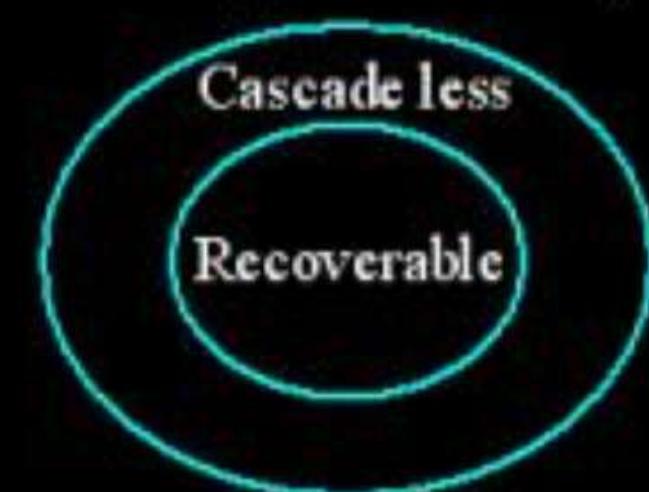
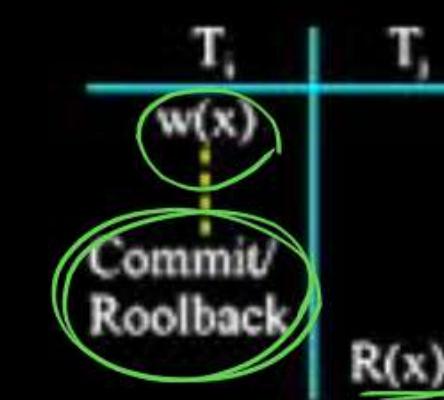
Failure of T_1 forced to rollback T_2, T_4, T_5 and T_6 .

- **Disadvantage of Cascading Rollback**

1. It waste the CPU execution time.
2. Wastage of I/O access cost.

Cascadeless Rollback Recoverable Schedule

- No dirty read in the schedule.
- Transaction T_j should delay read (x) which is updated by T_i , until T_i commit or rollback.



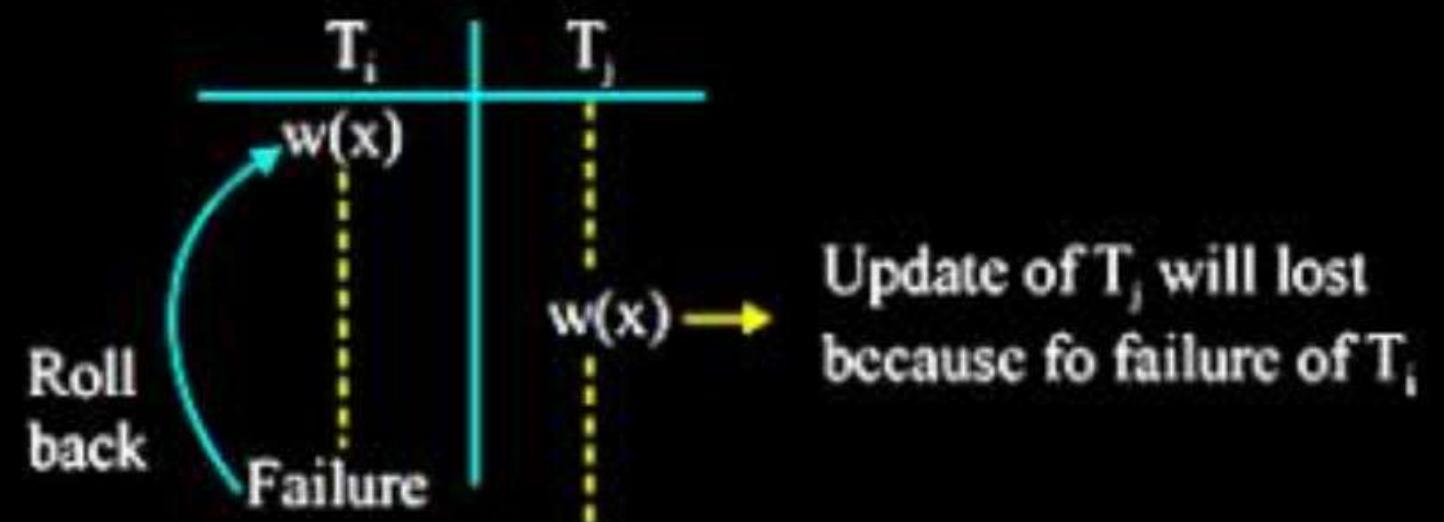
T_1	T_2
$w(A)$	
$c R$	$R(A)$

No Uncommitted Read

No Cascading Rollback

Lost Update Problem

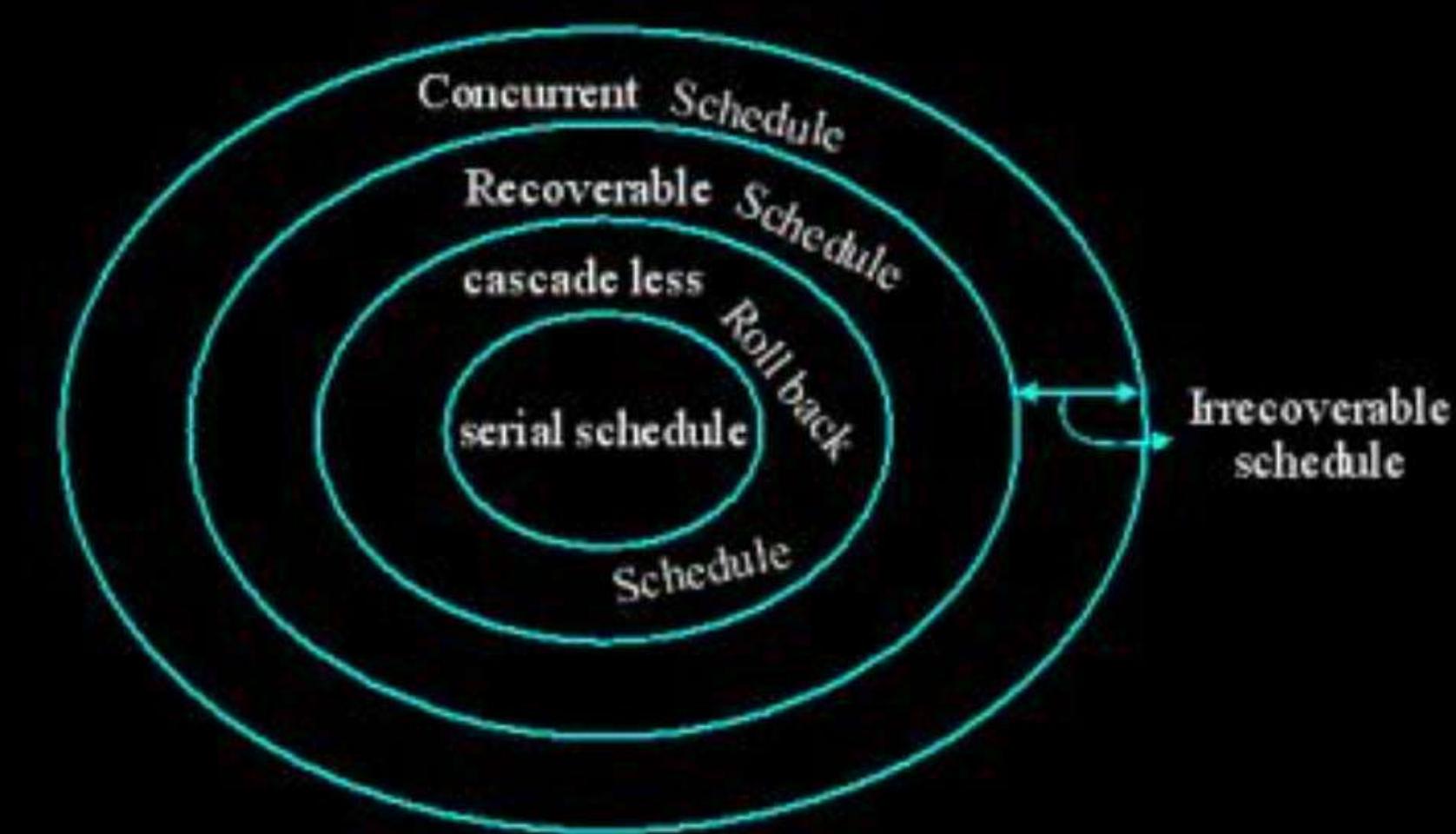
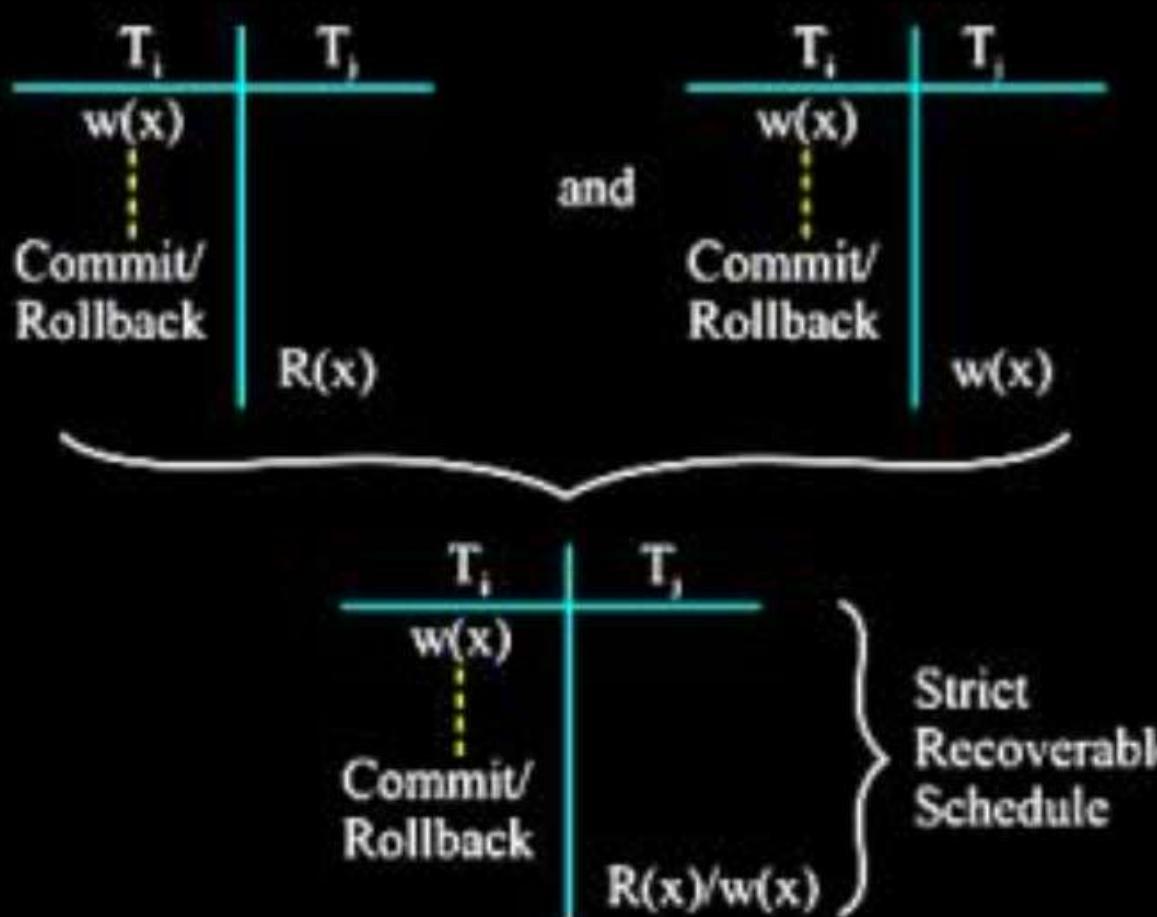
Lost update problem occur if T_j write (x) which is already written by uncommitted transaction T_i .



Strict Recoverable schedule

A schedule must satisfy

1. Cascadeless rollback recoverable schedule and
2. If T_i writes (x) then other transaction T_j write (x) must be delayed until T_i commit or rollback.



τ_1	τ_2
<u>W(A)</u>	
<u>C R</u>	Commit

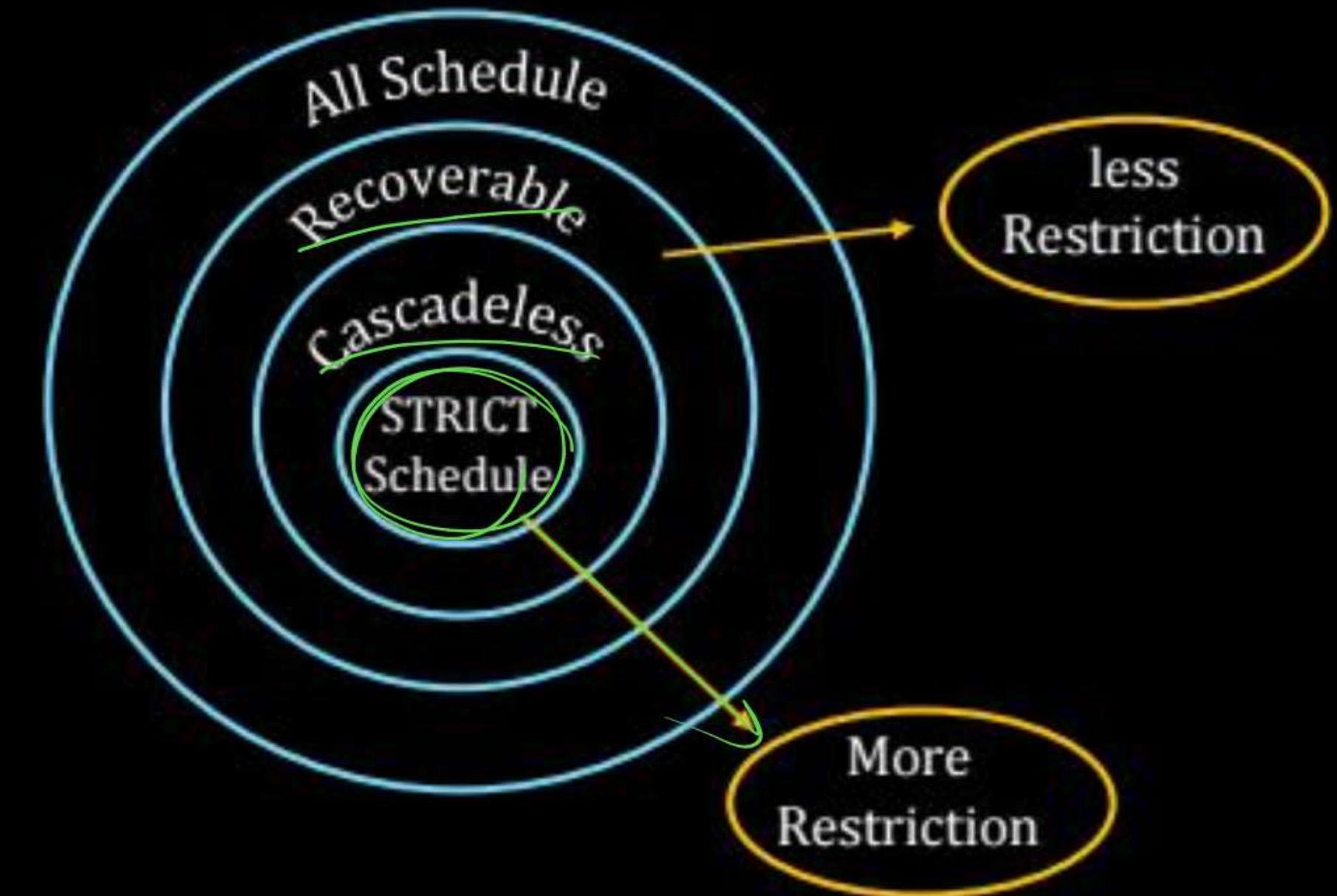
Recoverable

τ_1	τ_2
<u>W(A)</u>	
<u>C R</u>	R(A)

Cascadeless

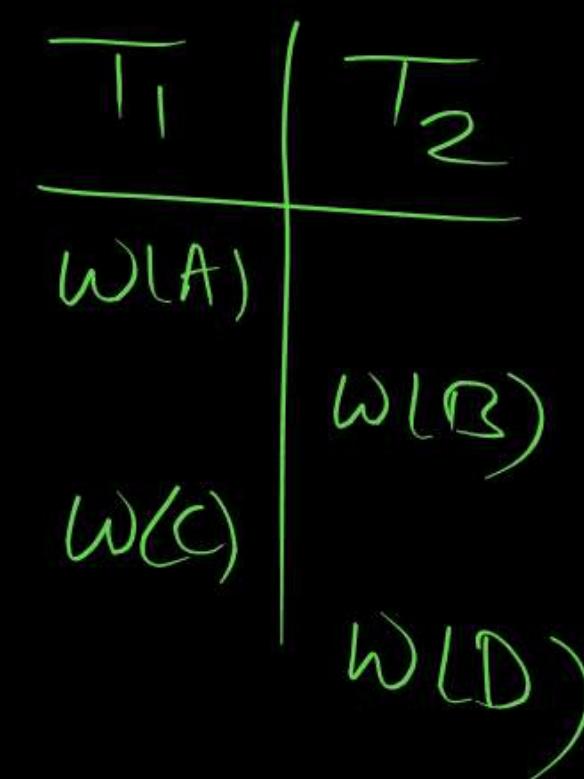
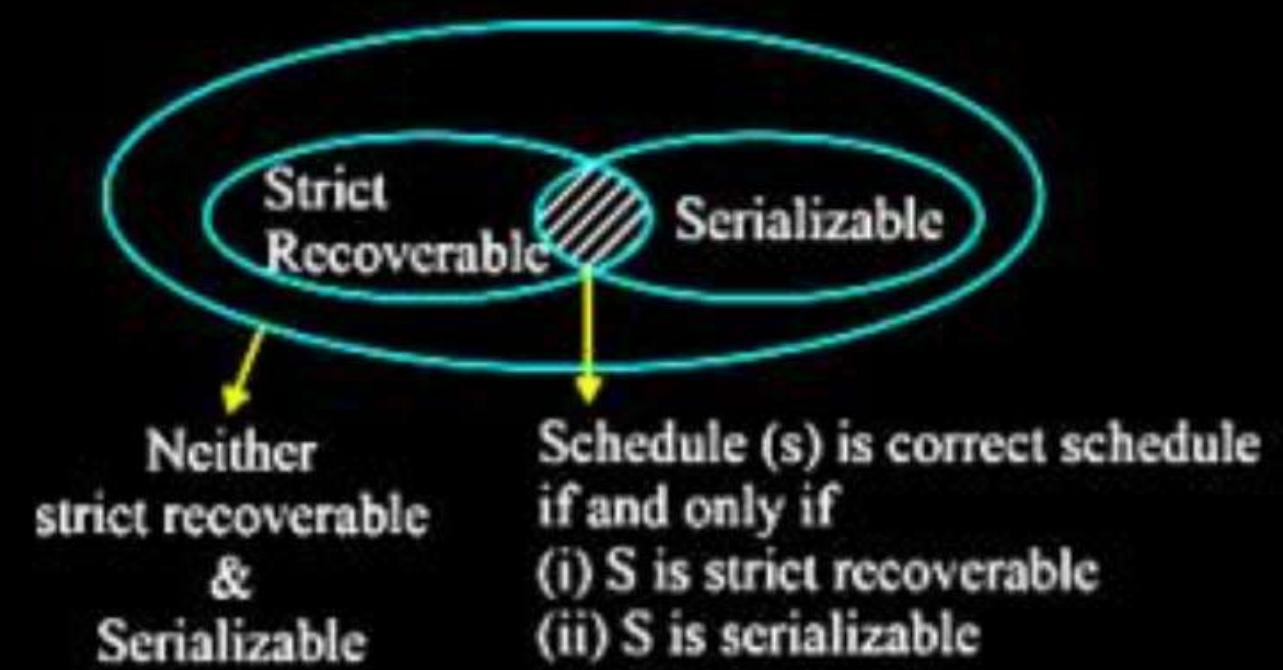
τ_1	τ_2
<u>W(A)</u>	
<u>C R</u>	
-	R(Q)/W(Q)

Strict
schedule



Note:

1. Strict recoverable schedule may or may not be serializable.
2. Serializable schedule may or may not be recoverable.



Concurrency control protocol

Concurrency control protocol should not allow to execute:

1. Non-serializable schedule (Violate Isolation).
2. Non-strict recoverable schedule (Violate atomicity and Durability).

Locking Protocol



Lock is a variable used to identify the status of data item.

Lock is a variable used to identify the status of data item.

Transaction (T_1)

Lock (A): Granted by concurrency controller

R(A)

W(A)

Lock(B): Denied by concurrency controller

Types of Lock

1. Shared Lock(s): Read only lock.
2. Exclusive lock (x): Read/write lock.

Shared
Exclusive

1. Shared (s mode)

- Exists when concurrent transaction granted READ access.
- Produce no conflict for Read only transactions.
- Issued when transaction wants to read and exclusive lock not held on item.

2. Exclusive (x) mode

- Exists when access reserved for locking transaction.
- Used when potential for conflict exists.
- Issued when transaction wants to update unlocked data.

Example:

Transaction (T_1)

S(A) : Granted shared lock

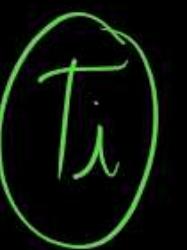
R(A) } only read allowed

X(B) : Granted exclusive lock

{ R(B) }
WB) } Read and write

Lock compatible table

P
W

Dataitem A		S	X 	Holded by T _i
Requested by T _j	{ X	Yes	No	
			No	No
	S	 X		
	X	YES No	No	No

Phase Locking Protocol (2PL)

- It guaranteed serializability.

① GROWING Phase [Acquire]

② Shrinking Phase [Release]

Growing Phase



Acquires all the required locks without unlocking any data. Once all locks have been acquired, the transaction is in it locked point.

Lock Point

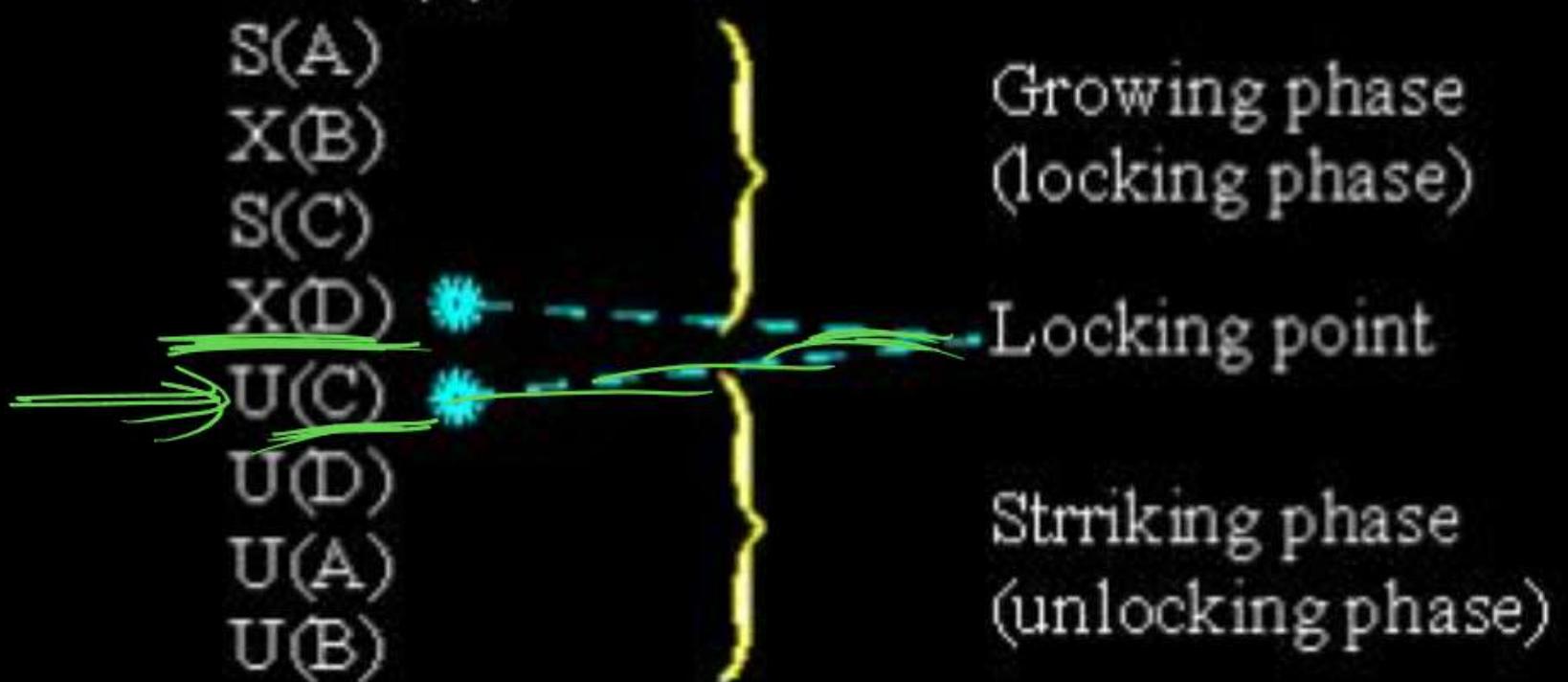
from where Shrinking Phase of the transaction starts.

Shrinking phase

Release all locks and can not obtain any new lock governing rules of 2 PL.

Example:

Transaction (T)



Important point 1

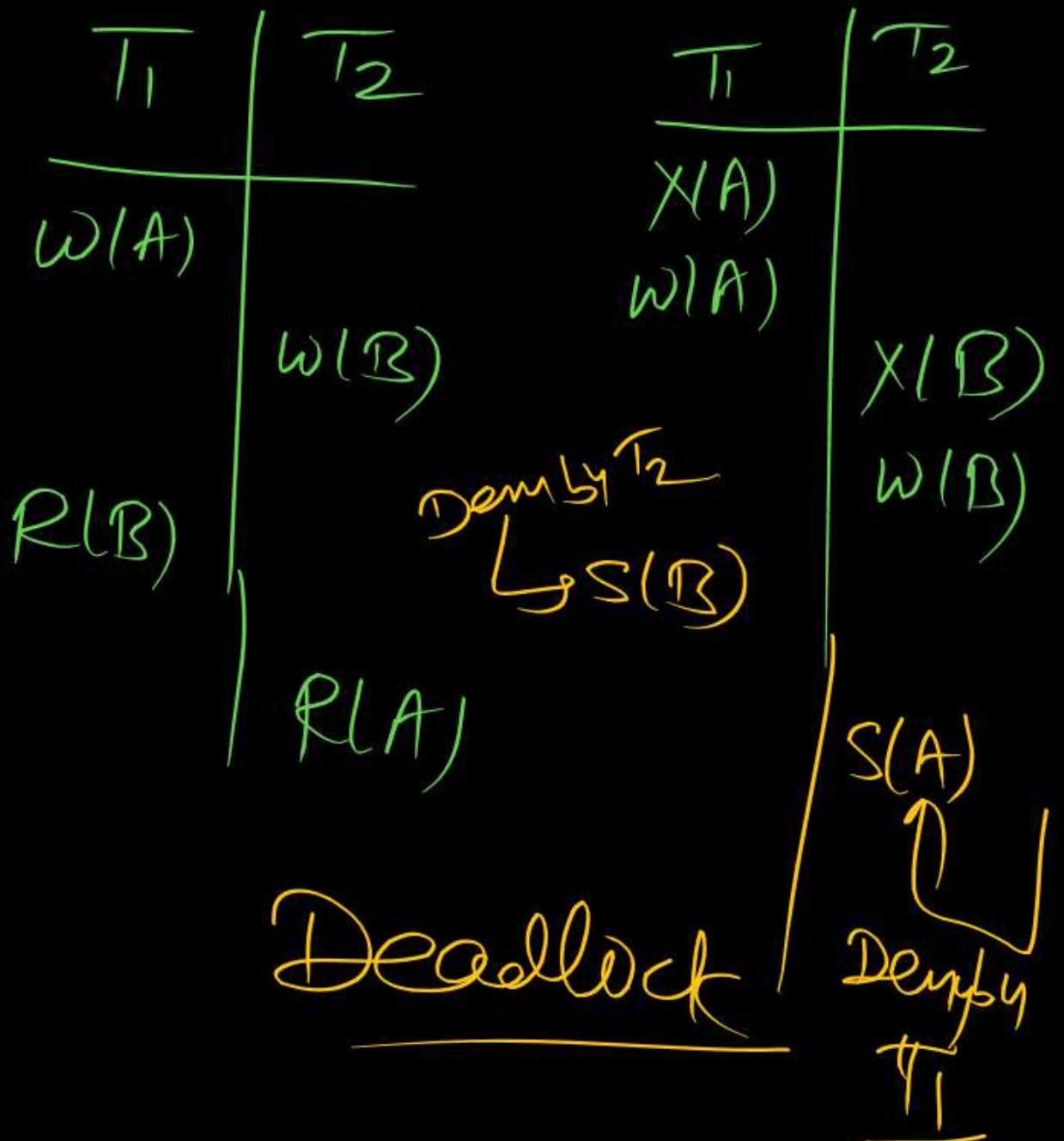
1. If schedule (s) is executed by 2PL then schedule guaranteed is conflict serializable schedule.
2. Conflict equal schedule is based on lock points order of the transaction.
3. If schedule is not conflict serializable schedule (cyclic precedence graph) then schedule not allowed to execute by 2 PL.

Note: Every schedule which is allowed by 2PL is always conflict serializable, but not every conflict serializable schedule is allowed by 2PL.



Limitations of 2PL

1. 2PL restriction may leads to deadlock.
2. 2PL restriction may leads to starvation.
3. 2PL condition not sufficient to avoid
 - (I) Ir-recoverable schedules
 - (II) Cascading rollback problem
 - (III) lost updated problem



Strict 2PL protocol

Basic 2PL: Lock request of transaction (T) not allowed in shrinking phase of transaction T

and

Strict recoverable: All exclusive lock of transaction must hold until commit/Rollback of transaction T.

2PL + All X lock Release After Commit

Strict 2PL protocol guarantees

- ✓ (a) Serializability.
- (b) Strict recoverable.

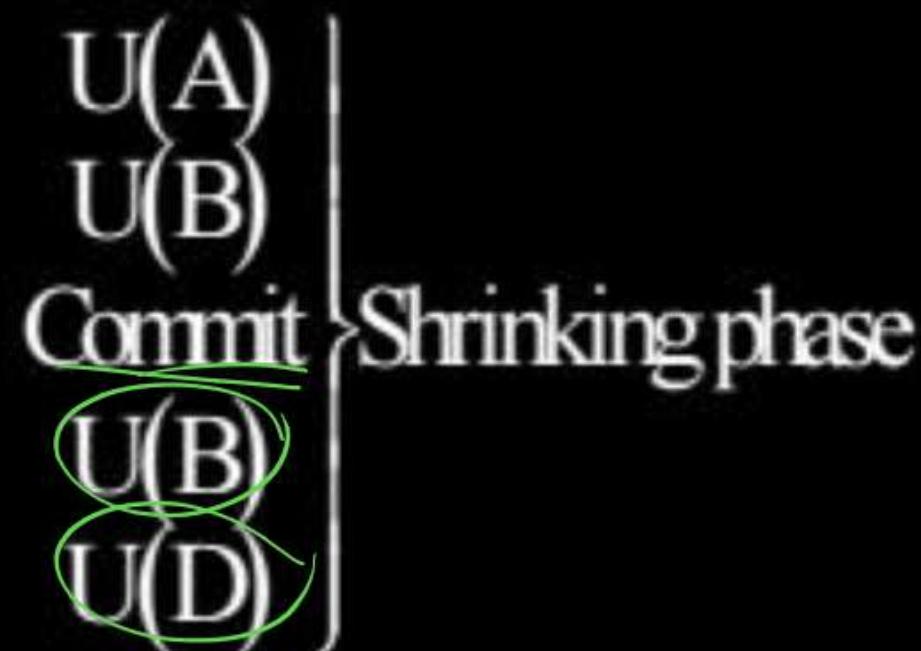
Recover ✓
Consistent ✓
Strict ✓

Disadvantage:

It is not free from deadlock and starvation.

Example:

Transaction (T)

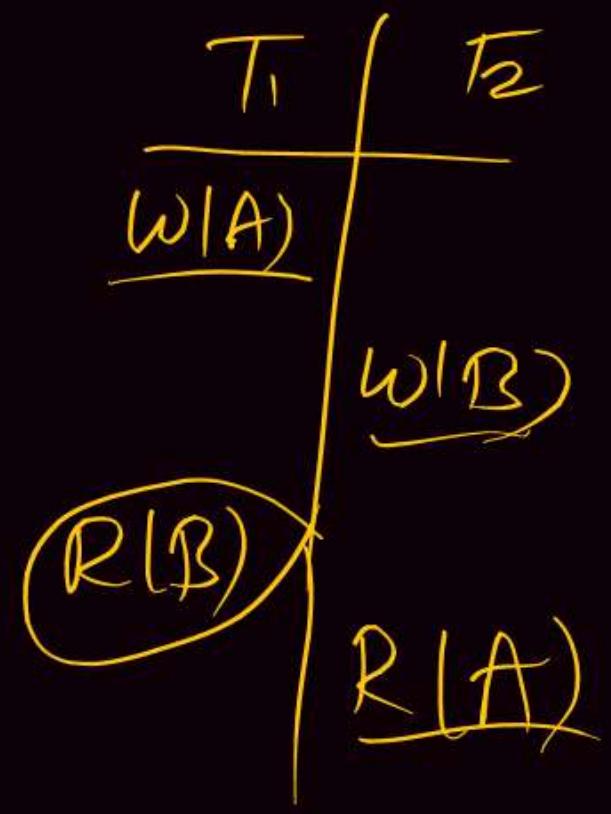


Rigorous 2PL



Basic 2 PL : A lock request allowed only in growing phase of transaction and
all locks (shared/Exclusive) of transaction T must be hold
until commit or Rollback.

2PL + All Lock Release After Commit / Rollback



2PL

↓

STRICT 2PL \longrightarrow X lock

Release After
Commit / Roll
back

Rigorous 2PL \rightarrow S&X lock Rel. After C/R.

Conservative 2PL

Conservative 2PL



Transaction (T) must lock all required data items in starting phase.

If locks are not available then unlock all data items and re-request all data item lock again.

	<u>Basic 2PL</u>	Strict 2PL	Rigorous 2PL	Conservative 2PL
Guaranteed Serializability	Yes ✓	Yes ✓	Yes ✓	Yes ✓
Guaranteed strict Recoverable	No ✗	Yes ✓	Yes ✓	No ✓ VB
Free from dead lock	No ✗	No ✗	No ✗	Yes ✓
Free from starvation	No ✗	No ✗	No ✗	No ✗

1	2	3	4	5
Lock-S(A)	Lock-S(A)	Lock-S(A)	Lock-S(A)	Lock-S(A)
R(A)	R(A)	R(A)	Lock-X(B)	R(A)
Lock-X(B)	Lock-X(B)	Lock-X(B)		Unlock(A)
R(B)	Unlock(A)	R(B)		R(B)
Unlock(A)	R(B)	R(B)		Lock-X(B)
W(B)	W(B)	W(B)		R(B)
Unlock(B)	Commit	Unlock(A)		W(B)
Commit	Unlock(B)	Unlock(B)		Unlock(B)

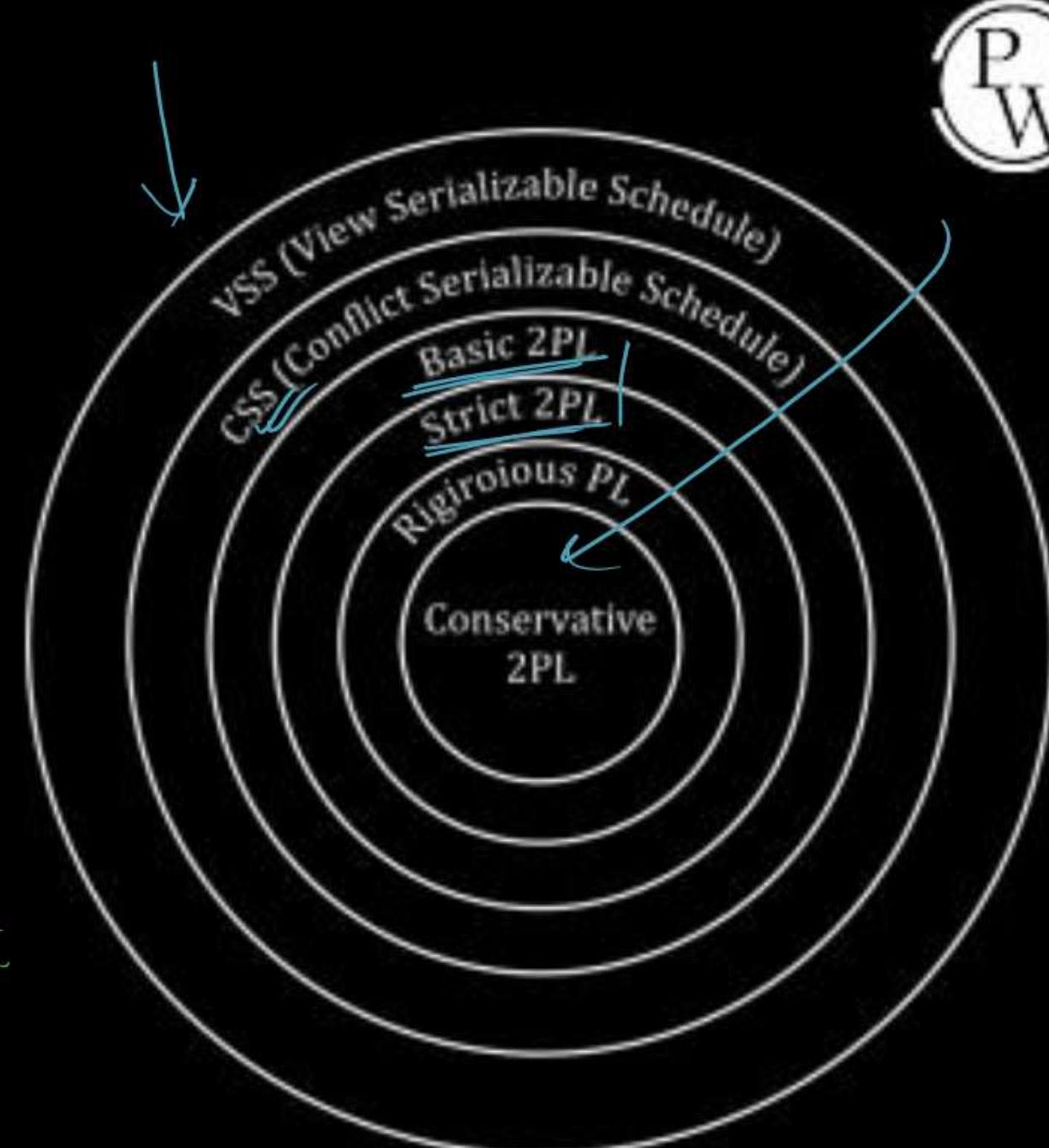
2PL

STRICT
2PL

RIGOROUS
2PL

Conservative
2PL

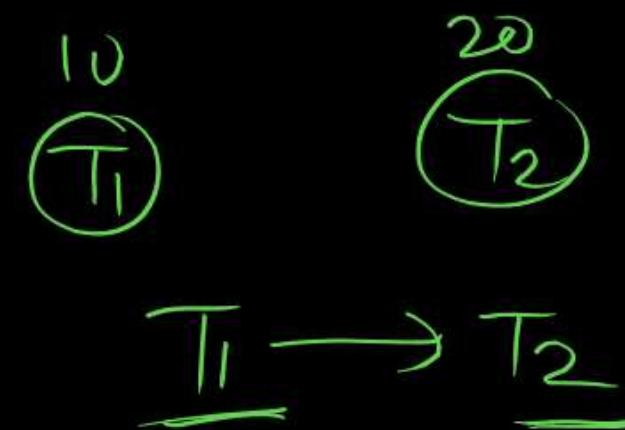
RIGOROUS
STRICT
Basic 2PL



Time stamp ordering protocol

Assign global unique time stamp value to each transaction and produces order for transaction submission.

The time stamp values of transaction can be used for transaction identification and to set priority between the transaction.



Time stamp value for each data item

Assume data item is A

- I. **Read TS value (A):** It is the highest transaction time stamp value that has executed R(A) successfully.
- II. **Write TS value (A):** It is the highest transaction TS value that has executed W(A) successfully.

1. Basic Time stamp ordering protocol:

- a. If transaction T issues R(A) operation.

If ($WTS(A)$) > $TS(T)$

then Rollback T

else

{Allow to execute R(A) successfully}

set $RTS(A) = \max \{RTS(A), TS(T)\}$

$$\underline{\text{TS}(\tau_i)} > \omega \text{TS}(Q)$$

$$\begin{matrix} R & \omega \\ \omega & R \\ \omega & \omega \end{matrix}$$

b. If transaction T issues W(A) operation

if ($RTS(A)$) > $TS(T)$

{then Rollback trans (T)}

else if ($WTS(A)$ > $TS(T)$)

{then Roll back T}

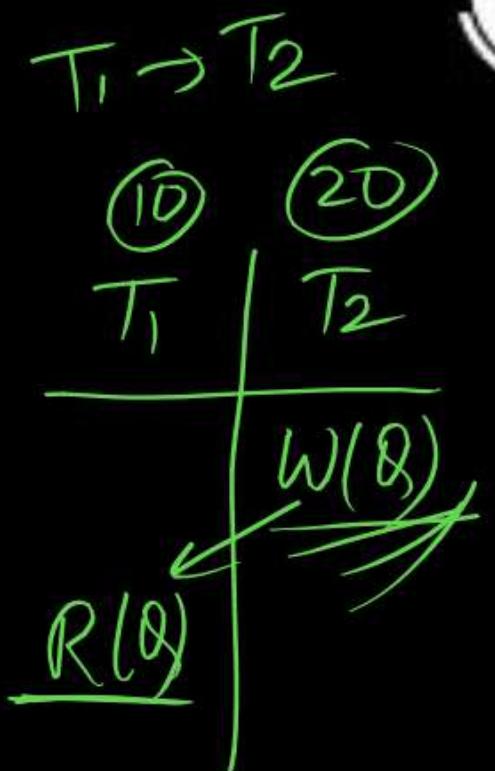
Else

{Allow to execute W(A) successfully}

set $WTS(A) = \{TS(T)\}$

I: T_i - Read(Q) (Transaction T_i Issue R(Q) Operation)

- (i) If $\text{TS}(T_i) < \text{WTS}(Q)$: Read operation Reject & T_i Rollback.
- (ii) If $\text{TS}(T_i) \geq \text{WTS}(Q)$: Read operation is allowed
and Set $\text{Read} - \text{TS}(Q) = \max[\text{RTS}(Q), \text{TS}(T_i)]$



II: T_i - Write(Q) (Transaction T_i Issue Write(Q) Operation)

- (i) If $\text{TS}(T_i) < \text{RTS}(Q)$: Write operation Reject & T_i Rollback.
- (ii) If $\text{TS}(T_i) < \text{WTS}(Q)$: Write operation Reject & T_i Rollback.
- (iii) Otherwise execute write (Q) operation
Set $\text{Read WTS}(Q) = \text{TS}(T_i)$

10
T₁

20
T₂

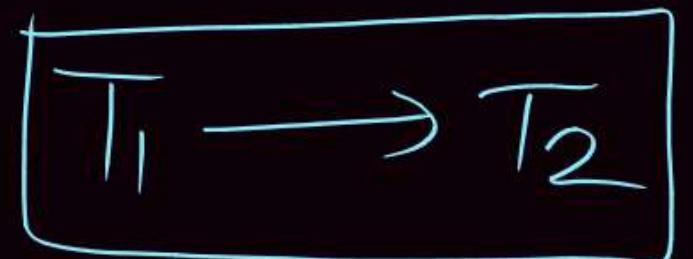
Transaction order

$$TS(T_1) = 10$$

$$TS(T_2) = 20$$

$$TS(T_1) < TS(T_2)$$

Order

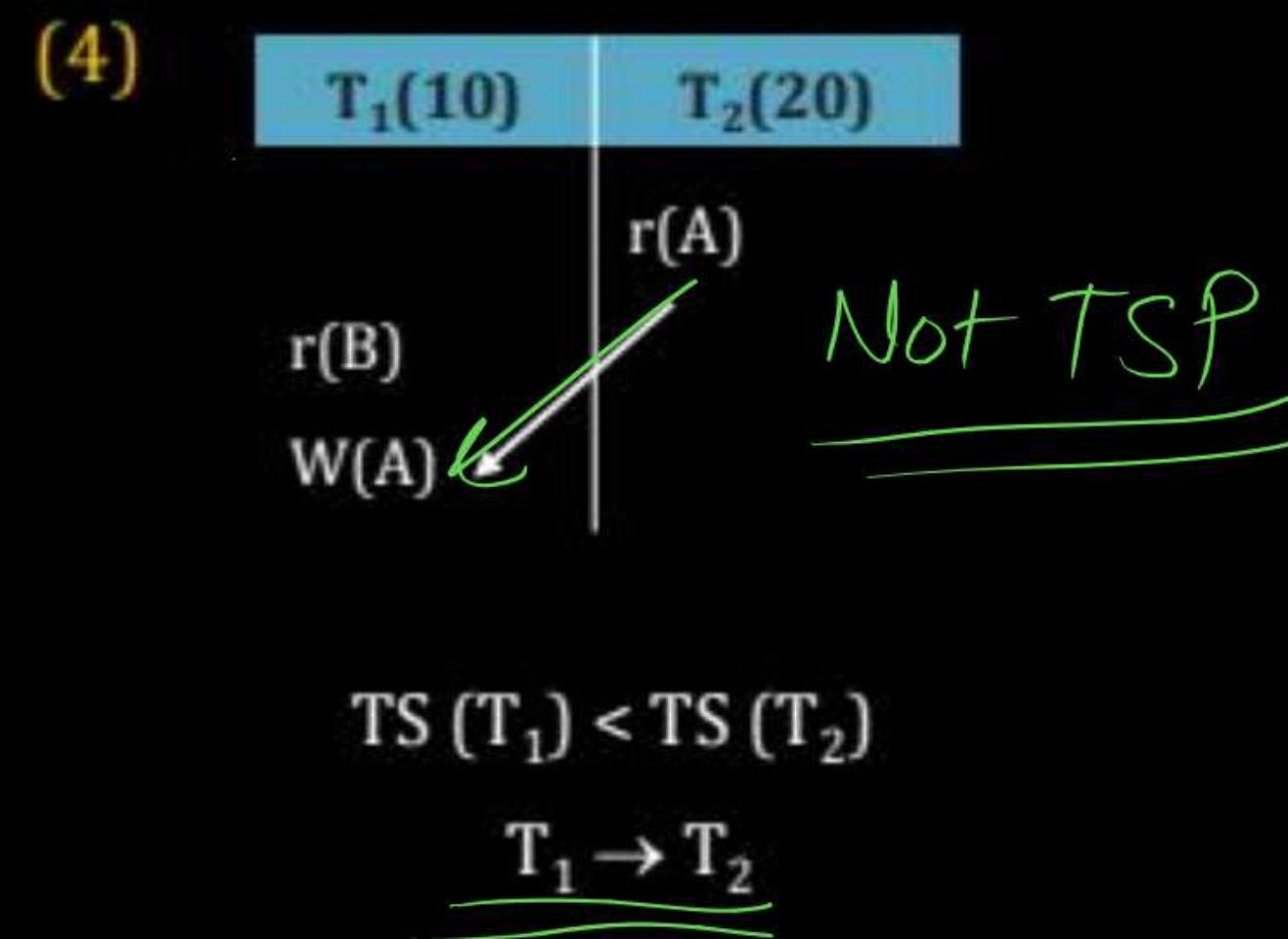
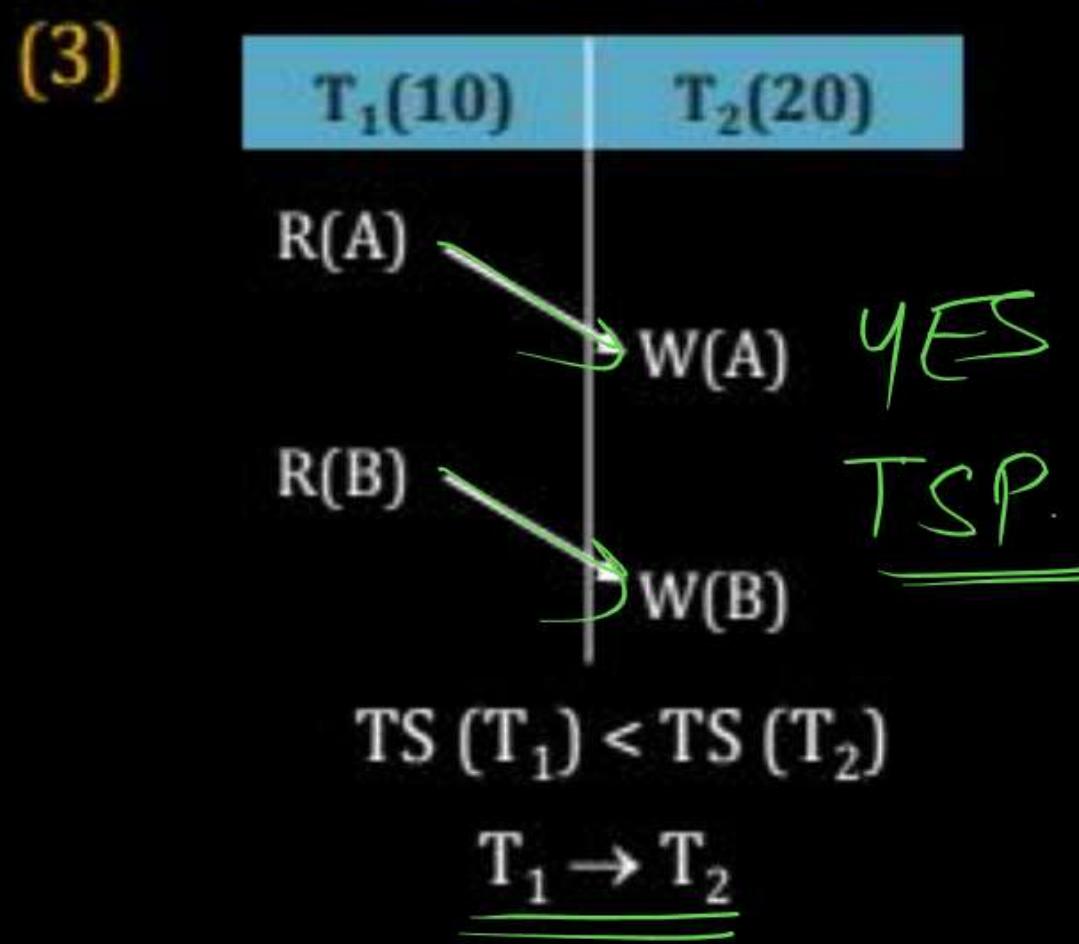
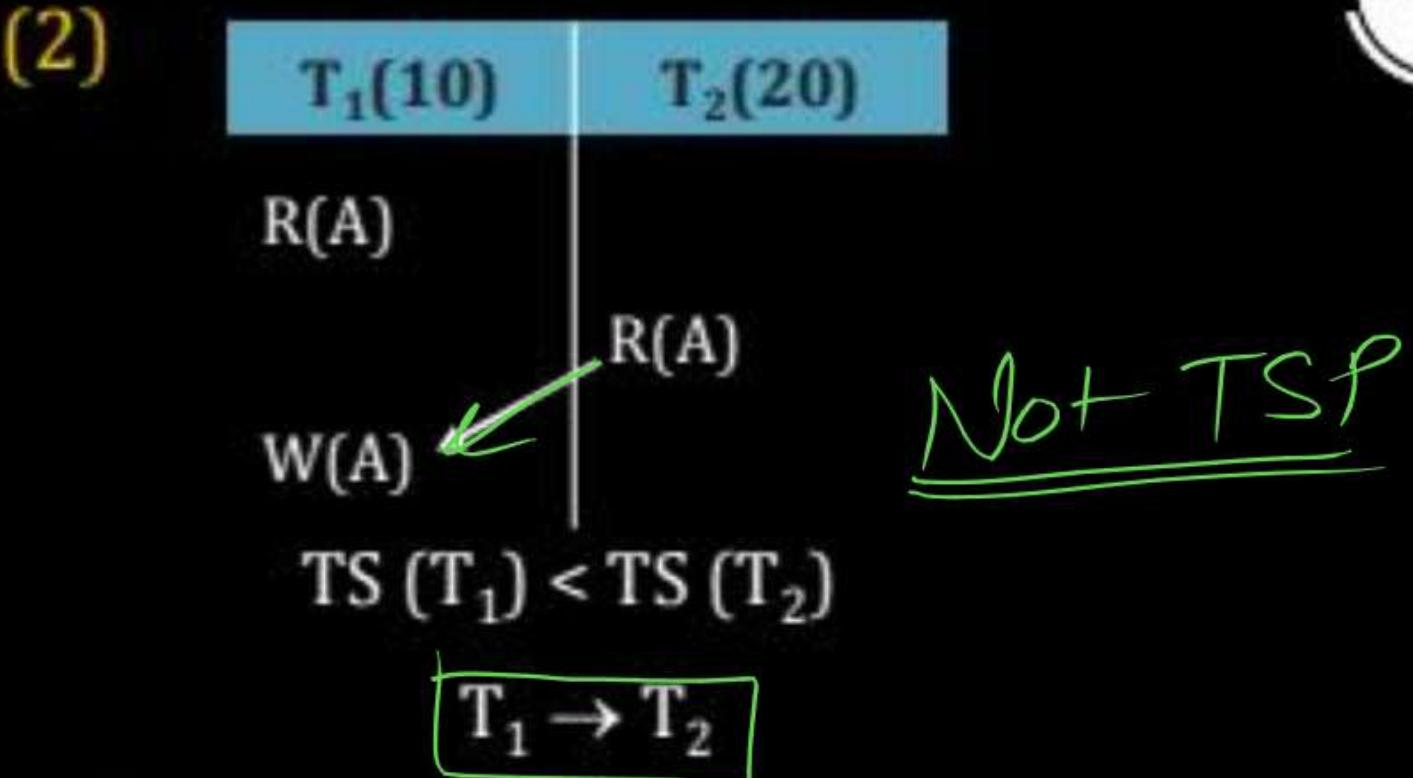
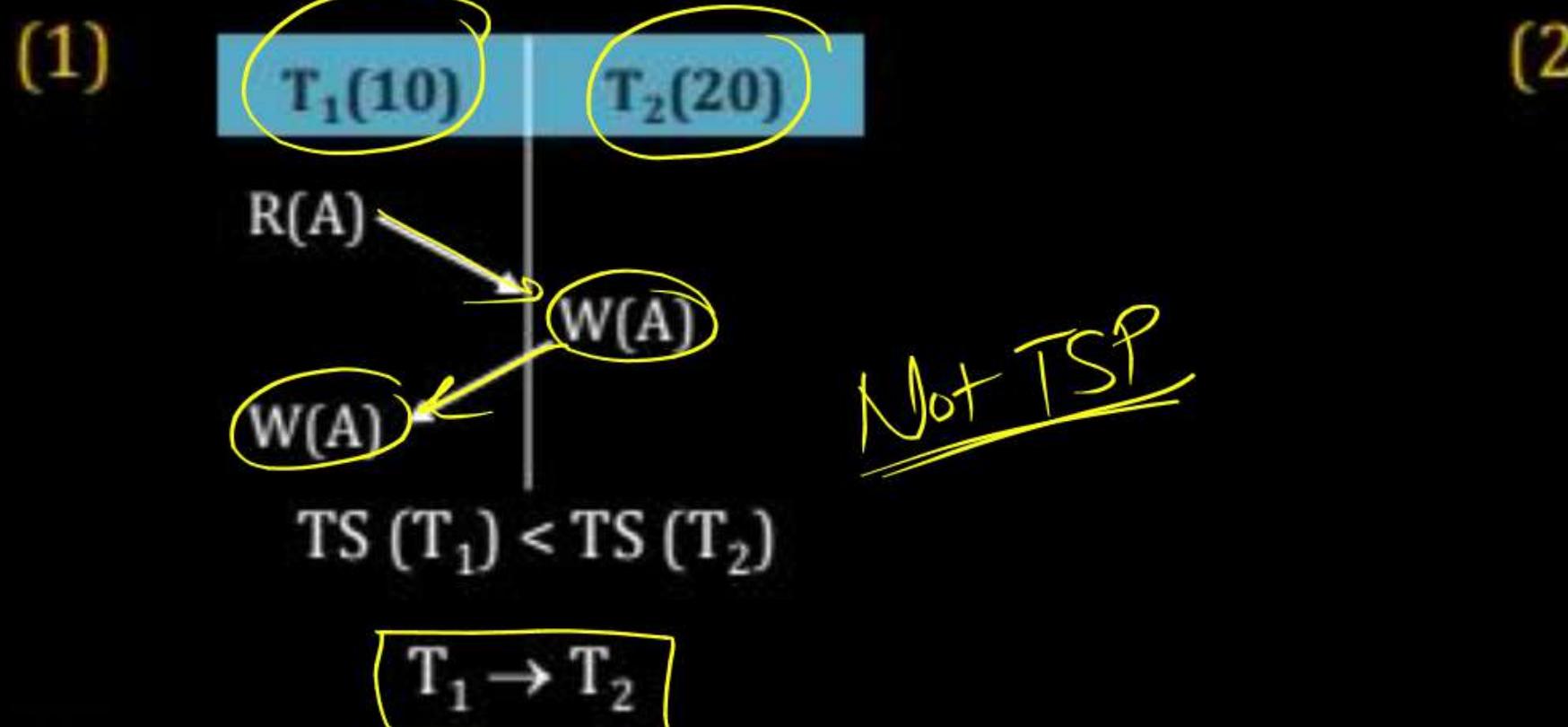


All Conflict operation
 $\overline{T_1} \rightarrow \overline{T_2}$

All Conflict Pair operation

Order Must be same
as Transaction Order

P
W



2. Thomas write rule stamp ordering protocol

If transaction T issues R(A) operation

if (WTS(A)) > TS(T)

{then Rollback transaction T}

else

{allow to execute R(A) successfully}

set RTS (A) = max {TS(T), RTS (A)}

(b) If transaction T issue W(A) operation
if ($RTS(A) > TS(T)$)
{then roll back T}
else if (WTS(A)) > TS(T)
{then ignore w(A) of transaction T and continue.}
else
{Allow to execute w(A) successfully
set $WTS(A) = TS(T)$ }

Thomas' Write Rule

→ View Serializability

- Modified version of the timestamp-ordering protocol in which obsolete write operations may be ignored under certain circumstances.
- When T_i attempts to write data item Q , if $TS(T_i) < W\text{-timestamp}(Q)$, then T_i is attempting to write an obsolete value of $\{Q\}$.
 - ❖ Rather than rolling back T_i as the timestamp ordering protocol would have done, this {write} operation can be ignored.
- Otherwise this protocol is the same as the timestamp ordering protocol.
- Thomas' Write Rule allows greater potential concurrency.
 - ❖ Allows some view-serializable schedules that are not conflict-serializable.

Thomas Write Rule (View Serializability)

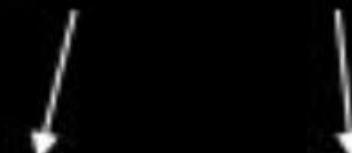
- TS(T_i) < RTS(Q) : Rollback
- TS(T_i) < WTS(Q) : Write operation is Ignored and No Roll back

Same as TSP

Time Stamp Protocol: Ensure serializability
deadlock free but starvation possible

Deadlock Prevention Algorithm

(1) Wait-Die

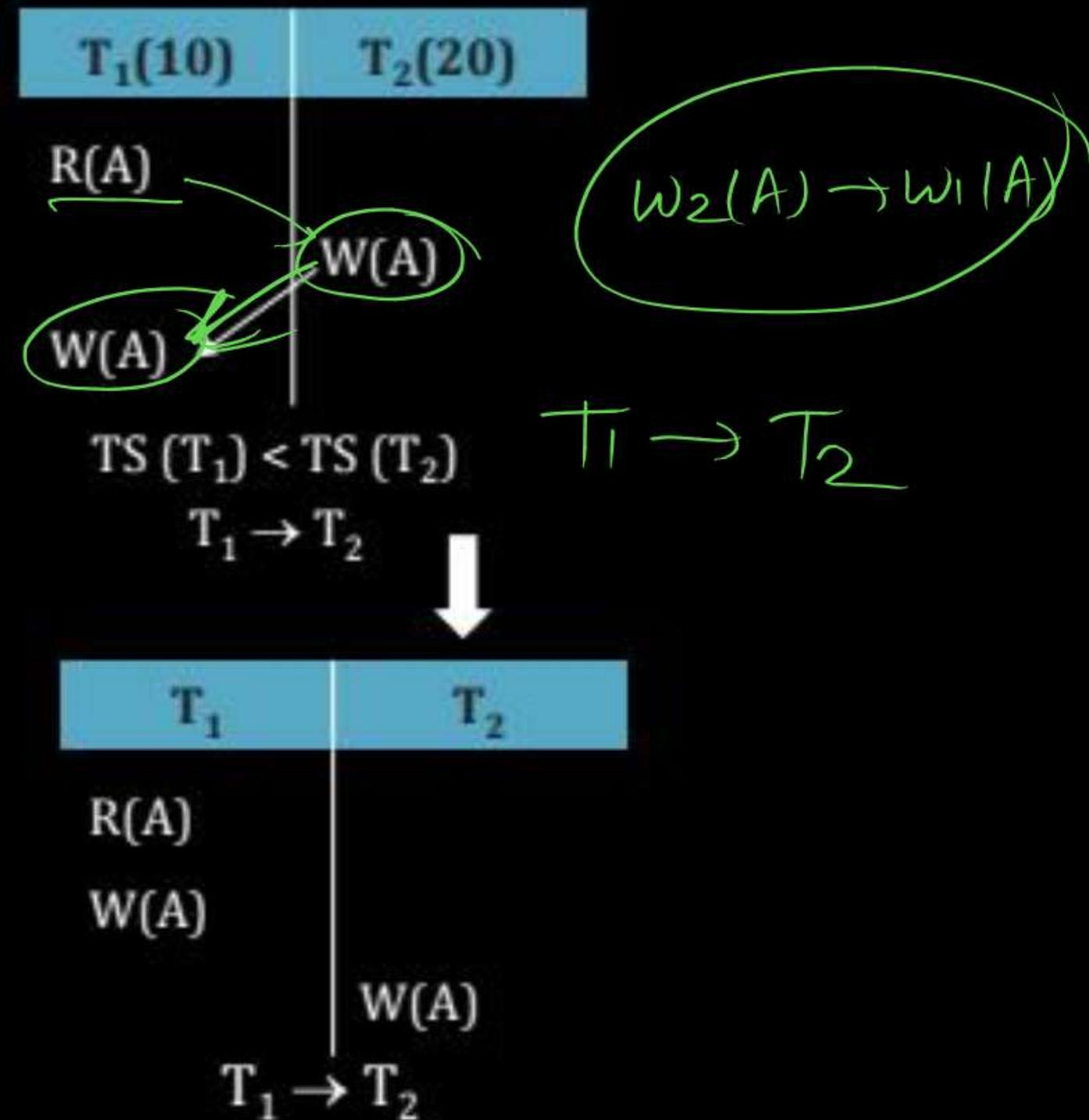


Older

(2) Wound-wait

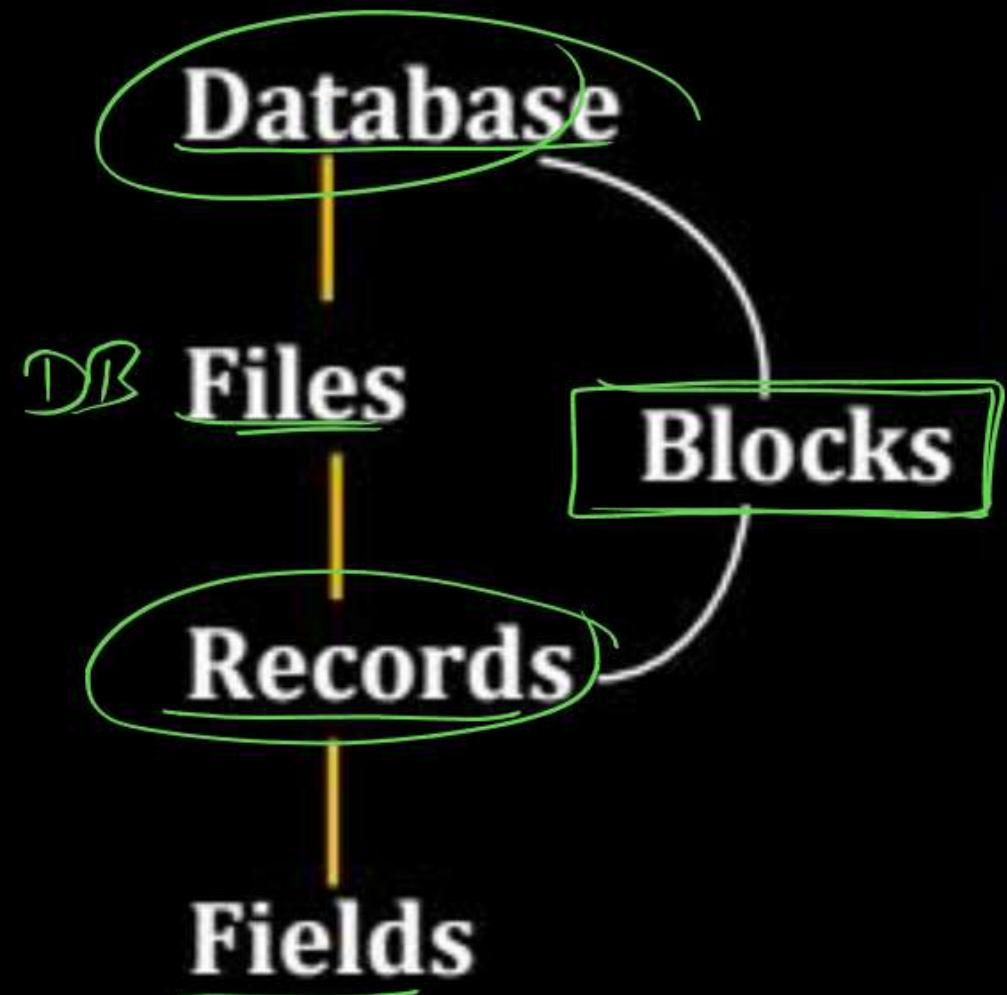


Younger



5. FILE ORGANIZATION AND INDEXING [B AND B+ TREE]

File Organization



- Database is collection of files.
- Each file is a collection of Records.
- Each record is a sequence of fields.

- DB is divided into number of blocks.
- Each block is divided into records.
- Record can be stored in a blocks.

Records of DB file



Fixed Length Records

{All records in the database file will be equal in size}

Example:

```
Create Table R
  ( A char (200)
    B char (200)
    C char (200) );
  )
```

Record of length 200 B



Variable Length Records

{Records of DB file can be different in size}

Example:

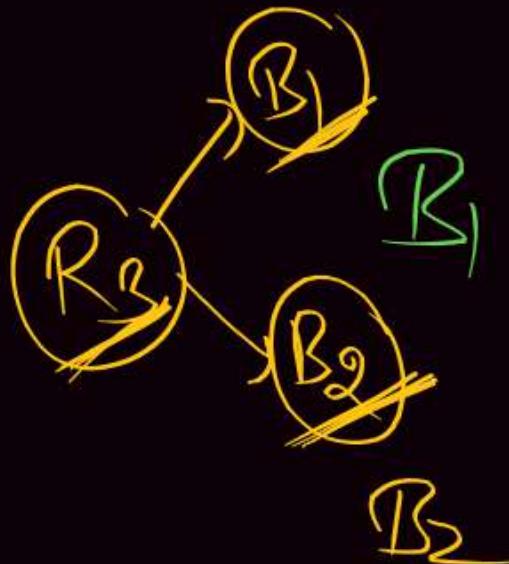
```
Create Table S
  ( D char (100)
    E char (500)
    F text (20)
  );
  )
```

Block Size = 100B

& Record Size = 40B

SPANNED

$$B_F = \frac{100B}{40B} = 2.5$$



<u>R1 (40B)</u>
<u>R2 (40B)</u>
<u>R3 (20B)</u>
<u>R3 (1/2 R3) 20</u>
<u>R4 (40B)</u>
<u>R5 (40B)</u>

Unspanned

$$B_F = \left\lfloor \frac{100}{40} \right\rfloor = \lfloor 2.5 \rfloor = 2$$

2 Records per Block

<u>40B (R1)</u>
<u>40B (R2)</u>
20 Free
R3
R4
20 Free

B1

B2

Records organization of DB file



Spanned Organization	Unspanned Organization
1. Record allowed to <u>span in more than one block</u> .	1. <u>Complete record must be in one block</u> .
2. Advantage: Possible to allocate file without any <u>internal fragmentation</u> .	2. Advantage: <u>less access cost and easy to organize DB file</u> .
3. Disadvantage: <u>More access cost to access spanned records</u> .	3. Disadvantage: May not possible to allocate DB file without any <u>internal fragmentation</u> .

NOTE



1. Spanned organization prefer to store database record of the file with variable length record.
2. Unspanned organization prefer to store DB record of file with fixed length record.

Default: Unspanned

$$\text{Block factor} = \frac{\text{Block Size}}{\text{Record Size}}$$

|

→ Avg # Records per Block.

Block factor [BF]

The maximum possible records which can be stored in block (maximum record per block).

Consider record size = R bytes

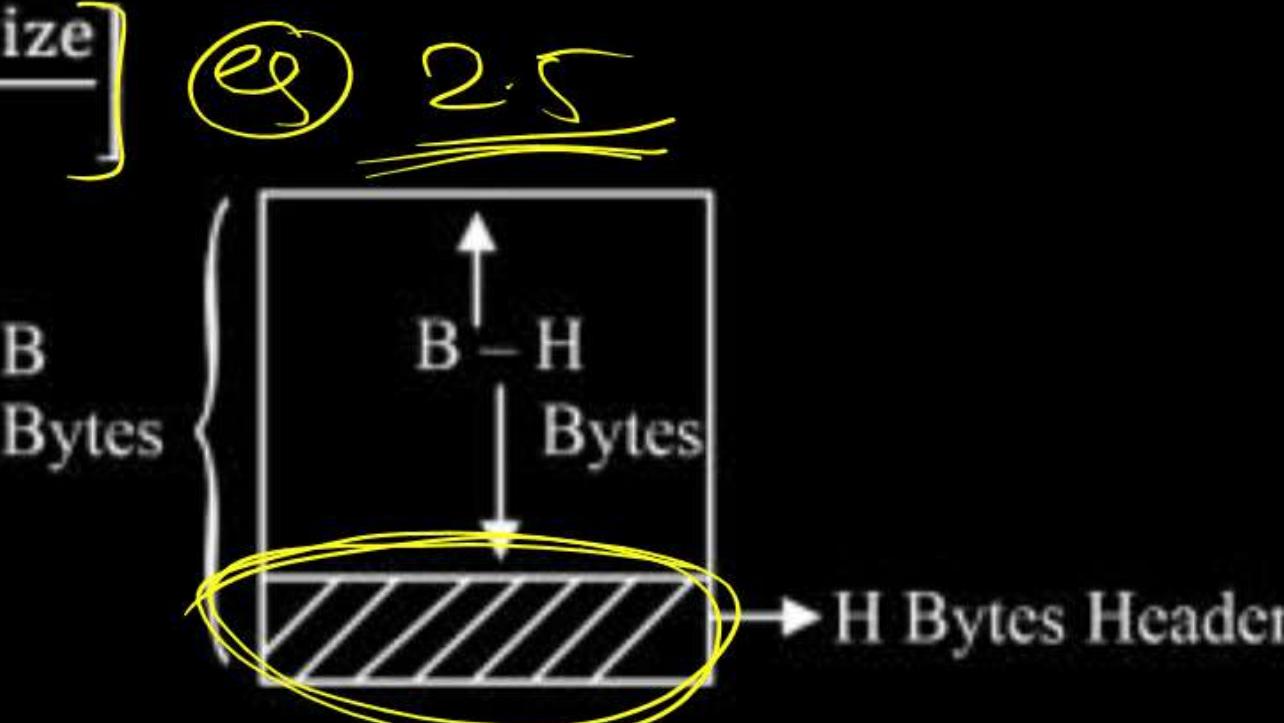
$$1. \therefore \text{Block Factor} = \left\lfloor \frac{\text{Block size} - \text{Header size}}{\text{Record size}} \right\rfloor$$

For unspanned organization

$$2. \therefore \text{Block Factor} = \left\lfloor \frac{\text{Block size} - \text{Header size}}{\text{Record size}} \right\rfloor$$

For spanned organization

Unspanned
 @ $\lfloor 2.5 \rfloor = 2$



Categories of Index

Dense Index

For each DB record of DB file there must be index entry in index file.

Sparse Index

For set of DB records (blocks) of DB file there exists index entry in index file.

Important Point I

- Sparse index possible only over ordered field of DB file.
- In Sparse indexing

$$\left[\begin{matrix} \text{No. of index} \\ \text{file entries} \end{matrix} \right] < \left[\begin{matrix} \text{No. of records} \\ \text{of DB file} \end{matrix} \right]$$

Category of Index

1) Dense Index Files

Number of Index entries = Number of DB Records

2) Sparse Index Files

Number of Index entries = Number of Blocks

~~Zebra
2049~~

Assume $lkey = 8$ $B_p = 2B$

#Record = 50,002

$BS = 1024B$

P
W

Record
size = 100B

(50,002)

$$B_F = \left\lceil \frac{1024}{100} \right\rceil = 10 \text{ Record per Block}$$

Total #Records = 50002

$$\#DB Block = \left\lceil \frac{50002}{10} \right\rceil$$

= 5002 DB Block

Dense Index Files

- Dense Index - Index record appears for every search-key values in the file.
- Example - index on *ID* attribute of *instructor* relation

10101	10101	Srinivasan	Comp. Sci.	65000	
12121	12121	Wu	Finance	90000	
15151	15151	Mozart	Music	40000	
22222	22222	Einstein	Physics	95000	
32343	32343	El Said	History	60000	
33456	33456	Gold	Physics	87000	
45565	45565	Katz	Comp. Sci.	75000	
58583	58583	Califieri	History	62000	
76543	76543	Singh	Finance	80000	
76766	76766	Crick	Biology	72000	
83821	83821	Brandt	Comp. Sci.	92000	
98345	98345	Kim	Elec. Eng.	80000	

10101		10101	Srinivasan	Comp. Sci.	65000	
32343		12121	Wu	Finance	90000	
76766		15151	Mozart	Music	40000	
		22222	Einstein	Physics	95000	
		32343	El Said	History	60000	
		33456	Gold	Physics	87000	
		45565	Katz	Comp. Sci.	75000	
		58583	Califieri	History	62000	
		76543	Singh	Finance	80000	
		76766	Crick	Biology	72000	
		83821	Brandt	Comp. Sci.	92000	
		98345	Kim	Elec. Eng.	80000	

Block Size = 100B

Record Size = 40B

$B_p = 2.5 \rightarrow 2$
DB Record

With Index

One Index Record Size = $\frac{\text{Size of key}}{\text{Size of Block pointer}}$

Assume key = 8B $B_p = 2\text{Byte}$

Block Size = 100B

One Index Records Size = $8 + 2 = 10\text{Byte}$

Block factor of Index File = $\frac{100}{10} = 10$ Index Record Per Block

Categories of Index

Block Factor of Index

Assume Block size = B Bytes

Header size = H Bytes

Search key = K Bytes

Pointer size = P Bytes

∴ block Factor of index = $\left\lfloor \frac{B-H}{K+P} \right\rfloor$

$$\frac{\text{Block size}}{\text{key} + \text{P size}}$$

NOTE: By default header size will be 0 bytes, if not given in question.

I/O Cost [Access cost]

The number of secondary memory block (DB file block) required to transfer from disk to main memory in order to access required record.

1. I/O cost to access data record without index from DB file of n blocks

- (a) Over ordered field:

$$\text{Access cost} = \lceil \log_2 n \rceil \text{block}$$

- (b) Over unordered field:

$$\text{Access cost} = n \text{ blocks}$$

$$\text{Avg} = \frac{n}{2}$$

I/O Cost [Access cost]

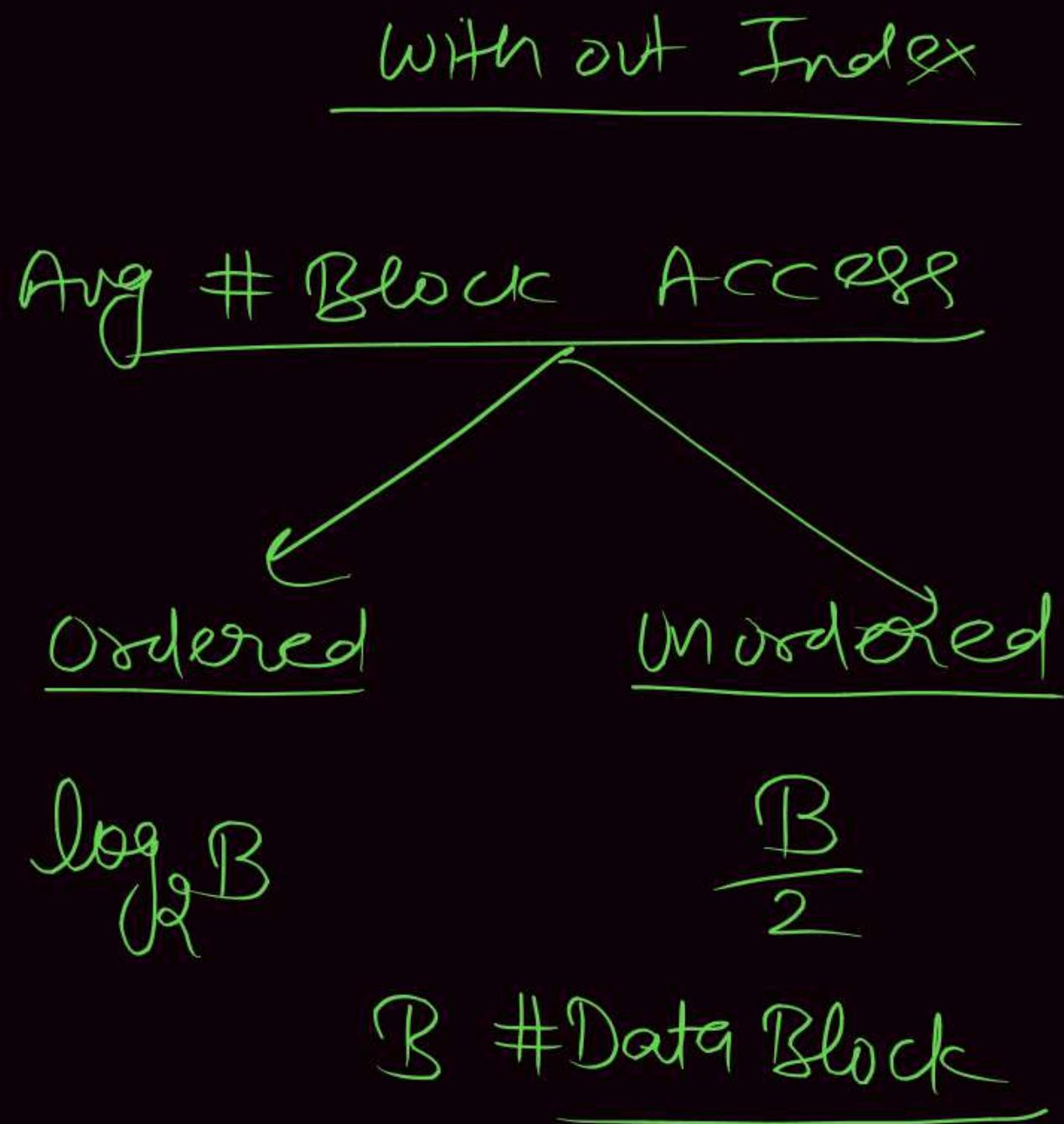
P
W

2. If Dense Index Used:

- Number of Dense Index block = $\left\lceil \frac{\text{number of records}}{\text{Block Factor of index}} \right\rceil$ $\left\lceil \frac{50002}{10} \right\rceil = 5001$ Index Block
- Access cost (Number of Block access)
 $= [\log_2(\text{number of Dense index blocks at 1st level})] + 1$

3. If sparse Index Used:

- Number of DB blocks = $\left\lceil \frac{\text{Number of records}}{\text{BF of DB}} \right\rceil$ $\left\lceil \frac{50002}{10} \right\rceil = 5001$ DB Block
- Number of sparse Index block (At 1st level) = $\left\lceil \frac{\text{number of DB blocks}}{\text{Block Factor of index}} \right\rceil$ $\left\lceil \frac{5001}{10} \right\rceil = 501$ SPAREDB
- Access cost (Number of block access)
 $= [\log_2(\text{number of sparse index blocks at 1st level})] + 1$ Index Block



With Index

\downarrow

Ordered

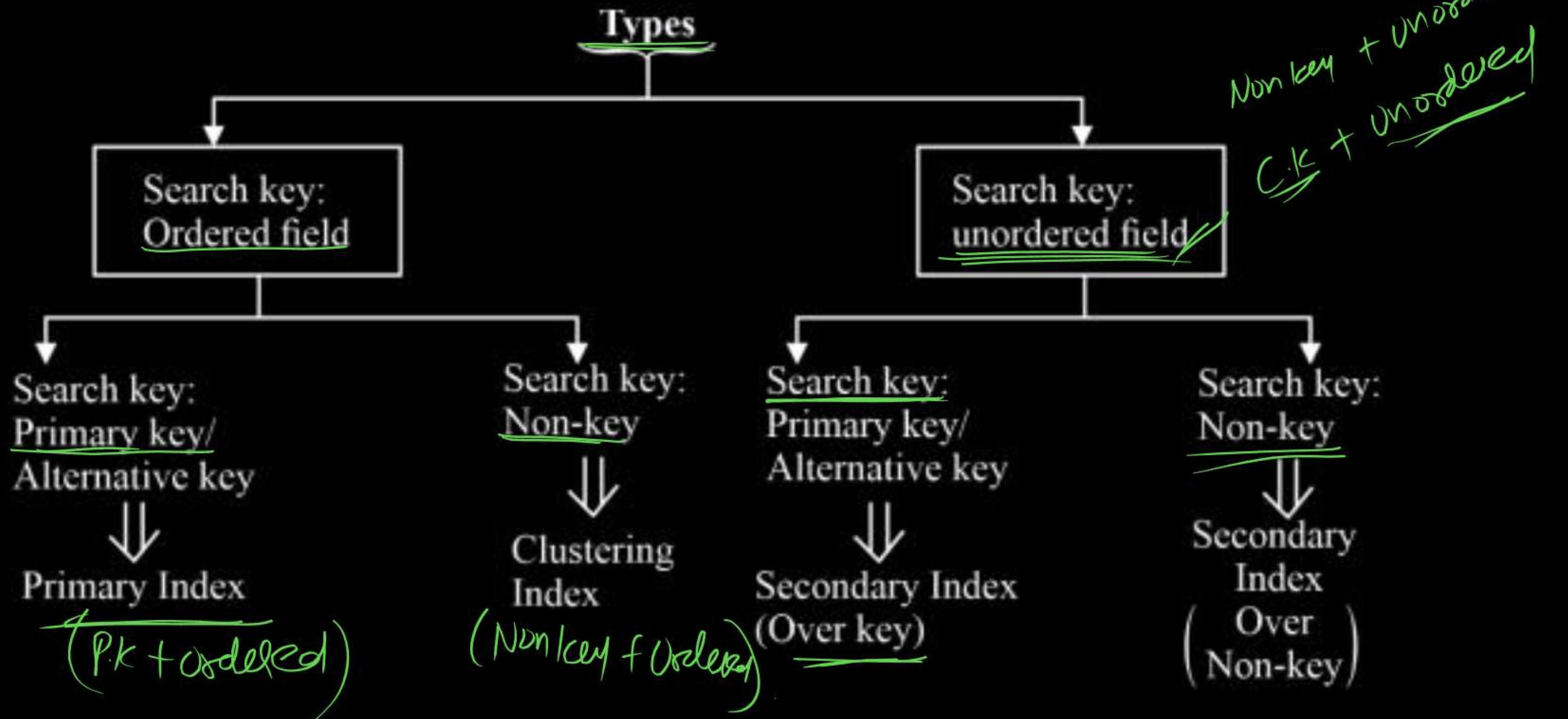
Avg # Block Access = $\log_2 B_i + 1$

$$\text{Avg # Block Access} = \log_2 B_i + 1$$

$B_i : \# \underline{\text{Index Block}}$

Types of Index

- Search key: Fields of DB file which is used for indexing



Primary Index



Search Key: Ordered field and key (candidate key)

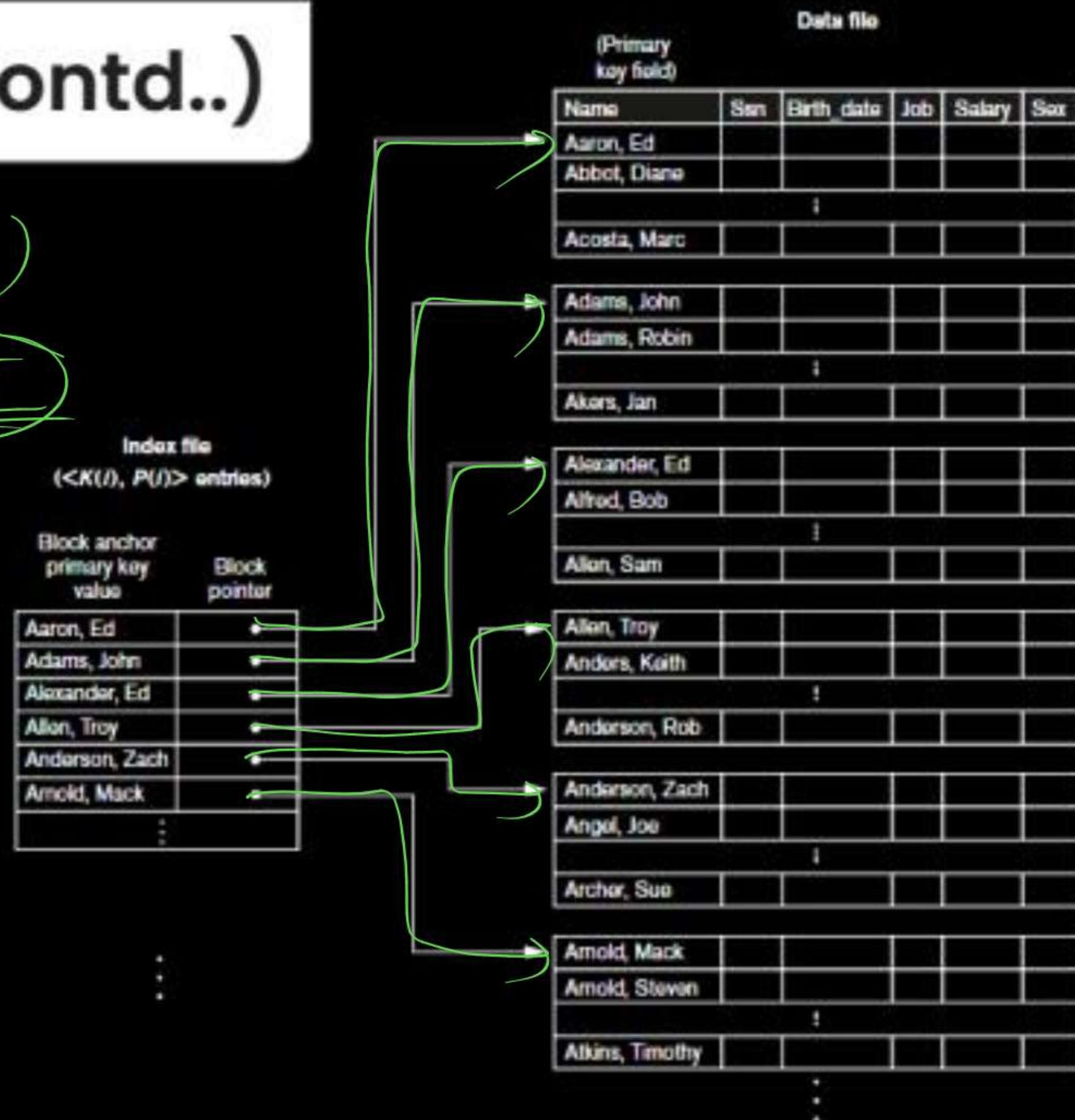
- ❑ I/O cost to access record using primary index with multilevel indexing is $(k + 1)$ blocks.
- ❑ Where k is level of indexing.
- ❑ For any database relation at most one primary index is possible because, search key must be ordered field.
- ❑ Primary index can be either dense or sparse but sparse primary index preferred.

Primary Index (Contd..)

(PK + ordered file)

Dense & SPARSE

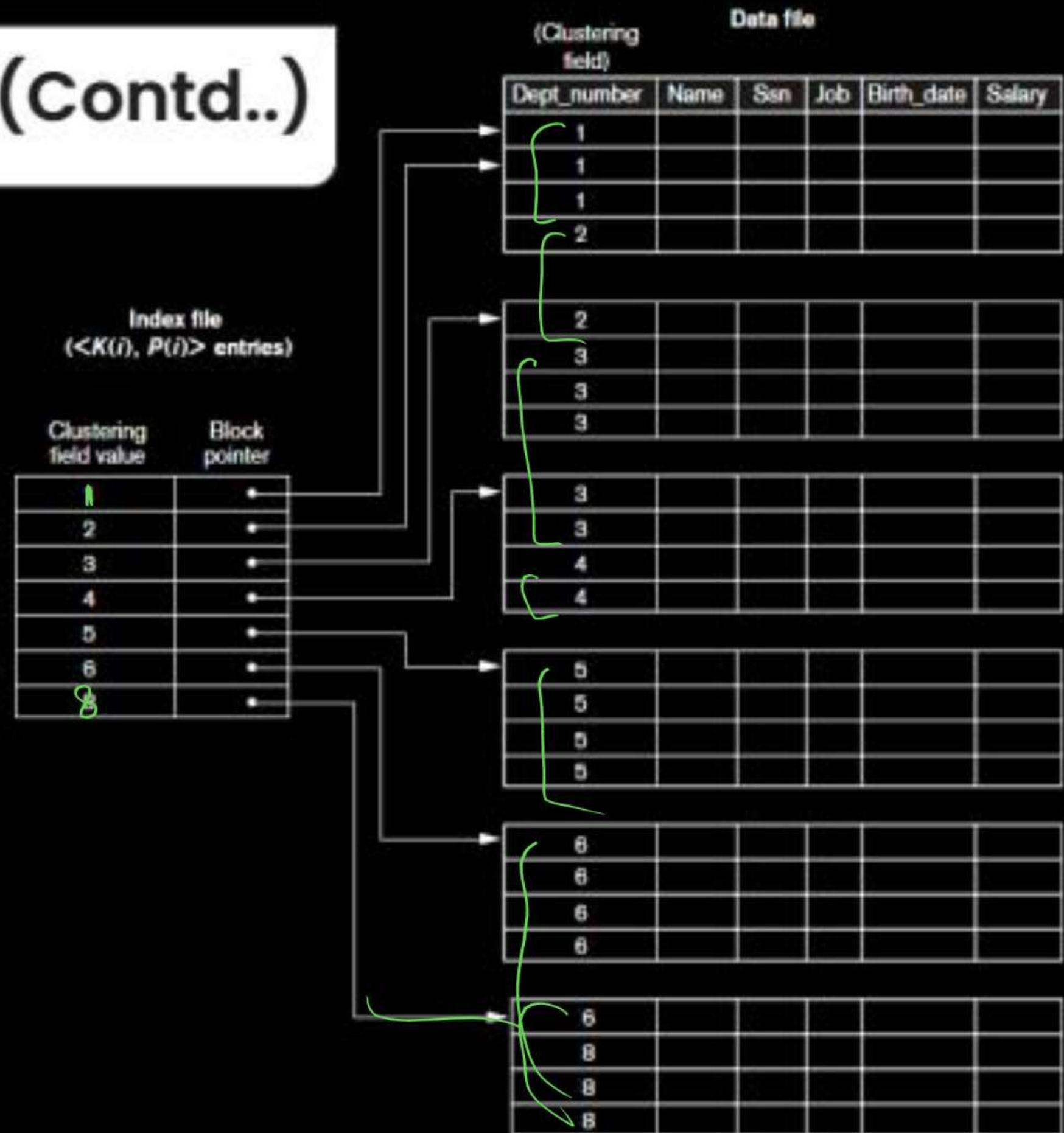
Primary index on
the ordering key
field of the file
shown in Figure



Clustering Indexes

- Clustering field (Non key + Ordered file)
 - ❖ File records are physically ordered on a nonkey field without a distinct value for each record
- Ordered file with two fields
 - ❖ Same type as clustering field
 - ❖ Disk block pointer

Clustering Indexes (Contd..)



NOTE:

For any DB relation either primary index or clustering index is possible but not both simultaneously.

Q.

Suppose that we have Ordered file of 30,000 records, stored on a disk with Block Size 1024 Byte, file records are of fixed length & unspanned of size 100 Byte (Record size) and suppose that we Have created a primary index on the key field of the file of size 9 Byte and Block pointer of size 6 Byte then find the average number of Block Access to search for a record using with and Without Index ?

#Records = 30,000 BS = 1024B Record Size = 100B Key = 9, BP = 6B

Unspanned, Ordered & Primary Index

(SOL)

Without Index

Block factor of DB file = $\left\lceil \frac{\text{Block size}}{\text{Record size}} \right\rceil \rightarrow \left\lceil \frac{1024}{100} \right\rceil = 10 \text{ Record per Block}$
[Unspanned]

Total # Records = 30,000

DB Blocks [B] = $\left\lceil \frac{30000}{10} \right\rceil = 3000 \text{ Data Block}$

To Access a Record Avg # Block Access = $\log_2 B \Rightarrow \log_2 3000$
(Ordered) = 12 Block Access

#Records = 30,000 BS = 1024B Record Size = 1001, key = 9, BP = 6B

Unspanned, Ordered & Primary Index

(Soln)

With Index

One Index Record Size = 9 + 6 = 15 Byte

Block factor of Index File = $\left\lfloor \frac{1024B}{15B} \right\rfloor = 68$ Index entry per Block

SPARSE Total #Index Entries = 3000 (# of DB Block)

Total # Index Block [B_i] = $\lceil \frac{3000}{68} \rceil = 45$ Index Block.

Avg # Block Access with Index = $\log_2 B_i + 1 \rightarrow \log_2 45 + 1 \rightarrow 6 + 1 = 7$ Block Access

Secondary Index [over key]



Search Key: Unordered field and key.

- ❑ Secondary index is alternative index to access data records even primary or clustering index already exists.
- ❑ Secondary index over key is always dense index.
- ❑ I/O cost to access record using secondary index over key with multilevel index: (k + 1) blocks.

Secondary Index [over non-key]

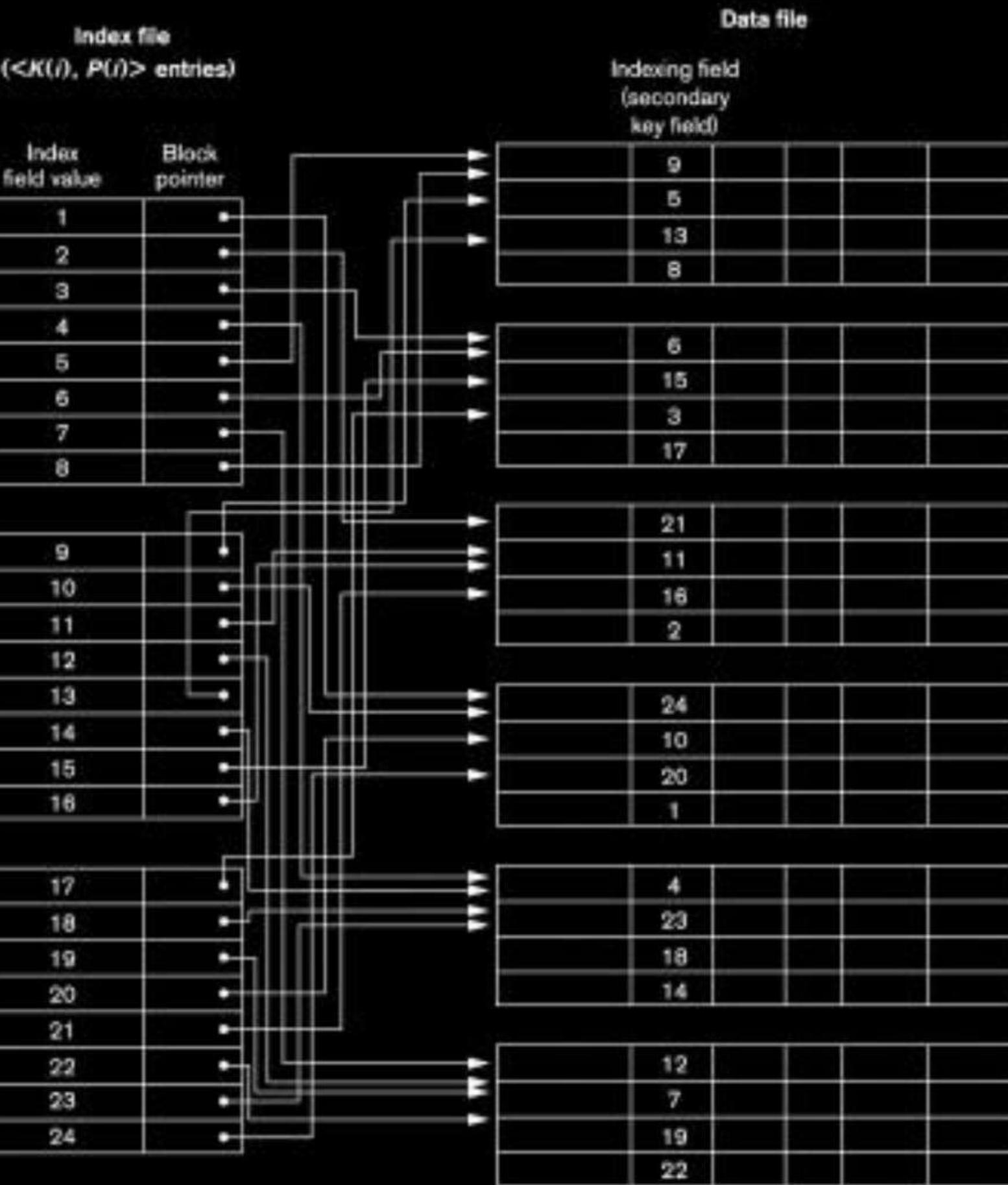
Search Key: Unordered field and non-key.

- ❑ I/O cost to access record using secondary index over non-key with multilevel index = [k blocks from k level of index] + [some more DB blocks]

NOTE:

Many secondary index's possible for DB file.

Secondary Indexes (Contd..)



Q.

Consider a secondary Index on the key field of the file of question number 1, then find the Average number of Block Access to Access a record using with & without Index?

Number of records = 30000 Block size = 1024 B

record size = 1000 B

Key = 9B Bp = 6 Byte

Unspanned & Unordered.

$$\text{Without Index} = \frac{B}{2} = \frac{30000}{2} = 1500 \text{ Block Access}$$

With Index #Index entries = 30,000 (Dense) #Records

$$\log_2 442 + 1 = 10 \text{ Block Access}$$

Total #Index Block = $\left\lceil \frac{30000}{68} \right\rceil = 442 \text{ Index Block}$

Multi Level Indexing

2nd level

Index Entries = 442 (1st level Index Block)

Index Block = $\lceil \frac{442}{68} \rceil = 7$ Index Block

3rd level

Index Entries = 7 (2nd level Index Block)

Index Block = $\lceil \frac{7}{68} \rceil = 1$ Index Block

Total 3 Level

Multi Level \Rightarrow n Level

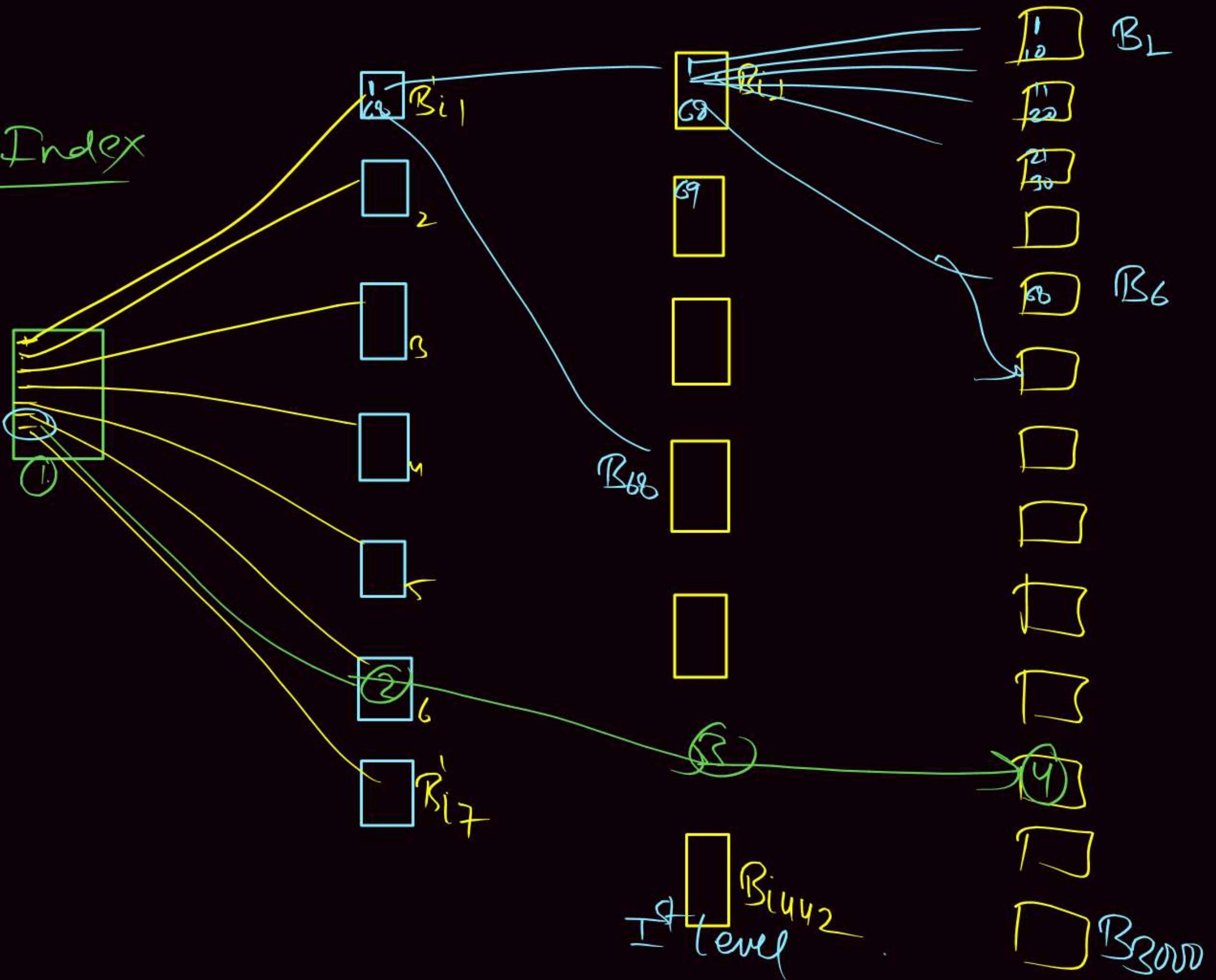
To Access a Record Avg # Block Access = $n + 1$
 $= 3 + 1$

$$\text{3rd Level} \leftarrow | + | + | + | = ④ \text{ Avg} = ④$$

↓ ↓ ↓ ↓
2nd 1st Data Block

Multilevel Index

B_{i1}'



Q.

A clustering index is defined on the fields which are of type

P
W

[GATE-2008 : 1 Mark]

- A Non-key and ordering
- B Non-key and non-ordering
- C key and ordering
- D key and non-ordering

Q.

(2^{14}) Consider a file of 16384 records. Each record is 32 bytes long and its key field is of size 6 bytes. The file is ordered on a non-key field, and the file organization is unspanned. The file is stored in a file system with block size 1024 bytes, and the size of block pointer is 10bytes. If the secondary index is built on the key field of the file, and a multilevel index scheme is used to store the secondary index, the number of first-level and second-level block in the multilevel index are respectively [GATE-2008 : 2 Marks]

- A 8 and 0
- B 128 and 6
- C 256 and 4
- D 512 and 5

Ans [C]

$$\text{One Index Record Size} = \text{Size of key} + \text{Rp Size}$$
$$6B + 10 = 16B \Rightarrow 2^4 \text{ Byte}$$

$$\text{Block Size} = 1024B (2^{10} \text{ Byte})$$

$$\text{Block factor of Index file} = \frac{2^{10}}{2^4} = 2^6 \Rightarrow 64 \text{ Index Entries per Block}$$

Secondary : Dense : #Index entries = 2^{14} (#Records)

$$\text{1st Level} \quad \# \text{Index Block} = \frac{2^{14}}{2^6} = 2^8 \Rightarrow 256 \text{ Index Block}$$

$$\text{2nd Level} : \# \text{Index entries} = 256$$

$$\text{Total} \quad \# \text{Index Block} = \frac{256}{2^6} = \textcircled{4} \text{ Index Block}$$

Balanced Search Tree [Height Restricted Search Tree]

- The maximum height of search tree for n distinct keys should not exceed O ($\log n$)
- Worst case search cost to search element is : $O (\log n)$

Why B/B⁺ Tree Index preferred rather than balanced BST/AVL for DB index

(a) **AVL:** If AVL tree (balanced BST) used for DB index:



One block of disk

Block [AVL Tree node]

- Disadvantage: Each index block with only one key so the number of levels of index is high. (I/O cost to access data is also very high).
- Wastage of disc space, which is allocated for index (only one key stored per block)

Why B/B⁺ Tree Index preferred rather than balanced BST/AVL for DB index

(b) B/B⁺ Tree:



(Disk block)

B/B⁺ Tree node

- The access cost (I/O cost) is less because many keys store in one node (number of levels of index will be less).
- The wastage of disk space is less.

ORDER: P

$$\max \text{key} = P-1$$

$$\min \text{key} = \lceil \frac{P}{2} \rceil - 1$$

1, 3, 5, 7, 9, 11, 13

ORDER: S

$$\max \text{key} = 4$$

ORDER: S

$$\min = \lceil \frac{S}{2} \rceil - 1 \Rightarrow 2$$

$$\max = 5-1 = 4$$

ORDER: S

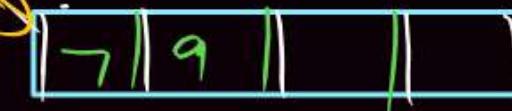
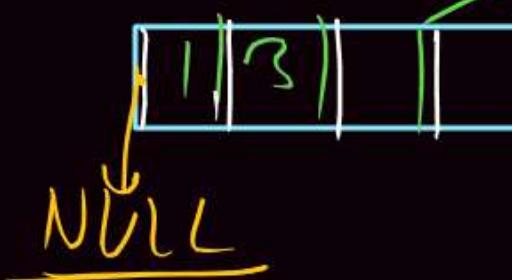
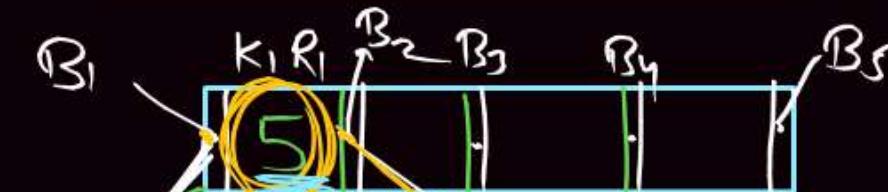
$$B_P = 5$$

$$\text{key} = 4$$

$$R_P = 4$$

L	3	5	7
---	---	---	---

$\leftarrow q$



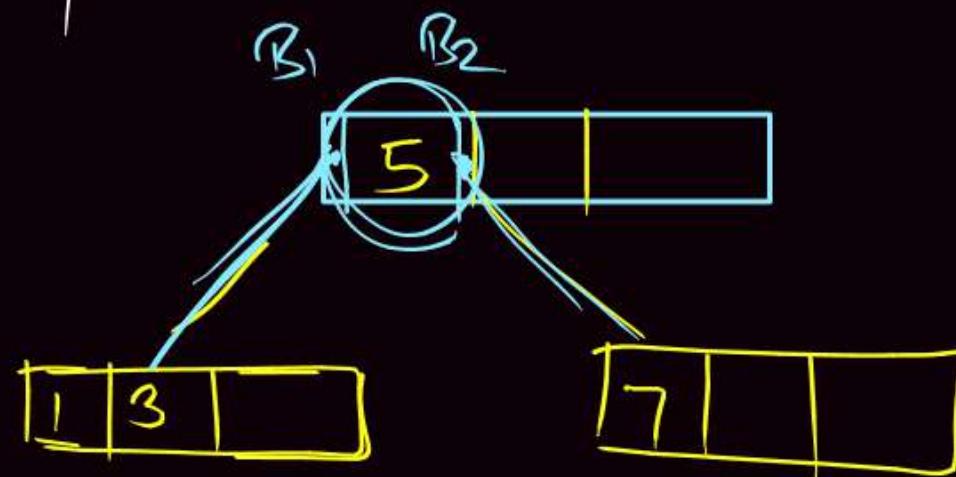
1, 3, 5, 7, 9, 11

ORDER = 4

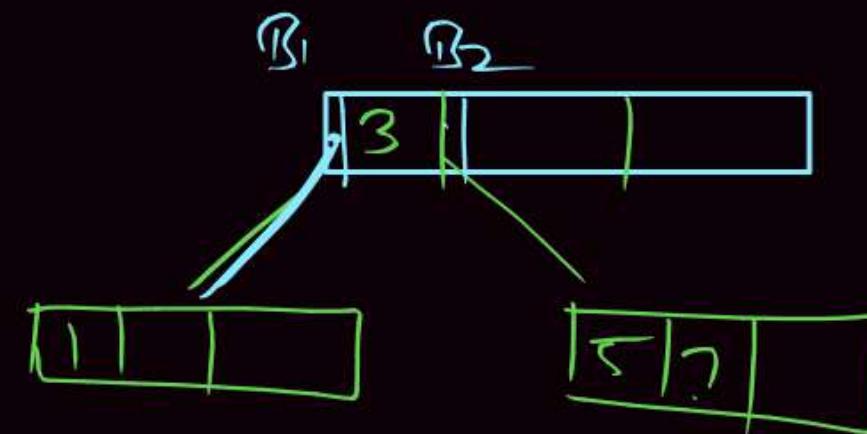
Max key = 3

R_p = 3

B_p = 4



Left Biasing
Left Node has
More weight



Right Biasing

B Tree



ORDER : P

$$B_P = P$$

$$\text{keys} = P - 1$$

$$R_P = P - 1$$

$$P \times B_P + (P-1) \text{key} + (P-1) R_P \leq \text{Block Size}$$

$$B_P = \text{key} + 1$$

ORDER : 5

$$\max B_P = 5$$

$$\begin{aligned} \max \text{key} &= 4 \\ \max R_P &= 4 \end{aligned}$$

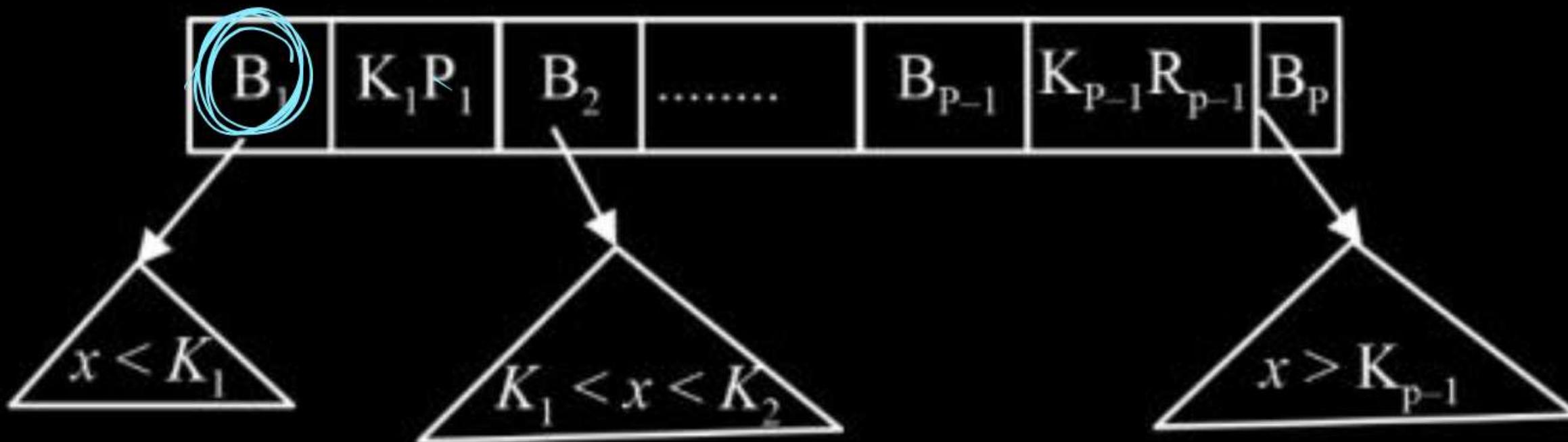
ORDER : 5

$$\min B_P [P] = \lceil \frac{5}{2} \rceil = 3 \quad \textcircled{3}$$

$$\min \text{keys} = [P] - 1 = 3 - 1 = 2 \quad \textcircled{2}$$

B Tree Definition

- Order P (degree): The maximum possible child pointers can be stored in B Tree node.
- Node structure: P child pointer, P - 1 Key and P - 1 Record Pointer (R_p).



\therefore order of node \Rightarrow $P \times (\text{size of block pointer}) + (P - 1) (\text{size of key field} + \text{size of record pointer}) \leq \text{Block size}$

B Tree Definition

1. Every internal node except root must be atleast $\lceil \frac{p}{2} \rceil$ block pointer and $\lceil \frac{p}{2} \rceil - 1$ keys and at most P block pointer and $(P - 1)$ keys.
2. Root can have at least 2 child/block pointer and 1 keys and at most P block pointer and $(p - 1)$ keys.
3. Every leaf node must be at same level.

& keys ascending order

B Tree Definition



❑ Advantages:

B Tree index best suitable for random access of any one record.

Example:

```
SELECT *
```

```
FROM R
```

```
WHERE A = 15;
```

∴ Worst case access cost = (k + 1) blocks

Best case access cost = 2 blocks

❑ Disadvantages:

B Tree index not suitable for sequential access of range of records.

Order P: The maximum possible pointers can be stored in B⁺ tree node.

1. Node Structure:

- Leaf node: (set of (key, RP) pairs and one block pointer)



$$\therefore \text{Order of leaf node} = (P - 1) [K + R_p] + 1 * B_p \leq \text{Block size.}$$

- Internal Node:

$$\text{Order of internal node} = P * B_p + (P - 1) * K \leq \text{Block size.}$$

- B⁺ Tree best suitable for random access of any one record and sequential access of range of records.
- I/O cost to access range of 'x' records sequential:

Internal | Non Leaf

$$P \times B_P + (P - 1) \log \leq \text{Block Size}$$

B^T Tree

- ① In Internal Node \Rightarrow No Read Pointer (B_j)
- ② All keys available in Leaf Node $x \leq k_1$
- ③ Leaf Node (key & R_p and \perp Block Pointer)

Q.

Consider a table T in a relational database with a key field K. A B-tree of order p is used as an access structure on K, where p denotes the maximum number of tree pointers in B-tree index node. Assume that K is 10 bytes long; disk block size is 512 bytes; each data pointer P_D is 8 bytes long and each block pointer P_B is 5 bytes long. In order for each B-tree node to fit in a single disk block, the maximum value of p is [GATE-2004 : 2 Marks]

P
W

- A 20
- B 22
- C 23
- D 32

$$P \times B_P + (P-1) \times \text{key} + (P-1) \times P_D \leq 512$$

$$P \times 5 + (P-1) \times 10 + (P-1) \times 8 \leq 512$$

$$5P + 10P + 8P - 18 \leq 512$$

$$23P \leq 540$$

$$P = \left\lfloor \frac{540}{23} \right\rfloor = \left\lfloor 23.04 \right\rfloor = 23$$

Q.

The order of an internal node in a B⁺-tree index is the maximum number of children it can have. Suppose that a child pointer takes 6 bytes, the search field value takes 14 bytes, and the block size is 512 bytes. What is the order of the internal node?

- A 24
- B 25
- C 26
- D 27

$$P \times B_P + (P-1) \text{ key} \leq \text{Block Size}$$

$$P \times 6 + (P-1) 14 \leq 512$$

$$6P + 14P - 14 \leq 512$$

$$20P = 526$$

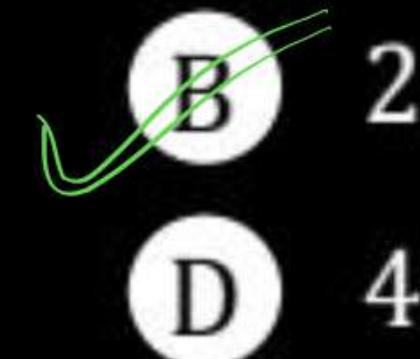
$$\begin{aligned} P &= \frac{526}{20} = 26.3 \\ &= 26 \text{ Ans} \end{aligned}$$

P
W

Q.

Consider a B^+ -tree in which the maximum number of keys in a node is 5. What is the minimum number of keys in any non-root node?

- A 1
- C 3



$$P-1 = 5$$

$$\boxed{P=6}$$

$$\text{Min key} = \lceil \frac{P}{2} \rceil - 1$$

$$\Rightarrow \lceil \frac{6}{2} \rceil - 1 = 3 - 1 = 2$$

$$\text{Max key} = \boxed{5} = P-1$$

$$B_P = \text{key} + 1$$

$$\boxed{B_P = 6}$$

$$\boxed{\text{odd} = 6}$$

$$\text{Min key} = \lceil \frac{P}{2} \rceil - 1$$

$$\Rightarrow \lceil \frac{6}{2} \rceil - 1 = 3 - 1$$

= 2 Ans

**THANK
YOU!**

