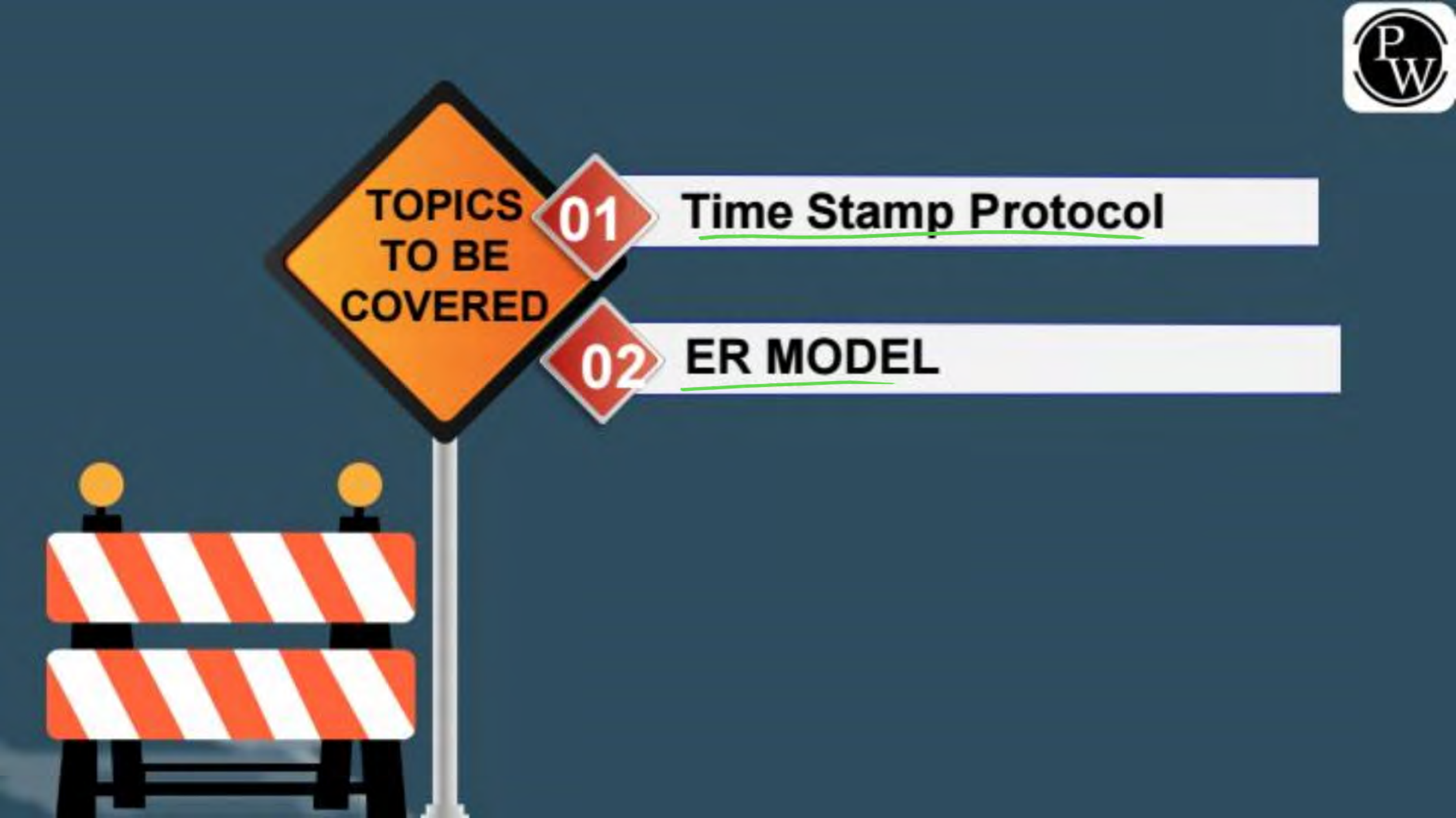
COMPUTER SCIENCE

Database Management System

ER Model

Lecture_1

Vijay Agarwal sir


TOPICS TO BE COVERED
01
Time Stamp Protocol
02
ER MODEL

Time Stamp Protocol

TS (T_i)

Data Item → RTS(Q)

Time Stamp → WTS(Q)

(1) T_i : issue Read(Q)

TS(T_i) < WTS(Q) ; Reject

TIMESTAMP BASED CONCURRENCY CONTROL

Timestamp-Based Protocols

- Each transaction T_i is issued a timestamp $TS(T_i)$ when it enters the system.
  - Each transaction has a unique timestamp
  - Newer transaction have timestamp strictly greater than earlier ones
  - Timestamp could be based on a logical counter
    - Real time may not be unique
    - Can use (wall-clock time, logical counter) to ensure
- Timestamp-based protocols manage concurrent execution such that **time-stamp order = serializability order**
- Several alternative protocols based on timestamps

Timestamp-Based Protocols

The **timestamp ordering (TSQ) protocol**

- Maintains for each data Q two timestamp values:
  - **W-timestamp**(Q) is the largest time-stamp of any transaction that executed write (Q) successfully.
  - **R-timestamp** (Q) is the largest time-stamp of any transaction that executed **read** (Q) successfully.
- Imposes rules on read and write operations to ensure that
  - any conflicting operations are executed in timestamp order
  - out of order operations cause transaction rollback

Note

T_i : Read(Q)

① $TS(T_i) < WTS(Q)$ — Reject & Rollback

T_i : Write(Q)

① $TS(T_i) < RTS(Q)$

② $TS(T_i) < WTS(Q)$

] Write operation Reject & Rollback

Conflict operation

T_i T_j

R(Q) - W(Q)

W(Q) - R(Q)

W(Q) - W(Q)

V.V.V.V Imp.

Note

All conflicting operation order Must be Same as Transaction order (Timestamp order)

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

(i) If TS (T_i) < WTS (Q): Read operation Reject & T_i Rollback.

(ii) If TS (T_i) ≥ WTS(Q): Read operation is allowed

and Set Read – TS(Q) = max[RTS(Q), TS (T_i)]

CHECK

R(Q) ⋮ WTS(Q)

W(Q) ⋮ RTS(Q)

W(Q) ⋮ WTS(Q)

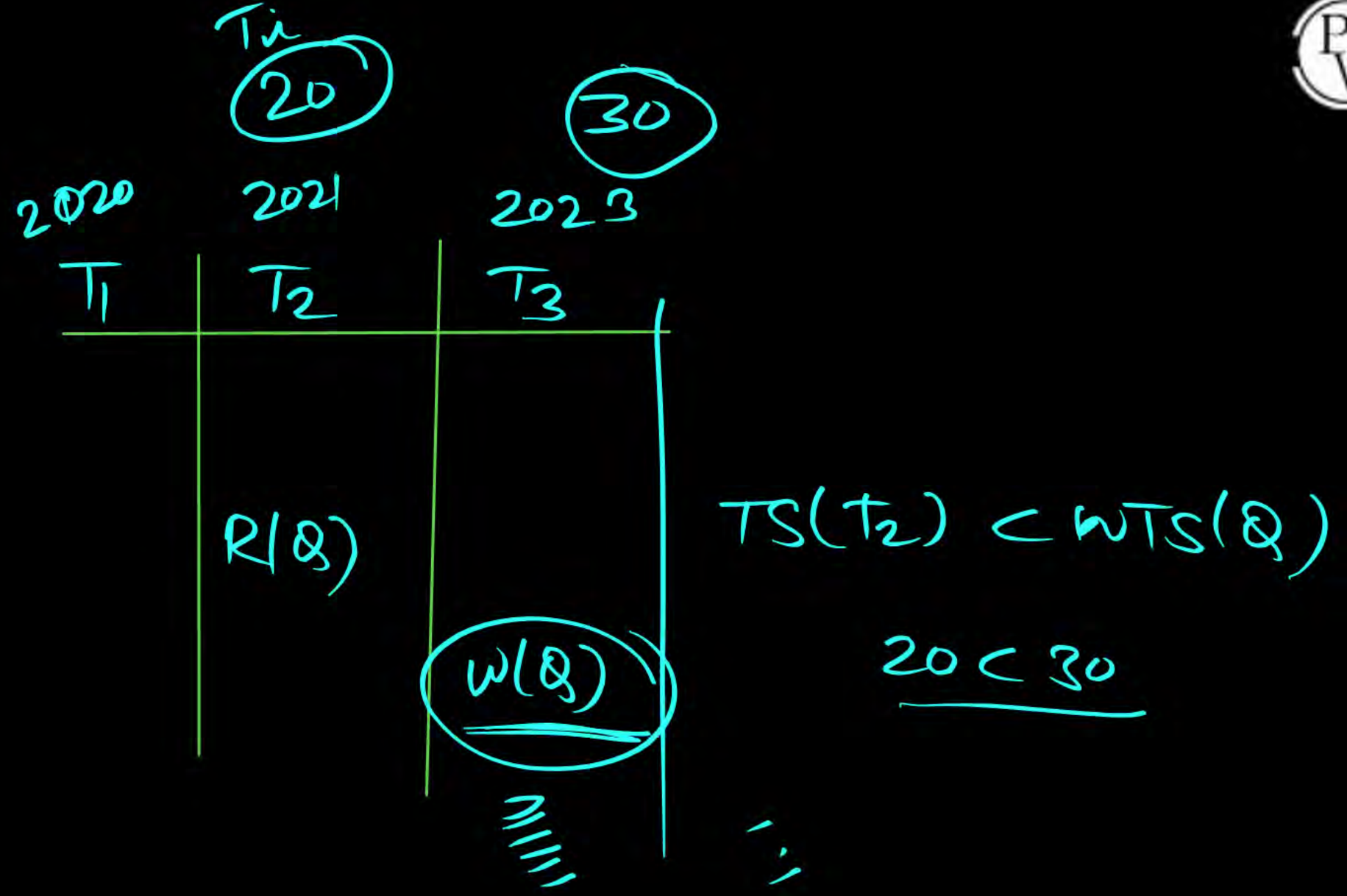
II: T_i - Write(Q) (Transaction T_i Issue Write(Q) Operation)

(i) If TS (T_i) < RTS (Q): Write operation Reject & T_i Rollback.

(ii) If TS (T_i) < WTS(Q): Write operation Reject & T_i Rollback.

(iii) Otherwise execute write (Q) operation

Set Read WTS(Q) = TS (T_i)


Ti
20
30
2020
2021
2023
T1
T2
T3
R(Q)
W(Q)
TS(T2) < WTS(Q)
20 < 30
∴

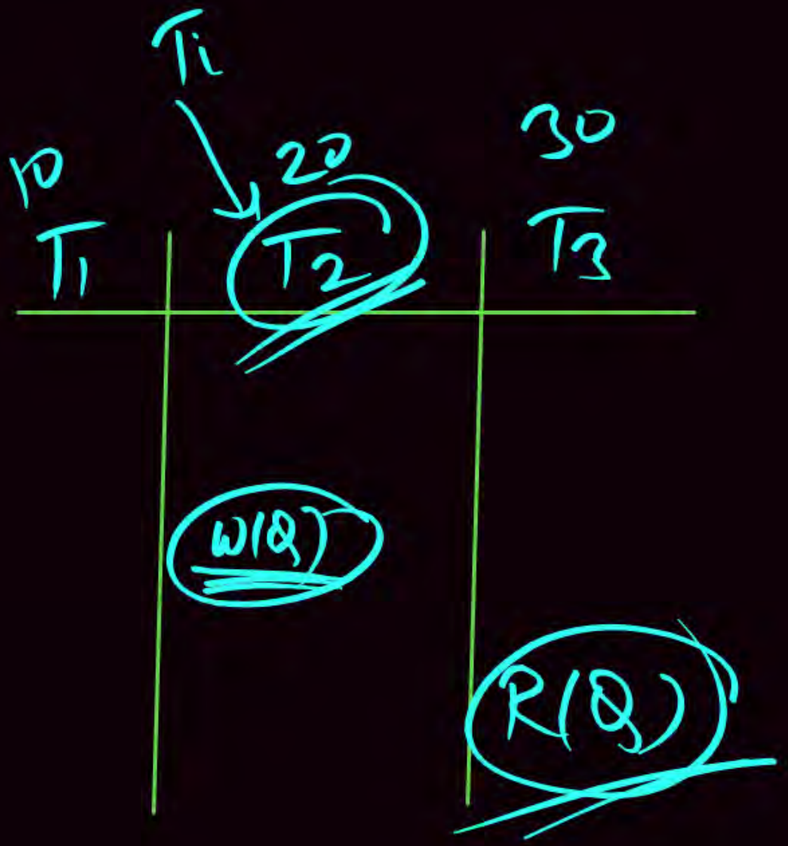
Timestamp-Based Protocols (Cont.)

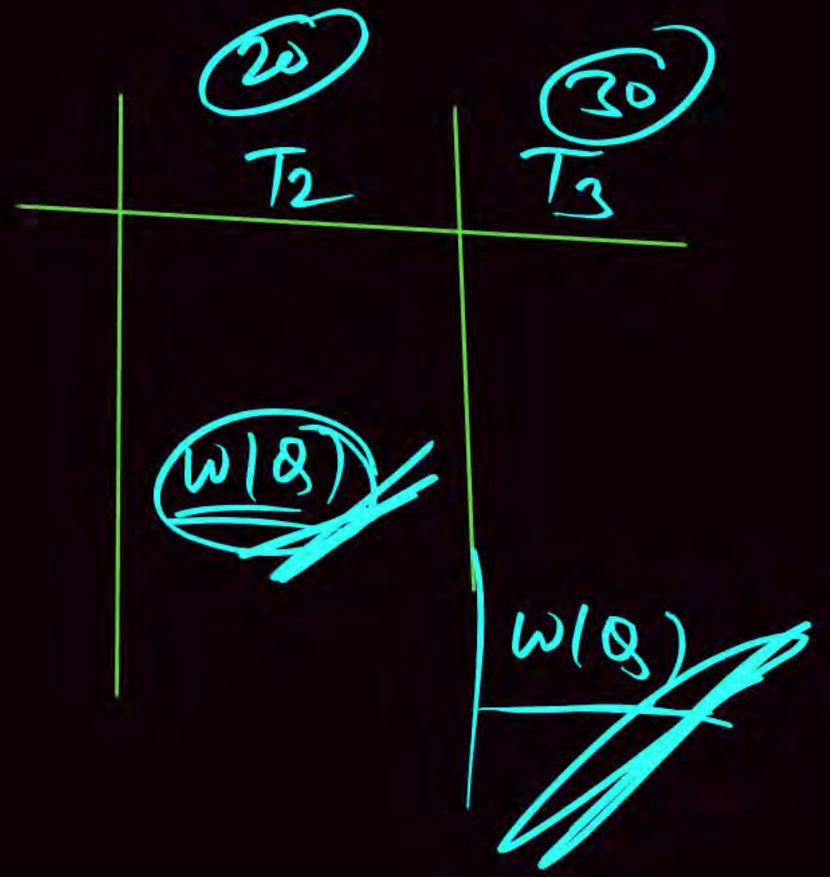
- Suppose a transaction T_i issues a **read** (Q)
  1. If $TS(T_i) \leq$ **W**-timestamp (Q), then T_i needs to read a value of Q that was already overwritten.
     - Hence, the **read** operation is rejected, and T_i is rolled back.
  2. If $TS(T_i) \geq$ **W**-timestamp (Q), then the **read** operation is executed, and R-timestamp(Q) is set to.

     max(R-timestamp (Q), TS (T_i)).

Timestamp-Based Protocols (Cont.)

- Suppose a transaction T_i issues **write**(Q)
  1. If $TS(T_i) <$ **R**-timestamp(Q), then the value of Q that T_i is producing was needed previously, and the system assumed that the value would never be produced.
     - Hence, the **write** operation is rejected, and T_i is rolled back.
  2. If $TS(T_i) <$ **W**-timestamp (Q), then T_i is attempting to write an obsolete value of Q.
     - Hence, this **write** operation is rejected, and T_i is rolled back.
  3. Otherwise, the **write** operation is executed, and W-timestamp(Q) is set to $TS(T_i)$.


Ti
10
T1
20
T2
30
T3
W(Q)
R(Q)


20
T2
30
T3
W(Q)
W(Q)

I

① $T_1 \Rightarrow R(A)$ $\quad$ W.TS(A) = 0

$TS(T_1) = 10$

T_1 Read allowed.

② $T_2 \Rightarrow W(A)$ (20) $\quad$ R(A) = 10, W(A) = 0

T_2 Write allowed

$TS(T_2) \not< RTS(A)$

$TS(T_2) \not< WTS(A)$

③ (10) $T_1 \Rightarrow W(A)$ Issue RTS(A) = 10, WTS(A) = 20

$TS(T_1) < WTS(A)$

10 < 20 ; Reject, Not TSP

$TS(T_1) = 10$

$TS(T_2) = 20$

$TS(T_1) < TS(T_2)$

All Conflict operation order Must be $T_1 \rightarrow T_2$.

$R(A) - W(A) \Rightarrow T_1 \rightarrow T_2$

$W_2(A) - W_1(A) = T_2 \rightarrow T_1$ X Not TSP.

(1)

R(A)

W(A)

W(A)

NOT TSP.

$TS(T_1) < TS(T_2)$

$T_1 \rightarrow T_2$

(2)

R(A)

R(A)

W(A)

10 < 20

$TS(T_1) < RTS(A)$

Reject & Roll back

$TS(T_1) < TS(T_2)$

$T_1 \rightarrow T_2$

$R_2(A) - W_1(A)$

$T_2 \rightarrow T_1$

Not TSP.

(3)

R(A)

W(A)

R(B)

W(B)

YES TSP.

$TS(T_1) < TS(T_2)$

$T_1 \rightarrow T_2$

(4)

r(A)

r(B)

W(A)

$TS(T_1) < TS(T_2)$

$T_1 \rightarrow T_2$

$R_2(A) - W_1(A)$

$T_2 \rightarrow T_1$

Not TSP.

Thomas Write Rule (View Serializable)

Ti: Read(Q)

TS(Ti) < WTS(Q) ; Reject & Rollback Same as TSP

Ti: write(Q)

TS(Ti) < RTS(Q) ; Reject & Ti Rollback. Same as TSP

TS(Ti) < WTS(Q): Ignore, No Rollback

Thomas' Write Rule

- 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 don, 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)

1. TS (T_i) < RTS(Q) : Rollback (Reject)
2. TS(T_i) < WTS(Q) : Write operation is Ignored and No Roll back

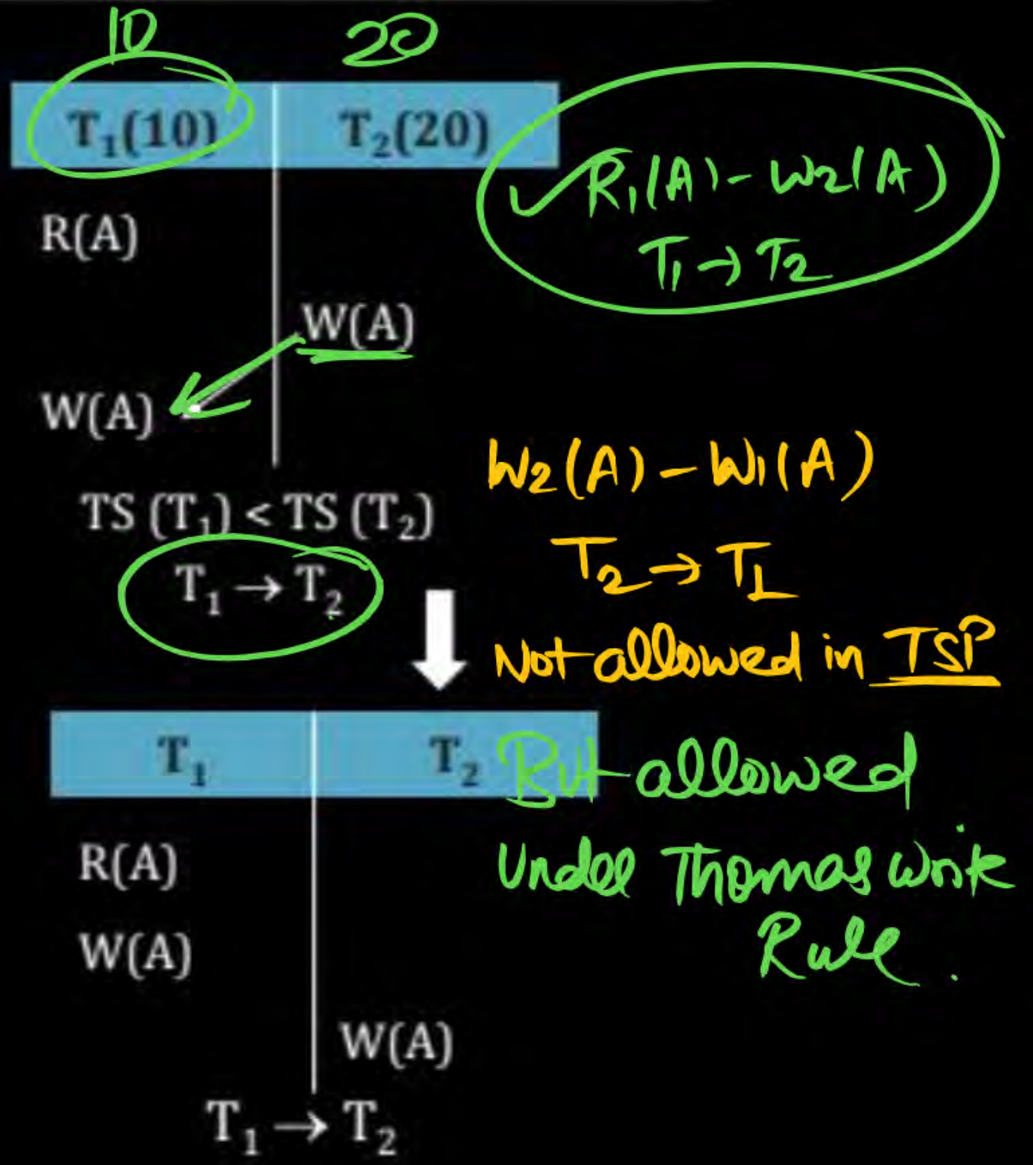
Same as TSP

Time Stamp Protocol: Ensure serializability deadlock free but starvation possible

Deadlock Prevention Algorithm

(1) Wait-Die (2) Wound-wait

Older Younger



Q.1

Consider the following database schedule with two transactions, T_1 and T_2.

$S = r_2(X); r_1(X); r_2(Y); w_1(X); r_1(Y); w_2(X); a_1; a_2$

where $r_i(Z)$ denotes a read operation by transaction T_i on a variable Z, $w_i(Z)$ denotes a write operation by T_i on a variable Z and a_i denotes an abort by transaction T_i

Which one of the following statements about the above schedule is TRUE?

[MCQ:2016–2M]

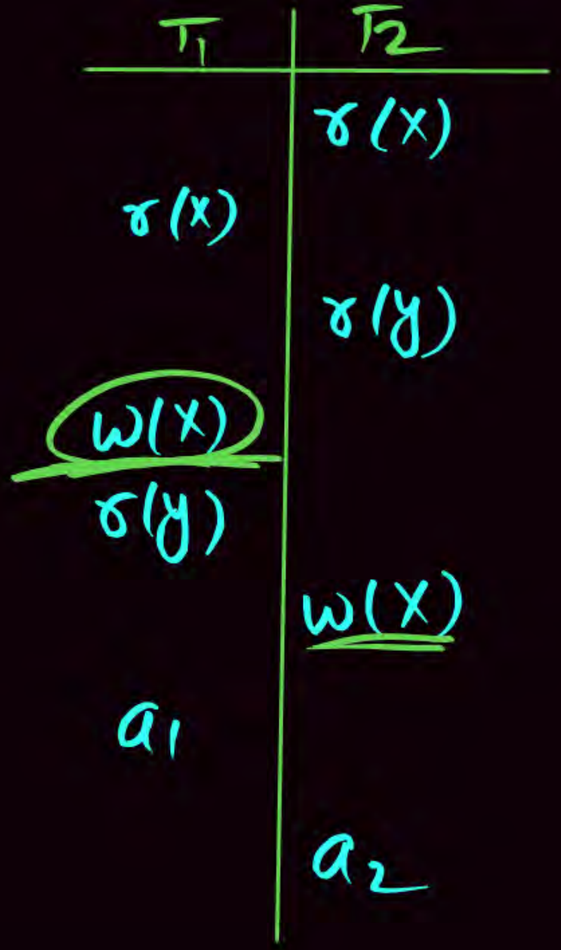
A. S is non-recoverable

B. S is recoverable, but has a cascading abort

C. S does not have a cascading abort

D. S is strict

Ans (C)



Recoverable

Cascadeless

But Not Strict Recoverable.

Q.2

Let S be the following schedule of operations of three transactions T_1, T_2 and T_3 in a relational database system:

$$R_2(Y), R_1(X), R_3(Z), R_1(Y), W_1(X), R_2(Z), W_2(Y), R_3(X), W_3(Z)$$

Consider the statements P and Q below:

P: S is conflict-serializable. — True

Q: If T_3 commits before T_1 finishes, then S is recoverable. — False

Which one of the following choices is correct?

[MCQ: 2021-2M]

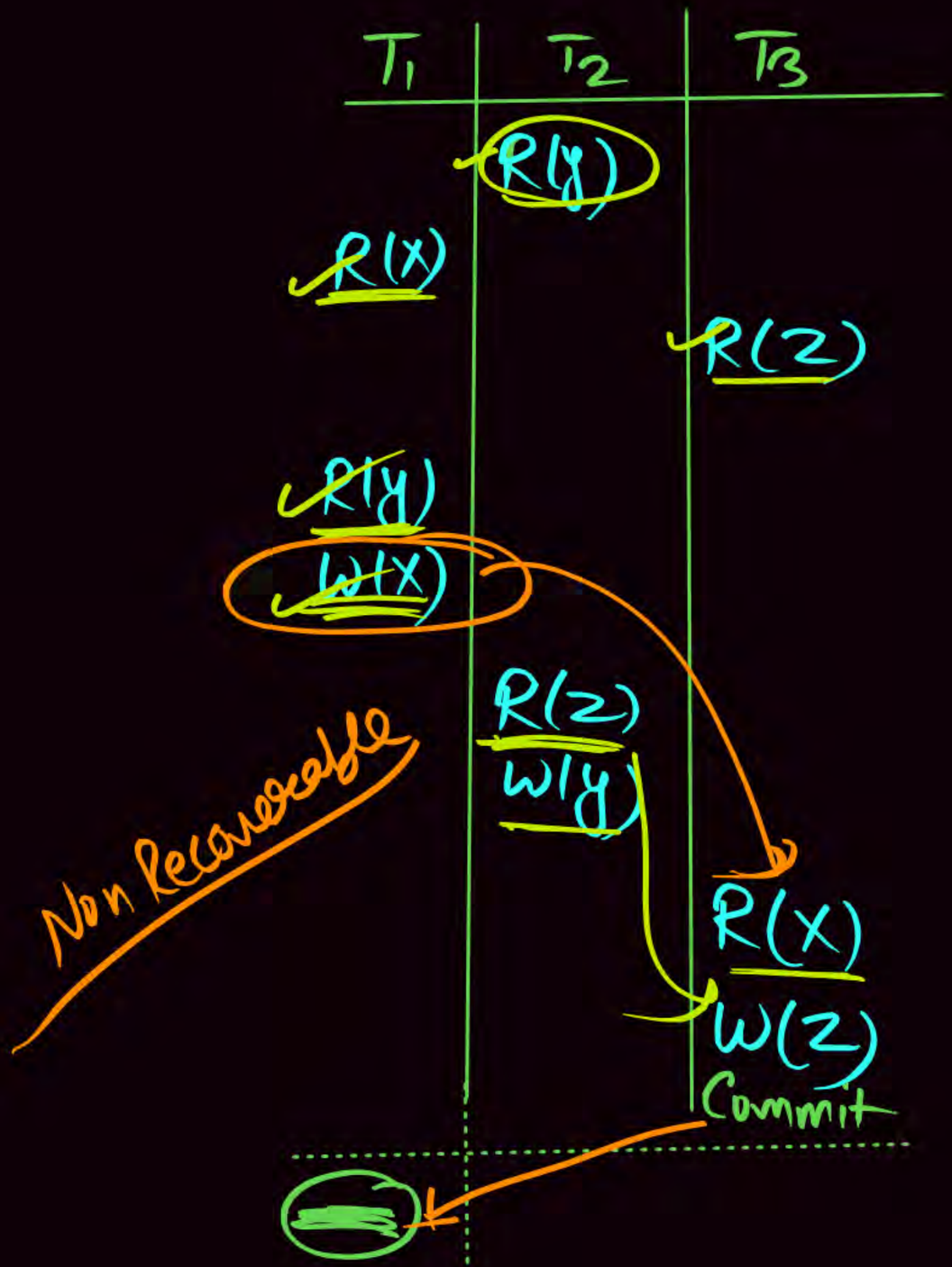
(A) Both P and Q are true.

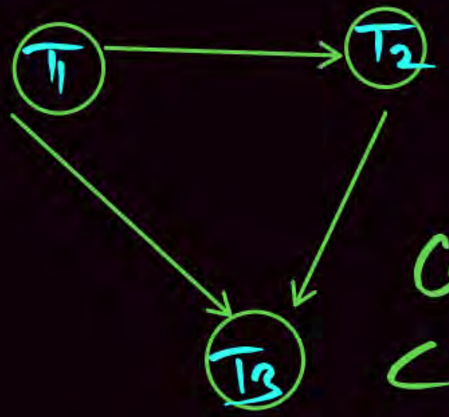
(B) P is true and Q is false. ✓

(C) P is false and Q is true.

(D) Both P and Q are false.

Ans (B)





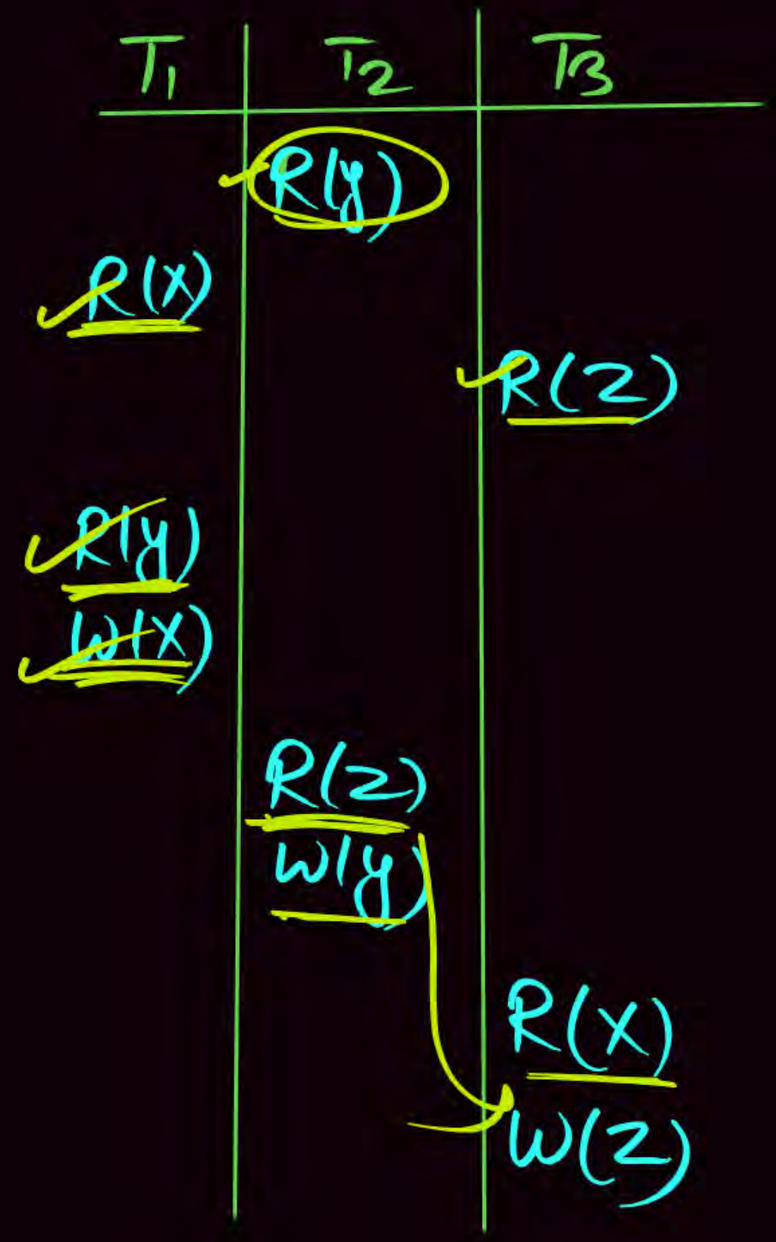
Conflict serial

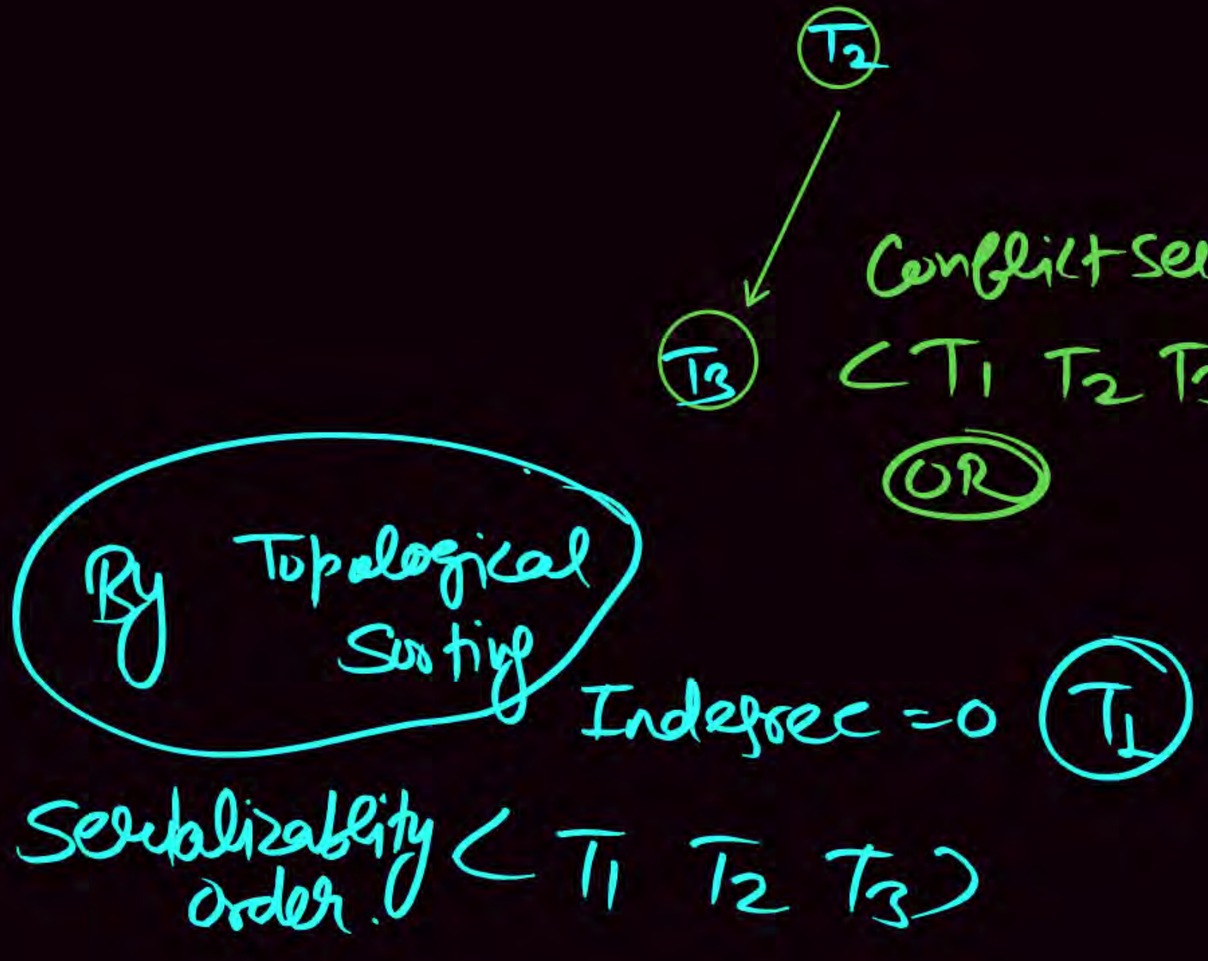
$\langle T_1\ T_2\ T_3 \rangle$

OR

Topological Sorting

T_1 : Indegree = 0


T1
T2
T3
R(y)
R(x)
R(z)
R(y)
W(x)
R(z)
W(y)
R(x)
W(z)


T2
T3
Conflict serial
< T1 T2 T3 >
OR
By Topological Sorting
Indegree = 0
T1
Serializability order : < T1 T2 T3 >

Q.

Consider a simple checkpointing protocol and the following set of operations in the log.

(start, T4); (write, T4, y, 2, 3); (start, Tl);
(commit, T4); (write, T1, z, 5, 7);
(checkpoint);
(start, T2); (write, T2, x, 1, 9); (commit, T2);
(start, T3); (write, T3, z, 7, 2);

If a crash happens now and the system tries to recover using both undo and redo operations, what are the contents of the undo list and the redo list

[MCQ: 2015–2M]

A. Undo: T3, T1; Redo: T2

B. Undo: T3, T1; Redo: T2, T4

C. Undo: none; Redo: T2, T4, T3, Tl

D. Undo: T3, Tl, T4; Redo: T2

Redo : T_2

Undo : T_1, T_3.

Ans (A).

CHECKPOINT : Those Transaction Commit before Checkpoint Neither Require Redo (nor) Undo operation.

But Those Transaction Commit After Checkpoint Require Redo operation.

& Those Not Committed till Now Require Undo operation.

**

UNDO & Redo

Concept

(Write T_i Data Item O.V N.V]

Transaction No.

Old Value

New Value

< Write T_9 A 5 7 >

< Write T_{10} A 7 11 >

UNDO : OLD Value

REDO : New Value

A = 11 ⇒ 7 ⇒ 5 ⇒ A = 5

UNDO CASE

OLD Value

Redo

New Value

A = 5 ⇒ 7 ⇒ 11

A = 11

CHAPTER 2 : Transaction

Chanuka &

(a) cc with Enjoying

(b) cc

(c) c

(d) Doubt

Entity – Relationship Model (Conceptual Design)

Entity : object

- Physical
  - Book.
- Logical
  - Account

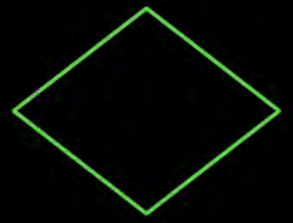
⇓

STUDENT

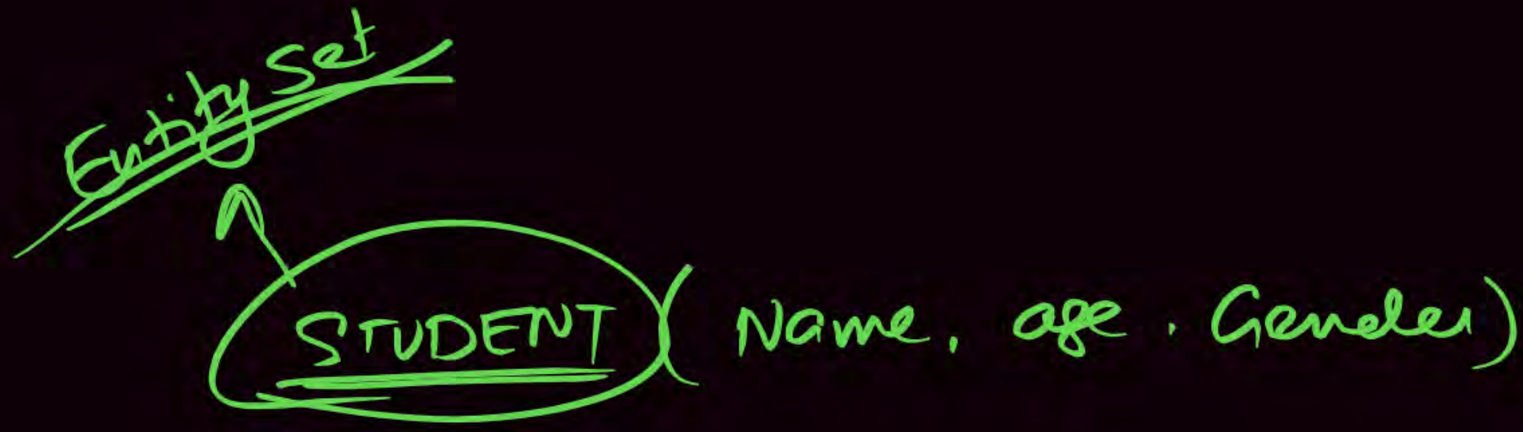
Entity Set : Collection of Similar Entity.

Relationship → Verb.

Relationship Set



Diamond



Entity → Abhay 21 Male

Mohita 22 Female

Ajay 25 Male.

Entity

An entity is an object that exists and is distinguishable from other objects.

- Example: Specific person

Entity Set

- An entity set is a set of entities of the same type that share the same properties.
  - Example: set of all persons, companies

- Entities have attributes
  - Example: people have names and addresses

Entity sets instructor and student

instructor_ID	instructor_name
76766	Crick
45565	Katz
10101	Srinivasan
98345	Kim
76543	Singh
22222	Einstein

instructor

advisior

student_ID	student_name
98988	Tanaka
12345	Shankar
00128	Zhang
76543	Brown
76653	Aoi
23121	Chavez
44553	Peltier

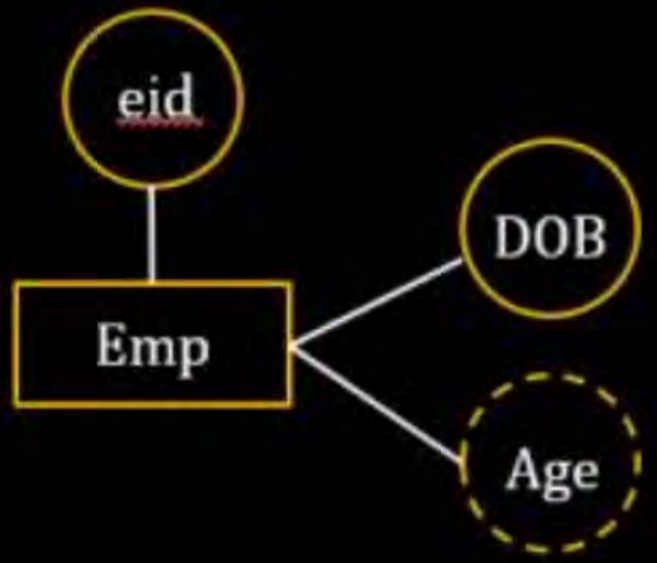
student

Entity Set

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:



Relationship Sets

- A relationship is an association among several entities
  - Example:

    44553(Peltier) advisor 22222(Einstein)

    student entity relationship set instructor entity
- A relationship set is a mathematical relation among n ≥ 2 entities, each taken from entity sets

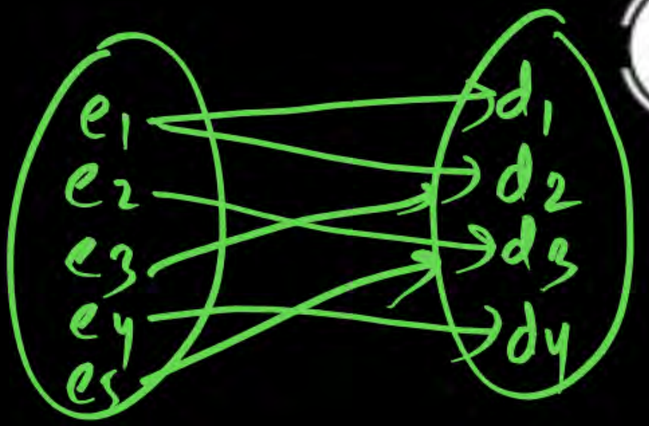
  $\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$

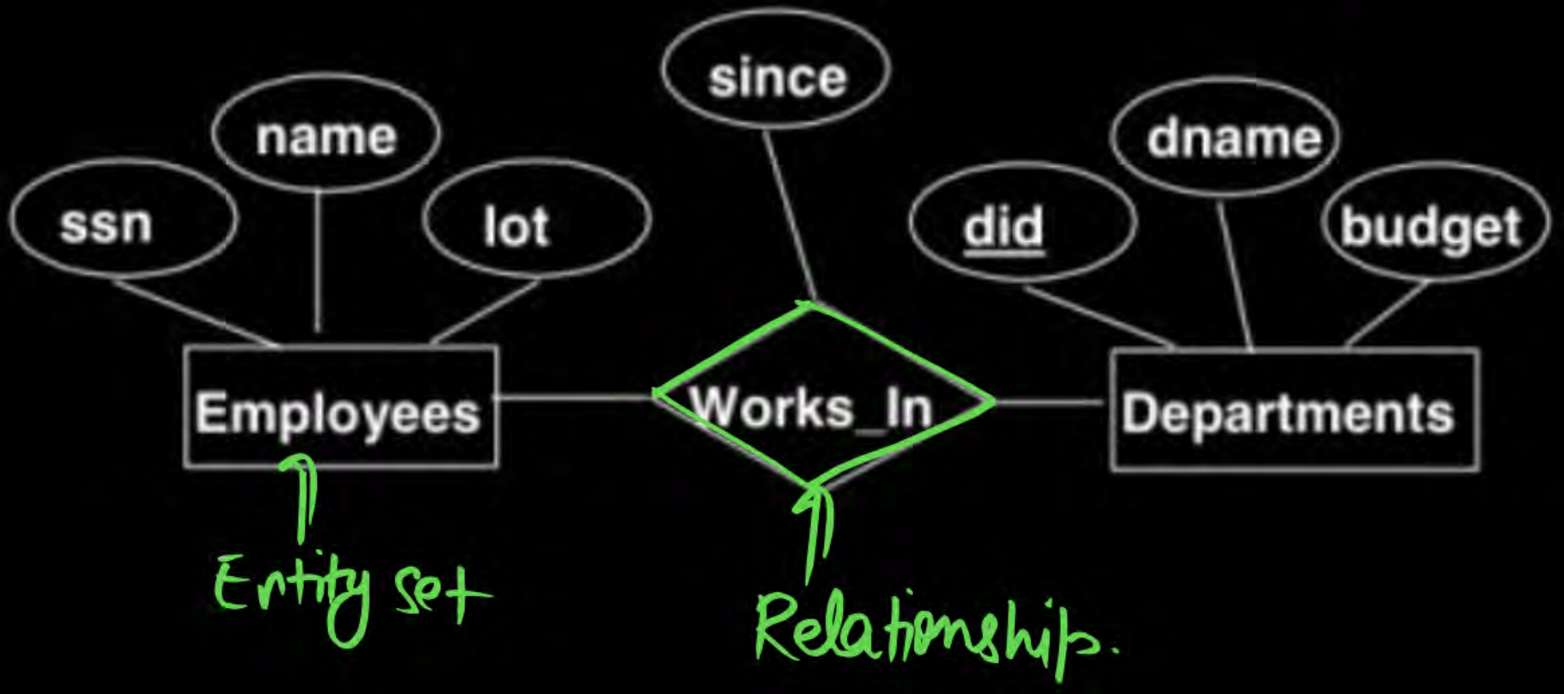
  where $(e_1, e_2, \ldots, e_n)$ is a relationship
  - Example:

    $(44553, 22222) \in$ advisor or

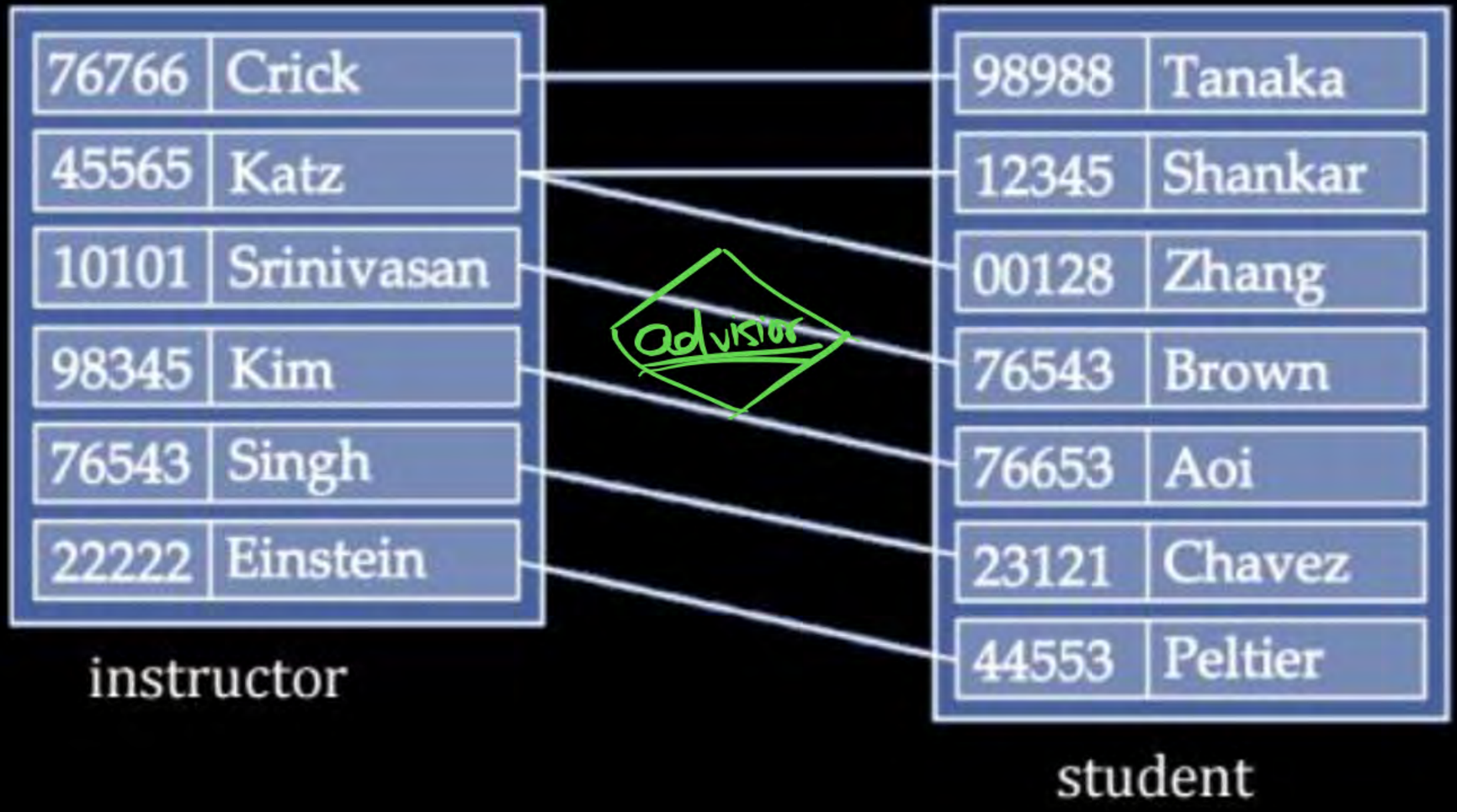
Relationship Sets

- Collection of similar relationships.





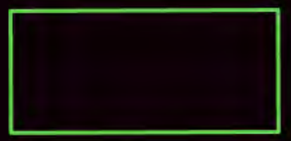
Relationship Set advisor



1

Entity

Entity Set



2

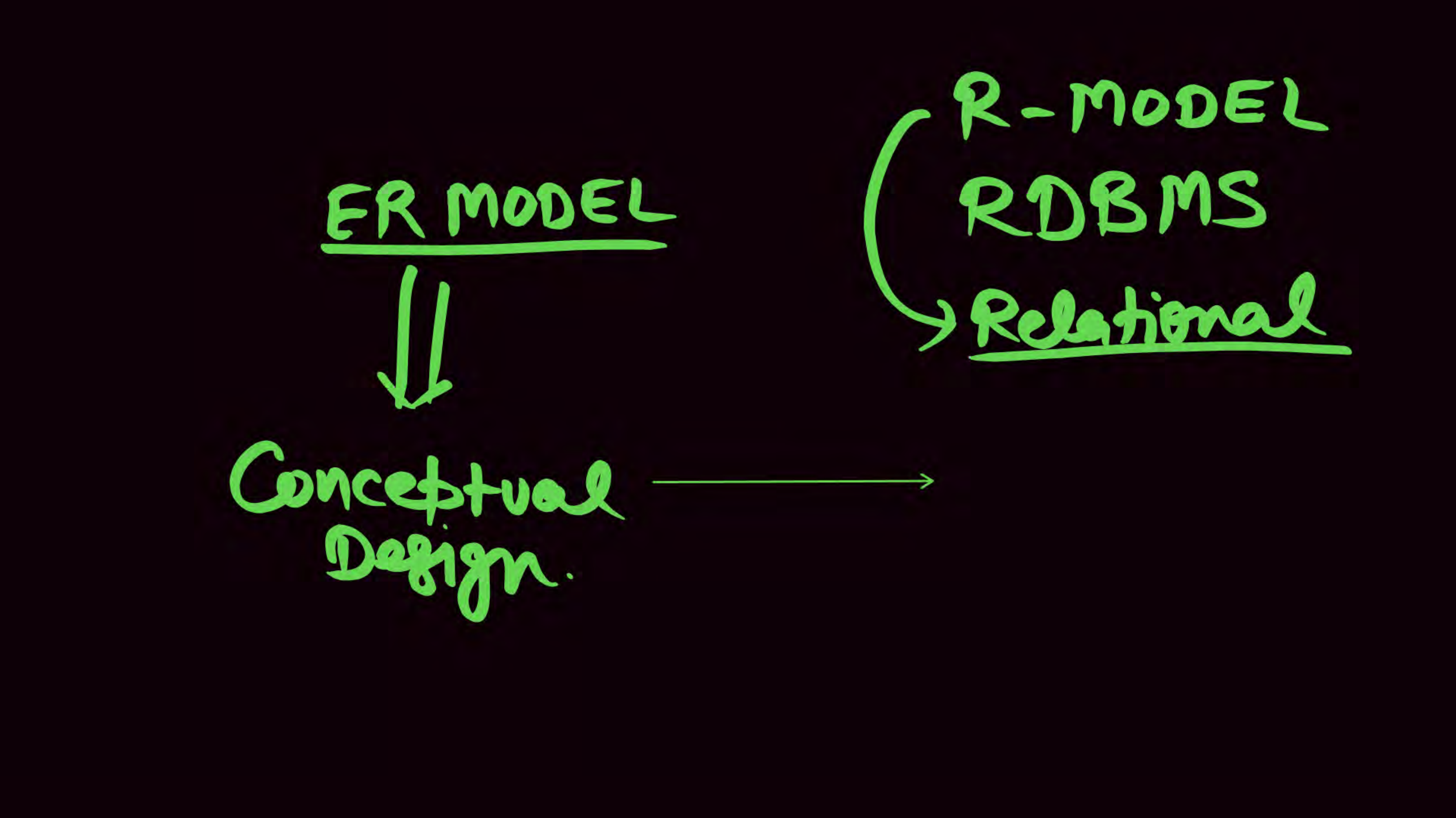
Relationship

Relationship Set

③ Attribute :

⇓

Which describe the Entity.


ER MODEL
Conceptual Design.
R-MODEL
RDBMS
Relational

Relationship Sets

- An attribute can also be property of a relationship set.
- For instance, the advisor relationship set between entity sets instructor and student may have the attribute date which tracks when the student started being associated with the advisor

Attributes

Attributes are properties used to describe an entity.

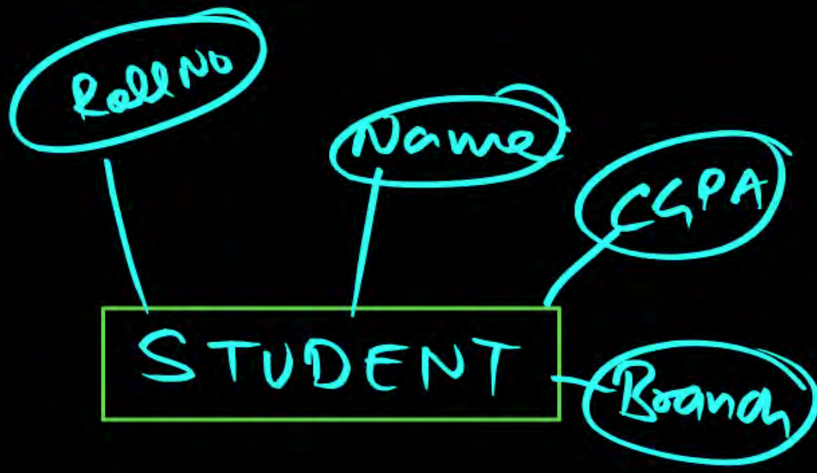
Attribute types:

(1) Simple and composite attributes.

(2) Single-valued and multivalued attributes

(3) Stored and Derived attributes

(4) key attribute



① Simple Attribute

Which Can not be Divide further.

eg RollNo
Gender

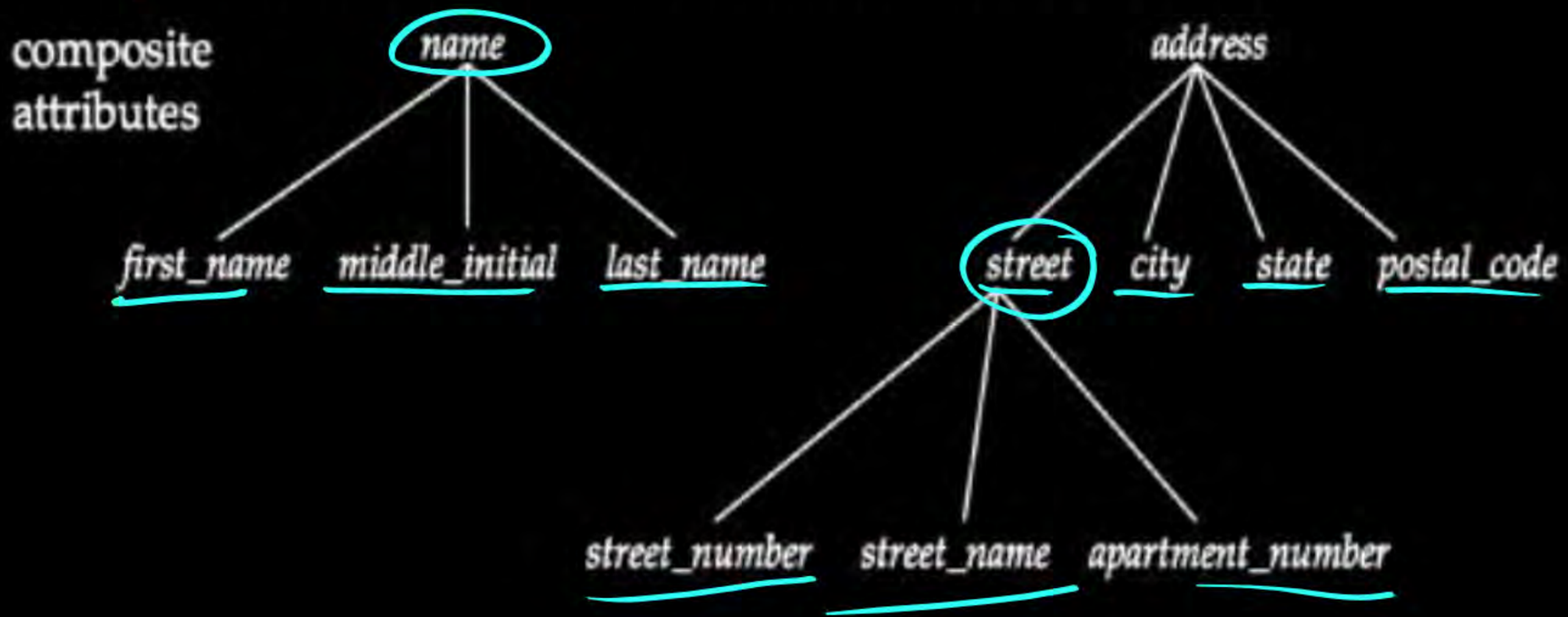
② Composite Attribute

Which Can be Divide further.

eg

1) Simple & Composite attribute

- Simple
  Each entity has a single atomic value for the attribute. For example, SSN or Sex (Gender). Aadhar Card Number,
- Composite
  The attribute may be composed of several components. For example



2) Single-valued and multivalued attributes

Single Valued Attribute: Which take One Value Per entity.

Eg Roll No
Gender.

Multivalued Attribute: Which takes More than One Value Per entity.

Eg

Note
Multivalued Attribute Represented by

3) Stored and Derived attributes

Stored Attribute: which does not Require Any Updation.

(eg) ① D.O.B (Date of Birth)

Derived Attribute: the value is Derived from other Attribute

(eg) Age.

If D.O.B : 18/04/2000

then

In 2023 : you are 23 year old

In 2050 : 50 year old

In 2090 : 90 year old

In 2110 : 110 year old.

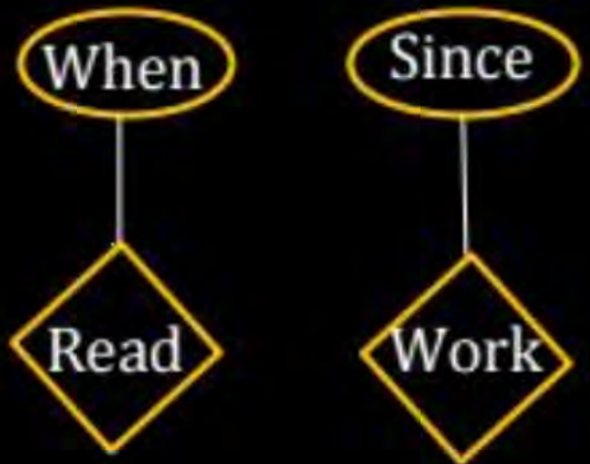
4) Key and Descriptive attributes

Key Attribute:

Which uniquely Identify an entity in the entity set.

Descriptive Attribute:

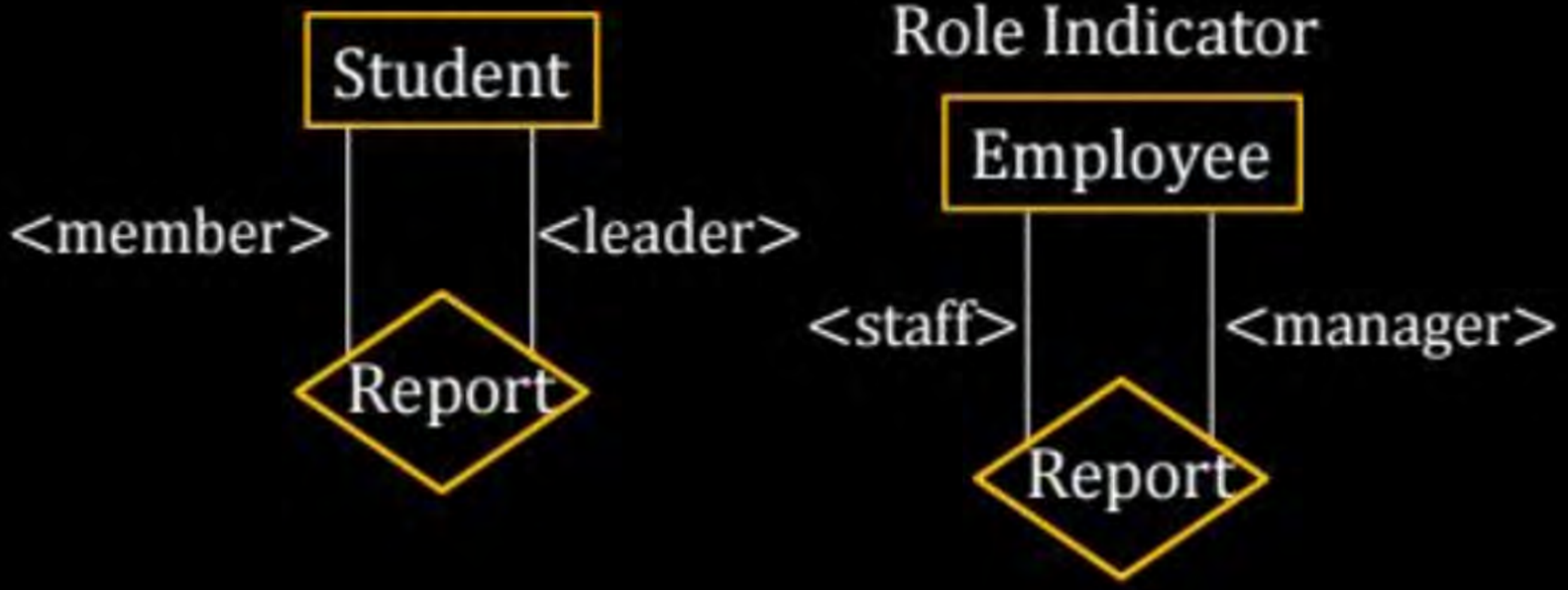
Which gives information about the relationship set



Degree of Relationship Set

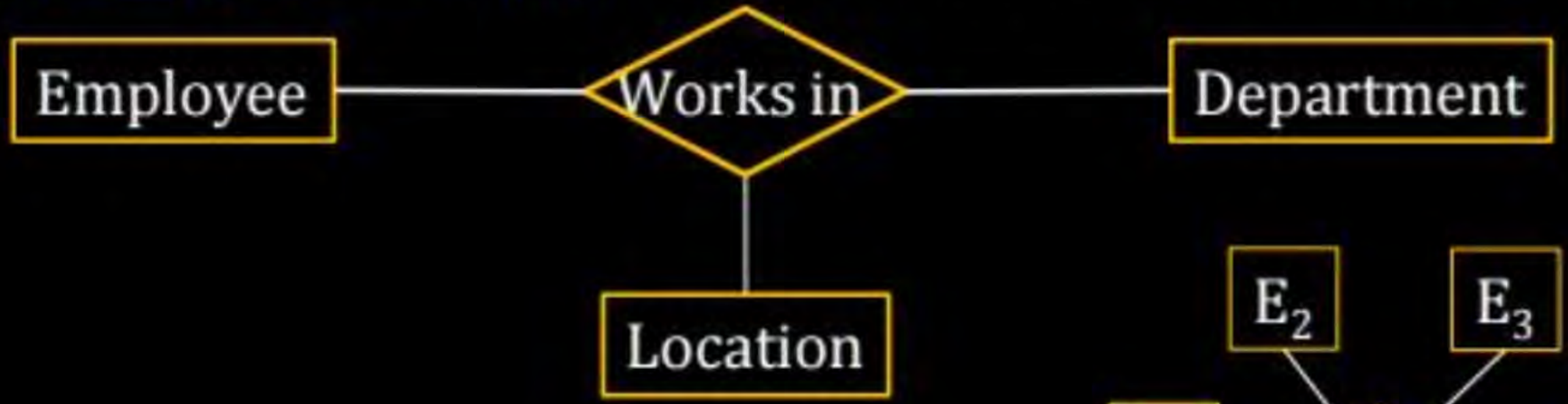
Degree of Relationship Set: Specifies the numbers of Entity set participate in a relationship set

1) UNARY: Relationship among two entities of the same entity set [Recursive Relationship Set]



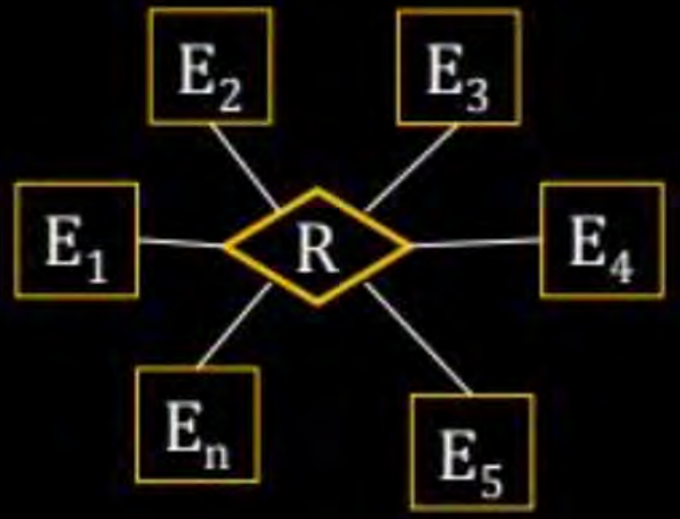
2) Binary Relationship: The relationship among two entity set

2) Ternary Relationship: The relationship among three entity set



3) n-ary Relationship:

The relationship among n-entity set

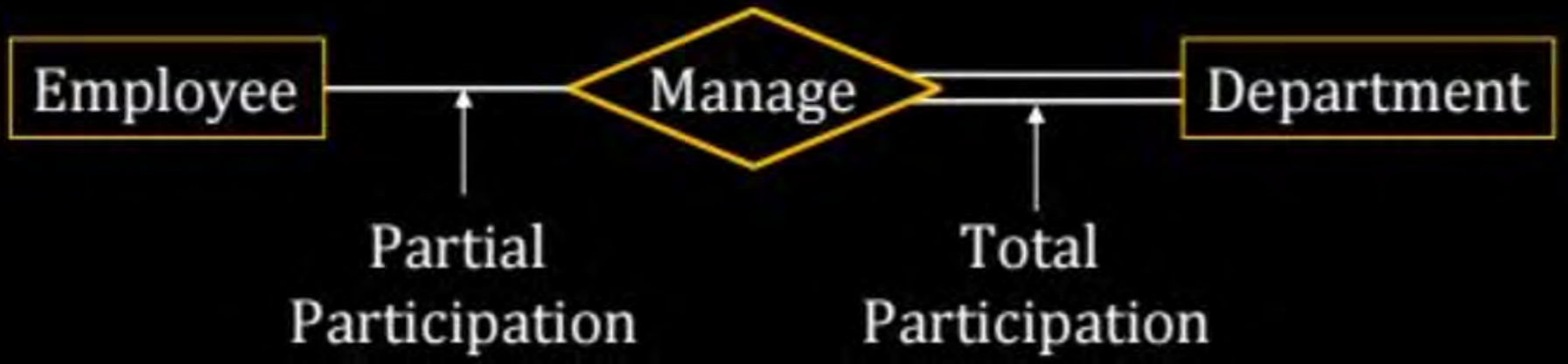


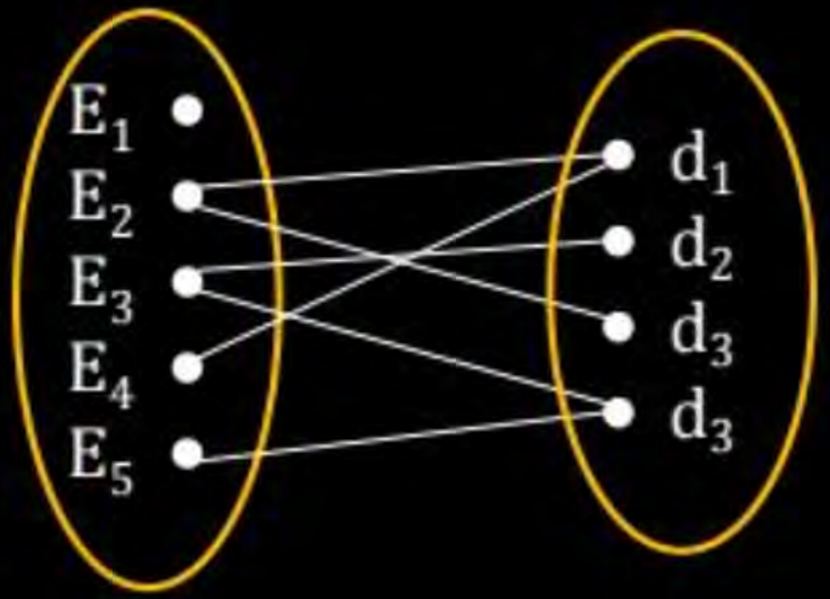
Participation Constraint

If every entity in the entity set participates in a relationship set is called

Total participation denoted by double line (thick line) otherwise it is called partial participation (thin line or single line)

Q. Each department is managed by at least one employee





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)

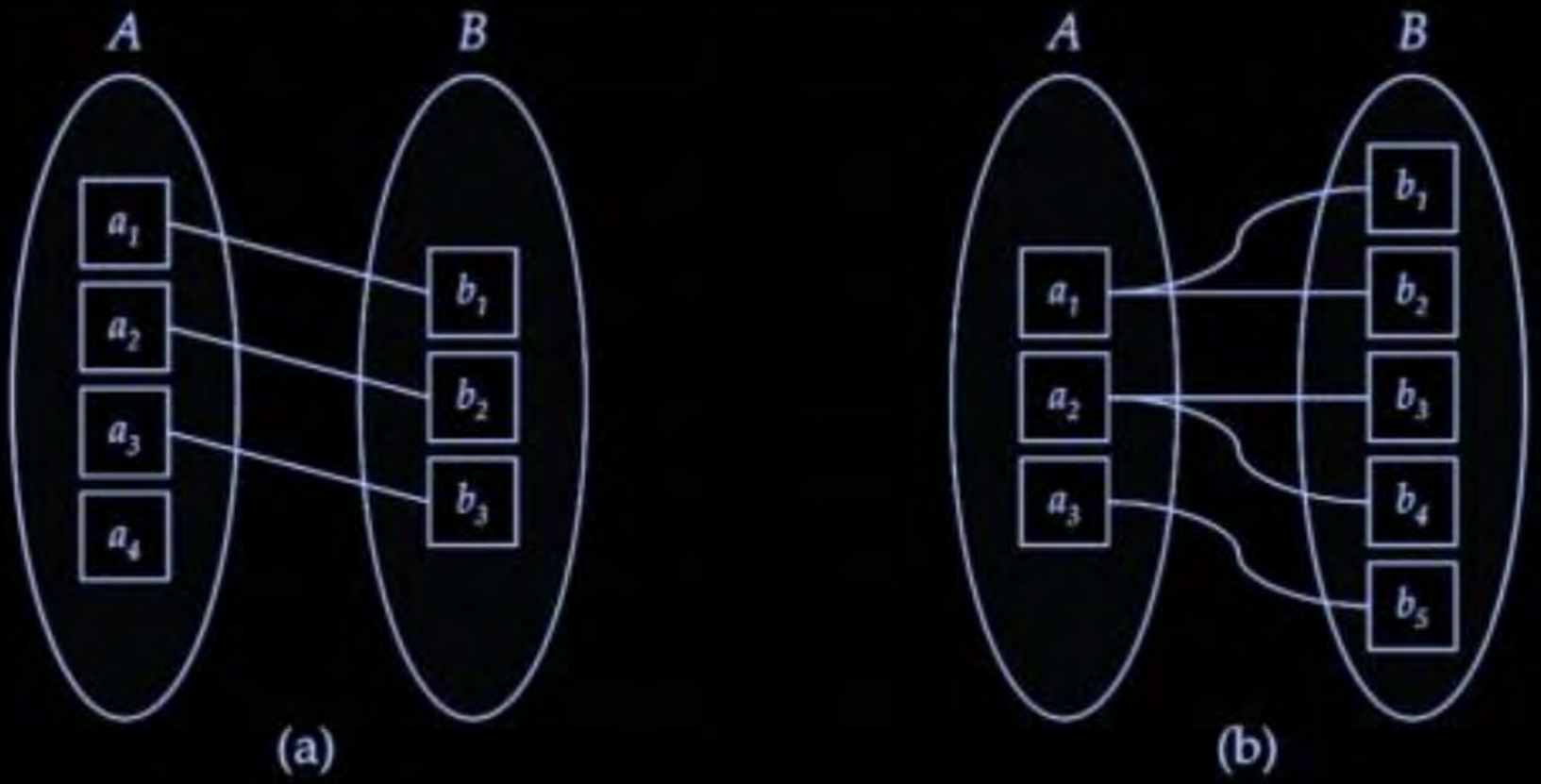
Example : Consider Emp and Dept entity set.

Manages relationship set such that each dept must have manager.

Mapping Cardinality Constraints

- Express the number of entities to which another entity can be associated via a relationship set.
- Most useful in describing binary relationship sets.
- For a binary relationship set the mapping cardinality must be one of the following types:
  - One to one
  - One to many
  - Many to one
  - Many to many

Mapping Cardinalities

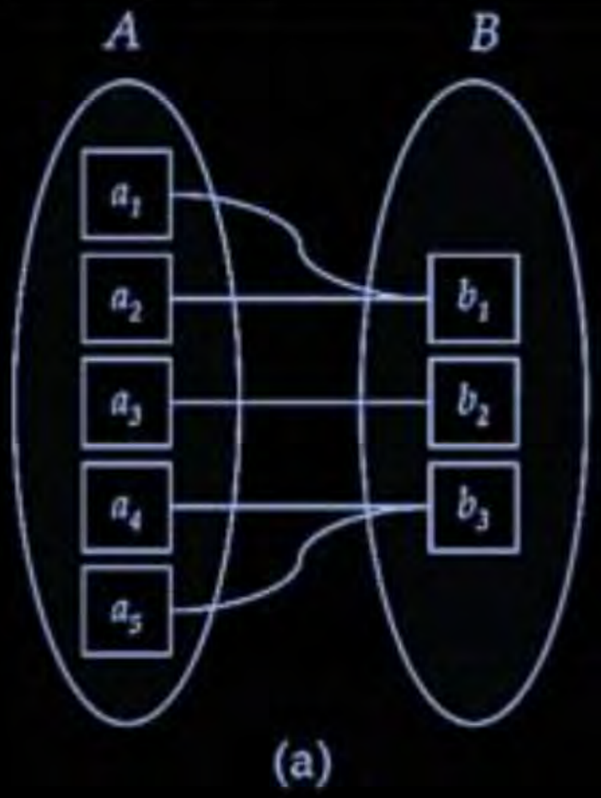


One to one

One to many

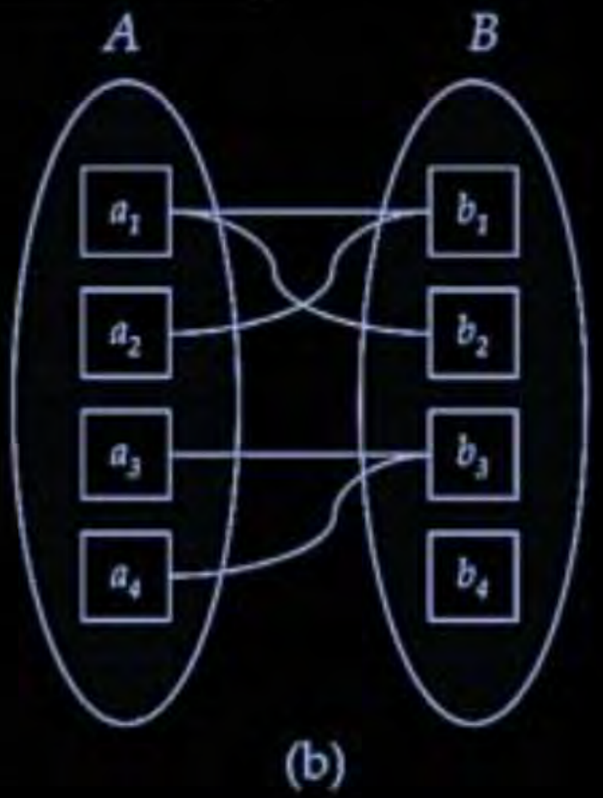
Note: Some elements in A and B may not be mapped to any elements in the other set

Mapping Cardinalities



(a)

Many to one



(b)

Many to many

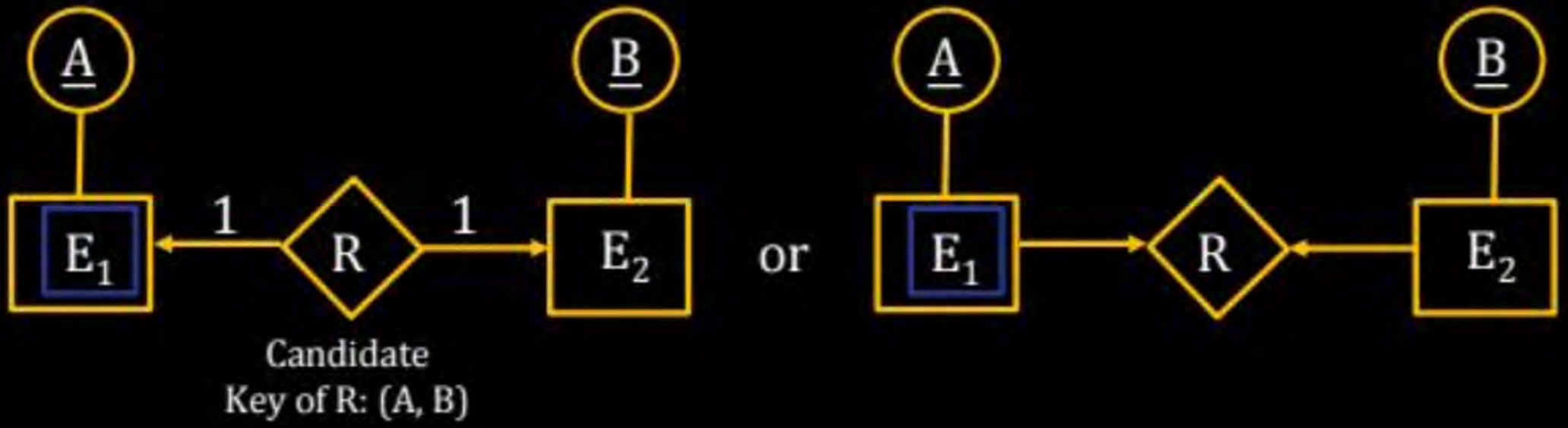
Note: Some elements in A and B may not be mapped to any elements in the other set

Mapping [Cardinality constraints of relationship set]

One mapping : At most one (0 or 1)

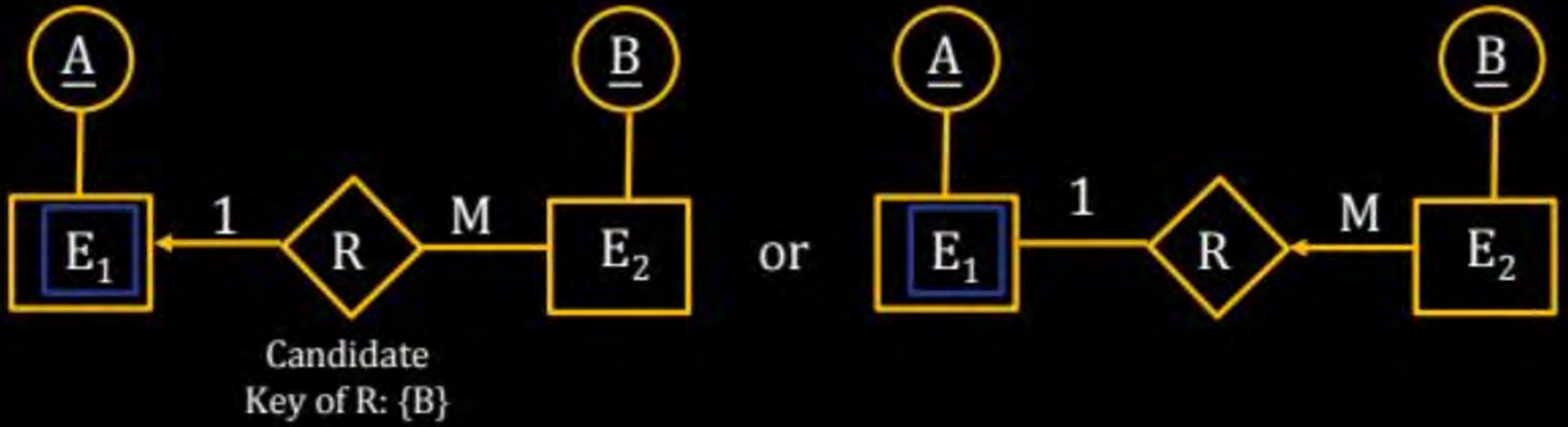
Many mapping : 0 or more (0 ..... *)

Binary Relationship Mapping (One : One)



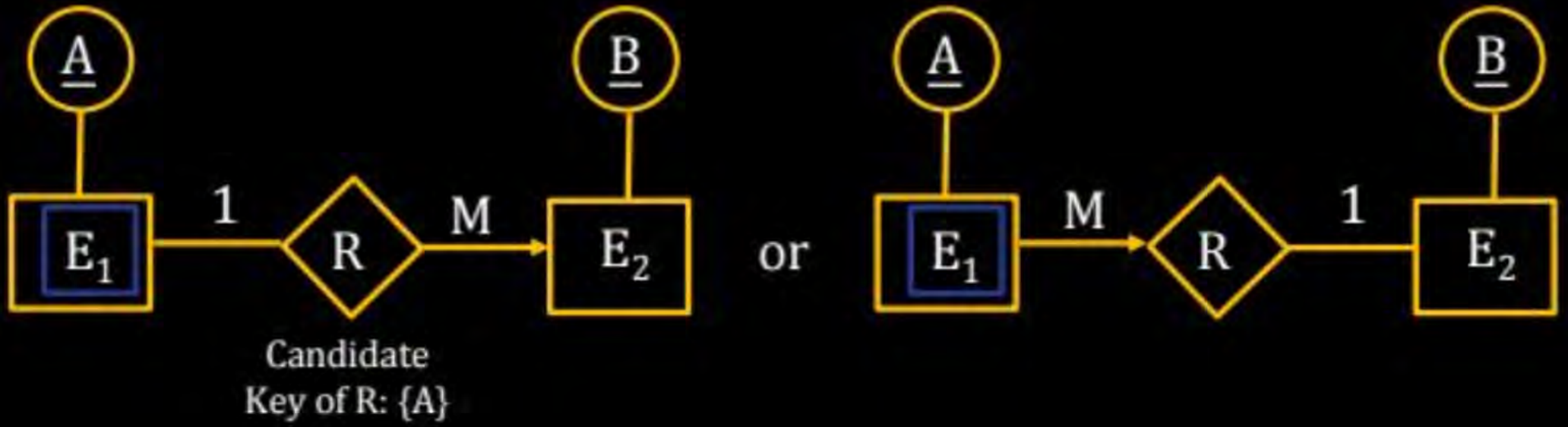
Mapping [Cardinality constraints of relationship set]

Binary Relationship Mapping (One : Many)



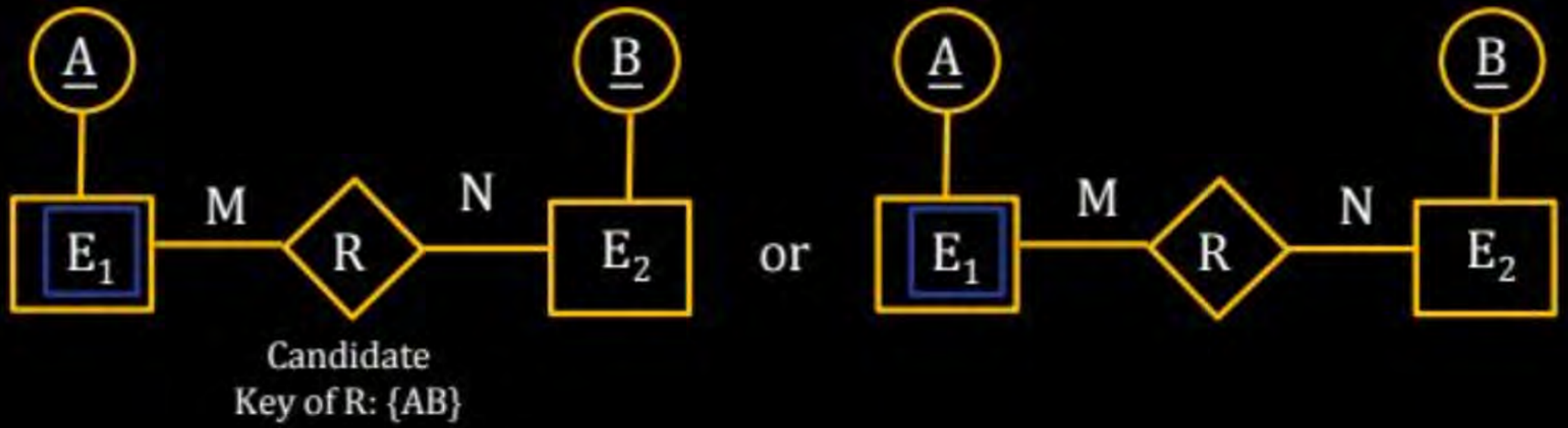
Mapping [Cardinality constraints of relationship set]

Binary Relationship Mapping (Many to One)



Mapping [Cardinality constraints of relationship set]

Binary Relationship Mapping (Many to Many)



Any Doubt ?

THANK YOU!