# ER Model

## Entity-Relationship Model

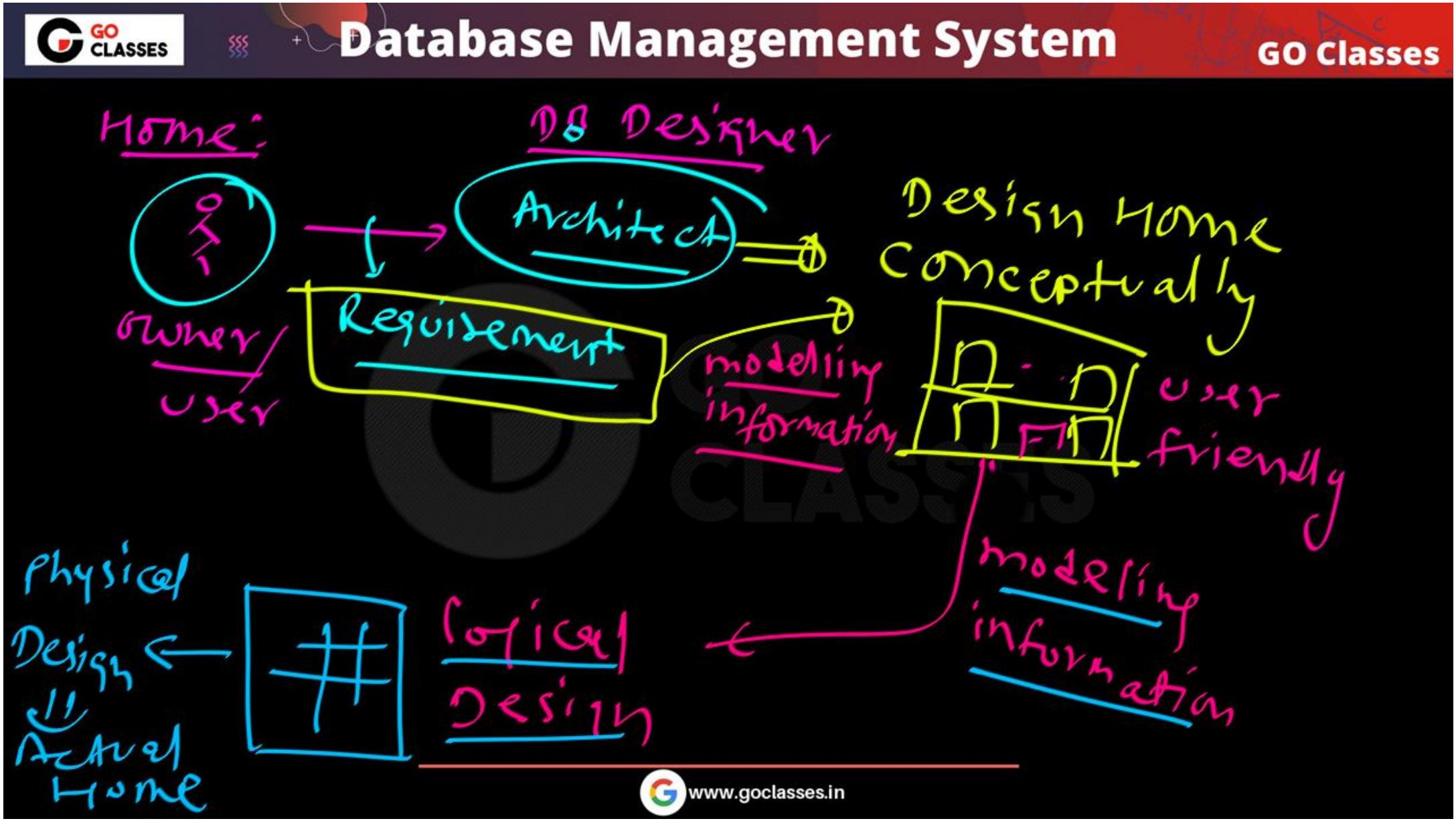
# Database Design Process


Interaction with user.

English

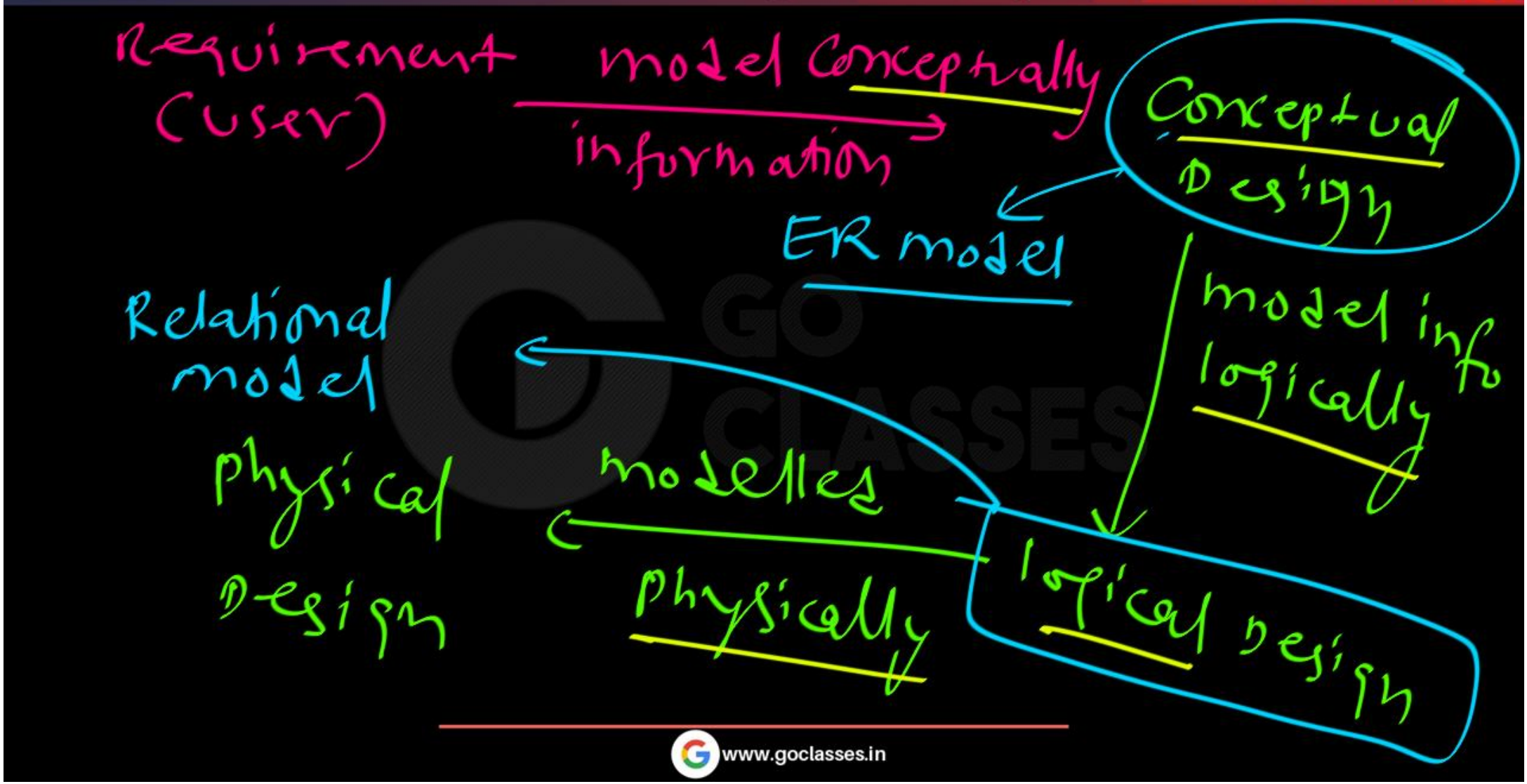
ER model → Relational model

Convert


Home:
DB Designer
Architect
Owner / User
Requirement
modeling information
Design Home Conceptually
user friendly
modeling information
Logical Design
Physical Design
||
Actual Home

# Database Management System





Relational model : proposes by Codd (1971)

ER model : proposes by Chen (1976)

DBMS: a Software, owned by IBM/ORACLE/Amazon

Database: a Collection of Related Data

owned by User/Company

DBMS; Providers: IBM, ORACLE, Amazon

DBMS s/w based on some (implementation) model.

most popular model on which most DBMS s/w today are based on :

Relational Model (1971)

Relation Based model

Relation / Table

Before 1971 → IBM

DBMS SW were Based on some old models

(1) Hierarchical model (Tree Based)

(2) Network model (Graph based)

After Relational model came :

ORACLE (1979) DBMS s/w

↳ Bases on Relational model

Now a days :

most of the DBMS s/w : Use Relational model

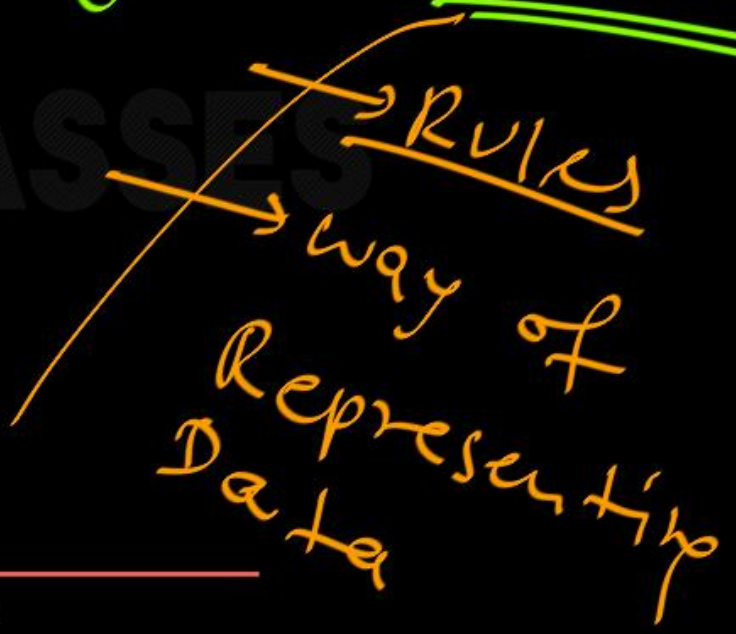
Based on

Some other DBMS s/w are based on Object oriented model etc.

No DBMS s/w is based on ER-model. So, ER model must be converted to the implementation model (on which your DBMS s/w is based)

→ Relational model.

Don't Confuse ER model with Relational model ; They are different.



# Conceptual and Logical Design

Entity, Relationship

Conceptual Model:

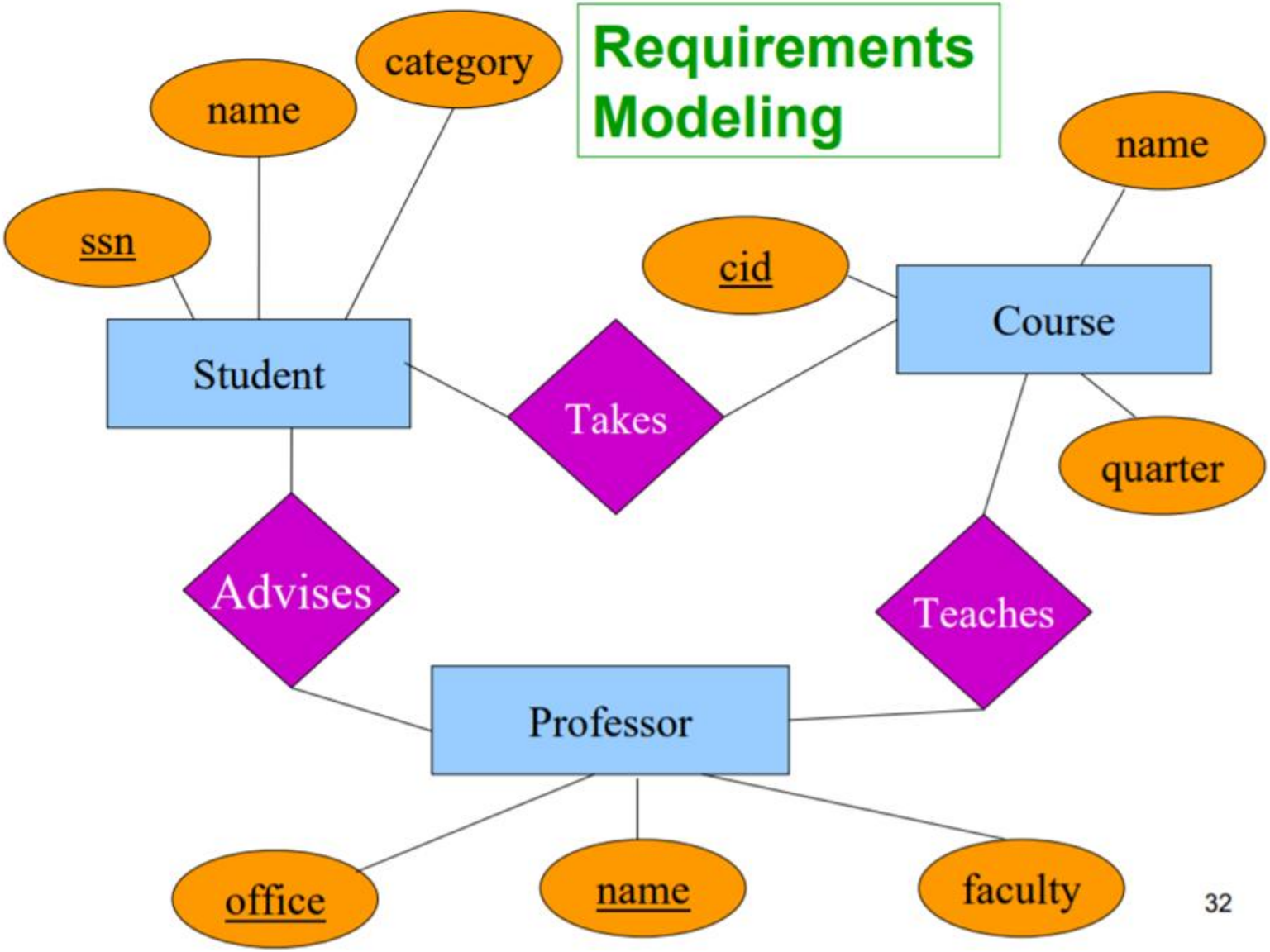
(ER model)


Relation

Relational Model:
(plus functional dependencies)

Normalization:

GO CLASSES






GO Classes

ER model

# Relational Model

- Tables:

Student:

| SSN | Name | Category |
|---|---|---|
| 123-45-6789 | Charles | undergrad |
| 234-56-7890 | Dan | grad |
| | ... | ... |

Takes:

| SSN | CID |
|---|---|
| 123-45-6789 | CSE444 |
| 123-45-6789 | CSE444 |
| 234-56-7890 | CSE142 |
| | ... |

Course:

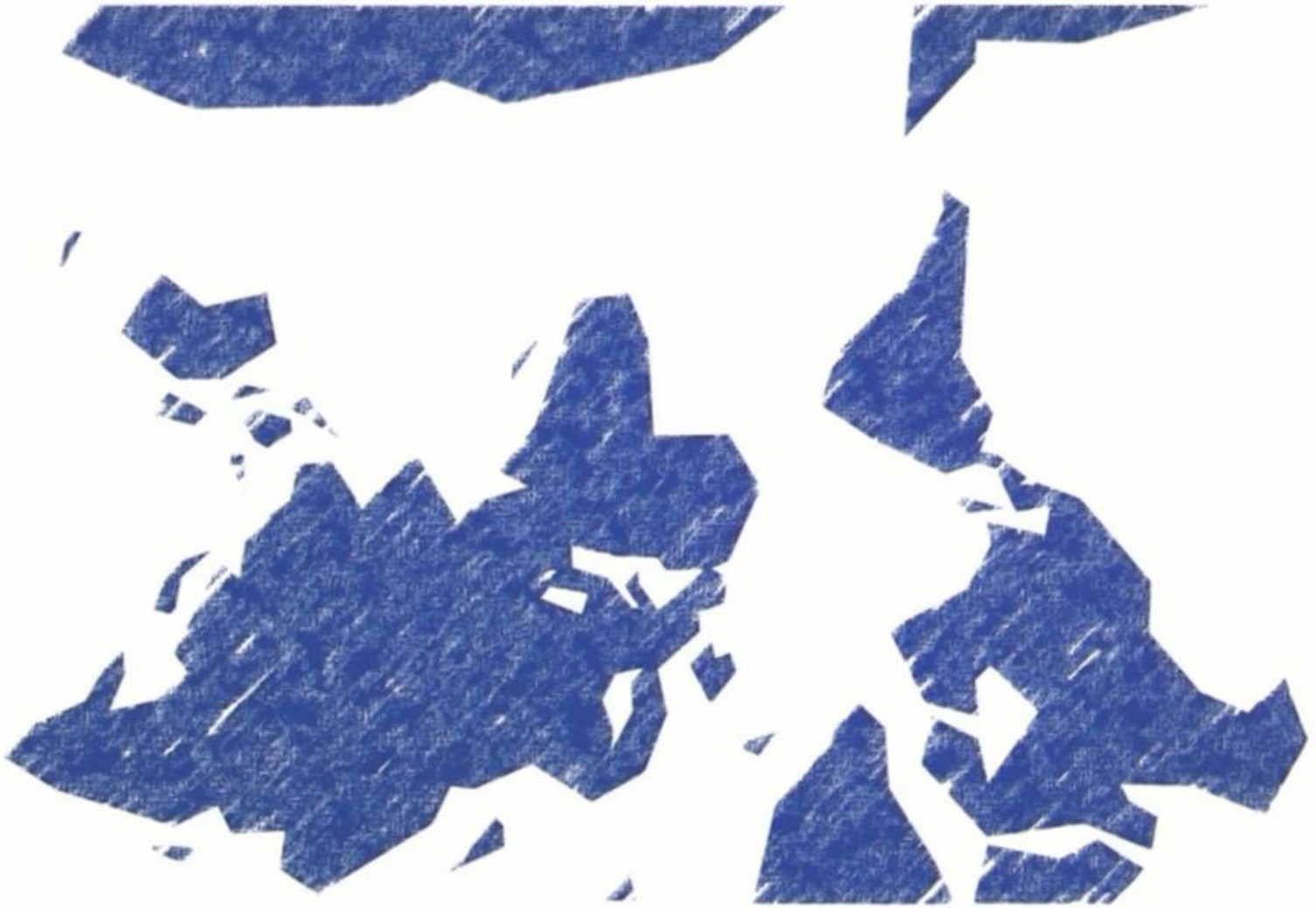
| CID | Name | Quarter |
|---|---|---|
| CSE444 | Databases | fall |
| CSE541 | Operating systems | winter |

# Why Use Models?

Some Observations:

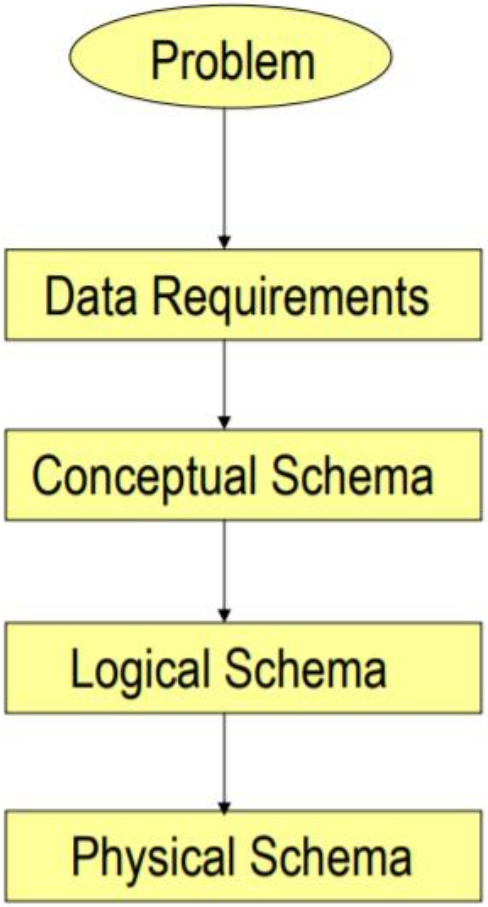
- models can be useful when we want to examine or manage part of the real world
- the costs of using a model are often considerably lower than the costs of using or experimenting with the real world itself

A Map is a Model of Reality



A Map is a Model of Reality

# Database Design Goes Through Stages




*Requirements Analysis*

*Data Analysis, Conceptual Design*

*Logical Database Design*

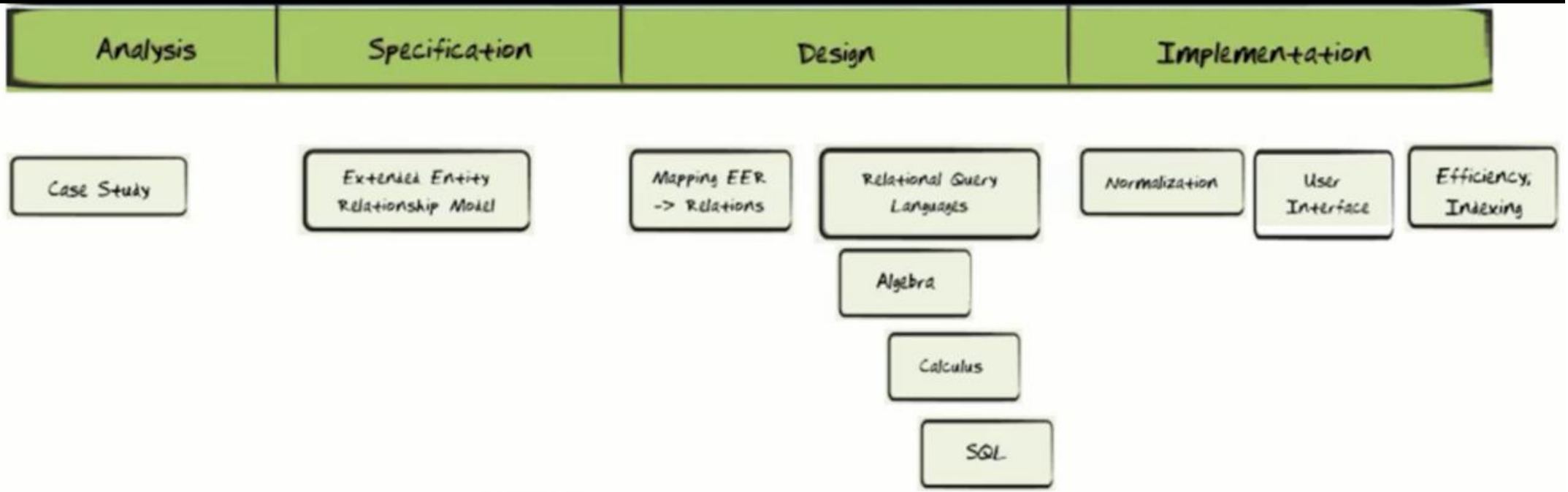
*Physical Database Design*

# Building an Application with a DBMS

- Requirements gathering (natural language, pictures)
- Requirements modeling (conceptual data model, ER)
  - Decide what *entities* should be part of the application and how they should be *related*
- Schema design and implementation
  - Decide on a set of *tables*, *attributes*
  - Create the tables in the database system
  - Populate database (insert records/tuples)

# Database Management System





- Most common model (currently) is the relational model
  - Most database applications use relational databases

# ENTITY-RELATIONSHIP MODEL

Entity–relationship modeling was developed for database and design by Peter Chen and published in a 1976 paper.

Who introduced first relational model?

E. F. Codd

The term "relational database" was invented by **E. F. Codd** at IBM in 1970. Codd introduced the term in his research paper "A Relational Model of Data for Large Shared Data Banks". In this paper and later papers, he defined what he meant by "relational".

# ER model Vs Relational model

(1) In Rel. model :

Database → Relation → set of tuples

Relation → Data structure of Relational model

Database → models/ Represented as

Relation → visualized as a Table

set → No two tuple Can be same.

But in ER model, two entities may be same (in case of weak entities)

(2) In Rel. model: Attribute Domain = Set of Atomic values.

ER model : many different type of attributes.

(3) for Relational model

DB as Relation

Codd:

Relational model

1. Data structure: Relation
2. Data manipulation languages
   (to access information)
   - Relational Algebra
   - Calculus

In ER model → No DBMS S/W even Designed bases on ER model.

- Data Structure: Entity set, Relationship set
- language: No agreed upon language to access Data

# ENTITY-RELATIONSHIP MODEL

The great successful men of the world have used their imaginations. They think ahead and create their mental picture, and then go to work materializing that picture in all its details, filling in here, adding a little there, altering this bit and that bit, but steadily building, steadily building.

—Robert Collier

The entity-relationship (ER) data model allows us to describe the data involved in a real-world enterprise in terms of objects and their relationships and is widely used to develop an initial database design. In this chapter, we introduce the ER model and discuss how its features allow us to model a wide range of data faithfully.

The ER model is important primarily for its role in database design. It provides useful concepts that allow us to move from an informal description of what users want from their database to a more detailed, and precise, description that can be implemented in a DBMS.

Within the larger context of the overall design process, the ER model is used in a phase called conceptual database design.

## OVERVIEW OF DATABASE DESIGN :

(1) Requirements Analysis: The very first step in designing a database application is to understand what data is to be stored in the database, what applications must be built on top of it, and what operations are most frequent and subject to performance requirements. In other words, we must find out what the users want from the database.

This is usually an informal process that involves discussions with user groups/Database owners.

(2) Conceptual Database Design: The information gathered in the requirements analysis step is used to develop a high-level description of the data to be stored in the database, along with the constraints that are known to hold over this data. This step is often carried out using the ER model, or a similar high-level data model, and is discussed in the rest of this chapter.

(3) Logical Database Design: We must choose a DBMS to implement our database design, and convert the conceptual database design into a database schema in the data model of the chosen DBMS. We will only consider relational DBMSs, and therefore, the task in the logical design step is to convert an ER schema into a relational database schema.

ER model

Entity : Distinguishable Object

Relationship : Association b/w entities.

Eg: Some particular student

" "

" Course

# The Entity-Relationship Model

- A *very* common model for schema design
- Allows for specification of complex schemas in graphical form
- Basic concepts are simple, but can also represent sophisticated concepts
  - e.g. type hierarchies
- Can be mapped to the relational model!
  - Simplifies implementation phase
  - Mapping process can be automated by design tools
- Also written as “E-R model”

# Entities and Entity-Sets

- An entity is any "thing" that can be uniquely represented
  - e.g. a product, an employee, a software defect
  - Each entity has a set of attributes
  - Entities can be uniquely identified by some set of attributes (but don't have to be)
- An entity-set is a named collection of entities of the same type, with the same attributes

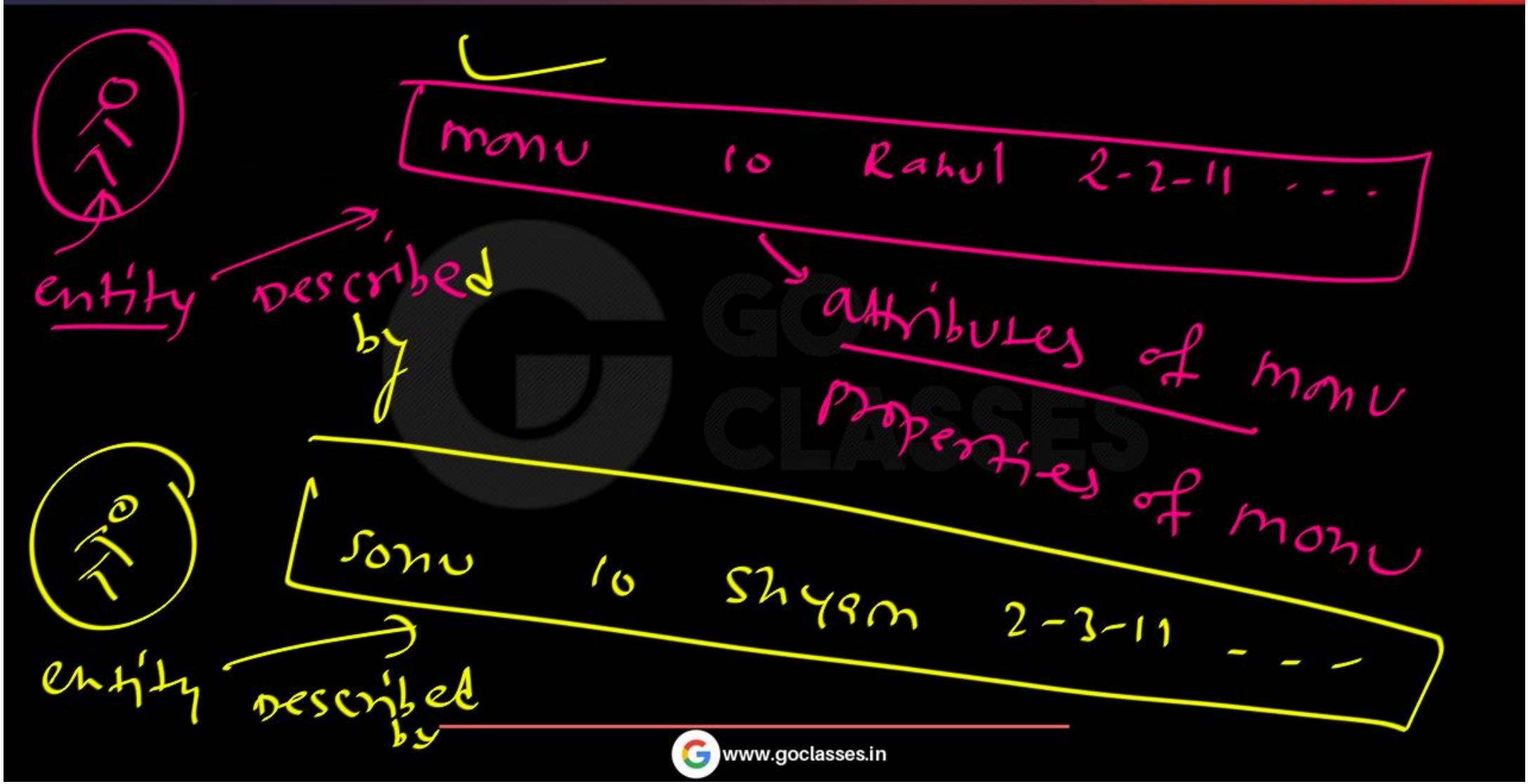
# Entities and Entity-Sets (2)

- An entity has a set of attributes
  - Each attribute has a name and domain
  - Each attribute also has a corresponding value
- Entity-sets also specify a set of attributes
  - Every entity in the entity-set has the same set of attributes
  - Every entity in the entity-set has its own value for each attribute

Entity : Distinguishable object (thing)

↳ has some properties : Attributes

∴ Entities are described by attributes.

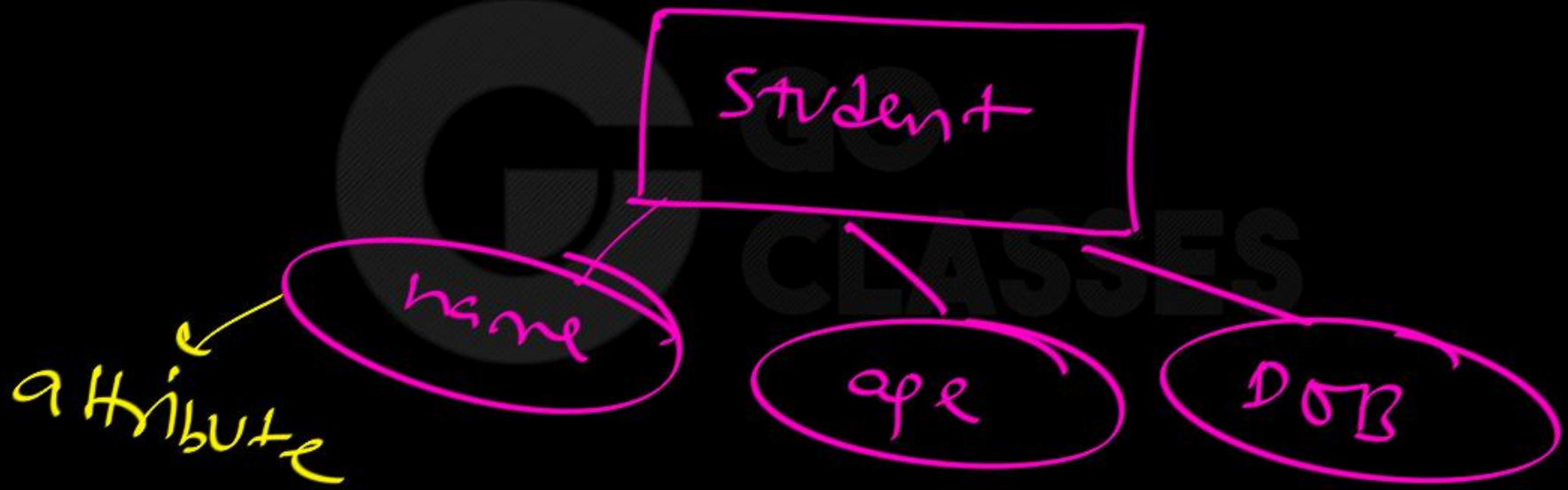

R1
entity
described by
manu 10 Rahul 2-2-11 ...
attributes of manu
properties of manu
R2
entity
described by
sonu 10 Shyam 2-3-11 ---

Entity set : Collection of similar entities.

↓

having same set of attributes.

Entity set :⟶ Represented as a Rectangle



In ER model:

attribute : Property of entity

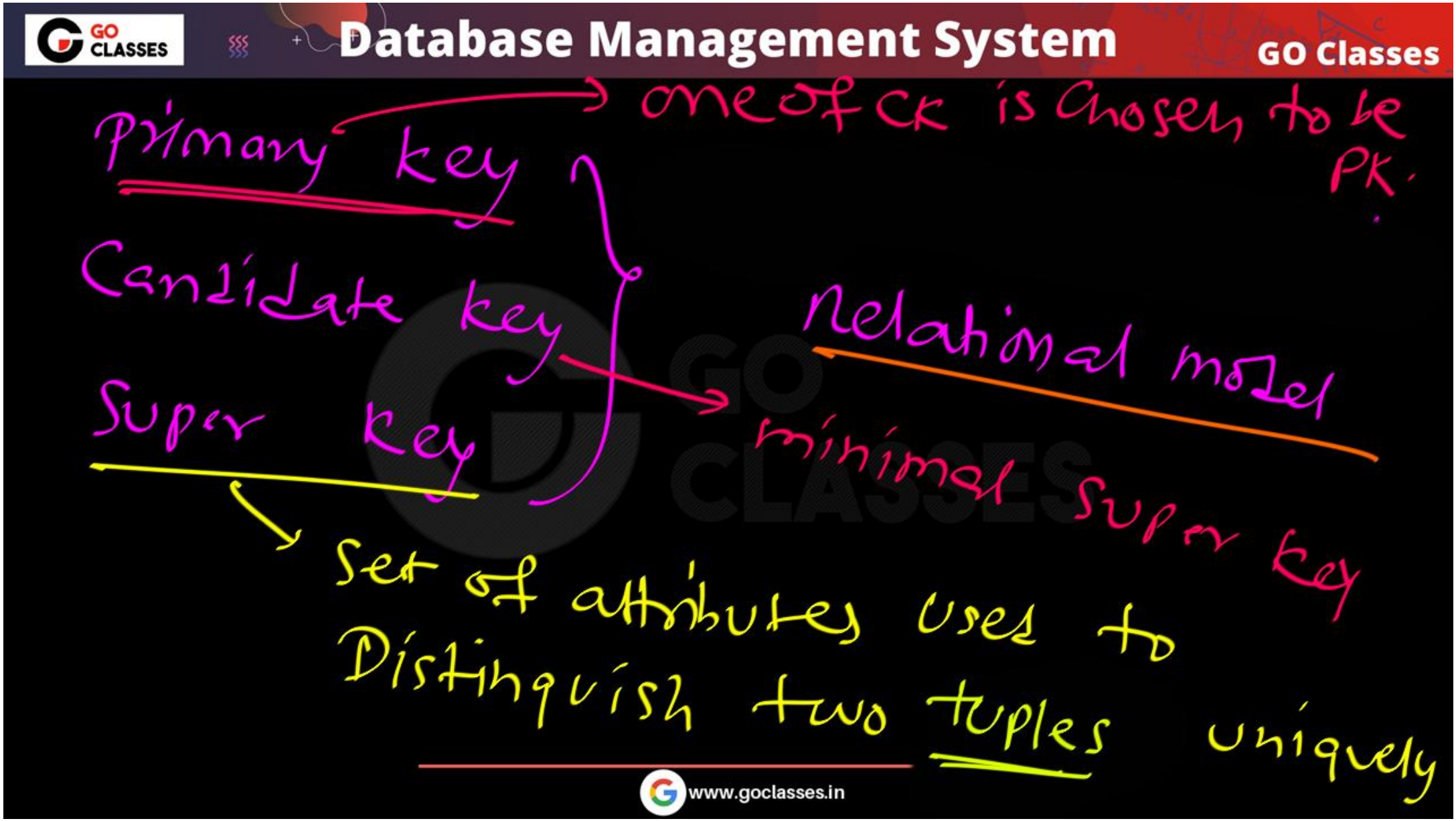
Domain of attribute : set of values that a attribute can take.

Primary key → one of CK is Chosen to be PK.

Candidate key → same in ER model, Relational model

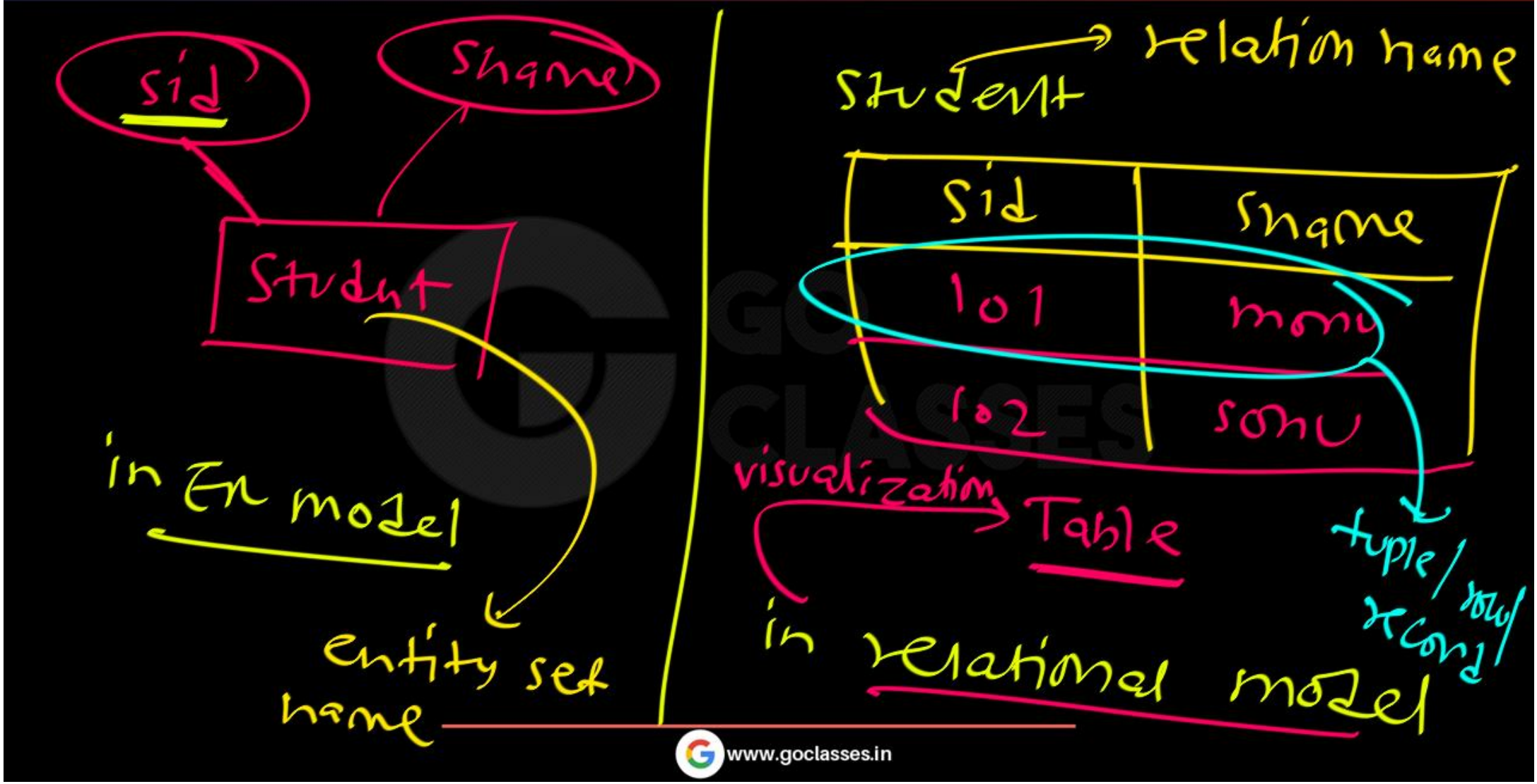
Super key → minimal super key

Set of attributes uses to Distinguish two entities uniquely

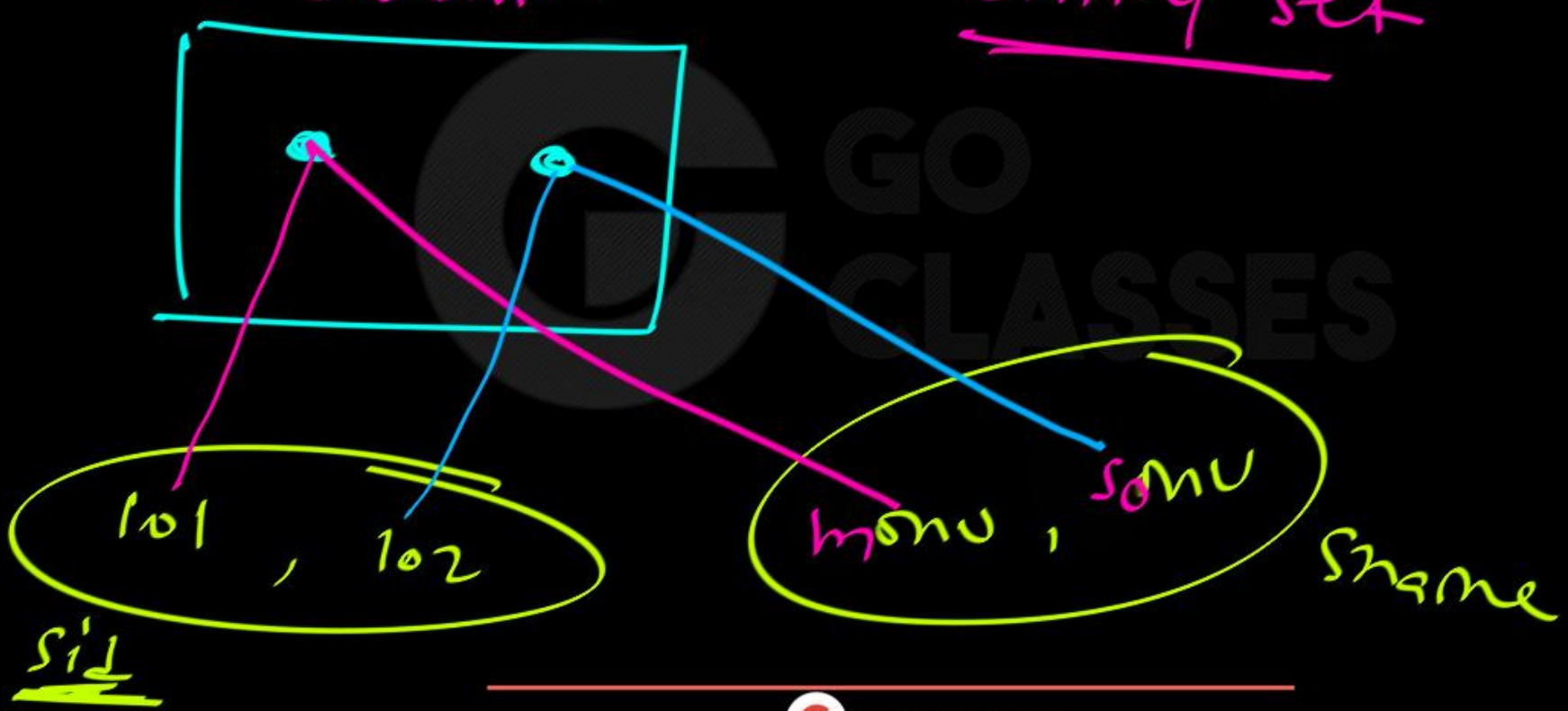

Primary key
one of CK is Chosen to be PK.
Candidate key
Super key
Relational model
minimal super key
Set of attributes Uses to
Distinguish two tuples uniquely

# Database Management System





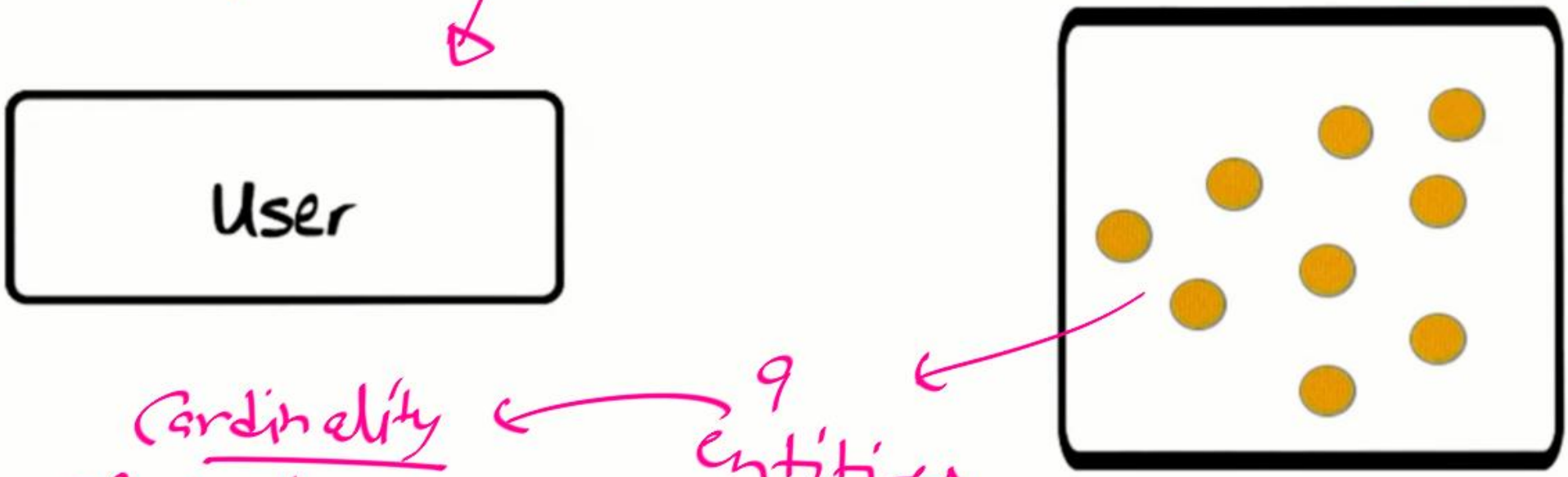
# Visualization in ER model:



# Entity Type and entity [redacted]

Entity set

Particular entity




Cardinality of entity type User ← 9 entities

- entity type names must be unique

An entity is an object in the real world that is distinguishable from other objects.

It is often useful to identify a collection of similar entities. Such a collection is called an entity set.

An entity is described using a set of attributes. All entities in a given entity set have the same attributes; this is essentially what we mean by similar.

For each attribute associated with an entity set, we must identify a domain of possible values. For example, the domain associated with the attribute name of Employees might be the set of 20-character strings.

As another example, if the company rates employees on a scale of 1 to 10 and stores ratings in a field called rating, the associated domain consists of integers 1 through 10. Further, for each entity set, we choose a key.

A key is a minimal set of attributes whose values uniquely identify an entity in the set. There could be more than one candidate key; if so, we designate one of them as the primary key. For now we will assume that each entity set contains at least one set of attributes that uniquely identifies an entity in the entity set; that is, the set of attributes contains a key.

The Employees entity set with attributes ssn, name, and lot is shown in Figure 2.1.

An entity set is represented by a rectangle, and an attribute is represented by an oval.

Each attribute in the primary key is underlined.

The domain information could be listed along with the attribute name, but we omit this to keep the figures compact.

The key is ssn.

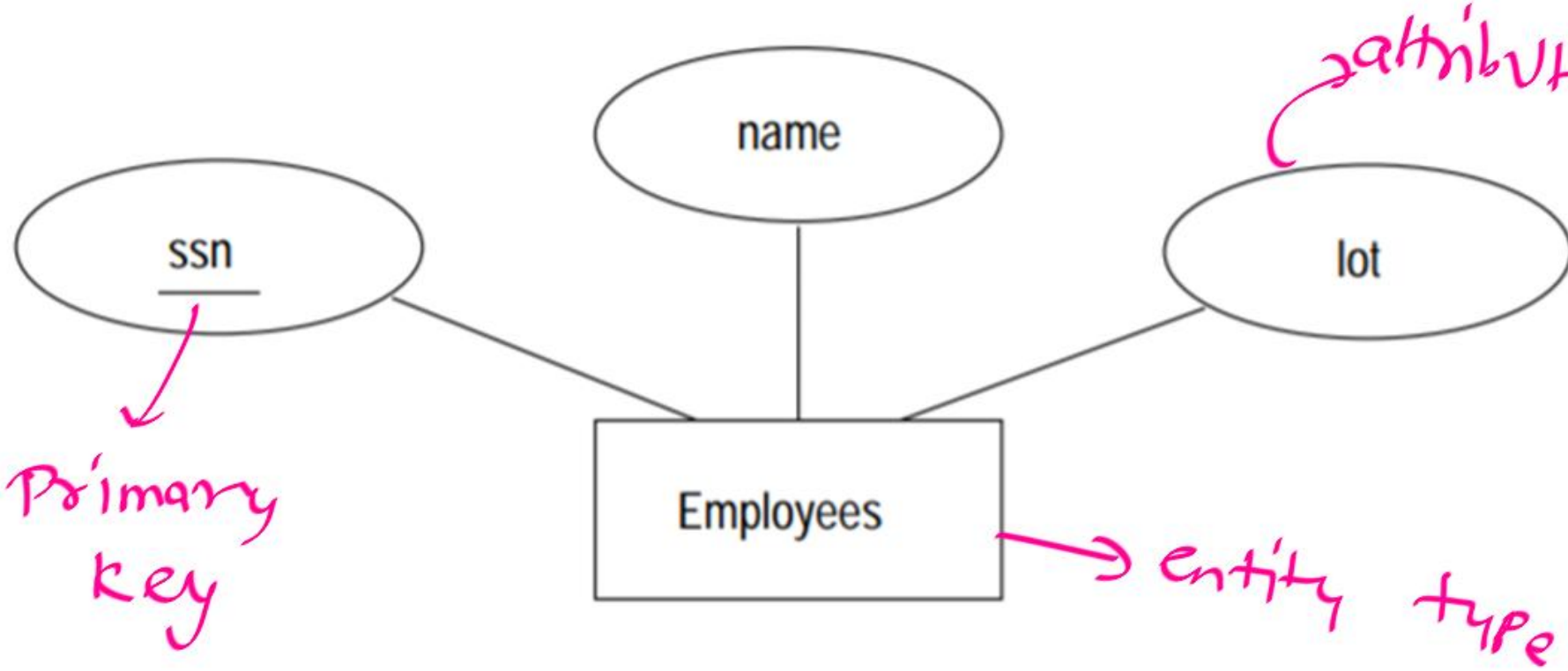
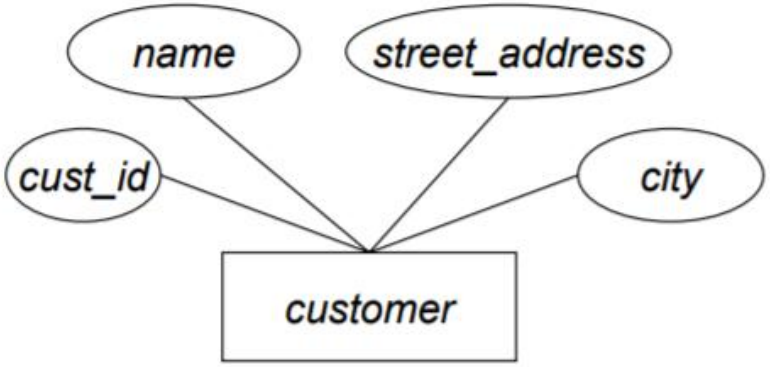



**Figure 2.1** The Employees Entity Set

# Diagramming an Entity-Set

Example: a *customer* entity-set

- Attributes:
  - *cust_id*
  - *name*
  - *street_address*
  - *city*




- Entity-set is a box
- Attributes are ovals
- Lines connect attributes to entity-set

| ER model | Relational model |
| --- | --- |
| ① Entity | tuple / Row / Record |
| ② Entity type (Entity Set) | Relation / Table |
| ③ PK, SK, CK | PK, SK, CK |

| ER model | Relational model |
| --- | --- |
| ④ attribute | attribute |
| Domain | Domain |
| ⑤ Cardinality of entity type | Cardinality of Relation instance |
| no. of entities | no. of rows. |

# Entity-Set Keys

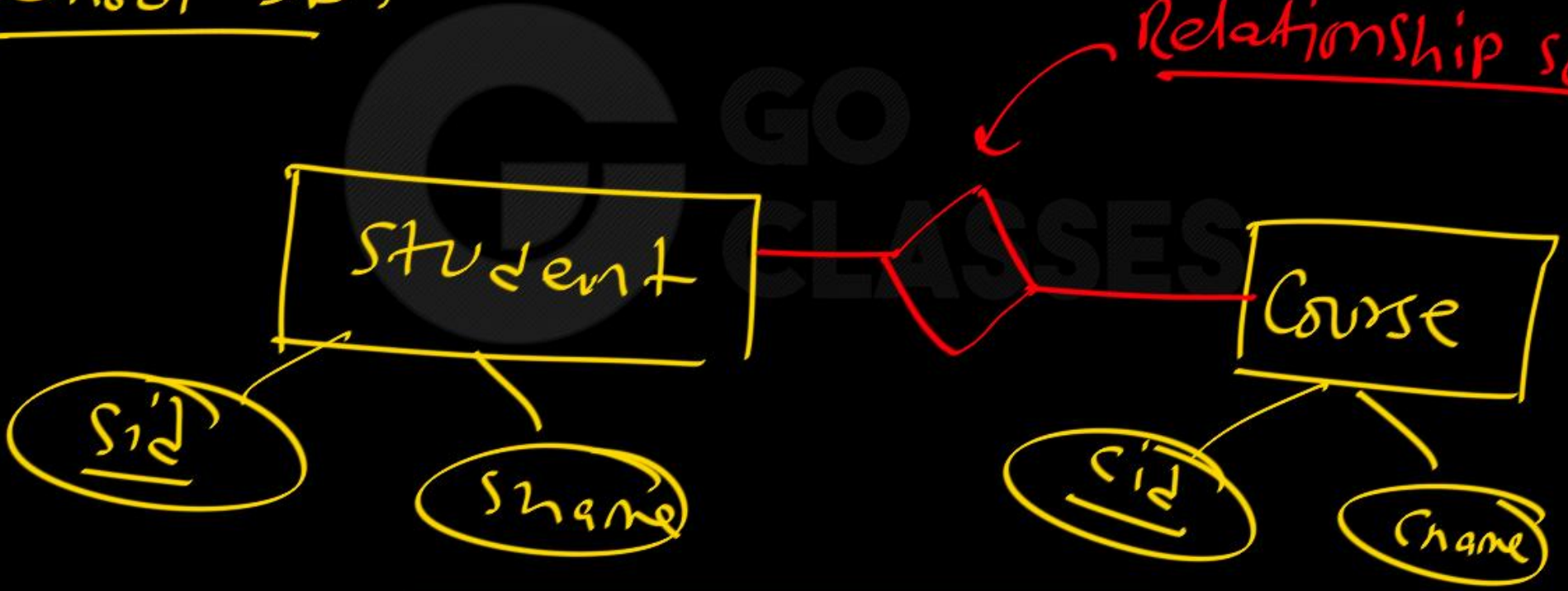
- Entities in an entity-set must be uniquely distinguishable using their values
  - Entity-set: each entity is unique
- E-R model also includes the notion of keys:
  - Superkey: a set of one or more attributes that can uniquely identify an entity
  - Candidate key: a *minimal* superkey
  - Primary key: a candidate key chosen by DB designer as the primary means of identifying entities
- Keys are a property of the entity-set
  - They apply to *all* entities in the entity-set

# Choosing Candidate Keys

- Candidate keys constrain the values of the key-attributes
  - No two entities can have the same values for those attributes
  - Need to ensure that database can actually represent all realistic circumstances
- Simple example: *customer* entity-set
  - Using customer name as a candidate key is bad design: customers can have same name

# Relationship: (Association b/w entities)

School DB:



Visualization of (Association b/w entities)


Visualization of Relationship

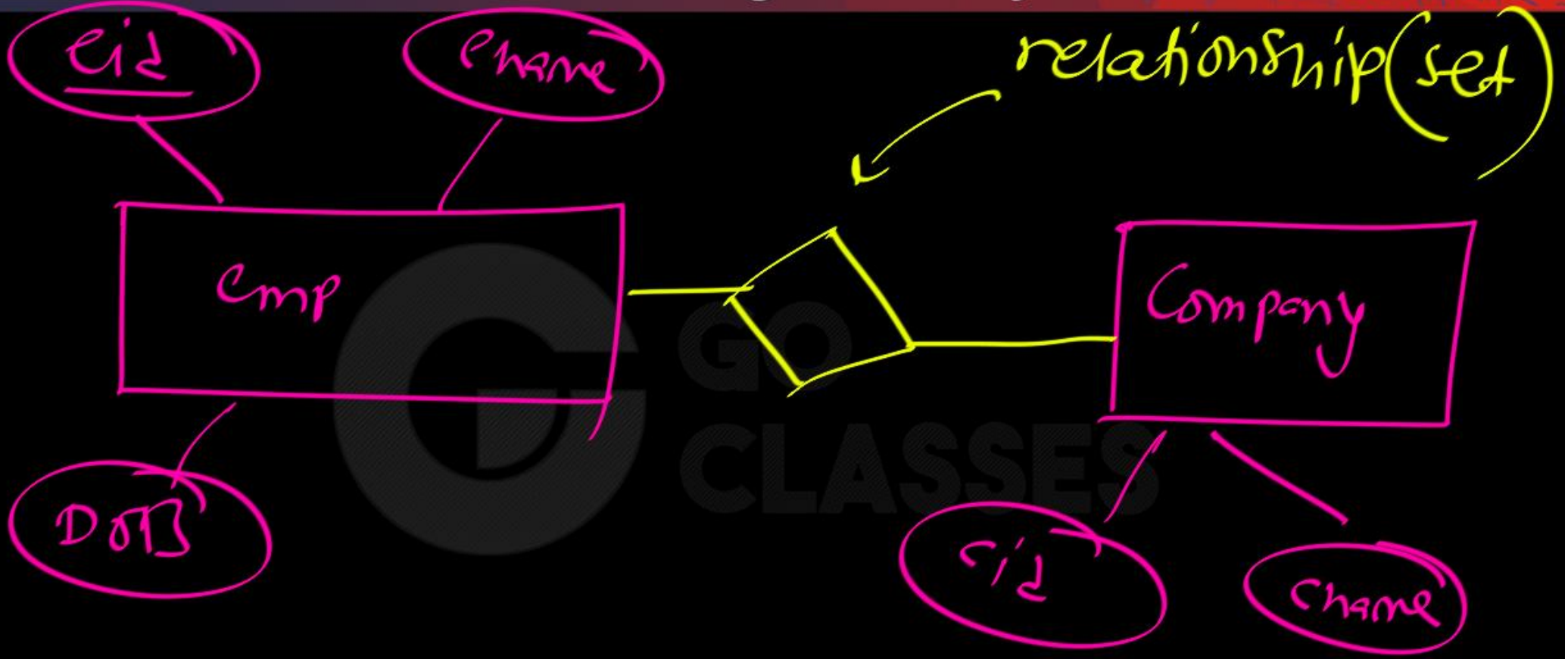
**Relationship:** Association b/w entities

**Relationship set:** set of all similar relationships

b/w Same Entity types.

# Database Management System



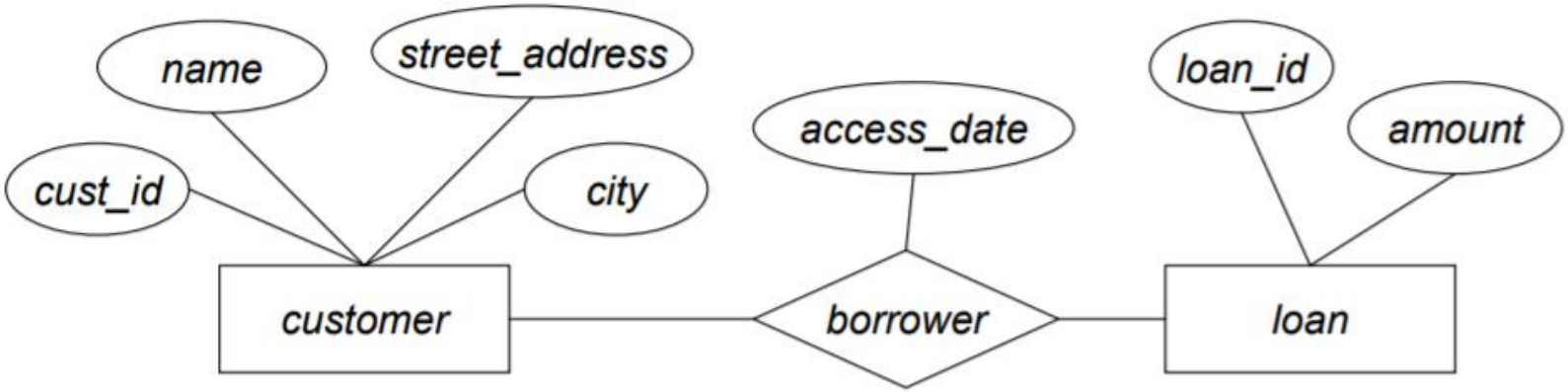


# Relationships

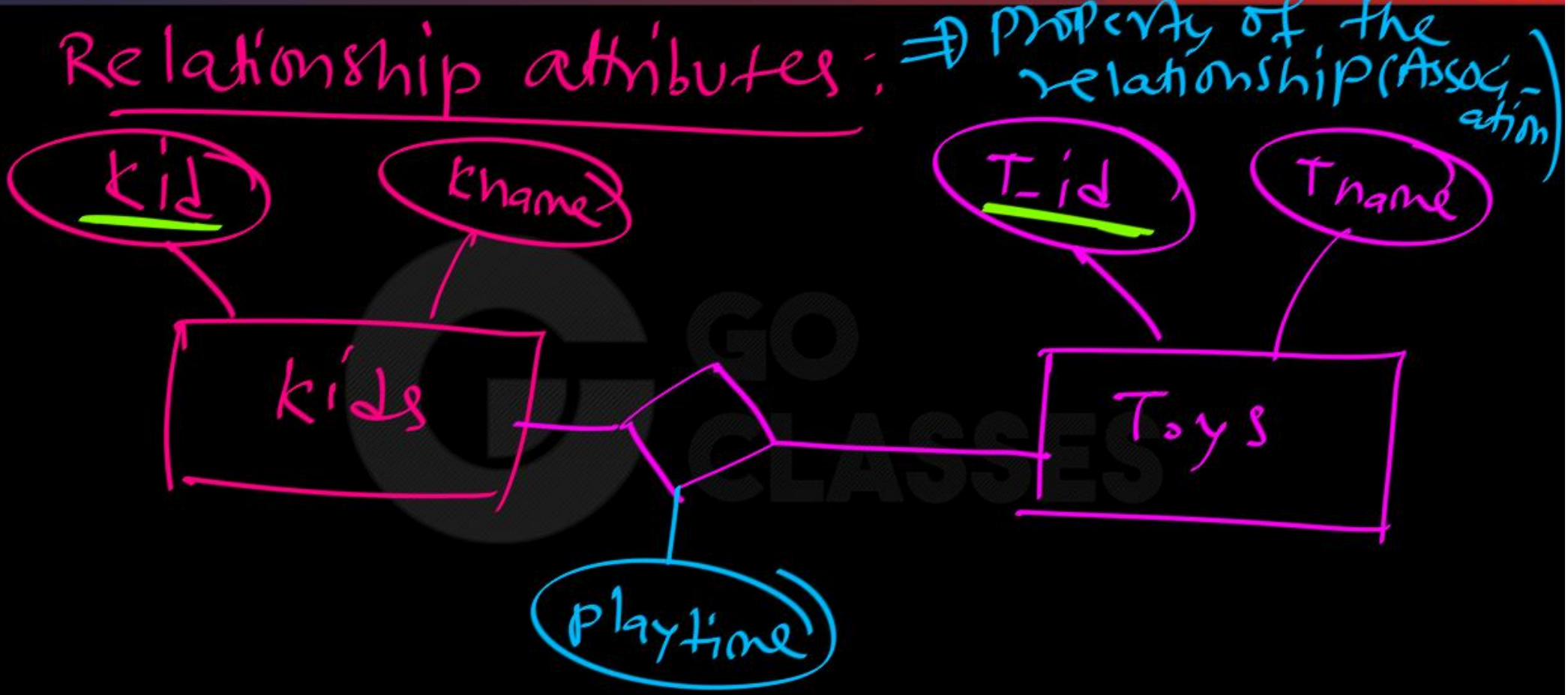
- A relationship is an association between two or more entities
  - e.g. a bank loan, and the customer who owns it
- A relationship-set is a named collection of relationships of the same type
  - i.e. involving the same entities
  - Formally, a relationship-set is a mathematical relation involving $n$ entity-sets, $n \geq 2$
    - $E_1, E_2, \ldots, E_n$ are entity sets
    - A relationship set $R$ is a subset of:
      $$\{ (e_1, e_2, \ldots, e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n \}$$
    - $(e_1, e_2, \ldots, e_n)$ is a relationship

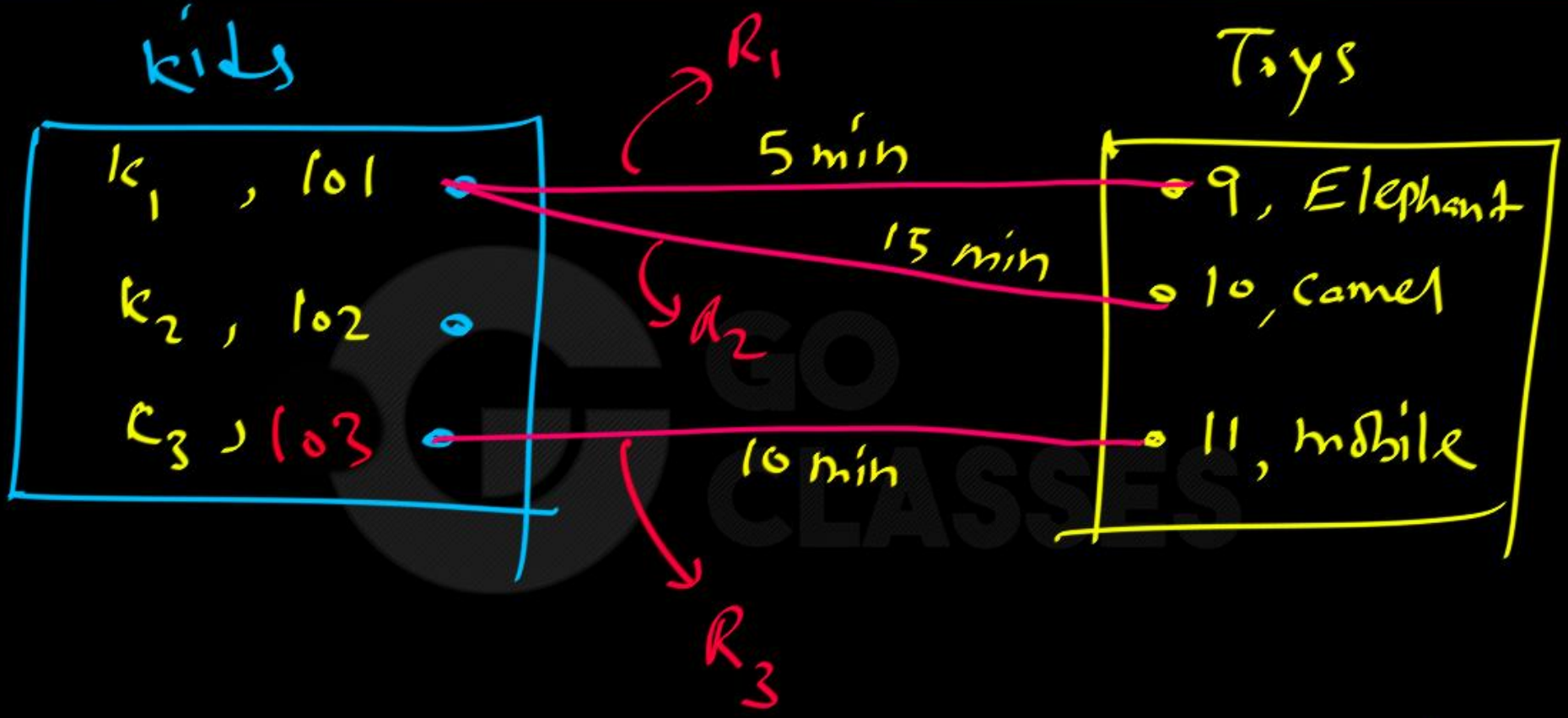
# Diagramming a Relationship-Set

Example: the *borrower* relationship-set between the *customer* and *loan* entity-sets




- Relationship-set is a diamond
- Descriptive attributes are ovals (again)
- Lines connect entities, relationships, etc.


Relationship attributes:
⇒ Property of the relationship (Association)
kid
kname
kids
T_id
Tname
Toys
playtime




Visualization of Relationship with attribute

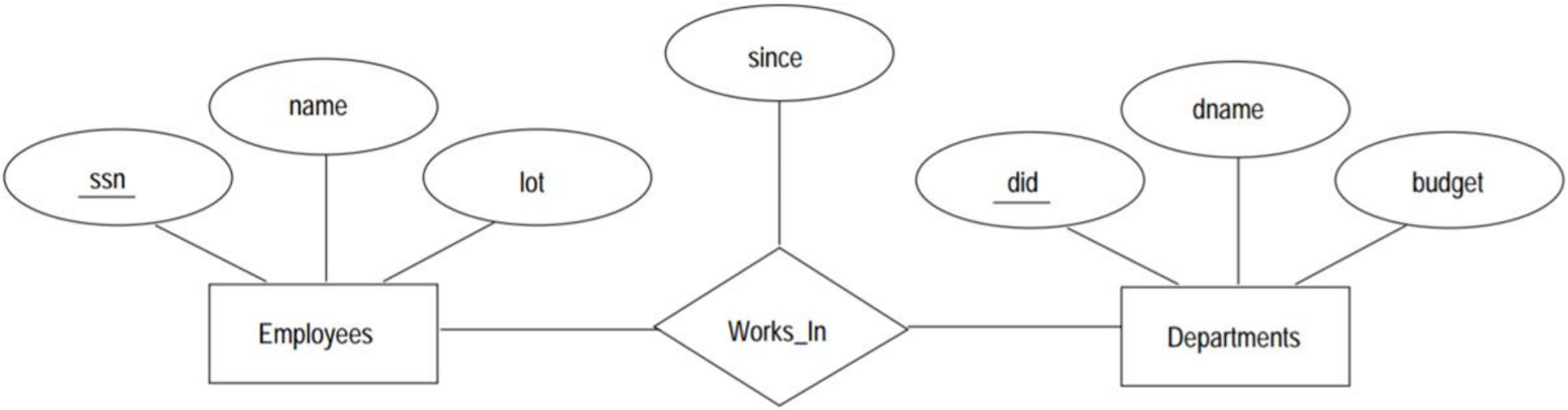
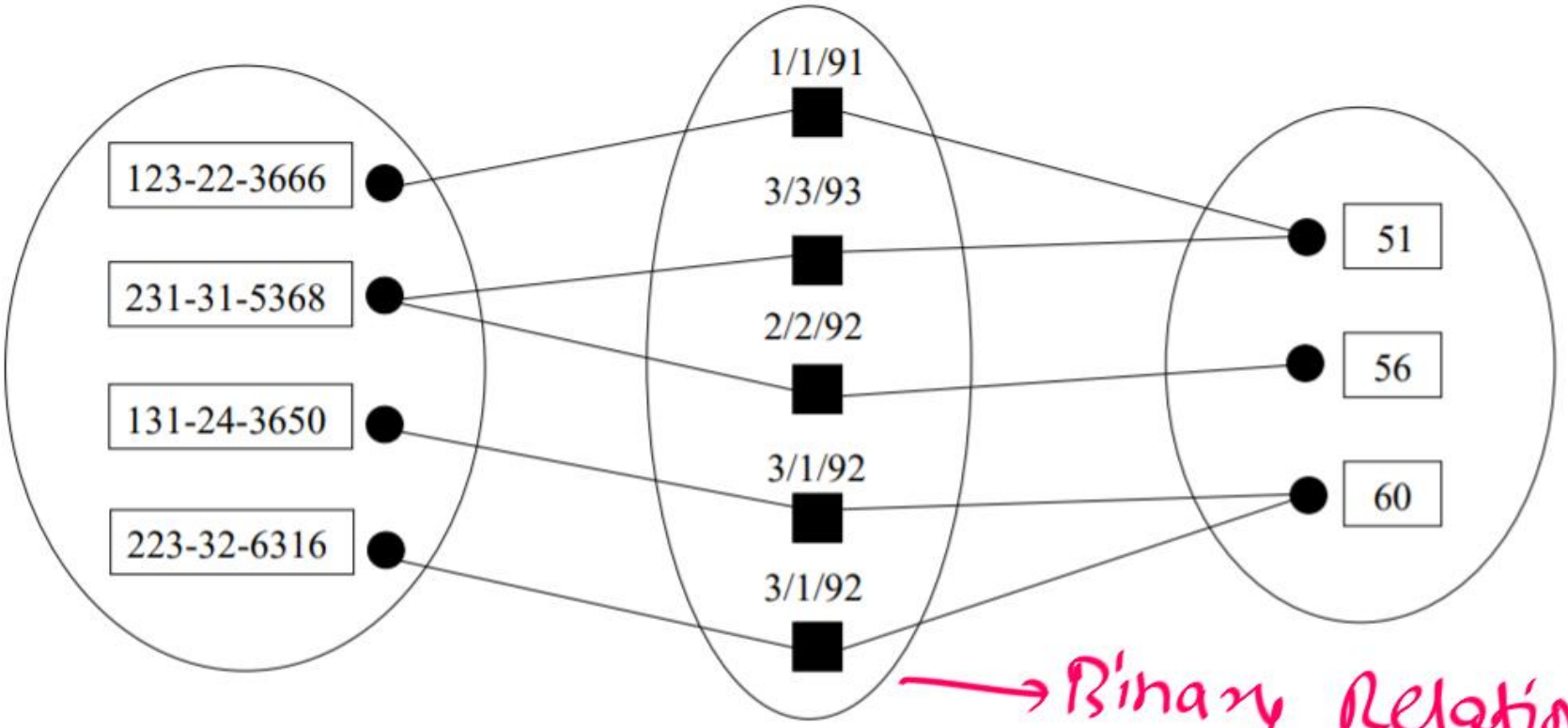



**Figure 2.2** The Works_In Relationship Set




Binary Relationship Set

**Figure 2.3** An Instance of the Works_In Relationship Set

# Relationships (2)

- Entity-sets participate in relationship-sets
  - Specific entities participate in a relationship instance
- Example: bank loans
  - *customer* and *loan* are entity-sets

    ( 555-55-5555, Jackson, Woodside ) is a *customer* entity

    ( L-14, 1500 ) is a *loan* entity
  - *borrower* is a relationship-set
    - *customer* and *loan* participate in *borrower*
    - *borrower* contains a relationship instance that associates customer “Jackson” and loan “L-14”

# Relationships and Attributes

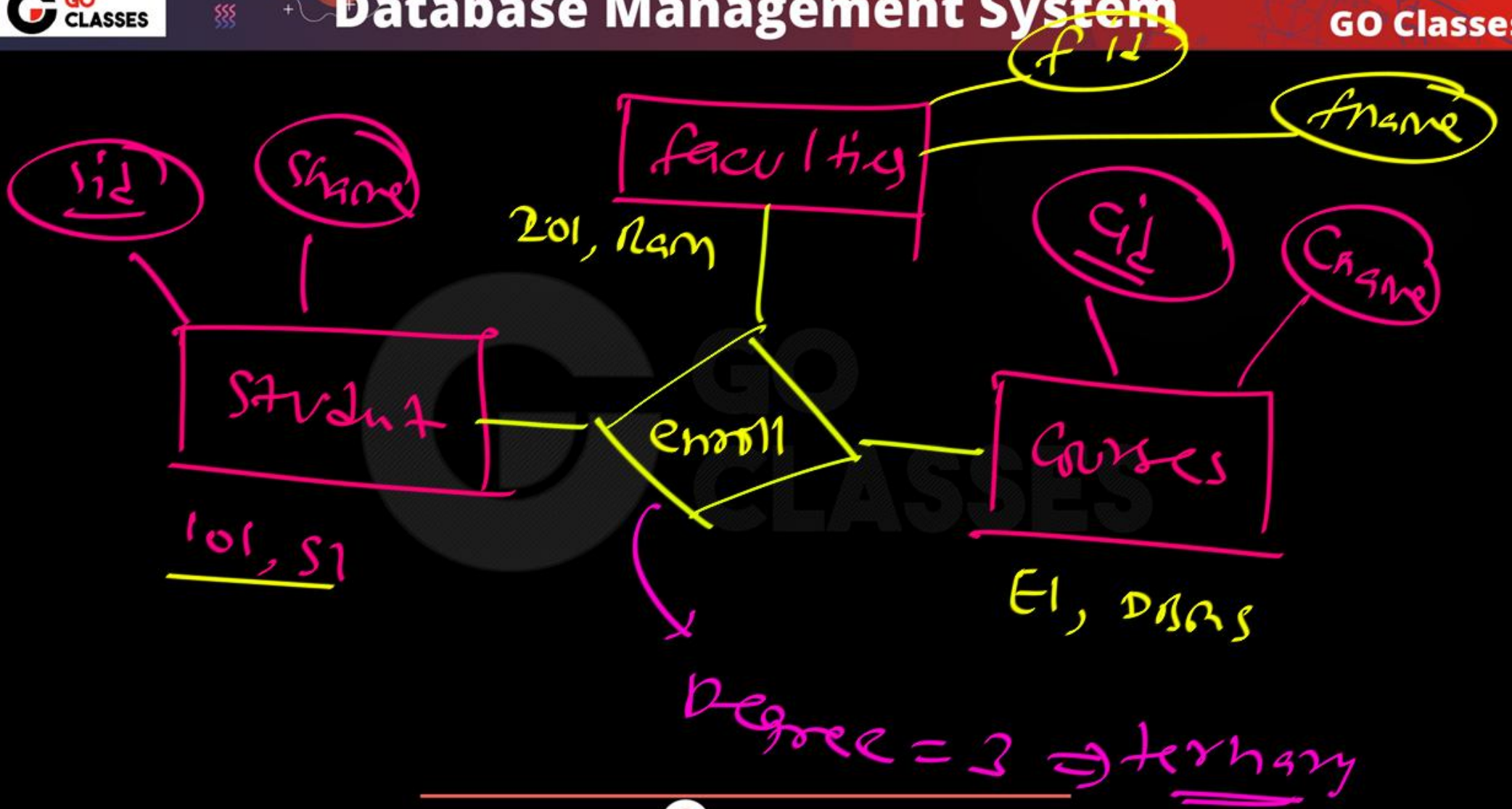
- Relationships can also have attributes!
  - Called descriptive attributes
  - They describe a particular relationship
- Example:
  - The relationship between a software defect and an employee can have a *date_assigned* attribute
- Distinction between entity attributes and relationship attributes is not made by relational model

## Degree of Relationship set :

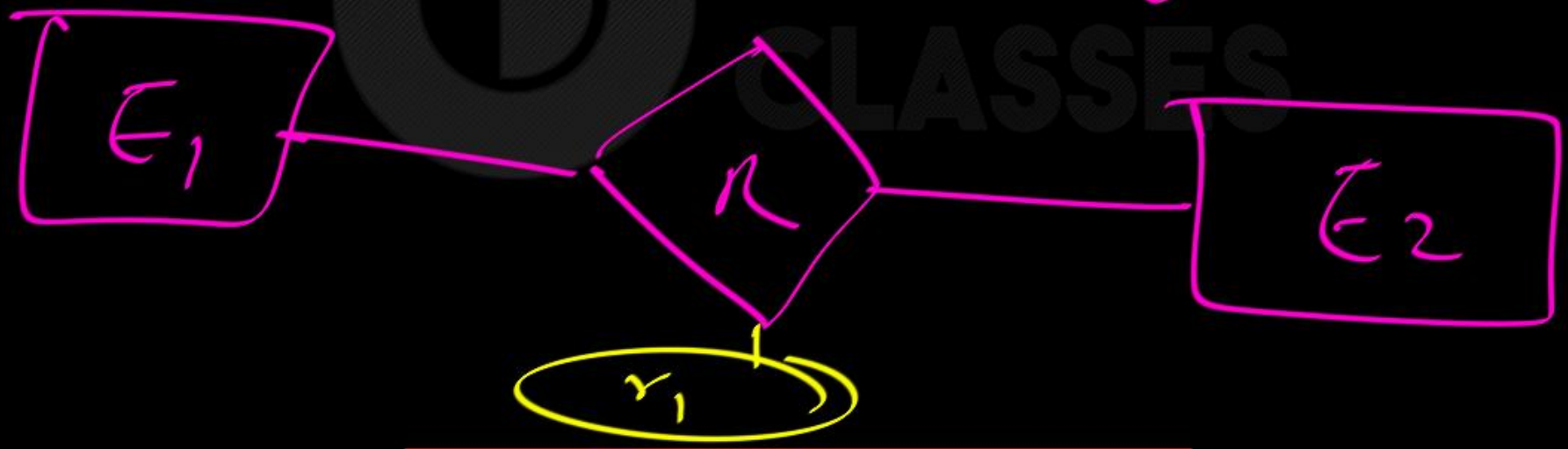
Relationship is in b/w how many entities ; is called Degree of Relationship set.

---

Relationship set is b/w how many entity sets, called Degree of Relationship set


f id
fname
faculty
sid
sname
201, Ram
cid
cname
Student
enroll
Courses
101, S1
E1, DBMS
Degree = 3 ⇒ ternary

Most of the time, we have binary relationships only.

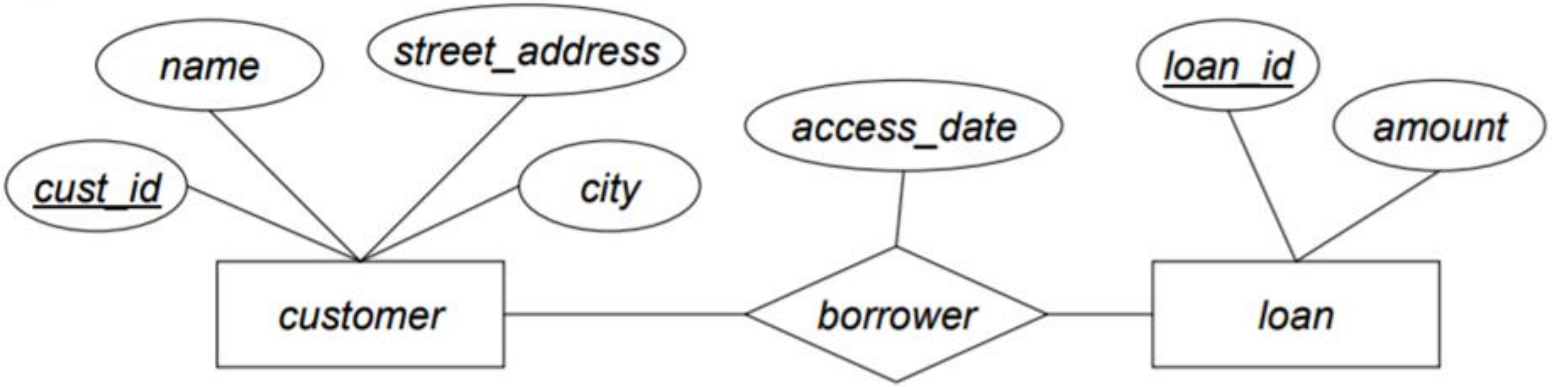


# Degree of Relationship Set

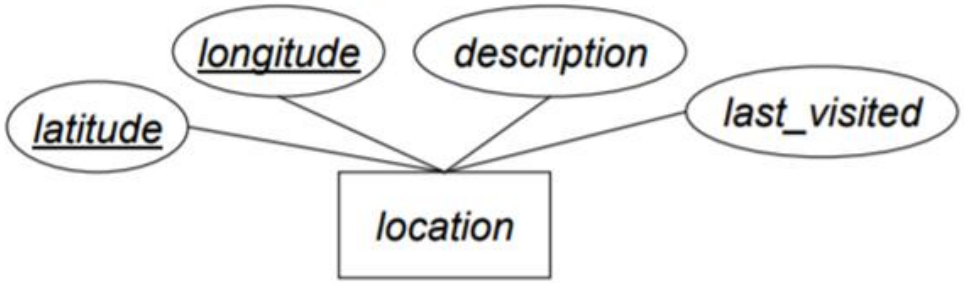
- Most relationships in a schema are binary
  - Two entities are involved in the relationship
- Sometimes there are ternary relationships
  - Three entities are involved
  - Definitely less common
- The number of entity-sets that participate in a relationship-set is called its degree
  - Binary relationship: degree = 2
  - Ternary relationship: degree = 3

# Diagramming Primary Keys

- In an entity-set diagram, all attributes in the primary key have an underlined name




- Another example: a geocache *location* entity-set



- Simple guideline for choosing entities or relationships
  - An entity is a “thing”
  - A relationship is an “action” involving entities

# Type of attributes in ER model:

1. Simple Vs Composite
2. Simple Values Vs multivalued
3. Derived Vs Base/source

# Attribute Structure

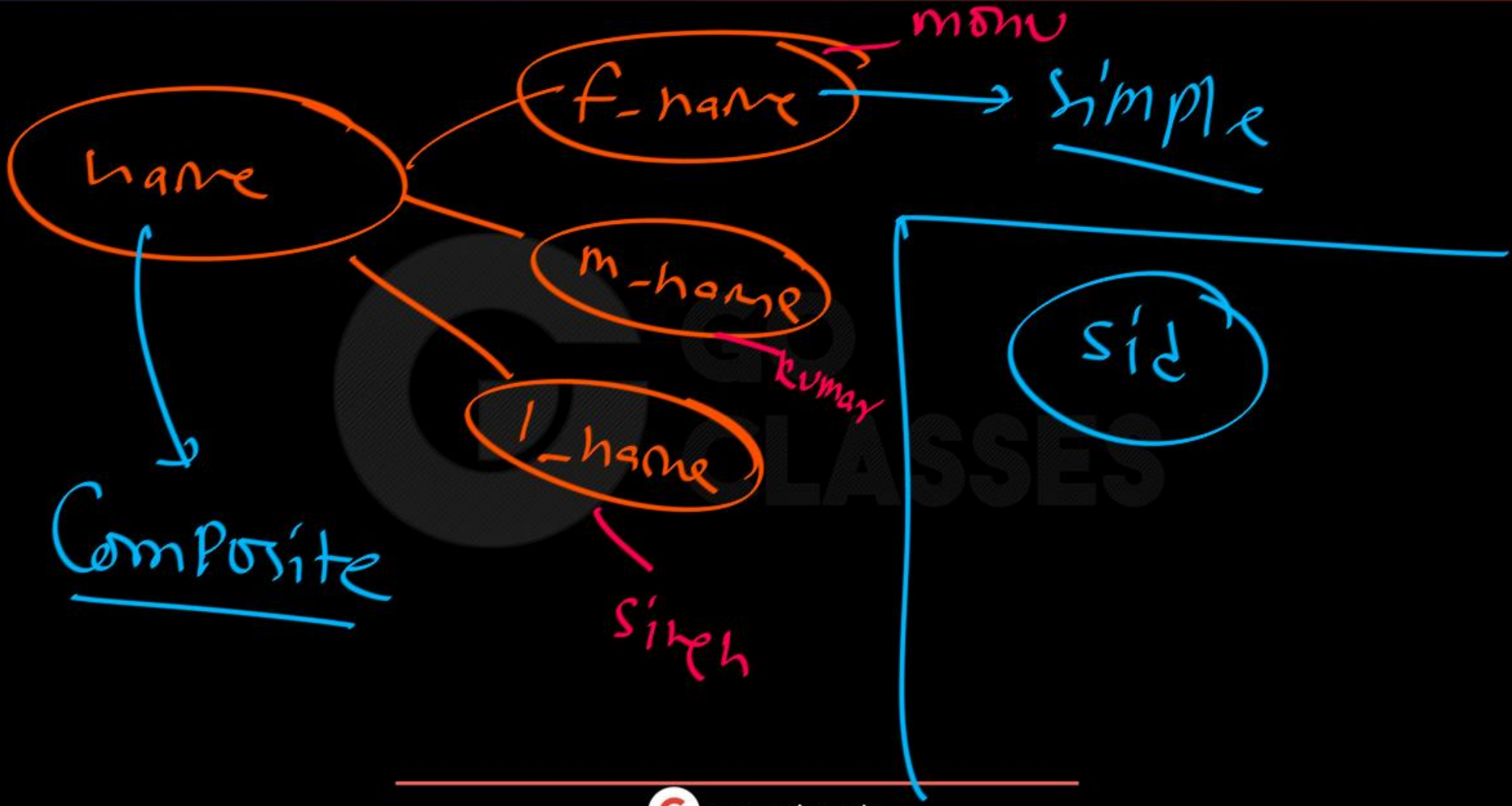
- Each attribute has a domain or value set
  - Values come from that domain or value set
- Simple attributes are atomic – they have no subparts
  - e.g. *amount* attribute is a single numeric value
- Composite attributes have subparts
  - Can refer to composite attribute as a whole
  - Can also refer to subparts individually
  - e.g. *address* attribute, composed of *street*, *city*, *state*, *postal_code* attributes

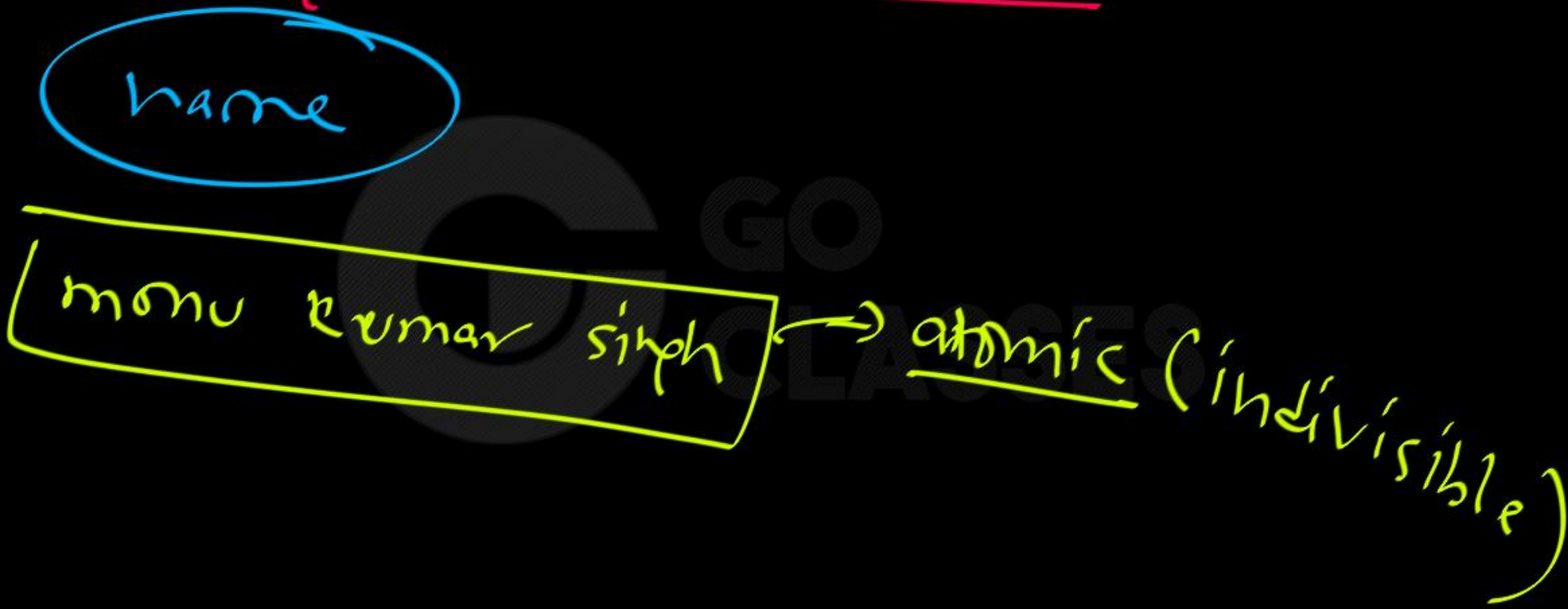
Simple attribute: No subparts

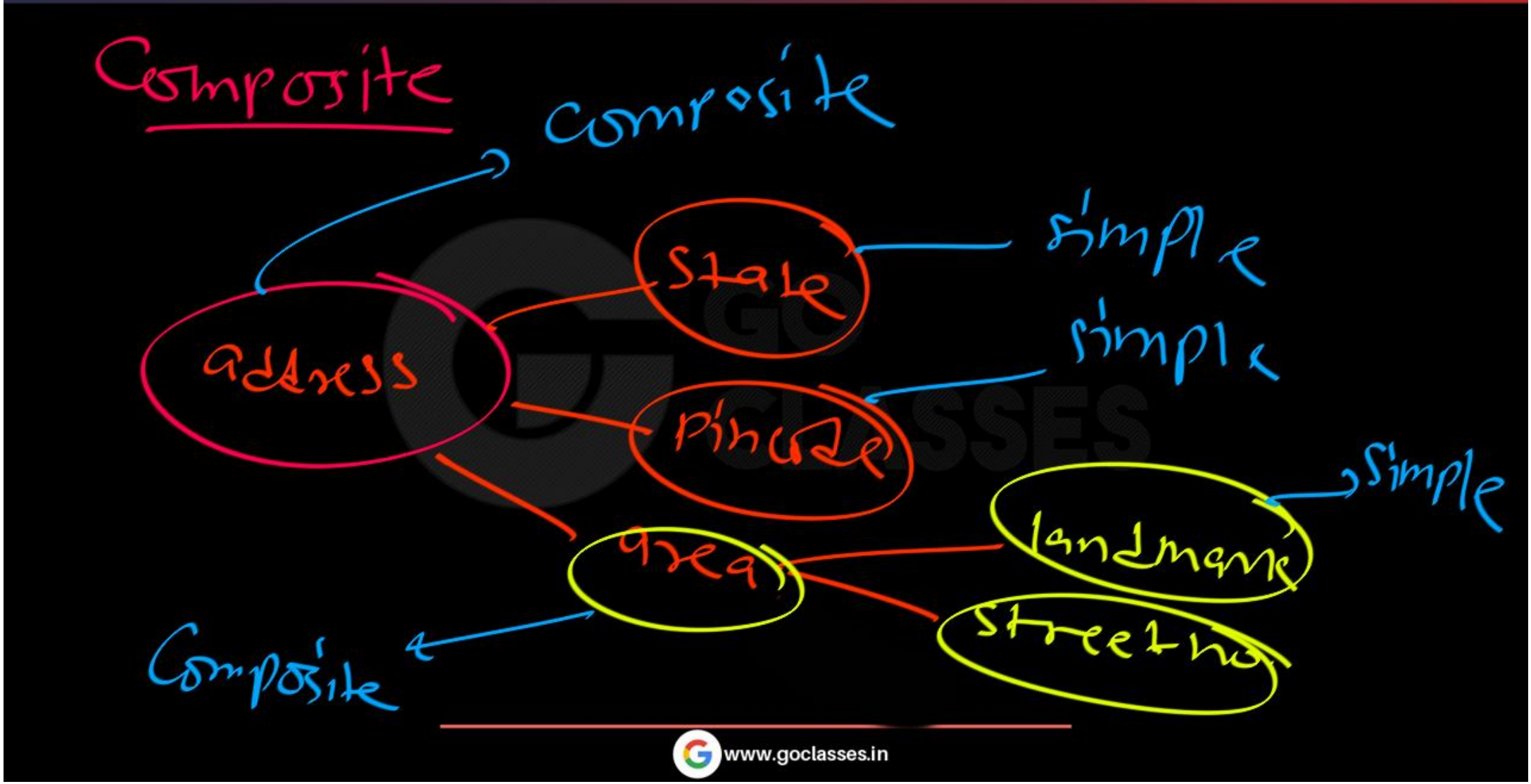
Composite attribute: Can be divided further

# Database Management System

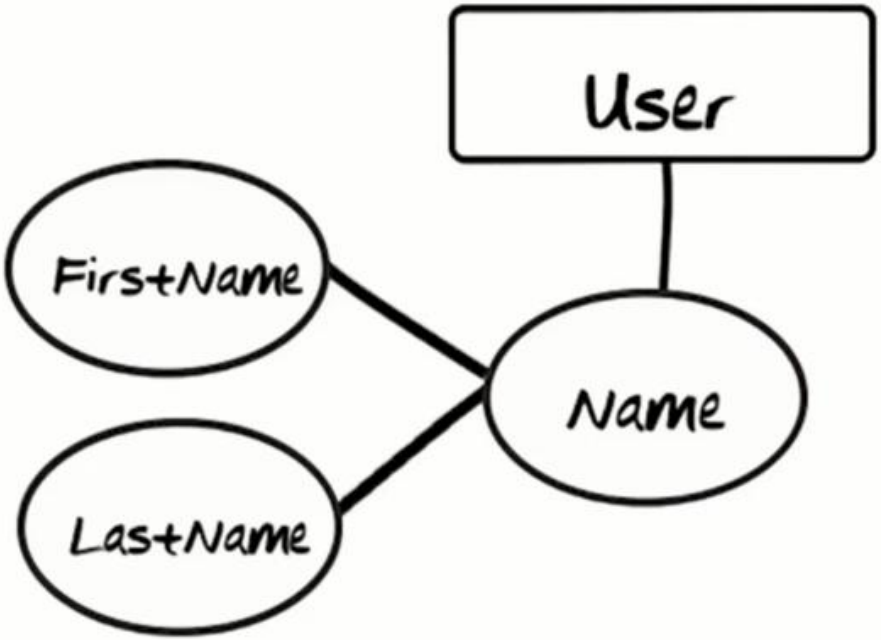


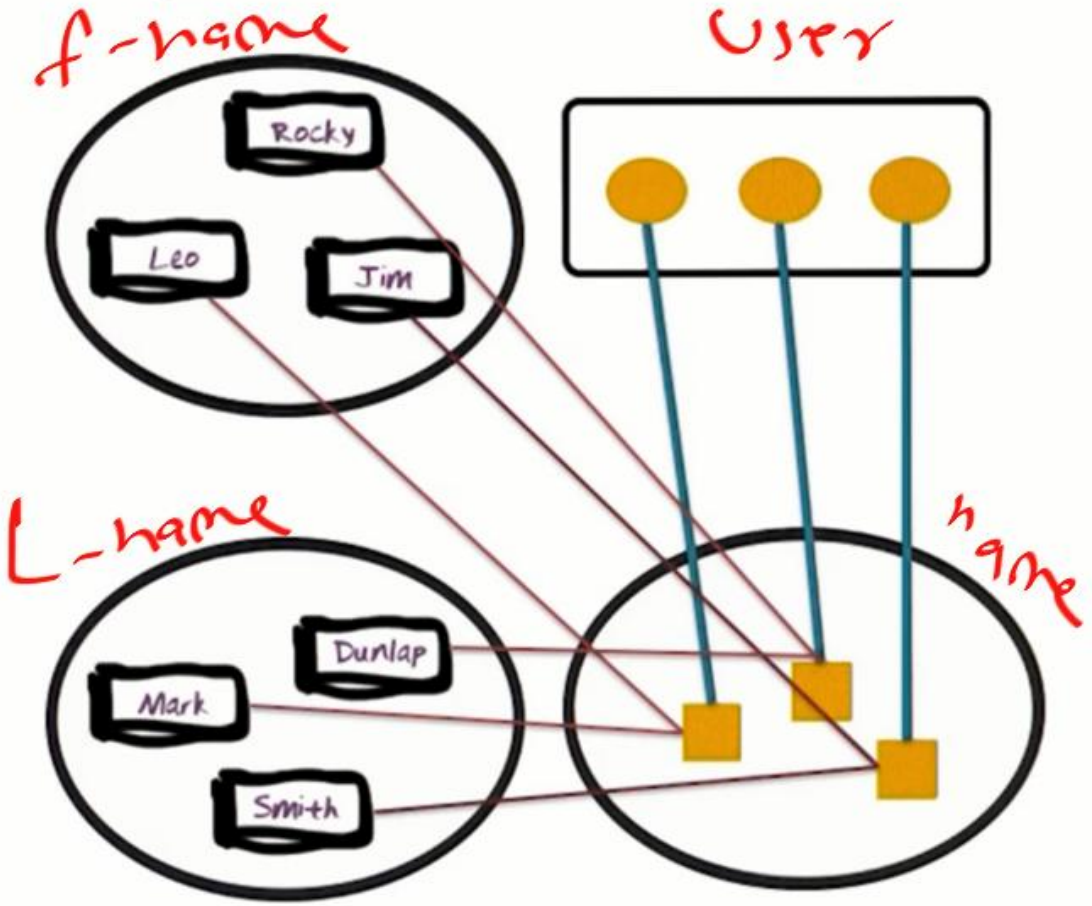



simple attribute
name
monu kumar singh
atomic (indivisible)

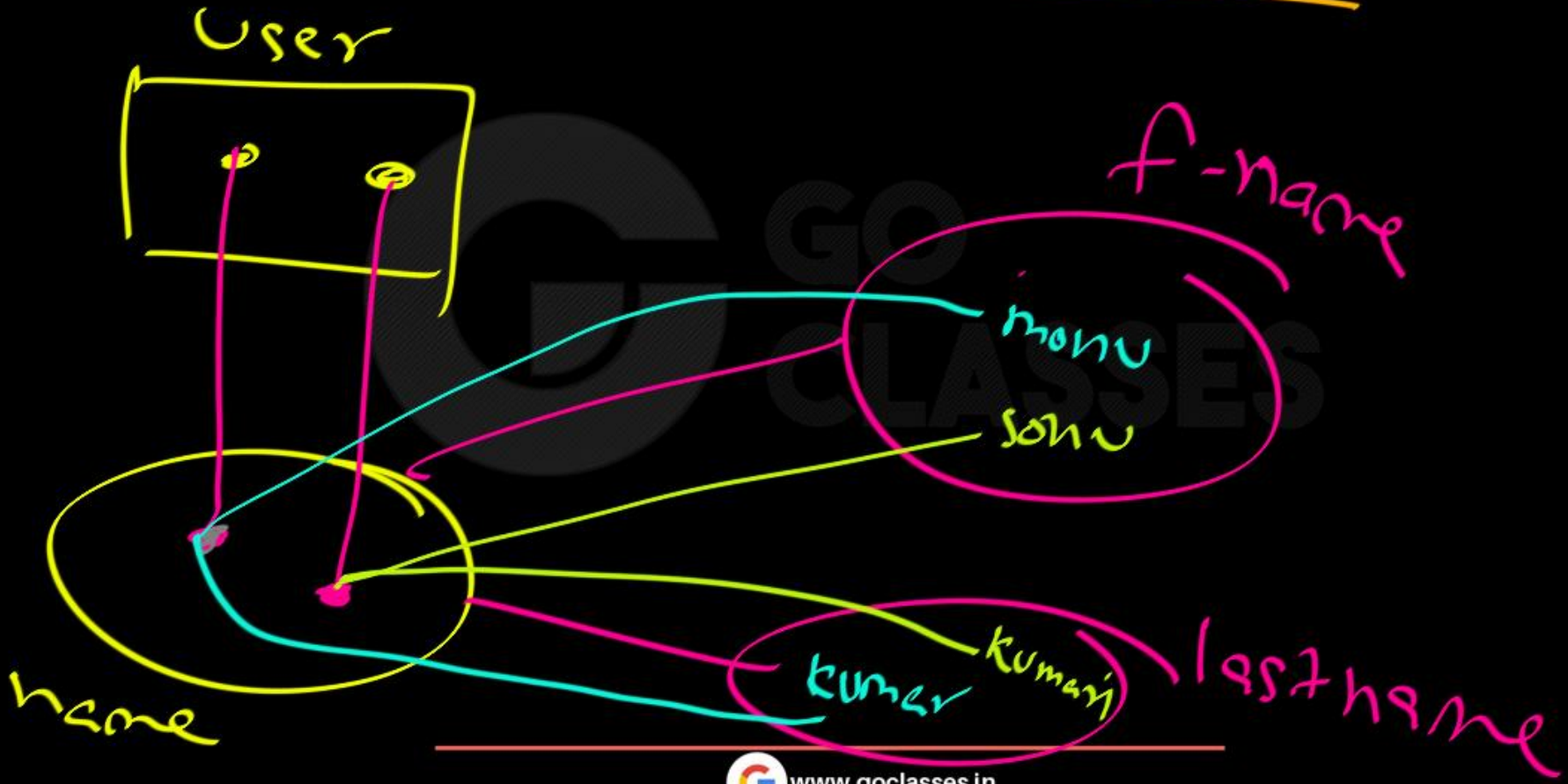

Composite
Composite
State
simple
address
simple
Pincode
Simple
landmark
Area
Composite
Streetno

# Composite Properties



# Visualization of Composite attribute:



# Attribute Cardinality

- Single-valued attributes only store one value
  - e.g. a *customer*'s *cust_id* only has one value
- Multi-valued attributes can store zero or more values
  - e.g. a *customer* can have multiple *phone_number* values
  - Different *customer* entities can have different sets of phone numbers
  - Lower and upper bounds can be specified too
    - Can set upper bound on *phone_number* to 2

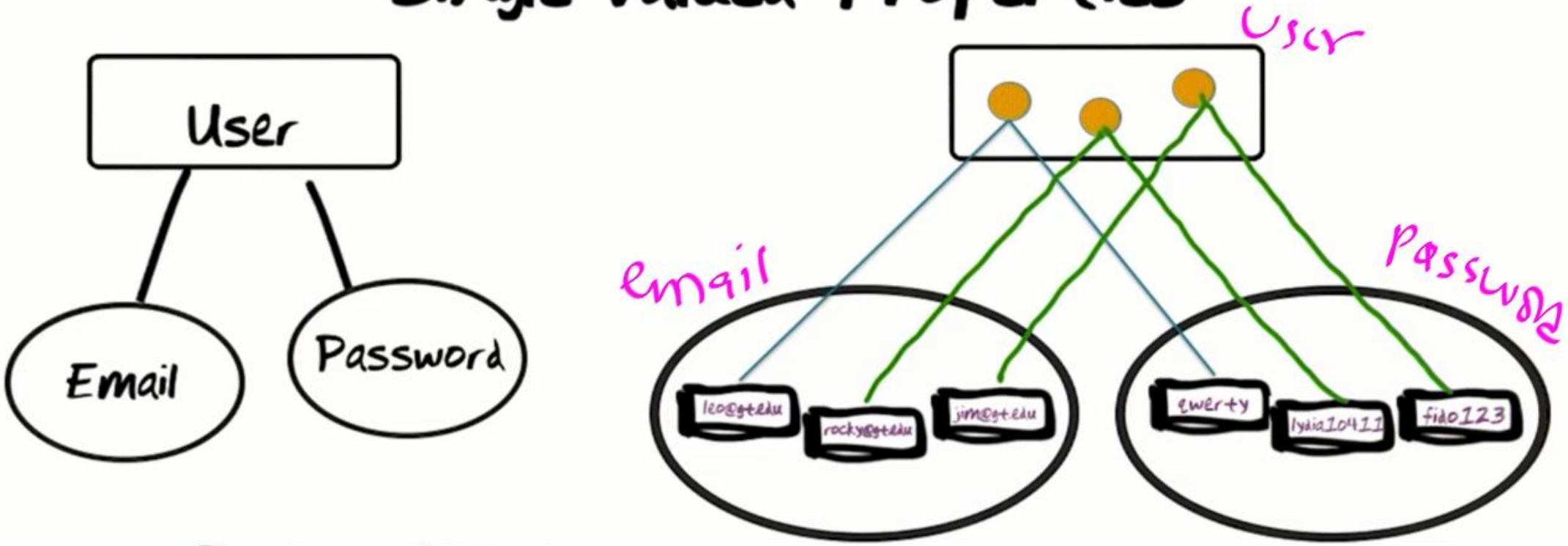
Single valued attribute : an entity has a simple value for that attribute.

Eg: Name, age, DOB

Multi valued attribute : an entity can have multiple values for that attribute.

Eg: Phone no, hobbies

# Single-valued Properties



name : Composite, simple values

address : " , multivalues

Simple, simple valued : age, password

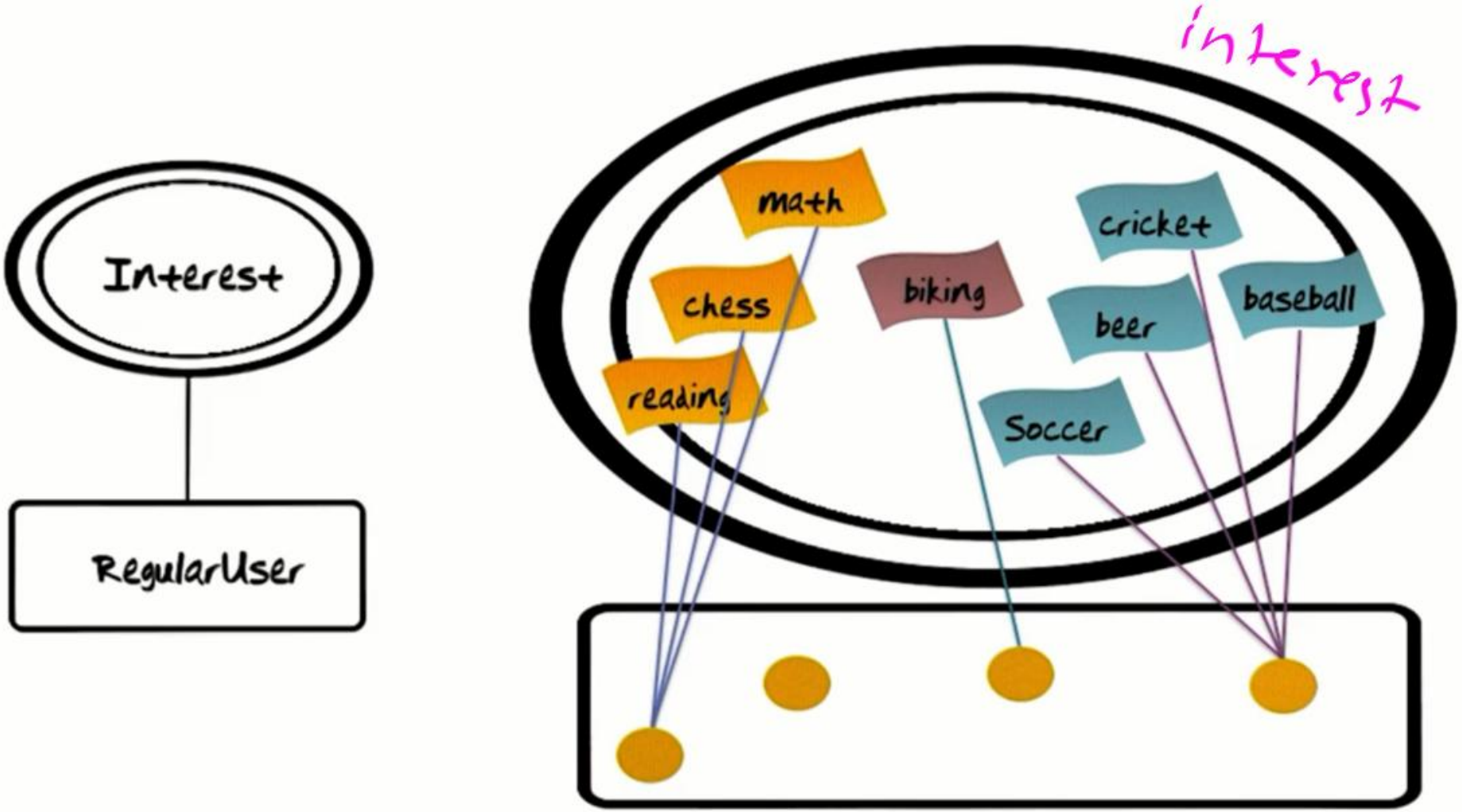
" , multivalued : phone no, hobbies

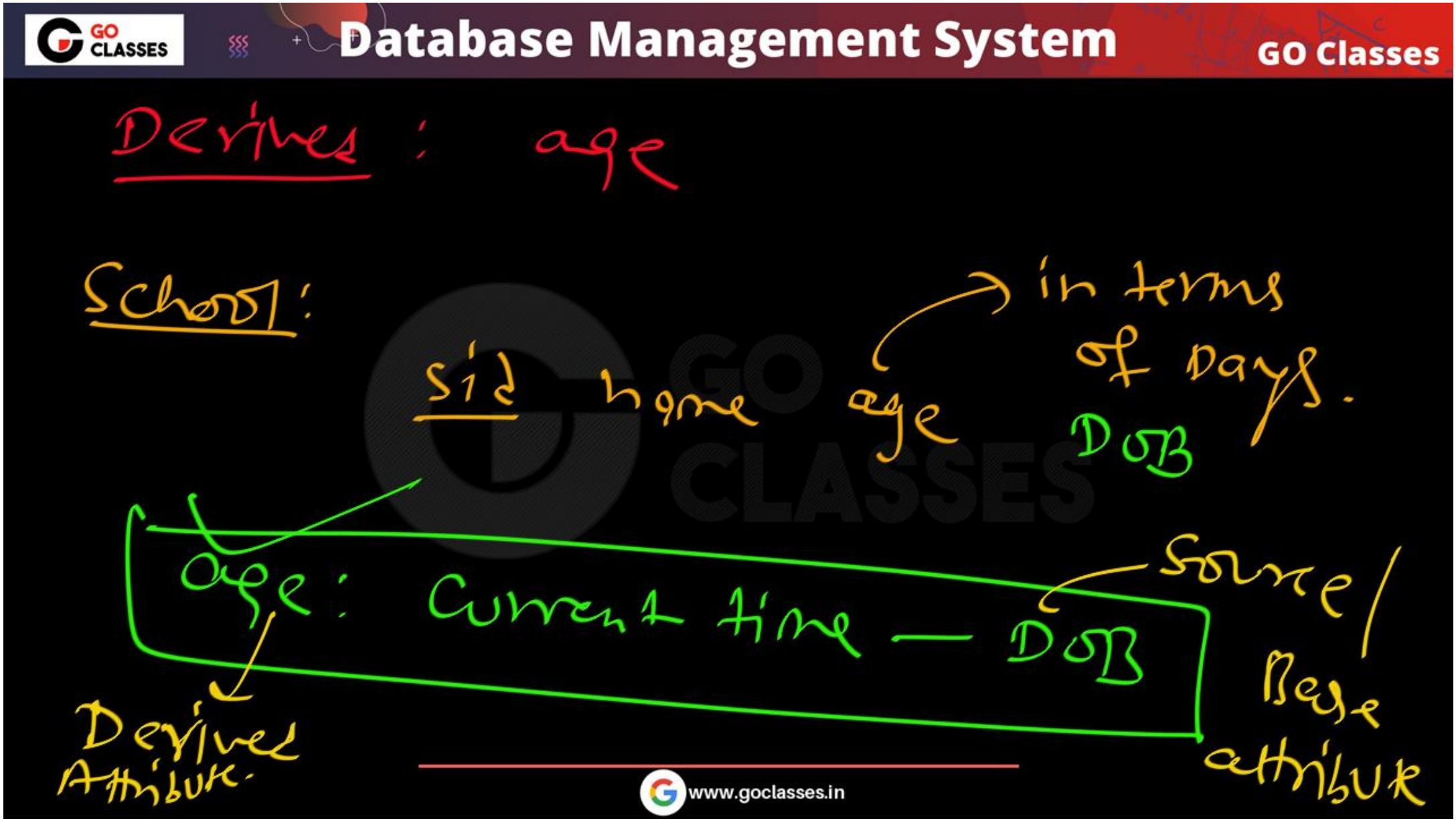
Composite, simple values : name

" , multi " : address

GO CLASSES

# Multi-valued Properties




Derives : age
School :
sid
name
age
in terms of days.
DOB
age : Current time — DOB
Source / Base attribute
Derived Attribute.

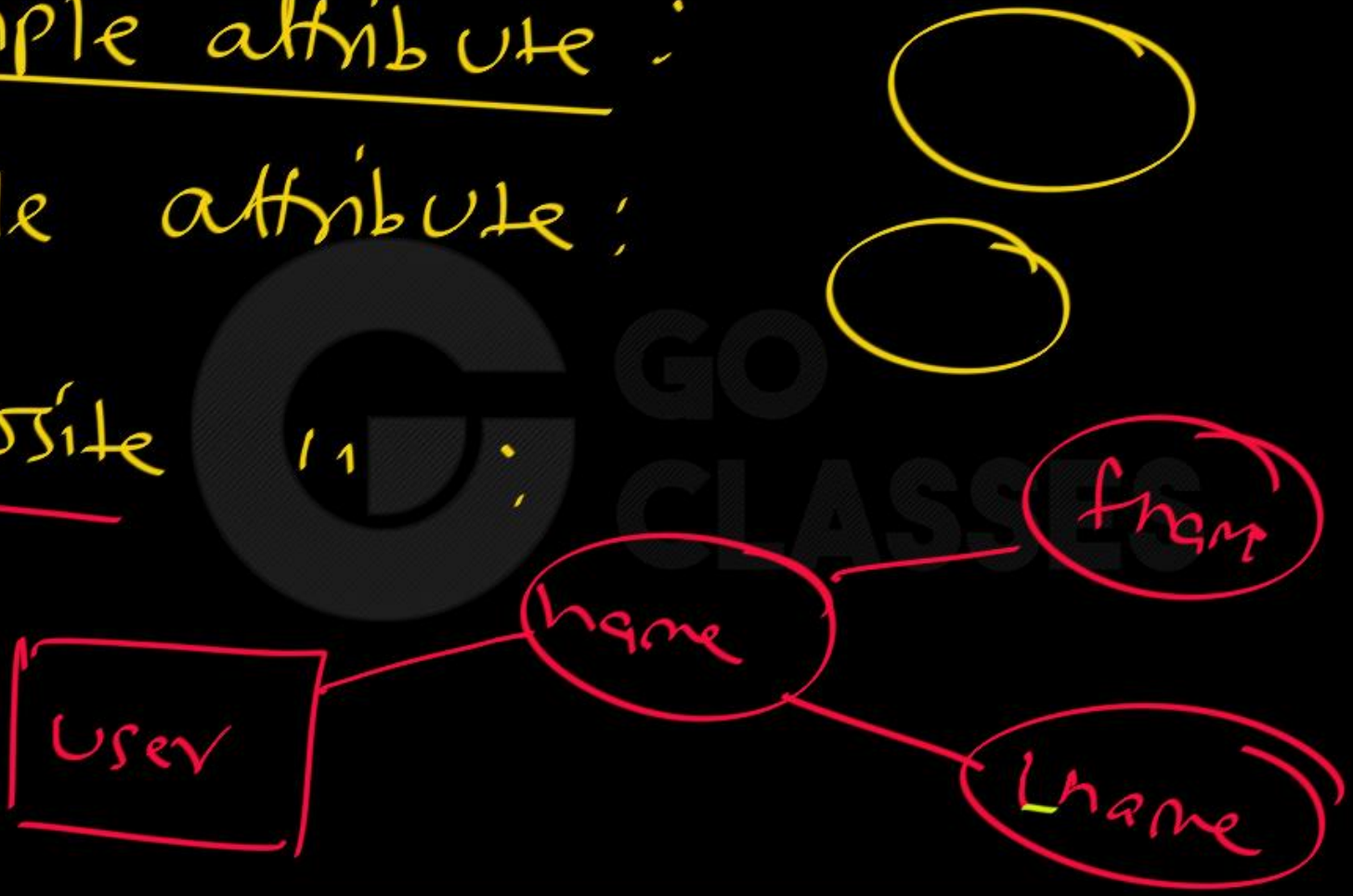
# Attribute Source

- Base attributes (aka source attributes) are stored in the database
  - e.g. the *birth_date* of a *customer* entity
- Derived attributes are computed from other attributes
  - e.g. the *age* of a *customer* entity would be computed from their *birth_date*

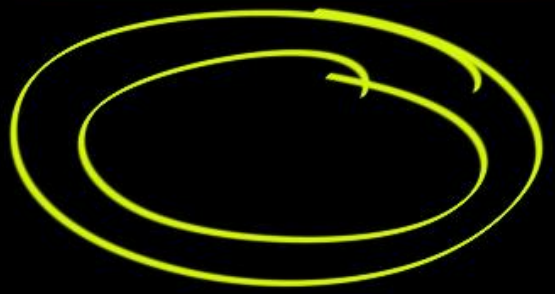
Simple attribute :

simple attribute :

Composite :



Multivalues :



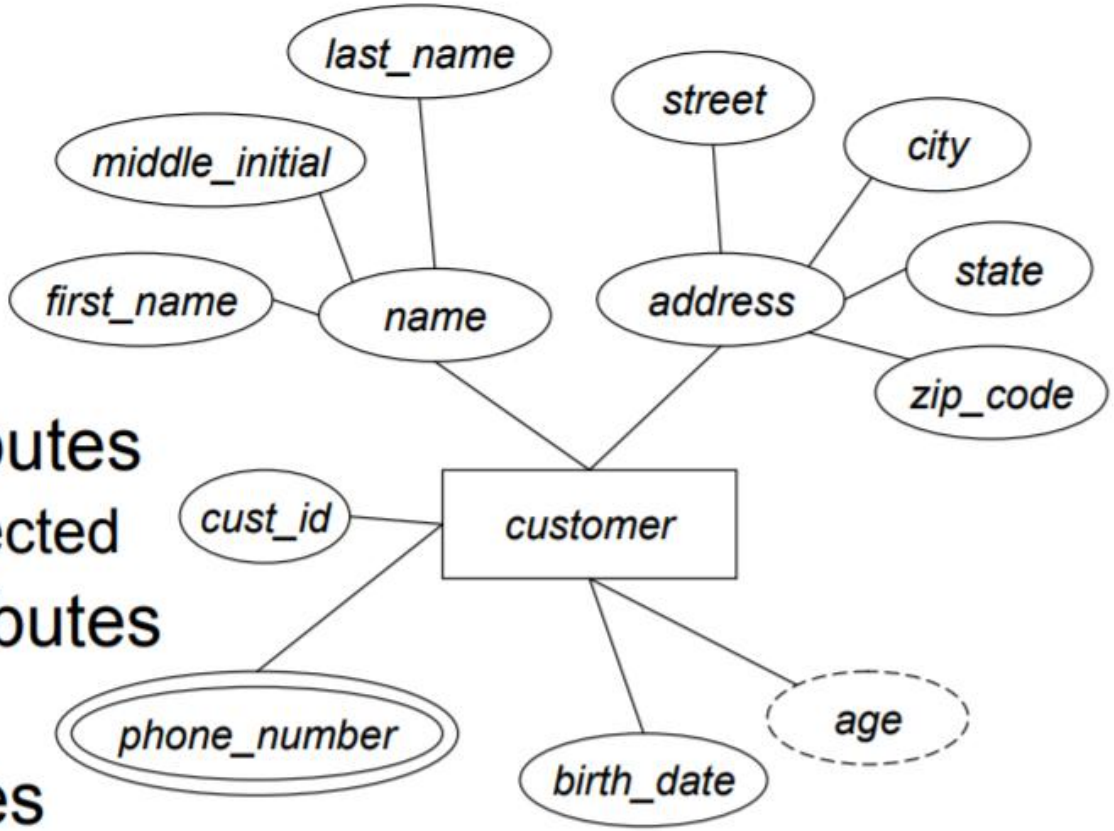
Double Oval

Derives :

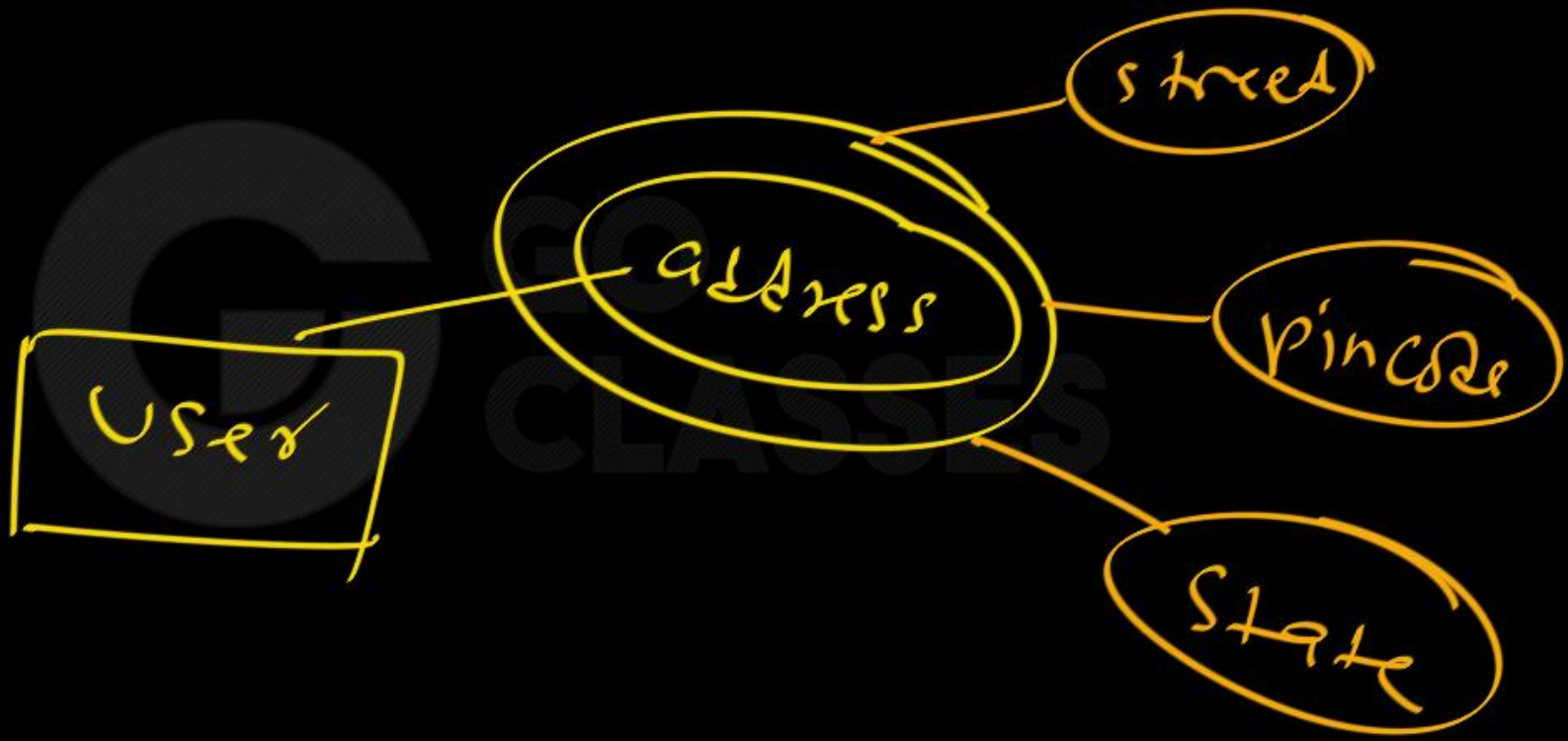
Dashes oval

# Diagramming Attributes

- Composite attributes
  - sub-parts connected
- Multivalued attributes
  - double oval
- Derived attributes
  - dashed ovals

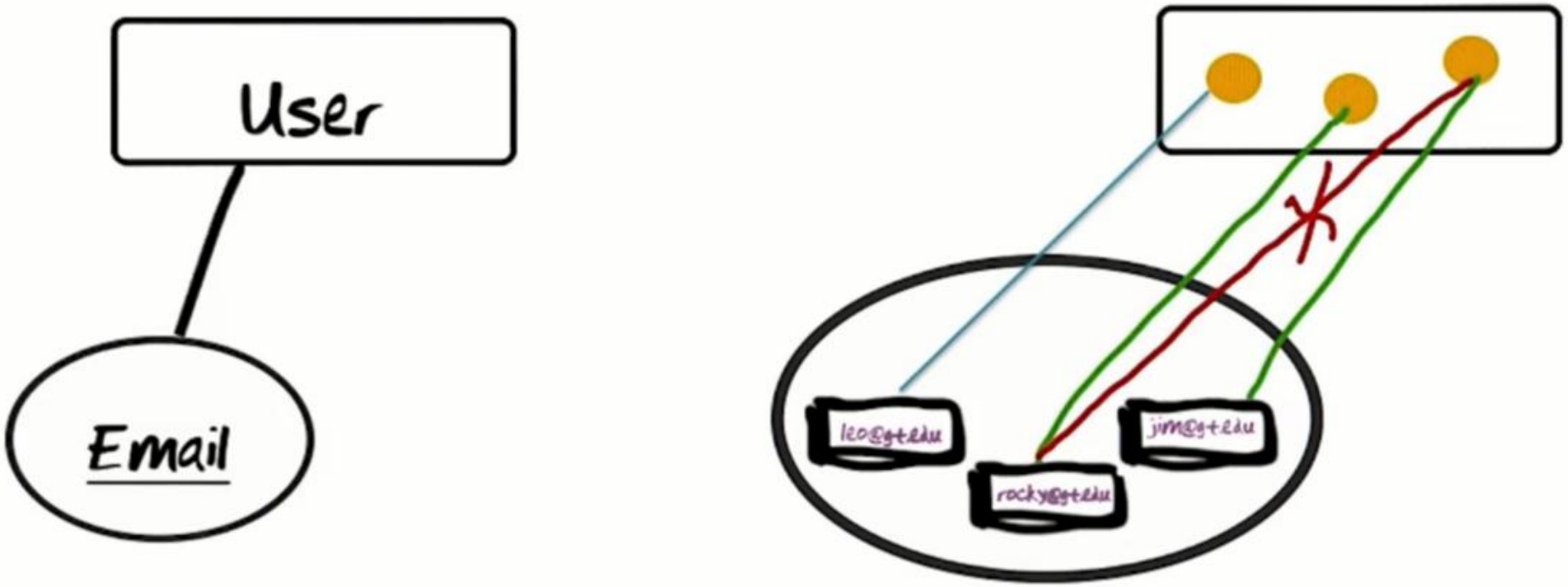


Multivalued, Composite : address



Identifying attribute : PK / CK / SK

# Identifying Properties




For each **identifying property value** there is at **most** one instance of the **identified entity**

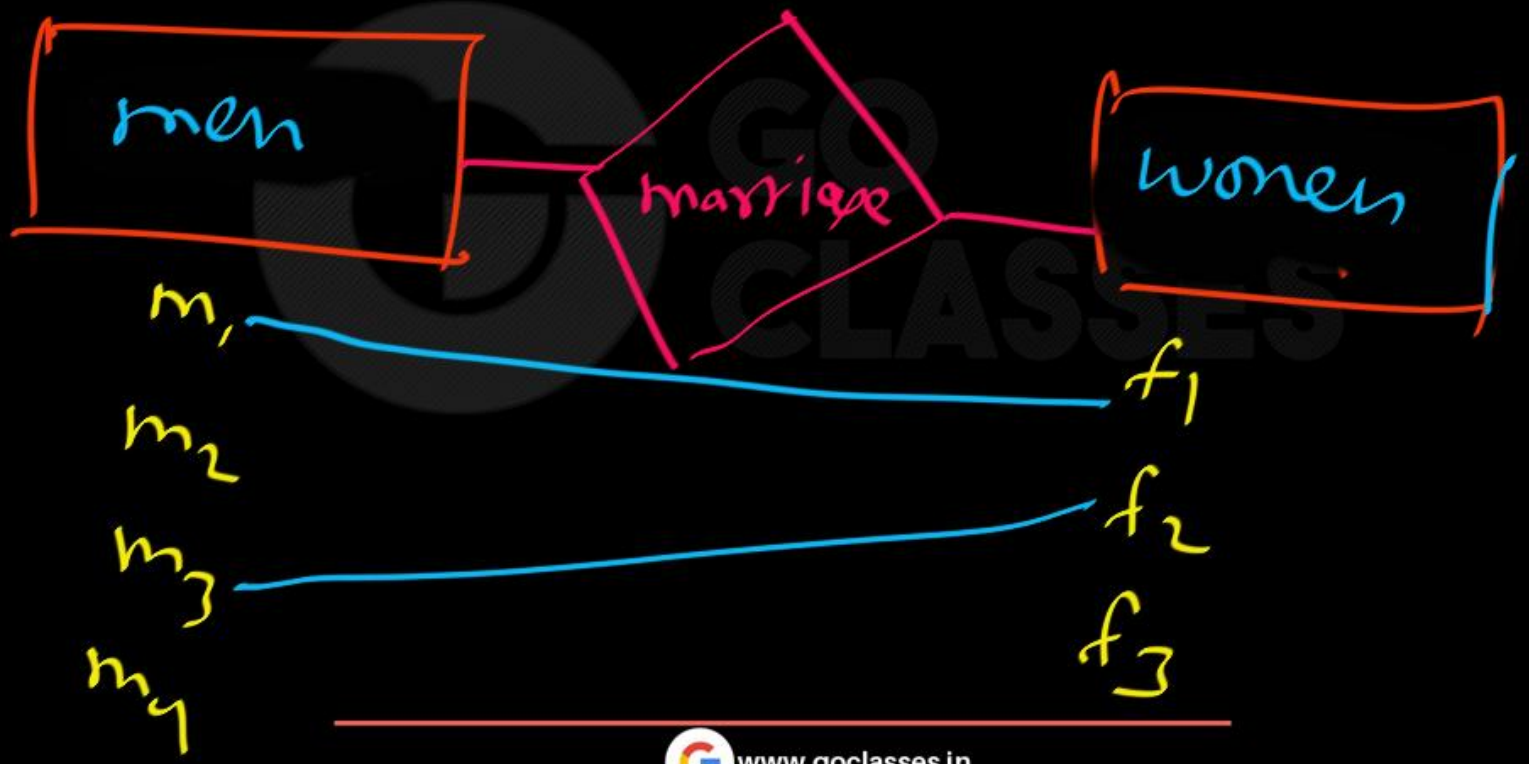
- every entity must be uniquely referenceable

# Representing Constraints

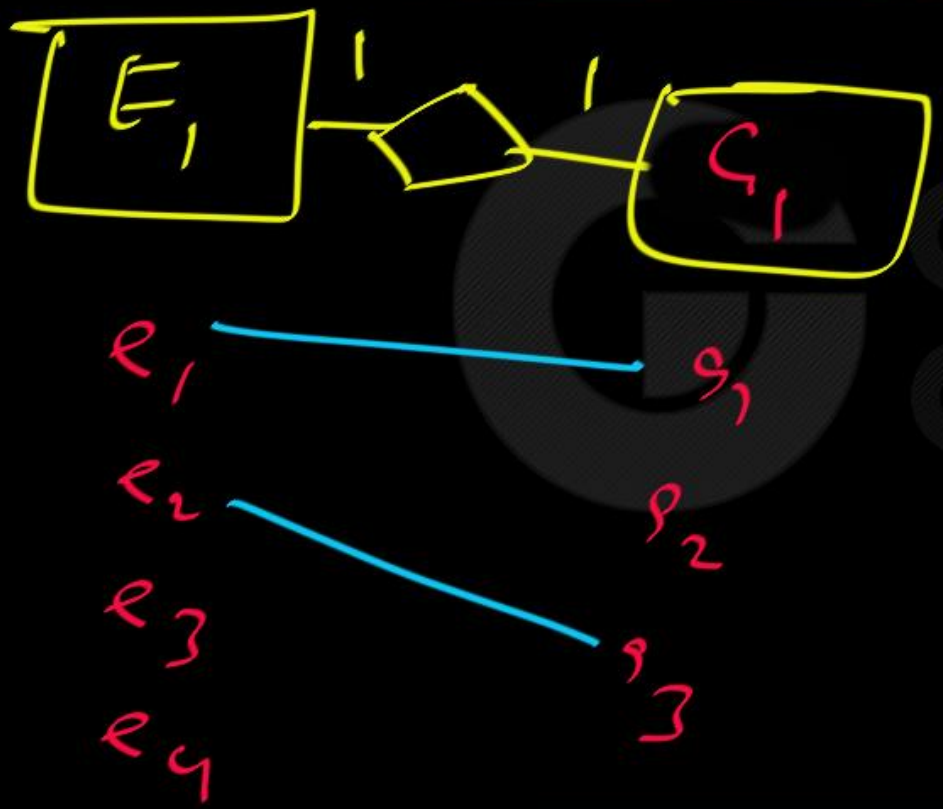
- E-R model can represent several different kinds of constraints
  - Mapping cardinalities
  - Key constraints in entity-sets
  - Participation constraints
- Allows more accurate modeling of application's data requirements
  - Constrain design so that schema can only represent valid information

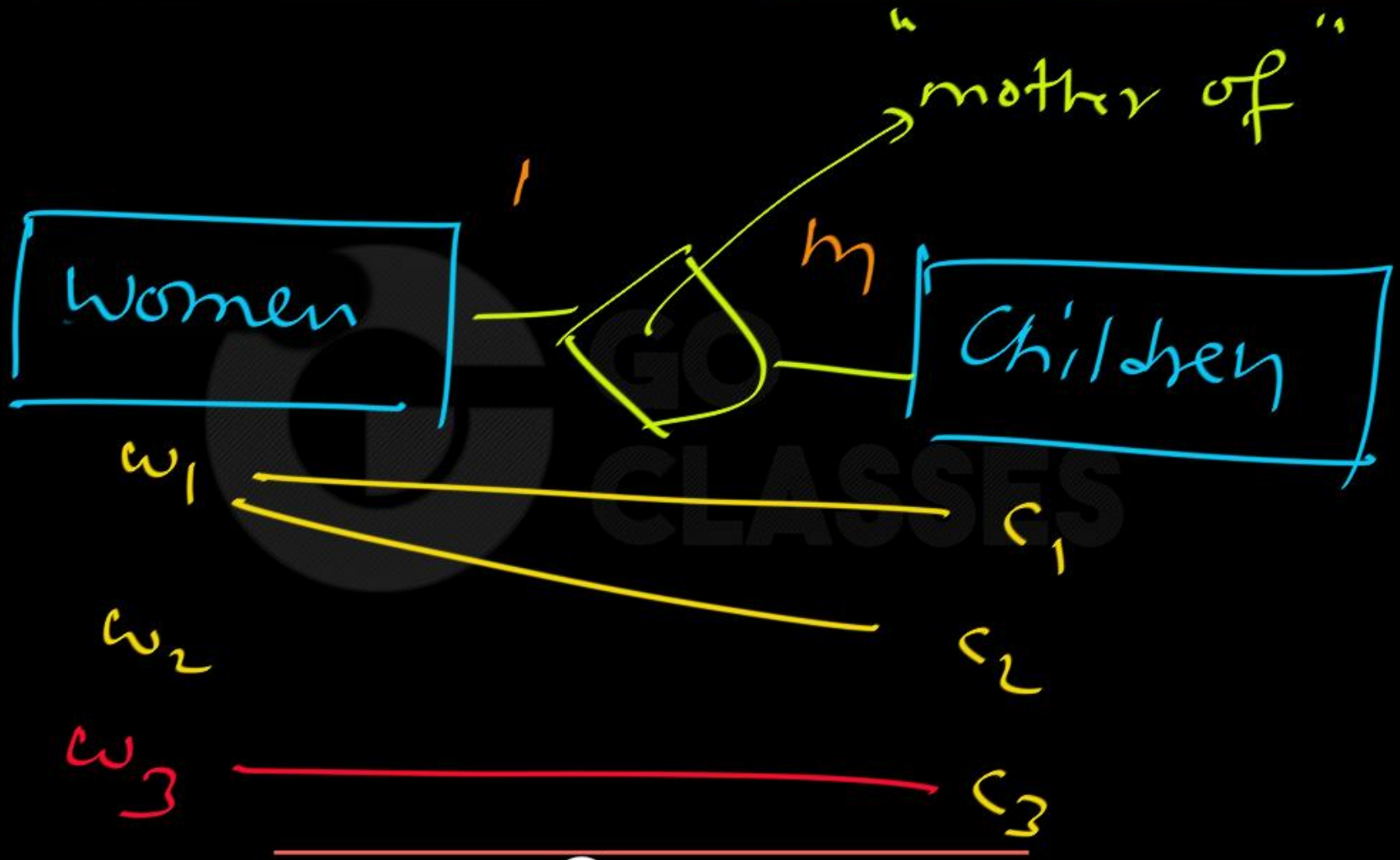
# Type of Relationships / mapping Cardinality ?

1. one-one relationship
2. one-to-many 1,1
3. many to one 1,1
4. many to many 1,1

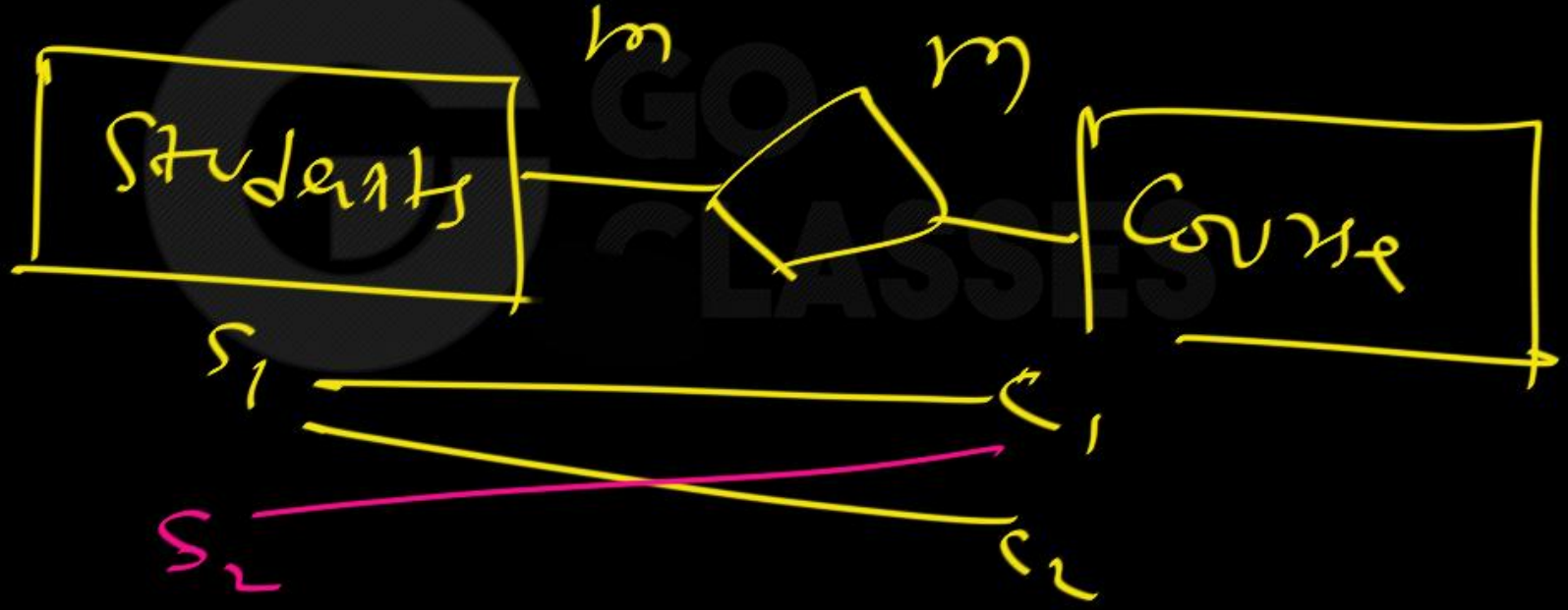

men
marriage
women
$m_1$
$m_2$
$m_3$
$m_4$
$f_1$
$f_2$
$f_3$

# 1-1 mapping / 1-1 relationship:




"mother of"
1
m
Women
Children
$w_1$
$w_2$
$w_3$
$c_1$
$c_2$
$c_3$

Many to many :



## Many to one :

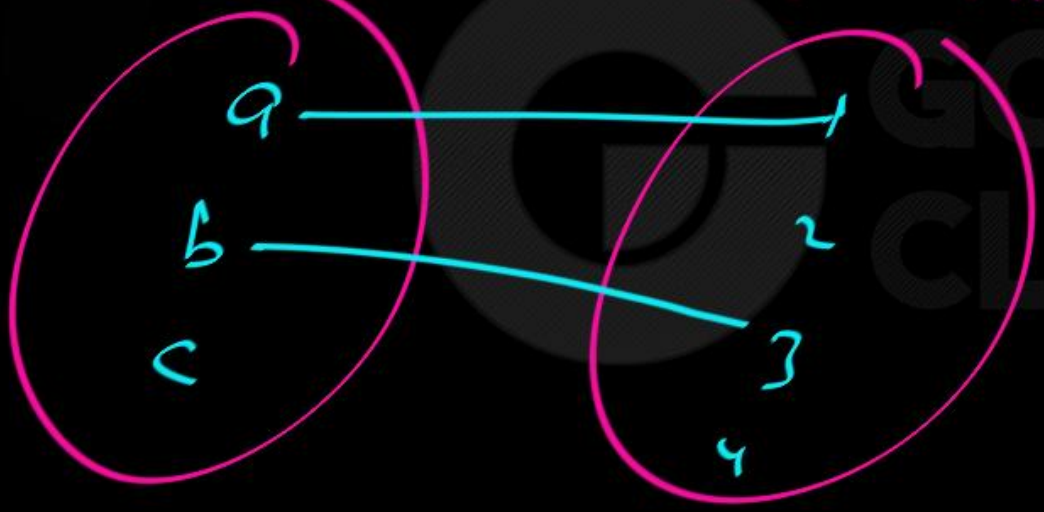

from students to College : many to one

" College to students : one to many

# Mapping Cardinalities

- Mapping cardinality represents:
  "How many other entities can be associated with an entity, via a particular relationship set?"
- Example:
  - How many *customer* entities can the *borrower* relation associate with a single *loan* entity?
  - How many *loans* can *borrower* relation associate with a single *customer* entity?
  - Specific answer depends on requirements
- Also known as the cardinality ratio
- Easiest to reason about with binary relations

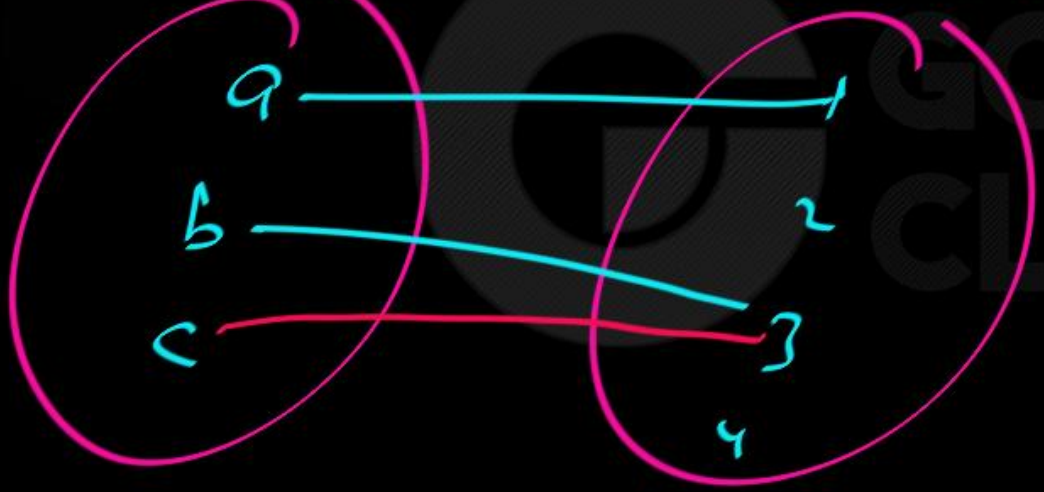
# Partial function:




Every element of Domain is mapped to at most one element of Codomain.

# Total function:

Every element of Domain is mapped to Exactly one element of Codomain.

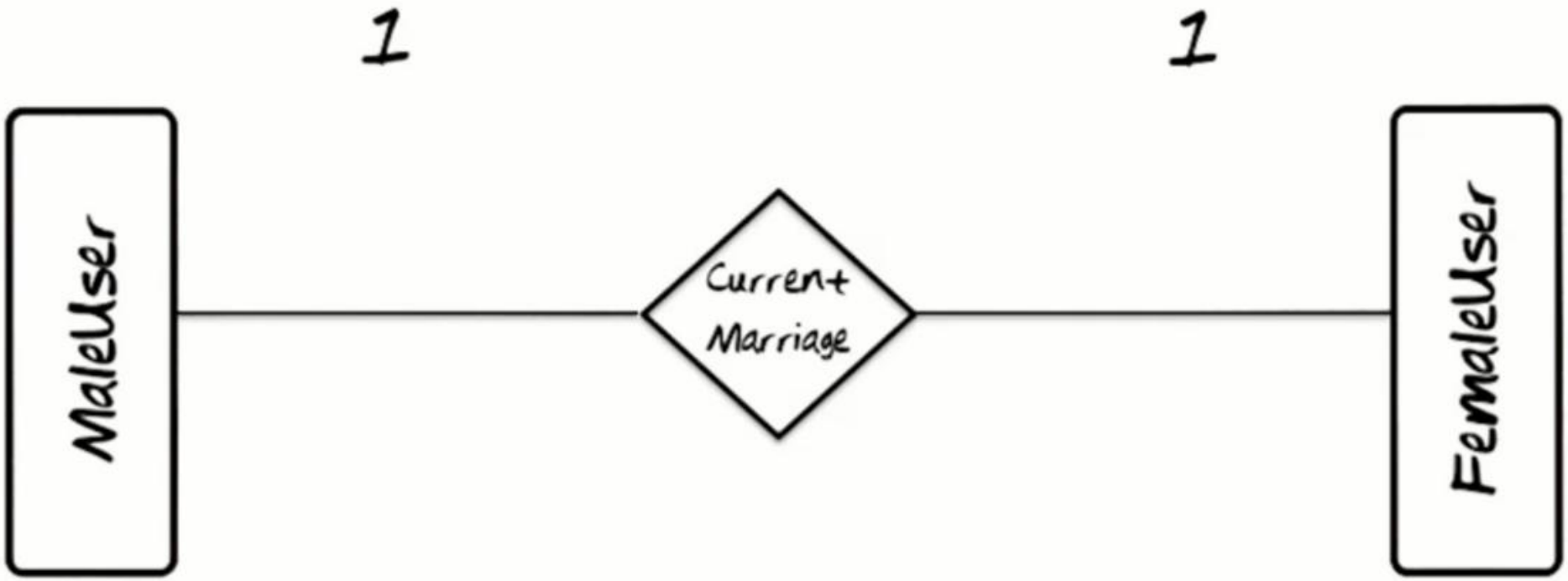


Every (Total) function is Partial function. But not vice versa.

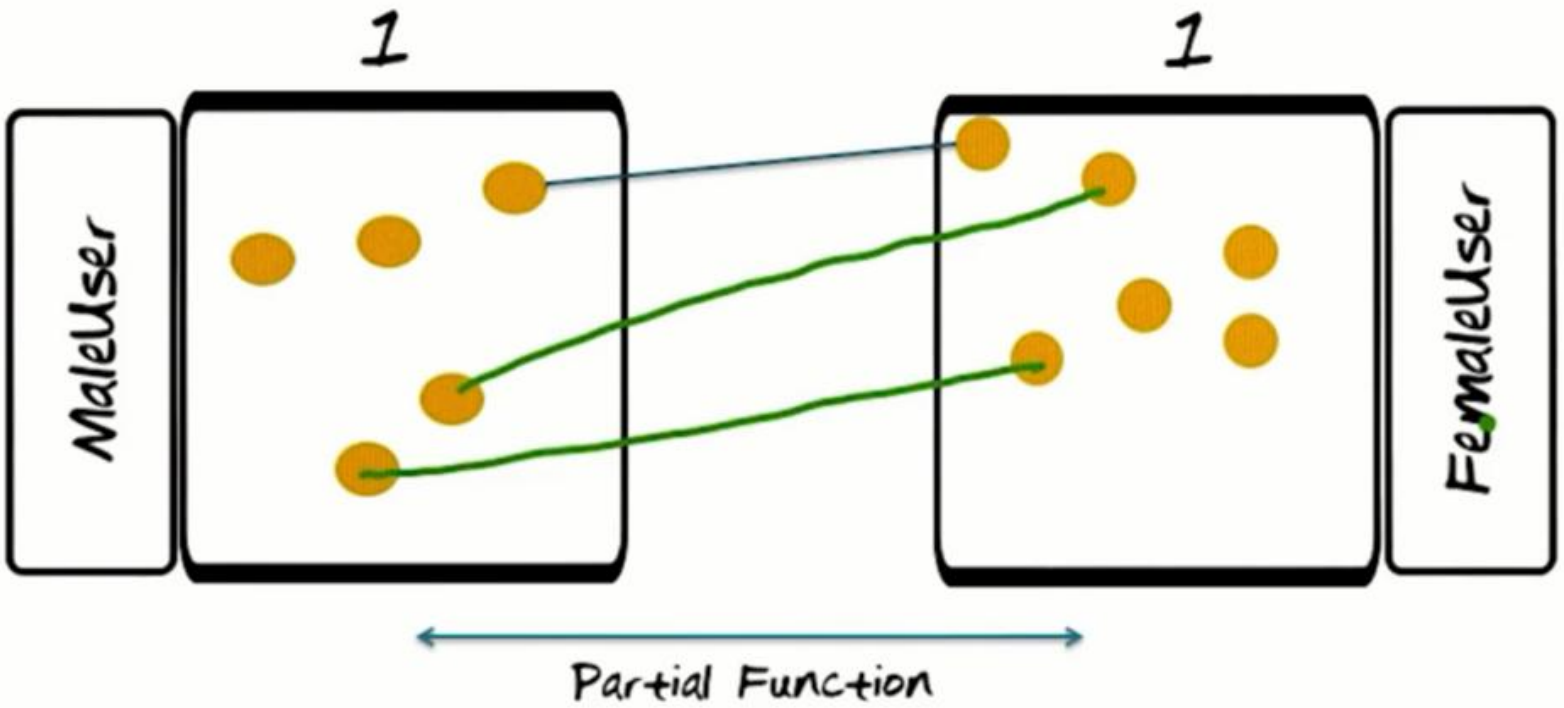
By default,

function $\equiv$ Total function.

# 1-1 relationship types



# 1-1 relationship types




the names of multiple relationship types between the same two entity types must be unique

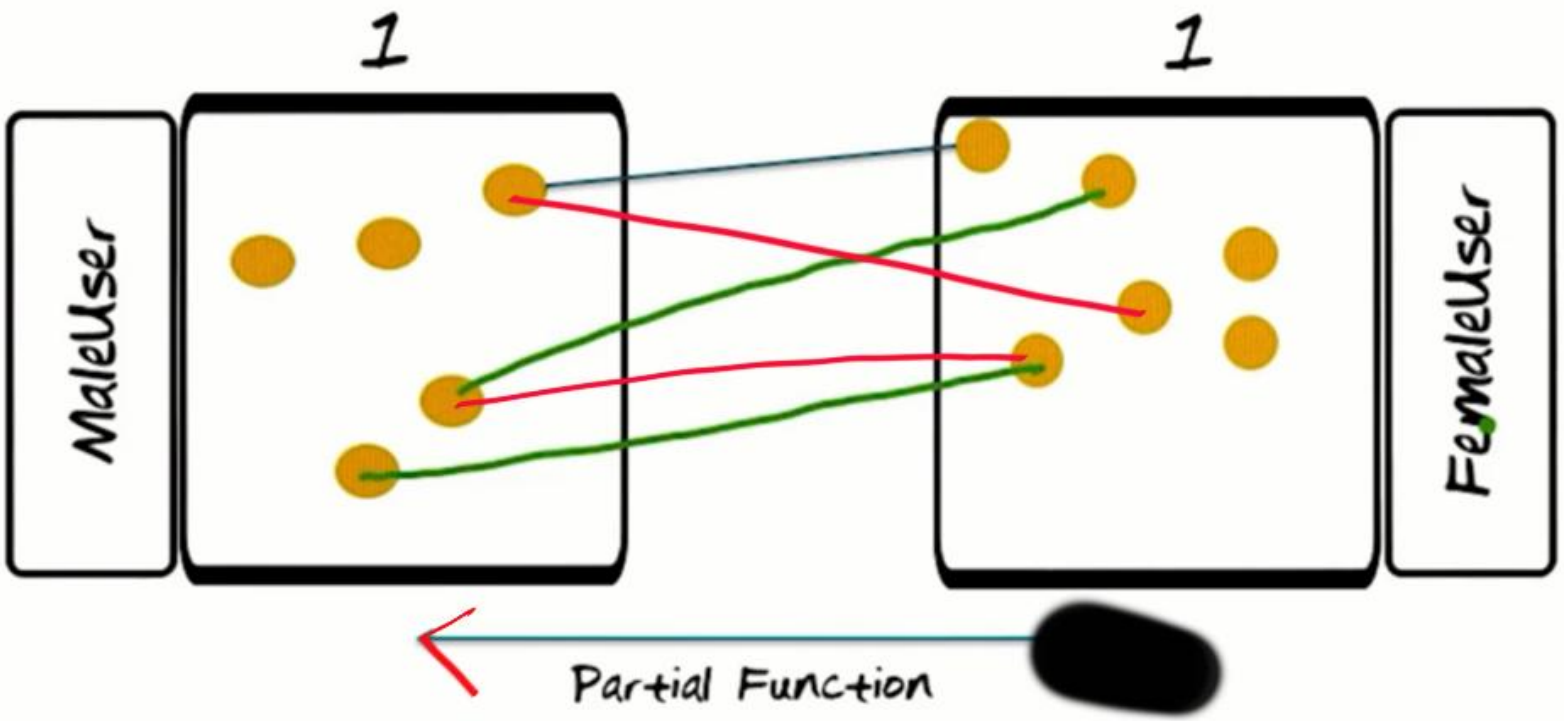
## 1-1 relationship:

from maleusers to female users : Partial function

" femaleuser to maleusers : "

"

# One to many relationship types




the names of multiple relationship types between the same two entity types must be unique

1-m relationship:

from maleusers to female users : Not a function (but a relation)

" femaleuser to maleusers : Partial function

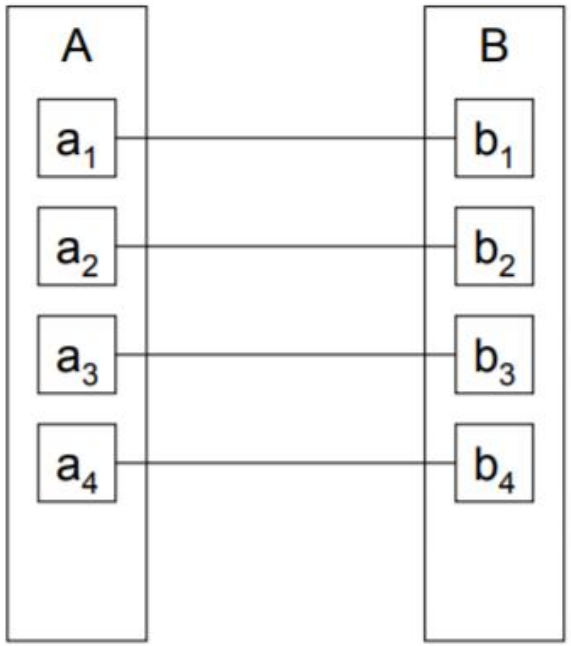
# Mapping Cardinalities (2)

Given:

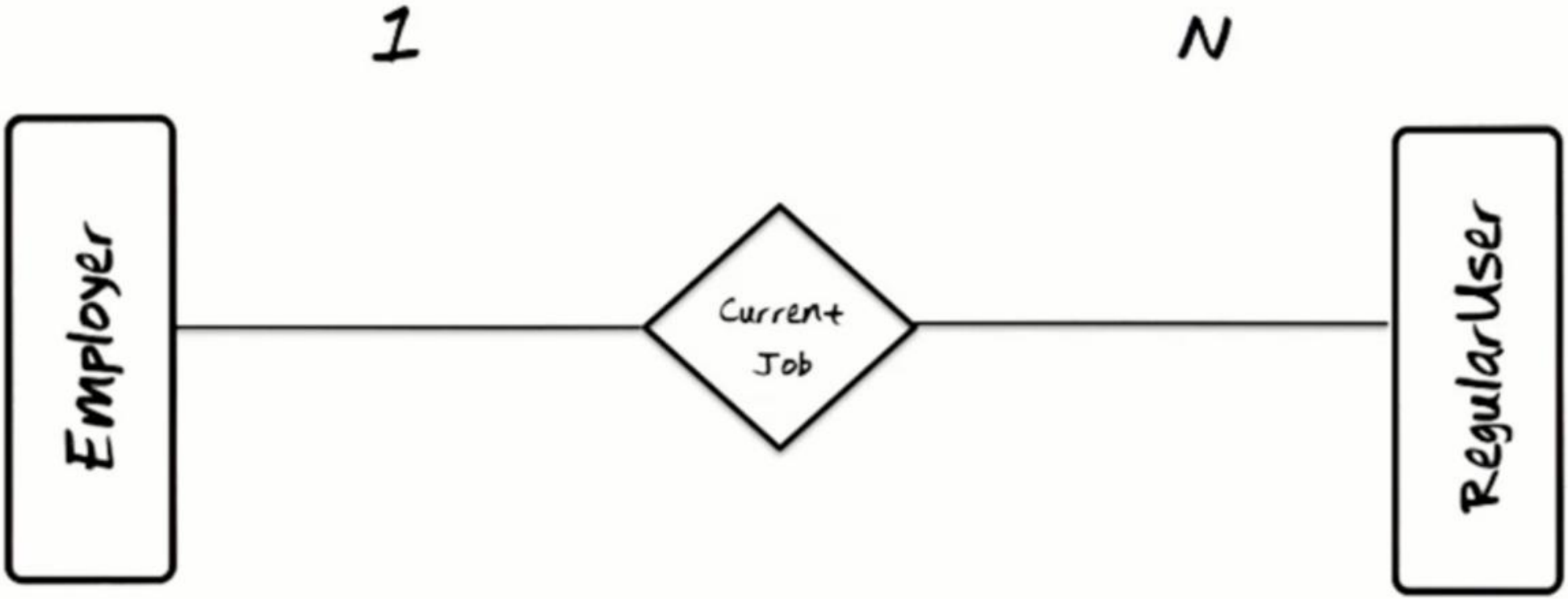
- Entity-sets *A* and *B*
- Binary relation-set *R* associating *A* and *B*

One-to-one mapping (1:1)

- An entity in *A* is associated with *at most* one entity in *B*
- An entity in *B* is associated with *at most* one entity in *A*

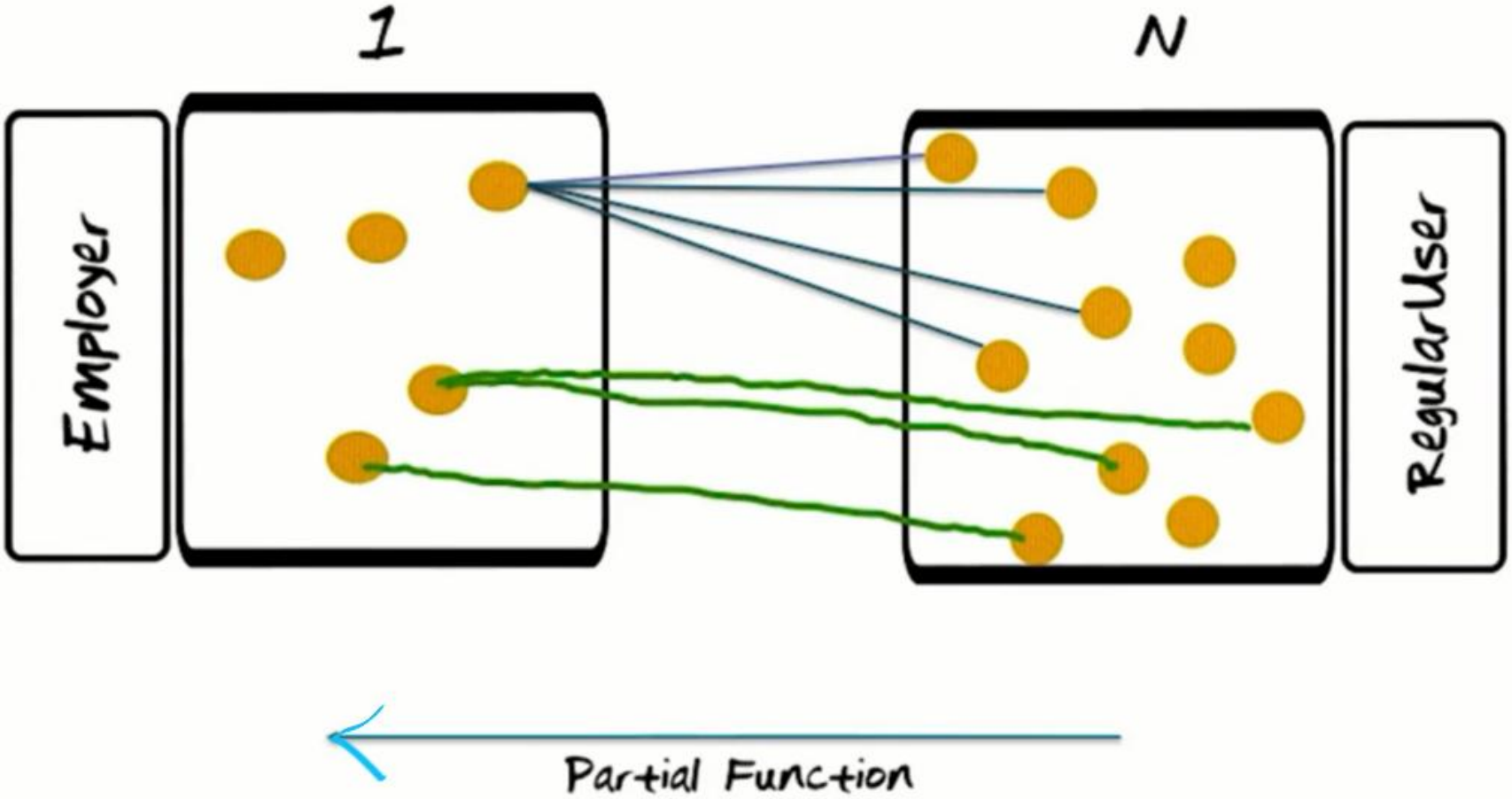


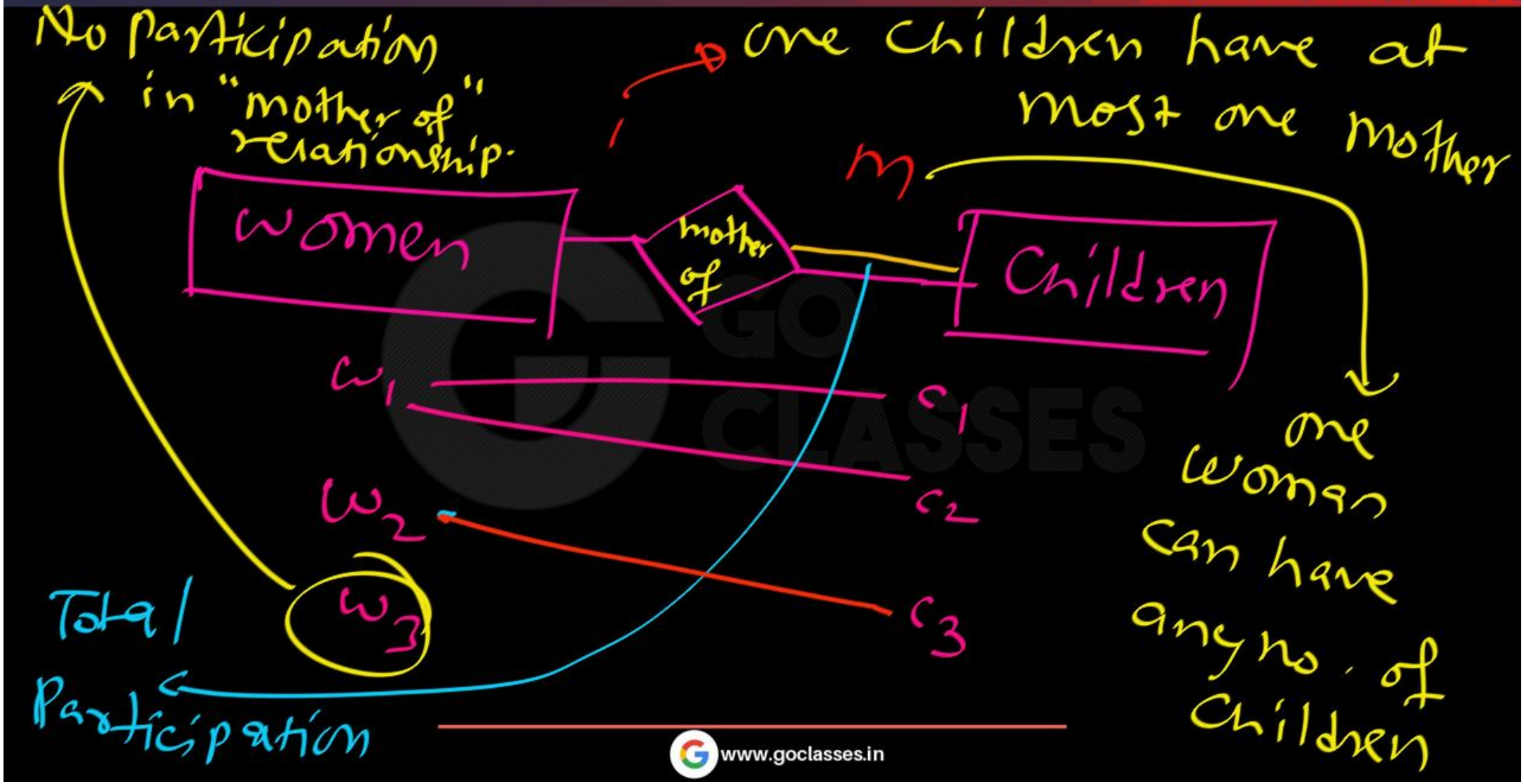
# 1-many relationship types



GO CLASSES

# 1-many relationship types

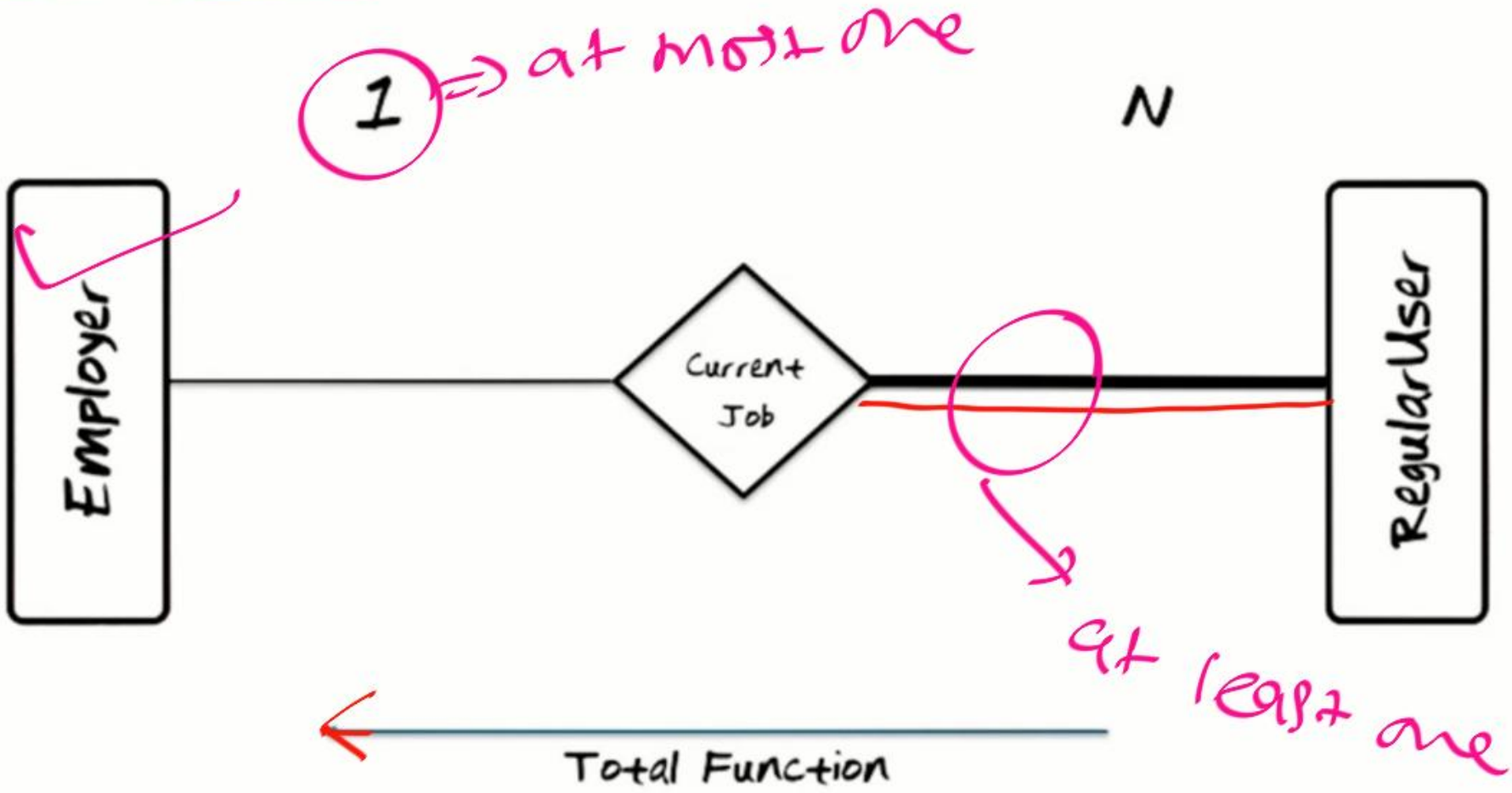



No participation in "mother of" relationship.
one children have at most one mother
1
M
women
mother of
Children
$w_1$
$c_1$
$c_2$
$w_2$
$c_3$
$w_3$
Total participation
one woman can have any no. of children

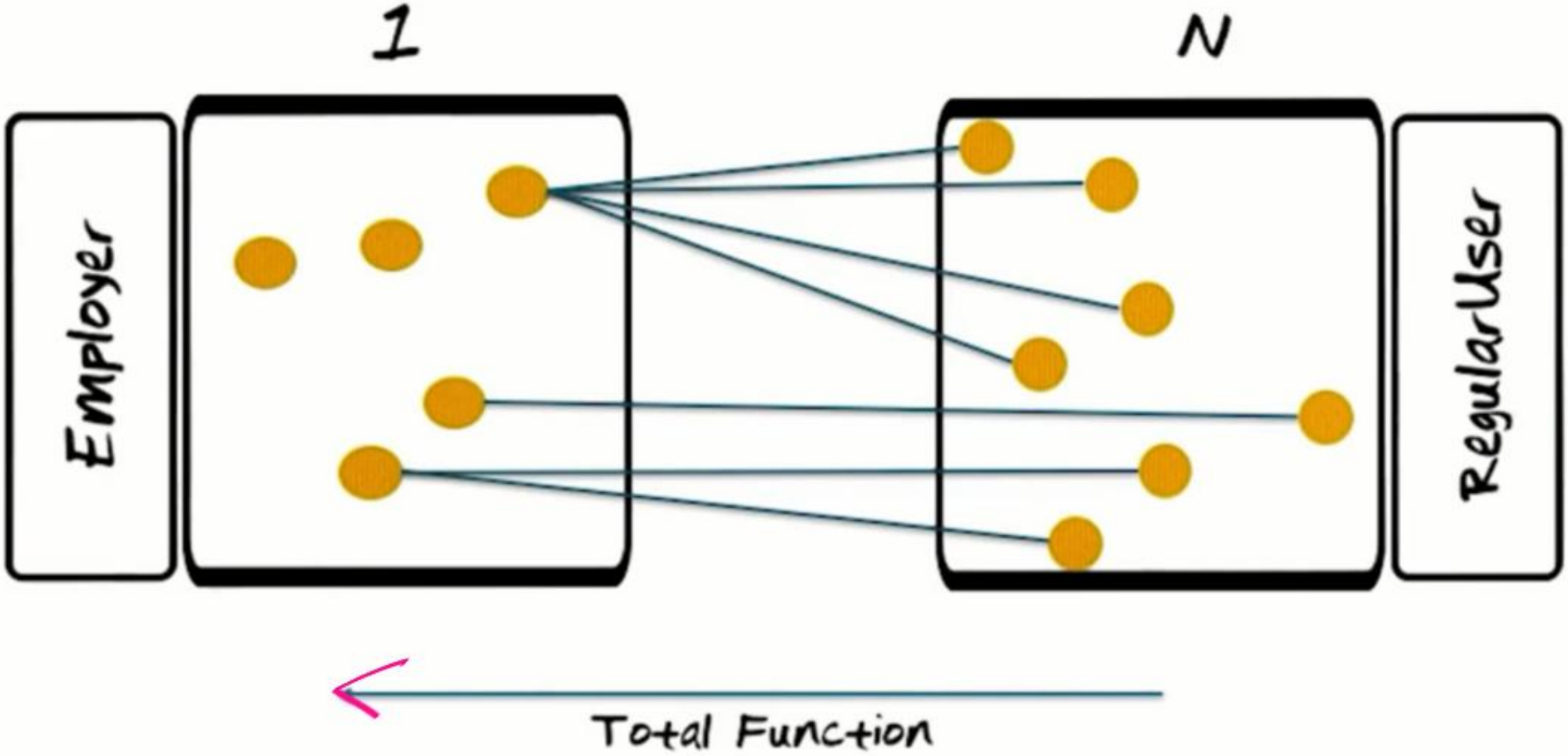
# Participation of entities in a Relationship:

1. Total : mandatory participation
2. Partial : Not mandatory "

# Mandatory 1-N relationship types



# Mandatory 1-N relationship types



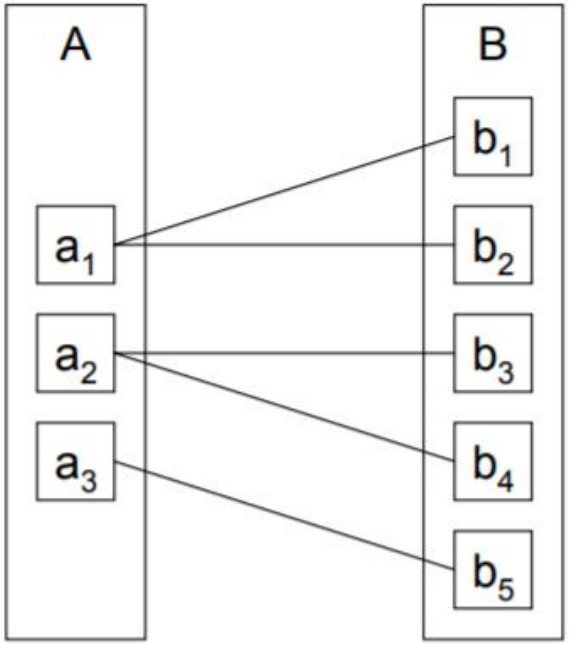
# Mapping Cardinalities (3)

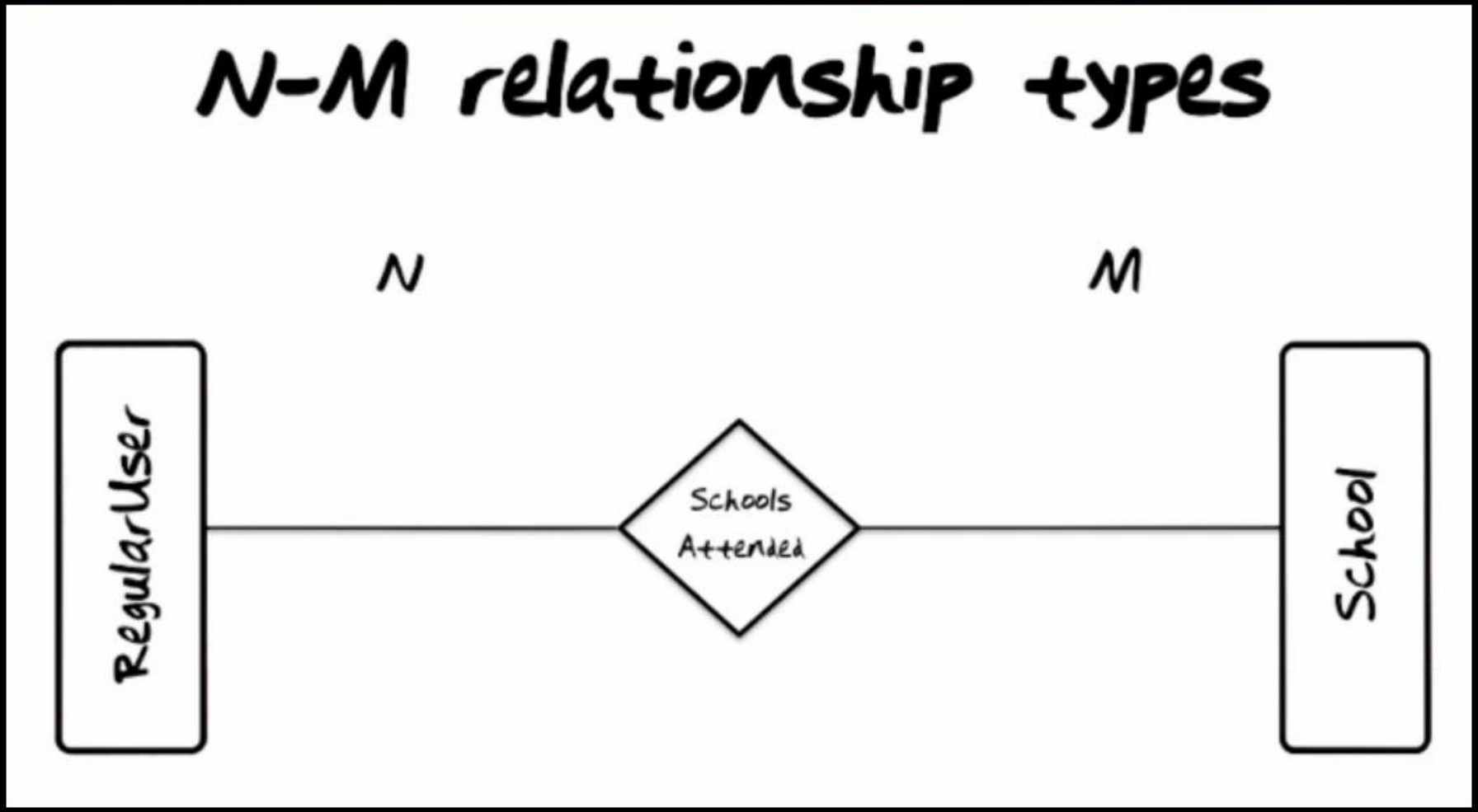
One-to-many mapping (1:M)

- An entity in *A* is associated with *zero or more* entities in *B*
- An entity in *B* is associated with *at most* one entity in *A*

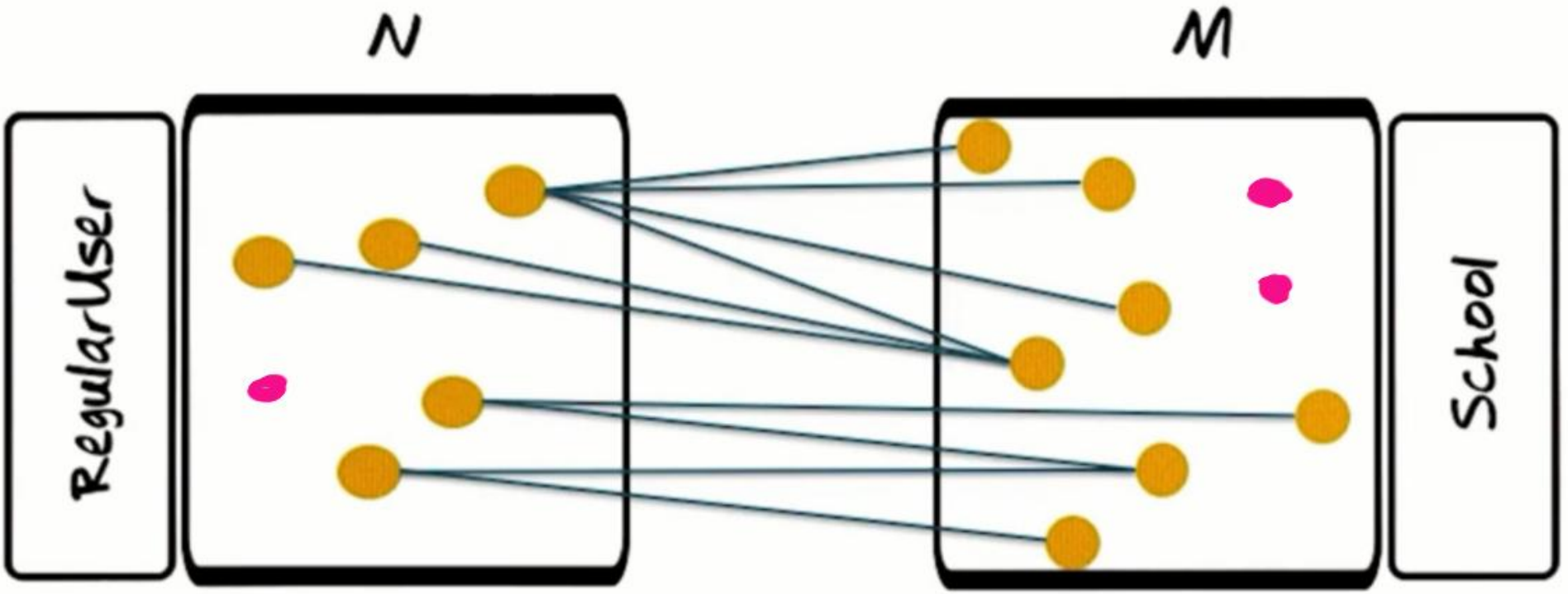
Many-to-one mapping (M:1)

- Opposite of one-to-many
- An entity in *A* is associated with *at most* one entity in *B*
- An entity in *B* is associated with *zero or more* entities in *A*




N-M relationship types
N
M
RegularUser
Schools Attended
School

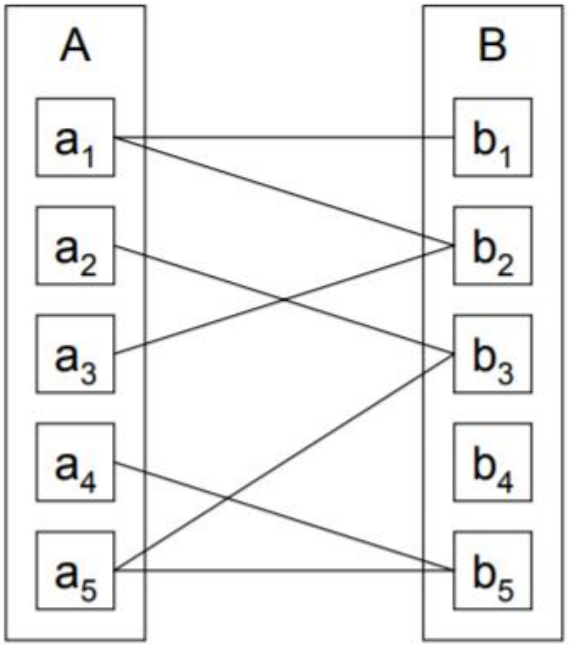
# N-M relationship types



# Mapping Cardinalities (4)

Many-to-many mapping

- An entity in A is associated with *zero or more* entities in B
- An entity in B is associated with *zero or more* entities in A



# Mapping Cardinalities (5)

- Which mapping cardinality is most appropriate for a given relation?
  - Answer depends on what you are trying to model!
  - Could just use many-to-many relations everywhere, but that would be dumb.
- Goal:
  - Constrain the mapping cardinality to most accurately reflect what should be allowed
  - Database can enforce these constraints automatically

# Example: *borrower* relationship between *customer* and *loan*

One-to-one mapping:

- Each customer can have only one loan
- Customers can't share loans

  (e.g. with spouse or business partner)

One-to-many mapping:

- A customer can have multiple loans
- Customers still can't share loans

Many-to-one mapping:

- Each customer can have only one loan
- Customers can share loans

Many-to-many mapping:

- A customer can have multiple loans
- Customers can share loans too

Best choice for *borrower* relation is many-to-many.

***Handles real-world needs!***

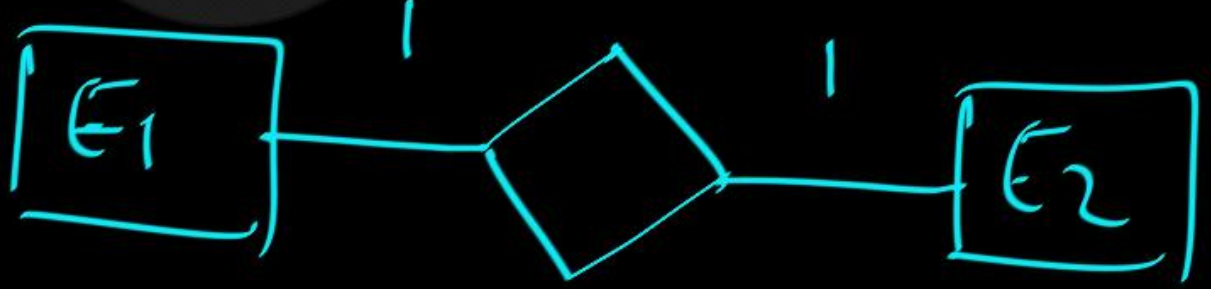
# Mapping Cardinalities:

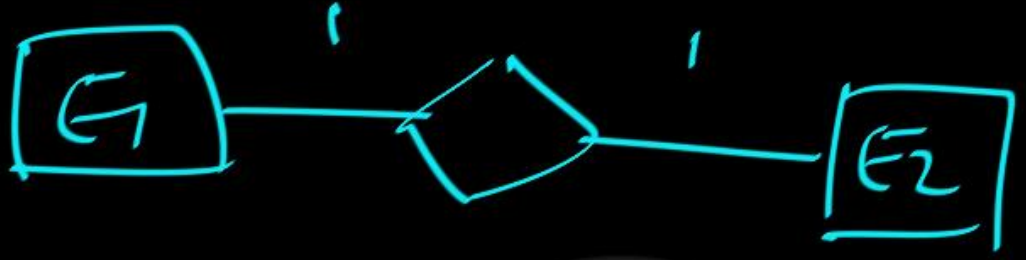
One entity is related to how many entities.

1. 1 – 1
2. 1 – m
3. m – 1
4. m – m

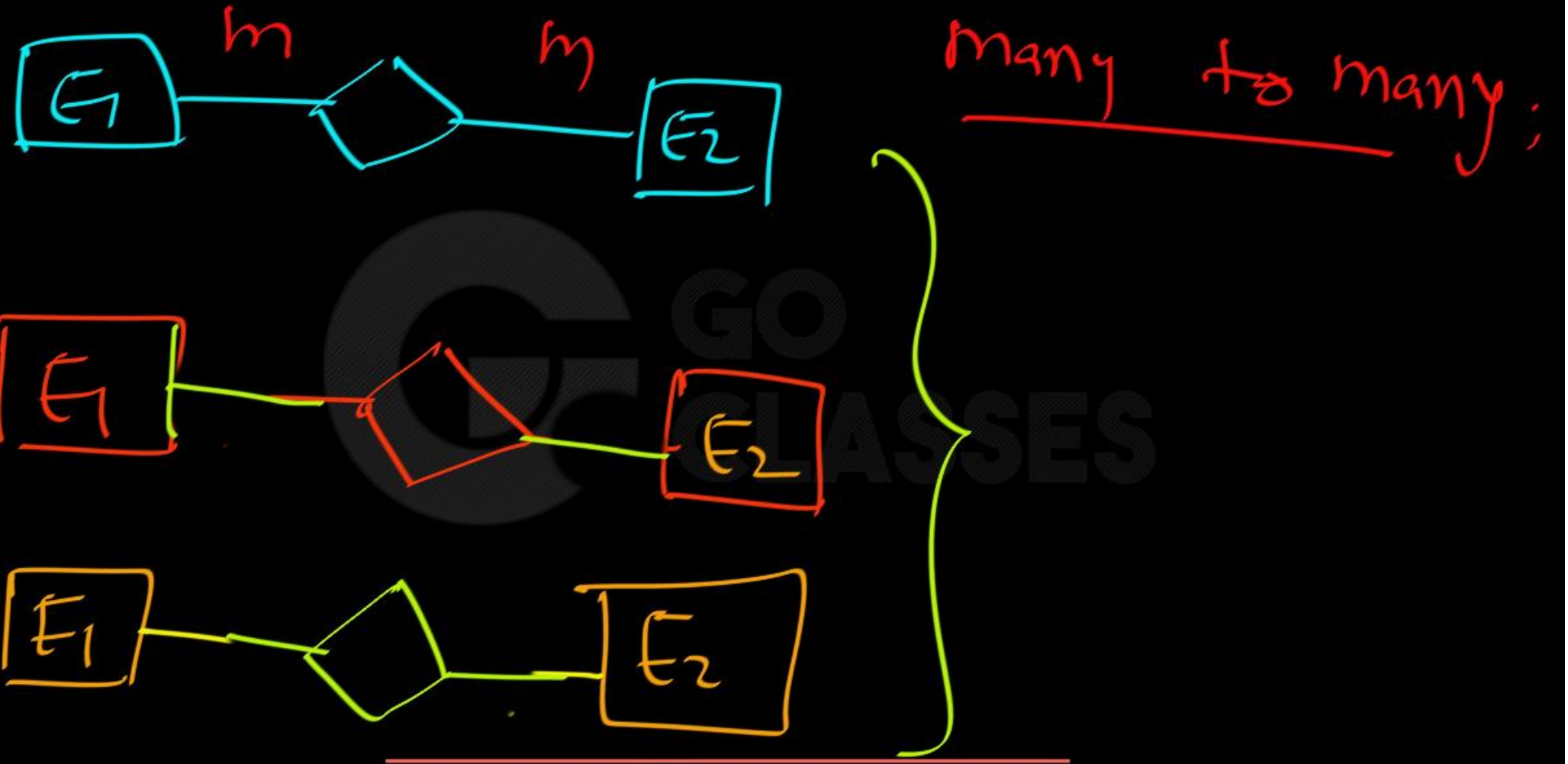
# Representation of Mapping Cardinalities in ER Diagram:

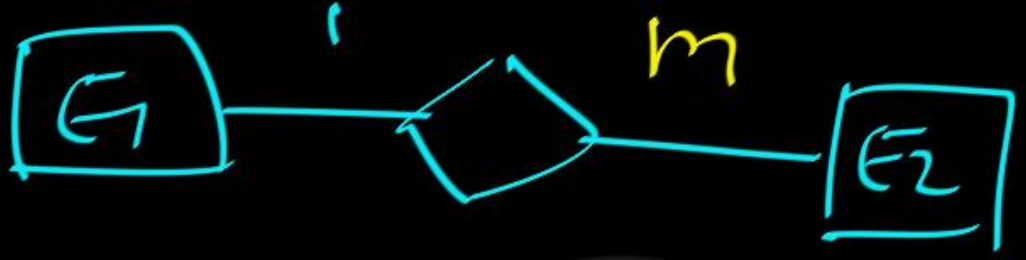
## (1) One-One mapping:






one to one

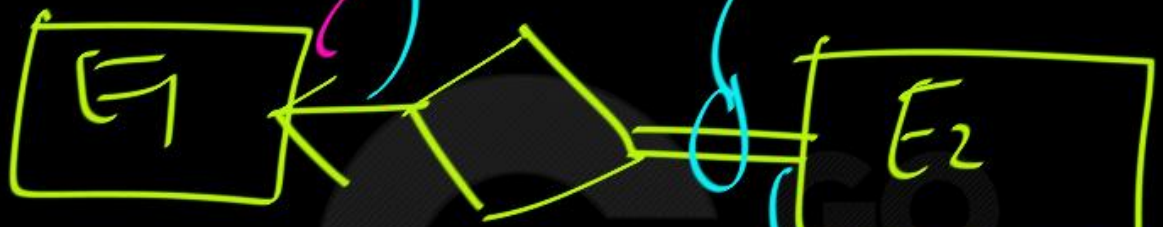


m
n
E1
E2
many to many;
E1
E2
E1
E2




One to many
(from E1 to E2)


Arrow : = 1


many to one
from E2 to E1




Partial Participation

1

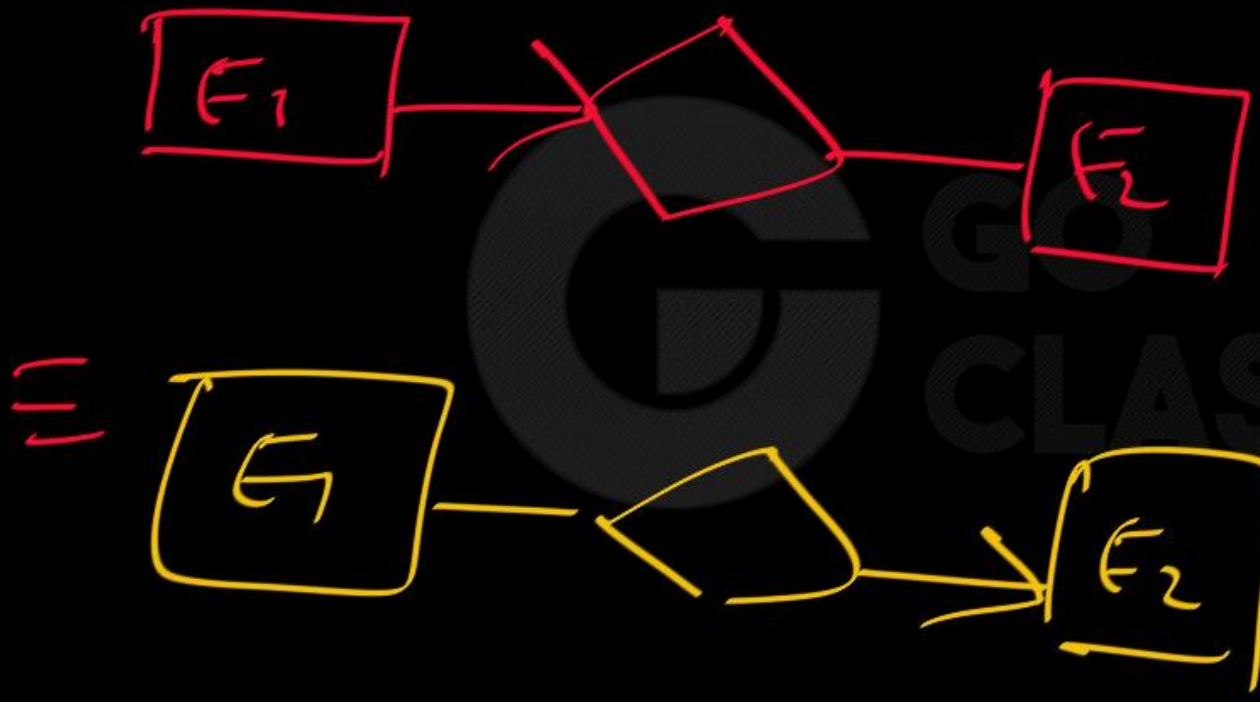
Total Participation

many

No Arrow

from E1 to E2 : one to many

E2 has Total Participation.

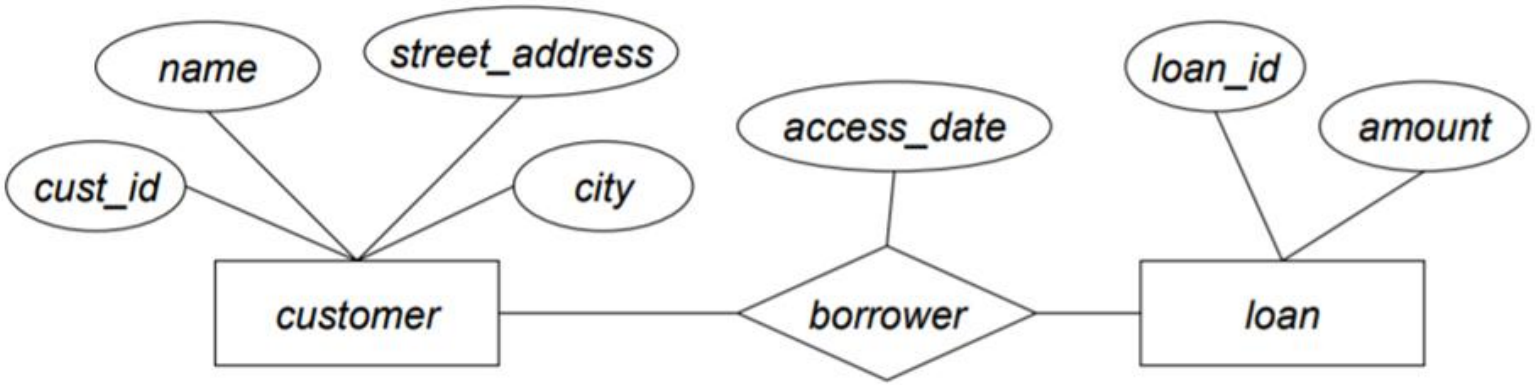



(Confusing Representation)

from E to E2, many to one mapping.

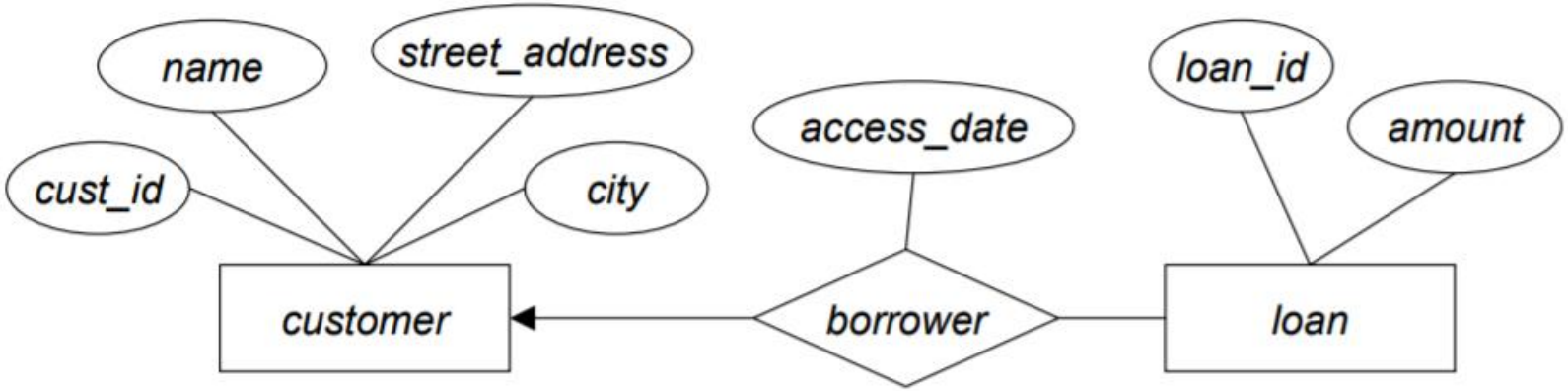
# Diagramming Cardinalities

- In relationship-set diagrams:
  - an arrow towards an entity represents “one”
  - a simple line represents “many”
  - arrow is *always* towards the entity
- Many-to-many mapping between *customer* and *loan*:



# Diagramming Cardinalities (2)

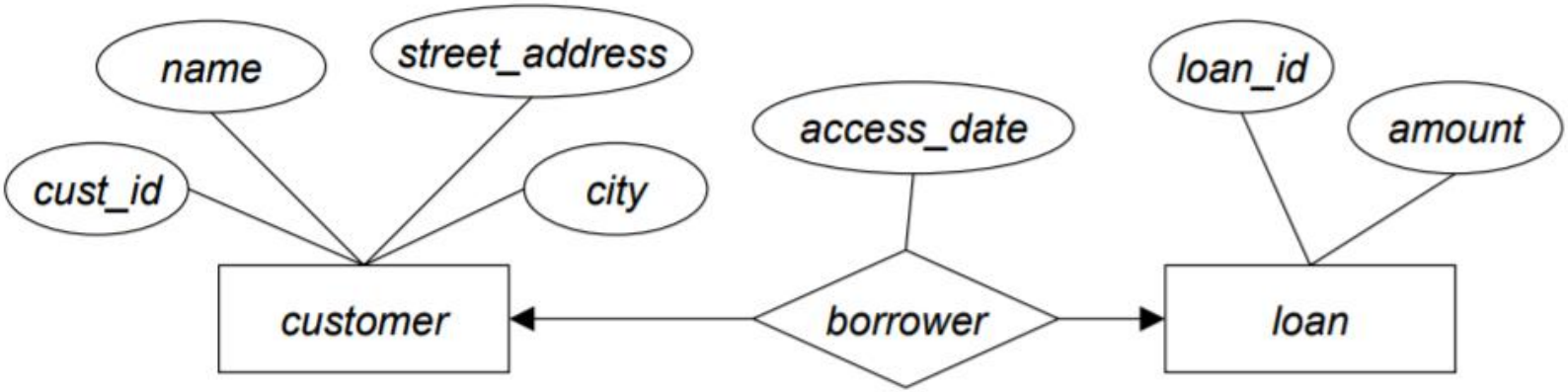
- One-to-many mapping between *customer* and *loan*:




  - Each customer can have multiple loans
  - A loan is owned by exactly one customer
    - (Or "at most one," technically…)

# Diagramming Cardinalities (3)

- One-to-one mapping between *customer* and *loan*:




  - Each customer can have only one loan
  - A loan is owned by exactly one customer

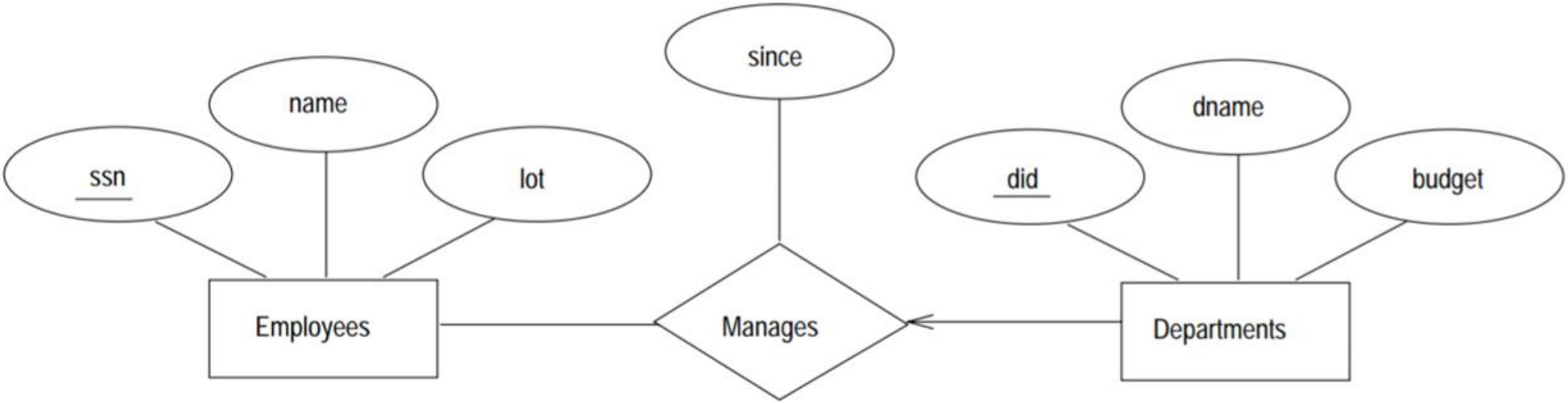



**Figure 2.6** Key Constraint on Manages

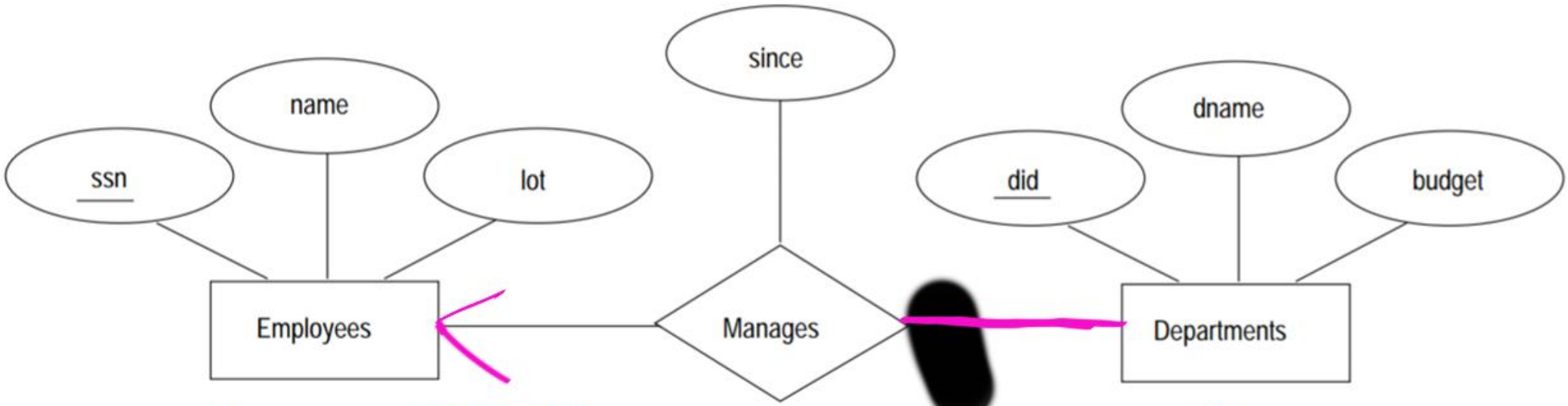



**Figure 2.6** Key Constraint on Manages

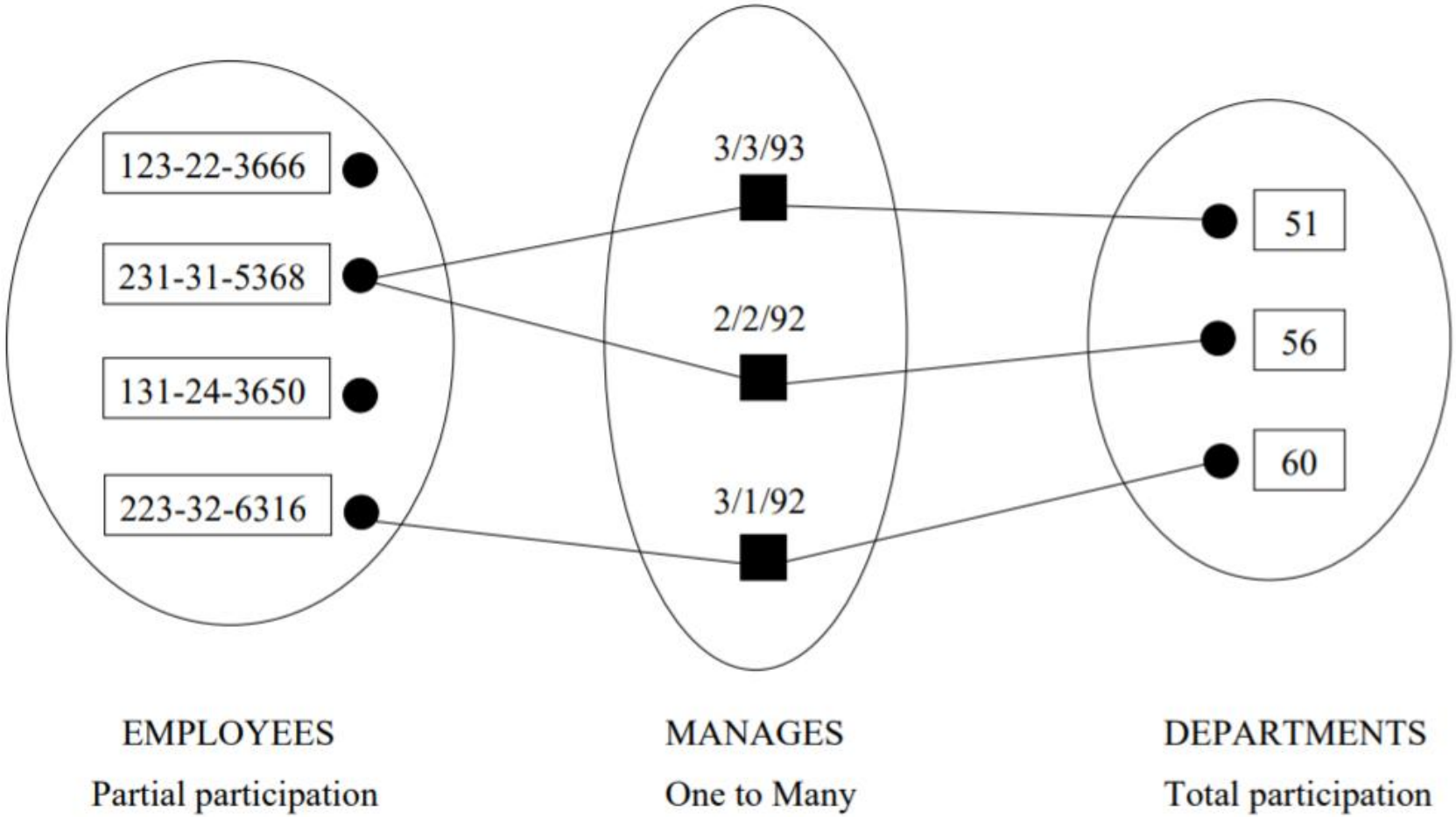
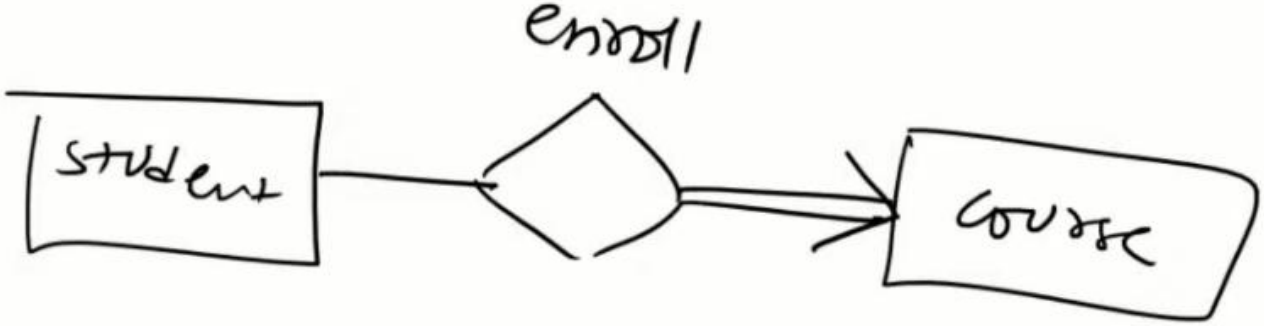



**Figure 2.7** An Instance of the Manages Relationship Set

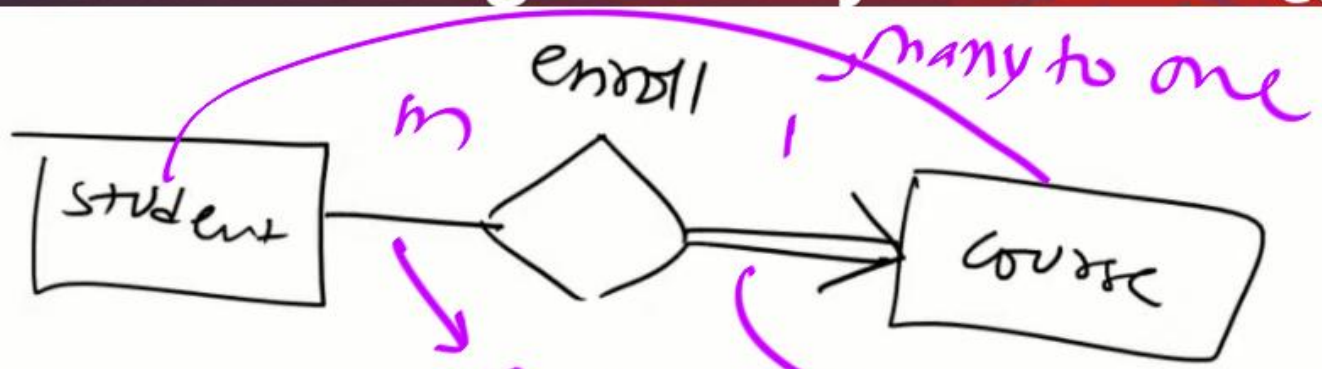
Ex:




Q: which is True?

(1) each course has at least one student
(2) every student has taken at least one course.
(3) a student can take 3 courses.
(4) a course can have 3 student.

Eg:




Q: which is True?

(1) each course has at least one student

(2) every student has taken at least one course.

(3) a student can take 3 courses.

(4) a course can have 3 student.

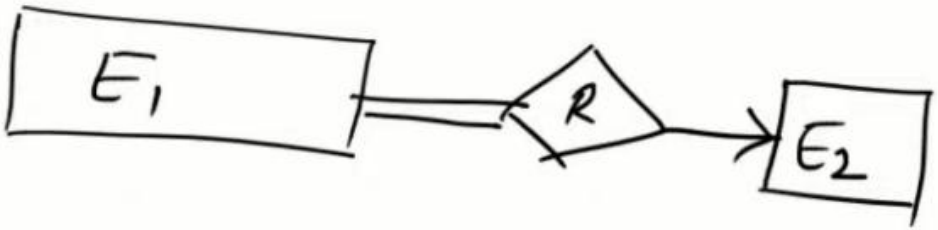
from Course to student : one to many mapping.

from students to course (many to one mapping)

a student can enroll in at most one course.

a course can have any no. of student.

Q:



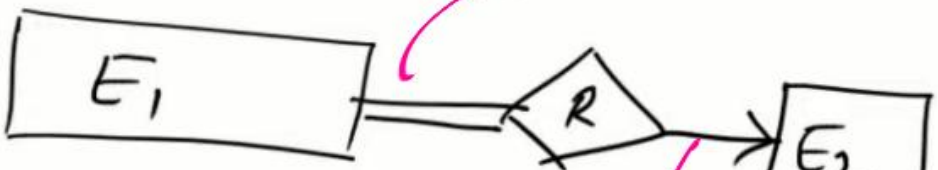

A. Every entity of $E_1$ is associated with exactly one entity of $E_2$

B. Every " " $E_1$ " " " at most one entity of $E_2$.

C. " " " " $E_1$ " at least one entity of $E_2$

D.

Q: (mcq)




total Participation

Partial Participation

A. Every entity of $E_1$ is associated with exactly one entity of $E_2$

B. Every " " $E_1$ " " " at most one entity of $E_2$.

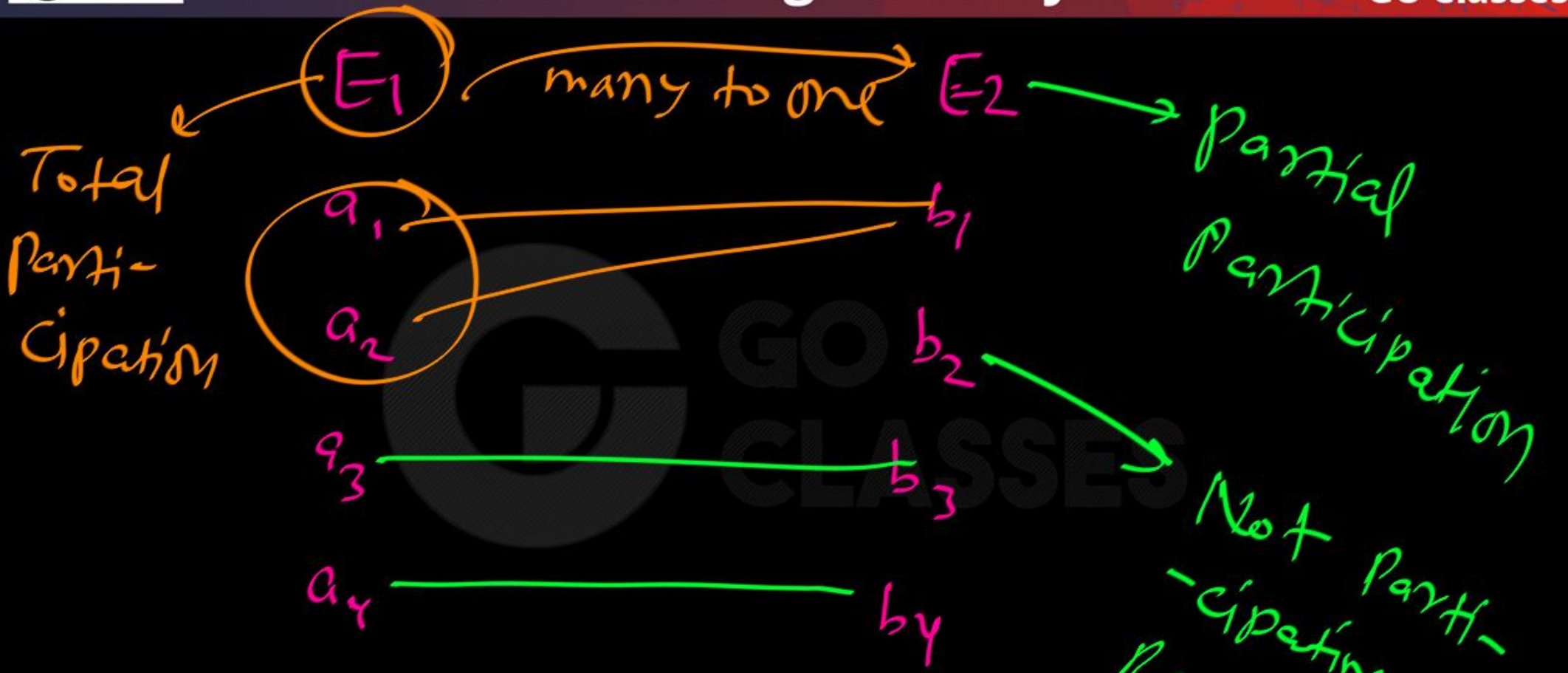
C. " " " " $E_1$ " at least one entity of $E_2$

D " " " " " " " $E_1$

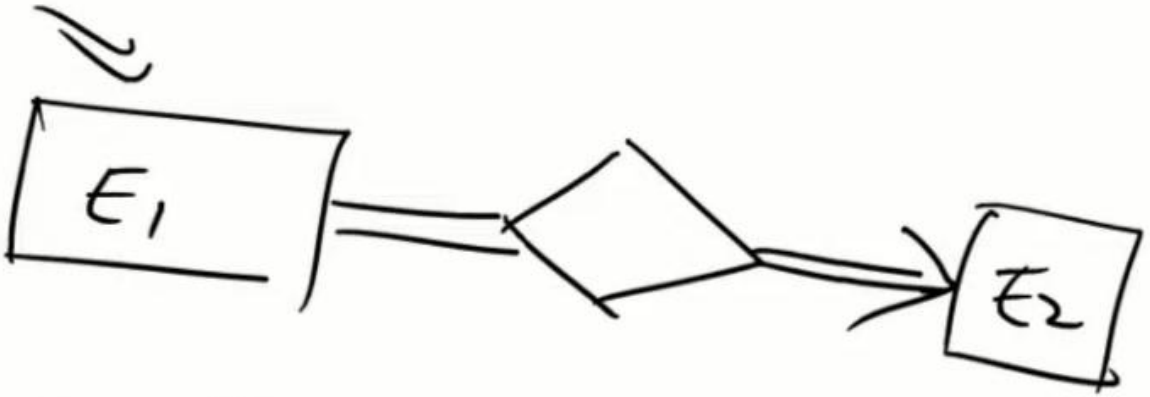
from $E_1$ to $E_2$ : many to one mapping

And $E_1$ has Total Participation

↳ Every entity of $E_1$ is associated with Exactly one entity of $E_2$.


E1
many to one
E2
Total Participation
Partial Participation
a1
a2
a3
ax
b1
b2
b3
by
Not Participating in Relationship

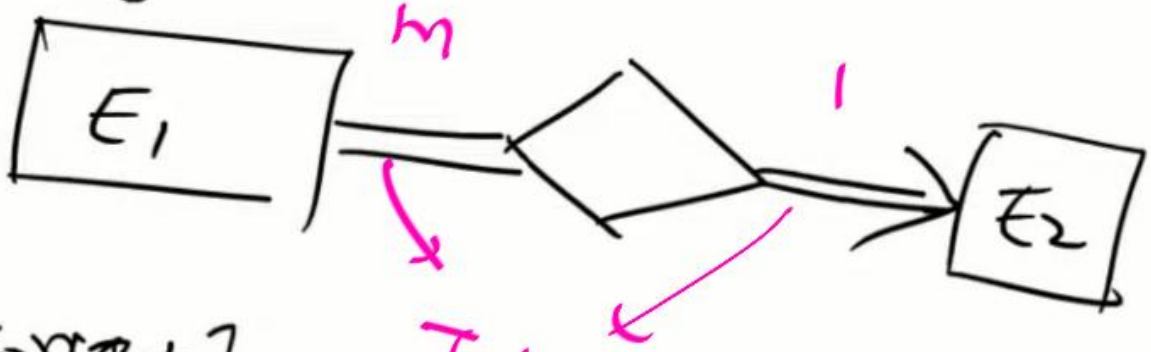
Q:




Which is correct?

1. $|E_1| \geq |E_2|$
2. $|E_1| > |E_2|$
3. $|E_1| < |E_2|$
4. $|E_1| = |E_2|$
5. $|E_1| \leq |E_2|$

Q: (MCQ)



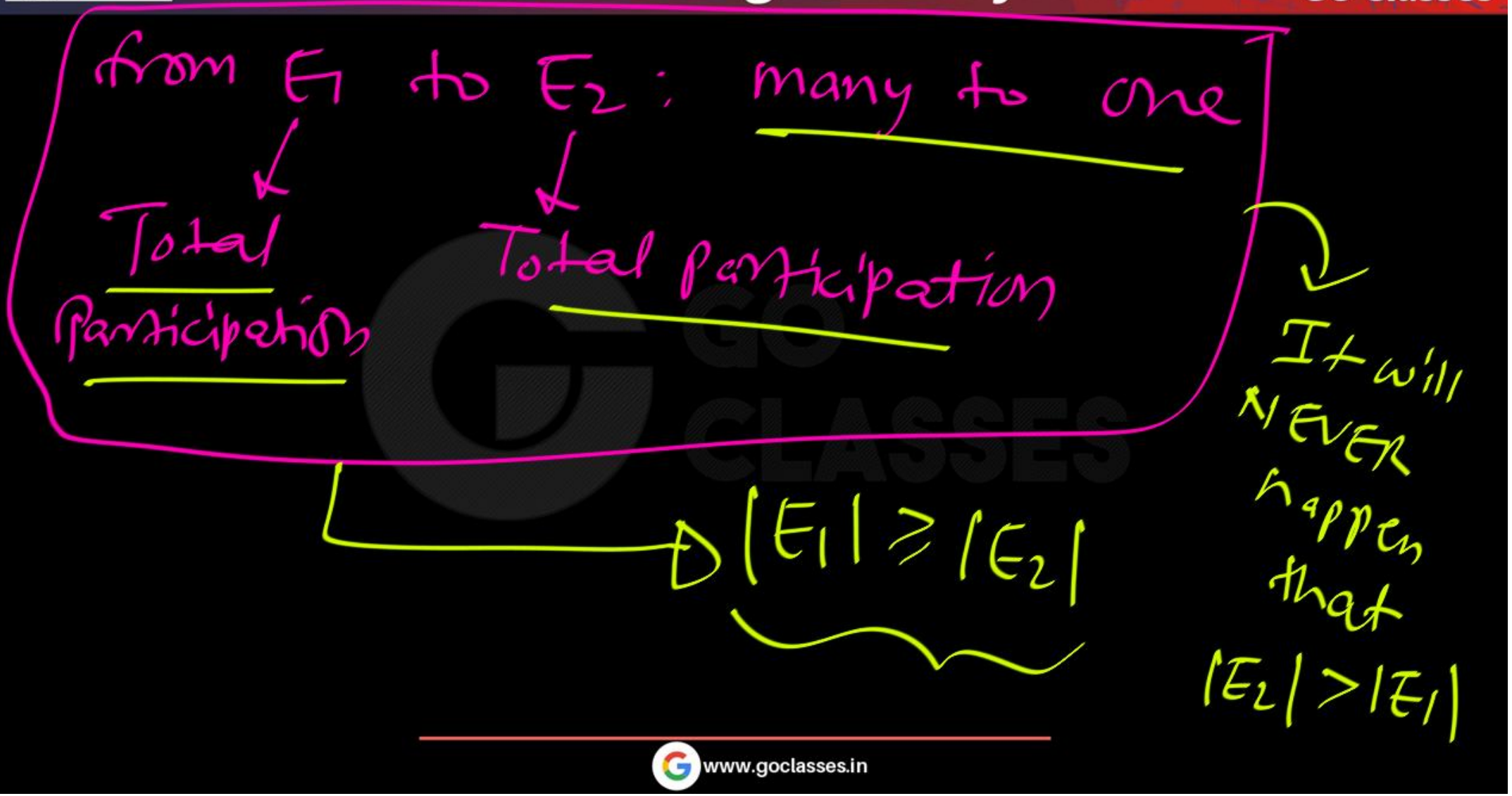

Total Participation

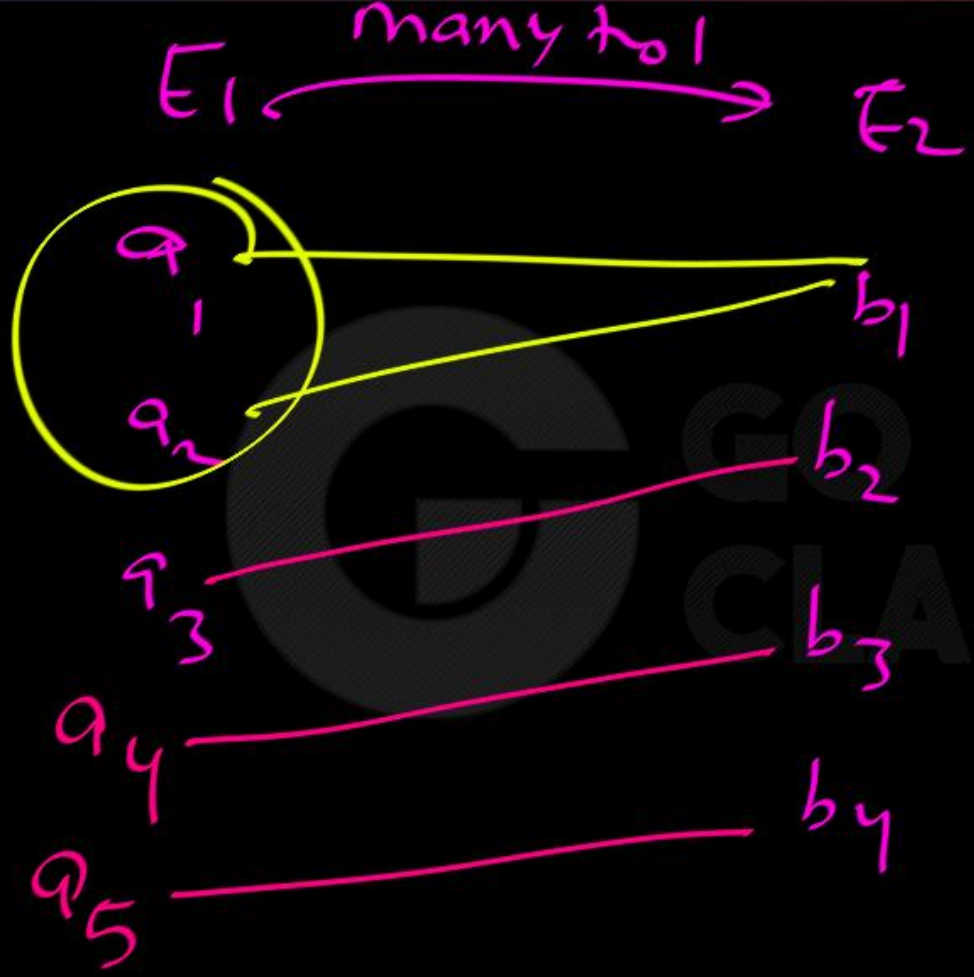
no. of entities in $E_2$

Cardinality of $E_2$

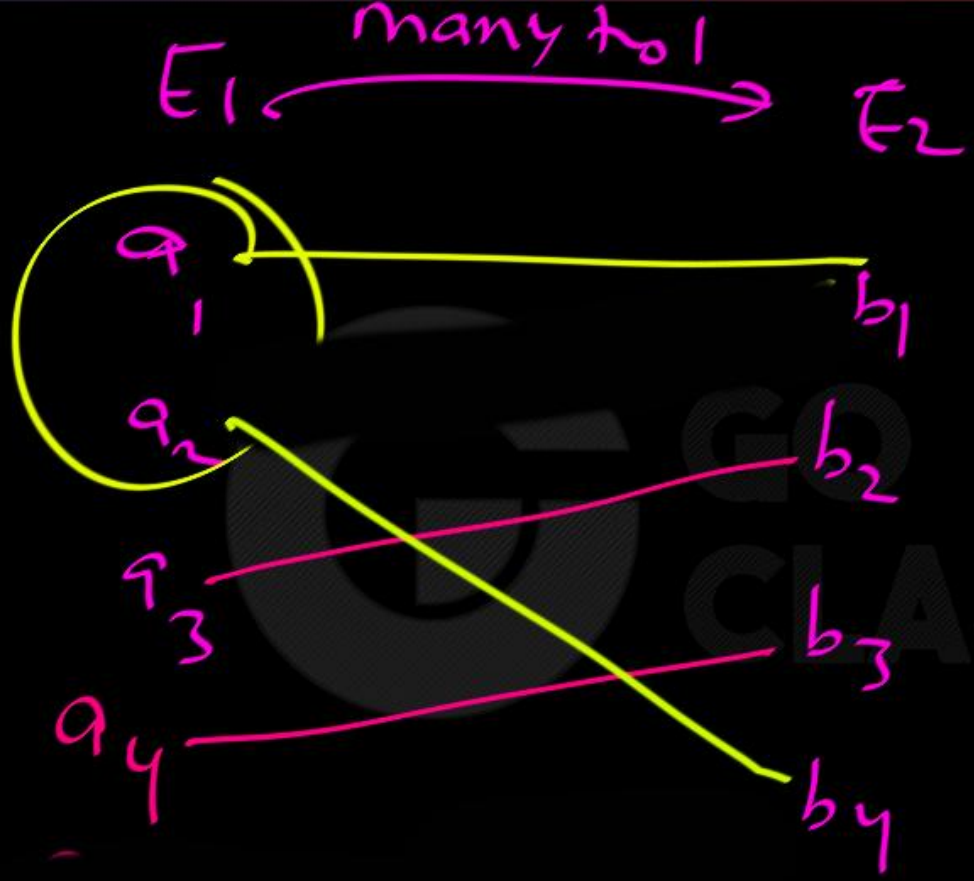
Which is correct?

1. $|E_1| \geq |E_2|$
2. $|E_1| > |E_2|$
3. $|E_1| < |E_2|$
4. $|E_1| = |E_2|$
5. $|E_1| \leq |E_2|$


from E1 to E2 : many to one
Total Participation
Total participation
It will NEVER happen that
|E2| > |E1|
|E1| ≥ |E2|




$|E_1| \geq |E_2|$




$|E_1| \geq |E_2|$

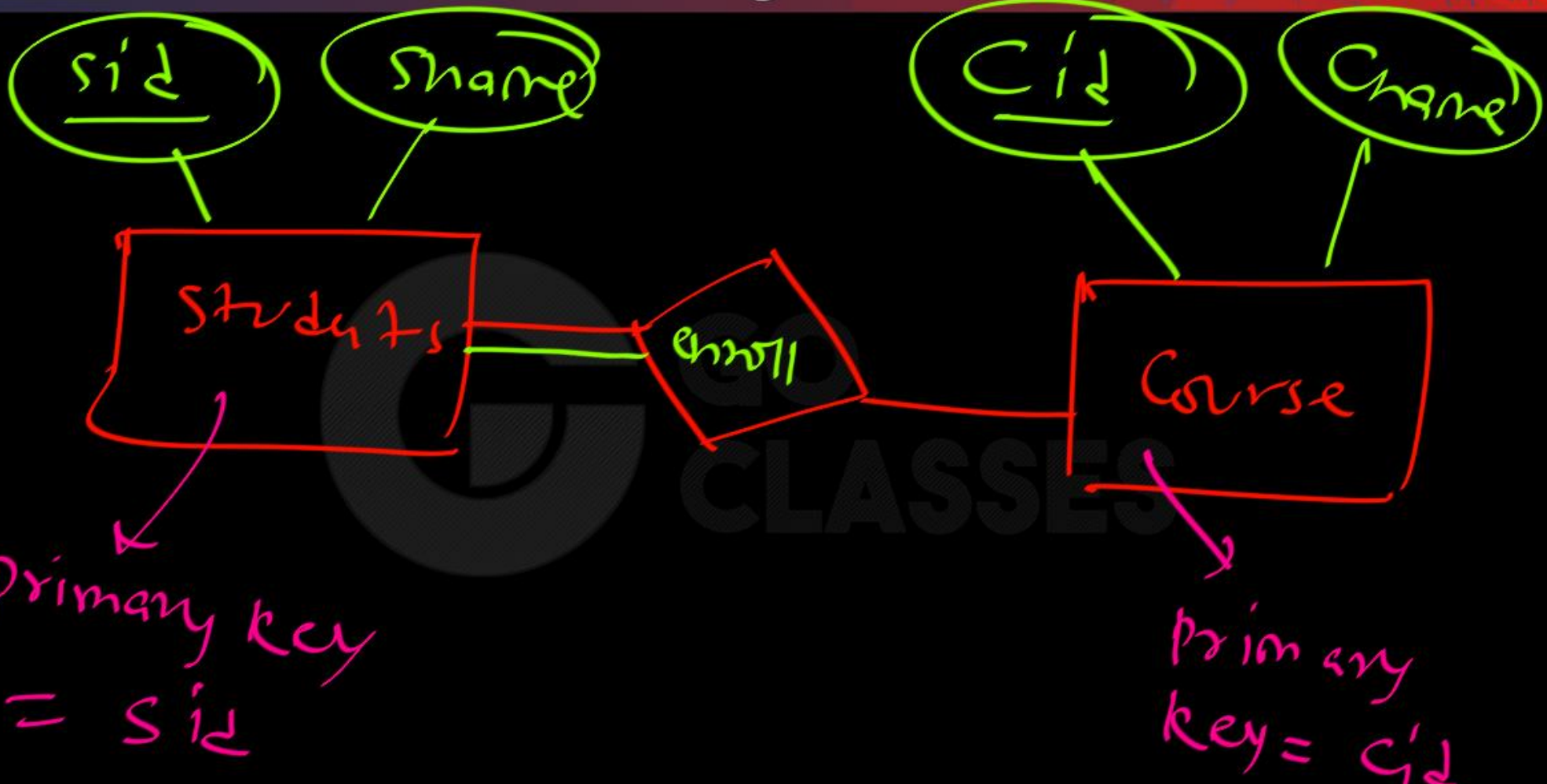
# Review

- Database schema design is a central step of database application development
  - Can make or break the application!
  - Requires familiarity with good schema design
  - Requires familiarity with the problem domain!
- Entity-relationship model as a design tool
  - Can diagram a conceptual schema
  - Can represent relationships between entities
  - Can represent certain constraints on data

# Primary key / Candidate key for Relationship:

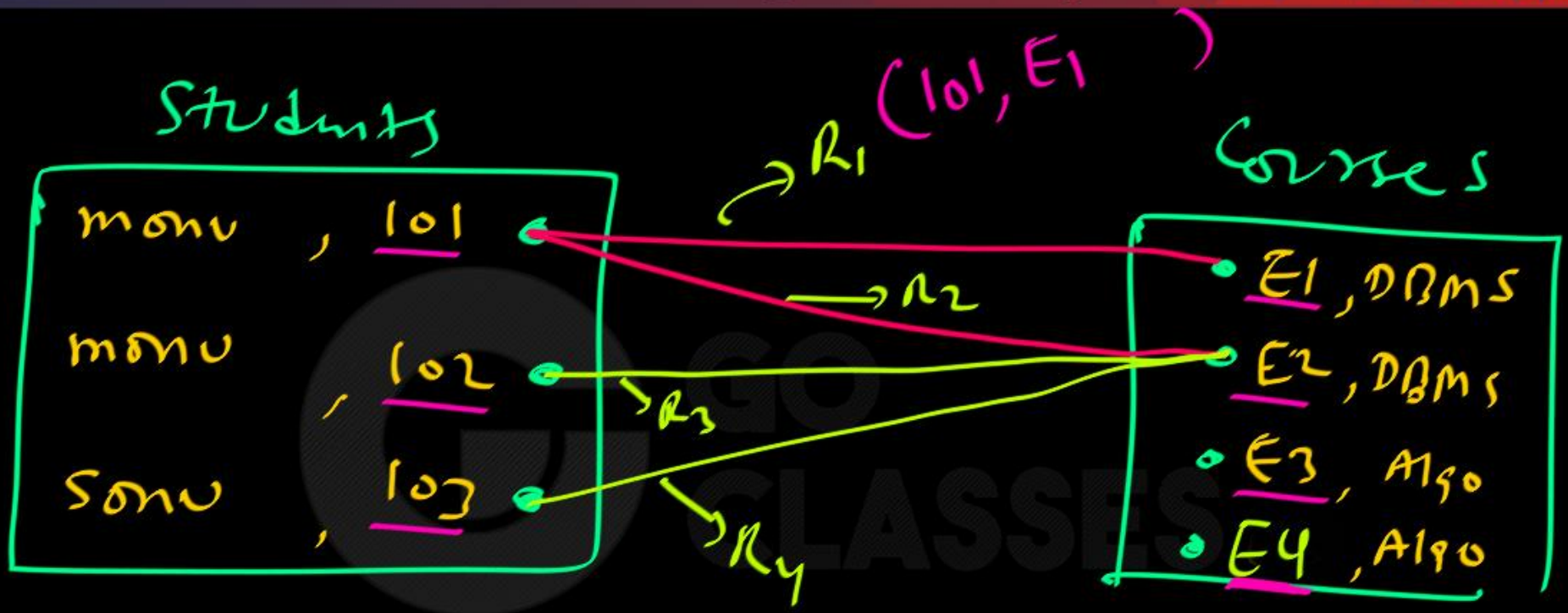
# Keys and Relationship-Sets

- Need to be able to distinguish between relationships in a relationship-set as well
  - Relationships aren't distinguished by their descriptive attributes
  - Might not even have descriptive attributes
- Relationships are identified by the entities participating in the relationship
  - Relationship instances are uniquely identified by primary keys of participating entities


Sid
Sname
Cid
Cname
Students
Enroll
Course
Primary Key = Sid
Primary key = Cid

To Describe a relationship (Association) b/w two entities:

Primary keys of both entity sets.



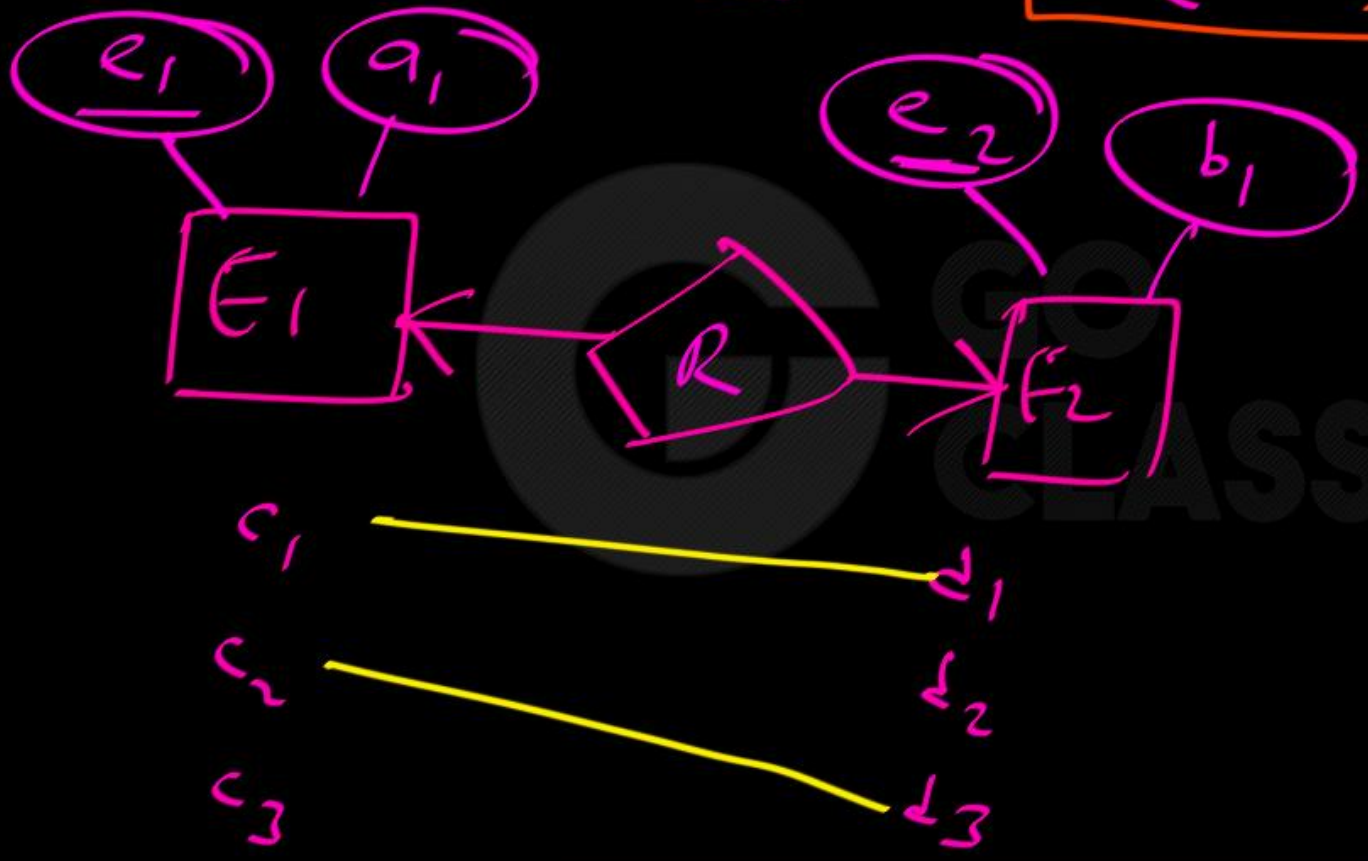

$R_2(101, E_2)$ ; $R_3(102, E_2)$, $R_4(103, E_2)$

Relationship set "Enroll" : (sid, cid)

(, primary key of Enroll : (sid, cid))

many-many mapping b/w student, course

# 1-1 Relationship:




$R(e_1, e_2)$

## Two relationships:

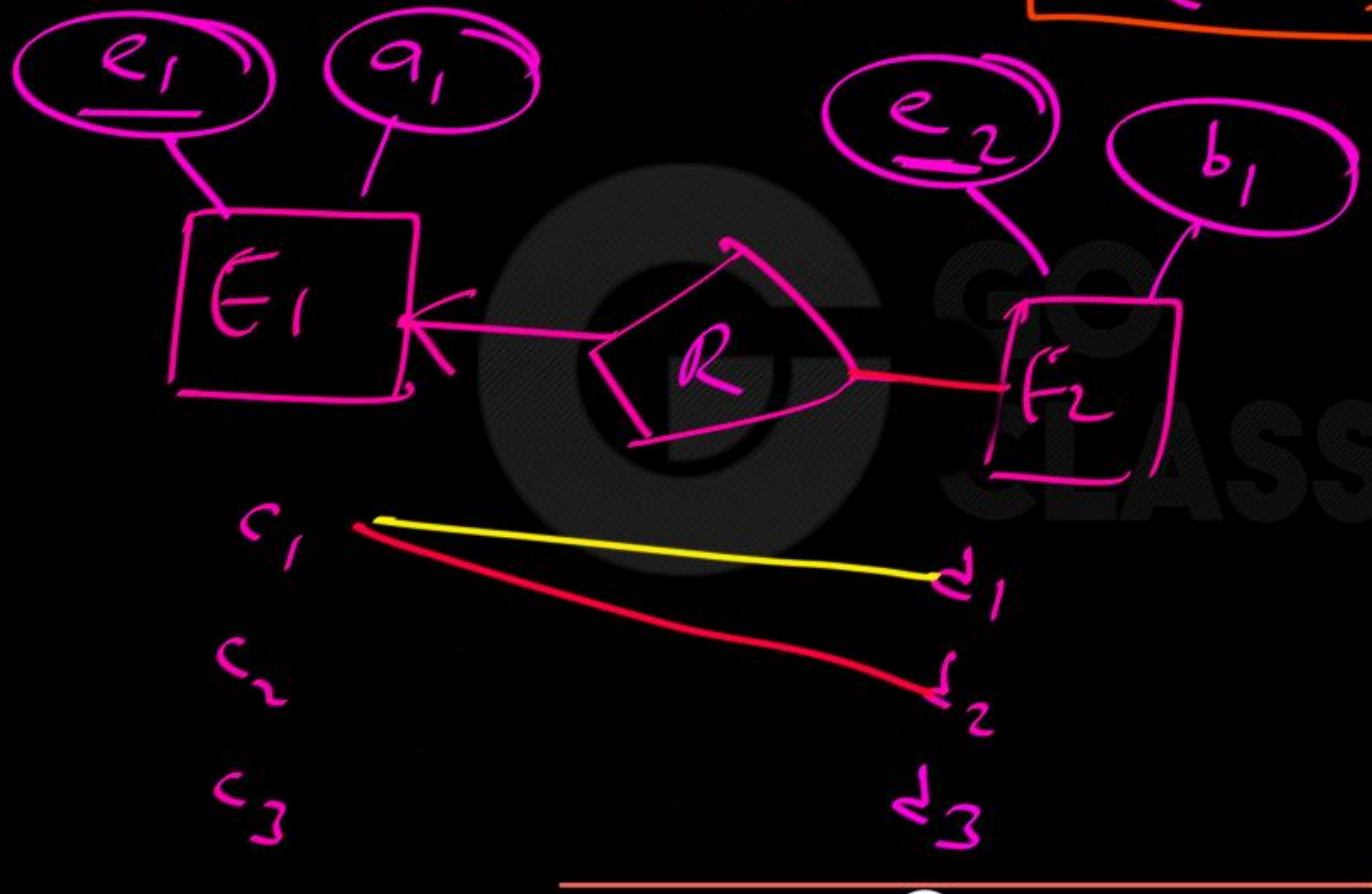
$(c_1, d_1)$

$(c_2, d_3)$

Primary key of R : $e_1$ or $e_2$

# 1-m Relationship:

$R(e_1, e_2)$




Two relationships:

$(c_1, d_1)$
$(c_1, d_2)$

Primary key of R : $e_2$

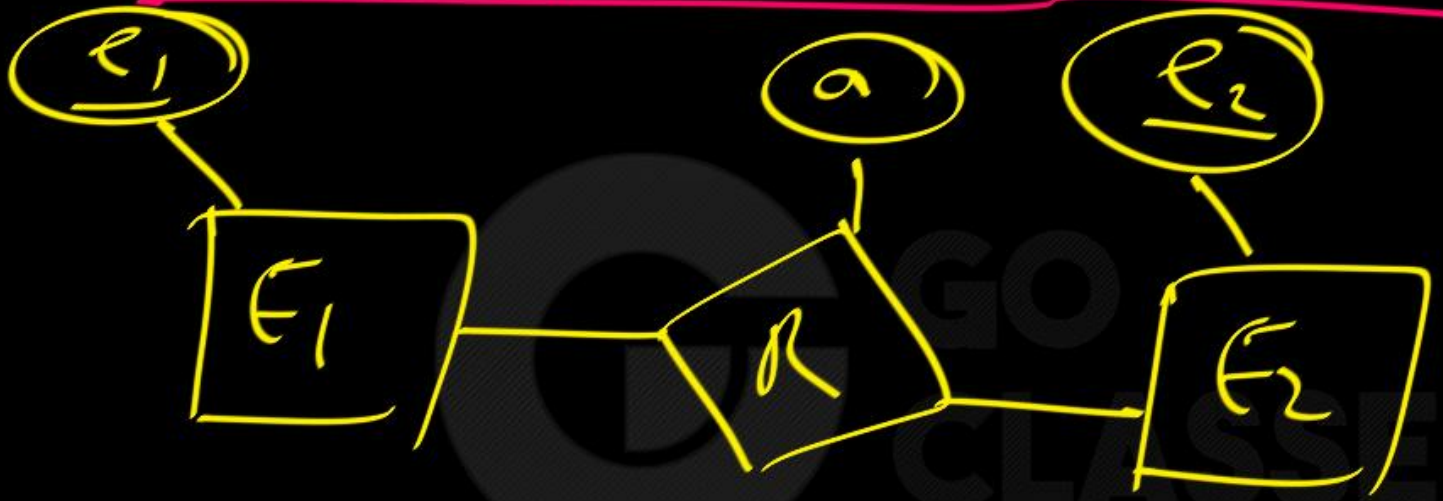
# Keys and Relationship-Sets (2)

- $R$ is a relationship-set with no descriptive attributes
- Entity-sets $E_1, E_2, \ldots, E_n$ participate in $R$
- $primary\_key(E_i)$ denotes set of attributes in $E_i$ that represent the primary key of $E_i$
- A relationship instance in $R$ is identified by

  $primary\_key(E_1) \cup primary\_key(E_2) \cup \ldots \cup primary\_key(E_n)$

  – This is a superkey

# Relationships with simple valued attribute:




$R(e_1, e_2, a)$

1-1 : Primary key of R : $e_1$ or $e_2$

1-m : " " " " : $e_2$

m-1 : " " " " : $e_1$

m-n : " " " " : $e_1 e_2$

1-1 : Candidate key of R : $e_1$ , $e_2$

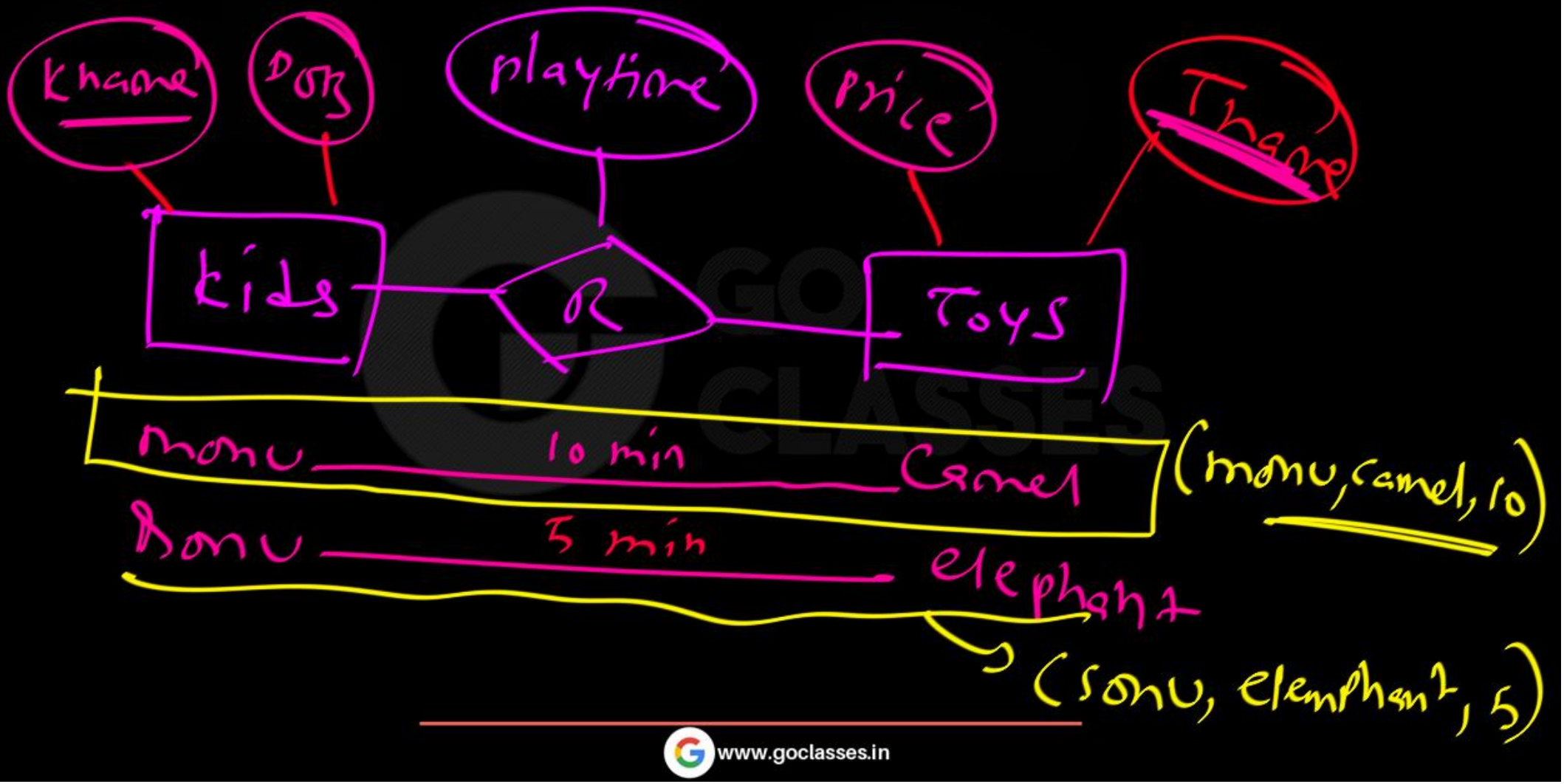
1-m : Candidate " " " : $e_2$

m-1 : " " " : $e_1$

m-n : " " " : $e_1 e_2$

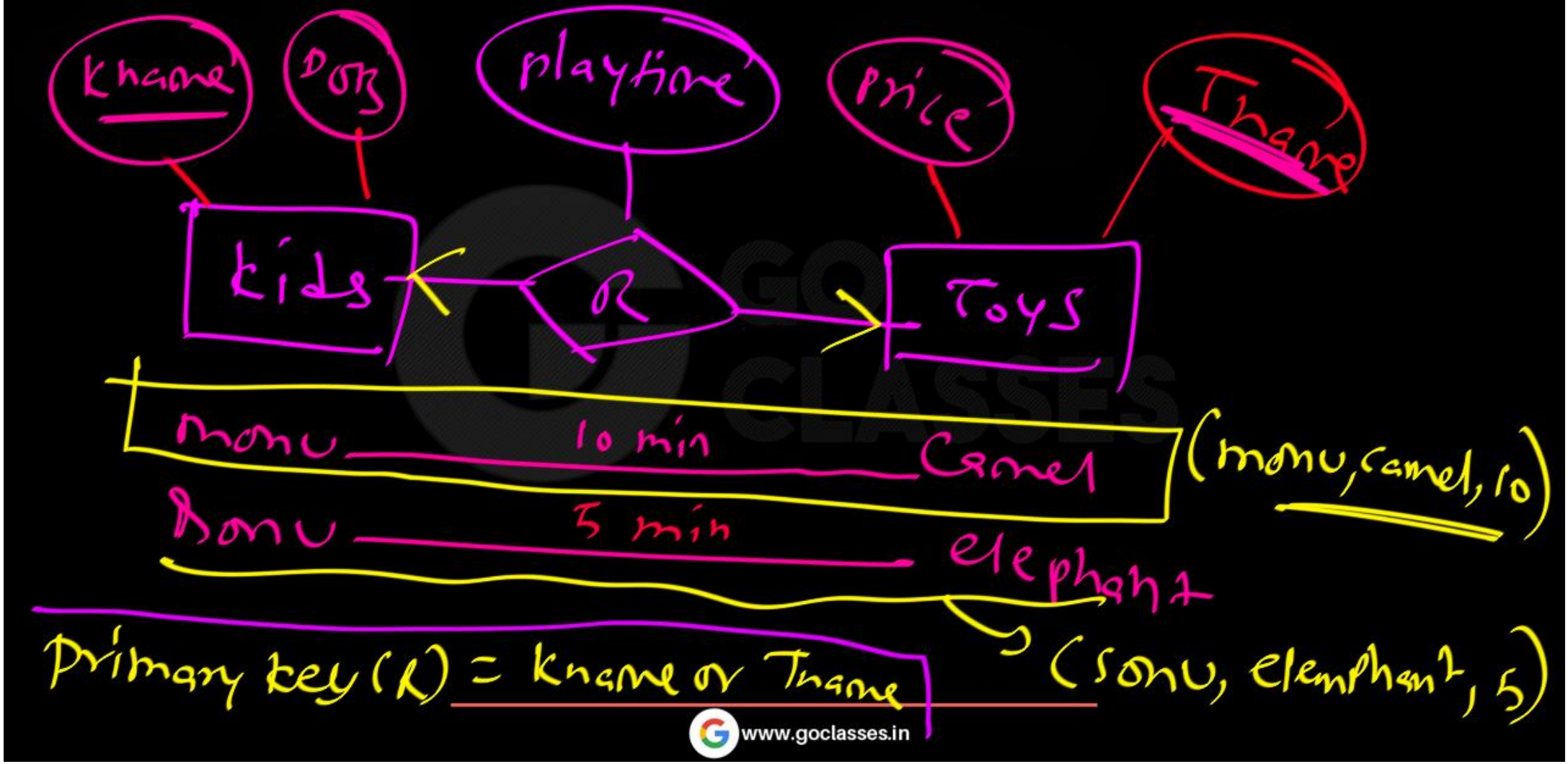
# Database Management System





# Database Management System





# Keys and Relationship-Sets (3)

- If $R$ has descriptive attributes $\{a_1, a_2, \ldots\}$, a relationship instance is described by:

  $primary\_key(E_1) \cup primary\_key(E_2) \cup \ldots \cup primary\_key(E_n) \cup \{ a_1, a_2, \ldots \}$

  – Not a minimal superkey

# Relationship-Set Primary Keys

- What is the primary key for a binary relationship-set?
  - Must also be a candidate key
  - Depends on the mapping constraints!
- Relationship-set *R*, involving entity-sets *A* and *B*
  - If mapping is many-to-many, primary key is: *primary_key*(*A*) ∪ *primary_key*(*B*)
  - *Any* entity's primary-key values can appear multiple times in *R*
  - Needs both entity-sets' primary key attributes to uniquely identify relationship instances

# Relationship-Set Primary Keys (2)

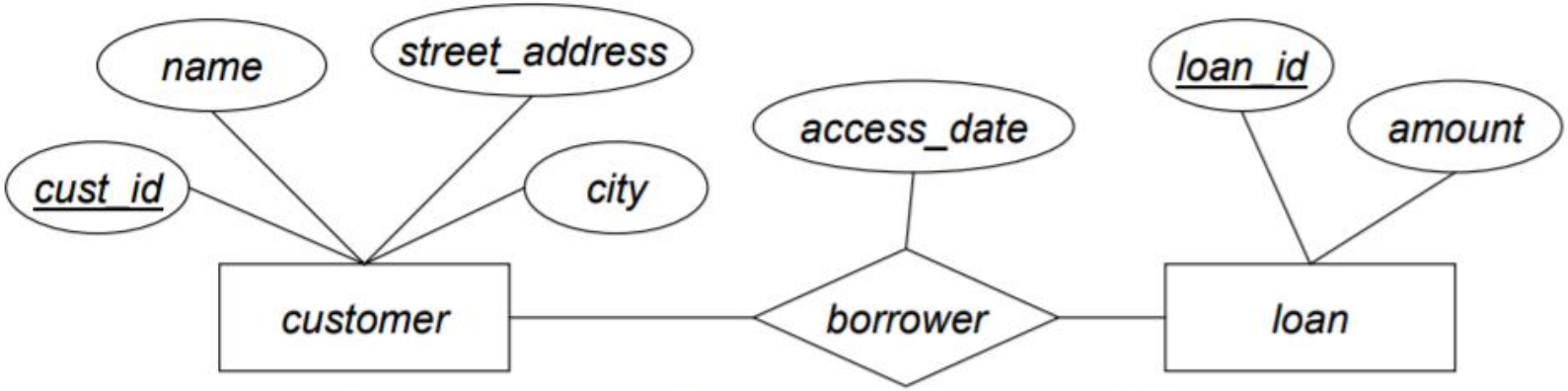
- Relationship-set *R*, involving entity-sets *A* and *B*
- If mapping is one-to-many, *primary_key*(*B*) is primary key of relationship-set
  - Entities in *B* associated with *at most* one entity in *A*
  - *primary_key*(*A*) values can appear multiple times in relationship instances
  - Relationships uniquely identified by *primary_key*(*B*)
- Many-to-one is exactly the opposite
  - *primary_key*(*A*) uniquely identifies relationships

# Relationship-Set Primary Keys (3)

- Relationship-set *R*, involving entity-sets *A* and *B*
- If mapping is one-to-one, either entity-set's primary key is primary key of *R*
  - Entities in *A* associated with *at most* one entity in *B*
  - Entities in *B* associated with *at most* one entity in *A*
  - Each entity's primary key value can appear only once in *R*
- For one-to-one mapping, *primary_key*(*A*) and *primary_key*(*B*) are both candidate keys
  - Enforce this constraint in implementation schema

# Example

- What is the primary key for *borrower* ?




- *borrower* is a many-to-many mapping
  - Relationship instances described by (*cust_id*, *loan_id*, *access_date*)
  - Primary key for relationship-set is (*cust_id*, *loan_id*)

# Participation Constraints

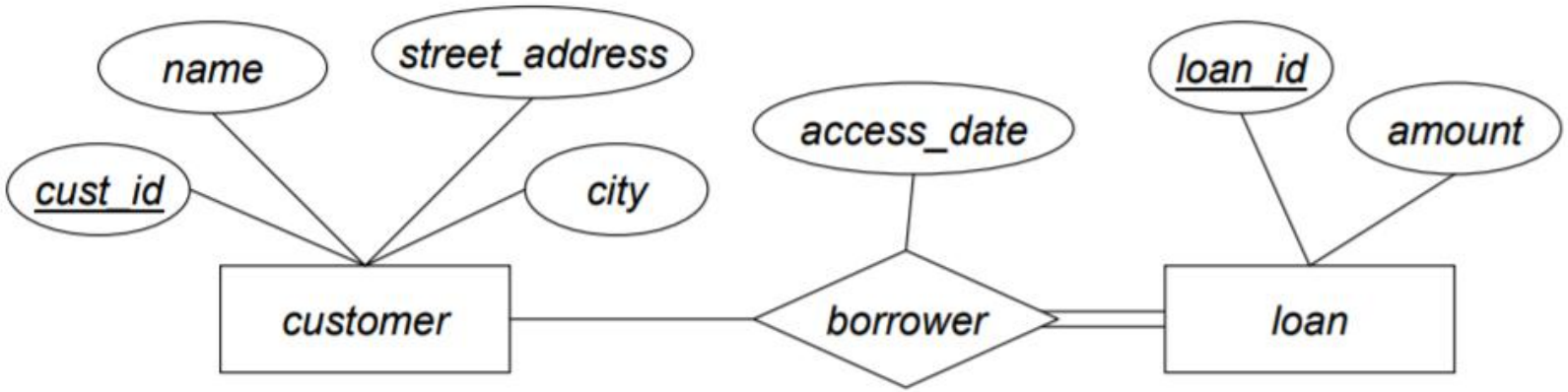
- Given entity-set *E*, relationship-set *R*
  - How many entities in *E* participate in *R* ?
- If every entity in *E* participates in at least one relationship in *R*, then:
  - *E*'s participation in *R* is total
- If only some entities in *E* participate in relationships in *R*, then:
  - *E*'s participation in *R* is partial

# Participation Constraints (2)

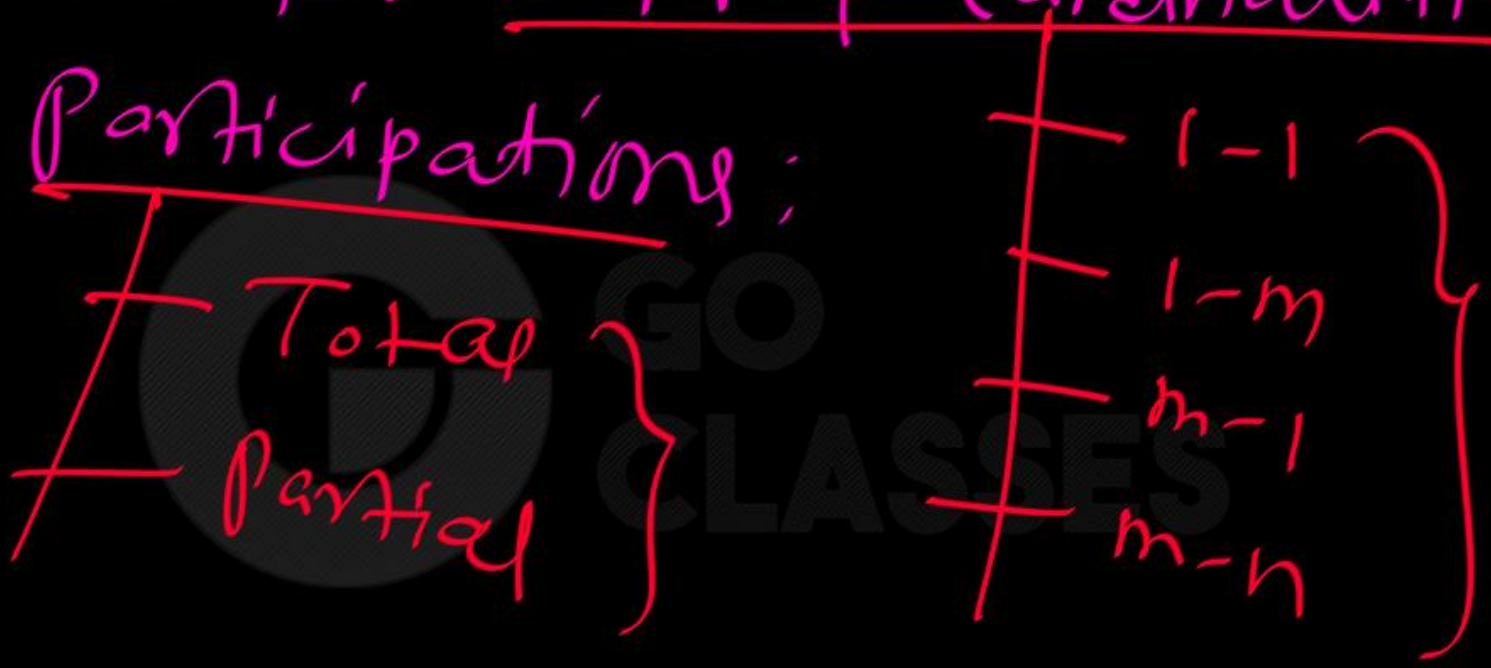
- Example: *borrower* relationship between *customer* and *loan*
- A customer might not have a bank loan
  - Could have a bank account instead
  - Could be a new customer
  - Participation of *customer* in *borrower* is partial
- Every loan definitely has at least one customer
  - Doesn't make any sense not to!
  - Participation of *loan* in *borrower* is total

# Diagramming Participation

- Can indicate participation constraints in entity-relationship diagrams
  - Partial participation shown with a single line
  - Total participation shown with a double line



# Notations for mapping Cardinalities and Participations:



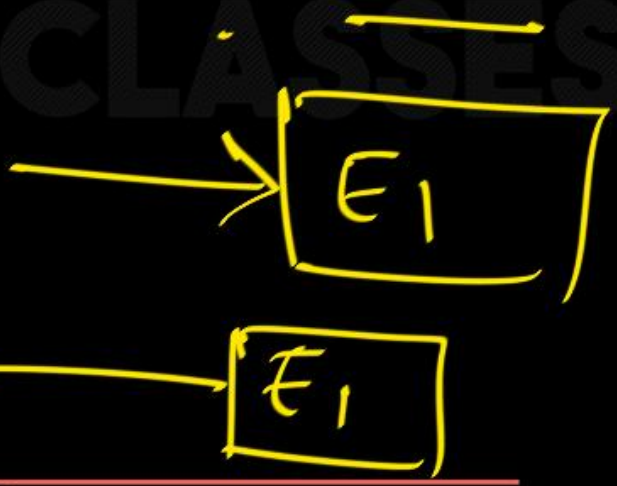
## Most Common Notation:

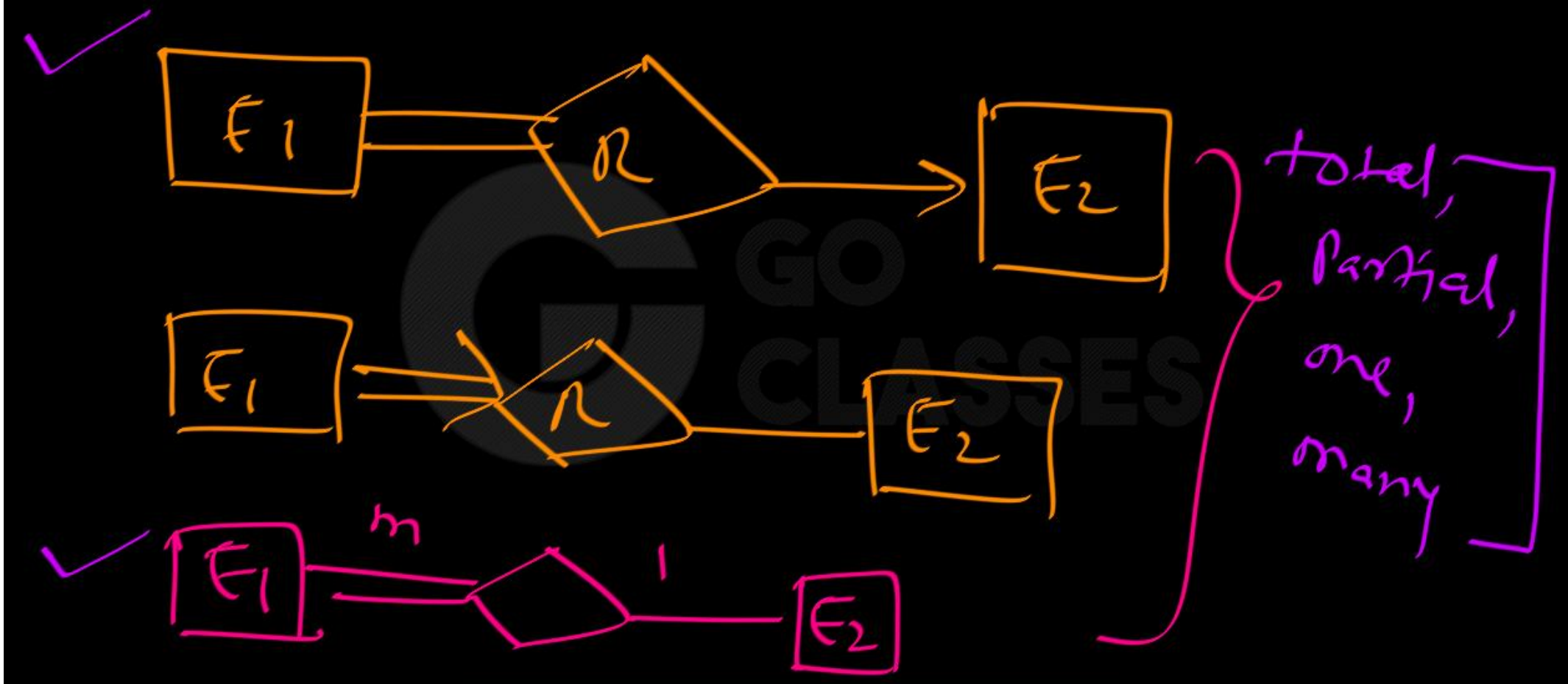
Total Participation : ——

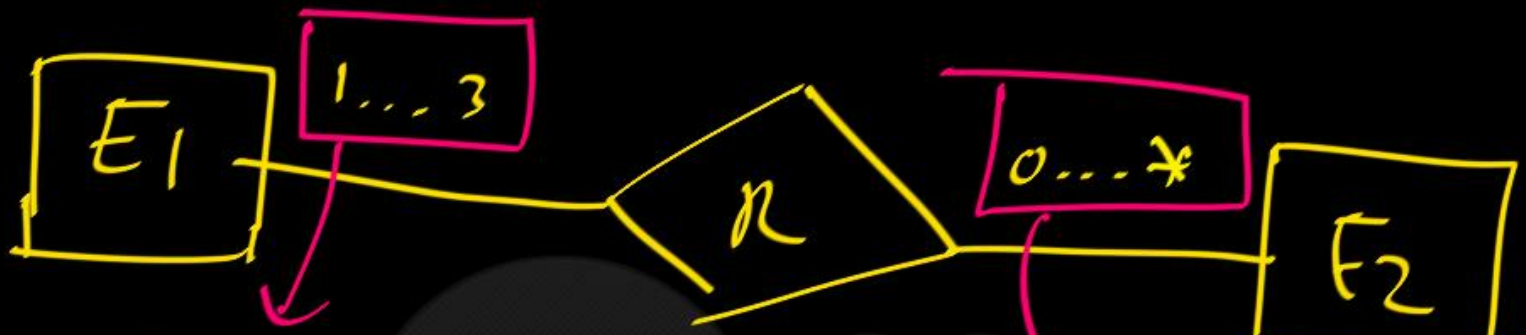
Partial " :

One :

Many :




E1
R
E2
total,
Partial,
one,
many
E1
R
E2
E1
m
1
E2

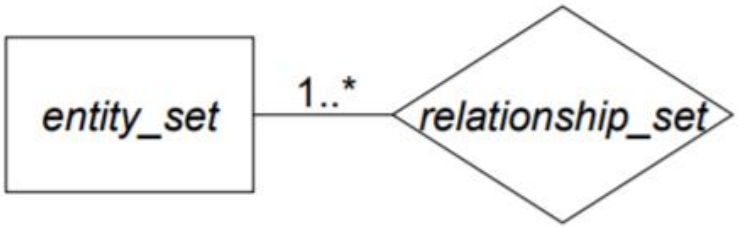



a entity of E1 can associate to "1" to at most 3 entities of E2

a entity of E2 can associate with 0 or more entities of E1

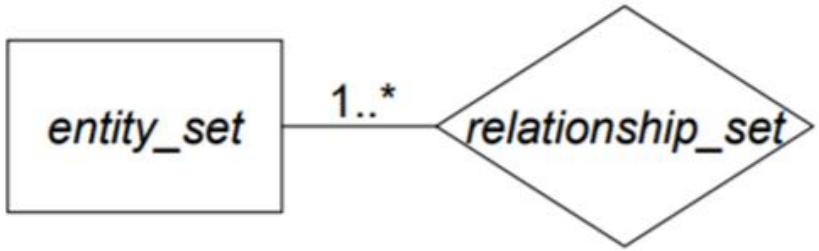
# Numerical Constraints

- Can state numerical participation constraints
  - Specifies how many different relationship instances each entity in the entity-set can participate in
  - Indicated on link between entity and relationship
- Form: lower..upper
  - * means “unlimited”
  - 1..* = one or more
  - 0..3 = between zero and three, inclusive
  - etc.



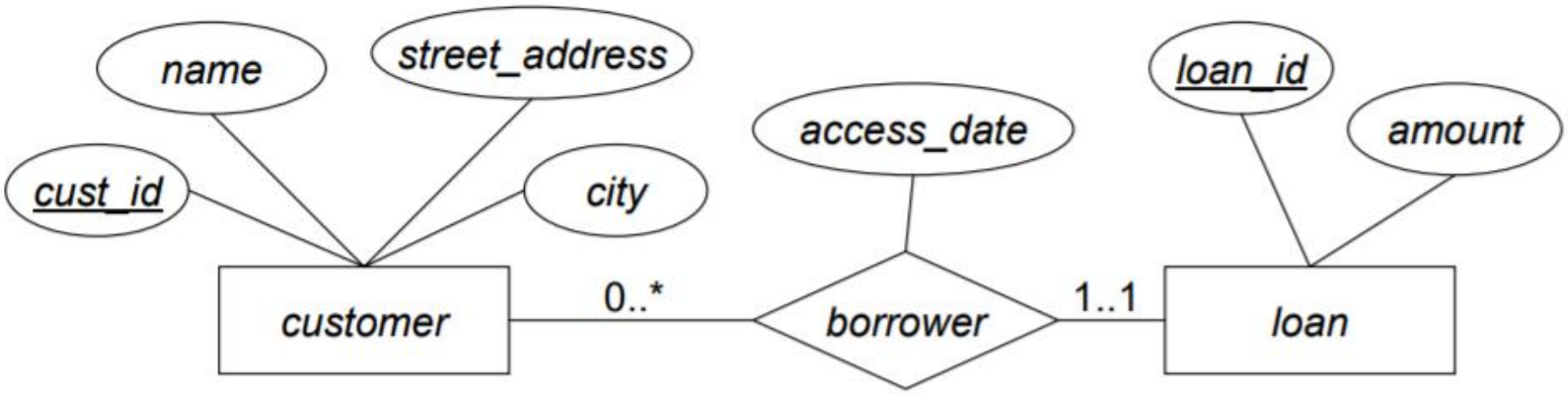
# Numerical Constraints (2)

- Total participation:
  - Lower bound at least 1
- Partial participation:
  - Lower bound is 0
- Can also state mapping constraints with numerical participation constraints



# Numerical Constraint Example

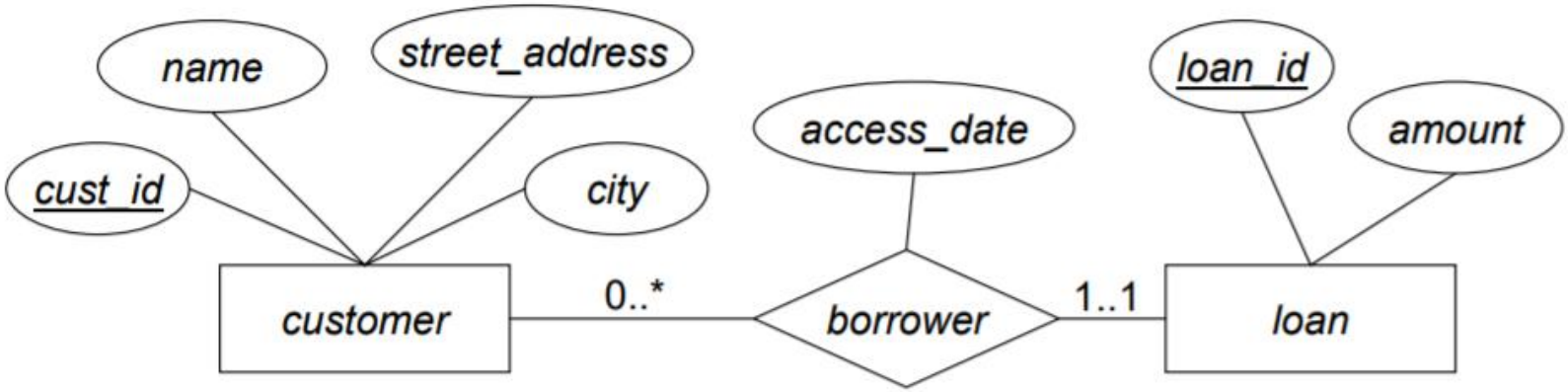
- What does this mean?




  - Each *customer* entity may participate in zero or more relationships in this relationship-set
    - *A customer can have zero or more loans.*
  - Each *loan* entity must participate in *exactly one* relationship (no more, no less) in this relationship-set
    - *Each loan must be owned by exactly one customer.*

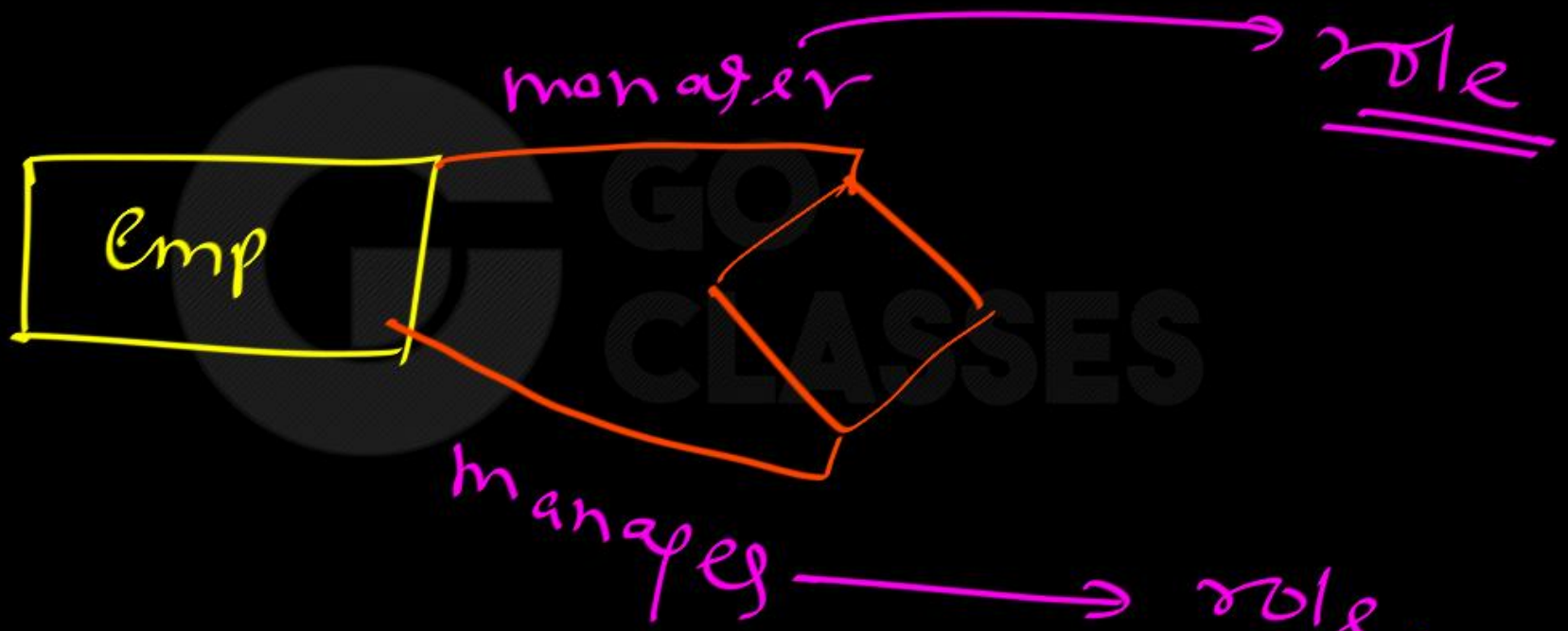
# Numerical Constraint Example (2)

- What is mapping cardinality of *borrower* ?



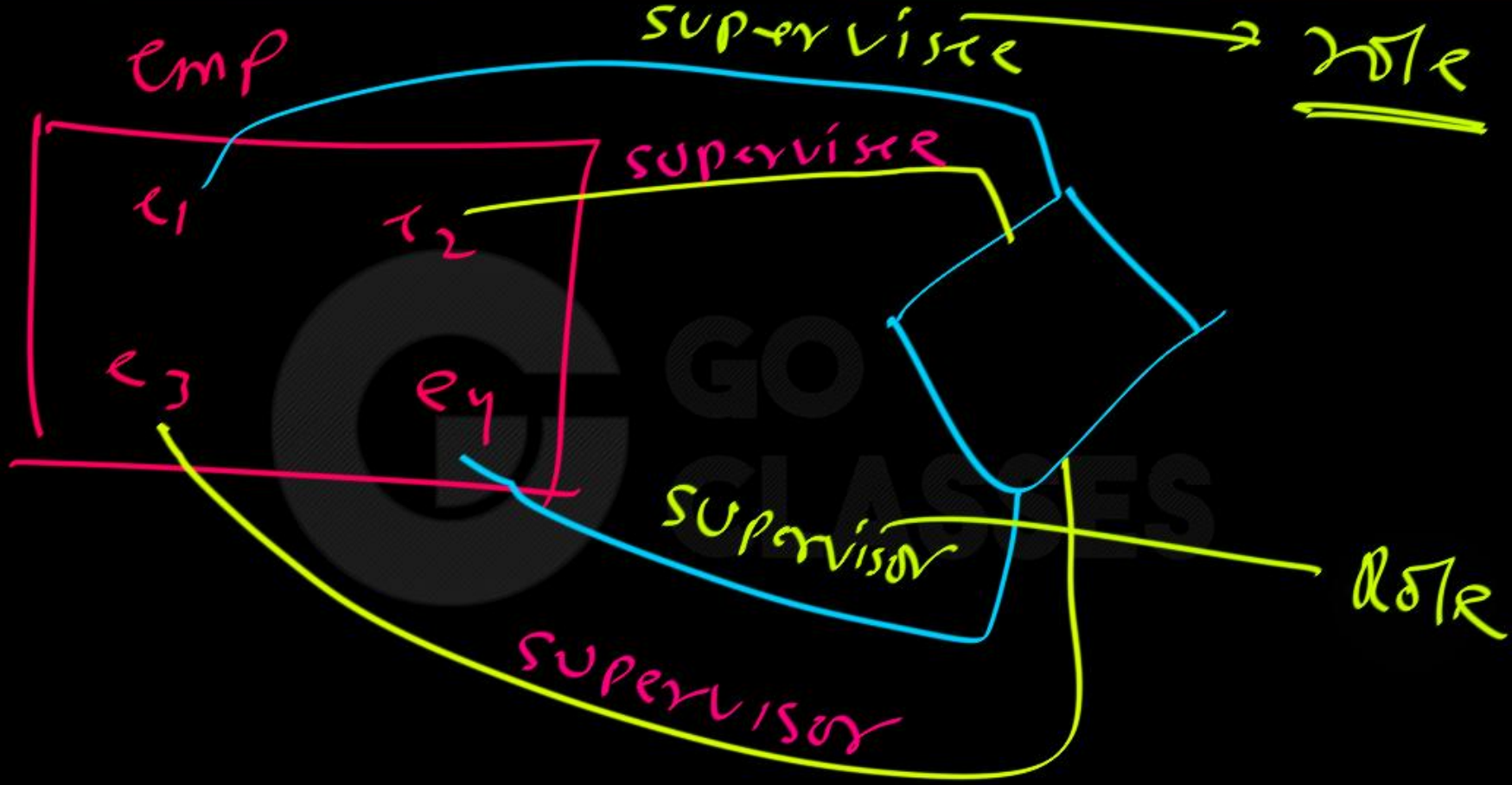

- From last slide:
  - *A customer can have zero or more loans*
  - *Each loan must be owned by exactly one customer.*
- This is a many-to-one mapping.

# Self-Referential relationship:



Emp:

| eid | supervisor |
|---|---|
| $e_1$ | $e_4$ |
| $e_2$ | $e_3$ |
| $e_3$ | — |
| $e_4$ | — |


emp
supervisee
role
e1
e2
supervisee
e3
e4
supervisor
role
supervisor

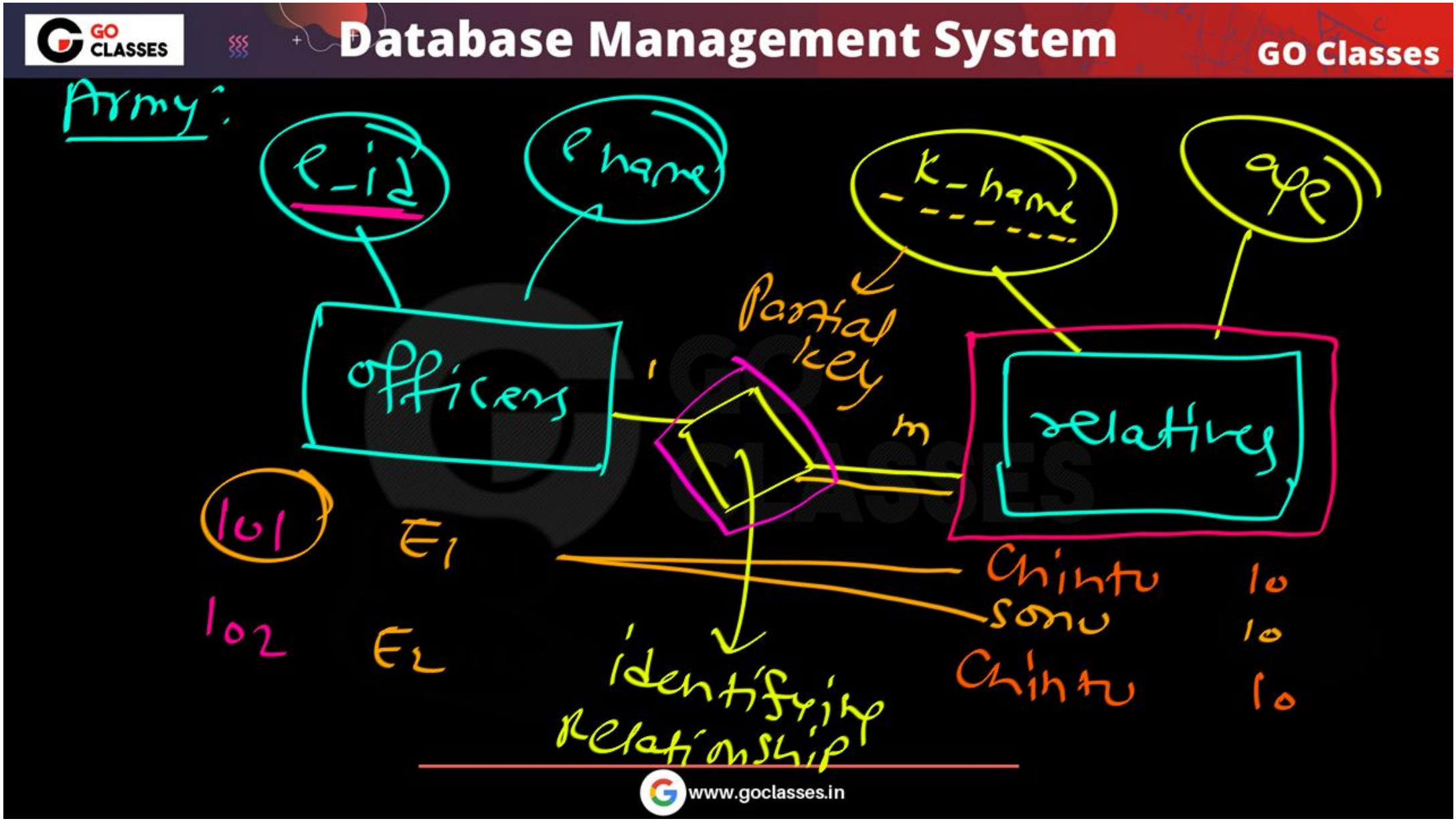
# Diagramming Roles

- Entities have roles in relationships
  - An entity's role indicates the entity's function in the relationship
  - e.g. role of customer in *borrower* relationship-set is that they own the loan
- Sometimes roles are ambiguous
  - e.g. when the same kind of entity is involved in a relationship multiple times
- Example: *works_for* relationship
  - Relationship is between two *employee* entities
  - One is the *manager*; the other is the *worker*

Weak entity set :⇒ a ES in which entities are not identifiable uniquely.

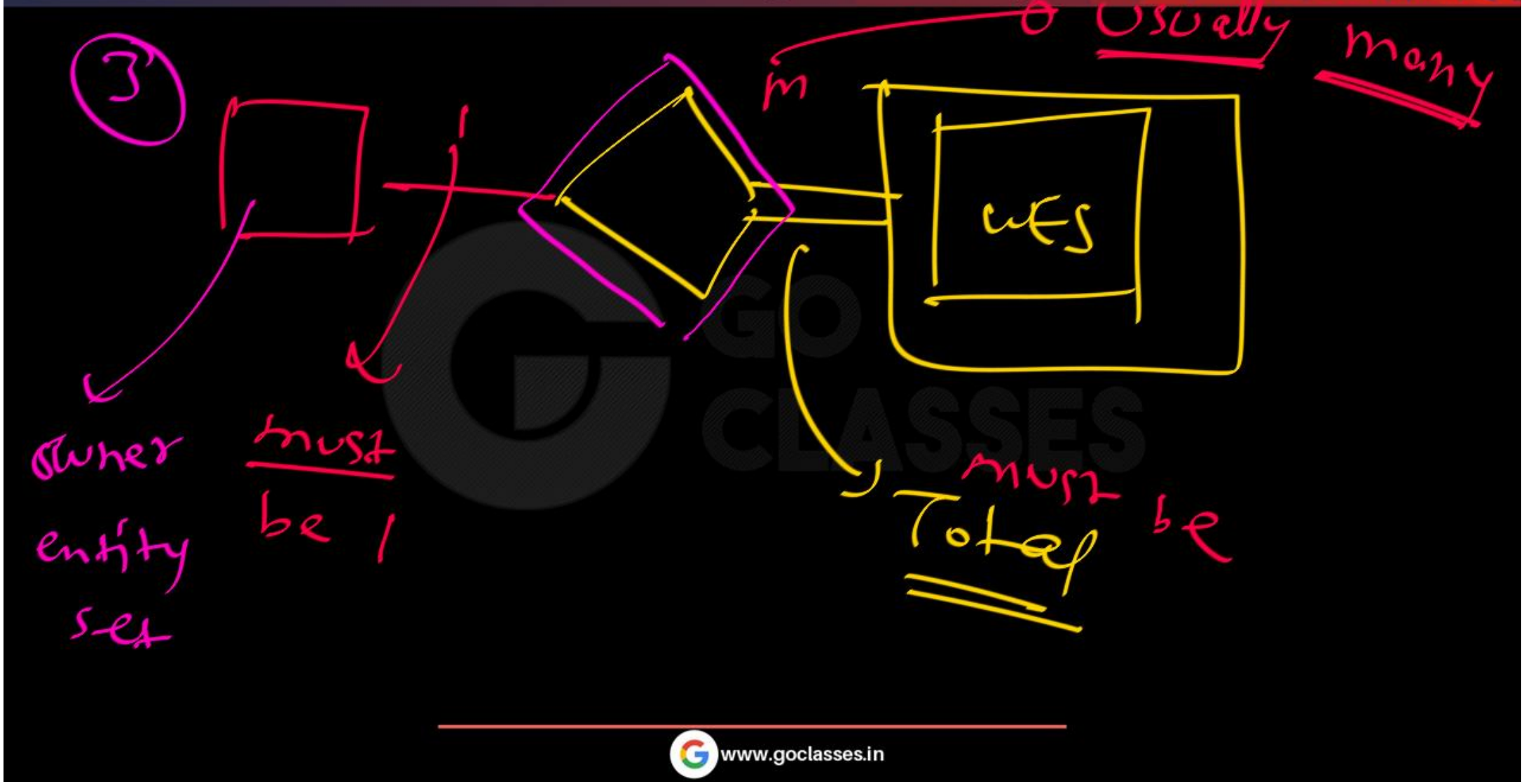
↳ entity set without any key.

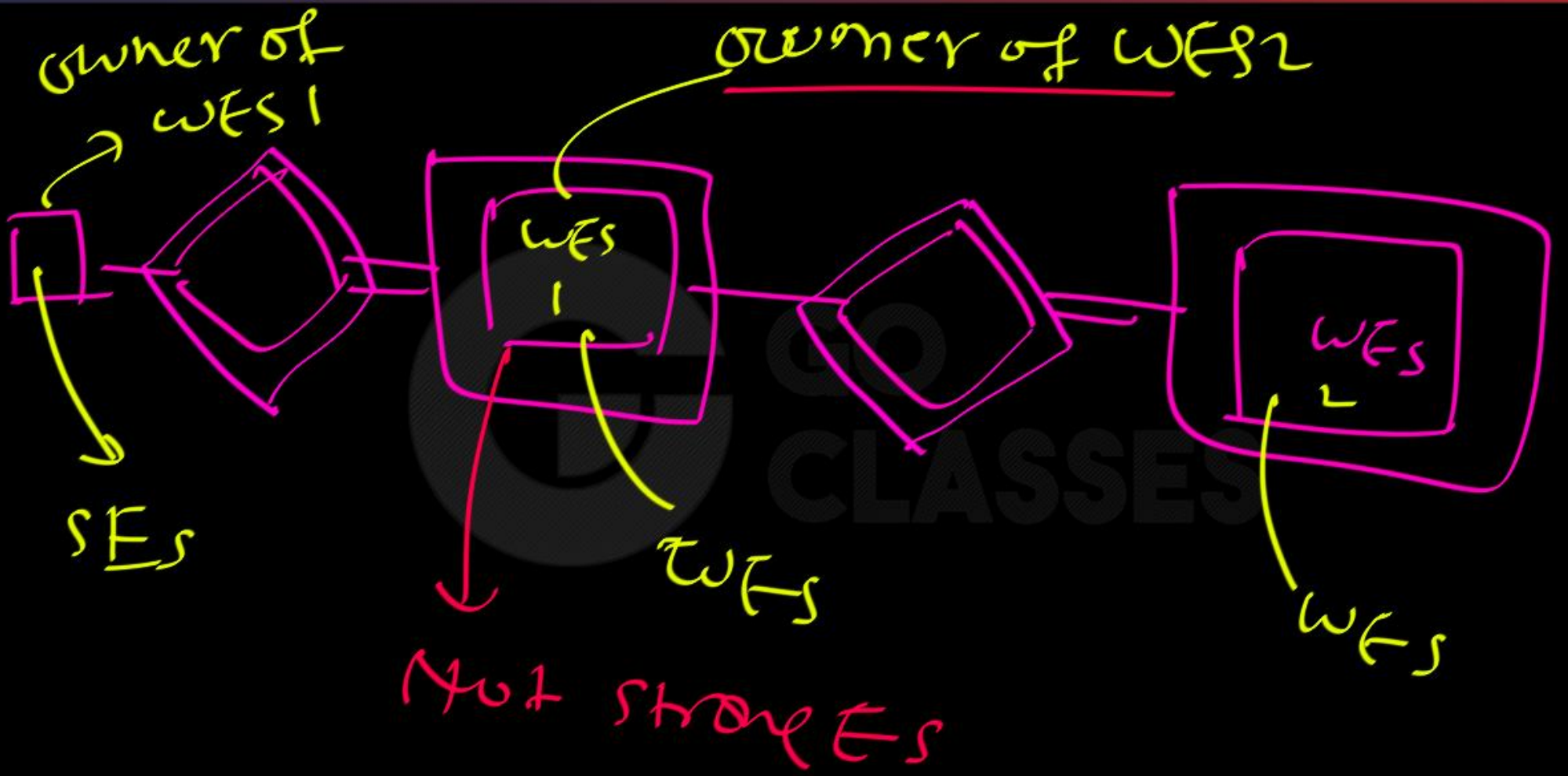
What is the need of this type of entity set?


Army:
e_id
e name
k_name
age
officers
1
Partial key
m
relatives
101
E1
Chintu
10
102
E2
Sonu
10
Chintu
10
identifying relationship

# Weak ES (WES):

1. Does not have any key.
2. Existance of weak entity Depends on existance of owner entity.


3
1
m
0 Usually
many
WES
Owner
entity
set
must
be 1
must
Total
be


owner of WES1
owner of WES2
WES 1
WES 2
SEs
TWEs
Mut strongEs
WEs

Strong Entity set : ⇒ Not dependent on any other Es for existence.
which has its own key.

Weak Es : which doesn't have its own key.
→ Dependent on some Es.
→ Dependent on owner entity.

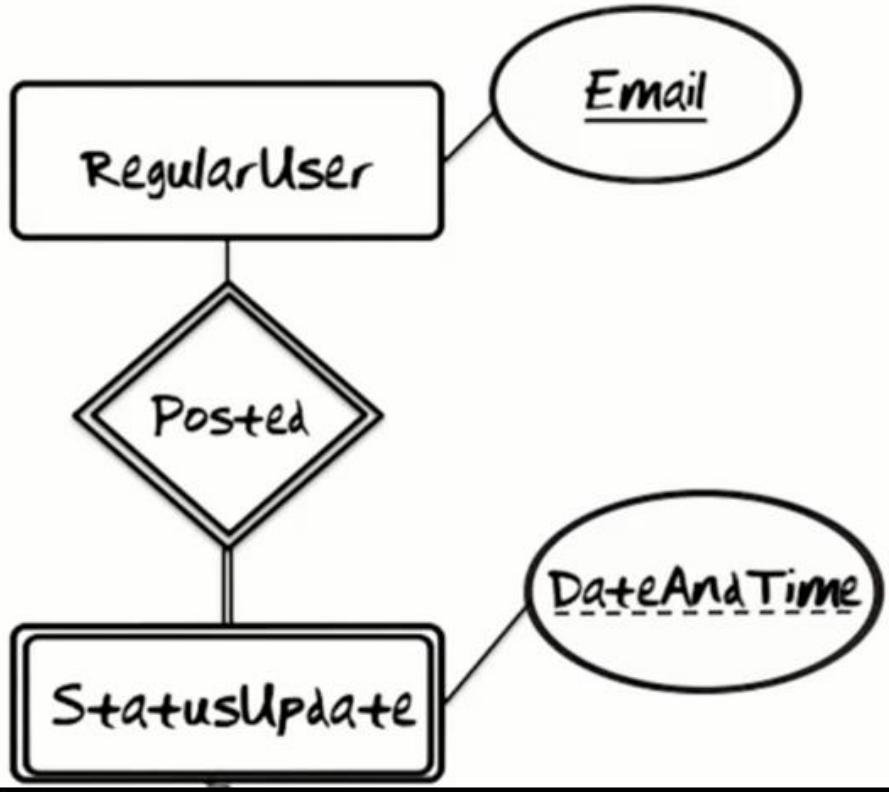
# Identifying relationships / weak entity types

StatusUpdate cannot

- exist without RegularUser
- be identified without RegularUser

(Email, DateAndTime) identifies

- StatusUpdate



# Identifying relationships / weak entity types

strong ES

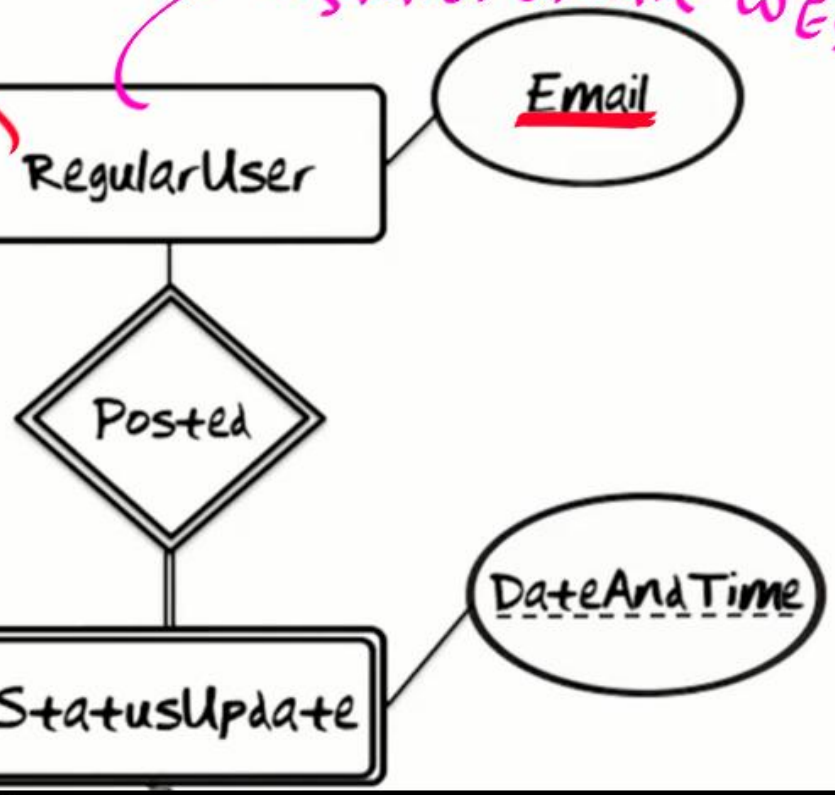
owner ES of "statusUpdate WES"

StatusUpdate cannot

- exist without RegularUser
- be identified without RegularUser

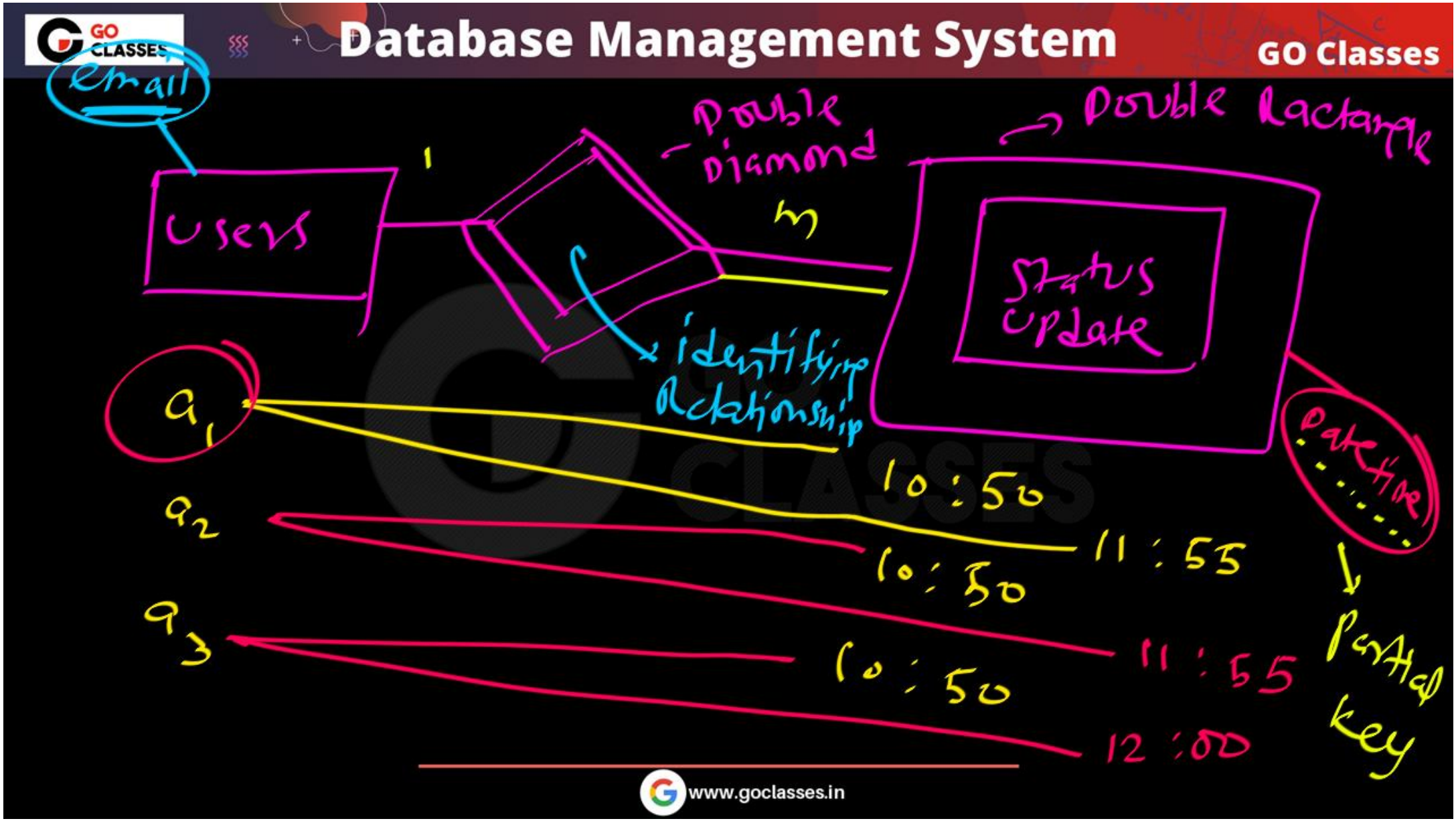
(Email, DateAndTime) identifies

- StatusUpdate



# Database Management System





Partial key ≡ Discriminator

Primary key of Identifying Relationship:

(PK of owner ES, Partial key of WES)

Ans: (Email, Datetime)

Partial key ≡ Discriminator

Primary key of WES : None

# Weak Entity-Sets

- Sometimes an entity-set doesn't have distinguishing attributes
  - Can't define a primary key for the entity-set!
  - Called a weak entity-set

# Weak Entity-Sets (2)

- Weak entity-sets *must* be associated with another (strong) entity-set
  - Called the identifying entity-set, or owner entity-set
  - The identifying entity-set owns the weak entity-set
  - Association called the identifying relationship
- Every weak entity *must* be associated with an identifying entity
  - Weak entity's participation in relationship-set is total
  - The weak entity-set is existence dependent on the identifying entity-set
  - If the identifying entity is removed, its weak entities should also cease to exist

# Weak Entity-Set Keys

- Weak entity-sets don't have a primary key
  - Still need to distinguish between weak entities associated with a particular strong entity
- Weak entities have a discriminator
  - A set of attributes that distinguishes between weak entities associated with a strong entity
  - Also known as a partial key

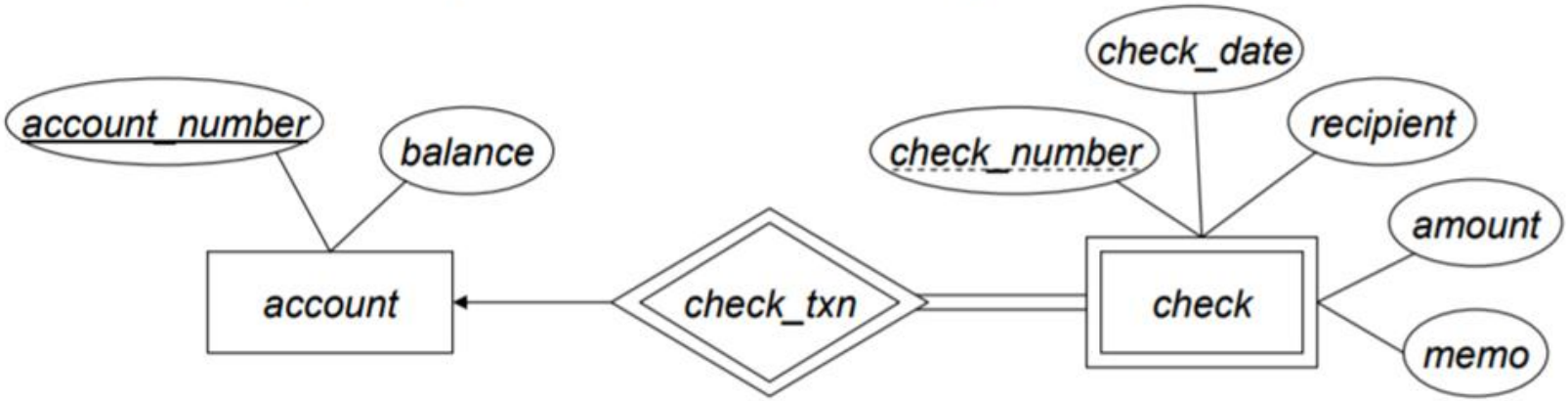
# Weak Entity-Set Keys (2)

- Using discriminator, can define a primary key for weak entity-sets
- For a weak entity-set *W*, and an identifying entity-set *S*, primary key of *W* is:

  $primary\_key(S) \cup discriminator(W)$

# Diagramming Weak Entity-Sets

- Weak entity-sets indicated by a double box
  - Discriminator attributes underlined with a dashed underline
- Identifying relationship to owning entity-set indicated with a double diamond
  - One-to-many mapping
  - Total participation on weak entity side



# Review

- Can represent constraints in E-R diagrams
  - Mapping cardinalities
  - Primary keys
  - Participation constraints
- Can also represent weak entity-sets
  - Doesn't have a primary key
  - Is owned by another entity-set via an identifying relationship

TOC Doubt:

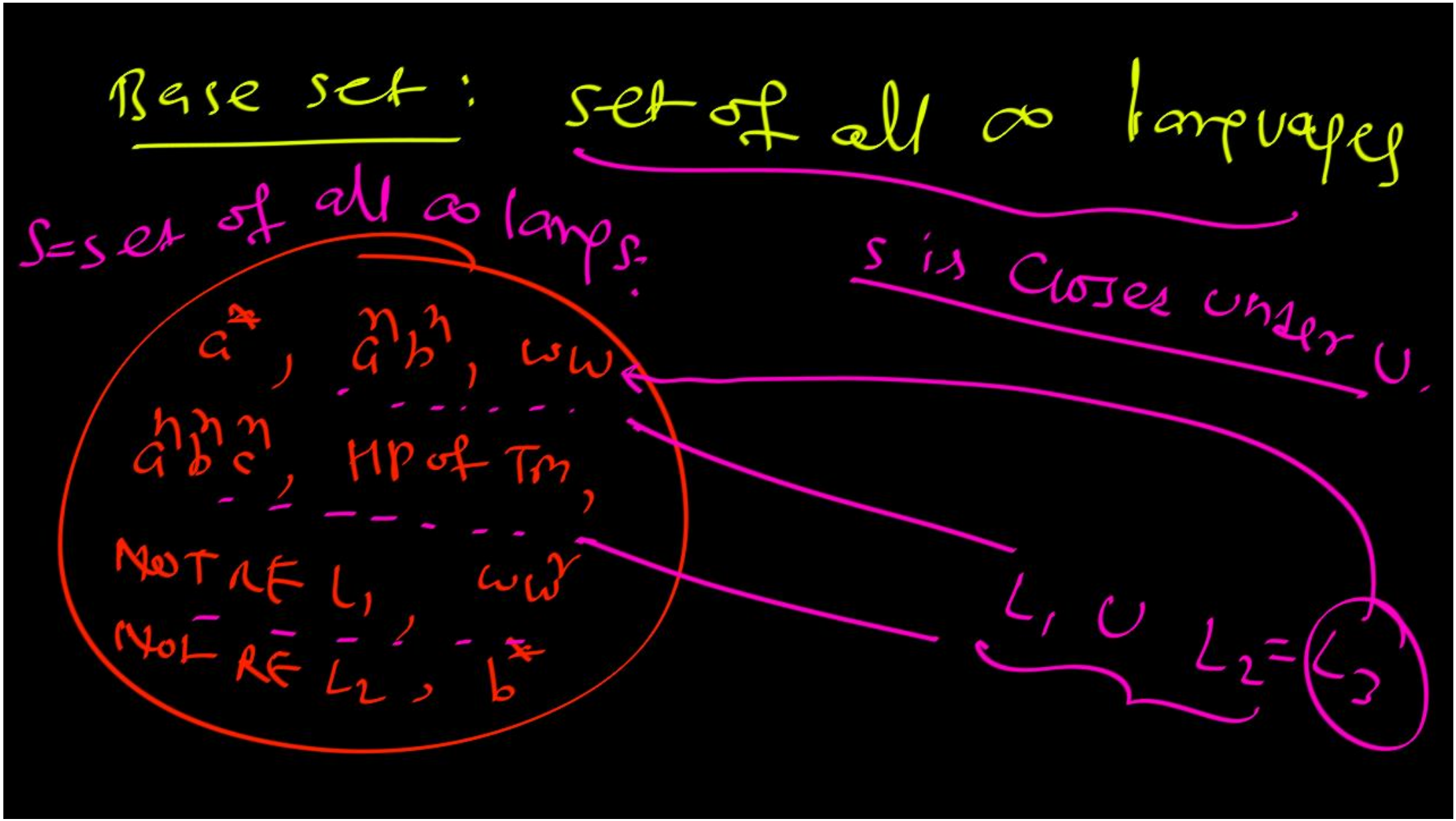
$L_i$ : a infinite lang; $i \geq 1$

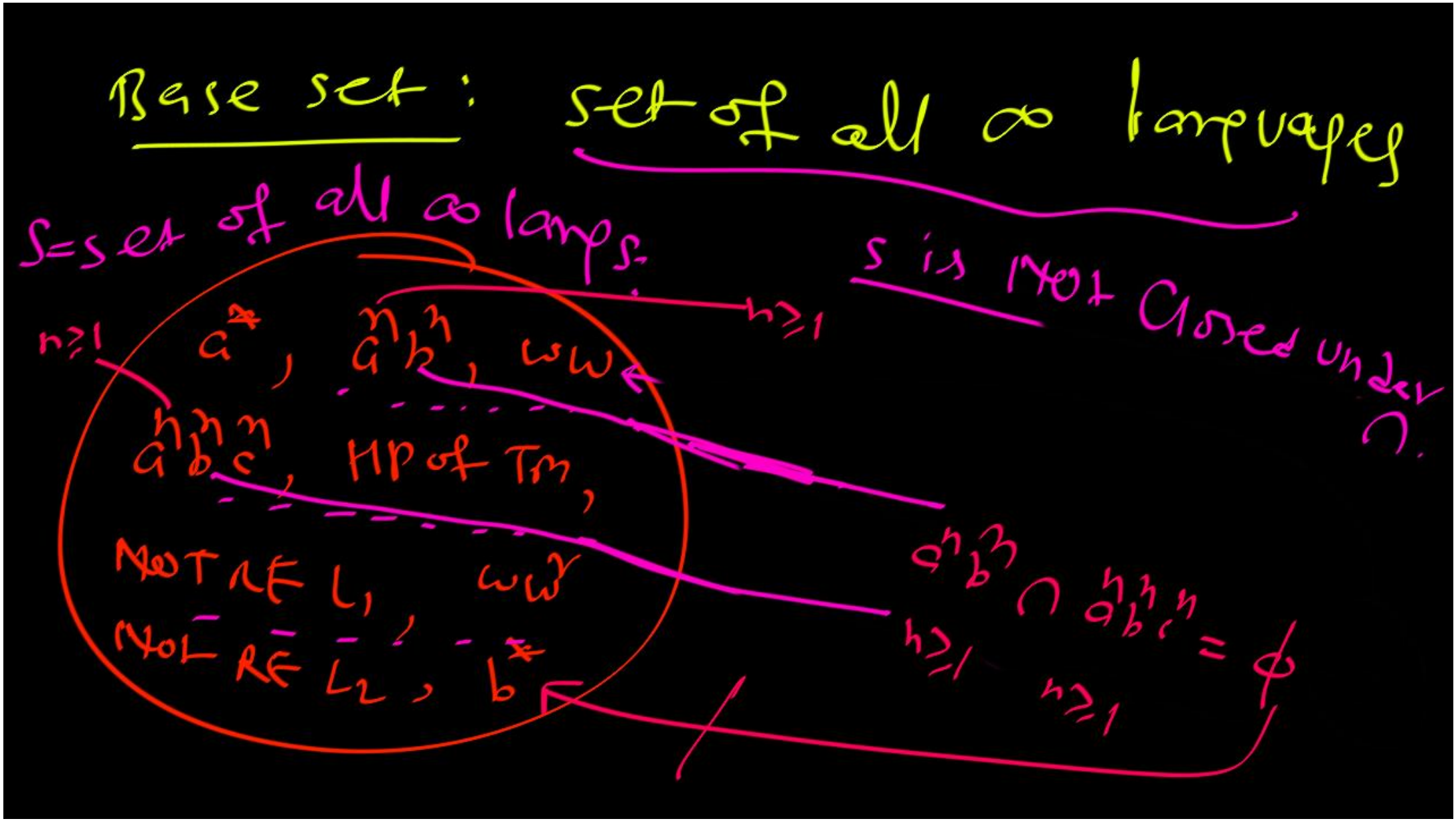
Set of infinite languages is closes under

$L_1, L_2, L_3, \ldots$ : each $L_i$ is a infinite lang.

(1) finite union $L_1 \cup L_2$ = inf. lang.

(2) infinite union $L_1 \cup L_2 \cup L_3 \cup L_4 \ldots$ : inf. lang.


Base set : set of all ∞ languages
S = set of all ∞ langs:
$a^*$, $a^n b^n$, ww
$a^n b^n c^n$, HP of Tm,
NOT RE $L_1$, $ww^r$
NOT RE $\bar{L}_2$, $b^*$
S is closed under ∪.
$L_1 \cup L_2 = L_3$


Base set : set of all ∞ languages
S = set of all ∞ langs :
n≥1
$a^n$ , $a^n b^n$ , ww
$a^n b^n c^n$ , HP of Tm ,
NOT RE $L_1$ , $ww^r$
NOT RE $L_2$ , $b^*$
S is NOT closed under ∩ .
$a^n b^n \cap a^n b^n c^n = \phi$
n≥1 n≥1

# TOC Doubt:

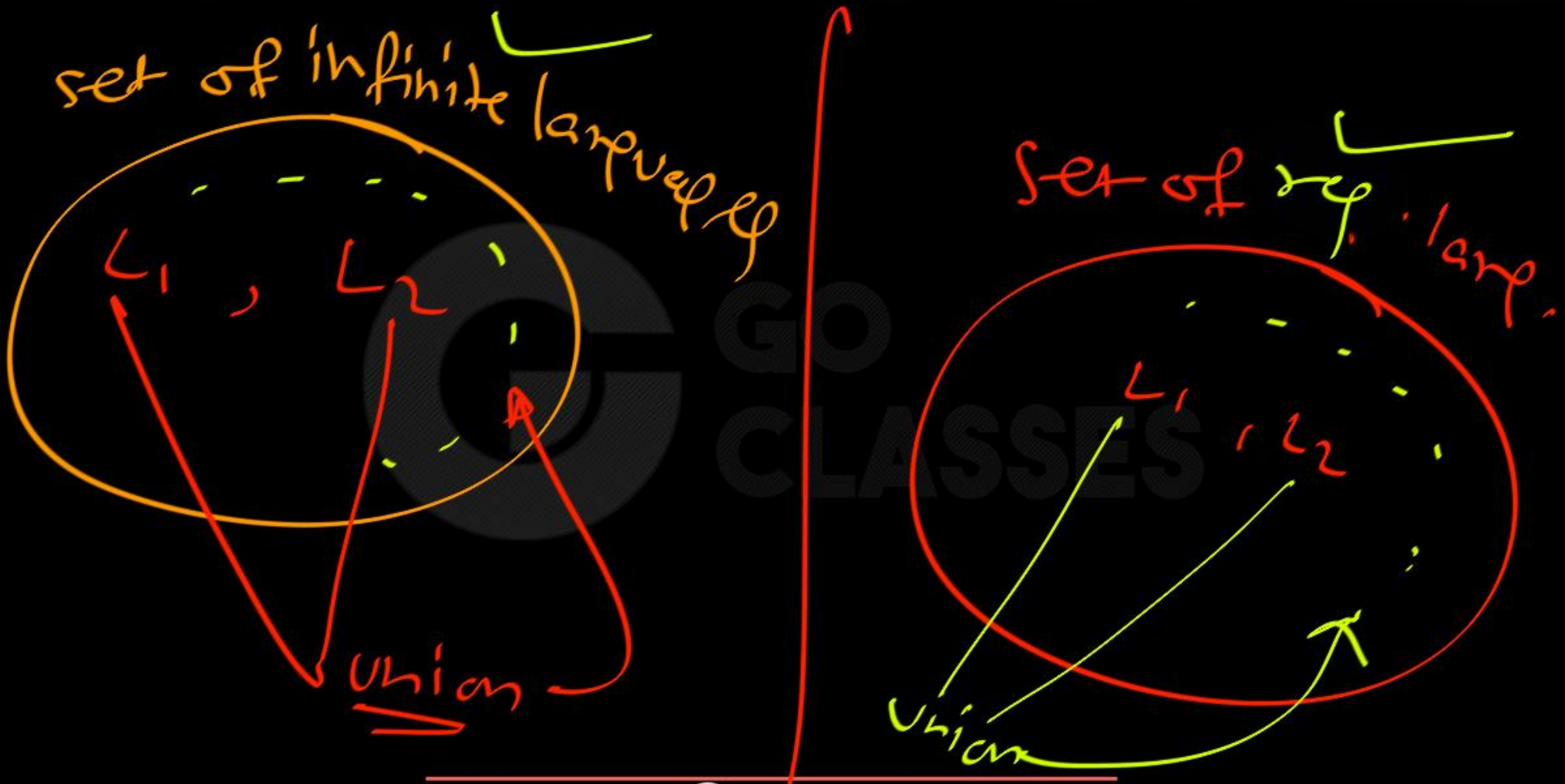
$L_i$ : a regular lang. $i \geq 1$

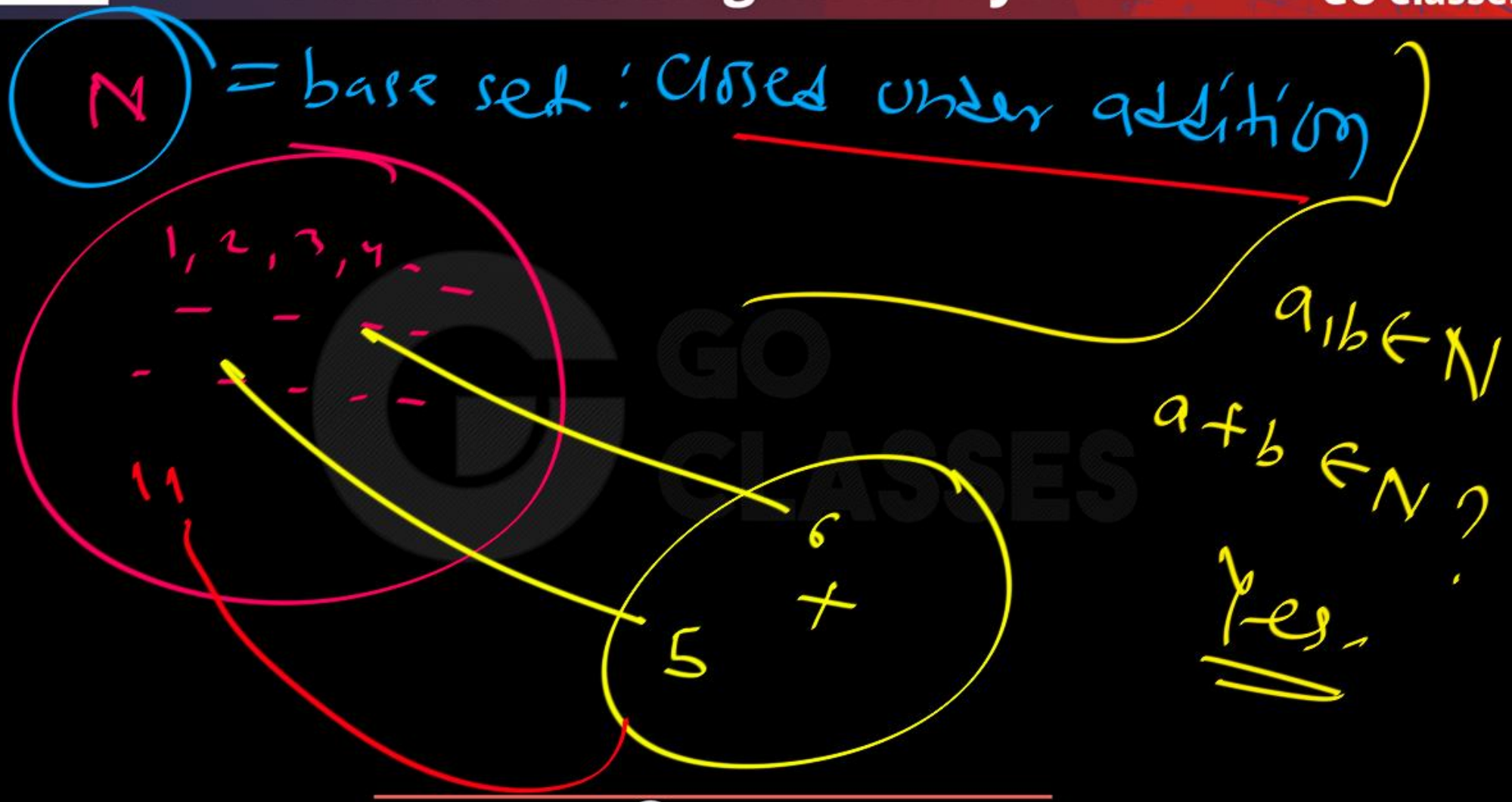
Set of Regular languages is Closes under

$L_1, L_2, L_3, \ldots$ : each $L_i$ is a regular lang.

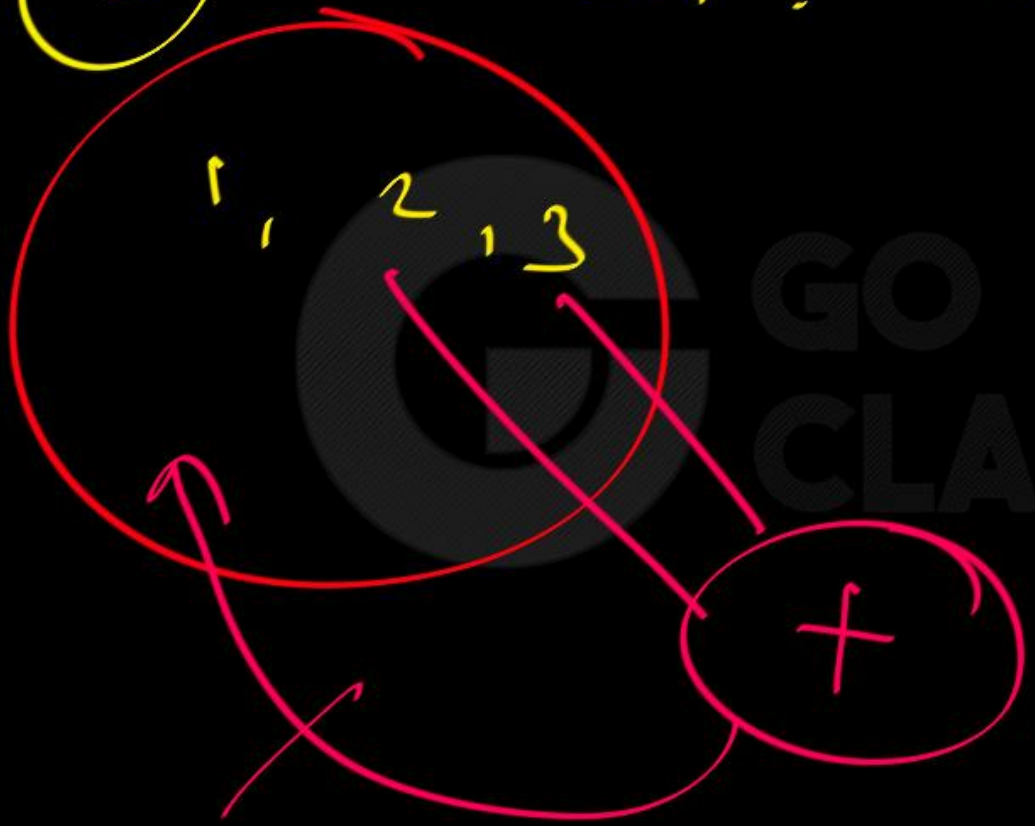
① finite union $L_1 \cup L_2$ = Regular lang

② Infinite union $L_1 \cup L_2 \cup L_3 \cup L_4 \ldots$ : may not be Reg.


set of infinite language
$L_1$, $L_2$, ...
Union
set of reg. lang.
$L_1$, $L_2$, ...
Union


N = base set : closed under addition
1, 2, 3, 4 - - - - - - - - - -
11
6
+
5
a, b ∈ N
a + b ∈ N ?
Yes.

S : base set : Not closed under Addition




$a, b \in S$

$a + b \in S$ ?

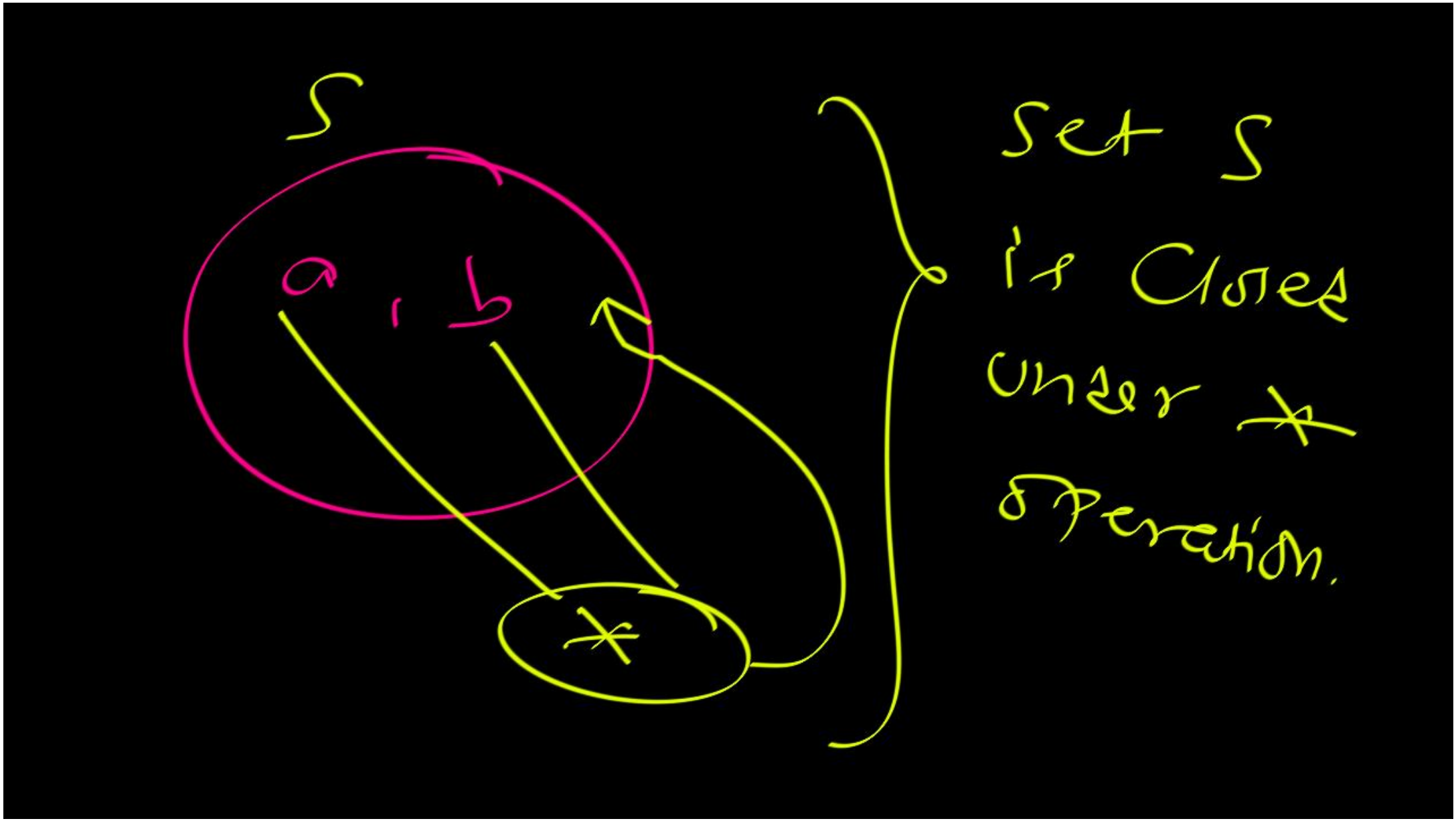
No.

Group Theory: (Base set, Operation)

Base set : S
Operation : *

S is closed under * if

$a, b \in S$ ; $(a * b) \in S$


S
a , b
*
Set S
is closed
under *
operation.


$S = \{1\}$

S is Closed under multiplication

S is Not Closed " addition