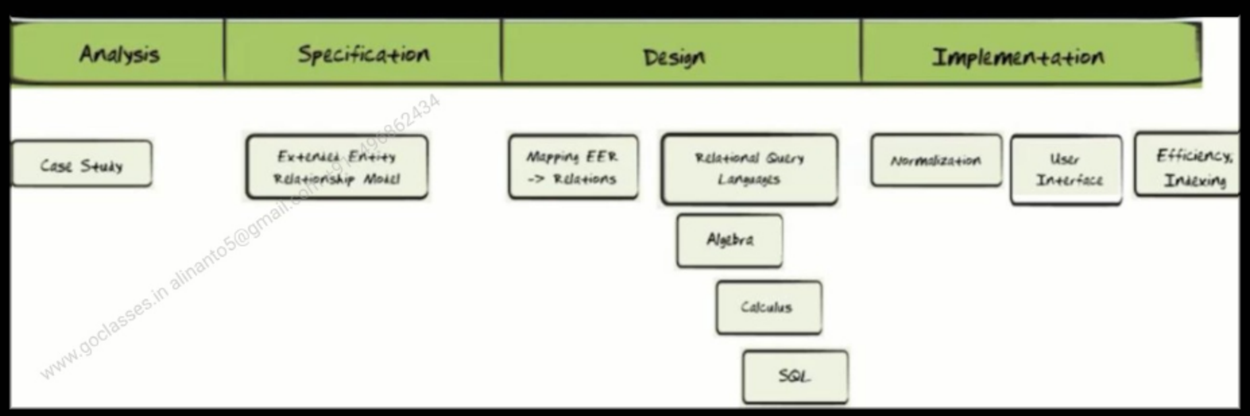
ER Model → Design Process (3) — Requirement Analysis
(Chen 1976) — Conceptual Design — ER Model
Logical design — Relational Model

→ No DBMS S/W on ER model → Conversion — why?

Why models? → Answers / examine / analyse — without the cost

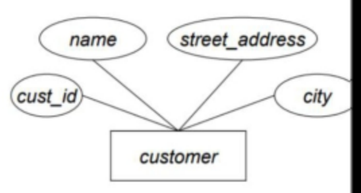


Entity → definition, Notation
Attribute → Attribute Name — unique & Domain → What underline means → Primary key attribute
Entity-Set → Collection of Similar Entity → Same set of attributes (Named-unique)

→ Keys in ER Model & Similarity to Relational Model

→ Cardinality of entity set

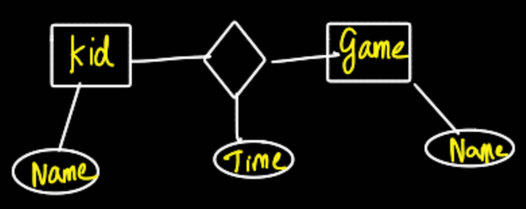
Example: a *customer* entity-set
- Attributes:
  - *cust_id*
  - *name*
  - *street_address*
  - *city*
- Entity-set is a box
- Attributes are ovals
- Lines connect attributes to entity-set

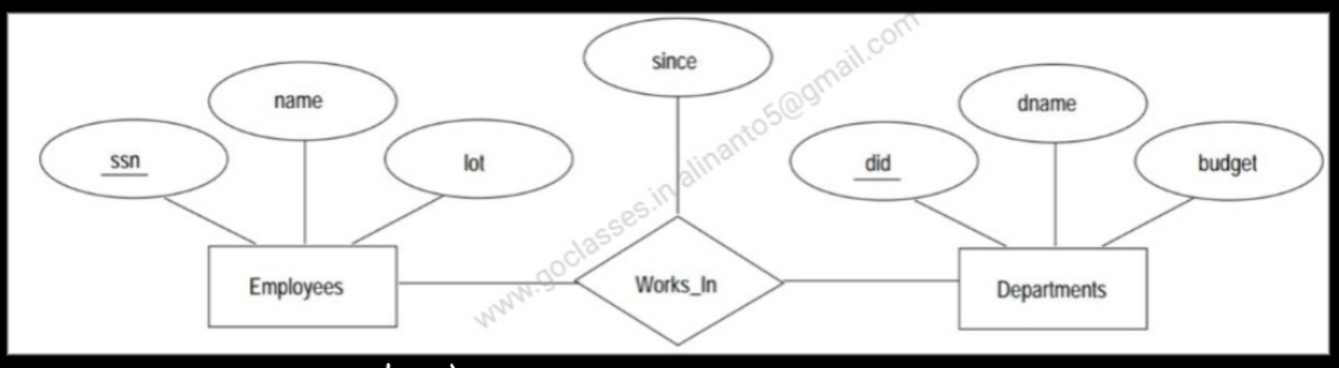


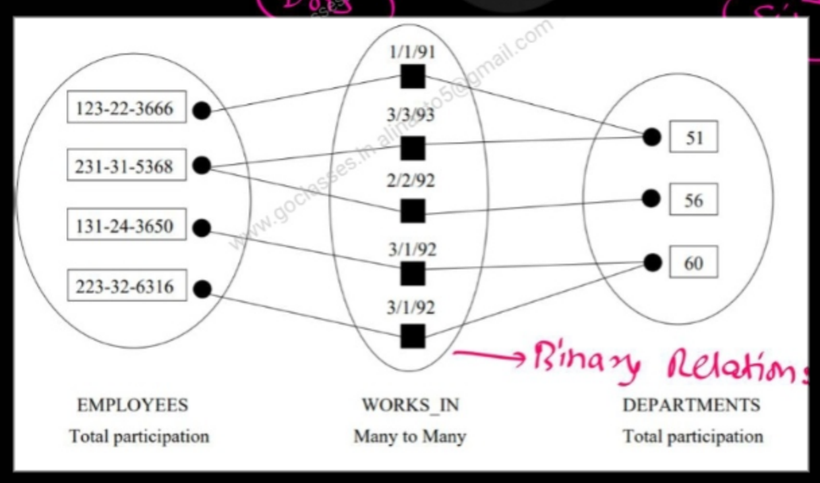
Relationships → Definition, Notation, Relationship set → similar Relations, btw same entity types, Named-unique

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

→ Relationship attributes (Descriptive attributes)

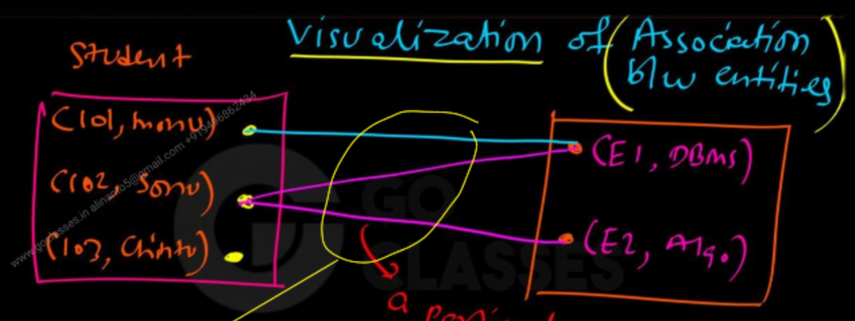






→ Binary Relations

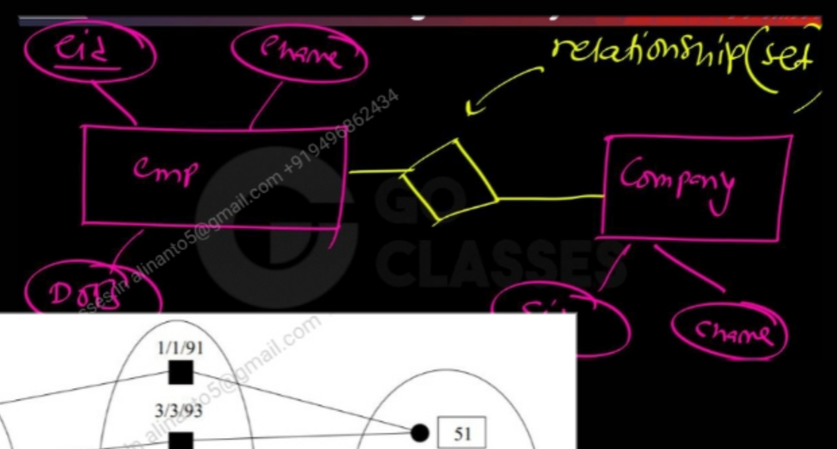
Visualization of (Association b/w entities)



a particular relationship

Visualization of Relationship

Relationship set



Relationship (cont...)
→ Degree of relationship set → # Entity set participating
→ Binary relationship set → Most common

Types of Attributes (3) → Simple vs Composite
→ Single Valued vs multivalued
→ Derived vs Base/Source
→ from Base eg) Age from DOB

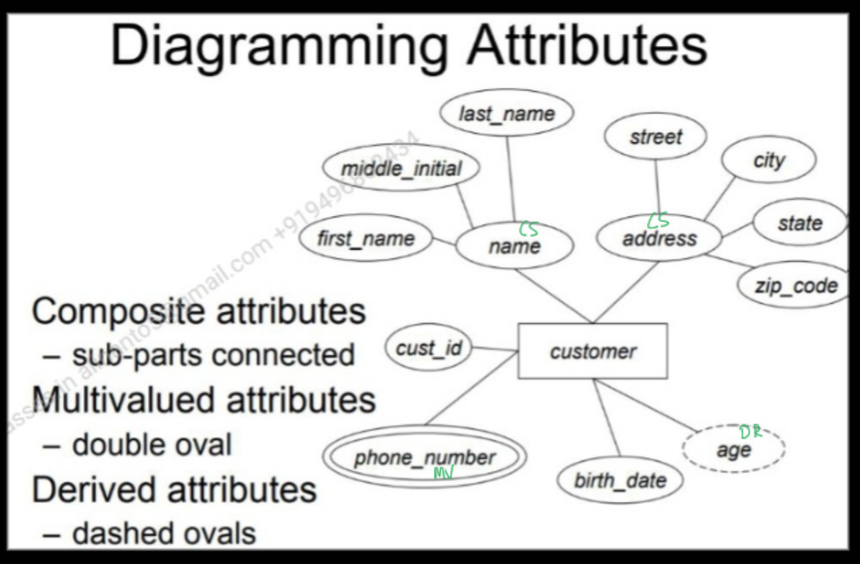
Simple attributes are atomic – they have no subparts
- e.g. *amount* attribute is a single numeric value

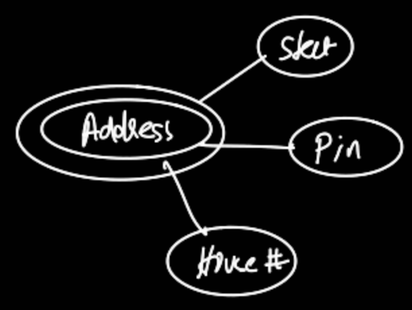
Composite attributes have subparts
- Can refer to composite attribute as a whole
- Can also refer to subparts individually
- e.g. *address* attribute, composed of *street*, *city*, *state*, *postal_code* attributes

Base attributes (aka source attributes) are stored in the database
- e.g. the *birth_date* of a *customer* entity

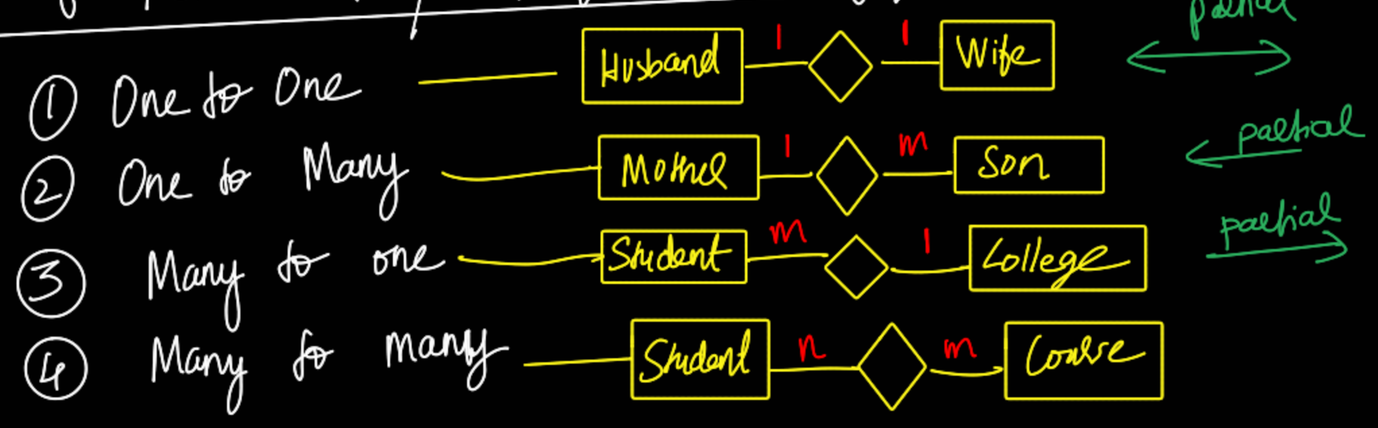
Derived attributes are computed from other attributes

- 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





→ Types of Relationships / Mappings cardinality / Cardinality ratio



→ Partial function → Every element of domain is mapped to atmost one element of codomain

→ Total function → Every element of domain is mapped to exactly one element of codomain

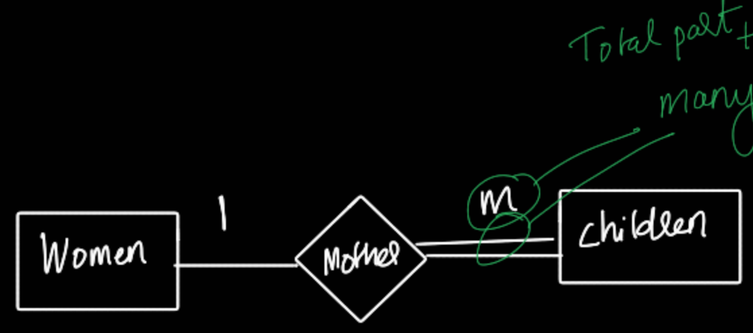
Total function ⟹ Partial function

↳ Default Meaning of "function"

participation → Total vs partial

T

atleast one



Total part + many ⟹ partial + Total part ⟹ Total function

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)

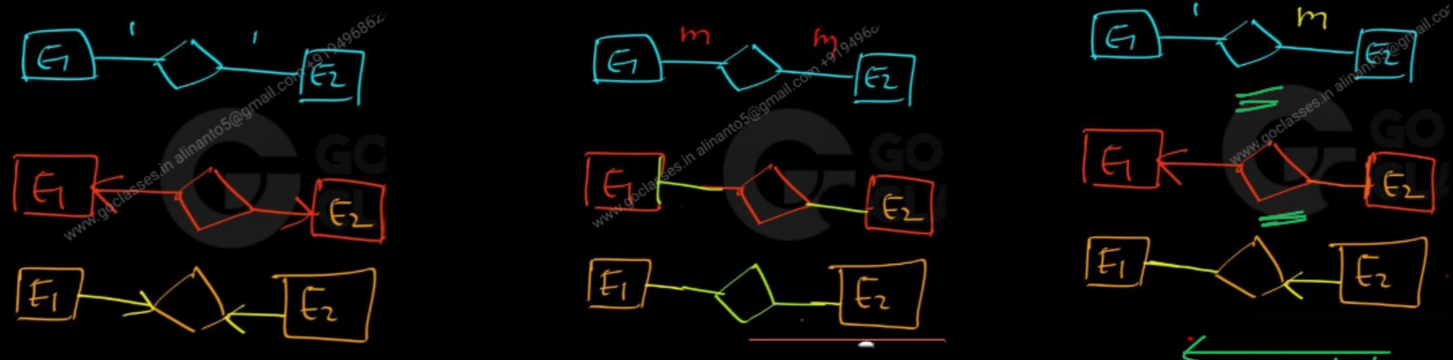
One-to-many mapping:
- A customer can have multiple loans
- Customers can't share loans

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

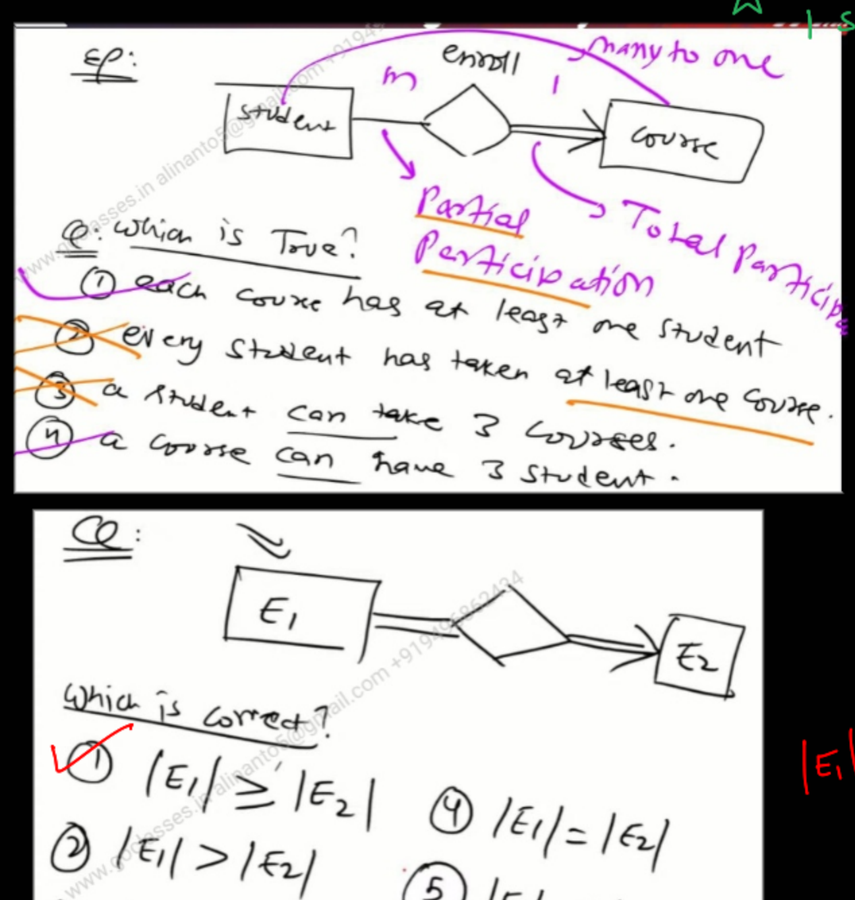
Best choice for *borrower* relation is many-to-many.

Handles real-world needs!



→ Relationships are identified by the entities participating in the relationship
- Relationship instances are uniquely identified by primary keys of participating entities

→ Not by descriptive attributes

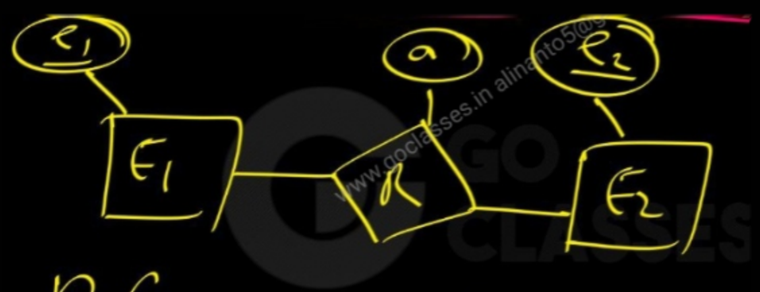


eg. Relationship set "Enroll" : (sid, cid)

↳ Primary key of Enroll : (sid, cid)

many-many mapping b/w student, course

|E1| = cardinality



$R(e_1, e_2, a)$

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

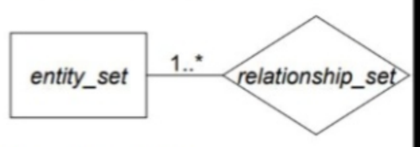
1-m : " " " " : e_2

m-1 : " " " " : e_1

m-n : " " " " : e_1 & e_2

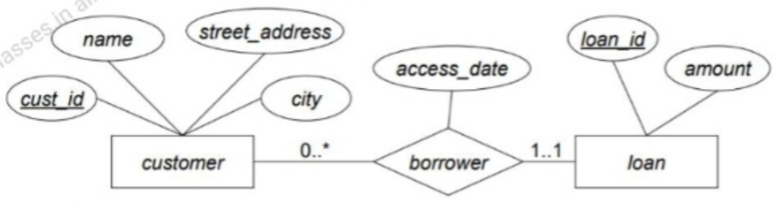
Numerical Constraints

- 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.

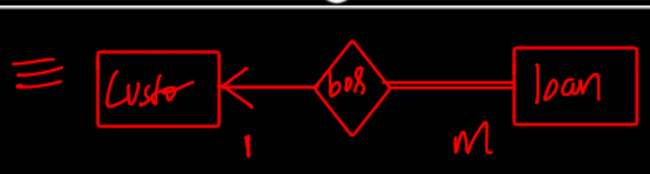


Numerical Constraint Example

- 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.*



1...1 → Total participation many side of 1 to many

1...* → Total participation one side of 1 to many

0...* → partial → 1 side of 1 to many

0...1 → partial → many side of 1 to many

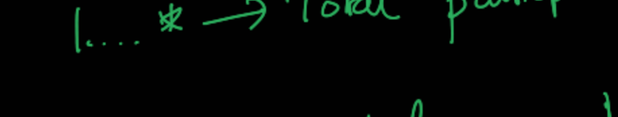
Lower bound → decided total or partial participation

Upper bound → 1 or many.

Self referential relationships → roles → Recursive relation – Cyclic relation

Weak entity set → Not uniquely identifyable → No key → Notation

→ Identifying relationship

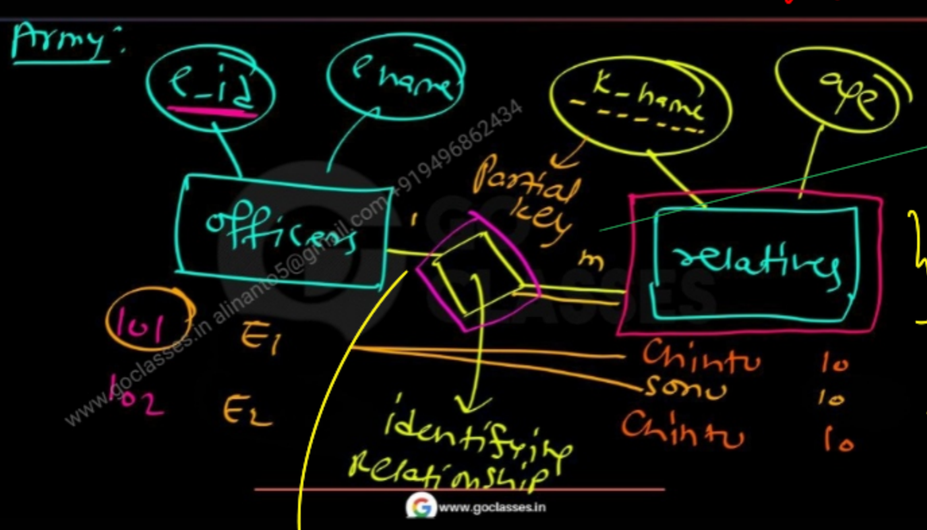


→ Partial key

→ Primary key ≡

< Primary key of owner's strong ES, partial key of WES >

Army:



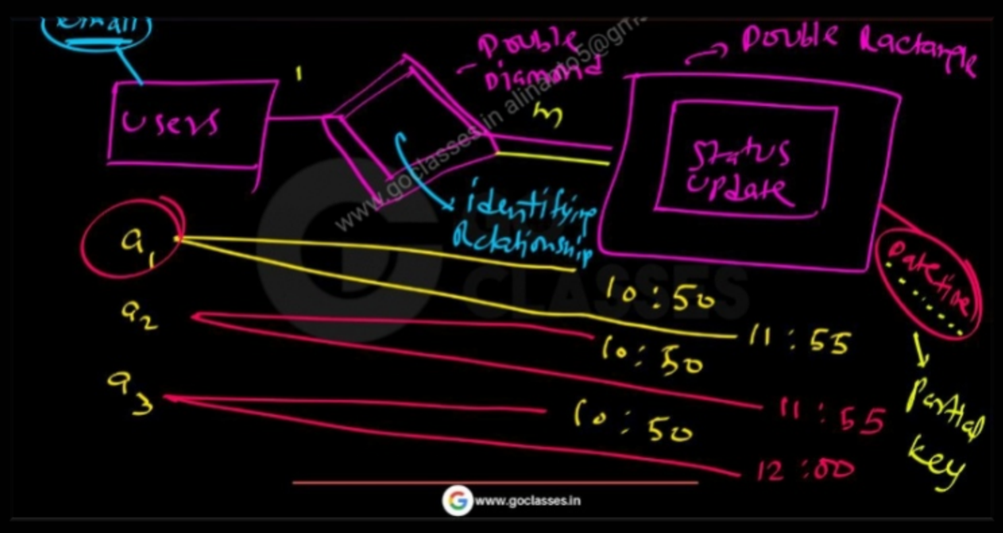
→ Visually many (not mandatory)

} existence depends on owner's entity

→ Must be total participation ☆☆

→ Must be 1 or cannot identify relationship uniquely ☆☆

- 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

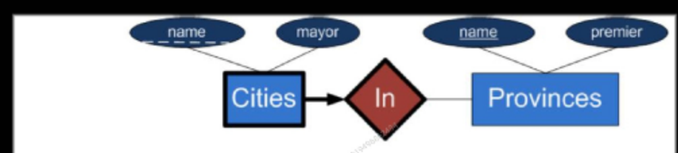


- Weak entity-sets don't have a primary key
  - Still need to distinguish between weak entities associated with a particular strong entity
- Weak entities have a discriminator
  - A set of attributes that distinguishes between weak entities associated with a strong entity
  - Also known as a partial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

- For a weak entity-set *W*, and an identifying entity-set *S*, primary key of *W* is:
  primary_key(*S*) ∪ *discriminator*(*W*)

Discriminator may not be a key ∵ can repeat.

But given a owner's entity's key, discriminator cannot repeat



Which of the following is necessarily true:

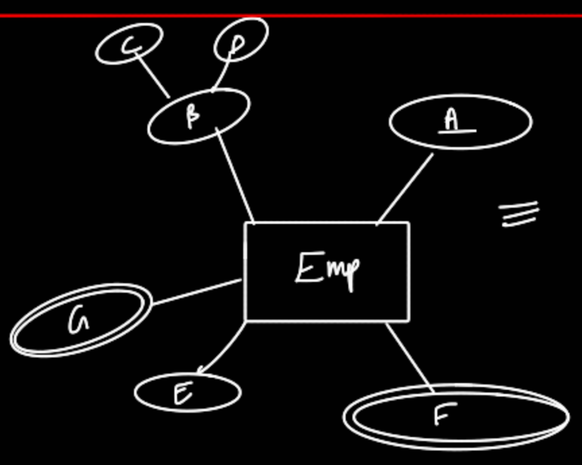
A. No two provinces can have premiers with the same name.
B. No two cities can have mayors with the same name.
C. No two cities can have the same name.
D. No person can be the mayor of Cities In two different province
E. None of the above E is correct

Conversion ER to relational

① Entity name → Relation name
② Attribute → Attributes
③ Key Attributes → Primary key
④ Composite Attributes → Not stored, only components stored
⑤ Multivalued → New relation for each MVA
⑥ Derived → Not stored

Why? → Problem 1 → Store in same table ⇒ Key has to include MVA
⇒ Null values not allowed

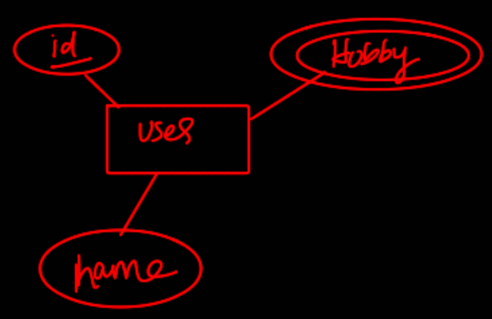
→ Problem 2 → Key → other attributes
Key + MVA = new key
} Redundancy in Relation.



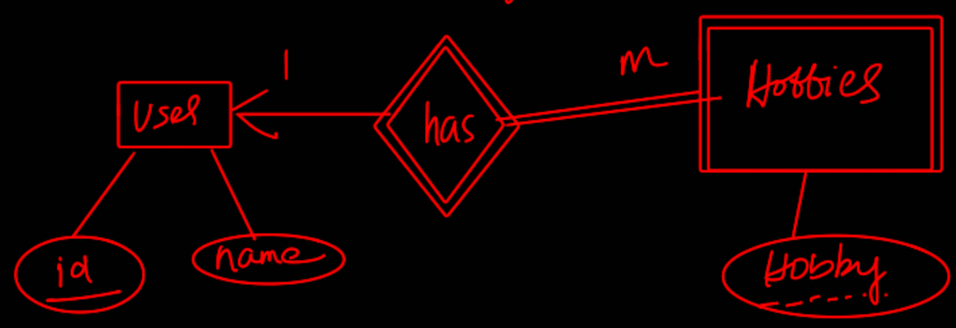
≡ Emp(A, C, D, E) — PK = A
Emp1(A, E) — PK = AF
Emp2(A, G) — PK = AG
A is the FK to Emp in both Emp1 & Emp2

→ No primary key required for strong entity sets
→ MVA can be converted to Weak Entity & new relation can be made for each MVA as a weak entity.

⇒ Every MVA can be converted in a Weak Entity set.

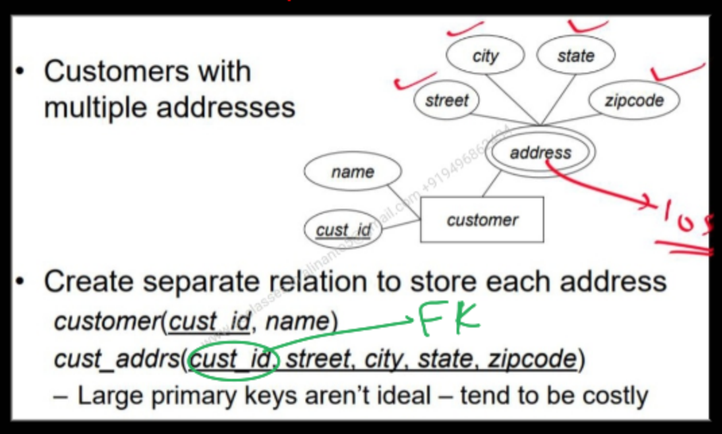


≡

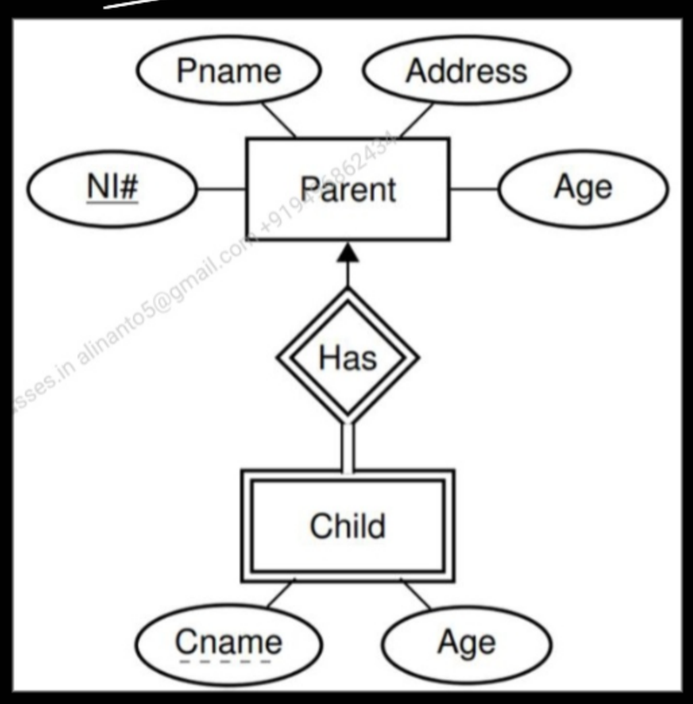


= User(id, name) — PK = id
Hobby(id, hobby) — PK = id, hobby
Partial key = hobby
FK for User relation = id

→ Multivalued + Composite Attributes.
= Complex Attributes



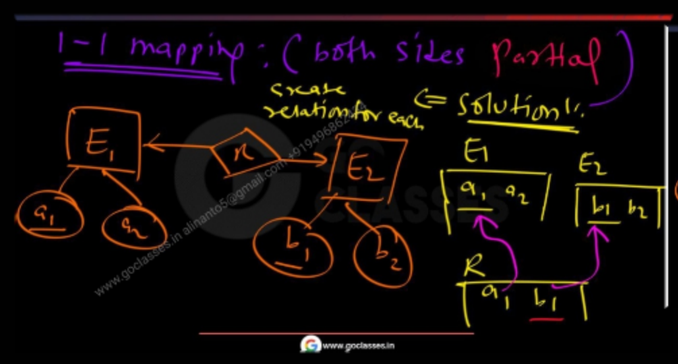
Weak Entity

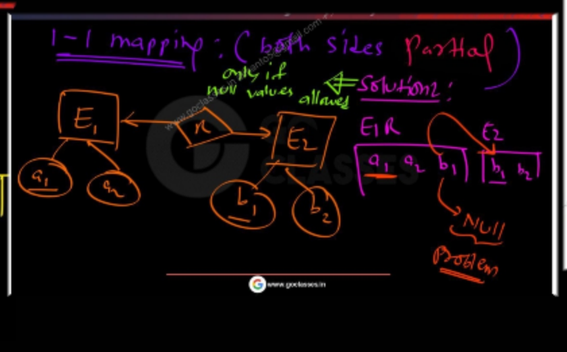


Parent(NI#, Pname, Age, Address)
Child(NI#, Cname, Age)
NI# → FK of child on parent

Mapping Cardinality | Participation

1-1 mapping (Both side partial partici)





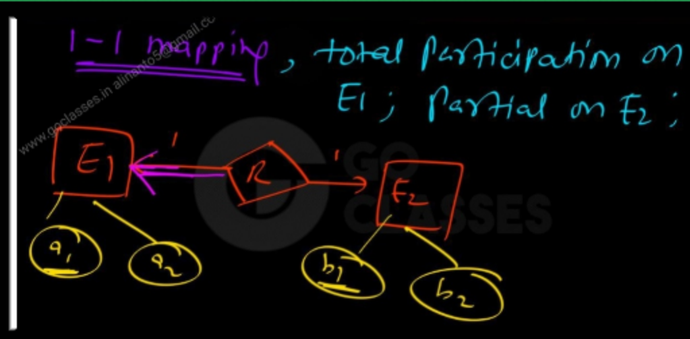
1-1 mapping, both sides optional:

Possible solutions:

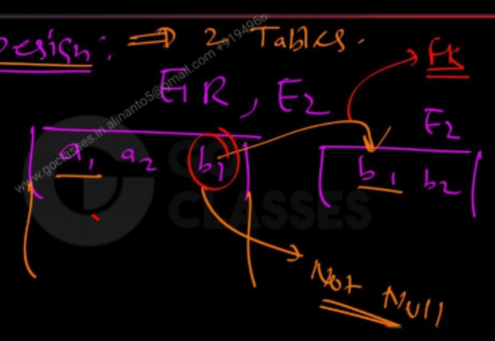
① E1 R, E2 } Problem of Null values
or E2 R, E1 } (better Design only if Null values allowed)

② E1, R, E2 ⇒ better Design

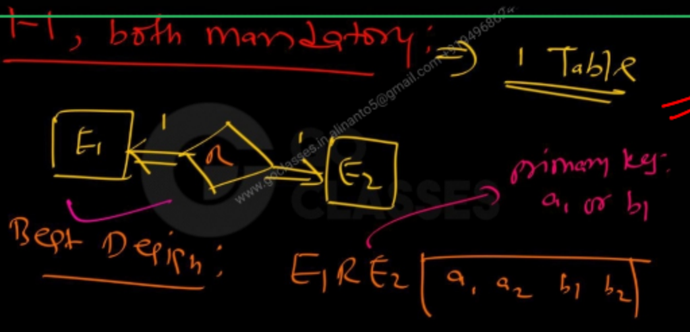
1-1 mapping (Any one side total participation)



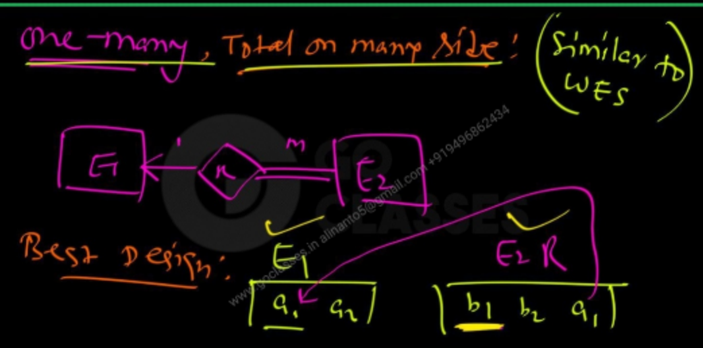
best design: ⇒ 2 Tables.



1-1 mapping (Both side total participation)

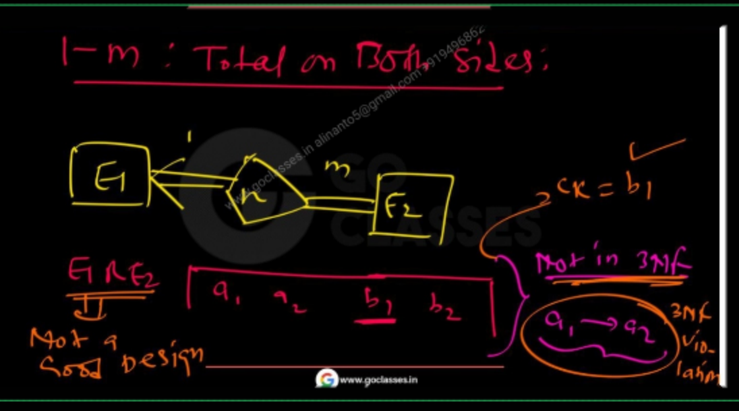


1-many mapping (Total on many side)

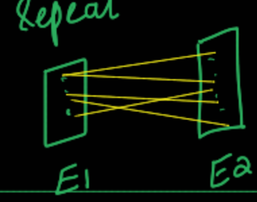


Similar to Weak Entity set with identifying Relationship

1-many mapping (Total participation on both sides)

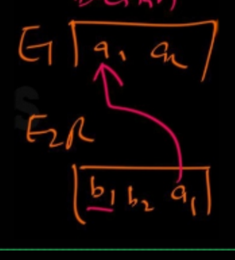


a1 not a SK since a1 can Repeat

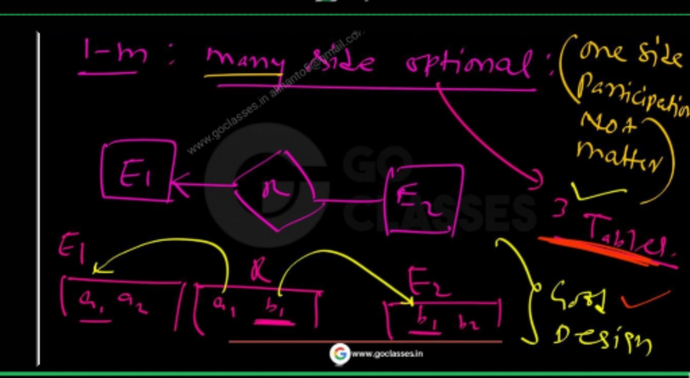


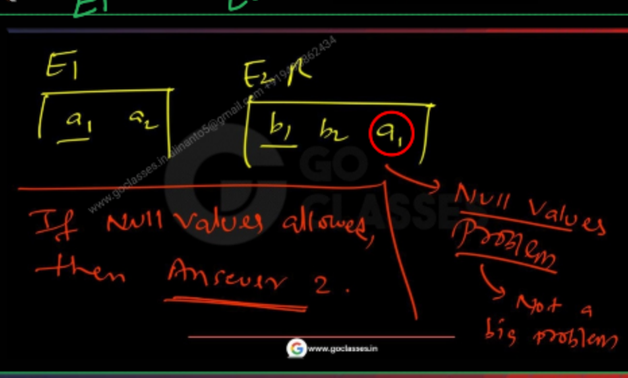
⇒ 2 Tables

Good Design

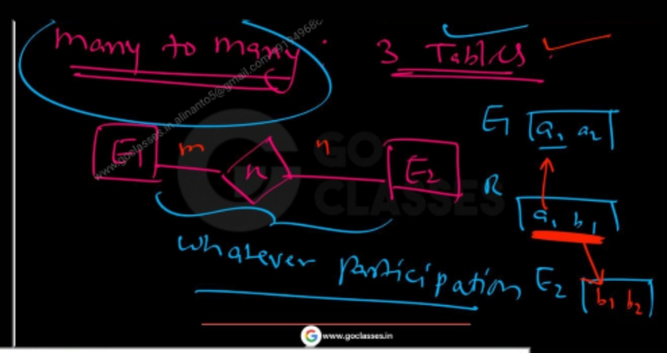


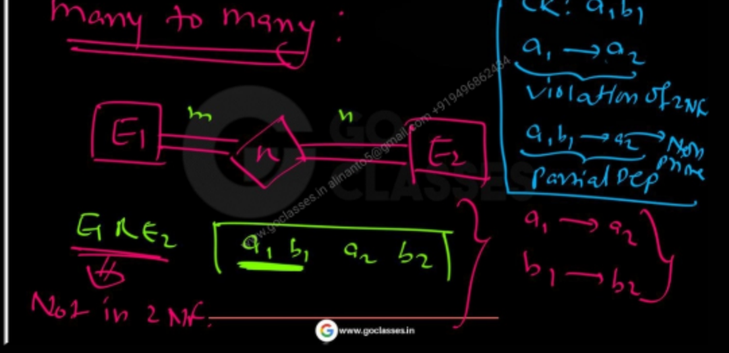
1-many mapping (both side partial partication.)





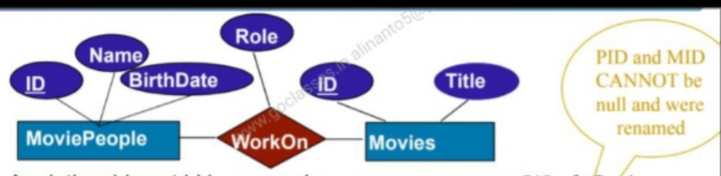
many - many (whatever participation)





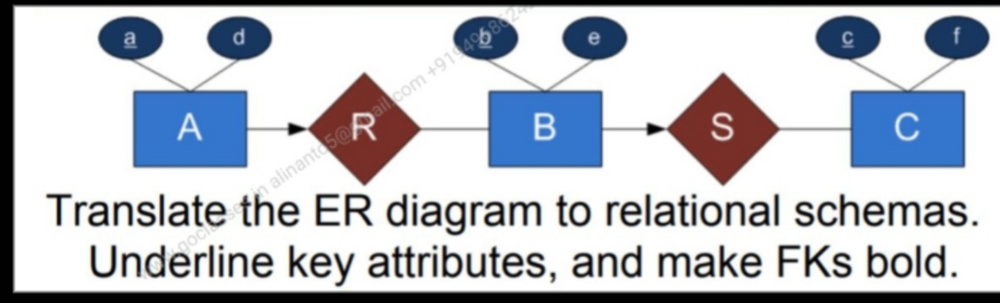
Cardinality	Membership Class	Number of Relations	Notes
1 : 1	Both Mandatory	1	all attributes in a single table
1 : 1	One Mandatory One Optional	2	Identifier of optional entity held in the mandatory entity relation
1 : 1	Both Optional	3	one relation for each entity and the relationship between them
1 : N	Both Mandatory	2	One relation for each entity and identifier of "one" end held in "many" end entity relation
1 : N	One Mandatory One Optional	2	*if the many end is mandatory:* one relation for each entity and identifier of the optional entity (the "one" end) held in the mandatory entity relation (the "many" end)
1 : N	One Mandatory One Optional	3	*if the many end is optional:* one relation for each entity and one for the relationship between them
1 : N	Both Optional	3	one relation for each entity and one for the relationship between them
M : N	Both Mandatory	3	one relation for each entity and one for the relationship between them
M : N	One Mandatory One Optional	3	one relation for each entity and one for the relationship between them
M : N	Both Optional	3	one relation for each entity and one for the relationship between them

Relationship set



- A relationship set id is mapped to a single relation (table).
- Simple case: relationship has no constraints (i.e. many-to-many)
- In this case, attributes of the table must include:
  - Keys for each participating entity set as foreign keys.
    - This is a *key* for the relation.
  - All descriptive attributes.

```
CREATE TABLE WorkOn(
  PID     CHAR(11),
  MID     INTEGER,
  Role    CHAR(20),
  PRIMARY KEY (PID, MID),
  FOREIGN KEY (PID)
     REFERENCES MoviePeople,
  FOREIGN KEY (MID)
     REFERENCES Movies)
```



A(a, d) R(a, b)
B(b, e) S(c, b)
C(c, f)

if NULL not allowed

AR(a, d, b)
BS(b, e, c)
C(c, f)

if NULL allowed

Relationship Sets to Tables (cont')

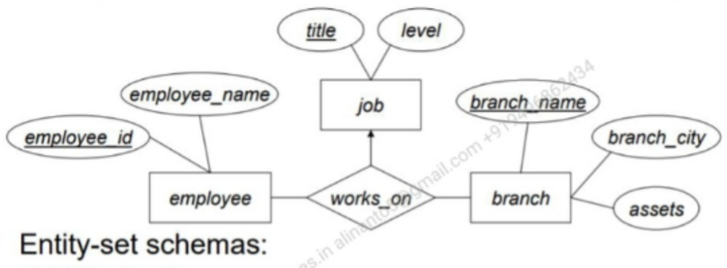
- In some cases, we need to use the roles:

```
CREATE TABLE Prerequisite(
  course_dept     CHAR(4),
  course_num      CHAR(3),
  prereq_dept     CHAR(4),
  prereq_num      CHAR(3),
  PRIMARY KEY (course_dept, course_num,
               prereq_dept, prereq_num),
  FOREIGN KEY (course_dept, course_num)
     REFERENCES Course(dept, num),
  FOREIGN KEY (prereq_dept, prereq_num)
```

N-ary Relationship Primary Keys

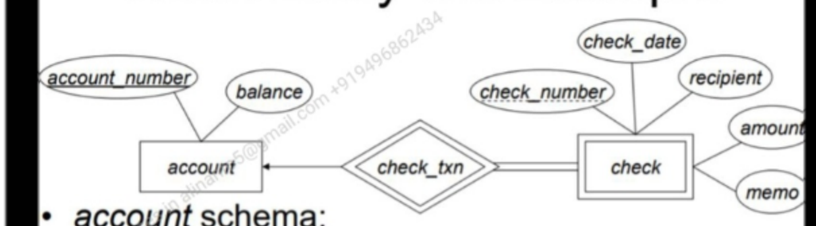
- For degree > 2 relationship-sets:
  - If no arrows ("many-to-many" mapping), relationship-set primary key is union of all participating entity-sets' primary keys
  - If one arrow ("one-to-many" mapping), relationship-set primary key is union of primary keys of entity-sets without an arrow
  - Don't allow more than one arrow for relationship-sets with degree > 2

N-ary Relationship-Set Example



- Entity-set schemas:
  job(*title*, *level*)
  employee(*employee_id*, *employee_name*)
  branch(*branch_name*, *branch_city*, *assets*)
- Relationship-set schema:
  - Primary key includes entity-sets on non-arrow links
    works_on(*employee_id*, *branch_name*, *title*)

Weak Entity-Set Example



- *account* schema:
  account(*account_number*, *balance*)
- *check* schema:
  - Discriminator is *check_number*
  - Primary key for *check* is:
    (*account_number*, *check_number*)

  check(*account_number*, *check_number*, *check_date*, *recipient*, *amount*, *memo*)

6.1 Er Diagram: GATE CSE 2005 | Question: 75 https://gateoverflow.in/1398

Let E_1 and E_2 be two entities in an E/R diagram with simple-valued attributes. R_1 and R_2 are two relationships between E_1 and E_2, where R_1 is one-to-many and R_2 is many-to-many. R_1 and R_2 do not have any attributes of their own. What is the minimum number of tables required to represent this situation in the relational model?

A. 2
B. 3
C. 4
D. 5