UNIT 2

(*) Database Design: Divided into 6 steps.

(i) Requirements Analysis:
① First step in designing a database application.
② Understand what data is to be stored in Database, what application must be built, what operations are performed.
③ We must find out what the user wants from the database.

(ii) Conceptual Database Design:
① The information gathered in the requirements analysis is used to develop high level description of data, along with all the constraints.
② Often carried out using ER Model.

(iii) Logical Database Design:
① The main aim is to convert the ER schema int relational database schema.

(iv) Schema Refinement:
① The next step is to analyze collection of relations in our relational database schema to identify problems and refine it.

(v) Physical Database Design:
① We must consider tasks that our database must support and further refine the database design to ensure it meets desired performance

(vi) Security Design:
① Identify the parts of database that a group can access and parts that it cannot access.
② Take steps to ensure that they can access only necessary parts.

1* Database Design Techniques: 2 different approaches

① Top-Down Design Methodology: It starts with major entities of their interest, their attributes and their relationships. And then we add other entities and add the relationships b/w these entities.

② Bottom up: It starts with a set of attributes. Group these attributes into entities and then find the relationship b/w these entities.

(*) ER Model

① Used to model the logical view of the system.
② Consists of these components: ⓐ Entity
ⓑ Entity Type
ⓒ Entity Set.

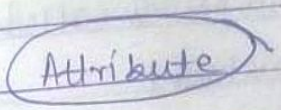
ⓐ Entity: May be an object with physical existence. Eg. a person, car, house, employee, etc.
Or it may be an object with conceptual existence: Eg. a company, a job, a university, etc.
→ Entity is a real world object that is easily identifiable.

ⓑ Entity Type: Entity is an object of Entity Type and set of all entities is called entity set.
eg: E1 is entity having Entity Type Student and set of all students is called Entity Set.

ⓒ Entity Set: Collection of similar type of entities. Need not be disjoint. Attributes can share similar values.

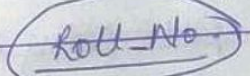
(*). Attributes and Constraints:

① Entities are represented by means of their properties called attributes.
② All attributes have values. Eg. a student entity will have name, class, age, etc. as attributes.
③ There exists a domain / Range of values that can be assigned to attributes. Eg. name cannot be numeric, age cannot be negative, etc.
④ Attributes are the properties that define entity type.



Types:

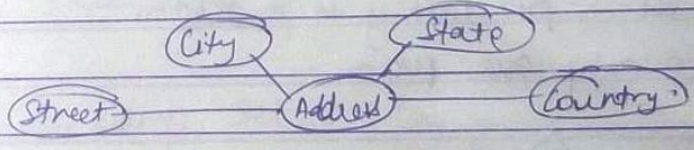
(a). Key attribute: The attribute which uniquely identifies each entity. For eg. Roll No will be unique for each student.

Representation: 

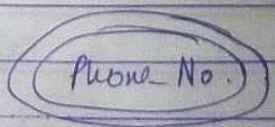
(b) Divided into:

① Simple Attribute: An atomic value that cannot be divided further. Eg. Phone no. is an atomic value of 10 digits.
② Single - value attributes: These contain single value. Eg

(b). Composite Attribute: Attributes composed of many other attributes. Eg. Address consists of Street, City, State, Country.



(c). Multivalued Attribute: Attribute consisting of more than one value for a given entity. Eg. Phone NO, email address, etc.



④ Derived Attributes: Can be derived from other attributes of the entity type. Eg. Age (Can be derived from DOB).
These attributes do not exist in physical database.

Age

(x) Relationship Type and Relationship Set

→ Relationship: The association among entities is called a relationship
Eg: An employee works at a department.

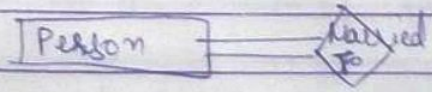
→ Relationship Set: A set of Relationships of Similar Type. These can also have attributes. These attributes are called descriptive attributes

→ Degree of Relationship: The number of participating entities.
Binary = degree 2, Ternary = degree 3, n-ary = degree n.

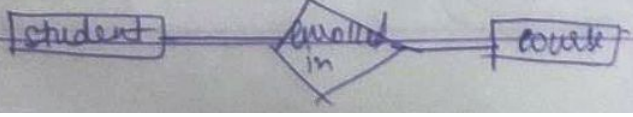
→ Relationship Type: Represents the association b/w Entity types.

→ Degree of Relationship Set: The no. of different entity sets participating in a relationship set.

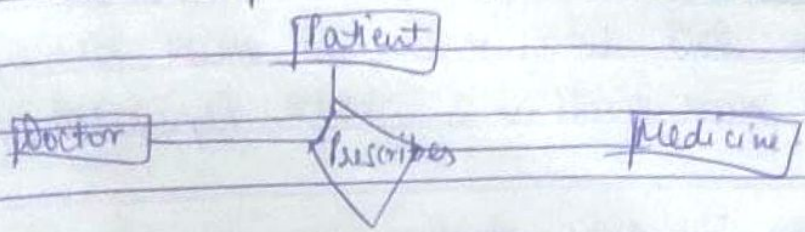
ⓐ UNARY: Only one entity set is participating in a relation.
Eg: one person is married to only one person.



ⓑ BINARY: Two entities participating in a relation.
Eg: Student enrolled in a course.



③. TERNARY: Relationship b/w 3 different entities Eg:

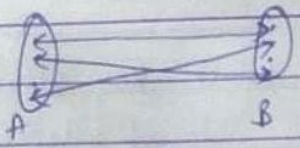


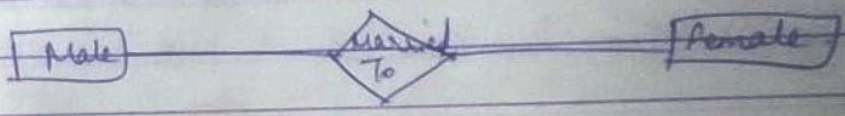
④. n-ary: when there are n-entities participating in a relation.

*). Constraints: Used for modelling limitations on relation b/w entities
2 types:

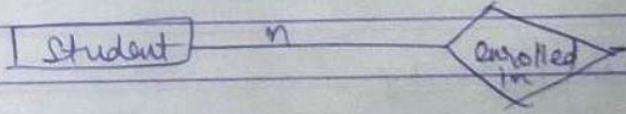
① Mapping Cardinality / Cardinality Ratio:
The no. of times an entity participates in a relation is called cardinality. It can be of different types:

→ One to one: Each entity can take part only once in the relationship.

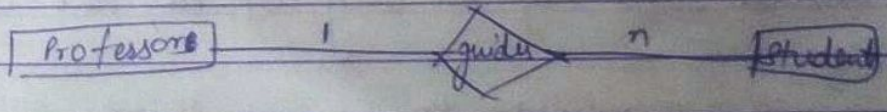


eg: 

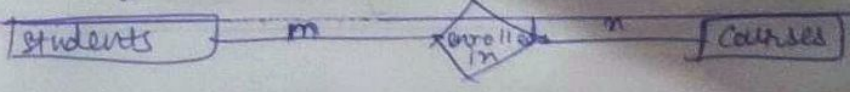
→ Many to One: Set A → More than one entities.
Set B → Only one entity.

eg. 

→ One to many: Set A → One entity
Set B → More than 1 entity



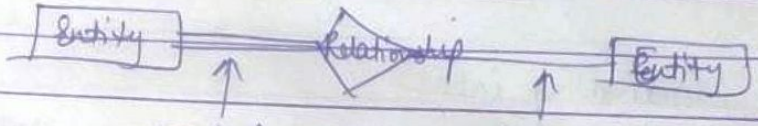
→ Many to many: Set A & Set B Both → More than 1 entity



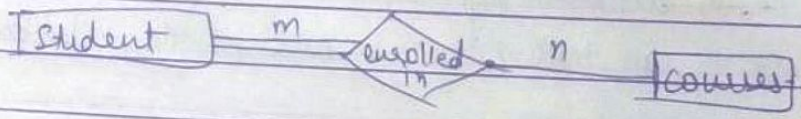
(b) Participation Constraint:

→ Total Participation: Each entity in the set must participate. Eg. Each student must enroll in a course. Represented by double lines.

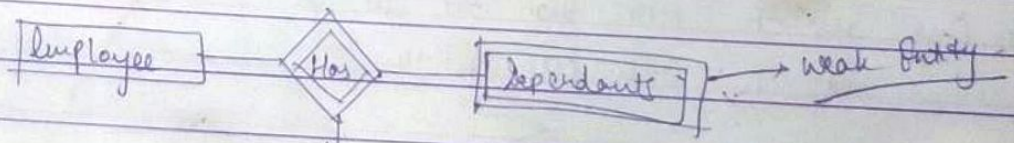
→ Partial Participation: The entity in the entity set may or may not participate in the relationship. Eg. Some courses are not enrolled by students.



Total Participation Partial Participation.



(c) Weak Entity: Some entity types for which key attribute cannot be defined. Eg. Company may store info of dependents (parents, children, spouse, etc). But Dependents don't have existence without employee.



Weak Entity Relationship

(d) KEY: (a) Super key: A set of attributes that collectively identifies an entity in Entity Set.

(b) Candidate key: Minimal Super key. An entity set may have more than one are candidate key.

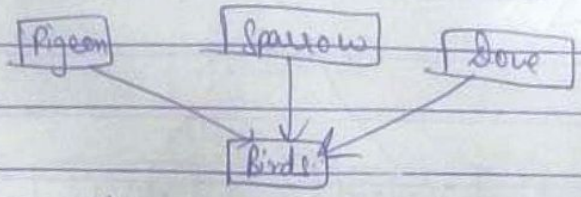
(c) Primary key: One of the candidate key uniquely chosen by database designer to uniquely identify the entity set.e.

*) Extended ER:

@ Generalization:

① A number of entities are brought together into one generalized entity based on their similar characteristics.

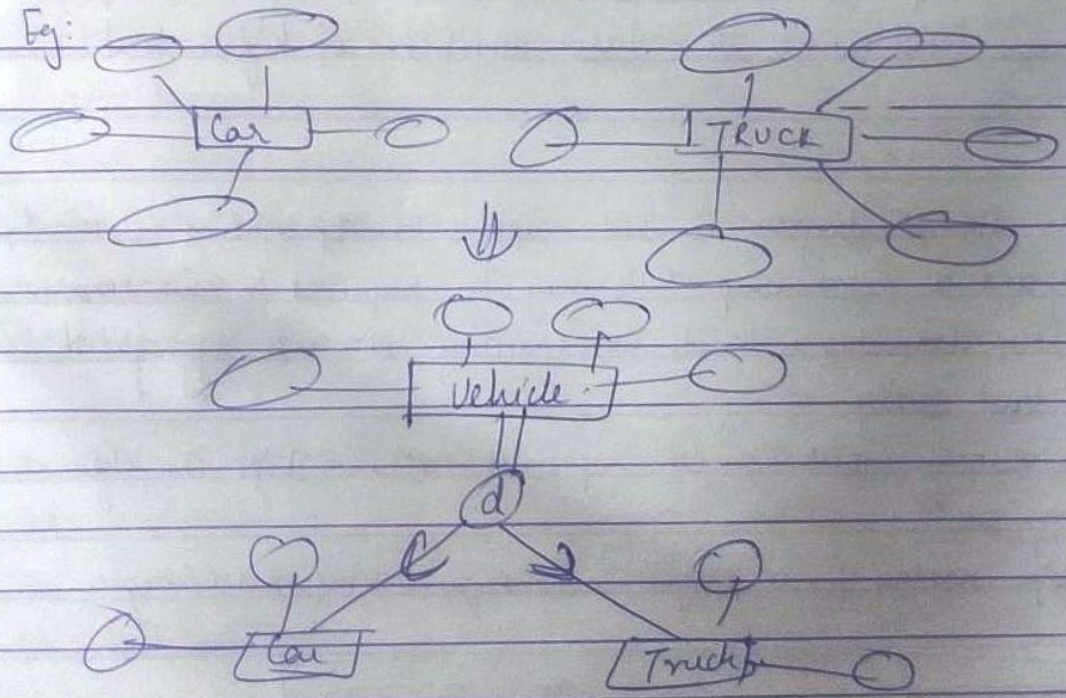
eg.



② It is a process of extracting common properties from a set of entities and create a generalized entity from it.

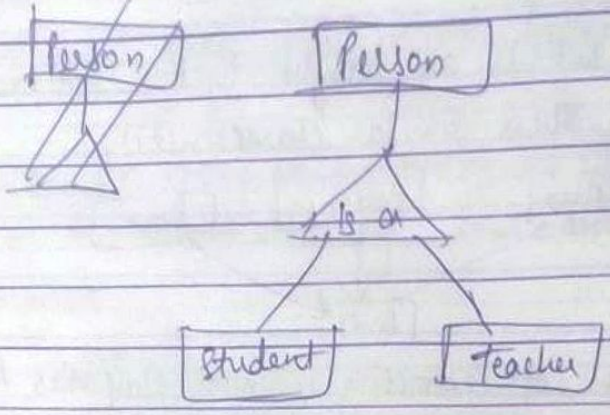
③ Bottom up approach where 2 or more entities can be generalized into a higher level entity.

Eg:



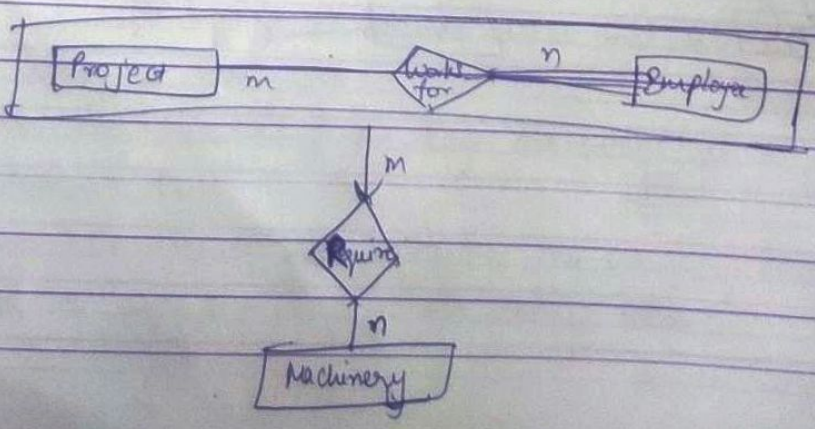
⑥ Specialization: Opp. of Generalization.
→ A group of entities is divided into sub-groups.

Eg.



→ Inheritance: The details of entities are generally hidden from the user. This process is known as abstraction.
Inheritance is imp. feature of Generalization & Specialization. It allows lower level entities to inherit attributes of higher level entities.

⑦ Aggregation: An ER diagram is not capable of representing relationship b/w entity and a relationship which may be required in some scenario. In those cases, relationship with its corresponding entities is aggregated into higher level entity.
→ Abstraction through which we can represent relationships as higher level entity sets.
Eg. An employee working for a project may require some machinery.



(x). ER Design Issues:

① Use of entity set Vs Attributes: It leads to a mistake when the user use the primary key of an entity set as an attribute of another entity set. Instead he should use relationship to do so.

② Use of Entity Sets Vs Relationship sets: It is difficult to examine if an object can be best expressed by entity or relationship set.

③ Use of Binary Vs n-ary Relationship sets: It is possible to represent a non-binary set by dividing it into n binary sets.

④ Placing Relationship Attributes: It is better to associate the attributes of one to one or one to many relationship sets with any participating entity sets instead of relationship set if it requires overall knowledge of each part.

(x). Relational Model:

→ It is a data Model based on predicate logic and set theory.
→ The fundamental assumption is that all data are represented as mathematical n-ary relations.

ⓐ Relation: Two dimensional table made up of rows & columns. Each relation is also called a table, stores data about entities
ⓑ Tuples: The rows in the relation. Represent specific occurances of entities.
ⓒ Attributes: Columns in a relation. Represent Characteristics of entities.

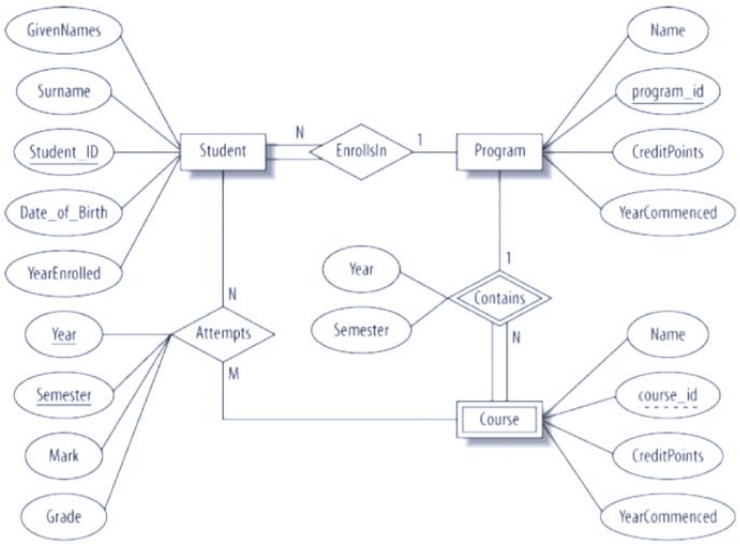
④ Domain: Set of permitted values for an attribute.
Domain is said to be atomic

Database schema: logical design of database.
Database Instance: Snapshot of data in database at a given instant of time.
Relation Schema: Definition of a domain of values. DB schema is a collection of Relational schemas that define a database.
Relation Instance: Corresponds to the programming language notation of a value of a variable.
Schema Diagram: A DB schema along with primary key & foreign key dependencies can be depicted pictorially by schema diagrams.
Relational algebra: Consists of a set of operations that take one or two relations as I/P and produce a new relation as result.
Fundamental Operations → select, project, union, set difference, Cartesian product, rename.

Thus, it requires the overall knowledge of each part that is involved inb desgining and modelling an ER diagram. The basic requirement is to analyse the real-world enterprise and the connectivity of one entity or attribute with other.

EXAMPLE

Question: Draw an ER Diagram for University Database?



HINT:

The university database stores details about university students, courses, the semester a student took a particular course (and his mark and grade if he completed it), and what degree program each student is

enrolled in. The database is a long way from one that'd be suitable for a large tertiary institution, but it does illustrate relationships that are interesting to query, and it's easy to relate to when you're learning SQL. We explain the requirements next and discuss their shortcomings at the end of this section.
Consider the following requirements list:

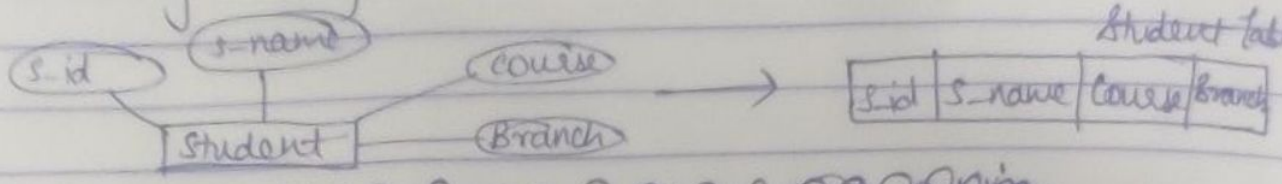
- The university offers one or more programs.
- A program is made up of one or more courses.
- A student must enroll in a program.
- A student takes the courses that are part of her program.
- A program has a name, a program identifier, the total credit points required to graduate, and the year it commenced.
- A course has a name, a course identifier, a credit point value, and the year it commenced.
- Students have one or more given names, a surname, a student identifier, a date of birth, and the year they first enrolled. We can treat all given names as a single object—for example, "John Paul."
- When a student takes a course, the year and semester he attempted it are recorded. When he finishes the course, a grade (such as A or B) and a mark (such as 60 percent) are recorded.
- Each course in a program is sequenced into a year (for example, year 1) and a semester (for example, semester 1).

In our design:

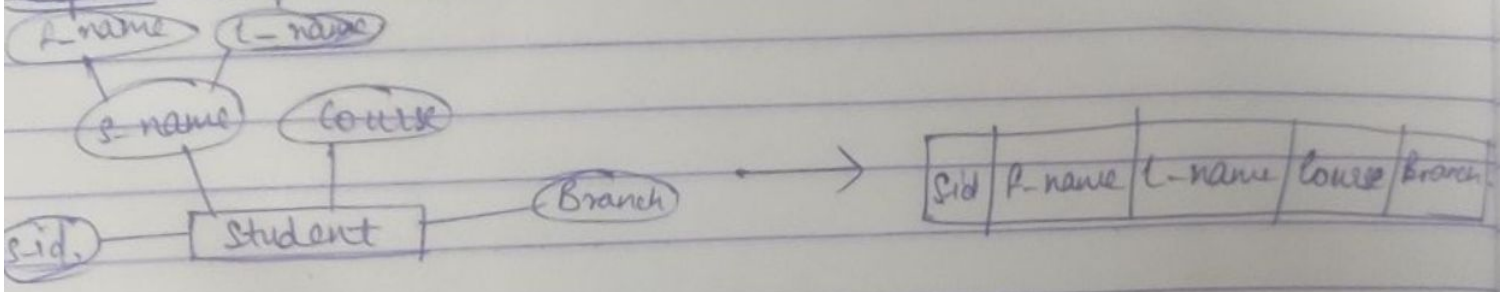
- Student is a strong entity, with an identifier, student_id, created to be the primary key used to distinguish between students (remember, we could have several students with the same name).
- Program is a strong entity, with the identifier program_id as the primary key used to distinguish between programs.
- Each student must be enrolled in a program, so the Student entity participates totally in the many-to-one EnrollsIn relationship with Program. A program can exist without having any enrolled students, so it participates partially in this relationship.
- A Course has meaning only in the context of a Program, so it's a weak entity, with course_id as a weak key. This means that a Course is uniquely identified using its course_id and the program_id of its owning program.
- As a weak entity, Course participates totally in the many-to-one identifying relationship with its owning Program. This relationship has Year and Semester attributes that identify its sequence position.
- Student and Course are related through the many-to-many Attempts relationships; a course can exist without a student, and a student can be enrolled without attempting any courses, so the participation is not total.
- When a student attempts a course, there are attributes to capture the Year and Semester, and the Mark and Grade.

Conversion of ER Diagram to Relational Table:

Step 1: Converting Entity Sets to Tables:

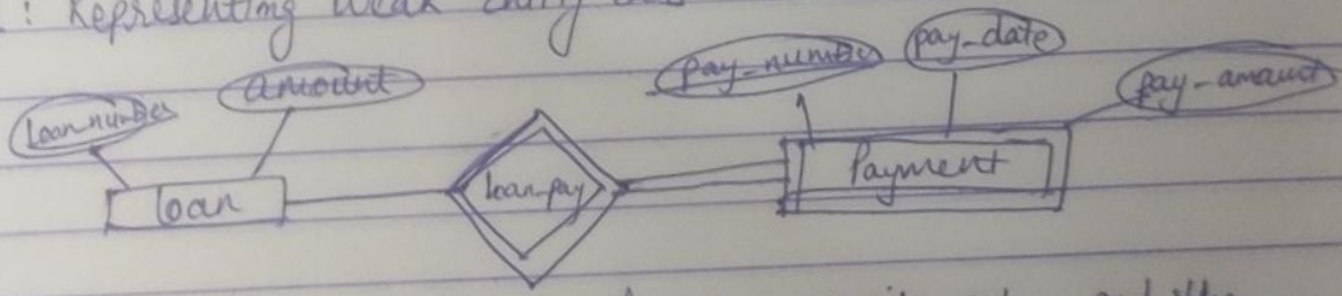


Step 2: Composite and Multivalued Attributes:



Make f_name and l_name as individual Attributes/Columns in the Relational Table.

Step 3: Representing Weak Entity Sets as tables.



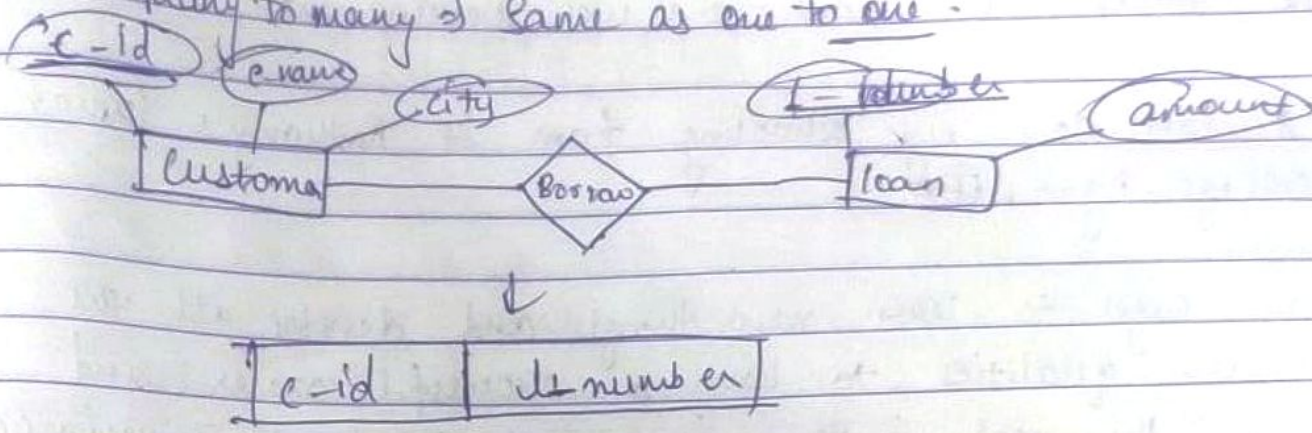
We include all the attributes of Weak entity set and the primary key of strong entity set.

loan_number	pay_number	pay_date	pay_amount

Representing Relationship Sets:

⇒ One to one → take primary keys of both the participating tables.

⇒ Many to many ⇒ same as one to one.



⇒ One to Many / Many to One: We take all the attributes of the entity which is in total participation & primary key of the entity which is in partial participation.