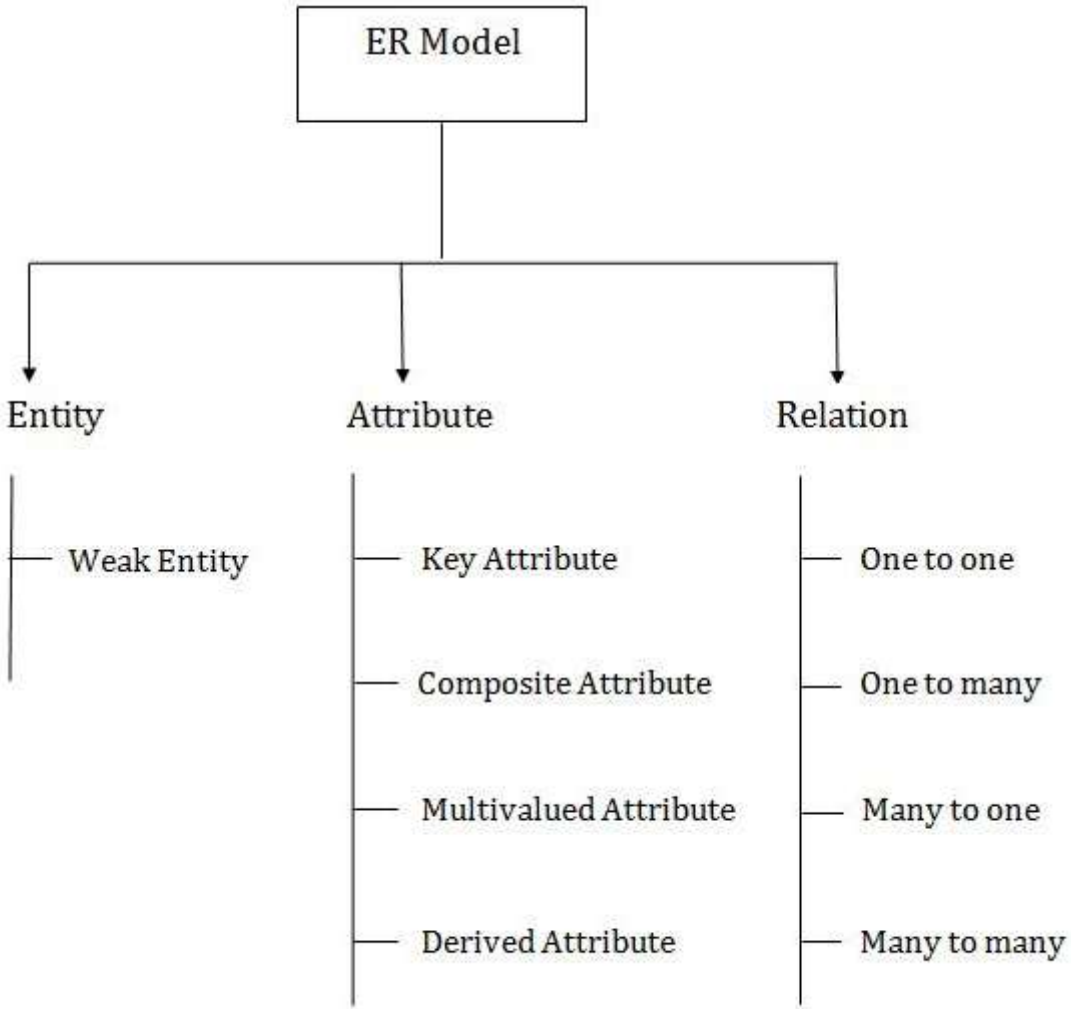
ER Model

Unit II

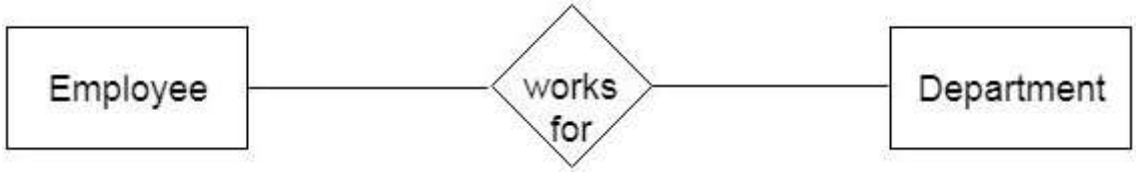
ER model

- ER model stands for an Entity-Relationship model.
- It is a high-level data model.
- This model is used to define the data elements and relationship for a specified system.
- It develops a conceptual design for the database.
- It also develops a very simple and easy to design view of data.
- In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

Components of ER diagram



- Entity:
  - An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.
  - Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.



  - **Weak Entity**

- An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own.
- The weak entity is represented by a double rectangle.

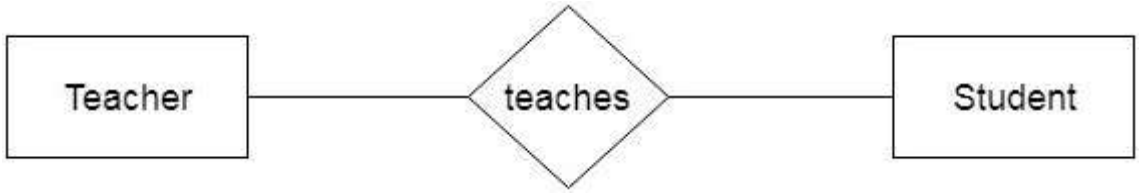
- Eg1: The existence of rooms is entirely dependent on the existence of a hotel. So room can be seen as the weak entity of the hotel.
- The bank account of a particular bank has no existence if the bank doesn’t exist anymore.

Difference between Strong and weak entity

Strong Entity Set	Weak Entity Set
Strong entity set always has a primary key.	It does not have enough attributes to build a primary key.
It is represented by a rectangle symbol.	It is represented by a double rectangle symbol.
It contains a Primary key represented by the underline symbol.	It contains a Partial Key which is represented by a dashed underline symbol.
The member of a strong entity set is called as dominant entity set.	The member of a weak entity set called as a subordinate entity set.
Primary Key is one of its attributes which helps to identify its member.	In a weak entity set, it is a combination of primary key and partial key of the strong entity set.
In the ER diagram the relationship between two strong entity set shown by using a diamond symbol.	The relationship between one strong and a weak entity set shown by using the double diamond symbol.
The connecting line of the strong entity set with the relationship is single.	The line connecting the weak entity set for identifying relationship is double.

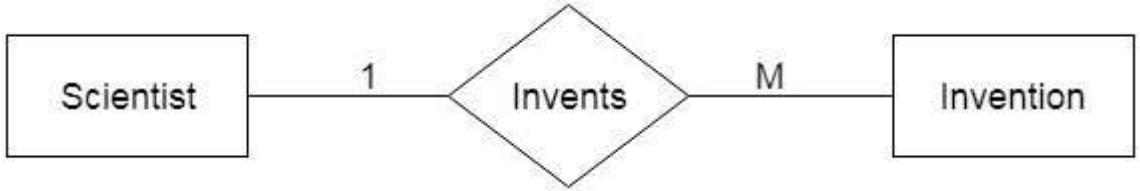
Relationship

- A relationship is used to describe the relation between entities.
- Diamond or rhombus is used to represent the relationship.

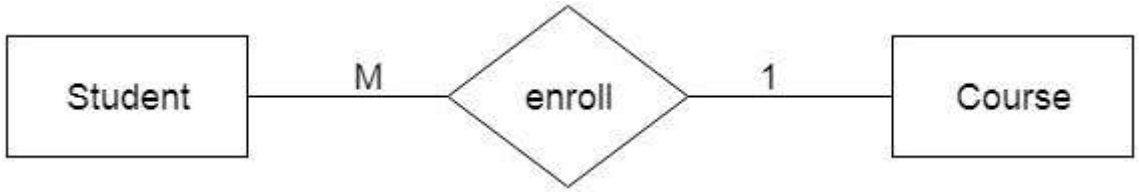


- Types
  - **One-to-One Relationship:** When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

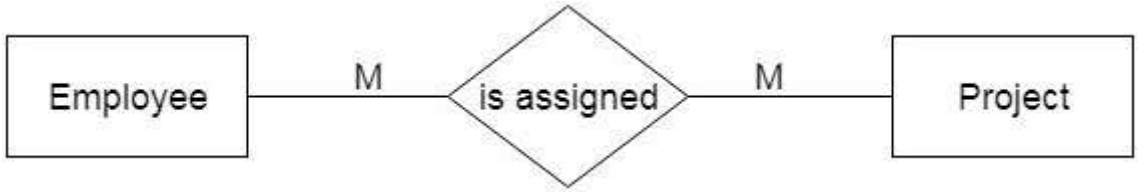
  - **One-to-many relationship:** When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.



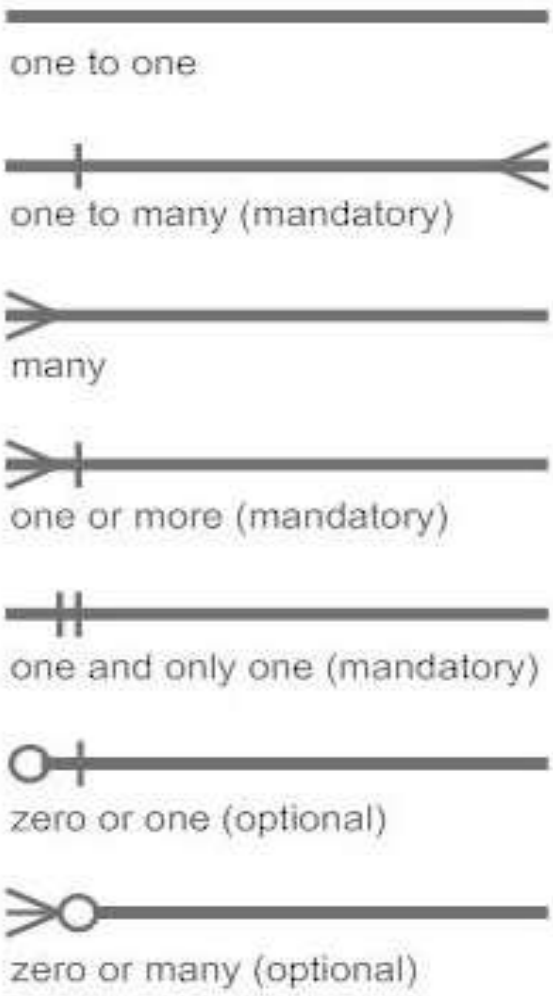
- **Many-to-one relationship:** When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

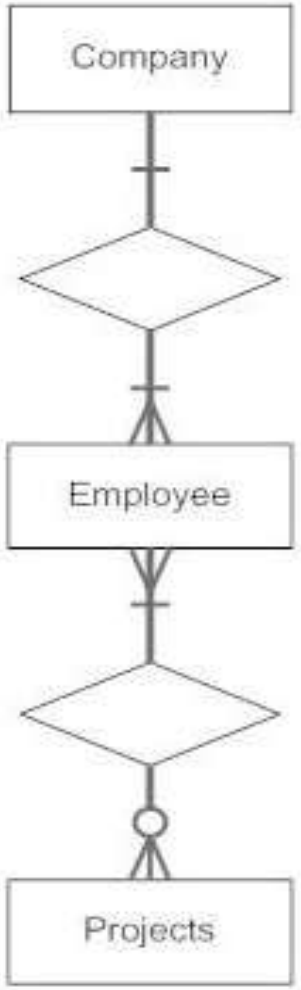


- **Many-to-many relationship:** When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.



Notation of ER diagram





Entity Sets *customer* and *loan*

customer-id	customer-name	customer-street	customer-city
321-12-3123	Jones	Main	Harrison
019-28-3746	Smith	North	Rye
677-89-9011	Hayes	Main	Harrison
555-55-5555	Jackson	Dupont	Woodside
244-66-8800	Curry	North	Rye
963-96-3963	Williams	Nassau	Princeton
335-57-7991	Adams	Spring	Pittsfield

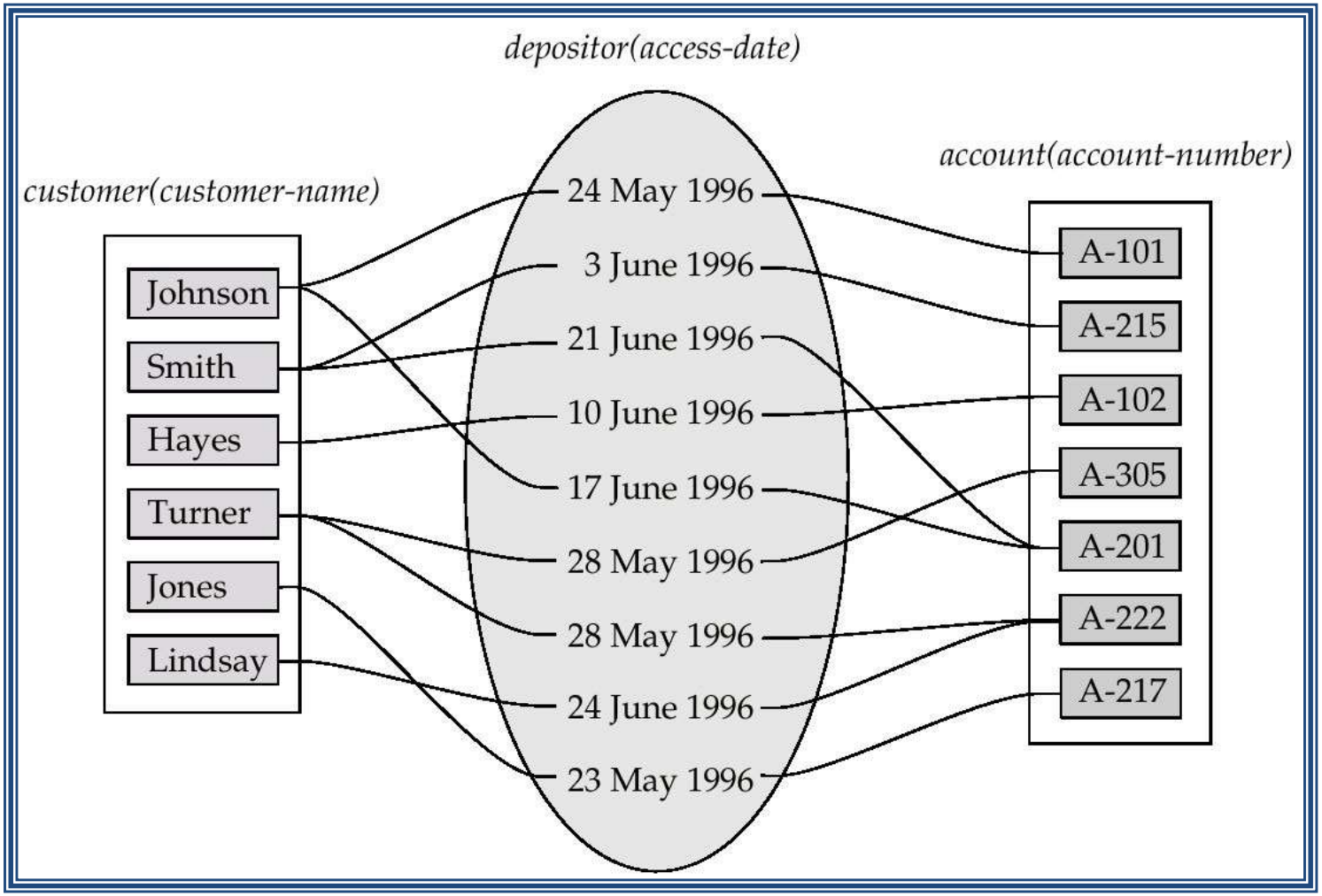
customer

loan-number	amount
L-17	1000
L-23	2000
L-15	1500
L-14	1500
L-19	500
L-11	900
L-16	1300

loan

Relationship Sets (Cont.)

- An *attribute* can also be property of a relationship set.
- For instance, the *depositor* relationship set between entity sets *customer* and *account* may have the attribute *access-date*



Relationship Sets

- A relationship is an association among several entities

  Example:

  Hayes *depositor* A-102

  customer entity relationship set *account* entity

- A *relationship* set is a mathematical relation among $n \geq 2$ entities, each taken from entity sets

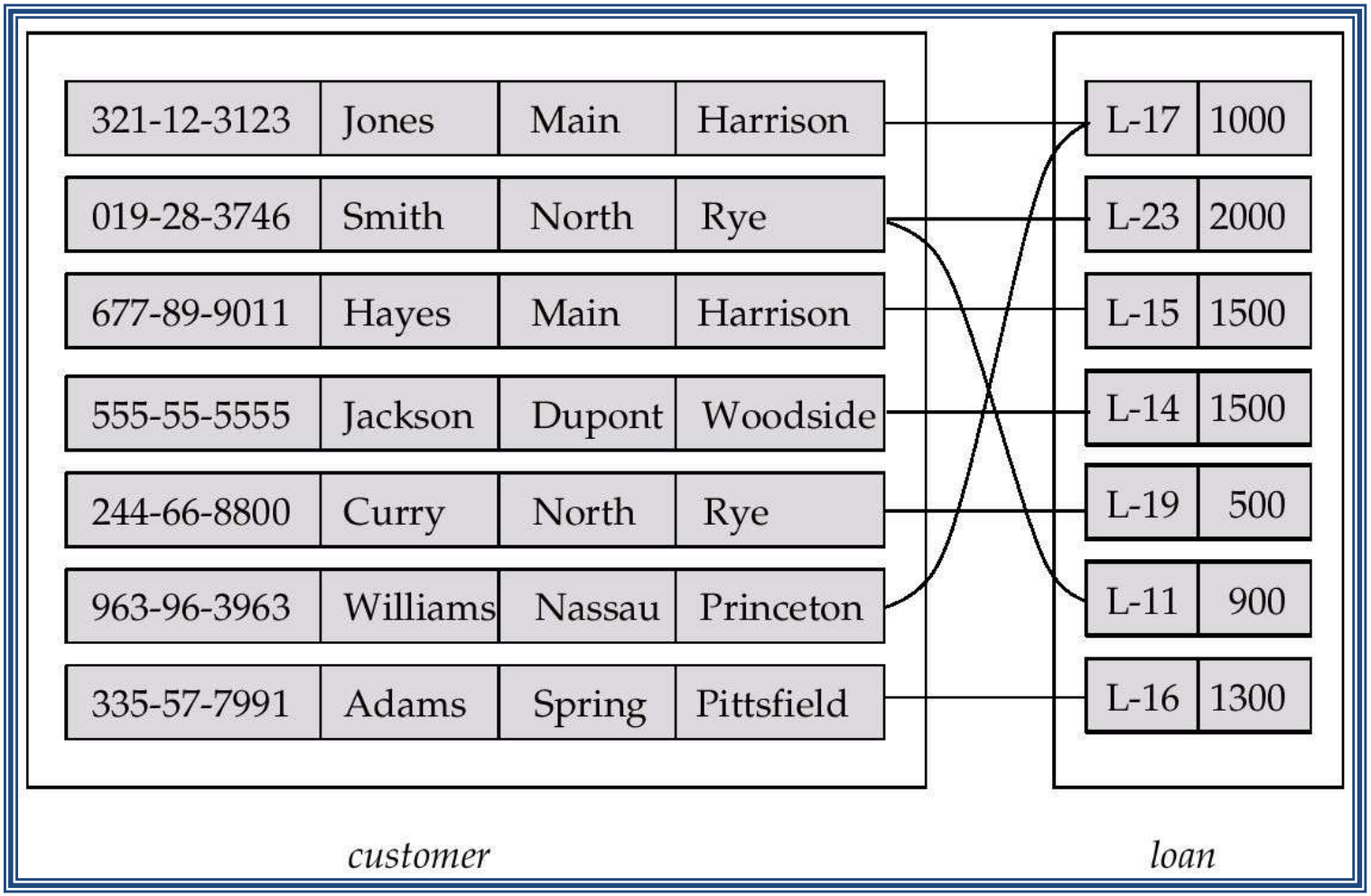
$$\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$

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

  – Example:

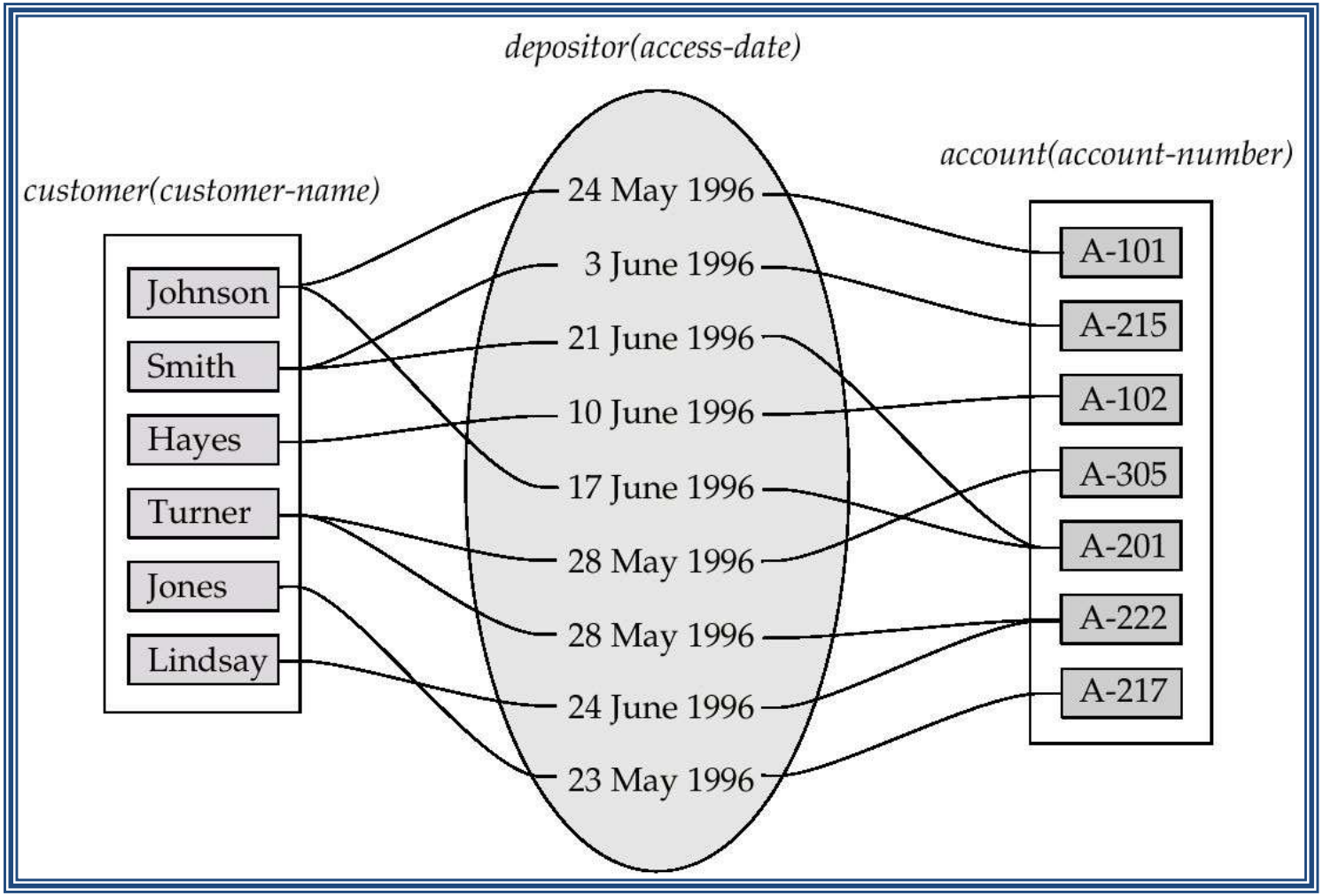
  (Hayes, A-102) $\in$ *depositor*

Relationship Set *borrower*



Relationship Sets (Cont.)

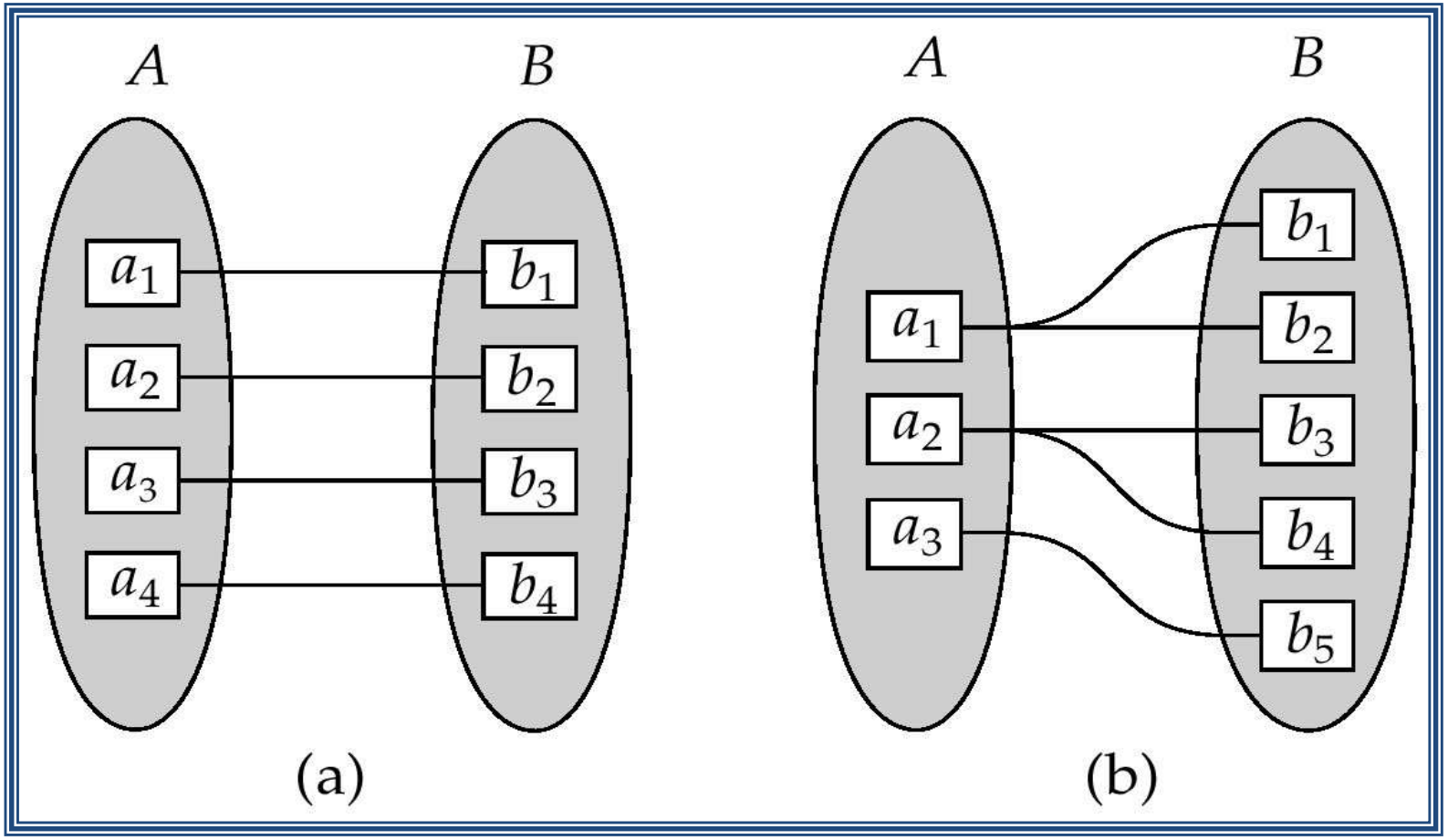
- An *attribute* can also be property of a relationship set.
- For instance, the *depositor* relationship set between entity sets *customer* and *account* may have the attribute *access-date*



Degree of a Relationship Set

- Refers to number of entity sets that participate in a relationship set.
- Relationship sets that involve two entity sets are *binary* (or degree two). Generally, most relationship sets in a database system are binary.
- Relationship sets may involve more than two entity sets.
  - E.g. Suppose employees of a bank may have jobs (responsibilities) at multiple branches, with different jobs at different branches. Then there is a ternary relationship set between entity sets *employee, job and branch*
- Relationships between more than two entity sets are rare. Most relationships are binary.

Mapping Cardinalities/Constraints

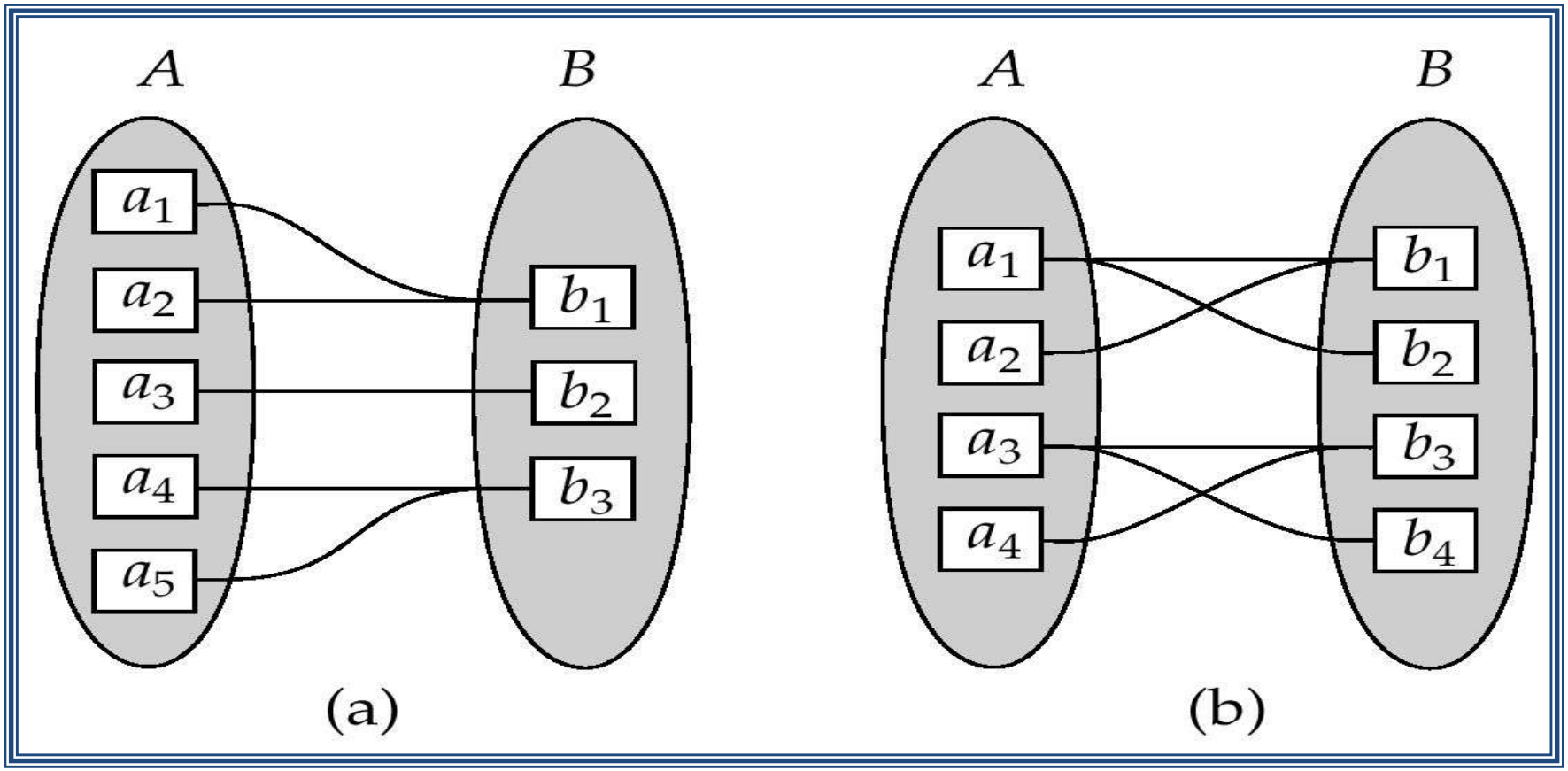


One to one

One to many

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

Mapping Cardinalities/Constraints

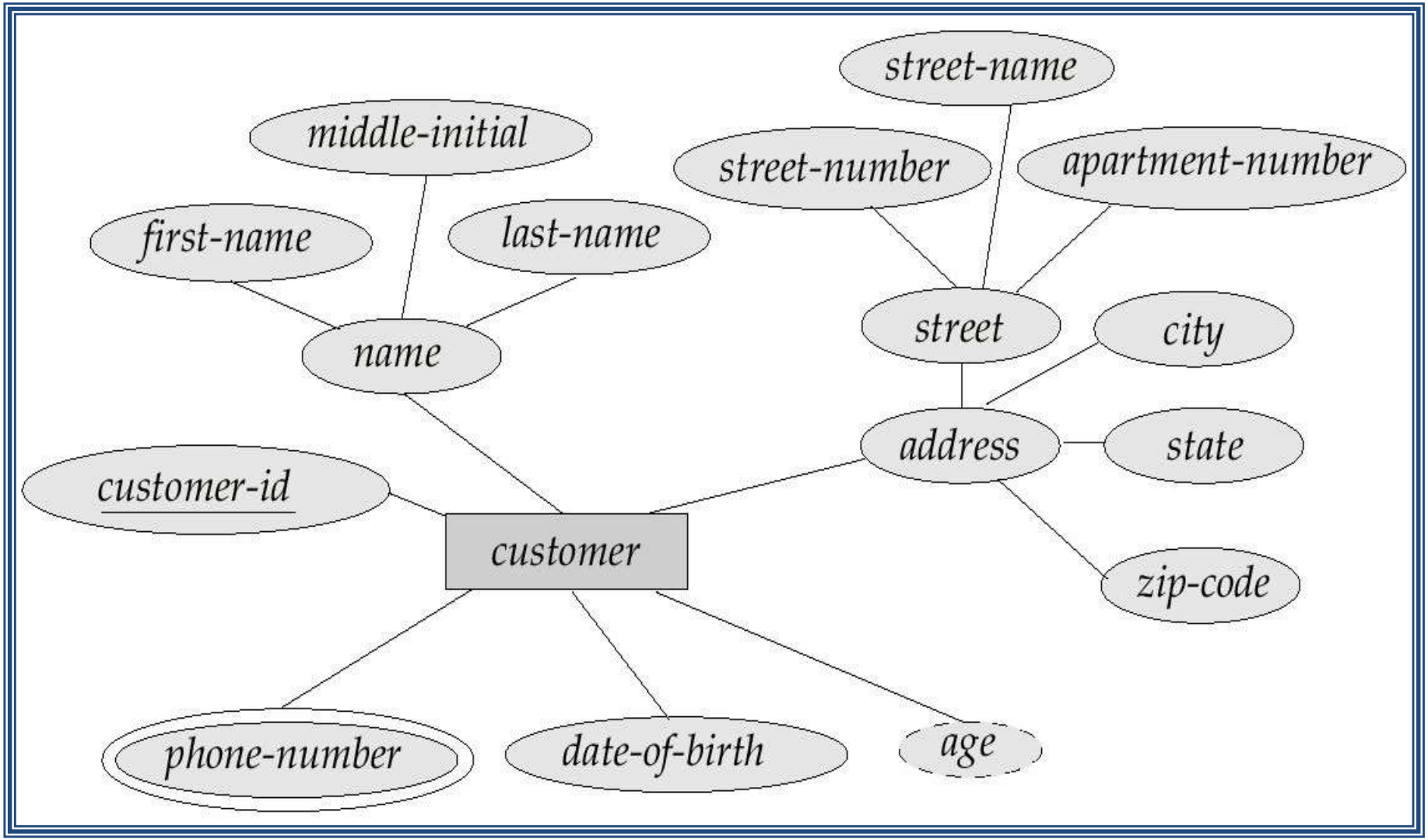


Many to one

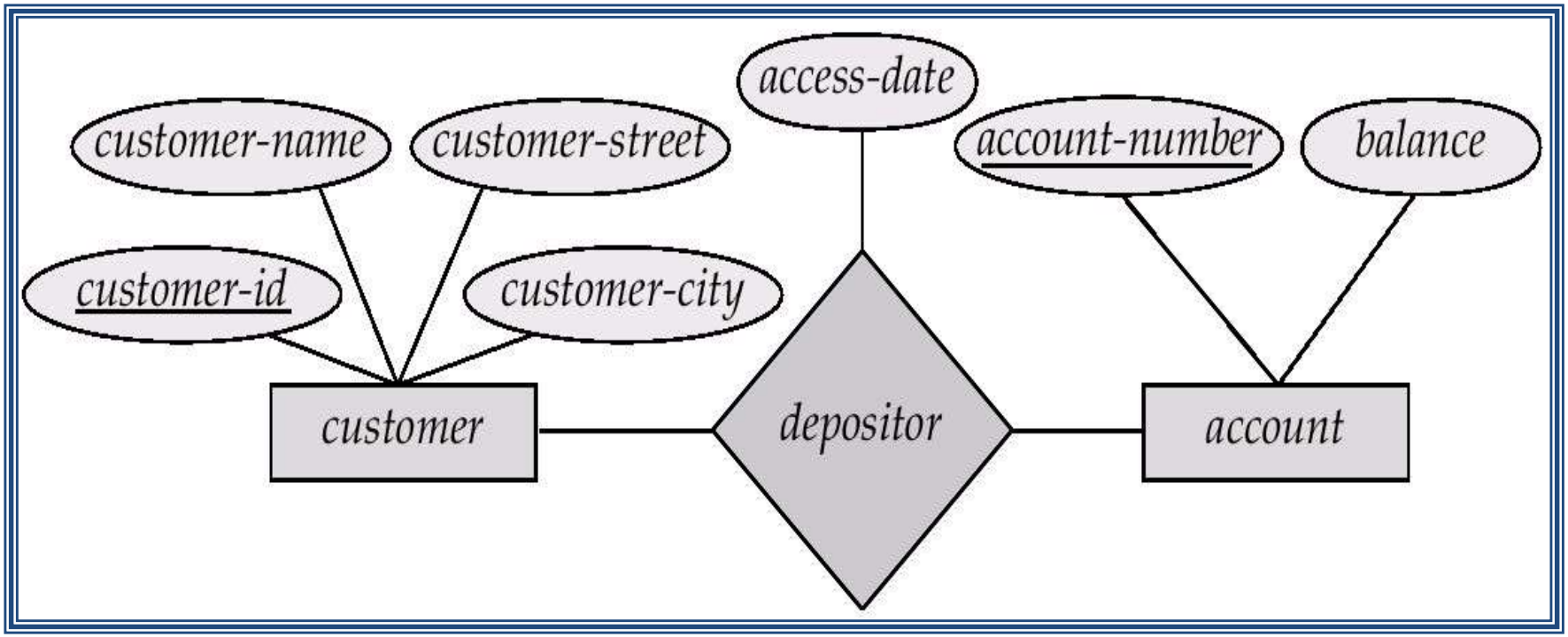
Many to many

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

E-R Diagram With Composite, Multivalued, and Derived Attributes

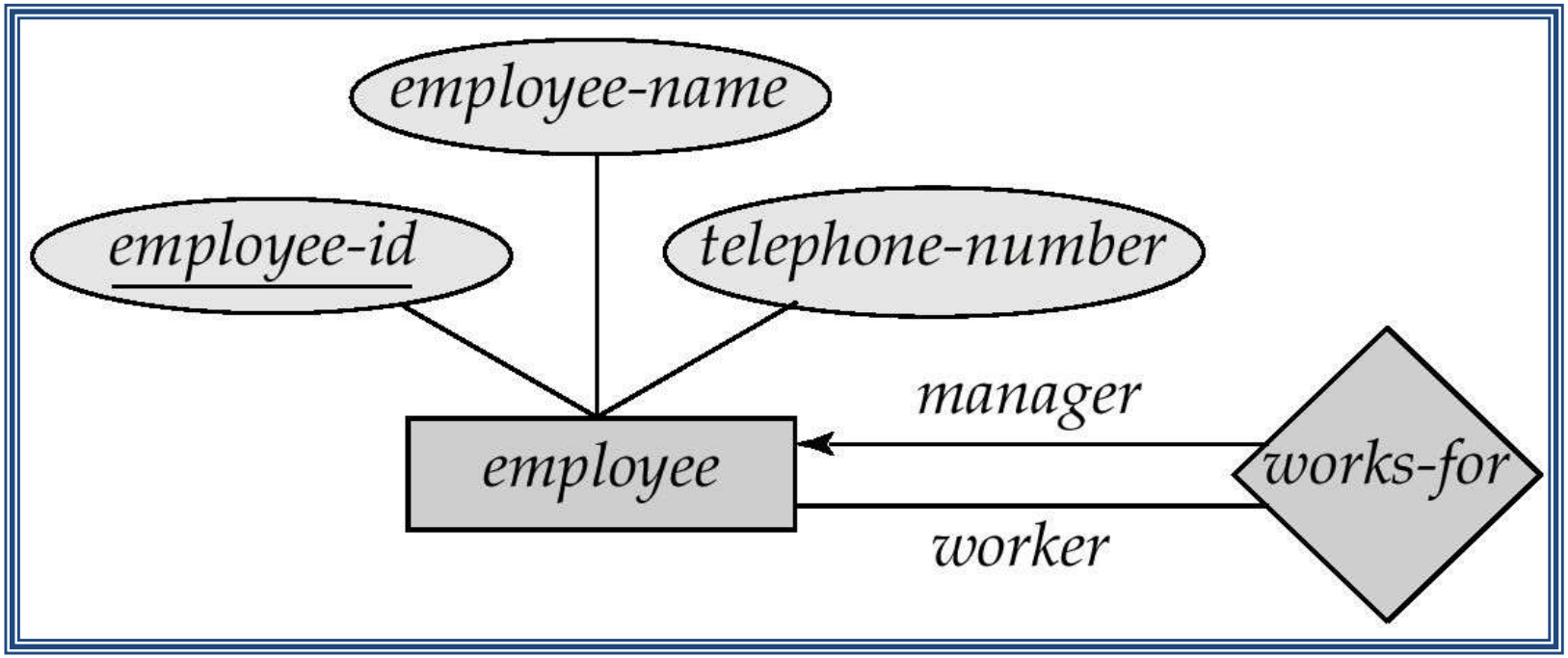


Relationship Sets with Attributes



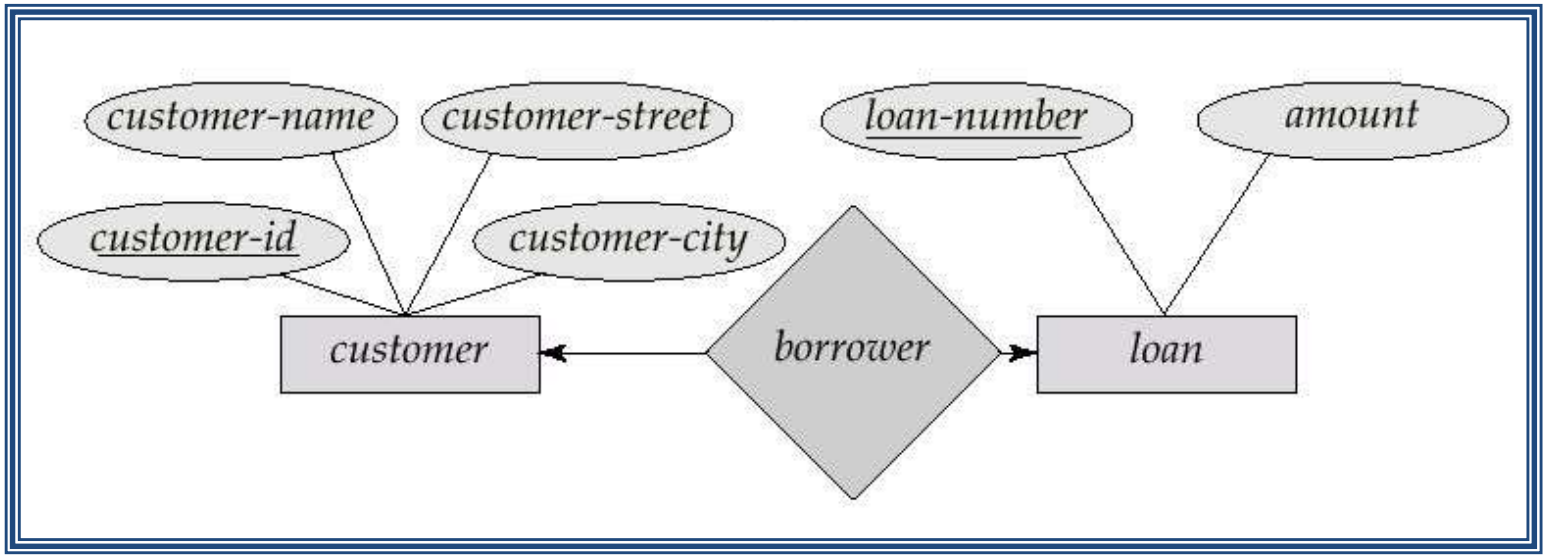
Roles

- Entity sets of a relationship need not be distinct
- The labels “manager” and “worker” are called roles; they specify how employee entities interact via the works-for relationship set.
- Roles are indicated in E-R diagrams by labeling the lines that connect diamonds to rectangles.
- Role labels are optional, and are used to clarify semantics of the relationship



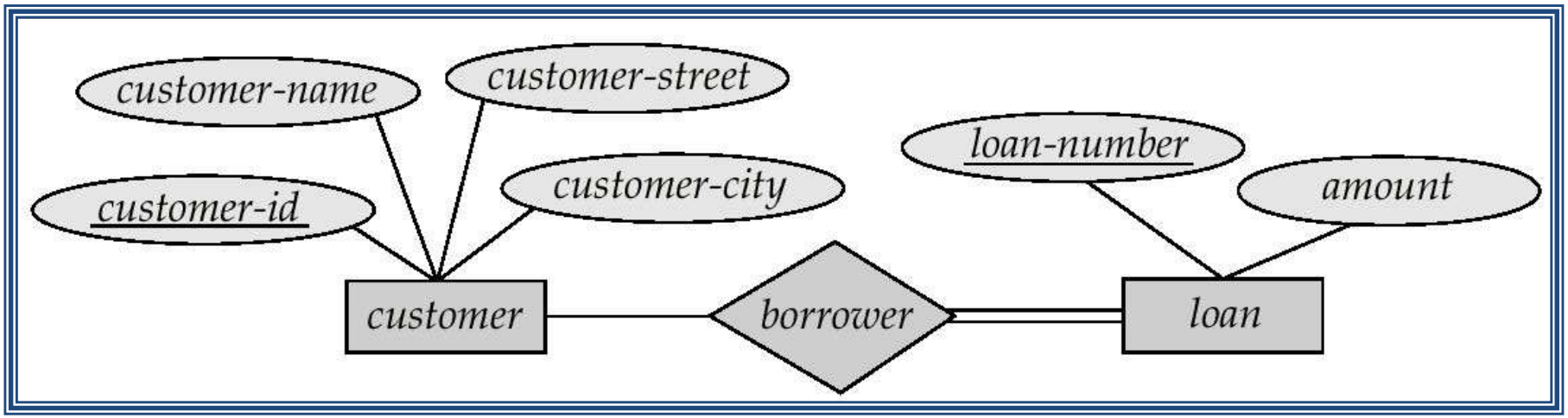
Cardinality Constraints

- We express cardinality constraints by drawing either a directed line (→), signifying "one," or an undirected line (—), signifying "many," between the relationship set and the entity set.
- E.g.: One-to-one relationship:
  - A customer is associated with at most one loan via the relationship *borrower*
  - A loan is associated with at most one customer via *borrower*



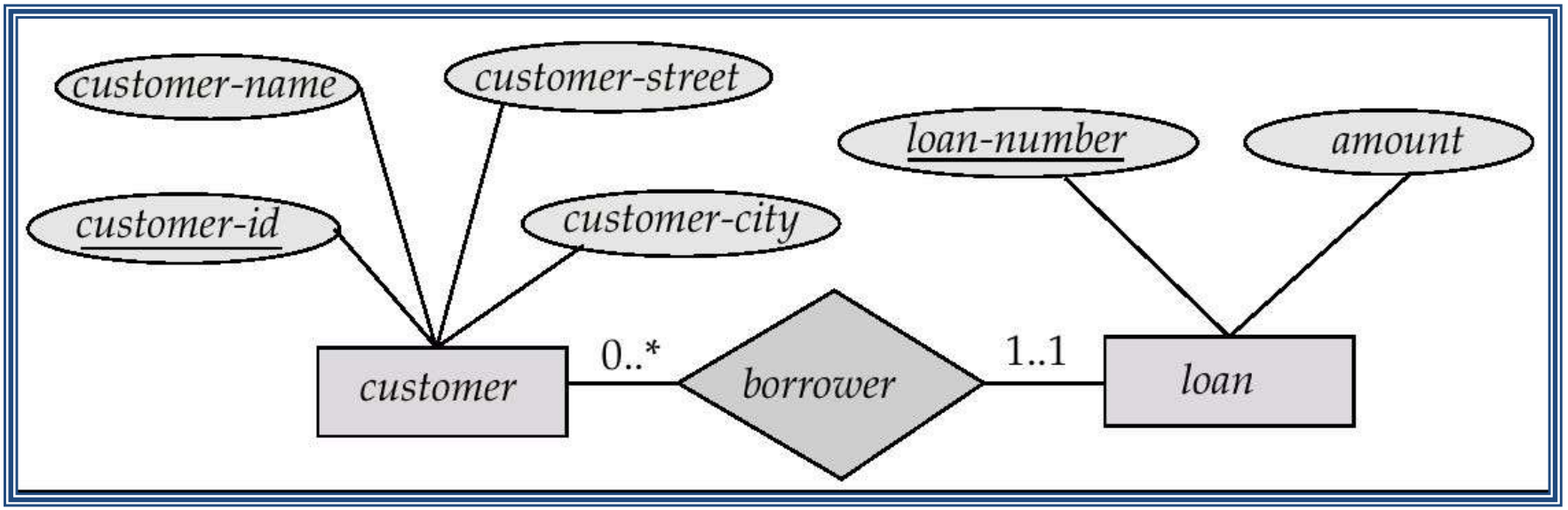
Participation of an Entity Set in a Relationship Set

- Total participation (indicated by double line): every entity in the entity set participates in at least one relationship in the relationship set
  - E.g. participation of *loan* in *borrower* is total
    - every loan must have a customer associated to it via borrower
- Partial participation: some entities may not participate in any relationship in the relationship set
  - E.g. participation of *customer* in *borrower* is partial

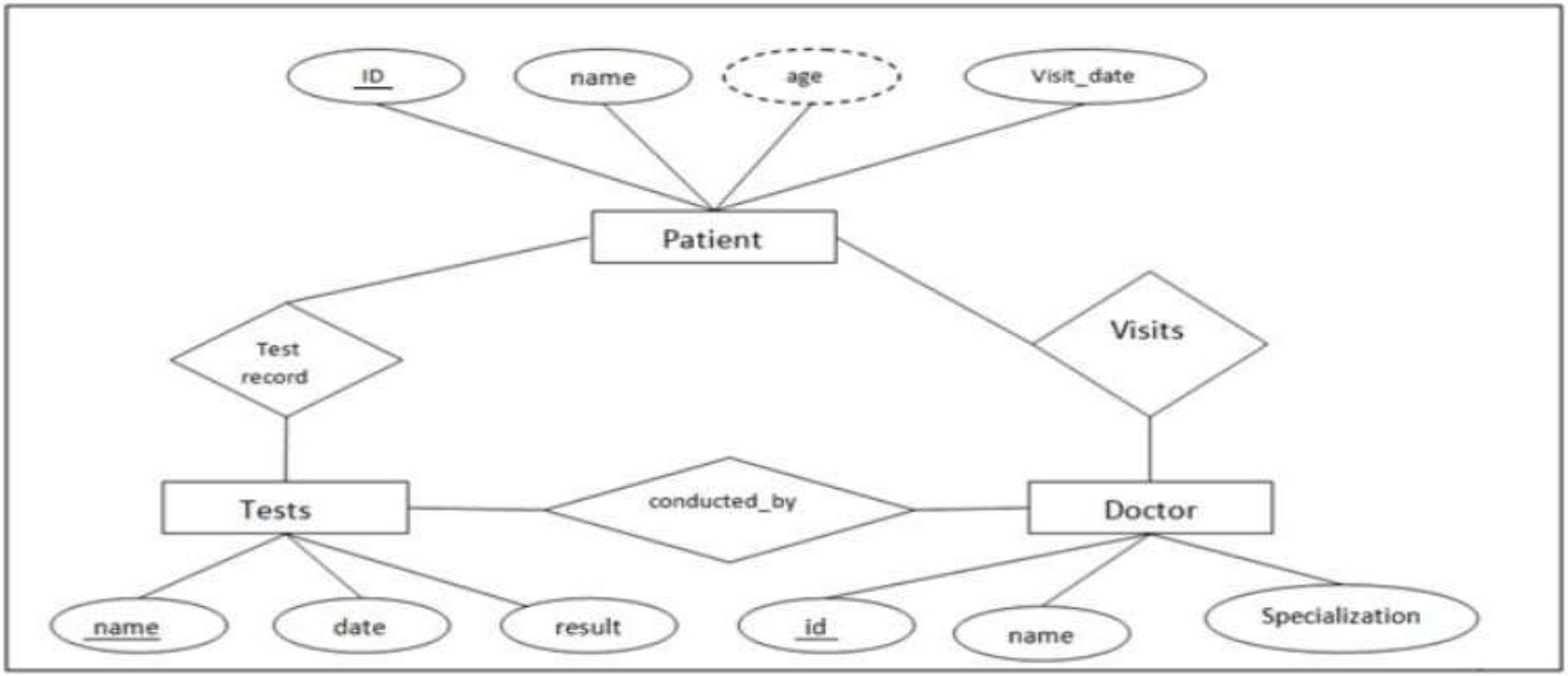


Alternative Notation for Cardinality Limits

- Cardinality limits can also express participation constraints

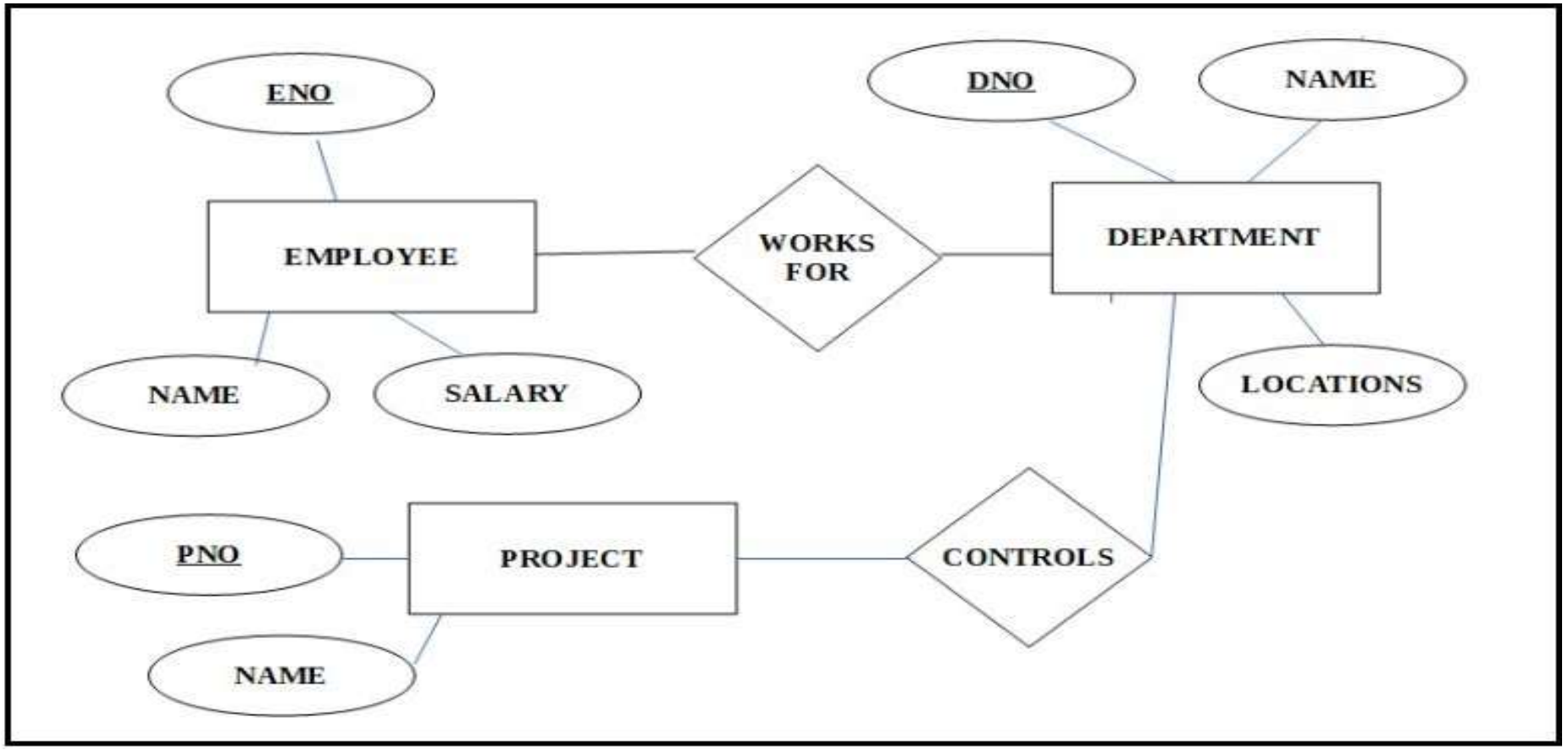


Hospital ER Model



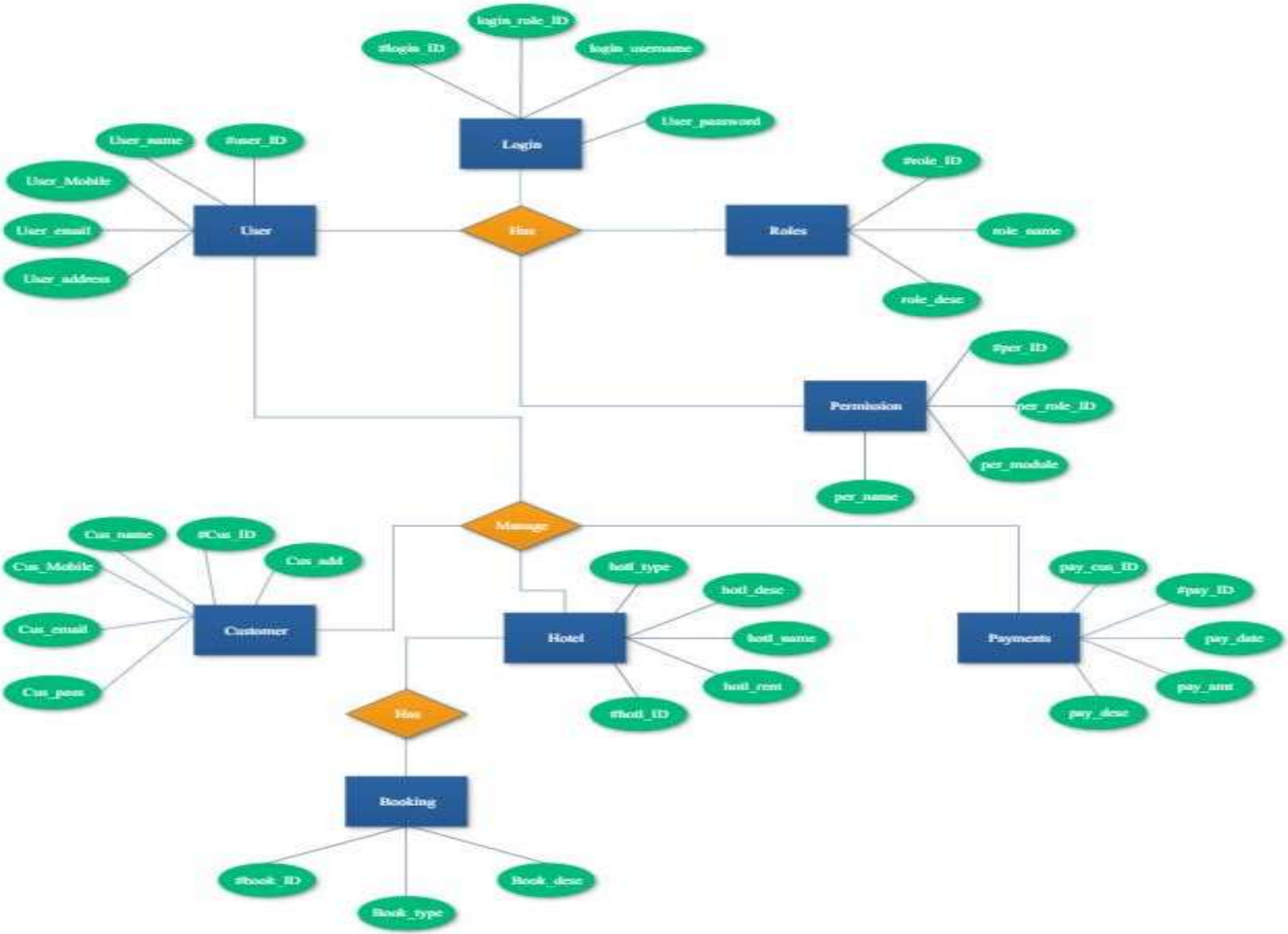
Each of these entities have their respective attributes which are −
Patients - ID(primary key), name, age, visit_date
Tests- Name(primary key), date, result
Doctor- ID(primary key), name, specialization
The relationships between different entities are represented by a diamond shaped box.

Company ER Model

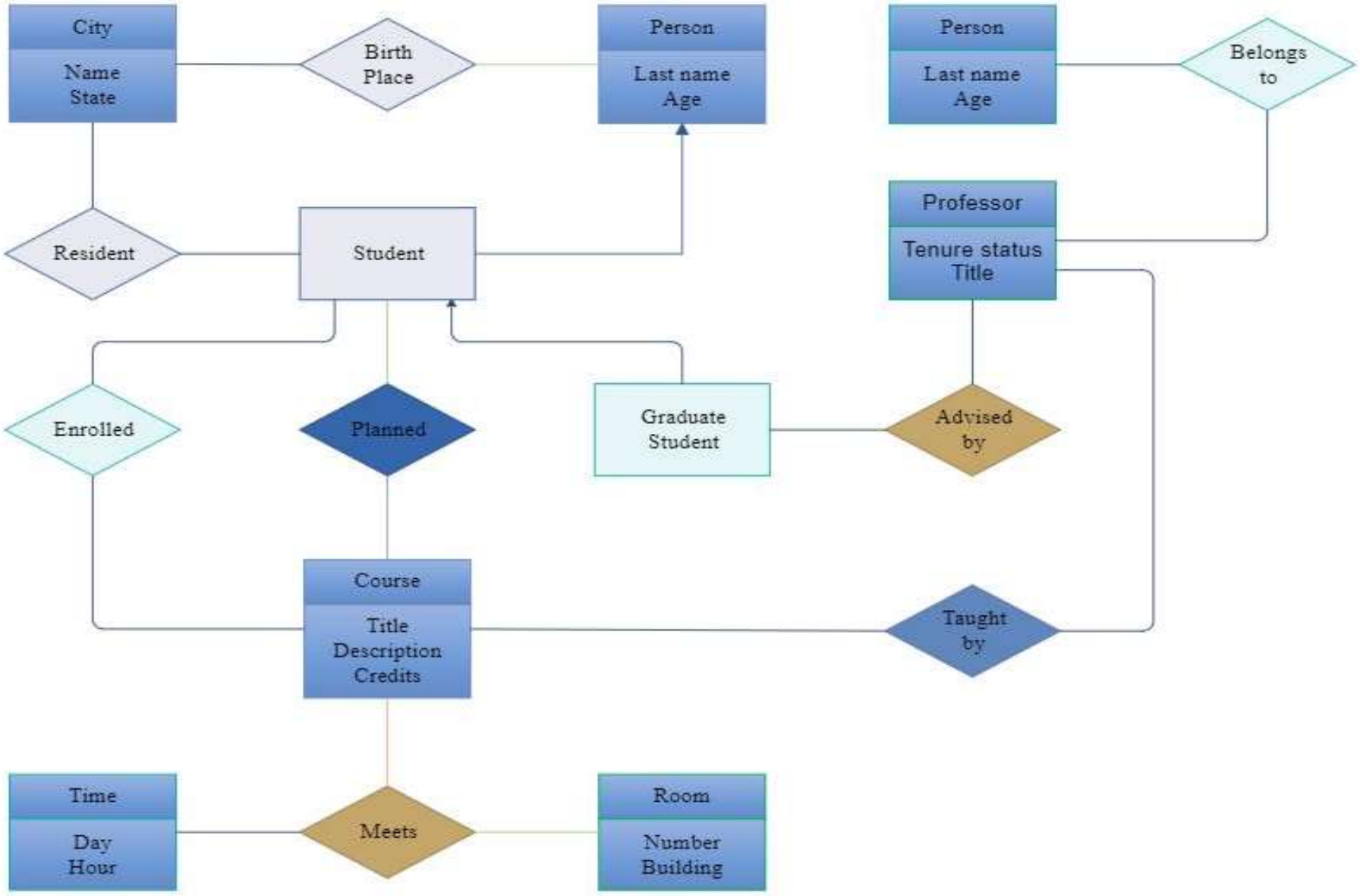


- Employee - ENO(Primary Key) , Name, Salary
- Department - DNO(Primary key), Name, Locations
- Project - PNO(Primary key), Name

ER diagram for Hotel Management System

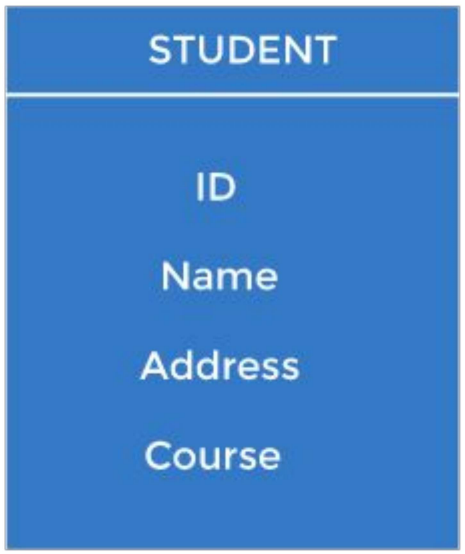


University Database



Keys

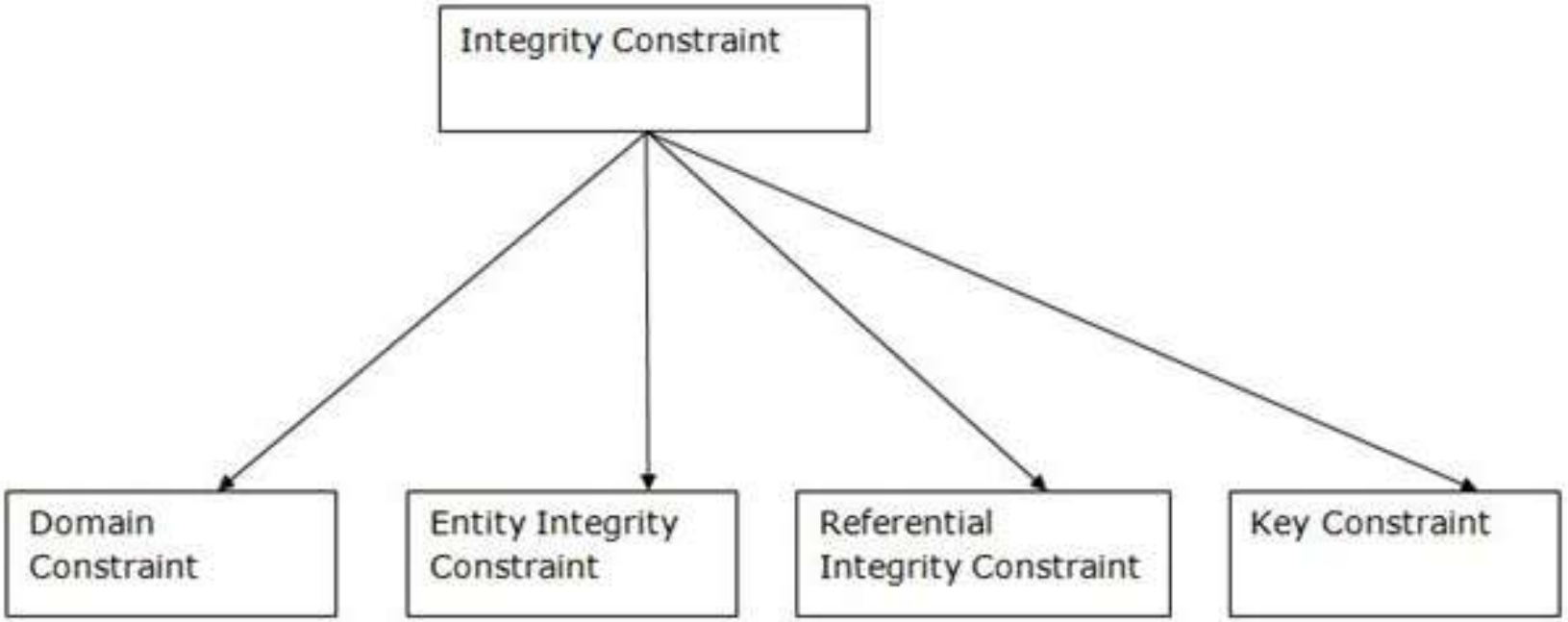
- Keys play an important role in the relational database.
- It is used to uniquely identify any record or row of data from the table. It is also used to establish and identify relationships between tables.



Constraints

- Constraints are the rules enforced on the data columns of a table.
- These are used to limit the type of data that can go into a table.
- This ensures the accuracy and reliability of the data in the database.

Types of Constraint



Types of Constraint

- Domain constraints
  - Domain constraints can be defined as the definition of a valid set of values for an attribute.
  - The data type of domain includes string, character, integer, time, date, currency, etc. The value of the attribute must be available in the corresponding domain.

ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1004	Morgan	8th	A

Not allowed. Because AGE is an integer attribute

Types of Constraint

- Entity integrity constraints
  - The entity integrity constraint states that primary key value can't be null.
  - This is because the primary key value is used to identify individual rows in relation and if the primary key has a null value, then we can't identify those rows.
  - A table can contain a null value other than the primary key field.

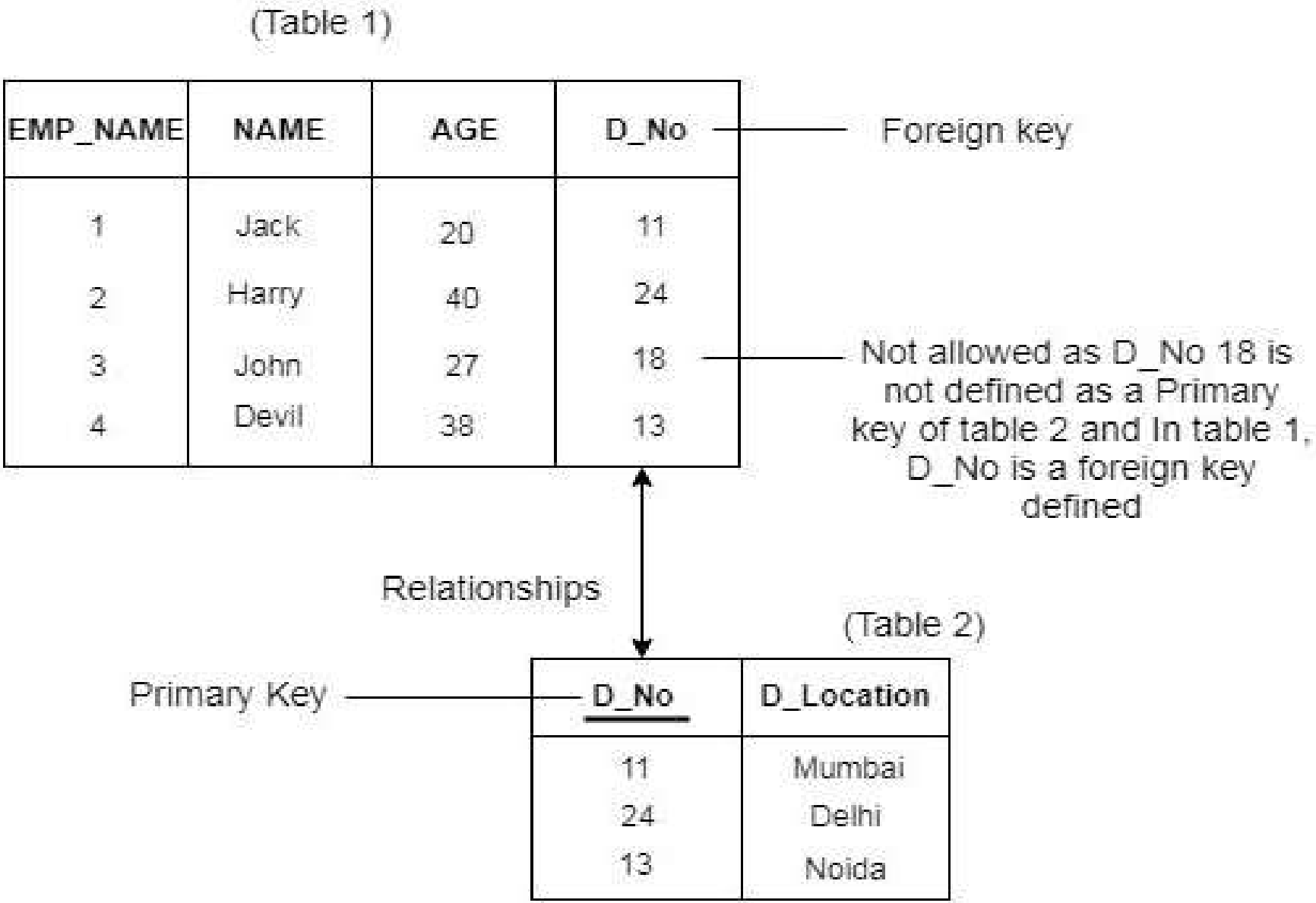
EMPLOYEE

EMP_ID	EMP_NAME	SALARY
123	Jack	30000
142	Harry	60000
164	John	20000
	Jackson	27000

Not allowed as primary key can't contain a NULL value

Referential Integrity Constraints

- A referential integrity constraint is specified between two tables.
- In the Referential integrity constraints, if a foreign key in Table 1 refers to the Primary Key of Table 2, then every value of the Foreign Key in Table 1 must be null or be available in Table 2.

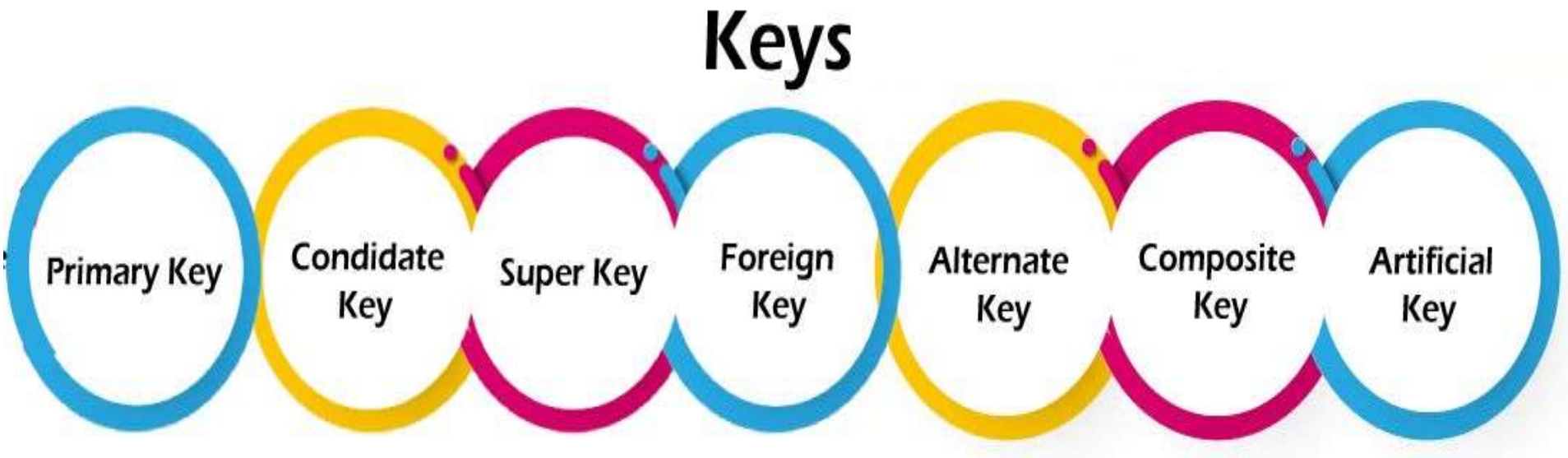

(Table 1)
EMP_NAME
NAME
AGE
D_No
Foreign key
1
Jack
20
11
2
Harry
40
24
3
John
27
18
Not allowed as D_No 18 is not defined as a Primary key of table 2 and In table 1, D_No is a foreign key defined
4
Devil
38
13
Relationships
(Table 2)
Primary Key
D_No
D_Location
11
Mumbai
24
Delhi
13
Noida

- Key constraints
  - Keys are the entity set that is used to identify an entity within its entity set uniquely.
  - An entity set can have multiple keys, but out of which one key will be the primary key. A primary key can contain a unique and null value in the relational table.

ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1002	Morgan	8th	22

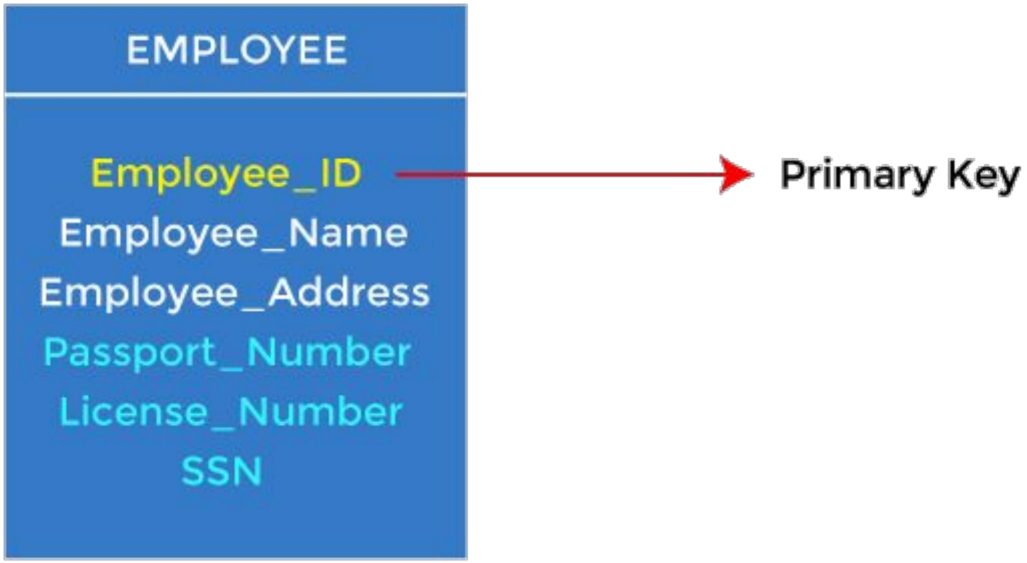
Not allowed. Because all row must be unique

Types of keys:



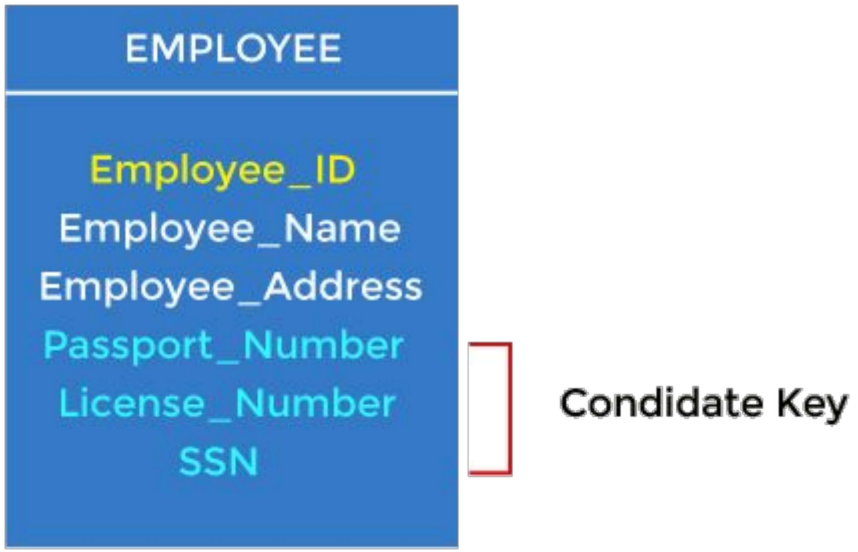
Primary key

- It is the first key used to identify one and only one instance of an entity uniquely.
- In the EMPLOYEE table, ID can be the primary key since it is unique for each employee.
- In the EMPLOYEE table, we can even select License_Number and Passport_Number as primary keys since they are also unique.
- For each entity, the primary key selection is based on requirements and developers.



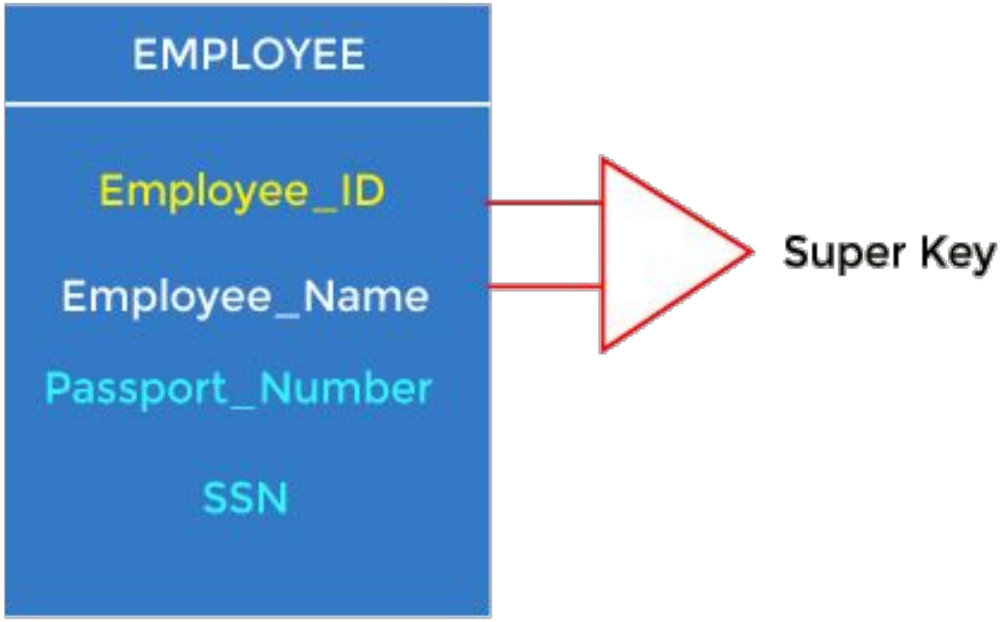
Candidate key

- A candidate key is an attribute or set of attributes that can uniquely identify a tuple.
- Except for the primary key, the remaining attributes are considered a candidate key.
- The candidate keys are as strong as the primary key.
- **For example:** In the EMPLOYEE table, id is best suited for the primary key. The rest of the attributes, like SSN, Passport_Number, License_Number, etc., are considered a candidate key.



Super Key

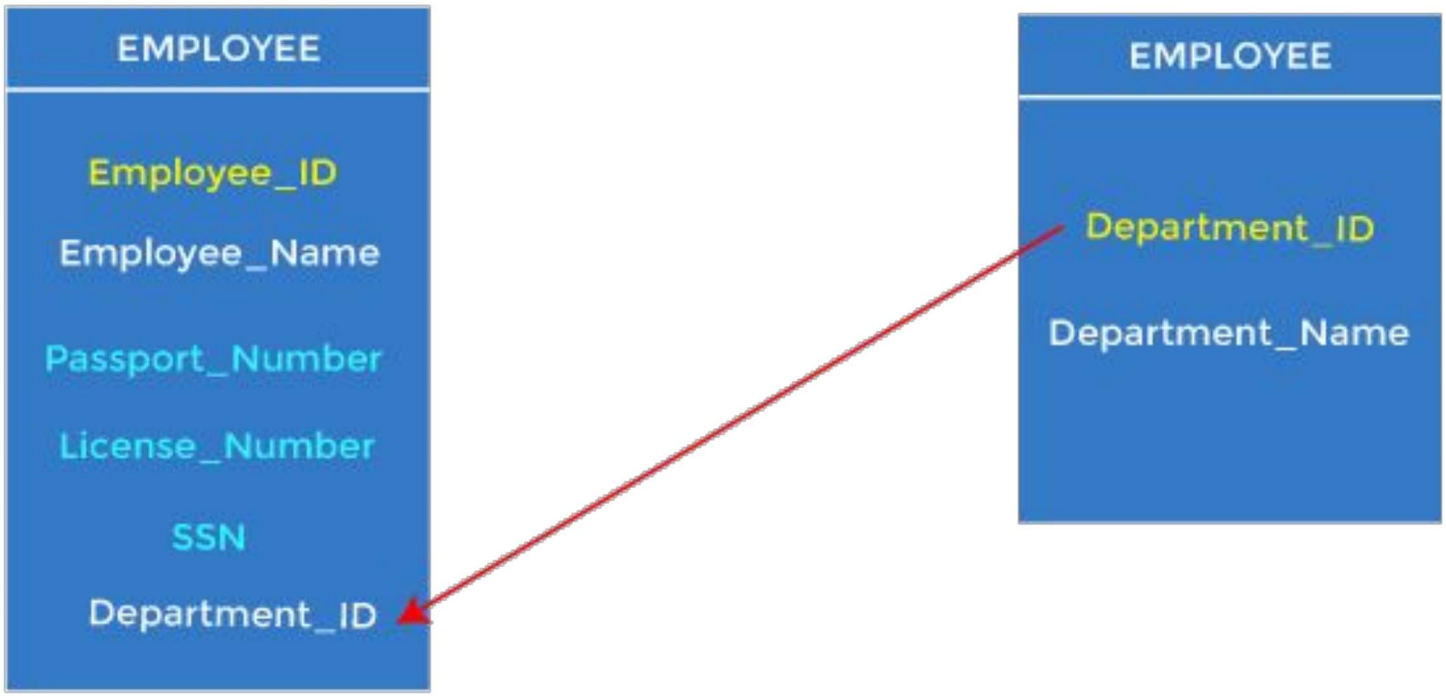
- Super key is an attribute set that can uniquely identify a tuple.
- A super key is a superset of a candidate key.
- **For example:** In the EMPLOYEE table, for(EMPLOEE_ID, EMPLOYEE_NAME), the name of two employees can be the same, but their EMPLYEE_ID can't be the same. Hence, this combination can also be a key.



- The super key would be EMPLOYEE-ID (EMPLOYEE_ID, EMPLOYEE-NAME), etc.

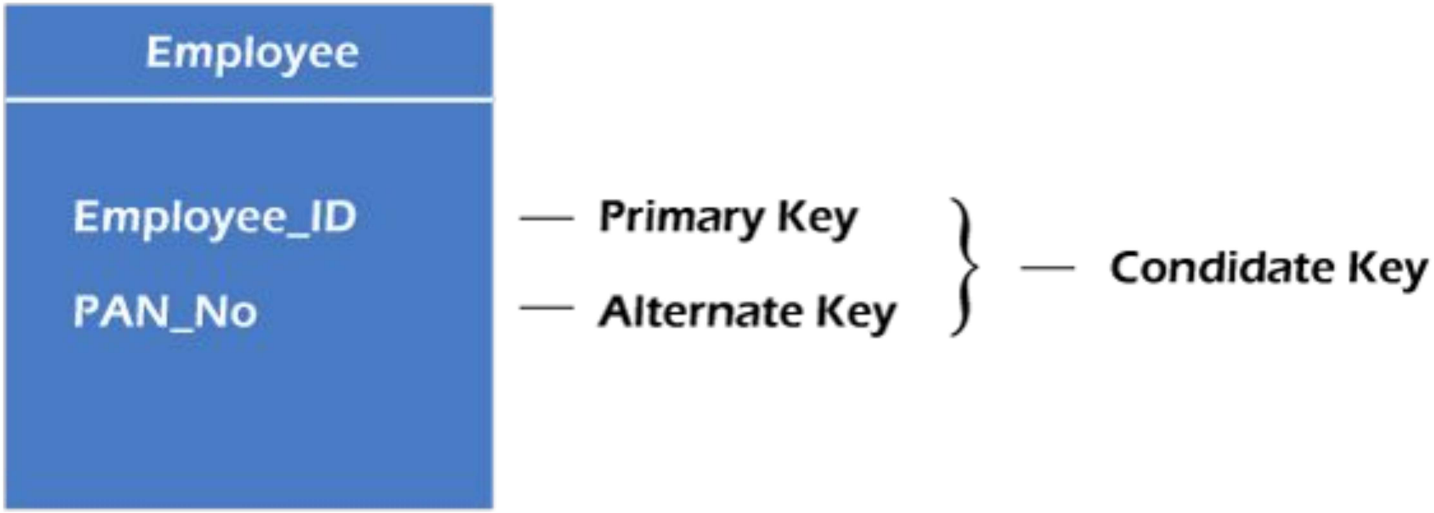
Foreign key

- Foreign keys are the column of the table used to point to the primary key of another table.
- Every employee works in a specific department in a company, and employee and department are two different entities.
- So we can't store the department's information in the employee table.
- That's why we link these two tables through the primary key of one table.
- We add the primary key of the DEPARTMENT table, Department_Id, as a new attribute in the EMPLOYEE table.
- In the EMPLOYEE table, Department_Id is the foreign key, and both the tables are related.


EMPLOYEE
Employee_ID
Employee_Name
Passport_Number
License_Number
SSN
Department_ID
EMPLOYEE
Department_ID
Department_Name

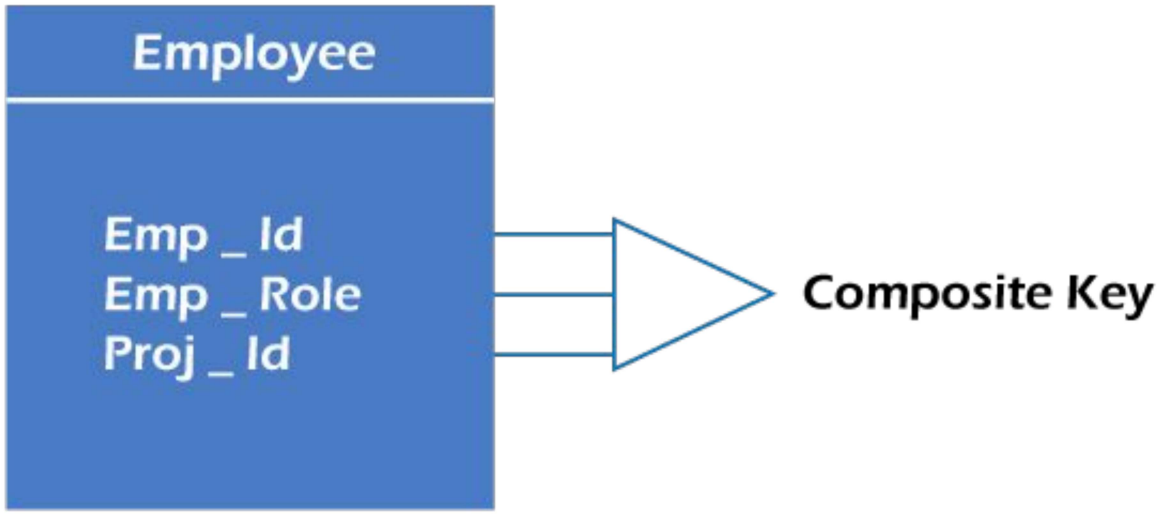
Alternate key

- There may be one or more attributes or a combination of attributes that uniquely identify each tuple in a relation.
- These attributes or combinations of the attributes are called the candidate keys.
- One key is chosen as the primary key from these candidate keys, and the remaining candidate key, if it exists, is termed the alternate key.



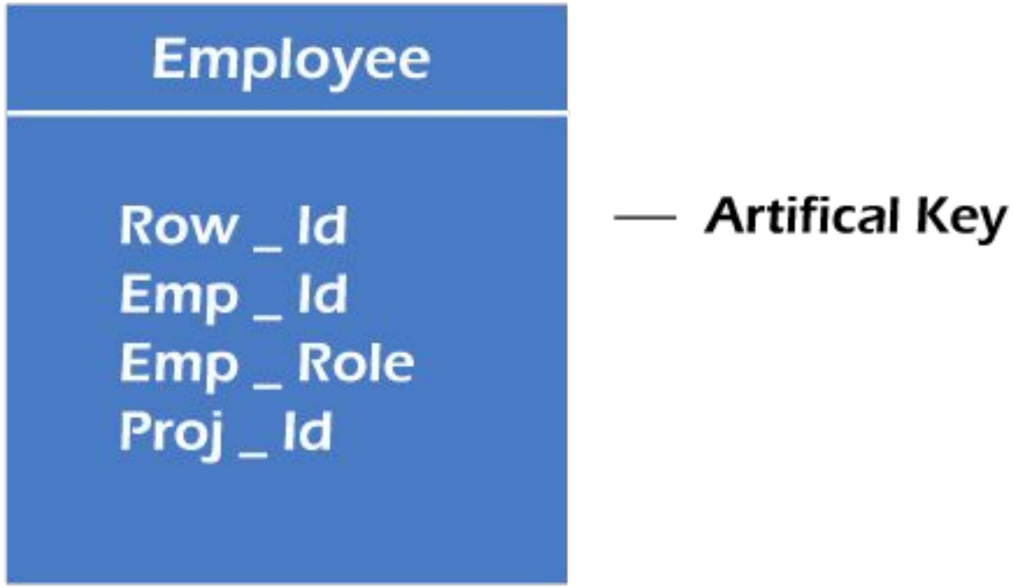
Composite key

- Whenever a primary key consists of more than one attribute, it is known as a composite key.
- This key is also known as Concatenated Key.
- **For example,** in employee relations, we assume that an employee may be assigned multiple roles, and an employee may work on multiple projects simultaneously. So the primary key will be composed of all three attributes, namely Emp_ID, Emp_role, and Proj_ID in combination. So these attributes act as a composite key since the primary key comprises more than one attribute.



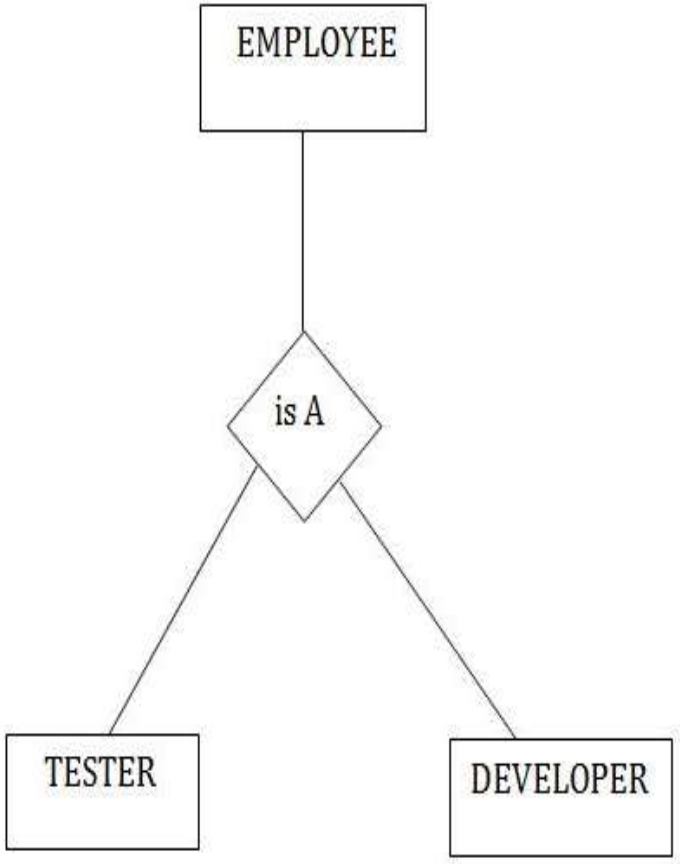
Artificial key

- The key created using arbitrarily assigned data are known as artificial keys.
- These keys are created when a primary key is large and complex and has no relationship with many other relations.
- The data values of the artificial keys are usually numbered in a serial order.
- **For example,** the primary key, which is composed of Emp_ID, Emp_role, and Proj_ID, is large in employee relations. So it would be better to add a new virtual attribute to identify each tuple in the relation uniquely.

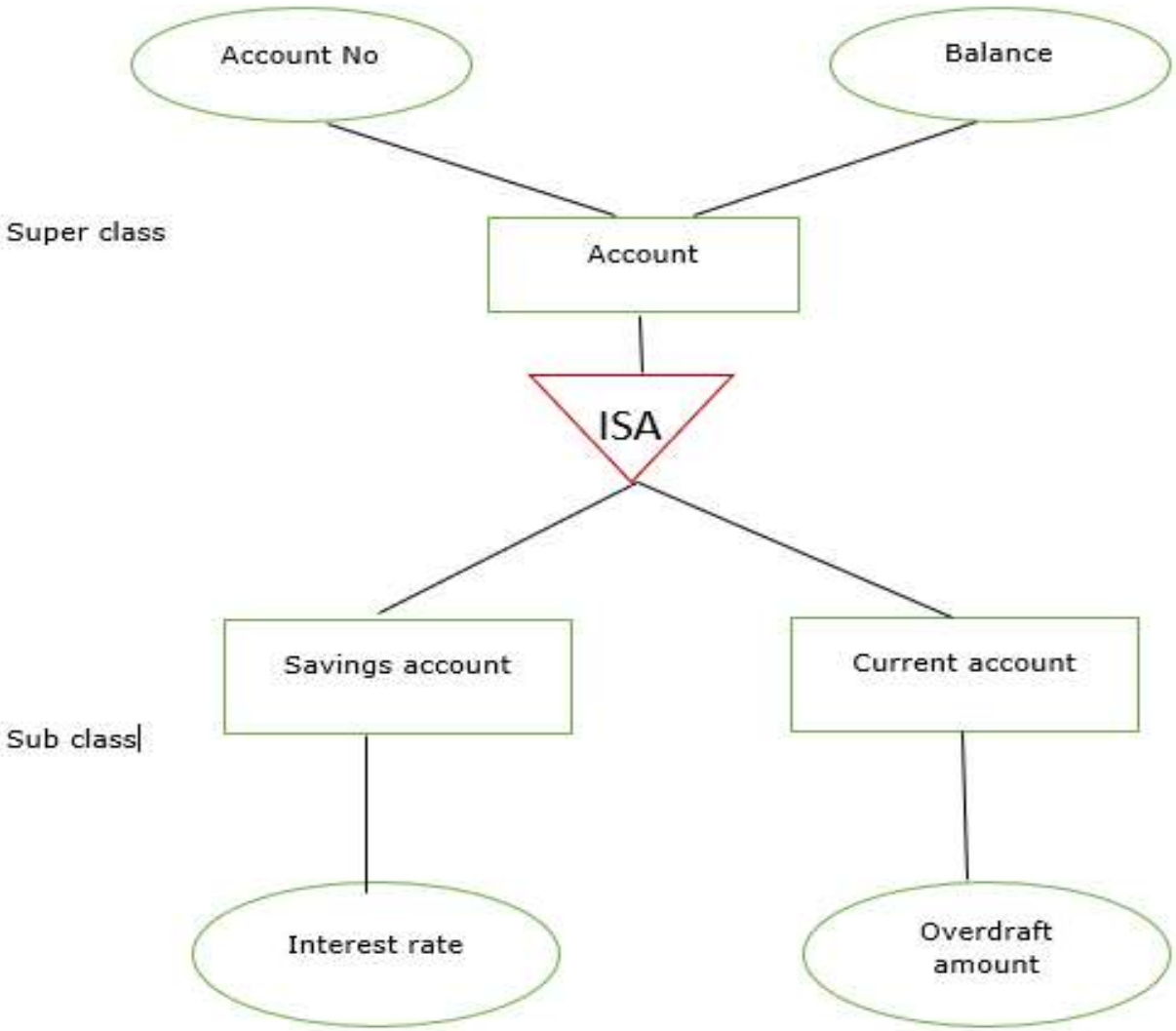


Specialization

- Specialization is a top-down approach, and it is opposite to Generalization. In specialization, one higher level entity can be broken down into two lower level entities.
- Specialization is used to identify the subset of an entity set that shares some distinguishing characteristics.
- Normally, the superclass is defined first , the subclass and its related attributes are defined next, and relationship set are then added.
- **For example**: In an Employee management system, EMPLOYEE entity can be specialized as TESTER or DEVELOPER based on what role they play in the company.

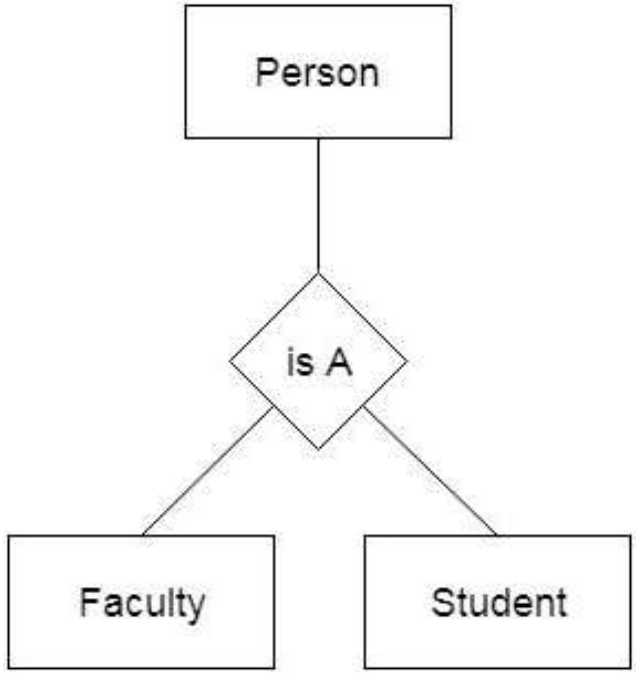


Example of Specialization

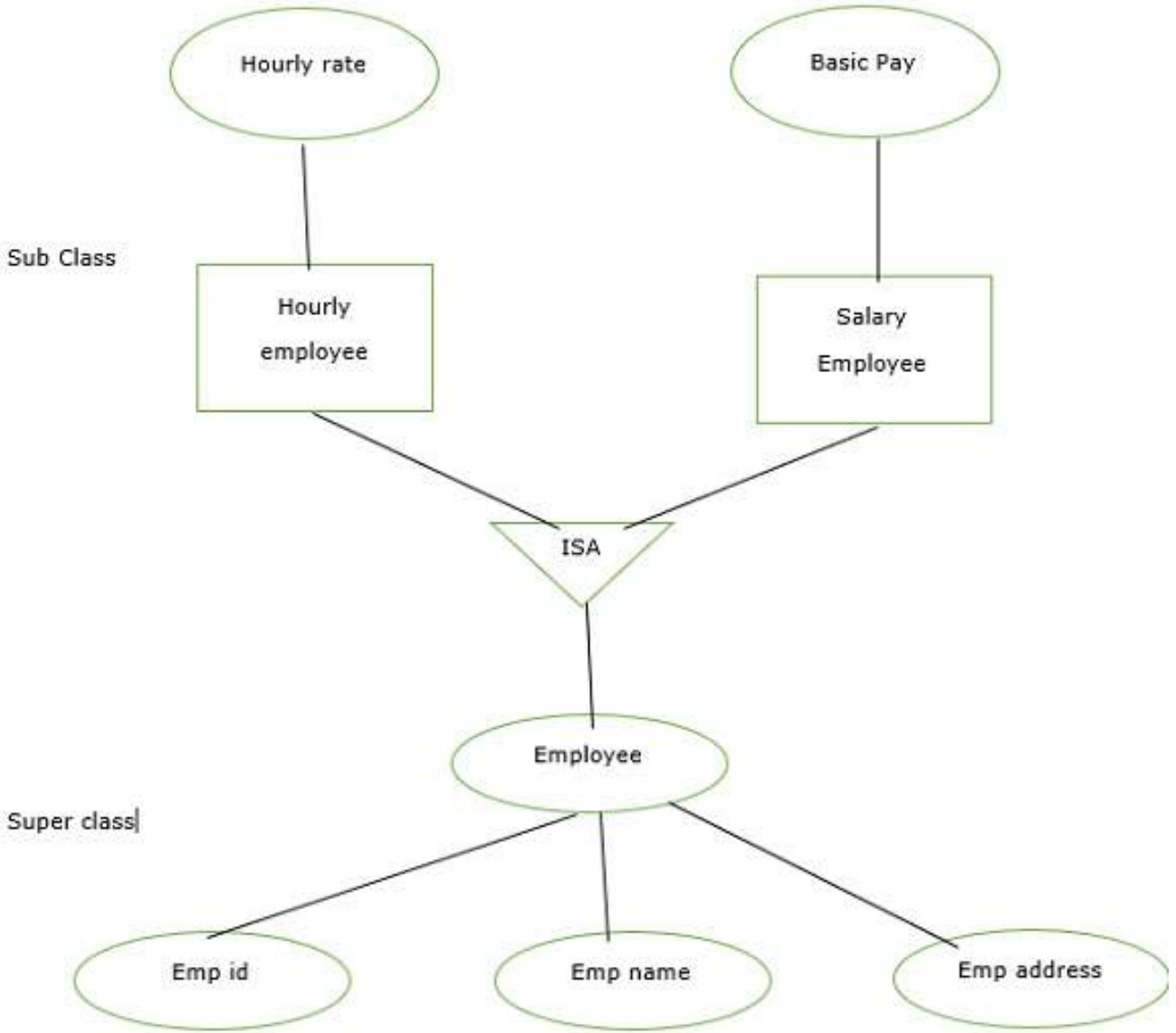


Generalization

- Generalization is like a bottom-up approach in which two or more entities of lower level combine to form a higher level entity if they have some attributes in common.
- In generalization, an entity of a higher level can also combine with the entities of the lower level to form a further higher level entity.
- Generalization is more like subclass and superclass system, but the only difference is the approach. Generalization uses the bottom-up approach.
- In generalization, entities are combined to form a more generalized entity, i.e., subclasses are combined to make a superclass.
- **For example,** Faculty and Student entities can be generalized and create a higher level entity Person.

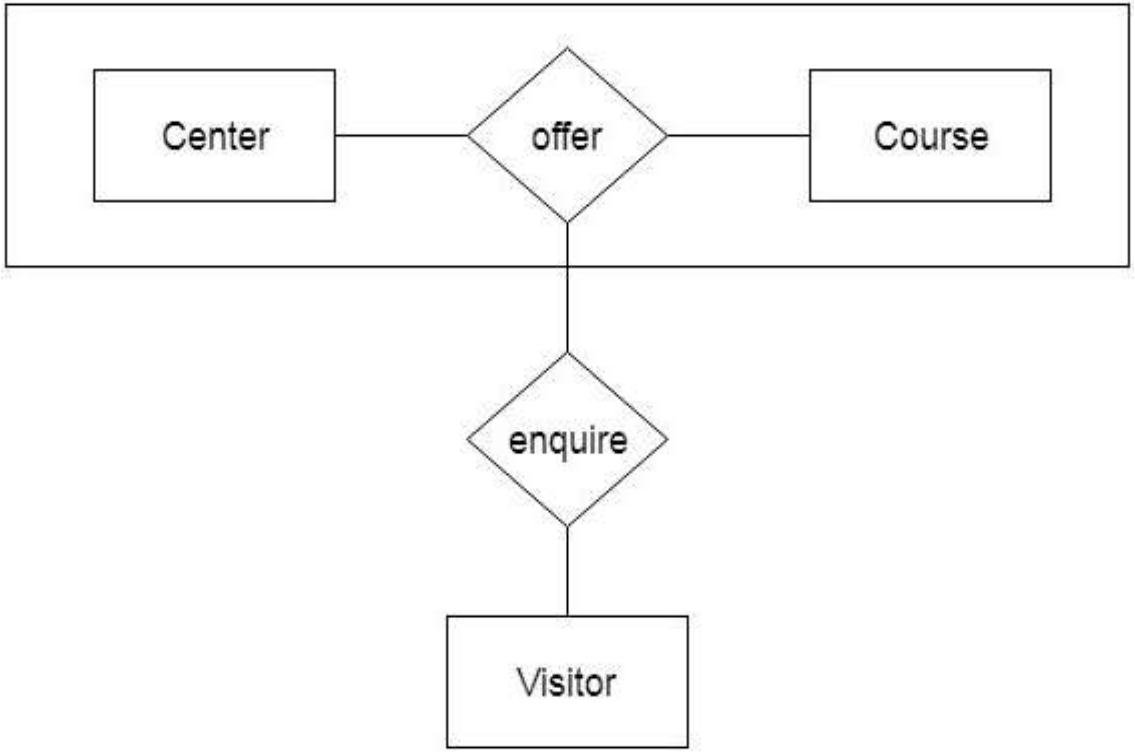


Example of Generalization

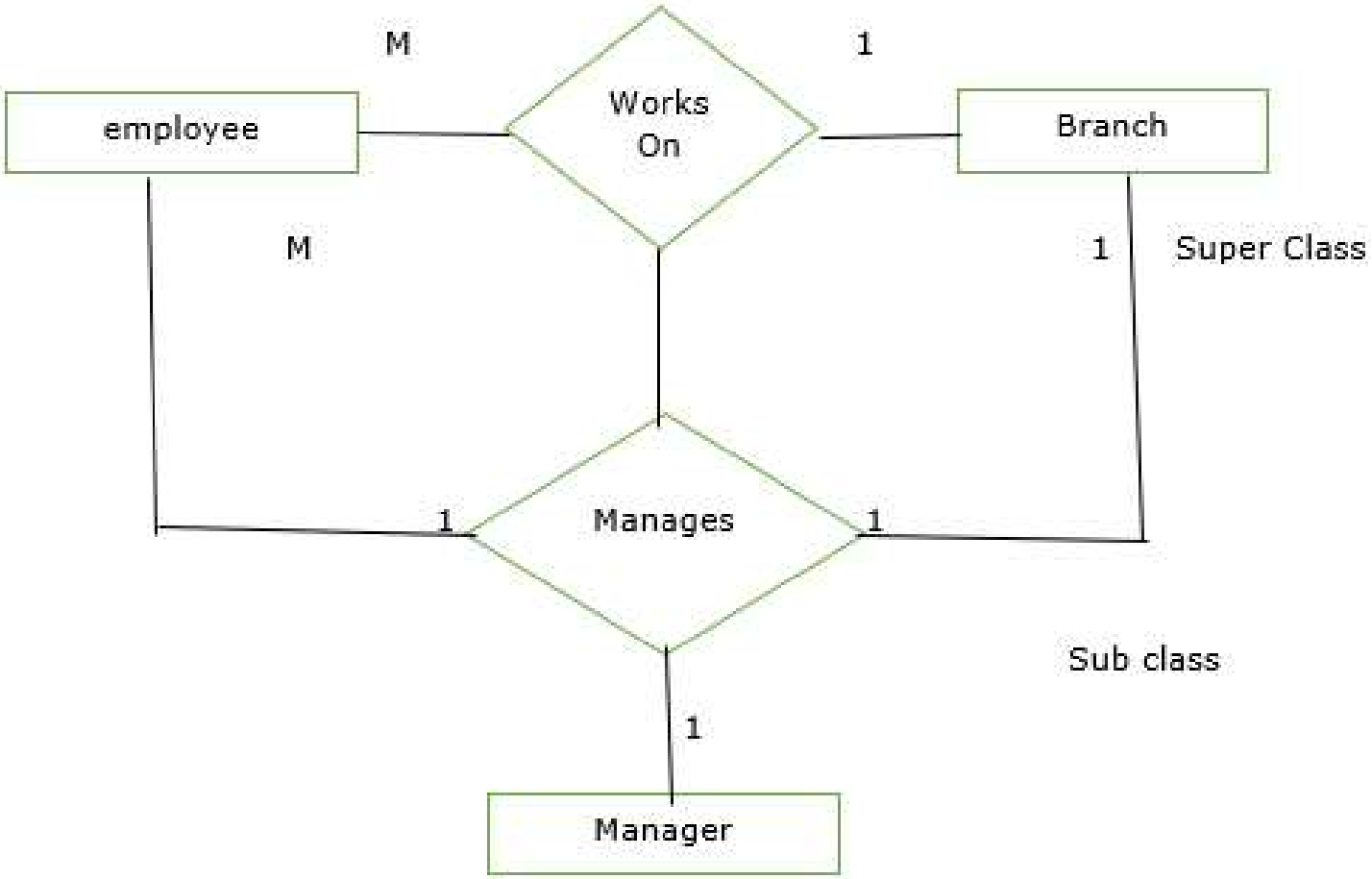


Aggregation

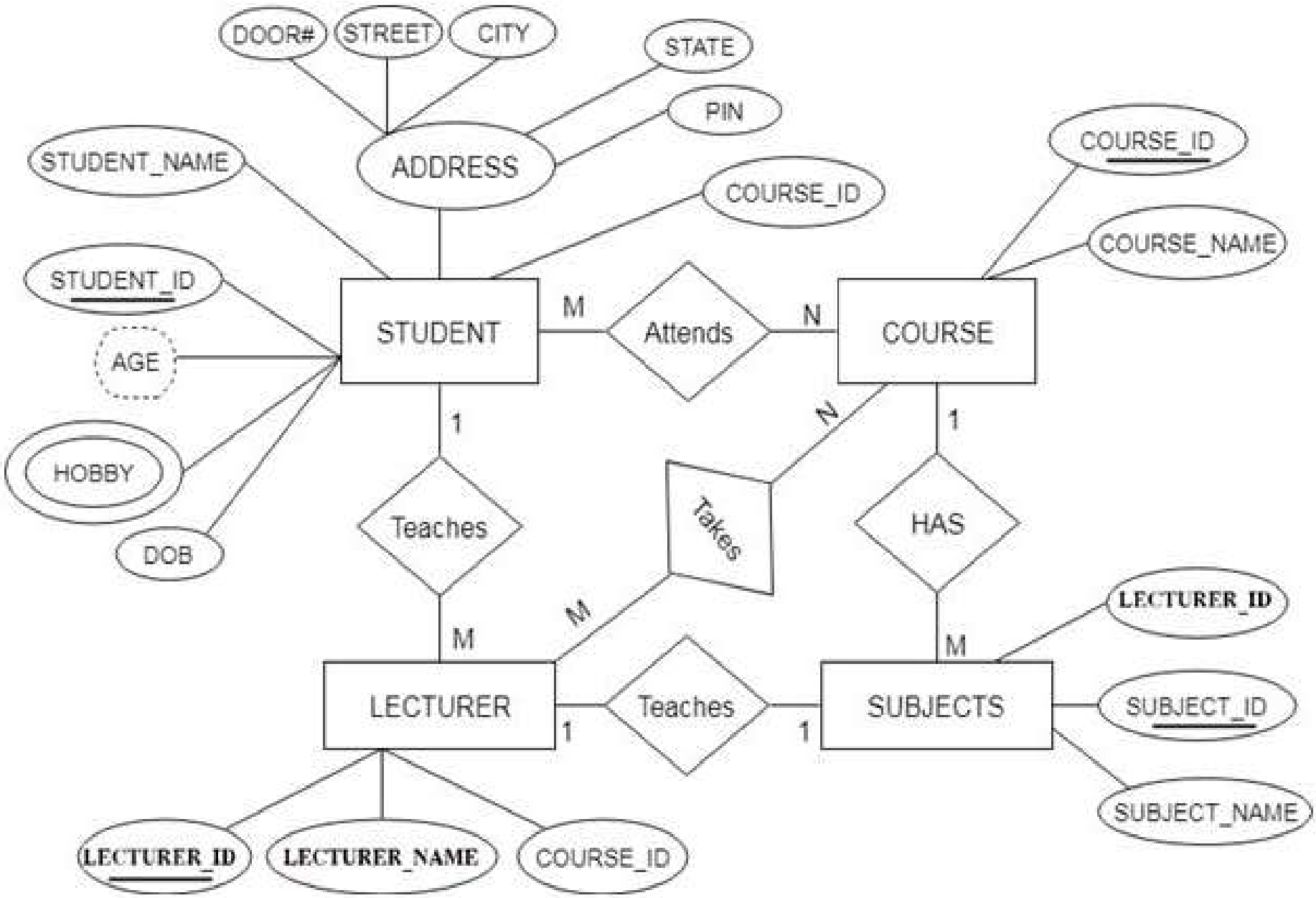
- In aggregation, the relation between two entities is treated as a single entity. In aggregation, relationship with its corresponding entities is aggregated into a higher level entity.
- **For example:** Center entity offers the Course entity act as a single entity in the relationship which is in a relationship with another entity visitor. In the real world, if a visitor visits a coaching center then he will never enquiry about the Course only or just about the Center instead he will ask the enquiry about both.



Example of Aggregation



Reduction of ER diagram to Table



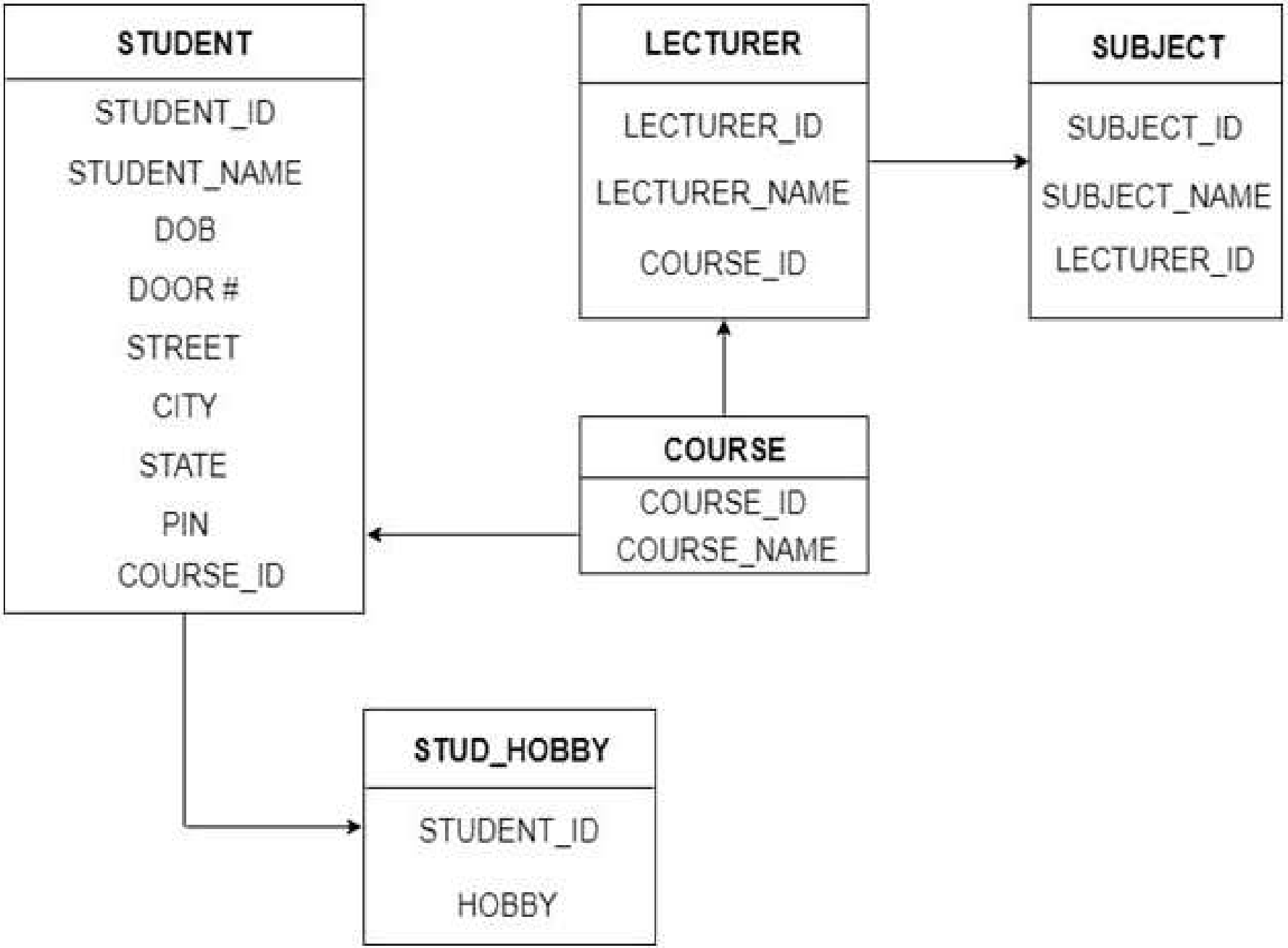
- **Entity type becomes a table.**
  - In the given ER diagram, LECTURE, STUDENT, SUBJECT and COURSE forms individual tables.
- **All single-valued attribute becomes a column for the table.**
  - In the STUDENT entity, STUDENT_NAME and STUDENT_ID form the column of STUDENT table. Similarly, COURSE_NAME and COURSE_ID form the column of COURSE table and so on.
- **A key attribute of the entity type represented by the primary key.**
  - In the given ER diagram, COURSE_ID, STUDENT_ID, SUBJECT_ID, and LECTURE_ID are the key attribute of the entity.

- **Conversion of weak entity**
  - For each weak entity create a separate table with the same name.
  - Include all attributes.
  - Include the P key of a strong entity as foreign key is the weak entity.
  - Declare the combination of foreign key and decimator attribute as P key from the weak entity.
- **Conversion of one-to-one relationship**
  - For each one to one relation, say A and B modify either A side or B side to include the P key of the other side as a foreign key.
  - If A or B is having total participation, then that should be a modified table.
  - If a relationship consists of attributes, include them also in the modified table.

- **Conversion of one-to-many relationship**
  - For each one to many relationships, modify the M side to include the P key of one side as a foreign key.
  - If relationships consist of attributes, include them as well.
- **Conversion of many-many relationship**
  - For each many-many relationship, create a separate table including the P key of M side and N side as foreign keys in the new table.
  - Declare the combination of foreign keys as P for the new table.
  - If relationships consist of attributes, include them also in the new table.

- **The multivalued attribute is represented by a separate table.**
- In the student table, a hobby is a multivalued attribute. So it is not possible to represent multiple values in a single column of STUDENT table. Hence we create a table STUD_HOBBY with column name STUDENT_ID and HOBBY. Using both the column, we create a composite key.
- **Composite attribute represented by components.**
- In the given ER diagram, student address is a composite attribute. It contains CITY, PIN, DOOR#, STREET, and STATE. In the STUDENT table, these attributes can merge as an individual column.

- **Derived attributes are not considered in the table.**
- In the STUDENT table, Age is the derived attribute. It can be calculated at any point of time by calculating the difference between current date and Date of Birth.
- Using these rules, you can convert the ER diagram to tables and columns and assign the mapping between the tables.


STUDENT
STUDENT_ID
STUDENT_NAME
DOB
DOOR #
STREET
CITY
STATE
PIN
COURSE_ID
LECTURER
LECTURER_ID
LECTURER_NAME
COURSE_ID
SUBJECT
SUBJECT_ID
SUBJECT_NAME
LECTURER_ID
COURSE
COURSE_ID
COURSE_NAME
STUD_HOBBY
STUDENT_ID
HOBBY