Database Management System

Subject Code : CSC403

Credits : 3

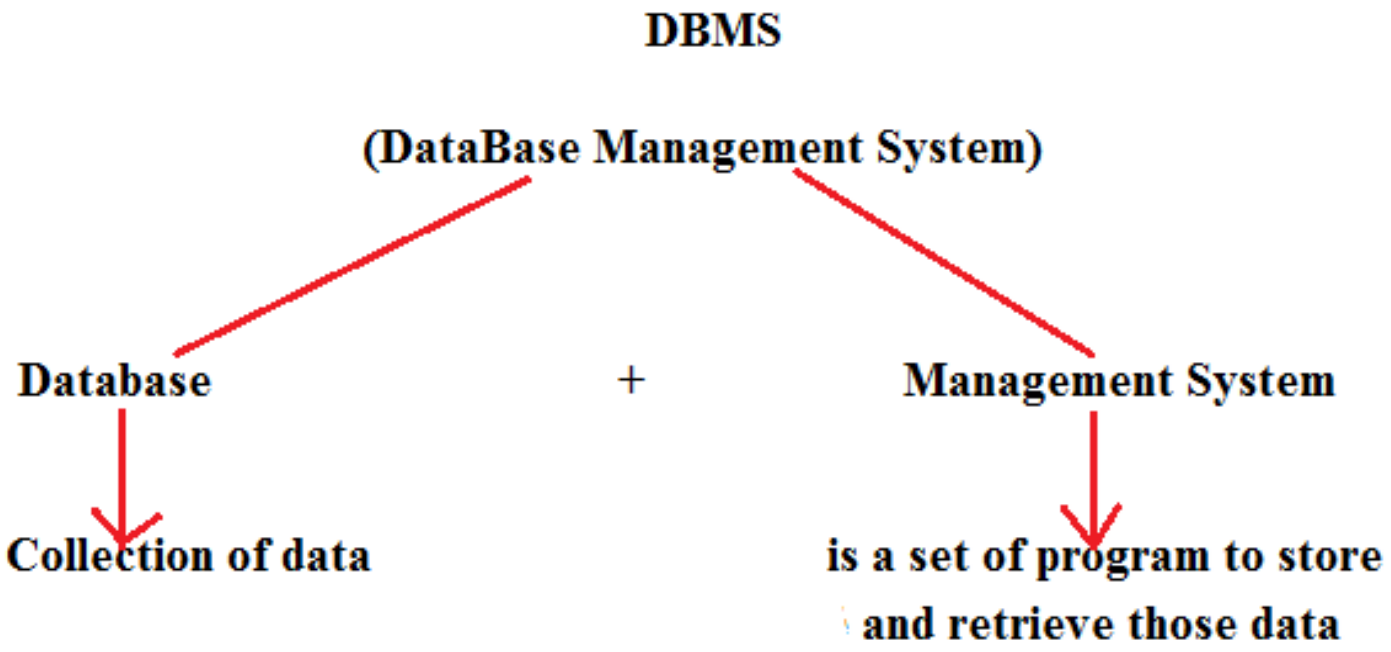
--Shilpa Ingoley

An introduction of Database

- Data: Data in raw or unorganized form of information and knowledge.

- Database can be defined as a collection of coherent, meaningful data.

- It is consist of numbers, image, text, document & voice, video.



- DBMS is a S/W which is used to manage database
- Eg : MySQL,Oracle,MS-Access,DB2 ,Ingres,Sybase etc

Relational Model

- **RDBMS stands for relational database management system.**

A relational database has following major components: Table, Record / Tuple, Field

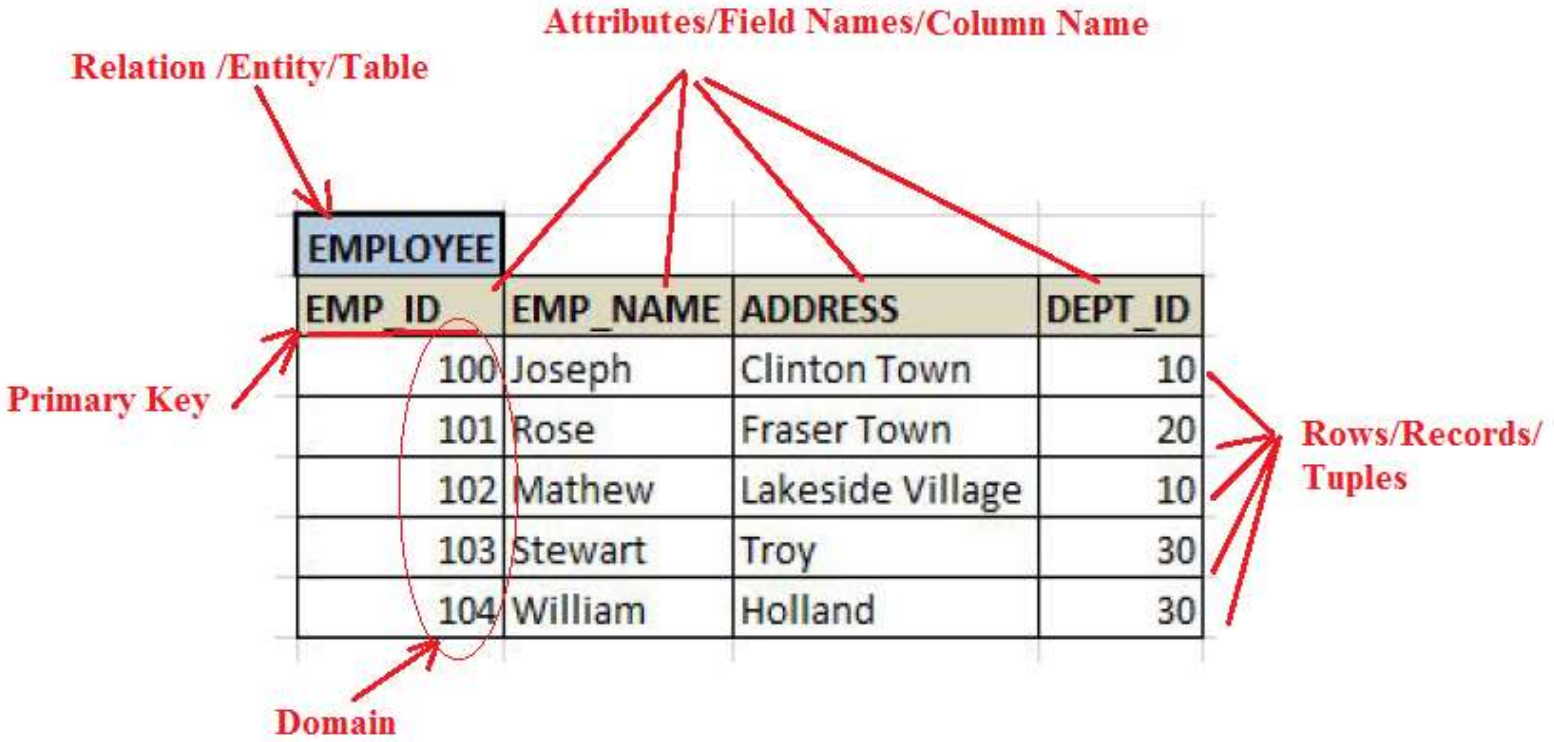
- **Table:**
- A table is a collection of data represented in rows and columns.
- For e.g. following table stores the information of Instructor/Teacher

- All the data is stored in various tables.
- Example of tabular data in the relational model

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

Employee Table(relation)



EMPLOYEE			
EMP_ID	EMP_NAME	ADDRESS	DEPT_ID
100	Joseph	Clinton Town	10
101	Rose	Fraser Town	20
102	Mathew	Lakeside Village	10
103	Stewart	Troy	30
104	William	Holland	30

Table:

A table is a collection of data represented in rows and columns

EMPLOYEE

EMP_ID	EMP_NAME	ADDRESS	DEPT_ID
100	Joseph	Clinton Town	10
101	Rose	Fraser Town	20
102	Mathew	Lakeside Village	10
103	Stewart	Troy	30
104	William	Holland	30

DEPARTMENT

DEPT_ID	DEPT_NAME
10	Accounting
20	Quality
30	Design

Records / Tuple :

Each row of a table is known as record or it is also known as tuple. For e.g. The below row is a record.

Field:

The above table has four fields: Emp_Id, Emp_Name, Address & Dept_Id.

Entity-Relationship (ER) Model

- It is a high-level data model. This model is used to define the data elements and relationship for a specified system.

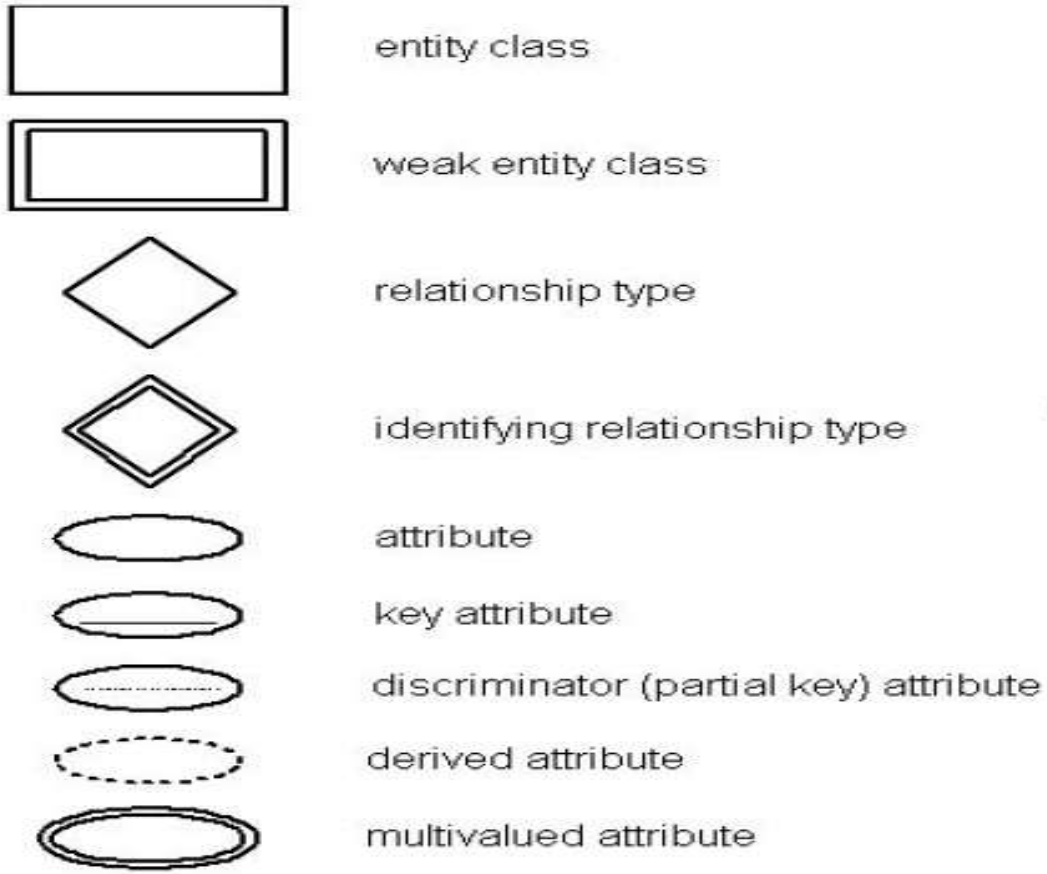
- It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.

- In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

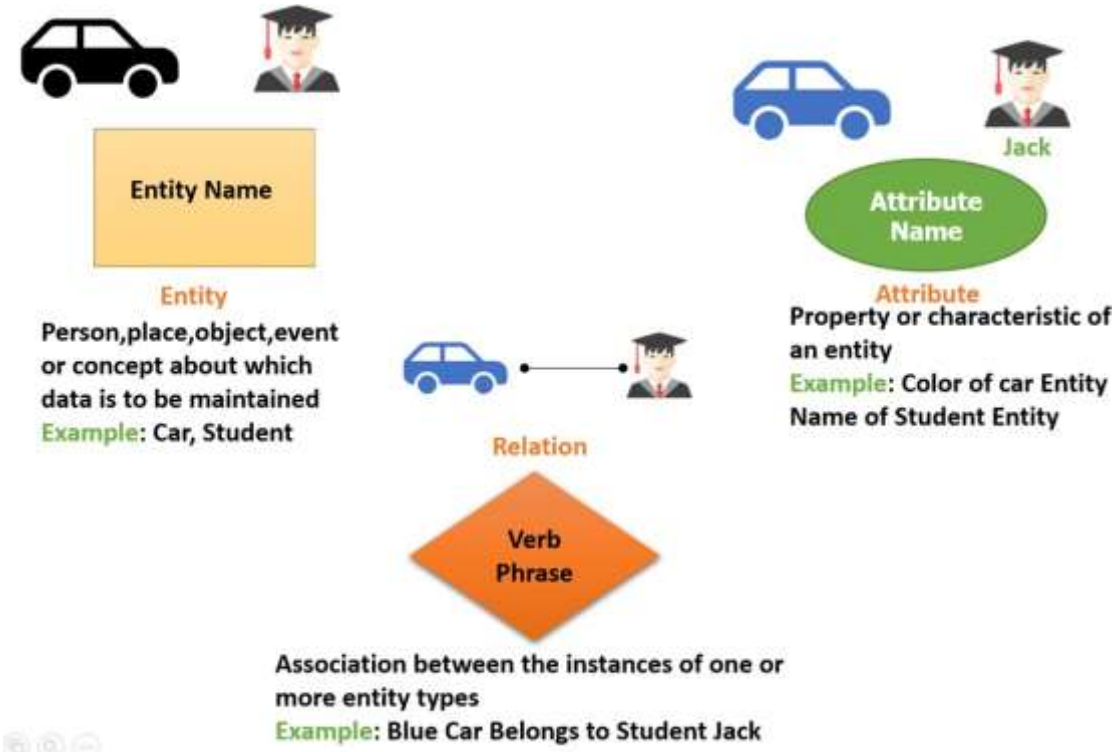
This model is based on three basic concepts:

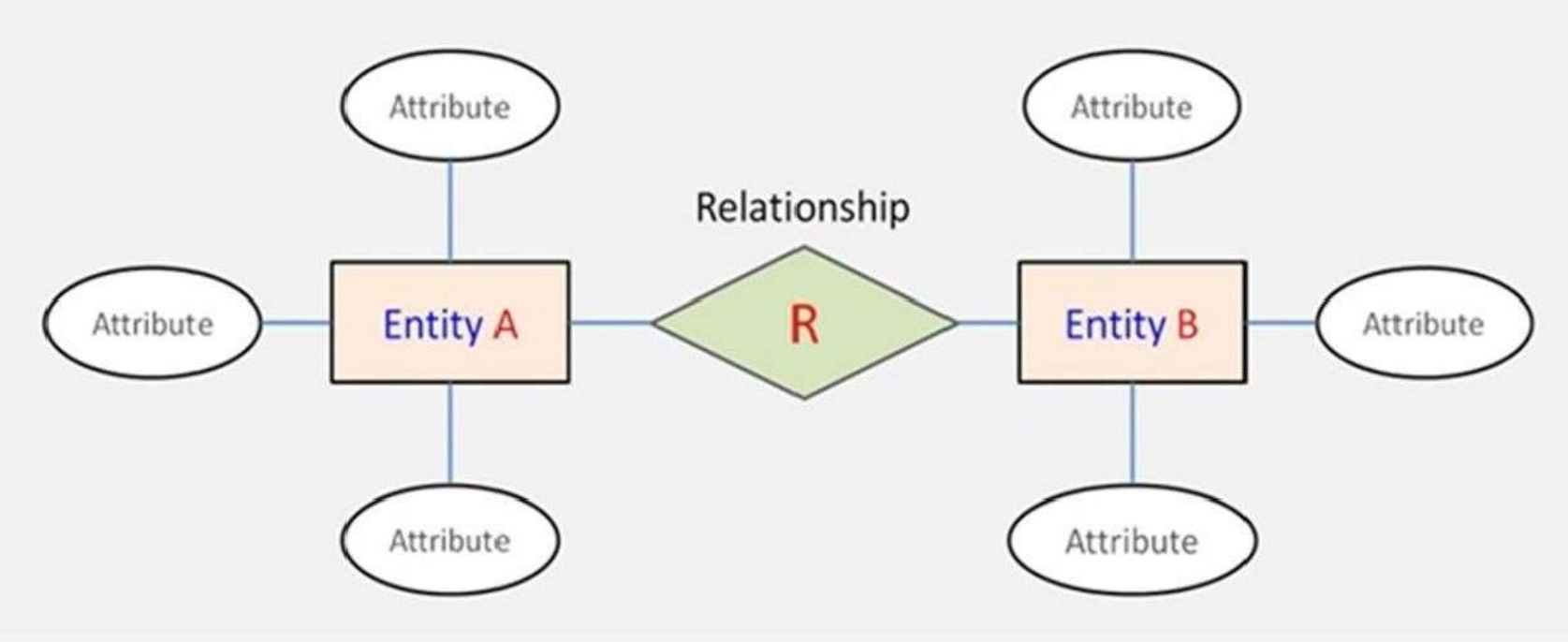
1. Entities
2. Attributes
3. Relationships

Components Of E-R model



For example: in a University database, we might have entities for Students, Courses, and Lecturers. Student entity can have attributes like Rollno, Name, and DeptID. They might have relationships with Courses and Lecturers.





Entity Relationship Diagram (ERD) In DBMS

Entity Relationship Diagram

- It is very much essential to understand the requirement properly and design them efficiently.
- It's a high level Conceptual Model developed by Chen in 1976 to facilitate database design
- Independent of hardware and software constraints
- It is like a foundation of the building(Architectural building plan)
- For this purpose, we use ER diagrams where we plan the database pictorially.
- It represent graphical representation of logical relationship of entities
- ER Data Model is based on the real world objects and their relationship
- A *database* can be modeled as:

  a collection of entities,

  relationship among entities.

Entities

- It may be an object with a physical existence – a particular person, car, house, or employee – or it may be an object with a conceptual existence – a company, a job, or a university course.
- Example:

Consider an organization - manager, product, employee, department etc. can be taken as an entity.

- An **entity set** is a set of entities of the same type (e.g., all persons having an account at a bank).

Basic Building Blocks

- ERD comprises of three main components
- 1) Entities/Entity is any thing about which data can be collected and stored
- 2) Attribute is a characteristics of an entity
- 3) Relationship describe the associations among(two or more entities)
  - One-to-one(1:1)
  - One –to-many (1:M)
  - Many-to-many(M:N)

Entity

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

  Example: an individual student, customer, account

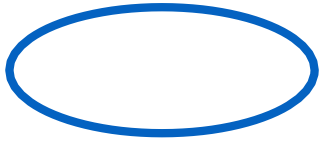
- An *entity set* is a set of entities of the same type that share the same properties.

  Example: set of all students, customers, accounts

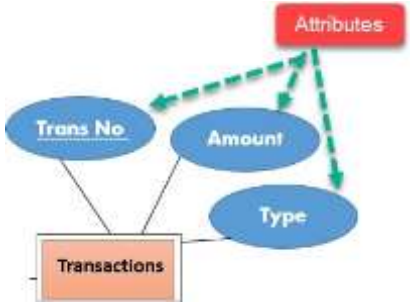
- Entities are represented by means of rectangles. Rectangles are named with the entity set they represent.

Attributes

- Each entity has a set of associated properties that describes the entity. These properties are known as ***attributes***.
- They are represented by an oval.

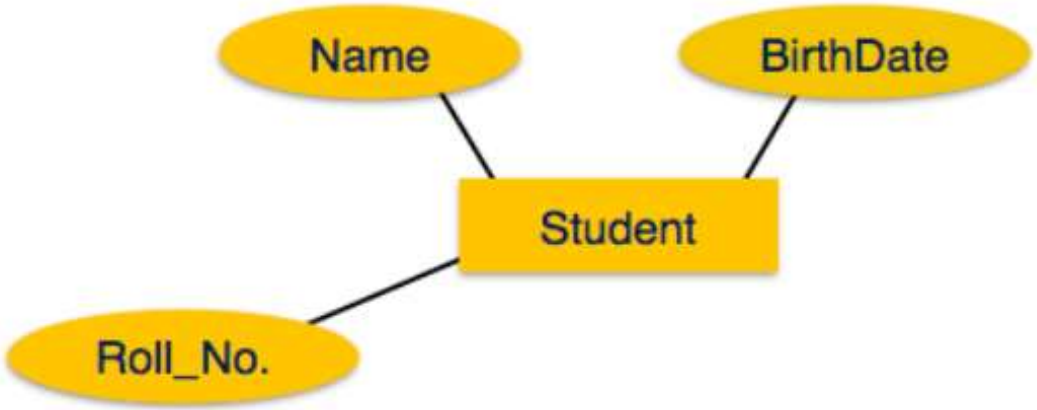


- **Attributes can be:**
  1. Key attribute
  2. Simple attribute
  3. Composite attribute
  4. Single valued attribute
  5. Multivalued attribute
  6. Stored attribute
  7. Derived attribute
  8. Complex attribute
  9. Null attribute
  10. Descriptive attribute



Attributes

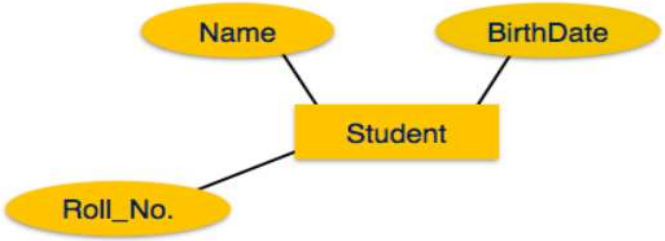
- **Entities have *attributes***
- Attributes are the properties of entities. Attributes are represented by means of ellipses. Every ellipse represents one attribute and is directly connected to its entity set (rectangle).
- E.g. Students have *names, Roll number, DOB etc.*
- It's also known as columns of the table.



Types of attributes

Depending on the values that an attribute can take, it is divided into different types.

- **Simple Attribute :** These kinds of attributes have values which cannot be divided further.
- Example: STUDENT_ID attribute which cannot be further divided . Passport Number is unique value and it cannot be divided.

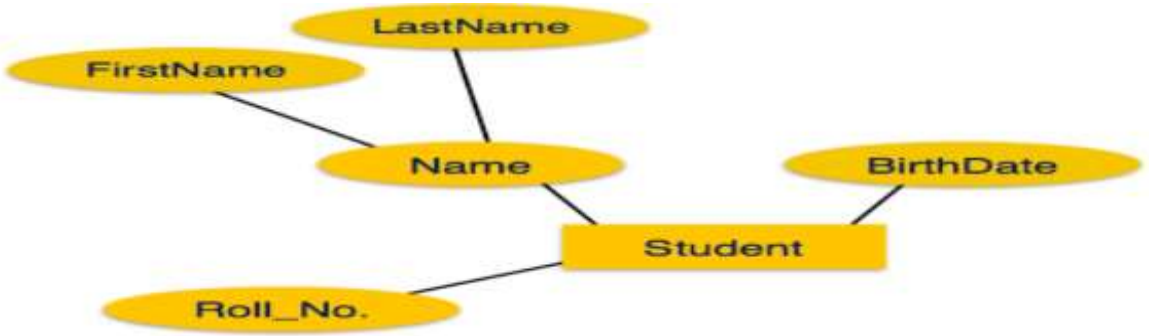


Types of attributes

- **Composite Attribute :** This kind of attribute can be divided further to more than one simple attribute.

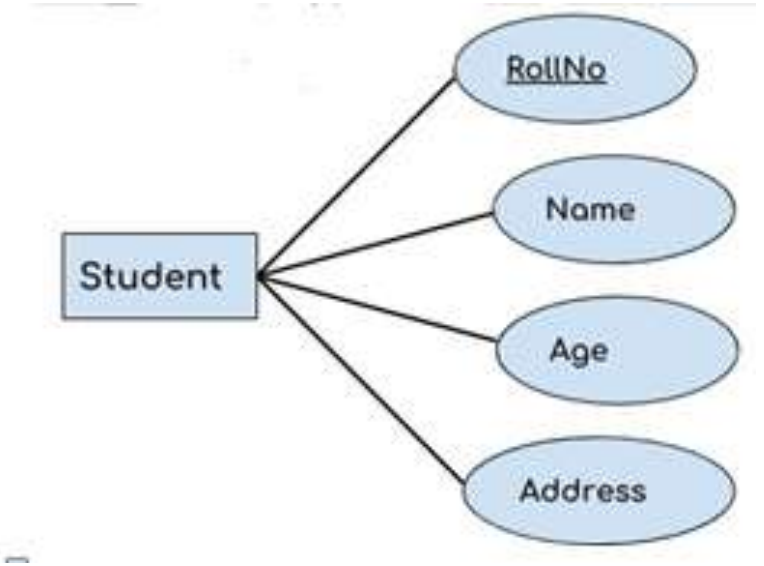
Eg 1: address of a person. Here address can be further divided as Door#, street, city, state and pin which are simple attributes.

Eg 2 : Name



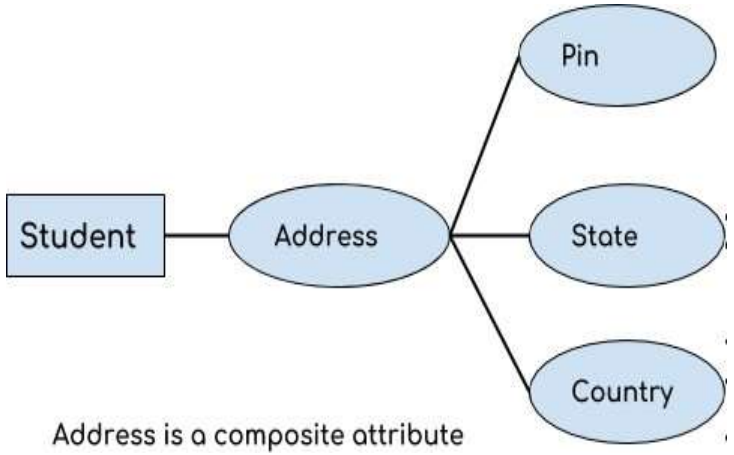
Key attribute

- It can uniquely identify an entity from an entity set.
- **Text of key attribute is underlined**.
- For example, student roll number can uniquely identify a student from a set of students

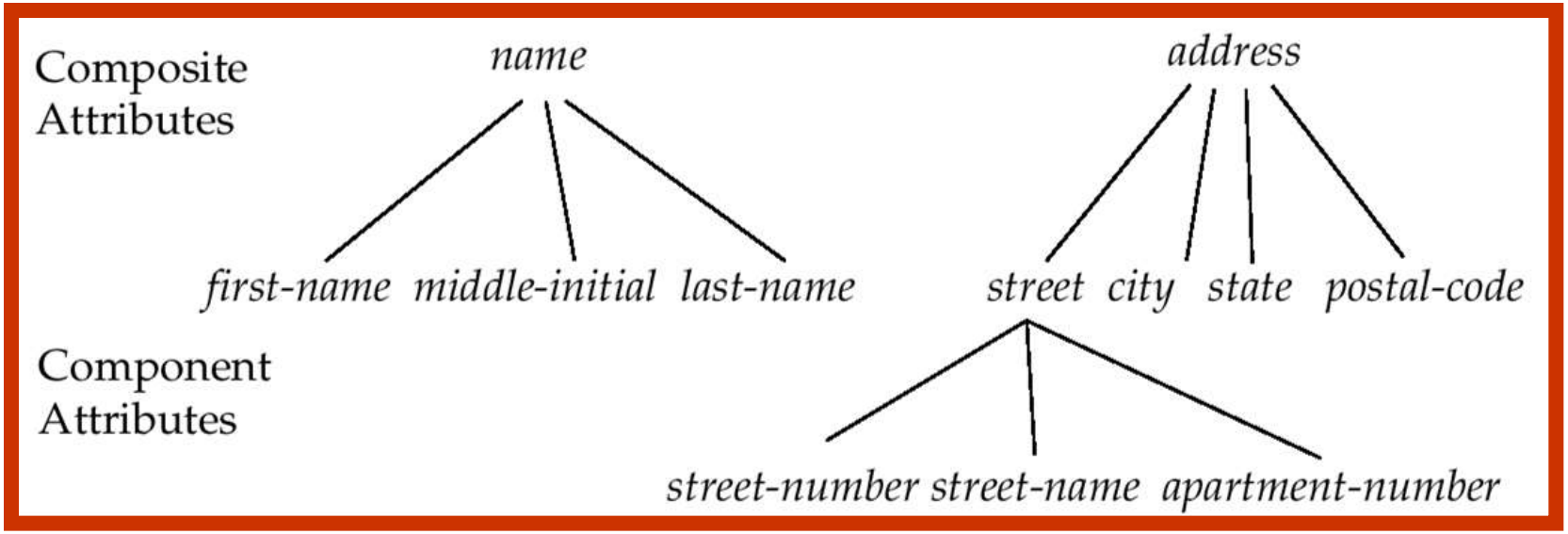


Composite attribute

- It is a combination of other attributes
- For example, In student entity, the student address is a composite attribute as an address is composed of other attributes such as pin code, state, country.



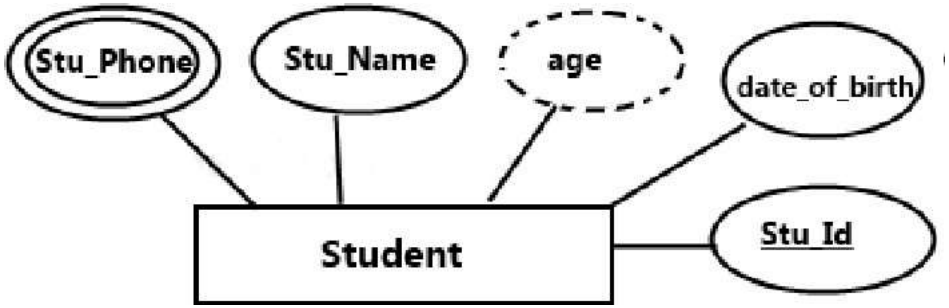
Composite Attributes



Multivalued attribute

- It can hold multiple values is known as multivalued attribute.
- It is represented with **double ovals** in an ER Diagram.
- For example – A person can have more than one phone numbers so the phone number attribute is multivalued.

- **Derived attribute:**
- It is one whose value is dynamic and derived from another attribute.
- It is represented by **dashed oval** in an ER Diagram.
- For example – Person age is a derived attribute as it changes over time and can be derived from another attribute (Date of birth).



● **Derived Attribute :** Derived attributes are the one whose value can be obtained from other attributes of entities in the database.

Eg1: Age of a person can be obtained from date of birth and current date.

Eg2: Average salary, annual salary,

Eg3: Total marks of a student etc are few examples of derived attribute.

Derived attributes are depicted by dashed ellipse.

- **Stored Attribute**

The attribute which gives the value to get the derived attribute are called Stored Attribute.

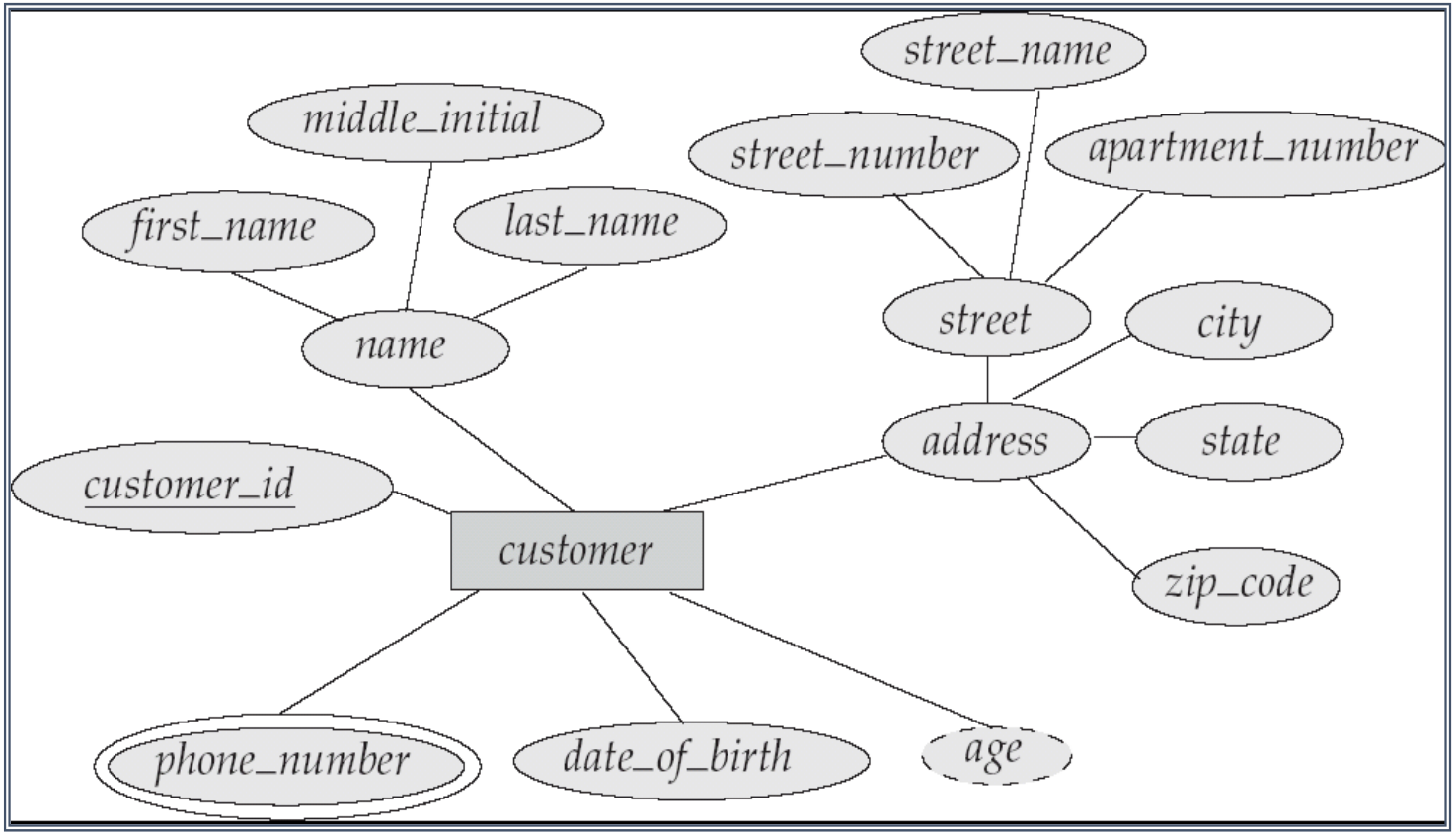
Eg: age is derived using Date of Birth. Hence Date of Birth is a stored attribute.

- **Key attribute**: An attribute whose values are distinct for each individual entity in the entity set. Such an attribute is called Key attribute.

  Eg: Employee ID,Student Id, Adhar card
- **Complex attribute**: composite and multivalued attributes can be nested arbitrarily

  Eg: Person having multiple addresses
- **Descriptive Attribute**: attributes of relationship set is called descriptive attributes

Customer entity with Composite, Multivalued, and Derived Attributes

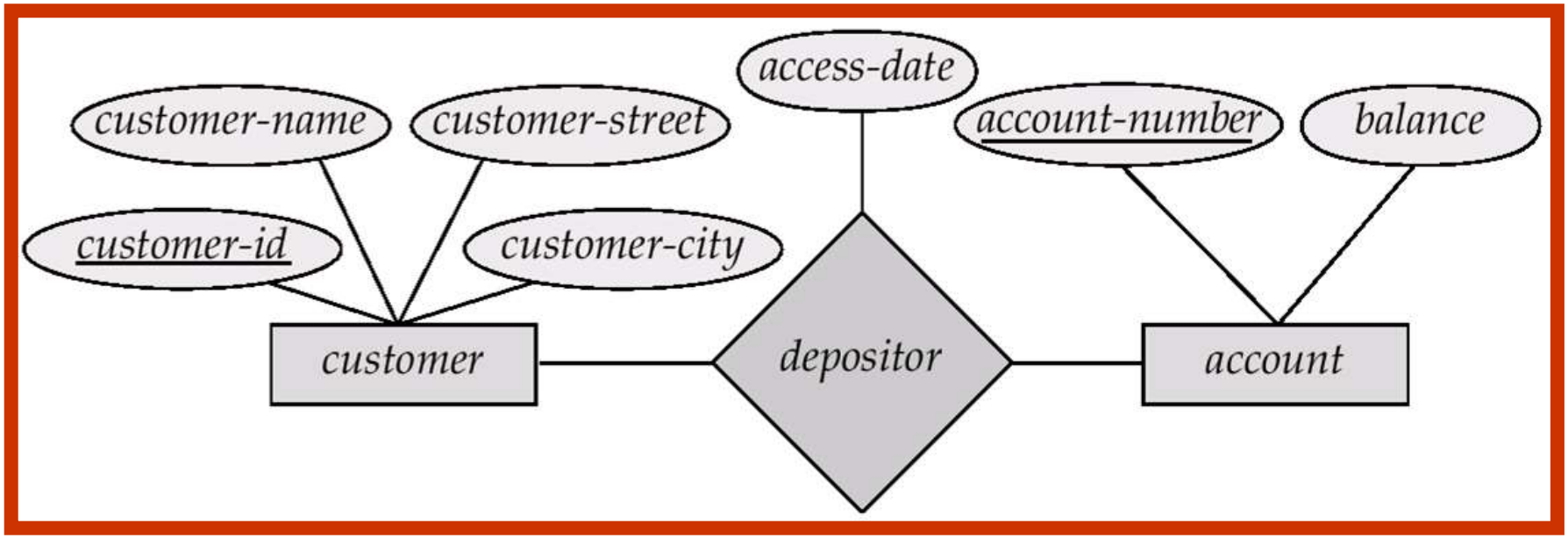


Q) Identify entities and their respective attributes for Library

Relationship

- Relationships are represented by diamond-shaped box. Name of the relationship is written inside the diamond-box. All the entities (rectangles) participating in a relationship, are connected to it by a line.

Relationship Sets with Attributes



Questions on E-R Diagrams

1)Rectangles represent _____.

2) Diamonds represent _____.

3)_____ attributes to entity sets and entity sets to relationship sets.

4)Ellipses represent _____

5) Double ellipses represent _____

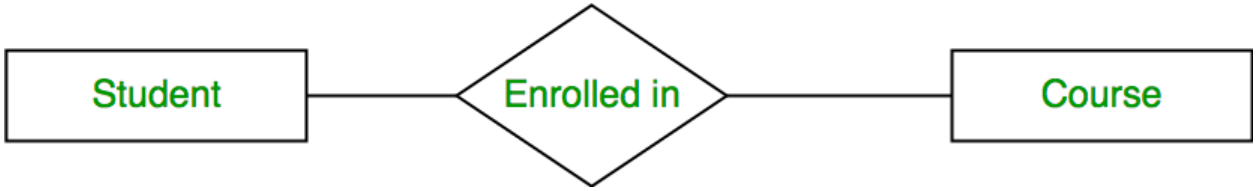
6) Dashed ellipses denote _____.

7) _____ indicates primary key attributes

Relationship Type & Relationship set

Relationship type:

- It represents the association between entity types.
- For example, 'Enrolled in' is a relationship type that exists between
  entity type Student and Course.
- In ER diagram, relationship type is represented by a **diamond** and connecting the entities with lines.



Keys - Key Concepts:

- Key is an attribute or collection of attributes that uniquely identifies an entity among entity set.
- There are different types of keys:

1. **Super key**
2. **Candidate key**
3. **Primary key**
4. **Foreign key**
5. **Unique key**
6. **Composite key**
7. **Alternate key**

Let's take an example to understand this:

Table: Employee

Emp_SSN	Emp_Number	Emp_Name
123456789	226	Steve
999999321	227	Ajeet
888997212	228	Chaitanya
777778888	229	Robert

Keys.

Super key:

A super key is a combination of columns that uniquely identifies any row within a relational database management system (RDBMS) table.

- Set of one or more attributes (columns), which can **uniquely identify a row** in a table.
- All the following sets of super key can uniquely identify a row of the employee table.
- {Emp_SSN}
- {Emp_Number}
- {Emp_SSN, Emp_Number}
- {Emp_SSN, Emp_Name}
- {Emp_SSN, Emp_Number, Emp_Name}
- {Emp_Number, Emp_Name}

Contd…

Candidate key:

A candidate key is a closely related concept where the super key is reduced to the minimum number of columns required to uniquely identify each row.

- It is a minimal super key with no redundant attributes.
- The following two set of super keys are chosen as candidate key.
- {Emp_SSN}
- {Emp_Number}

- **The value of a key attribute(s) can be used to uniquely identify each tuple in the relation**
  - **Simple:** Single attribute
  - **Composite:** Combination of attributes

Primary key(PK):

A primary key is a special relational database table column (or combination of columns) designated to uniquely identify all table records.

- A primary key's main features are:
  - It must contain a unique value for each row of data.
  - It enforces that there should be no duplicate value
  - It cannot contain null values.
  - One table will contain one PK
  - It is Underline
- It is selected from a set of candidate keys.
- This is done by database admin or database designer.
- We can say that either {Emp_SSN} or {Emp_Number} can be chosen as a primary key for the table Employee.

Contd…

- **Foreign key (Concept of Referential Integrity)**

- A foreign key is a column or group of columns in a relational database table that provides a link between data in two tables. It acts as a cross-reference between tables because it references the primary key of another table, thereby establishing a link between them.
- The majority of tables in a relational database system adhere to the foreign key concept.
- In complex databases and data warehouses, data in a domain must be added across multiple tables, thus maintaining a relationship between them. The concept of referential integrity constraints is derived from foreign key theory.

Foreign key:

- Foreign keys are the columns of a table that points to the candidate key of another table.

Example:

Stu_Id column in Course_enrollment table is a foreign key as it points to the primary key of the Student table.

EMPLOYEE

EMP_ID	EMP_NAME	ADDRESS	DEPT_ID
100	Joseph	Clinton Town	10
101	Rose	Fraser Town	20
102	Mathew	Lakeside Village	10
103	Stewart	Troy	30
104	William	Holland	30

DEPARTMENT

DEPT_ID	DEPT_NAME
10	Accounting
20	Quality
30	Design

Contd…

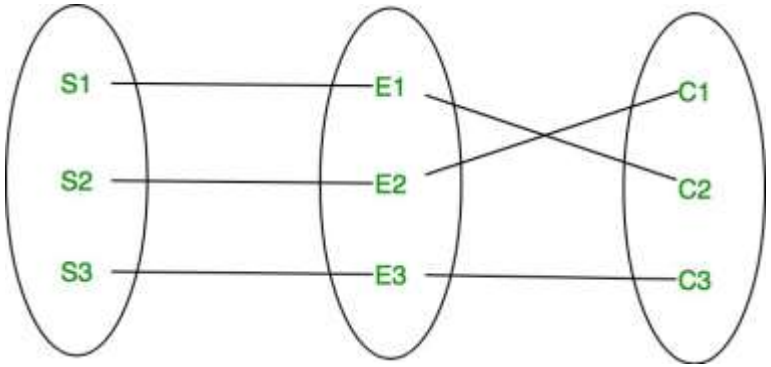
Alternate Key/Secondary Key

- One candidate key is designated as the primary key and the remaining candidate keys are called Alternate keys

- In relational models, there are Candidate Key, Primary key, alternate key, foreign key, search key, parent key, encryption keys, decryption key and so on.

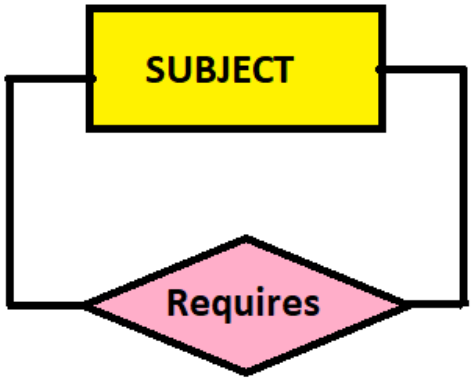
Relationship set:

A set of relationships of same type is known as relationship set. The following relationship set depicts S1 is enrolled in C2, S2 is enrolled in C1 and S3 is enrolled in C3.



Degree of relationship set

- The number of different entity sets participating in a relationship set is called as degree of a relationship set.

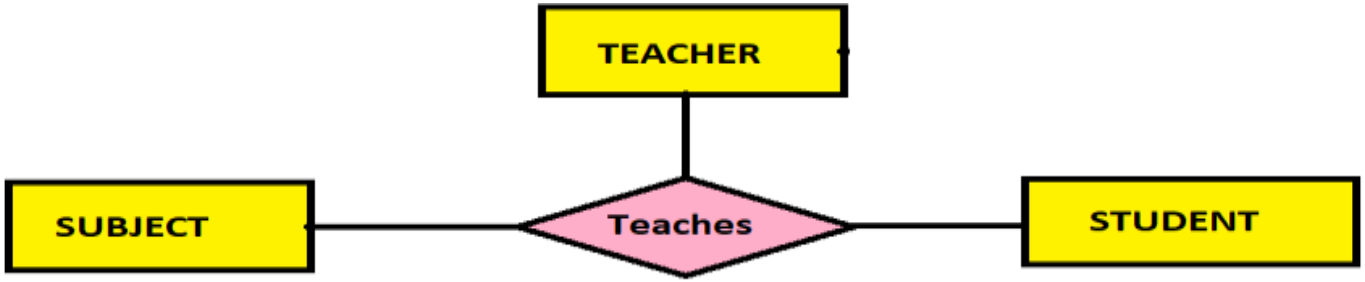


Unary relationship - Only ONE entity set participating in a relation

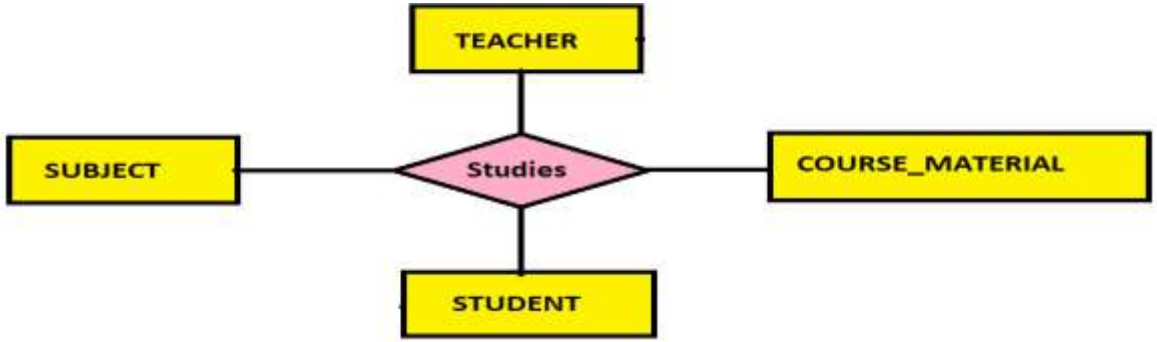
Binary relationship - TWO entities set participating in a relation

Contd…

- **Ternary:** exists when three entities are associated



- **Quaternary :** four entities are associated



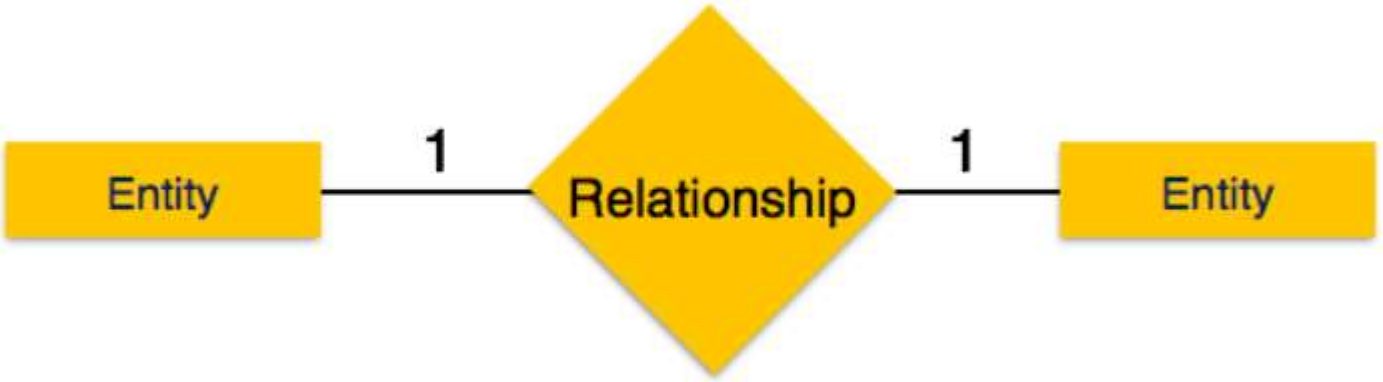
Cardinality

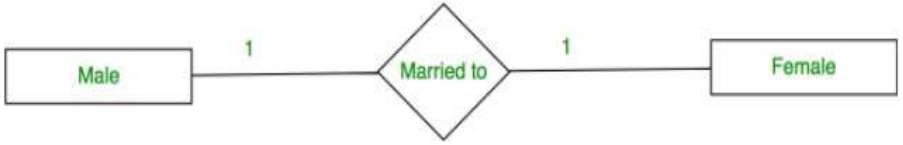
- The number of times an entity of an entity set participates in a relationship set is known as cardinality.

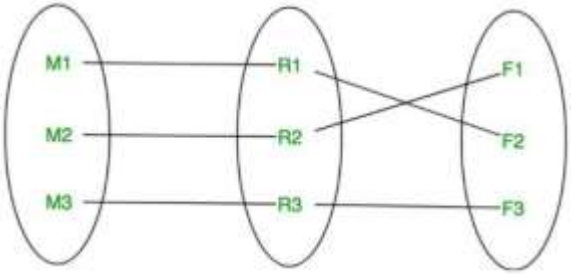
- Cardinality can be of different types:

1. One to one
2. Many to many
3. One to many
4. Many to one

- **One-to-one** − When only one instance of an entity is associated with the relationship, it is marked as '1:1'.







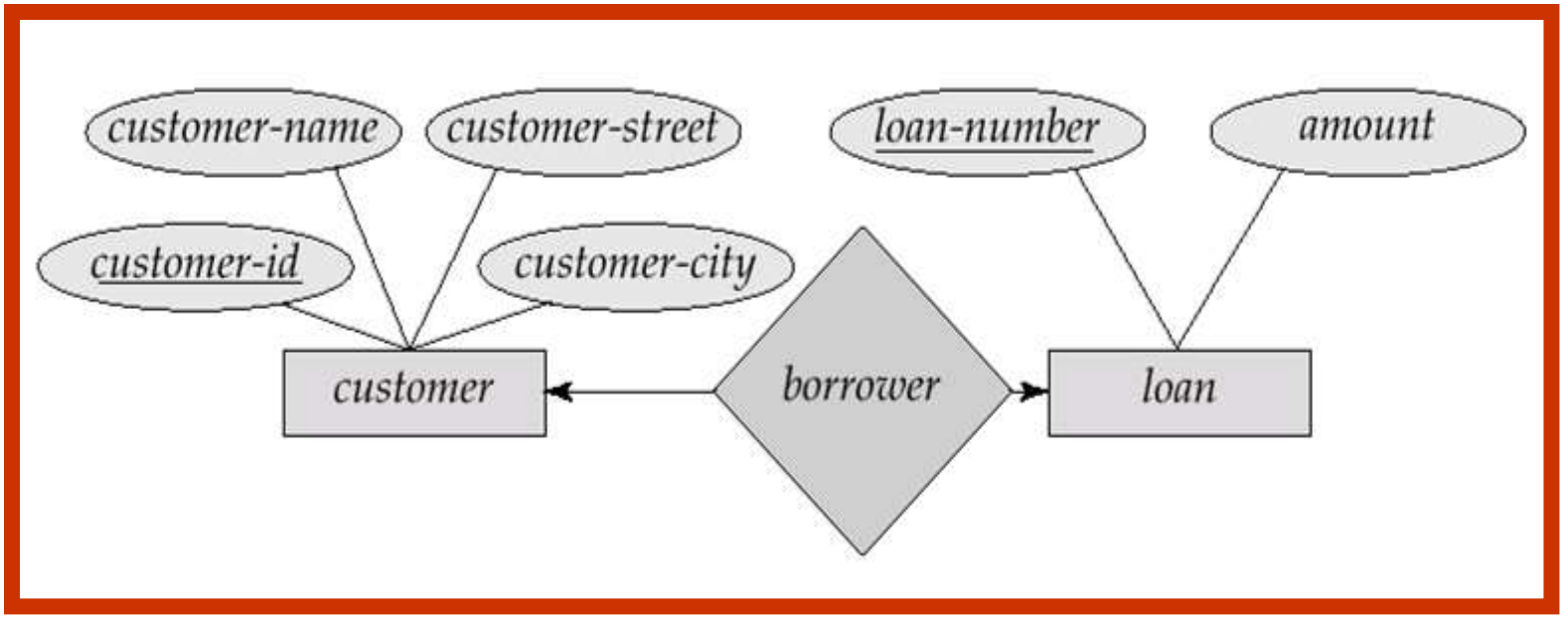
EID	Name	Qualification
1002	Ashwini Kunte	PhD
1003	Mita B	PhD
1007	Tanuja Sarode	PhD
1004	Madhuri Maam	PhD
1111	Ajay Vijay	ME

EID	DID
1002	40
1003	30
1007	10
1004	20

DID	Dname	Location
10	COMPS	NB1FLOOR
20	IT	NB8FLOOR
30	BIOMED	OB3FLOOR
40	EXTC	NB6FLOOR
50	CHEMICAL	OB5FLOOR

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 Constraints

- **Total Participation** − Each entity is involved in the relationship. Total participation is represented by double lines.
- **Partial participation** − Not all entities are involved in the relationship. Partial participation is represented by single lines.

Partial participation

EID	Name	Qualification
1002	Ashwini Kunte	PhD
1003	Mita B	PhD
1007	Tanuja Sarode	PhD
1004	Madhuri Maam	PhD
1111	Ajay Vijay	ME

EID	DID
1002	40
1003	30
1007	10
1004	20

DID	Dname	Location
10	COMPS	NB1FLOOR
20	IT	NB8FLOOR
30	BIOMED	OB3FLOOR
40	EXTC	NB6FLOOR
50	CHEMICAL	OB5FLOOR

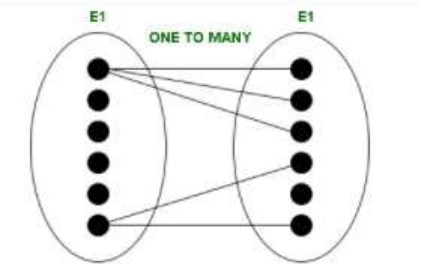
Total Participation

EID	Name	Qualification
1002	Ashwini Kunte	PhD
1003	Mita B	PhD
1007	Tanuja Sarode	PhD
1004	Madhuri Maam	PhD
1111	Ajay Vijay	ME

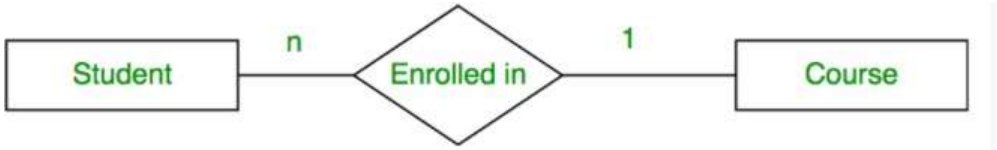
EID	DID
1002	40
1003	30
1007	10
1004	20
1111	50

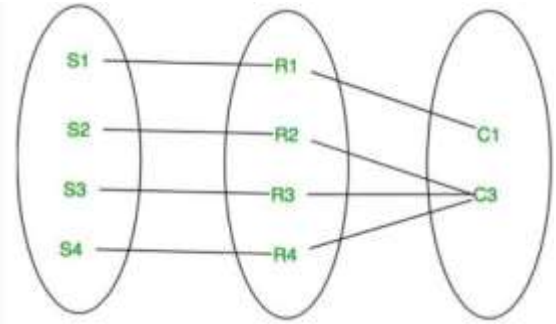
DID	Dname	Location
10	COMPS	NB1FLOOR
20	IT	NB8FLOOR
30	BIOMED	OB3FLOOR
40	EXTC	NB6FLOOR
50	CHEMICAL	OB5FLOOR

- **One-to-many** − When more than one instance of an entity is associated with a relationship, it is marked as '1:N'. The following image reflects that only one instance of entity on the left and more than one instance of an entity on the right can be associated with the relationship.



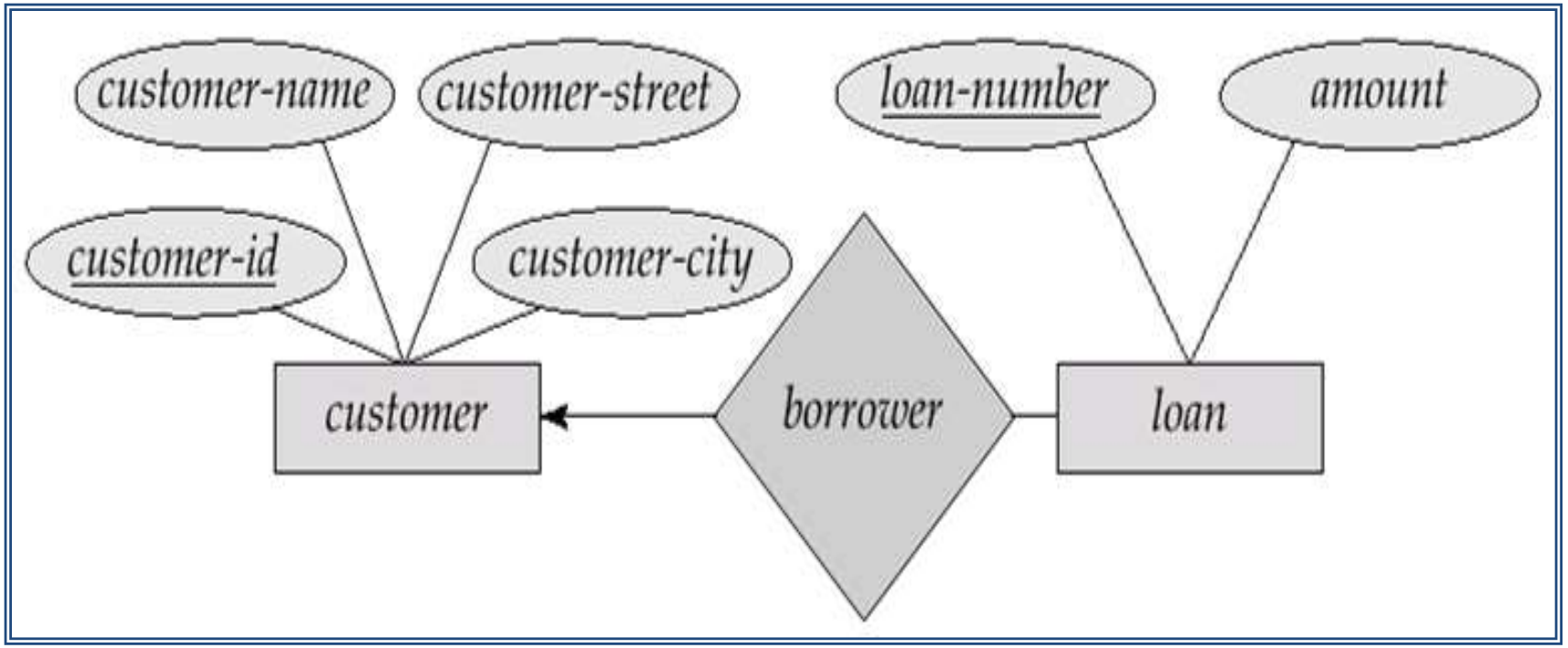
- **<u>Many to one:</u>**





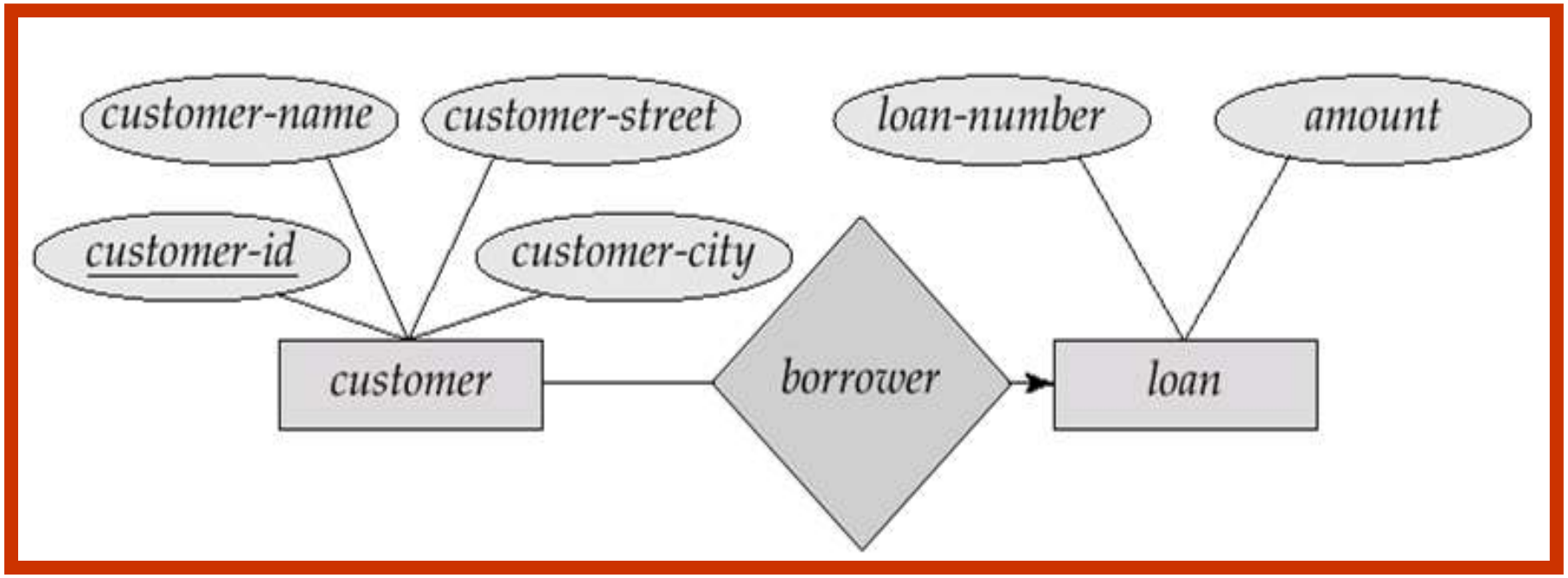
One-To-Many Relationship

- In the one-to-many relationship a loan is associated with at most one customer via *borrower*, a customer is associated with several (including 0) loans via *borrower*

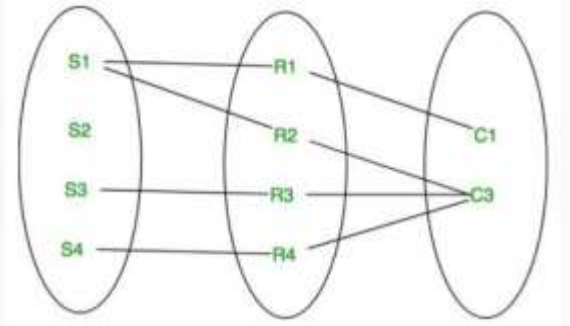


Many-To-One Relationships

- In a many-to-one relationship a loan is associated with several (including 0) customers via *borrower*, a customer is associated with at most one loan via *borrower*

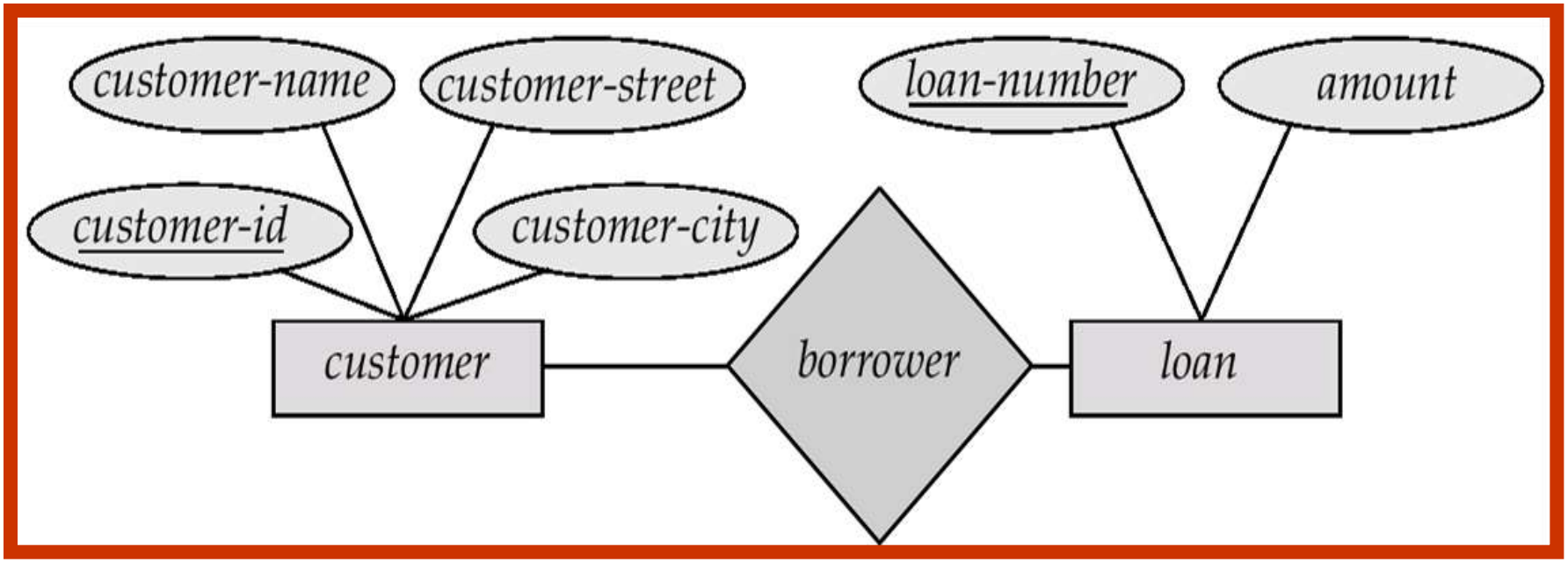


- **Many-to-many** − The following image reflects that more than one instance of an entity on the left and more than one instance of an entity on the right can be associated with the relationship. It depicts many-to-many relationship.



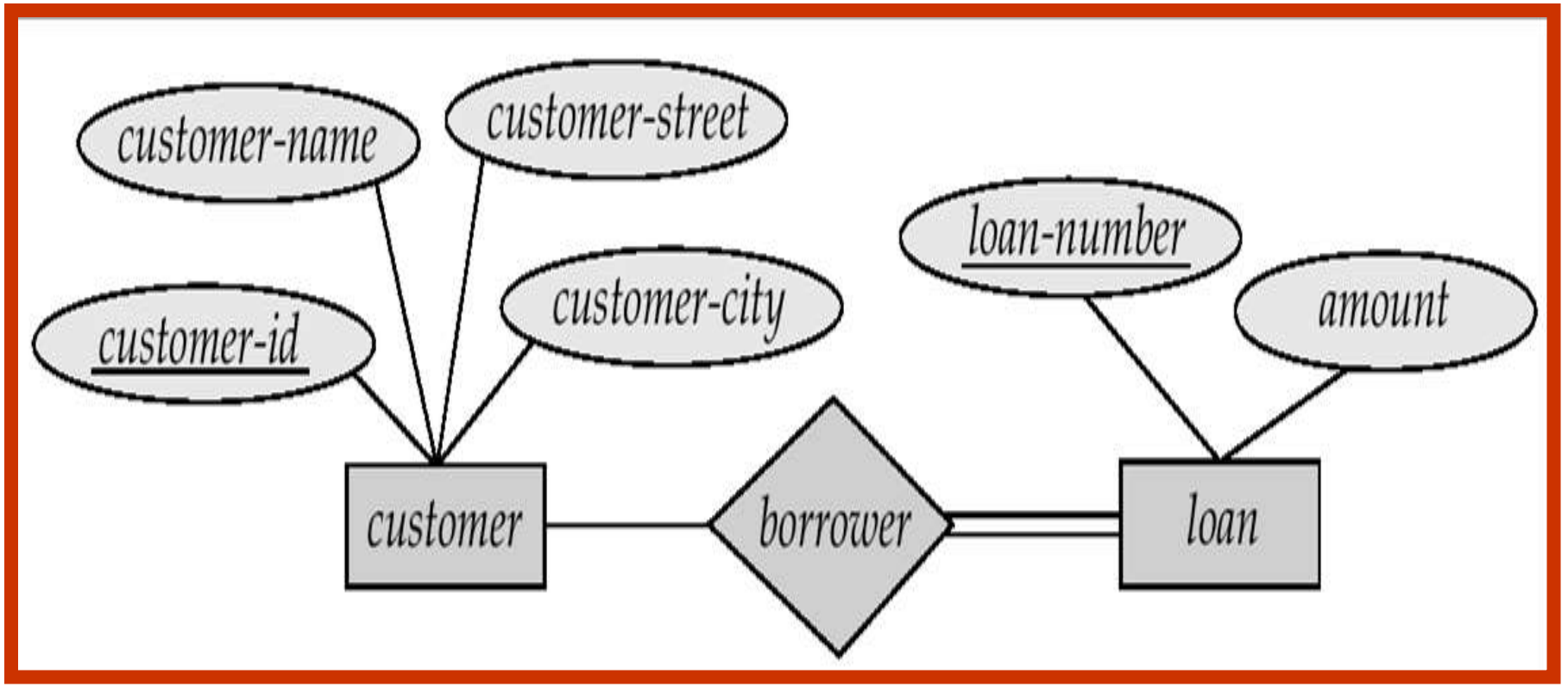
Many-To-Many Relationship

- A customer is associated with several (possibly 0) loans via borrower
- A loan is associated with several (possibly 0) customers via borrower

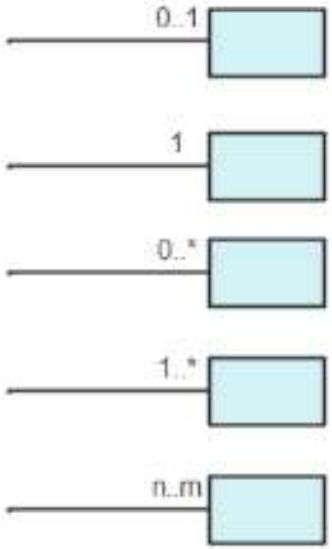


Participation of an Entity Set in a Relationship Set

- Total participation: indicated by double line
- Partial participation



Alternative notations



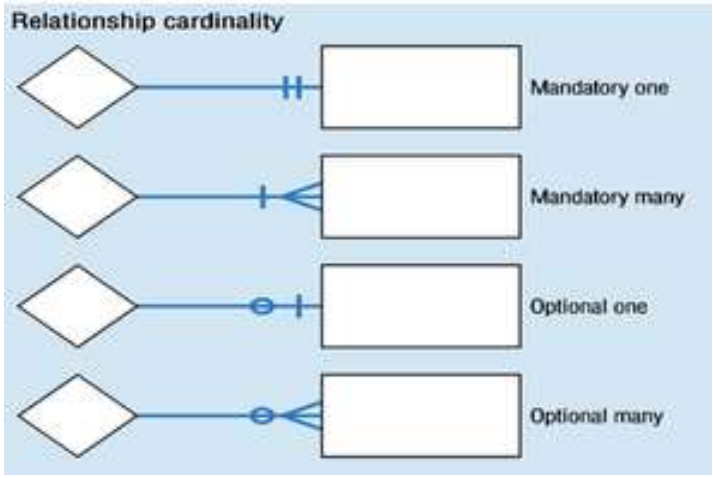
Zero or One

One and only One

Zero or More

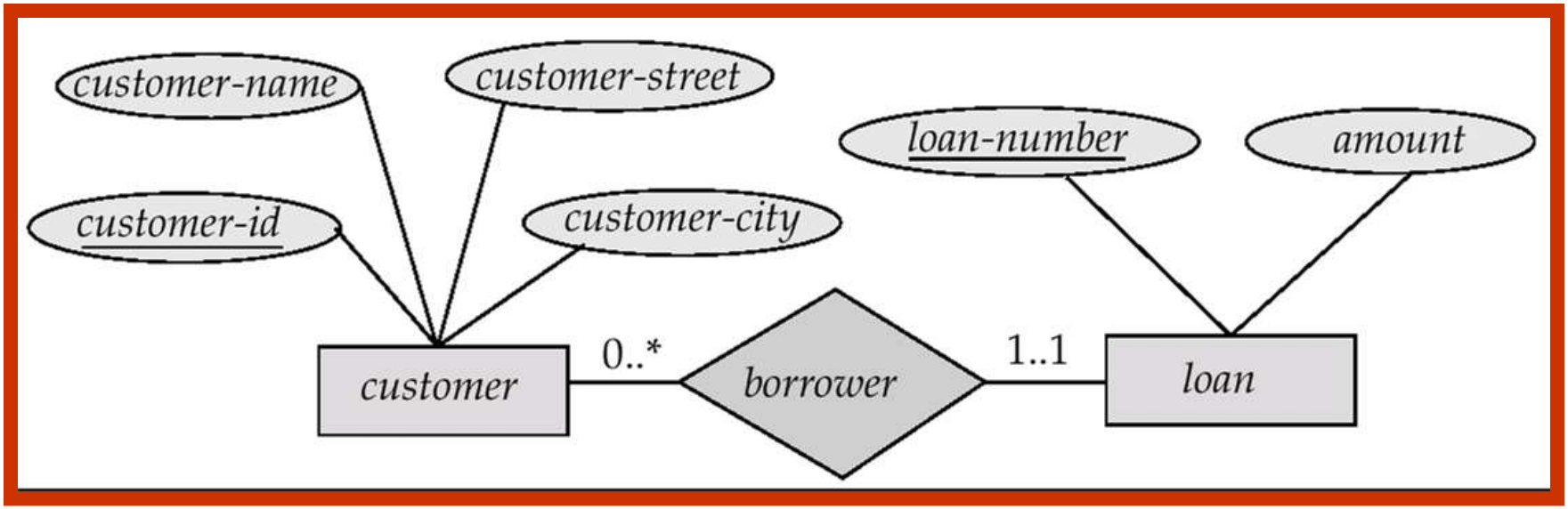
One or More

The range is specified



Alternative Notation for Cardinality Limits

- Cardinality limits can also express participation constraints



Entity set Types

- **Strong Entity : Entities having its own attribute as primary keys are called strong entity.**

  E.g. : STUDENT has STUDENT_ID as primary key.

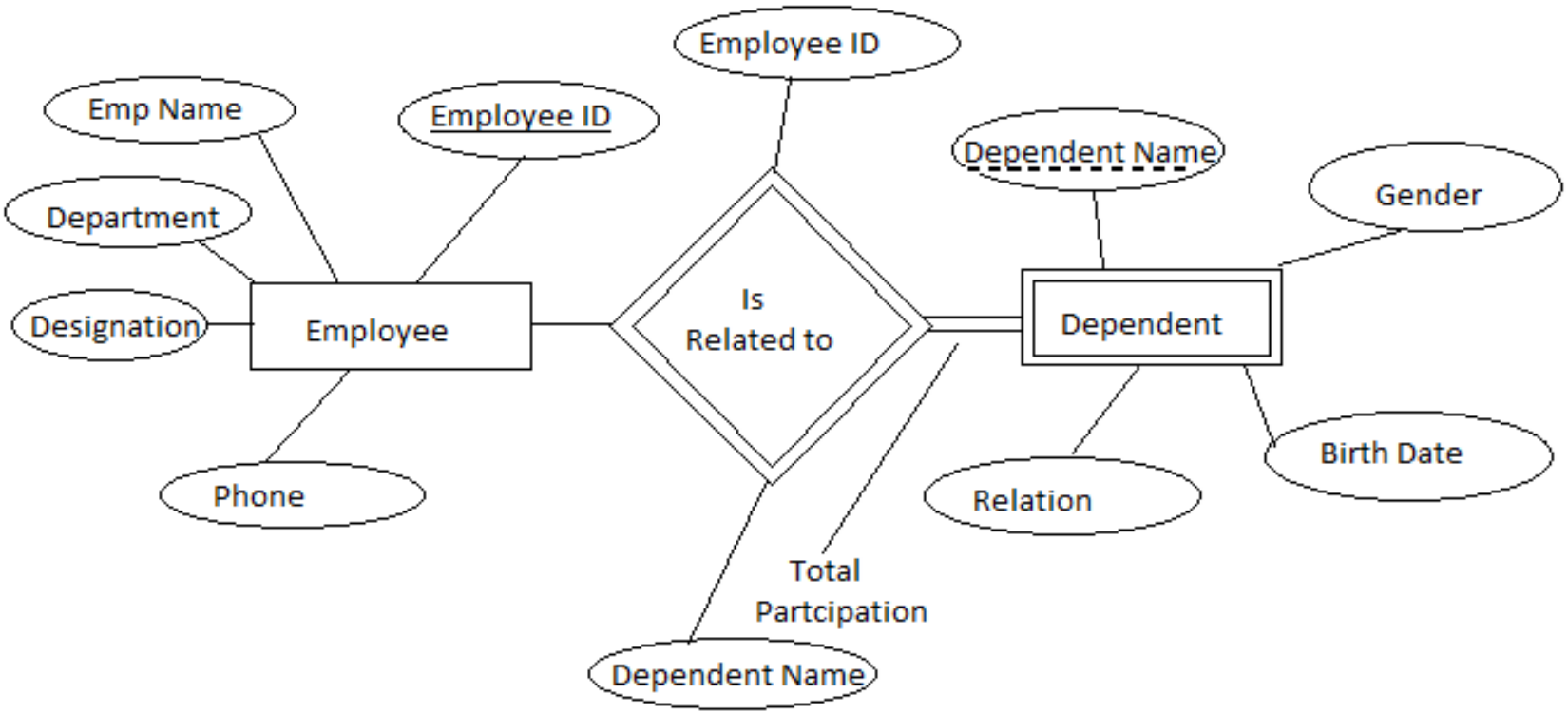
- **Weak Entity : Entities which cannot form their own attribute as primary key are known** weak entities.

- These entities will derive their primary keys from the combination of its attribute and primary key from its mapping entity.

Dependency

- Entities can be classified as
  - Strong / Parent /Owner/ Dominate
  - Weak entity type / existence- dependent/child/ Subordinate
- Existence- dependency means that if the existence of entity x depends on the existence of entity y, then x is said to be existence dependent on y
- A strong entity is not existence dependent whereas weak entities are

Contd…



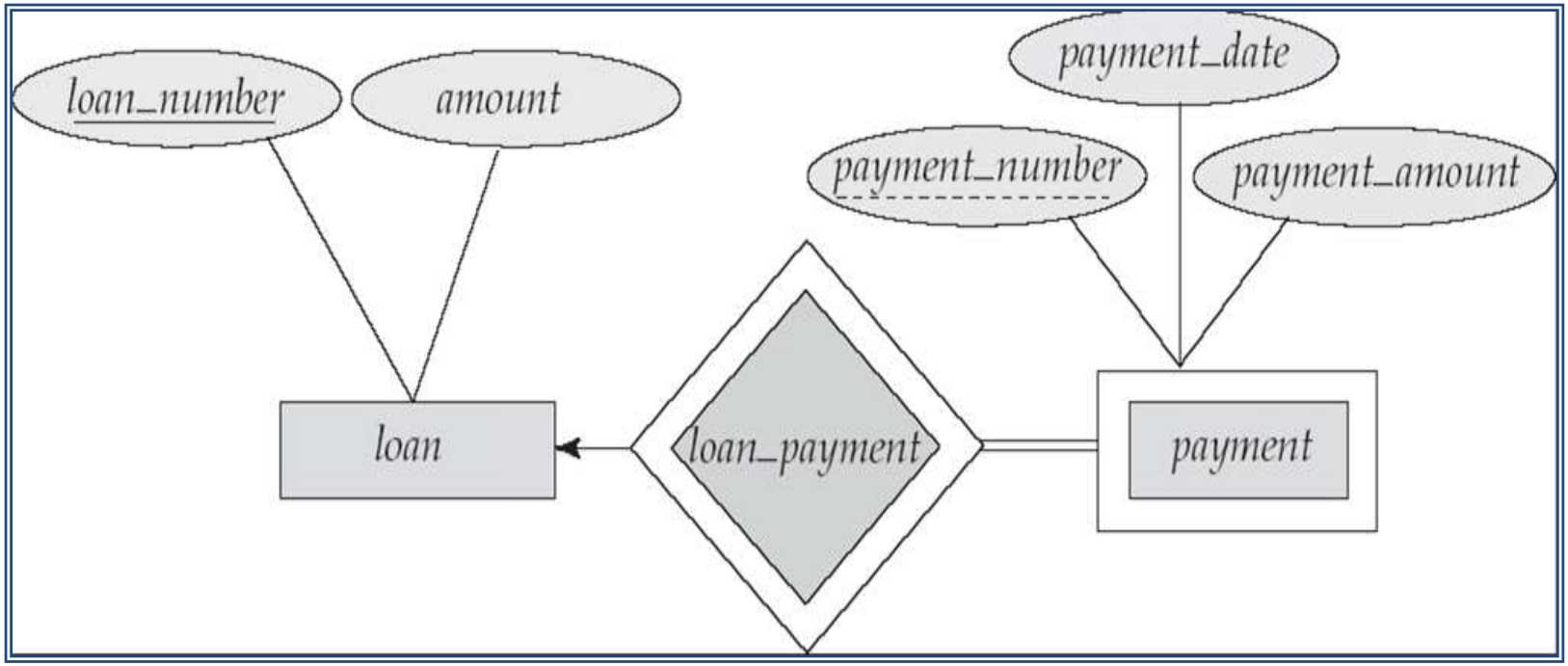
Identifying Relationship set attributes are {Employee ID, Dependent Name}

Weak Entity Sets

- The existence of a weak entity set depends on the existence of a **identifying entity set**
  - It must relate to the identifying entity set via a total, one-to-many relationship set from the identifying to the weak entity set
  - Identifying relationship depicted using a double diamond
- The **discriminator** (*or partial key)* of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set.
- The primary key of a weak entity set is formed by the primary key of the strong entity set on which the weak entity set is existence dependent, plus the weak entity set's discriminator.

Weak Entity Sets (Cont.)

- We depict a weak entity set by double rectangles.
- We underline the discriminator of a weak entity set with a dashed line.
- payment_number – discriminator of the *payment* entity set
- Primary key for *payment* – (*loan_number, payment_number)*

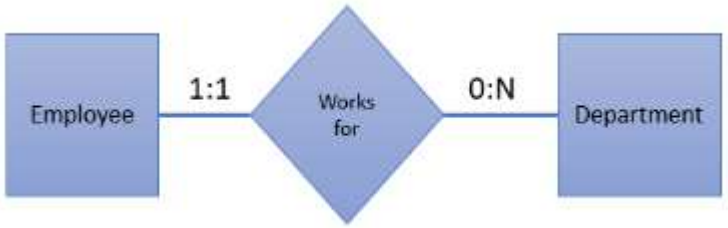


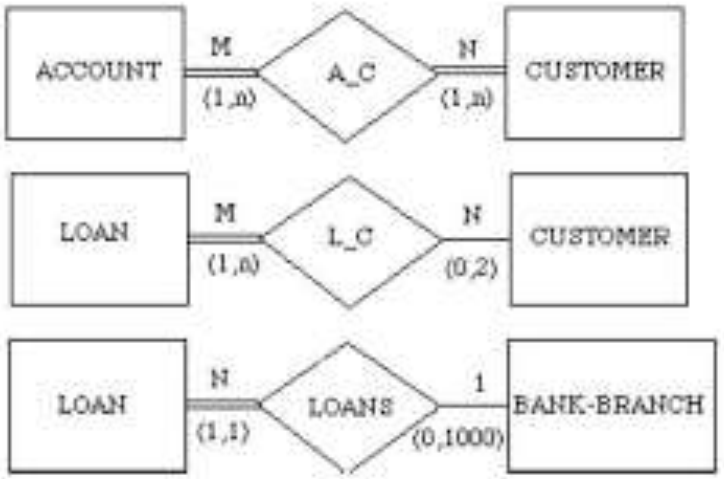
Comparison between Strong and Weak entity

Strong Entity Set	Weak Entity Set
it has its own primary key.	It does not save sufficient attributes to form a primary Key on its own.
It is represented by a rectangle.	It is represented by a double rectangle.
It contains a primary key represented by an underline.	It contains a Partial Key or discriminator represented by a dashed underline.
The member of strong entity set is called as dominant entity set.	The member of weak entity set is called as subordinate entity set.
The Primary Key is one of its attributes which uniquely Identifies its member.	The Primary Key of weak entity set is a combination of partial Key and Primary Key of the strong entity set.
The relationship between two strong entity set is represent by a diamond symbol.	The relationship between one strong and a weak entity set is represented by a double diamond sign. It is known as identifying relationship.
The line connecting strong entity set with the relationship is single	The line connecting weak entity set with the identifying relationship is double.
Total participation in the relationship may or may not exist.	Total participation in the identifying relationship always exists.

ER diagram with cardinality limits

- E-R diagrams also provide a way to indicate more complex constraints on the number of times each entity participates in relationships in a relationship set.
- An edge between an entity set and a binary relationship set can have an associated minimum and maximum cardinality, shown in the form **m..n**, where m is the minimum and n the maximum cardinality.





Steps in ER Modelling

- Usually the following five steps are followed to generate ER models
  1. Identify the entity set.
  2. Identify the relevant attributes.
  3. Identify the prime attribute.
  4. Find relationships between entity set.
  5. Draw a complete ER model.

Example:

Draw an ER Model for an University database application where

a) A University has many departments.
b) Each department has multiple instructors; one among them is the head of the department.
c) An instructor belongs to only one department.
d) Each department offers multiple courses, each of which is taught by a single instructor.
e) A student may enroll for many courses offered by different departments.

Example:

Step 1: Identify the entity set

- From the given question, we can identify the following entity sets.
  1. DEPARTMENT
  2. COURSE
  3. INSTRUCTOR
  4. STUDENT
- "Head of the department" is NOT an entity set; it is relationship between the INSTRUCTOR and DEPARTMENT entities.

Step 2: Identify the relevant attributes

- For the **DEPARTMENT** entity set the relevant attributes are Dept_No, Dept_Name and Location.
- For the **COURSE** entity set the relevant attributes are Course_No, Course_Name, Duration, and Pre-requisite.
- For the **INSTRUCTOR** entity set the relevant attributes are Inst_Id, Inst_Name, Room_No, and Telephone_No.
- For the **STUDENT** entity set the relevant attributes are Student_No, Student_Name, and Dob.

Step 3: Identify the Prime (key) attribute

- Dept_No is the key attribute for **DEPARTMENT** entity set.
- Course_No is the key attribute for **COURSE** entity set.
- Inst_Id is the key attribute for **INSTRUCTOR** entity set.
- Student_No is the key attribute for **STUDENT** entity set.

Step 4: Identify the relationship between entity sets

1. Each department has multiple instructors and an instructor belongs to only one department.

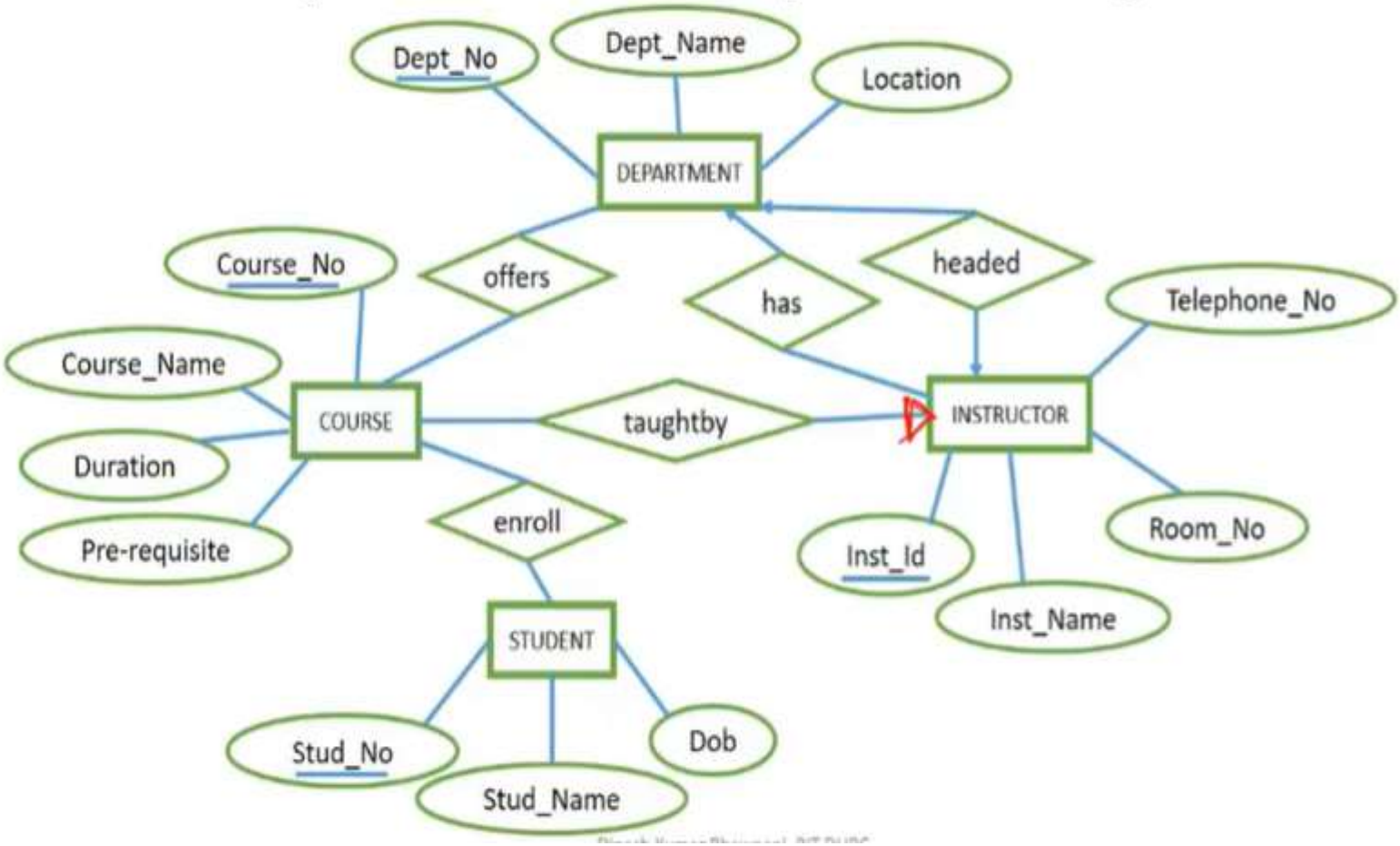
2. Each department has multiple instructors; one among them is the head of the department.

3. Each department offers multiple courses.

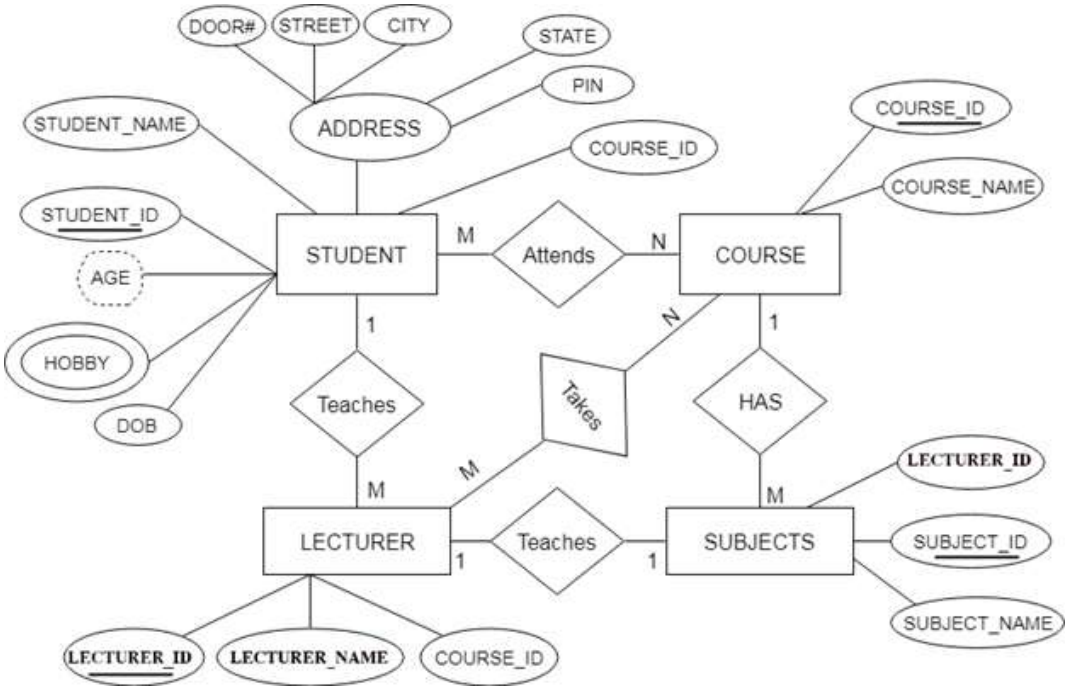
4. Each department offers multiple courses, each of which is taught by a single instructor.

5. A student may enroll for many courses offered by different departments

Step 5: Draw the complete E R Diagram

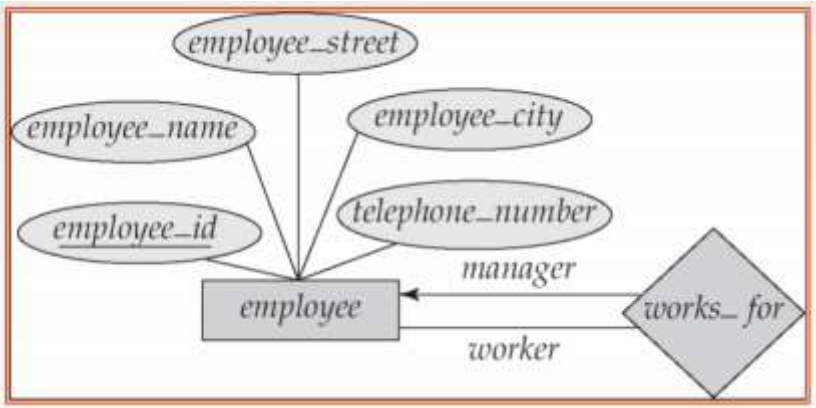


College ER diagram



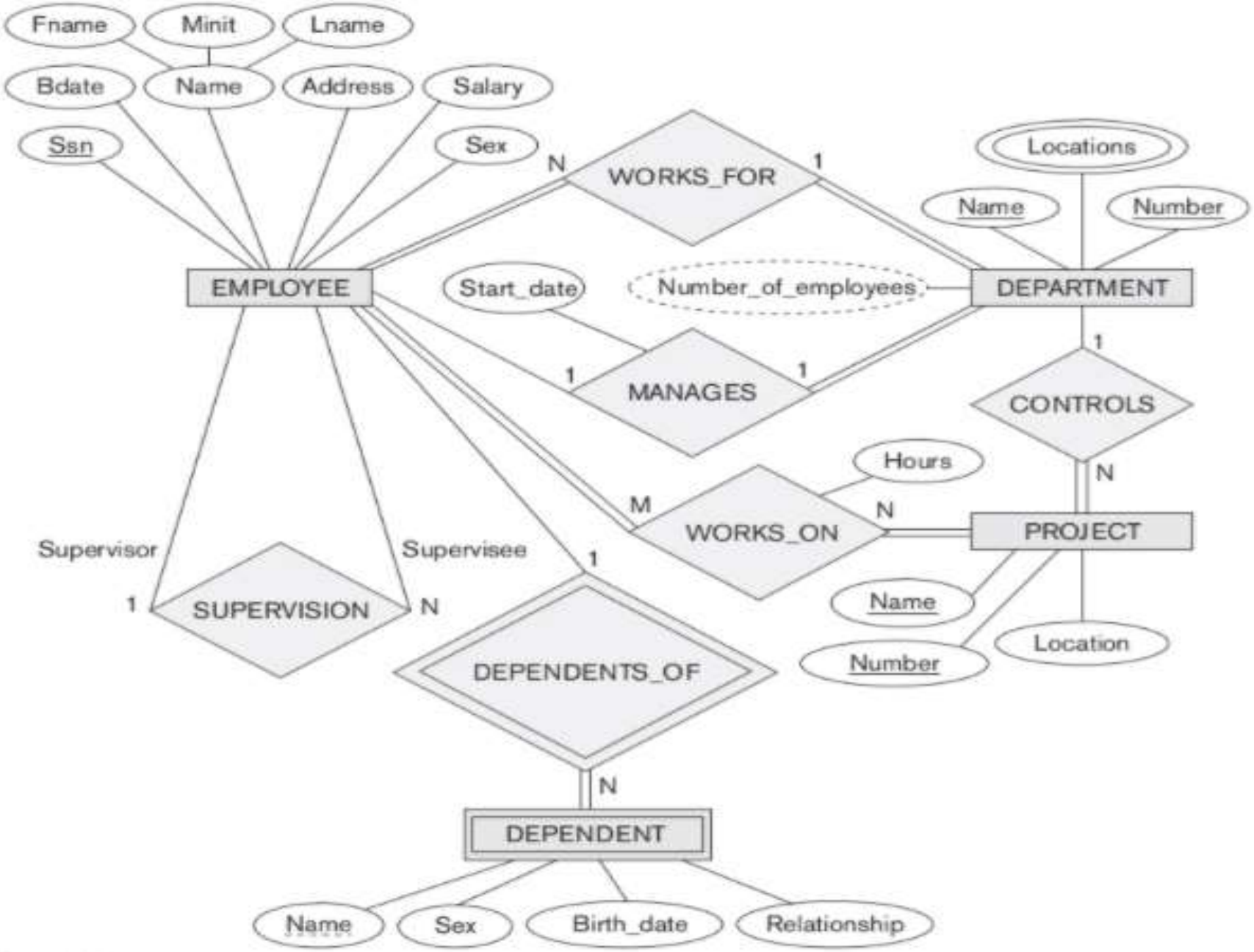
Roles

- 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



Sample Database Application

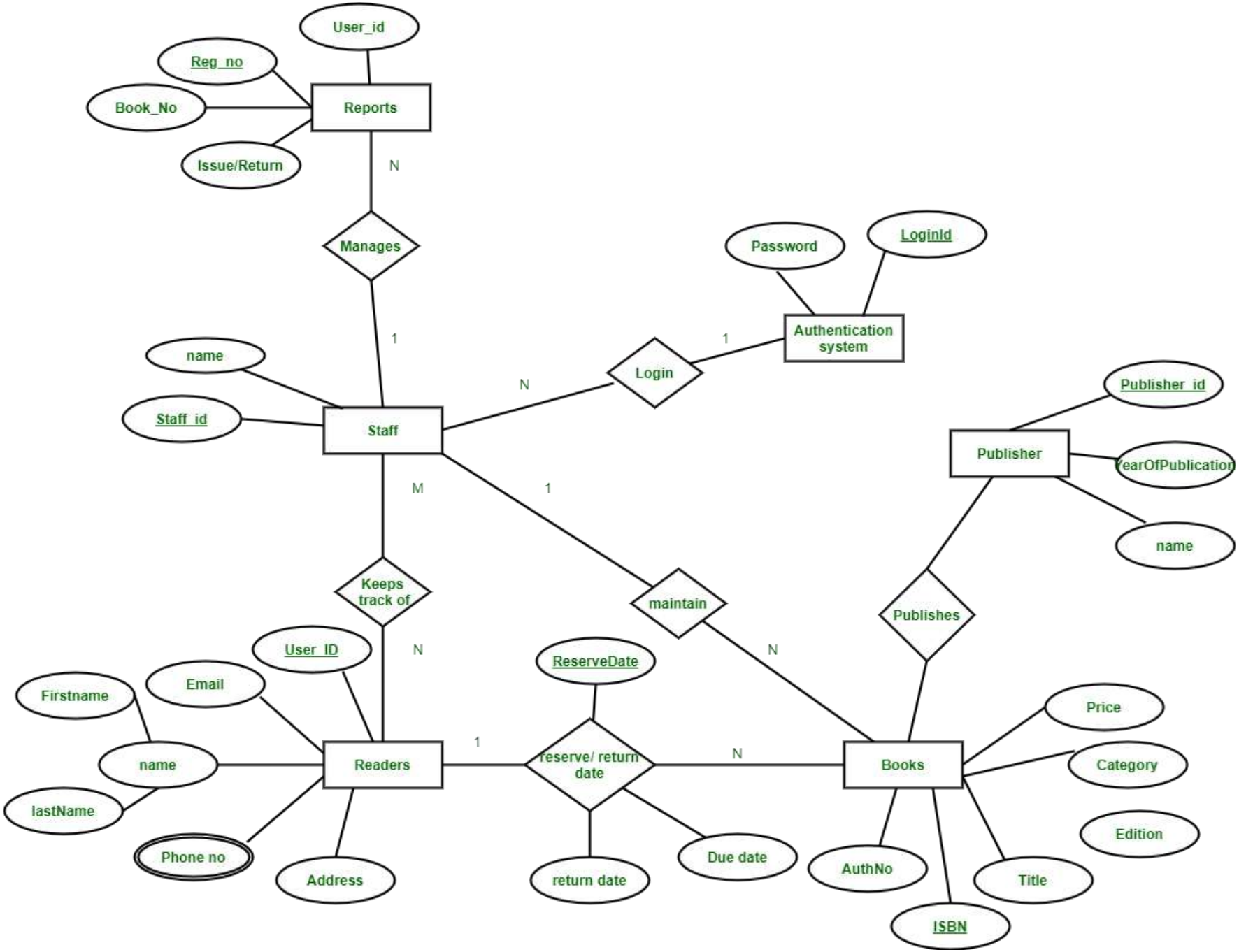
- The company is organized into departments. Each department has a unique name, a unique number, and a particular employee who manages the department. We keep track of the start date when that employee began managing the department. A department may have several locations.
- A department controls a number of projects, each of which has a unique name, a unique number, and a single location.
- We store each employee's name, Social Security number,[2] address, salary, sex (gender), and birth date. An employee is assigned to one department, but may work on several projects, which are not necessarily controlled by the same department. We keep track of the current number of hours per week that an employee works on each project. We also keep track of the direct supervisor of each employee (who is another employee).
- We want to keep track of the dependents of each employee for insurance purposes. We keep each dependent's first name, sex, birth date, and relationship to the employee.


Fname
Minit
Lname
Bdate
Name
Address
Salary
Ssn
Sex
N
WORKS_FOR
1
Locations
Name
Number
EMPLOYEE
Start_date
Number_of_employees
DEPARTMENT
1
1
MANAGES
1
CONTROLS
Hours
N
M
WORKS_ON
N
PROJECT
Supervisor
Supervisee
1
1
SUPERVISION
N
Name
Location
Number
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
Birth_date
Relationship

Library Management System

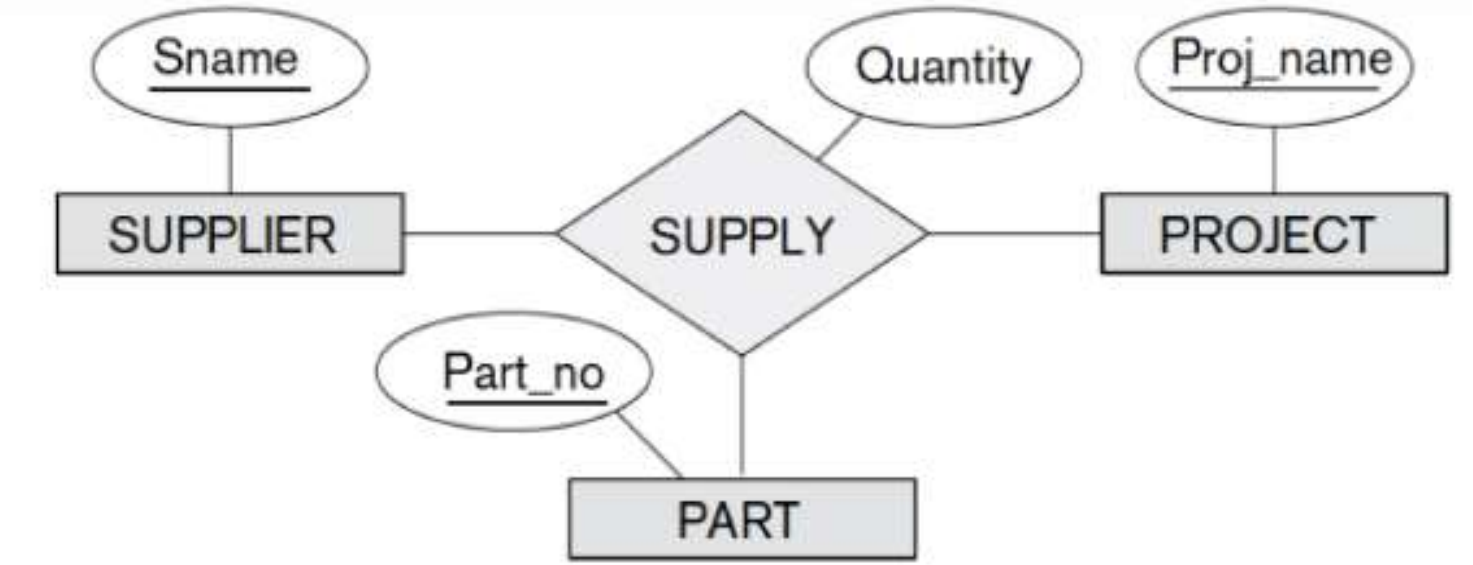
The Library Management System database keeps track of readers with the following considerations –

- The system keeps track of the staff with a single point authentication system comprising login Id and password.
- Staff maintains the book catalog with its ISBN, Book title, price(in INR), category(novel, general, story), edition, author Number and details.
- A publisher has publisher Id, Year when the book was published, and name of the book.
- Readers are registered with their user_id, email, name (first name, last name), Phone no (multiple entries allowed), communication address. The staff keeps track of readers.
- Readers can return/reserve books that stamps with issue date and return date. If not returned within the prescribed time period, it may have a due date too.
- Staff also generate reports that has readers id, registration no of report, book no and return/issue info.

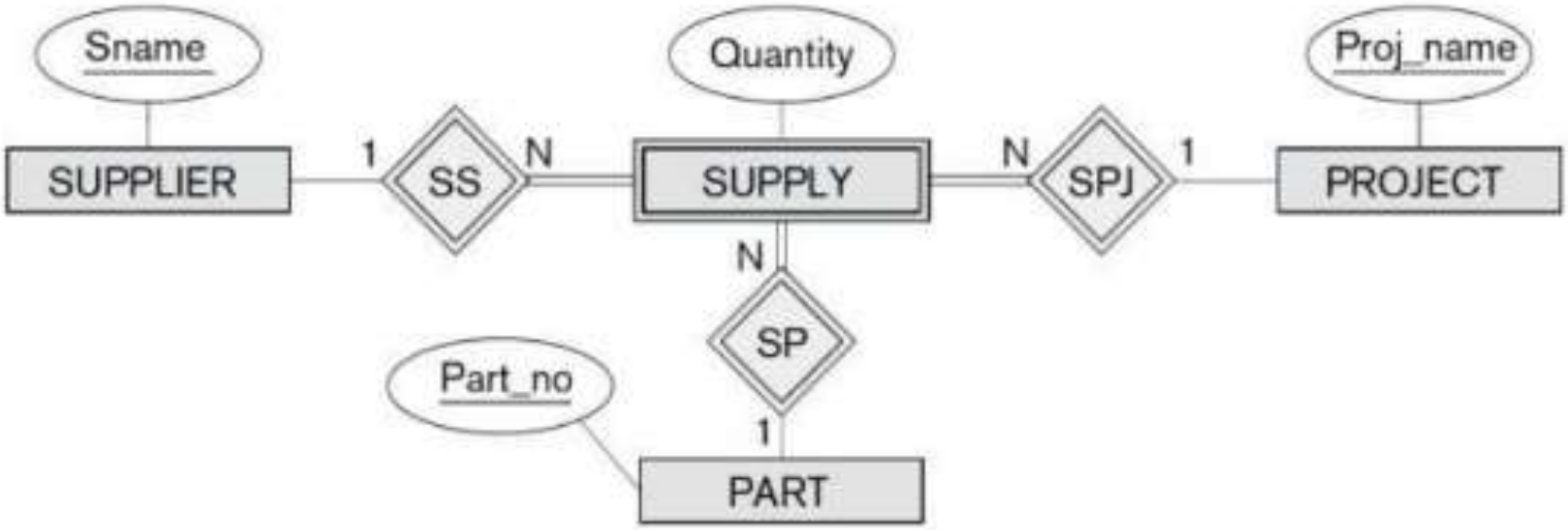

User_id
Reg_no
Book_No
Reports
Issue/Return
N
Manages
1
Password
LoginId
Authentication system
1
name
Staff_id
Staff
N
Login
Publisher_id
Publisher
YearOfPublication
name
M
1
Keeps track of
maintain
Publishes
User_ID
N
N
ReserveDate
Email
Firstname
name
lastName
Readers
1
Reserve/ return date
N
Books
Price
Category
Edition
Title
ISBN
AuthNo
Phone no
Address
return date
Due date

Convert ternary relationship to binary

- How to convert ternary relationship to binary. One way is using weak entity relationship as follows (each relationship is M:N cardinality):
- **Example 1:**

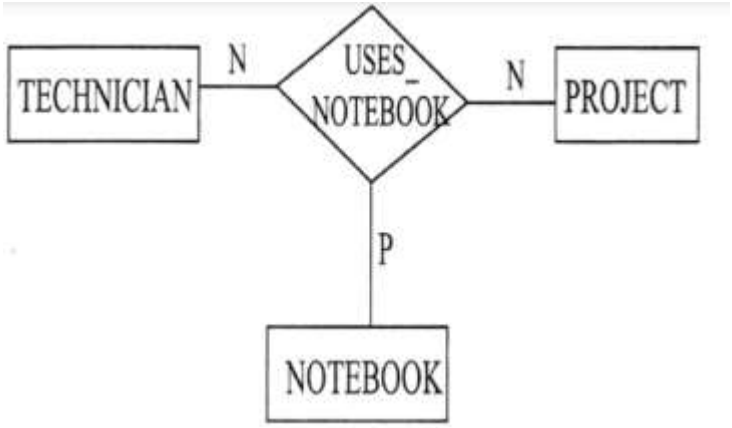


- Convert the upper relationship with weak relationship

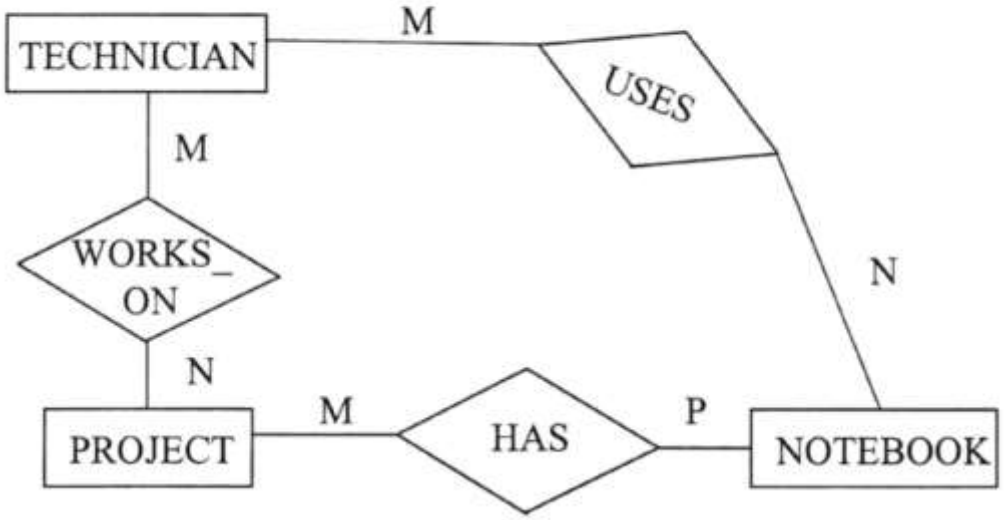


Example 2:

- "if each technician can be working on several projects and uses the same notebooks on each project, then we can decompose 3-ary relationship into binary relationships“ as follows:



=> can *decompose* 3-ary relationship into binary relationships



Design Issues

- **Use of entity sets vs. attributes**
  Choice mainly depends on the structure of the enterprise being modeled, and on the semantics associated with the attribute in question.
- **Use of entity sets vs. relationship sets**
  Possible guideline is to designate a relationship set to describe an action that occurs between entities
- **Binary versus n-ary relationship sets**
  Although it is possible to replace any nonbinary (*n*-ary, for $n > 2$) relationship set by a number of distinct binary relationship sets, a *n*-ary relationship set shows more clearly that several entities participate in a single relationship.
- **Placement of relationship attributes**

Extended Entity Relationship (EER) Diagram

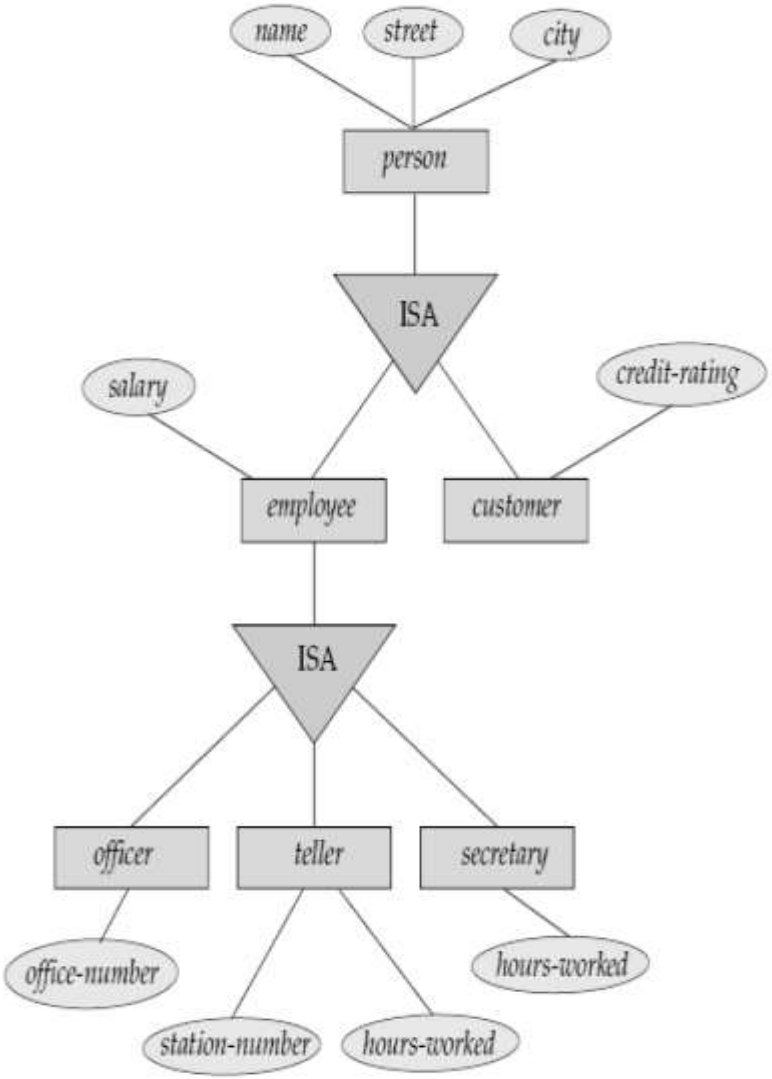
Extended/Enhanced Entity Relationship Model

- Basic ER Model concept were not able to represent requirement of newer and complex applications
- It support additional semantic concepts
- EER model support all the concept of the original ER models together with additional concepts:
  - Super class and subclass Entity -set types
  - Specialization
  - Generalization
  - Attributes Inheritance
  - Categorization
  - Aggregation

Contd…

- EER = ER Model + Specialization
  - Generalization
  - Attributes Inheritance
  - Categorization
  - Aggregation

Contd…


name
street
city
person
ISA
salary
credit-rating
employee
customer
ISA
officer
teller
secretary
office-number
station-number
hours-worked
hours-worked

Attribute Inheritance

- **Attribute Inheritance**
- A crucial property of the higher- and lower-level entities created by specialization and generalization is **attribute inheritance**.
- The attributes of the higher-level entity sets are said to be **inherited** by the lower-level entity sets.
- Eg: *customer* and *employee* inherit the attributes of *person*. Thus, *customer* is described by its *name*, *street*, and *city* attributes, and additionally a *customer-id* attribute

employee is described by its *name*, *street*, and *city* attributes, and additionally *employee-id* and *salary* attributes.

- A lower-level entity set (or subclass) also inherits participation in the relationship sets in which its higher-level entity (or superclass) participates.

• The *officer*, *teller*, and *secretary* entity sets can participate in the *works-for* relationship set, since the superclass *employee* participates in the *works-for* relationship

Specialization

- An entity set may be specialized by more than one distinguishing feature.

  Eg: specialization could be based on whether the person is
  - *temporary-employee and*
  - *permanent-employee.*
- When more than one specialization is formed on an entity set, a particular entity may belong to multiple specializations.

  Eg: a given employee may be a temporary employee who is a secretary.

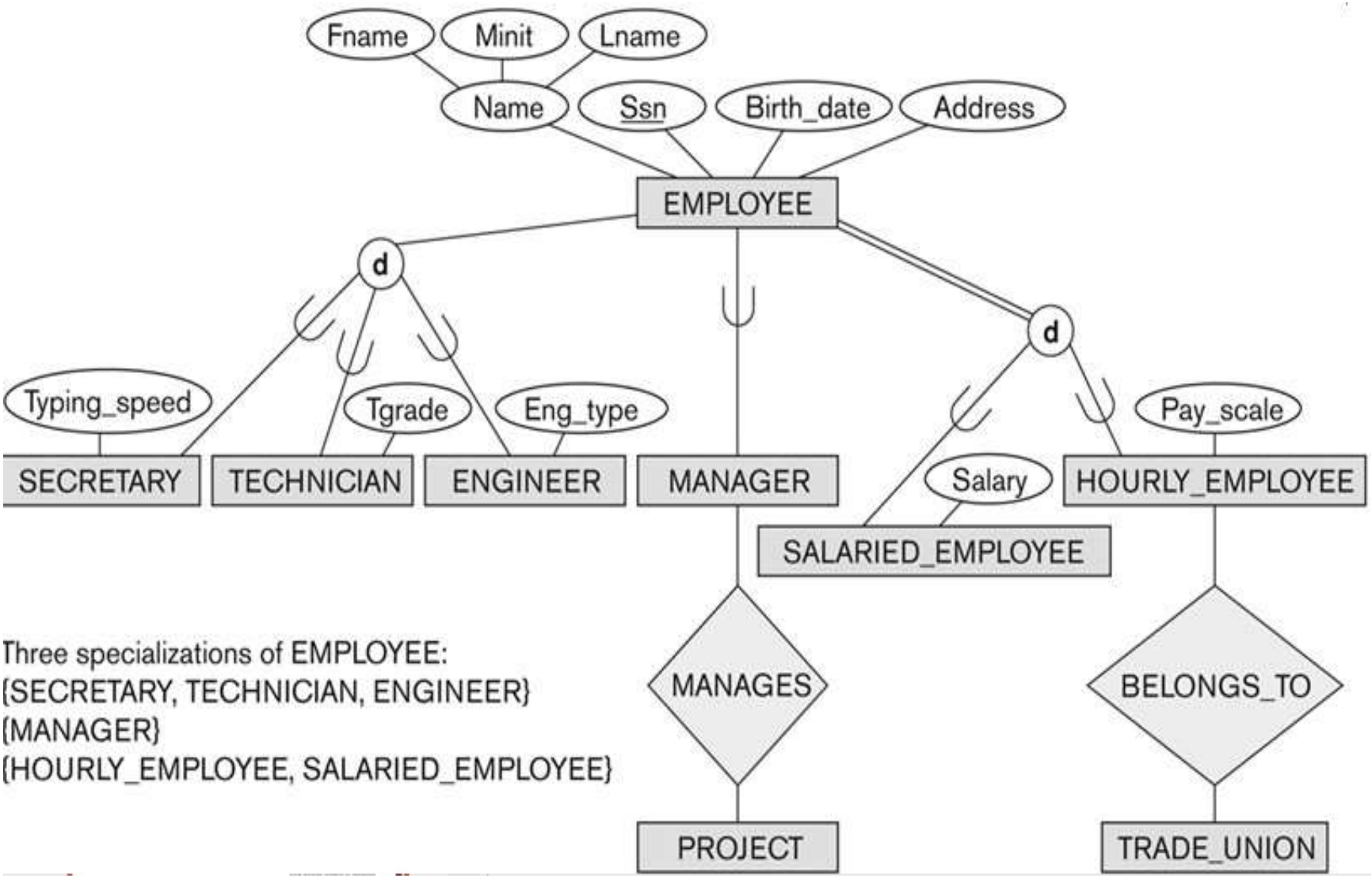
- Specialization is depicted by a *triangle* component labelled **ISA**
- The label ISA stands for "is a" and represents, for example, that a customer "is a" person.
- The ISA relationship may also be referred to as a **superclass-subclass** relationship.
- Higher- and lower-level entity sets are depicted as regular entity a set—that is, as rectangles containing the name of the entity set.

Specialization

...

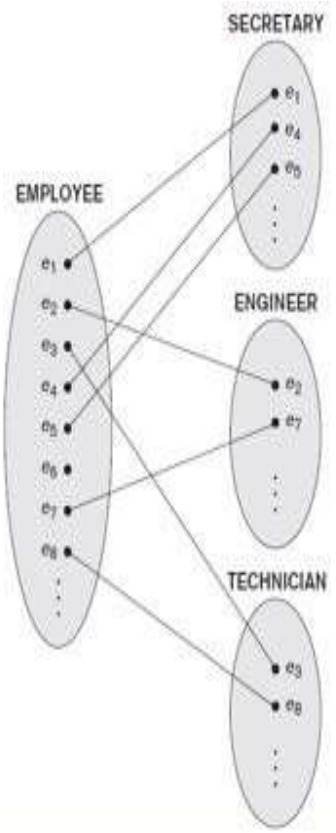
- Example: Another specialization of EMPLOYEE based on *method of pay* is {SALARIED_EMPLOYEE, HOURLY_EMPLOYEE}.
  - Superclass/subclass relationships and specialization can be diagrammatically represented in EER diagrams
  - Attributes of a subclass are called ***specific* or *local* attributes**.
    - For example, the attribute TypingSpeed of SECRETARY
  - The subclass can also participate in specific relationship types.
    - For example, a relationship BELONGS_TO of HOURLY_EMPLOYEE

Specialization…



Specialization…

- **Specialization**
  - Process of defining a set of subclasses of an entity type
  - Defined based on some distinguishing characteristic of the entities in the superclass
- Subclass can define:
  - **Specific attributes**
  - **Specific relationship types**

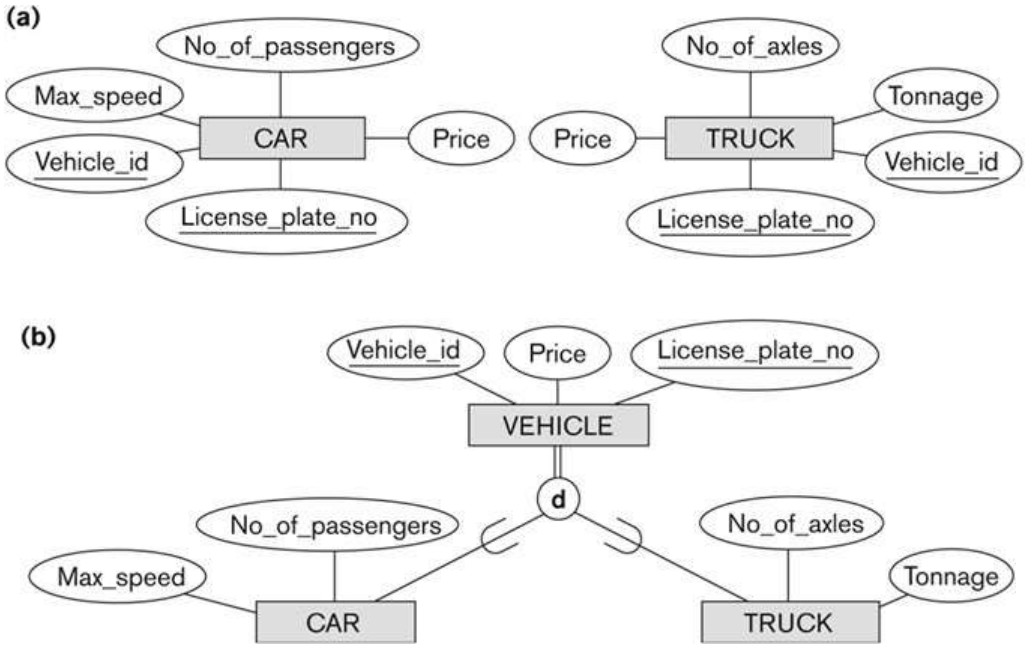


Instances of specialization

Generalization

- The refinement from an initial entity set into successive levels of entity sub-groupings represents a **top-down** design process
- The design process may also proceed in a **bottom-up** manner, in which multiple entity sets are synthesized into a higher-level entity set on the basis of common features.
- The database designer may have first identified a ***customer entity set*** with the attributes *name*, *street*, *city*, and *customer-id*, and an **employee entity set** with the attributes *name*, *street*, *city*, *employee-id*, and *salary*.
- **Generalization** is a containment relationship that exists between a *higher-level* entity set and one or more *lower-level* entity sets.
- Higher- and lower-level entity sets also may be designated by the terms **superclass** and **subclass**, respectively.

Generalization…

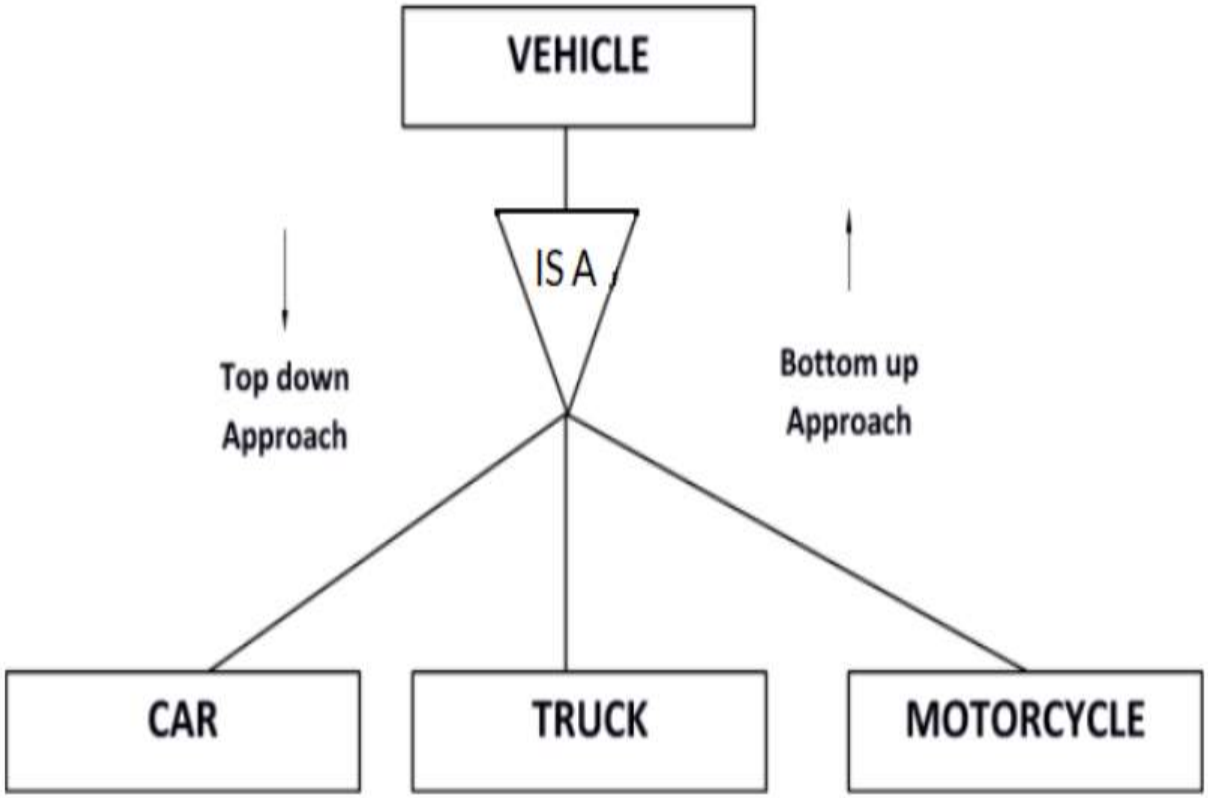


Generalization (a) Two entity types, CAR and TRUCK
(b) Generalizing CAR and TRUCK into the superclass VEHICLE

Specialization and Generalization

- **Generalization** is a process of generalizing an entity which contains generalized attributes or properties of generalized entities.
- It is a Bottom up process i.e. consider we have 3 sub entities Car, Truck and Motorcycle. Now these three entities can be generalized into one super class named as Vehicle.
- **Specialization** is a process of identifying subsets of an entity that share some different characteristic. It is a top down approach in which one entity is broken down into low level entity.
- In above example Vehicle entity can be a Car, Truck or Motorcycle.

Specialization and Generalization



Constraints on Generalization

- **Condition-defined.** In condition-defined lower-level entity sets, membership is evaluated on the basis of whether or not an entity satisfies an explicit condition or predicate. Since all the lower-level entities are evaluated on the basis of the same attribute, this type of generalization is said to be **attribute-defined.**

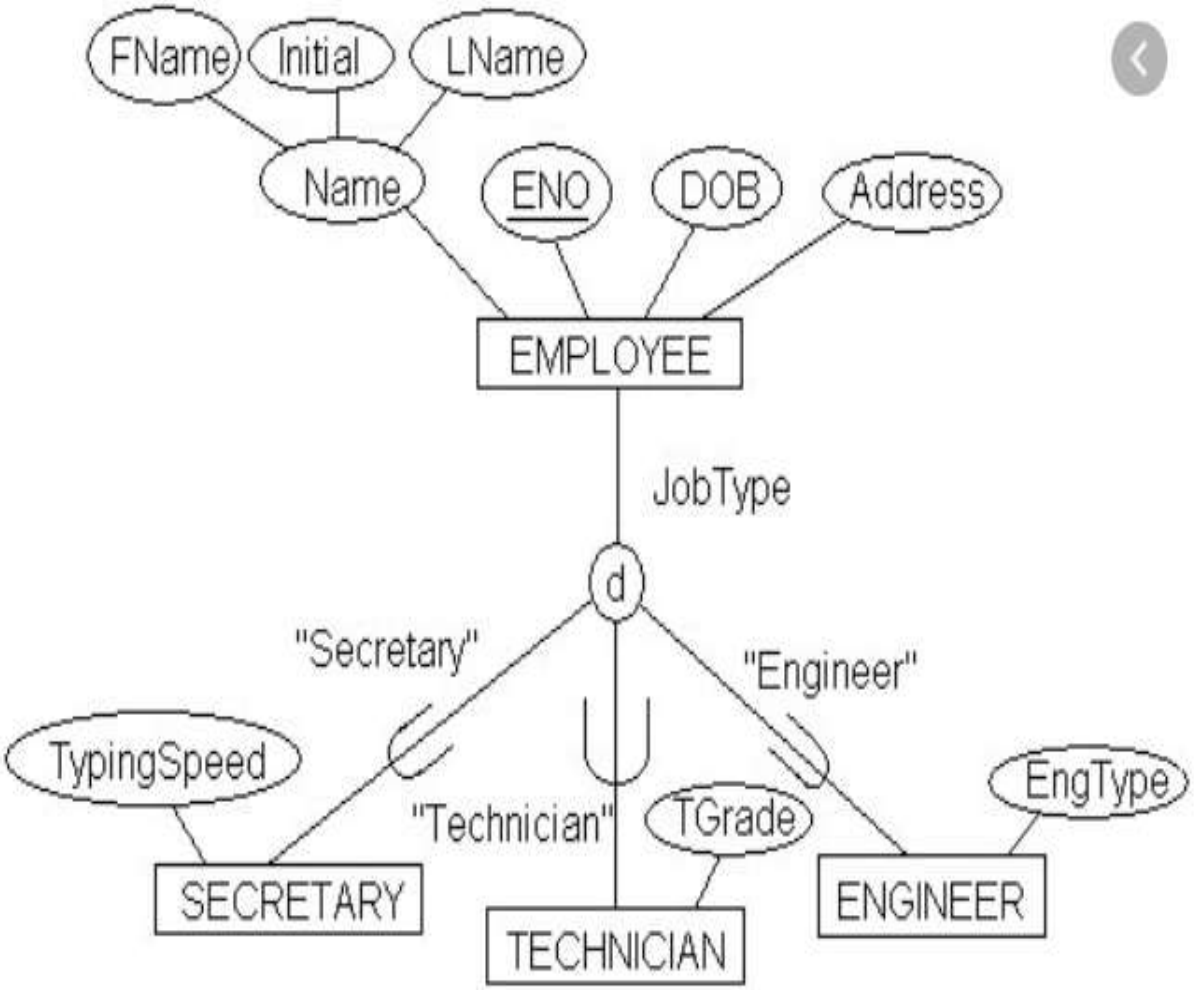
For example, assume that the higher-level entity set *account* has the attribute *account-type*. All *account* entities are evaluated on the defining *account-type* attribute.

Entities that satisfy the condition

account-type = "savings account" : are allowed to belong to the lower-level entity set *saving account*.

All entities that satisfy the condition

account-type = "checking account" : are included in *checking account*.


FName
Initial
LName
Name
ENO
DOB
Address
EMPLOYEE
JobType
d
"Secretary"
"Technician"
"Engineer"
TypingSpeed
TGrade
EngType
SECRETARY
TECHNICIAN
ENGINEER

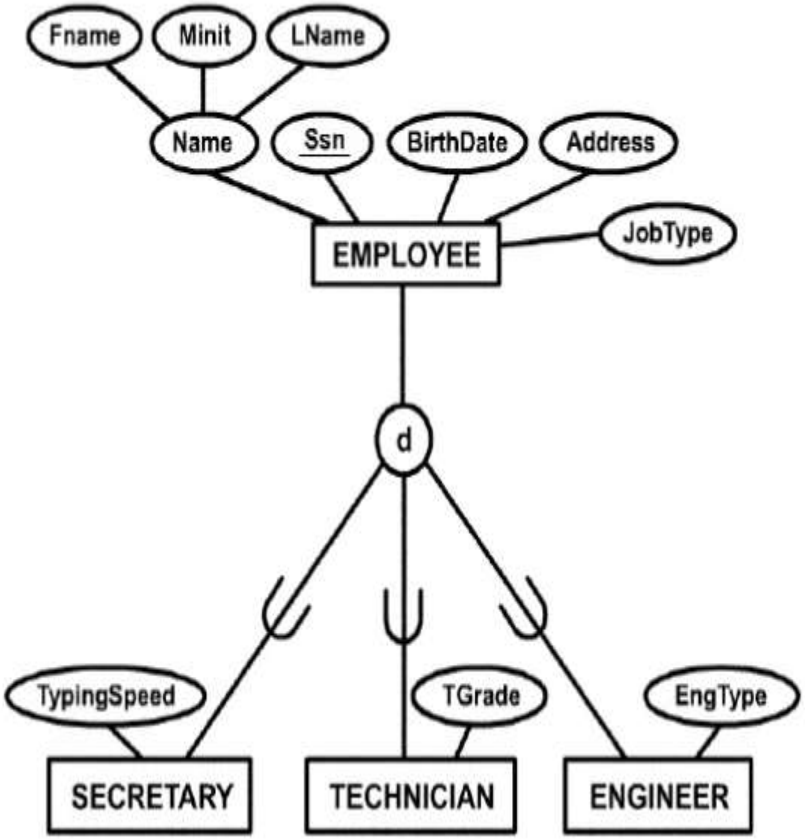
Contd…

- **User-defined.** User-defined lower-level entity sets are not constrained by a membership condition; rather, the database user assigns entities to a given entity set.

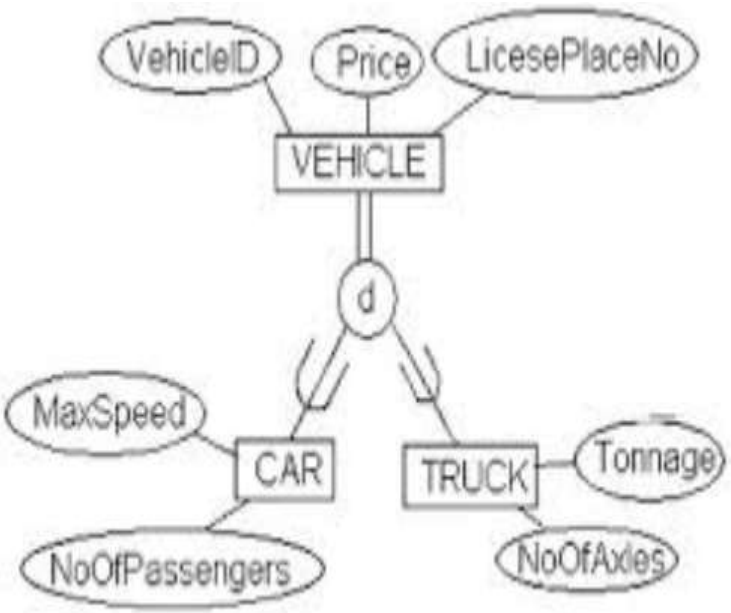
• For instance, let us assume that, after 3 months of employment, bank employees are assigned to one of four work teams.

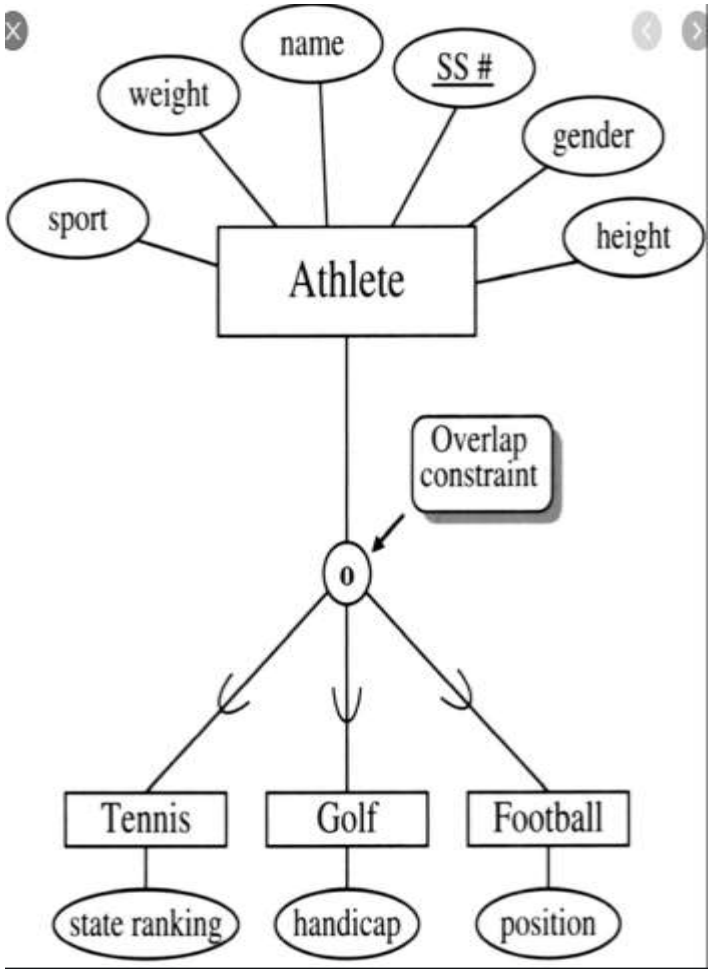
We therefore represent the teams as four lower-level entity sets of the higher-level *employee* entity set. A given employee is not assigned to a specific team entity automatically on the basis of an explicit defining condition.

Instead, the **user in charge of this decision makes the team assignment** on an individual basis.


Fname
Minit
LName
Name
Ssn
BirthDate
Address
JobType
EMPLOYEE
d
TypingSpeed
TGrade
EngType
SECRETARY
TECHNICIAN
ENGINEER

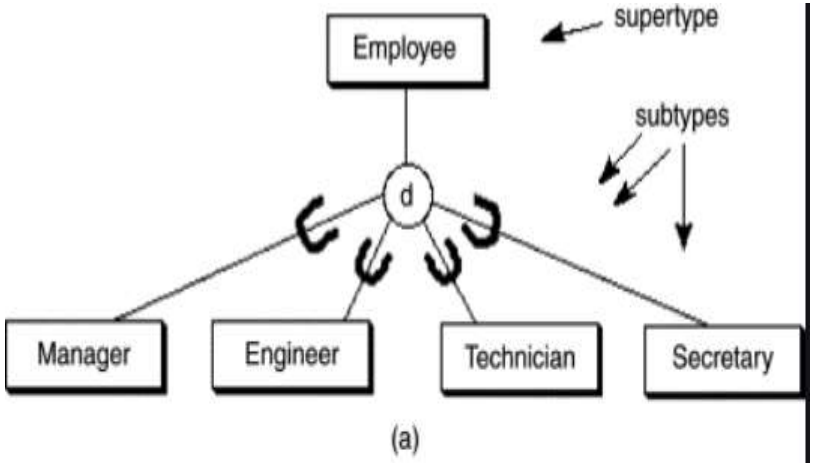
Contd…


VehicleID
Price
LicesePlaceNo
VEHICLE
d
MaxSpeed
CAR
TRUCK
Tonnage
NoOfPassengers
NoOfAxles


name
weight
SS #
gender
sport
height
Athlete
Overlap constraint
o
Tennis
Golf
Football
state ranking
handicap
position

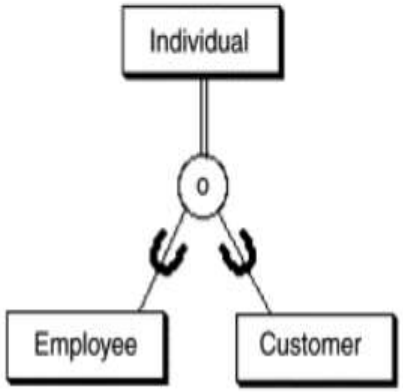
Contd…

- Fig(a): Disjoint



(a)

- Fig(b):Overlapping



Contd…

- **Constraints on Generalization**
- **Completeness constraint** on a generalization or specialization, specifies whether or not an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within the generalization/specialization. This constraint may be one of the following:

- **Total generalization** or **specialization**. Each higher-level entity must belong to a lower-level entity set. The *account generalization is total: All account entities must be either a savings* account or a checking account.

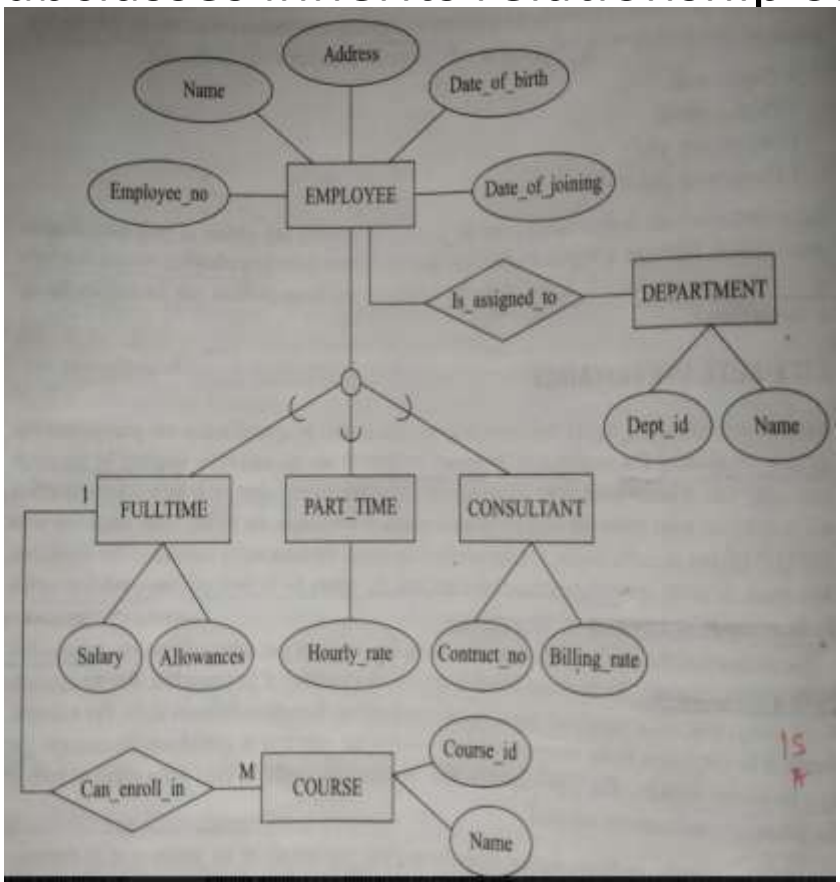
- **Partial generalization** or **specialization**. Some higher-level entities may not belong to any lower-level entity set. The work team entity sets illustrate a partial specialization. Since employees are assigned to a team only after 3 months on the job, some *employee entities may not be members of any of the lower-level team entity sets.*

Four possible constraints on Generalization/Specialization

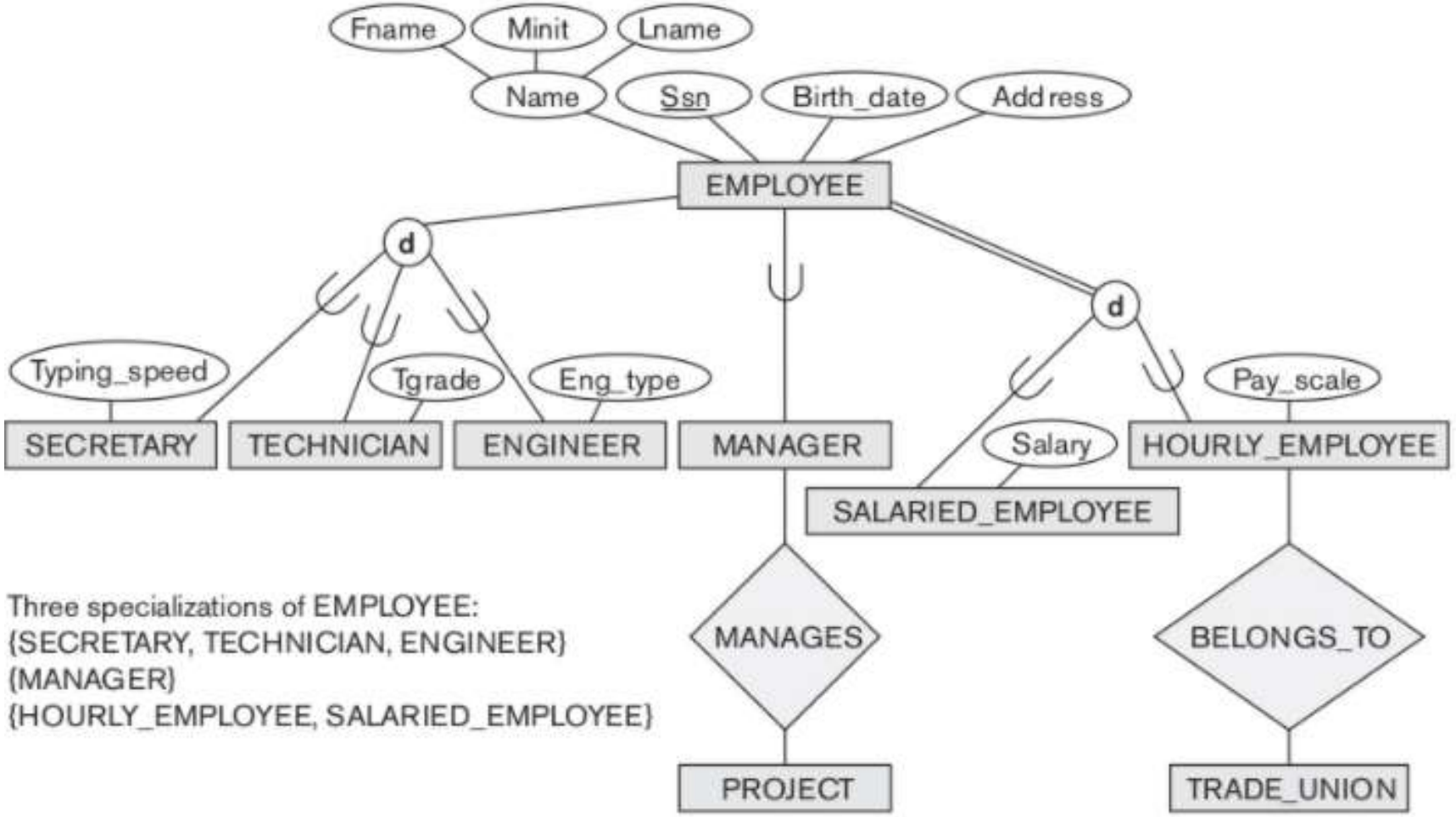
- Disjoint, Total
- Disjoint, Partial
- Overlapping ,Total
- Overlapping ,Partial

Employee Superclass/Subclass relationship

- The attributes of the higher level entity sets are said to be inherited by lower level entity set
- The Subclass also inherits participation in the relationship sets in which superclass participates- (Subclasses inherits relationship of superclass)



EER diagram notation to represent subclasses and specialization



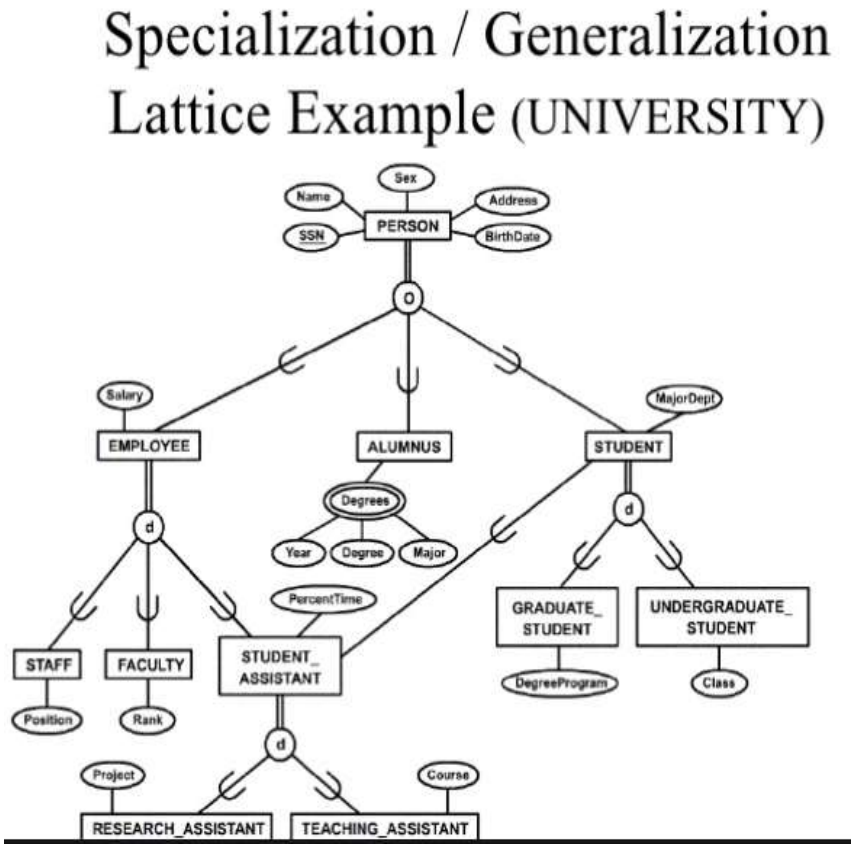
Specialization/Generalization hierarchy and Lattices

- A specialization hierarchy has the constraint that every subclass has only one parent, which result in tree structure
- A specialization lattice can be a subclass in more than one class/subclass has more than one level relationship
- Eg: Student_Assistant
- A Subclass with more than one

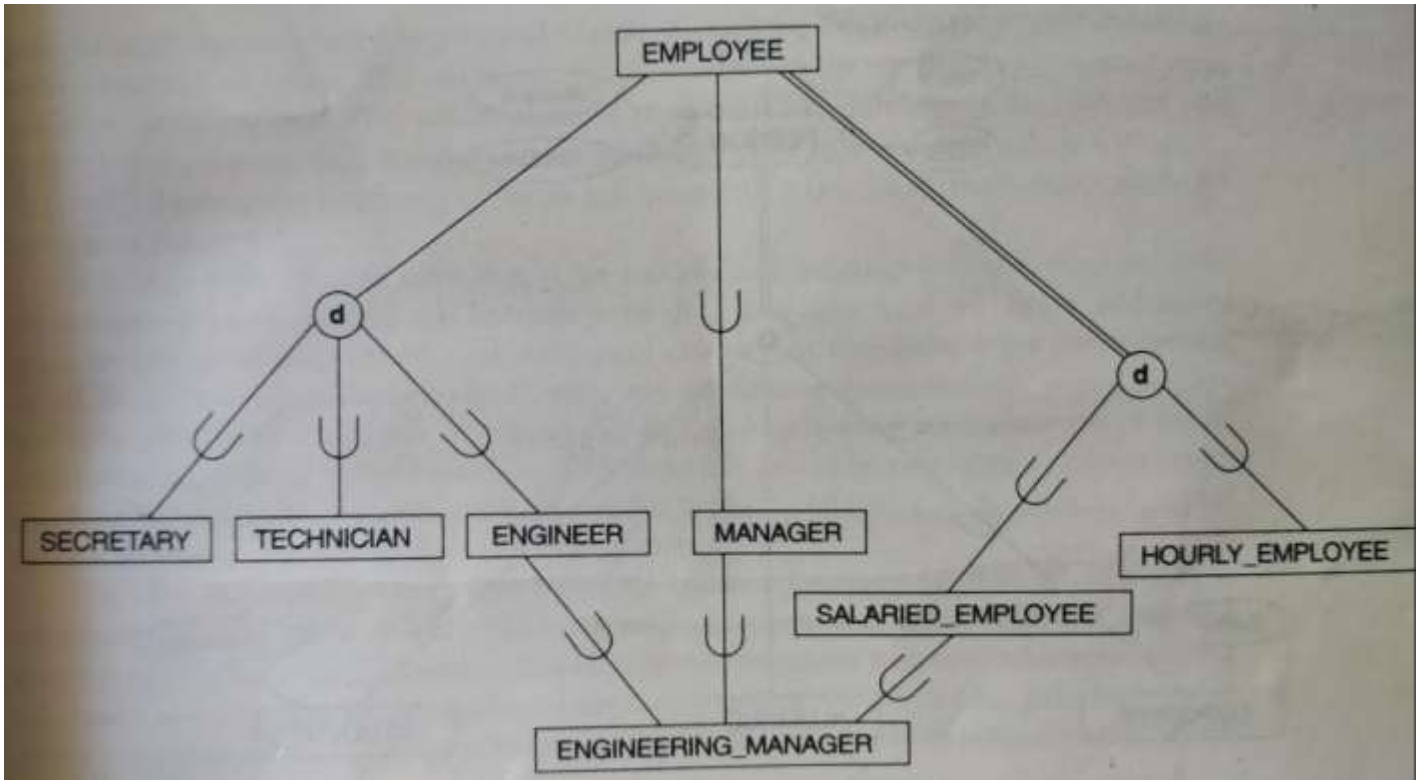
Superclass is called Shared subclass

- This leads to the concept of Multiple

Inheritance- inherits all the attributes and relationships of multiple classes

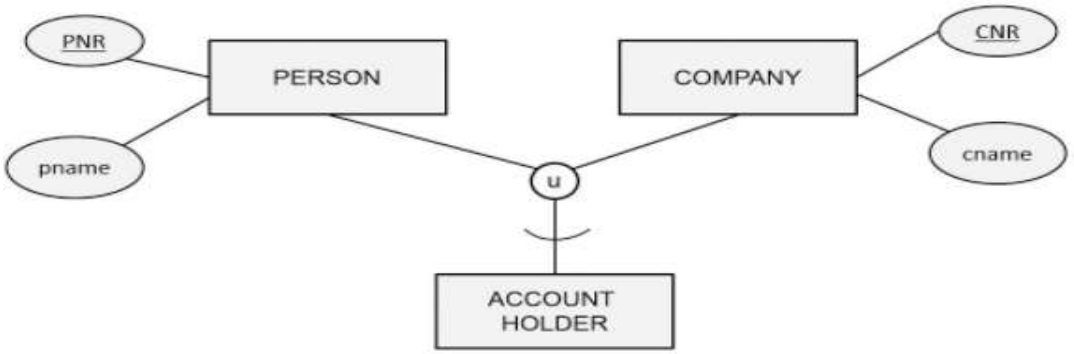


Contd…


EMPLOYEE
d
d
SECRETARY
TECHNICIAN
ENGINEER
MANAGER
HOURLY_EMPLOYEE
SALARIED_EMPLOYEE
ENGINEERING_MANAGER

Categorization or Union

- The modeling of a single subclass with a relationship that involves more than one distinct superclass is called Categorization
- A subclass having more than one superclass- is called a category and a process of defining a category is called Categorization
- A subclass will represent a collection of objects that is a subset of the UNION of distinct entity types



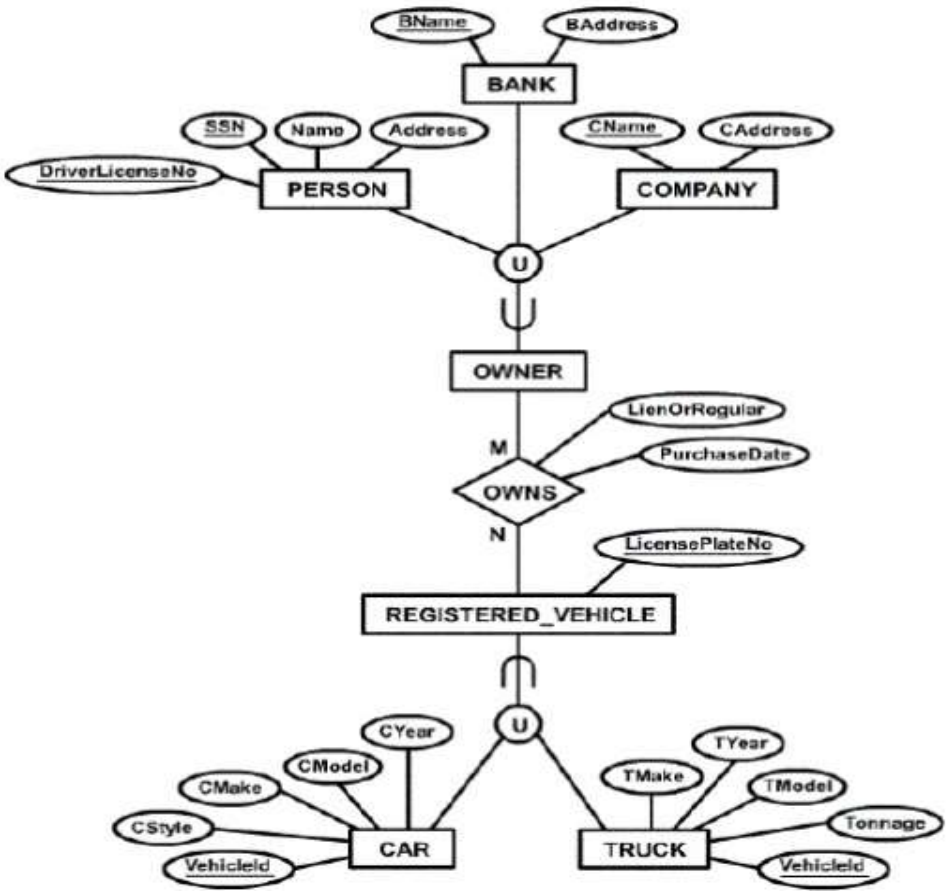
Contd…

- The line connecting the category subclass to the categorization circle has the subset symbol(c) and the circle itself contains the union symbol(U)
- The category subclass has selective inheritance. The category will inherit only the attributes of one of the superclass at a time.
- A category can be subdivided depending on total or partial participation

Example of categorization Union type

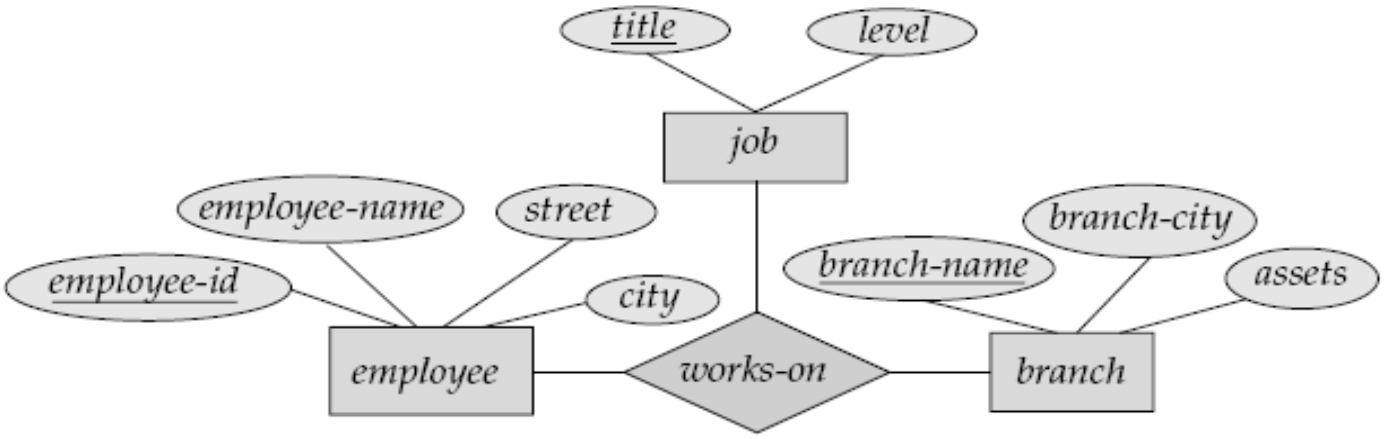
- A Superclass- Company,Person, bank are connected to circle with U which stands for set union operation
- Eg: A Category owner is a subclass of union of superclasses -three entity sets: Company, bank, person
- Owner must exist in only one

of the superclass



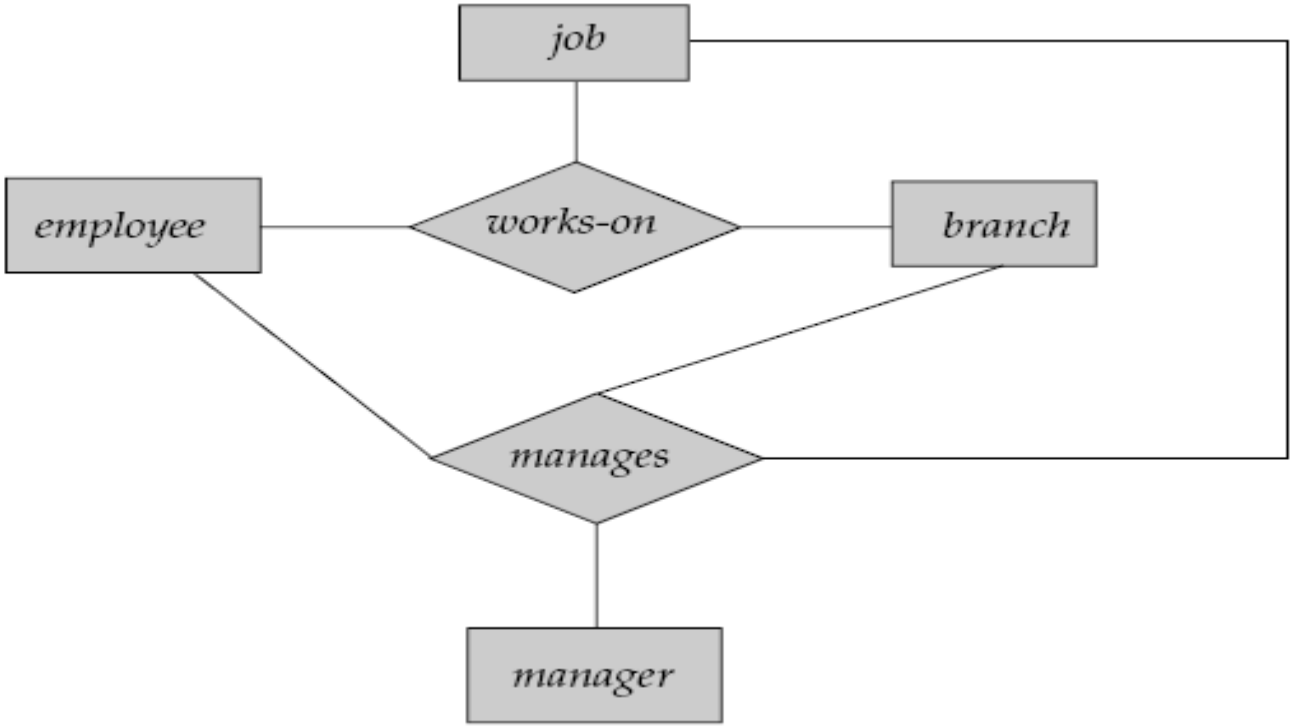
Aggregation

- One limitation of the E-R model is that it cannot express relationships among relationships
- Aggregation is an abstraction through which relationships are treated as higher-level entities
- Consider the ternary relationship *works-on*, between a *employee*, *branch*, and *job*



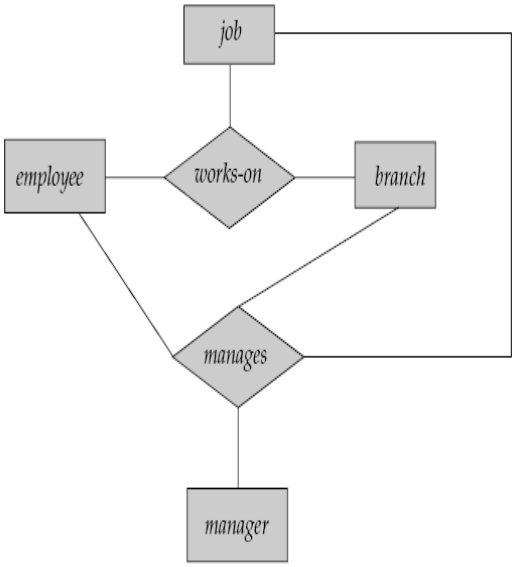
Contd…

- Suppose we want to record managers for (*employee*, *branch*, *job*) combinations. Let us assume that there is an entity set *manager*.
- One alternative for representing this relationship is to create a quaternary relationship *manages* between *employee*, *branch*, *job*, and *manager*.
- Using the basic E-R modeling constructs, we obtain the E-R diagram



Contd…

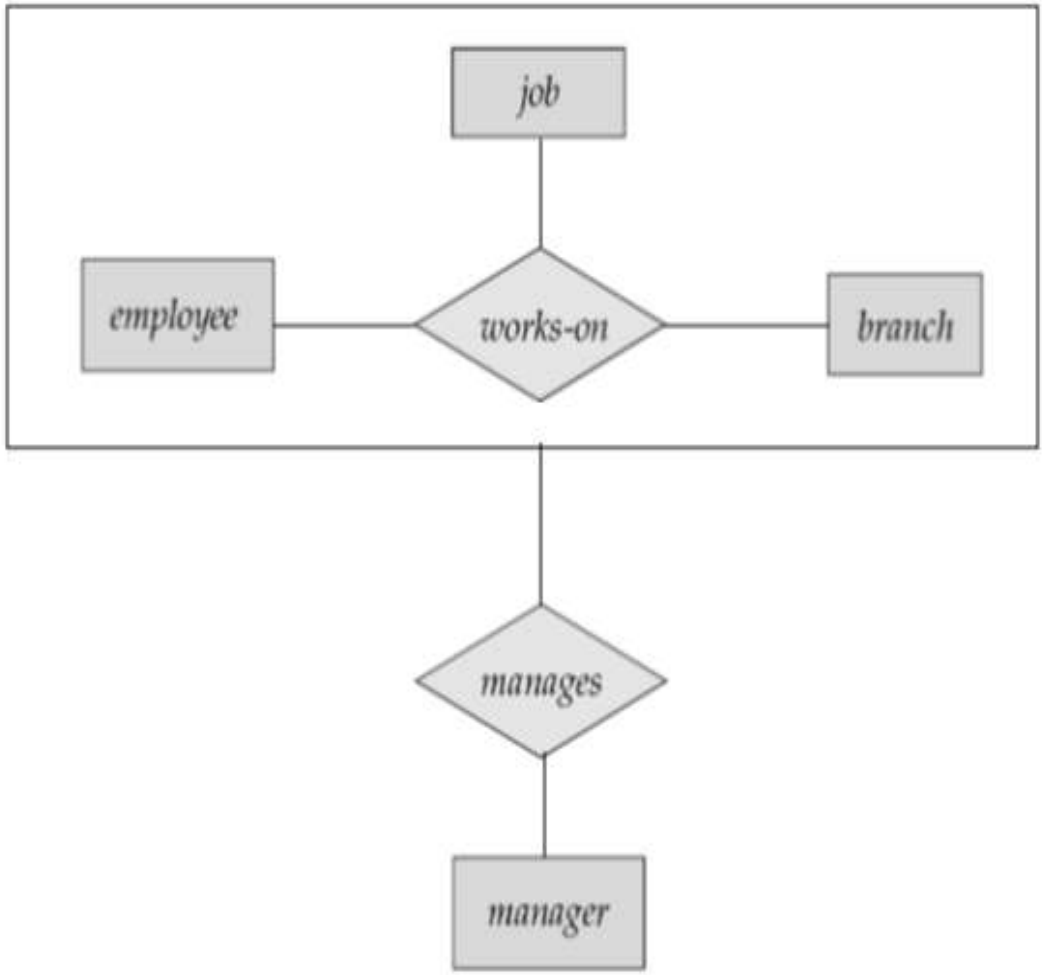
- Relationship sets *works_on* and *manages* represent overlapping information
  - Every *manages* relationship corresponds to a *works_on* relationship
  - However, some *works_on* relationships may not correspond to any *manages* relationships
    - So we can't discard the *works_on* relationship



- Eliminate this redundancy via *aggregation*
  - Treat relationship as an abstract entity
  - Allows relationships between relationships
  - Abstraction of relationship into new entity

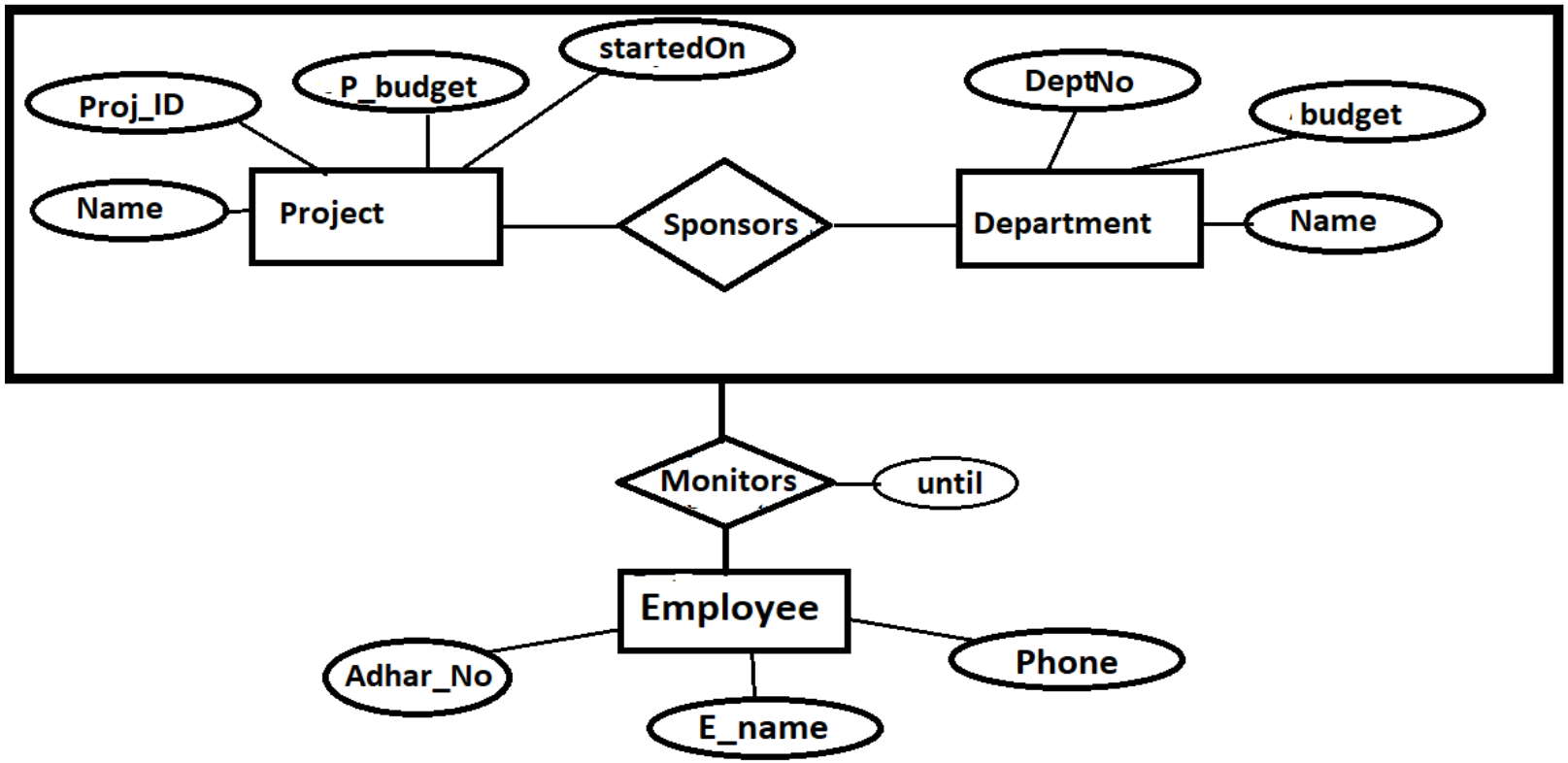
- Without introducing redundancy, the following diagram represents:
  - An employee works on a particular job at a particular branch
  - An employee, branch, job combination may have an associated manager

ıtd...



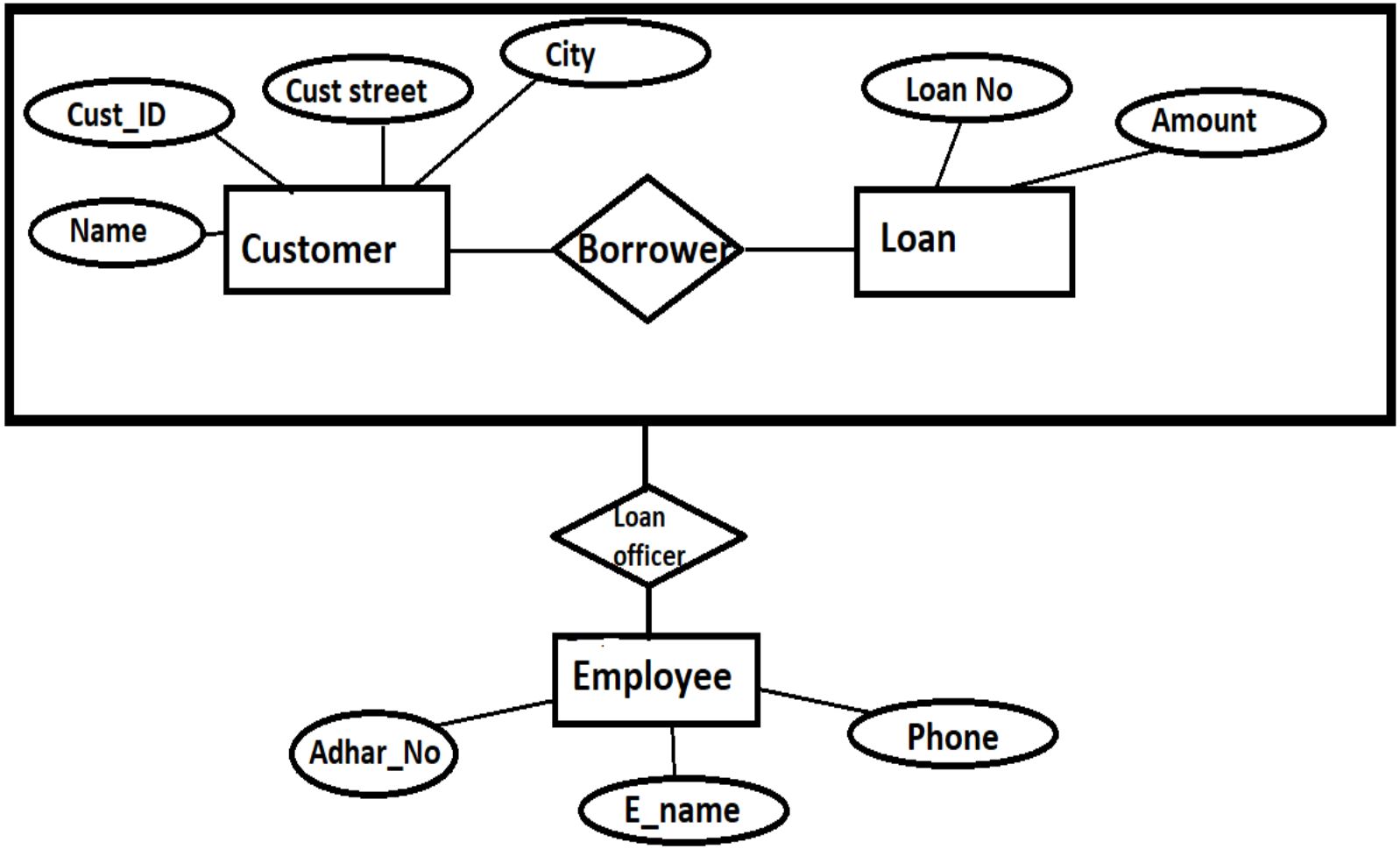
Example 2:

- Aggregation is an abstraction, relationships are treated as higher-level entities



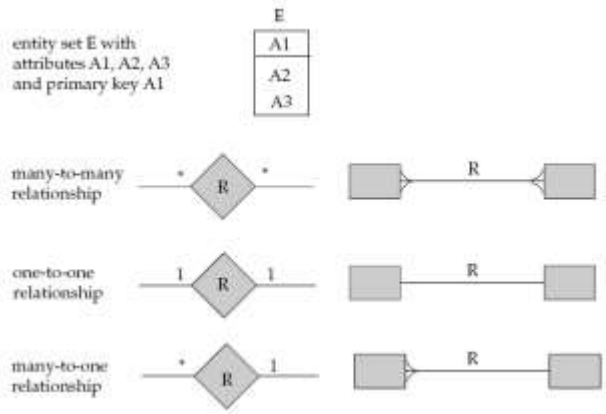
Contd…

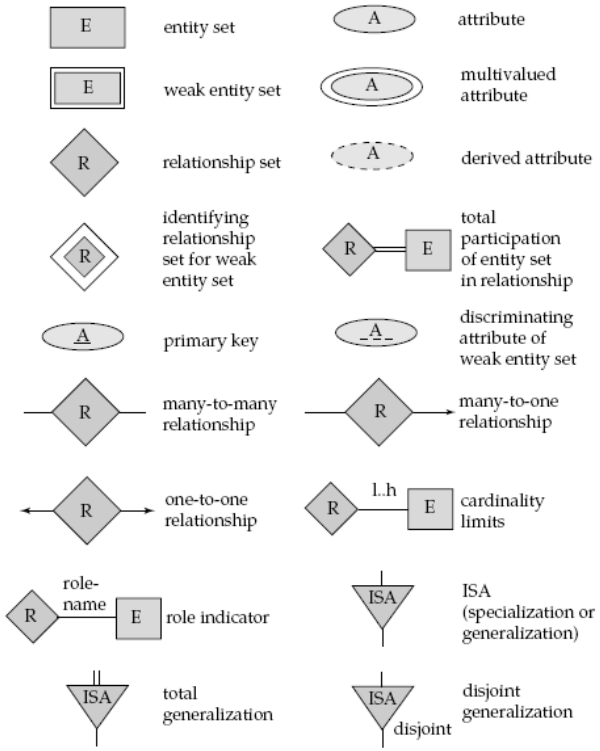
- **Example 3:**



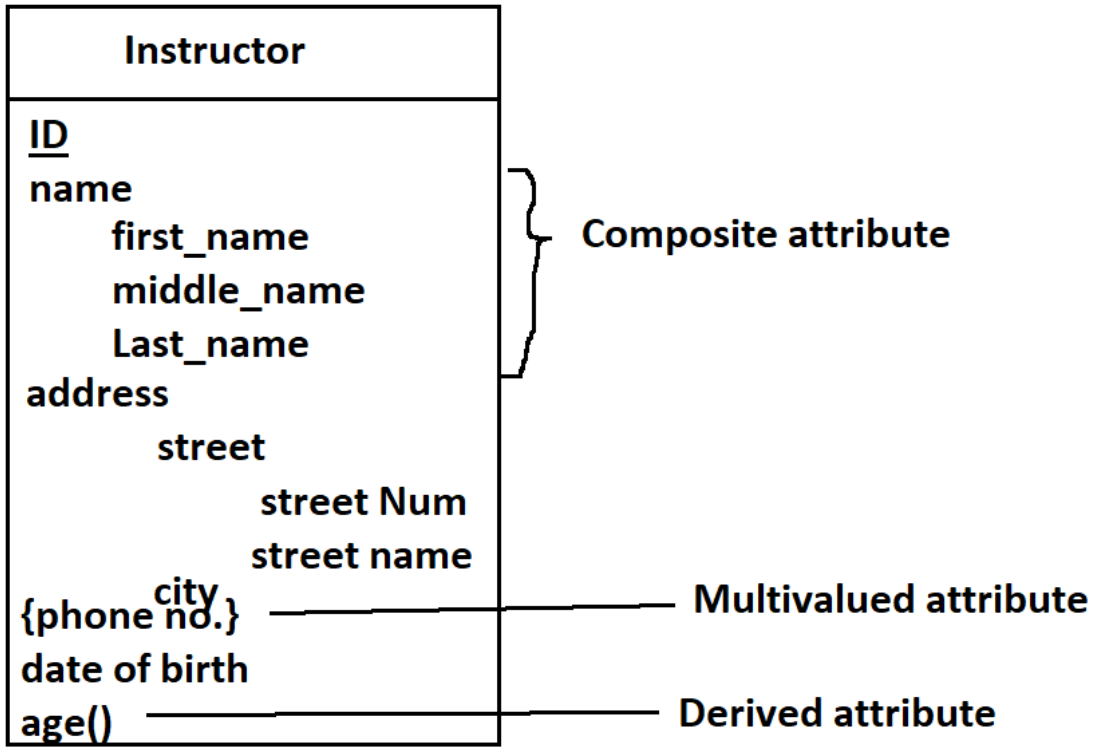
Extended E-R Features

- **Alternative E-R Notations**

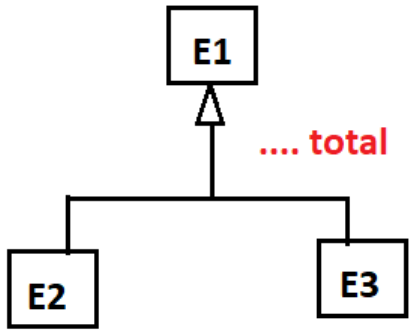




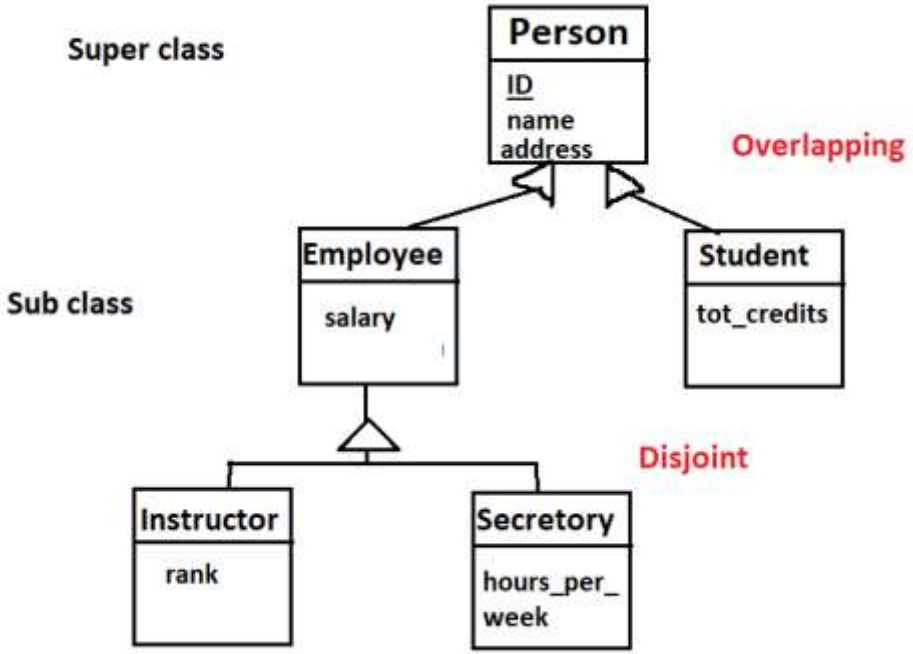
One more way for Entity set representation in EER diagram



One more way for Disjoint and overlapping in Specialization and Generalization



Total Specialization and Generalization



Database Design for Banking Enterprise

- The bank is organized into branches. Each branch is located in particular city and is identified by a unique name. The bank monitors the assets of each branch.

- Bank customers are identified by their customer_id values. The bank store each customer's name and street and city where the customer lives. Customers may have account and can take out the loans. A customer may be associated with a particular banker who may act as a loan officer or personal banker for that customer.

- Bank employees are identified by their employee_id values. The bank administration stores the name and the telephone number of each employee, the name of employee's dependents and employee_id number of employee's manager. The bank also keeps track of the employees' start date and thus the length of employment.
- Bank offers two types of account- saving and checking account. Accounts can be held by more than one customer and a customer can have more than one account. Each account is assigned unique account number. The bank maintains a record of each account' balance and most recent date on which the account was accessed each customer holding the account. Each saving account has an interest rate and overdrafts are recorded for each checking account.

- The bank provides its customer with loans. A loan is originates at particular branch and can be held by one or more customers. A loan is identified by a unique loan number. For each loan, bank keeps track of the loan amount and the loan payment. Although a loan payment number does not uniquely identify a particular payment among those for all bank's loan, a payment number does identify a particular payment for a specific loan. The date and amount are recorded for each payment.

Contd…

- **Entity Sets for the Bank Database**
- The *branch* entity set, with attributes *branch-name*, *branch-city*, and *assets*.

- The *customer* entity set, with attributes *customer-id*, *customer-name*, *customer-street*; and *customer-city*. A possible additional attribute is *banker-name*.

- The *employee* entity set, with attributes *employee-id*, *employee-name*, *telephone-number*,*salary*, and *manager*. Additional descriptive features are the multivalued attribute *dependent-name*, the base attribute *start-date*, and the derived attribute *employment-length*.
- Two account entity sets—*savings-account* and *checking-account*—with the common attributes of *account-number* and *balance*; in addition, *savings-account* has the attribute *interest-rate* and *checking-account* has the attribute *overdraft-amount*.

- The *loan* entity set, with the attributes *loan-number*, *amount*, and *originating-branch*.
- The weak entity set *loan-payment*, with attributes *payment-number*, *payment-date*, and *payment-amount*.

Contd…

- **Relationship Sets Designation**
- *borrower*, a many-to-many relationship set between *customer* and *loan*.
- *loan-branch*, a many-to-one relationship set that indicates in which branch loan originated.

Note that this relationship set replaces the attribute *originating-branch* of the entity set *loan*.

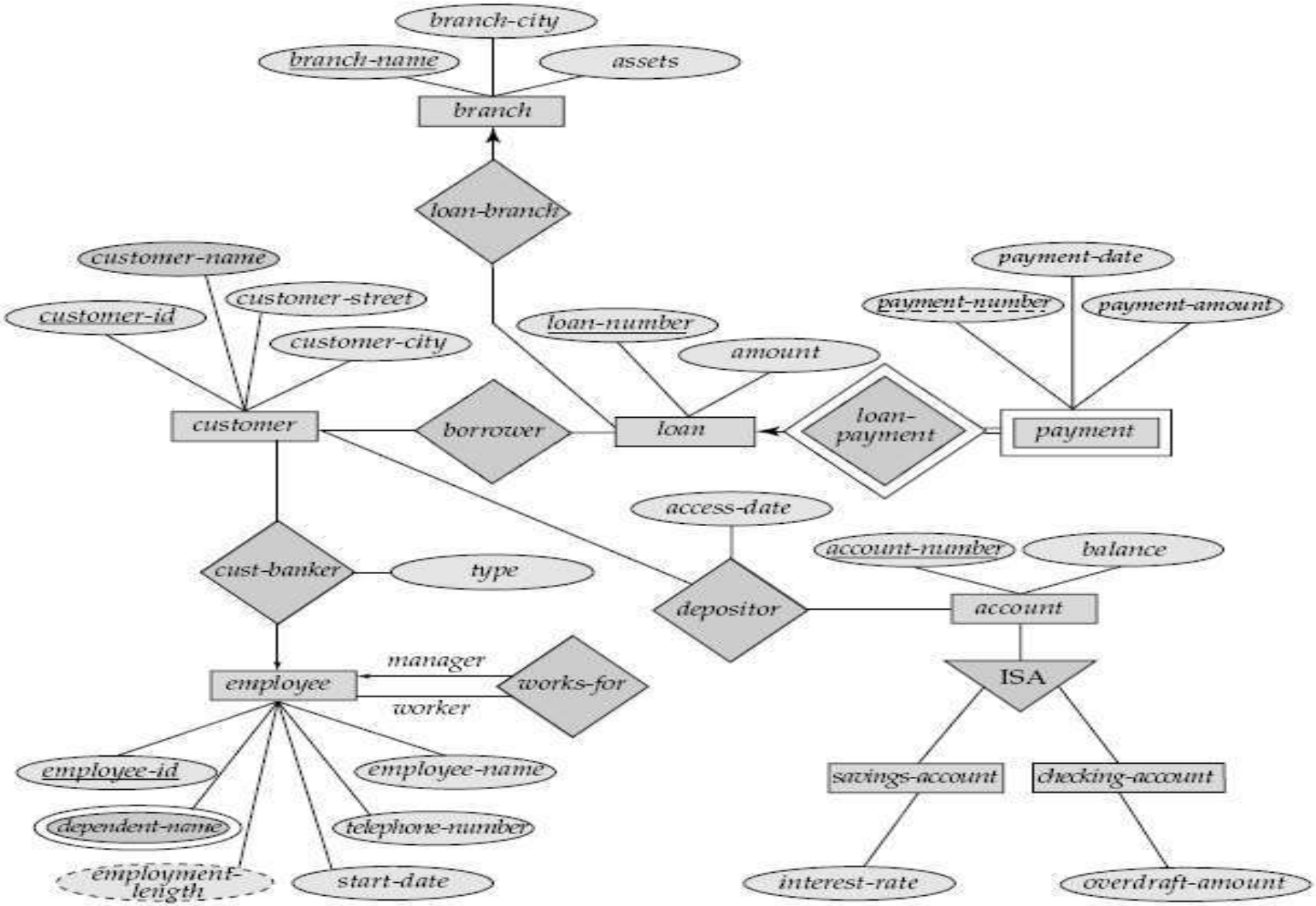
- *loan-payment*, a one-to-many relationship from *loan* to *payment*, which documents that a payment is made on a loan.
- *depositor*, with relationship attribute *access-date*, a many-to-many relationship set between *customer* and *account*, indicating that a customer owns an account.

Contd…

- *cust-banker*, with relationship attribute *type*, a many-to-one relationship set expressing that a customer can be advised by a bank employee, and that a bank employee can advise one or more customers. Note that this relationship set has replaced the attribute *banker-name* of the entity set *customer*.

- *works-for*, a relationship set between *employee* entities with role indicators *manager* and *worker*; the mapping cardinalities express that an employee works for only one manager and that a manager supervises one or more employees. Note that this relationship set has replaced the *manager* attribute of *employee*.

E-R Diagram: Database Design for Banking Enterprise



Summary of Symbols Used in E-R Notation