Logical Database Design - Mapping ERD to Relational

Dr. Enosha Hettiarachchi

Objectives

- How to remove features from a local conceptual model that are not compatible with the relational model.

- How to derive a set of relations from a logical data model (ER (EER) diagram).

Transforming ERD into Relations

Transforming (mapping) E-R diagrams to relations is a relatively straightforward process with a well-defined set of rule

Step 1: Map (Regular) Strong Entities
2: Map Weak Entities
3: Map Binary Relationships
4: Map Unary Relationships
5: Map Complex Relationships
6: Map Supertype/Subtype Relationships
7: Map Categories

Transforming ERD into Relations

- Remove features not compatible with the relational model (optional step)
- To refine the local conceptual data model to remove features that are not compatible with the relational model. This involves:
  - **remove M:N binary relationship types;**
  - **remove M:N recursive relationship types;**
  - **remove complex relationship types;**
  - **remove multi-valued attributes.**

Map Regular (Strong) Entity Types

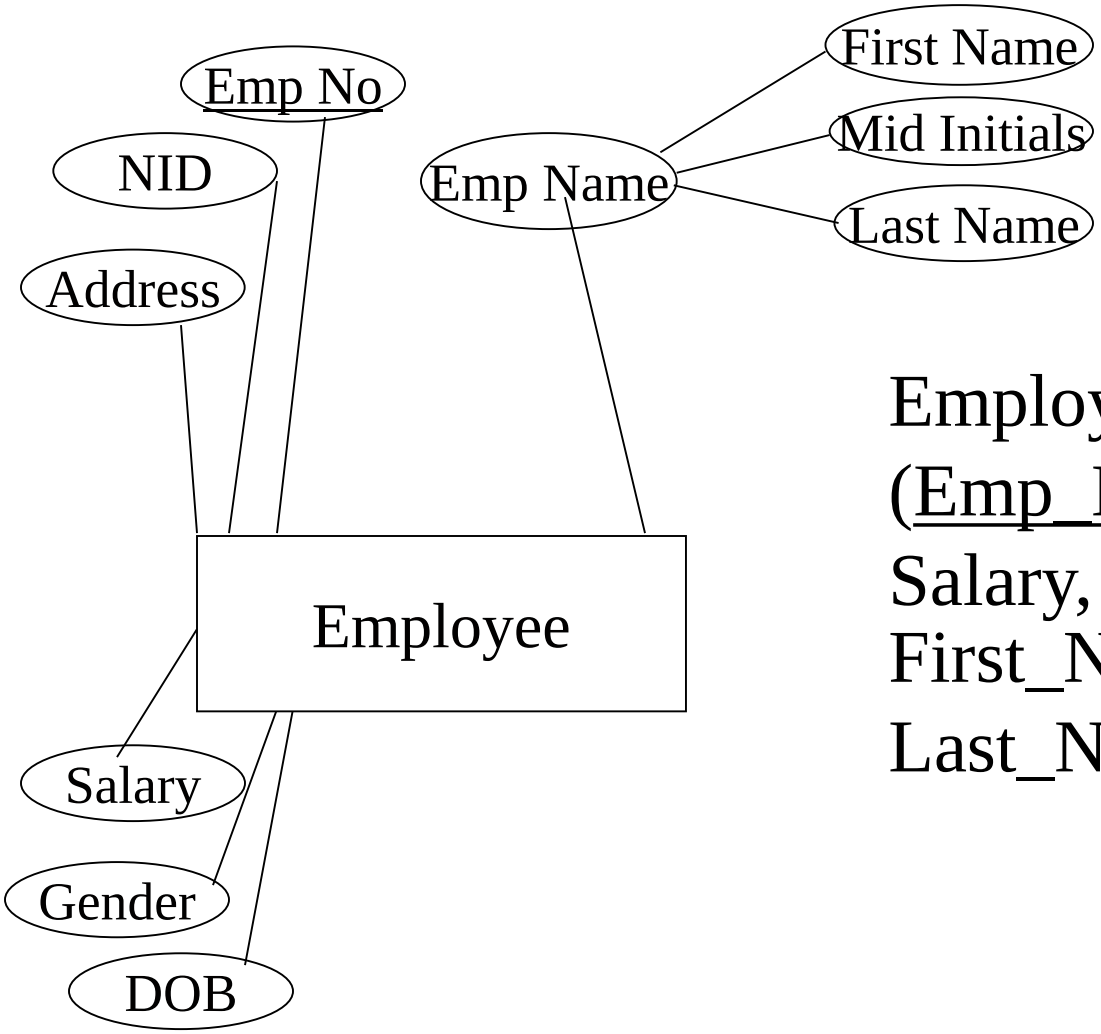
Strong entity types

Each regular entity type in an ERD is transformed into a relation.

Create a relation that includes all simple attributes of that entity. For composite attributes, include only constituent simple attributes.

Entity	Relation
entity name	relation name
simple attribute	attribute of the relation
entity identifier	primary key of relation
composite attribute	component attributes
multi-valued attribute	new relation with PK

Map Strong Entity Types



Employee
(Emp_No, NID, Address,
Salary, Gender, DOB,
First_Name, Mid_Initials,
Last_Name)

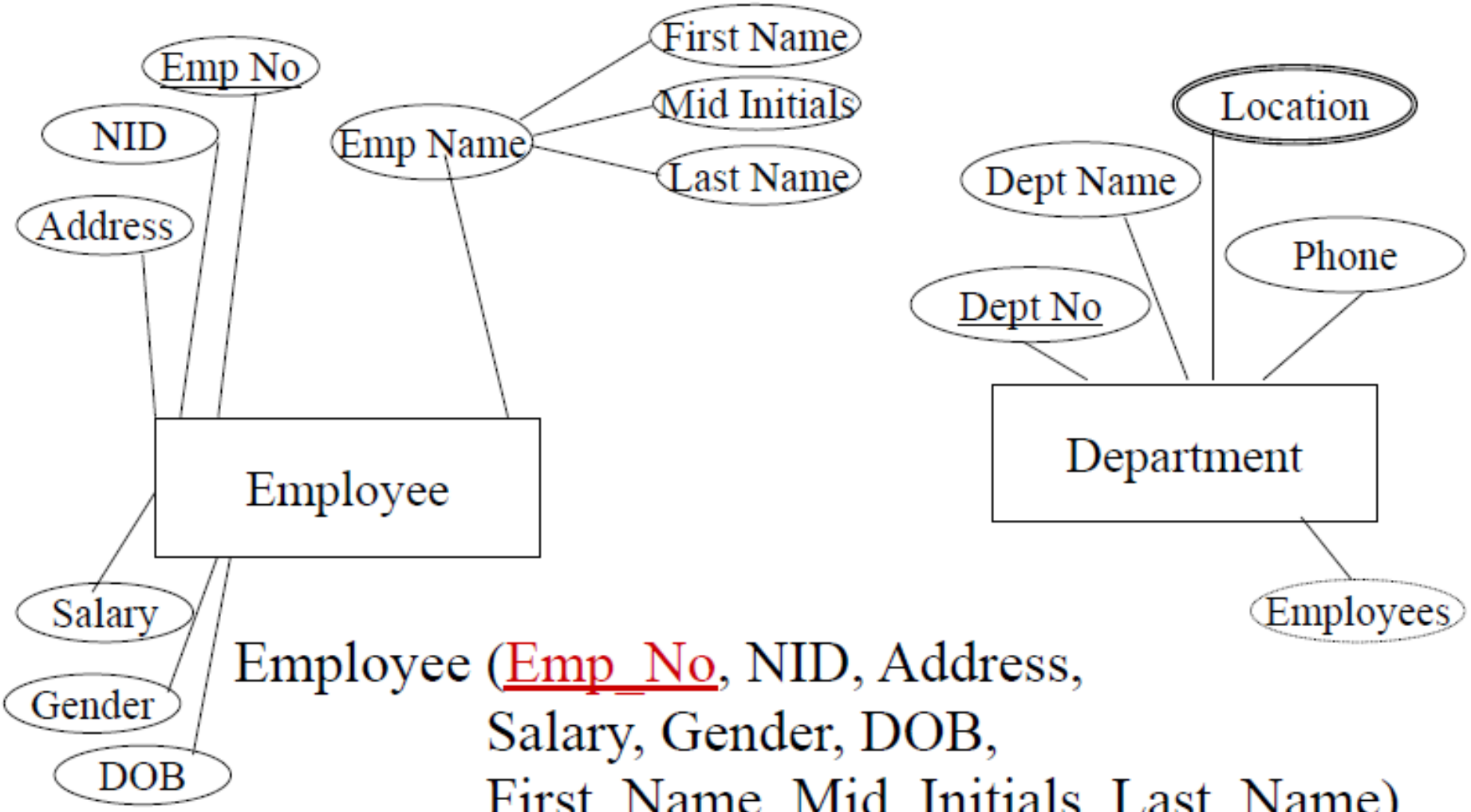
Remove multi-valued attributes

Multi-valued attributes

- Create new relation to represent multi-valued attribute and include primary key of entity in new relation, to act as a foreign key.

- Unless the multi-valued attribute is itself an alternate key of the entity, **primary key of new relation is combination of the multi-valued attribute and the primary key of the entity.**

Dept_Location(Dept_No, Location)



Employee (Emp_No, NID, Address,
Salary, Gender, DOB,
First_Name, Mid_Initials, Last_Name)

Department(Dept_No, Dept_Name, Phone)

FK/NN NN

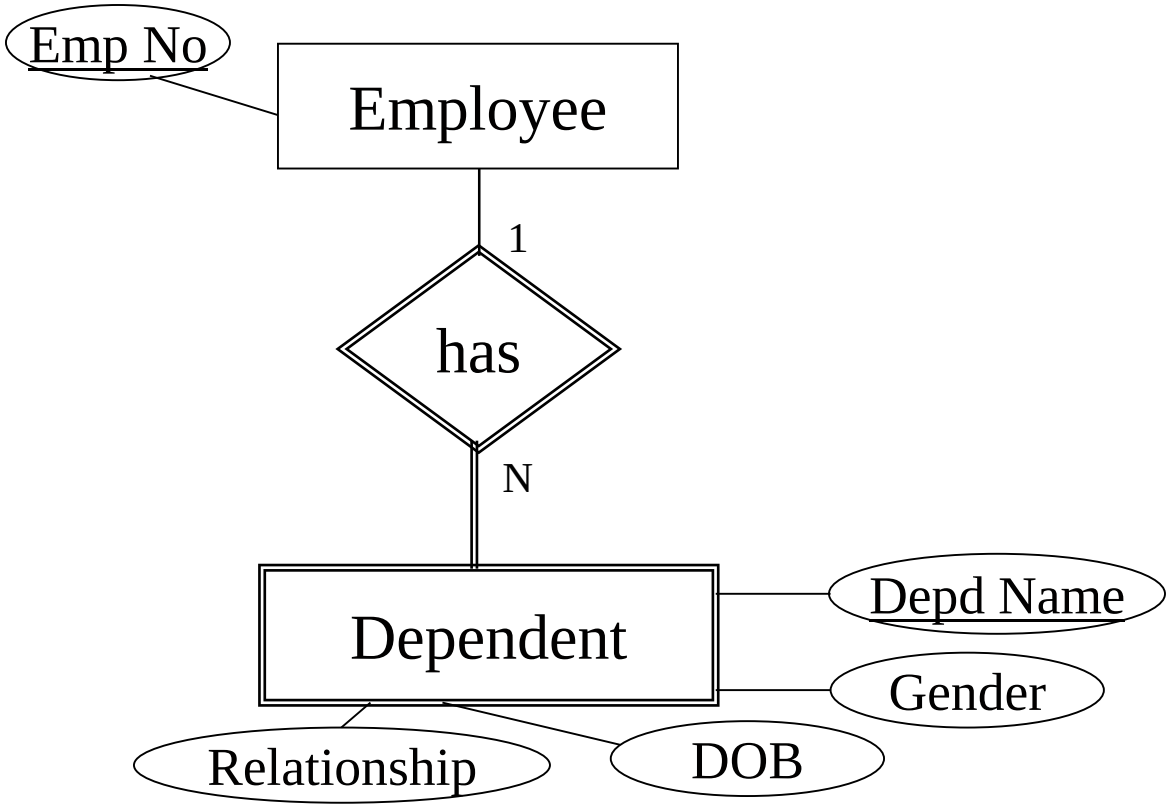
Dept_Location(*Dept_No*, Location)

Map Weak Entity Types

Weak entity types

- Create a relation that includes all simple attributes of that entity.
- Primary key is partially or fully derived from each owner entity.

Map Weak Entity Types



Dependent(*Emp_No*, Depd_Name, Gender, DOB, Relationship)

Employee (Emp_No, ….)

Map Binary Relationships

Procedure depends on both the degree of the binary relationships and the cardinalities of the relationships

- Map Binary One-to-Many (1:N) Relationships
- Map Binary Many-to-Many (M:N) Relationships
- Map Binary One-to-One (1:1) Relationships

1:N binary relationship types

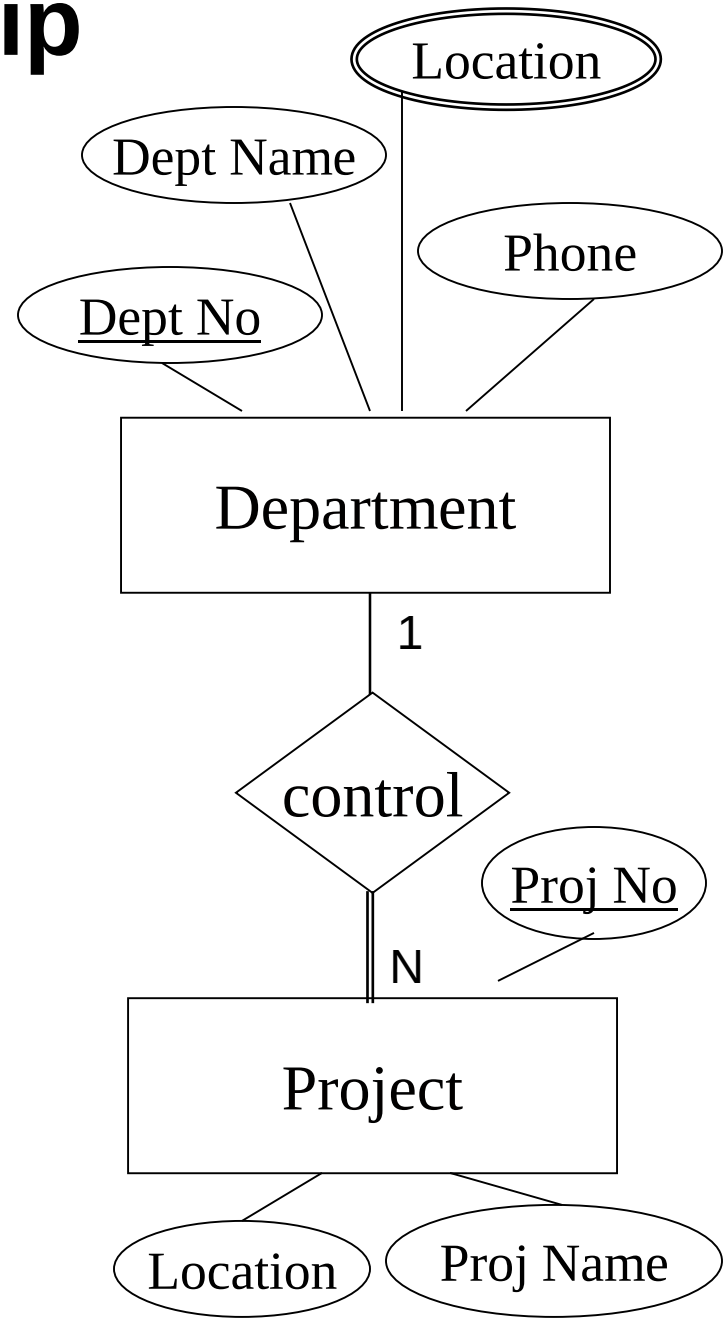
- Entity on ‘one side’ is designated the parent entity and entity on ‘many side’ is the child entity.
- Paste a copy of the primary key attribute(s) of parent entity into relation representing child entity, to act as a foreign key.
- Include any attributes of the relationship to the relation of the many side

1:N binary relationship types

FK

Department(Dept_No,
Dept_Name, Phone)

Project(Proj_ No, Proj_Name,
Location, ***Dept_No***)



1:1 binary relationship types

- More complex as cardinality cannot be used to identify parent and child entities in a relationship.
- Instead, participation used to decide whether to combine entities into one relation or to create two relations and paste a copy of primary key from one relation to the other. Consider following:

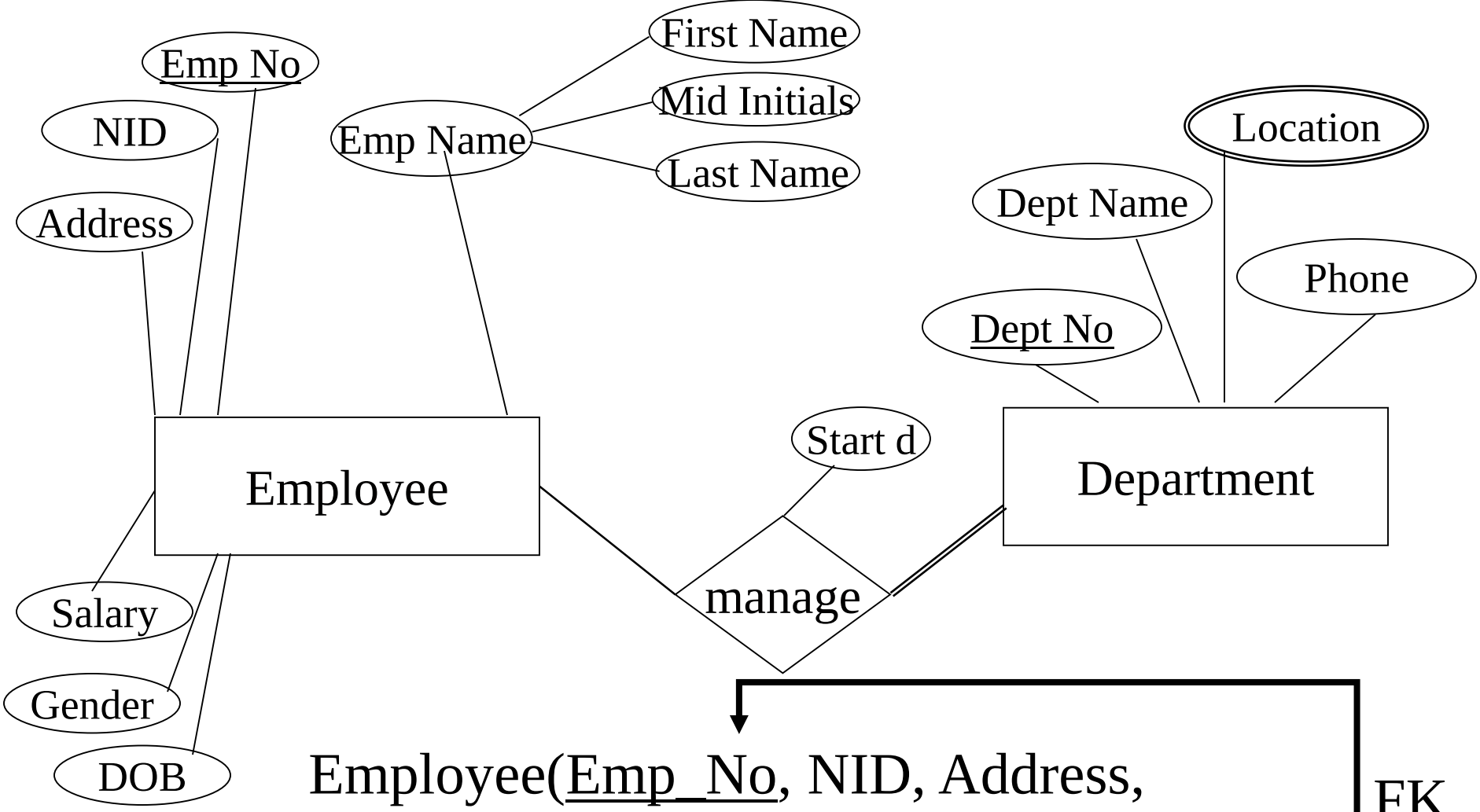
- *(a) total* participation on *both* sides of 1:1 relationship;
- *(b) total* participation on *one* side of 1:1 relationship;
- *(c) partial* participation on *both* sides of 1:1 relationship.

Total participation on both sides of 1:1 relationship

Combine entities involved into one relation and choose one of the primary keys of original entities to be primary key of new relation, while other (if one exists) is used as an alternate key.

Total participation on *one* side of a 1:1 relationship

- Identify parent and child entities using participation constraints.
- **Entity with partial participation is designated parent entity,** and other entity designated child entity.
- Copy of primary key of parent placed in relation representing child entity.
- If relationship has one or more attributes, these attributes should also be included in the child relation.



Employee(Emp_No, NID, Address, Salary, Gender, DOB, First_Name, Mid_Initials, Last_Name)

Department(Dept_No, Dept_Name, Phone, *Manager*, Start_D)

FK

Partial participation on *both* sides of a 1:1 relationship

- Designation of the parent and child entities is arbitrary unless can find out more about the relationship.
- Consider employee Uses Car is a 1:1 relationship with partial participation on both sides. Assume majority of cars, but not all, are used by employees.

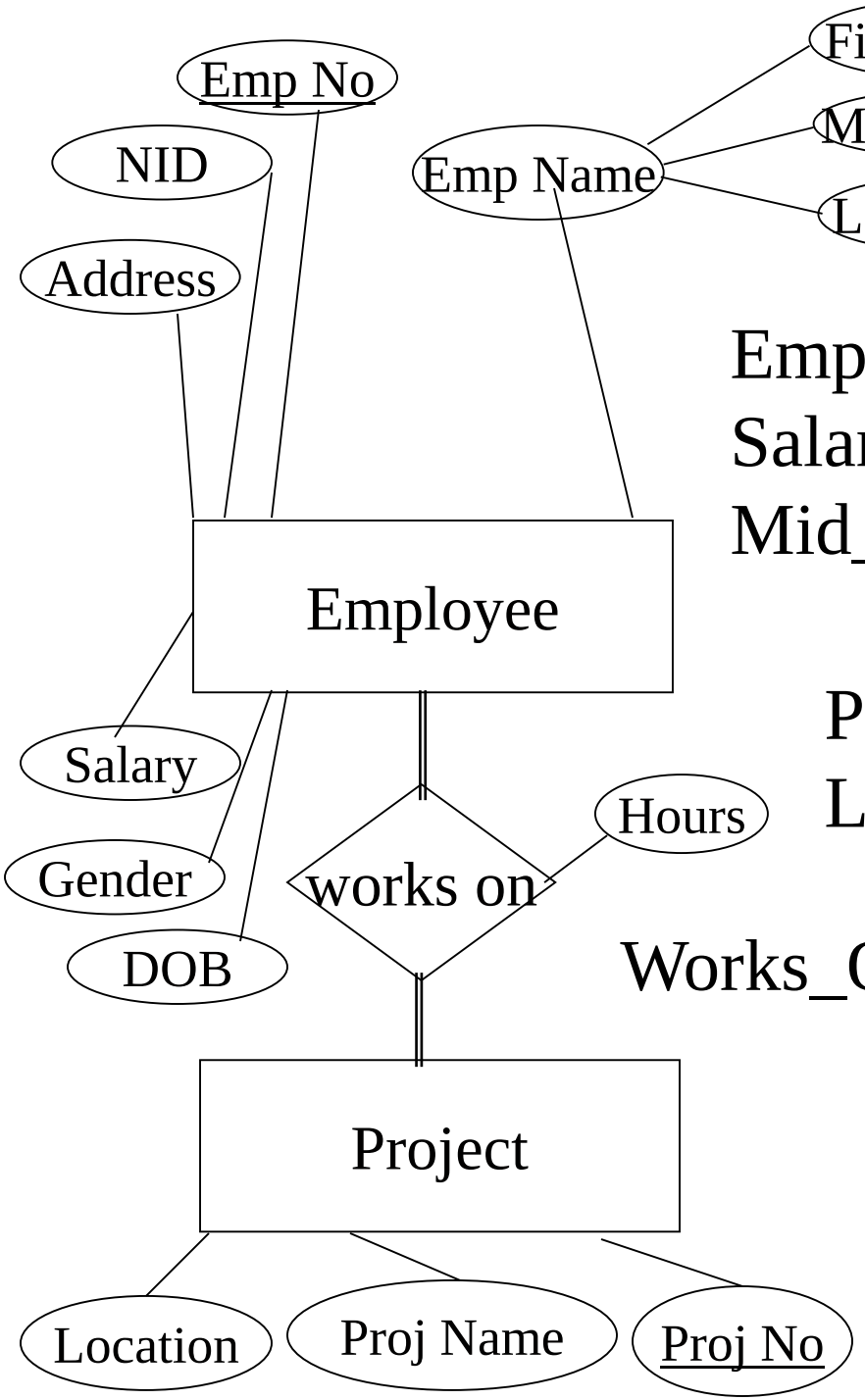
Partial participation on *both* sides of a 1:1 relationship

- Car entity, although partial, is closer to being total than Employee entity. Therefore designate Employee as parent entity and Car as child entity.

Map M:N binary relationship types

Create relation to represent relationship and include any attributes that are part of relationship.

- Post a copy of the primary key attribute(s) of the entities that participate in relationship into new relation, to act as foreign keys.
- These foreign keys will also form primary key of new relation, possibly in combination with some of the attributes of the relationship.



FK/NN

Employee(Emp_No, NID, Address, Salary, Gender, DOB, First_Name, Mid_Initials, Last_Name, Dept_No)

Project(Proj_ No, Proj_Name, Location, Dept_No)

FK/NN

Works_On(*Emp_No, Proj_No* , Hours)

Map Unary Relationships

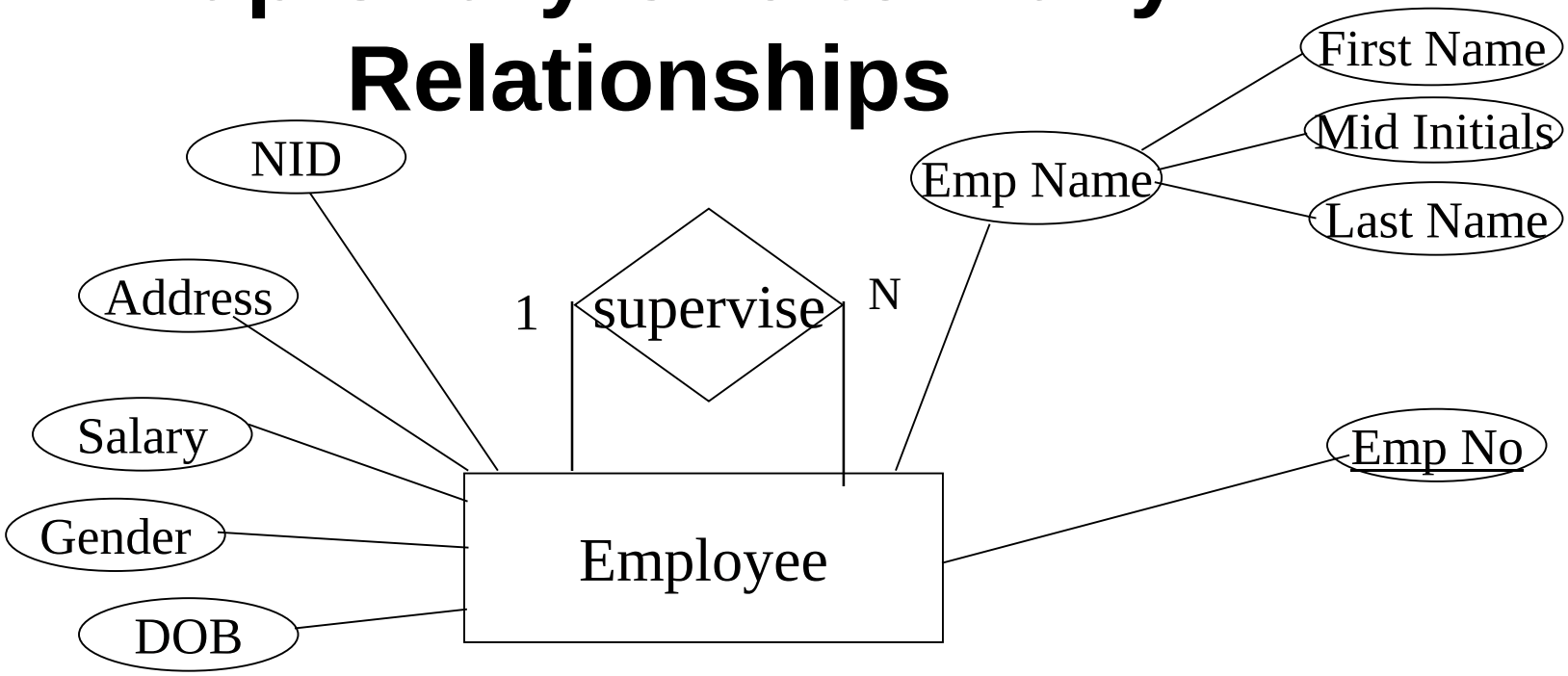
Procedure depends on both the degree of the binary relationships and the cardinalities of the relationships

- Map Unary One-to-Many Relationships
- Map Unary Many-to-Many Relationships
- Map Unary One-to-One Relationships

Map Unary One-to-Many Relationships

- Create a relation for the entity type
- Include PK of the entity as a foreign key within the same relation
- Include any attributes of the relationship to the relation of the many side

Map Unary One-to-Many Relationships



Employee(Emp_No, NID, Address, Salary, Gender, DOB, First_Name, Mid_Initials, Last_Name, *Supervisor*)

FK/Null

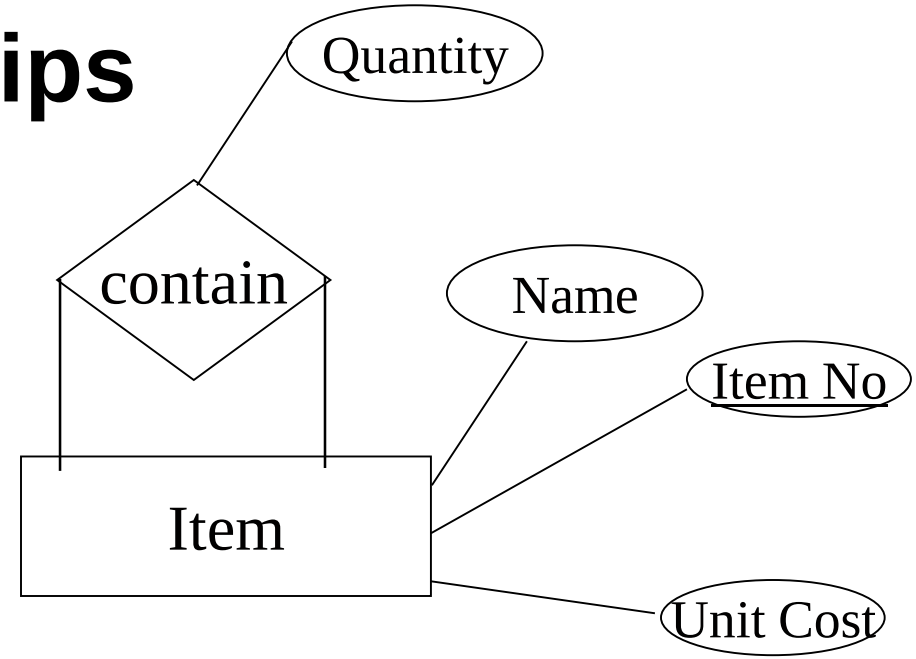
Map Unary Many-to-Many Relationships

- Create a relation for the entity type

- Create new relation and include PK of the entity type as FK twice. These attributes become the PK (composite)

- Include any attributes of the relationship to the new relation

Map Unary Many-to-Many Relationships



Item(Item_No, Name, Unit_Cost)

FK/NN

Component(Item_No, Component_No, Quantity)

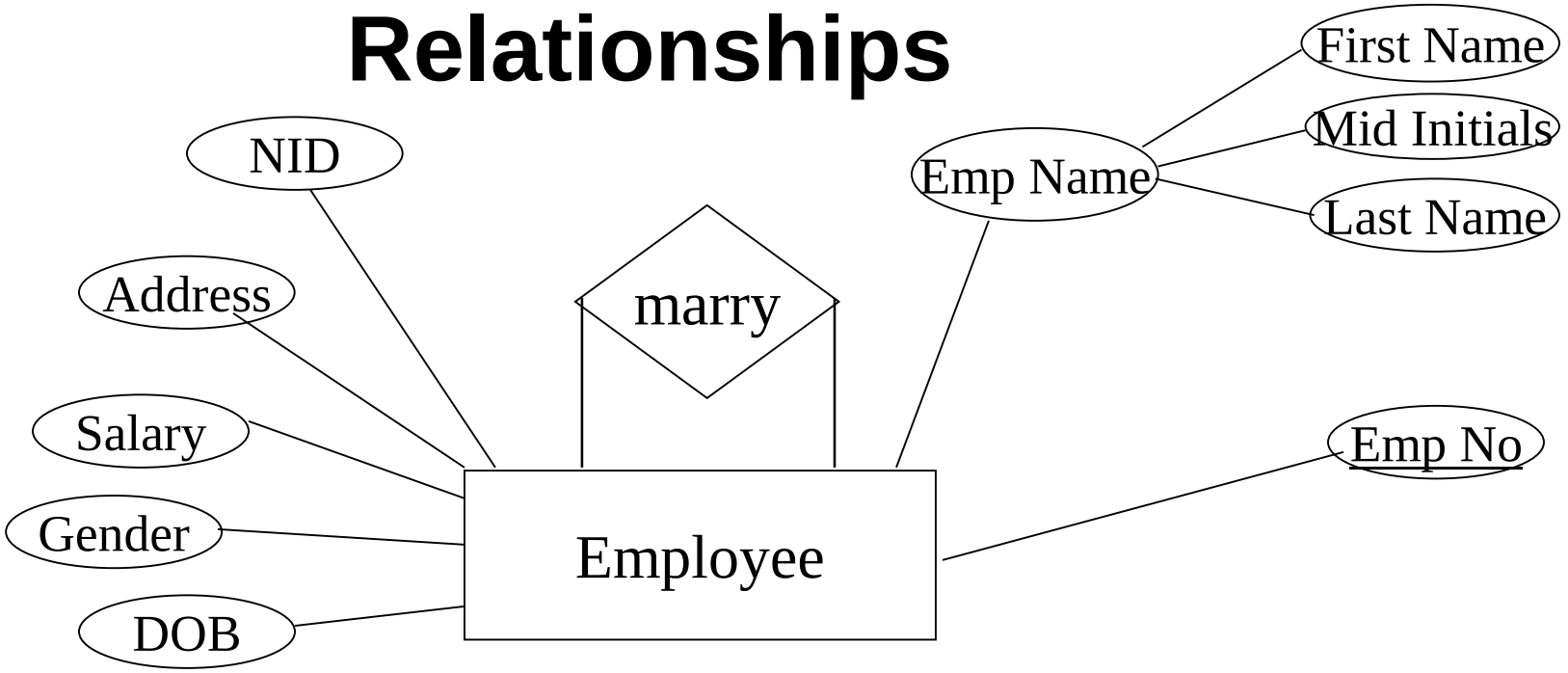
FK/NN

Map Unary One-to-One Relationships

Create a relation for the entity type.

Include PK of the entity as a foreign key within the same relation.

Map Unary One-to-One Relationships



Employee(Emp_No, NID, Address, Salary, Gender, DOB, First_Name, Mid_Initials, Last_Name, Dept_No, *Marry*)

FK/Null

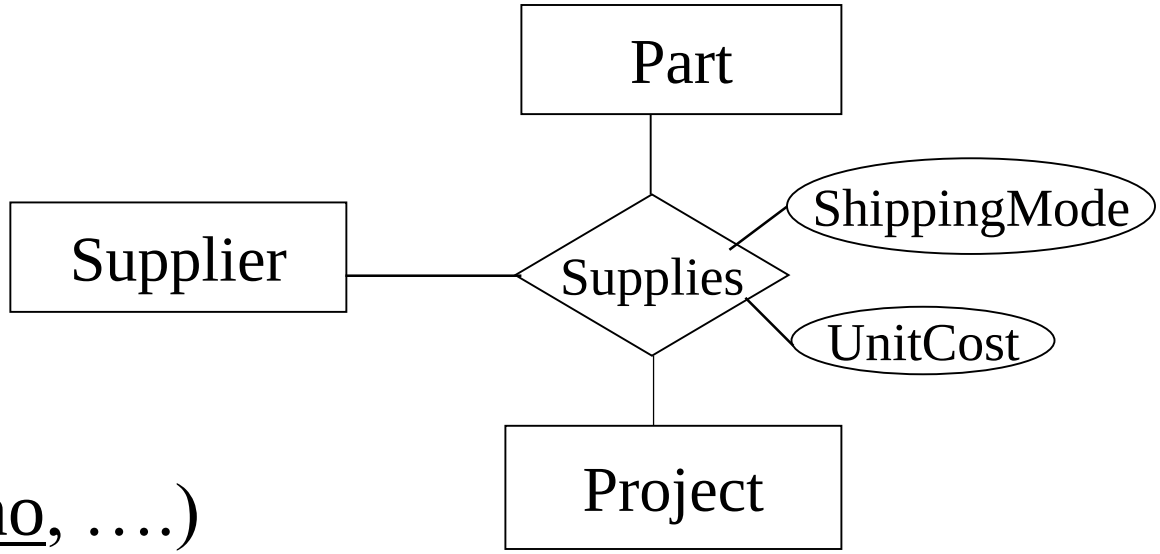
Map Complex relationship types

Create relation to represent relationship and include any attributes that are part of the relationship.

- Post copy of primary key attribute(s) of entities that participate in the complex relationship into new relation, to act as foreign keys.

- Any foreign keys that represent a ‘many’ relationship generally will also form the primary key of new relation, possibly in combination with some of the attributes of the relationship.

Map Complex relationship types



Supplier(<u>supplier_no</u>, ….)

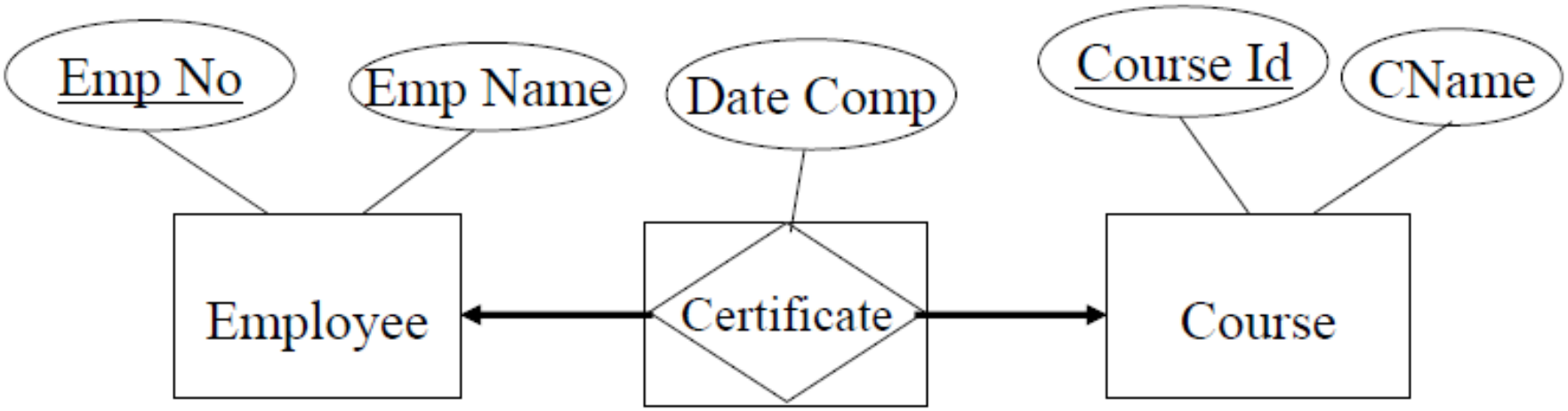
Project(<u>project_no</u>, ….)

Part(<u>part_no</u>, ….)

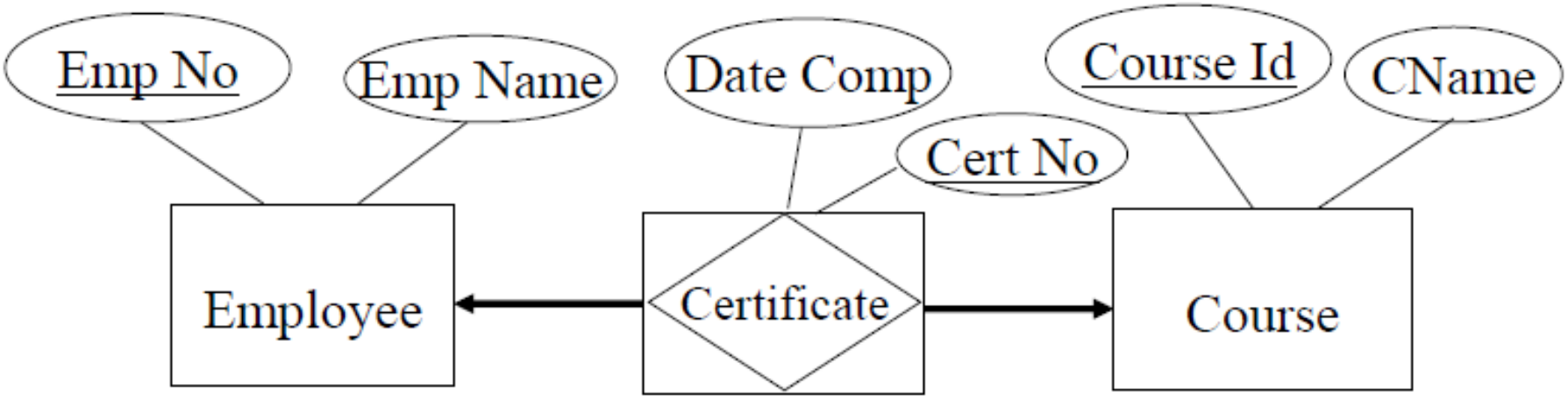
Supply(<u>*supplier_no*</u>, <u>project_*no*</u>, <u>*part_no*</u>,
Shipping_mode, Unit_Cost)

Mapping Associative Entities

- Essentially the same steps as mapping M:N, except if there is a special identifier for associative entity
- Create new associative relation for the associative entity and include PK of each of the two participating entity types as FK. These attributes become the PK (composite) if there is no identifier assigned. Otherwise the PK is the identifier of the associative entity
- include any attributes of the relationship to the new relation



Certificate(*Emp_No*, *Course_Id*, Date_Comp)
FK/NN FK/NN



Certificate(Cert_No, *Emp_No*, *Course_Id*, Date_Comp)
FK/NN FK/NN

Identifier Not Assigned

- The default primary key for the associative relation consists of the two primary key attributes from the other two relations.

- These attributes are then foreign keys that reference the other two relations.

Identifier Assigned

- Sometimes an identifier (called a surrogate identifier or key) is assigned to the associative entity type on the ER diagram. There are 2 possible reasons:

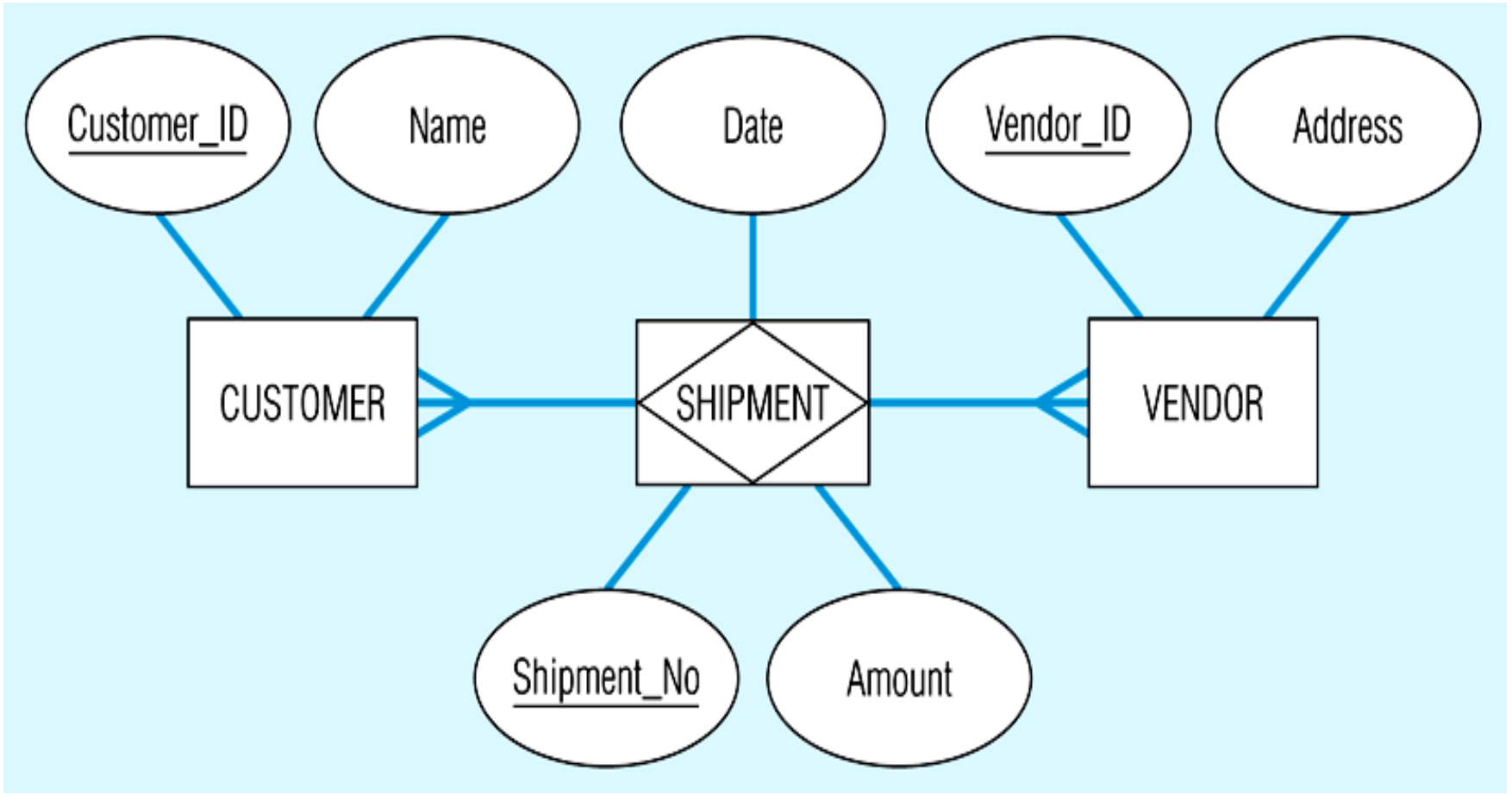
  i) The associative identity type has a natural identifier that is familiar to end users

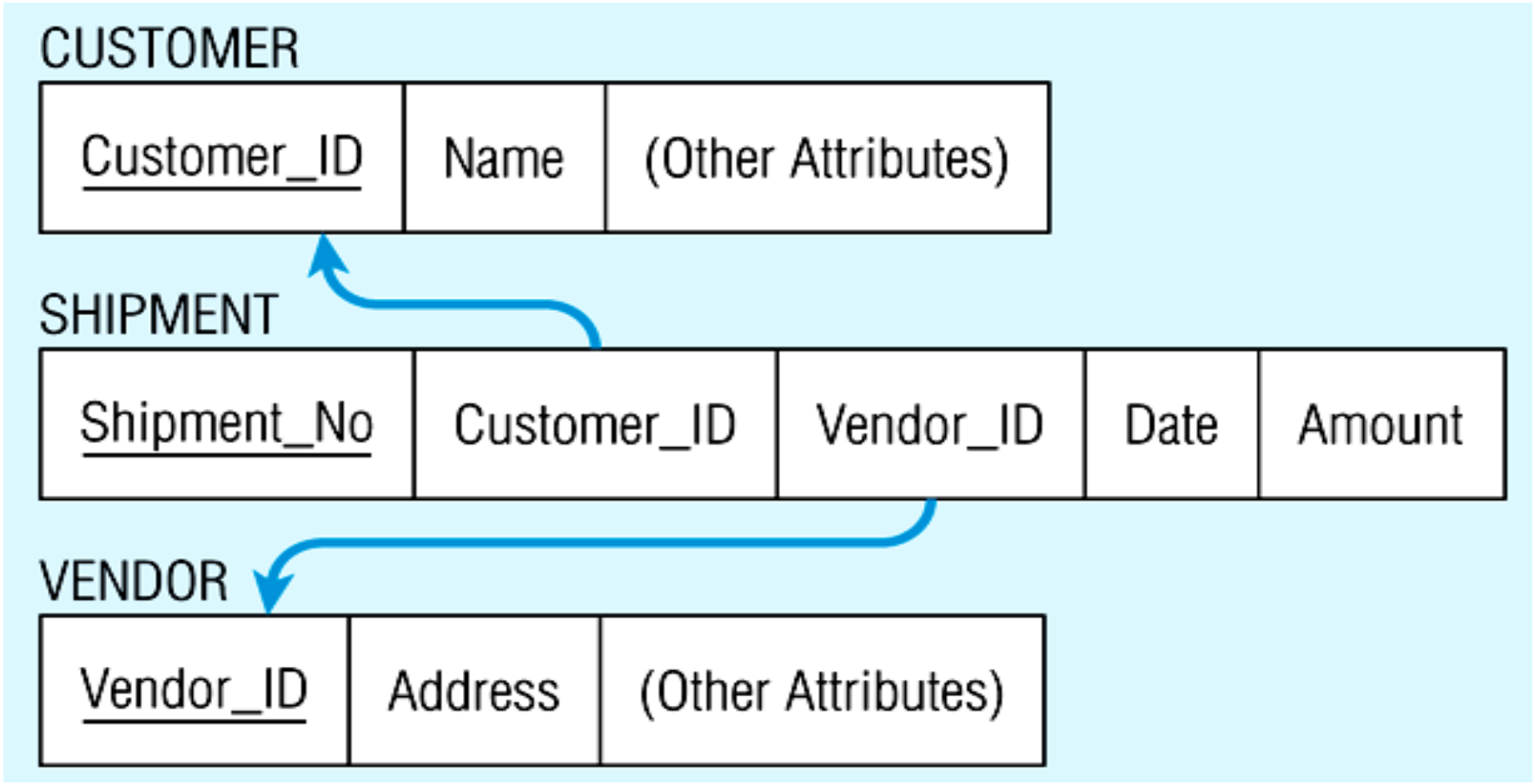
  ii)The default identifier (consisting of identifiers for each of the participating entity types) may not uniquely identify instances of the associative identity

Identifier Assigned

- A new associative relation is created to represent the associative entity
- However, the primary key for this relation is the identifier assigned on the ER diagram
- The primary keys for the two participating entity types are then included as foreign keys in the associative relation

Mapping an associative entity




CUSTOMER
Customer_ID
Name
(Other Attributes)
SHIPMENT
Shipment_No
Customer_ID
Vendor_ID
Date
Amount
VENDOR
Vendor_ID
Address
(Other Attributes)

Map Supertype/Subtype Relationships

The relational model does not directly support supertype/subtype relationships.

There are various strategies that database designers can use to represent these relationships with the relational data model.

Map Superclass/subclass relationship types

- Identify superclass as parent entity and subclass entity as child entity.
- There are various options on how to represent such a relationship as one or more relations.
- Most appropriate option dependent on number of factors such as:
  - disjointness and participation constraints on the superclass/subclass relationship,
  - whether subclasses are involved in distinct relationships,
  - number of participants in superclass/subclass relationship.

Mapping Superclass / Subclass Relationship

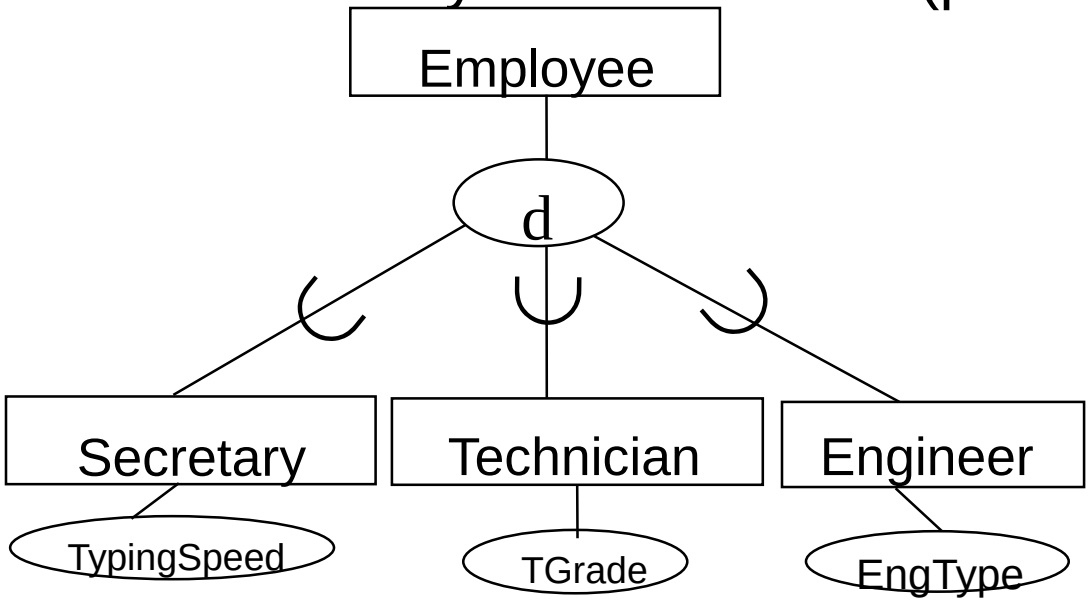
Participation Constraint – Partial

Disjoint Constraint - Disjoint

1. Create a relation for superclass
2. Create a relation for each subclass such that:

   {primary_key of superclass} U {attributes of subclass}

   key for subclass is (primary_key of superclass)



Partial , Disjoint

Employee

Eid	FName	Mlnit	LName	B'date	Address	JobType

Secretary

Eid	TypingSpeed

Technician

Eid	TGrade

Engineer

Eid	EngType

Mapping Superclass / Subclass Relationship

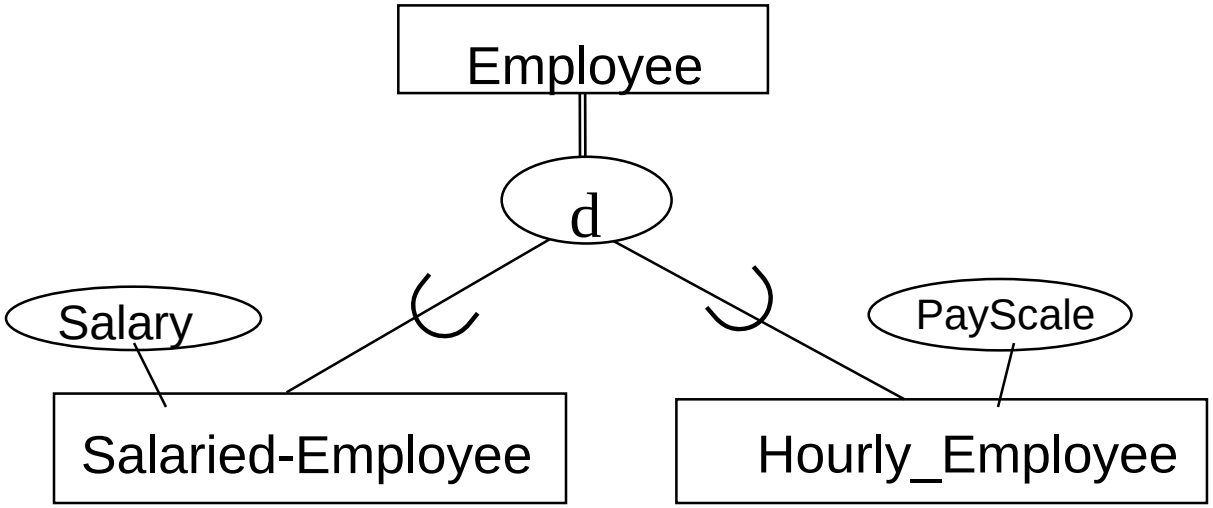
Participation Constraint – Total

Disjoint Constraint - Disjoint

Create a relation for each subclass such that:

{primary_key of superclass} U {attributes of superclass} U {attributes of subclass}

key for each relation is (primary_key of superclass)



Total, Disjoint

Salaried_ Employee

Ssn	FName	MInit	LName	B'date	Address	JobType	Salary

Hourly_ Employee

Ssn	FName	MInit	LName	B'date	Addr.	JobType	PayScale

Mapping Superclass / Subclass Relationship

Participation Constraint – Partial

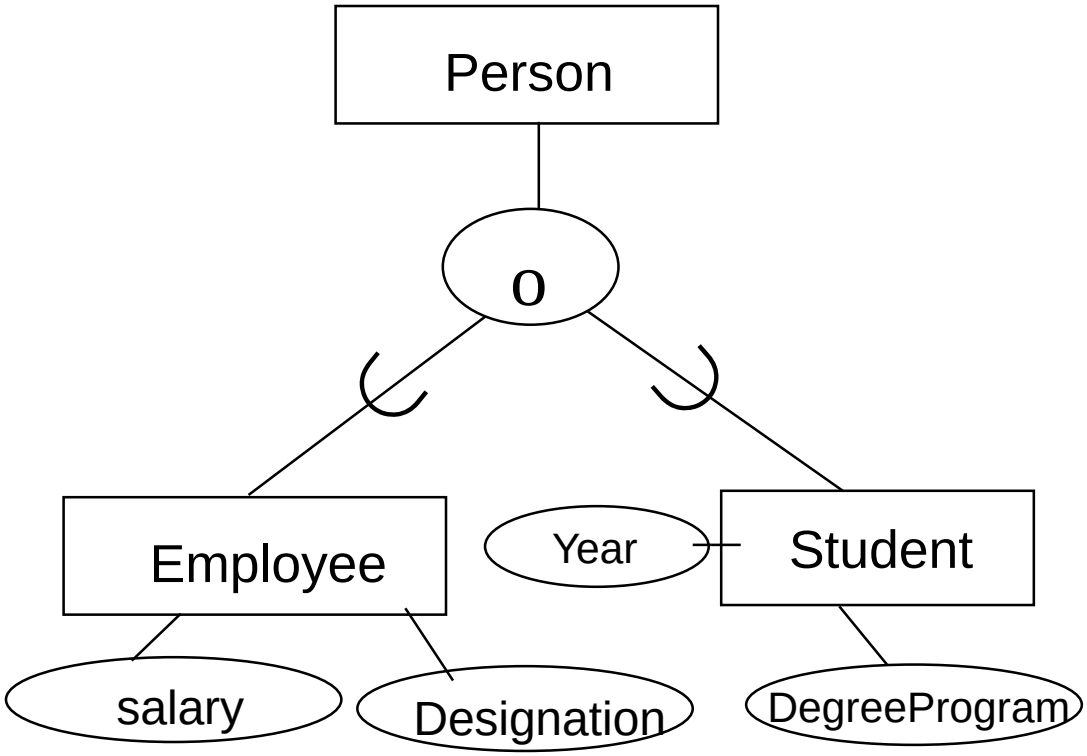
Disjoint Constraint - Overlap

- Create a relation for superclass
- Create a relation for all subclasses such that:

{primary_key of superclass} U {attributes of all subclasses}
U {with one or more discriminators to distinguish the subclass type}

key for each relation is (primary_key of superclass)

Partial, Overlap



Person

NID	FName	Mlnit	LName	B’date	Sex	Address

Emp_Stud

NID	Salary	Designation	DegreeProgram	Year	Emp.Flag	Student Flag

Mapping Superclass / Subclass Relationship

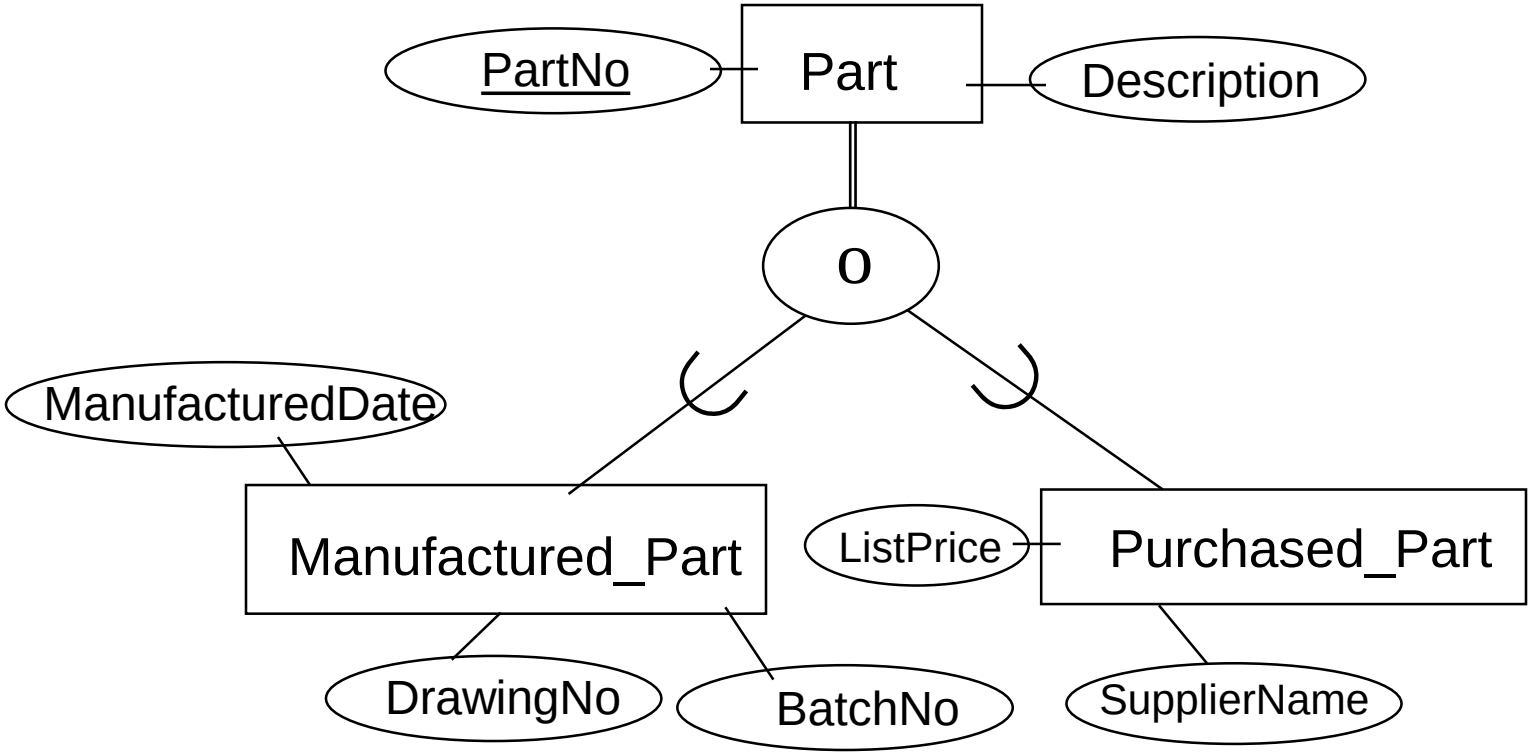
Participation Constraint – Total

Disjoint Constraint - Overlap

Create one relation such that:

{primary_key of superclass attributes} U {attributes of superclass} U {attributes of all subclasses} U {type attribute}

Total, Overlap



Part

{PartNo,Description,ManufacturedDate,DrawingNo,BatchNo,

SupplierName,ListPrice,ManufacFlag,PurchFlag}

Guidelines for Representation of Superclass / Subclass Relationship

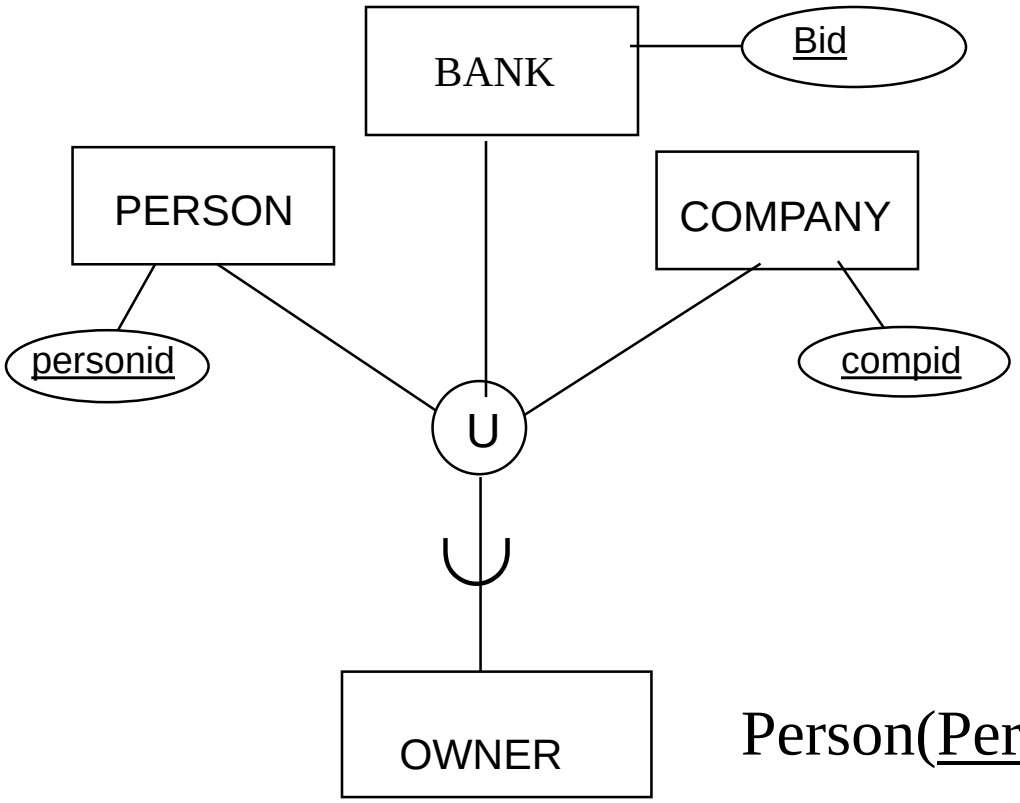
Table 15.1 Guidelines for the representation of a superclass/subclass relationship based on the participation and disjoint constraints.

Participation constraint	Disjoint constraint	Relations required
Mandatory	Nondisjoint {And}	Single relation (with one or more discriminators to distinguish the type of each tuple)
Optional	Nondisjoint {And}	Two relations: one relation for superclass and one relation for all subclasses (with one or more discriminators to distinguish the type of each tuple)
Mandatory	Disjoint {Or}	Many relations: one relation for each combined superclass/subclass
Optional	Disjoint {Or}	Many relations: one relation for superclass and one for each subclass

Mapping of categories

- If the super-classes have different keys we have to define a new key attribute called a ‘surrogate key’. When creating a relation to correspond to the category.
- Create a relation to correspond to the category, include any attributes of the category. The primary key is the surrogate key.
- Each super-class is also mapped into a relation with its own primary key, the surrogate key becomes a foreign key for this.

Mapping of categories



Person(PersonID,name,…,OwnerID)

Company(CompID,name,…,OwnerID)

Bank (BID, ….. OwnerID)

Owner(OwnerID, OwnerType)

END