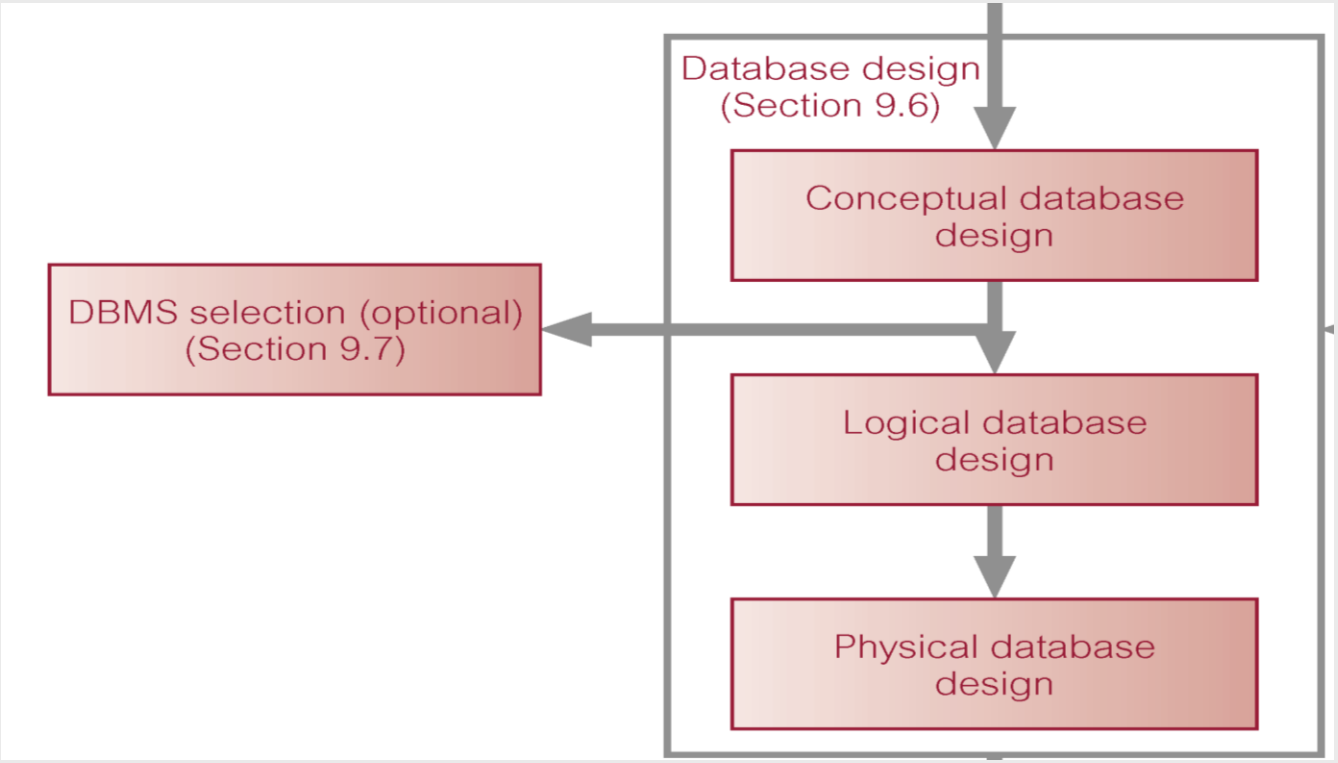
Chapter 5

Logical Database Design for the Relational Model

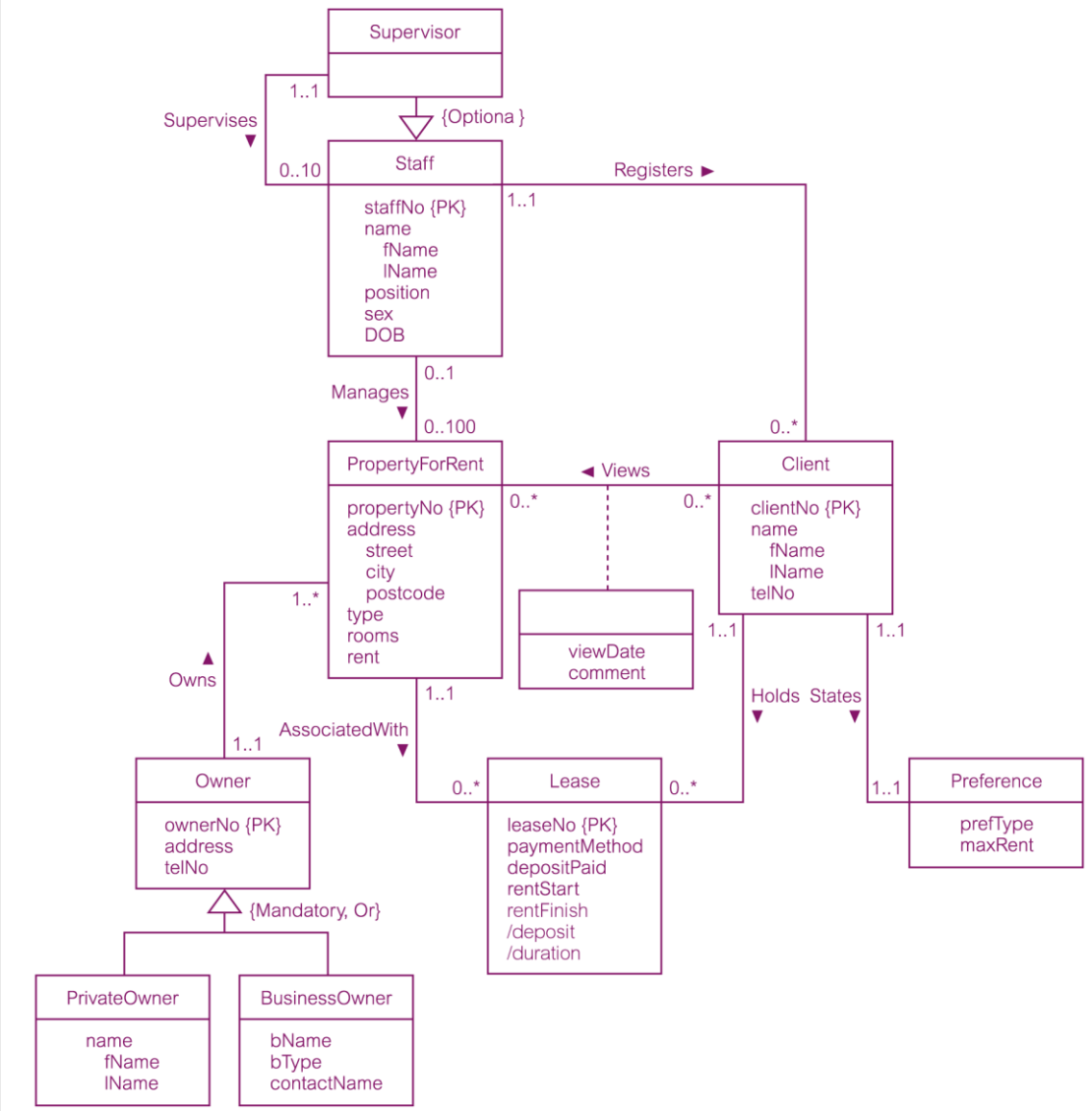
Chapter 5 - Objectives

- How to derive a set of relations from a conceptual data model.
- To create relations for the logical data model to represent the entities, relationships, and attributes that have been identified.

Build Logical Data Model



Conceptual data model for Staff view showing all attributes

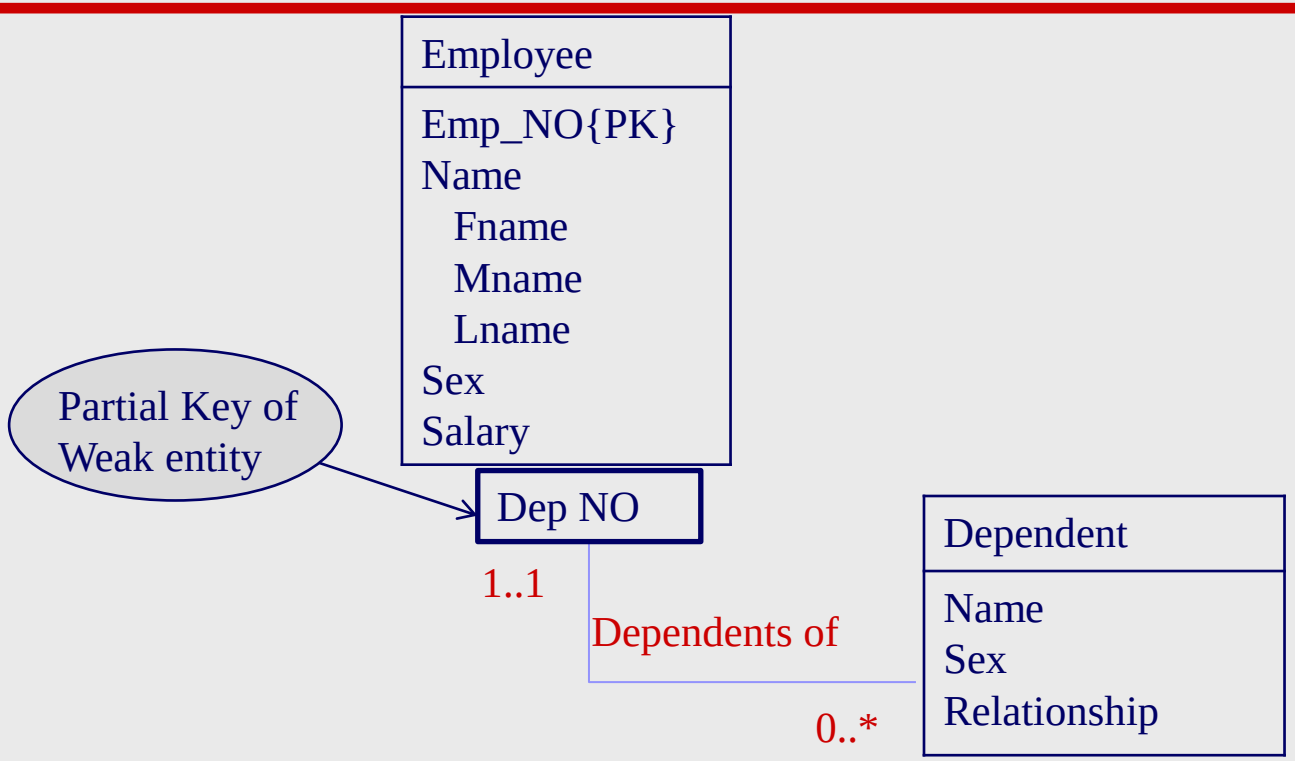


Derive relations for logical data model

- **(1) Strong entity types**
  - For each strong entity in the data model, create a relation that includes all the simple attributes of that entity. For composite attributes, include only the constituent simple attributes.

  (2) Weak entity types
  - For each weak entity in the data model, create a relation that includes all the simple attributes of that entity.
  - A weak entity must include its Partial key and its owner entity type PK as a FK. The combination of the two keys form the PK of the weak entity.

Example



Employee (Emp_NO, Fname,Mname ,Lname, Sex, Salary)
Primary Key Emp_NO
DEPENDENT (DepNo, EmpNo, Name, Sex, Relationship)
Primary Key DepNo, EmpNo
Foreign Key EMpNo refrences Employee (Emp_NO)

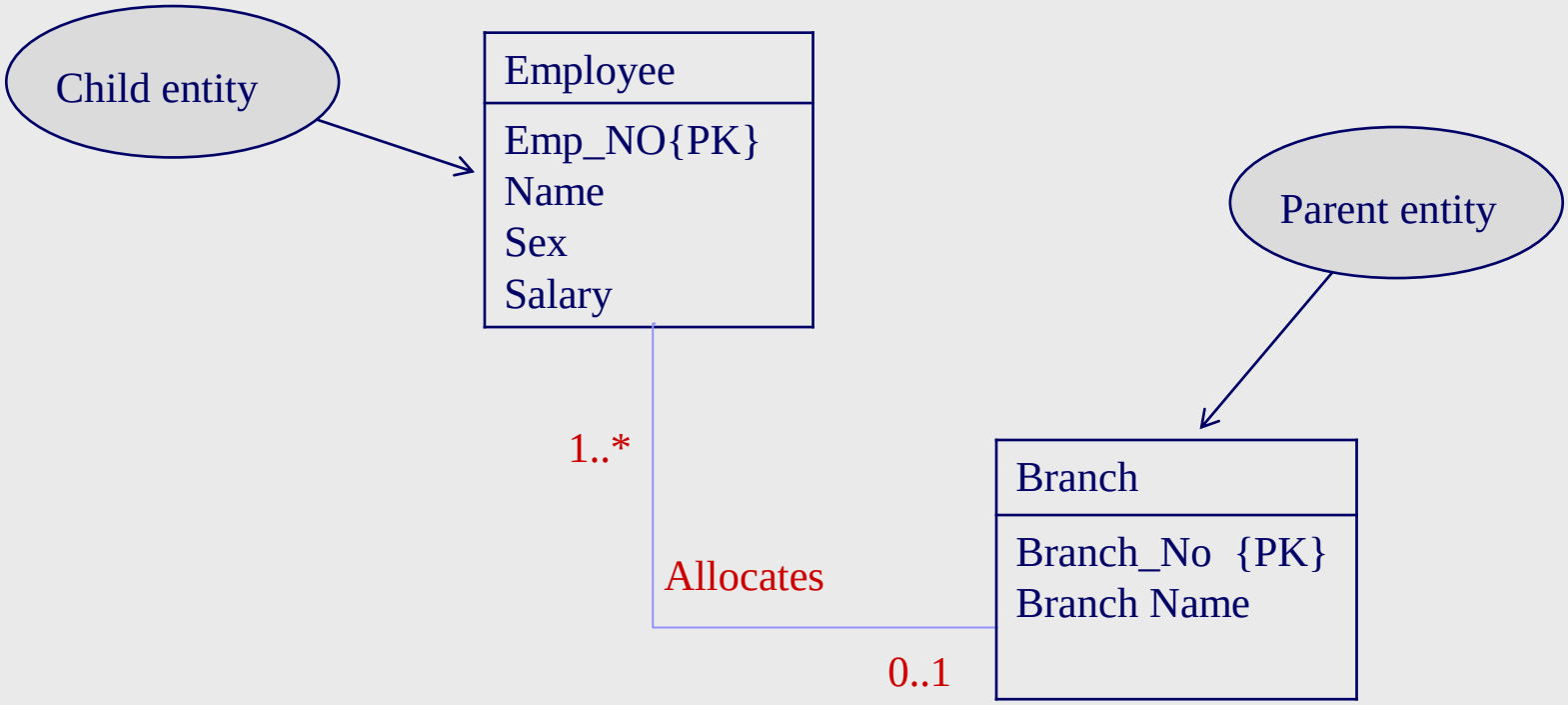
Derive relations for logical data model

- **(3) One-to-many (1:*) binary relationship types**
  - For each 1:* binary relationship, the entity on the ‘one side’ of the relationship is designated as the parent entity and the entity on the ‘many side’ is designated as the child entity. To represent this relationship, post a copy of the Primary Key attribute(s) of parent entity into the relation representing the child entity, to act as a foreign key.

OR

  - Entity with many cardinality in relationship is designated as parent entity, and entity with one cardinality is designated as child entity.

Example



Employee (Emp_NO, Name, Sex, Salary,BranchNo)
Primary Key Emp_NO
Foreign Key BranchNo refrences Branch (Branch_No)
Branch (Branch_No, Branch Name)
Primary Key Branch_No

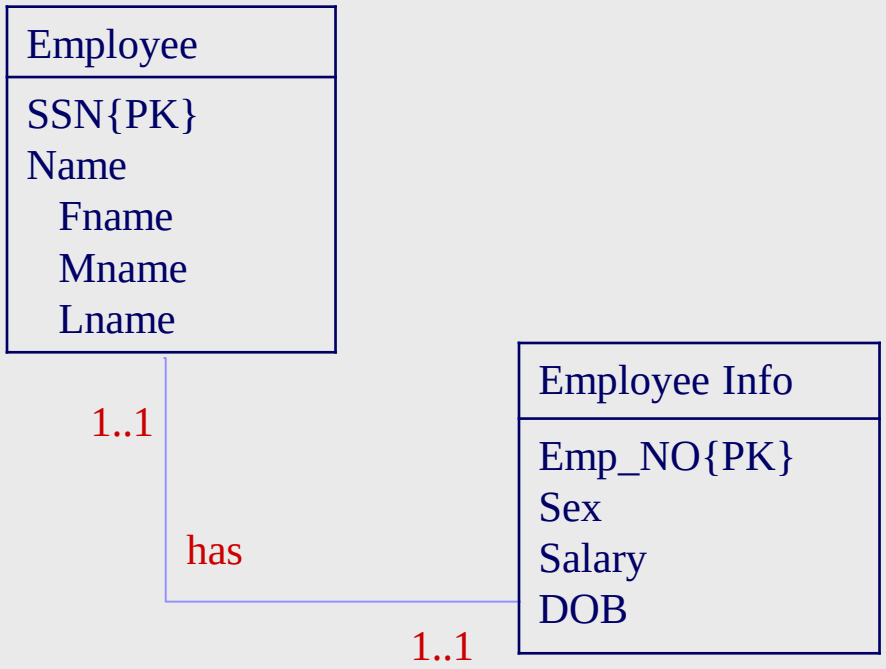
Derive relations for logical data model

- **(4) One-to-one (1:1) binary relationship types**
  - Creating relations to represent a 1:1 relationship is more complex as the cardinality cannot be used to identify the parent and child entities in a relationship. Instead, the participation constraints are used to decide whether it is best to represent the relationship by combining the entities involved into one relation or by creating two relations and posting a copy of the Primary Key from one relation to the other.
  - Consider the following
    - » *(a) mandatory* participation on *both* sides of 1:1 relationship;
    - » *(b) mandatory* participation on *one* side of 1:1 relationship;
    - » *(c) optional* participation on *both* sides of 1:1 relationship.

Derive relations for logical data model

- **(a) *Mandatory* 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 the new relation, while the other (if one exists) is used as an alternate key.

Example

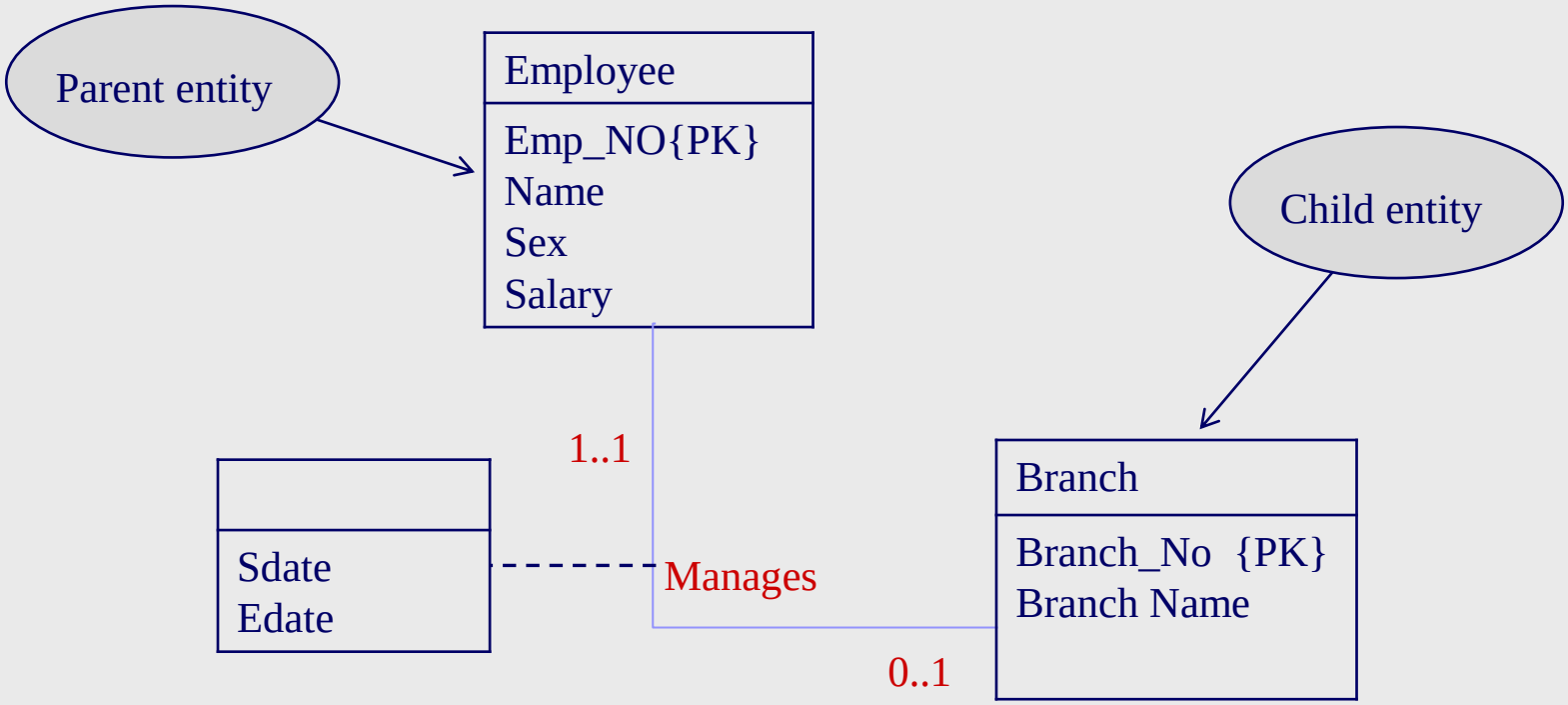


Employee (Emp_NO, SSN, Fname,Mname,Lname, Sex, Salary, DOB)
Primary Key Emp_NO
Alternate key SSN
OR
Employee (SSN , Emp_NO, Fname,Mname ,Lname, Sex, Salary,DOB)
Primary Key SSN
Alternate key Emp_NO

Derive relations for logical data model

- **(b) *Mandatory* participation on *one* side of a 1:1 relationship**
  - Identify parent and child entities using participation constraints. Entity with optional participation in relationship is designated as parent entity, and entity with mandatory participation is designated as child entity. A copy of Primary Key of the parent entity is placed in the relation representing the child entity. If the relationship has one or more attributes, these attributes should follow the posting of the Primary Key to the child relation.

Example

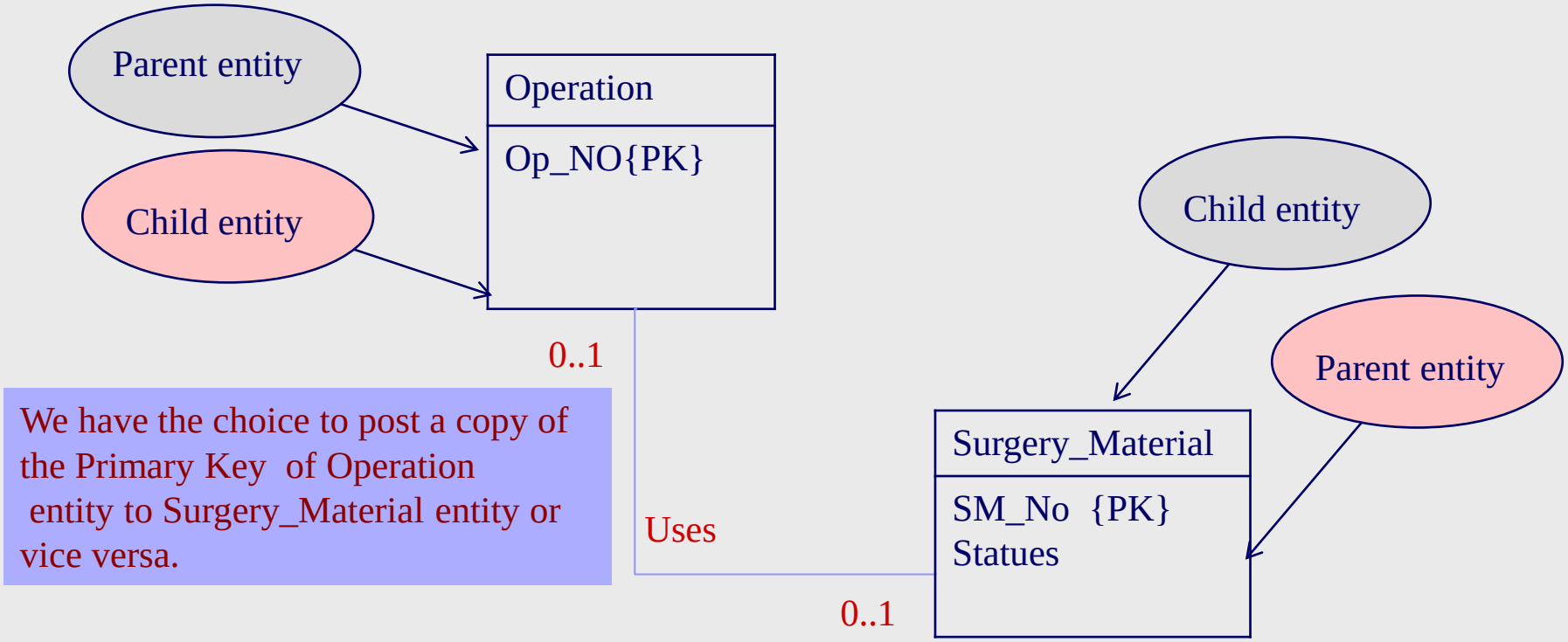


Employee (Emp_NO, Name, Sex, Salary)
Primary Key Emp_NO
Branch(Branch_No, Branch Name,Emp_NO, Sdate,Edate)
Primary Key Branch_No
Foreign key Emp_No refrences Employee (Emp_NO)

Derive relations for logical data model

- **(c) *Optional* participation on *both* sides of a 1:1 relationship**
  - » In this case, the designation of the parent and child entities is arbitrary unless we can find out more about the relationship that can help a decision to be made one way or the other.

Example

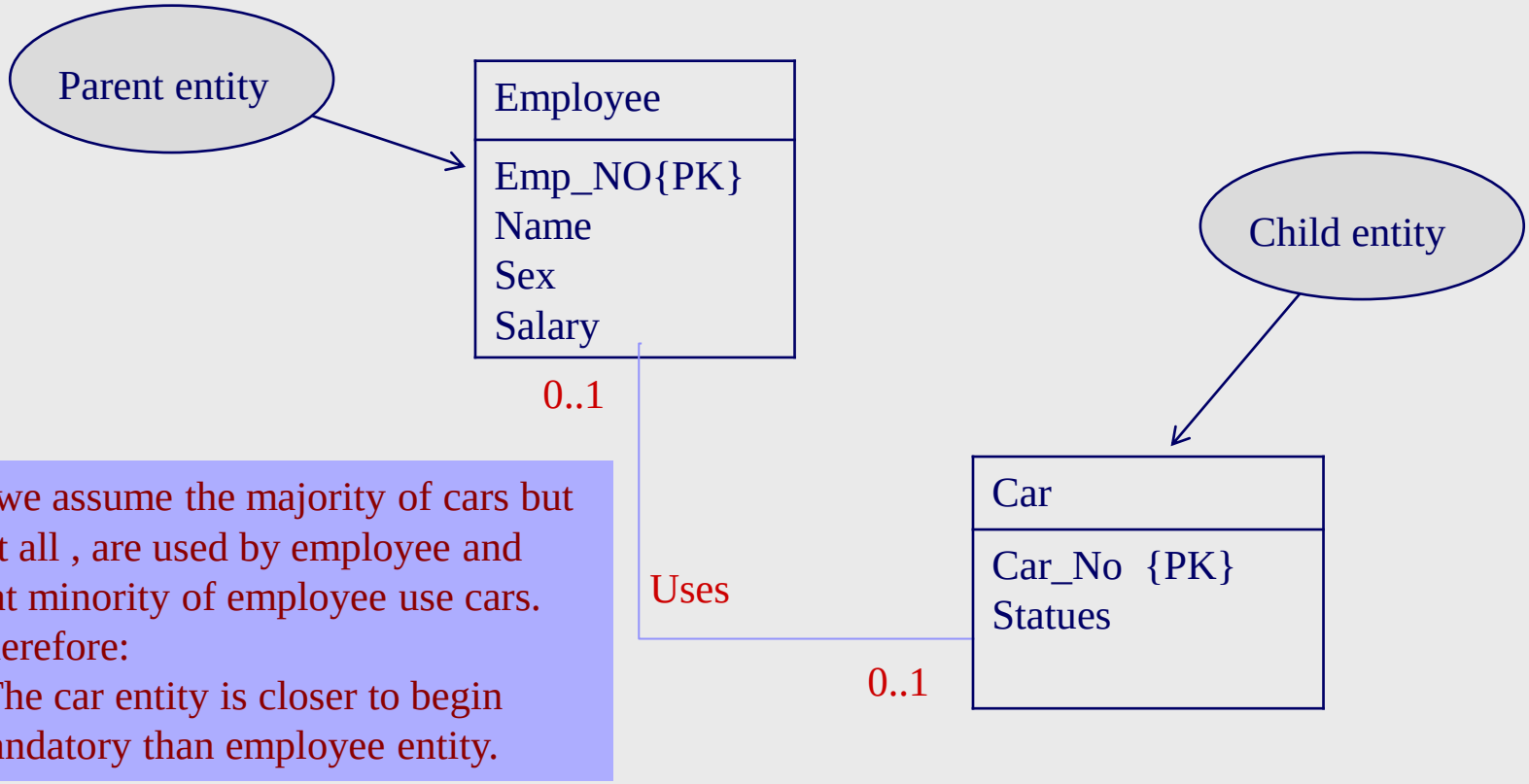


We have the choice to post a copy of the Primary Key of Operation entity to Surgery_Material entity or vice versa.

Operation(Op_NO, ……)
Surgery_Material(SM_No, Statues,Op_NO)

Operation(Op_NO, ……, SM_No)
Surgery_Material(SM_No, Statues)

Example



If we assume the majority of cars but not all , are used by employee and that minority of employee use cars. Therefore:
- The car entity is closer to begin mandatory than employee entity.

Employee (Emp_NO, Name, Sex, Salary)
Primary Key Emp_NO
Car (Car_NO, Statues, Emp_NO)
Primary Key Car_NO

(5)Recursive relationships Mapping conclusion

a) **One-to-one (1:1) recursive relationships**
Single relation with two copies for the primary key with different names.

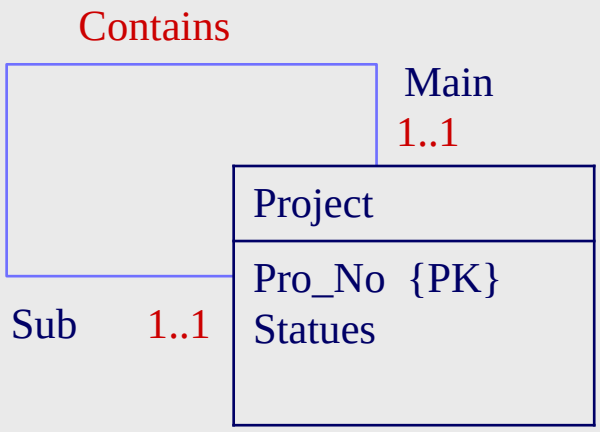
b) **One-to-many (1:*) recursive relationships**
Single relation with two copies for the primary key with different names.

c) **Many-to-many (*:*) recursive relationships**
Two relations
   - » One relation for the entity type.
   - » And create a new relation to represent the relationship. The new relation would only have two attributes, both copies of the primary key.

Example a)
One-to-one (1:1) recursive relationships

Business rule:
Each project must contain anther project, no more.

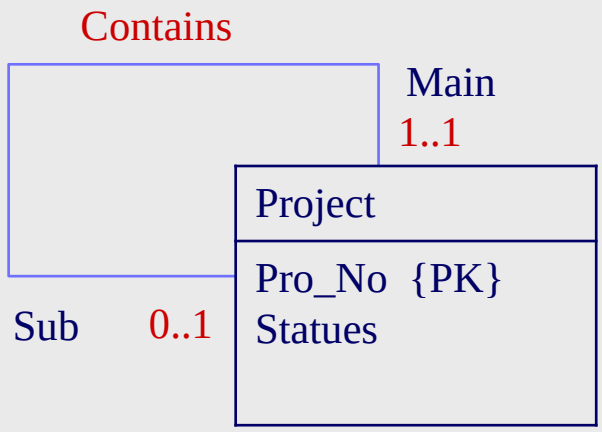
This case occurs rarely in real life applications.



Project(Pro_No,SubPro_No, Statues)
OR
Project(Pro_No,MainPro_No, Statues)

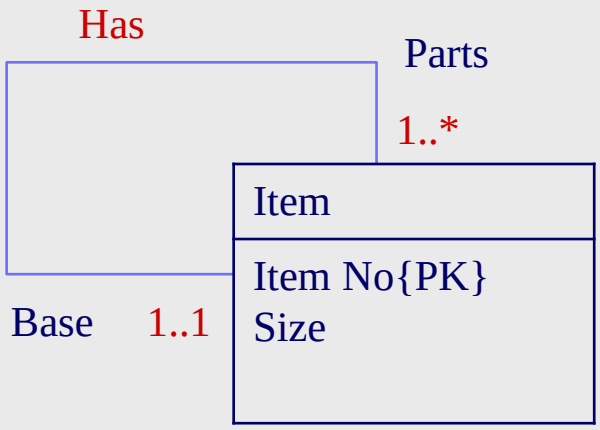
Example a)
One-to-one (1:1) recursive relationships

Business rule:
Each project may contain another project, no more.



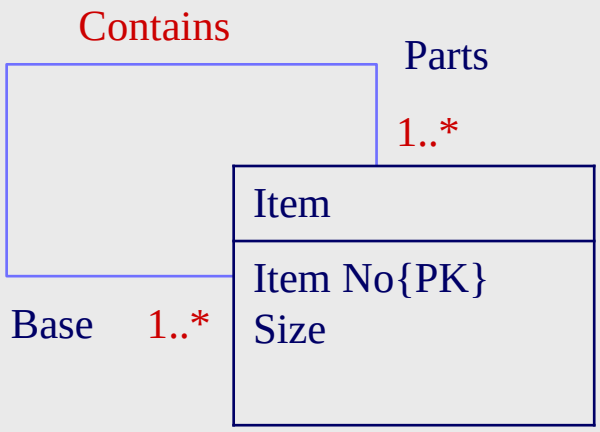
Project(Pro_No,MainPro_No, Statues)

Example b)
One-to-many (1:*) recursive relationships



Item(Item_No, Base_No, Size)

Example c)
Many-to-many (*:*) recursive relationships

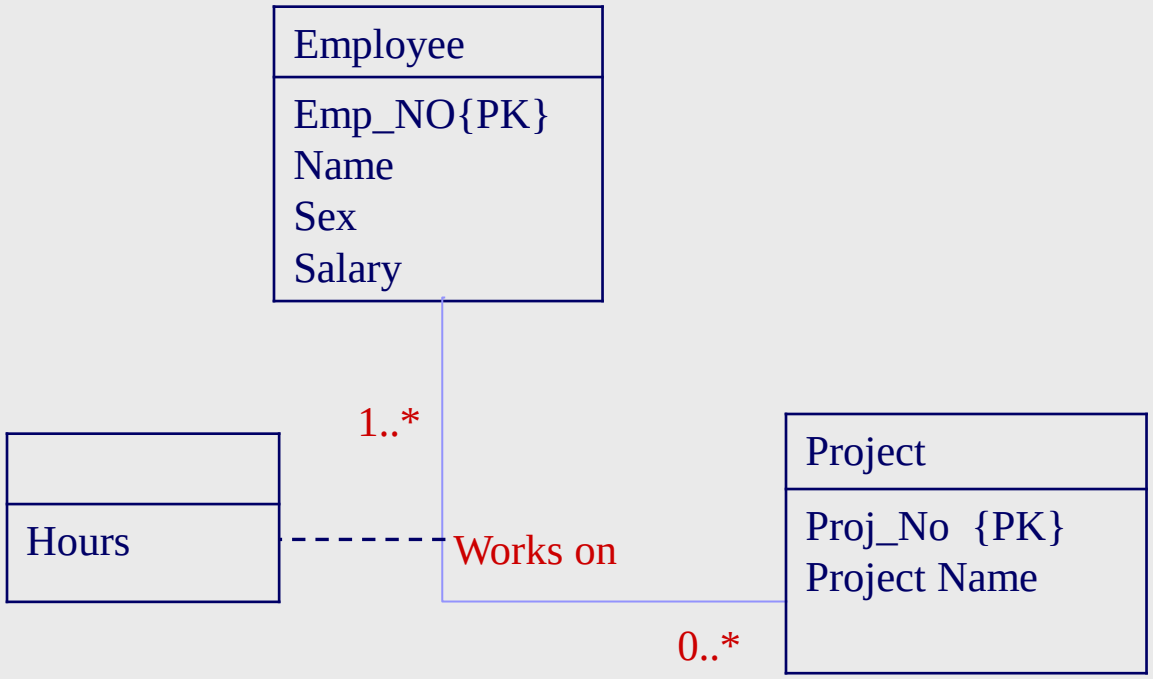


Item(Item_No, Size)
Parts(Item_No, Parts_No)

Derive relations for logical data model

- **(6) Many-to-many (*:*) binary relationship types**
  - Create a relation to represent the relationship and include any attributes that are part of the relationship. We post a copy of the Primary Key attribute(s) of the entities that participate in the relationship into the new relation, to act as foreign keys. These foreign keys will also form the Primary Key of the new relation, possibly in combination with some of the attributes of the relationship.

Example

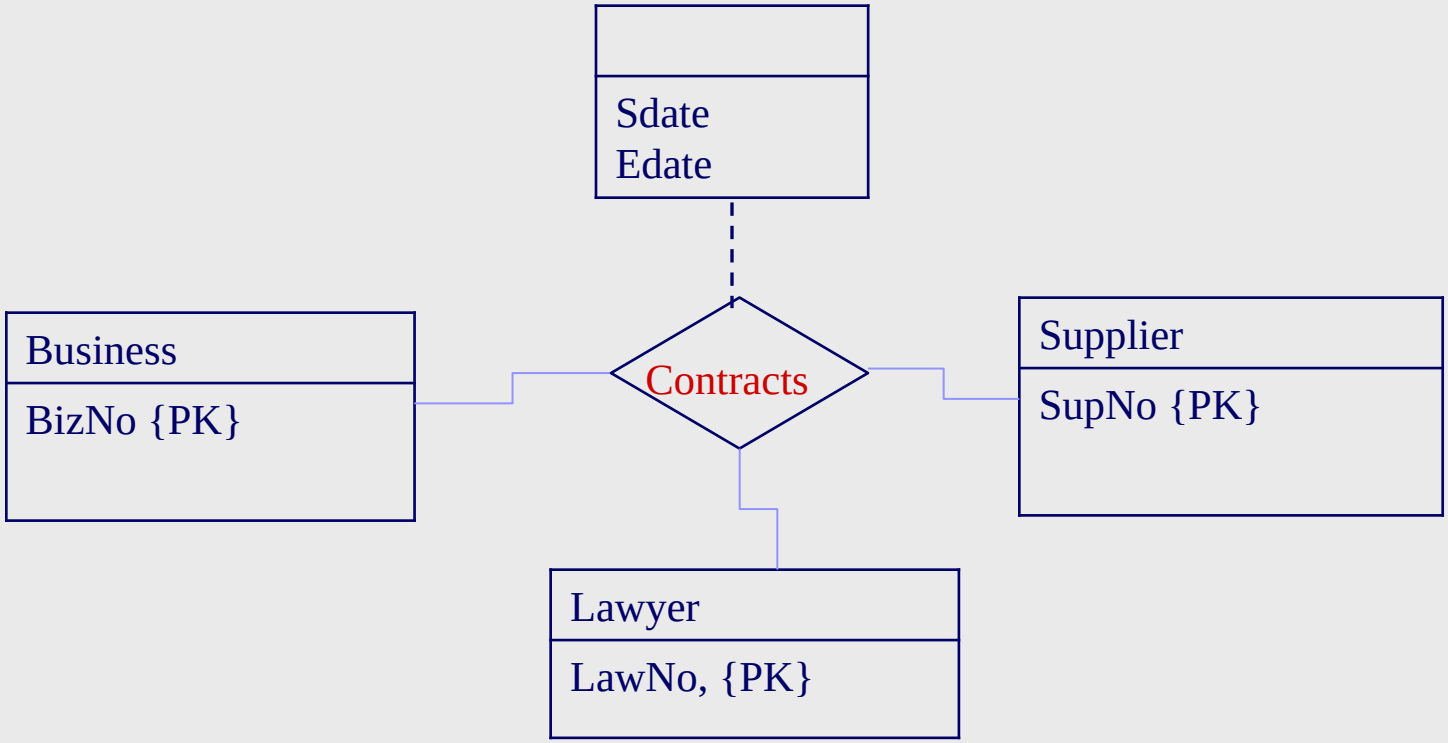


Employee (Emp_NO, Name, Sex, Salary, Branch No)
Primary Key Emp_NO
Project (Proj_No, ProjectName)
Primary Key Proj_No
Works-on(EmpNo,ProjNo, hours)
Primary Key Emp_NO Proj_No,

Derive relations for logical data model

- **(7) Complex relationship types**
  - Create a relation to represent the relationship and include any attributes that are part of the relationship. Post a copy of the Primary Key attribute(s) of the entities that participate in the complex relationship into the new relation, to act as foreign keys. Any foreign keys that represent a ‘many’ relationship (for example, 1..*, 0..*) generally will also form the Primary Key of this new relation, possibly in combination with some of the attributes of the relationship.

Example



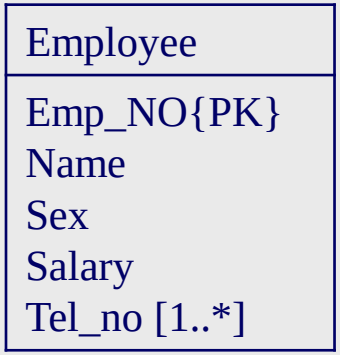
Business (BizNo., ……)
Supplier(SupNo, ……..)
Lawyer(LawNo,……)
Contract(BizNo, SupNo, LawNo, StDate, EDate)

Derive relations for logical data model

- **(8) Multi-valued attributes**
  - Create a 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, the Primary Key of the new relation is the combination of the multi-valued attribute and the Primary Key of the entity.

Example

The Primary Key of new relation:
- the multi-valued attribute if itself an alternate key. (e.g. we sure that there are no duplicated tel_nos for employees).
-OR the combination of the multi-valued attribute and the Primary Key of the entity. (e.g. 2 brothers work in same company).



Employee (Emp_NO, Name, Sex, Salary)
Primary Key Emp_NO
Telephone(Tel_no, EmpNo)
Primary Key Tel_no
OR
Telephone(Tel_no, EmpNo)
Primary Key Tel_no, EmpNo

Summary of how to map entities and relationships to relations

Entity/Relationship	Mapping
Strong entity	Create relation that includes all simple attributes.
Weak entity	Create relation that includes all simple attributes (primary key still has to be identified after the relationship with each owner entity has been mapped).
1:* binary relationship	Post primary key of entity on 'one' side to act as foreign key in relation representing entity on 'many' side. Any attributes of relationship are also posted to 'many' side.
1:1 binary relationship:	
(a) Mandatory participation on both sides	Combine entities into one relation.
(b) Mandatory participation on one side	Post primary key of entity on 'optional' side to act as foreign key in relation representing entity on 'mandatory' side.
(c) Optional participation on both sides	Arbitrary without further information.
Superclass/subclass relationship	See Table 16.1.
: binary relationship, complex relationship	Create a relation to represent the relationship and include any attributes of the relationship. Post a copy of the primary keys from each of the owner entities into the new relation to act as foreign keys.
Multi-valued attribute	Create a relation to represent the multi-valued attribute and post a copy of the primary key of the owner entity into the new relation to act as a foreign key.

Class Exercise

- In your groups
  - One Conceptual Data Model
  - Transform to Logical database design
    - Derive a set of relations
      - Map Entities and Relationships to relations
  - Prepare a Presentation