Chapter 17

Methodology

Logical Database Design for the Relational Model

Chapter 17 - Objectives

- How to derive a set of relations from a conceptual data model.
- How to validate these relations using the technique of normalization.
- How to validate a logical data model to ensure it supports the required transactions.

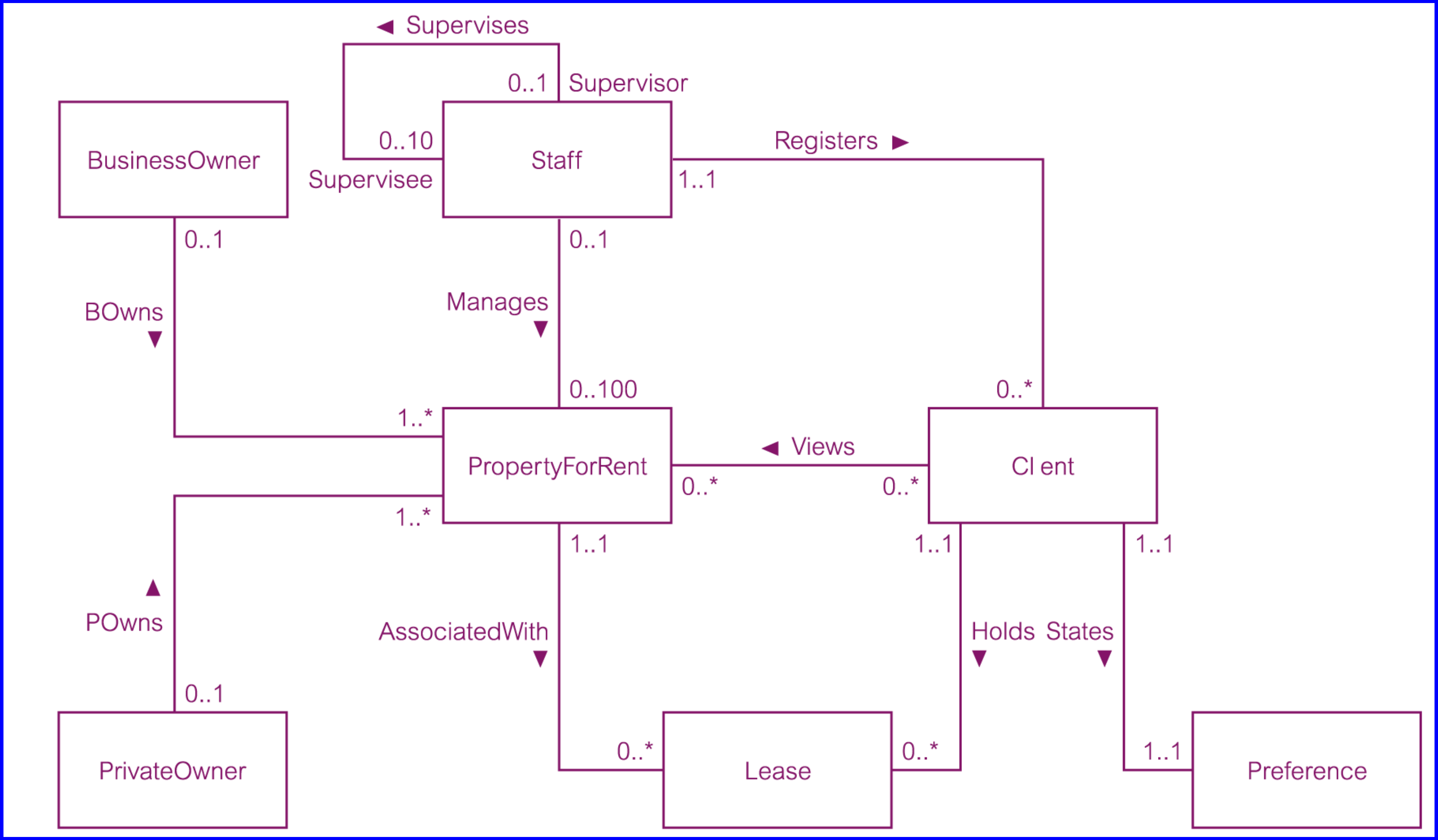
Step 2 Build and Validate Logical Data Model

- To translate the conceptual data model into a logical data model and then to validate this model to check that it is structurally correct using normalization and supports the required transactions.

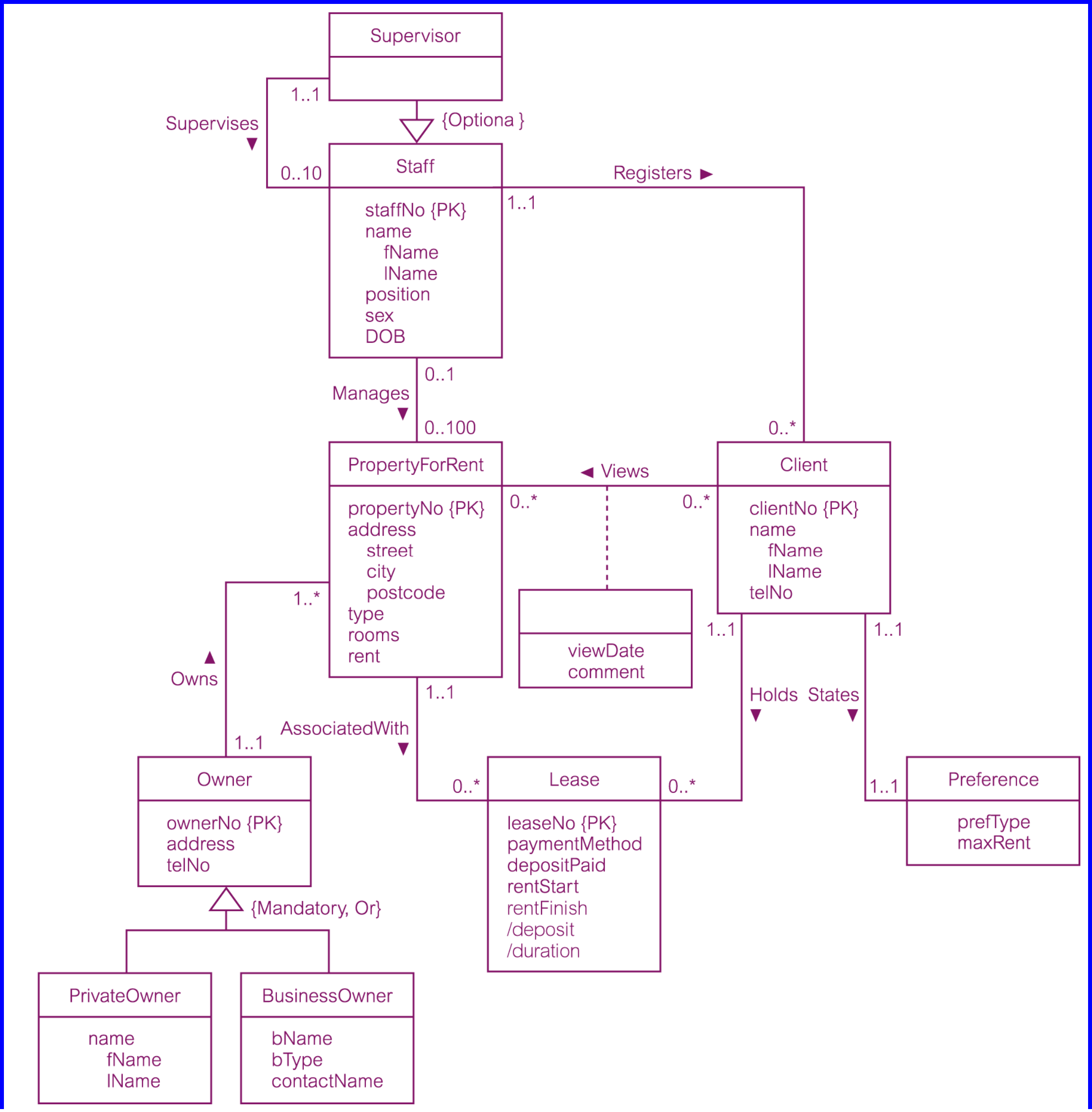
Step 2 Build and Validate Logical Data Model

- **Step 2.1 Derive relations for logical data model**
  - **To create relations for the logical data model to represent the entities, relationships, and attributes that have been identified.**

Conceptual data model for Staff view



Conceptual data model for Staff view showing all attributes



Step 2.1 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.
  - Example: **Staff**(staffNo, fName, lName, position, sex, DOB)
    Primary Key: staffNo

  (2) Weak entity types
  - For each weak entity in the data model, create a relation that includes all the simple attributes of that entity. The primary key of a weak entity is partially or fully derived from each owner entity and so the identification of the primary key of a weak entity cannot be made until after all the relationships with the owner entities have been mapped.
  - Example: **Preference**(prefType, maxRent)
  - **Primary Key**: None (at present)

Step 2.1 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.
  - Example:
    Staff(staffNo, fName, lName, position, sex, DOB)
    Primary Key: staffNo
  - **Client**(clientNo, fName, lName, telNo, staffNo)
    Primary Key: clientNo
    Alternate Key: telNo
    Foreign Key: staffNo **references** Staff(staffNo)

Step 2.1 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.

Step 2.1 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: Client States Preference
    Client(clientNo, fName, lName, telNo, prefType, maxRent, staffNo)
    Primary Key: clientNo
    Foreign Key: staffNo **references** Staff(staffNo)

Step 2.1 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: If not every client specifies preferences, then
    Client(clientNo, fName, lName, telNo, staffNo)

    Preference(clientNo, prefType, maxRent)
    Primary Key: clientNo
    Foreign Key: clientNo **references** Client(clientNo)

Step 2.1 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 (see Text for further details).

Step 2.1 Derive relations for logical data model

- **(5) One-to-one (1:1) recursive relationships**
  - For a 1:1 recursive relationship, follow the rules for participation as described above for a 1:1 relationship.
    - » mandatory participation on both sides, represent the recursive relationship as a single relation with two copies of the primary key. One copy of the primary key represents a foreign key and should be renamed.
      Example: Student relation (primary key: ID) with a recursive "Buddy" relationship; Add a new attribute "buddyID" which is a foreign key identifying the buddy of each student.

Step 2.1 Derive relations for logical data model

- **(5) One-to-one (1:1) recursive relationships (con't)**
  - For a 1:1 recursive relationship, follow the rules for participation as described above for a 1:1 relationship.
    - » mandatory participation on only one side, option to create a single relation with two copies of the primary key, or to create a new relation to represent the relationship. The new relation would only have two attributes, both copies of the primary key. As before, the copies of the primary keys act as foreign keys and have to be renamed to indicate the purpose of each in the relation.
      Example: Staff relation with recursive "Supervises" relationship.
      (1) Add a "supervisor" attribute to Staff relation OR
      (2) Create a new relation SuperInfo(staffNo, supervisor) where "supervisor" is a foreign key referencing Staff(staffNo).

Step 2.1 Derive relations for logical data model

- **(7) 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: Client *Views* PropertyForRent

    Client(clientNo, fName, lName, telNo, prefType, maxRent, staffNo)

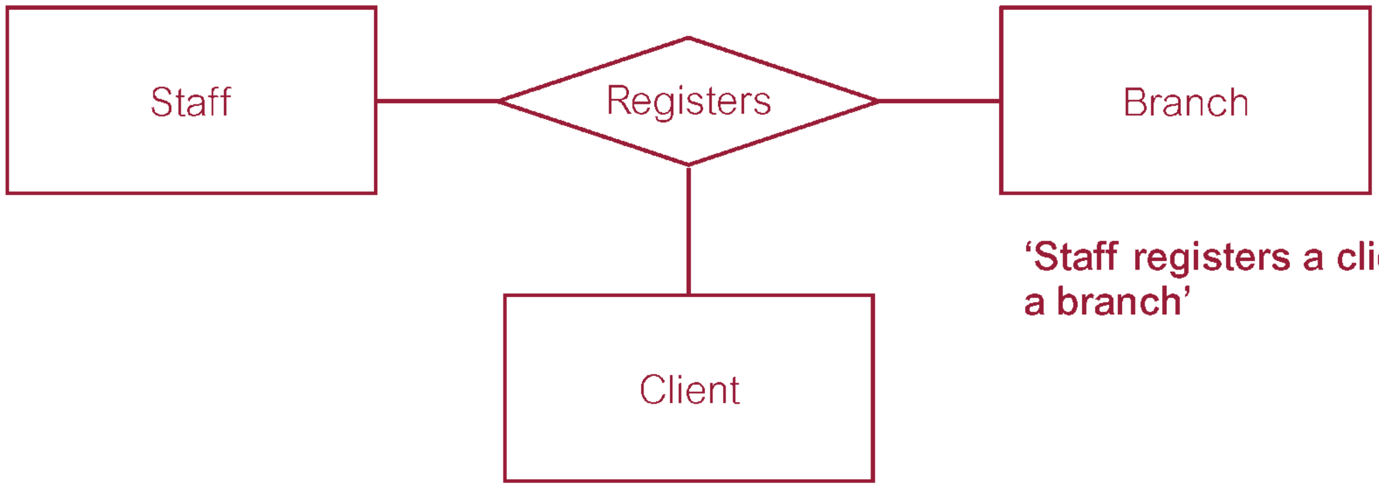
    PropertyForRent(propertyNo, street, city, postcode, type, rooms, rent)

    Viewing(clientNo, propertyNo, dateView, comment)
    Primary Key: clientNo, propertyNo
    Foreign Key: clientNo **references** Client(clientNo)
    Foreign Key: propertyNo **references** PropertyForRent(propertyNo)

Step 2.1 Derive relations for logical data model

- **(8) 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.

Ternary relationship called *Registers*



'Staff registers a client at a branch'

Example: Complex Relationships

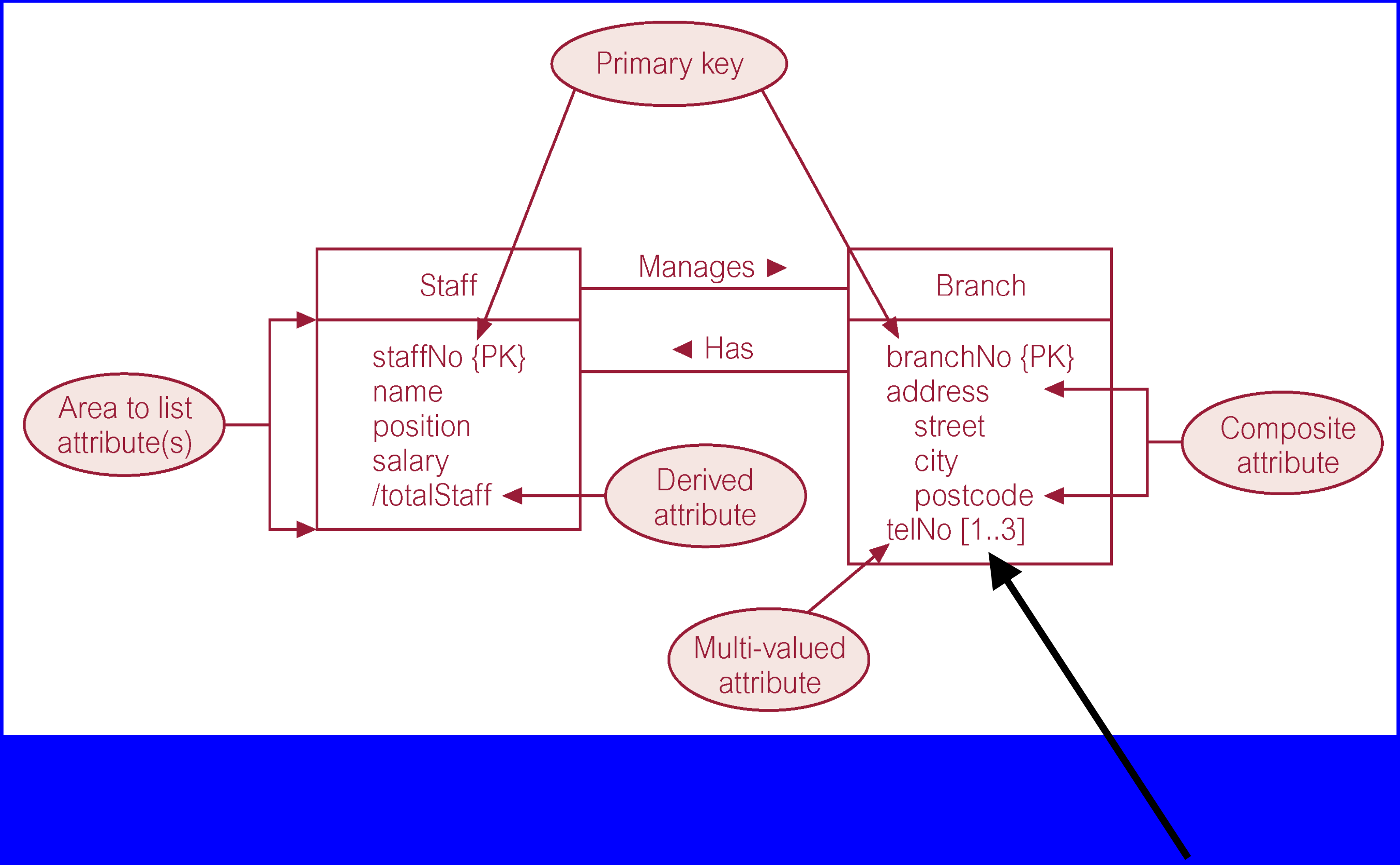
- **Staff(staffNo**, fName, lName, position, …)
  Primary Key: staffNo
- **Branch**(branchNo, street, city, postcode)
  Primary Key: branchNo
- **Client**(clientNo, fName, lName, …)
  Primary Key: clientNo

- **Registration**(clientNo, branchNo, staffNo, dateJoined)
  Primary Key: clientNo
  Foreign Key: branchNo **references** Branch(branchNo)
  Foreign Key: clientNo **references** Client(clientNo)
  Foreign Key: staffNo **references** Staff(staffNo)

Step 2.1 Derive relations for logical data model

- **(9) 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: Multi-Valued Attribute

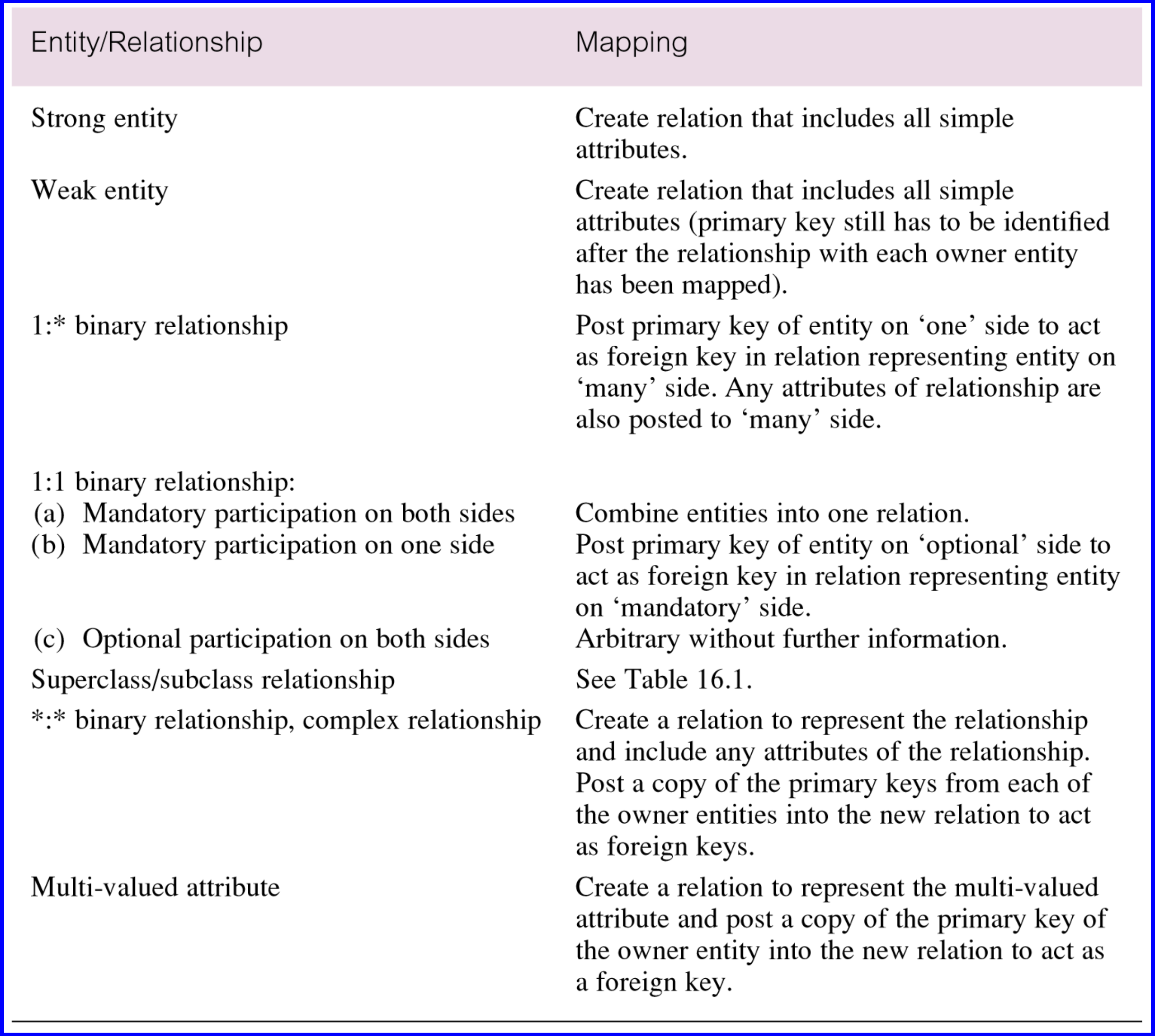


Multi-Valued Attribute

- **Branch**(branchNo, street, city, postcode)
  Primary Key: branchNo

- **Telephone**(telNo, branchNo)
  Primary Key: telNo
  Foreign Key: branchNo **references** Branch(branchNo)

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.

Step 2.2 Validate relations using normalization

- To validate the relations in the logical data model using normalization.

Step 2.3 Validate relations against user transactions

- To ensure that the relations in the logical data model support the required transactions.

Step 2.5 Review logical data model with user

- To review the logical data model with the users to ensure that they consider the model to be a true representation of the data requirements of the enterprise.