Chapter 17

Methodology

Logical Database Design for the Relational Model

Logical Database Design for the Relational Model

- **Step 2 Using a Data Model (e.g., Relational Data Model), Build logical data model and validate it**
  - Step 2.1 Derive relations for logical data model
  - Step 2.2 Validate relations using normalization
  - Step 2.3 Check integrity constraints

Step 2.1 Derive relations for logical data model

- Create relations for logical data model to represent entities, relationships, and attributes that have been identified

Step 2.1 Derive relations for logical data model

- In this step, we derive relations for the logical data model to represent the entities, relationships, and attributes
- We describe the composition of each relation using a Database Definition Language (DBDL) for relational databases
- Using the DBDL, we first specify the name of the relation followed by a list of the relation’s simple attributes enclosed in brackets
- We then identify the primary key and any alternate and/or foreign key(s) of the relation
- Following the identification of a foreign key, the relation containing the referenced primary key is given
- Any derived attributes are also listed together with how each one is calculated

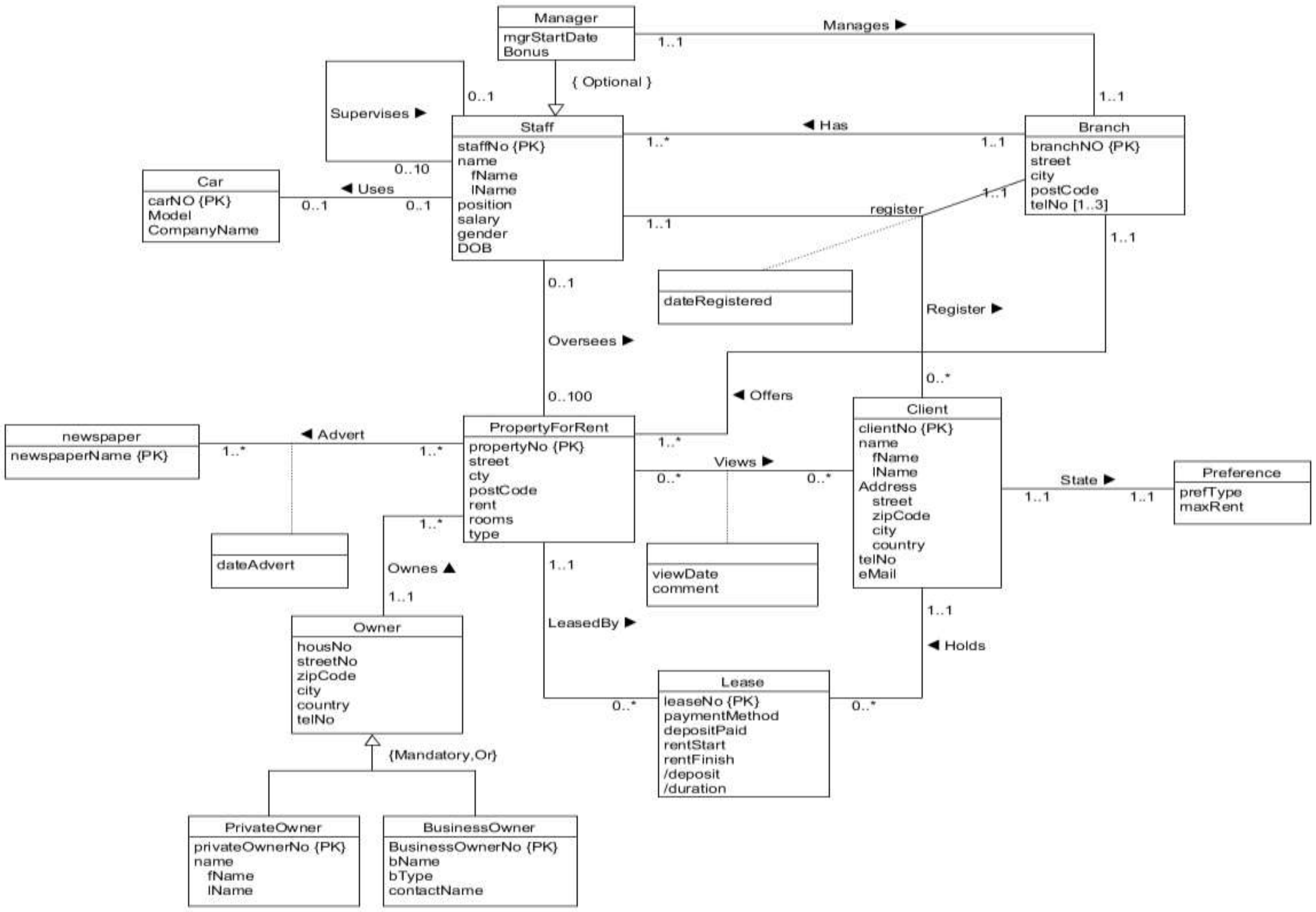
Step 2.1 Derive relations for logical data model

- The relationship that an entity has with another entity is represented by the primary key/foreign key mechanism
- In deciding where to post (or place) the foreign key attribute(s), we must first identify the ‘parent’ and ‘child’ entities involved in the relationship
- The parent entity refers to the entity that posts a copy of its primary key into the relation that represents the child entity, to act as the foreign key

Step 2.1 Derive relations for logical data model

- We describe how relations are derived for the following structures that may occur in a conceptual data model:
- (1) strong entity types;
- (2) weak entity types;
- (3) one-to-many (1:*) binary relationship types;
- (4) one-to-one (1:1) binary relationship types;
- (5) one-to-one (1:1) recursive relationship types;
- (6) superclass/subclass relationship types;
- (7) many-to-many (*:*) binary relationship types;
- (8) multi-valued attributes.

Conceptual Data Model of all Users' Views



Step 2.1 Derive relations for logical data model

- **(1) Strong entity types**
  - For each strong entity in data model,
    - Create **relation** that includes all simple attributes of entity
    - For **composite attributes**, include **only constituent simple** attributes: fName and lName in the relation

Staff (staffNo, fName, lName, position, gender, DOB)

Primary Key staffNo

Step 2.1 Derive relations for logical data model

(2) Weak entity types

– For each weak entity in data model,
  - Create relation that includes all simple attributes of entity
  - Primary key of weak entity is partially or fully derived from each owner entity

Preference (prefType, maxRent)

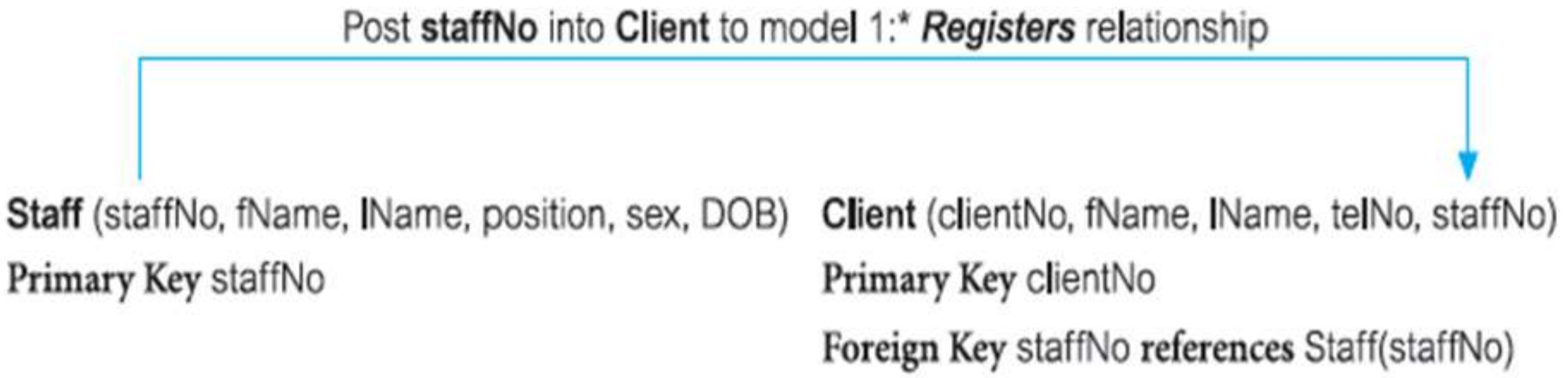
Primary Key None (at present)

Step 2.1 Derive relations for logical data model

- **(3) One-to-many (1:*) binary relationship types**
  - For each 1:* binary relationship,
    - Entity on ‘one side’ of relationship designated as parent entity
    - Entity on ‘many side’ designated as child entity
  - Represent relationship by posting copy of primary key attribute(s) of parent entity into relation representing child entity
    - Acts as foreign key

Step 2.1 Derive relations for logical data model

- **(3) One-to-many (1:*) binary relationship types**



Post **staffNo** into **Client** to model 1:* ***Registers*** relationship

Staff (staffNo, fName, lName, position, sex, DOB)
Primary Key staffNo

Client (clientNo, fName, lName, telNo, staffNo)
Primary Key clientNo
Foreign Key staffNo **references** Staff(staffNo)

Step 2.1 Derive relations for logical data model

- **(4) One-to-one (1:1) binary relationship types**
  - **Cardinality** cannot be used to identify parent and child entities
  - **Participation constraints** are used instead to decide:
  - Options:
    1) Combine entities involved into **one relation**
    2) Create two relations and post copy of primary key from one relation to other
  - Consider the following:
    - *(a) mandatory* participation (1:1 vs 1:1)on *both* sides of 1:1 relationship
    - *(b) mandatory* participation (1:1 vs 0:1) on *one* side of 1:1 relationship
    - *(c) optional* participation (0:1 vs 0:1) on *both* sides of 1:1 relationship

Step 2.1 Derive relations for logical data model

(a) *Mandatory* participation (1:1 vs 1:1) on *both* sides of 1:1 relationship

– Combine entities involved into 1 relation
– Choose one primary key of original entities to be primary key of new relation
– Other primary key (if one exists) used as alternate key

Step 2.1 Derive relations for logical data model

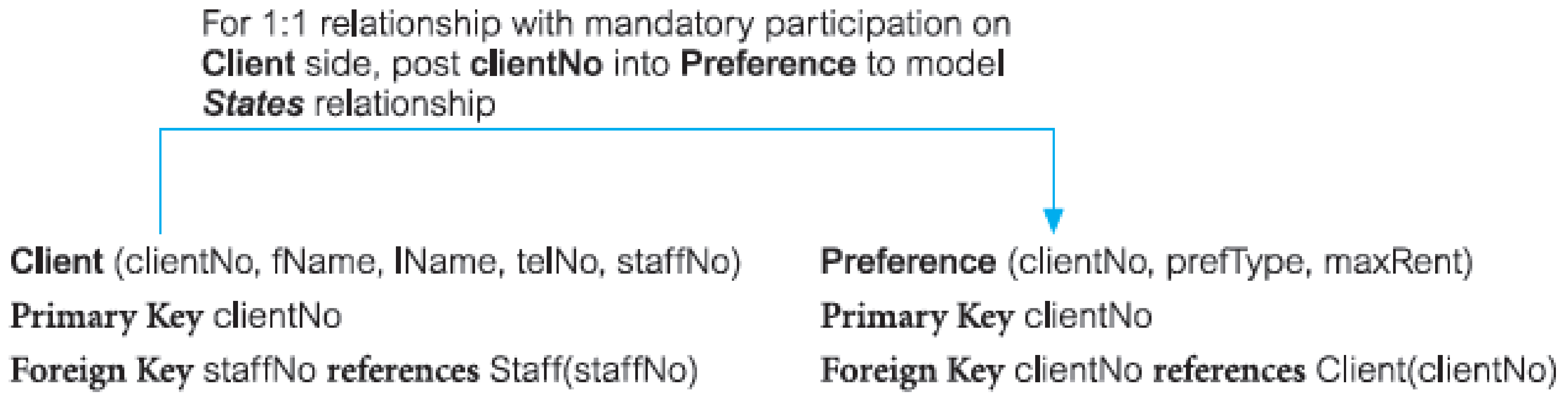
(a) *Mandatory* participation (1:1 vs 1:1) on *both* sides of 1:1 relationship example:

- The **Client *States* Preference relationship** is an example of a 1:1 relationship with mandatory participation on both sides

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 (1:1 vs 0:1) on** ***one*** **side of a 1:1 relationship**
  - Identify parent and child entities using participation constraints (assuming not every client specifies preferences)
  - Entity with optional participation side in relationship designated as **child**
  - Entity with mandatory participation designated as **parent**
  - Place copy of primary key of parent in relation representing child



For 1:1 relationship with mandatory participation on **Client** side, post **clientNo** into **Preference** to model *States* relationship

Client (clientNo, fName, lName, telNo, staffNo)
Primary Key clientNo
Foreign Key staffNo **references** Staff(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 (0:1 vs 0:1) on *both* sides of a 1:1 relationship**
  - Designation of parent and child entities arbitrary
  - ‘Staff Uses Car’ Example

Step 2.1 Derive relations for logical data model

- **(5) one-to-one (1:1) recursive relationship types;**
- For a 1:1 recursive relationship, follow the rules for participation as described previously for a 1:1 relationship
- However, in this special case of a 1:1 relationship, the entity on **both sides** of the relationship is the **same**
- For a 1:1 recursive relationship with **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 to indicate the relationship it represents e.g. manager_id

Step 2.1 Derive relations for logical data model

- **(5) one-to-one (1:1) recursive relationship types;**
- For a 1:1 recursive relationship with mandatory participation on only one side, we have the option to create a single relation with two copies of the primary key as described previously,

  OR
- to create a new relation to represent the relationship
- The new relation would have only two attributes, both copies of the primary key
- For a 1:1 recursive relationship with optional participation on both sides, again create a new relation as described earlier

Step 2.1 Derive relations for logical data model

- **(6) superclass/subclass relationship types;**

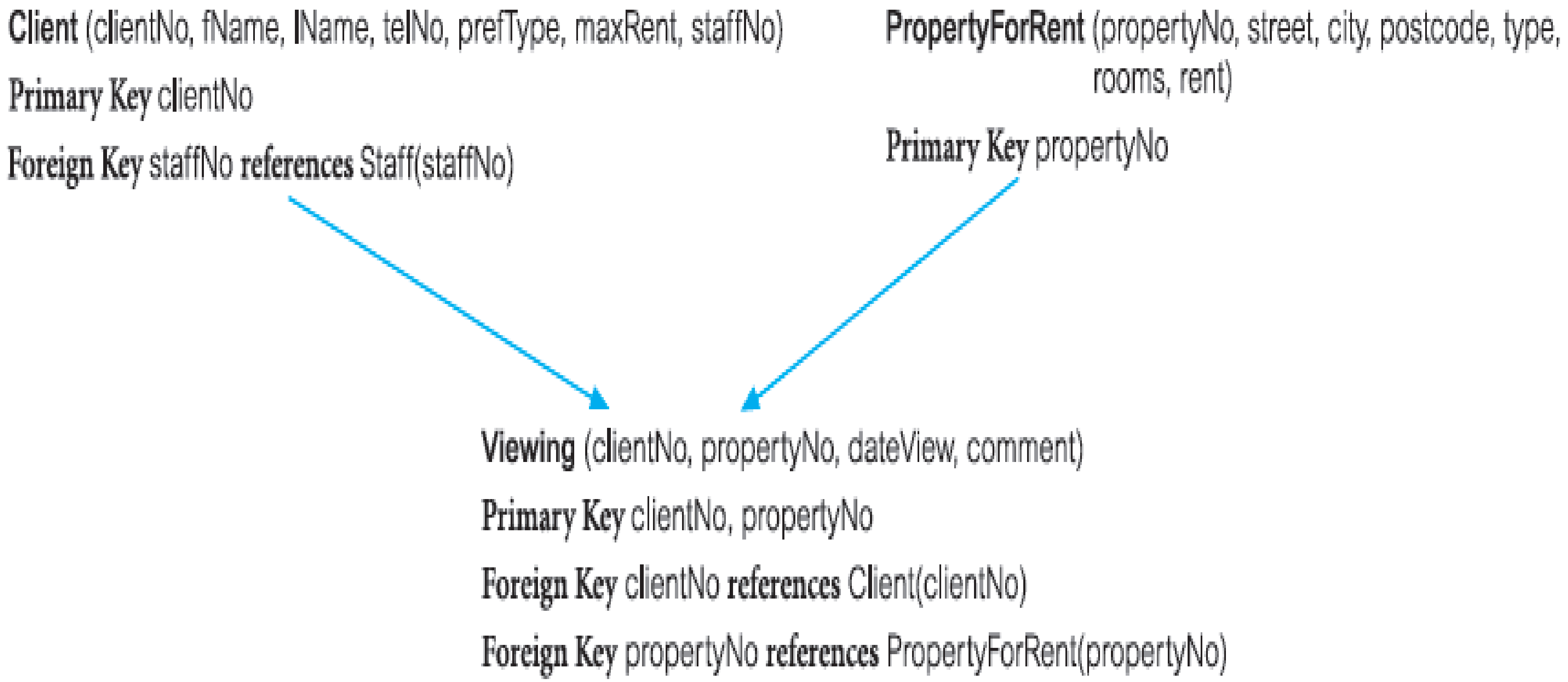
  As described earlier in case of EERM

Step 2.1 Derive relations for logical data model

- **(7) Many-to-many (*:*) binary relationship types**
  - For each *:* binary relationship **create a relation to represent relationship (associative entity) and include any attributes that are part of relationship**
  - Post copy of primary key attribute(s) of entities that participate in relationship into new relation - act as foreign keys
  - Foreign keys also form primary key of new relation
    - Possibly in combination with other attributes of relationship

Step 2.1 Derive relations for logical data model

(7) Many-to-many (*:*) binary relationship types



Client (clientNo, fName, lName, telNo, prefType, maxRent, staffNo)
Primary Key clientNo
Foreign Key staffNo **references** Staff(staffNo)

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

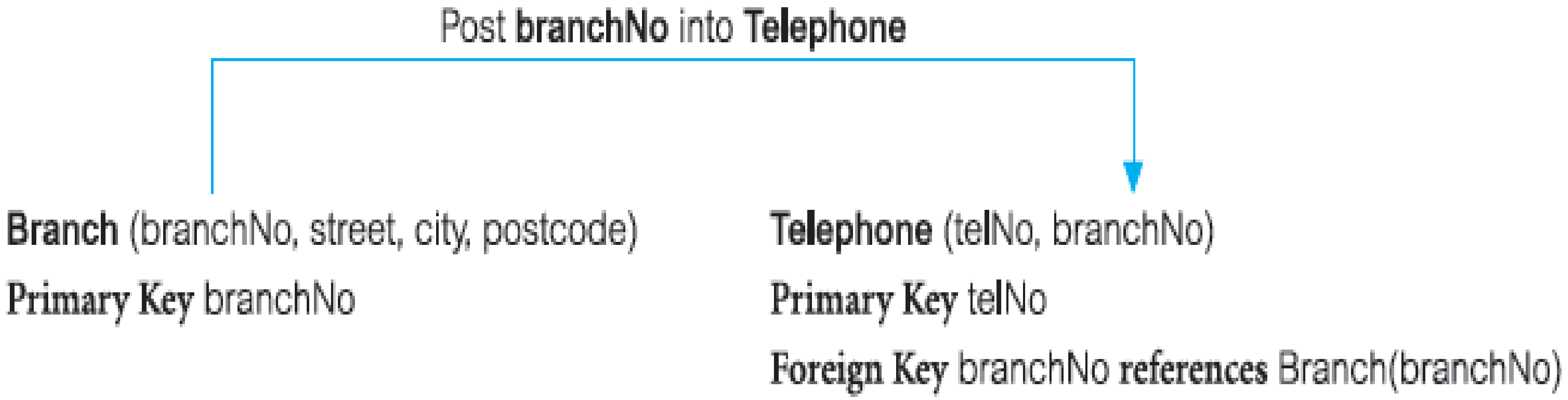
Viewing (clientNo, propertyNo, dateView, comment)
Primary Key clientNo, propertyNo
Foreign Key clientNo **references** Client(clientNo)
Foreign Key propertyNo **references** PropertyForRent(propertyNo)

Step 2.1 Derive relations for logical data model

- **(8) Multi-valued attributes**
  - Create new relation to represent multi-valued attribute
  - Include primary key of main entity in new relation - acts as 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 main entity

Step 2.1 Derive relations for logical data model

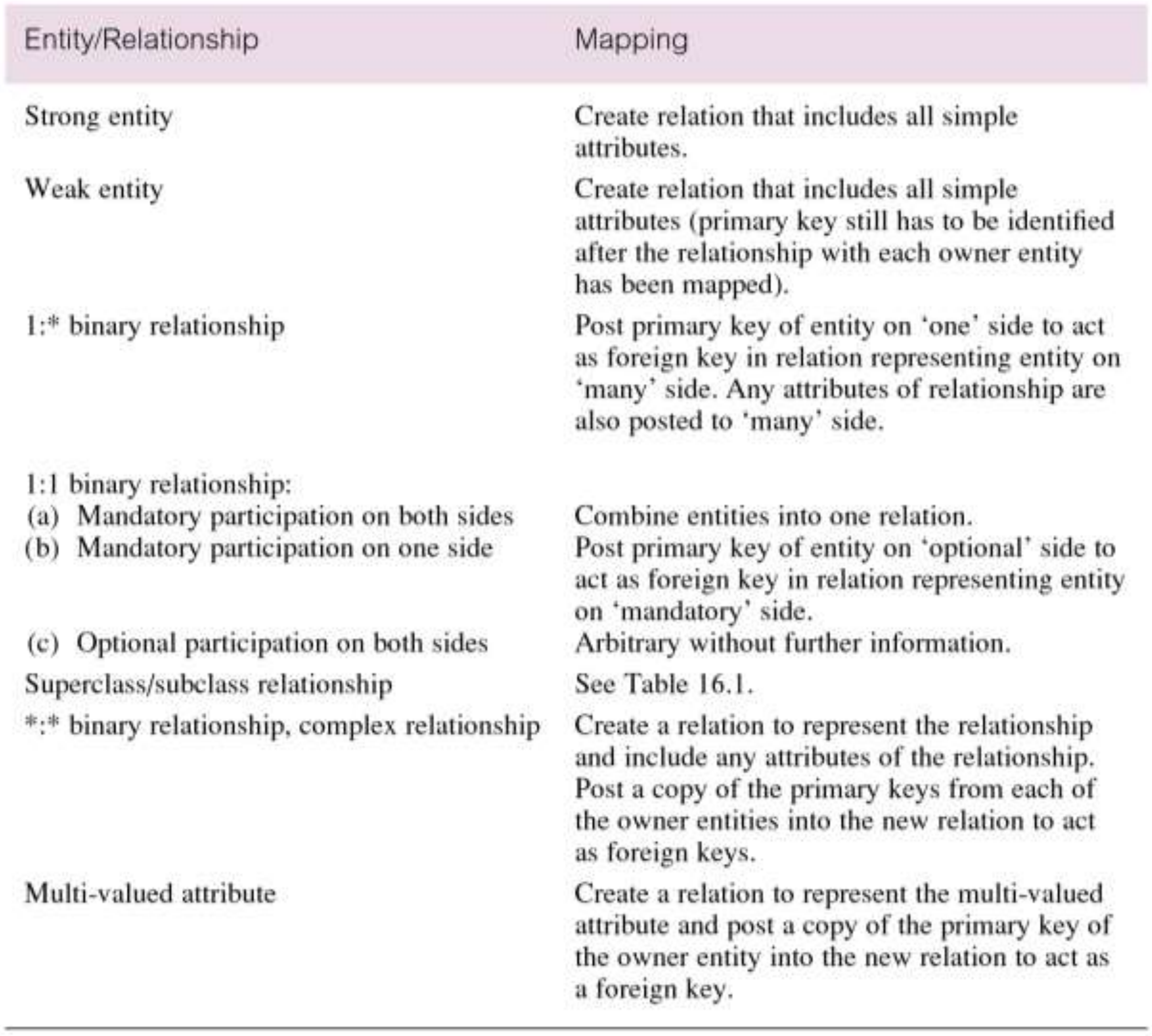
(8) Multi-valued attributes


Post **branchNo** into **Telephone**

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

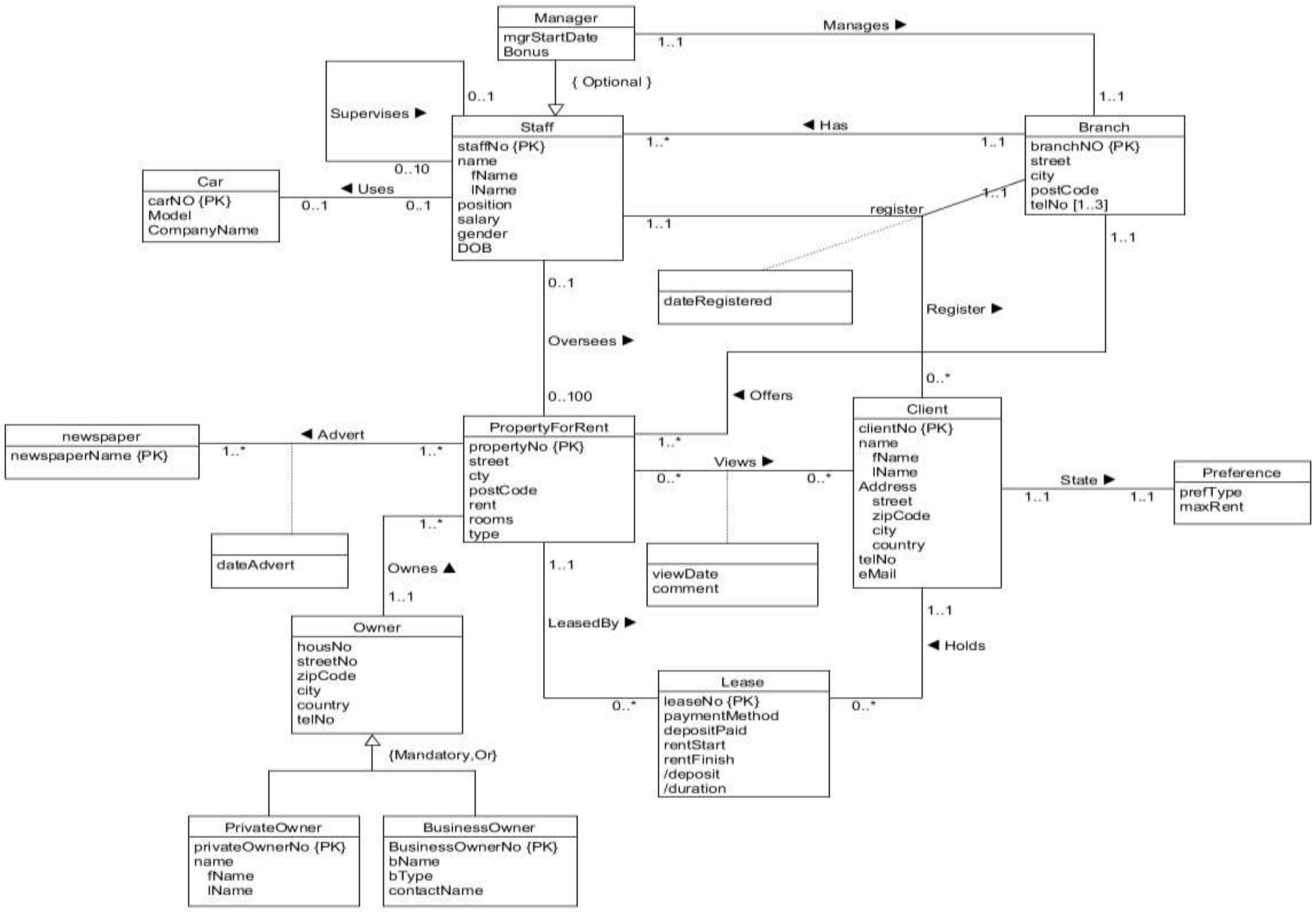


Entity/Relationship	Mapping
Strong entity	Create relation that includes all simple attributes.
Weak entity	Create relation that includes all simple attributes (primary key still has to be identified after the relationship with each owner entity has been mapped).
1:* binary relationship	Post primary key 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.

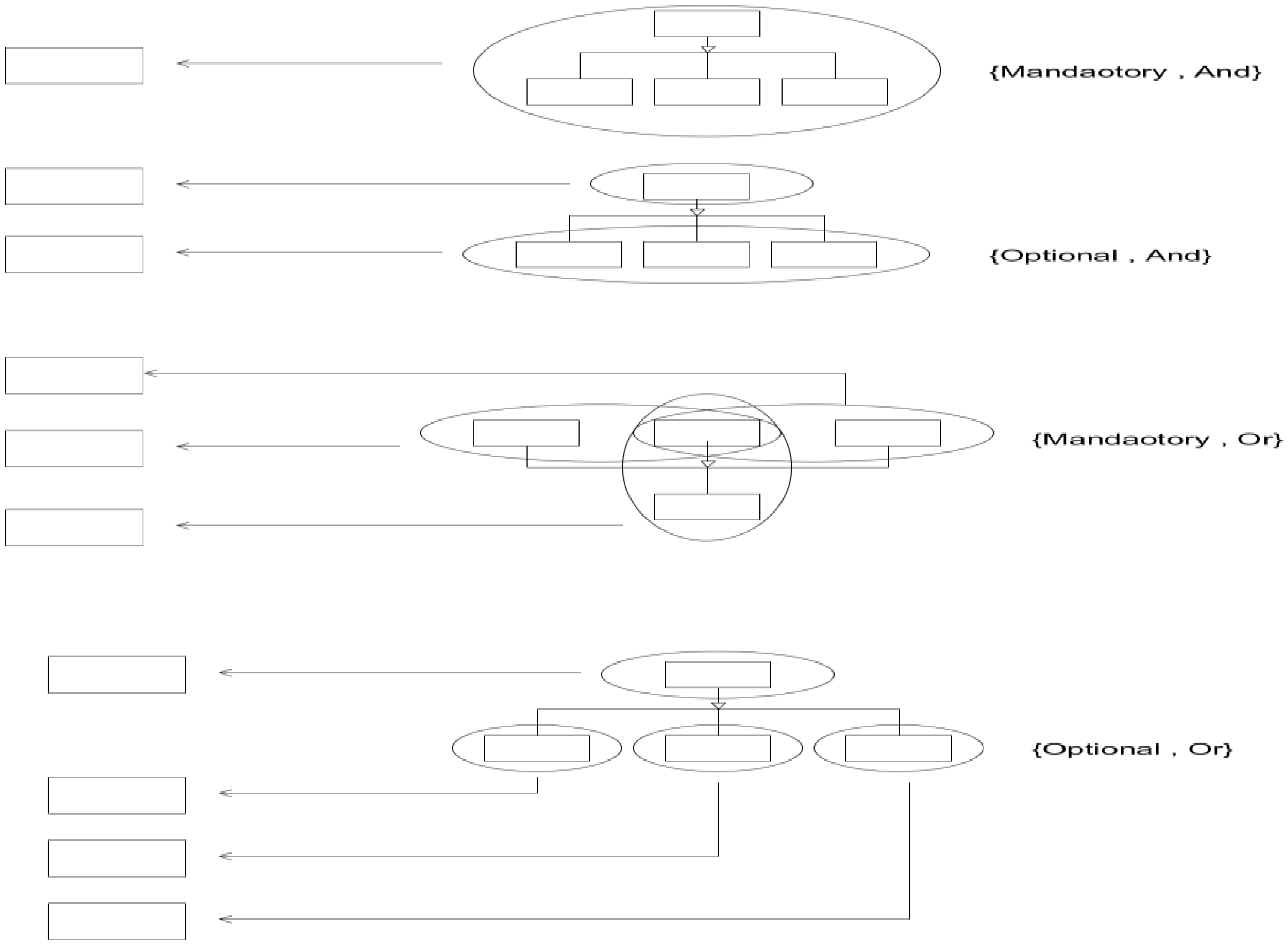
Step 2.1 Derive relations for logical data model

- **Document relations and foreign key attributes**
  - At the end of Step 2.1, document the composition of the relations derived for the logical data model using the DBDL
  - Double check all primary and foreign keys again

Conceptual Data Model of all Users' Views



Creating tables to represent specialization/generalization



Relations for the *DreamHome* example (Relational Schema)

Manager (staffNo, mgrStartDate, Bonus) ***Primary Key*** staffNo ***Foreign Key*** staffNo references Staff(staffNo)	**Branch** (branchNo,street,city,postcode,mgrNo) ***Primary Key*** branchNo ***Foreign Key*** mgrNo references Manger(staffNo)
Telephone (telNO, branchNo) ***Primary Key*** telNO ***Foreign Key*** branchNo references Branch(branchNo)	**Staff** (staffNo, fName, lName, position, gender, DOB, supervisorNo, branchNo) ***Primary Key*** staffNo ***Foreign Key*** supervisorNo references Staff(staffNo) ***Foreign Key*** branchNo references Branch(branchNo)
Car (carNo, model, companyName ,staffNo) ***Primary Key*** carNo ***Foreign Key*** staffNo references Staff(staffNo)	**Client** (clientNo, fName, lName, telNo, prefType, maxRent, housNo, streetNo, zipCode, city, country, staffNo) ***Primary Key*** clientNo ***Foreign Key*** staffNo references Staff(staffNo)
Viewing (clientNo, propertyNo, dateView, comment) ***Primary Key*** clientNo, propertyNo ***Foreign Key*** clientNo references Client(clientNo) ***Foreign Key*** propertyNo references PropertyForRent(propertyNo)	**Registration** (clientNo, branchNo, staffNo, dateRegistered) ***Primary Key*** clientNo, branchNo, staffNo, dateRegistered ***Foreign Key*** clientNo references Client(clientNo) ***Foreign Key*** branchNo references Branch(branchNo) ***Foreign Key*** staffNo references Staff(staffNo)
PropertyForRent (propertyNo, street, city, postcode, rent, room, type, privateOwnerNo, buisnessOwnerNo, staffNo, branchNo) ***Primary Key*** propertyNo ***Foreign Key*** privateOwnerNo references PrivateOwner(ownerNo) ***Foreign Key*** buisnessOwnerNo references BuisnessOwner(buisnessOwnerNo) ***Foreign Key*** staffNo references Staff(staffNo) ***Foreign Key*** branchNo references Branch(branchNo)	**Lease** (leaseNo, paymentMethod, depositPaid, rentStart, rentFinsh, deposit, duration, clientNo, propertyNo) ***Primary Key*** leaseNo ***Alternate Key*** propertyNo, rentStart ***Alternate Key*** clientNo, rentStart ***Foreign Key*** clientNo references Client(clientNo) ***Foreign Key*** propertyNo references PropertyForRent(propertyNo) ***Derived*** deposit (propertyForRent.rent * 2) ***Derived*** duration (rentFinish - rentStart)
PrivateOwner (privateOwnerNo, fName, lName, housNo, streetNo, zipCode, city, country, telNo) ***Primary Key*** privateOwnerNo ***Alternate Key*** telNo	**BusinessOwner** (businessOwnerNo, bName, bType, contactName, housNo, streetNo, zipCode, city, country, telNo) ***Primary Key*** businessOwnerNo ***Alternate Key*** bName ***Alternate Key*** telNo
Advert (propertyNo, newspaperName, advertDate) ***Primary Key*** propertyNo, newspaperName, advertDate ***Foreign Key*** propertyNo references PropertyForRent(propertyNo) ***Foreign Key*** newspaperName references Newspaper(newspaperName)	**Newspaper** (newspaperName) ***Primary Key*** newspaperName

Step 2.2

- **Step 2.2 Validate relations using normalization**
  - **To validate relations in logical data model using normalization**
  - **Typically already in 3NF at this point**

Step 2.3 Check integrity constraints

- To check integrity constraints represented in logical data model
  - Keep database accurate, consistent, complete
- Identify:
    - Required data
    - Unique data
    - Attribute domain constraints
    - Entity integrity
    - Referential integrity

Step 2.3 Check integrity constraints

- **Required data**
  - **Non null attributes contain valid value**
- **Unique data**
  - **Some attributes values are unique**
- **Attribute domain constraints**
  - **Every attribute has domain**

Step 2.3 Check integrity constraints

- Entity integrity
  - Primary key not null
- Referential integrity
  - Foreign key must reference existing value in parent relation
  - Foreign key null if participation optional
  - Existence constraints

Step 2.3 Check integrity constraints

- **Existence constraints**
  - **Delete tuple from child relation**
    - Referential integrity is unaffected
  - **Insert tuple into child relation**
    - Foreign key value will be either set to null or to a value from the primary key attribute of parent relation
    - Automatically maintained by foreign key constraint
  - **Update foreign key of child tuple**
    - Foreign key value will be either set to null or to a value from the primary key attribute of parent relation
    - Automatically maintained by foreign key constraint

Step 2.3 Check integrity constraints

- **Existence constraints**
  - **Delete tuple from parent relation**
    - Referential integrity is lost, if there exists a tuple in child relation referencing the deleted tuple of parent relation
    - Several strategies:
      - ON DELETE NO ACTION
        - » Prevents a deletion from the parent relation
        - » Safe option
      - ON DELETE CASCADE
        - » Automatically delete all related tuples in child relation
        - » Dangerous option
      - ON DELETE SET NULL
        - » Sets to null the related foreign key values in child relation
      - ON DELETE SET DEFAULT
        - » Sets to default the related foreign key values in child relation

Step 2.3 Check integrity constraints

- **Existence constraints**
  - **Insert tuple into parent relation**
    - Referential integrity is unaffected
  - **Update primary key of parent tuple**
    - Referential integrity is lost, if there exists a tuple in child relation referencing the updated tuple of parent relation
    - Strategy:
      - ON UPDATE CASCADE (Typical option)

Document all integrity constraints

- Document all integrity constraints in the data dictionary for consideration during physical design.

Existence constraints for relations of *DreamHome*

Staff (staffNo, fName, lName, position, gender, DOB, salary, managerNo, mgrStartDate)
Primary Key staffNo
Foreign Key managerNo **references** Staff(staffNo) ON UPDATE CASCADE ON DELETE SET NULL

Owner (ownerNo, fName, lName, telNo)
Primary Key ownerNo

PropertyForRent (propertyNo, address, street, city, postcode, type, rooms, rent, ownerNo, staffNo)
Primary Key propertyNo
Foreign Key ownerNo **references** Owner(ownerNo) ON UPDATE CASCADE ON DELETE NO ACTION
Foreign Key staffNo **references** Staff(staffNo) ON UPDATE CASCADE ON DELETE SET NULL

Client (clientNo, fName, lName, prefType, maxRent)
Primary Key clientNo
Foreign Key staffNo **references** Staff(staffNo) ON UPDATE CASCADE ON DELETE NO ACTION

ClientTelNo (clientNo, telNo)
Primary Key clientNo, telNo
Foreign Key clientNo **references** Client(clientNo)

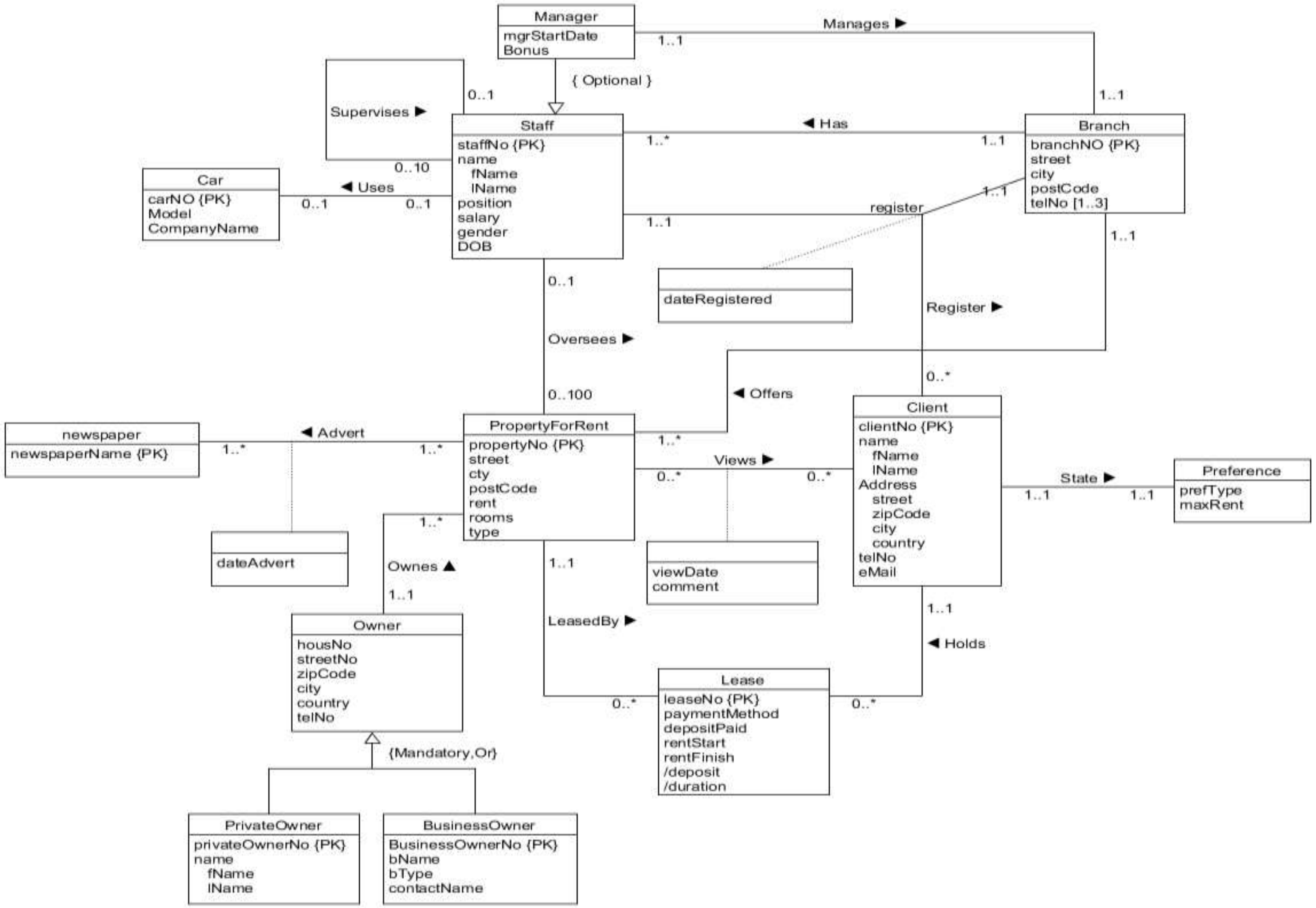
Existence constraints for relations of *DreamHome*

Advertise (propertyNo, newspaperName, dateAdvert, cost)
Primary Key propertyNo, newspaperName, dateAdvert
Foreign Key propertyNo **references** PropertyForRent(propertyNo) ON UPDATE CASCADE ON DELETE NO ACTION
Foreign Key newspaperName **references** Newspaper (newspaperName) ON UPDATE CASCADE ON DELETE NO ACTION
Newspaper (newspaperName, address, telNo, contactName)
Primary Key newspaperName
Alternate Key telNo

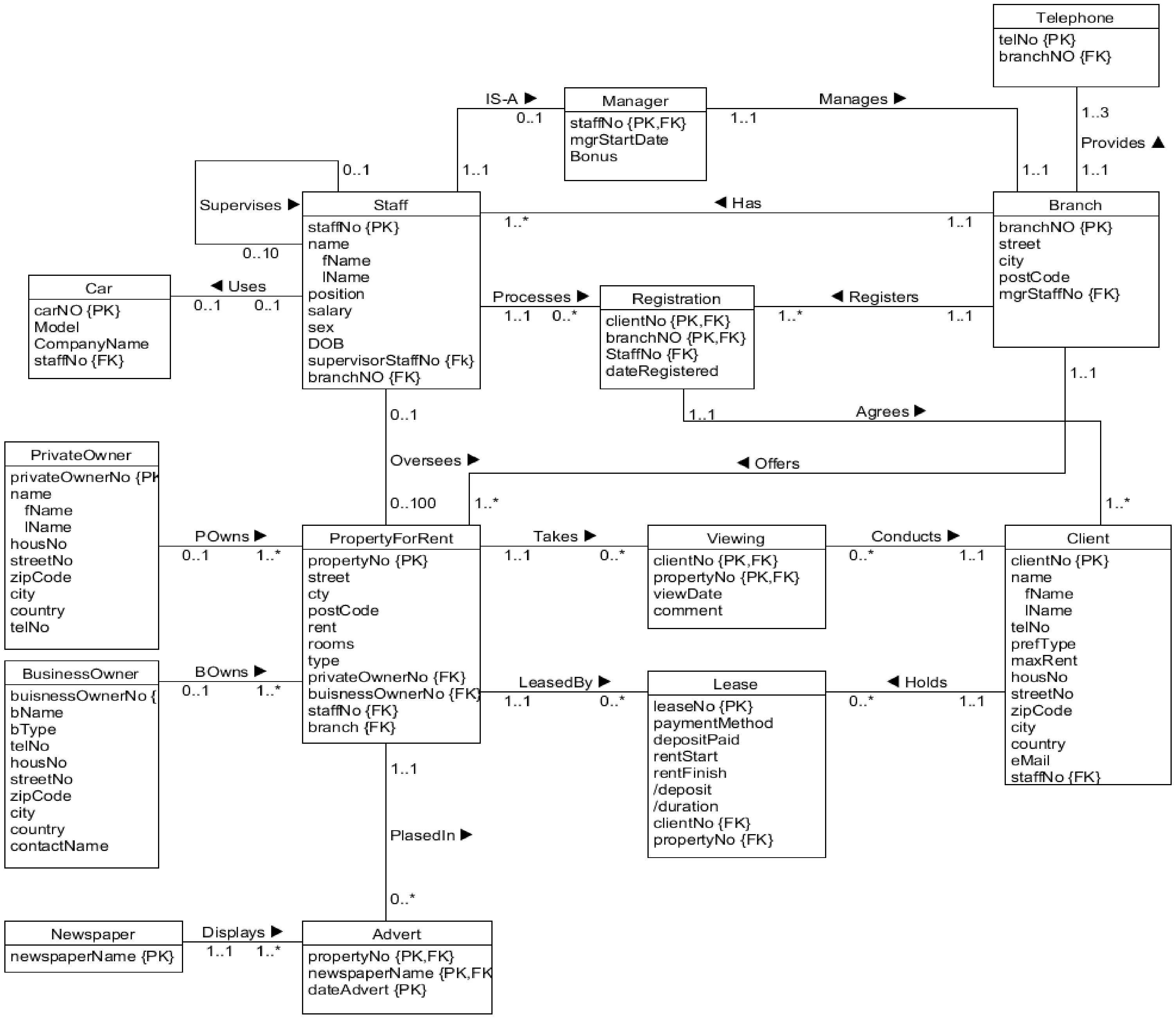
SignLease (leaseNo, signDate, paymentMethod, depositPaid, rentStart, rentFinish, deposit, duration, clientNo, propertyNo)
Primary Key leaseNo, signDate, clientNo, propertyNo
Foreign Key clientNo **references** Client(clientNo) ON UPDATE CASCADE ON DELETE NO ACTION
Foreign Key propertyNo **references** PropertyForRent(propertyNo) ON UPDATE CASCADE ON DELETE NO ACTION
Derived deposit **(**PropertyForRent.rent * 2)
Derived duration **(**rentFinish - rentStart)

Viewing (clientNo, propertyNo, viewDate, comment)
Primary Key clientNo, propertyNo
Foreign Key clientNo **references** Client(clientNo) ON UPDATE CASCADE ON DELETE NO ACTION
Foreign Key propertyNo **references** PropertyForRent(propertyNo) ON UPDATE CASCADE ON DELETE NO ACTION

Conceptual Data Model of all Users' Views



Build the Logical Data Model (Relation Diagram) of all Users’ Views for DreamHome



Chapter Summary

- **Logical database design** is the process of constructing a model of the data used in an enterprise based on a specific data model but independent of a particular DBMS and other physical considerations
- A **logical data model** includes *relational schema*, and supporting documentation such as the *data dictionary*, which is produced throughout the development of the model