SECD2523 DATABASE

TOPIC 6A | LOGICAL DATABASE DESIGN

Content adapted from Connolly, T., Begg, C., 2015. Database Systems: A Practical Approach to Design, Implementation, and Management, Global Edition. Pearson Education.

LECTURE LEARNING OUTCOME

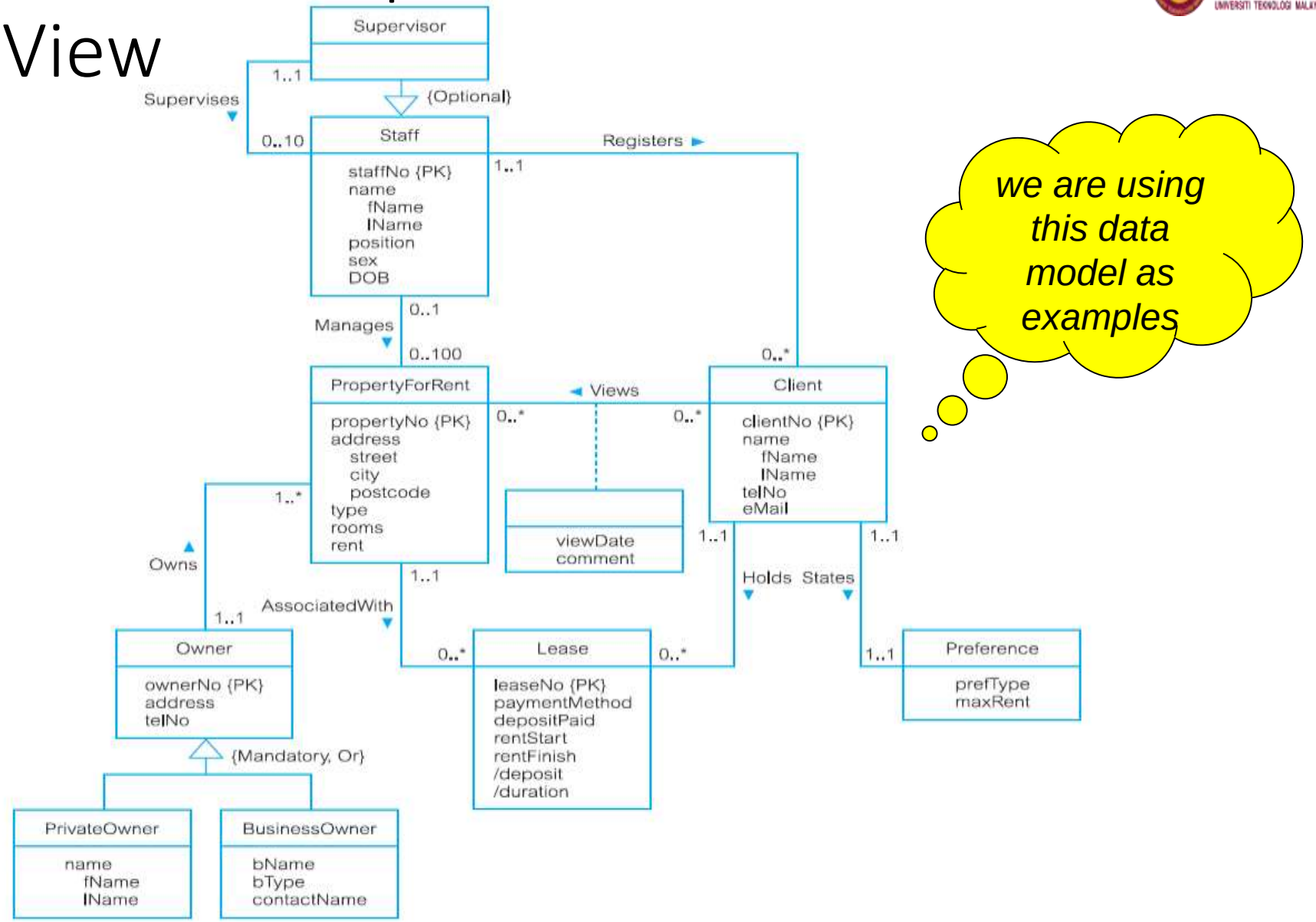
By the end of this lecture, students should be able to:

01	**Derive a logical data model from conceptual ERD.**
02	**Derive a set of relations from a local logical data model.**
03	**Validate these relations using the technique of normalization.**
04	**Validate a logical data model to ensure it supports required user transactions.**
05	**Merge local logical data models based on specific views into a global logical data model of the enterprise.**

Logical Database Design Phase:

- **METHODOLOGY REVIEW:**
  - Build and validate local logical data model
    - **Input**: Conceptual data model (e.g.: conceptual ERD) & Data dictionary
    - **Output**: Relational DB schemas (e.g.: the relations) & Logical data model (e.g.: Logical ERD)
  - Steps:
    1. **Derive relations for local logical data model**
    2. Validate relations using normalization
    3. Validate relations against user transactions
    4. Define integrity constraints
    5. Review local logical data model with user
    6. Merge logical data models into global model (optional)
    7. Check for future growth

Local conceptual data model: Staff View



we are using this data model as examples

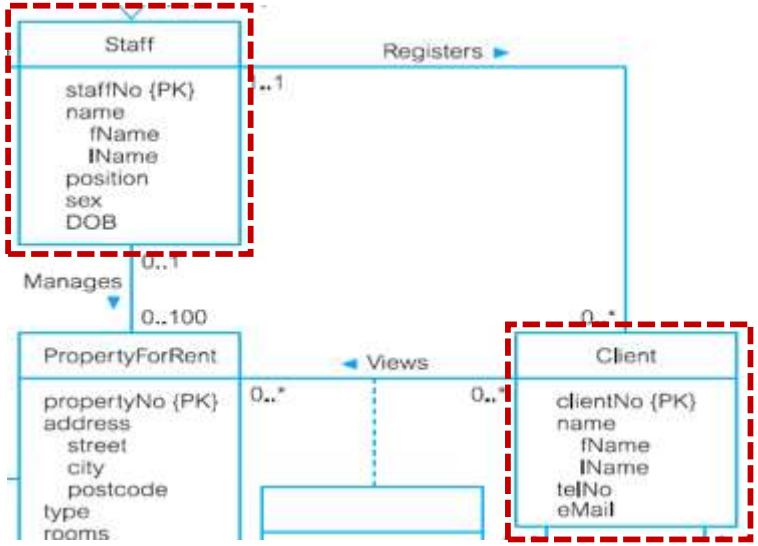
Figure 17.1 Conceptual data model for the StaffClient user views showing all attributes.

STEP 1:
Derive Relations for Local Logical Data Model

- Purpose:
  - **To create relations** for the local logical data model that represent the entities, relationships, and attributes identified in the conceptual data model (conceptual ERD – cERD).
  - Must consider following structure you may have in cERD:
    i. Strong entity type
    ii. Weak entity type
    iii. One-to-many (1:*) binary relationship types
    iv. One-to-one (1:1) binary relationship types
    v. Superclass/subclass relationship types
    vi. Many-to-many (*:*) binary relationship types
    vii. Recursive relationship types
    viii. Complex relationship types
    ix. Multi-valued attributes

STEP 1:
Derive Relations for Local Logical Data Model

- **Strong entity types**
  - Create a relation for each strong entity that includes all simple attributes of that entity.
  - For composite attributes, include only constituent simple attributes.



Portion of ERD from slide 4

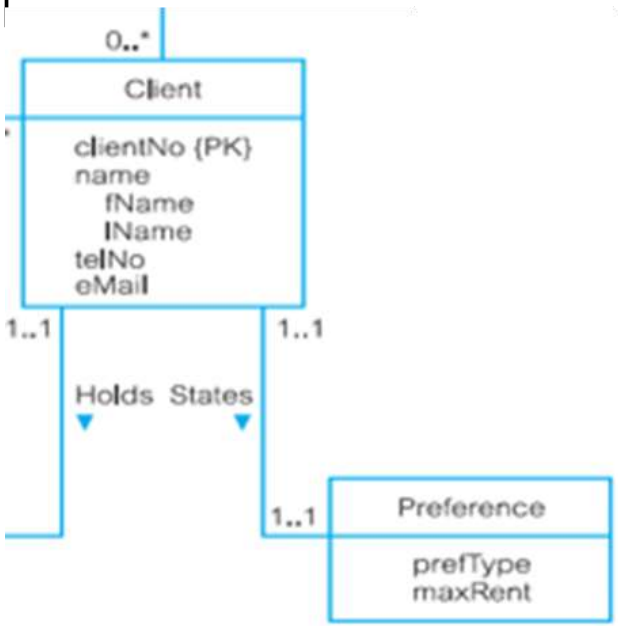
Strong entity (from slide 4): *** entity with PK identified*

•Staff, Client, PropertyForRent, Lease, Owner

STAFF (staffNo, fName, lName, position, sex, DOB)
CLIENT (clientNo, fName, lName, telNo)

STEP 1:
Derive Relations for Local Logical Data Model

- **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
    - Usually can be made after all relationship with owner entities have been mapped.



Example of weak entity:

- Preference

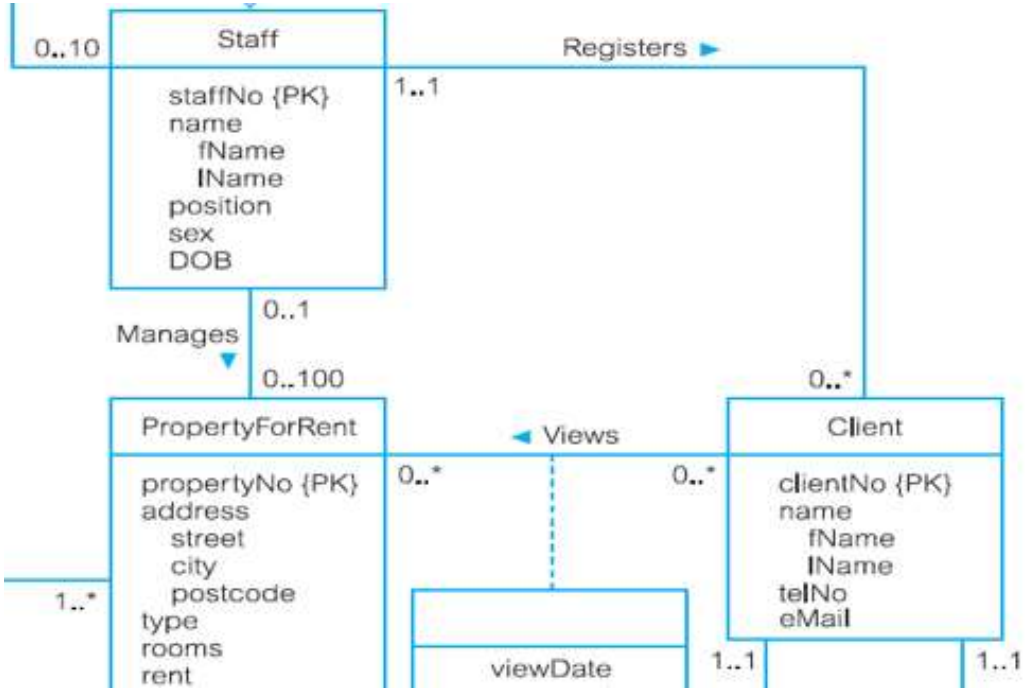
PREFERENCE (prefType, maxRent)

Primary Key : None (at present)

STEP 1:
Derive Relations for Local Logical Data Model

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

- Identify parent & child entity by using cardinality.
  - Entity on '**one**' side is designated as **parent** entity.
  - Entity on '**many**' side is the **child** entity.
- Copy the primary key attribute of parent entity into relation representing child entity, to act as a foreign key.



Relationship Registers between Staff and Client

Parent → Staff Child → Client

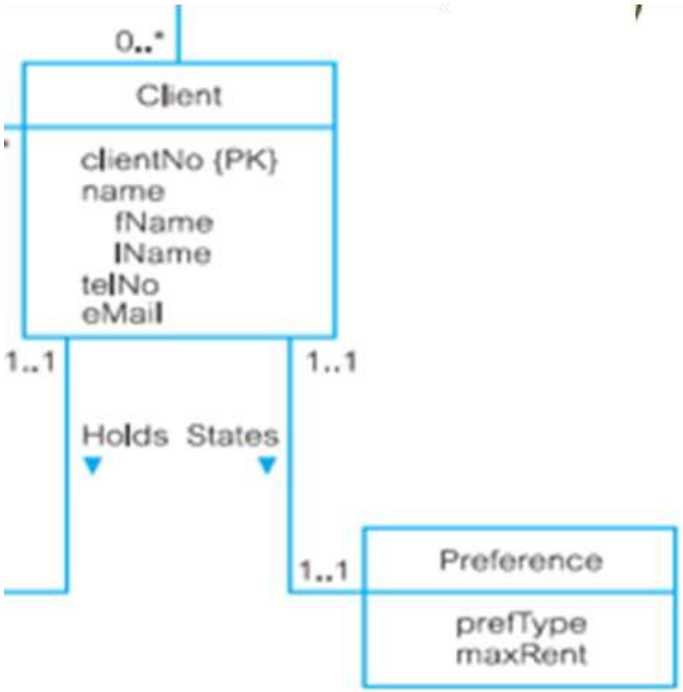
STAFF (staffNo, fName, lName, position, sex, DOB)
CLIENT (clientNo, fName, lName, telNo, staffNo)

STEP 1:
Derive Relations for Local Logical Data Model

One-to-one (1:1) binary relationship types

- More complex as cardinality cannot be used to identify parent and child entities in a relationship.
- Use **participation** to decide whether to combine entities to produce one relation or to create two relations and post copy of primary key from one relation to another.
- Consider the following participation constraint:
  a. Mandatory participation on both sides;
  b. Mandatory participation on one side, optional on the another;
  c. Optional participation on both sides;

STEP 1:
Derive Relations for Local Logical Data Model



One-to-one (1:1) binary relationship types

a) Mandatory participation on both sides

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

Relationship States between Client and Preference

→ combine Client & Preference

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

Fk: staffNo references Staff(staffNo)

*Notes: ** Relation PREFERENCE no longer exist*

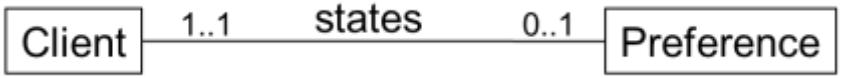
*** Relation CLIENT is renamed as CLIENT_PREF*

STEP 1:
Derive Relations for Local Logical Data Model

One-to-one (1:1) binary relationship types

b) Mandatory participation on one side, optional on another

- Entity with optional participation is the parent.
- Entity with mandatory participation is child.
- Create one relation for each entity, then copy PK of parent into child relation.



Client → optional → parent Preference → mandatory → child

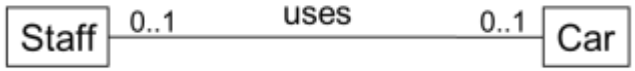
CLIENT (clientNo, fName, lName, telNo, staffNo)
FK: staffNo references Staff(staffNo)
PREFERENCE (clientNo, prefType, maxRent)
FK: clientNo references Client(clientNo)

STEP 1:
Derive Relations for Local Logical Data Model

One-to-one (1:1) binary relationship types

c) Optional participation on both sides

- Designation of the parent and child entities is arbitrary unless can find out more about the relationship.
- Must try to understand more about the relationship
  - Entity which is closer to being optional is the parent.
  - Entity which is closer to being mandatory is the child.
- Create a relation for each entity, then copy PK of parent into child.



- Assume majority of cars, but not all, are used by staff and only minority of staff use cars.
- Car entity, although optional, is closer to being mandatory than Staff entity.

Therefore, designate Staff as parent entity and Car as child entity.

STAFF (staffNo, sName)

CAR (carNo, staffNo)

STEP 1:
Derive Relations for Local Logical Data Model

Superclass/subclass relationship types

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

STEP 1:
Derive Relations for Local Logical Data Model

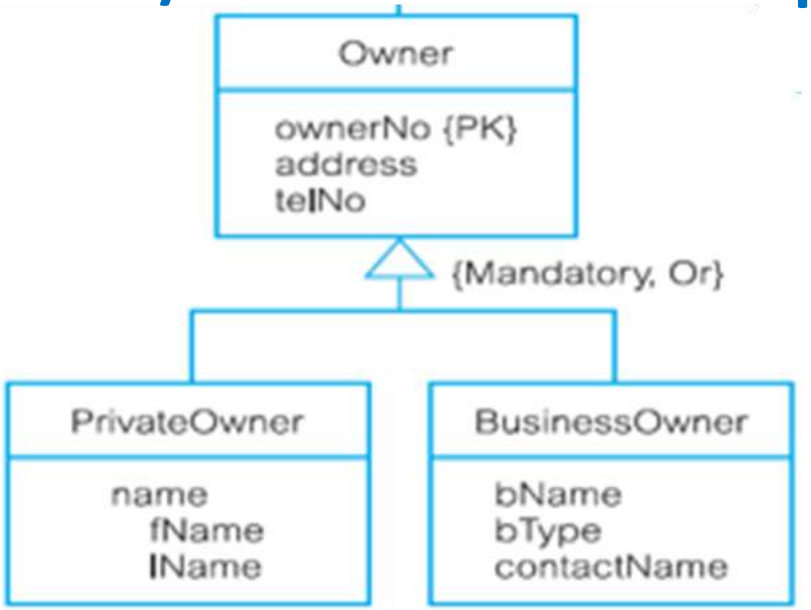
Superclass/subclass relationship types

TABLE 17.1 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

STEP 1:
Derive Relations for Local Logical Data Model

Superclass/subclass relationship types



Guideline No 3:

PRIVATEOWNER (ownerNo, fName, lName, address, telNo)
BUSINESSOWNER (ownerNo, bName, bType, contactName, address, telNo)

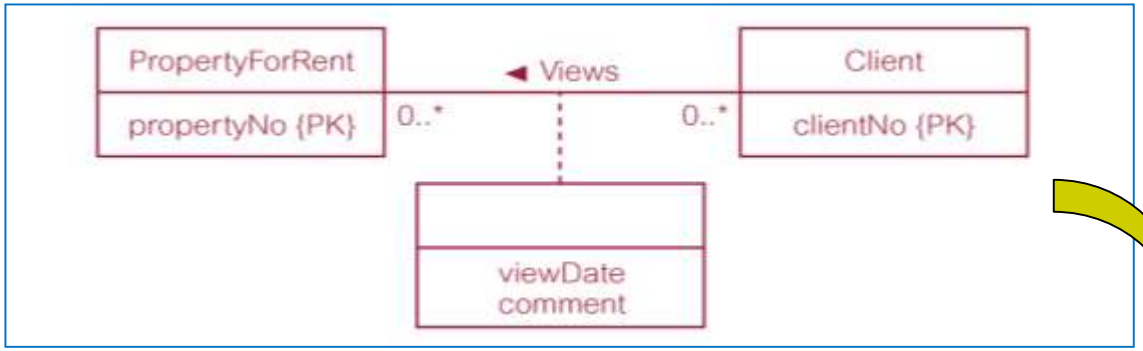
STEP 1:
Derive Relations for Local Logical Data Model

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

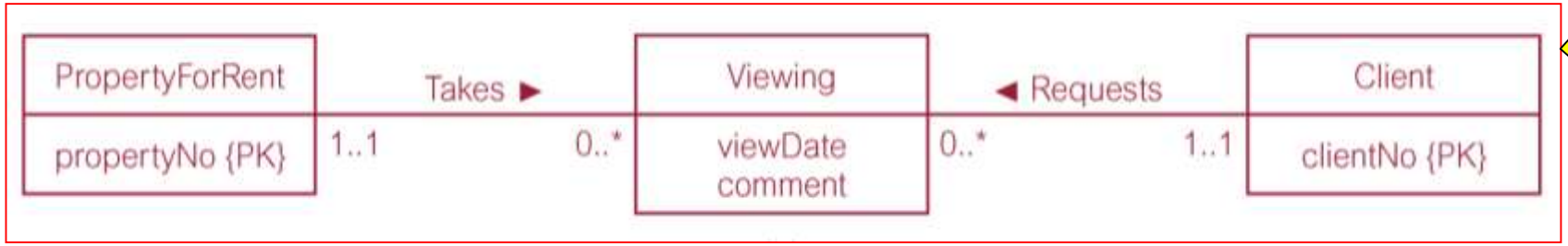
Option 1

- First, transform *:* relationships by creating two 1:* relationships
- Then, create relations from the transformed diagram

Original:



Transformed:



STEP 1:
Derive Relations for Local Logical Data Model

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

Option 1

- Relations:

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

  VIEWING (propertyNo, clientNo, viewDate, comment)

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

  Note: recall that relation CLIENT_PREF is the result from our earlier work on deriving relation for 1..1 binary relationship

Derive Relations for Local Logical Data Model

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

Option 2

- Create a **new relation** to represent the relationship and include any attributes that are part of relationship.
- Copy the primary key attribute(s) of the entities that participate in relationship into the new relation.
  - This attribute(s) is now acting as **foreign key** in the new relation.
  - This attribute (the FK) is also the primary key of new relation, possibly in combination with some of the attributes of the relationships

CLIENT_PREF (clientNo, fName, lName, telNo, prefType, maxRent, staffNo)
PROPERTYFORRENT (propertyNo, street, city, postcode, type, rooms, rent)

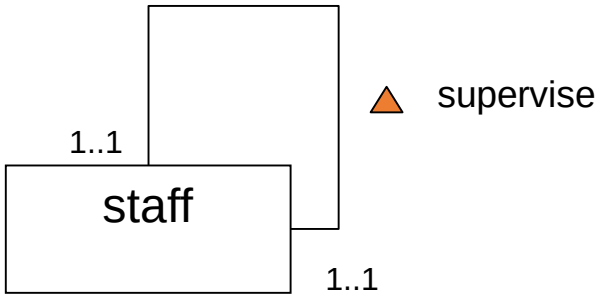
VIEWING (clientNo, propertyNo, viewDate, comment)

Derive Relations for Local Logical Data Model

Recursive relationship type

1:1 recursive relationships

- Use rules for participation for a 1:1 relationship
- **Mandatory participation on both sides:**
  - Represent 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 represent the relationship it presents.



SUPERVISION (staffNo, superviseeNo)
PK : staffNo
FK1: staffNo references Staff(staffNo)
FK2: superviseeNo references Staff(staffNo)

Derive Relations for Local Logical Data Model

Recursive relationship type

1:1 recursive relationships

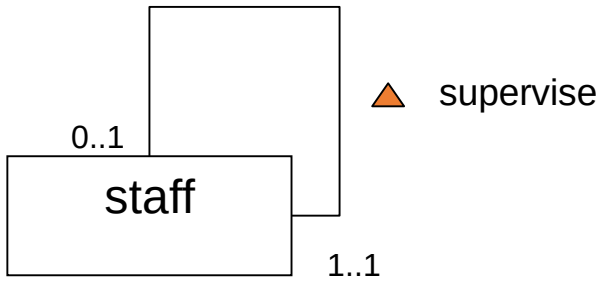
- **Mandatory participation on only one side:**
  - Option 1 → Create a single relation with two copies of primary key
  - Option 2 → Create a new relation to represent the relationship.
    The new relation would only have two attributes, both copies of the primary key.

Option 1

STAFF (staffNo, superviseeNo)
PK: staffNo
FK: superviseeNo reference STAFF(staffNo)

Option 2

SUPERVISE (staffNo, superviseeNo)
PK: staffNo
FK: superviseeNo reference STAFF(staffNo)

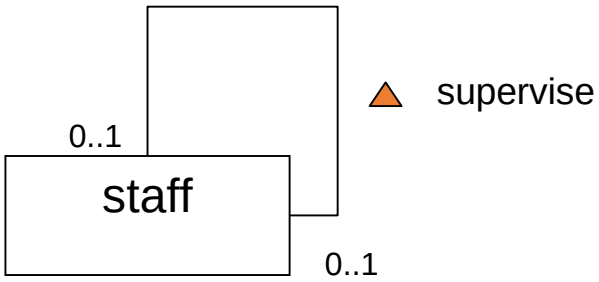


Derive Relations for Local Logical Data Model

Recursive relationship type

1:1 recursive relationships

- **Optional participation on both sides:**
  - Create a new relation as described above



SUPERVISE (staffNo, superviseeNo)
PK: staffNo
FK: superviseeNo reference STAFF(staffNo)

Recursive relationship type

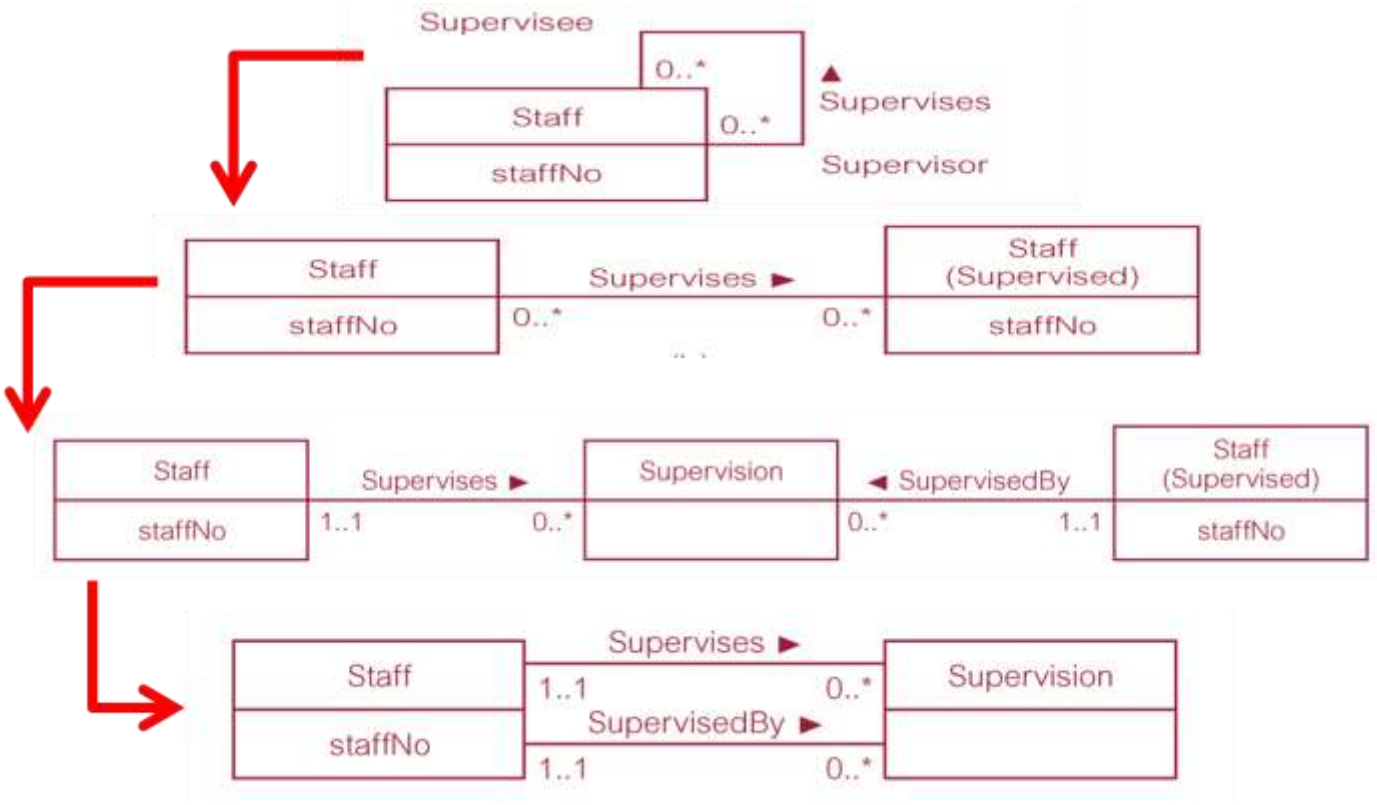
: recursive relationships

- Transform *:* recursive relationship into 1:* binary relationship
- Create relations to represent relationships from the transformed diagram, follow rules for participation 1:*
  - Entity on ‘one’ side is designated the parent entity and entity on ‘many’ side is the child entity.
  - Copy the primary key attribute(s) of parent entity into relation representing child entity, to act as a foreign key

Derive Relations for Local Logical Data Model

Recursive relationship type

: recursive relationships

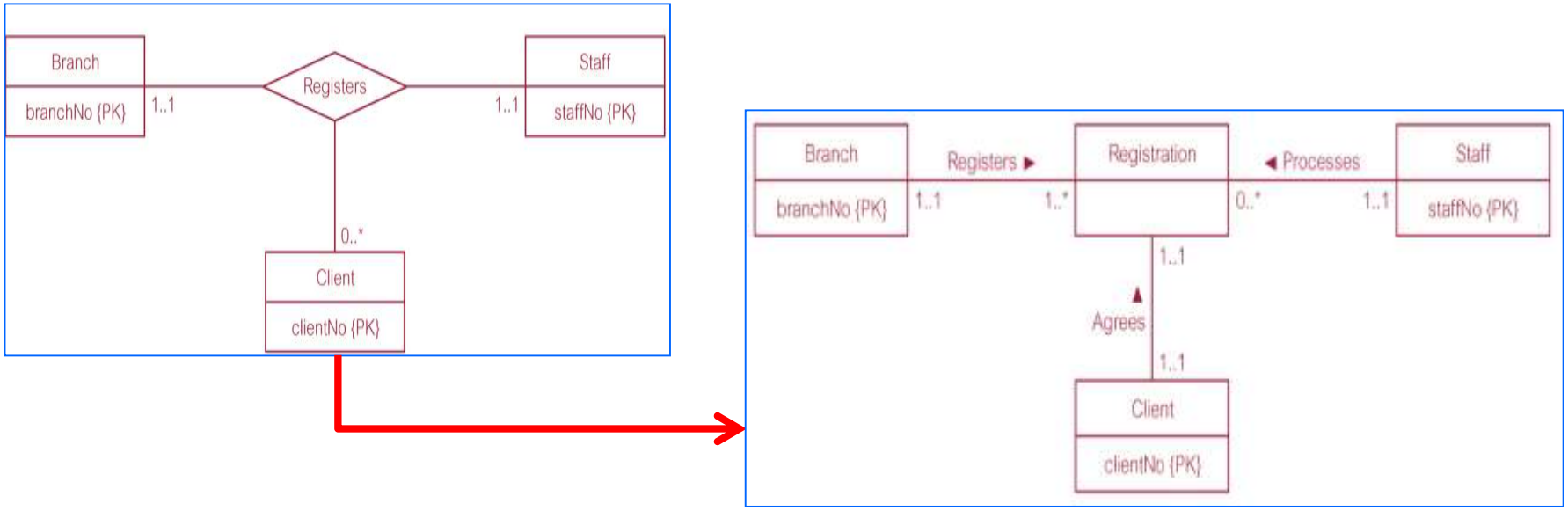


Derive Relations for Local Logical Data Model

Complex relationship types

Option 1:

- Transform *:* relationship by creating a new entity and binary relationships between original entities with this new entity.
- Create relations for all entities (follow appropriate rules for binary relationships)



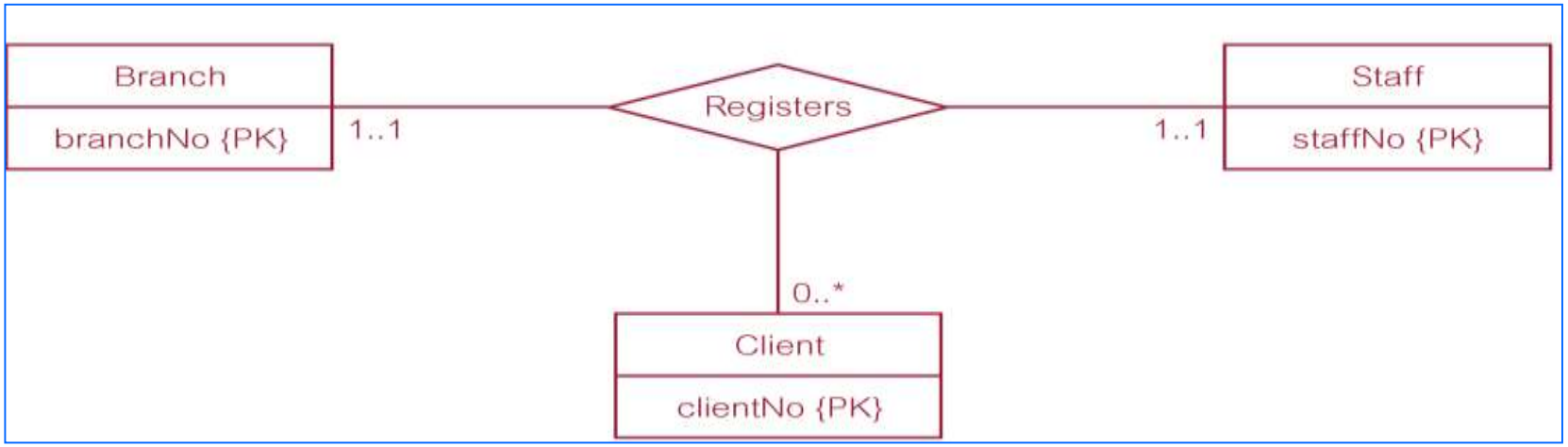
Complex relationship types

Option 2:

- Create relation to represent relationship and include any attributes that are part of the relationship.
- Copy 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 (for example, 1..*, 0..*) generally will also form the primary key of new relation, possibly in combination with some of the attributes of the relationship.

Complex relationship types

Option 2:



REGISTRATION (clientNo, branchNo, staffNo)

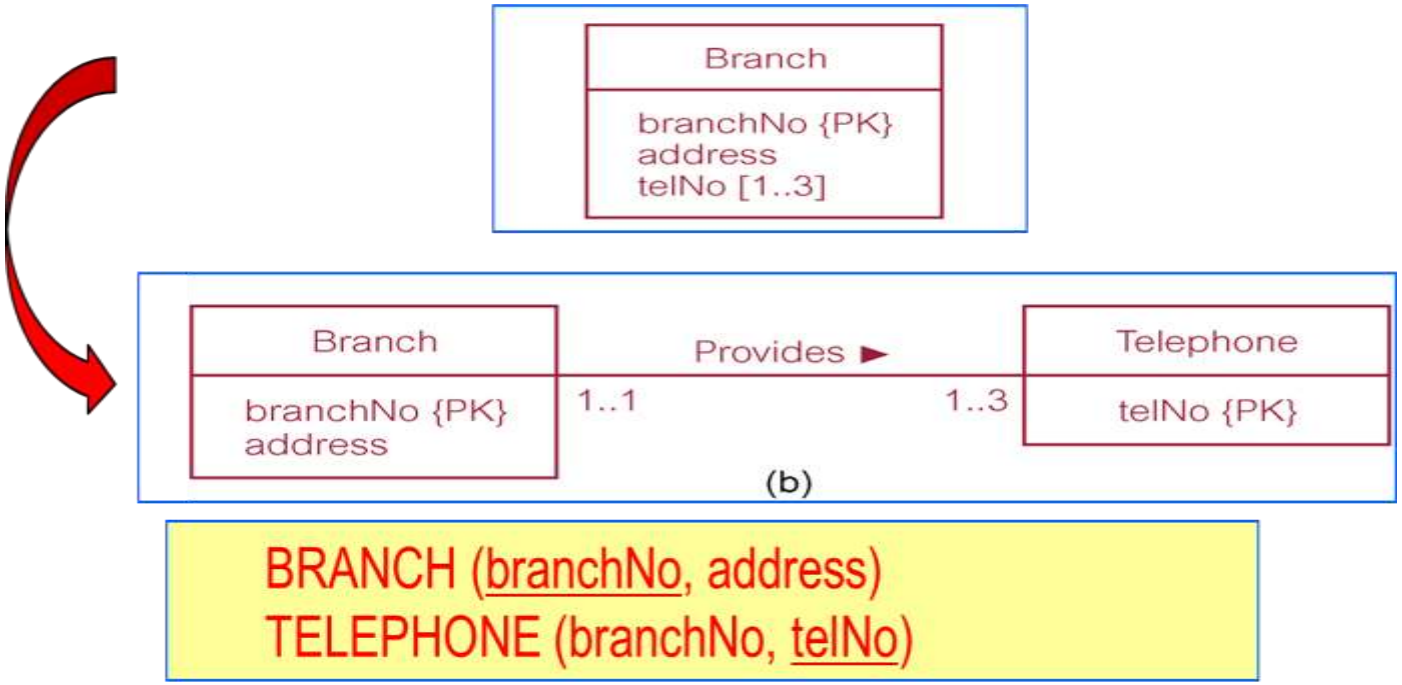
PK: clientNo
FK1: clientNo references CLIENT (clientNo)
FK2: branchNo references BRANCH (branchNo)
FK3: staffNo reference STAFF (staffNo)

Derive Relations for Local Logical Data Model

Multi-valued attributes

Option 1:

- Remove multi-valued attribute from original entity and put in new entity
- Create relationship between original entity to newly formed entity



Summary Derive Relation

TABLE 17.2 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 the "one" side to act as foreign key in relation representing entity on the "many" side. Any attributes of relationship are also posted to the "many" side.
1:1 binary relationship: (a) Mandatory participation on both sides (b) Mandatory participation on one side (c) Optional participation on both sides	 Combine entities into one relation. Post primary key of entity on the "optional" side to act as foreign key in relation representing entity on the "mandatory" side. Arbitrary without further information.
Superclass/subclass relationship	See Table 17.1.
: 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.

Local conceptual data model: Staff View

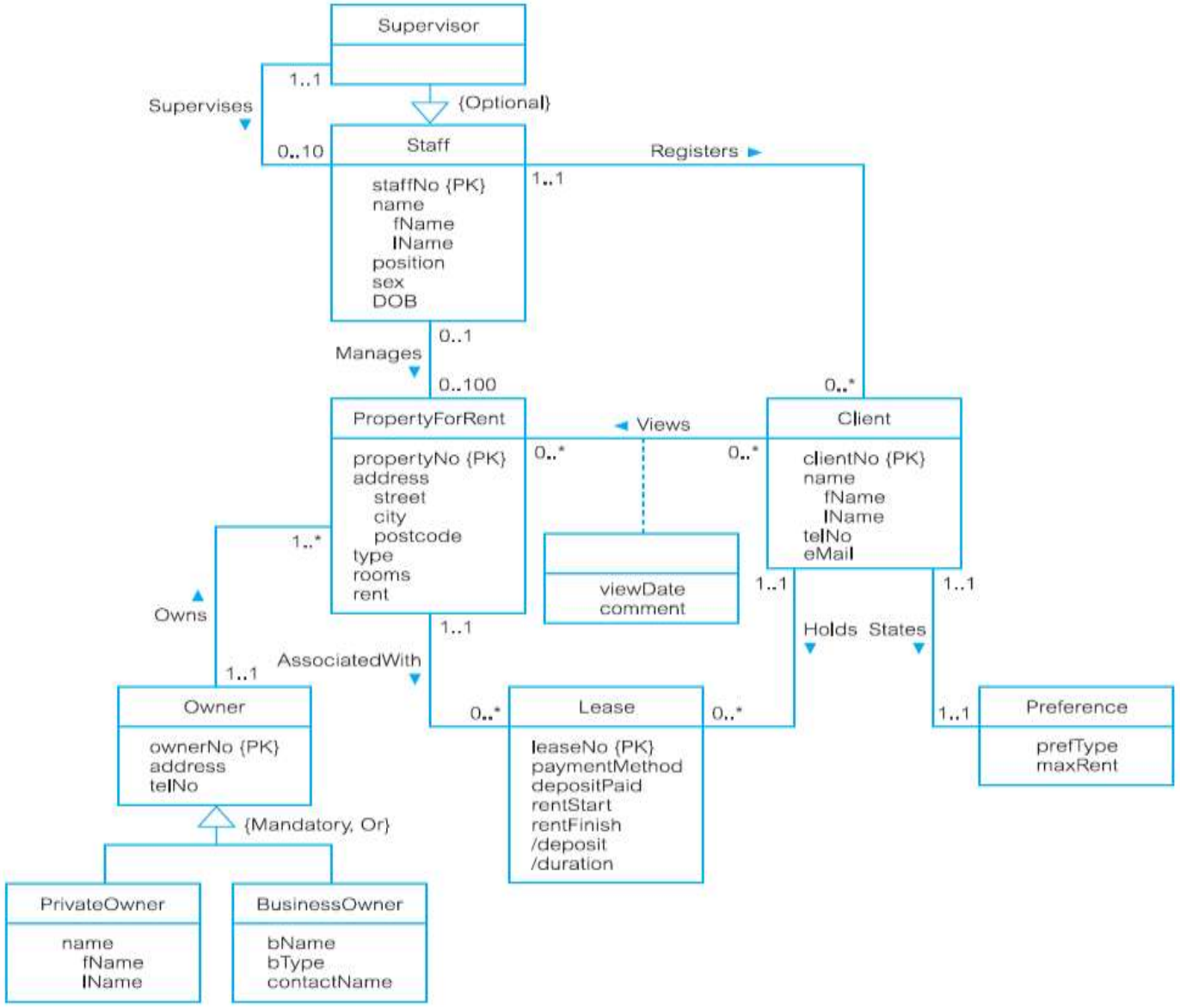


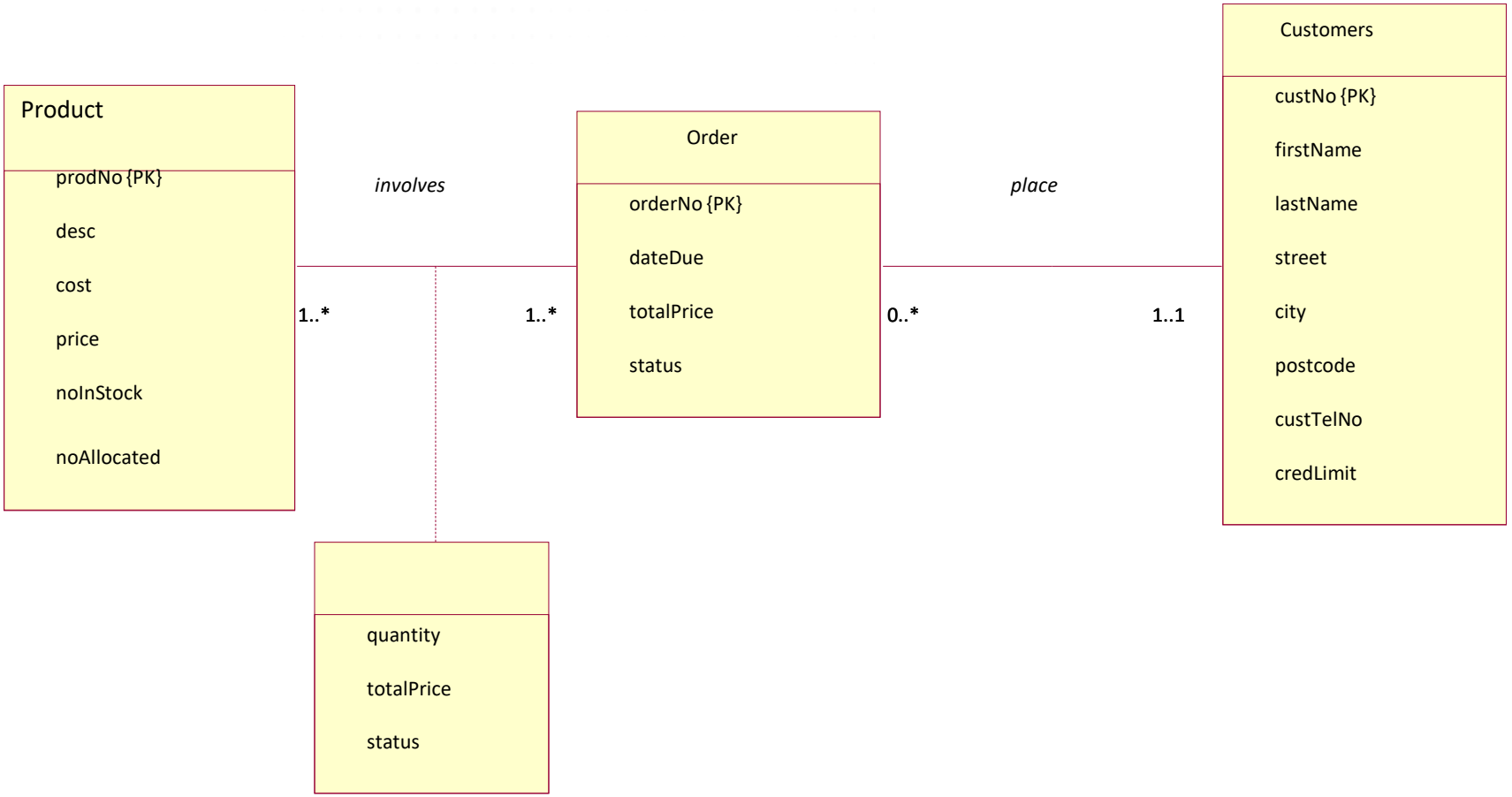
Figure 17.1 Conceptual data model for the StaffClient user views showing all attributes.

Relations for the Staff View of DreamHome

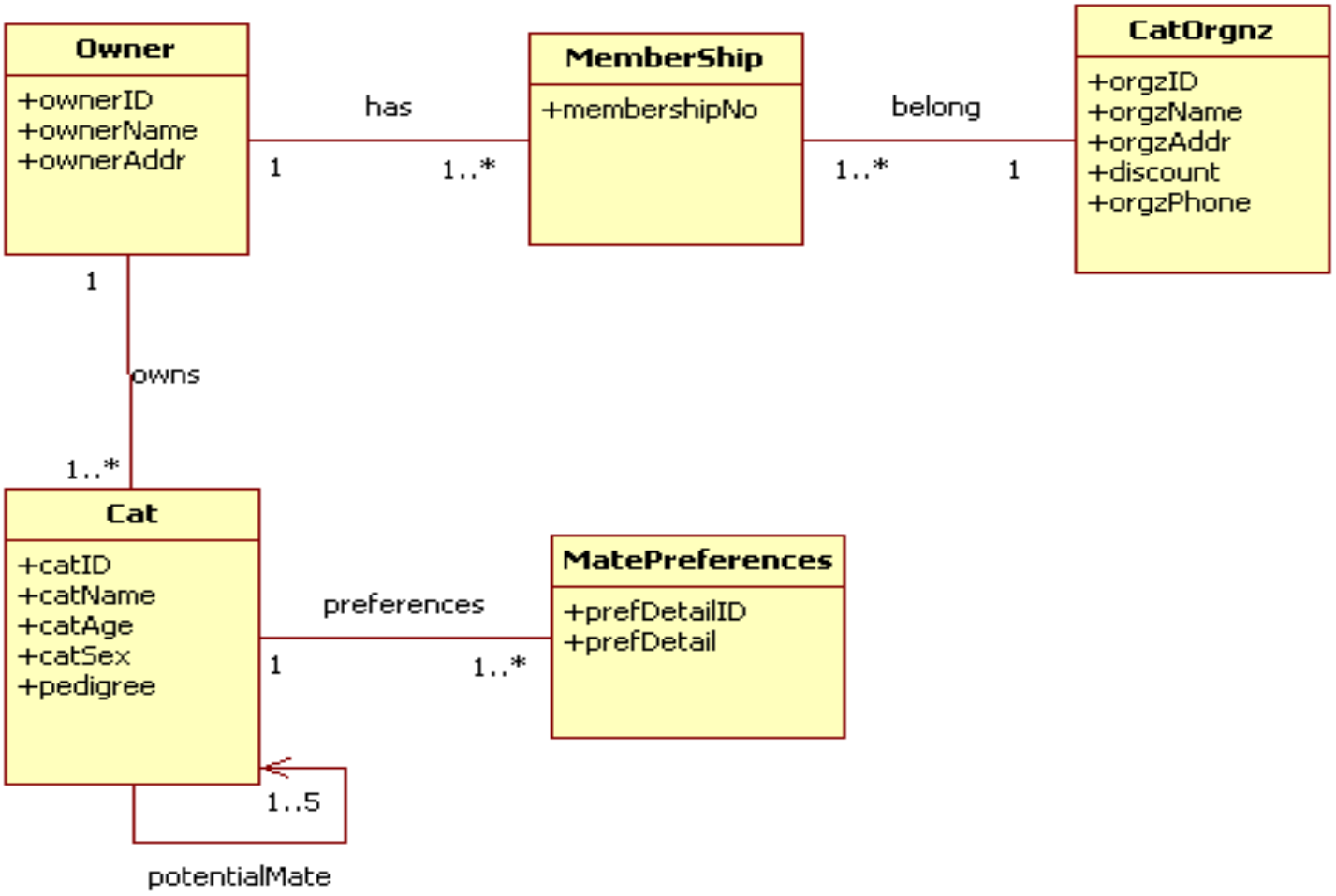
Branch (branchNo, street, city, postcode, mgrStaffNo) **Primary Key** branchNo **Alternate Key** postcode **Foreign Key** mgrStaffNo **references** Manager(staffNo)	**Telephone** (telNo, branchNo) **Primary Key** telNo **Foreign Key** branchNo **references** Branch(branchNo)
Staff (staffNo, name, position, salary, supervisorStaffNo, branchNo) **Primary Key** staffNo **Foreign Key** supervisorStaffNo **references** Staff(staffNo) **Foreign Key** branchNo **references** Branch(branchNo)	**Manager** (staffNo, mgrStartDate, bonus) **Primary Key** staffNo **Foreign Key** staffNo **references** Staff(staffNo)
PrivateOwner (ownerNo, name, address, telNo) **Primary Key** ownerNo	**BusinessOwner** (bName, bType, contactName, address, telNo) **Primary Key** bName **Alternate Key** telNo
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, bName, branchNo) **Primary Key** propertyNo **Foreign Key** ownerNo **references** PrivateOwner(ownerNo) **Foreign Key** bName **references** BusinessOwner(bName) **Foreign Key** staffNo **references** Staff(staffNo) **Foreign Key** branchNo **references** Branch(branchNo)	**Client** (clientNo, name, telNo, eMail, prefType, maxRent) **Primary Key** clientNo **Alternate Key** eMail
Lease (leaseNo, paymentMethod, depositPaid, rentStart, rentFinish, 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)	**Registration** (clientNo, branchNo, staffNo, dateJoined) **Primary Key** clientNo, branchNo **Foreign Key** clientNo **references** Client(clientNo) **Foreign Key** branchNo **references** Branch(branchNo) **Foreign Key** staffNo **references** Staff(staffNo)
Advert (propertyNo, newspaperName, dateAdvert, cost) **Primary Key** propertyNo, newspaperName, dateAdvert **Foreign Key** propertyNo **references** PropertyForRent(propertyNo) **Foreign Key** newspaperName **references** Newspaper(newspaperName)	**Newspaper** (newspaperName, address, telNo, contactName) **Primary Key** newspaperName **Alternate Key** telNo

Figure 17.5 Relations for the Branch user views of *DreamHome*.

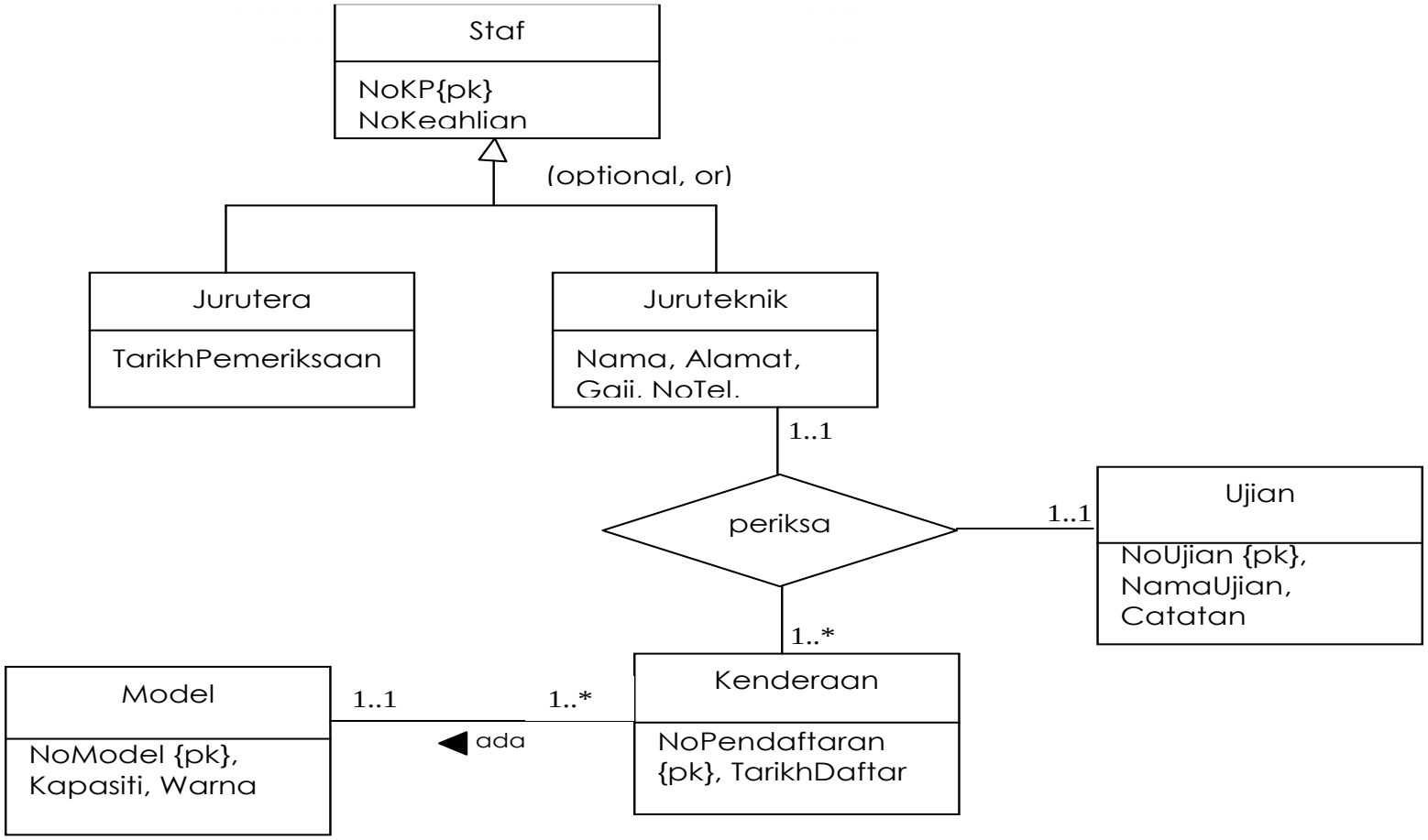
Practice 1: Derive Relations



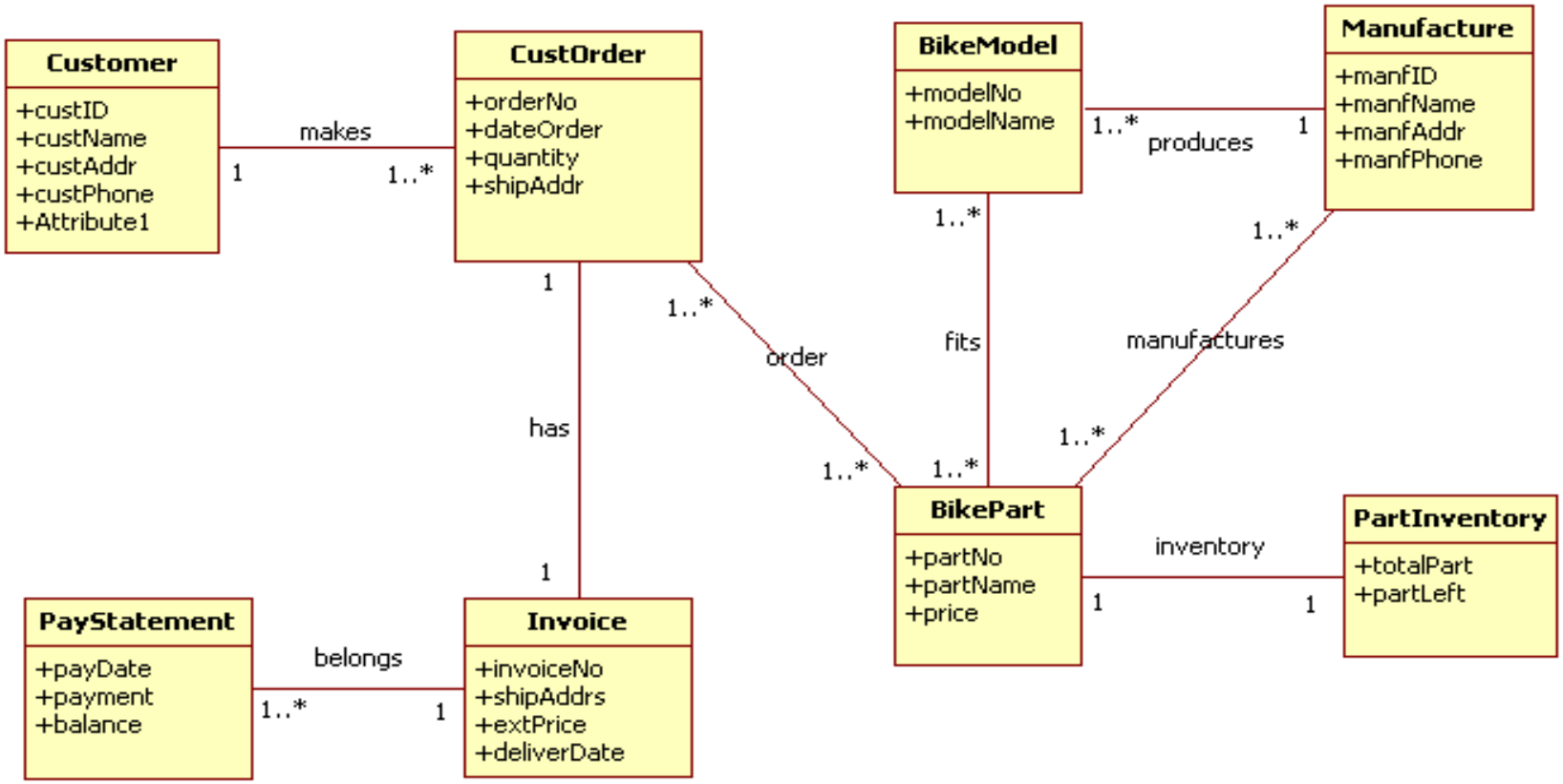
Practice 2: Derive Relations



Practice 3: Derive Relations



Practice 4: Derive Relations



The rest of steps in logical DB design:

- **Step 2.2 Validate relations using normalization**
  - To validate the relations in the local logical data model using the technique of normalization (details will be discussed in Chapter Normalization).
- **Step 2.3 Validate relations against user transactions**
  - To ensure that the relations in the local logical data model support the transactions required by the view.
- **Step 2.4 Check integrity constraints**
  - To check that integrity constraints are represented in logical data model (i.e. required data, entity and referential integrity, domains, and enterprise constraints).
- **Step 2.5 Review local logical data model with user**
  - To ensure that the local logical data model and supporting documentation that describes the model is a true representation of the view.

The rest of steps in logical DB design:

- **Step 2.6 Merge local logical data models into global model (optional step)**
  - To merge the individual local logical data models into a single global logical data model of the enterprise that represents all user views of a database.
    - 2.6.1 Merge logical data models into global logical data model
    - 2.6.2 Validate global logical data model
    - 2.6.3 Review global logical data model with users
- **Step 2.7 Check for future growth**
  - To determine whether there are any significant changes likely in the foreseeable future and to assess whether the global logical data model can accommodate these changes.

Innovating Solutions
Menginovasi Penyelesaian