UNTAR
Universitas Tarumanagara

Terakreditasi BAN PT A (Unggul)

UNTAR untuk INDONESIA

DATABASE DESIGN & MANAGEMENT

SI10317

PROGRAM STUDI SISTEM INFORMASI

UIVERRSITAS TARUMANAGARA

Course Schedule

1 Entity Relationship Modeling dan Alternative ER Notation – Appendix C
2 Exercises
3. Enhanced Entity–Relationship Modeling
4. Exercises
5. Normalization dan Exerises
6. Advanded Normalization dan Exercises
7. Reiew and the *DreamHome* Case Study
8. UTS - Presentasi Project

Course Schedule

9. Methodology—Conceptual Database Design
10. Methodology—Logical Database Design
11. Exercises: Case Study Appendix A, B1, B2
12. Presentasi Project: Case Study
13. Query Processing
14. Distributed DBMSs—Concepts and Design
15. Replication and Mobile Databases
16. Presentasi Project UAS

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 and able to support the required transactions.

Step 2 Build and Validate Logical Data Model

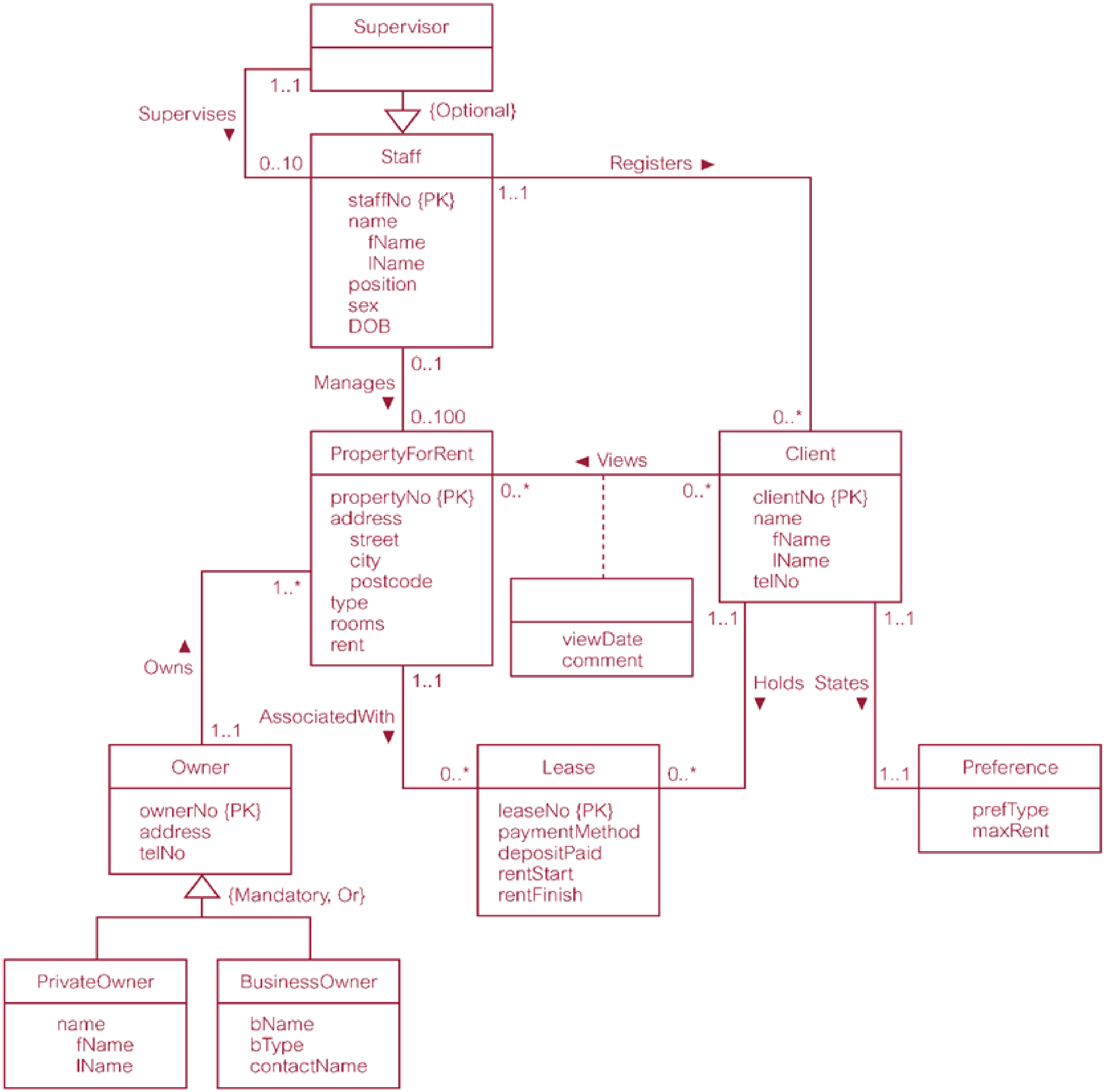
- Step 2.1 Derive relations for logical data model
- Step 2.2 Validate relations using normalization
- Step 2.3 Validate relations against user transactions
- Step 2.4 Check integrity constraints
- Step 2.5 Review logical data model with user
- Step 2.6 Merge logical data models into global model (optional step)
- Step 2.7 Check for future growth

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.

- For most of the examples discussed below we use the conceptual data model for the Staff user views of DreamHome, which is represented as an ER diagram in Figure 16.1.

Figure 16.1 Conceptual data model for the Staff user views showing all attributes.



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) complex relationship types;

  (9) multi-valued attributes.

Step 2.1 Derive relations for logical data model

(1) Strong entity types

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

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

Primary Key staffNo

Step 2.1 Derive relations for logical data model

(2) 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.

Preference (prefType, maxRent)

Primary Key None (at present)

Step 2.1 Derive relations for logical data model

(3) 1:* binary relationship types

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

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
Alternate Key telNo
Foreign Key staffNo **references** Staff(staffNo)

Step 2.1 Derive relations for logical data model

(4) 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 post copy of primary key from one relation to the other. Consider 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 new relation, while other (if one exists) is used as an alternate key.

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 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 follow the posting of the primary key to the child relation.

Step 2.1 Derive relations for logical data model

(b) *Mandatory* participation on *one* side of a 1:1 relationship - Example

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 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.

Example:

- Consider 1:1 Staff *Uses* Car relationship with optional participation on both sides. 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.

Step 2.1 Derive relations for logical data model

(5) 1:1 recursive relationships - follow rules for participation for a 1:1 relationship.

- mandatory participation on both sides: single relation with two copies of the primary key.
- mandatory participation on only one side: option to create a single relation with two copies of the primary key, or create a new relation to represent the relationship. The new relation would only have two attributes, both copies of the primary key.
- optional participation on both sides, again create a new relation as described above.

Step 2.1 Derive relations for logical data model

(6) 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.

UNTAR
Universitas Tarumanagara
UNTAR untuk INDONESIA

Table 16.1. Guidelines for Representation of Superclass / Subclass

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

Step 2.1 Derive relations for logical data model

Example.

Consider the Owner superclass/subclass relationship shown in Figure 16.1. From Table 16.1 there are various ways to represent this relationship as one or more relations, as shown in **Figure 16.2**.

UNTAR
Universitas Tarumanagara
UNTAR untuk INDONESIA

Step 2.1 Derive relations for logical data model

Option 1 – Mandatory, nondisjoint

AllOwner (ownerNo, address, telNo, fName, lName, bName, bType, contactName, pOwnerFlag, bOwnerFlag)

Primary Key ownerNo

Option 2 – Optional, nondisjoint

Owner (ownerNo, address, telNo)

Primary Key ownerNo

OwnerDetails (ownerNo, fName lName, bName, bType, contactName, pOwnerFlag, bOwnerFlag)

Primary Key ownerNo

Foreign Key ownerNo **references** Owner(ownerNo)

Step 2.1 Derive relations for logical data model

Figure 16.2. Various representations of the Owner superclass/ subclass relationship based on the participation and disjointness constraints shown in Table 16.1.

Option 3 – Mandatory, disjoint

PrivateOwner (ownerNo, fName, lName, address, telNo)
Primary Key ownerNo

BusinessOwner (ownerNo, bName, bType, contactName, address, telNo)
Primary Key ownerNo

Option 4 – Optional, disjoint

Owner (ownerNo, address, telNo)
Primary Key ownerNo

PrivateOwner (ownerNo, fName, lName)
Primary Key ownerNo
Foreign Key ownerNo **references** Owner(ownerNo)

BusinessOwner (ownerNo, bName, bType, contactName)
Primary Key ownerNo
Foreign Key ownerNo **references** Owner(ownerNo)

Step 2.1 Derive relations for logical data model

(7) *:* 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.

Step 2.1 Derive relations for logical data model

(7) *:* binary relationship types - Example

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) 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 (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.

Step 2.1 Derive relations for logical data model

(8) Complex relationship types - Example

Staff (staffNo, fName, lName, position, sex, DOB, supervisorStaffNo)
Primary Key staffNo
Foreign Key supervisorStaffNo **references** Staff(staffNo)

Branch (branchNo, street, city, postcode)
Primary Key branchNo

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

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 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.

Step 2.1 Derive relations for logical data model

(9) Multi-valued attributes - Example

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

Table 15.2 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 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 15.1.
: 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 the technique of normalization.

Step 2.3 Validate relations against user transactions

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

  - Langkah ini dilakukan seperti saat melakukan langkah 1.8 yaitu melakukan validasi model data logis terhadap kesesuaian dengan transaksinya.

Step 2.4 Check integrity constraints

- To check integrity constraints are represented in the logical data model.
- We consider the following types of integrity constraint:
  - required data: harus ada atribut mengandung nilai valid dan tidak boleh null, seharusnya sudah teridentifikasi di langkah 1.3 Identify and associate attributes with entity or relationship types.
  - attribute domain constraints: misalnya atribut sex diisi dengan karakter tunggal “M” atau “F”, ada dilngkah 1.4;
  - Multiplicity: menetapkan batasan relationship misalnya setiap brach memiliki banyak staff dan setiap staff hanya bekerja pada satu cabang, dokumentasinya di langkah 1.2;

UNTAR
Universitas Tarumanagara
UNTAR untuk INDONESIA

Step 2.4 Check integrity constraints

- entity integrity: harus menetapkan bahwa primary key tidak boleh null, batasan ini harus sudah tercermin di langkah 1.5;
- referential integrity: maksudnya jika foreign key mengandung nilai, maka nilai tersebut harus mengacu pada tuple yang ada di relasi induksnya (parent relation)
- general constraints: batasan yang diterapkan terhadap jumlah instance yang muncul pada relationship, misal jumlah karyawan di suatu cabang tidak boleh lebih dari 100 staff.

Referential integrity constraints for the relations in the Staff user views of DreamHome.

Staff (staffNo, fName, lName, position, sex, DOB, supervisorStaffNo)

Primary Key staffNo

Foreign Key supervisorStaffNo **references** Staff(staffNo) ON UPDATE CASCADE ON DELETE SET NULL

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

Primary Key clientNo

Foreign Key staffNo **references** Staff(staffNo) ON UPDATE CASCADE ON DELETE NO ACTION

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

Primary Key propertyNo

Foreign Key ownerNo **references** PrivateOwner(ownerNo) and BusinessOwner(ownerNo)
ON UPDATE CASCADE ON DELETE NO ACTION

Foreign Key staffNo **references** Staff(staffNo) ON UPDATE CASCADE ON DELETE SET NULL

Referential integrity constraints for the relations in the Staff user views of DreamHome.

Viewing (clientNo, propertyNo, dateView, 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 CASCADE

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) ON UPDATE CASCADE ON DELETE NO ACTION

Foreign Key propertyNo **references** PropertyForRent(propertyNo)
ON UPDATE CASCADE ON DELETE NO ACTION

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.
  - Relationship between logical data model and data flow diagrams: tentang DFD akan dipelajari di mata kuliah Analysis and Design.

Step 2.6 Merge logical data models into global model (optional step)

- To merge local logical data models into a single global logical data model that represents all user views of a database.
- The activities in this step include:
  - Step 2.6.1 Merge local logical data models into global model
  - Step 2.6.2 Validate global logical data model
  - Step 2.6.3 Review global logical data model with users

Step 2.6.1 Merge local logical data models into global model

- To merge local logical data models into a single global logical data model.
- Some typical tasks in this approach are as follows:
    - (1) Review the names and contents of entities/relations and their candidate keys.
    - (2) Review the names and contents of relationships/foreign keys.
    - (3) Merge entities/relations from the local data models.
    - (4) Include (without merging) entities/relations unique to each local data model.
    - (5) Merge relationships/foreign keys from the local data models.

Step 2.6.1 Merge local logical data models into global model

- Some typical tasks in this approach are as follows:
  - (6) Include (without merging) relationships/foreign keys unique to each local data model.
  - (7) Check for missing entities/relations and relationships/foreign keys.
  - (8) Check foreign keys.
  - (9) Check integrity constraints.
  - (10) Draw the global ER/relation diagram.
  - (11) Update the documentation.

(1)
Table 16.3. A comparison of the names of entities/relations and their candidate keys in the Branch and Staff user views

Branch user views		Staff user views	
Entity/Relation	Candidate keys	Entity/Relation	Candidate keys
Branch	branchNo postcode		
Telephone	telNo		
Staff	**staffNo**	**Staff**	**staffNo**
Manager	staffNo		
PrivateOwner	**ownerNo**	**PrivateOwner**	**ownerNo**
BusinessOwner	**bName** **telNo**	**BusinessOwner**	**bName** **telNo** ownerNo
Client	**clientNo**	**Client**	**clientNo**
PropertyForRent	**propertyNo**	**PropertyForRent**	**propertyNo**
		Viewing	clientNo, propertyNo
Lease	**leaseNo** **propertyNo, rentStart** **clientNo, rentStart**	**Lease**	**leaseNo** **propertyNo, rentStart** **clientNo, rentStart**
Registration	clientNo		
Newspaper	newpaperName telNo		
Advert	(propertyNo, newspaperName, dateAdvert)		

(2) Table 16.4. A comparison of the foreign keys in the Branch and Staff user views

Branch user views			Staff user views		
Child relation	Foreign keys	Parent relation	Child relation	Foreign keys	Parent relation
Branch	mgrStaffNo →	Manager(staffNo)			
Telephone[a]	branchNo →	Branch(branchNo)			
Staff	**supervisorStaffNo →**	**Staff(staffNo)**	Staff	**supervisorStaffNo →**	**Staff(staffNo)**
	branchNo →	Branch(branchNo)			
Manager	staffNo →	Staff(staffNo)			
PrivateOwner			PrivateOwner		
BusinessOwner			BusinessOwner		
Client			Client	staffNo →	Staff(staffNo)
PropertyForRent	**ownerNo →**	**PrivateOwner(ownerNo)**	PropertyForRent	**ownerNo →**	**PrivateOwner(ownerNo)**
	bName →	BusinessOwner(bName)		ownerNo →	BusinessOwner(ownerNo)
	staffNo →	**Staff(staffNo)**		**staffNo →**	**Staff(staffNo)**
	branchNo →	Branch(branchNo)			
			Viewing	clientNo →	Client(clientNo)
				propertyNo →	PropertyForRent(propertyNo)
Lease	**clientNo →**	**Client(clientNo)**	Lease	**clientNo →**	**Client(clientNo)**
	propertyNo →	**PropertyForRent(propertyNo)**		**propertyNo →**	**PropertyForRent(propertyNo)**
Registration[b]	clientNo →	Client(clientNo)			
	branchNo →	Branch(branchNo)			
	staffNo →	Staff(staffNo)			
Newspaper					
Advert[c]	propertyNo →	PropertyForRent(propertyNo)			
	newspaperName →	Newspaper(newspaperName)			

(3) Merging the PrivateOwner relations from the Branch and Staff user views.

Figure 16.6. Merging entities/relations with the same name and the same primary key

Branch user views

PrivateOwner (ownerNo, name, address, telNo)
Primary Key ownerNo

Staff user views

PrivateOwner (ownerNo, fName, lName, address, telNo)
Primary Key ownerNo

Global user views

PrivateOwner (ownerNo, fName, lName, address, telNo)
Primary Key ownerNo

(3) Merging the BusinessOwner relations with different primary keys

Figure 16.7. Merging entities/relations with the same name but different primary keys

Branch user views

BusinessOwner (bName, bType, contactName, address, telNo)

Primary Key bName

Alternate Key telNo

Staff user views

BusinessOwner (ownerNo, bName, bType, contactName, address, telNo)

Primary Key ownerNo

Alternate Key bName

Alternate Key telNo

Global user views

BusinessOwner (ownerNo, bName, bType, contactName, address, telNo)

Primary Key ownerNo

Alternate Key bName

Alternate Key telNo

(4) Include (without merging) entities/relations unique to each local data model.
(5) Merge relationships/foreign keys from the local data models
(6) Include (without merging) relationships/foreign keys unique to each local data model
(7) Check for missing entities/relations and relationships/foreign keys

(8) Check foreign keys: sampai pada langkah ini, entitas/relasi dan relasionship/foreign key mungkin telah digabungkan, mungkin primary key berubah, dan relationship yang baru teridentifikasi. Lihat Figure 16.8.

(9) Check integrity constraints
(10) Draw the global ER/relation diagram: pada langkah ini dihasilkan model ER secara global. Lihat Figure 16.9.

(11) Update the documentation

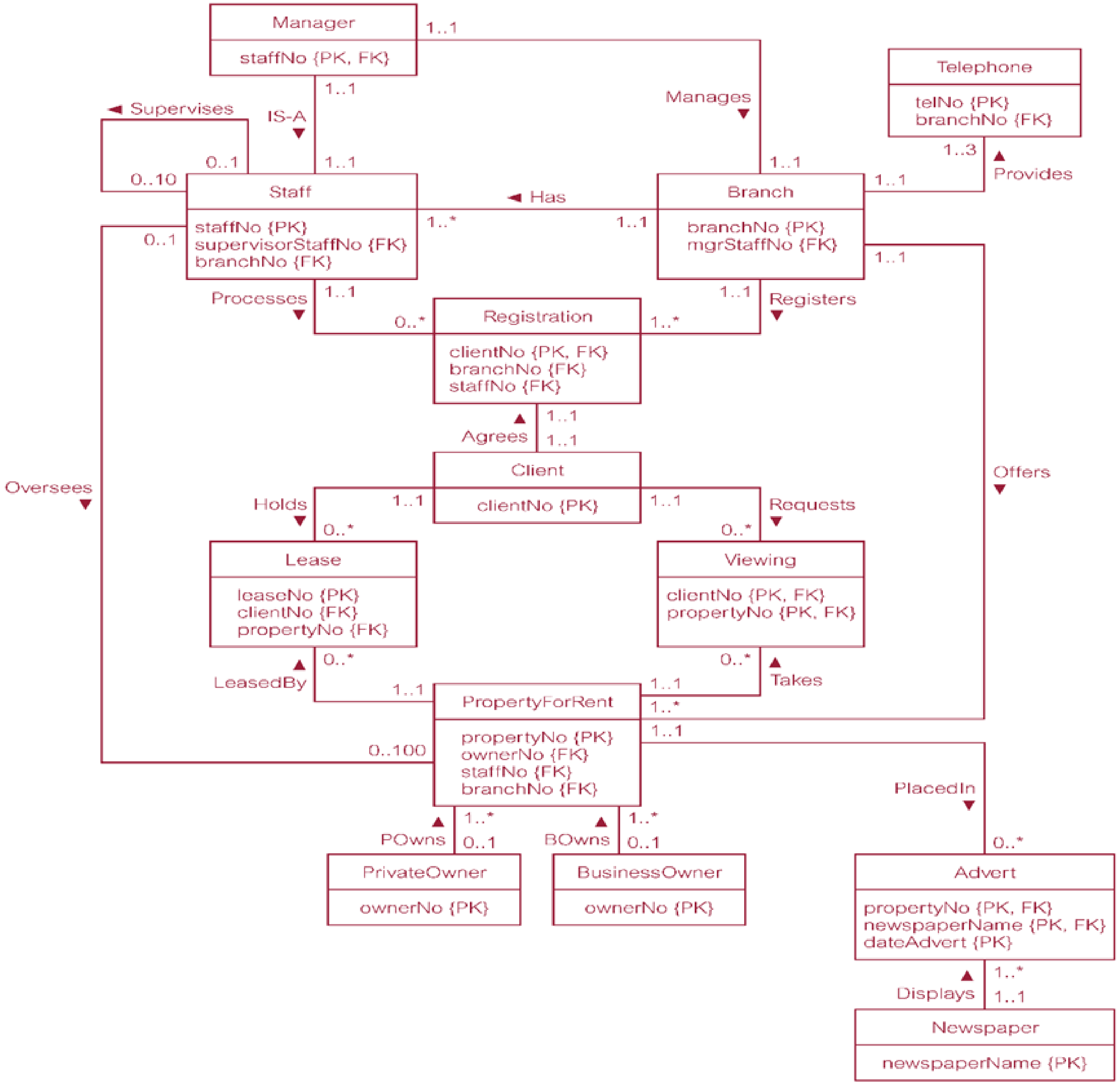
Figure 16.8 Relations that represent the global logical data model for DreamHome

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, fName, lName, position, sex, DOB, 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, fName, lName, address, telNo) **Primary Key** ownerNo	**BusinessOwner** (ownerNo, bName, bType, contactName, address, telNo) **Primary Key** ownerNo **Alternate Key** bName **Alternate Key** telNo
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo) **Primary Key** propertyNo **Foreign Key** ownerNo **references** PrivateOwner(ownerNo) and BusinessOwner(ownerNo) **Foreign Key** staffNo **references** Staff(staffNo) **Foreign Key** branchNo **references** Branch(branchNo)	**Viewing** (clientNo, propertyNo, dateView, comment) **Primary Key** clientNo, propertyNo **Foreign Key** clientNo **references** Client(clientNo) **Foreign Key** propertyNo **references** PropertyForRent(propertyNo)

Figure 16.8 Relations that represent the global logical data model for DreamHome

Client (clientNo, fName, lName, telNo, prefType, maxRent) **Primary Key** clientNo	**Registration** (clientNo, branchNo, staffNo, dateJoined) **Primary Key** clientNo **Foreign Key** clientNo **references** Client(clientNo) **Foreign Key** branchNo **references** Branch(branchNo) **Foreign Key** staffNo **references** Staff(staffNo)
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)	**Newspaper** (newspaperName, address, telNo, contactName) **Primary Key** newspaperName **Alternate Key** telNo
Advert (propertyNo, newspaperName, dateAdvert, cost) **Primary Key** propertyNo, newspaperName, dateAdvert **Foreign Key** propertyNo **references** PropertyForRent(propertyNo) **Foreign Key** newspaperName **references** Newspaper(newspaperName)	

Figure 16.9 Global relation diagram for DreamHome



Step 2.6.2 Validate global logical data model

- To validate the relations created from the global logical data model using the technique of normalization and to ensure they support the required transactions, if necessary.

Step 2.6.3 Review global logical data model with users

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

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 logical data model can accommodate these changes.

Thank You

Reference: Database Systems A Practical Approach to Design, Implementation, and Management Fourth Edition.

Thomas M. Connolly and Carolyn E. Begg