DBMS
Conceptual Design

University of Basrah
College of CS & IT
Assist Lec. Mohammed Dheyaa

Objectives

In this Lecture you will learn:

- The purpose of a design methodology.
- The three main phases of database design: conceptual, logical, and physical design.
- How to decompose the scope of the design into specific views of the enterprise.
- How to use ER modeling to build a local conceptual data model based on the Information given in a view of the enterprise.
- How to validate the resultant conceptual data model to ensure that it is a true and accurate representation of a view of the enterprise.
- How to document the process of conceptual database design.
- End-users play an integral role throughout the process of conceptual database design.

1. Introduction

The methodology is presented as a step-by-step guide to the three main phases of database design, namely: conceptual, logical, and physical design.
The main aim of each phase is as follows:

- **Conceptual database design**—to build the conceptual representation of the database, which includes identification of the important entities, relationships, and attributes.
- **Logical database design**—to translate the conceptual representation to the logical structure of the database, which includes designing the relations.
- **Physical database design**—to decide how the logical structure is to be physically implemented (as base relations) in the target DBMS.

1.2 What Is a Design Methodology?

Design Methodology: A structured approach that uses procedures, techniques, tools, and documentation aids to support and facilitate the process of design.

The design process is divided into three main phases: conceptual, logical, and physical database design:

Conceptual database design: The process of constructing a model of the data used in an enterprise, independent of *all* physical considerations.

Logical database design: 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.

Physical database design: The process of producing a description of the implementation of the database on secondary storage; it describes the base relations, file organizations, and indexes used to achieve efficient access to the data, and any associated integrity constraints and security measures.

1.3 Overview of the Database Design Methodology

The desgin steps in the methodology are as follows.

Conceptual database design

Step 1 Identify entity types
Step 2 Identify relationship types
Step 3 Identify and associate attributes with entity or relationship types
Step 4 Determine attribute domains
Step 5 lDetermine candidate, primary, and alternate key attributes
Step 6 Consider use of enhanced modeling concepts (optional step)
Step 7 Check model for redundancy
Step 8 Validate conceptual data model against user transactions
Step 9 Review conceptual data model with user

Logical database design for the relational model

Step 1 Derive relations for logical data model
Step 2 Validate relations using normalization
Step 3 Validate relations against user transactions
Step 4 Check integrity constraints
Step 5 Review logical data model with user
Step 6 Merge logical data models into global model (optional step)
Step 7 Check for future growth.

Physical database design for relational databases

Step 1 Translate logical data model for target DBMS
- Step 1.1 Design base relations
- Step 1.2 Design representation of derived data
- Step 1.3 Design general constraints

Step 2 Design file organizations and indexes
- Step 2.1 Analyze transactions
- Step 2.2 Choose file organizations
- Step 2.3 Choose indexes
- Step 2.4 Estimate disk space requirements

Step 3 Design user views
Step 4 Design security mechanisms
Step 5 Consider the introduction of controlled redundancy
Step 6 Monitor and tune the operational system

2. Conceptual Database Design Methodology

Objective To build a conceptual data model of the data requirements of the enterprise.

A conceptual data model comprises:

- entity types;
- relationship types;
- attributes and attribute domains;

The tasks involved are:

Step 1 Identify entity types
Step 2 Identify relationship types
Step 3 Identify and associate attributes with entity or relationship types
Step 4 Determine attribute domains
Step 5 Determine candidate, primary, and alternate key attributes
Step 6 Consider use of enhanced modeling concepts (optional step)
Step 7 Check model for redundancy
Step 8 Validate conceptual data model against user transactions
Step 9 Review conceptual data model with user.

2.1 Step 1 Identify entity types

Objective: To identify the required entity types.

The first step in building a conceptual data model is to determine and define the main objects that the users are interested in.

One method of identifying entities is to examine the users' requirements specification. For example, we could group (staff number and staff name) with an object or entity called ***Staff*** and group (property number, property address, rent, and number of rooms) with an entity called *PropertyForRent*. For the ***StaffClient*** user views of ***DreamHome***, we identify the following entities:

Staff	PropertyForRent
PrivateOwner	BusinessOwner
Client	Preference
Lease	

2.2 Step 2: Identify relationship types

Objective: To identify the important relationships that exist between the entity types.

Typically, relationships are indicated by verbs or verbal expressions. For example:

- Staff *Manages* PropertyForRent
- PrivateOwner *Owns* PropertyForRent
- PropertyForRent *AssociatedWith* Lease

2.2.1 Use Entity–Relationship (ER) diagrams

It is often easier to visualize a complex system using ER diagrams to represent entities and how they relate to one another more easily.

2.2.2 Determine the multiplicity constraints of relationship types

Having identified the relationships to model, we next determine the multiplicity of each relationship.

Examples:

1. **One-to-One (1:1) Relationships**

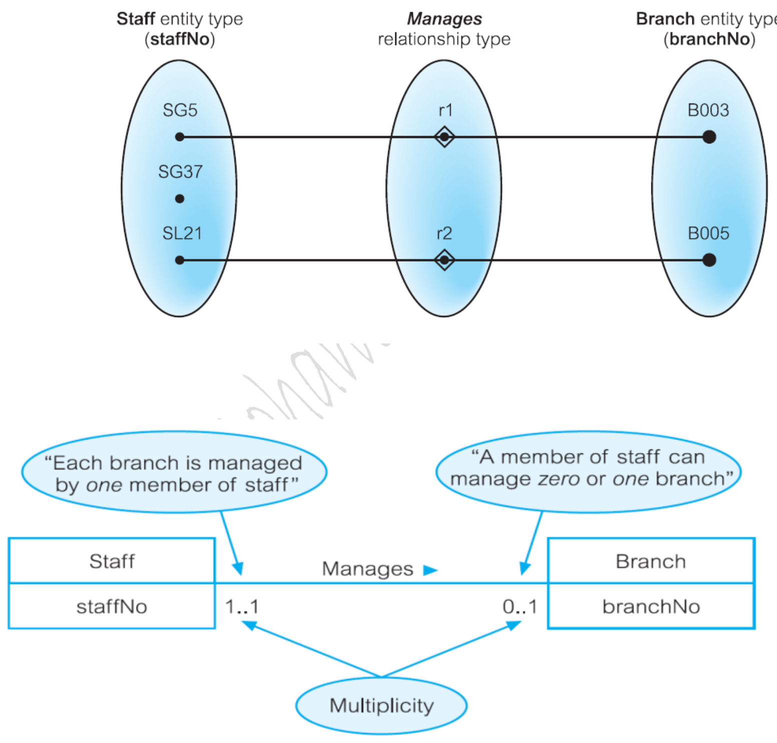
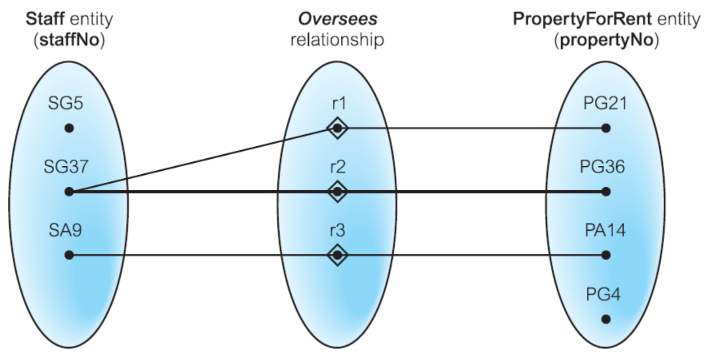

Staff entity type (**staffNo**)
Manages relationship type
Branch entity type (**branchNo**)
SG5
SG37
SL21
r1
r2
B003
B005
"Each branch is managed by *one* member of staff"
"A member of staff can manage *zero* or *one* branch"
Staff
staffNo
Manages ►
Branch
branchNo
1..1
0..1
Multiplicity

Fig.1. One-to-One Multiplicity.

2. One-to-Many (1:*) Relationships



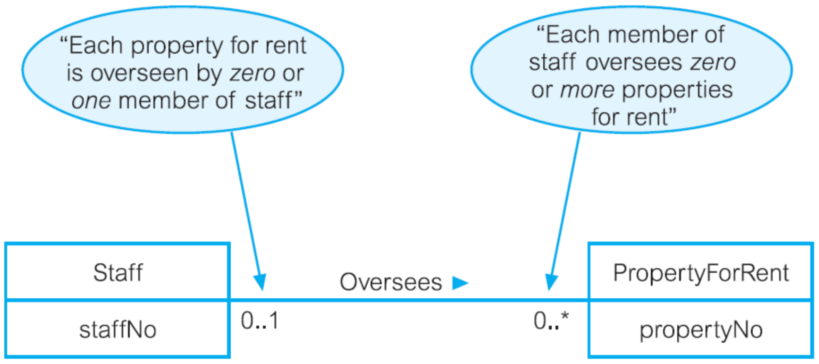
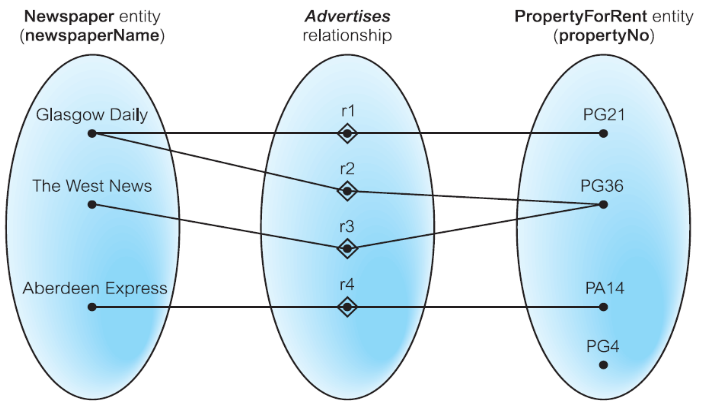

"Each property for rent is overseen by *zero* or *one* member of staff"
"Each member of staff oversees *zero* or *more* properties for rent"
Staff
staffNo
Oversees ►
0..1
0..*
PropertyForRent
propertyNo

Fig. 2 One-to-Many Multiplicity.

3. Many-to-Many (*:*) Relationships



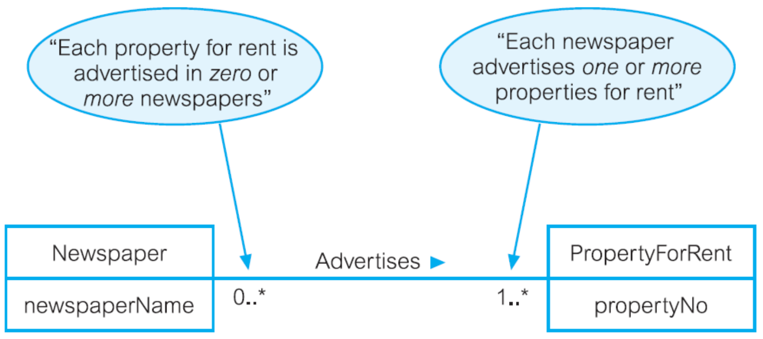

"Each property for rent is advertised in *zero* or *more* newspapers"
"Each newspaper advertises *one* or *more* properties for rent"
Newspaper
newspaperName
Advertises ▶
0..*
1..*
PropertyForRent
propertyNo

Fig. 3 Many-to-Many Multiplicity.

The step Output:

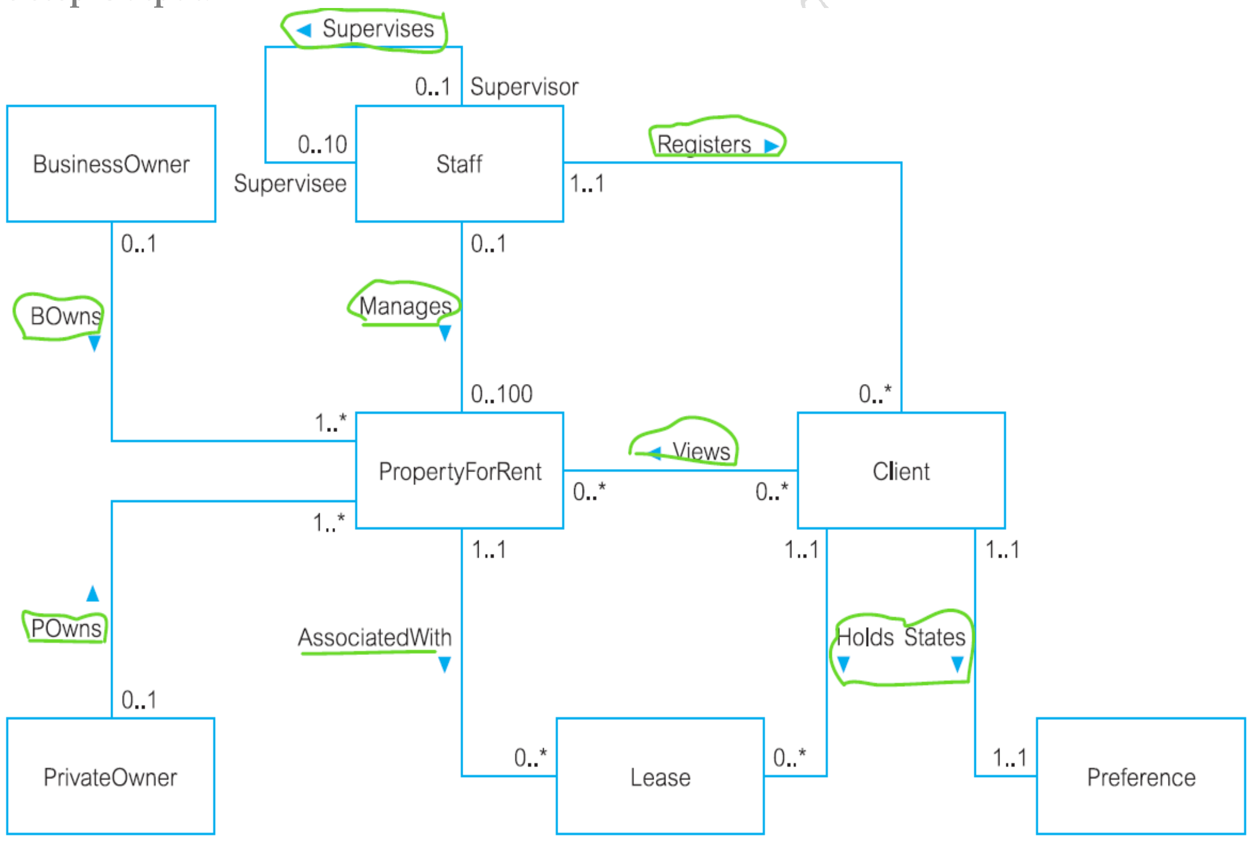


Fig.4 First-cut ER diagram showing entity and relationship types for the StaffClient user views of DreamHome.

2.3 Step 3 Identify and associate attributes with entity or relationship types

Objective: To associate attributes with appropriate entity or relationship types.

The next step in the methodology is to identify the ***types of facts*** about the entities and relationships that we have chosen to be represented in the database.

2.3.1 Simple/composite attributes

It is important to note whether an attribute is ***simple*** or ***composite***.

For example, the *address* attribute can be simple and hold all the details of an address as a single value, such as "115 Dumbarton Road, Glasgow, G11 6YG." Also, the ***address*** attribute may also represent a composite attribute, made up of simple attributes *street* ("115 Dumbarton Road"), *city* ("Glasgow"), and *postcode* ("G11 6YG").

2.3.2 Derived attributes

Attributes whose values are based on the values of other attributes are known as derived attributes. Examples of derived attributes include:

• The age of a member of staff;
• The number of properties that a member of staff manages;
• The rental deposit (calculated as twice the monthly rent).

Step 3 Output:

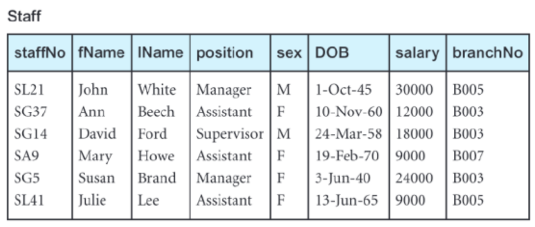
Staff	staffNo, name (composite: fName, lName), position, sex, DOB
PropertyForRent	propertyNo, address (composite: street, city, postcode), type, rooms, rent
PrivateOwner	ownerNo, name (composite: fName, lName), address, telNo
BusinessOwner	ownerNo, bName, bType, address, telNo, contactName
Client	clientNo, name (composite: fName, lName), telNo, eMail
Preference	prefType, maxRent
Lease	leaseNo, paymentMethod, deposit (derived as PropertyForRent.rent*2), depositPaid, rentStart, rentFinish, duration (derived as rentFinish – rentStart)

2.4 Step 4 Determine attribute domains

Objective: To determine domains for the attributes in the conceptual data model.

A **domain** is a pool of values from which one or more attributes draw their values. For example, we may define:

- The attribute domain of valid staff numbers (***staffNo***) as being a five character variable-length string, with the first two characters as letters and the next one to three characters as digits in the range 1–999;
- The possible values for the ***sex*** attribute of the Staff entity as being either "M" or "F." The domain of this attribute is a single character string consisting of the values "M" or "F."



Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

2.5 Step 5 Determine candidate, primary, and alternate key attributes

Objective To identify the candidate key(s) for each entity type and, if there is more than one candidate key, to choose one to be the primary key and the others as alternate keys.

Step 5 Output:

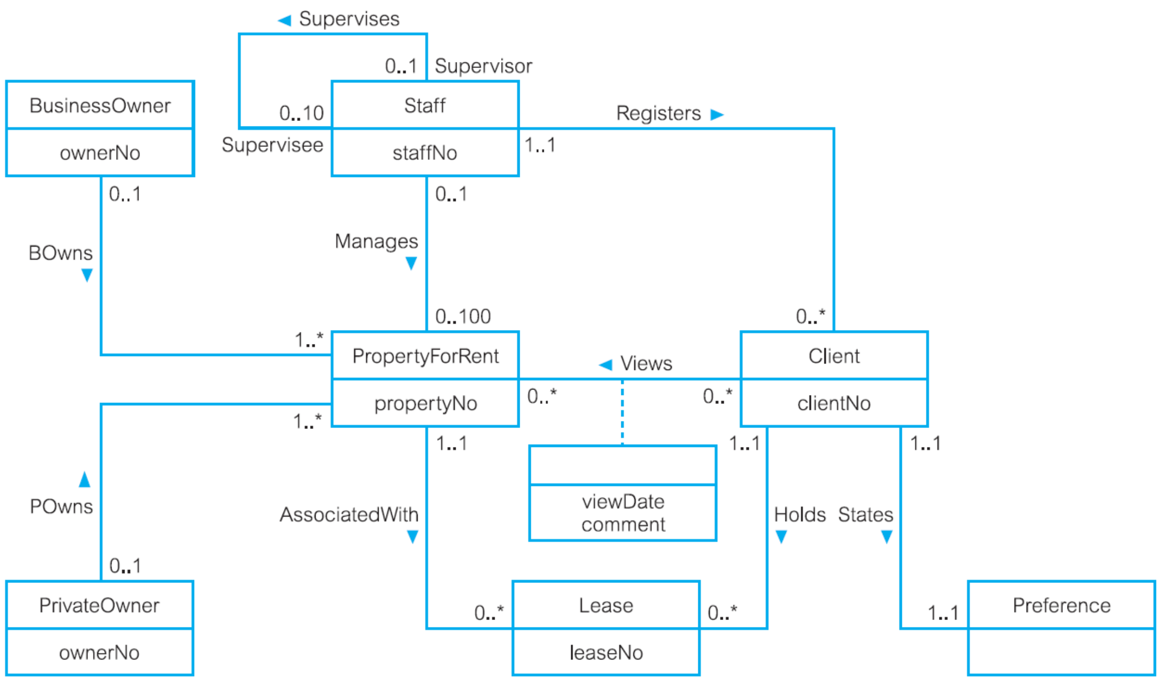


Fig.6 ER diagram for the StaffClient user views of DreamHome with primary keys added.

2.6 Step 6 Consider use of enhanced modeling concepts (optional step)

Objective: To consider the use of enhanced modeling concepts, such as specialization/ generalization, aggregation, and composition.

2.6.1 Specialization/ Generalization

The concept of specialization/generalization is associated with special types of entities known as **superclasses** and **subclasses**, and the process of **attribute inheritance**.

2.6.2 Superclasses and Subclasses

Entity types that have distinct subclasses are called **superclasses**. For example, the entities that are members of the ***Staff*** entity type may be classified as Manager, SalesPersonnel, and Secretary.

In other words, the ***Staff*** entity is referred to as the superclass of the ***Manager***, ***SalesPersonnel***, and ***Secretary*** subclasses.

The relationship between a superclass and any one of its subclasses is called a superclass/subclass relationship.

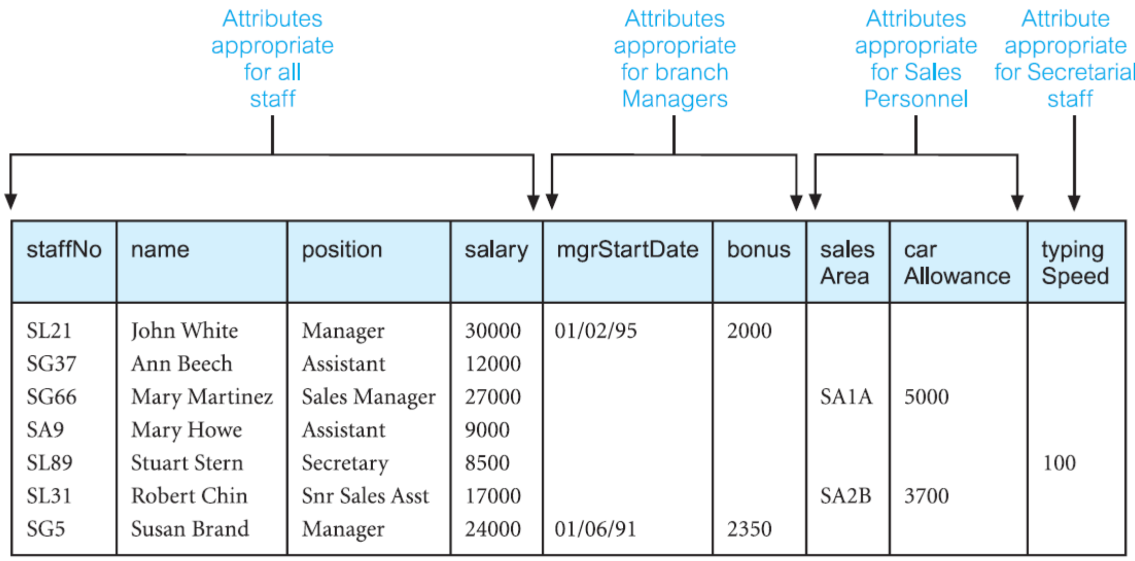
2.6.3 Superclass / Subclass Relationship.

- Each member of a subclass is also a member of the superclass.

- Some superclasses may contain overlapping subclasses, as illustrated by a member of staff who is both a ***Manager*** and a member of ***Sales Personnel***. In this example, ***Manager*** and ***SalesPersonnel*** are overlapping subclasses of the ***Staff*** superclass.

- Not every member of a superclass is necessarily a member of a subclass; for example, members of ***staff*** without a distinct job role such as a ***Manager*** or a member of ***SalesPersonnel***.

- We can use superclasses and subclasses to avoid describing different types of staff with possibly different attributes within a single entity. For example, **SalesPersonnel** may have special attributes such as ***salesArea*** and ***carAllowance***. If all staff attributes and those specific to particular jobs are described by a single ***Staff*** entity, this may result in a lot of ***nulls*** for the job-specific attributes.


Attributes appropriate for all staff
Attributes appropriate for branch Managers
Attributes appropriate for Sales Personnel
Attribute appropriate for Secretarial staff

staffNo	name	position	salary	mgrStartDate	bonus	sales Area	car Allowance	typing Speed
SL21	John White	Manager	30000	01/02/95	2000			
SG37	Ann Beech	Assistant	12000					
SG66	Mary Martinez	Sales Manager	27000			SA1A	5000	
SA9	Mary Howe	Assistant	9000					
SL89	Stuart Stern	Secretary	8500					100
SL31	Robert Chin	Snr Sales Asst	17000			SA2B	3700	
SG5	Susan Brand	Manager	24000	01/06/91	2350			

Fig. 7 The ***AllStaff*** relation holding details of all staff.

2.6.4 Attribute Inheritance

An entity in a subclass represents the same "real world" object as in the superclass, and may possess subclass-specific attributes, as well as those associated with the superclass.

For example, a member of the ***SalesPersonnel*** subclass inherits all the attributes of the ***Staff*** superclass, such as ***staffNo, name, position, and salary*** together with those specifically associated with the ***SalesPersonnel*** subclass, such as ***salesArea and carAllowance***.

2.6.4 Specialization / Generlization Process

Specialization is a top-down approach to defining a set of superclasses and their related subclasses.

For example, If we apply the process of specialization on the ***Staff*** entity, we attempt to identify **differences between members of this entity.** As described earlier, ***staff*** with the job roles of ***Manager, Sales Personnel, and Secretary*** have distinctive attributes and therefore we identify ***Manager, SalesPersonnel, and Secretary*** as subclasses of a specialized ***Staff*** superclass.

Generalization is the process of minimizing the differences between entities by identifying their common characteristics.

For example, consider a model where ***Manager, SalesPersonnel, and Secretary*** are represented as distinct entity types. If we apply the process of generalization on these entities, we attempt to identify similarities between them, such as common attributes and relationships. As stated earlier, these entities share attributes common to all staff, and therefore we identify ***Manager, SalesPersonnel, and Secretary*** as subclasses of a generalized Staff superclass.

2.6.5 Constraints on Specialization/Generalization

1. Participation constraint: determines **whether every member in the superclass must participate as a member of a subclass**. A participation constraint may be **mandatory** or **optional**.

A **mandatory participation** specifies that every member in the superclass must also be a member of a subclass. To represent mandatory participation, "Mandatory" is placed in curly brackets.

An optional participation specifies that a member of a superclass need not belong to any of its subclasses. To represent optional participation, "Optional" is placed in curly brackets.

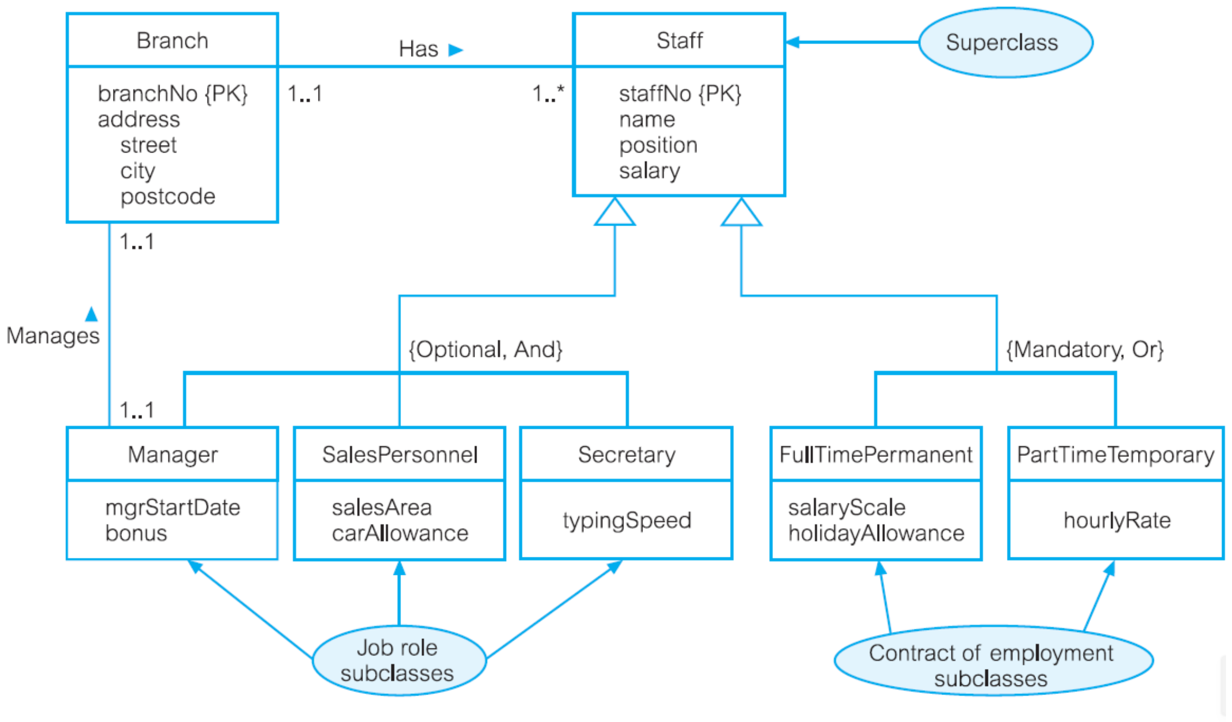


Fig. 9 Mandatory/Optional Constraints.

2. Disjoint constraint: Describes the relationship **between members of the subclasses** and indicates whether it is possible for a member of a superclass to be a member of one, or more than one, subclass.

If the subclasses are **disjoint**, then an entity occurrence can be a member of only **one** of the subclasses. To represent a disjoint superclass/subclass relationship, "**Or**" is placed.

If subclasses of a specialization/generalization are not disjoint (called **nondisjoint**), then an entity occurrence may be a member of **more** than one subclass.

2.7 Step 7 Check model for redundancy

Objective: To check for the presence of any redundancy in the model.

In this step, we examine the conceptual data model with the specific objective of identifying whether there is any **redundancy present** and **removing** any that does exist.

Remove redundant relationships :

For example, consider the relationships between the PropertyForRent, Lease, and Client entities shown in Figure 16.7. There are two ways to find out which clients rent which properties. There is the direct route using the *Rents* relationship between the Client and PropertyForRent entities, and there is the indirect route, using the *Holds* and *AssociatedWith* relationships via the Lease are required, we need to establish the purpose of each relationship.

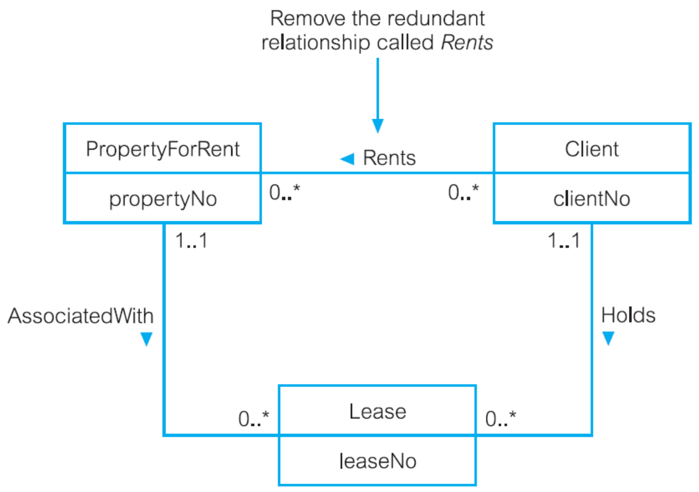


Fig. 9 Remove the redundant relationship called Rents.

The *Rents* relationship indicates which client rents which property. On the other hand, the *Holds* relationship indicates which client holds which lease, and the *AssociatedWith* relationship indicates which properties are associated with which leases. Although it is true that there is a relationship between clients and the properties they rent, this is not a direct relationship and the association is more accurately represented through a lease. The *Rents* relationship is therefore redundant and does not convey any additional information about the relationship between PropertyForRent and Client that cannot more correctly be found through the Lease entity. To ensure that we create a mini- mal model, the redundant *Rents* relationship must be removed.

2.8 Step 8 Validate conceptual data model against user transactions

Objective: To ensure that the conceptual data model supports the required transactions.

We examine two possible approaches to ensuring that the conceptual data model supports the required transactions:

(1) describing the transactions;
(2) using transaction pathways.

1- Describing the transaction

Using the first approach, we check that all the information (entities, relationships, and their attributes) required by each transaction is provided by the model, by documenting a description of each transaction's requirements.

Transaction (d): List the details of properties managed by a named member of staff at the branch.

The details of properties are held in the PropertyForRent entity and the details of staff who manage properties are held in the Staff entity. In this case, we can use the Staff Manages PropertyForRent relationship to produce the required list.

2- Using transaction pathways

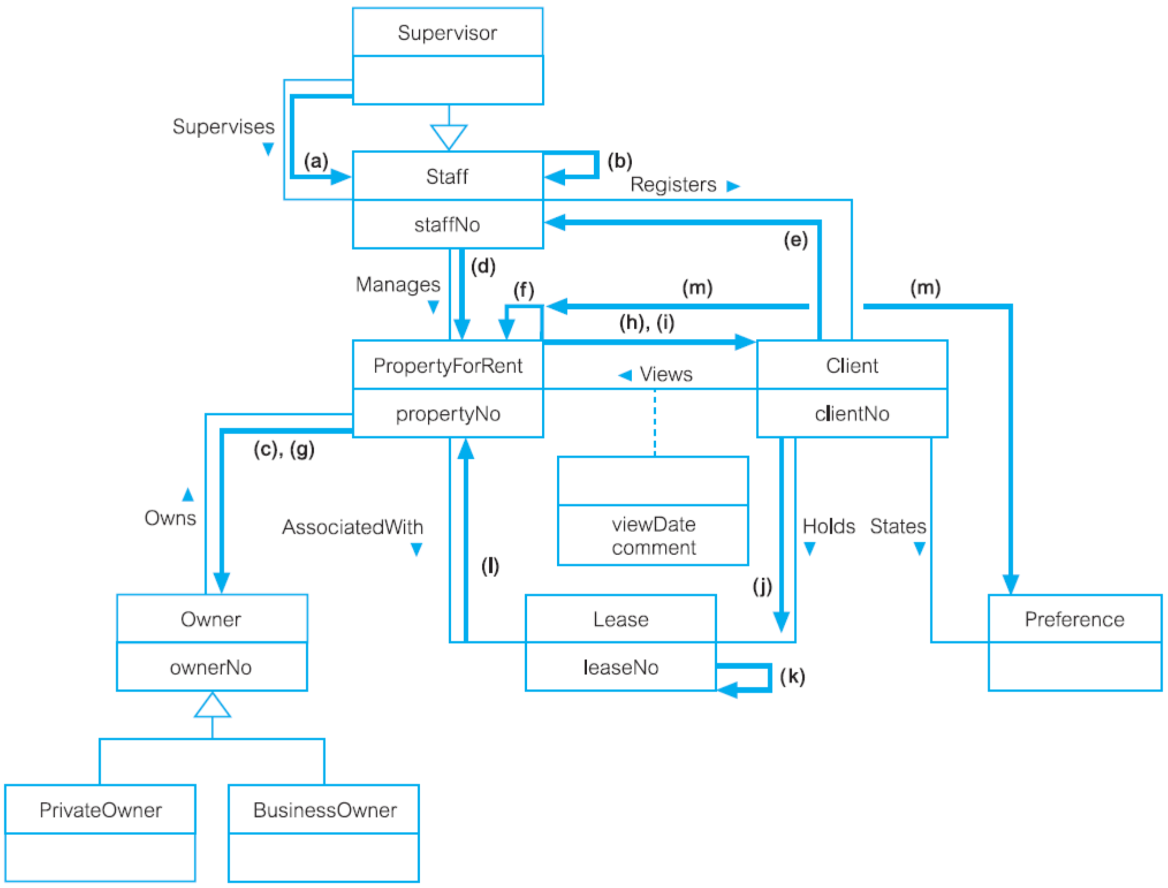


Fig. 10 Remove the redundant relationship called Rents.

The second approach to validating the data model against the required transactions involves diagrammatically representing the pathway taken by each transaction directly on the ER diagram.

Clearly, the more transactions that exist, the more complex this diagram would become.

This approach allows the designer to visualize areas of the model that are not required by transactions and those areas that are critical to transactions.

We are therefore in a position to directly review the support provided by the data model for the transactions required.

If there are areas of the model that do not appear to be used by any transactions, we may question the purpose of representing this information in the data model.

On the other hand, if there are areas of the model that are inadequate in providing the correct pathway for a transaction, we may need to investigate the possibility that critical entities, relationships, or attributes have been missed.

2.9 Step 9 Review conceptual data model with user

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

Checkpoints:

The *DreamHome* case study

1. Create a conceptual data model for the Branch user views of *DreamHome* documented in Appendix A. Compare your ER diagram with Figure 13.8 and justify any differences found.
2. Analyze the *DreamHome* case study and examine if there are situations that call for enhanced modeling. Present the enhanced data model of the case.

The *University Accommodation Office* case study

1. Describe how you will approach the task of creating the conceptual model for university accommodation documented in Appendix B.1.
2. Create a conceptual data model for the case study. State any assumptions necessary to support your design. Check that the conceptual data model supports the required transactions.

The *EasyDrive School of Motoring* case study

1. Analyze the *EasyDrive School of Motoring* case study documented in Appendix B.2 and prepare a data dictionary detailing all the key conceptual issues.
2. Create a conceptual data model for the case study. State any assumptions necessary to support your design. Check that the conceptual data model supports the required transactions.

The *Wellmeadows Hospital* case study

1. Identify user views for the Medical Director and Charge Nurse in the *Wellmeadows Hospital* case study described in Appendix B.3.
2. Provide a users' requirements specification for each of these user views.

3. Create conceptual data models for each of the user views. State any assumptions necessary to support your design.