Chapter 12

Enhanced ER modelling techniques

Objectives

- **The limitations of the basic ER modeling concepts and the requirements to model more complex applications using enhanced data modeling concepts.**
- **The main concepts associated with the Enhanced Entity-Relationship (EER) model is called specialization/generalization.**

The EER model

- Basic concepts of ERM are often perfectly adequate for the representation of the data requirements for many different database applications.
- However, basic concepts can be limiting when modeling more complex database applications with a large amount of data and data with complex interrelationships.

The EER model

- **Original ER model with additional semantic concepts is referred to as the Enhanced Entity-Relationship (EER) model.**

The EER model

- **One of the most useful concepts associated with the EER model is called specialization/generalization.**

Specialization/generalization

- **Associated with special types of entities known as superclasses and subclasses, and the process of attribute inheritance.**

Superclasses and subclasses

- Superclass
  - An entity type that includes one or more distinct subgroupings of its occurrences, which require to be represented in a data model.
- Subclass
  - A distinct subgrouping of occurrences of an entity type, which require to be represented in a data model.

Superclasses

Subclasses

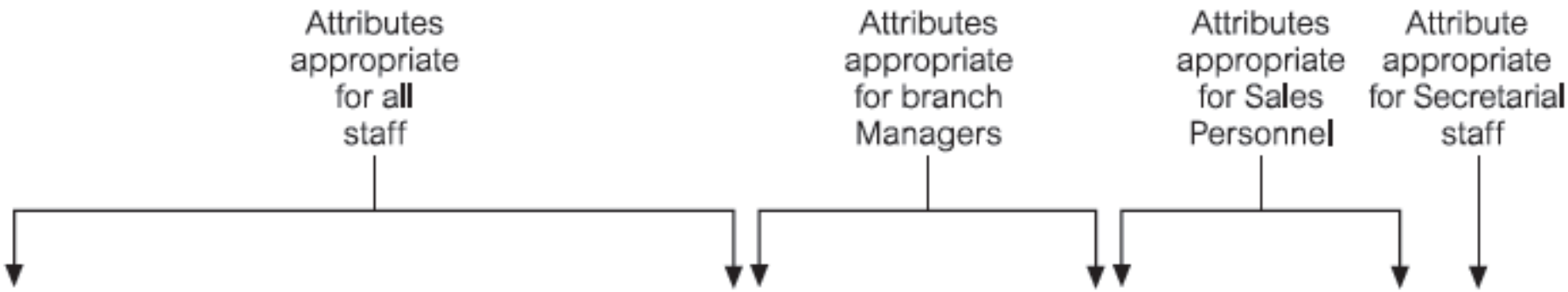
Need of Superclasses and subclasses

- 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**.

Superclasses and subclasses

- We can also show **relationships** that are only associated with particular types **of subclasses** and **not with superclasses**. For example, SalesPersonnel may have **distinct relationships** that are not appropriate for all staff, such as SalesPersonnel *Uses* Car.
- To illustrate these points, consider the relation called AllStaff as shown on next slide

The AllStaff relation holding details of all 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			

Attribute inheritance

- An entity occurrence in a subclass represents the same ‘real world’ object as in the superclass.
- Hence, a member of a subclass inherits those attributes associated with the superclass, but may also have **subclass-specific attributes**.
- 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.

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.

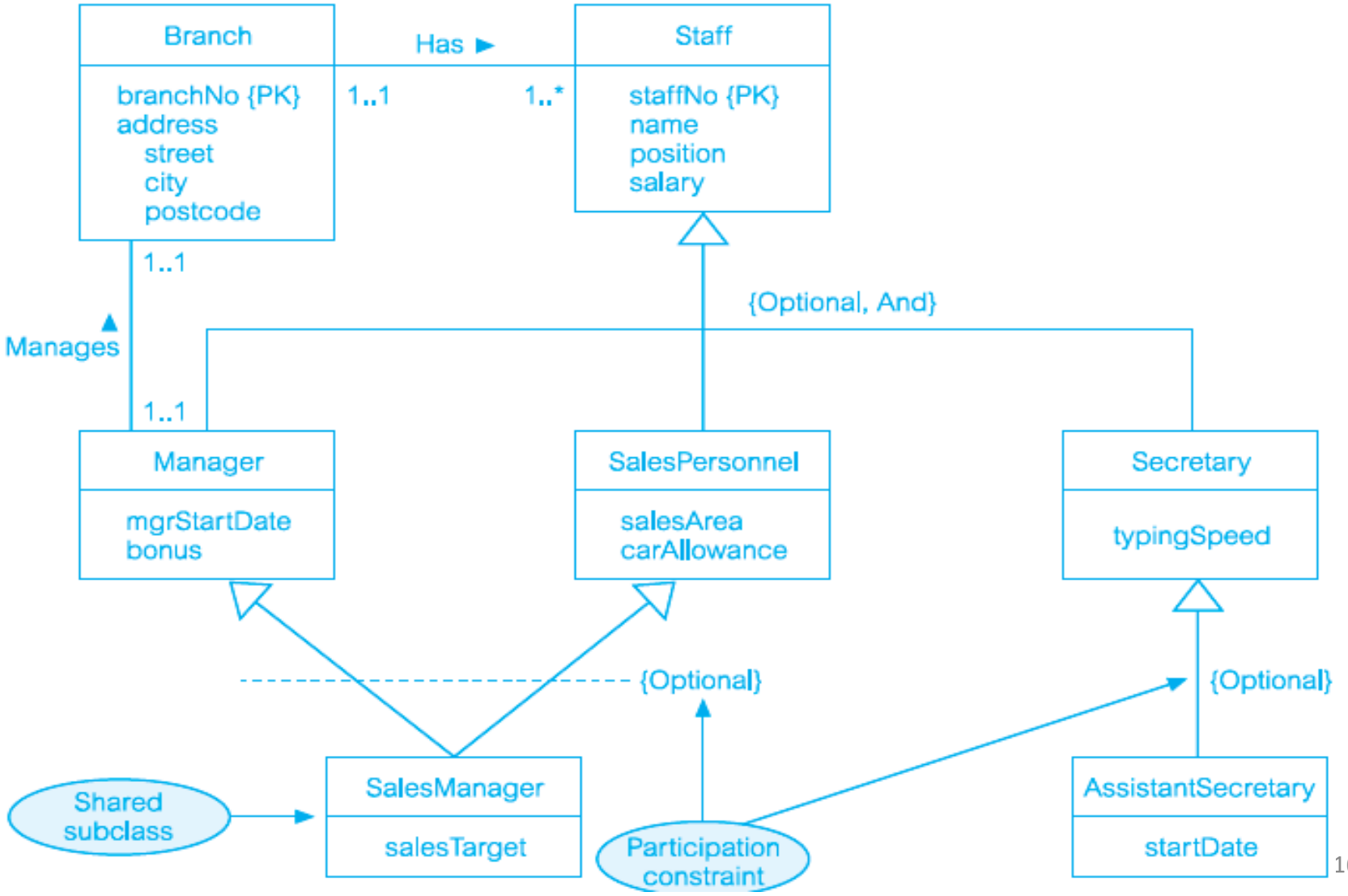
Type Hierarchy

- A subclass is an entity in its own right and so it may also have one or more subclasses.
- An entity and its subclasses and their subclasses, and so on, is called a **type hierarchy**.
- Type hierarchies are known by a variety of names including: **specialization hierarchy** (for example, Manager is a specialization of Staff), **generalization hierarchy** (for example, Staff is a generalization of Manager), and **IS-A hierarchy** (for example, Manager IS-A (member of) Staff).

Multiple Inheritance

- A subclass with more than one superclasses is called a **shared subclass**.
- In other words, a member of a shared subclass must be a member of the associated superclasses.
- As a consequence, the attributes of the superclasse**s** are inherited by the shared subclass, which may also have its own additional attributes.
- This process is referred to as **multiple inheritance**.

Specialization/generalization of the Staff entity into job roles including a shared subclass called SalesManager and a subclass called Secretary with its own subclass called AssistantSecretary



Specialization/generalization

- Specialization
  - The process of maximizing the differences between members of an entity by identifying their distinguishing characteristics.
- Generalization
  - The process of minimizing the differences between entities by identifying their common characteristics.

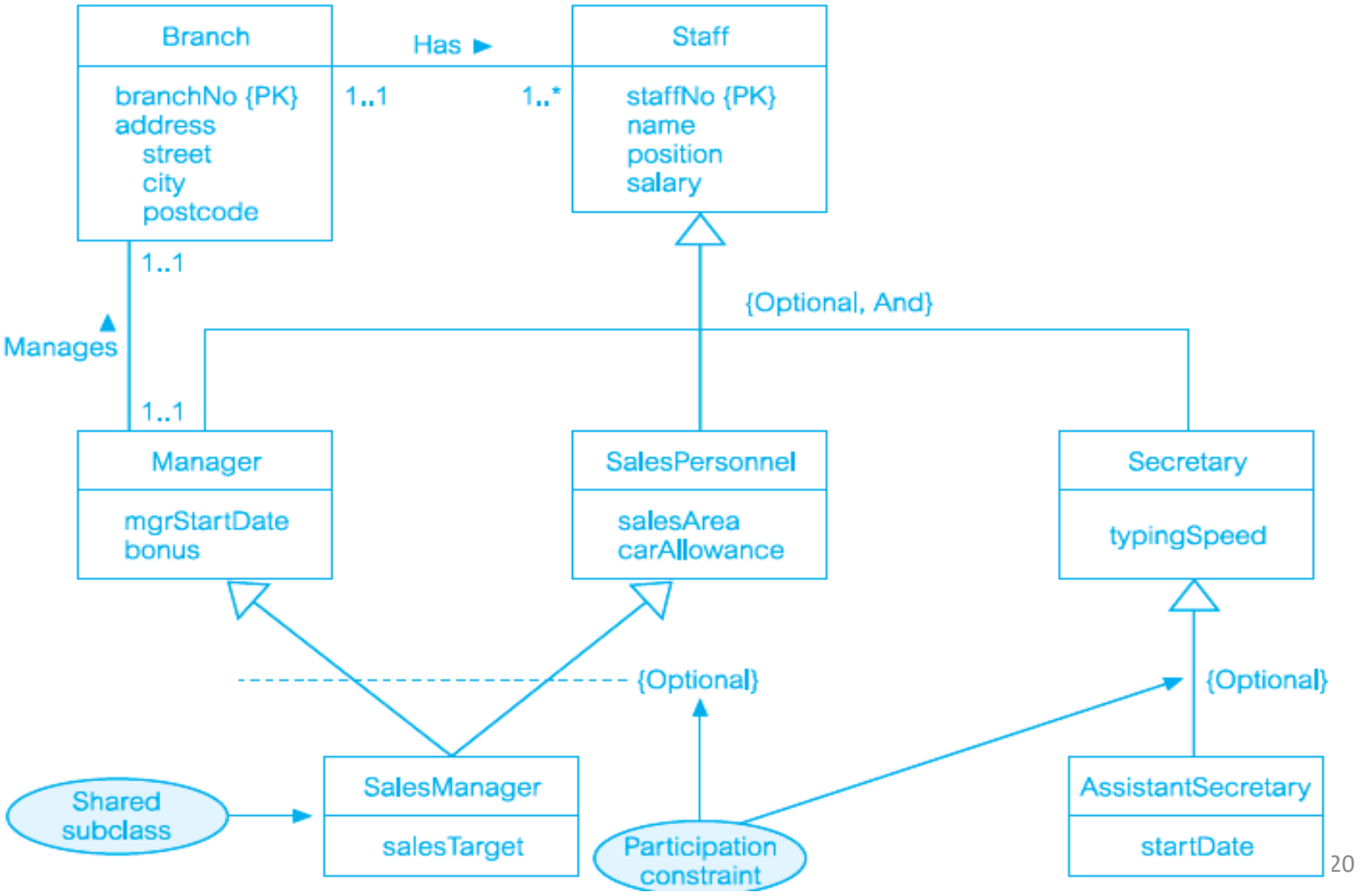
Specialization

- Specialization is a top-down approach to define related subclasses of superclasses.
- The set of subclasses is defined on the basis of some distinguishing characteristics of the entities in the superclass.
- When we identify a set of subclasses of an entity type, we then associate attributes specific to each subclass (where necessary), and also identify any relationships between each subclass and other entity types or subclasses (where necessary).

Specialization

- For example, consider a model where all members of staff are represented as an entity called Staff.
- If we apply the process of specialization on the Staff entity, we attempt to identify differences between members of this entity such as members with distinctive attributes and/or relationships.
- 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.

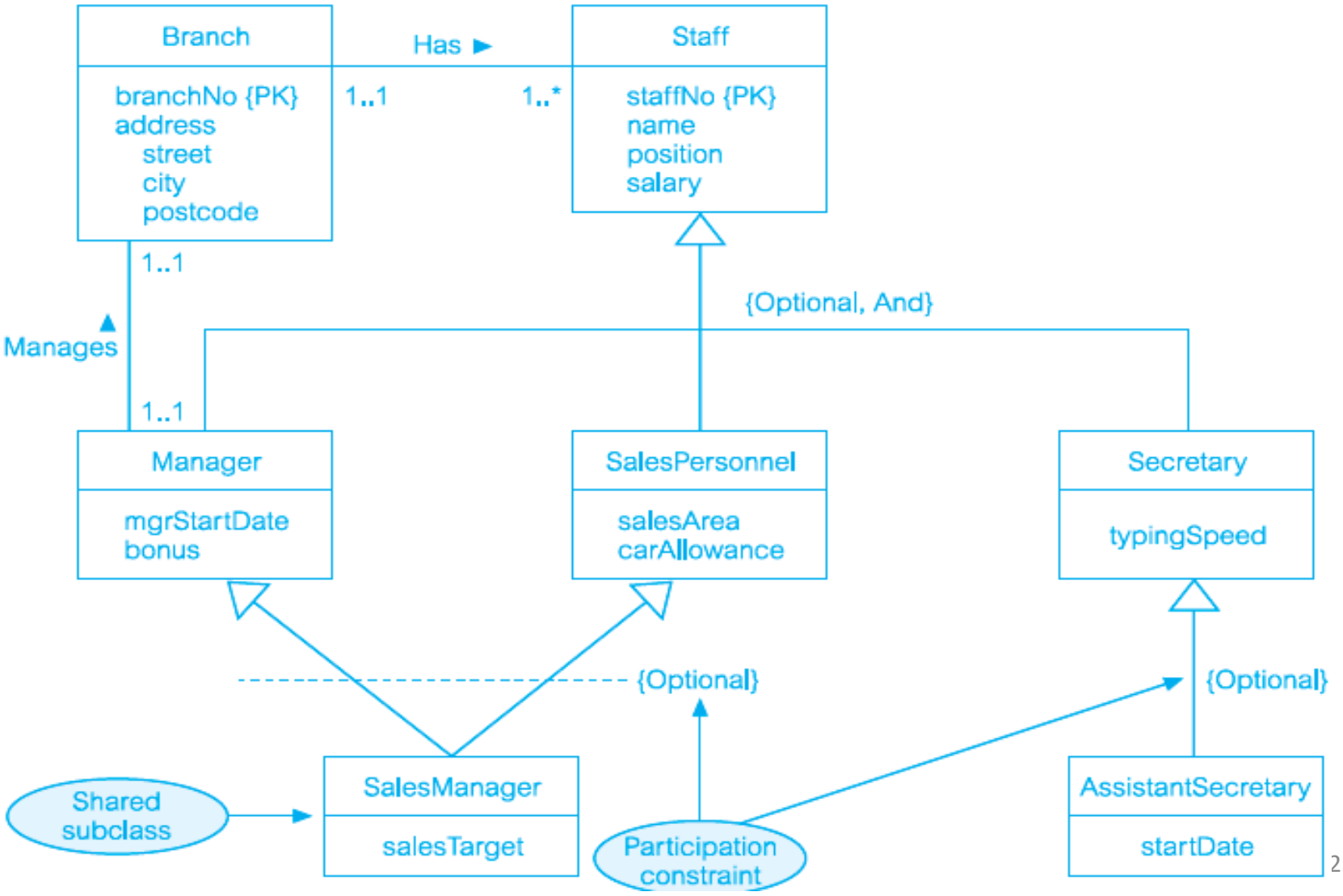
Specialization



Generalization

- The process of generalization is a bottom-up approach, which results in the identification of a generalized superclass from the original entity types.
- 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 these entities share attributes common to all staff, and therefore we identify Manager, SalesPersonnel, and Secretary as subclasses of a generalized Staff superclass.

Generalization



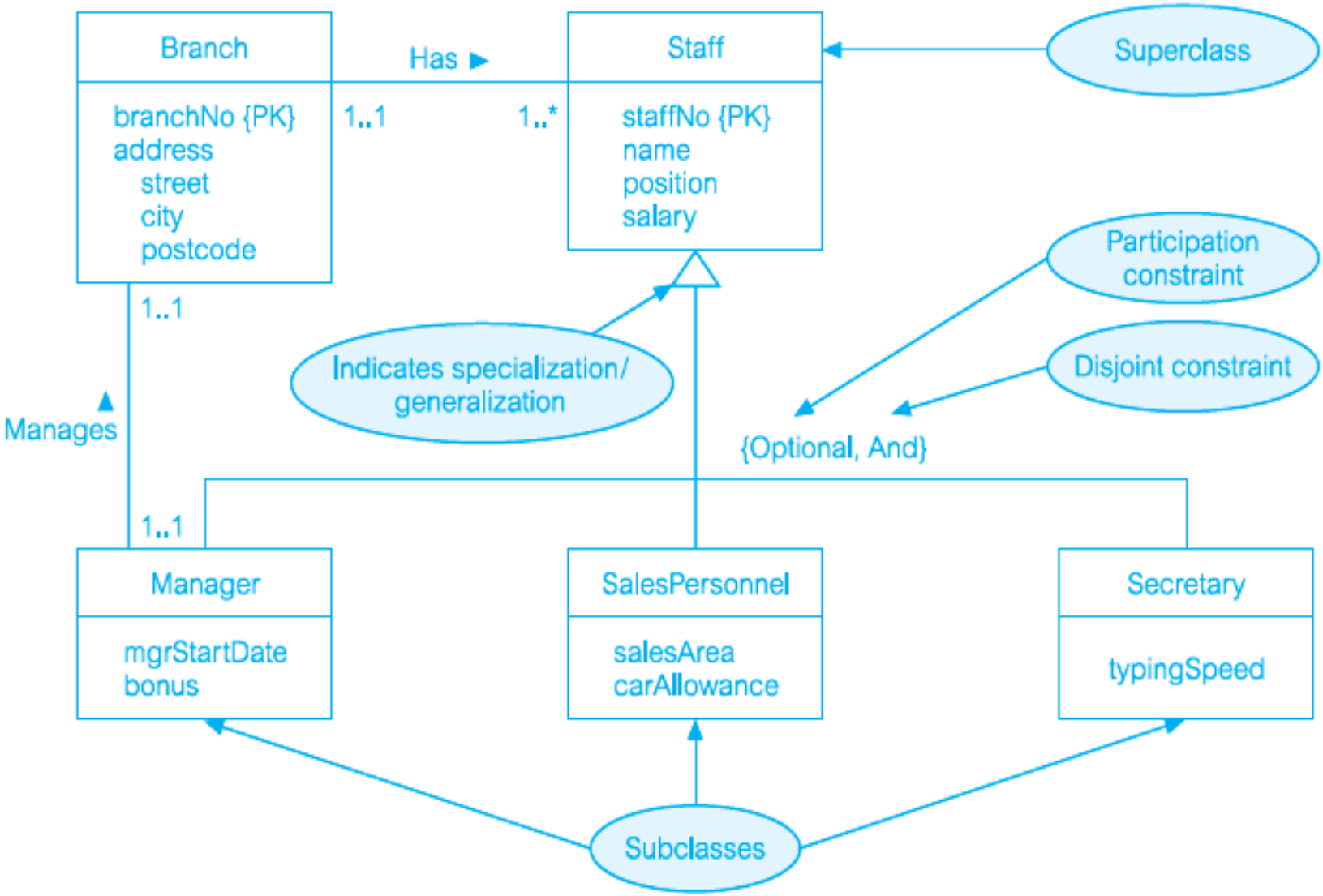
Generalization

- As the process of generalization can be viewed as the reverse of the specialization process, we refer to this modeling concept as ‘specialization/generalization’.

Diagrammatic representation of specialization/generalization

- Entities, are represented as rectangles.
- The subclasses are attached by lines to a triangle that points toward the superclass.
- The label below the specialization/generalization triangle, shown as {Optional, And}, describes the ***constraints*** on the relationship between the superclass and its subclasses.

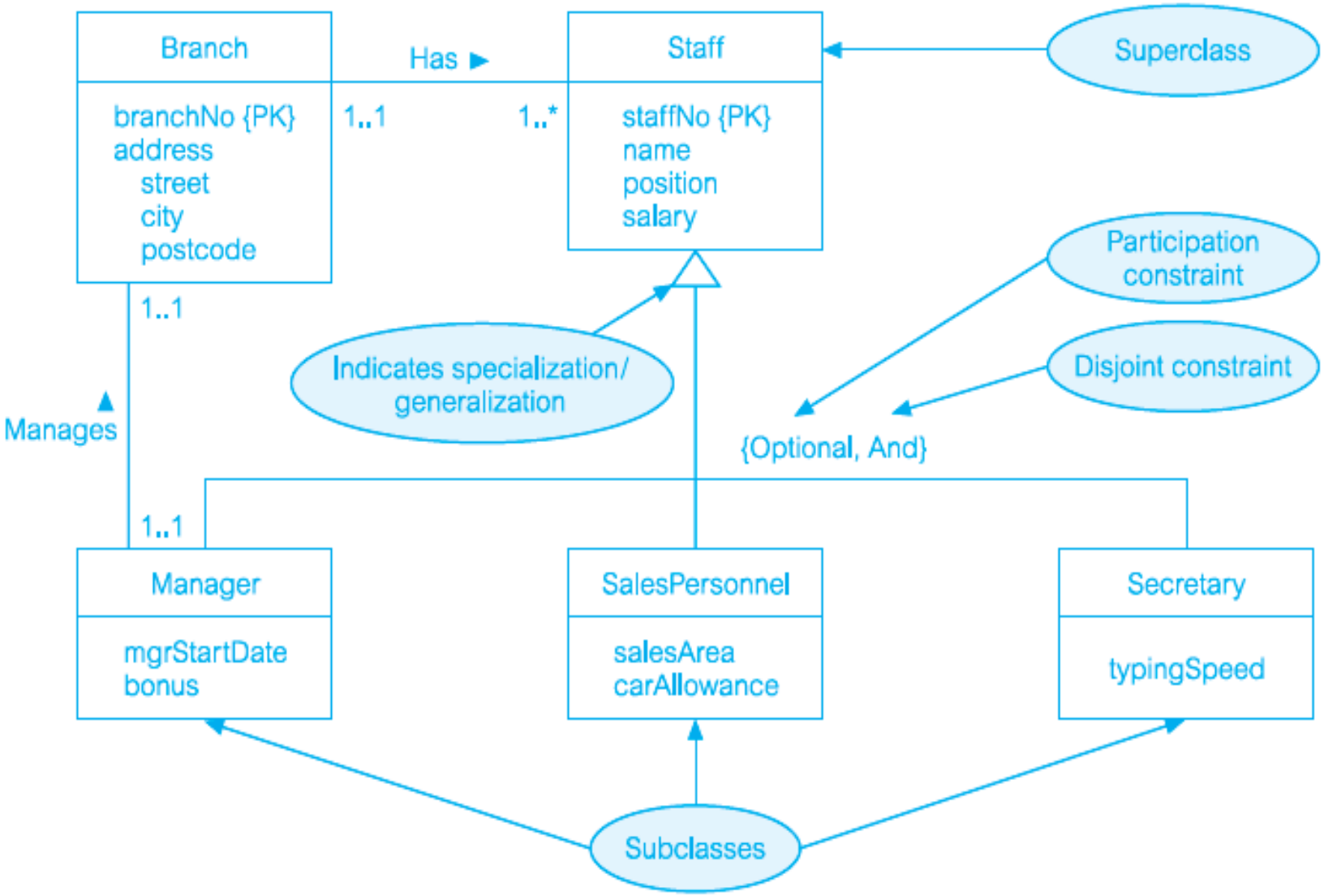
Diagrammatic representation of specialization/generalization



Diagrammatic representation of specialization/generalization

- Attributes that are specific to a given subclass are listed in the lower section of the rectangle representing that subclass.
- For example, salesArea and carAllowance attributes are only associated with the SalesPersonnel subclass, and are not applicable to the Manager or Secretary subclasses.
- Similarly, we show attributes that are specific to the Manager (mgrStartDate and bonus) and Secretary (typingSpeed) subclasses.

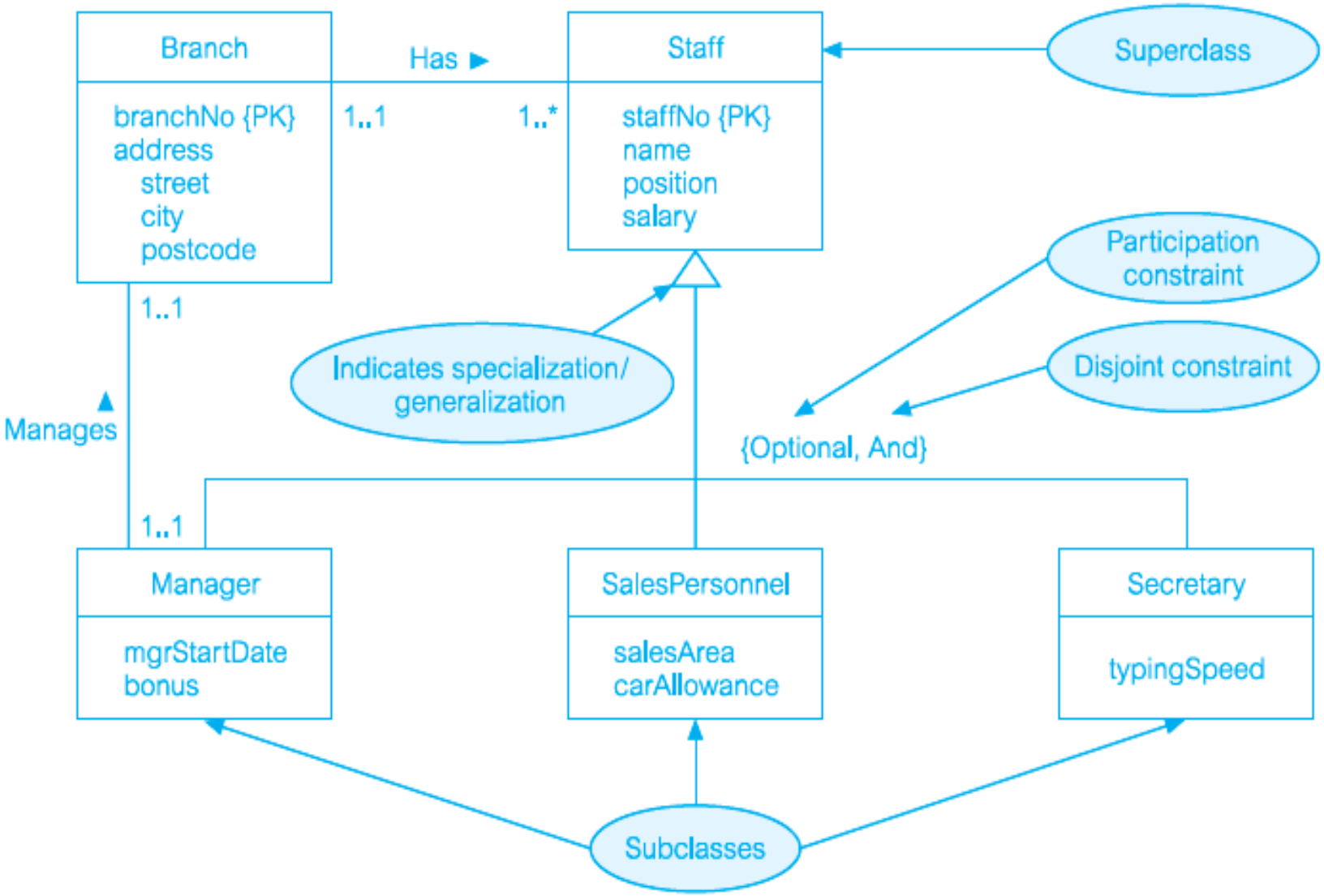
Diagrammatic representation of specialization/generalization



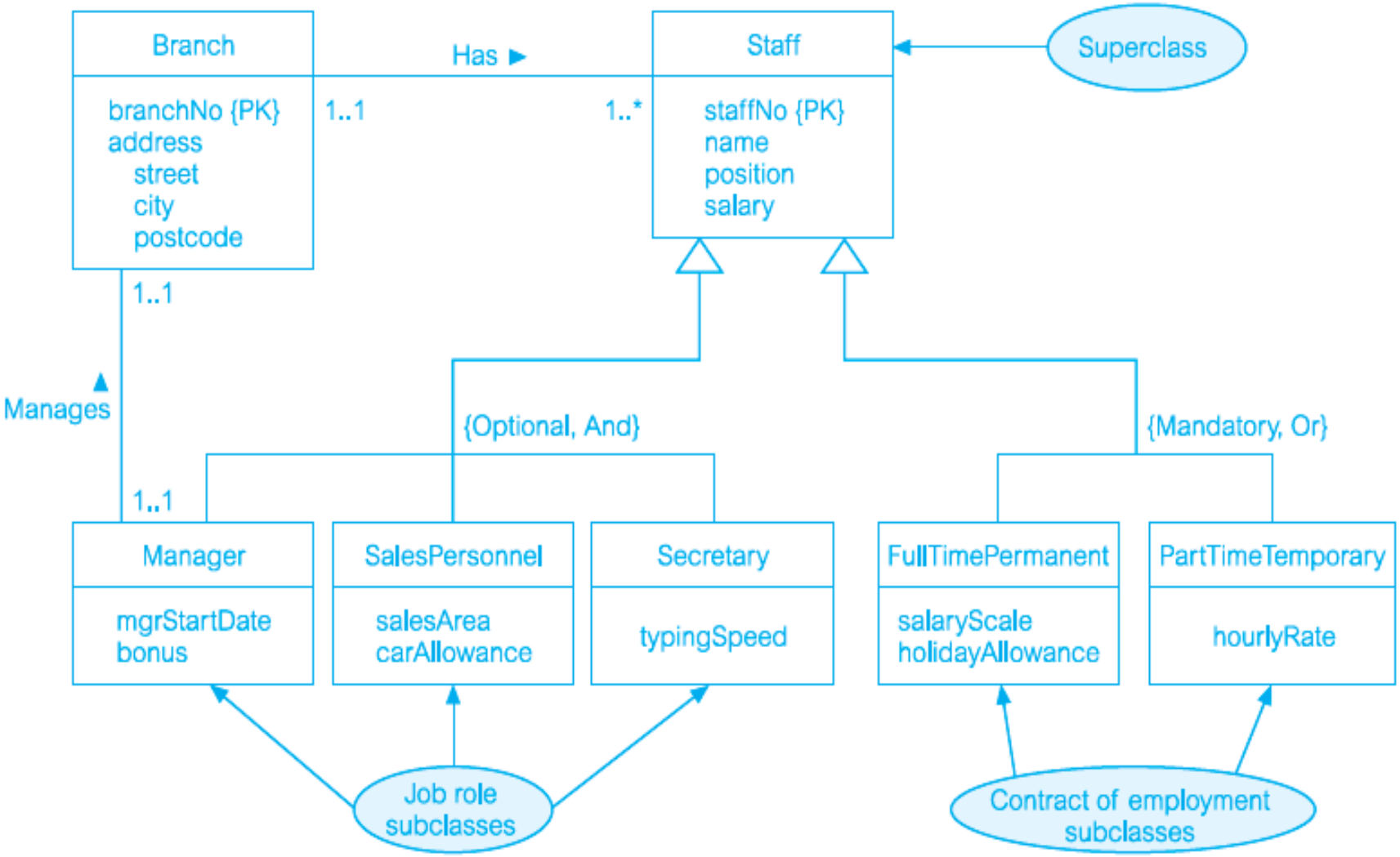
Diagrammatic representation of specialization/generalization

- Attributes that are common to all subclasses are listed in the lower section of the rectangle representing the superclass.
- For example, staffNo, name, position, and salary attributes are common to all members of staff and are associated with the Staff superclass. Note that we can also show ***relationships*** that are only applicable to specific subclasses. For example, in Figure on next slide, the Manager subclass is related to the Branch entity through the ***Manages*** relationship, whereas the Staff superclass is related to the Branch entity through the ***Has*** relationship.

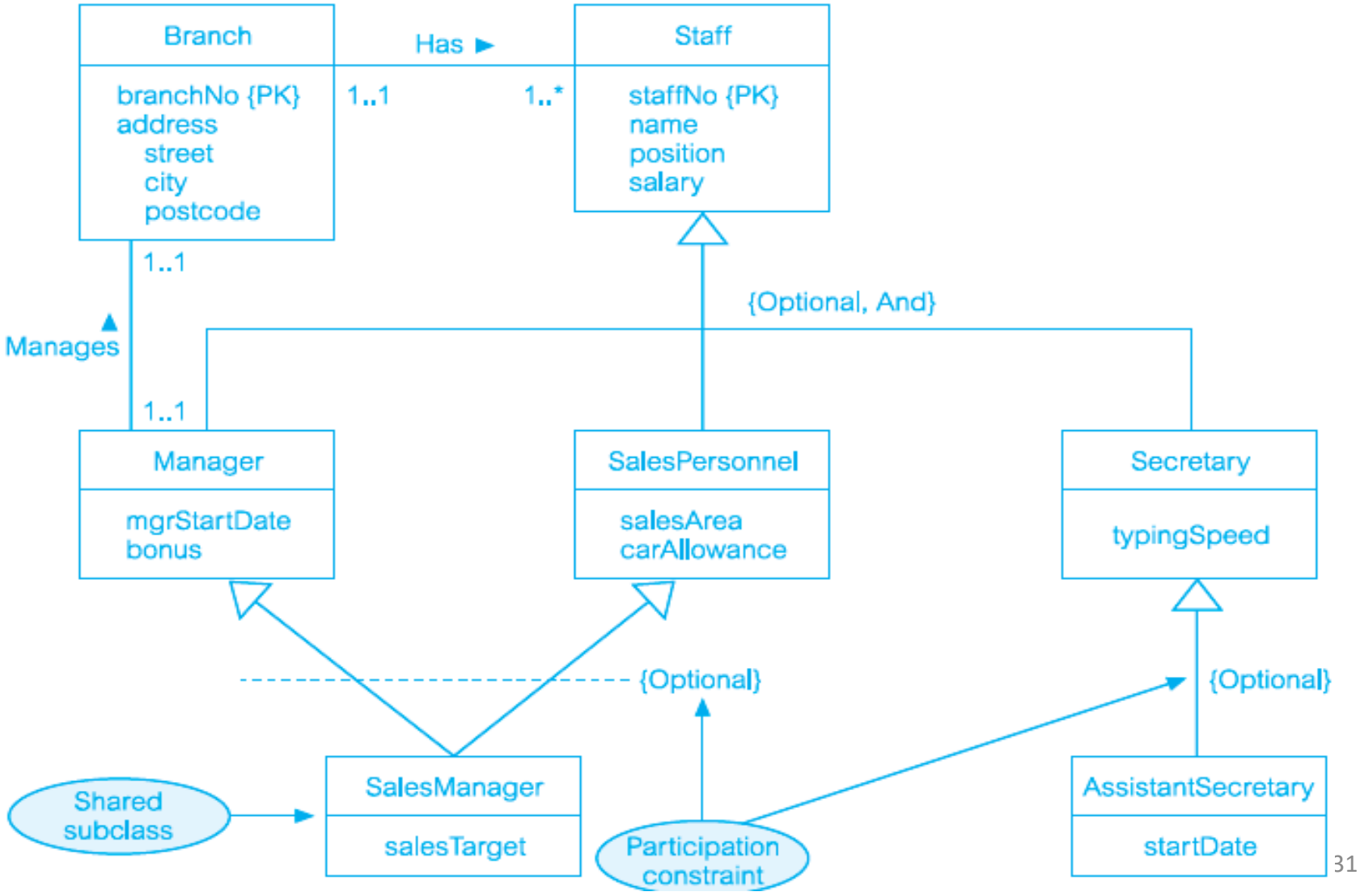
Specialization/generalization of the Staff entity into subclasses representing job roles



Specialization/generalization of the Staff entity into subclasses representing job roles and contracts of employment



Specialization/generalization of the Staff entity into job roles including a *shared* subclass called *SalesManager* and a subclass called Secretary with its own subclass called AssistantSecretary



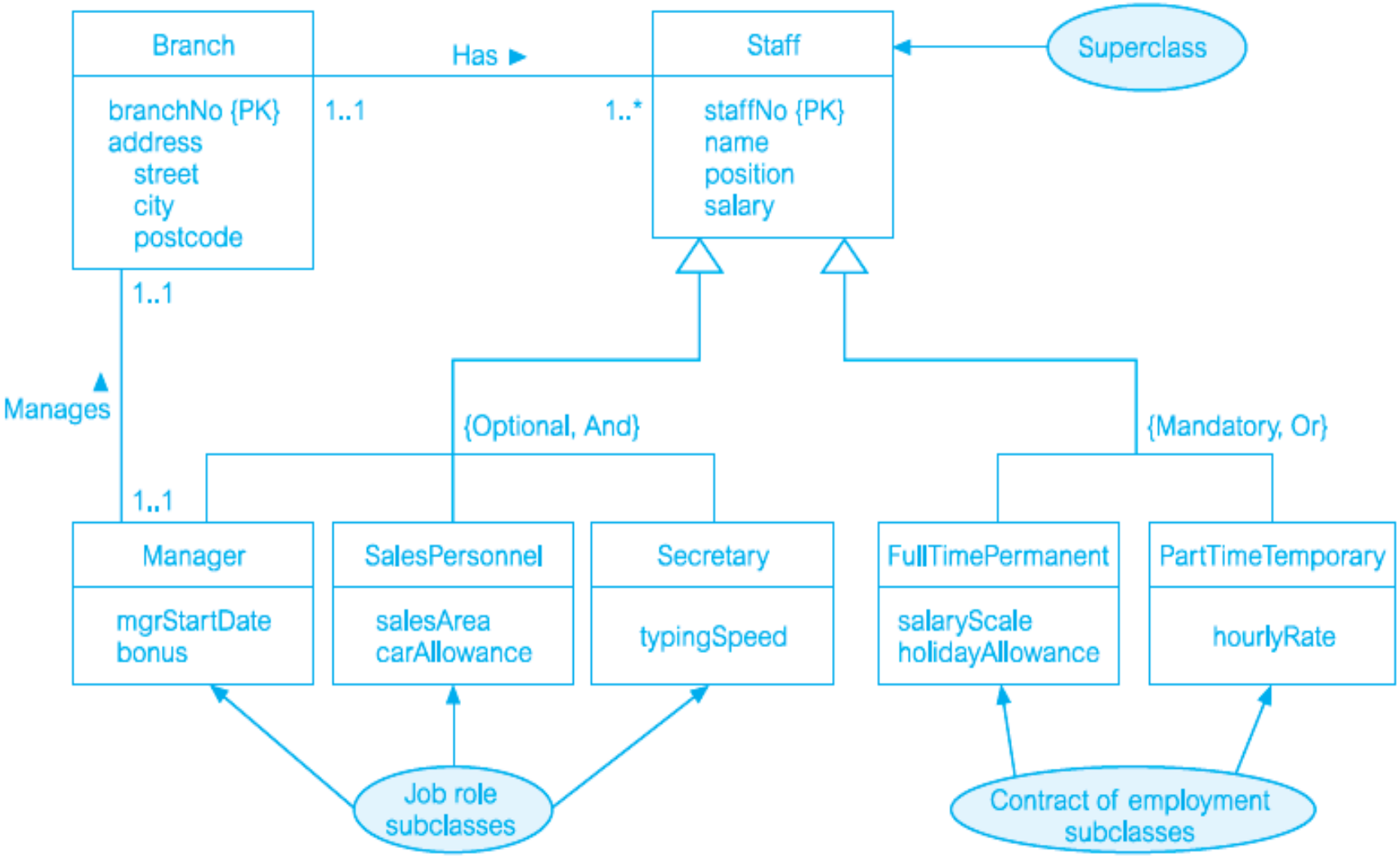
Constraints on specialization/ generalization

- Two types:
  - Participation Constraints
  - Disjoint Constraints
- Participation Constraint
  - Determines whether every occurrence in the superclass must participate as a member of a subclass or not.
  - May be mandatory or optional

Mandatory Participation constraint

- A superclass/subclass relationship with **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 below the triangle that points towards the superclass.
- For example, in Figure on next slide, the contract of employment specialization/generalization is mandatory participation, which means that every **member of staff** must have a contract of employment (i.e. a member of staff must be either Permanent or Temporary employee)

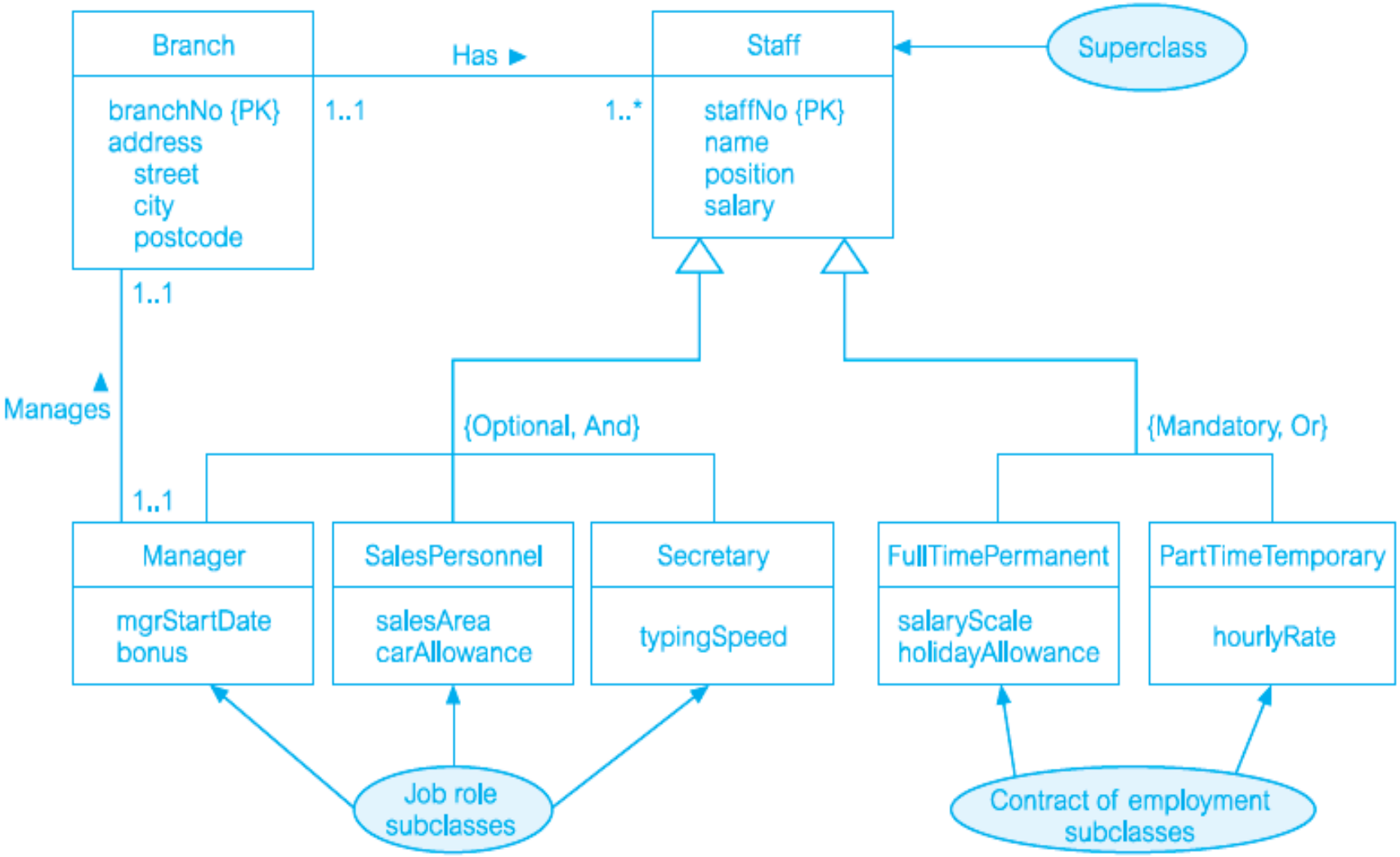
Specialization/generalization of the Staff entity into subclasses representing job roles and contracts of employment



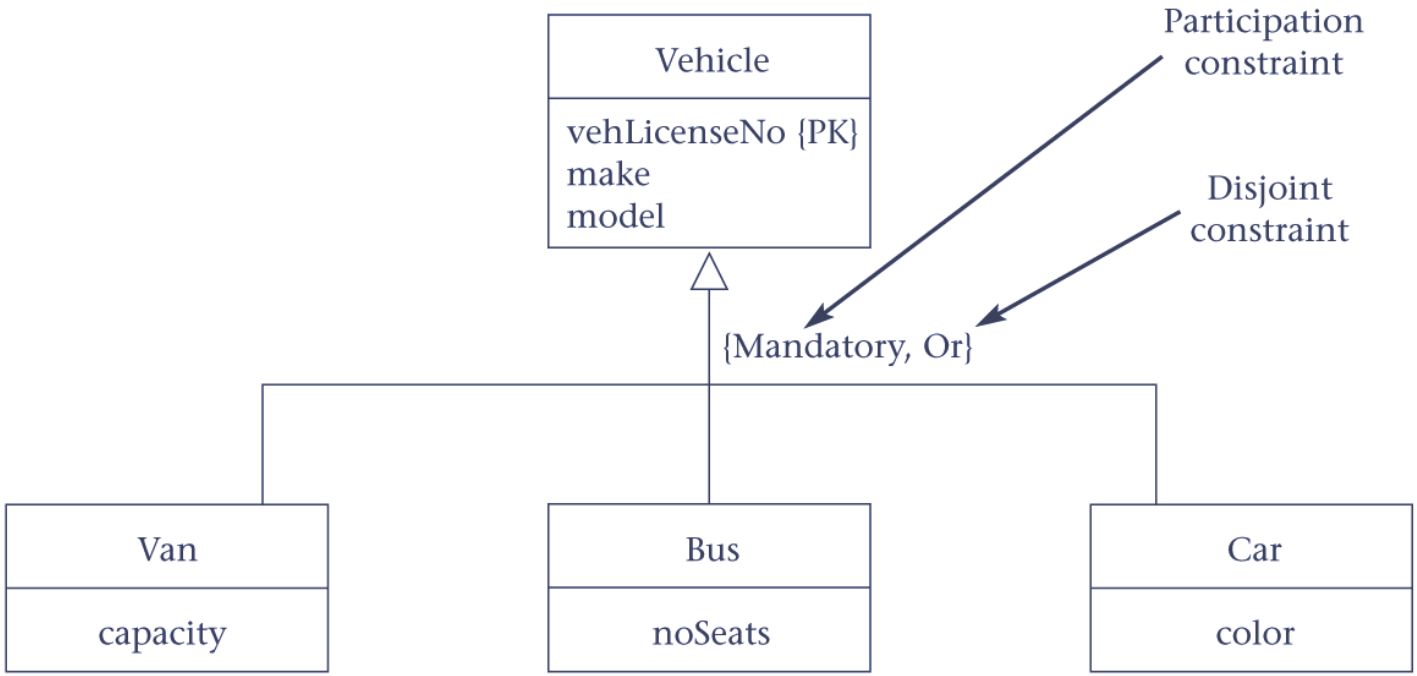
Optional Participation constraint

- A superclass/subclass relationship with **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 below the triangle that points towards the superclass.
- For example, in Figure on next slide, the **job role** specialization/generalization has optional participation, which means that a **member of staff** need not have an additional job role such as a **Manager, Sales Personnel, or Secretary**

Specialization/generalization of the Staff entity into subclasses representing job roles and contracts of employment



Vehicle entity into vehicle types



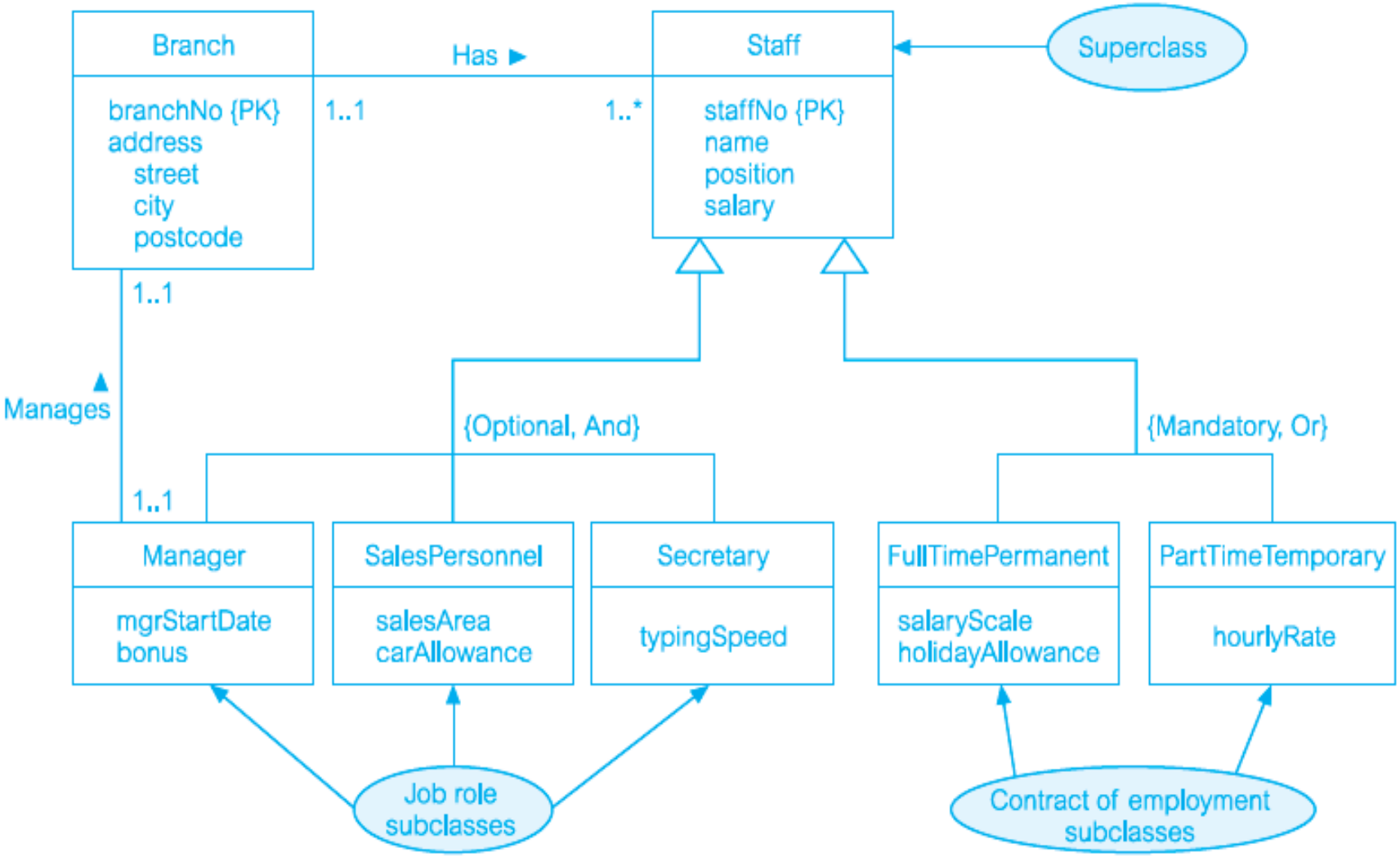
Constraints on specialization / generalization

- **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.**
  - **May be disjoint or nondisjoint**

Disjoint constraint

- The disjoint constraint only applies when a superclass has more than one subclasses.
- If the subclasses are **disjoint**, then an entity occurrence of the superclass can be a member of only ***one*** of the subclasses.
- To represent a disjoint superclass/subclass relationship, 'Or' is placed next to the participation constraint within the curly brackets.
- For example, in Figure on next slide, the subclasses of the contract of employment specialization/generalization is disjoint, which means that a member of staff must have a full-time permanent *or* a part-time temporary contract, but not both.

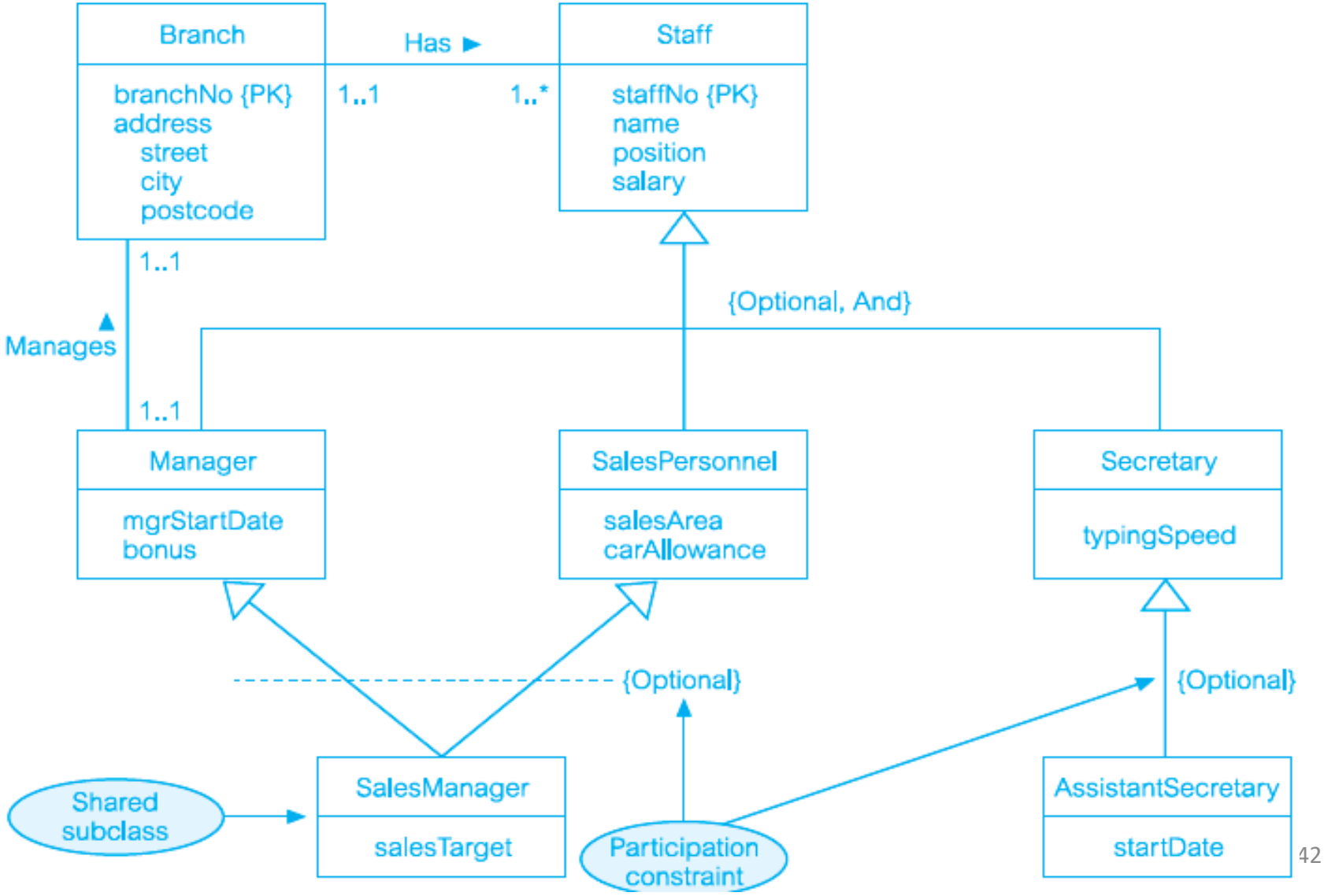
Specialization/generalization of the Staff entity into subclasses representing job roles and contracts of employment



Nondisjoint constraint

- If subclasses of a specialization/generalization are not disjoint (called **nondisjoint**), then an entity occurrence of the superclass may be a member of **more than one** subclass.
- To represent a nondisjoint superclass/subclass relationship, 'And' is placed next to the participation constraint within the curly brackets.
- For example, in Figure on **previous slide**, the **job role** specialization/ generalization is nondisjoint, which means that an entity occurrence can be a member of both the Manager and SalesPersonnel subclasses.
- This is confirmed by the presence of the shared subclass called SalesManager shown in Figure on next slide.
- Note that it **is not necessary to include the disjoint constraint** for hierarchies that have a **single subclass** at a given level and for this reason only the **participation constraint is shown** for the SalesManager and AssistantSecretary subclasses of Figure on next slide.

Specialization/generalization of the Staff entity into job roles including a shared subclass called SalesManager and a subclass called Secretary with its own subclass called AssistantSecretary

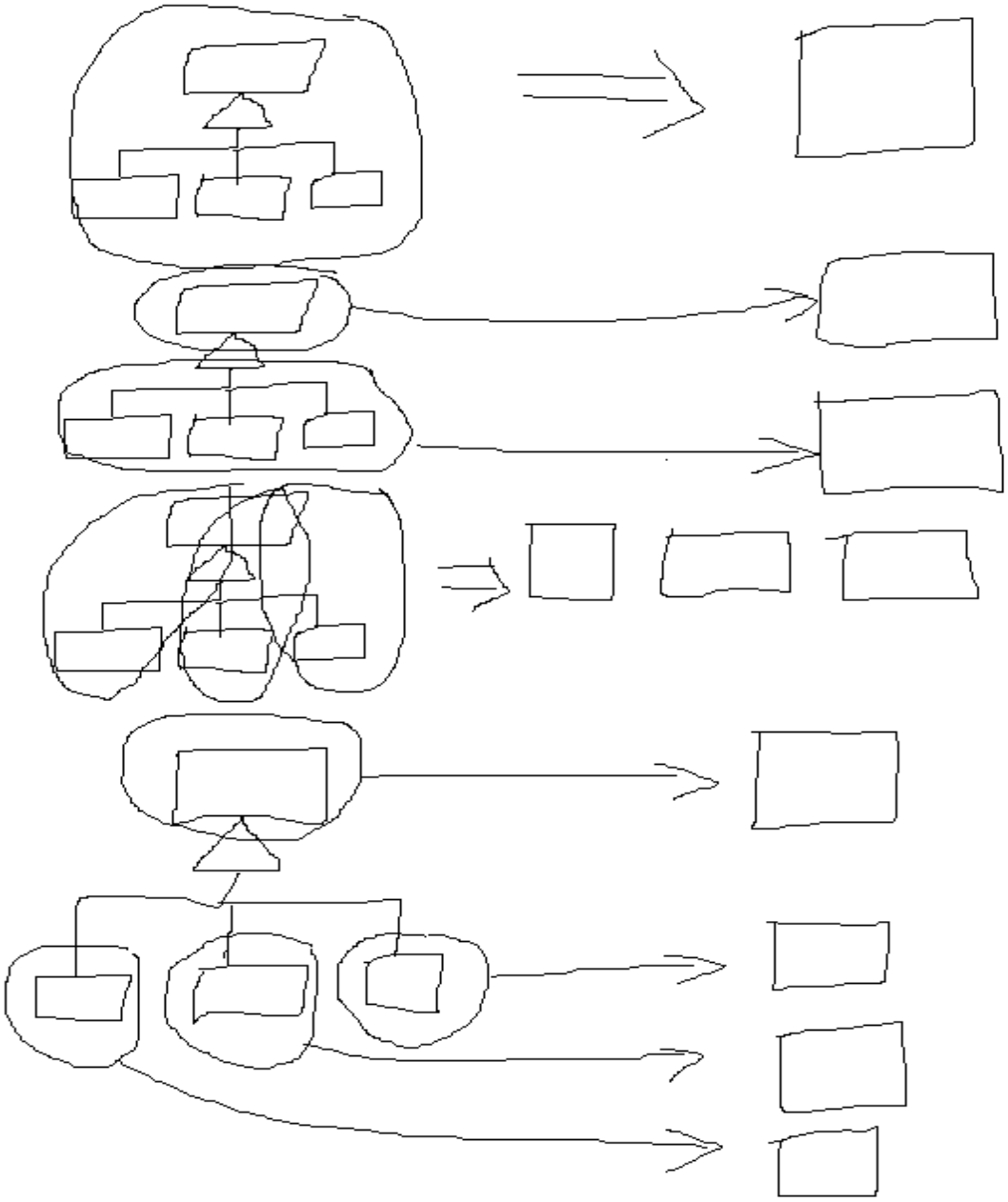


Constraints on specialization / generalization

- There are four categories of constraints of specialization and generalization:
  - mandatory and disjoint {Mandatory, Or}
  - optional and disjoint {Optional, Or}
  - mandatory and nondisjoint {Mandatory, And}
  - optional and nondisjoint {Optional, And}

Creating tables to represent specialization/generalization

Participation constraint	Disjoint constraint	Tables required
Mandatory	Nondisjoint {And}	Single table
Optional	Nondisjoint {And}	Two tables: one table for superclass and one table for all subclasses
Mandatory	Disjoint {Or}	Many tables: one table for each combined superclass/subclass
Optional	Disjoint {Or}	Many tables: one table for superclass and one table for each subclass



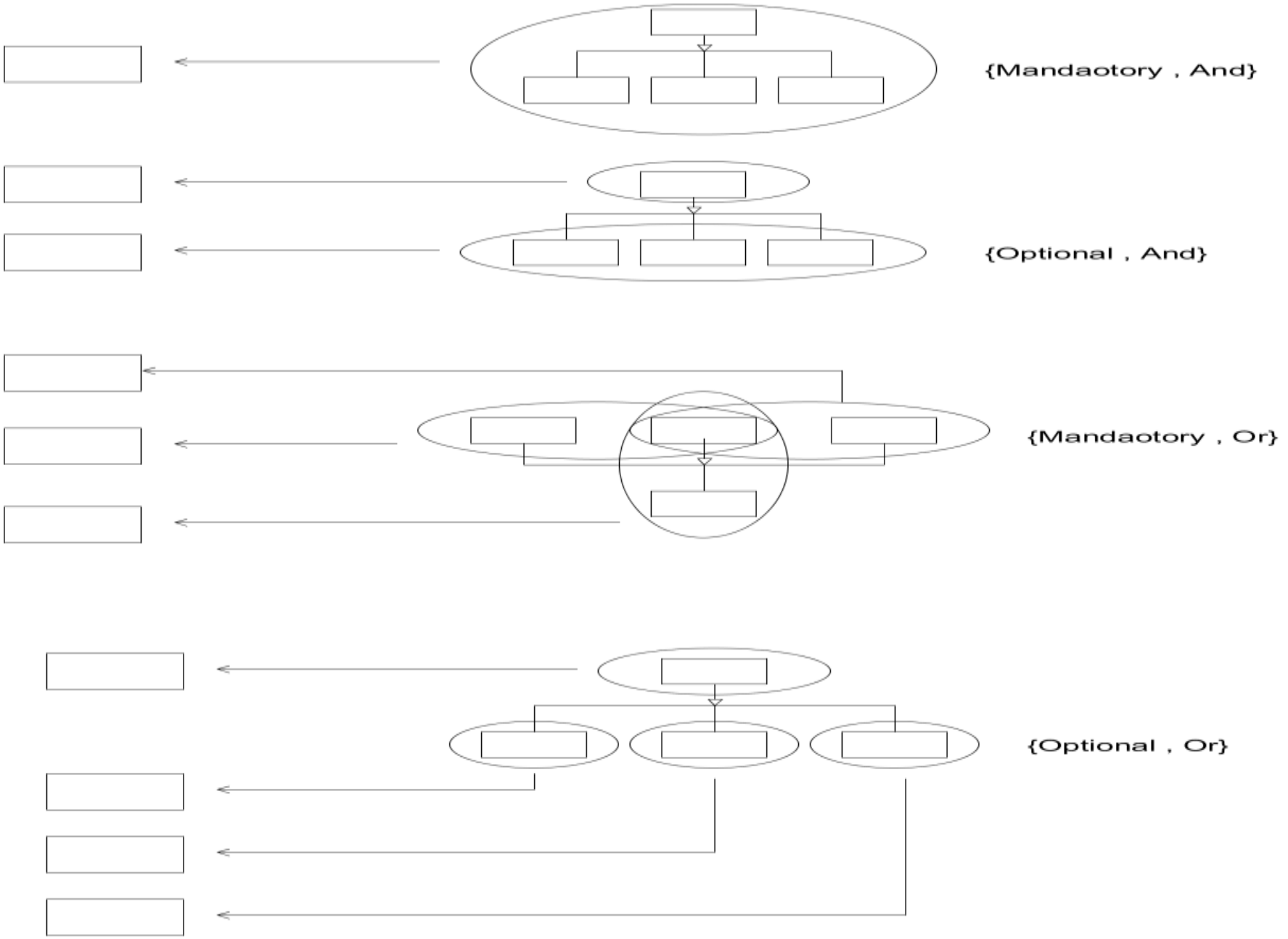
{Mandatory, And}

{Optional, And}

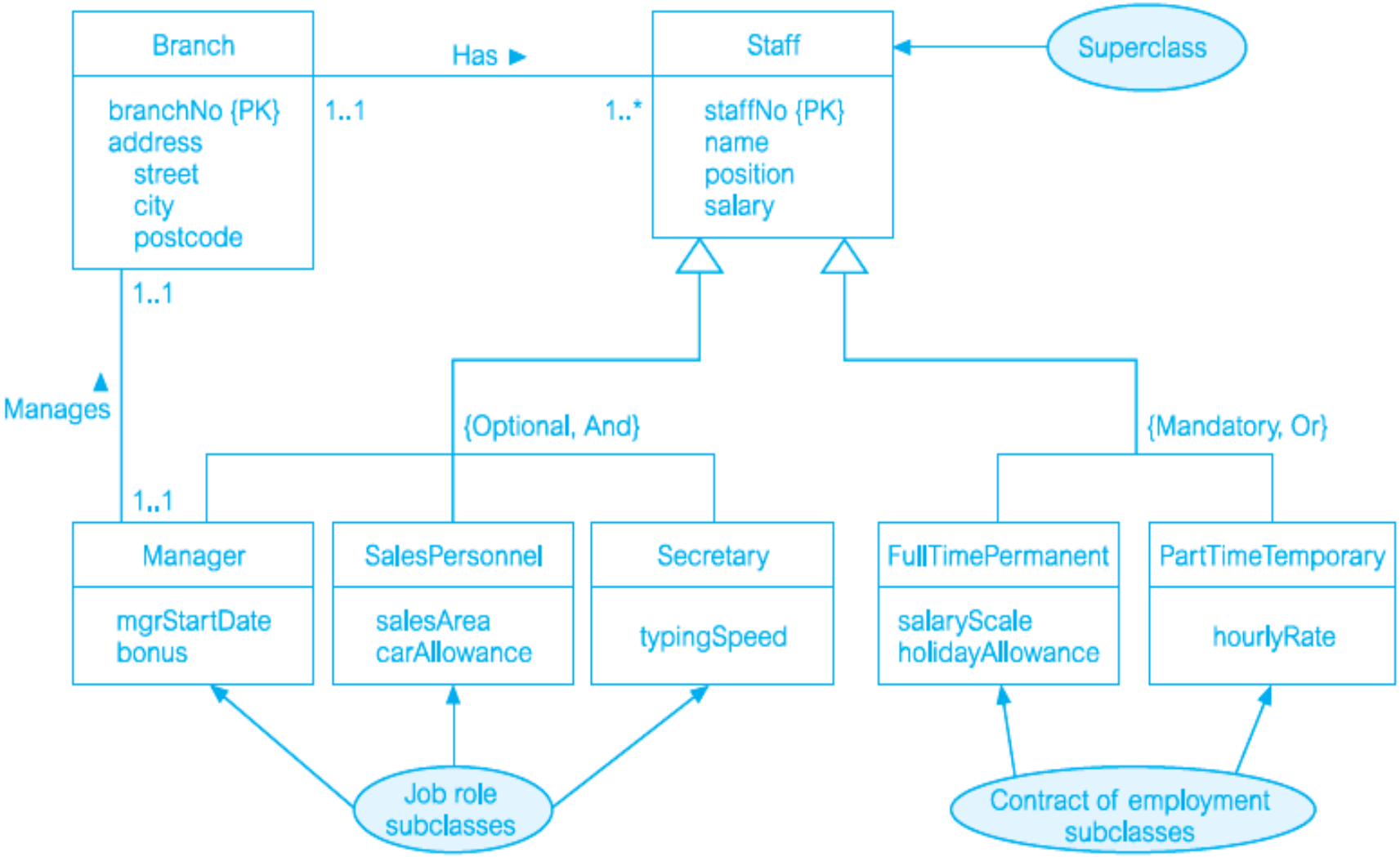
{Mandatory, Or}

{Optional, Or}

Creating tables to represent specialization/generalization



Specialization/generalization of the Staff entity into subclasses representing job roles and contracts of employment



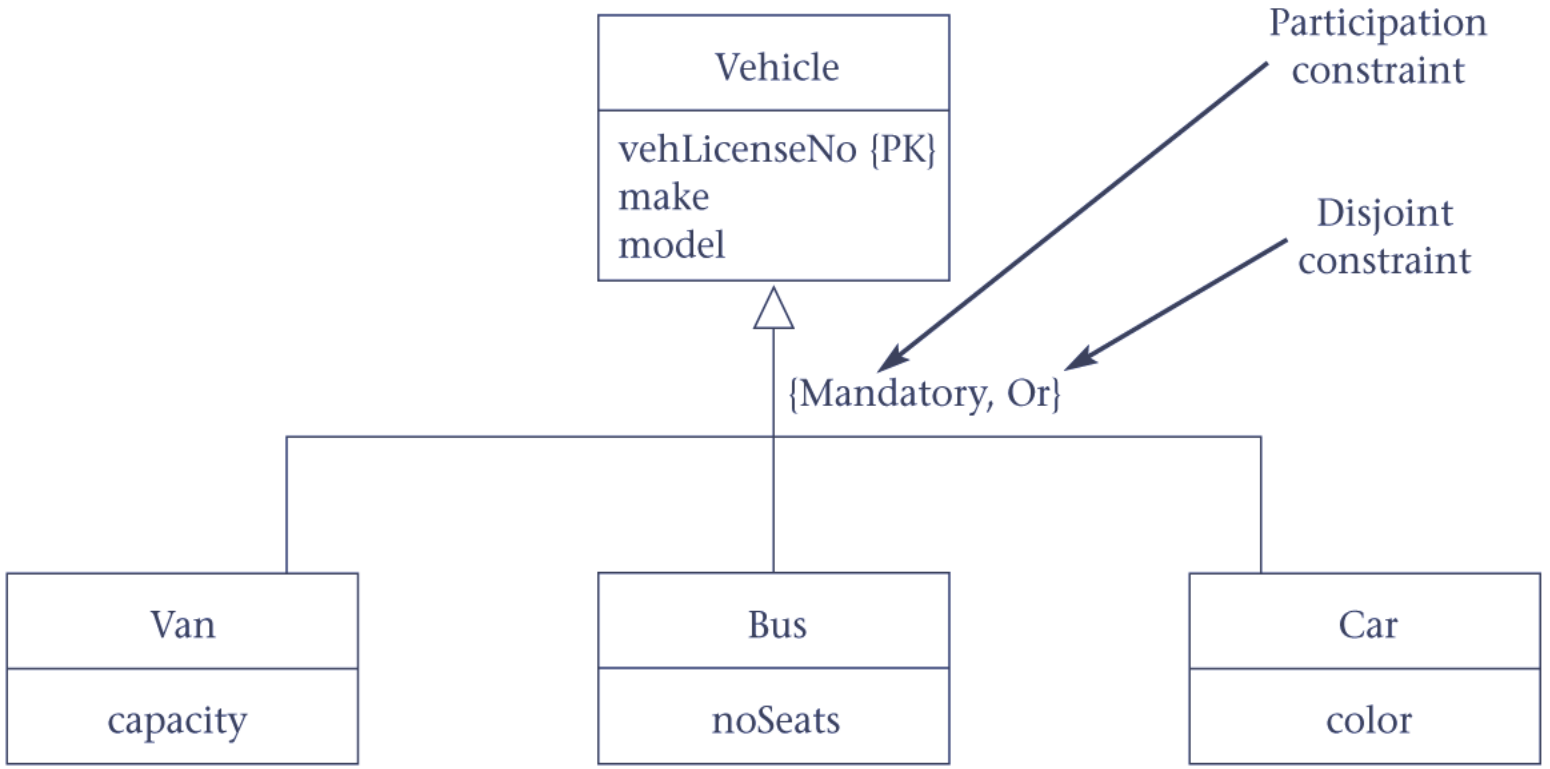
Tables representing Staff and the Branch entities for {Optional, And}

Staff superclass
staff (staffNo, name, position, salary, branchNo)
Primary Key staffNo
Foreign Key branchNo **references** Branch(branchNo)

Staff subclasses
AllStaffSubclasses (subclassStaffNo, bonus, salesArea, vehLicenseNo, carAllowance, typingSpeed)
Primary Key subclassStaffNo
Foreign Key subclassStaffNo **references** Staff(staffNo)

Branch
Branch (branchNo, street, city, state, zipCode, mgrStaffNo)
Primary Key branchNo
Foreign Key mgrStaffNo **references** AllStaffSubclasses(subclassStaffNo)

Vehicle entity into vehicle types



Tables representing the Vehicle entity for {Mandatory, Or}

Van subclass
Van (vehLicenseNo, make, model, capacity)
Primary Key vehLicenseNo

Car subclass
Car (vehLicenseNo, make, model, color)
Primary Key vehLicenseNo

Bus subclass
Bus (vehLicenseNo, make, model, noSeats)
Primary Key vehLicenseNo