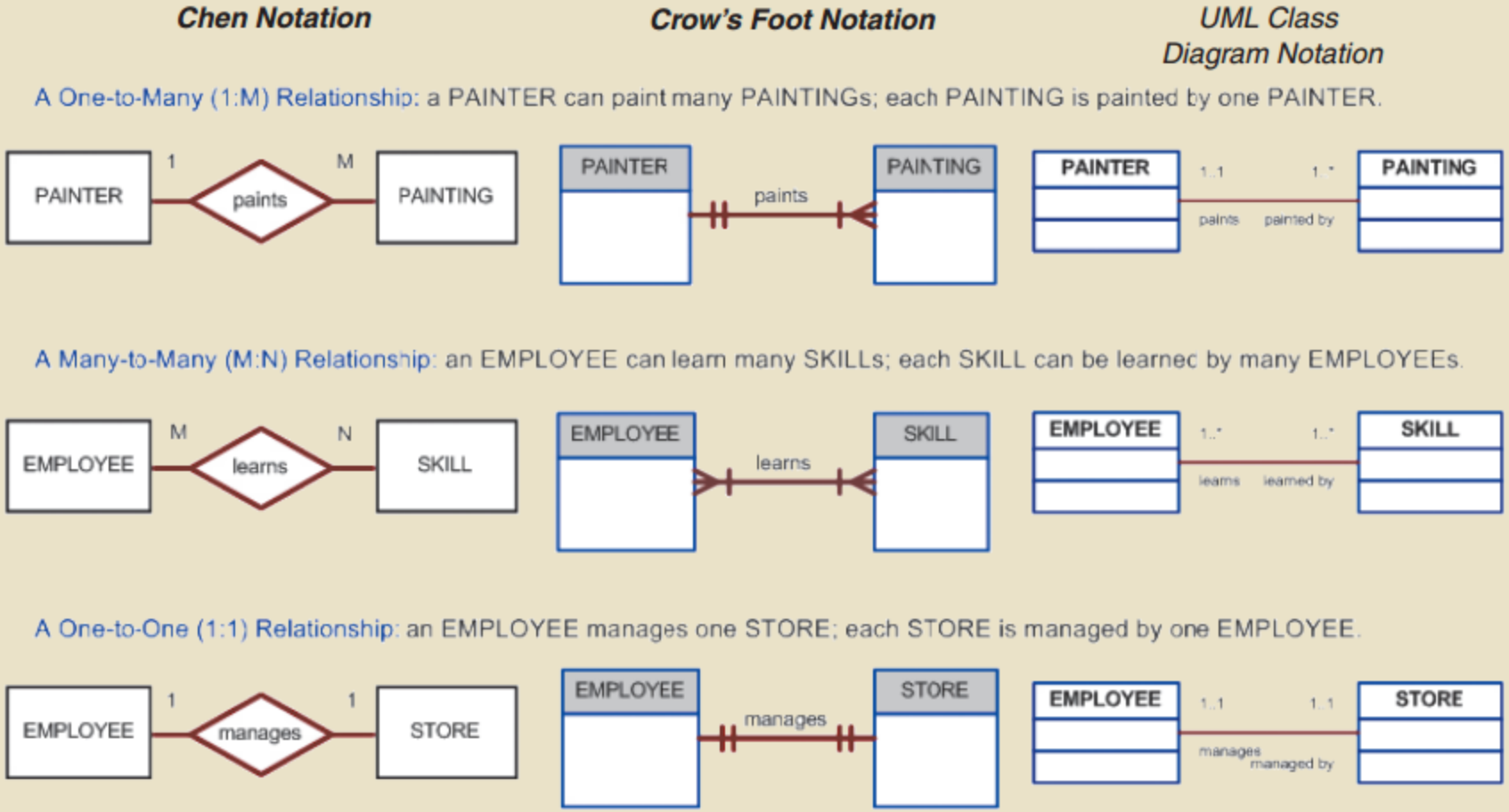
# Enhanced Entity Relationship (EER)

# Enhanced ER (EER) Model

- Semantic concepts are incorporated into the original ER model and called the ***Enhanced*** Entity-Relationship (EER) model.
  - Specialization/generalization
  - Aggregation
  - Composition.

## FIGURE 2.3 THE ER MODEL NOTATIONS



# UML vs ERD

| | UML | ERD |
|---|---|---|
| Full form | Unified Modeling Language | Entity Relationship Diagram |
| Definition | An easy to read system of symbols, shapes, and notations used in software system modeling and planning | A diagram that shows real-world items that exist in a database, and uses symbols and shapes to show relationships, attributes, and other important details |
| Key attributes | Class diagrams<br>Object diagrams<br>Sequence diagrams<br>Activity diagrams<br>Communication diagrams | Entities<br>Attributes<br>Cardinality<br>Ordinality<br>Number of relationship instances |
| Uses | Plan and model software systems<br>Show how entities operate within a system, with all possible interactions | Plan databases<br>Ensure all entities function properly<br>Defines attributes of entities |

# Specialization / Generalization

- **Superclass**
  - An entity type that includes one or more distinct subgroupings of its occurrences.

- **Subclass**
  - A distinct subgrouping of occurrences of an entity type.

# Specialization / Generalization

- Superclass/subclass relationship is **one-to-one (1:1).**
- Superclass may contain *overlapping* or *distinct* subclasses.
- Not all members of a superclass need be a member of a subclass.

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

# Specialization / Generalization

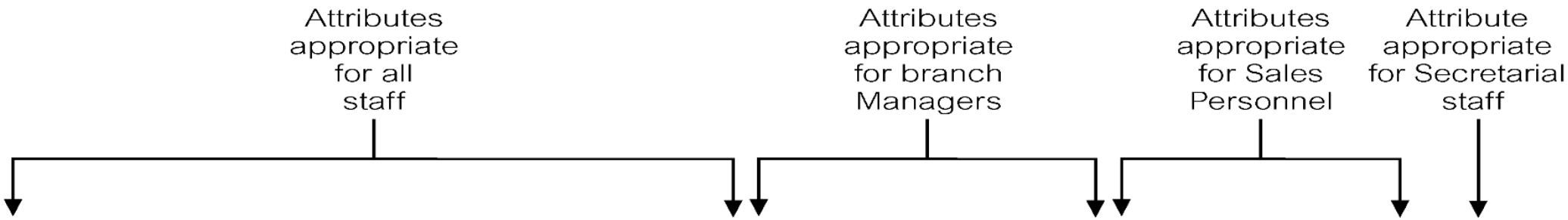
## Specialization

- Process of maximizing differences between members of an entity by identifying their ***distinguishing*** characteristics.

## Generalization

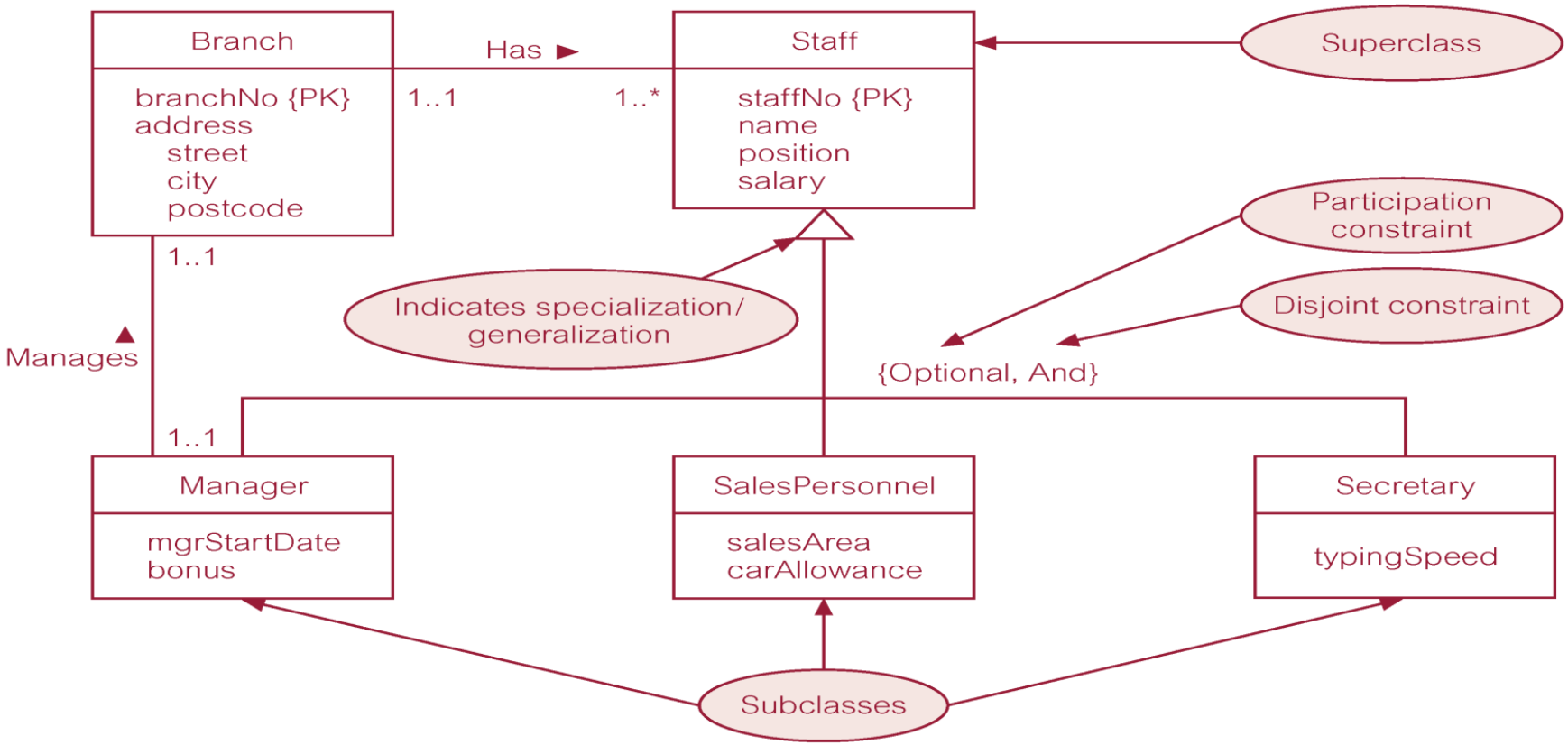
- Process of minimizing differences between entities by identifying their ***common*** characteristics.

# Staff Table with details of all staff

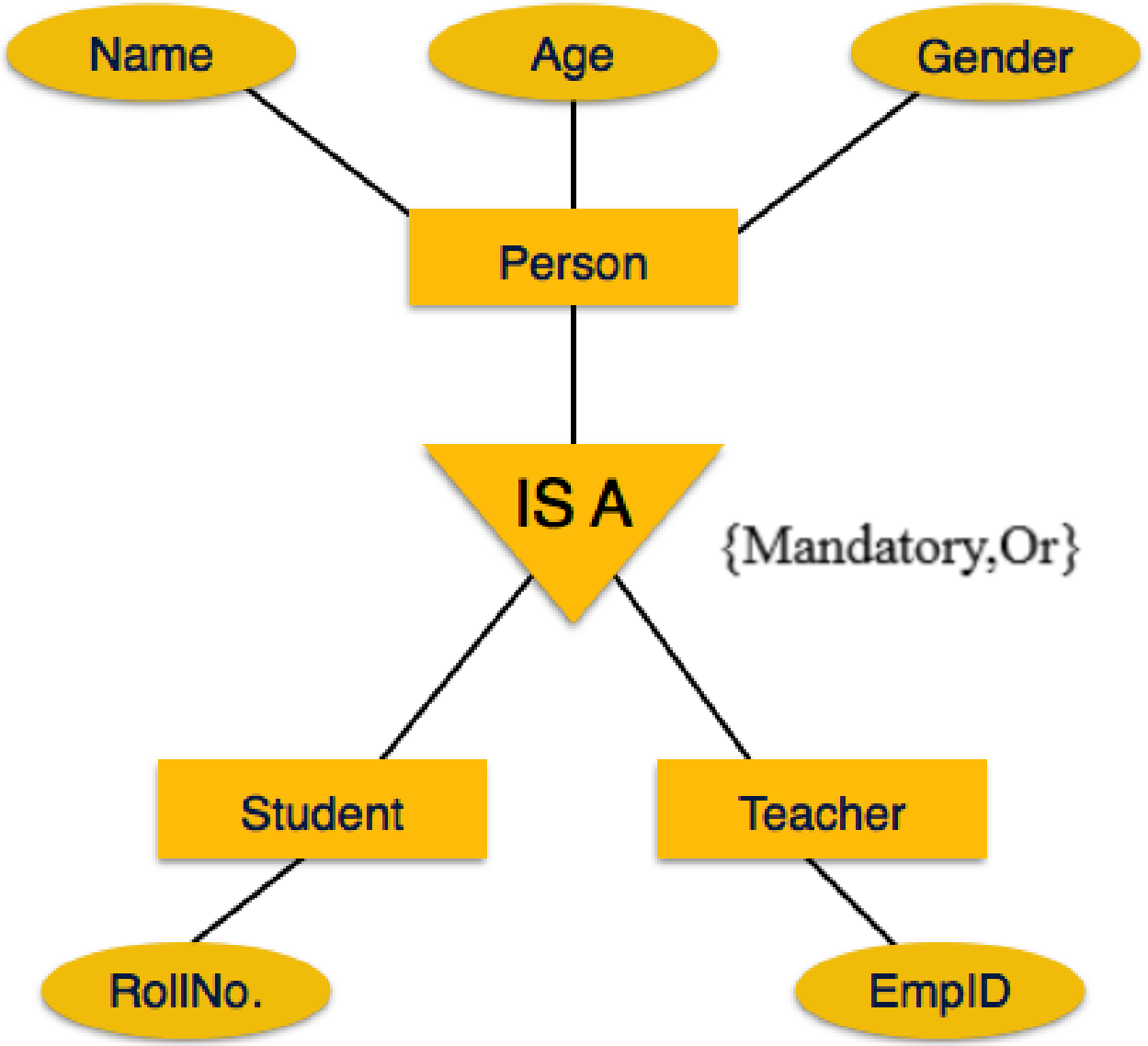



| staffNo | name | position | salary | mgrStartDate | bonus | sales Area | car Allowance | typing Speed |
|---|---|---|---|---|---|---|---|---|
| SL21 | John White | Manager | 30000 | 01/02/95 | 2000 | | | |
| SG37 | Ann Beech | Assistant | 12000 | | | | | |
| SG66 | Mary Martinez | Sales Manager | 27000 | | | 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 | | | |

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



# IS A relation – Chen's notation



# Constraints on Specialization / Generalization

- Two constraints that may apply to a specialization/generalization:
  - Participation constraints
  - Disjoint constraints.

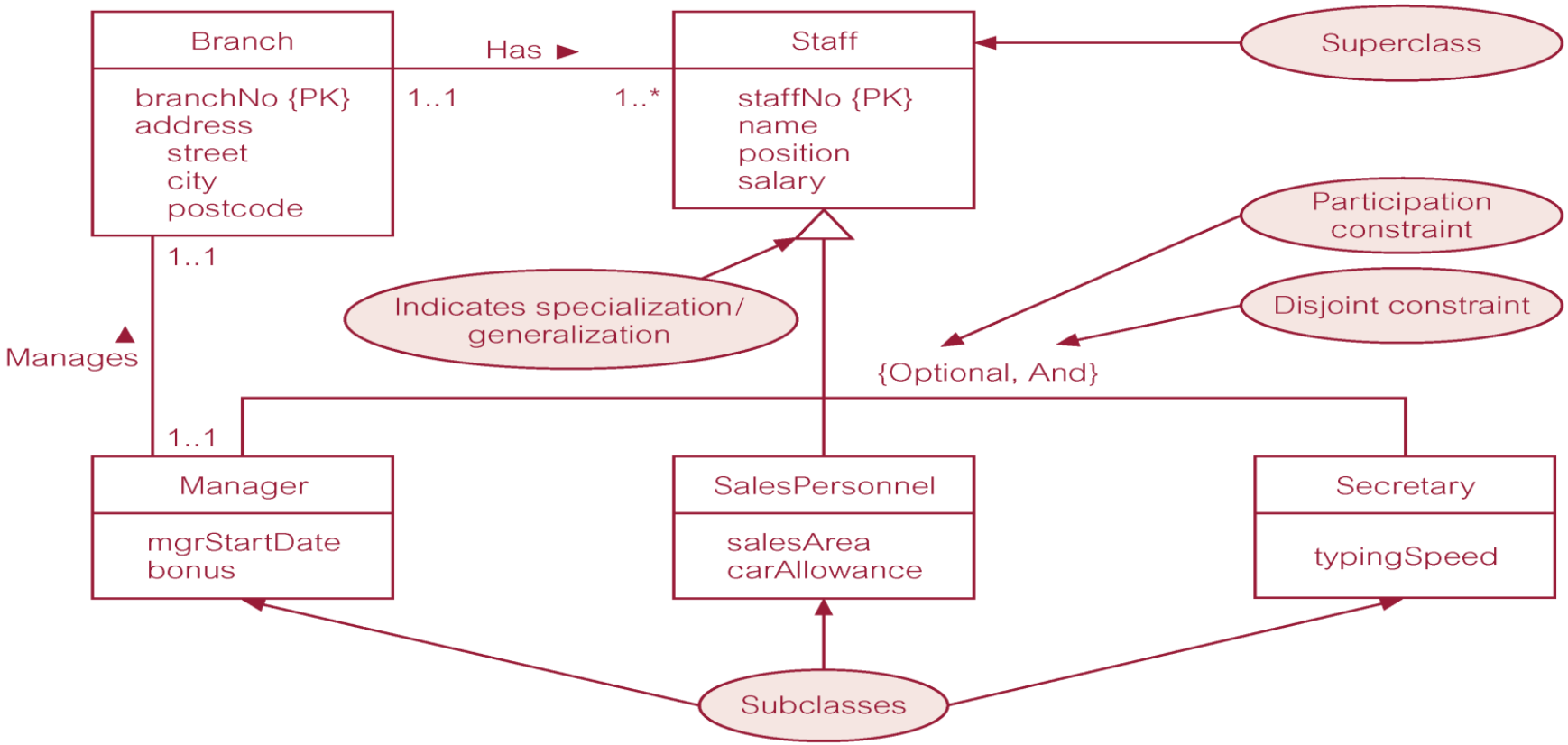
- **Categories of constraints**
  - All combinations of Participation and Disjoint

# Constraints on Specialization / Generalization

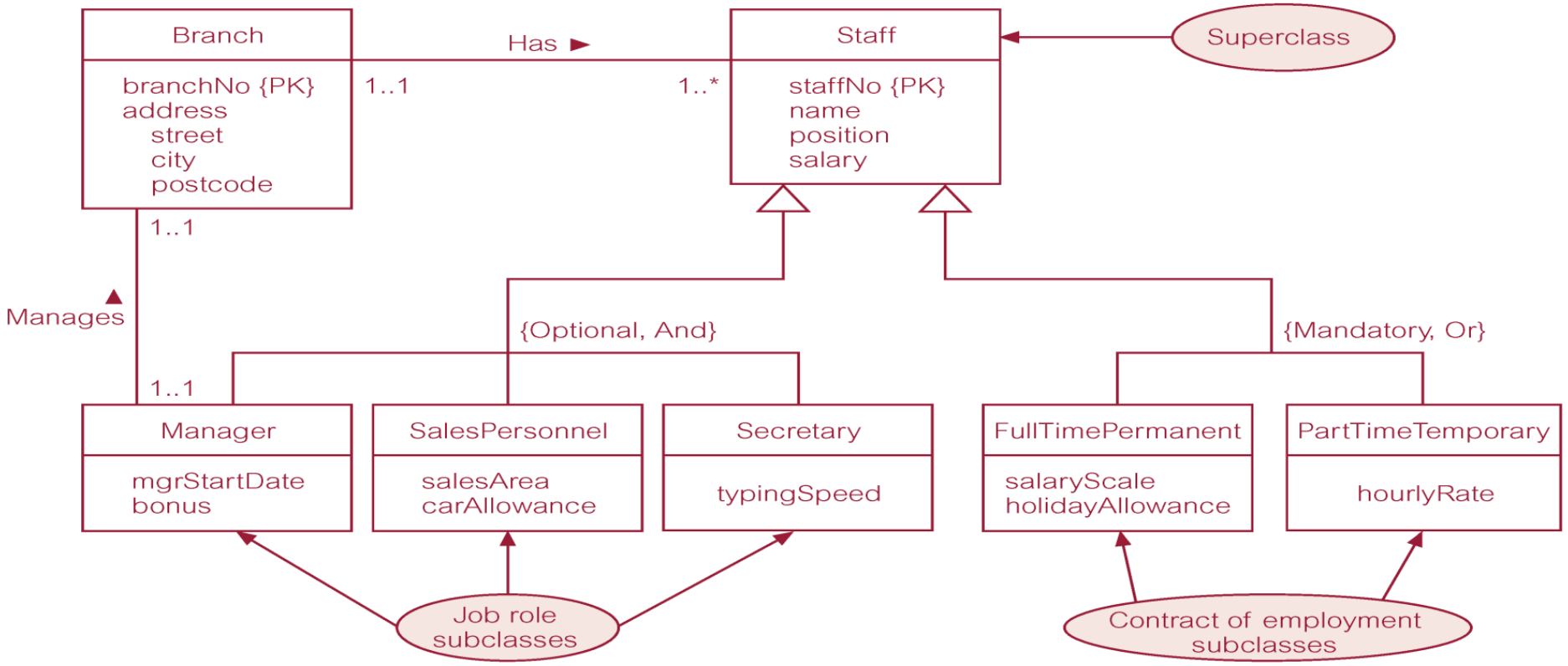
- **Participation constraint**
  - Determines whether every member in superclass must participate as a member of a subclass.
  - May be **mandatory** or **optional**.

- **Disjoint constraint**
  - Describes relationship between members of the subclasses and indicates whether member of a superclass can be a member of one, or more than one, subclass.
  - May be disjoint (can be member of only **one** of the subclasses - **OR**) or non-disjoint/overlapping (Multiple subclasses- **AND**).

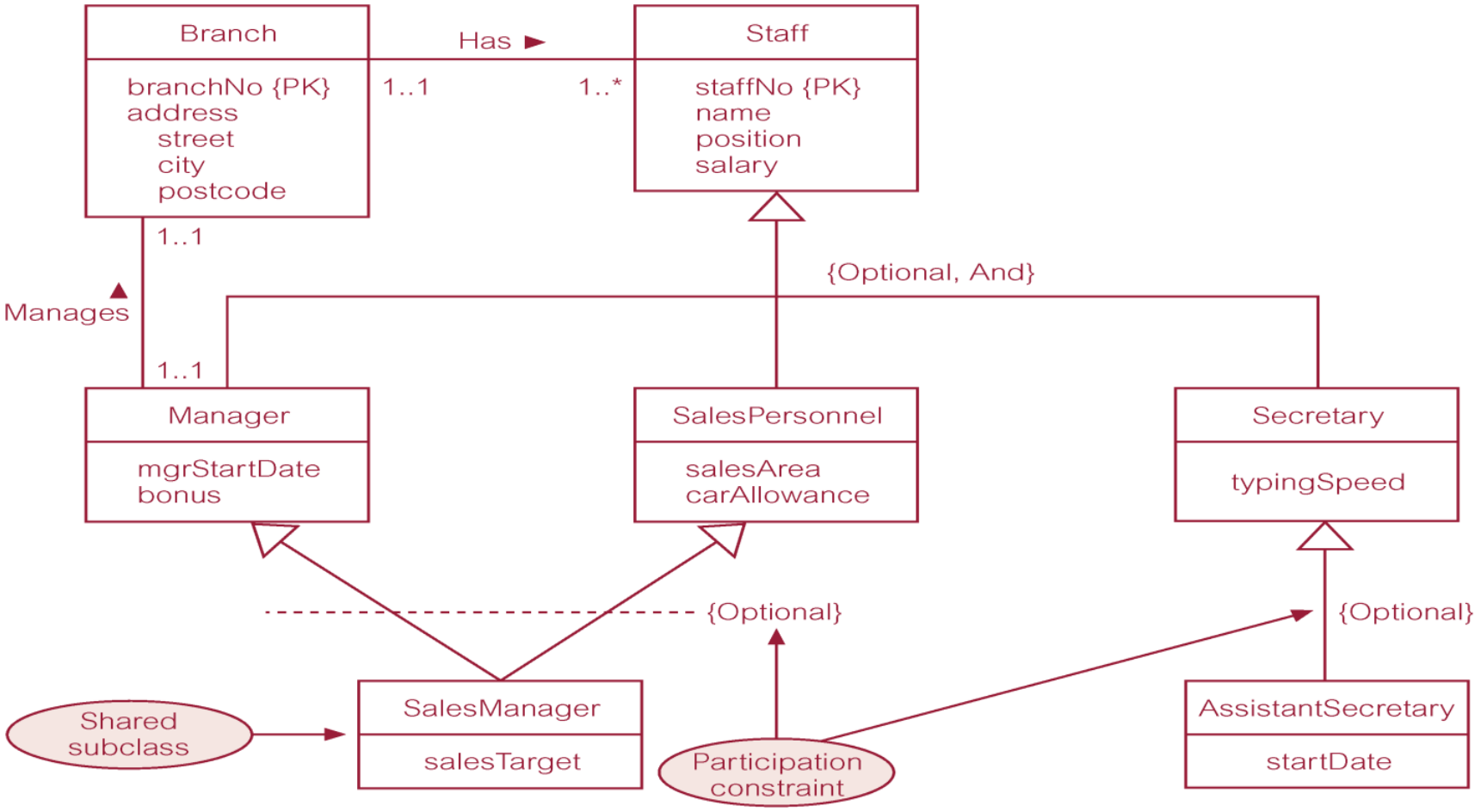
# Specialization/generalization of Staff entity into subclasses representing job roles



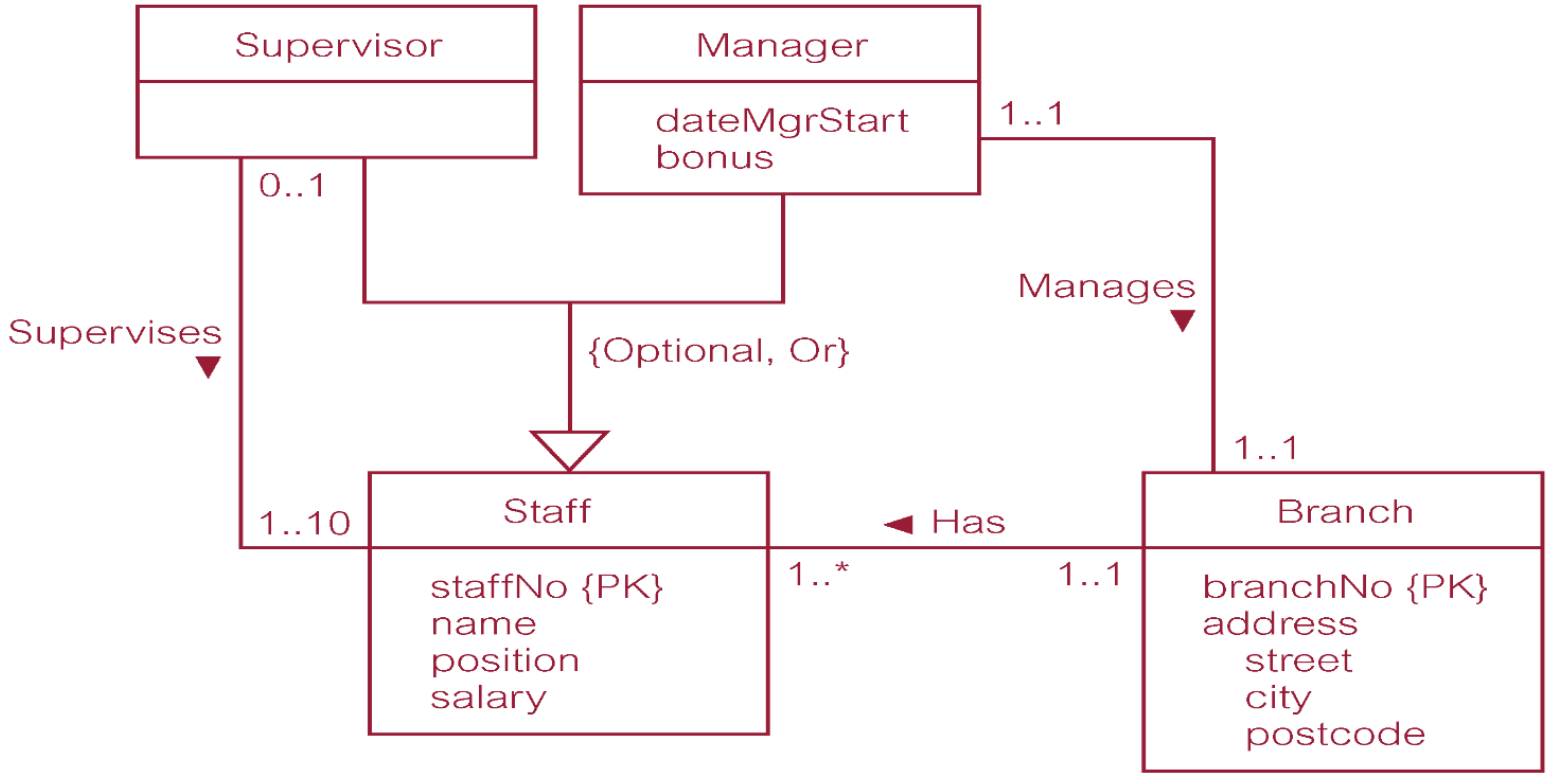
# Specialization/generalization of Staff entity into job roles and contracts of employment



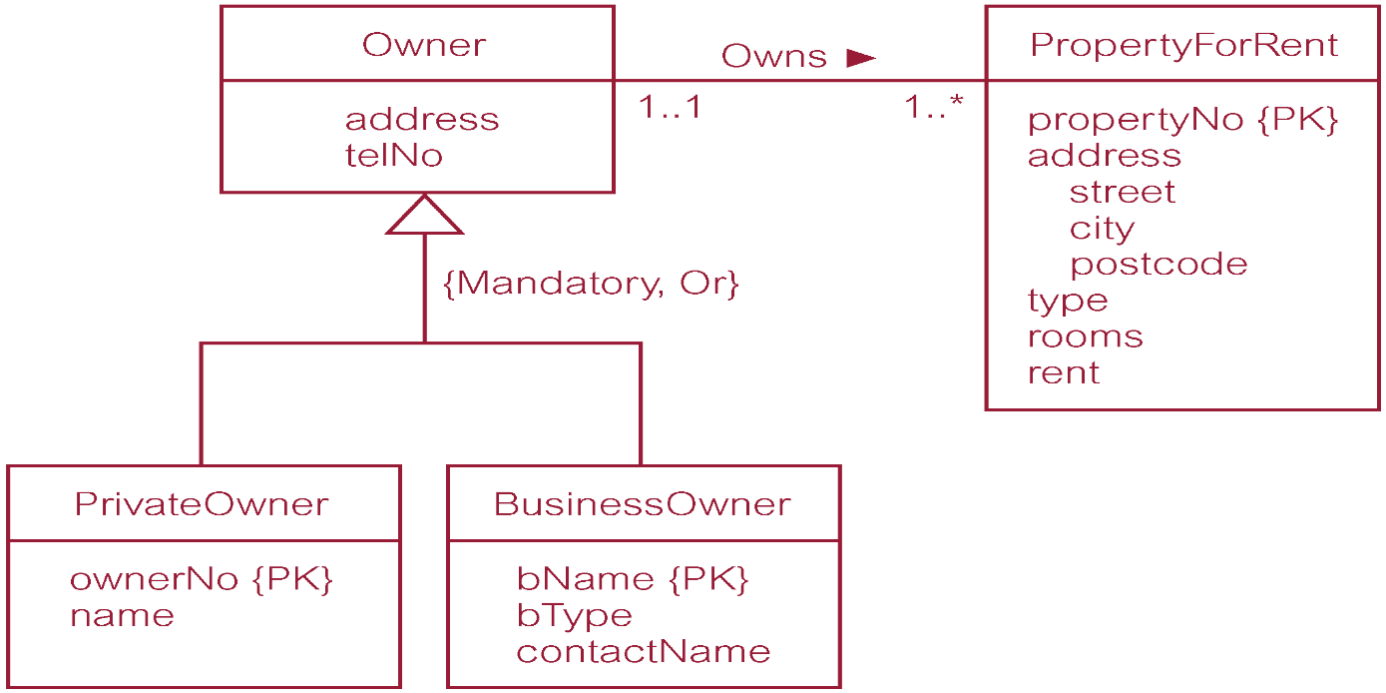
# EER diagram with shared subclass and subclass with its own subclass



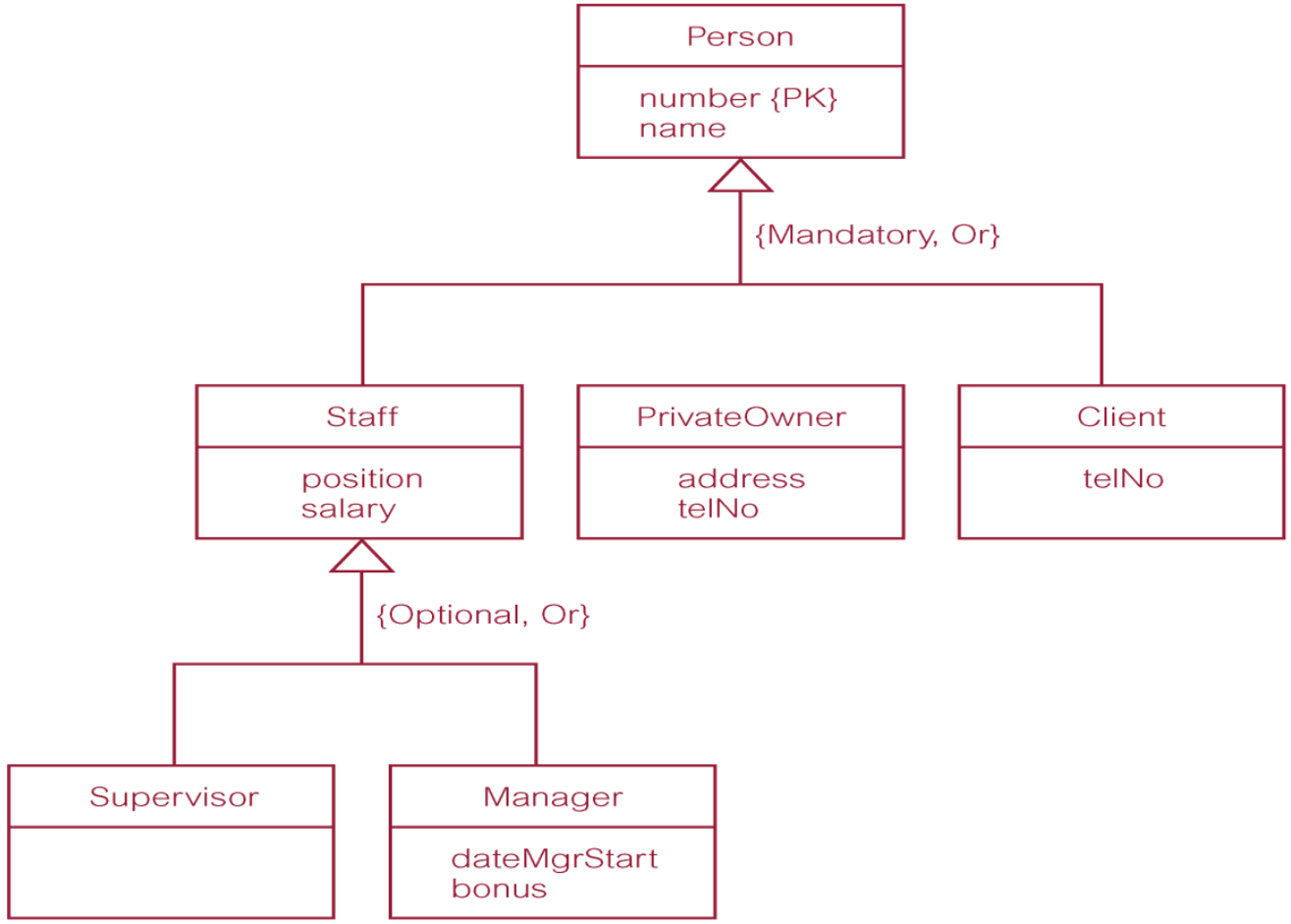
# DreamHome worked example - Staff Superclass with Supervisor and Manager subclasses



# DreamHome worked example - Owner Superclass with PrivateOwner and BusinessOwner subclasses

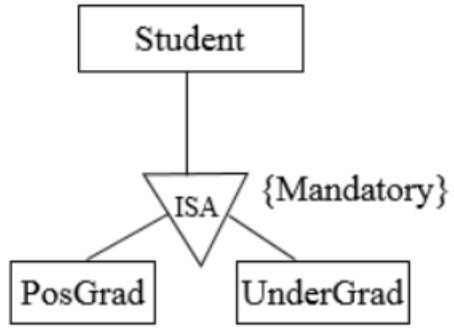


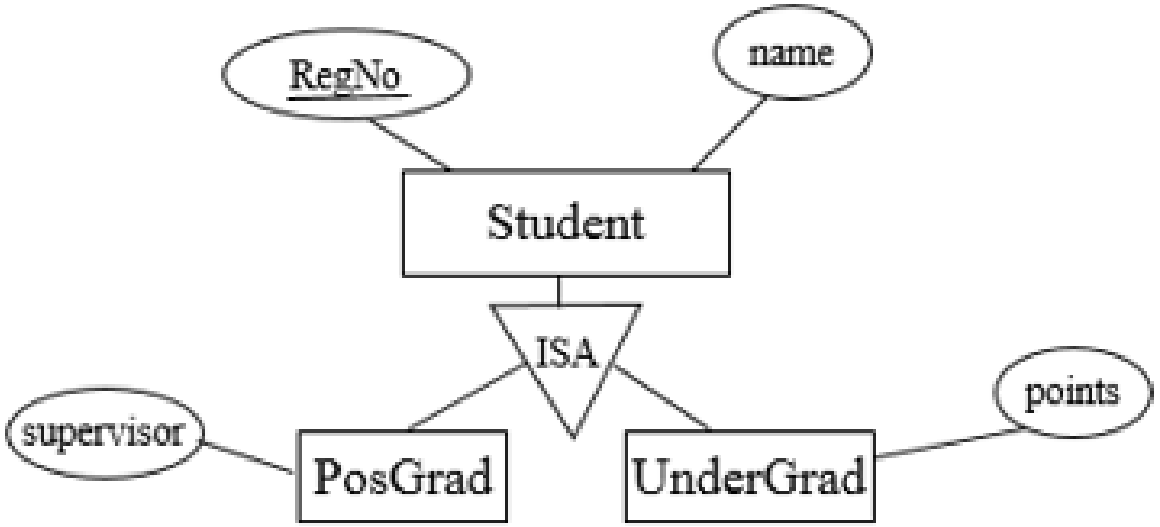
# DreamHome worked example - Person superclass with Staff, PrivateOwner, and Client subclasses



# Mapping to relational tables

- Student is postgraduate or undergraduate.
- A student in the university has a registration number and a name.
- Only *postgraduate* students have supervisors
- *Undergraduates* accumulates points through their coursework.



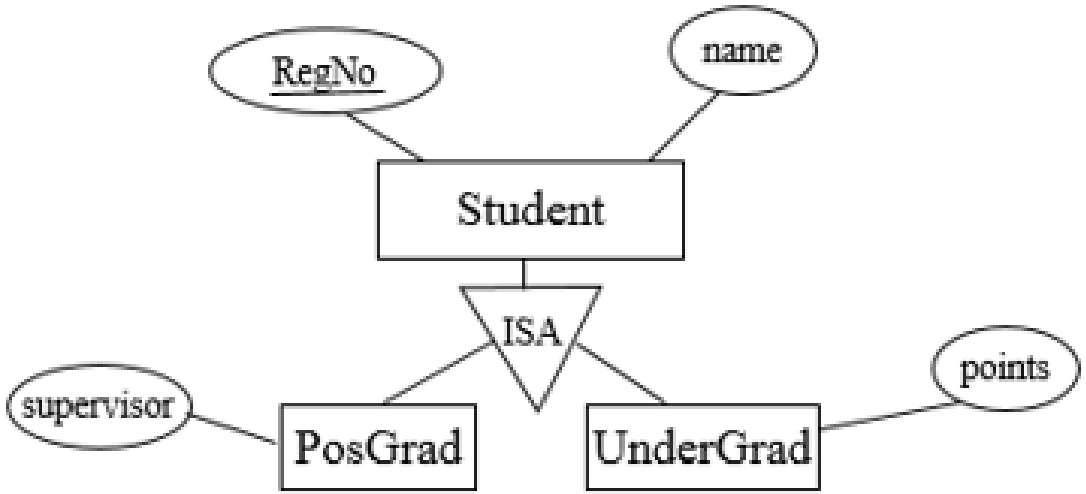
# Method 1

All the entities in the relationship are mapped to individual tables.

**Student** (Regno (PK), name)

**PosGrad** (Regno (PK,FK), supervisor)

**UnderGrad** (Regno (PK,FK), points)



# Method 2

Only subclasses are mapped to tables. The attributes in the superclass are duplicated in all subclasses.

**PosGrad** (Regno, name, supervisor)

**UnderGrad** (Regno, name, points)

This method is most preferred when inheritance is
*disjoint (OR) and mandatory*,
e.g. every student is either PosGrad or UnderGrad and nobody is both.

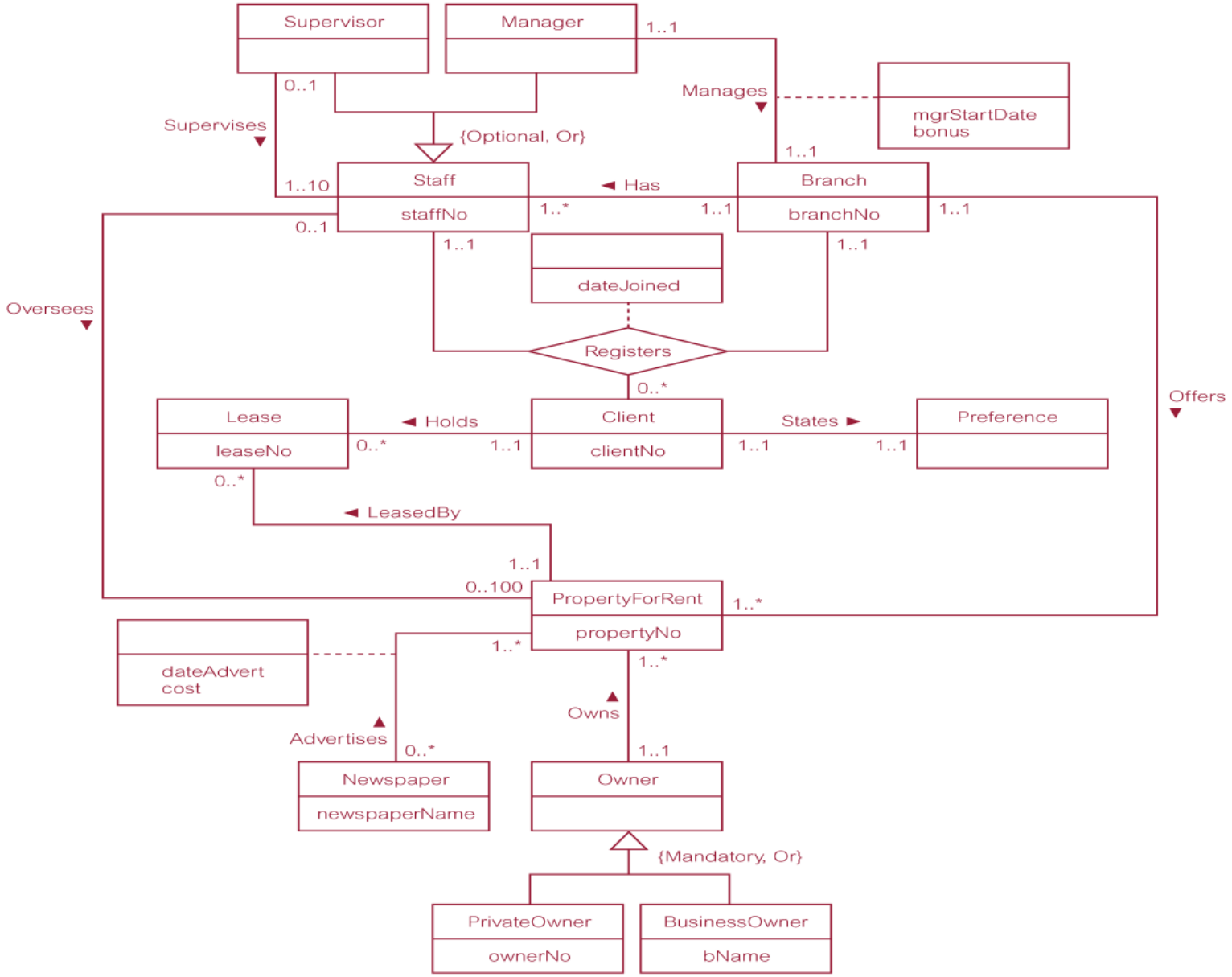
# Method 3

Only the superclass is mapped to a table. The attributes in the subclasses are taken to the superclass.

**Student** (Regno, name, supervisor, points)

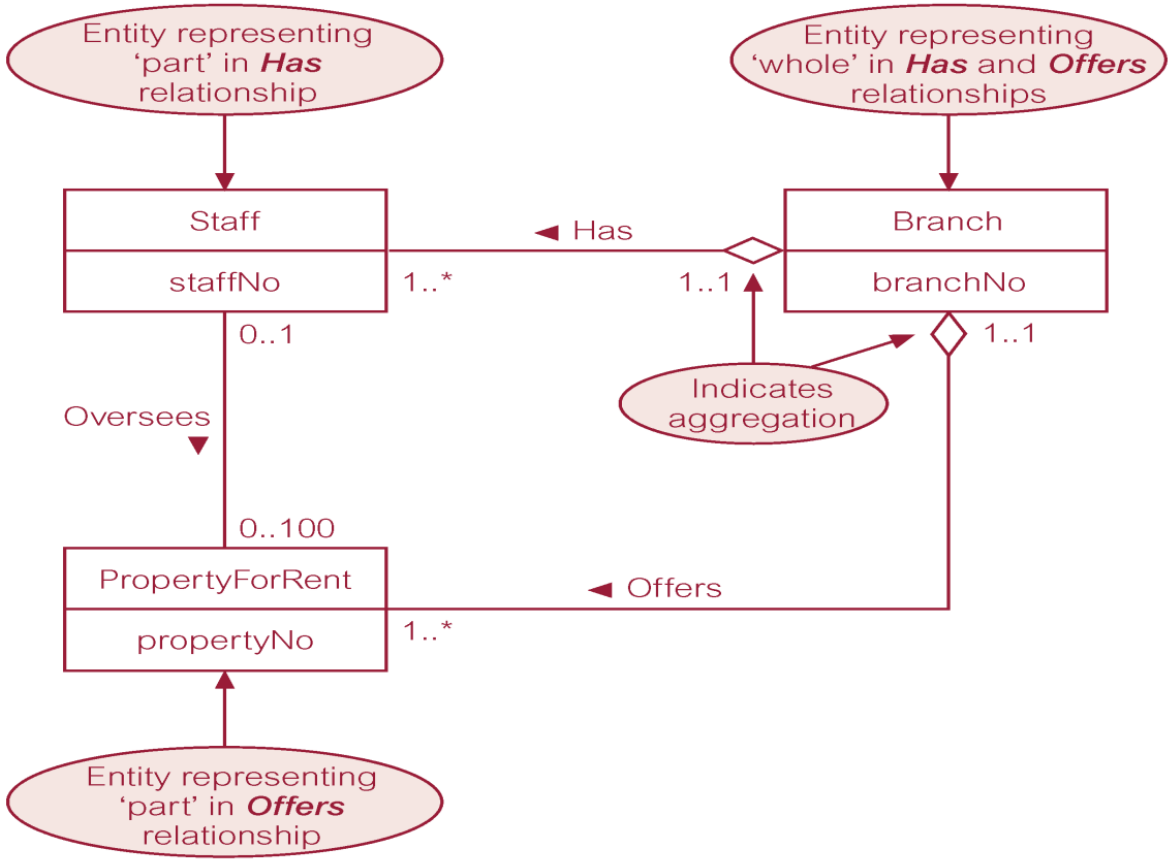
This method will introduce null values.

# EER diagram of Branch view of DreamHome with specialization/ generalization



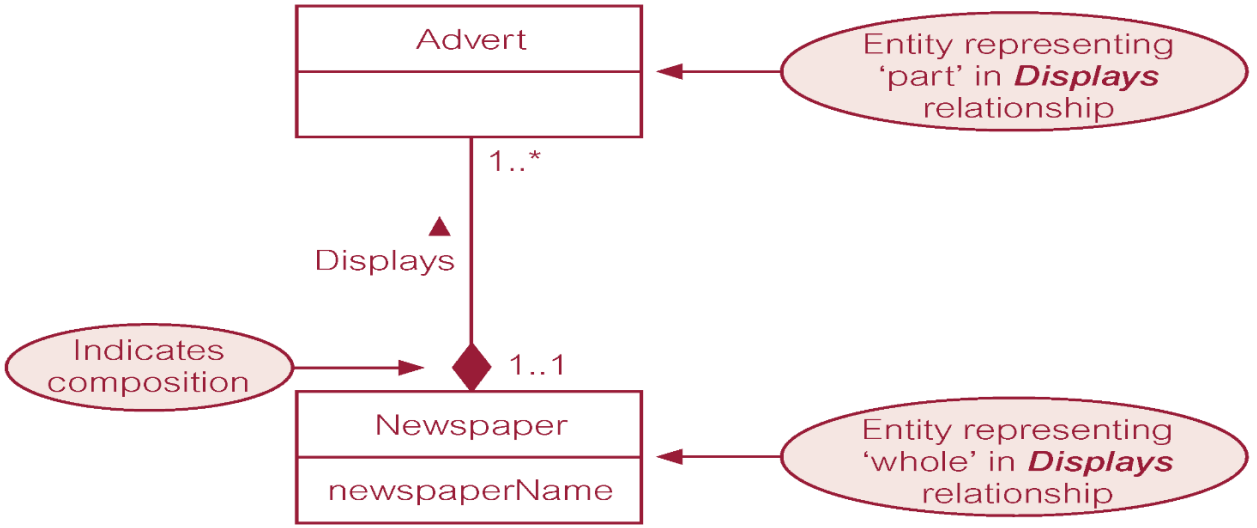
# Aggregation

- Represents a '**has-a**' or 'is-part-of' relationship between entity types, where one represents the 'whole' and the other 'the part'.



# Composition

- Specific form of aggregation that represents an association between entities, where there is a strong ownership and coincidental lifetime between the ‘whole’ and the ‘part




In Chen’s ER, we show this specialized relation of composition using **weak entities.**
This is the UML notation which shows composition but says nothing about the keys.