The Enhanced Entity Relationship Model(EER Model)

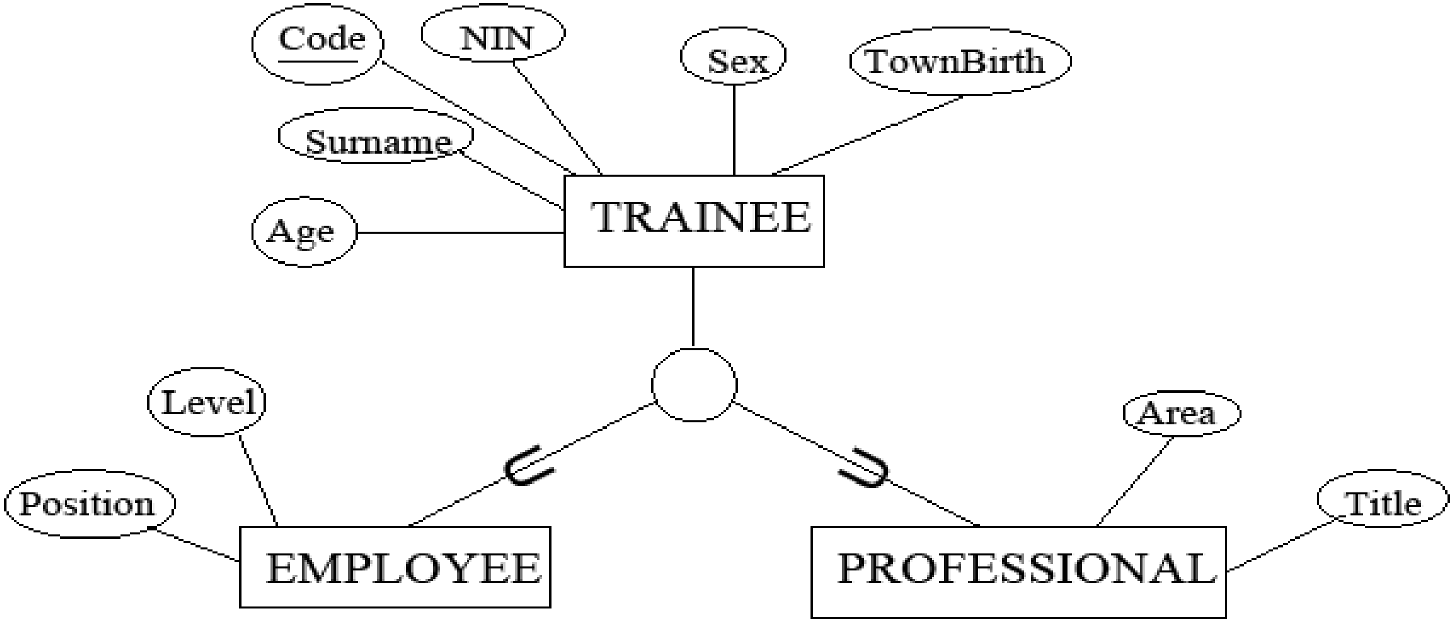
Contents

- Subclasses, Superclasses, and Inheritance
- Specialization and Generalization
- Constraints and Characteristics of Specialization and Generalization
- Hierarchies
- Modeling of UNION Types Using Categories

Why EER Modeling is required?

- Created to design more accurate database Schemas
- Reflect the data properties and constraints more precisely
- More complex requirements than traditional applications

EER Diagrams



Why EER reguired?

- Certain attributes apply to some but not to all members of the entity type
- Some relationship types are specific to only certain members of the entity type

Subclasses, superclasses and inheritance

- Terms for relationship between a superclass and any one of its subclasses
  - **Superclass/subclass**
  - **Supertype/subtype**
  - **Class/subclass** relationship
  - **Type inheritance**
- Subclass entity inherits all attributes and relationships of superclass

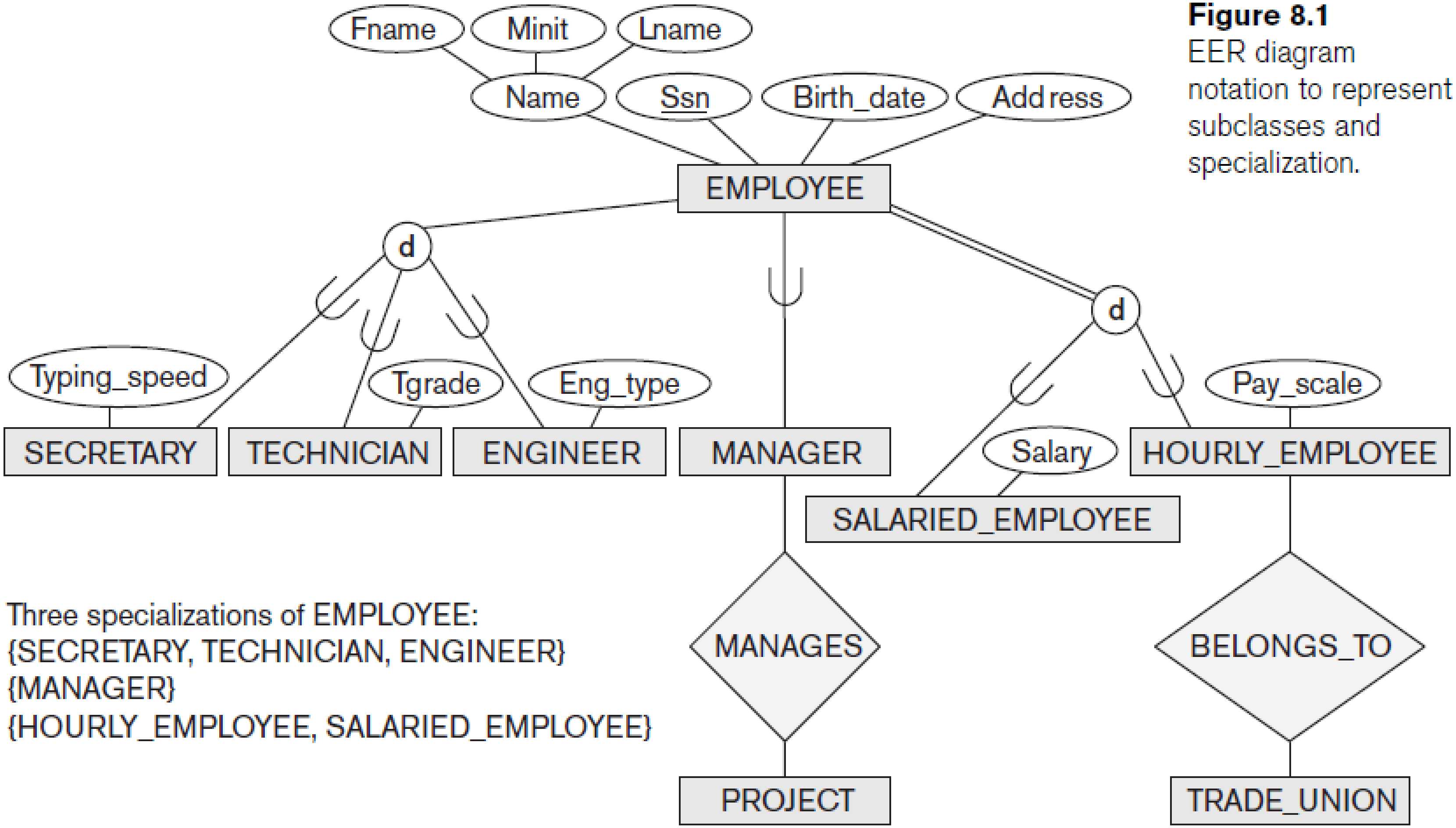


Figure 8.1
EER diagram notation to represent subclasses and specialization.

Three specializations of EMPLOYEE:
{SECRETARY, TECHNICIAN, ENGINEER}
{MANAGER}
{HOURLY_EMPLOYEE, SALARIED_EMPLOYEE}

Generalization

- Reverse process of abstraction
- **Generalize** into a single **superclass**
- Original entity types are special subclasses **Generalization**
- Process of defining a generalized entity type from the given entity types

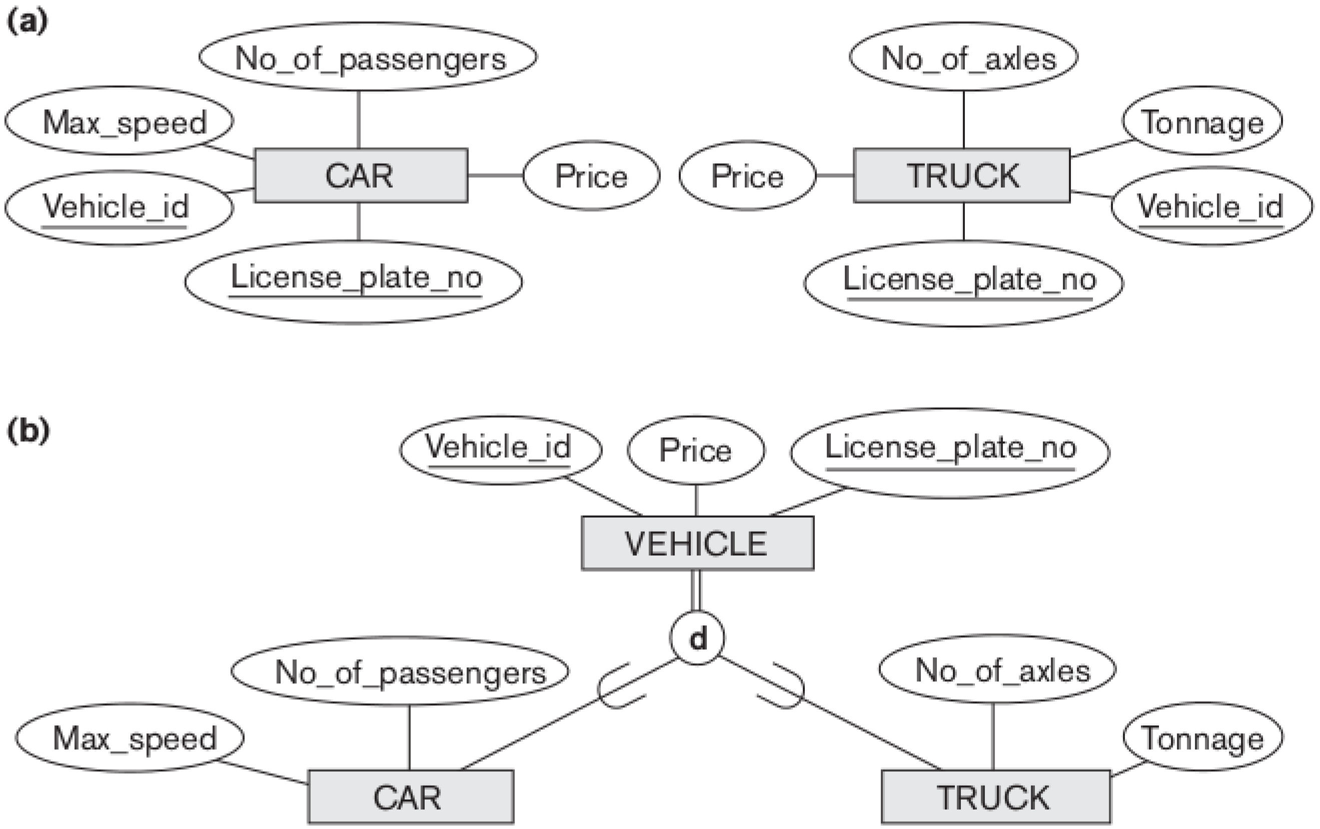


Figure 8.3
Generalization. (a) Two entity types, CAR and TRUCK. (b) Generalizing CAR and TRUCK into the superclass VEHICLE.

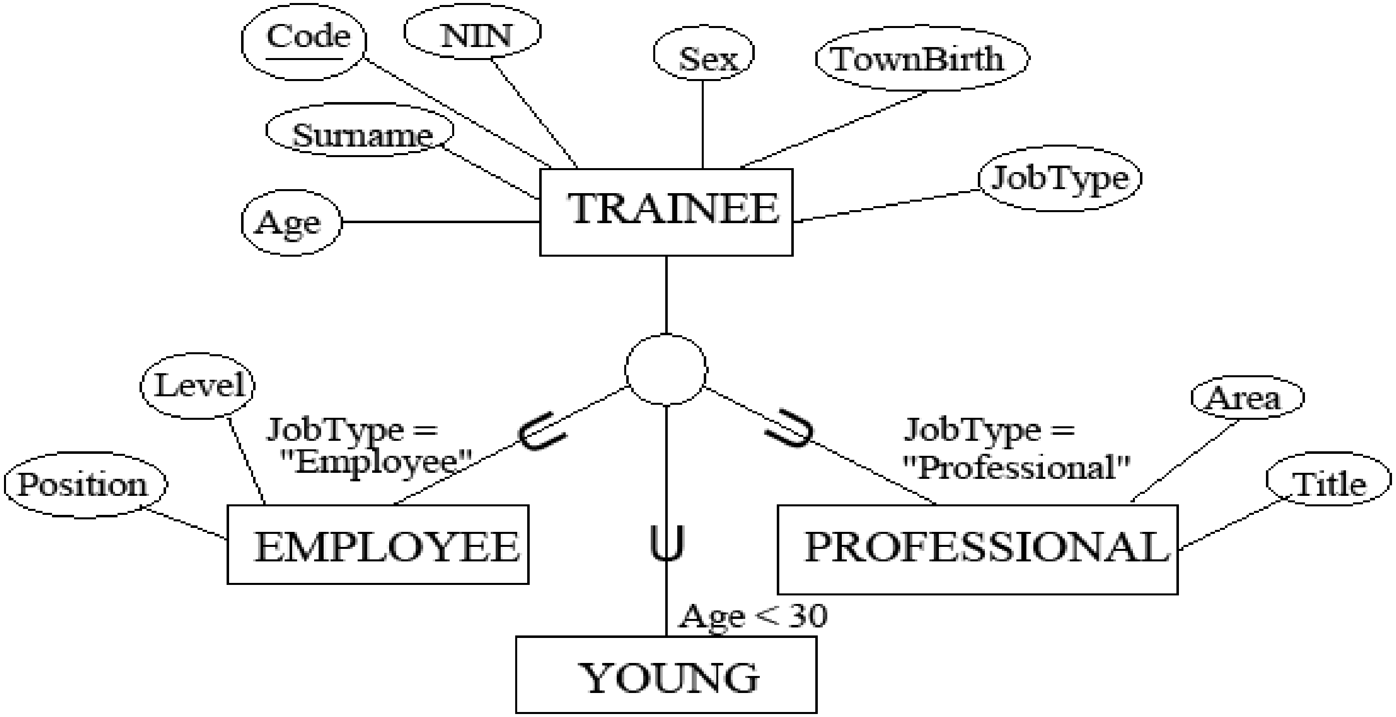
EER Diagrams

- subclasses that define a specialisation are attached by lines to a circle, which is connected with the superclass
- subset symbol indicates direction of relationship superclass/subclass
- *specific attributes* (those that apply only to the entities of the subclass) are attached to that subclass
- relationship types that apply only to a subclass are called *specific relationship types* (e.g. WORKS FOR between EMPLOYEES and EMPLOYER)

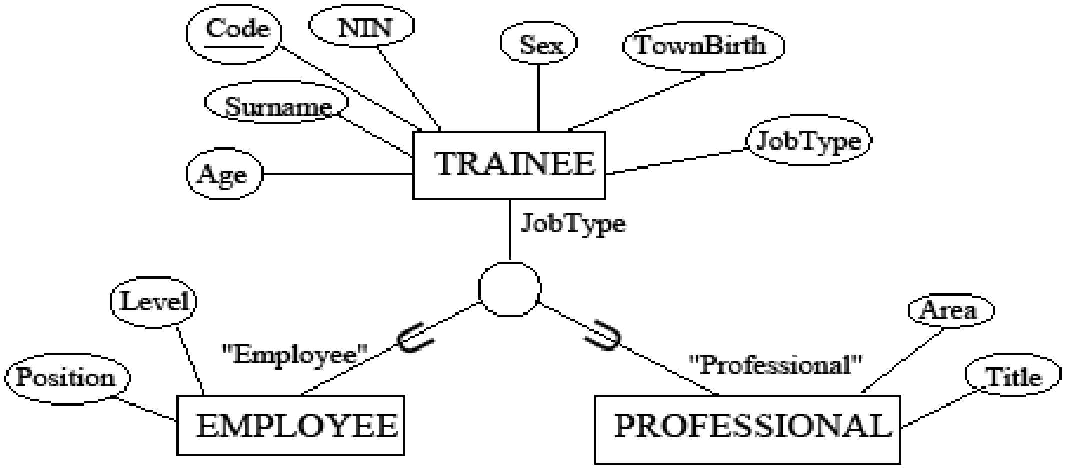
Constraints on Subclasses

- can sometimes determine exactly the entities that will become members of each subclasses
- e.g. for EMPLOYEE subclass we may specify the condition of membership to be predicate JobType="Employee"
  such condition act as constraints on the members of the EMPLOYEE subclass
- then we might have a class of "Young" trainees, whose age is less than 30 (no matter the job)
- we have therefore *predicate-defined subclasses*
- predicate condition is placed next to line joining subclass to circle

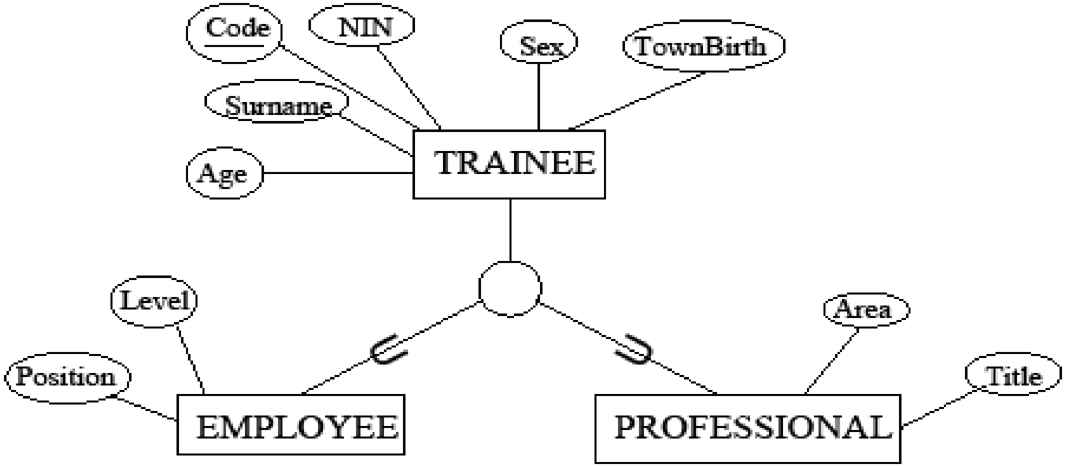
Constraints on Subclasses



- if all subclasses in a specialisation have the membership condition on the same attribute of the superclass we have *attribute-defined subclasses*
- attribute is the *defining attribute* of the specialisation
- shown by placing the name of the attribute on the arc from circle to superclass

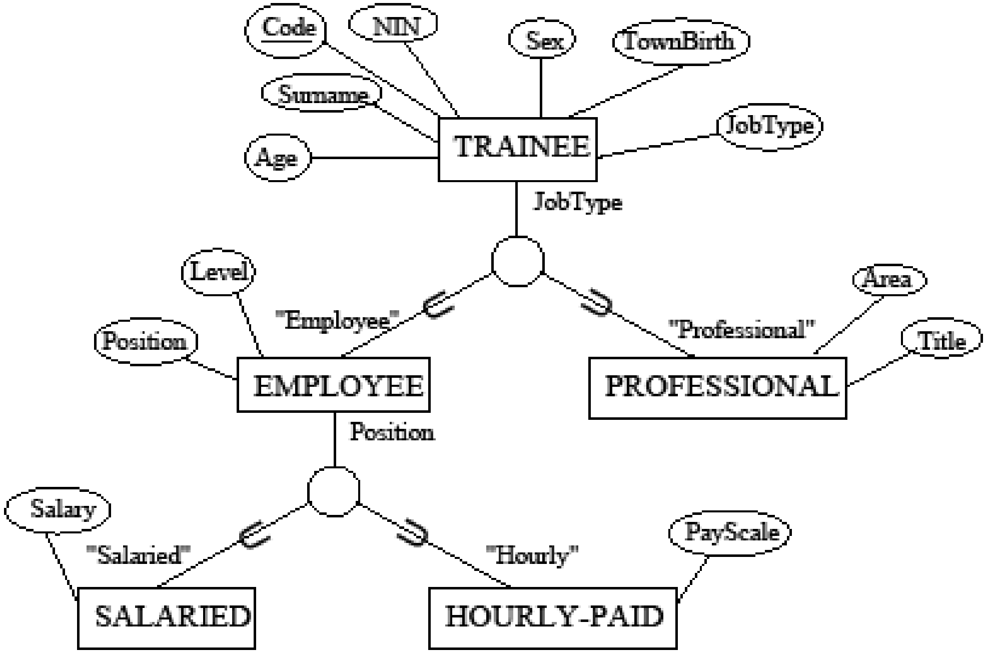


- where there is no condition we have a *user-defined subclass*
  - membership is determined by database users when adding an entity to the subclass
  - e.g. the original TRAINEE definition without JobType attribute



- we can have several specialisations of the same entity type using different distinguishing characteristics

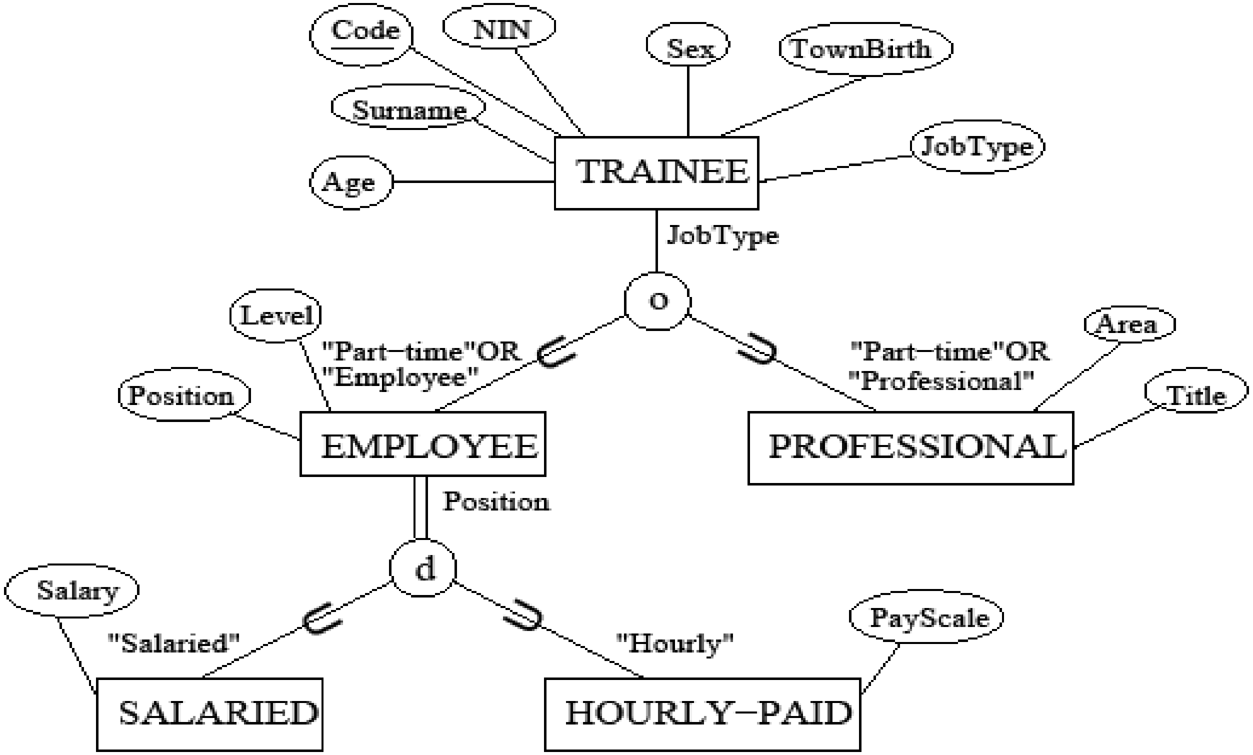
  e.g. EMPLOYEE can be specialised on the basis of Position into SALARIED-EMPLOYEE, HOURLY-PAID-EMPLOYEE



Note that SALARIED and HOURLY-PAID are both EMPLOYEE and TRAINEE

Disjointness Constraint

- specifies that subclasses of a specialisation are disjoint
  - an entity can be a member of at most one of the subclasses of the specialisation
  - attribute defined specialisation implies disjoint subclasses if the defining attribute is single-valued
- subclasses that are not disjoint may *overlap*
  - some entity may be a member of more than one subclass of the specialisation

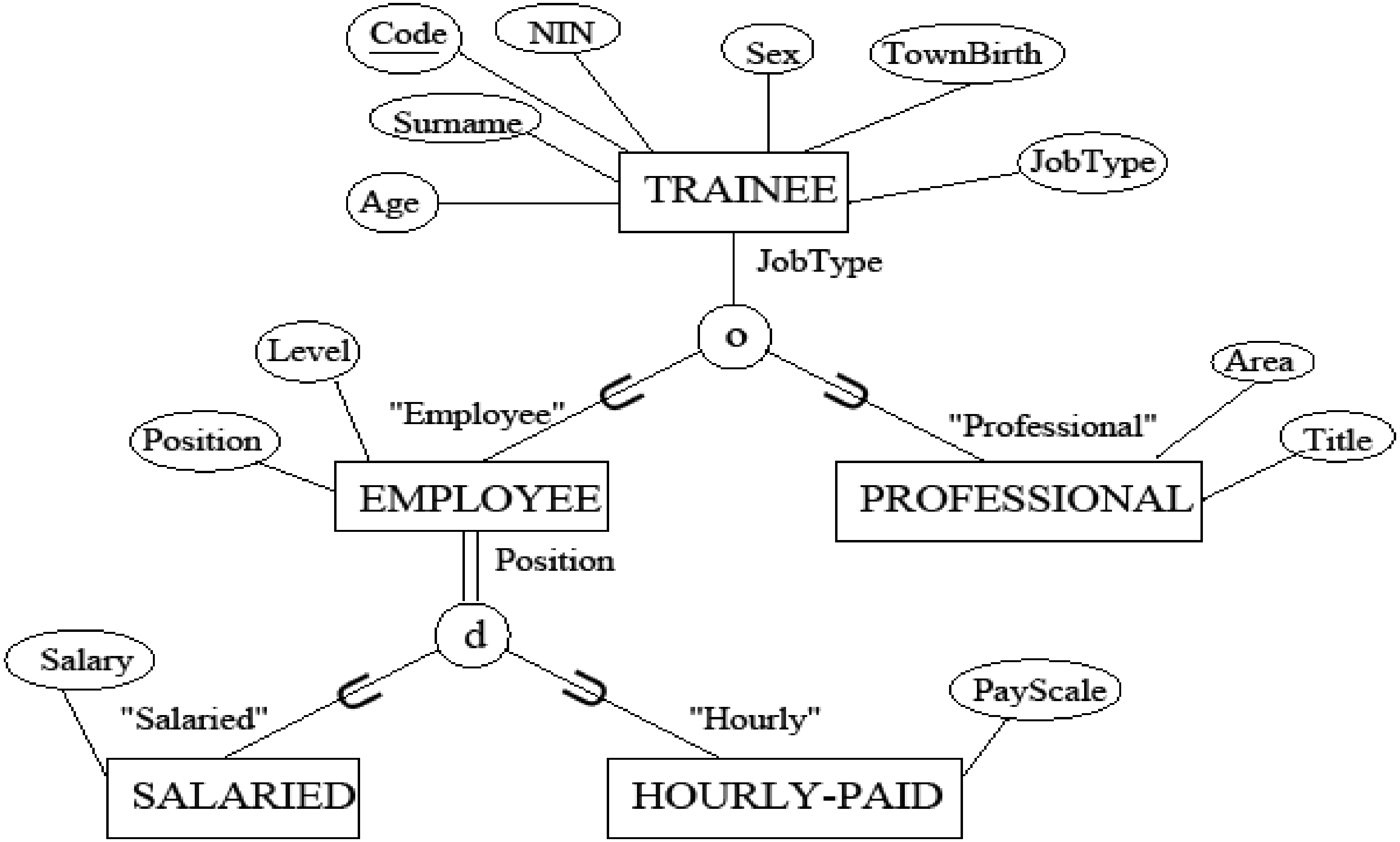


- professional trainees may have also a part time employment,
- but an employment can either be salaried or hourly paid

Completeness Constraints

1. *Total Specialisation*
    - every entity in a superclass must be a member of some subclass in some specialisation
    - e.g. every EMPLOYEE must be either HOURLY-PAID or SALARIED
2. *Partial Specialisation*
    - allows an entity not to belong to any of the subclasses
    - e.g. can have TRAINEES who are neither EMPLOYERS nor PROFESSIONALS

Completeness Constraints



Insertion and Deletion Rules

1. deleting an entity from a superclass implies that it is automatically deleted from all of the subclasses it belongs to
2. inserting an entity in a superclass implies that the entity is inserted in all predicate-defined subclasses for which the entity satisfies the defining predicate
3. inserting an entity in a superclass of total specialisation implies that the entity is inserted in at least one of the subclasses of the specialisation
4. inserting an entity in a superclass of disjoint, total specialisation implies that the entity is inserted in one and only one of the subclasses of the specialisation

Specialization and Generalization Hierarchies and Lattices

- **Specialization hierarchy**
  - Every subclass participates as a subclass in only one class/subclass relationship
  - Results in a **tree structure** or **strict hierarchy**
- **Specialization lattice**
  - Subclass can be a subclass in more than one class/subclass relationship

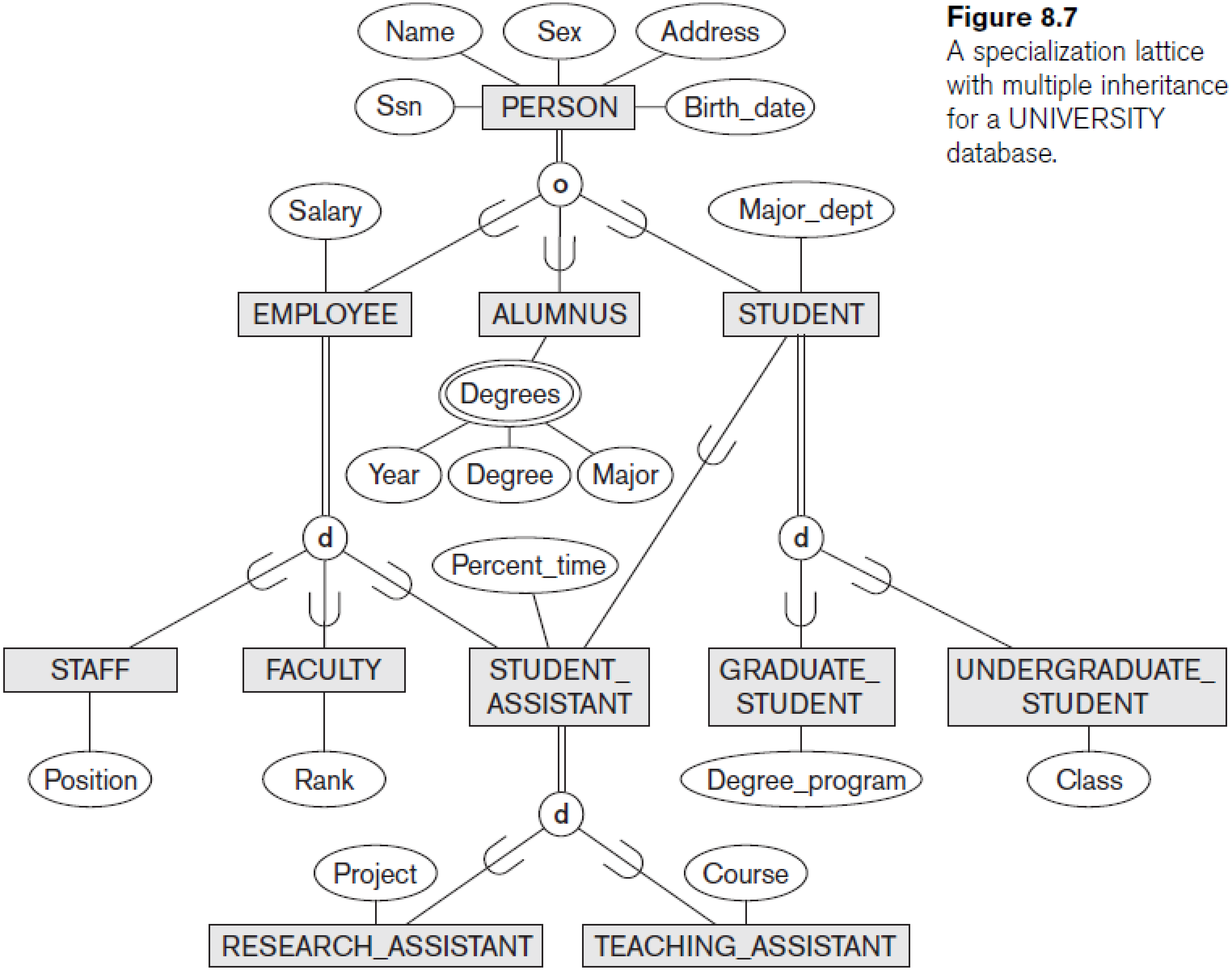


Figure 8.7
A specialization lattice with multiple inheritance for a UNIVERSITY database.

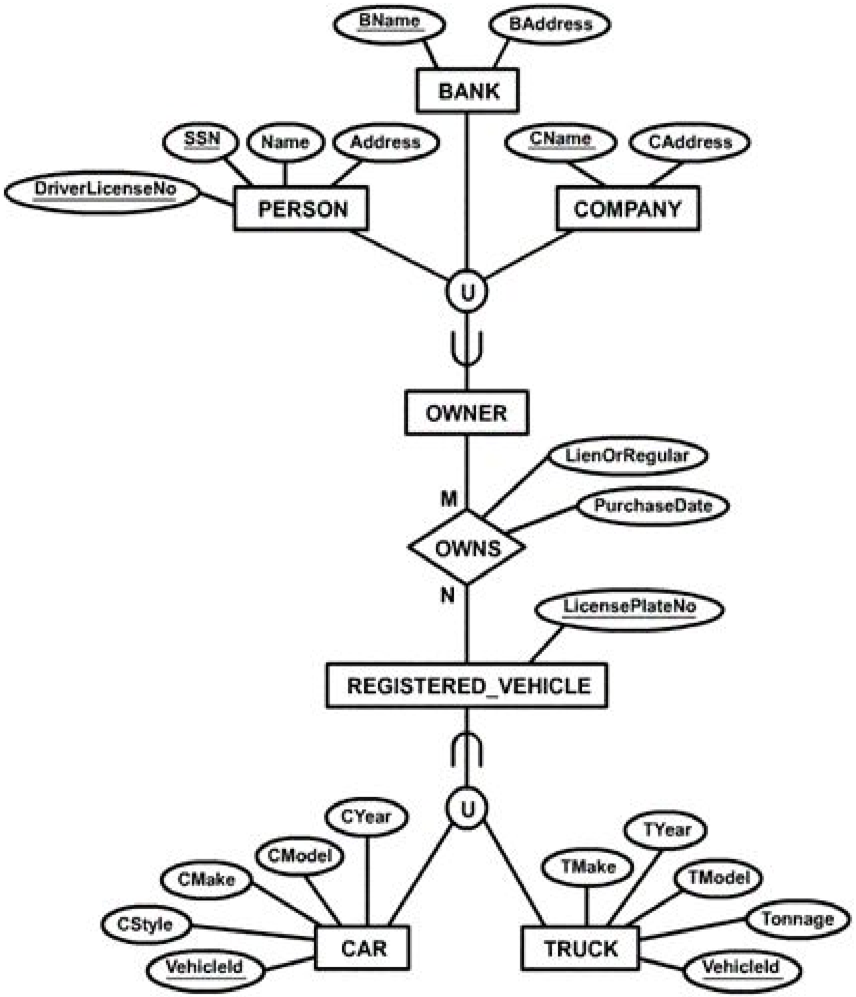
Specialization and Generalization Hierarchies and Lattices (cont'd.)

- **Multiple inheritance**
  - Subclass with more than one superclass
  - If attribute (or relationship) originating in the same superclass inherited more than once via different paths in lattice
  - Included only once in shared subclass
- **Single inheritance**
  - Some models and languages limited to single inheritance

Modeling of UNION Types Using Categories

- **Union type** or a **category**
  - Represents a single superclass/subclass relationship with more than one superclass
  - Subclass represents a collection of objects that is a subset of the UNION of distinct entity types
- Attribute inheritance works more selectively
- Category can be **total** or **partial**
- Some modeling methodologies do not have union types

Example of categories (UNION TYPES)



Discussion of n-ary relationships (n > 2)

- In general, 3 binary relationships can represent different information than a single ternary relationship
- If needed, the binary and n-ary relationships can all be included in the schema design
- In some cases, a ternary relationship can be represented as a weak entity if the data model allows a weak entity type to have multiple identifying relationships

Example of a ternary relationship

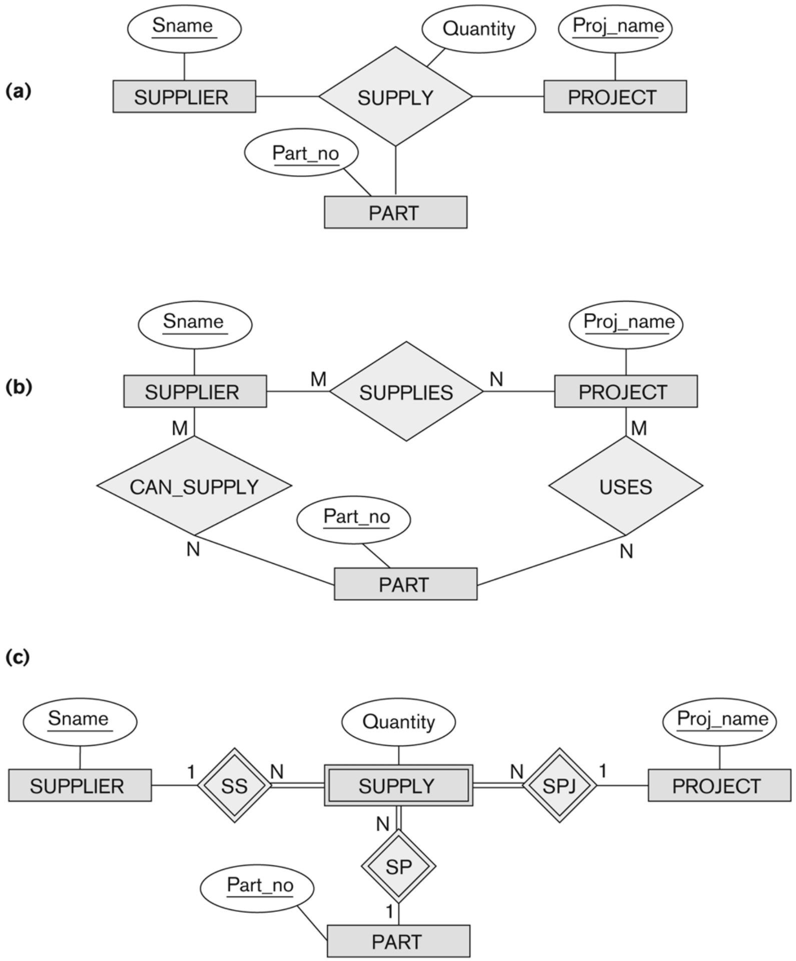
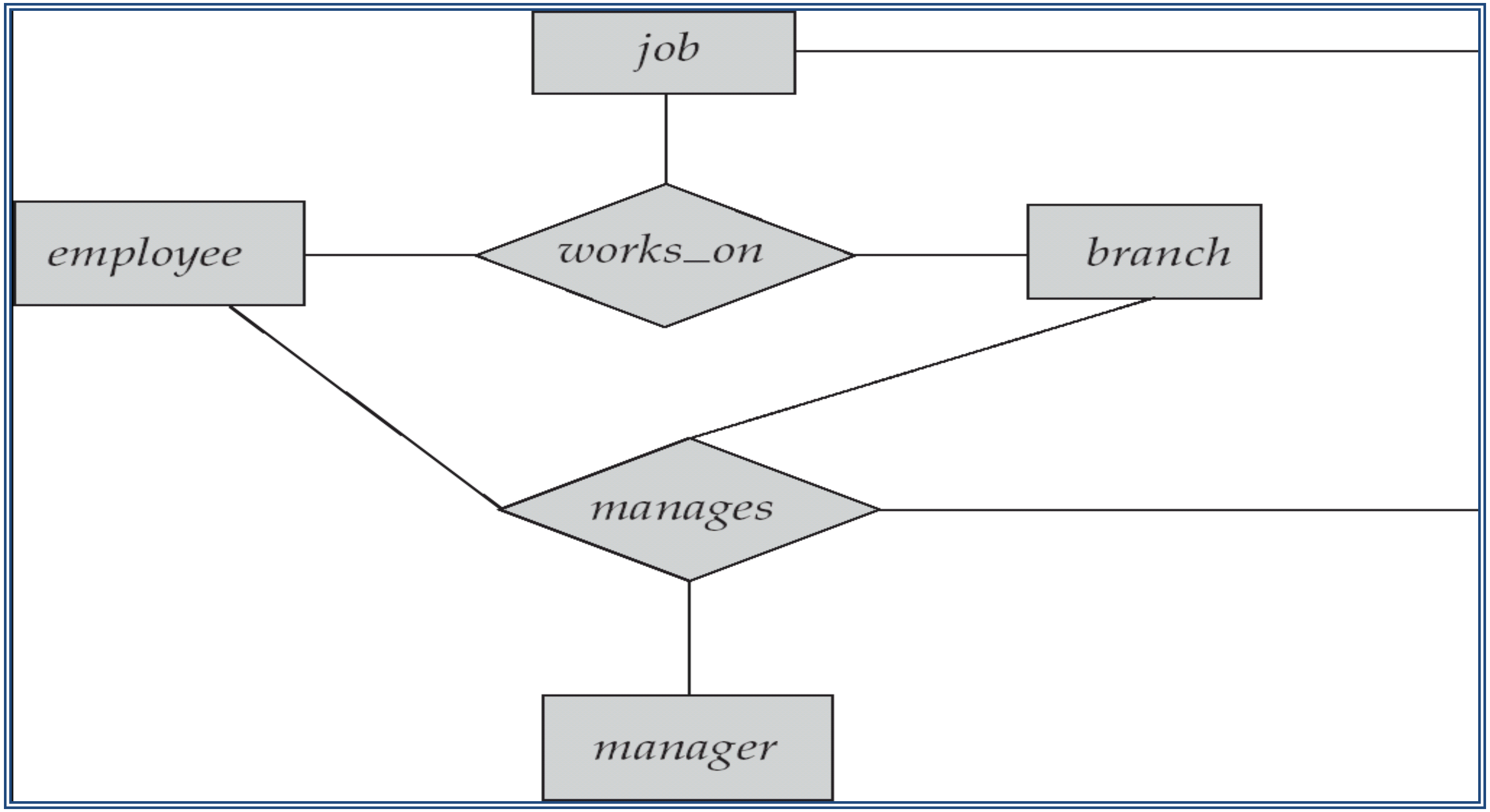


Figure 3.17
Ternary relationship types. (a) The SUPPLY relationship. (b) Three binary relationships not equivalent to SUPPLY. (c) SUPPLY represented as a weak entity type.

Aggregation

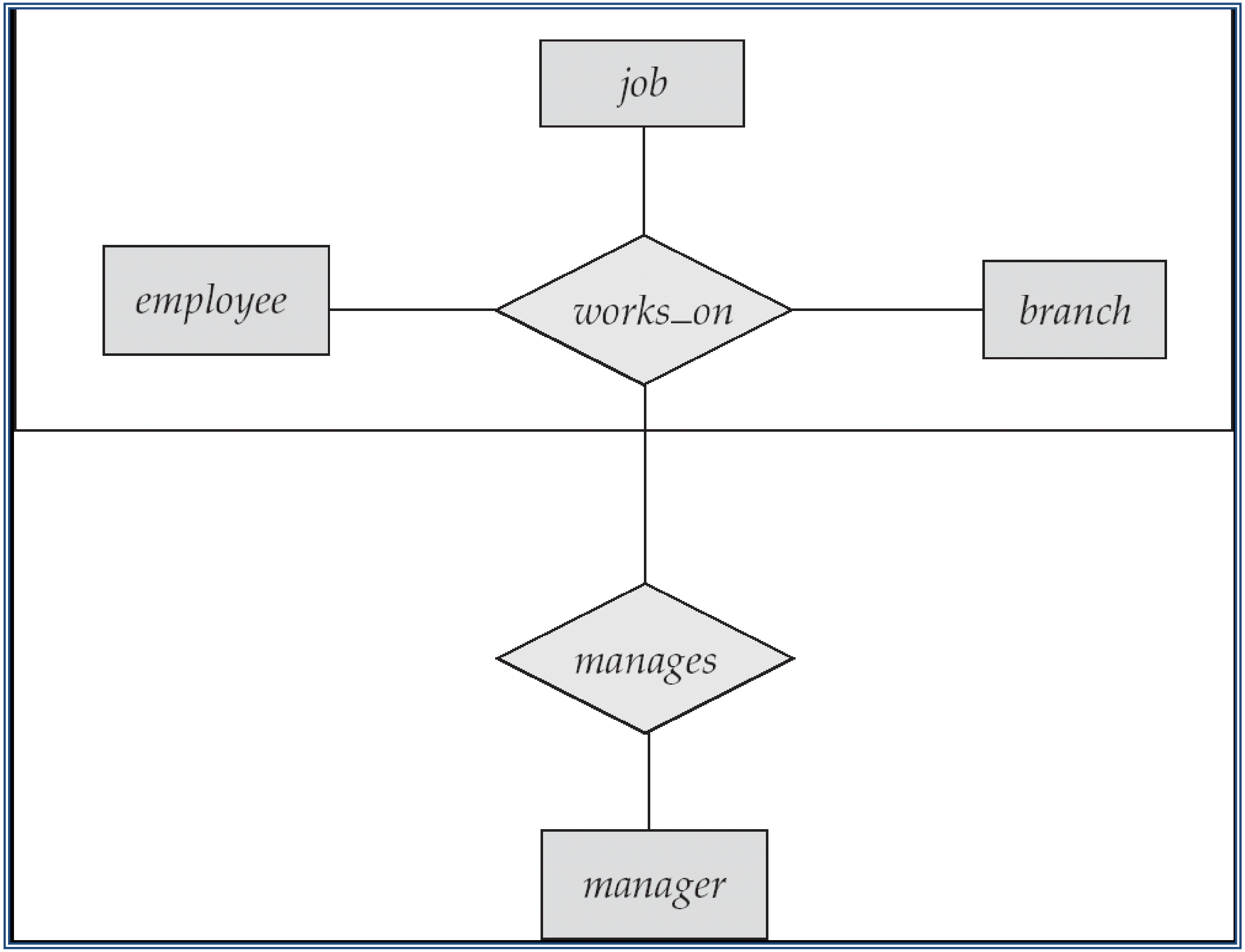
- Consider the ternary relationship *works_on*
- Suppose we want to record managers for tasks performed by an employee at a branch



Aggregation (Cont.)

- Relationship sets *works_on* and *manages* represent overlapping information
  - Every *manages* relationship corresponds to a *works_on* relationship
  - However, some *works_on* relationships may not correspond to any *manages* relationships
    - So we can’t discard the *works_on* relationship
- Eliminate this redundancy via *aggregation*
  - Treat relationship as an abstract entity
  - Allows relationships between relationships
  - Abstraction of relationship into new entity
- Without introducing redundancy, the following diagram represents:
  - An employee works on a particular job at a particular branch
  - An employee, branch, job combination may have an associated manager

E-R Diagram With Aggregation



keys

- Primary key:- used to uniquely identify a tuple
- Referential integrity:- this is specified between two relations and is used to maintain the consistency among tuples in the two relations
- Foreign key:- is used to specify referential integrity between the two relation schemas R1 and R2.

Evaluating Data Model Quality

- List of quality criteria
  - Completeness:- complete with respect to client requirements
  - Correctness:-should be checked by client and database experts
  - Consistency:-should be checked with the users of the system
  - Minimality:-model should be compact and should not include redundancy

Evaluating Data Model Quality contd…

- Readability:- Subjective
- Self-explanation:-the names should be so chosen that it explains the hidden meaning
- Extensibility:- Model should be able to accommodate business changes
- Performance:- It should be possible to make some changes without impacting the model’s logical structure