CONCEPTUAL MODELLING

Chapters 3,4: the Entity/Relationship (ER) and the Enhanced ER (EER) models

Outline of Chapters 3 and 4

1 Conceptual Modeling

2 Database Design Process

3 Example Database Application (COMPANY)

4 ER Model Concepts

5 Notation

6 Relationships of Higher Degree

7 Extended ER model concepts

1. Conceptual Modeling

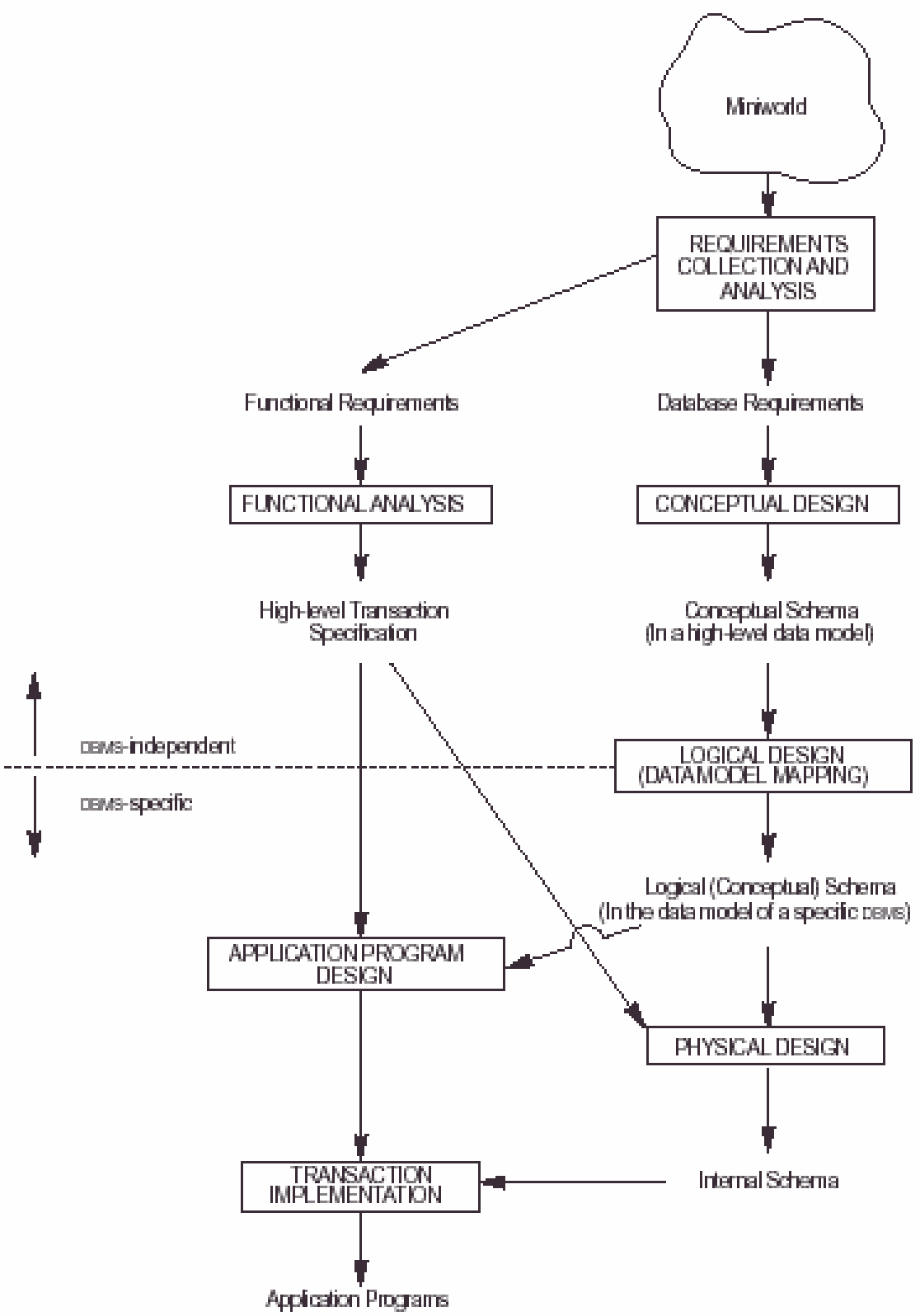
A high-level **conceptual model** provides concepts for describing the database structure that are close to the way users perceive data.

A **conceptual schema** describes the structure of the database by hiding the details of physical storage structures.

Conceptual modeling is the process of creating a conceptual schema using a high-level conceptual model.

2. Database Design Process

The main phases of database design:



3. Example Database Application (COMPANY)

Requirements for the COMPANY Database:

- The company is organized into DEPARTMENTs. Each department has a unique name, a unique number, and an a particular employee who *manages* the department. We keep track of the start date of the department manager. A department may have several locations.

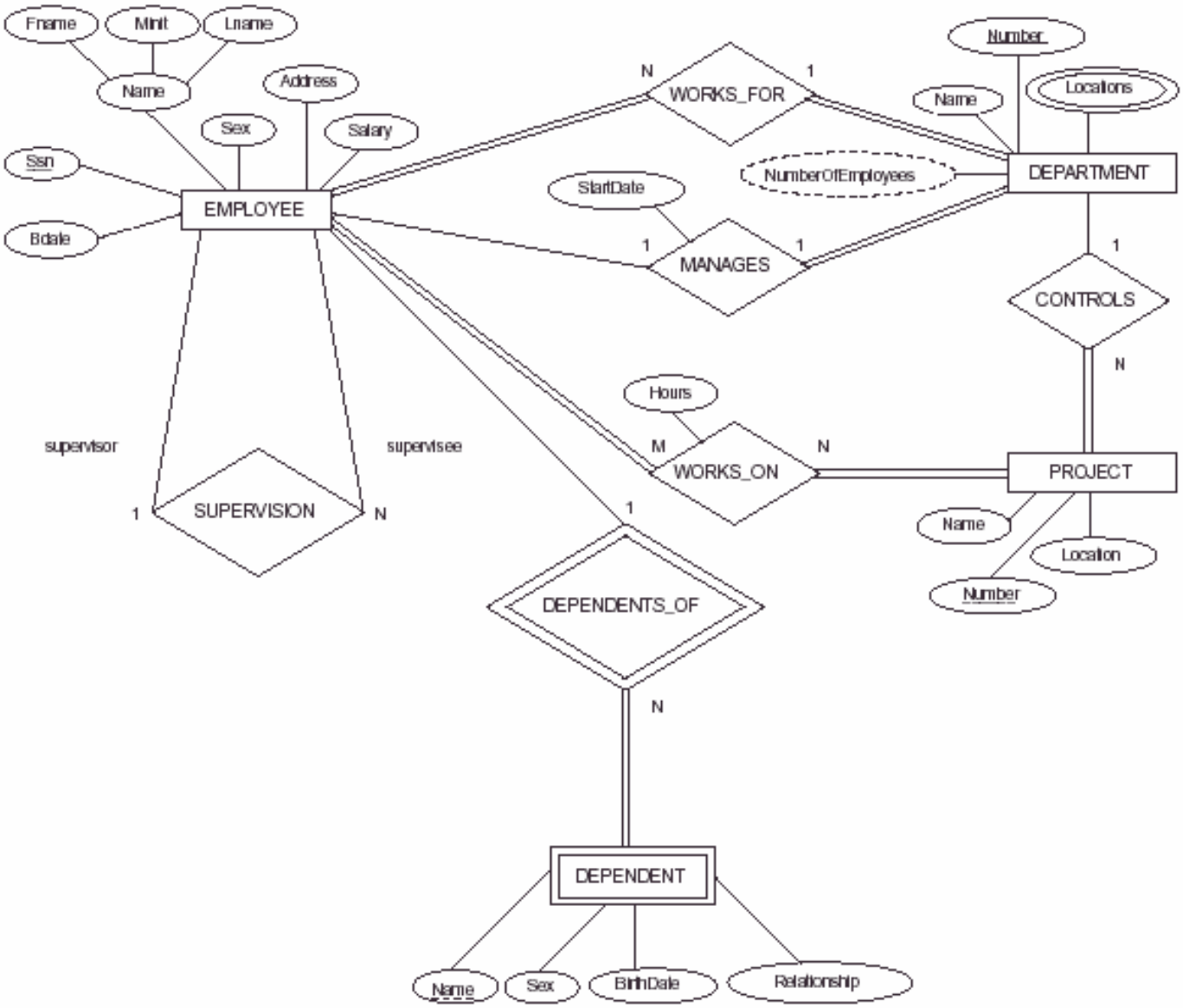
- A department *controls* a number of PROJECTs. Each project has a unique name, a unique number, and is located at a single location.

- We store each EMPLOYEE's name, social security number, address, salary, sex, and birth date. Each employee *works for* one department but may *work on* several projects. We keep track of the number of hours per week that an employee currently works on each project. We also keep track of the *direct supervisor* of each employee.

- Each employee may have a number of DEPENDENTs. For each dependent, we keep its first name, sex, birth date, and relationship to the employee.

3. Example Database Application – ER diagram

Figure 3.2 ER schema diagram for the company database.



4. ER model concepts

- The **Entity-Relationship (ER) model** is a popular high-level conceptual model.

- It employs the notions of **entities, attributes** and **relationships**.

- This model and its variations are frequently used for the conceptual design of database applications, and many database design tools employ its concepts

- It was introduced by P. Chen in

"The Entity-Relationship model – Toward a Unified View of Data",
TODS 1:1, March 1976.

4.1. Entities and Attributes (1)

- An **entity** is a specific object or *thing* in the mini-world with an independent existence.

  Example: the EMPLOYEE John Smith, the Research DEPARTMENT, the ProductX PROJECT.

- **Attributes** are *properties* used to describe an entity

  Example: an EMPLOYEE entity may have a Name, SSN, Address, Sex, BirthDate.

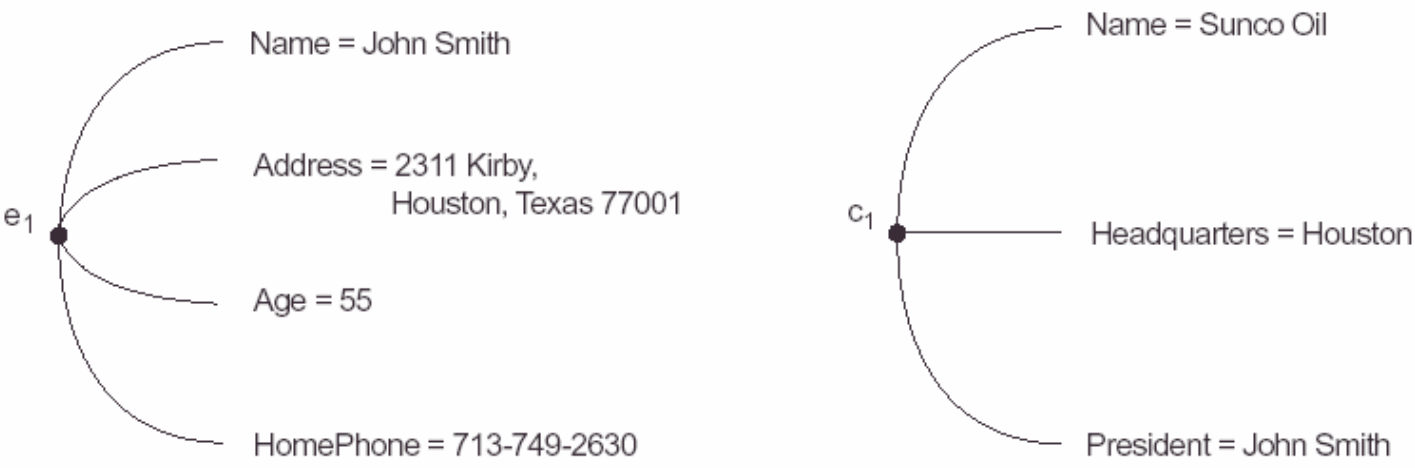
- A *specific entity* will have a **value** for each of its attributes

  Example: a specific EMPLOYEE entity may have Name='John Smith', SSN='123456789', Address='731 Fondren, Houston, TX', Sex='M', BirthDate='09-JAN-55'.

4.1. Entities and Attributes (2)

Example

Figure 3.3 Two entities, an employee e_1 and a company c_1, and their attribute values.



4.1. Entities and Attributes (3)

Types of attributes

Simple (Atomic) vs. Composite Attributes:

- **Simple attribute**: Each entity has a *single atomic value* for this attribute.
  These attributes are not divisible.

  Example: SSN or Sex.

- **Composite attribute**: This attribute may be composed of several components.

  Example: Address(Apt#, House#, Street, City, State, ZipCode, Country) or

  Name(FirstName, MiddleName, LastName).

  The *value of a composite attribute* is the concatenation of the values of its component simple attributes.

4.1. Entities and Attributes (4)

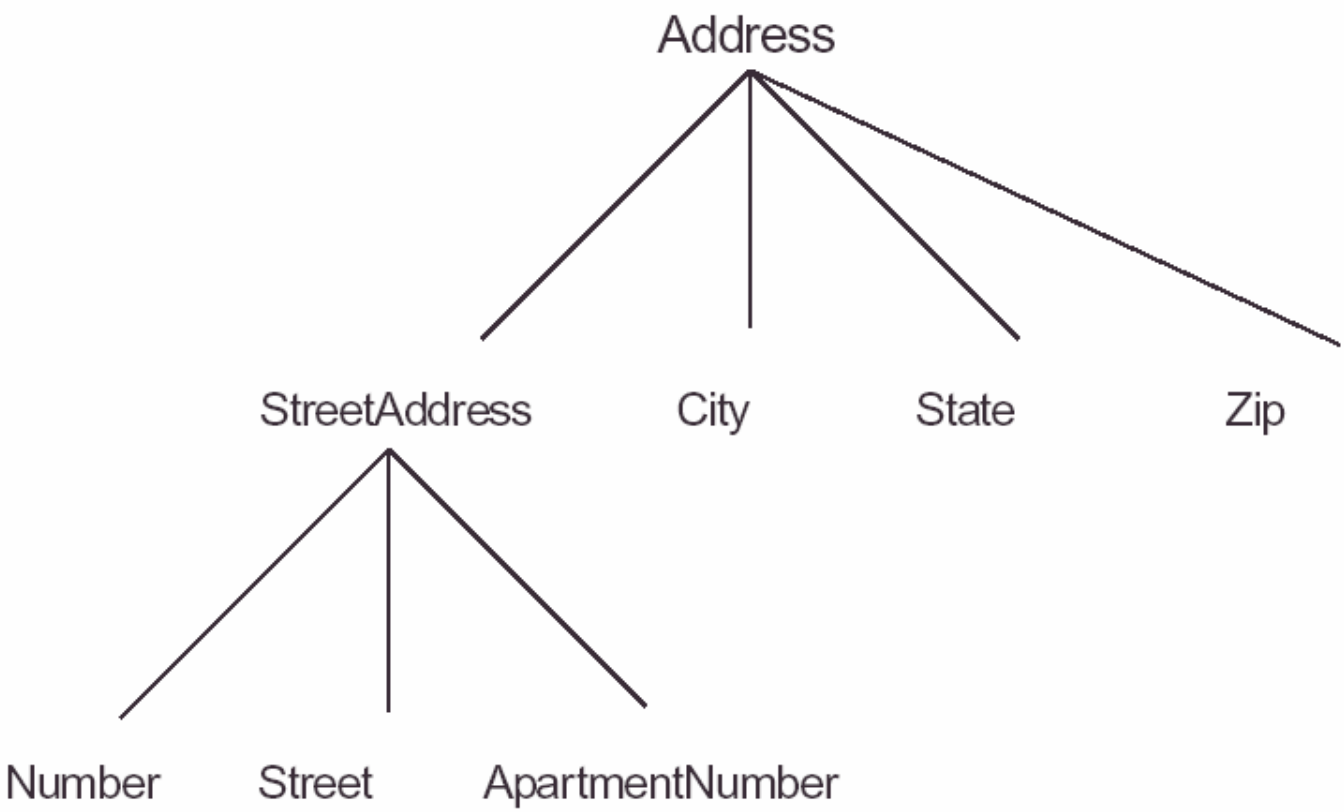
Types of attributes

Simple (Atomic) vs. Composite Attributes:

Composition may form a hierarchy where some components are themselves composite.

Example

Figure 3.4 A hierarchy of composite attributes; the StreetAddress component of an Address is further composed of Number, Street, and ApartmentNumber.



4.1. Entities and Attributes (5)

Types of attributes

Single-valued vs. Multivalued Attributes:

- **Single-valued attribute**: It has *a single value* for a particular entity.

  Example: Age.

- **Multi-valued**: An entity may have *multiple values* for that attribute.

  Example: Color of a CAR or PreviousDegrees of a STUDENT.

  Notation: {Color} or {PreviousDegrees}

4.1. Entities and Attributes (6)

Types of attributes

Nesting composite and multivalued attributes:

In general, *composite and multi-valued attributes may be nested arbitrarily* to any number of levels although this is rare.

Example: PreviousDegrees of a STUDENT can be composite multi-valued attribute denoted by

{PreviousDegrees(College, Year, Degree, Field)}.

Example:

{AddressPhone({**Phone**(AreaCode, PhoneNumber)}, **Address**(**StreetAddress**(Number, Street, AptNumber), City, State, Zip) **) }**

4.1. Entities and Attributes (7)

Types of attributes

Stored vs. Derived Attributes:

- **Derived attribute**: An attribute whose value can be derived from the value of another attribute (the latter is called **stored attribute**).

  Example: Age (Derived) and BirthDate (Stored) attributes.

Remark: The values of some attributes can be *derived from related entities*.

Example: The attribute NumberOfEmployees of a DEPARTMENT entity can be derived by counting the EMPLOYEE entities of this department.

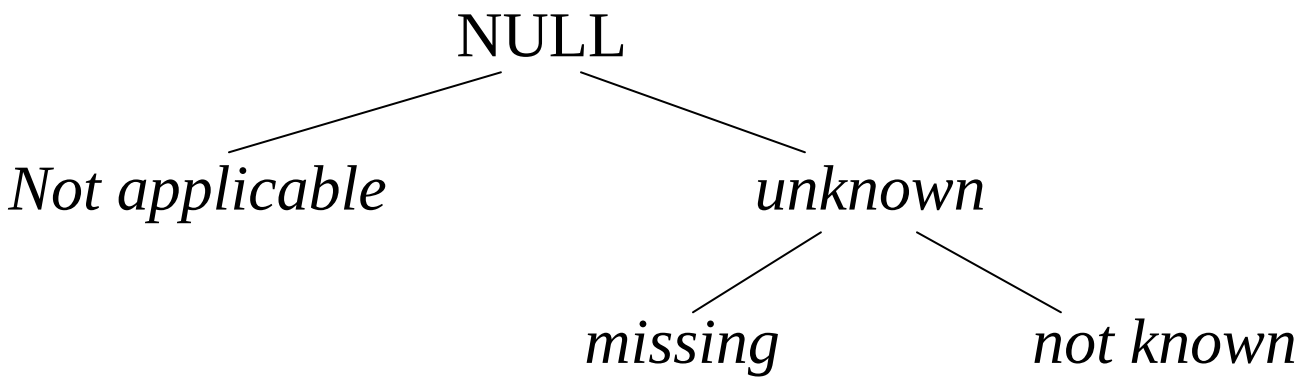
4.1. Entities and Attributes (8)

Null values

- In some cases a particular entity *may not have an applicable value* for an attribute. In such situations a special value **NULL** is used.

  Example: For the ApartmentNumber attribute of an address for the PreviousDegrees attribute of a student.

- *There are different meanings for null values:*



4.2. Entity Types (1)

Entity types and Entity sets

- An **entity type** defines a collection of entities that have the same attributes (that are of the same type).

  Example: The EMPLOYEE entity type or the PROJECT entity type.

- An **entity set** is the collection of all entities of a particular entity type in the database at any point in time.

  Remark: The same name is used for both the entity type and its entity set.

- An entity type describes the **schema** or **intension** of a set of entities that have the same structure.

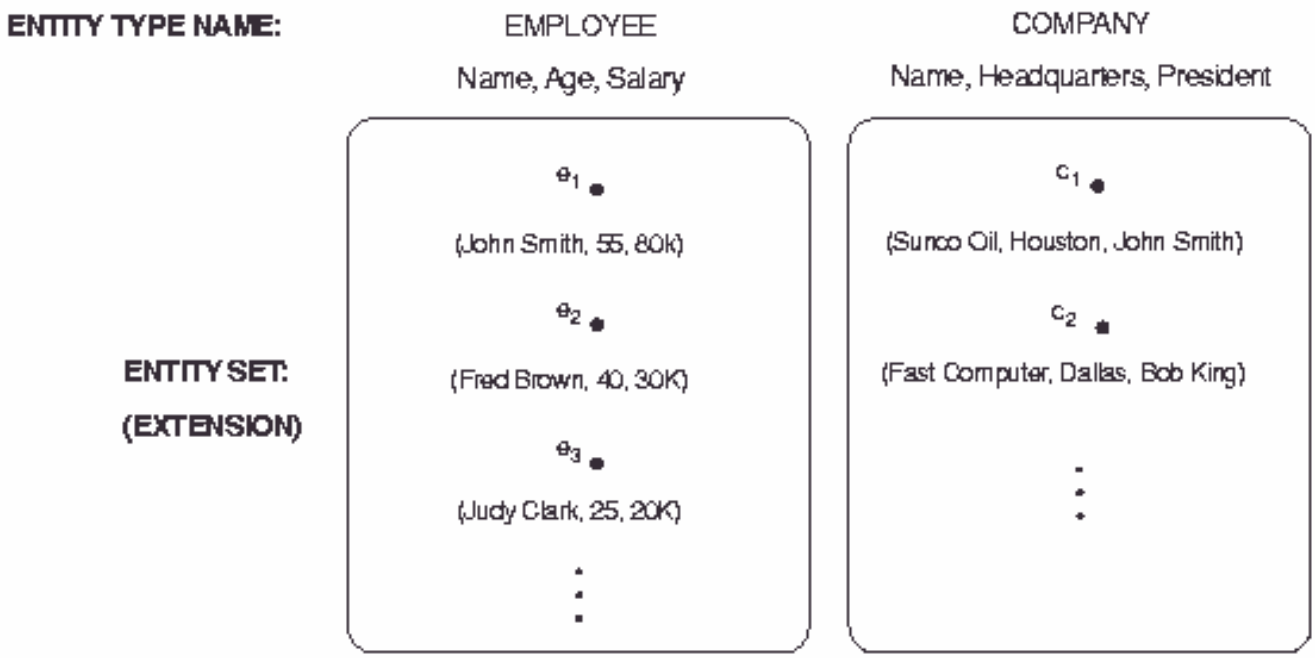
  The entity set of an entity type is also called **extension** of this entity type.

4.2. Entity Types (2)

Entity types and Entity sets

Example:

Figure 3.6 Two entity types named EMPLOYEE and COMPANY, and some of the member entities in the collection of entities (or entity set) of each type.



Notation in ER diagrams:

4.2. Entity Types (3)

Key attributes of entity types

- An attribute of an entity type for which each entity must have a *unique value* is called a **key attribute** of the entity type.

  Example: The attribute SSN of the entity type EMPLOYEE.

- A key attribute may be *composite*.

  Example: VehicleRegistrationNumber is a key of the CAR entity type with components (Number, State).

- An entity may not have a key *(weak entity type)*.

- Remark: A key of an entity type is not a property of an extension. It is a constraint that all the extensions of the entity type have to satisfy.

- Notation in ER diagrams:

4.2. Entity Types (4)

Key attributes of entity types

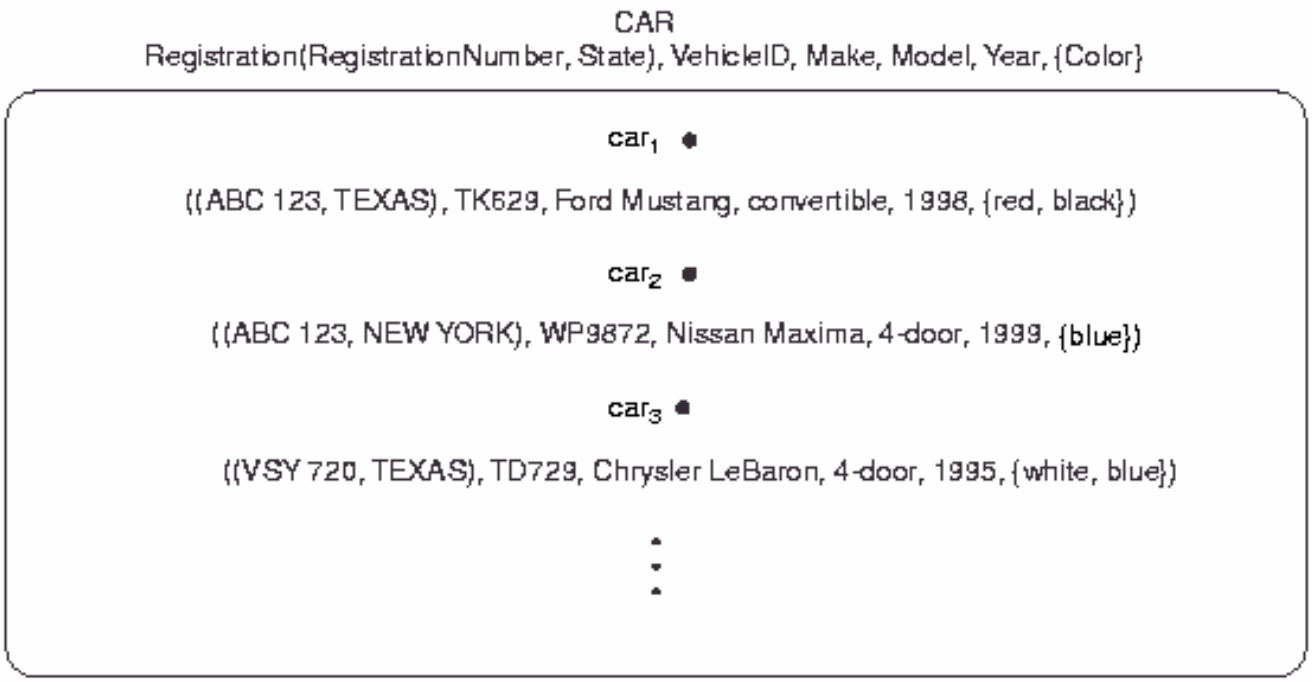
- A key may be formed by *more than one attribute*.

  In this case, the *combination of the attribute values* must be distinct for each entity.

  Such a key must be *minimal*.

Example

Figure 3.7 The CAR entity type, with two key attributes Registration and VehicleID. Multivalued attributes are shown between set braces { }. Components of a composite attribute are shown between parentheses ().



4.2. Entity Types (5)

Initial design of the COMPANY database

3 Preliminary design of entity types for the COMPANY da
whose requirements are described in Section 3.2.

DEPARTMENT
Name, Number, {Locations}, Manager, ManagerStartDate

PROJECT
Name, Number, Location, ControllingDepartment

EMPLOYEE
Name (FName, MInit, LName), SSN, Sex, Address, Salary,
BirthDate, Department, Supervisor, {WorksOn (Project, Hours)}

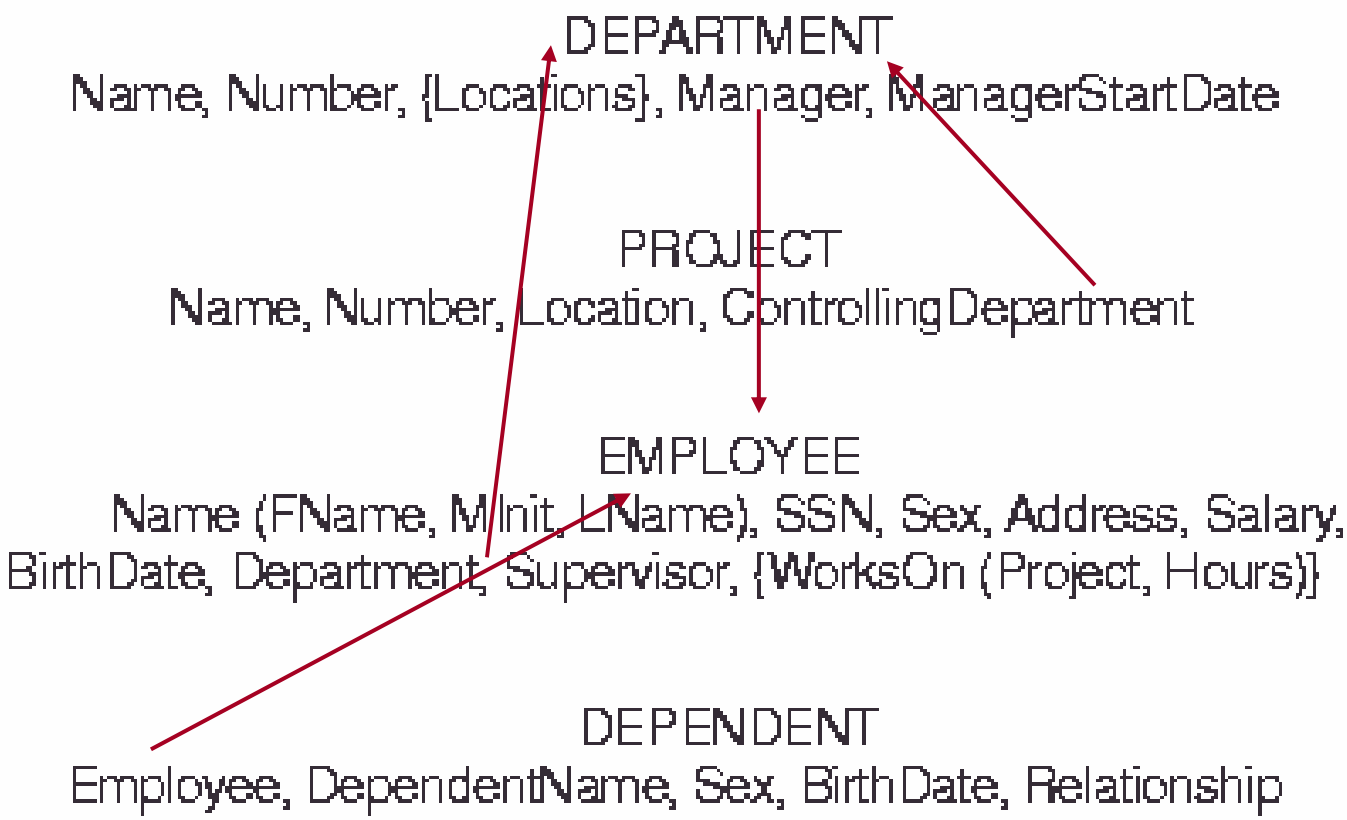
DEPENDENT
Employee, DependentName, Sex, BirthDate, Relationship

4.2. Entity Types (6)

Initial design of the COMPANY database

Intuitive relationships among entity types

3 Preliminary design of entity types for the COMPANY da



4.3. Relationship Types (1)

- A **relationship type** among n entity types E_1, …, E_n defines a set of associations (a **relationship set**) among entities from these types.

  Example: For example, the WORKS_ON relationship type in which EMPLOYEEs and PROJECTs participate, or the MANAGES relationship type in which EMPLOYEEs and DEPARTMENTs participate.

- A **relationship set** is a set of **relationship instances** where each relationship instance associates entities, one from each **participating** entity type.

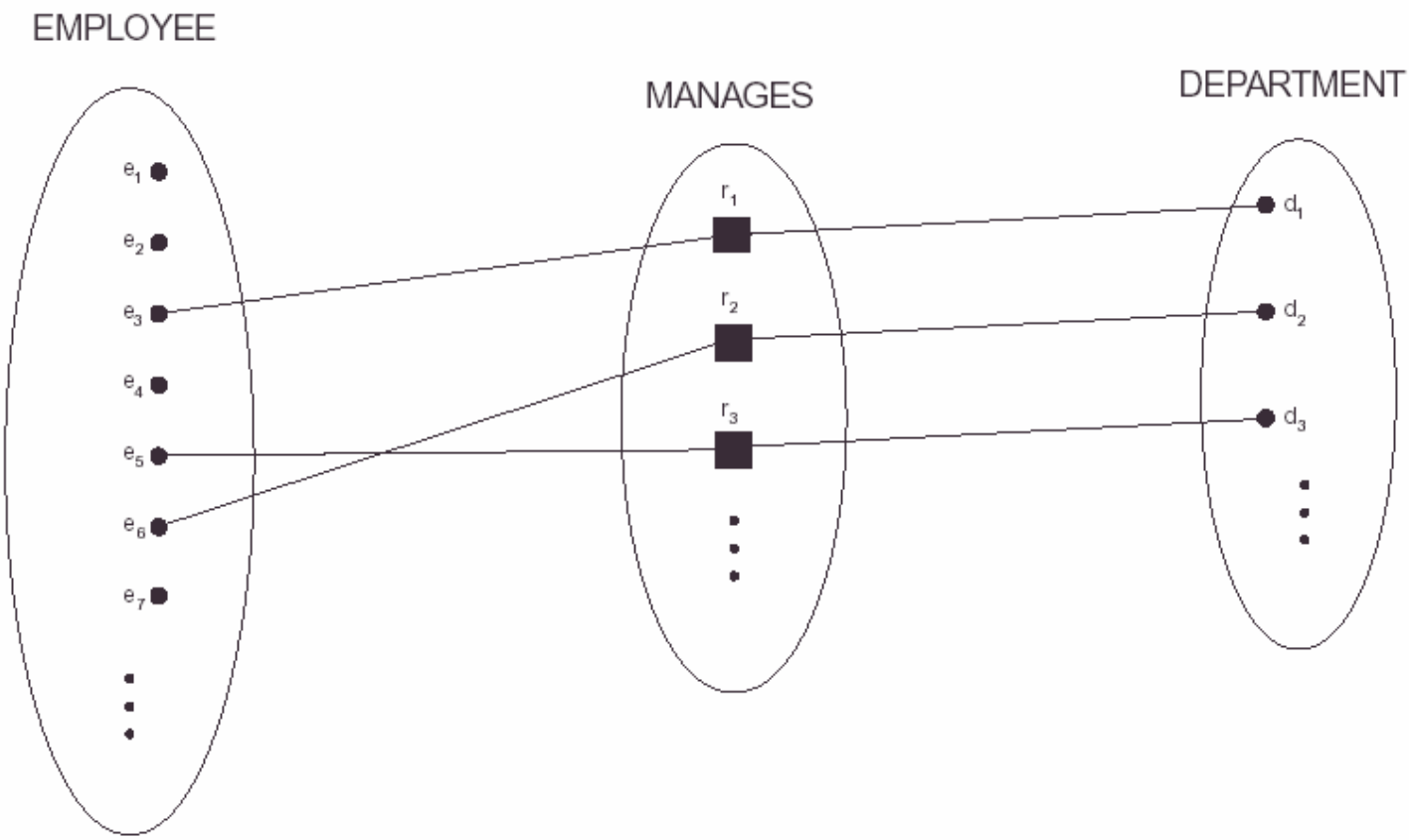
- Notation in ER diagrams.

4.3. Relationship Types (2)

- The degree of a **relationship type** is the number of participating entity types.

  Example: Both MANAGES and WORKS_ON are **binary** relationship types.

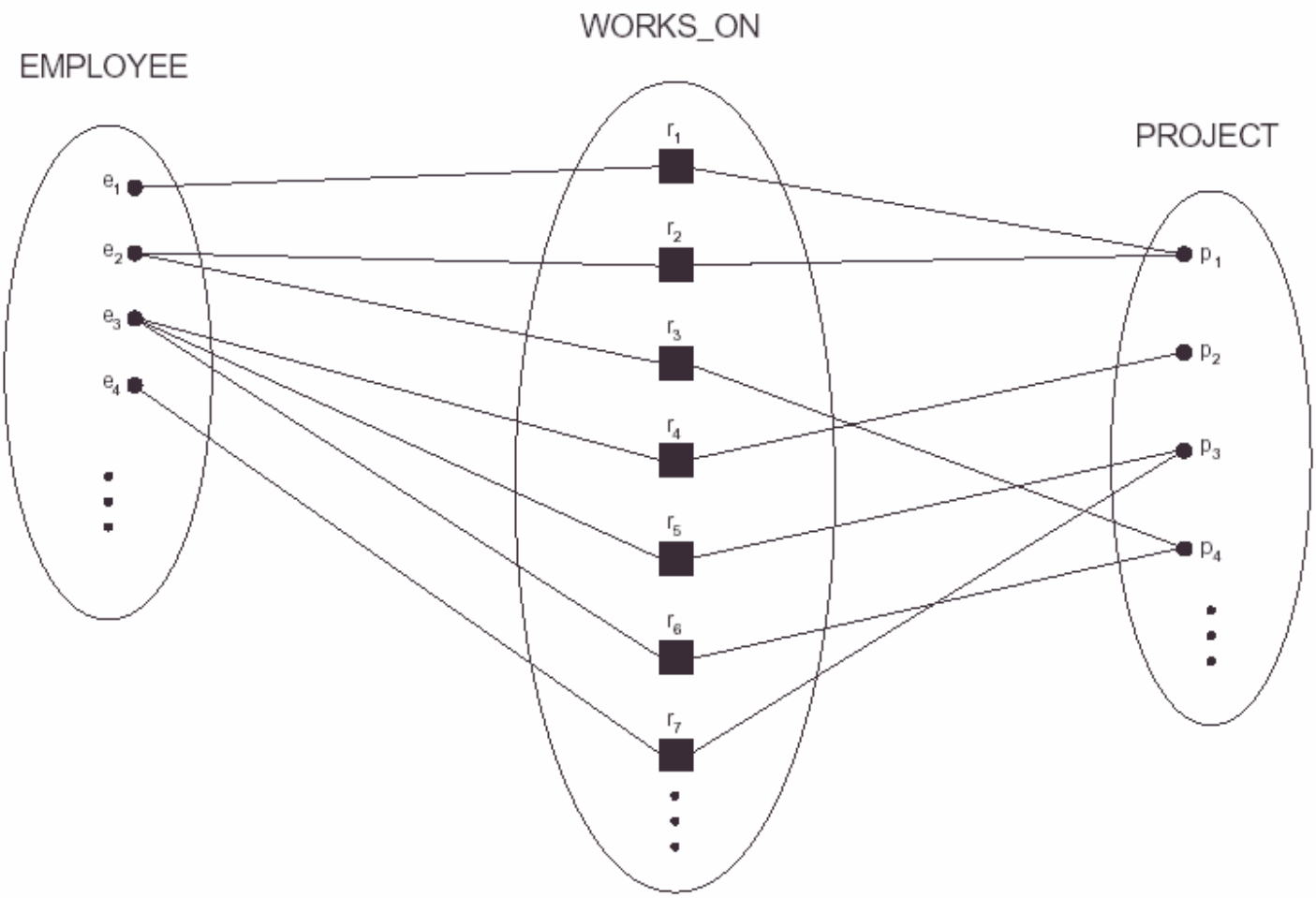
Figure 3.12 The 1:1 relationship MANAGES, with partial participation of employee and total participation of DEPARTMENT.



4.3. Relationship Types (3)

Example of another binary relationship type.

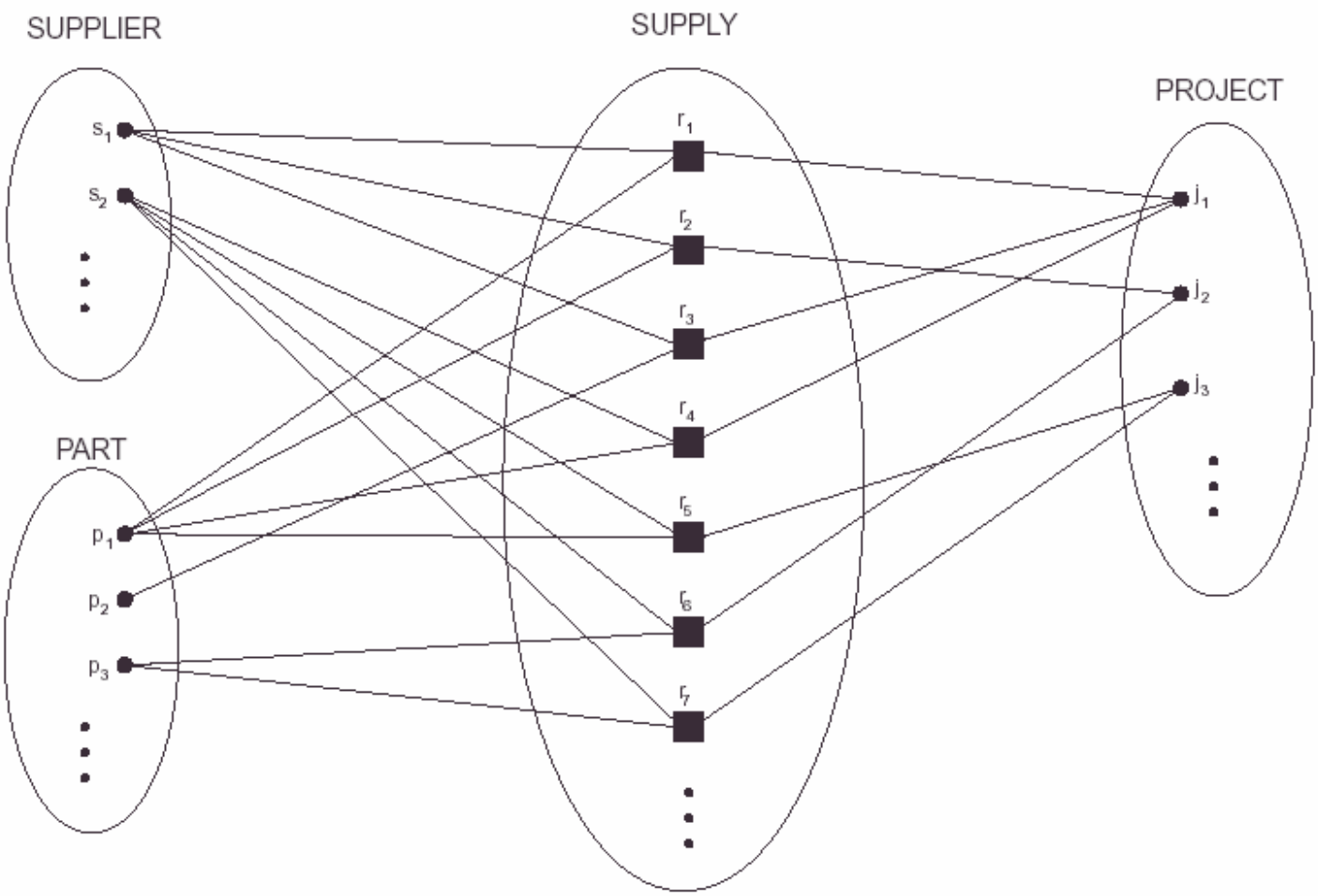
Figure 3.13 The M:N relationship WORKS_ON between EMPLOYEE and PROJECT.



4.3. Relationship Types (4)

Example of a ternary relationship type.

Figure 3.10 Some relationship instances of a ternary relationship SUPPLY.



4.3. Relationship Types (5)

- *More than one relationship type* can exist among the same participating entity types:

  Example: MANAGES and WORKS_FOR are distinct relationship types between EMPLOYEE and DEPARTMENT entity types.

Recursive relationships

- A relationship can relate two entities *of the same entity type*. This is called a **recursive** relationship type.

  Example: A SUPERVISION relationship type relates one EMPLOYEE to another EMPLOYEE.

- ER diagram:

4.3. Relationship Types (6)

Role names

- A **role** can be associated with a relationship entity in a relationship type.
  It signifies the role that a participating entity of the entity type plays in the each relationship instance of the relationship type.

- In a recursive relationship type, role names are essential for distinguishing the meaning of the participation of each entity type.

  Example: EMPLOYEE participates in SUPERVISION once in the **role** of *supervisee* and once in the **role** of *supervisor*.

- ER diagram:

4.4. Structural Constraints (1)

Cardinality ratio

- The **cardinality ratio** for a binary relationship specifies the number of relationship instances an entity can participate in.

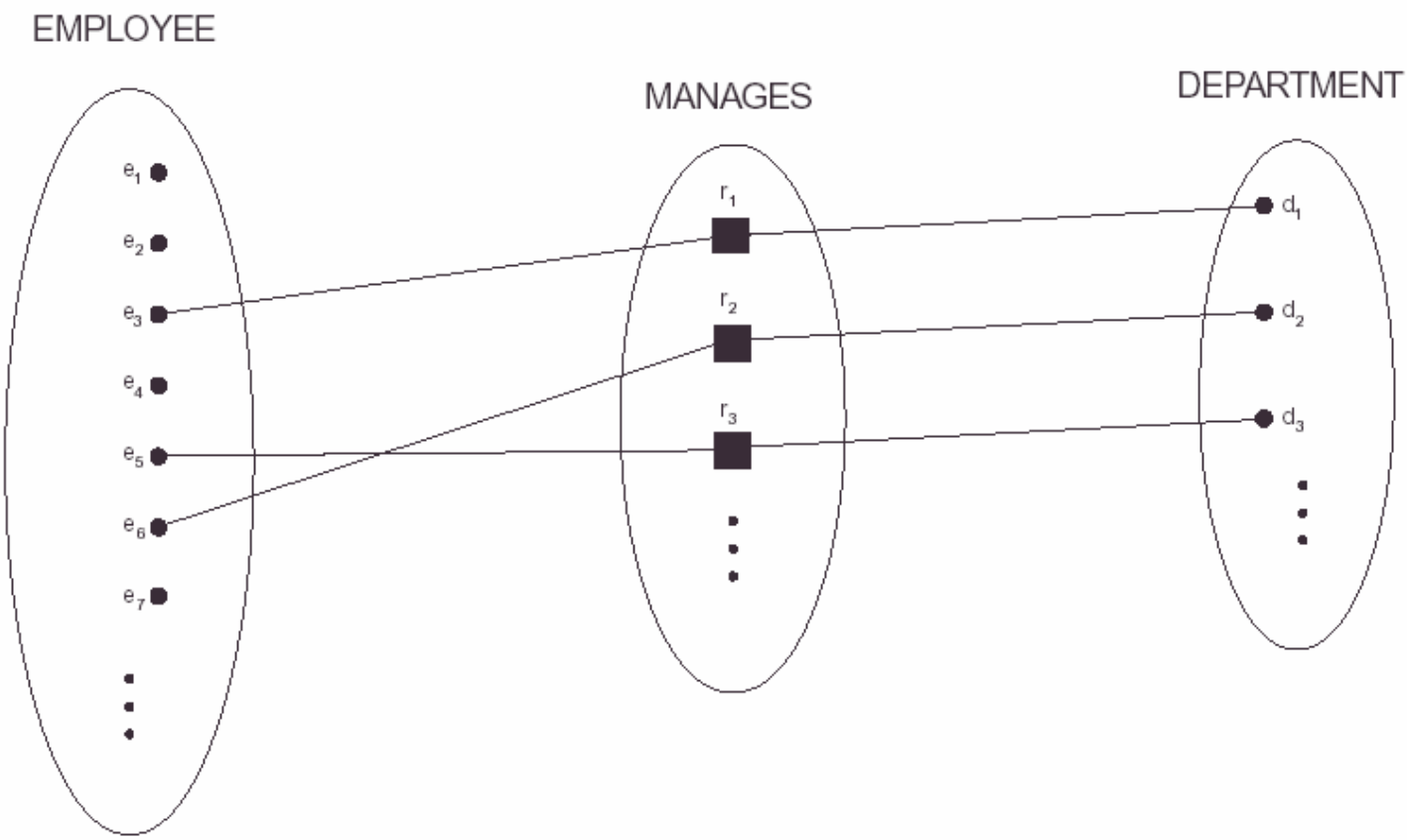
- Possible cardinality ratios for binary relationships are: **1:1, 1:N, M:N**.

4.4. Structural Constraints (2)

Cardinality ratio

- **Example:**
  MANAGES (EMPLOYEE:DEPARTMENT) is a 1:1 relationship type.

Figure 3.12 The 1:1 relationship MANAGES, with partial participation of employee and total participation of DEPARTMENT.

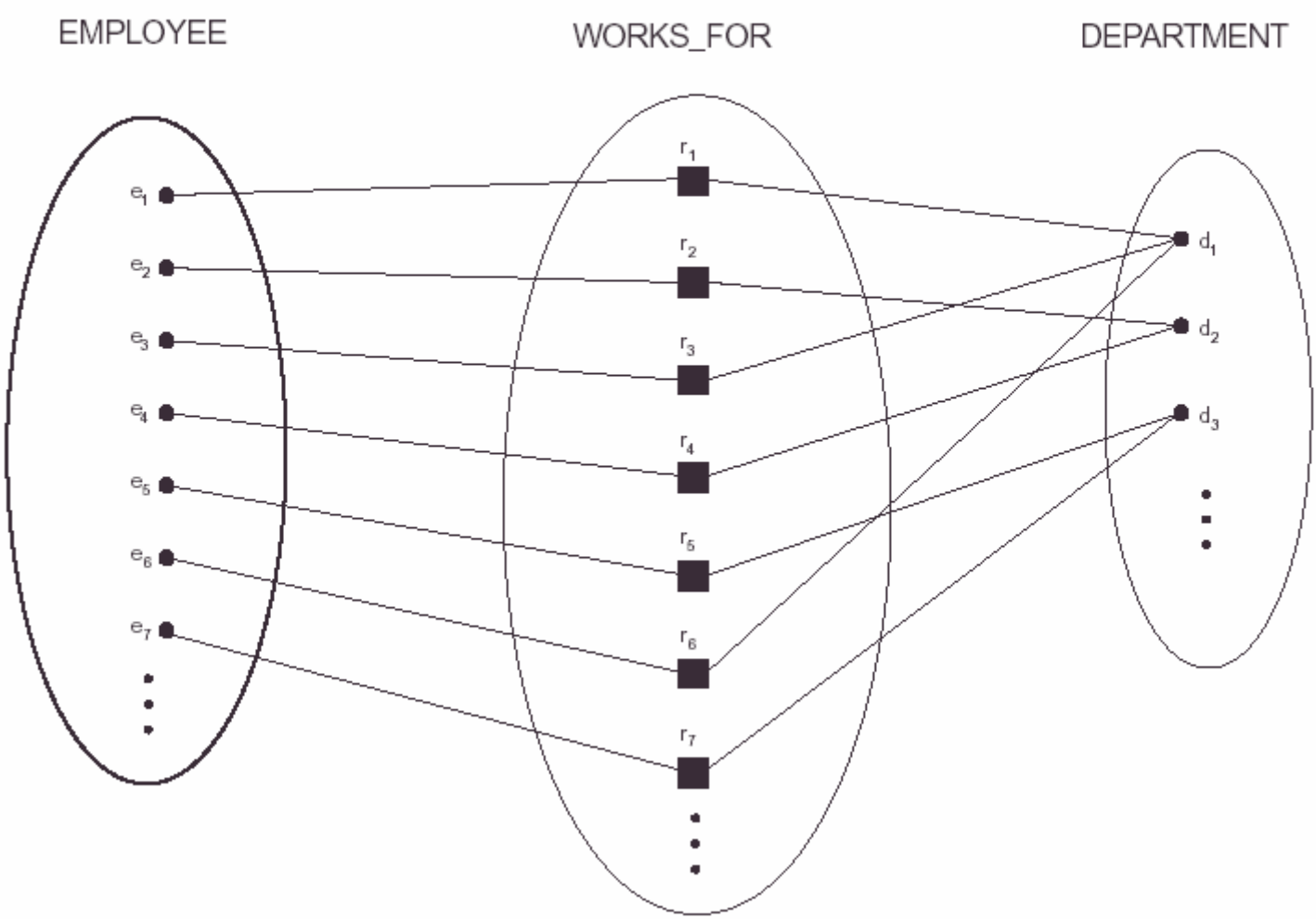


4.4. Structural Constraints (3)

Cardinality ratio

- **Example:**
  WORKS_FOR (EMPLOYEE:DEPARTMENT) is a N:1 relationship type.

Figure 3.9 Some instances of the WORKS_FOR relationship between EMPLOYEE and DEPARTMENT.

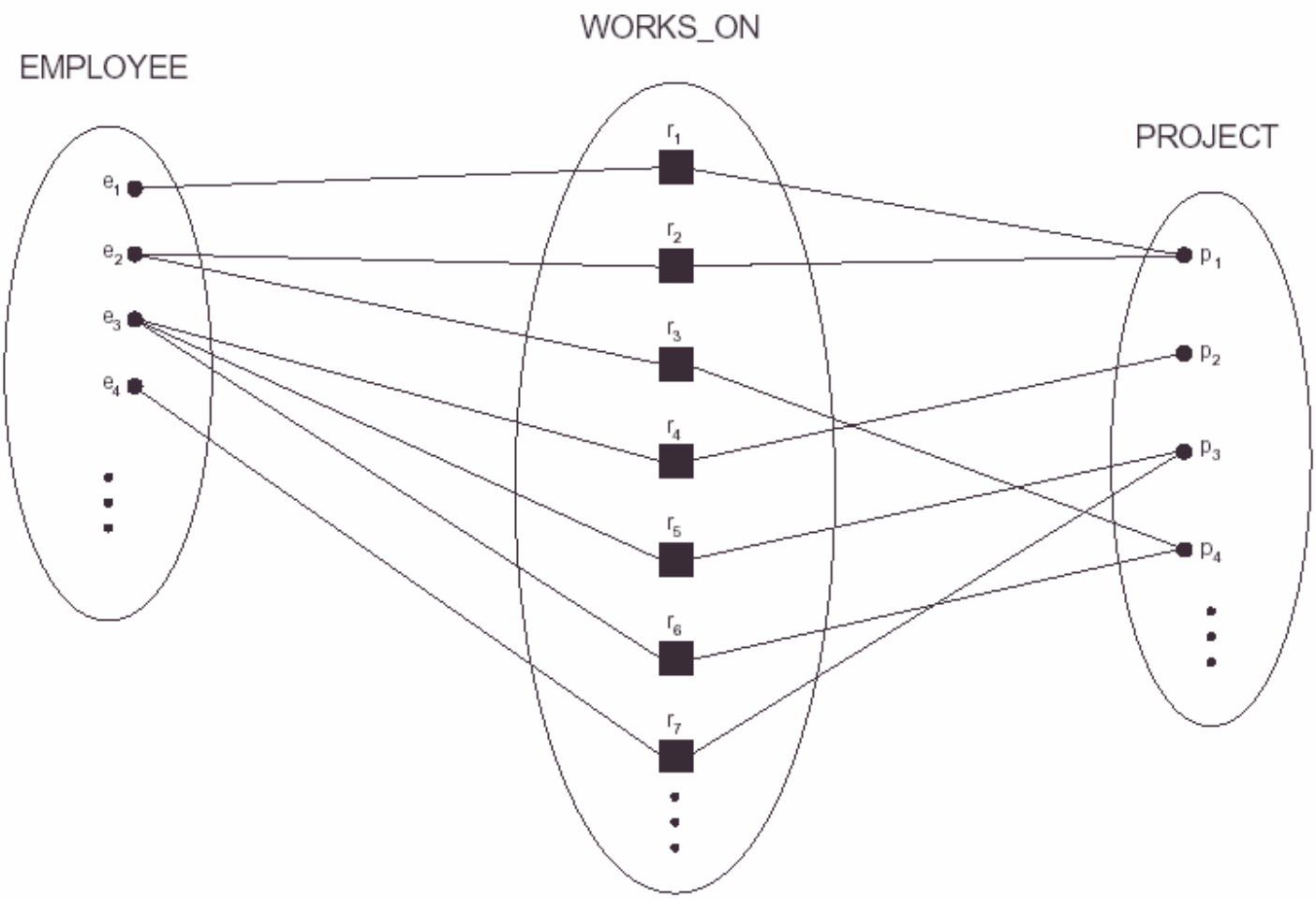


4.4. Structural Constraints (4)

Cardinality ratio

- **Example:**
  WORKS_ON (EMPLOYEE:PROJECT) is a M:N relationship type.

Figure 3.13 The M:N relationship WORKS_ON between EMPLOYEE and PROJECT.



- Notation in ER diagrams:

4.4. Structural Constraints (5)

Participation Constraints

- Two types: **Total participation** (also called **existence dependency**) and **Partial participation**.

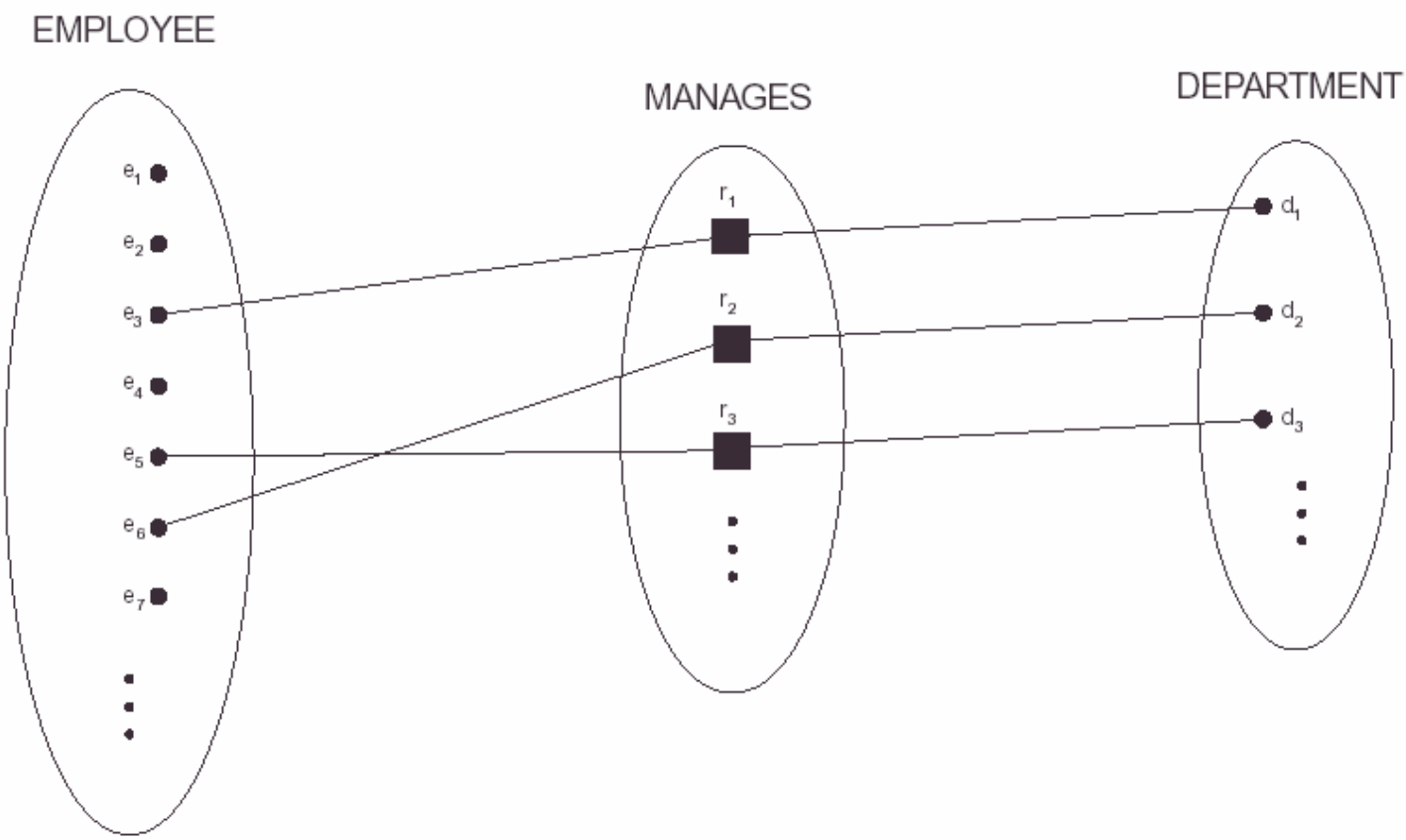
- The participation of an entity type E to a relationship type is a **total participation** if *every entity of E* is associated with an instance of the relationship type. Otherwise, it is a **partial participation**.

4.4. Structural Constraints (6)

Participation Constraints

- **Example:**
  The participation of DEPARTMENT to MANAGES is total. The participation of EMPLOYEE to MANAGES is partial.

Figure 3.12 The 1:1 relationship MANAGES, with partial participation of employee and total participation of DEPARTMENT.



- Notation in ER diagrams:

4.5. Relationship Attributes

- A relationship type *can also have attributes*.

  Example: HoursPerWeek of WORKS_ON; its value for each relationship instance describes the number of hours per week that an EMPLOYEE *works on* a PROJECT.

  Also StartDate on MANAGES

- Sometimes, an attribute of a relationship type *can migrate* to the participating entity types.

  - Notation in ER diagrams:

4.6. Weak entity types (1)

- An entity type that does not have a key attribute
- A weak entity type must participate in an **identifying relationship type** with an **owner** or **identifying entity type.**
- The **partial key** of weak entity type is the set of attributes of the weak entity type that uniquely identify the entities of the weak entity type that are associated with the same identifying entity.
- Entities are identified by *the combination of* :

  -A **partial key** of the weak entity type.
  -The *particular entity* they are related to in the identifying entity type

4.6. Weak entity types (2)

- Example:

  Suppose that a DEPENDENT entity is identified by the dependent's first name, *and* the specific EMPLOYEE that the dependent is related to. DEPENDENT is a weak entity type with EMPLOYEE as its identifying entity type via the identifying relationship type DEPENDENT_OF.

- Notation in ER diagrams:

4.7. Adding relationships to the COMPANY ER schema

Relationship types:

- MANAGES (EMPLOYEE, DEPARTMENT)
- WORKS_FOR (EMPLOYEE, DEPARTMENT)
- CONTROLS (DEPARTMENT, PROJECT)
- SUPERVISION (EMPLOYEE – role:supervisor, EMPLOYEE – role: supervisee)
- WORKS_ON (EMPLOYEE, PROJECT)
- DEPENDENTS_OF (EMPLOYEE, DEPENDENT)

5. Notation (1)

(Min, Max) notation for structural constraints

- Specified on *each participation* of an entity type E in a relationship type R.
- Specifies that each entity e in E participates in *at least* min and *at most* max relationship instances in R.
- Default (no constraint): min=0, max=n.
- Must have min≤max, min≥0, and max≥1.
- Derived from the mini-world constraints.

5. Notation (2)

(Min, Max) notation for structural constraints

Examples:

(a) A department has *exactly one* manager.

Specify (1,1) on the participation of DEPARTMENT into MANAGES.

(b) An employee can manage *at most one* department.

Specify (0,1) on the participation of EMPLOYEE into MANAGES.

(c) An employee works for *exactly one* department.

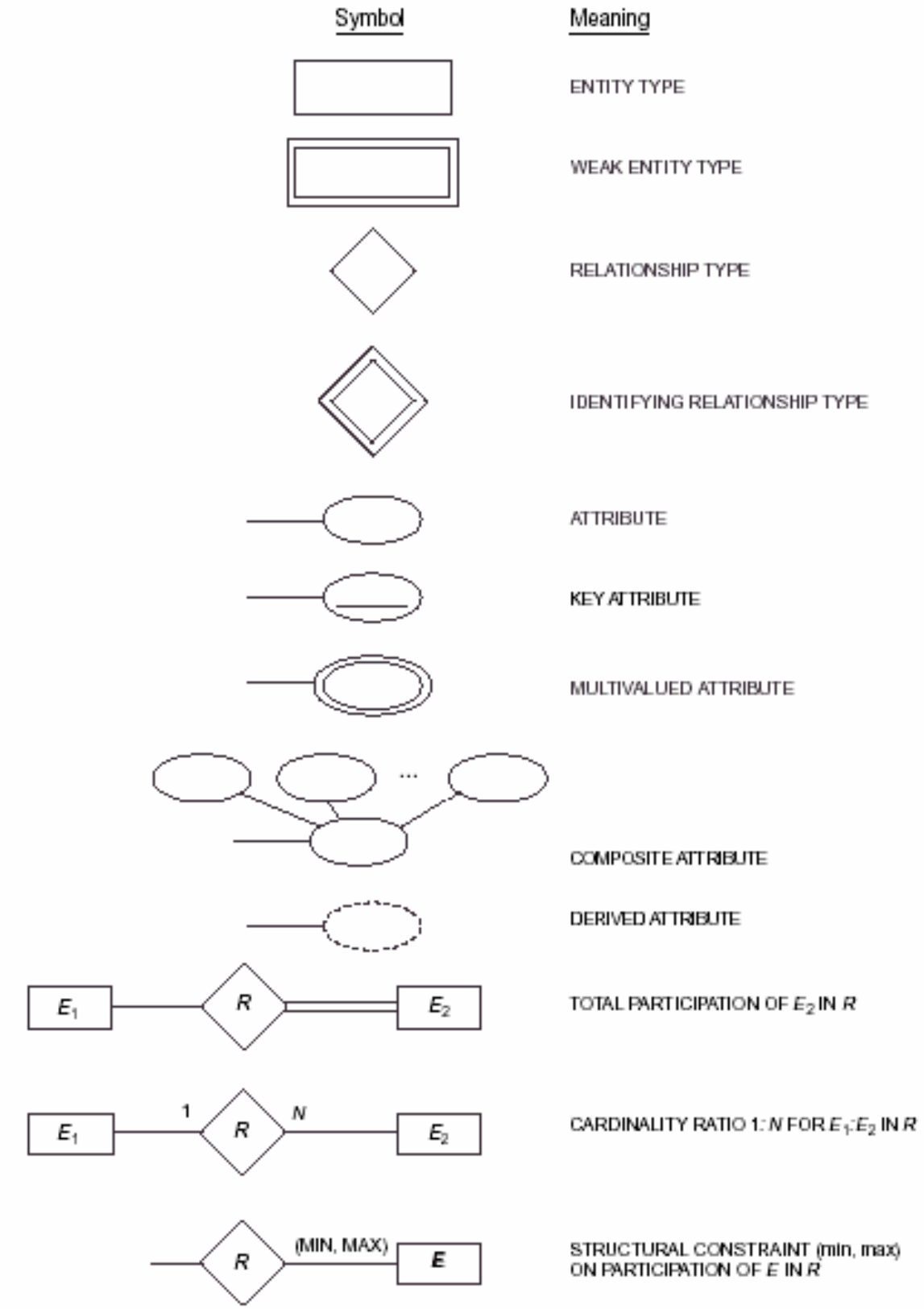
Specify (1,1) on the participation of EMPLOYEE into WORKS_FOR.

(d) A department can have *any number of* employees but *at least four*.

Specify (4,n) on the participation of DEPARTMENT into WORKS_FOR.

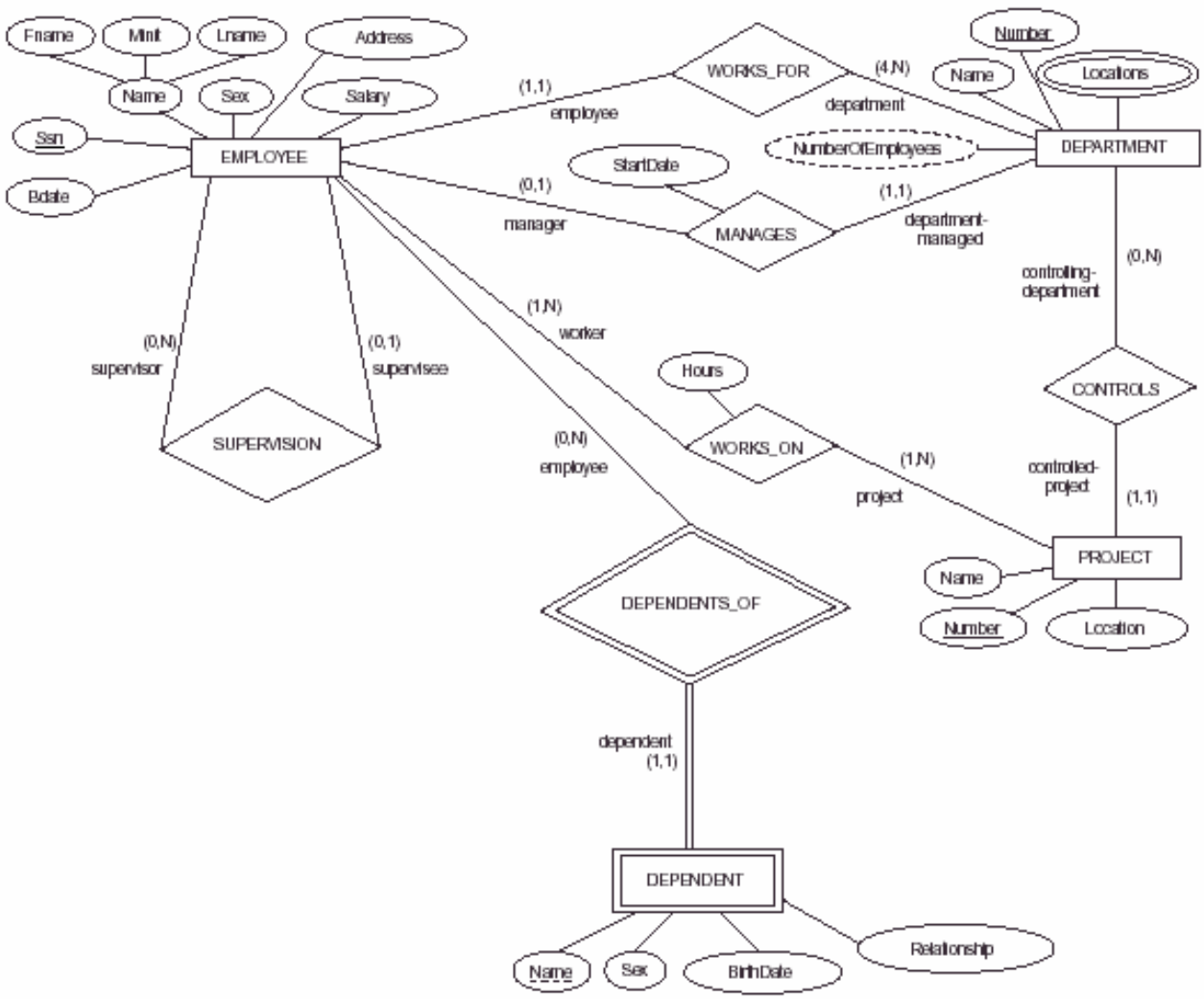
5. Notation (3)

Summary of ER diagram notation



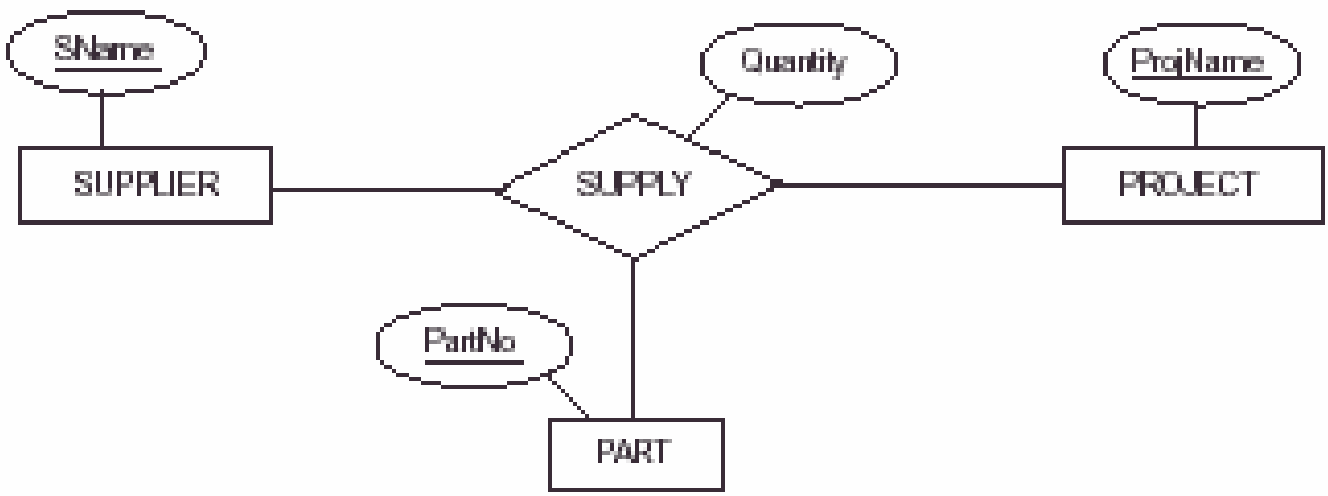
5. Notation (4)

Figure 3.15 ER diagram for the COMPANY schema, with all role names included and with structural constraints on relationships specified using the alternate notation (min, max).



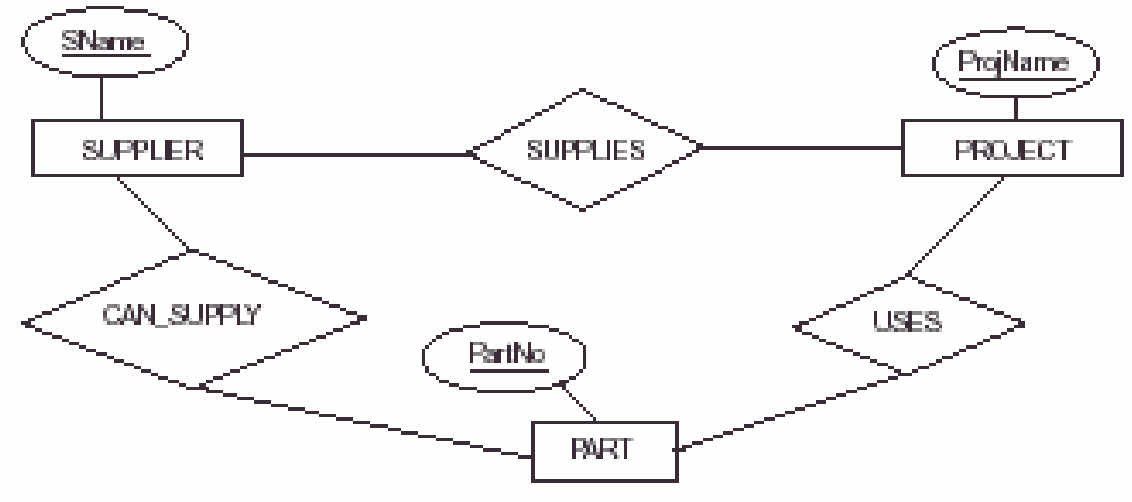
6. Relationships of higher degree (1)

- A *ternary relationship type* SUPPLIES involving entity types SUPPLIER, PART, PROJECT
- Meaning: a supplier s supplies a part p to a project j.
- Notation



6. Relationships of higher degree (2)

- In general, a ternary relationship type represents more information than do three binary relationship types.

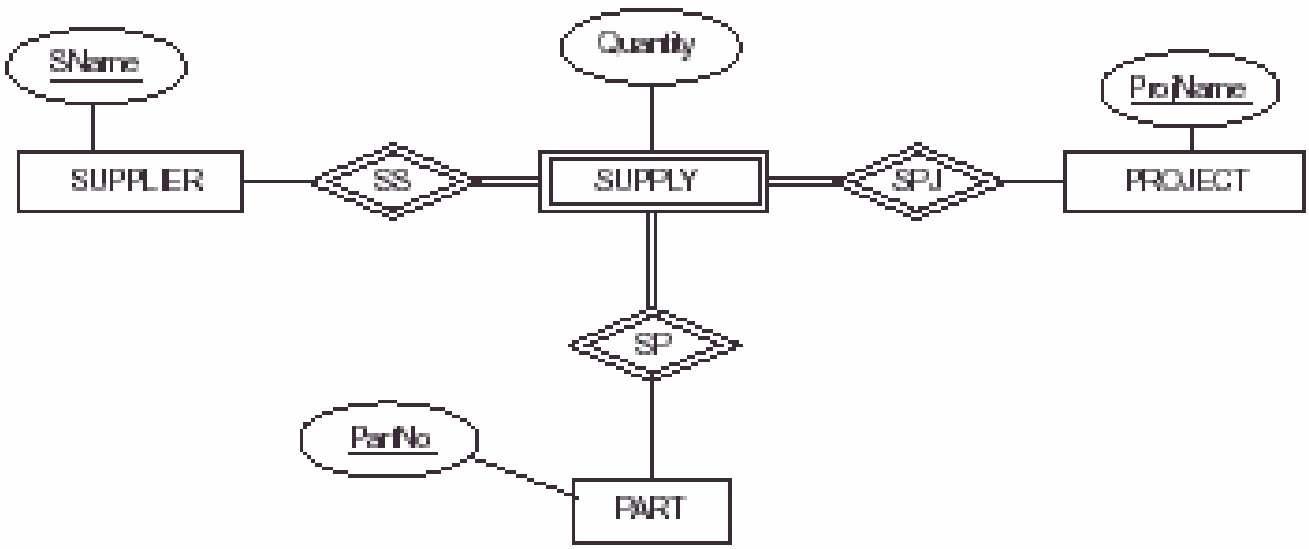


- SUPPLIES: supplier s supplies project j.
  CAN_SUPPLY: supplier s can supply part p.
  USES: project j uses part p.
- The existence of relationship instances (s, j), (s, p), and (j, p) does not imply the existence of a relationship instance (s, p, j).

- A typical solution is to include one ternary relationship type plus one or binary relationship types as needed.

6. Relationships of higher degree (3)

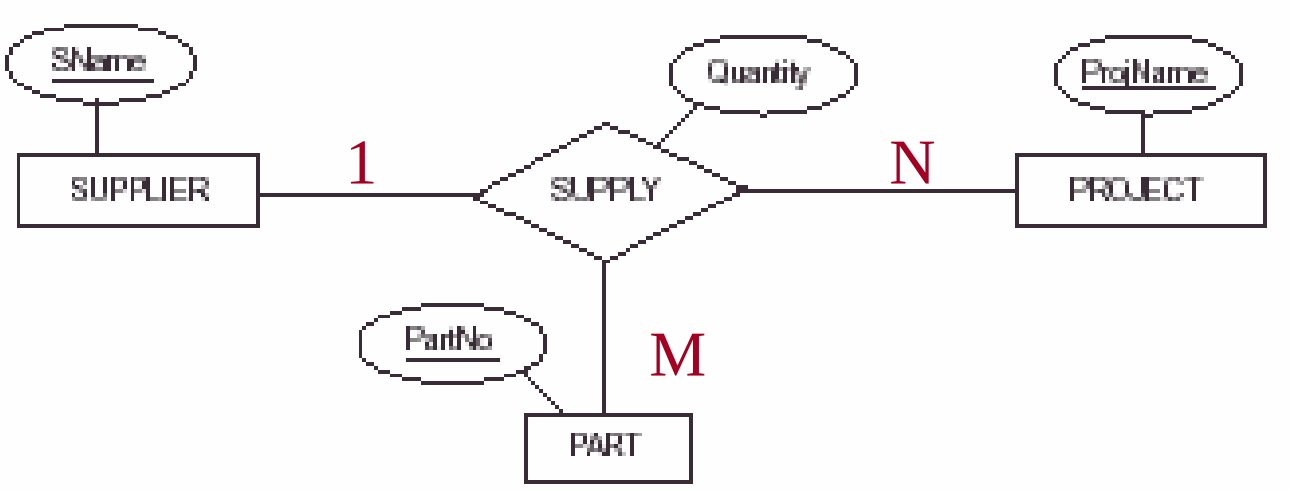
- Some variations of the ER model use only binary relationships. In this case a ternary relationship type is represented by a weak entity type with *no partial key*, and *three identifying relationship types*.



6.1. Constraints on Higher Degree Relationships (1)

Cardinality ratio notation

Example



A 1 in the participation of SUPPLIER to SUPLLY means that for a particular project-part combination, *only one supplier can be used*.

It specifies the constraint that a particular (j, p) combination can appear *at most once* in the relationship set of SUPPLY.

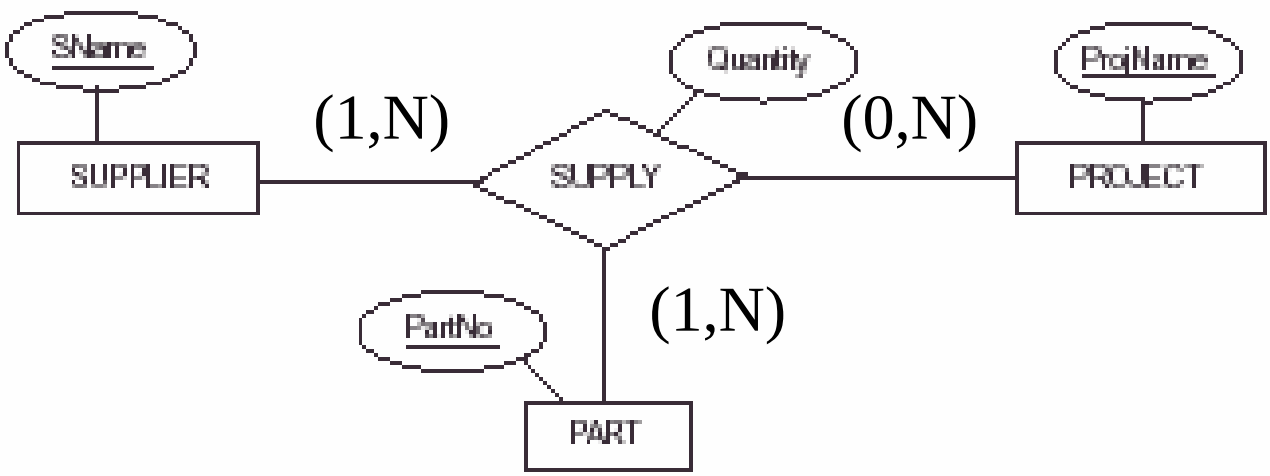
Each (project, part) combination *uniquely determines a single supplier* in the relationship set of SUPPLY.

6.1. Constraints on Higher Degree Relationships (2)

(Min, Max) notation

- In relationship types of degree higher than two, a (min, max) on a participation of an entity type E into a relationship type R specifies that each entity of E is related to at least min and at most max *relationship instances* in the relationship set of R.

Example



7. Enhanced ER (EER) model (1)

New database applications have more *complex requirements* that *cannot be captured by the ER model*.

Limitations of the ER model

- No relationship may be defined between an entity type and a relationship type.

- No relationship may be defined between an entity type and a collection of entity types from which any one type may participate.

  E.g. Entity type1 : POLICY-HOLDER may be an individual, multiple individuals, one organization, or many organizations

  Entity type2 : POLICY

- No constraints (exclusion, co-existence etc.) among relationship types.

7. Enhanced ER (EER) model (2)

Semantic data models have been suggested to capture many of these new requirements.

The **Enhanced ER (EER)** model includes all the modeling concepts of the ER model plus the concepts of **subclass** and **superclass**.

Object data models include many of the concepts of semantic data models but also concepts for modeling *operations* (which specify *functional requirements*).

Popular object modeling methodologies in software design and engineering:
OMT (Object Modeling Technique) and
UML (Universal Modeling Language).

7.1. Subclasses and Superclasses (1)

- An entity type E' is a **subclass** of an entity type E if the entity set of E' is a subset of the entity set of E. E is called **superclass** of E'.

- Example:
  The set of entities of the EMPLOYEE entity type can be grouped into the SECRETARY, ENGINEER, MANAGER, TECHNICIAN, SALARIED_EMPLOYEE, HOURLY_EMPLOYEE etc. entity types.

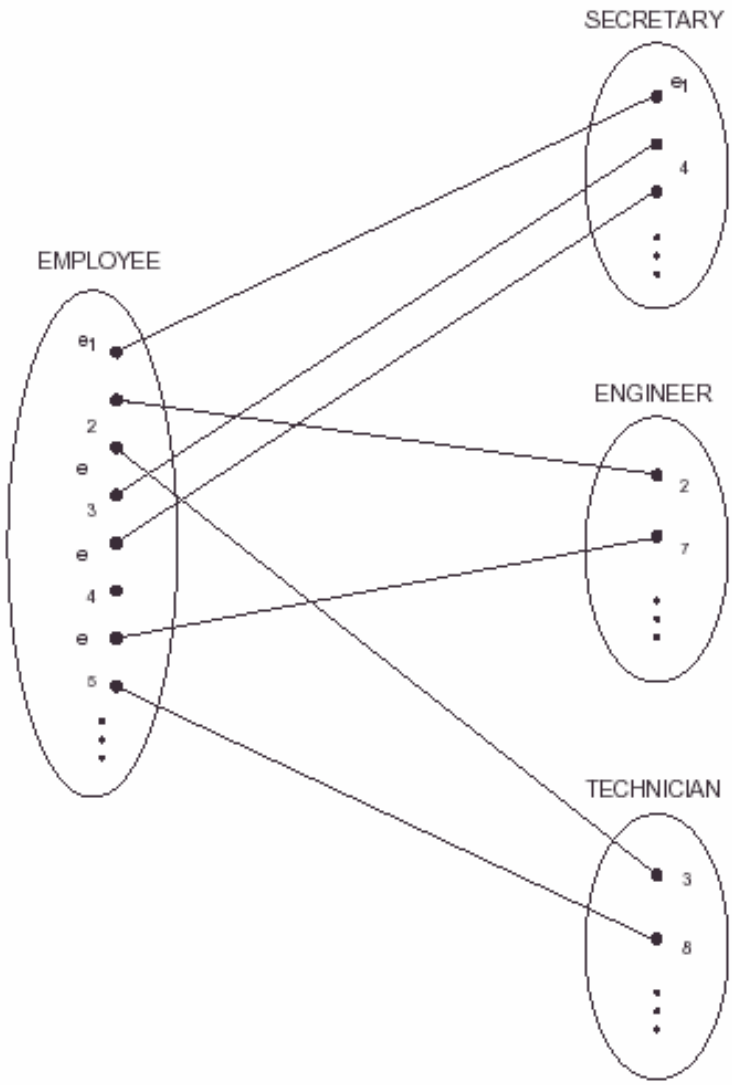
- A member of a superclass *need not necessarily be a member of some subclass*.

- The relationship between a subclass and any of its superclasses is called **superclass/subclass relationship** (or **IS-A relationship**).

7.1. Subclasses and Superclasses (2)

- Example:

4.2 Some instances of the specialization of EMPLOYEE into the {SECRETARY, ENGINEER, TECHNICIAN] set of subclasses.



7.2. Type inheritance

- A subclass entity type inherits all the **attributes** of the supeclass entity type and all the **relationship types** in which the superclass participates.

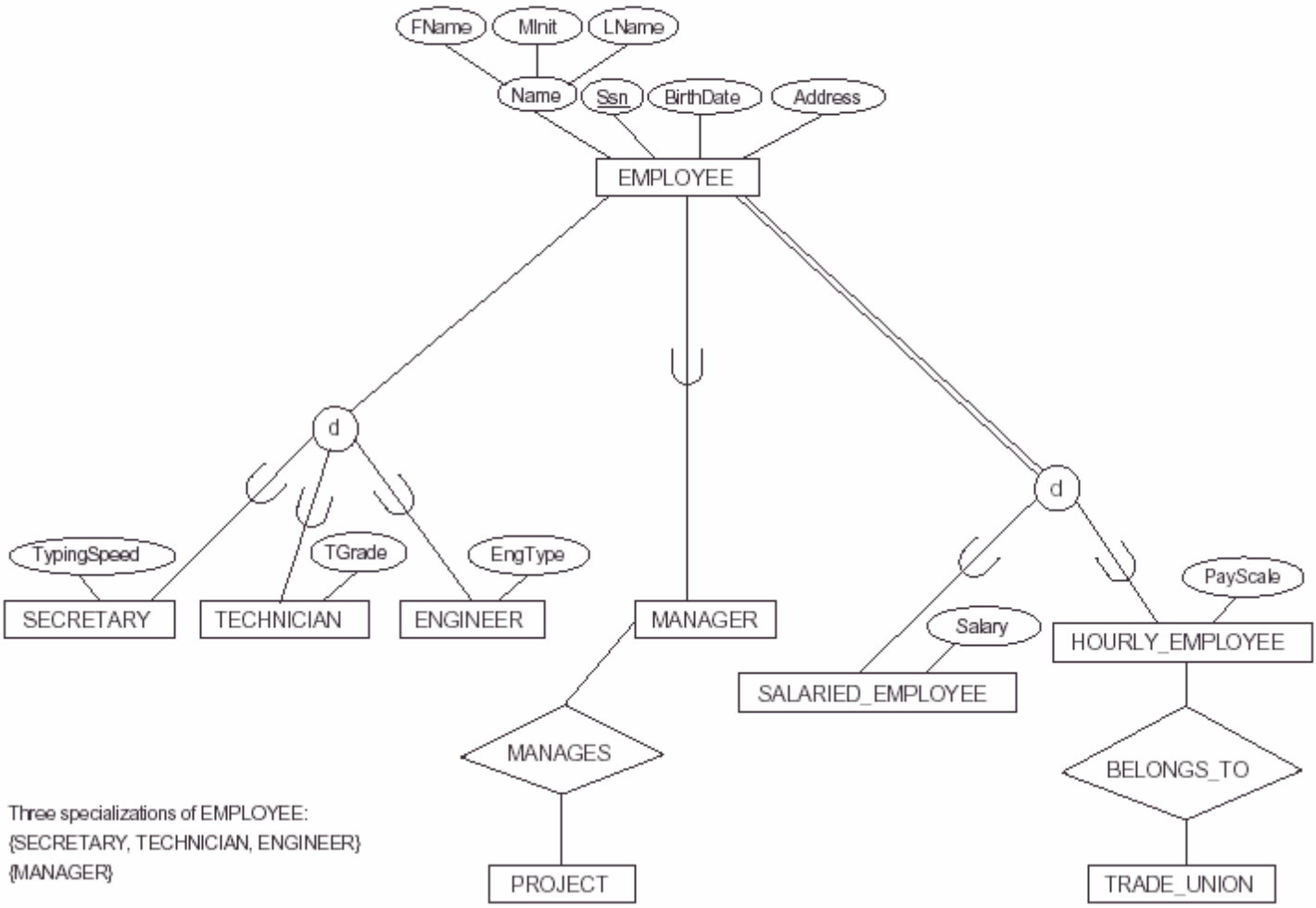
- An entity cannot exist in the database by being merely a member of a subclass; it must also be a member of a superclass.

- A subclass *with its own attributes together with the attributes and the relationship types it inherits* can be considered as an entity type in its own right.

7.3. Specialization

- **Specialization** is the process of defining a set of subclasses of an entity type on the basis of a some **distinguishing characteristic** of the entities in the superclass.

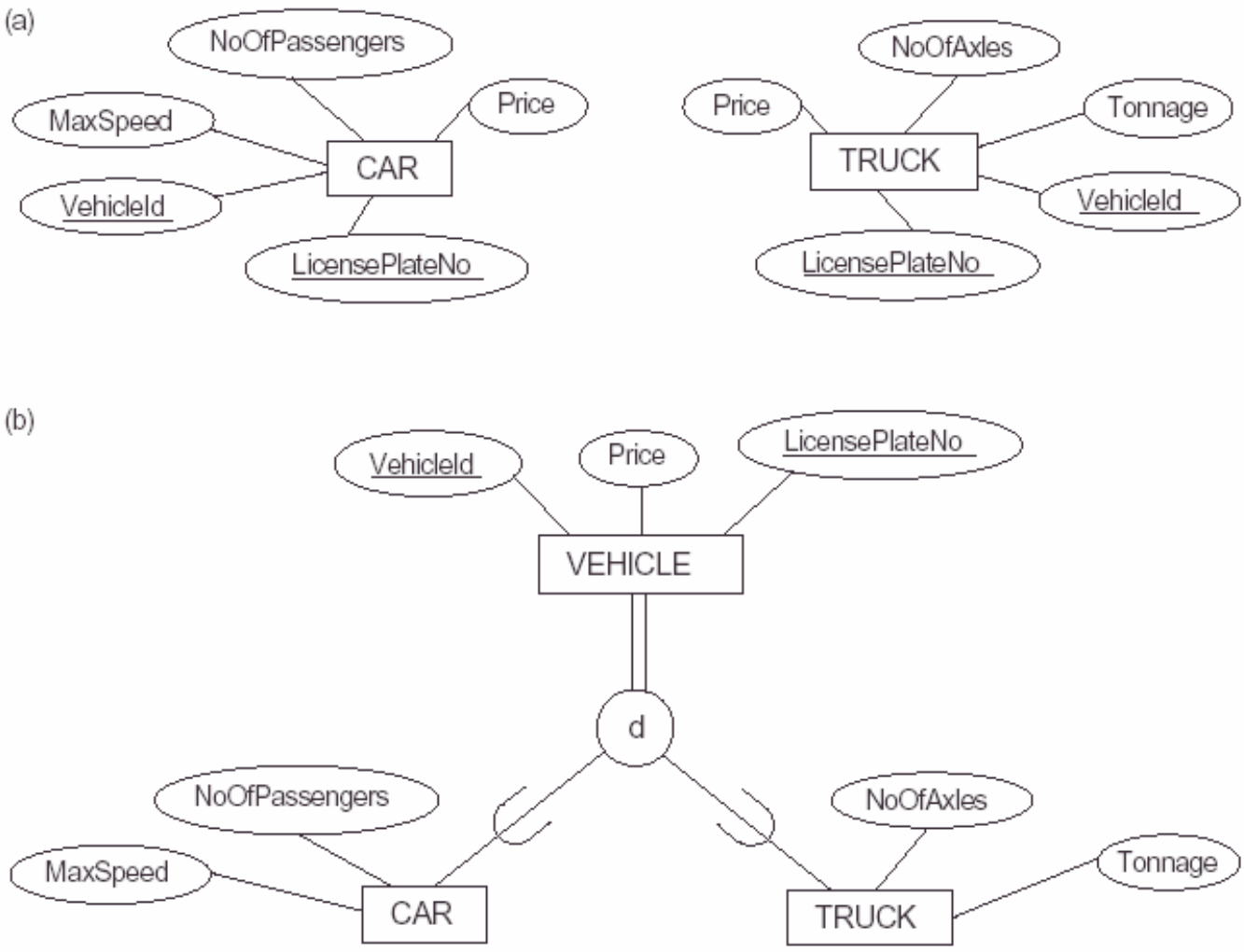
Figure 4.1 EER diagram notation for representing specialization and subclasses.



7.4. Generalization

- **Generalization** is the process of defining a single superclass from a set of entity types (subclasses) by identifying their common characteristics.

gure 4.3 Examples of generalization. (a) Two entity types CAR and TRUCK. (b) Generalizing car and TRUCK into VEHICLE.

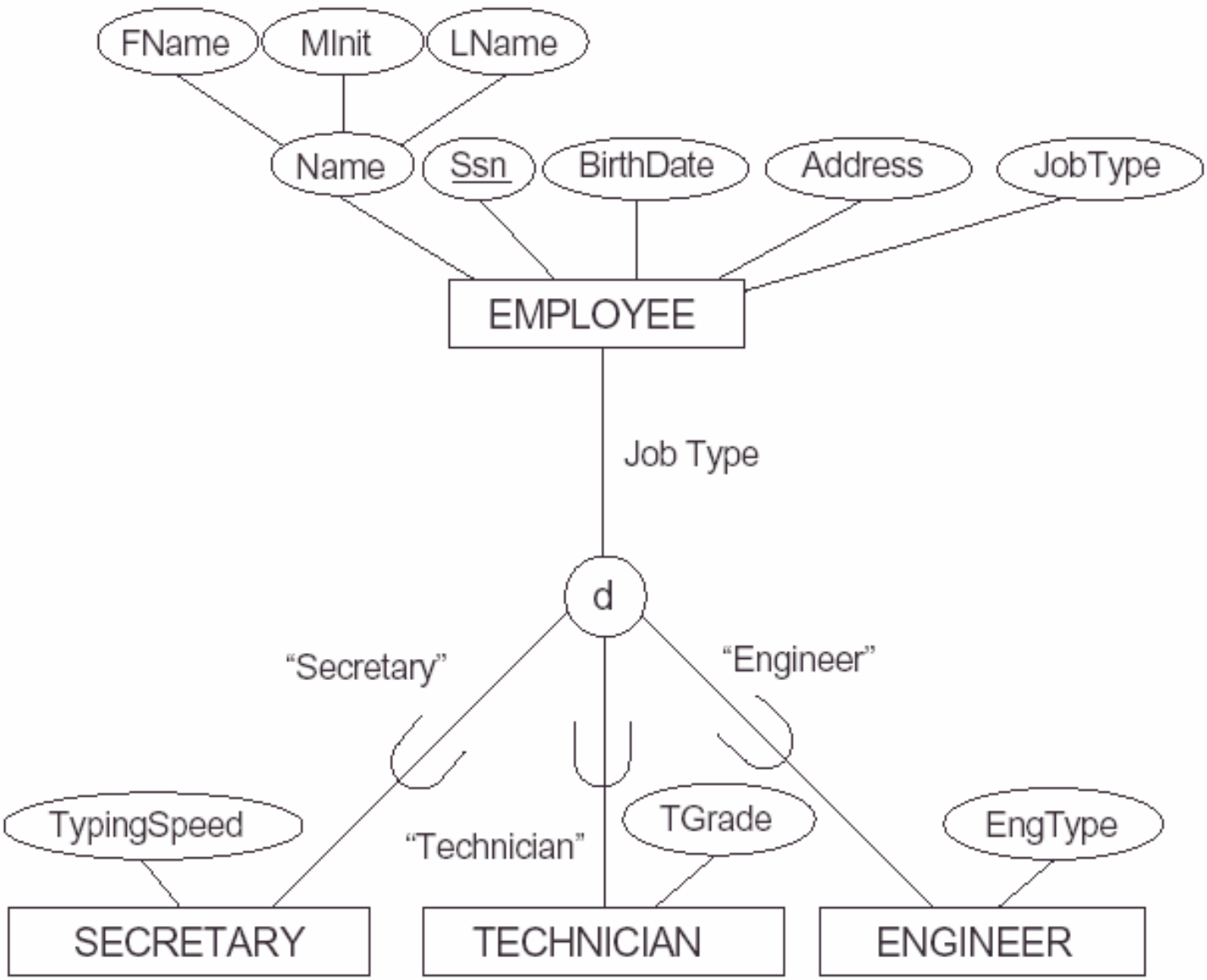


7.5. Constraints of Specialization (1)

Disjointness constraint

- The **disjointness constraint** states that the subclasses of the specialization must be disjoint.
- **d** notation.

Figure 4.4 An attribute-defined specialization on the JobType attribute of EMPLOYEE.

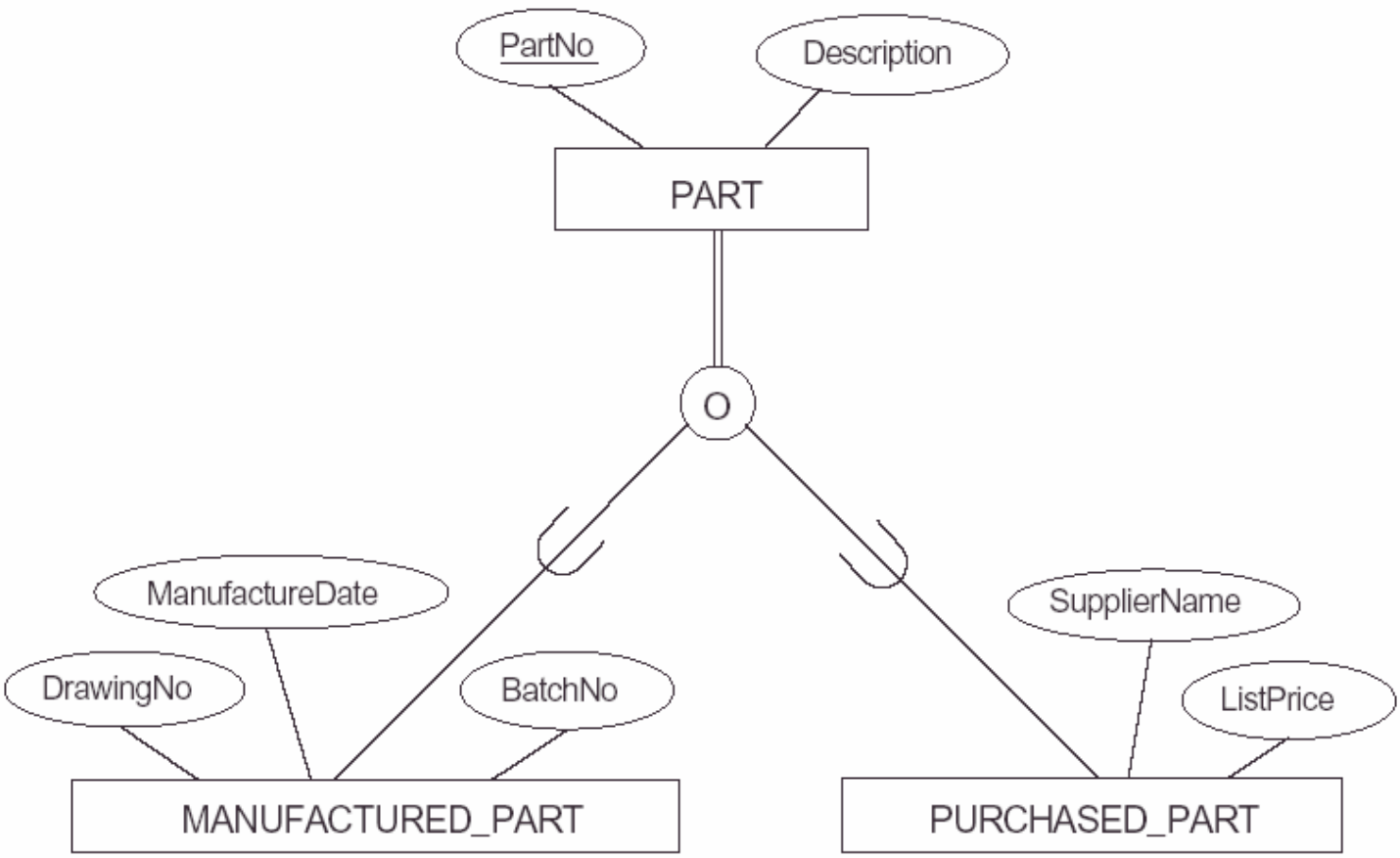


7.5. Constraints of Specialization (2)

Disjointness constraint

- If the subclasses are not constraint to be disjoint their entities may **overlap**.
- **o** notation.

Figure 4.5 Notation for specialization with overlapping (nondisjoint) subclasses.



7.5. Constraints of Specialization (3)

Completeness constraint

- A **completeness constraint** states that every entity in the supercalss must be a member of some subclass in the specialization (this is called **total specialization**).

- A specialization where an entity of the superclass is allowed not to participate in any of the subclasses is called **partial specialization**.

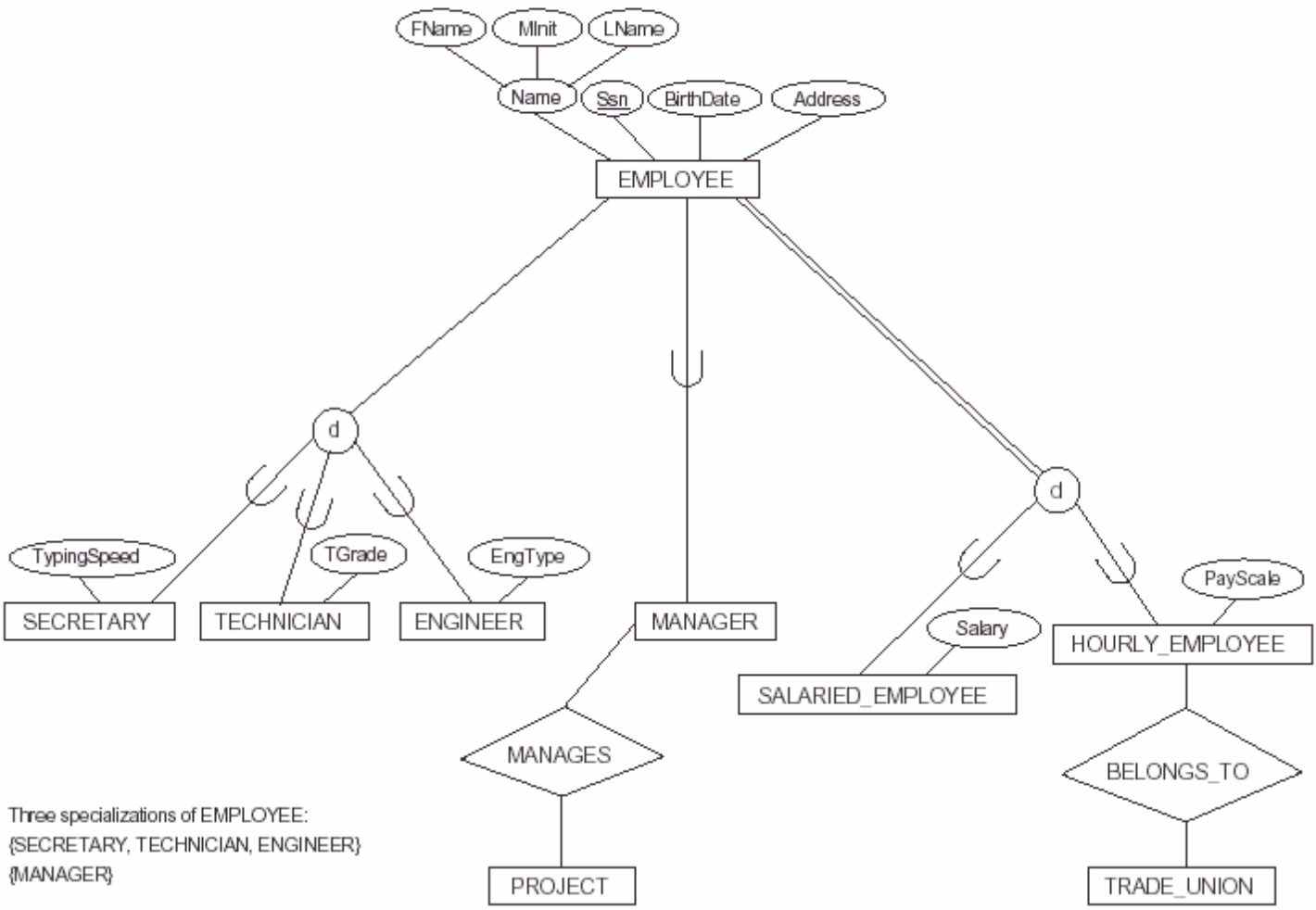
7.5. Constraints of Specialization (4)

Completeness constraint

Example

- **Total specialization: doule line**
- **Partial specialization: single line.**

Figure 4.1 EER diagram notation for representing specialization and subclasses.



7.5. Constraints of Specialization (5)

- Disjointness and completeness constraints are *independent*

  Thus, all four combinations are possible for a specialization:

  Disjoint, Total
  Disjoint, Partial
  Overlapping, Total
  Overlapping, Partial

- Insertion and deletion rules apply to specialization.

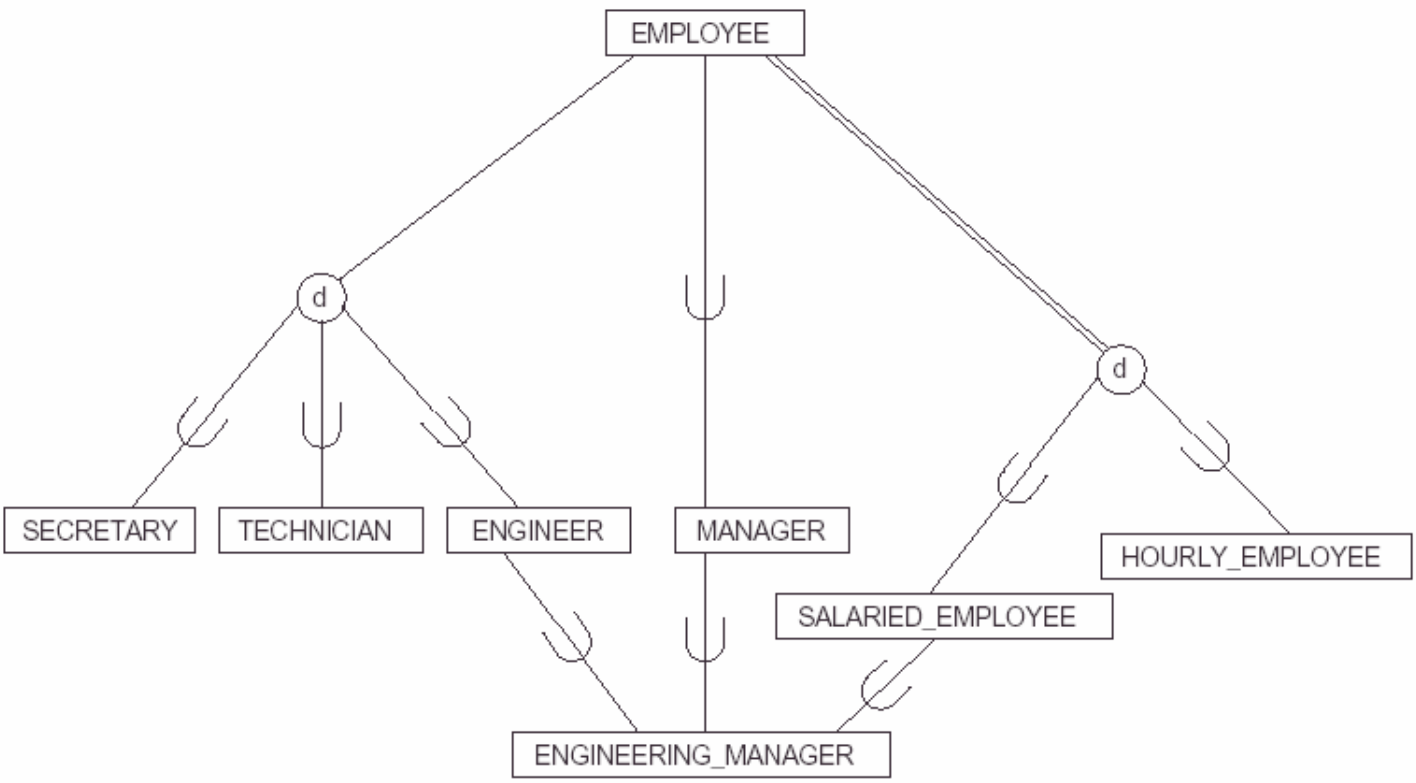
- Once again: the correct constraints to impose to a schema are determined by the *real-world meaning that applies to a specialization*.

7.6. Specialization Hierarchies and Lattices

- A subclass may have further subclasses forming a **hierarchy or lattice of specializations**.

  In a **specialization hierarchy** *every subclass has at most one direct superclass*.
  In a **specialization lattice** *a subclass may have more than one direct superclass*.

Figure 4.6 A specialization lattice with the shared subclass ENGINEERING_MANAGER.



7.6. Multiple inheritance

- In a specialization hierarchy or lattice a subclass inherits attributes and relationship types *from all its direct superclasses and predecessor superclasses* all the way to the root.

- In a specialization lattice a subclass inherits attributes and relationship types *from all its direct superclasses* (**multiple inheritance**).

- Rule: if the same attribute (or relationship) is inherited by a subclass more than once via different paths, in the lattice, then it should be included only once in this subclass.

7.7. Bigger example