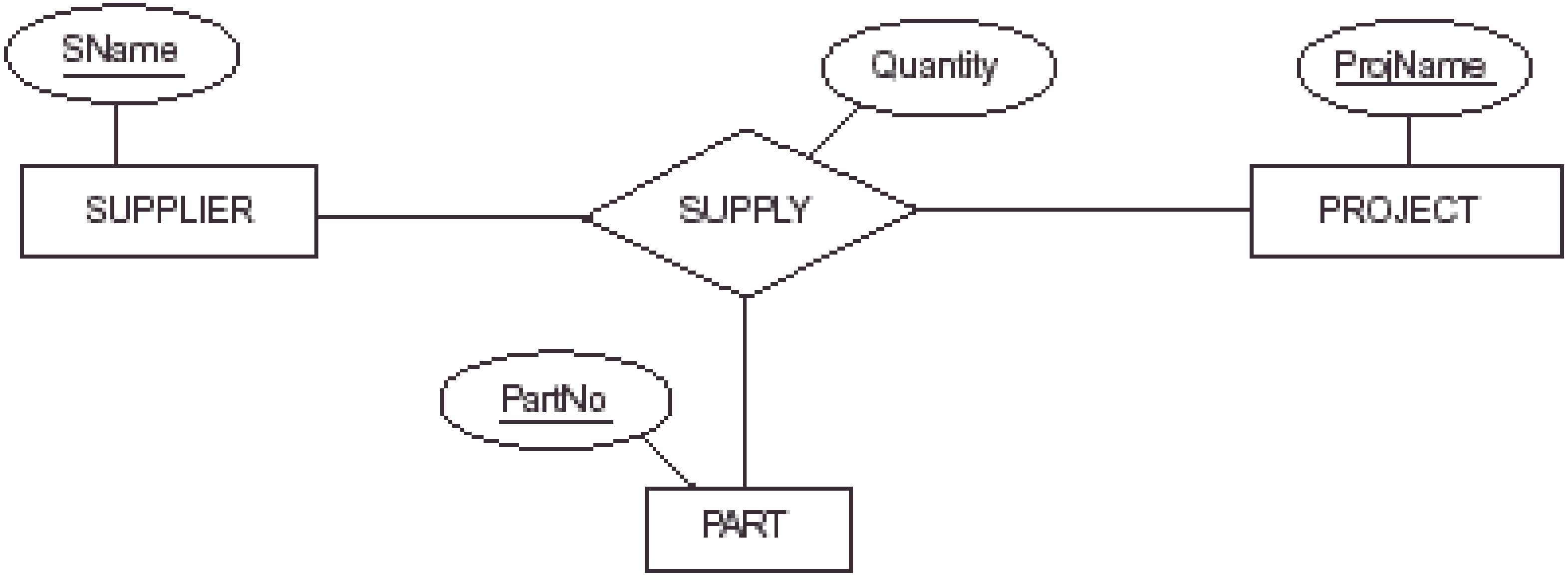
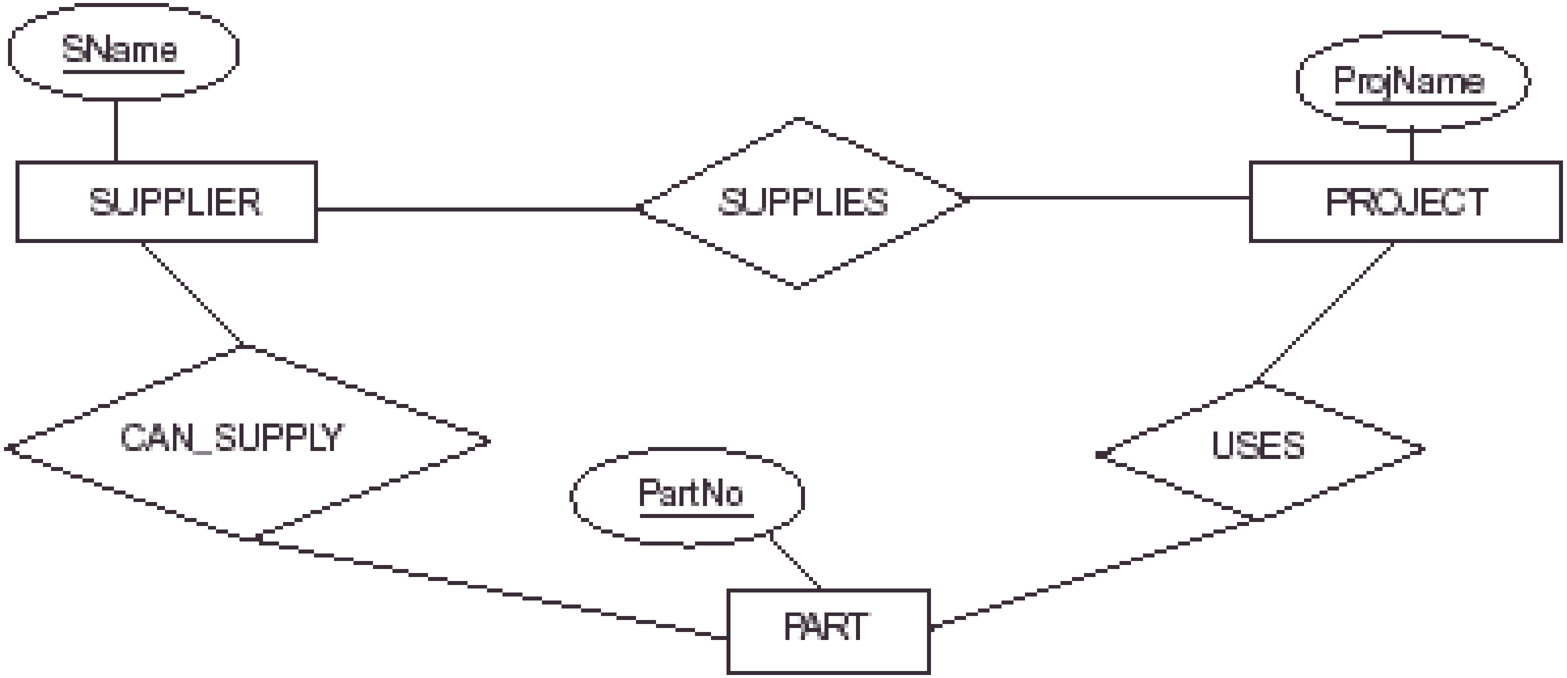
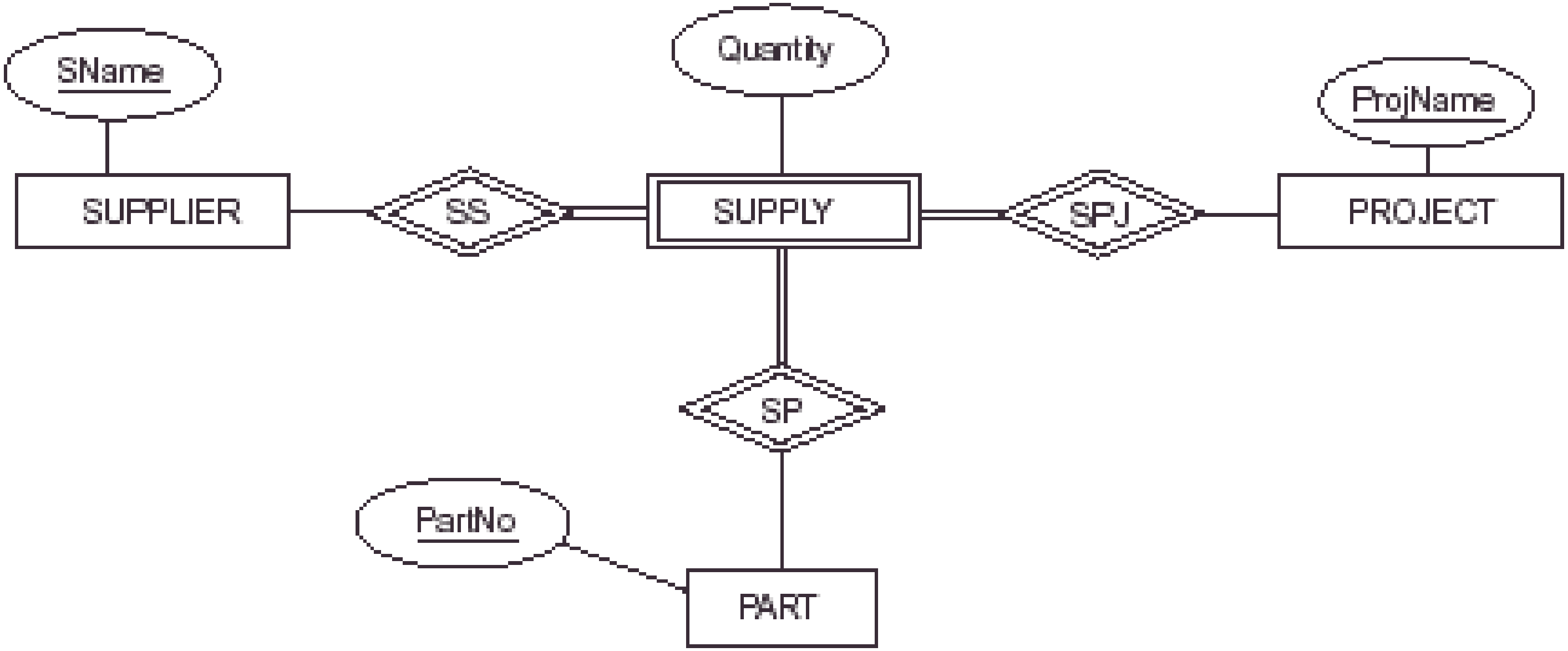
# Ch. 4 Enhanced E-R Model

- Ternary Relationships
- Sub-classes, Super-classes, Inheritance
- Specialization and Generalization
- Constraints on Specializations and Generalizations
- Aggregation and Association

# TERNARY RELATIONSHIPS




SName
SUPPLIER
SUPPLIES
ProjName
PROJECT
CAN_SUPPLY
PartNo
USES
PART


SName
Quantity
ProjName
SUPPLIER
SS
SUPPLY
SPJ
PROJECT
SP
PartNo
PART

# Subclasses and Superclasses

- Entity type may have sub-grouping that need to be represented explicitly.
  - Example: Employee may grouped into secretary, engineer, manager, technician, exempt and non-exempt.
  - Sub-groups are called subclass and employee superclass
  - relationship can be described as class/subclass
  - presenting member of subclass as distinct object (related via a key attribute of its superclass)
  - entity that is a member of subclass inherits all attributes of superclass
  - It also inherits all relationship that superclass participate in

# Specialization

- Top-down design process defines subgroupings within an entity type that are distinctive from other entities in the set.
  - Example: subclasses {secretary, engineer, etc..} is a specialization of superclass employee based on job type.
  - May have another specialization {exempt, non-exempt} based on method of pay
- These subclasses become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set.
- Attached by lines to a circle connected to superclass (for superclass that have 2 or more subclasses)

**Figure 4.1** EER diagram notation for representing specialization and subclasses.

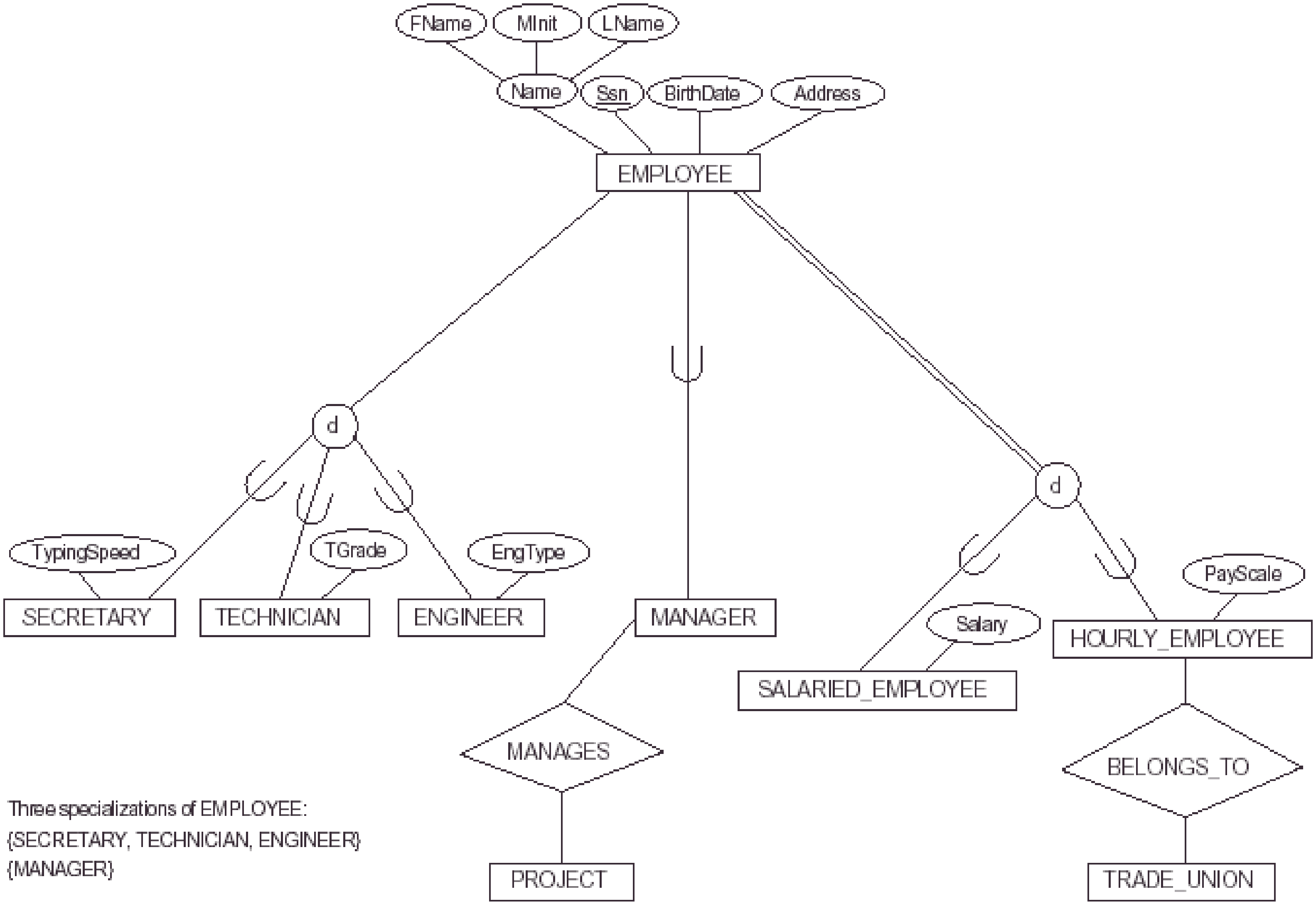
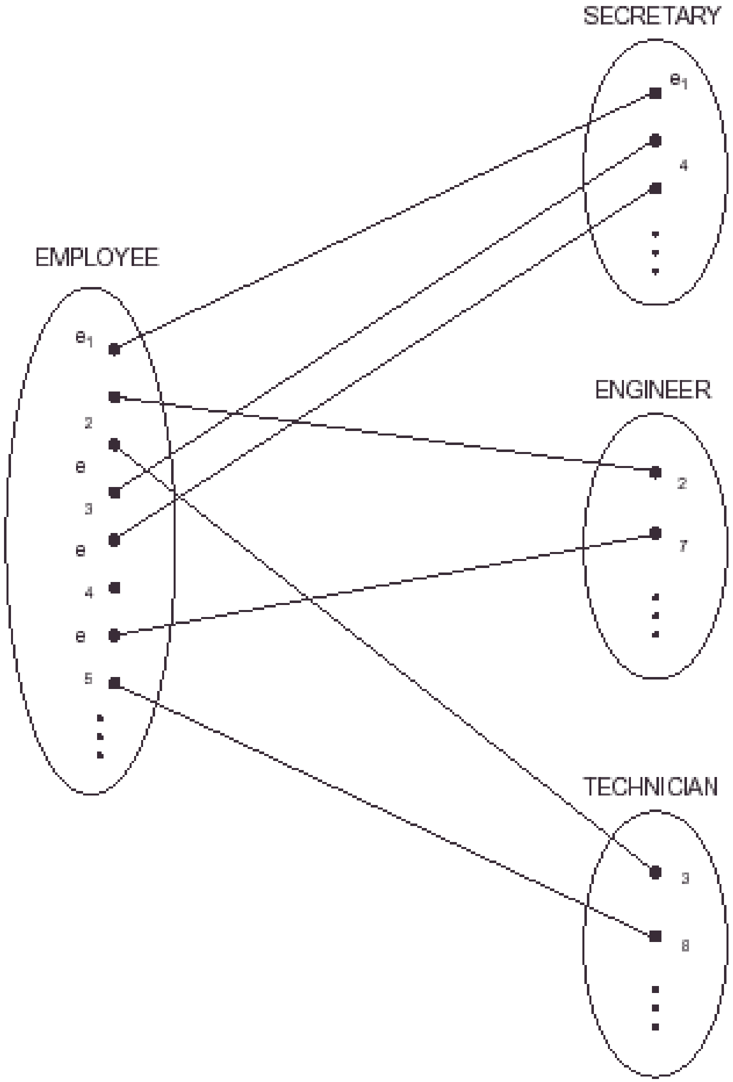


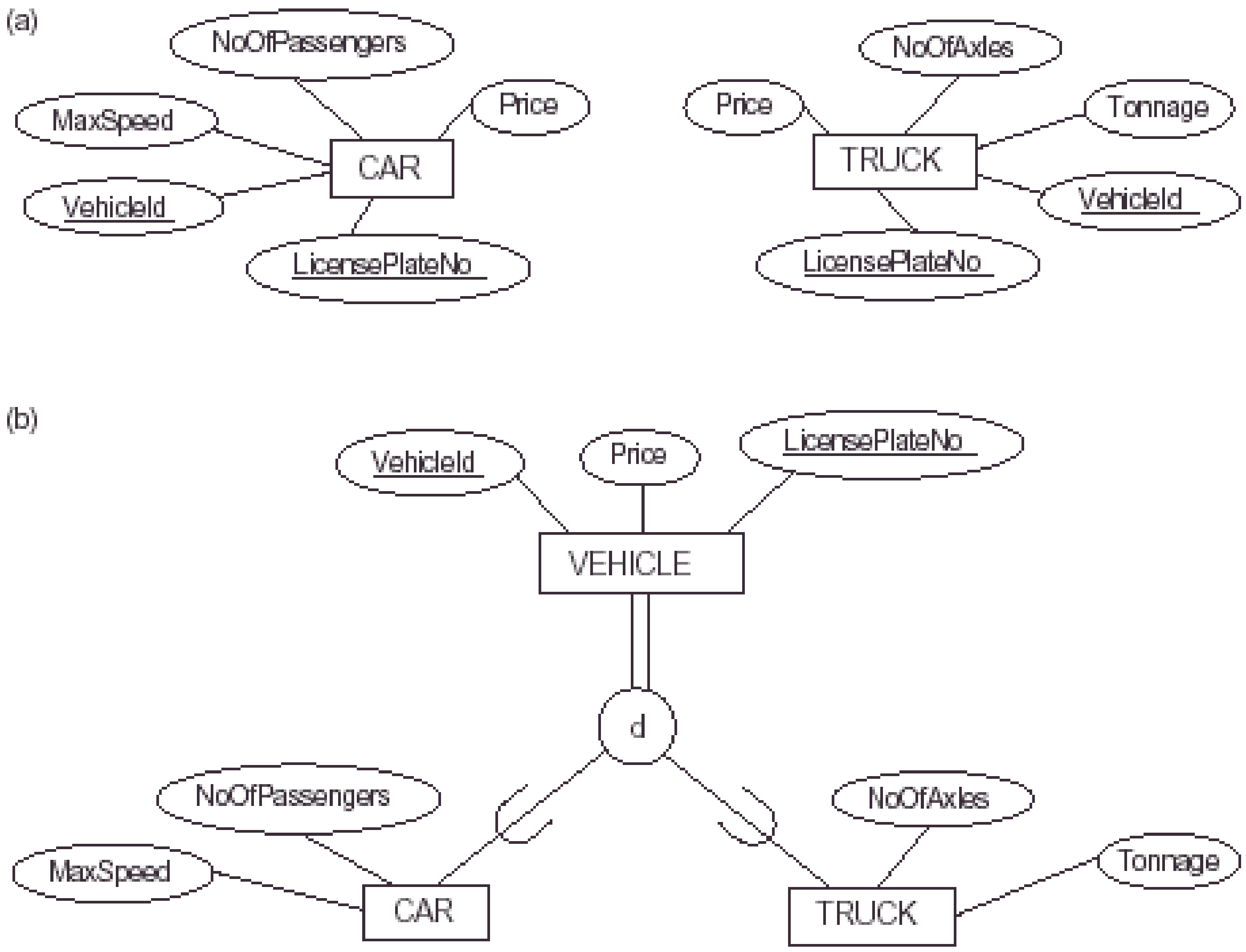
**Figure 4.2** Some instances of the specialization of EMPLOYEE into the {SECRETARY, ENGINEER, TECHNICIAN] set of subclasses.



# Generalization

- A bottom-up design process – combine a number of entity sets that share the same features into a higher-level entity set.
- Specialization and generalization are simple inversions of each other; they are represented in an E-R diagram in the same way.
- The terms specialization and generalization are used interchangeably.

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



# Design Constraints on Specialization/Generalization

- Constraint on which entities can be members of a given lower-level entity set.
  - Predicate/condition-defined (superclass has attribute specifying the condition of subclass membership)
  - User-defined (no condition specified)
- Constraint on whether or not entities may belong to more than one lower-level entity set within a single generalization.
  - Disjoint (entity can be member of at most one subclass in the specialization) denoted by **d** inside circle
  - overlapping ( subclasses are not constrained to be disjoint)

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

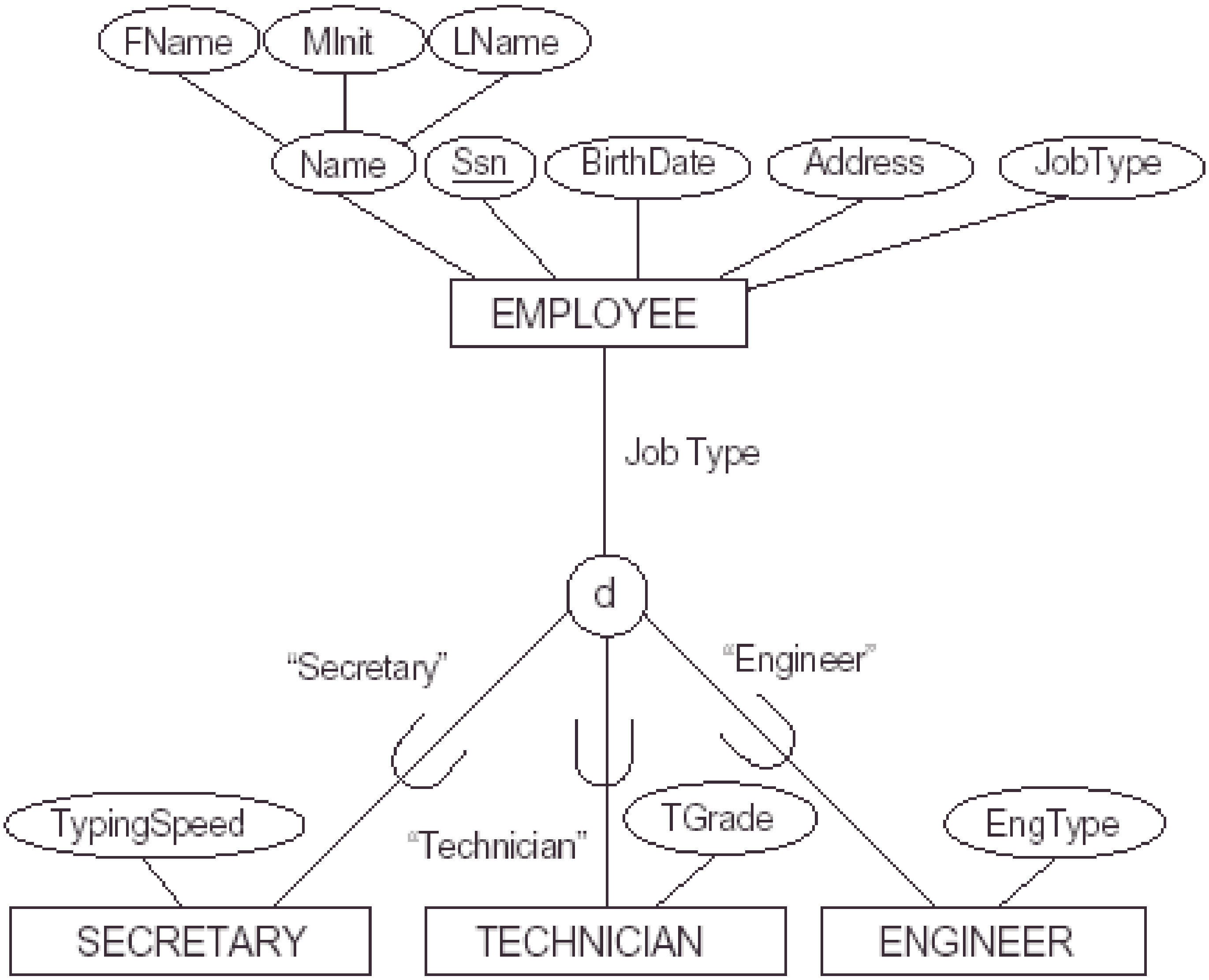
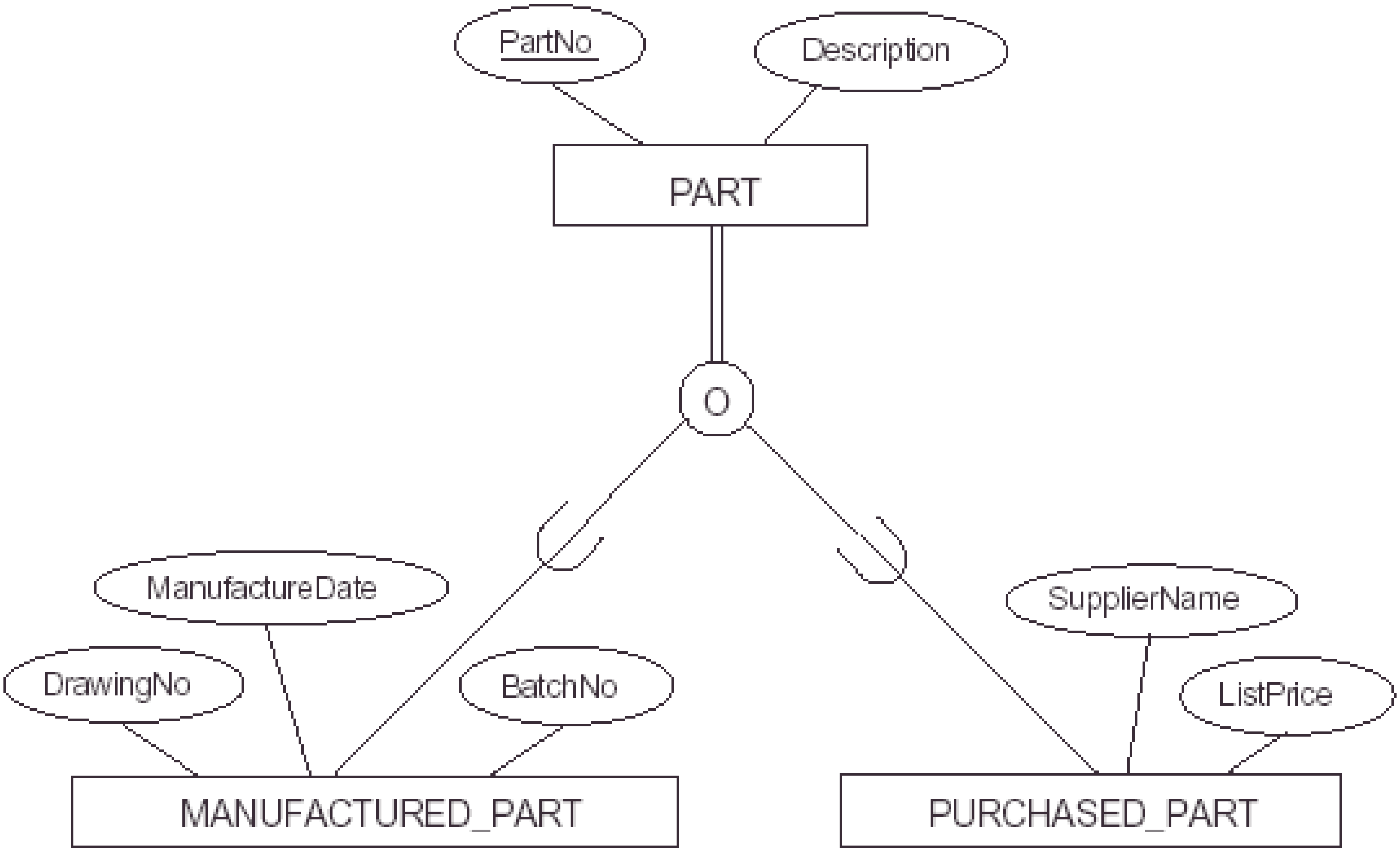
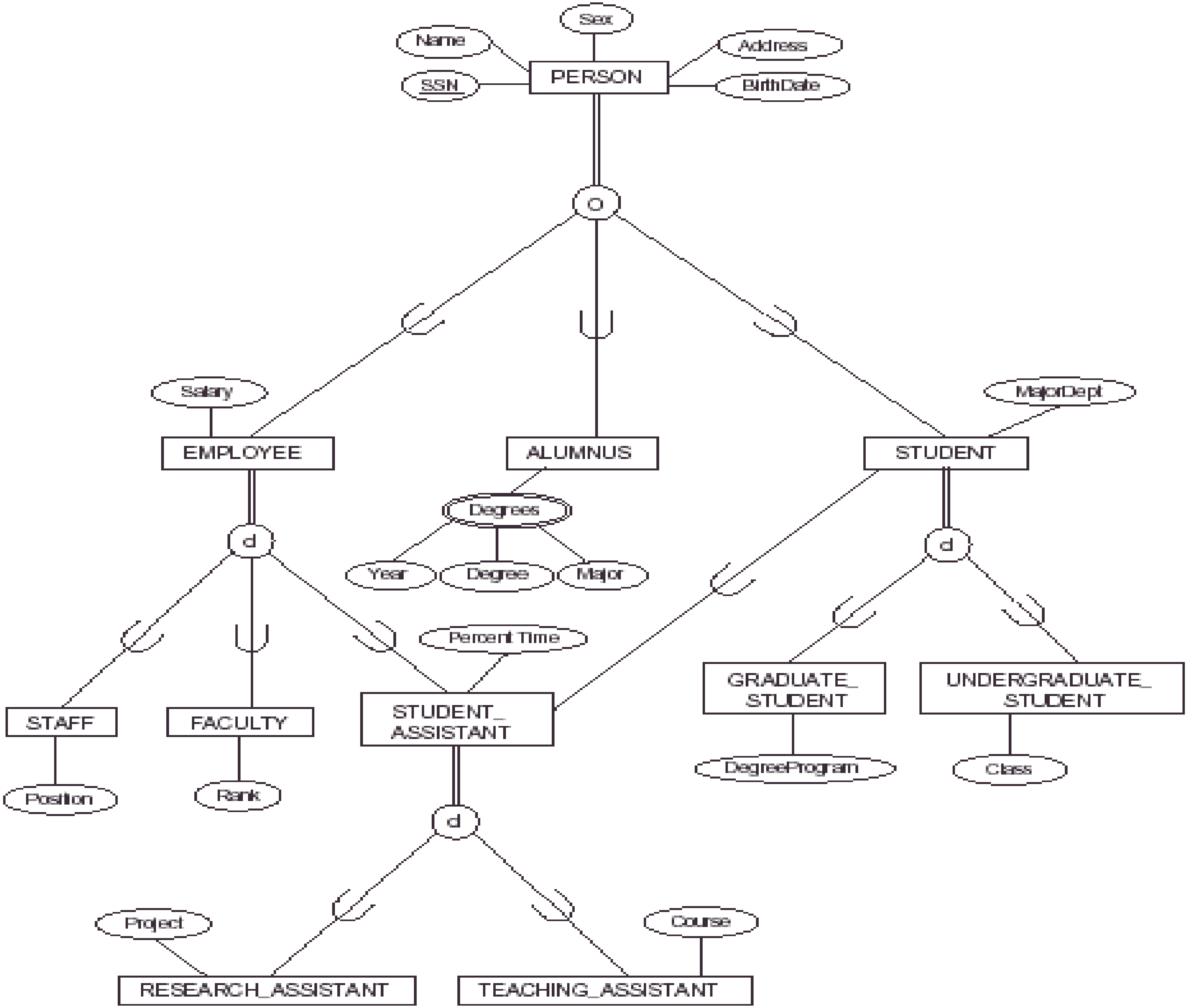


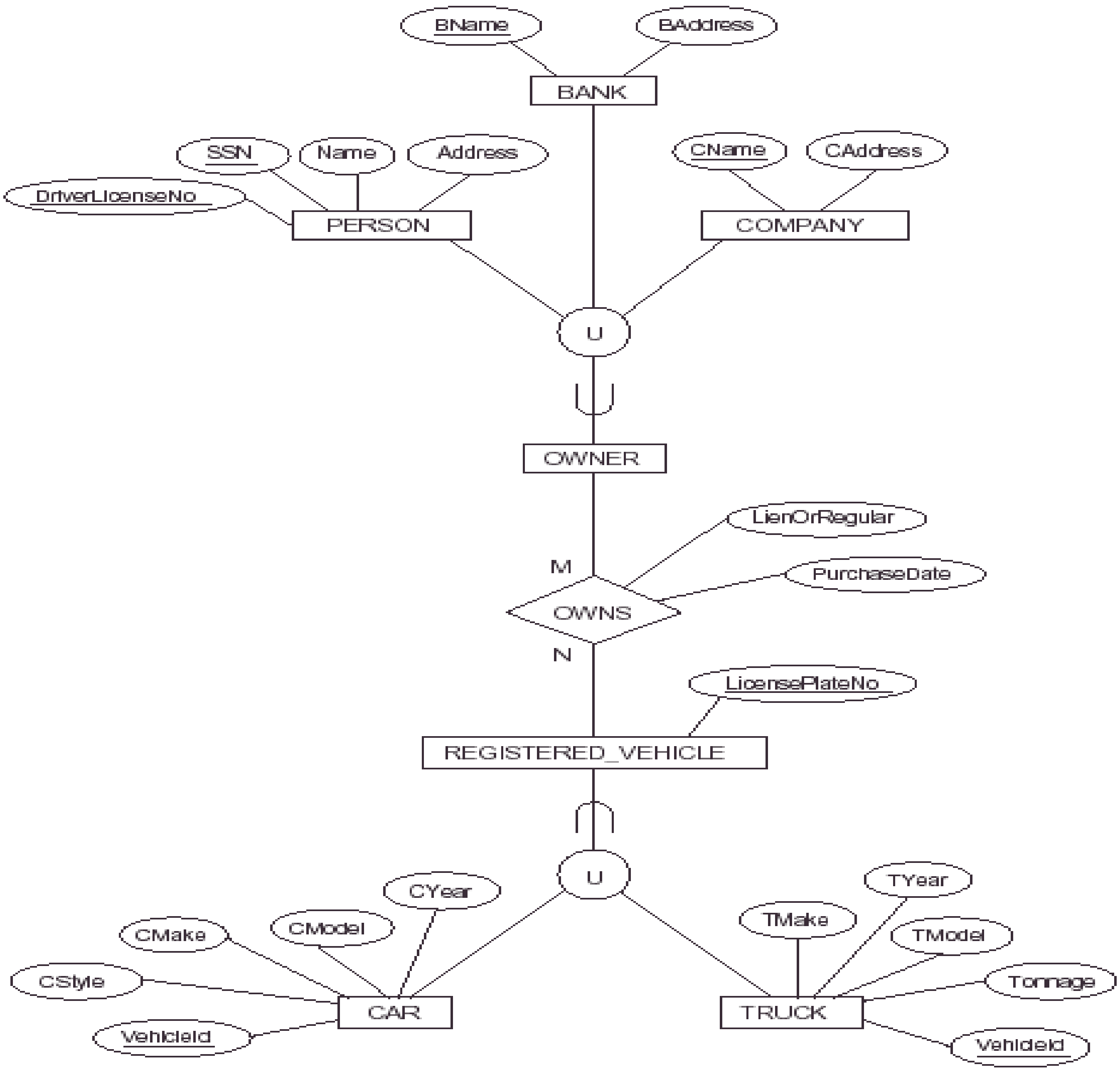
Figure 4.5 Notation for specialization with overlapping (nondisjoint) subclasses.

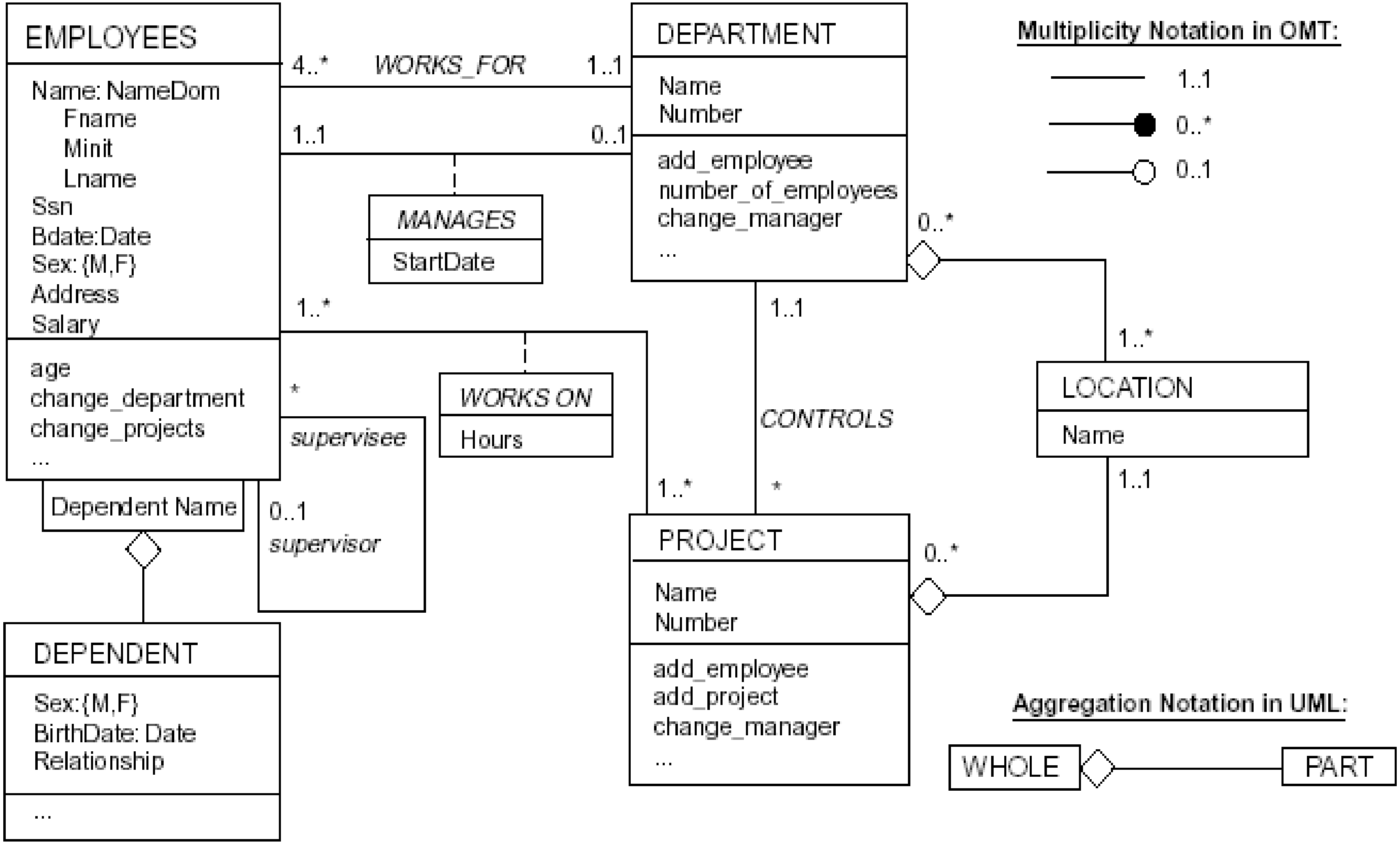


# Design Constraints on Specialization/Generalization

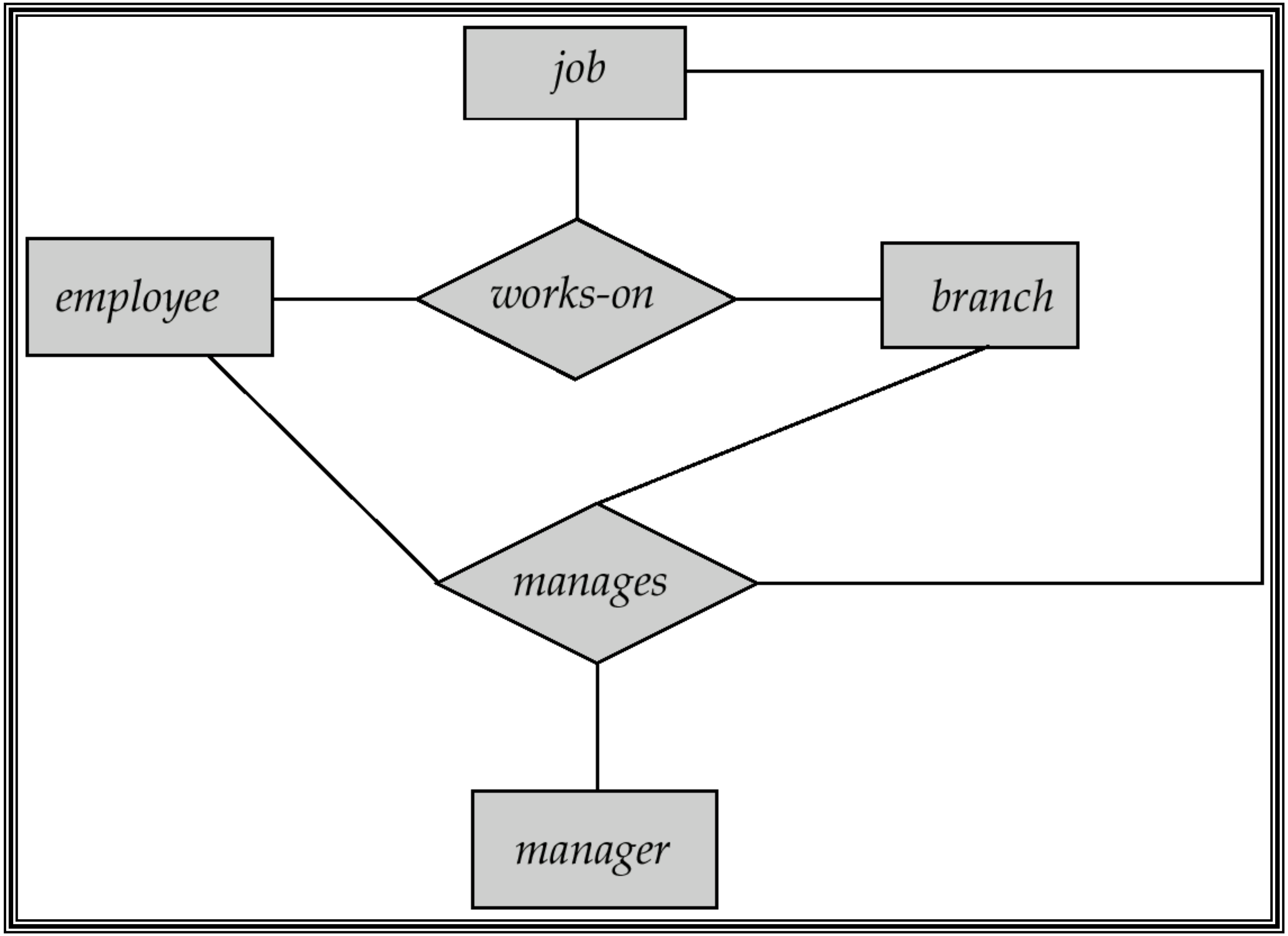
- Completeness constraint – specifies whether or not an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within a specialization.
  - Total (every entity in superclass is a member of some subclass in the specialization)
  - Total is defined by double lines connecting the circle to superclass
  - example: employee can be either exempt or non-exempt.
  - Partial (not every entity in superclass is a member of some subclass in the specialization)
  - defined by single line connecting the circle to superclass.
- Disjoint and Completeness are independent.


Sex
Name
Address
PERSON
SSN
BirthDate
O
Salary
MajorDept
EMPLOYEE
ALUMNUS
STUDENT
Degrees
d
d
Year
Degree
Major
Percent Time
STAFF
FACULTY
STUDENT_
ASSISTANT
GRADUATE_
STUDENT
UNDERGRADUATE_
STUDENT
Position
Rank
DegreeProgram
Class
d
Project
Course
RESEARCH_ASSISTANT
TEACHING_ASSISTANT


BName
BAddress
BANK
SSN
Name
Address
DriverLicenseNo
PERSON
CName
CAddress
COMPANY
U
OWNER
LienOrRegular
M
PurchaseDate
OWNS
N
LicensePlateNo
REGISTERED_VEHICLE
U
CYear
CMake
CModel
CStyle
CAR
VehicleId
TYear
TMake
TModel
Tonnage
TRUCK
VehicleId


EMPLOYEES
Name: NameDom
Fname
Minit
Lname
Ssn
Bdate:Date
Sex: {M,F}
Address
Salary
age
change_department
change_projects
...
Dependent Name
DEPENDENT
Sex:{M,F}
BirthDate: Date
Relationship
...
4..*
WORKS_FOR
1..1
1..1
0..1
MANAGES
StartDate
1..*
WORKS ON
Hours
*
supervisee
0..1
supervisor
DEPARTMENT
Name
Number
add_employee
number_of_employees
change_manager
...
0..*
1..1
CONTROLS
1..*
*
PROJECT
Name
Number
add_employee
add_project
change_manager
...
0..*
1..*
LOCATION
Name
1..1
Multiplicity Notation in OMT:
1..1
0..*
0..1
Aggregation Notation in UML:
WHOLE
PART

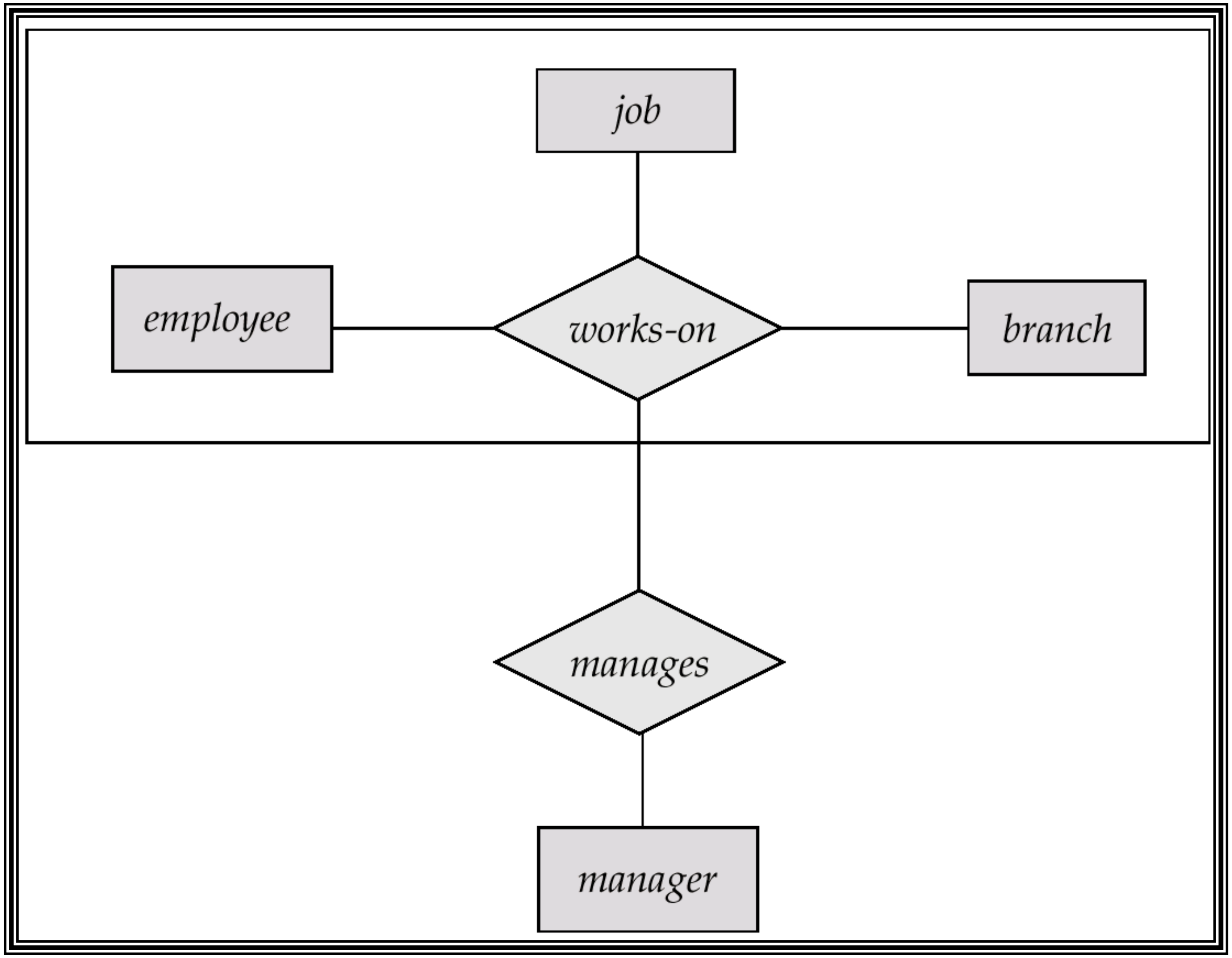
# E-R Diagram With Redundant Relationships



# Aggregation (Cont.)

- Relationship sets *works-on* and *manages* represent overlapping information
- 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 that:
  - An employee works on a particular job at a particular branch (and may work on different jobs at different branches)
  - An employee, branch, job combination may have an associated manager

# E-R Diagram With Aggregation



# E-R Design Decisions

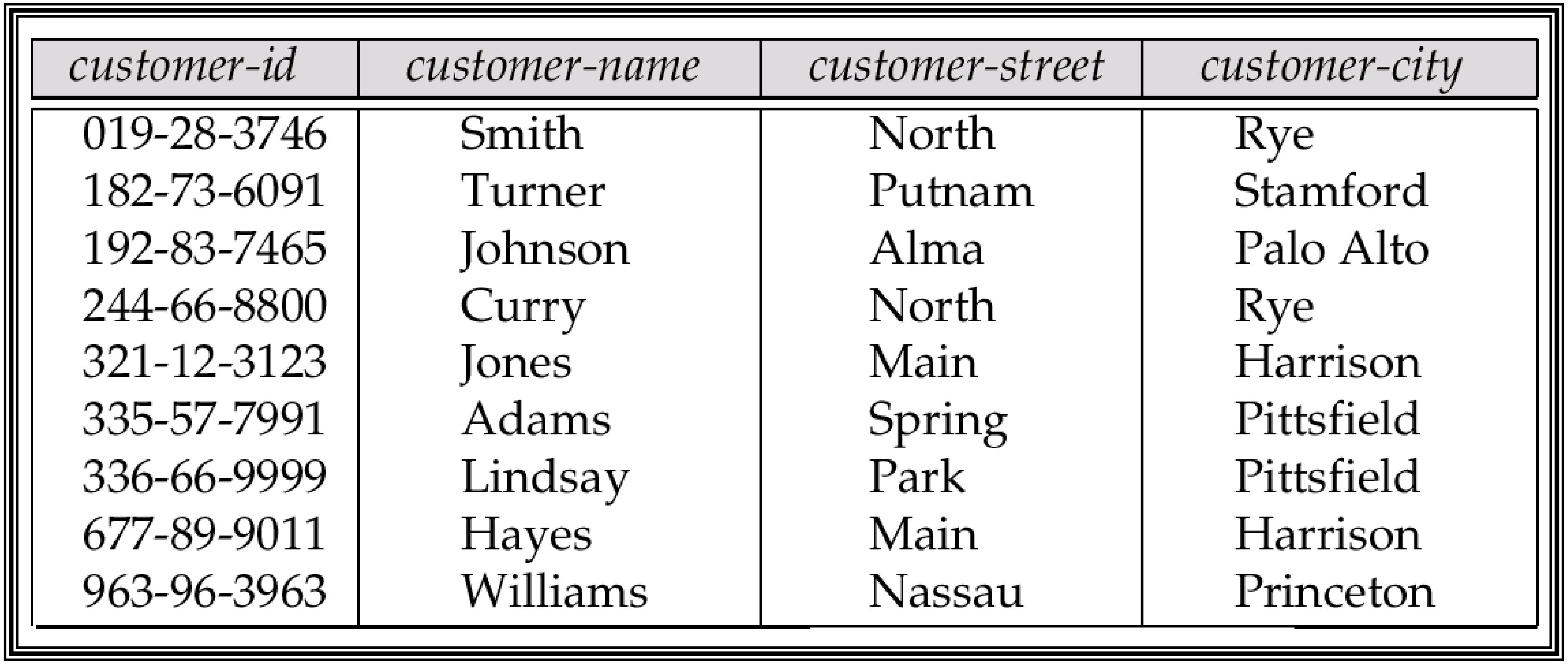
- The use of an attribute or entity type to represent an object.
- Whether a real-world concept is best expressed by an entity type or a relationship type.
- The use of a ternary relationship versus a pair of binary relationships.
- The use of a strong or weak entity type.
- The use of specialization/generalization – contributes to modularity in the design.
- The use of aggregation – can treat the aggregate entity type as a single unit without concern for the details of its internal structure.

# Reduction of an E-R Schema to Tables

- Primary keys allow entity types and relationship types to be expressed uniformly as *tables* which represent the contents of the database.
- A database which conforms to an E-R diagram can be represented by a collection of tables.
- For each entity type and relationship type there is a unique table which is assigned the name of the corresponding entity set or relationship set.
- Each table has a number of columns (generally corresponding to attributes), which have unique names.
- Converting an E-R diagram to a table format is the basis for deriving a relational database design from an E-R diagram.

# Representing Entity Sets as Tables

- A strong entity set reduces to a table with the same attributes.



| *customer-id* | *customer-name* | *customer-street* | *customer-city* |
|---|---|---|---|
| 019-28-3746 | Smith | North | Rye |
| 182-73-6091 | Turner | Putnam | Stamford |
| 192-83-7465 | Johnson | Alma | Palo Alto |
| 244-66-8800 | Curry | North | Rye |
| 321-12-3123 | Jones | Main | Harrison |
| 335-57-7991 | Adams | Spring | Pittsfield |
| 336-66-9999 | Lindsay | Park | Pittsfield |
| 677-89-9011 | Hayes | Main | Harrison |
| 963-96-3963 | Williams | Nassau | Princeton |

# Composite and Multivalued Attributes

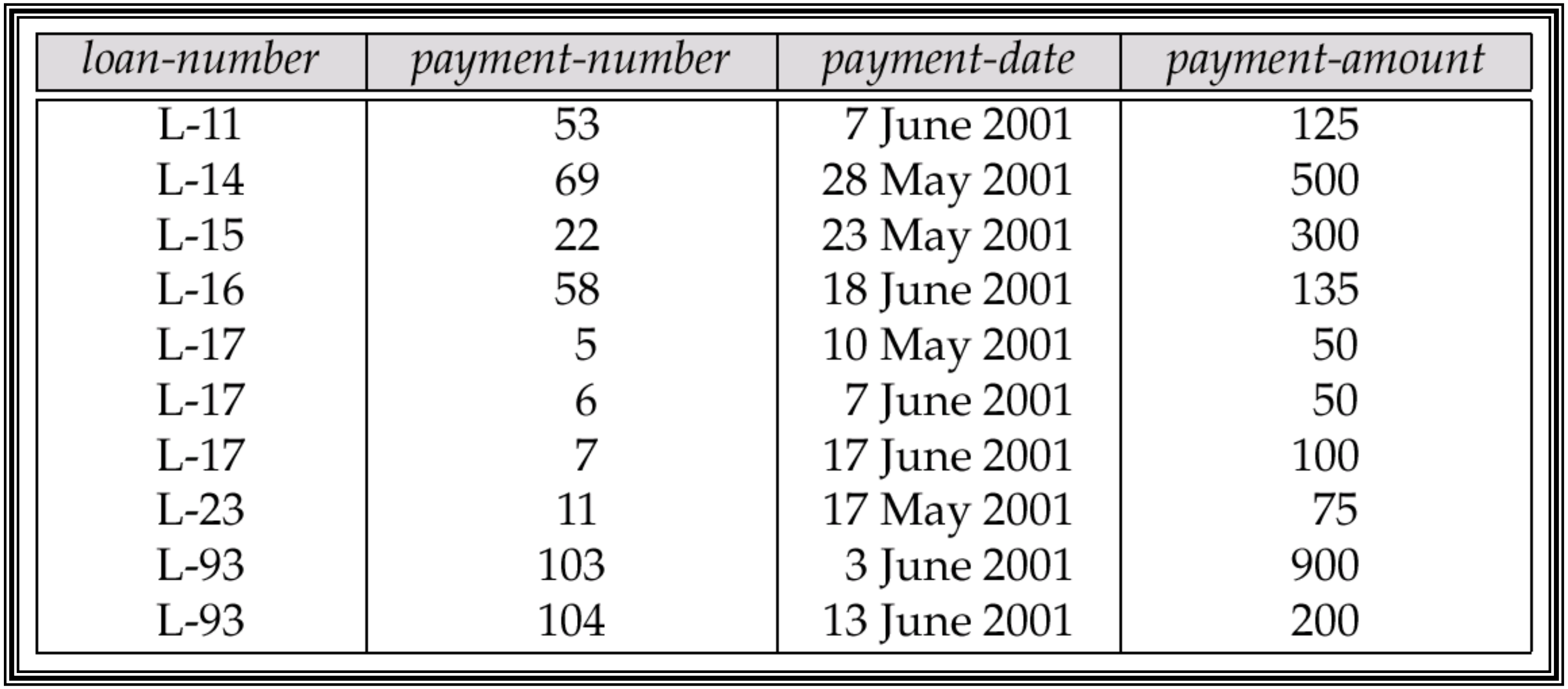
- Composite attributes are flattened out by creating a separate attribute for each component attribute
  - E.g. given entity set *custome*r with composite attribute *name* with component attributes *first-name* and *last-name* the table corresponding to the entity set has two attributes

    *name.first-name* and *name.last-name*
- A multivalued attribute M of an entity E is represented by a separate table EM
  - Table EM has attributes corresponding to the primary key of E and an attribute corresponding to multivalued attribute M
  - E.g. Multivalued attribute *dependent-names* of *employee* is represented by a table

    *employee-dependent-names*( *employee-id, dname*)
  - Each value of the multivalued attribute maps to a separate row of the table EM
    - E.g., an entity with primary key John and dependents Johnson and Peter maps to two rows: (John, Johnson) and (John, Peter)

# Representing Weak Entity Sets

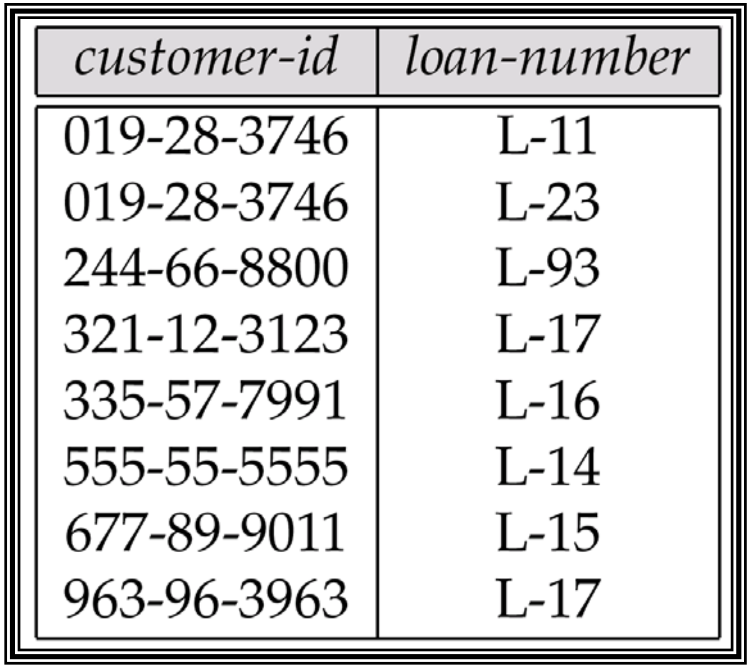
- A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set



| *loan-number* | *payment-number* | *payment-date* | *payment-amount* |
|---|---|---|---|
| L-11 | 53 | 7 June 2001 | 125 |
| L-14 | 69 | 28 May 2001 | 500 |
| L-15 | 22 | 23 May 2001 | 300 |
| L-16 | 58 | 18 June 2001 | 135 |
| L-17 | 5 | 10 May 2001 | 50 |
| L-17 | 6 | 7 June 2001 | 50 |
| L-17 | 7 | 17 June 2001 | 100 |
| L-23 | 11 | 17 May 2001 | 75 |
| L-93 | 103 | 3 June 2001 | 900 |
| L-93 | 104 | 13 June 2001 | 200 |

# Representing Relationship Sets as Tables

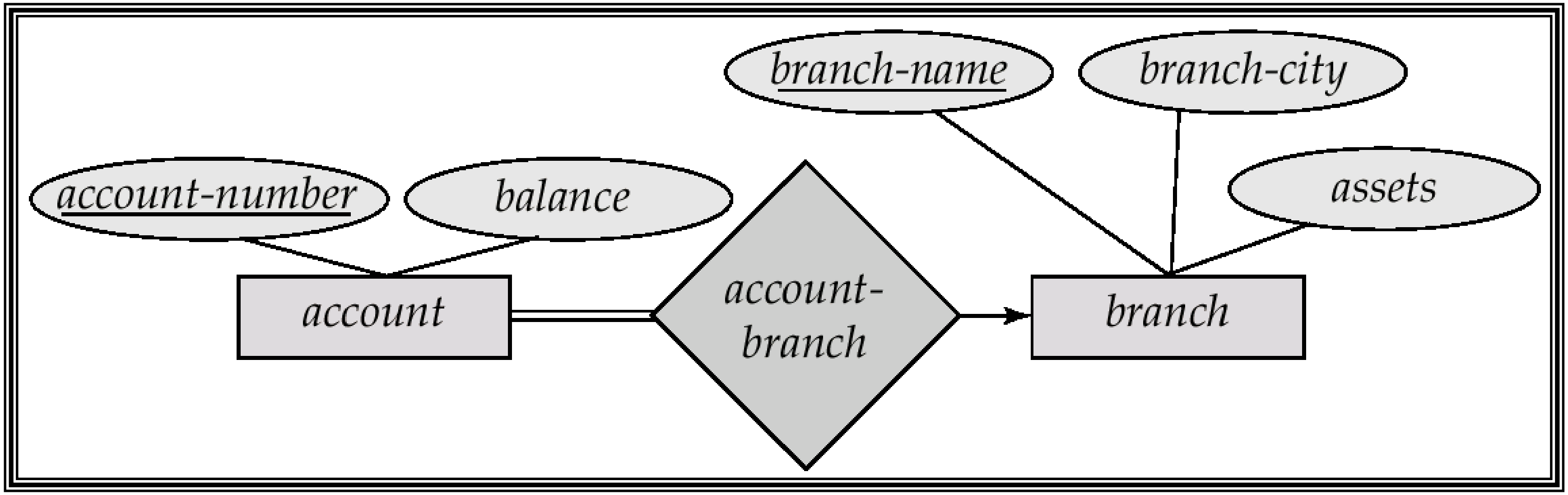
- A many-to-many relationship set is represented as a table with columns for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.
- E.g.: table for relationship set *borrower*



| *customer-id* | *loan-number* |
|---|---|
| 019-28-3746 | L-11 |
| 019-28-3746 | L-23 |
| 244-66-8800 | L-93 |
| 321-12-3123 | L-17 |
| 335-57-7991 | L-16 |
| 555-55-5555 | L-14 |
| 677-89-9011 | L-15 |
| 963-96-3963 | L-17 |

# Redundancy of Tables

- Many-to-one and one-to-many relationship sets that are total on the many-side can be represented by adding an extra attribute to the many side, containing the primary key of the one side
- E.g.: Instead of creating a table for relationship *account-branch*, add an attribute *branch* to the entity set *account*



# Redundancy of Tables (Cont.)

- For one-to-one relationship sets, either side can be chosen to act as the “many” side
  - That is, extra attribute can be added to either of the tables
- If participation is *partial* on the many side, replacing a table by an extra attribute in the relation corresponding to the “many” side could result in null values
- The table corresponding to a relationship set linking a weak entity set to its identifying strong entity set is redundant.
  - E.g. The *payment* table already contains the information that would appear in the *loan-payment* table (i.e., the columns loan-number and *payment-number*).

# Representing Specialization as Tables

- Form a table for the higher level entity
- Form a table for each lower level entity set, include primary key of higher level entity set and local attributes

| table | table attributes |
|---|---|
| *person* | *<u>name</u>, street, city* |
| *customer* | *name, credit-rating* |
| *employee* | *name, salary* |

Drawback: getting information about, e.g., *employee* requires accessing two tables

Form a table for each entity set with all local and inherited attributes

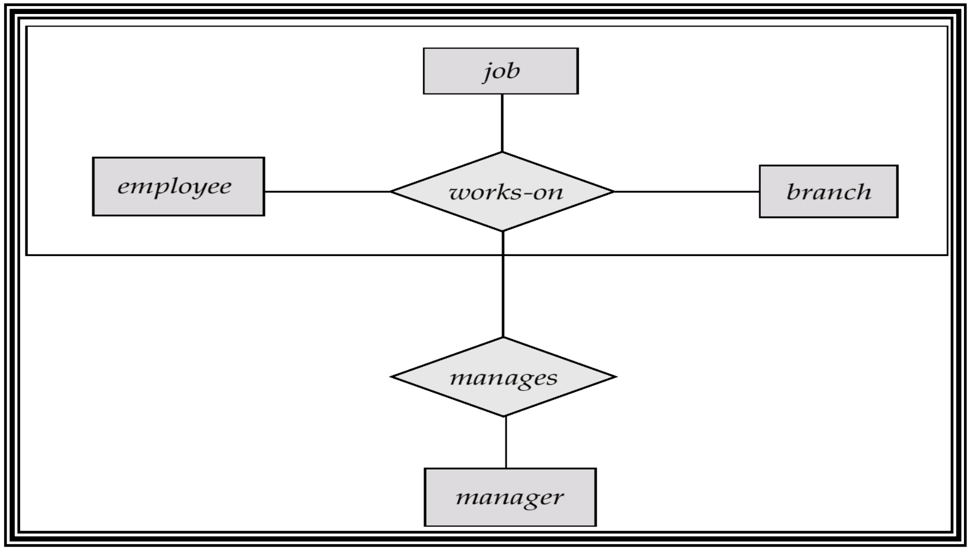
| table | table attributes |
|---|---|
| *person* | *name, street, city* |
| *customer* | *name, street, city, credit-rating* |
| *employee* | *name, street, city, salary* |

If specialization is total, no need to create table for generalized entity

Drawback: street and city may be stored redundantly for persons who are both customers and employees

# Relations Corresponding to Aggregation

- To represent aggregation, create a table containing primary key of the aggregated relationship and the primary key of the associated entity set
  - E.g. to represent aggregation *manages* between relationship *works-on* and entity set *manager*, create a table
    *manages*(*employee-id, branch-name, title, manager-name*)
  - Table *works-on* is redundant **provided** we are willing to store null values for attribute *manager-name* in table *manages*


job
employee
works-on
branch
manages
manager