# CS157A Lecture 5

# Enhanced ER-diagram

Prof. Sin-Min Lee
Department of Computer Science

# Database Modeling and Implementation Process

$$\begin{tikzcd}
\text{Ideas} \arrow[r] & \text{ER Design} \arrow[r] & \text{Relational Schema} \arrow[d] \\
 & & \text{Relational DBMS Implementation}
\end{tikzcd}$$

## Entity Types

- All similar (same attributes) entities are grouped into sets, an *entity type*
- Entity type schema specifies the common structure:
  - type name
  - entity attributes (Domain, value set)
  - constraints on entities
- E.g.,

  FACULTY: Name(FN,LM,MI), DoB, SSN, {Degree},Rank
  - FN:String(15), LN: String(15), SSN: String(9), etc.
  - BoD: DD/MM/YYYY
  - Degree: {BS,MS,PhD}
  - Rank: {Lecturer, Assistant, Associate, Full}

- Example:  A library database contains a listing of authors that have written books on various subjects (one author per book).  It also contains information about libraries that carry books on various subjects.

# RELATIONSHIPS (Cont…)

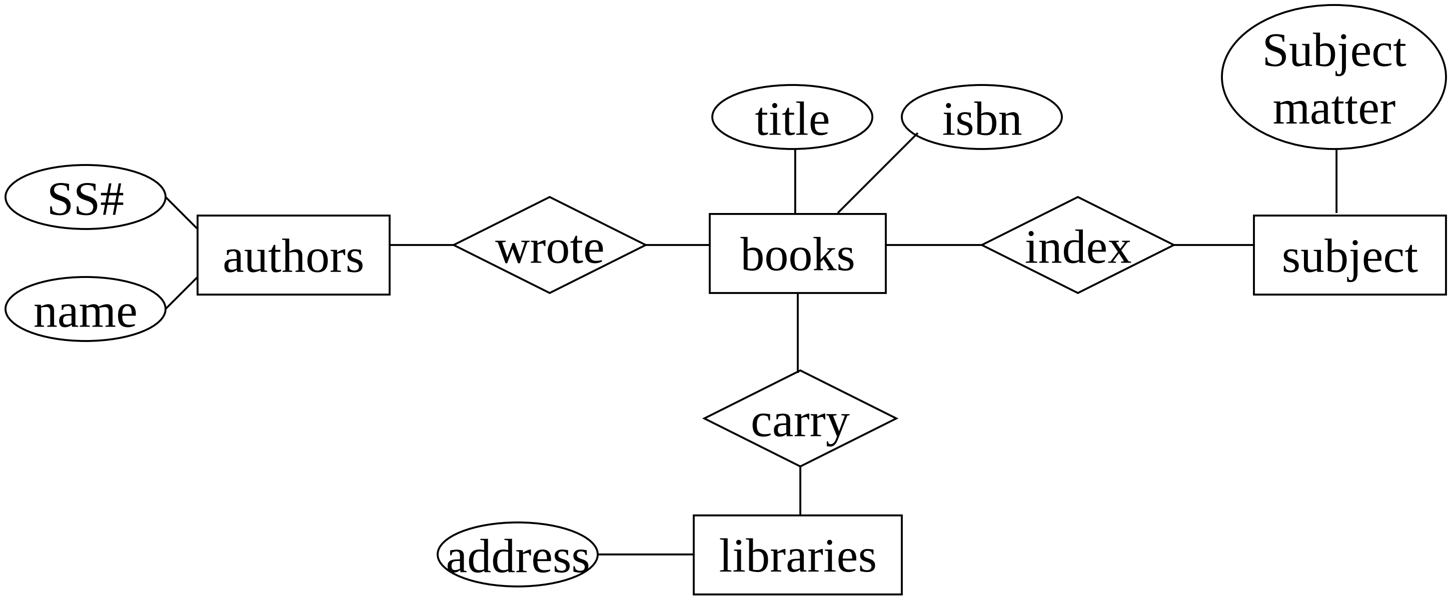
- Example: A library database contains a listing of authors that have written books on various subjects (one author per book). It also contains information about libraries that carry books on various subjects.

  Entity sets: authors, subjects, books, libraries

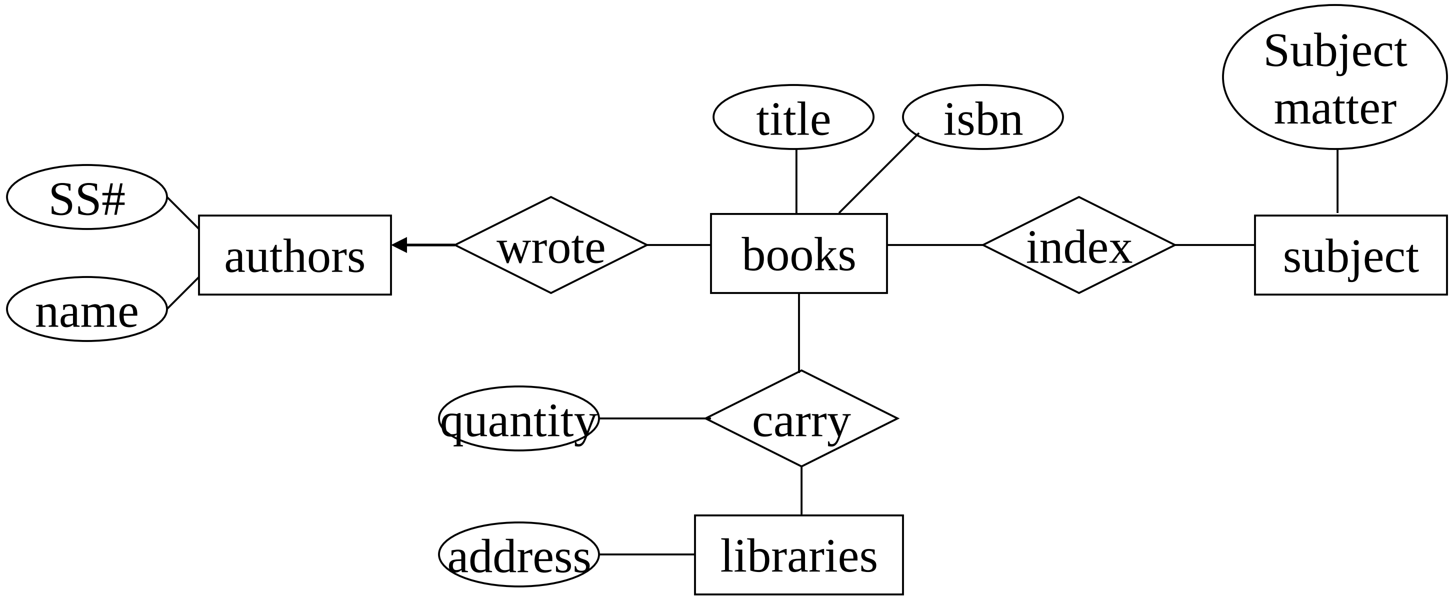
# RELATIONSHIPS (Cont…)

- Example: A library database contains a listing of authors that have written books on various subjects (one author per book). It also contains information about libraries that carry books on various subjects.
  - Entity sets: authors, subjects, books, libraries
  - Relationship sets: wrote, carry, indexed

# RELATIONSHIPS (Cont…)



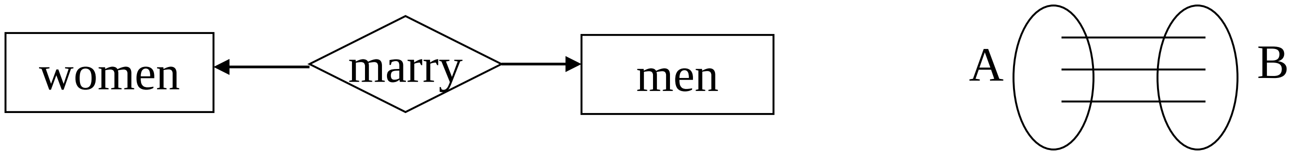
# RELATIONSHIPS (Cont…)



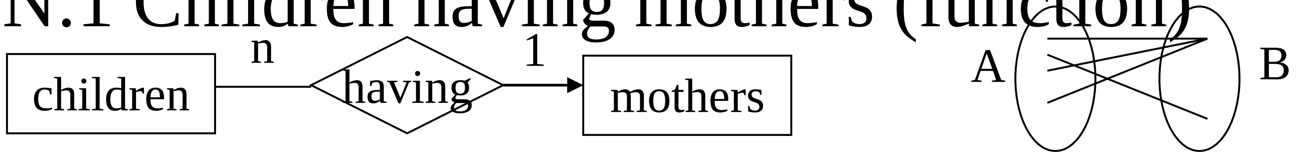
# BINARY RELATIONSHIP

A binary relationship between entity set *A* and *B* might be:

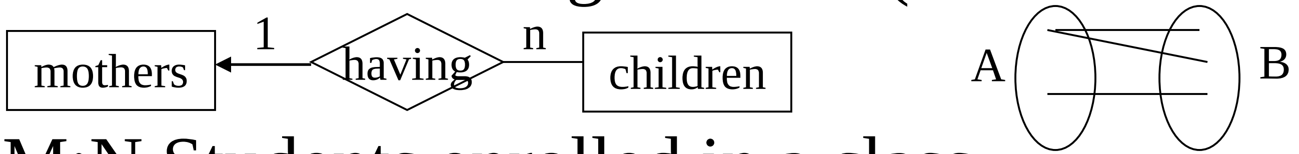
- 1:1 Women marrying Men (function)



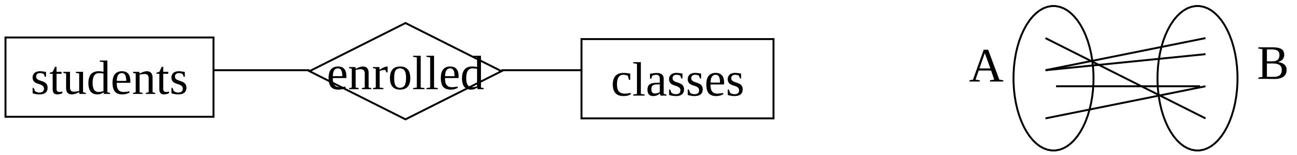
- N:1 Children having mothers (function)



- 1:N Mothers having children (inverse function)



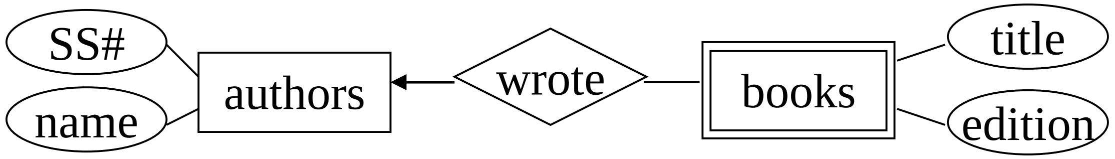
- M:N Students enrolled in a class



# KEY

- Entities and relationships are distinguishable using various keys
- A ***key*** is a combination of one or more attributes, e.g., social-security number, combination of name and social-security number.
- A ***superkey*** is a key defined either for an entity set or relationship set that uniquely identifies an entity, e.g., social-security number, phone number, combination of name and social-security number.
- A ***candidate key*** is a minimal superkey that uniquely identifies either an entity or a relationship, e.g., social-security number, phone number.
- A ***primary key*** is a candidate key that is chosen by the database designer to identify the entities of an entity set.

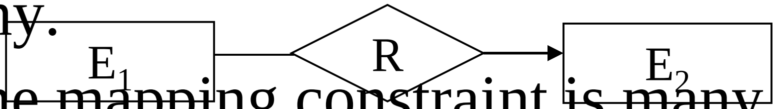
- A ***foreign key*** is a set of one or more attributes of a strong entity set that are employed to construct the discriminator of a weak entity set. The primary key of a weak entity set is formed by the primary key of the strong entity set on which it is existence-dependent.



- Relationship sets also have primary keys. Assume *R* is a relationship set involving entity sets $E_1, E_2, \ldots, E_n$. Let primary-key($E_i$) denote the primary key for entity set $E_i$. Assume primary-key($E_i$) is unique for $1 \leq i \leq n$. If *R* has no attributes then its superkey is:

  $primary\text{-}key(E_1) \cup primary\text{-}key(E_2) \cup \ldots \cup primary\text{-}key(E_n)$

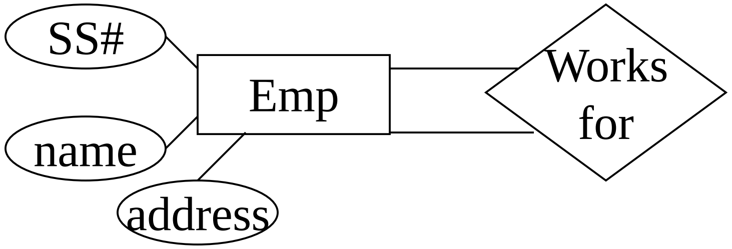
- This is a primary key if the mapping constraint is many-to-many.



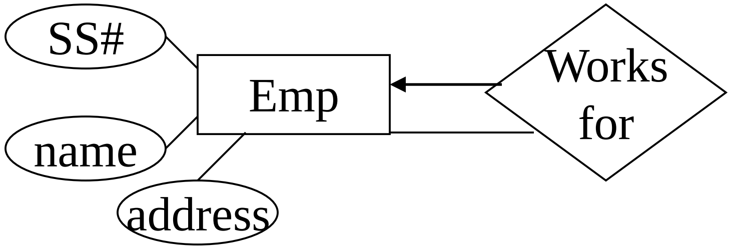
- If the mapping constraint is many to one from $E_1$ to $E_2$ then the primary key of *R* is primary key of $E_1$.

## EXAMPLE

- Employees of a large company, e.g., IBM, where an employee reports to a manager. The manager is also an employee who reports to another manager. This chain of command continues to the very top where the CEO is the only employee who is not reporting to a manager. Draw the ER diagram for this example.

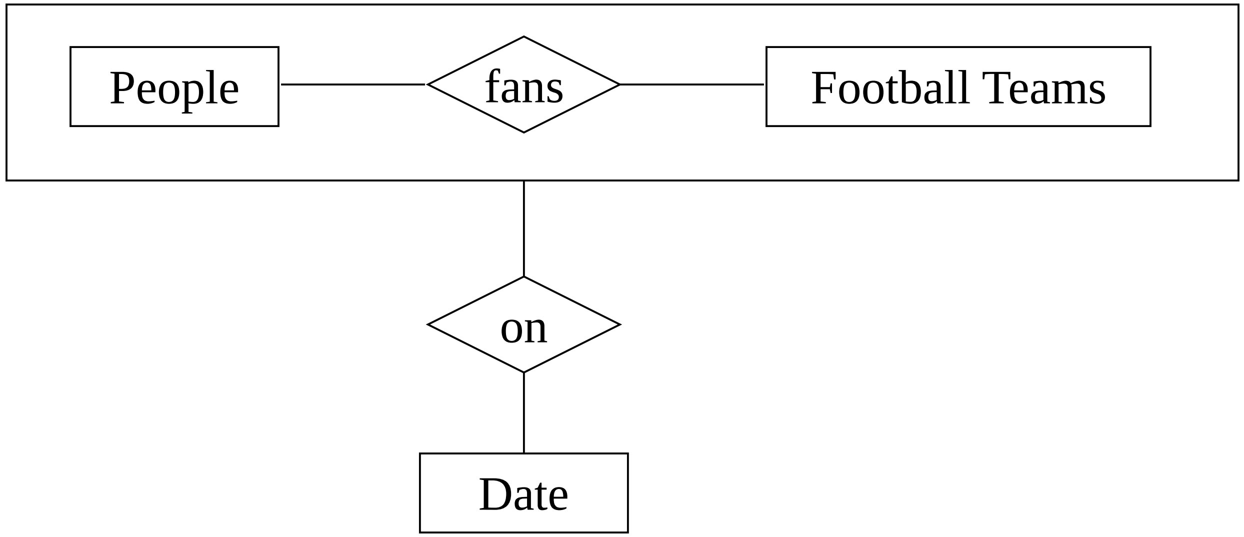


Primary keys:
Emp: SS#
Works-for: (empSS#, mgrSS#)



Primary keys:
Emp: SS#
Works-for: (empSS#)

- A relationship may involve *n* entities, N-ary relationship
- It is always possible to replace a non-binary relationship set by a number of distinct binary relationship sets



## Uniqueness or Key Constraint

- Entities are distinguished by using various *keys*
- A key is a uniqueness constraint on attributes
- A Key is defined over one or more attributes
  - SSN, StudentID, Car License Plate: State and Number
- **Superkey**: Any combination of attributes that uniquely identifies an entity
  - Name and SSN, Name and StudentID,
- **Candidate Key** is a minimal superkey
  - E.g., SSN and StudentID
- **Primary Key** is one of the candidate keys (<u>SSN</u>)
- **Alternative keys** are the remaining candidate keys
  - *Primary key is underlined, alternative are over-lined*

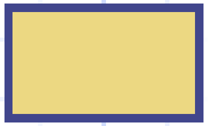
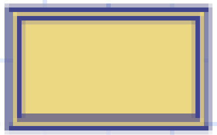
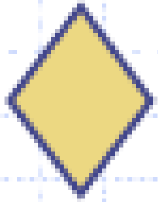
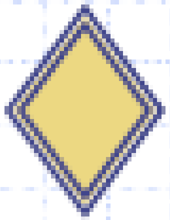
## Relationship Types

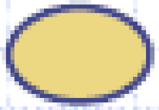
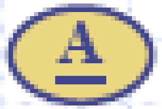
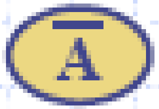
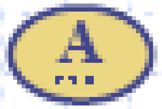
- ***Relationship Types***: sets of relationships that are homogeneous in participating entities
  - BELONG:<FACULTY, DEPARTMENT>
  - ENROLLS:<STUDENT, SECTION>
- ***Degree*** of a relationship is the number of participating entity types:
  - 2-entities → binary relationship
  - 3-entities → tenary relationship
  - …
  - N-entities → N-ary relationship
- Recursive relationships that involve more than once the same entity type with different **Roles**:
  - SUPERVISES:<supervisor-faculty, supervisee-faculty>

## Strong and Weak Entities

- Strong or ordinary Entities:
  - Have independent existence in the mini-world
  - They are part of the care of the application
- Weak Entities:
  - They are dependent on another entity
  - *Identify owner* is the specific entity on which the weak entity depends
  - No key attribute; are distinguishable through an *identifying relationship* and a *discriminator* or *partial key*
  - Identifying relationship is always total participation
  - It may be represented as multi-value, composite attribute of owner (When isn't this possible?)

## ER-Diagrams

| | |
|---|---|
|  | (strong) entity set |
|  | weak entity set |
|  | relationship set |
|  | identifying relationship set for weak entity |

| | |
|---|---|
|  | attribute |
|  | primary key |
|  | alternative key |
|  | Discriminator partial key |

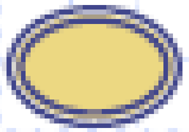
## Observation

- nouns -> entity types/sets
- verbs -> relationship types

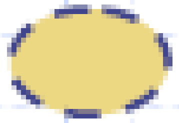
## Constraints on Relationship Types

- Cardinality ration: Specifies the number of relationship instances that an entity can participate in.
  - 1:1 Departments having Chairpersons
  - N:1 Children having Mothers
  - 1:N Mothers having children (inverse of N:1)
  - M:N Students enrolling in Class Sections
- Participation:
  - *Total* → Existence of entity depends on the existence of a related entity. E.g., Classes have total participation to OFFER_BY dept.
  - *Partial* → Some entities are not related to other entities. E.g., Faculty have partial participation to CHAIR of a dept.

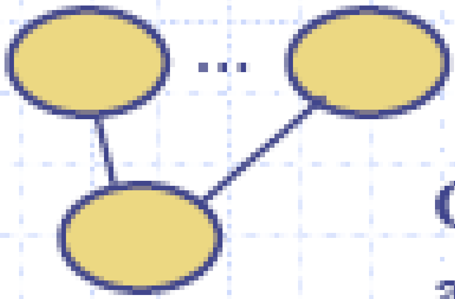
# ER Diagrams…



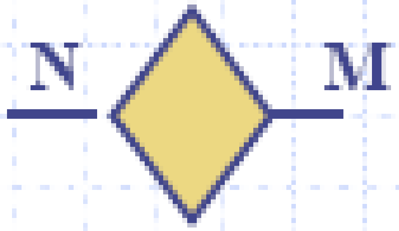
**Multivalued attribute**



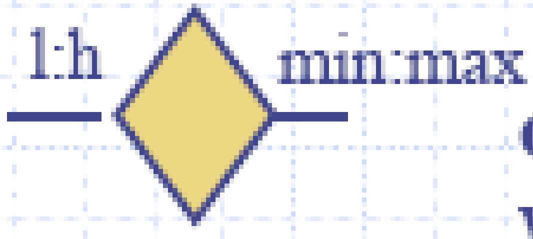
**Derived attribute**



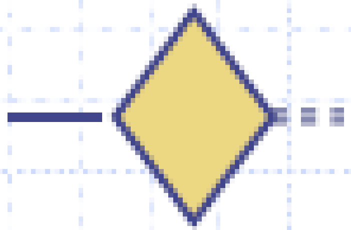
**Composite attribute**



**Cardinalities**



**Cardinalities with limits**

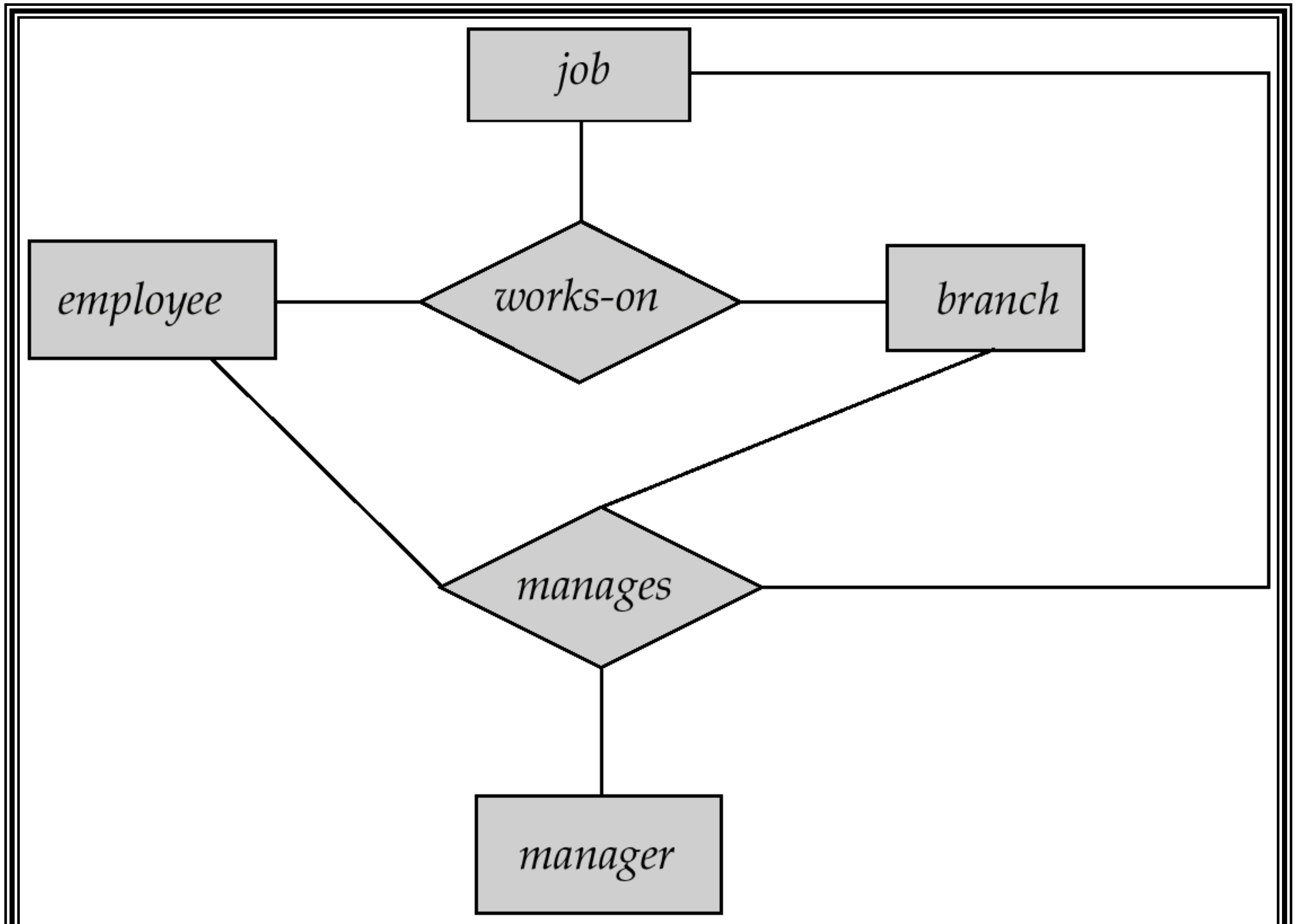


**Participation partial/total**

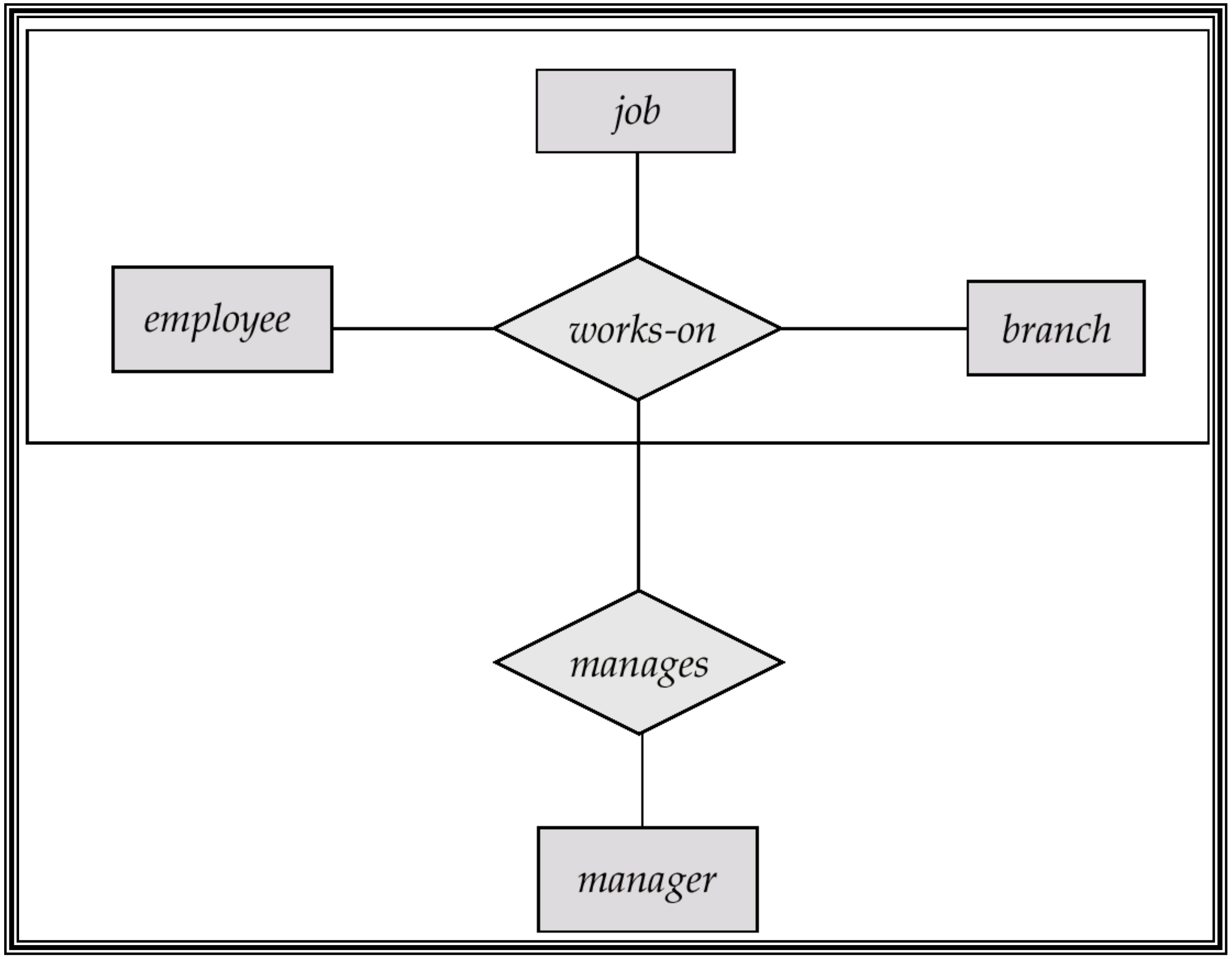
(dash line should be double line)

# Aggregation

- Consider the ternary relationship *works-on*, which we saw earlier
- Suppose we want to record managers for tasks performed by an employee at a branch



## E-R Diagram With Aggregation



## Case Study: Library Database System

- Library organized into sections, like art, children, computing, science, etc. Each section has name and a number and its headed by a head librarian
- Each book title belongs to a section and has a title, authors, ISBN, call number, year and publisher
- For each copy of the book keep track the current borrower, the due date and the librarian who charged it out.
- Members have membership number, a driver's license, an address, a phone number and birthday
- Members can have up to 5 borrowed books and can put a hold request on a book.
- Librarians have a name, ssn, address, phone

- Example:  A library database contains a listing of authors that have written books on various subjects (one author per book).  It also contains information about libraries that carry books on various subjects.
  - Entity sets: authors, subjects, books, libraries
  - Relationship sets: wrote, carry, indexed
  - E-R diagram:

## Entities

1. TITLE: CallNumber, Name, Author{(Name(Fname, MI, Lname),Order)}, ISBN, Year, Publisher;
2. MEMBER: MemNo, DriverLic(State,No), Name(Fname, MI, Lname), Address, PhoneNumber;
3. BOOK: BookID, Edition;
4. LIBRARIAN: SSN, Name, Address, Salary, Gender, Date of Birth;
5. SECTION: SectNo, Name;

## Weak Entity

- Assume the additional requirement that all the dependents of each librarian are stored in the DB

1. DEPENDENT: Name, Date of Birth, Kinship

## Relationships

1. COPY: <TITLE, BOOK> 1:M, PARTIAL/TOTAL;
2. BELONGS: <BOOK, SECTION> 1:N,TOTAL/PARTIAL;
3. HOLD: <MEMBER, TITLE> M:N, PARTIAL/PARTIAL, Date;
4. BORROW: <MEMBER, BOOK> 1:5, PARTIAL/PARTIAL, BorrowDueDate;
5. CHECKS: <LIBRARIAN, BOOK> 1:N, PARTIAL/PARTIAL;
6. MANAGES: <LIBRARIAN, SECTION> 1:1, PARTIAL/PARTIAL;
7. WORKS: <LIBRARIAN, SECTION> 1:N, TOTAL/PARTIAL;
8. DEPENDS: <LIBRARIAN, DEPENDENT> 1:N, PARTIAL/TOTAL;
9. SUPERVISES: <supervisor-LIBRARIAN, supervisee-LIBRARIAN> 1:N, PARTIAL/PARTIAL;

## Enhanced Entity-Relationship Model

- Since 1980s there has been an increase in emergence of new database applications with more demanding requirements.

- Basic concepts of ER modeling are not sufficient to represent requirements of newer, more complex applications.

- Response is development of additional ‘semantic’ modeling concepts.

## EER Model: Enhanced ER Model

- The EER model introduced the concepts of *superclass* and *subclass* entity types in the ER model
  - MEMBER (superclass): LIFE-MEMBER, REGULAR-MEMBER, and SEASON-MEMBER (Subclasses)
  - LIBRARIAN (superclass): HEAD LIBRARIANS, SALARY LIBRARIANS, and HOURLY LIBRARIANS (subclasses)
- Why?
  - To add more semantic clarity to the design
    E.g., if only salary-librarians can belong to the librarian guild, then this can be expressed as a BelongTo:<SALARY-LIBRARIAN and LIB-GUILD> and not as BelongTo:<LIBRARIAN,LIB-GUILD>
  - Minimize NULL values

## Specialization, Generalization and Inheritance

- Specialization: identifying subclasses, and their distinguishing characteristics (attributes & relationships) (Top-down design)
- Generalization: aggregate entities to a superclass entity type by identifying their common characteristics (Bottom-up design)
- Inheritance: IS_A (instance) relationship that supports attribute inheritance and relationship participation
  - Single inheritance results in a hierarchy
  - Multiple inheritance results in a lattice
    E.g., EMPLOYEE → STUDENT-ASSISTANT ←STUDENT

## Inclusion Constraints on Specialization and Generalization

- The *disjoint* constraint: the subclasses of a superclass are disjoint.
  - This means that an entity can be a member of only one subclass.
  - The entities for each class can be *user-defined* or specified with a *predicate-defined subclass.*
  - In a predicate-defined subclass, we use a selection condition on one or more attributes to define the entities of the subclass. E.g., MembershipStatus
- The *non-disjoint* constraints: specify that the subclasses are overlapping and an entity may be a member of more than one subclass.

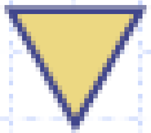
## Completeness Constraints on Specialization and Generalization

- A *total* specialization: specifies that every entity in the superclass must be a member of some of its subclasses
  - E.g., a librarian must belong to one of the subclasses of LIBRARIAN.
- A *partial* specialization: specifies that an entity may not belong to any subclass
  - E.g. , an honorary member may not belong to any of the specializations (subclasses) of MEMBER.
- Superclass via generalization is always total

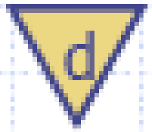
## Union Types or Categories

- Collection of entities of distinct entity types
  - E.g., Vehicle owner: person, bank, company
- Multiple Inheritance with superclasses of different types
- Category OWNER is a subclass of the set **union** of the entity types: PERSON, BANK, COMPANY
- An instance in category must exists only in one of the superclasses
- Category can be:
  - total
  - partial (with predicate definition)

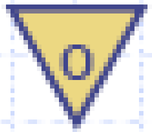
## EER Diagrams



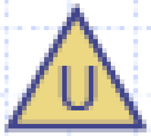
IS-A



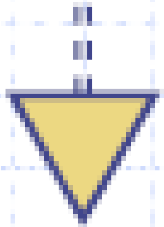
Disjoint



Non-Disjoint overlapping



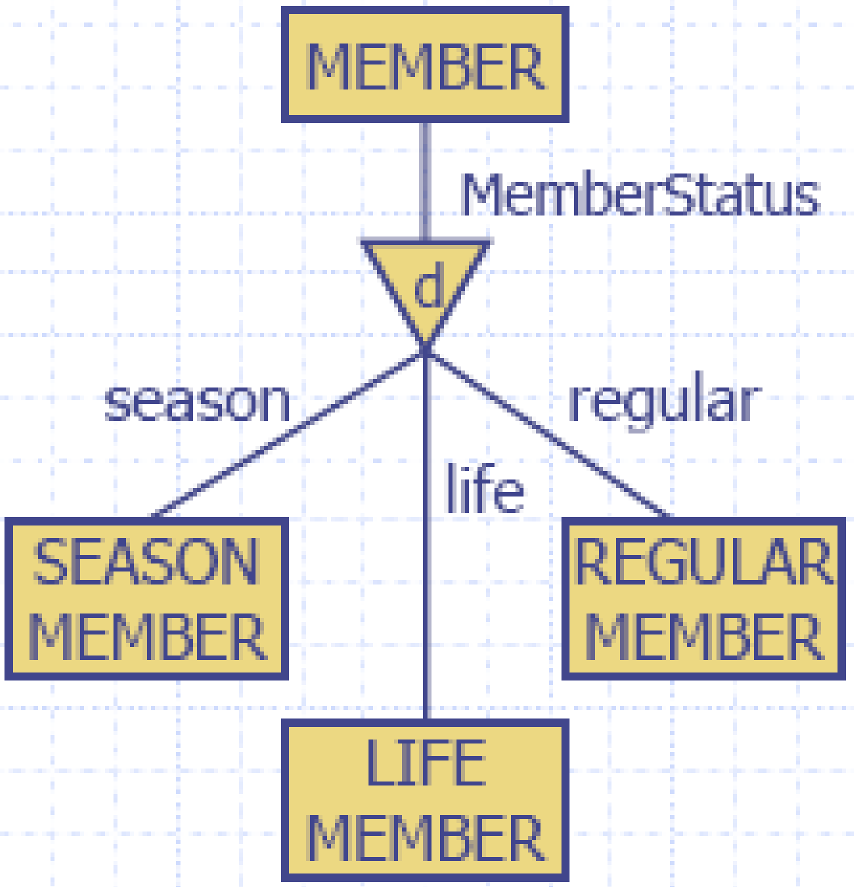
Category class union

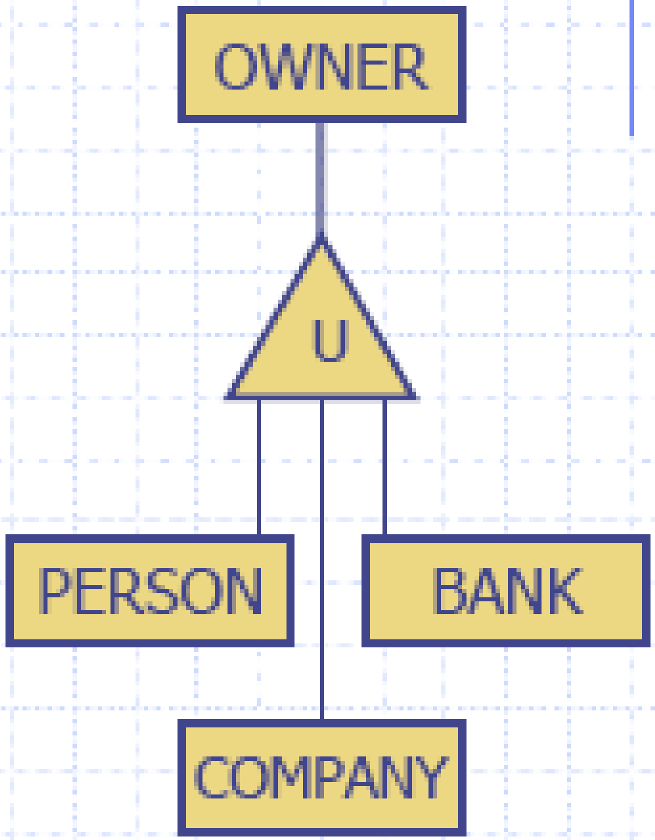


Total generalization

(dash line should be double line)

## EER Diagram: Examples

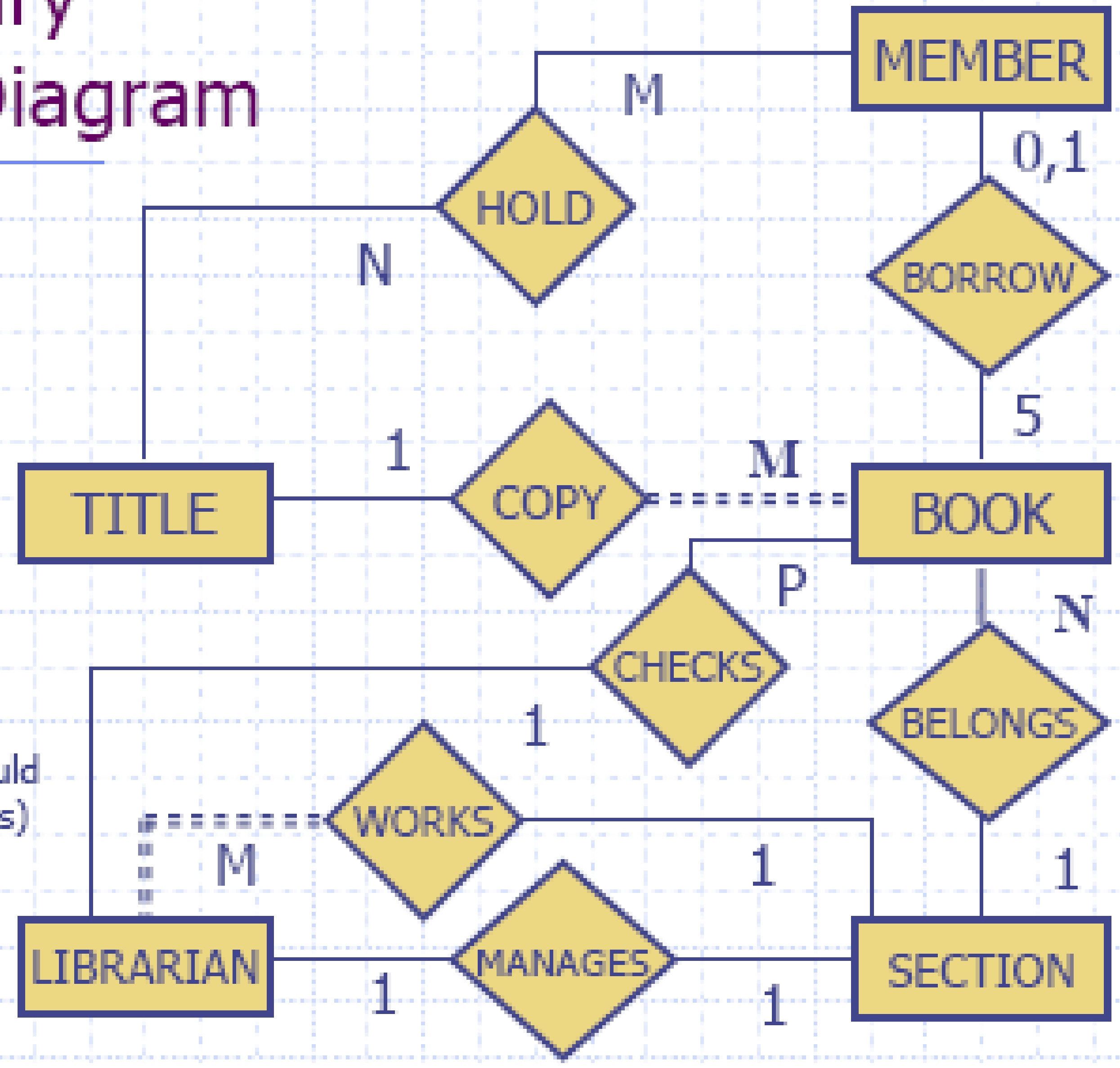




## Assumptions/Clarifications:

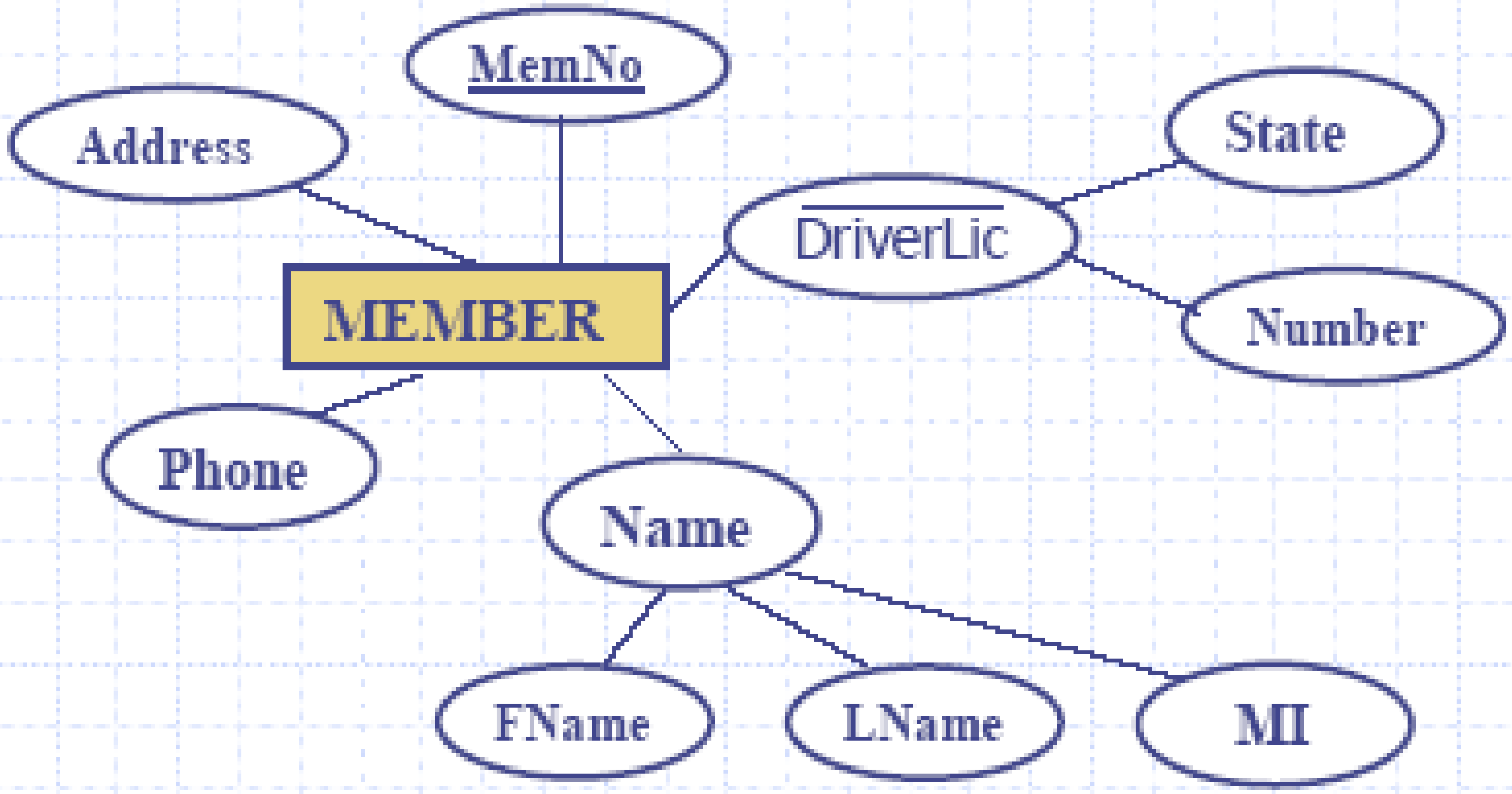
- One author writes one or more titles.
- Several co-authors write one or more titles.
- A book is a copy of a title. A title can have one or more copies of the book.
- A book has a unique id (not a copy id). If a copy id is used then book is a weak entity type.
- A particular member places a hold on a particular title.
- Not all members necessarily borrow books. Not all books are necessarily borrowed.
- Not all titles need necessarily be of books. However, all books must have a title and only one title.

# Library ER Diagram



(dash lines should be double lines)

# Library
ER Diagram…

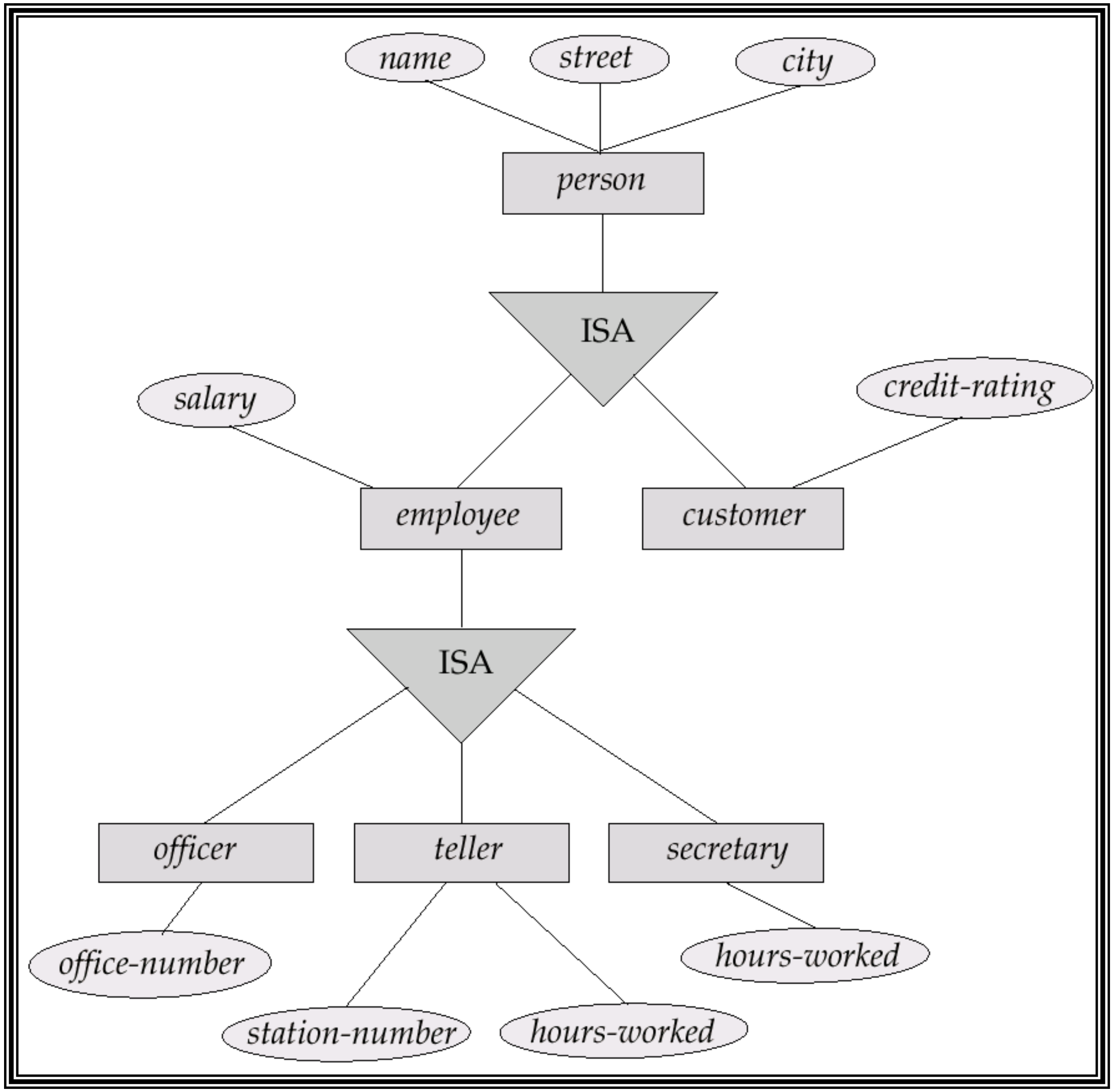


# Enhanced-ER (EER) Model Concepts

- Includes all modeling concepts of basic ER
- Additional concepts: subclasses/superclasses, specialization/generalization, categories, attribute inheritance
- The resulting model is called the enhanced-ER or Extended ER (E2R or EER) model
- It is used to model applications more completely and accurately if needed
- It includes some object-oriented concepts, such as inheritance

## Subclasses and Superclasses (1)

- An entity type may have additional meaningful subgroupings of its entities
- Example: EMPLOYEE may be further grouped into SECRETARY, ENGINEER, MANAGER, TECHNICIAN, SALARIED_EMPLOYEE, HOURLY_EMPLOYEE,…
  - Each of these groupings is a subset of EMPLOYEE entities
  - Each is called a subclass of EMPLOYEE
  - EMPLOYEE is the superclass for each of these subclasses
- These are called superclass/subclass relationships.
- Example: EMPLOYEE/SECRETARY, EMPLOYEE/TECHNICIAN


name
street
city
person
ISA
salary
credit-rating
employee
customer
ISA
officer
teller
secretary
office-number
station-number
hours-worked
hours-worked

## Subclasses and Superclasses (2)

- These are also called IS-A relationships (SECRETARY IS-A EMPLOYEE, TECHNICIAN IS-A EMPLOYEE, …).
- Note: An entity that is member of a subclass represents the same real-world entity as some member of the superclass
  - The Subclass member is the same entity in a distinct specific role
  - An entity cannot exist in the database merely by being a member of a subclass; it must also be a member of the superclass
  - A member of the superclass can be optionally included as a member of any number of its subclasses
- Example: A salaried employee who is also an engineer belongs to the two subclasses ENGINEER and SALARIED_EMPLOYEE
  - It is not necessary that every entity in a superclass be a member of some subclass

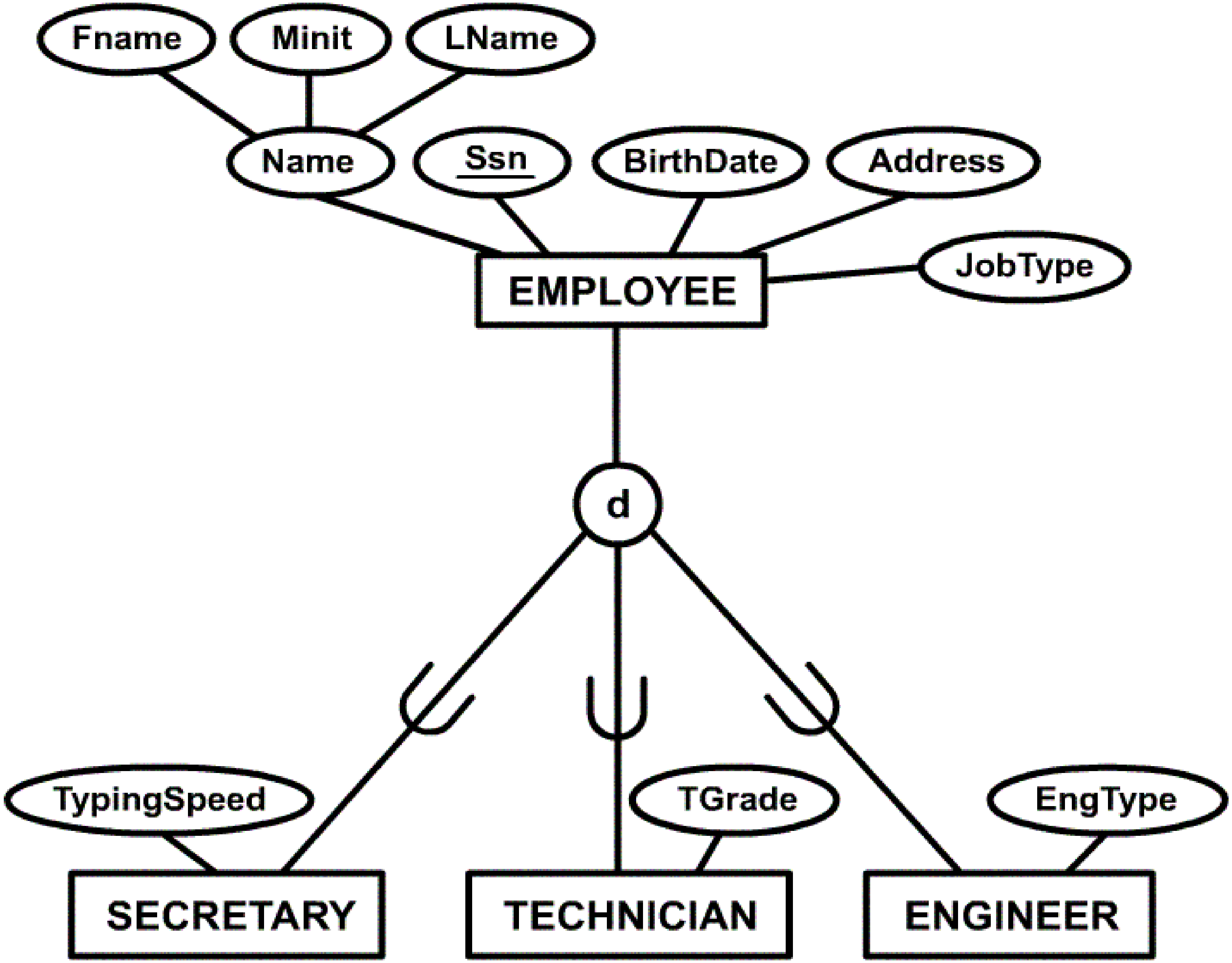
## Attribute Inheritance in Superclass / Subclass Relationships

- An entity that is member of a subclass *inherits* all attributes of the entity as a member of the superclass
- It also inherits all relationships

# Specialization

- Is the process of defining a set of subclasses of a superclass
- The set of subclasses is based upon some distinguishing characteristics of the entities in the superclass
- Example: {SECRETARY, ENGINEER, TECHNICIAN} is a specialization of EMPLOYEE based upon *job type*.
  - May have several specializations of the same superclass
- Example: Another specialization of EMPLOYEE based in *method of pay* is {SALARIED_EMPLOYEE, HOURLY_EMPLOYEE}.
  - Superclass/subclass relationships and specialization can be diagrammatically represented in EER diagrams
  - Attributes of a subclass are called specific attributes. For example, TypingSpeed of SECRETARY
  - The subclass can participate in specific relationship types. For example, BELONGS_TO of HOURLY_EMPLOYEE

# Example of a Specialization



## Generalization

- The reverse of the specialization process
- Several classes with common features are generalized into a superclass; original classes become its subclasses
- Example: CAR, TRUCK generalized into VEHICLE; both CAR, TRUCK become subclasses of the superclass VEHICLE.
  - We can view {CAR, TRUCK} as a specialization of VEHICLE
  - Alternatively, we can view VEHICLE as a generalization of CAR and TRUCK

## Generalization and Specialization

- Arrow pointing to the generalized superclass represents a generalization
- Arrows pointing to the specialized subclasses represent a specialization
- We do not use this notation because it is often subjective as to which process is more appropriate for a particular situation
- We advocate not drawing any arrows in these situations
- A superclass or subclass represents a set of entities
- Shown in rectangles in EER diagrams (as are entity types)
- Sometimes, all entity sets are simply called classes, whether they are entity types, superclasses, or 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.

## Constraints on Specialization and Generalization (1)

- If we can determine exactly those entities that will become members of each subclass by a condition, the subclasses are called *predicate-defined* (or condition-defined) subclasses
  - Condition is a constraint that determines subclass members
  - Display a predicate-defined subclass by writing the predicate condition next to the line attaching the subclass to its superclass
- If all subclasses in a specialization have membership condition on same attribute of the superclass, specialization is called an *attribute defined*-specialization
  - Attribute is called the defining attribute of the specialization
  - Example: JobType is the defining attribute of the specialization {SECRETARY, TECHNICIAN, ENGINEER} of EMPLOYEE
- If no condition determines membership, the subclass is called *user-defined*
  - Membership in a subclass is determined by the database users by applying an operation to add an entity to the subclass
  - Membership in the subclass is specified individually for each entity in the superclass by the user

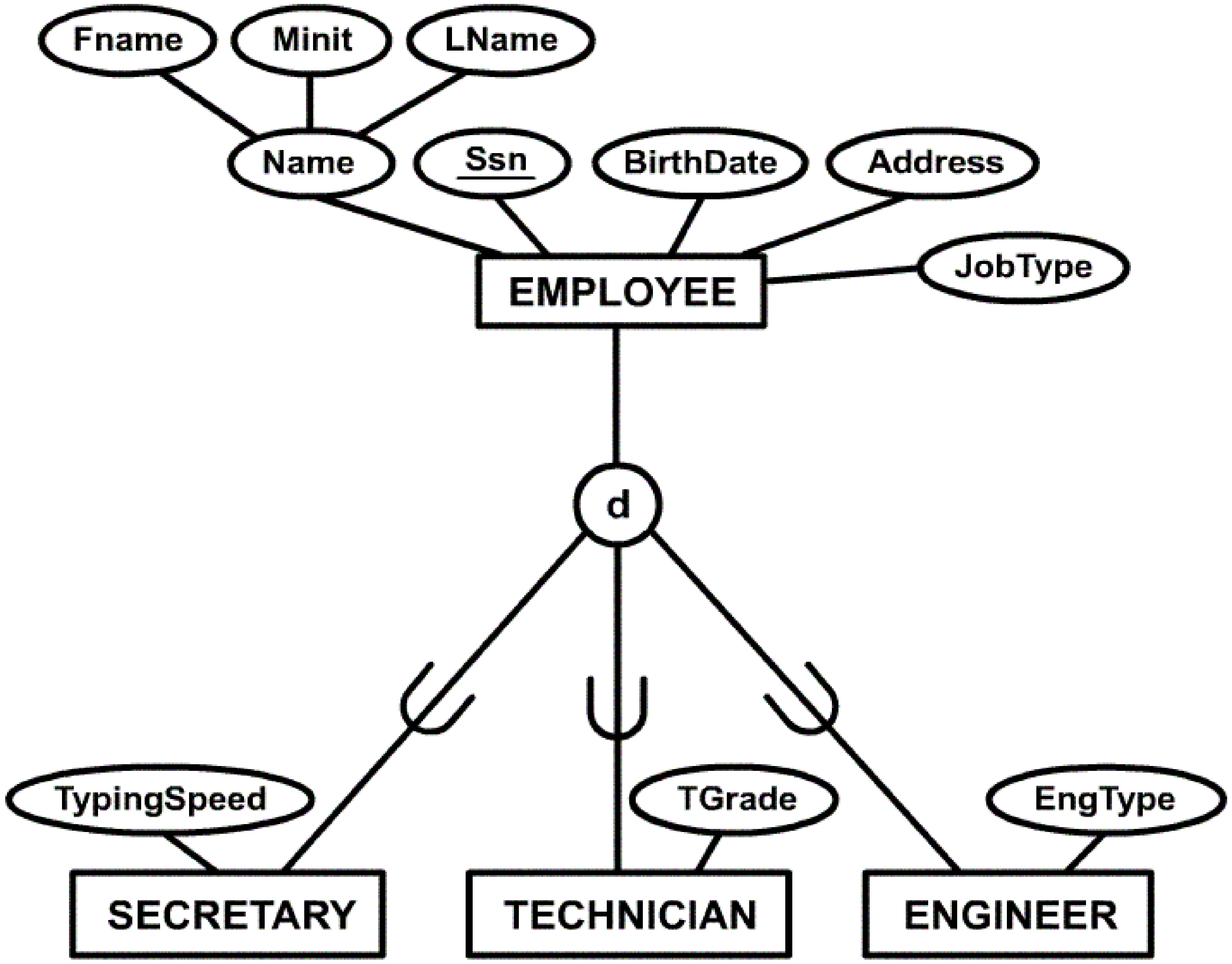
## Constraints on Specialization and Generalization (2)

- Two other conditions apply to a specialization/generalization:
- **Disjointness Constraint**:
  - Specifies that the subclasses of the specialization must be disjointed (an entity can be a member of at most one of the subclasses of the specialization)
  - Specified by d in EER diagram
  - If not disjointed, overlap; that is the same entity may be a member of more than one subclass of the specialization
  - Specified by o in EER diagram
- **Completeness Constraint**:
  - Total specifies that every entity in the superclass must be a member of some subclass in the specialization/ generalization
  - Shown in EER diagrams by a double line
  - Partial allows an entity not to belong to any of the subclasses
  - Shown in EER diagrams by a single line

## Constraints on Specialization and Generalization (3)

- Hence, we have four types of specialization/generalization:
  - Disjoint, total
  - Disjoint, partial
  - Overlapping, total
  - Overlapping, partial
- Note: Generalization usually is total because the superclass is derived from the subclasses.

# Example of disjoint partial Specialization

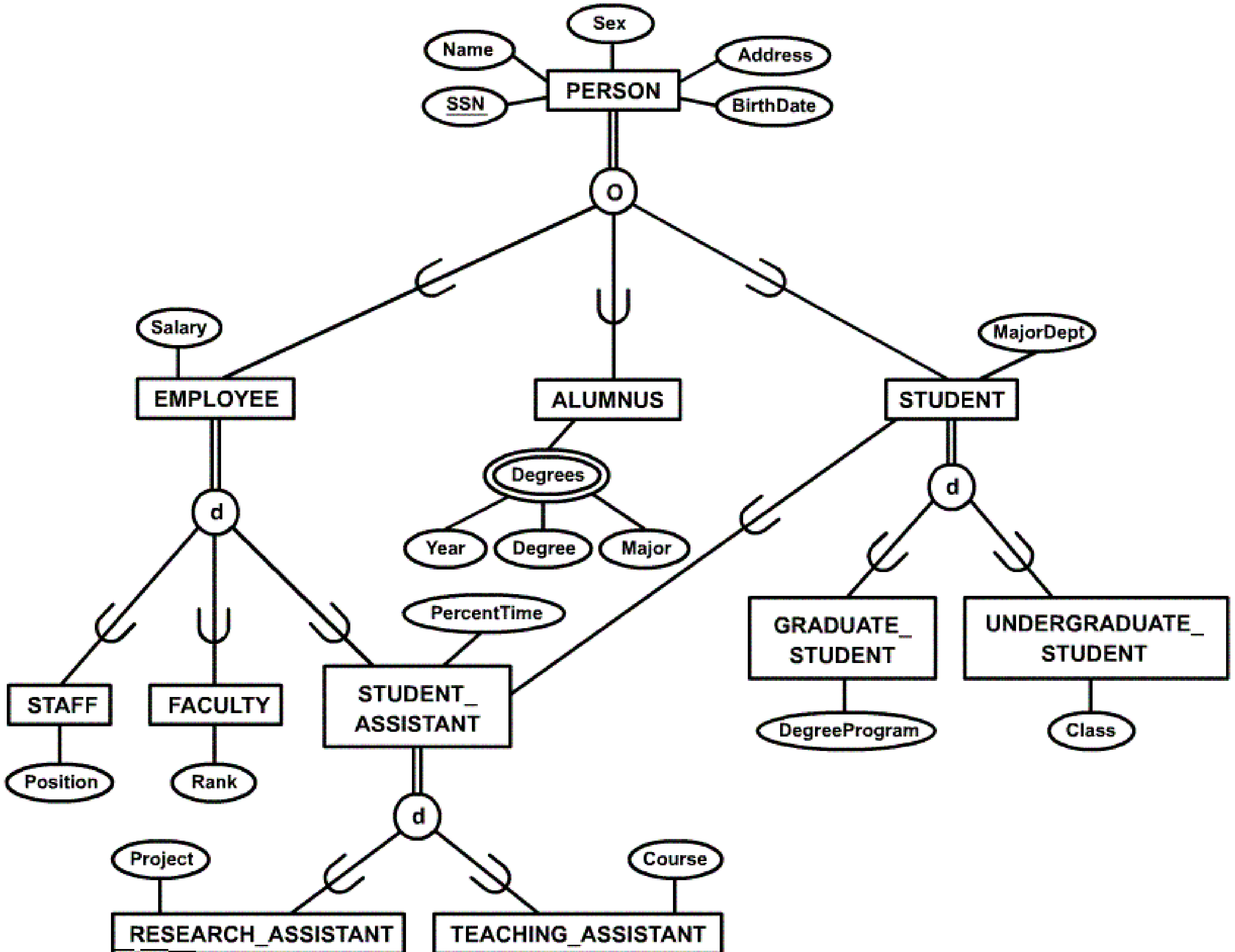


## Specialization / Generalization Hierarchies, Lattices and Shared Subclasses

- A subclass may itself have further subclasses specified on it
- Forms a hierarchy or a lattice
- Hierarchy has a constraint that every subclass has only one superclass (called *single inheritance*)
- In a lattice, a subclass can be subclass of more than one superclass (called *multiple inheritance*)

- In a lattice or hierarchy, a subclass inherits attributes not only of its direct superclass, but also of all its predecessor superclasses
- A subclass with more than one superclass is called a shared subclass
- Can have specialization hierarchies or lattices, or generalization hierarchies or lattices
- In specialization, start with an entity type and then define subclasses of the entity type by successive specialization (top down conceptual refinement process)
- In generalization, start with many entity types and generalize those that have common properties (bottom up conceptual synthesis process)
- In practice, the combination of two processes is employed

# Specialization / Generalization Lattice Example (UNIVERSITY)

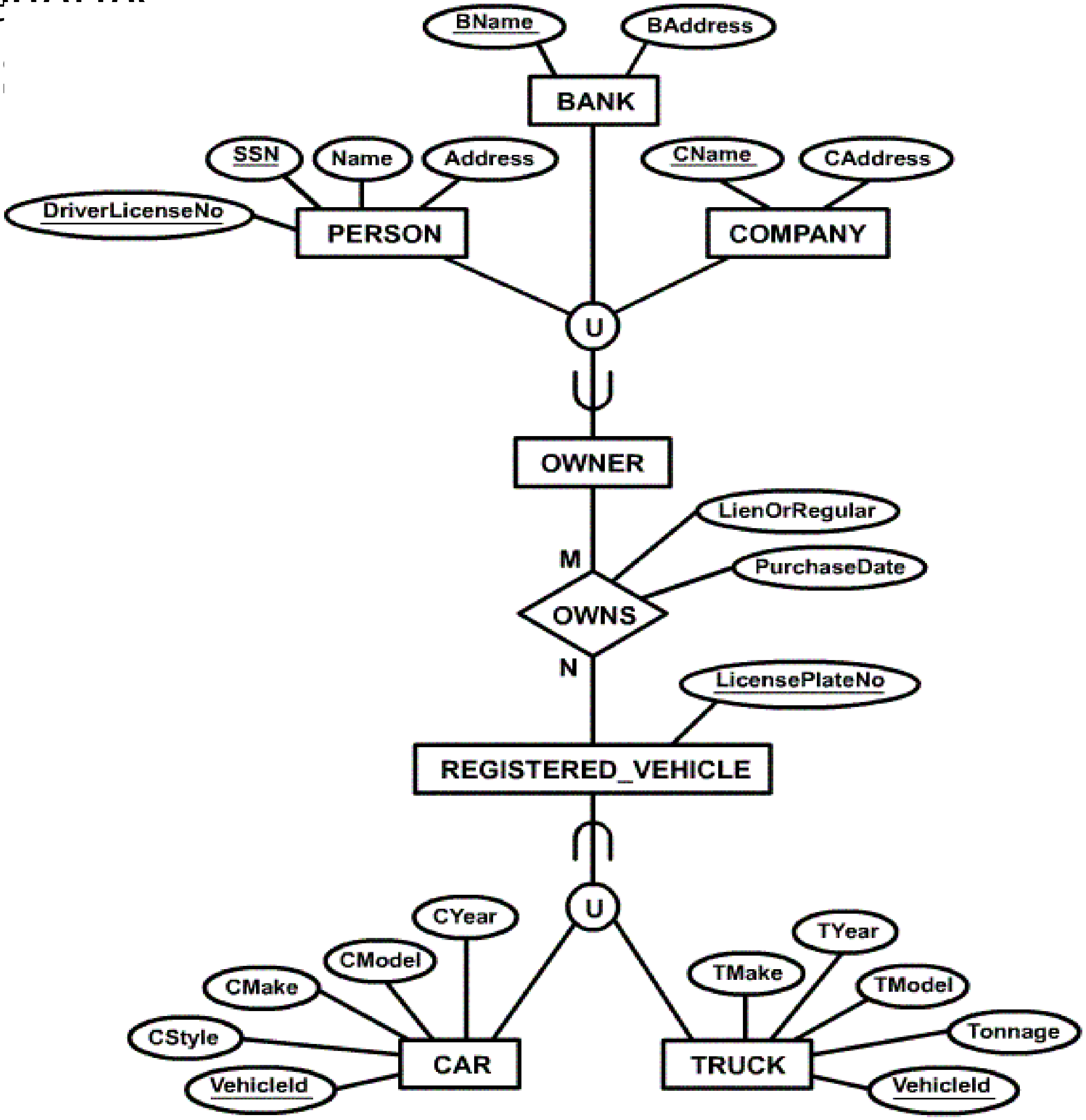


# Categories (UNION TYPES)

- All of the superclass/subclass relationships we have seen thus far have a single superclass
- A shared subclass is subclass in more than one distinct superclass/subclass relationships, where each relationships has a single superclass (multiple inheritance)
- In some cases, need to model a single superclass/subclass relationship with more than one superclass
- Superclasses represent different entity types
- Such a subclass is called a category or UNION TYPE

- Example: Database for vehicle registration, vehicle owner can be a person, a bank (holding a lien on a vehicle) or a company.
  - Category (subclass) OWNER is a subset of the union of the three superclasses COMPANY, BANK, and PERSON
  - A category member must exist in at least one of its superclasses
- Note: The difference from shared subclass, which is subset of the intersection of its superclasses (shared subclass member must exist in all of its superclasses).

# Example of cate
# (UNION TYPE



## Specialization

This is the process of maximising the differences between members of an entity by identifying their distinguishing characteristics.

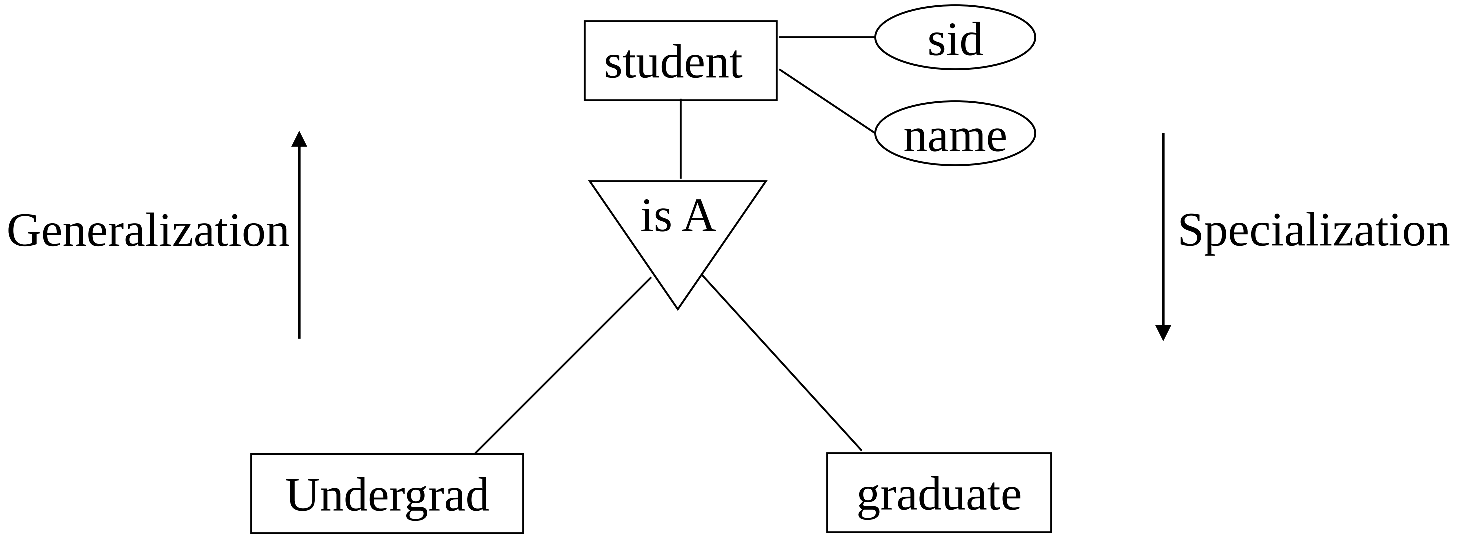
- Staff(<u>staff_no</u>,name,address,dob)
- Manager(bonus)
- Secretary(wp_skills)
- Sales_personnel(sales_area, car_allowance)

## GENERALIZATION AND SPECIALIZATION

- ***Generalization*** is the result of computing the union of two or more entity sets to produce a higher-level entity set. It represents the containment relationship that exists between the higher-level entity set and one or more lower-level entity sets.

- ***Specialization*** constructs the lower level entity sets that are a subset of a higher level entity set.

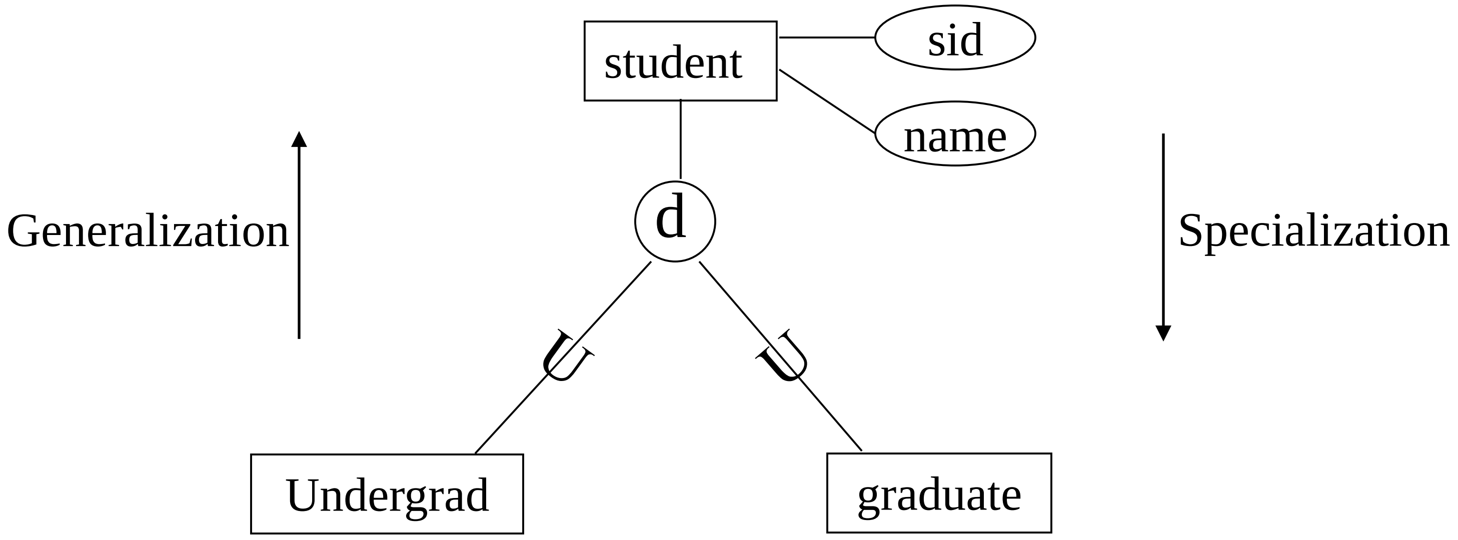
# GENERALIZATION AND SPECIALIZATION



## GENERALIZATION AND SPECIALIZATION

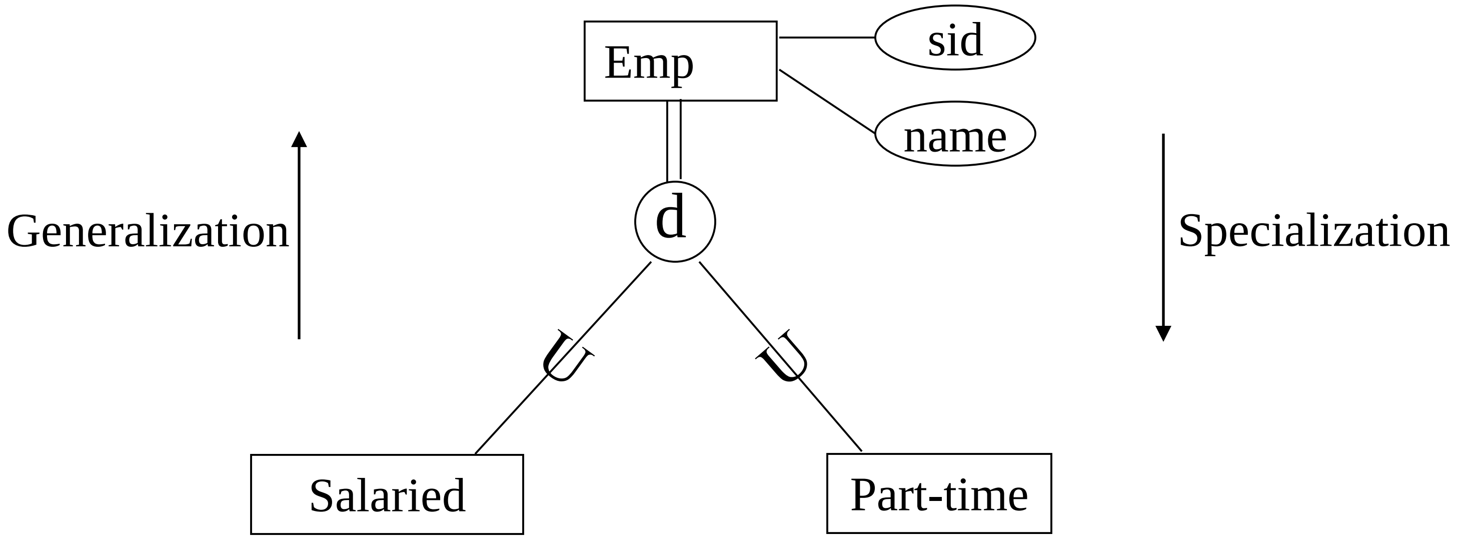
- Undergrad and graduate are termed subclasses of the superclass student.
- This is a superclass/subclass or simply class/subclass relationship.
- A member of a subclass MUST be a member of the superclass.
- An alternative notation is the Union symbol
- The circle with d specifies that the specializations are disjoint.  A member of Undergrad entity set may NOT be a member of the graduate entity set.

# GENERALIZATION AND SPECIALIZATION



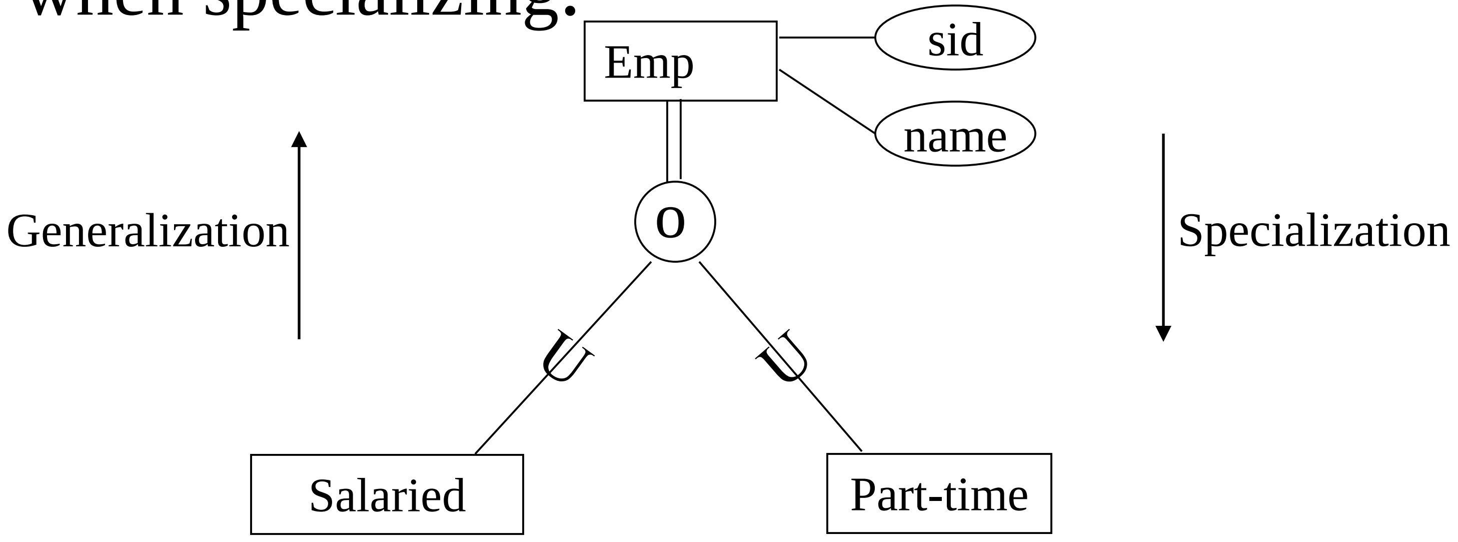
- A design may require all members of an entity-set to be specialized. For example, an employee MUST be a member of either a Salaried or Part-time. Use double lines to dictate this constraint

# GENERALIZATION AND SPECIALIZATION



# GENERALIZATION AND SPECIALIZATION

- One may allow the specialized entity sets to overlap.  For example, an entity might be both a Salaried and Part-time.  “o” stands for Overlap when specializing.

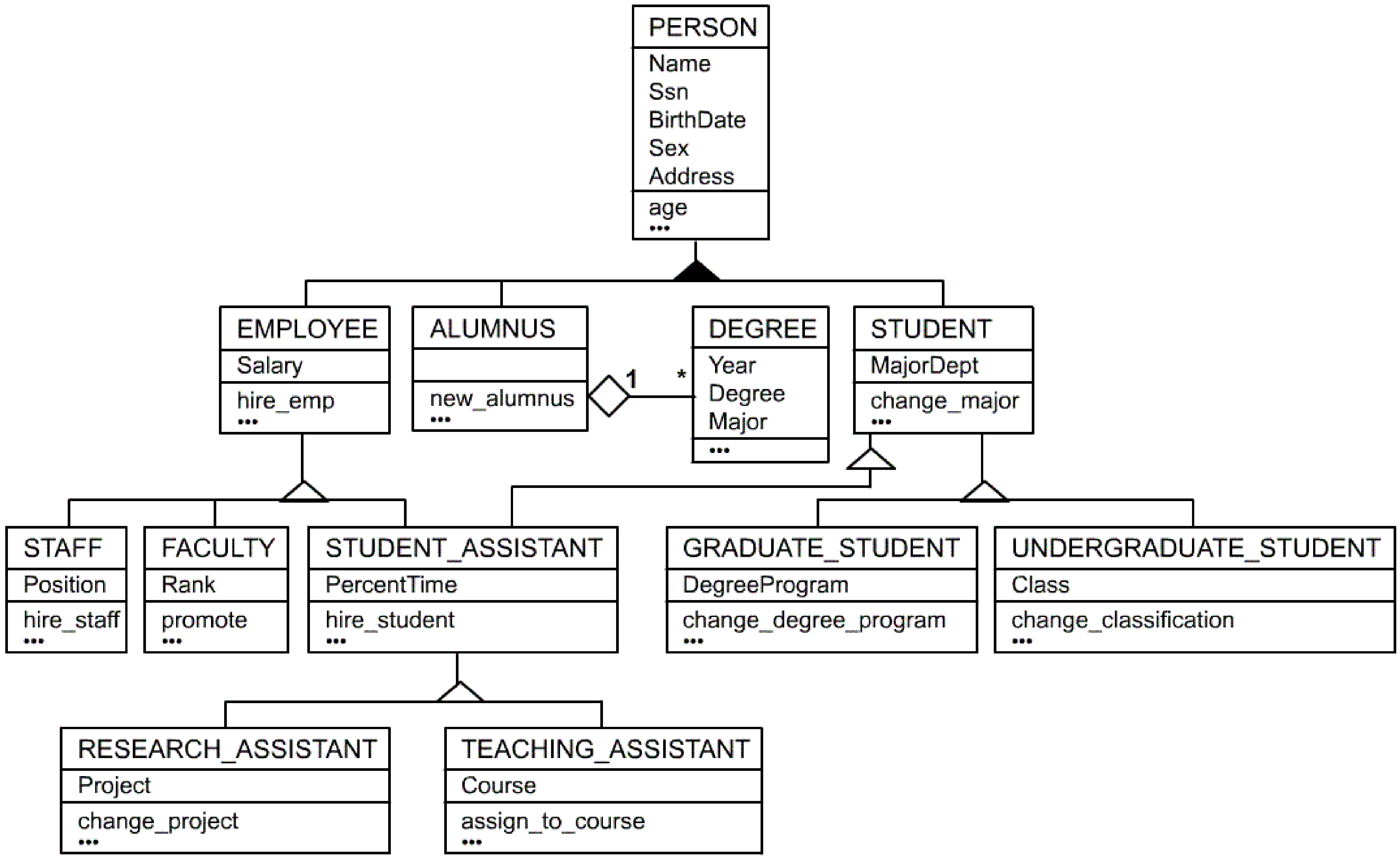


## Generalisation

Generalisation is the process of minimising the differences between entities byidentifying common features.

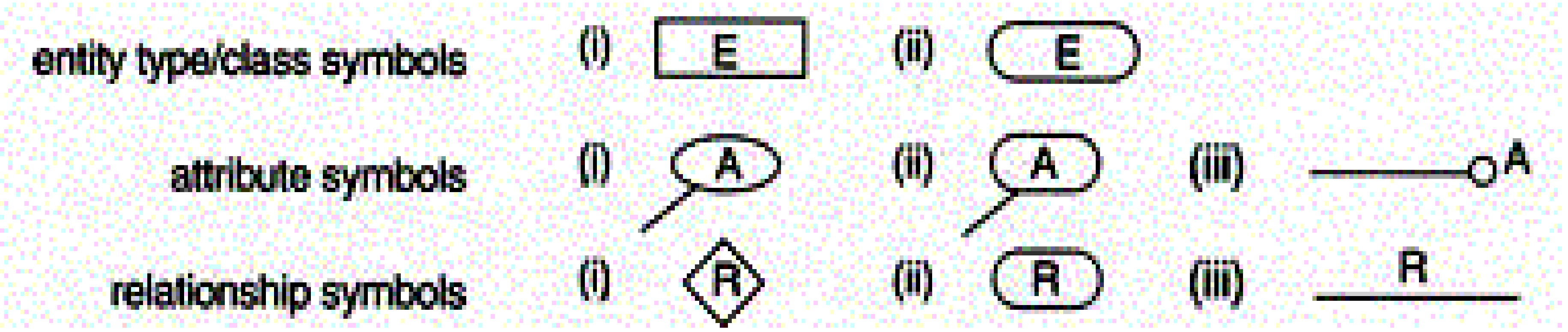
This is the identification of a generalised superclass from the original subclasses. This is the process of identifying the common attributes and relationships.

# UML Example for Displaying Specialization / Generalization

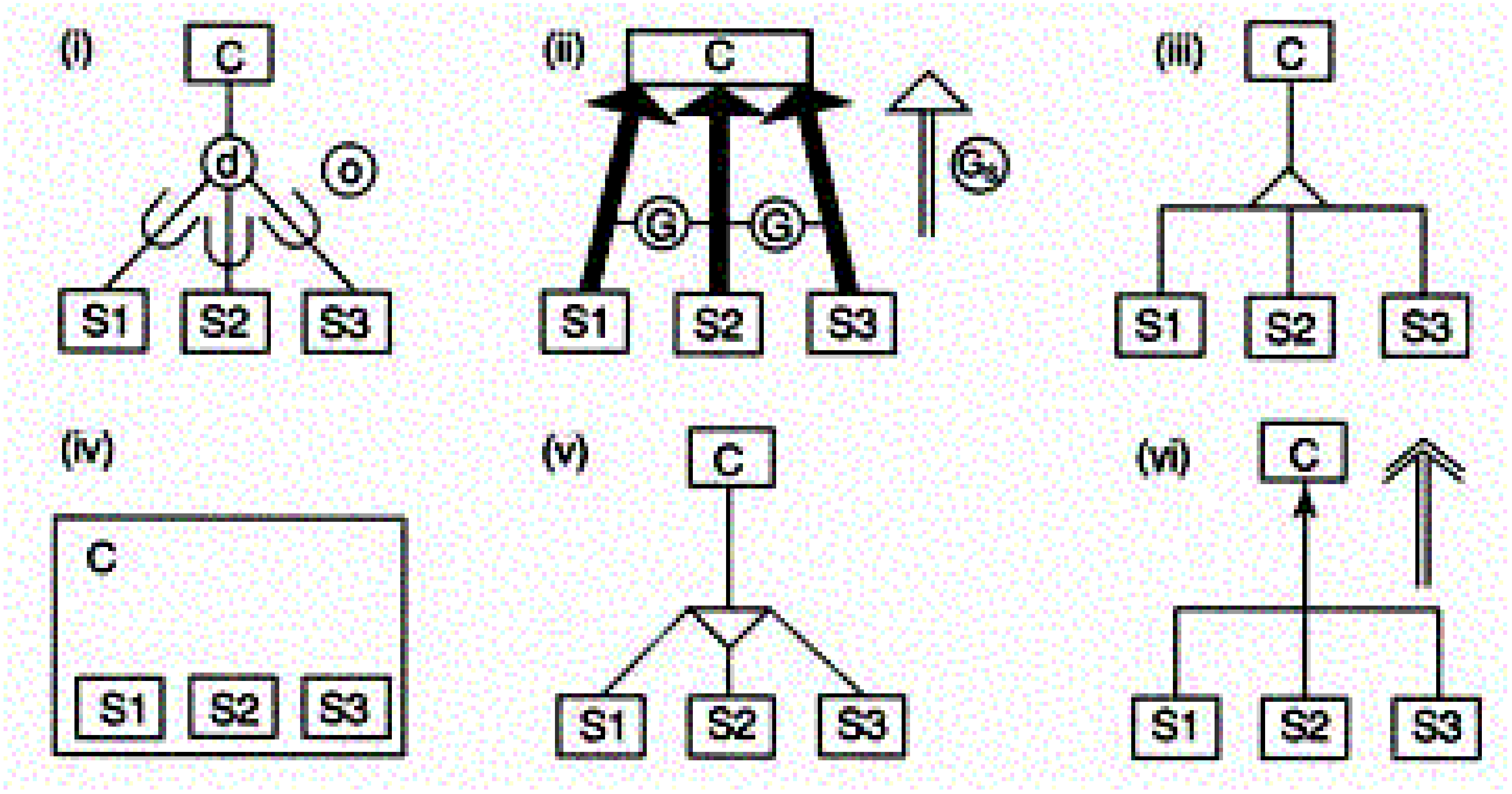


## Alternative Diagrammatic Notations

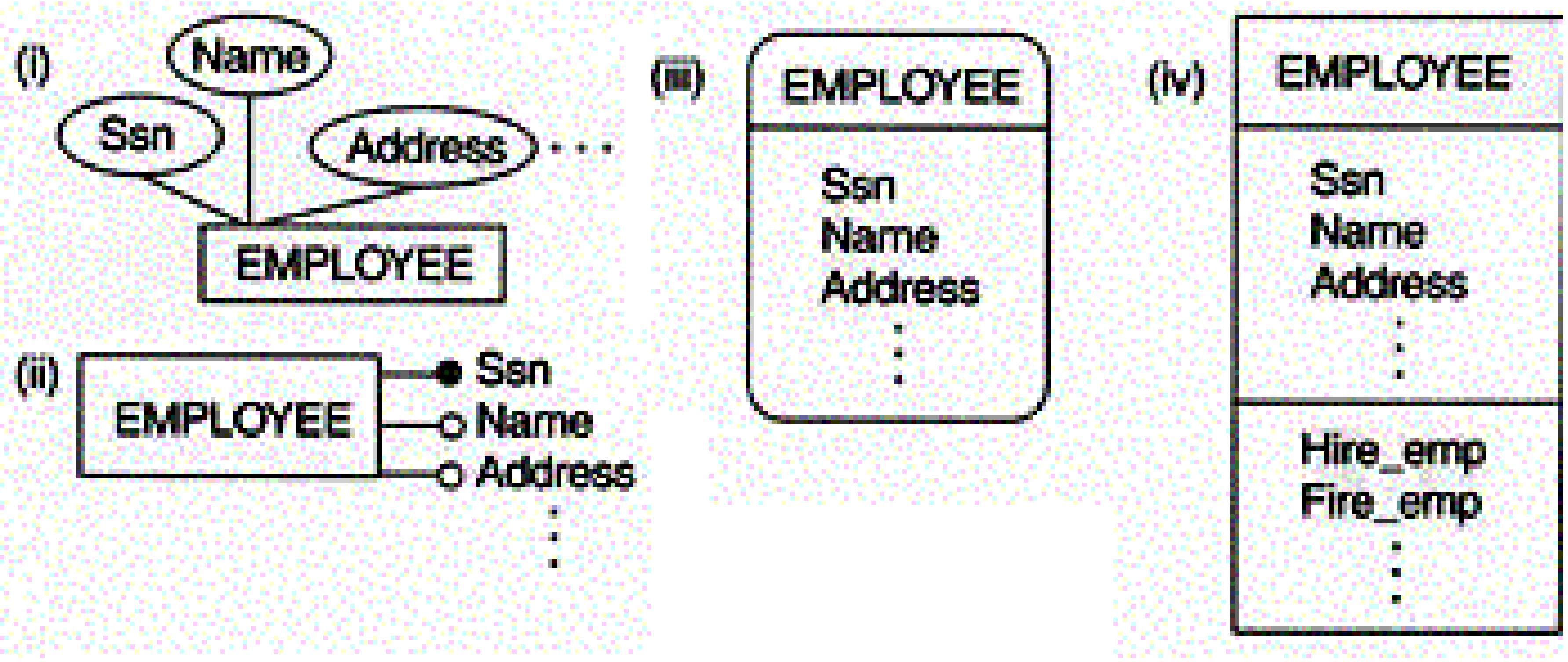
Symbols for entity type / class,
attribute and relationship



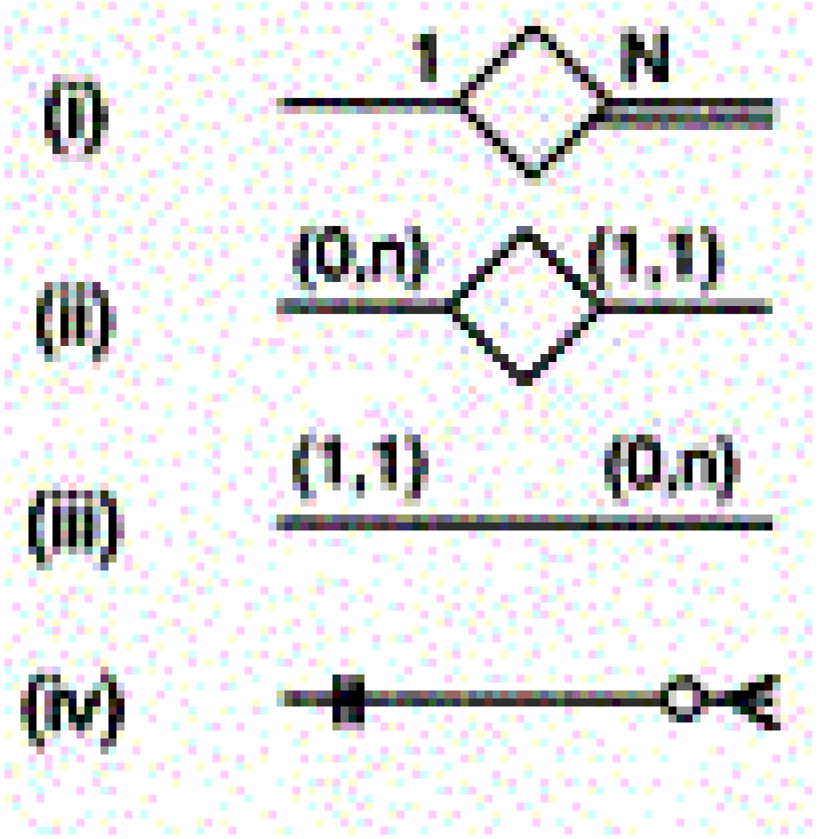
# Notations for displaying specialization / generalization



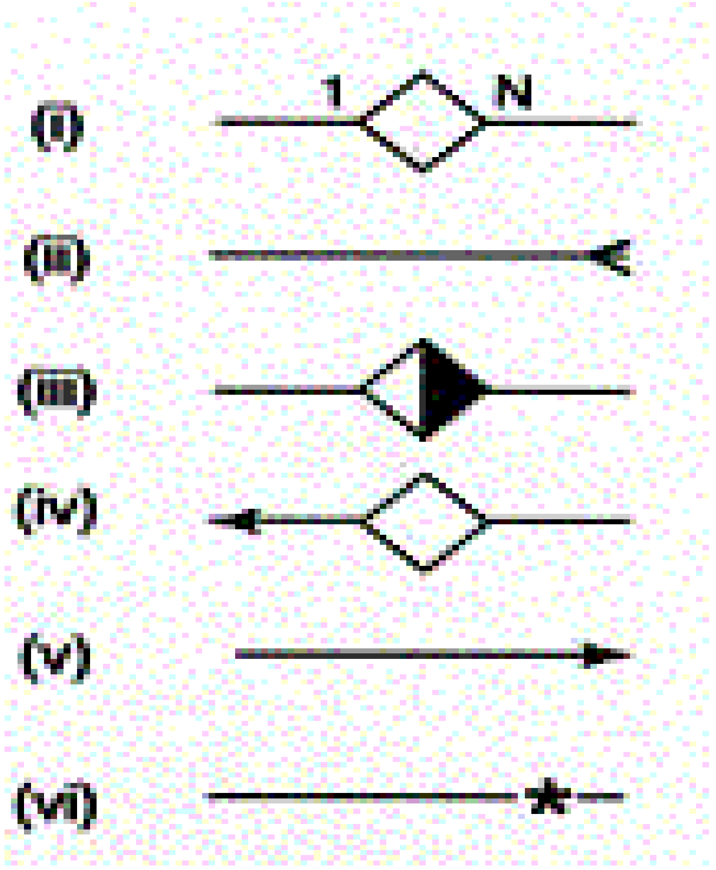
## Displaying attributes



## Various (min, max) notations



Displaying cardinality ratios



(i)
1
N
(ii)
(iii)
(iv)
(v)
(vi)