



# Conceptual DB Design: ER & EER Models

**Dr. Dang Tran Khanh**

Department of Information Systems

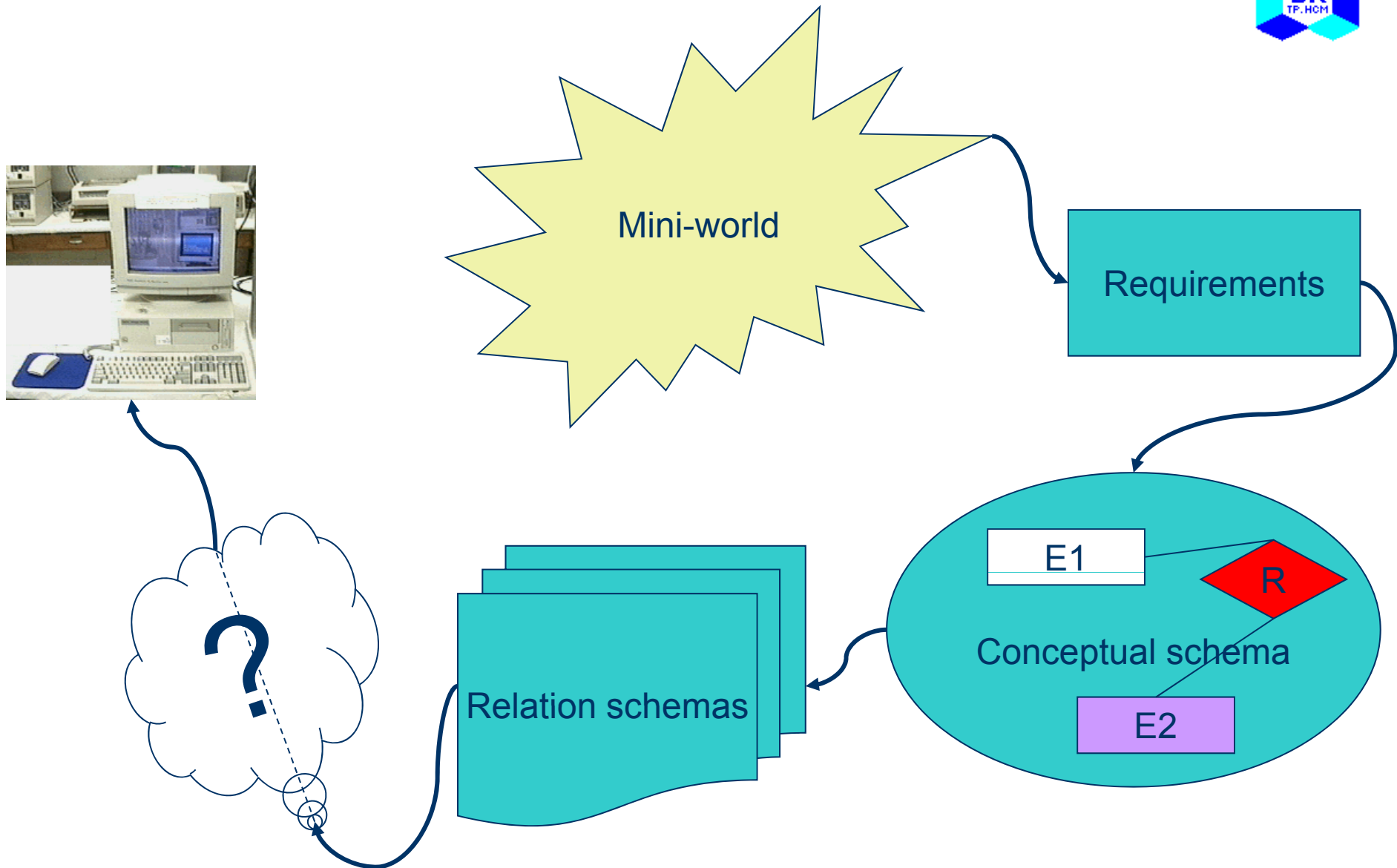
Faculty of Computer Science and Engineering

[khanh@cse.hcmut.edu.vn](mailto:khanh@cse.hcmut.edu.vn)

# Outline

- Overview of DB Design Phases
- ER Model
- EER Model
- Discussion: Problems with ER Model
- Exercises
- Reading Suggestion:
  - [1]: Chapters 3, 4
  - [4]: Chapters 11, 12
  - A. Badia: "*Entity-Relationship Modeling Revisited*", SIGMOD Record, 33(1), March 2004, 77-82

# Overview of DB Design Phases



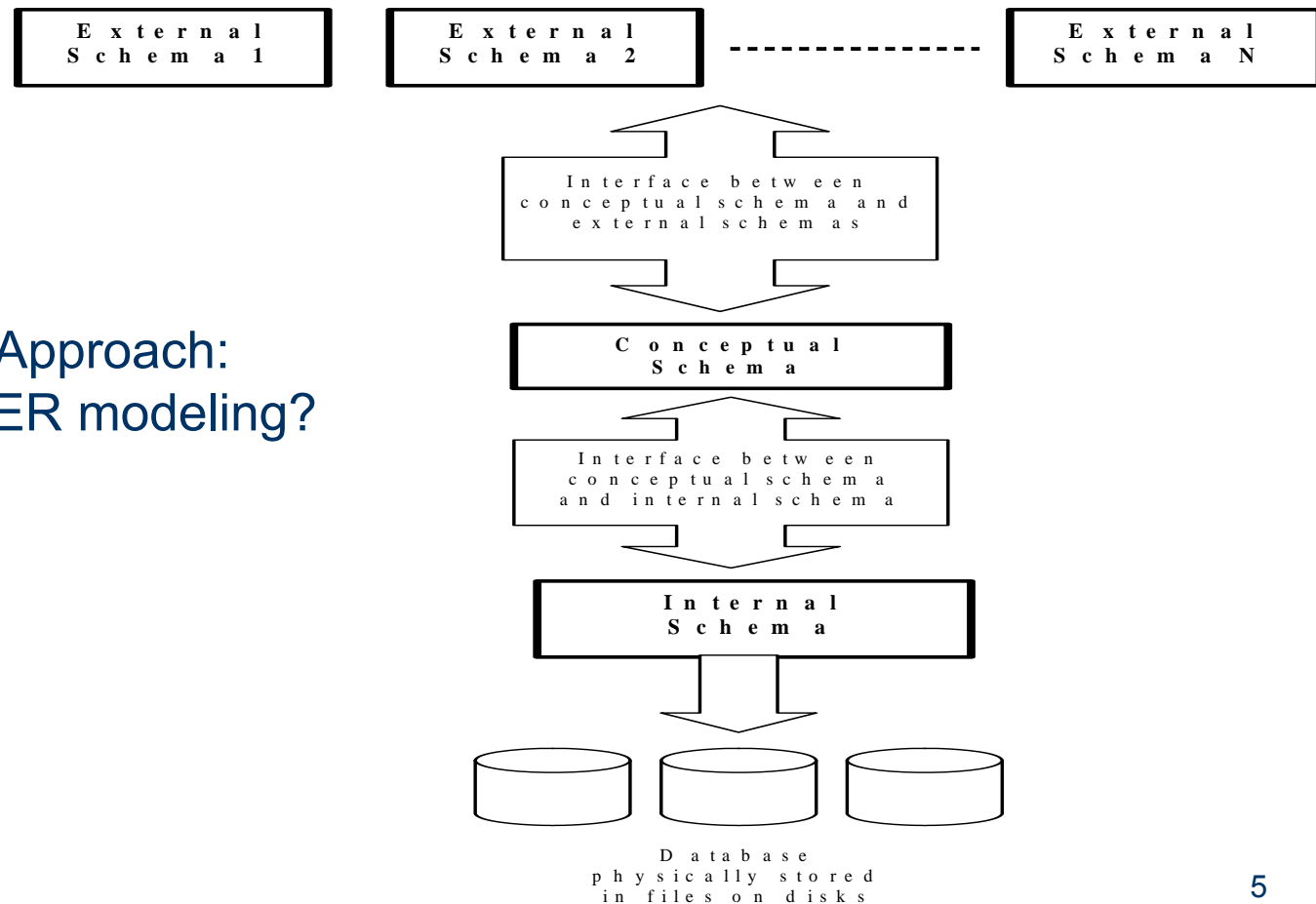
**Top-Down DB Design Phases**

# ER Model

- What is ER Model? And Why?
- Example COMPANY Database
- ER Model Concepts
- Alternative Diagrammatic Notations

# What is ER Model? And Why?

- Database Approach:  
where is ER modeling?



# What is ER Model? And Why?

- ER modeling is a logical organisation of data within a database system
- ER modeling technique is based on relational data model
- Why use ER data modelling:
  - User requirements can be specified formally & unambiguously
  - The conceptual data model is independent of any particular DBMS
  - It does not involve any physical or implemental details
  - It can be easily understood by ordinary users.
  - It provides an effective bridge between informal user requirements and logical database design and implementation

# Example COMPANY Database

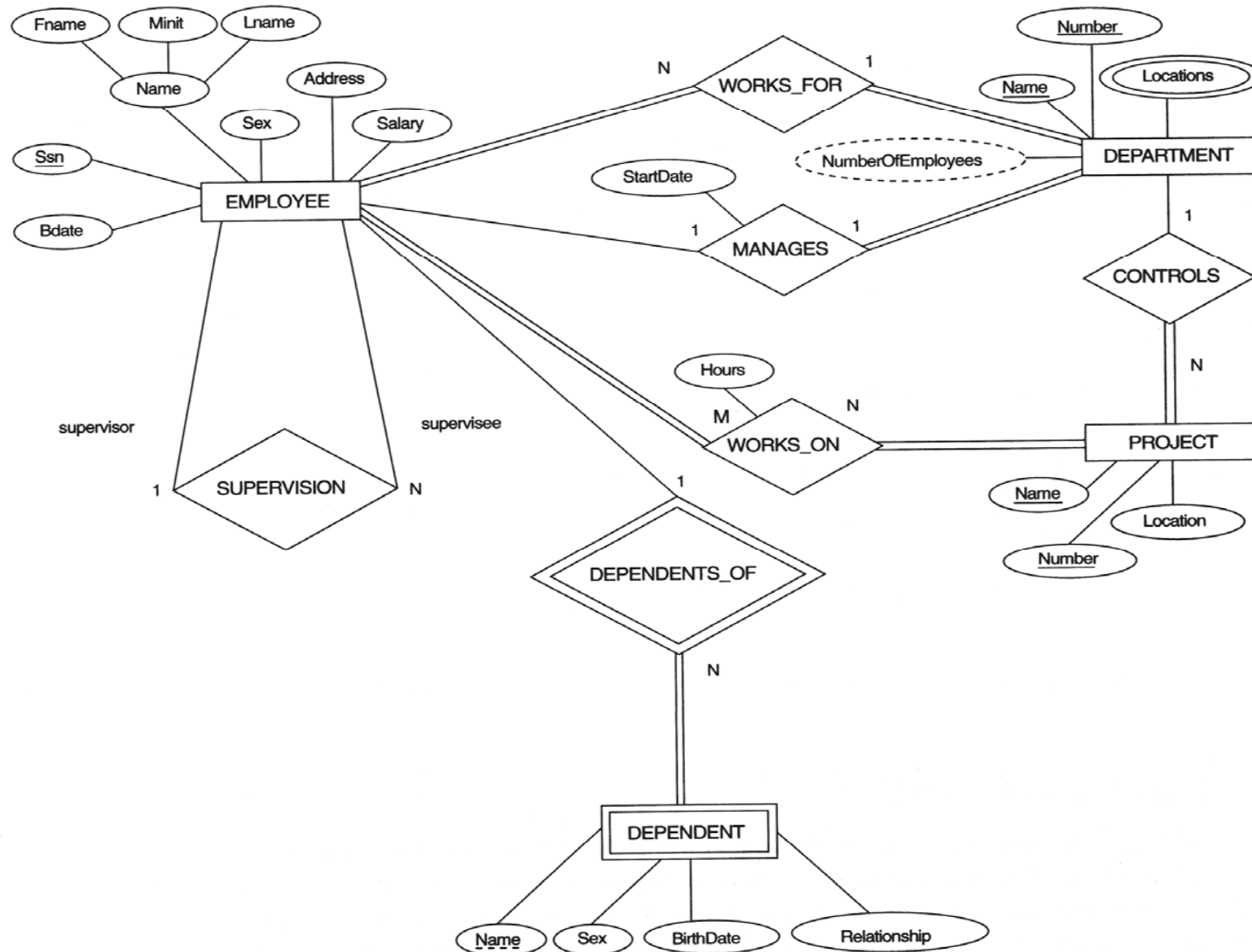
- Requirements of the Company (oversimplified for illustrative purposes)
  - The company is organized into DEPARTMENTS. Each department has a name, number and an employee who *manages* the department. We keep track of the start date of the department manager
  - Each department controls a number of PROJECTs. Each project has a name, number and is located at a single location

# Example COMPANY Database

- Requirements of the Company (oversimplified for illustrative purposes)
  - We store each EMPLOYEE's social security number, address, salary, sex, and birthdate. 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 track of their name, sex, birthdate, and relationship to employee



# Example COMPANY Database



# ER Model Concepts

- Entities and Attributes
  - Entities are specific objects or things in the mini-world that are represented in the database
    - E.g., the EMPLOYEE John Smith, the Research DEPARTMENT, the ProductX PROJECT, etc.
  - Attributes are properties used to describe an entity
    - E.g., an EMPLOYEE entity may have a Name, SSN, Address, Sex, BirthDate
  - A specific entity will have a value for each of its attributes
    - E.g., a specific employee entity may have Name='John Smith', SSN='123456789', Address ='731, Fondren, Houston, TX', Sex='M', BirthDate='09-JAN-55'
  - Each attribute has a *value set* (or data type) associated with it
    - E.g., integer, string, subrange, enumerated type, etc.

# ER Model Concepts

- Types of Attributes
  - Simple
    - Each entity has a single atomic value for the attribute. For example, SSN or Sex
  - Composite
    - The attribute may be composed of several components. For example, Address (Apt#, House#, Street, City, State, ZipCode, Country) or Name (FirstName, MiddleName, LastName). Composition may form a hierarchy where some components are themselves composite
  - Multi-valued
    - An entity may have multiple values for that attribute. For example, Color of a CAR or PreviousDegrees of a STUDENT. Denoted as {Color} or {PreviousDegrees}

# ER Model Concepts

- Types of Attributes
  - In general, composite and multi-valued attributes may be nested arbitrarily to any number of levels although this is rare
    - E.g., PreviousDegrees of a STUDENT is a composite multi-valued attribute denoted by  
 $\{ \textit{PreviousDegrees} (\textit{College}, \textit{Year}, \textit{Degree}, \textit{Field}) \}$
  - Derived Attribute
    - Attribute that represents a value that is derivable from value of a related attribute, or set of attributes, not necessarily in the same entity type

# ER Model Concepts

- Entity Types and Key Attributes
  - Entities with the same basic attributes are grouped or typed into an entity type. For example, the EMPLOYEE entity type or the PROJECT entity type
  - An attribute of an entity type for which each entity must have a unique value is called a key attribute of the entity type. For example, SSN of EMPLOYEE
  - A key attribute may be composite. For example, VehicleTagNumber is a key of the CAR entity type with components (Number, State)
  - An entity type may have more than one key. For example, the CAR entity type may have two keys:
    - VehicleIdentificationNumber (popularly called VIN) and
    - VehicleTagNumber (Number, State), also known as license\_plate number

# ER Model Concepts

- **ENTITY SET** corresponding to the **ENTITY TYPE CAR**  
*car*

Registration(RegistrationNumber, State), VehicleID, Make, Model, Year, (Color)

*car1*

*((ABC 123, TEXAS), TK629, Ford Mustang, convertible, 1999, (red, black))*

*car2*

*((ABC 123, NEW YORK), WP9872, Nissan 300ZX, 2-door, 2002, (blue))*

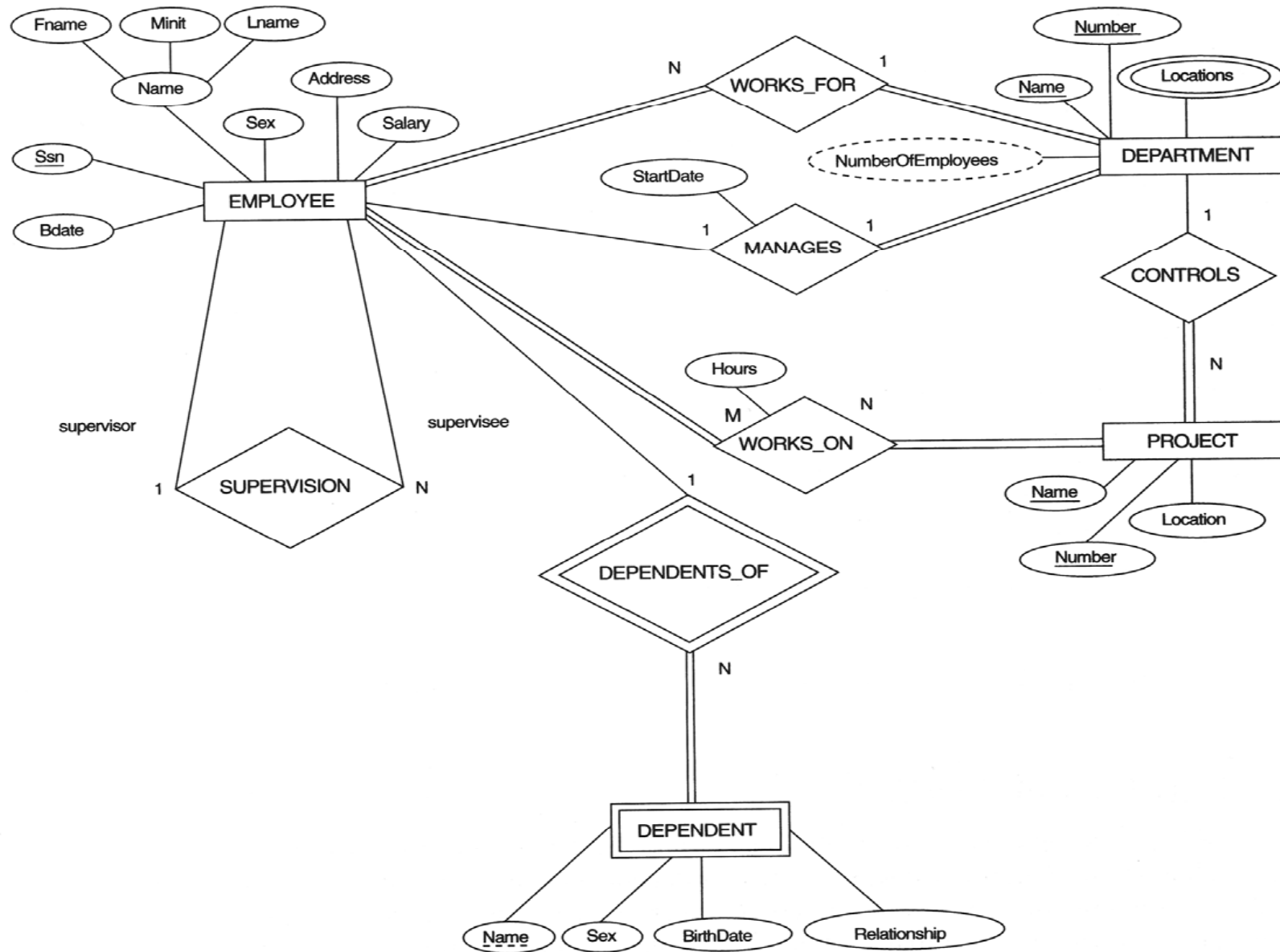
*car3*

*((VSY 720, TEXAS), TD729, Buick LeSabre, 4-door, 2003, (white, blue))*

# ER Model Concepts

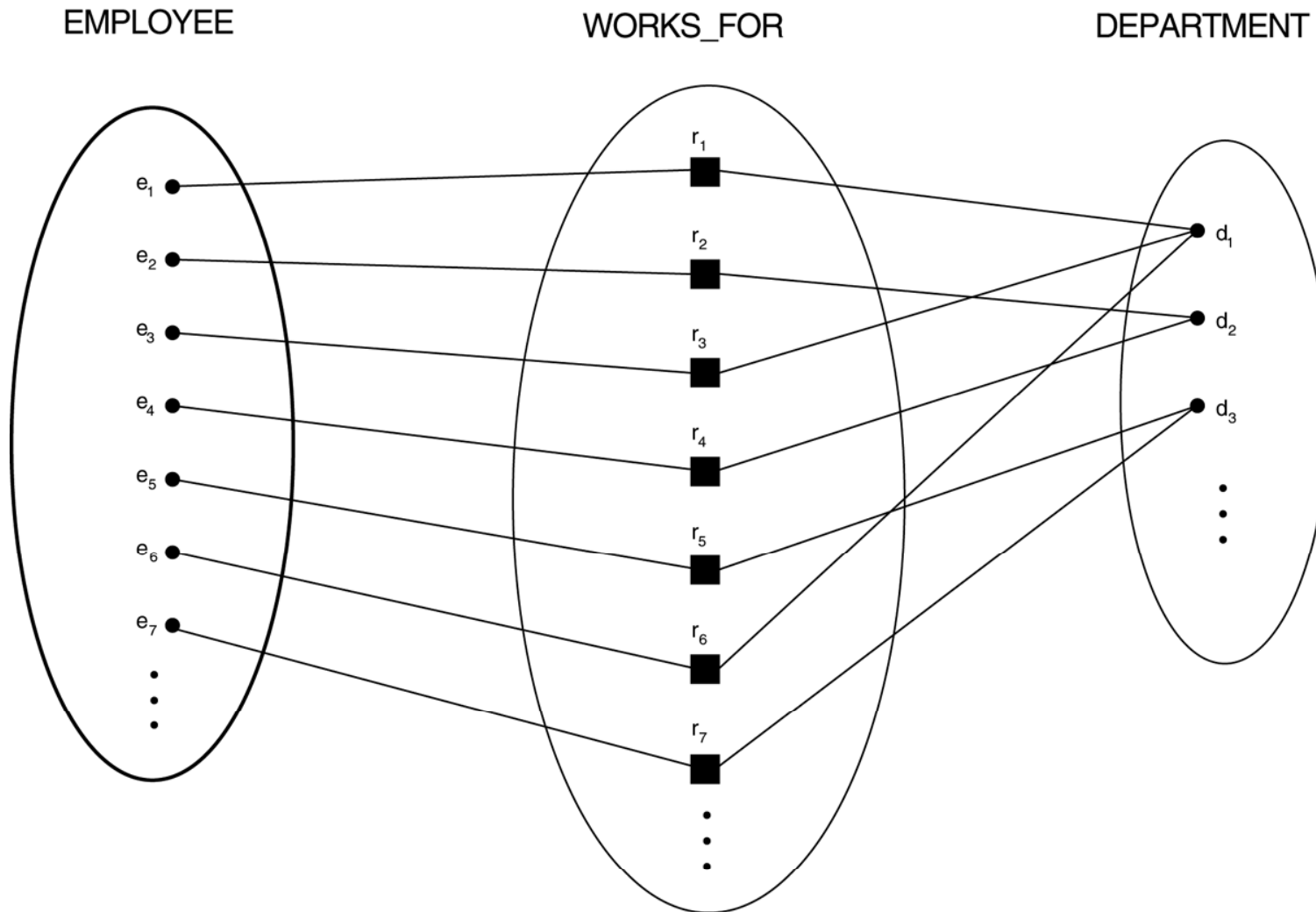
- Relationships and Relationship Types
  - A relationship relates two or more distinct entities with a specific meaning. For example, EMPLOYEE John Smith works on the ProductX PROJECT or EMPLOYEE Franklin Wong manages the Research DEPARTMENT
  - Relationships of the same type are grouped or typed into a relationship type. For example, the WORKS\_FOR relationship type in which EMPLOYEES & DEPARTMENTS participate, or the MANAGES relationship type in which EMPLOYEES & DEPARTMENTS participate
  - The degree of a relationship type is the number of participating entity types. Both MANAGES and WORKS\_ON are **binary** relationships

# Example COMPANY Database





# Example relationship instances

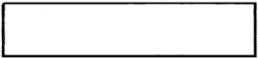
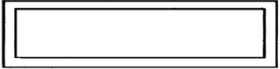
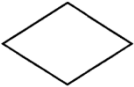
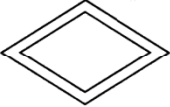




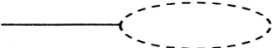


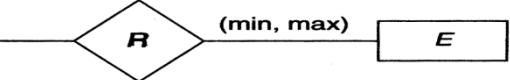


# ER Model Concepts

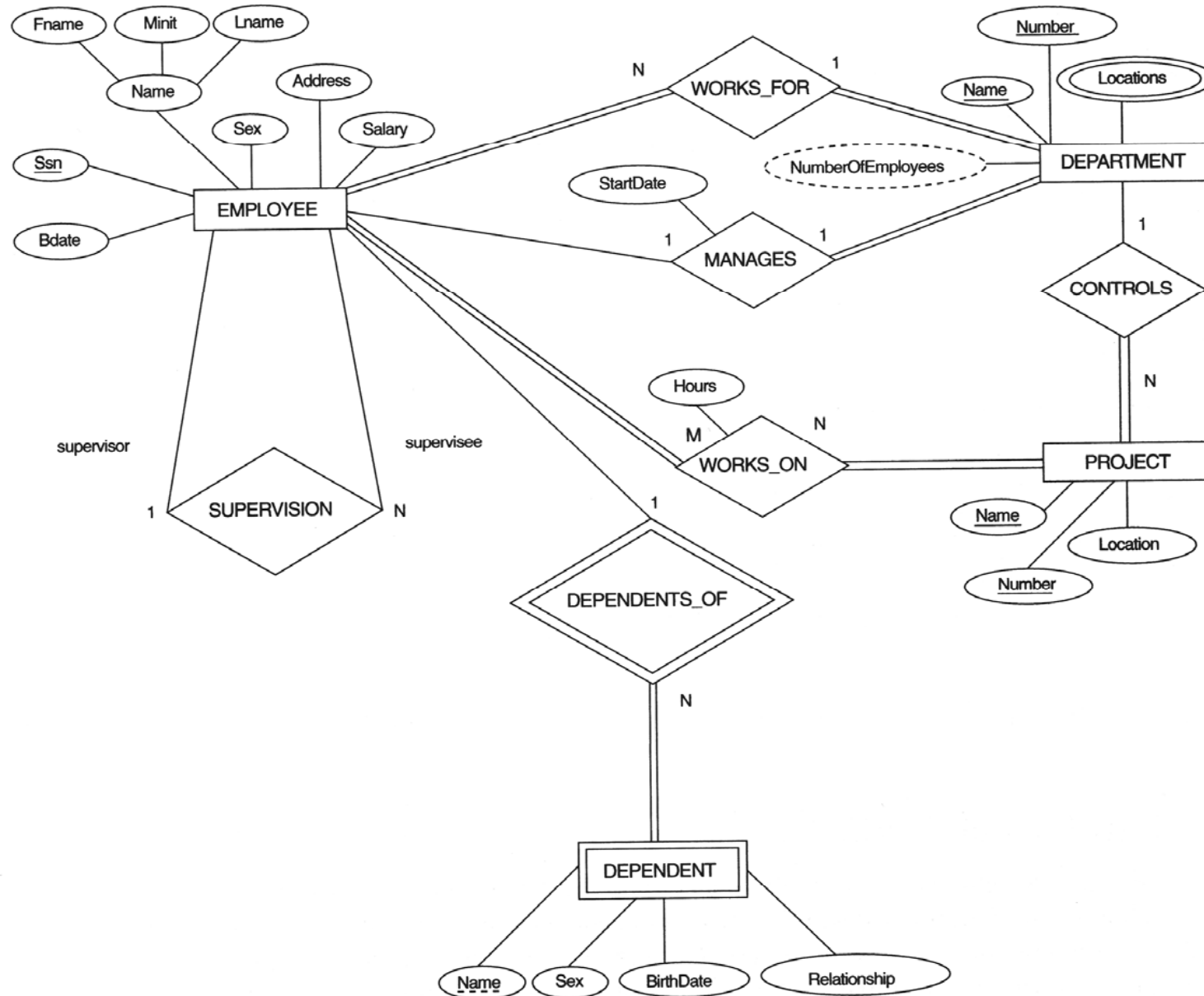
- Relationships and Relationship Types
  - More than one relationship type can exist with the same participating entity types. For example, MANAGES and WORKS\_FOR are distinct relationships between EMPLOYEE and DEPARTMENT, but with different meanings and different relationship instances

# Summary of the Notation for ER Diagrams



Symbol	Meaning
	ENTITY
	WEAK ENTITY
	RELATIONSHIP
	IDENTIFYING RELATIONSHIP
	ATTRIBUTE
	KEY ATTRIBUTE
	MULTIVALUED ATTRIBUTE
	COMPOSITE ATTRIBUTE
	DERIVED ATTRIBUTE
	TOTAL PARTICIPATION OF $E_2$ IN $R$
	CARDINALITY RATIO 1: $N$ FOR $E_1:E_2$ IN $R$
	STRUCTURAL CONSTRAINT (min, max) ON PARTICIPATION OF $E$ IN $R$

# Example COMPANY Database



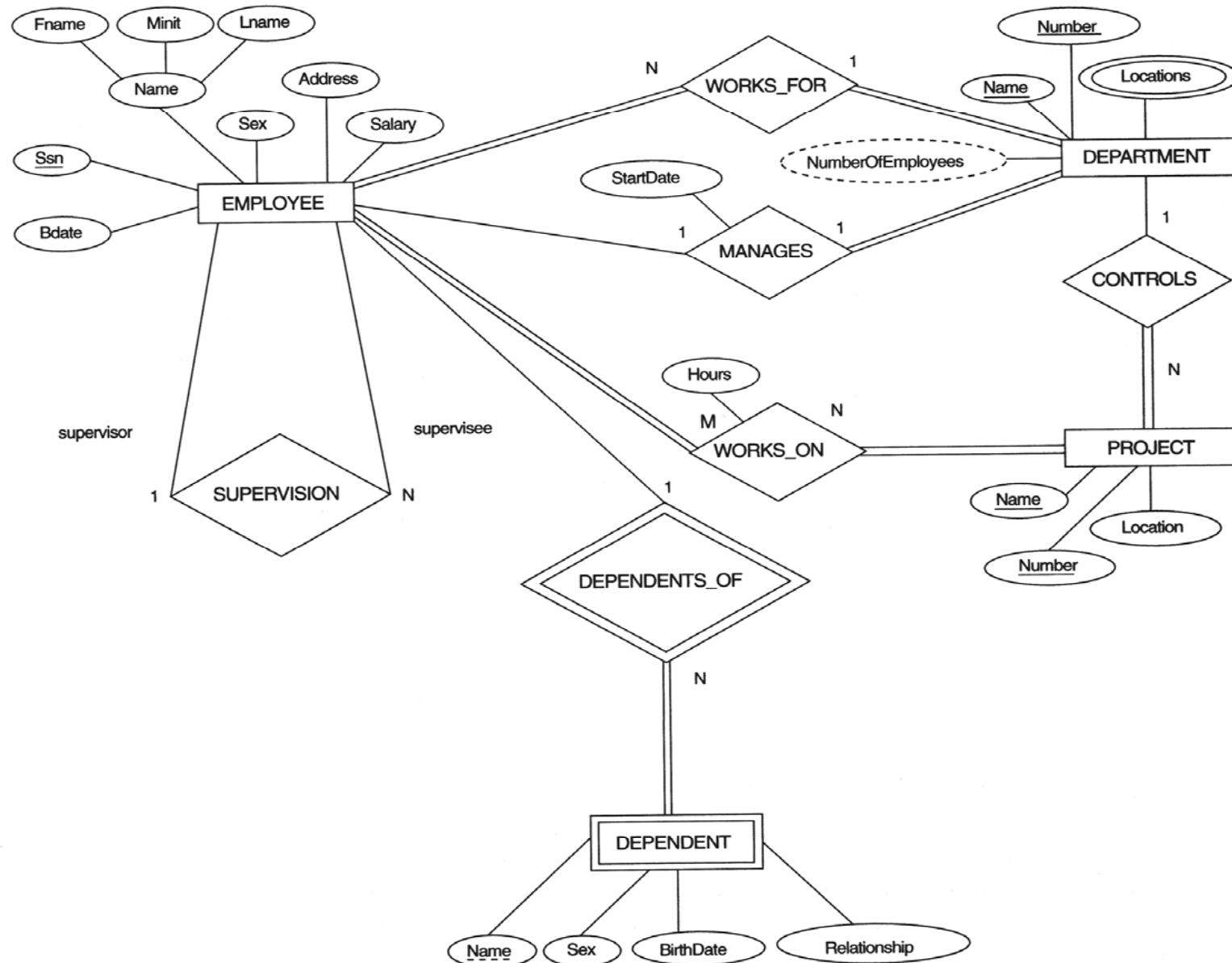
# ER Model Concepts

- Attributes of Relationship Types:
  - A relationship type can have attributes; for 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

# ER Model Concepts

- Weak Entity Types
  - An entity that does not have a key attribute
  - A weak entity must participate in an identifying relationship type with an owner or identifying entity type
  - 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
  - Example: Suppose that a DEPENDENT entity is identified by the dependent's first name (unique wrt. each EMPLOYEE), *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

# Example COMPANY Database



# ER Model Concepts

- **Structural constraints:** one way to express semantics of relationship: cardinality ratio and membership class
- **Cardinality ratio (functionality):** It specifies the number of relationship instances that an entity can participate in a **binary** relationship
  - one-to-one (1:1)
  - one-to-many (1:M) or many-to-one (M:1)
  - many-to-many (M:N)
- An example of a 1:1 binary relationship is MANAGES which relates a department entity to the employee who manages that department. This represents the miniworld constraints that an employee can manage only one department and that a department has only one manager
- Relationship types of degree 2 are called binary. Relationship types of degree 3 are called **ternary** and of degree n are called **n-ary**. In general, an n-ary relationship *is not* equivalent to n binary relationships (**reading suggestion !!**)



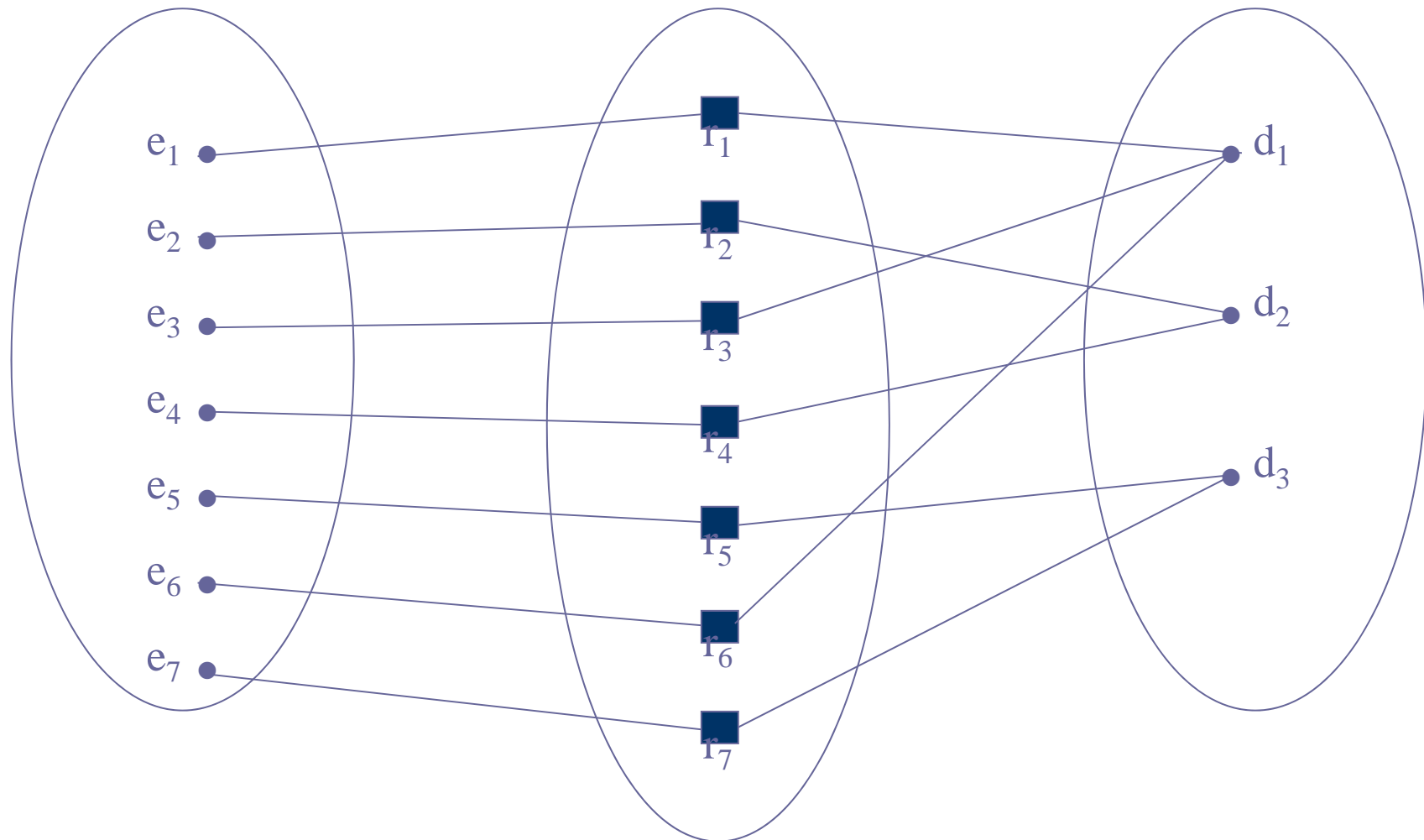
# One-to-many (1:N) or Many-to-one (N:1) RELATIONSHIP



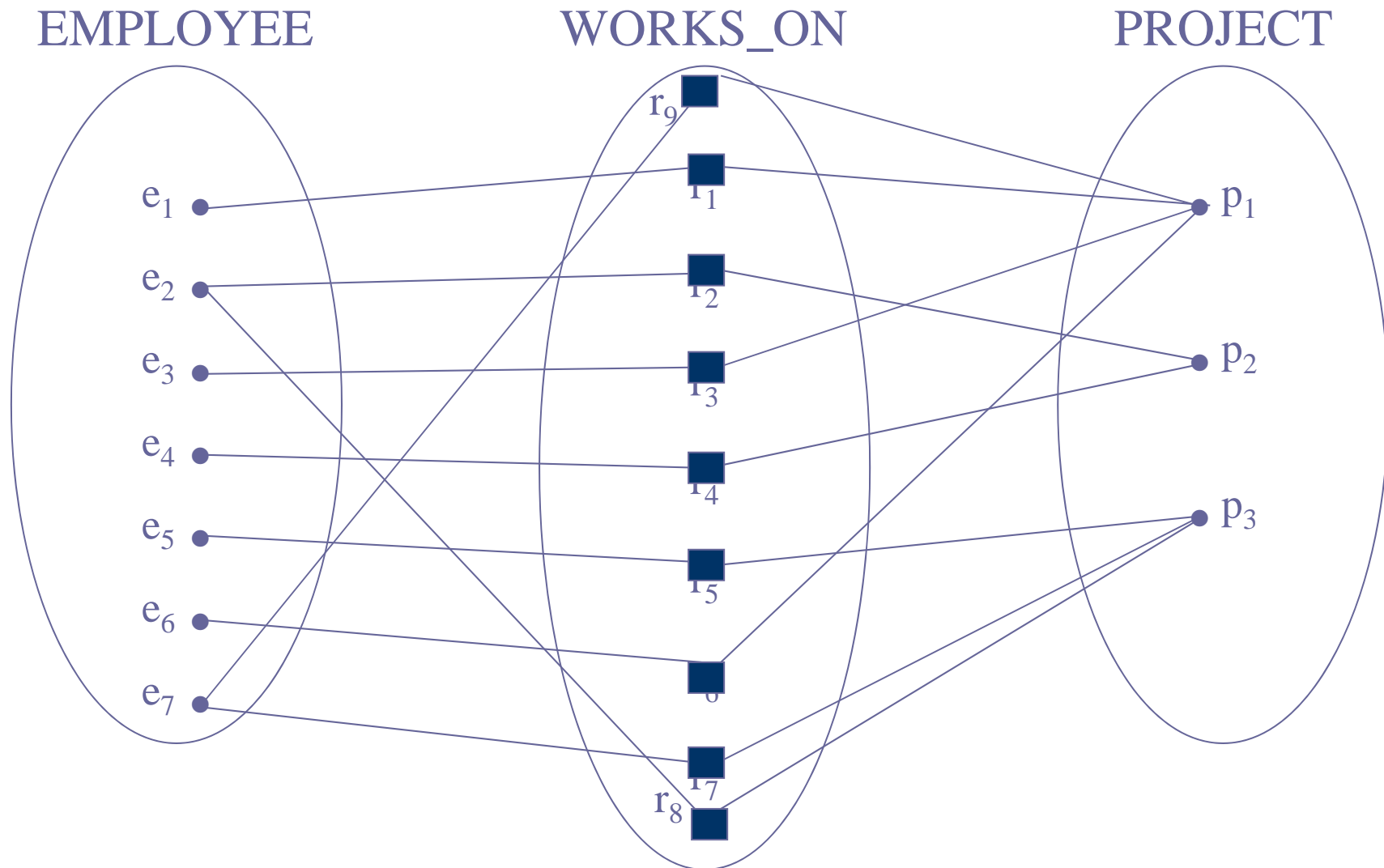
EMPLOYEE

WORKS\_FOR

DEPARTMENT



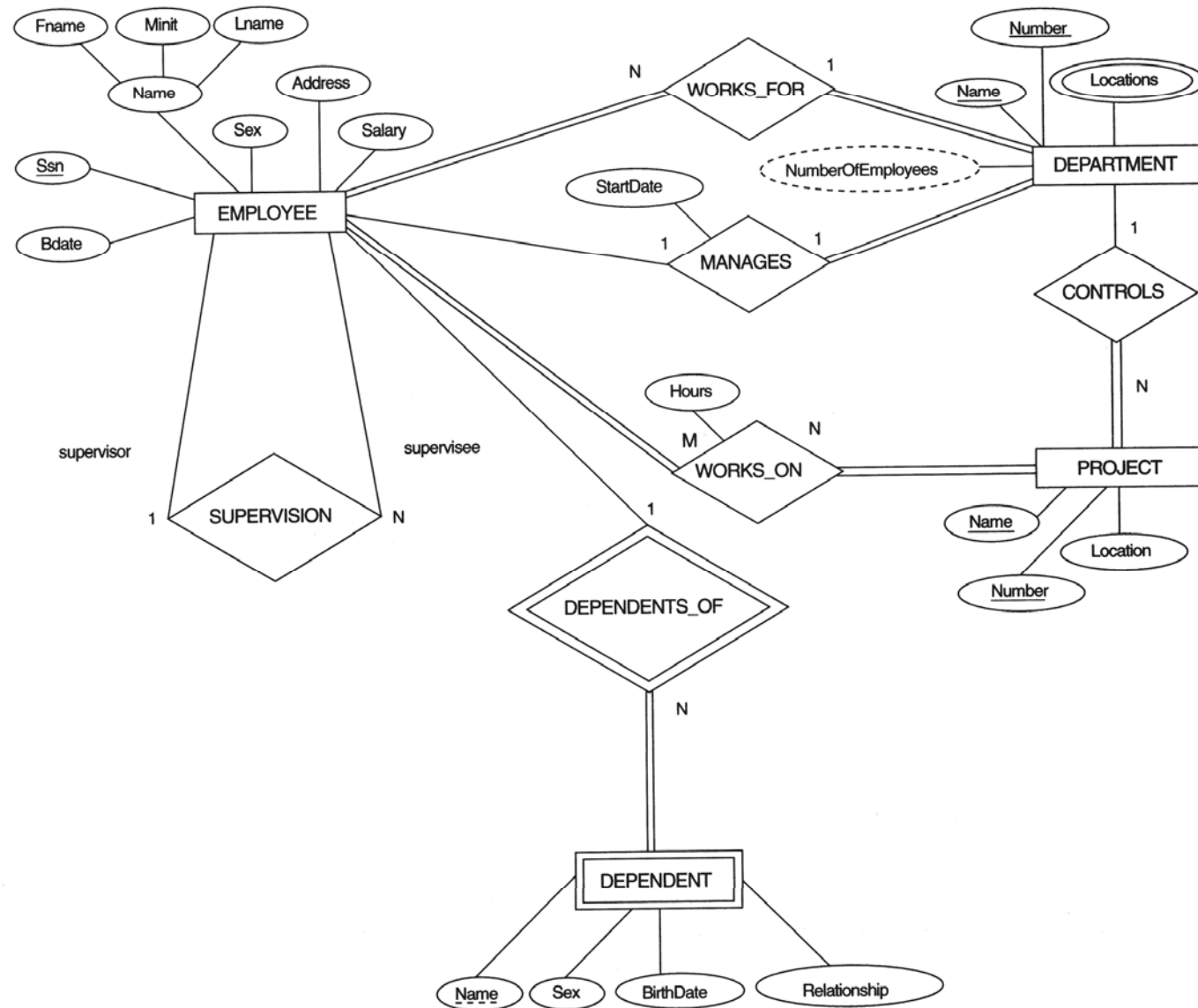
# Many-to-many (M:N) RELATIONSHIP



# ER Model Concepts

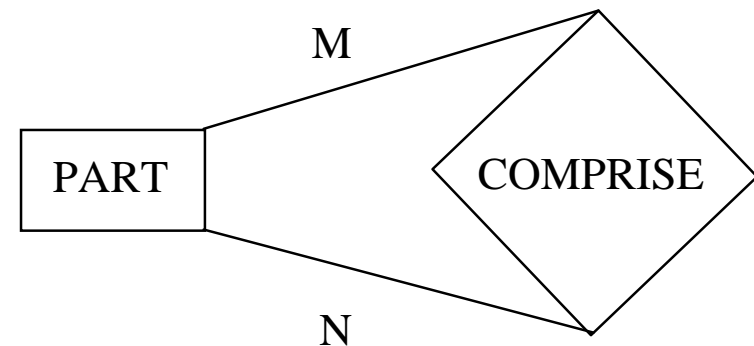
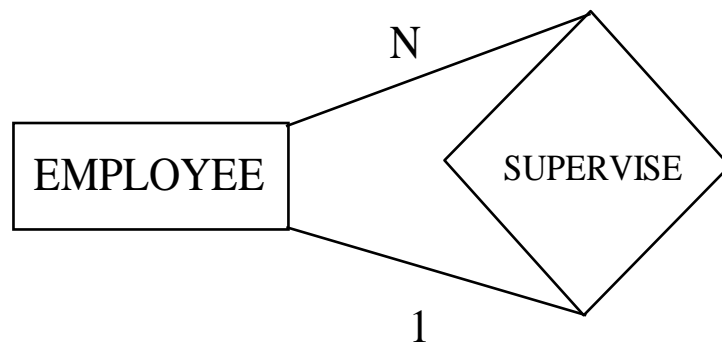
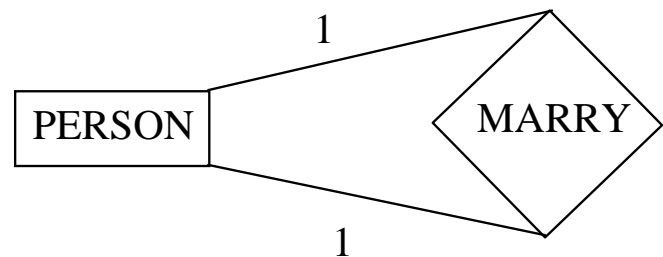
- **Membership class (participation constraint):**
  - **Mandatory (total participation)** - every instance of a participating entity type must participate in the relationship. Example: ATTEND relationship between STUDENTS and COURSE
  - **Optional (partial participation)** - not every instance of a participating entity type must participate in the relationship. Example: OFFER relationship between SCHOOL and MODULE is optional for SCHOOL but mandatory for MODULE
- **Notation:**
  - **Cardinality ratio** (of a binary relationship): 1:1, 1:N, N:1, or M:N  
**SHOWN BY PLACING APPROPRIATE NUMBER ON THE LINK**
  - **Participation constraint** (on each participating entity type): total (called existence dependency) or partial.  
**IN ER DIAGRAMS, TOTAL PARTICIPATION IS DISPLAYED AS A DOUBLE LINE CONNECTING THE PARTICIPATING ENTITY TYPE TO THE RELATIONSHIP, WHEREAS PARTIAL PARTICIPATION IS REPRESENTED BY A SINGLE LINE**

# Example COMPANY Database



# ER Model Concepts

- Recursive relationships (involved relationship): relationship among different instances of the same entity



# ER Model Concepts

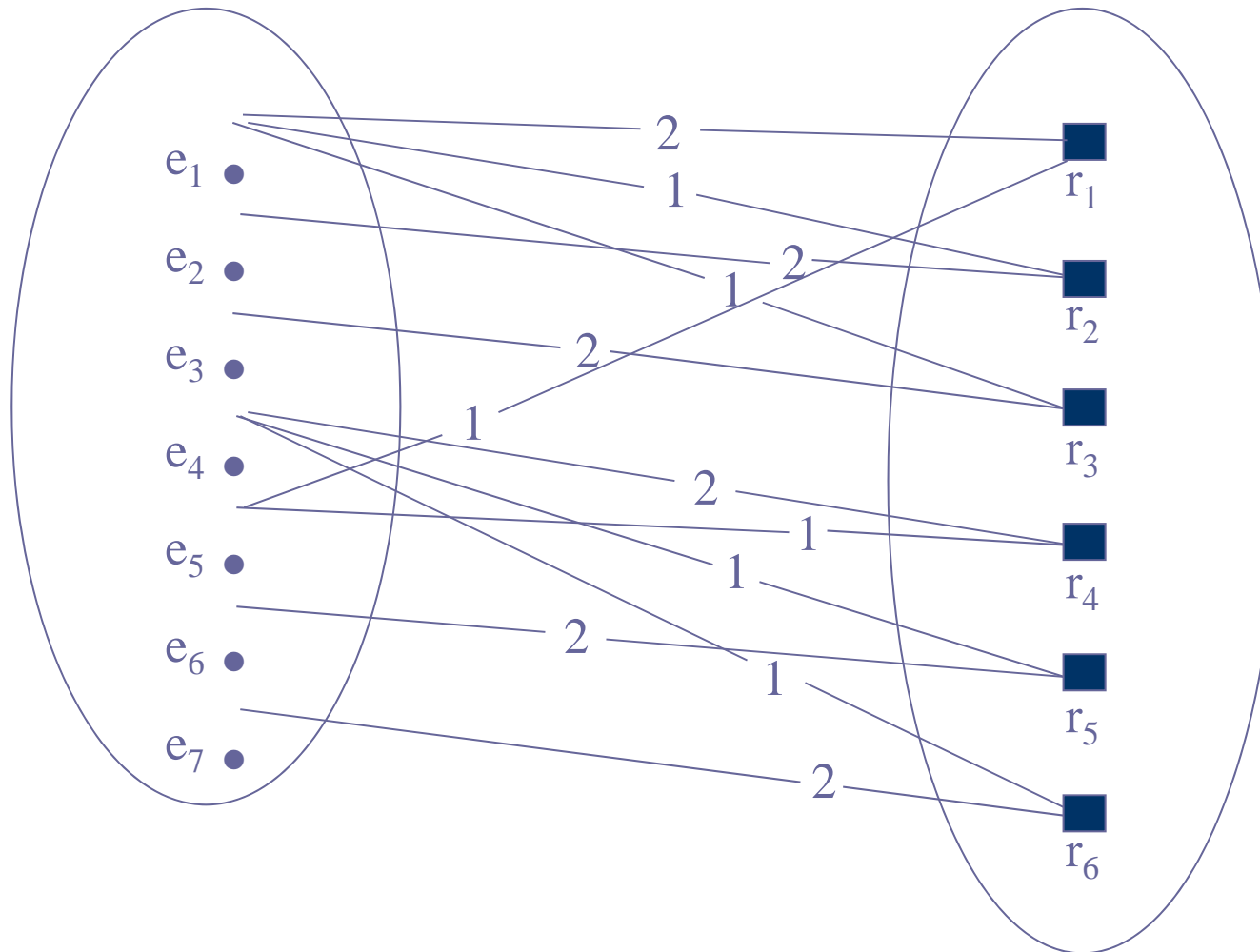
- Recursive relationships:
  - Both participations are same entity type in different roles
  - For example, SUPERVISION relationships between EMPLOYEE (in role of supervisor or boss) and (another) EMPLOYEE (in role of subordinate or worker)
  - In following figure, first role participation labeled with 1 and second role participation labeled with 2
  - In ER diagram, need to display role names to distinguish participations

# A Recursive Relationship SUPERVISION

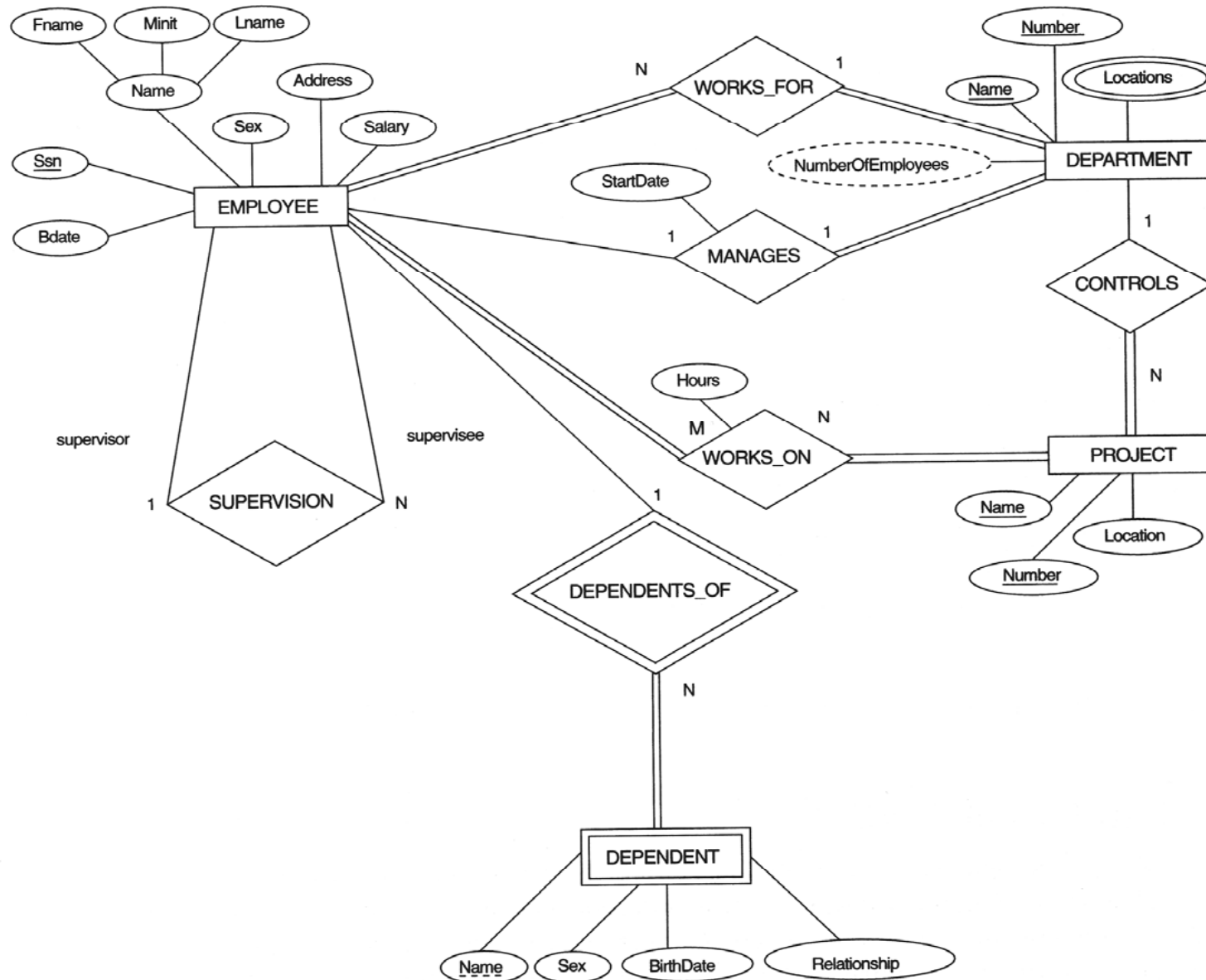


EMPLOYEE

SUPERVISION



# Example COMPANY Database





# ER Model

- 
- 
- 
- Alternative Diagrammatic Notations

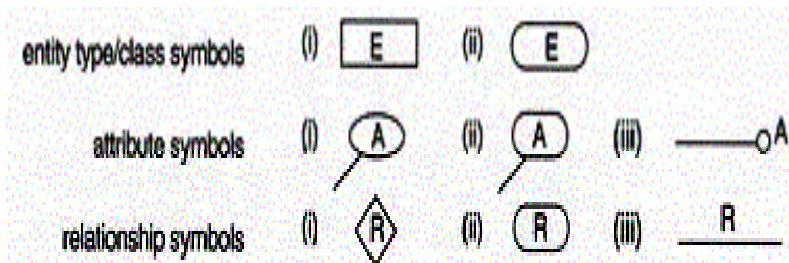
# Alternative Diagrammatic Notations

- Current use (in this class):
  - Chen notation
- Some others:
  - Crow's Feet notation
  - UML (Unified Modeling Language): Rational Rose
  - ...

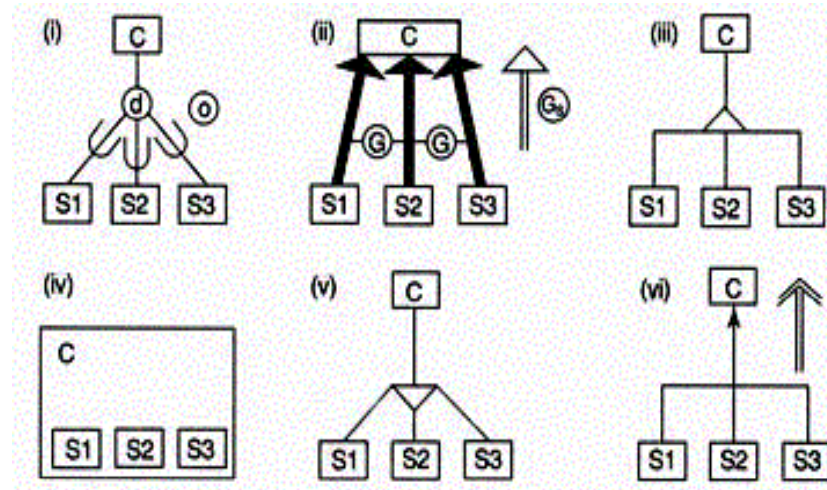
# Alternative Diagrammatic Notations



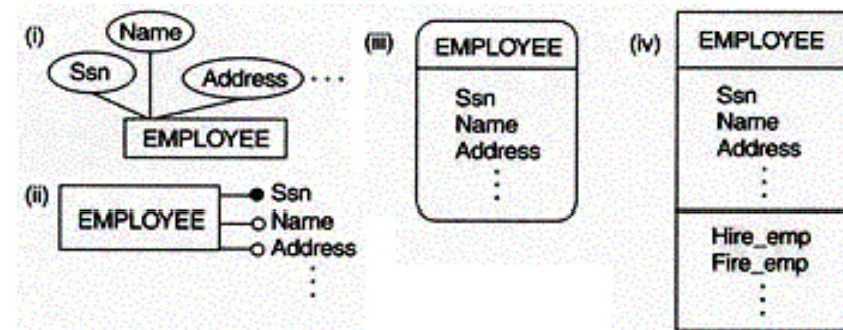
Symbols for entity type / class,  
attribute and relationship



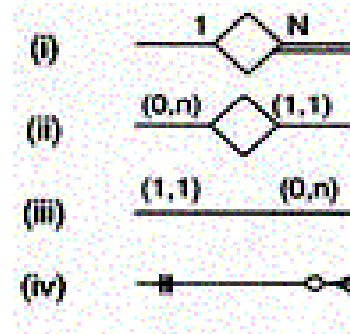
Notations for displaying  
specialization / generalization



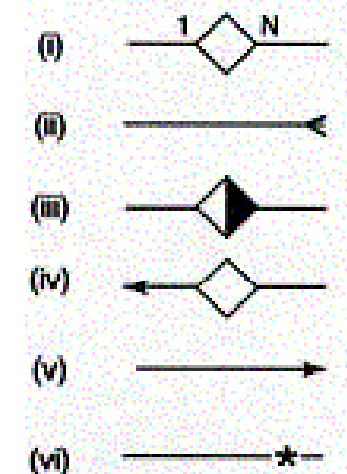
Displaying attributes



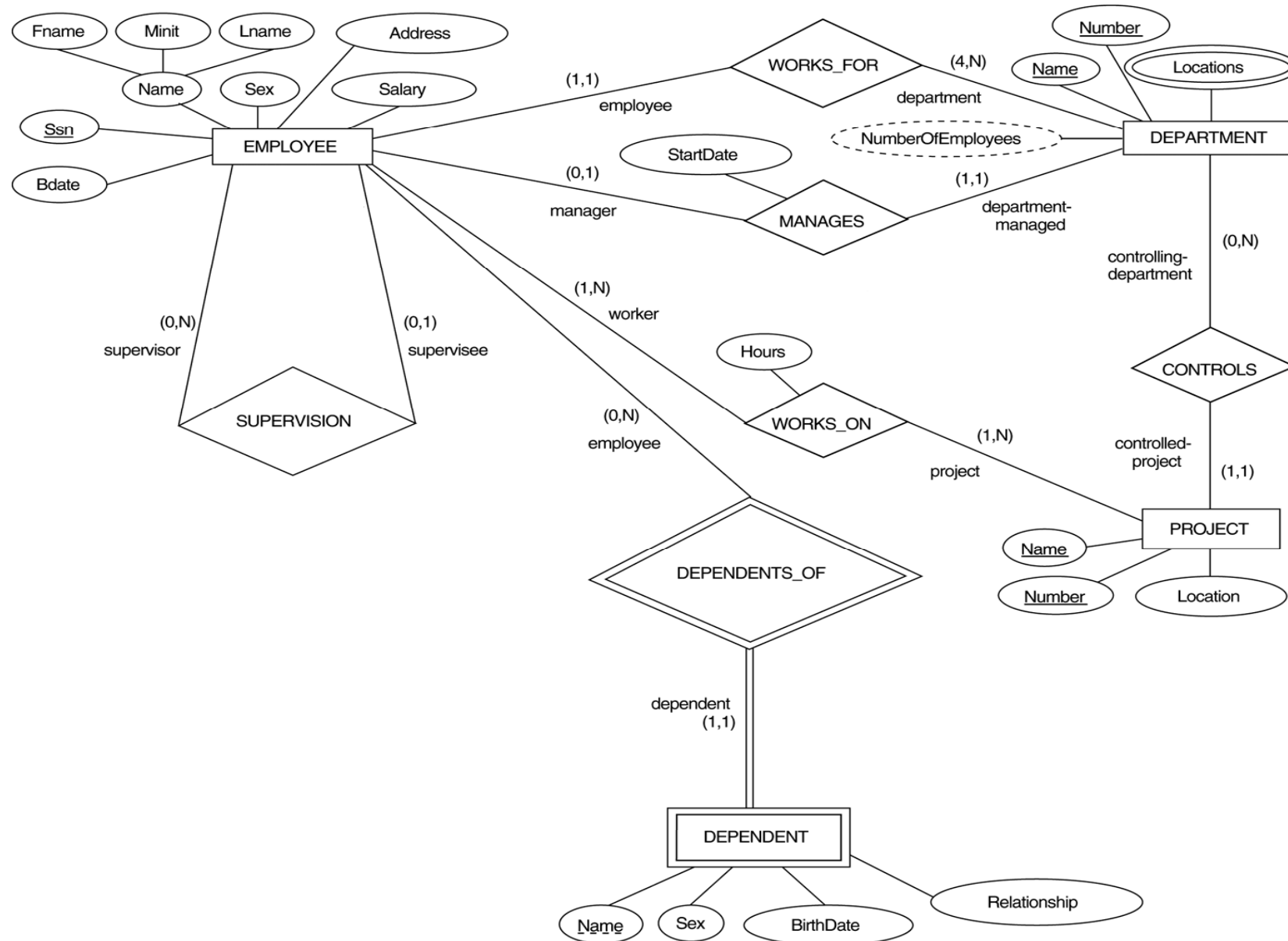
Various (min,  
max) notations



Displaying  
cardinality ratios



# ER diagrams for the COMPANY schema, with structural constraints specified using (min, max) notation



# Outline

- 
- 
- EER Model
- Discussion: Problems with ER Model
- Exercises
- Reading Suggestion:
  - [1]: Chapters 3, 4
  - [4]: Chapters 11, 12
  - A. Badia: "*Entity-Relationship Modeling Revisited*", SIGMOD Record, 33(1), March 2004, 77-82

# EER Model

- Limitations of Basic Concepts of the ER Model
- Enhanced-ER (EER) Model Concepts
- Subclasses and Superclasses
- Specialization and Generalization
- Specialization / Generalization Hierarchies, Lattices and Shared Subclasses
- Categories
- Formal Definitions of EER Model
- Database Design Modeling Tools

# Limitations of Basic Concepts of the ER 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 (see [2]: chapter 25 for more details)
- Response is development of additional 'semantic' modeling concepts

# Enhanced-ER Model Concepts

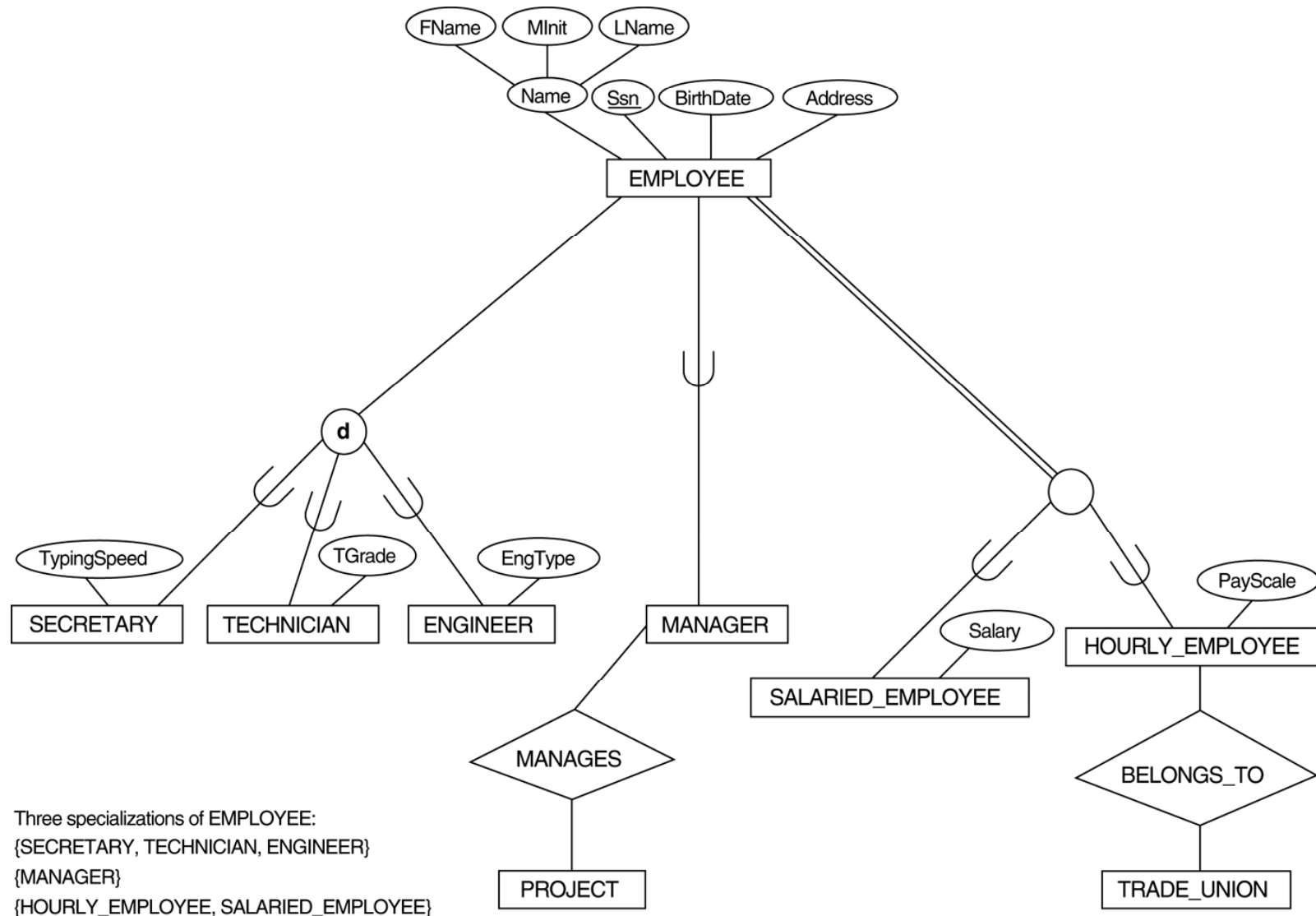
- Includes all modeling concepts of basic ER
- Additional concepts: subclasses/superclasses, specialization/generalization, categories, attribute inheritance
- The resulting model is called the enhanced-ER or Extended ER (E2R or EER) model
- It is used to model applications more completely and accurately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



# Subclasses and Superclasses

- An entity type may have additional meaningful subgroups of its entities
- Example: EMPLOYEE may be further grouped into SECRETARY, ENGINEER, MANAGER, TECHNICIAN, SALARIED\_EMPLOYEE, HOURLY\_EMPLOYEE, ...
  - Each of these groups 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

# EER diagram notation to represent subclasses and specialization



# Subclasses and Superclasses

- These are also called IS-A (IS-AN) relationships (SECRETARY IS-A EMPLOYEE, TECHNICIAN IS-A EMPLOYEE, ...).
- Note: An entity that is a 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
  - **Superclass/subclass relationship is one-to-one (1:1)**

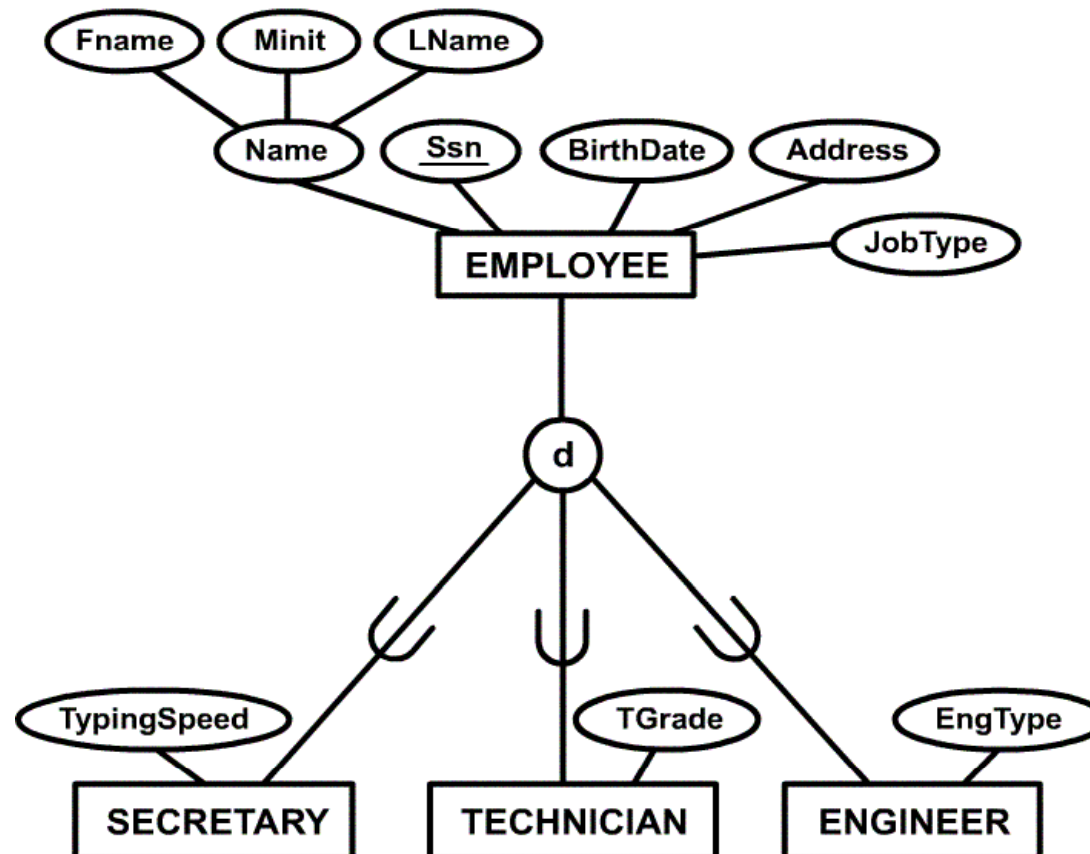
## Attribute Inheritance in Superclass/Subclass Relationships

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

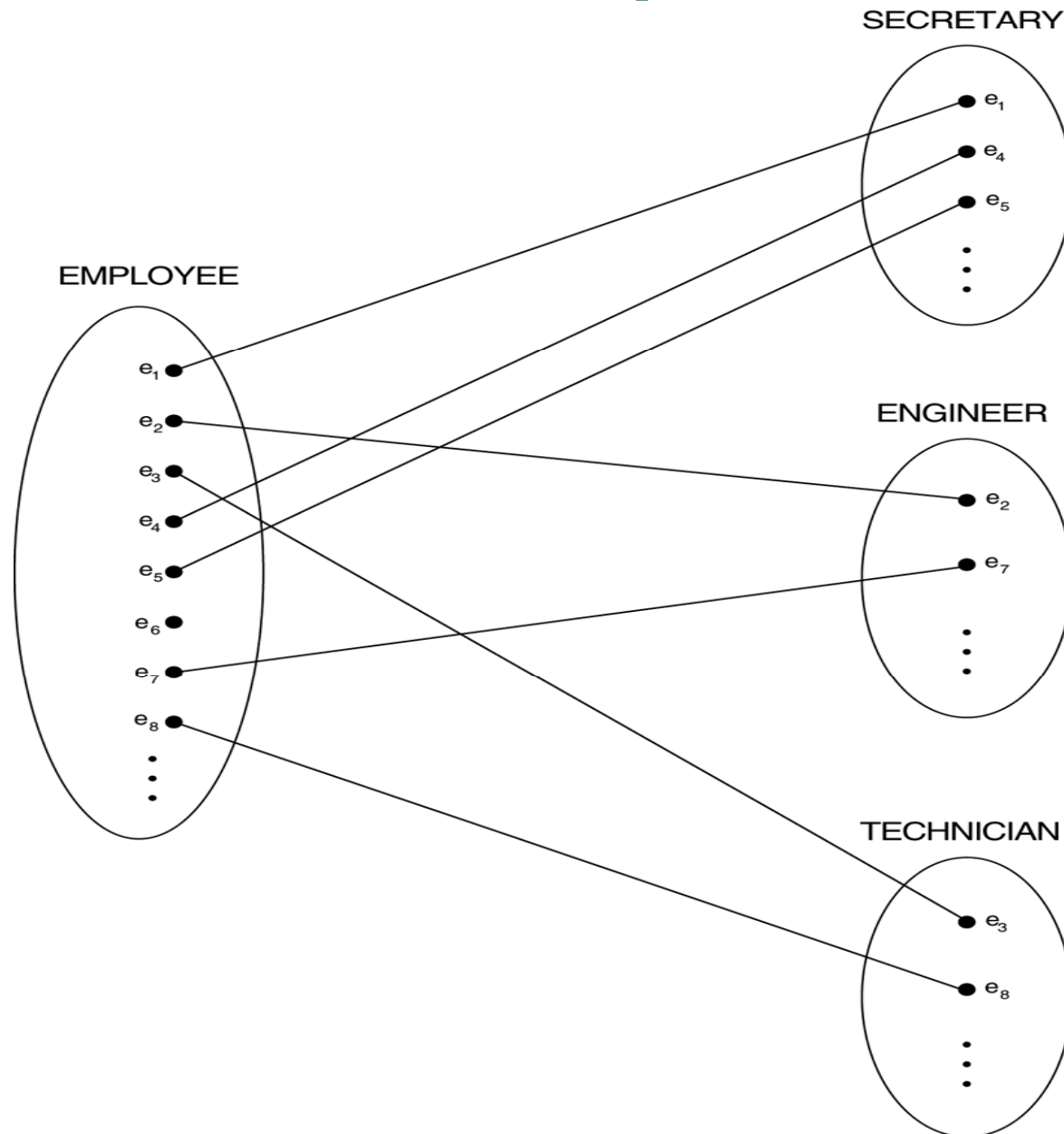
# Specialization

- 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 on the *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



# Instances of a specialization

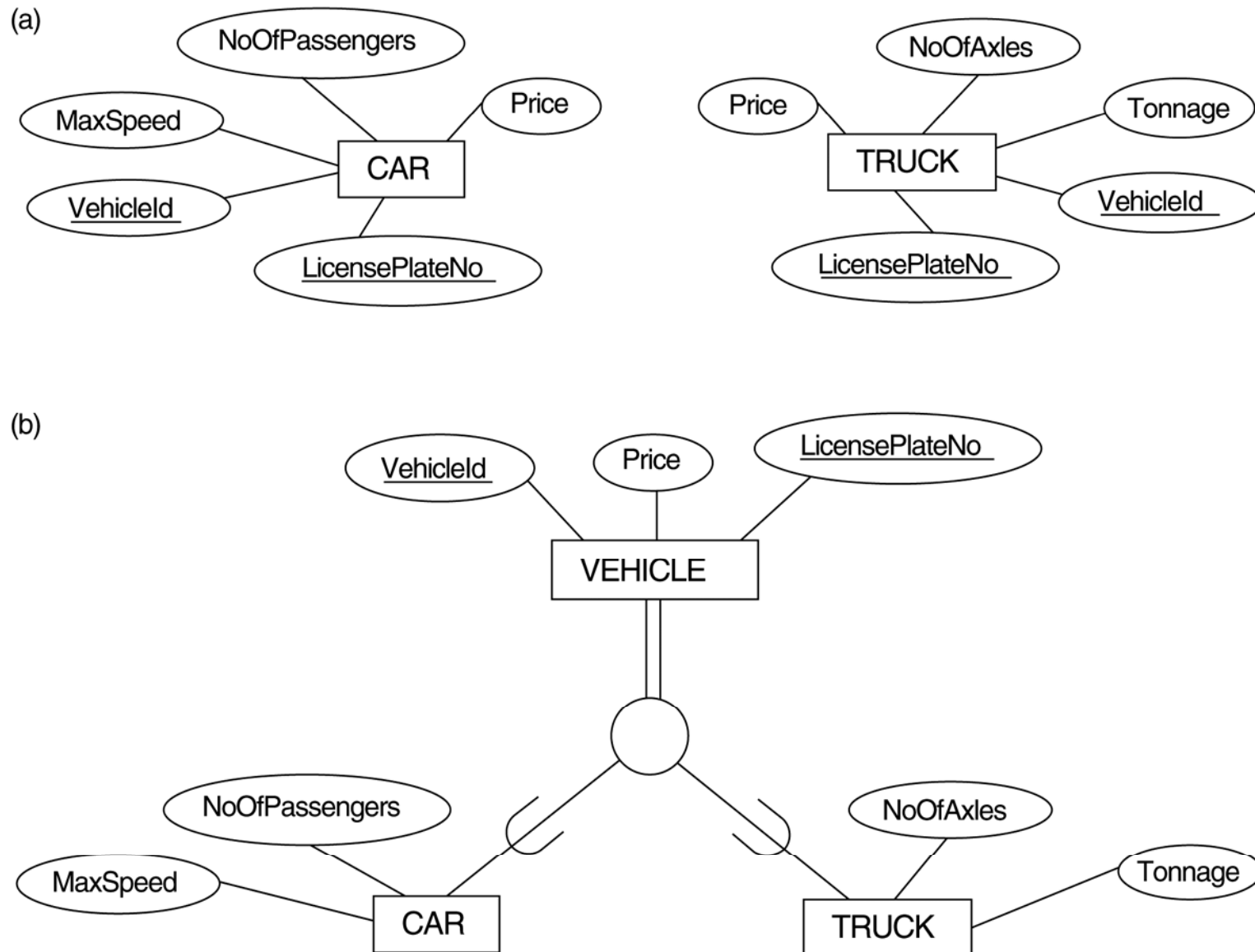


# Generalization

- 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 Example



# Specialization and Generalization

- Diagrammatic notation sometimes used to distinguish between 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
- Data Modeling with Specialization and Generalization
  - 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

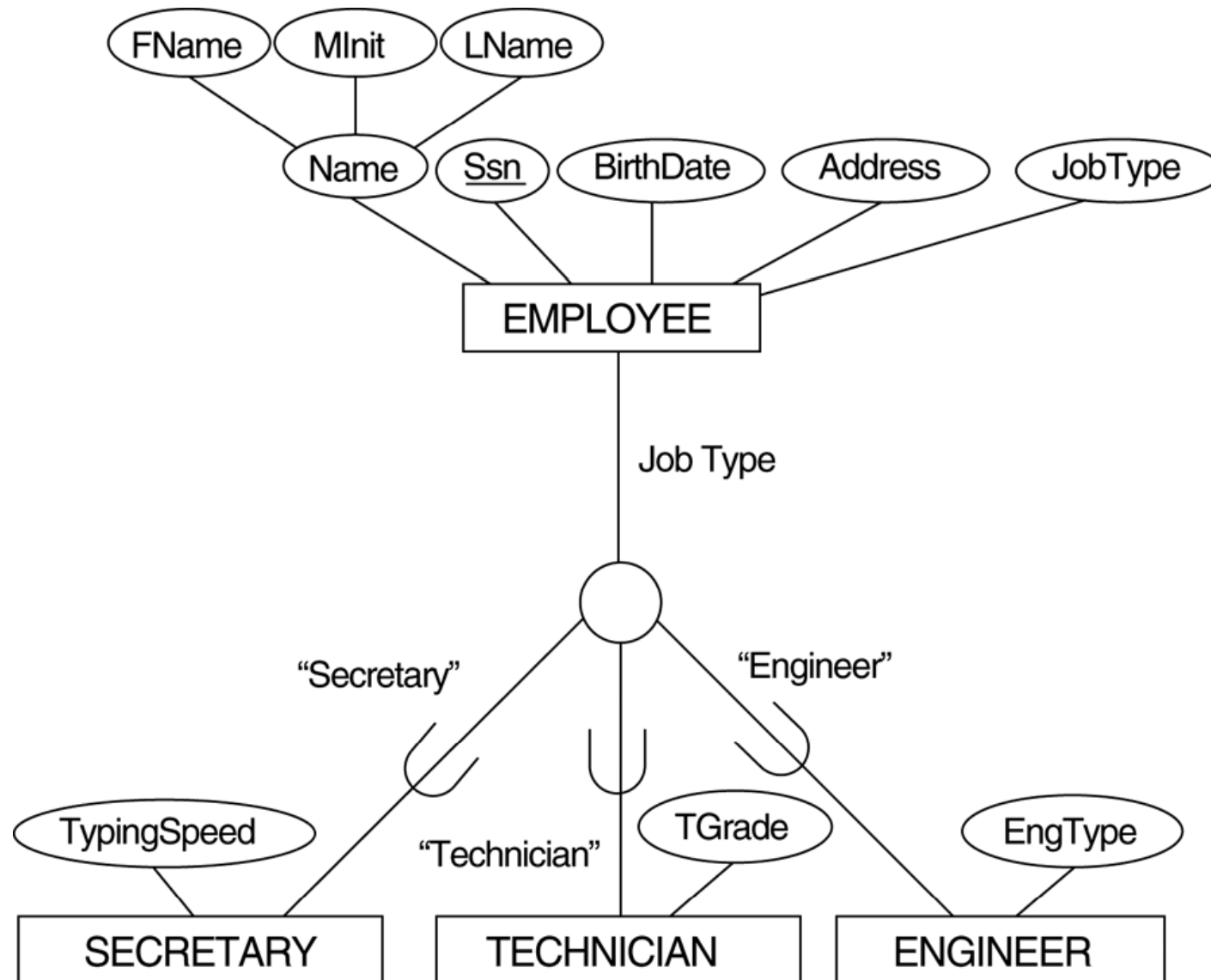
# Constraints on Specialization and Generalization

- 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

# Constraints on Specialization and Generalization

- 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

# EER diagram notation for an attribute-defined specialization on JobType



# Constraints on Specialization and Generalization

- 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

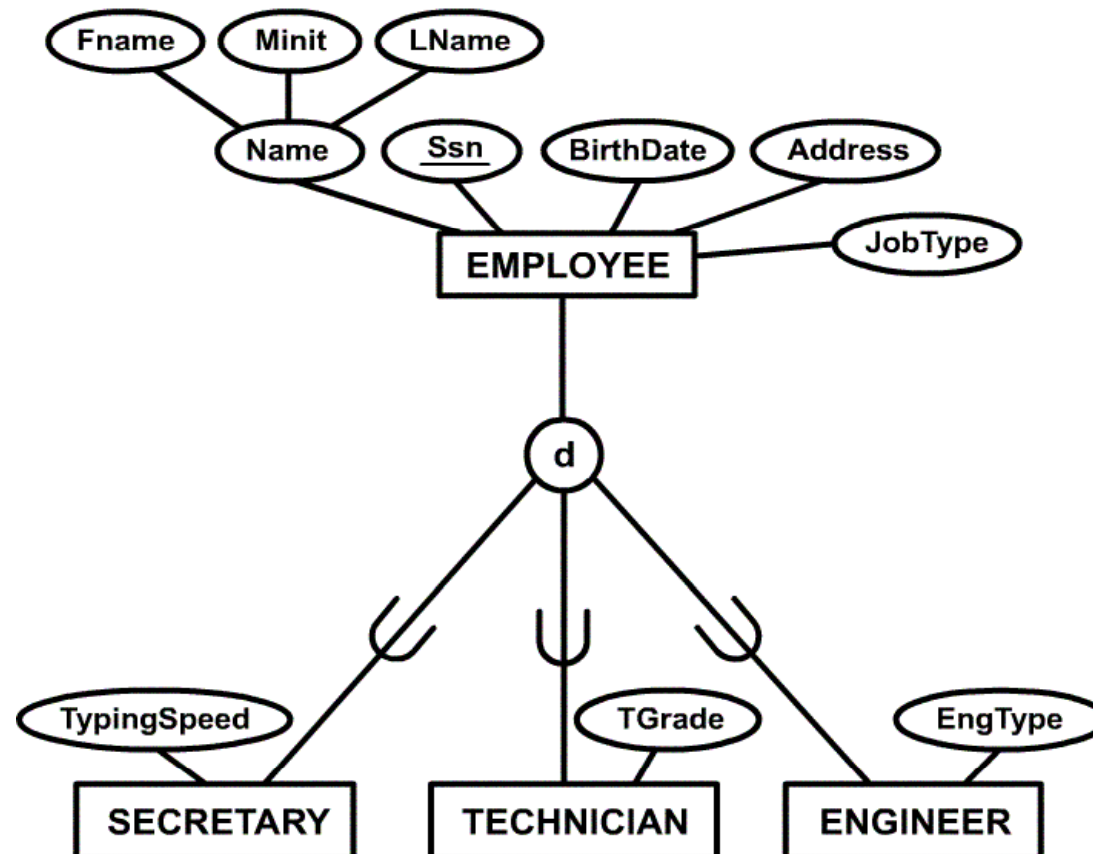
- Two other conditions apply to a specialization/generalization: disjointness and completeness constraints
- **Disjointness Constraint:**
  - Specifies that the subclasses of the specialization must be disjointed (an entity can be a member of at most one of the subclasses of the specialization)
  - Specified 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

# Constraints on Specialization and Generalization

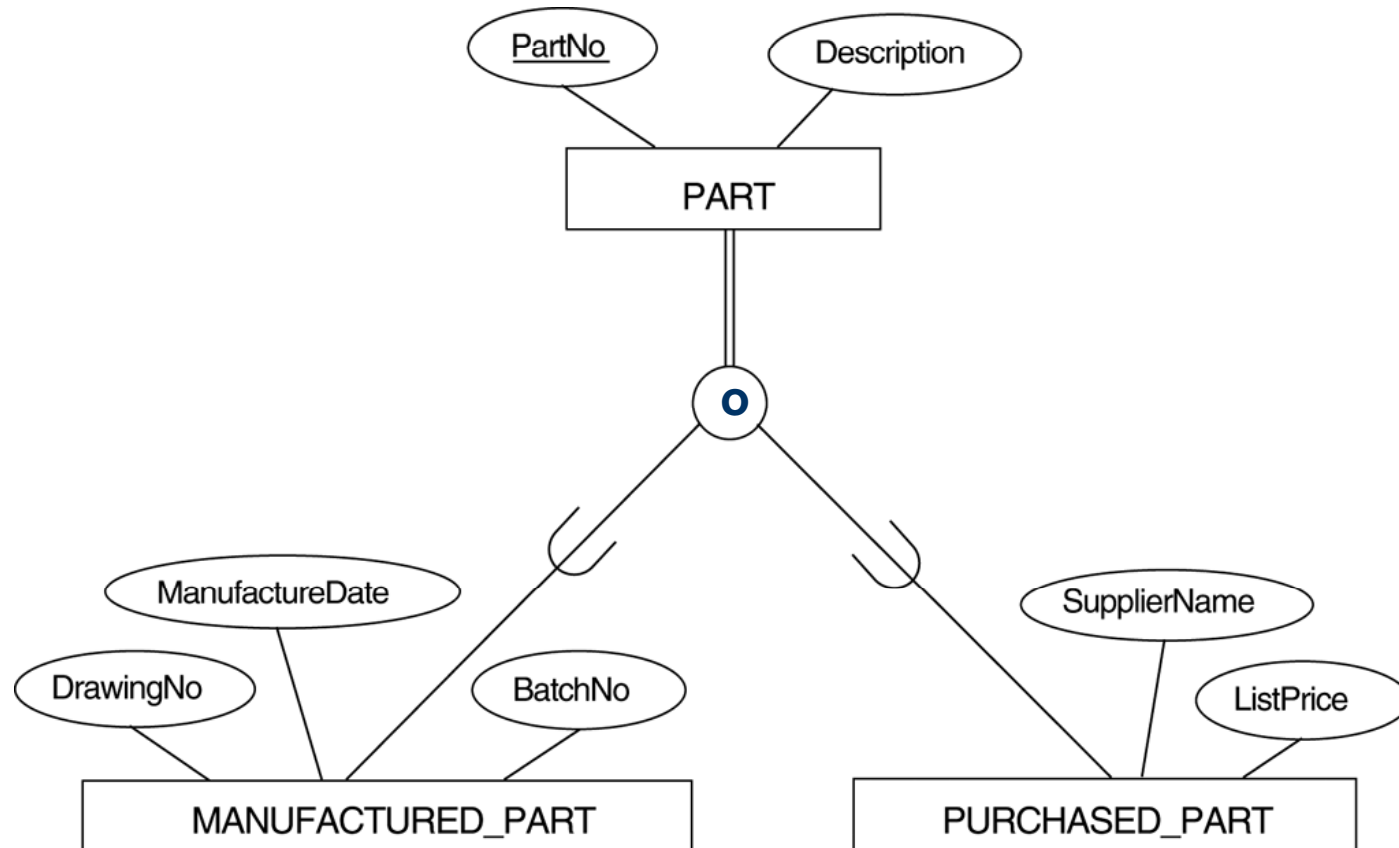
- **Completeness Constraint:**
  - Total specifies that every entity in the superclass must be a member of some subclass in the specialization/ generalization
  - Shown in EER diagrams by a double line
  - Partial allows an entity not to belong to any of the subclasses
  - Shown in EER diagrams by a single line



# Example of Disjoint Partial Specialization



# Example of Overlapping Total Specialization



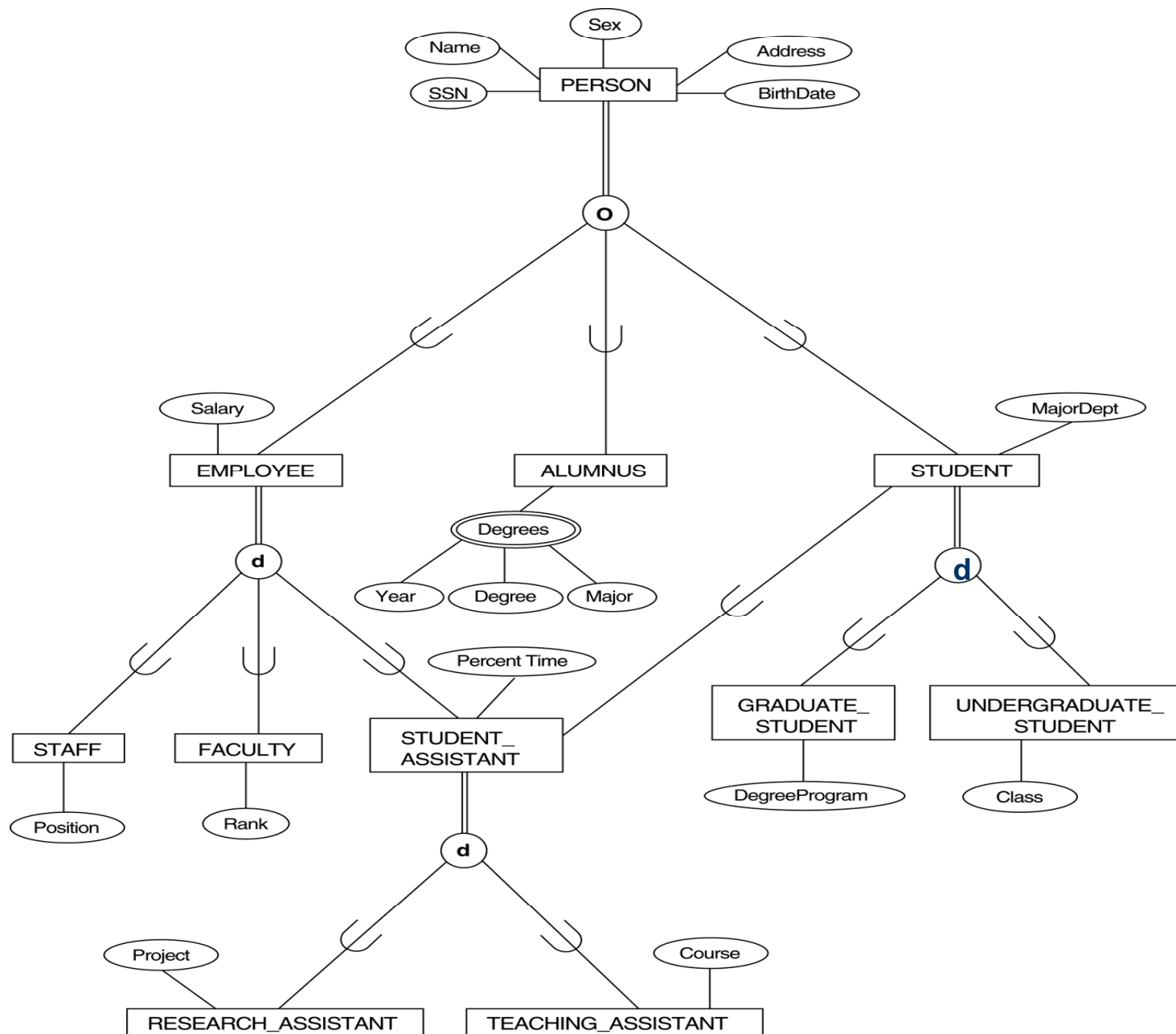
# Constraints on Specialization and Generalization

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

## Specialization / Generalization Hierarchies, Lattices and Shared Subclasses

- A subclass may itself have further subclasses specified on it, forming 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

# Specialization / Generalization Lattice Example (UNIVERSITY)



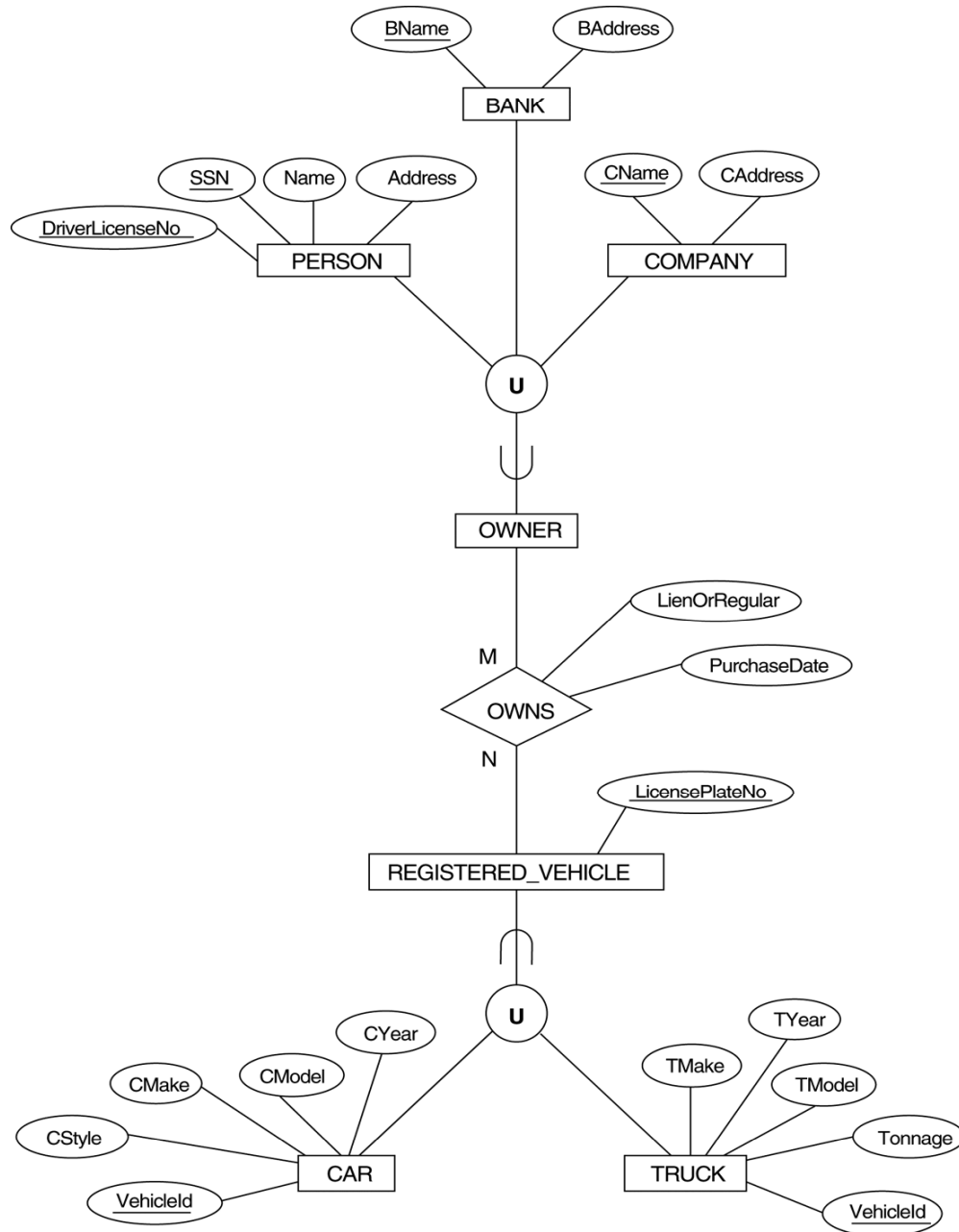
# Categories

- 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

# Categories

- 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 a subset of the intersection of its superclasses (shared subclass member must exist in all of its superclasses)

Two categories (union types):  
**OWNER** and  
**REGISTERED\_VEHICLE**





# Formal Definitions of EER Model

- Class C: A set of entities; could be entity type, subclass, superclass, category.
- Subclass S: A class whose entities must always be subset of the entities in another class, called the superclass C of the superclass/subclass (or IS-A) relationship S/C:
$$S \subseteq C$$
- Specialization Z:  $Z = \{S_1, S_2, \dots, S_n\}$  is a set of subclasses with the same superclass G; hence, G/S<sub>i</sub> is a superclass/subclass relationship for  $i = 1, \dots, n$ 
  - G is called a generalization of the subclasses  $\{S_1, S_2, \dots, S_n\}$
  - Z is total if we always have:
$$S_1 \cup S_2 \cup \dots \cup S_n = G;$$
Otherwise, Z is partial.
  - Z is disjoint if we always have:
$$S_i \cap S_j = \emptyset \text{ (empty-set) for } i \neq j;$$
Otherwise, Z is overlapping.

# Formal Definitions of EER Model

- Subclass  $S$  of  $C$  is predicate-defined if predicate  $p$  on attributes of  $C$  is used to specify membership in  $S$ ; that is,  $S = C[p]$ , where  $C[p]$  is the set of entities in  $C$  that satisfy  $p$
- A subclass not defined by a predicate is called user-defined
- Attribute-defined specialization: if a predicate  $A = c_i$  (where  $A$  is an attribute of  $G$  and  $c_i$  is a constant value from the domain of  $A$ ) is used to specify membership in each subclass  $S_i$  in  $Z$
- Note: If  $c_i \neq c_j$  for  $i \neq j$ , and  $A$  is single-valued, then the attribute-defined specialization will be disjoint.
- Category or UNION type  $T$ 
  - A class that is a subset of the union of  $n$  defining superclasses  $D_1, D_2, \dots, D_n$ ,  $n > 1$ :  
$$T \subseteq (D_1 \cup D_2 \cup \dots \cup D_n)$$
  
A predicate  $p_i$  on the attributes of  $T$ .
  - A predicate  $p_i$  on the attributes of  $D_i$  can specify entities of  $D_i$  that are members of  $T$ .
  - If a predicate is specified on every  $D_i$ :  $T = (D_1[p_1] \cup D_2[p_2] \cup \dots \cup D_n[p_n])$
  - Note: The definition of relationship type should have 'entity type' replaced with 'class'.

# Database Design Modeling Tools



COMPANY	TOOL	FUNCTIONALITY
Embarcadero Technologies	ER Studio	Database Modeling in ER and IDEF1X
	DB Artisan	Database administration and space and security management
Oracle	Developer 2000 and Designer 2000	Database modeling, application development
Popkin Software	System Architect 2001	Data modeling, object modeling, process modeling, structured analysis/design
Platinum Technology	Platinum Enterprise Modeling Suite: Erwin, BPWin, Paradigm Plus	Data, process, and business component modeling
Persistence Inc.	Pwertier	Mapping from O-O to relational model
Rational	Rational Rose	Modeling in UML and application generation in C++ and JAVA
Rogue Ware	RW Metro	Mapping from O-O to relational model
Resolution Ltd.	Xcase	Conceptual modeling up to code maintenance
Sybase	Enterprise Application Suite	Data modeling, business logic modeling
Visio	Visio Enterprise	Data modeling, design and reengineering Visual Basic and Visual C++
...		

# Outline

- 
- 
- 
- Discussion: Problems with ER Model
- Exercises
- Reading Suggestion:
  - [1]: Chapters 3, 4
  - [4]: Chapters 11, 12
  - A. Badia: "*Entity-Relationship Modeling Revisited*", SIGMOD Record, 33(1), March 2004, 77-82

## Discussion: Problems with ER Model

- A. Badia: "*Entity-Relationship Modeling Revisited*", SIGMOD Record, 33(1), March 2004, 77-82
- [4]: Chapters 11, 12
- Some: semantic constraints, transition constraints, traps, ...

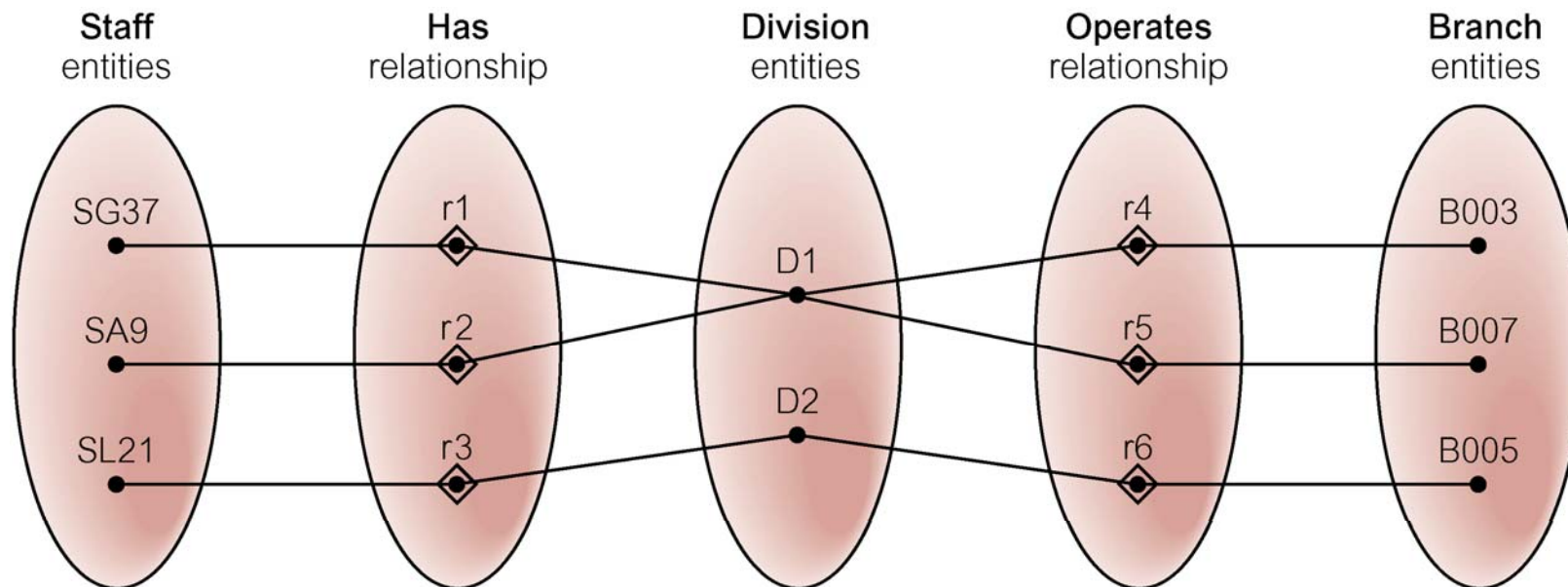
# Problems with ER Models

- Problems may arise when designing a conceptual data model called *connection traps*
- Often due to a misinterpretation of the meaning of certain relationships
- Two main types of connection traps are called *fan traps* and *chasm traps*

# Problems with ER Models

- Fan Trap
  - Where a model represents a relationship between entity types, but pathway between certain entity occurrences is ambiguous
  - Usually: two or more 1:N relationships fan out from the same entity
- Chasm Trap
  - Where a model suggests the existence of a relationship between entity types, but pathway does not exist between certain entity occurrences
  - Usually: optional participation

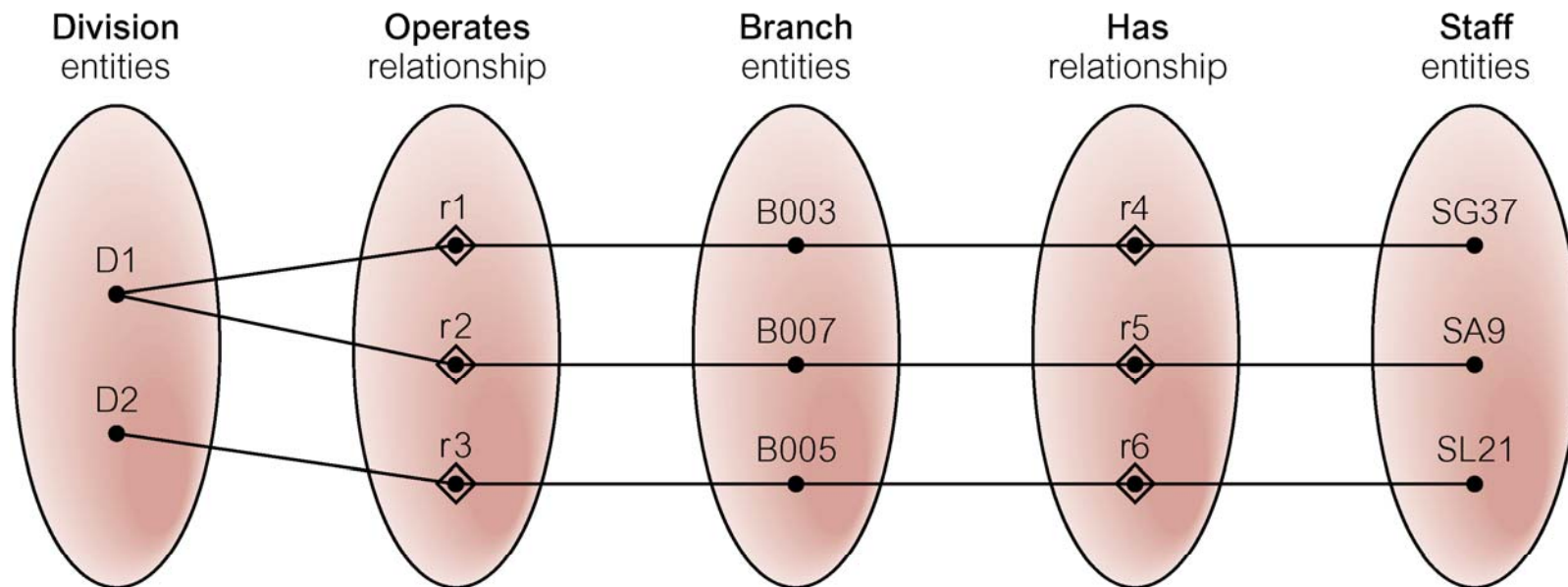
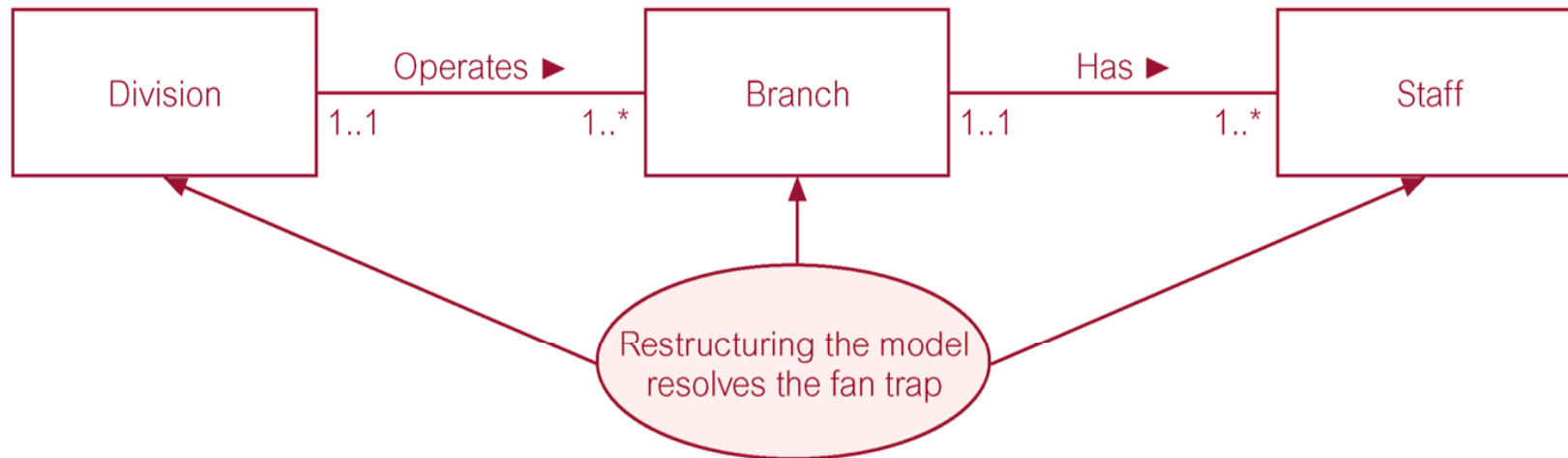
# An Example of a Fan Trap



**At which branch office does staff number SG37 work?**

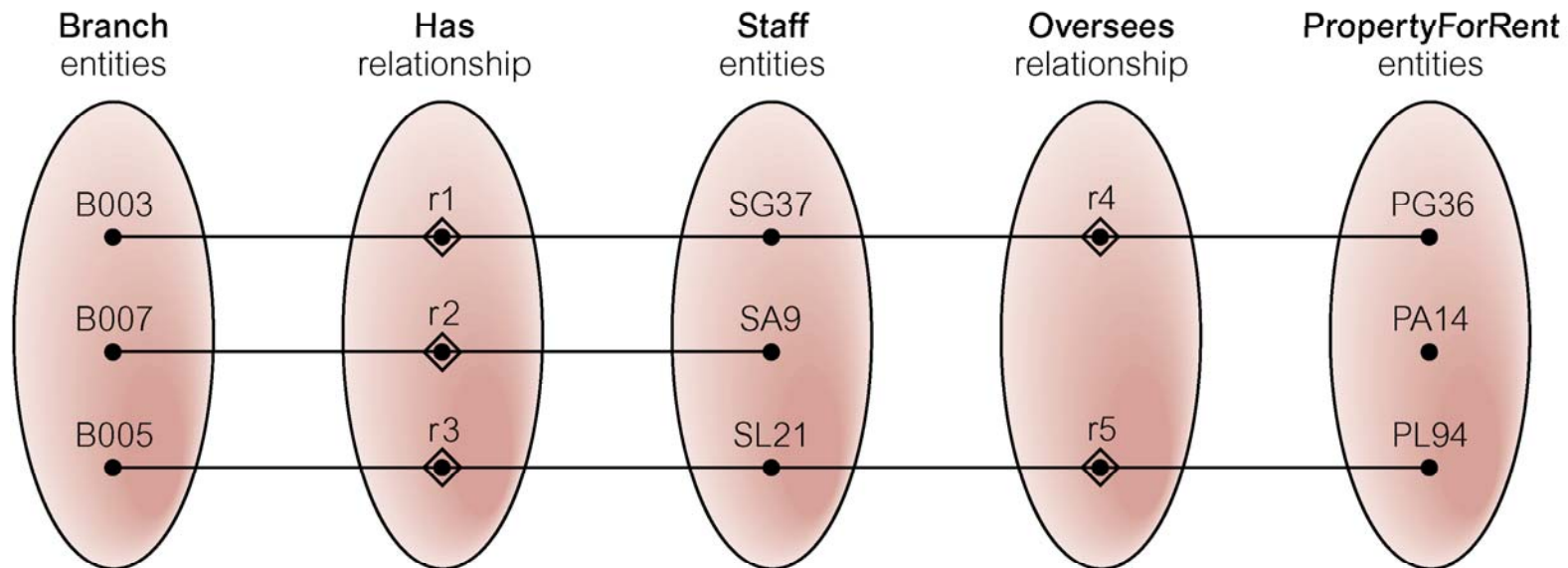


# Restructuring ER model to remove Fan Trap



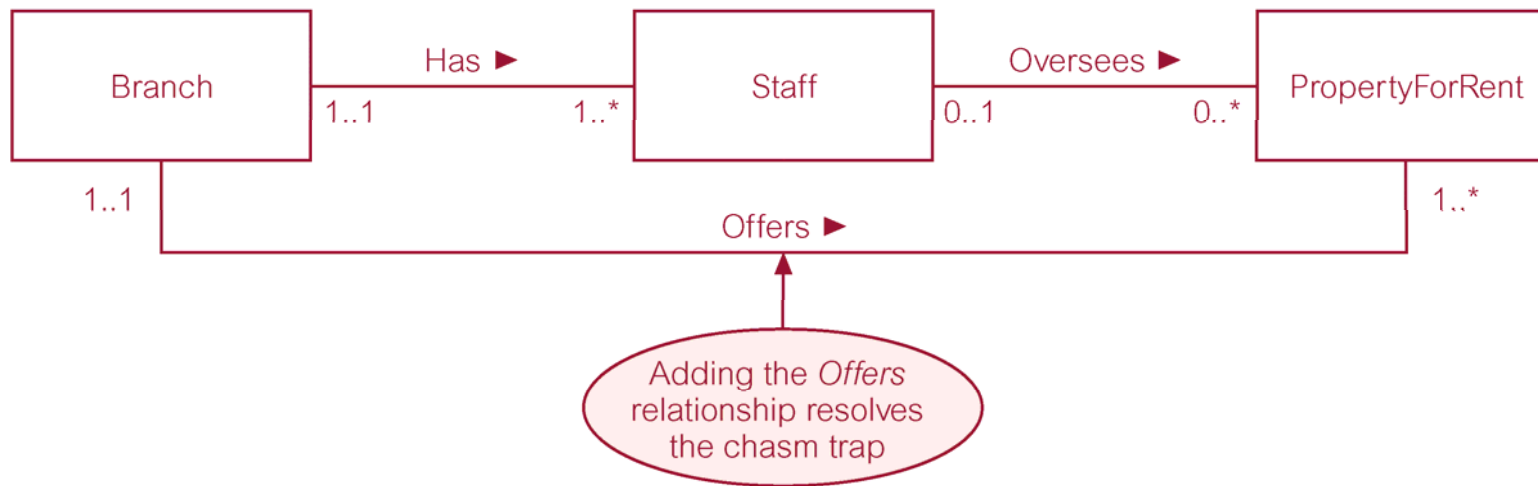
**SG37 works at branch B003**

# An Example of a Chasm Trap

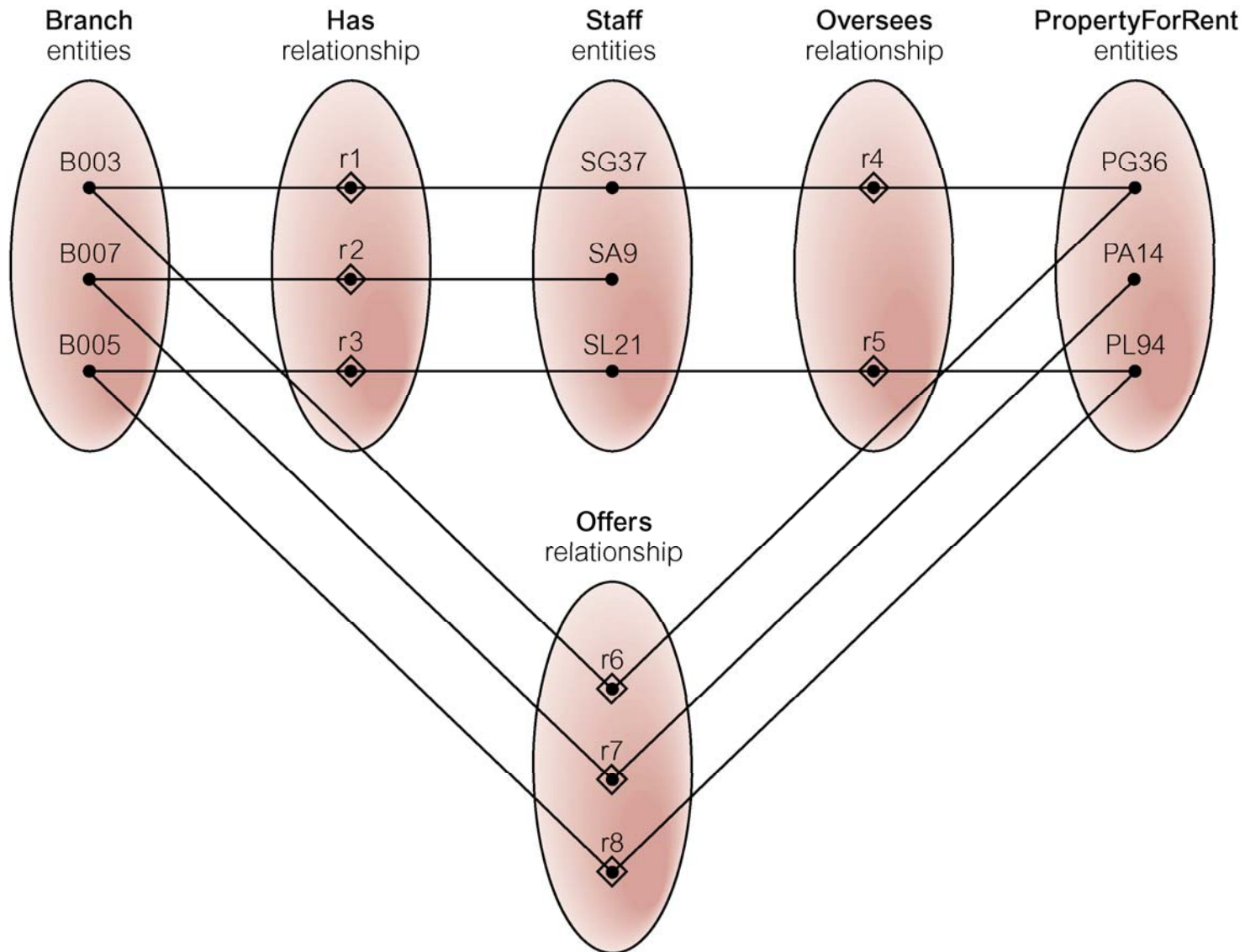


**At which branch office is property PA14 available?**

# ER Model restructured to remove Chasm Trap



# ER Model restructured to remove Chasm Trap





# Exercise

## Review questions

# Summary

- Overview of DB Design Phases
- ER Model
- EER Model
- Discussion: Problems with ER Model
- Exercises: Review Questions
- **Reading Suggestion: do not forget !!**
- Next lecture:
  - Logical DB Design - Relational Model:
    - **students' talks**
    - [1] chapters 5, 7, 12
    - [4] chapters 15, 16
  - Exercises

## Q&A

# *Questions ??*

Dr. Dang Tran Khanh ([khanh@cse.hcmut.edu.vn](mailto:khanh@cse.hcmut.edu.vn))