Faculty of Computer Science and Engineering
Ho Chi Minh City University of Technology

Chapter 2:
The Entity-Relationship Model

Database Systems
(CO2013)

Computer Science Program

Assoc. Prof. Dr. Võ Thị Ngọc Châu

(chauvtn@hcmut.edu.vn)

Semester 1 – 2022-2023

Content

- Chapter 1: An Overview of Database Systems

- **Chapter 2: The Entity-Relationship Model**

- Chapter 3: The Relational Data Model

- Chapter 4: The SQL Language

- Chapter 5: Relational Database Design

- Chapter 6: Physical Storage and Data Management

- Chapter 7: Database Security

Chapter 2:
The Entity-Relationship Model

- 2.1. Database design process from conceptual modeling

- 2.2. Conceptual data modeling

- 2.3. The entity-relationship model

- 2.4. The extended entity-relationship model

Main References

Text:

- [1] R. Elmasri, S. R. Navathe, Fundamentals of Database Systems- 6th Edition, Pearson- Addison Wesley, 2011.
  - ***R. Elmasri, S. R. Navathe, Fundamentals of Database Systems- 7th Edition, Pearson, 2016.***

References:

[1] S. Chittayasothorn, *Relational Database Systems: Language, Conceptual Modeling and Design for Engineers*, Nutcha Printing Co. Ltd, 2017.

[3] A. Silberschatz, H. F. Korth, S. Sudarshan, Database System Concepts – 7th Edition, McGraw-Hill, 2020.

[4] H. G. Molina, J. D. Ullman, J. Widom, *Database Systems: The Complete Book - 2nd Edition,* Prentice-Hall, 2009.

[5] R. Ramakrishnan, J. Gehrke, *Database Management Systems – 4th Edition,* McGraw-Hill, 2018.

[6] M. P. Papazoglou, S. Spaccapietra, Z. Tari, *Advances in Object-Oriented Data Modeling*, MIT Press, 2000.

[7]. G. Simsion, *Data Modeling: Theory and Practice*, Technics Publications, LLC, 2007.

2.1. Database design process from conceptual modeling

- ❑ The main phases
  - Input? Output?
  - ■ Requirements collection and analysis
    - ❑ Database designers
  - ■ *Conceptual* design
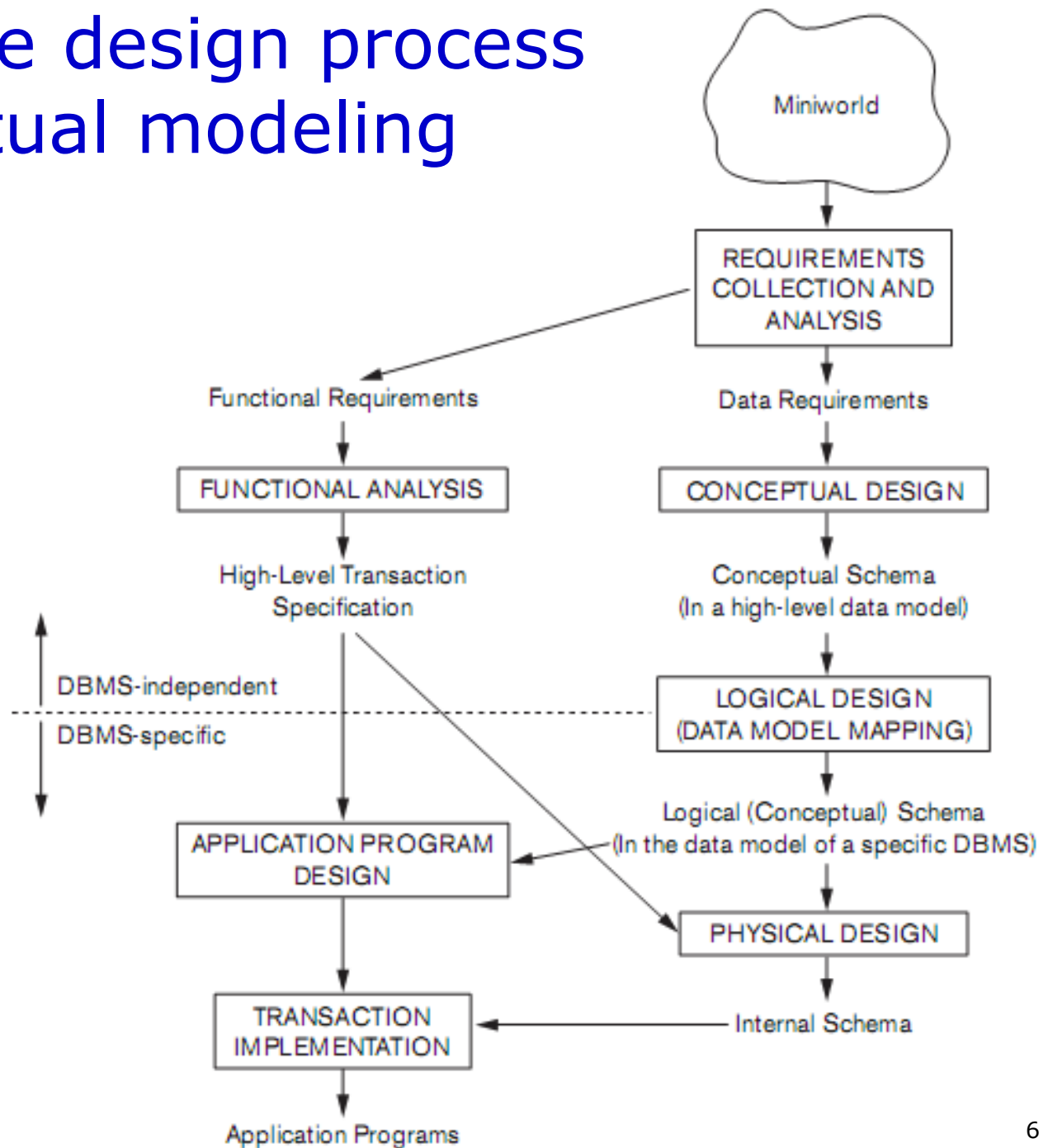   - ❑ The entity-relationship model
  - ■ *Logical* design (data model mapping)
    - ❑ The relational data model
  - ■ Physical design

2.1. Database design process from conceptual modeling

The main phases of database design

Miniworld

REQUIREMENTS COLLECTION AND ANALYSIS

Functional Requirements

Data Requirements

FUNCTIONAL ANALYSIS

CONCEPTUAL DESIGN

High-Level Transaction Specification

Conceptual Schema
(In a high-level data model)

DBMS-independent

DBMS-specific

LOGICAL DESIGN
(DATA MODEL MAPPING)

APPLICATION PROGRAM DESIGN

Logical (Conceptual) Schema
(In the data model of a specific DBMS)

PHYSICAL DESIGN

TRANSACTION IMPLEMENTATION

Internal Schema

Application Programs

2.2. Conceptual Data Modeling

- ❏ Data modeling

- ❏ Conceptual data model

  - ▪ The entity-relationship model

- ❏ Representational data model

  - ▪ The relational data model

- ❏ Database design

  - ▪ Relational database design

2.2. Conceptual Data Modeling

□ Modeling - Cambridge dictionary

- Model = a *representation* of something, either as a physical object which is usually smaller than the real object, or as a *simple description* of the object which might be used in calculations

- Modeling = *constructing* a representation of something, either as a physical object which is usually smaller than the real object, or as a simple description of the object which might be used in calculations

2.2. Conceptual Data Modeling

- Data modeling
  - "formalizing and representing the data structures of reality"
    - Shoval and Frumermann, 1994, p.28
  - " a representation of the things of significance to an enterprise and the relationships among those things"
    - Hay, 1996a
  - "an attempt to capture the essence of things both concrete and abstract"
    - Keuffel, 1996

Source: G. Simsion, *Data Modeling: Theory and Practice*, Technics Publications, LLC, 2007. Data modeling - **pp. 31-32.**

2.2. Conceptual Data Modeling

- Data modeling

  - "an abstract representation of the data about entities, events, activities, and their associations within an organization"

    - McFadden, Hoffer et al., 1999

  - "The core idea underlying all the definitions is the same: a data model is used for describing entities and their relationships within a core domain."

    - Topi and Ramesh, 2002

Source: G. Simsion, *Data Modeling: Theory and Practice*, Technics Publications, LLC, 2007. Data modeling - **pp. 31-32.**

2.2. Conceptual Data Modeling

- Data modeling
  - "Data modeling is generally viewed as a design activity"
    - Srinivasan and Te'eni, 1990
  - "an activity that involves the creation of abstractions"
    - Davydov, 1994
  - "the art and science of arranging the structure and relationship of data"
    - McComb, 2004, p.293
  - "data modeling is a design discipline"
    - Simsion and Witt, 2005, p.7

Source: G. Simsion, *Data Modeling: Theory and Practice*, Technics Publications, LLC, 2007. Data modeling - **pp. 31-32.**

Why is data modeling important?

- **Leverage**
  - Make programming simpler and cheaper
  - Poor data organization can be expensive to fix.
- **Conciseness**
  - Take more directly to the heart of the business requirements
- **Data quality**
  - Problems with data quality can be traced back to a lack of consistency in:
    - Defining and interpreting data
    - Implementing mechanisms to enforce the definitions
  - → A key role in achieving good data quality by establishing a common understanding of what is to be held in each table and column and how it is to be interpreted

G. C. Simsion, G. C. Witt, Data modeling essentials – 3rd edition, Elsevier Inc, 2005, pp. 8-10.

2.2. Conceptual Data Modeling

- A data model (aka semantic data model)

  - Provides concepts close to the way many users perceive data

  - Provides the concepts essential for supporting the application environment at a very high non-system-specific level

  - Used for a *conceptual schema* of a database from data requirements

  - Example: the entity-relationship model

2.2. Conceptual Data Modeling

- Characteristics of a conceptual data model
  - Expressiveness
    - Distinctions between data, relationships, constraints
  - Simplicity
    - Simple enough for an end user to use and understand → an easy diagrammatic notation
  - Minimality
    - A small number of basic concepts that are distinct and orthogonal in their meaning
  - Formality
    - Concepts must be formally defined → state criteria for the validity of a schema in the model
  - Unique interpretation
    - Complete and unambiguous semantics for each modeling construct

S. Navathe. Evolution of data modeling for databases. Communications of the ACM 35(9)(1992) 112-123.

Representational data model

- A data model (aka implementation/logical data model)

  - Provides concepts understood by end-users and able to be used to describe *the structure of a database*

    - the data types, relationships, and constraints that should hold for the data

    - a set of basic operations for specifying retrievals and updates on the database

  - Hides some details of data storage but able to be implemented on a computer system in a direct way (in some DBMS)

  - <u>Example</u>: the relational data model

Database design

- Design the logical and physical structures of one or more databases to accommodate the information needs of the users in an organization for a defined set of applications

- The overall database design activity has to undergo a systematic process called the design methodology, whether the target database is managed by an RDBMS, ODBMS, or ORDBMS, …

- The result of the design activity is a rigidly (fixedly) defined database schema that cannot easily be modified once the database is implemented.

Database design

- Goals:

    - Satisfy the information content requirements of the specified users and applications

    - Provide a natural and easy-to-understand *structuring* of the information

    - Support processing requirements and any performance objectives, such as response time, processing time, and storage space

Database design

Phases of database design and implementation for large databases

		Data content, structure, and constraints	Database applications
Phase 1:	Requirements collection and analysis	Data requirements	Processing requirements
Phase 2:	Conceptual database design	Conceptual Schema design (DBMS-independent)	Transaction and application design (DBMS-independent)
Phase 3:	Choice of DBMS		
Phase 4:	Data model mapping (logical design)	Logical Schema and view design (DBMS-dependent)	Frequencies, performance constraints
Phase 5:	Physical design	Internal Schema design (DBMS-dependent)	
Phase 6:	System implementation and tuning	DDL statements SDL statements	Transaction and application implementation

DDL: *data* definition language
SDL: *storage* definition language
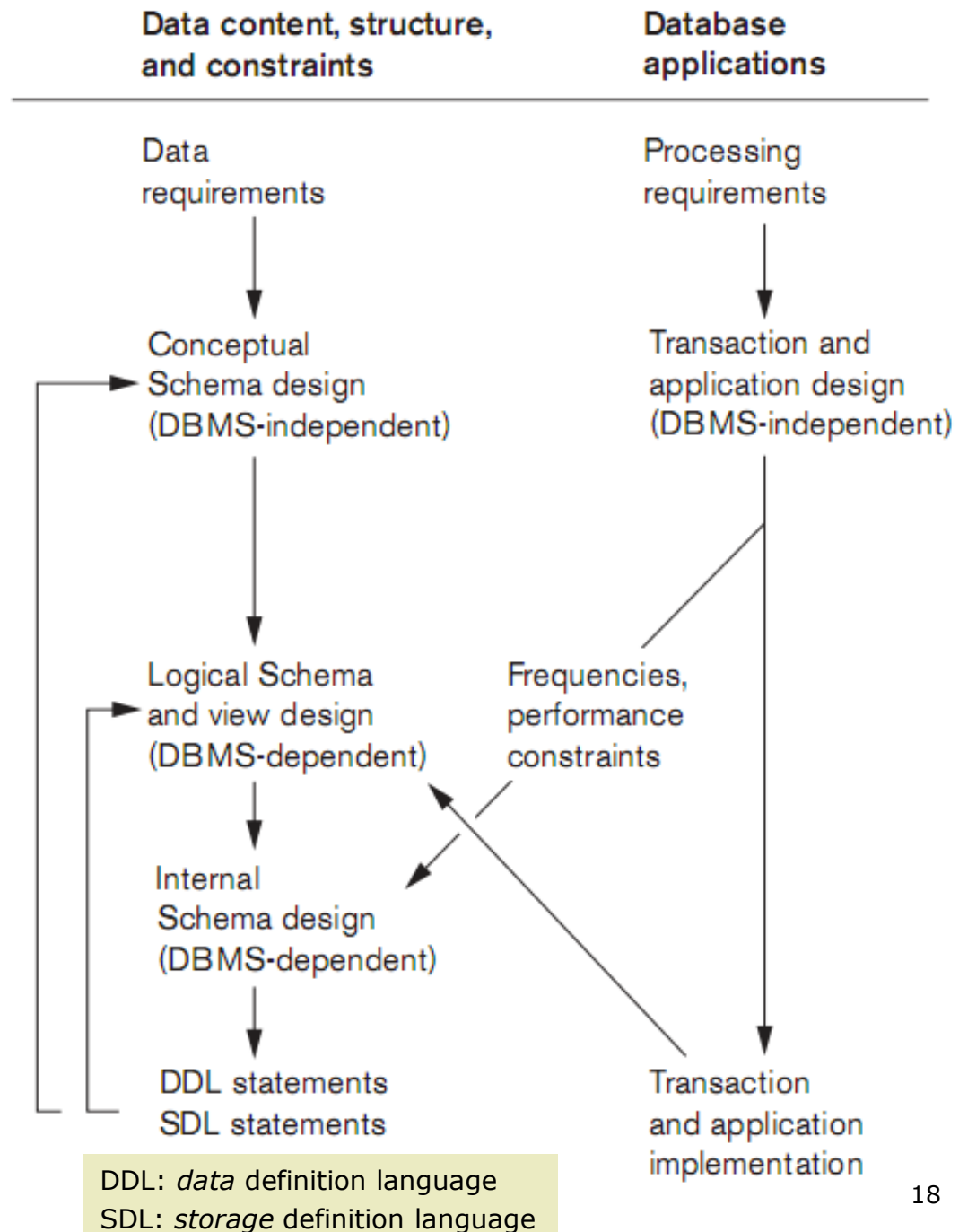
Database design

- Six main phases of the overall database design and implementation process

    - Requirements collection and analysis

    - Conceptual database design

    - Choice of a DBMS

    - Data model mapping (aka logical database design)

    - Physical database design

    - Database system implementation and tuning

Database design

- Phase 1: Requirements Collection and Analysis
    - Identify the major application areas and user groups that will use the database or whole work will be affected by the database
    - Study and analyze existing documentation concerning the applications
    - Study the current operating environment and planned use of the information
        - Analyze the types of transactions and their frequencies as well as of the flow of information within the system
        - Study geographic characteristics regarding users, origin of transactions, destination of reports, …
        - Specify the input and output data for the transactions
    - Collect written responses to sets of questions from the potential database users or user groups
        - Users' priorities and the importance users place on various applications
- → Requirements from users and applications of the information system that will interact with the database system
- → Time-consuming but crucial to the success of the information system
- → Identify and analyze the expectations of the users and the intended uses of the database in as much detail as possible

Database design

- **Phase 2**: Conceptual Database Design
  - **Phase 2a**: Conceptual Schema Design
    - Examine the data requirements resulting from Phase 1
    - Produce a *conceptual* database schema in a *DBMS-independent* high-level data model
  - **Phase 2b**: Transaction Design
    - Design the characteristics of known database transactions (applications) in a DBMS-independent way to ensure that the database schema will include all the information required by these transactions
      - Identify transactions' input/output and functional behavior
    - Group transactions into three categories: retrieval transactions, update transactions, mixed transactions

Database design

- **Phase 2**: Conceptual Database Design

  - **Phase 2a**: Conceptual Schema Design

    - Goal = a complete understanding of the database structure, meaning (semantics), interrelationships, and constraints

      - Identify: entity types, relationship types, attributes, key attributes, cardinality and participation constraints on relationships, weak entity types, specialization/generalization hierarchies, …

    - Approaches

      - The centralized (or one-shot) schema design approach

      - The view integration approach

Database design

- ## Phase 2a: Conceptual Schema Design
  - ### The centralized (or one-shot) schema design approach
    - The requirements of the different applications and user groups from Phase 1 are merged into a single set of requirements before schema design begins.
    - A single schema corresponding to the merged set of requirements is then designed.
    - The database administrator (DBA) is responsible for deciding how to merge the requirements and for designing the conceptual schema for the whole database.
    - Once the conceptual schema is designed and finalized, external schemas for the various user groups and applications can be specified by the DBA.

Database design

- Phase 2a: Conceptual Schema Design
  - The view integration approach
    - The requirements are not merged.
    - A schema (or view) is designed for each user group or application based only on its own requirements.
      - Each user group or application
    - These schemas are merged or integrated into a global conceptual schema for the entire database.
      - The DBA
    - The individual views can be reconstructed as external schemas after view integration.

Database design

- **<u>Phase 3</u>: Choice of a DBMS**
  - **Technical factors**
    - Suitability & type of the DBMS for the task
    - Storage structures and access paths, the user and programmer interfaces, high-level query languages, development tools, architectural options, … supported by DBMS
    - DBMS portability among different types of hardware
  - **Non-technical factors**
    - Cost: software acquisition cost, maintenance cost, hardware acquisition cost, database creation and conversion cost, personnel cost, training cost, operating cost
    - Availability of vendor services
  - …

Database design

- **Phase 4**: Data Model Mapping (Logical Database Design)

  - Create a *conceptual* schema and external schemas in the data model of the selected DBMS in the three-schema architecture.

  - The result of this phase should be DDL statements in the language of the *chosen DBMS* that specify the conceptual and external level schemas of the database system.

Database design

- **Phase 5**: Physical Database Design
  - Restricted to choosing the most appropriate structures for the *database files* from among the options offered by that DBMS
  - Choose *specific storage structures* and *access paths* for the database files
    - Response time
    - Space utilization
    - Transaction throughput
  - Estimate record size and number of records in each database file
  - Estimate the update and retrieval patterns for the file cumulatively from all the transactions
  - Estimate the file growth, either in the record size because of new attributes or in the number of records

Database design

- Phase 6: Database System Implementation and Tuning
  - Create the database schemas and (empty) database files
    - Responsibility of the DBA in conjunction with the database designers
  - Reformat the data for loading into the new database if needed
  - Load/populate with the data if needed
  - Implement database transactions referring to the conceptual specifications of transaction, then write and test program code with embedded DML commands
    - Responsibility of the application programmers
  - Database tuning continues as long as the database is in existence, as long as performance problems are discovered, and while the requirements keep changing.

Database design

Entity Relationship Model
P. P-S. Chen. *The Entity-Relationship Model – Toward a Unified View of Data*. ACM Transactions on Database Systems 1(1)(March 1976) 9-36.

DATA CONTENT AND STRUCTURE

DATABASE APPLICATIONS

Phase 1: REQUIREMENTS COLLECTION AND ANALYSIS

DATA REQUIREMENTS

PROCESSING REQUIREMENTS

Phase 2: CONCEPTUAL DATABASE DESIGN

CONCEPTUAL SCHEMA DESIGN (DBMS-independent)

TRANSACTION AND APPPLICATION DESIGN (DBMS-independent)

Phase 3: CHOICE OF DBMS

Phase 4: DATA MODEL MAPPING (LOGICAL DESIGN)

LOGICAL SCHEMA AND VIEW DESIGN (DBMS-dependent)

frequencies performance constraints

Phase 5: PHYSICAL DESIGN

INTERNAL SCHEMA DESIGN (DBMS-dependent)

Phase 6: SYSTEM IMPLEMENTATION AND TUNING

DDL statements SDL statements

TRANSACTION AND APPLICATION IMPLEMENTATION

Figure 2.1 Phases of database design and implementation for large databases [1], pp. 368
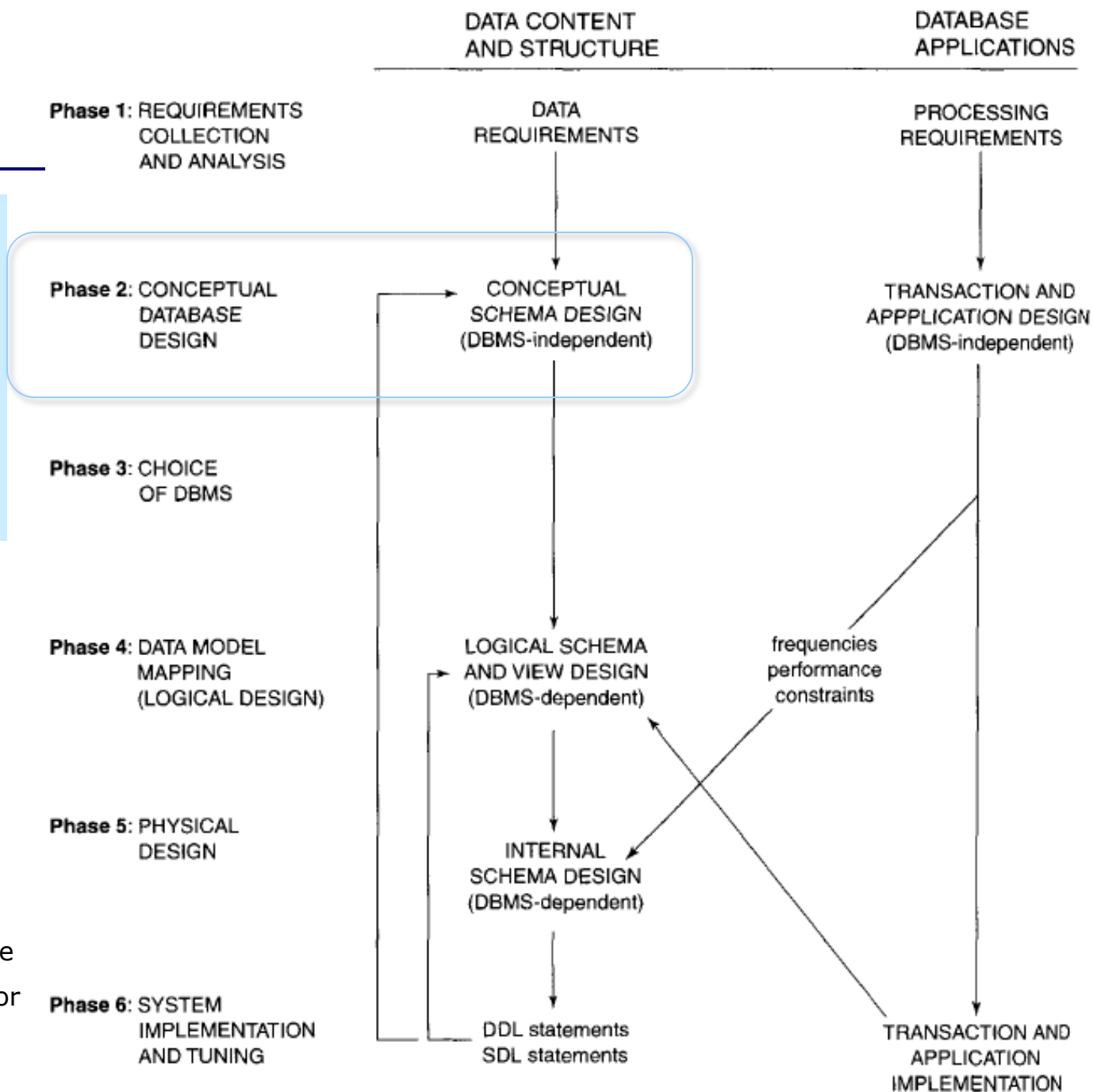
The Entity-Relationship Model

P. P-S. Chen. *The Entity-Relationship Model – Toward a Unified View of Data*. ACM Transactions on Database Systems 1(1)(March 1976) 9-36.

- The entity-relationship model adopts the more natural view that the real world consists of *entities*, *relationships*, and their *attributes*.

- The model can achieve a high degree of data independence and is based on set theory and relation theory.

- The entity-relationship model can be used as a basis for a *unified* view of data.

- A special diagrammatic technique, the entity-relationship diagram, is introduced as a tool for database design.

The Entity-Relationship Model
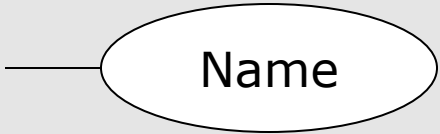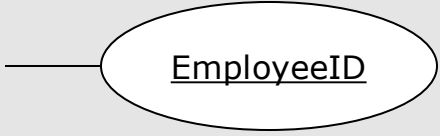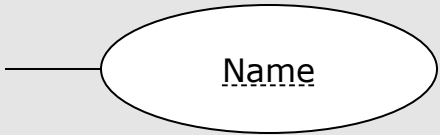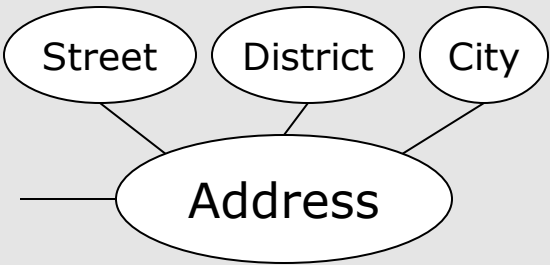
- The entity-relationship (ER) modeling concepts
  - Entity types
  - Relationship types
  - Attributes
  - Key attributes
  - Structural constraints
  - …

The Entity-Relationship Model

P. P-S. Chen. *The Entity-Relationship Model – Toward a Unified View of Data*. ACM Transactions on Database Systems 1(1)(March 1976) 9-36.

Symbol	Meaning	Example
Employee	Entity type	Employees
Dependent	Weak entity type	Dependents of an employee
works_on	Relationship type	Employee *works_on* Project
dependents_of	Identifying relationship type of the weak entity type	Dependents *dependents_of* Employees

Symbol	Meaning	Example
Name	Attribute of an *entity* or *relationship*	Name of an employee
EmployeeID	Key Attribute	Distinct identifier of an employee
Name	Partial Key of a Weak Entity Type	Name of a dependent of an employee
PhoneNumber	Multivalued Attribute	Phone numbers of an employee
Street, District, City / Address	Composite Attribute	Address (Street, District, City) of an employee
Age	Derived Attribute	Age of an employee (derived from attribute "*date of birth*")

The Entity-Relationship Model

P. P-S. Chen. *The Entity-Relationship Model – Toward a Unified View of Data*. ACM Transactions on Database Systems 1(1)(March 1976) 9-36.

Constraints
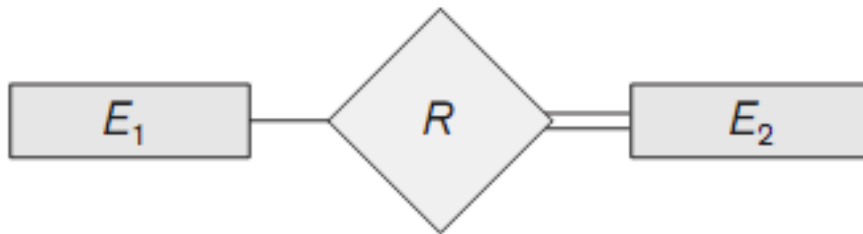
Total Participation of E_2 in R
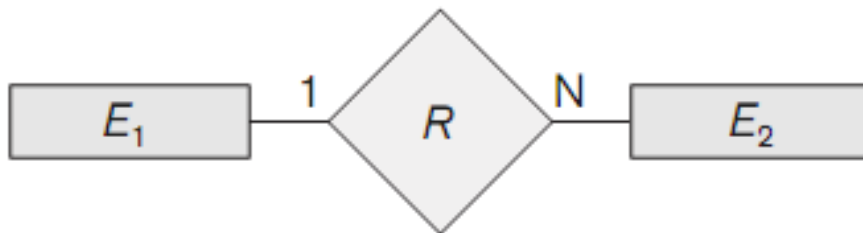
Cardinality Ratio 1: N for $E_1 : E_2$ in R
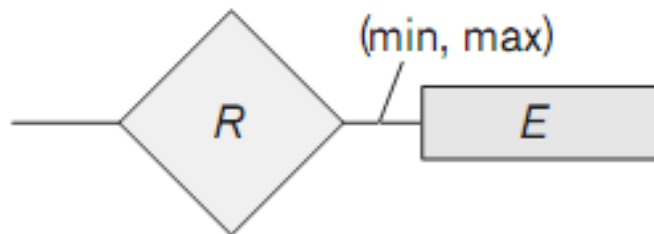
Structural Constraint (min, max) on Participation of E in R

A sample database application - COMPANY

The COMPANY database keeps track of a company's employees, departments, and projects. Suppose that after the requirements collection and analysis phase, the database designers provide the following description of the miniworld that will be represented in the database:

- The company is organized into departments. Each department has a unique name, a unique number, and a particular employee who manages the department. We keep track of the start date when that employee began managing the department. A department may have several locations.

- A department controls a number of projects, each of which has a unique name, a unique number, and a single location.

- The database will store each employee's name, Social Security number (SSN), address, salary, sex (gender), and birth date. An employee is assigned to one department, but may work on several projects, which are not necessarily controlled by the same department. It is required to keep track of the current number of hours per week that an employee works on each project, as well as the direct supervisor of each employee (who is another employee).

- The database will keep track of the dependents of each employee for insurance purposes, including each dependent's first name, sex, birth date, and relationship to the employee.
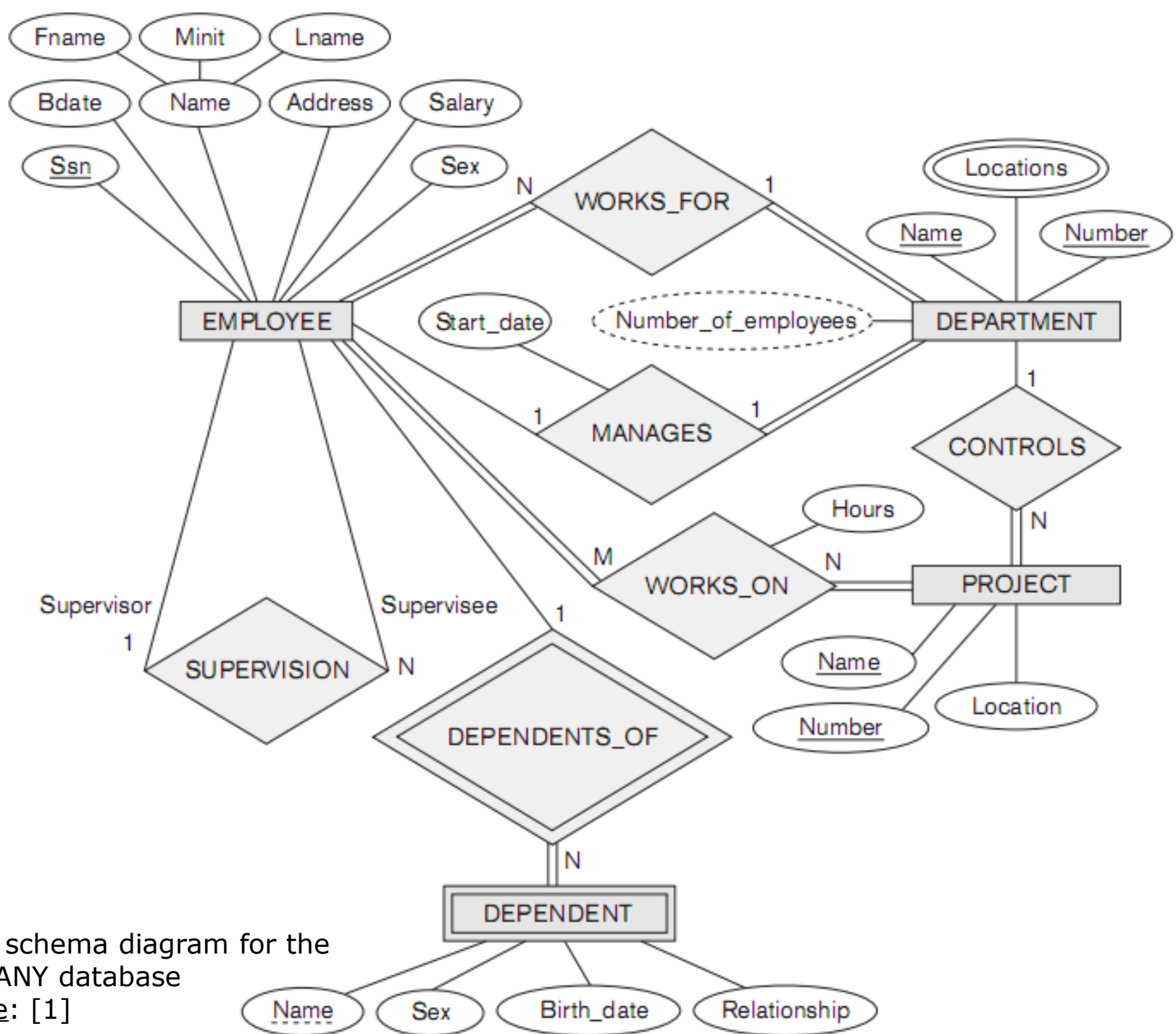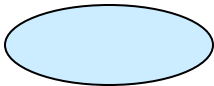
An ER schema diagram for the COMPANY database
Source: [1]

36

The Entity-Relationship Model

P. P-S. Chen. *The Entity-Relationship Model – Toward a Unified View of Data*. ACM Transactions on Database Systems 1(1)(March 1976) 9-36.

- **Entity**: a thing/ object in the real world with an independent existence, being distinguishable
  - physical existence (e.g. *person*, *book*, or *employee*)
  - conceptual existence (e.g. *company*, *job*, or *course*)
- **Entity type**: a *collection* (or *set*) of entities that have the same attributes (i.e. share the same *structure*)

  > EMPLOYEE

- **Attribute**: a particular property that describes an entity via a value
- **Key attribute**: attributes whose values are distinct for each entity in the entity set of an entity type

The Entity-Relationship Model

P. P-S. Chen. *The Entity-Relationship Model – Toward a Unified View of Data*. ACM Transactions on Database Systems 1(1)(March 1976) 9-36.

- **Attribute**: a particular property that describes an entity
 - An entity has a value for each of its attributes.
    - Entity *employee* has three **attributes**: *SSN*, *name*, *address* with the corresponding **values**: 123456789, 'Peter Pan', '1, Missing Path, Dream World'.
    - **NULL** is a special value to say "*not applicable*" or "*unknown*" for an attribute of a particular entity.
  - A **value set** (**domain**) specifies the set of values that may be assigned to an attribute for each entity.

SNN   name   address
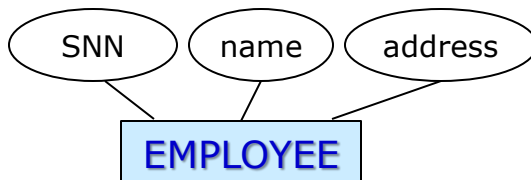
EMPLOYEE

Entity employee:

(123456789, 'Peter Pan', '1, Missing Path, Dream World')

The Entity-Relationship Model

- ❑ **Entity type**: a *collection* (or *set*) of entities that have the same attributes
    - ■ An entity type describes the schema or intension for a set of entities that share the same *structure*.
- ❑ **Entity set** (aka the extension of the entity type): the collection of entities of a particular entity type
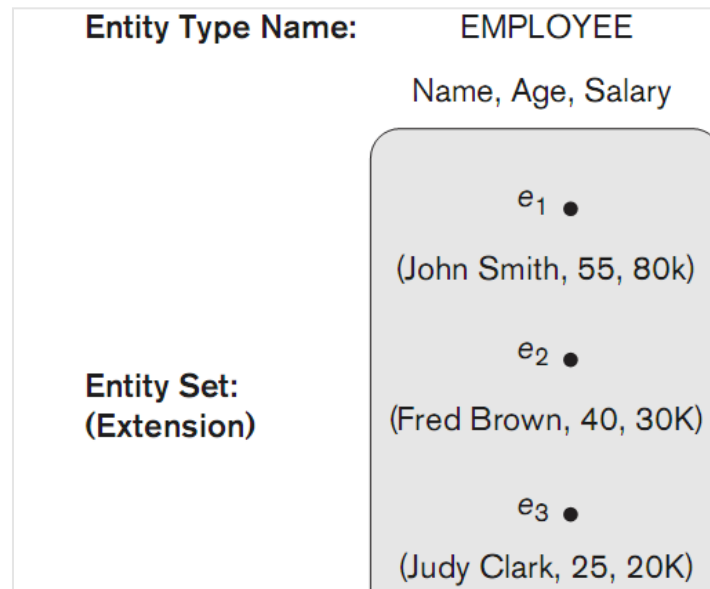
Entity Type Name:	EMPLOYEE
	Name, Age, Salary

e_1 •

(John Smith, 55, 80k)

e_2 •

(Fred Brown, 40, 30K)

**Entity Set:
(Extension)**

e_3 •

(Judy Clark, 25, 20K)

39

The Entity-Relationship Model

P. P-S. Chen. *The Entity-Relationship Model – Toward a Unified View of Data*. ACM Transactions on Database Systems 1(1)(March 1976) 9-36.

- **Entity type**: a *collection* (or *set*) of entities that have the same attributes

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

- **Entity set** (the extension of the entity type): the collection of entities of an entity type

- **Key attribute**: attributes whose values are distinct for each entity in the entity set

  - Key attribute values can be used to *identify each entity uniquely*.

  - Sometimes *several* attributes together form a key.

  - For example, attribute *SSN* is a key attribute of entity type EMPLOYEE.
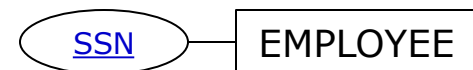
The Entity-Relationship Model

- **Attribute**: a particular property that describes an entity

  - Simple vs. Composite

  - Single-valued vs. Multivalued

  - Stored vs. Derived

  - Complex

The Entity-Relationship Model

❑ **Attribute**: a particular property that describes an entity

- **Simple vs. Composite**

  SSN

  EMPLOYEE

  - ❑ **Simple**: atomic attributes that are not divisible.
    - ▪ Attribute *SSN* is a simple attribute of entity *employee*.

  - ❑ **Composite**: attributes that can be divided into subparts. The value of a composite attribute is the concatenation of the values of its component simple attributes.
    - ▪ Attribute *address* is a composite attribute of entity *employee* because it can be divided into subparts: *street_number*, *street_name*, *district*.
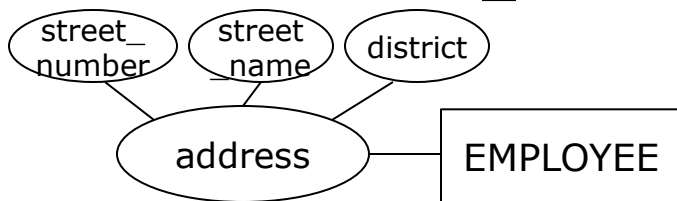
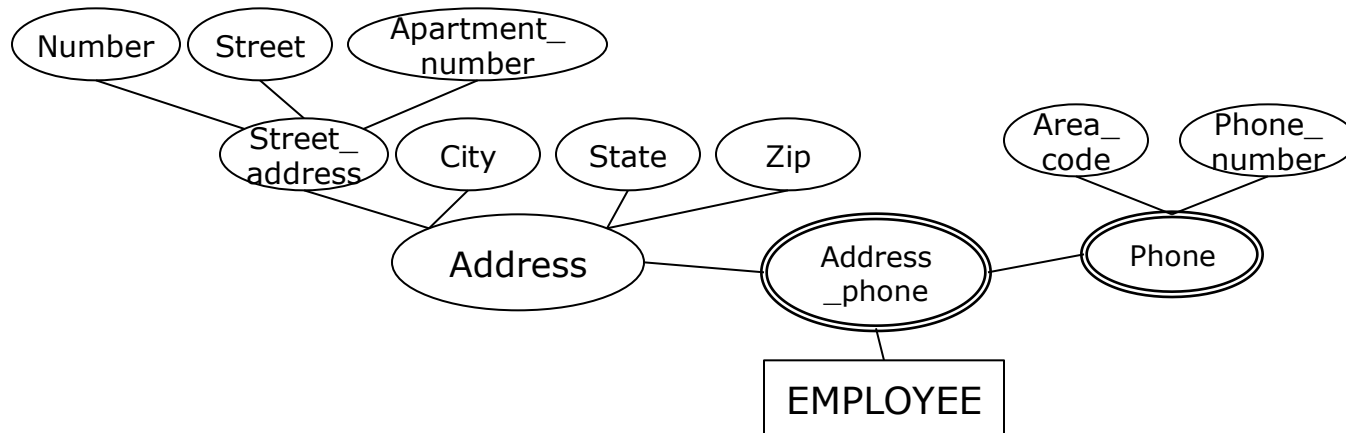 street_ number   street name   district

  address   EMPLOYEE

The Entity-Relationship Model

P. P-S. Chen. *The Entity-Relationship Model – Toward a Unified View of Data*. ACM Transactions on Database Systems 1(1)(March 1976) 9-36.

- **Attribute**: a particular property that describes an entity

  - **Simple vs. Composite**

  - **Single-valued vs. Multivalued**

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

      SSN

      EMPLOYEE

      - Attribute *SSN* is a single-valued attribute of entity *employee*.

    - **Multivalued**: an attribute has different values for a particular entity. A multivalued attribute may have *lower and upper bounds* to constrain the number of values allowed for each individual entity.

      phone_
      number
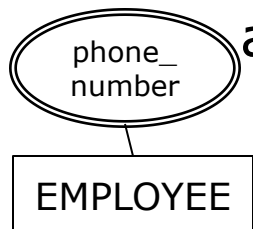
      EMPLOYEE

      - Attribute *phone_number* is a multivalued attribute of entity *employee*.

The Entity-Relationship Model

- **Attribute**: a particular property that describes an entity
  - **Simple vs. Composite**
  - **Single-valued vs. Multivalued**
  - **Stored vs. Derived**

    - **Stored**: an attribute whose value is recorded from the fact in the real world.

      SSN
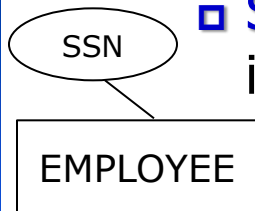
      EMPLOYEE

      - Attribute *SSN* is a **stored** attribute of entity *employee*.

    - **Derived**: an attribute whose value is computed (derived) from other values in the database.

      age   DOB
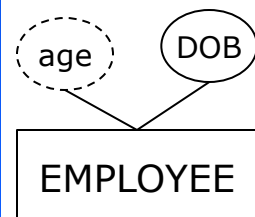
      EMPLOYEE

      - Attribute *age* is a **derived** attribute of entity *employee* because its value is derived from the value of attribute *DOB* of its corresponding entity *employee*.

The Entity-Relationship Model

- **Attribute**: a particular property that describes an entity
  - Simple vs. Composite
  - Single-valued vs. Multivalued
  - Stored vs. Derived
  - **Complex** Attributes: composite and multivalued attributes that can be *nested arbitrarily*.
    - {Address_phone({Phone(Area_code,Phone_number)}, Address(Street_address (Number,Street,Apartment_number), City,State,Zip))}
      - Phone: composite, multivalued
      - Street_address: composite
      - Address: composite
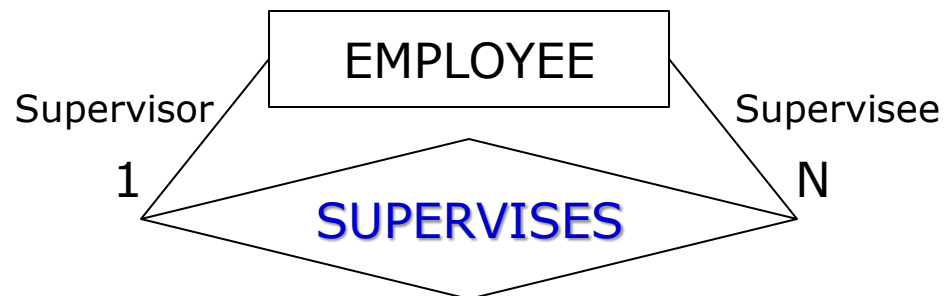   - Address_phone: composite, multivalued

The Entity-Relationship Model

P. P-S. Chen. *The Entity-Relationship Model – Toward a Unified View of Data*. ACM Transactions on Database Systems 1(1)(March 1976) 9-36.

- ## Attribute
  - ### Complex Attributes
    - {Address_phone({Phone(Area_code,Phone_number)}, Address(Street_address (Number,Street,Apartment_number), City,State,Zip))}
      - Phone: composite, multivalued
      - Street_address: composite
      - Address: composite
      - Address_phone: composite, multivalued

List all the entity types and their attributes. Classify their attributes.
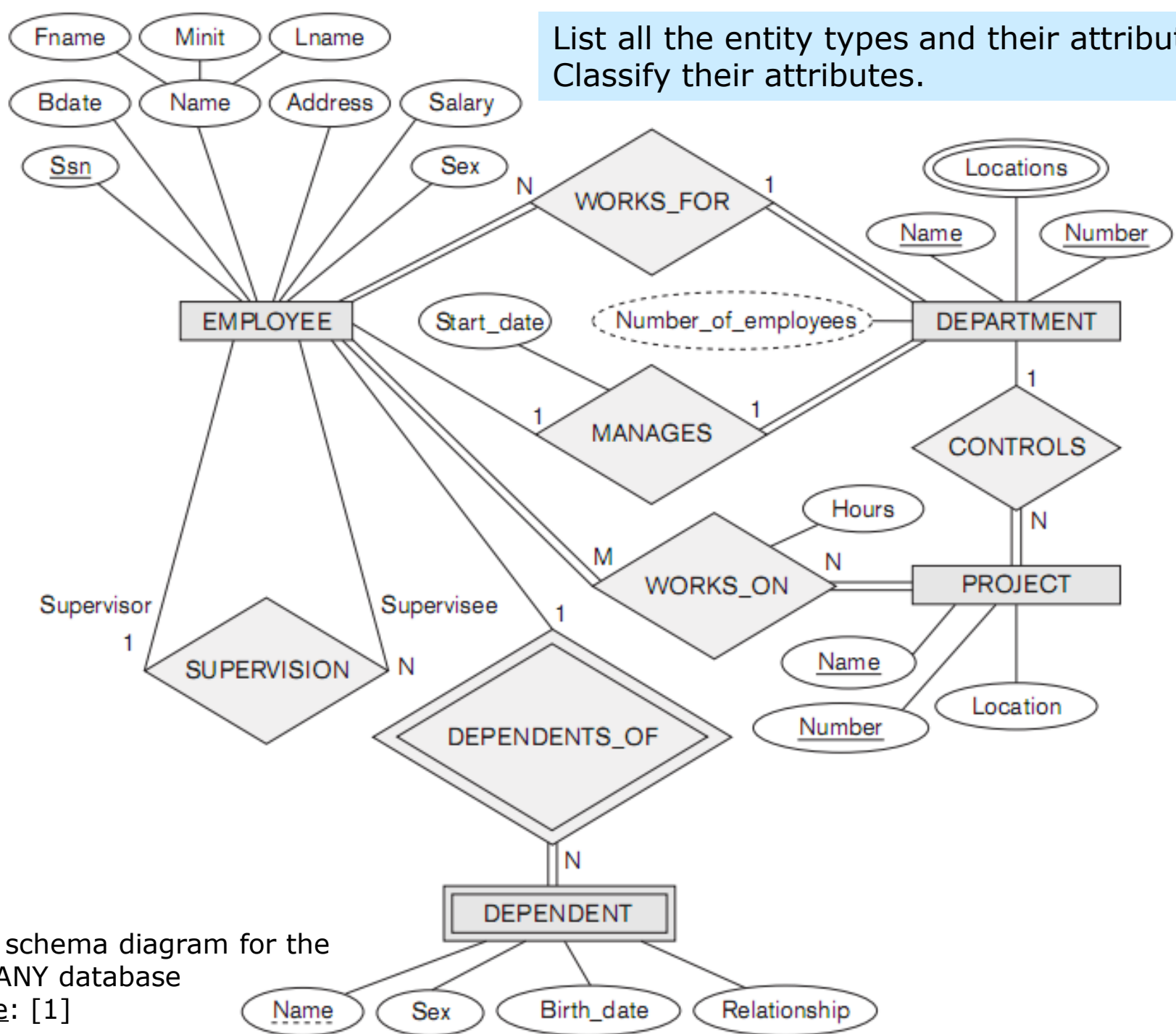
An ER schema diagram for the COMPANY database

Source: [1]

47

The Entity-Relationship Model

❑ **Relationship type *R* among *n* entity types $E_1, E_2, \ldots, E_n$**: a set of associations—or a relationship set—among entities from these entity types
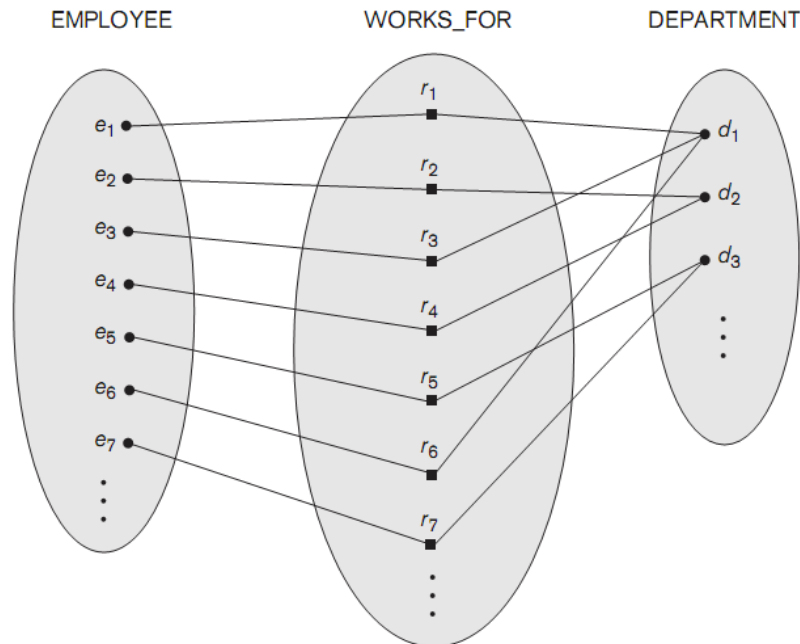
Relationship type WORKS_FOR
between EMPLOYEE and DEPARTMENT

ER diagram

48

The Entity-Relationship Model

- ❑ **Relationship type *R* among *n* entity types *E_1, E_2, . . . , E_n*:** a set of associations—or a relationship set—among entities from these entity types

  - ■ **Degree of a relationship type**: the number of participating entity types
    - ❑ For example, degree of relationship type WORKS_FOR is 2.
    - ❑ Degree 1: unary, Degree 2: binary, Degree 3: ternary
    - ❑ Unary relationship types: **recursive** (**self-referencing**) relationships

```
       ┌──────────────┐
       │  EMPLOYEE    │
Supervisor      Supervisee
      1 ╲    ╱    ╲    ╱ N
         ╲  ╱      ╲  ╱
          ╲╱_____╲╱
          ╱ SUPERVISES ╲
```

The Entity-Relationship Model

- ❒ **Relationship type *R* among *n* entity types *E_1, E_2, . . . , E_n*:** a set of associations—or a relationship set—among entities from these entity types
  - ■ **Degree of a relationship type**: the number of participating entity types
    - ❑ For example, degree of relationship type WORKS_FOR is 2.
    - ❑ Degree 1: unary, Degree 2: binary, Degree 3: ternary
    - ❑ Unary relationship types: **recursive** (**self-referencing**) relationships

Roles:
- Supervisor
- Supervisee

EMPLOYEE

Supervisor

Supervisee

1

N

SUPERVISES

The Entity-Relationship Model

- Relationship type R among n entity types E_1, E_2, . . . , E_n: a set of associations—or a relationship set—among entities from these entity types
  - Degree of a relationship type
  - Structural constraints on relationship types
    - Cardinality ratios: the *maximum* number of relationship instances an entity can participate in: 1:1, 1:N, N:1, N:M
    - Participation: the *minimum* number of relationship instances that each entity can participate in, i.e. whether the existence of an entity depends on its being related to another entity via the relationship type.
      - Total participation (existence dependency): *every* entity participates in the relationship type.
      - Partial participation: just *some* (not every) entities

51

The Entity-Relationship Model

EMPLOYEE   MANAGES   DEPARTMENT
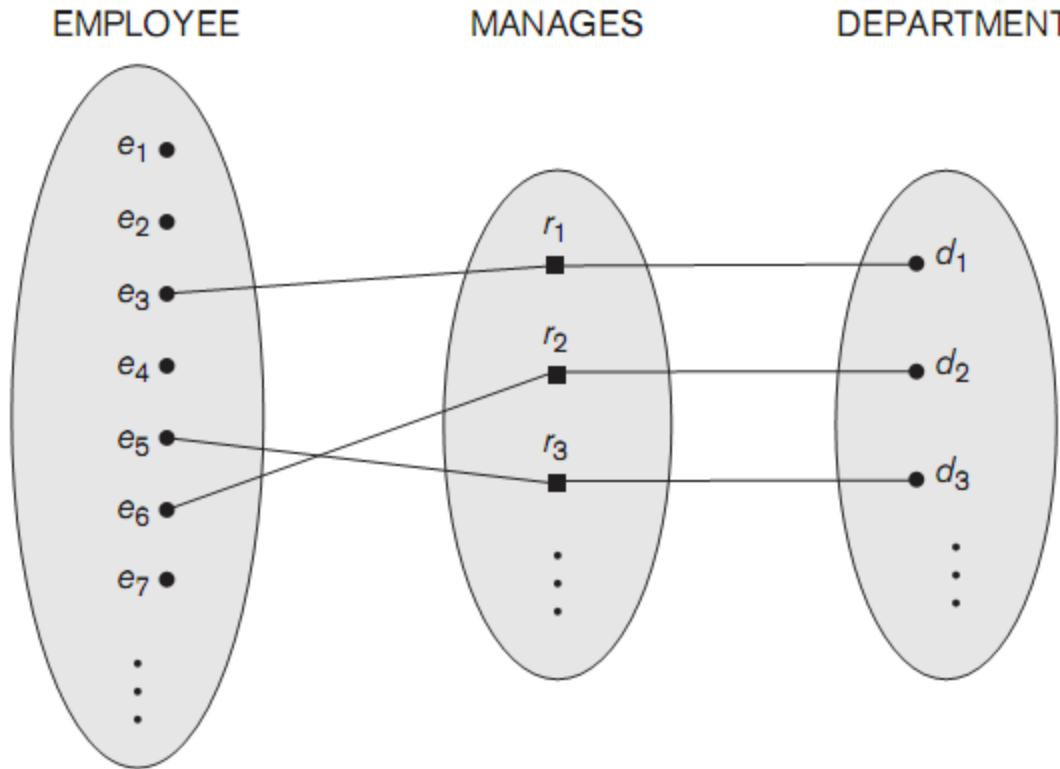
A 1:1 binary relationship type MANAGES
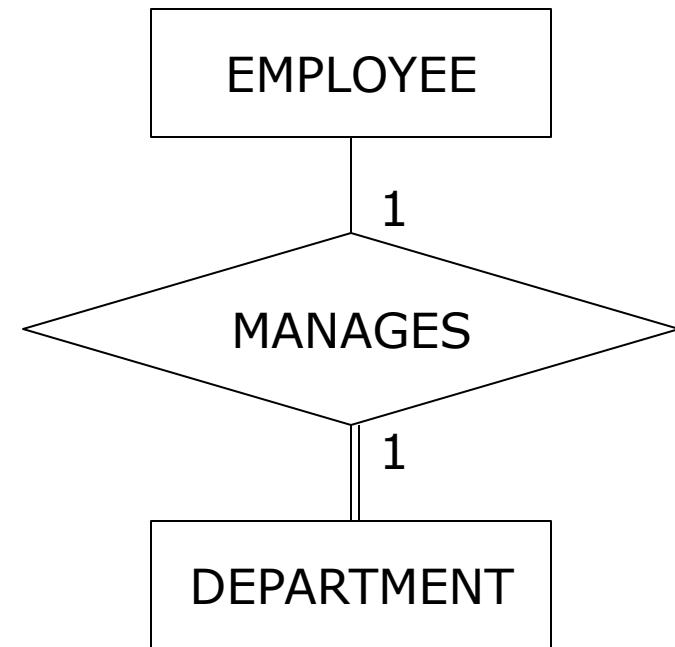
ER diagram

Not every employee manages *one* department. -> partial participation
Every department has *one* manager (employee). => total participation

52

The Entity-Relationship Model

P. P-S. Chen. *The Entity-Relationship Model – Toward a Unified View of Data*. ACM Transactions on Database Systems 1(1)(March 1976) 9-36.
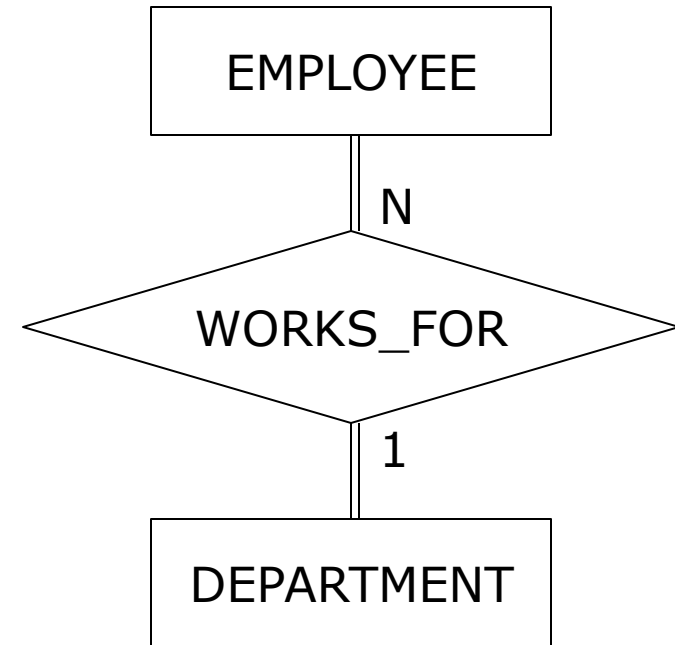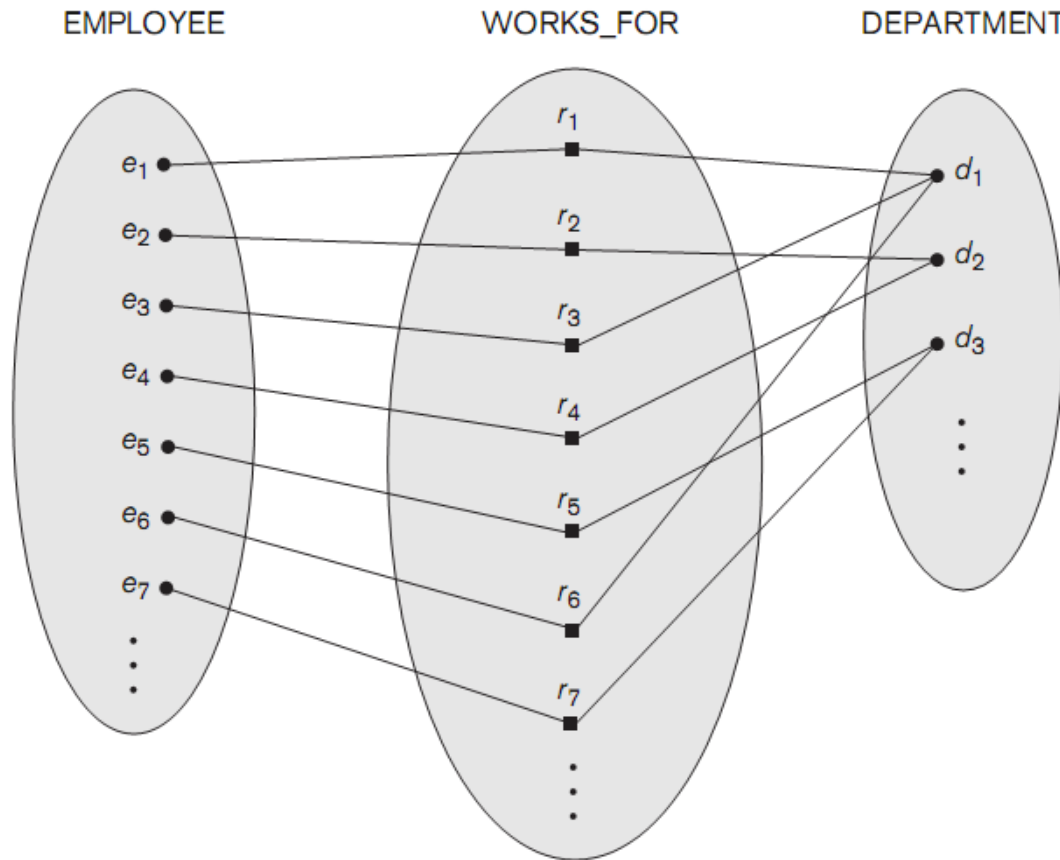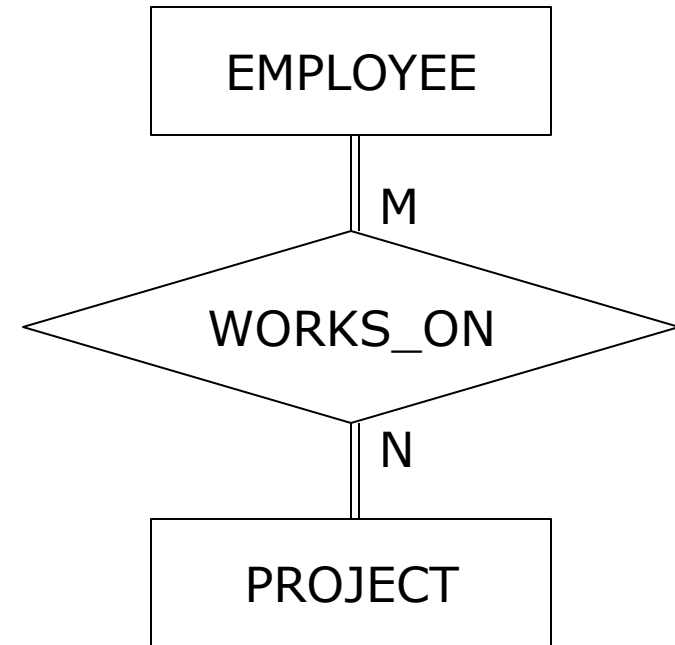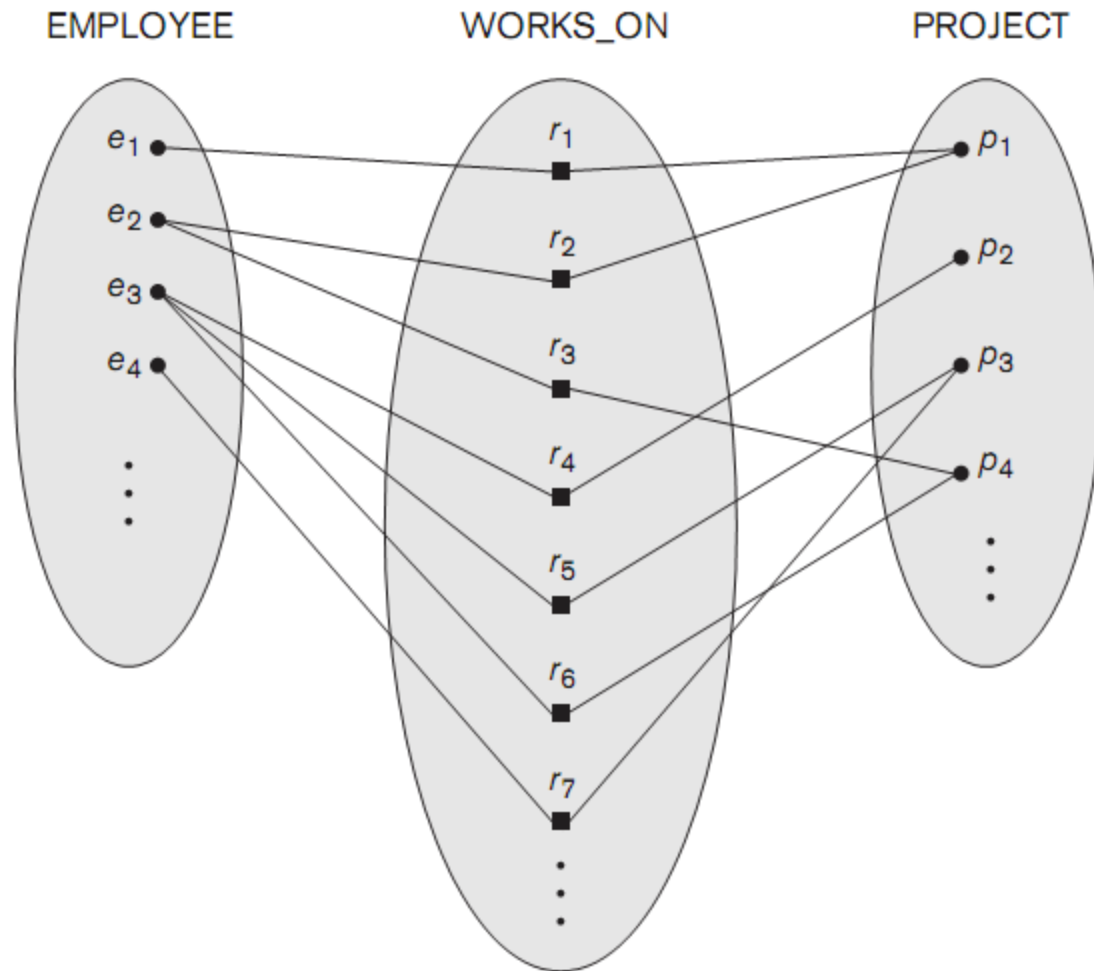

A N:1 binary relationship type WORKS_FOR

ER diagram

Every employee works for *one* department. => total participation
Every department has *one* to *N* employees. => total participation

The Entity-Relationship Model

P. P-S. Chen. *The Entity-Relationship Model – Toward a Unified View of Data*. ACM Transactions on Database Systems 1(1)(March 1976) 9-36.


An M:N binary relationship type WORKS_ON

ER diagram

Every employee works on *one* to *N* projects. => total participation
Every project has *one* to *M* employees. => total participation

The Entity-Relationship Model

- Relationship type *R* among *n* entity types E_1, E_2, . . . , E_n: a set of associations—or a relationship set—among entities from these entity types
  - Degree of a relationship type
  - Structural constraints on relationship types
  - Attribute of a relationship type: Relationship types can also have attributes, similar to those of entity types.
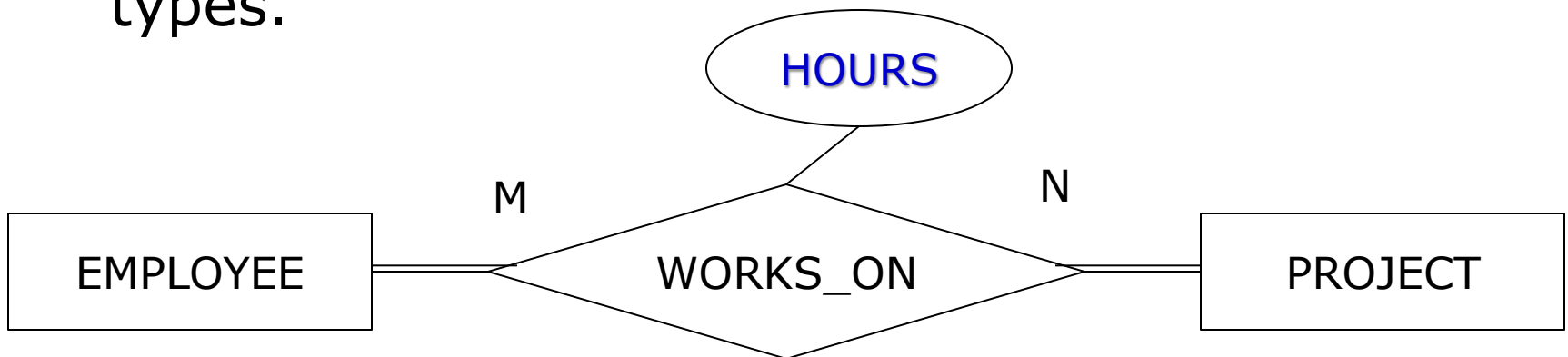
HOURS

M   N

EMPLOYEE   WORKS_ON   PROJECT

The Entity-Relationship Model

- Relationship type R among n entity types E_1, E_2, . . . , E_n: a set of associations—or a relationship set—among entities from these entity types
  - Degree of a relationship type
  - Structural constraints on relationship types
  - Attribute of a relationship type
  - → Put them altogether:
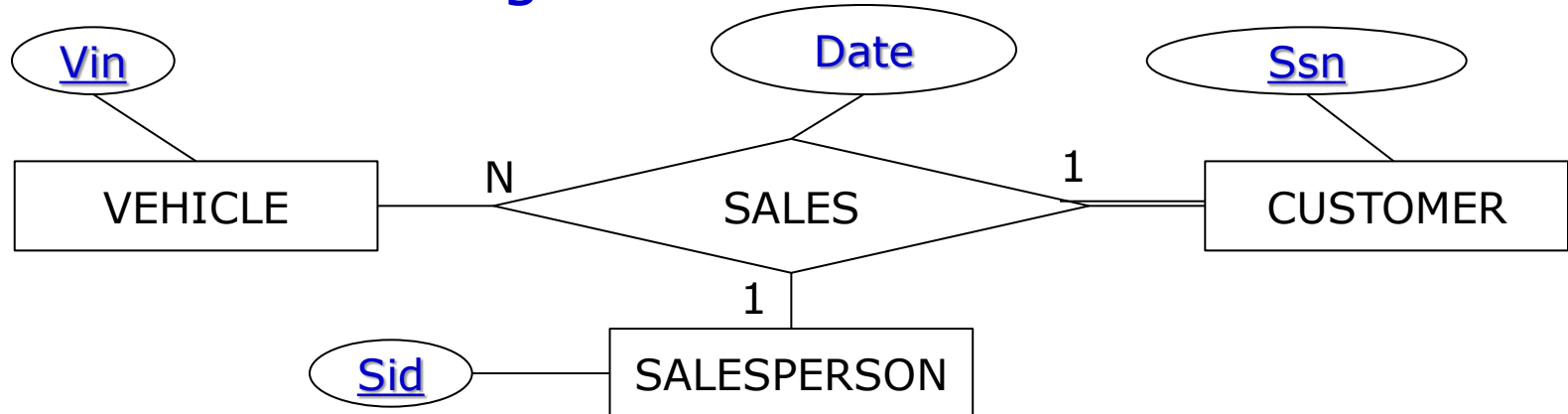
The Entity-Relationship Model

- **Entity type**: a *collection* (or *set*) of entities that have the same attributes

- **Key attribute**: attributes whose values are distinct for each entity in the entity set of an entity type

- **Weak entity type**: entity types that *do not* have key attributes of their own (in UoD)

  - Entities belonging to a weak entity type are *identified by* being related to specific entities from other entity types in combination with one of their attribute values.

    - *identifying or owner entity type, identifying relationship*

The Entity-Relationship Model

- **Entity type**: a *collection* (or *set*) of entities that have the same attributes

- **Weak entity type**: entity types that *do not* have key attributes of their own (in UoD)
    - Entities belonging to a weak entity type are *identified by* being related to specific entities from other entity types in combination with one of their attribute values.
        - *identifying or owner entity type*, *identifying relationship*
    - A weak entity type always has a *total participation* constraint (*existence dependency*) with respect to its identifying relationship.
    - A weak entity type *normally* has a *partial key*, which is the attribute that uniquely identifies weak entities that are related to the same owner entity.
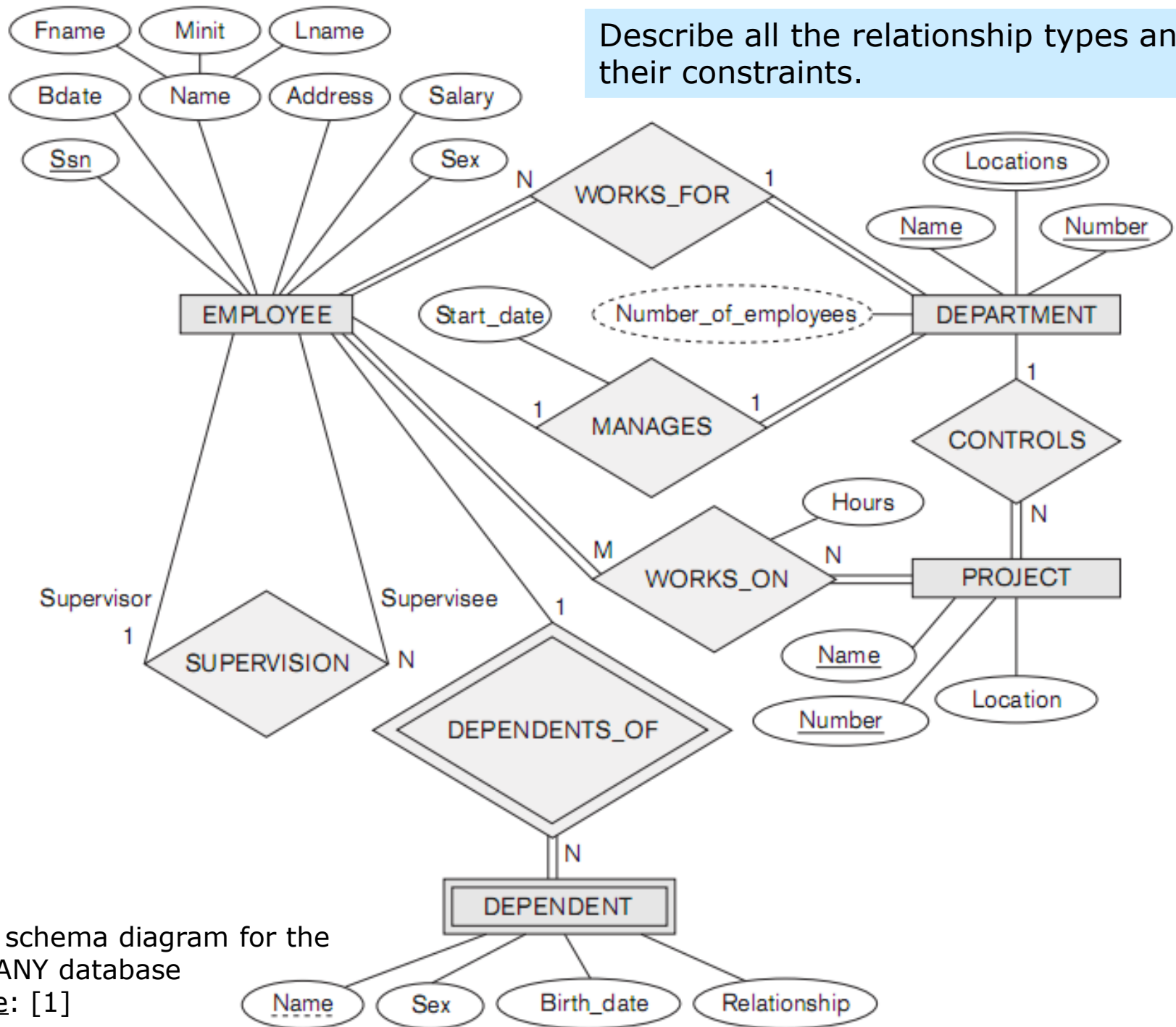
The Entity-Relationship Model

- **Entity type**: a *collection* (or *set*) of entities that have the same attributes
- **Weak entity type**: entity types that *do not* have key attributes of their own (in UoD)

SSN NAME

EMPLOYEE 1 DEPENDENTS_OF N DEPENDENT
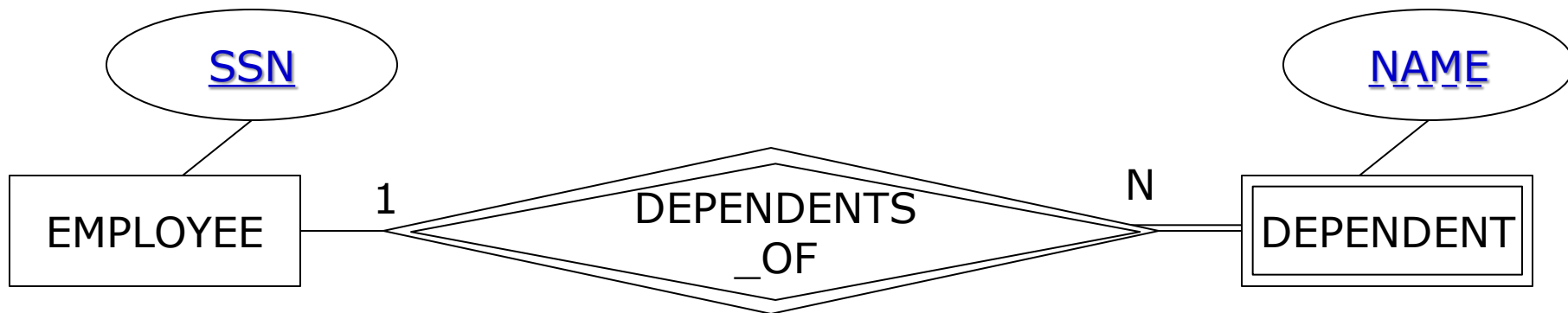
Dependents of an employee have *different Names*.

Every dependent is a dependent of *one* employee.
Not every employee has *one to N* dependents.

Describe all the relationship types and their constraints.

An ER schema diagram for the COMPANY database
Source: [1]

60

An ER diagram for a BANK database schema

Source: [1]
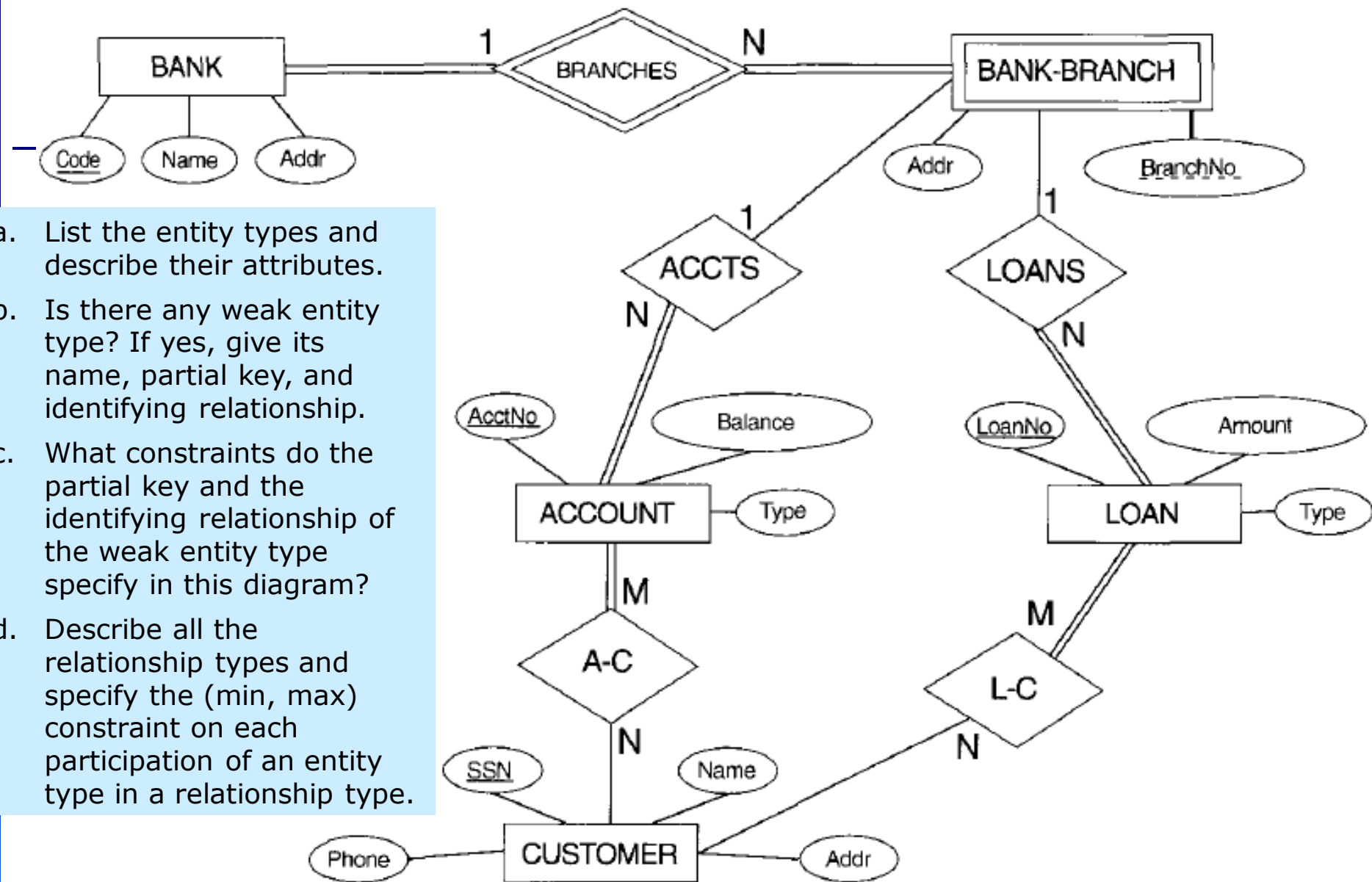
a. List the entity types and describe their attributes.

b. Is there any weak entity type? If yes, give its name, partial key, and identifying relationship.

c. What constraints do the partial key and the identifying relationship of the weak entity type specify in this diagram?

d. Describe all the relationship types and specify the (min, max) constraint on each participation of an entity type in a relationship type.
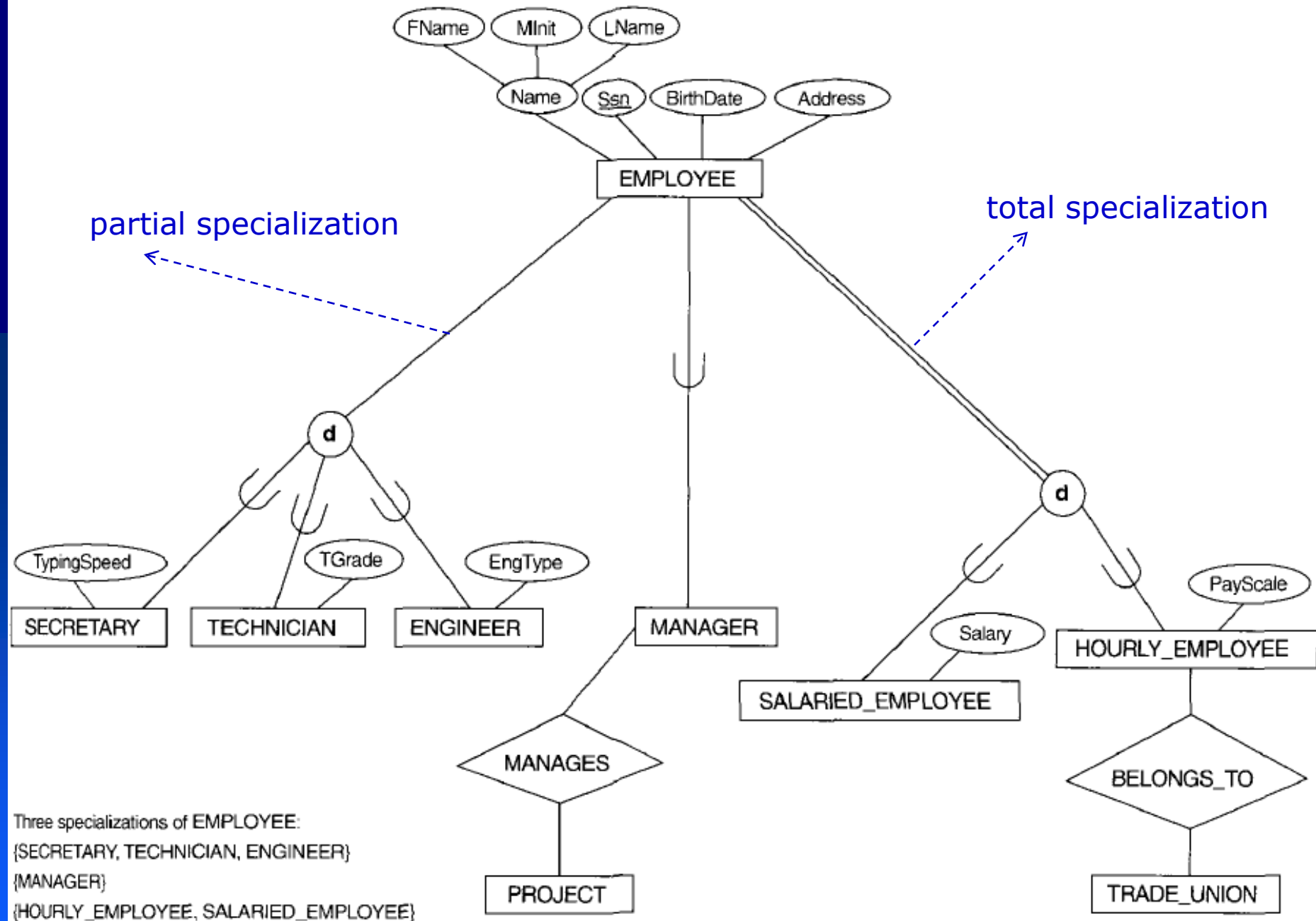
Design an Entity–Relationship diagram for the CONFERENCE_REVIEW database

Consider a CONFERENCE_REVIEW database in which researchers submit their research papers for consideration. Reviews by reviewers are recorded for use in the paper selection process. The database system primarily supports reviewers who record answers to evaluation questions for each paper they review and make recommendations regarding whether to accept or reject the paper. The data requirements are summarized as follows:

- Authors of papers are uniquely identified by email id. First and last names are also recorded.

- Each paper is assigned a unique identifier by the system and is described by a title, abstract, and the name of the electronic file containing the paper.

- A paper may have multiple authors, but one of the authors is designated as the contact author.

- Reviewers of papers are uniquely identified by email address. Each reviewer's first name, last name, phone number, affiliation, and topics of interest are also recorded.

- Each paper is assigned between two and four reviewers. A reviewer rates each paper assigned to him or her on a scale of 1 to 10 in four categories: technical merit, readability, originality, and relevance to the conference. Finally, each reviewer provides an overall recommendation regarding each paper.

- Each review contains two types of written comments: one to be seen by the review committee only and the other as feedback to the author(s).
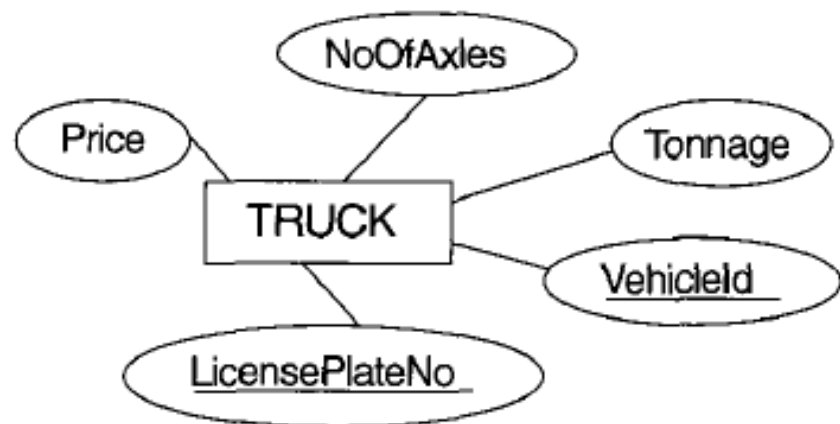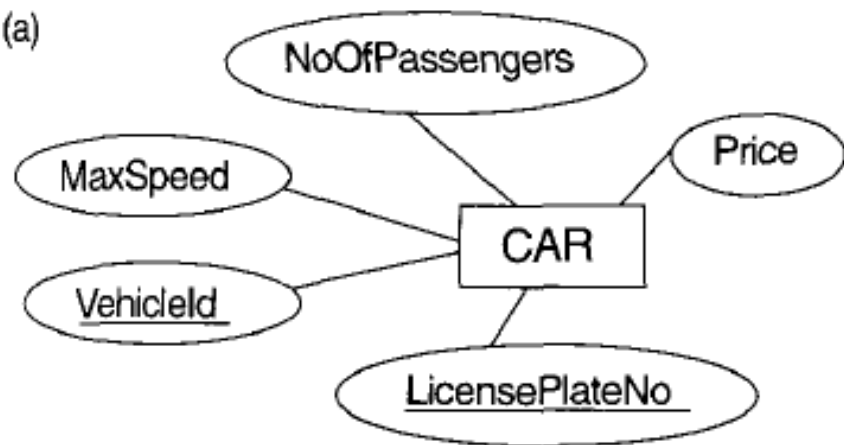
Enhanced Entity-Relationship Modeling

- Class/subclass relationships and type inheritance
  - The relationship between a *superclass* and any one of its subclasses
- *Specialization*
  - Define a set of subclasses of an entity type
  - Establish additional specific attributes with each subclass
  - Establish additional specific relationship types between each subclass and other entity types or other subclasses
- *Generalization*
  - A reverse process of abstraction in which we suppress the differences among several entity types
  - Identify their common features
  - Generalize them into a single superclass of which the original entity types are special subclasses
- *Union types* using categories
  - The union (U) of objects of different entity types

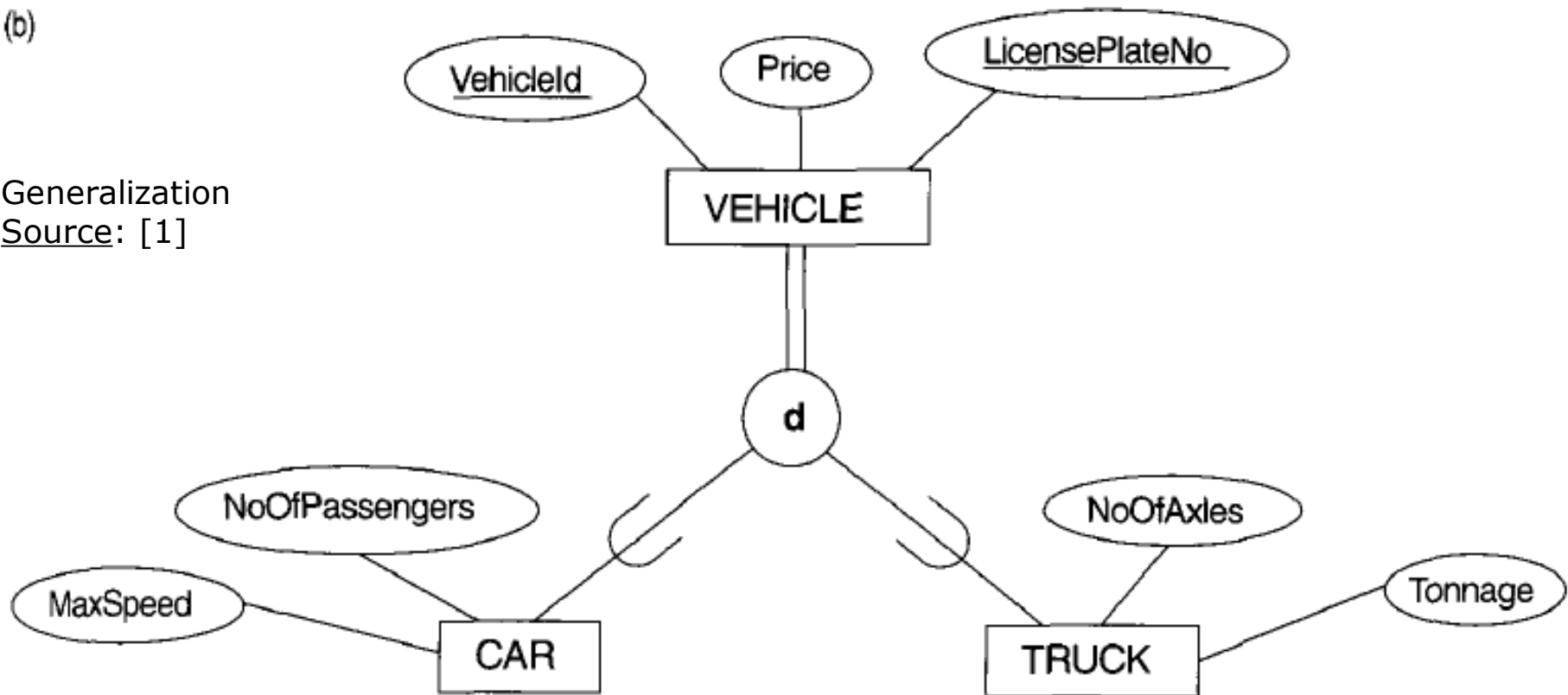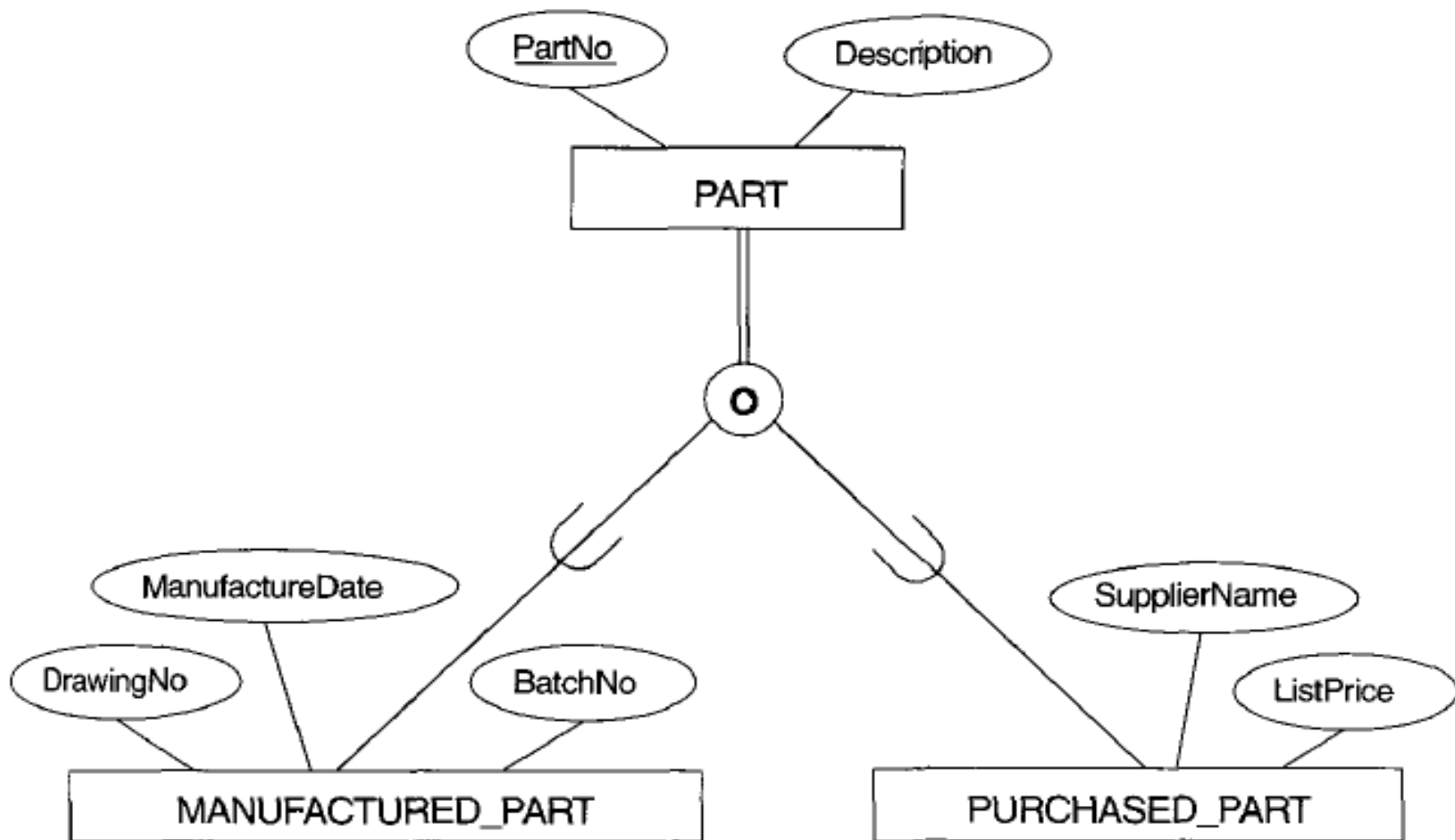EER diagram notation to represent subclasses and specialization.

(a)

NoOfPassengers

MaxSpeed

Price

VehicleId

CAR

LicensePlateNo

NoOfAxles

Price

Tonnage

TRUCK

VehicleId

LicensePlateNo

(b)

VehicleId

Price

LicensePlateNo

VEHICLE

d

Generalization
Source: [1]

NoOfPassengers

MaxSpeed

CAR

NoOfAxles

Tonnage

TRUCK

- Disjoint (d): the subclasses of the specialization must be disjoint.

- Overlapping (o): the subclasses are not constrained to be disjoint, their sets of entities may overlap.

EER diagram notation for overlapping (nondisjoint) specialization.

A specialization lattice with multiple inheritance for a UNIVERSITY database

Source: [1]

a. List superclasses, subclasses of each superclass

b. List class/subclass relationships

c. Describe constraints on each specialization
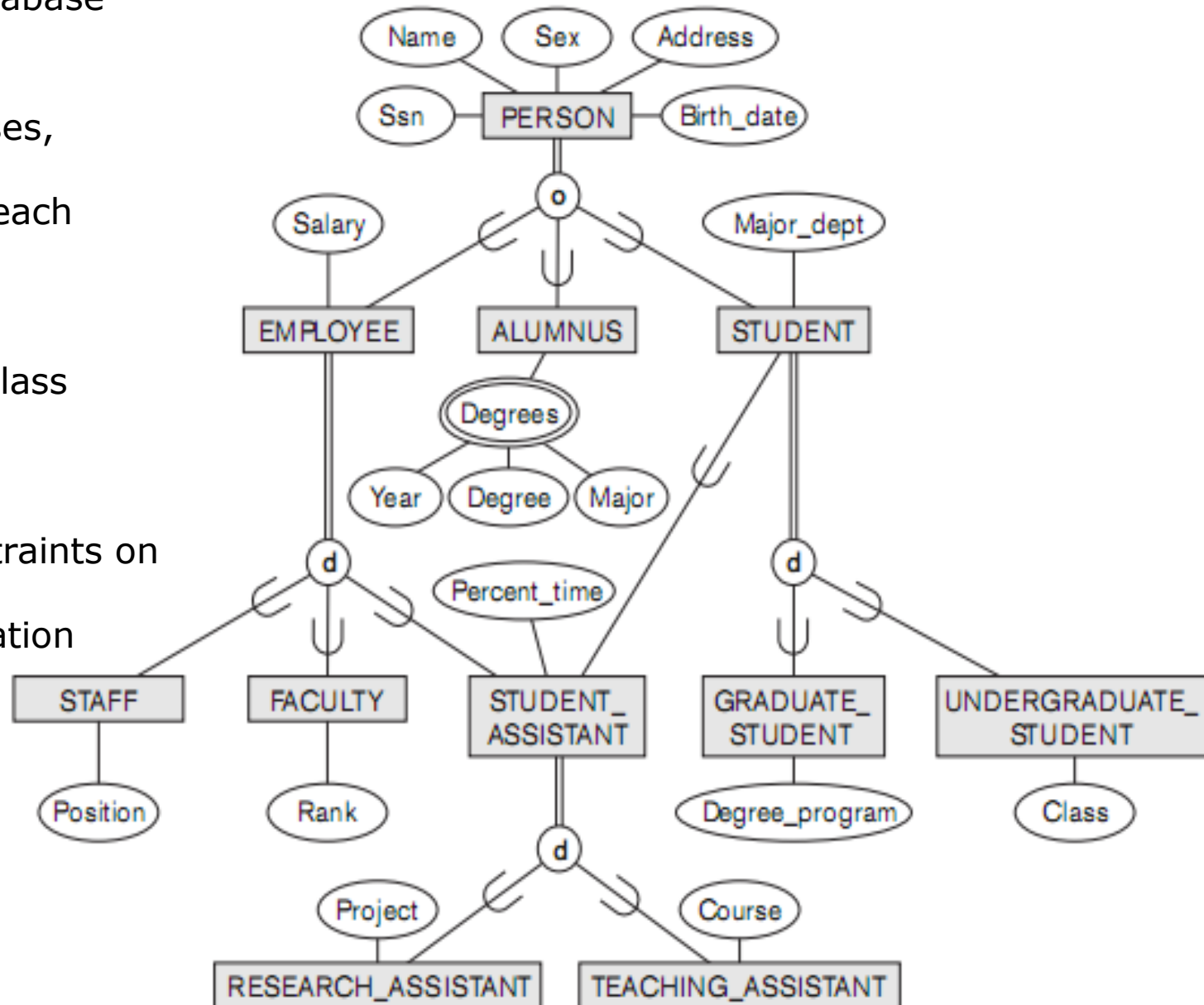
Two categories (union types): OWNER and REGISTERED_VEHICLE

<u>Source</u>: [1]

A category can be total or partial.

- A total category holds the union of all entities in its superclasses.

    + A double line (=) connecting the category and the circle

- A partial category can hold a subset of the union.

    + A single line (−) connecting the category and the circle

What are differences between a category and a superclass/subclass relationship?
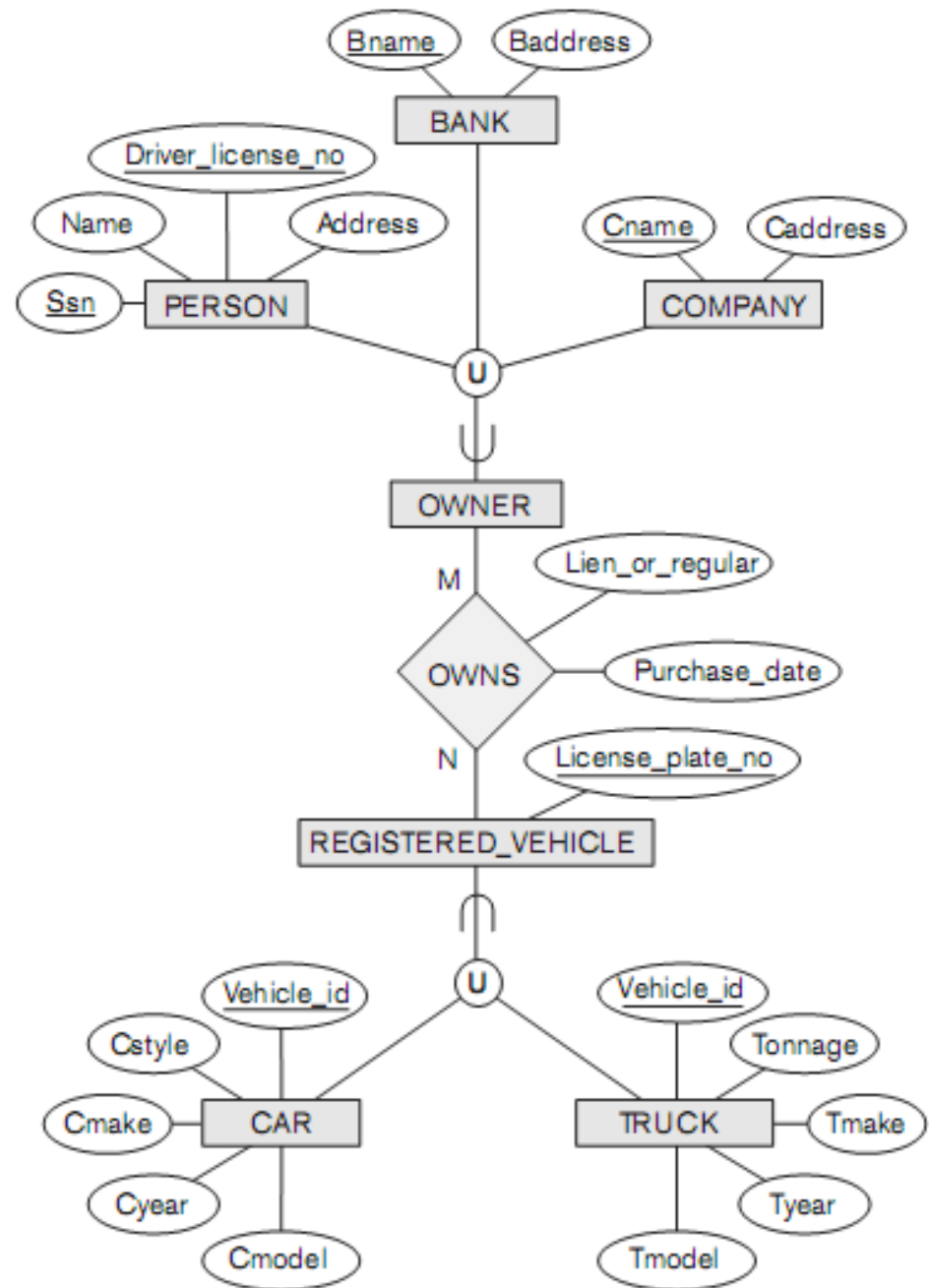
Summary

- Data modeling: conceptual, logical
- Database design process: 6 phases
  - Requirements collection and analysis
  - Conceptual database design
    - The entity-relationship model
  - Choice of a DBMS → a representational data model
    - The relational data model
  - Data model mapping (aka logical database design)
  - Physical database design
  - Database system implementation and tuning

Summary

□ The Entity-Relationship Model

■ Entity – Entity set – Entity type – Weak entity type

■ Relationship – Identifying Relationship - Relationship types

■ Attributes (simple vs. composite, single-valued vs. multivalued, stored vs. derived)

■ Key attributes – Partial keys

■ Structural constraints

■ …

Summary

- Enhanced Entity-Relationship Modeling

  - Subclass, superclass, attribute and relationship inheritance

  - Specialization vs. Generalization

    - Disjointness constraint: disjoint (d), overlapping (o)
    - Completeness (totalness) constraint: total (=), partial (−)

  - Category (Union type) (U)

    - Total (=)
    - Partial (−)

Chapter 2:
The Entity-Relationship Model

Review

- 2.1. Distinguish entity types and weak entity types, relationships and identifying relationships.

- 2.2. Give examples to differentiate between simple and composite attributes, between single-valued and multivalued attributes, between stored and derived attributes.

- 2.3. How can we make a choice of entity, attribute, and relationship in conceptual data modeling with the Entity-Relationship model? Give an example to justify your suggestions.

Review

□ 2.4. Design an E-R diagram of a university database application. The university database maintains records of its departments, lecturers, course modules, and students. The requirements are summarised as follows:

- The university consists of departments. Each department has a unique name and some other descriptive attributes.

- A department must also have a number of lecturers. One of them is the head of the department.

- All lecturers have unique identifiers and different names. They must teach one or more modules. A lecturer can only belong to one department.

- Modules are offered by departments and taught by lecturers. They must also be attended by some students. Each module has a unique module number.

- Students must enrol for a number of modules. Each student is given a unique student number.
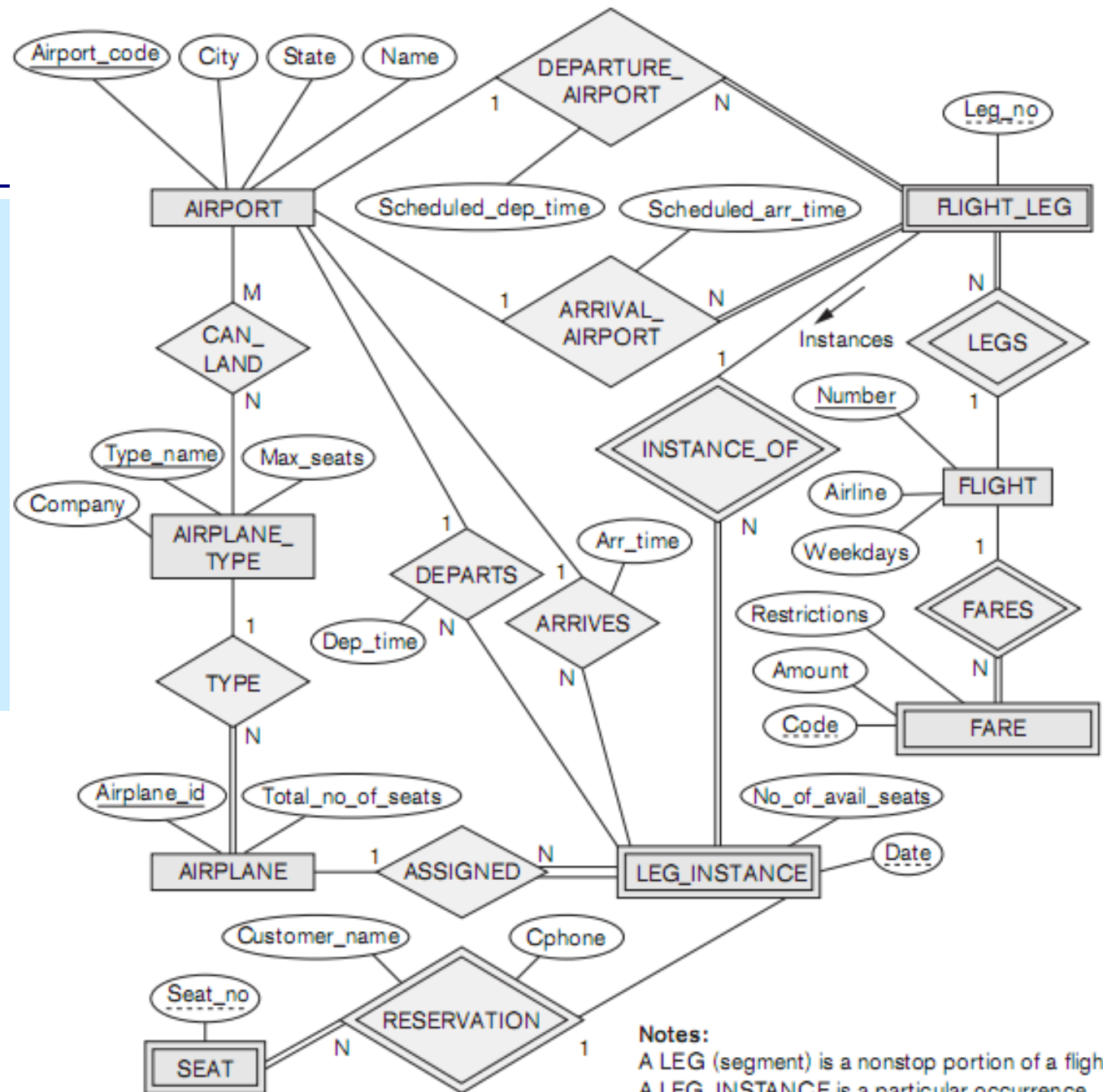
Review

2.5. Consider the E-R diagram which shows a simplified schema for an airline reservations system.

- a. List the strong (non-weak) entity types in the ER diagram.

- b. Is there any weak entity type? If yes, give its name, partial key, and identifying relationship.

- c. What constraints do the partial key and the identifying relationship of the weak entity type specify in this diagram?

- d. Describe all the relationship types and specify the (min, max) constraint on each participation of an entity type in a relationship type. Justify your choices.
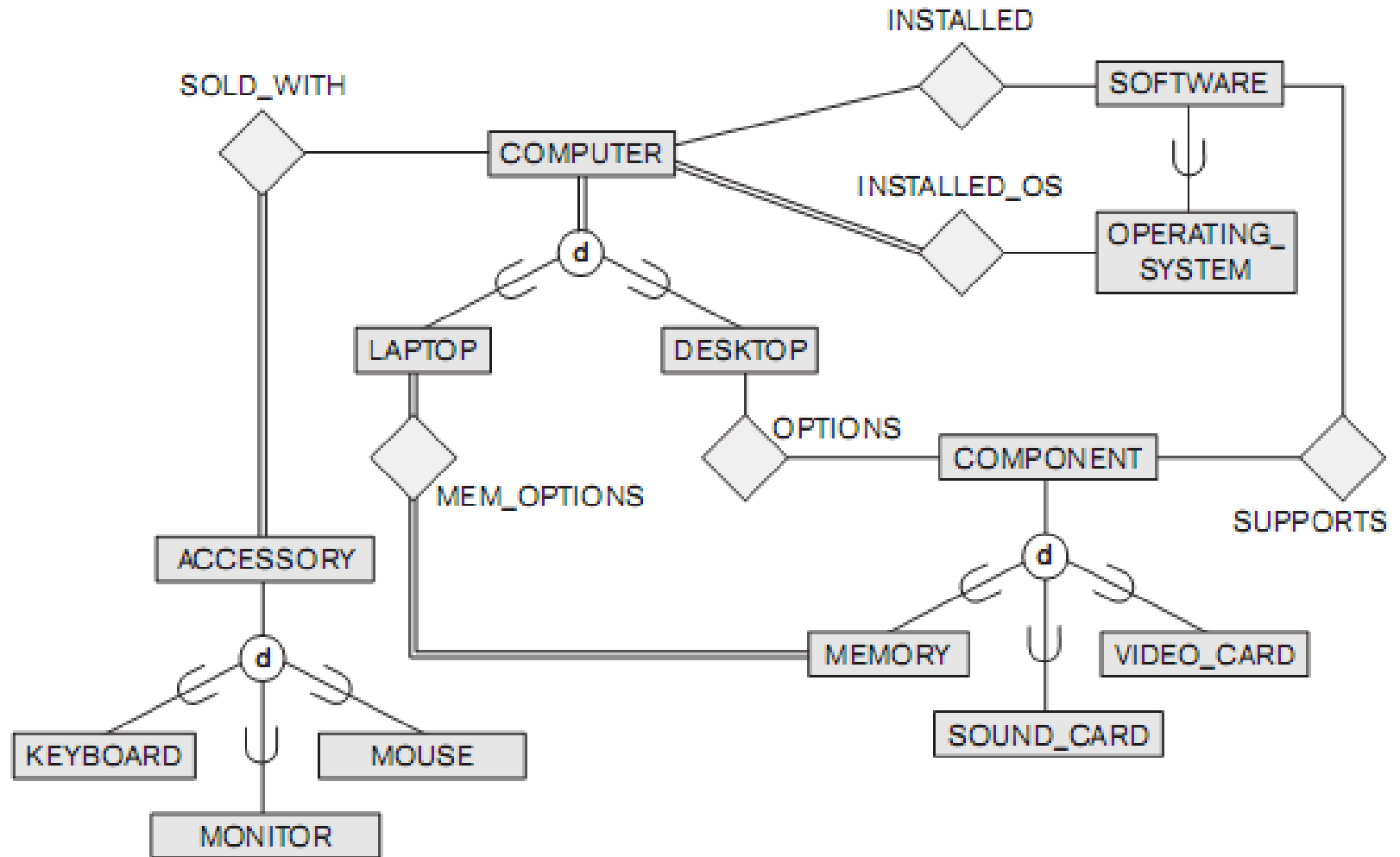
Review

2.5. Consider the E-R diagram which shows a simplified schema for an airline reservations system.

Notes:
A LEG (segment) is a nonstop portion of a flight.
A LEG_INSTANCE is a particular occurrence of a LEG on a particular date.

Review

- 2.6. Consider the following EER diagram that describes the computer systems at a company. Provide your own attributes and key for each entity type. Supply max cardinality constraints justifying your choice. Write a complete narrative description of what this EER diagram represents.

Review

2.6. Consider the following EER diagram that describes the computer systems at a company.
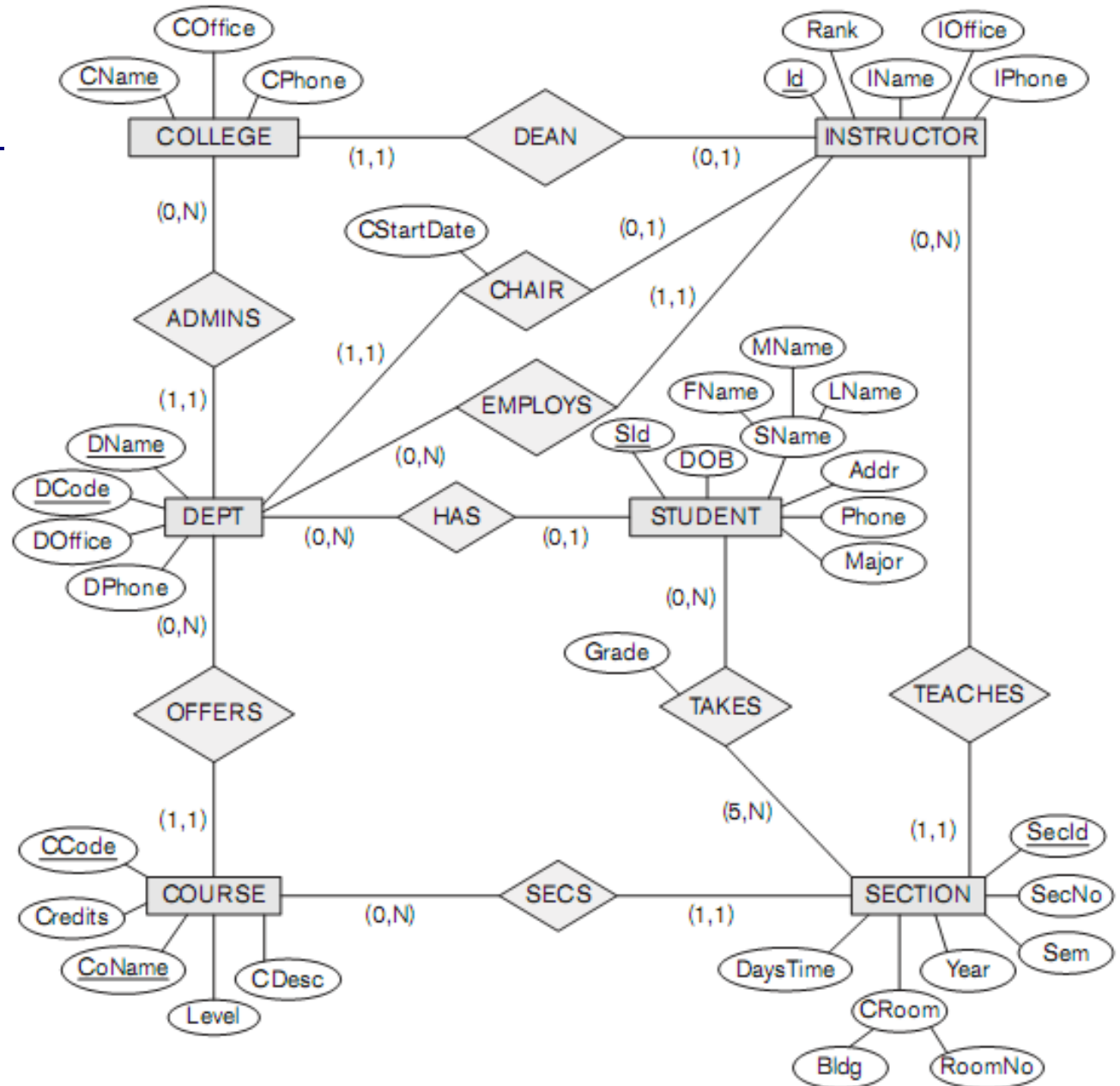
Review

- 2.7. Design an enhanced entity–relationship diagram for a GRADE_BOOK database in which instructors within an academic department record points earned by individual students in their classes. The data requirements are summarized below:

  - Each student is identified by a unique identifier, first and last name, and an email address.

  - Each instructor teaches certain courses each term. Each course is identified by a course number, a section number, and the term in which it is taught. For each course he or she teaches, the instructor specifies the minimum number of points required in order to earn letter grades A, B, C, D, and F. For example, 90 points for an A, 80 points for a B, 70 points for a C, etc.

  - Students are enrolled in each course taught by the instructor.

  - Each course has a number of grading components (such as midterm exam, final exam, project, and so forth). Each grading component has a maximum number of points (such as 100 or 50) and a weight (such as 20% or 10%). The weights of all the grading components of a course usually total 100.

  - Finally, the instructor records the points earned by each student in each of the grading components in each of the courses. For example, student 1234 earns 84 points for the midterm exam grading component of the section 2 course CSc2310 in the fall term of 2009. The midterm exam grading component may have been defined to have a maximum of 100 points and a weight of 20% of the course grade.

Review

- 2.8. Consider the UNIVERSITY database as follows. Modify this diagram by classifying COURSES as either UNDERGRAD_COURSES or GRAD_COURSES and INSTRUCTORS as either JUNIOR_PROFESSORS or SENIOR_PROFESSORS. Include appropriate attributes for these new entity types. Then establish relationships indicating that junior instructors teach undergraduate courses whereas senior instructors teach graduate courses.

Review

2.8. Consider the UNIVERSITY database.

Review

- 2.9. Some people consider that problems called *connection traps* may arise with the Entity-Relationship model when designing a conceptual database schema. Two main types of connection traps are called *fan traps* and *chasm traps*. What are *fan traps* and *chasm traps*? Give examples to illustrate them. Are they *really* the problems with the Entity-Relationship model? If *not*, explain your answer and then fix the problems faced in your previously given examples.

Review

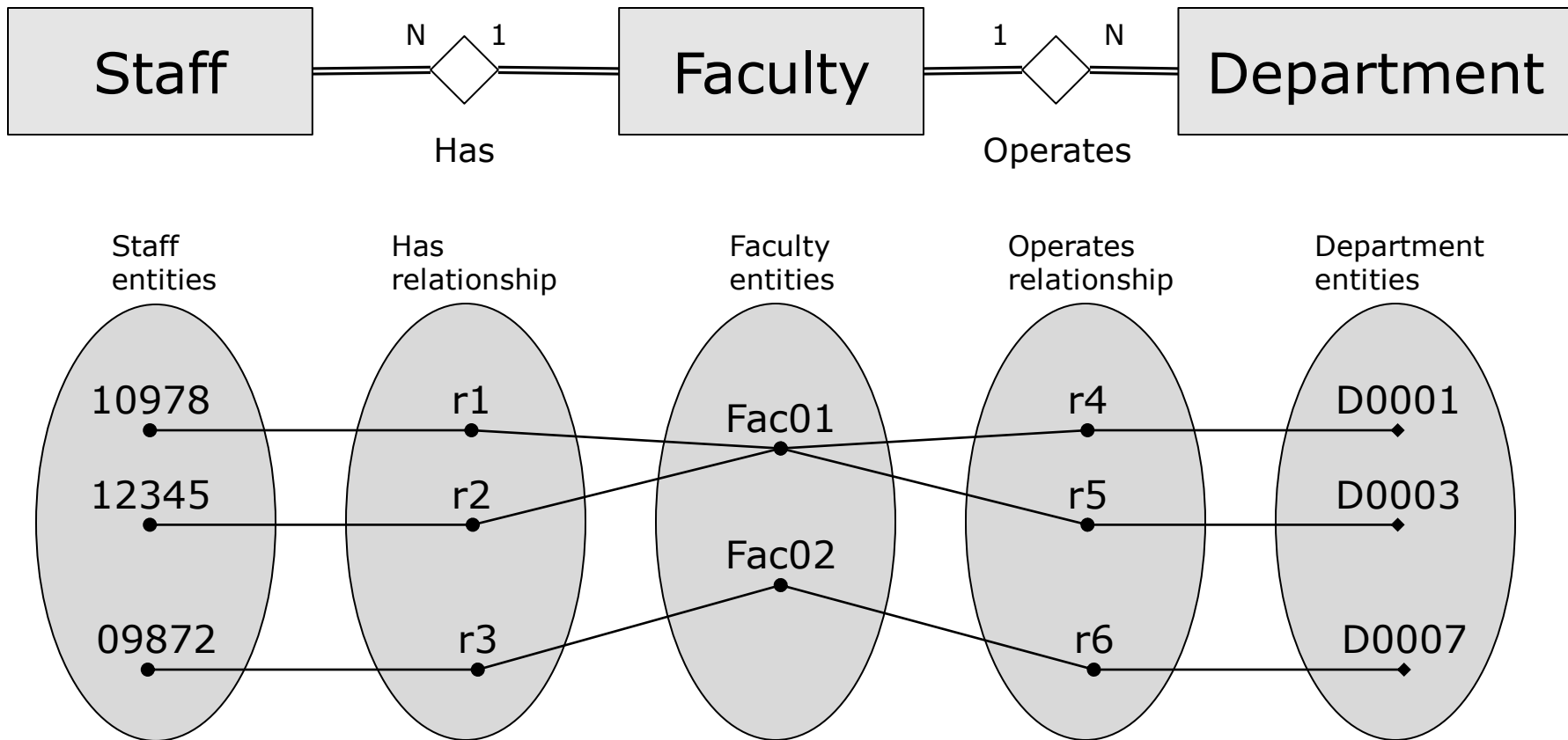- 2.9. Connection Traps
- Fan Trap
  - Where a diagram represents a relationship between entity types, but pathway between certain entity occurrences is ambiguous
  - Usually: two or more 1:N (M:N) relationships fan out from the same entity
- Chasm Trap
  - Where a diagram suggests the existence of a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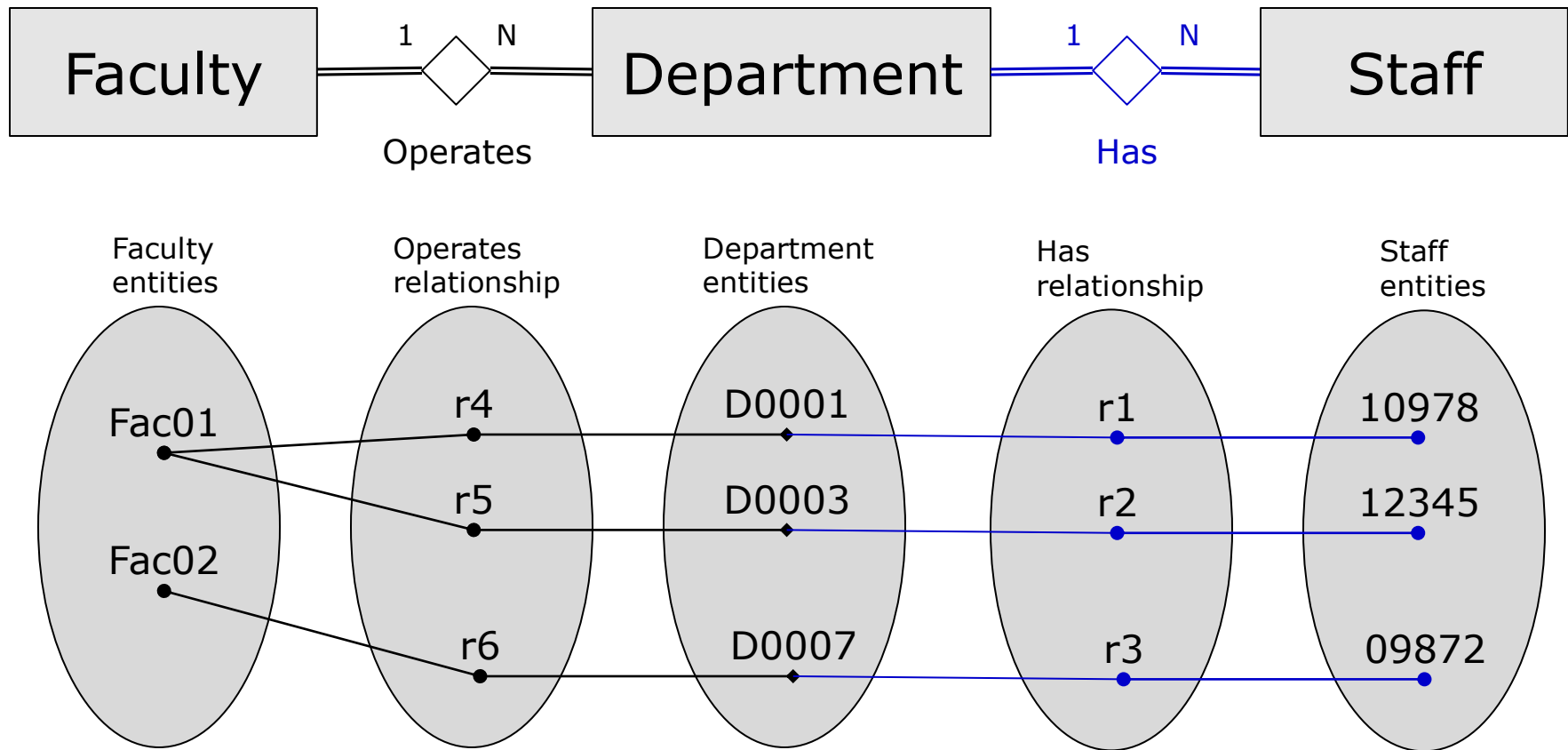 - Usually: optional participation

Review

An Example of a Fan Trap

| Staff | N | 1 | Faculty | 1 | N | Department |

Has   Operates

Staff entities — Has relationship — Faculty entities — Operates relationship — Department entities

10978   r1   Fac01   r4   D0001

12345   r2   r5   D0003

        Fac02

09872   r3   r6   D0007

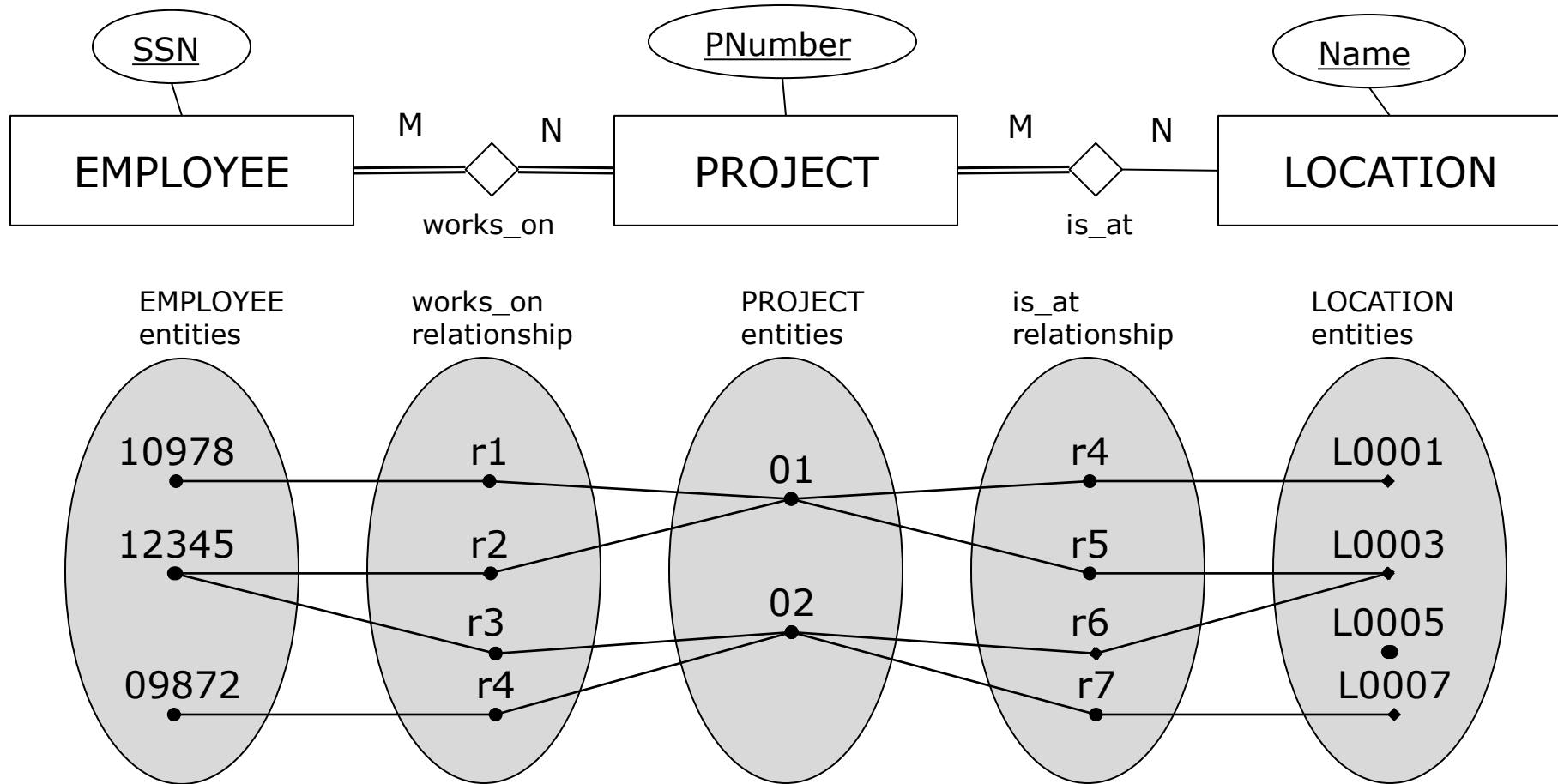**At which department does staff number 12345 work?**

Review

Restructured to remove the Fan Trap
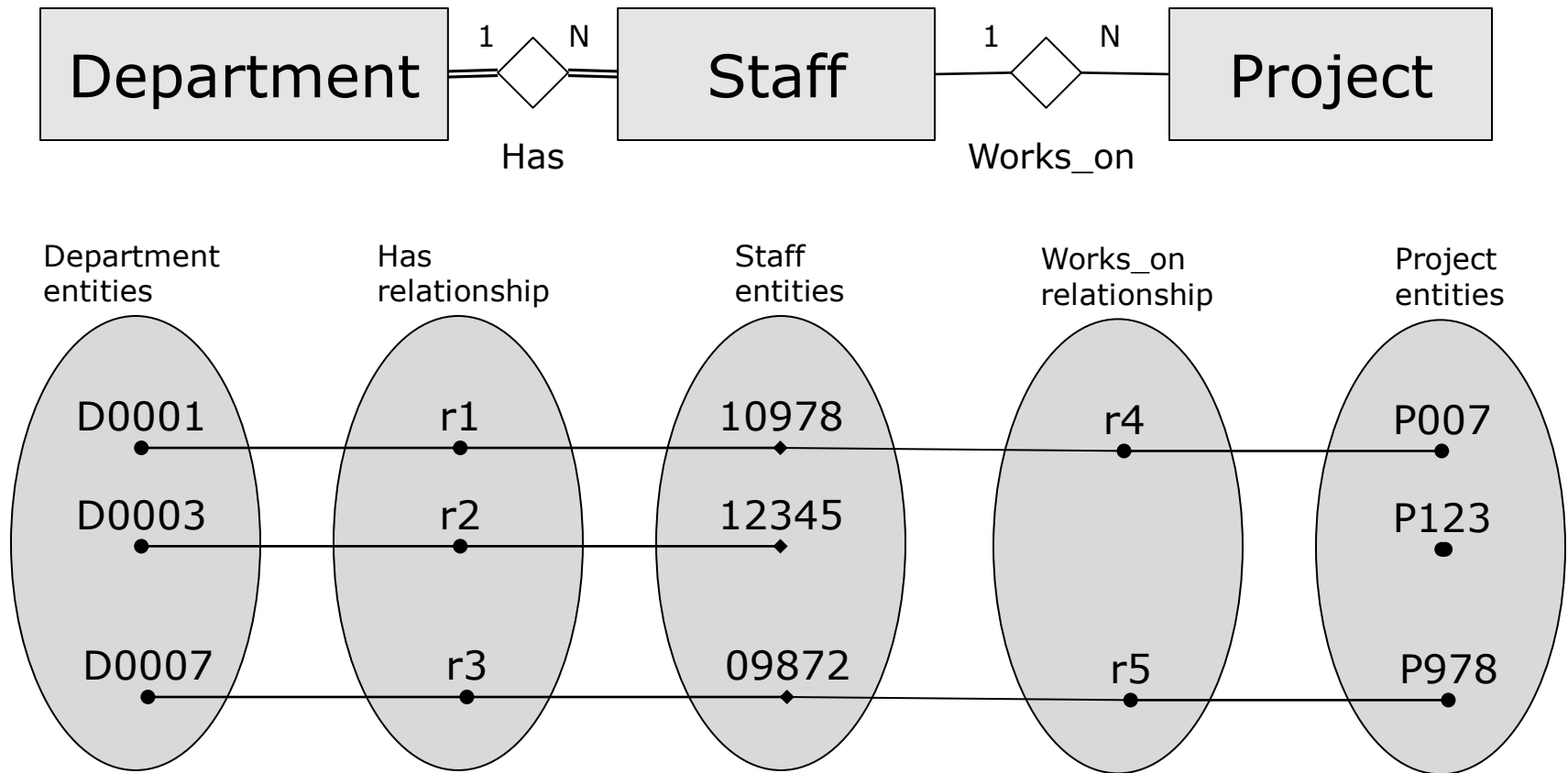

12345 works at department D0003.

Review

An Example of a Fan Trap

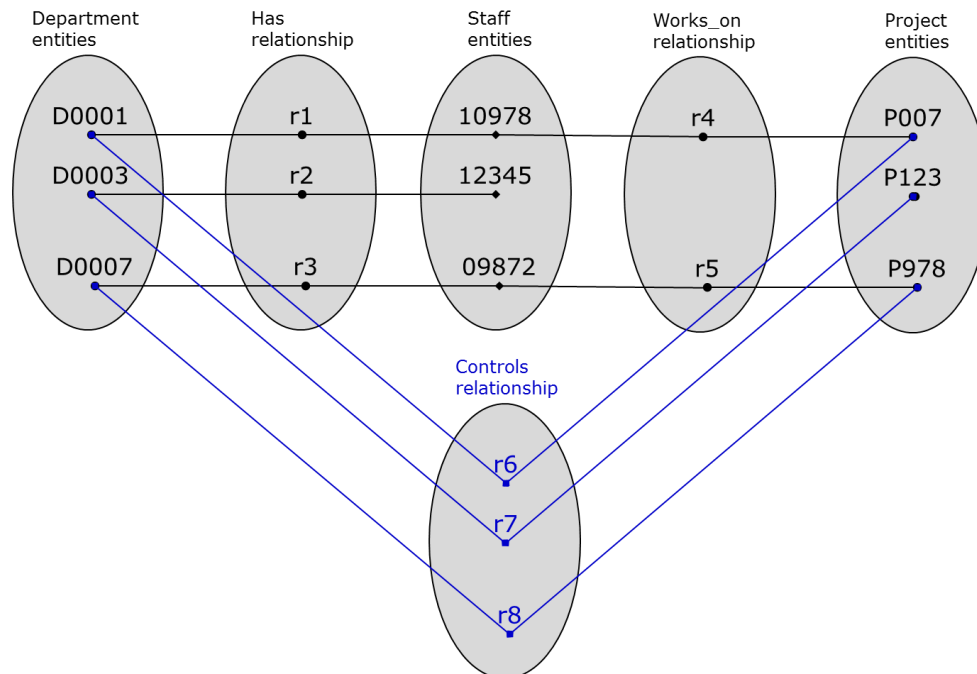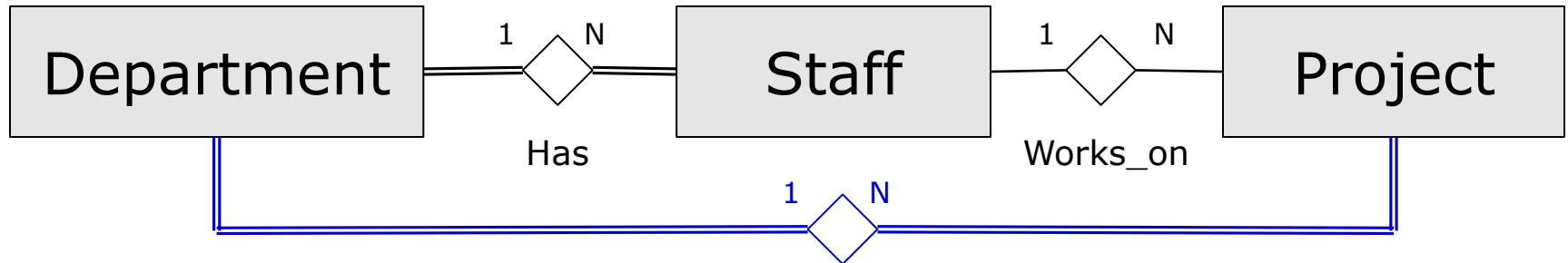**At which location does employee 12345 work?**

Review

An Example of a Chasm Trap

Department entities	Has relationship	Staff entities	Works_on relationship	Project entities
D0001	r1	10978	r4	P007
D0003	r2	12345		P123
D0007	r3	09872	r5	P978

Which department does project P123 belong to?

Review

Restructured to remove the Chasm Trap

Project P123 belongs to department D0003.

Next
Chapter 3: The Relational Data Model

- 3.1. Concepts

- 3.2. Relation schemas. Relations

- 3.3. Mapping an entity-relationship schema into a relational database schema

- 3.4. The Relational algebra