



DATABASE MANAGEMENT SYSTEM 2015

by Jean Baptiste MINANI

Email: baptisteauca@gmail.com

Matthew 19:26 “With men this is impossible; but
with God all things are possible”

Why Data Modeling?

- Represent “reality” of the actual database
- **Blue print:** documentation
- Effective Communication Tool
- User involvement
- Represent abstraction of requirements
- Identify the business rules to be stored in the database
- Independence from a particular DBMS

The Entity-Relationship model

- The E-R model is a detailed, logical representation of the data for an organisation or business area
- It should be understandable to both the user and to the IT technologist
- The model must be as 'open' as possible and not tied to any technology or to any particular business methodology
- It must be flexible enough so that it can be used and understood in practically any environment where information is modelled

The ER model

- It is expressed in terms of entities in the business environment, the relationships (or associations) among those entities and the attributes (properties) of both the entities and their relationships
- The E-R model is usually expressed as an E-R diagram

ER

- 1976 proposed by Peter Chen
- ER diagram is widely used in database design
 - Represent conceptual level of a database system
 - Describe things and their relationships in high level

Basic Concepts

- Entity set – an abstraction of similar things, e.g. cars, students
 - An entity set contains many entities
- Attributes: common properties of the entities in a entity sets
- Relationship – specify the relations among entities from two or more entity sets

E-R Model Constructs

- Entity - person, place, object, event, concept
- Entity Type - is a collection of entities that share common properties or characteristics. Each entity type is given a name, since this name represents a set of items, it is always singular. It is placed inside the box representing the entity type (Fig. 3-1)
- Entity instance – is a single occurrence of an entity type. An entity type is described just once (using metadata) in a database, while many instances of that entity type may be represented by data stored in the database. e.g. – there is one EMPLOYEE entity type in most organisations, but there may be hundreds of instances of this entity stored in the database

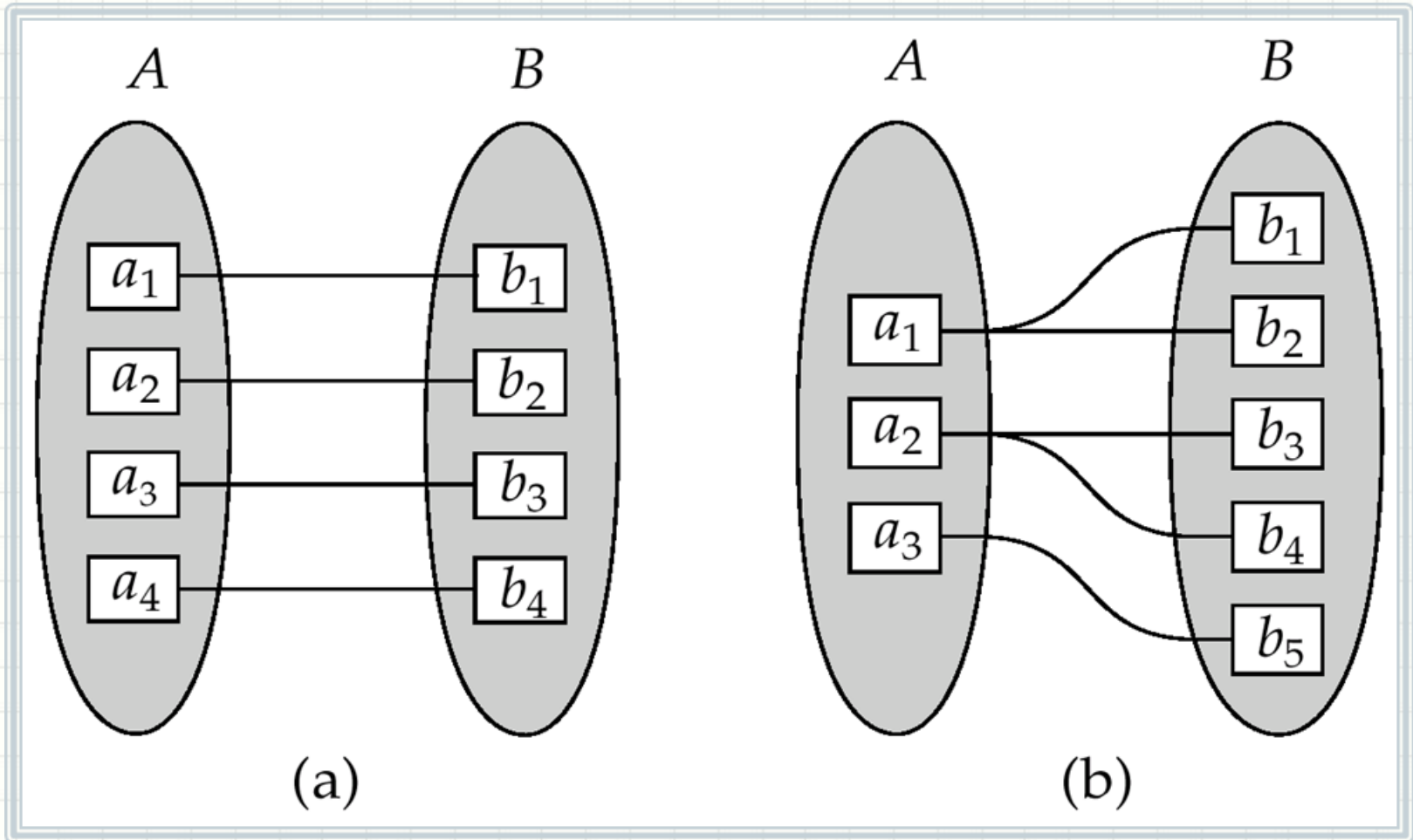
Relationship

- A relationship may be thought as a set as well
 - For binary relationship, it enumerates the pairs of entities that relate to each other
 - For example, entity set $M = \{Mike, Jack, Tom\}$ entity set $F = \{Mary, Kate\}$. The relationship set *married* between M and F may be $\{ \langle Mike, Mary \rangle, \langle Tom, Kate \rangle \}$

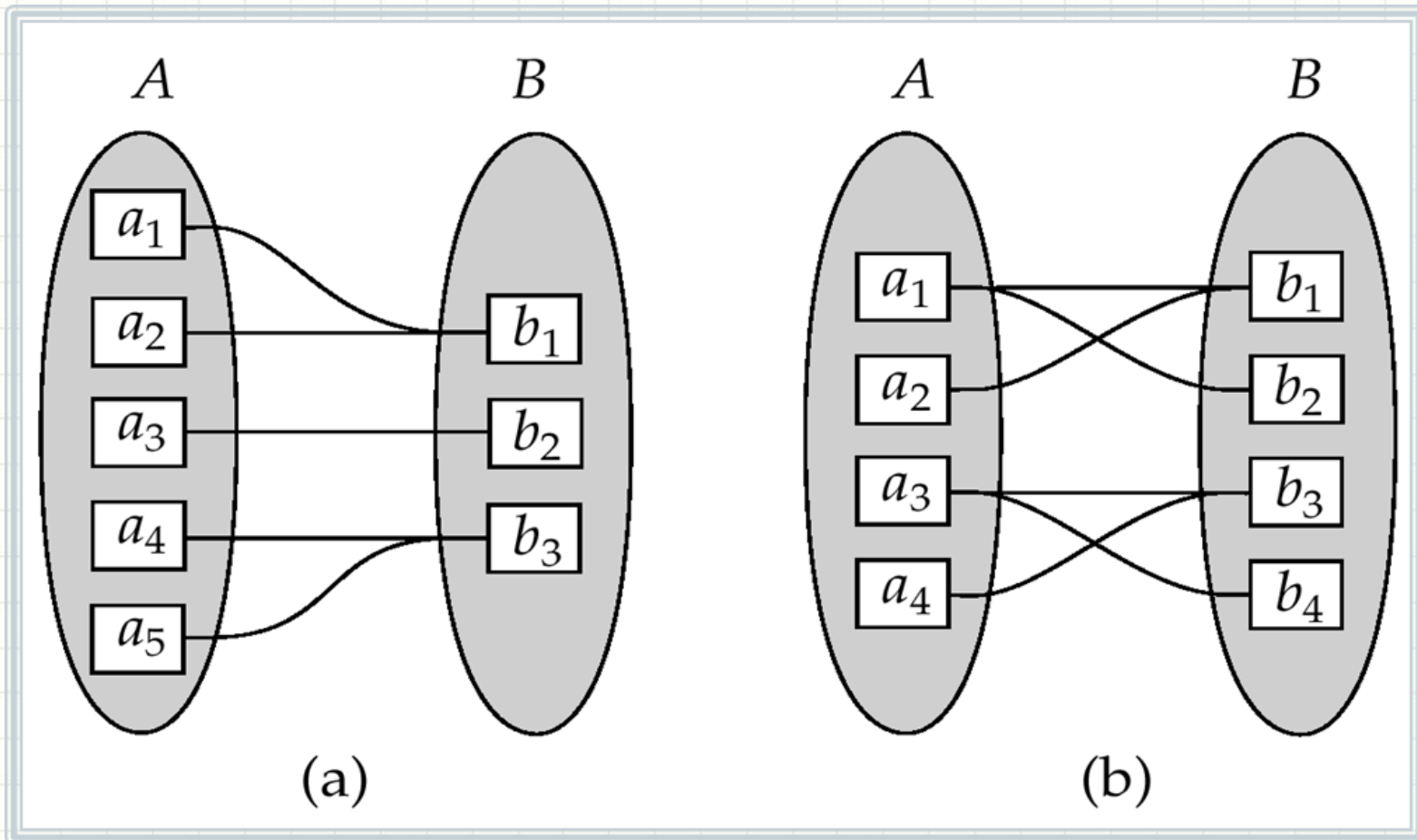
Relationship

- The degree of a relationship = the number of entity sets that participate in the relationship
 - Mostly binary relationships
 - Sometimes more
- Mapping cardinality of a relationship
 - 1 – 1
 - 1 – many
 - many – 1
 - Many-many

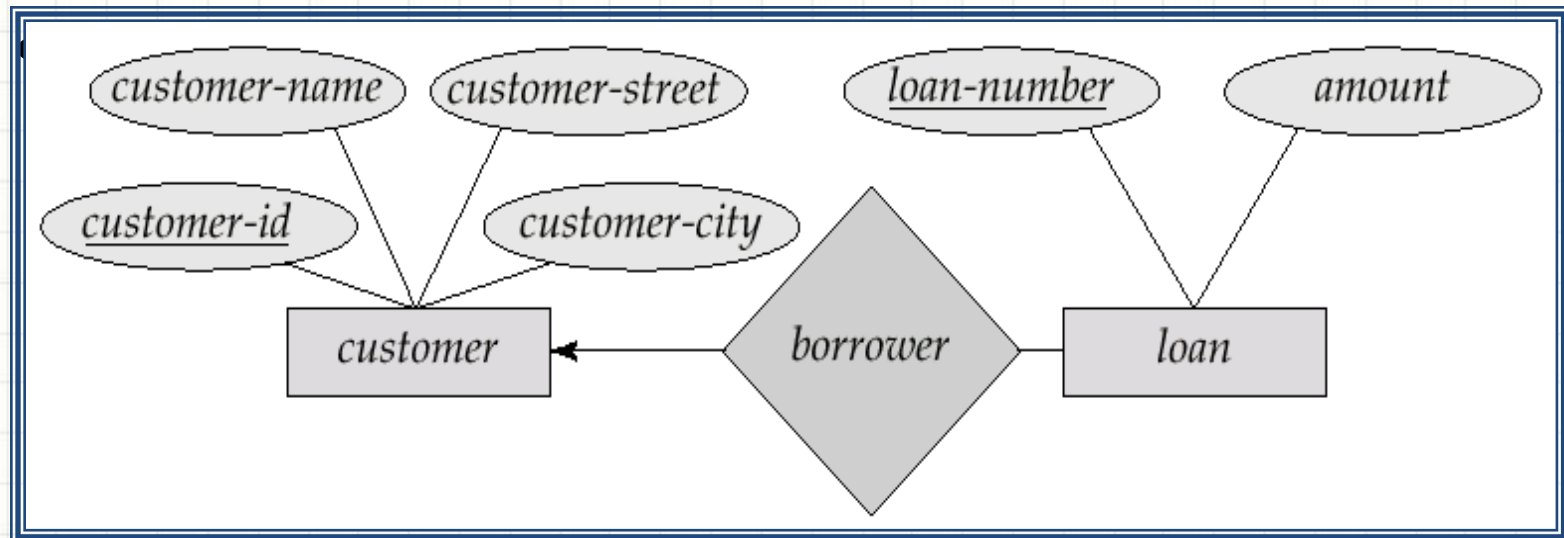
One-One and One-Many



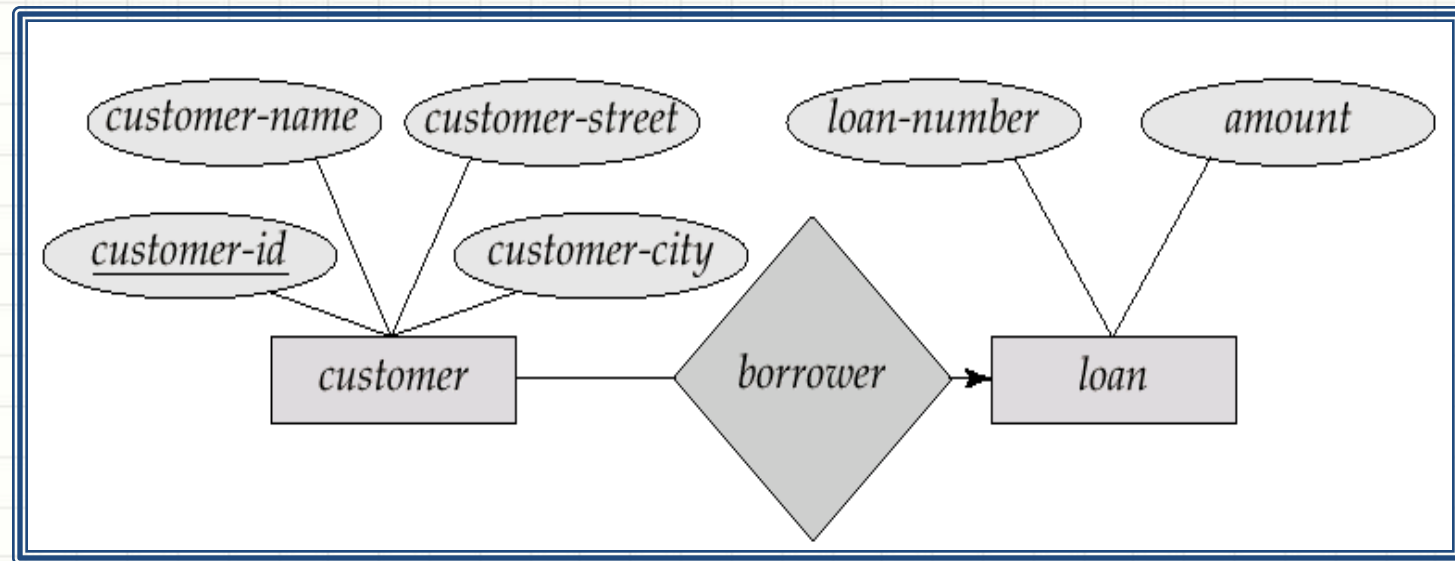
Many-one and many-many



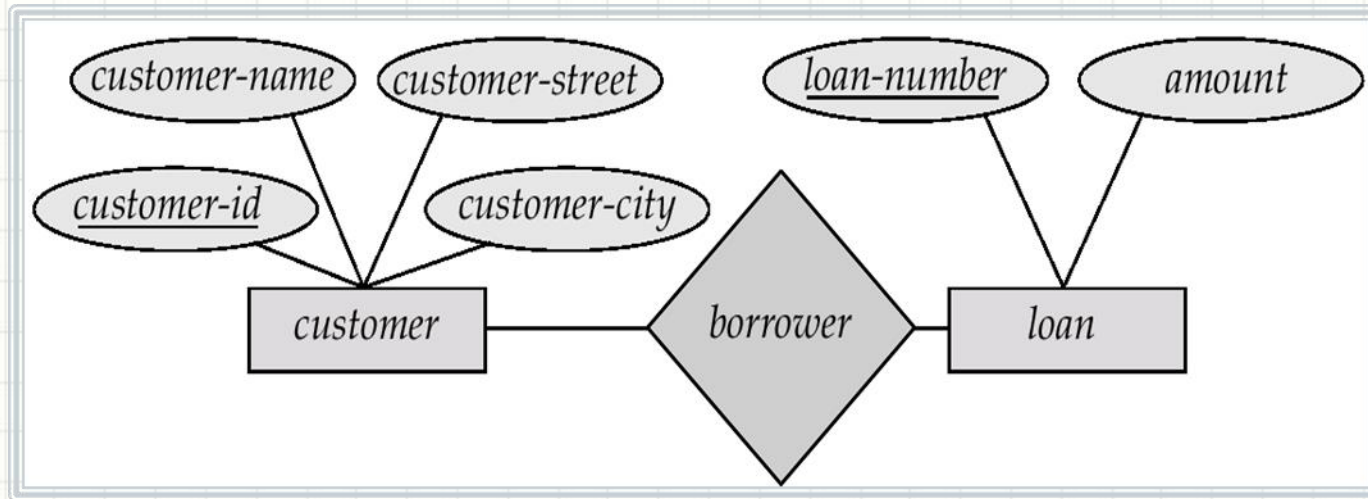
1- many



Many - 1



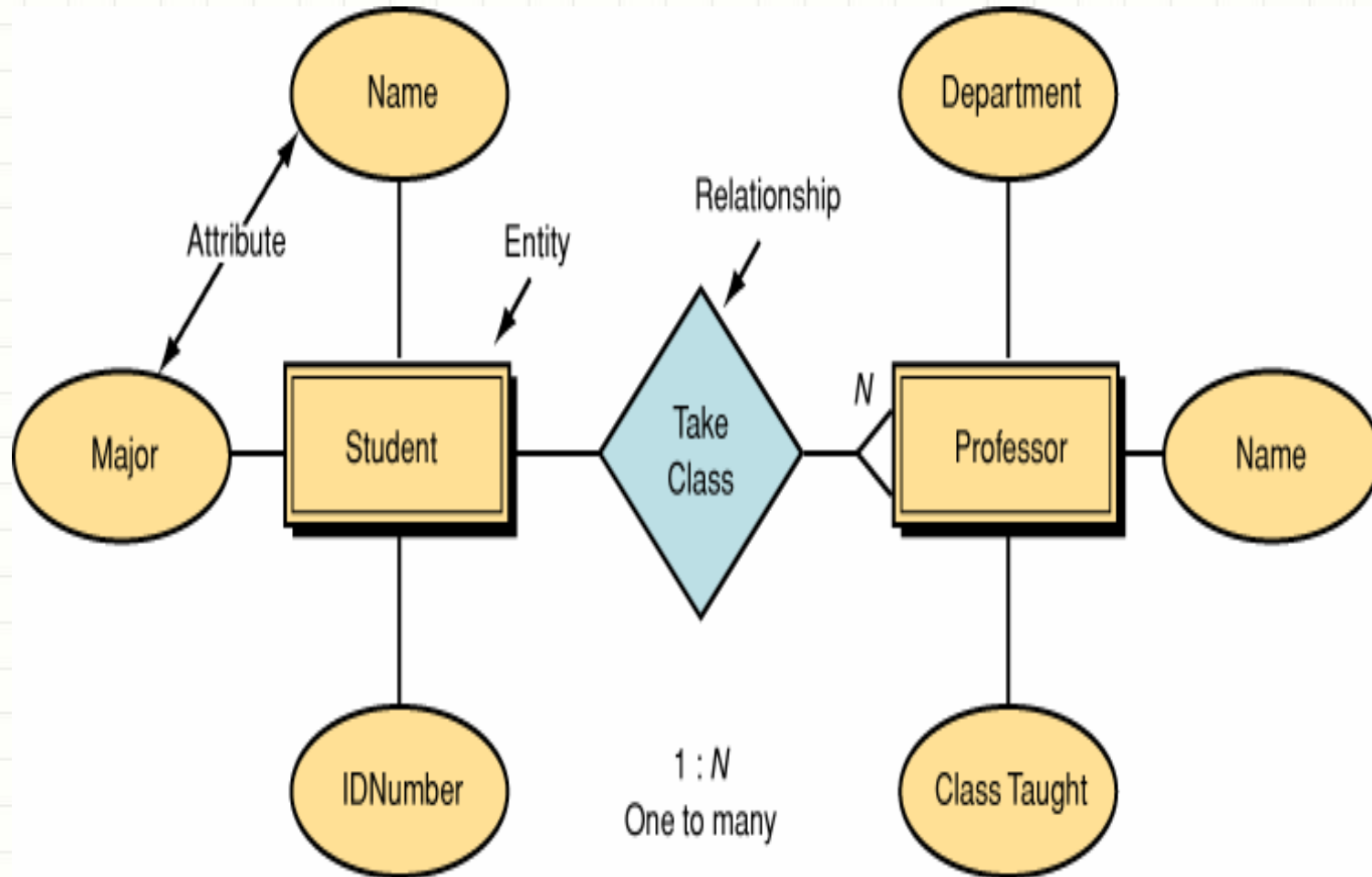
Many - many



Basic ERD Elements in summary

- **Entity** : a collection of people, places, objects, events, concepts of interest (a table)
 - **Entity instance** – a member of the Entity : a person, a place, an object ... (a row in a table)
- **Attribute** - property or characteristic of interest of an entity (a field in a table)
- **Relationship** – association between entities (corresponds to primary key-foreign key equivalencies in related tables)

ERD using Chen' Notation (first - original)



Chen's Notation

- Entities
 - rectangle containing the entity's name.
- Attributes
 - oval containing the attribute's name.
- Relationships
 - diamond containing the relationship's name.

Steps for creating an ERD

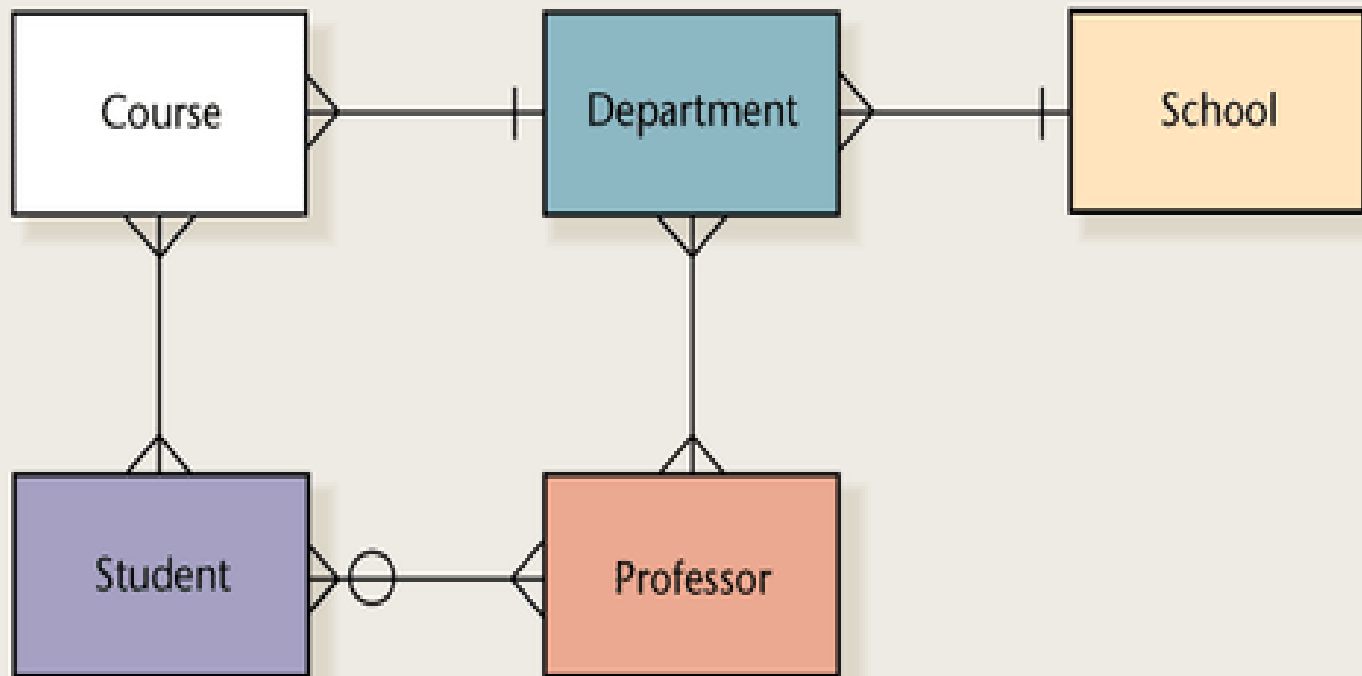
1. Identify entities
2. Identify attributes
3. Identify relationships

Entity

*“A fundamental **THING** of relevance to the enterprise about which data may be kept”*

- What should be an Entity: **both tangible & intangible**
 - An object that will have many instances in the database
 - An object that will be composed of multiple attributes
 - An object that we are trying to model
- What should NOT be an Entity:
 - A user of the database system
 - An output of the database system (e.g. a report)

ERD using IE Notation (most popular)



Attributes

“describe property or characteristic of an entity ”

- **Entity:** Employee
- **Attributes:**
 - Employee-Name
 - Address (composite)
 - Phone Extension
 - Date-Of-Hire
 - Job-Skill-Code
 - Salary

Classes of attributes

- Simple attribute
- Composite attribute
- Derived attributes
- Single-valued attribute
- Multi-valued attribute

Simple/Composite attribute

- A **simple attribute** cannot be subdivided.
 - Examples: Age, Gender, and Marital status
- A **composite attribute** can be further subdivided to yield additional attributes.
 - Examples:
 - ADDRESS --→ Street, City, State, Zip
 - PHONE NUMBER --→ Area code, Exchange number

Derived attribute

- is not physically stored within the database
- instead, it is derived by using an algorithm.
 - Example: AGE can be derived from the date of birth and the current date.
 - MS Access: $\text{int}(\text{Date}() - \text{Emp_Dob})/365$

Single-valued attribute

- can have only a single (atomic) value.
 - Examples:
 - A person can have only one social security number.
 - A manufactured part can have only one serial number.
 - **A single-valued attribute is not necessarily a simple attribute.**
 - Part No: CA-08-02-189935
 - Location: CA, Factory#:08, shift#: 02, part#: 189935

Multi-valued attributes

- can have many values.
 - Examples:
 - A person may have several college degrees.
 - A household may have several phones with different numbers
 - A car color

Example - "Movie Database"

- Entity:
 - Movie Star
- Attributes:
 - SS#: "123-45-6789" (single-valued)
 - Cell Phone: "(661)123-4567, (661)234-5678" (multi-valued)
 - Name: "Harrison Ford" (composite)
 - Address: "123 Main Str., LA, CA" (composite)
 - Birthdate: "1-1-50" (simple)
 - Age: 50 (derived)

How to find entities?

- Entity:
 - A fundamental **THING** of relevance to the enterprise about which data may be kept: things acted on by business activities
 - people, places, objects, events....
 - Tangible: customer, product
 - intangible (active/conceptual): equipment breakdown
 - **look for nouns (beginner)** BUT a proper noun is not a good candidate....

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 (Cont.)

- 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.

Initial Design of Entity Types for the COMPANY Database Schema

- Based on the requirements, we can identify four initial entity types in the COMPANY database:
 - DEPARTMENT
 - PROJECT
 - EMPLOYEE
 - DEPENDENT
- Their initial design is shown on the following slide
- The initial attributes shown are derived from the requirements description

Department

- 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. A department may have several locations.

Projects

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

EMPLOYEE

- We store each EMPLOYEE's name, 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.

Dependent

- Each employee may *have* a number of DEPENDENTS.
 - For each dependent, we keep track of their name, sex, birthdate, and relationship to the employee.

Initial Design of Entity Types: EMPLOYEE, DEPARTMENT, PROJECT, DEPENDENT

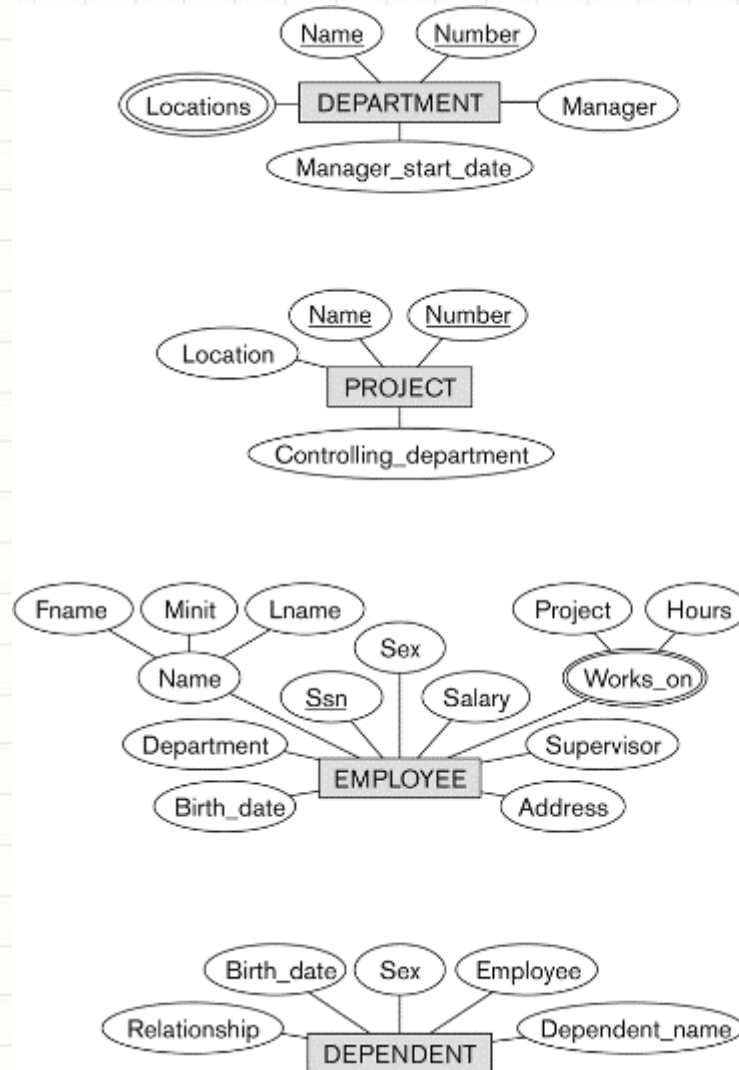


Figure 3.8

Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

Relationship

- The initial design is typically not complete
- Refining the initial design by introducing **relationships**
- ER model has three main concepts:
 - Entities (and their entity types and entity sets)
 - Attributes (simple, composite, multivalued)
 - Relationships (and their relationship types and relationship sets)

Relationship type vs. relationship set

- Relationship Type:
 - Is the **schema** description of a relationship
 - Identifies the relationship name and the participating entity types
 - Also identifies certain relationship constraints
- Relationship Set:
 - The current set of relationship instances represented in the database

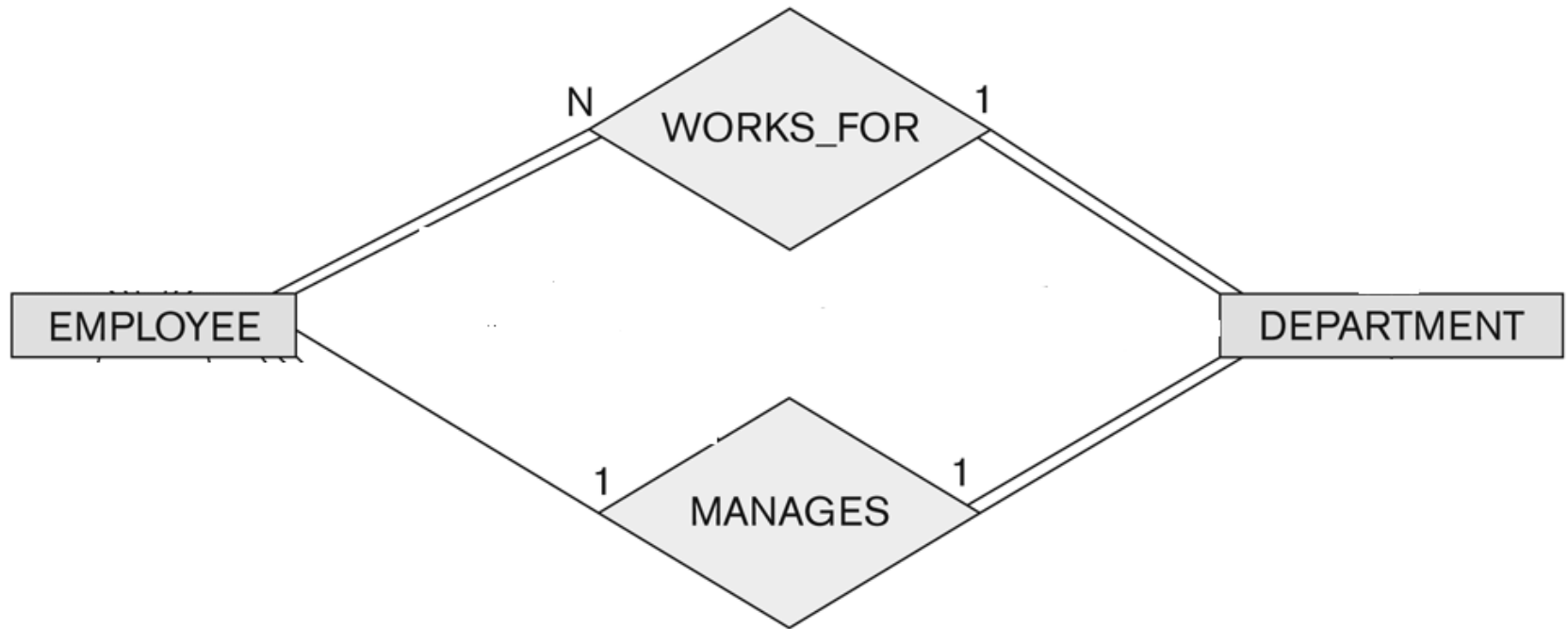
Relationship

- In ER diagrams, we represent the *relationship type* as follows:
 - Diamond-shaped box is used to display a relationship type
 - Connected to the participating entity types via straight lines

Relationship example

- Consider a relationship type **work_for** between the two entities type EMPLOYEE and DEPARTMENT
- Each relationship instance in the relationship set associates one EMPLOYEE entity and one DEPARTMENT entity

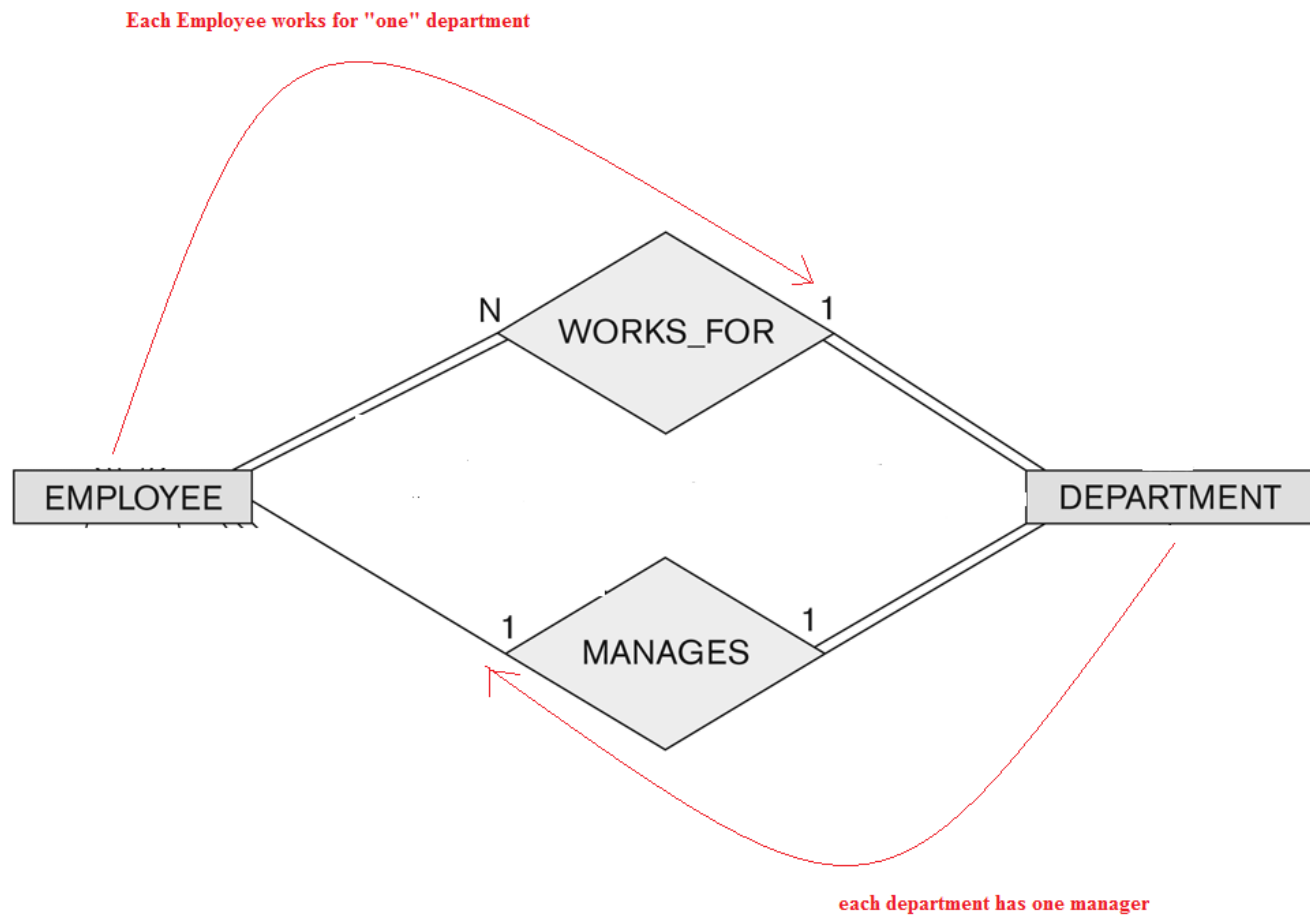
Relationship between EMPLOYEE and DEPARTMENT



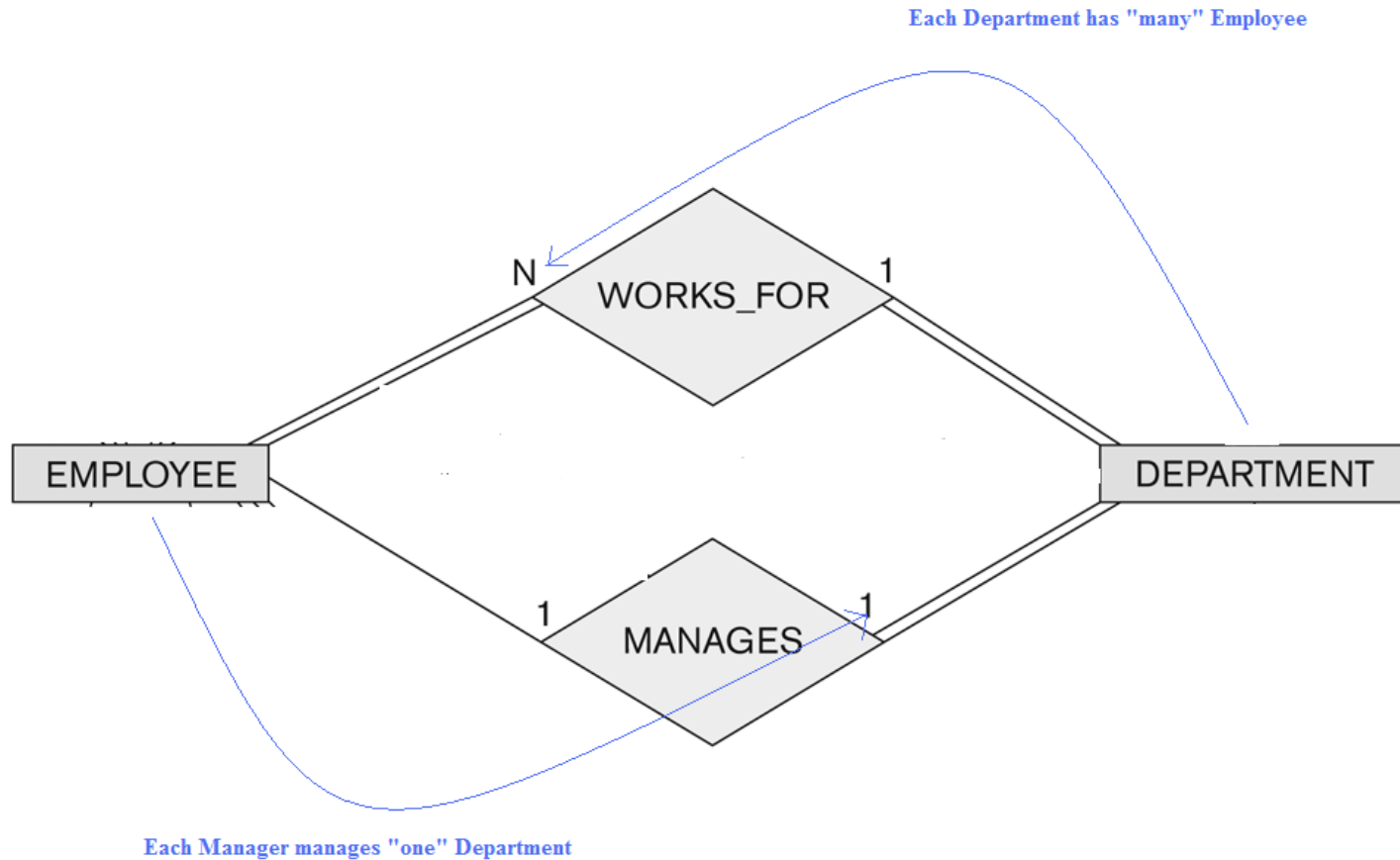
Constraints on Relationship types

- Sometimes if we want to describe “each employee must work for exactly one department”, then we would like to describe this constrain in the schema
- The **cardinality ratio** for a binary relationship specifies the max number of relationship instances that an entity can participate in.
- For example---in the Works_for binary relationship, DEPARTMENT:EMPLOYEE is of cardinality ration 1:N, meaning each department can be related to any number of employees, but an employee can only be related to one department

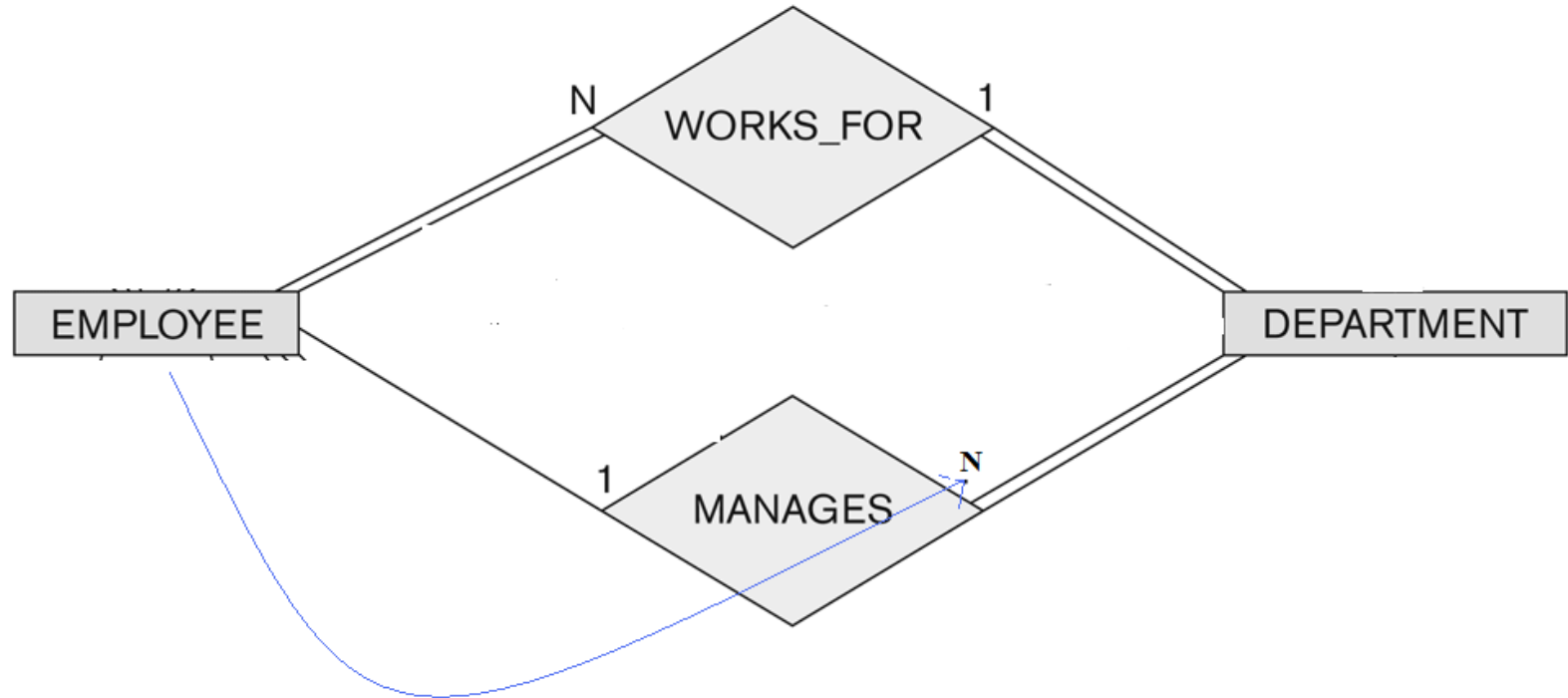
Relationship between EMPLOYEE and DEPARTMENT



Relationship between EMPLOYEE and DEPARTMENT



Relationship between EMPLOYEE and DEPARTMENT



Each Manager manages MANY Department

cardinality ratio

- The possible **cardinality ratio** for binary relationships are 1:1, 1:N, N:1, M:N
- Example:
 - 1:1 Manages relationship between employee and department
 - M:N an employee can work on several projects and a project can have several employees

Participation Constrain

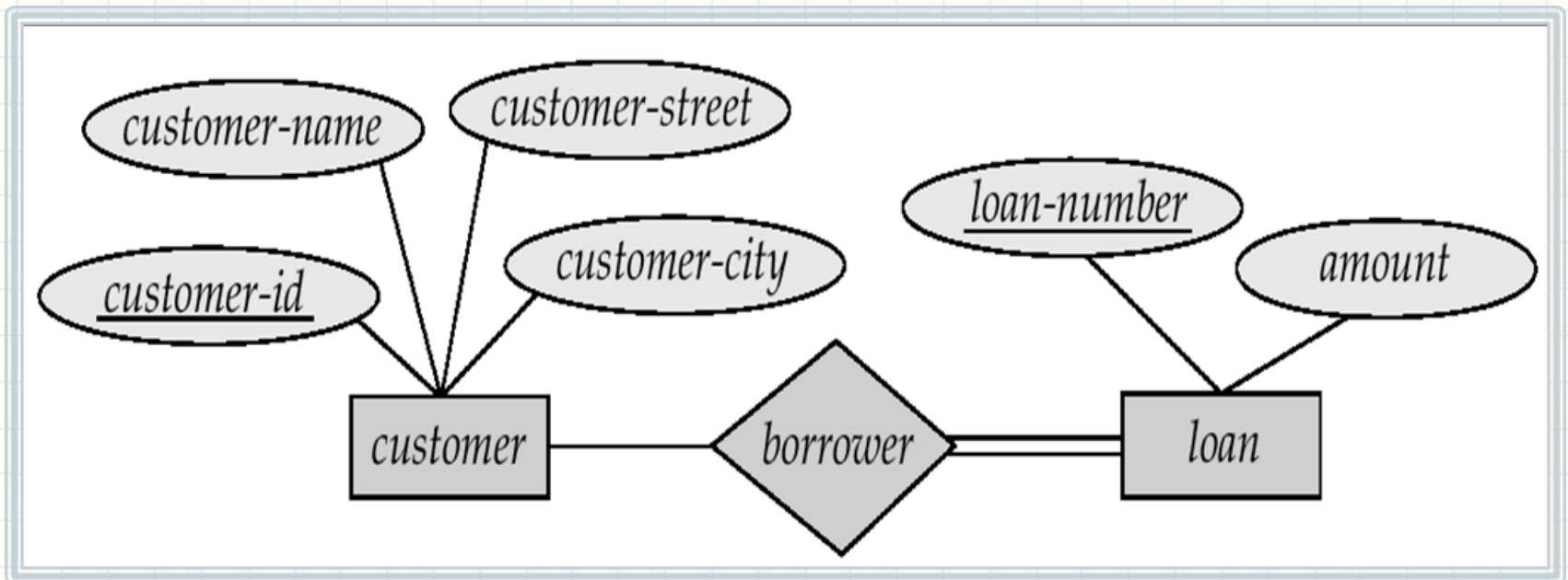
- The participation constrain specifies whether the existence of an entity depends on its being related to another entity via the relationship type
- There are two types of participation constrains:
 - **Total**
 - **Partial**

Total Participation

- When we require all entities to participate in the relationship (total participation), we use double lines to specify

-

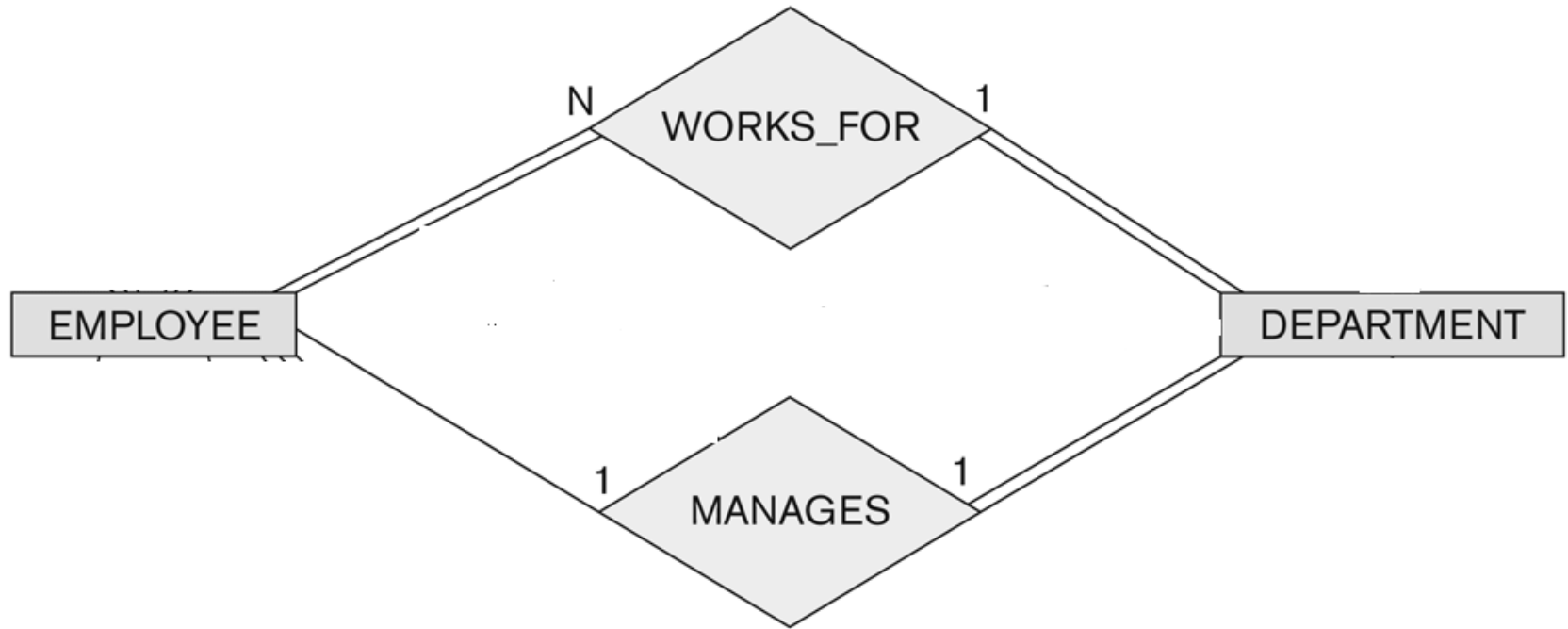
Every loan has to have at least one customer



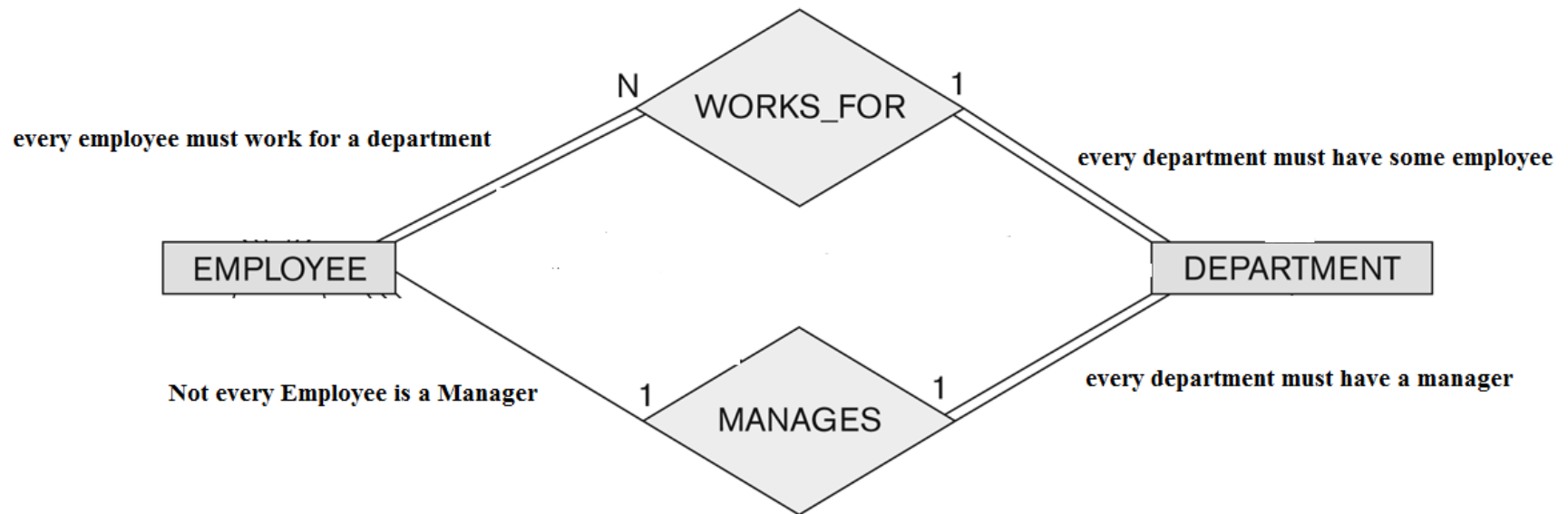
Participation Constrain

- For example
- If a company policy states that **every** employee must work for a department, then it's **total**
- **Not every** Employee is a Manager, so this relationship is **partial**

Relationship between EMPLOYEE and DEPARTMENT

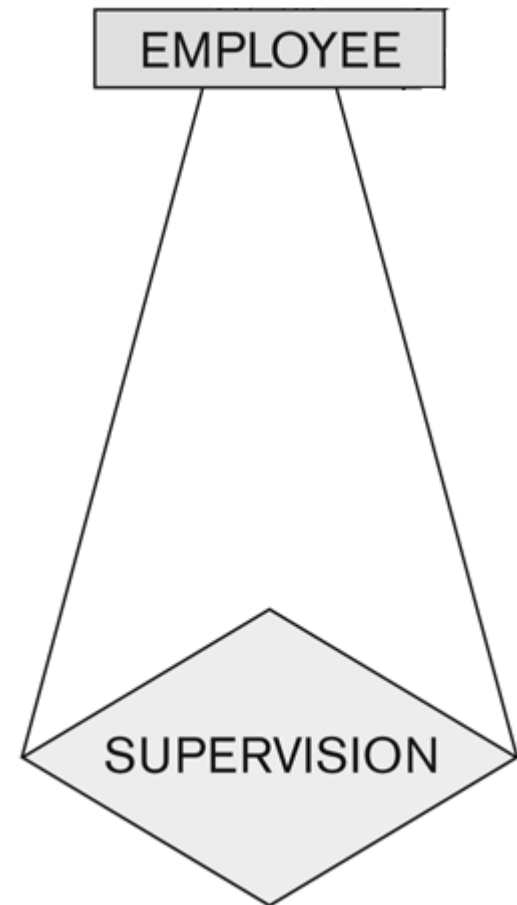


Relationship between EMPLOYEE and DEPARTMENT



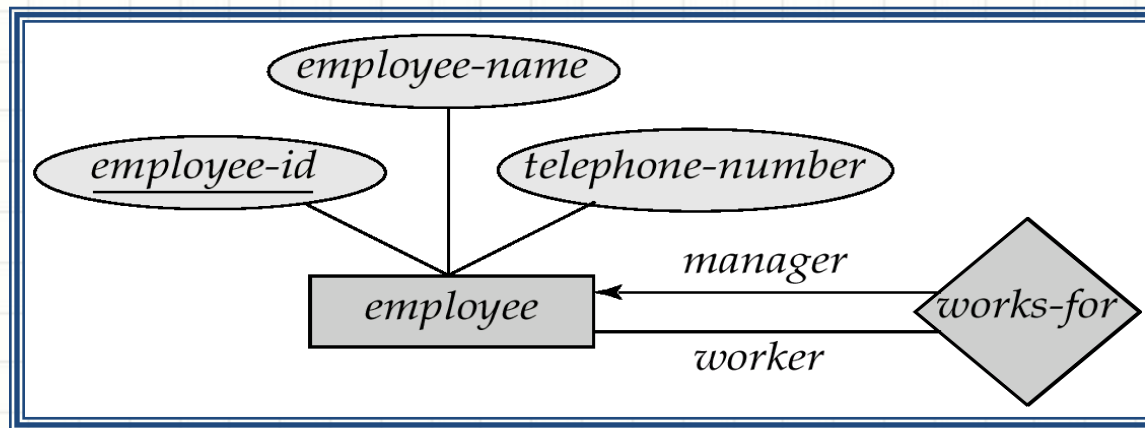
Recursive Relationship(Self Relationship)

- In some cases, the same entity type participates more than once in a relationship type in **different** roles
- Example Employee and supervised



Self Relationship

- Sometimes entities in a entity set may relate to other entities in the same set. Thus self relationship
- Here employees manage some other employees
- The labels “manger” and “worker” are called *roles* the self relationship



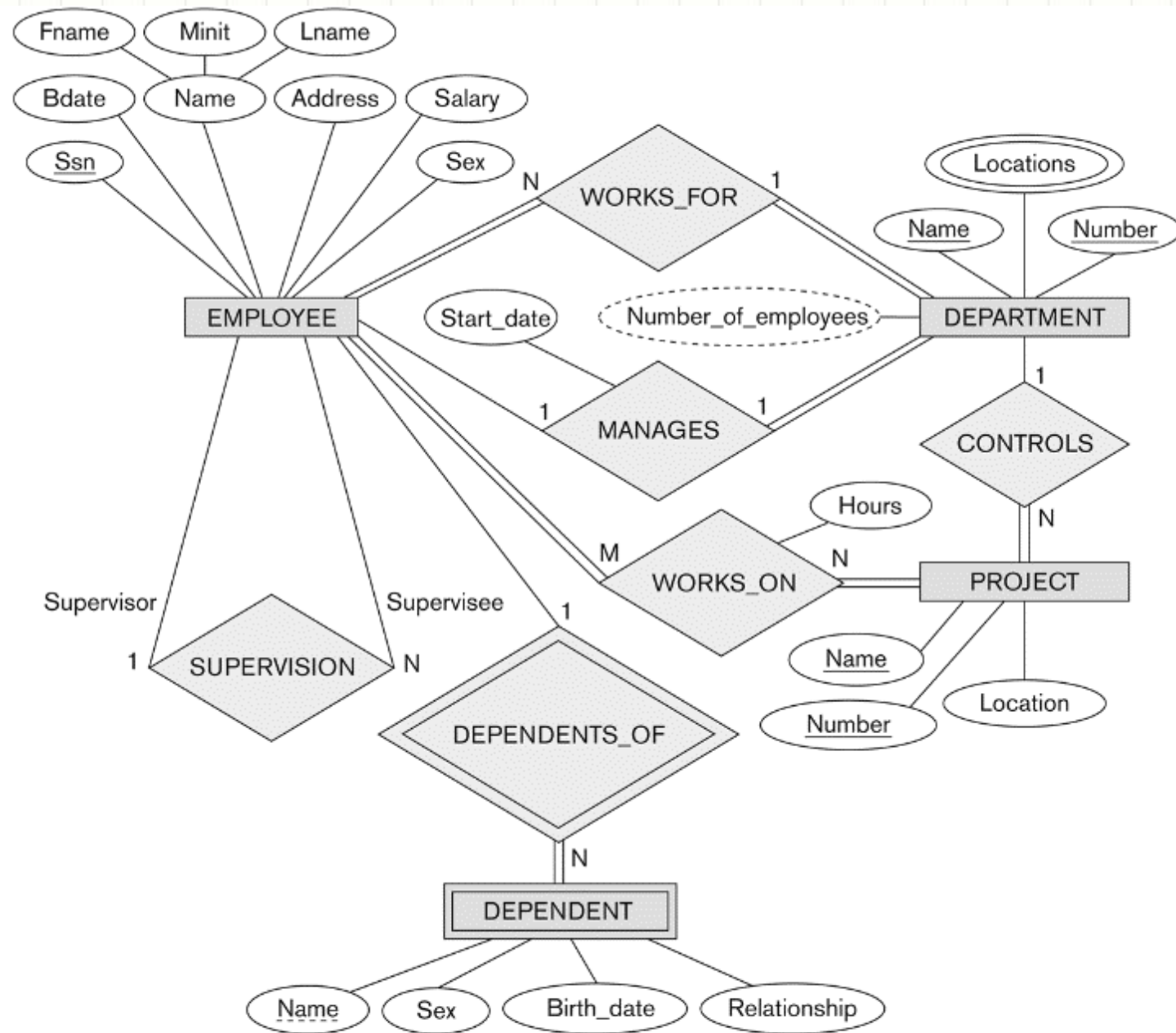


Figure 3.2

An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

Weak Entity Types

- Entity types do not have key attribute of their own are called **weak entity types**
- In contrast, regular entity types that do have key attribute are called **strong entity types**

Weak entity type Example

- Consider the entity type DEPENDENT, related to EMPLOYEE
- They are identified as distinct entities only after determining the particular employee entity to which dependent is related
- A weak entity type normally has a **partial key**, which is the set of attributes that can uniquely identify weak entities that are related to the same owner entity

Weak entity type representation in ER diagram

- In ER diagrams, both weak entity type and its identifying relationship are distinguished by surrounding their **boxes** and **diamonds** with double lines
- The partial key attribute is underlined with a **dashed** line

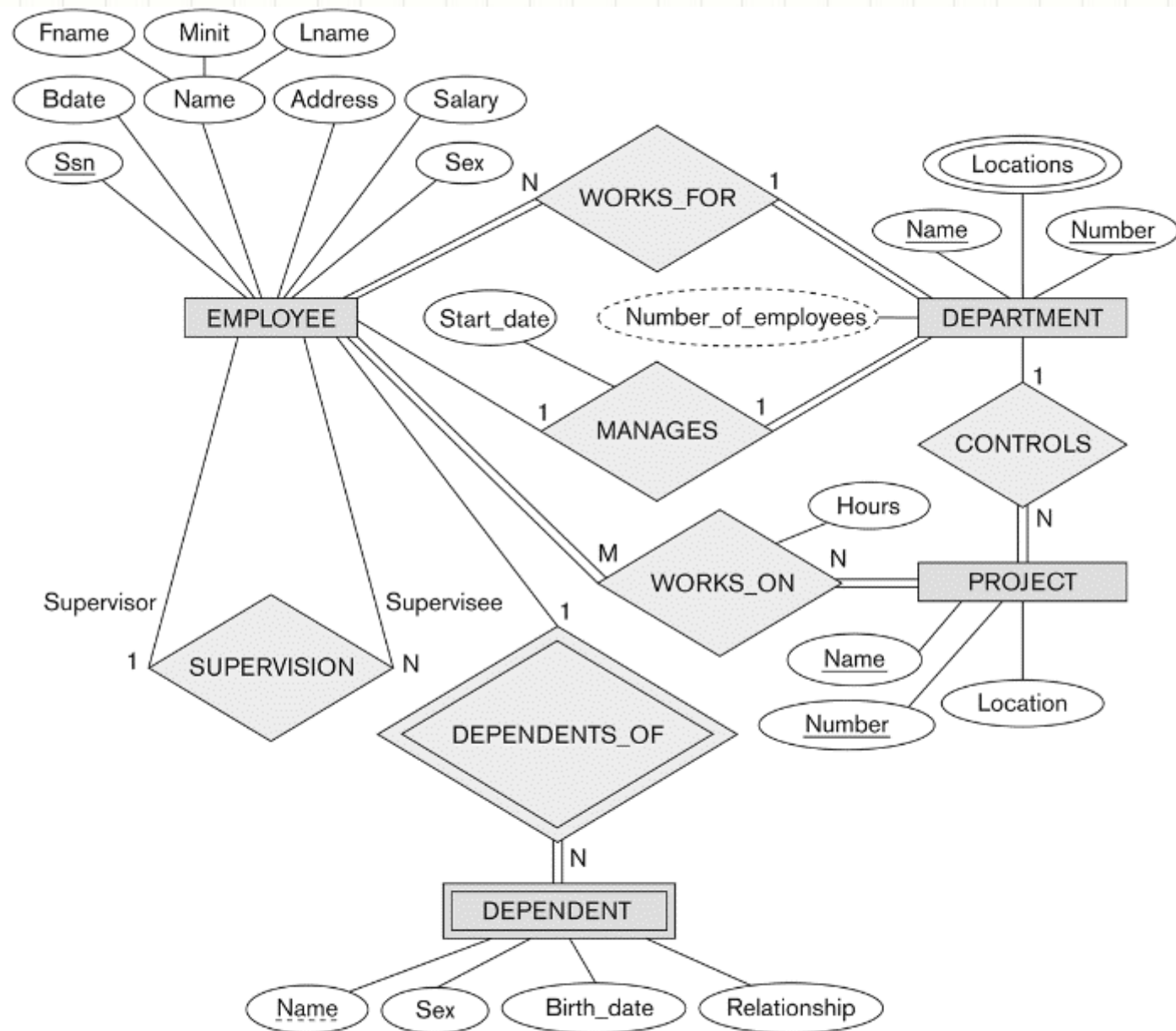


Figure 3.2

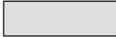
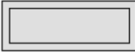





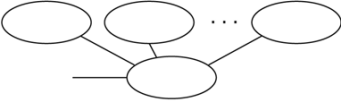



An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

Weak Entity Type

- Weak entity types can sometimes be represented as complex attributes
- **Complex Attributes:** combination of composite and multi-valued attributes
- In the example, we could specify a multi-valued attribute Dependents for EMPLOYEE, which is a composite attribute with component attributes Name, Birthday, Sex and Relationship
- The choice of which representation to use is made by the database designer

Summary of notation for ER diagrams

Figure 3.14
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

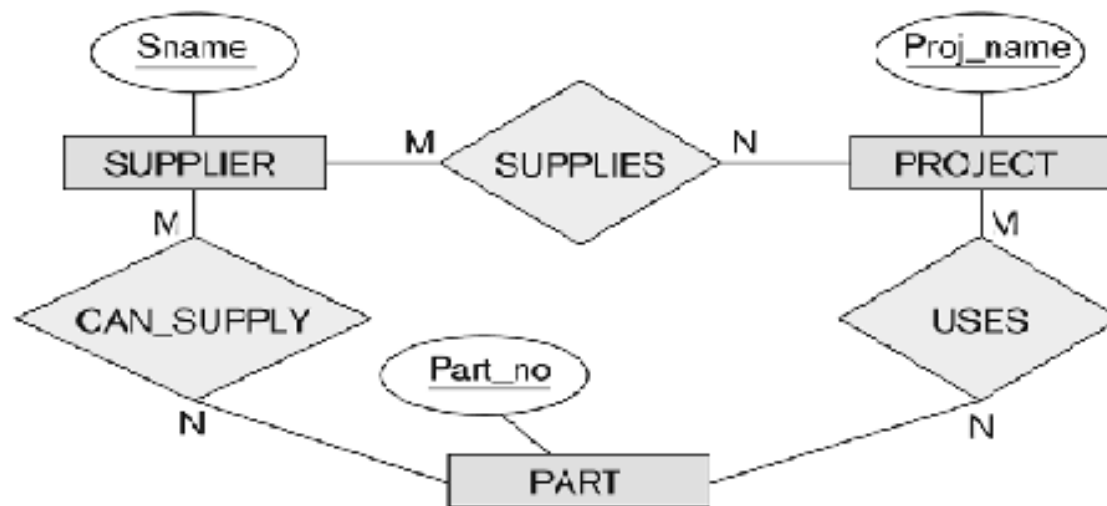
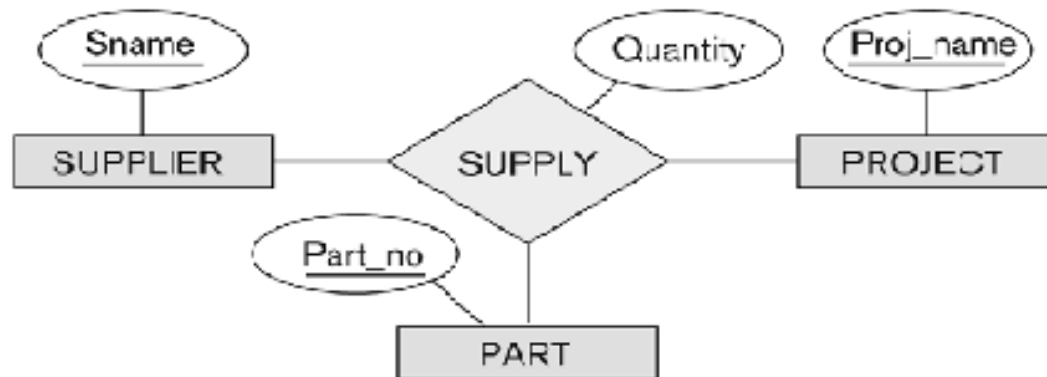
Relationships of Higher Degree

- 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

Choosing between Binary and Ternary relationships

- SUPPLY relationship, is a set of relationship instances (s, j, p) , where s is a SUPPLIER who is currently supplying a PART p to a PROJECT j
- In general, a relationship type R of degree n will have n edges in an ER diagram, one connecting R to each participating entity type

The Supply Relationship



Choosing between Binary and Ternary relationships

- The figure below shows an ER diagram for three binary relationship types CAN_SUPPLY, USES, SUPPLIES
- In general, 3 binary relationships can represent different information than a single ternary relationship
- If needed, the binary and n-ary relationships can all be included in the schema design

Another example of a ternary relationship

Figure 3.18

Another example of ternary versus binary relationship types.

