Lecture 2 Data Modeling using Entity Relational Models (Chapter 3)

Alisa Lincke (alisa.lincke@lnu.se)

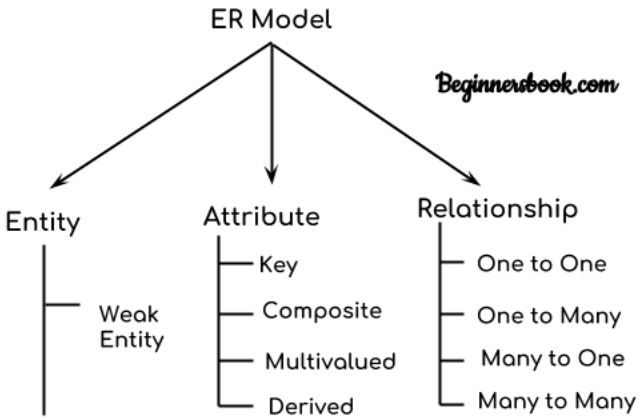
Linnæus University

Outline

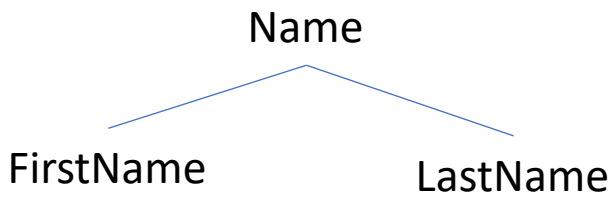
- ER Model Concepts
  - Entities and attributes
  - Entity Types, Value Sets, and Key Attributes
  - Relationships and Relationship Types
  - Weak Entity Types
  - Roles and Attributes in Relationship Types
- Break 10 min
- ER Diagrams –Notation
- Example COMPANY database and ER Diagram
- Assignment 1

ER Model Concepts

- Entities and Attributes:
  - Entity is a basic concept which is a ***thing*** or ***object*** in the real world with an independent existence. An entity may be an object with a *physical existence* (e.g., a particular person, car, animal) or it may be an object/thing with a *conceptual existence* (e.g., a company, a job, a university course).
  - Attributes are properties used to describe an object/thing (e.g., a person entity can be described by name, age, gender, etc.)
  - A particular entity will have a value for each of its attributes (e.g., a person has Name='John Smith', Age='45',Gender='Male')
  - Each attribute has a *value set* or *data type* associated with it (e.g., integer, enumerated type)



Types of Attributes (1)

Attribute Type	Example
Simple is a single atomic value for the attribute (e.g., number, string,)	Gender="Male" has one value in *string* type Age=20 attribute has one value '20' and it is *int* type
Composite can be composed of several subattributes with independent meanings. Composition may form a hierarchy with nested composite attributes	Name (FirstName and LastName) 
Multi-valued can contain multiple values for that attribute. My have lower and upper boundary.	Color of a Car, Hobbies, Roles, Interests. Car with one color counted as single-value attribute, car with having two colors counted as multi-valued attribute. A person may not have a degree, may have a one degree, or two degrees.

Example of a nested composite attribute

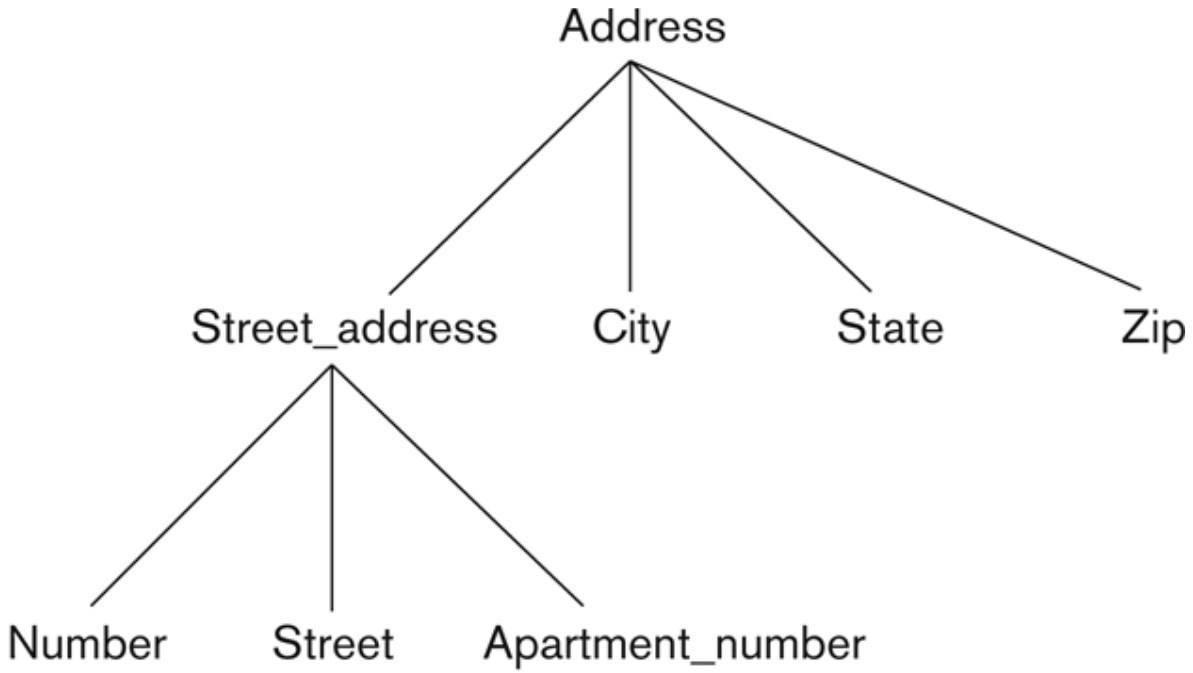


Figure 3.4
A hierarchy of composite attributes.

Types of Attributes (2)

Attribute Type	Example
Derived can be calculated from existing attributes	Person's Age can be derived from Birth_date attribute and current date. Person's BMI can be derived from person's Weight and Height attribute
NULL values when some entity does not have an applicable value to the attribute. A special value NULL is created in database	Person's phone number is NULL (is *unknown*) Person's apartment address is NULL ,because he/she lives in private house (apartment attribute does not exists for this person, *not applicable*)
Complex attributes are nested composite and multi-valued attributes	{Address_phone({Phone{Country_code,Phone_number), Address(Street_address(Number,Street,Apartment_number),City,Postal_Code)}

Entity Types and Key Attributes

- Entities with the same basic attributes are grouped into an entity type
  - Each entity described by *name* and set of *attributes*
  - For example, students entities share the same attributes and can be grouped to STUDENT entity type, but each entity has *its own values* for each attribute.
- The important constraint on the entities of an entity type is the ***key*** or **uniqueness constrain** on attributes.
- An entity **must** have at least ***one*** or ***more*** attributes whose values are **distinct** for each individual entity in the entity set.
- Such attribute (or set of attributes) called a **key** (or ***composite key***) and its values used to identify each entity uniquely.
  - For example, for Person entity type the unique key is social number (snn). VechilTagNumber is a composite key (Number, State)
- Key is not just a property of an entity but also a constraint on any entity set of the entity type.
- An entity type may have **more** than one key
- Each **key** in **underline** in ER diagram

Entity Set

- Each entity type will have a collection of entities stored in the database
  - Called the entity set or sometimes entity collection
- Entity type is a description of entity, while entity set is actual data stored in database structured in particular entity type.
- Entity set is the current state of the entities of that type that are stored in the database
  - For example, database contains one entity type Employee, and entity set of Employee type is 10 employees for some particular moment in time (e.g., one week ago). Another time, this value may change, so the entity set values are changing over time, but entity type description is not changing over time.

Value Sets of Attributes

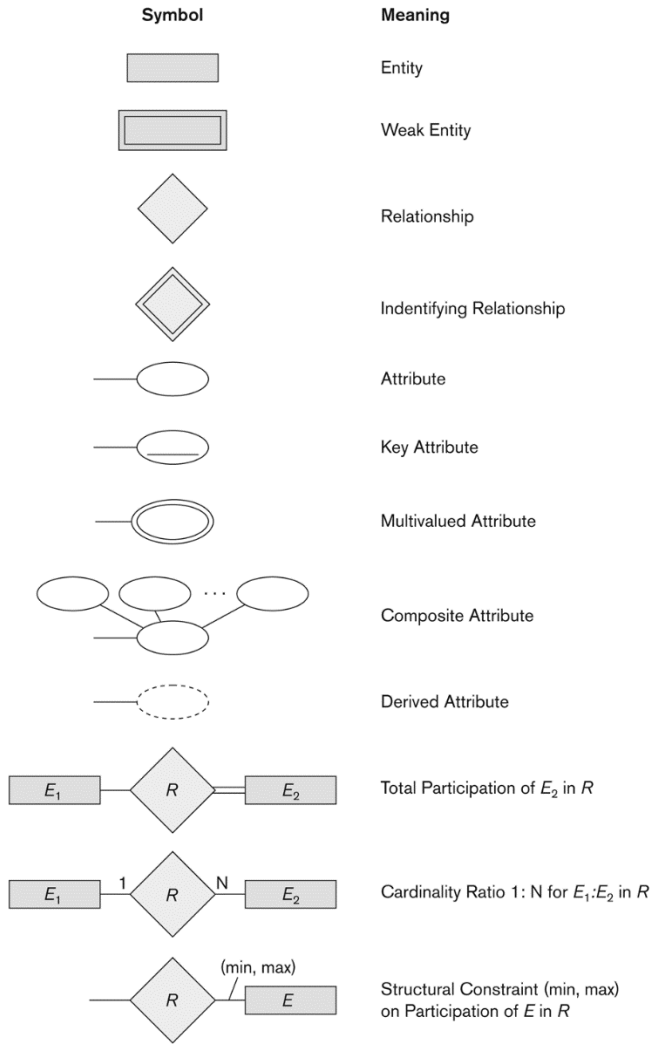
- Value sets are similar to data types in most programming languages (e.g., integer, character, int)
- Each simple attribute is associated with a value set (max,min)
  - For example, Person's Lastname has a value which is a character string up to 100 characters. Or if we have constraint about employee age should be between 16 and 70, we can directly specify it in database.
  - Values sets are not displayed in ER diagrams

Displaying an Entity type

- In ER diagrams, an entity type is displayed in a rectangular box
  - Attributes are displayed in ovals

- Each attribute is connected to its entity type
  - Components of a composite attribute are connected to the oval representing the composite attribute
  - Each key attribute is underlined
  - Multivalued attributes displayed in double ovals
- See the full ER notation in advance on the next slide

NOTATION for ER diagrams



Example of Entity Type CAR with two keys and a corresponding Entity Set

(a)

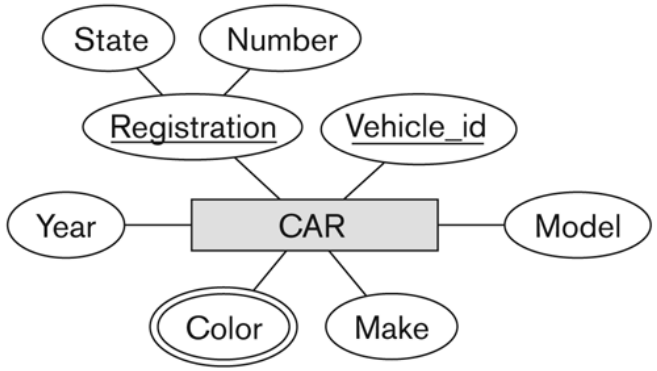


Figure 3.7
The CAR entity type with two key attributes, Registration and Vehicle_id. (a) ER diagram notation. (b) Entity set with three entities.

(b)

CAR
Registration (Number, State), Vehicle_id, Make, Model, Year, {Color}

CAR_1
((ABC 123, TEXAS), TK629, Ford Mustang, convertible, 2004 {red, black})

CAR_2
((ABC 123, NEW YORK), WP9872, Nissan Maxima, 4-door, 2005, {blue})

CAR_3
((VSY 720, TEXAS), TD729, Chrysler LeBaron, 4-door, 2002, {white, blue})

Example COMPANY data

- We need to create a database schema design based on the following (simplified) **requirements** of the COMPANY Database:
  - 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.
  - Each department *controls* a number of PROJECTs. Each project has a unique name, unique number and is located at a single location.
  - The database will store each EMPLOYEE's social security number, address, salary, sex, and birthdate.
    - Each employee *works for* one department but may *work on* several projects.
    - The DB will keep track of the number of hours per week that an employee currently works on each project.
    - It is required to keep track of the *direct supervisor* of each employee.
  - Each employee may *have* a number of DEPENDENTs.
    - For each dependent, the DB keeps a record of name, sex, birthdate, and relationship to the employee.

Conceptual Design for Company Database Schema

- Based on the requirements, we can identify **four** initial entity types in the COMPANY database:
  - DEPARTMENT (Name, Number, Locations, Manager)
  - PROJECT (Name, Number, Location)
  - EMPLOYEE (…)
  - DEPENDENT (Relatinoship,Gender,Birth_date,…)
- This is initial design which is not complete. Some of aspects in the requirements will be presented as ***relationships*** (not as attributes)
- ER has three main concepts:
  - Entities (types and sets)
  - Attributes (types and sets)
  - Relationships (types and sets)

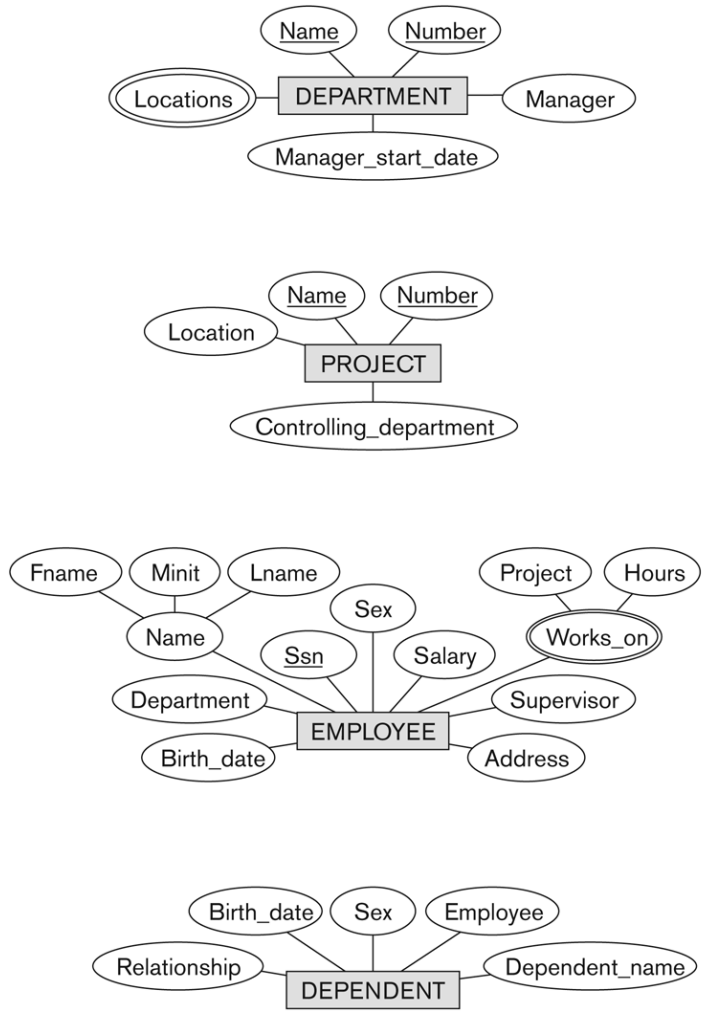


Figure 3.8
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

Relationships

WORKS_FOR N:1 relationship between EMPLOYEE and DEPARTMENT

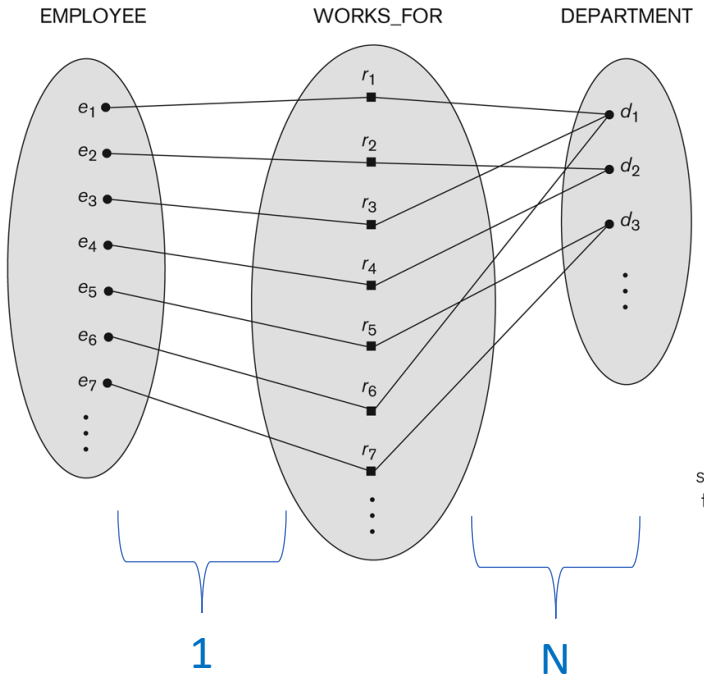


Figure 3.9
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

the N:M WORKS_ON relationship between EMPLOYEE and PROJECT

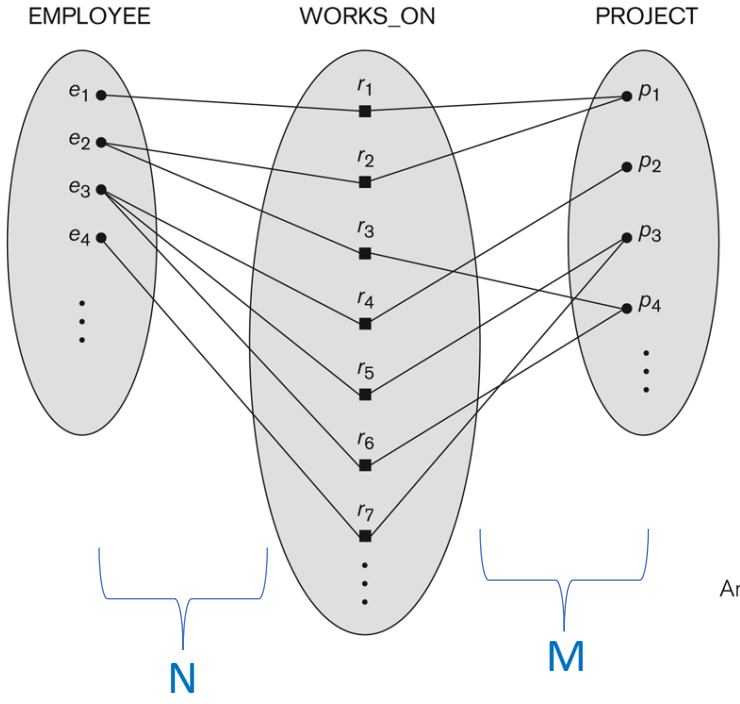
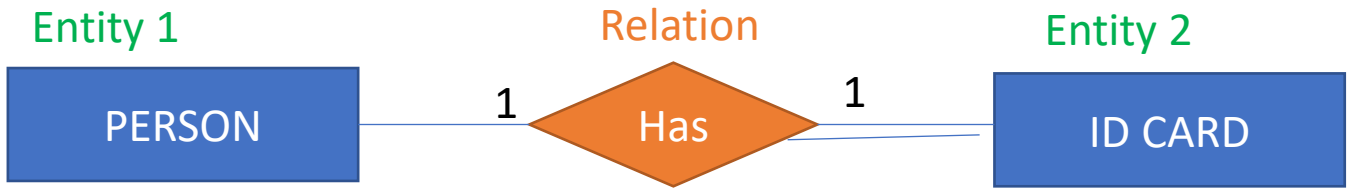


Figure 3.13
An M:N relationship, WORKS_ON.

Relationship Explained

Relationship Type	Examples
One-to-One (1:1) relationship is when each entity of one entity type is related to **only** one entity of the other entity type	 Reading the relationships from left to right: one person can have 0 or 1 ID CARD (partial participation) Reading the relationship from right to the left: one ID CARD can belong to only 1 person (total participation).
One-to-Many (1:N) relationship exists when each entity of one entity type can be related to one or more than one entity of the other entity type	 Relationships: One person can have 0 or more credit cards (partial participation (single line)) One credit card belongs only to one person (total participation (double line))
Many-to-Many (N:M) relationship exists when each entity of the one entity type can be related one or more the other entity of other entity type and vice-versa	 Relationships: An article may have 1 or more authors (total participation (double line) An author may write 0 or more articles (partial participation (single line)

Relationships described in COMPANY database

- By examining the requirements, six relationship types are identified
- All are ***binary*** relationships (degree 2, the relationship only between two entities)
- Listed below with their participating entity types:
  - WORKS_FOR (between EMPLOYEE, DEPARTMENT)
  - MANAGES (also between EMPLOYEE, DEPARTMENT)
  - CONTROLS (between DEPARTMENT, PROJECT)
  - WORKS_ON (between EMPLOYEE, PROJECT)
  - SUPERVISION (between EMPLOYEE (as subordinate), EMPLOYEE (as supervisor))
  - DEPENDENTS_OF (between EMPLOYEE, DEPENDENT)

COMPANY ER Diagram

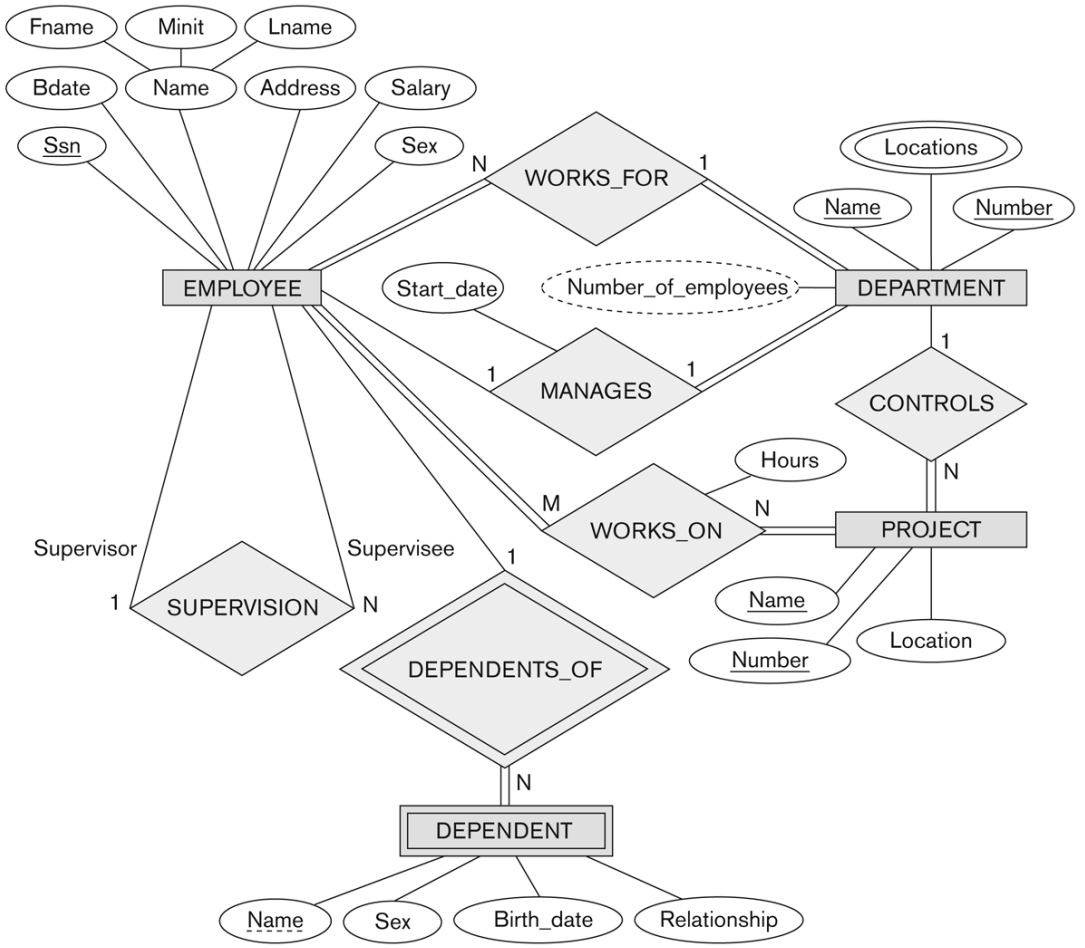


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

Discussion on Relationship Types

- In the refined design, some attributes from the initial entity types are refined into relationships:
  - Manager of DEPARTMENT -> MANAGES
  - Works_on of EMPLOYEE -> WORKS_ON
  - Department of EMPLOYEE -> WORKS_FOR
  - Etc.
- In general, more than one relationship type can exist between the same participating entity types
  - MANAGES and WORKS_FOR are distinct relationship types between EMPLOYEE and DEPARTMENT
  - Different meanings and different relationship instances.

Constraints on Relationships

- Constraints on Relationship Types
  - (Also known as ratio constraints)
  - Cardinality Ratio (specifies *maximum* participation)
    - One-to-one (1:1)
    - One-to-many (1:N) or Many-to-one (N:1)
    - Many-to-many (M:N)
  - Existence Dependency Constraint (specifies *minimum* participation) (also called participation constraint)
    - zero (optional participation, not existence-dependent)
    - one or more (mandatory participation, existence-dependent)

Recursive Relationship Type

- A relationship type between the same participating entity type in **distinct roles**
- Also called a **self-referencing** relationship type.
- Example: the SUPERVISION relationship
- EMPLOYEE participates twice in two distinct roles:
  - supervisor (or boss) role
  - supervisee (or subordinate) role
- Each relationship instance relates two distinct EMPLOYEE entities:
  - One employee in *supervisor* role
  - One employee in *supervisee* role

Displaying a recursive relationship

- In a recursive relationship type.
  - 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.

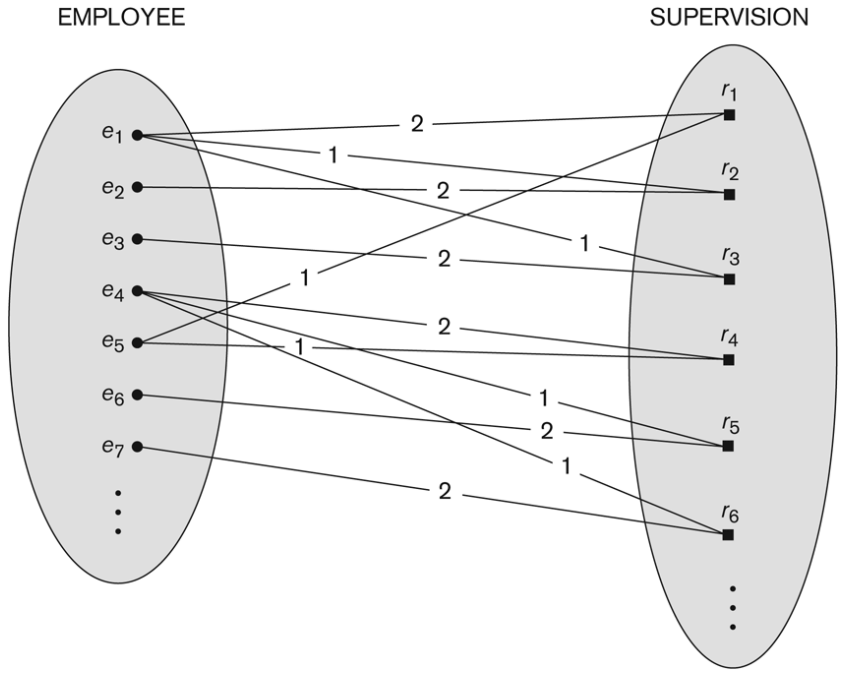


Figure 3.11
A recursive relationship SUPERVISION between EMPLOYEE in the *supervisor* role (1) and EMPLOYEE in the *subordinate* role (2).

Recursive Relationship Type is: SUPERVISION (participation role names are shown)

Recursive Relationship

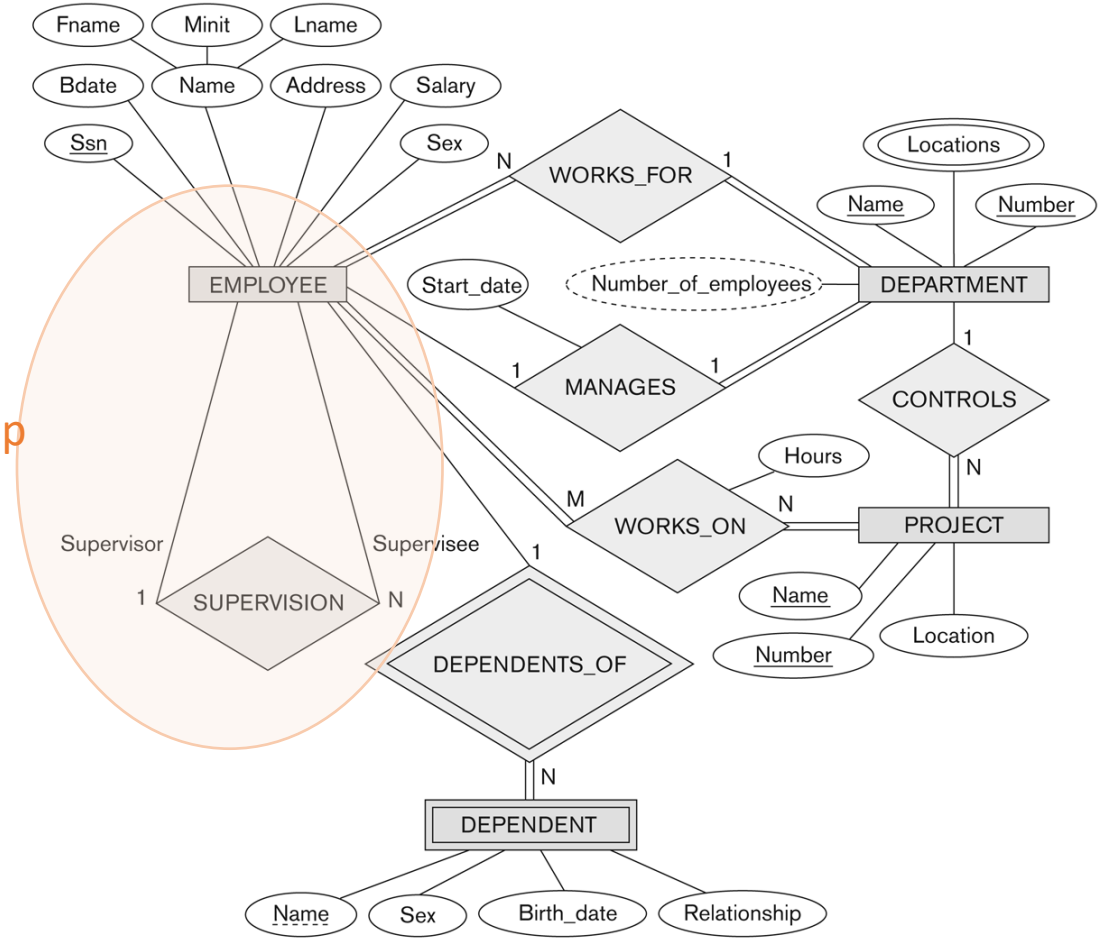


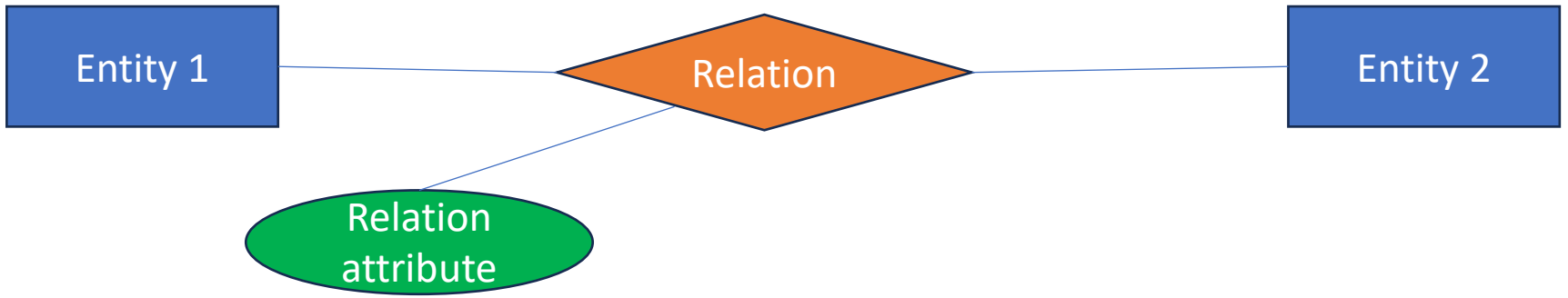
Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

Weak Entity Types

- A **weak entity** has a partial key attribute and that is identification-dependent on another entity type.
- A weak entity must participate in an identifying relationship type with an *owner* of identifying entity type
- Entities are identified by the combination of:
  - A partial key of the weak entity type and
  - The particular entity's (owner) key they are related to
- **Example:**
  - A DEPENDENT entity is identified by the dependent's first name, *and* the specific EMPLOYEE with whom the dependent is related
  - Name of DEPENDENT is the *partial key*
  - DEPENDENT is a *weak entity type*
  - EMPLOYEE is its identifying entity type (owner) via the identifying relationship type DEPENDENT_OF

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.
    - A value of HoursPerWeek depends on a particular (employee, project) combination
  - Most relationship attributes are used with M:N relationships
    - In 1:N relationships, they can be transferred to the entity type on the N-side of the relationship



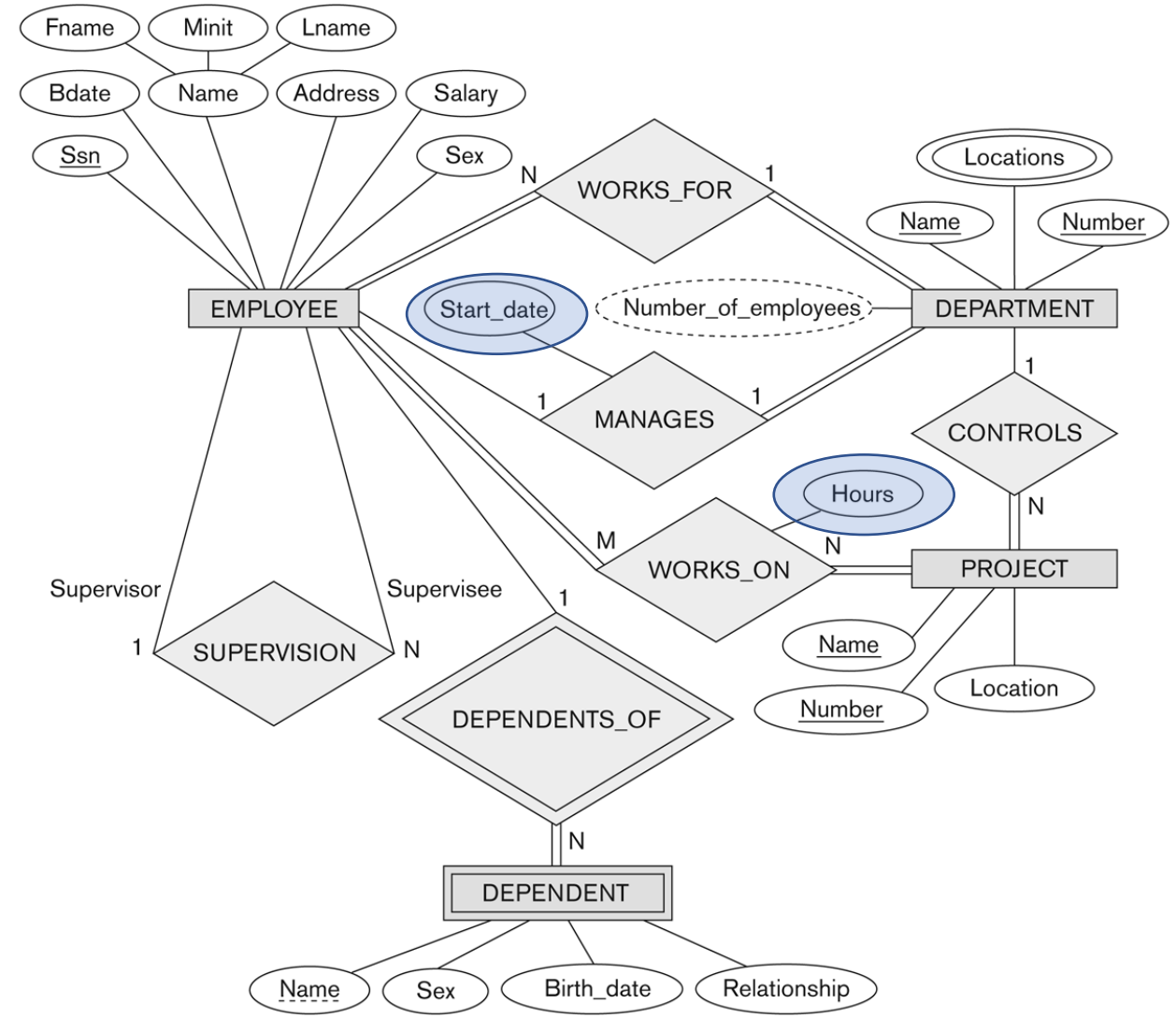


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

Relationship Attributes

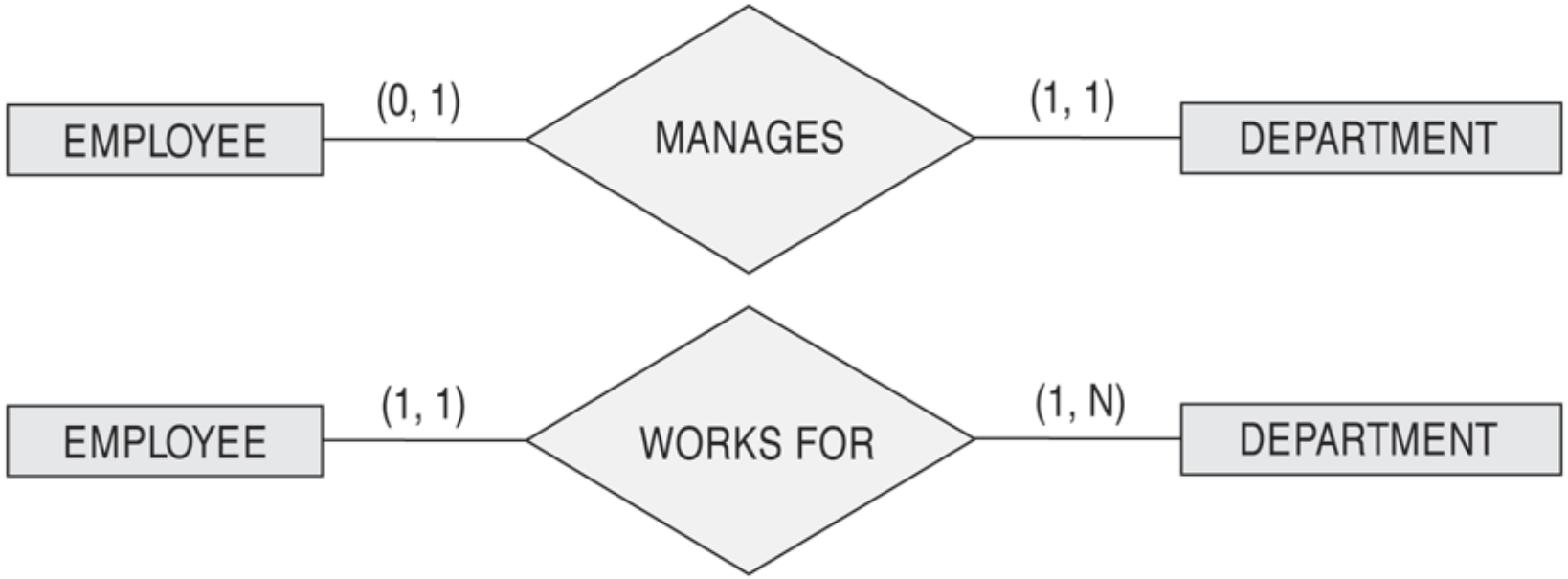
Notation for Constraints on Relationships

- Cardinality ratio (of a binary relationship): 1:1, 1:N, N:1, or M:N
  - Shown by placing appropriate numbers on the relationship edges.
- Participation constraint (on each participating entity type): total (called existence dependency) or partial.
  - Total shown by double line, partial by single line.
- NOTE: These are easy to specify for Binary Relationship Types.

Alternative (min, max) notation for relationship structural constraints:

- Specified on each participation of an entity type E in a relationship type R
- Specifies that each entity e in E participates in at least *min* and at most *max* relationship instances in R
- Default(no constraint): min=0, max=n (signifying no limit)
- Must have min≤max, min≥0, max ≥1
- Derived from the knowledge of mini-world constraints
- Examples:
  - A department has exactly one manager and an employee can manage at most one department.
    - Specify (0,1) for participation of EMPLOYEE in MANAGES
    - Specify (1,1) for participation of DEPARTMENT in MANAGES
  - An employee can work for exactly one department but a department can have any number of employees.
    - Specify (1,1) for participation of EMPLOYEE in WORKS_FOR
    - Specify (0,n) for participation of DEPARTMENT in WORKS_FOR

Example: The (min,max) notation for relationship constraints



Read the min,max numbers next to the entity type and looking **away from** the entity type

COMPANY ER Schema Diagram using (min, max) notation

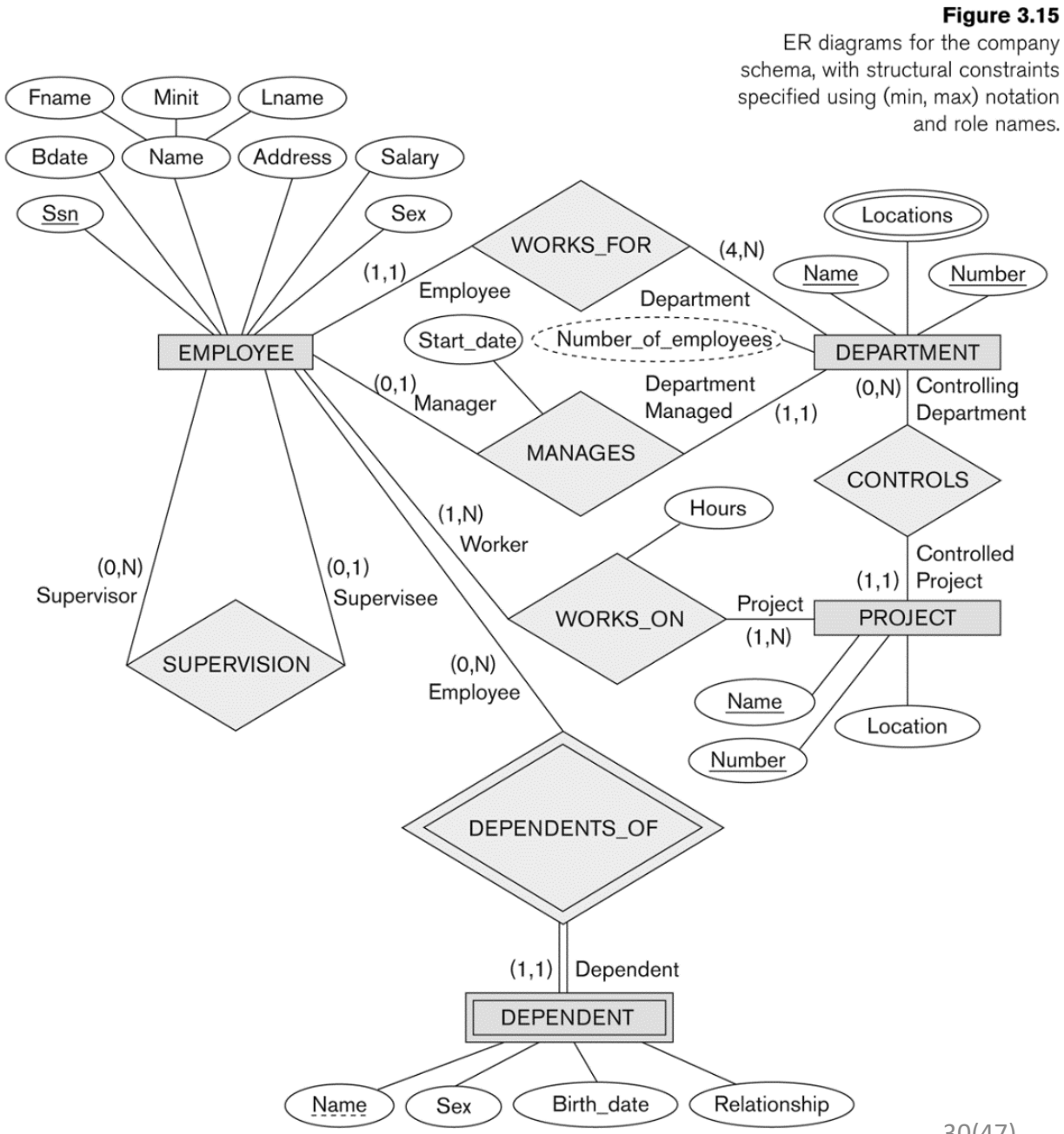


Figure 3.15
ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.

Break 10 min

Relationships has a 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
- Constraints are harder to specify for higher-degree relationships (n > 2) than for binary relationships

Example of a ternary relationship

Challenges with ternary relationships:

- Complexity
- Normalization Challenges
- Maintenance Difficulty
- Data Redundancy

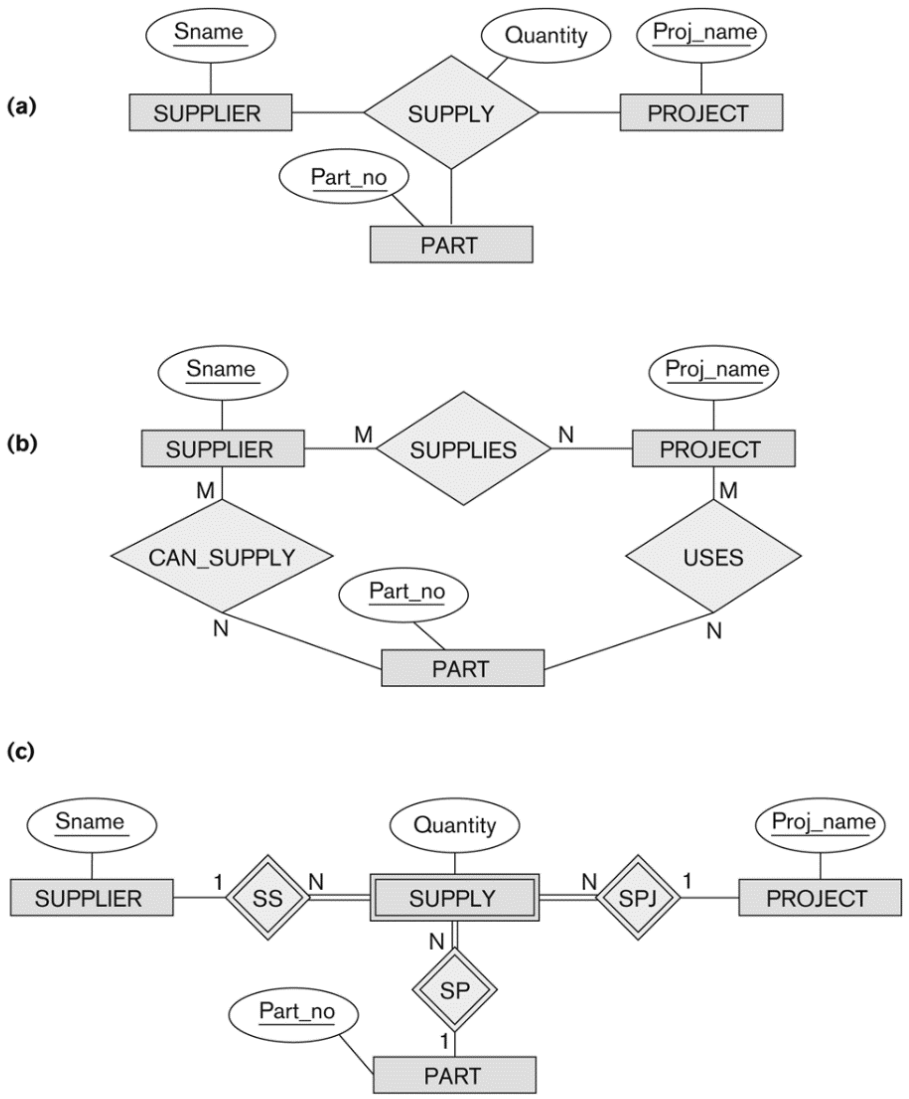


Figure 3.17
Ternary relationship types. (a) The SUPPLY relationship. (b) Three binary relationships not equivalent to SUPPLY. (c) SUPPLY represented as a weak entity type.

Data Modeling Tools

- A number of popular tools that cover conceptual modeling and mapping into relational schema design.
  - Examples: ERWin, S- Designer (Enterprise Application Suite), ER- Studio, etc.
- Advantages:
  - Serves as documentation of application requirements, easy user interface - mostly graphics editor support
- Disadvantages:
  - Most tools lack a proper distinct notation for relationships with relationship attributes
  - Mostly represent a relational design in a diagrammatic form rather than a conceptual ER-based design

The recommended modeling tool is: https://app.diagrams.net/

Lecture 2 Concepts to Understand for Exam

- **Entity**: a thing or object in the real world with an independent existence that can be differentiated from other objects
- **Entity set/state**: a collection of entities of an entity type at a point in time
- **Entity type**: a collection of similar entities
- **Types of attributes**: simple, single-valued/multi-valued, stored/derived, null, composite
- **Key attribute**: a single (or composite) attribute whose values can be used to uniquely identify an individual entity in an entity set
- Difference between **attribute** and **value set**: an attribute is a particular property that describes an entity. A value set specifies the set of values that may be assigned to that attribute for each individual entry (e.g., age between 16 and 70 only).
- Difference between **entity type** and **entity set**: entity type describes data structure, entity set contains instances (real data) of a given structure
- **Relationships**: the association or interactions between entities
- **Types of relationships**: one-to-one, one-to-many, and many-to-many (1:N, N:1, N:M).
- **Difference between binary and n-ary relationships**: binary relationship is between two different entities. In n-ary relationships type, there is the relationship between n number of different entity types.
- **Recursive relationship type**: a relationship exists between occurrences of the same entity set
- **Partial participation:** when all the entities of an entity type are not associated with one or the other entity of another entity type
- **Total Participation**: when all entities of an entity type are associated with one or the other entity of another entity type;
- **Weak entity**: an entity that has no primary key attribute to uniquely identify the records existing in it. Therefore, it has to be dependent on the strong entity set for its unique identification

Exercise

Entity 1	Participation	Cardinality Ration	Participation	Entity 2
Student				Course
Course				Instructor
University				Department
Department				Head of Department
User				Post (Facebook)
Post (Facebook)				Comment
User				User Profile
Customer				Order
Product				Order
Book				BookDetails