Entity-Relationship Model

CS 377: Database Systems

Course Announcements

- First homework is out
  - Hand in a hard-copy of the homework
  - Due Wed Feb 3rd IN CLASS
- My office hours on Thursday (Jan 28th) are canceled this week
- My office hours on Tuesday (Feb 2nd) are extended by an hour from 1 - 5 pm

Entity-Relationship (ER) Model

- Specification/design language
  - Information the DB must hold
  - Relationships amongst the components of that information
- Proposed by Peter Chen in 1976
- Still very popular with many styles/notations

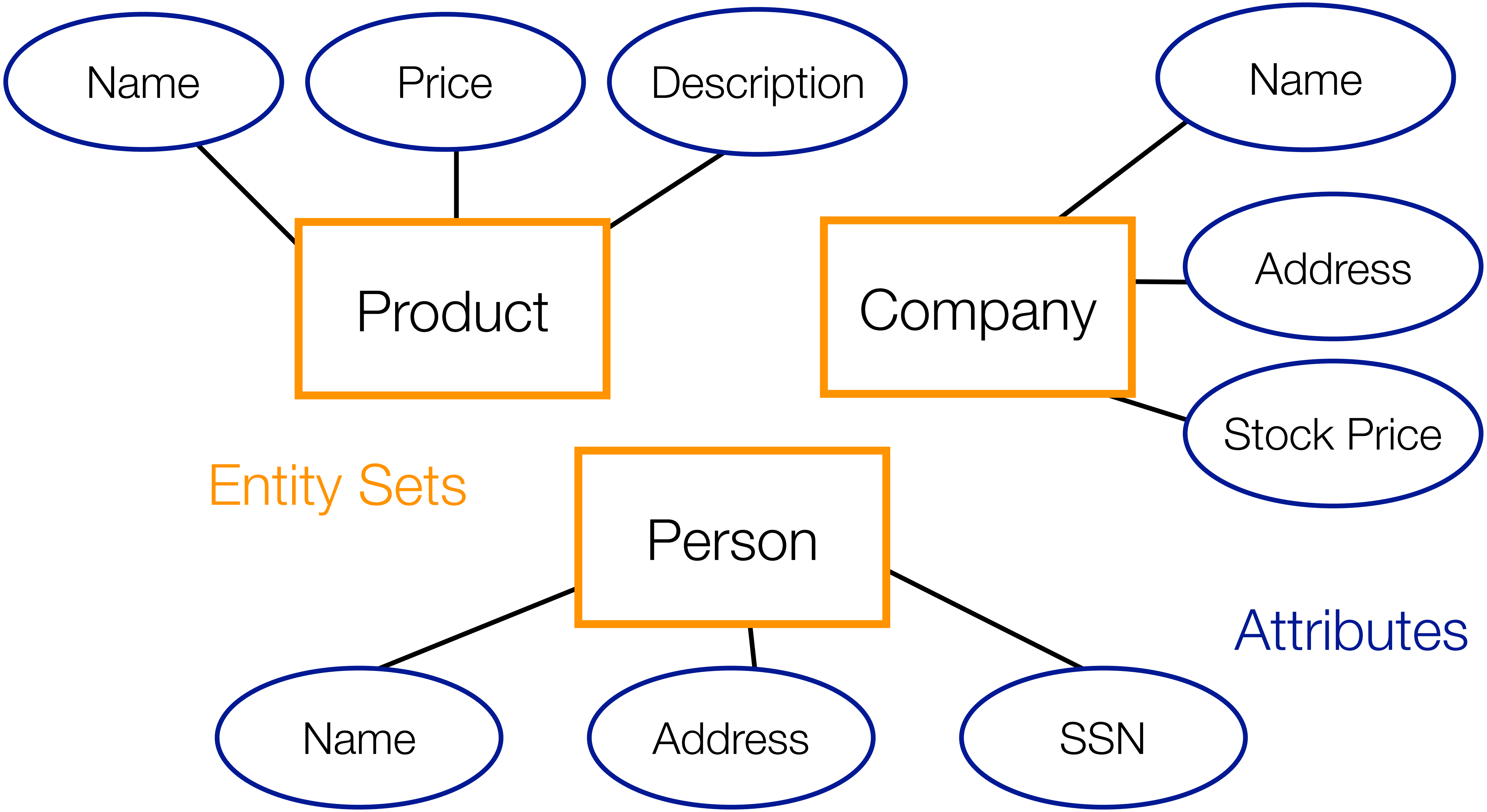
ER Basics

- **Entity**: thing or object
- **Attribute**: properties used to describe an entity
  - Each attribute has a value set (data type) associated with it (e.g., integer, string, …)
  - A specific entity will have a value for each of its attributes
- **Entity set**: a collection of similar entities

Example: Company database

- Each company has a name and address
- Each company has a list of employees
- List of products manufactured by the companies
- Each product has a name and description

Example: Entities & Attributes



Attributes

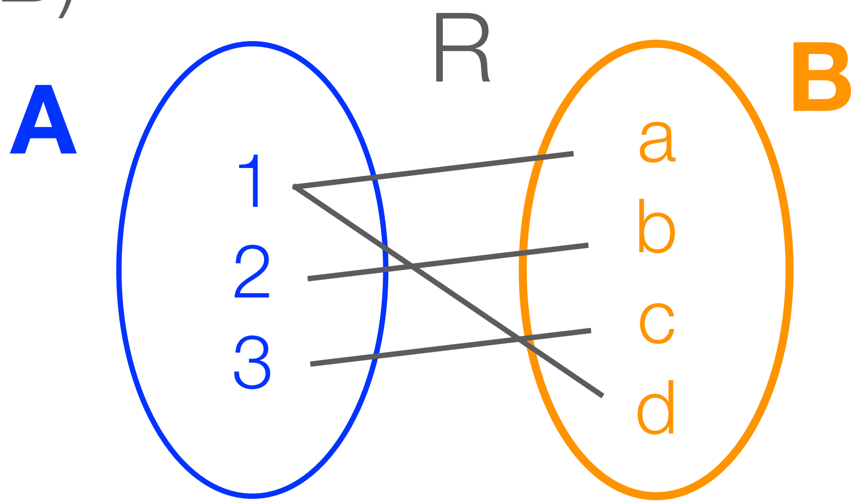
- **Simple**: attribute only takes on atomic values (e.g., age, salary, SSN)
- **Composite/Compound**: attribute has a structure and may be composed of several components (e.g., address, name)
- **Multi-valued**: multiple values for an attribute (e.g., previous degrees of a student)
- **Complex**: composite or multi-valued attributes nested to any number of levels (e.g., previous degrees of a student with {college, year, degree, and field})

Special Attributes

- **Derived attributes**: values that can be computed or derived from other attributes (e.g., age can be derived from birth date)
  - Should not store a derived attribute as it introduces redundancy
- **Key attributes**: a set of attributes for which *no two different entities* will have the same values (e.g., SSN for people, VIN for cars)
  - Can be used to identify the entity uniquely
- **NULL** value: can mean not available or not applicable
  - Equality comparison of two attribute values both equal to NULL should return FALSE

Relation

- Mathematical definition:
  - If **A, B** are sets, then a relation **R** is a subset of **A** x **B** (cartesian product of the sets **A** and **B**)



- Example:
  - **A** = {1,2,3}, **B** = {a, b, c, d}
  - **A** x **B** = all pairs of tuples {(1,a), (1,b), (1,c) (1,d), (2,a), (2,b), (2,c), (2,d), (3,a), (3,b), (3,c), (3,d)}
  - **R** = {(1, a), (1,d), (2,b), 3(c)}

Relationships and Relationship Types

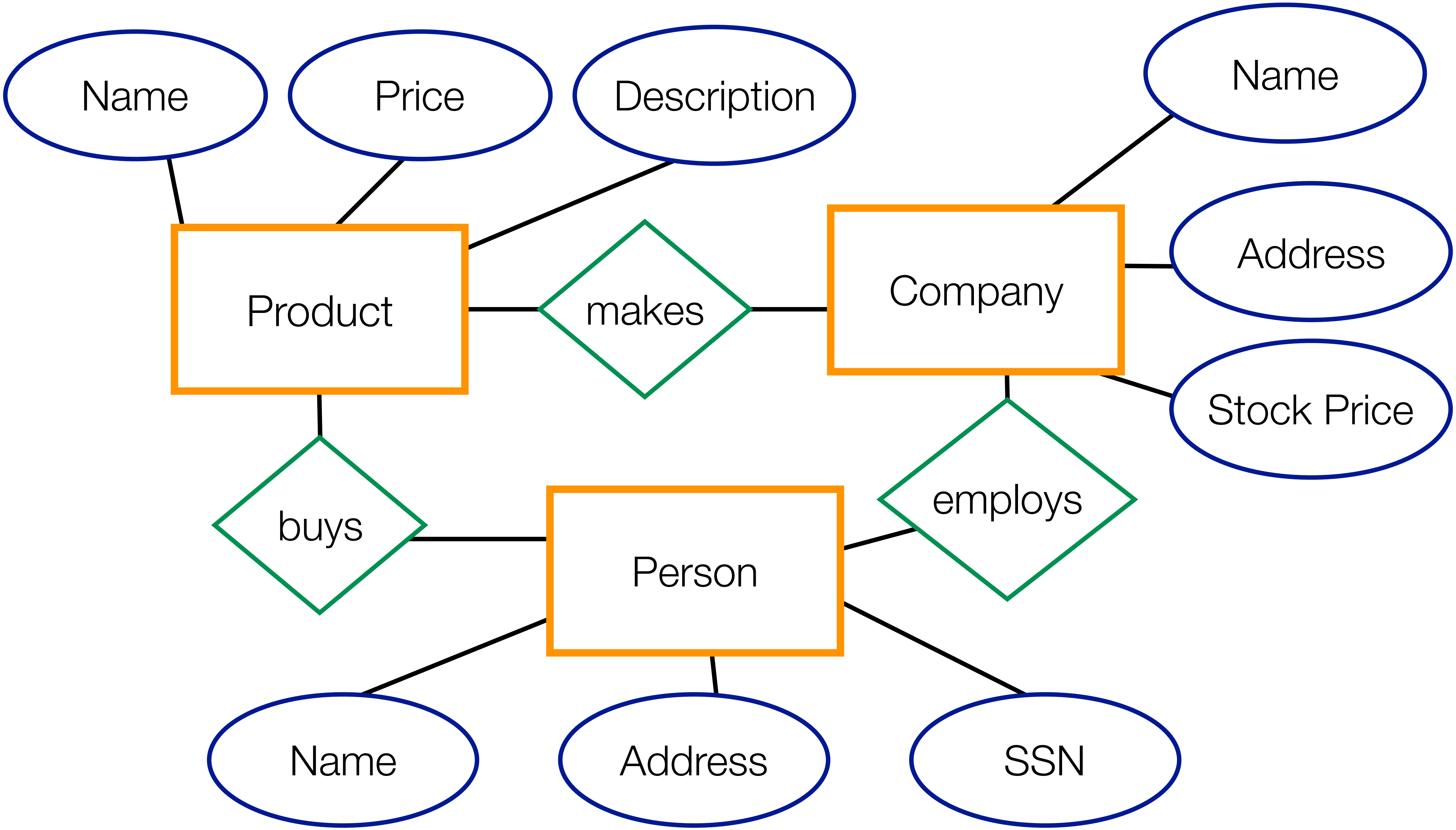
- **Relationship** relates two or more distinct entities with a specific meaning or an association amongst entities (e.g., Coca-Cola company makes Sprite)

- Relationships of the same type are grouped or typed together into a **relationship type** (e.g., company MAKES products)

  - Relationship type **R** = any subset of the cartesian product among entity types **E1, E2, …, $\mathbf{E_n}$**

- More than one relationship type can existing between two participating entity types

Example: Relationships



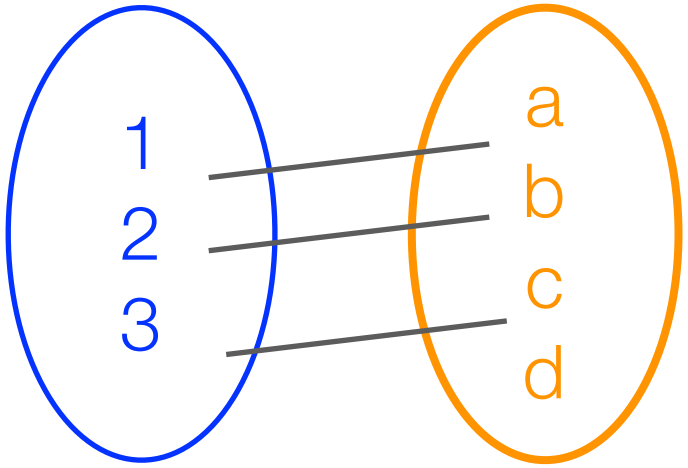
Relationship Degree

- **Degree** of the relationship is the number of participating entity types
- Most common type of relationship is **binary** involving 2 entity types
  (e.g., Coca-Cola Company makes Sprite)
- Less common are **ternary** relationship with 3 entity types
  (e.g., PERSON purchases PRODUCT from STORE)
- Relationship types of degree n are called **n-ary**
  - n-ary relationships can be converted to n binary relationships

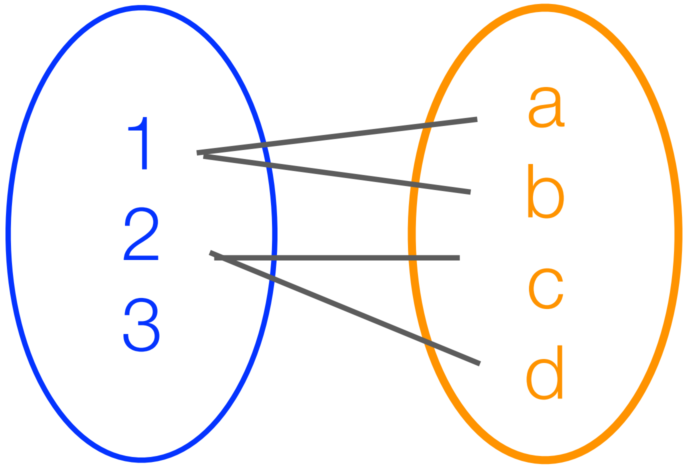
Constraints on Relationship Types

Cardinality ratio constraints: maximum number of relationship instances that an entity can participate in a binary relationship

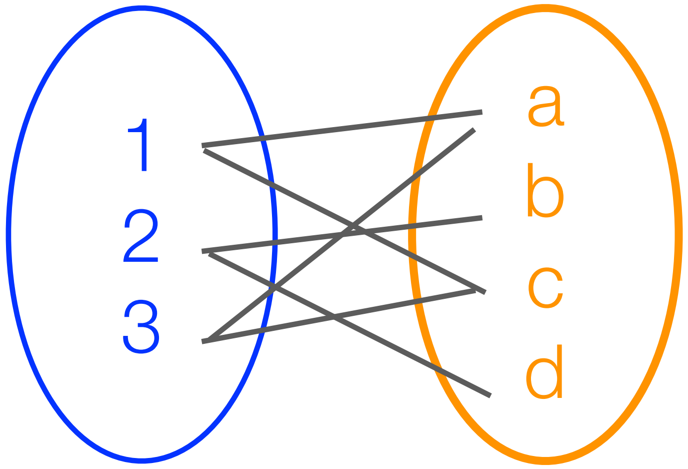
- One-to-one (1:1)



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



- Many-to-many (N:N)



Constraints on Relationship Types

Participation constraint or **existence dependency constraints**: whether the participation of an entity in a relationship is compulsory or not

- Zero: partial participation, optional participation, not existence-dependent
  (e.g., COMPANY may not produce any PRODUCT)

- One or more: total participation, mandatory, existence-dependent
  (e.g., PRODUCT must be produced by a COMPANY)

Relationship Properties

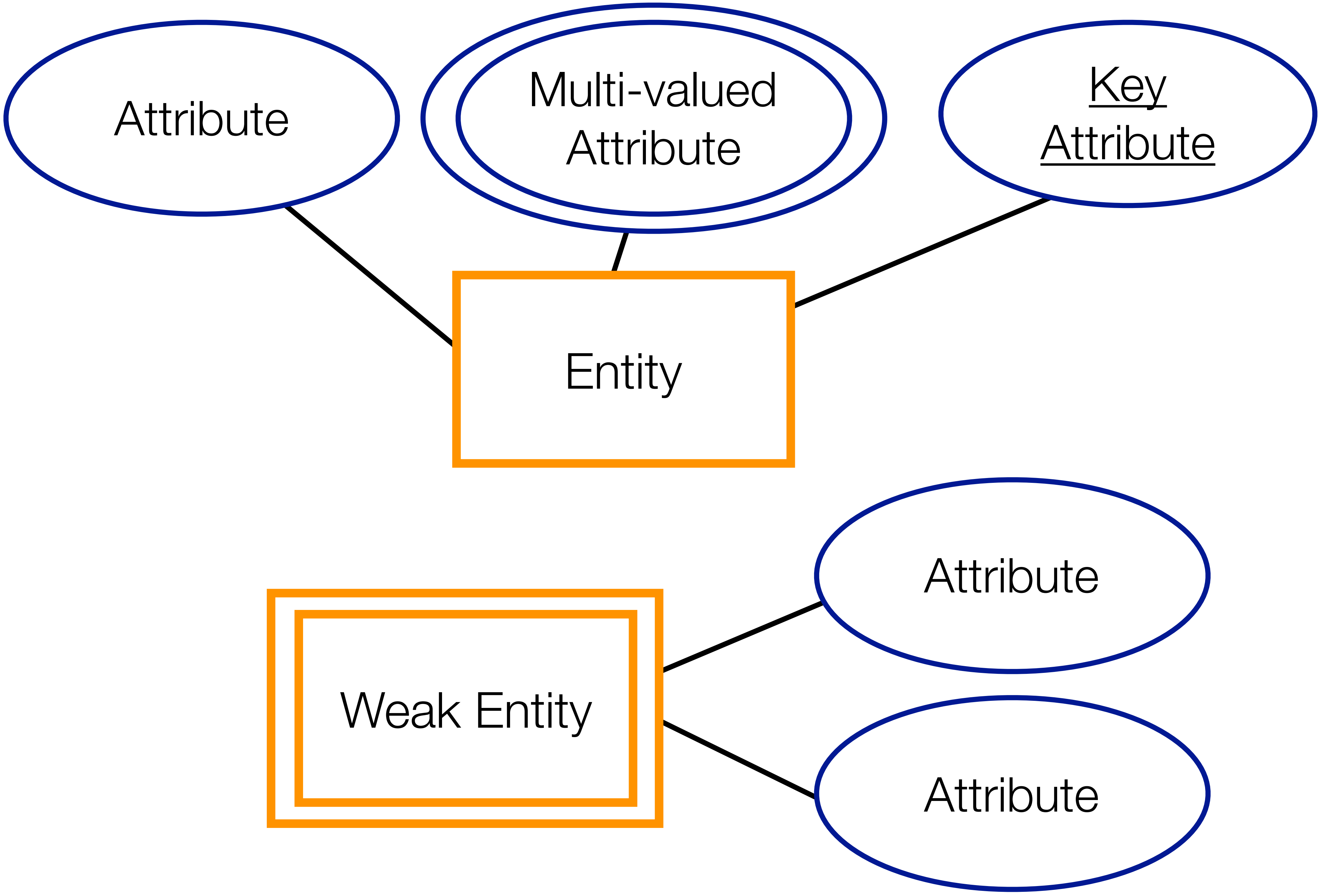
- Relationships can be **recursive** with both participants having same entity type in different roles
(e.g., DEAN is a PROFESSOR that SUPERVISES another PROFESSOR)

- Relationship type can have attributes
(e.g., DATE is an ATTRIBUTE for a PERSON purchasing a PRODUCT)

Weak Entity Types

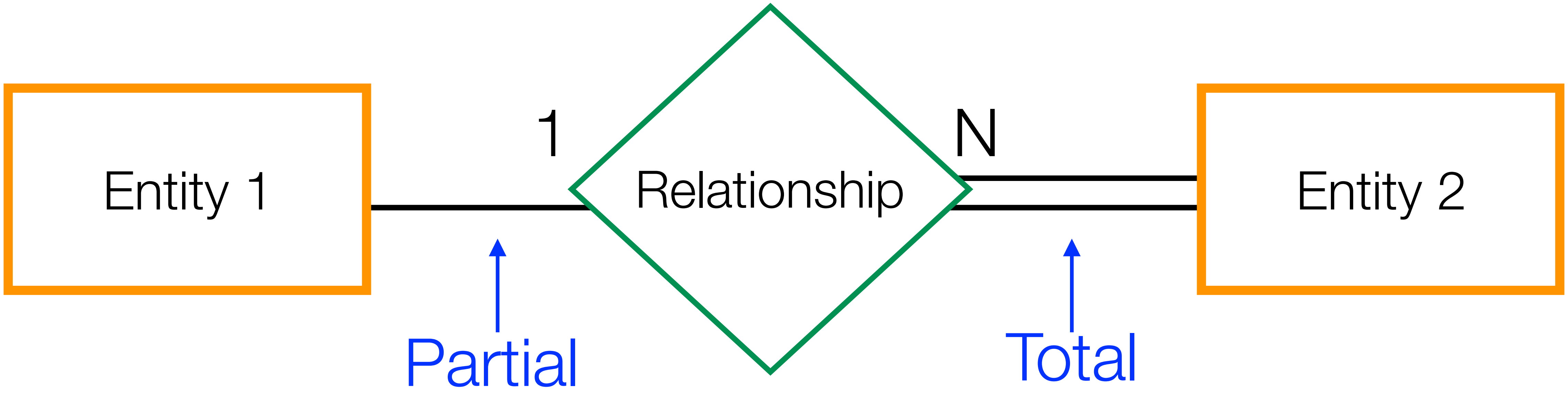
- Entity that does not have a key attribute and participates in an identifying relationship with an owner or identifying entity type
- Identified by a combination of:
  - Partial key of the weak entity type
  - Particular entity they are related to in the identifying entity type

ER Diagram Basics: Entities



ER Diagram Basics: Relationships

Entity 1: Entity 2 = 1 : N



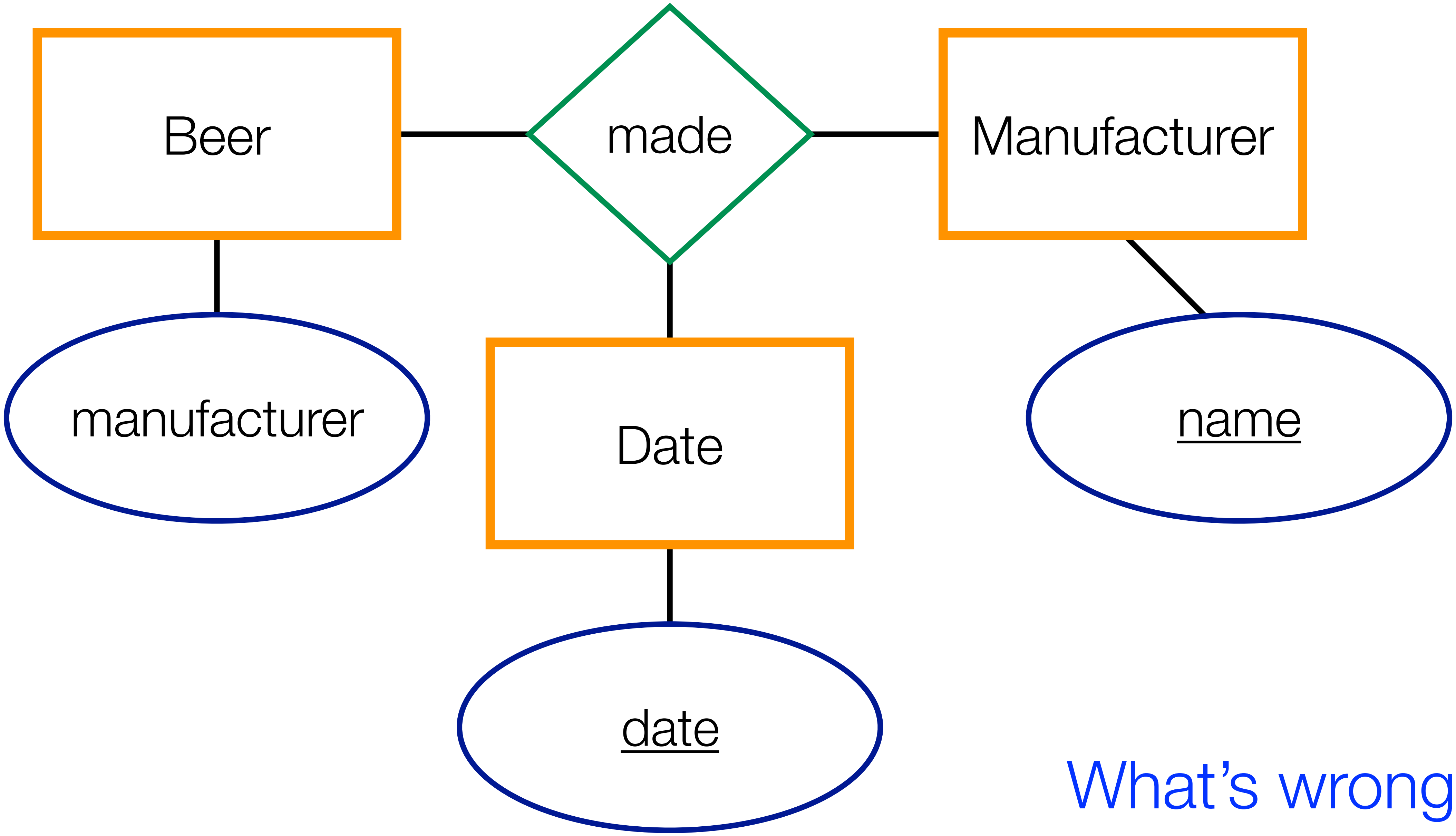
Steps for ER Design

- Gather information or requirements
- Identify the entities - which things are important enough to be identified with a key
- Identify the properties/attributes of the entities
- Determine the relationships (usually properties that occur between 2 or more entities)
    - Cardinality ratio constraints on binary relationships
    - Participation constraints

General Design Principles

- Avoid redundancy: wastes storage space and encourages inconsistency
- Keep it simple
- Attributes over entities: entities should have at least one non-key attribute
- Don’t overuse weak entity sets: in practice, you can create unique IDs for entity sets

Example of a Bad ER Model



What’s wrong?

Example: Company Database (from book)

- Company is organized into departments
- Each department has a unique name, a unique number, and is managed by one employee
- Company keeps track of the start date when that employee began managing the department (e.g., for bonus reward purposes)
- A department may have several locations (e.g., Atlanta, Boston, LA)
- A department controls a number of projects, each of which has a unique name, a unique number, and a single location where the project is performed
- Each employee has a name, social security number (SSN), address, salary, sex, and birth date.

Example: Company Database (2)

- An employee works for one department but may work on several projects, which are not necessarily controlled by the same department (that the employee is assigned to).
- Company tracks the number of hours per week an employee works on each of his/her projects
- Each employee has one direct supervisor (also an employee of the company)
- Information about the dependents of the employee (for benefit calculation purposes) is painted but is less detailed than those for employees
- Each dependent has a first name, sex, birth date, and the relationship to the employee

Example: Identify Company Entities

- Each department has a unique name, a unique number, and is managed by one employee

- A department controls a number of projects, each of which has a unique name, a unique number, and a single location where the project is performed

- Each employee has a name, social security number (SSN), address, salary, sex, and birth date.

- Each dependent has a first name, sex, birth date, and the relationship to the employee

Example: Identify Entity Attributes

- Each department has a unique name, a unique number, and is managed by one employee
- Company keeps track of the start date when that employee began managing the department (e.g., for bonus reward purposes)
- A department controls a number of projects, each of which has a unique name, a unique number, and a single location where the project is performed

Department(DName, DNumber, Manager, ManStartDate, {Locations}, {Projects})

Example: Identify Entity Attributes (2)

- Each employee has a name, social security number (SSN), address, salary, sex, and birth date.

Project (PName, PNumber, Location)

Example: Identify Entity Attributes (3)

- Each department has a unique name, a unique number, and is managed by one employee
- An employee works for one department but may work on several projects, which are not necessarily controlled by the same department (that the employee is assigned to).
- Company tracks the number of hours per week an employee works on each of his/her projects
- Each employee has one direct supervisor (also an employee of the company)

Employee (SSN, Name, Addr, Salary, Sex, BDate, Dept, {Proj, hour}, Supervisor)

Example: Identify Entity Attributes (4)

- Information about the dependents of the employee (for benefit calculation purposes) is painted but is less detailed than those for employees

- Each dependent has a first name, sex, birth date, and the relationship to the employee

Dependent(FName, Sex, BDate, RelationToEmp)

Example: Determine Relationships

- Note that some attributes identified in previous step are not attributes but relationships as they reference other entities
- Manager(Employee, Department)
  - 1 employee can manage at most 1 departments
  - 1 department has 1 manager
  - A department must have a manager employee (total)
  - Employee need not manage any department (partial)

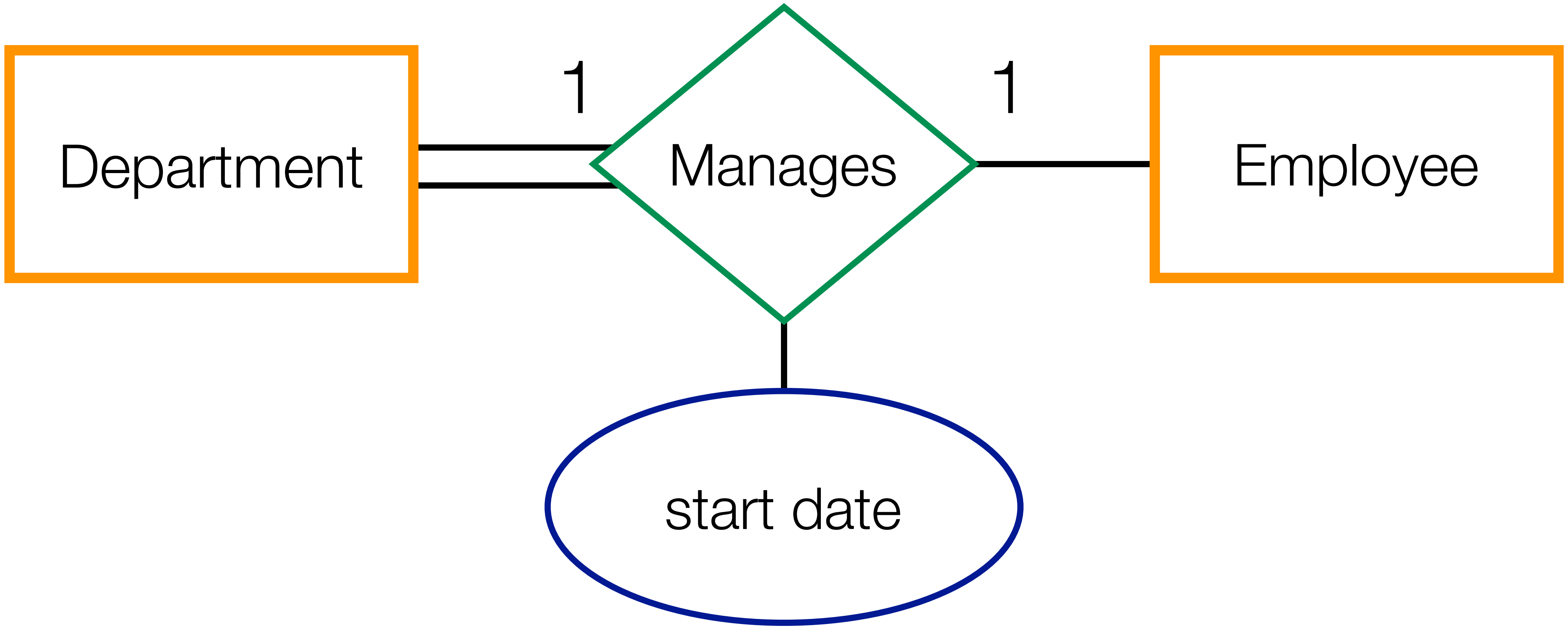
Example: Manager Relationship

Department(DName, DNumber, Manager, ManStartDate, {Locations}, {Projects})

Employee (SSN, Name, Addr, Salary, Sex, BDate, Dept, {Proj, hour}, Supervisor)

- Note that Manager is now converted to a relationship rather than an attribute
- Rather than having Manager start date be an attribute of department, consider it as an attribute about the relationship between the department and employee

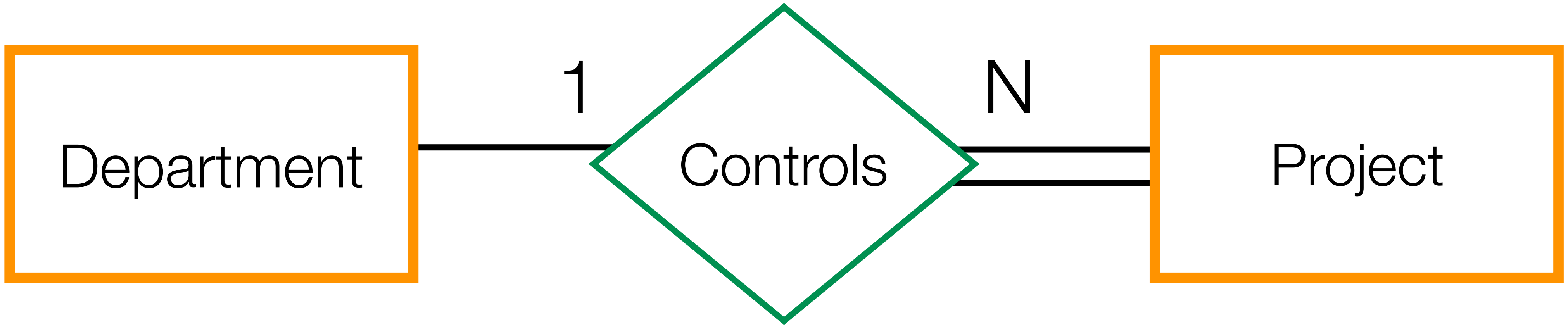
Example: Manager Relationship



Example: Determine Relationships (2)

- Controls(Department, Projects)
  - 1 department controls N projects
  - 1 project is controlled by 1 department
  - A project must have a controlling department (total)
  - A department need not manage any project (partial)

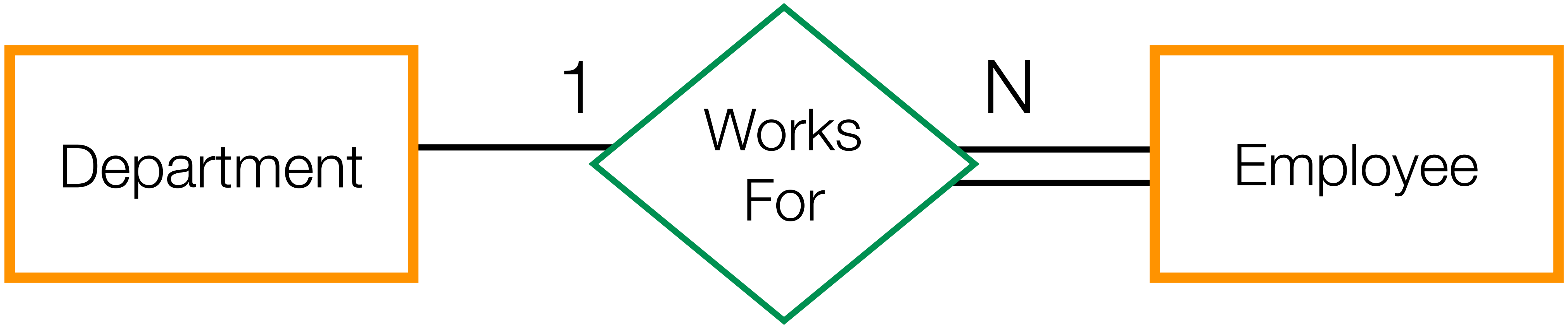
Example: Controls Relationship



Example: Determine Relationships (3)

- WorksFor(Employee, Department)
  - 1 employee works for 1 department
  - 1 department has N employees
  - An employee must work for a department (total)
  - A department need not have any employees (partial)

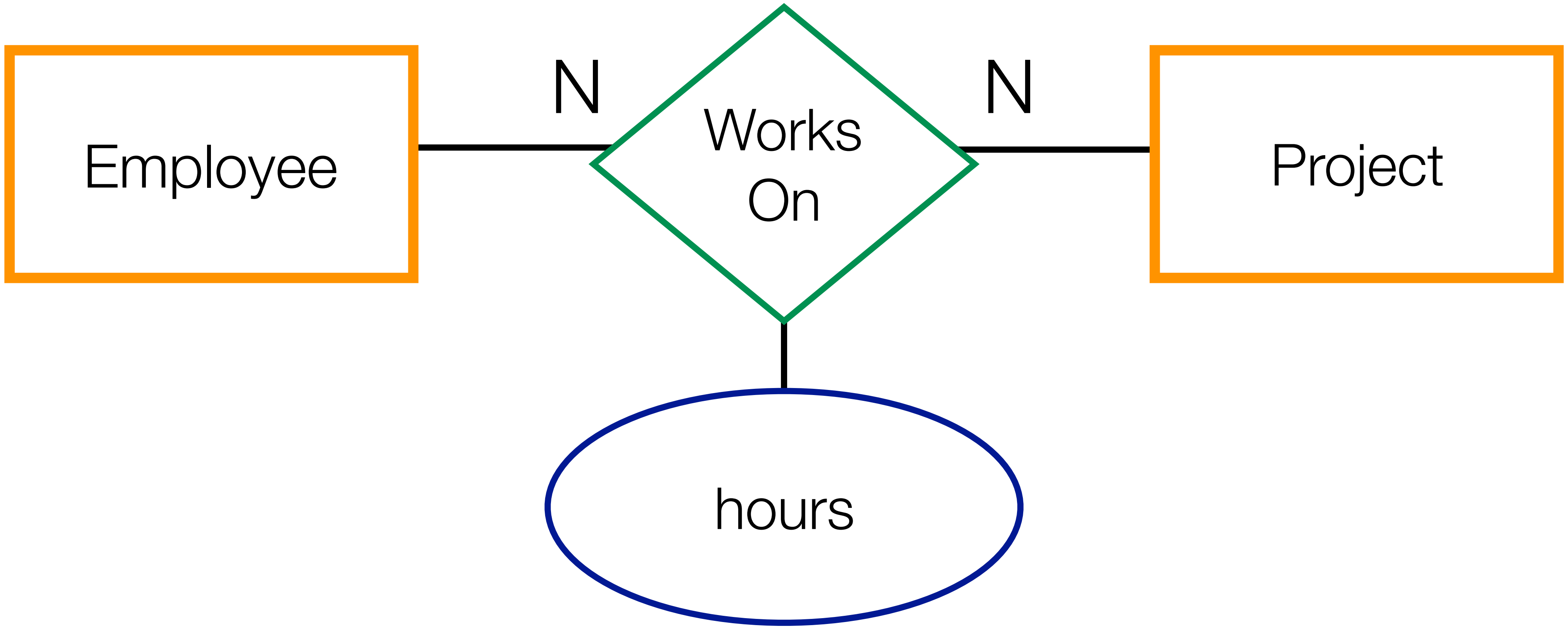
Example: WorksFor Relationship



Example: Determine Relationships (4)

- WorksOn(Employee, Project)
  - 1 employee works on N projects
  - 1 project is worked on by N employees
  - An employee need not work on any project (partial)
  - A project need not have any employees (partial)
- Note that hour attribute for employee provides information about relationship between an employee and a project

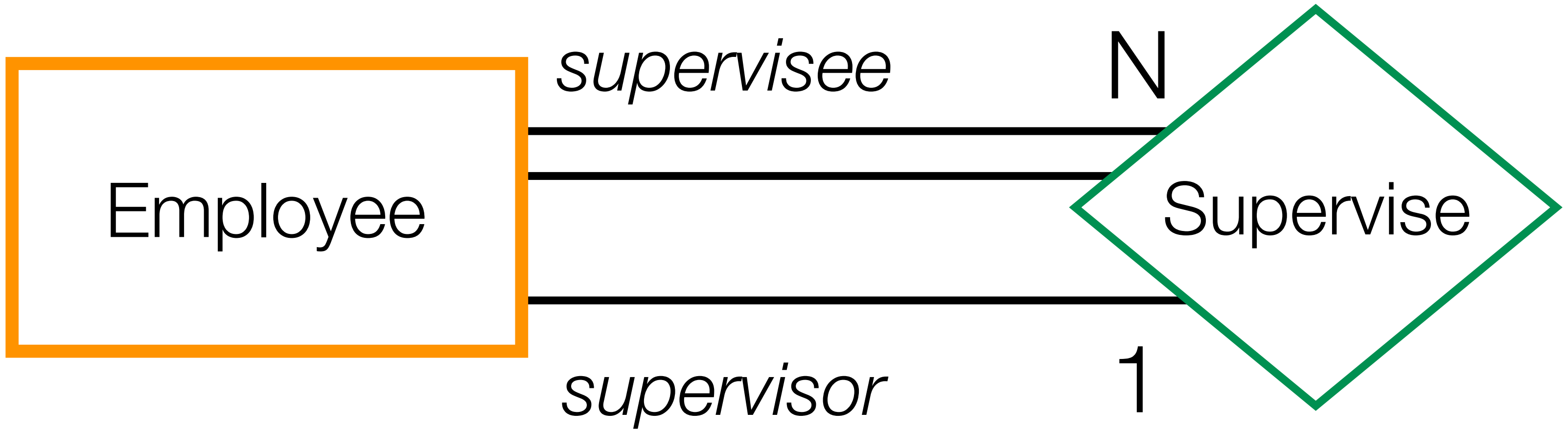
Example: WorksOn Relationship



Example: Determine Relationships (5)

- Supervisor(Employee, Employee)
    - To distinguish the two different entities, we assign two different roles: supervisor and supervisee
    - 1 supervisor employee supervises N employees
    - 1 supervisee employee has 1 supervisor employee
    - A employee need not manage any employee (partial)
    - A employee must have a supervisor (total)
- This is an example of a recursive relationship as it is a relationship between two entities from the same entity set

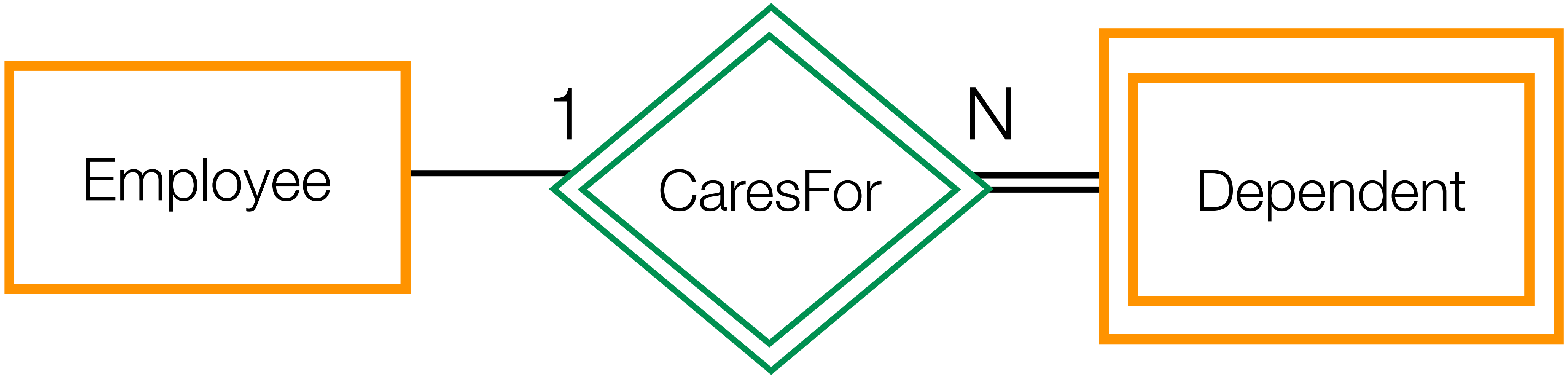
Example: Supervisor Relationship



Example: CareFor Relationships (6)

- CaresFor(Employee, Dependent)
  - 1 employee has N dependents
  - 1 dependent belongs to 1 employee
  - An employee need not have any dependents (partial)
  - A dependent must belong to an employee
- Since dependents can have all their attributes having the same value, then this must be a WEAK entity
- The relationship in which a weak entity obtains additional identifying information is called a WEAK relationship

Example: CaresFor Relationship



weak relationship type is represented by a double diamond

Example: Company ER Diagram

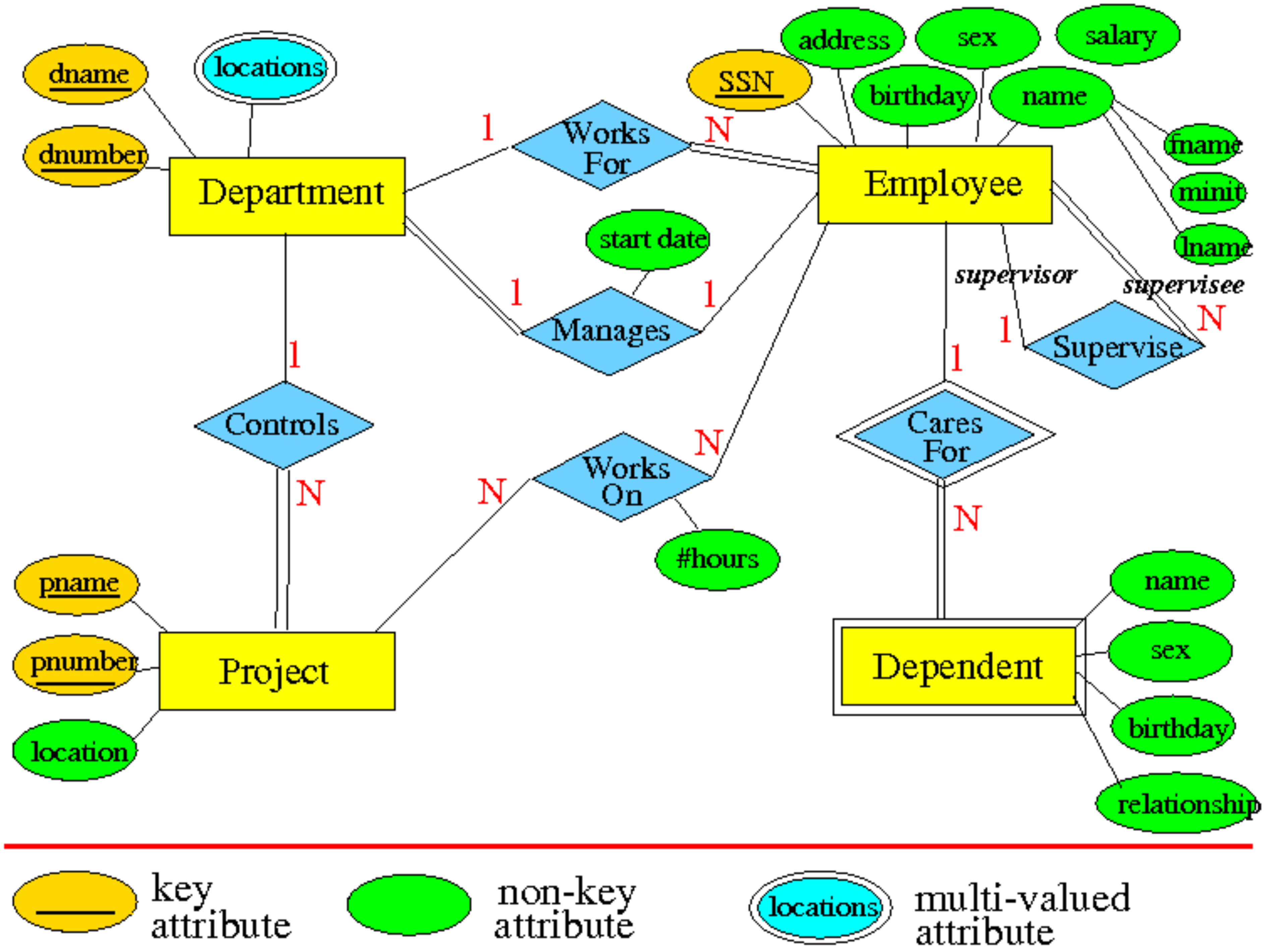


Diagram from Prof Cheung's lecture

ER Model: Recap

- Entity and attributes
- Relationships
  - Degrees
  - Constraints
- Weak entity type
- ER diagram basics
- Design principles