Entity-Relationship Model

CS 377: Database Systems

Homework Announcement

- Homework #1 out
- Due Friday Jan 27th at 11:59 PM
- 3 problems
    - 2 ER (covered today)
    - 1 ER —> relational model (covered next class)

Homework Submission Details

- Gradescope (Entry code: M4W4D9)
  - Homework 0 is a dummy assignment for you to familiarize yourself — worth 0 points
  - Make sure to tag all the pages for each problem according to Gradescope's submission directions — TAs may deduct points on problems that are difficult to find
  - Use of late days requires the instructor to upload

Intro & Database Concepts: Recap

Course Logistics

- Course website contains syllabus, lectures, assignments and example code
  http://joyceho.github.io/cs377_s17/index.html
- Piazza: Main form of communication
  - Announcements, slide corrections, homework clarifications
  - Sign up (use OPUS name or emory email)
    http://piazza.com/emory/spring2016/cs377

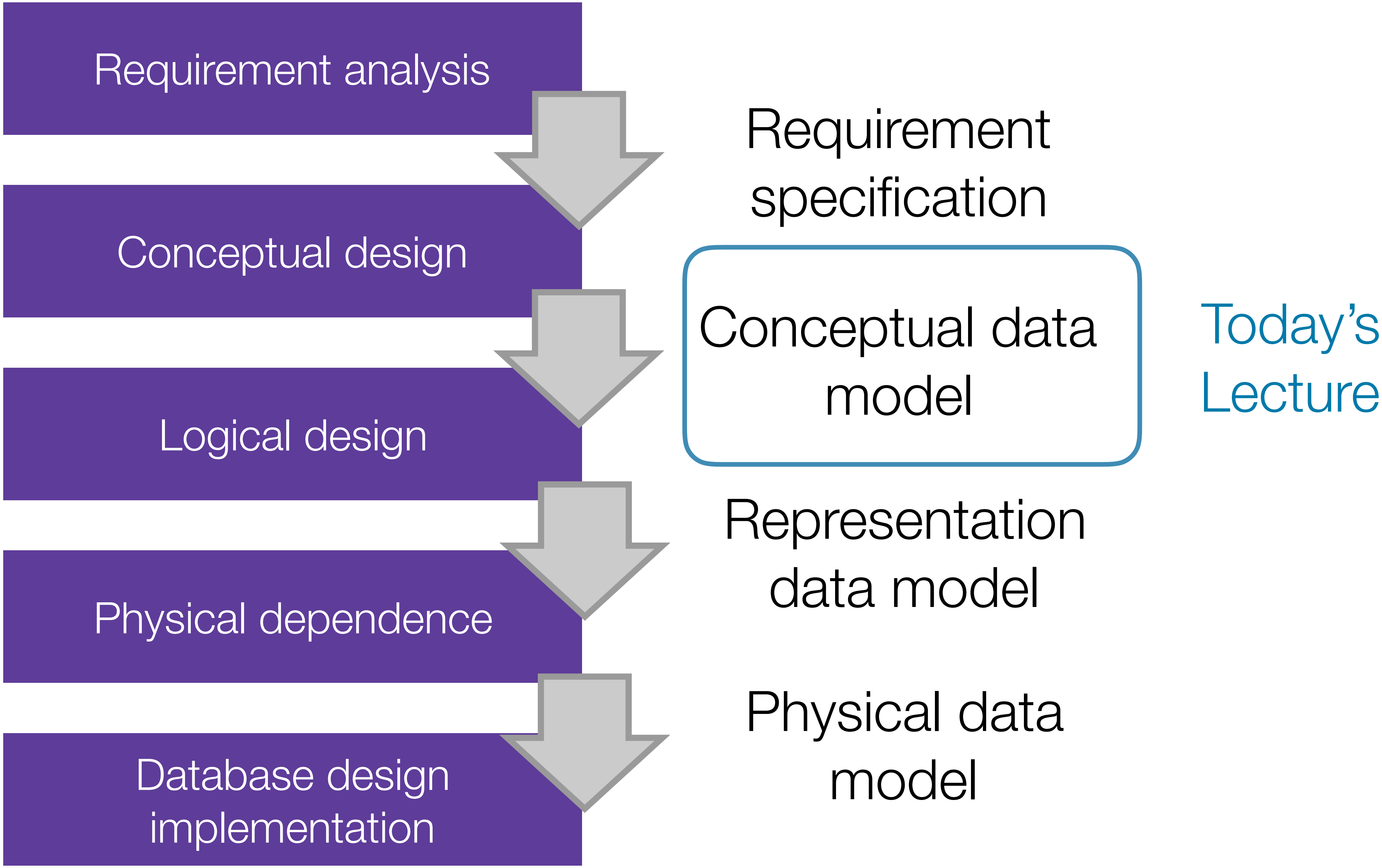
Teaching Staff

- Instructor: Joyce Ho
    - Email: joyce.c.ho@emory.edu
    - Office Hours @ MSC W414
        - M 1:00 pm-3:30 pm
        - W 9:30 am-12:00 pm
- TA: Camilo Valderrama
    - Email: cvalder@emory.edu
    - Office Hours: F 4-5:30 PM @ MSC N410
- TA: Henry Chen
    - Email: henry.chen@emory.edu
    - Office Hours: F 3-4 PM @ MSC E433

Recap: Database Concepts

- Database properties (e.g., scalability, concurrency, etc.)
- Steps for building a database system
- Categories of data models
- Three-schema architecture
- Meta-data
- Physical data & logical independence

Recap: Building a Database System



Requirement analysis

Conceptual design

Logical design

Physical dependence

Database design implementation

Requirement specification

Conceptual data model

Representation data model

Physical data model

Today's Lecture

Today's Lecture

1. ER Basics: Entities & Relationships

2. ER Design

   - Example: Company Database

   - Exercise: Football

Need for Database Design

- Agree on a structure of the database before deciding on a particular implementation
- Consider:
  - What entities to model
  - How entities are related
  - What constraints exist in the domain
  - How to achieve good design (later in course)

Recap: Conceptual Models

- A high-level description of the database
- Sufficiently precise that technical people can understand it
- But, not so precise that non-technical people can participate in the process

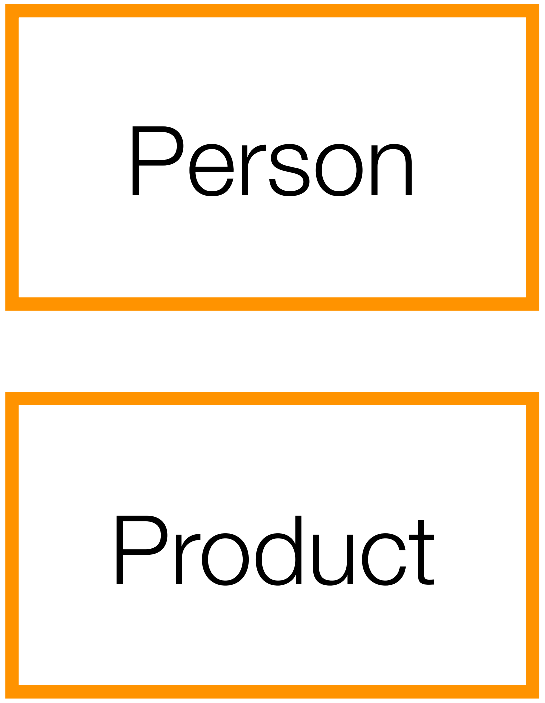
This is where ER models fit in

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
  - One of the most cited articles in CS
- Still very popular with many styles/notations

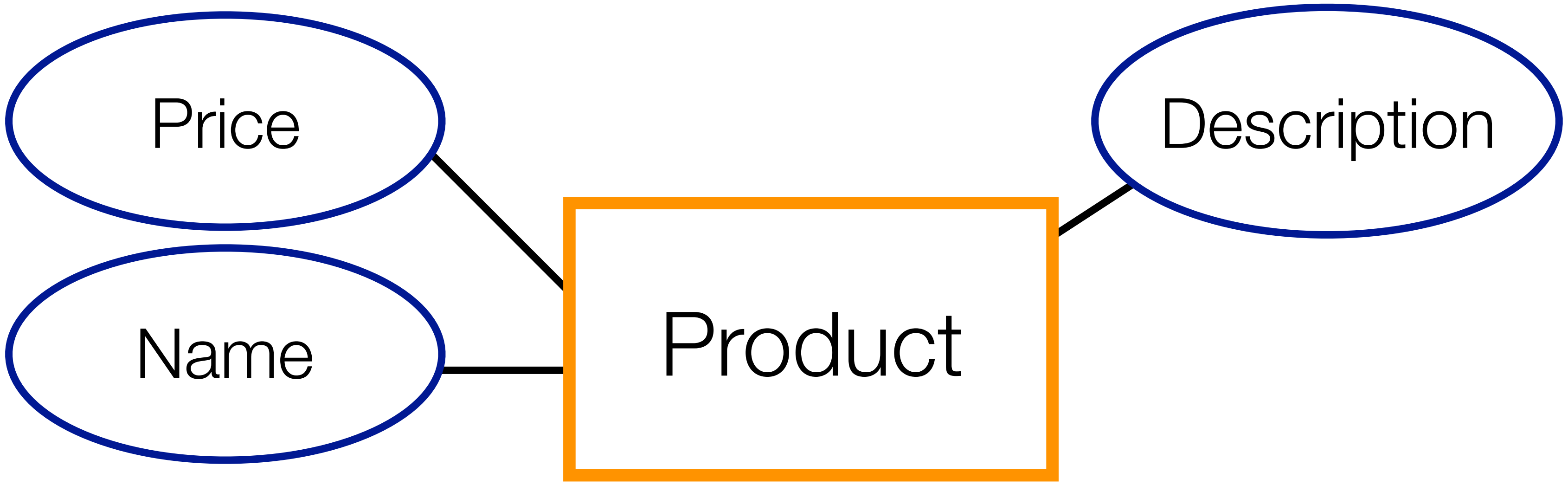
ER Basics: Entity & Entity Set

- **Entity**: individual thing or object
    - Example: A specific person or product
- **Entity set**: a collection of similar entities; classes or types of objects
    - Example: Person, Product
    - Represent sets of all possible entities
    - Shown in diagram as rectangles



ER Basics: Entity Attributes

- **Attribute**: properties used to describe an entity
  - Represented by ovals attached to an entity set
  - 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

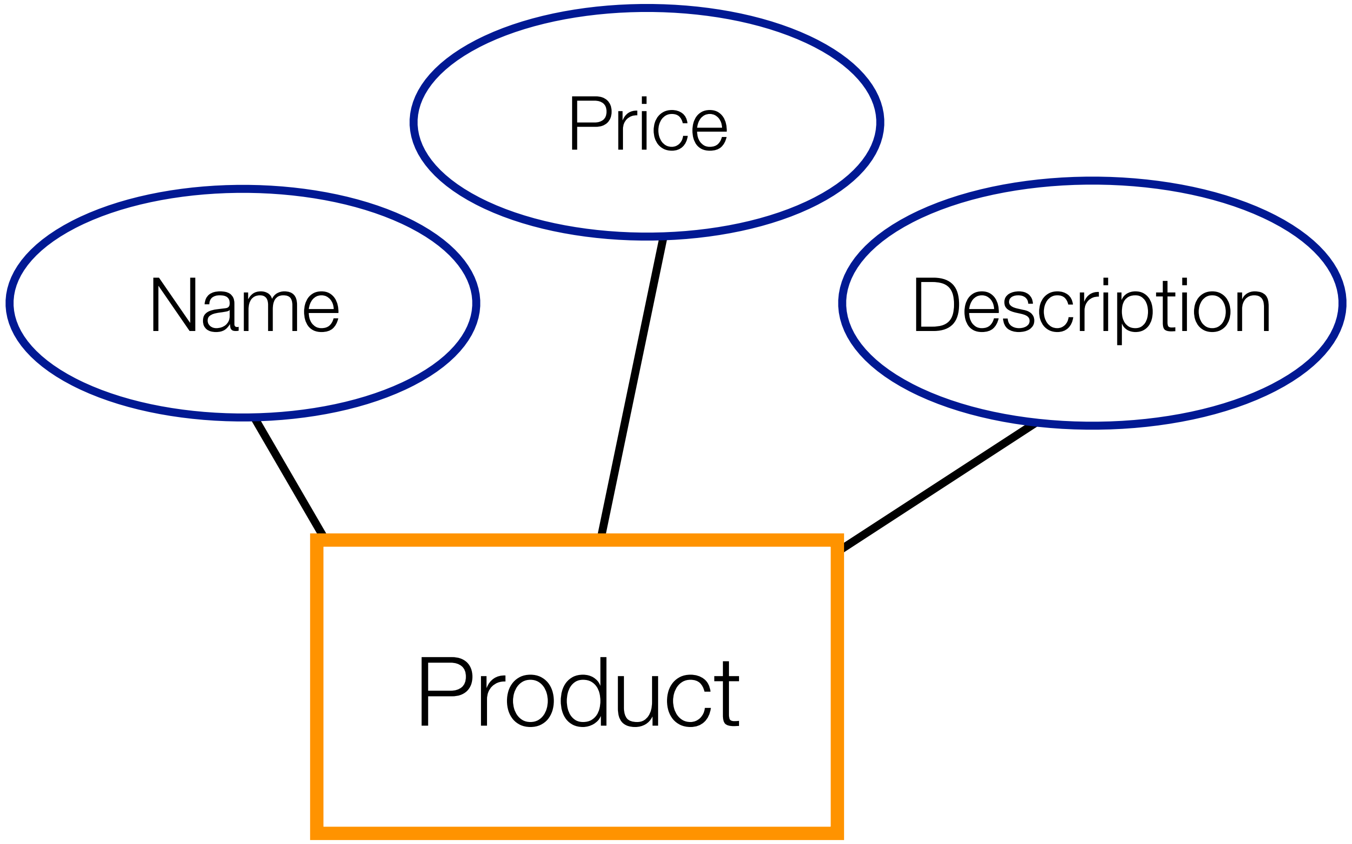


Entities and Entity Sets

Product

Name: Xbox
Price: $250
Description: Multimedia System

Name: Fisher-Price Deluxe Gym
Price: $50
Description: Baby Gym & Playmat



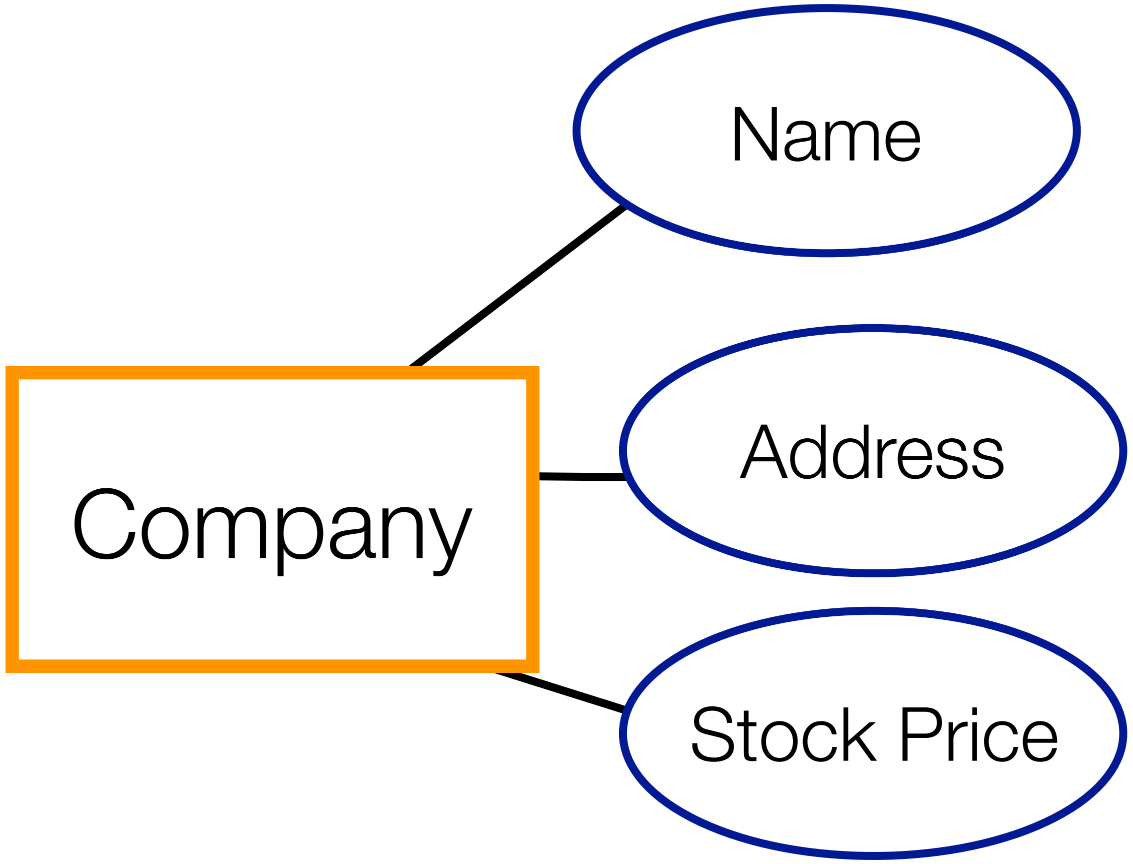
Entities are NOT explicitly represented in ER diagrams

Example: Company database

- Each company has a name, address, and the stock price
- Each product has a name and description
- Each employee has a name, address, and social security number

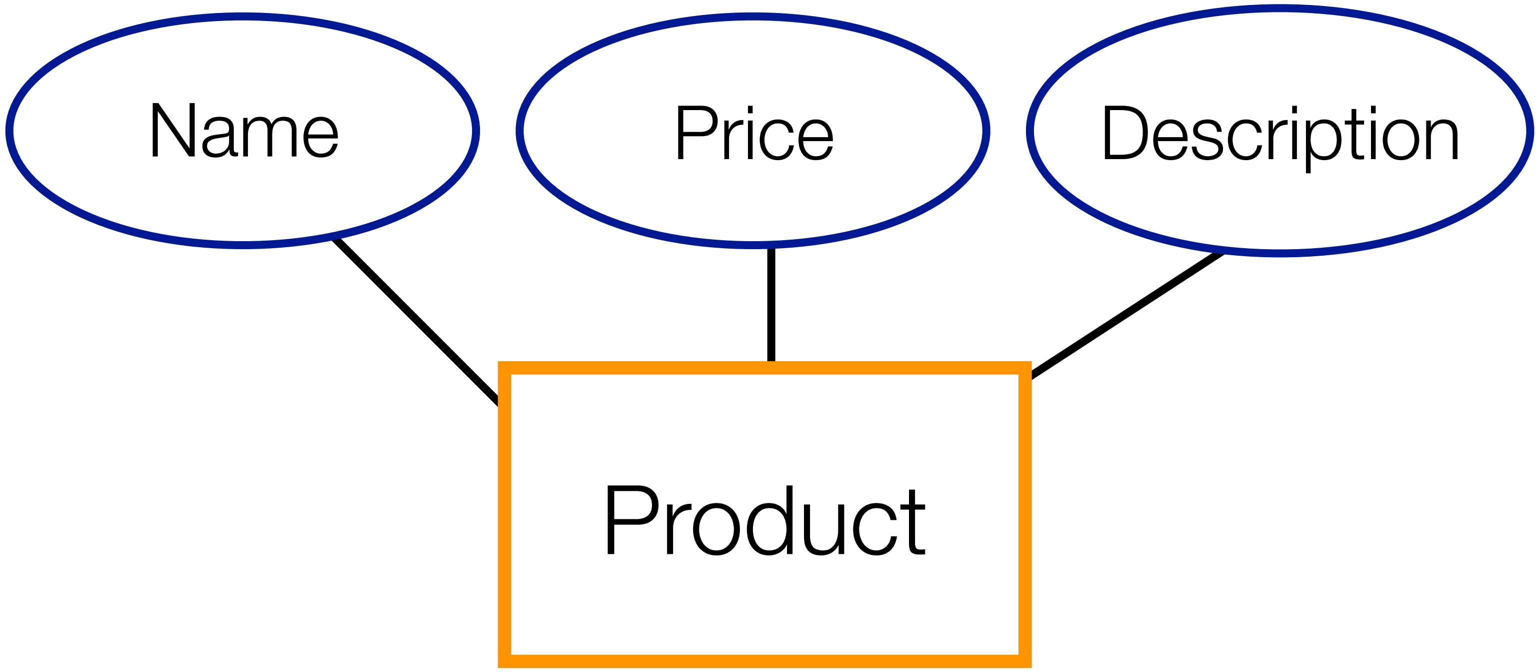
Example: Company database

- Each company has a name, address, and the stock price



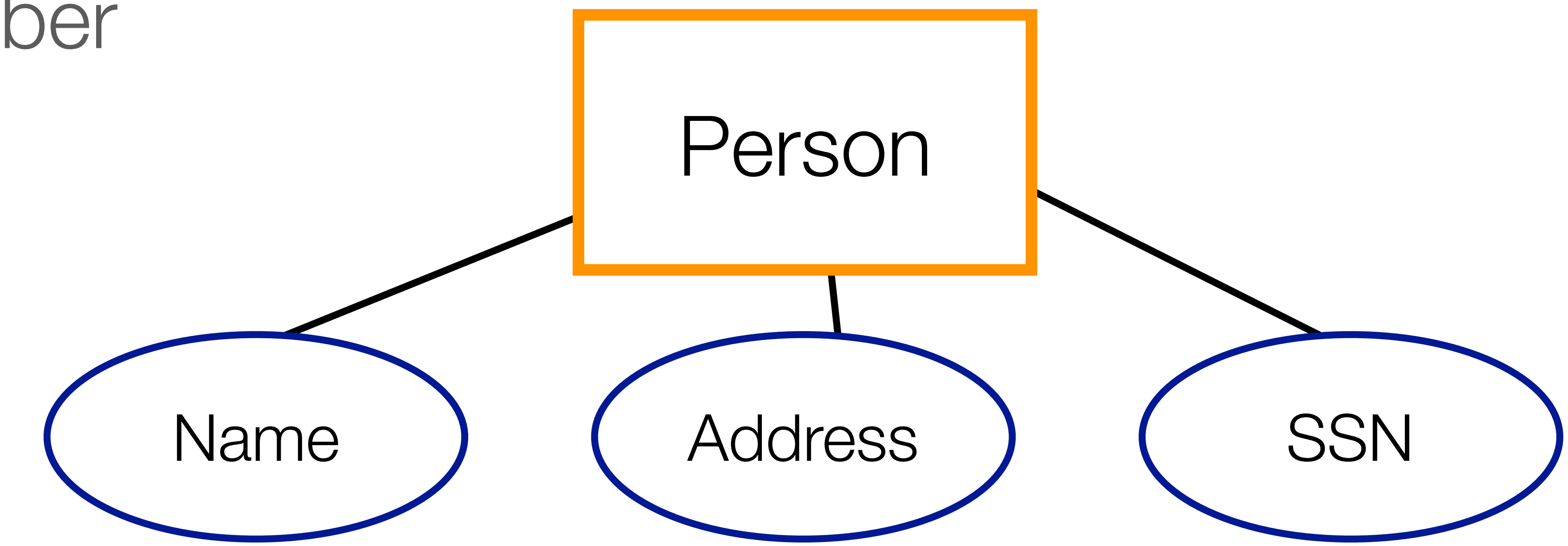
Example: Company database

- Each company has a name, address, and the stock price
- Each product has a name and description

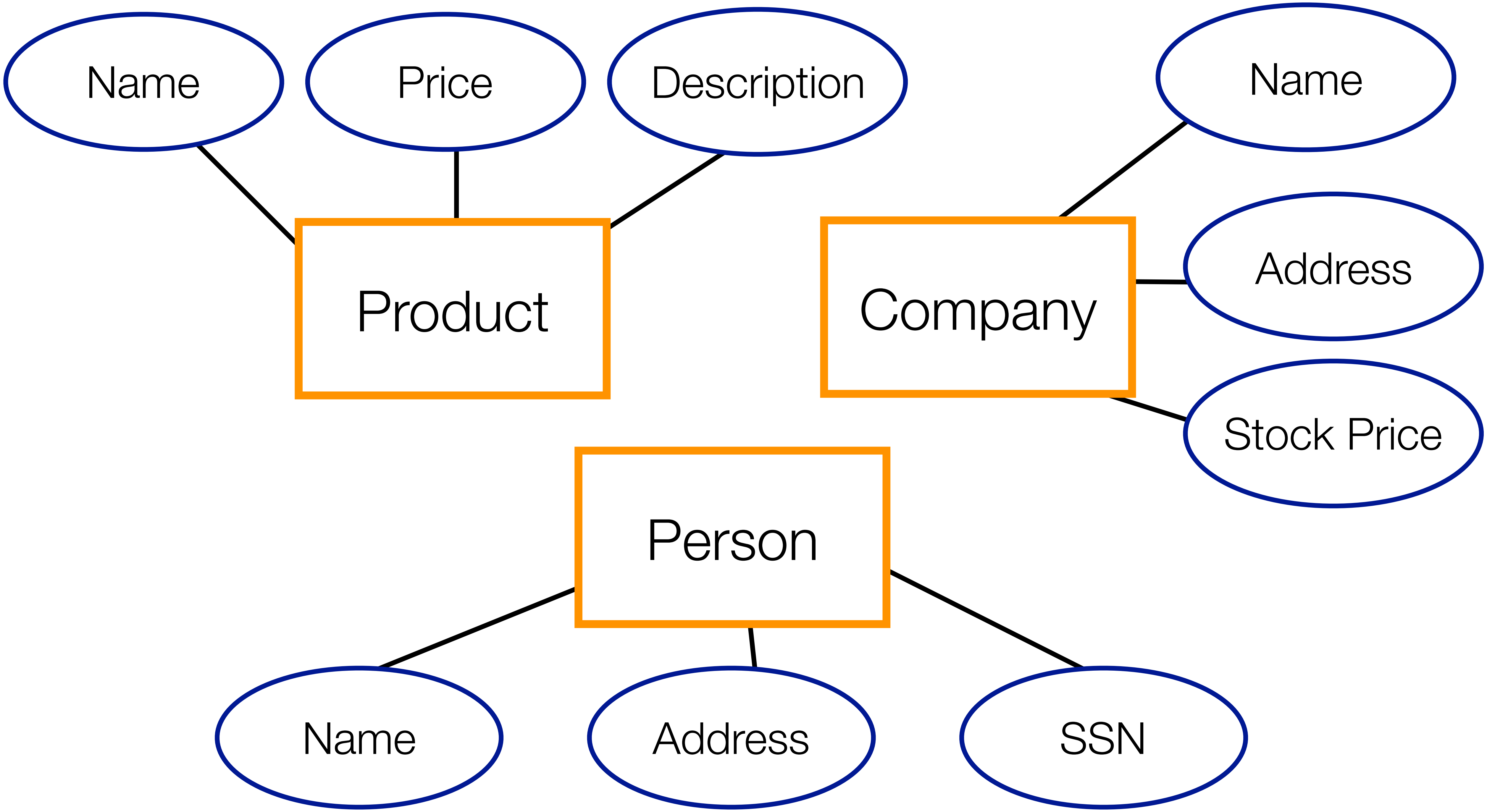


Example: Company database

- Each company has a name, address, and the stock price
- Each product has a name and description
- Each employee has a name, address, and social security number



Example: Company database

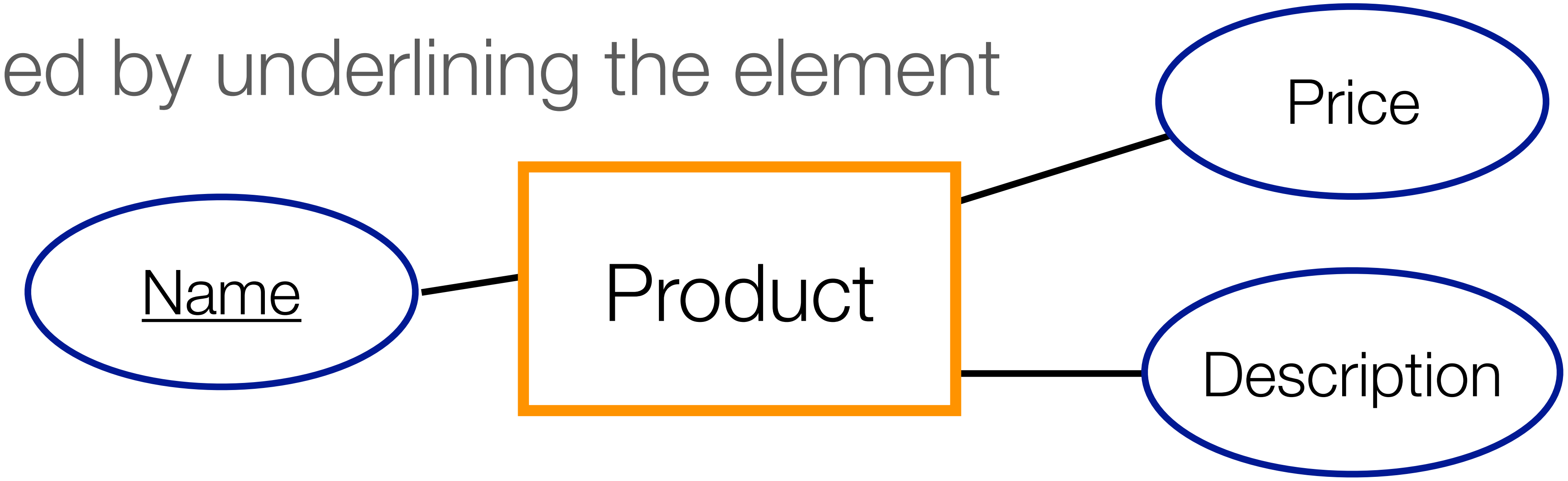


ER Basics: Attribute Types

- **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})

ER Basics: Key Attributes

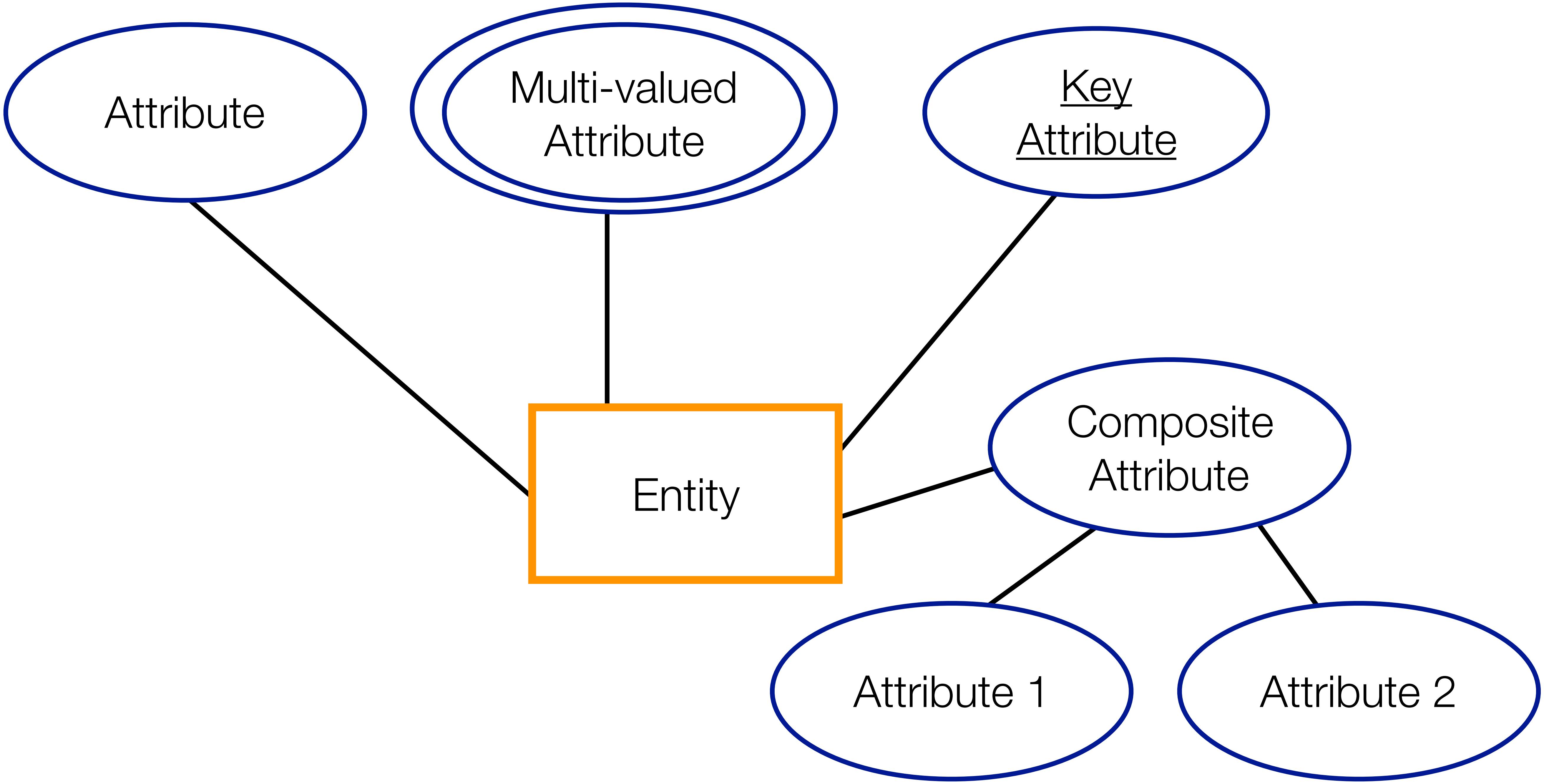
- **Key attributes**: a set of attributes for which *no two different entities* will have the same values (e.g., SSN for people, VIN for cars)
  - Can be used to identify the entity uniquely
  - Entity should have at least one key attribute
  - Represented by underlining the element



ER Basics: Other 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
- **NULL** value: can mean not available or not applicable
  - Equality comparison of two attribute values both equal to NULL should return FALSE

ER Basics: Attribute Types



Attribute

Multi-valued Attribute

Key Attribute

Entity

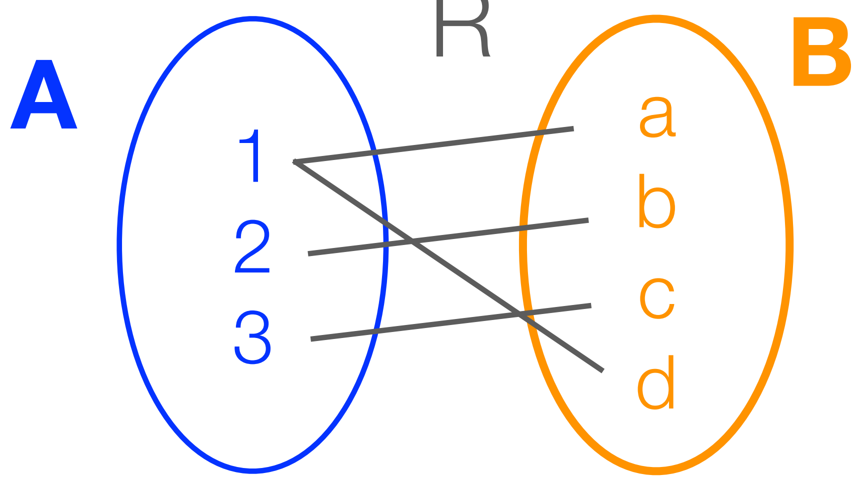
Composite Attribute

Attribute 1

Attribute 2

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:



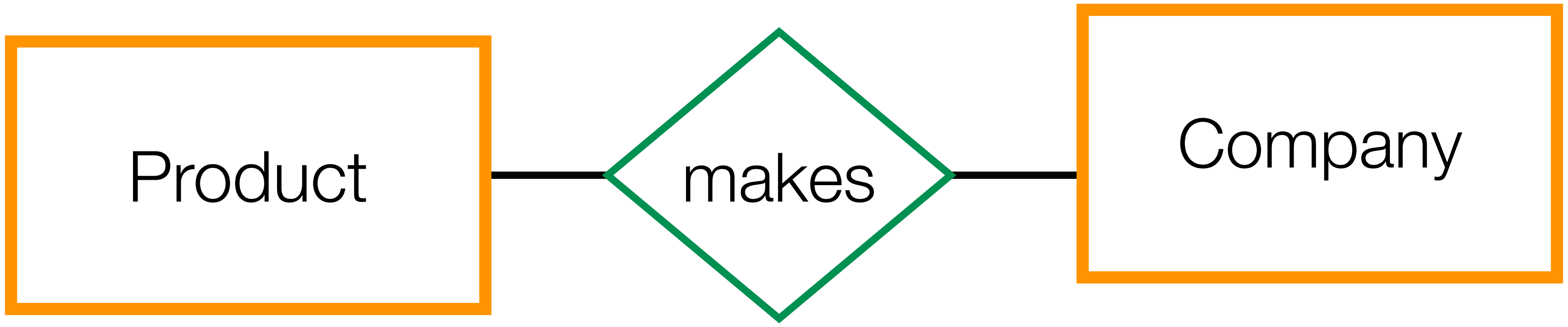
  - **A** = {1,2,3}, **B** = {a, b, c, d}
  - **A** x **B** = all pairs of tuples {(1,a), (1,b), (1,c) (1,d), (2,a), (2,b), (2,c), (2,d), (3,a), (3,b), (3,c), (3,d)}
  - **R** = {(1, a), (1,d), (2,b), 3(c)}

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)

  - Denoted with a diamond connecting two entities



Relationship Types

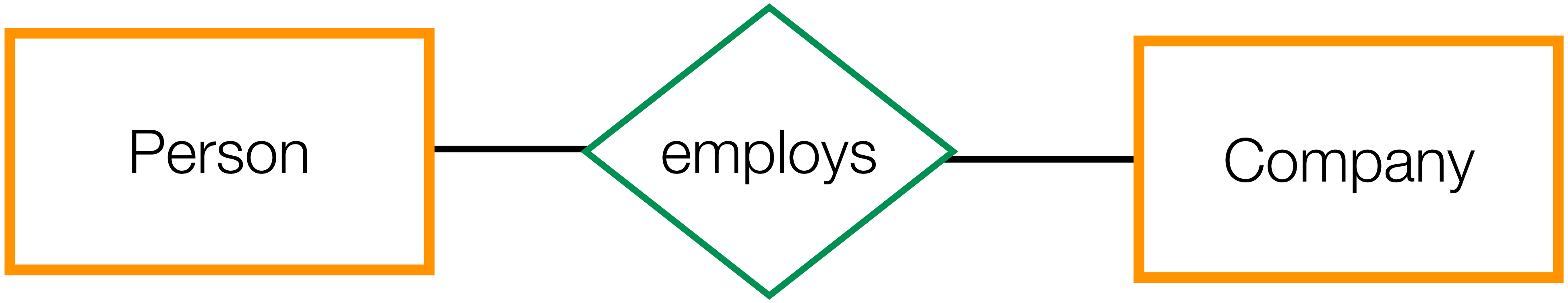
- Relationship type **R** = any subset of the cartesian product among entity types **E1, E2, …, E_n**
- More than one relationship type can existing between two participating entity types
- Relationships can have attributes as well

Example: Company database

- Each company has a name, address, and the stock price
- Each product has a name and description
- Each employee has a name, address, and social security number
- Each company has a list of employees
- List of products manufactured by the companies

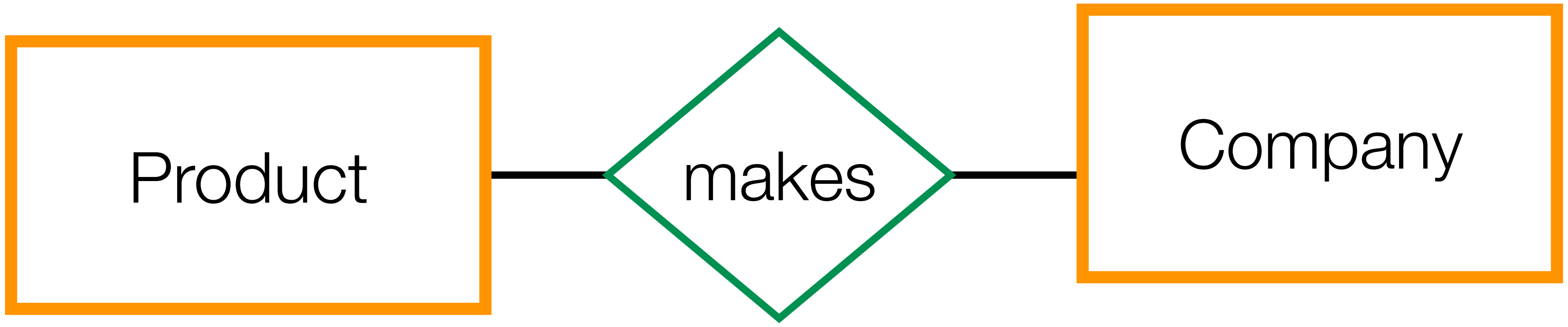
Relationship between 2 entities

Example: Company database



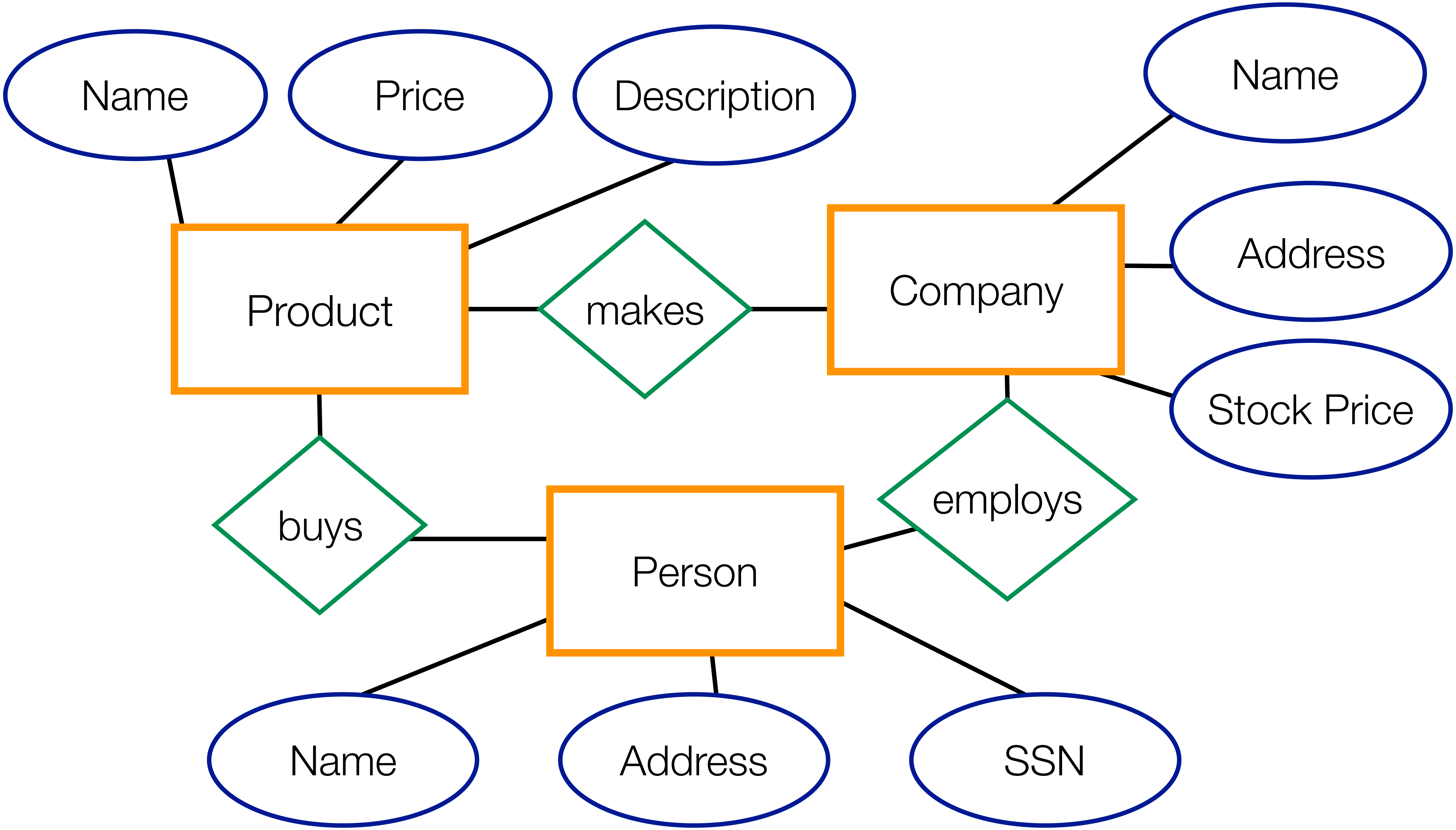
- Each company has a list of employees

Example: Company database



- List of products manufactured by the companies

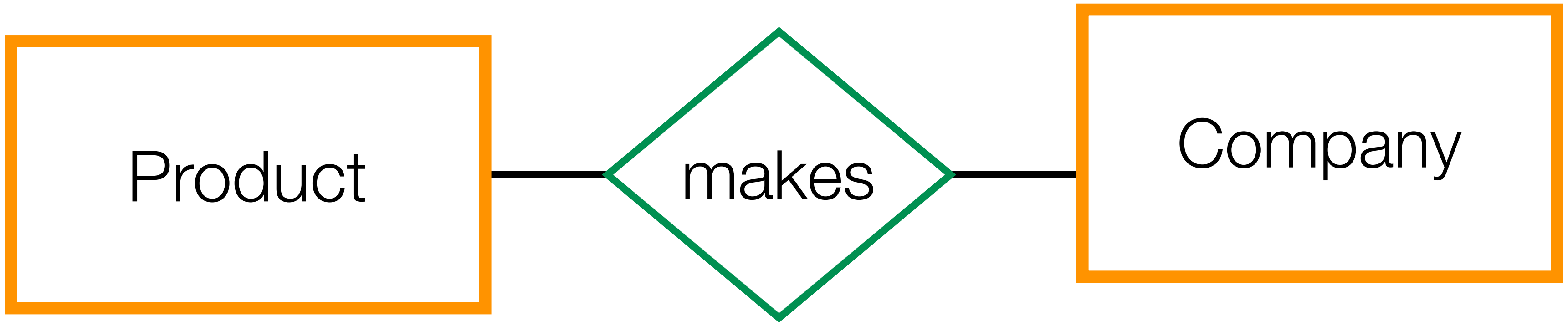
Example: Product Database



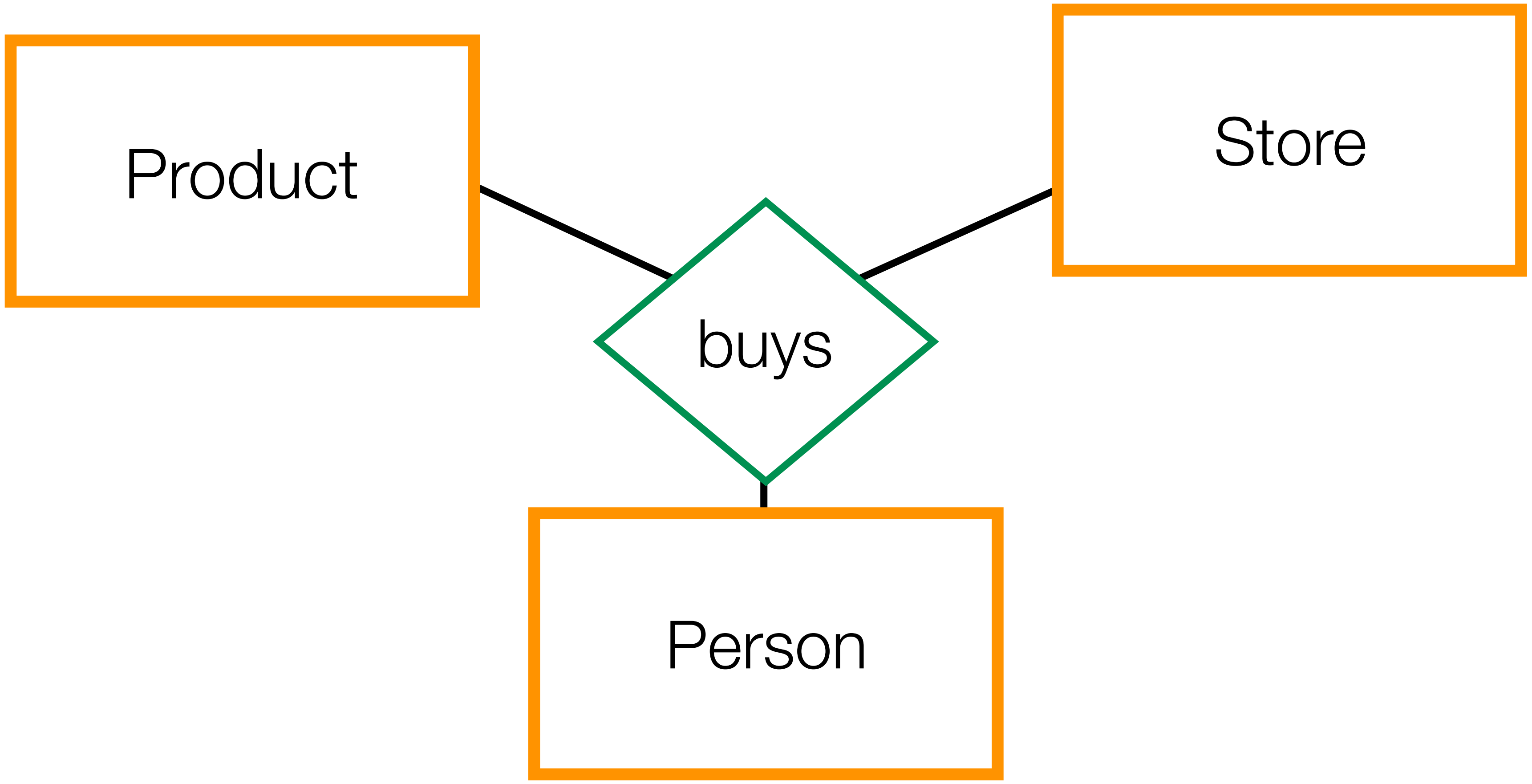
Relationship Degree

- **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

Example: Relationship Degree



Binary degree

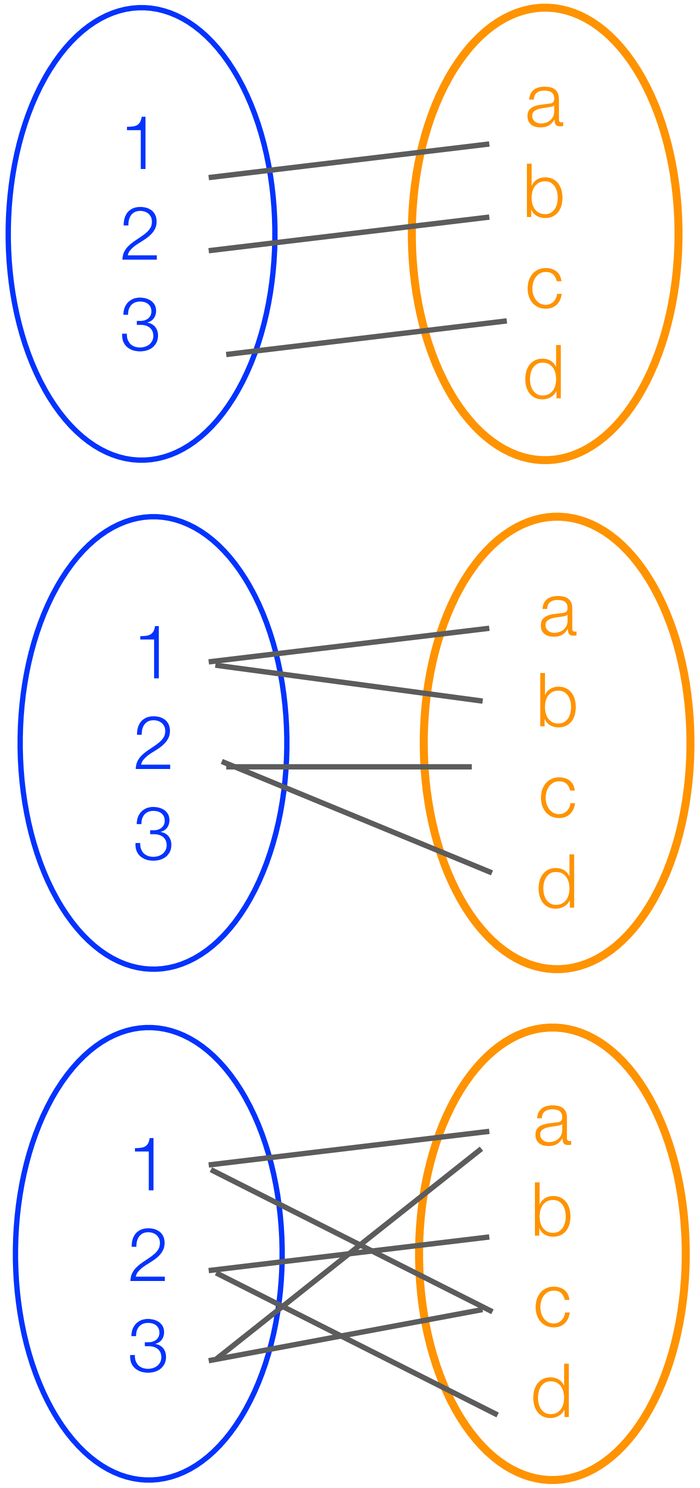


Ternary degree

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

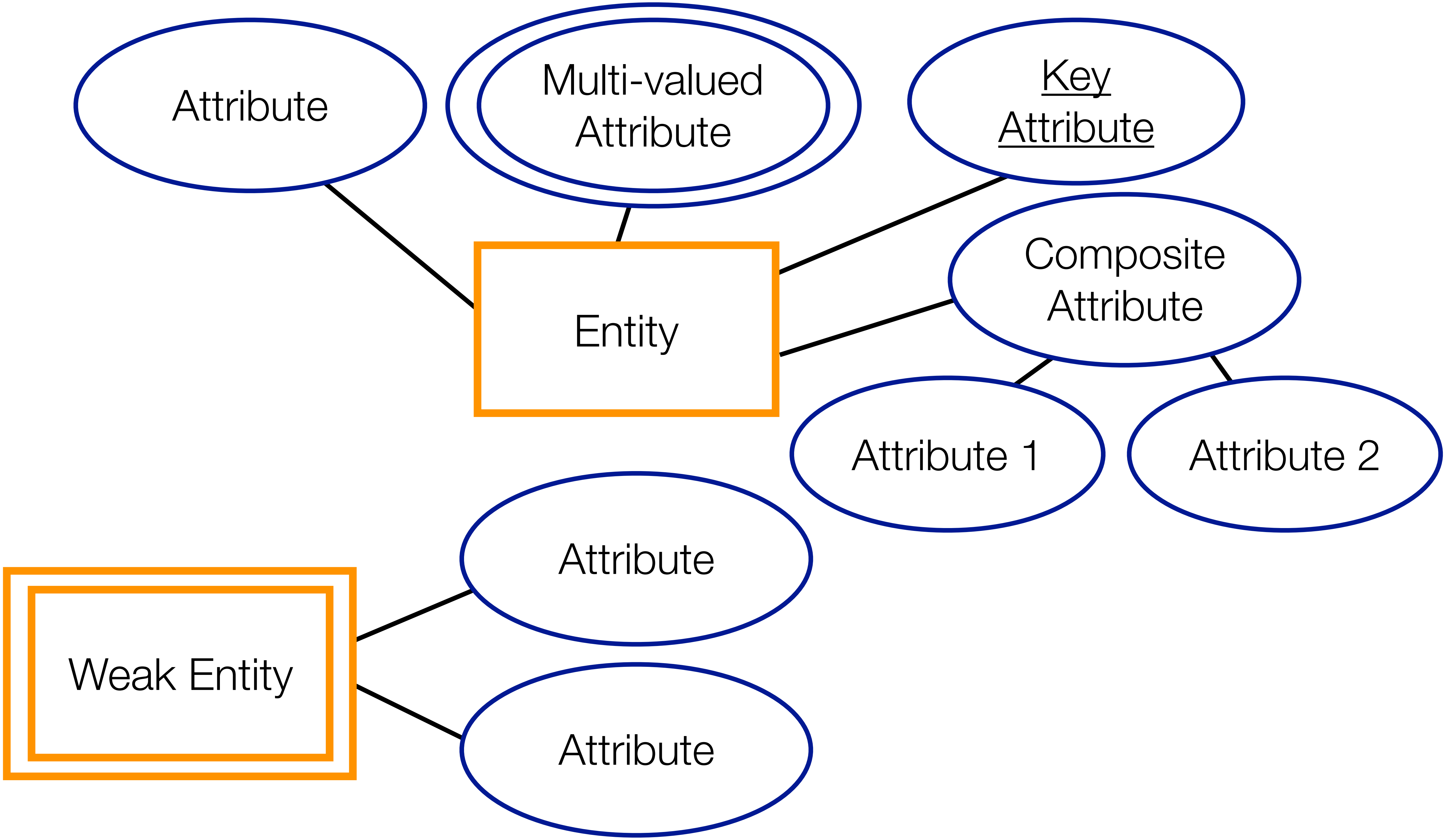
- Relationships can be **recursive** with both participants having same entity type in different roles
  (e.g., DEAN is a PROFESSOR that SUPERVISES another PROFESSOR)

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

Weak Entity Types

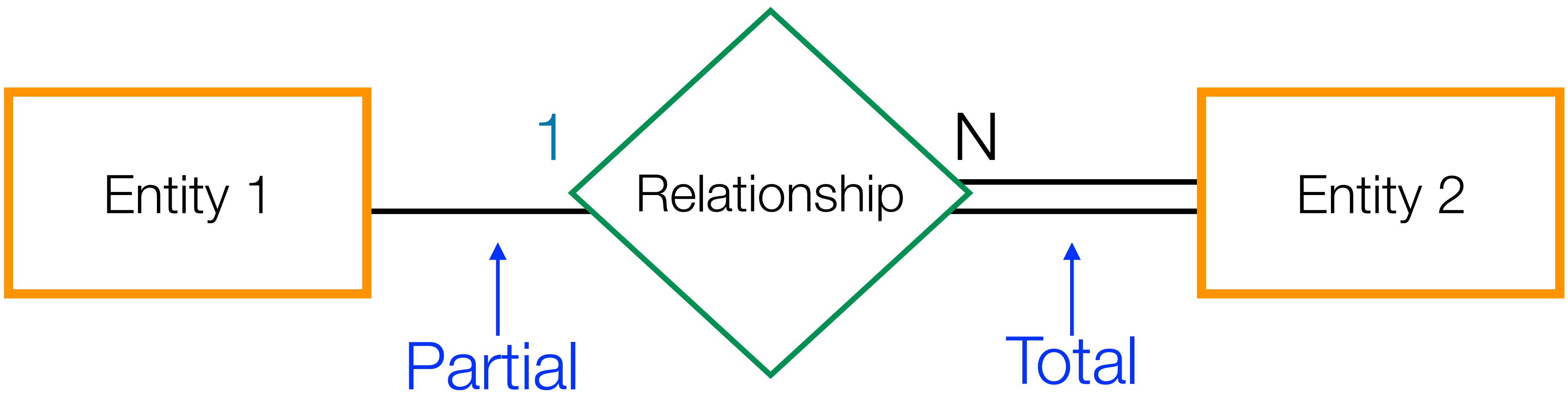
- Entity that does not have a key attribute and participates in an identifying relationship with an owner or identifying entity type
- Identified by a combination of:
  - Partial key of the weak entity type
  - Particular entity they are related to in the identifying entity type

ER Diagram: Entities & Attributes



ER Diagram: Relationships

Entity 1: Entity 2 = 1 : N



Steps for ER Design

1. Gather information or requirements

2. Identify the entities - which things are important enough to be identified with a key

3. Identify the properties/attributes of the entities

4. Determine the relationships (usually properties that occur between 2 or more entities)

    1. Cardinality ratio constraints on binary relationships

    2. Participation constraints

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)

Example: Company Database (2)

- 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

- 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)

Example: Company Database (3)

- 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

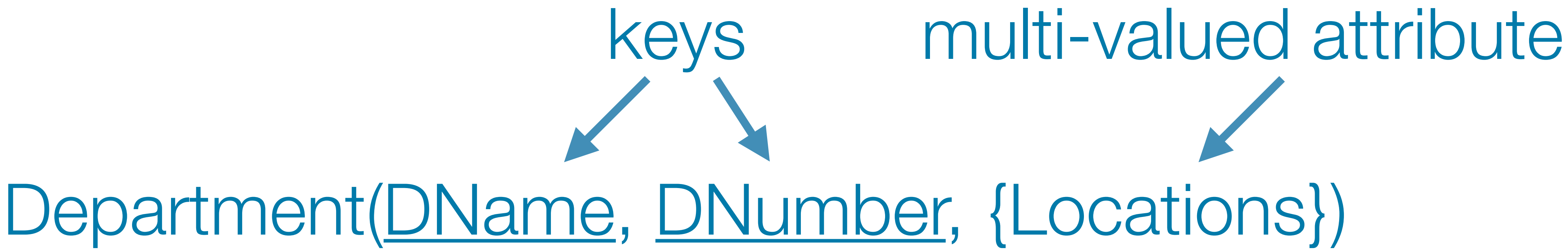
Example: Identify Company Entities

- 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

- A department may have several locations (e.g., Atlanta, Boston, LA)



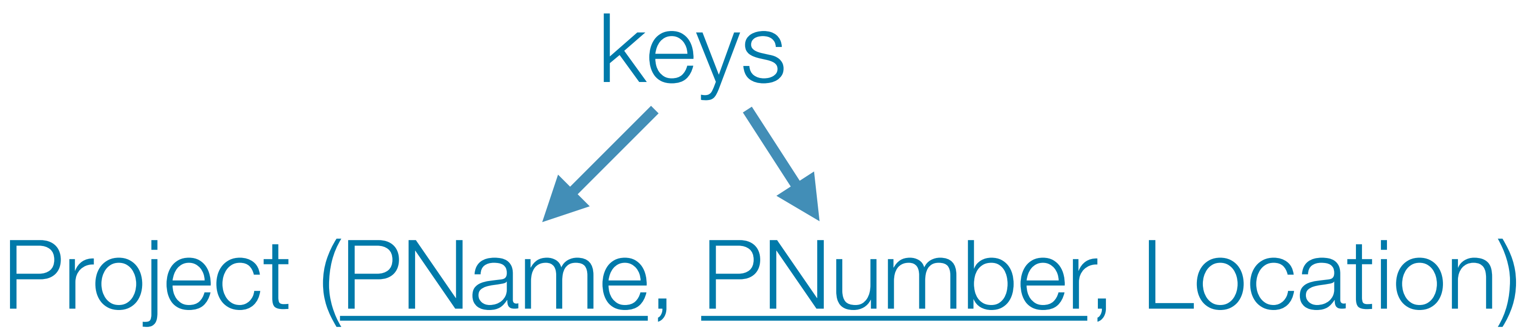
keys

multi-valued attribute

Department(DName, DNumber, {Locations})

Example: Identify Entity Attributes (2)

- 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



keys

Project (PName, PNumber, Location)

Example: Identify Entity Attributes (3)

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

key

Employee ($\underline{\text{SSN}}$, Name, Addr, Salary, Sex, BDate)

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

weak entity without a key

Dependent(FName, Sex, BDate, RelationToEmp)

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)

Example: Company Database (2)

- 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

- 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)

Example: Company Database (3)

- 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: Determine Relationships

- 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)

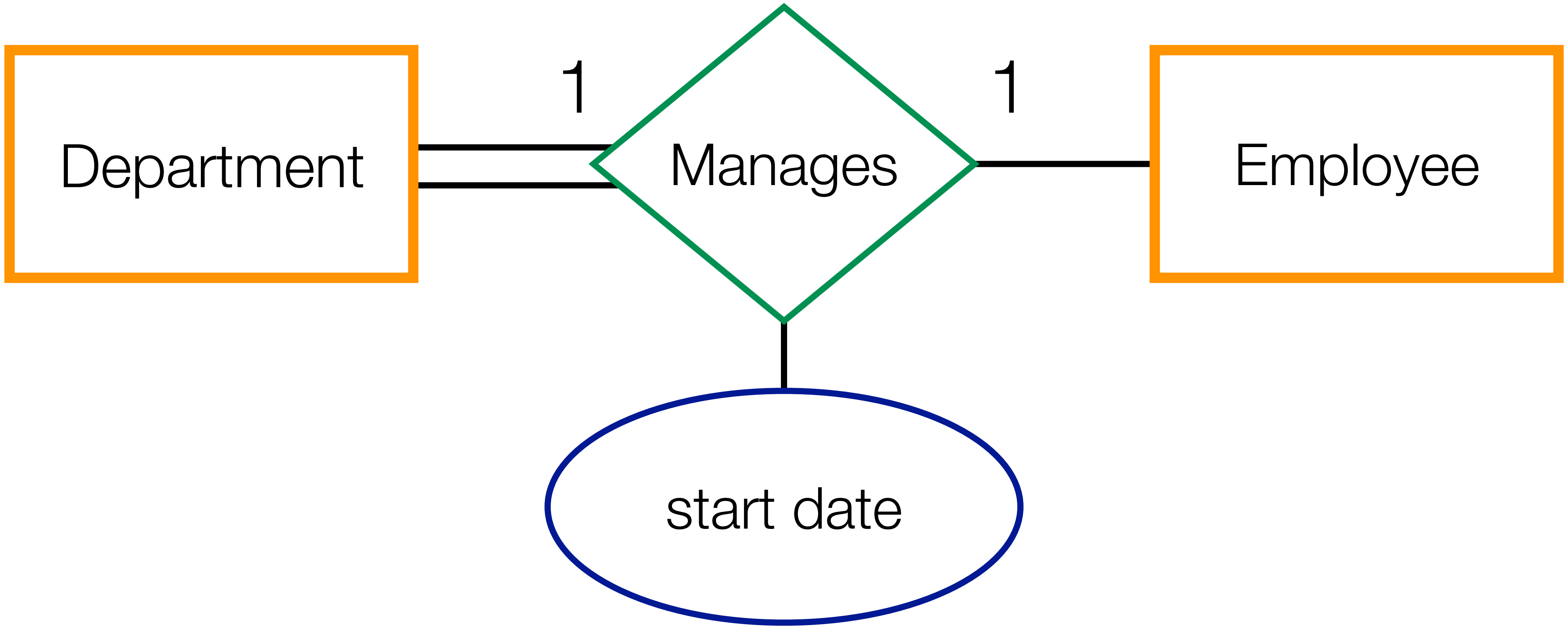
Manager(Employee, Department)

- 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

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: Determine Relationships (2)

- 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

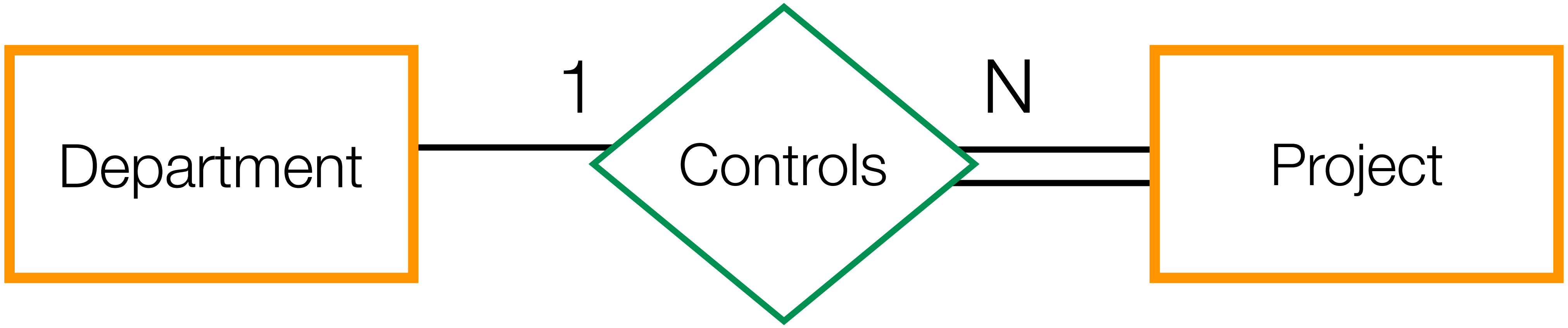
Controls(Department, Projects)

- 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

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: Determine Relationships (3)

- 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)

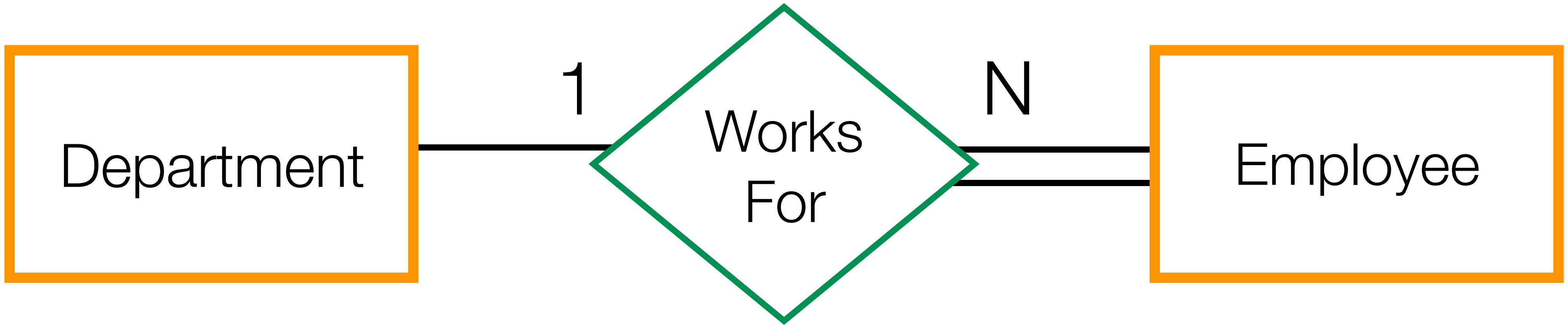
WorksFor(Employee, Department)

- 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

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: Determine Relationships (4)

- 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)

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)

Example: Determine Relationships (5)

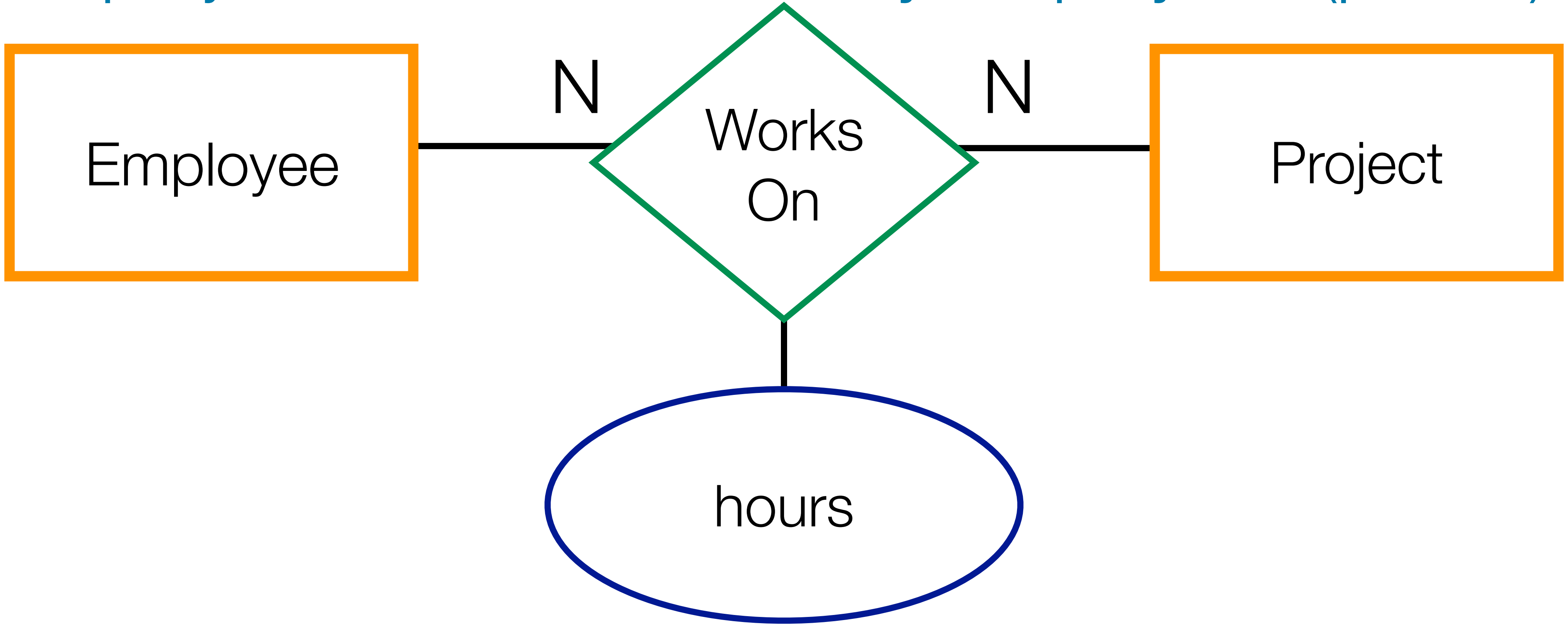
- Company tracks the number of hours per week an employee works on each of his/her projects

Hour attribute for employee provides information about relationship between an employee and a project

Example: WorksOn Relationship

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)



Example: Determine Relationships (5)

- Each employee has one direct supervisor (also an employee of the company)

Supervisor(Employee, Employee)

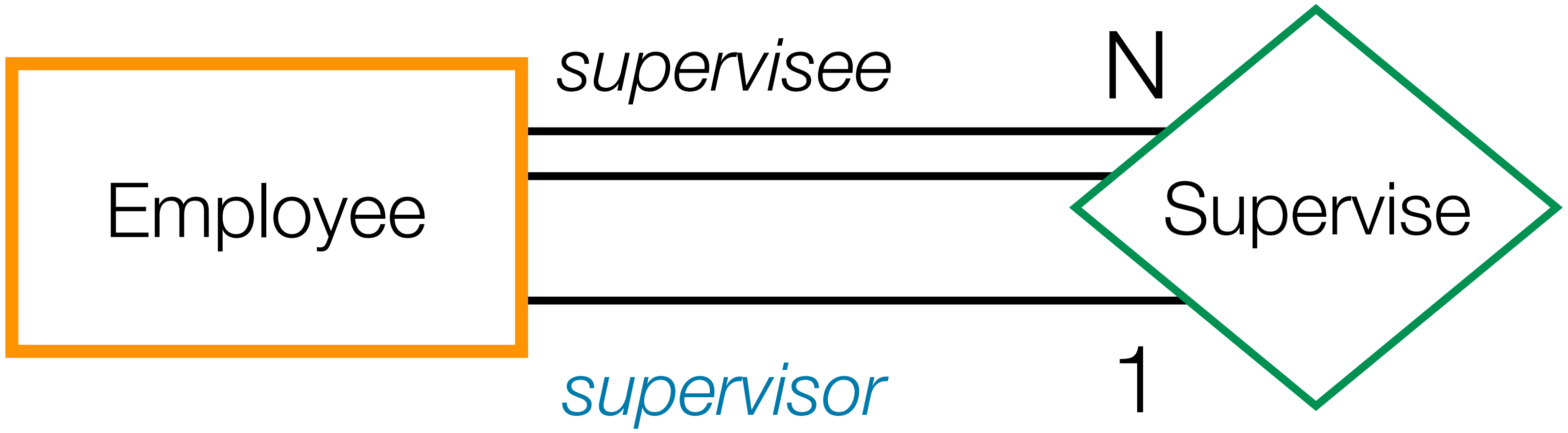
- To distinguish the two different entities, we assign two different roles: supervisor and supervisee
- 1 supervisor employee supervises N employees
- 1 supervisee employee has 1 supervisor employee
- A employee need not manage any employee (partial)
- A employee must have a supervisor (total)

A recursive relationship as it is a relationship between two entities from the same entity set

Example: Supervisor Relationship

Supervisor(Employee, Employee)

- 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)



Example: Determine Relationships (6)

- 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

- 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

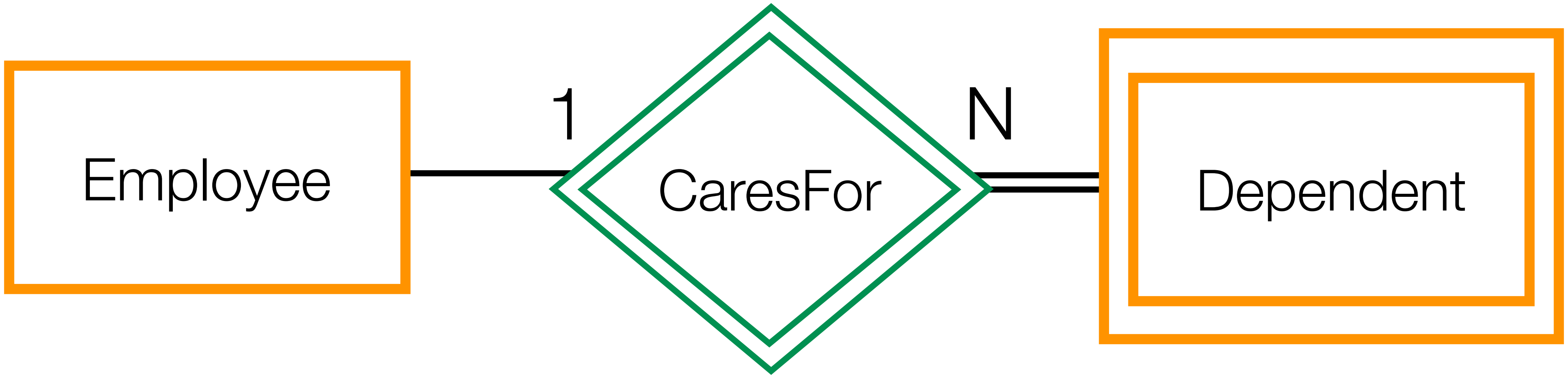
CaresFor(Employee, Dependent)

- 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

Example: CaresFor Relationship

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



weak relationship type is represented by a double diamond

Example: Company ER Diagram

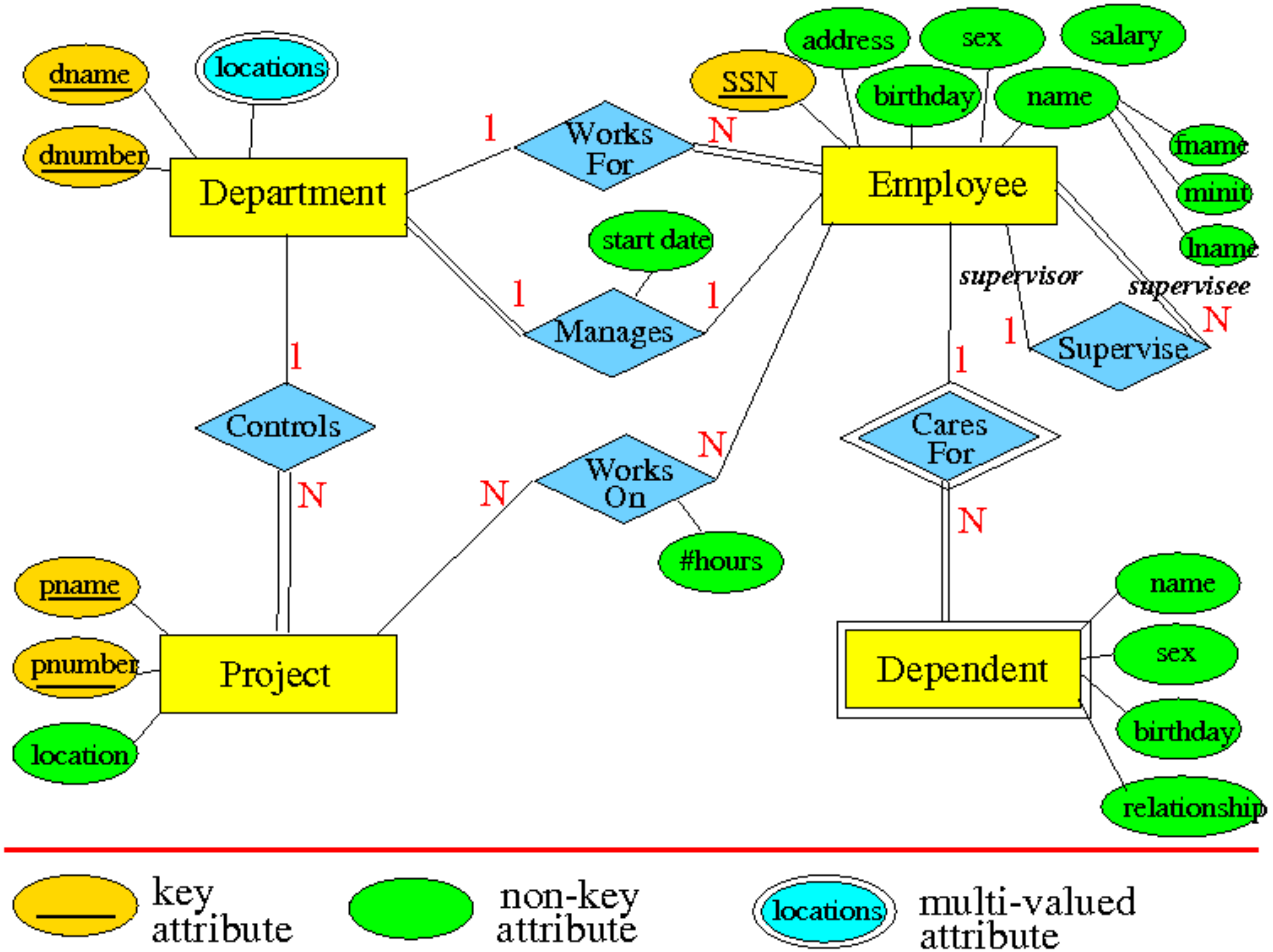
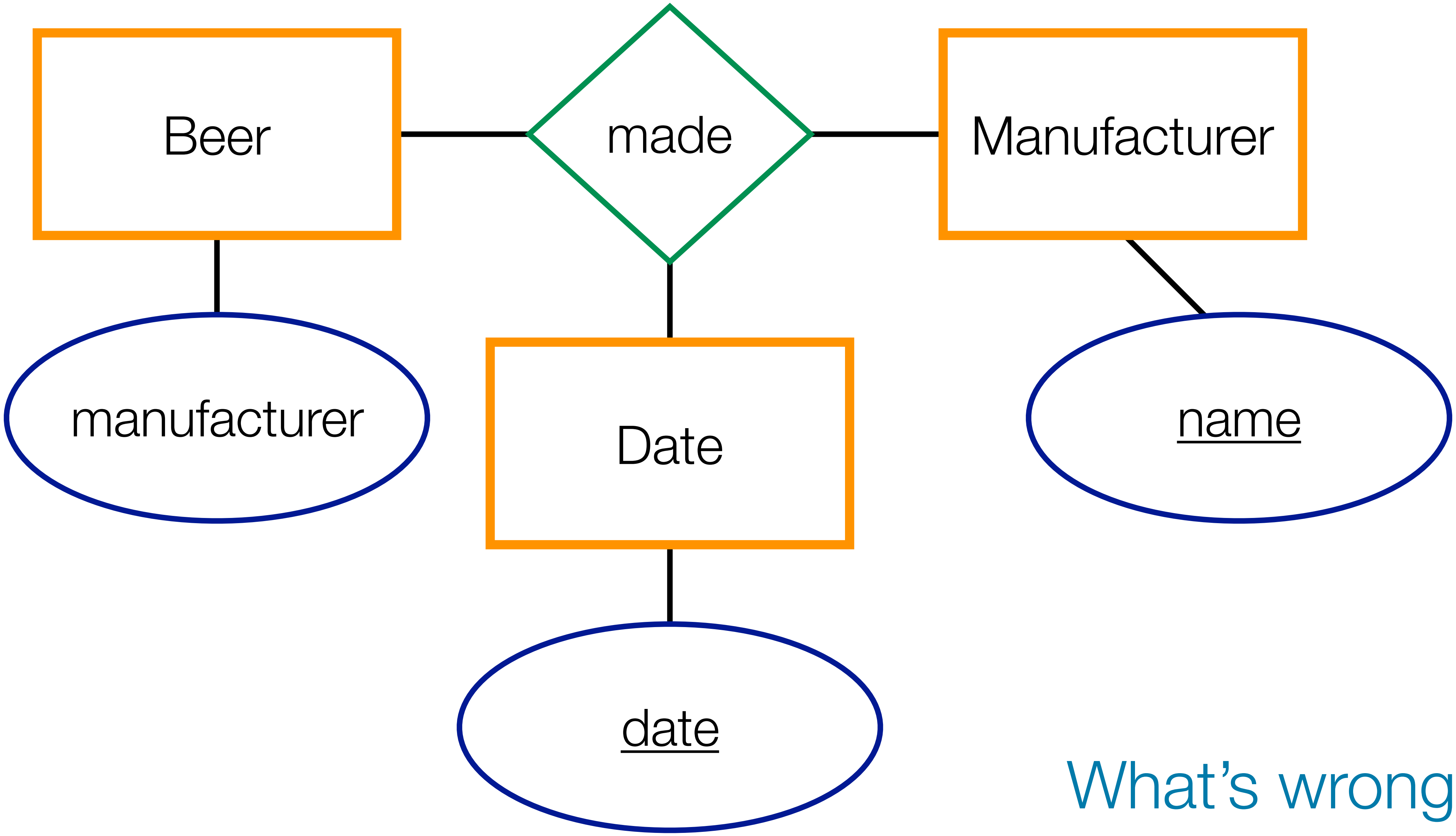


Diagram from Prof Cheung's lecture

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

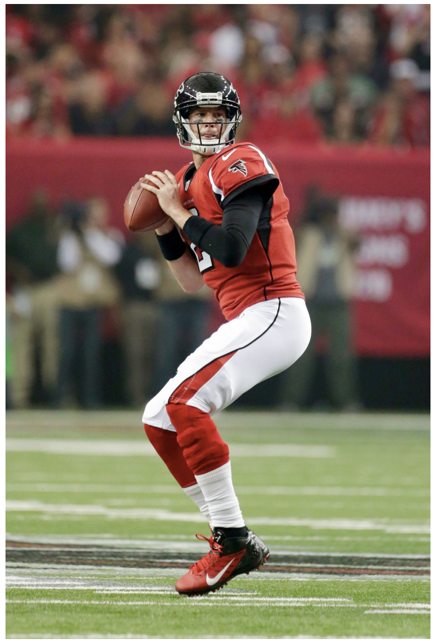
Example of a Bad ER Model



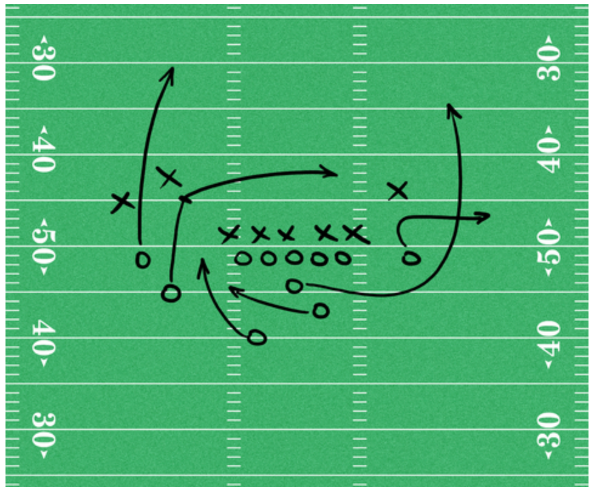
What's wrong?

Exercise: ER Diagram for Football

Teams play each other in games. Each pair of teams can play each other multiple times.



Players belong to teams (assume no trades or changes)



A game is made up of plays that result in yardage gain or loss and potentially a touchdown

A play will contain a pass from one player to another or a run by one player

ER Model: Recap

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