

Entity Relationship (ER) Model

CSCI 220: Database Management and Systems Design

Practice Quiz

- Sign in
- Describe at least **three capabilities** of database management systems
 - Scrap paper is available at the front
- Discuss with a neighbor
- After five minutes, a student will be randomly selected to share with the class

Today you will learn:

- How to design a database. In particular:
 - Requirements elicitation
 - Entity Relationship (ER) modeling

Where Do Databases Come From?

- When implementing an application, list functional and non-functional requirements
 - **Functional requirements:** What should the system do, from a user's perspective?
 - **Non-functional requirements:** execution and evolution qualities. For example: performance, maintainability, portability, etc.
- An app's database is implemented to meet the app's requirements

Where Do Requirements Come From?

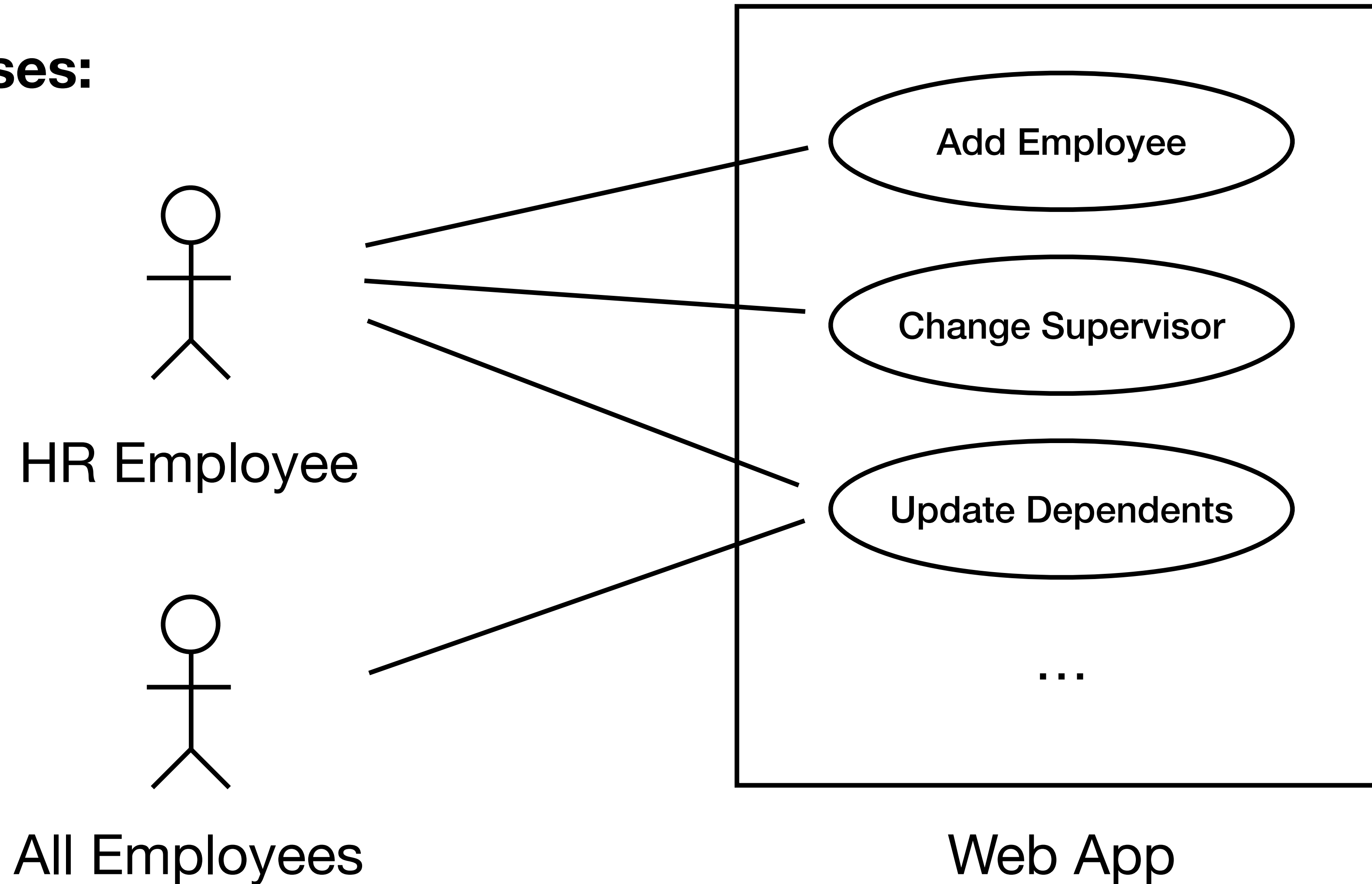
- Interview stakeholders!
 - Don't assume you already know what the application should do: use your knowledge to ask good questions
- Example application: a COMPANY database used to track employees, supervisors, projects, etc.
 - Implementation is straightforward if you have a (near) complete list of requirements

COMPANY Database Functional Requirements

- **Main idea:** human resources needs a web app to track employees
- **User stories:**
 - “As an HR employee, I need to reassign an employee to a different supervisor if their current supervisor leaves the company.”
 - “As an employee, I need to notify HR when my child is born, so I can receive parental leave.”
 - ...

COMPANY Database Functional Requirements

- **Use cases:**



COMPANY Database Functional Requirements

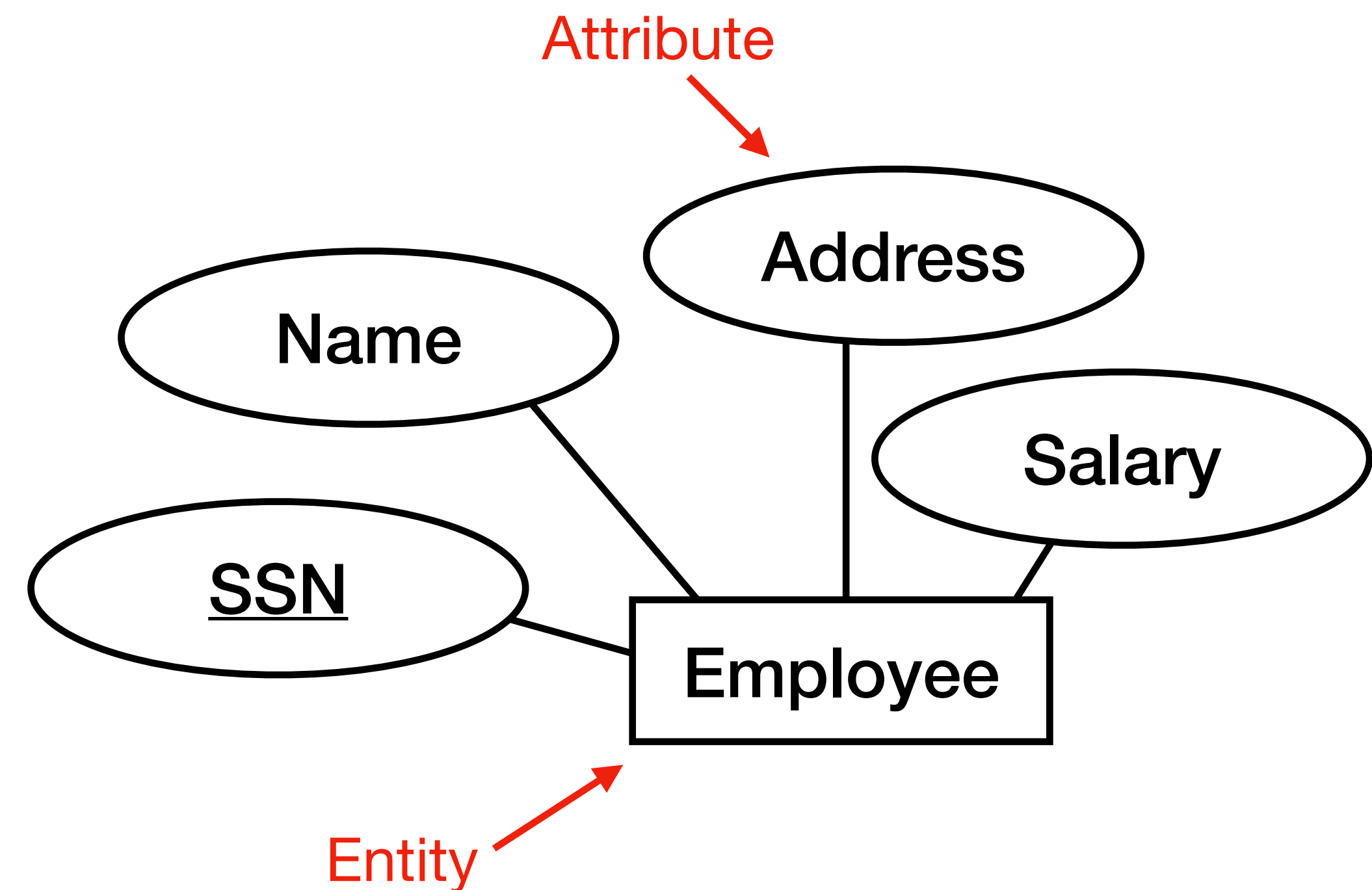
- **Data requirements: Store...**
 - Each Employee's name, SSN, dependents, ...
 - Each Department's name, number, locations, ...
 - Supervisor and management relationships
 - ...

Modeling Data Requirements

- Textual descriptions of data requirements are helpful, but language is imprecise
- An **entity-relationship model (ER model)** shows relationships and constraints in a detailed and unambiguous way
- Eventually, we will convert this to the tabular **relational model**
 - We start with the ER model because it is easy to iterate on. In particular, it lets us defer decisions about exactly which tables to create.

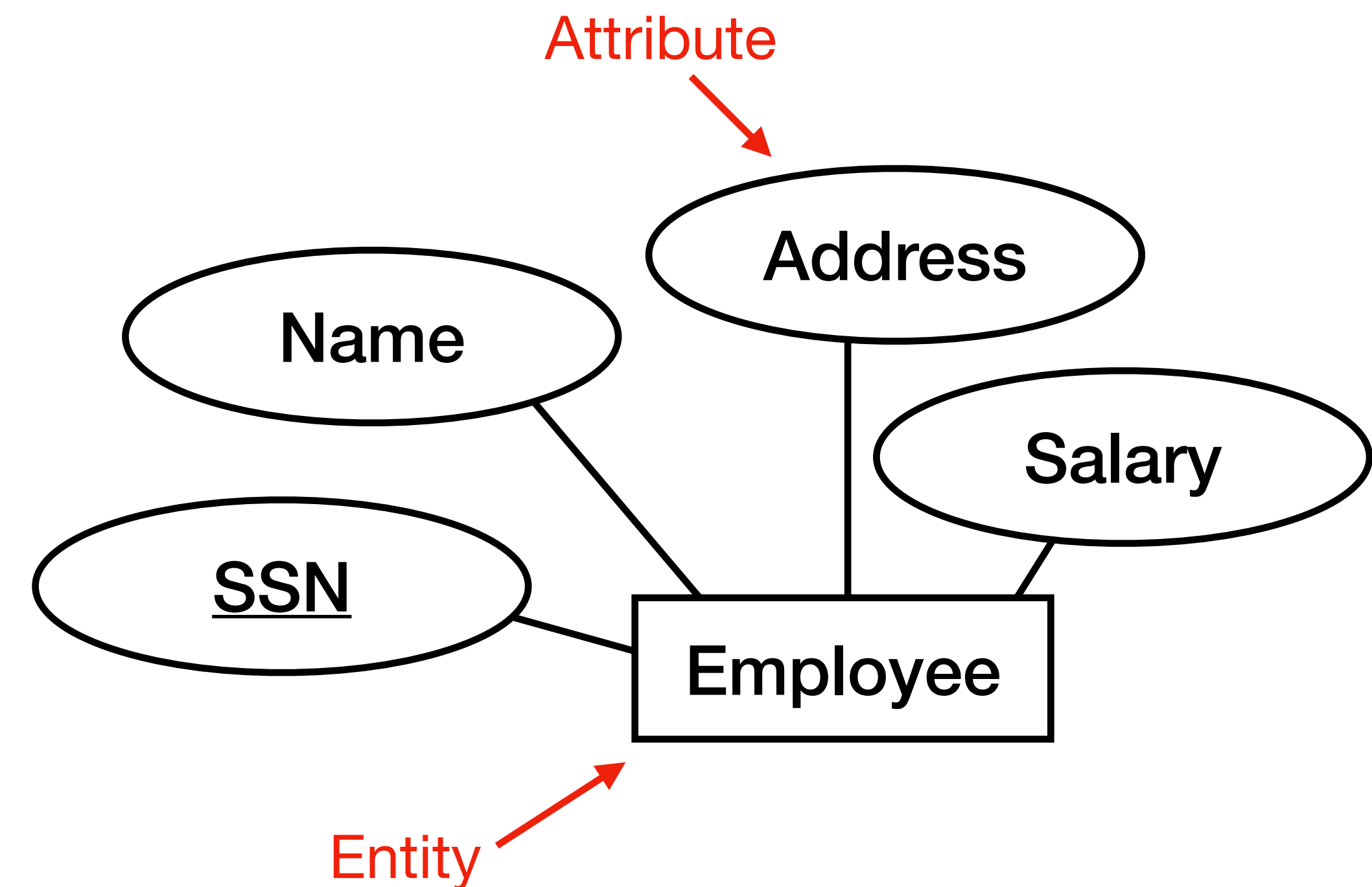
ER Model: Entities

- **Entity:** Real-world thing, distinguishable from other objects
 - Noun phrase (e.g., Bob Smith, Main Street Branch, Account 1234, etc.)
 - Described by a set of **attributes**



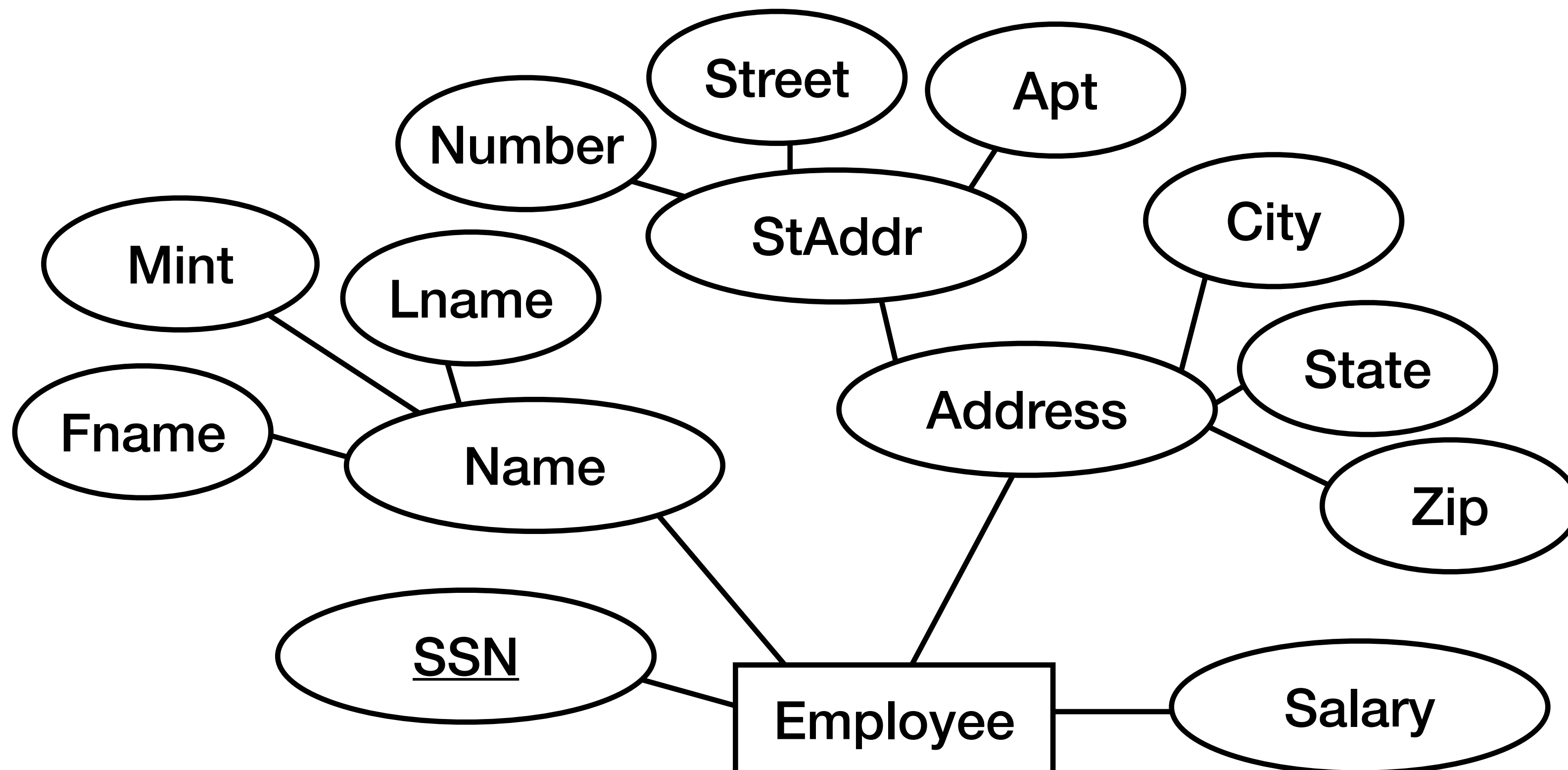
ER Model: Attributes

- **Attributes:** Properties that describe an entity. Each has a **domain**.
- Special attributes:
 - Composite attributes vs simple (atomic)
 - Multivalued attributes vs single-valued
 - Keys attributes
 - Derived attributes vs stored



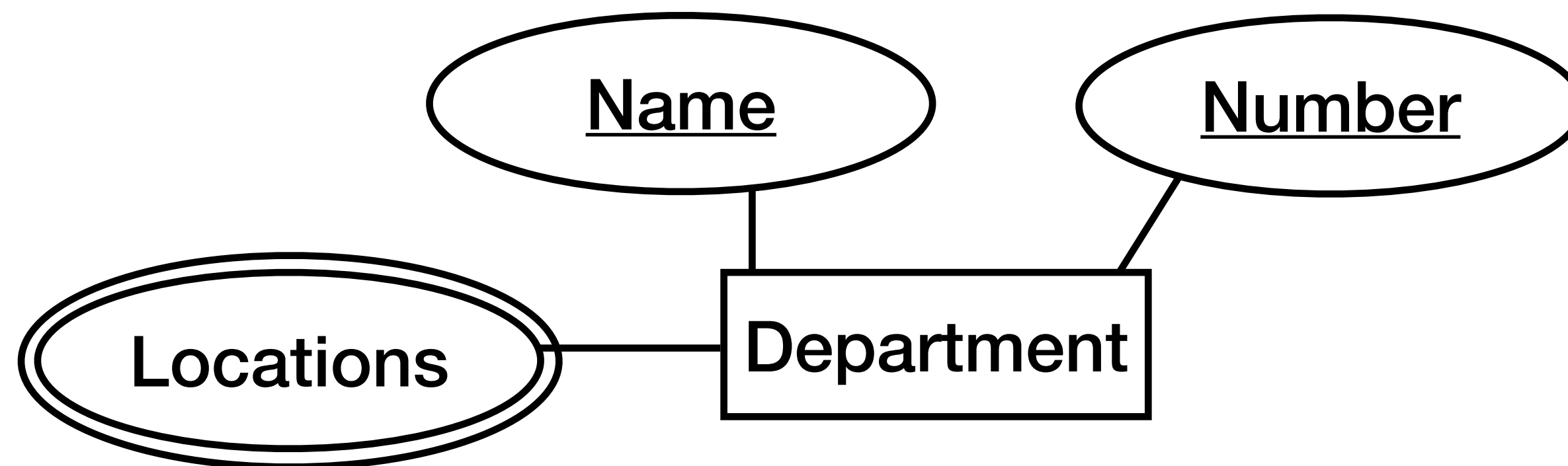
Composite Attributes

- If you need access to particular parts of an attribute, the attribute may be **composite** (i.e., having components)



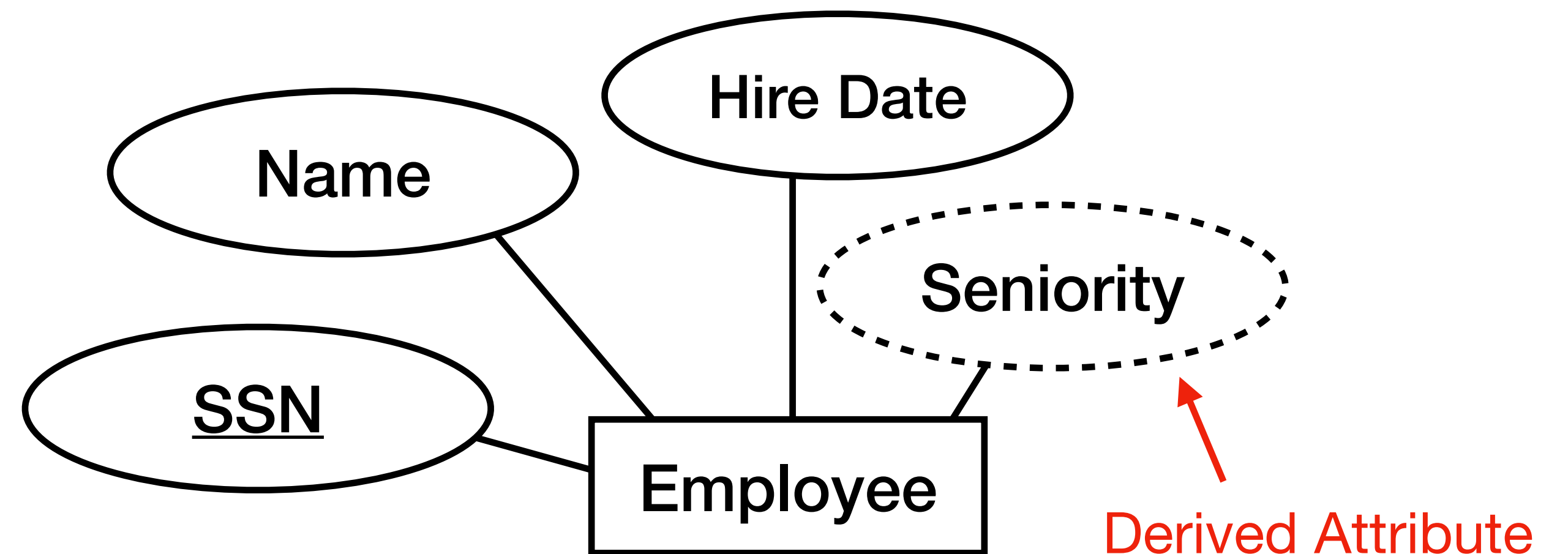
Multivalued Attributes

- If a given entity can have multiple values for a particular attribute, the attribute may be **multivalued**
- For example, if some departments are in multiple buildings



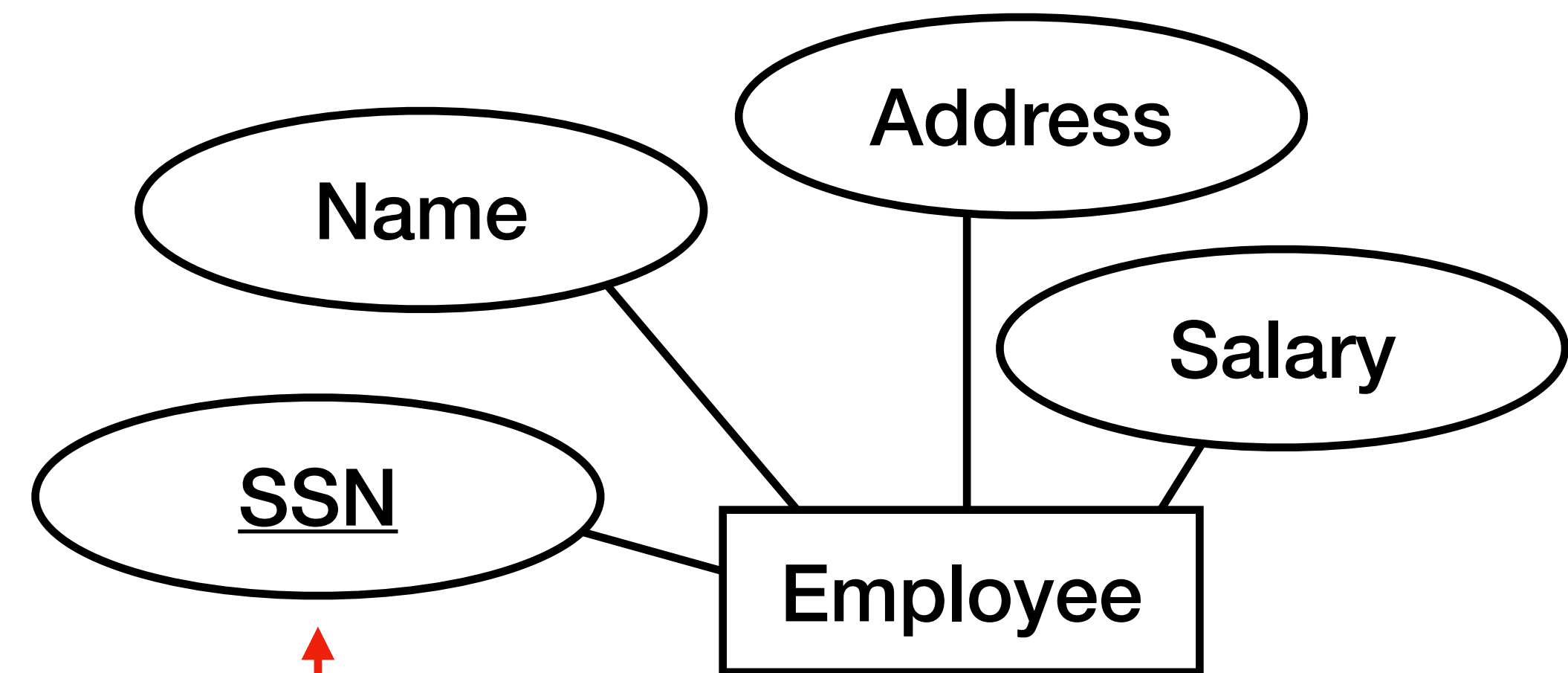
Derived Attributes

- Most attributes are explicitly recorded in the database
- **Derived attributes** can be calculated from other attributes
- Avoid redundancy to avoid inconsistencies!



Key Attributes

- **Key attributes:** unique constraint
- **Candidate key:** attribute(s) which *can* uniquely identify an entity
- **Primary key:** attribute(s) which **are** used to uniquely identify an entity
- Used to quickly locate an entity, and to store relationships

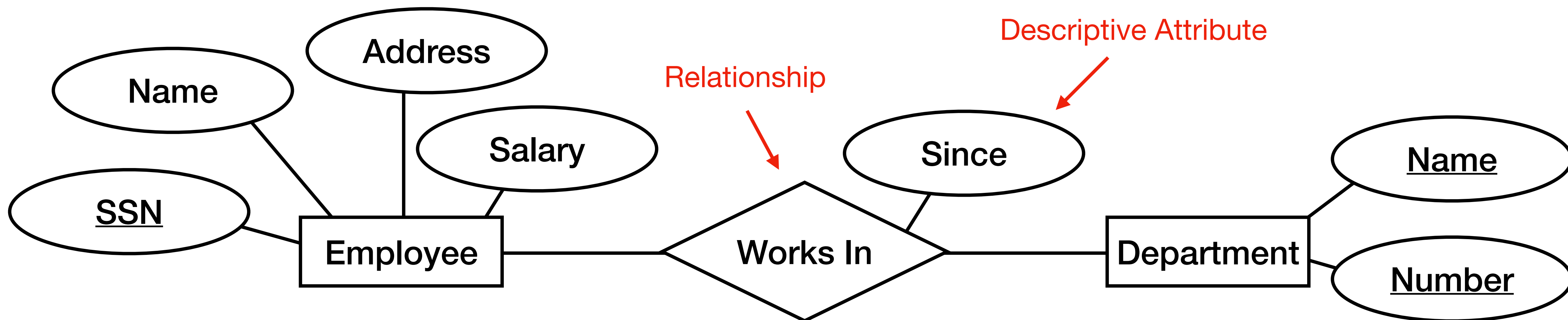


Unique, so a "key."

Also a candidate key, and *could* be chosen as a primary key.

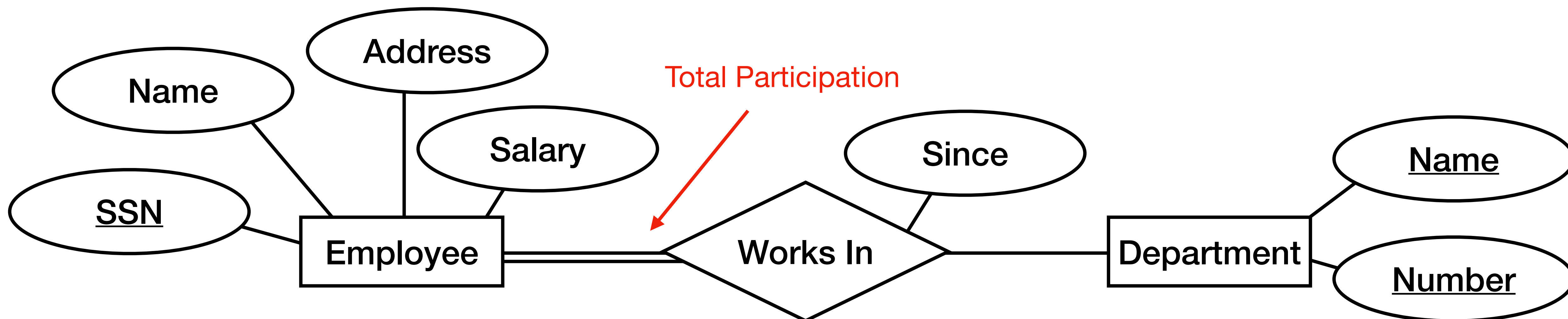
ER Model: Relationships

- Association among two or more entities (e.g., Jane Smith works in Pharmacy department)
- Verb phrases (e.g., works_in, votes_on, etc)
- Relationships can have their own attributes (descriptive attributes)



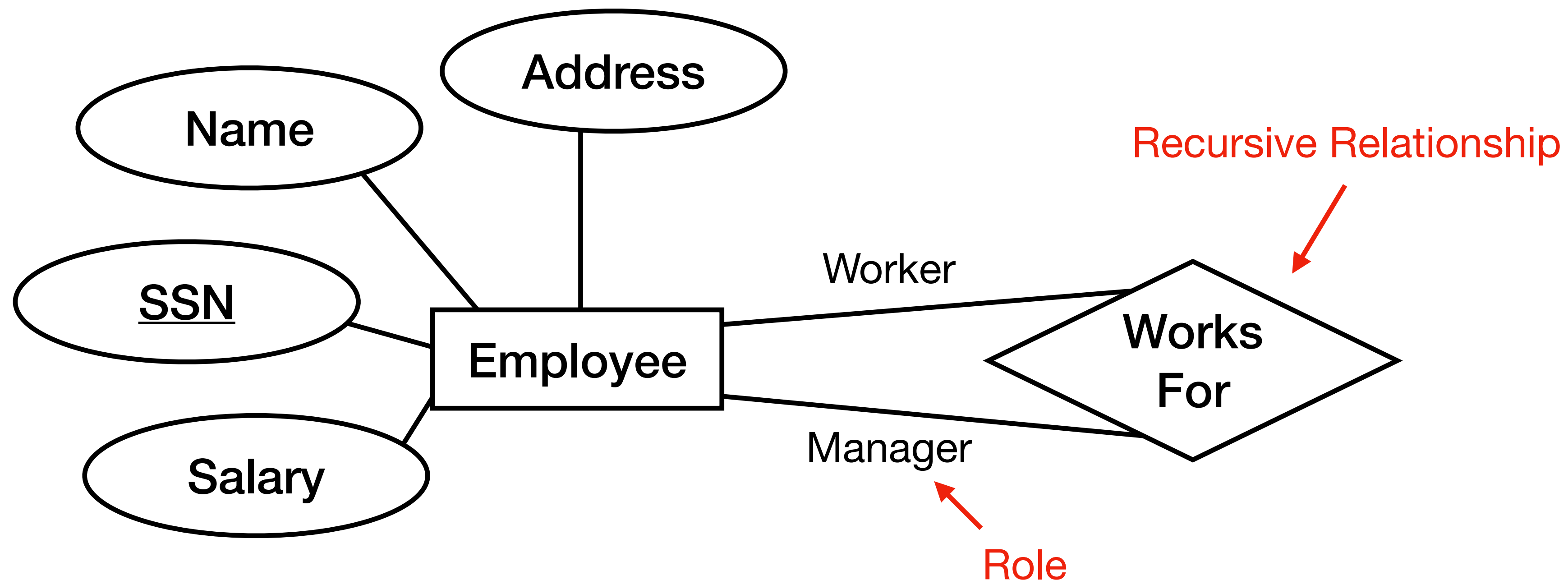
Total Participation

- All employees **must** work for a department
- What if a department is dissolved?







Recursive Relationships

- Association between an entity and itself
- Must be declared with **roles**



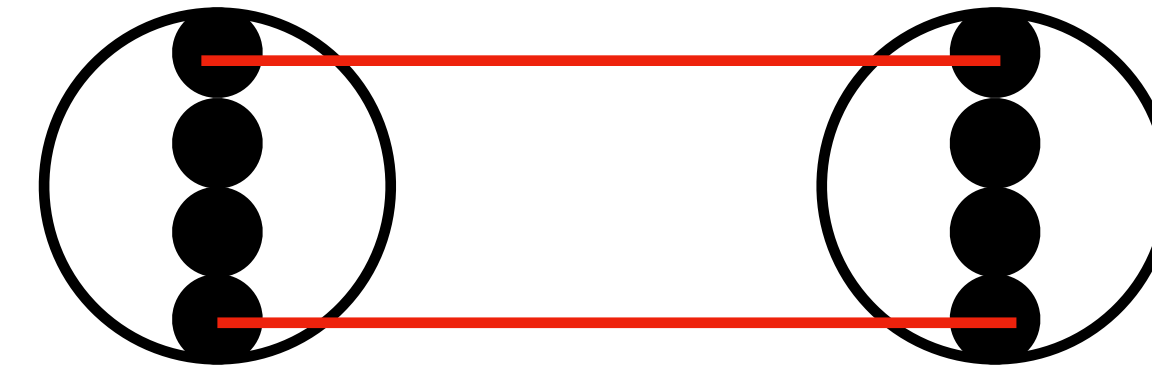
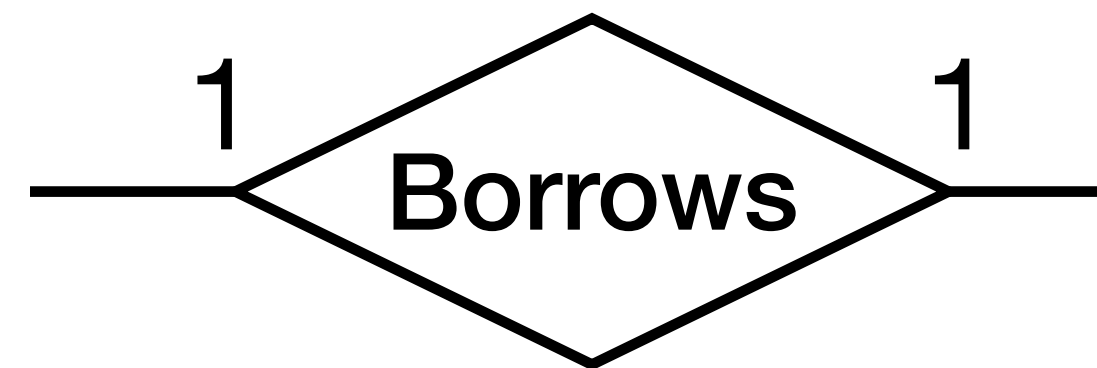
Relationship Cardinality

- How many relationships can an entity participate in?

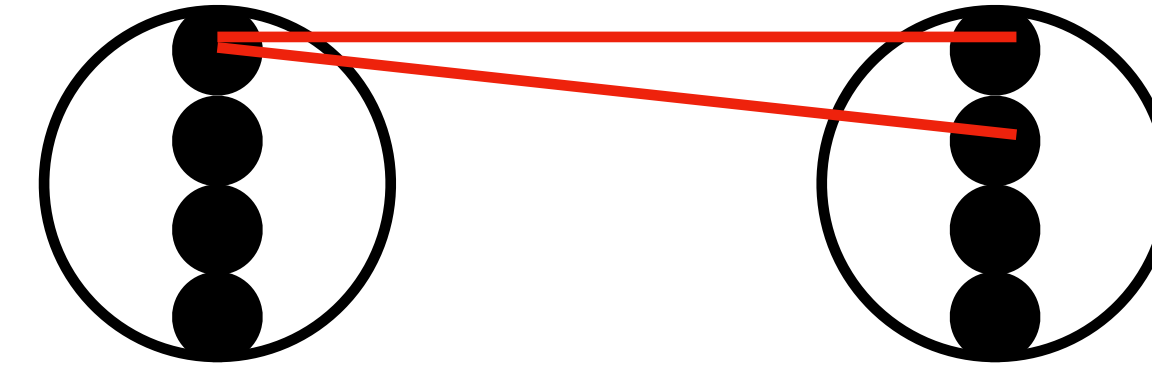
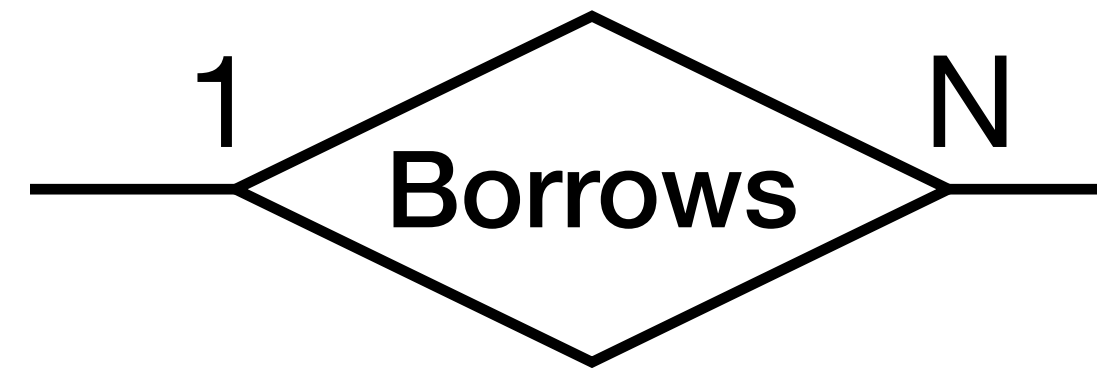
		Multiple Loans?	Joint Loans?
One-to-One (1:1)		No	No
One-to-Many (1:N)		Yes	No
Many-to-One (N:1)		No	Yes
Many-to-Many (N:M)		Yes	Yes

Relationship Cardinality

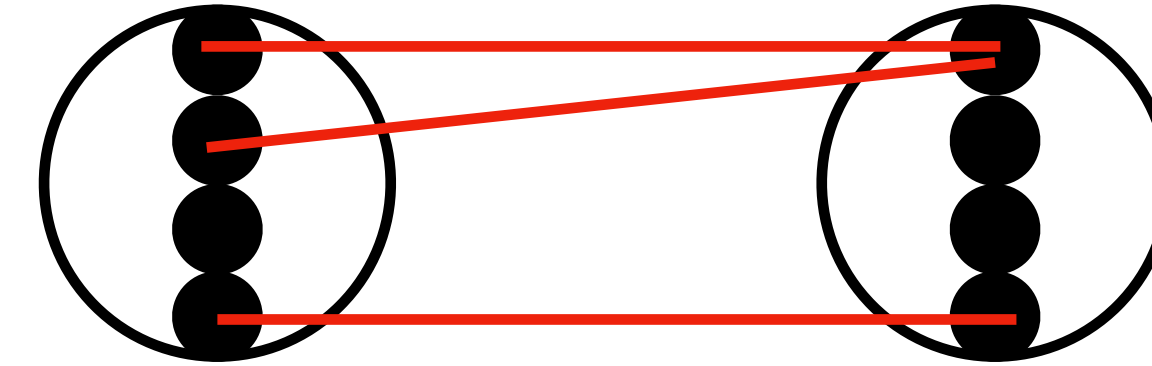
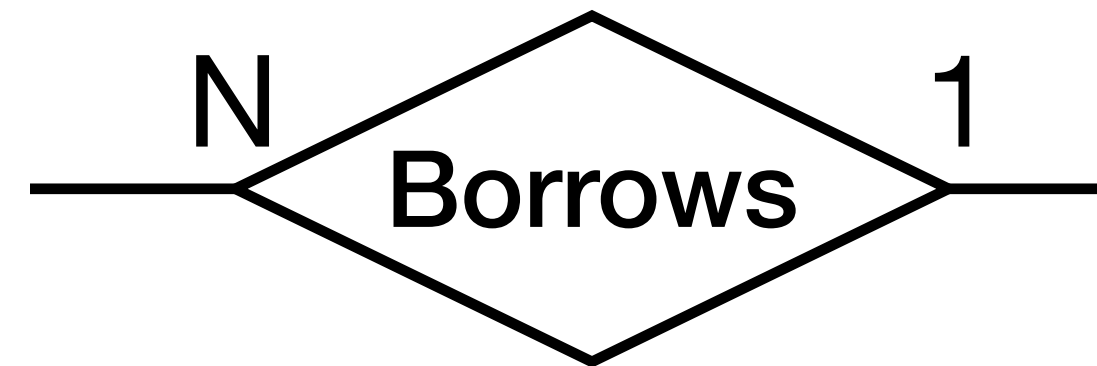
One-to-One (1:1)



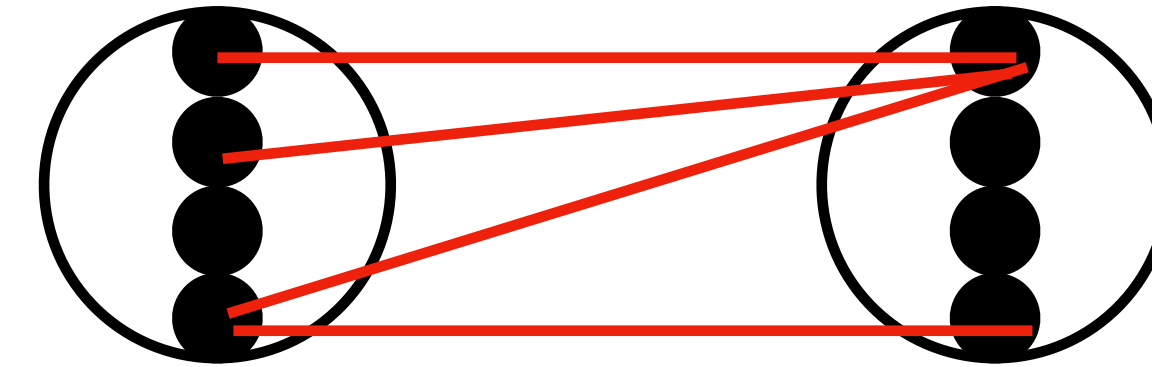
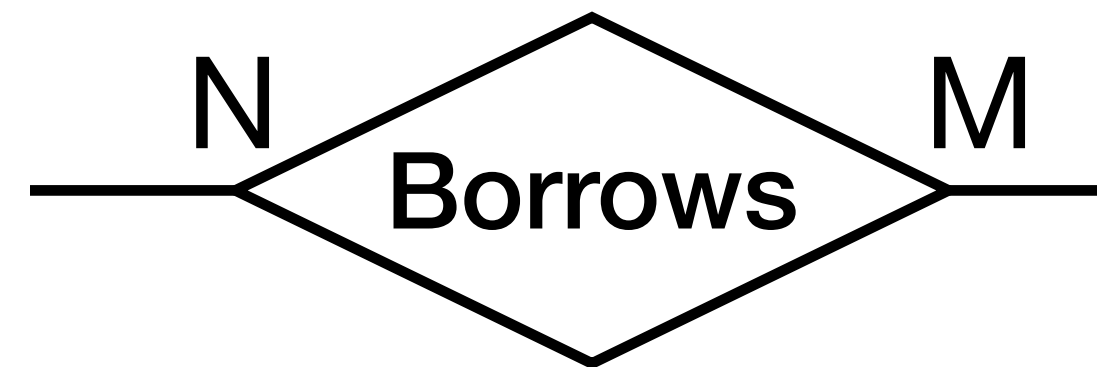
One-to-Many (1:N)



Many-to-One (N:1)

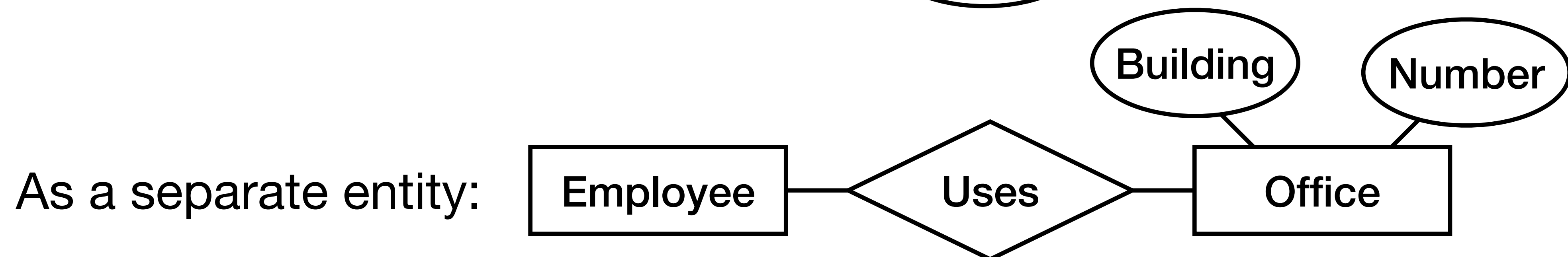
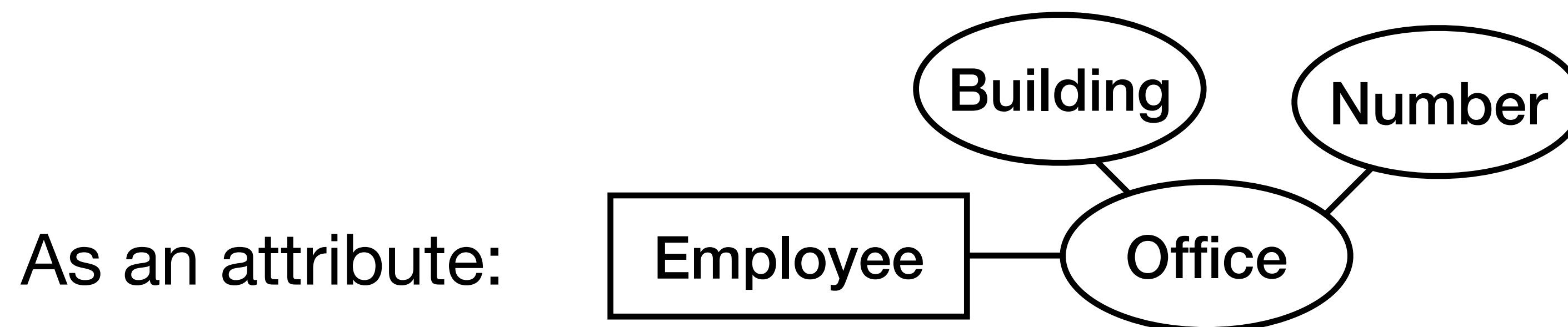


Many-to-Many (N:M)



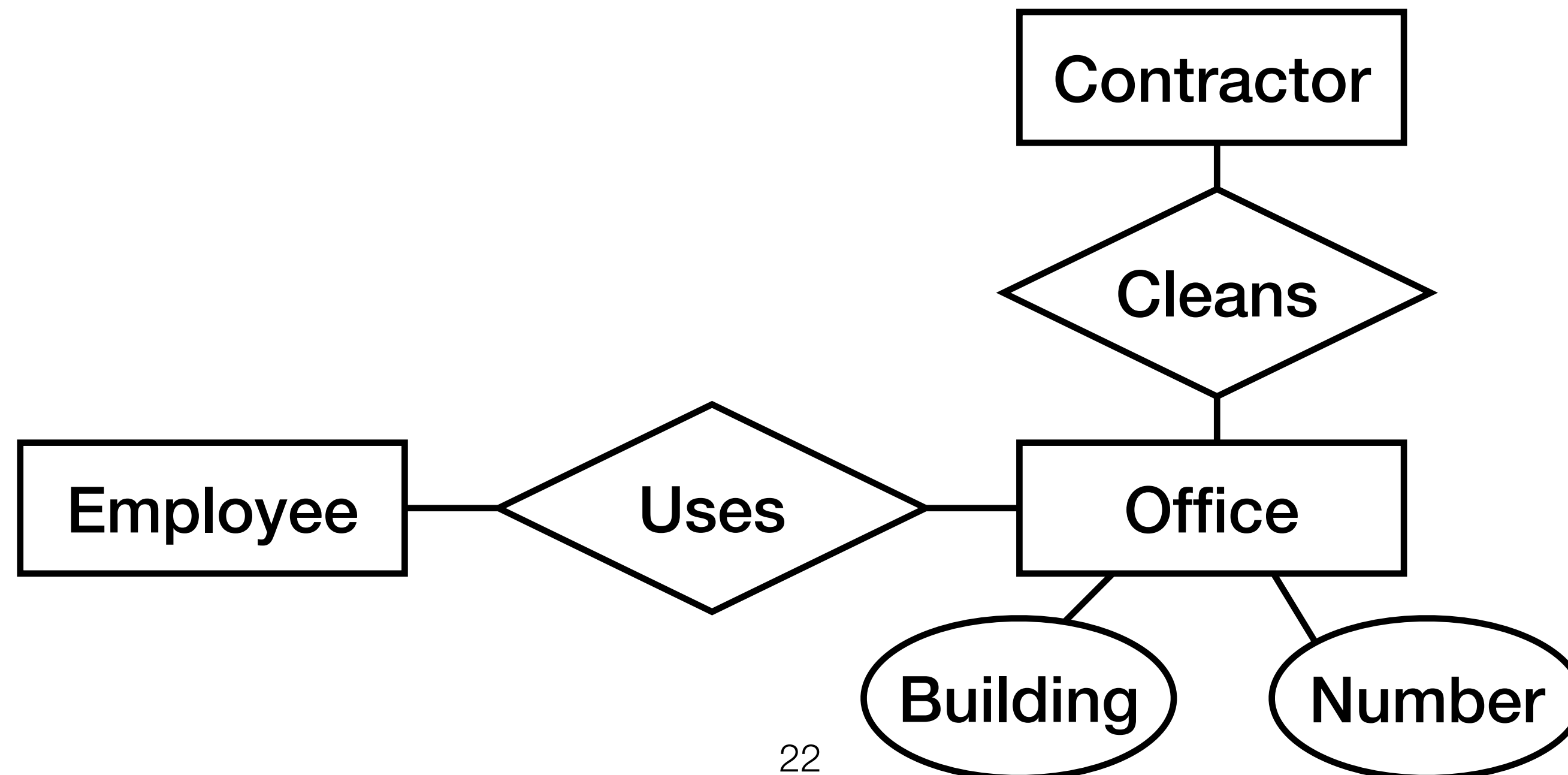
Entities vs Attributes

- How to model employee offices? As entities or attributes? Generally:
 - If 1:1, attributes work well. If 1:N, multivalued attributes work okay.
 - Otherwise, use a separate entity



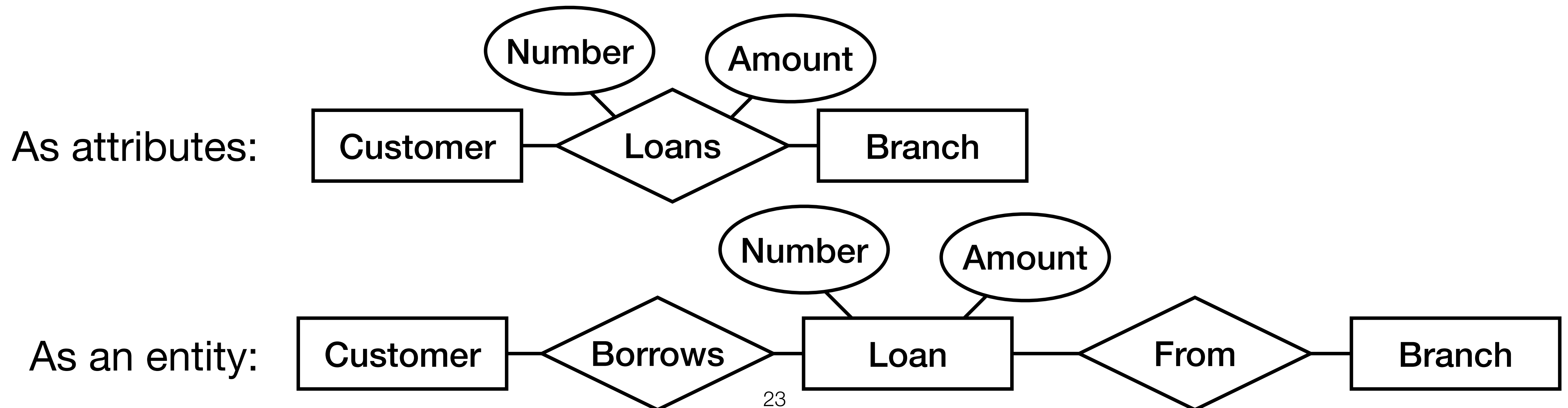
Entities vs Attributes

- If several entities have the same attribute, consider modeling the attribute as an entity
- Conversely, if an entity is only associated with one entity, consider modeling the entity as an attribute of the other



Entities vs Relationships

- How to model loans? As entities or relationships with descriptive attributes?
- Are joint loans allowed (more than one customer per loan)? Is loan a noun or a verb? **If yes to either:** model as an entity. Avoid redundancy!



Entity and Relationship Types and Sets

- An entity instance: the employee “Jane Smith” has SSN 111-22-3333
- **Entity type:** an Employee entity has name and SSN attributes
- **Entity set:** all employees in the database
- A relationship instance: “Jane Smith” works in “Accounting”
- **Relationship type:** “works in” is a N:1 relationship between Employee and Department
- **Relationship set:** all “works in” relationships in the database

Best Practices for Naming

- Choose meaningful names:
 - Nouns give rise to entity type names
 - Verbs indicate names of relationship types
- Name relationships so the ER diagram is readable from left to right and from top to bottom

ER Modeling Practice

- Ideas for real world things we can create ER models for?
- Websites from lab?

Recap: Where Do Databases Come From?

1. Requirements collection
2. Conceptual design (ER model) ← Our focus today
3. Logical design (first with ER and relational models, then with SQL DDL)
4. Physical design (DMBS interprets the SQL DDL, storing the DB on disk)

Often an iterative process (e.g., when ambiguity is discovered, you need to meet with stakeholders again)

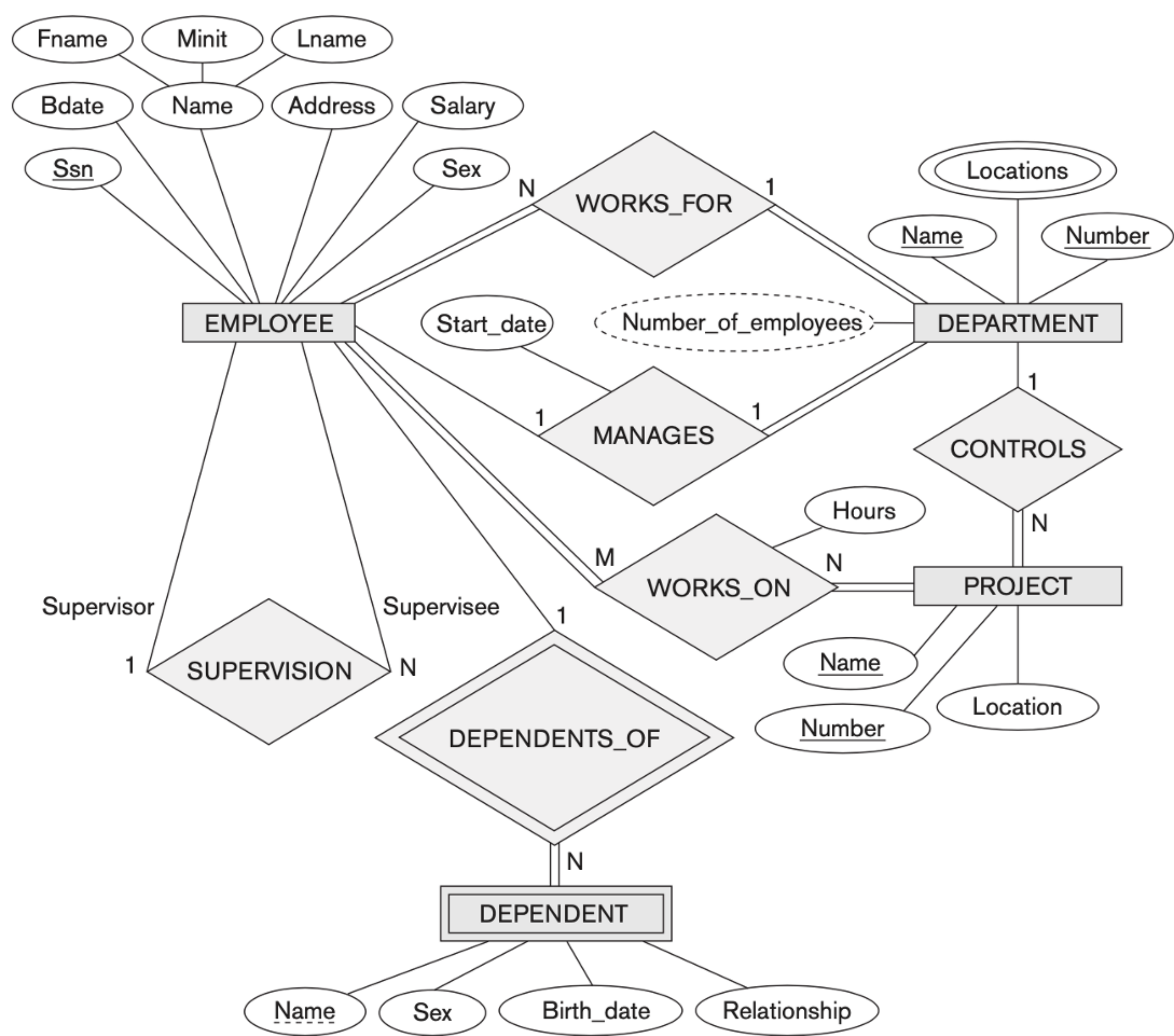


Figure 3.2

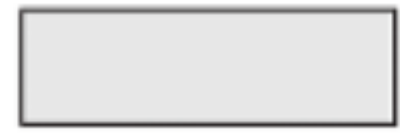
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter and is summarized in Figure 3.14.

Symbol

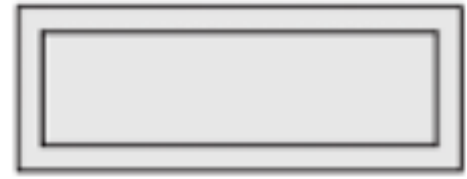
Meaning

Figure 3.14

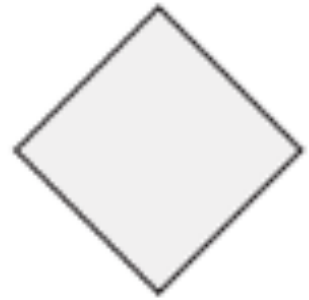
Summary of the notation for ER diagrams.



Entity



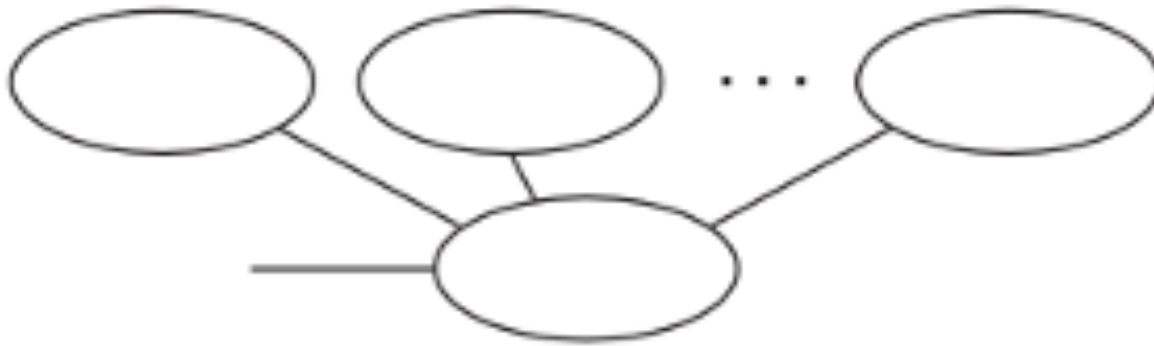
Weak Entity



Relationship



Identifying Relationship



Composite Attribute



Derived Attribute



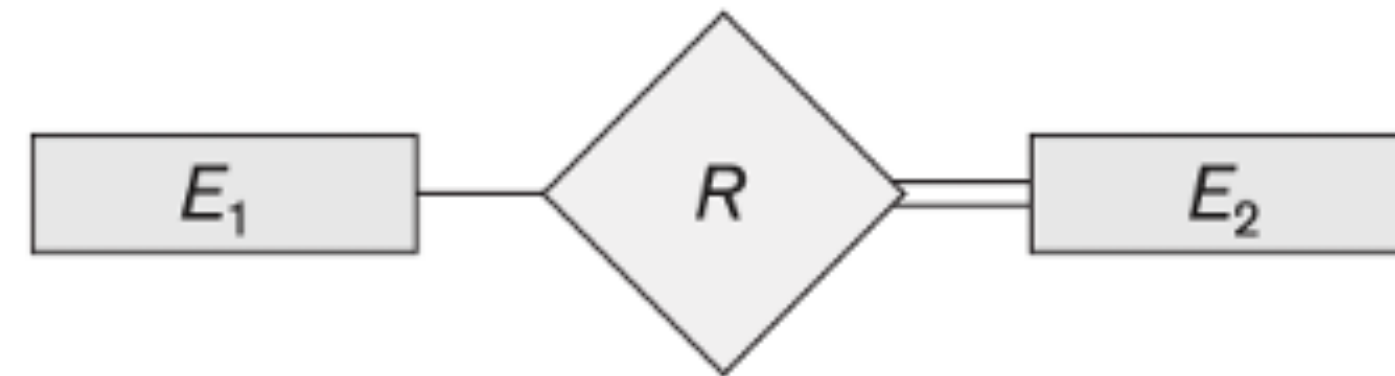
Attribute



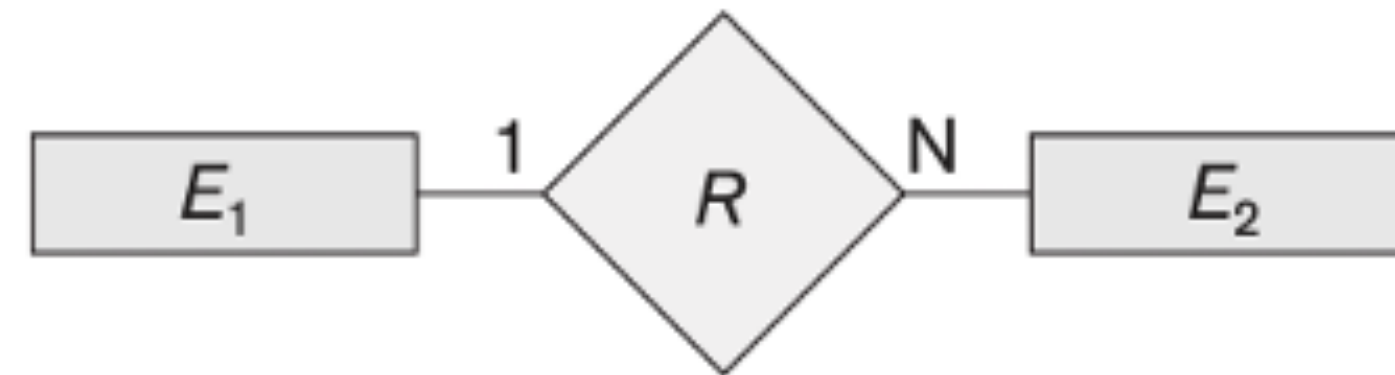
Key Attribute



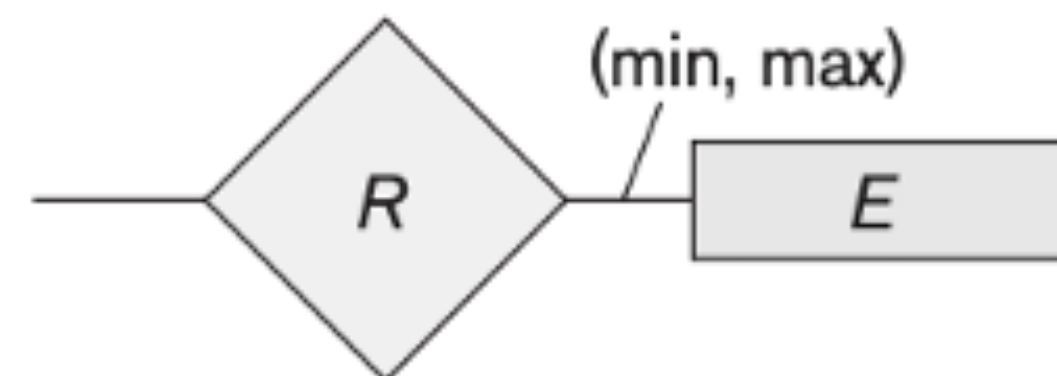
Multivalued Attribute



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

Advanced ER Models

CSCI 220: Database Management and Systems Design

Practice Quiz

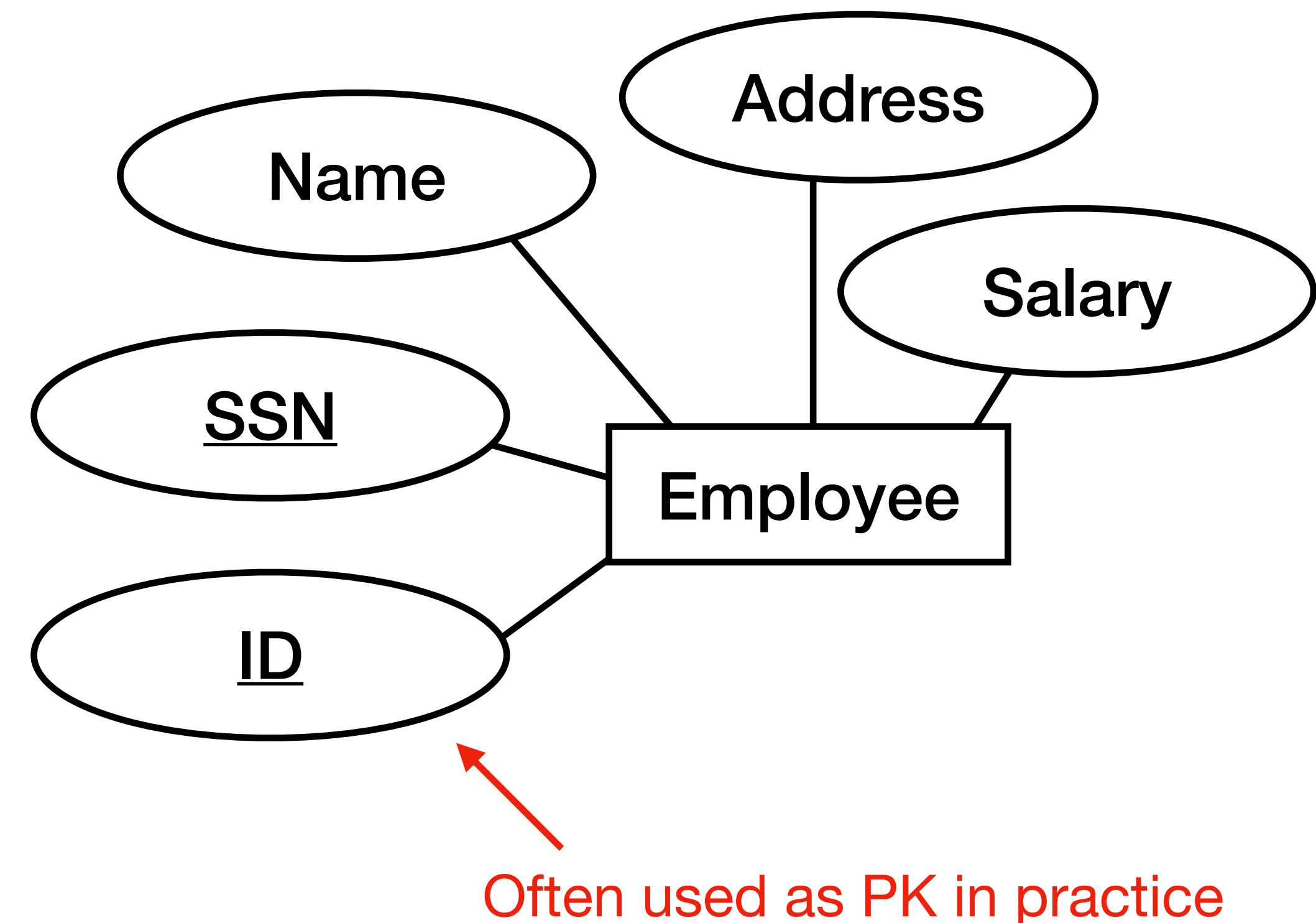
- Draw a simple ER diagram representing a pizza ordering system. Capture all the following information:
 - Customers can place orders for pizzas. Customers must have a phone number, which is used to identify them. Customers also have a name and address.
 - An order contains one or more pizzas. It should be possible to determine the order total (e.g., \$5.00).
 - A pizza has a size and can have any number of toppings (e.g., cheese, pepperoni, tomato sauce, etc.). Each size and topping has a price.

Today you will learn:

- How to uniquely identify entities
- If time: how to avoid getting sued by government regulators

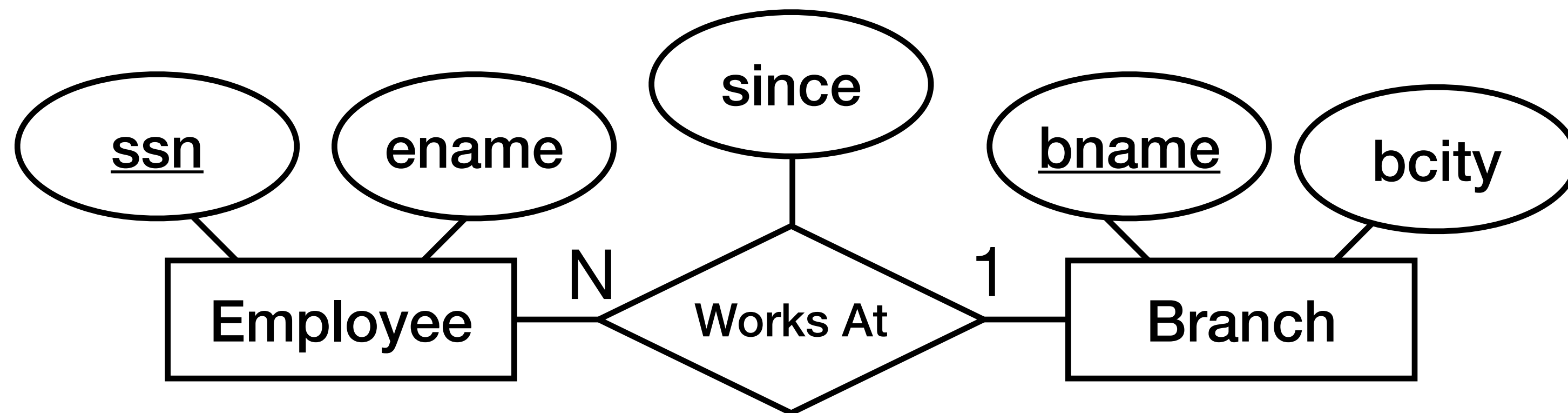
Key, Superkey, Candidate Key, Primary Key

- **Key attributes:** unique constraint
- **Superkey:** any attribute set that distinguishes identities. Superkeys are possible even if no single attribute is unique.
 - e.g., {ssn}, {ssn, name, address, ...}, ...
- **Candidate key:** “minimal superkey” (can’t remove unnecessary attributes)
 - e.g., {ssn}, {name, address}
- **Primary key:** the candidate key chosen to uniquely identifying entities



Keys and Relationships

- Keys help describe relationships
- The attributes {ssn, bname, since} describe the relationship completely
- What attributes are the candidate keys of the relationship?
 - Since an employee can only work at one branch, {ssn} suffices

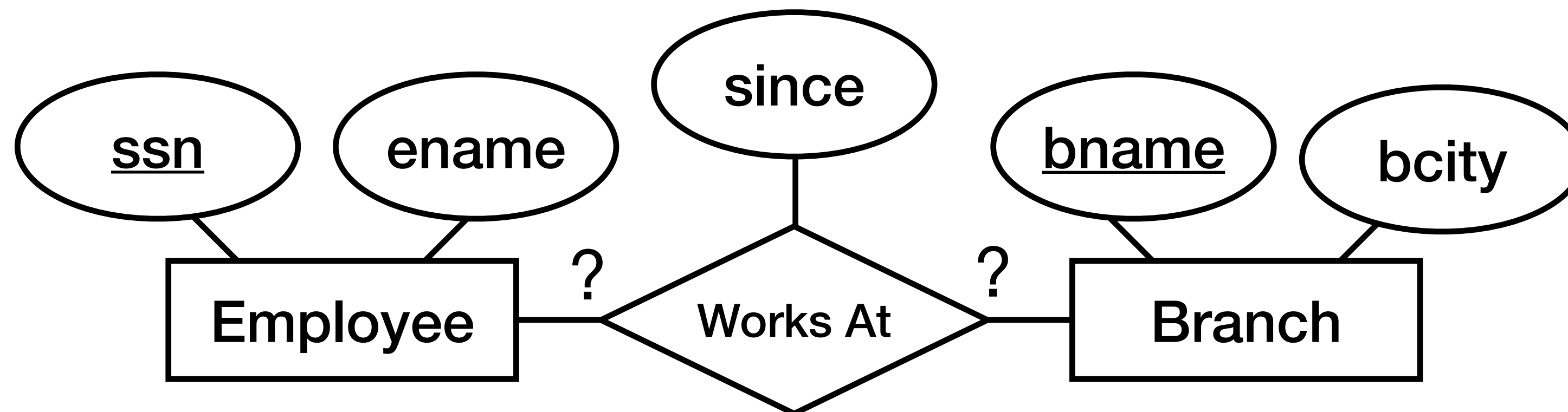


Keys and Relationships

- A relationship's candidate keys depend on its cardinality:

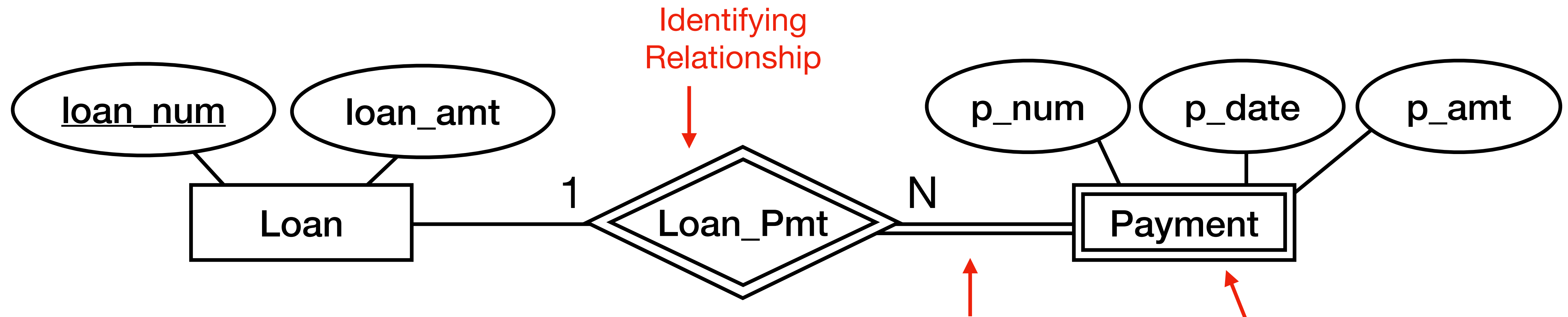
Cardinality	Candidate Keys
n:1	{ssn}
1:n	{bname}
n:m	{ssn, bname}
1:1	{ssn}, {bname}

← Assuming ≤ 1 relationship for each employee-branch pair



Weak Entities

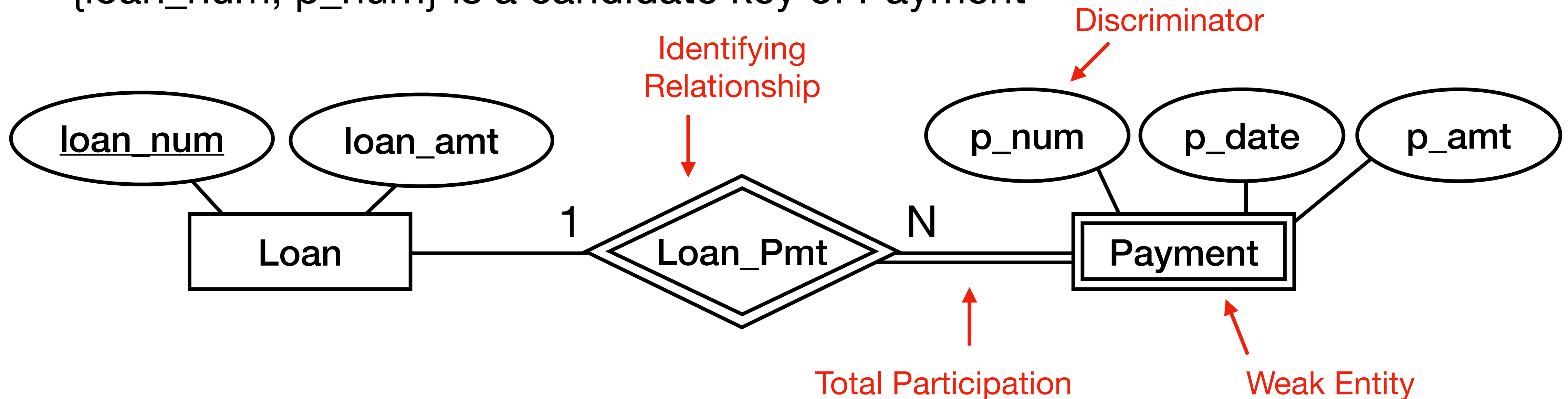
- Entities without keys are called **weak entities**
- Weak entities have a **total participation** constraint with an **identifying relationship**



**Although abbreviations like Loan_Pmt are used in the slides, avoid unnecessary abbreviations in a real DB*

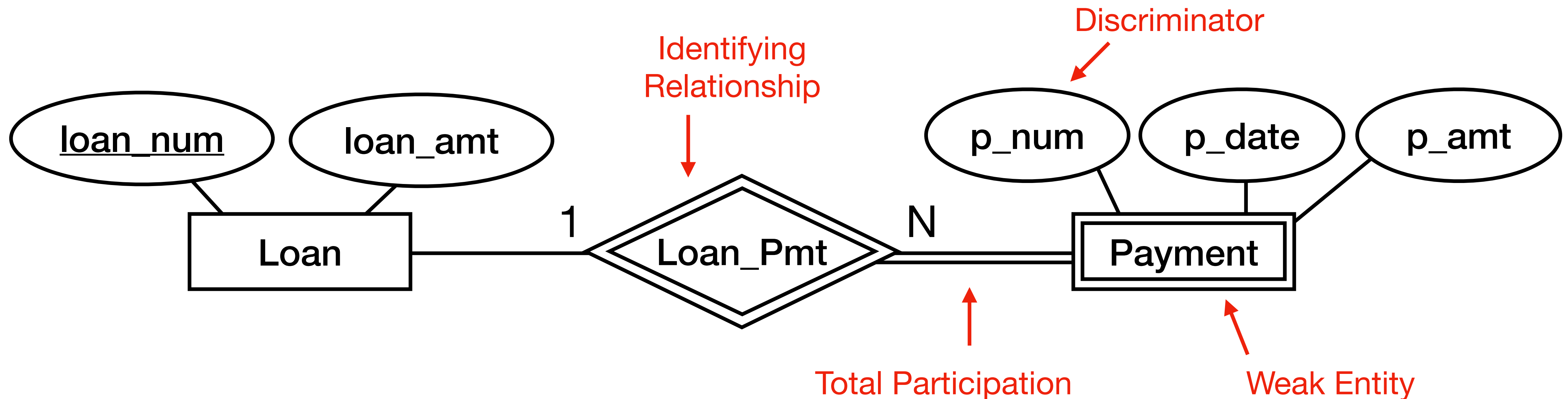
Weak Entities

- For example, Payment has no superkeys: p_num starts at 1 for each Loan
- Instead, Payment's p_num is a **discriminator**. For a given Loan, p_num uniquely identifies Payments.
- {loan_num, p_num} is a candidate key of Payment



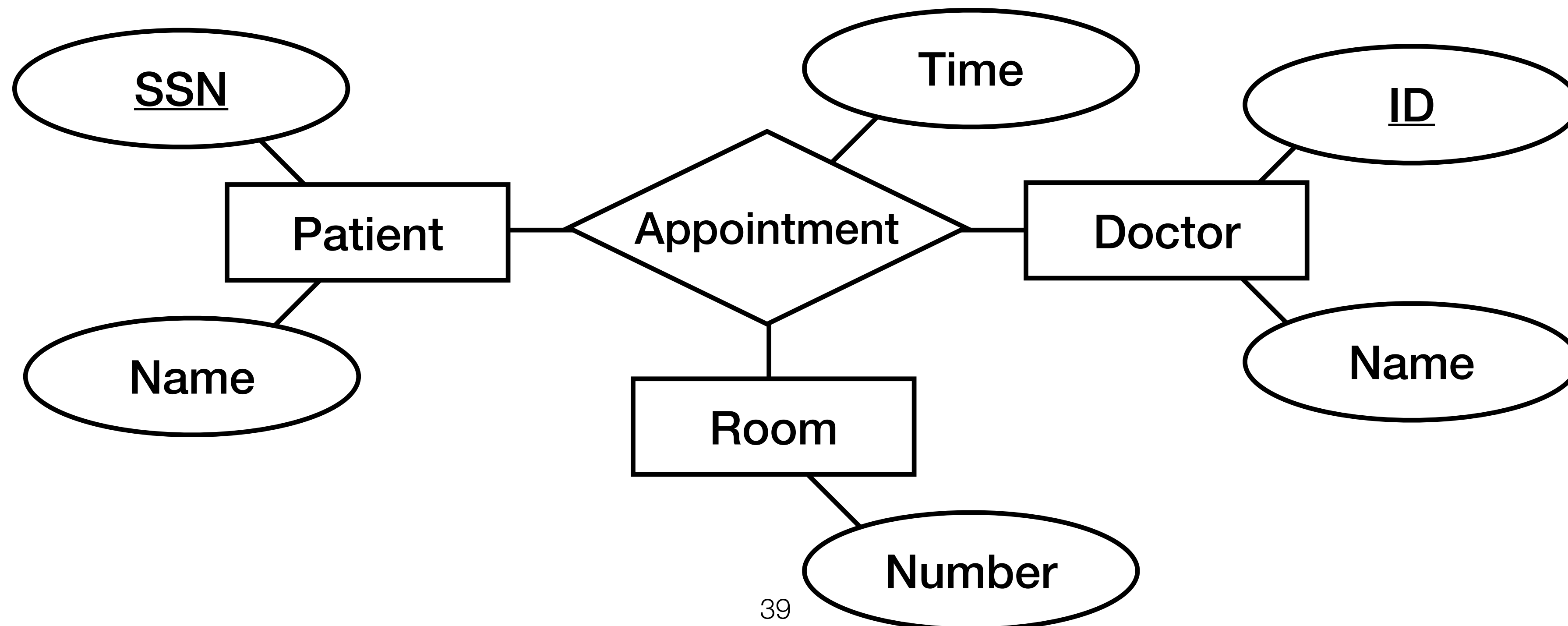
Weak Entities

- Loan is the **identifying owner** in Loan_Pmt
- Payment is the **weak entity set** in Loan_Pmt
- Payment is **existence dependent** on Loan



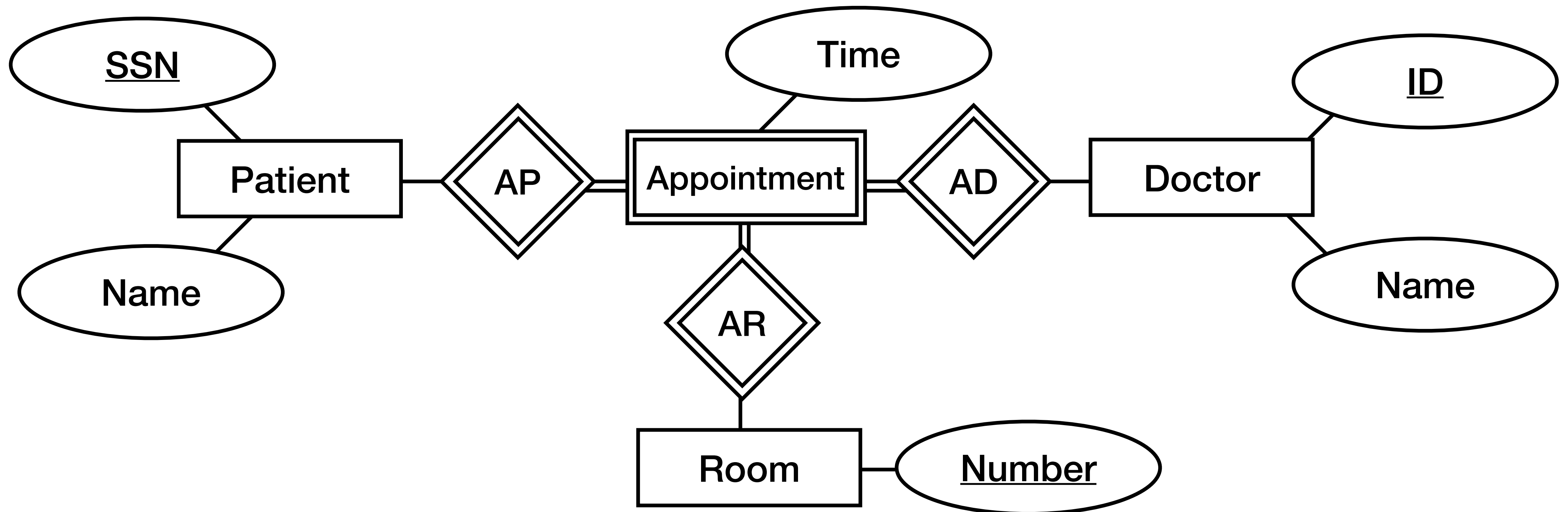
Ternary Relationships

- How to model “a doctor sees a patient in a particular room?” For example: “Dr. Nick sees Homer Simpson in Room 311.”
- Could be viewed as a relationship between three entities!

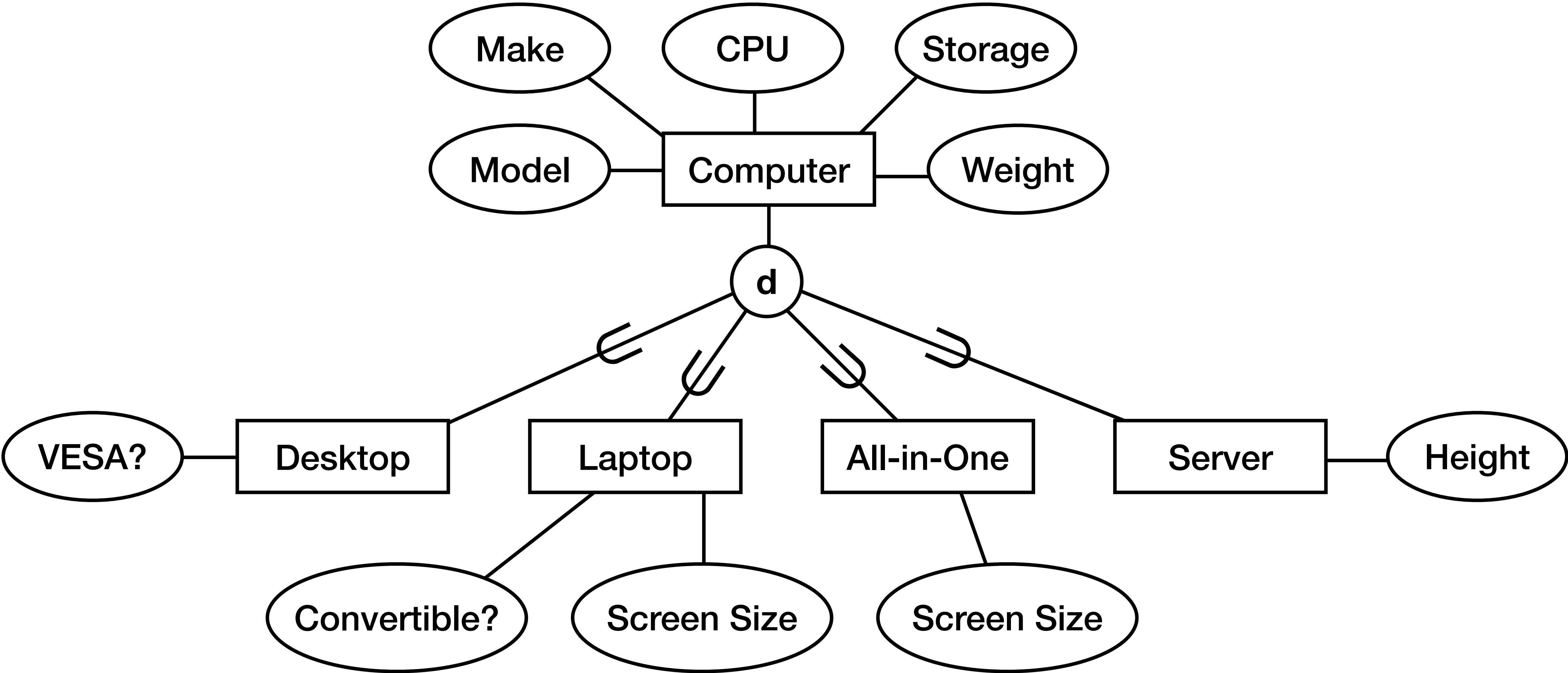


Ternary Relationships

- Alternatively, create an Appointment weak entity



Subclasses



Other Notations: UML Class Diagrams

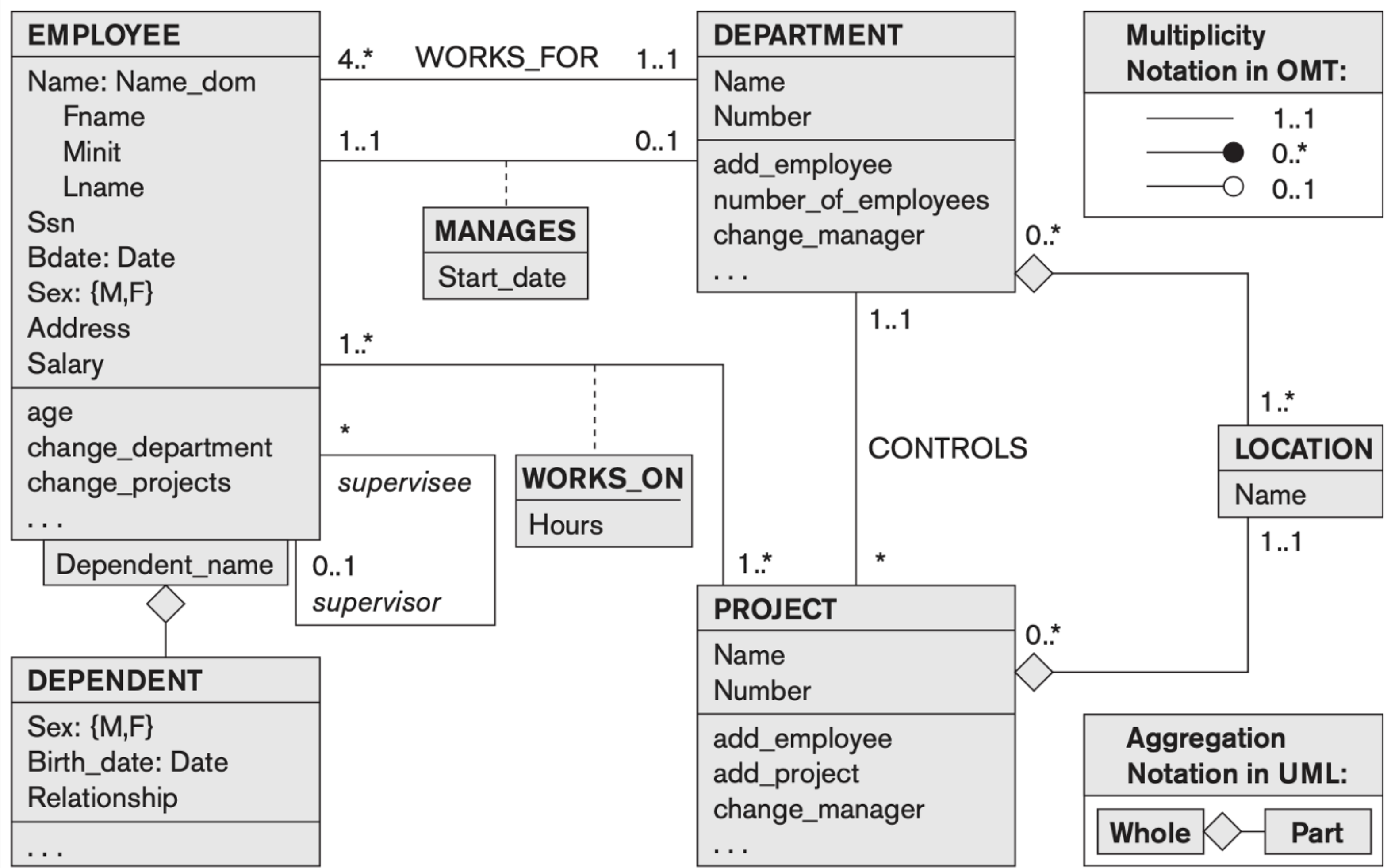


Figure 3.16
The COMPANY conceptual schema in UML class diagram notation.

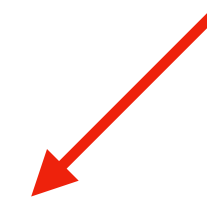
ER Modeling Practice

- Ideas for real world things we can create ER models for?
- We'll focus on keys, and may identify weak entities

Recap: Where Do Databases Come From?

1. Requirements collection
2. Conceptual design (ER model)
3. Logical design (first with ER and relational models, then with SQL DDL)
4. Physical design (DMBS interprets the SQL DDL, storing the DB on disk)

Next class



Often an iterative process (e.g., when ambiguity is discovered, you need to meet with stakeholders again)