CS 6016

Database Systems

Entity Relationship Model

Database Design Steps

This Week

- Database design: **E**ntity **R**elationship (**ER**) Model
- ER to Schemas

Database Design Steps

1. Requirements Analysis
    - What does user need? What must it do?

Database Design Steps

1. Requirements Analysis
    - What does user need? What must it do?

2. Conceptual Design
    - High level formal description

Database Design Steps

1. Requirements Analysis
    - What does user need? What must it do?

2. Conceptual Design
    - High level formal description

3. Schema Refinement
    - Consistency and “normalization”

Database Design Steps

1. Requirements Analysis
   - What does user need? What must it do?

2. Conceptual Design
   - High level formal description

3. Schema Refinement
   - Consistency and “normalization”

4. Physical Design
   - Indexes, disk layout

Database Design Steps

1. Requirements Analysis
   - What does user need? What must it do?

2. Conceptual Design
   - High level formal description

3. Schema Refinement
   - Consistency and “normalization”

4. Physical Design
   - Indexes, disk layout

5. Security Design
   - Who accesses it, and how?

Database Design Steps

1. Requirements Analysis
    - What does user need? What must it do?

2. Conceptual Design
    - High level formal description

•Accomplished using “Entity-Relationship” (ER) model

Problem → Solution

- Problem specification → C++ → assembly

Problem → Solution

- Problem specification → **C++** → assembly
- Problem specification → **ER Model** → Schema

Problem → Solution

- Problem specification → **C++** → assembly
- Problem specification → **ER Model** → Schema
- In both of these cases, you *could* skip the middle step

Problem → Solution

- Problem specification → **C++** → assembly
- Problem specification → **ER Model** → Schema
- In both of these cases, you *could* skip the middle step
- In both of these cases, that would be a bad idea

Problem → Solution

- Problem specification → **C++** → assembly
- Problem specification → **ER Model** → Schema
- In both of these cases, you *could* skip the middle step
- In both of these cases, that would be a bad idea
- In both of these cases, there is a mechanical (algorithmic) translation to the final result

ER Model

- What are the **entities**, and their **relationships**?

ER Model

- What are the **entities**, and their **relationships**?

- For remainder of today, **forget about tables**!


[shouting] And I don't want any questions
about the tables!

Entity

- A real-world object, distinguishable from other entities
  - `{u0123456, "Daniel"}`
- An entity is described by a set of **attributes**
  - `(uID string, name string)`

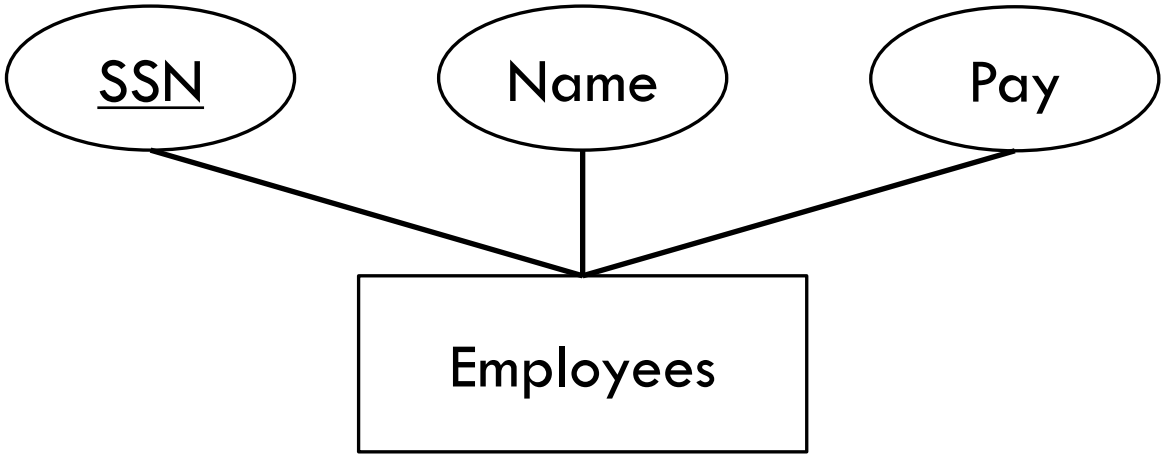
Entity Set

- A collection of entities of the same type, e.g.
  - All students
  - All buildings
  - All people

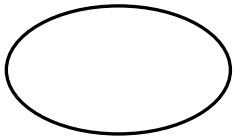
- All entities in the set have the same types of attributes

- An entity set has a key

ER Diagram

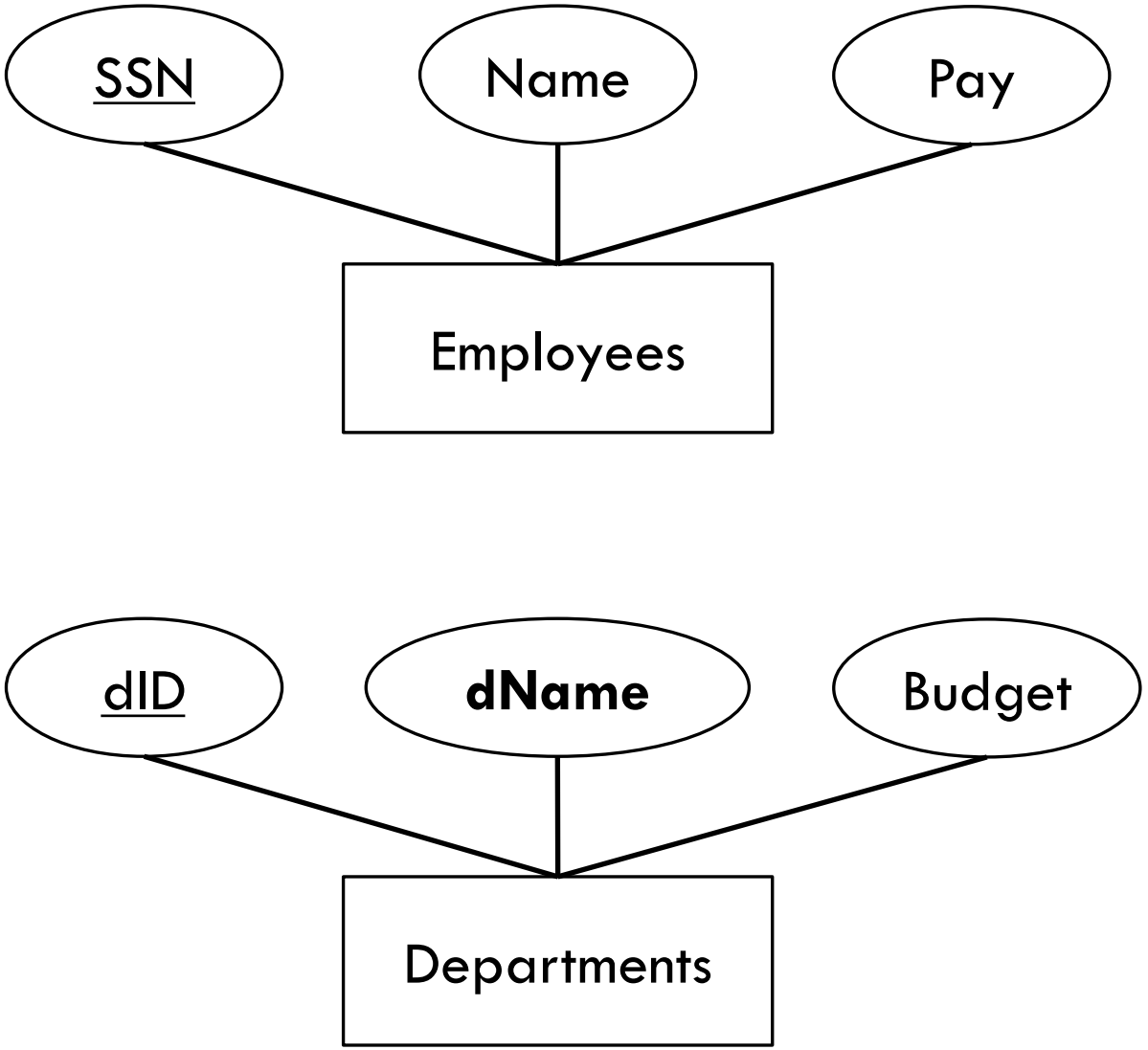


 = entity set

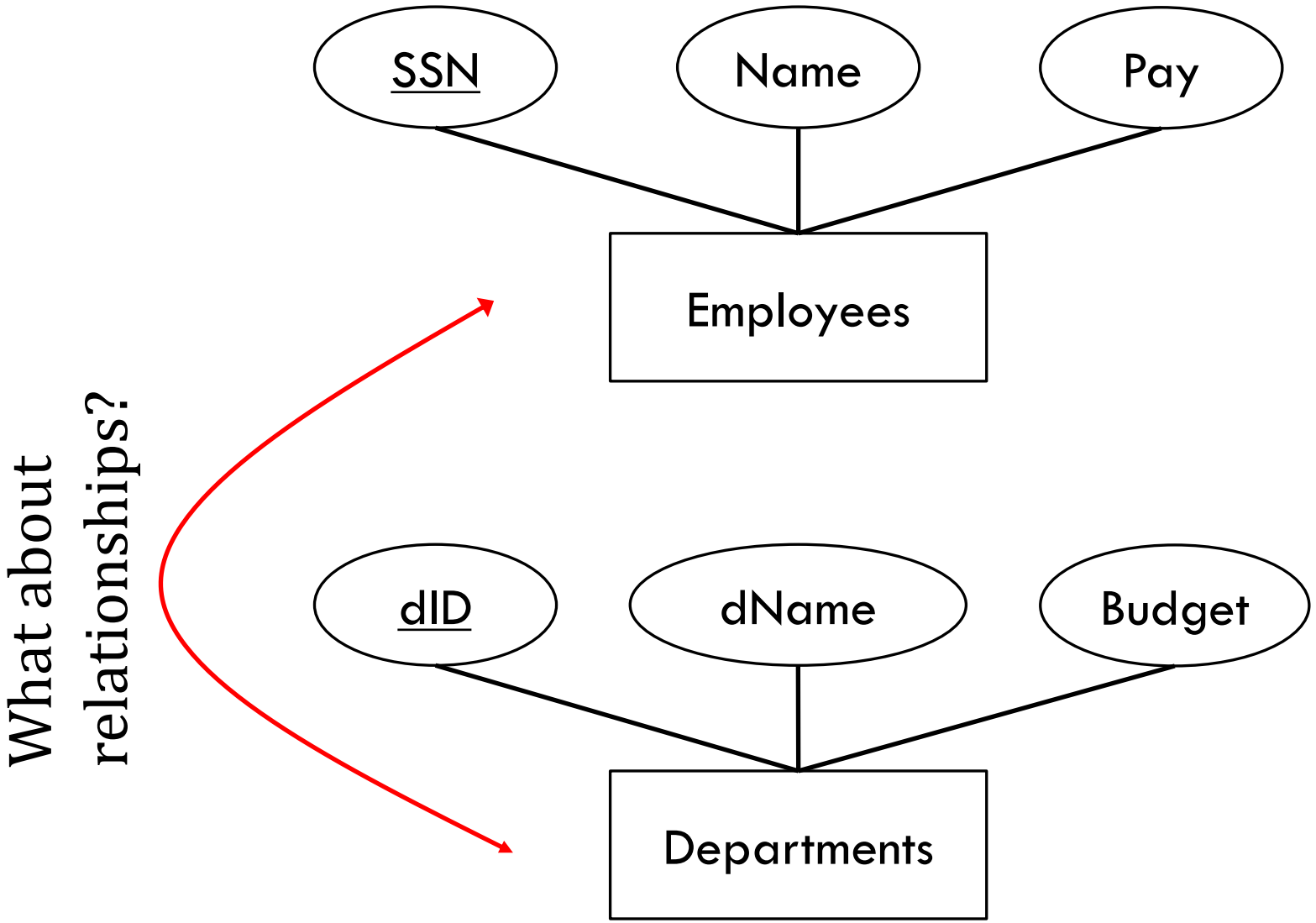
 = attribute

underline = key

ER Diagram

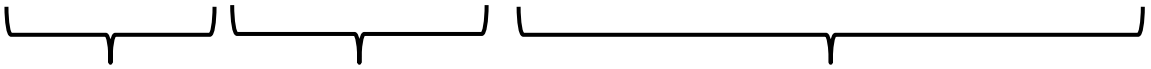


ER Diagram


SSN
Name
Pay
Employees
What about relationships?
dID
dName
Budget
Departments

Relationship

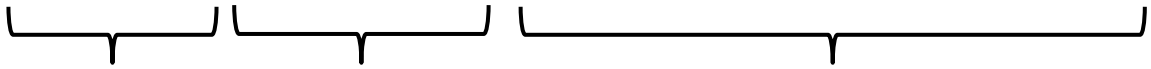
- Relationship between 2 or more entities:
  - "Daniel works in School of Computing"



entity relationship entity

Relationship

- Relationship between 2 or more entities:
  - "Daniel works in School of Computing"



entity relationship entity

- Relationship Set:
  - Set of relationships between entities of same type
  - e.g. **works in** relates **Employees** to **Departments**

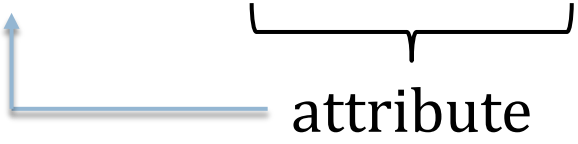
Relationship Attributes

- Relationships can have attributes as well:
  - Daniel works in SoC since 2010

attribute

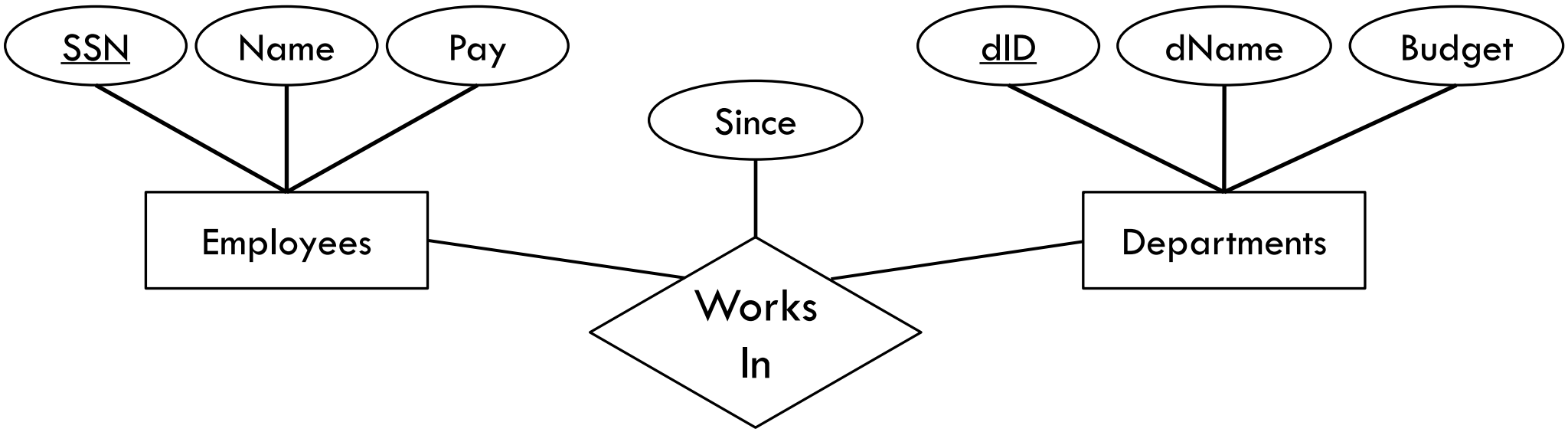
Relationship Attributes

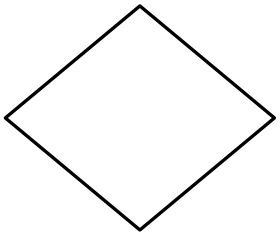
- Relationships can have attributes as well:
  - Danny works in SoC since 2010



- Starting date does not belong to Danny or SoC
  - It belongs to the relationship

ER Diagram

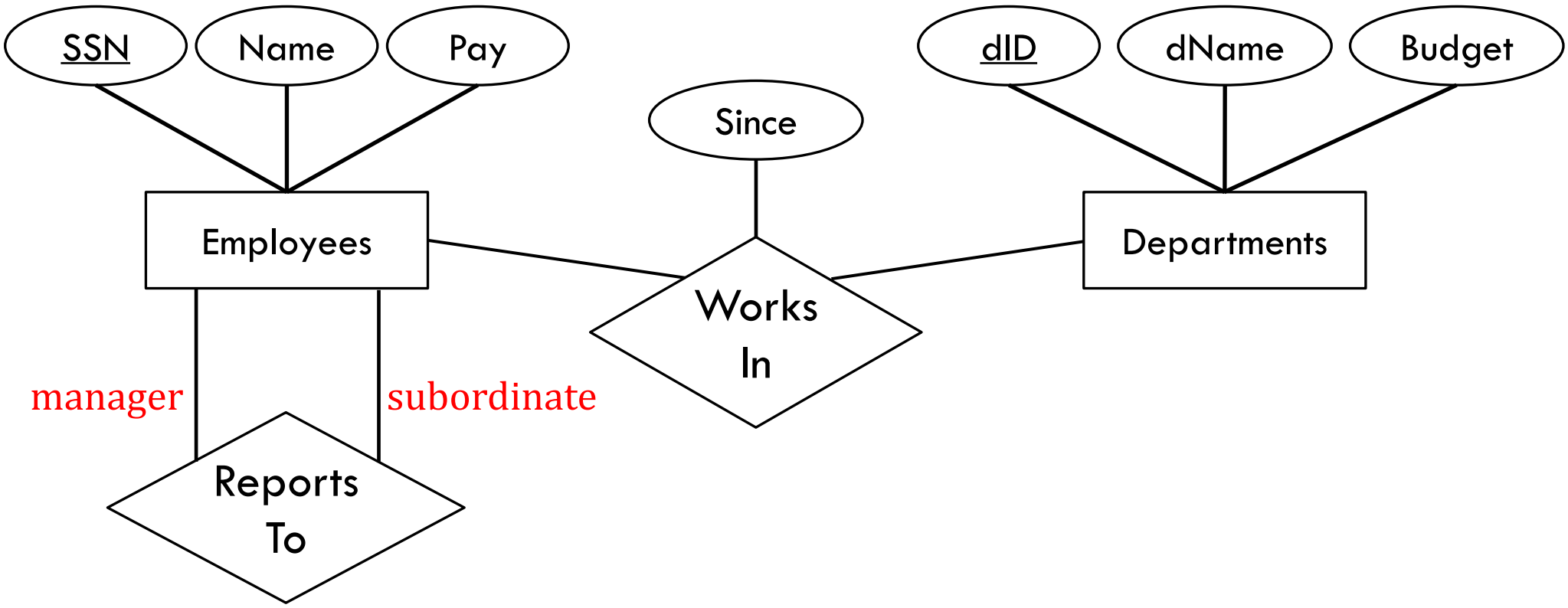




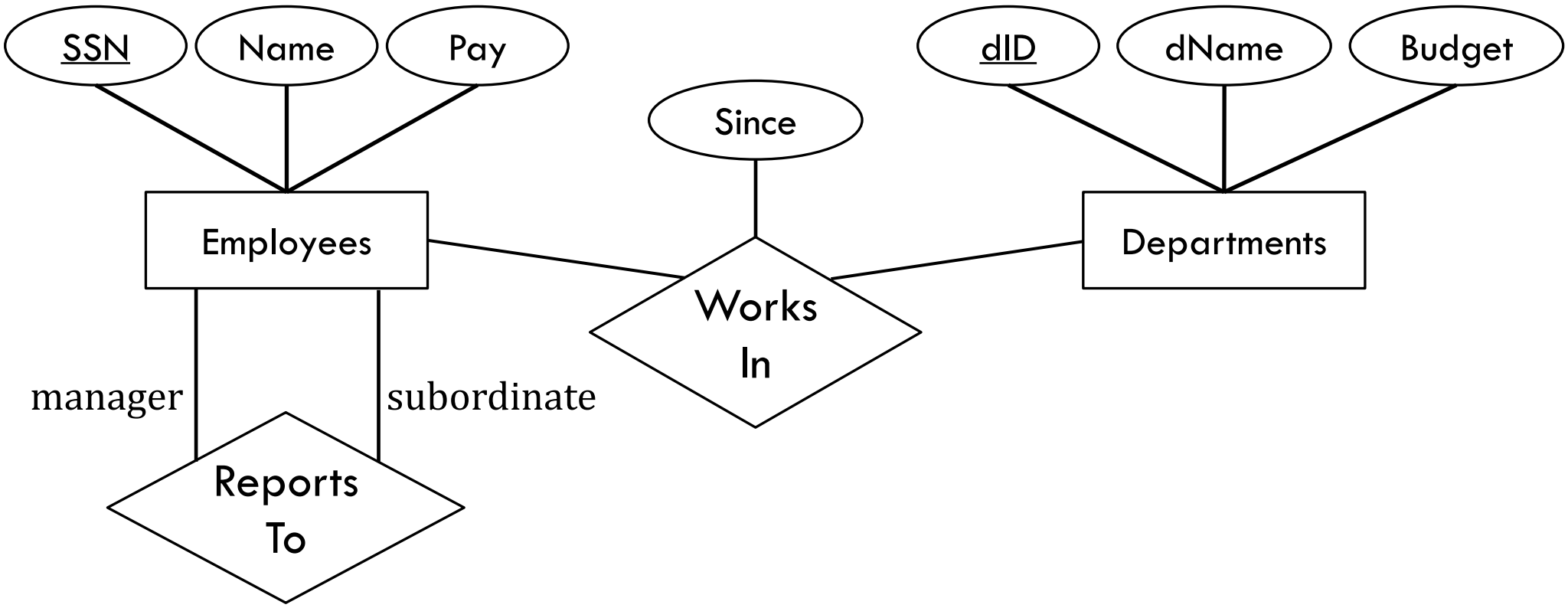
= relationship set

relationship sets do not need a key attribute

Inter-Relationship

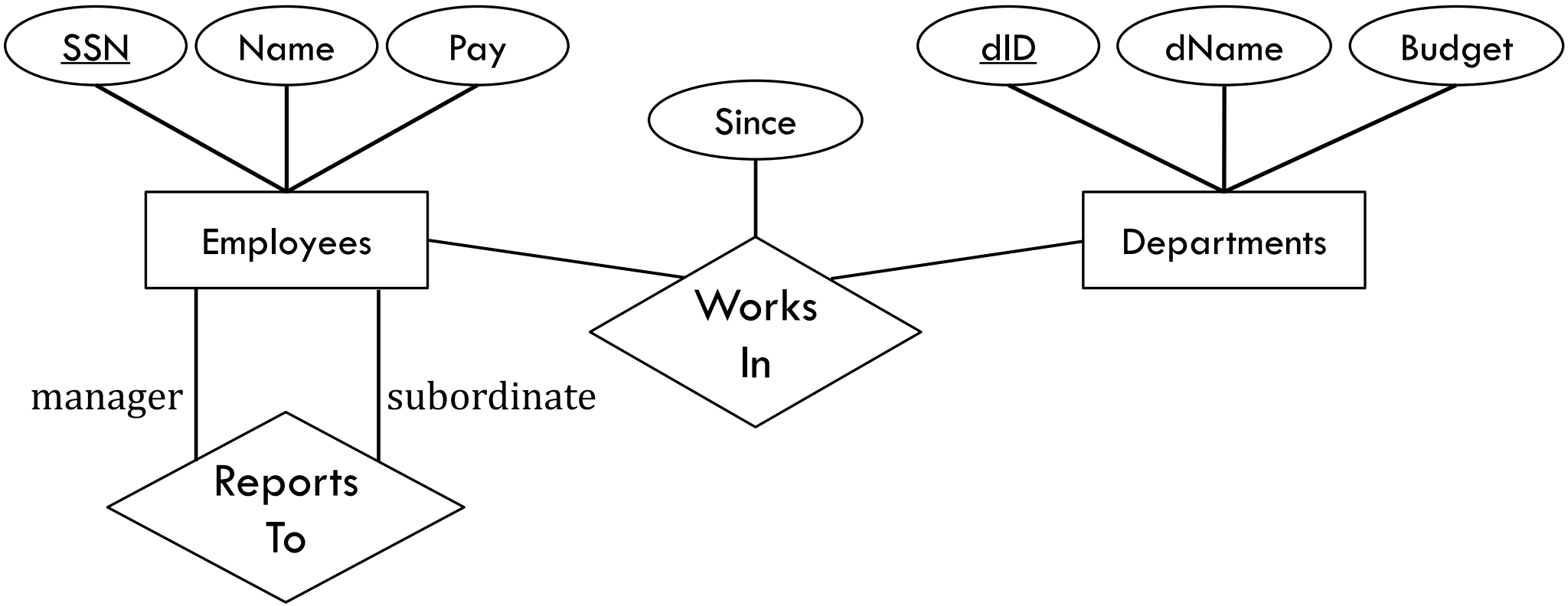


Cardinality



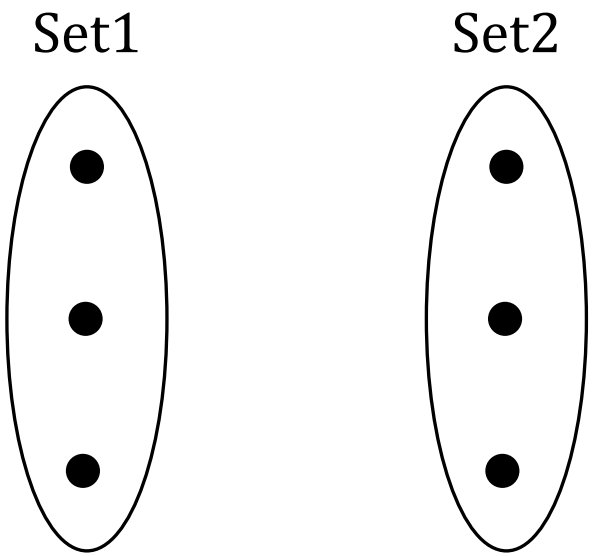
- Can an employee work in multiple departments?
- Can a department have multiple employees?

Cardinality



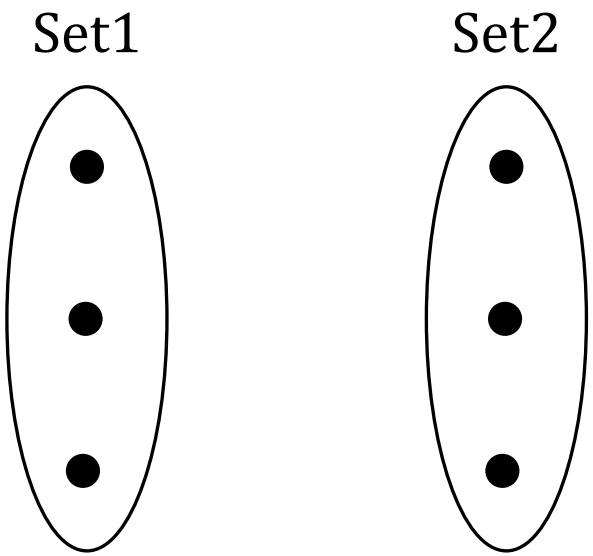
- The diagram does not specify this yet…

Cardinality



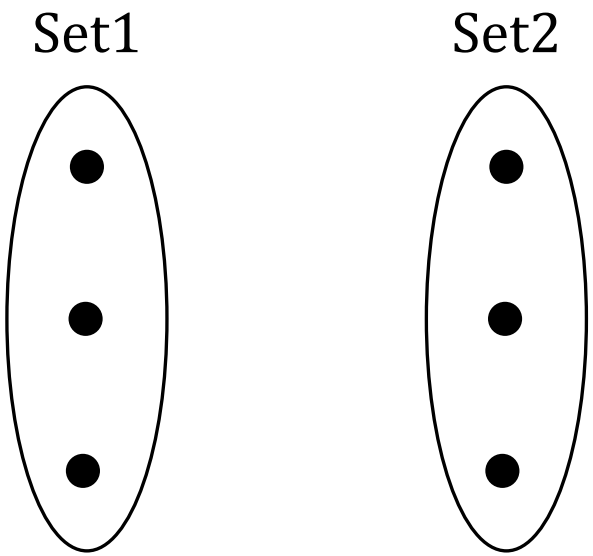
One to one

Cardinality



Many to many

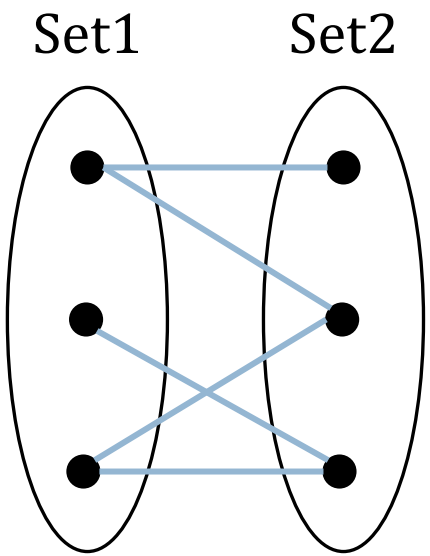
Cardinality



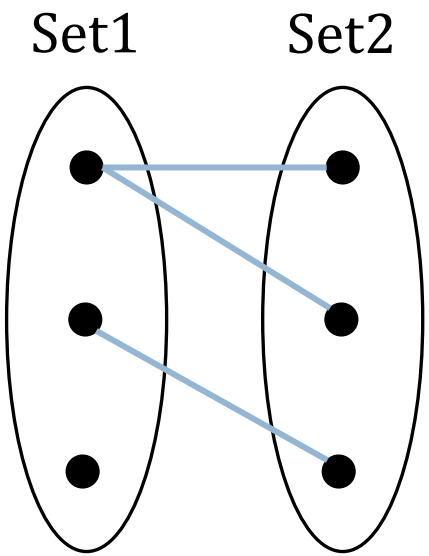
One to many

Cardinality

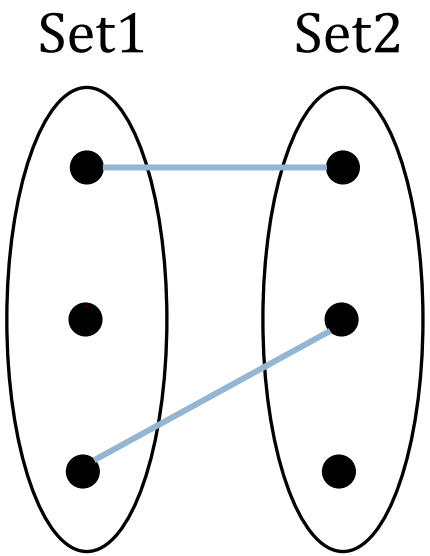
Three major classifications to restrict relationship sets



Many-to-Many



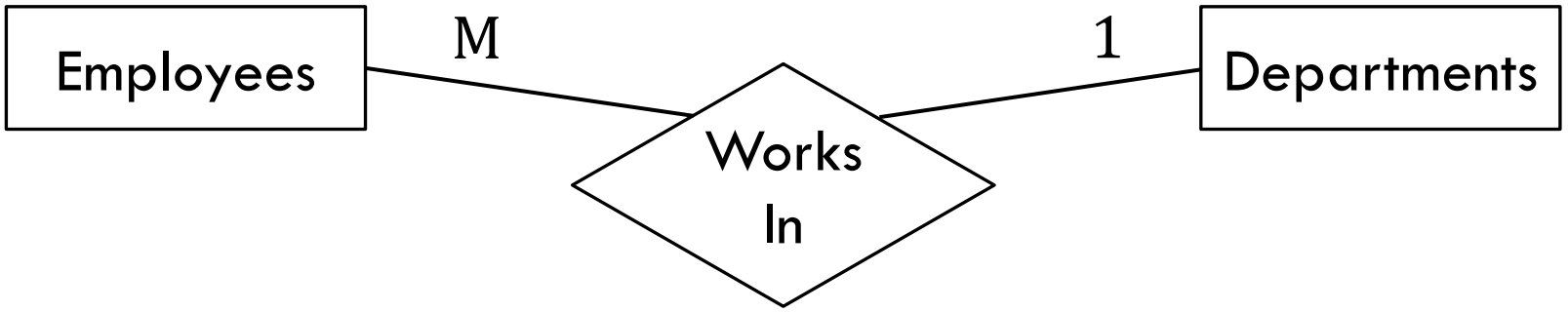
One-to-Many
vs
~~Many-to-One?~~



One-to-One

Cardinality

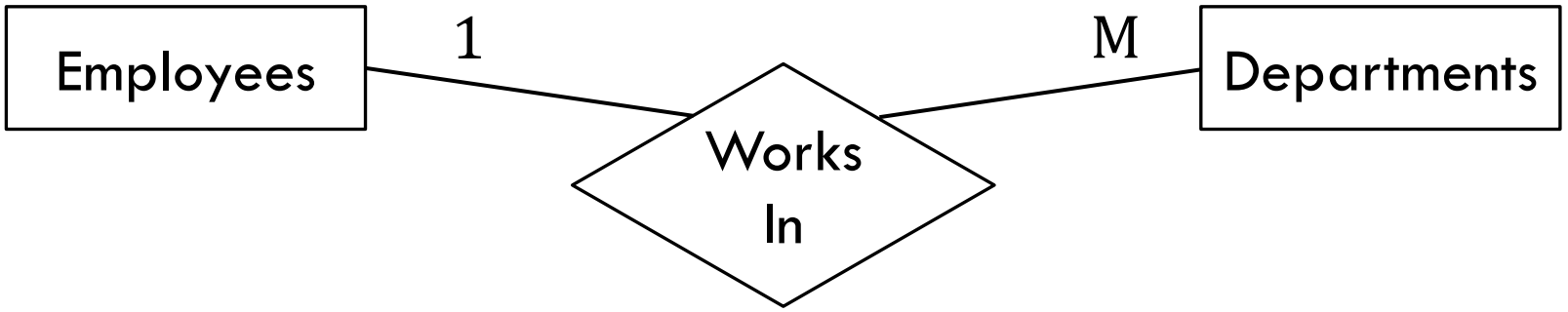
- Annotate **opposite edge** of relationship with cardinality



- "An employee can work for one department, but a department can have many employees"
  - 1-to-Many

Cardinality

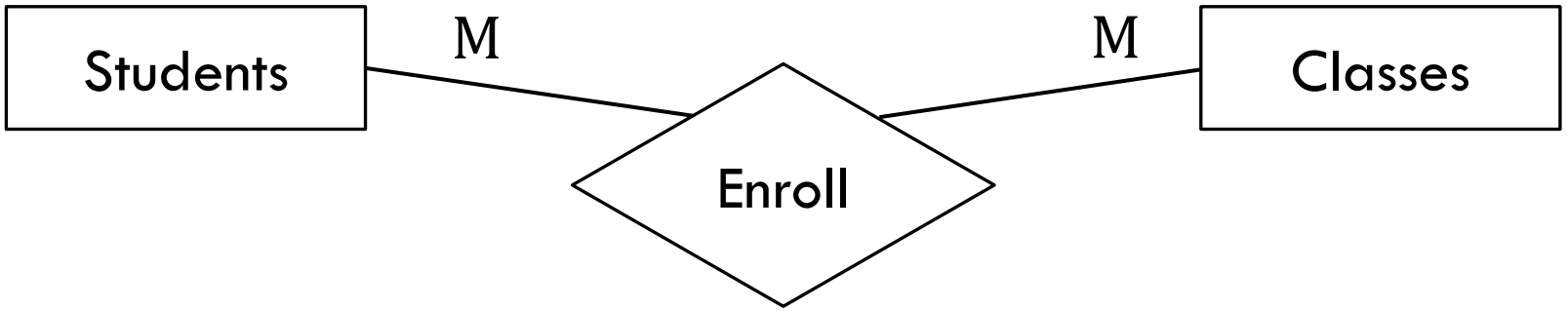
- Annotate opposite edge of relationship with cardinality



- "An employee can work for multiple departments, but a department can only have one employee"
  - 1-to-Many (reversed)

Cardinality

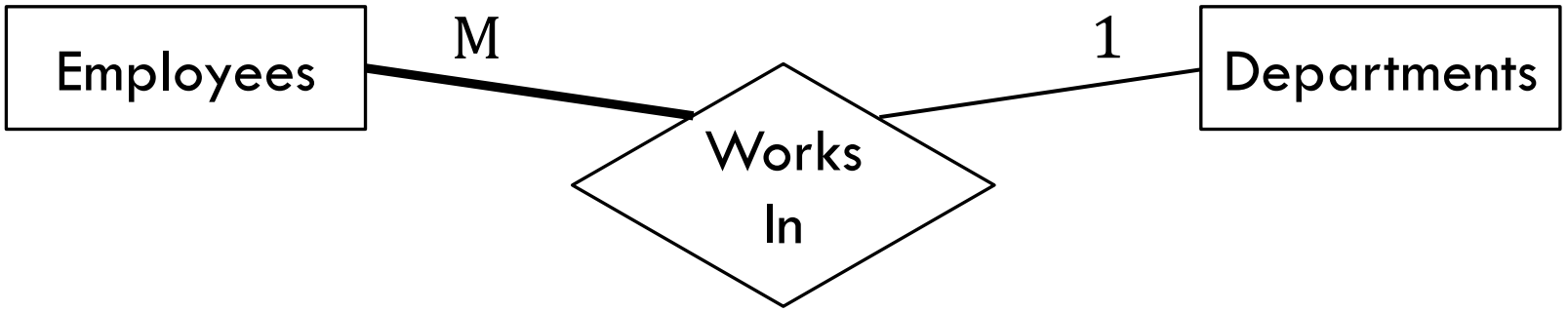
- Annotate opposite edge of relationship with cardinality



- "A student can take multiple classes, and a class can have multiple students"
  - Many-to-Many

Participation Constraint

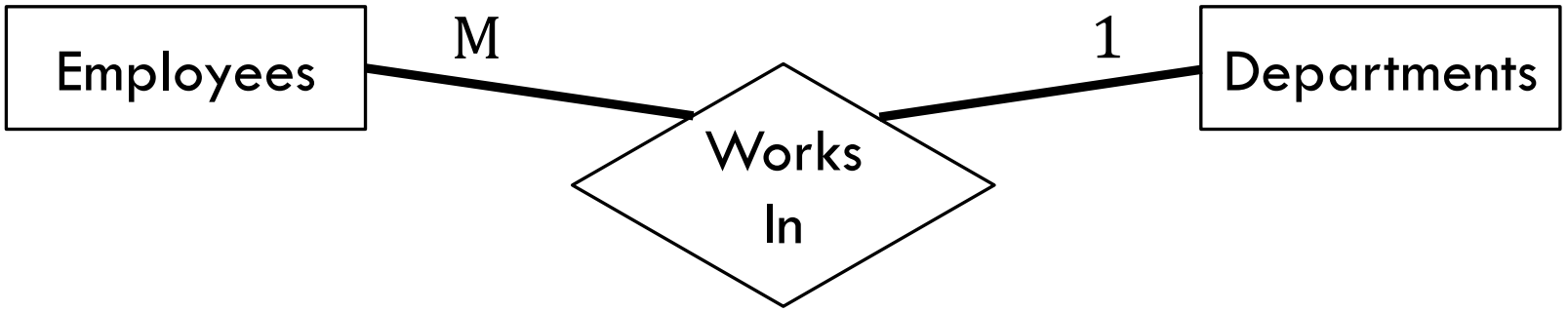
- Bold line indicates all entities in the set *must* participate
- Can also use a double line (two parallel lines).



- "An employee must work in 1 department, but a department does not necessarily have any employees"

Participation Constraint

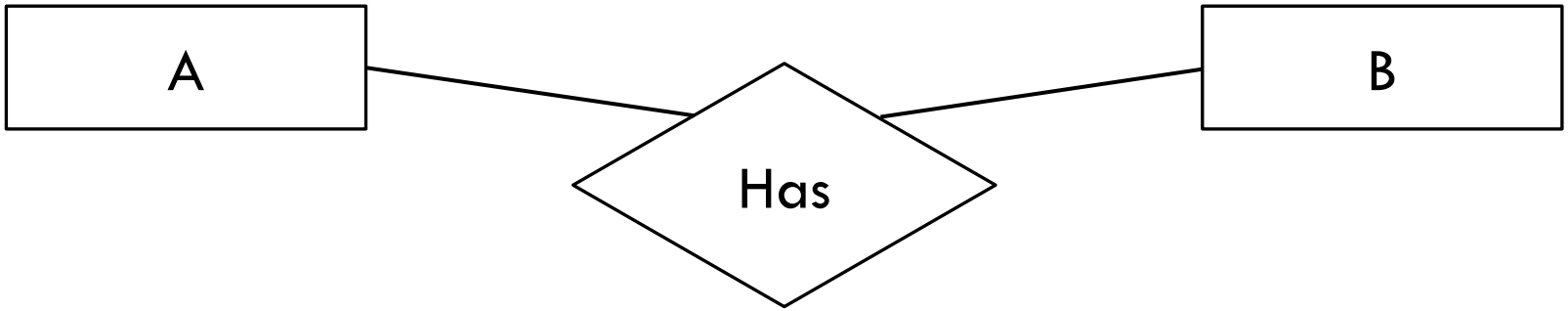
- Bold line indicates all entities in the set *must* participate
  - At least once



- "An employee must work in one department, a department must have at least one employee"

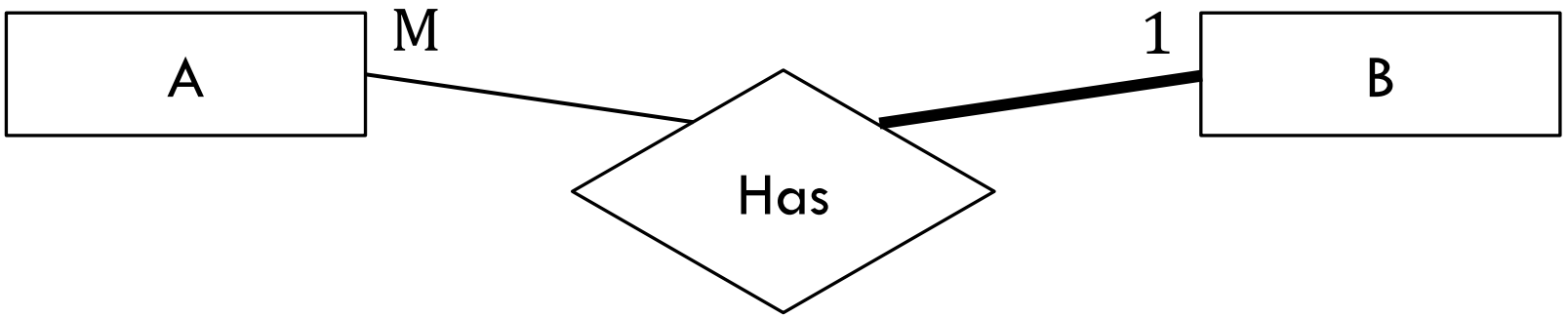
Practice (Annotate Diagram)

- An A has *at most* one B
- A B has *at least* one A



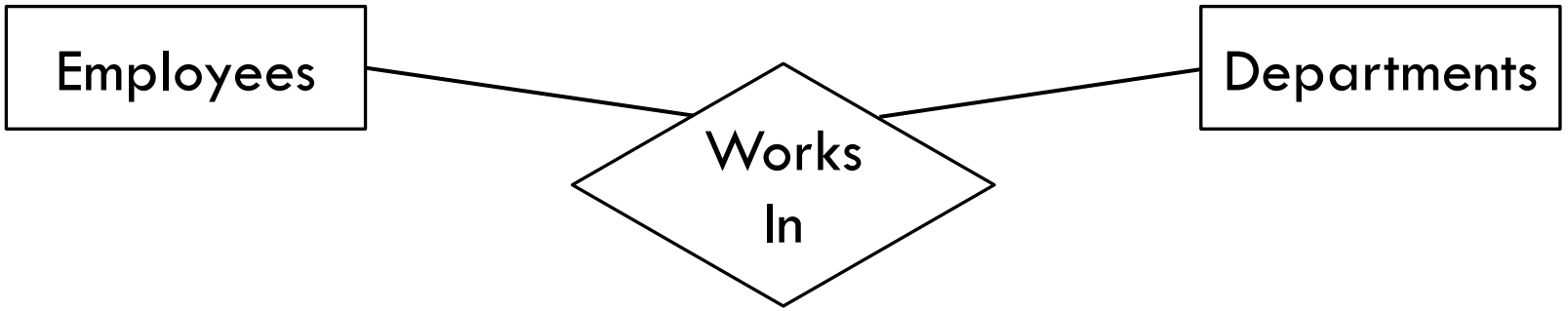
Practice (Annotate Diagram)

- An A has *at most* one B
- A B has *at least* one A



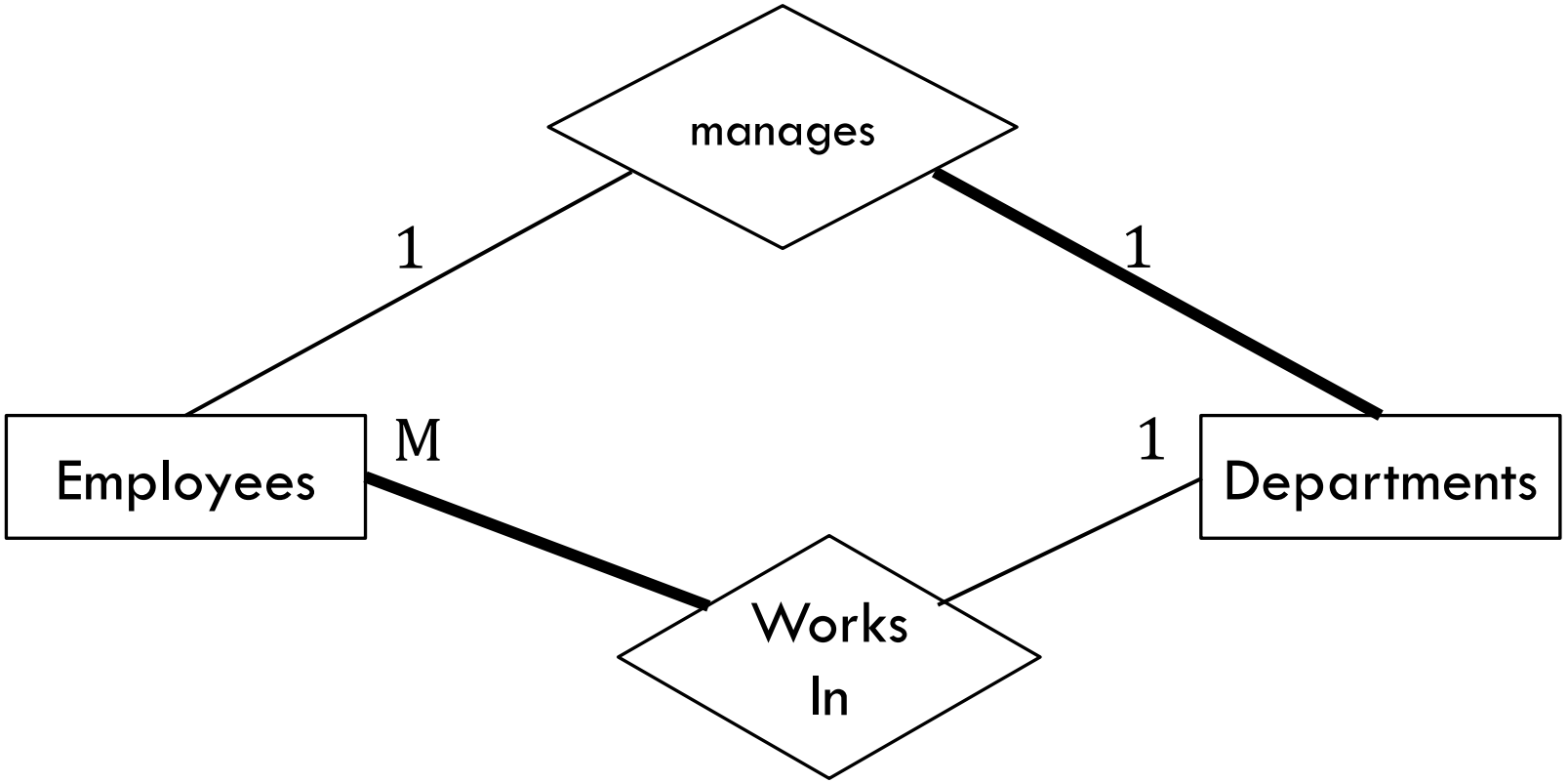
Practice

- A department can have multiple employees
- An employee works for exactly one department
- A department has exactly one manager
- An employee can manage up to one department



Practice

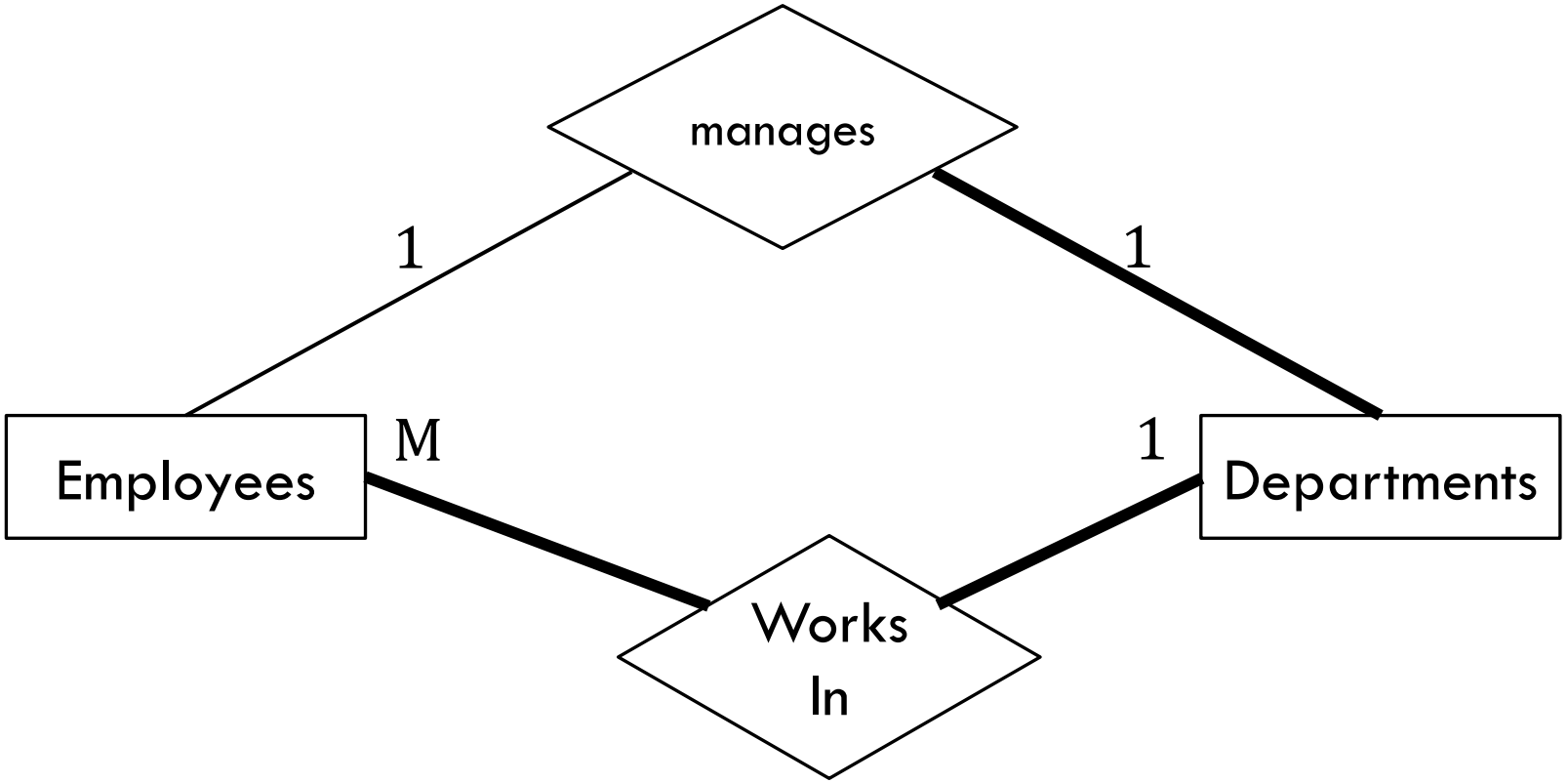
- A department can have multiple employees
- An employee works for exactly one department
- A department has exactly one manager
- An employee can manage up to one department



Practice

Same as before, but what if

- A department has at least one employee!!!



Try it Out

- Good options
  - draw.io – free!
  - lucidchart – better, but not free
- Working options
  - Powerpoint
  - Illustrator
  - MS Visio (online only for macs)
  - MS Paint