DATA SCI 7030: Database and Analytics

Tozammel Hossain
Oct 27, 2020

Data Science & Analytics
University of Missouri

Agenda

- Programmatic Access to Postgres
- Workflow in a Database Design
  - The Entity-Relationship Model

Accessing database

- Module 1
  - psql: interactive shell for accessing dbase
    - Provides access at a raw level
    - Useful for running large sql script
  - sql magic function
    - Works only in Jupyter notebook
- 3rd way of accessing database
  - psycopg2 and SQLAlchemy
    - Works within and outside Jupyter notebook

The Entity-Relationship Model

Mapping your client's enterprise information needs and policies into a relational database design.

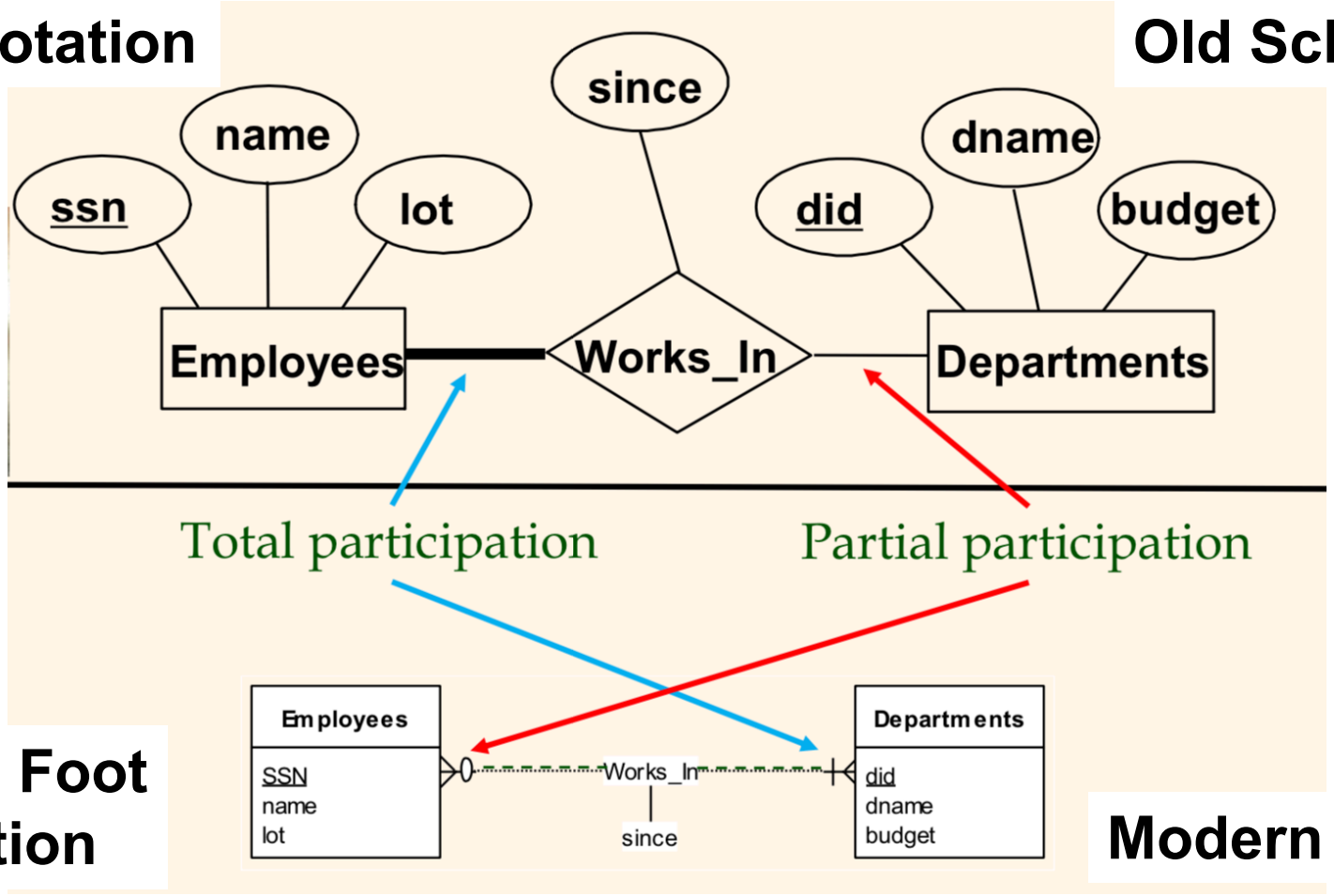
Steps in Database Design

- Requirements analysis
- Conceptual database design
- Logical database design
- Schema refinement
- Physical database design
- Application and security implementation
- ….

ER Diagrams

Chen Notation

Old School



Crow's Foot Notation

Modern Design

Conceptual design

- ER Model is used at this stage
- What are the **entities** and **relationships** in the enterprise?
- What **information** (i.e attributes) about these entities and relationships we should store in the database?
- What are the **integrity constraints** or **business rules** that hold?

Conceptual design

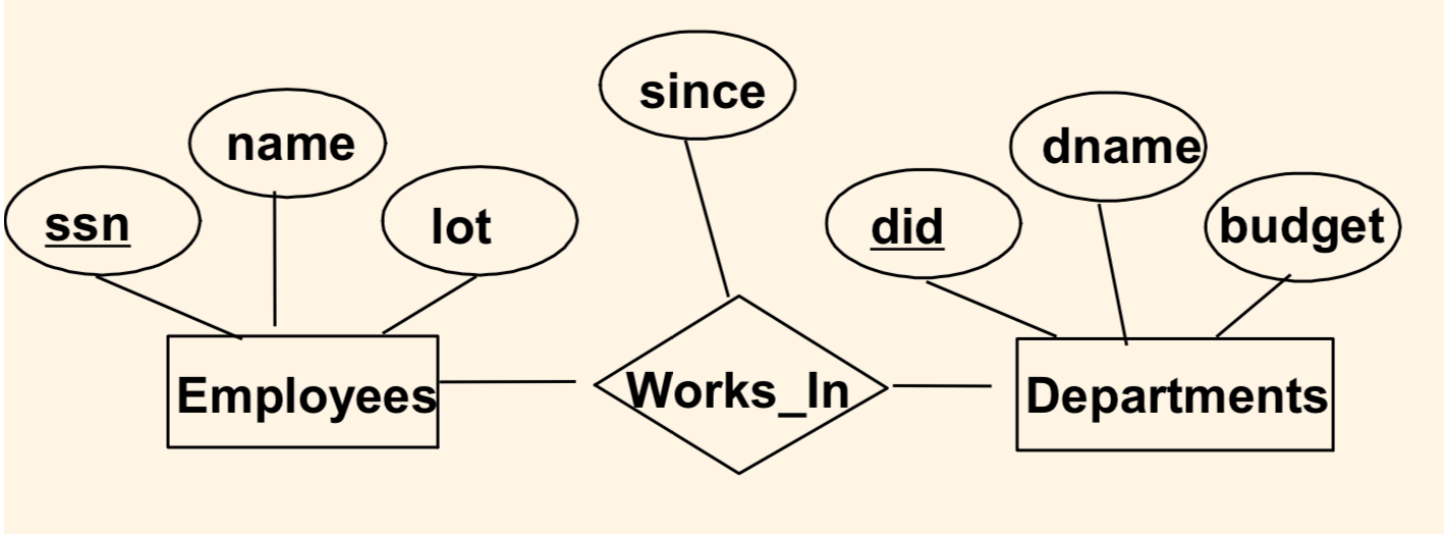
- A database **schema** in the ER Model can be represented pictorially (*ER diagrams*)
- We then can map an ER diagram into a relational schema

Why ER diagrams?

- Your client tells you
  - We would like to build a DB application software package such that we can query the system like **who works in what department since when**. The company is interested in keeping the following information:
    - Every **employee** has a **SSN**, **name**, and **lot number**
    - Each **department** has a **department ID**, **name**, and **budget**
- The above is the output of a **requirement analysis**

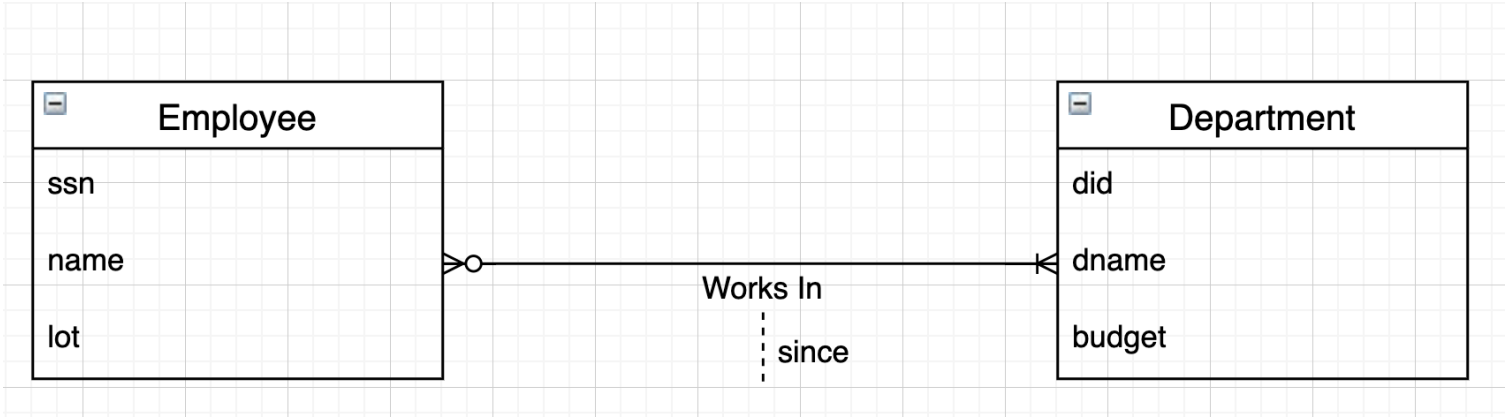
Why ER diagrams?

- Instead of coding SQL DDL directly, model the problem pictorially first
- "Translate" what users want to a more detail and precise description that can be implemented in a DBMS



Chen Notation

Why ER diagrams?



Crow's Foot Notation

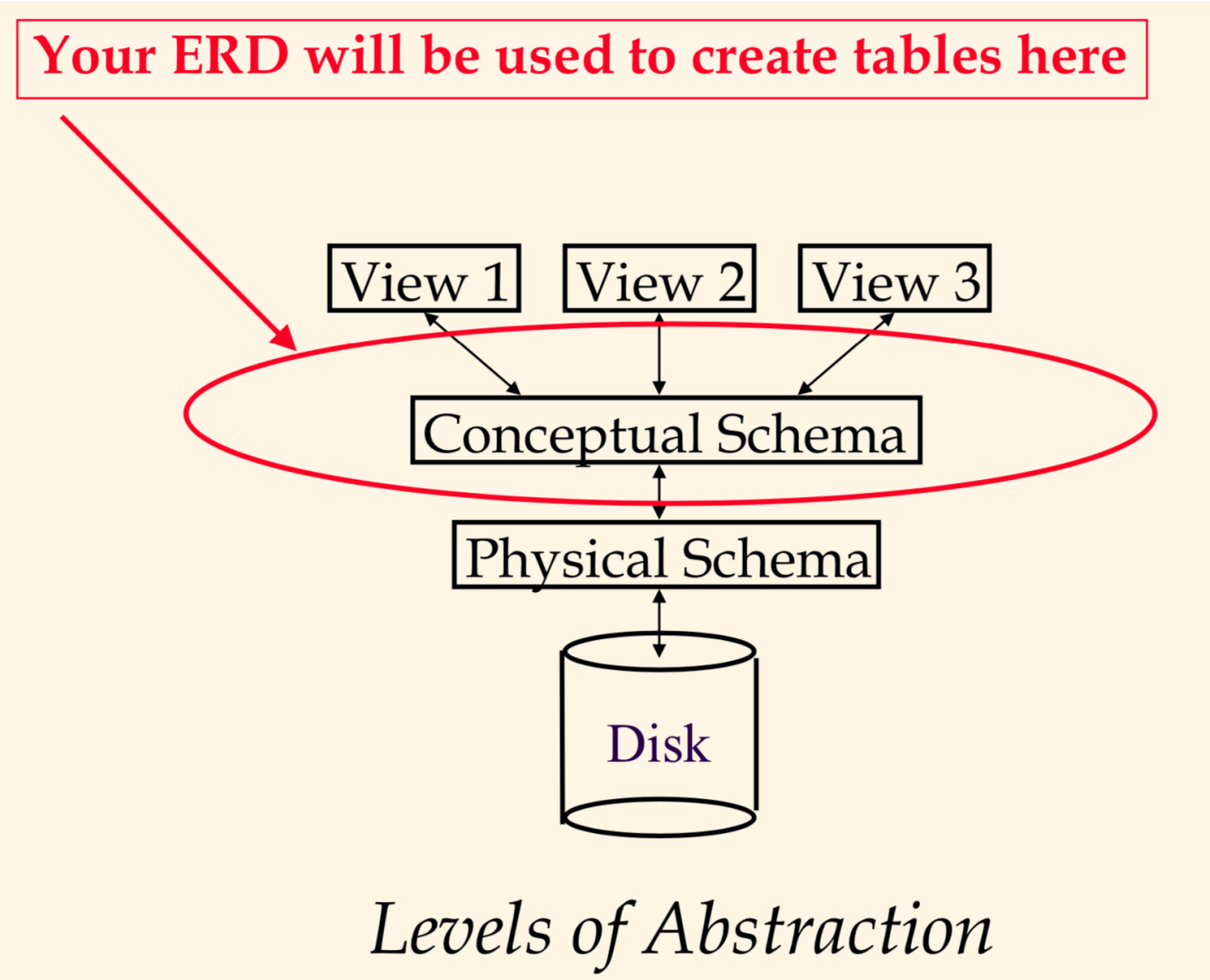
- We will be using Crow's Foot Notation
- Preferred Notation

Available Tools

- pen + paper
- power point
- MS Visio
- draw.io
- lucid diagram
- mermaid

  - https://mermaid-js.github.io/mermaid-live-editor

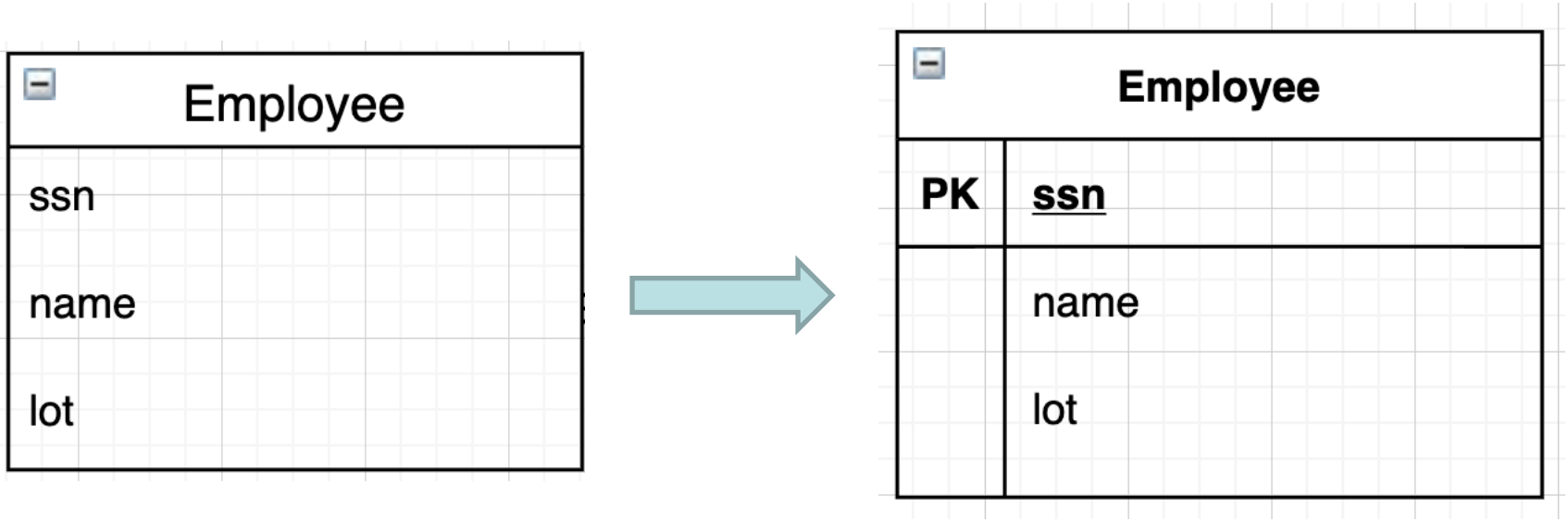
ERD in Abstraction Level



Entity & Attributes

- What is an entity?
  - An object for which we need to store data
  - Physical existence vs conceptual existence
    - Physical: students, cars, houses, employees
    - Conceptual: courses, jobs, companies
- Entity vs Entity type vs Entity Sets
  - An entity is an object of an entity type (OOP)
    - E.g., E1 is an entity of type Employee
  - Defined by a set of properties (i.e. **attributes**)
  - **Entity set** is a collection of entities

Entity & Attributes



Attribute Types

- Simple attribute
  - cannot be further subdivided into components
- Composite attribute
  - can be splitted into components
- Single-valued attribute
  - Eg.??
- Multi-valued attribute
  - E.g.??
- Derived attributes
  - E.g. ??

Advantages and Disadvantages of Storing Derived Attributes

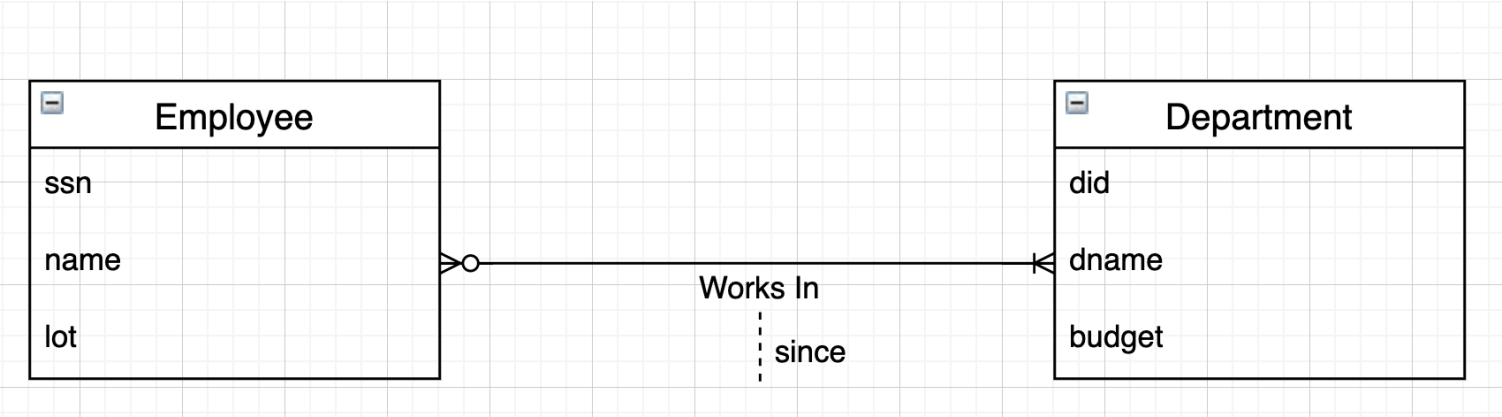
TABLE 4.2

ADVANTAGES AND DISADVANTAGES OF STORING DERIVED ATTRIBUTES

	DERIVED ATTRIBUTE	
	STORED	**NOT STORED**
Advantage	Saves CPU processing cycles Saves data access time Data value is readily available Can be used to keep track of historical data	Saves storage space Computation always yields current value
Disadvantage	Requires constant maintenance to ensure derived value is current, especially if any values used in the calculation change	Uses CPU processing cycles Increases data access time Adds coding complexity to queries

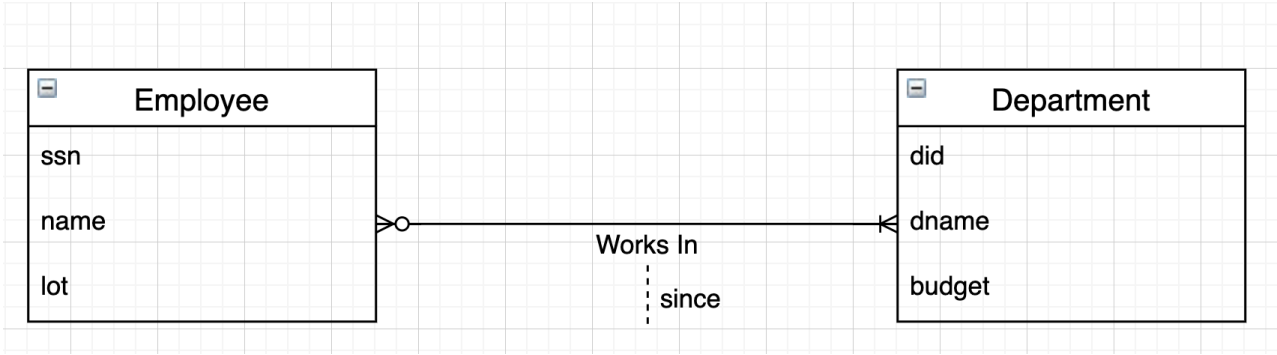
Relationships

- What is a relationship?
  - Association among two or more entities
    - E.g., **John Smith** works for **Marketing Department** since **2020-01-17**



Constraints

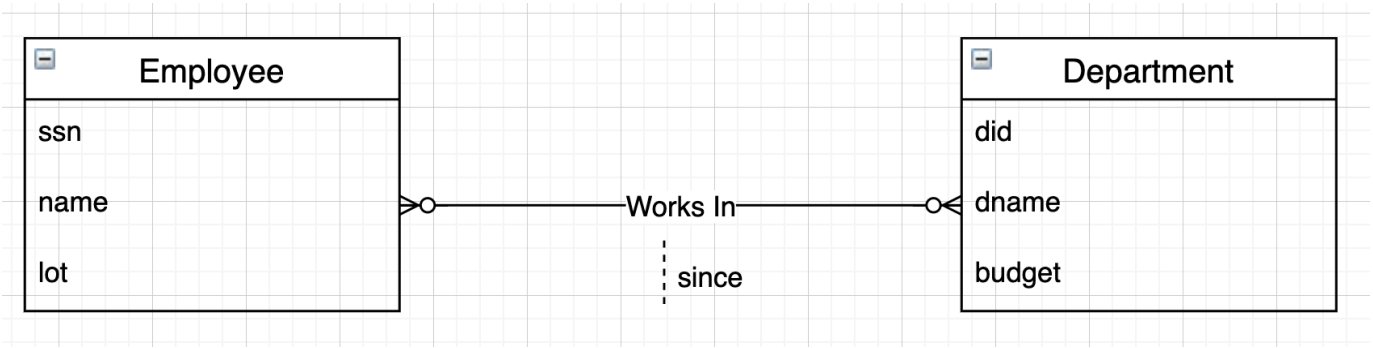
- What are the **integrity constraints** or **business rules** that hold?



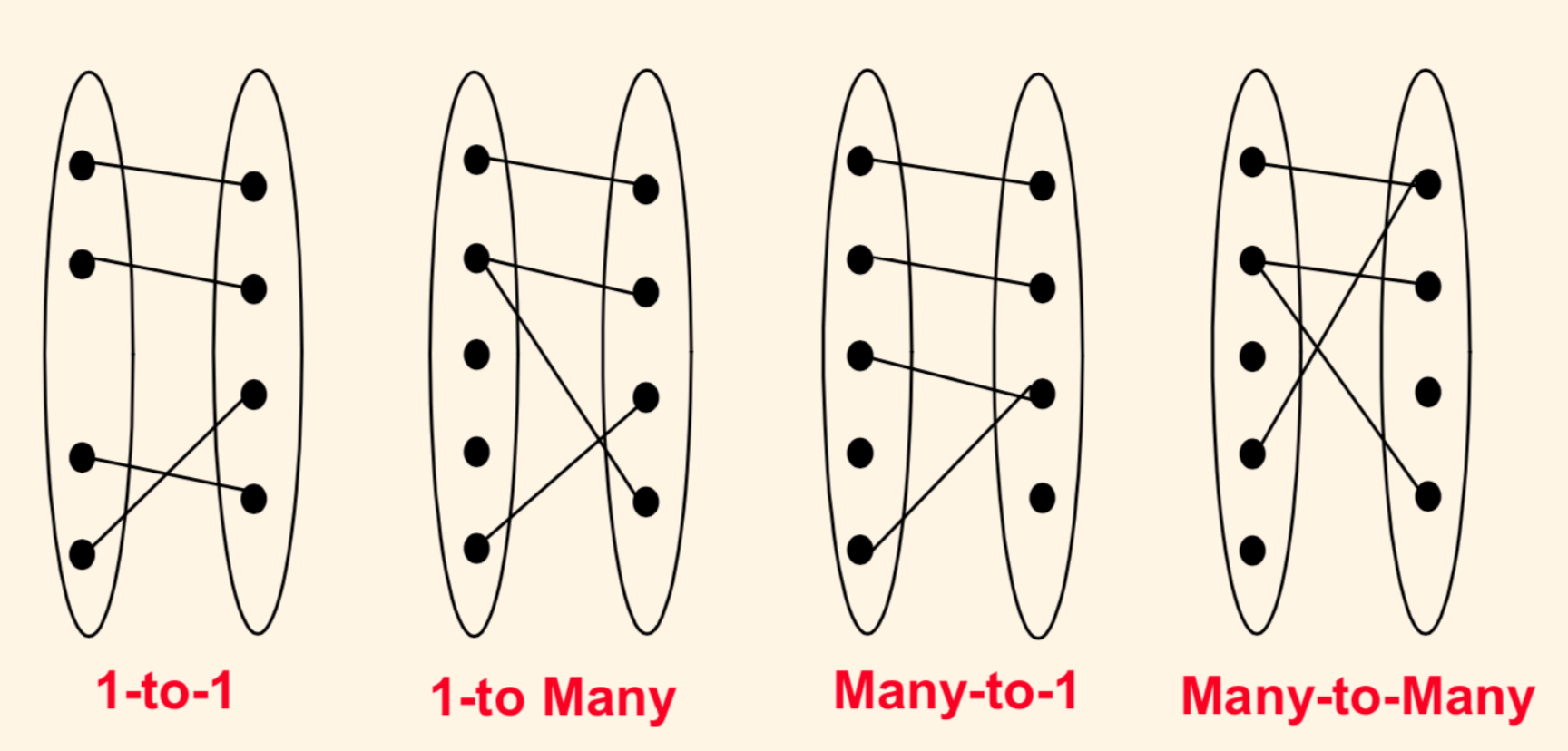
- Consider **Works_In**: An employee can work in many departments; a dept can have many employees
  - A prof could be affiliated with many departments
- Q: how to read the above diagram?

Constraints

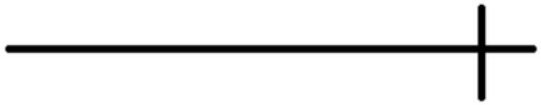
- ERD changes according to the business rules



Participation Constraint

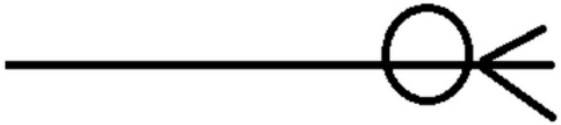


Constraints



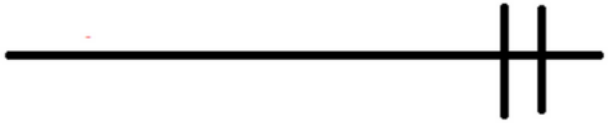
One

Many



Zero or Many

One or Many

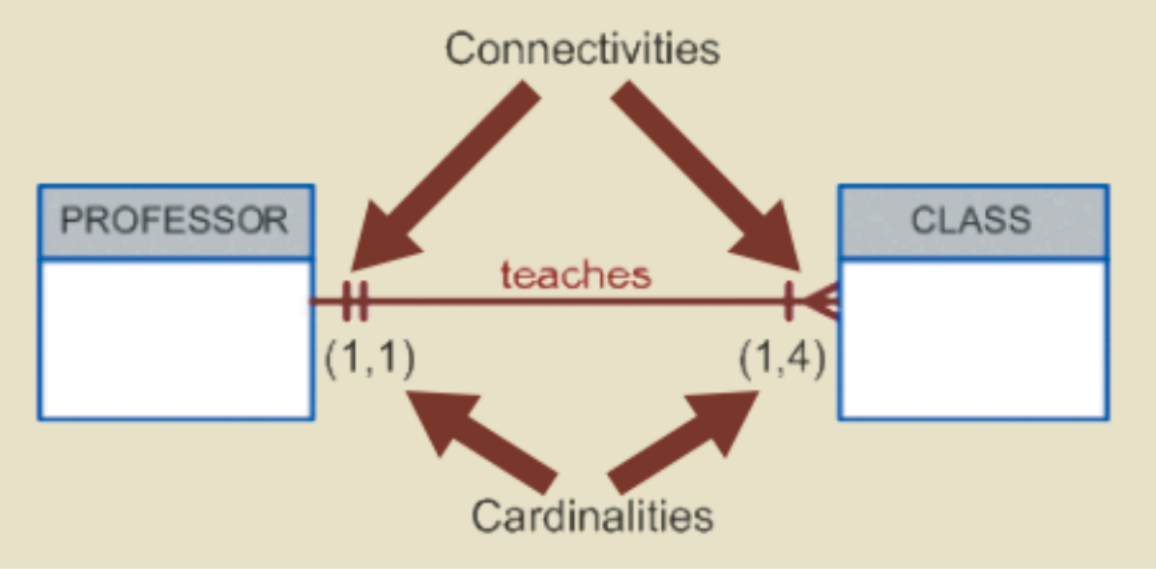


One and only one

Zero or one

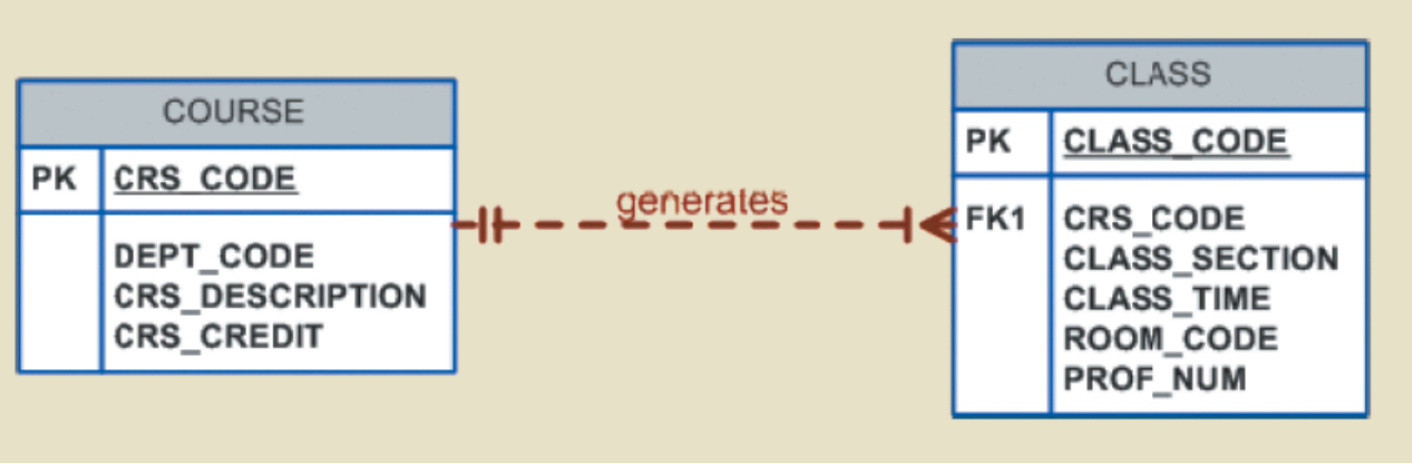
link: https://www.cs.dartmouth.edu/~cs61/Resources/ERD_Relationship_Symbols_Quick_Reference.pdf

Connectivity and Cardinality



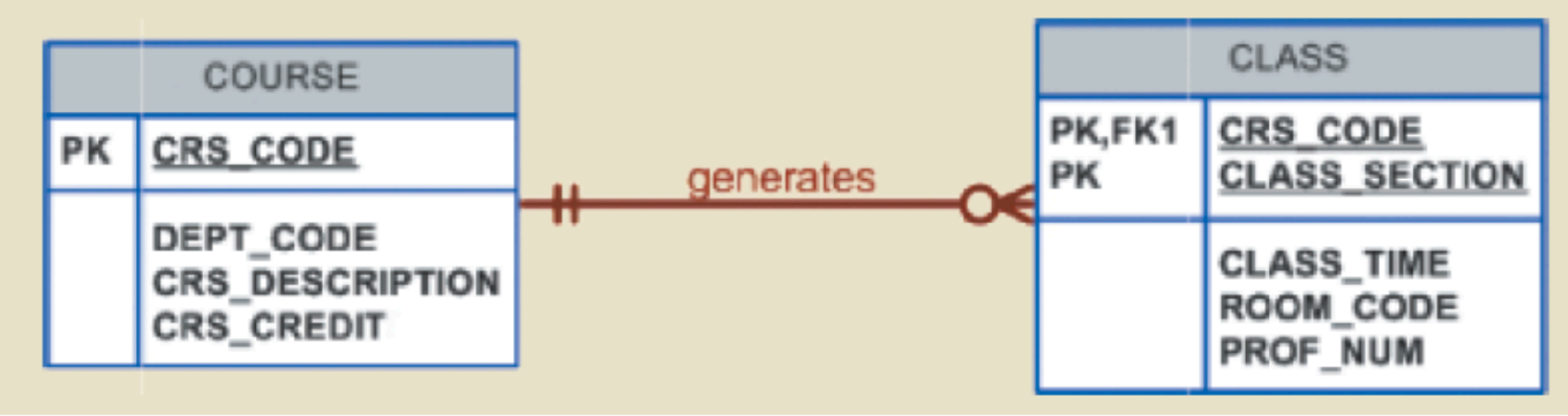
Weak Relationship

- Aka Non-Identifying
  - A class **is** for a course
  - An **employee** has **dependents** in his **policy**

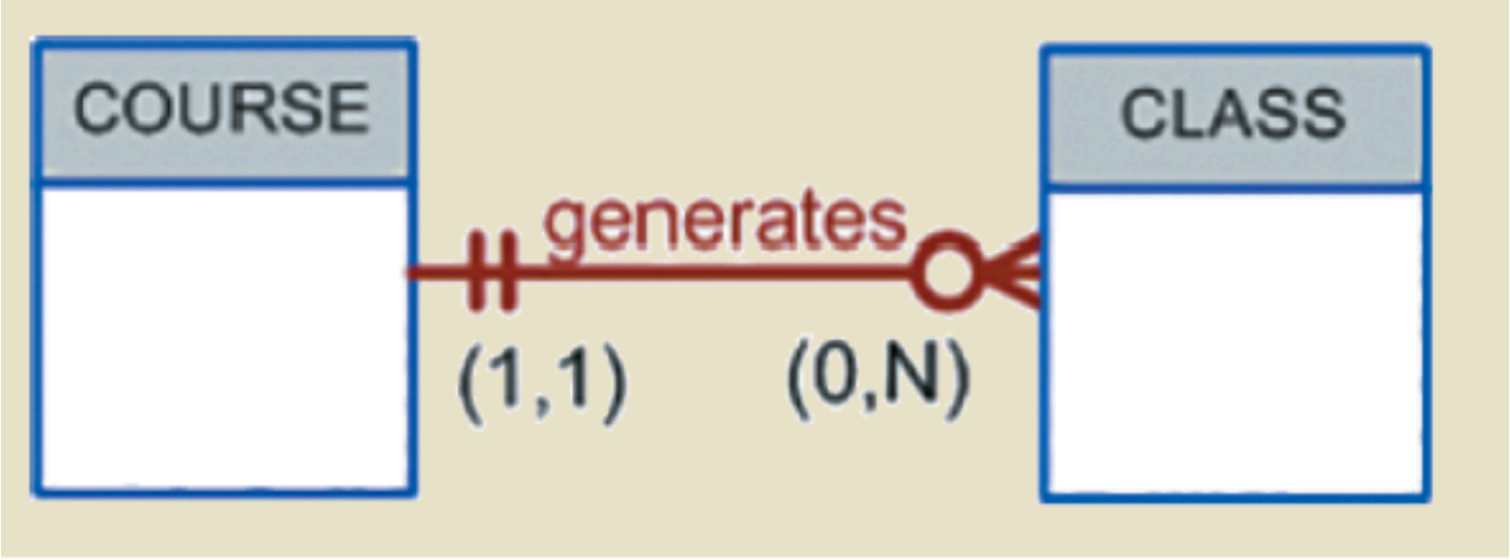


Strong Relationship

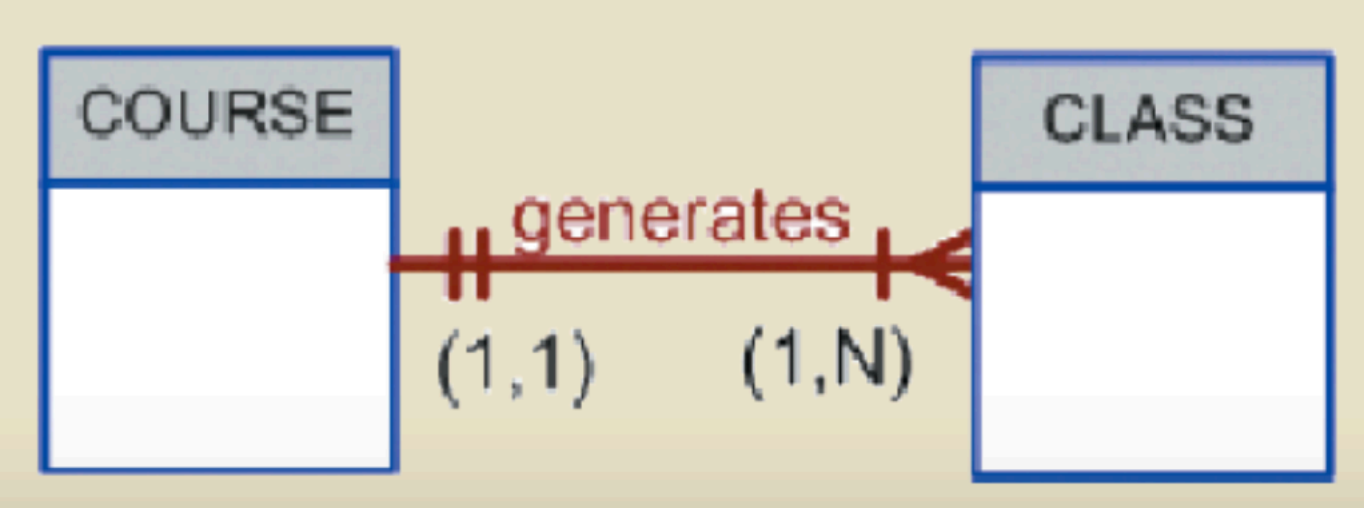
- Aka identifying relationship



Optional Relation

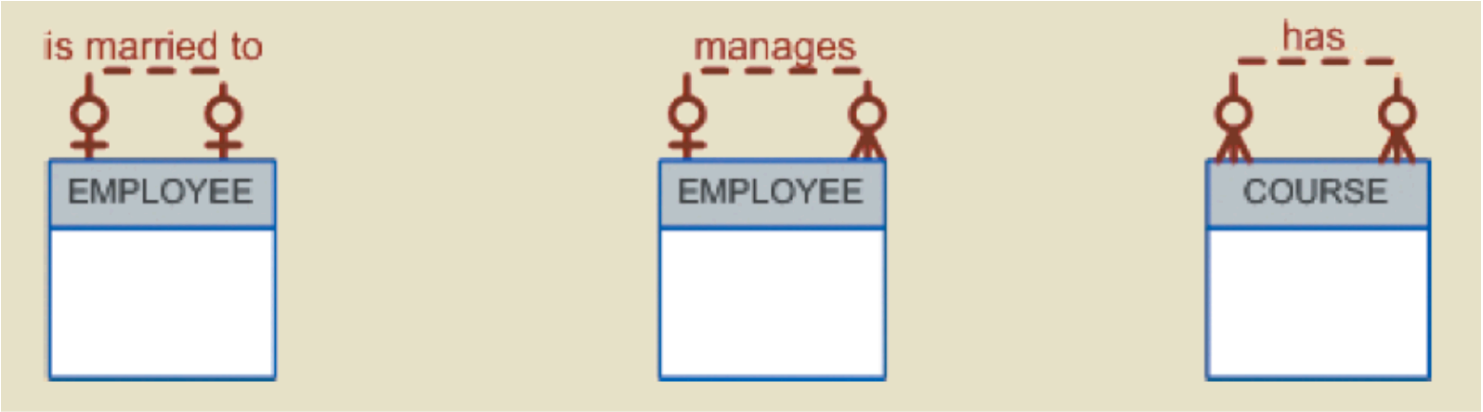


Mandatory Relationship



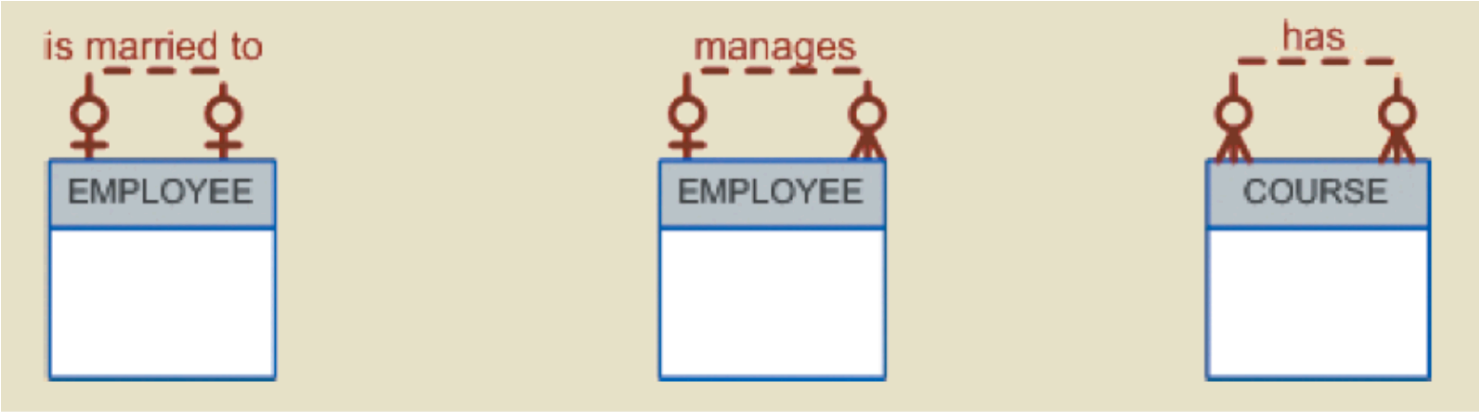
Relationship Degree

- Unary Relation (aka recursive relation)

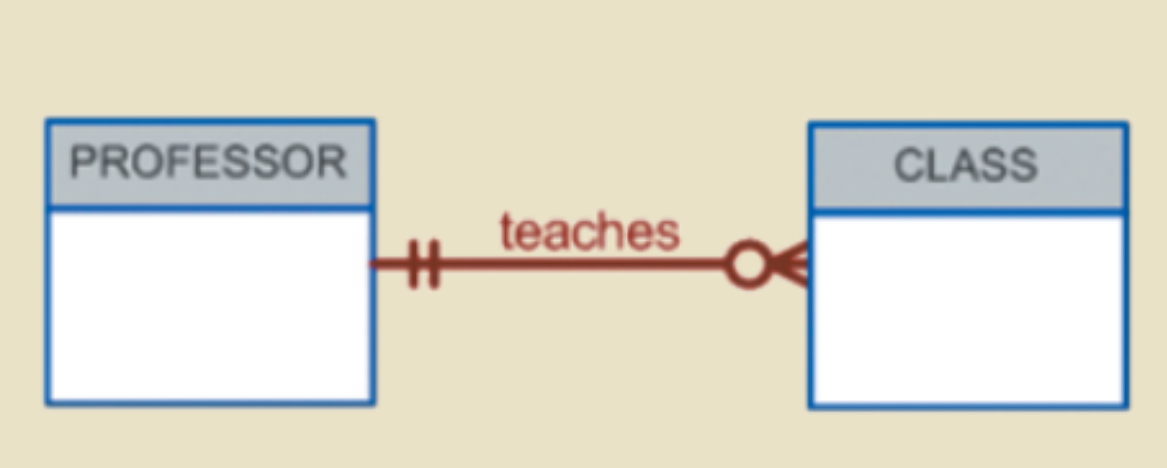


Relationship Degree: Unary

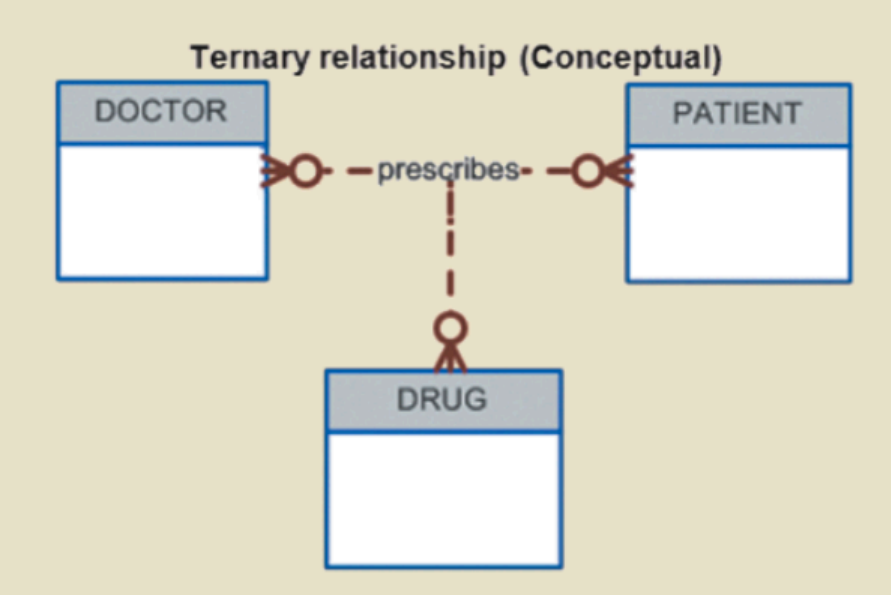
- Unary Relation (aka recursive relation)



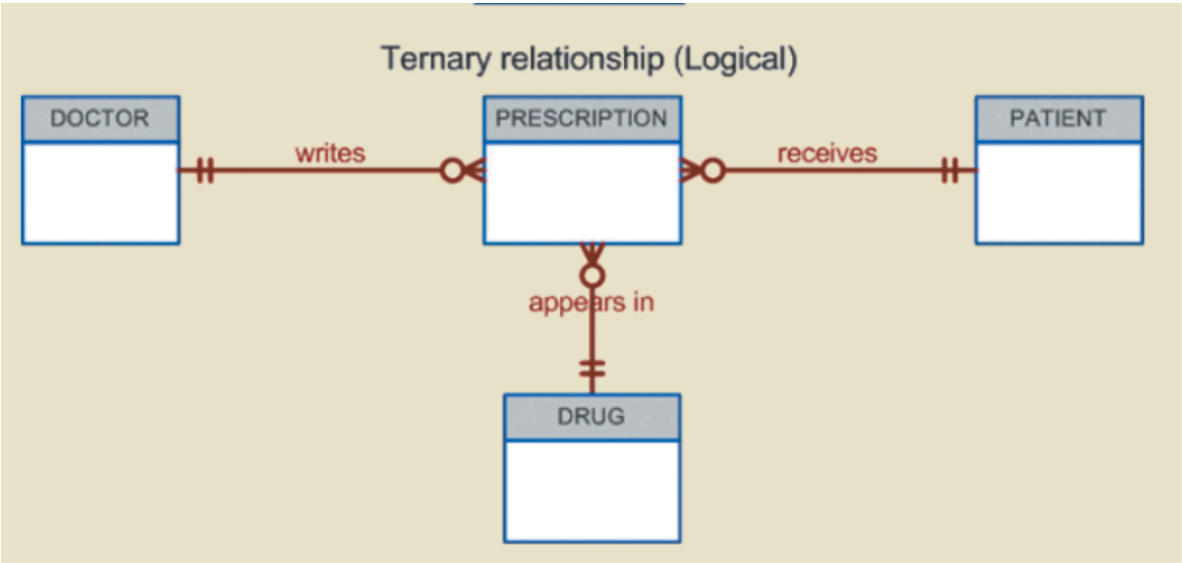
Binary Relationship



Ternary Relationship



Ternary Relationship



Quiz

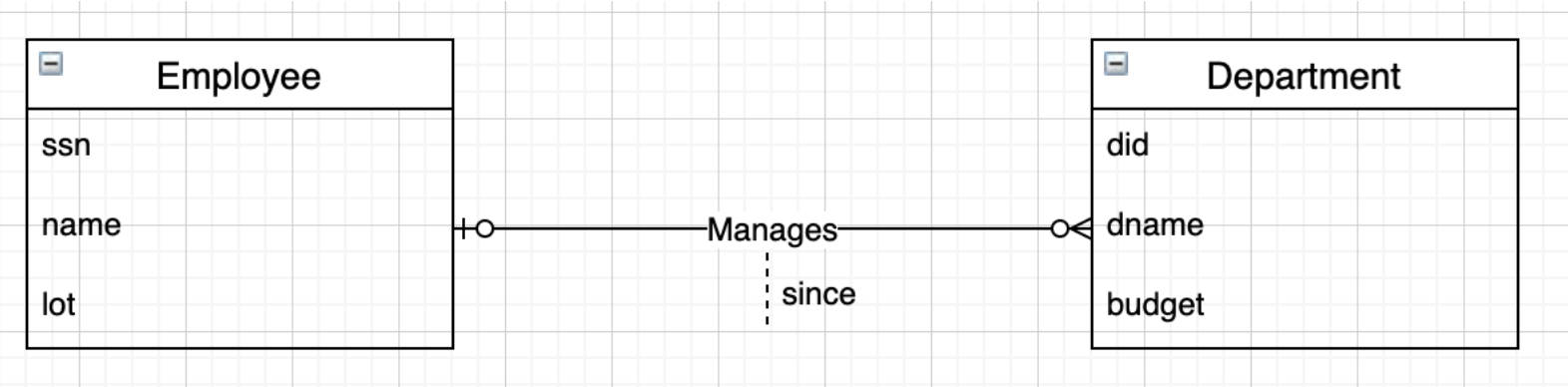
- Q: How to conceptualize **manages** relationship?
  - Requirement: Each dept has **at most one** manager

Quiz

- You use the following notation to express your ER diagram
  - o: Zero
  - I: one
  - < or >: many
  - E.g. To express a many to many relationship between entities A & B, we can write
    - A >o----o< B

Constraints

- Q: How to conceptualize **manages** relationship?
  - Requirement: Each dept has **at most one** manager



Thanks