BLG 317E
DATABASE SYSTEMS

WEEK 8

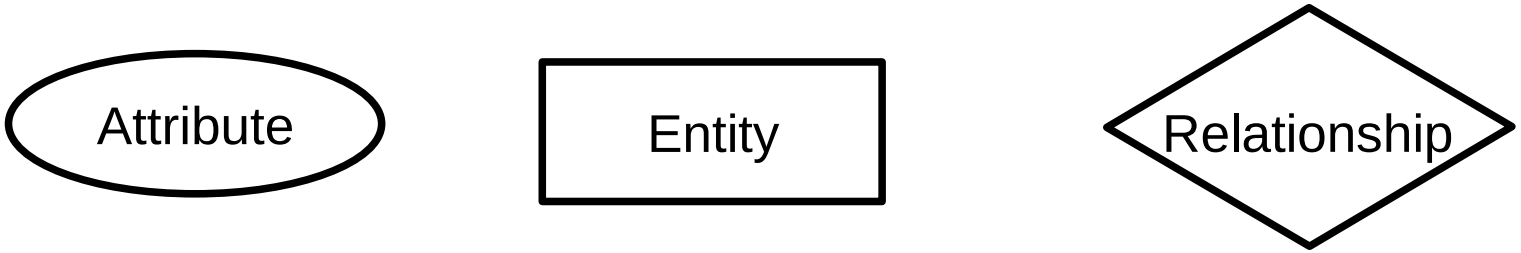
Entity – Relationship (ER) Modeling

Entity-Relationship Diagrams

- ERD is a data modeling method that shows
  - entities (things/objects/concepts)
  - and how they relate to each other in a database.
- ER diagrams are used in the **Conceptual Design** of a database.
- Entities have attributes (data fields).
- A relationship in an ERD defines how two entities are related.

Entity-Relationship Diagram (Chen Notation)

- **Rectangles** represent entities.
- **Diamonds** represent relationships between entities.
- **Lines** link attributes to entities, and entities to relationships.
- **Ellipses** represent attributes
- **Underlined names** indicate key attributes

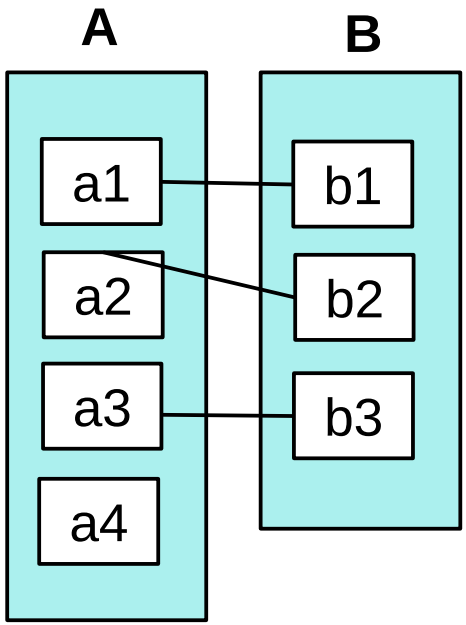


Cardinalities of Entity Relations

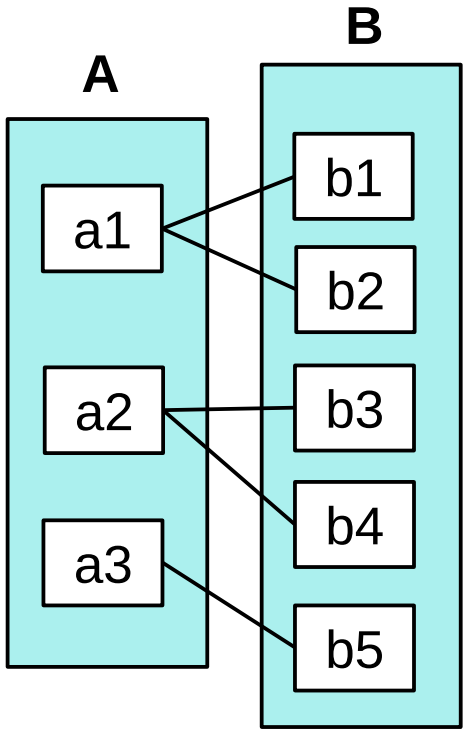
- **Cardinality** represents the number of instances matched between two related entities.
- Common cardinality values are zero, one, or many.
- Cardinality constraints are represented by drawing a line between the entities.
- Directed line ($\rightarrow$) means One
- Undirected line (—) means Many

Cardinalities of Entity Relations

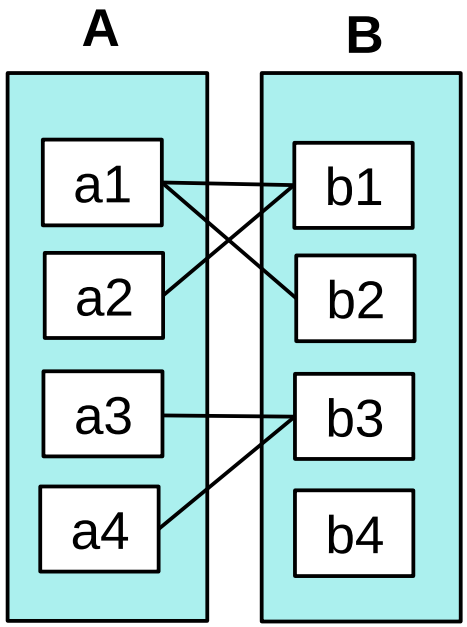
- A and B are entities (table A and table B.)
- a1, a2, ... , b1, b2, ... are tuples (records).
- Some tuples in A and B may not be mapped to any tuples in the other entity.



One to one



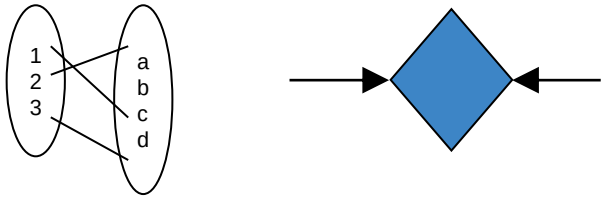
One to many



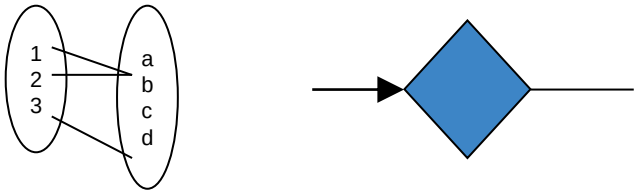
Many to many

Multiplicity of E/R Relationships

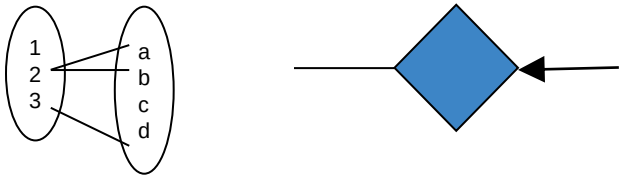
One-to-one:



Many-to-one:



One-to-many:



Many-to-many:

Indicated using arrows

A relationship between X and Y means that **there exists a function mapping from X to Y** (*recall the definition of a function)*

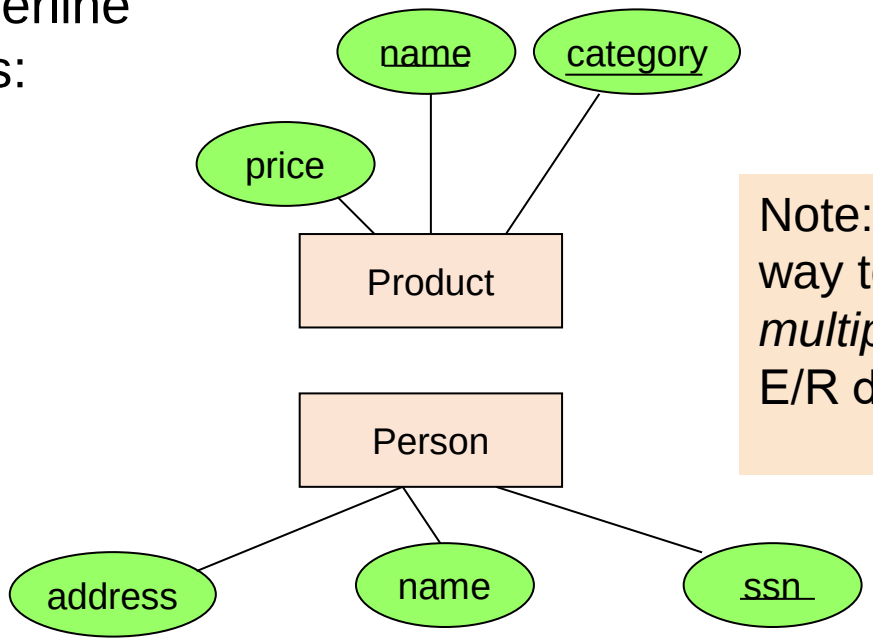
Constraints in E/R Diagrams

- Finding constraints is part of the E/R modeling process. Commonly used constraints are:
  - Keys: Implicit constraints on uniqueness of entities
    - *Ex: An SSN uniquely identifies a person*
  - Single-value constraints:
    - *Ex: a person can have only one father*
  - Referential integrity constraints: Referenced entities must exist
    - *Ex: if you work for a company, it must exist in the database*
  - Other constraints:
    - *Ex: peoples' ages are between 0 and 150 (cannot be represented in ER)*

Recall FOREIGN KEYs!

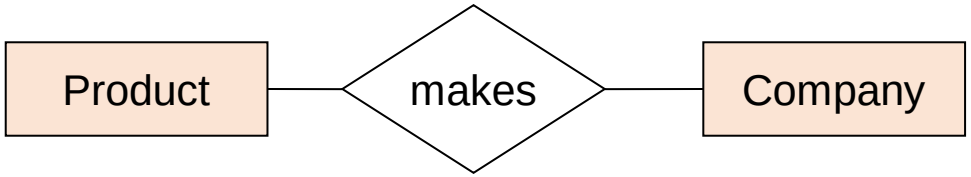
Keys in E/R Diagrams

Underline keys:

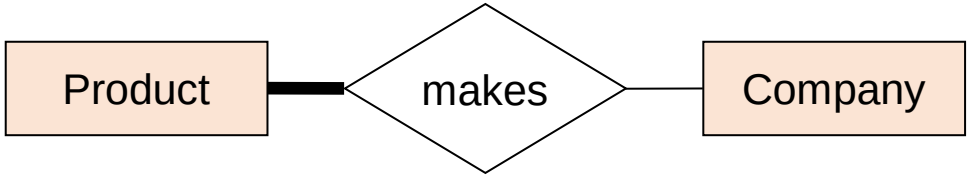


Note: no formal way to specify *multiple* keys in E/R diagrams…

Participation Constraints: Partial v. Total

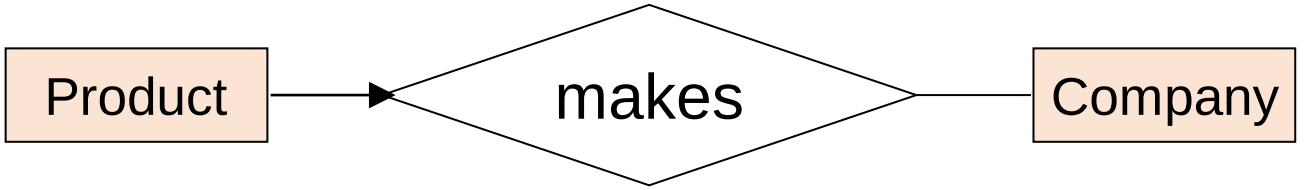


Are there products made by no company?
Companies that don't make any product?

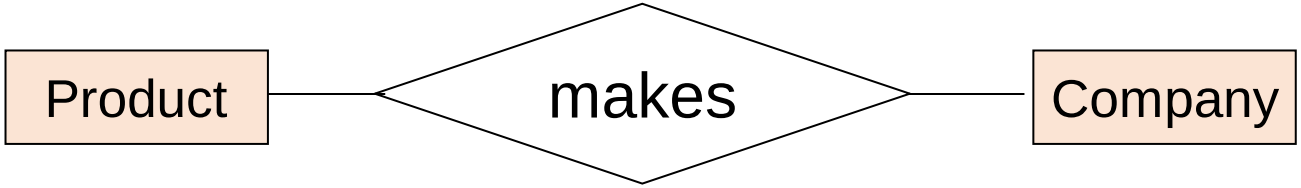


Bold/Double line indicates *total participation* (i.e. here: all products are made by at least one company)

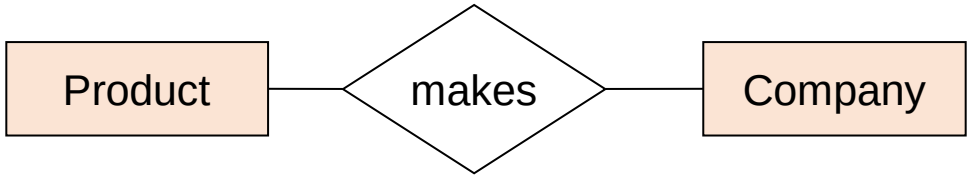
Single Value Constraints



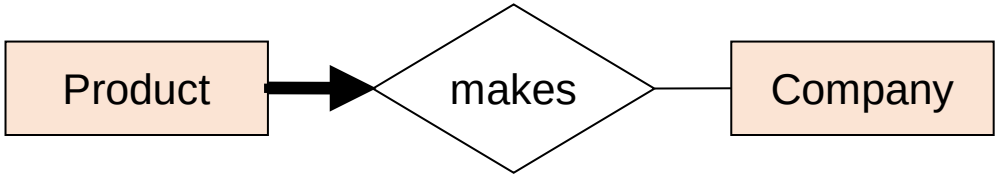
v. s.



Referential Integrity Constraints



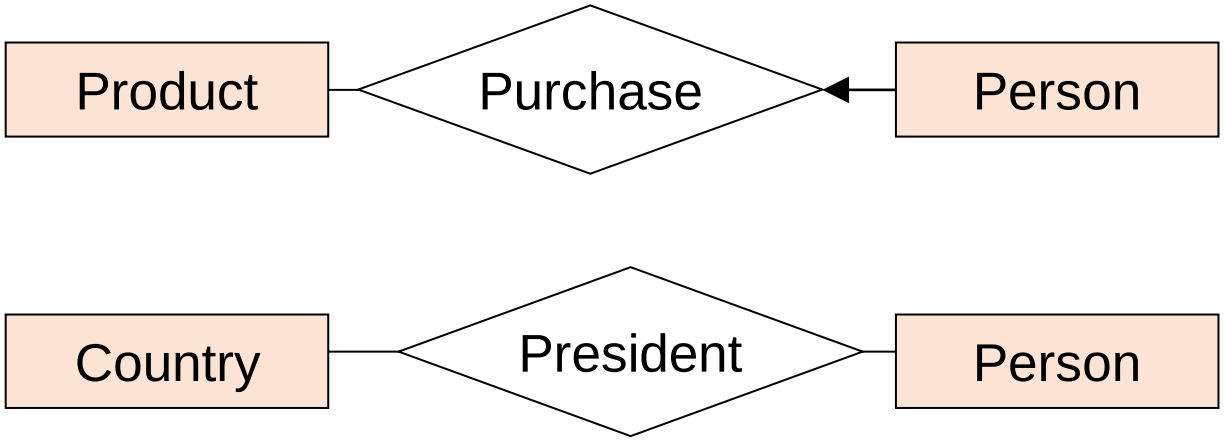
Each product made by at most one company.
Some products made by no company?



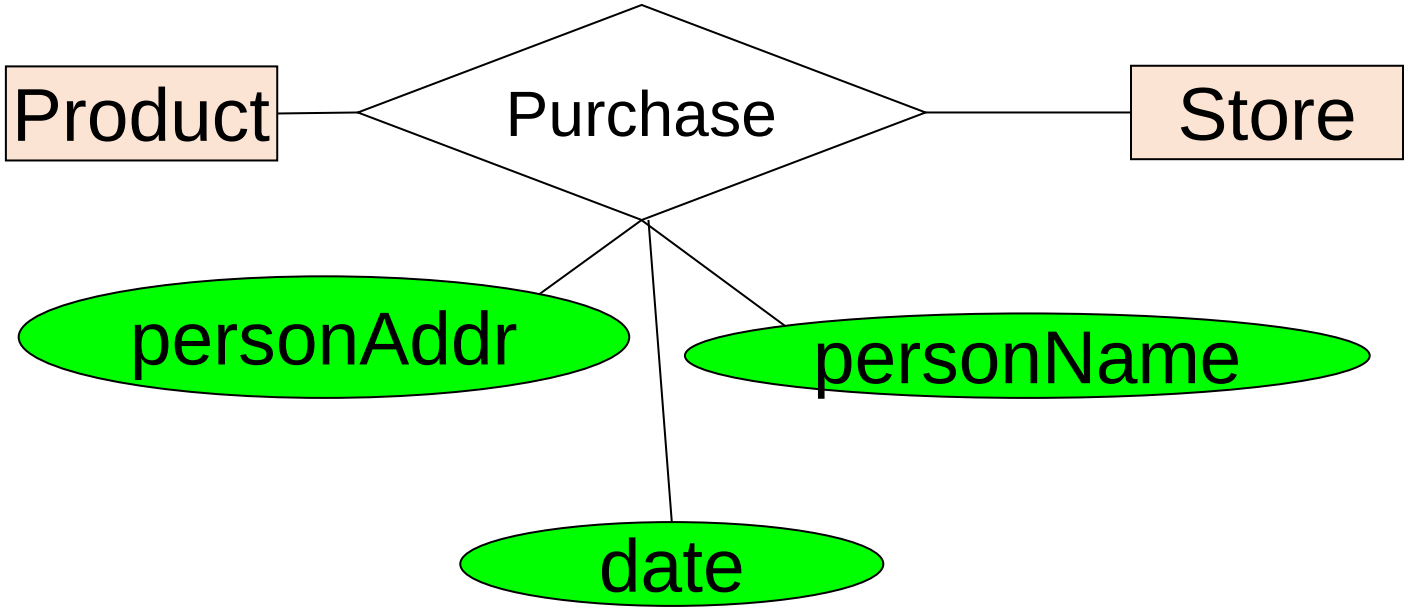
Each product made by *exactly* one company.

Design Principles

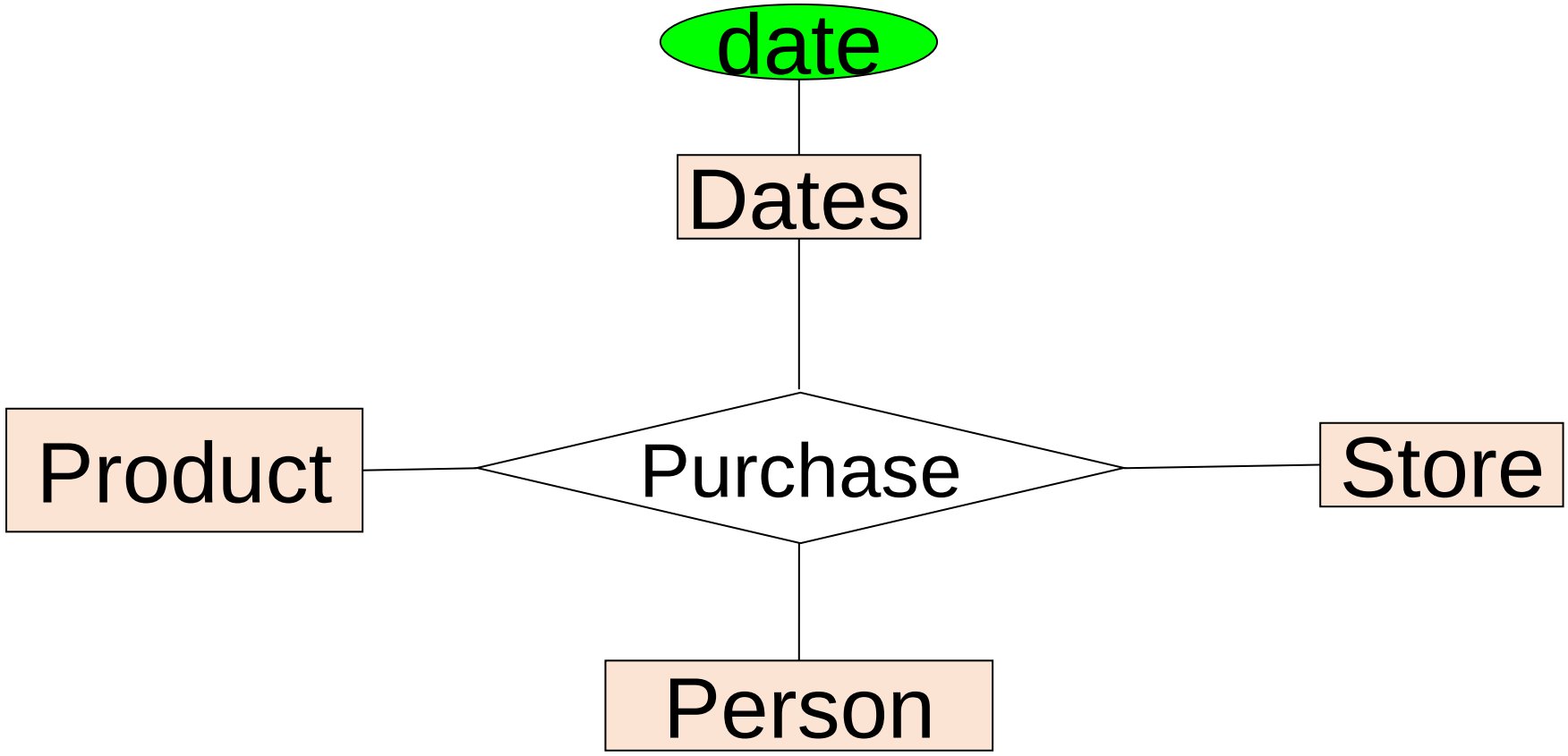
What's wrong with these examples?



Design Principles: What's Wrong?

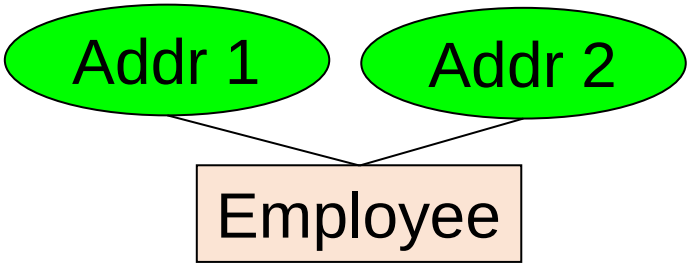


Design Principles: What's Wrong?

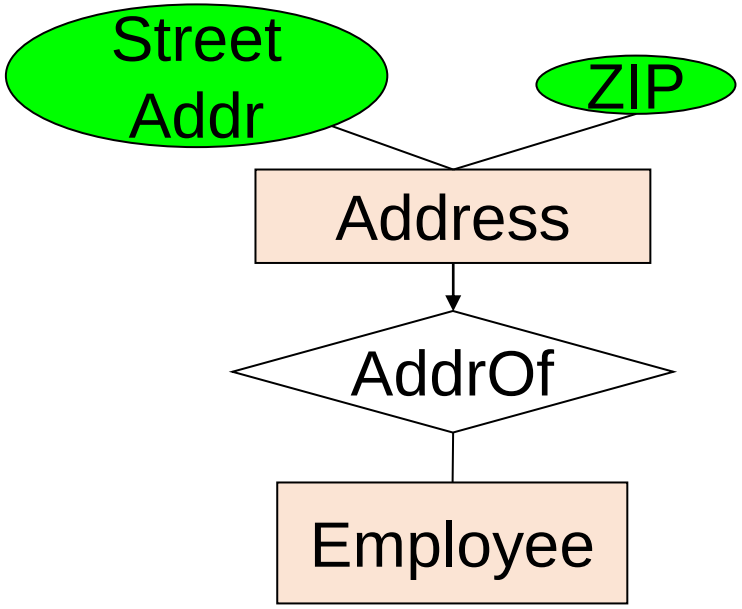


Examples: Entity vs. Attribute

Should address (A) be an attribute?

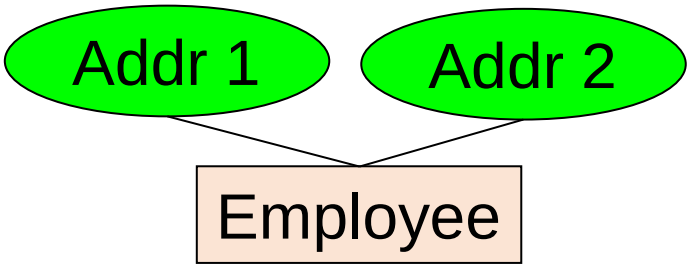


Or (B) be an entity?



Examples: Entity vs. Attribute

Should address (A) be an attribute?

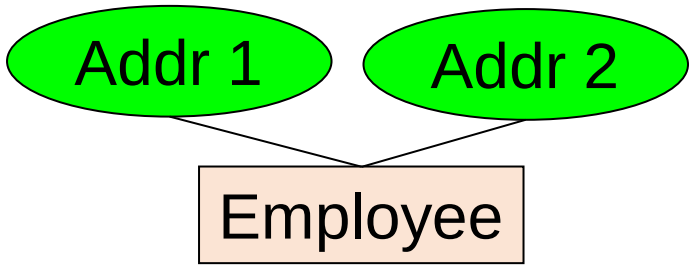


How do we handle employees with multiple addresses here?

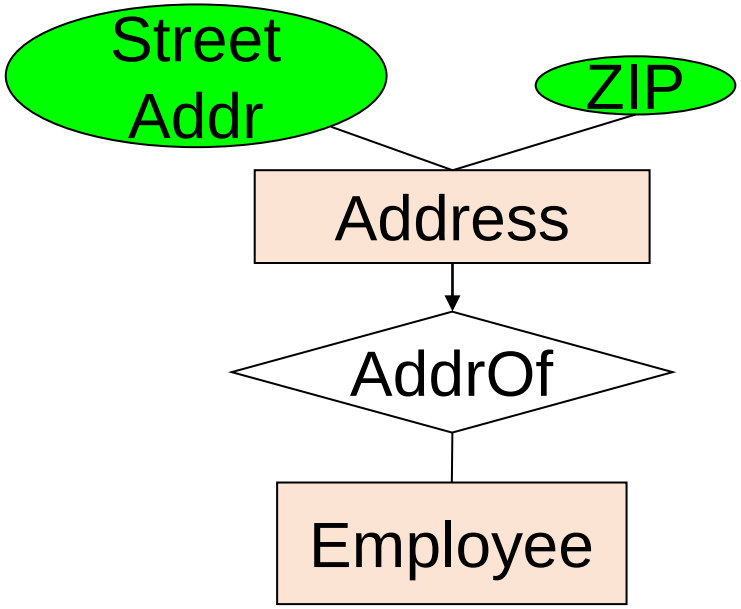
How do we handle addresses where internal structure of the address (e.g. zip code, state) is useful?

Examples: Entity vs. Attribute

Should address (A) be an attribute?



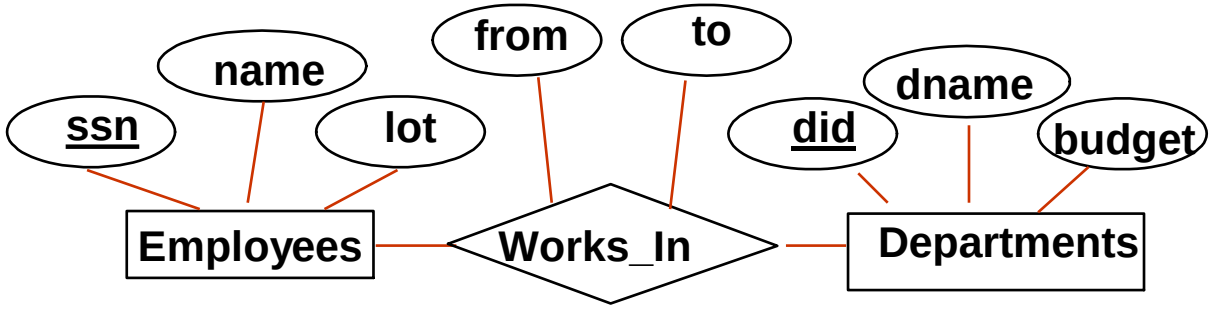
Or (B) be an entity?



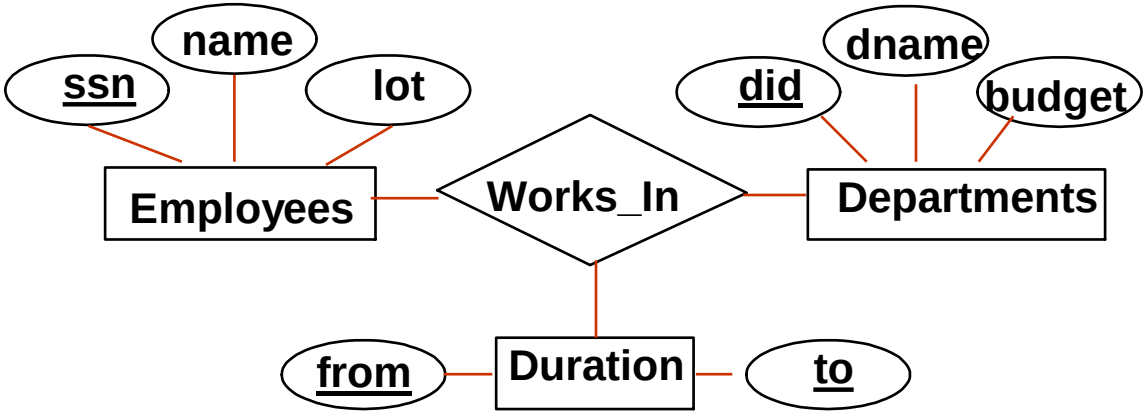
In general, when we want to record several values, we choose new entity

Entity vs. Attribute?

- Works_In does not allow an employee to work in a department for two or more periods.

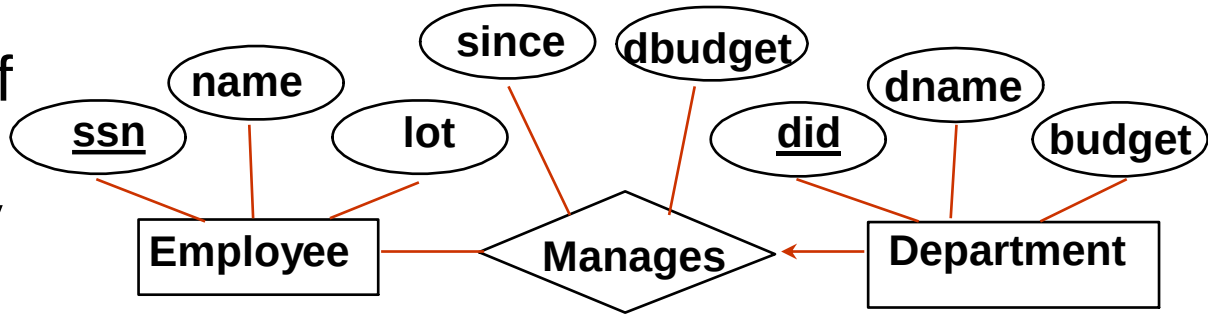


- Similar to the problem of wanting to record several addresses for an employee: We want to record *several values of the descriptive attributes for each instance of this relationship.* Accomplished by introducing new entity set, Duration.

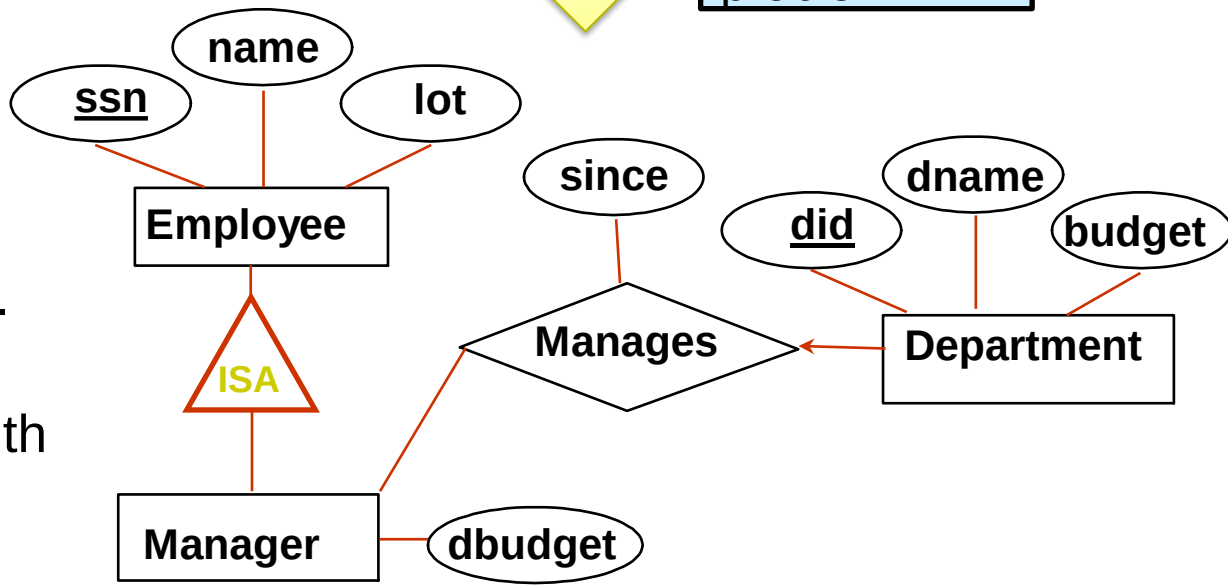


Entity vs. Relationship

- First ER diagram OK if a manager gets a separate discretionary budget for each dept.
- What if a manager gets a discretionary budget that covers *all* managed depts?
  - Redundancy: *dbudget* stored for each dept managed by manager.
  - Misleading: Suggests *dbudget* associated with department-mgr combination.



This fixes the problem!

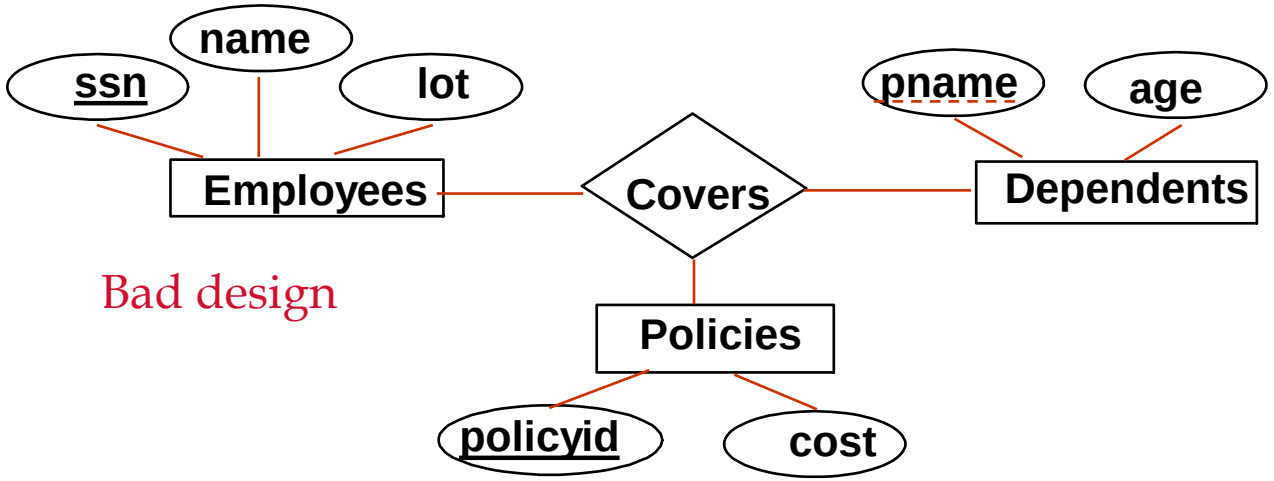


Binary vs. Ternary Relationships

- An example of a case a case when one ternary relationship is better than multiple binary relationships:
  - a ternary relation Contracts relates entity sets Parts, Departments, and Suppliers, and has descriptive attribute *qty*.
  - No combination of binary relationships is an adequate substitute:
    - S “can-supply” P,
    - D “needs” P, and
    - D “deals-with” S does not imply that D has agreed to buy P from S.
  - How do we record *qty*?

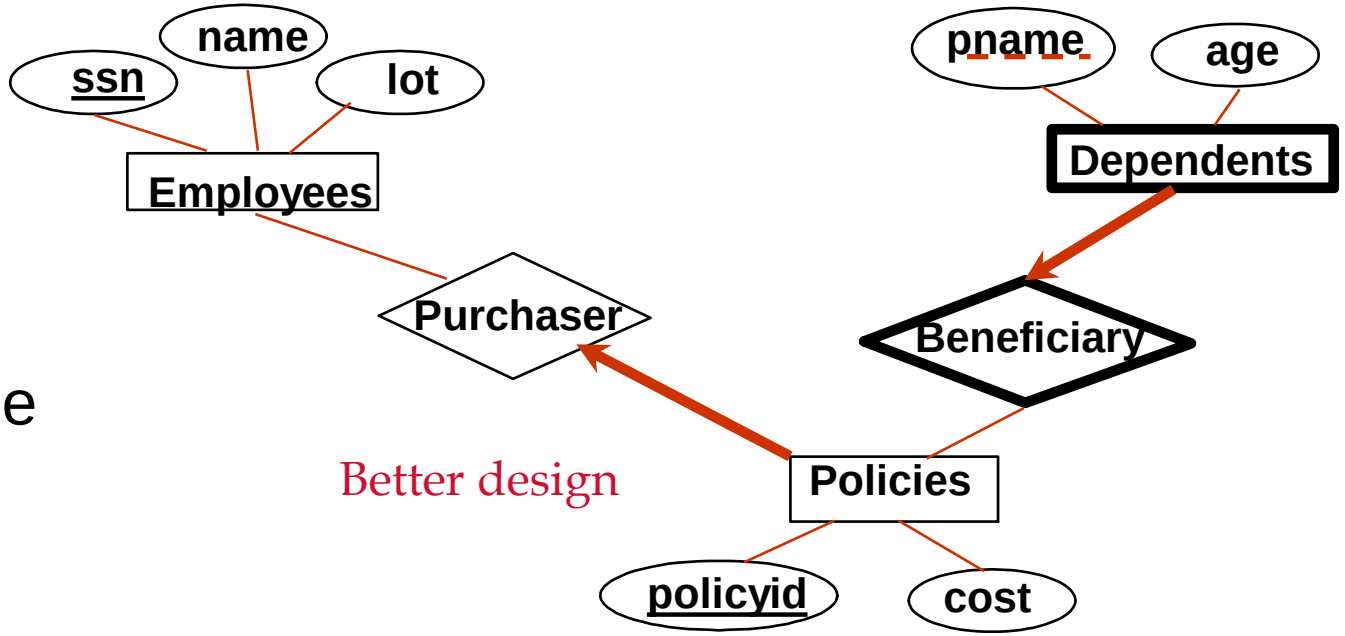
Binary vs. Ternary Relationships

- If each policy is owned by just 1 employee, and each dependent is tied to the covering policy, first diagram is inaccurate.



Bad design

- What are the additional constraints in the 2nd diagram?



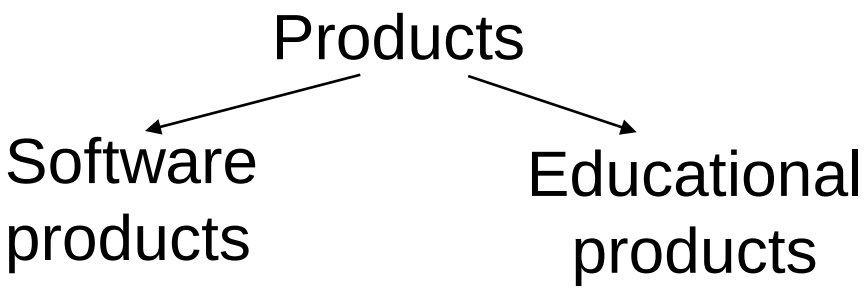
Better design

Modeling Subclasses

- Some objects in a class may be special, i.e. worthy of their own class

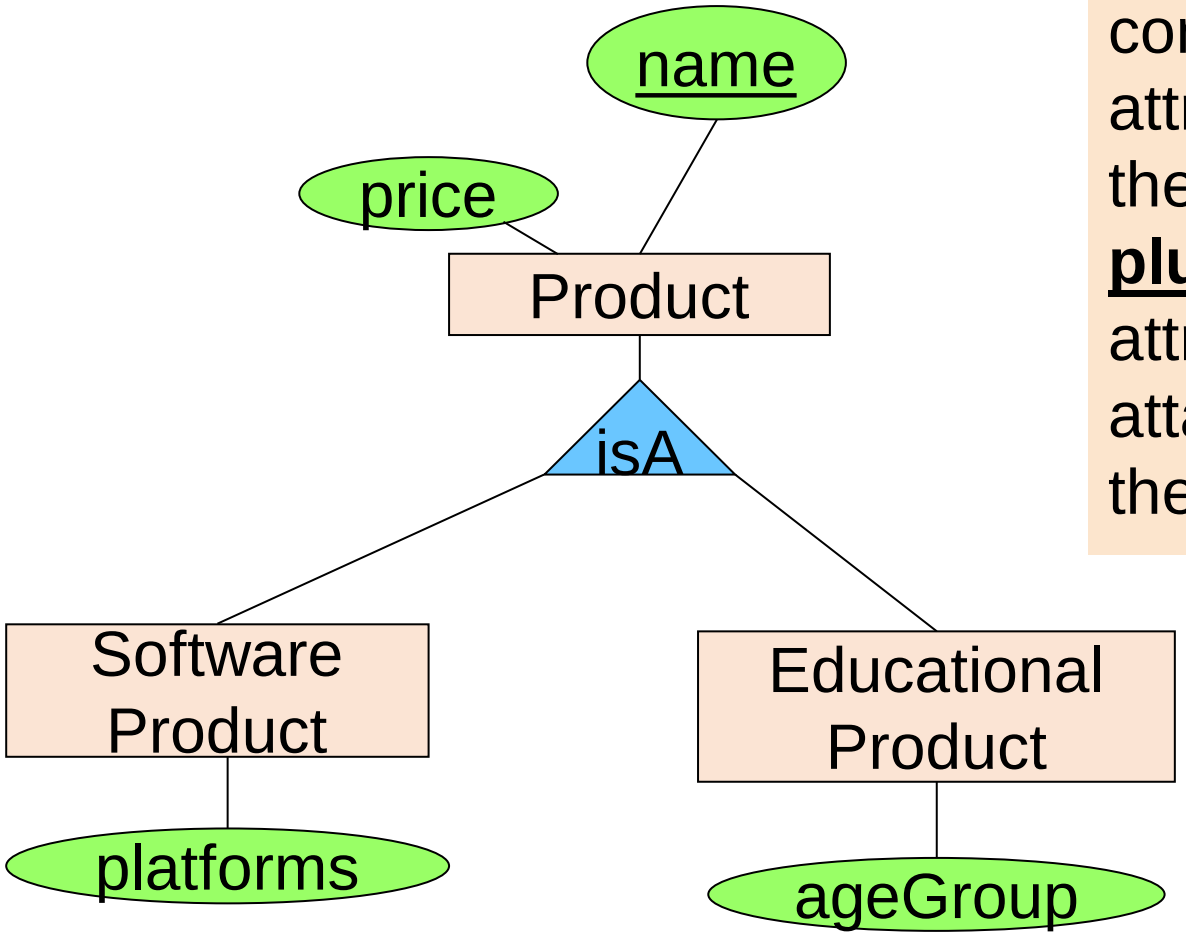
- Define a new class?
  - *But what if we want to maintain connection to current class?*

- Better: define a *subclass*
  - *Ex:*



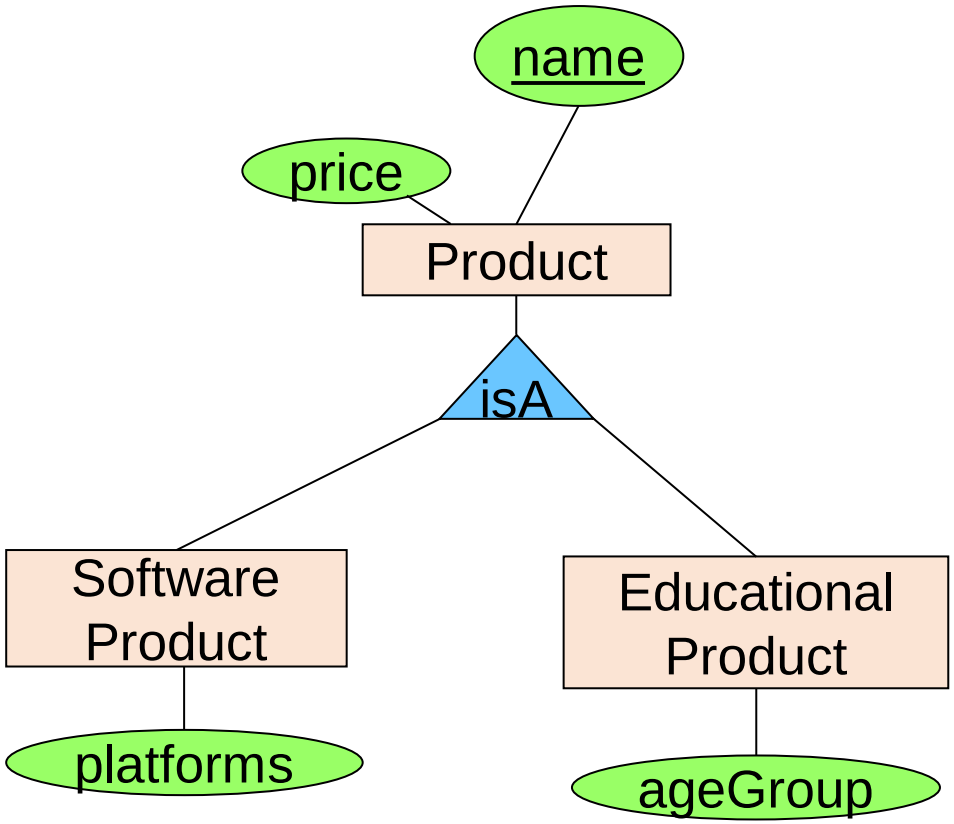
We can define **subclasses** in E/R!

Modeling Subclasses



Child subclasses contain all the attributes of *all* of their parent classes **plus** the new attributes shown attached to them in the E/R diagram

Understanding Subclasses



- Think in terms of objects; ex:

Product

name
price

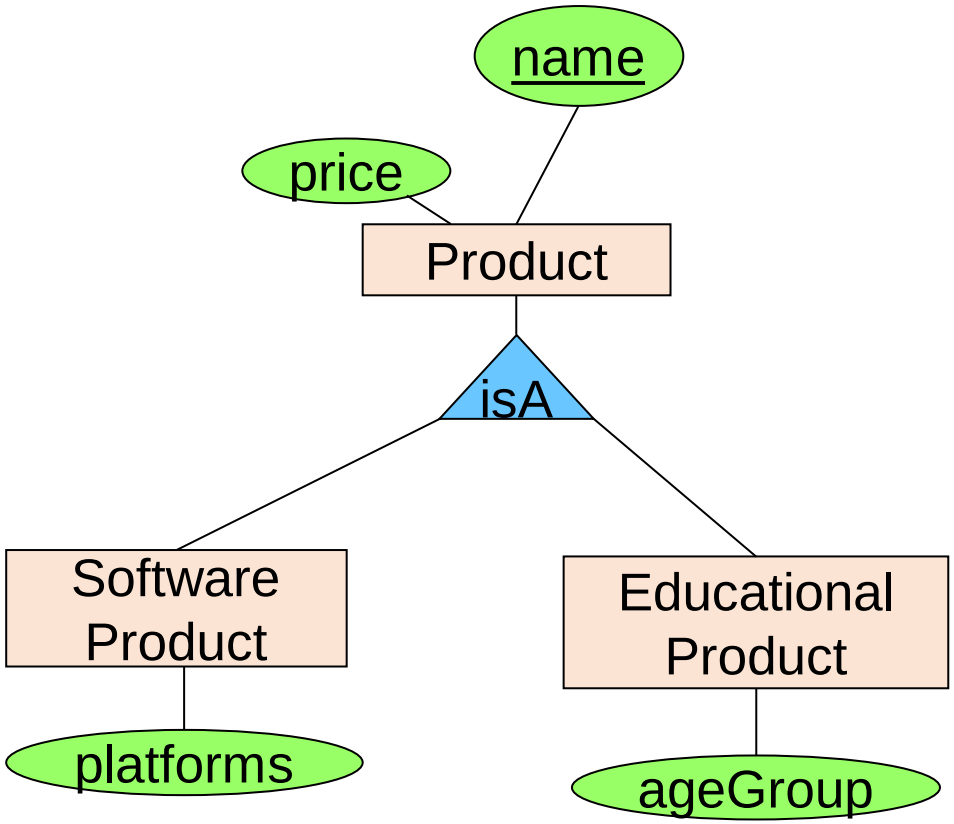
SoftwareProduct

name
price
platforms

EducationalProduct

name
price
ageGroup

Think like tables…



Product

name	price
Gizmo	99
Camera	49
Toy	39

Sw.Product

name	platforms
Gizmo	unix

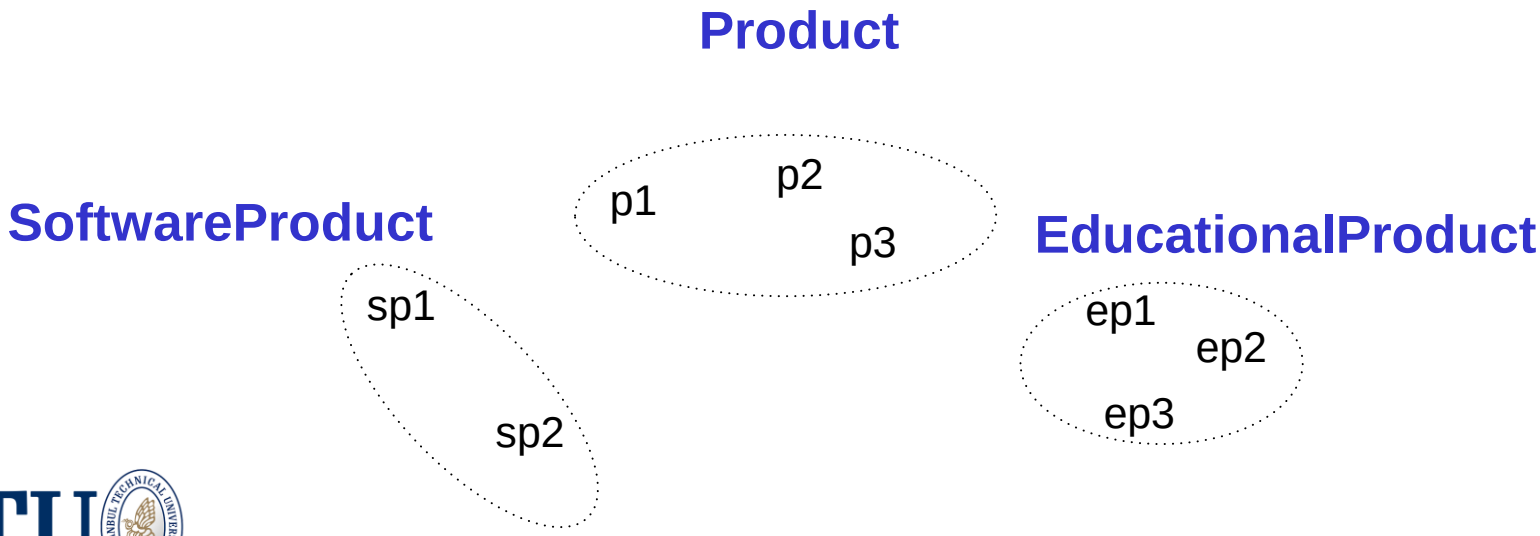
Ed.Product

name	ageGroup
Gizmo	toddler
Toy	retired

Difference between OO and E/R inheritance

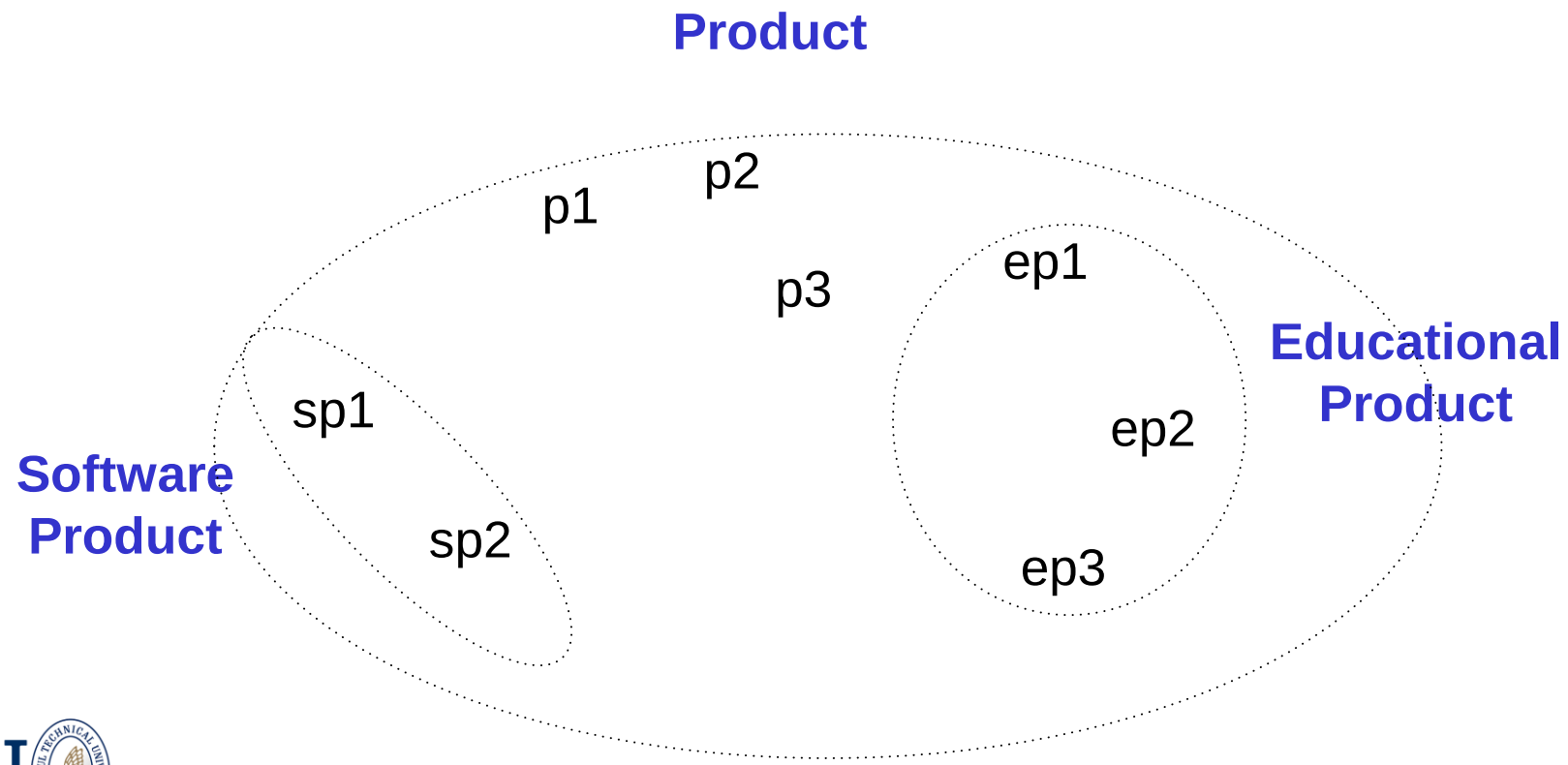
- OO: Classes are disjoint

OO = **Object Oriented**. E.g. classes as fundamental building block, etc…



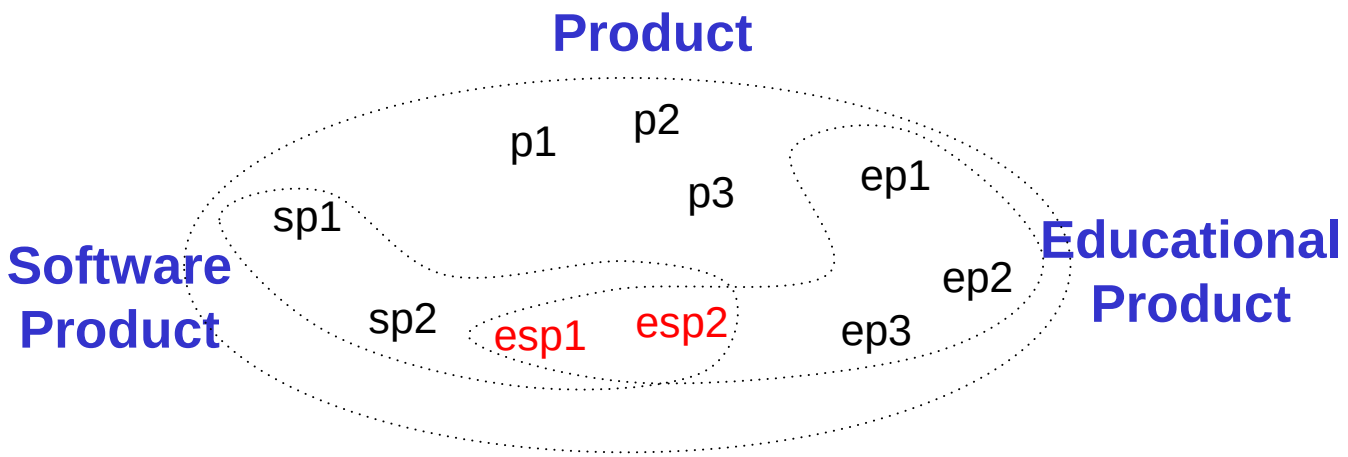
Difference between OO and E/R inheritance

- E/R: entity sets overlap



Difference between OO and E/R inheritance

We have three entity sets, but four different kinds of objects



No need for multiple inheritance in E/R

Modeling UnionTypes With Subclasses

Person

FurniturePiece

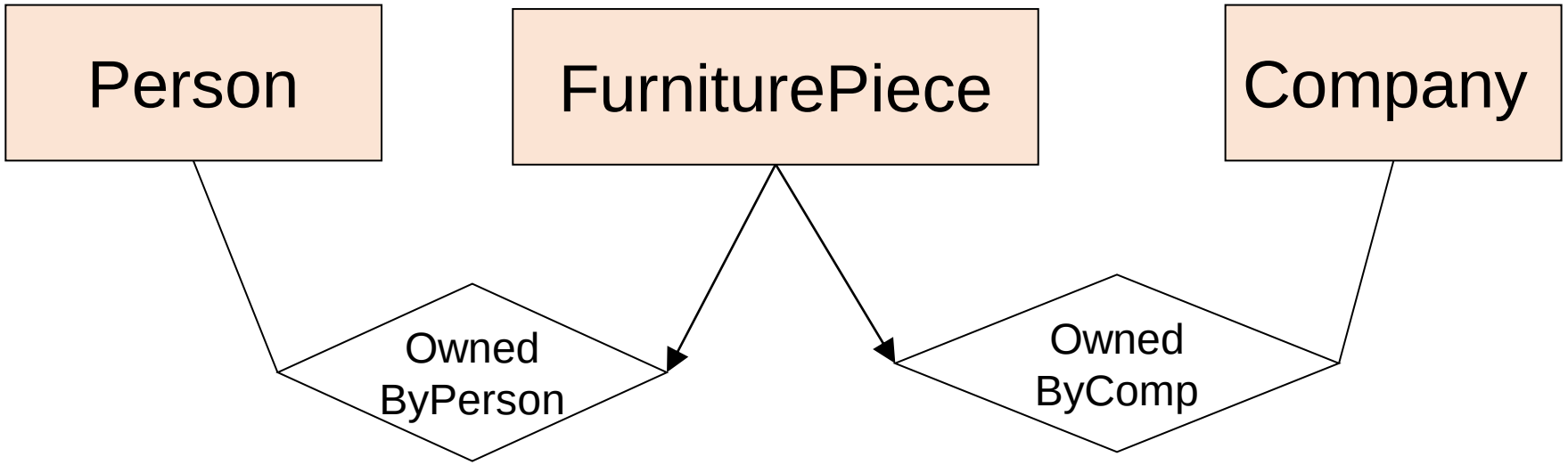
Company

Suppose each piece of furniture is owned either by a person, or by a company. *How do we represent this?*

Modeling Union Types with Subclasses

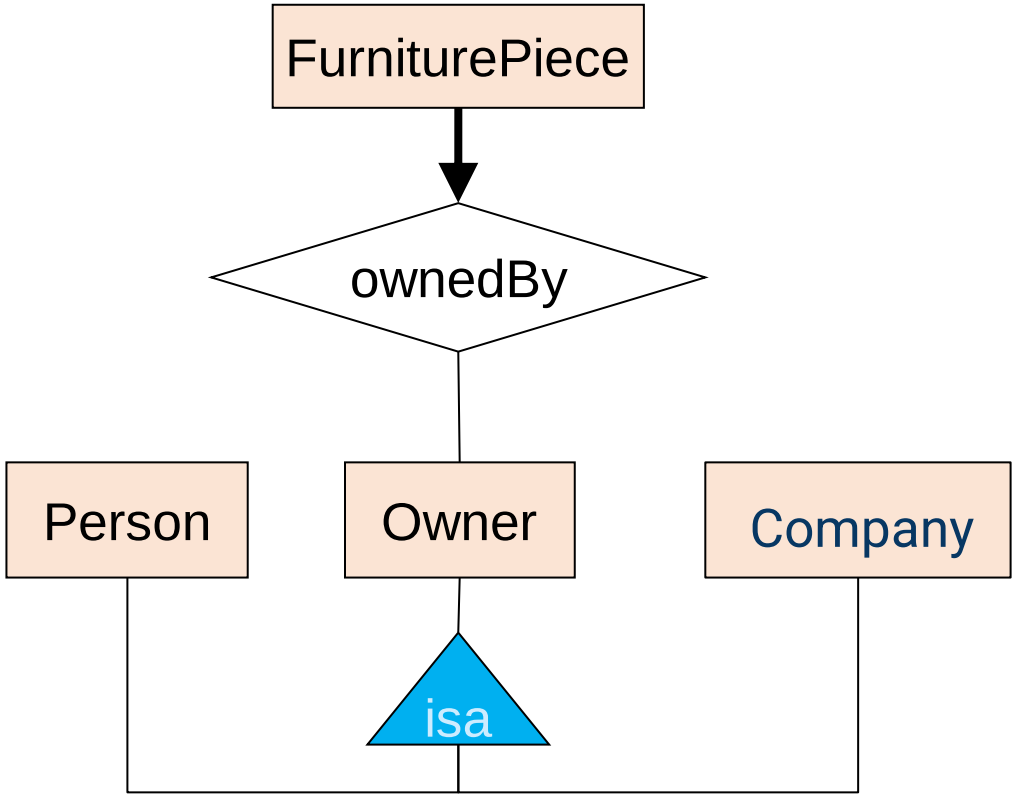
Say: each piece of furniture is owned either by a person, or by a company

Solution 1. Acceptable, but imperfect (What's wrong ?)



Modeling Union Types with Subclasses

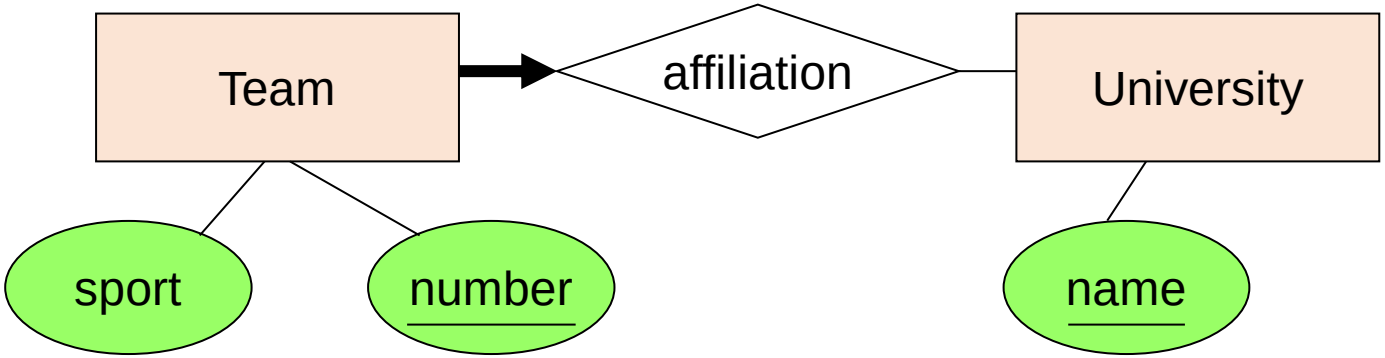
Solution 2: better (though more laborious)



What is happening here?

Weak Entity Sets

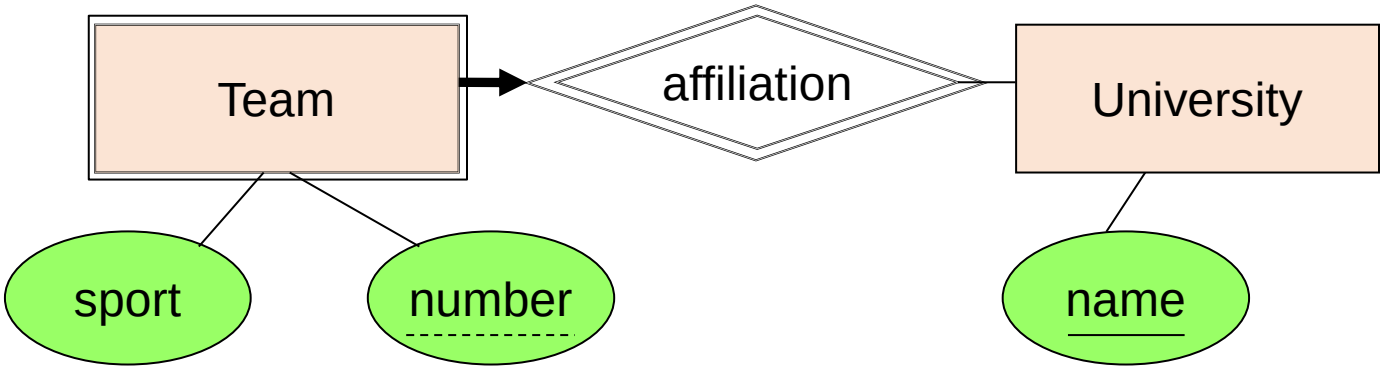
Entity sets are *weak* when their existence depends on other entities to which they are related.



"*The **ITU** Football team"*
(e.g., Stanford has a football team too)

Weak Entity Sets

Entity sets are *weak* when their existence depends on other entities to which they are related.



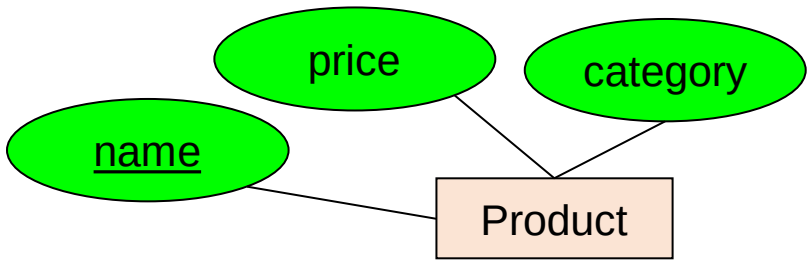
- number is a *partial key*. (denote with dashed underline).
- University is called the *identifying owner*.
- Participation in affiliation must be total. Why?

E/R Summary

- E/R diagrams are a visual syntax that allows technical and non-technical people to talk
  - For conceptual design
- Basic constructs: **entity**, **relationship**, and **attributes**
- A good design is faithful to the constraints of the application, but not overzealous

From E/R Diagrams to Relational Schema

- Key concept:
  - Both Entity sets and Relationships become relations (tables in RDBMS)
- An entity set becomes a relation (multiset of tuples / table)
  - Each tuple is one entity
  - Each tuple is composed of the entity's attributes, and has the same primary key



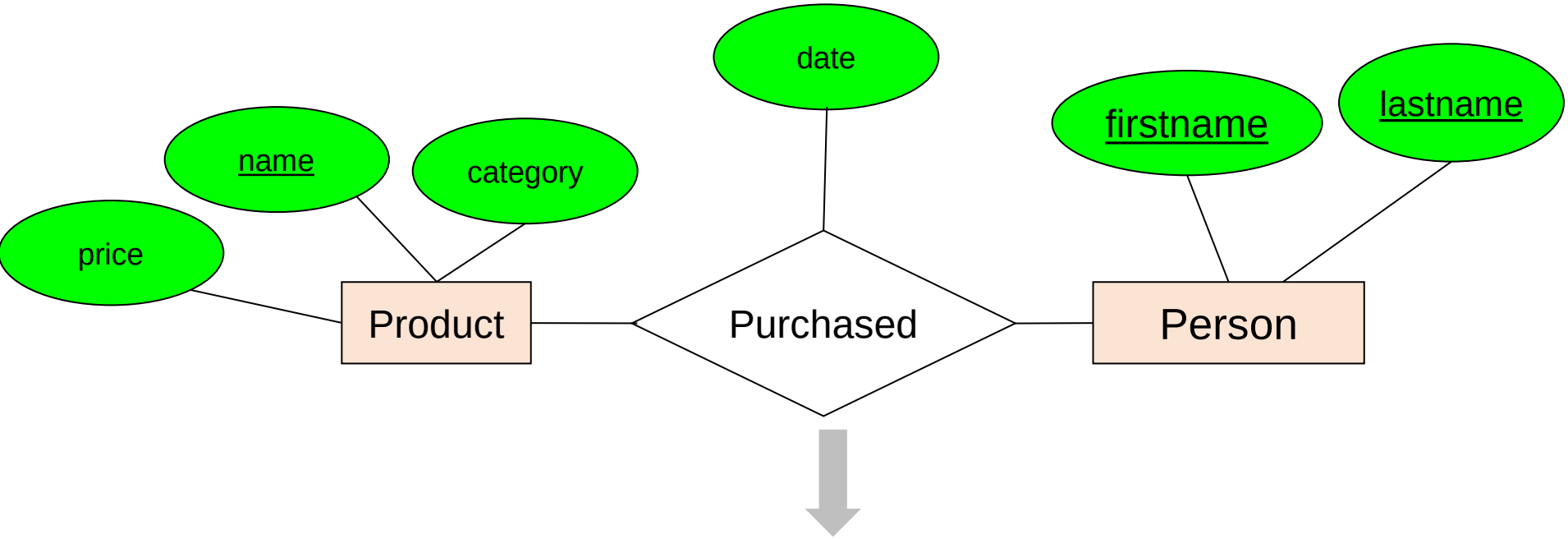
Product

name	price	category
Gizmo1	99.99	Camera
Gizmo2	19.99	Edible

From E/R Diagrams to Relational Schema

- A many-to-many relation between entity sets A_1, …, A_N *also* becomes a multiset of tuples / a table
  - Each row/tuple is one relation, i.e., one unique combination of entities (a_1,…,a_N)
  - Each row/tuple is
    - composed of the **union of the entity sets' keys + relationship attributes**
    - has the entities' primary keys as foreign keys
    - has the union of the entity sets' keys as primary key

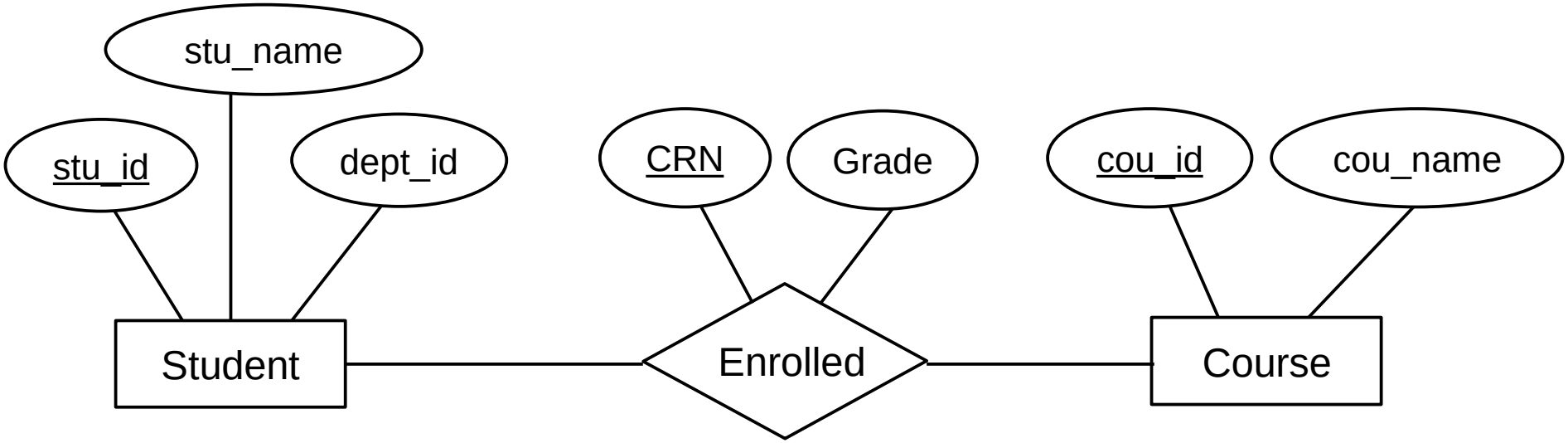
From E/R Diagrams to Relational Schema



Purchased

name	firstname	lastname	date
Gizmo1	Bob	Joe	01/01/15
Gizmo2	Joe	Bob	01/03/15
Gizmo1	JoeBob	Smith	01/05/15

From E/R Diagrams to Relational Schema



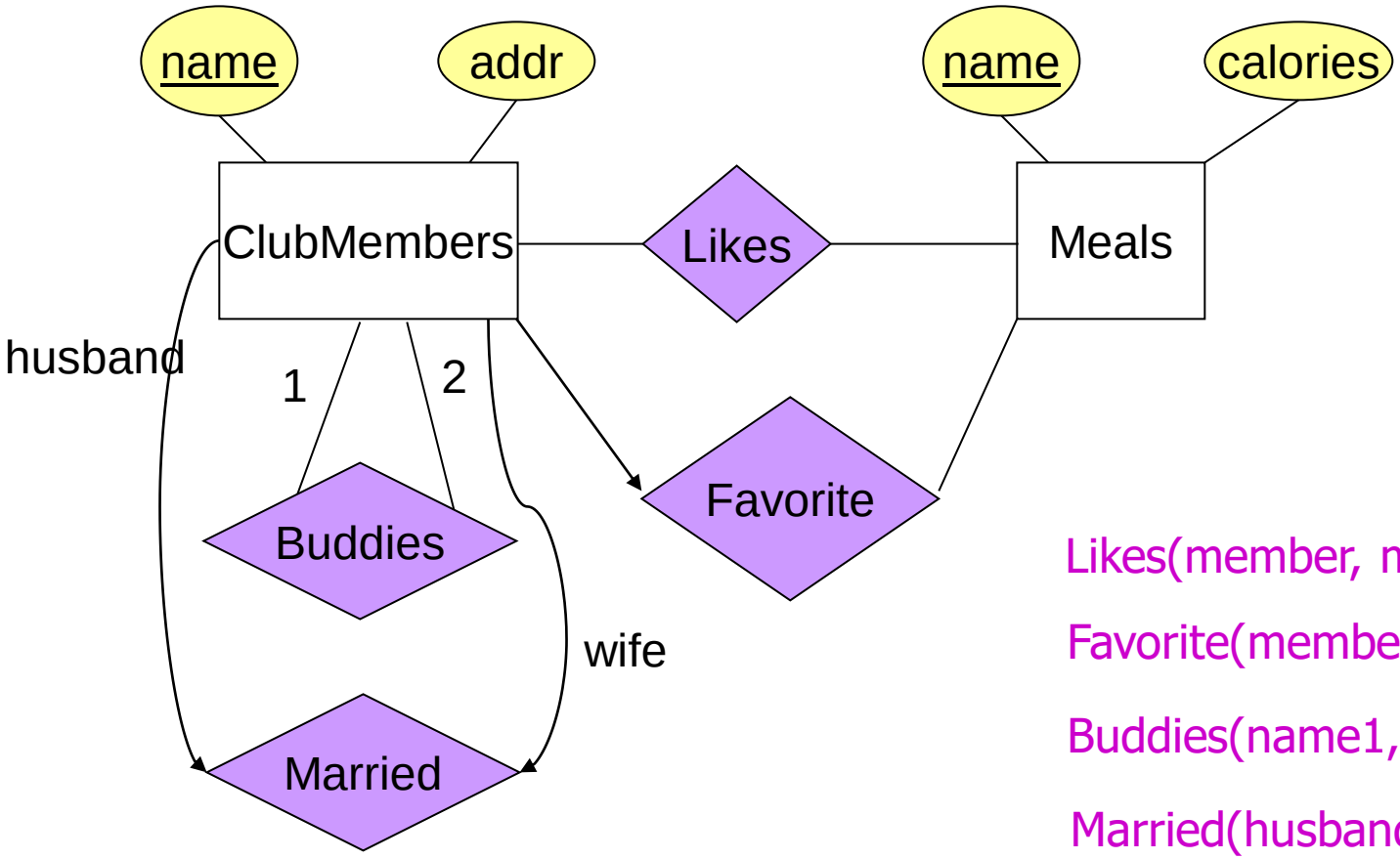
Database Schema:

```
Student (stu_id, stu_name, dept_id)

Course (cou_id, cou_name)

Enrolled (stu_id, cou_id, CRN, Grade)
```

From E/R Diagrams to Relational Schema



Likes(member, meal)

Favorite(member, meal)

Buddies(name1, name2)

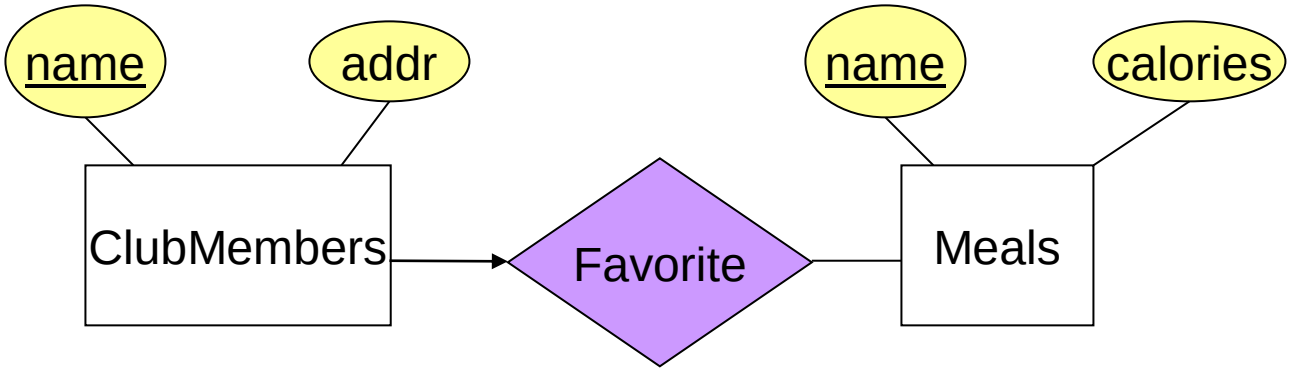
Married(husband, wife)

ClubMembers(name, addr)

Meals(name, calories)

Mapping Many-to-One Relations

- OK to combine into one relation:
  - The relation for an entity-set *E*
  - The relations for many-to-one relationships from *E* ("many") to *F*
- Example: ClubMembers(name, addr) and Favorite(member, meal) combine to make ClubMembers(name, addr, favMeal).

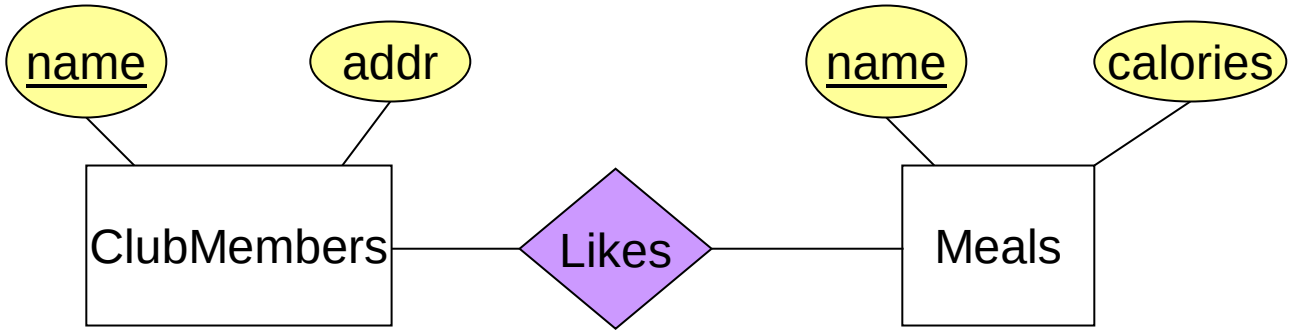


Mapping Many-to-One Relations

- The combined relation schema consists of
  - All attributes of *E*
  - The key attributes of *F*
  - Any attributes belonging to the relationship *R*
- Can we combine one-one relationship?
- What about many-many?

Risk with Many-Many Relationships

- Combining ClubMembers with Likes would be a mistake. It leads to redundancy, as:

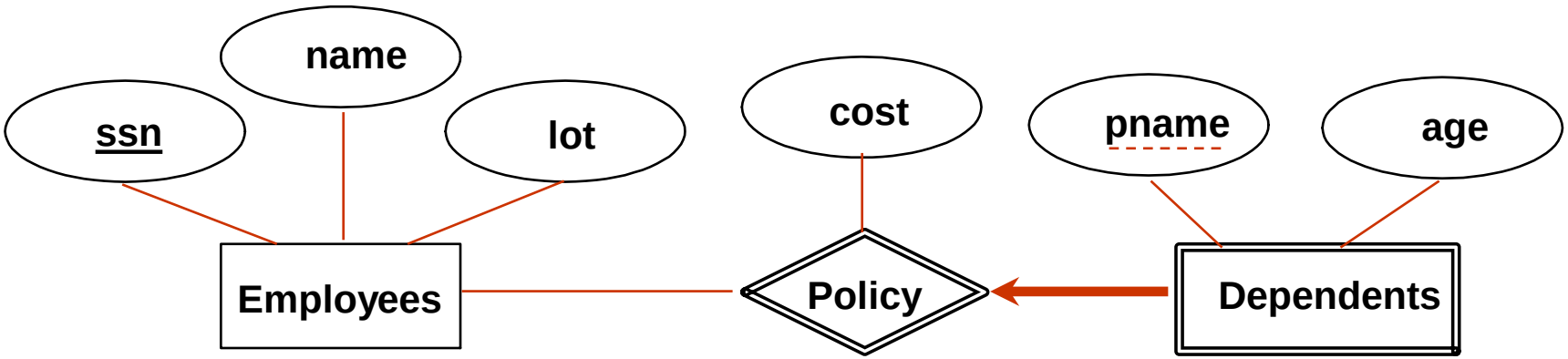


name	addr	meal
Sally	123 Maple	Kofte
Sally	123 Maple	Manti

Redundancy

Review: Weak Entities

- A *weak entity* can be identified uniquely only by considering the key of another (*owner*) entity.
  - Owner entity set and weak entity set must participate in a one-to-many relationship set (1 owner, many weak entities).
  - Weak entity set must have total participation in this *identifying* relationship set.

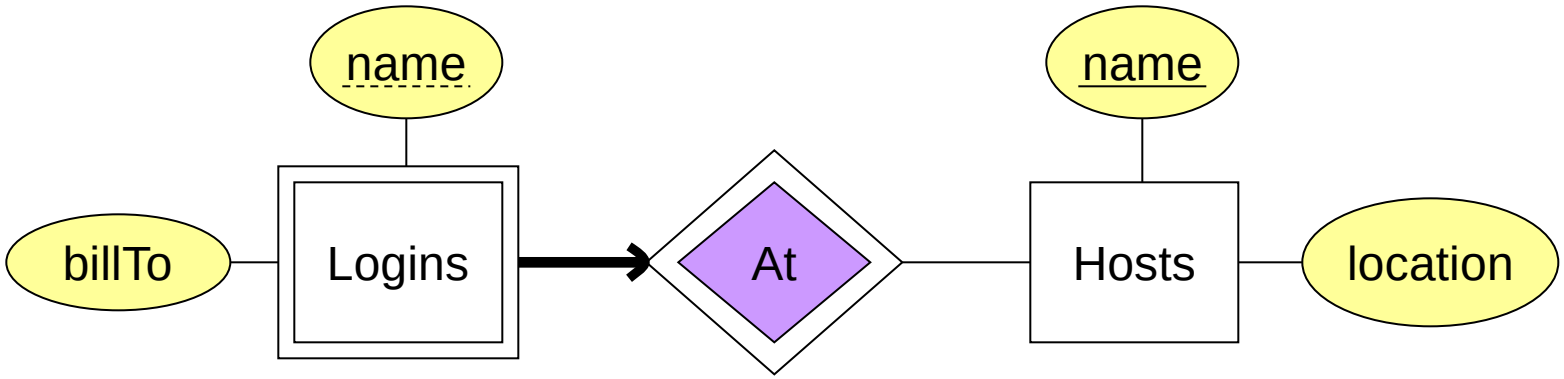


Translating Weak Entity Sets

- A weak entity and its identifying relationship are translated into a single table.
  - The table also includes the key attribute(s) of the owner entity.
  - When the owner entity is deleted, all owned weak entities must also be deleted.

```
CREATE TABLE  Dep_Policy (
   pname  CHAR(20),
   age  INTEGER,
   cost  REAL,
   ssn  CHAR(11) NOT NULL,
   PRIMARY KEY  (pname, ssn),
   FOREIGN KEY  (ssn) REFERENCES Employees
      ON DELETE CASCADE);
```

Example



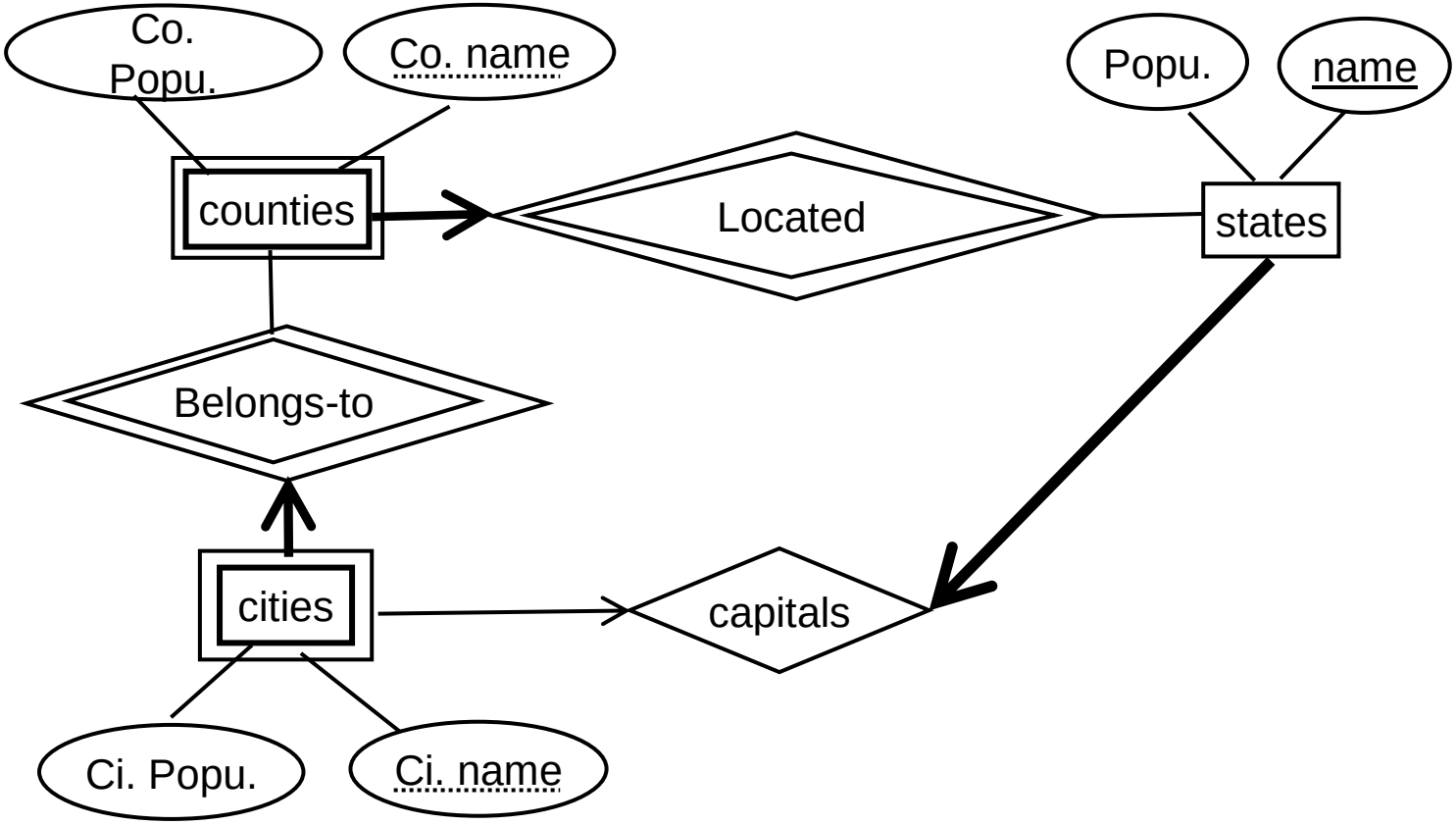
Hosts(hostName, location)
Logins(loginName, hostName, billTo)
~~At(loginName, hostName, hostName2)~~

Must be the same

At becomes part of Logins

What if “At” has some attributes ?

Case Study


Co.
Popu.
Co. name
Popu.
name
counties
Located
states
Belongs-to
cities
capitals
Ci. Popu.
Ci. name

Sample Solution

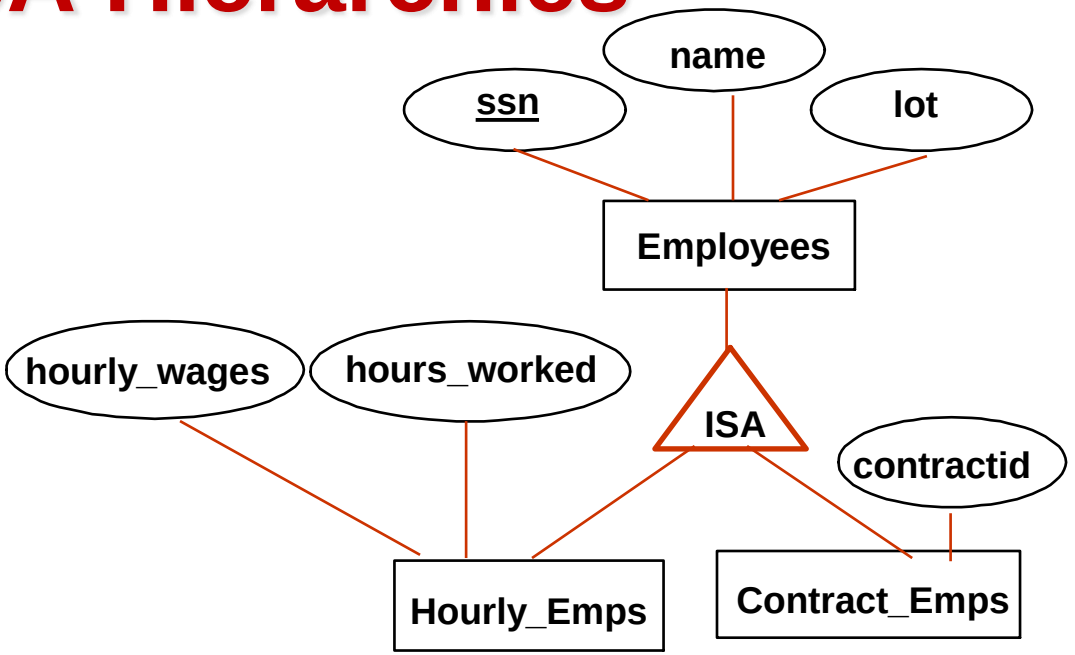
- States (name, popu, capital_city, capital_co)
- Counties (co_name, state_name, co_popu)
- Cities (ci name, co_name, state_name, ci_popu)

Alternatively:

- States (name, popu)
- Counties (co_name, state_name, co_popu)
- Cities (ci name, co_name, state_name, ci_popu, is_capital)

Review: ISA Hierarchies

- As in Python, or other PLs, attributes are inherited.
- If we declare A **ISA** B, every A entity is also considered to be a B entity.



- Overlap constraints: Can Joe be an Hourly_Emps as well as a Contract_Emps entity? (Allowed/disallowed)
- Covering constraints: Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity? (Yes/no)

Translating ISA Hierarchies to Relations

- **General approach*:***
  - 3 relations: Employees, Hourly_Emps and Contract_Emps.
    - Hourly_Emps:
      - Every employee is recorded in Employees.
      - For hourly emps, extra info recorded in Hourly_Emps (hourly_wages, hours_worked, ssn);
      - Must delete Hourly_Emps tuple if referenced Employees tuple is deleted).
    - Contract_Emps is similar.
    - Queries
      - Listing all employees is easy (Select * From Employee)
      - Those involving just Hourly_Emps or Contract_Emps require a join to get some attributes.

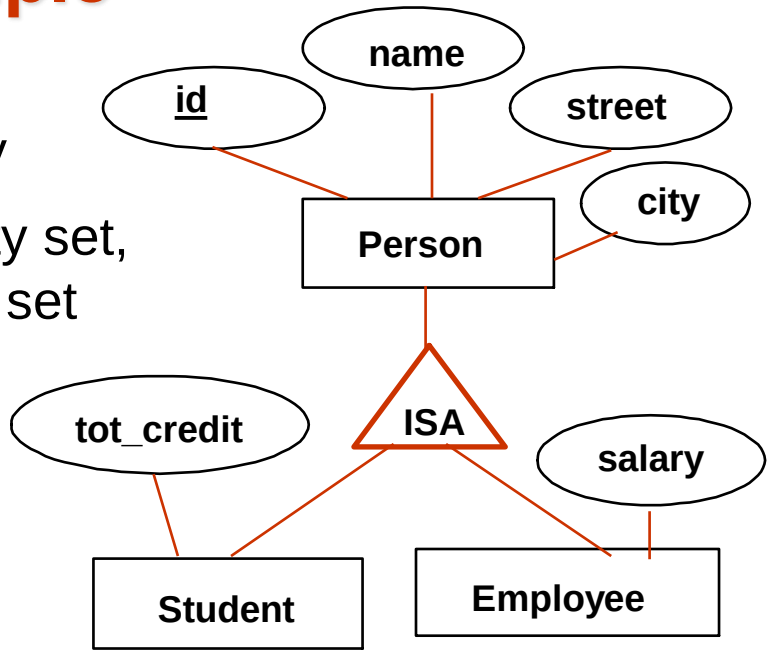
Translating ISA Hierarchies to Relations

- **Alternative Approach**: Just Hourly_Emps and Contract_Emps tables.
  - Hourly_Emps: ssn, name, lot, hourly_wages, hours_worked.
  - Contract_Emps: ssn, name, lot, contract_id.
  - Each employee must be in one of these two subclasses.
  - Queries
    - Listing all employees is not that easy
      - (Take a union of Hourly and Contract employees)
    - Those involving just Hourly_Emps or Contract_Emps is easy
      - Select * From Hourly_Emps;

Translating ISA Hierarchies to Relations Another Example

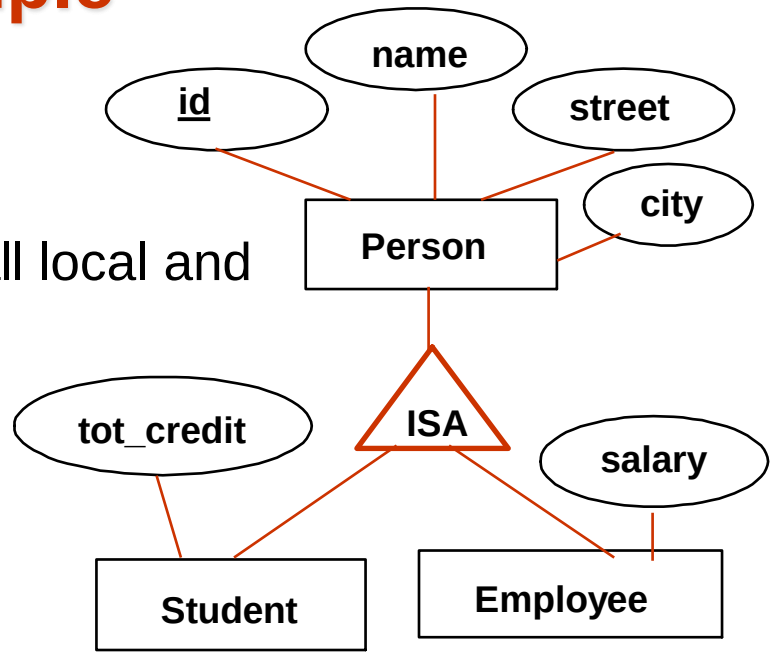
- Method 1:
  - Form a schema for the higher-level entity
  - Form a schema for each lower-level entity set, include primary key of higher-level entity set and local attributes

schema	attributes
person	ID, name, street, city
student	ID, tot_cred
employee	ID, salary



  - Drawback: getting information about, an *employee* requires accessing two relations, the one corresponding to the low-level schema and the one corresponding to the high-level schema

Translating ISA Hierarchies to Relations Another Example

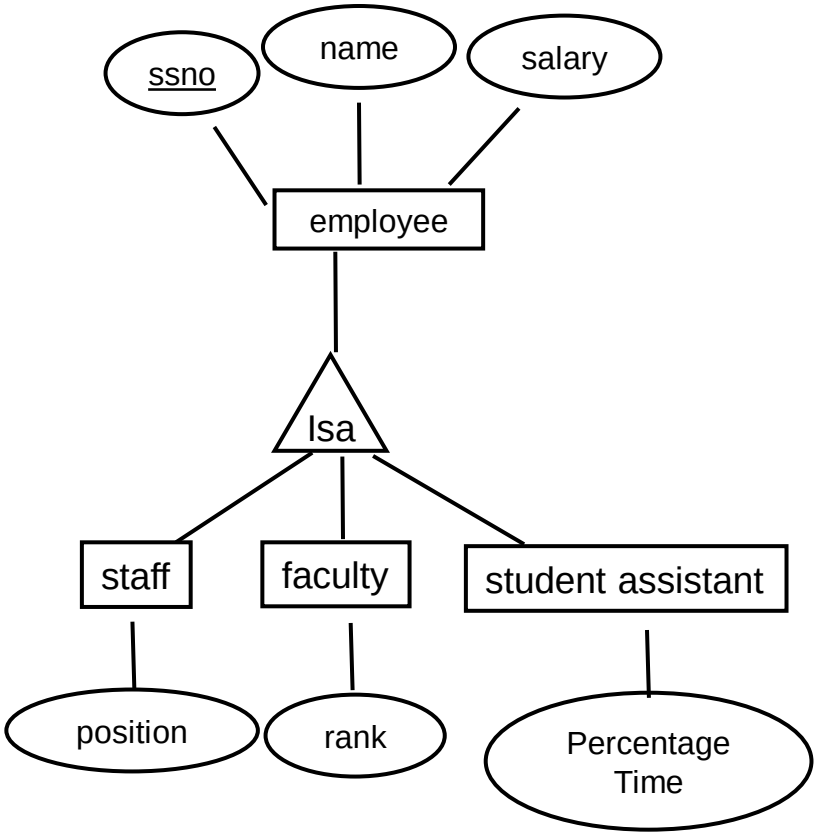


- Method 2:
  - Form a schema for each entity set with all local and inherited attributes

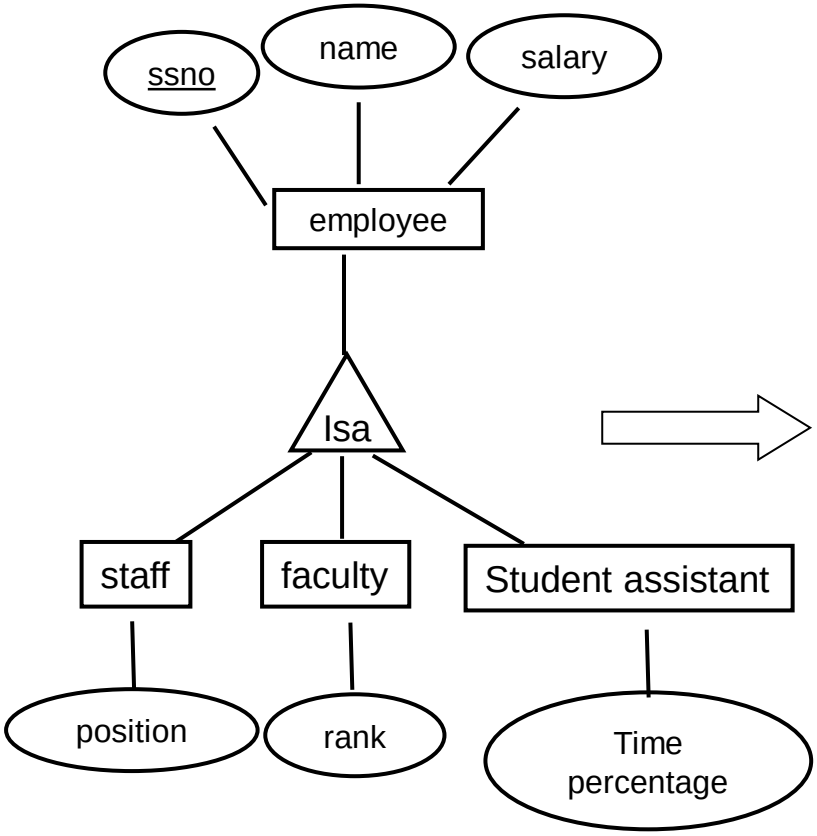
schema	attributes
person	ID, name, street, city
student	ID, name, street, city, tot_cred
employee	ID, name, street, city, salary

  - Drawback: *name, street* and *city* may be stored redundantly for people who are both students and employees

Case Study


ssno
name
salary
employee
Isa
staff
faculty
student assistant
position
rank
Percentage
Time

Subclass – Object-oriented



Relations:

employee(ssno, name, salary)

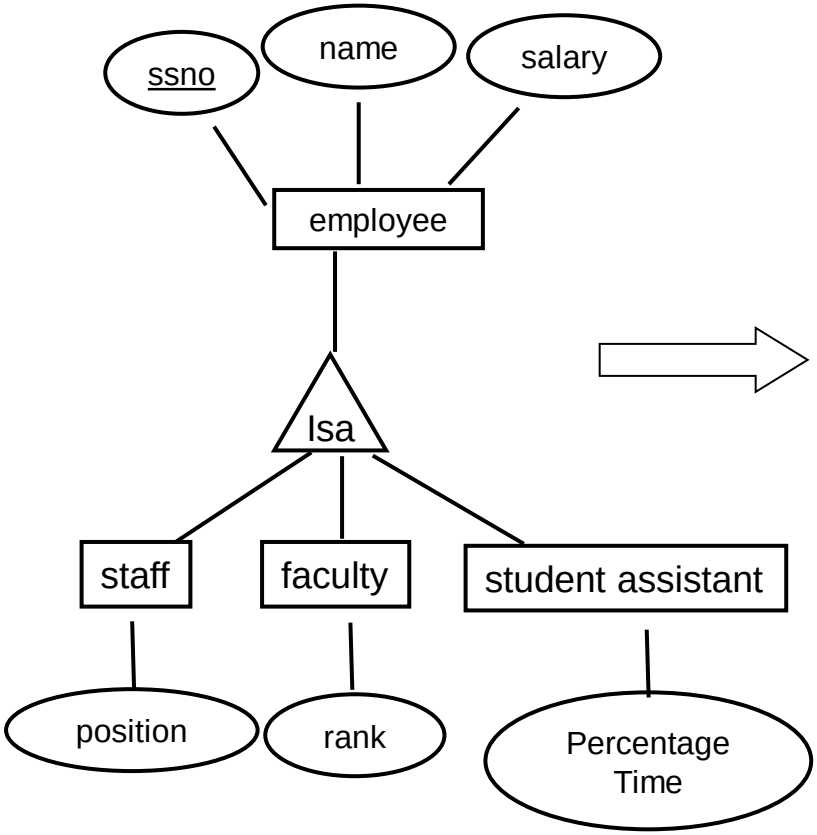
staff(ssno, name, salary,position)

faculty(ssno, name, salary, rank)

studentassistant(ssno, name, salary, percentagetime)

Key: ssno for all the relations

Subclass – E/R Style



Relations:

employee(ssno, name, salary)

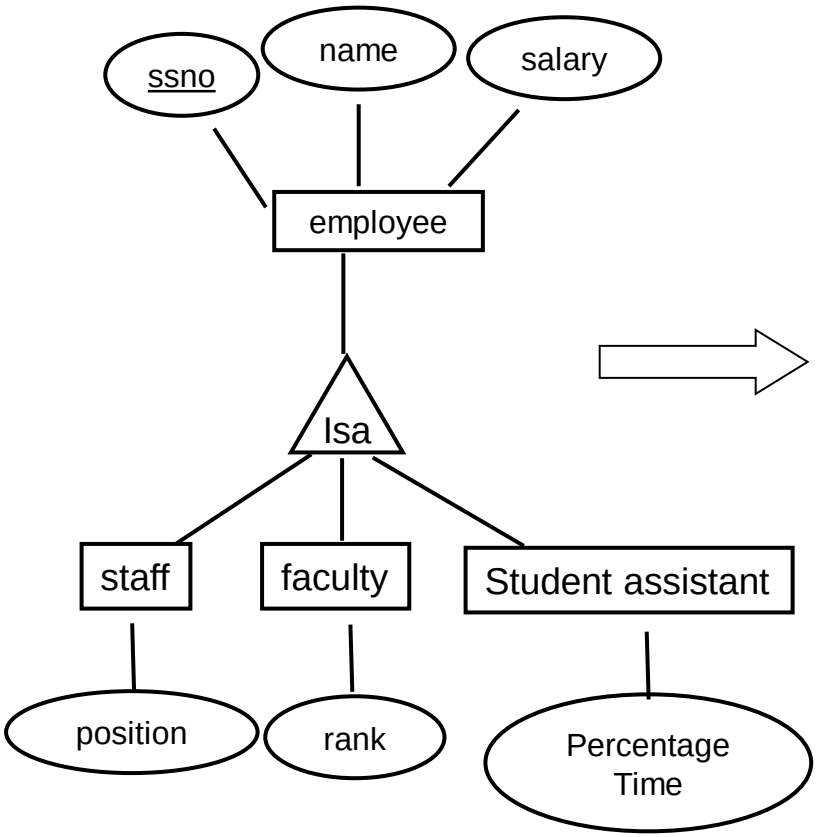
staff(ssno, position)

faculty(ssno, rank)

studentassistant(ssno, percentage_time)

Key: ssno for all relations

Subclass – null value



Relation:

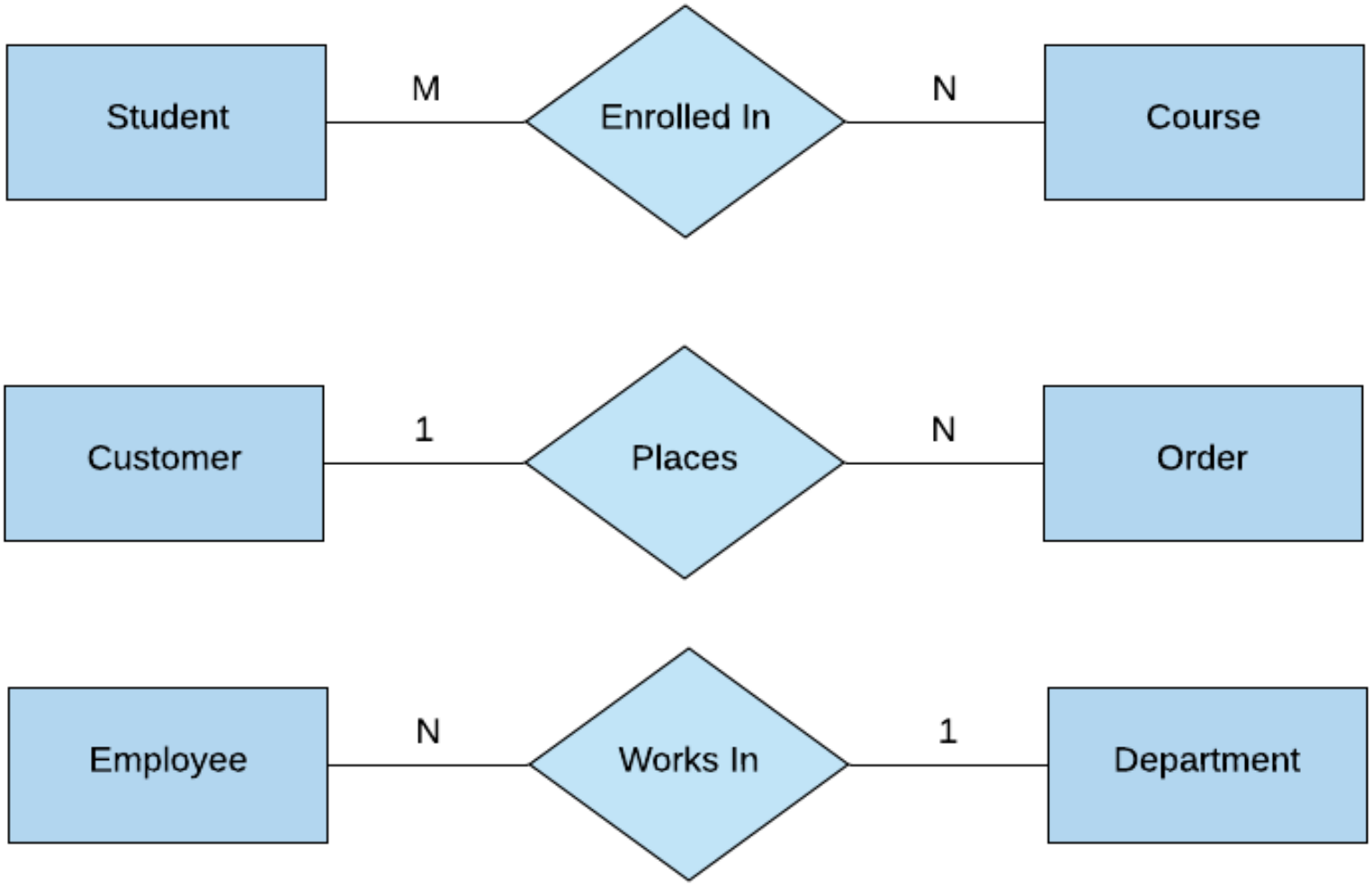
employee(ssno, name, salary, position, rank, percentage-time)

Key : ssno as key

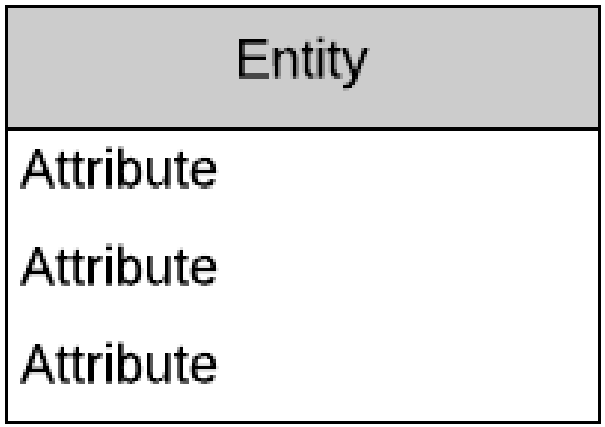
Note: Sometimes we add an attribute “jobType” to make queries easier.

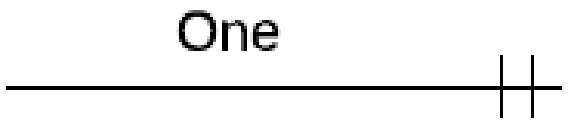
Alternative ER Representations

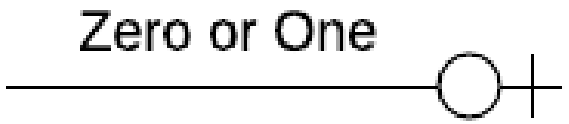
Alternative Cardinality Representation



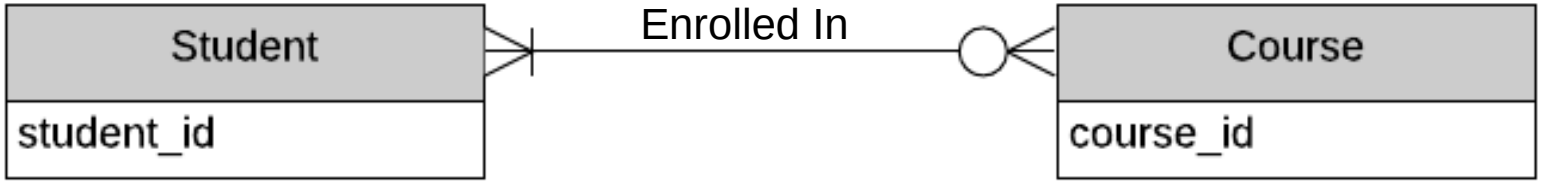
Entity-Relationship Diagram (Crow's Foot Notation)

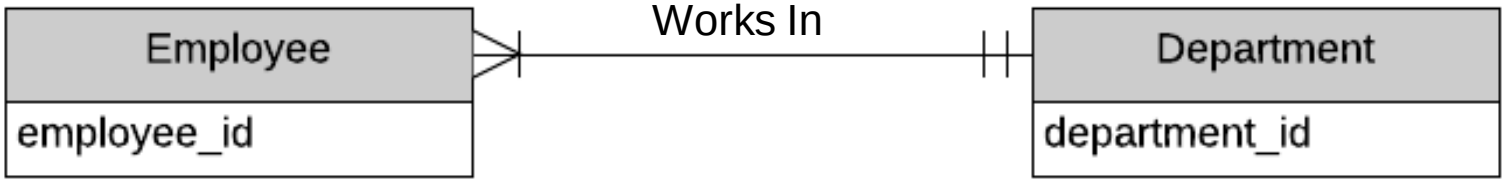






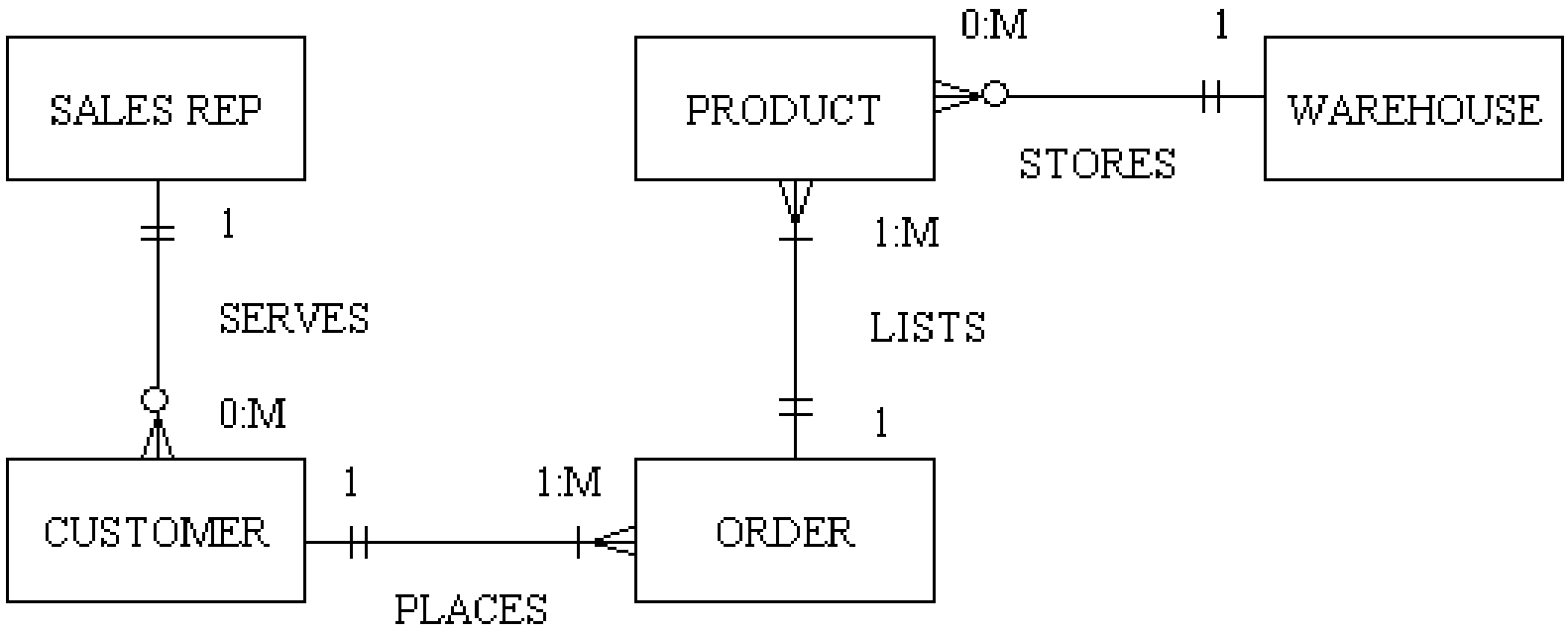
Alternative ER Notation: Crow's Foot Notation



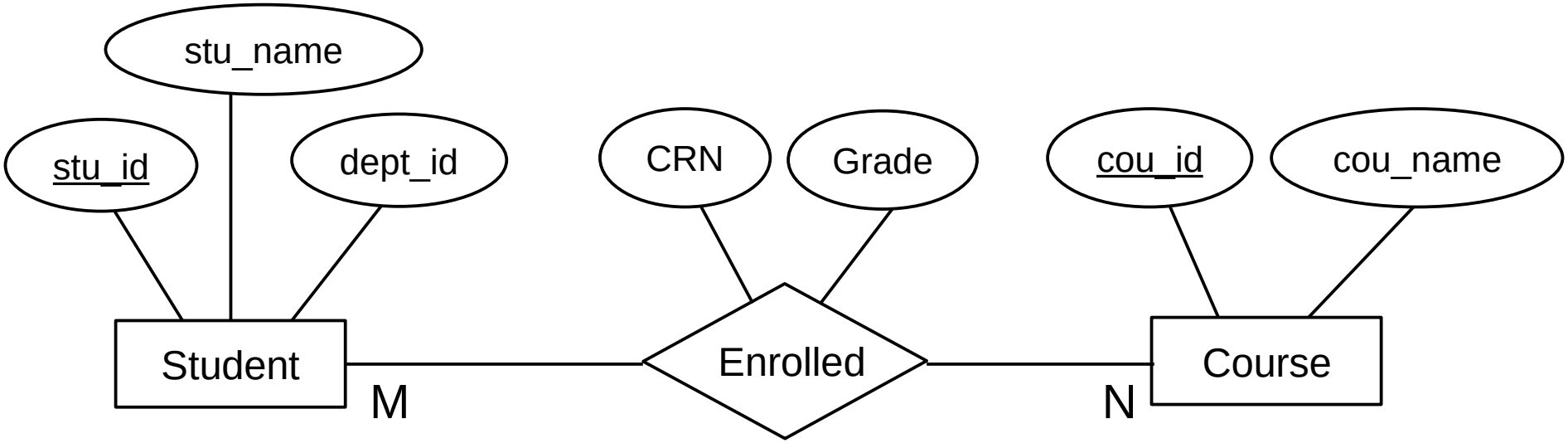


A Richer Crow's Foot Notation

- 1 SALES REPRESENTATIVE serves 0 to many CUSTOMERS
- 1 CUSTOMER places 1 to many ORDERS
- 1 ORDER lists 1 to many PRODUCTS
- 1 WAREHOUSE stores 0 to many PRODUCTS



Another Example



Database Schema:

```
Student (stu_id, stu_name, dept_id)

Course (cou_id, cou_name)

Enrolled (stu_id, cou_id, CRN, Grade)
```

SQL DDL

```
create table Student (stu_id int primary key,
                      stu_name varchar(20),
                      dept_id varchar(5) );

create table Course (cou_id varchar(5) primary key,
                     cou_name varchar(50) );

create table Enrolled (stu_id int, cou_id varchar(5),
                       CRN int, Grade char(2),
            primary key (stu_id, cou_id, CRN),
            foreign key (stu_id) references Student(stu_id) ,
            foreign key (cou_id) references Course(cou_id) );
```

DB Schema

Note: This is a table Diagram – Not an ER Diagram)