Database Management Systems - I, CS 157A

## Relational Database Design Overview

Agenda

- Entity Sets Versus Attributes

- Weak Entity Sets

- Design Choices
  - Case Study

Entity Sets Versus Attributes

■ An entity set, to be kept, it should satisfy at least one of the following conditions, i.e., otherwise the entity set should be merged with another entity set and not to stand on its own:

❑ It is more than the primary key; it has at least one non-key attribute.

<div align="center">or</div>

❑ It is the "many" in a many-one or many-many relationship.
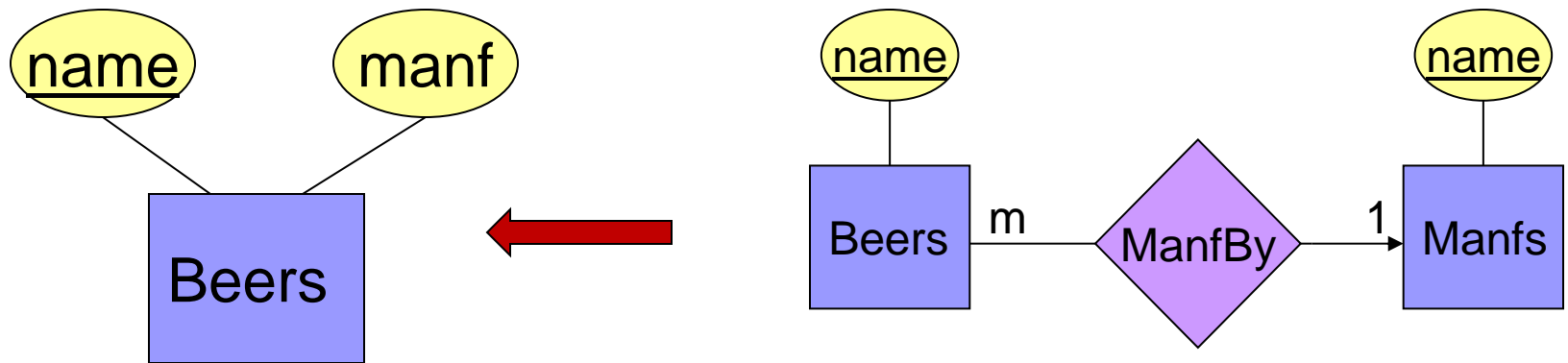
Example: Good
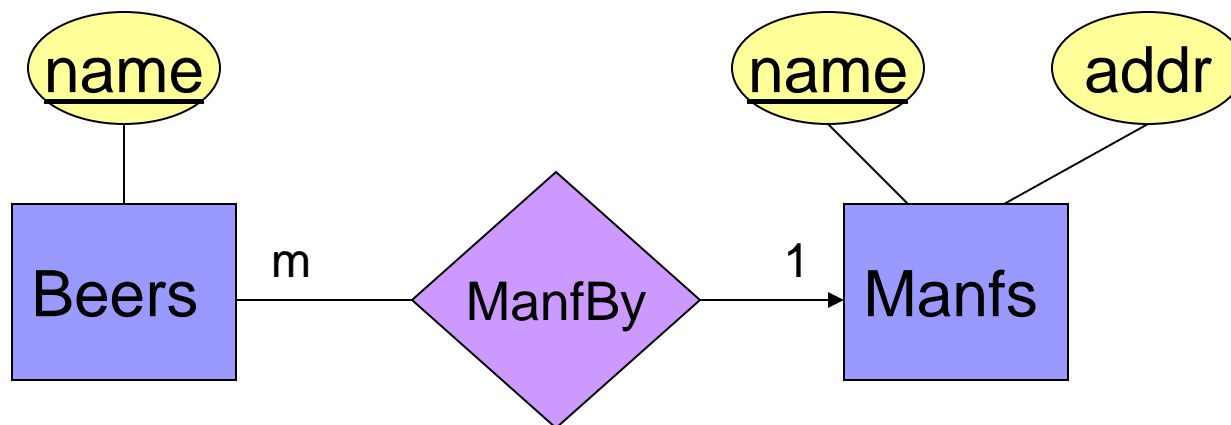
- Manfs deserves to be an entity set because of the nonkey attribute addr.
- Beers deserves to be an entity set because it is the "many" of the many-one relationship ManfBy.

Example: Bad
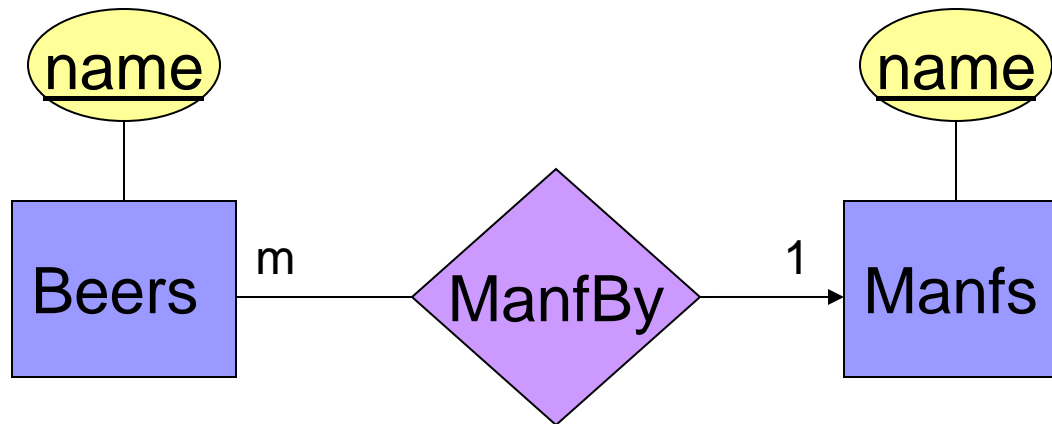
Since the manufacturer is nothing but a name, and is not at the "many" end of any relationship, it should not be an entity set.

Example: Good

There is no need to make the manufacturer an entity set, because we record nothing about manufacturers besides their name.

Weak Entity Sets

Don't Overuse Weak Entity Sets

■ Beginning database designers often doubt that anything could be a key by itself.

   ❑ They make all entity sets weak, supported by all other entity sets to which they are linked.

■ In reality, we usually create unique ID's for entity sets.

   ❑ Examples include social-security numbers, automobile VIN's etc.

When Do We Need Weak Entity Sets?

- The usual reason is that there is no global authority capable of creating unique ID's.
- Example: it is unlikely that there could be an agreement to assign unique player numbers across all football teams in the world.

ER Design

- What are the entities and relationships in the enterprise?

- What information about these entities and relationships should we store in the database?

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

Design Choices

Design Choices

- Should a concept be modeled as an entity or an attribute?

- Should a concept be modeled as an entity or a relationship?

- What constraints should be captured?

Entity vs. Attribute

■ Should *address* be an attribute of Employees or an entity (connected to Employees by a relationship)?

address

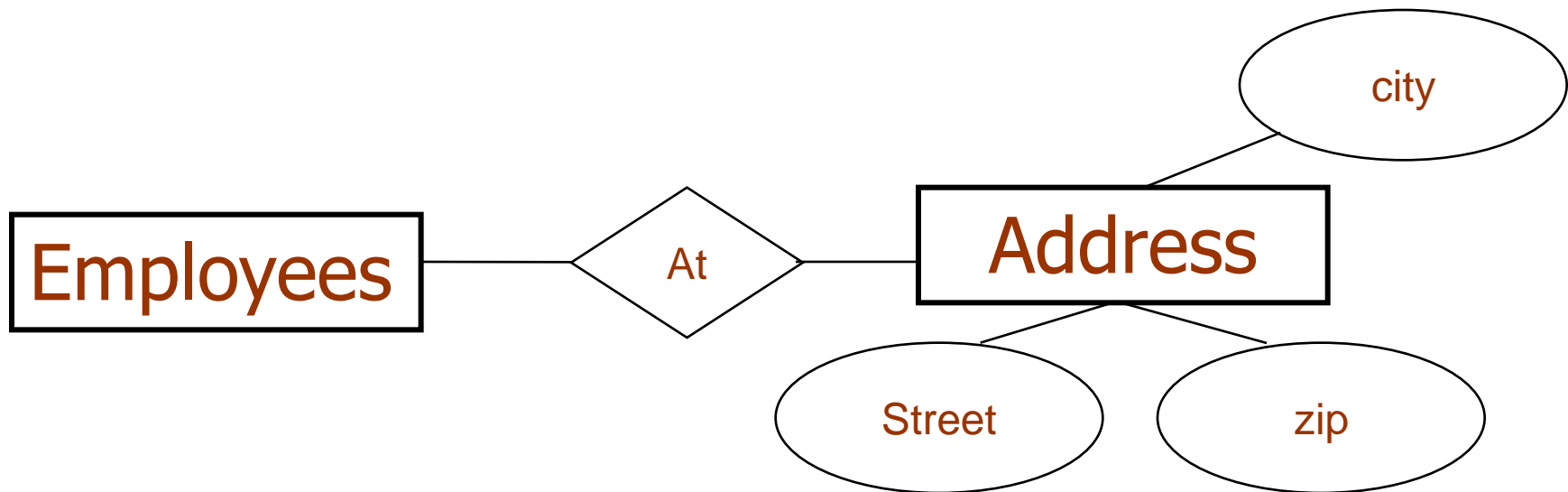Employees

?

Address

...
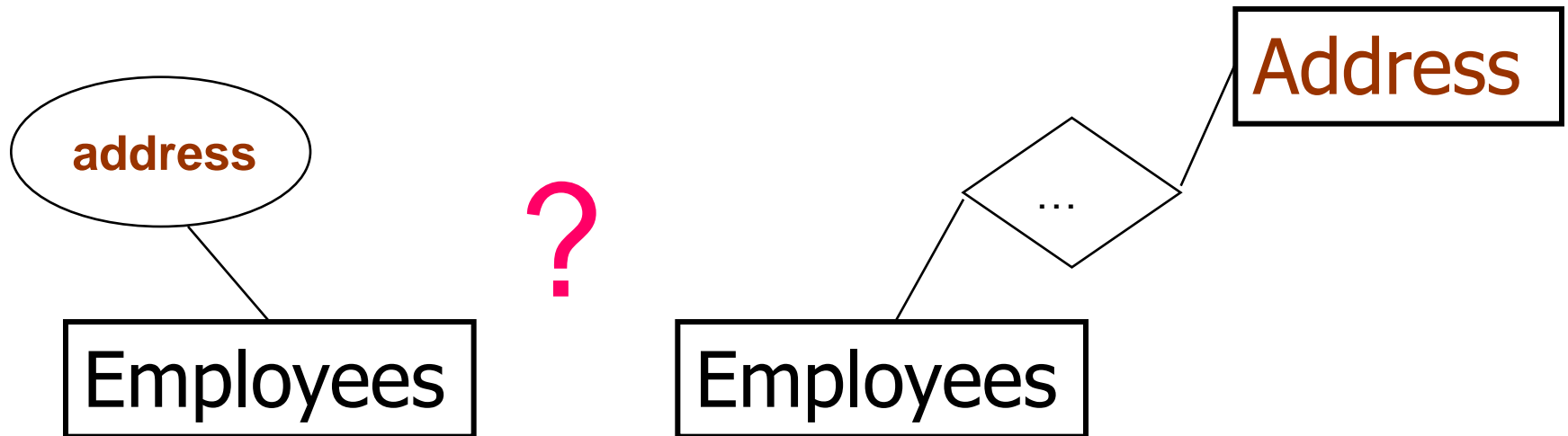
Employees

Entity vs. Attribute

- Depend upon the use we want to make of address information, and the semantics of the data

- If we have several addresses per employee, *address* must be an entity (since attributes cannot be set-valued)

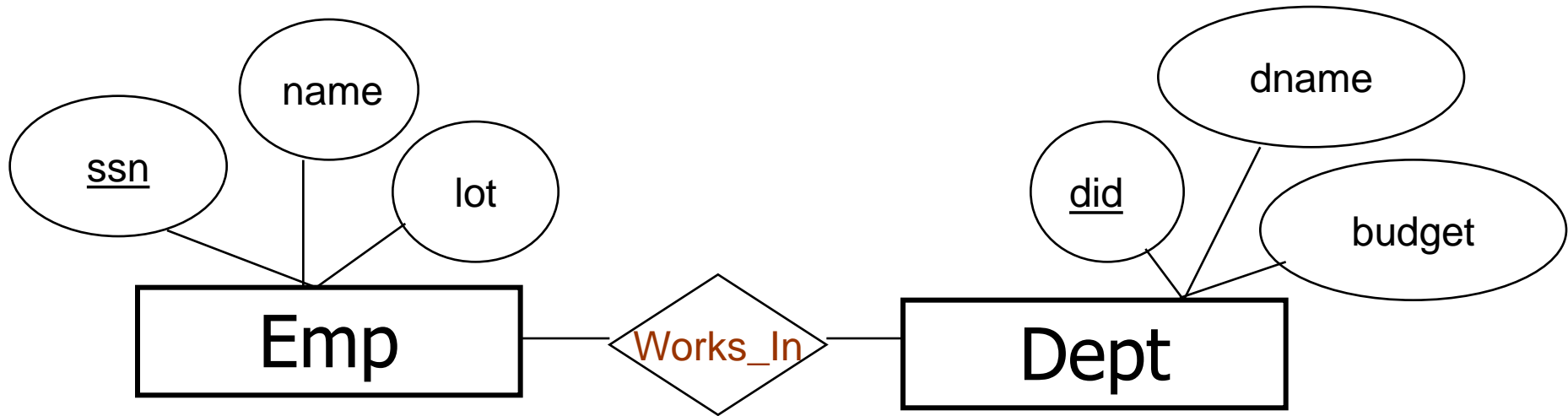Employees ——— At ——— Address
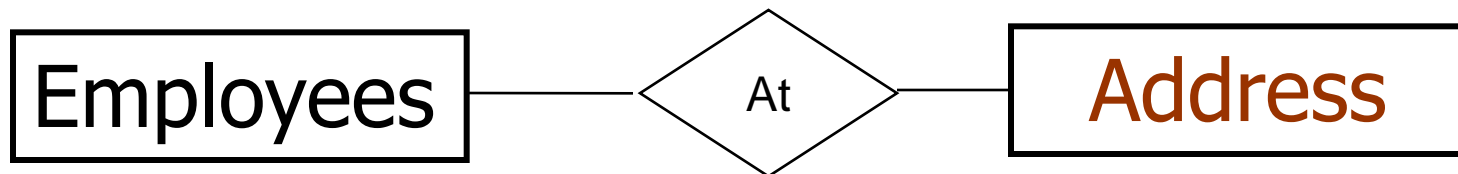
Entity vs. Attribute

- If the structure (city, street, etc.) is important, e.g., we want to retrieve employees in a given city, *address* must be modeled as an entity (since attribute values are atomic)
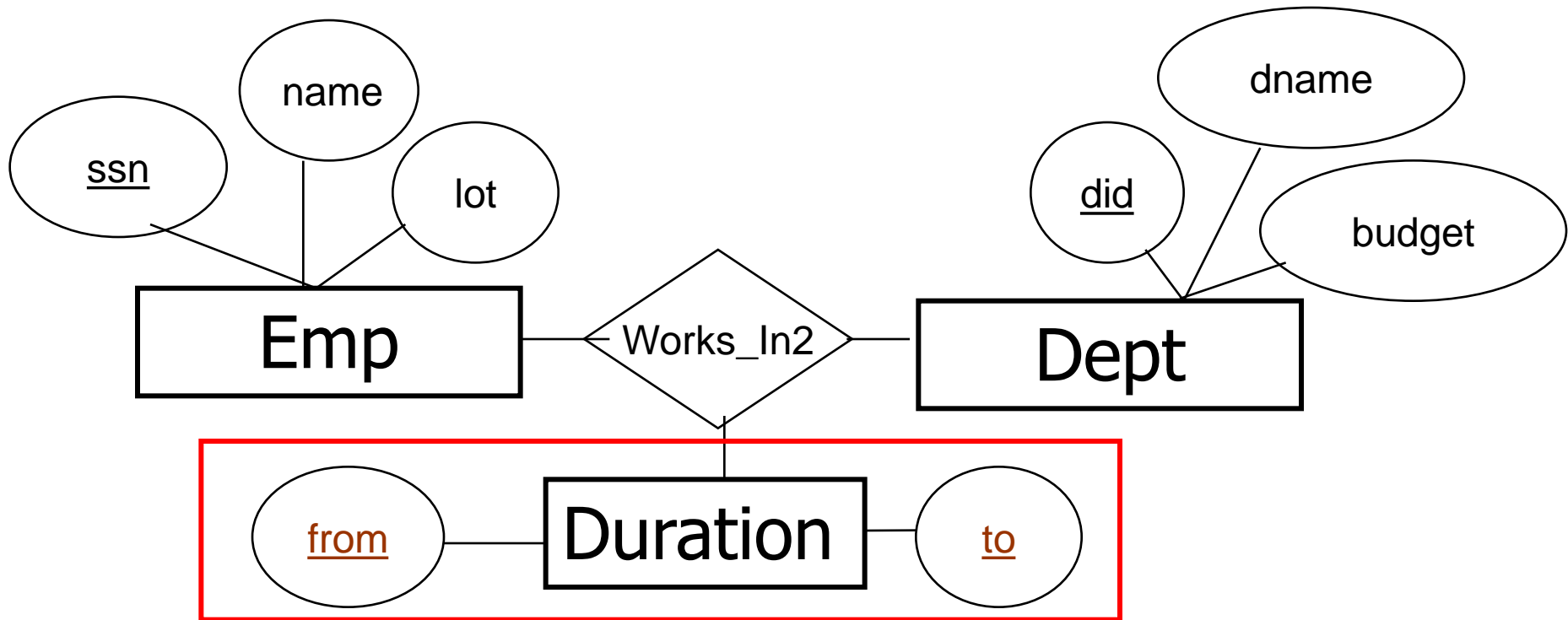
Entity vs. Attribute

- What Problem?
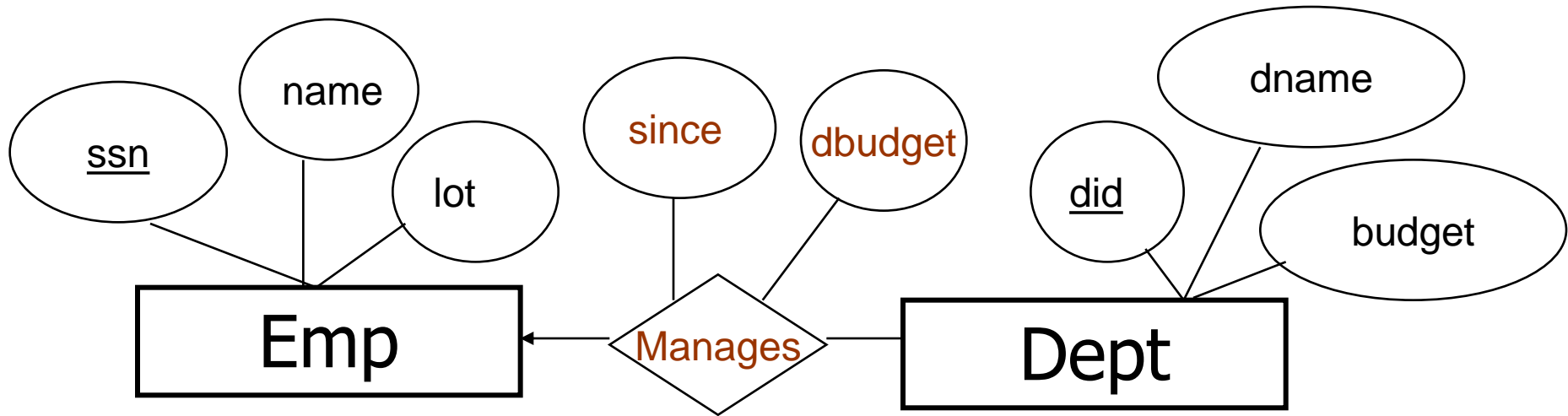- **Works_In** does not allow an employee to work in a department for two or more periods
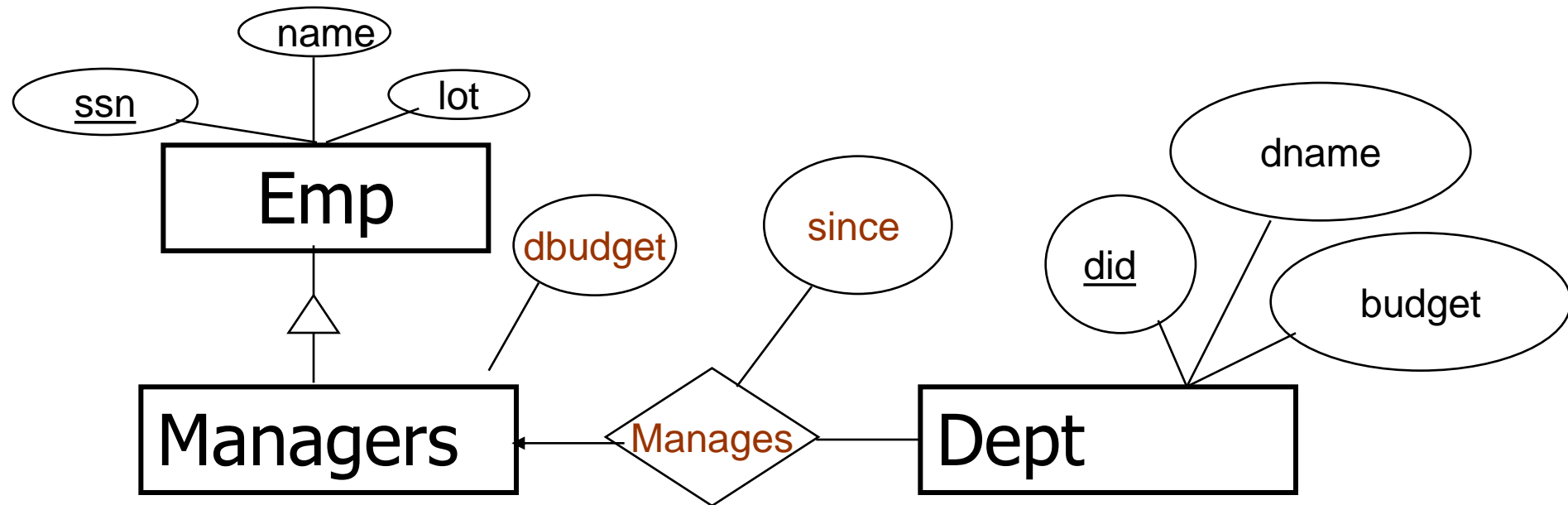
Entity vs. Attribute

- **Works_In2** now allows an employee to work in a department for two or more periods

Entity vs. Relationship

- OK if a manager gets a budget for each dept.
- What if a manager gets a budget that covers *all* managed depts?
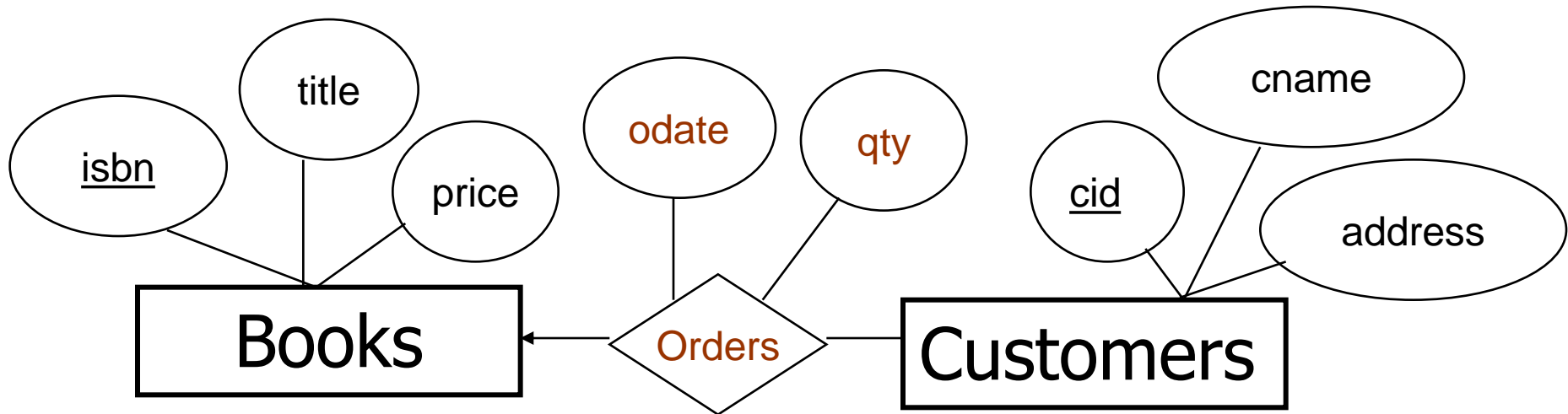
Entity vs. Relationship

- New entity set **Managers** (isa Emp)
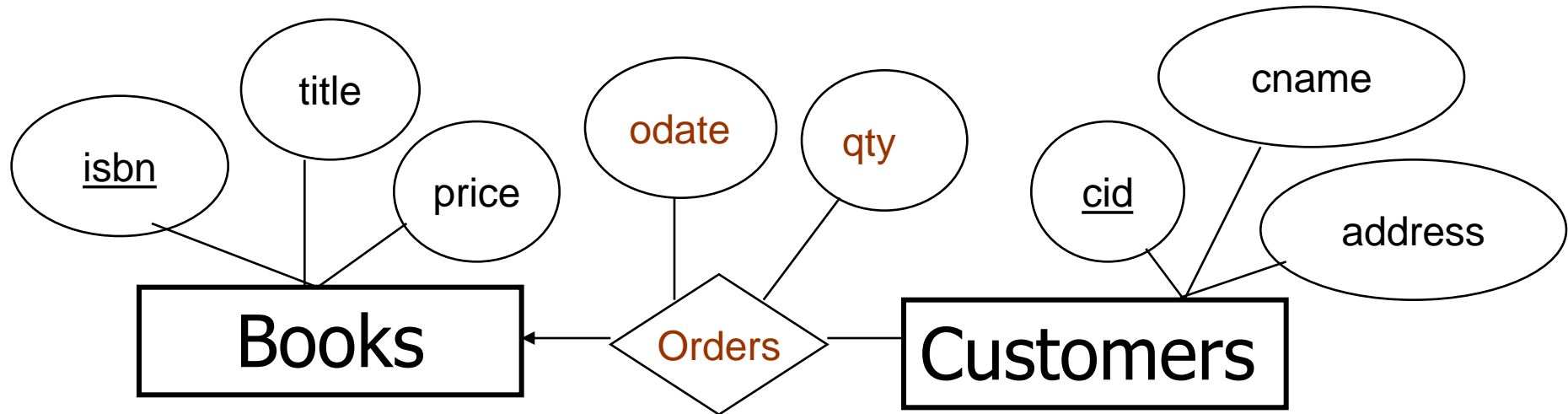- *dbudget* describes a manager, as intended

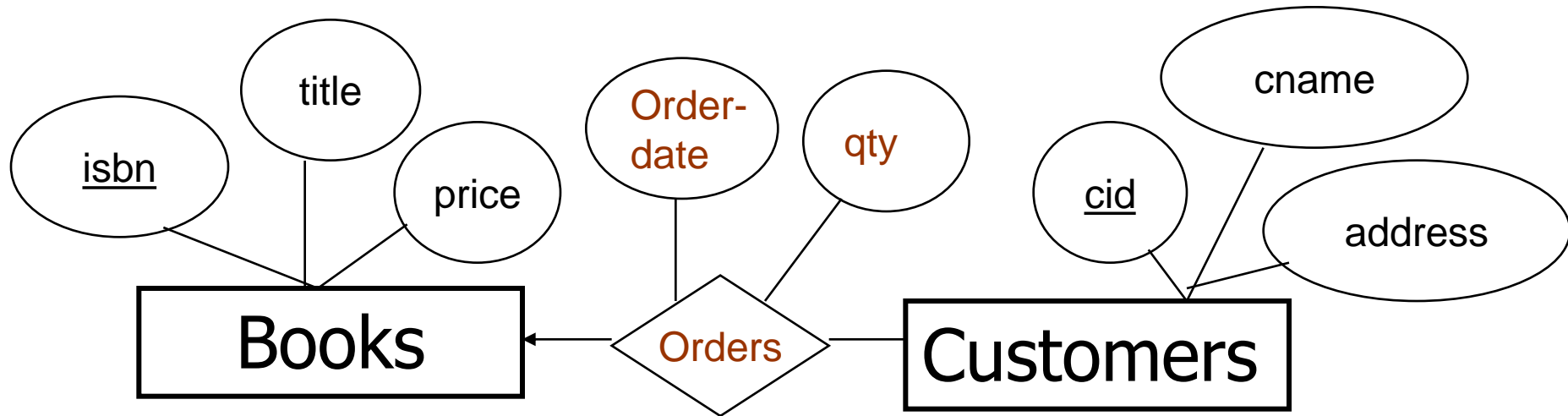# Case Study

ER Design Case Study

- What if a customer places two orders for the same book in one day?
- The second order is handled by updating the value of quantity

ER Design Case Study

- What if a customer places two orders for two different books in one day?
- No problem. Each order relates the customer to a different book.

ER Design Case Study

- What if a customer places two orders for the same book on different days?

- Can we use the order-date to distinguish the two orders?

- No we can't. The attributes of Books and Customers must jointly contain a key for Orders. Thus this design does not allow a customer to place orders for the same book on different days.
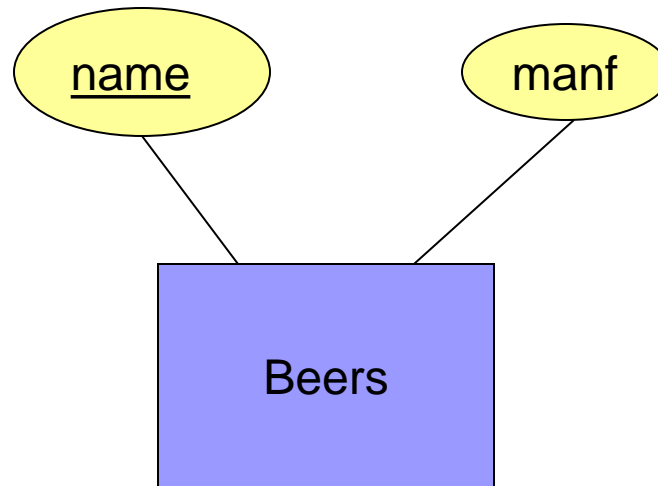
From E-R Diagram to Relations

From E/R Diagrams to Relations

- Entity set → relation.
  - Entity → tuple or row
  - Attributes → attributes.
- Relationships → relations whose attributes are only:
  - The keys of the connected entity sets.
  - Attributes of the relationship itself.

Entity Set → Relation

Relation: Beers(name, manf)

Relationship → Relation

name addr

name manf

Drinkers ── Likes ── Beers

husband

1   2

Buddies

Favorite

wife

Married

Likes(drinker, beer)
Favorite(drinker, beer)
Buddies(name1, name2)
Married(husband, wife)
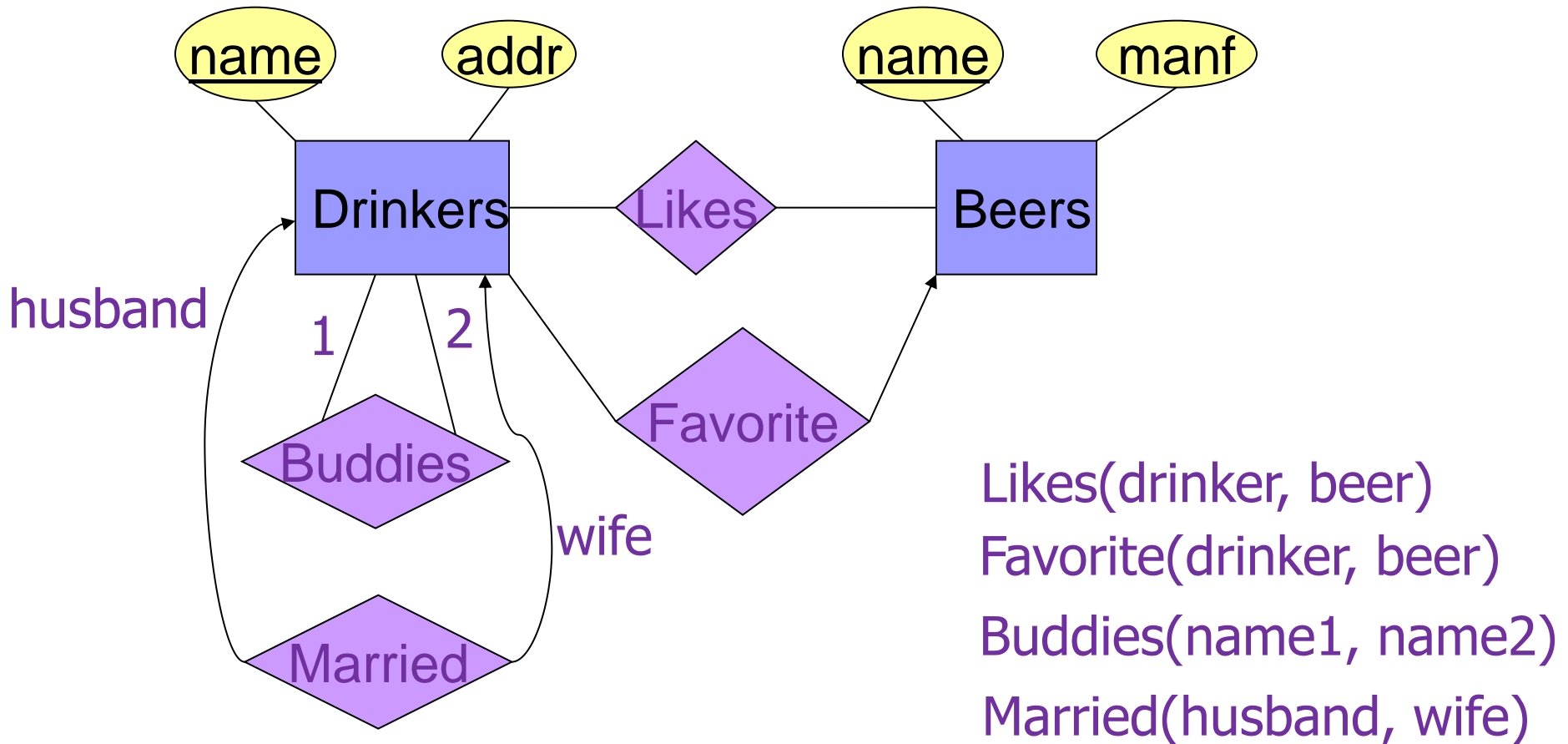
Combining Relations

- OK to combine into one relation:

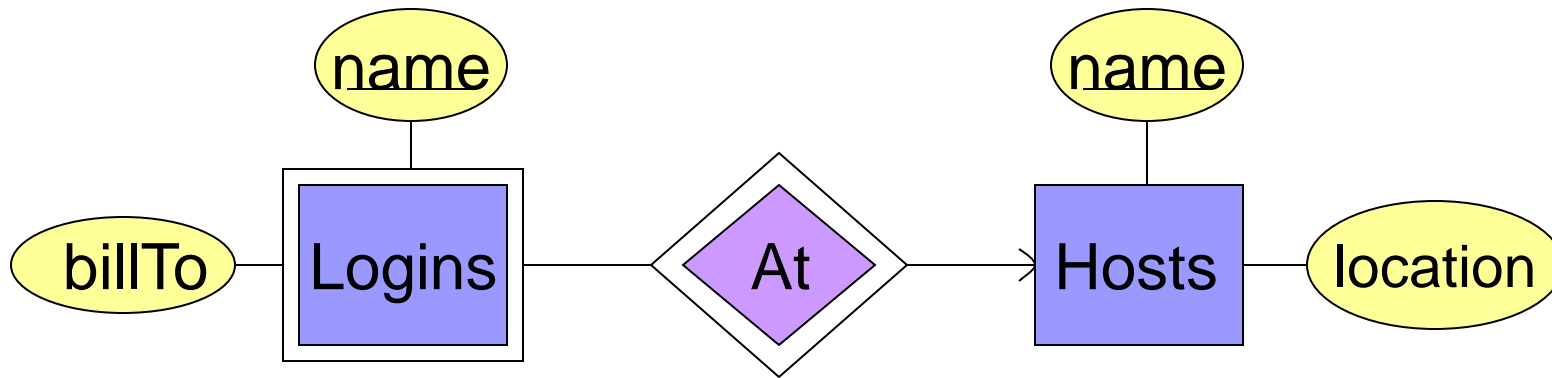 1. The relation for an entity-set *E*
  2. The relations for many-one relationships of which *E* is the "many."

- Example: Drinkers(name, addr) and Favorite(drinker, beer) combine to make Drinker1(name, addr, favBeer).

Example: Weak Entity Set → Relation

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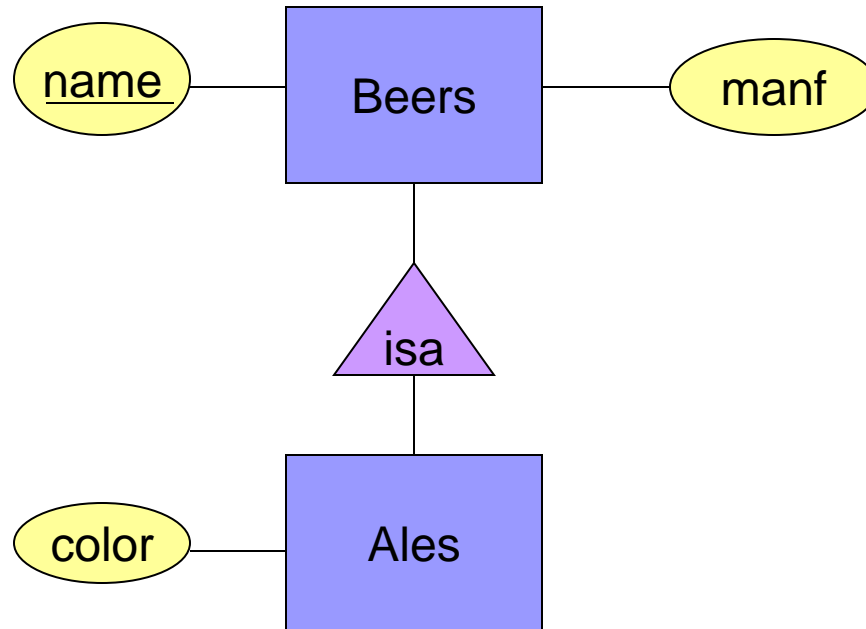At(loginName, hostName)

At becomes part of
Logins

Subclasses: Three Approaches

1. *Object-oriented*: One relation per subset of subclasses, with all relevant attributes.

2. *Use nulls*: One relation; entities have NULL in attributes that don't belong to them.

3. *E/R style*: One relation for each subclass:

   ❑ Key attribute(s).

   ❑ Attributes of that subclass.

Example: Subclass

Object-Oriented

name	manf
Bud	Anheuser-Busch

Beers

name	manf	color
Summerbrew	Pete's	dark

Ales

Good for queries like "find the color of Ales made by Pete's."

E/R Style

name	manf
Bud	Anheuser-Busch
Summerbrew	Pete's

Beers

name	color
Summerbrew	dark

Ales

Good for queries like
"find all Beers (including ales) made by Pete's."

Using Nulls

name	manf	color
Bud	Anheuser-Busch	NULL
Summerbrew	Pete's	dark

Beers

No JOIN + Saves space unless there are *lots* of attributes that are usually NULL.

Summary

- ER design is *subjective*. There are often many ways to model a given scenario.

- Entity vs. attribute

- Entity vs. relationship

- From E/R Diagrams to Relations

END