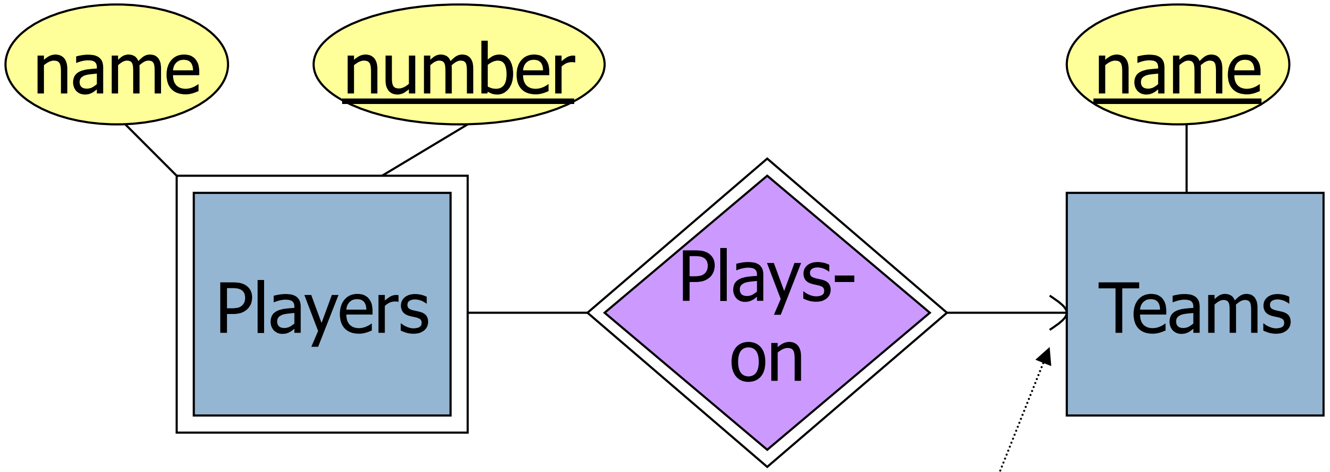
Weak Entity Sets

- Occasionally, entities of an entity set need “help” to identify them uniquely.
- Entity set *E* is said to be *weak* if in order to identify entities of *E* uniquely, we need to follow one or more many-one relationships from *E* and include the key of the related entities from the connected entity sets.

Example: Weak Entity Set

- name is almost a key for football players, but there might be two with the same name.
- number is certainly not a key, since players on two teams could have the same number.
- But number, together with the team name related to the player by Plays-on should be unique.

In E/R Diagrams



Note: must be rounded because each player needs a team to help with the key.

- Double diamond for *supporting* many-one relationship.
- Double rectangle for the weak entity set.

Weak Entity-Set Rules

- A weak entity set has one or more many-one relationships to other (supporting) entity sets.
  - Not every many-one relationship from a weak entity set need be supporting.
  - But supporting relationships must have a rounded arrow (entity at the “one” end is guaranteed).

Weak Entity-Set Rules – (2)

- The key for a weak entity set is its own underlined attributes and the keys from the supporting entity sets.
  - E.g., (player) number and (team) name is a key for Players in the previous example.

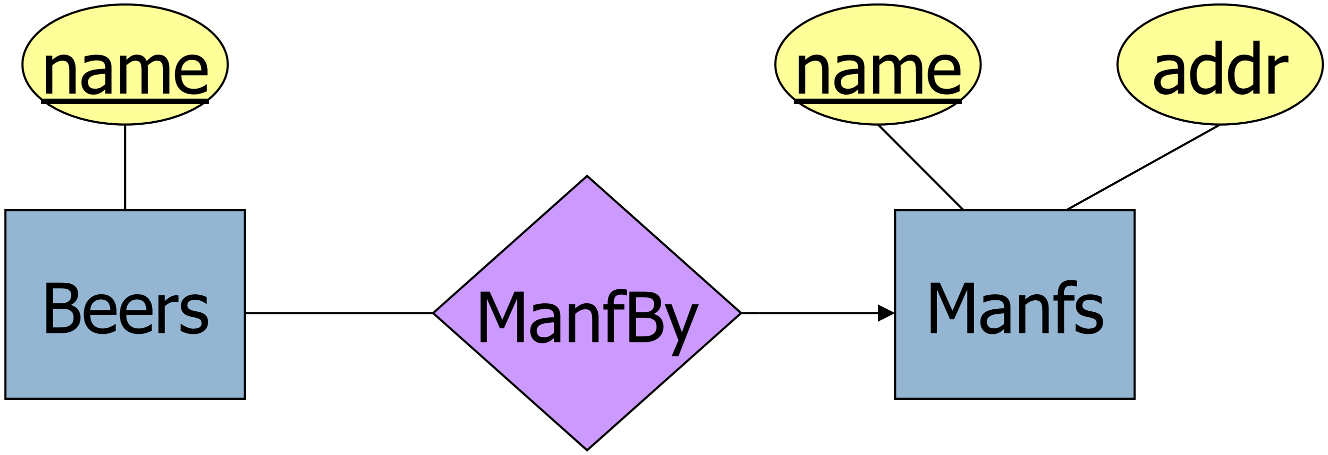
Design Techniques

1. Avoid redundancy.
2. Limit the use of weak entity sets.
3. Don’t use an entity set when an attribute will do.

Avoiding Redundancy

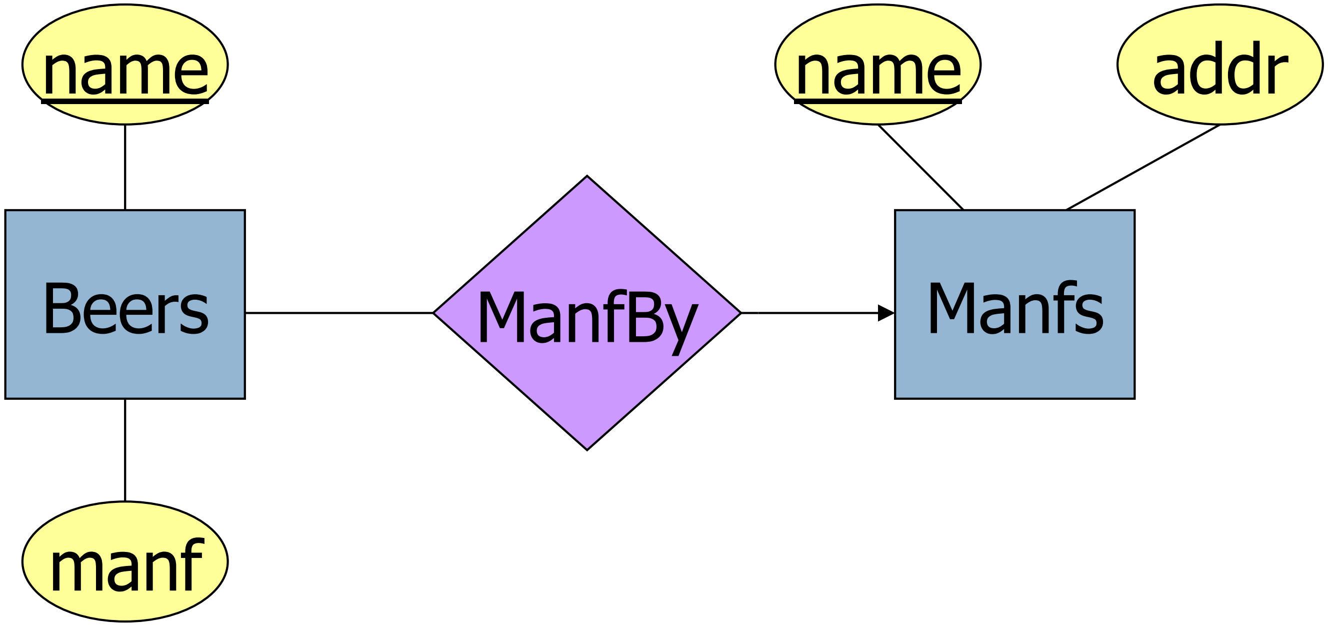
- *Redundancy* = saying the same thing in two (or more) different ways.
- Wastes space and (more importantly) encourages inconsistency.
  - Two representations of the same fact become inconsistent if we change one and forget to change the other.

Example: Good



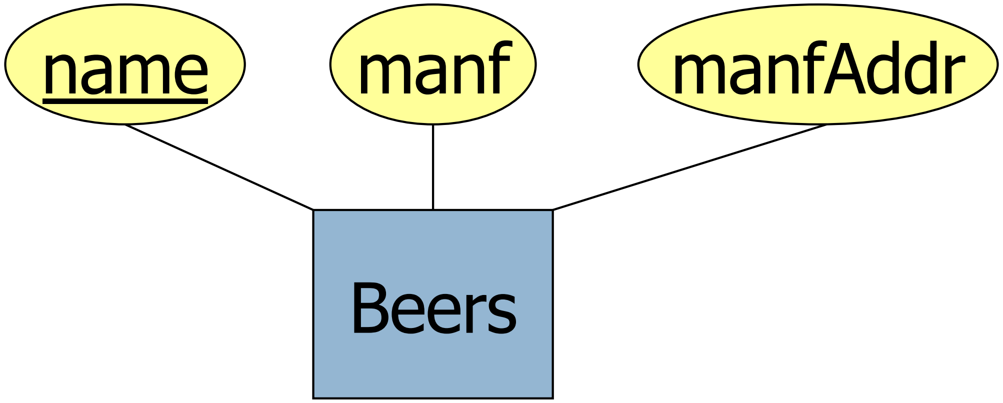
This design gives the address of each manufacturer exactly once.

Example: Bad



This design states the manufacturer of a beer twice: as an attribute and as a related entity.

Example: Bad

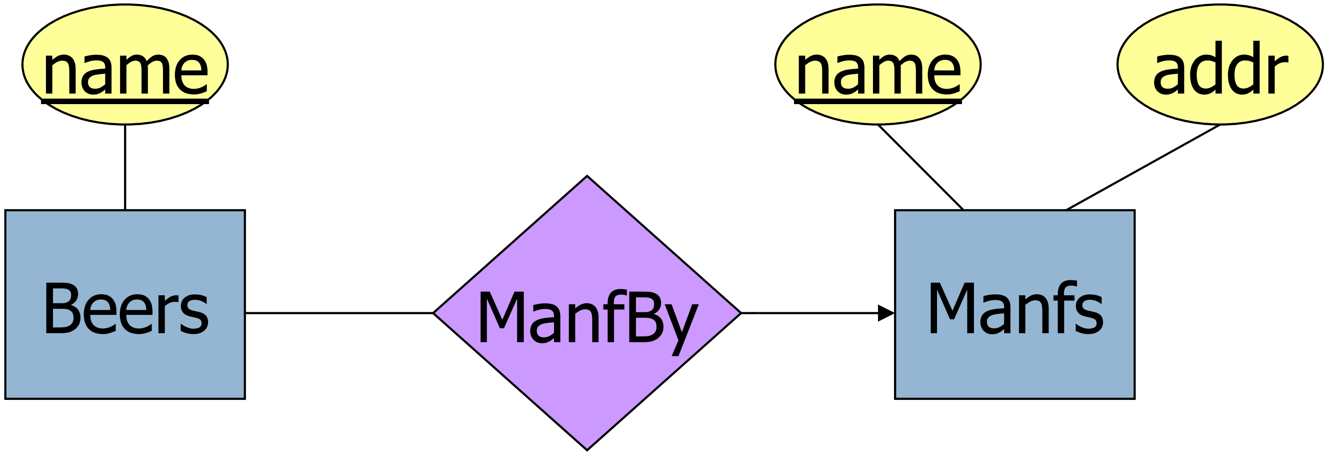


This design repeats the manufacturer’s address once for each beer and loses the address if there are temporarily no beers for a manufacturer.

Entity Sets Versus Attributes

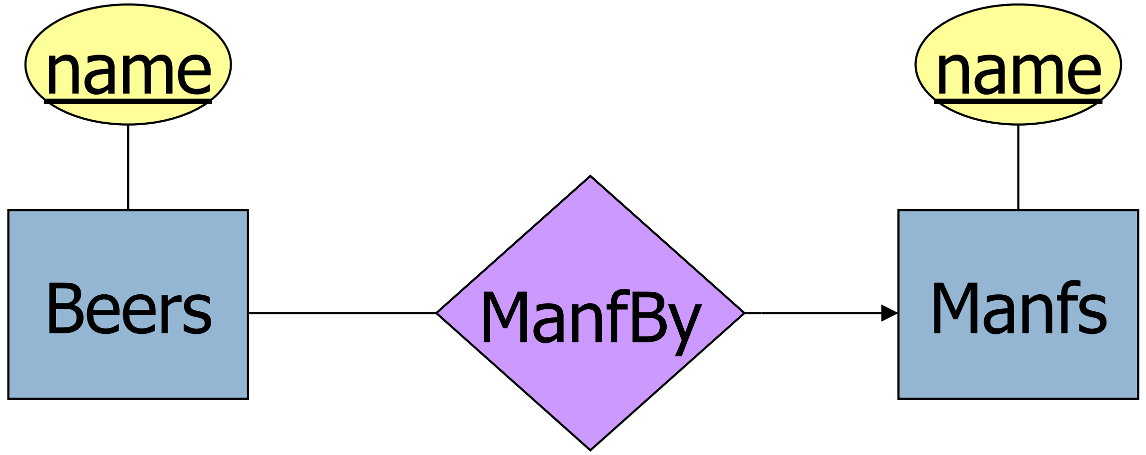
- An entity set should satisfy at least one of the following conditions:
  - It is more than the name of something; it has at least one non-key attribute. OR
  - It is the “many” in a many-one or many-many relationship.
- Depends on the application requirements:
  - If we have several addresses per employee, *address* must be an entity (since attributes cannot be set-valued).
  - 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).

Example: Good



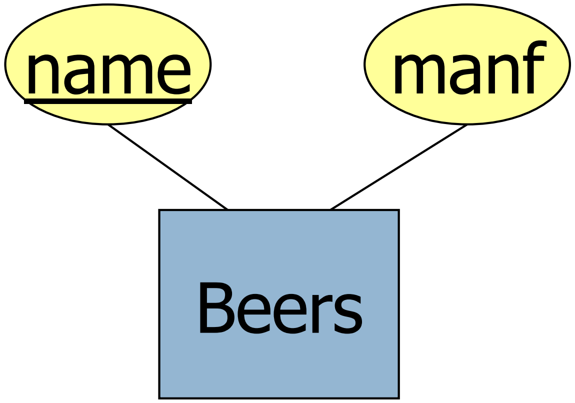
- Manfs deserves to be an entity set because of the nonkey attribute addr.
- Beers deserves to be an entity set because it is the "many" of the many-one relationship ManfBy.

Example: Bad



Since the manufacturer is nothing but a name,
and is not at the “many” end of any relationship,
it need 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.

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.

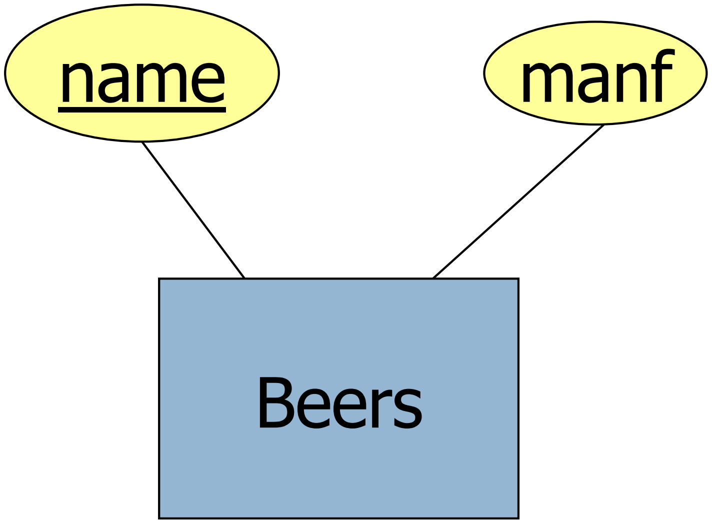
When Do We Need Weak Entity Sets?

- 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.

From E/R Diagrams to Relations

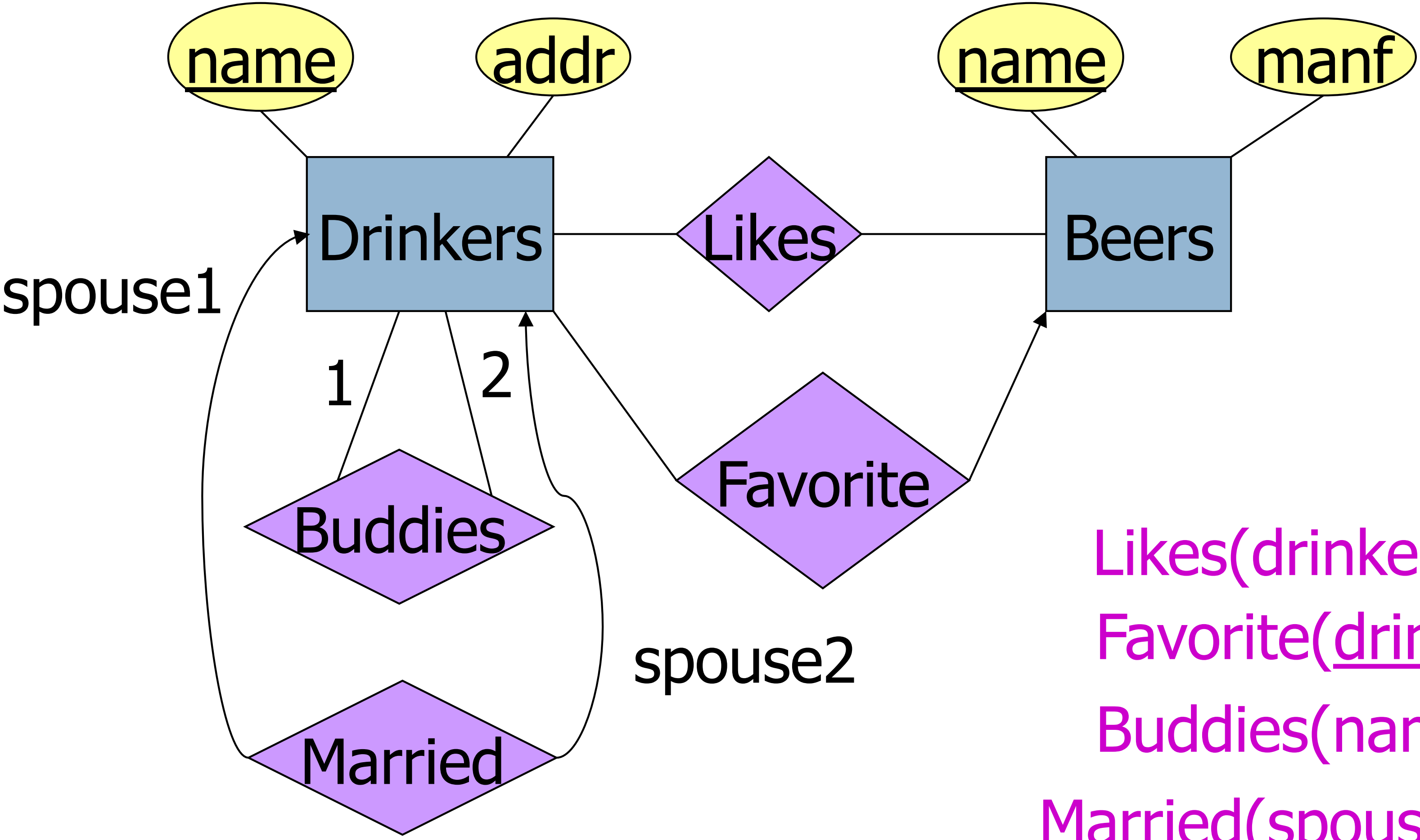
- Entity set -> relation.
  - Attributes -> attributes.
- Relationships -> relations whose attributes are only:
  - The keys of the connected entity sets.
  - Attributes of the relationship itself.

Entity Set -> Relation



Relation: Beers(<u>name</u>, manf)

Relationship -> Relation



Likes(drinker, beer)

Favorite(drinker, beer)

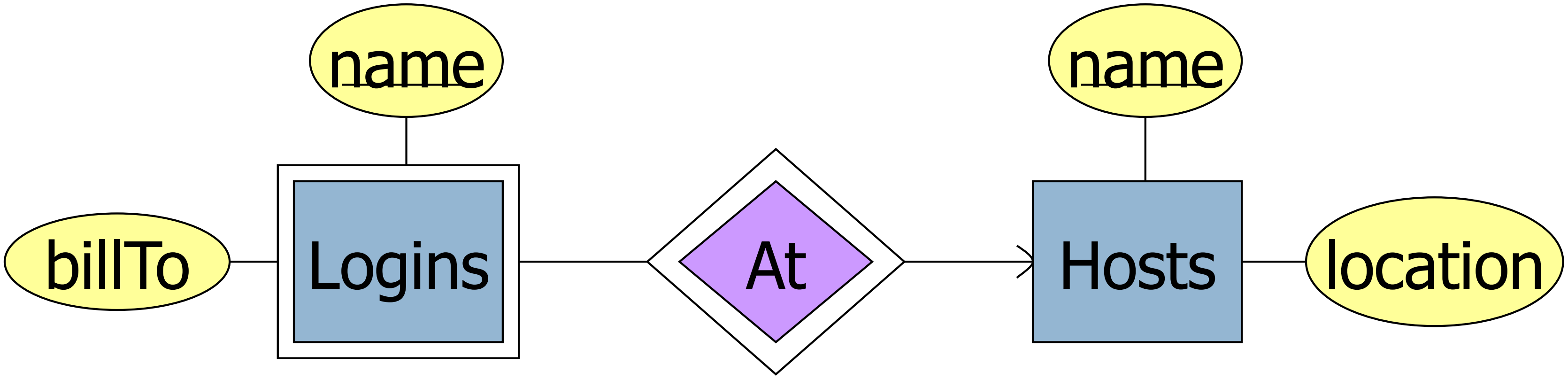
Buddies(name1, name2)

Married(spouse1, spouse2)

Handling Weak Entity Sets

- □ Relation for a weak entity set must include attributes for its complete key (including those belonging to other entity sets), as well as its own, nonkey attributes.
- □ A supporting relationship is redundant and yields no relation (unless *it* has attributes).

Example: Weak Entity Set -> Relation

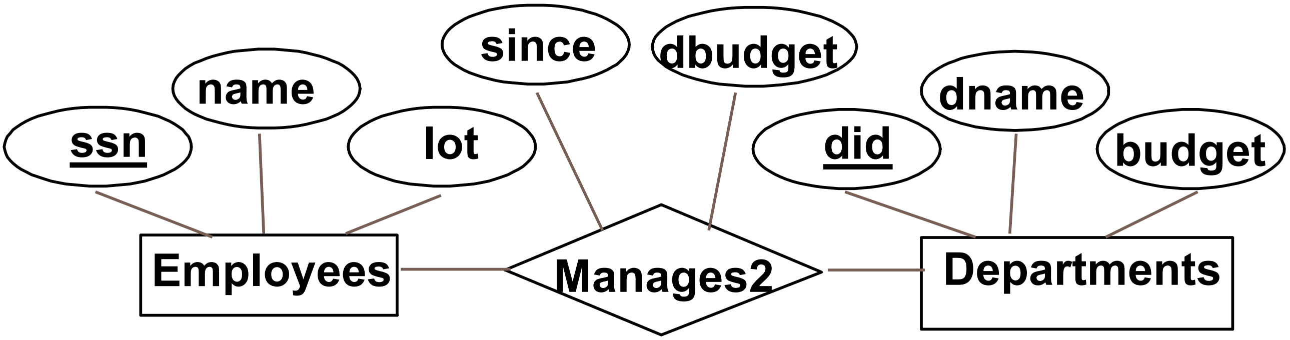


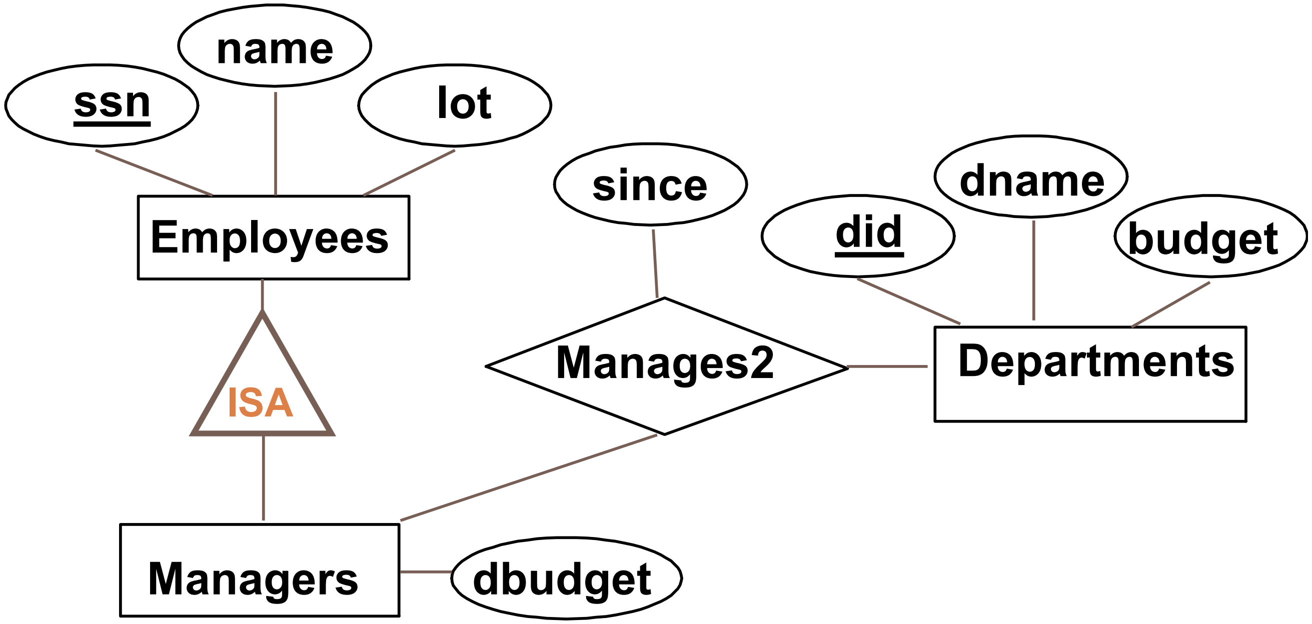
Hosts(hostName, location)
Logins(loginName, hostName, billTo)
~~At(loginName, hostName)~~

At becomes part of Logins

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.





Summary

- *Conceptual design* follows *requirements analysis*,
  - Yields a high-level description of data to be stored
- ER model popular for conceptual design
  - Constructs are expressive, close to the way people think about their applications.
- Basic constructs: *entities*, *relationships*, and *attributes* (of entities and relationships).
- Some additional constructs: *weak entities*, *ISA hierarchies*.

Summary of ER (cont'd.)

Several kinds of integrity constraints can be expressed in the ER model:

- *key constraints,*
- *participation constraints*
- *overlap/covering constraints* for ISA hierarchies.

Constraints play an important role in determining the best database design for an enterprise.

Summary (cont'd)

- ER design is *subjective*. There are often many ways to model a given scenario!
- Analyzing alternatives can be tricky, especially for a large enterprise. Common choices include:
  - Entity vs. attribute
  - entity vs. relationship
  - binary or n-ary relationship
  - whether or not to use ISA hierarchies

Ensuring good database design: resulting relational schema should be analyzed and refined further.