The Entity-Relationship Model

csc343, winter 2011

Diane Horton

University of Toronto

Based on slides by Jeff Ullman

Relational Database Design

- The ultimate goal is a relational schema in either BCNF or 3NF.
- The first step is requirements elicitation.
- Then we create a draft schema.
- Then use decomposition to get into normal form.

Ways to Draft a Schema

- Options:
  - Put all attributes into one relation!
  - Use your judgment to draft a schema.
  - Use a higher-level model to describe the data, then convert to a relational schema.
- Two higher-level models: ER and UML.
- The conversion to a schema is fairly mechanical, but still involves judgment.

Intro to the E/R model

- We describe our data in terms of entities (things/objects) and the relationships between them.
- The model we come up with is expressed as a diagram.

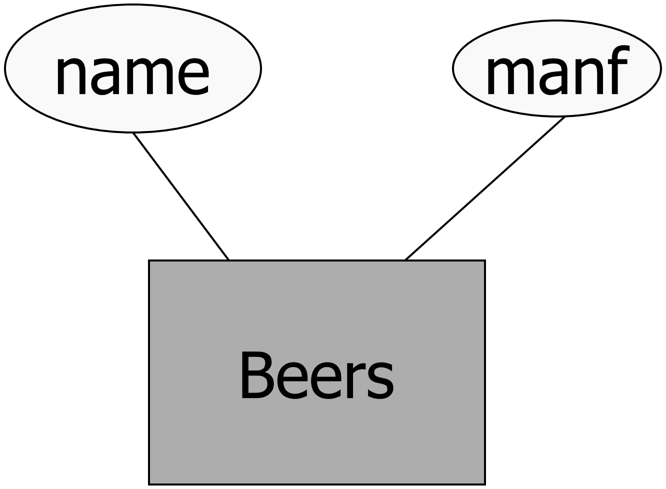
Entities

- *Entity set* = a collection of similar entities.
  - E.g., person
  - Similar to a class in OOP.
- *Entity* = a “thing” or object.
  - E.g., Rohinton Mistry
  - Similar to an instance of a class in OOP.
- *Attribute* = a property of (the entities in) an entity set.
  - Attributes are simple values, e.g. integers or character strings, not structs, sets, etc.

Entities in E/R Diagrams

- Entity set = rectangle.
- Attribute = oval,
  with a line to the rectangle representing
  its entity set.

Example:

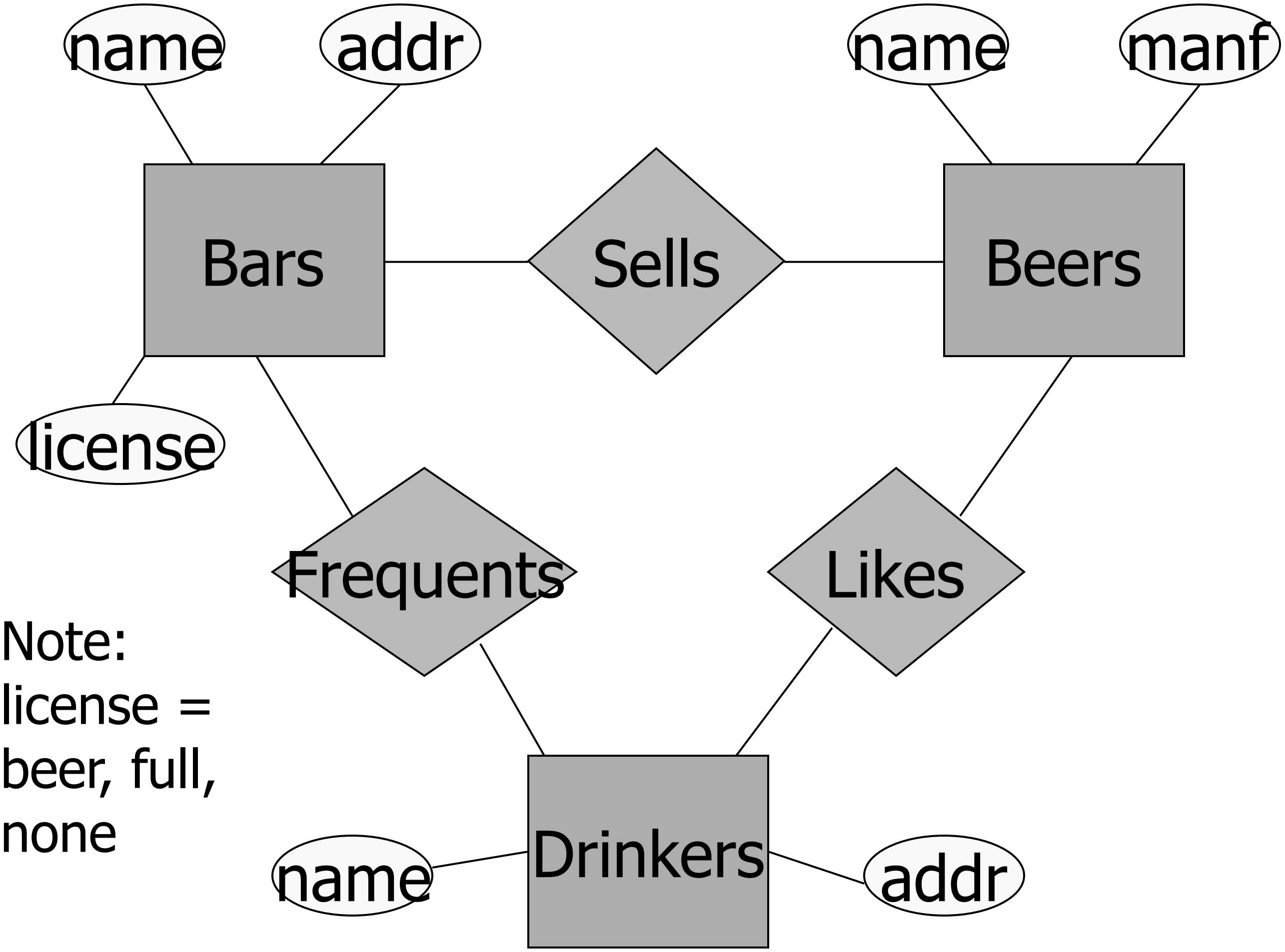


- Entity set Beers has two attributes, name and manf (manufacturer).
- Each Beers entity has values for these two attributes, e.g. (Bud, Anheuser-Busch)

Relationships

- ◆A relationship connects two or more entity sets.
- ◆It is represented by a diamond, with lines to each of the entity sets involved.

Example: Relationships



Note: license = beer, full, none

Bars sell some beers.

Drinkers like some beers.

Drinkers frequent some bars.

Relationship Set

- The current "value" of an entity set is the set of entities that belong to it.
  - Example: the set of all bars in our database.
- The "value" of a relationship is a *relationship set*, a set of tuples with one component for each related entity set.

Example: Relationship Set

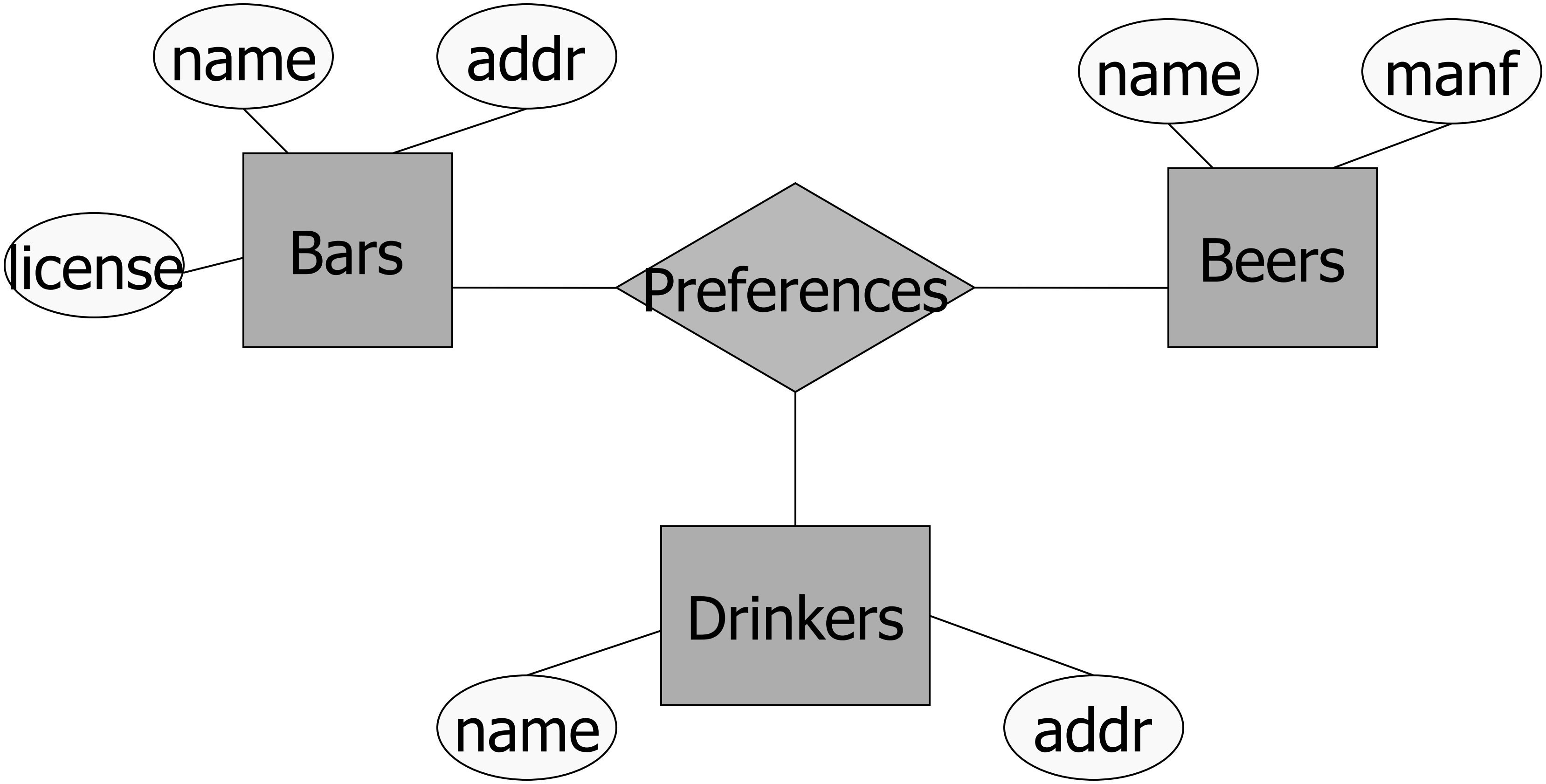
- For the relationship Sells, we might have a relationship set like:

Bar	Beer
Joe’s Bar	Bud
Joe’s Bar	Miller
Sue’s Bar	Bud
Sue’s Bar	Pete’s Ale
Sue’s Bar	Bud Lite

Multiway Relationships

- Sometimes, we need a relationship that connects more than two entity sets.
- Suppose that drinkers will only drink certain beers at certain bars.
  - Our three binary relationships Likes, Sells, and Frequents do not allow us to make this distinction.
  - But a 3-way relationship would.

Example: 3-Way Relationship



A 3-Way Relationship Set

Bar	Drinker	Beer
Joe’s Bar	Ann	Miller
Sue’s Bar	Ann	Bud
Sue’s Bar	Ann	Pete’s Ale
Joe’s Bar	Bob	Bud
Joe’s Bar	Bob	Miller
Joe’s Bar	Cal	Miller
Sue’s Bar	Cal	Bud Lite

Multiplicity of Relationships

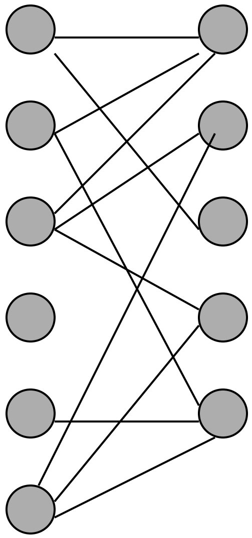
- Consider a binary relationship between entity sets *E* and *F*.
- Some binary relationships can connect any number of members of *E* to any number of members of *F*.
- But some binary relationships have restrictions on their multiplicity.

Many-Many Relationships

- In a *many-many* relationship, an entity of either set can be connected to many entities of the other set.
  I.e., there is no restriction.
- E.g., a bar sells many beers; a beer is sold by many bars.

In Pictures:

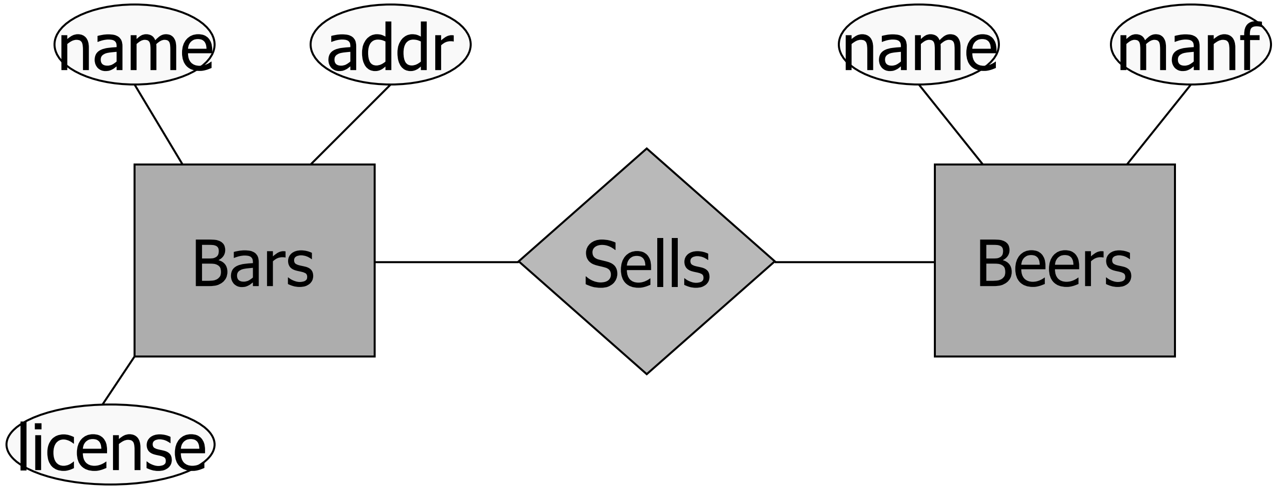
Instances of entity set E



Instances of entity set F

many-many

Many-many in an E/R Diagram

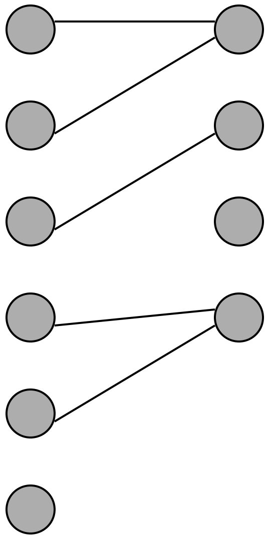


Just draw the line from entity set to relation to entity set with no arrows.

Many-One Relationships

- Some binary relationships are *many -one* from one entity set to another.
- Each entity of the first set is connected to at most one entity of the second set.
- But an entity of the second set can be connected to zero, one, or many entities of the first set.

In Pictures:



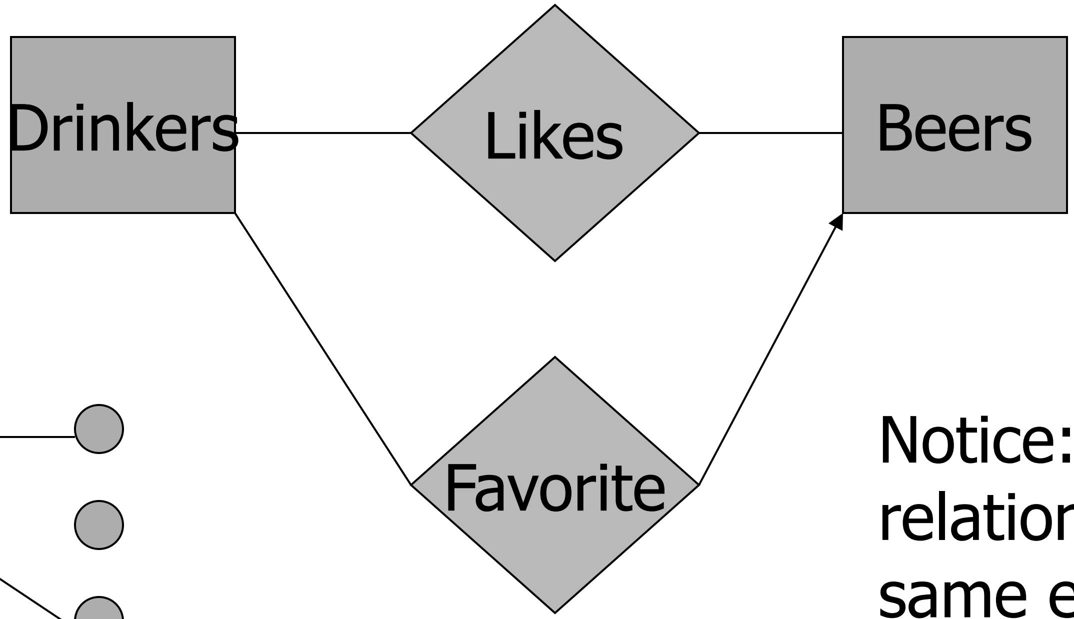
many-one

Example: Many-One Relationship

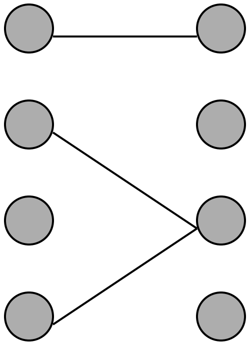
- Favorite, from Drinkers to Beers is many-one.
- A drinker has at most one favorite beer.
- But a beer can be the favorite of any number of drinkers, including zero.

Many-One in an E/R Diagram

Draw an arrow into the “one” side.



Notice: here two different relationships connect the same entity sets.

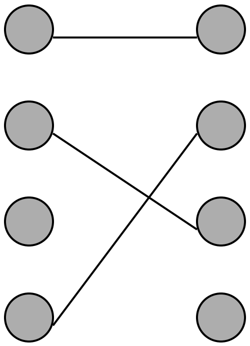


A valid instance of Favorite (a partial function)

One-One Relationships

- In a *one-one* relationship, each entity of either entity set is related to at most one entity of the other set.
- Example: Relationship Best-seller between entity sets Manfs (manufacturer) and Beers.
  - A beer cannot be made by more than one manufacturer, and no manufacturer can have more than one best-seller (assume no ties).

In Pictures:



one-one

One-one in an E/R Diagram

- Show a one-one relationship by arrows entering both entity sets.
- If the relationship is not optional,

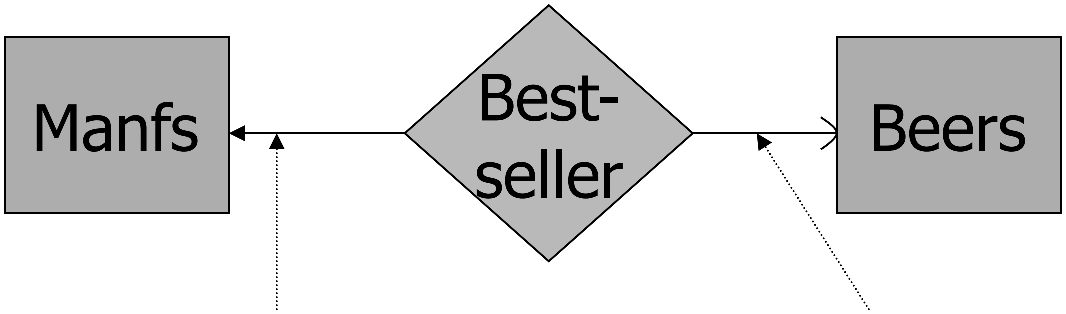
  i.e., if each entity of the first set is related to *exactly* one entity of the target set (vs at most one),

  use a rounded arrow.

Example: One-One Relationship

- Consider Best-seller between Manfs and Beers.
- Some beers are not the best-seller of any manufacturer, so a rounded arrow to Manfs would be inappropriate.
- But a beer manufacturer has to have a best-seller.

In the E/R Diagram



Manfs

Best-seller

Beers

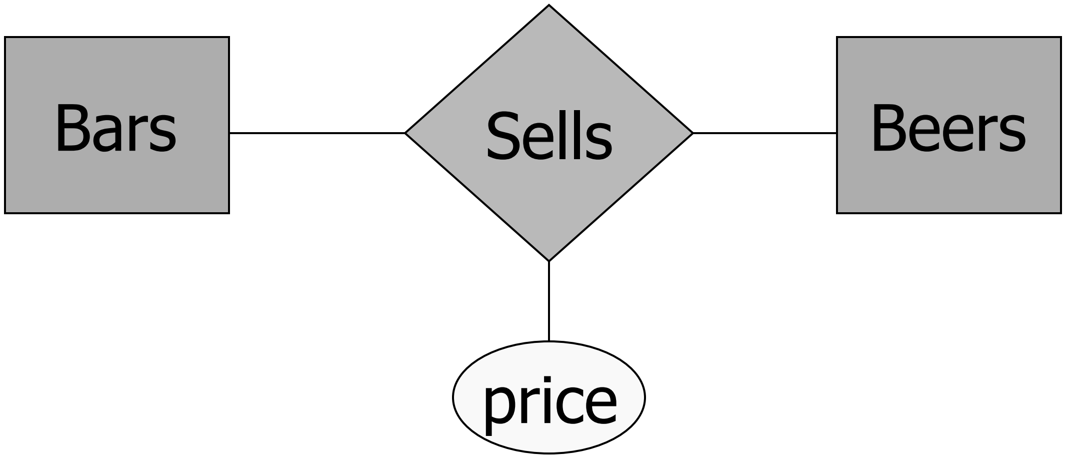
A beer is the best-seller for at most 1 Manufacturer.

A manufacturer has exactly one best seller.

Attributes on *Relationships*

- Sometimes it is useful to attach an attribute to a relationship.
- Think of this attribute as a property of tuples in the relationship set.

Example: Attribute on Relationship

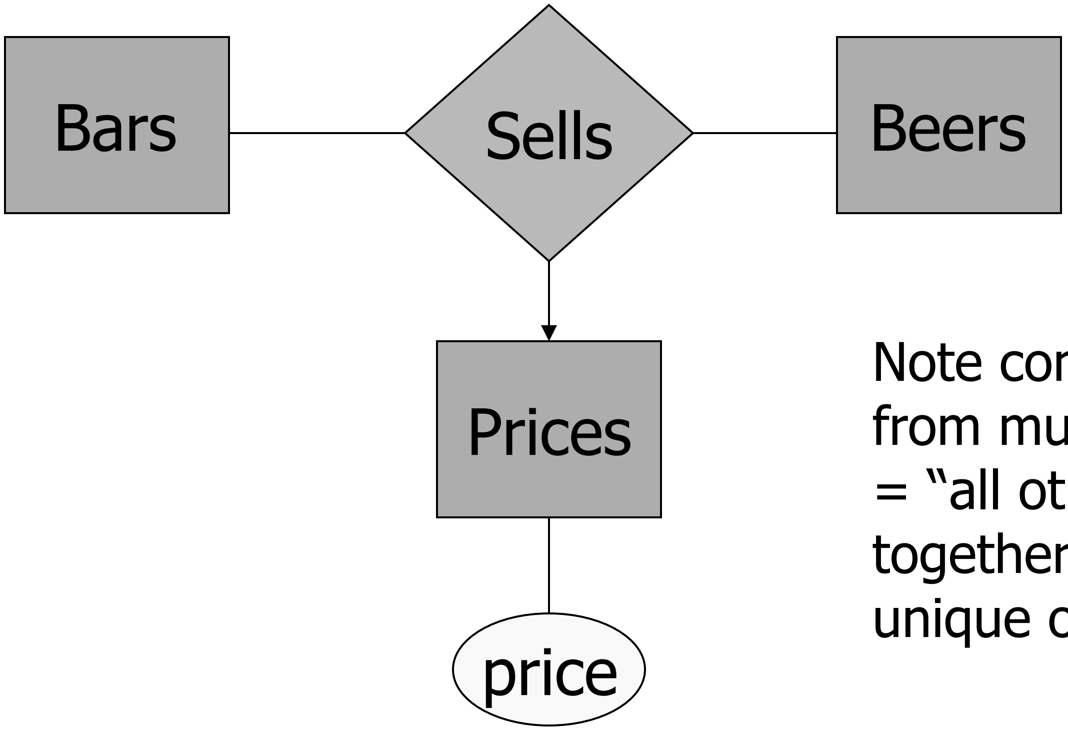


Price is a function of both the bar and the beer, not of one alone.

Removing Attributes on Relationships

- There is always an equivalent E/R diagram without any attributes on relationships.
- Create an entity set representing values of the attribute.
- Make that entity set participate in the relationship.

Example: Removing an Attribute from a Relationship

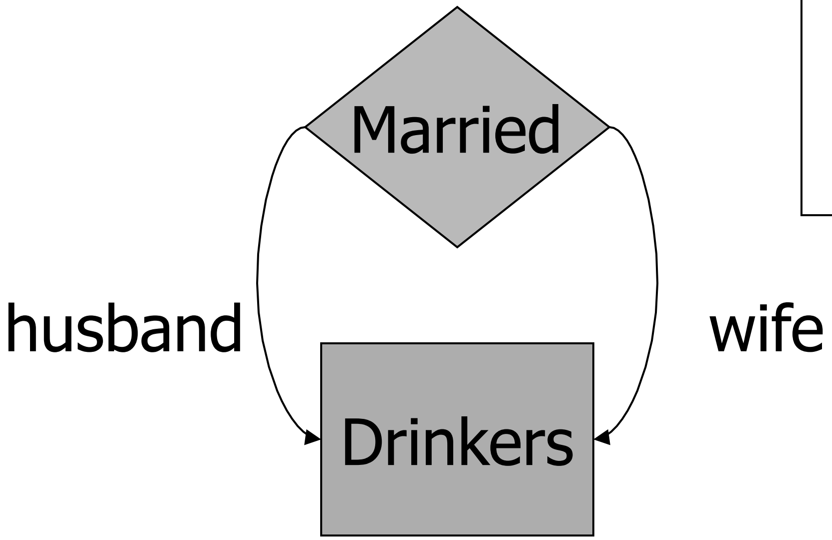


Note convention: arrow from multiway relationship = "all other entity sets together determine a unique one of these."

Roles

- Sometimes an entity set appears more than once in a relationship.
- Label the edges between the relationship and the entity set with names called *roles*.

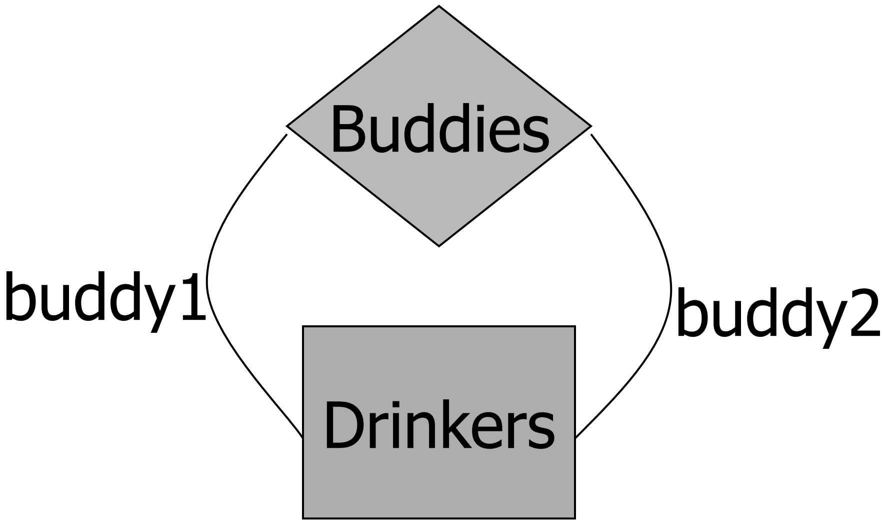
Example: Roles



Relationship Set

Husband	Wife
Bob	Ann
Joe	Sue
…	…

Example: Roles



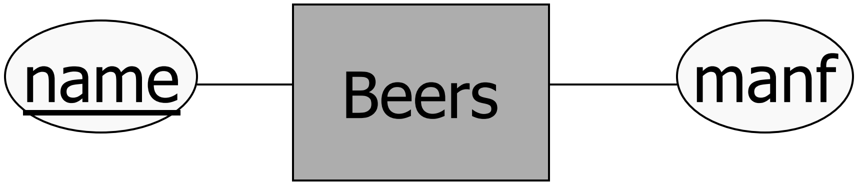
Relationship Set

Buddy1	Buddy2
Bob	Ann
Joe	Sue
Ann	Bob
Joe	Moe
…	…

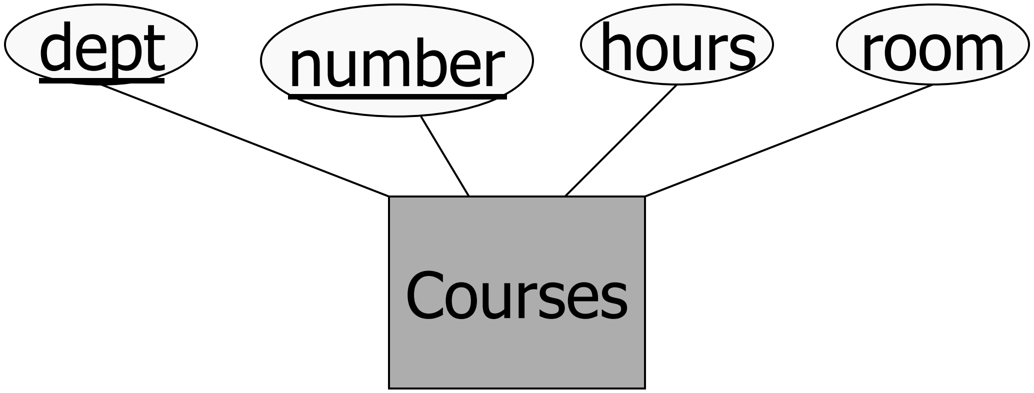
Keys in E/R Diagrams

- A *key* is a set of attributes for one entity set such that no two entities in this set agree on all the attributes of the key.
  - It is allowed for two entities to agree on some, but not all, of the key attributes.
- We must designate a key for every entity set.
- Notation: underline the key attribute(s).

Example: name is Key for Beers



Example: a Multi-attribute Key



- Note that hours and room could also serve as a key, but we must select only one key.

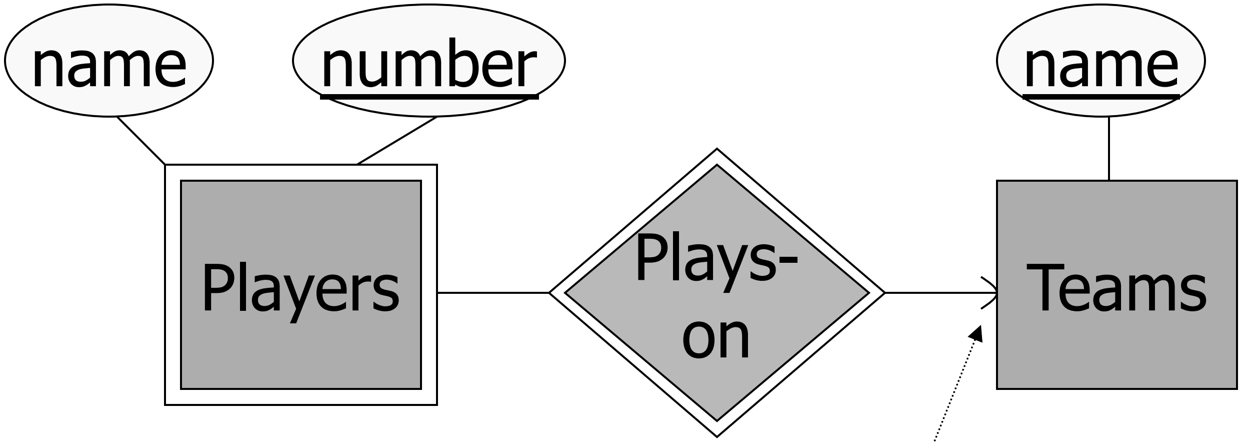
Weak Entity Sets

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

Weak Entity-Set Rules

- A weak entity set must have one or more many-one relationships to other (supporting) entity sets.
  - Must be many-one because there must be no more than one entity in that entity set to help ID it.
- In fact, must have a rounded arrow.
  - Because there must indeed be one entity to help ID it.

Key of a weak entity set

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

Good E/R Design

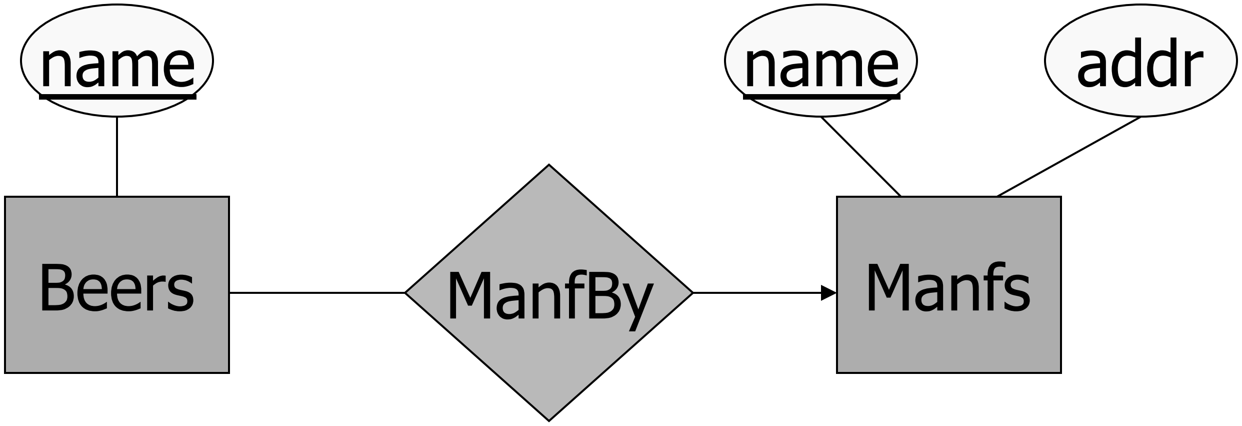
Summary

- There are always many different E/R diagrams one could come up with for a given domain.
- Some principles:
  1. Avoid redundancy.
  2. Don’t use an entity set when an attribute will do.
  3. Limit the use of weak entity sets.

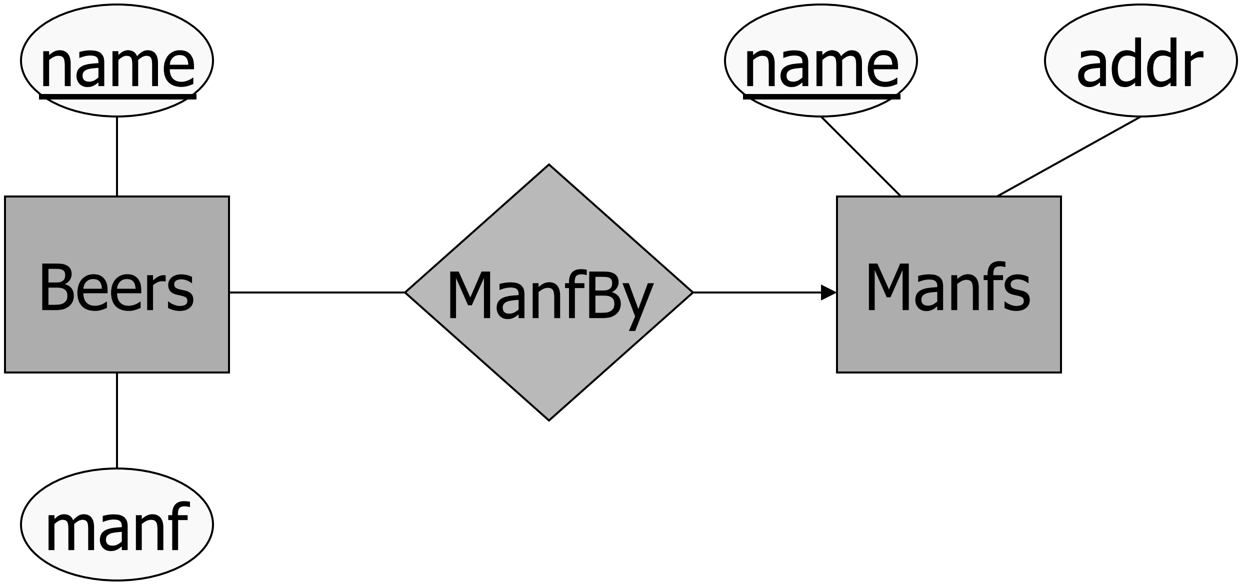
1. Avoid redundancy

- *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.
  - Recall anomalies due to FD's.

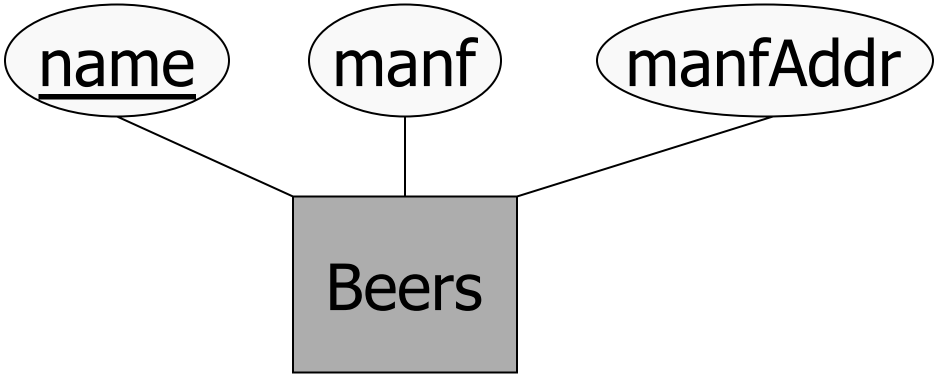
Example: Good



Example: Bad



Example: Bad



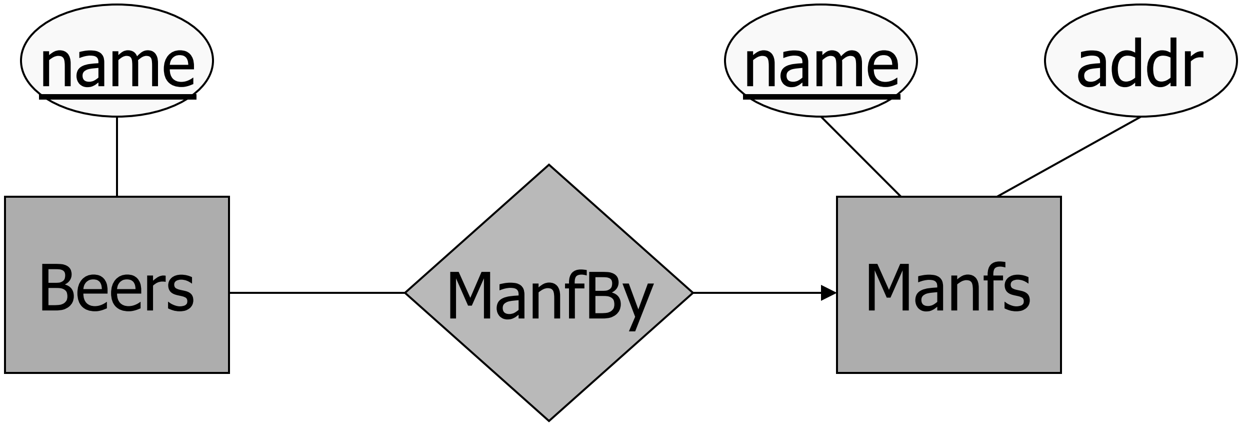
2. Prefer attributes over entity sets

- If an attribute will do, use it.
- You need an entity set if:
  - It is more than the name of something; it has at least one non-key attribute.

    or

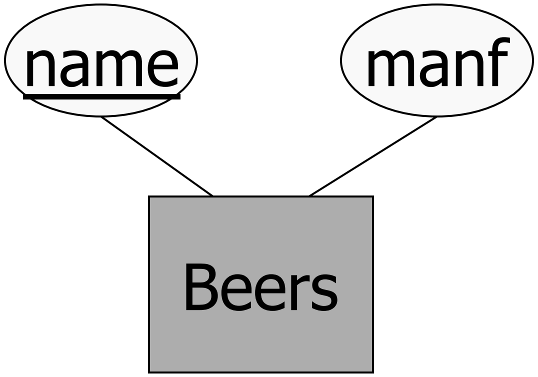
  - It is the "many" in a many-one or many-many relationship.

Example: Good



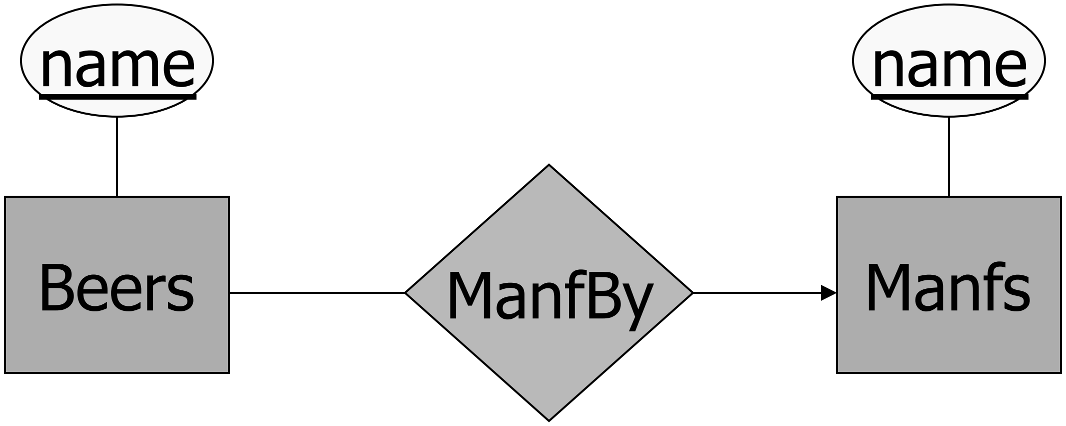
- Manfs deserves to be an entity set because of the nonkey attribute addr.
- Beers deserves to be an entity set because it is the “many” of the many-one relationship ManfBy.

Example: Good



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

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.

3. Limit use of weak entity sets

- Beginning database designers often overuse weak entity sets.
- It is usually better to create unique IDs in order to avoid weak 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.

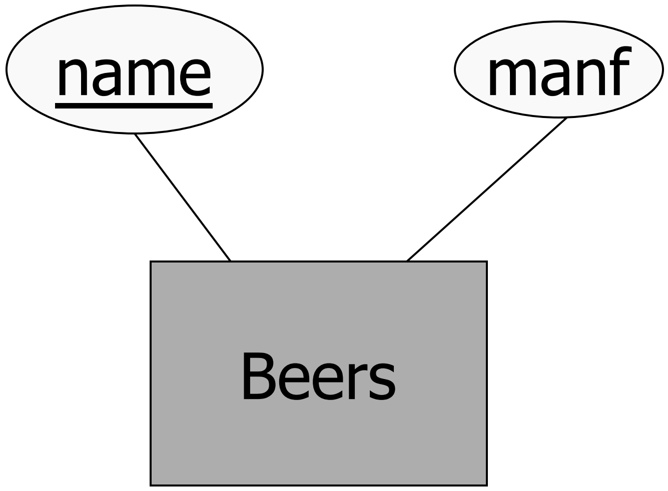
E/R Diagram ➔ Schema

Relations

The general idea:

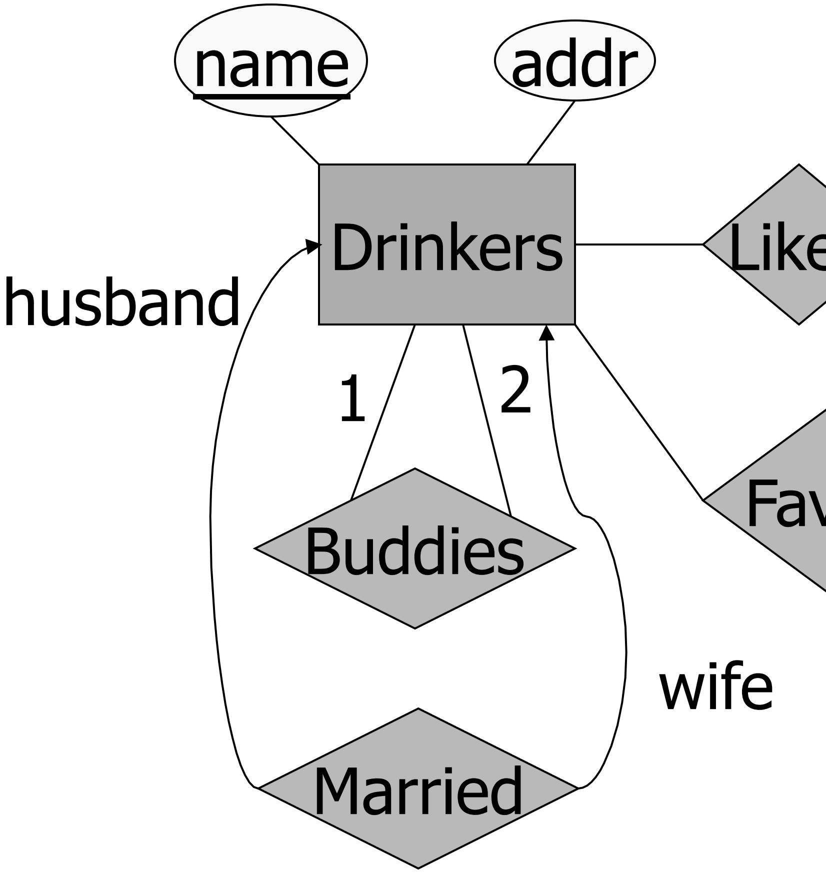
- Entity set -> relation.
  - Attributes of the entity set -> attributes of the relation.
  - If a weak entity set, include key attributes of the supporting entity set.
- Relationship -> relation whose attributes are only:
  - The keys of the connected entity sets.
  - Attributes of the relationship itself.

Entity Set -> Relation



Relation: Beers(name, manf)

Relationship -> Relation



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,* plus
  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).

Risk with Many-Many Relationships

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

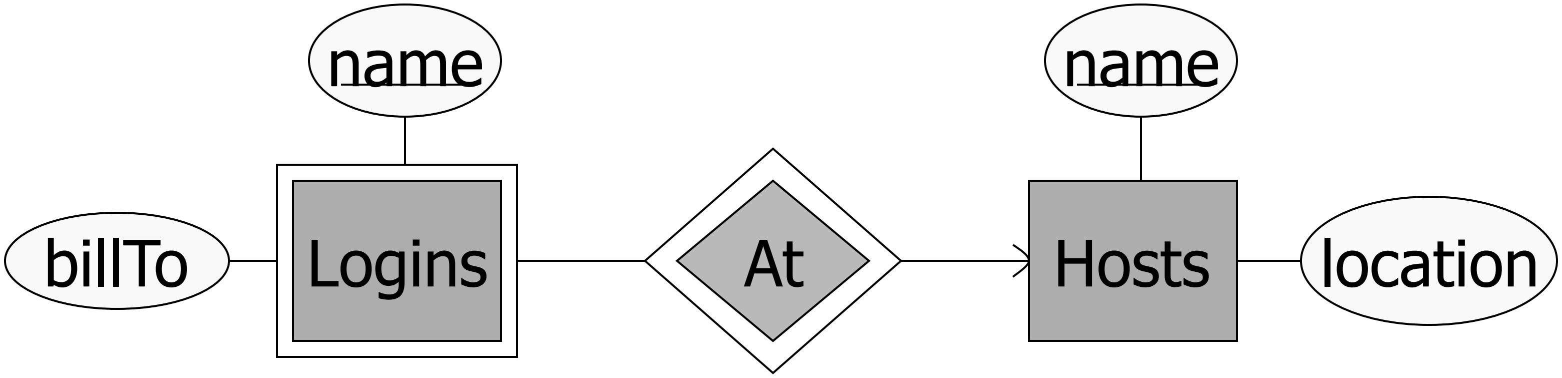
name	addr	beer
Sally	123 Maple	Bud
Sally	123 Maple	Miller

Redundancy

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 is not a relation; it becomes part of Logins