ENTITY-RELATIONSHIP MODEL

Overview of Database Design

- *Conceptual 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?

Purpose of E/R Model

- The Entity/Relationship (E/R) model allows us to sketch database schema designs.
  - Includes some constraints
- Schema designs are pictures called *entity-relationship diagrams*.
- Later: convert E/R designs to relational DB designs.

Credit: Renee J. Miller

Framework for E/R

- Design is a necessity.
- Management know they want a database, but they don’t know what they want in it.
- Sketching the key components is an efficient way to develop a working database.

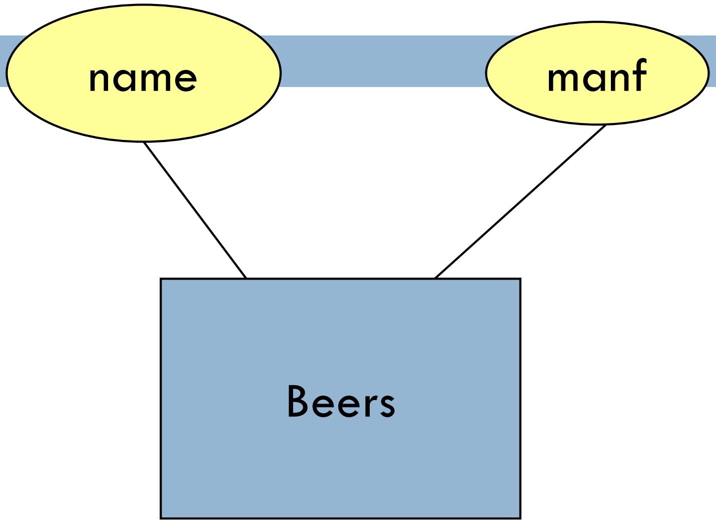
Entity Sets

- *Entity* = “thing” or object.
- *Entity set* = collection of similar entities.
  - Similar to a class in object-oriented languages.
- *Attribute* = property of an entity set.
  - Attributes are simple values, e.g. integers or character strings, not structs, sets, etc.
  - Each attribute has a *domain.*

E/R Diagrams

- In an entity-relationship diagram:
  - Entity set = rectangle.
  - Attribute = oval, with a line to the rectangle representing its entity set.
  - Notation varies: some textbooks represents attributes within the (entity) rectangle

Example

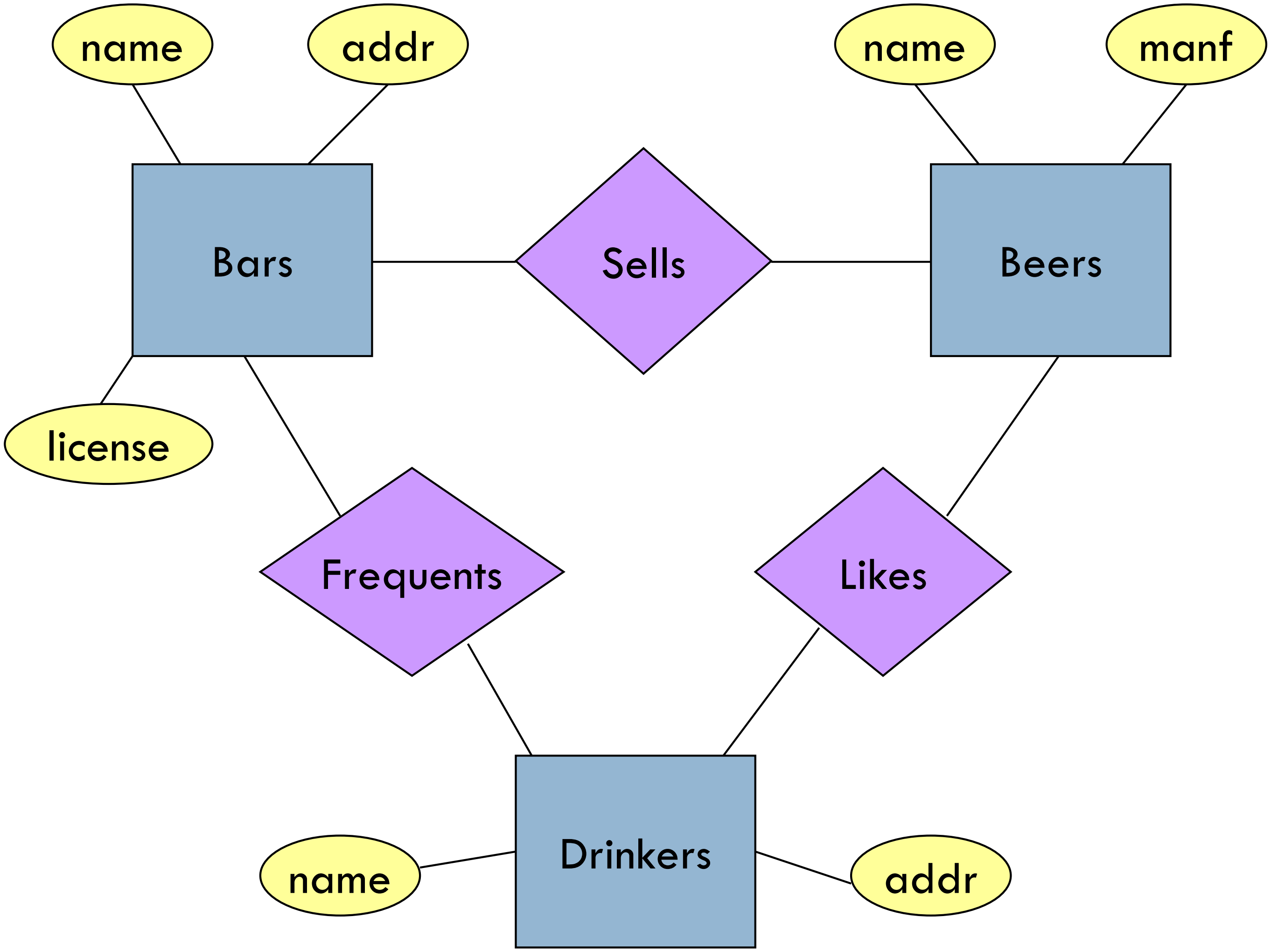


- Entity set Beers has two attributes, name and manf (manufacturer).
- Each Beers entity has values for these two attributes, e.g. (Bud, Anheuser-Busch)

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



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

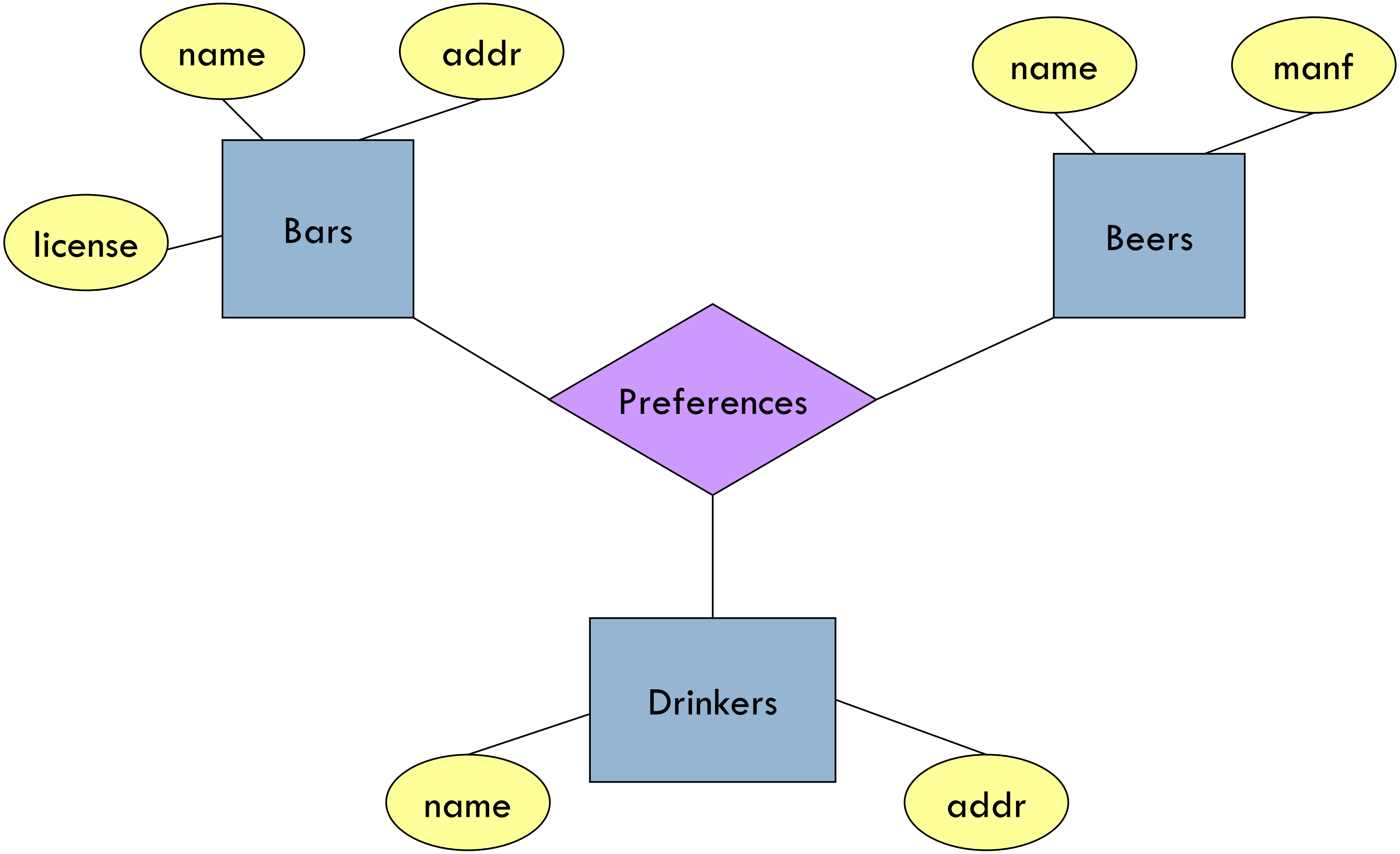
- For the relationship Sells, we might have a relationship set like:

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



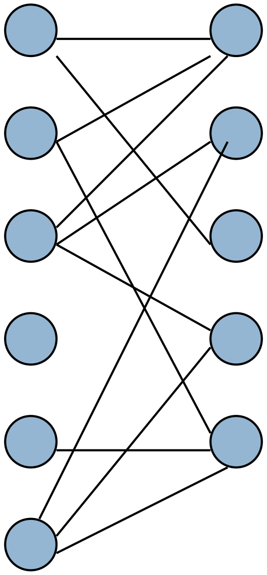
A Typical Relationship Set

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

Many-Many Relationships

- Focus: binary relationships, such as Sells between Bars and Beers.
- In a *many-many* relationship, an entity of either set can be connected to many entities of the other set.
    - E.g., a bar sells many beers; a beer is sold by many bars.

In Pictures:



many-many

Note: each line is an instance of the binary relationship

Many-One Relationships

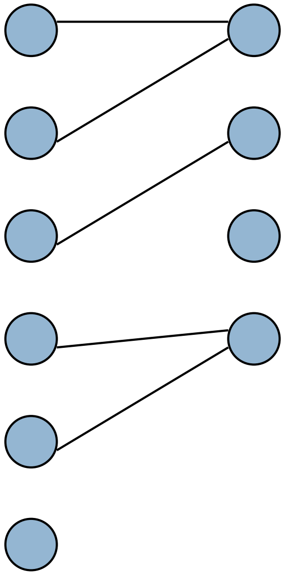
- □ Some binary relationships are *many -one* from one entity set to another.
- □ Each entity of the first set is connected to at most one entity of the second set.
- □ But an entity of the second set can be connected to zero, one, or many entities of the first set.

In Pictures:

Bars

Beers

Drinkers



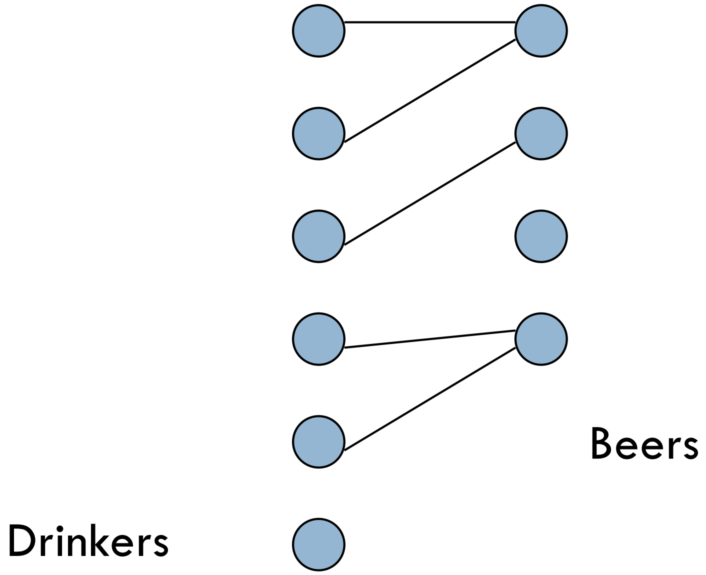
many-one

(Partial) Function on entity set

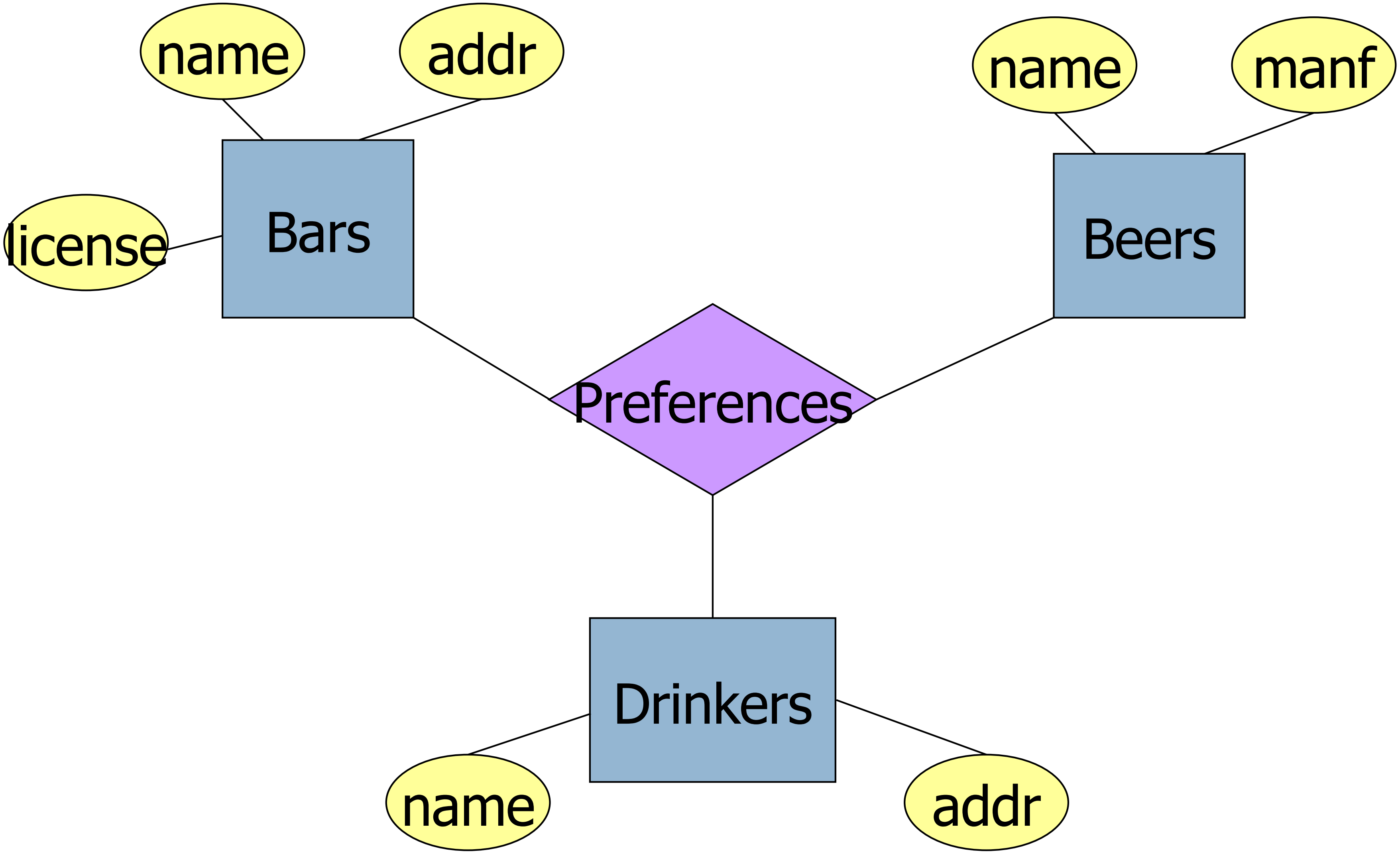
Example: Many-One Relationship

- Favourite, from Drinkers to Beers is many-one.

- A drinker has at most one favourite beer.
- But a beer can be the favorite of any number of drinkers, including zero.



Example: 3-Way Relationship



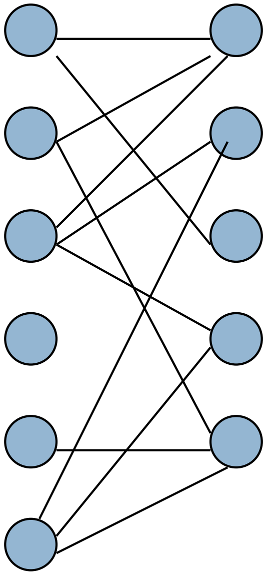
A Typical Relationship Set

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

Many-Many Relationships

- Focus: binary relationships, such as Sells between Bars and Beers.
- In a *many-many* relationship, an entity of either set can be connected to many entities of the other set.
  - E.g., a bar sells many beers; a beer is sold by many bars.

In Pictures:



many-many

Note: each line is an instance of the binary relationship

Many-One Relationships

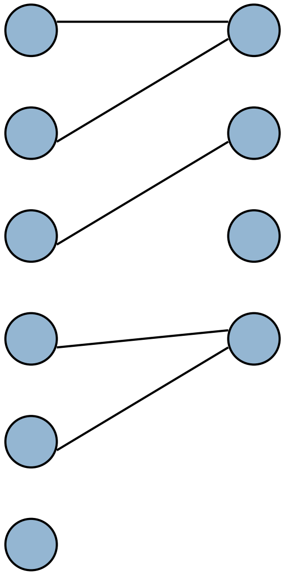
- Some binary relationships are *many -one* from one entity set to another.
- Each entity of the first set is connected to at most one entity of the second set.
- But an entity of the second set can be connected to zero, one, or many entities of the first set.

In Pictures:

Bars

Beers

Drinkers



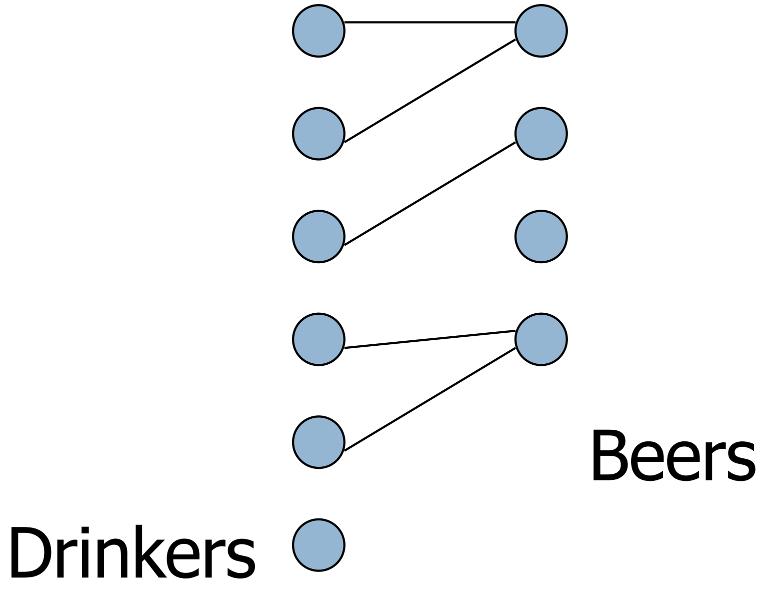
many-one

(Partial) Function on entity set

Example: Many-One Relationship

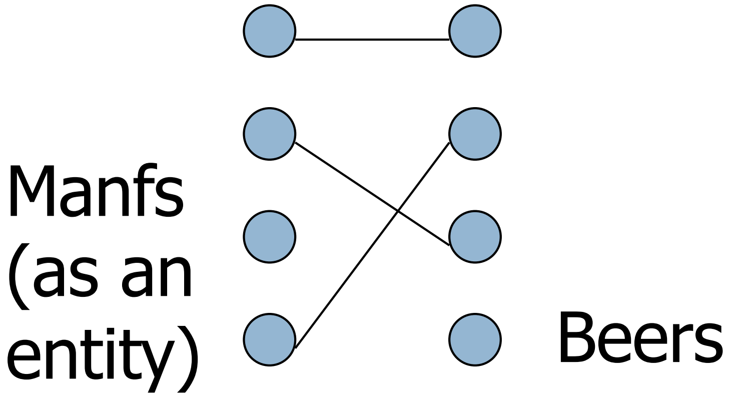
- Favourite, from Drinkers to Beers is many-one.

- A drinker has at most one favourite beer.
- But a beer can be the favorite of any number of drinkers, including zero.



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 is the best seller for 0 or 1 manufacturers, and no manufacturer can have more than one best-seller (assume no ties).

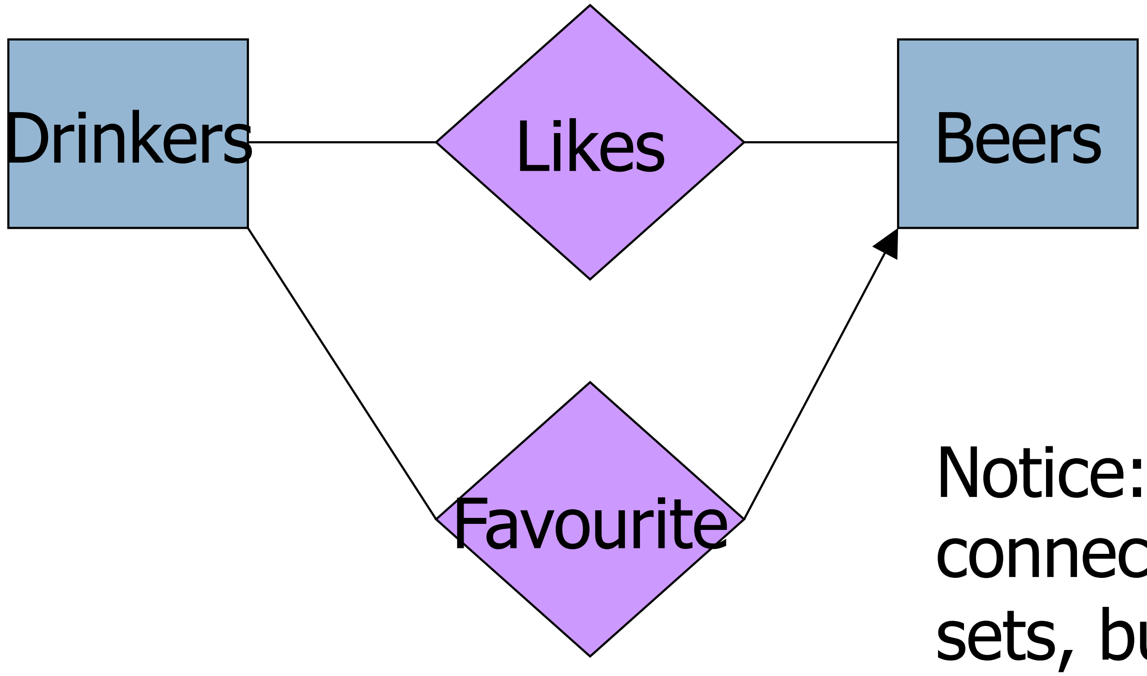


Representing “Multiplicity”

- Show a many-one relationship by an arrow entering the “one” side.
  - “at most one”
- Show a one-one relationship by arrows entering both entity sets.

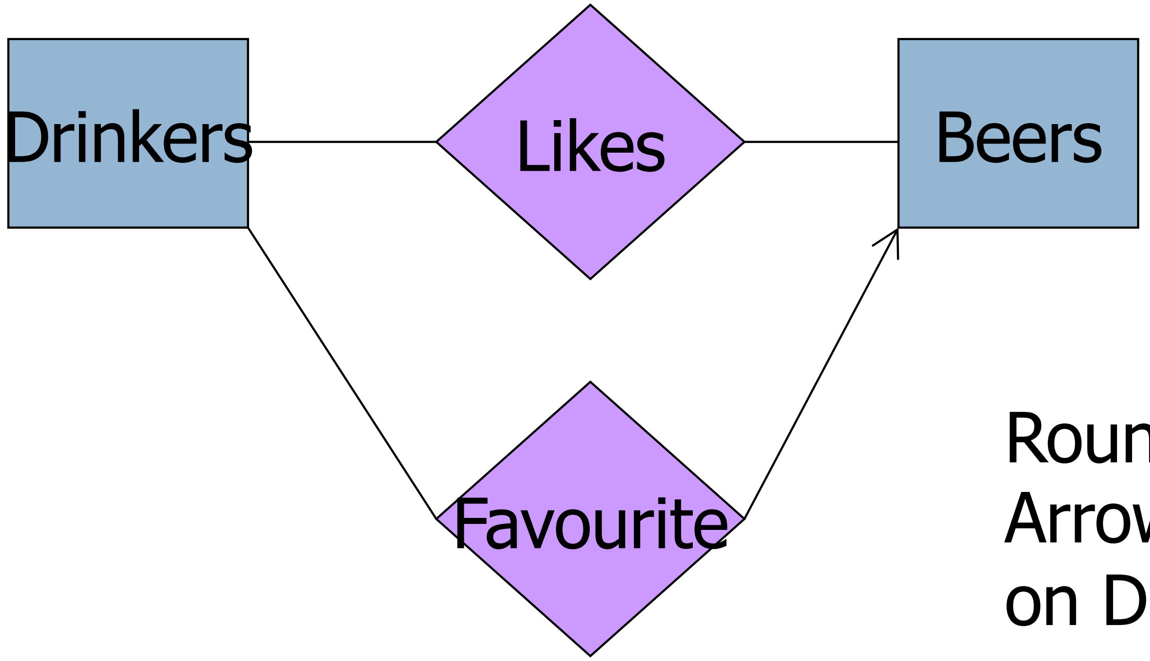
Rounded (open) arrow = “exactly one,” i.e., each entity of the first set is related to exactly one entity of the target set.

Example: Many-One Relationship



Notice: two relationships connect the same entity sets, but are different.

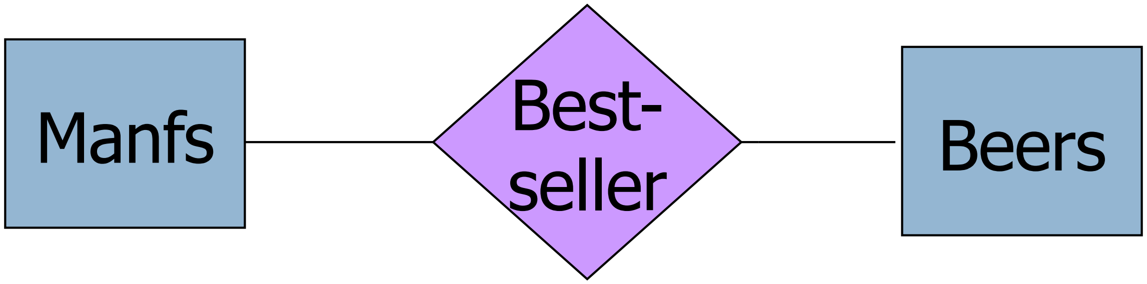
Example: Many-One Relationship



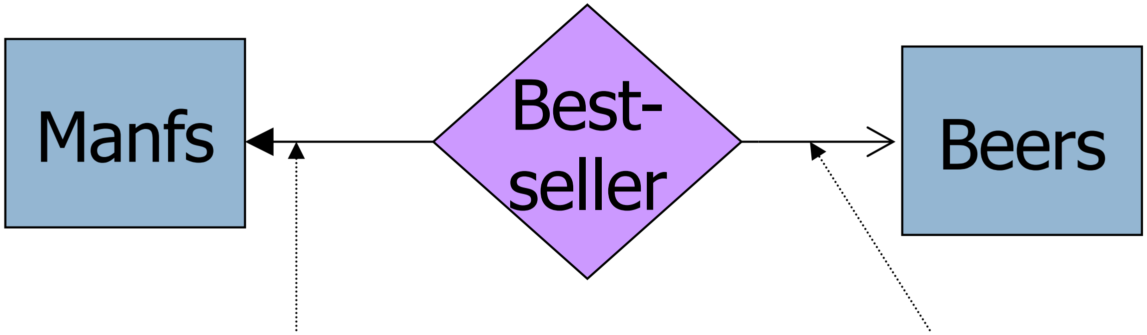
Rounded (open)
Arrow = total function
on Drinkers

Example: One-One Relationship

- Consider Best-seller between Manfs and Beers.
- Some beers are not the best-seller of any manufacturer
- But a beer manufacturer has to have a best-seller.



In the E/R Diagram



A beer is the best-seller for 0 or 1 manufacturer.

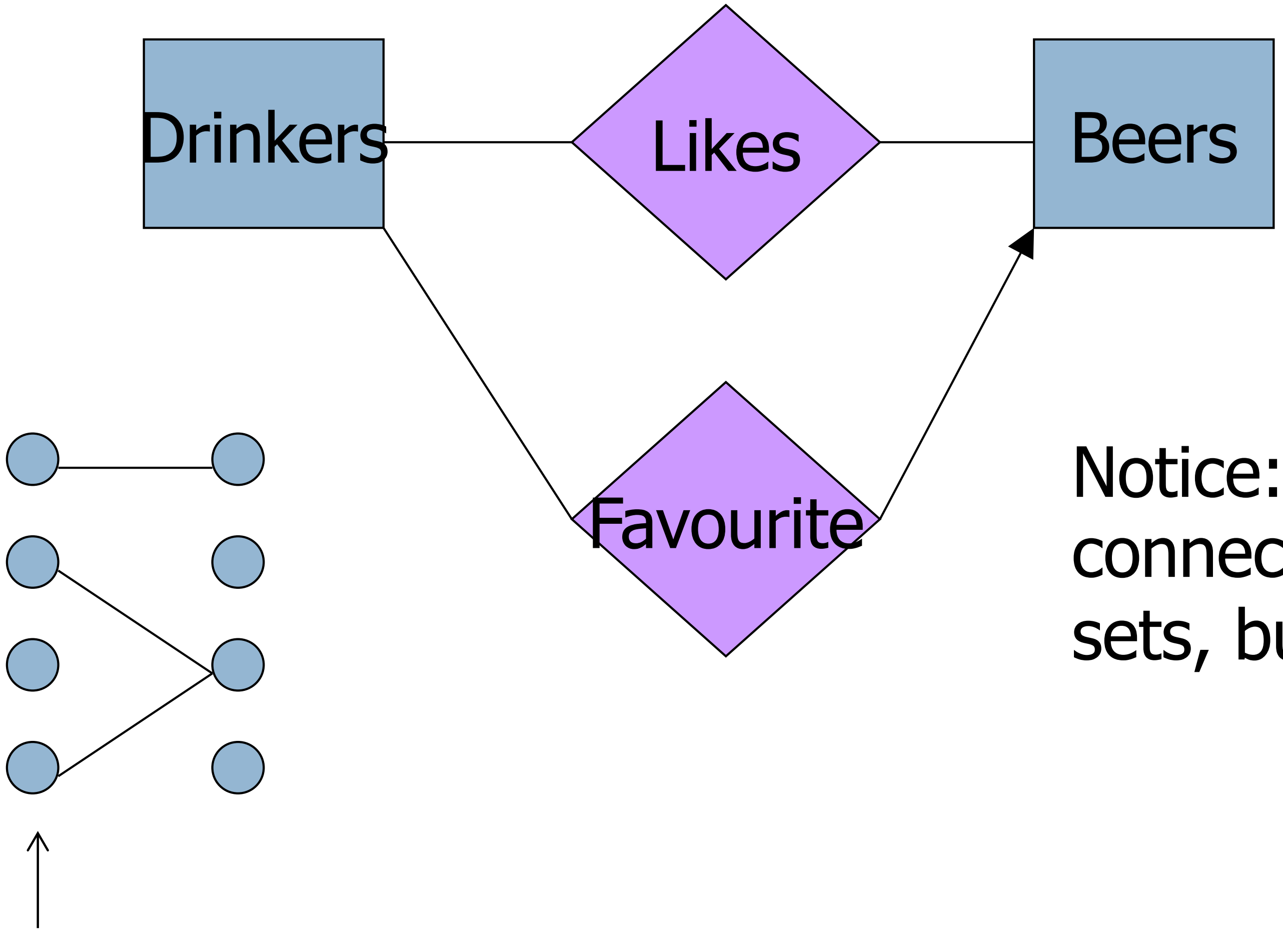
A manufacturer has exactly one best seller.

Participation Constraints

- Does every student have to take a course?
  - If so, this is a *participation constraint*: the participation of Students in Enrolled is said to be *total* (vs. *partial*).
  - Every *sid* value in Students table must appear in a row of the Enrolled table (with a non-null *sid* value!)

- Textbook notation: total participation represented by a thick (bolded) line originating from entity

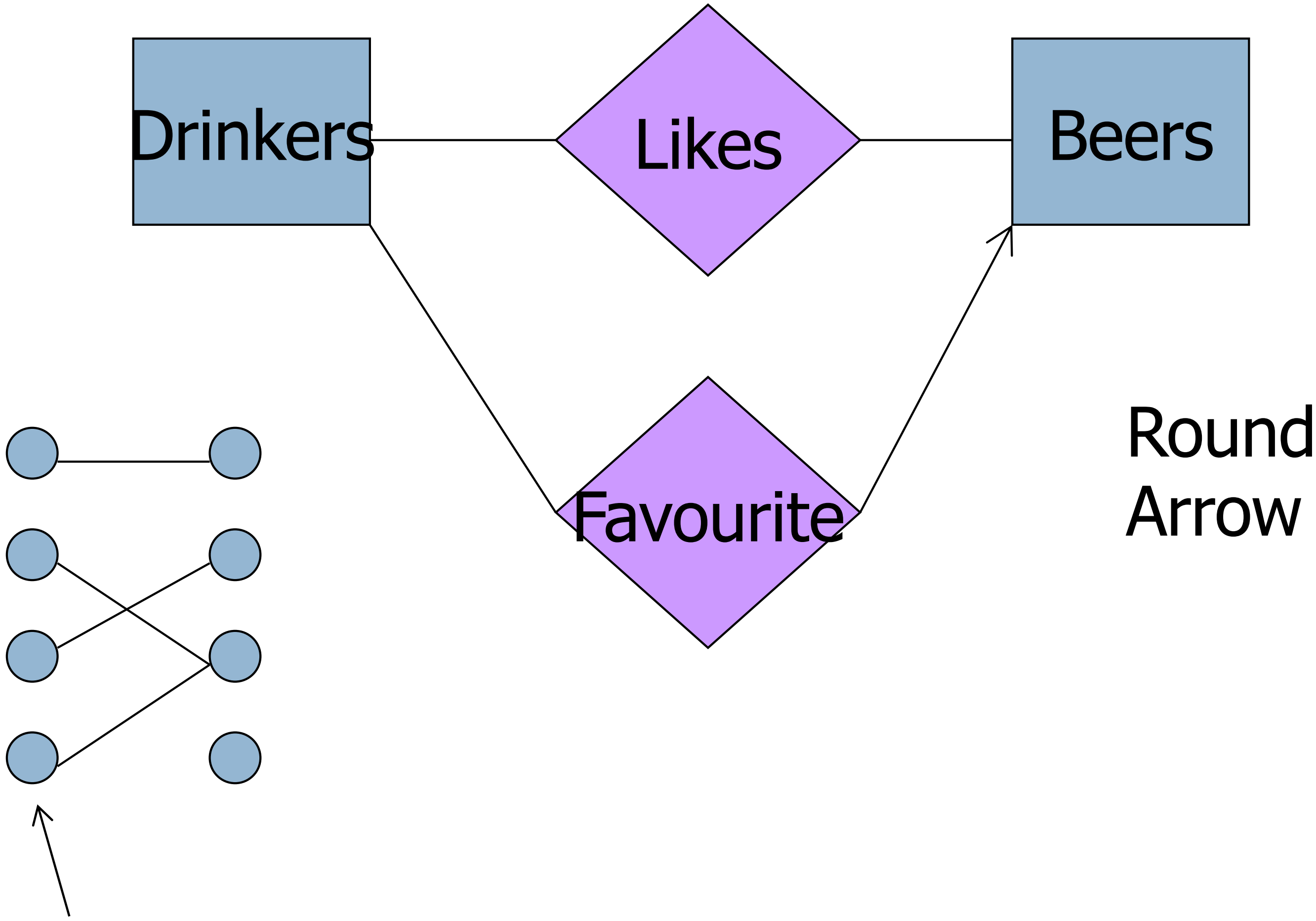
Example: Many-One Relationship



Notice: two relationships connect the same entity sets, but are different.

Participation of this entity is partial

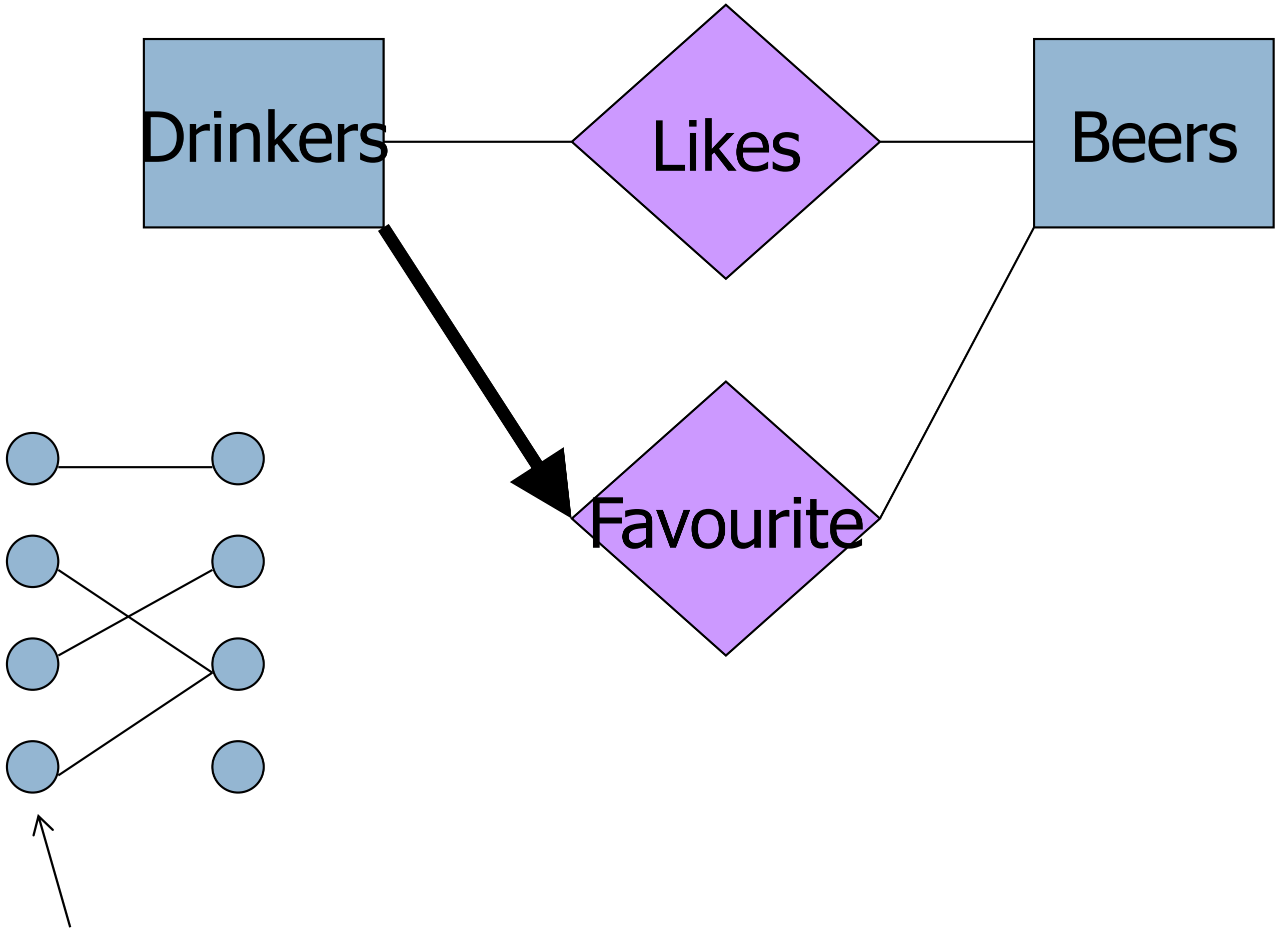
Example: Many-One Relationship



Rounded (open)
Arrow = total function

Participation of this entity is total

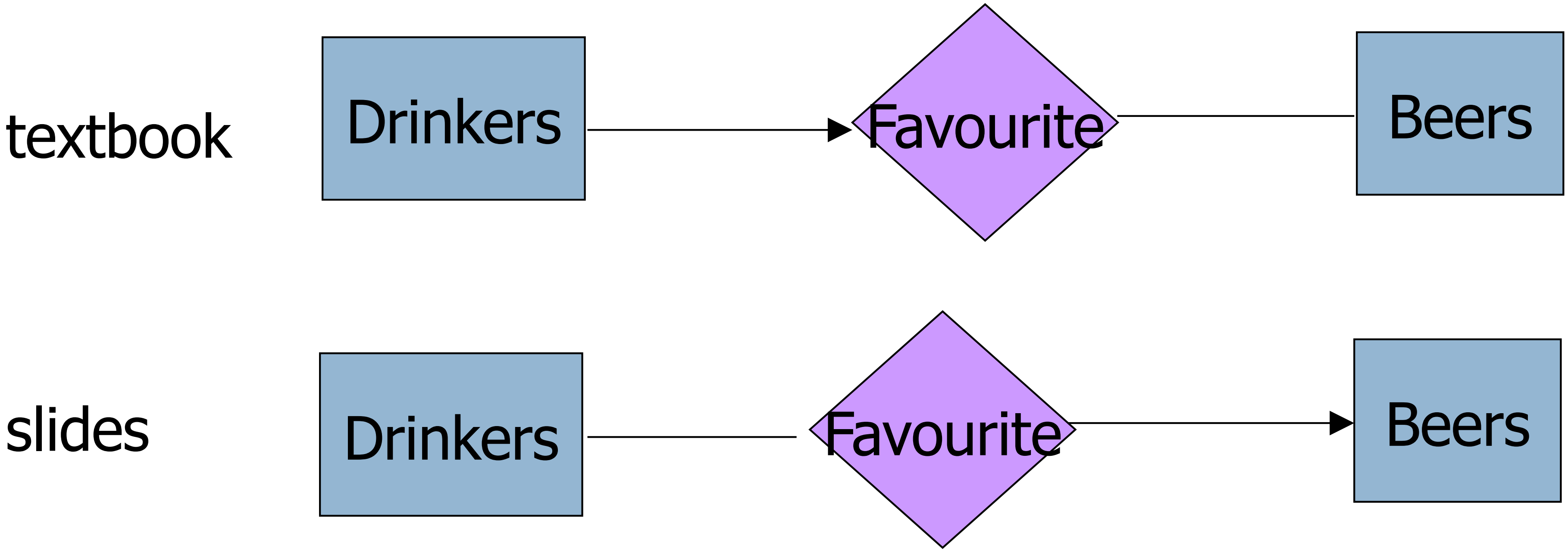
Alternative (Textbook) Notation



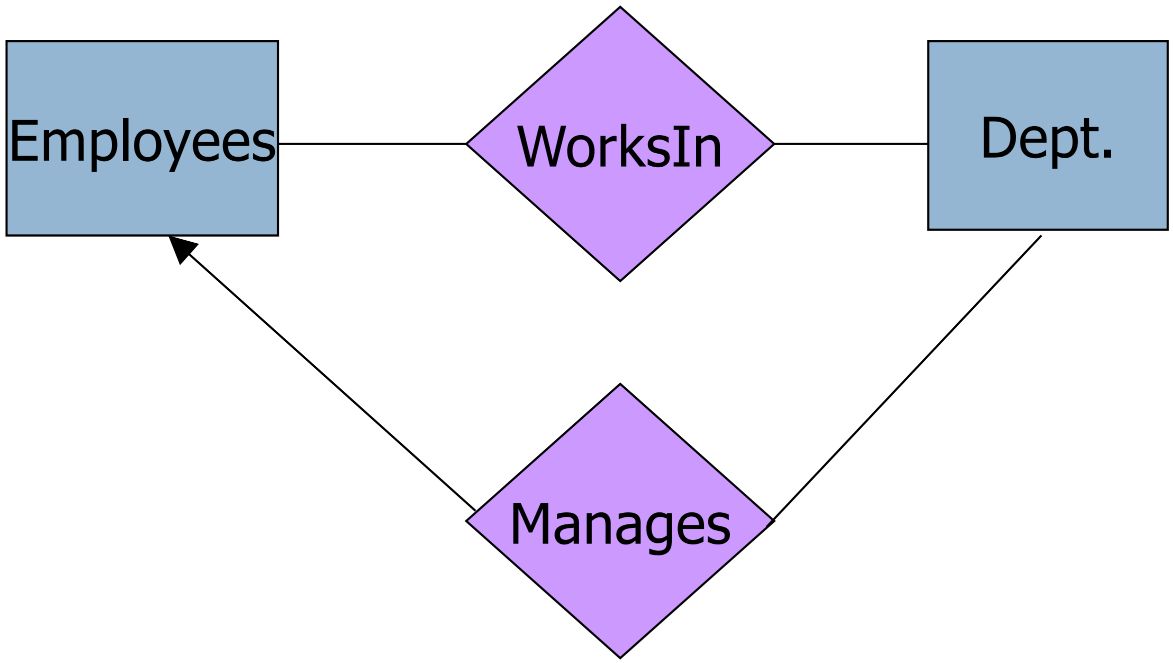
Participation of this entity is total

Notation

- Be consistent with your chosen notation!



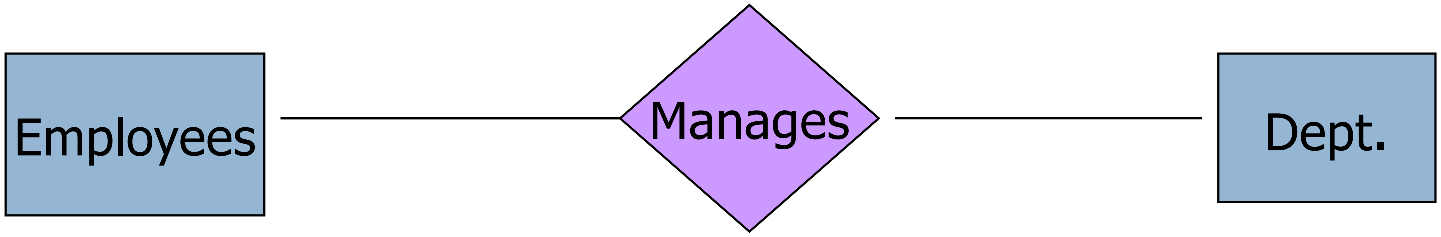
Key Constraints

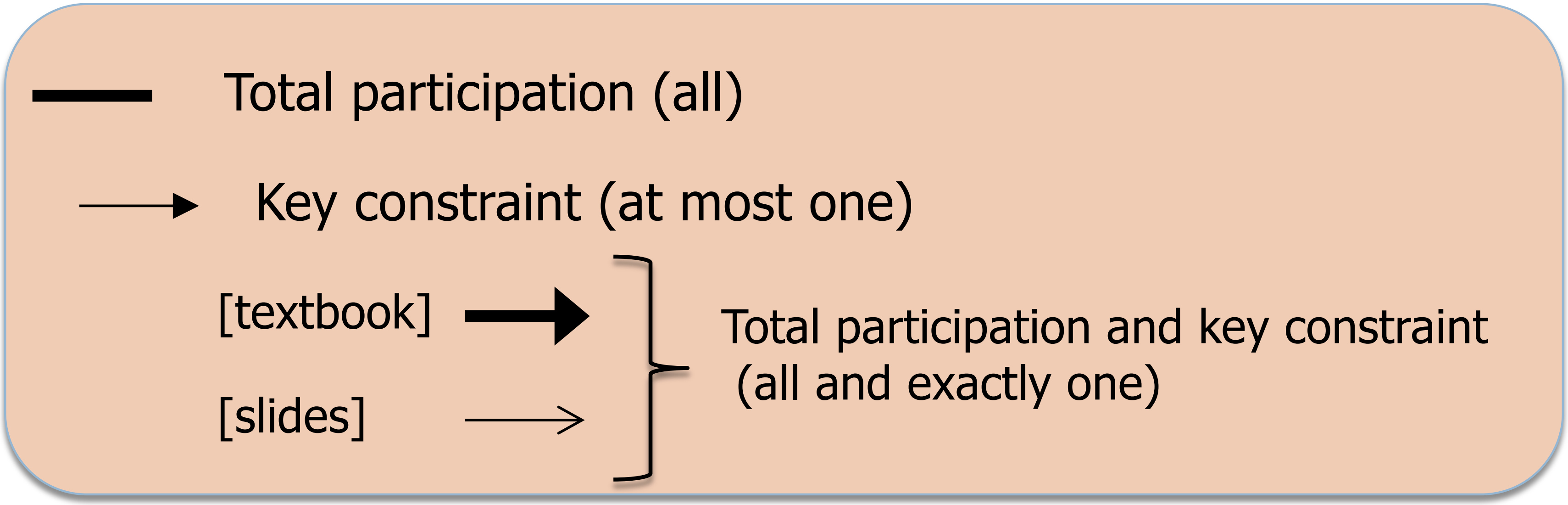


- Many-many: “An employee can work in many depts, and a dept. can have many employees
- One-many: A dept has at most one manager, and employees can manage many departments

Participation Constraints

- Does every dept. have to have a manager?
  - If yes, then every dept. must appear in the manages relation: total participation (vs. partial)





Total participation (all)

Key constraint (at most one)

[textbook]

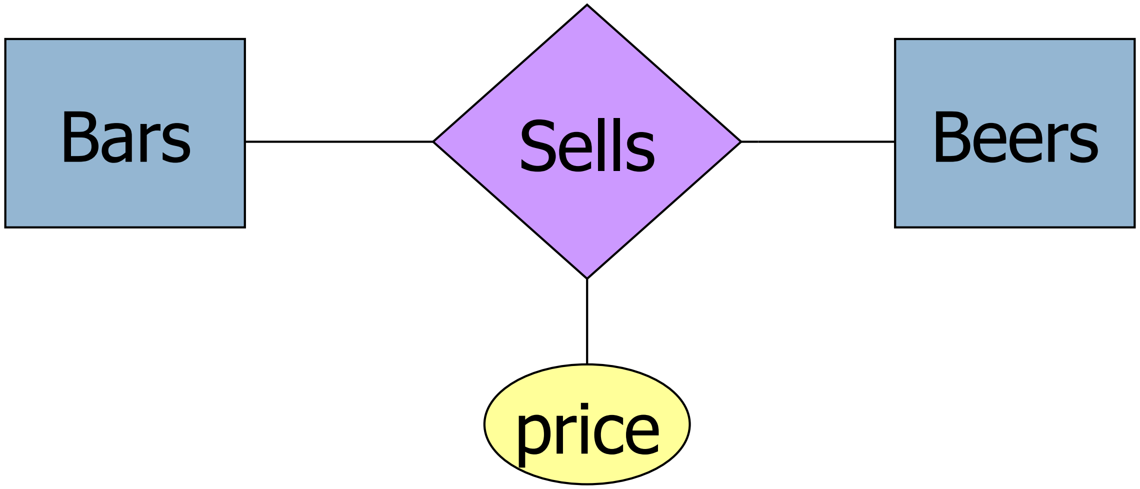
[slides]

Total participation and key constraint (all and exactly one)

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.
E.g., "The price of Miller beer at Joe's bar"

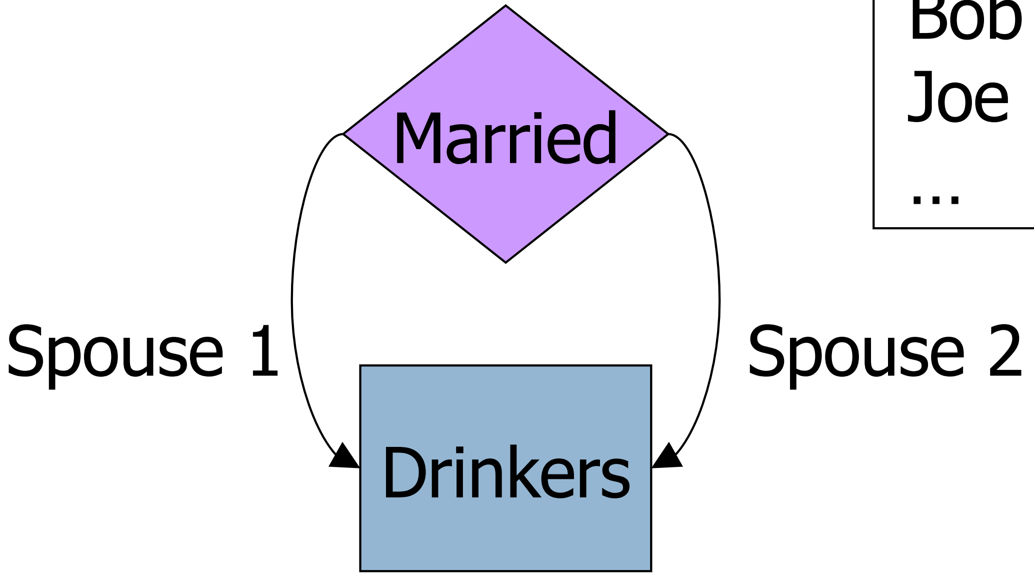
Roles

- 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*.

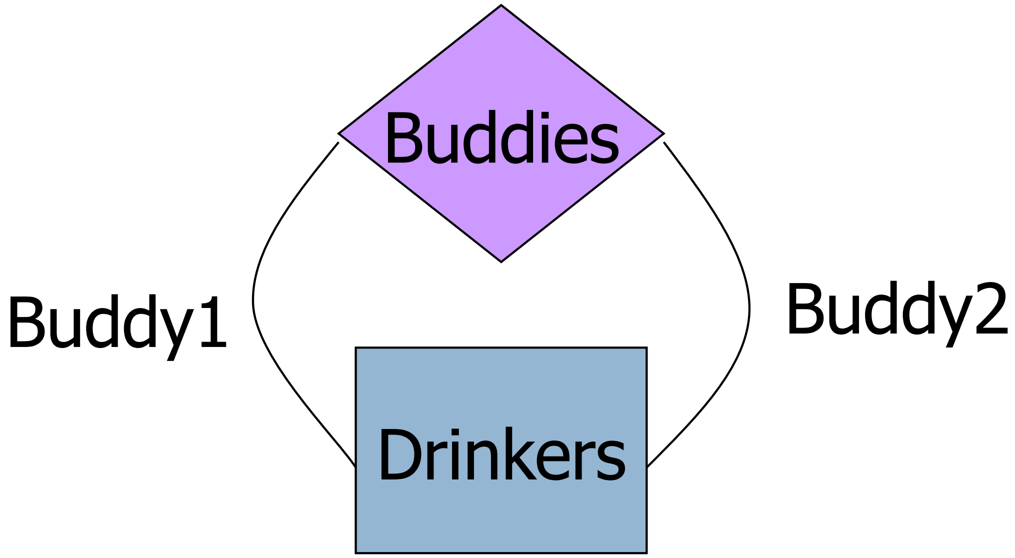
Example: Roles

Relationship Set

Spouse1	Spouse2
Bob	Ann
Joe	Sue
…	…



Example: Roles



Relationship Set

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

Subclasses

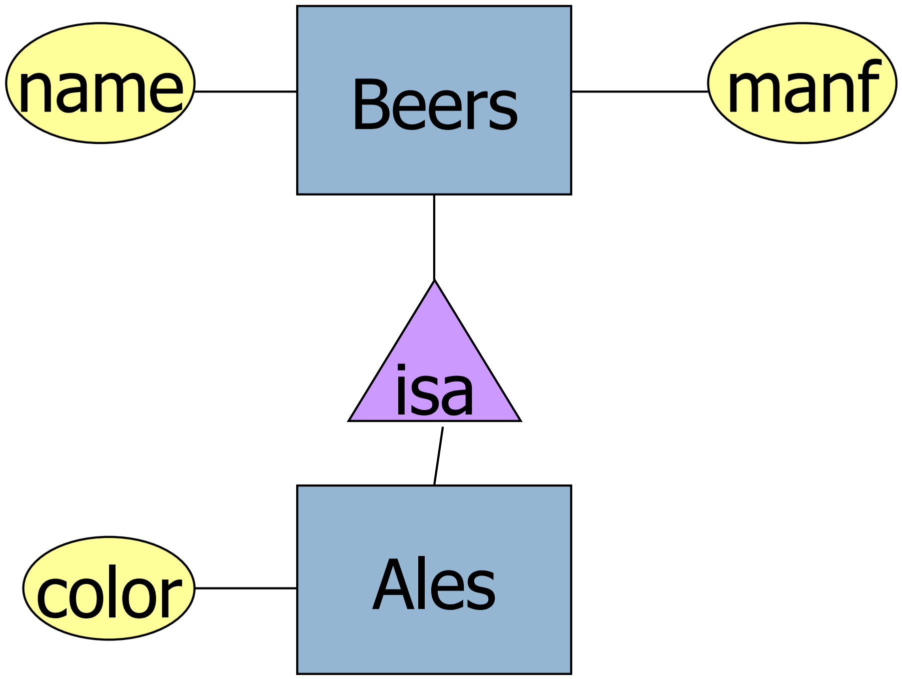
- *Subclass* = special case = more properties.
- Example: Ales are a kind of beer.
  - Not every beer is an ale, but some are.
  - Let us suppose that in addition to all the *properties* (attributes and relationships) of beers, ales also have the attribute color.

Subclasses in E/R Diagrams

- isa triangles indicate the subclass relationship.
  - Point to the superclass.

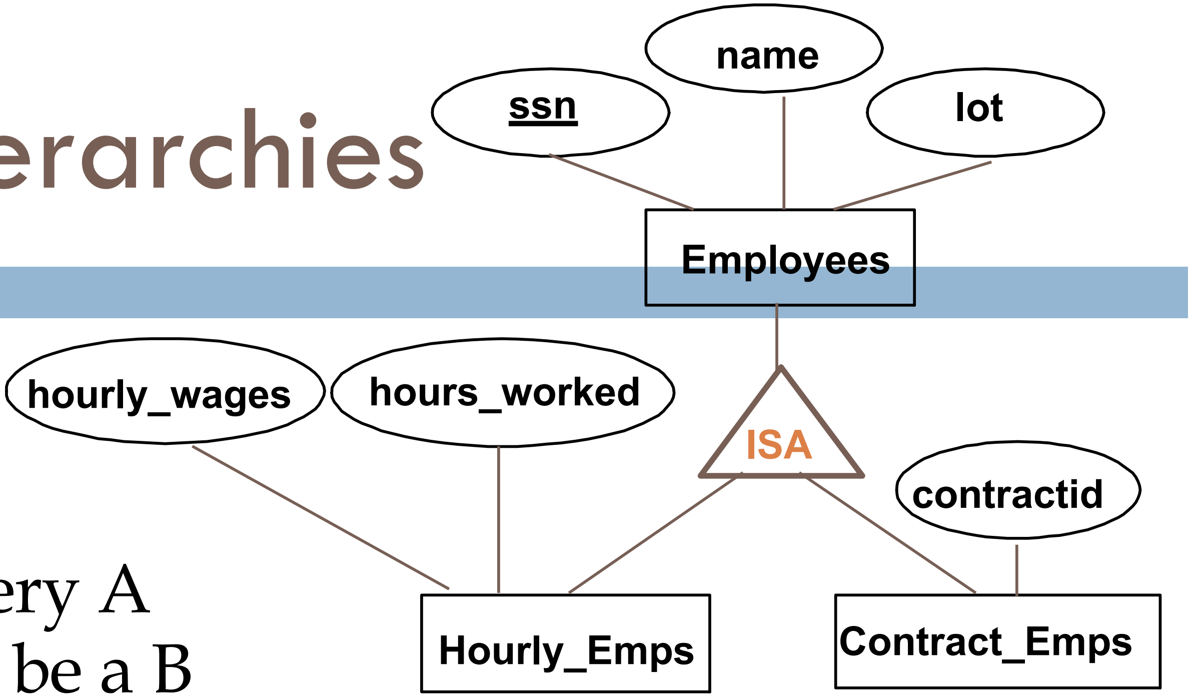
- Reasons for using isa:
  - To add descriptive attributes specific to a subclass.
  - To identify entities that participate in a relationship.

Example: Subclasses



Assume subclasses form a tree.

ISA (`is a') Hierarchies



- As in C++, 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 two sub-classes contain the same entity? E.g., Can Joe be an Hourly_Emps as well as a Contract_Emps entity?

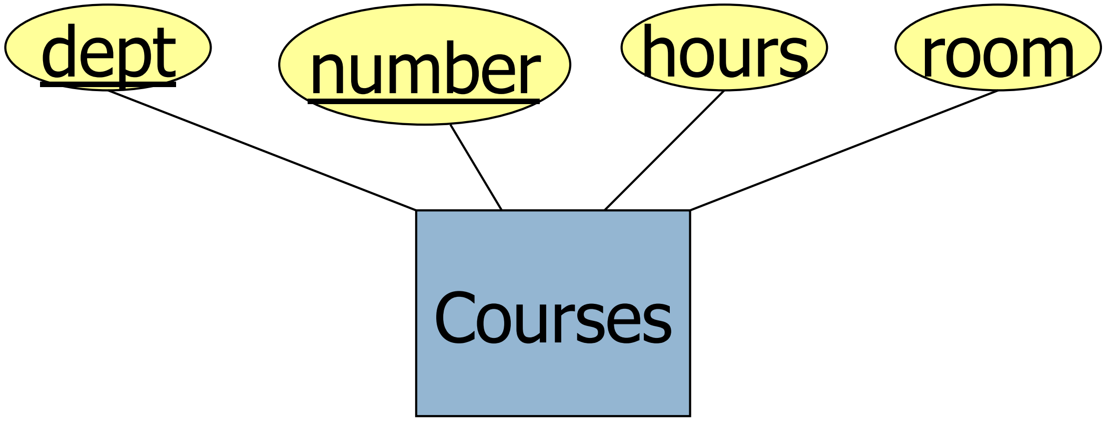
□ *Covering constraints*: Does every Employees entity have to be an Hourly_Emps or a Contract_Emps entity?

Keys

- 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.
- Underline the key attribute(s).

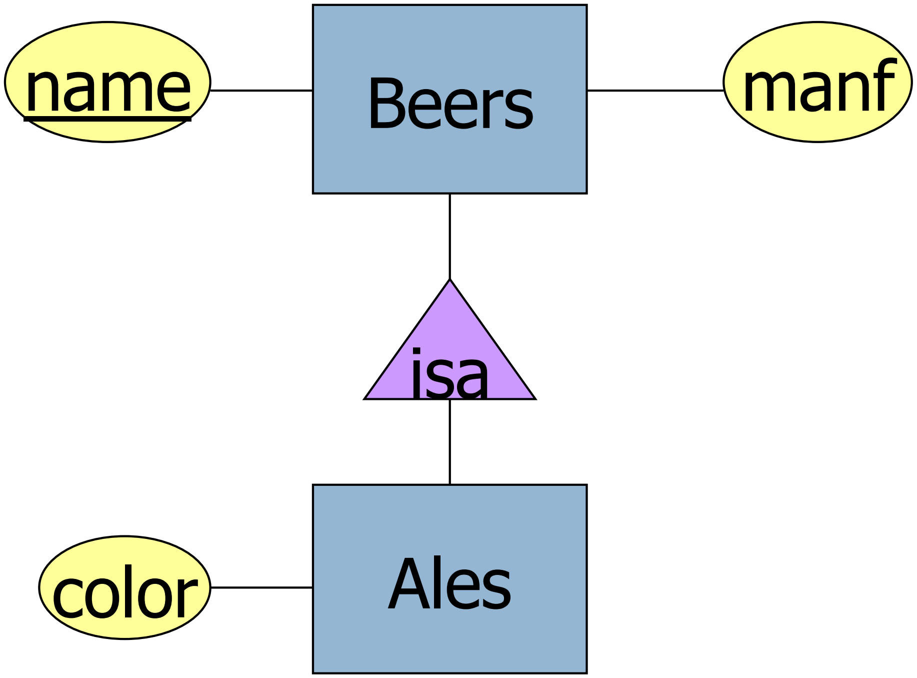
Example: a Multi-attribute Key



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

Keys

In an Isa hierarchy, only the root entity set has a key, and it must serve as the key for all entities in the hierarchy.



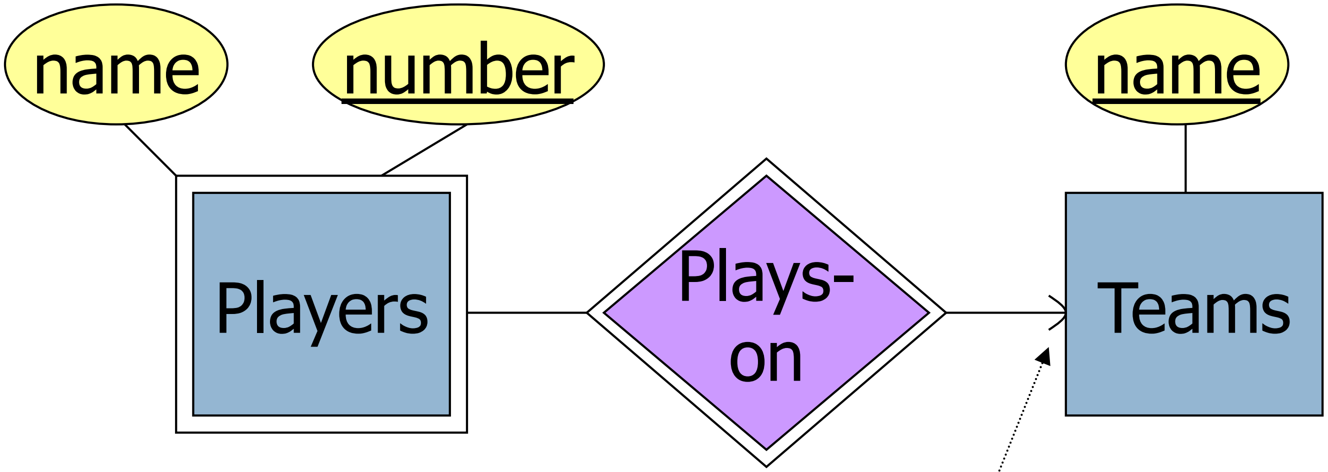
Weak Entity Sets

- 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.

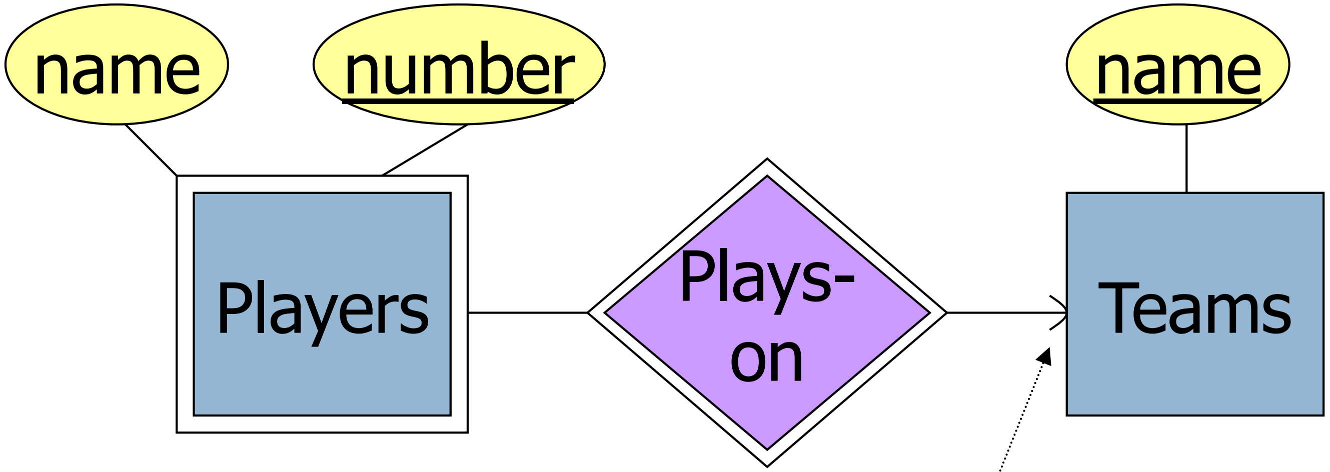
Weak Entity Sets

- 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.

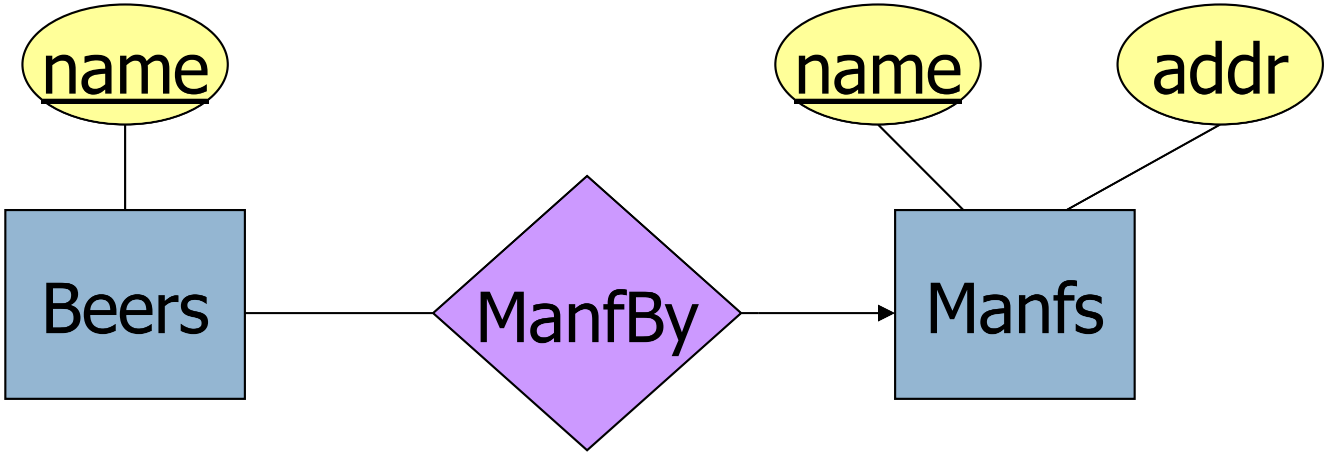
Design Techniques

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

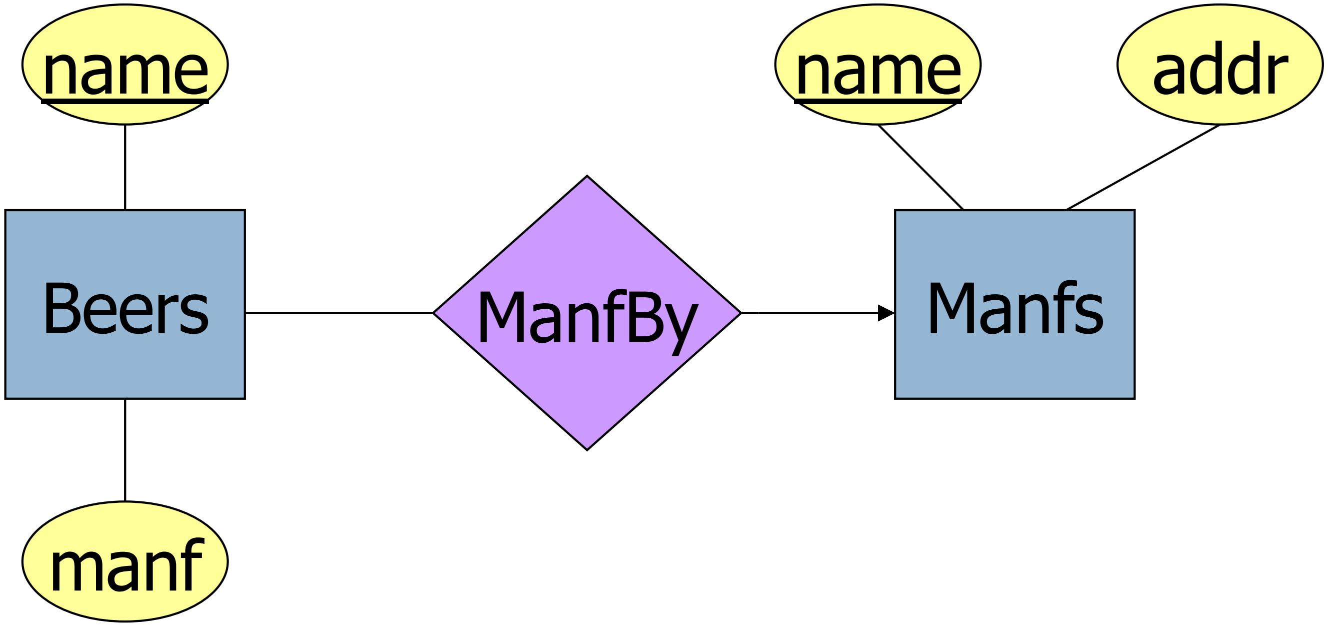
- *Redundancy* = saying the same thing in two (or more) different ways.
- Wastes space and (more importantly) encourages inconsistency.
  - Two representations of the same fact become inconsistent if we change one and forget to change the other.

Example: Good



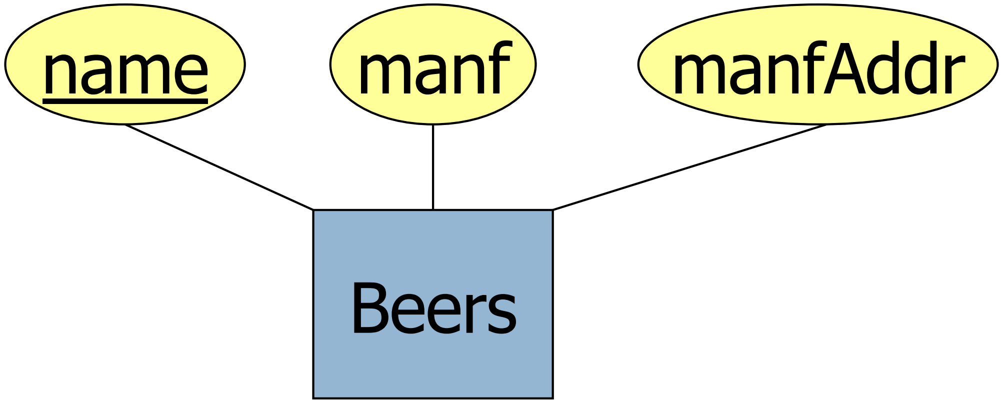
This design gives the address of each manufacturer exactly once.

Example: Bad



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

Example: Bad

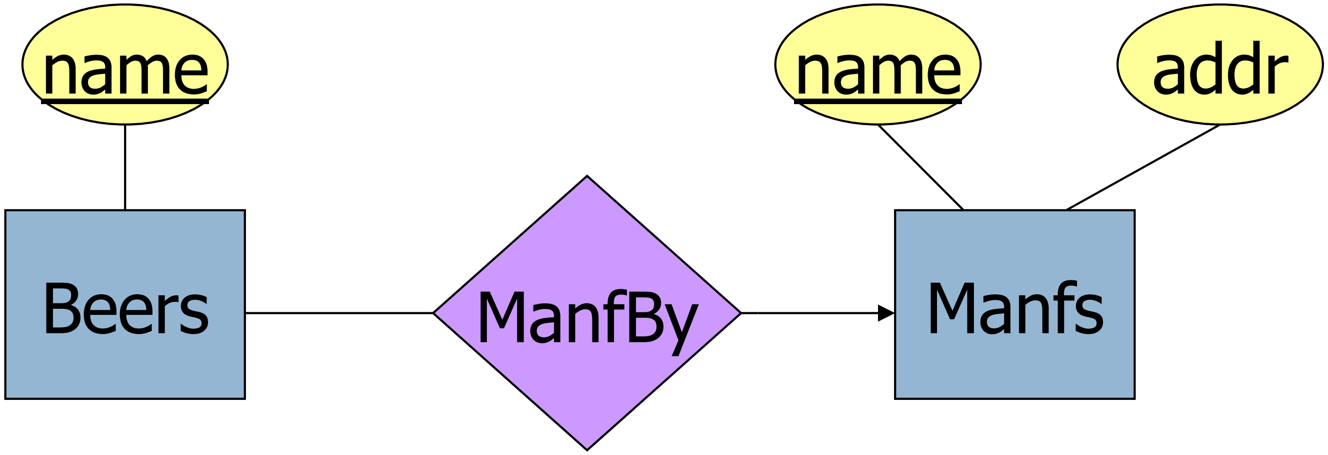


This design repeats the manufacturer’s address once for each beer and loses the address if there are temporarily no beers for a manufacturer.

Entity Sets Versus Attributes

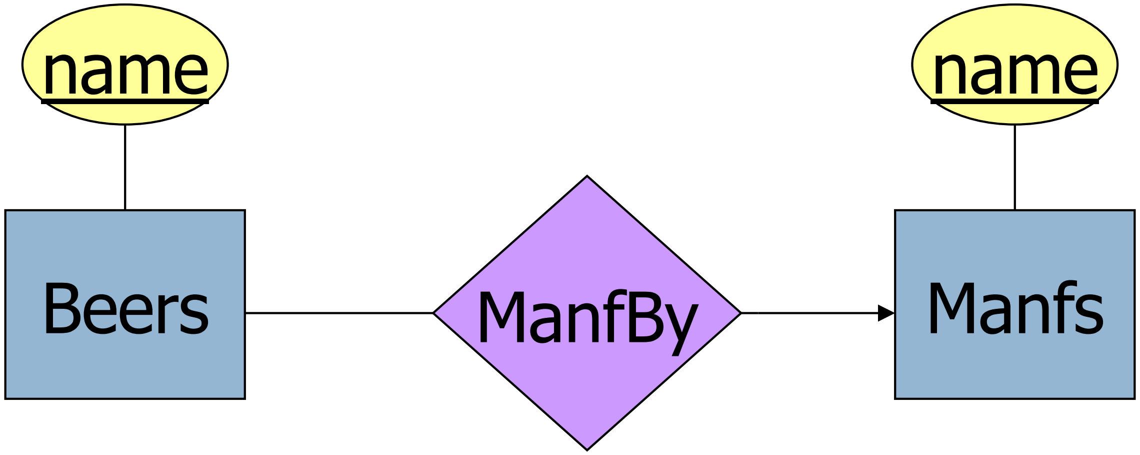
- An entity set should satisfy at least one of the following conditions:
    - It is more than the name of something; it has at least one non-key attribute. OR
    - It is the “many” in a many-one or many-many relationship.
- Depends on the application requirements:
    - If we have several addresses 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



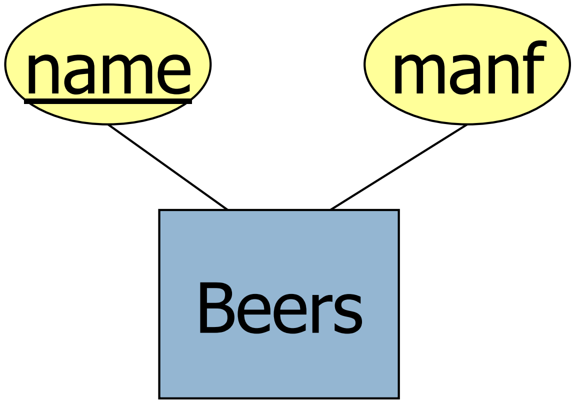
- Manfs deserves to be an entity set because of the nonkey attribute addr.
- Beers deserves to be an entity set because it is the “many” of the many-one relationship ManfBy.

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.

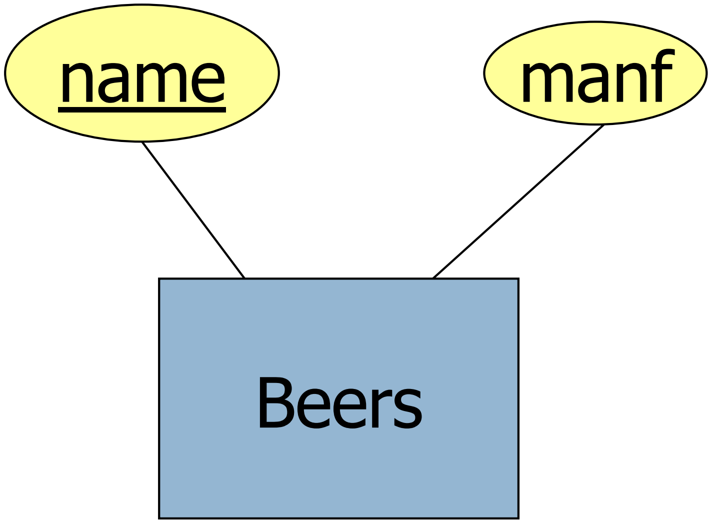
When Do We Need Weak Entity Sets?

- 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

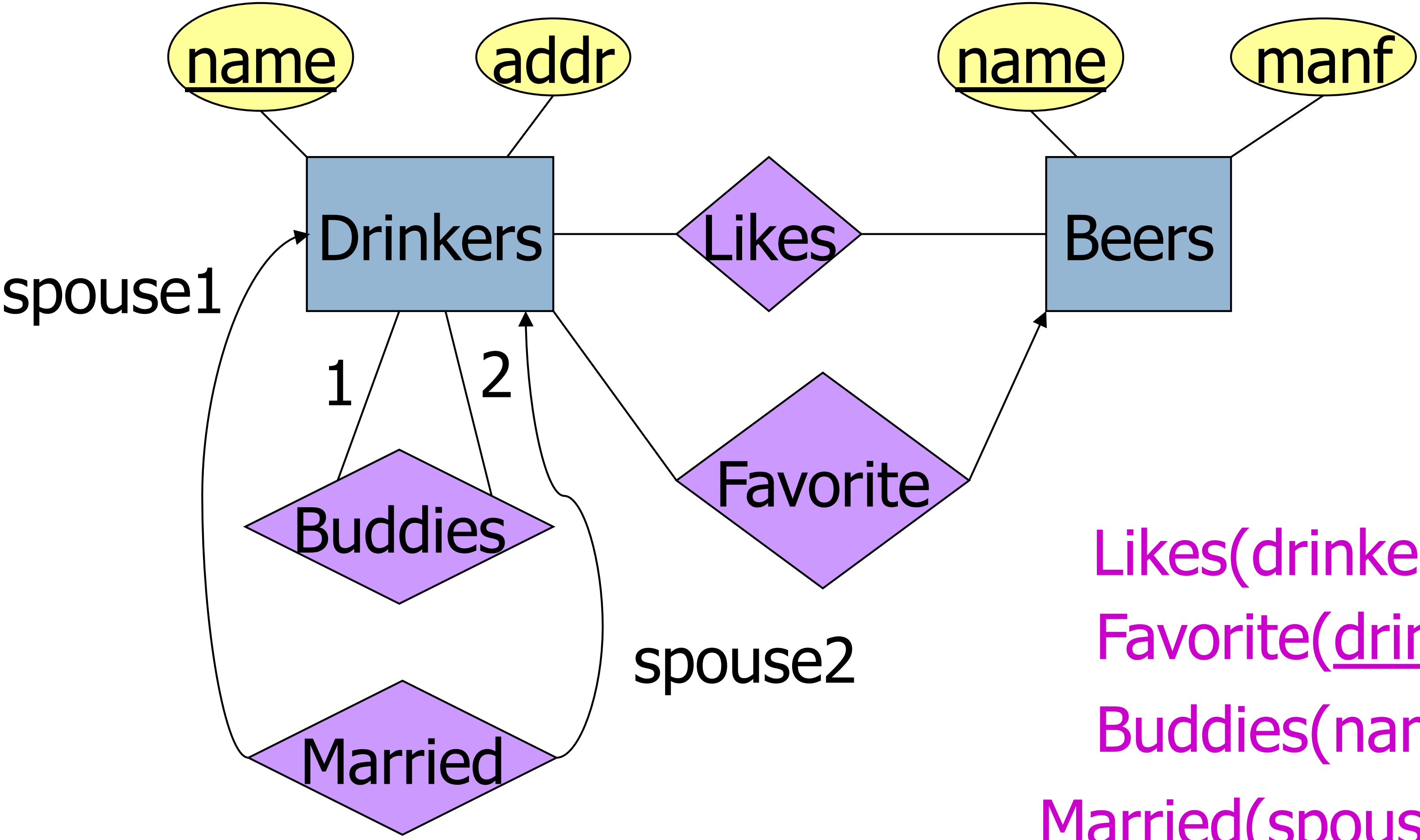
- Entity set -> relation.
  - Attributes -> attributes.
- Relationships -> relations whose attributes are only:
  - The keys of the connected entity sets.
  - Attributes of the relationship itself.

Entity Set -> Relation



Relation: Beers(name, manf)

Relationship -> Relation



Likes(drinker, beer)

Favorite(drinker, beer)

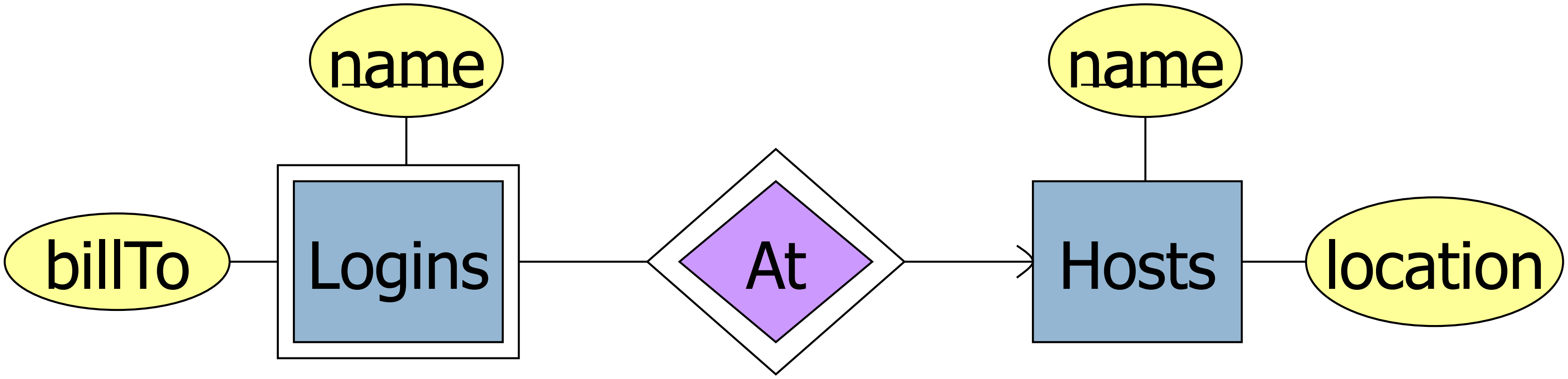
Buddies(name1, name2)

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)

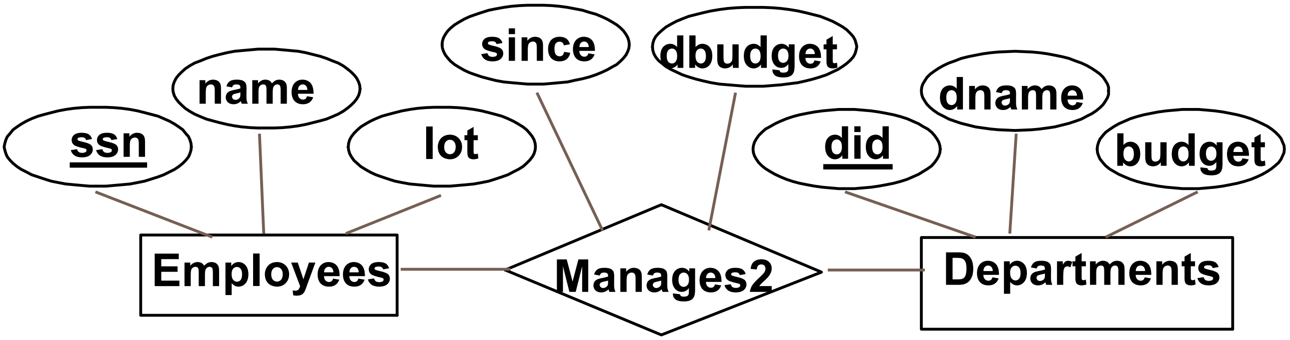
Logins(loginName, hostName, billTo)

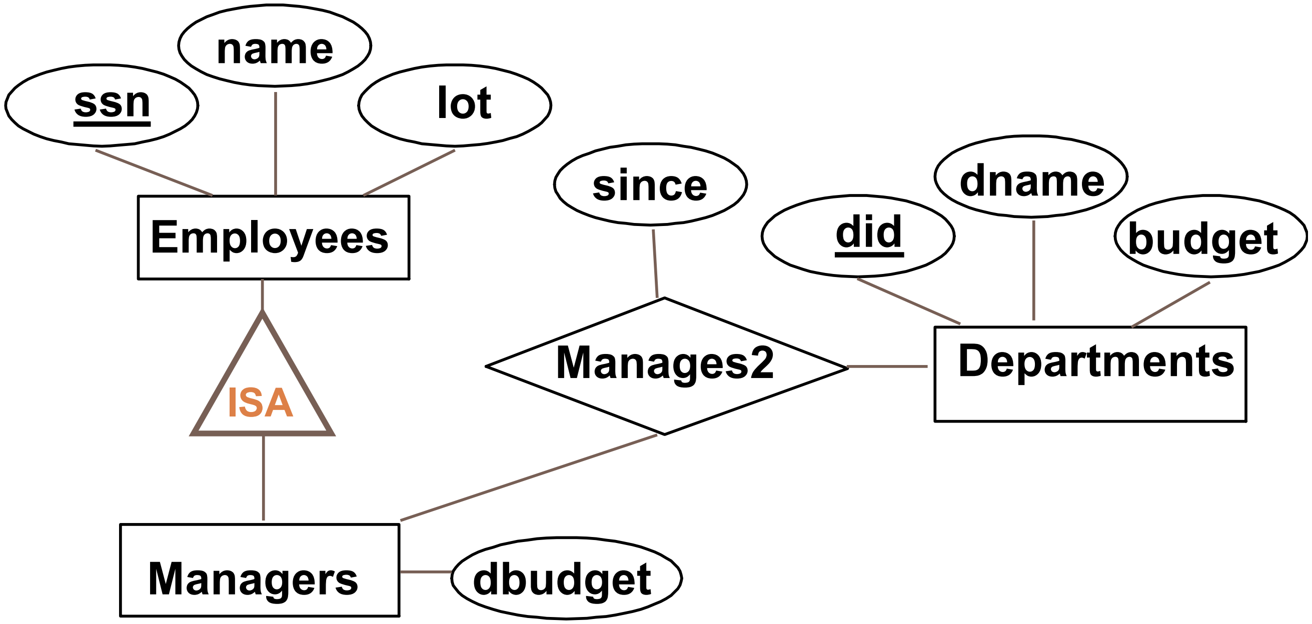
~~At(loginName, hostName)~~

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.