DM505 Database Design and Programming
DM576 Database Systems

Panagiotis Tampakis

ptampakis@imada.sdu.dk

Entity-Relationship Model

Purpose of E/R Model

- The E/R model allows us to sketch database schema designs
  - Includes some constraints, but not operations
- Designs are pictures called *entity-relationship diagrams*
- Later: convert E/R designs to relational DB designs

Framework for E/R

- Design is a serious business
- The “boss” knows 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

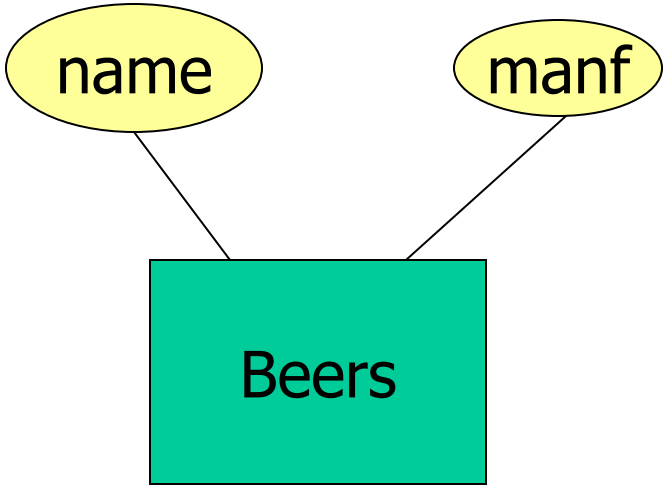
Entity Sets

- *Entity* = "thing" or object
- *Entity set* = collection of similar entities
  - Similar to a class in object-oriented languages
- *Attribute* = property of (the entities of) an entity set
  - Attributes are simple values, e.g. integers or character strings, not structs, sets, etc.

E/R Diagrams

- In an entity-relationship diagram:
  - Entity set = rectangle
  - Attribute = oval, with a line to the rectangle representing its entity set

Example:

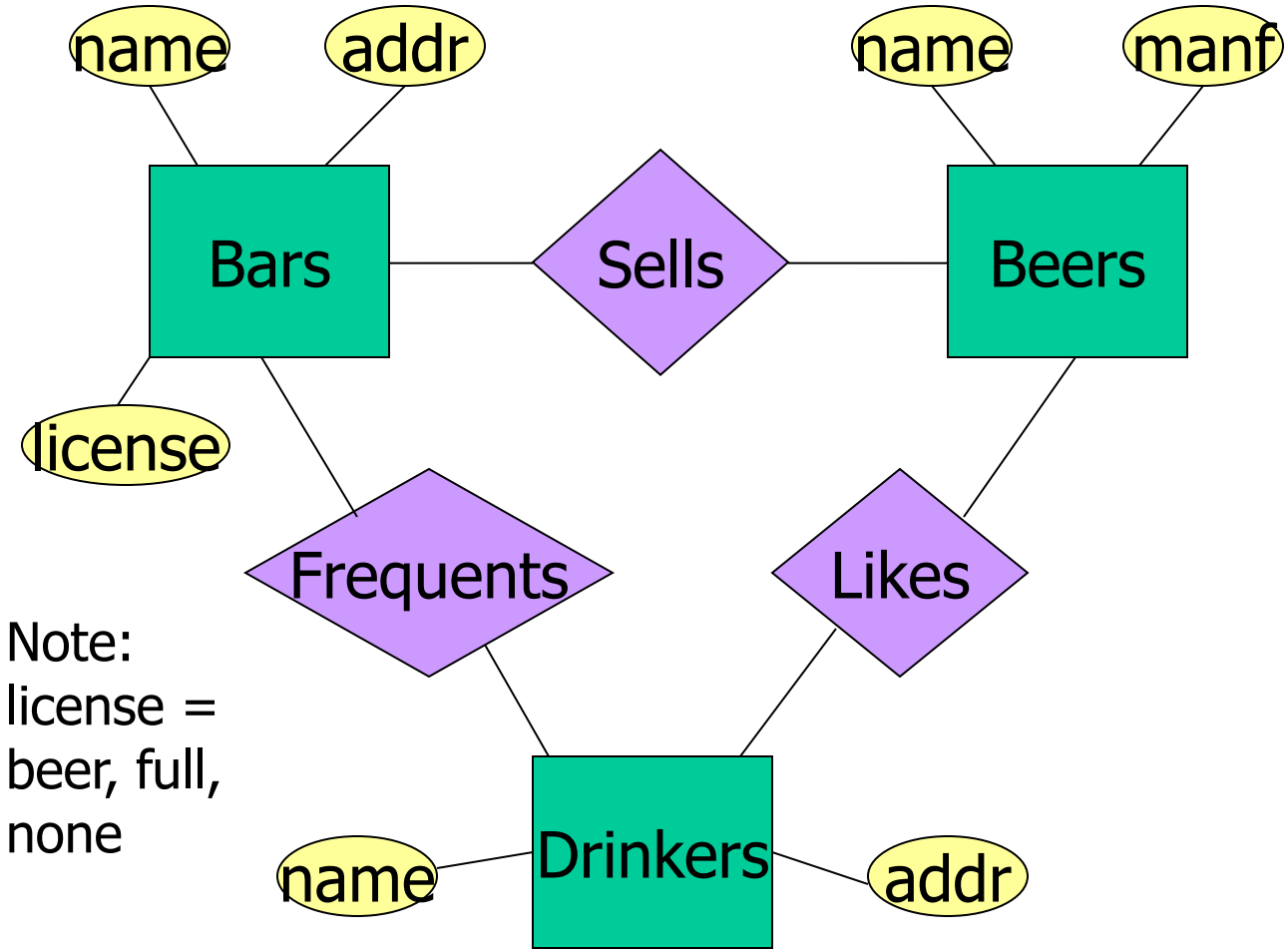


- Entity set Beers has two attributes, name and manf (manufacturer)
- Each Beers entity has values for these two attributes, e.g. (Odense Classic, Albani)

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
C.Ch.	Od.Cl.
C.Ch.	Erd.Wei.
C.Bio.	Od.Cl.
Brygg.	Pilsener
C4	Erd.Wei.

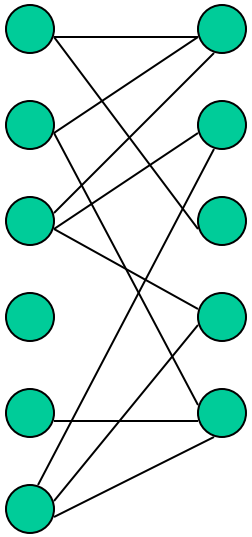
A Typical Relationship Set

Bar	Drinker	Beer
C.Ch.	Peter	Erd.Wei.
C.Ch.	Lars	Od.Cl.
C.Bio.	Peter	Od.Cl.
Brygg.	Peter	Pilsener
C4	Peter	Erd.Wei.
C.Bio.	Lars	Tuborg
Brygg.	Lars	Ale

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:

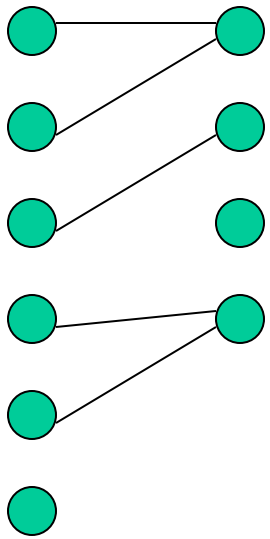


many-many

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

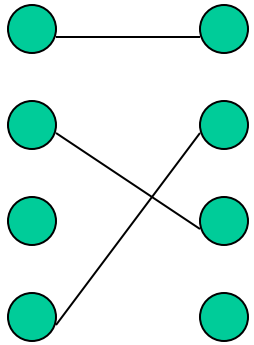
Example: Many-One Relationship

- 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

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:

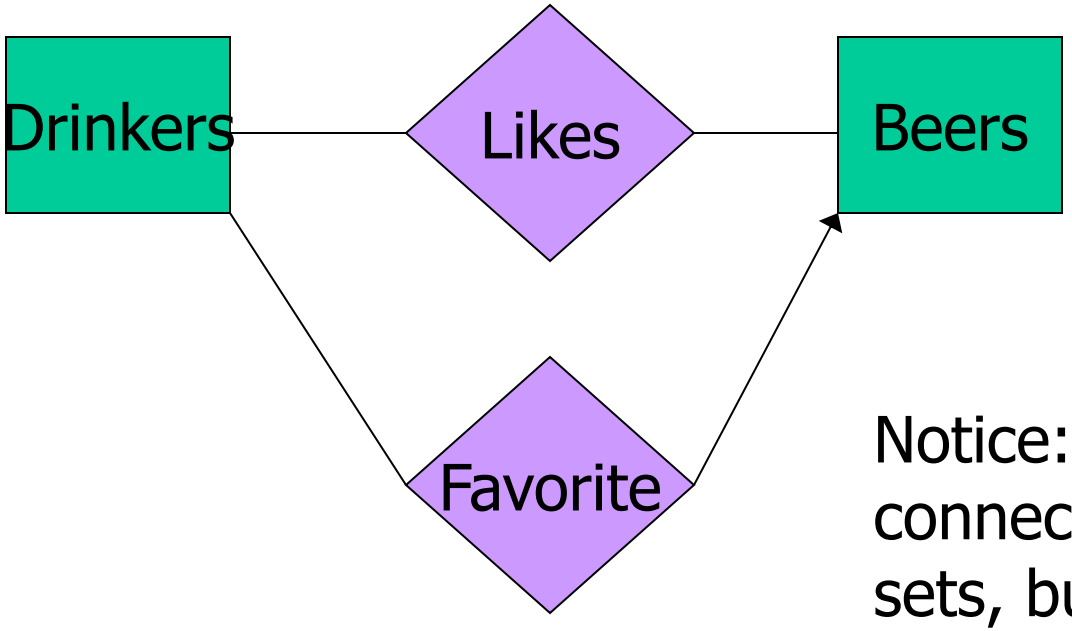


one-one

Representing "Multiplicity"

- Show a many-one relationship by an arrow entering the "one" side
  - Remember: Like a functional dependency
- Show a one-one relationship by arrows entering both entity sets
- Rounded 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: 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

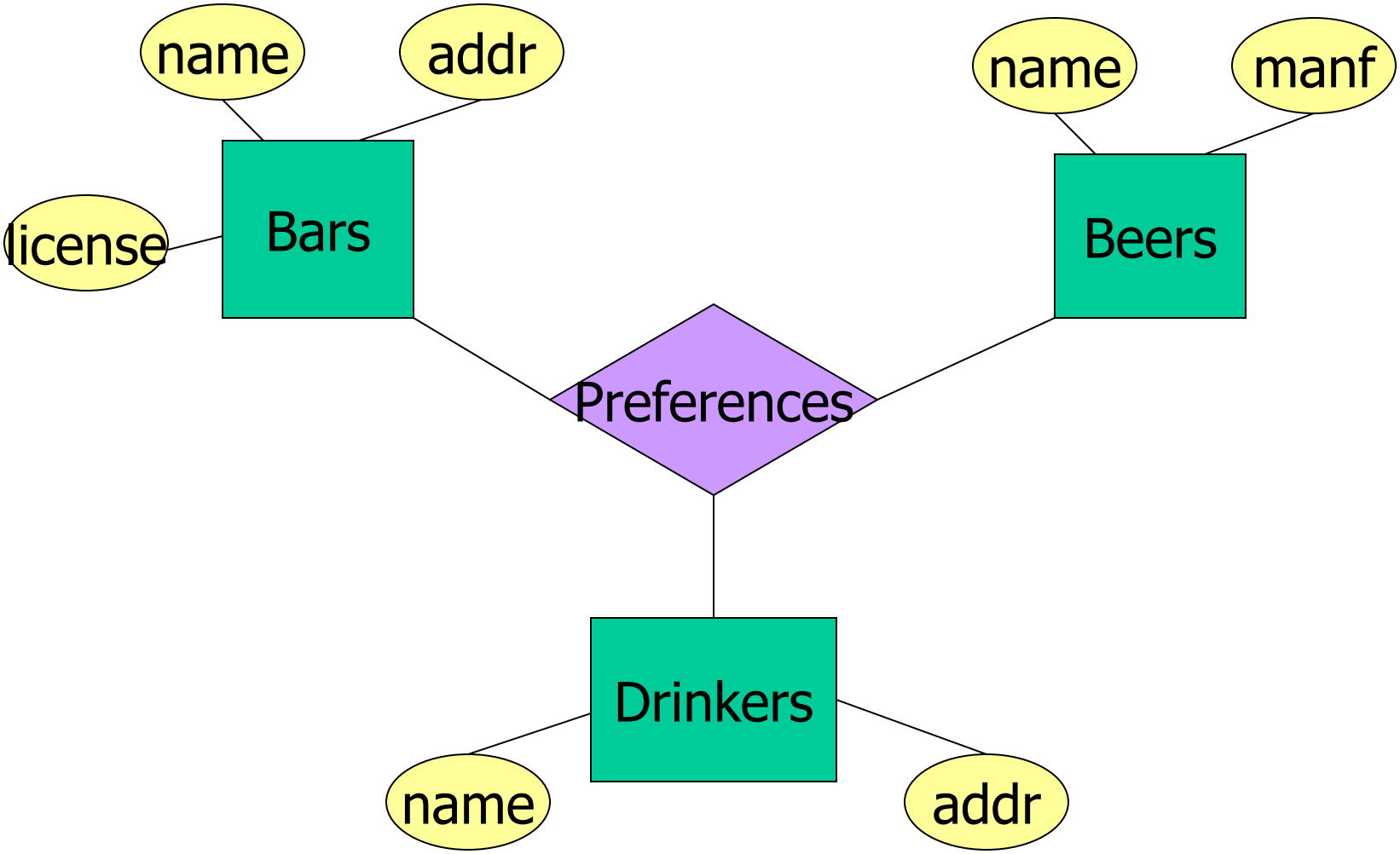
A beer is the best-seller for 0 or 1 manufacturer(s)

A manufacturer has exactly one best seller

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

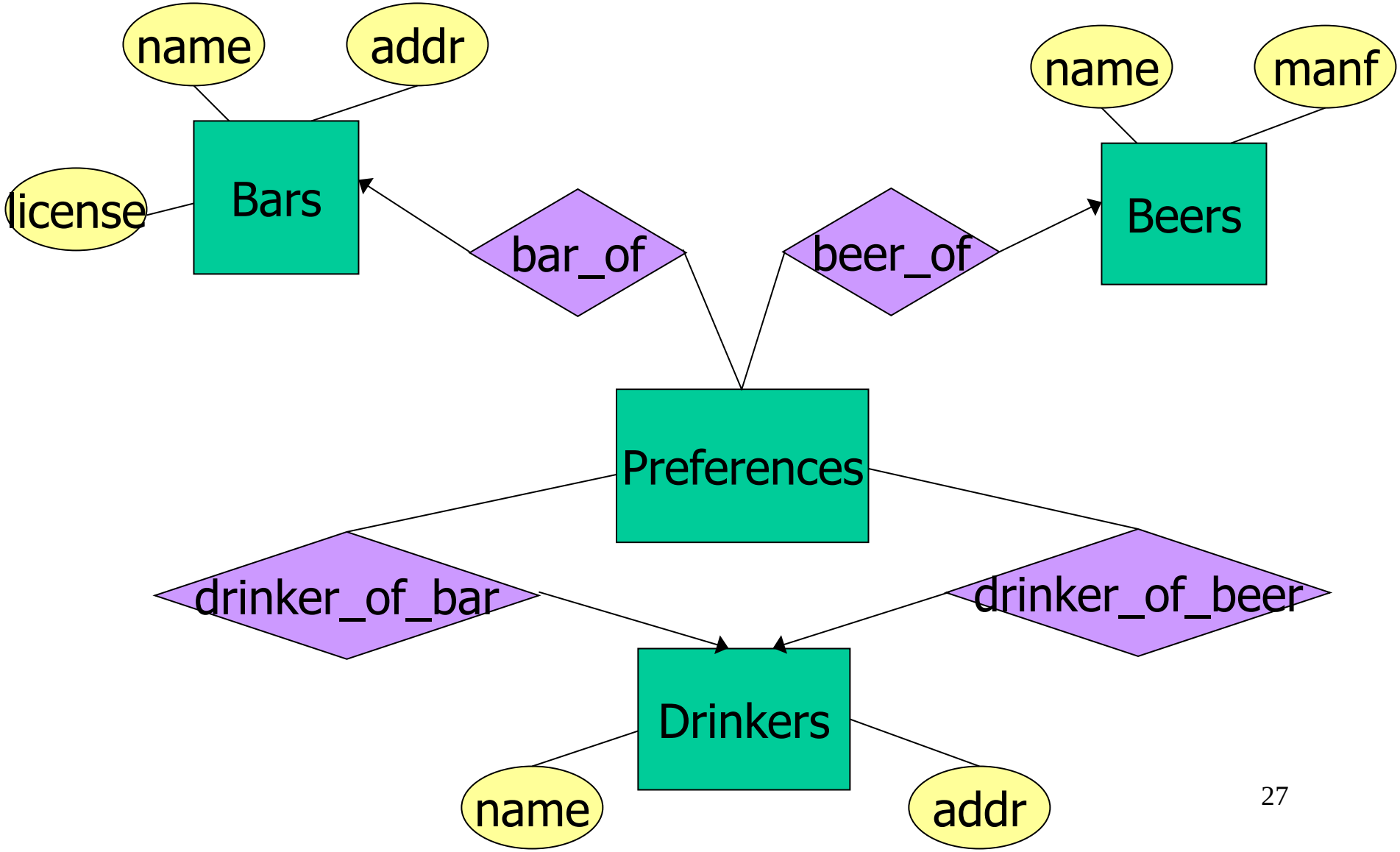
Example: 3-Way Relationship



Multiway Relationships

- Converting Multiway to Binary
  - Any multiway relationship can be converted to a collection of binary, many-one relationships.
  - By introducing a new entity set whose entities we may think of as tuples of the relationship set for the multiway relationship.

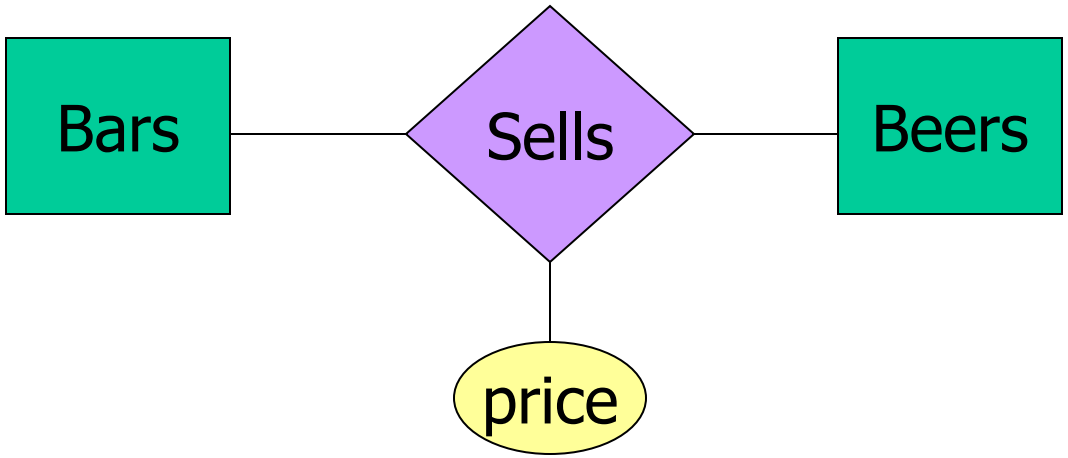
Example: 3-Way Relationship



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

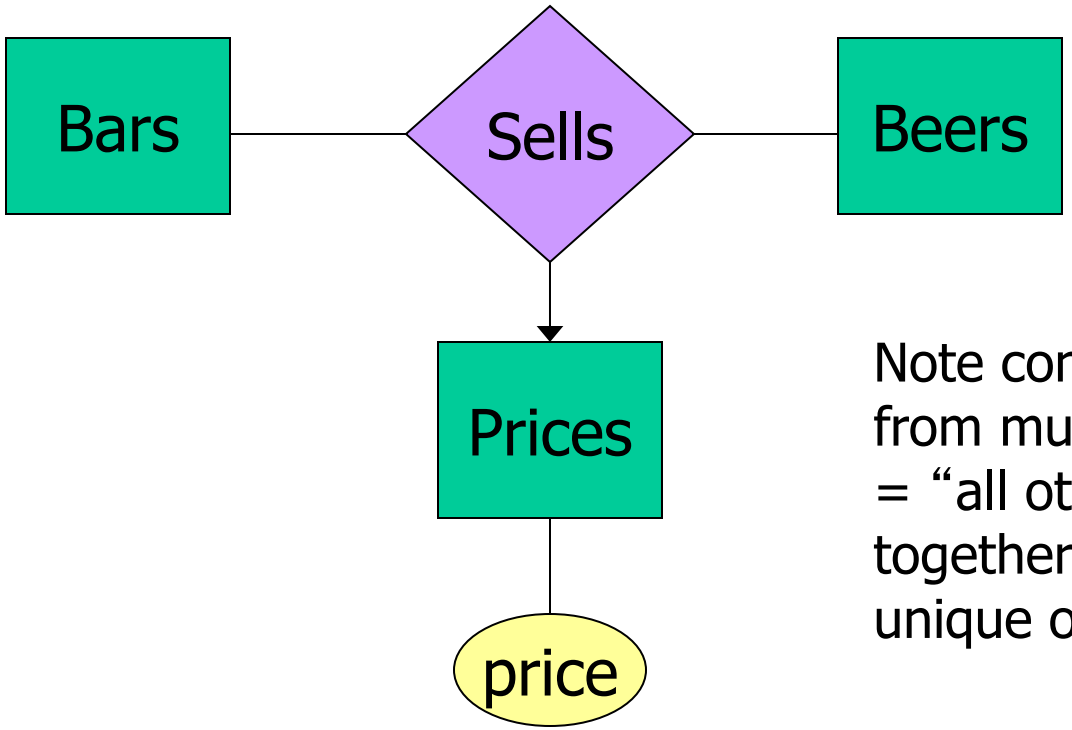


Price is a function of both the bar and the beer, not of one alone

Equivalent Diagrams Without 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

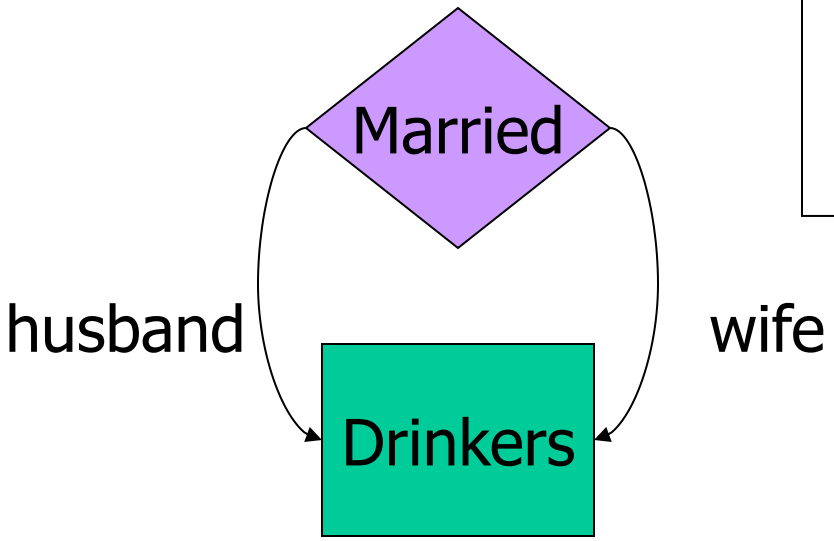


Note convention: arrow from multiway relationship = “all other entity sets together determine a unique one of these”

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*

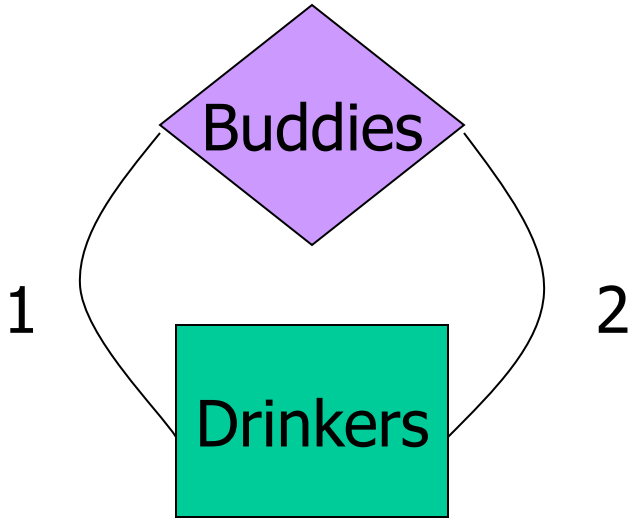
Example: Roles



Relationship Set

Husband	Wife
Lars	Lene
Kim	Joan
…	…

Example: Roles



Relationship Set

Buddy1	Buddy2
Peter	Lars
Peter	Pepe
Pepe	Bea
Bea	Rafa
…	…

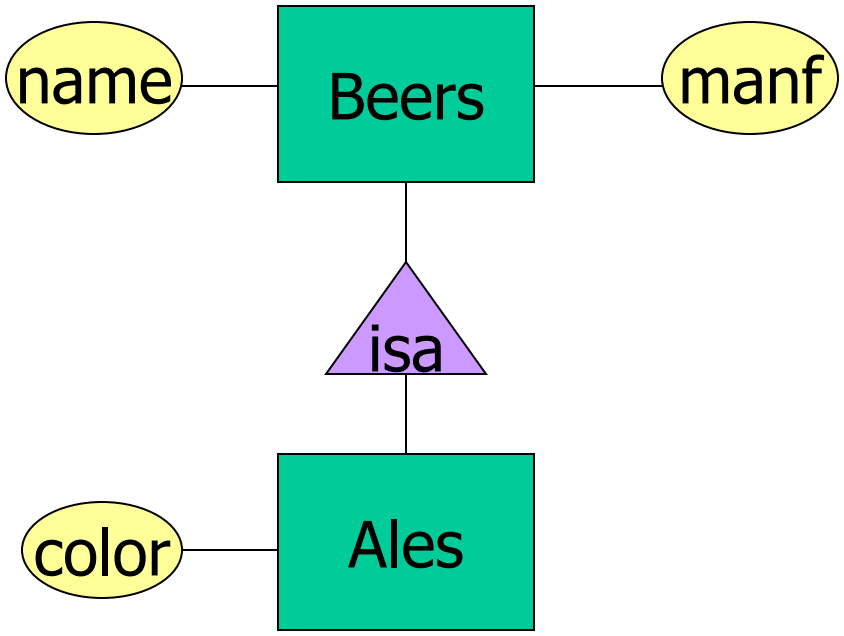
Subclasses

- *Subclass* = special case = fewer entities = 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

- Assume subclasses form a tree
  - I.e., no multiple inheritance
- Isa triangles indicate the subclass relationship
  - Point to the superclass

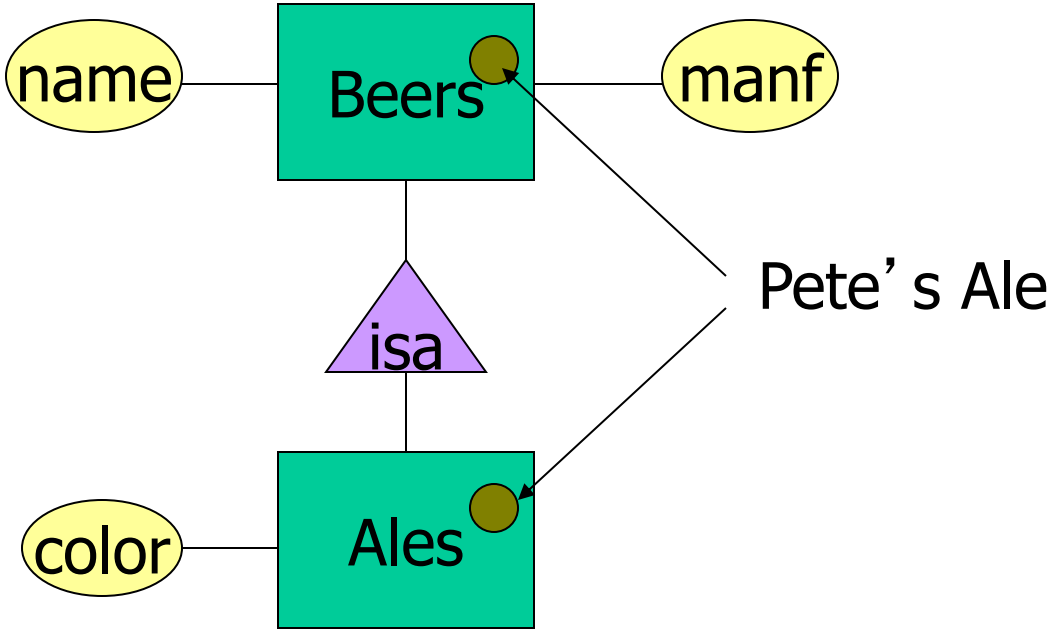
Example: Subclasses



E/R Vs. Object-Oriented Subclasses

- In OO, objects are in one class only
  - Subclasses inherit from superclasses.
- In contrast, E/R entities have *representatives* in all subclasses to which they belong
  - Rule: if entity *e* is represented in a subclass, then *e* is represented in the superclass (and recursively up the tree)

Example: Representatives of Entities



Exercise

A library wants to make a database system. They want the following objects modeled in their system.

Publisher having Name, Address

Book having Title, Number of Pages , ISBN

Author having CPR-number, Name

Reader having CPR-number, Name

They also describe the relationships between the objects: Each book can be published by one publisher. Each book is written by exactly one author. Each author has written one or more books.

Each reader may have borrowed any number of books any number of times. Authors may also be readers.

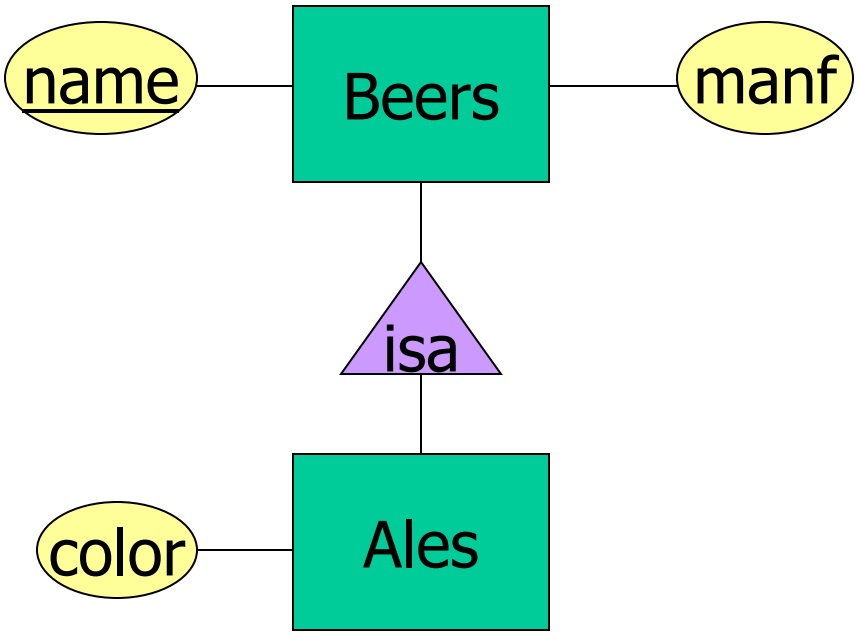
Keys

- 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

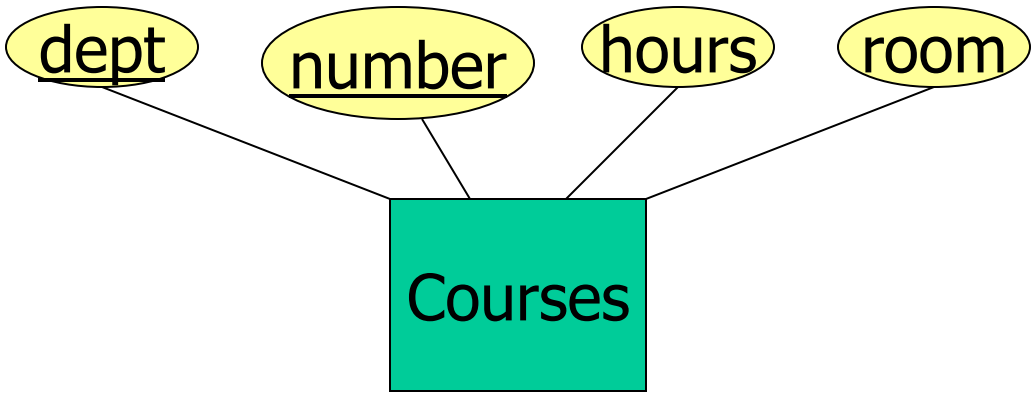
Keys in E/R Diagrams

- Underline the key attribute(s)
- 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

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

Referential integrity

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

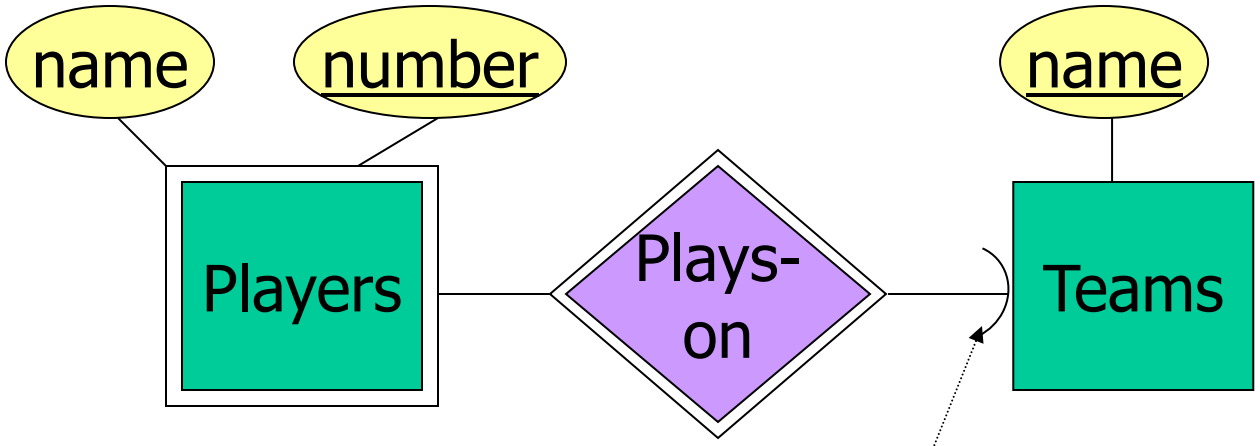
Weak Entity Sets

- Two reasons for weak entity sets:
  - Having a hierarchy based on classifications unrelated to the "isa hierarchy"
    - If entities of a set are subunits of entities in another set
  - When we eliminate a multiway relationships by introducing the connecting entity
    - These entity sets often have no attributes.
    - Their key is formed from the attributes that are the key attributes for the entity sets they connect.

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 needs support
  - But supporting relationships must have a rounded arrow (entity at the “one” end is guaranteed)
    - Referential integrity

Weak Entity-Set Rules – (2)

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

Exercise

A library wants to make a database system. They want the following objects modeled in their system.

Publisher having Name, Address

Book having Title, Number of Pages ~~, ISBN~~

Author having CPR-number, Name

Reader having CPR-number, Name

They also describe the relationships between the objects: Each book can be published by one publisher. Each book is written by exactly one author. Each author has written one or more books. Multiple books may have the same title, but one author cannot be the author of more than one book with the same title.

Each reader may have borrowed any number of books any number of times. Authors may also be readers.

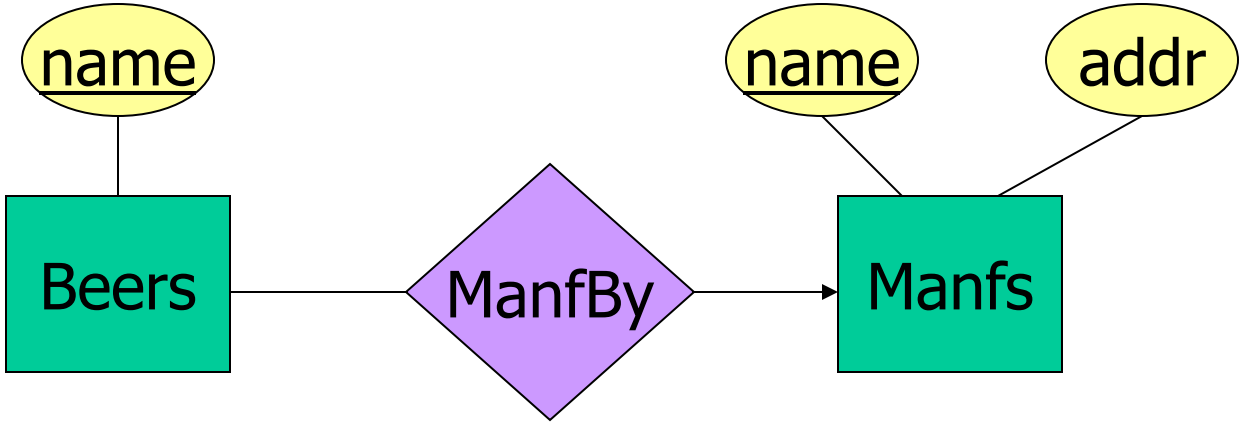
Design Techniques

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

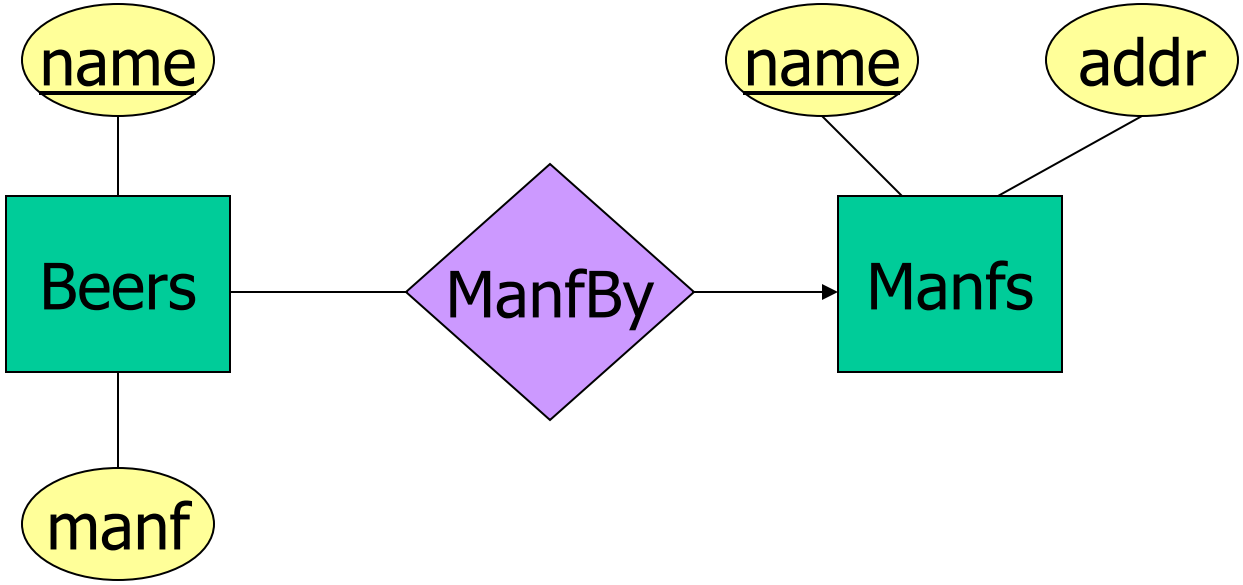
- *Redundancy* = saying the same thing in two (or more) different ways
- Wastes space and (more importantly) encourages inconsistency
  - Two representations of the same fact become inconsistent if we change one and forget to change the other
  - Recall anomalies due to FD's

Example: Good



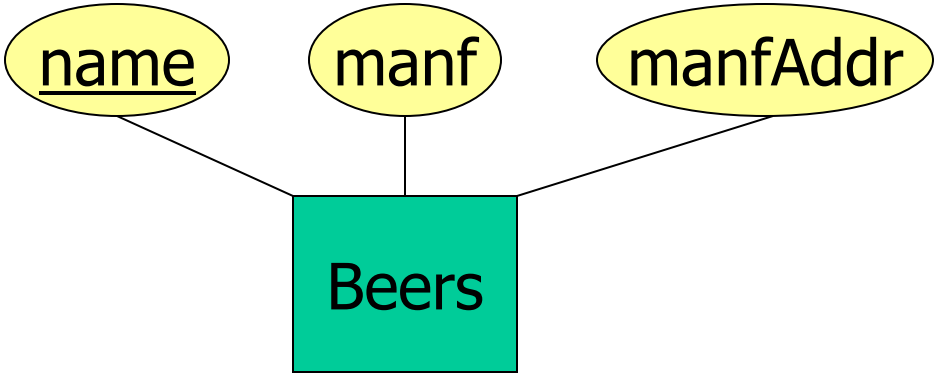
This design gives the address of each manufacturer exactly once.

Example: Bad



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

Example: Bad



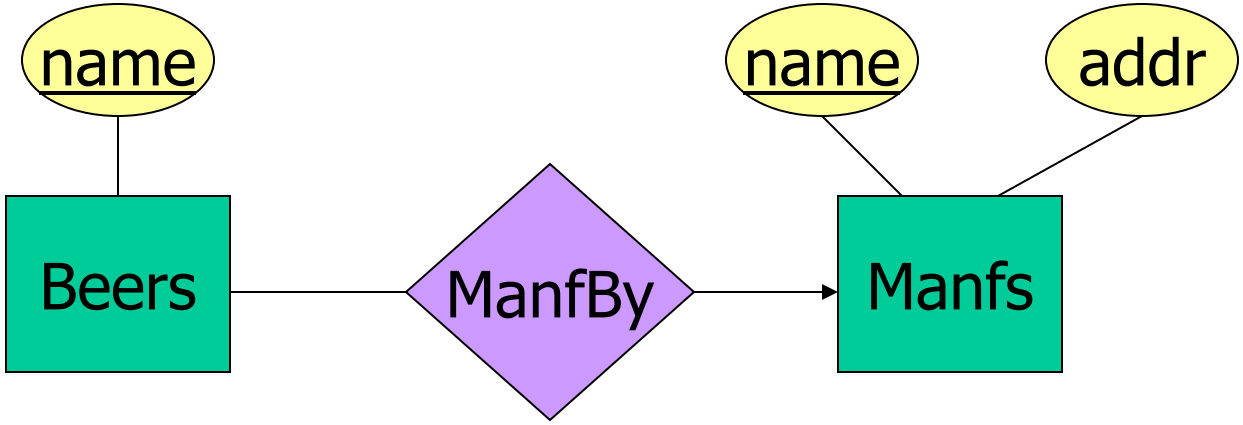
This design repeats the manufacturer's address once for each beer and loses the address if there are temporarily no beers for a manufacturer

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

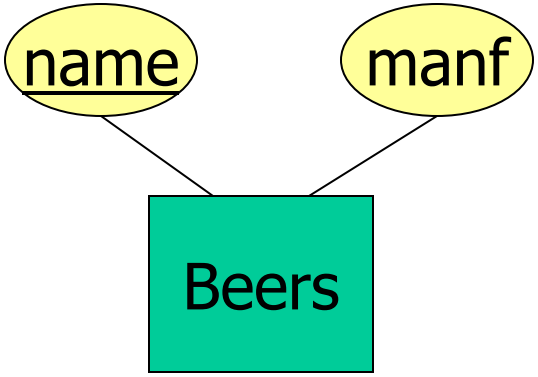
    or
  - It is the “many” in a many-one or many-many relationship

Example: Good



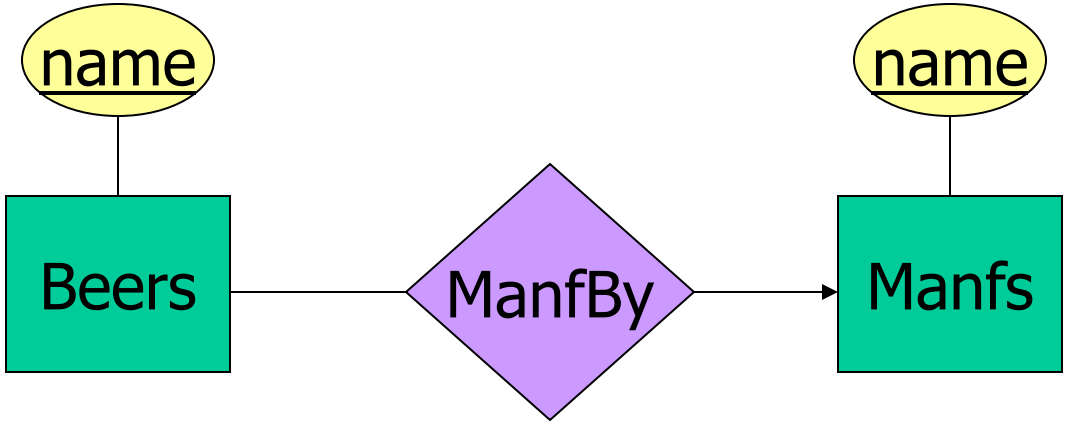
- Manfs deserves to be an entity set because of the nonkey attribute addr
- Beers deserves to be an entity set because it is the "many" of the many-one relationship ManfBy

Example: Good



There is no need to make the manufacturer an entity set, because 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

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 CPR numbers, car's license plates, etc.

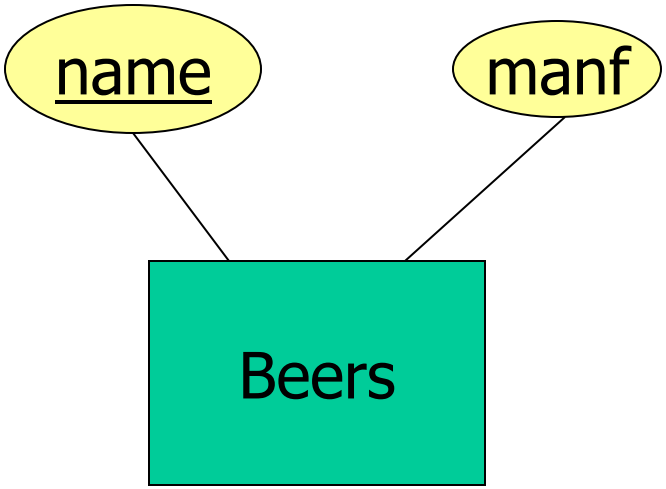
When Do We Need Weak Entity Sets?

- 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

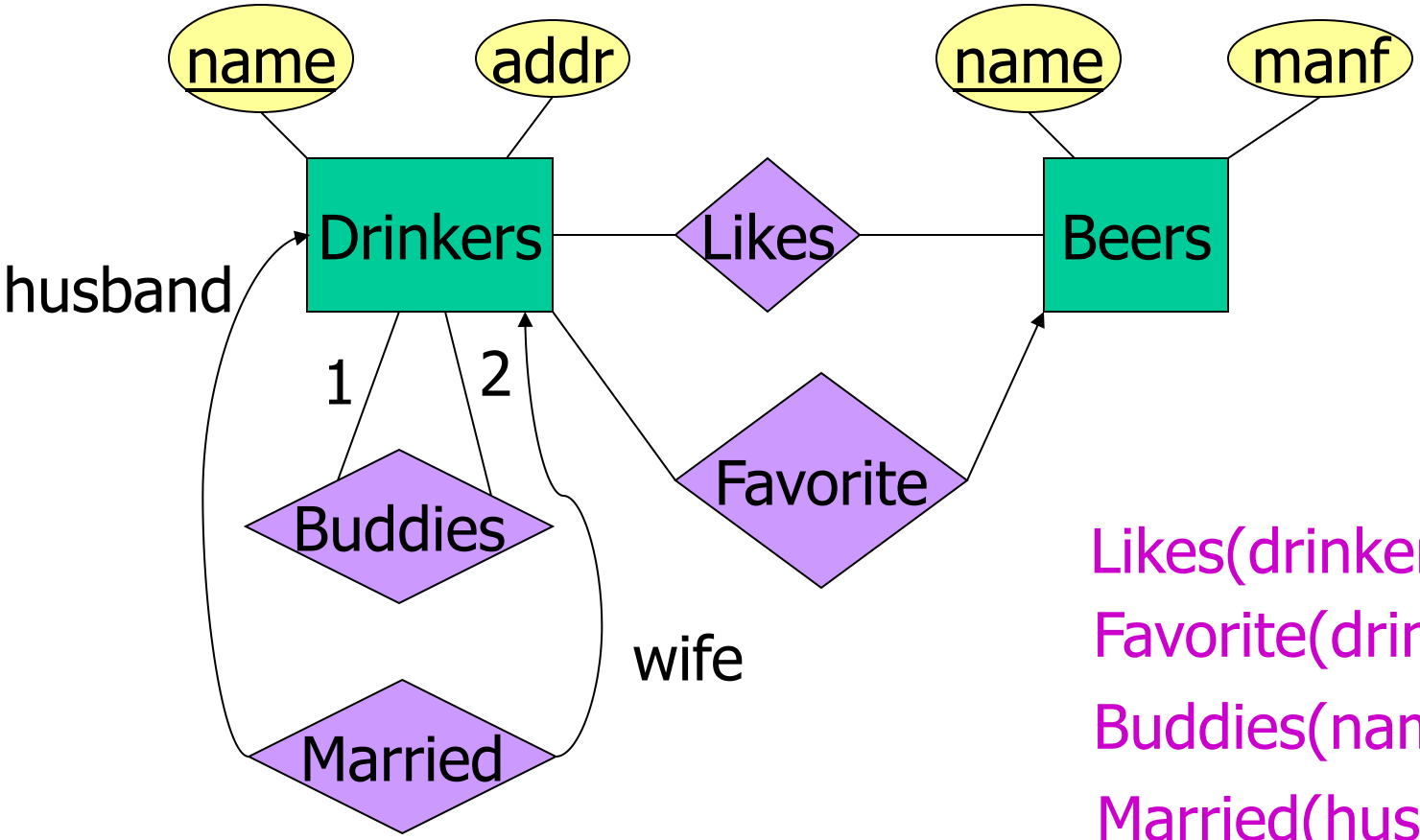
- Entity set –> relation
  - Attributes -> attributes
- Relationships –> relations whose attributes are only:
  - The keys of the connected entity sets
  - Attributes of the relationship itself

Entity Set –> Relation



Relation: Beers(name, manf)

Relationship -> Relation



Likes(drinker, beer)

Favorite(drinker, beer)

Buddies(name1, name2)

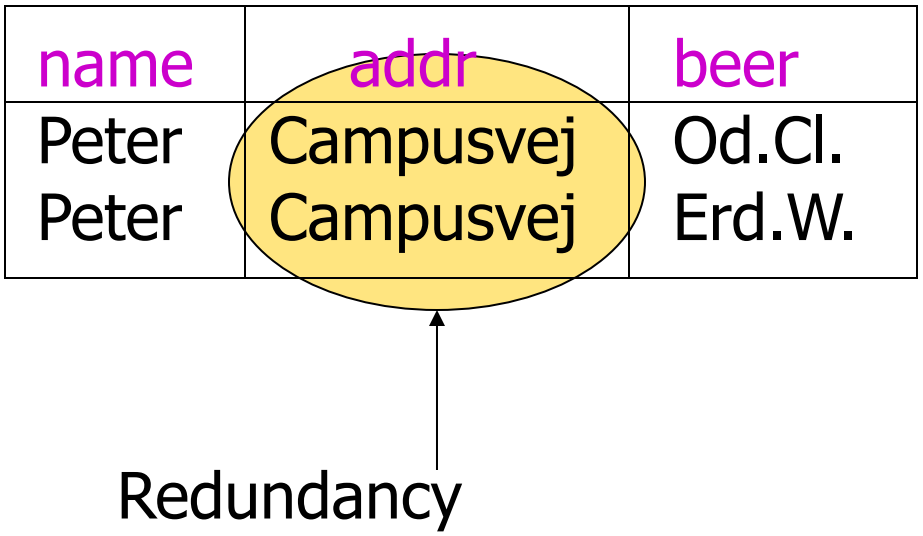
Married(husband, wife)

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)

Risk with Many-Many Relationships

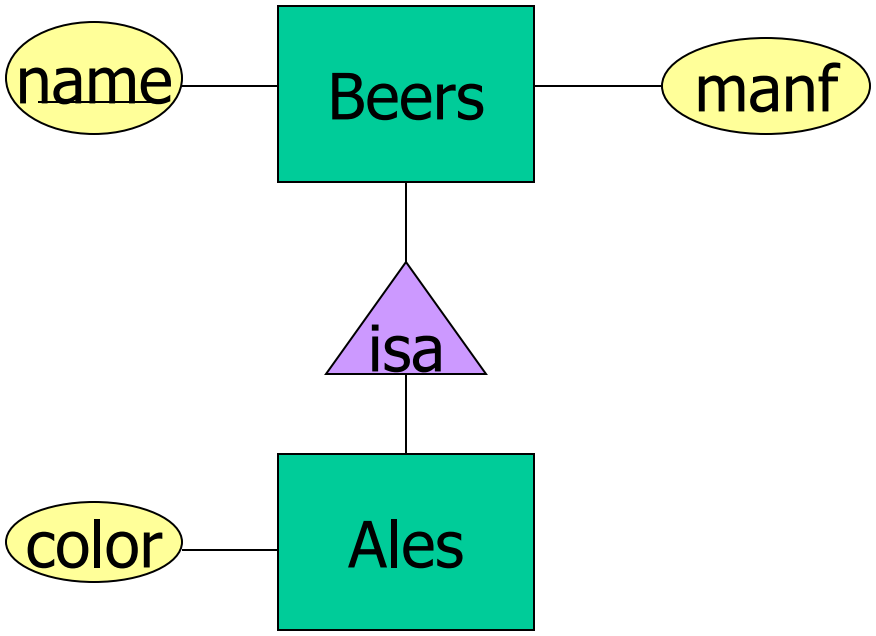
- Combining Drinkers with Likes would be a mistake. It leads to redundancy, as:



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 –> Relations



Object-Oriented

name	manf
Odense Classic	Albani

Beers

name	manf	color
HC Andersen	Albani	red

Ales

Good for queries like “find the color of ales made by Albani”

E/R Style

name	manf
Odense Classic	Albani
HC Andersen	Albani

Beers

name	color
HC Andersen	red

Ales

Good for queries like "find all beers (including ales) made by Albani"

Using Nulls

name	manf	color
Odense Classic	Albani	NULL
HC Andersen	Albani	red
	Beers	

Saves space unless there are *lots* of attributes that are usually NULL

Exercise

A library wants to make a database system. They want the following objects modeled in their system.

Publisher having Name, Address

Book having Title, Number of Pages ~~, ISBN~~

Author having CPR-number, Name

Reader having CPR-number, Name

They also describe the relationships between the objects: Each book can be published by one publisher. Each book is written by exactly one author. Each author has written one or more books. Multiple books may have the same title, but one author cannot be the author of more than one book with the same title.

Each reader may have borrowed any number of books any number of times. Authors may also be readers.

Summary 6

More things you should know:

- Entities, Attributes, Entity Sets,
- Relationships, Multiplicity, Keys
- Roles, Subclasses, Weak Entity Sets
- Design guidelines
- E/R diagrams –> relational model