Lecture 2: The Entity-Relationship Model

EECS 339

Winter 2016

Entity-Relationship Model

E/R Diagrams

Weak Entity Sets

Converting E/R Diagrams to Relations

Purpose of E/R Model

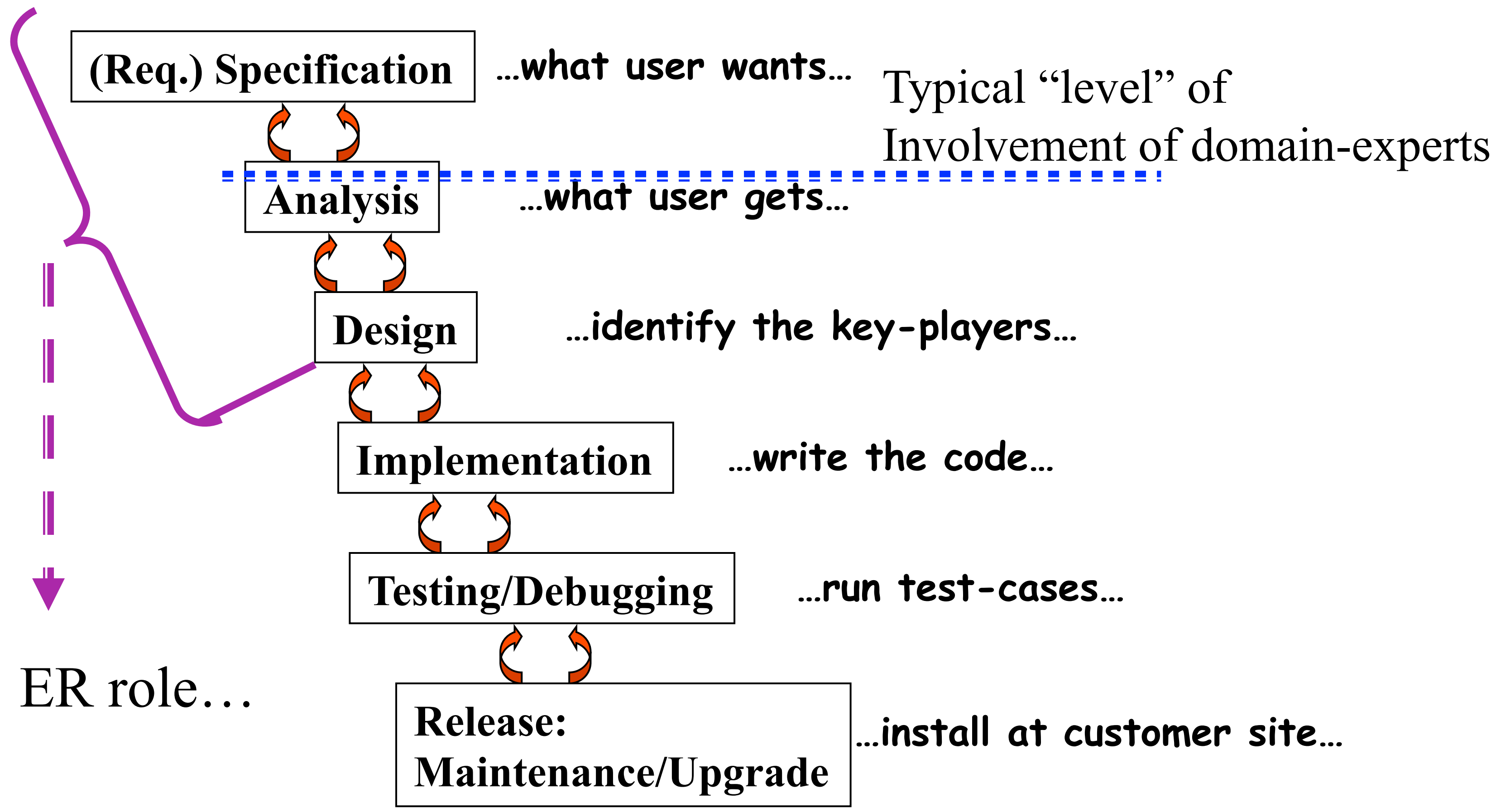
- The E/R model allows us to sketch database schema designs.
  - Includes some constraints, but not operations.
- Designs are pictures called *entity-relationship diagrams*.
- Later: convert E/R designs to relational DB designs.

Framework for E/R

- Design is a serious business.
- Management knows they want a database, but they don’t know what they want in it.
  - Nor, for that matter, how to build it…
- Sketching the key components is an efficient way to develop a working database.

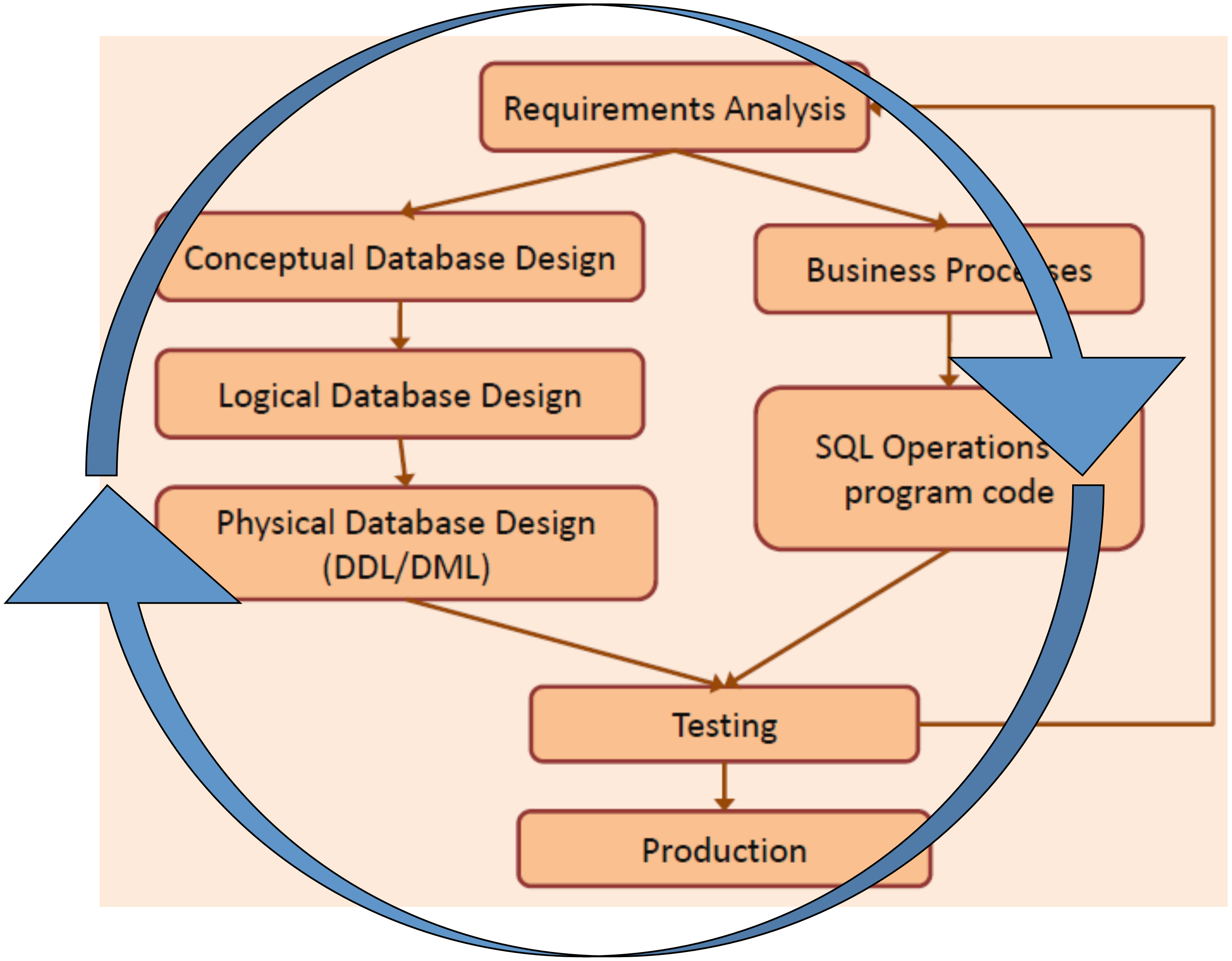
Software Lifecycle

"Waterfall Model" (old, but…):



ER role…

Software lifecycle – database-ish perspective


Requirements Analysis
Conceptual Database Design
Business Processes
Logical Database Design
SQL Operations program code
Physical Database Design (DDL/DML)
Testing
Production

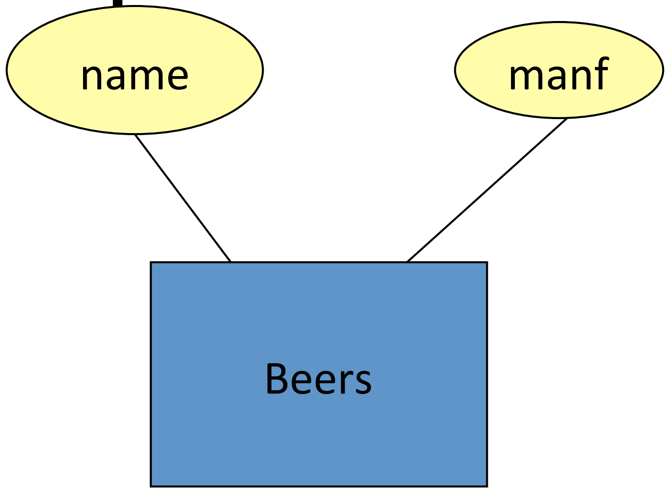
Entity Sets

- *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.

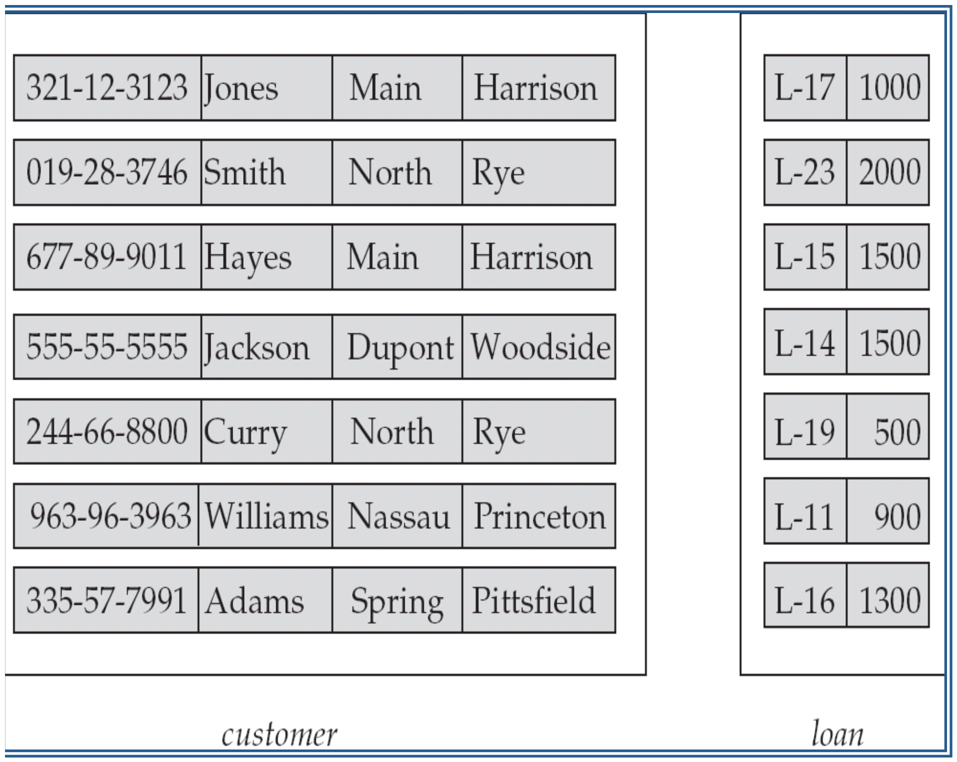
Example:



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

Another Example

Example of (a collection of) entity-instances



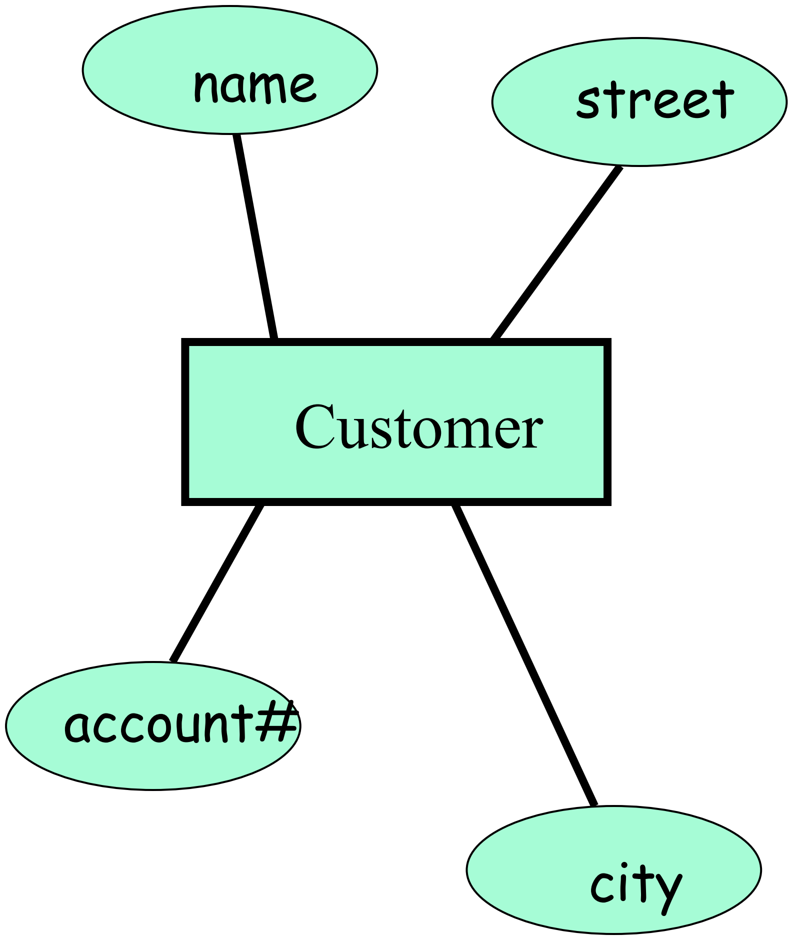
321-12-3123	Jones	Main	Harrison
019-28-3746	Smith	North	Rye
677-89-9011	Hayes	Main	Harrison
555-55-5555	Jackson	Dupont	Woodside
244-66-8800	Curry	North	Rye
963-96-3963	Williams	Nassau	Princeton
335-57-7991	Adams	Spring	Pittsfield

customer

L-17	1000
L-23	2000
L-15	1500
L-14	1500
L-19	500
L-11	900
L-16	1300

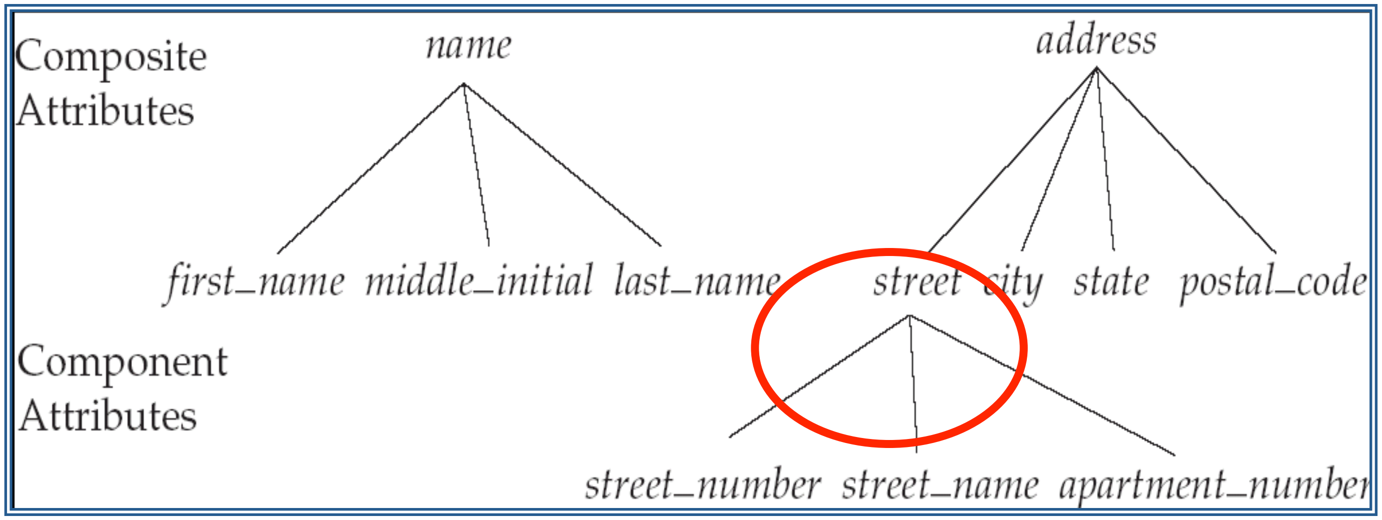
loan

Example of the ER-diagram



ER Modeling: Attributes

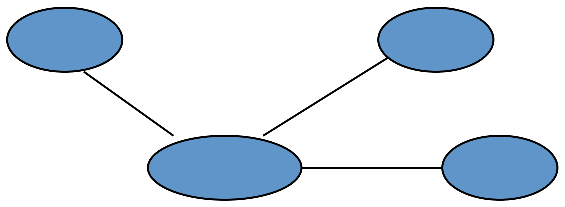
- ***Simple vs. Composite***
  - ***Composite attributes can always be split into a collection of attributes that are "atomic" (i.e., their domain is "simpler")***



Composite Attributes
name
address
first_name middle_initial last_name
street city state postal_code
Component Attributes
street_number street_name apartment_number

It is possible to have a hierarchy of nesting/compositions…

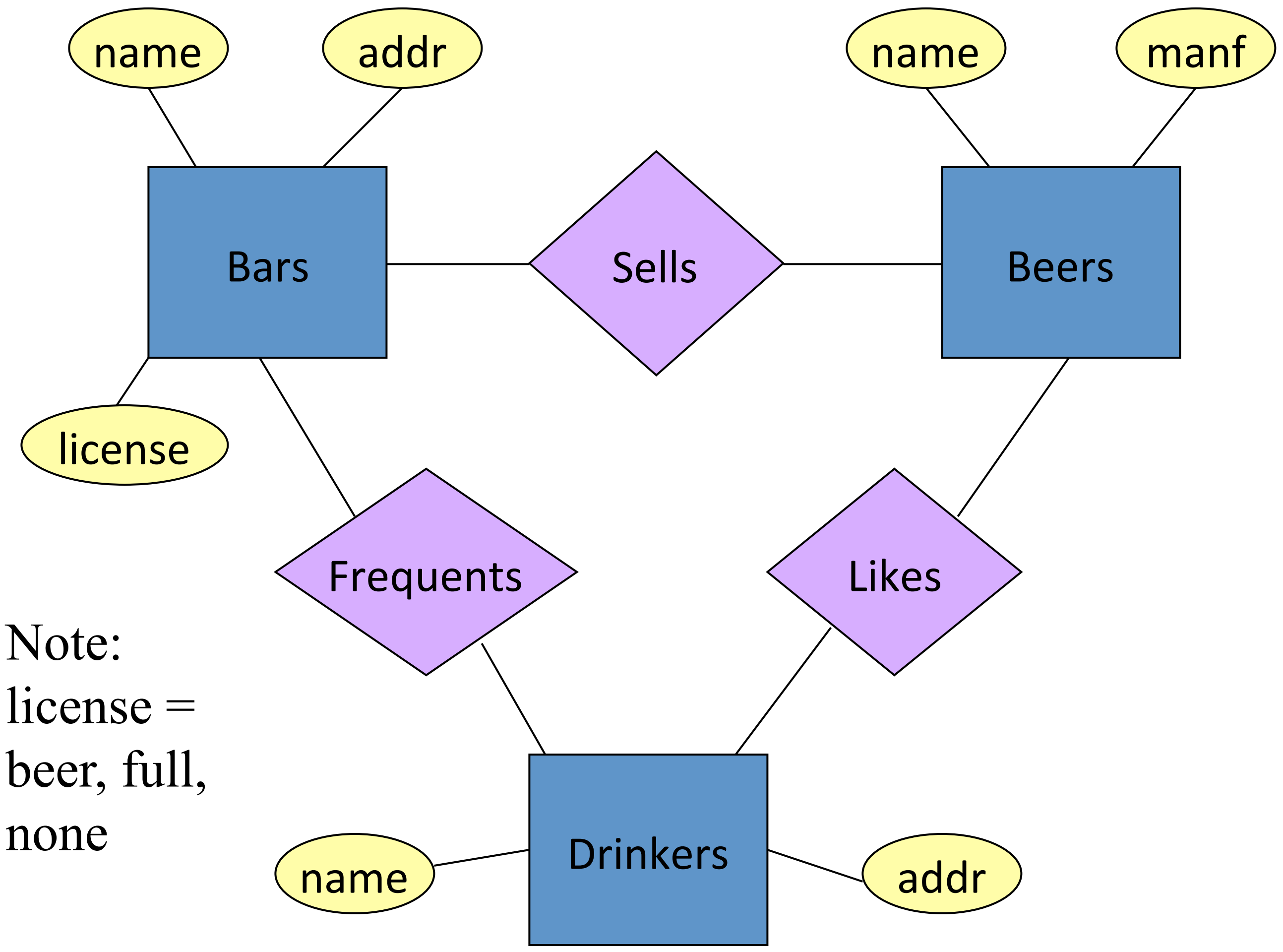
Symbol:



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

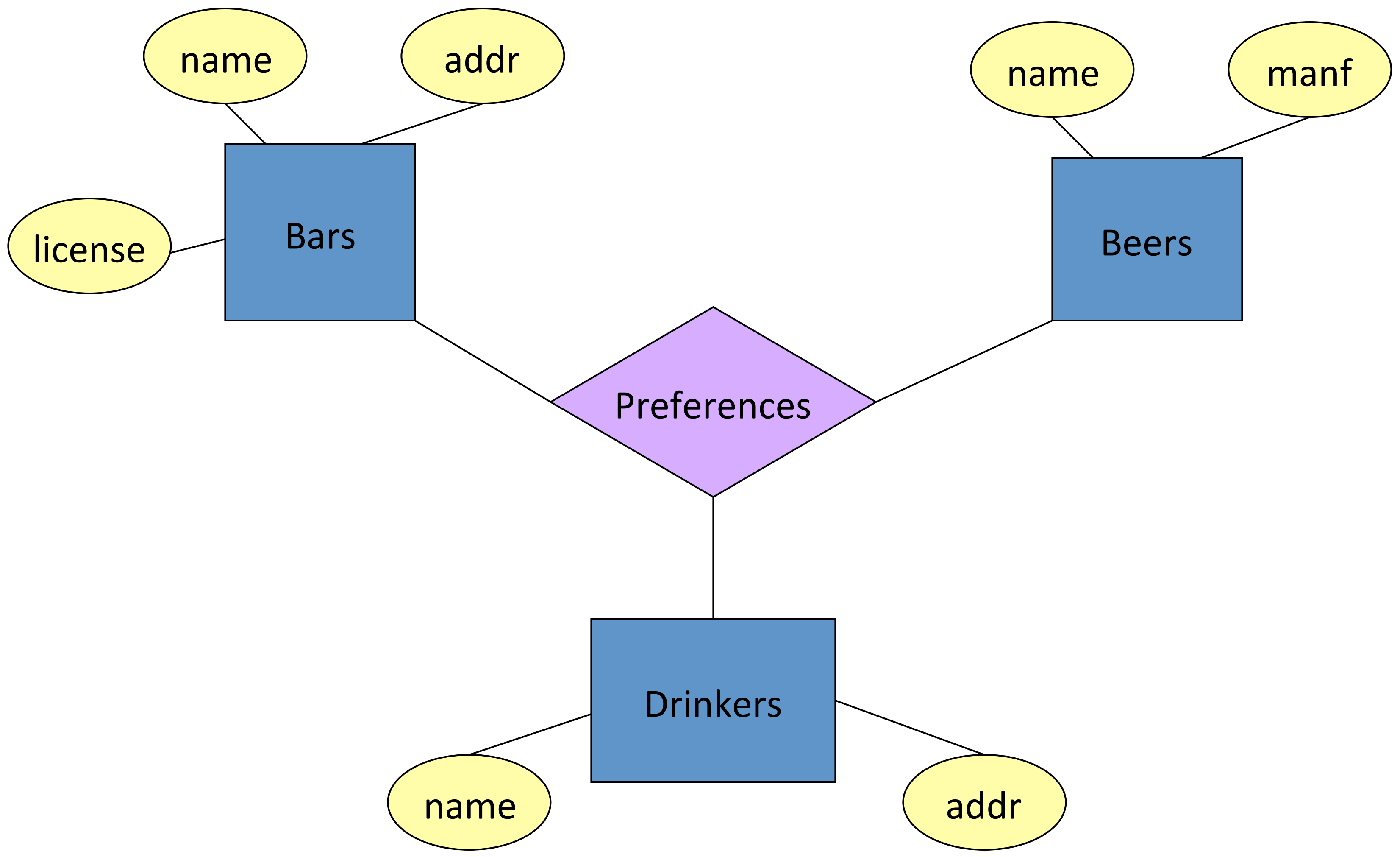
- For the relationship Sells, we might have a relationship set like:

Bar	Beer
Joe’s Bar	Sam Adams
Joe’s Bar	PBR
Sue’s Bar	Pliney
Sue’s Bar	Pete’s Ale
Sue’s Bar	Rogue

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.

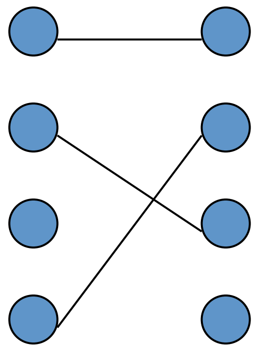
Example: 3-Way Relationship



An Example Relationship Set

Bar	Drinker	Beer
Joe’s Bar	Ann	Sam Adams
Sue’s Bar	Ann	Blue Moon
Sue’s Bar	Ann	Pete’s Ale
Joe’s Bar	Bob	Rogue
Joe’s Bar	Bob	PBR
Joe’s Bar	Cal	Blue Moon
Sue’s Bar	Cal	Blue Moon

One-One Relationships

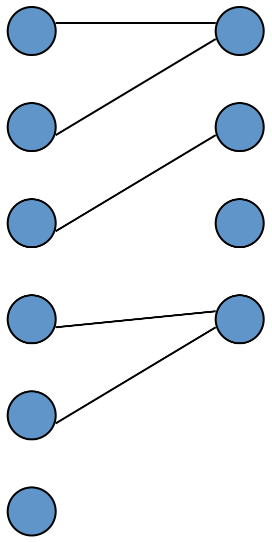


- In a *one-one* relationship, each entity of either entity set is related to at most one entity of the other set.
- Example: Relationship Best-seller between entity sets Manfs (manufacturer) and Beers.
  - A beer cannot be made by more than one manufacturer, and no manufacturer can have more than one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

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.

Many-One Relationships

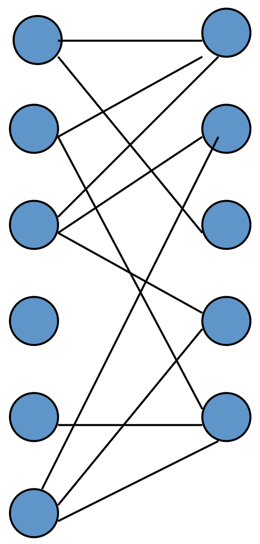


- Some binary relationships are *many-one* from one entity set to another.
- Each entity of the first set is connected to at most one entity of the second set.
- But an entity of the second set can be connected to zero, one, or many entities of the first set.

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-Many Relationships

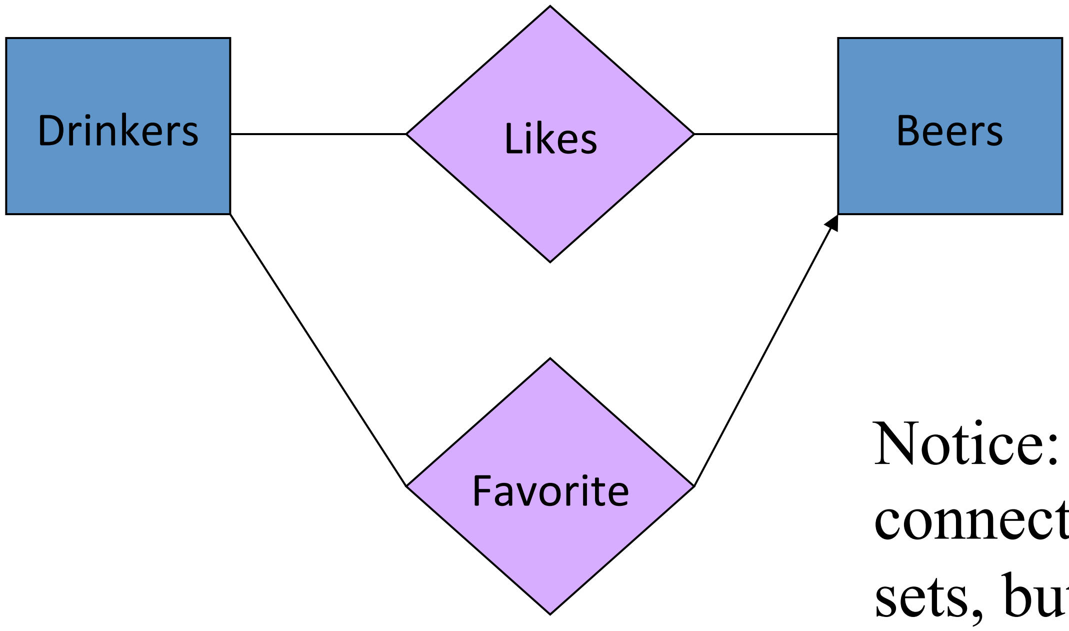


- Focus: binary relationships, such as Sells between Bars and Beers.
- In a *many-many* relationship, an entity of either set can be connected to many entities of the other set.
  - E.g., a bar sells many beers; a beer is sold by many bars.

Representing “Multiplicity”

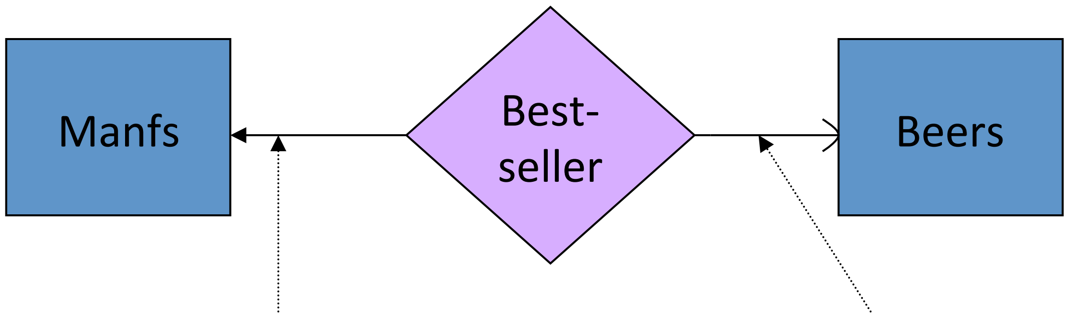
- Show a many-one relationship by an arrow entering the “one” side.
  - Remember: Like a functional dependency.
- Show a one-one relationship by arrows entering both entity sets.
- Rounded arrow = “exactly one,” i.e., each entity of the first set is related to exactly one entity of the target set.

Example: Many-One Relationship



Notice: two relationships connect the same entity sets, but are different (have different semantics).

In the E/R Diagram



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

A manufacturer has exactly one best seller.

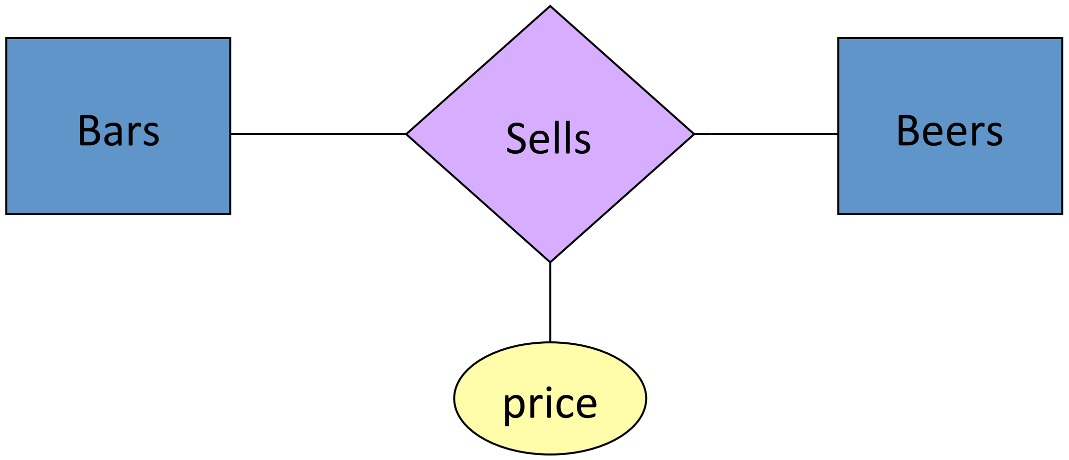
Study Break: E/R Diagrams

- Design an E/R diagram for a zoo that has:
  - Animals with a name, type, cage, dob
  - Types of animals and their expected height, weight
  - Cages that contain one or more animals and have a location
  - Zookeepers that have a name, work shift, and cages they clean. More than one zookeeper may be in charge of a cage.

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

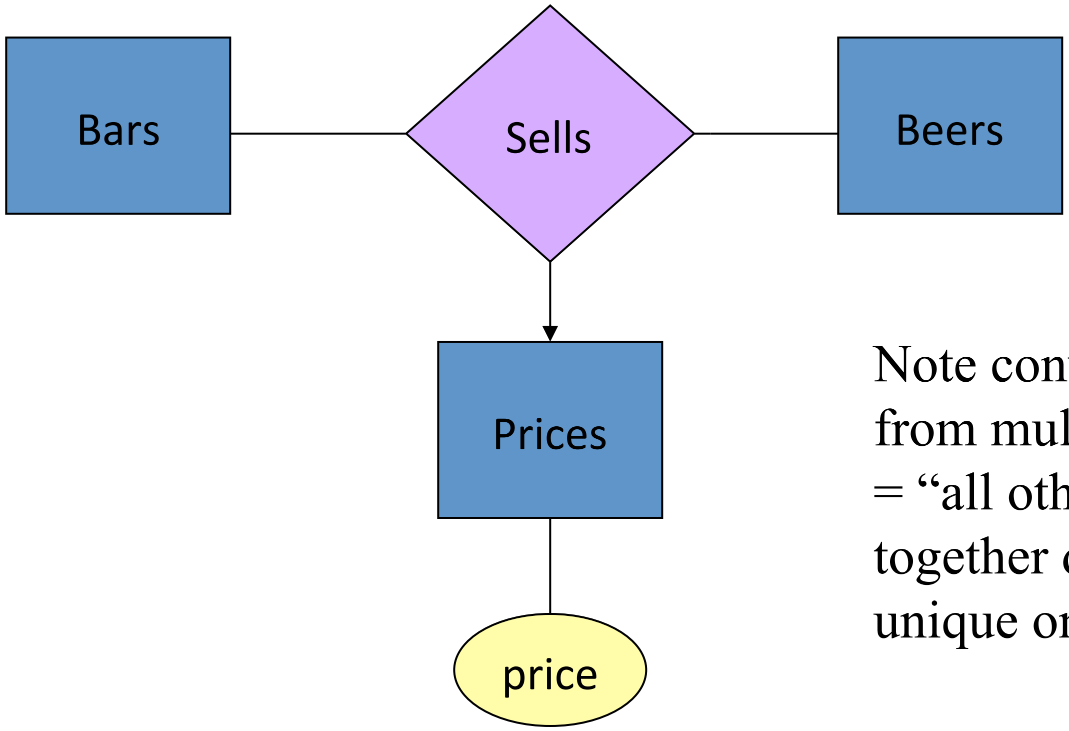


Price is a function of both the bar and the beer,
not of one alone.

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

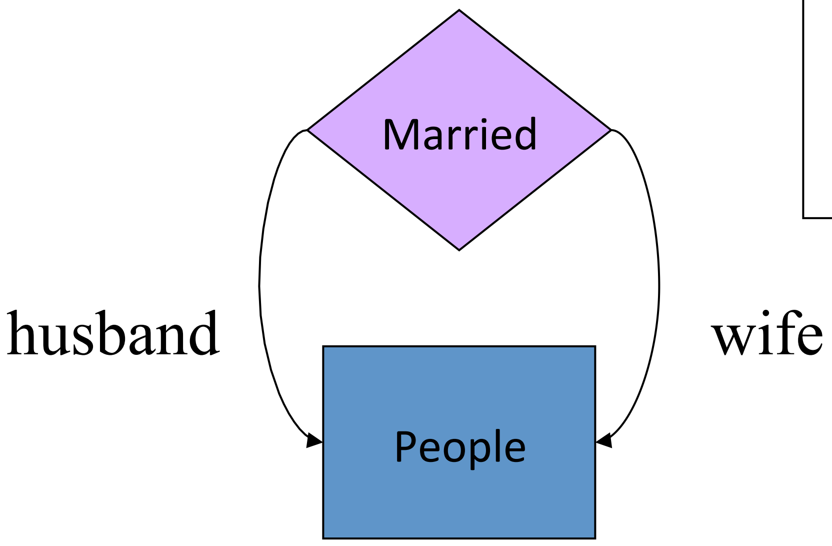


Note convention: arrow from multiway relationship = "all other entity sets together determine a unique one of these."

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*.

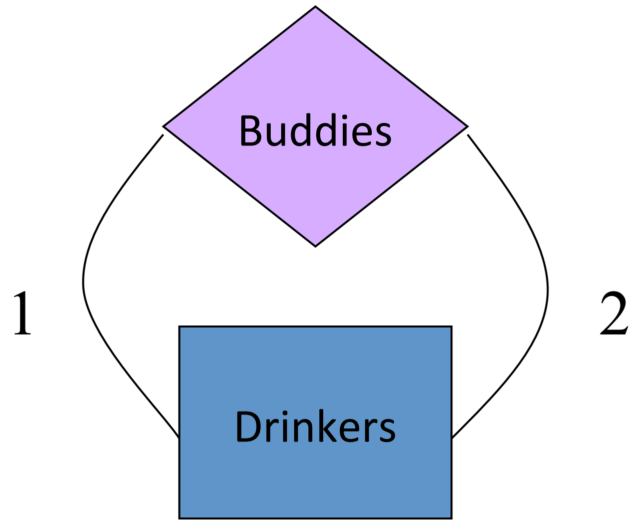
Example: Roles



Relationship Set

Husband	Wife
Bob	Ann
Joe	Sue
… …	

Example: Roles



Relationship Set

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

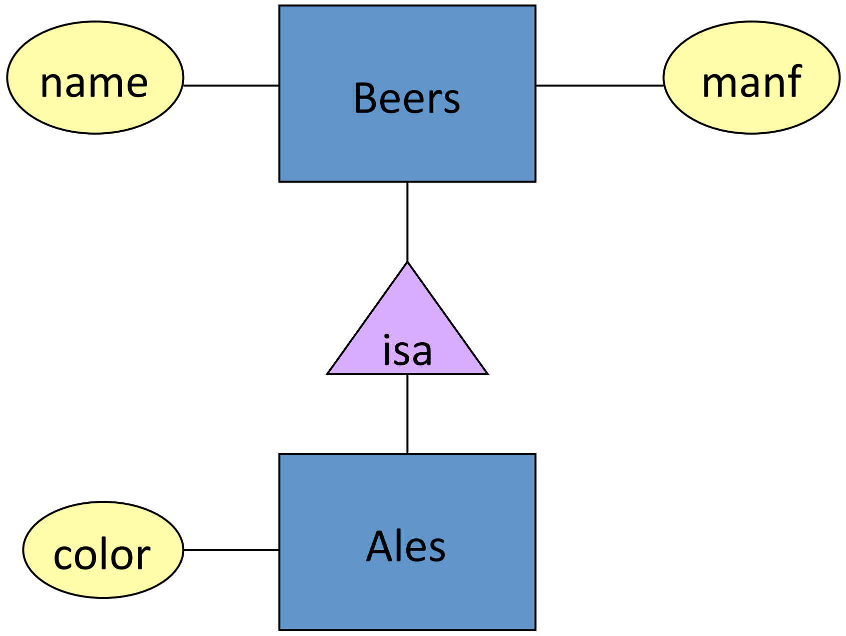
Subclasses

- *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.

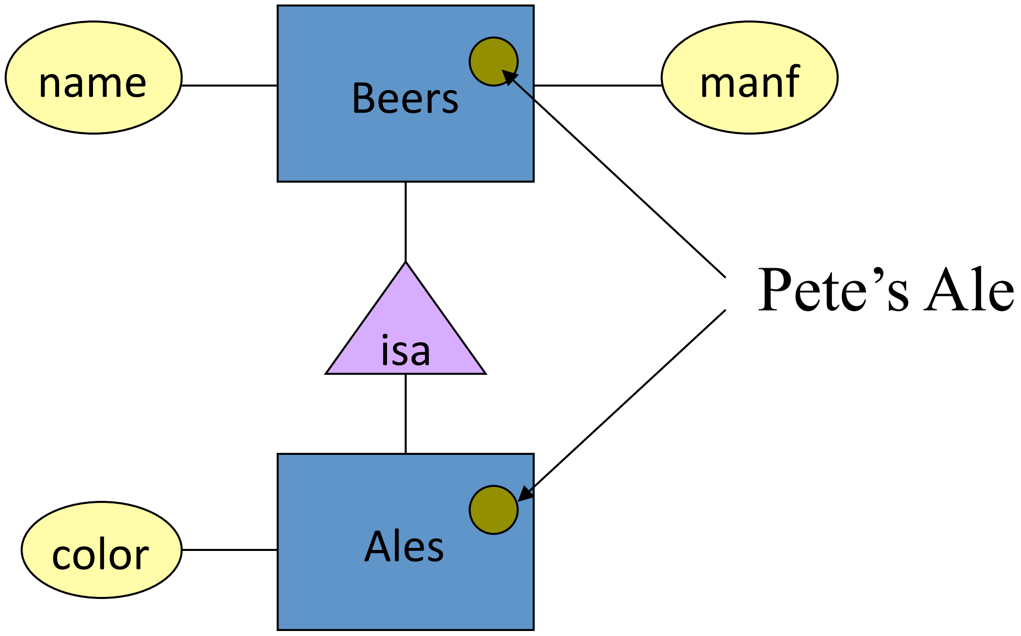
Example: Subclasses



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



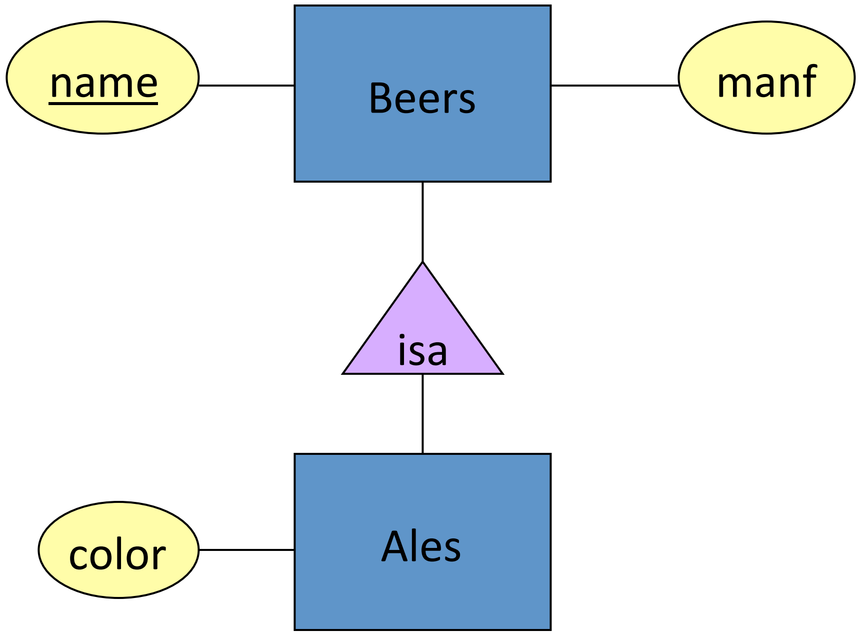
Keys

- 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.

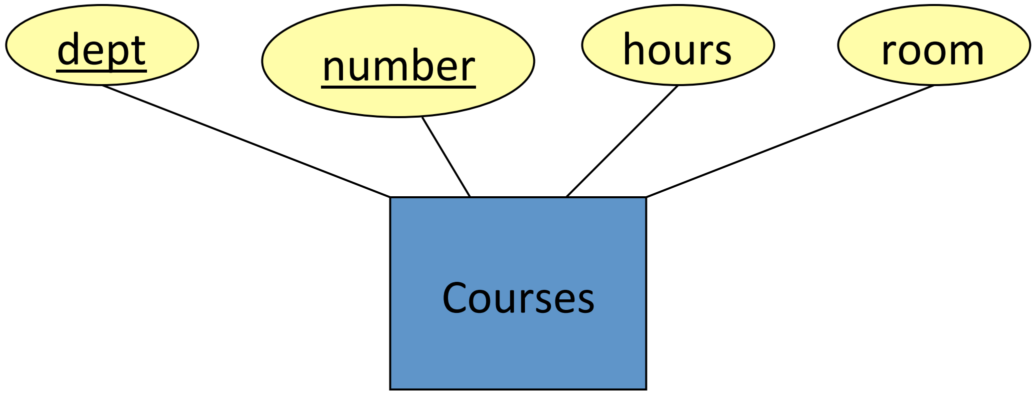
Keys in E/R Diagrams

- 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.

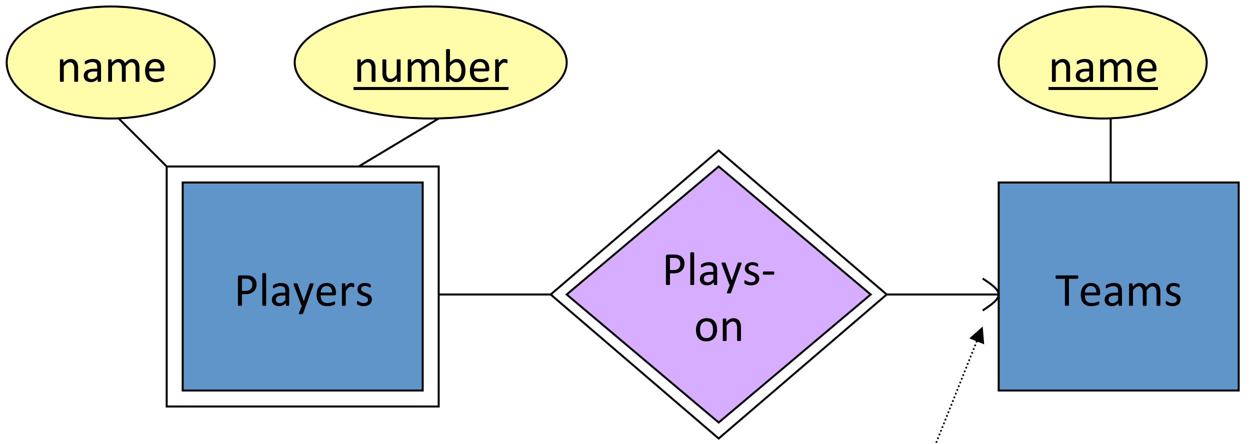
Weak Entity Sets

- 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 for the supporting entity sets.
  - E.g., (player) number and (team) name is a key for Players in the previous example.

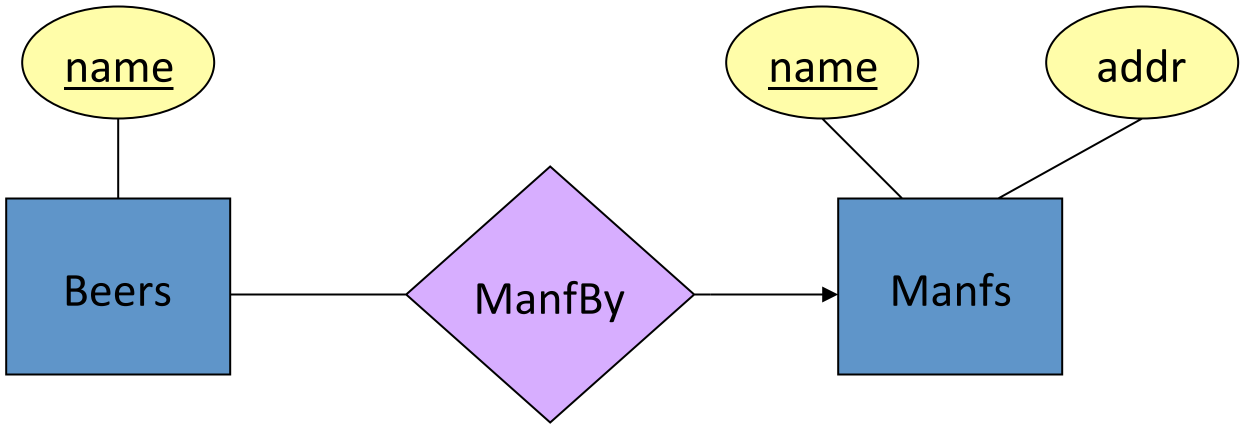
Design Techniques

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

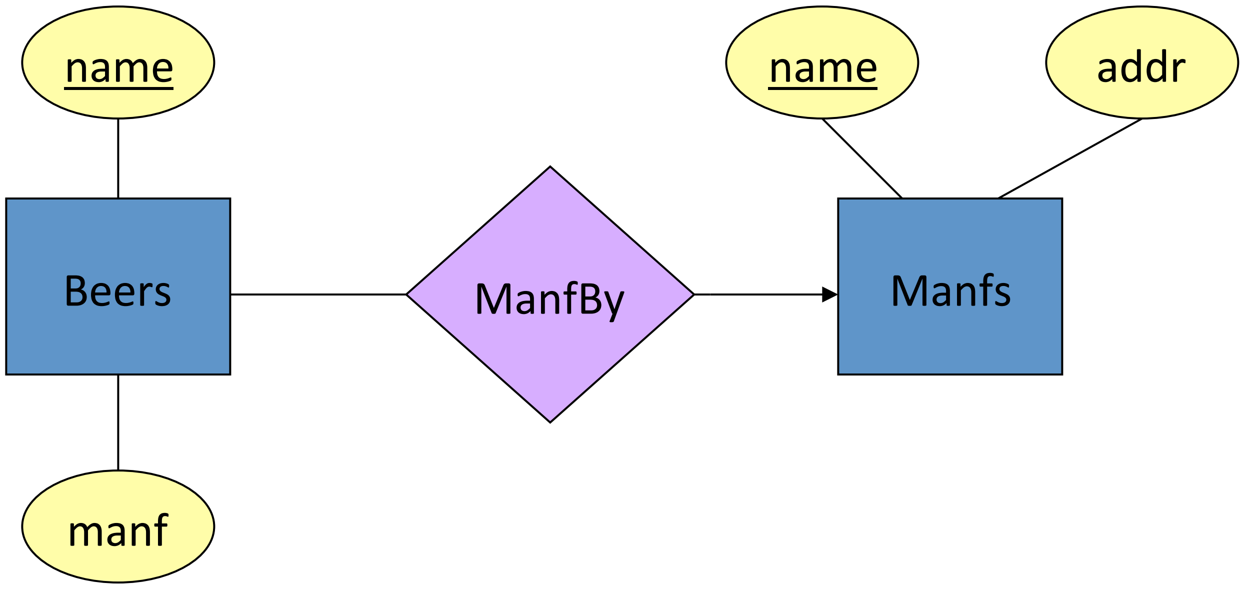
- *Redundancy* = saying the same thing in two (or more) different ways.
- Wastes space and (more importantly) facilitates inconsistency.
  - Two representations of the same fact become inconsistent if we change one and forget to change the other.

Example: Good



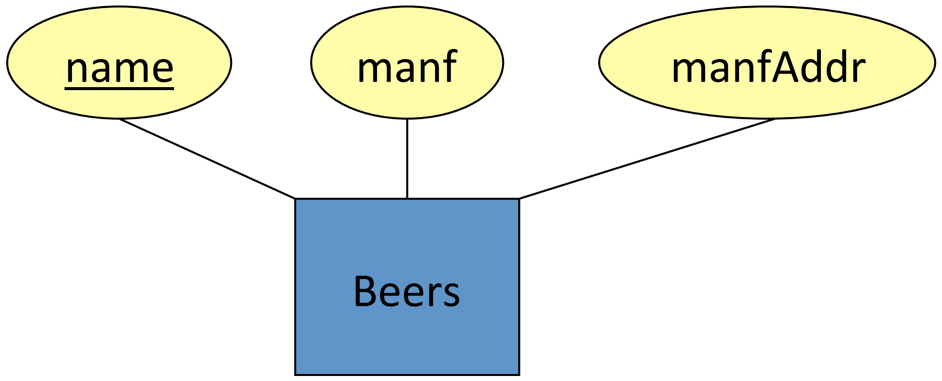
This design gives the address of each manufacturer exactly once.

Example: Bad



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

Example: Bad



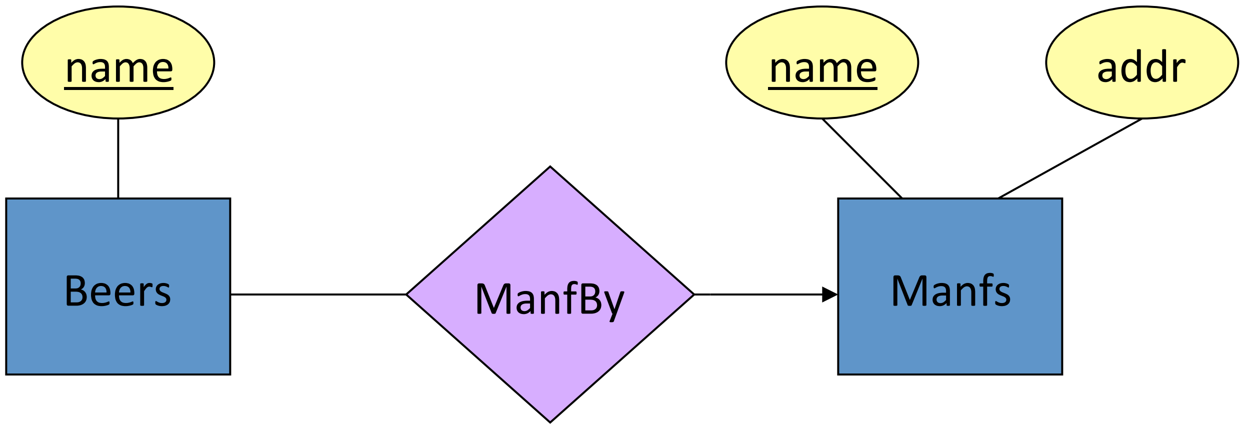
This design repeats the manufacturer’s address once for each beer and loses the address if there are temporarily no beers for a manufacturer.

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 nonkey attribute.

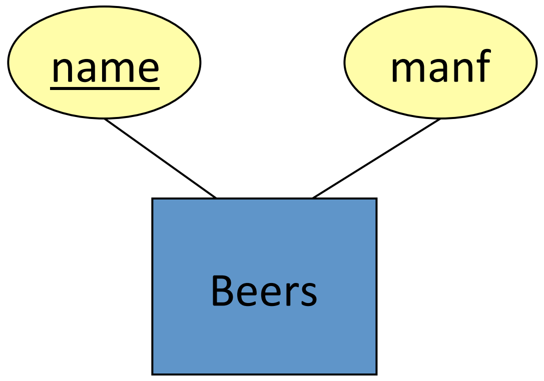
    or
  - It is the "many" in a many-one or many-many relationship.

Example: Good



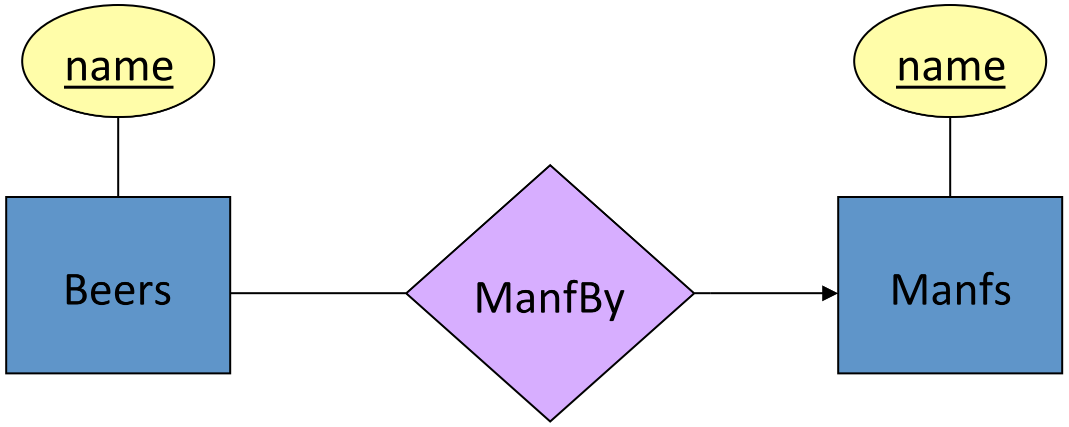
- Manfs deserves to be an entity set because of the non-key attribute addr.
- Beers deserves to be an entity set because it is the "many" of the many-one relationship ManfBy.

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 social-security numbers, automobile VIN’s etc.

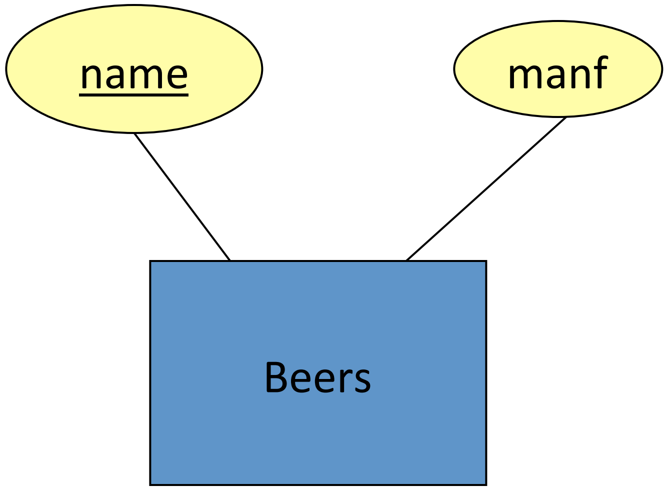
When Do We Need Weak Entity Sets?

- 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.

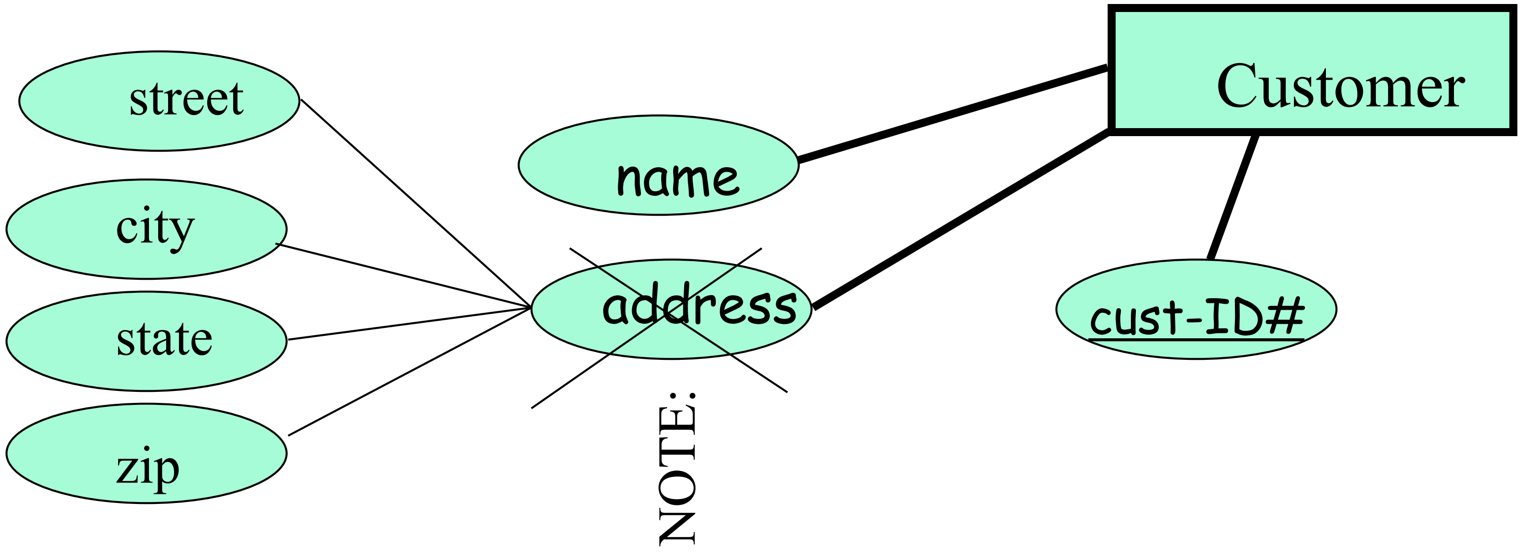
Entity Set -> Relation



Relation: Beers(name, manf)

ER-to-Relational

- Example of composite attribute

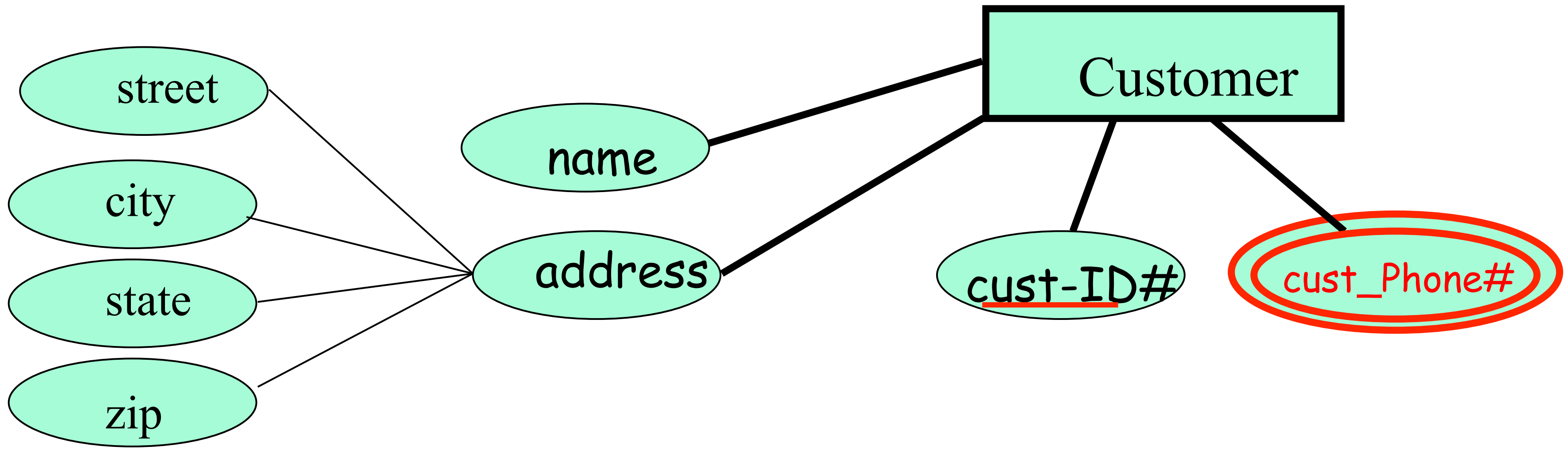


Becomes:

Customer(cust_ID#, name, street, city, state, zip)

ER-to-Relational

- Example of multi-valued attribute – we know that a customer entity-instance may have one or more phone #s:

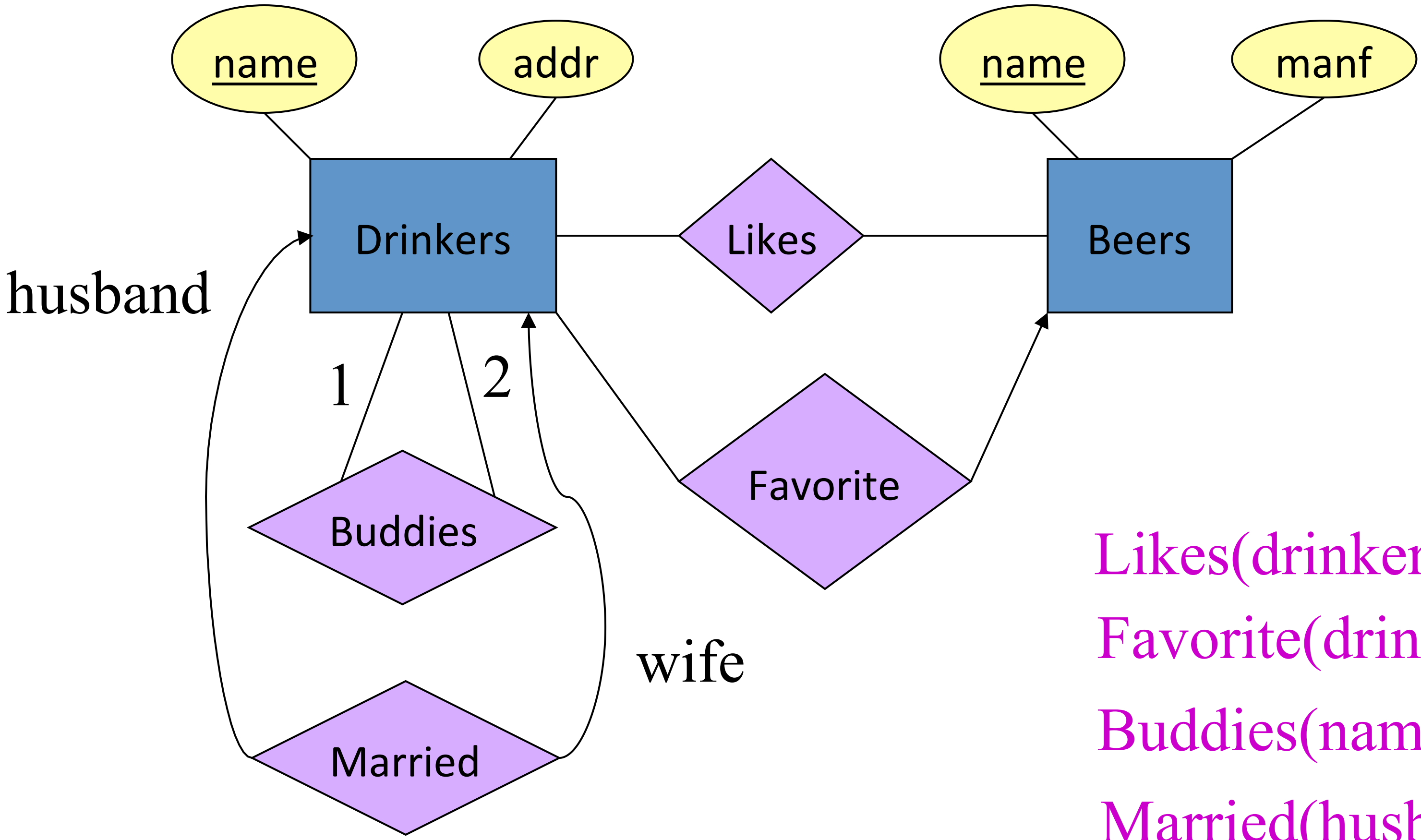


Becomes:

Customer(cust_ID#, name, street, city, state, zip)

CustomerMV(cust_ID#, cust_Phone#)

Relationship -> Relation (basic example)



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:

name	addr	beer
Sally	123 Maple	Bud
Sally	123 Maple	Miller

Redundancy

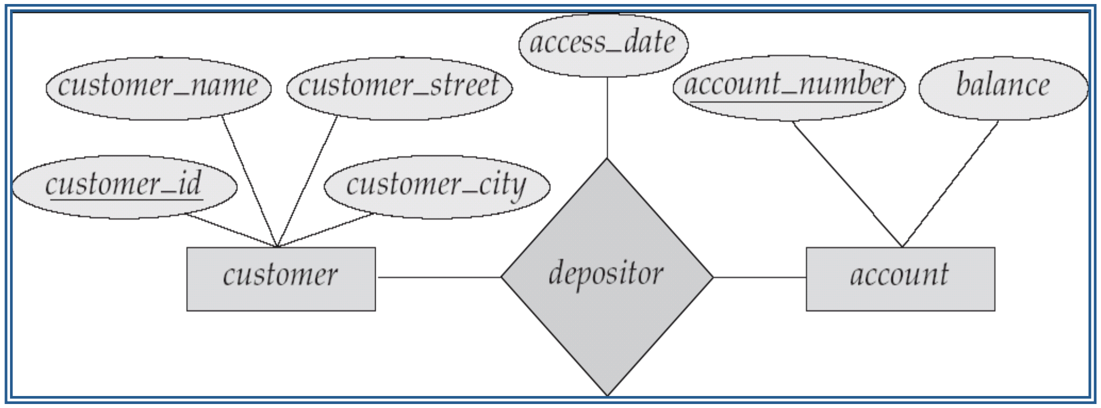
Relationships to Relations (basic rules)

- What about relationships?
  - The “ ♦ “ still needs to become ***some type of a table***…

Basic “temptation”:

Get the attributes from each participating entity set, plus the attributes of the relationship itself, and form a separate table with all three of them “dumped”…

Ex:



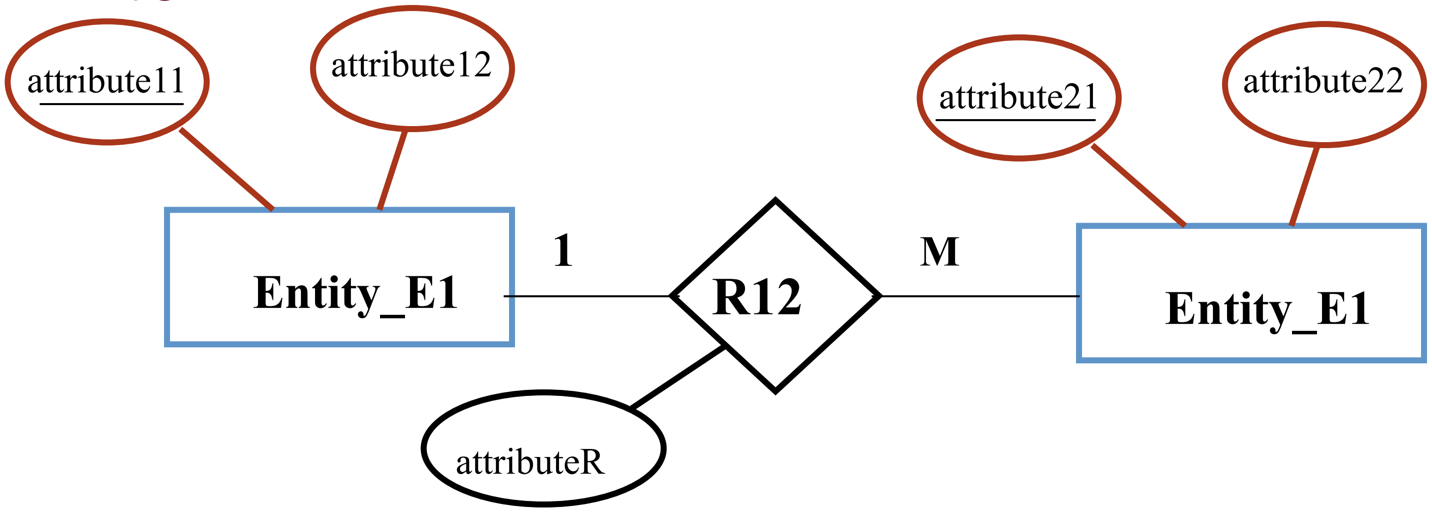
Create the schema:

depositor(customer_id,customer_name,customer_street,customer_sity,acces_date,account_number,balance)

ER-to-Relational

- **The impact of the cardinality type/constraint of the relationship:**

1-to-M



RULE:
- Add the key of "1" entity, as a ***foreign key*** to the table of "M" entity
AND
- Add the attributes of the relationship type

Entity_1(attribute11, attribute12)

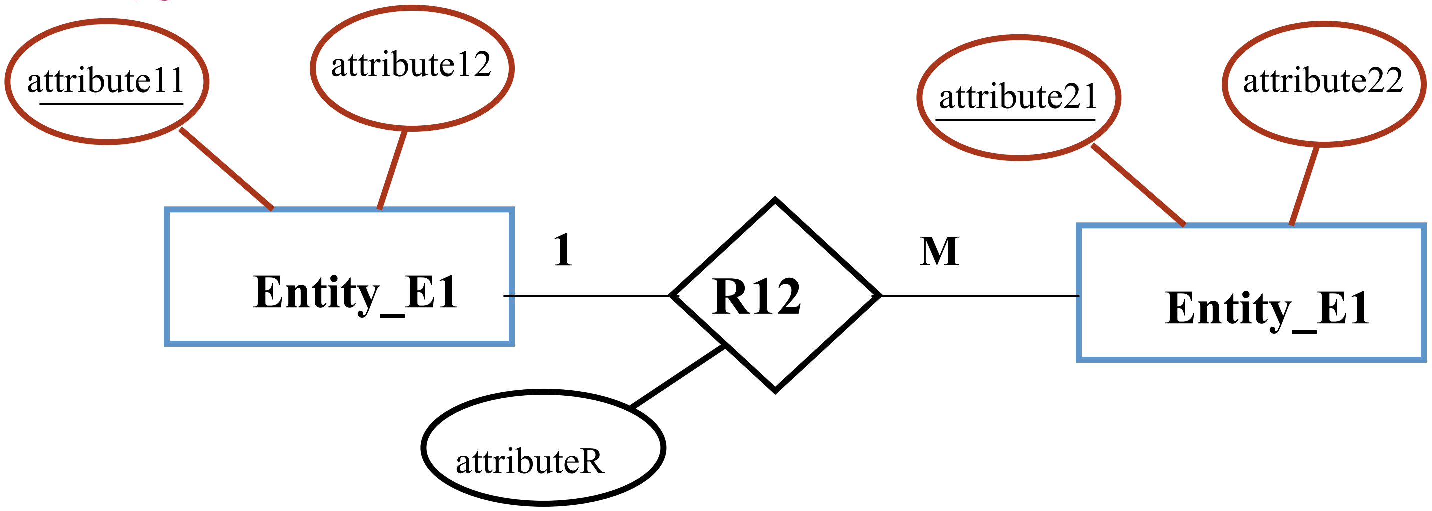
Entity_2(attribute21, attribute22, attribute11, attributeR)

Q: why not "the other way around"???

ER-to-Relational

- **The impact of the cardinality type/constraint of the relationship:**

1-to-M



RULE:
- Add the key of "1" entity, as a ***foreign key*** to the table of "M" entity
AND
- Add the attributes of the relationship type

Entity_1(attribute11, attribute12)

Entity_2(attribute21, attribute22, attribute11, attributeR)

Q: why not "the other way around"???

ER-to-Relational

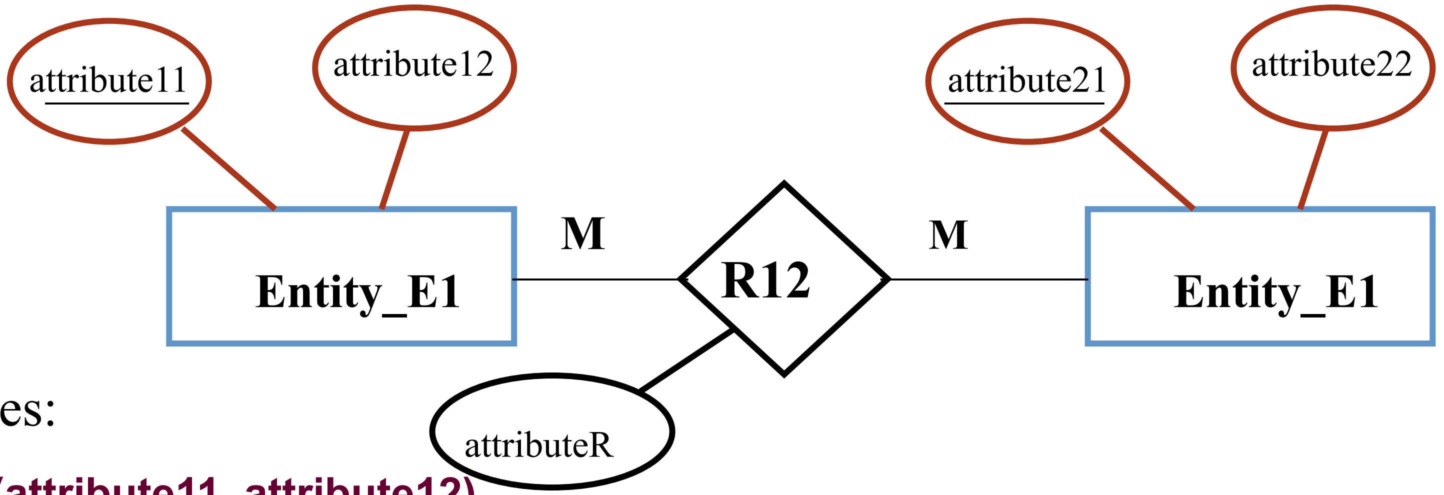
- **Dealing with M-to-1 relationships:**
  - **mirror_image of 1-to-M**

- **Dealing with 1-to-1 relationships:**
  - **similar "in spirit" to 1-to-M: just pick one of the relationships to "act" as if it's "M" and add the foreign keys (+ the attributes of the relationship types)**
  - **NOTE: it may be tempting to create a single table with all of the Entity_1,Entity_2 and R12 attributes, because, if we know it's 1-t-1, then there will be no redundancy, in the sense of the previously-discussed "example"**
  - **HOWEVER, this is a *poor design* because it blurs the actual semantics of the problem domain (in reality, Entity_1 and Entity_2 are separate classes!).**

- **Q: how about if a given entity participates in > 1 relationships**
  - **A: handle each relationship "one-at-a-time"**

ER-to-Relational

- **M-to-M:**
  - **Observe that 1-to-M was handled by "squeezing" the relationship type as an attribute, plus taking care of the keys (foreign) for the purpose of proper maintenance of the data-relationships**
  - **Now we must create a separate table for the relationship type, with added foreign keys = the keys of the participating entity classes (plus the "own" attributes of the relationship)**



Becomes:

Entity_1(attribute11, attribute12)

Entity_2(attribute21, attribute22)

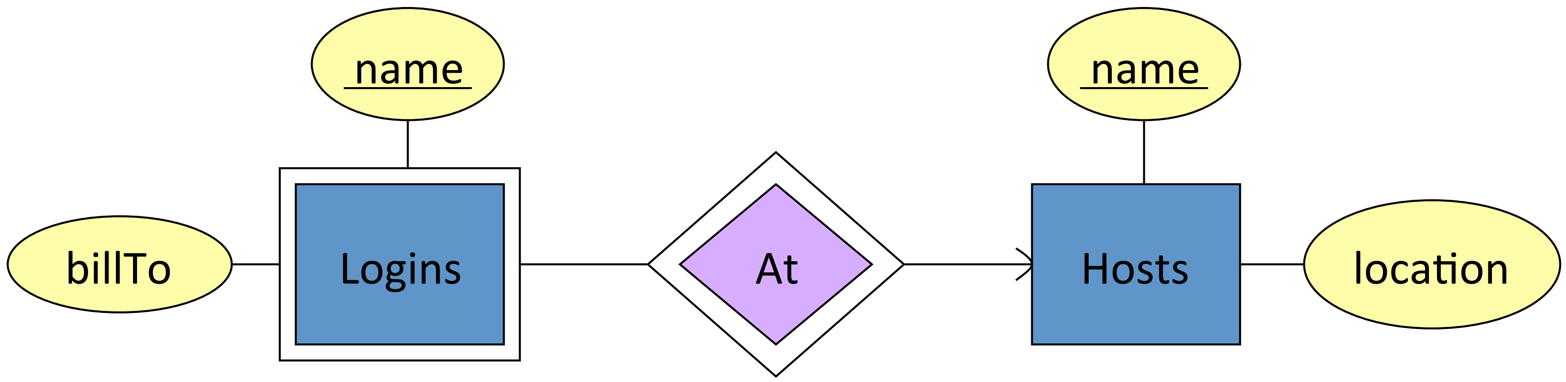
R12(attribute11, attribte12, attributeR)

Q: what is the problem if we try to "mimick" 1-to-M?

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, ~~hostName2)~~

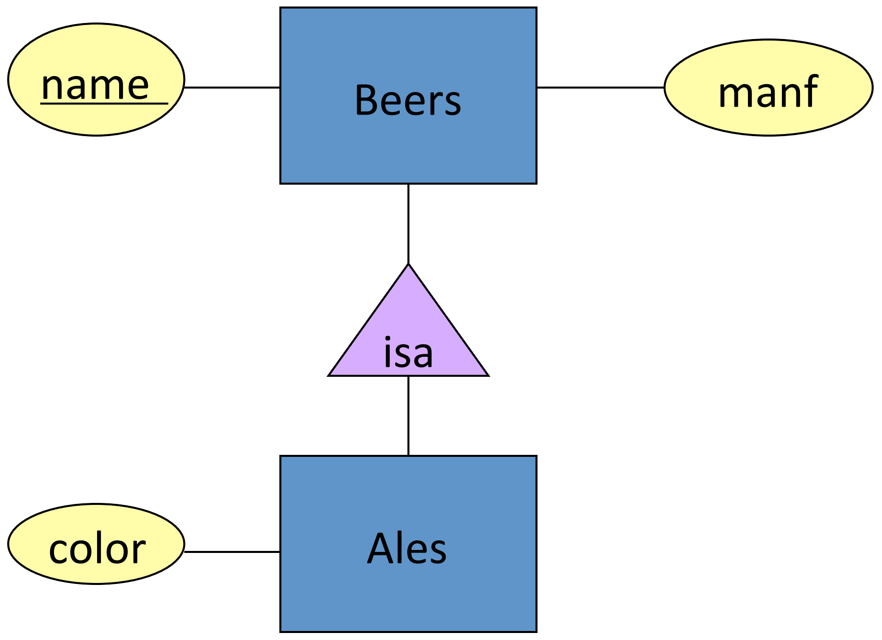
At becomes part of Logins

Must be the same

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
Bud	Anheuser-Busch

Beers

name	manf	dcolor
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

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

Conclusions

- E/R diagrams enable database designers to reason about how to represent their data
- Translate data into *entities* (and sets thereof) and connect them via *relationships*
- Diagrams are simple to translate into relations