

By Marina Barsky

Entity-Relationship diagrams: refinements

Lecture 2

Recap

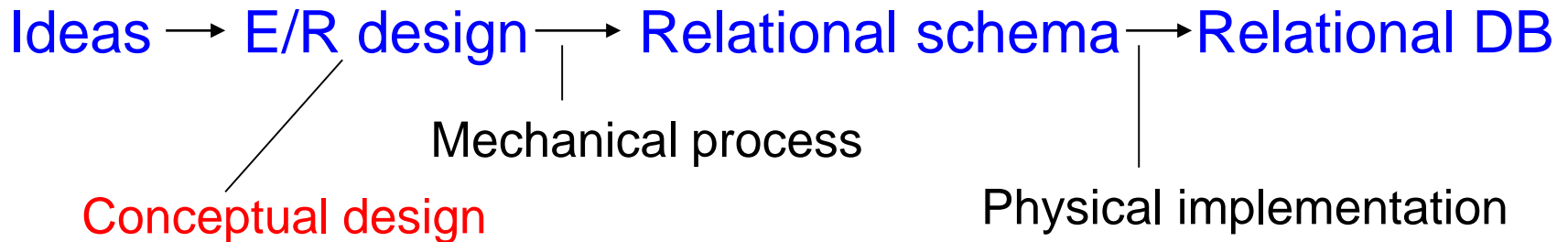
A *data model* is a collection of concepts

A *schema* is a description of data, using data model.

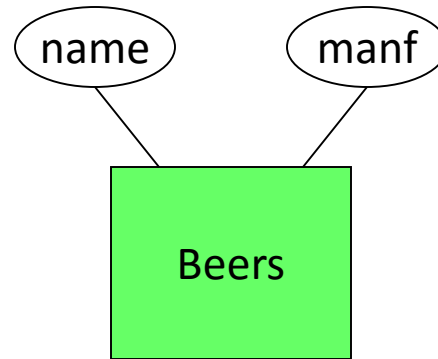
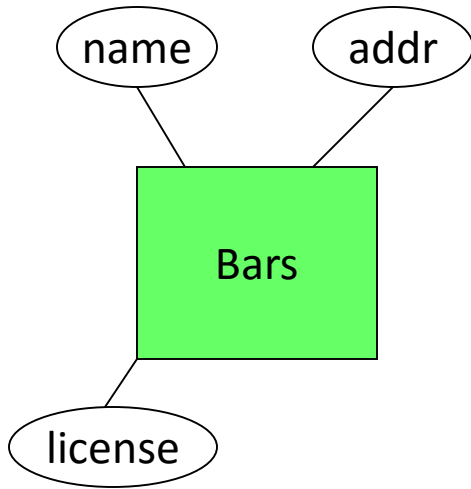
A *database (instance)* is a collection of data compliant with the schema

Process of database design

- **Notation** for expressing designs: Entity-Relationship (E/R) model



Step 1. Identify **entities** (entity sets) and their attributes



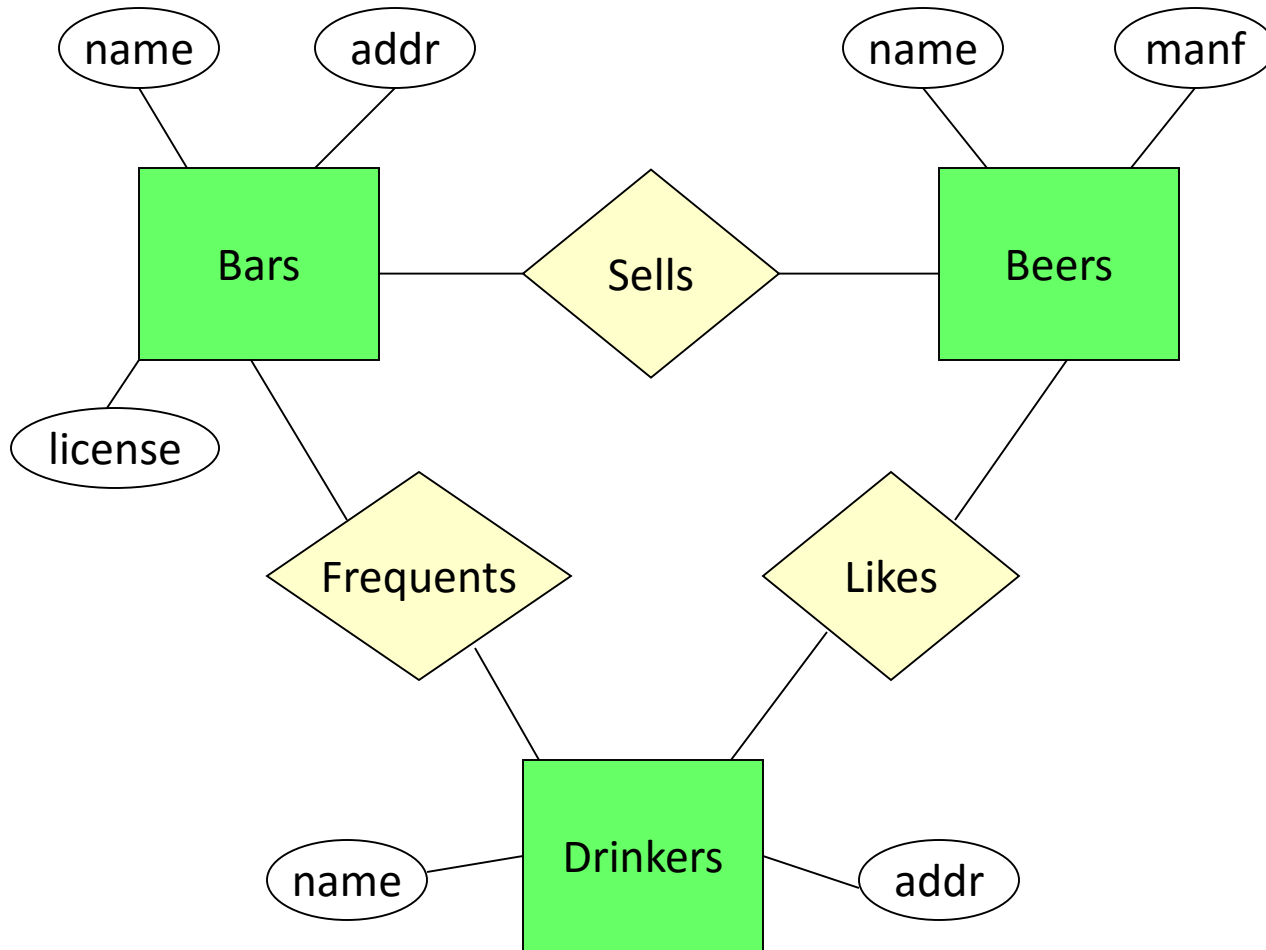
Bars sell some beers.

Drinkers like some beers.

Drinkers frequent some bars.



Step 2. Identify **relationships** between entities (relationship sets) and their attributes

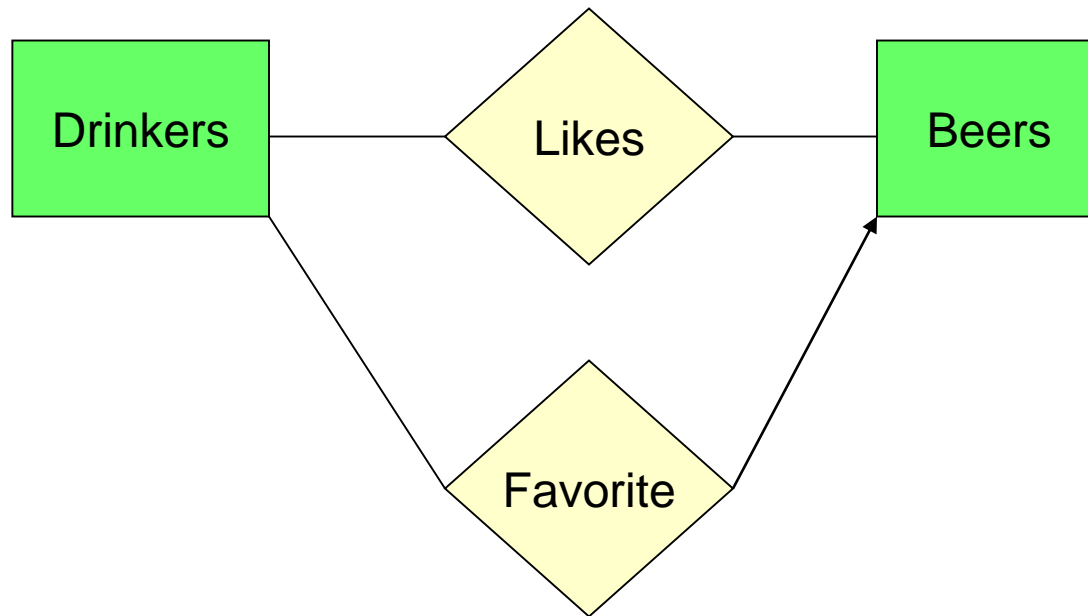


Bars **sell** some beers.

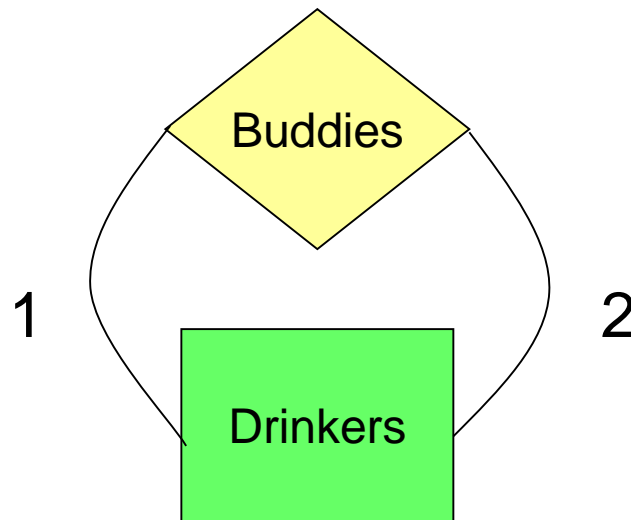
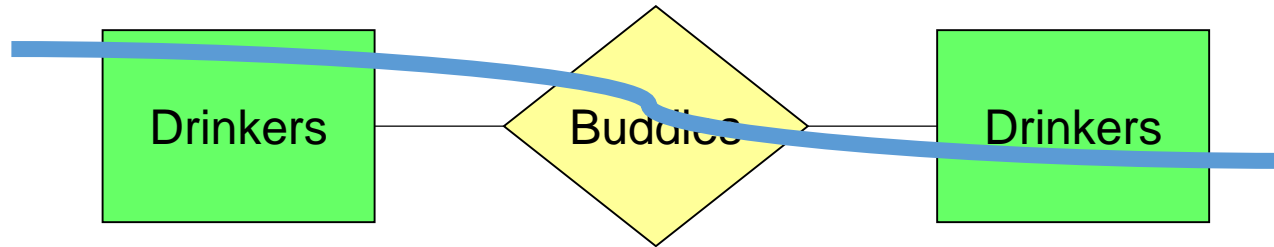
Drinkers **like** some beers.

Drinkers **frequent** some bars.

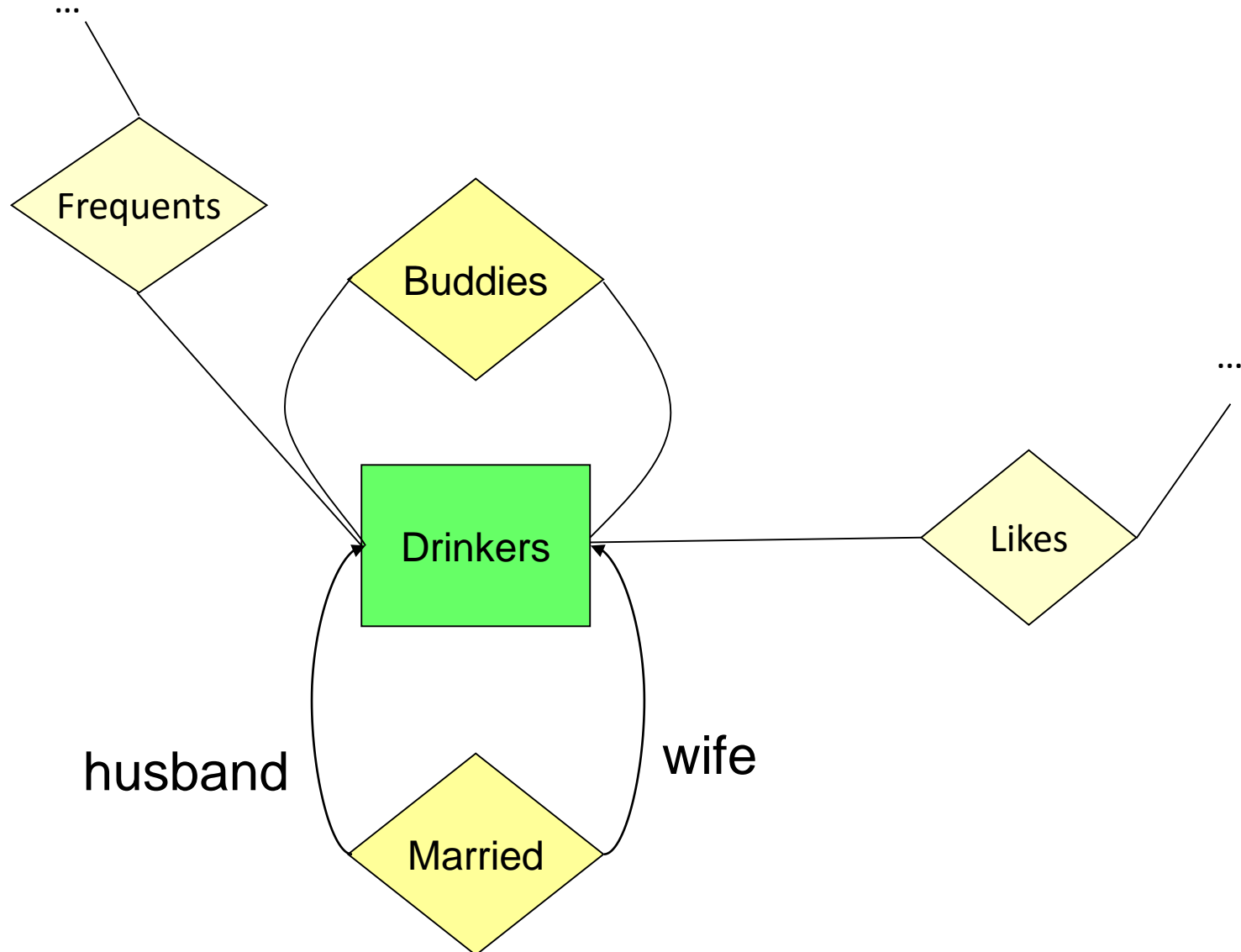
Multiple relationships may exist between two entity sets



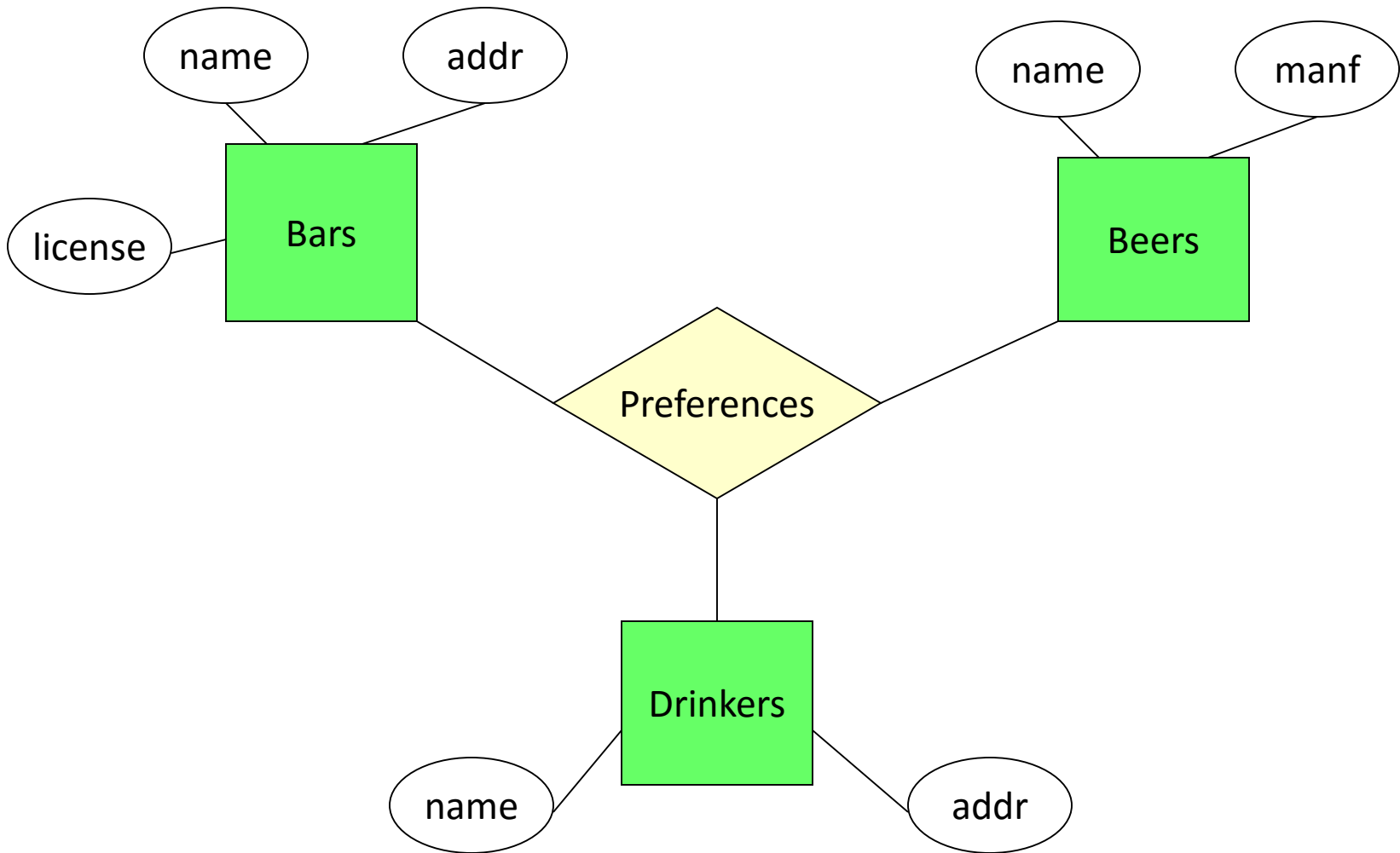
Recursive (self)-relationships



Same entity in different roles

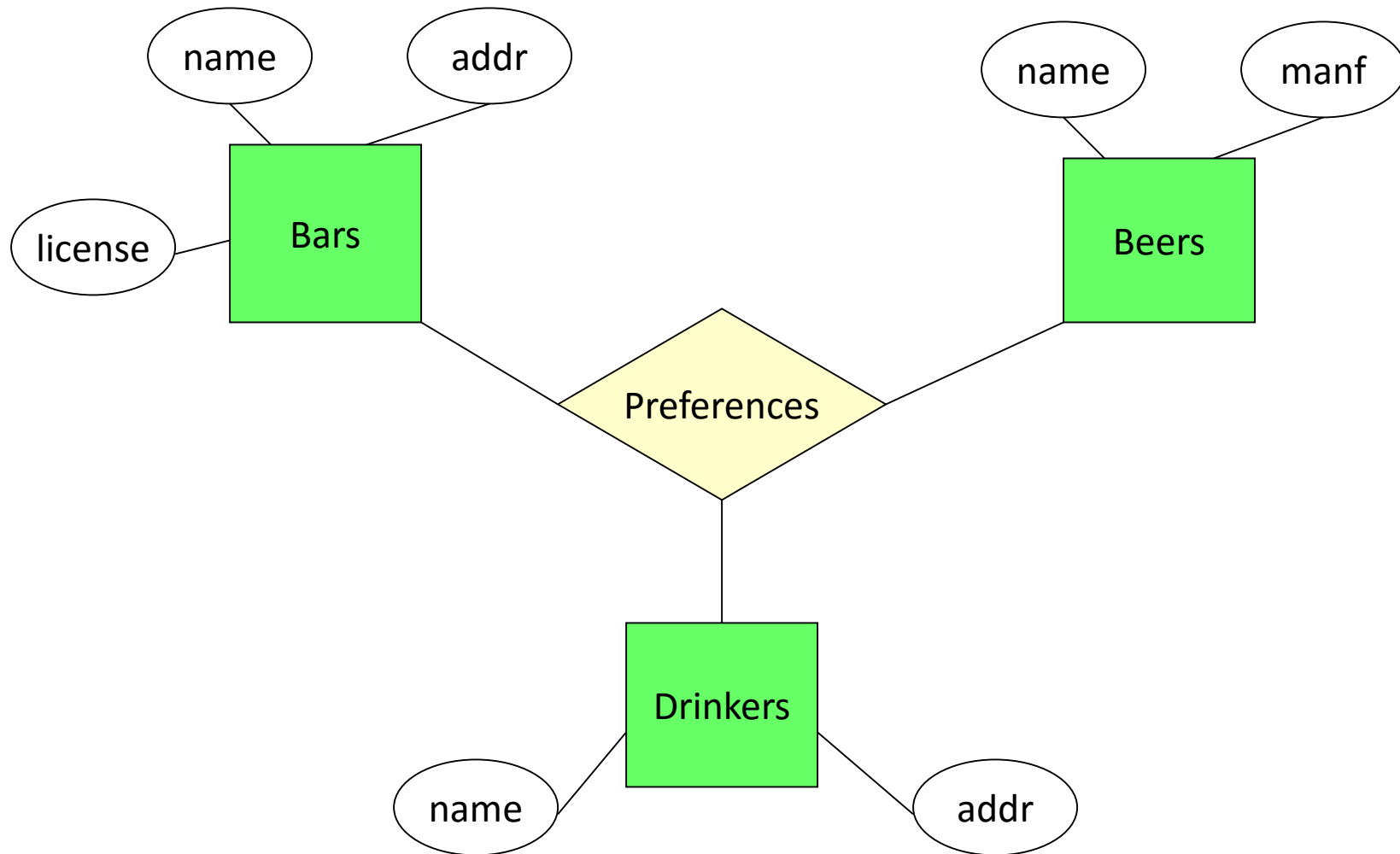


Ternary relationships



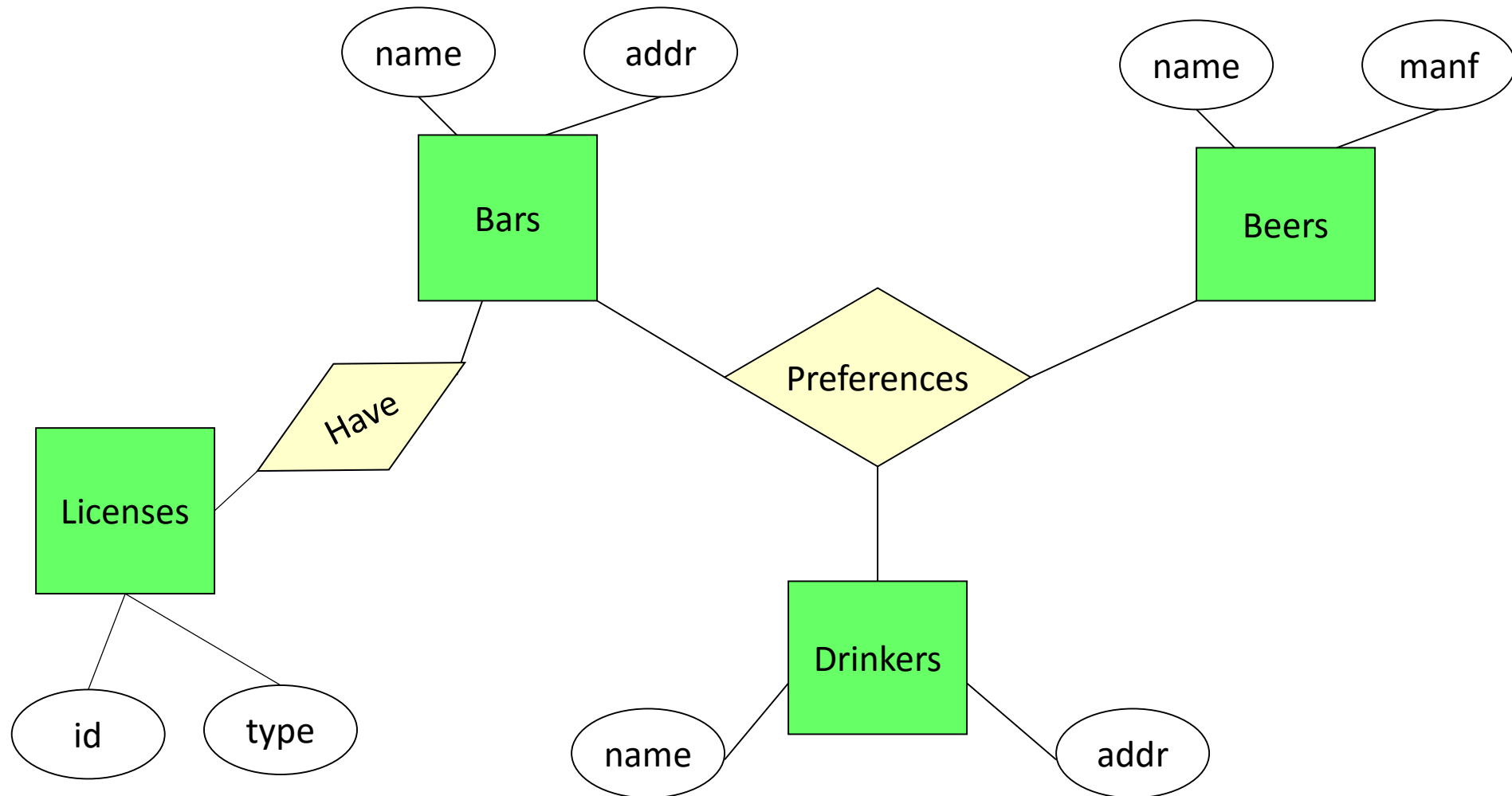
Multivalued attributes

- Several licenses per bar

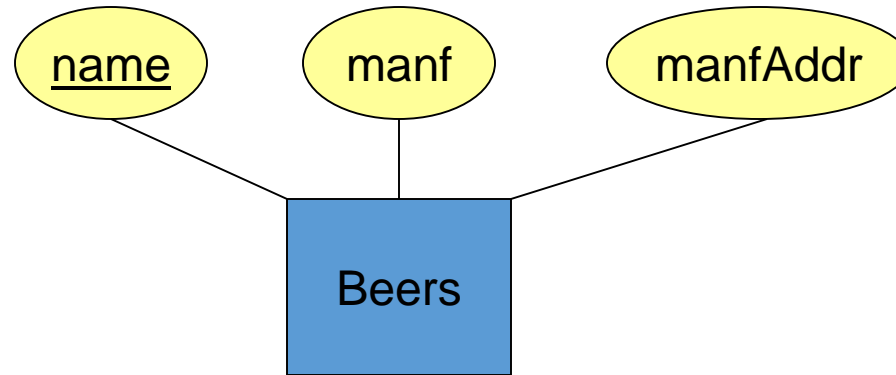


Multivalued attributes

- Several licenses per bar



Entity Sets Versus Attributes



Example of bad design. Why?

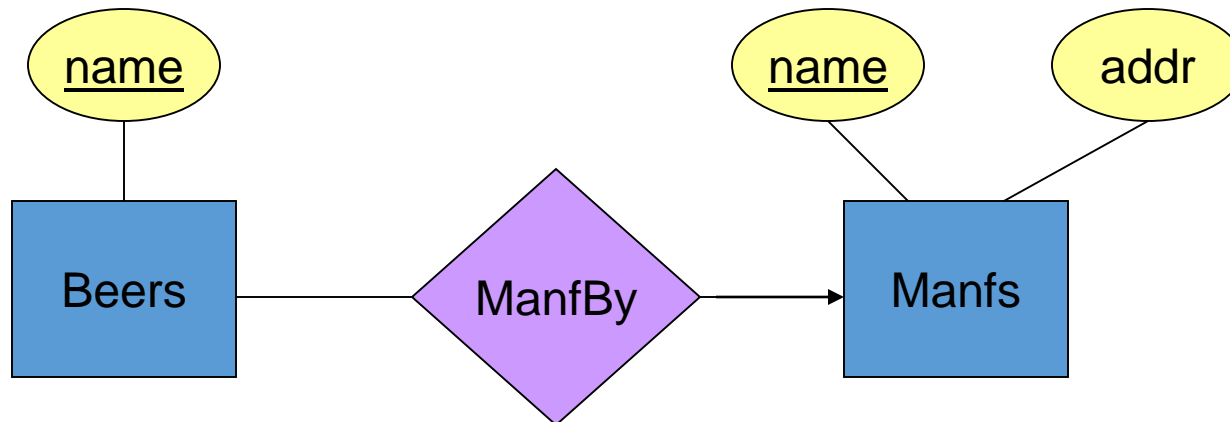
1. Repeats the manufacturer's address once for each beer;
2. Loses the address if there are temporarily no beers for a manufacturer.

When to replace attribute with an entity set

- 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.

From attributes to entity sets






- *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*.



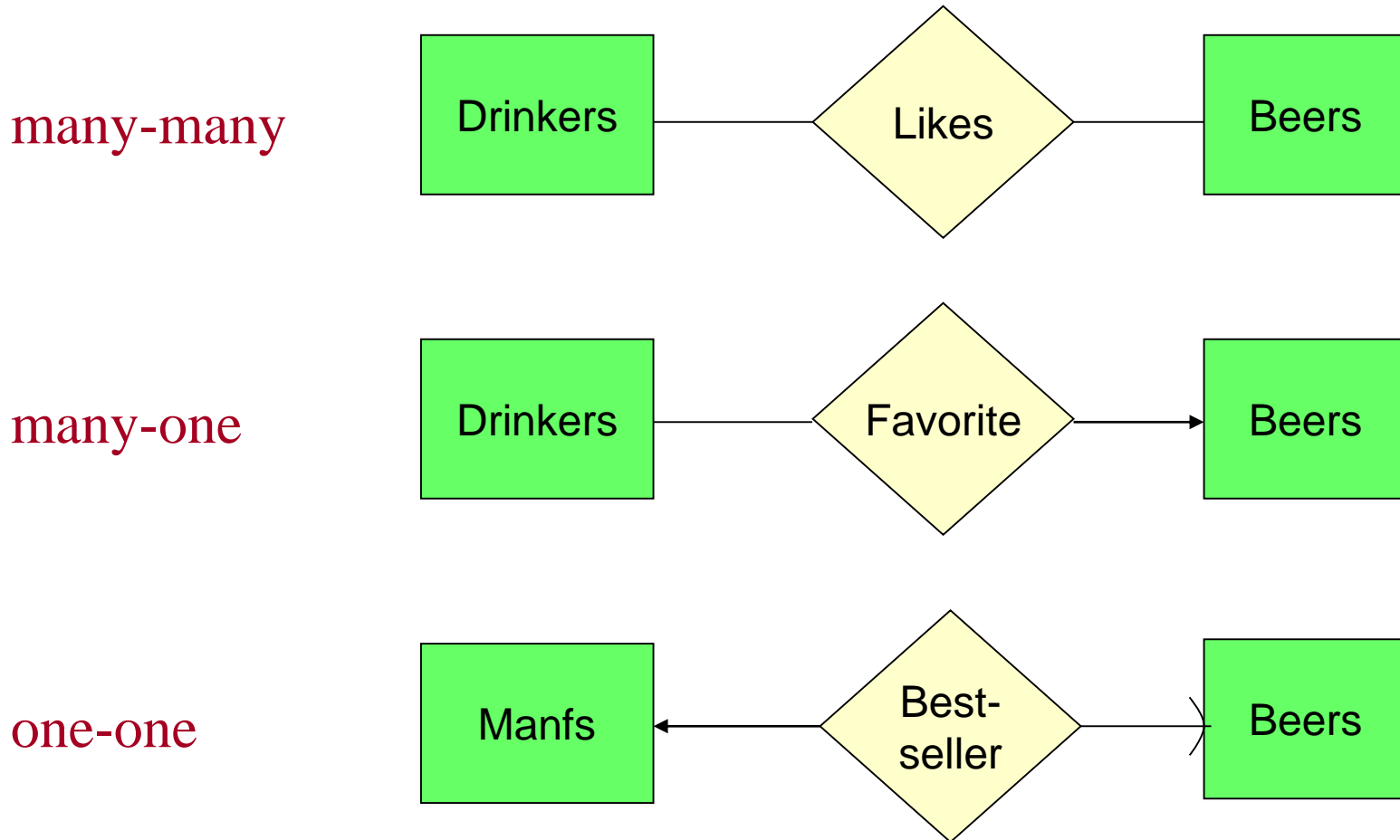
Attribute vs. entity: summary

- 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 additional attribute.
 - or
 - It is the “many” in a many-one or many-many relationship.
- Intuition
 - A “thing” in its own right => Entity Set
 - A “detail” about some other “thing” => Attribute

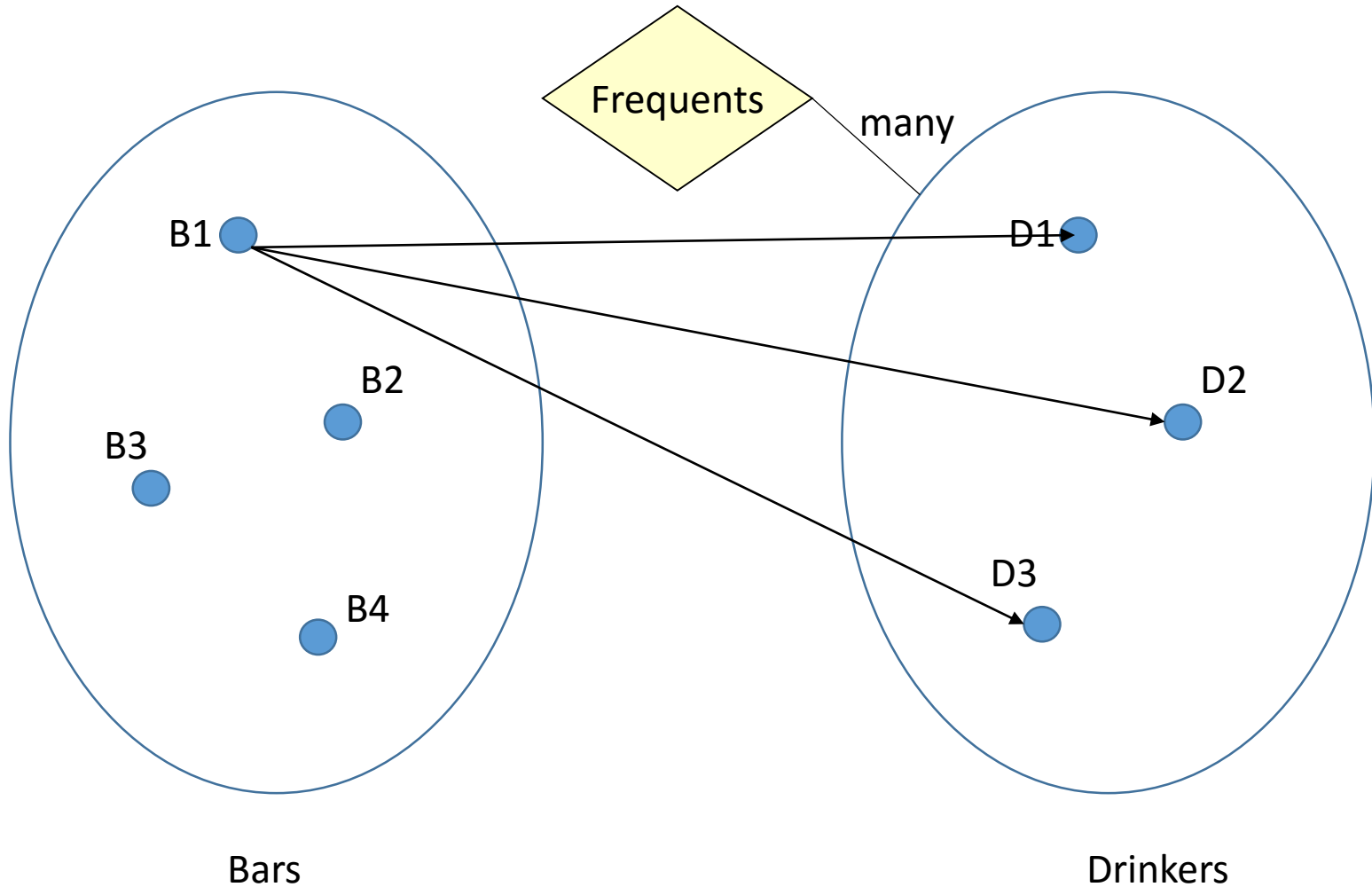
Multiplicity of relationships

- many-to-many (binary or ternary) 
- one-to-many
 - mandatory: 
 - optional: 
- one-to-one
 - both mandatory: 
 - one mandatory, one optional: 

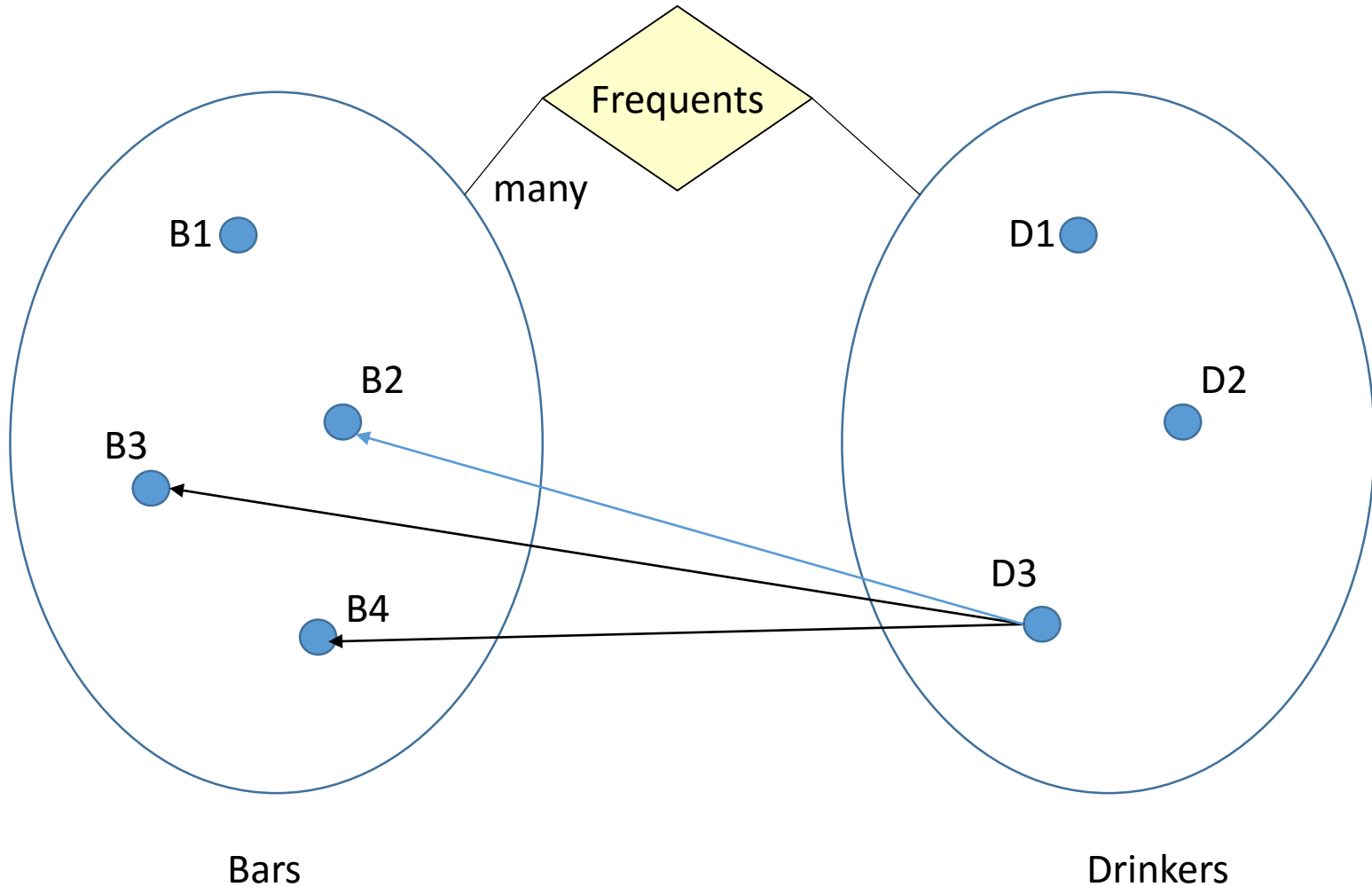
Multiplicity of Relationships



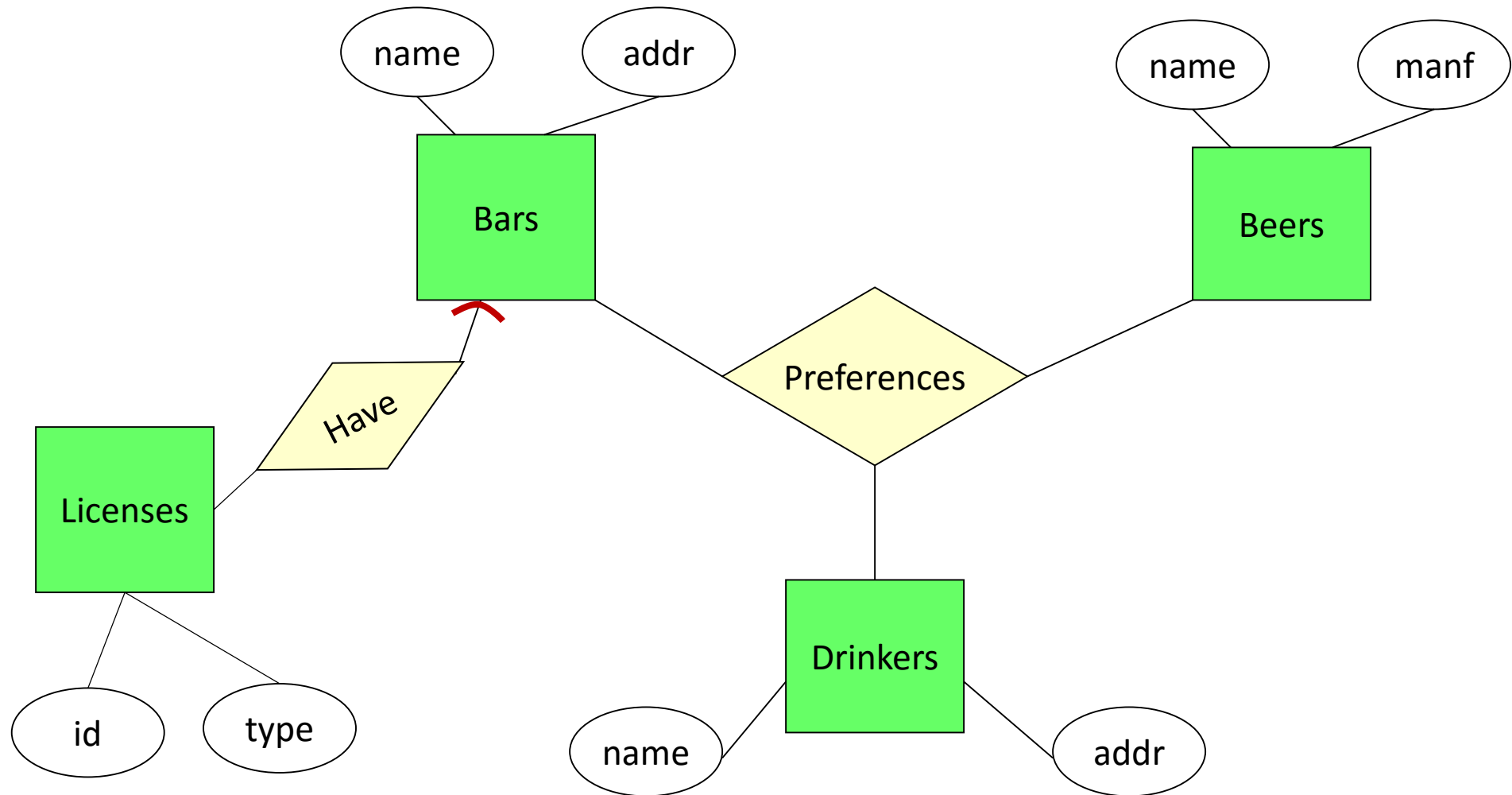
Testing multiplicities



Testing multiplicities



Multiplicity of Relationships



Basic steps

- Identify entities (entity sets) and their attributes
- Identify relationships between entities (relationship sets) and their attributes
- Are there recursive (self)-relationships?
- Are there different roles for the same entity?
- Do we need ternary relationships?
- Attribute or entity?
- Mark multiplicity

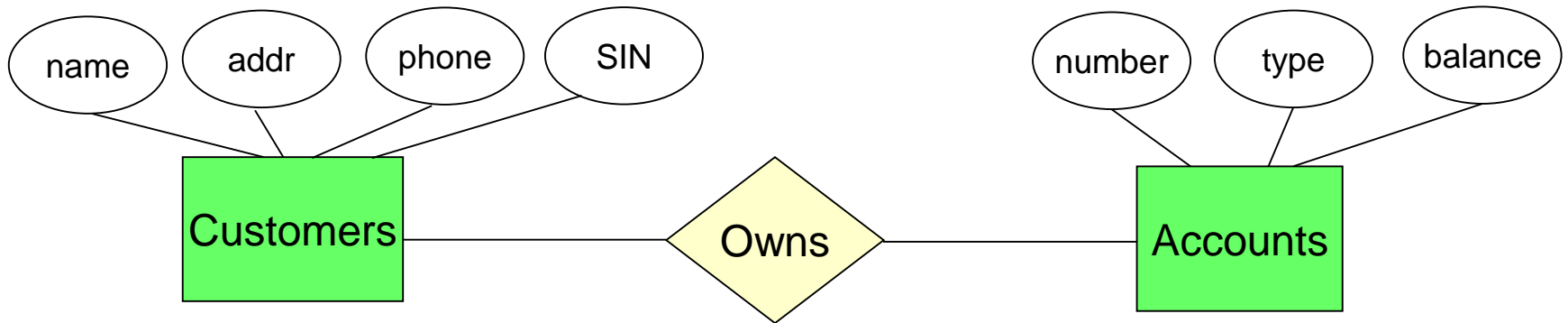
Exercise

Exercise 1. Bank database

- Let us design a database for a bank, including information about customers and their accounts.

Information about a **customer** includes their **name**, **address**, **phone**, and **SIN** number. **Accounts** have **numbers**, **types** (e.g., savings, checking) and **balances**. We also need to record the customer(s) who own an account. Draw the E/R diagram for this database.

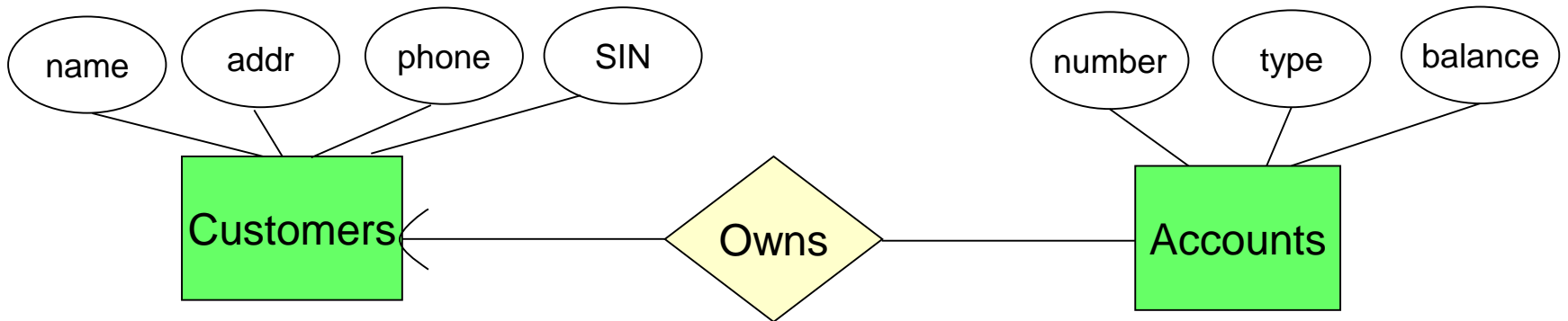
Exercise 1 solution



Exercise 1A. Bank database

- Modify your solution as follows:
 - a) Change your diagram so an account can belong to **only one customer**.

Exercise 1A solution

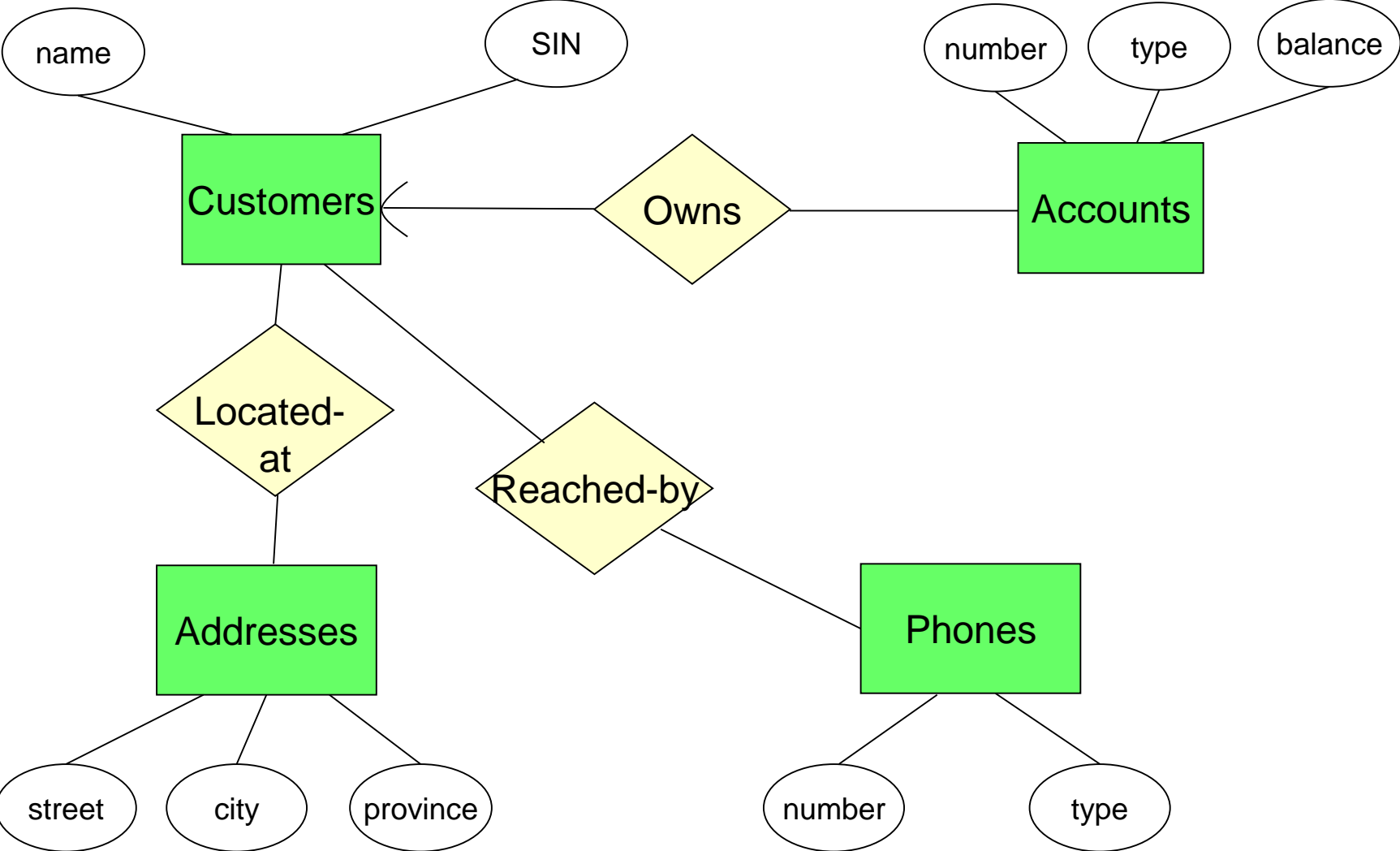


Exercise 1B. Bank database

- Modify your solution as follows:
 - a) Change your diagram so an account can have only one customer.
 - b) Change your diagram so that **a customer can have a set of addresses** (which are street-city-province triples) and a set of phones.

Remember that we do not allow attributes to have non-atomic types, such as sets, in the E/R model.

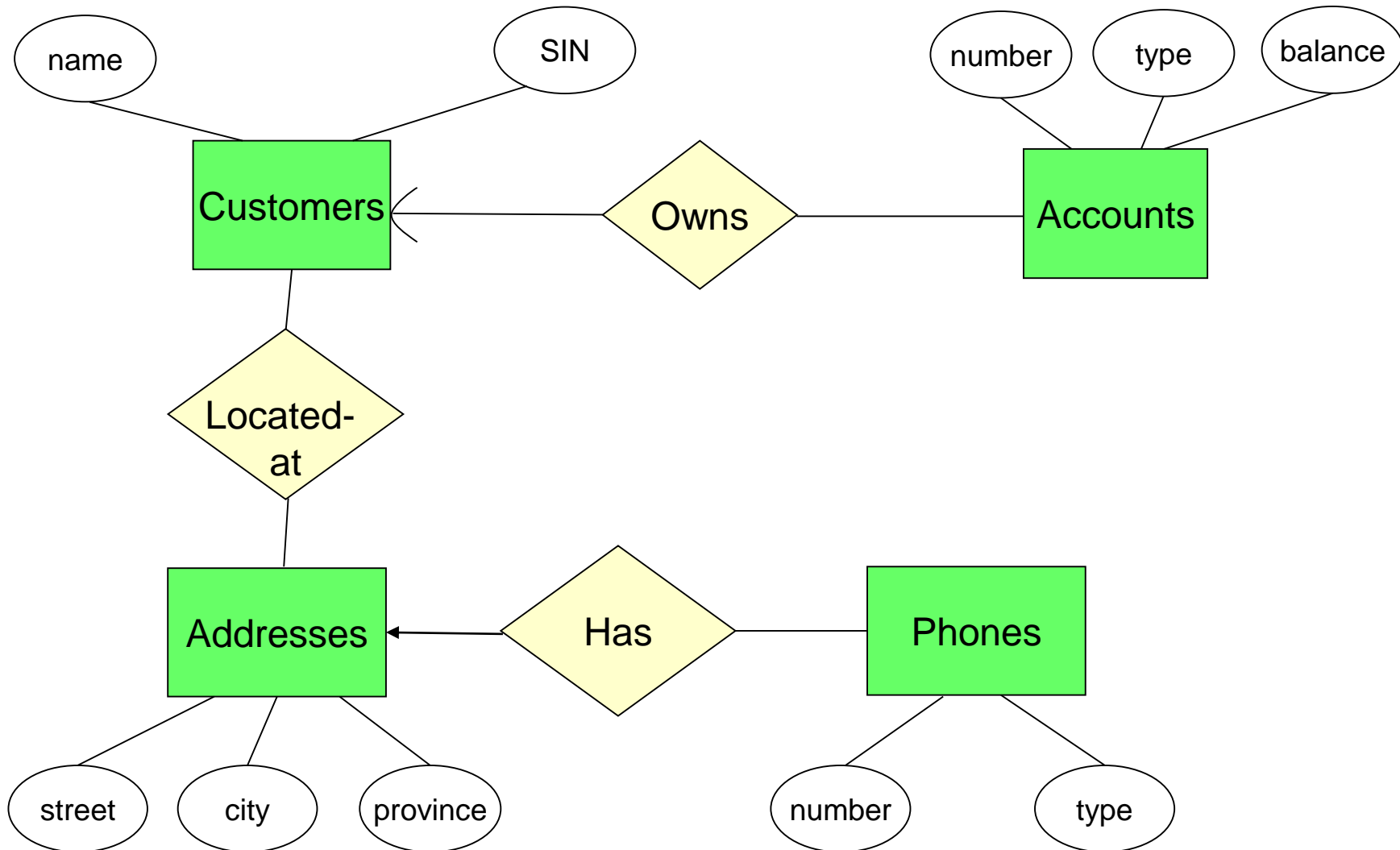
Exercise 1B solution



Exercise 1C. Bank database

- Modify your solution as follows:
 - a) Change your diagram so an account can have only one customer.
 - b) Change your diagram so that a customer can have a set of addresses (which are street-city-province triples) and a set of phones.
 - c) Further modify your diagram so that customers can have a set of addresses, and **at each address there is a set of phones.**

Exercise 1C solution



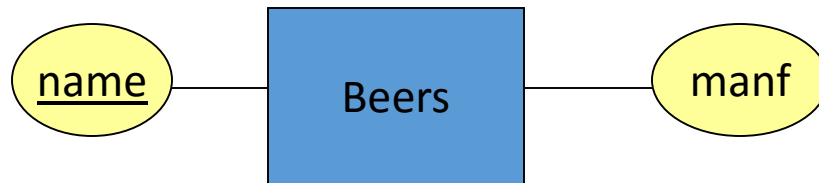
E/R refinements

Keys

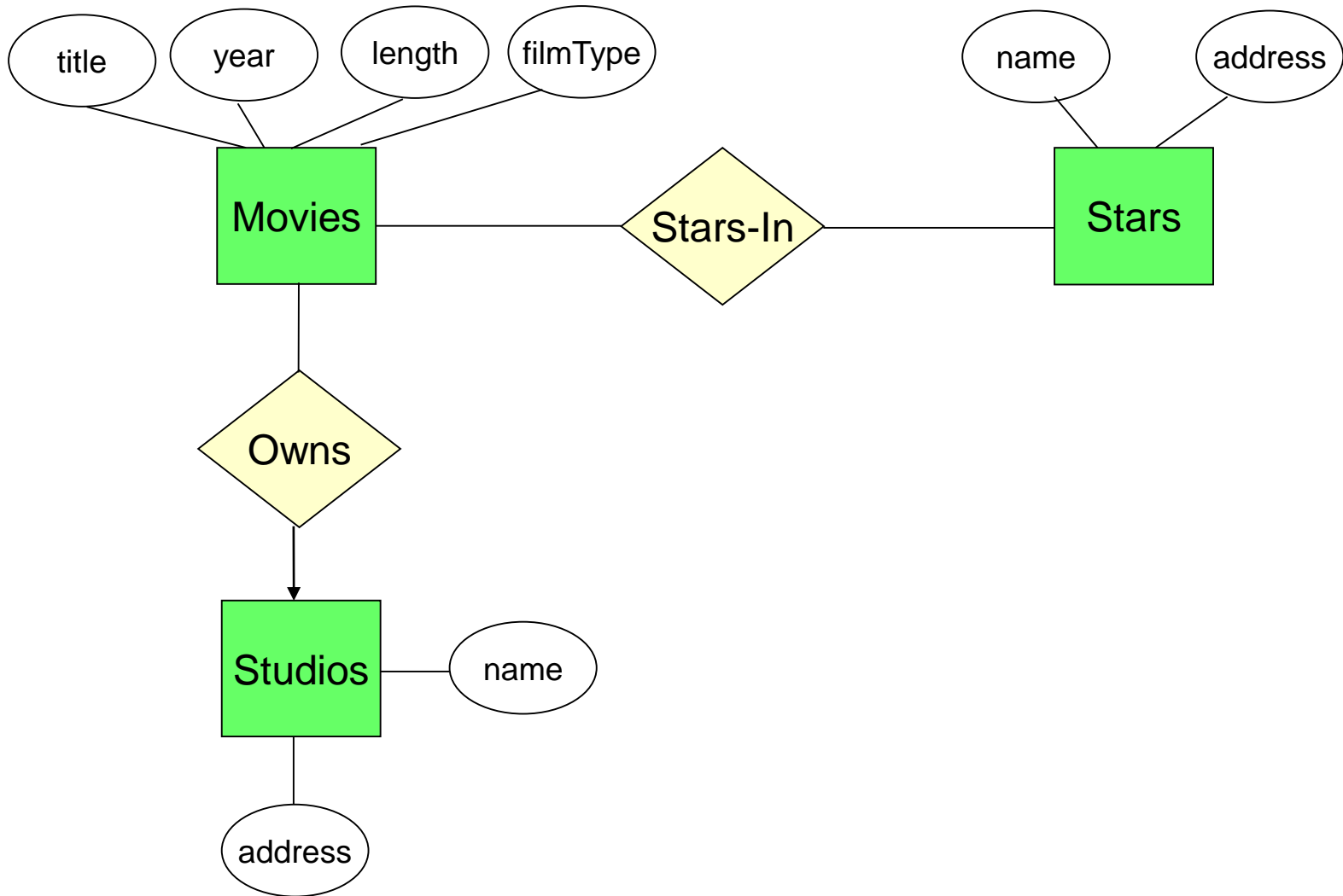
for entity sets

Keys

- A **key** (for an entity set) is a set of attributes such that **no two entities agree on all the attributes of the key.**
- In E/R, we underline the key attribute(s).



Keys?



Key for Movies

Let's consider entity set **Movie**

{title, year}

Key for Studios and Stars

- For **Studios:**
name

- For **Stars:**
name

Internal vs. surrogate Keys

- In most cases, a key is formed by one or more attributes of the entity itself (*internal* keys)
- Often, people introduce attributes whose role is to serve as a *surrogate* key.
 - Companies assign employee ID's to all employees, and these ID's are carefully chosen to be unique numbers.
 - In Canada everyone has a SIN.
 - Students ID's in universities
 - Driver license numbers
 - Automobile registration numbers

Every entity should have a key

- Attributes with possible missing values cannot form a key
- Internal keys preferable to surrogate key
- One/few attributes is preferable to many attributes

Keeping keys simple

Multiple attribute and/or string keys...

- **...are wasteful**
 - e.g. Movies (title, year,...): 2 attributes, ~16 bytes
 - Number Of movies ever made $\ll 2^{32}$ (4 bytes) \Rightarrow Integer movieID key saves 75% space
- **...break encapsulation**
 - e.g. Parent(firstName, lastName, phone,...)
 - Security/privacy hole \Rightarrow Integer parentID prevents information leaks
- **...can change**
 - Name or phone number change?
 - Parent and child with same name?
 - Parent with no phone?

Keeping keys simple

Multiple attribute and/or string keys...

- **...are wasteful**
- **...break encapsulation**
- **...can change**

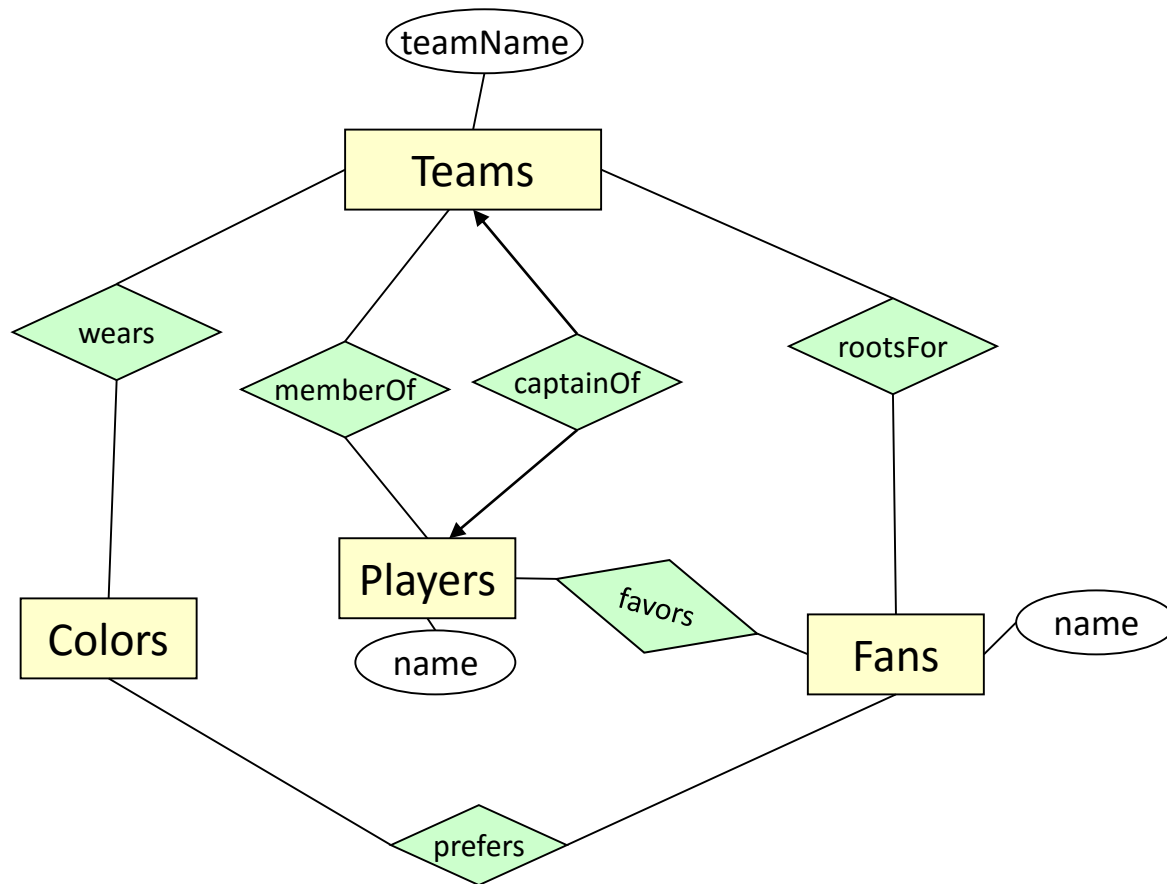
Numeric surrogate IDs always exist, are immutable, unique

Also: computers are really good at integers

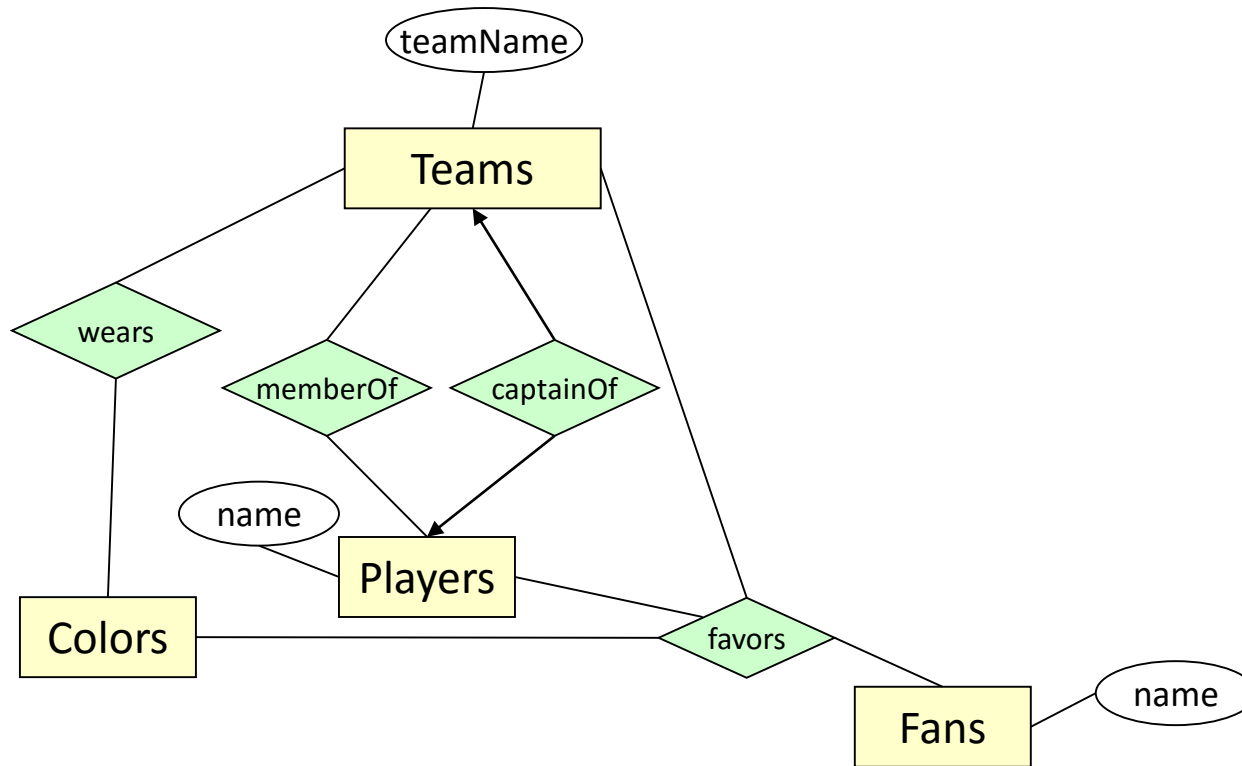
Exercise 2. Teams and fans database

- Give an E/R diagram for a database recording information about **teams**, **players**, and their **fans**, including:
 - For each team, its **name**, its **players**, its team **captain** (one of its players), and the **colors of its uniform**.
 - For each player, his/her **name**.
 - For each fan, his/her **name**, **favorite teams**, **favorite players**, and **favorite color**.

Exercise 2 solution (Variant I)



Exercise 2 solution (Variant II)



Exercise 2A. Teams and fans database

- Modification A:

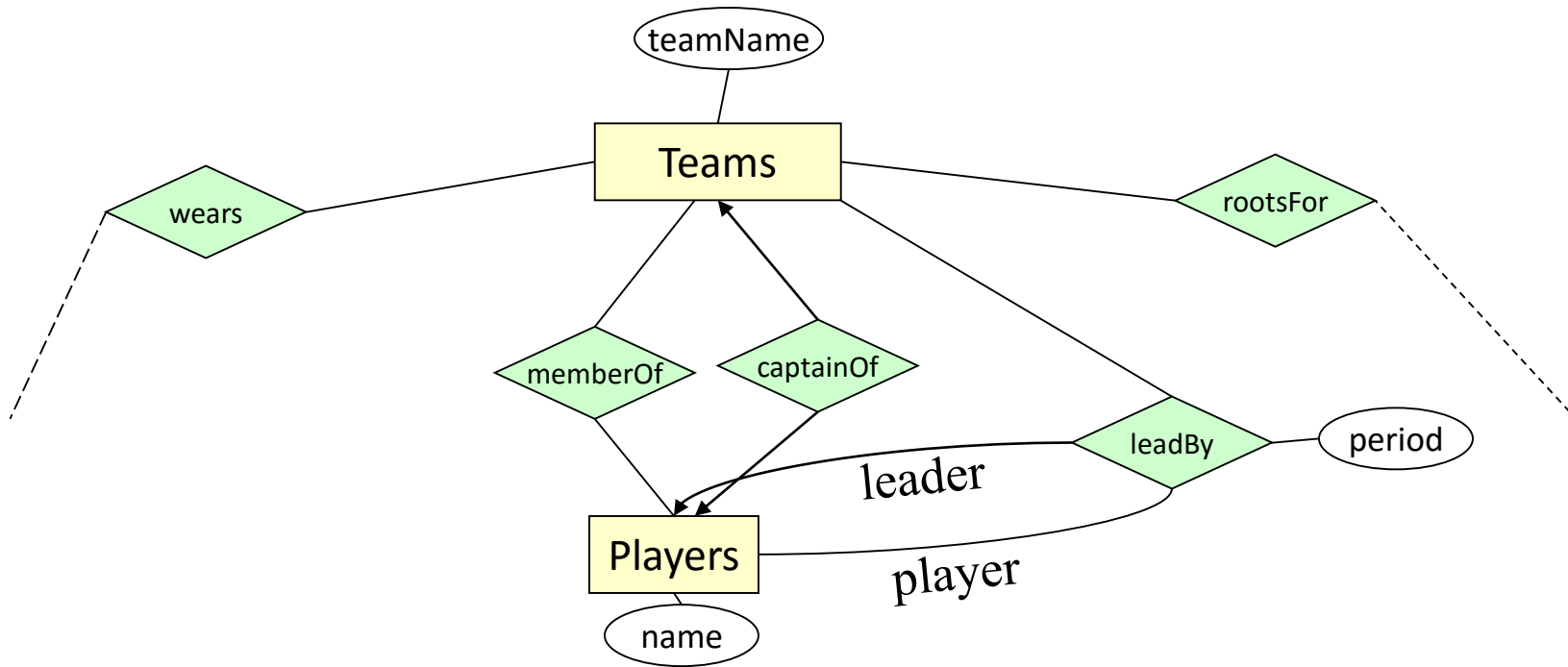
Suppose we wish to add to the schema a relationship “Led-by” among two players and a team. The intention is that this relationship set consists of triples

(player1, player2, team)

such that player 1 played on the team at a time when some other player 2 was the team captain.

Draw the modification to the E/R diagram.

Exercise 2A solution



Exercise 2B. Teams and Fans

- Record for each player the history of teams on which they have played, including the start date and ending date (if they were traded) for each such team.

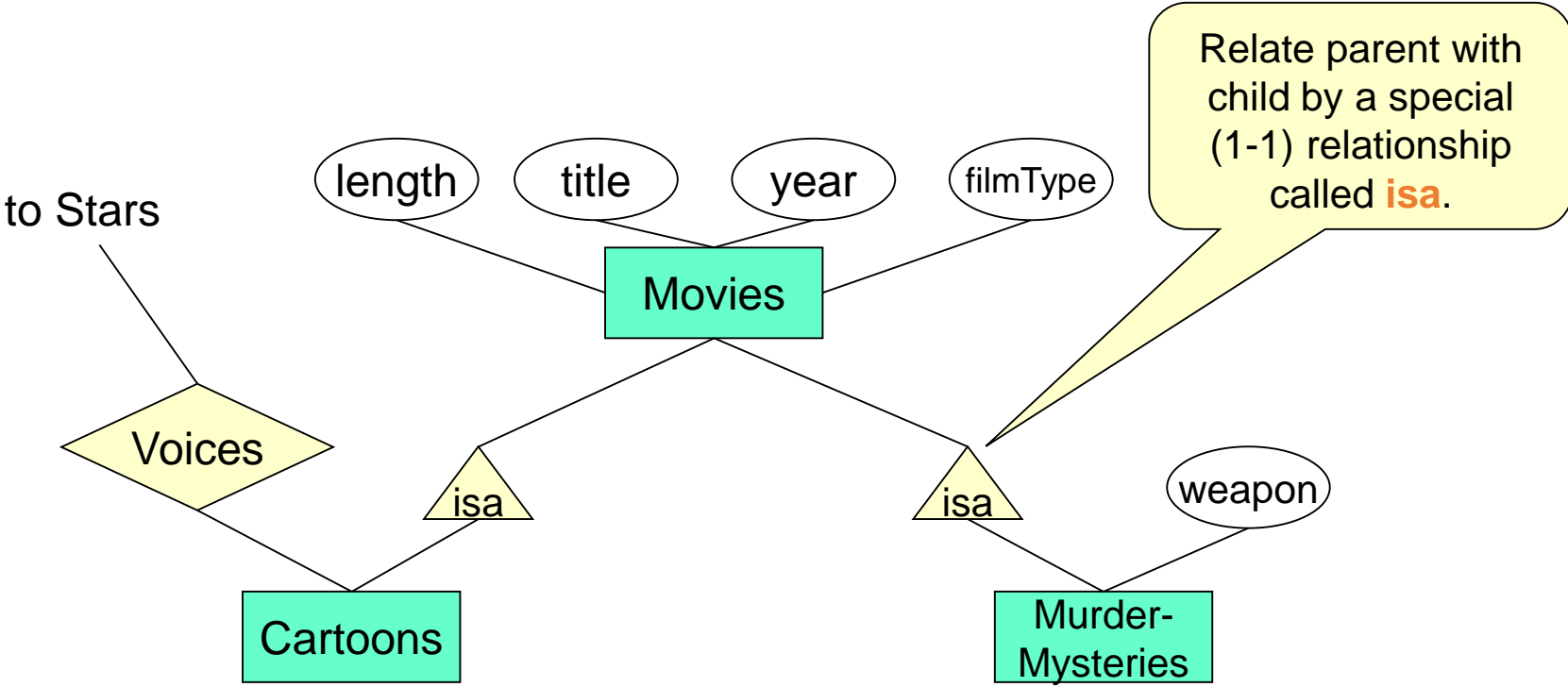
Inheritance

in the ER model

Subclasses

- Sometimes, an entity set contains certain entities that have special properties not associated with all members of this entity set.
- In this case it is useful to define **special-case entity sets**, or *subclasses*, each with its own attributes and relationships

Subclasses

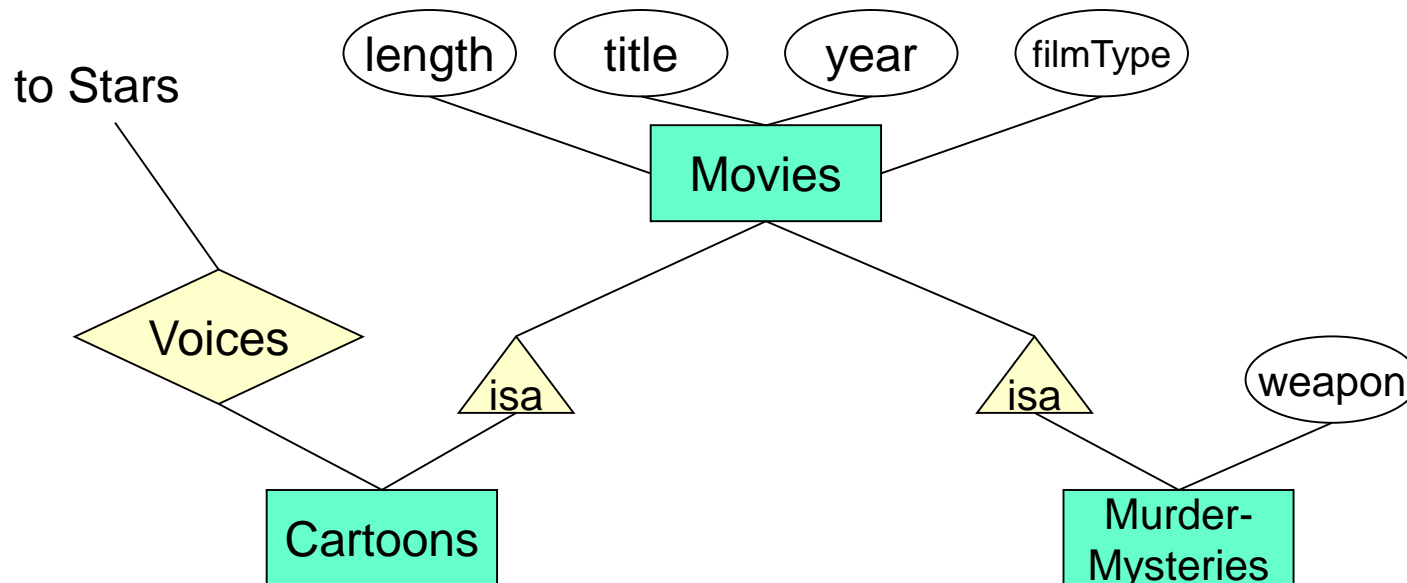


Inheritance in the E/R Model

- In the object-oriented world, property values are in one class only.
 - Subclasses inherit the property definition from superclasses.
- In contrast, E/R entities participate in all subclasses to which they belong.

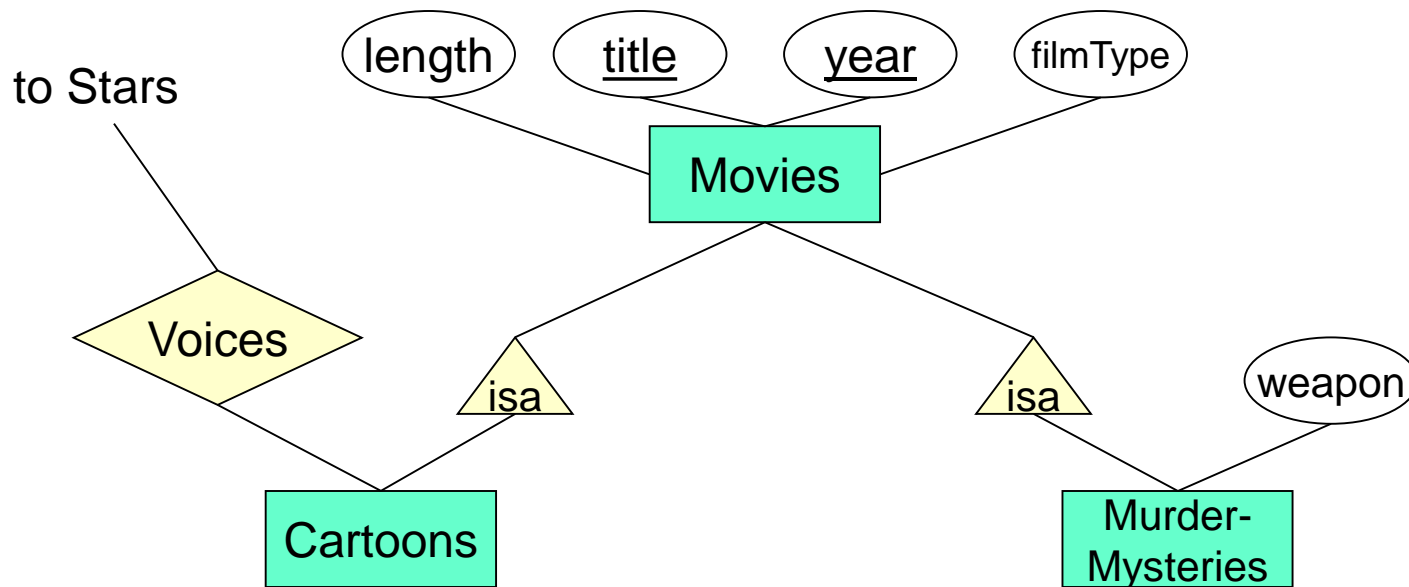
Example

- **Roger Rabbit**, which is both a **cartoon** and **murder-mystery**
 - will have one tuple in each of all three entity sets: **Movies**, **Cartoons**, and **Murder-Mysteries**.
 - i.e. it will have all four attributes of **Movies**, the attribute **weapon**, and finally will participate in the relationship **voices**.



Keys for entity set hierarchies

In **entity set hierarchies** the key at root is **key** for all.
{title,year} is the key for **Movies**, **Cartoons** and **Murder-Mysteries**.



Exercise 3

- Suppose we wish to keep a genealogy. We shall have one entity set, *Person*. The information we wish to record about persons includes their name (an attribute) and the following relationships: mother, father, and children. Give an E/R diagram involving the *Person* entity set and all the relationships in which it is involved. Include relationships for mother, father, and children.
- Modify your “people” database design to include the following special types of people:
 - a) Females.
 - b) Males.
 - c) People who are parents.

Weak entity sets

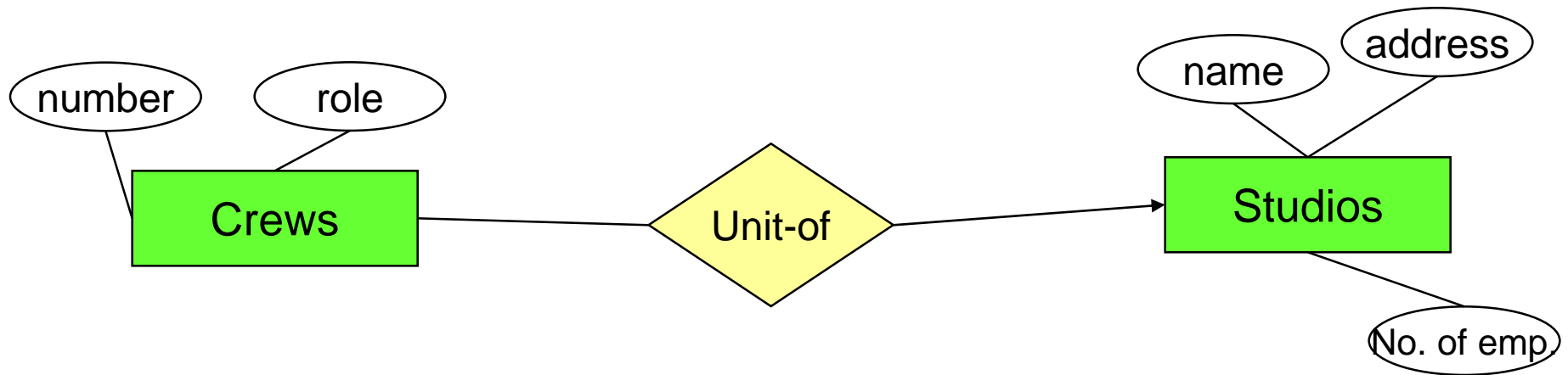
Weak entity sets

- It is possible that the key of an entity set is composed of attributes, some or all of which do not belong to this entity set
- Such an entity set is called a ***weak entity set***
- We use weak entity sets to identify **sub-units** of the main entity, rather than sub-classes

Supporting relationships

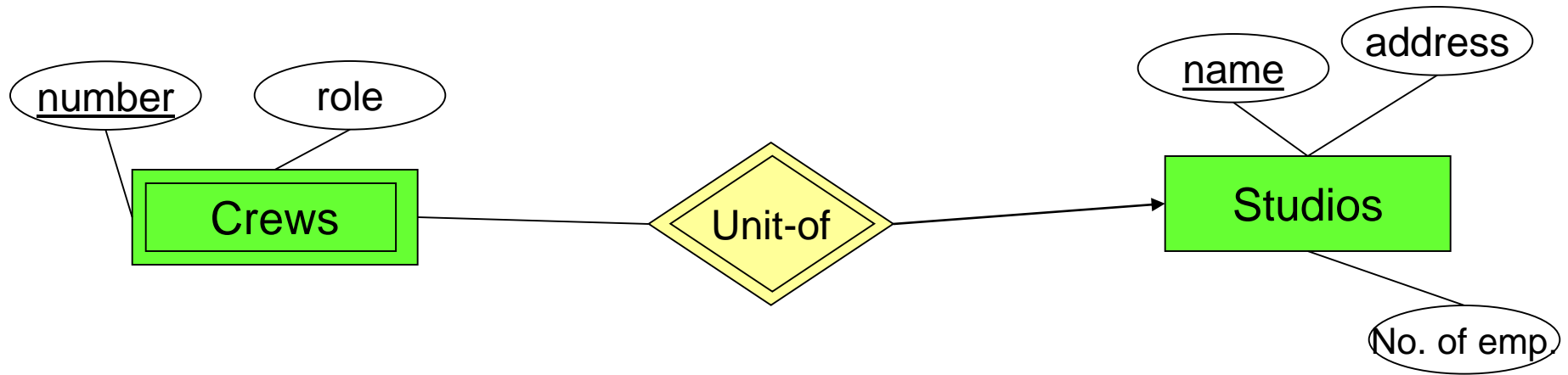
- In a weak entity set E the key consists of:
 - Zero or more its own attributes
 - Keys from other entities reached by many-one relationship from E
- These relationships are called **supporting relationships**

Example of a weak entity set



- E.g. “Crew 1, Special Effects” for Paramount, “Crew 1, Special Effects” for Fox,
- Need to add the key for **Studios**, in order to have a key for **Crews**.
- **Crews** is a weak entity set.

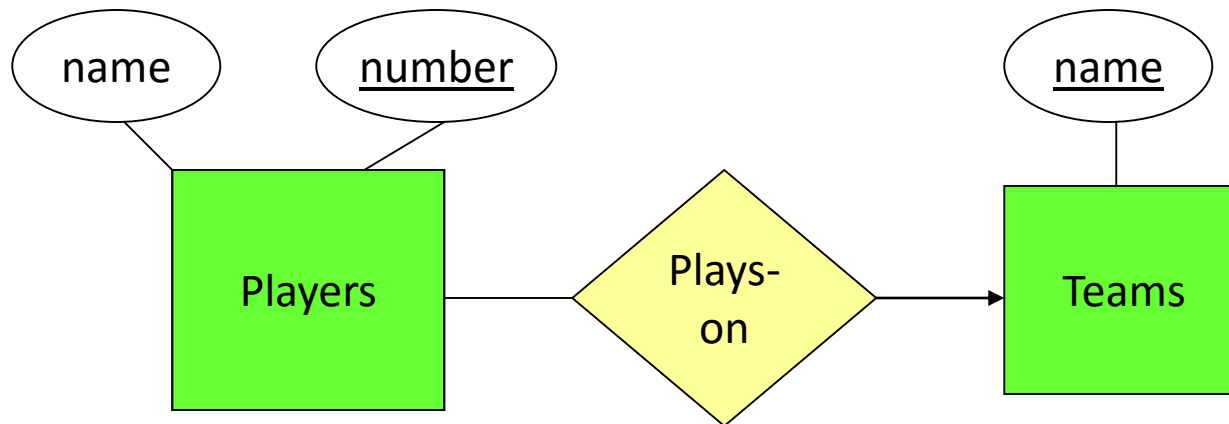
In E/R diagrams:



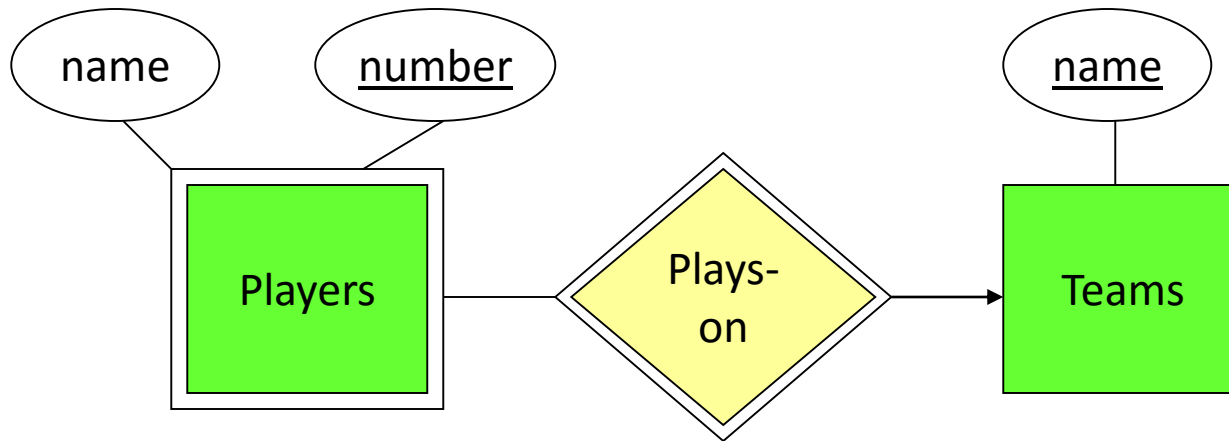
- Double rectangle for the weak entity set.
- Double diamond for a *supporting* many-one relationship.

Another Example – Football Players

- **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.

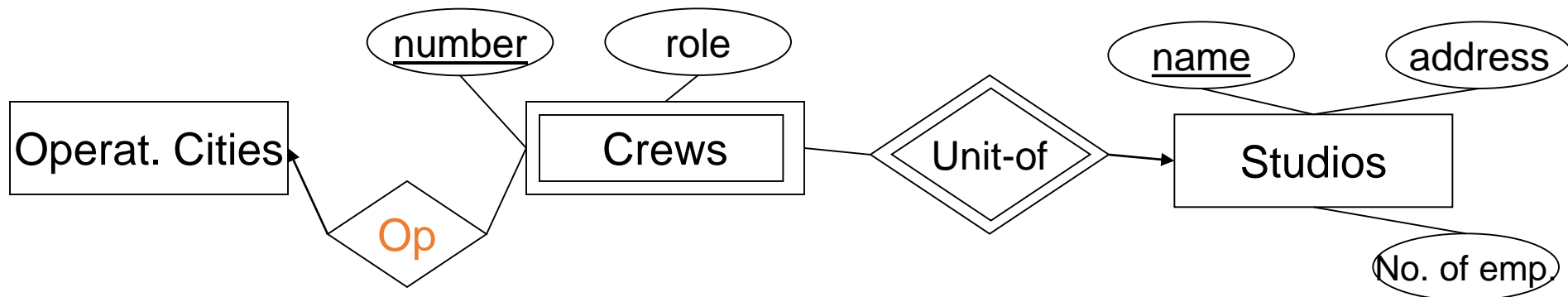


Another Example – Football Players



Supporting vs. regular relationships

- **Not all** the many-one relationships connecting a weak entity set to other entity sets are supporting relationships. E.g.



When Do We Need Weak Entity Sets?

- Usual reason: no global authority capable of creating unique ID's (surrogate key).
- **E.g.:** Unlikely there could be an agreement to assign unique player numbers across all football teams in the world.

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
- It is usually better to create unique or use existing IDs
 - Social insurance number, automobile VIN, etc.
 - Employee ID

Summary

- Limitations of the ER Model:
 - A lot of data semantics can be captured but some cannot
- Key to successful model: **parsimony**
 - As complex as necessary, but no more
 - Choose to represent only “relevant” things