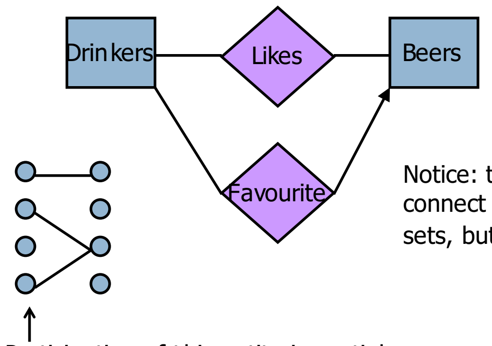
## Participation Constraints

- 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

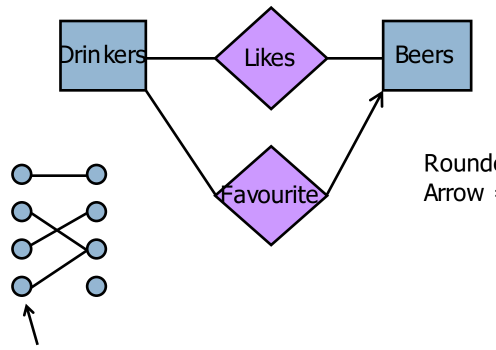
## Example: Many-One Relationship



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

Participation of this entity is partial

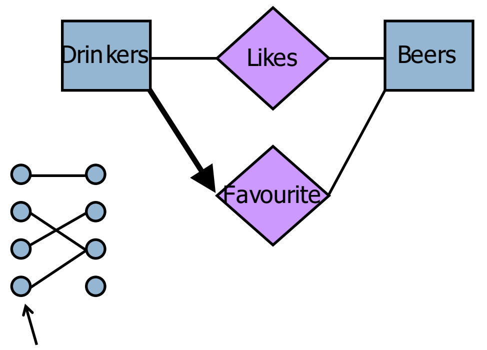
## Example: Many-One Relationship



Rounded (open) Arrow = total function

Participation of this entity is total

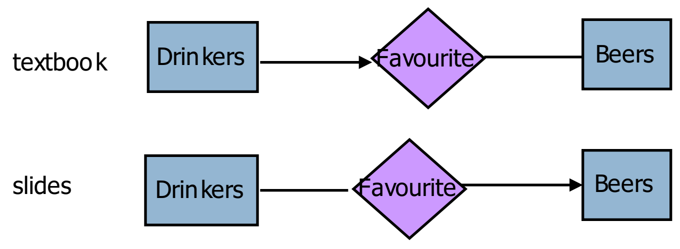
## Alternative (Textbook) Notation



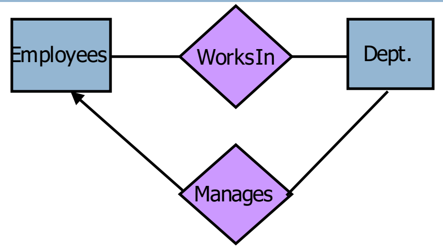
Participation of this entity is total

## Notation

- Be consistent with your chosen notation!



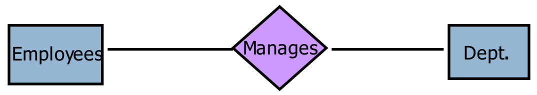
## Key Constraints

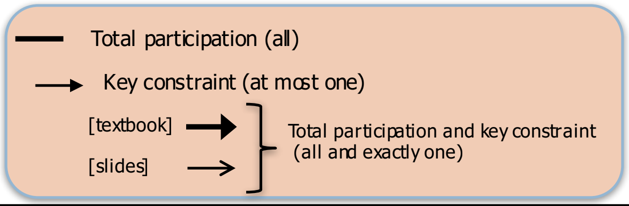


- Many-many: "An employee can work in many depts, and a dept. can have many employees
- One-many: A dept has at most one manager, and employees can manage many departments

## 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)



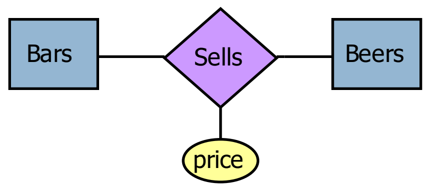

Total participation (all)
Key constraint (at most one)
[textbook]
[slides]
Total participation and key constraint (all and exactly one)

## 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

34



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"

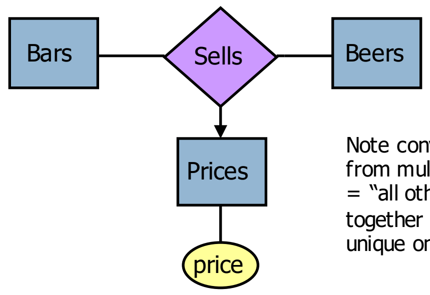
## Equivalent Diagrams Without Attributes on Relationships

35

- Create an entity set representing values of the attribute.
- Make that entity set participate in the relationship.

## Example: Removing an Attribute from a Relationship

36



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

## Roles

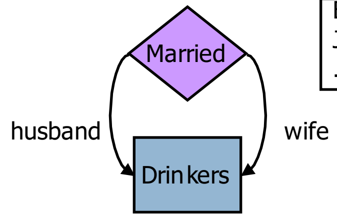
37

- 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

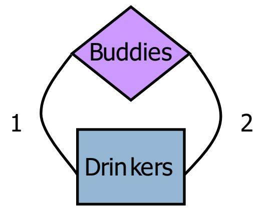
| Husband | Wife |
|---|---|
| Bob | Ann |
| Joe | Sue |
| … | … |



## Example: Roles

Relationship Set

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



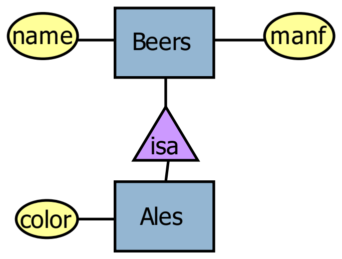
## Subclasses

- *Subclass* = special case = 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

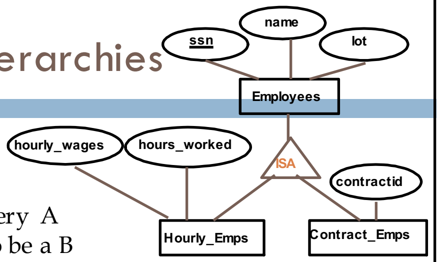
- isa triangles indicate the subclass relationship.
  - Point to the superclass.
- Reasons for using isa:
  - To add descriptive attributes specific to a subclass.
  - To identify entities that participate in a relationship.

## 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.

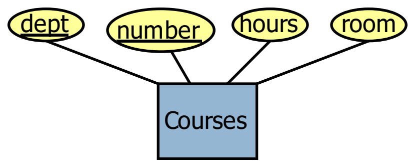
- *Overlap constraints*: Can two sub-classes contain the same entity? E.g., Can Joe be an Hourly_Emps as well as a Contract_Emps entity?
- *Covering constraints*: Does every Employees entity have to be an Hourly_Emps or a Contract_Emps entity?

R. Ramakrishnan & J. Gehrke

## 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

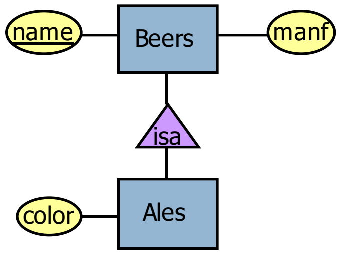


- Note that hours and room could also serve as a key, but we must select only one primary key.

## Keys

46

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

47

- 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.

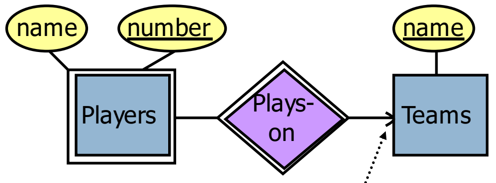
## Example: Weak Entity Set

48

- 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

49



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

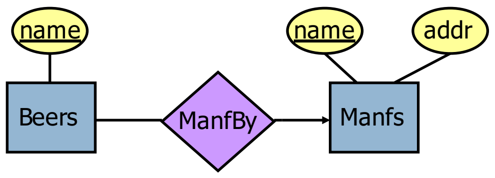
1. Avoid redundancy.
2. Limit the use of weak entity sets.
3. Don't use an entity set when an attribute will do.

## 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

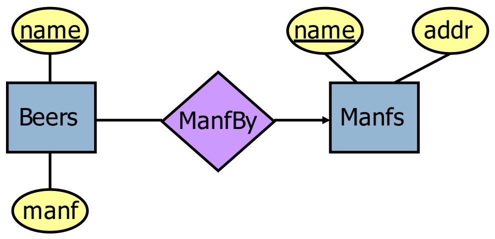
54



This design gives the address of each manufacturer exactly once.

## Example: Bad

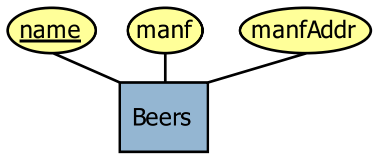
55



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

## Example: Bad

56



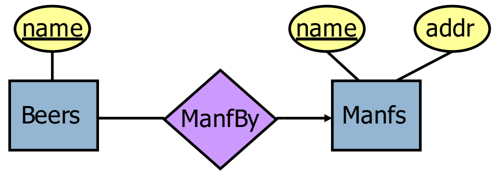
This design repeats the manufacturer's address once for each beer and loses the address if there are temporarily no beers for a manufacturer.

## Entity Sets Versus Attributes

57

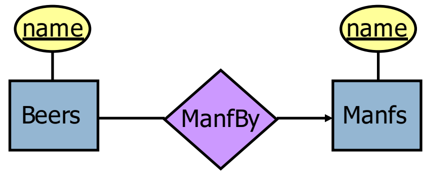
- An entity set should satisfy at least one of the following conditions:
  - It is more than the name of something; it has at least one non-key attribute. OR
  - It is the "many" in a many-one or many-many relationship.
- Depends on the application requirements:
  - If we have several addresses per employee,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



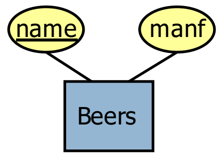
- Manfs deserves to be an entity set because of the nonkey attribute addr.
- Beers deserves to be an entity set because it is the "many" of the many-one relationship ManfBy.

## 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.