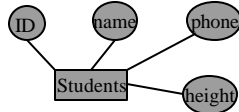


### Entity/Relationship Model

Diagrams to represent designs.

- *Entity* like object, = “thing.”
- *Entity set* like class = set of “similar” entities/objects.
- *Attribute* = property of entities in an entity set, similar to fields of a struct.
- In diagrams, entity set → rectangle; attribute → oval.



1-1

### Relationship Set

Think of the “value” of a relationship set as a table.

- One column for each of the connected entity sets.
- One row for each list of entities, one from each set, that are connected by the relationship.

<u>Students</u>	<u>Courses</u>
Sally	CS180
Sally	CS111
Joe	CS180
...	...

1-3

### Relationships

- Connect two or more entity sets.
- Represented by diamonds.

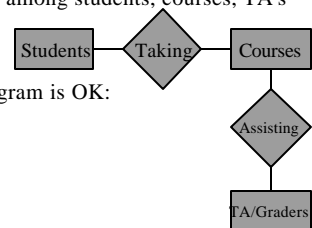


1-2

### Multiway Relationships

Usually binary relationships (connecting two E.S.) suffice.

- However, there are some cases where three or more E.S. must be connected by one relationship.
- Example: relationship among students, courses, TA's (and graders).



Possibly, this E/R diagram is OK:

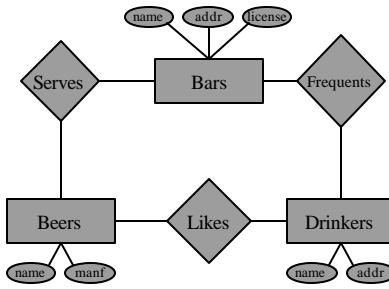
1-4

- Works if each TA (or grader) is a TA of all students. Connection student-TA is *only* via the course.
- But what if students were divided into sections, each headed by a TA?
  - ◆ Then, a student in CSC343 is related to only one of the TA's for the course. Which one?
- Need a 3-way relationship to tell.

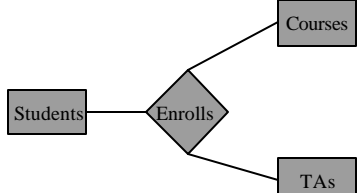
1-5

### Beers-Bars-Drinkers Example

- The drinkers-bars-beers example.



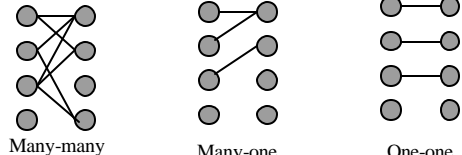
1-7



Students	Courses	TAs
Ann	CS343	Miles
Sue	CS343	Edward
Bob	CS343	Vladimir
...	...	...

1-6

### Multiplicity of Relationships

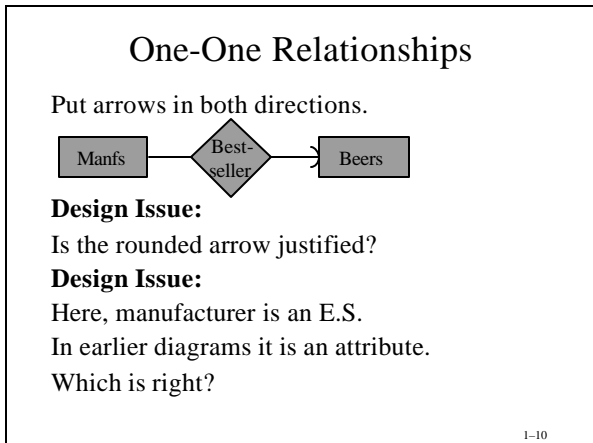
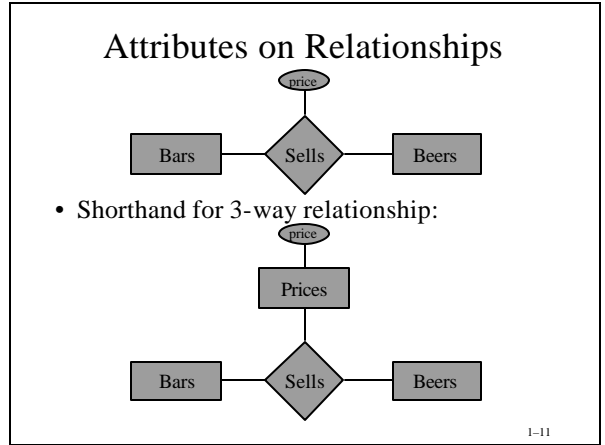
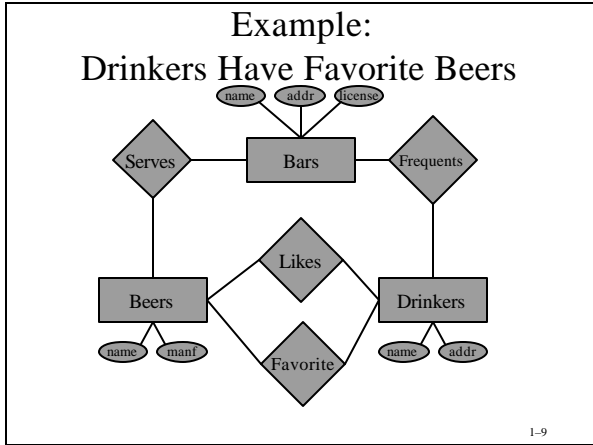


Many-many                  Many-one                  One-one

**Representation of Many-One**

- E/R: arrow pointing to “one.”
- ◆ Rounded arrow = “exactly one.”

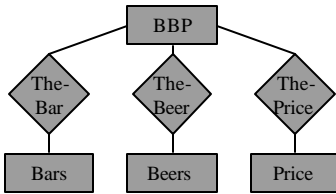
1-8



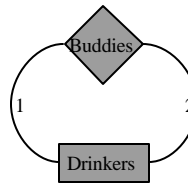
- A true 3-way relationship.
    - ◆ Price depends jointly on beer and bar.
  - Notice arrow convention for multiway relationships: “all other E.S. determine one of these.”
    - ◆ Not sufficiently general to express any possibility.
    - ◆ However, if price, say, depended only on the beer, then we could use two 2-way relationships: price-beer and beer-bar.
    - ◆ Or better: just make price an attribute of beer.
- 1-12

### Converting Multiway to 2-Way

- Baroque in E/R, but necessary in certain “object-oriented” models.
- Create a new connecting E.S. to represent rows of a relationship set.
  - ◆ E.g., (Joe's Bar, Bud, \$2.50) for the *Sells* relationship.
- Many-one relationships from the connecting E.S. to the others.



1-13



<u>Buddy1</u>	<u>Buddy2</u>
$d_1$	$d_2$
$d_1$	$d_3$
$d_2$	$d_1$
$d_2$	$d_4$
...	...

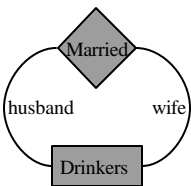
- Notice *Buddies* is symmetric, Married not.
  - ◆ No way to say “symmetric” in E/R.

1-15

### Roles

Sometimes an E.S. participates more than once in a relationship.

- Label edges with *roles* to distinguish.



<u>Husband</u>	<u>Wife</u>
$d_1$	$d_2$
$d_3$	$d_4$
...	...

1-14

### More Design Issues

1. Subclasses.
2. Keys.
3. Weak entity sets.

1-16

### Subclasses

Subclass = special case = fewer entities = more properties.

- Example: Ales are a kind of beer. In addition to the *properties* (= attributes and relationships) of beers, there is a “color” attribute for ales.

1-17

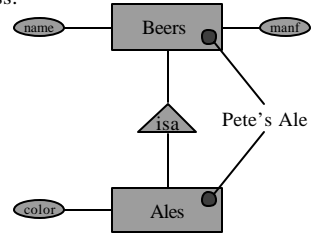
### Different Subclass Viewpoints

1. *E/R viewpoint*: An entity has a *component* in each entity set to which it logically belongs.

- ◆ Its properties are the union of the properties of these E.S.

2. Contrasts with *object-oriented viewpoint*: An object (entity) belongs to exactly one class.

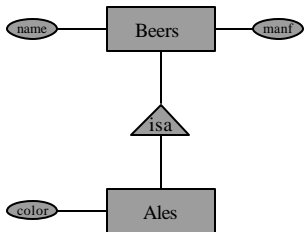
- ◆ It *inherits* properties of its superclasses.



1-19

### E/R Subclasses

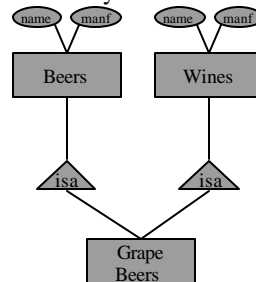
- Assume subclasses form a tree (no multiple inheritance).
- *isa* triangles indicate the subclass relation.



1-18

### Multiple Inheritance

Theoretically, an E.S. could be a subclass of several other entity sets.



1-20

## Problems

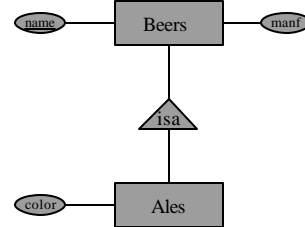
How should conflicts be resolved?

- Example: *manf* means vintner for wines, bottler for beers. What does *manf* mean for “grape beers”?
- Need ad-hoc notation to resolve meanings.
- In practice, we shall assume a tree of entity sets connected by *isa*, with all “isas” pointing from child to parent.

1-21

## Example

- Suppose name is key for *Beers*.



- Beer name is also key for ales.
  - ◆ In general, key at root is key for all.

1-23

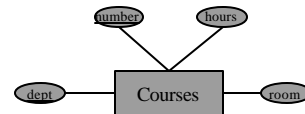
## Keys

A *key* is a set of attributes whose values can belong to at most one entity.

- In E/R model, every E.S. must have a key.
  - ◆ It could have more than one key, but one set of attributes is the “designated” key.
- In E/R diagrams, you should underline all attributes of the designated key.

1-22

## Example: A Multiattribute Key



- Possibly, the combination of hours + room also forms a key.

1-24

### Weak Entity Sets

Sometimes an E.S. *E*'s key comes not (completely) from its own attributes, but from the keys of one or more E.S.'s to which *E* is linked by a *supporting* many-one relationship.

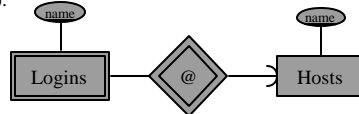
- Called a *weak* E.S.
- Represented by putting double rectangle around *E* and a double diamond around each supporting relationship.
- Many-one-ness of supporting relationship (includes 1-1) essential.
  - ◆ With many-many, we wouldn't know which entity provided the key value.
- "Exactly one" also essential, or else we might not be able to extract key attributes by following the supporting relationship.

1-25

### Example: Logins (Email Addresses)

Login name = user name + host name, e.g., ark@soe.ucsc.edu.

- A "login" entity corresponds to a user name on a particular host, but the passwd table doesn't record the host, just the user name, e.g., ark.
- Key for a login = the user name at the host (which is unique for that host only) + the IP address of the host (which is unique globally).

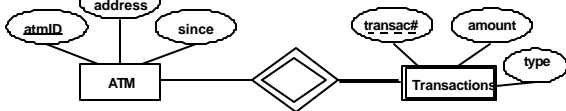


- Design issue: Under what circumstances could we simply make login-name and host-name be attributes of logins, and dispense with the weak E.S.?

1-27

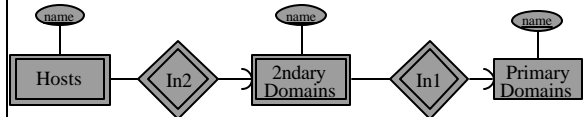
### Weak Entity Sets (cont.)

- A *weak entity* can be identified uniquely only by considering the primary key of another (*owner*) entity.
  - ◆ Owner entity set and weak entity set must participate in a one-to-many relationship set (one owner, many weak entities).
  - ◆ Weak entity sets must have total participation in this *identifying* relationship set.
  - ◆ transac# is a **discriminator** within a group of transactions in an ATM.



### Example: Chain of "Weakness"

Consider IP addresses consisting of a primary domain (e.g., edu), subdomain (e.g., ucsc), and host (e.g., soe).



- Key for primary domain = its name.
- Key for secondary domain = its name + name of primary domain.
- Key for host = its name + key of secondary domain = its name + name of secondary domain + name of primary domain.

1-28

### All “Connecting” Entity Sets Are Weak

- In this special case, where bar and beer determine a price, we can omit price from the key, and remove the double diamond from ThePrice.
- Better: price is attribute of BBB.

1-29

### Example

Bad: repeats manufacturer address for each beer they manufacture.

Bad: manufacturer’s name said twice.

1-31

### Design Principles

Setting: client has (possibly vague) idea of what he/she wants. You must design a database that represents these thoughts and only these thoughts.

#### Avoid redundancy

= saying the same thing more than once.

- Wastes space and encourages inconsistency.

#### Example

Good:

1-30

### Use Schema to Enforce Constraints

- The design *schema* should enforce as many constraints as possible.
  - ◆Don't rely on future data to follow assumptions.

#### Example

- If registrar wants to associate only one instructor with a course, don't allow sets of instructors and count on departments to enter only one instructor per course.

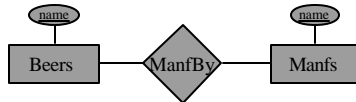
1-32

## Entity Sets Vs. Attributes

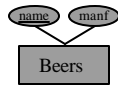
You may be unsure which concepts are worthy of being entity sets, and which are handled more simply as attributes.

- Especially tricky for the class design project, since there is a temptation to create needless entity sets to make project “larger.”

Wrong:



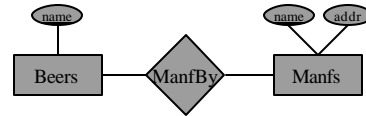
Right:



1-33

## Example

The following design illustrates both points:



- Manfs* deserves to be an E.S. because we record *addr*, a nonkey attribute.
- Beers* deserves to be an E.S. because it is at the “many” end.
  - If not, we would have to make “set of beers” an attribute of *Manfs* – something we avoid doing, although some may tell you it is OK in E/R model.

1-35

## Intuitive Rule for E.S. Vs. Attribute

Make an entity set only if it either:

- Is more than a name of something; *i.e.*, it has nonkey attributes or relationships with a number of different entity sets, or
- Is the “many” in a many-one relationship.

1-34

## Don't Overuse Weak E.S.

- There is a tendency to feel that no E.S. has its entities uniquely determined without following some relationships.
- However, in practice, we almost always create unique ID's to compensate: social-security numbers, VIN's, etc.
- The only times weak E.S.'s seem necessary are when:
  - We can't easily create such ID's; *e.g.*, no one is going to accept a “species ID” as part of the standard nomenclature (species is a weak E.S. supported by membership in a genus).
  - There is no global authority to create them, *e.g.*, crews and studios.

1-36