

Computer Science 180: Database Systems – Slide 1

What is a Database Management System?

1. Manages very large amounts of data.
2. Supports efficient access to very large amounts of data.
3. Supports concurrent access to very large amounts of data.
 - ◆ Example: bank and its ATM machines.
4. Supports secure, atomic access to very large amounts of data.
 - ◆ Contrast two people editing the same UNIX file – last to write “wins” – with the problem if two people deduct money from the same account via ATM machines at the same time – new balance is wrong whichever writes last.

Relational Model

- Based on tables, as:

acct #	name	balance
12345	Sally	1000.21
34567	Sue	285.48
...

- Today used in *most* DBMS's.

The DBMS Marketplace

- Relational DBMS companies – Oracle, Sybase – are among the largest software companies in the world.
- IBM offers its relational DB2 system. With IMS, a nonrelational system, IBM is by some accounts the largest DBMS vendor in the world.
- Microsoft offers SQL-Server, plus Microsoft Access for the cheap DBMS on the desktop, answered by “lite” systems from other competitors.
- Relational companies also challenged by “object-oriented DB” companies.
- But countered with “object-relational” systems, which retain the relational core while allowing type extension as in OO systems.

Three Aspects to Studying DBMS's

1. Modeling and design of databases.

- ◆ Allows exploration of issues before committing to an implementation.

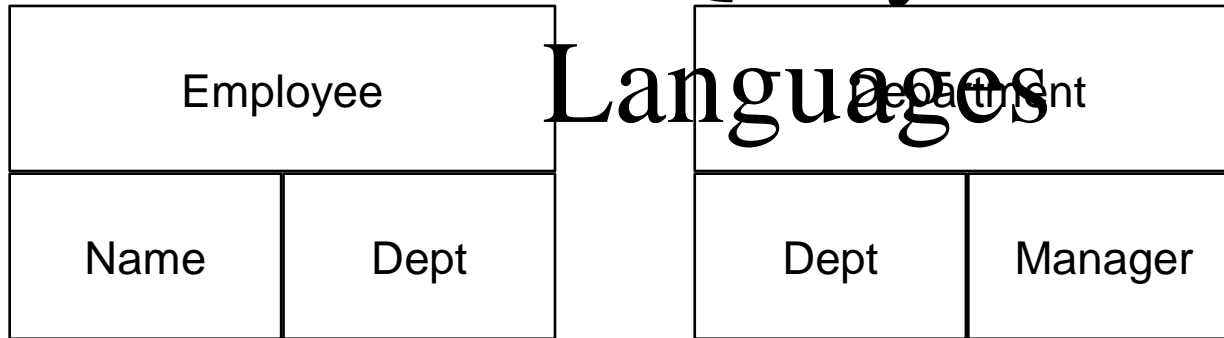
2. Programming: queries and DB operations like update.

- ◆ SQL = “intergalactic dataspeak.”

3. DBMS implementation.

CS180 = (1) + (2), while (3) is covered partly in CS277.

Query



Languages

SQL

```
SELECT Manager
FROM Employee, Department
WHERE Employee.name = "Clark Kent"
      AND Employee.Dept = Department.Dept
```

Query Language

Data definition language (DDL) ~ like type defs in C or Pascal

Data Manipulation Language (DML)

Query (SELECT)

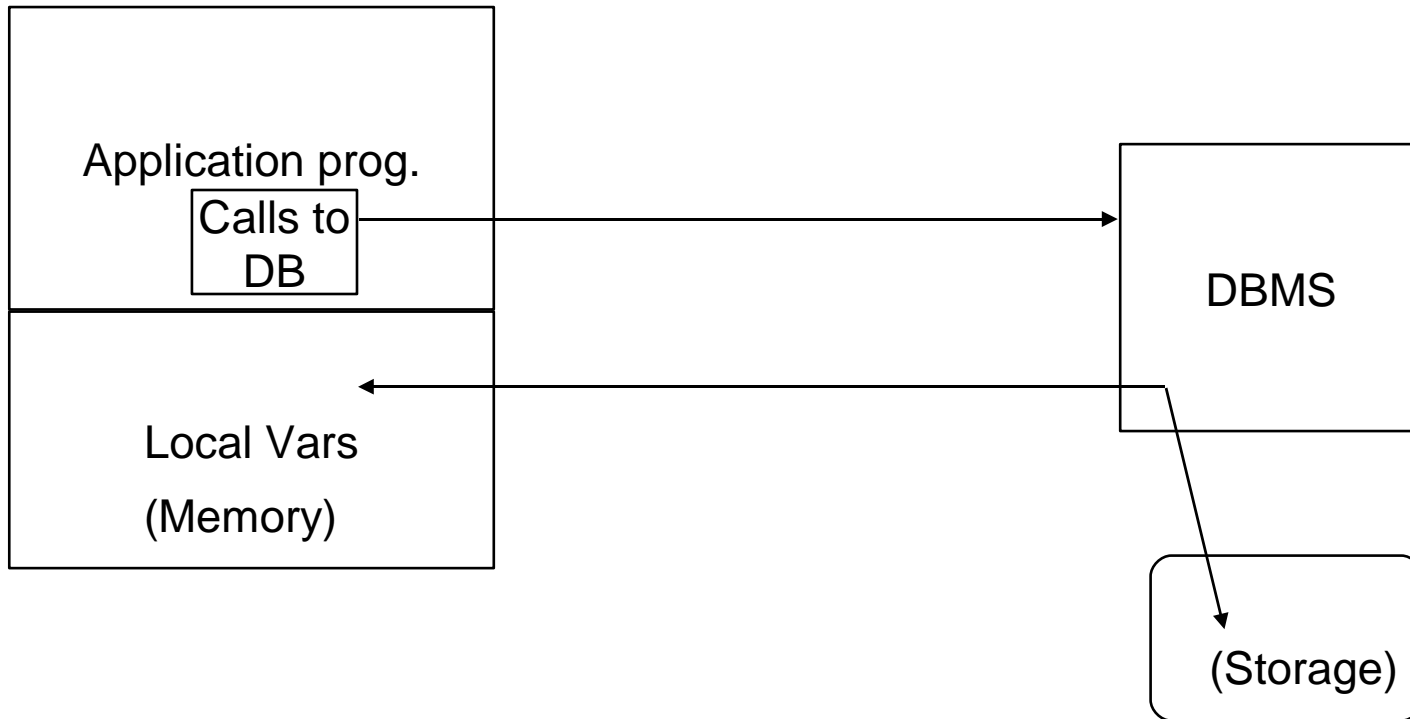
UPDATE <relation name >

SET <attribute> = <new-value>

WHERE <condition>

Host Languages

C, C++, Fortran, Lisp, COBOL



Host language is completely general (Turing complete)
but gives you no support
Query language—less general "non procedural" and
optimizable

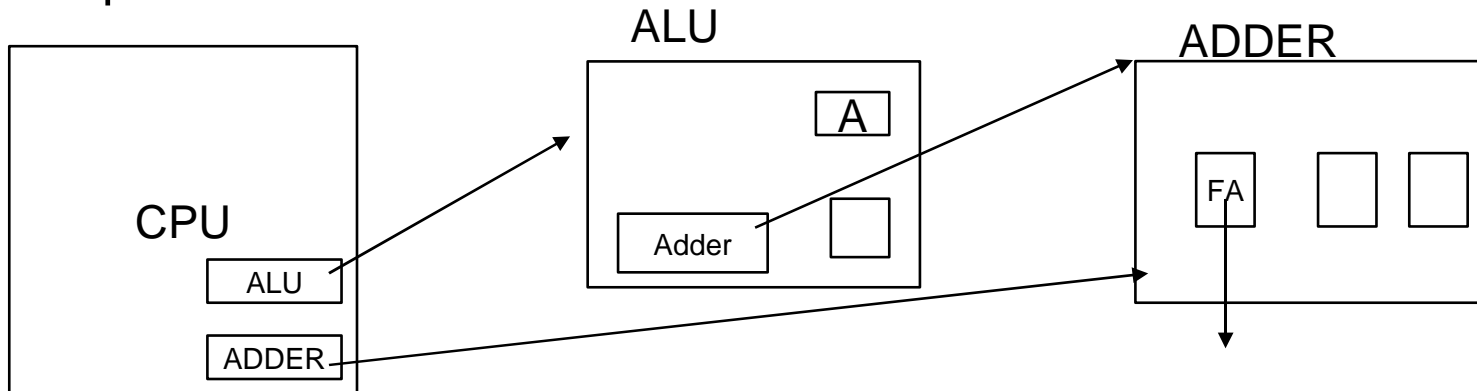
Relational model is good for:

Large amounts of data → simple operations

Navigate among small number of relations

Difficult Applications for relational model:

- VLSI Design (CAD in general)
- CASE
- Graphical Data



Bill of Materials or
transitive closure

Where number of "relations" is large, relationships are complex

- Object Data Model
- Logic Data Model

OBJECT DATA MODEL

1. Complex Objects – Nested Structure (pointers or references)
2. Encapsulation, set of Methods/Access functions
3. Object Identity
4. Inheritance – Defining new classes like old classes

Object model: usually find objects via explicit navigation

Also query language in some systems

LOGIC (Horn Clause) DATA MODEL

- Prolog, Datalog

if A1 and A2 then B

prolog B:- A1 and A2

Functions $s(5) = 6$ (successor)

Predicates with Arguments $\text{sum}(X,Y,Z) \quad X + Y = Z$

$\text{sum}(X,0,X)$ means $X + 0 = X$ (always true for all X)

$\text{sum}(X,s(Y),s(Z)):-\text{sum}(X,Y,Z)$

means $X+(Y+1) = (Z+1)$ if $X + Y = Z$

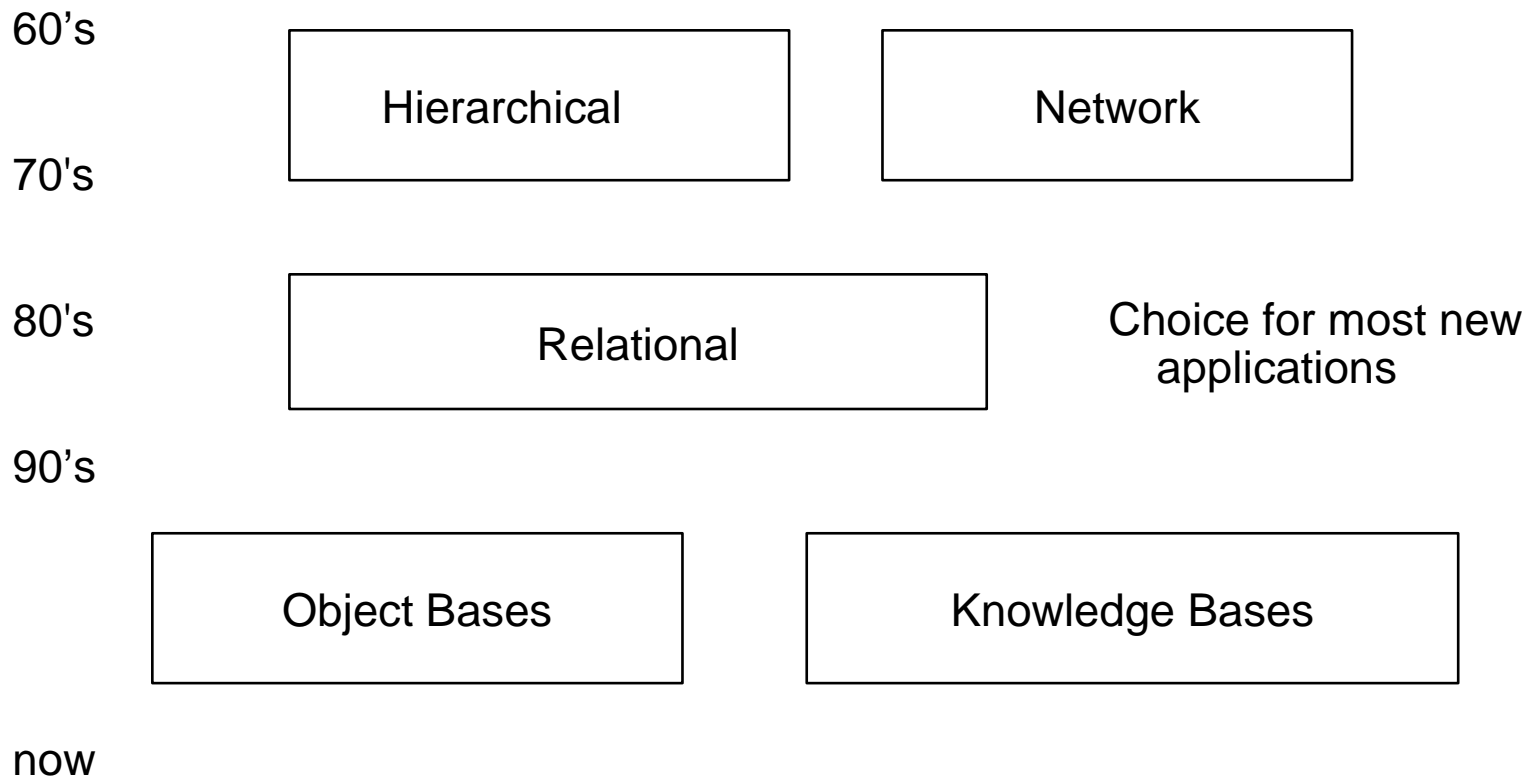
More power than relational

Can Compute Transitive Closure

$\text{edge}(X,Y)$

$\text{path}(X,Y) :- \text{edge}(X,Y)$

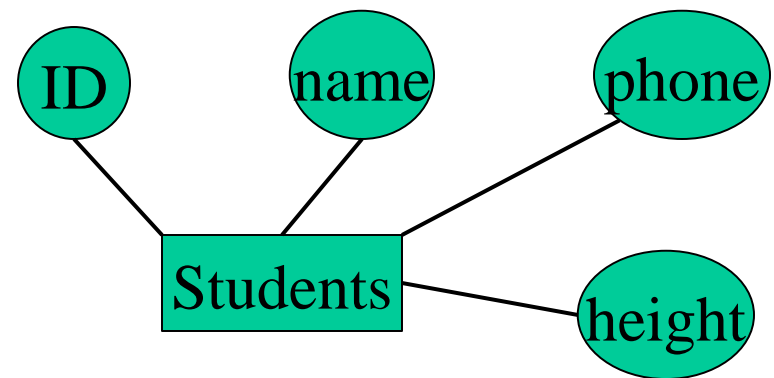
$\text{path}(X,Z) :- \text{path}(X,Y) \ \& \ \text{edge}(Y,Z)$



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.



Relationships

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



Relationship Set

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

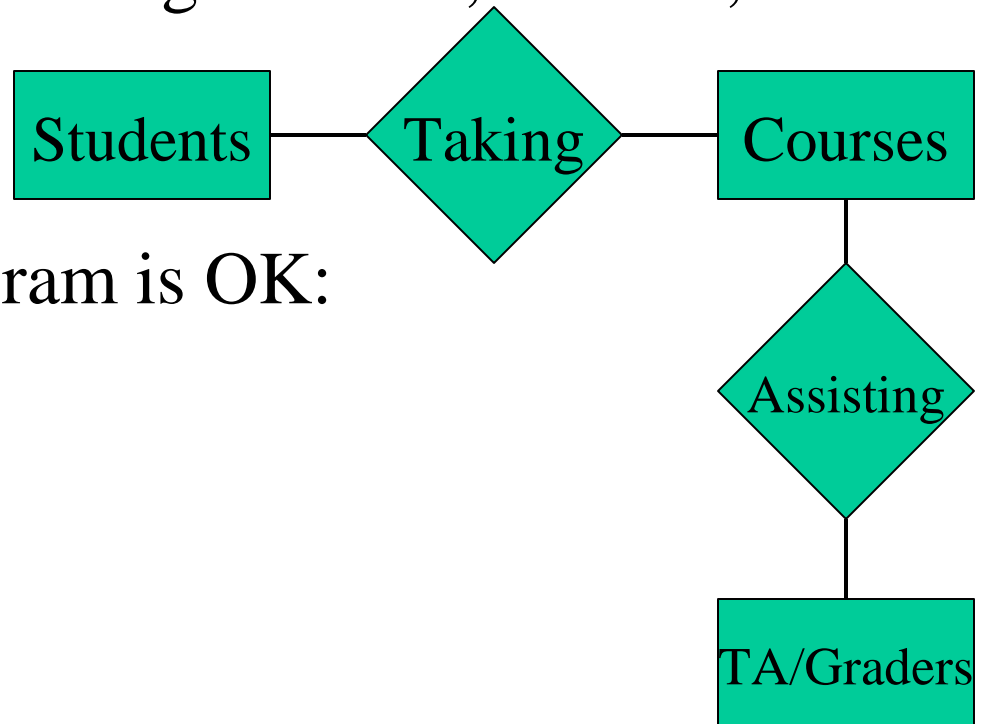
- One column for each of the connected entity sets.
- One row for each list of entities, one from each set, that are connected by the relationship.

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

Multiway Relationships

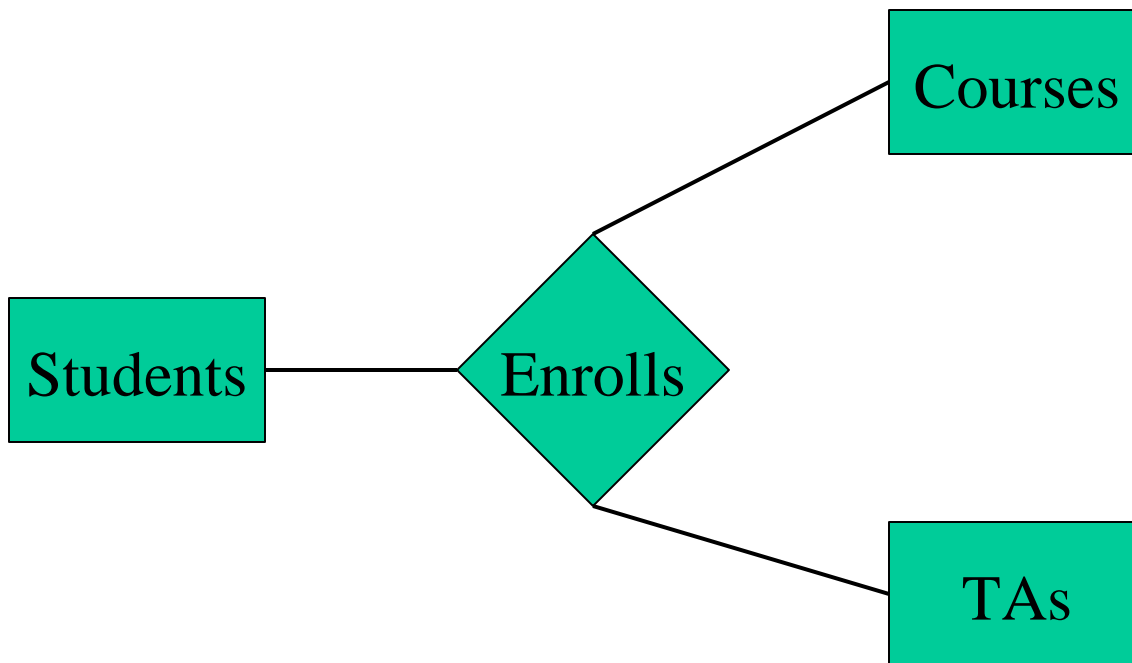
Usually binary relationships (connecting two E.S.) suffice.

- 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:

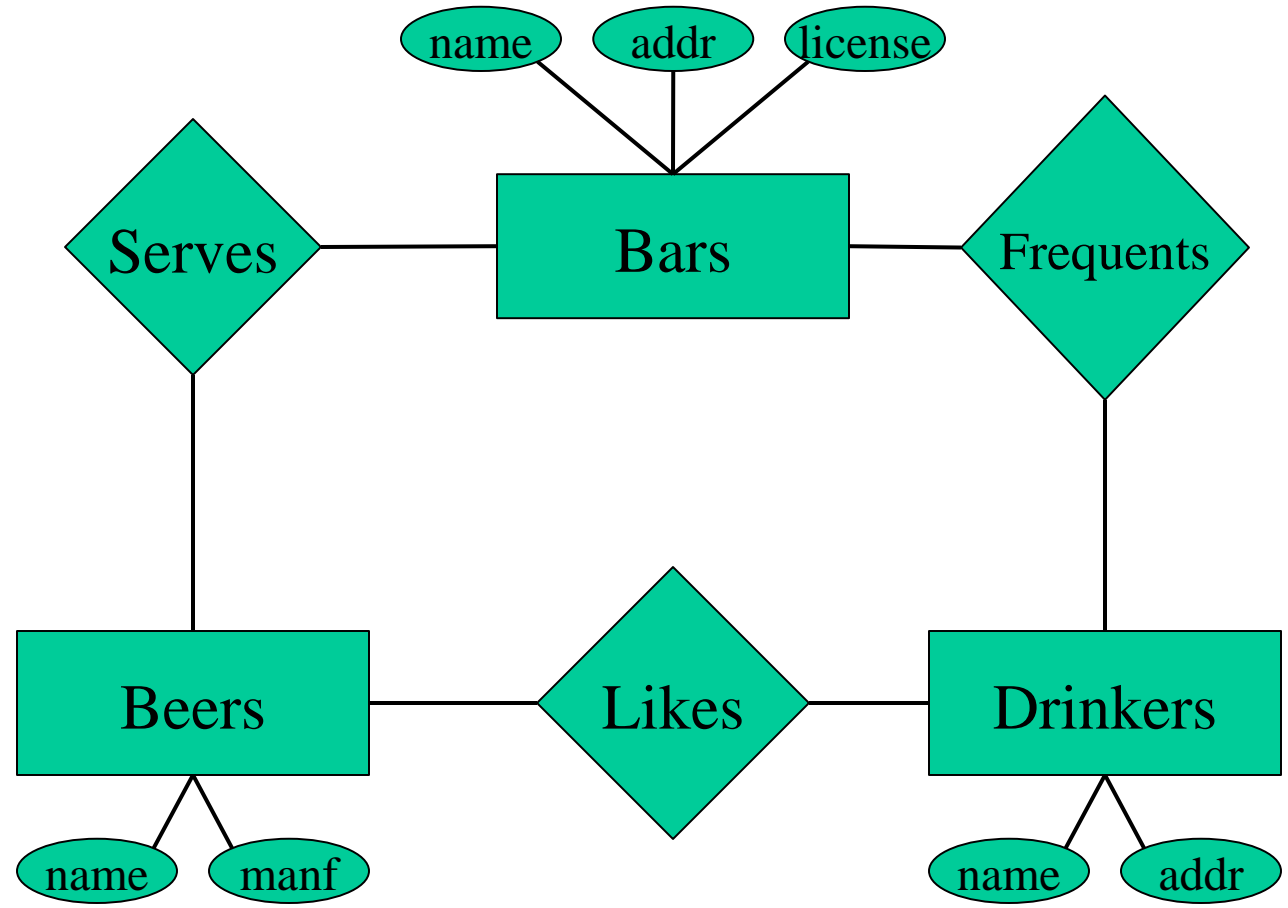
- Works in CS180, because each TA (or grader) is a TA of all students. Connection student-TA is *only* via the course.
- But what if students were divided into sections, each headed by a TA?
 - ◆ Then, a student in CS180 would be related to only one of the TA's for CS180. Which one?
- Need a 3-way relationship to tell.



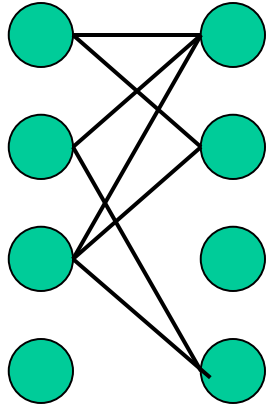
Students	Courses	TAs
Ann	CS180	Jan
Sue	CS180	Pat
Bob	CS180	Jan
...

Beers-Bars-Drinkers Example

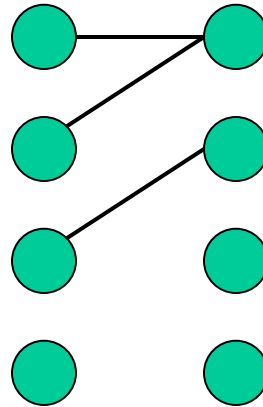
- Our running example for the course.



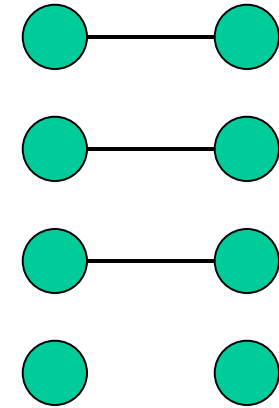
Multiplicity of Relationships



Many-many



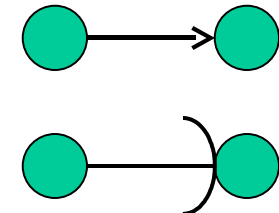
Many-one



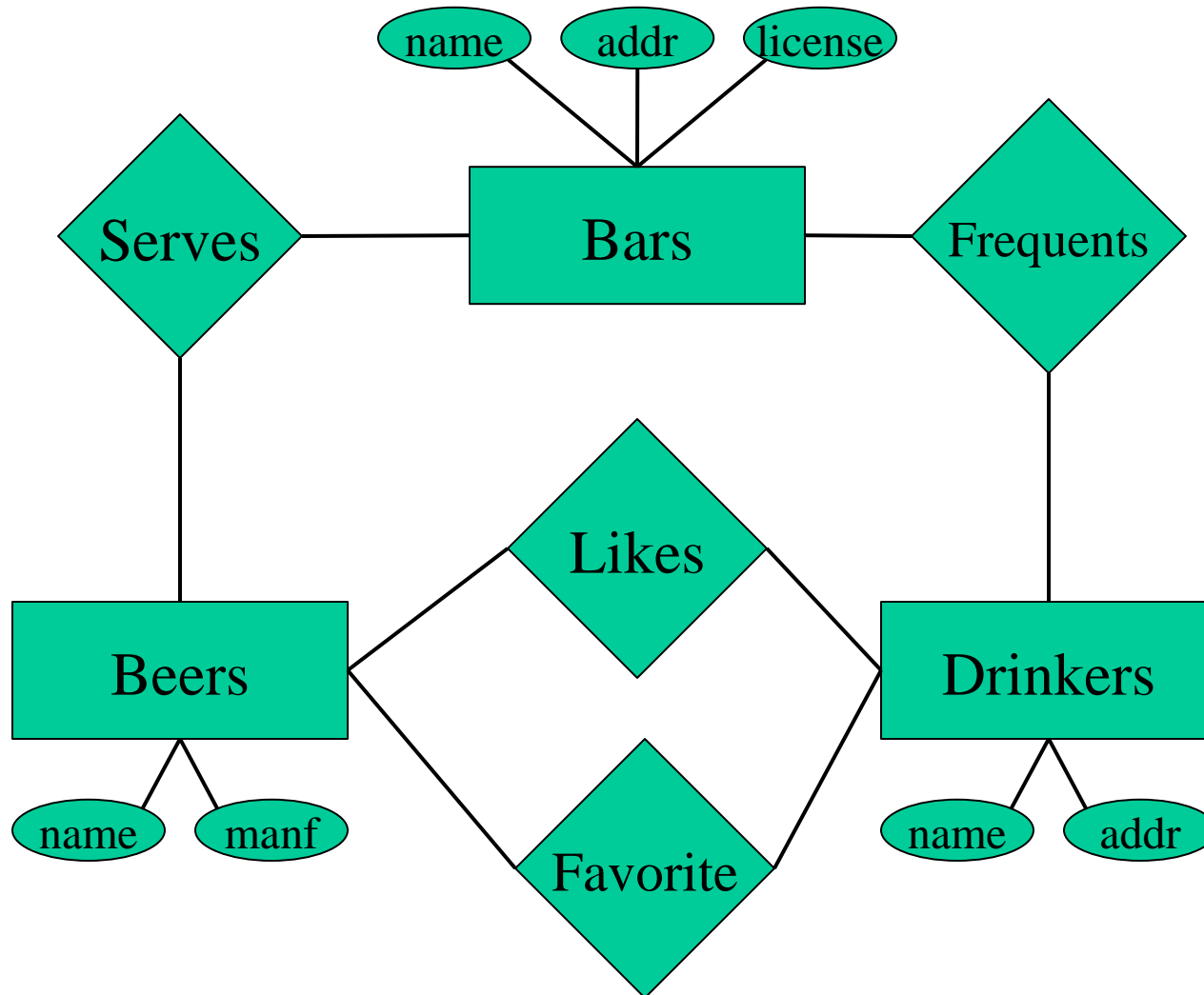
One-one

Representation of Many-One

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



Example: Drinkers Have Favorite Beers



One-One Relationships

Put arrows in both directions.



Design Issue:

Is the rounded arrow justified?

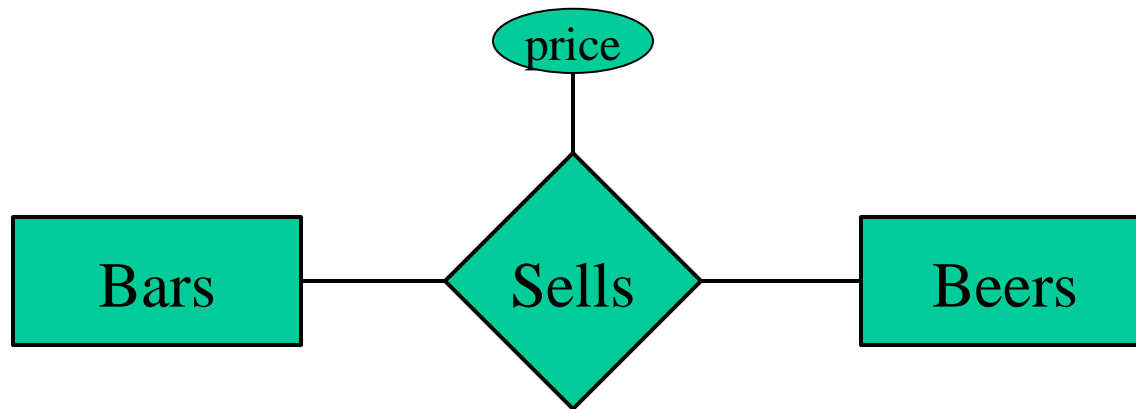
Design Issue:

Here, manufacturer is an E.S.

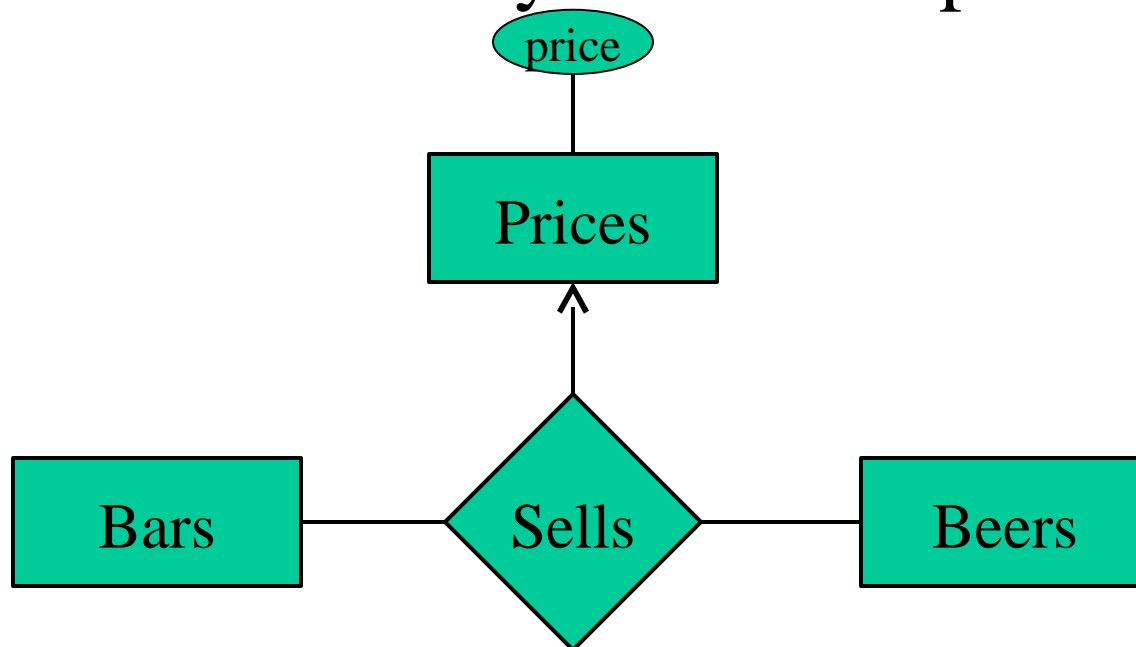
In earlier diagrams it is an attribute.

Which is right?

Attributes on Relationships



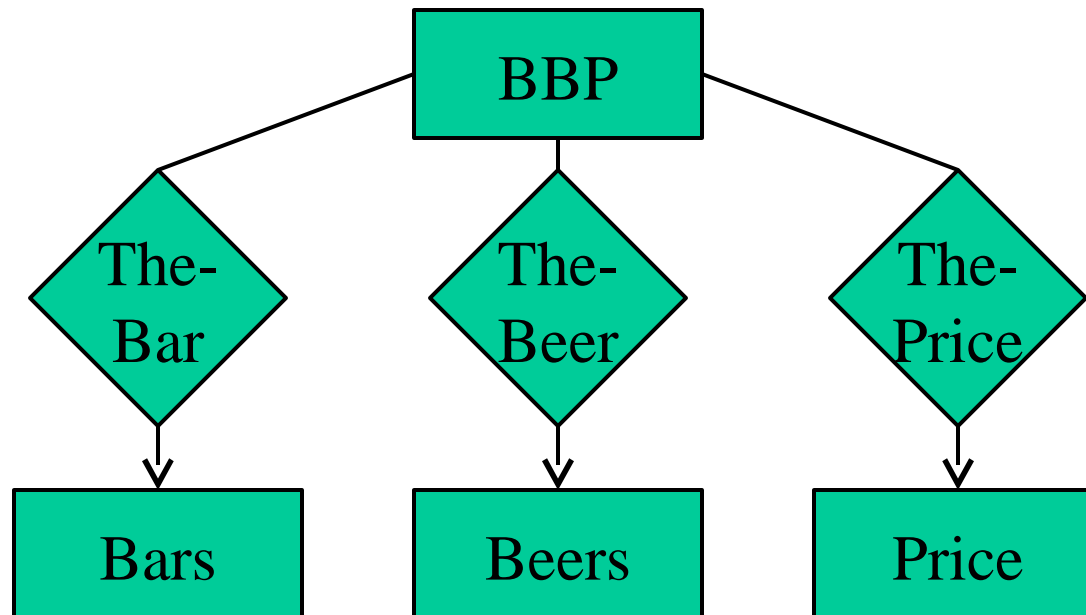
- Shorthand for 3-way relationship:



- 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.

Converting Multiway to 2-Way

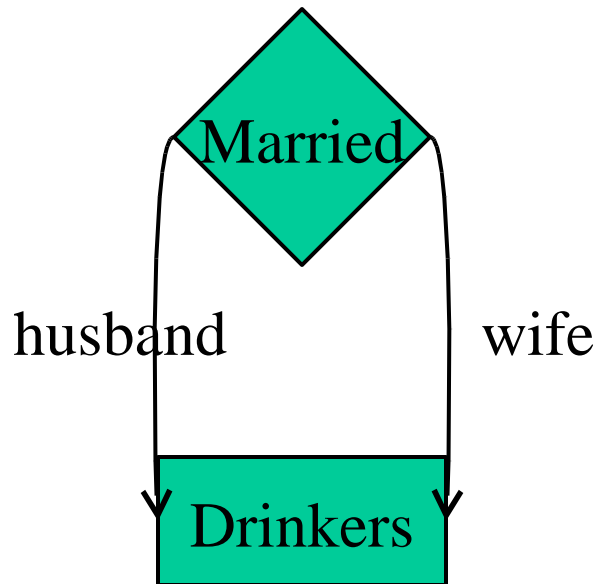
- Baroque in E/R, but necessary in certain “object-oriented” models.
- Create a new connecting E.S. to represent rows of a relationship set.
 - ◆ E.g., (Joe's Bar, Bud, \$2.50) for the *Sells* relationship.
- Many-one relationships from the connecting E.S. to the others.



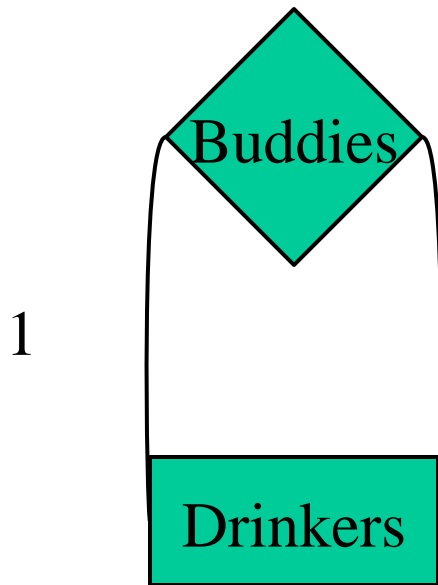
Roles

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

- Label edges with *roles* to distinguish.



Husband	Wife
d_1	d_2
d_3	d_4
...	...



2

Buddy1	Buddy2
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.

Design Question

Should we replace *husband* and *wife* by one relationship *spouse*?

More Design Issues

1. Subclasses.
2. Keys.
3. Weak entity sets. (Next class.)

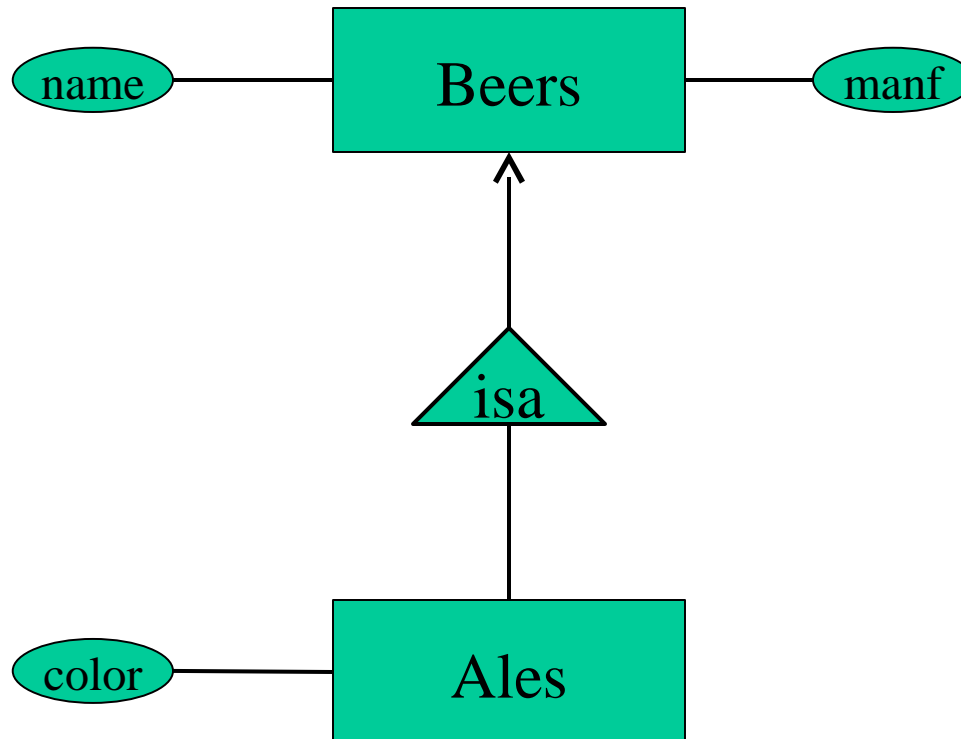
Subclasses

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.

E/R Subclasses

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



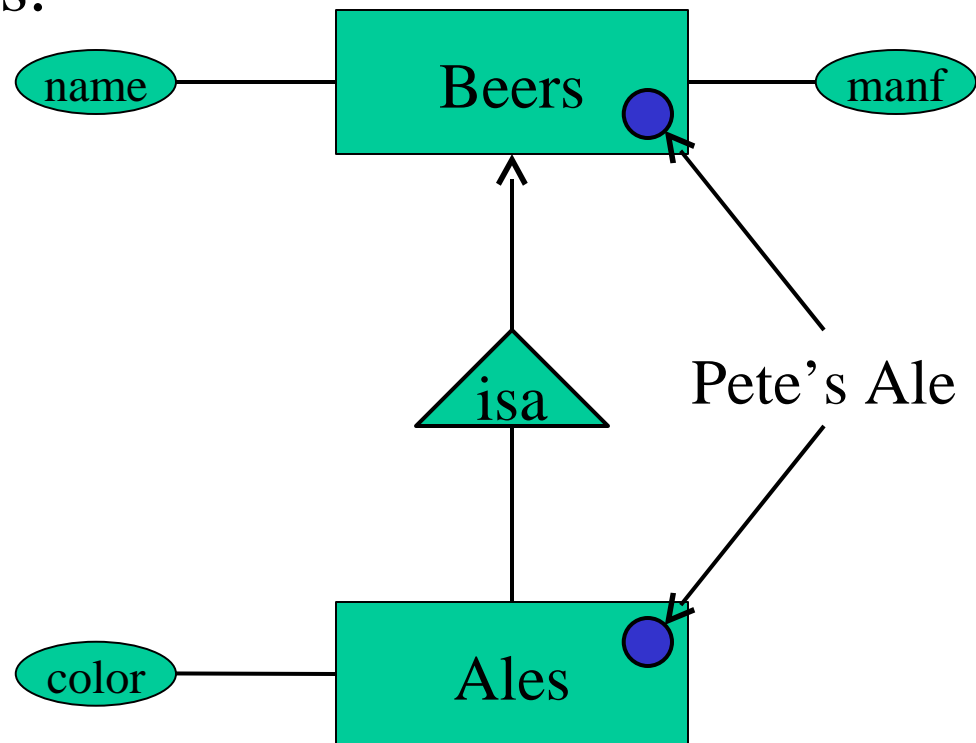
Different Subclass Viewpoints

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.

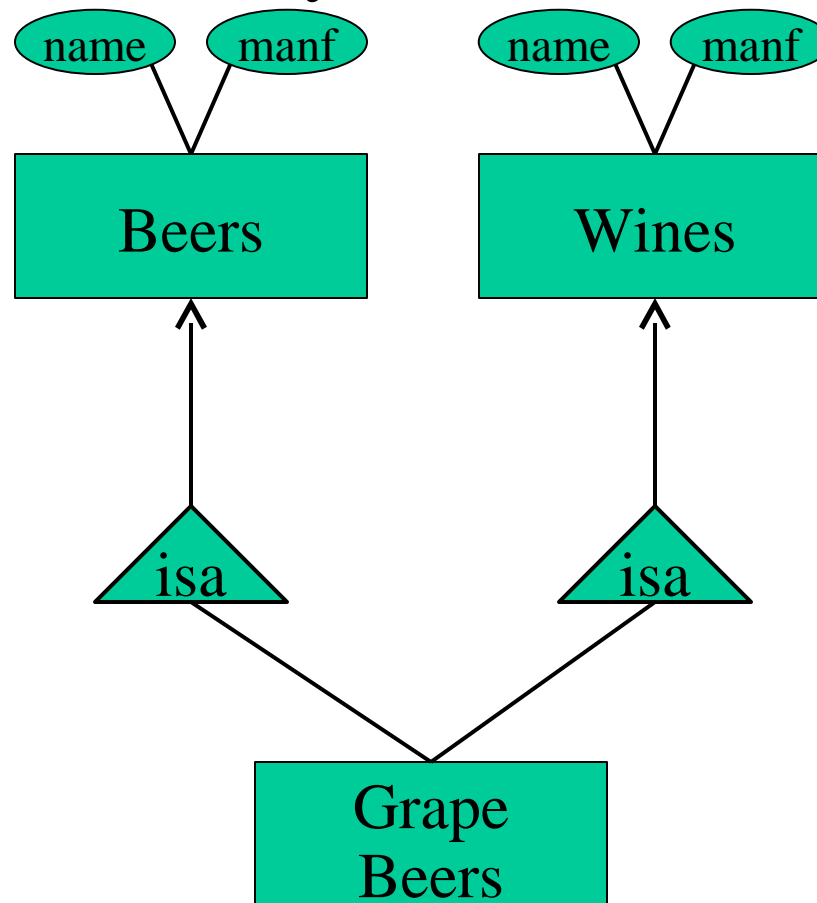
2. Contrasts with *object-oriented viewpoint*: An object (entity) belongs to exactly one class.

◆ It *inherits* properties of its superclasses.



Multiple Inheritance

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



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.

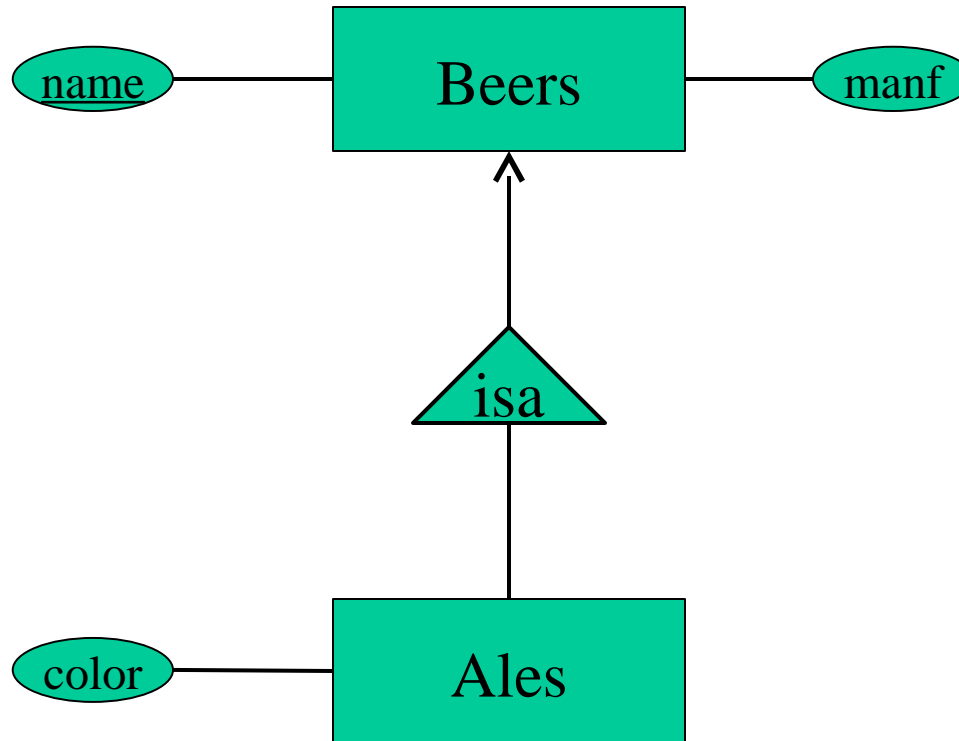
Keys

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.

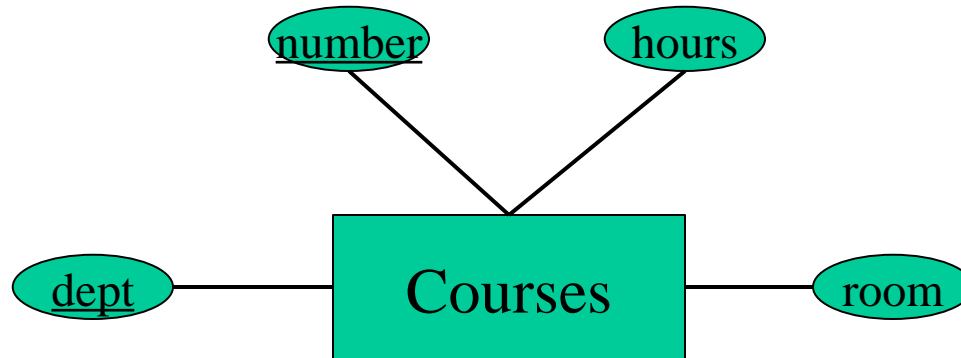
Example

- Suppose name is key for *Beers*.



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

Example: A Multiattribute Key



- Possibly, the combination of hours + room also forms a key, but we have not designated it as such.