PURDUE UNIVERSITY

CS54100: Database Systems

Data Modeling

13 January 2012

Prof. Chris Clifton

Indiana Center for Database Systems

Main categories of data models

- **Logical models**: used to describe, organize and access data in DBMS; application programs refers to such models. They are independent from the physical data structures
  - examples: relational data model, hierarchical data model, object-relational data model
- **Conceptual models**: support the representation of data independently from specific DBMS. Their goal is to provide representations, which are rich in semantics, of the real word entities, their properties and relationships. These models are mainly used for the conceptual design of databases
  - The **Entity-Relationship** is the most well known model in such category

Logical Models: Evolution

- First generation (60):
  - Network data model – Codasyl
  - Hierarchical data model
- Relational data model (70)
- Post-relational data models:
  - Nested relational data models
  - Object-oriented data models
  - Deductive data models (e.g. Datalog and its extensions)
  - Object-relational data models
  - XML

DBMS: languages

- **Data Definition Language** (DDL). Defines:
  - Logical schema of the DB
  - Semantic integrity constraints
  - Authorizations for data accesses
- **Data Manipulation Language** (DML)
  - Used for data retrieval (query language) and for data updates
- **Storage Definition Language** (SDL)
  - Used to define physical access structures

Data Definition Language

- The DDL specifies the schema of the DB
- Provides a syntax to specify the constructs of the data model
- Supports the specification of the
  - database name
  - names and definitions of all the logical units in the schema
    - relations and attributes in the case of the relational model
  - semantic integrity constraints

PURDUE
UNIVERSITY

Entity-Relationship Data Model

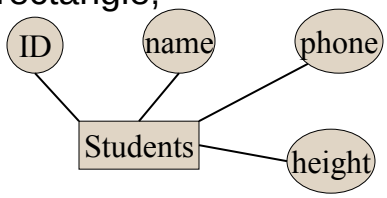
Reading: Chapter 2

Slides adapted from material by Profs. Jeff Ullman (Stanford) and Art Keller (UCSC)

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.

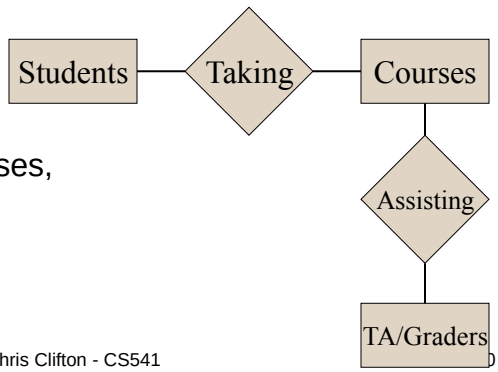
Students	Courses
Sally	CS541
Sally	CS555
Joe	CS541
…	…

Multiway Relationships

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

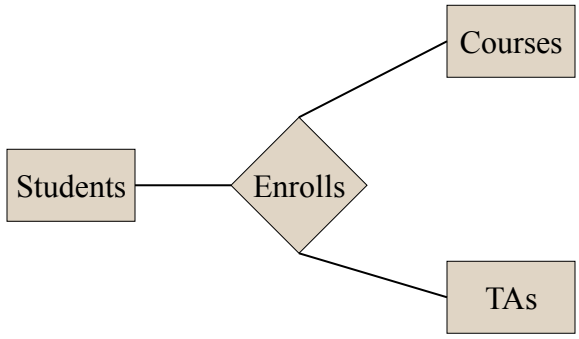
- However, there are some cases where three or more E.S. must be connected by one relationship.
- Example: relationship among students, courses, TA's (and graders).

Possibly, this E/R diagram is OK:



Multiway relationships

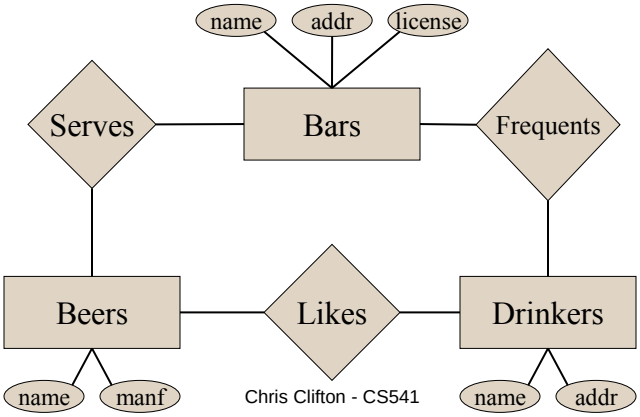
- Works in CS541, because each TA (or grader) is a TA of all students. Connection student-TA is *only* via the course.
- But what if students were divided into sections, each headed by a TA?
  - Then, a student in CS541 would be related to only one of the TA's for CS541. Which one?
- Need a 3-way relationship to tell.



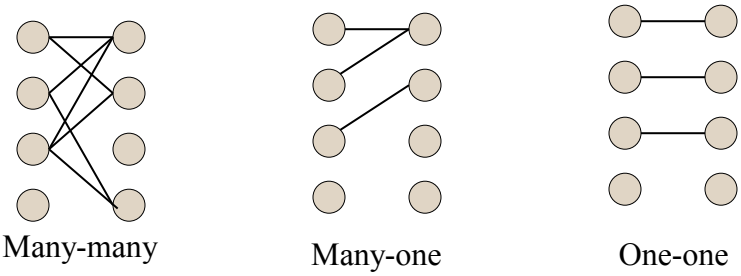
Students	Courses	TAs
Ann	CS541	Li
Sue	CS541	Li
Bob	CS541	Chris
…	…	…

Beers-Bars-Drinkers Example

- Our *(or to give credit where due, Prof. Ullman's)* running example for the course.

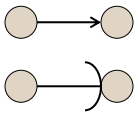


Multiplicity of Relationships

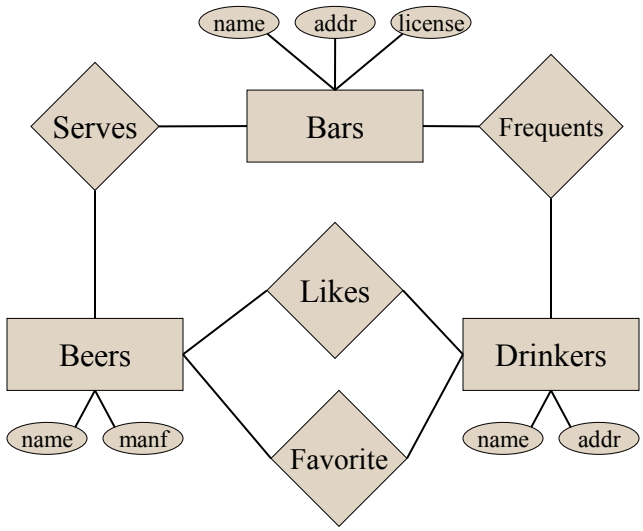


Representation of Many-One

- E/R: arrow pointing to "one."
  - Rounded arrow = "exactly one."

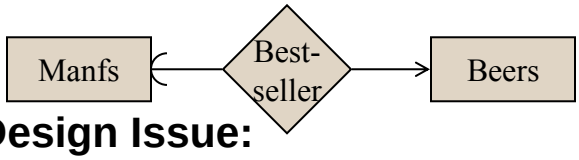


Example: Drinkers Have Favorite Beers



One-One Relationships

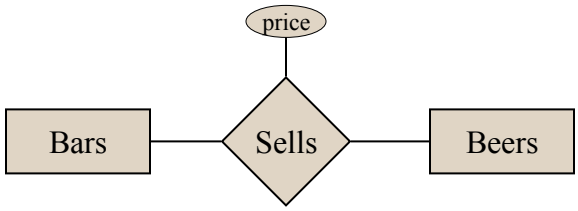
Put arrows in both directions.



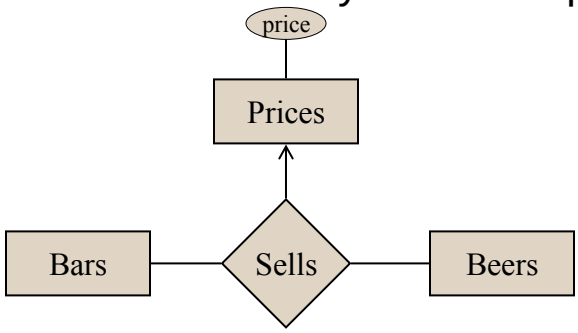
Design Issue:
Is the rounded arrow justified?

Design Issue:
Here, manufacturer is an E.S.
In earlier diagrams it is an attribute.
Which is right?

Attributes on Relationships



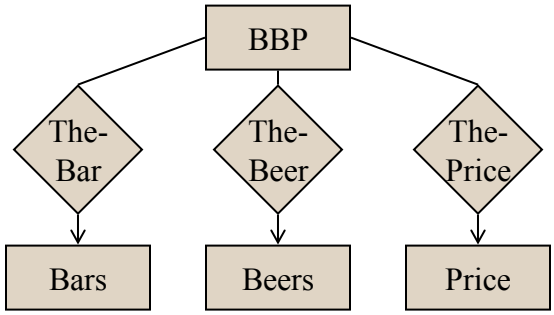
Shorthand for 3-way relationship:



- 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

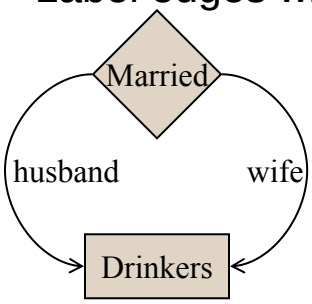
- Baroque in E/R, but necessary in certain "object-oriented" models.
- Create a new connecting E.S. to represent rows of a relationship set.
  - E.g., (Joe's Bar, Bud, $2.50) for the *Sells* relationship.
- Many-one relationships from the connecting E.S. to the others.



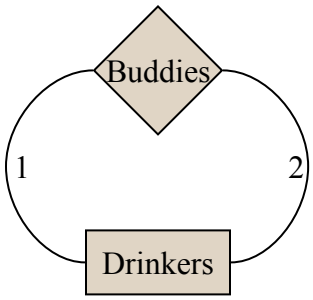
Roles

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
…	…



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*?

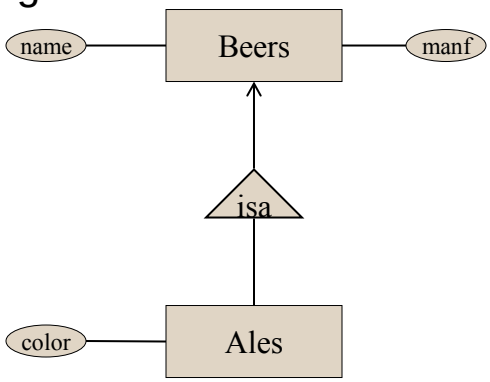
Subclasses

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.

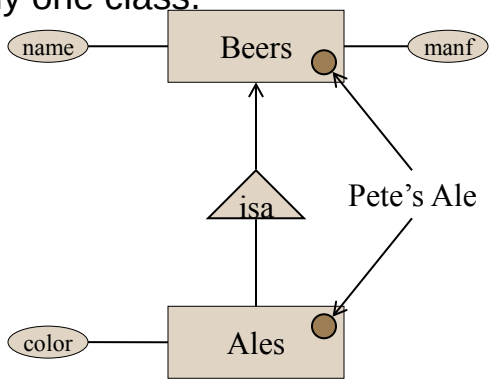
E/R Subclasses

- 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.



PURDUE UNIVERSITY

CS54100: Database Systems

Relational Data Model

18 January 2012

Prof. Chris Clifton

Indiana Center for Database Systems

Relational Database Design

- Goals
  - Capture all the data
    - Nothing lost
  - Represent only the data
    - Prevent data not matching the real world
- How?
  - Division of data into relations
  - Key constraints

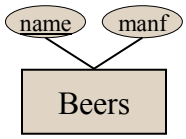
Relational Design

Simplest approach (not always best): convert each E.S. to a relation and each relationship to a relation.

Entity Set → Relation

E.S. attributes become relational attributes.

Becomes:



`Beers(name, manf)`

Keys in Relations

An attribute or set of attributes *K* is a *key* for a relation *R* if we expect that in no instance of *R* will two different tuples agree on all the attributes of *K*.

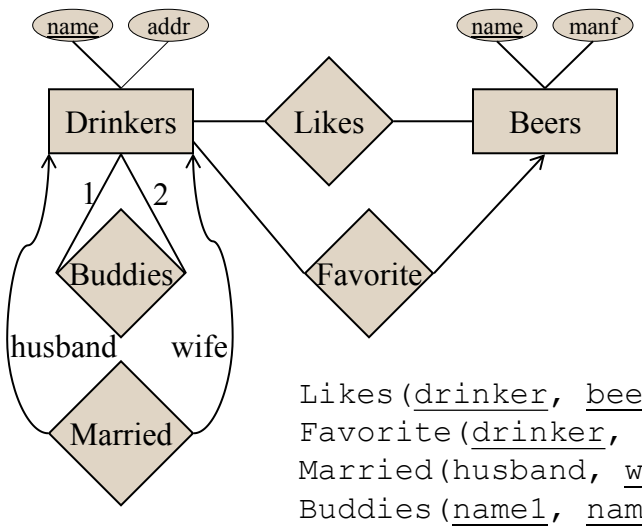
- Indicate a key by underlining the key attributes.
- Example: If `name` is a key for `Beers`:

  `Beers(name, manf)`

E/R Relationships → Relations

Relation has attribute for *key* attributes of each E.S. that participates in the relationship.

- Add any attributes that belong to the relationship itself.
- Renaming attributes OK.
  - Essential if multiple roles for an E.S.



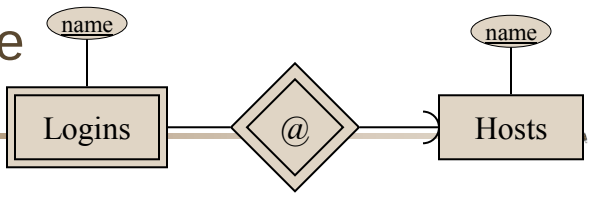
`Likes(`<u>`drinker`</u>`, `<u>`beer`</u>`)`
`Favorite(`<u>`drinker`</u>`, beer)`
`Married(husband, `<u>`wife`</u>`)`
`Buddies(`<u>`name1`</u>`, `<u>`name2`</u>`)`

- For one-one relation `Married`, we can choose either `husband` or `wife` as key.

Weak Entity Sets, Relationships → Relations

- Relation for a weak E.S. must include its full key (*i.e.*, attributes of related entity sets) as well as its own attributes.
- A supporting (double-diamond) relationship yields a relation that is actually redundant and should be deleted from the database schema.

Example



`Hosts(`<u>`hostName`</u>`)`

`Logins(`<u>`loginName`</u>`, `<u>`hostName`</u>`)`

`At(`<u>`loginName`</u>`, `<u>`hostName`</u>`, hostName2)`

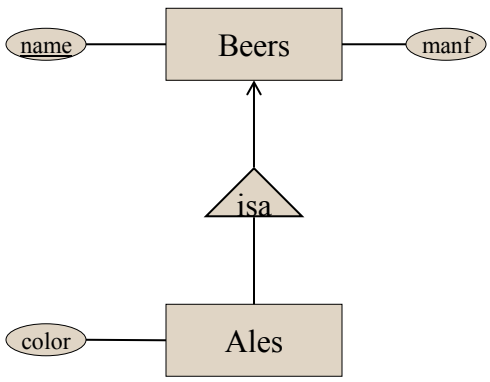
- In `At`, `hostName` and `hostName2` must be the same host, so delete one of them.
- Then, `Logins` and `At` become the same relation; delete one of them.
- In this case, `Hosts`' schema is a subset of `Logins`' schema. Delete `Hosts`?

Subclasses → Relations

Three approaches:

1. Object-oriented: each entity is in one class. Create a relation for each class, with all the attributes for that class.
   - Don't forget inherited attributes.
2. E/R style: an entity is in a network of classes related by *isa*. Create one relation for each E.S.
   - An entity is represented in the relation for each subclass to which it belongs.
   - Relation has only the attributes attached to that E.S. + key.
3. Use nulls. Create one relation for the root class or root E.S., with all attributes found anywhere in its network of subclasses.
   - Put `NULL` in attributes not relevant to a given entity.

Example



OO-Style

name	manf
Bud	A.B.

Beers

name	manf	color
SummerBrew	Pete's	dark

Ales

E/R Style

name	manf
Bud	A.B.
SummerBrew	Pete's

Beers

name	Color
SummerBrew	dark

Ales

Using NULLS

name	manf	color
Bud	A.B.	NULL
SummerBrew	Pete's	dark

Beers