Slides to be used with Ullman/Widom *A First Course in Database Systems*, Prentice-Hall, 1997

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 v.l.a.d.
   - Example: bank and its ATM machines.
4. Supports secure, atomic access to v.l.a.d.
   - 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.
- "Big three" (relational) DBMS companies — Oracle, Informix, Sybase — are among the largest software companies in the world.
  - ❖ IBM also plays, with its DB2.
  - ❖ Microsoft *Access* becoming a force for cheap DBMS on the desktop.
- Relational companies also challenged by startup "object-oriented DB" companies.
- But countered with "object-relational" systems, which retain the relational core while allowing type extension as in OO systems.

Slides to be used with Ullman/Widom *A First Course in Database Systems*, Prentice-Hall, 1997

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."

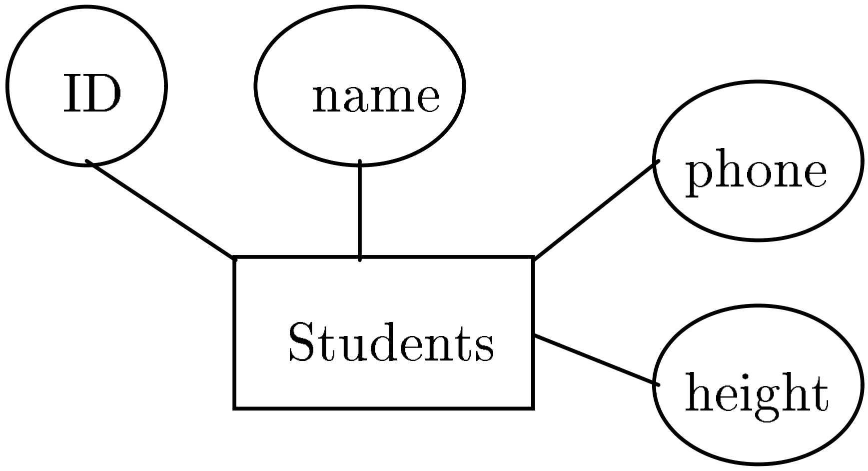
3. DBMS implementation.

CS145 = (1) + (2), while (3) is covered in CS245, CS346, CS347.

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 or "instance variables."
- 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.

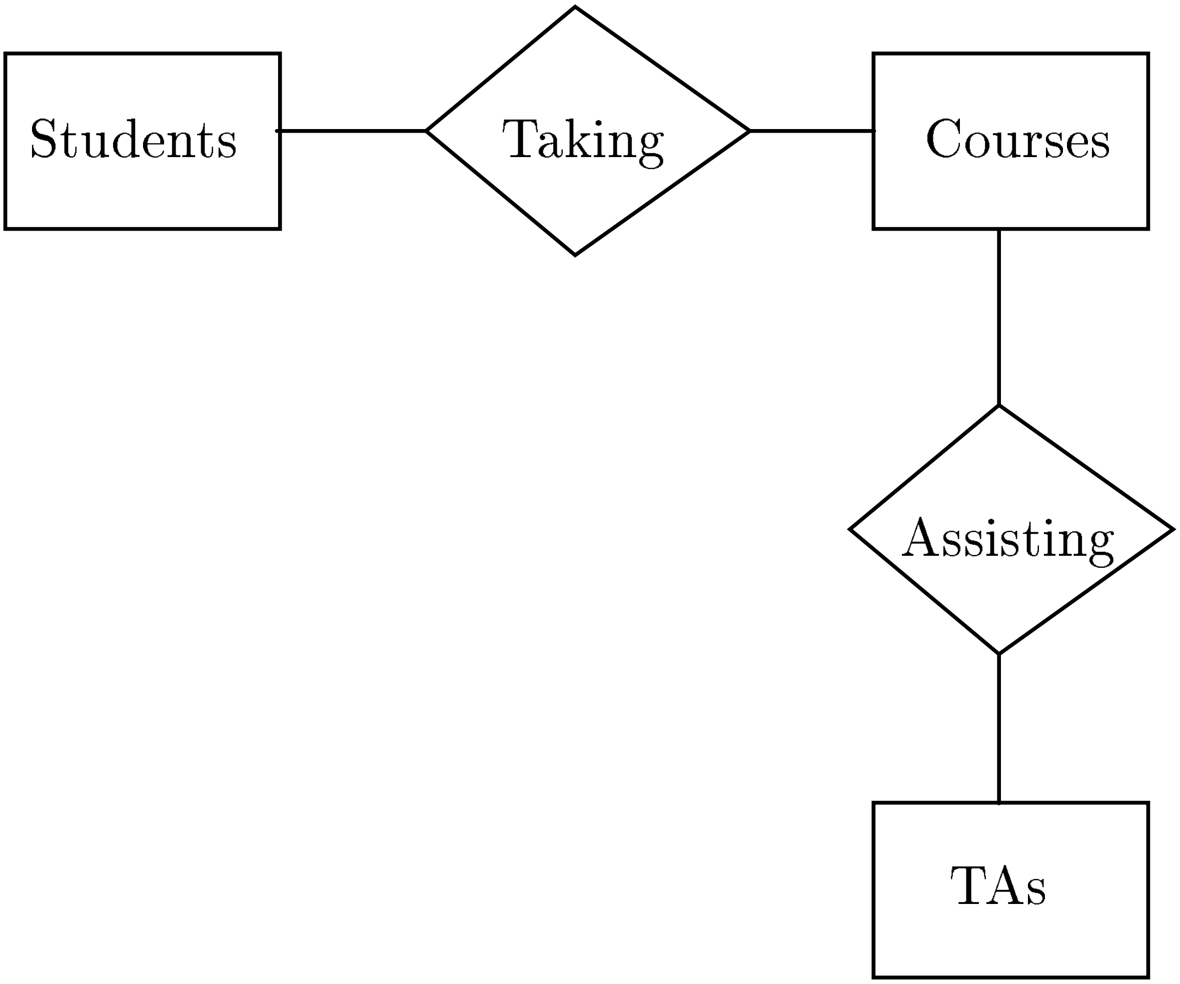
Students	Courses
Sally	CS145
Sally	CS244
Joe	CS145
...	...

Multiway Relationships

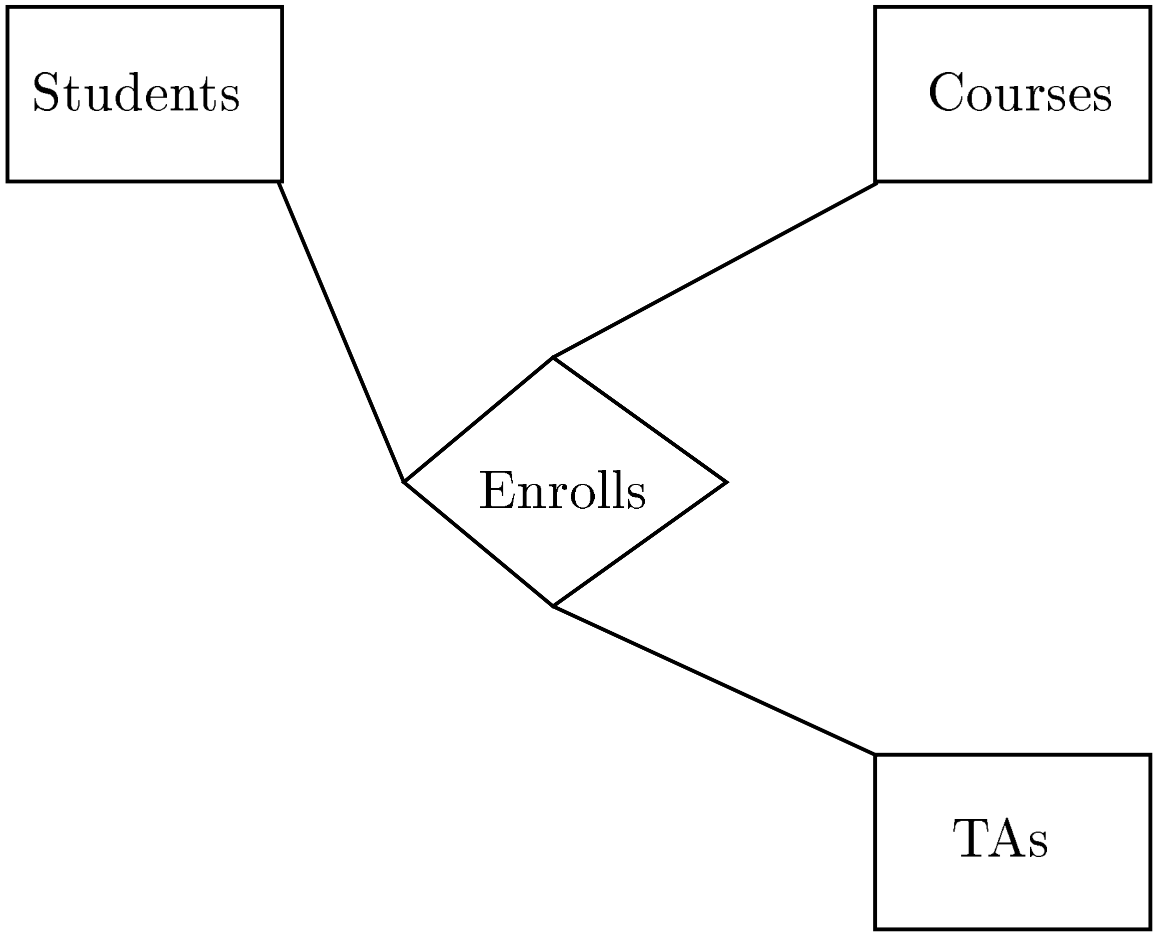
Usually *binary* relationships (connecting two E.S.) suffice.

- However, there are some cases where three or more E.S. must be connected by one relationship.

- Example: relationship among students, courses, TA's. Possibly, this E/R diagram is OK:



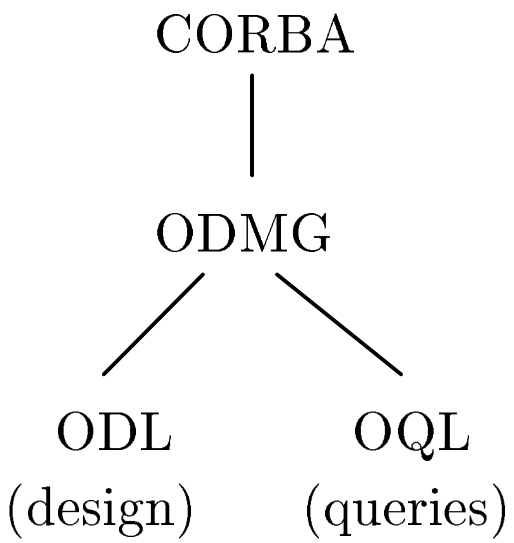
- Works in CS145, because each TA is a TA of all students. Connection student-TA is *only* via the course.
- But what if students were divided into sections, each headed by a TA?
  - ❖ Then, a student in CS145 would be related to only one of the TA's for CS145. Which one?
- Need a 3-way relationship to tell.

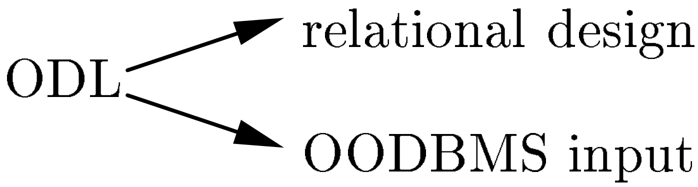


Students	Courses	TAs
Sally	CS145	Chen
Sue	CS145	Himanshu
Joe	CS145	Chen
...	...	...

ODL

- Design language derived from the OO community:





- Can be used like E/R as a preliminary design for a relational DB.
- It can also be direct input to some OODBMS's.

ODL Class Declarations

```
interface <name> {
    elements = attributes, relationships,
        methods
}
```

Element Declarations

```
attribute <type> <name>;
relationship <rangetype> <name>;
```

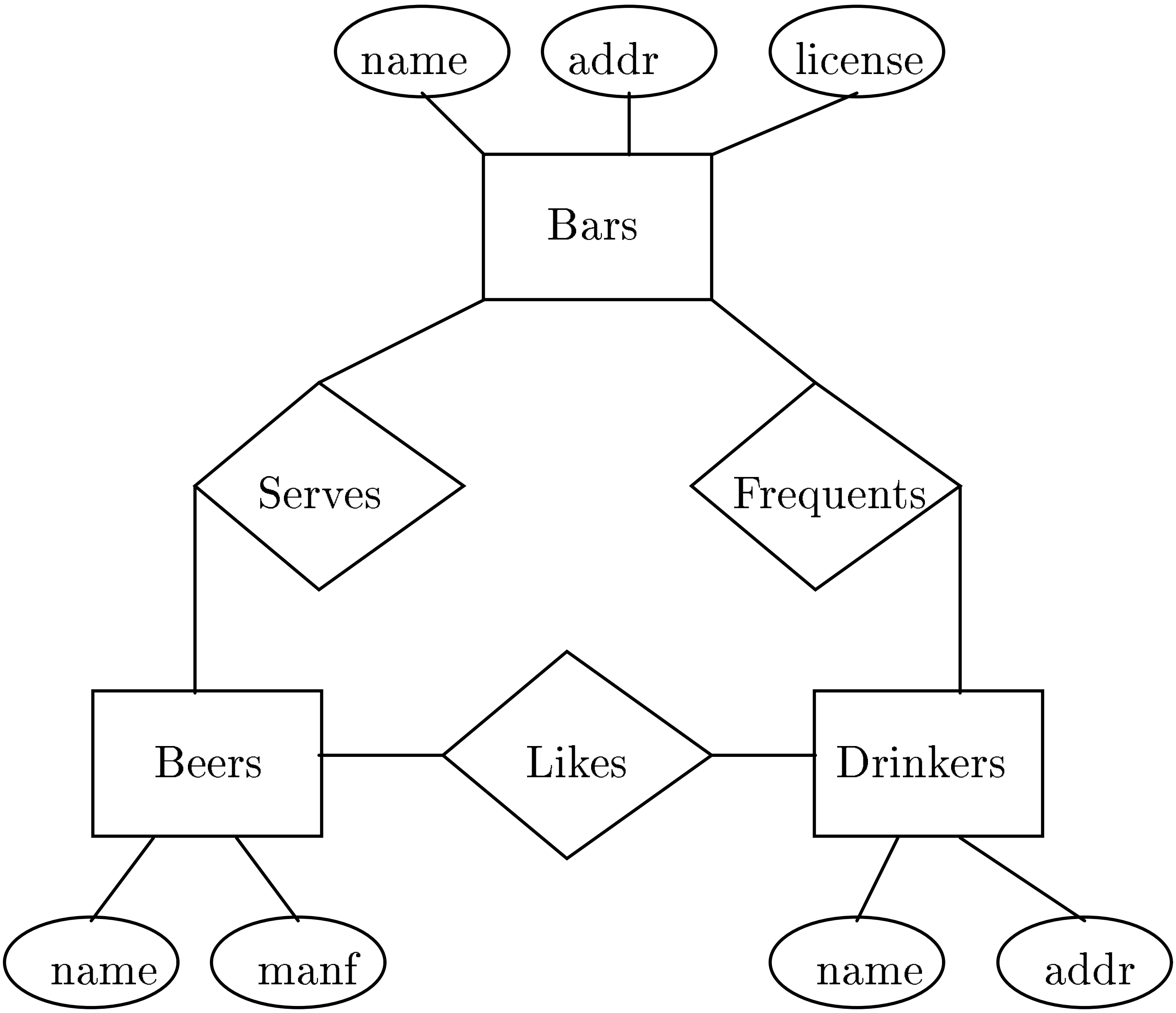
Method Example

```
float gpa(in: Student) raises(noGrades)
```

- `float` = return type.
- `in:` indicates `Student` argument is read-only.
  - Other options: `out`, `inout`.
- `noGrades` is an exception that can be raised by method `gpa`.

Beers-Bars-Drinkers Example

- Our running example for the course.



```
interface Beers {
    attribute string name;
    attribute string manf;
    relationship Set<Bars> servedAt
        inverse Bars::serves;
    relationship Set<Drinkers> fans
        inverse Drinkers::likes;
}
```

- Relationships have inverses.
- An element from another class is indicated by <class>`::`
- Form a set type with `Set<type>`.

```
interface Bars {
    attribute string name;
    attribute Struct Addr
        {string street, string city, int zip}
            address;
    attribute Enum Lic {full, beer, none}
            licenseType;
    relationship Set<Drinkers> customers
        inverse Drinkers::frequents;
    relationship Set<Beers> serves
        inverse Beers::servedAt;
}
```

- Structured types have names and bracketed lists of field-type pairs.
- Enumerated types have names and bracketed lists of values.

```
interface Drinkers {
    attribute string name;
    attribute Struct Bars::Addr
            address;
    relationship Set<Beers> likes
        inverse Beers::fans;
    relationship Set<Bars> frequents
        inverse Bars::customers;
}
```

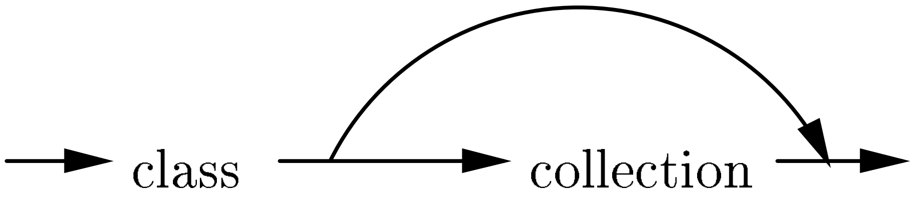
- Note reuse of `Addr` type.

ODL Type System

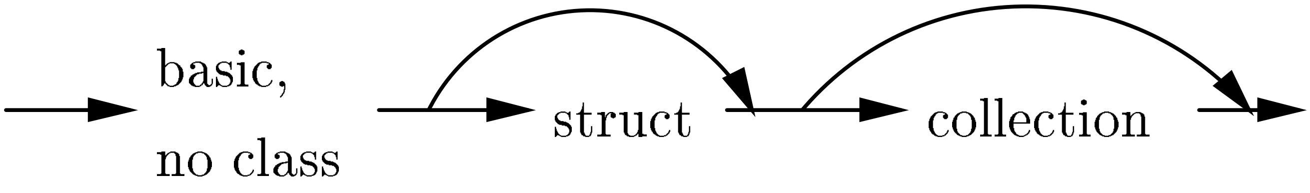
- Basic types: int, real/float, string, enumerated types, and classes.
- Type constructors: `Struct` for structures and four *collection types*: `Set`, `Bag`, `List`, and `Array`.

Limitation on Nesting

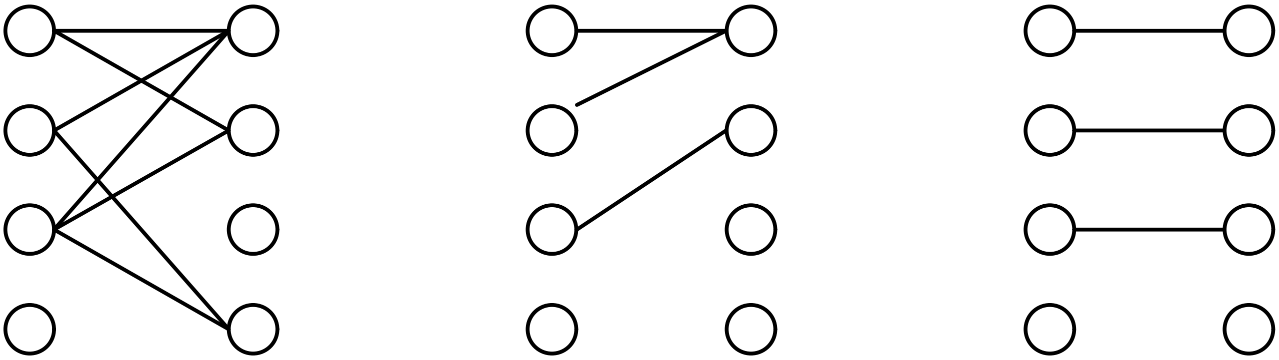
Relationship



Attribute



Multiplicity of Relationships



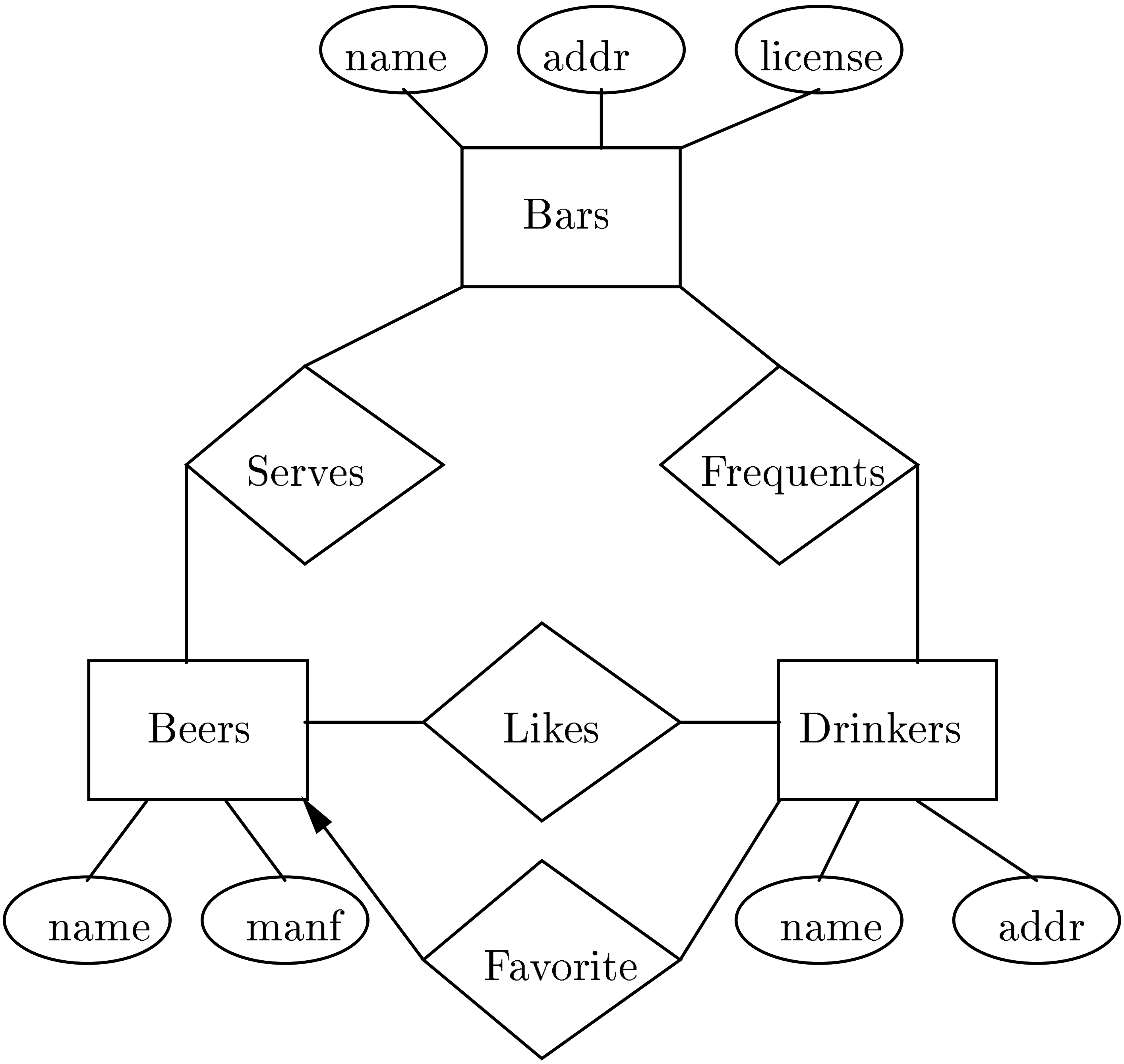
Many-many Many-one One-one

Representation of Many-One

- E/R: arrow pointing to "one."
  - Rounded arrow = "exactly one."
- ODL: don't use a collection type for relationship in the "many" class.
  - Collection type remains in "one."

Slides to be used with Ullman/Widom *A First Course in Database Systems*, Prentice-Hall, 1997

Example: Drinkers Have Favorite Beers



```
interface Drinkers {
    attribute string name;
    attribute Struct Bars::Addr
            address;
    relationship Set<Beers> likes
        inverse Beers::fans;
    relationship Beers favoriteBeer
        inverse Beers::realFans;
    relationship Set<Bars> frequents
        inverse Bars::customers;
}
```

- Also add to `Beers`:

```
    relationship Set<Drinkers> realFans
        inverse Drinkers::favoriteBeer;
```

One-One Relationships

- E/R: arrows in both directions.
- ODL: omit collection types 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?