More Design Issues

1. Roles.
2. Subclasses.
4. Weak entity sets.
5. Design principles and some hard examples.
Roles

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

- Label edges with *roles* to distinguish.

![Diagram showing roles in a relationship]

<table>
<thead>
<tr>
<th>Husband</th>
<th>Wife</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_1$</td>
<td>$d_2$</td>
</tr>
<tr>
<td>$d_3$</td>
<td>$d_4$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Notice *Buddies* is symmetric, *Married* not.

* No way to say “symmetric” in E/R.
* But in ODL, symmetric relations are their own inverse.
Roles in ODL

No problem; names of relationships handle “roles.”

interface Drinkers {
    attribute string name;
    attribute Struct Bars::Addr address;
    relationship Set<Beers> likes
        inverse Beers::fans;
    relationship Set-bars frequents
        inverse Bars::customers;
    relationship Drinkers husband
        inverse wife;
    relationship Drinkers wife
        inverse husband;
    relationship Set<Drinkers> buddies
        inverse buddies;
}

- Notice that Drinkers:: is optional when the inverse is a relationship of the same class.

Design Issue

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

Subclass = special case = fewer entities/objects = more properties.

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

ODL Subclasses

Follow name of subclass by colon and its superclass.

```java
interface Ales:Beers {
    attribute String color;
}
```

- Objects of the Ales class acquire all the attributes and relationships of the Beers class.
E/R Subclasses

- *isa* triangles indicate the subclass relation.
Difference in Subclass Viewpoints

- In ODL, an entity is in exactly one class.
  - It inherits properties of its superclass(es).

- In E/R, an entity has “representation” in all the subclasses to which it logically belongs.
  - Its properties are the union of the properties of these classes.

- The distinction matters later, when we convert ODL and E/R to relations.

![Diagram of subclasses and properties]
Multiple Inheritance

Theoretically, a class/E.S. could be a subclass of several classes.
Problems

How should conflicts be resolved?

- Example: `manf` means grower for wines, bottler for beers. What does `manf` mean for “grape beers”?

- E/R mute on multiple inheritance; ODL leaves it to implementer.
Keys

= set of attributes whose values can belong to at most one entity or object.

- In E/R: underline all key attributes.
- In E/R, each E.S. must have a key.
- Example: Suppose name is key for Beers.

- Beer name is also key for ales.
  - In general, key at root (no multiple inheritance!) is key for all.
Keys in ODL

Indicate with key(s) following the class name, and a list of attributes forming the key.

✦ Several lists may be used to indicate several alternative keys.
✦ Parentheses group members of a key, and also group key to the declared keys.
✦ Thus, \(\text{key}(a_1, a_2, \ldots, a_n)\) = “one key consisting of all \(n\) attributes.”
  \(\text{key} a_1, a_2, \ldots, a_n\) = “each \(a_i\) is a key by itself.

• Example:

  interface Beers
    (key name)
  {
    attribute string name ...
  }
A Multiattribute Key

E/R requires exactly one key, but ODL allows zero or more keys.

Very Important: In ODL, “object identity” serves to distinguish objects, so no key at all is necessary.
Weak Entity Sets

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

- Called a *weak E.S.*

- Represented by putting double rectangle around the weak E.S. and a double diamond around each relationship to which the weak E.S. is linked to an E.S. that provides part of its key.

- Use of many-one relationship (includes 1-1) essential.

  - With a many-many, we wouldn’t know which entity provided the key value.

- **Very Important:** There is no such thing as a “weak class” in ODL.

  - Because objects have object-identity, classes don’t need keys, and therefore they don’t need to “borrow” keys from related classes.
Example: Logins

Login name = user name + host name, e.g., ullman@shalmaneser.stanford.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. ullman.

- Key for a login = the user name at the host (which is unique for that host only) + the name of the host (which is unique globally).
Example: Chain of “Weakness”

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

- 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.
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.