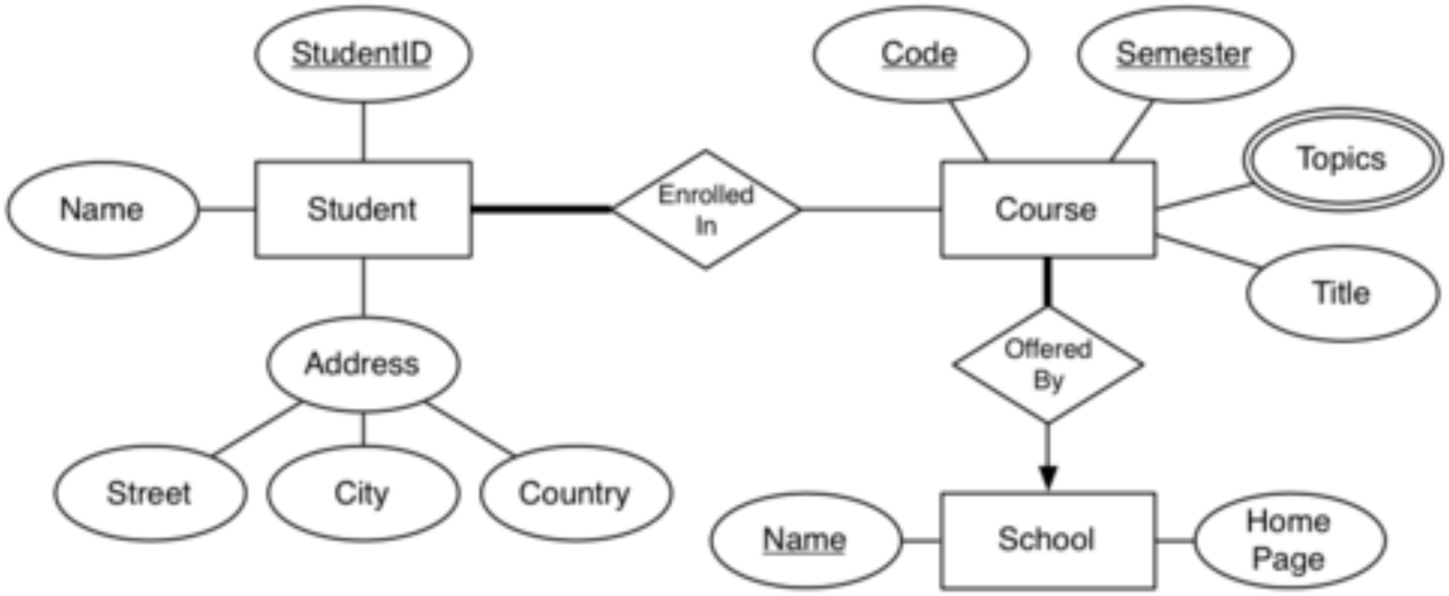
ER: the story so far

Entities, relationships, attributes, keys, cardinality, participation, ...

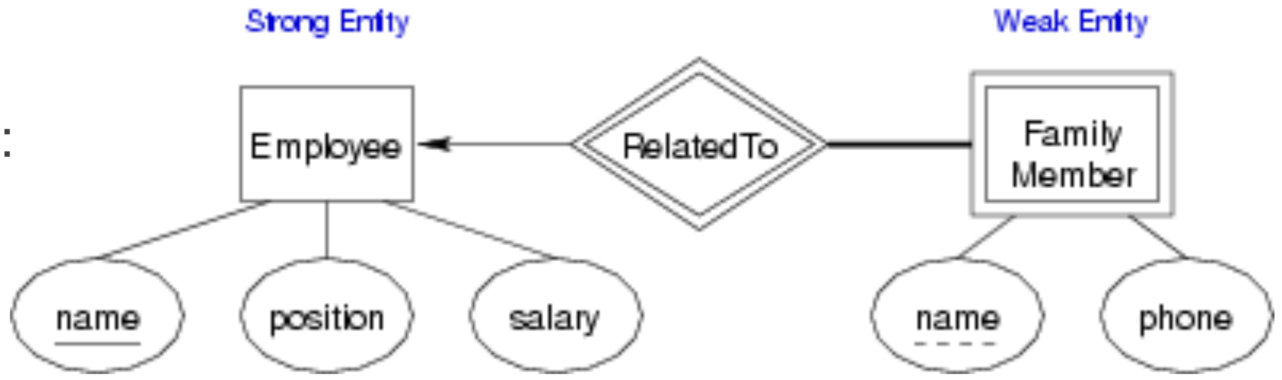


Weak Entity Sets

Weak entities

- exist only because of association with strong entities.
- typically, these entities do not have key of their own; can only be identified by considering the primary key of another (owner) entity
- must have total participation in the relationship with the owner entity
- could have a *discriminator*

Example:

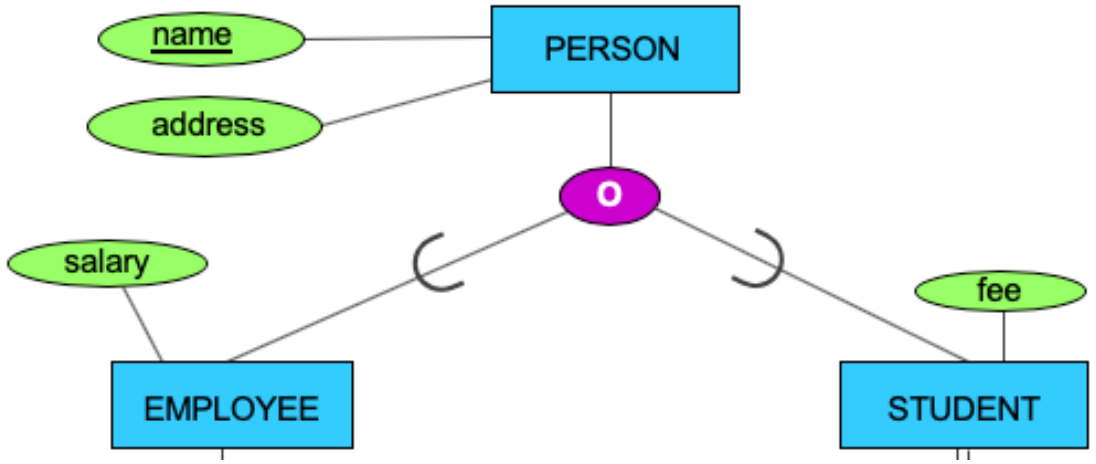


Subclasses (Specialisation/Inheritance)

An entity can be *specialised* into sub grouping:

A *subclass* of an entity set *A* is a set of entities:

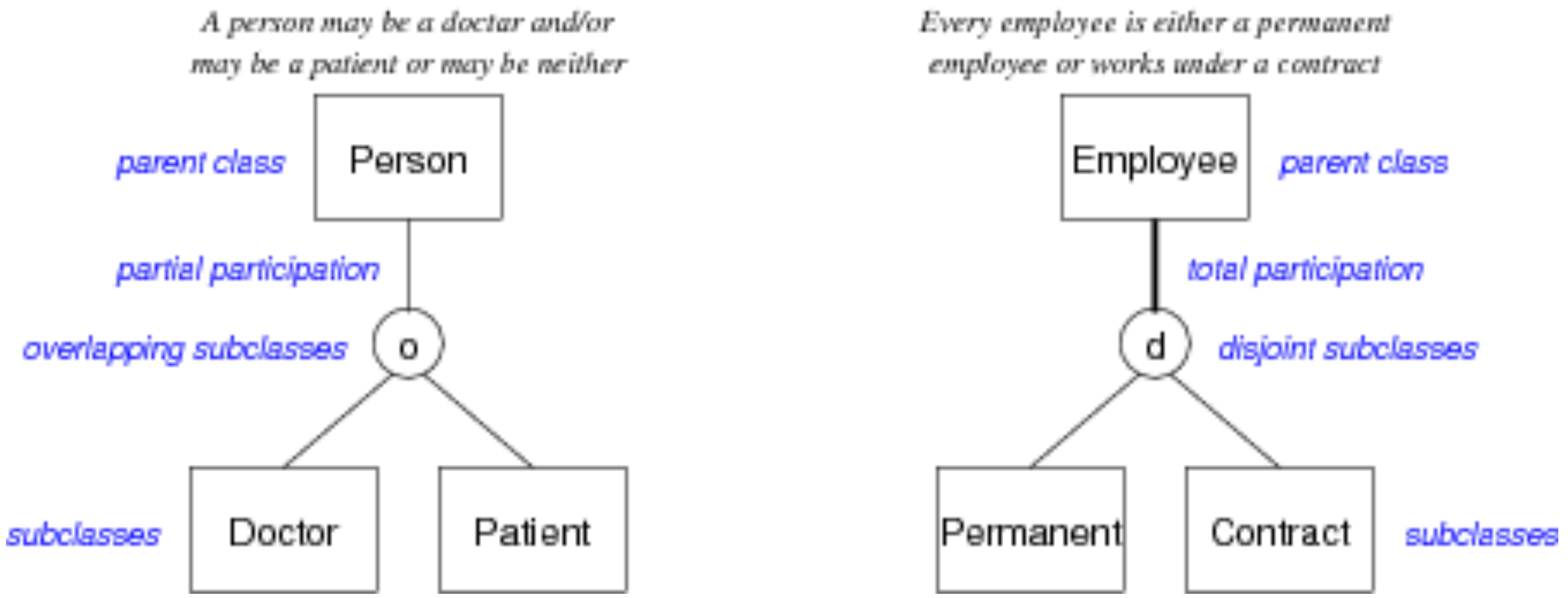
- with all attributes of *A*, plus (usually) its own attributes
- that is involved in all of *A*'s relationships, plus its own
- i.e., subclass *inherits* attributes and relationships from its parent



Subclasses (Specialisation/Inheritance)

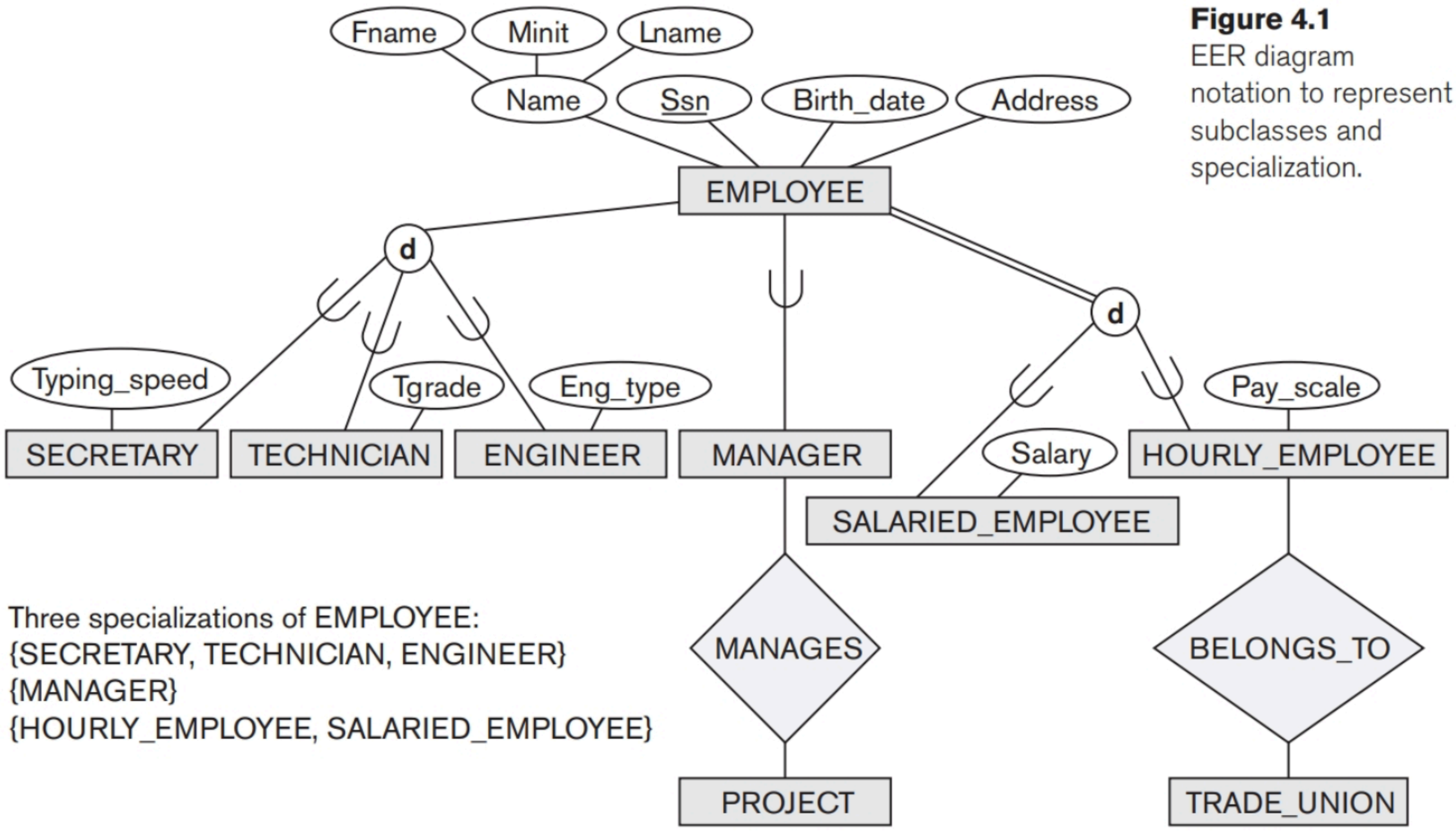
Properties of subclasses:

- *overlapping* or *disjoint* (can an entity be in multiple subclasses?)
- *total* or *partial* (does every entity have to also be in a subclass?)



A class/subclass relationship is often called an IS-A (or IS-AN) relationship because of the way we refer to the concept. We say a DOCTOR is a PERSON …

Subclasses and Inheritance



Three specializations of EMPLOYEE:
{SECRETARY, TECHNICIAN, ENGINEER}
{MANAGER}
{HOURLY_EMPLOYEE, SALARIED_EMPLOYEE}

Figure 4.1
EER diagram notation to represent subclasses and specialization.

Design Using the ER Model

ER model: simple, powerful set of data modelling tools

Some considerations in designing ER models:

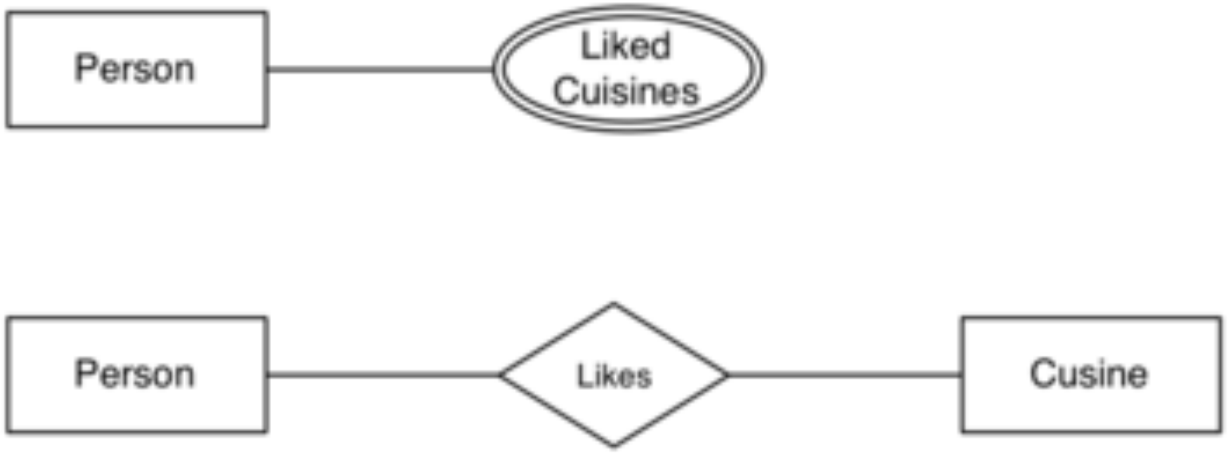
- should an "object" be represented by an attribute or entity?
- is a "concept" best expressed as an entity or relationship?
- should we use n-way relationship or several 2-way relationships?
- is an "object" a strong or weak entity? (usually strong)
- are there subclasses/superclasses within the entities?

Answers to above are worked out by thinking about the application domain.

Entities vs. Attributes

The following two diagrams both represent

a person has some types of food that they like



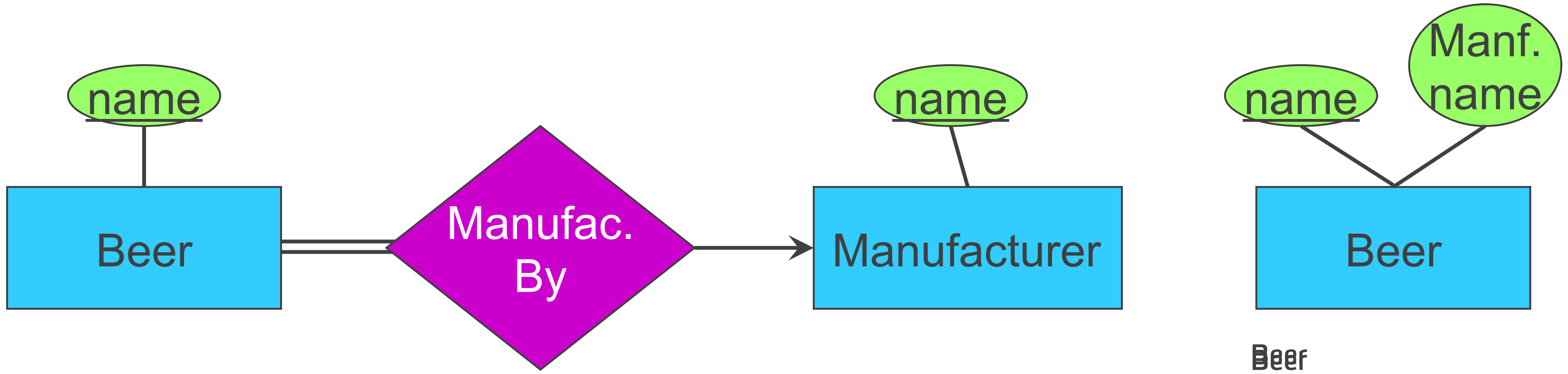
Why might we favour one over the other?

Entities Vs. Attributes

Sometimes it is not clear which concepts are worthy of being entities, and which are handled more simply as attributes …

Example:

Which are the pros and cons of each of the two designs below?

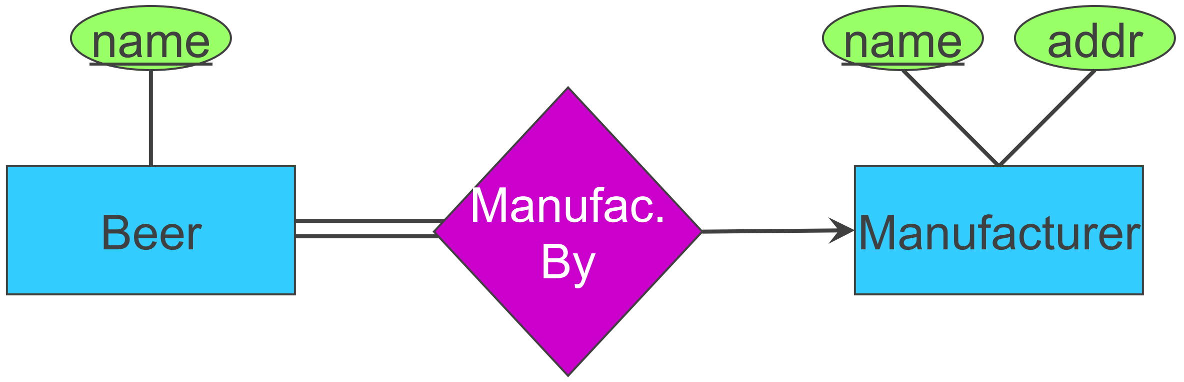


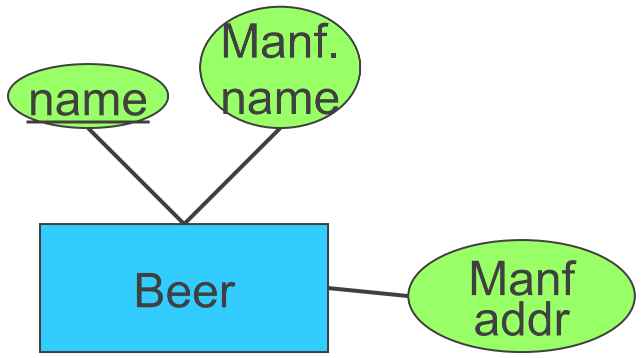
Beer
(Tooheys New, Tooheys)
(Tooheys Old, Tooheys)
(Coopers Larger, Coopers) …

"Don't Say the Same Thing More Than Once"

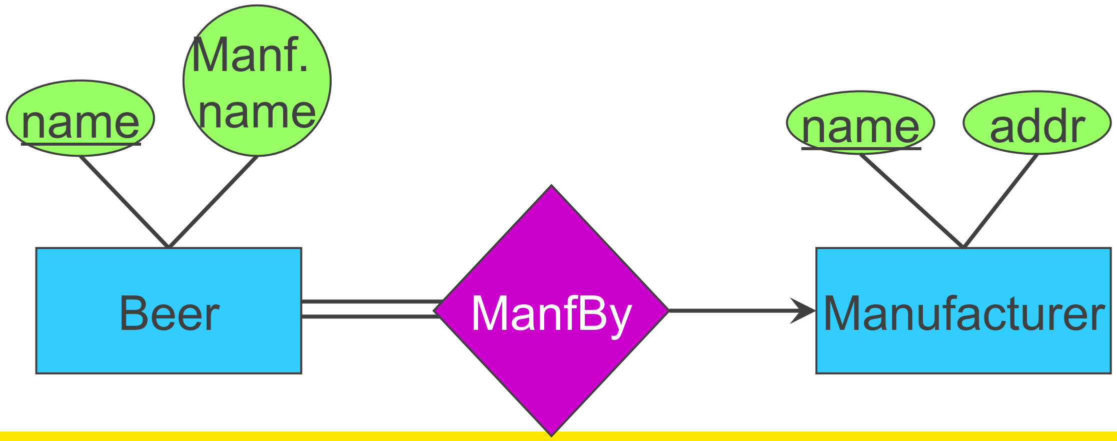
Redundancy wastes space and encourages inconsistency

Examples:





Which is good design, and which is bad? Why?

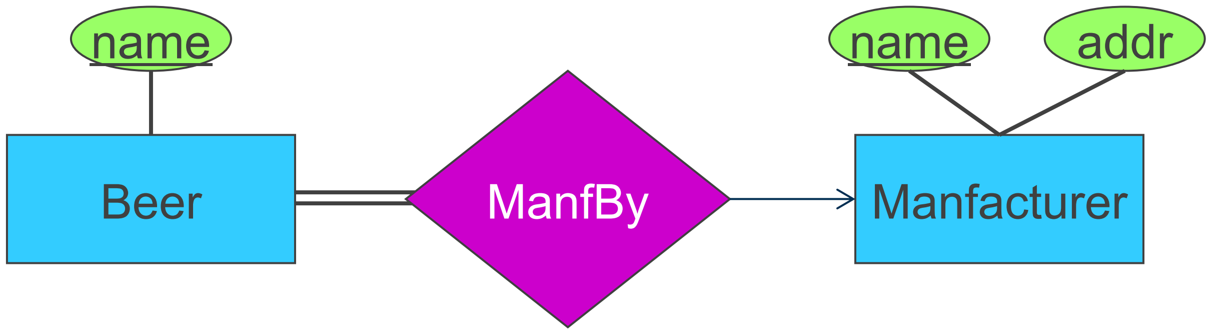


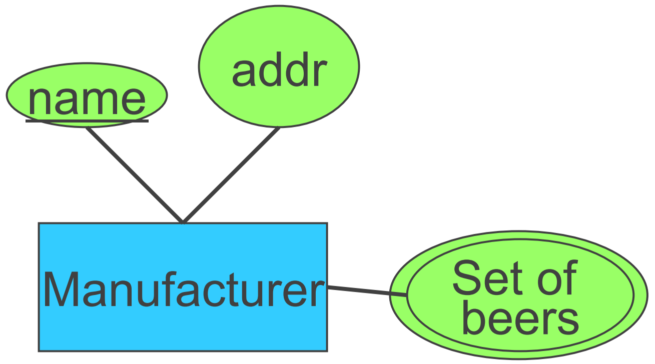
Beer
(Tooheys New, Tooheys, Lidcombe Sydney)
(Tooheys Old, Tooheys, Lidcombe Sydney)
(Coopers Larger, Coopers, Regency Park Adelaide)
(Coppers Light, Coopers, Regency Park Adelaide) ...

Entity Vs. Attribute

Make an entity if either:

- It is more than a name of something; i.e., it has non-key attributes or relationships with other entities, or
- It can be placed in the "many" side in a many-one relationship





The above design illustrates both points:

- Manfs deserves to be an entity because we record addr, a non-key attribute
- Beers deserves to be an entity because it is at the "many" end
- If not, we would have to make "set of beers" an attribute of Manfs

Design Using the ER Model

ER diagrams are typically too large to fit on a single screen (or a single sheet of paper, if printing)

One commonly used strategy:

- define entity sets separately, showing attributes
- combine entities and relationships on a single diagram (but without showing entity attributes)
- if very large design, may use several linked diagrams