Data Models

Critical aspects of data

- Semantics
  - What do the data mean?
  - Ontology, taxonomy, schema
- Syntax
  - How are the data structured
  - XML, HDF, CSV

What is a data model?

- Structure of the data
- Operations on the data
- Constraints on the data

- Types of data models
  - Structured data
  - Semi-structured

Data Modeling/Design

- Conceptual design
  - Model independent of the choice of implementation, concepts and relationships
  - Create a model of a “mini-world”
- Logical Design
  - Full details of data without regard to physical implementation
    - all attributes, types, relationships, cardinality, etc…
- Physical Data Model
  - Mapping into specific implementation details (DBMS, XML, etc).

Steps in Building a DB Application

- Step 0: pick an application domain
  - E.g., course management
- Step 1: conceptual design
  - Decide on what to model in the application domain
    - E.g., instructors, students, courses, etc.
  - need a modeling language to express what you want
  - ER model is the most popular such language
  - output: an ER diagram of the app. domain

Steps in Building a DB Application

- Step 2: pick a type of DBMS
  - Here we use relational DBMS
- Step 3: translate ER design to a relational schema
  - use a set of rules to translate ER to rel. schema
  - use a set of schema refinement rules to transform the above rel. schema into a good rel. schema
- At this point
  - you have a good relational schema on paper

Steps in Building a DB Application

- Subsequent steps include
  - implement your relational DBMS using a "database programming language" called SQL
  - ordinary users cannot interact with the database directly
  - and the database also cannot do everything you want
  - hence write your application program in Php, C++, Java, Python, etc. to handle the interaction and take care of things that the database cannot do
- So, the first thing we should start with is to learn ER model ...

ER Model

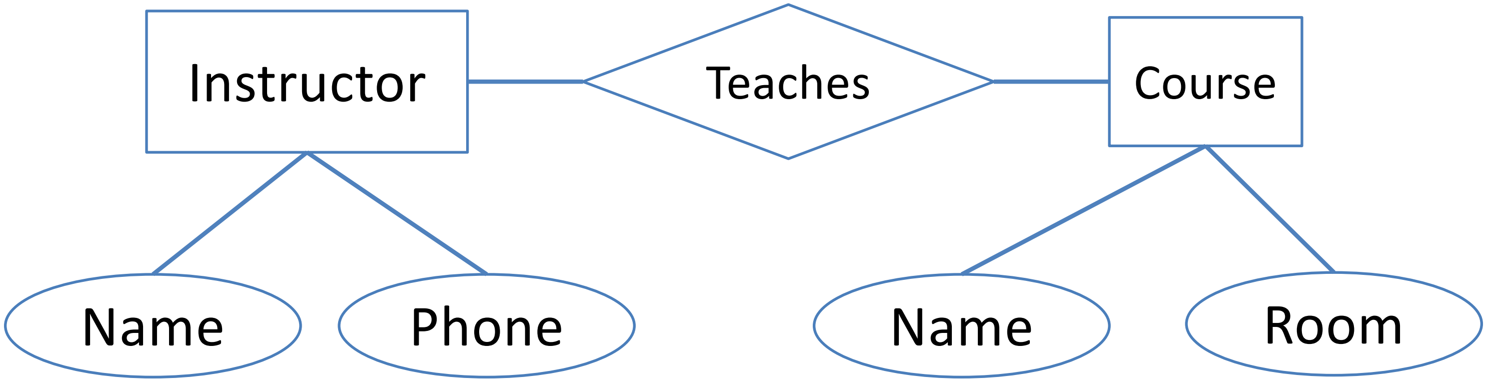
- Gives us a language to specify
  - what information the db must hold
  - what are the relationships among components of that information
- Proposed by Peter Chen in 1976
- What we will cover
  - basic stuff
  - subclasses
  - constraints
  - weak entity sets
  - design principles

http://www.csc.lsu.edu/~chen/

Entity Relationship Models

- The E/R model allows us to sketch data model designs (schema).
  - Includes some constraints, but not operations.
- Designs are pictures called *entity-relationship diagrams*.

A Simple ER Model

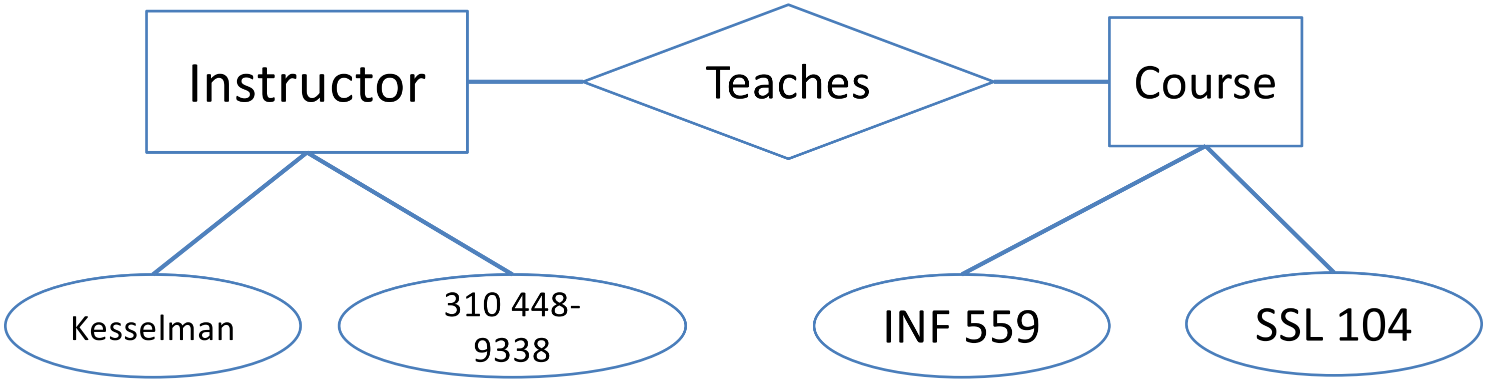


Mini-world of instructors and courses

In English:

- Instructors teach courses
- Courses have instructors
- Instructors have a name and phone number
- Courses have a name and a room number

One possible instance…



What about the second section?

Entity-Relationship Model

- Semantic model
- We model concepts
  - Above abstraction in database
- Identify set of entities and relationships between those entities

Entity Sets

- *Entity* = "thing" or object.
  - What are the set of things we care about? May be physical (person) or conceptual (e.g. vacation)
  - Must be possible to distinguish one entity from another
- *Entity set* = collection of similar entities.
  - Similar to a class in object-oriented languages.

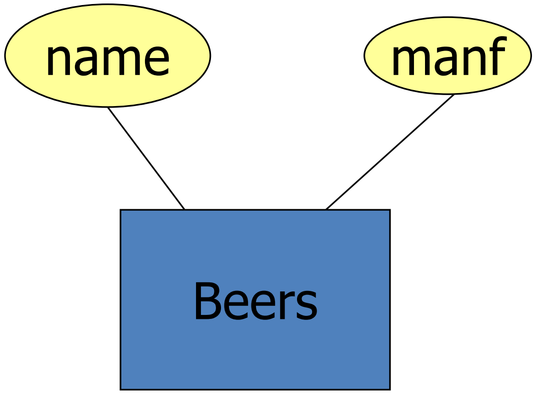
Attributes

- A fact, aspect, or property of (the entities of) an entity set.
  - Attributes are simple values, e.g. integers or character strings, not structs, sets, etc.
- Attributes have
  - A name
  - An associated entity
  - Domains of possible values
  - Values from the domain for each instance of the entity

E/R Diagrams

- In an entity-relationship diagram:
  - Entity set = rectangle.
  - Attribute = oval, with a line to the rectangle representing its entity set.

Example:



- Entity set Beers has two attributes, name and manf (manufacturer).
- Each Beers entity has values for these two attributes, e.g. (Bud, Anheuser-Busch)

Entity or Attribute?

- How do you choose?
  - They both represent objects or facts about the world
  - They both often capture “nouns:
- General guidelines
  - Entity can have smaller subparts, attributes cannot
  - Entities can have relationships, but attributes cannot

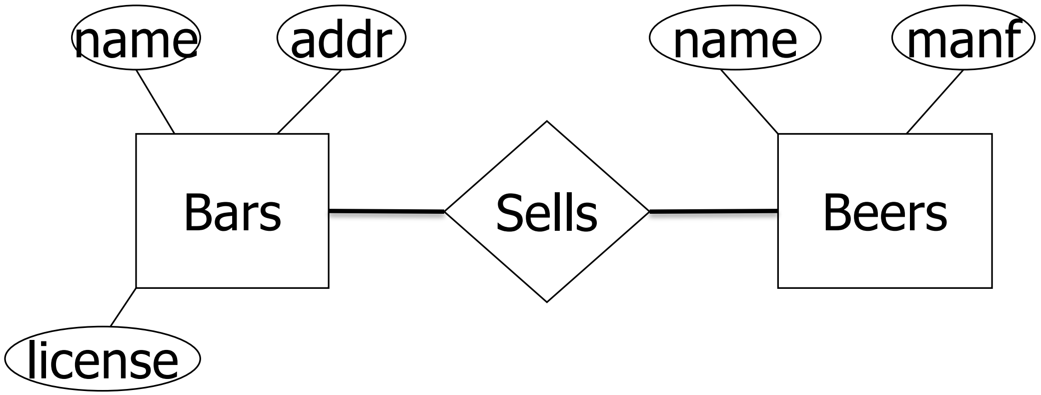
Entity or Attribute?

- Beer
- Supplier
- Manufacturer
- Bar
- Drinker
- License
- Address
- Price
- Bottle size

Relationships

- A relationship connects two or more entity sets.
- It is represented by a diamond, with lines to each of the entity sets involved.
- Relations have
  - A name
  - A set of entities that participate in them
  - A degree
  - A cardinality ratio

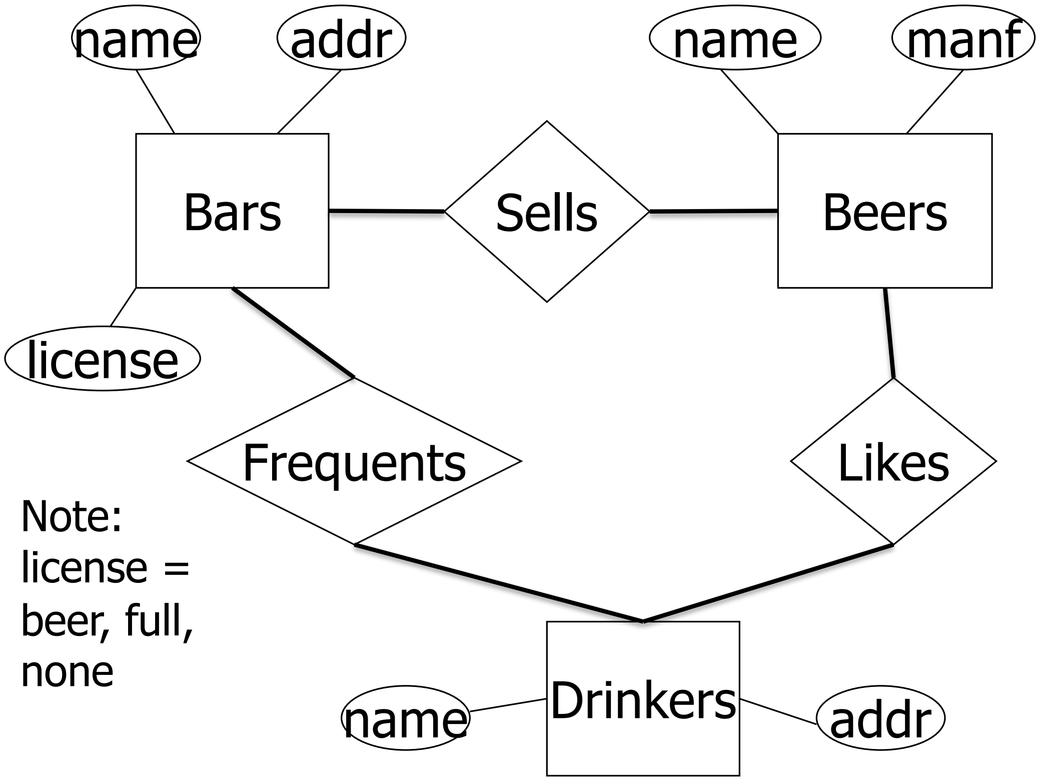
Example: Relationships



Bars sell some beers.

Note:
license = beer, full, none

Example: Relationships



Bars sell some beers.

Drinkers like some beers.

Drinkers frequent some bars.

Note:
license =
beer, full,
none

Relationship Set

- The current “value” of an entity set is the set of entities that belong to it.
  - Example: the set of all bars in our database.
- The “value” of a relationship is a *relationship set*, a set of tuples with one component for each related entity set.

Example: Relationship Set

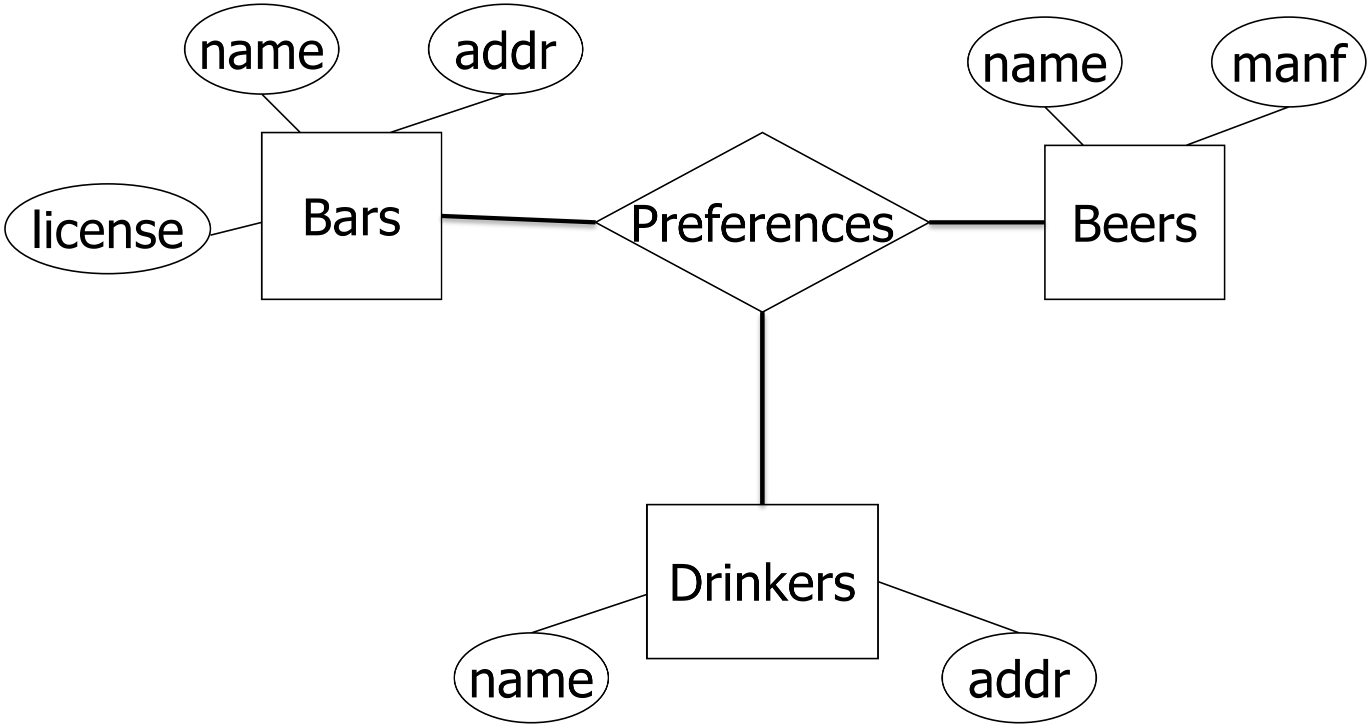
- For the relationship Sells, we might have a relationship set like:

Bar	Beer
Joe’s Bar	Bud
Joe’s Bar	Miller
Sue’s Bar	Bud
Sue’s Bar	Pete’s Ale
Sue’s Bar	Bud Lite

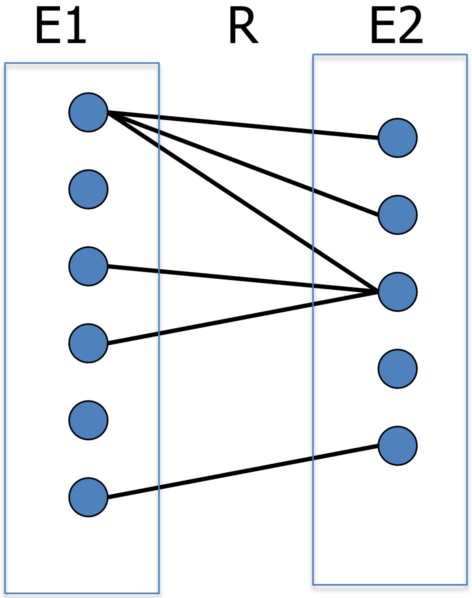
Multiway Relationships

- Sometimes, we need a relationship that connects more than two entity sets.
- Suppose that drinkers will only drink certain beers at certain bars.
  - Our three binary relationships Likes, Sells, and Frequents do not allow us to make this distinction.
  - But a 3-way relationship would.

Example: 3-Way Relationship

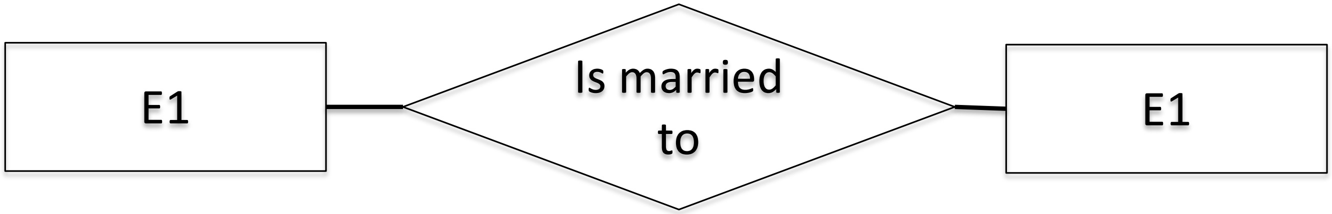


Cardinality



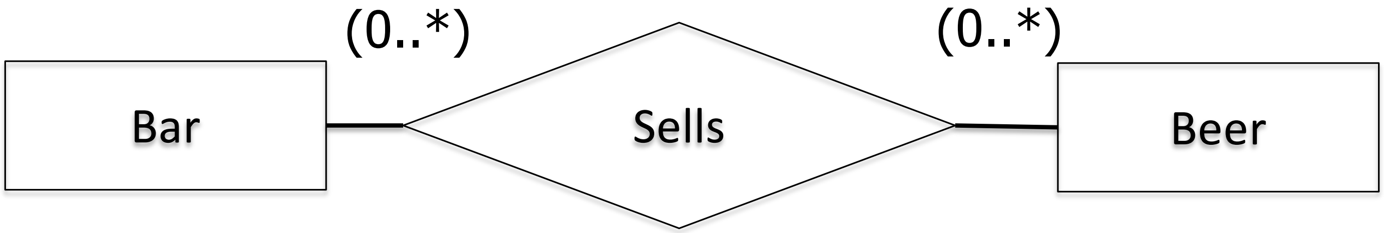
Cardinalities

- No restriction in how often an entity can be in a relationship R
  - Can connect to zero, one, or many
- Mini-world semantics might require more specificity



Many-Many Relationships

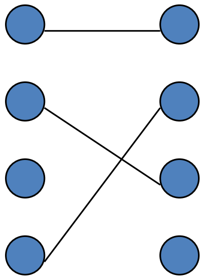
- Focus: binary relationships, such as Sells between Bars and Beers.
- In a *many-many* relationship, an entity of either set can be connected to many entities of the other set.
  - E.g., a bar sells many beers; a beer is sold by many bars.



One-One Relationships

- In a *one-one* relationship, each entity of either entity set is related to at most one entity of the other set.

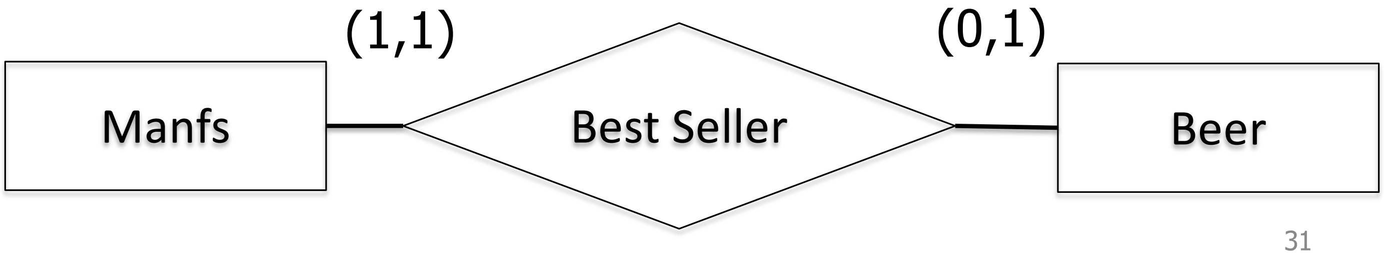
In Pictures:



one-one

Example: Relationship Best-seller between entity sets Manfs (manufacturer) and Beers.

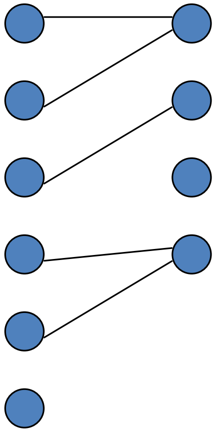
- A beer cannot be made by more than one manufacturer, and no manufacturer can have more than one best-seller (assume no ties).



Many-One Relationships

- Some binary relationships are *many -one* from one entity set to another.
- Each entity of the first set is connected to at most one entity of the second set.
- But an entity of the second set can be connected to zero, one, or many entities of the first set.
- Cardinality of 1 is on the “many” side!

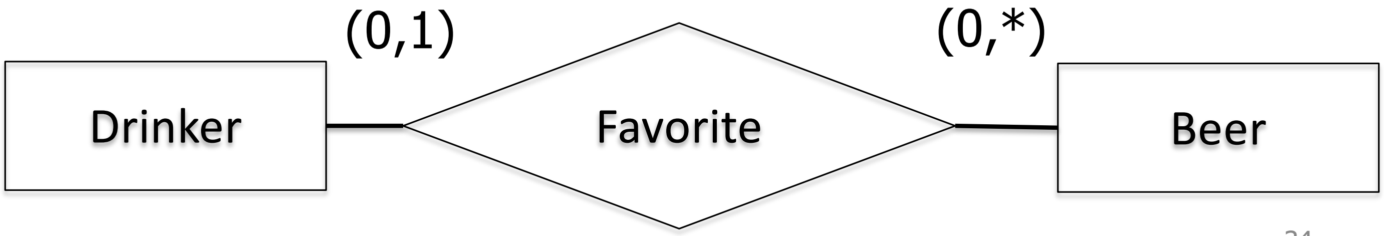
In Pictures:



many-one

Example: Many-One Relationship

- Favorite, from Drinkers to Beers is many-one.
- A drinker has at most one favorite beer.
- But a beer can be the favorite of any number of drinkers, including zero.



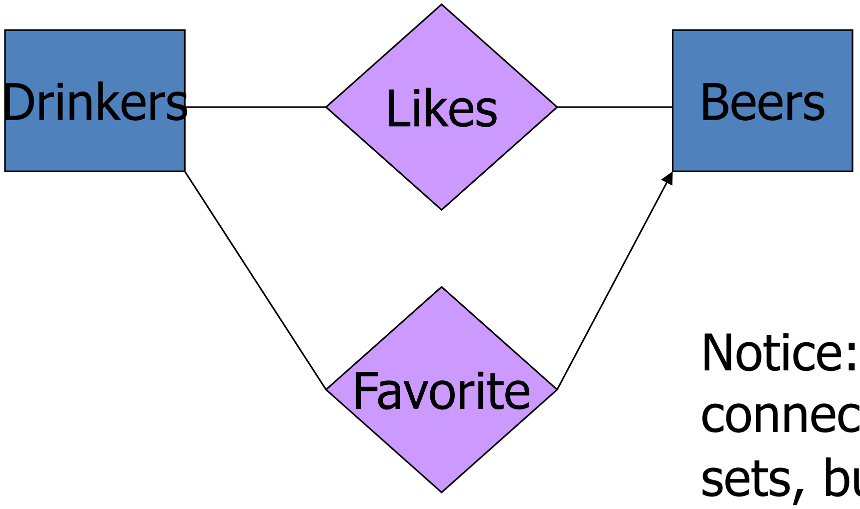
One to Many

- Same idea as many-to-one but from opposite perspective:
  - Each entity in the second set is connected to at most one entity in the first

Representing “Multiplicity”

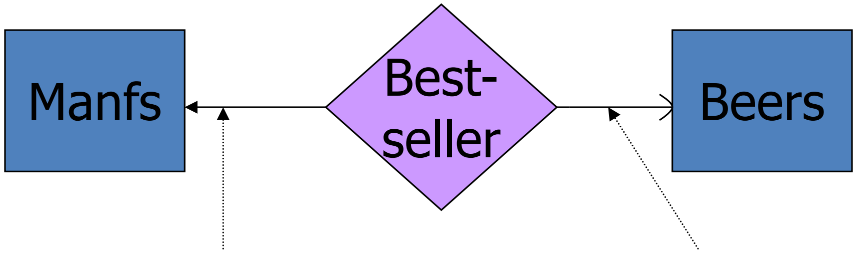
- Show a many-one relationship by an arrow entering the “one” side.
  - Remember: Like a functional dependency.
- Show a one-one relationship by arrows entering both entity sets.
- Rounded arrow = “exactly one,” i.e., each entity of the first set is related to exactly one entity of the target set.

Example: Many-One Relationship



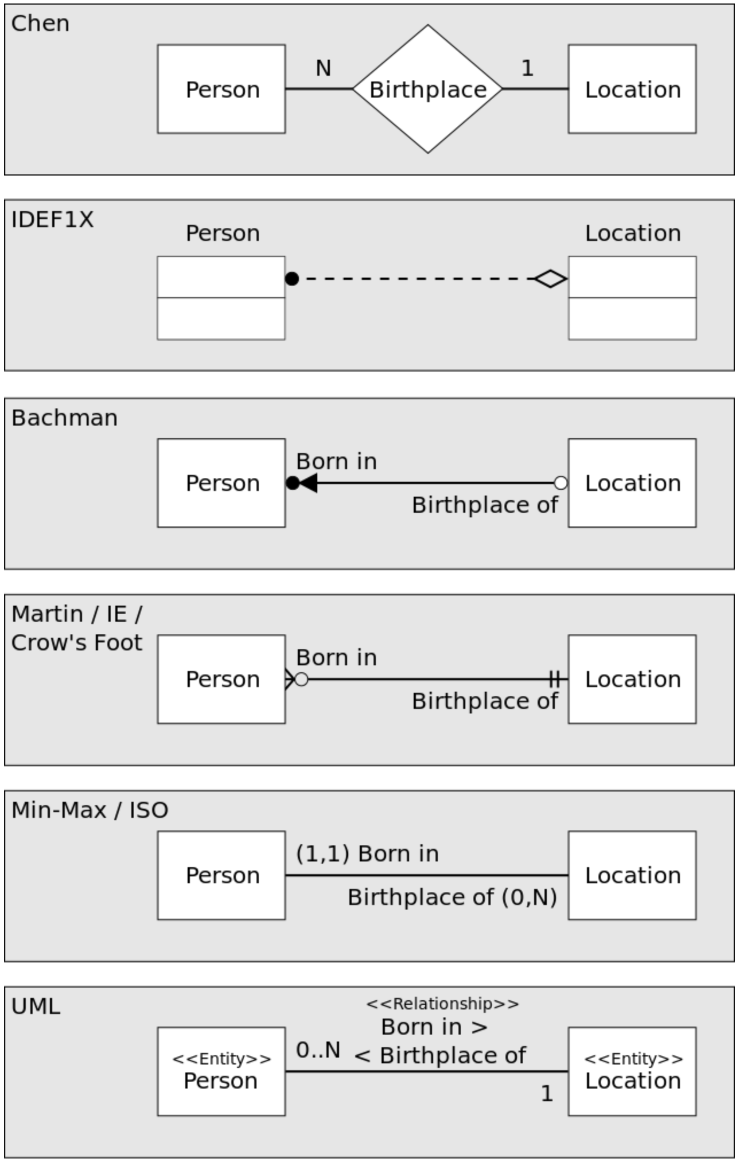
Notice: two relationships connect the same entity sets, but are different.

In the E/R Diagram



A beer is the best-seller for 0 or 1 manufacturer.

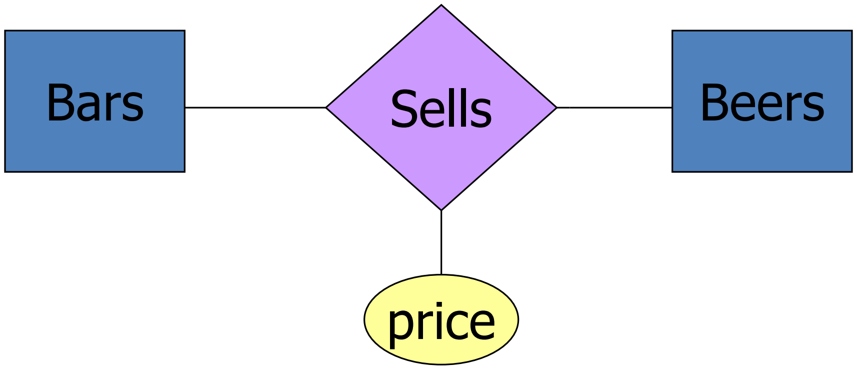
A manufacturer has exactly one best seller.


Chen
Person
N
Birthplace
1
Location
IDEF1X
Person
Location
Bachman
Person
Born in
Birthplace of
Location
Martin / IE /
Crow's Foot
Person
Born in
Birthplace of
Location
Min-Max / ISO
Person
(1,1) Born in
Birthplace of (0,N)
Location
UML
<<Relationship>>
Born in >
< Birthplace of
<<Entity>>
Person
0..N
1
<<Entity>>
Location

Attributes on Relationships

- Sometimes it is useful to attach an attribute to a relationship.
- Think of this attribute as a property of tuples in the relationship set.

Example: Attribute on Relationship

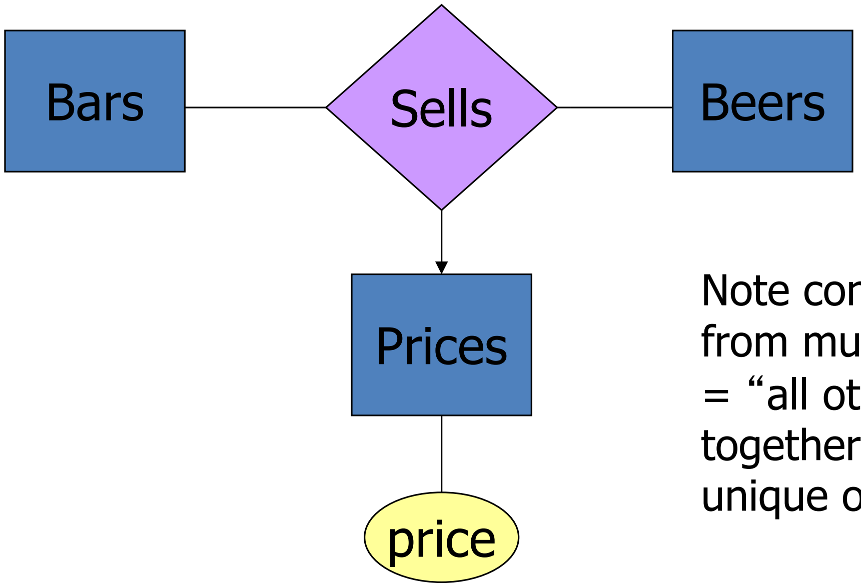


Price is a function of both the bar and the beer,
not of one alone.

Equivalent Diagrams Without Attributes on Relationships

- Create an entity set representing values of the attribute.
- Make that entity set participate in the relationship.

Example: Removing an Attribute from a Relationship

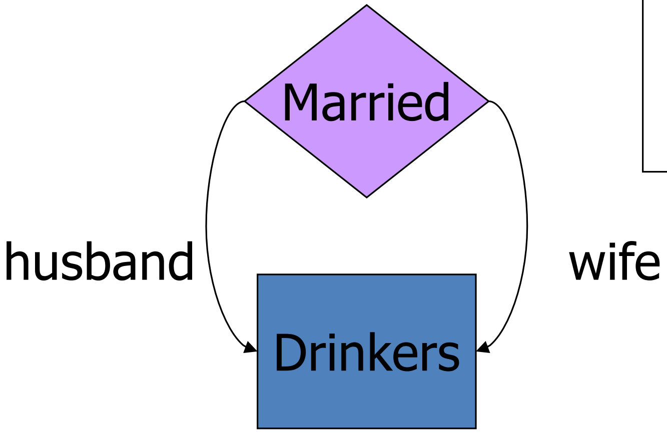


Note convention: arrow from multiway relationship = “all other entity sets together determine a unique one of these.”

Roles

- Sometimes an entity set appears more than once in a relationship.
- Label the edges between the relationship and the entity set with names called *roles*.

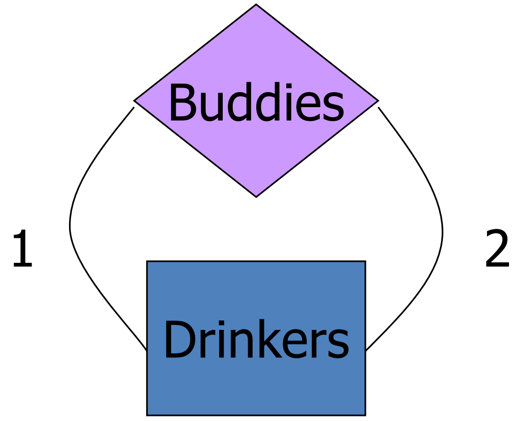
Example: Roles



Relationship Set

Husband	Wife
Bob	Ann
Joe	Sue
…	…

Example: Roles



Relationship Set

Buddy1	Buddy2
Bob	Ann
Joe	Sue
Ann	Bob
Joe	Moe
…	…

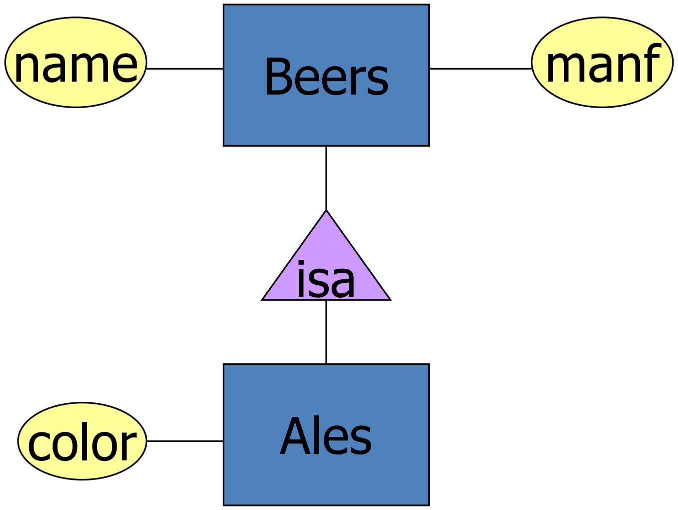
Subclasses

- *Subclass* = special case = fewer entities = more properties.
- Example: Ales are a kind of beer.
  - Not every beer is an ale, but some are.
  - Let us suppose that in addition to all the *properties* (attributes and relationships) of beers, ales also have the attribute color.

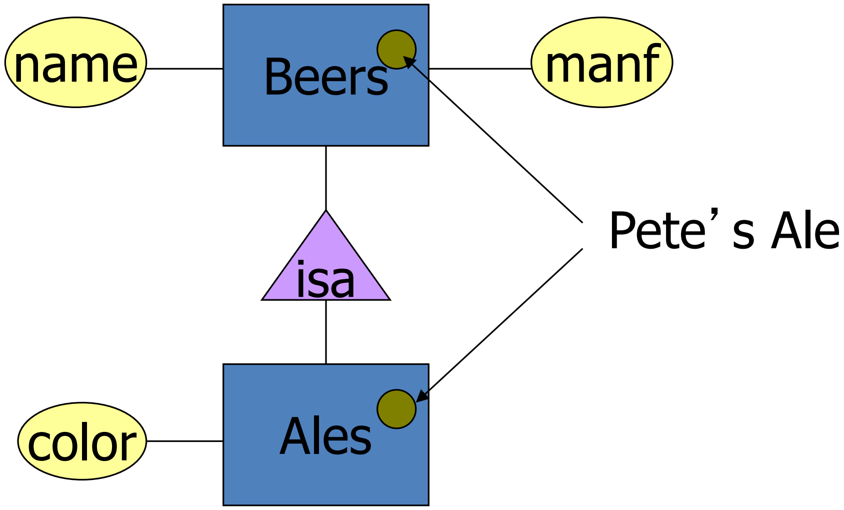
Subclasses in E/R Diagrams

- Assume subclasses form a tree.
  - I.e., no multiple inheritance.
- Isa triangles indicate the subclass relationship.
  - Point to the superclass.

Example: Subclasses



Example: Representatives of Entities



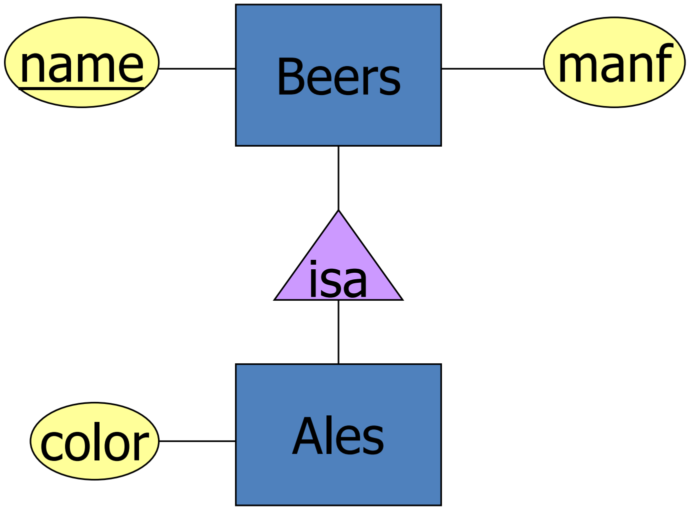
Keys

- A key is a set of attributes for one entity set such that no two entities in this set have the same value for the key.
  - It is allowed for two entities to agree on some, but not all, of the key attributes.
- We must designate a key for every entity set.

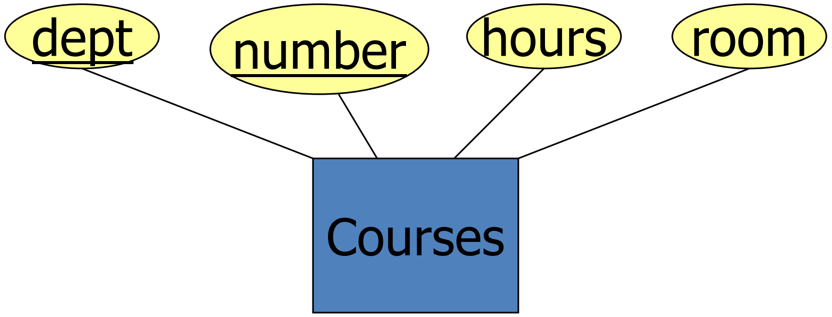
Keys in E/R Diagrams

- Underline the key attribute(s).
- In an Isa hierarchy, only the root entity set has a key, and it must serve as the key for all entities in the hierarchy.

Example: name is Key for Beers



Example: a Multi-attribute Key



- Note that hours and room could also serve as a key, but we must select only one key.

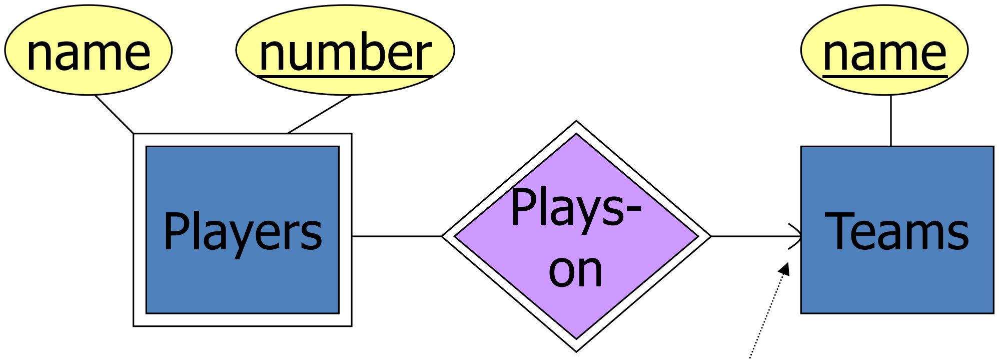
Weak Entity Sets

- Occasionally, entities of an entity set need “help” to identify them uniquely.
  - If you have a many to one relationship with another entity, you can use attributes of that entity to form a key
- We call this entity set *weak*

Example: Weak Entity Set

- name is almost a key for football players, but there might be two with the same name.
- number is certainly not a key, since players on two teams could have the same number.
- But number, together with the team name related to the player by Plays-on should be unique.

In E/R Diagrams



Note: must be rounded because each player needs a team to help with the key.

- Double diamond for *supporting* many-one relationship.
- Double rectangle for the weak entity set.

Weak Entity-Set Rules

- A weak entity set has one or more many-one relationships to other (supporting) entity sets.
  - Not every many-one relationship from a weak entity set need be supporting.
  - But supporting relationships must have a rounded arrow (entity at the “one” end is guaranteed).

Weak Entity-Set Rules – (2)

- The key for a weak entity set is its own underlined attributes and the keys for the supporting entity sets.
  - E.g., (player) number and (team) name is a key for Players in the previous example.

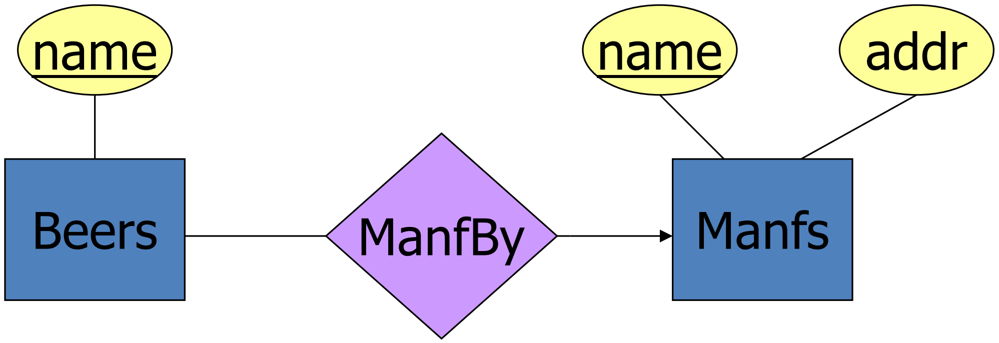
Design Techniques

1. Avoid redundancy.
2. Limit the use of weak entity sets.
3. Don’t use an entity set when an attribute will do.

Avoiding Redundancy

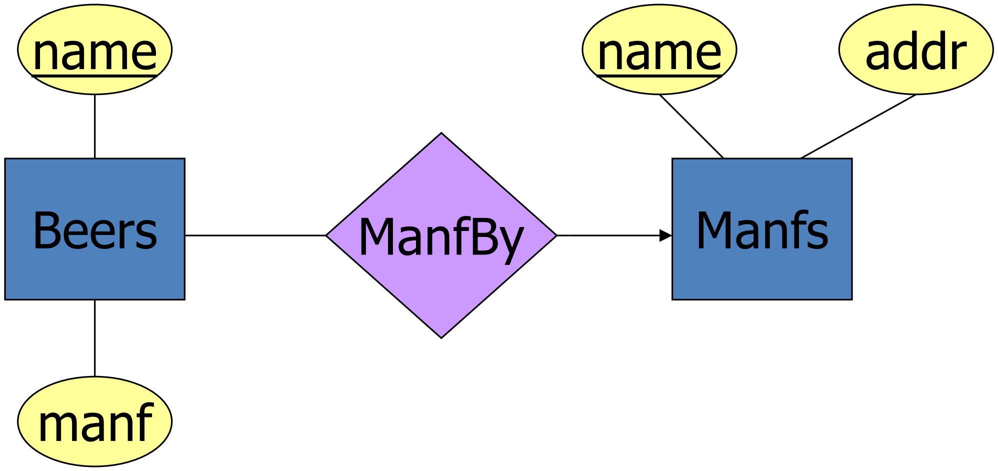
- *Redundancy* = saying the same thing in two (or more) different ways.
- Wastes space and (more importantly) encourages inconsistency.
  - Two representations of the same fact become inconsistent if we change one and forget to change the other.
  - Recall anomalies due to FD's.

Example: Good



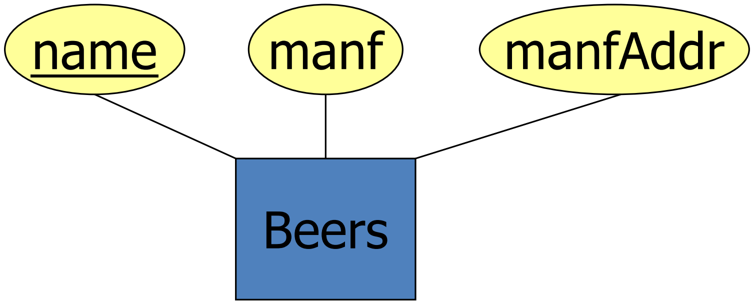
This design gives the address of each manufacturer exactly once.

Example: Bad



This design states the manufacturer of a beer twice: as an attribute and as a related entity.

Example: Bad



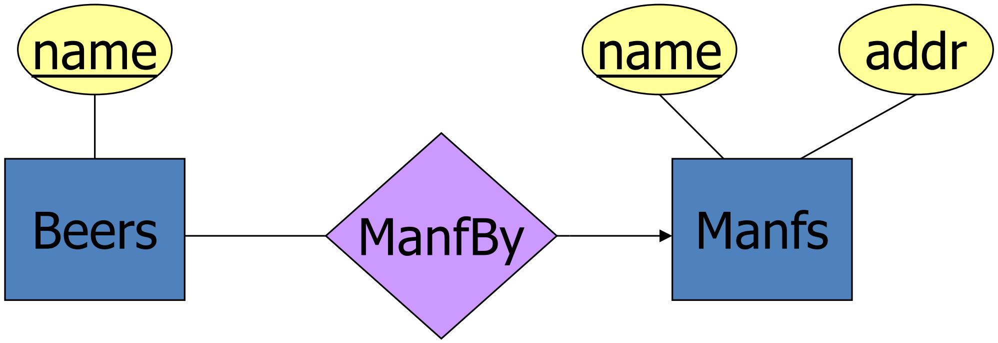
This design repeats the manufacturer's address once for each beer and loses the address if there are temporarily no beers for a manufacturer.

Entity Sets Versus Attributes

- An entity set should satisfy at least one of the following conditions:
  - It is more than the name of something; it has at least one nonkey attribute.

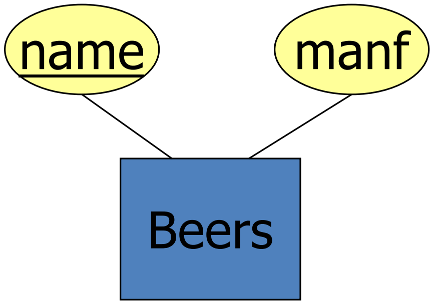
    or
  - It is the “many” in a many-one or many-many relationship.

Example: Good



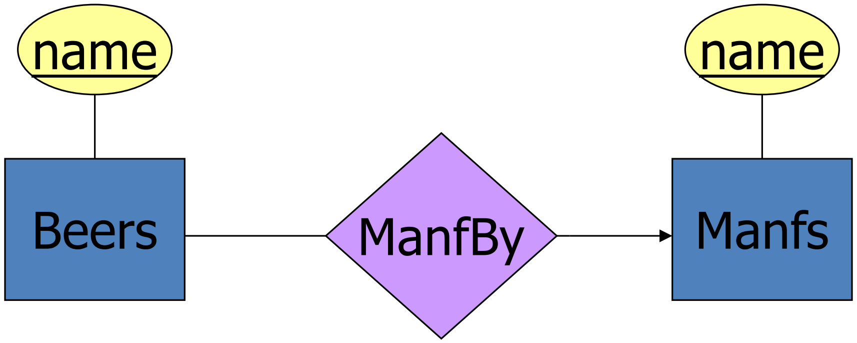
- Manfs deserves to be an entity set because of the nonkey attribute addr.
- Beers deserves to be an entity set because it is the "many" of the many-one relationship ManfBy.

Example: Good



There is no need to make the manufacturer an entity set, because we record nothing about manufacturers besides their name.

Example: Bad



Since the manufacturer is nothing but a name, and is not at the "many" end of any relationship, it should not be an entity set.

Don’t Overuse Weak Entity Sets

- Beginning database designers often doubt that anything could be a key by itself.
  - They make all entity sets weak, supported by all other entity sets to which they are linked.
- In reality, we usually create unique ID’s for entity sets.
  - Examples include social-security numbers, automobile VIN’s etc.

When Do We Need Weak Entity Sets?

- The usual reason is that there is no global authority capable of creating unique ID’s.
- Example: it is unlikely that there could be an agreement to assign unique player numbers across all football teams in the world.

Next up….

- Relational databases