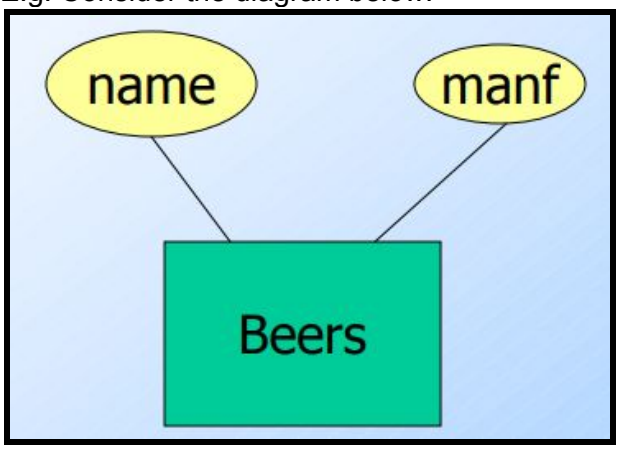
Introduction to Entity-Relationship Models:

- An **ER model** is a high-level data model. This model is used to define the data elements and relationship for a specified system.
- An ER model describes the structure of a database with the help of an **entity-relationship diagram (ER diagram)**. An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of a ER model are the **entity set** and **relationship set**.
- The purpose of an ER model is that it allows us to sketch database schema designs, called ER diagrams.
  Note: ER models may include some constraints, but not operations.
- ER models are very useful in planning and communicating database schemas. Sketching the key components is an efficient way to develop a working database.
- Later, we can convert ER models to relational database designs.

Introduction to Entities:

- **Entity:** A real-world thing which can be distinctly identified. It is an object which is distinguishable from others. In a table, an entity is a tuple.
  E.g. If we have the table Student(SID, First_Name, Last_Name) then each student in that table is an entity and can be uniquely identified by their SID.
- **Entity Set:** A collection of similar entities.
  The current value of an entity set is the set of entities that belong to it.
  In the ER diagram, an entity set is represented as a rectangle.
- **Attribute:** A property of the entities of an entity set. In a table, an attribute is a column. Note that attributes are simple values such integers or character strings. They are not structs, sets, etc.
  In the ER diagram, an attribute is represented as an oval, with a line to the rectangle representing its entity set.
- E.g. Consider the diagram below:



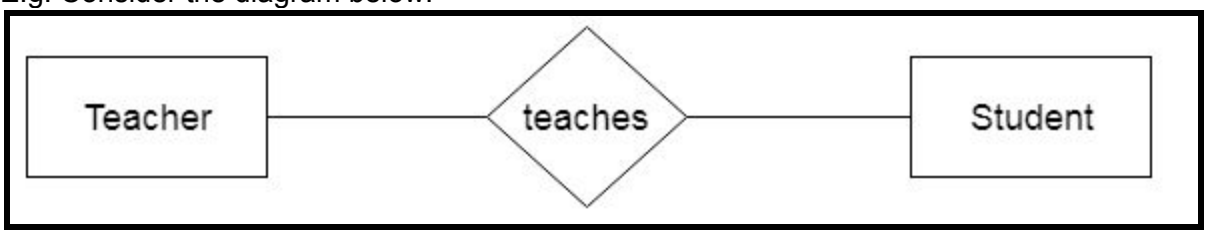
We see that:

- Beers is an entity set, because it's denoted by a rectangle.
- name and manf are attributes of Beers because they are denoted by an oval and they are connected to Beers by a line.

Each Beers entity has values for these two attributes.

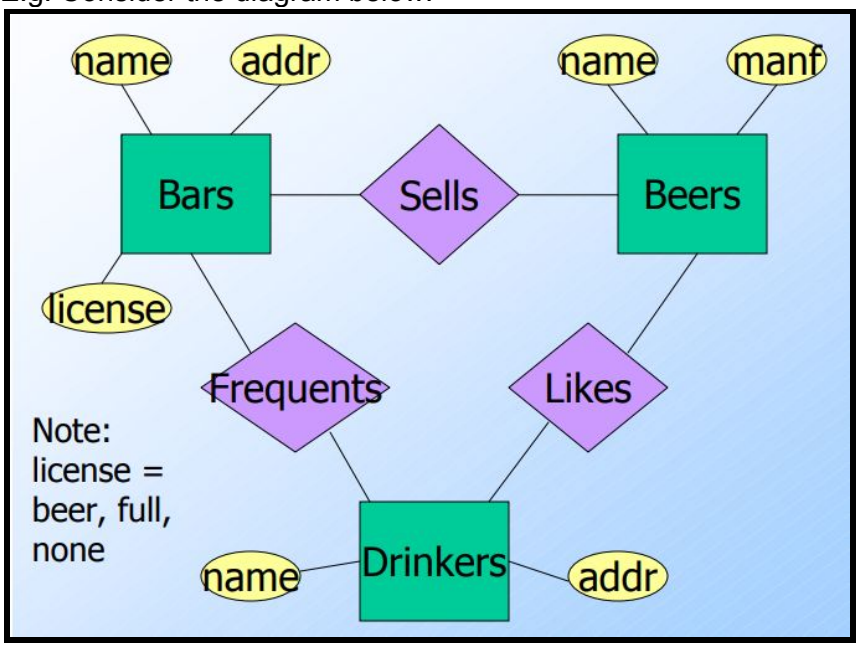
Introduction to Relationships:

- A **relationship** connects two or more entity sets. It is represented by a diamond, with lines to each of the entity sets involved.
- A relationship between two entities signifies that the two entities are associated with each other. Think of relationships as joins.
- E.g. Consider the diagram below:



We can see that the Teacher and Student entity sets are connected by the relationship teaches.

- E.g. Consider the diagram below:



We can see that:

- Bars sell some beers.
- Drinkers like some beers.
- Drinkers frequent some bars.

- The value of a relationship is a **relationship set**, which is a set of tuples with one component for each related entity set.

- E.g. Consider the Sells relationship from above. The below picture is a possible relationship set.

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

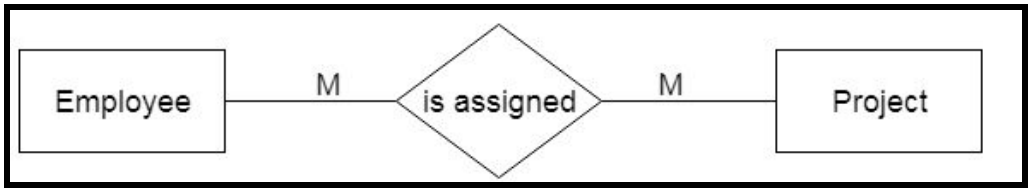
- The **degree of a relationship set** is the number of different entity sets participating in a relationship set.
    - When there is only one entity set participating in a relation, the relationship is called a **unary relationship**.
    - When there are two entities set participating in a relation, the relationship is called a **binary relationship**.
    - When there are n entities set participating in a relation, the relationship is called a **n-ary relationship**.

Cardinality of Relationships:

- **Cardinality** specifies how many instances of an entity relate to one instance of another entity.
  There are 4 types of cardinality:
    1. Many-to-Many
    2. Many-to-One
    3. One-to-One
    4. One-to-Many
- In a **many-many relationship**, an entity of either set can be connected to many entities of the other set.
  I.e. A many-to-many relationship refers to the relationship between two entities X and Y in which X may be linked to many instances of Y and vice versa.

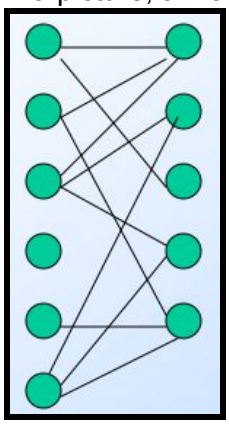
  E.g. Sells is a many-to-many relationship because a bar sells many beers and a beer is sold by many bars.

  E.g. Consider the diagram below:



Is assigned is a many-to-many relationship because an employee can be assigned many projects and a project can have many employees working on it.

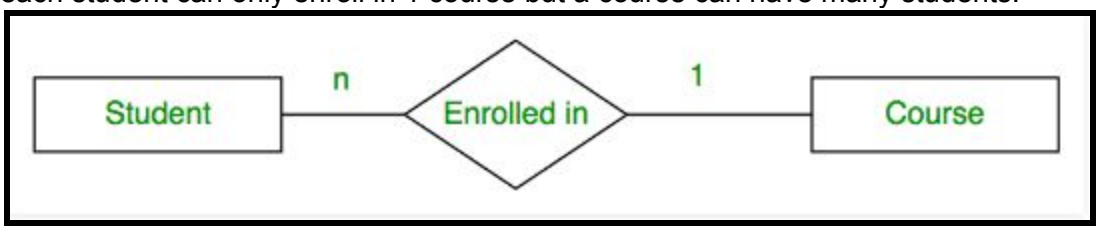
In a picture, a many-to-many relationship looks like:



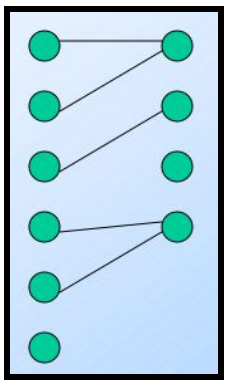
- A **many-to-one relationship** occurs when more than one instance of the entity on the left and only one instance of an entity on the right associates with the relationship.
  Note: In a many-to-one relationship, each entity of the first set is connected to at most one entity of the second set but an entity of the second set can be connected to zero, one, or many entities of the first set.

E.g. Suppose there is a Favourites relationship between entity sets Drinkers and Beers. Favourites is a many-to-one relationship because a drinker can only have 1 favourite beer but a beer can be the favourite of many drinkers.

E.g. Assume that at UTSC, each student can enroll in 1 course at most.
Then, the Enrolled in relationship shown below is a many-to-one relationship because each student can only enroll in 1 course but a course can have many students.



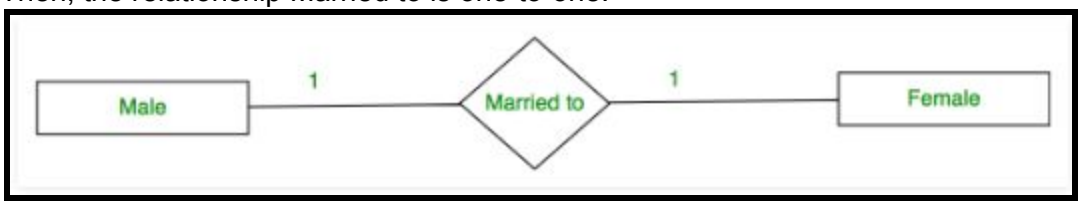
In a picture, a many-to-one relationship looks like:



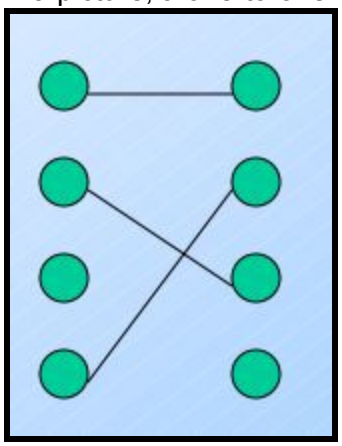
- A **one-to-one relationship** occurs when each entity in each entity set can take part only once in the relationship.
  I.e. In a one-one relationship, each entity of either entity set is related to at most one entity of the other set.

  E.g. Suppose we have a relationship Best-Seller between Manufacturer and Beers. Best-Seller is a one-to-one relationship because a manufacturer can only have 1 best selling beer (assume no ties) and a beer can only be made by 1 manufacturer.

  E.g. Assume that a male can marry to one female and a female can marry to one male. Then, the relationship Married to is one-to-one.

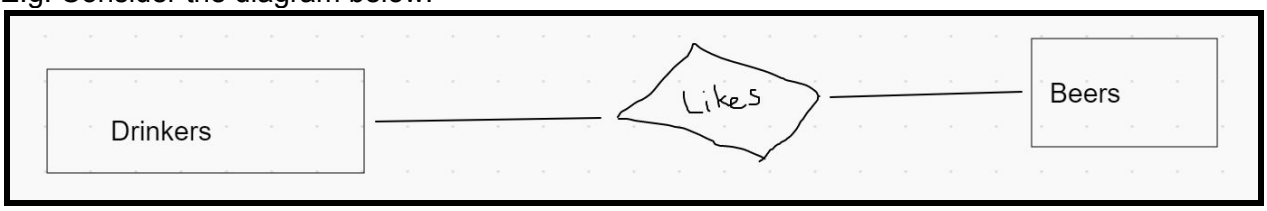


  In a picture, a one-to-one relationship looks like:



- We can represent these relationships in the ER diagram by using various lines.
  - We can show a many-to-one relationship by an arrow entering the “one” side.
  - We can show a one-to-one relationship by arrows entering both entity sets.
  - A rounded arrow means exactly one.
    I.e. Each entity of the first set is related to exactly one entity of the target set.

  E.g. Consider the diagram below.



  This is a many-to-many diagram. We can tell because there’s no arrow going into either Drinkers or Beers.

To interpret it, start from one entity set and follow the line to the second entity set. Then, start from the second entity set and follow the line to the first entity set.

We can interpret this as:

- A drinker likes some beers.
  (We started from drinker and followed the arrow to beers.)
- A beer is liked by some drinkers.
  (We started from beer and followed the arrow to drinker.)

E.g. Consider the diagram below.

This is a many-to-one diagram. We can tell because there's an arrow going into Beers while there's no arrow going into Drinkers.

We can interpret this as:

1. A drinker has at most 1 favourite beer.
2. A beer is the favourite of some drinkers.

Note: In the first 2 examples, two relationships are connecting the same entity sets, but are different.

E.g. Consider the diagram below.

Note: The arrow going to Beers is a rounded arrow, meaning exactly one.

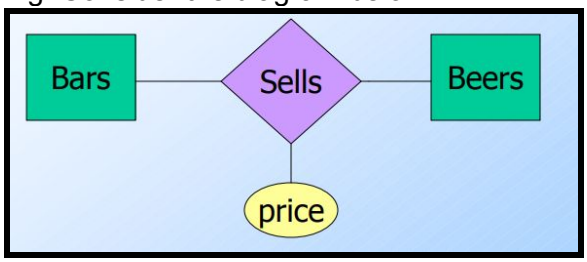
We can interpret this as:

1. A manufacturer has exactly one best-selling beer.
2. A beer is the best seller of at most one manufacturer.

Attributes on Relationships:

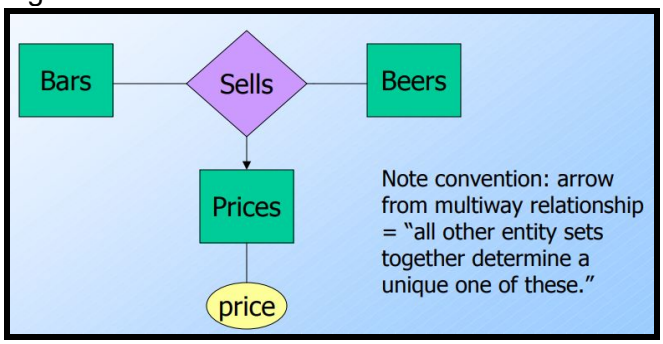
- Sometimes it is useful to attach an attribute to a relationship.
- We can think of this attribute as a property of tuples in the relationship set.

- E.g. Consider the diagram below.



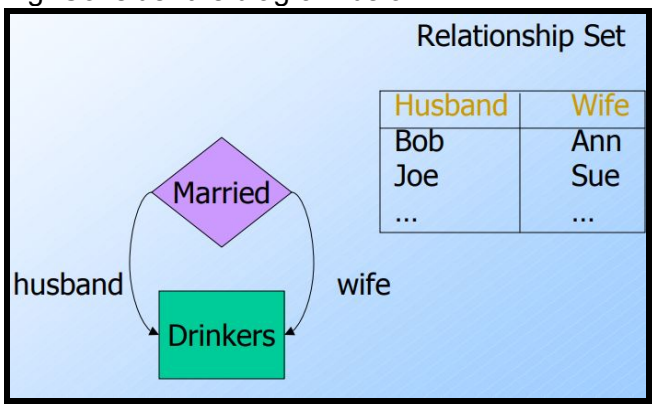
Price is a function of both the bar and the beer, not of one alone.

- An equivalent way of showing the diagram but without having attributes on relationships is to create an entity set representing values of the attribute and make that entity set participate in the relationship.
- E.g.



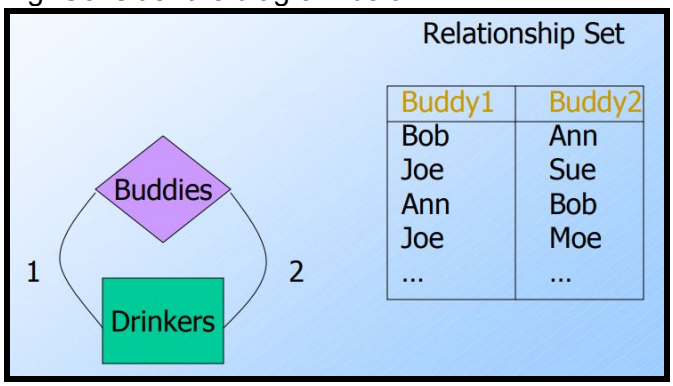
Roles:

- Entity sets of a relationship do not need to be distinct and sometimes an entity set appears more than once in a relationship.
- To show this on an ER diagram, we label the edges between the relationship and the entity set with names called **roles**.
- Roles are indicated in E-R diagrams by labeling the lines that connect diamonds to rectangles.
- Role labels are optional, and are used to clarify semantics of the relationship.
- E.g. Consider the diagram below:



In this example, husband and wife are the roles.

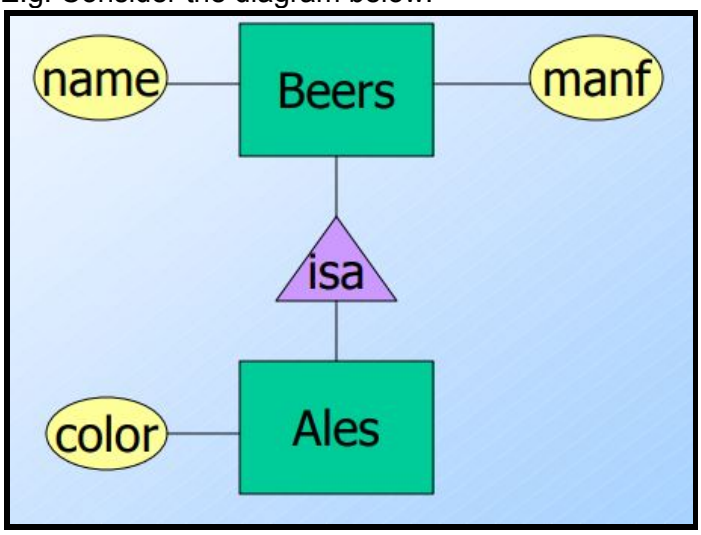
- E.g. Consider the diagram below.



The roles in this example are buddy1 and buddy2.

Subclasses:

- An entity set may contain entities that have special properties not associated with all members of the set. These entities are called **subclasses**.
  I.e. A subclass is a subgroup of entities with special properties.
- E.g. Ales are a kind of beer. Not every beer is an ale, but some are.
- Assume subclasses form a tree in the ER diagram.
  I.e. There's no multiple inheritance.
- Isa triangles indicate the subclass relationship. They always point to the superclass.
- E.g. Consider the diagram below.

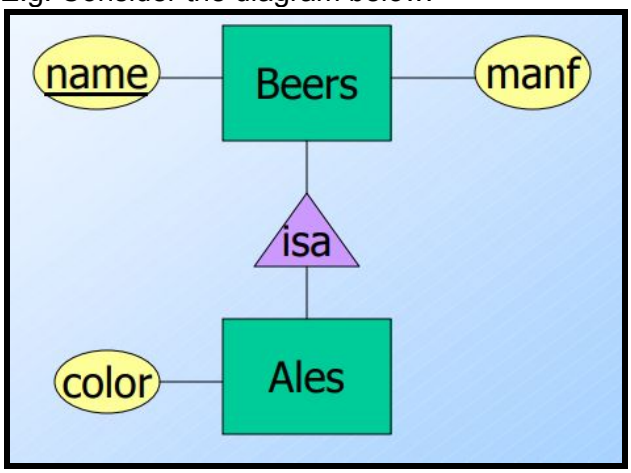


- ER entities have representatives in all subclasses to which they belong.
  Rule: If entity e is represented in a subclass, then e is represented in the superclass and recursively up the tree.

Keys:

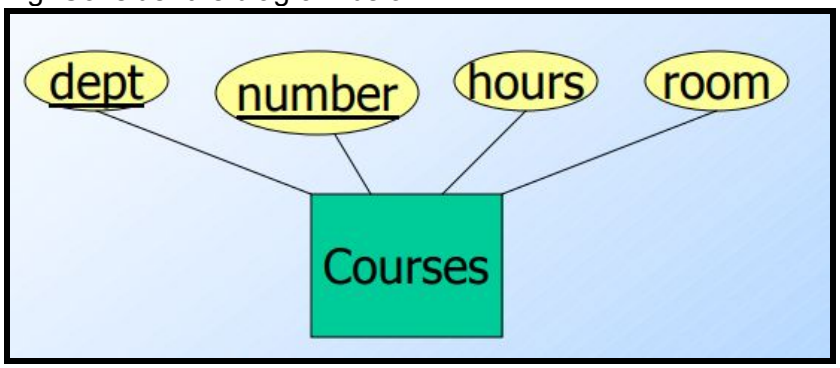
- A **key** is a set of attributes for one entity set such that no two entities in this set agree on all the attributes of the key.
- **Note:** It is allowed for two entities to agree on some, but not all, of the key attributes.
- We must designate a key for every entity set.

- There could be multiple keys, but we choose one to use.
- To represent a key in an ER diagram, we underline the attribute(s) that make up the key.
- 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.
- E.g. Consider the diagram below.



In this case, name is key for Beers and for Ales because Ales is a subclass of Beers.

- E.g. Consider the diagram below.

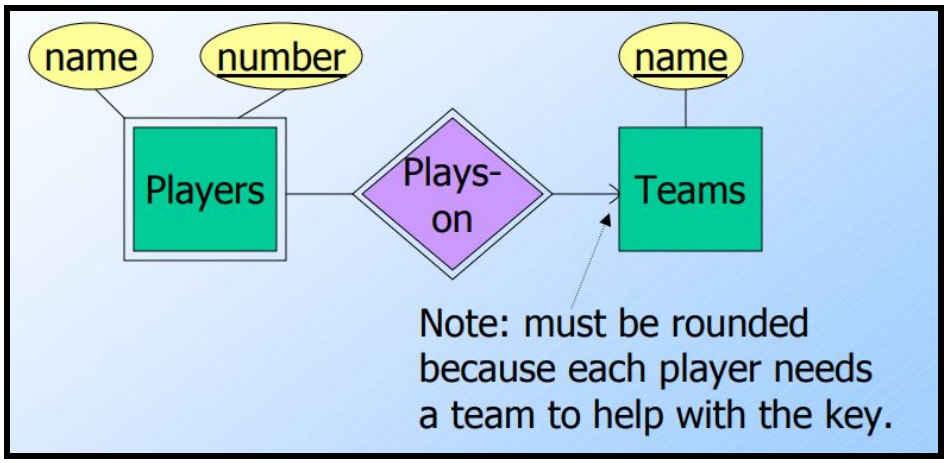


Here, we see that dept and number form the key.
Note that hours and room could also serve as a key, but we must select only one key.

Weak Entity Set:

- A **weak entity** is an entity that depends on another entity.
- Entity set E is said to be weak if in order to identify entities of E uniquely, we need to follow one or more manyone relationships from E and include the key of the related entities from the connected entity sets.
- An entity set that does not have a primary key is referred to as a weak entity set.
  I.e. A weak entity set does not have a primary key.
- The weak entity is represented by a double rectangle in an ER diagram.
- The relationships connecting a weak entity set to a strong entity set is represented by a double diamond in an ER diagram.
- E.g.
  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.

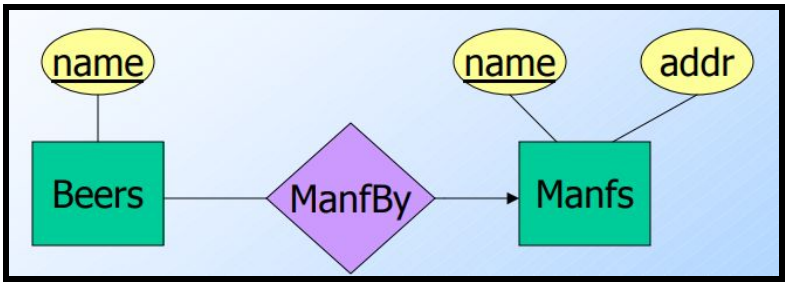
But number, together with the team name related to the player by Plays-on should be unique.
I.e.



- Weak entity set rules:
    1. A weak entity set has one or more many-one relationships to other supporting entity sets.
        - **Note:** Not every many-one relationship from a weak entity set needs to be supporting.
        - All supporting relationships must have a rounded arrow pointing towards its “one” end.
    2. The key for a weak entity set is its own underlined attributes and the keys for the supporting entity sets.

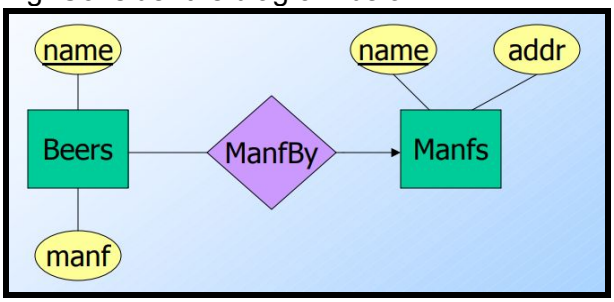
Design Techniques:

- **Avoid redundancy:**
    - Redundancy is saying the same thing in two or more different ways.
    - It 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 the anomalies from function dependencies.
    - E.g. Consider the diagram below.



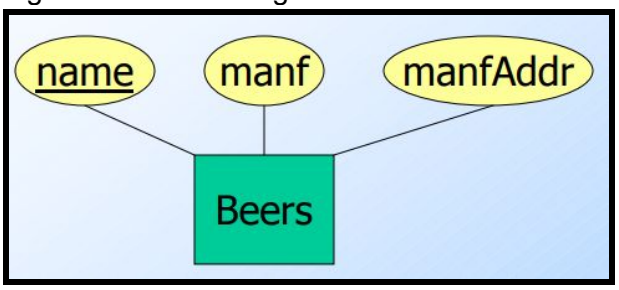
This is an example of good design because there is no duplicate information.

- E.g. Consider the diagram below.



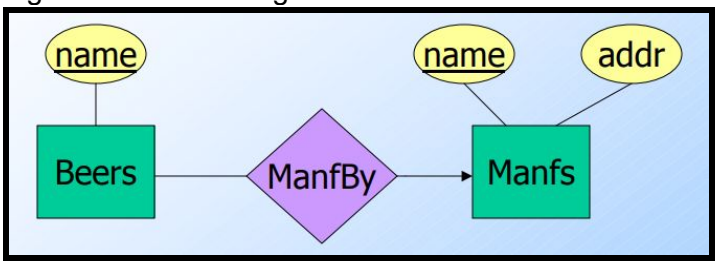
This is an example of bad design because the manf information is duplicated. This design states the manufacturer of a beer twice: once as an attribute and second as a related entity.

- E.g. Consider the diagram below.



This is an example of bad design because this design repeats the manufacturer's address once for each beer. Suppose that a manufacturer temporarily stopped producing beers. We would lose its address.

- **Limit the use of weak entity sets:**
  - We use weak entity sets if there is no global authority capable of creating unique IDs.
  - E.g. It is unlikely that there could be an agreement to assign unique player numbers across all football teams in the world.
- **Don't use an entity set when an attribute will do:**
  - An entity set should satisfy at least one of the following conditions:
    1. It is more than the name of something. It has at least one non-key attribute.
    2. It is the "many" in a many-one or many-many relationship.
  - E.g. Consider the diagram below.

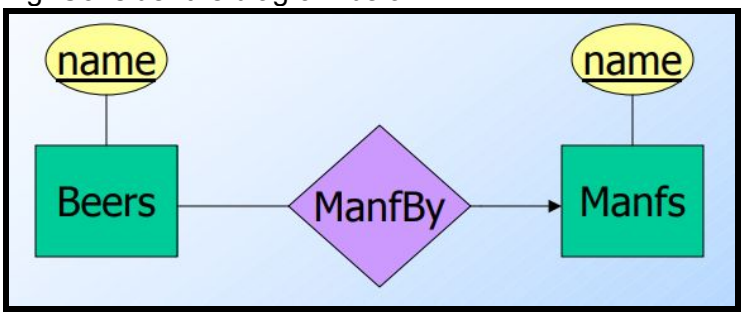


This is an example of good design because:

1. Manfs deserves to be an entity set because of the nonkey attribute addr.

2. Beers deserves to be an entity set because it is the “many” of the many-one relationship ManfBy.

- E.g. Consider the diagram below.



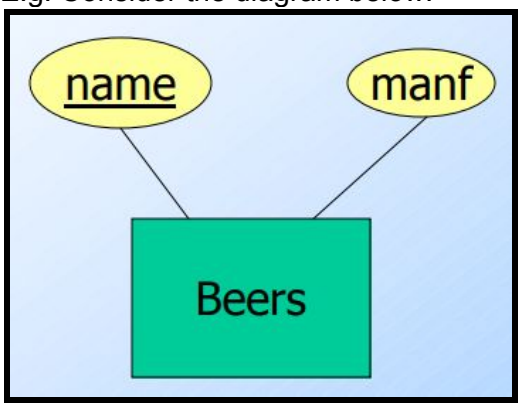
This design is bad because:

1. Manfs is just a name.
2. Manfs is not at the “many” end.

Hence, Manfs should not be an entity set.

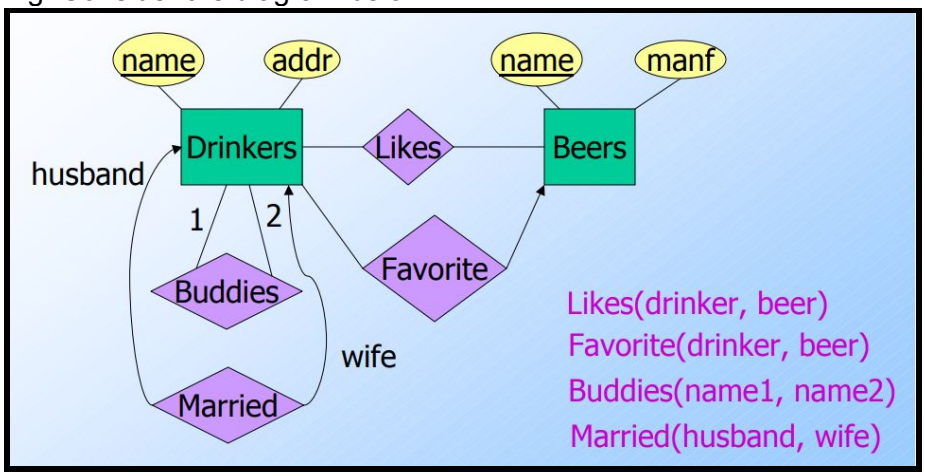
ER Diagrams to Relations:

- Entity set → relation.
- Attributes → attributes.
- Relationships → relations whose attributes are only:
    1. The keys of the connected entity sets. Or
    2. Attributes of the relationship itself.
- E.g. Consider the diagram below.



The corresponding relation is Beers(name, manf).

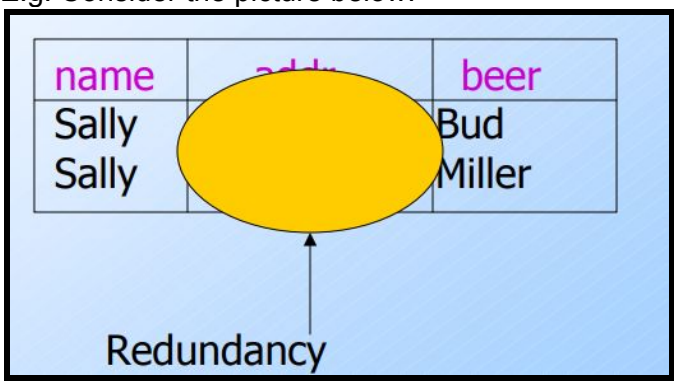
- E.g. Consider the diagram below.



The relations we get from the picture are:

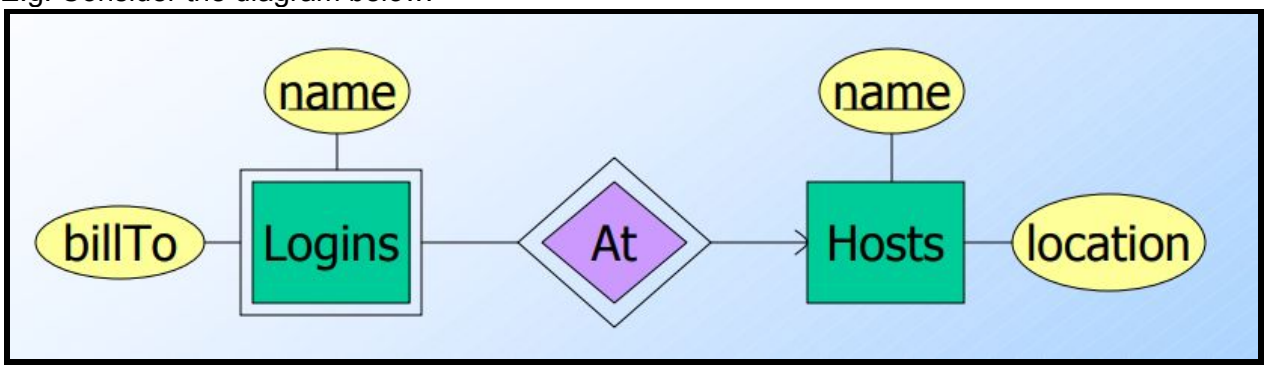
1. Drinkers(name, addr)
2. Beers(name, manf)
3. Married(husband, wife)
4. Buddies(name1, name2)
5. Likes(drinker, beer)
6. Favorite(drinker, beer)

- We can also combine relations. It is ok to combine the following 2 relations into one relation:
    1. The relation for an entity-set E
    2. The relations for many-one relationships of which E is the "many."
- E.g.
  Drinkers(name, addr) and Favorite(drinker, beer) combine to make Drinker1(name, addr, favBeer).
- **Note:** The reason why we don't combine many-to-many relationships is because it could lead to redundancy.
- E.g. Consider the picture below.



We see that Sally's address is used twice.

- We can turn a weak entity set into a relation too.
- A relation for a weak entity set must include attributes for its complete key including those belonging to other entity sets, as well as its own, nonkey attributes.
- A supporting relationship is redundant and yields no relation unless it has attributes.
- E.g. Consider the diagram below.



We have 2 relations:

1. Hosts(<u>hostName</u>, location)
2. Logins(<u>loginName</u>, <u>hostName</u>, billTo)

We don't need a relation for At as it does not have any attributes.