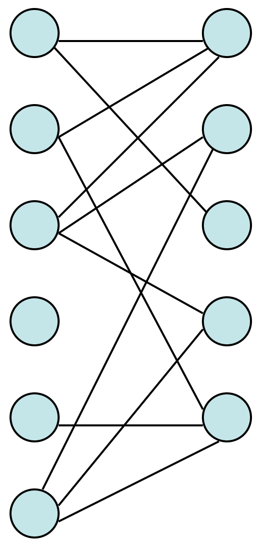
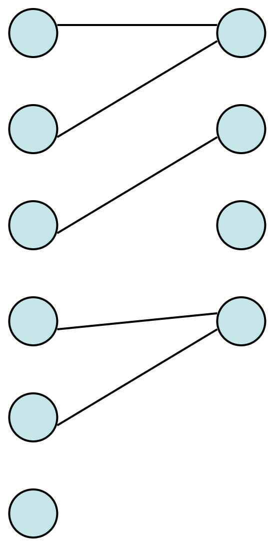
Relationship arity

- We focus for now on binary relationships
- Many-to-many relationships
  - In these relationships, an entity of either set can be connected to many entities of the other set
  - Example: In the **Sells** relation, a **pub sells many beers**, and a **beer is sold in many pubs**
- Many-to-one relationships
  - Each entity in the first set of the relationship is connected to at most one entity of the second set.
  - However, an entity of the second set can be connected to zero, one, or many entities of the first set
  - Example: in the **Favorite** relationship (from **Patrons** to **Beer**), a **patron has at most one favorite beer**, but the **beer can be the favorite of zero, one, or many patrons**.

Visuals...



many-many

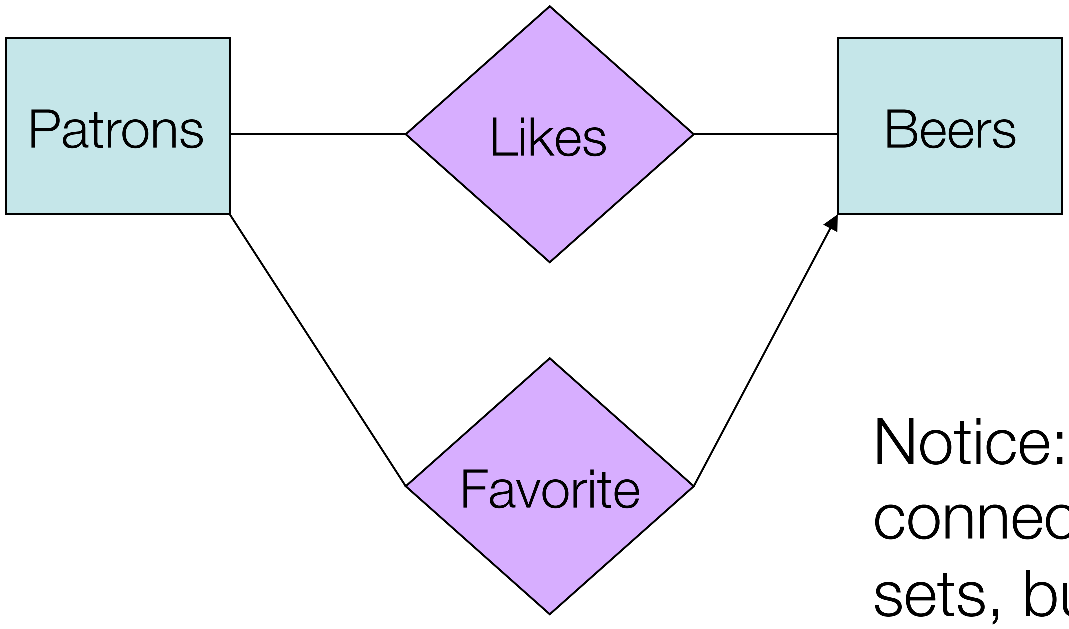


many-one

Multiplicity: notation

- Many-to-many and many-to-one are different forms of multiplicity.
- To denote a **many-to-one relationship**:
  - We connect an entity to a relationship with an **arrow on the "one" side**
  - Tip: We can use functional dependencies – and the direction of the arrows – to discover where the arrow must go in an E/R diagram
- To denote a **one-to-one relationship**:
  - **Arrows enter both entity sets**
- To denote an **exactly one** arity:
  - That is, each entity of the first set is related to exactly one entity of the target set.
  - **Use a hollow arrow** into the entity

Example: Many-One Relationship

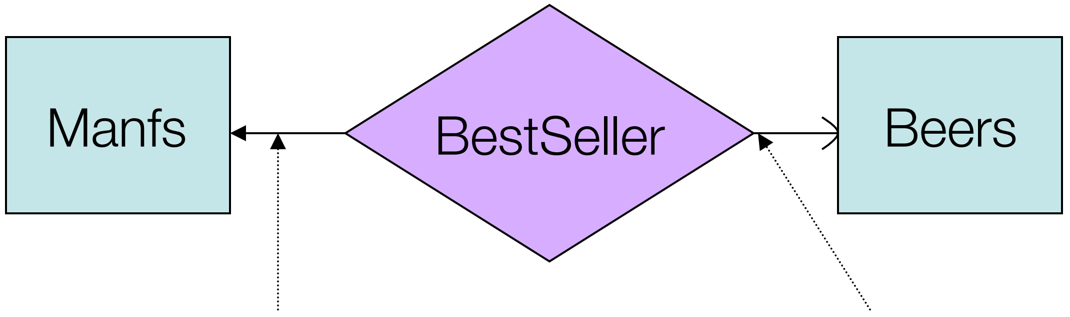


Notice: Two relationships connect the same entity sets, but can result in different relationship sets.

Example: One-to-one relationship

- Consider a **BestSeller** relationship
  - It relates the entities Manfs and Beers
- Some beers are not the best seller of any manufacturer
  - As an arrow, we would choose the filled arrow (and not the hollow arrow) to connect to Manf
- However, a beer manufacturer must have a best seller, however odd it might seem
  - For this we can use the "exactly one" notation of the hollow arrow to connect to Beers

In the E/R Diagram



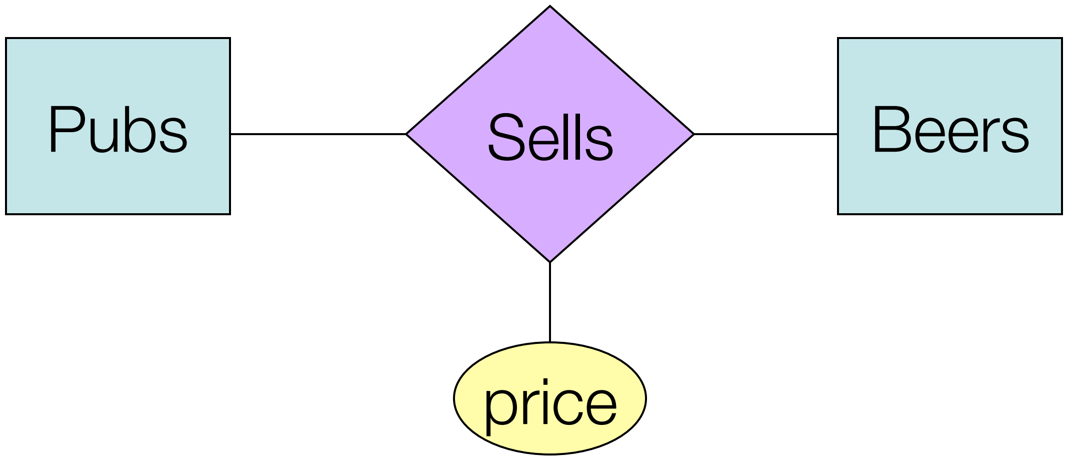
A beer is the best-seller for 0 or 1 manufacturer.

A manufacturer has exactly one best seller.

Attributes on relationships

- Sometimes data which we want to have associated with entities involved a relationship does not belong with either entity
- E/R permits us to attach an attribute to the relationship instead
- This is the same as adding the attribute to the tuples stored in the relationship set.
- Example:
  - Pubs relate to Beers via the Sells relation
  - Those beers are sold at a specific price
  - However, it would be awkward to place the "price" attribute into either Pubs or Beers
  - Instead we place it with Sells

Example: Attribute on Relationship

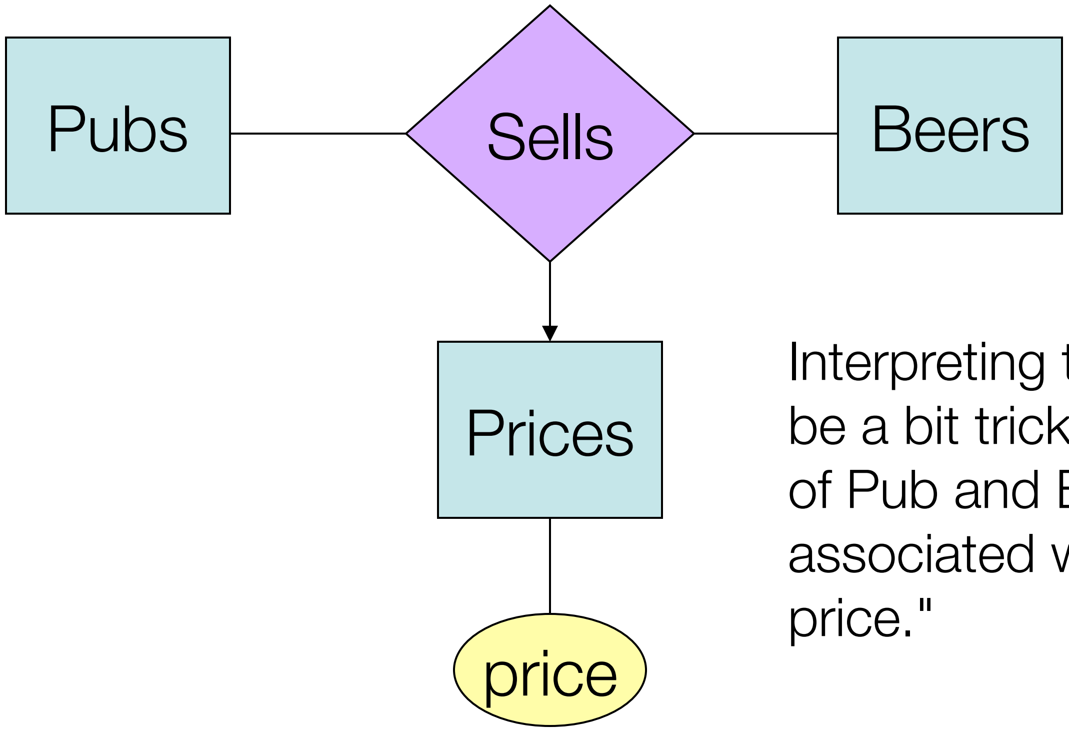


Price is a function of both the **pub** and the **beer**, but not of either one alone.

Going without relationship attributes?

- Strictly speaking, we are not forced to use relationship attributes
- Instead, can create an entity set representing values of the attribute.
- We then make that attribute participate in the relationship, making the original n-way relationship now (n+1)-way.
- Regarding our previous example: We could instead create an entity name Prices which has a single attribute price

Example: Alternate style of relationship attribute

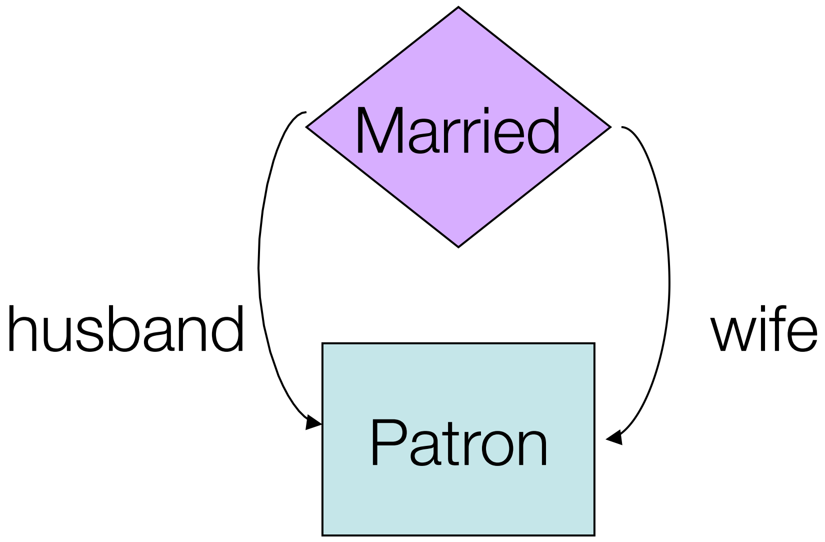


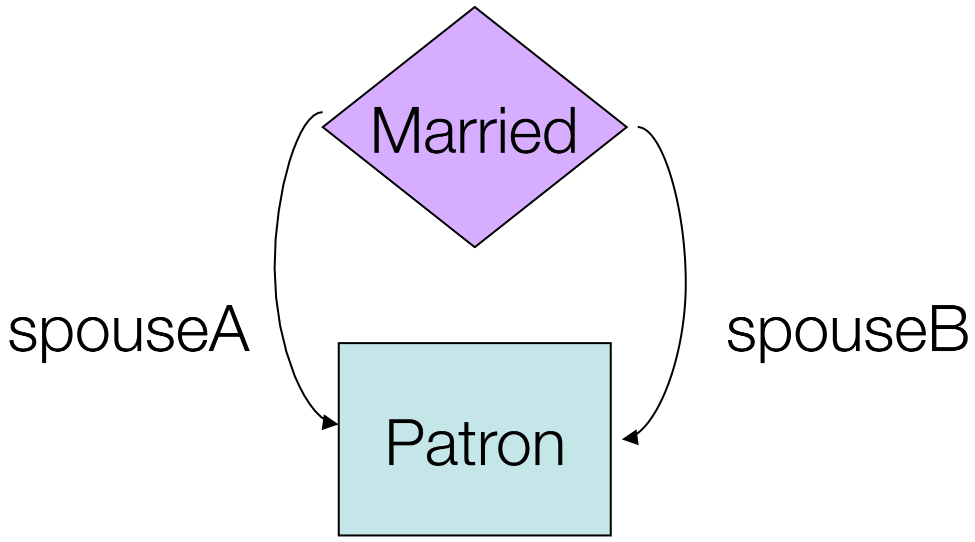
Interpreting these arrows can be a bit tricky. "Each combination of Pub and Beers may be associated with zero prices or one price."

Roles

- Entity sets may appear more than once in the same relationship!
  - This is not quite as odd as it might seem.
  - In real-world data modelling, the same person might be an employee of a company from one perspective, and a customer from another.
  - Regardless, we need to know how the entity is being used in each place it appears connected to the relation
- We label the edge between the entity set and relation with a name
  - This name is called the **role.**

Example: Roles

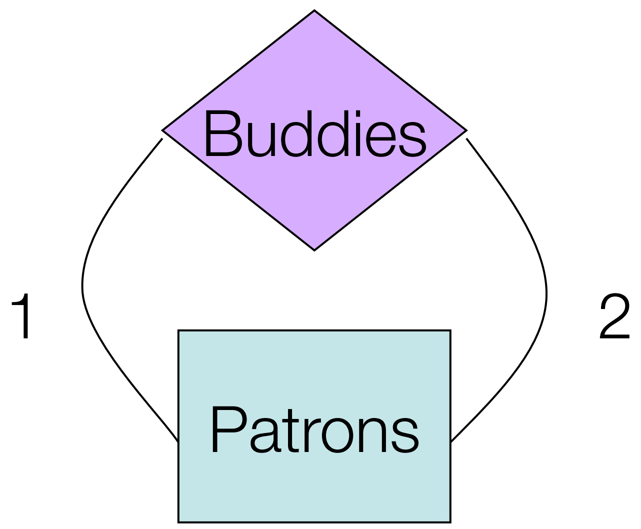




husband	**wife**
Bob	Ann
Joe	Sue
...	...

spouseA	**spouseB**
Pat	Chris
Sam	Stevie
Ernie	Bert
...	...

Example: Roles



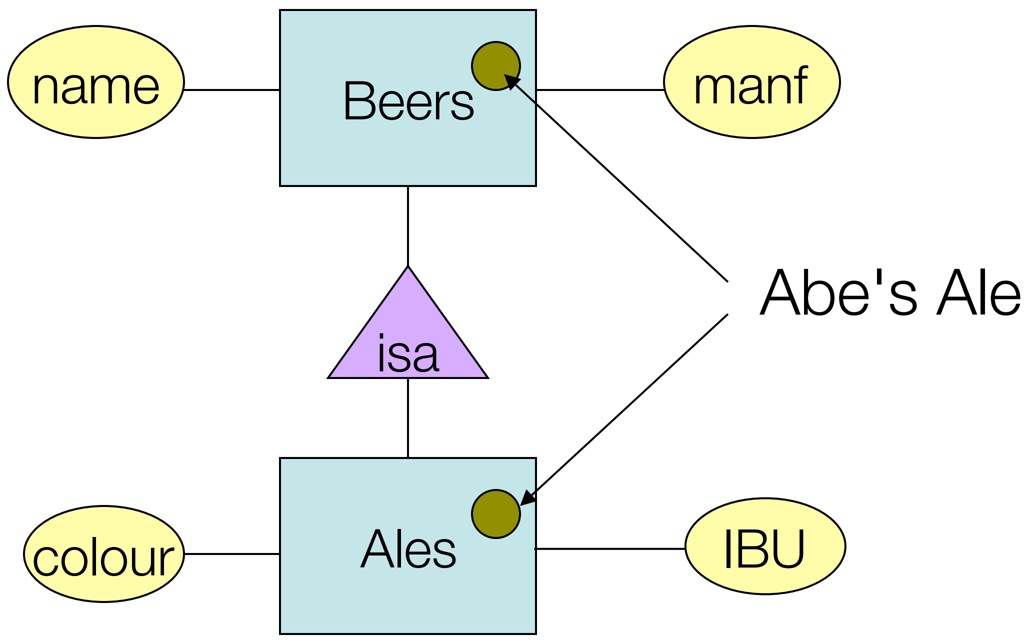
buddy1	buddy2
Cliff	Norm
Moe	Crusty
Ann	Bob
Mike	Tom
...	...

What does this diagram say about the ways Patrons may have buddies?

"Is-a" hierarchies

- We can denote special cases of entities
  - The text refers to these special cases a subclasses
  - However, they are not the same thing we use in object-oriented languages
- Example:
  - Ales are a kind of beer
  - However, not every beer is an ale (but some are), and some ales are more bitter than others
  - Let us indicate an ale by its colour and its bitterness (IBU)
  - Therefore some beers – which are ales – will also have an additional attribute for colour
- We'll assume that classes in a "is-a" hierarchy form parent-child relationships
  - is-a triangles indicate the parent in the relationship

Example: Representatives of Entities

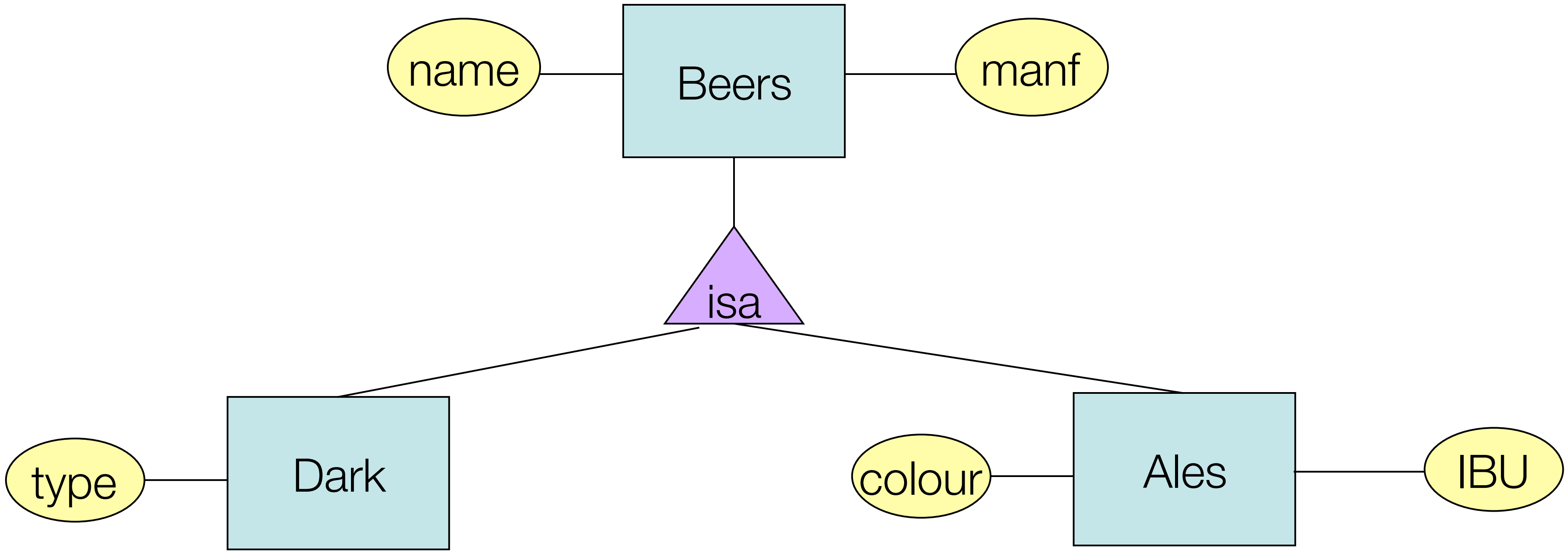


Trivia: Philips "Amnesiac" is medium-light in colour & has an IBU of 85. Alexander Keiths IPA is light in colourand has an IBU < 20.

E/R vs. OO subclasses

- In object-oriented languages, an object is in one class only
  - In multiple-inheritance languages, an object's class may have more than one parent class...
  - ... but the object itself belongs to one class.
- In E/R, entities have representatives in all "is-a" entities to which they belong
  - If entity **e** is represented in a hierarchy, then **e** is represented in each parent in the hierarchy.

Example: Representatives of Entities



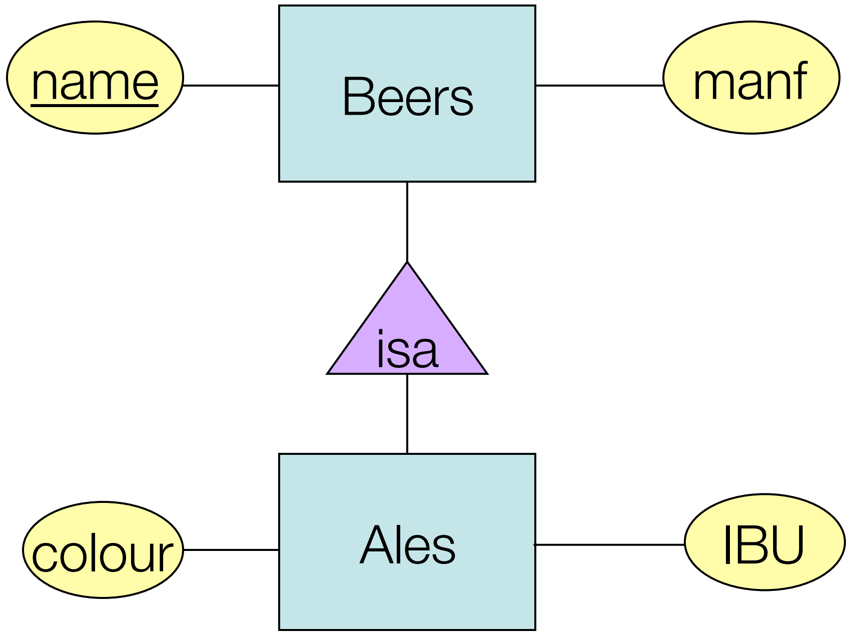
Dark beers can of type "dry stout", "porter", "baltic stout", "oatmeal", etc.

A "Cascadian Dark Ale", however, would belong to both "Dark" and "Ales"

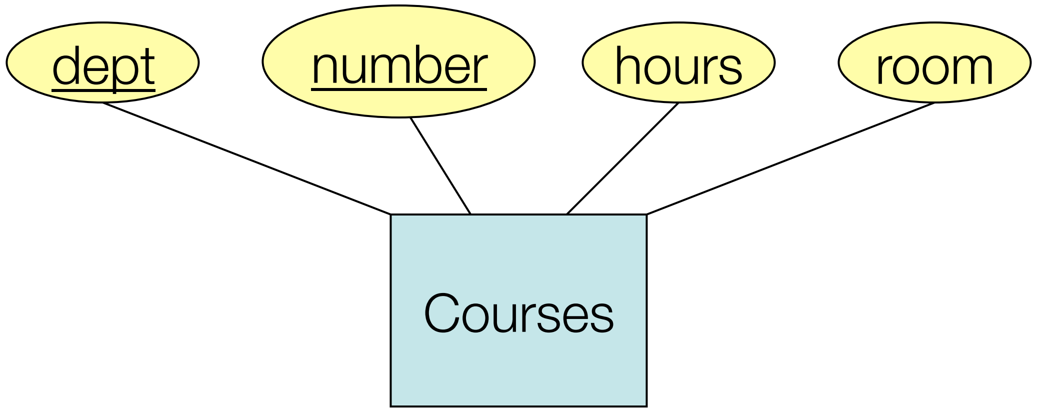
Keys

- A **key** is a set of attributes for one entity set such that...
  - ... no two entities in the set agree on all attributes of the key.
  - However, we do allow two entities to agree on some, but not all, of the key attributes.
- In E/R diagrams, we must designate a key for every entity set
  - Underline the key attributes.
  - With an ISA hierarchy, only root entity has a key (and it must serve as the key for all entities in the hierarchy).

Example: key for Beers



Example: Multi-attribute key



- Note that **hours** and **room** could also serve as a key, but there may exist only one key for an entity.

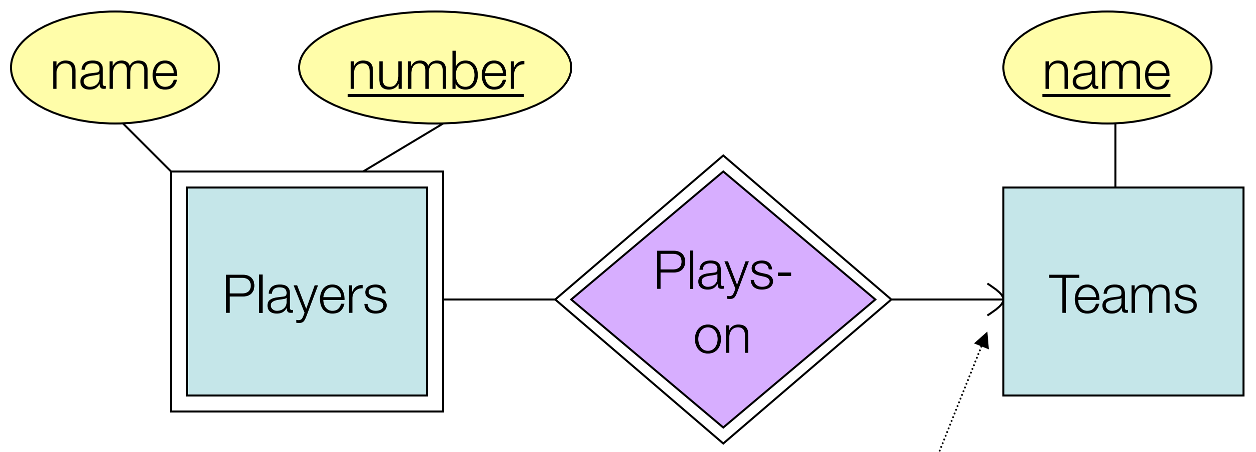
Weak entity sets

- Earlier we stated that entities are things that have some identity
- Sometimes, however, entities need more information to identify them uniquely
- An entity set E is said to be **weak** if:
  - in order to identify entities of E uniquely...
  - ... we need to follow one or more many-to-one relationships from E ...
  - ... and include the key of the related entities from the connected entity sets.

Example

- **name** is almost a key for rugby players, but there might be two with the same name
- **number** is definitely not a key since a player's number indicates their position on the field
  - and therefore players on two different teams could have the same number
- However, the **number** along with the **team nam**e for the player will help us **uniquely identify the player**
  - We will therefore use the PlaysOn relationship to connect the weak entity Players to Teams

Example: in an E/R Diagram



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

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

Rules for weak entity sets

- A weak entity set has one or more many-to-one relationships to other (supporting) entity sets
  - Not every many-to-one relationship from a weak entity set need be supporting.
  - However, supporting relationships must have the hollow arrow (i.e., entity at the "one" end is guaranteed to be there).
- The key for a weak entity set is its own underline attributes plus the keys for the supporting entity sets
  - Recall: (player) number + (team) name is the key for Players in our example.