ER Model

CSC 343
Fall 2019

MICHAEL LIUT (MICHAEL.LIUT@UTORONTO.CA)

DEPARTMENT OF MATHEMATICAL AND COMPUTATIONAL SCIENCES

UNIVERSITY OF TORONTO MISSISSAUGA

UNIVERSITY OF TORONTO MISSISSAUGA

Overview of Database Design

Conceptual Designs

What are the entities and relationships in the enterprise?

What information about these entities and relationships should we store in our database?

What are the *integrity constraints* and *business rules* that hold?

i.e. We need to think about “Data Governance”

Purpose of Entity-Relationship (ER) Model

Allows us to create a visual representation of the database schema design.

- These are called entity-relationship diagrams.
- This visual also allows us to depict some constraints imposed in our schema.

Conversion of ER designs to relational database designs.

- This will come later!

Framework for ER Model

Design is a serious business! You are the architect of the database!

Business and management know they want/need a database, but they usually don't have sufficient background or expertise to tell you what they want in it.

Sketching the key components is a great way to view the hierarchal structure, as well as an efficient way to develop a "good" working database.

Entity Sets

Entity → a “thing” or object.

Entity Set → a collection of similar entities.

- For those programmers out there; it’s similar to a class in an OO language.
- Each entity set has a key.

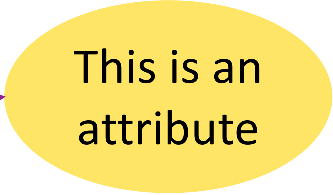
Attribute → property of an entity set.

- Attributes are simple values. e.g. int, char, or str, NOT struct, sets, etc…
- Each attribute has a domain.

ER Diagrams

Technicalities for drawing Diagrams (in this course):

- **Entity Set** → a rectangle.
- **Attribute** → an oval.
  - Requires a line to the oval, from the rectangle, representing its Entity Set.



NOTE: Notation varies! Some textbook represent attributes within the (entity) rectangle.

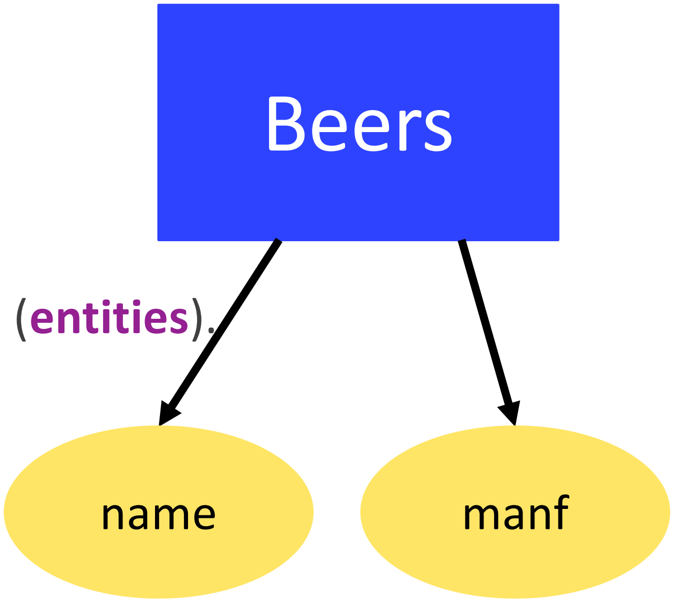
Example

Entity Set **Beers** has two attributes:

1. name
2. manf (manufacturer).

Each **Beers** entity has values for these two attributes (**entities**).

e.g. (Bud, Anheurser-Busch)



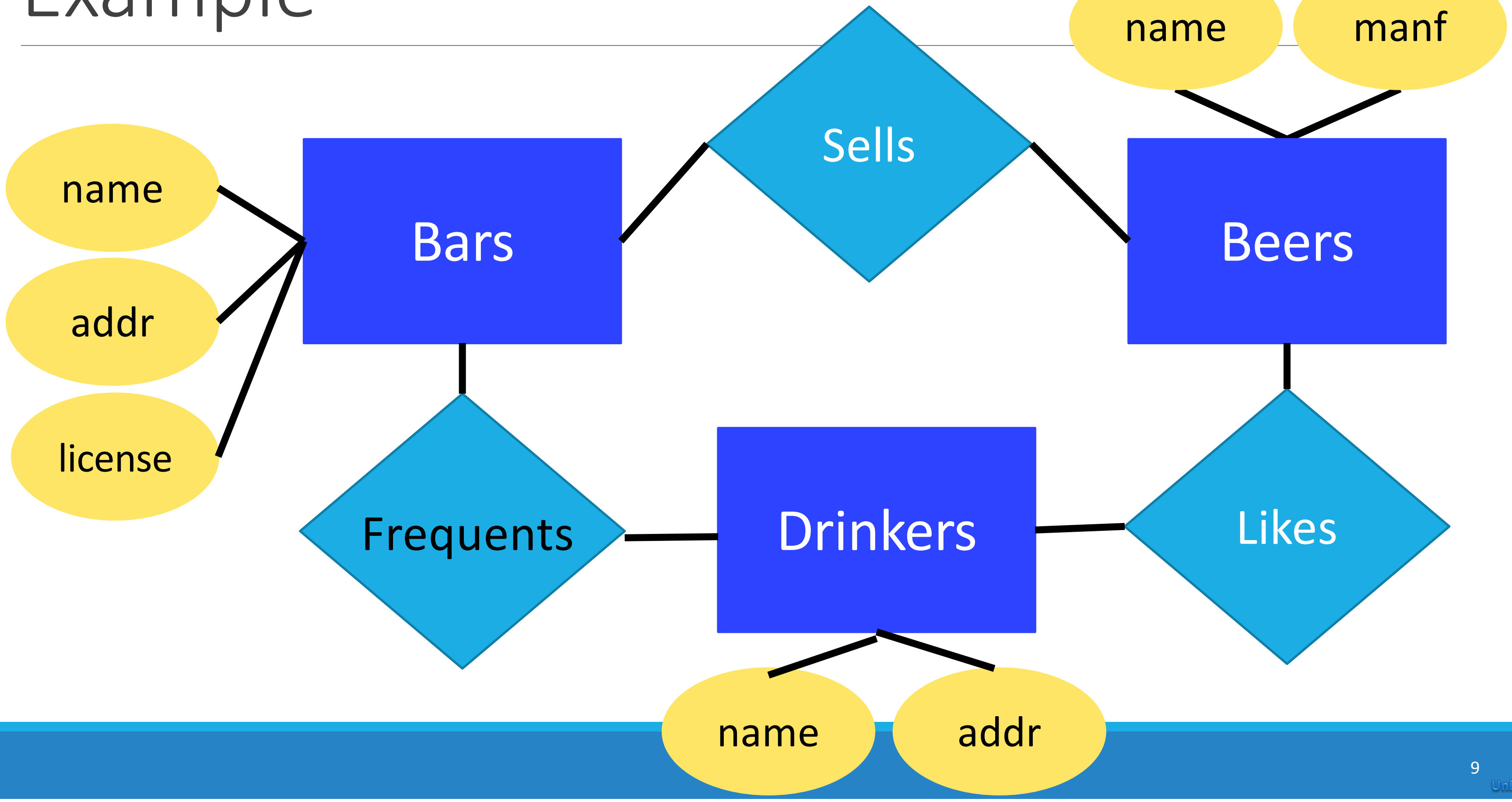
Relationships

A **relationship** is a connection between two or more entity sets.

It is represented by a diamond on an E/R diagram.

- Lines connecting it to each of the entity sets involved is required.

Example


name
manf
Sells
name
Bars
Beers
addr
license
Frequents
Drinkers
Likes
name
addr

Relationship Set

The current "value" of an entity set is the set of entities that belong to it.

e.g. 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.

Formally:

An n-ary relationship set R relates n entity sets E_1, …, E_n; where each relationship in R involves E_1, …, E_n.

- Same entity set could participate in different relationship sets, or different "roles" in same set.

Example

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

Bar	Beer
Joe’s Bar	Canadian
Joe’s Bar	Stella
Joe’s Bar	Miller
Tammy’s Bar	Canadian
Tammy’s Bar	Corona

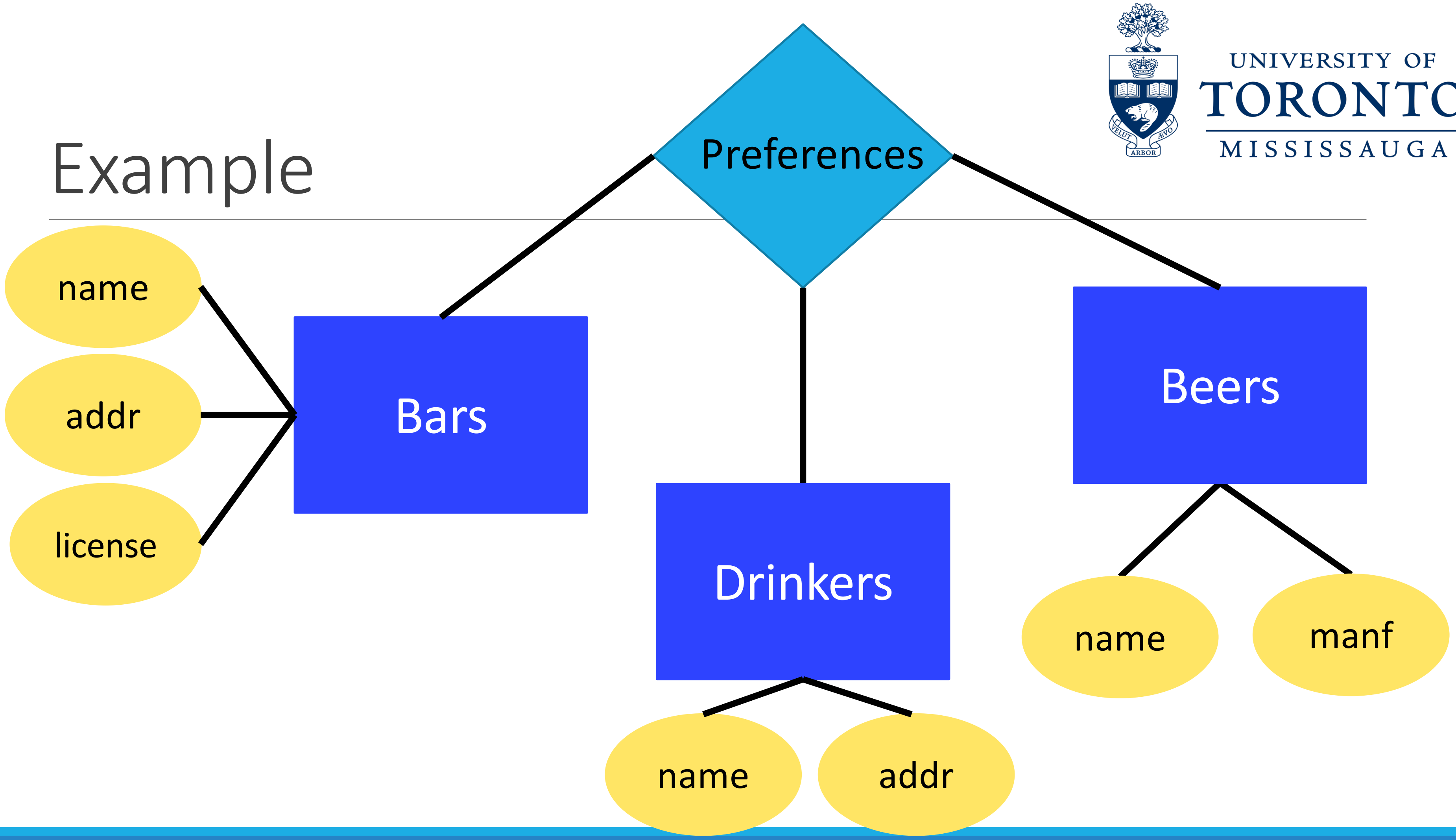
Multi-way Relationships

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


Preferences
name
addr
license
Bars
Beers
Drinkers
name
manf
name
addr

A Typical Relationship Set

Bar	Drinker	Beer
Joe’s Bar	Jenna	Canadian
Joe’s Bar	Abdi	Stella
Joe’s Bar	James	Miller
Tammy’s Bar	Jenna	Canadian
Tammy’s Bar	Abdi	Corona
Joe’s Bar	Abdi	Bud
Tammy’s Bar	James	Bud
Tammy’s Bar	James	Sleemans

Many-Many Relationship

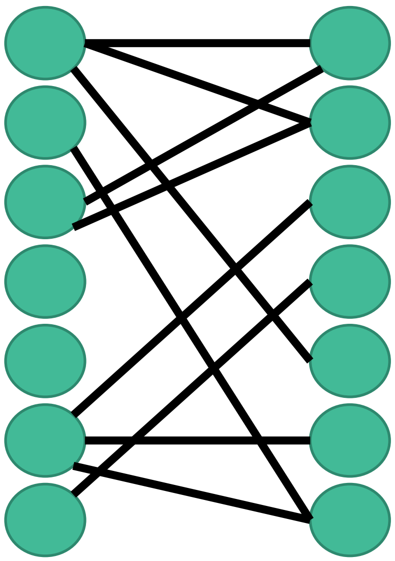
Focus: binary relationships

e.g. **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.

Many-Many Illustrated



<u>Note:</u> each line is an instance of the binary relationship!

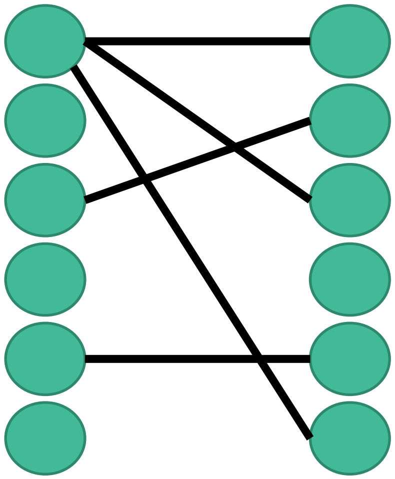
Many-One Relationships

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.

Many-One Illustrated



(Partial) Function on entity set.

Bars

Beers

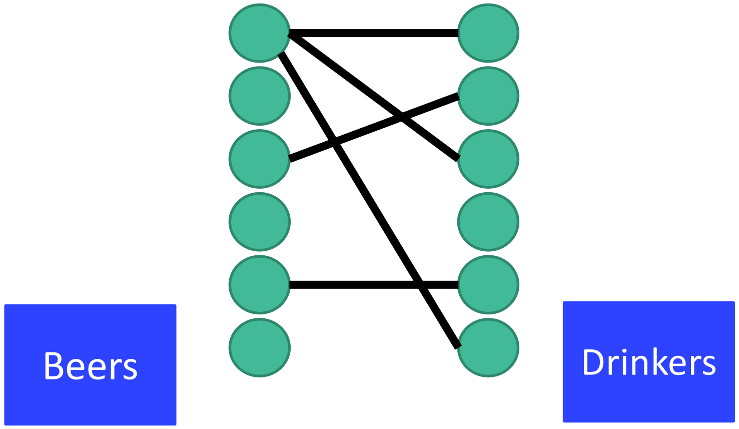
Drinkers

Example

Favourite, from **Drinkers** to **Beers** is many-to-one

A drinker has at most one favourite beer.

A beer can be the favourite of any number of drinkers (0 included).

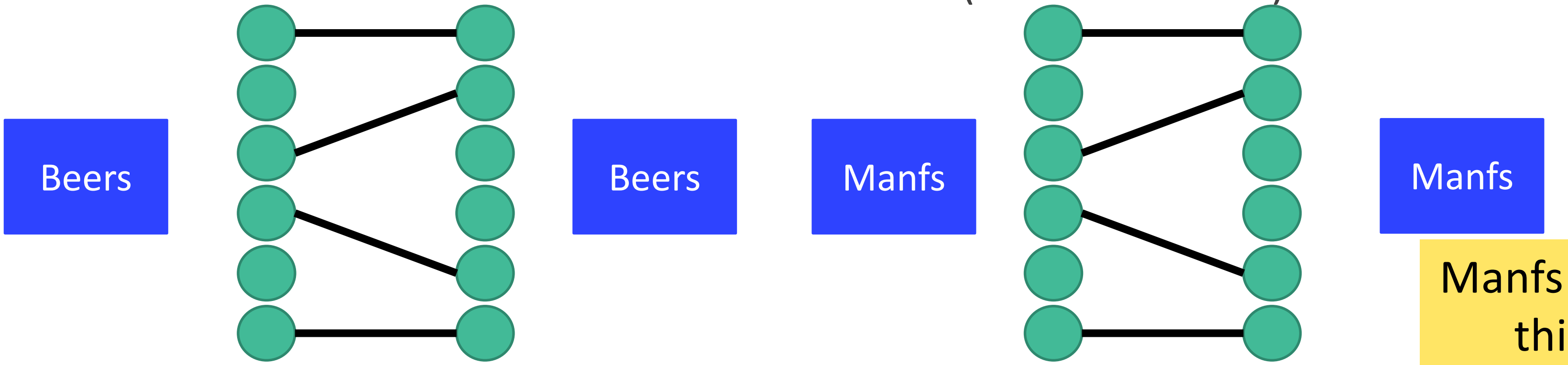


One-One Relationship

Each entity of either entity set is related to (at most) one entity of the other set.

e.g. Relationship **Best-Seller** between entity-sets **Manufacturers** and **Beers**.

- A beer is the best seller for {0|1} manufacturers, and no manufacturer can have more than one best-seller (assume no ties).



Manfs is an entity in this example

Representing “Multiplicity”

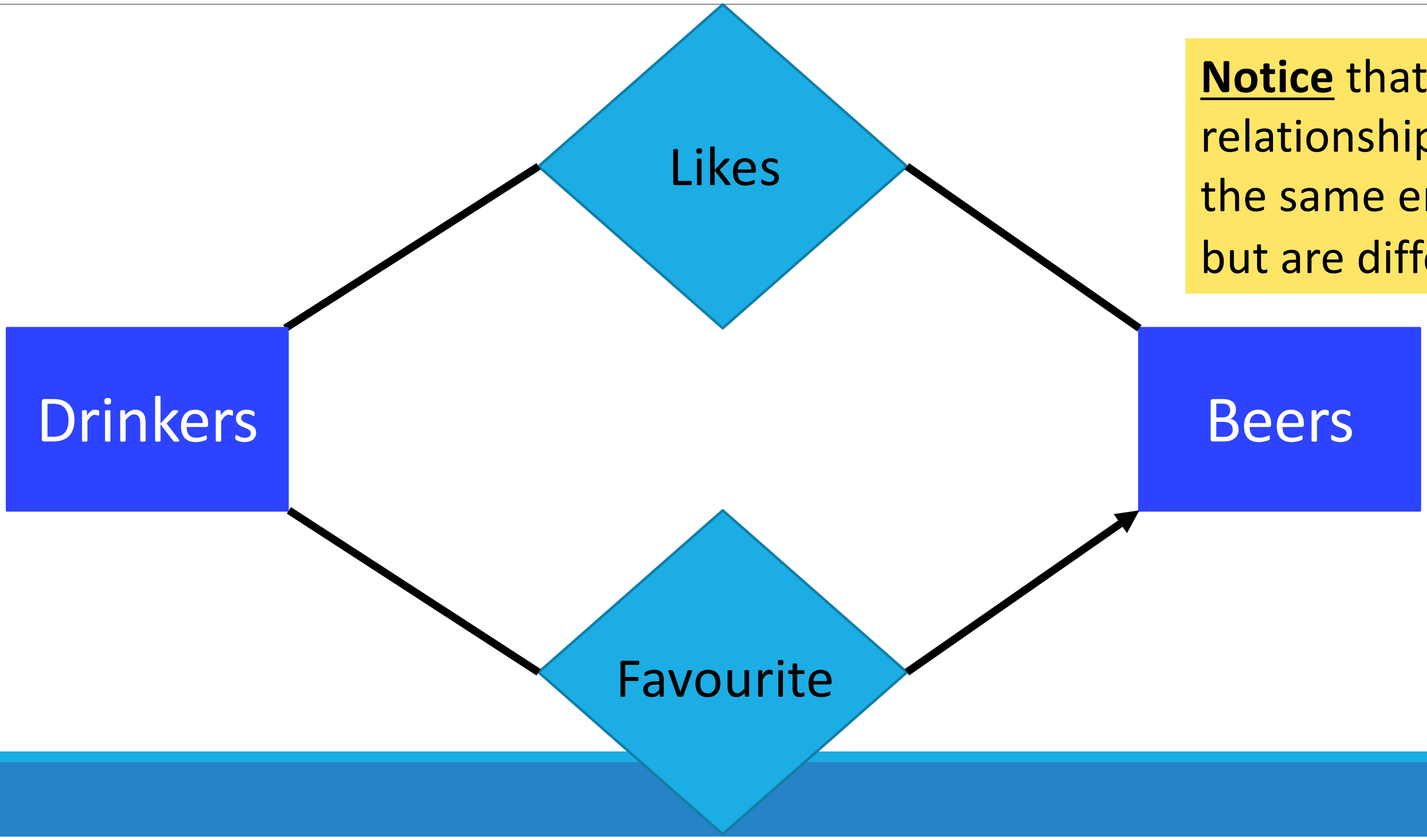
A many-one relationship is depicted by an arrow entering (at most) “one” side.

A one-one relationship is depicted by an arrow entering both entity sets.

Notation:

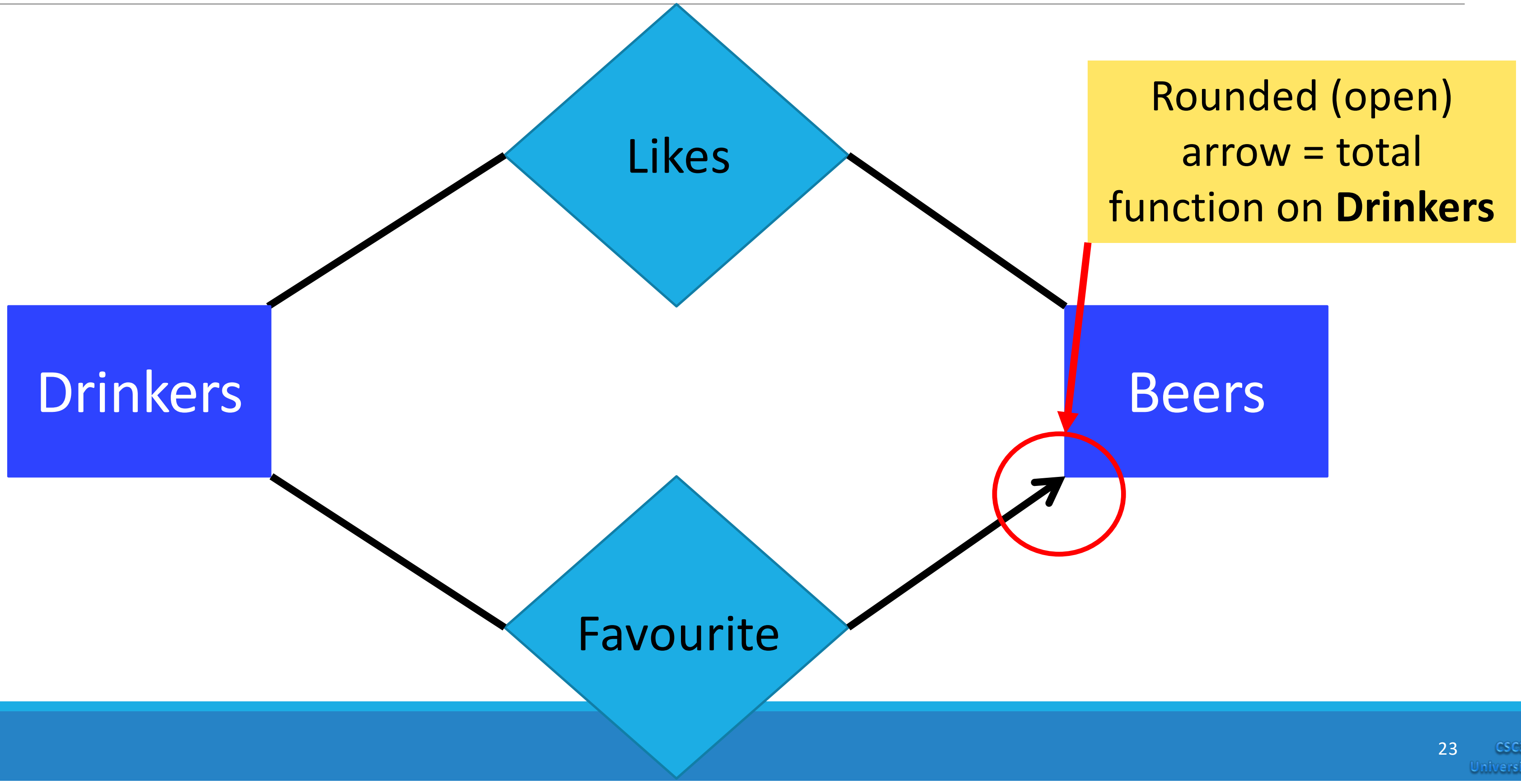
- Rounded (open arrow) = “exactly one”
  - i.e. each entity of the first set is related to exactly one entity of the target set.

Example: Many-One Relationship



<u>Notice</u> that two relationships connect the same entity-sets, but are different!

Example: Many-One Relationship

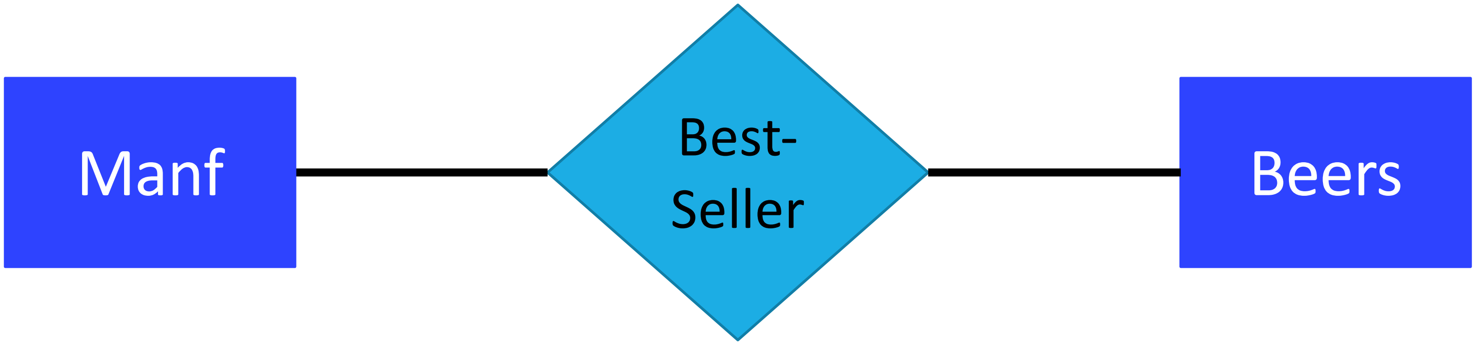


Rounded (open) arrow = total function on **Drinkers**

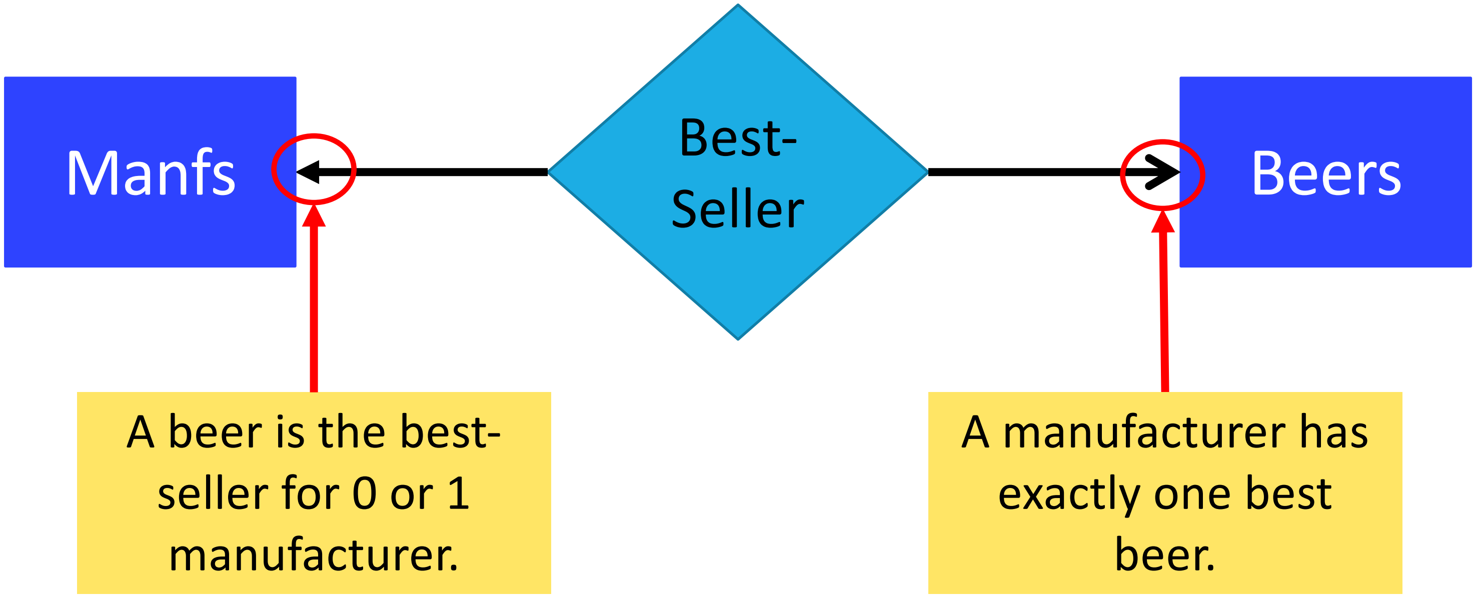
Example: One-One Relationship

Consider **Best-Seller** between **Manfs** and **Beers**.

- Some Beers <u>are not</u> the best-seller of any manufacturer.
- But a Beer manufacturer has to have a best-seller.



In the E-R Diagram



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

A manufacturer has exactly one best beer.

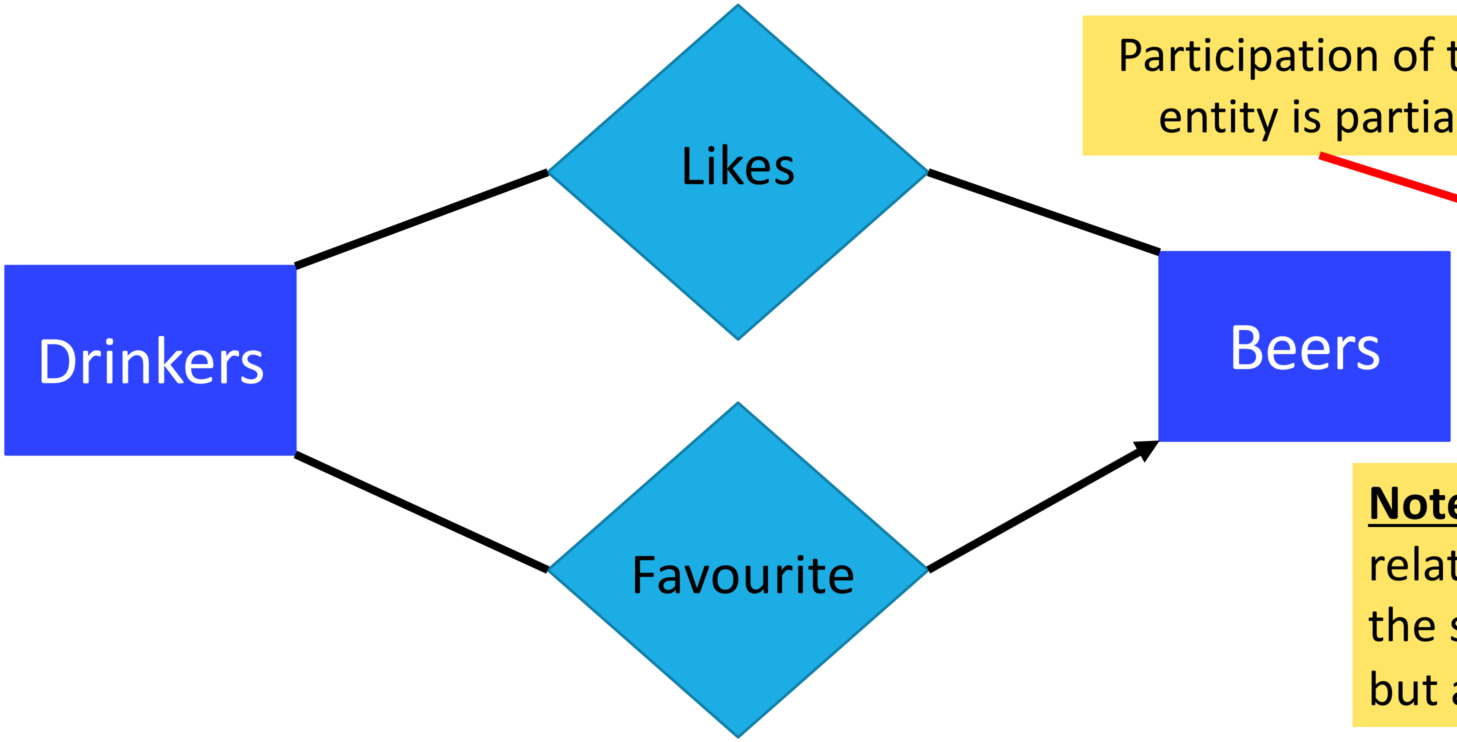
Participation Constraint

Does every student have to take a course?

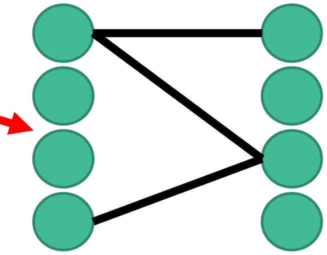
- If so, this is a *participation constraint*: the participation of **Students** in **Enrolled** is said to be *total* (vs. *partial*).
- Every **SID** value in **Students** table must appear in a row of the **Enrolled** table (with a non-null ***SID*** value!)

Textbook Notation: total participation represented by a thick (**bolded**) line originating from entity

Example: Many-One Relationship

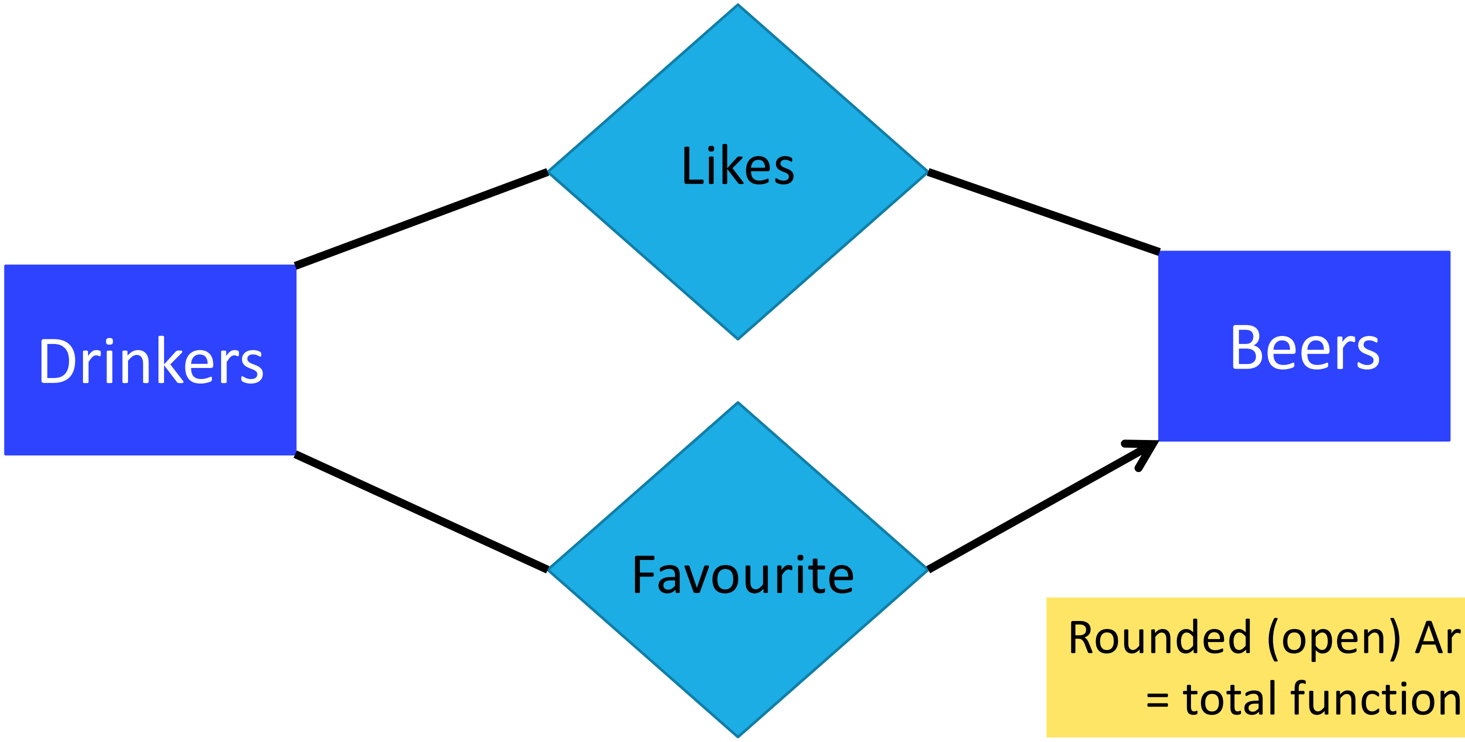


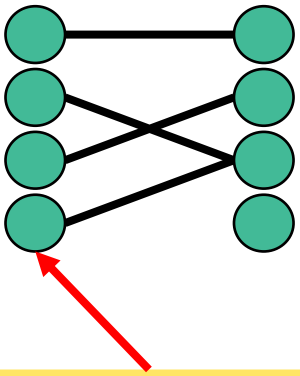
Participation of this entity is partial.



<u>Note:</u> Two relationships connect the same entity sets, but are different.

Example: Many-One Relationship

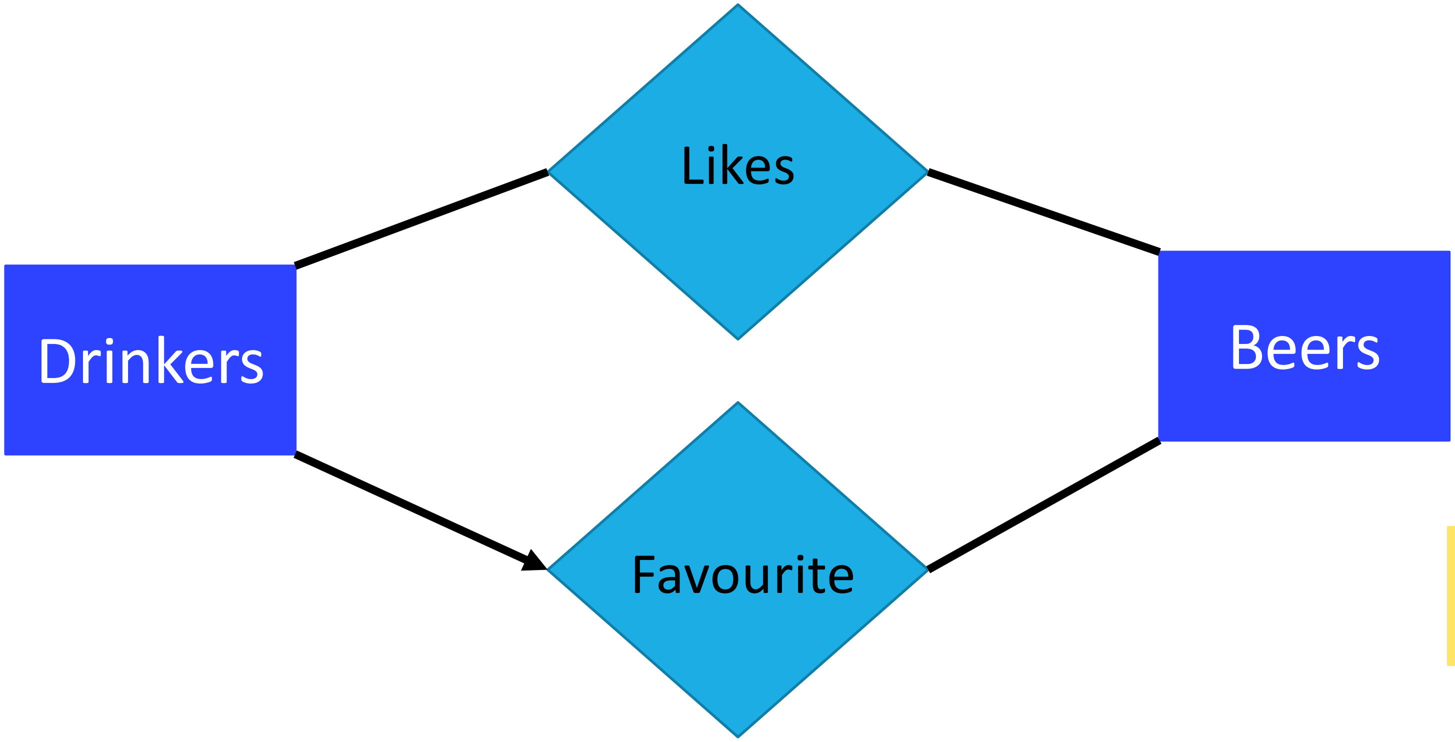


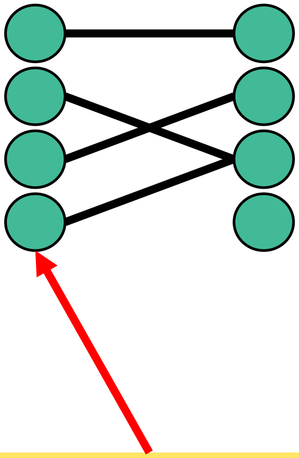


Participation of this entity is total.

Rounded (open) Arrow = total function

Alternate/Textbook Notation

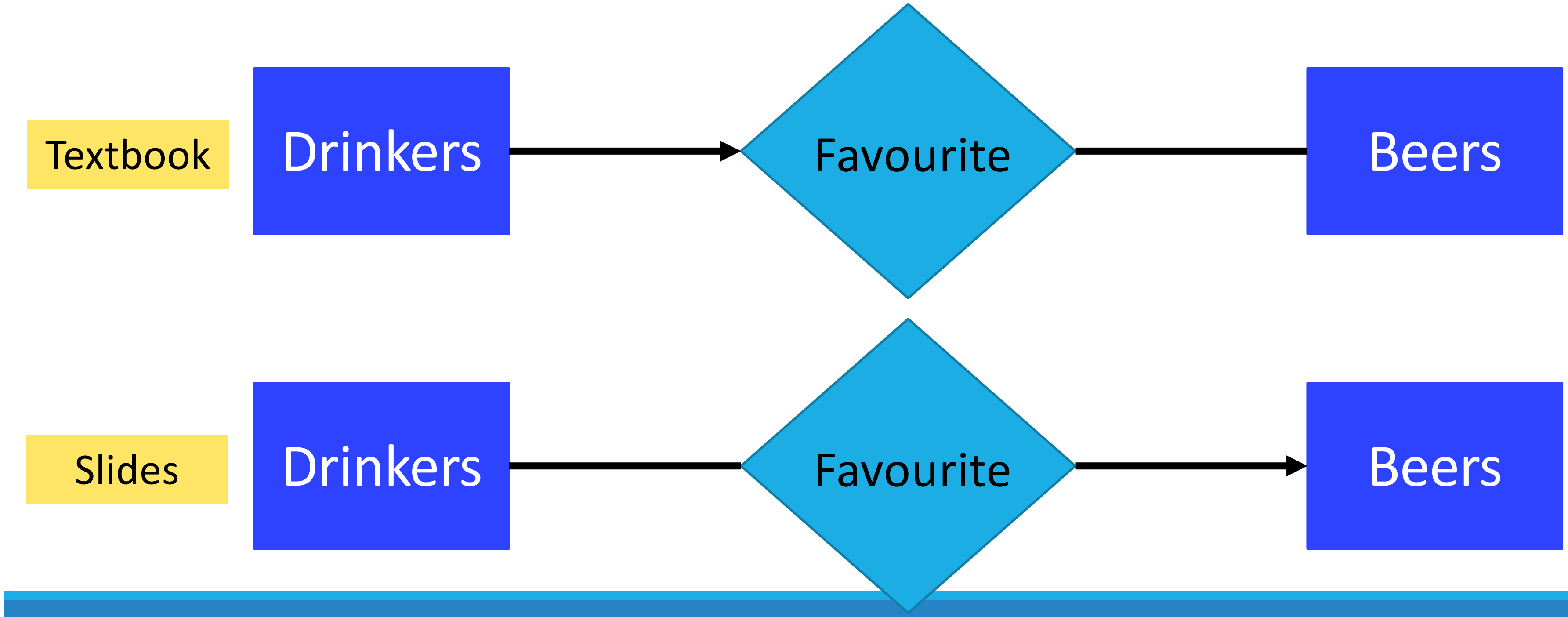




Participation of this entity is total.

Notation

Be consistent with your notation! You cannot interchange them!



“Chen” vs. “Crow’s Foot” Notation

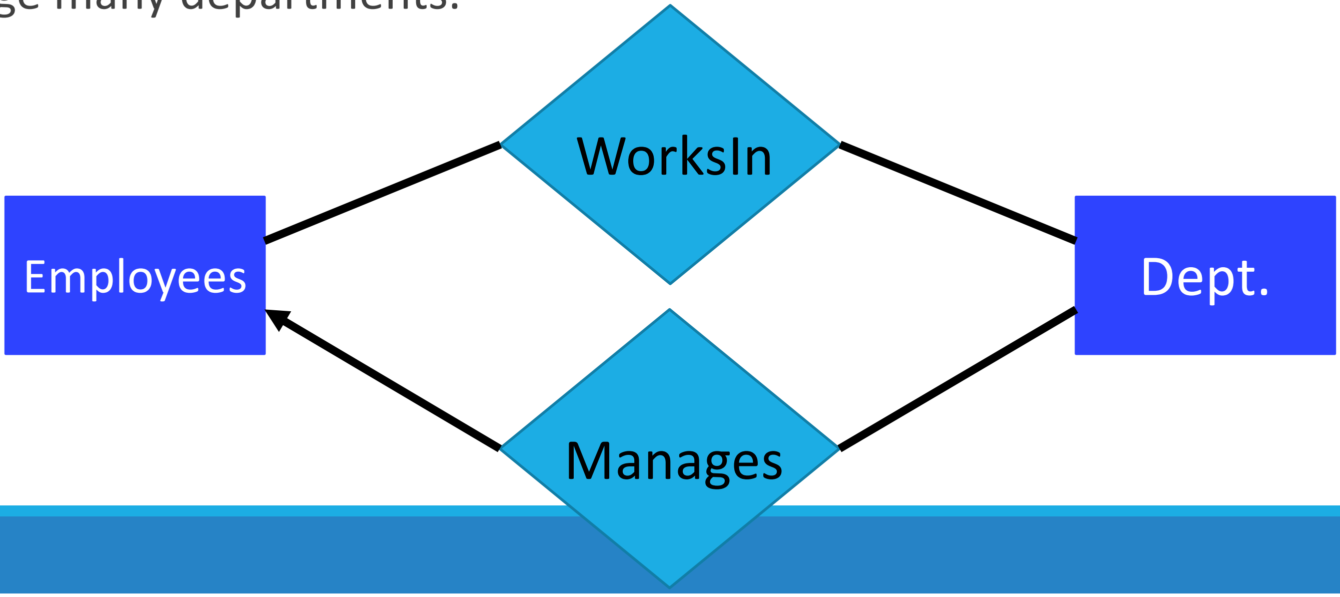
- **Take a look at the attachments provided!**

1. ERD Chen Notation
2. ERD Crow’s Notation

Key Constraints

Many-Many: “An employee can work in many departments, and a department can have many employees.”

One-Many: “A department has **<u>at most one</u>** manager, and employees can manage many departments.”



Employees

WorksIn

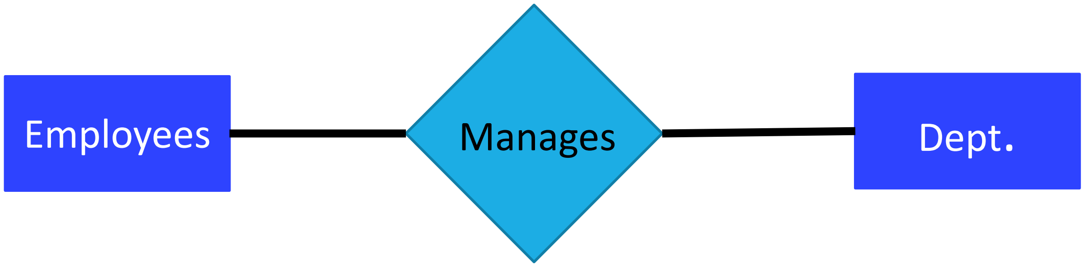
Dept.

Manages

Participation Constraints

Does every department have a manager?

- If yes, then then every department must appear in the manages relation: total participation vs. partial participation.



Employees

Manages

Dept.

━━━━ Total participation (all)

━━━▶ Key constraint (at most one)

[textbook] ━━━▶
[slides] ━━━>
} Total participation and key constraints (all and exactly one)

Attributes on Relationships

In certain instances, it is useful to attach an attribute to a relationship.

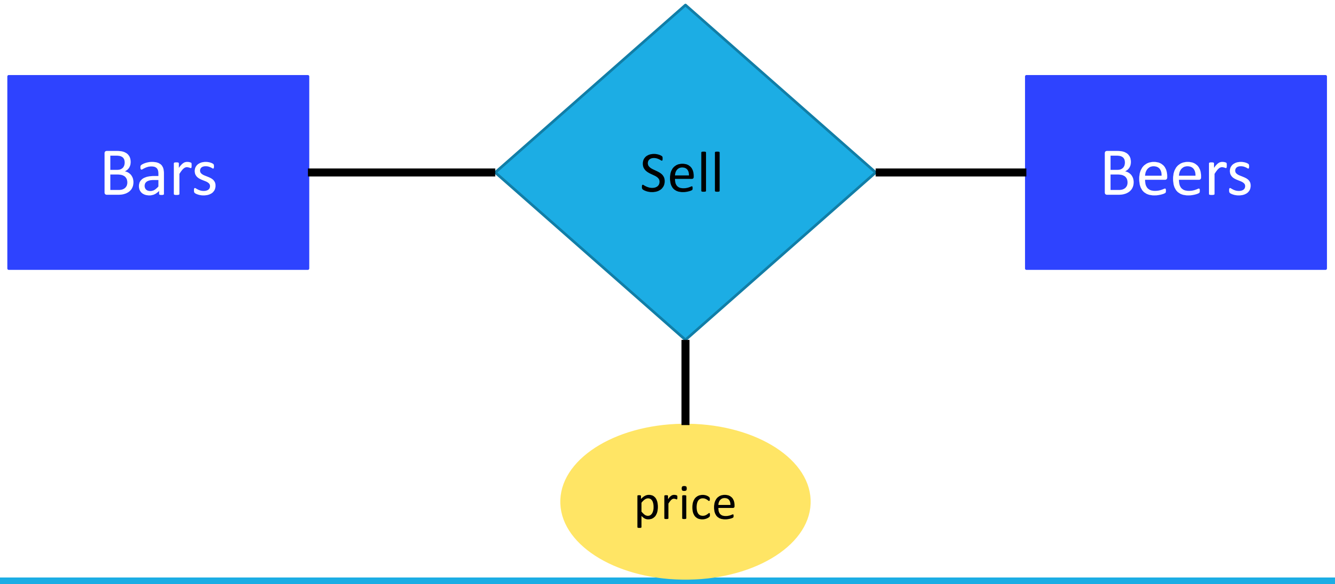
Think of this attribute as a property of tuples in the relationship set.

- i.e. a type of connector/bridge between entity sets to satisfy an entity.

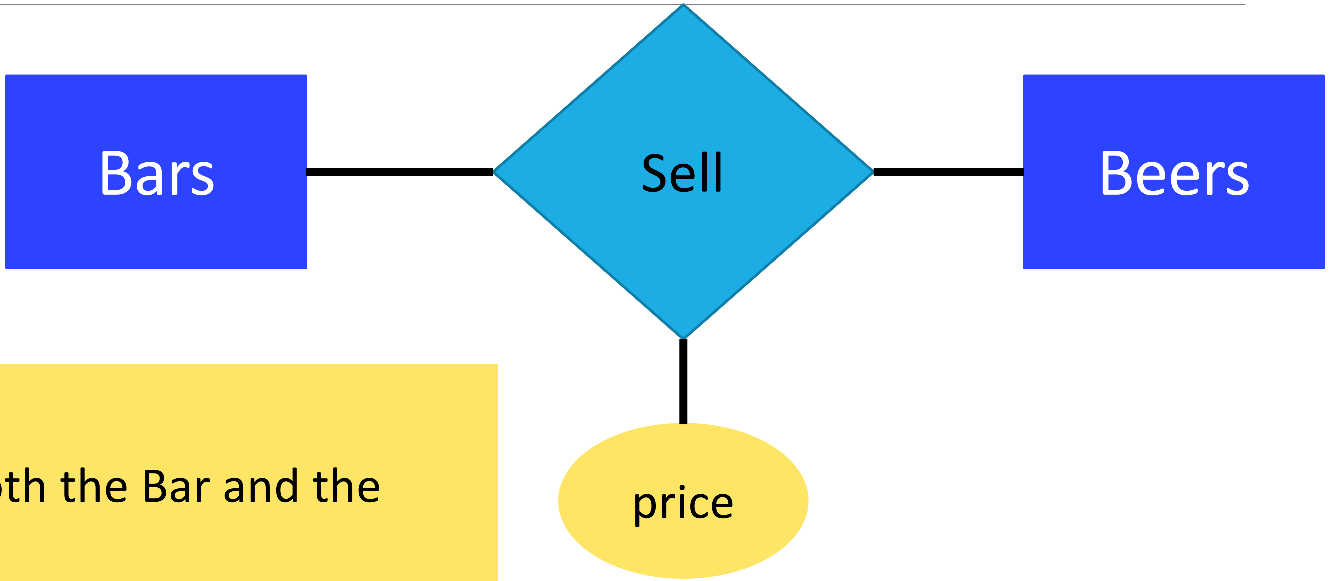
Let’s see an example!

Example

Bars **sell** **Beers**; **Beers** have a **price**; **Bars** have a **price**.



Example



Explanation:

Price is a function of both the Bar and the Beer, not of one alone.

e.g. “The price of Miller beer at Joe’s bar.”

Equivalent Diagrams

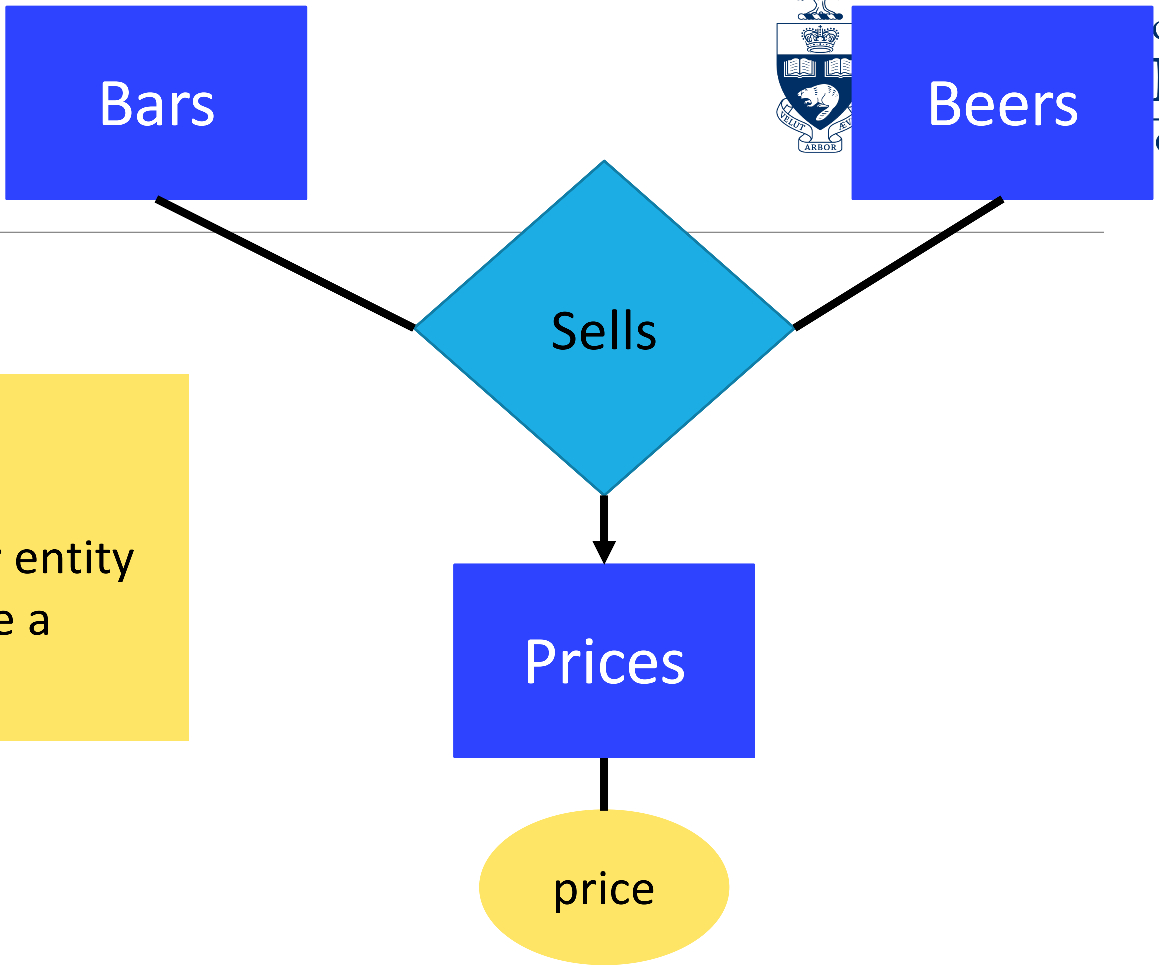
i.e. without attributes on relationships

Create an **entity set** representing values of the **attribute** (**entity**).

Make that **entity set** participate in the **relationship**.

Let’s see an example!

Example



Notation:

Arrow from multi-way relationship = “all other entity sets together determine a unique one of these”.

Roles

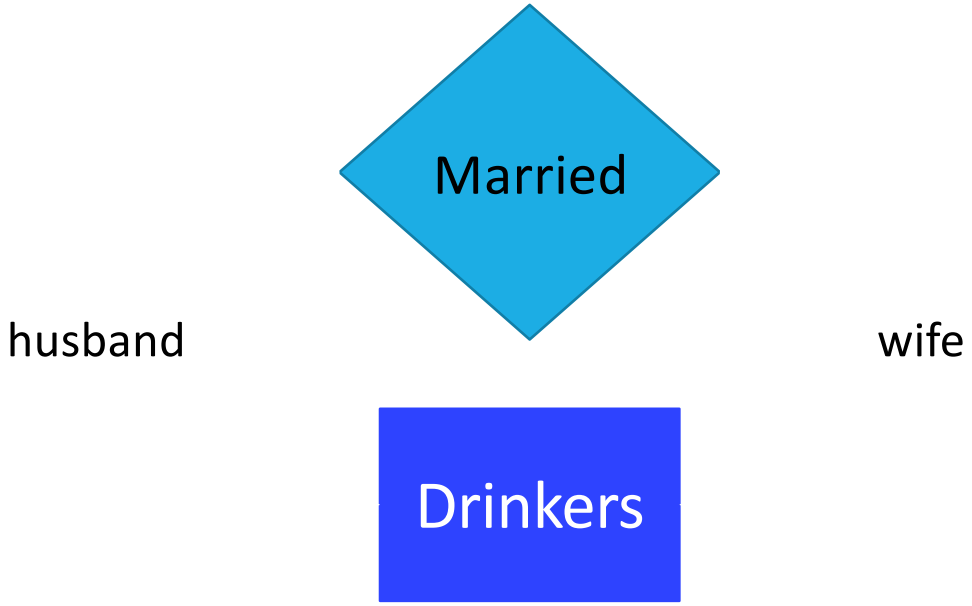
Sometimes an entity set appears more than once in a relationship.

Label the edges between the relationship and entity set with names called **roles**.

EXAMPLE

Relationship Set

Husband	Wife
John	Elizabeth
Warren	Alice
…	…



Married

husband

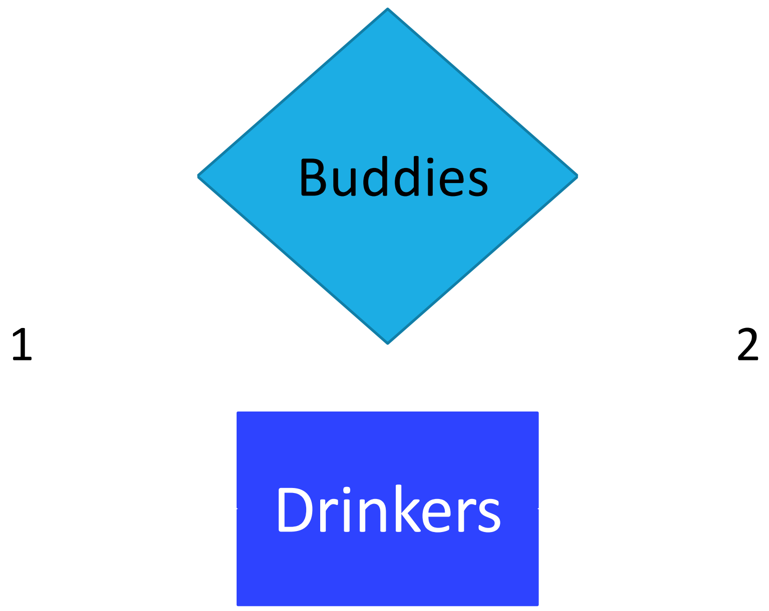
wife

Drinkers

EXAMPLE

Relationship Set

Buddy1	Buddy2
Mike	Joe
Liz	Lisa
Jenny	Peter
Courtney	Moe
…	…



Buddies

1

2

Drinkers

Subclasses

Subclass = special case = more properties

e.g. Ales are a kind of beer.

- Not ever beer is an ale, but some are.
- Let us suppose that in addition to all the *properties* (attributes and relationships) of beers also have the attribute **colour**.

Subclasses in ER Diagrams

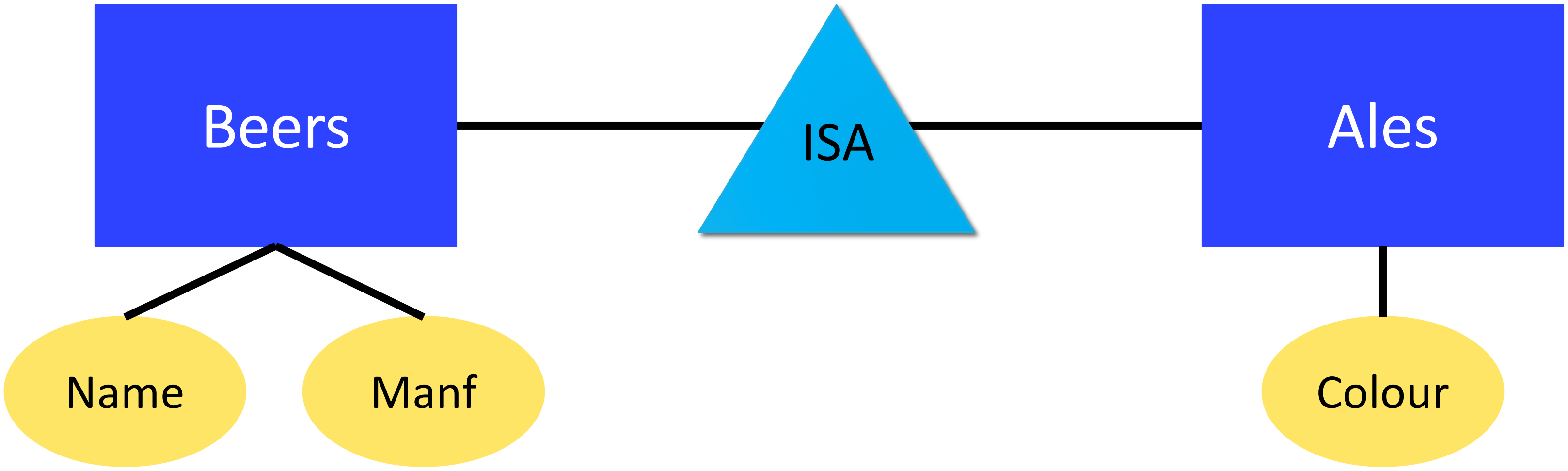
ISA triangles indicate the subclass relationship.

- Point to the superclass.

Reasons for using ISA:

- To add descriptive attributes specific to a subclass.
- To identify entities that participate in a relationship.

Example



Note:

Assume subclasses form a tree.

ISA ("is a") Hierarchies

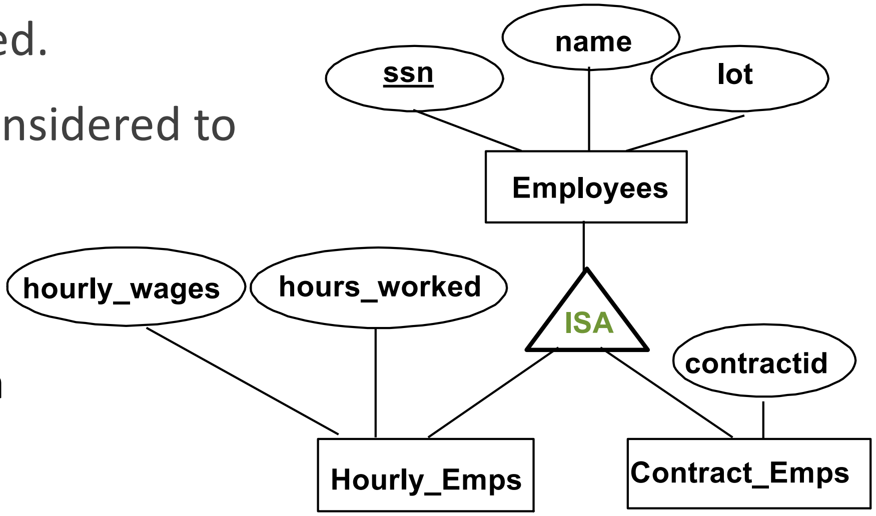
As in C++, or other PLs, attributes are inherited.

If we declare A **ISA** B, every A entity is also considered to be a B entity.

- Overlap constraints: Can two sub-classes contain the same entity?

  e.g. Can Mike be an Hourly_Emps as well as a Contract_Emps entity?

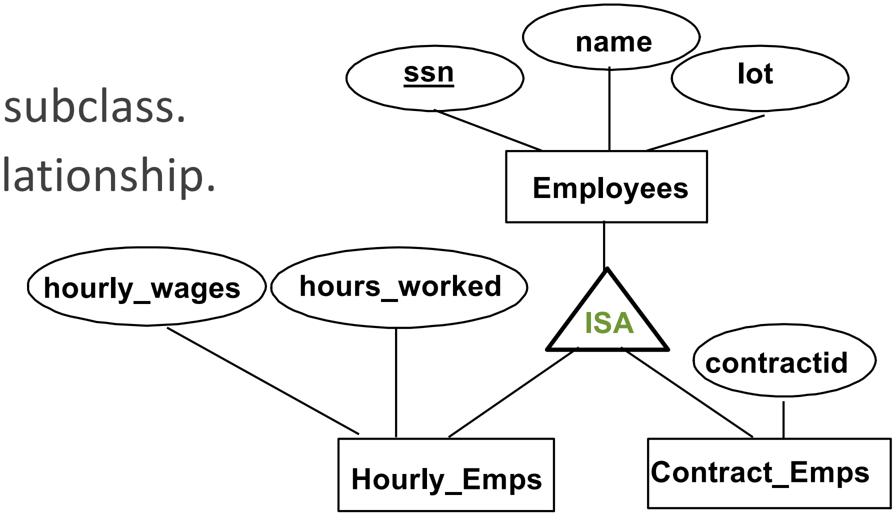
- Covering constraints: Does every Employees entity have to be an Hourly_Emps or a Contract_Emps entity?



ISA ("is a") Hierarchies

Reasons for using **ISA**:

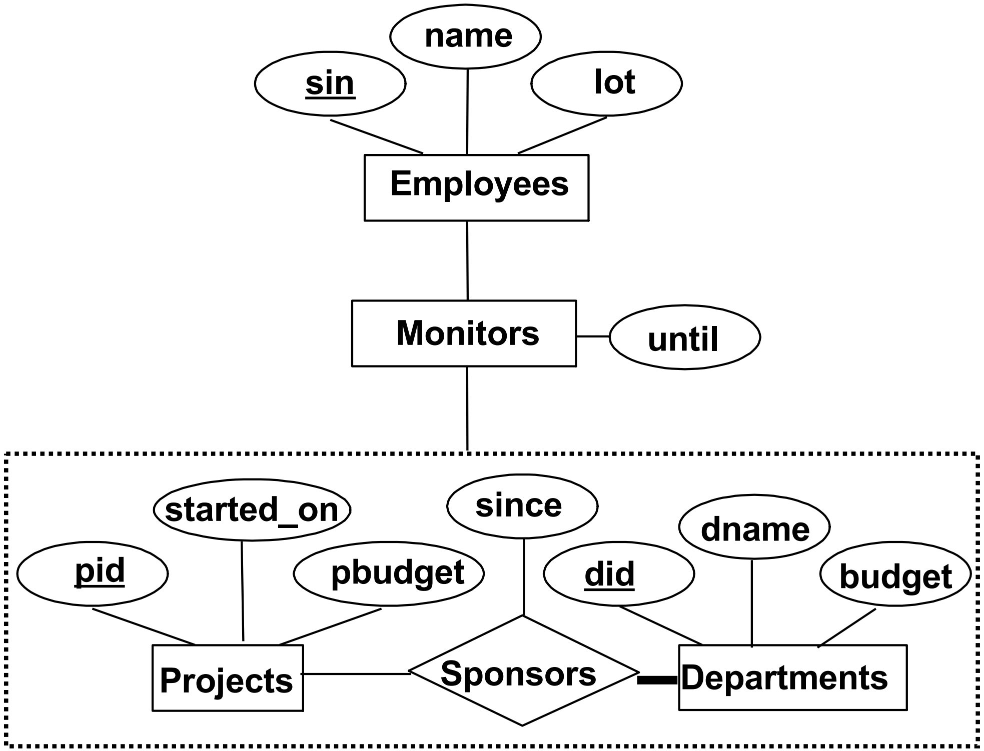
- To add descriptive attributes specific to a subclass.
- To identify entities that participate in a relationship.



Aggregation

Used when we have to model a relationship involving (entity sets and) a relationship set.

Aggregation allows us to treat a relationship set as an entity set for the purposes of participation in (other) relationships.



Keys

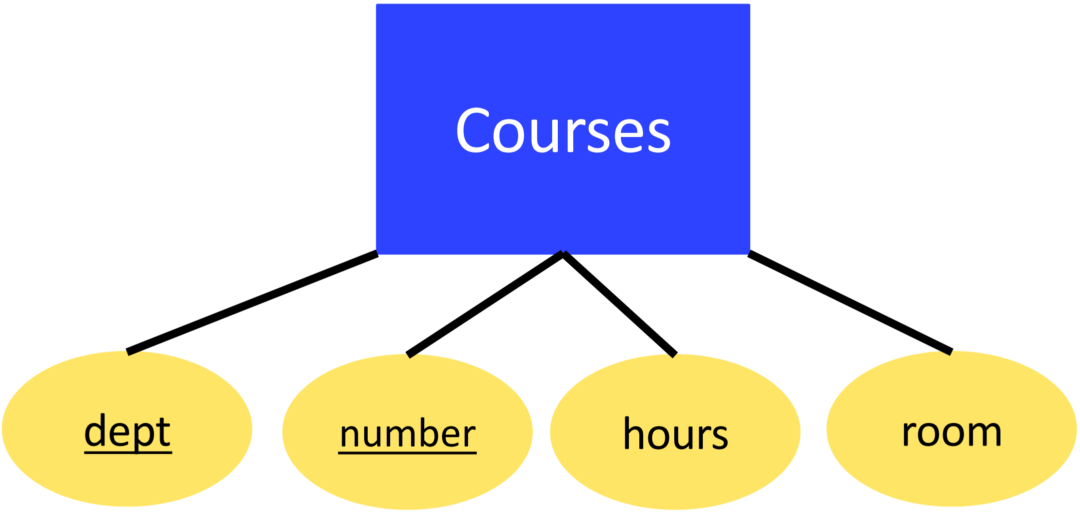
A ***key*** is a set of attributes for one entity set such that no two entities in this set agree on all the attributes of the key.

- 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.

- This is identified by underlining the key attribute(s).

Example: Multi-Attribute Key

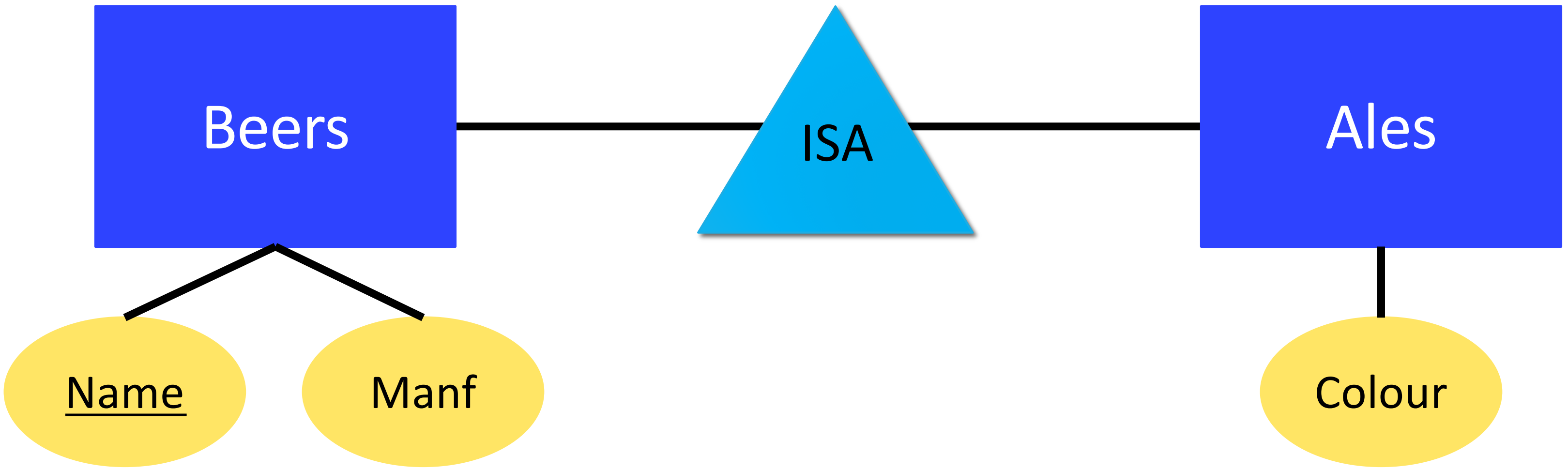


Note:

hours and **room** could also serve as a key, but we must select only one primary key (compound in this case).

Keys

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.



Weak Entity Sets

Occasionally, entities of an entity set need "help" to identify them uniquely.

Let E represent an entity set; E is said to be ***weak*** if:

- E is uniquely identifiable by more than one many-one relationships from E

and

- E includes the key of the related entities from the connected entity sets

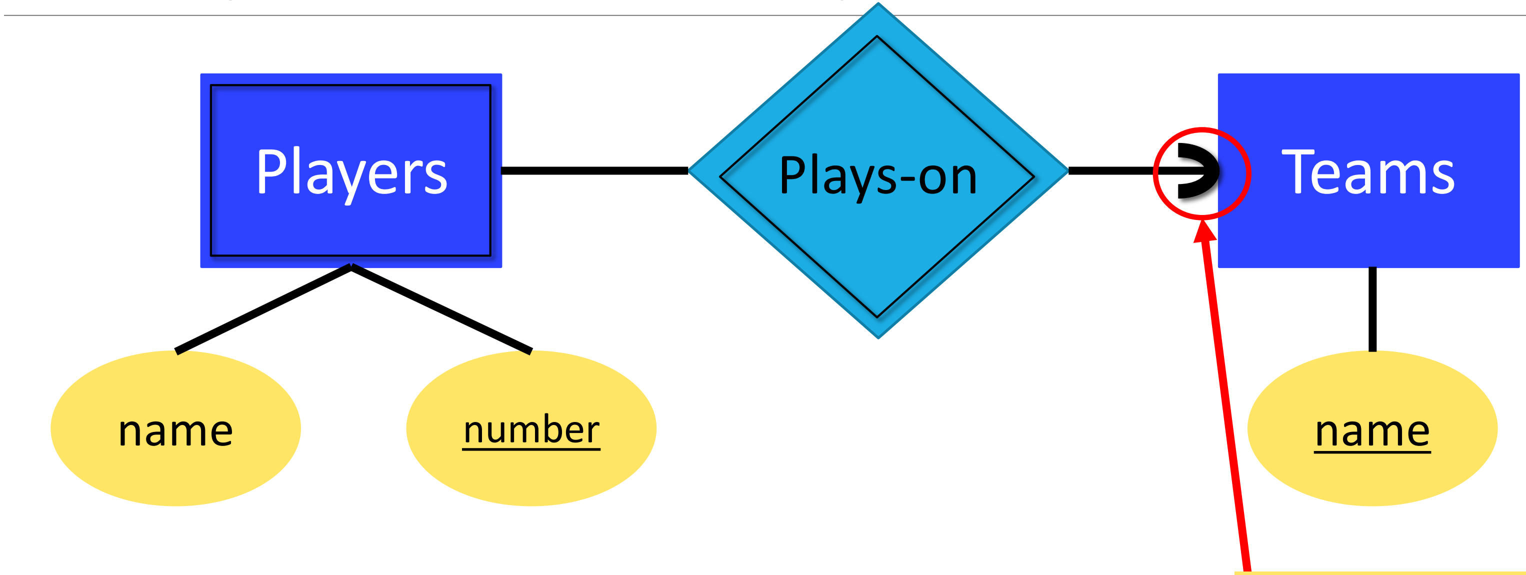
Example: Weak Entity Set

name is almost a key for football players, but there could potentially be two players with the same name.

number is certainly not a key, since players on two teams could have the same number.

But Teams name and number (i.e. {name, number}) with relation to the player by Plays-on is unique.

Example: Weak Entity Set



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.

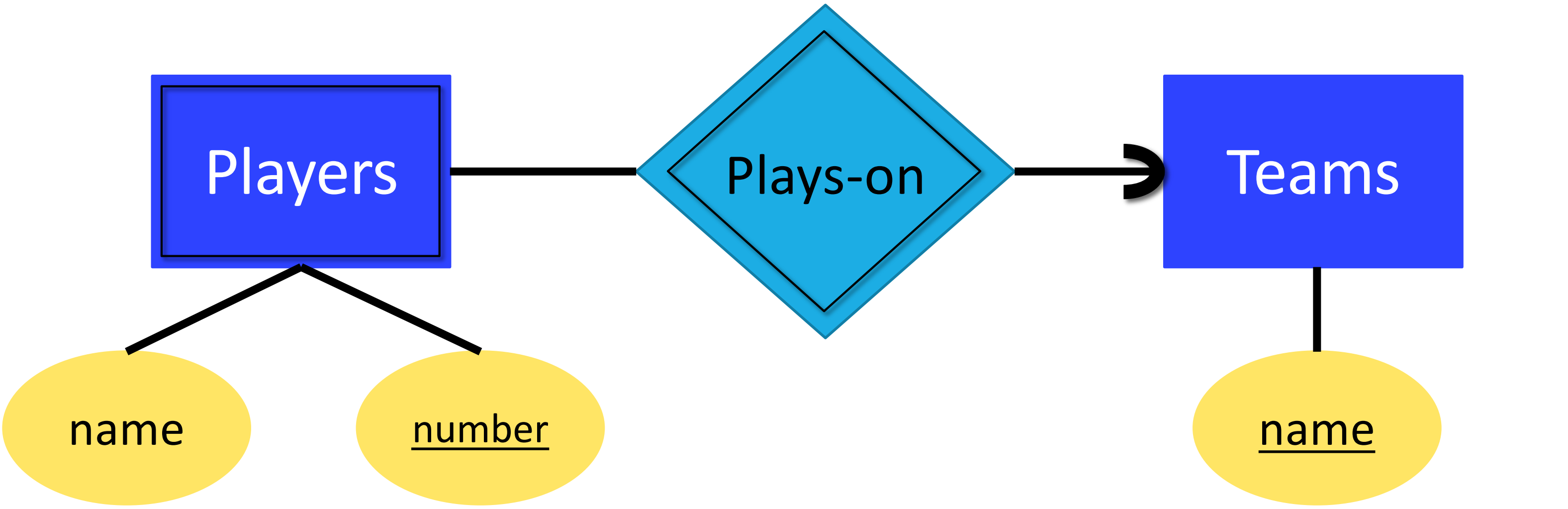
- Not every many-one relationship from a weak entity set need be supporting.
- But supporting relationships must have a rounded arrow (entity at the “one” end is guaranteed).

Weak Entity-Set Rules

The key for a weak entity set is its own underlined attributes and the keys from supporting entity sets.

e.g. number (Players) and name (Teams) is a key for **Players**.

Example



e.g. number (Players) and name (Teams) is a key for **Players**.

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.

1. Avoiding Redundancy

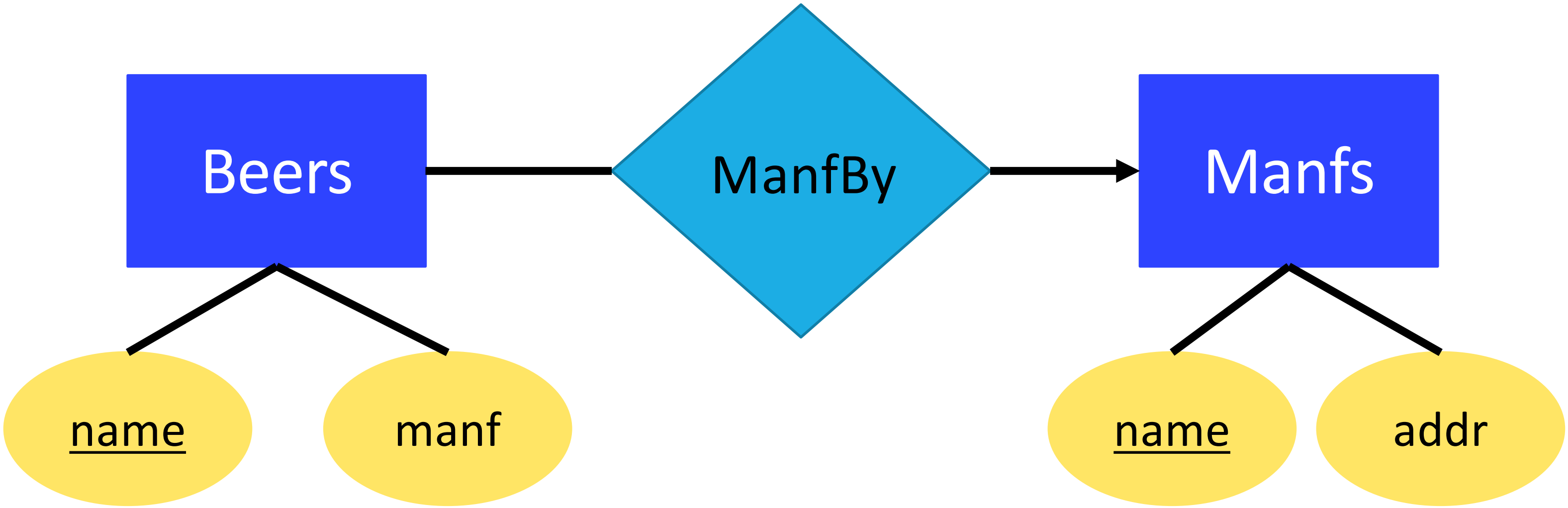
Redundancy is saying the same thing in two (or more) different ways.

- "the inclusion of extra components that are not strictly necessary to functioning; repetition or superfluity of information."

Redundancy wastes space and (more importantly) encourages inconsistency.

- Multiple representations of the same fact become inconsistent if we modify one and forget/do not modify its counterpart.

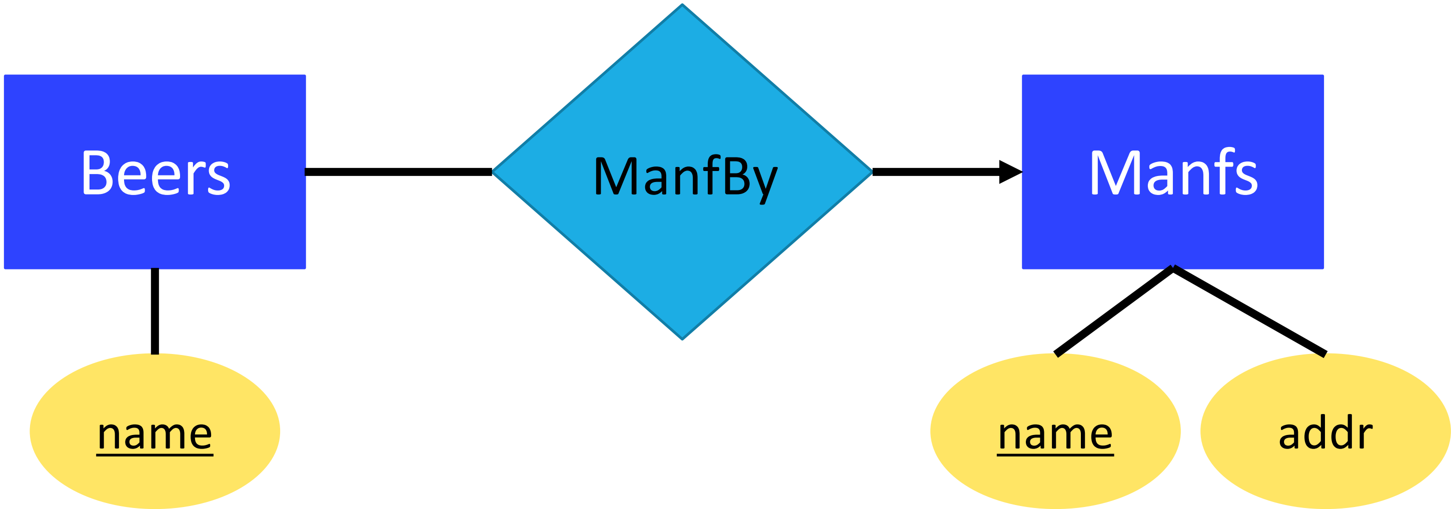
Example: Bad



Note:

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

Example: Good

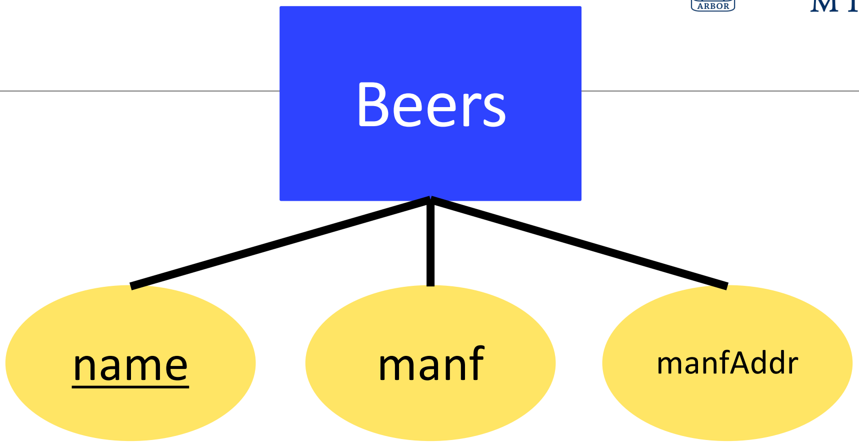


Note:

This design gives the address of each manufacturer exactly one.

Example: Bad

Can somebody tell me why?



Note:

This design is repetitive! The manufacturer’s address will be repeated for each beer. Also, in the event there are temporarily no beers for a manufacturer the value is lost.

Entity Sets vs. Attributes

An entity set should satisfy (at least) one of the following:

- It is more than the name of something (*i.e.* it has at least one non-key attribute).

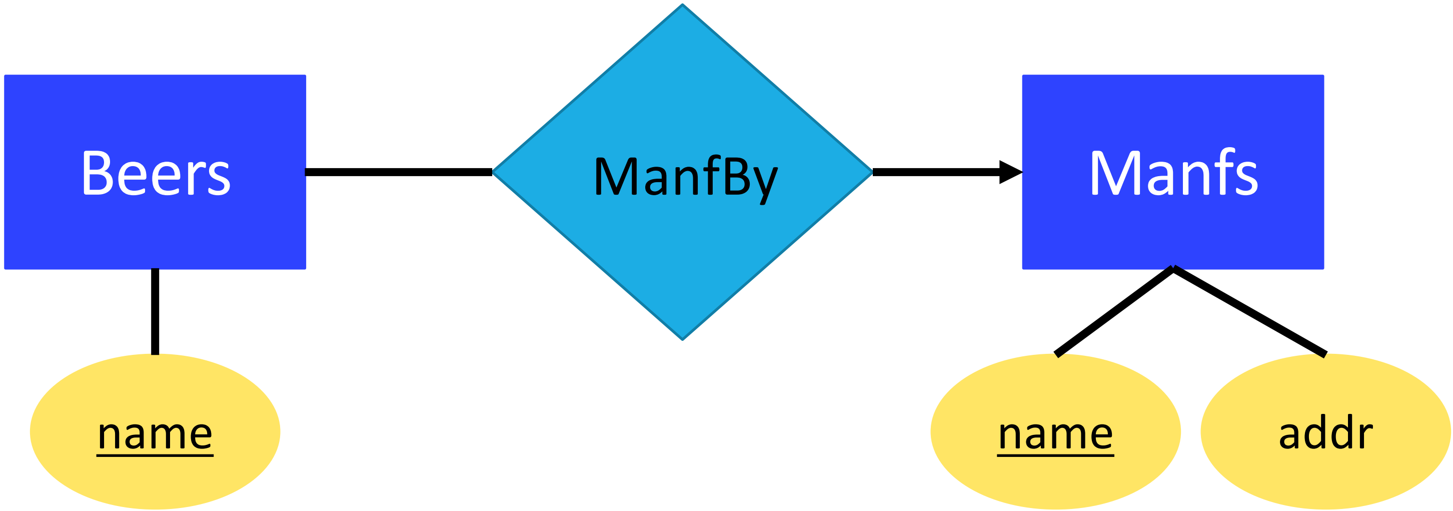
OR

- It is the "many" in a many-one or many-many relationship.

Depends on the application requirements:

- If we have several addresses per employee, *address* must be an entity (since attributes cannot be set-valued).
- If the structure (city, street, etc…) is important, e.g., we want to retrieve employees in a given city, *address* must be modeled as an entity (since attribute values are atomic).

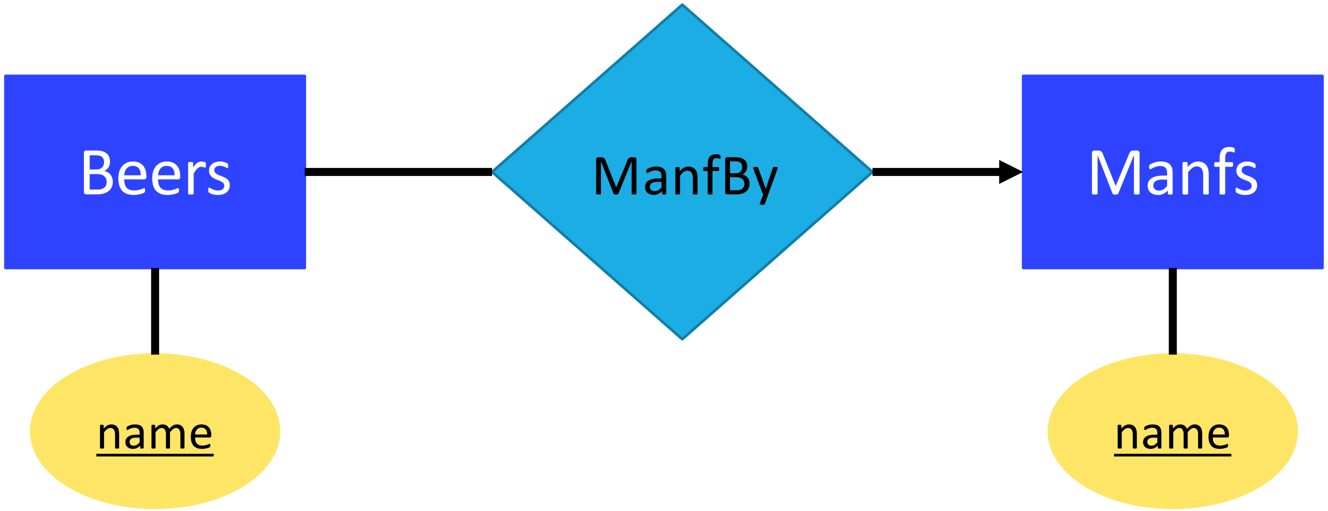
Example: Good



Manfs deserves to be an entity set because of the non-key attribute **addr**.

Beers deserves to be an entity set because it is the "many" of the many-one relationship **ManfBy**.

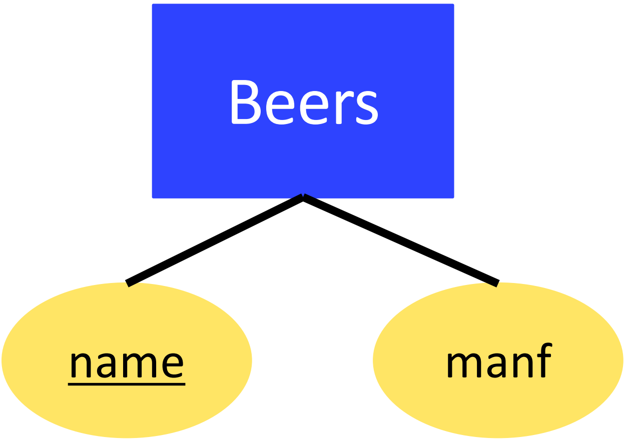
Example: Bad



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

Example: Good

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



2. Limit the use of Weak Entity Sets

Novice 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 Insurance Numbers, automobile’s VINs, etc…

When do we need Weak Entity Sets?

The usual reason is that there is no 'global authority' capable of creating unique IDs.

e.g. it is unlikely that there could be an agreement to assign an unique player number across all football teams in the world.

From E/R Diagrams to Relations

Entity Set → relation

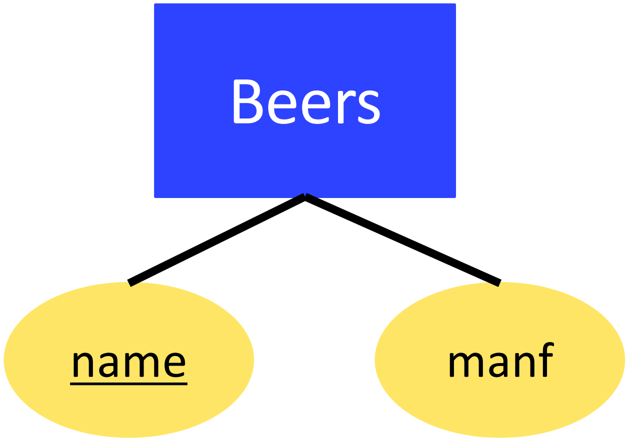
- Attributes → attributes

Relationships → relations whose attributes are only:

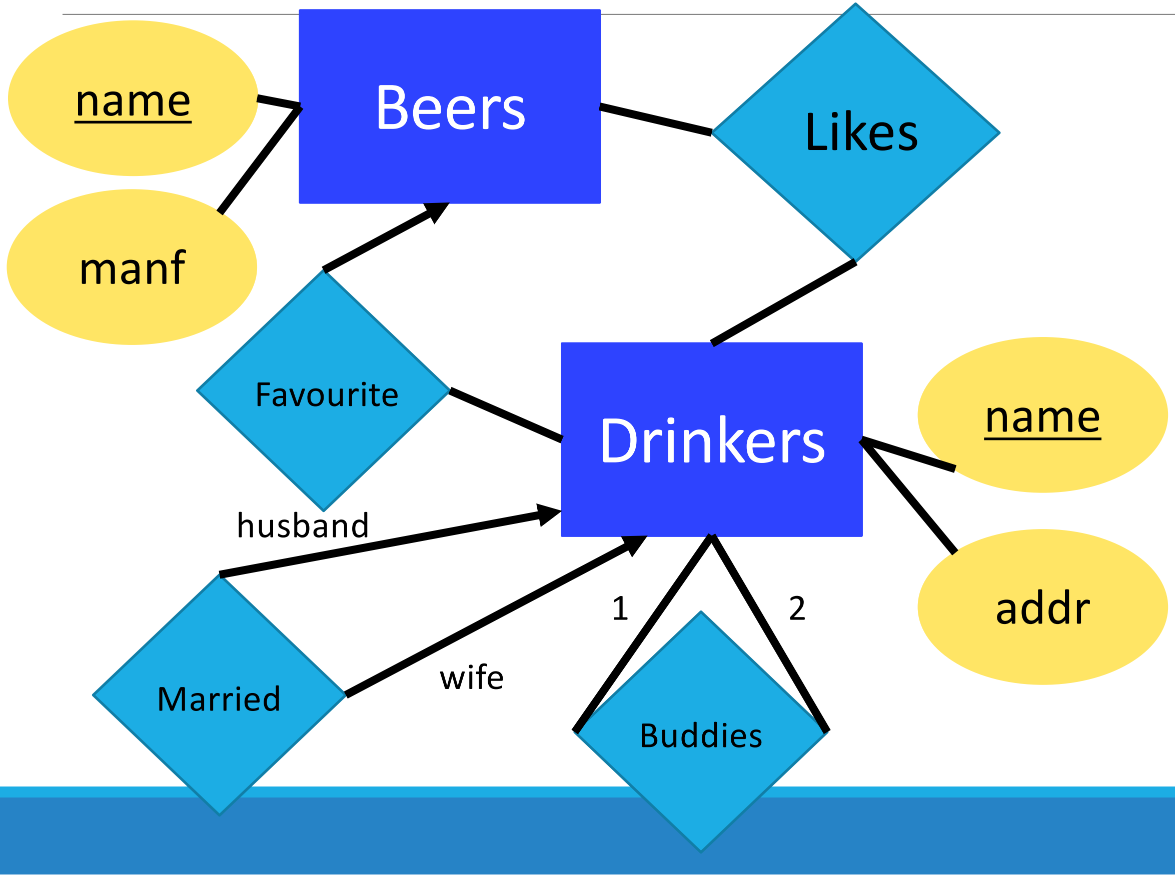
- The keys of the connected entity sets.
- Attributes of the relationship itself.

Entity Set → Relation

Relation: Beers(<u>name</u>, manf)



Relationship → Relation



Likes(drinker, beer)

Favourite(drinker, beer)

Buddies(name1, name2)

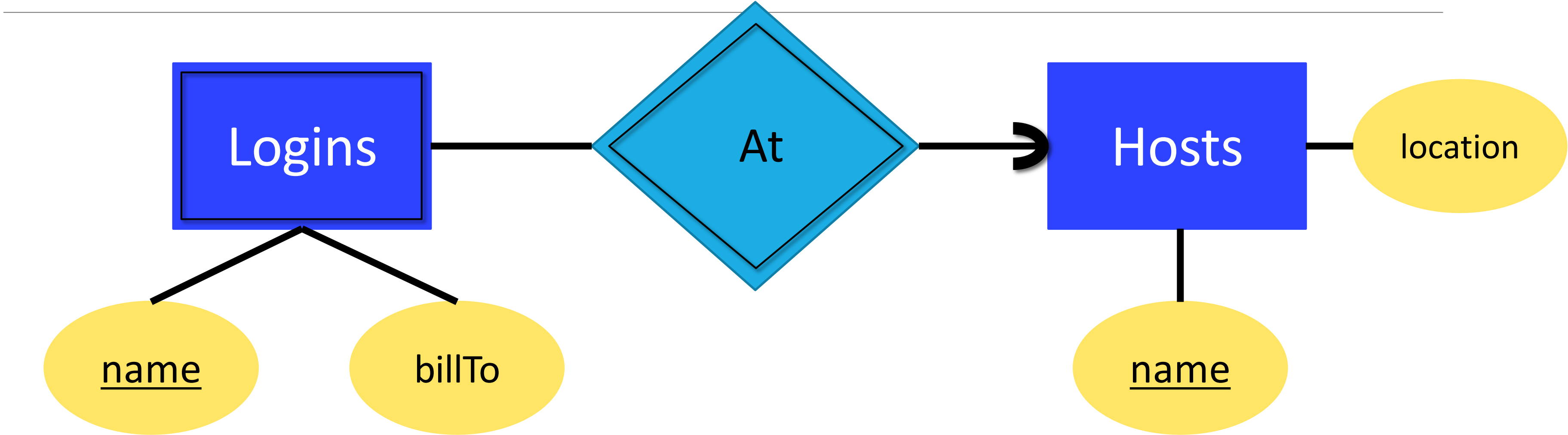
Married(husband, wife)

Handling Weak Entity Sets

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, non-key attributes.

A supporting relationship is redundant and yields no relation (unless it has attributes).

Example: Weak Entity Set → Relation

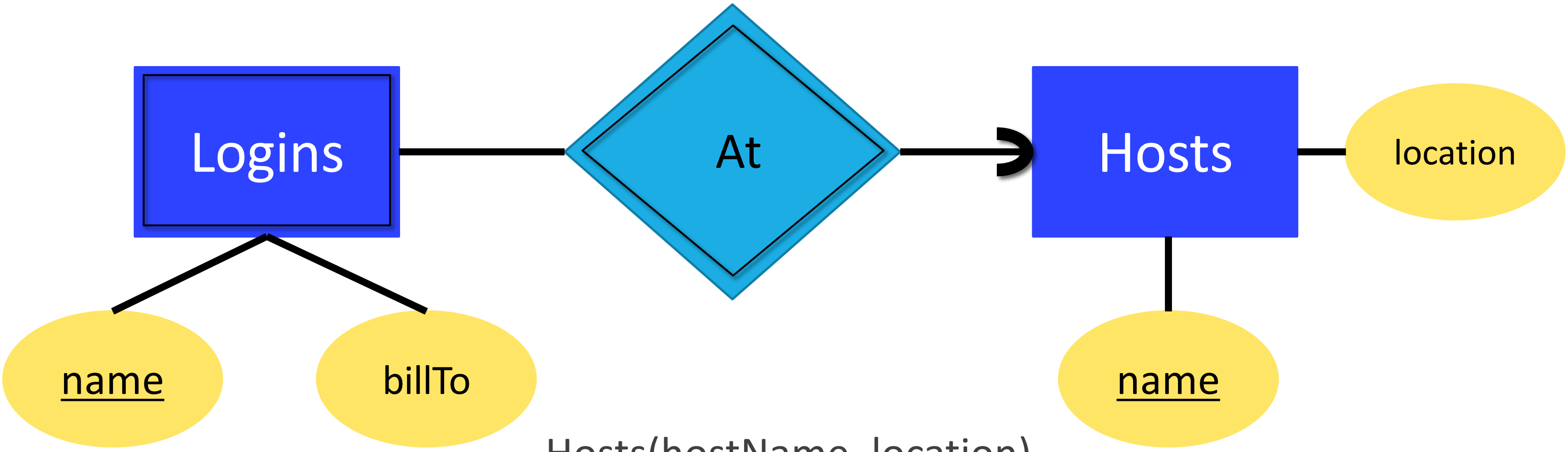


Hosts(<u>hostName</u>, location)

Logins(<u>loginName</u>, <u>hostName</u>, billTo)

At(loginName, hostName)

Example: Weak Entity Set → Relation



Hosts(<u>hostName</u>, location)

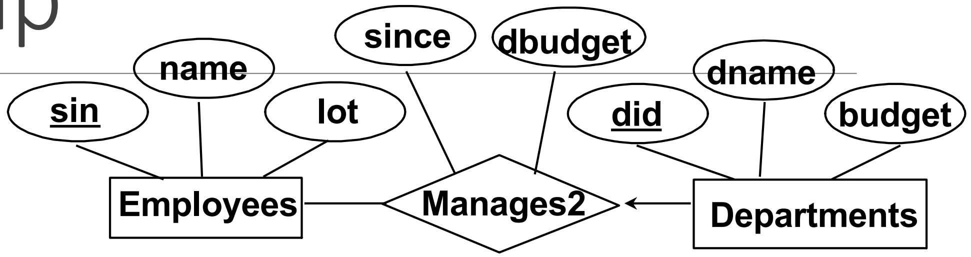
Logins(<u>loginName</u>, <u>hostName</u>, billTo)

~~At(loginName, hostName)~~

<u>Note:</u>

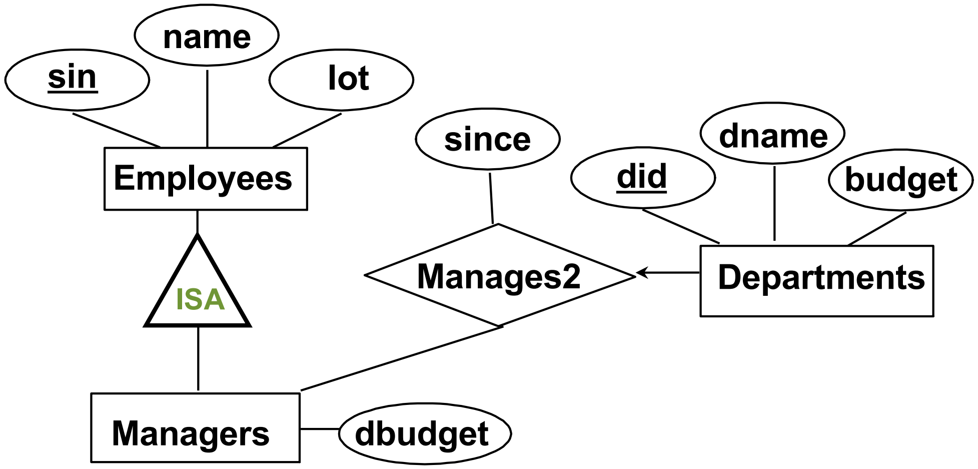
At becomes part of Logins.

Entity vs. Relationship



ER diagram is OK if a manager gets a separate discretionary budget for each Departments.

What if a manager gets a discretionary budget that covers all managed Departments?



Redundancy

- ❖dbudget stored for each dept managed by manager.

Misleading

- ❖Suggest dbudget associated with department manager combination.

Summary

Conceptual design follows requirements analysis,

- Yields a high-level description of data to be stored

ER model popular for conceptual design

- Constructs are expressive, close to the way people think about their applications.

Basic constructs: *entities*, *relationships*, and *attributes* (of entities and relationships).

Some additional constructs: *weak entities*, *ISA hierarchies*.

Summary

Several kinds of integrity constraints can be expressed in the ER model: *key constraints*, *participation constraints*, and *overlap/covering constraints* for ISA hierarchies. Some *foreign key constraints* are also implicit in the definition of a relationship set.

- Some constraints cannot be expressed in the ER model.
  - Notably, *functional dependencies* (FDs). We will talk about these later in the course.
- Constraints play an important role in determining the best database design for an enterprise.

Summary

ER design is *subjective*. There are often many ways to model a given scenario!

Analyzing alternatives can be tricky, especially for a large enterprise. Common choices include:

- Entity vs. attribute, entity vs. relationship, binary or n-ary relationship, whether or not to use ISA hierarchies
- Ensuring good database design: resulting relational schema should be analyzed and refined further.

Flowchart Building

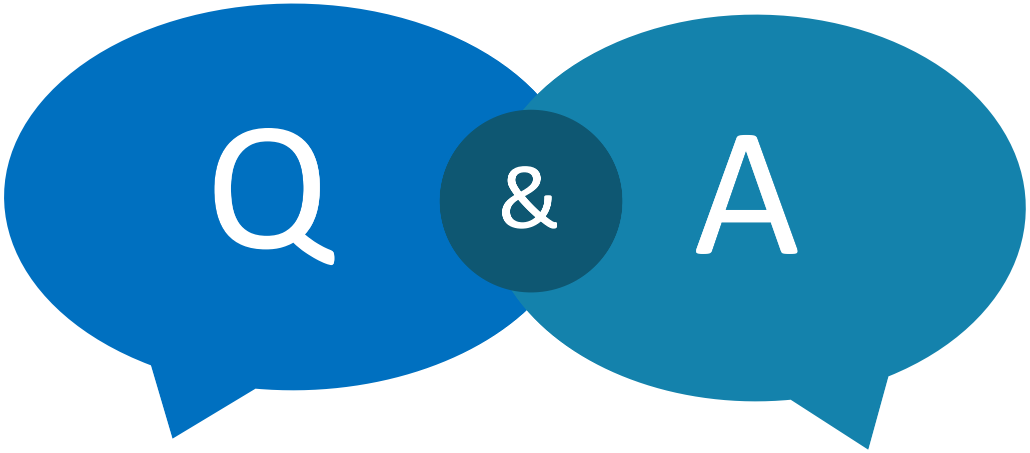
Online Tools:

- https://www.draw.io

Applications:

- OmniGraffle Pro
  - https://www.omnigroup.com/omnigraffle
- Microsoft PowerPoint or Word
- There are many more...

Questions?



THANKS FOR LISTENING

I'LL BE ANSWERING QUESTIONS NOW

Citations, Images and Resources

Database Management Systems (3rd Ed.), Ramakrishnan & Gehrke

Some content is based off the slides of Dr. Fei Chiang - http://www.cas.mcmaster.ca/~fchiang/

https://en.oxforddictionaries.com/definition/redundancy