THE ENTITY – RELATIONSHIP MODEL

The E-R Model

- The **E-R** (entity-relationship) data model views the real world as a set of basic **objects** (entities) and **relationships** among these objects.

- It is intended primarily for the DB design process by allowing the specification of an **enterprise scheme**. This represents the overall logical structure of the DB.

Entities and Entity Sets

- An **entity** is an object that exists and is distinguishable from other objects. For instance, Michelle Lee with S.S.N. 890-12-3456 is an entity, as she can be uniquely identified as one particular person in the universe.

- An entity may be **concrete** (a person or a book, for example) or **abstract** (like a holiday or a disease or a concept).

- An **entity set** is a set of entities of the same type (e.g., all persons having an account at a bank).

- Entity sets **need not be disjoint**. For example, the entity set *Student* (all students in a university) and the entity set *professor* (all professors in a university) may have members in common. (i.e a computer science professor might take a class in anthropology).

Entities and Entity Sets Continued

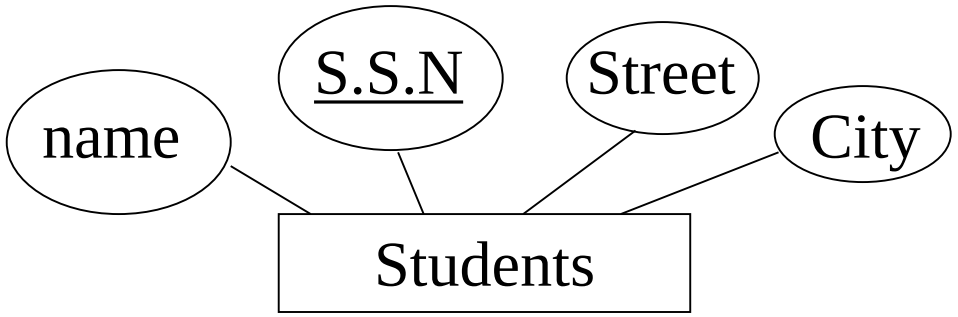
- An entity is represented by a set of **attributes**. (e.g. *name*, *SSN*, *Phone-Num* for “customer” entity.)
- The **domain** of the attribute is the set of permitted values (e.g. the telephone number must be seven positive integers).
- Formally, an attribute is a **function** which maps an entity set into a domain.
- Every entity is described by a set of (attribute, data value) pairs. There is one pair for each attribute of the entity set.
- e.g. a particular *student* entity is described by the set {(name, Lee), (SSN, 890-123-456), (street, Blaine), (city, Riverside)}.

E-R diagrams

We can express the overall logical structure of a database **graphically** with an E-R diagram.

Its components are:

- **rectangles** representing entity sets.
- **ellipses** representing attributes.
- **diamonds** representing relationship sets.
- **lines** linking attributes to entity sets and entity sets to relationship sets.



Note that this is a poor example of a entity, since the name is represented as one attribute and there is no street number attribute.
We will consider the problem of designing *good* entities later, here we are just concerned with explaining their graphical representation.

Also note that one of the attributes is underlined, we will explain why later.

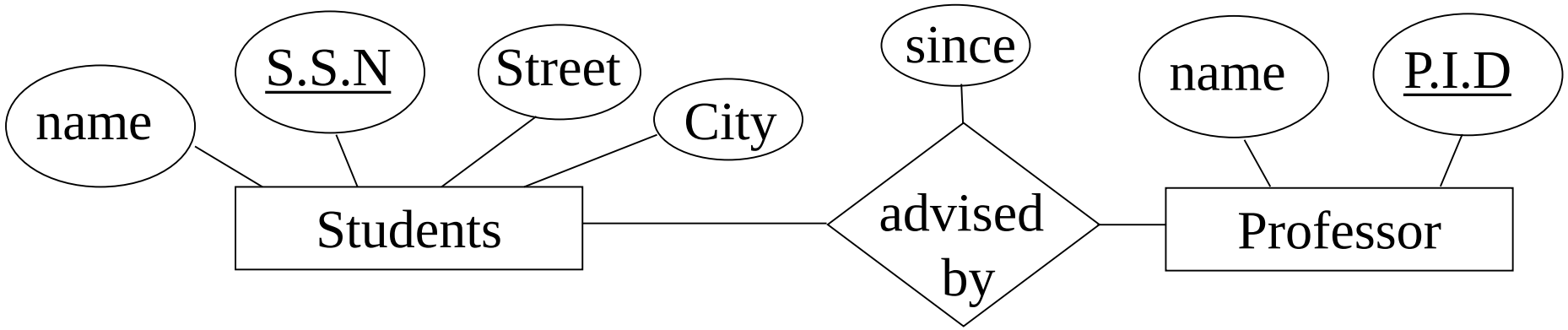
E-R diagrams Continued

We can express the overall logical structure of a database **graphically** with an E-R diagram.

Its components are:

- **rectangles** representing entity sets.
- **ellipses** representing attributes.
- **diamonds** representing **relationship** sets.
- **lines** linking attributes to entity sets and entity sets to relationship sets.

The "since" attribute in this example is called a descriptive attribute, since it describes the mapping from A to B



How to represent Constraints

Hugely important:

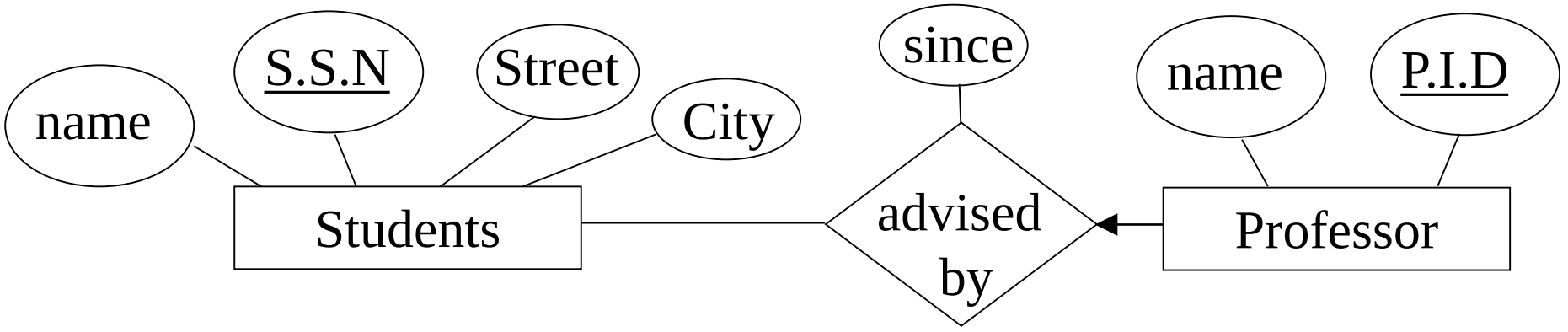
The constraints are ***given*** to us from the application

Key Constraints

- We can also use arrows to indicate **key constraints** (often simply referred to as **Cardinality Constraints**)

- Suppose the university has the following rule: A professor is allowed to advise at most one student. However two or more professors are allowed to advise the same student.

- This is an example of a **one-to-many constraints**, that is *one* student can be advised by *many* professors. We can represent this with an arrow as shown below.



Key Constraints Continued

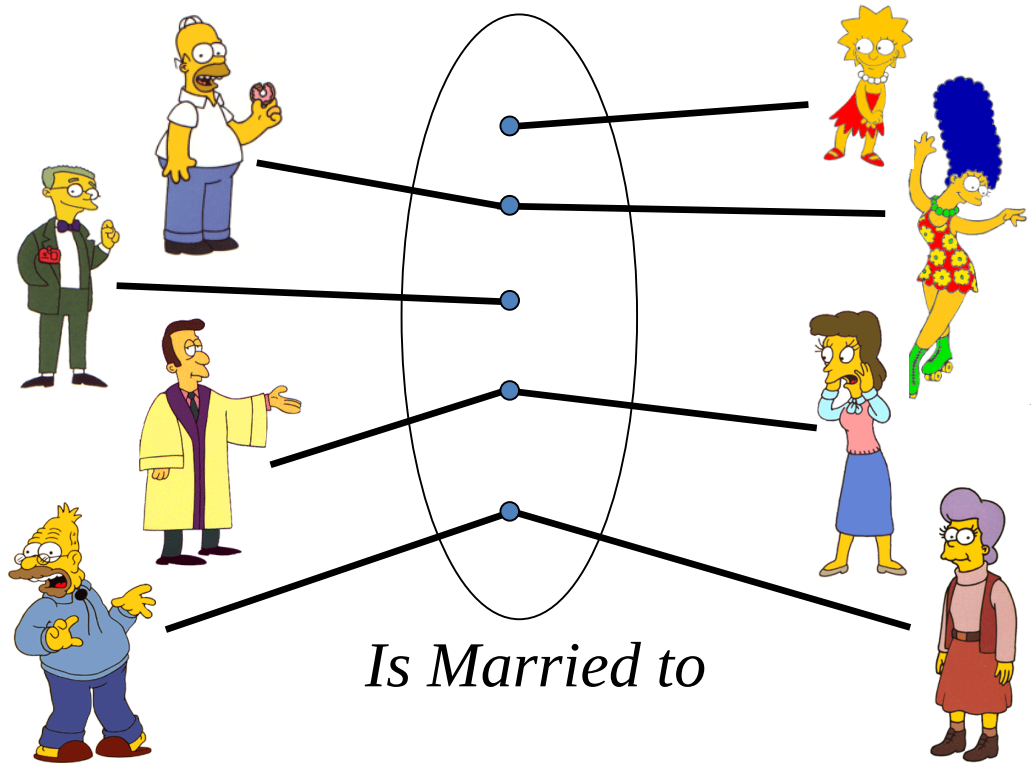
- There are four possible **key constraints,** they express the number of entities to which another entity can be associated via a relationship. For binary relationship sets between entity sets A and B, the mapping cardinality must be one of:

1. **One-to-one**: An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.
2. **One-to-many**: An entity in A is associated with any number in B. An entity in B is associated with at most one entity in A.
3. **Many-to-one**: An entity in A is associated with at most one entity in B. An entity in B is associated with any number in A.
4. **Many-to-many**: Entities in A and B are associated with any number from each other.

The appropriate **key constraint** for a particular relationship set depends on the real world being modeled.

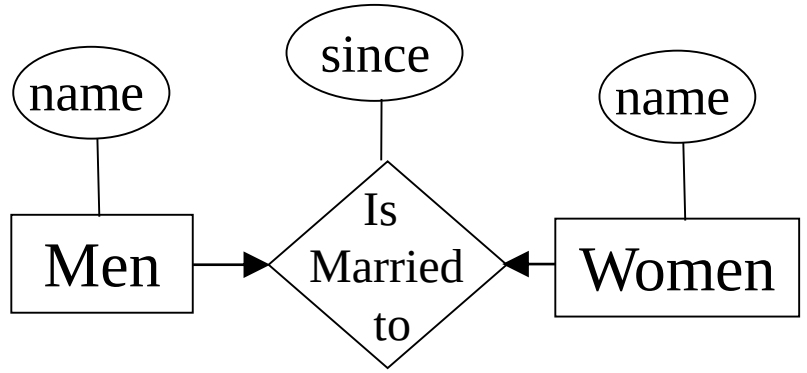
Key Constraints: Examples

- **One-to-one**: An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.



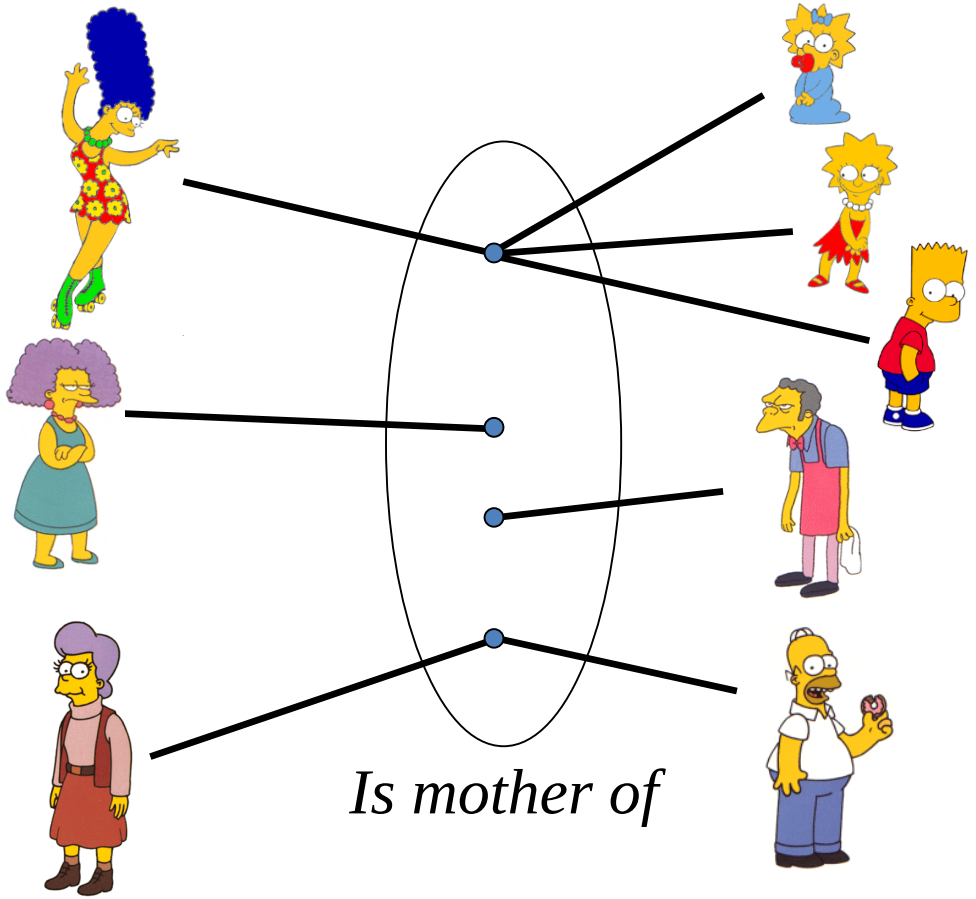
This diagram is not a part of the ER model! It is just an intuitive picture to explain a concept

Assume this holds: A man may be married to at most one women, and woman may be married to at most one man (both men and women can be unmarried)

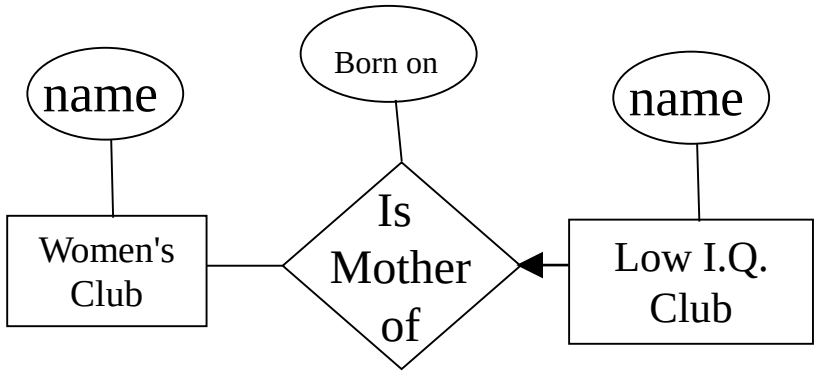


Key Constraints: Examples

- **One-to-many**: An entity in A is associated with any number in B. An entity in B is associated with at most one entity in A.



Assume this holds: A woman may be the mother of many (or no) children. A person may have at most one mother.

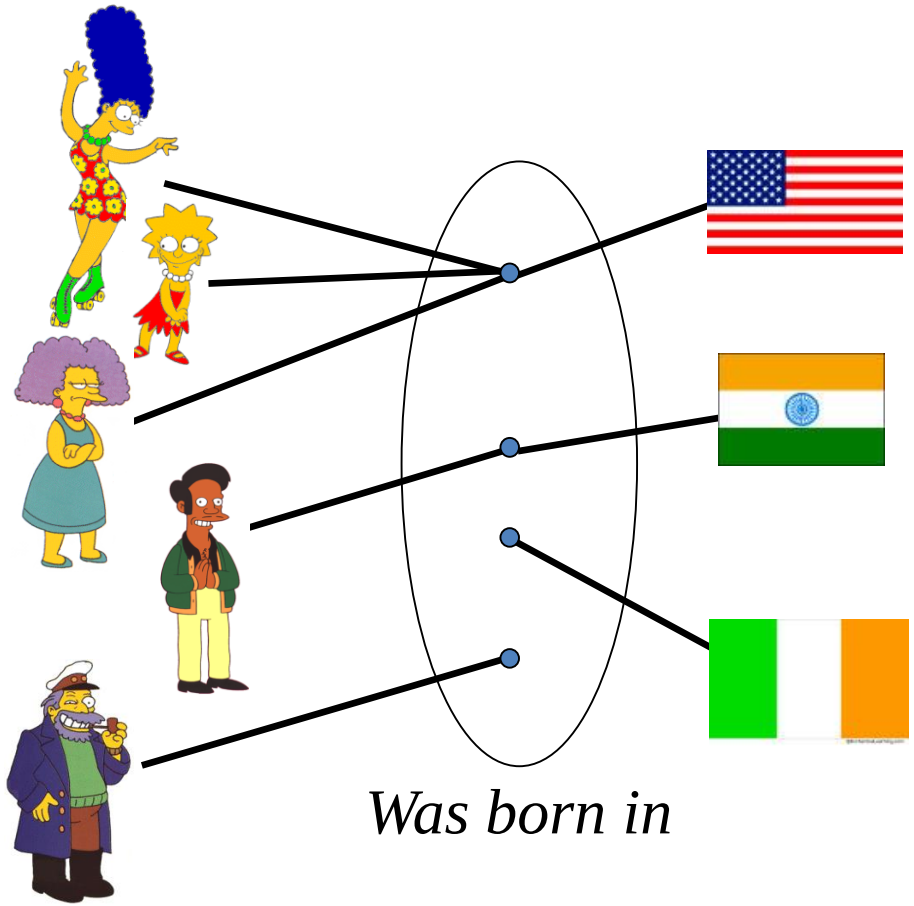


Note that this example is not saying that Moe does not have a mother, since we know as a biological fact that everyone has a mother.

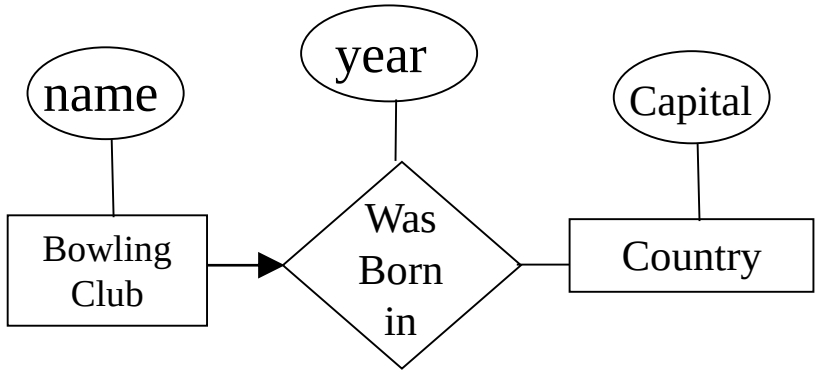
It is simply the case that Moe's mom is not a member of the Women's club.

Key Constraints: Examples

- **Many-to-one**: An entity in A is associated with at most one entity in B. An entity in B is associated with any number in A.



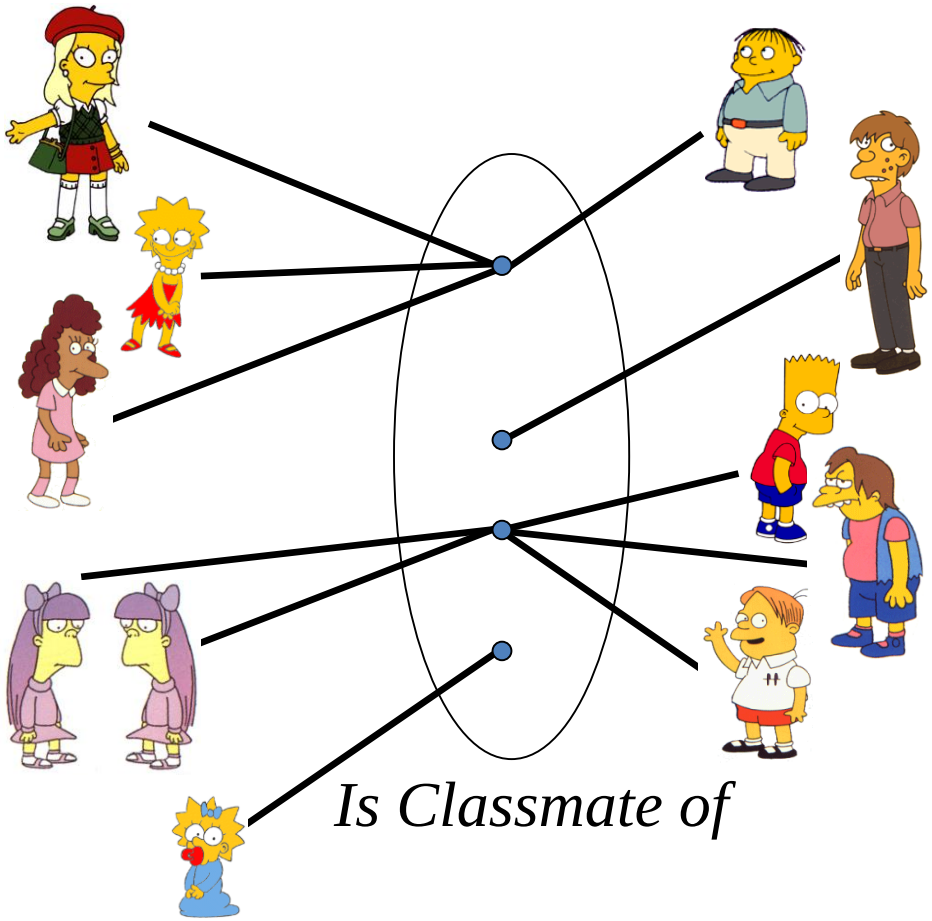
Assume this holds: Many people can be born in any county, but any individual is born in at most one country.

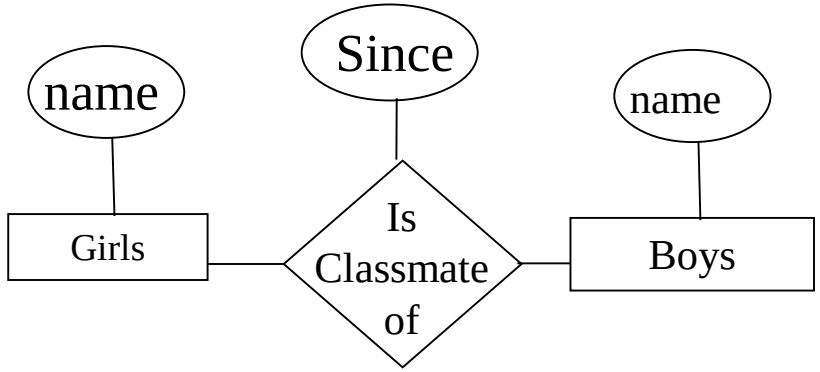


Note that we are not saying that the Sea Captain was not born in any country, he almost certainly was, we just don't know which country, or it is not in our Country entity set.
Also note that we are not saying that no one was born in Ireland, it is just that no one in the Bowling Club was.

Key Constraints: Examples

- **Many-to-many**: Entities in A and B are associated with any number from each other.





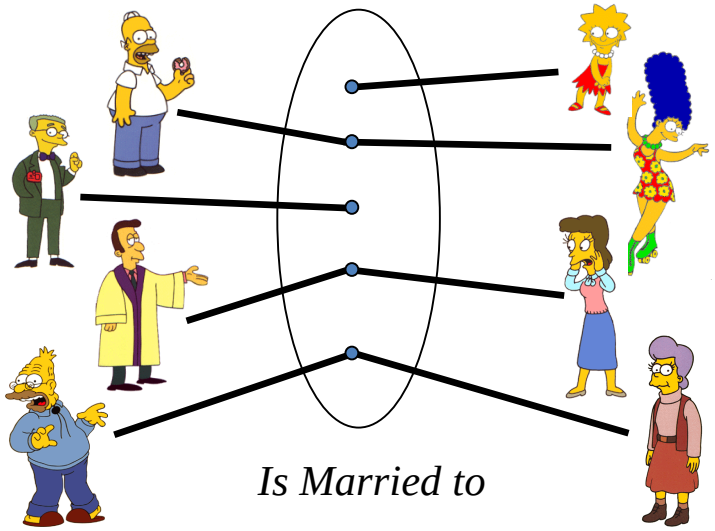
Key Constraints

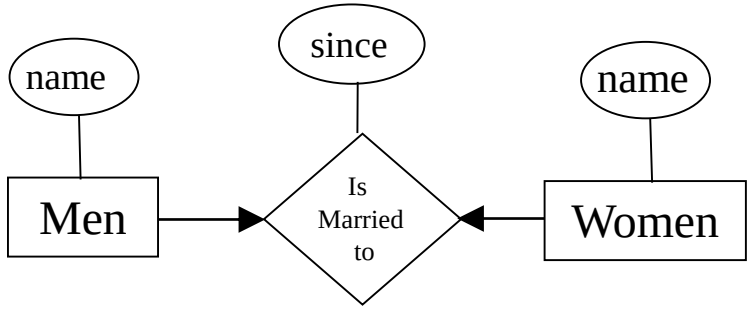
- The arrow positioning is simple once you get it straight in your mind, so do some examples. Think of the arrow head as pointing to the entity that “one” refers to.

- Some people call use the term “**Mapping Cardinalities**” to refer to key constraints.

Participation Constraints

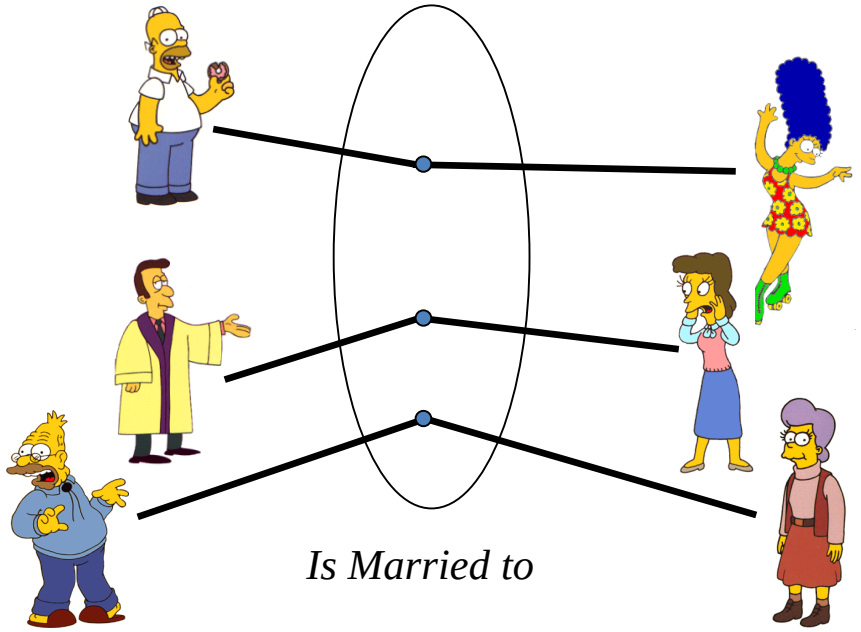
- Earlier, we saw an example of a one-to-one key constraint, noting that a man may be married to at most one women, and woman may be married to at most one man (both men and women can be unmarried).
- Suppose we want to build a database for the “Springfield Christian Married Persons Association”. In this case *everyone* must be married! In database terms their participation must be **total**. (the previous case that allows unmarried people is said to have **partial** participation.
- How do we represent this with ER diagrams? (answer on next slide)

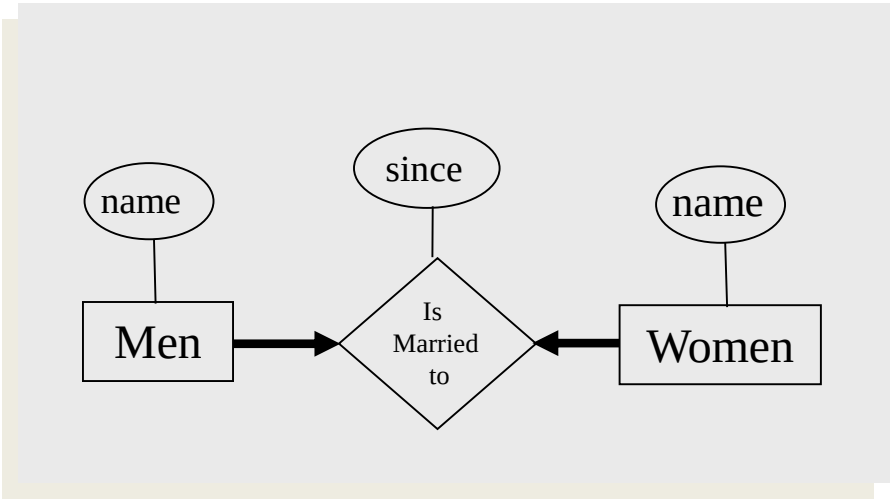




Participation Constraints Cont.

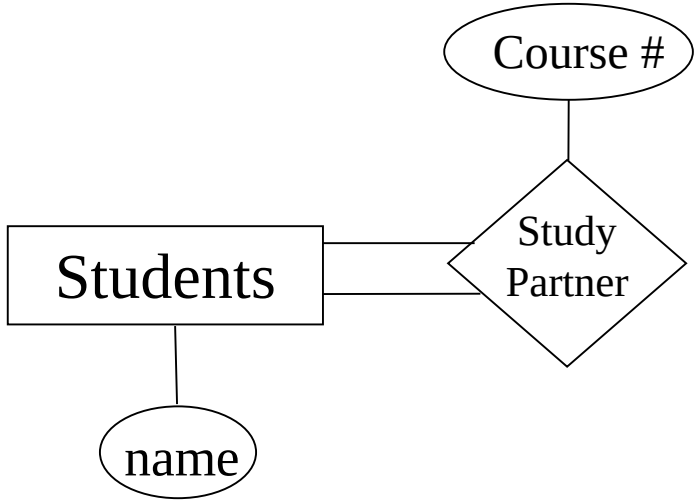
- Participation Constraints are indicated by bold lines in ER diagrams.
- We can use bold lines (to indicate participation constraints), and arrow lines (to indicate key constraints) independently of each other to create an expressive language of possibilities.



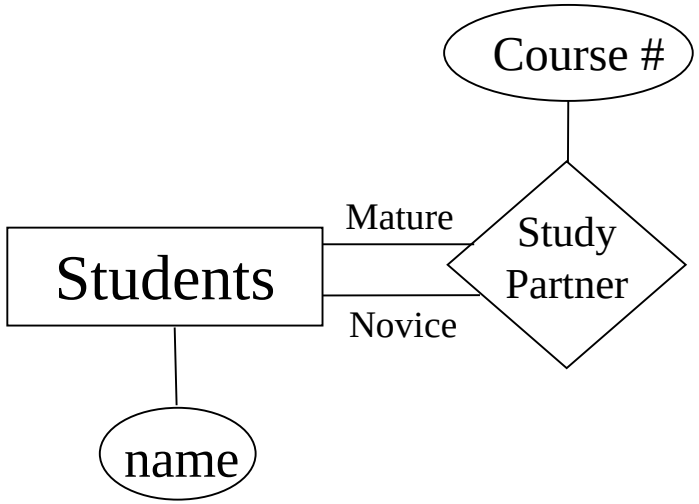


More on Relations

- Entities sets can be related to themselves.



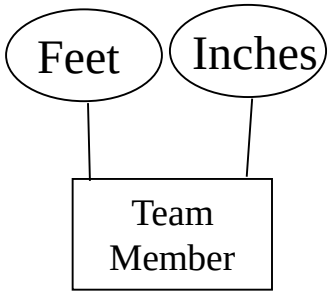
We can annotate the **roles** played by the entities in this case. Suppose that we want to pair a mature student with a novice student...

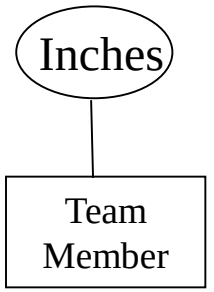


When entities are related to themselves, it is almost always a good idea to indicate their roles.

Entity Attributes Revisited

- What is the correct choice of attributes to represent height?
  - Using two fields, one for *feet* the other for *inches* is probably the best solution if we are never going to do arithmetic on the height, and you need to report the height in a human intuitive way (dating agency).
  - If we are going to do calculations on the data (i.e calculate the BMI) we would be much better off with just the height in inches. (medical records)

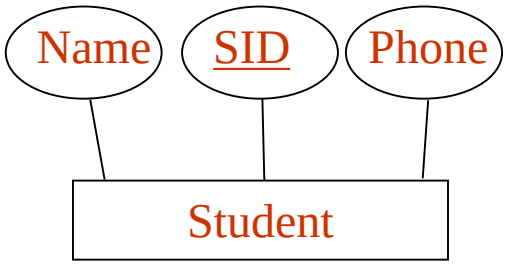


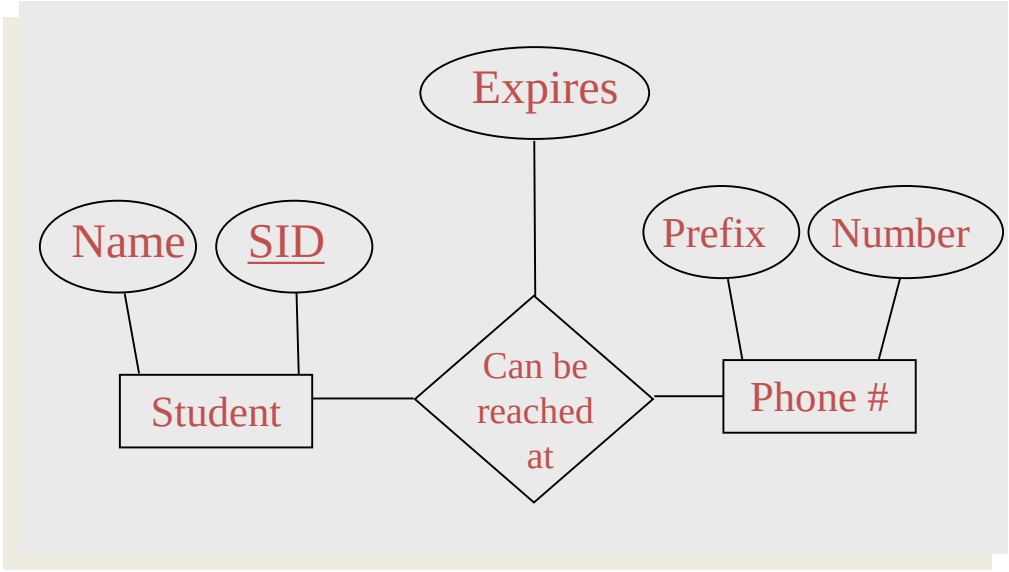


Entity versus Attribute

- Sometimes we have to decide whether a property of the world we want to model should be an attribute of an entity, or an entity set which is related to the attribute by a relationship set.

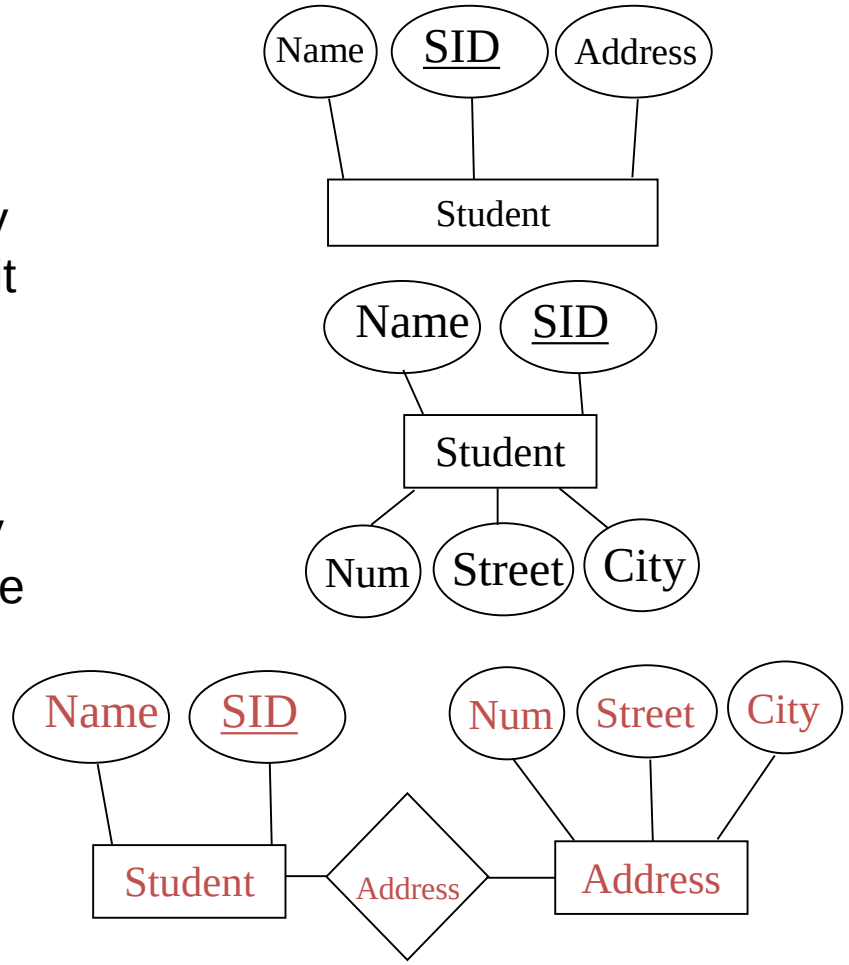
- A major advantage of the latter approach is that we can easily model the fact that a person can have multiple phones, or that a phone might be shared by several students. (attributes can not be set-valued)





Entity versus Attribute Cont.

- A classic example of a feature that is best modeled as a an entity set which is related to the attribute by a relationship set is an *address*.
- Very bad choice for most applications. It would make it difficult to pretty print mailing labels, it would make it difficult to test validity of the data, it would make it difficult/impossible to do queries such as “how many students live in riverside?”
- A better choice, but it only allows a student to have one address. Many students have a two or more address (i.e. a different address during the summer months) This method cannot handle this.
- The best choice for this problem



Domains Revisited

- We already said…
  - The **domain** of the attribute is the set of permitted values (e.g. the telephone number must be seven positive integers).

- Defining the correct **domain** for an attribute is an important skill.

- Making a mistake at this stage can cause huge problems later on.

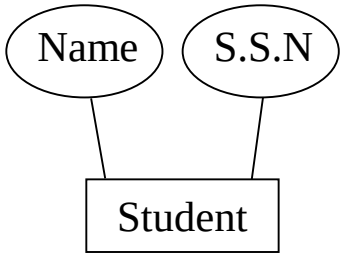
Domains Revisited Cont.

- What is the correct domain for GPA?
- What is the correct domain for street address?
- What is the correct domain for Phone Number? (What about foreign …)
- What is the correct domain for Bank Account Balance?

Keys

Differences between entities must be expressed in terms of attributes.

- A **superkey** is a set of one or more attributes which, taken collectively, allow us to identify uniquely an entity in the entity set.
- For example, in the entity set *student*; *name* and *S.S.N.* is a superkey.
- Note that *name* alone is not, as two students could have the same name.
- A superkey may contain extraneous attributes, and we are often interested in the smallest superkey. A superkey for which no subset is a superkey is called a **candidate key**.



Name	S.S.N
Lisa	1272
Bart	5592
Lisa	7552
Sue	3532

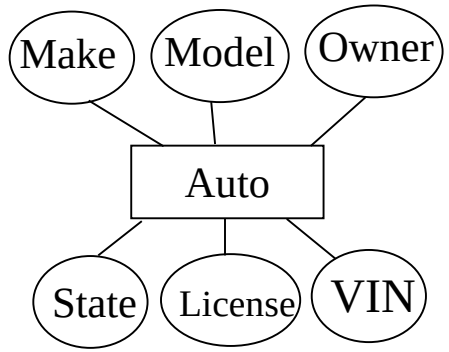
We can see that {*Name*,*S.S.N*} is a **superkey**

In this example, *S.S.N.* is a **candidate** key, as it is minimal, and uniquely identifies a students entity.

Keys Cont.

- A **primary key** is a candidate key (there may be more than one) chosen by the DB designer to identify entities in an entity set.

In the example below…



{Make,Model,Owner,State,License#,VIN#} is a superkey
{State,License#,VIN#} is a superkey
{Make,Model,Owner} is *not* a superkey

{State,License#} is a candidate key
{VIN#} is a candidate key

VIN# is the logical choice for **primary key**

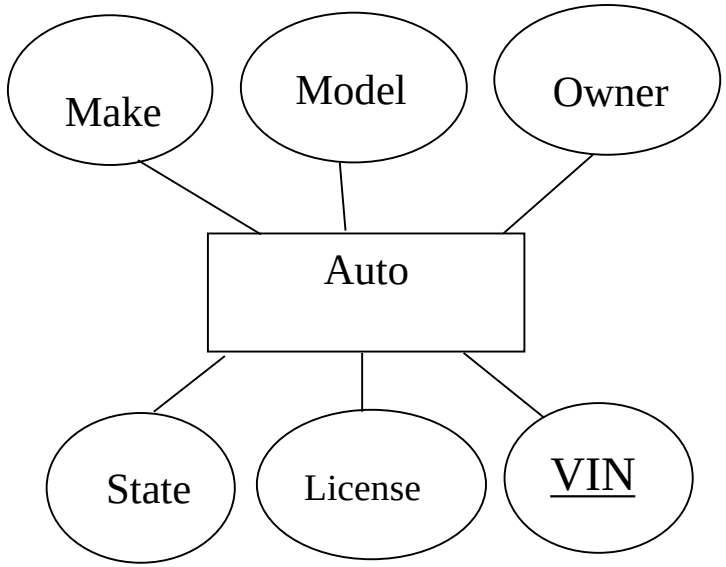
Make	**Model**	**Owner**	**State**	**License #**	**VIN #**
Ford	Focus	Mike	CA	SD123	34724
BMW	Z4	Joe	CA	JOE	55725
Ford	Escort	Sue	AZ	TD4352	75822
Honda	Civic	Bert	CA	456GHf	77924

Keys Cont.

- The **primary key** is denoted in an ER diagram by underlining.
- An entity that has a primary key is called a **strong entity**.

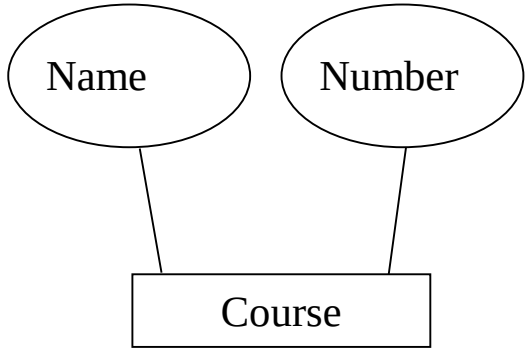
Note that a good choice of primary key is very important!

For example, it is usually much faster to search a database by the primary key, than by any other key (we will see why later).



Keys Cont.

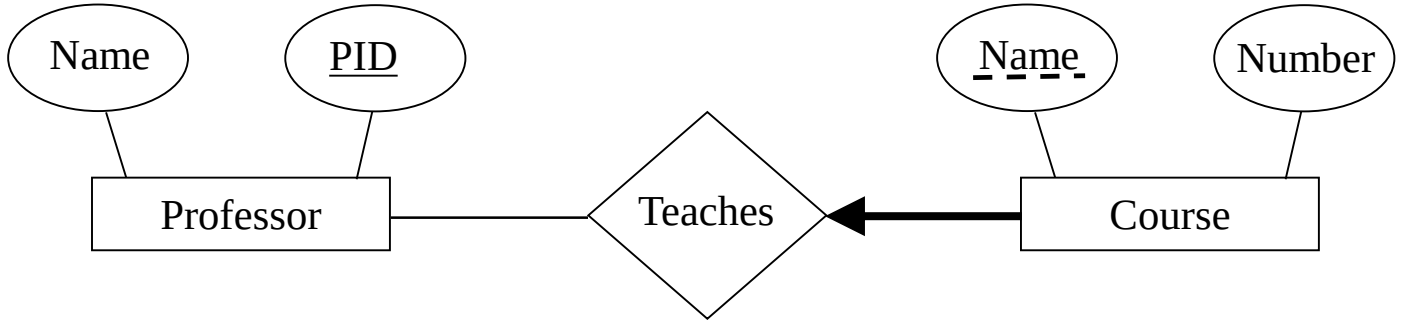
- An entity set that does not possess sufficient attributes to form a primary key is called a **weak entity set**.
- In the example below there are two different sections of C++ being offered (lets say, for example, one by Dr. Smith, one by Dr. Lee).
  - {Name,Number} is not a superkey, and therefore *course* is a **weak entity**.
- This is clearly a problem, we need some way to distinguish between different courses….



Name	Number
C++	CS12
Java	CS11
C++	CS12
LISP	CS15

Keys Cont

- In order to be able to uniquely refer to an item in a weak entity set we must consider some (or all) of its attributes in conjunction with some strong entities primary key. The entity whose primary key is being used is called the **identifying owner**.



For this to work, two conditions must be met.

- The weak entity set must have total participation in the relationship
- The identifying owner and the weak entity must participate is a one-to-many relationship.

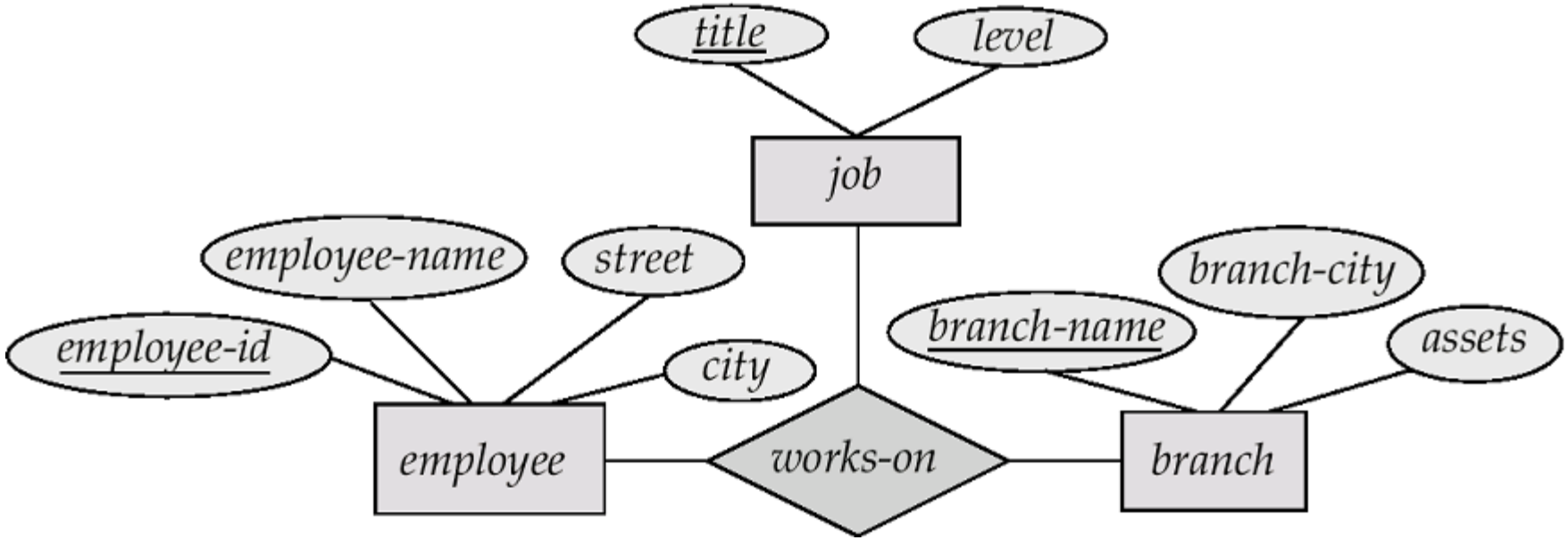
Name	PID
Smith	2345
Lee	7356
Chan	3571

Name	Number
C++	CS12
Java	CS11
C++	CS12
LISP	CS15

Ternary Relationships

- So far, we have only considered binary relationships, however it is possible to have **higher order** relationships, including **ternary** relationships.

- Consider the following example that describes the fact that employees at a bank work in one or more bank branches, and have one or more job descriptions.



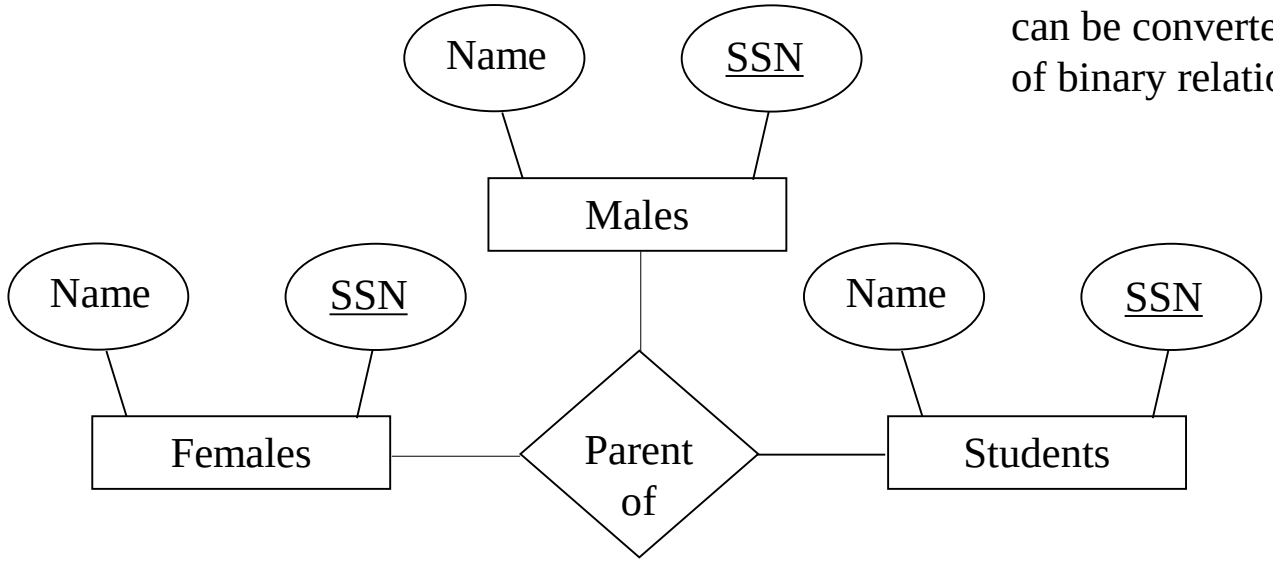
Why not remove the *job* entity by placing the *title* and *level* attributes as part of employee?

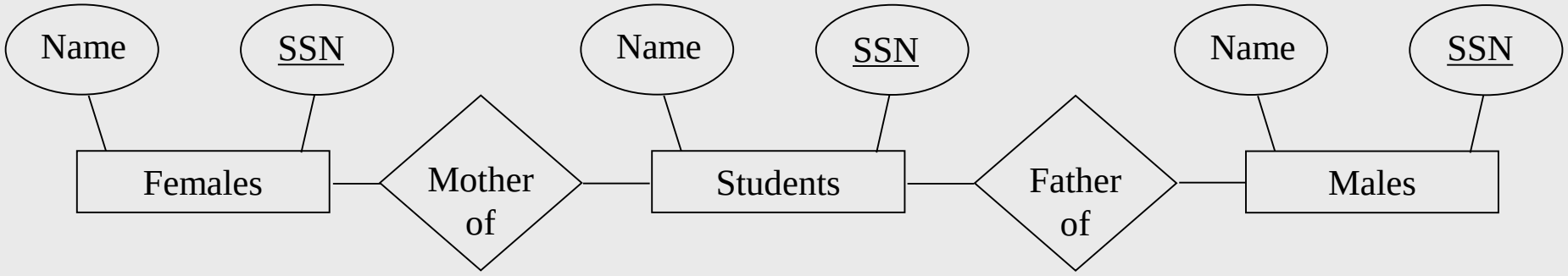
Ternary Relationships

- Sometimes you have a choice of a single ternary relationship or two binary relationships…

In general, unless you really need a ternary relationship, use binary relationships.

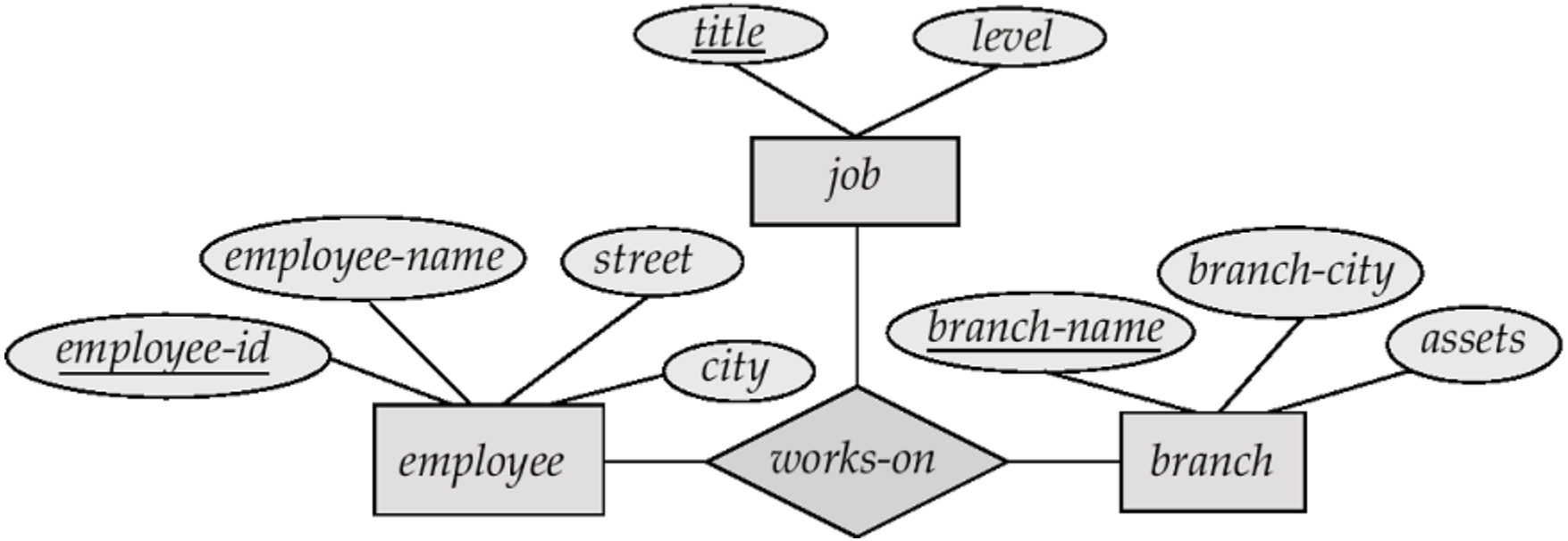
FACT: Every ternary (and higher order) relationship can be converted into a set of binary relationships.





Aggregation

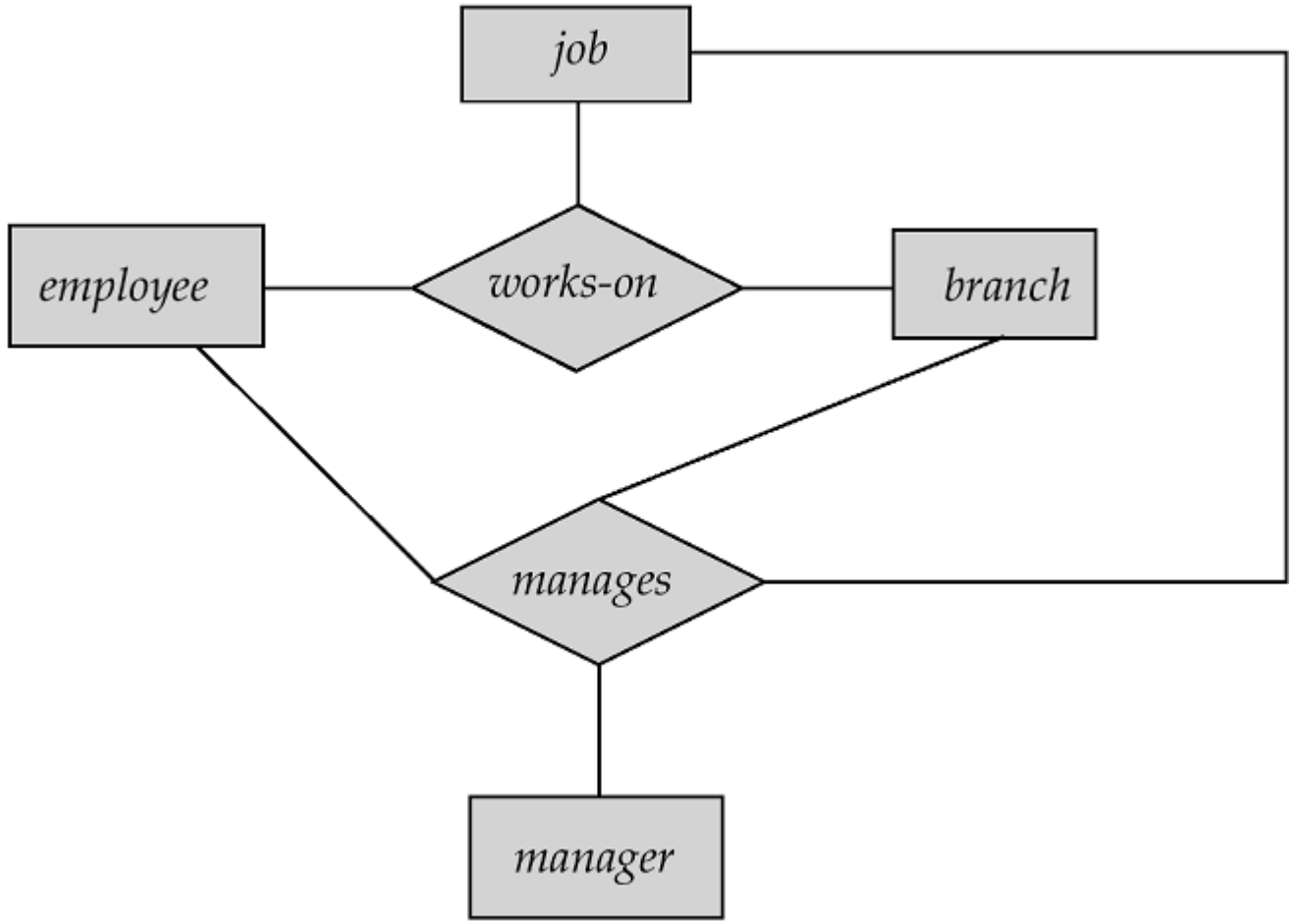
- Consider this ER model, which we have seen before…
- We need to add to it, to reflect that managers manage the various tasks performed by an employee at a branch



Aggregation Cont.

- Relationship sets *works-on* and *manages* represent overlapping information
- Every *manages* relationship corresponds to a works-on relationship
- However, some *works-on* relationships may not correspond to any manages relationships
- So we can't discard the *works-on* relationship

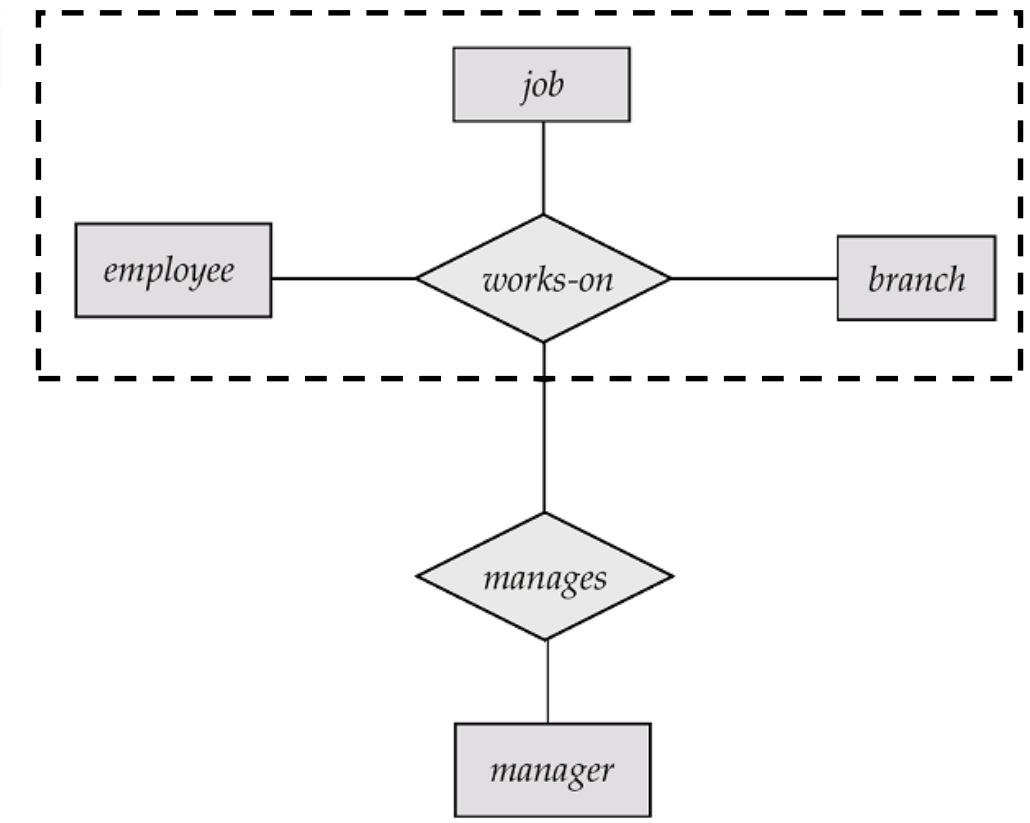
Note that the attributes are omitted for graphical simplicity.



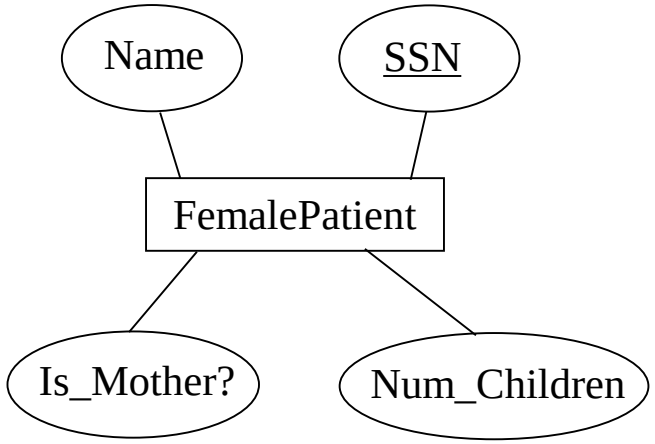
Aggregation Cont.

We can eliminate this redundancy via aggregation
- Allows relationships between relationships
- Abstraction of relationship into new entity

- Without introducing redundancy, the new diagram represents:
  - An employee works on a particular job at a particular branch
  - An employee, branch, job combination may have an associated manager.



Redundancy is an enemy



What's wrong with this ER Model?

ER Design Decisions

- The use of an attribute or entity set to represent an object.
- Whether a real-world concept is best expressed by an entity set or a relationship set.
- The use of a ternary relationship versus a pair of binary relationships.
- The use of a strong or weak entity set.
- The use of aggregation – can treat the aggregate entity set as a single unit without concern for the details of its internal structure.

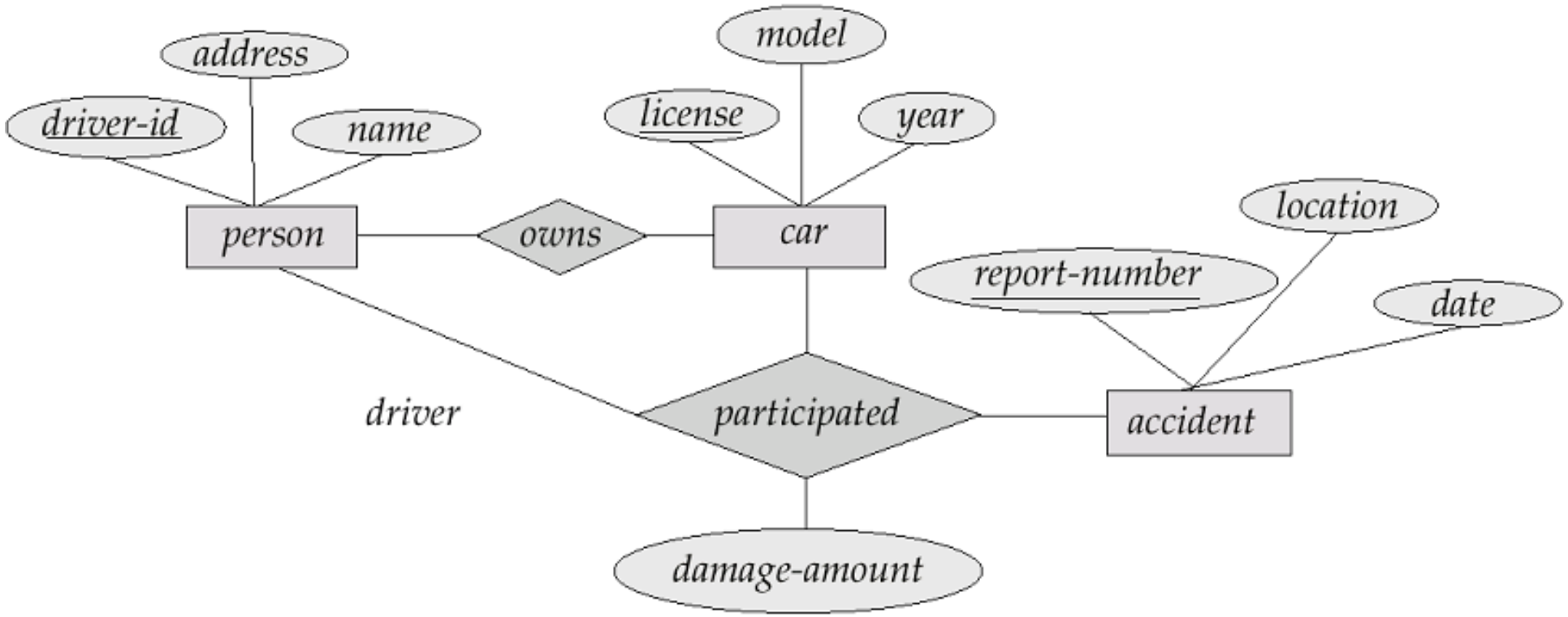
Next Time

- How to convert an ER diagram into a relational database, and introduction to relational algebra.

Study Suggestion

It is tempting to think you understand the many concepts introduced here because you understand the examples. To fully understand reinforce your understanding, you should try to come up with several more examples of each concept.

- Think up 3 examples of a many to one relationship
- Think up 3 examples of a weak entity set.
- Think up 3 examples of a …


address
driver-id
name
person
owns
model
license
year
car
location
report-number
date
driver
participated
accident
damage-amount