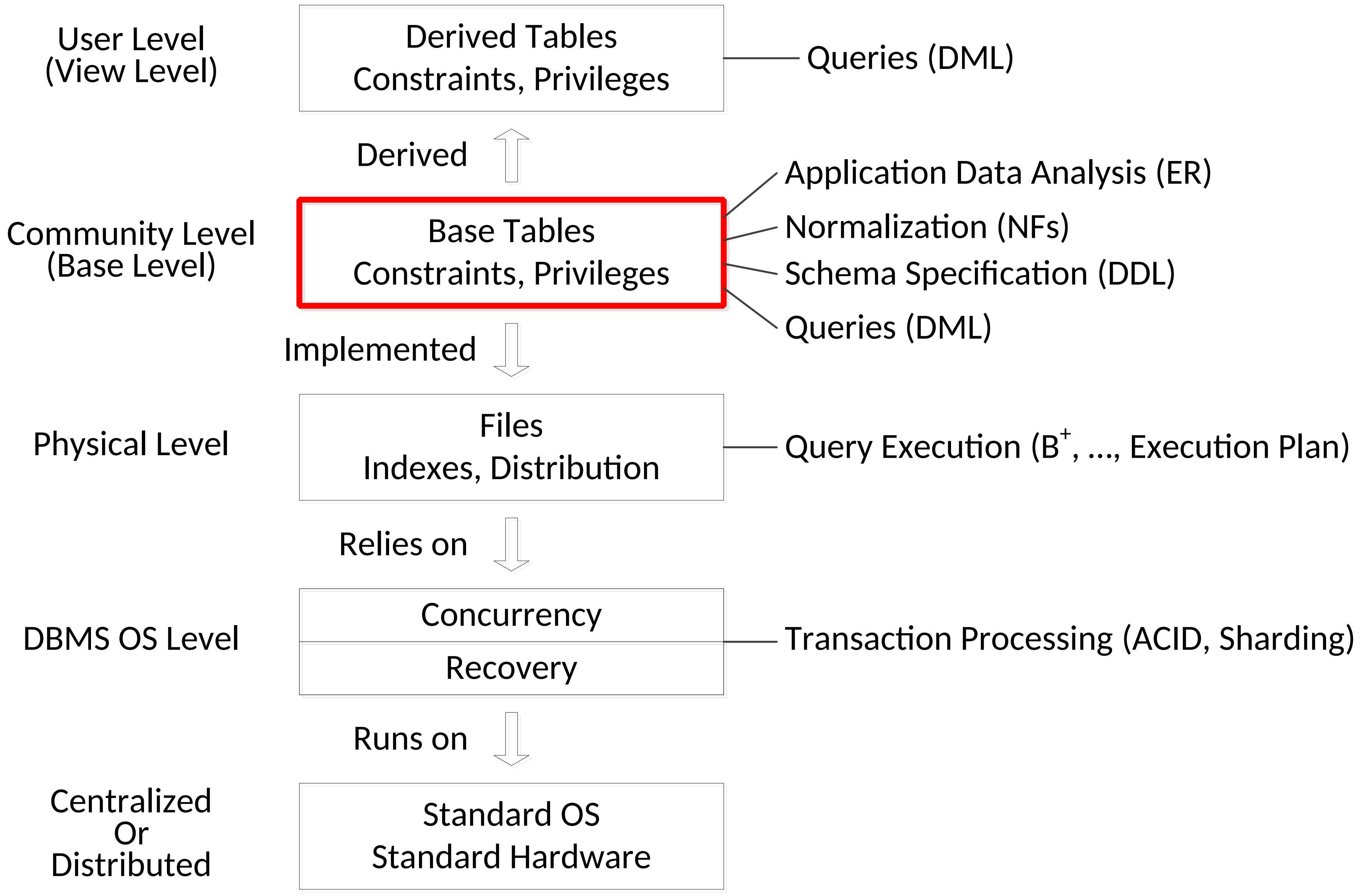
Unit 2
Modeling the Information of an Enterprise Using Enhanced Entity/Relationship Diagrams

ER Diagrams in Context



Introduction

Purpose of ER Model and Basic Concepts

- ***Entity/relationship (ER) model*** provides a common, informal, and convenient method for communication between ***application end users (customers) and the database designers*** in order to model the information's structure
- This is ***conceptual design***, though some call this ***logical design***
- This is a preliminary stage towards defining the database using a formal model, such as the relational model, to be described later
- The ER model, typically employs ***ER diagrams***, which are pictorial descriptions to visualize information's structure
- ER models are well-suited for database designs and are both ***simple*** and ***powerful***
- ***But whatever needs to specified about the application that cannot be specified using ER diagrams, needs to be stated as textual comments***

Purpose of ER Model and Basic Concepts

- There are three basic concepts appearing in the original ER model, which has since been enhanced[
  - We will present the model from simpler to more complex concepts, with examples on the way
- We will go beyond the original ER model, and cover a version of the ***Enhanced ER (EER)*** model, while still referring to as an ER model, as is customary
- While the ER model's ***concepts are standard***, there are several ***varieties of pictorial representations*** of ER diagrams
  - We will focus on one of them: ***Chen's*** notation
  - We will also cover ***Crow's foot*** notation, which is particularly suited for relational database design
  - Others are simple variations, so if we understand the above, we can easily understand all of them
- You can look at some examples at: http://en.wikipedia.org/wiki/Entity-relationship_model

Basic Concepts

- The three basic concepts are (elaborated on very soon):
- ***Entity***. This is an "object/thing." Cannot be defined even close to a formal way but must be identifiable. Examples of identification:
  - Zvi Kedem (assuming that's unique)
  - The eldest child of Priyanka Laskshminarayan (assuming that Priyanka Laskshminarayan is unique)
  - Person with NYU Net Id abc1234
  - Los Angeles, CA, USA
  - The country whose capital is Paris
- ***Relationship***. Entities participate in relationships with each other. Examples:
  - Alice and Boston are in relationship Likes (Alice Likes Boston)
  - Bob and Atlanta are not in this relationship
- ***Attribute*** (property). Examples:
  - Age is an attribute of persons
  - Size is an attribute of cities

Entity

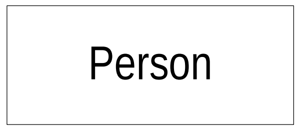
Entity and Entity Set

- ***Entity*** is a "thing" that is distinguished from others in our application
  - Example: Alice
- All entities of the same "type" form an ***entity set***; we use the term "type" informally
  - Example: Person (actually, a set of persons). Alice is an entity in this entity set
- What types (entity sets) are appropriate for an application may be a little tricky sometimes

- We will see examples soon

Entity and Entity Set

- Pictorially, an entity set is denoted by a ***rectangle*** with its type written inside
  - It contains the relevant entities
  - Example: Person contains all the “relevant” individual persons
- By convention, singular noun, though we may not adhere to this convention if not adhering to it makes things clearer
- By convention capitalized or all capitals if acronym



- We will frequently use the term “entity” while referring to entity sets, unless that causes confusion

Entity and Entity Set

- Example. When we say, “the set of all Boeing airplanes,” is this
  - The set of all the models in Boeing’s catalog (abstract objects), or
  - The set of airplanes that Boeing manufactured (concrete objects)?
- We may be interested in both and have two entity sets with appropriate “relations” between them

Entity and Entity Set

- Example: Do we partition people by sex or not?
- We will assume in the course that each Person is of exactly one of two sexes
  - Woman
  - Man
- We do this to have simple but rich enough examples
- Do we have entity set Person or two entity sets: Woman and Man?
- If our client is the U.S. Department of State and wants us to design a database of people renewing their passports with the input form https://eforms.state.gov/Forms/ds82.pdf; it requires sex to be one of the two
  - F
  - M
- But the applicants are (likely) not treated differently by sex

Entity and Entity Set

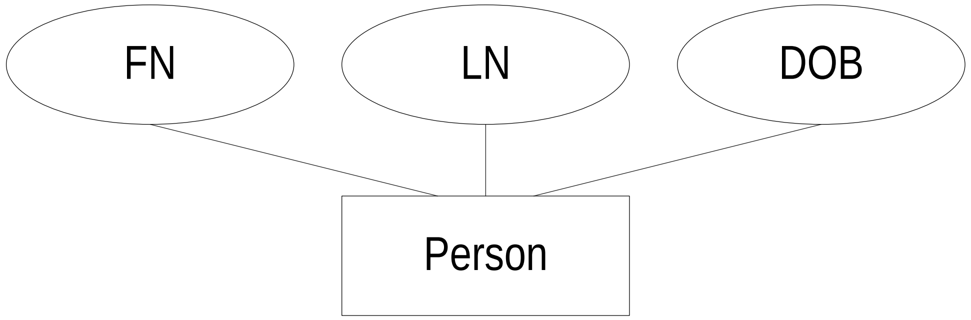
- Example. Do we partition people by sex or not?
- Do we have entity set Person or several sex-defined entity sets, such as (biological) Man and (biological) Woman?
  - Sometimes maybe makes sense: check biological men for prostate cancer

    This allows better enforcement of constraints. You could "automatically" make sure that only entities in the entity set of (biological) men could be scheduled for such a test
  - Sometimes maybe does not make sense: job candidates

Attribute

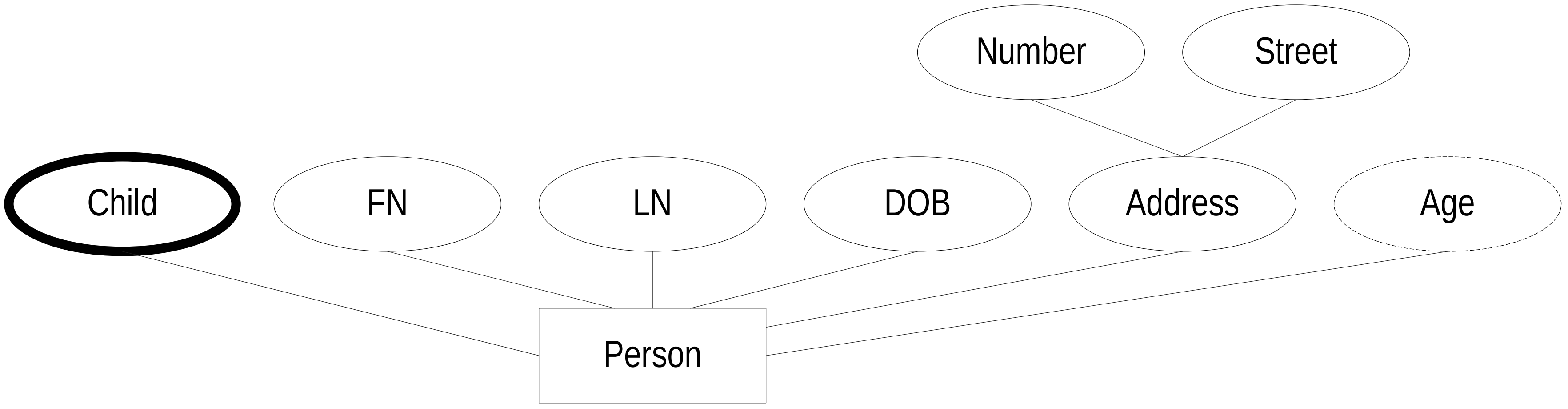
Attribute

- An entity has a set of zero or more ***attributes***, which are some properties
- Each attribute is drawn from some domain (such as integers) possibly augmented by **NULL** (more about NULLs later; for now, think as indicating "unknown")
- All the entities in an entity set have the same set of attributes, though generally not with the same values
- Attributes of an entity are written in ellipses (for now solid lines) connected to the entity
  - Example: FN: "First Name." LN: "Last Name." DOB: "Date of Birth."



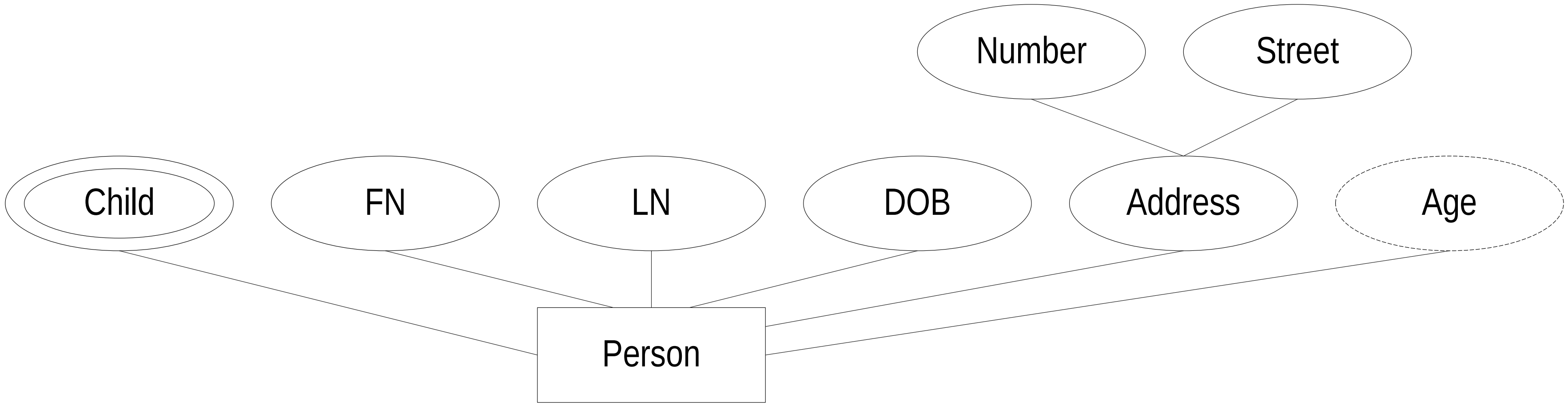
Attribute

- Attributes can be
  - ***Base*** (such as DOB) or ***derived*** (from other sources) denoted by dashed ellipses (such as Age, derived from DOB and the current date)
  - ***Simple*** (such as DOB) or ***composite*** (made of components) having their component attributes attached to them (such as Address, when we think of it explicitly as consisting of street and number and restricting ourselves to one city only)
  - ***Singlevalued*** (such as DOB) or ***multivalued*** (with zero or more values) denoted by thick-lined ellipses (such as Child; a person may have any number of children; we do not consider here children as persons; they are not elements of Person, just attributes of the entities in that set)



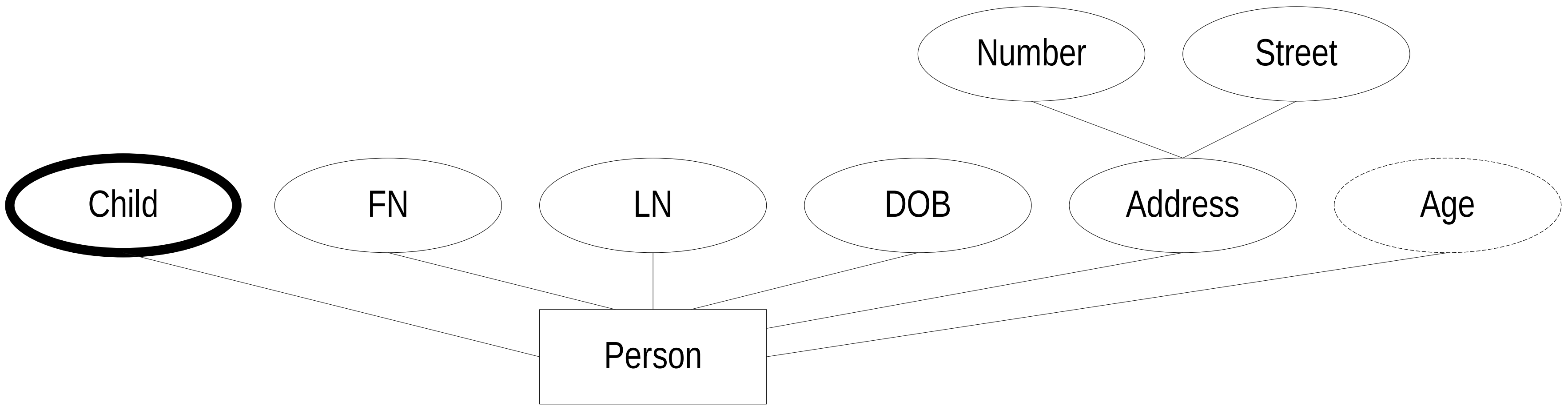
Thick Lines or Double Lines

- Some people (not we) use the convention of using double lines whenever we use thick lines
  - This is now obsolete and was used when the diagrams were drawn by hand
  - But many systems, e.g., draw.io (https://app.diagrams.net/), still use it
- We will use thick lines
- Using our example, the ER diagram using double lines would look like below



Attribute

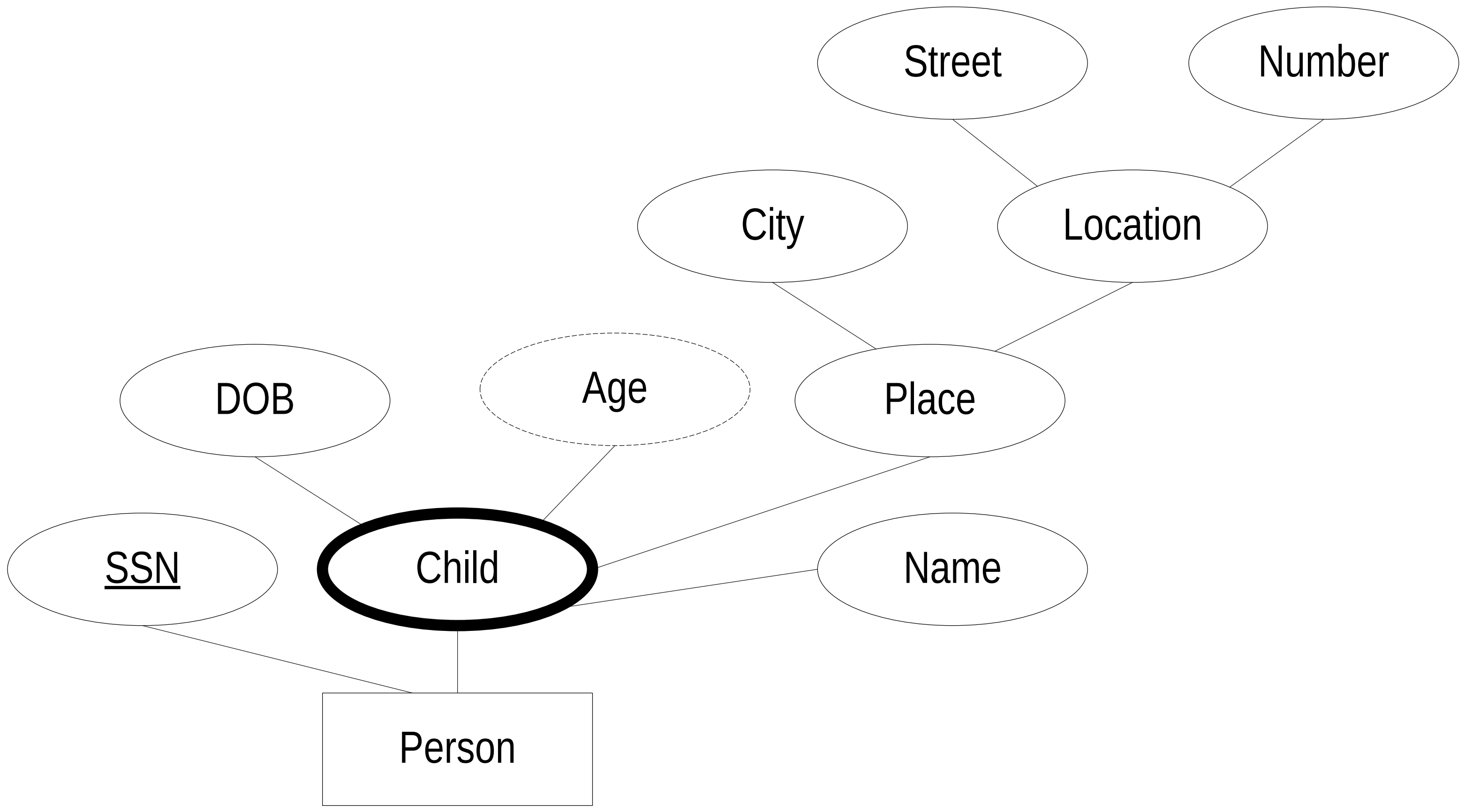
- To have a simple example of a person with attributes
  - Child: Bob
  - Child: Carol
  - FN: Alice; we use FN for first name or given name
  - LN: Xie; we use LN for last name or family name
  - DOB: 1980-01-01
  - Address.Number: 100
  - Address.Street: Mercer
  - Age: Current Date minus DOB specified in years (rounded down)



Attributes

- By obvious extensions, we can have more complex structures, such as in the example below

  We will discuss the meaning of underlines next



Key

How to Identify/Specify an Entity?

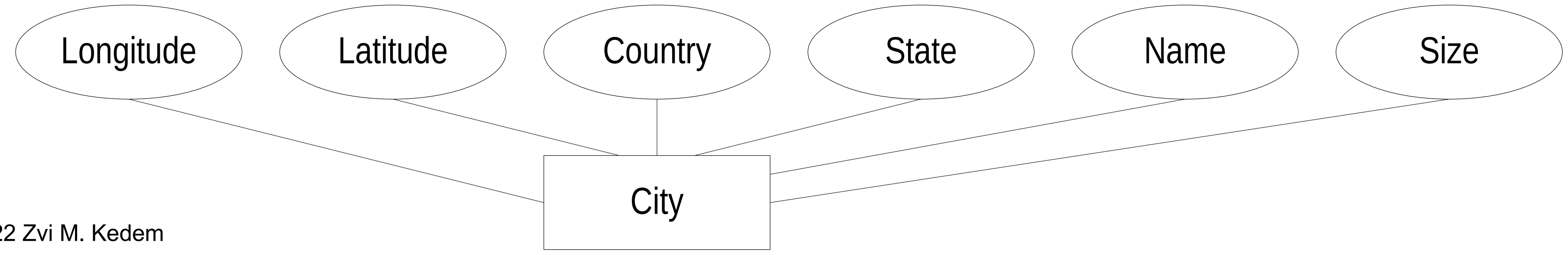
- We have an entity set and we want to identify/specify an individual entity in it
- We have 3 ways of doing it
  - Using its attributes
  - Using its relationship to an entity in a different entity set
  - Using its attributes and a relationship to an entity in a different entity set
- We start with "using its attributes" and talk about ***keys*** next
- We will talk about the other options when we study weak entities

Sets, Subsets, and Supersets

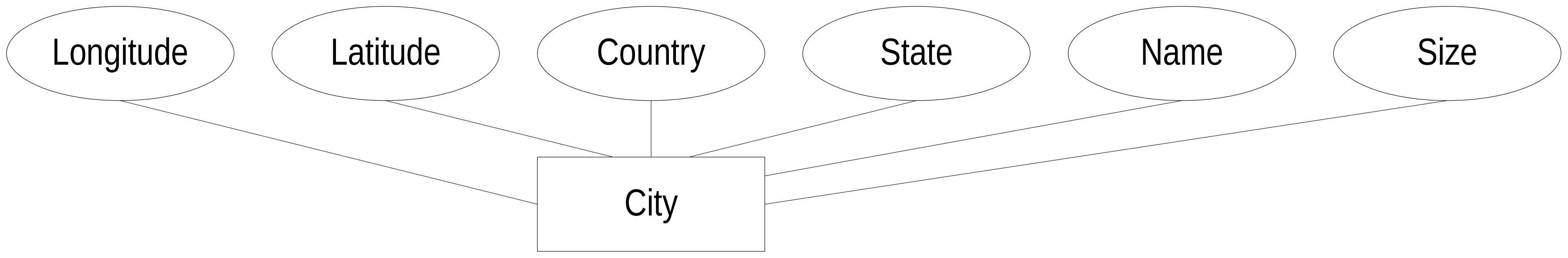
- Relations **subset** and **superset** are defined among sets $\subseteq$
- This is analogous to $\leq$
- Let us review by an example of three sets
  - A = {2,5,6}
  - B = {1,2,5,6,8}
  - C = {2,5,6}
- Then we have
  - A $\subseteq$ B and A is a subset of B
    and A is a **proper subset**, as it is not all of B; A $\subset$ B
  - A $\subseteq$ C and A is a subset of C
    and A is not a proper subset of C, as it is equal to C; A = C
- Caution: sometimes $\subset$ is used to denote what we denote by $\subseteq$

Keys

- Most of the times (not always), some subset *S* (proper or not) of the attributes of an entity has the property
  - ***The values of attributes in S are always known (for every entity)***
  - ***Any two possible entities that are equal on the attributes in S are equal (are really one entity); can also say that once the values of key attributes are fixed, all other values are fixed also***
  - ***Minimality: No proper subset of S has this property***
- Such an *S* is called a ***key*** (and sometimes: ***candidate*** key)
- Therefore, ***two different entities in an entity set must differ on at least one attribute in a key***
- Example: S consisting of Longitude and Latitude



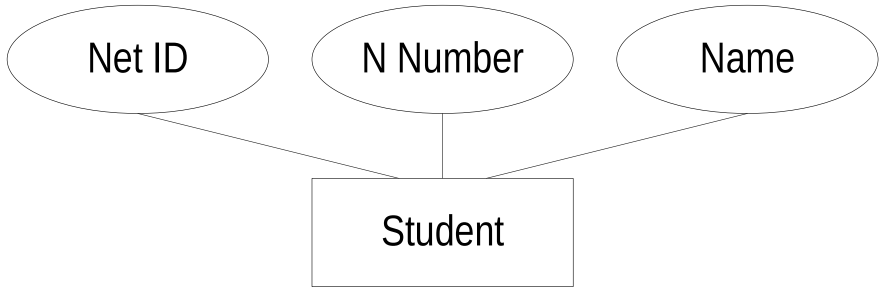
Keys



- Examples of non-keys
  - Longitude (not enough)
  - Longitude and Name (not enough)
  - Longitude, Latitude, and Name (enough, but too much)

The Need For Minimality

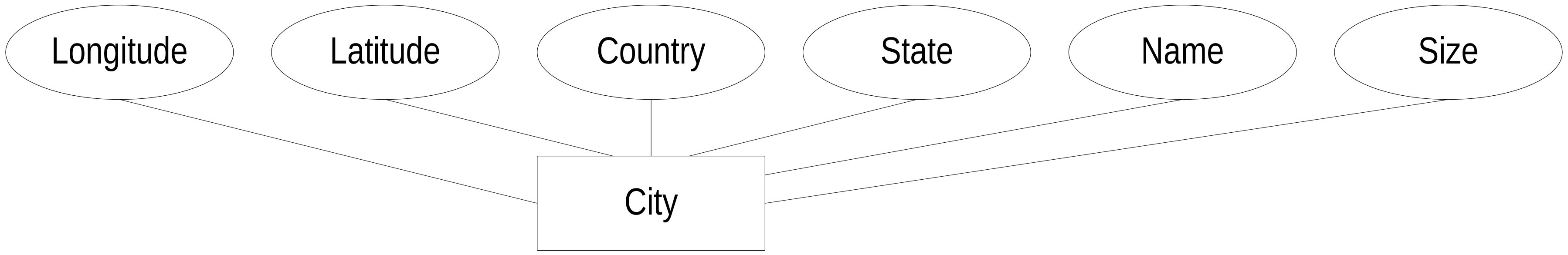
- A key must be ***minimal***, no proper subset could serve as a key



- Obvious semantics:
  - Net ID specifies a Student: Net ID could be a key
  - N Number specifies a Student: N Number could be a key
- ***Net ID and N Number together will not work as a key***
  - A mistake I have seen before, even from people who worked several years on DBMSs
- Counter-example: Assume that (Net Id, N Number) are declared as a key
- Then the two entities below have different keys, but cannot be both in the database as the N Numbers are the same but Net Ids are not
  - abc123, N12345678, Xie efg456, N12345678, Xie

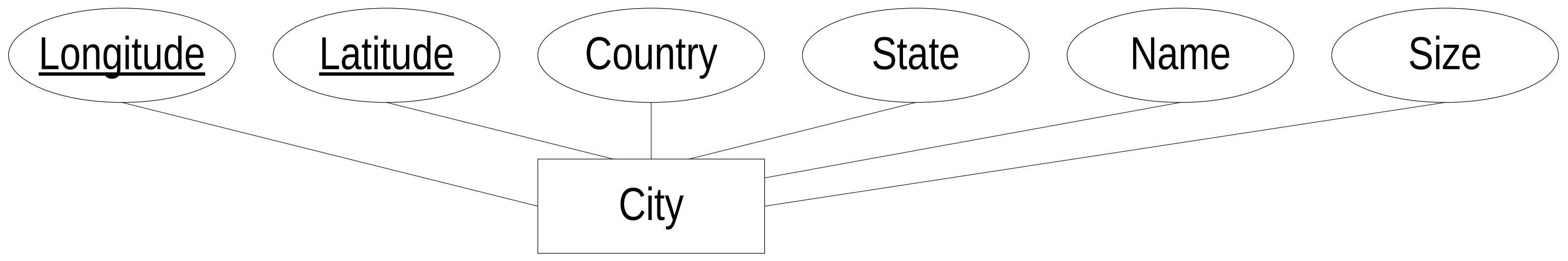
Primary Keys

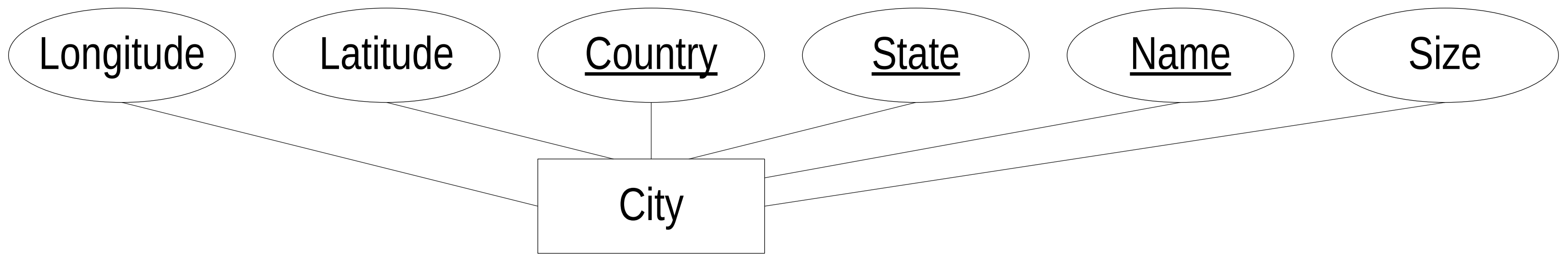
- If an entity set has one or more keys, one of them (no formal rule which one) is chosen as the ***primary key***
- In the ER diagram, the attributes of the primary key are underlined
- Let's assume that every country is partitioned into states and for every
  - Longitude and Latitude are always known
  - No two cities can have the same values of (Longitude, Latitude)
  - Country, State, and Name are always known
  - No two cities can have the same values of (Country, State, Name)



Primary Keys

- Under our assumptions, two primary key are possible, and we must choose one of them
  - But we will need to remember the key-type properties of the other possible primary key
- So, in our example two ER diagrams are possible
  - And we choose whichever we like, unless we have guidance from our customer

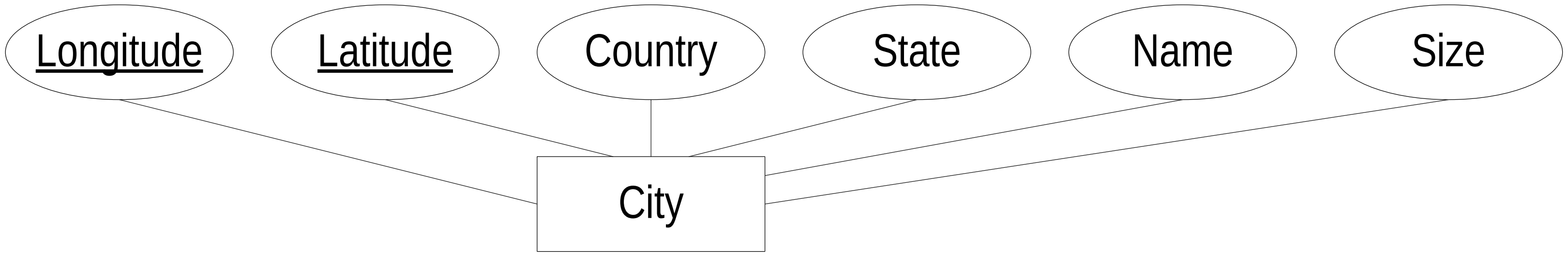




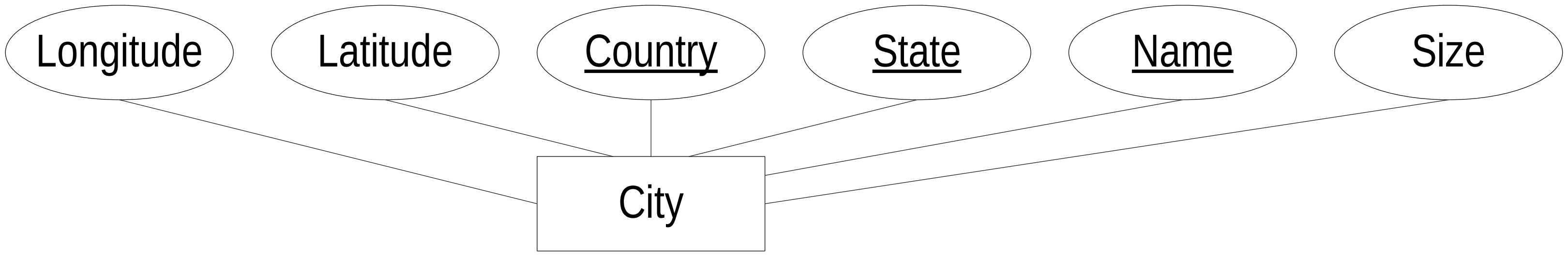
UNIQUE Constraint

- Let’s assume for now that some countries are partitioned into states, and some are not
- For countries with states
  - No two cities in the same state can have the same name
- For countries without states
  - No two cities (in that country) can have the same name
- So sometimes the value of State is not given
- (Country, State, City) cannot be the primary key
- But no two cities can have the same values of (Country, State, Name), even if State is not given
- This is specified by stating that (Country, State, City) is ***UNIQUE***
  - We will review UNIQUE when we study SQL DDL
- Of course, every key satisfies the UNIQUE constraint “automatically,” though do not specify UNIQUE for them

Primary Keys And UNIQUE



- (Country, State, Name) is UNIQUE



- (Latitude, Longitude) is UNIQUE

- You do not specify what's the PRIMARY KEY in text, the drawing tells you that
- Reminder: in UNIQUE, depending on the actual application not all the attributes have to be known, but whatever is known suffices to identify an individual entity

Relationship

Relationship

- Several entity sets (one or more) can participate in a ***relationship***
- Relationships are denoted by diamonds, to which the participating entities are “attached” at the vertices of the diamond
- A relationship could be unary, binary, ternary, ….
  - These refer to the number of “slots” in a relationship
  - < is binary as there are two “slots” for variables, e.g., x < y (the “slots” are x and y)
  - < 0 is unary as there is one “slot” for variables, e.g., x < 0 (the “slot” is x)
- By convention, a capitalized verb in third person singular (e.g., Likes), though we may not adhere to this convention if not adhering to it makes things clearer

Relationship

- We will have some examples of relationships
- We will use three entity sets, with entities (and their attributes) in those entity sets listed below

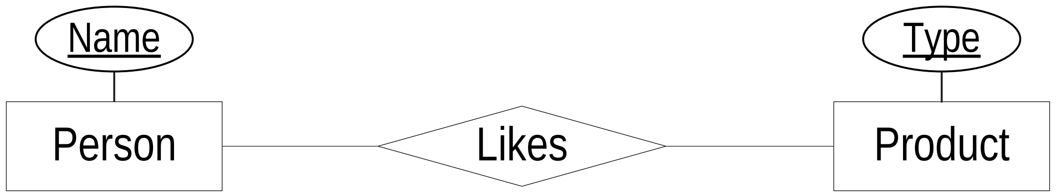
Person	Name
	Chee
	Lakshmi
	Marsha
	Michael
	Jinyang

Vendor	Company
	IBM
	Apple
	Dell
	HP

Product	Type
	computer
	monitor
	printer

Binary Relationship

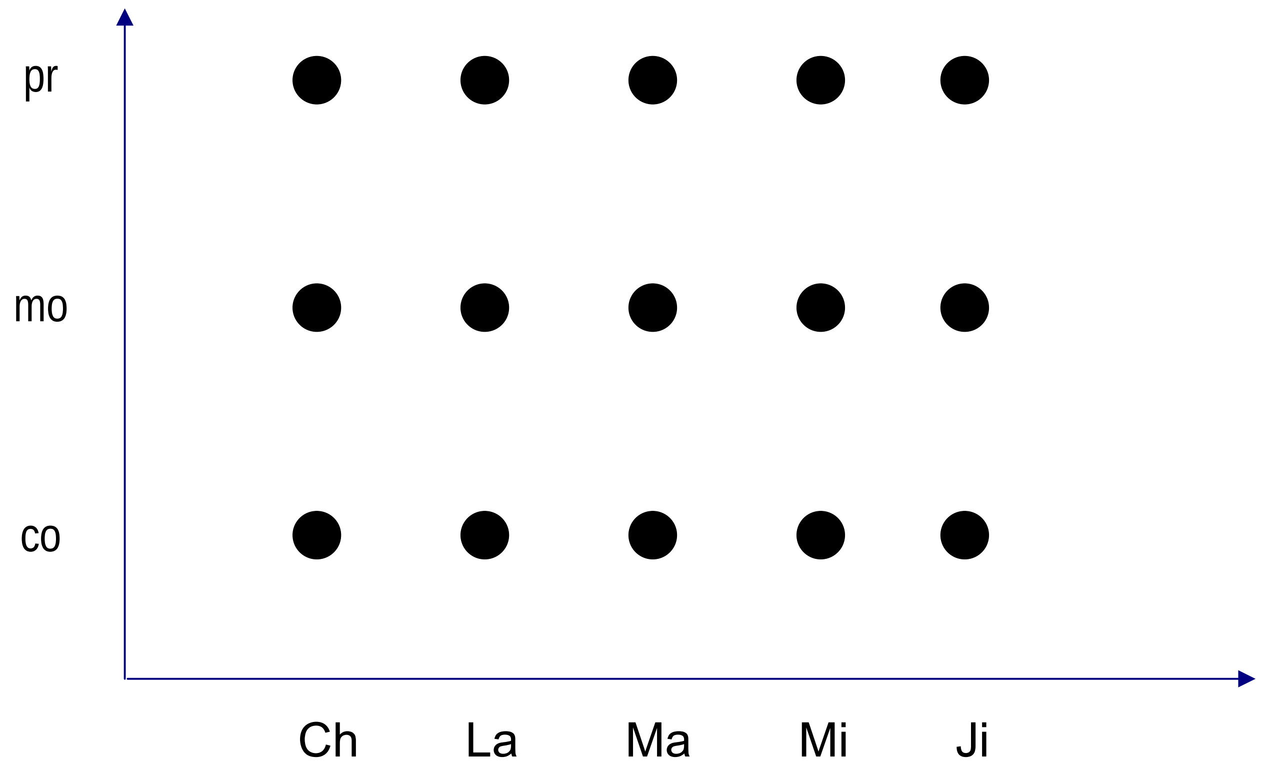
- Let’s look at Likes, listing all pairs of (*x*,*y*) where person *x* Likes product *y*, and the associated ER diagram
- First listing the relationship informally:
  - Chee likes computer
  - Chee likes monitor
  - Lakshmi likes computer
  - Marsha likes computer
- Note
  - Not every person has to Like a product
  - Not every product has to have a person who Likes it (informally, be Liked)
  - A person may Like many products
  - A product may have many persons each of whom Likes it
  - “Likes” is just a word in the diagram: maybe product likes person



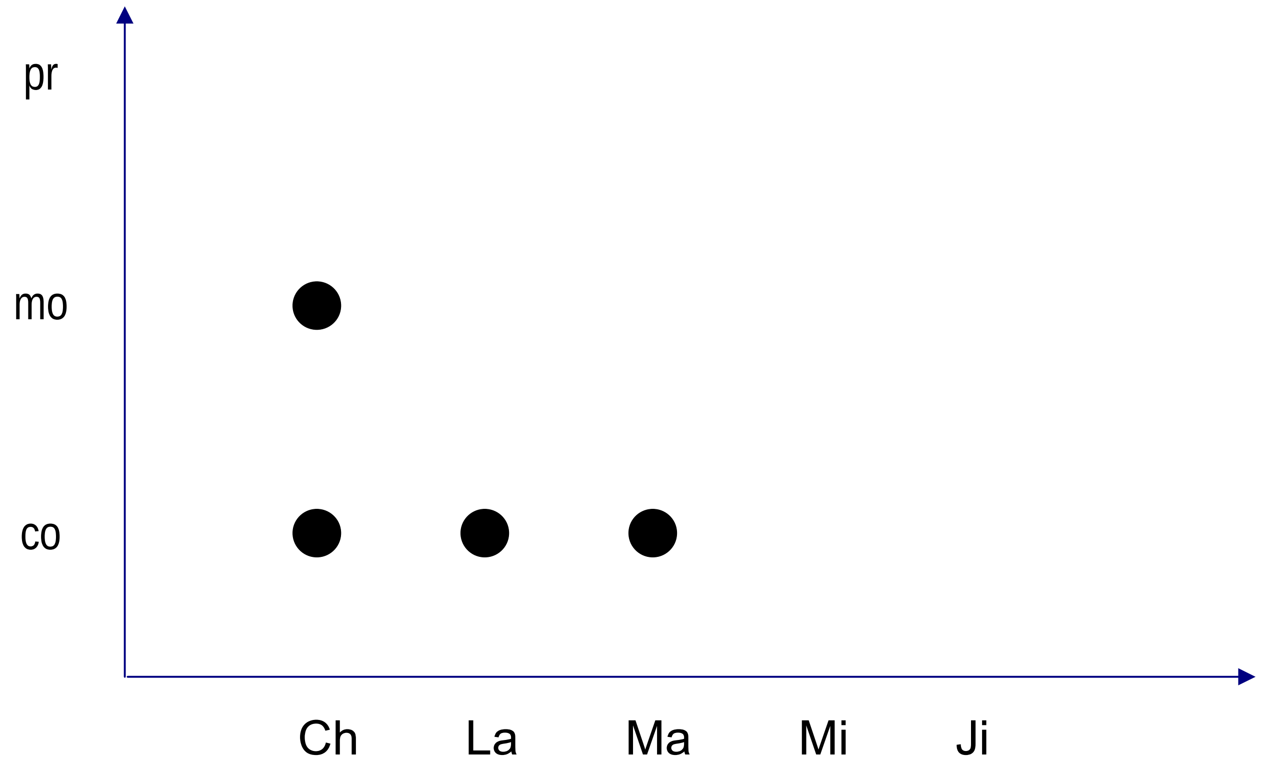
Relationships

- Formally we say that R is a ***relationship*** among (not necessarily distinct) entity sets E_1, E_2, …, E_n if and only if R is a subset of $E_1 \times E_2 \times \ldots \times E_n$ (Cartesian product)
- In our example above:
  - $n = 2$
  - E_1 = {Chee, Lakshmi, Marsha, Michael, Jinyang}
  - E_2 = {computer, monitor, printer}
  - $E_1 \times E_2$ = { (Chee,computer), (Chee,monitor), (Chee,printer), (Lakshmi,computer), (Lakshmi,monitor), (Lakshmi,printer), (Marsha,computer), (Marsha,monitor), (Marsha,printer), (Michael,computer), (Michael,monitor), (Michael,printer), (Jinyang,computer), (Jinyang,monitor), (Jinyang,printer) }
  - R = { (Chee,computer), (Chee,monitor), (Lakshmi,computer), (Marsha,monitor) }
- R is a set (unordered, as every set) of ordered tuples, or sequences (here of length two, that is pairs)

Our Cartesian Product



Our Instance Of The Relationship



Relationships

- Let us elaborate
- $E_1 \times E_2$ was the "universe"
  - It listed all possible pairs of a person liking a product
- At every instance of time, in general only some of this pairs corresponded to the "actual state of the universe"; *R* was the set of such pairs

Important Digression

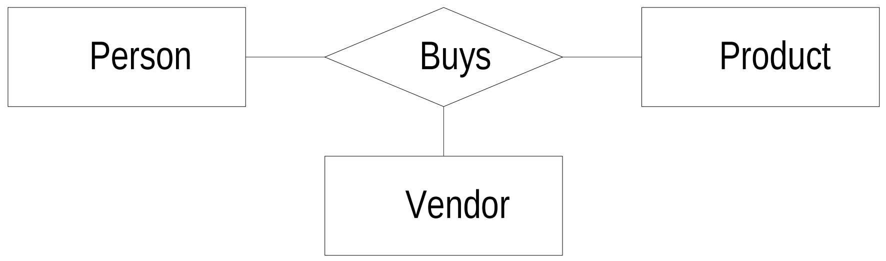
- Ultimately, we will store (most) relationships as tables
- So, our example for Likes could be

Likes	Name	Type
	Chee	Computer
	Chee	Monitor
	Lakshmi	Computer
	Marsha	Computer

- And we identify the “participating” entities using their primary keys

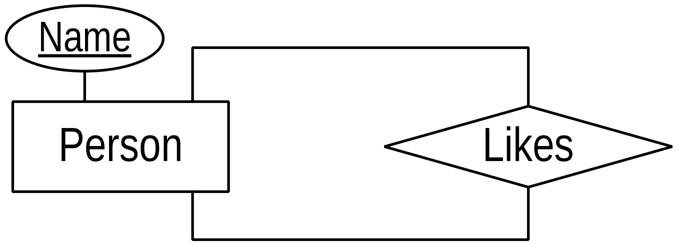
Ternary Relationship

- Let's look at Buys listing all tuples of (*x*,*y*,*z*) where person *x* buys product *y* from vendor *z*
- Let us just state it informally:
  - Chee buys computer from IBM
  - Chee buys computer from Dell
  - Lakshmi buys computer from Dell
  - Lakshmi buys monitor from Apple
  - Chee buys monitor from IBM
  - Marsha buys computer from IBM
  - Marsha buys monitor from Dell



Relationship With Nondistinct Entity Sets

- Let's look at the ***binary*** relationship Likes, listing all pairs of (x,y) where person x Likes person y
- Relationships in which a set participates more than once are sometimes called ***recursive***; this term is not important
- Let us just state it informally
  - Chee likes Lakshmi
  - Chee likes Marsha
  - Lakshmi likes Marsha
  - Lakshmi likes Michael
  - Lakshmi likes Lakshmi
  - Marsha likes Lakshmi
- Note that pairs must be ordered to properly specify the relationship, Chee likes Lakshmi, but Lakshmi does not like Chee



Relationship With Nondistinct Entity Sets

- Again:
  - Chee likes Lakshmi
  - Chee likes Marsha
  - Lakshmi likes Marsha
  - Lakshmi likes Michael
  - Lakshmi likes Lakshmi
  - Marsha likes Lakshmi

- Formally Likes is a subset of the Cartesian product Person × Person, which is the set of all ordered pairs of the form (person,person)
- Likes is the set { (Chee,Lakshmi), (Chee,Marsha), (Lakshmi,Marsha), (Lakshmi,Michael), (Lakshmi,Lakshmi), (Marsha,Lakshmi) }
- Likes can be though of as a directed graph in which persons serve as vertices and arcs specify who likes whom

Important Digression

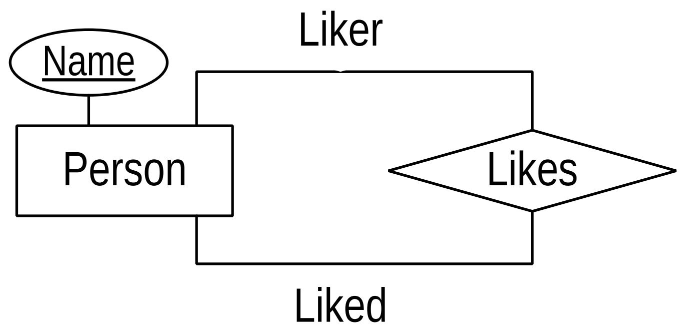
- Ultimately, we will store (most) relationships as tables
- So, our example for Likes could be

Likes	Name	Name
	Chee	Lakshmi
	Chee	Marsha
	Lakshmi	Marsha
	Lakshmi	Michael
	Lakshmi	Lakshmi
	Marsha	Lakshmi

- We identify the "participating" entities using their primary keys
- But it is difficult to see (unless we keep track of columns order) whether Lakshmi Likes Michael or Michael Likes Lakshmi

Relationship With Nondistinct Entity Sets

- Frequently it is useful to give ***roles*** to the participating entities, when, as here, they are drawn from the same entity set.
- So, we may say that if Chee likes Lakshmi, then Chee is the “Liker” and Lakshmi is the “Liked”
- Roles are explicitly listed in the diagram, but the semantics of what they mean cannot be deduced from looking at the diagram only



Important Digression

- Ultimately, we will store (most) relationships as tables
- So, our example for Likes could be

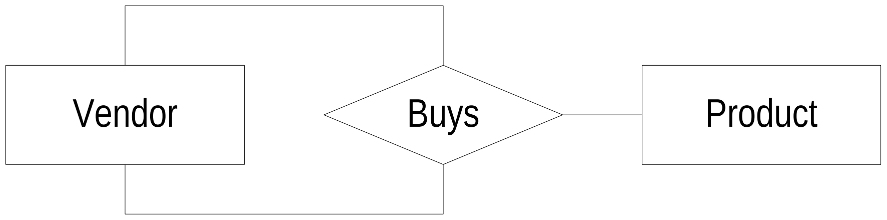
Likes	Liker	Liked
	Chee	Lakshmi
	Chee	Marsha
	Lakshmi	Marsha
	Lakshmi	Michael
	Lakshmi	Lakshmi
	Marsha	Lakshmi

- We identify the "participating" entities using their primary keys, but we rename them using ***roles***
- So, we do not need to keep track of columns order and we know that Lakshmi Likes Michael and Michael does not Like(s) Lakshmi, though we still do not know what "Likes" really means

Relationship With Nondistinct Entity Sets

- Consider Buys, listing all triples of the form (*x*,*y*,*z*) where vendor *x* buys product *y* from vendor *z*

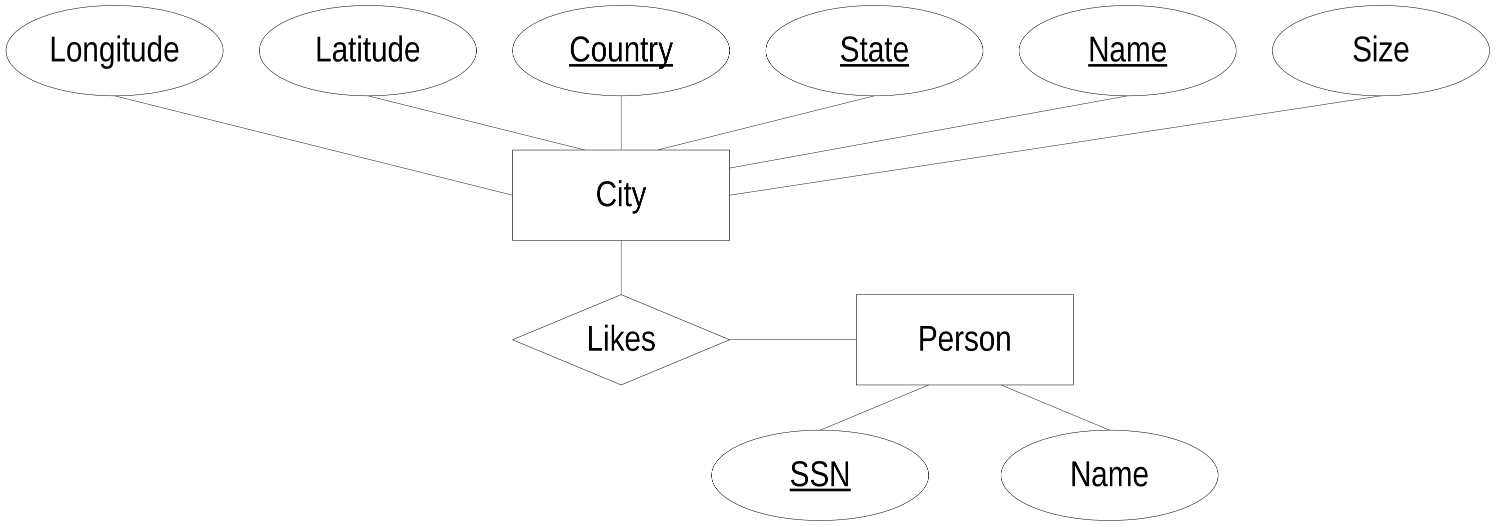
- A typical tuple might be (Dell,printer,HP), meaning that Dell buys a printer from HP



ER Diagrams

ER Diagrams

- To show which entities participate in which relationships, and which attributes participate in which entities, we draw line segments between:
  - Entities and relationships they participate in
  - Attributes and entities they belong to
- We also underline the attributes of the primary key for each entity that has a primary key
- Below is a simple ER diagram (with a simpler Person than we had before):



Further Refinements to the ER Model

- We will present, in steps, further refinements to the model and associated diagrams
- The previous modeling concepts and the ones that follow are needed for producing a data base design that models a given application well
- We will then put that together in a larger comprehensive example

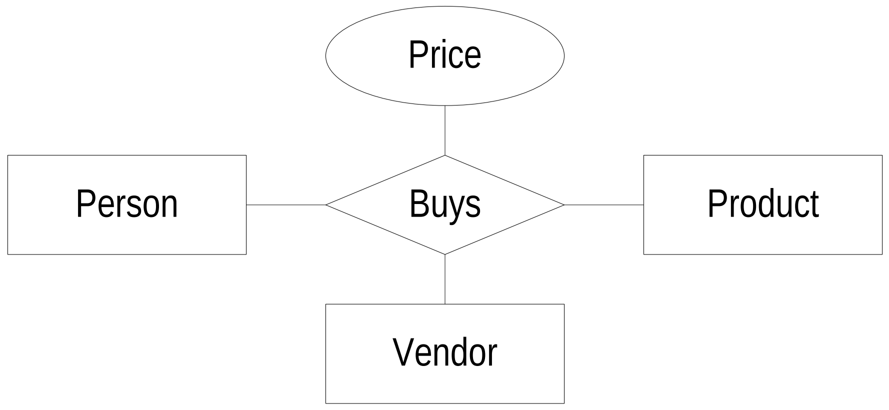
Relationship With Attributes

- Consider relationship Buys among Person, Vendor, and Product
- We want to specify that a person Buys a product from a vendor at a specific price
- Price
  - is not a property of a vendor, because different products may be sold by the same vendor at different prices
  - is not a property of a product, because different vendors may sell the same product at different prices
  - is not a property of a person, because different products may be bought by the same person at different prices

Relationship With Attributes

- So, Price is really an attribute of the relationship Buys
- For each tuple (person, product, vendor) there is a value of price

  More formally, there is at most one value of price



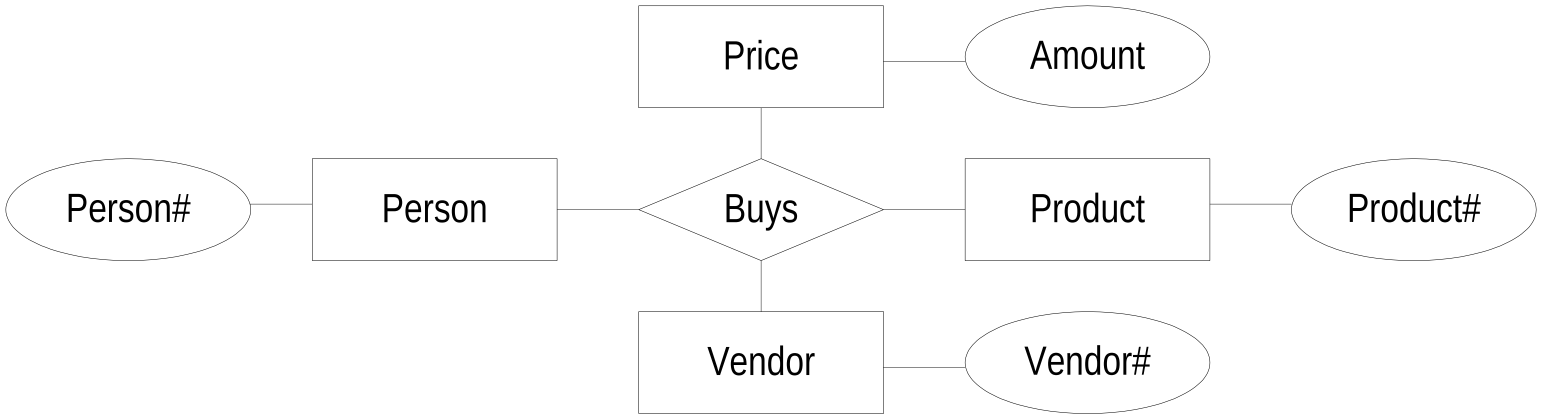
Entity Versus Attribute

- Entities can model situations that attributes cannot model naturally
- Entities can
  - Participate in relationships
  - Have attributes
- Attributes cannot do any of these (though you could think that in a composite attribute an attribute may have attributes)

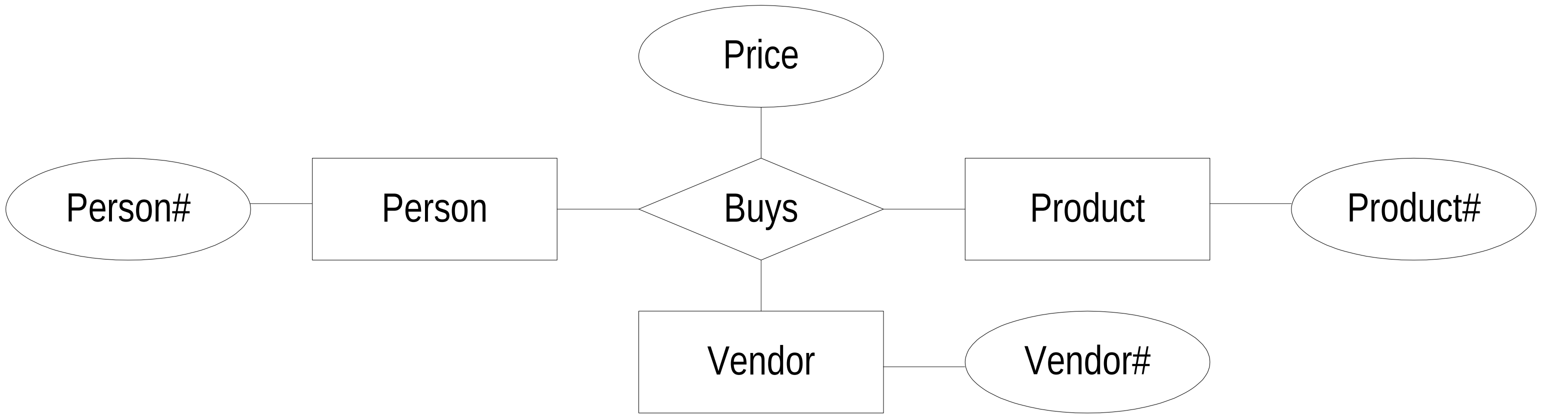
- Let us look at a “fleshed-out example” for possible alternative modeling of Buys

Other Choices for Modeling Buys

- Price is just the actual amount, the number in $’s
- So there likely is no reason to make it an entity as we have below

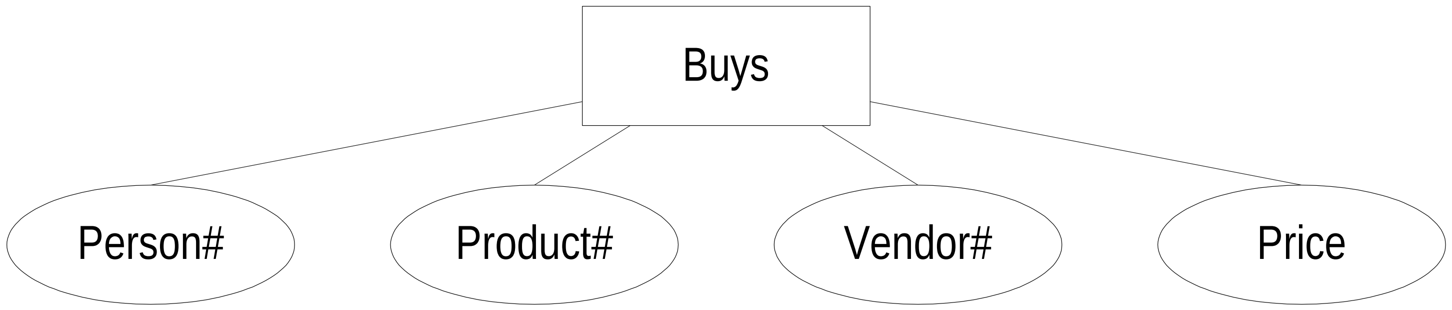


- We should probably have (as we had earlier less fleshed out)



Other Choices for Modeling Buys

- Or should we just have this?

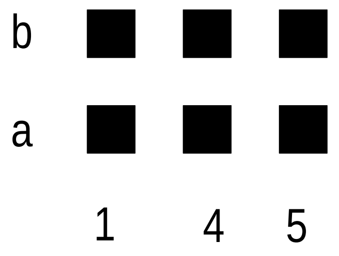


- Not if we want to model something about a person, such as whom a person likes
  - This requires a person to be able to enter into a relationship (with other persons)
  - And we cannot model this situation if person is an attribute of Buy
- Similarly, for product and vendor

Functionality (As in Mathematical Functions)

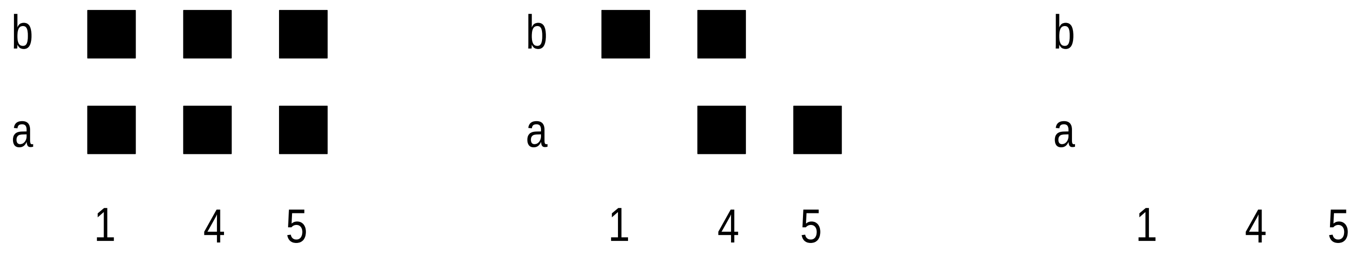
Relationships/Relations And Functions (Some Review Of Previous Material)

- First, we discuss the concepts generally, then in the context of databases
- We will use the term “relation” right now for relationships, to follow standard mathematical terminology
- We will restrict ourselves, for simplicity, ***to binary relations***, which are the most common relations
- We have sets and (and could be the same set)
- We have a cartesian product which is the set of all pairs
- A (binary) relation is a subset of such a cartesian product
- If and then can be drawn, with black squares standing for the pairs, as

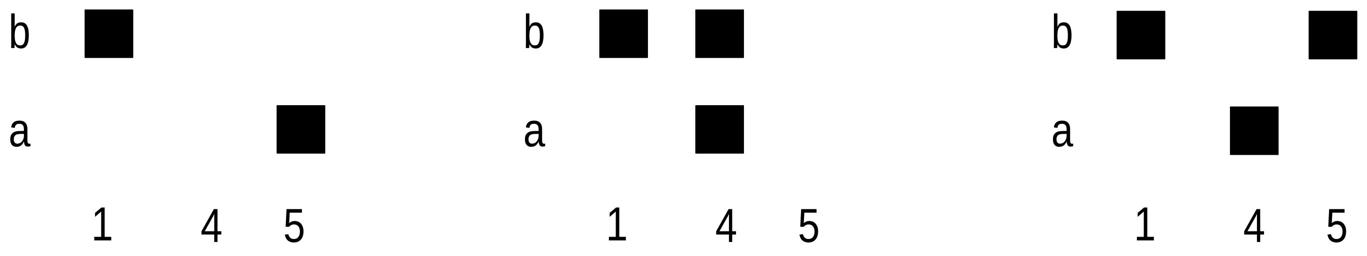


Relationships/Relations And Functions

- A ***relation*** is just any ***subset*** of the given cartesian product
- Below are 3 relations (the last one is empty)



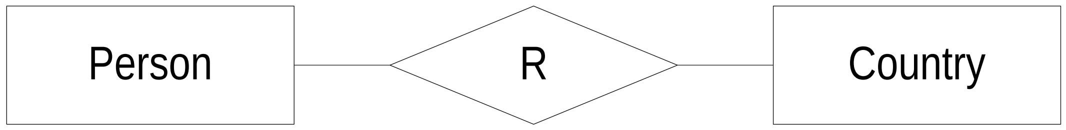
- A relation is a (***partial***) ***function*** iff for every there is at most one such that the pair is in the relation
- Below we have, respectively, a ***partial function***, ***not a function***, and a ***total*** function (for every there is exactly one such that the pair is in the relation)



Binary Relationships and Their Functionality

- "***Functionality***" means here that sometimes a relationship is a ***partial function***, that is
  - A function
  - But maybe not necessarily defined on the whole domain

    For example, if varies over all real numbers, the function is not defined for ; it is a partial function
  - ***A partial function that is defined on the whole domain is total***
- Consider a relationship R between two entity sets A, B.
- We will look at examples where A is the set of persons and B is the set of all countries



- We will be making some simple assumptions about persons and countries, which we list when relevant

Binary Relationships As Directed Graphs

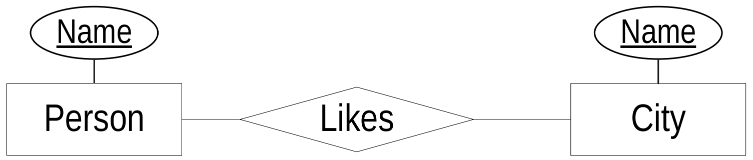
- Binary relationships are by far the most important relationships
- In fact, jumping ahead, ***all*** relationships in relational databases are binary (and many-to-one, which we will understand in the next unit)
  - Formally, all relationships in a relations databases are partial functions

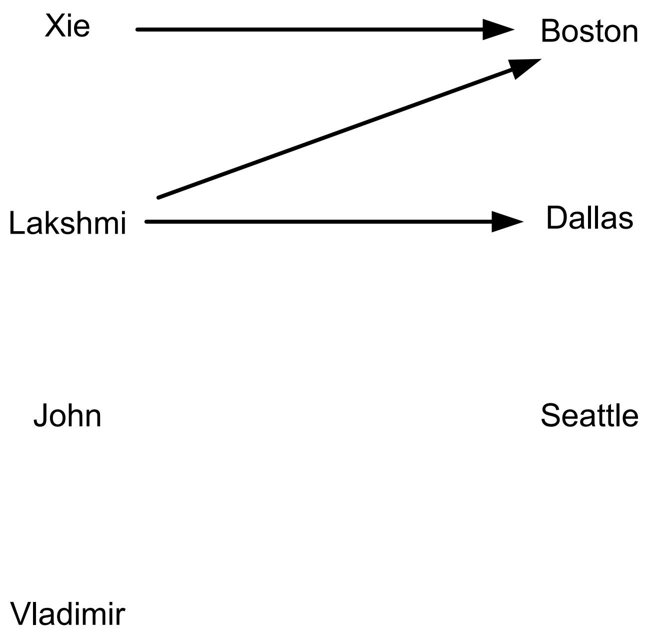
- Binary relationships can be naturally represented by directed graphs

- There will be 2 important cases to consider for a binary relationship between R and S
  - R and S are distinct sets
  - R and S are the same set

R and S Are Distinct Sets

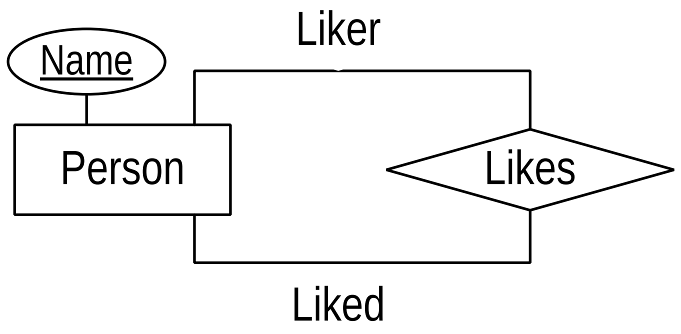
- For the example, let R be Persons and S be Cities
- The graph is ***bipartite:*** all the arcs (directed edges) are from the vertices in R to vertices in S
- The relationship is Likes

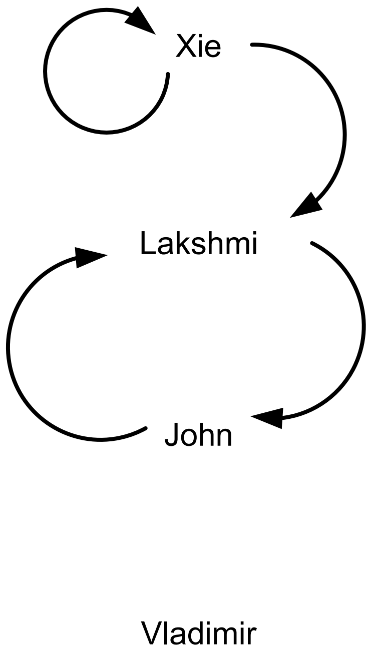




R and S Are The Same Set

- For the example, let’s have R to be Persons
- For example, the relationship is Likes



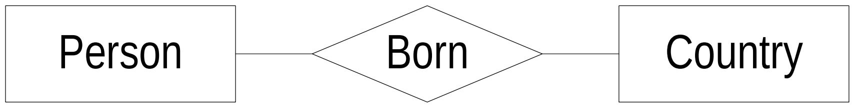


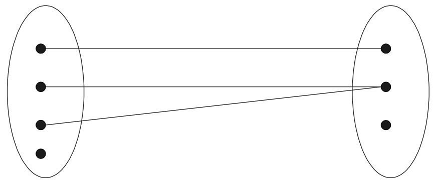
Binary Relationships and Their Functionality

- Relationship R is ***many to one*** from A to B if and only if for each element of A there exists ***at most one*** element of B related to it
  - Example: R is Born (in)

    Each person was born in at most one country (maybe not in a country but on a ship in the middle of an ocean or maybe we do not know the country for the person), so there is a ***partial function*** from set Person to set Country

    Maybe nobody was born in some country as it has just been established





- The picture on the right is a bipartite graph describing the universe of 4 persons and 3 countries, with lines indicating which person was born in which country
  - We will have similar diagrams for other examples

Many-To-One Relationships

- Formal definition follows

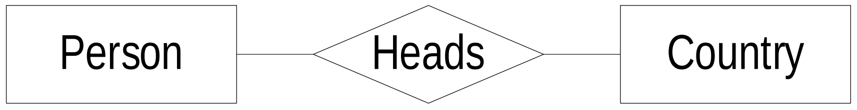
  if) and) are in the relationship then

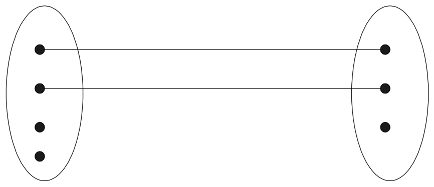
Binary Relationships and Their Functionality

- Relationship R is called ***one to one*** between A and B if and only if for each element of A there exists ***at most one*** element of B to which it is related and for each element of B there exists ***at most one*** element of A which is related to it
  - Example: R is Heads

    Each Person is a Head (President, Queen, etc.) of at most one country (not true in reality)

    Each country has at most one head (maybe the queen died, and no monarch for that country exists now)





One-To-One Relationships

- Formal definition follows

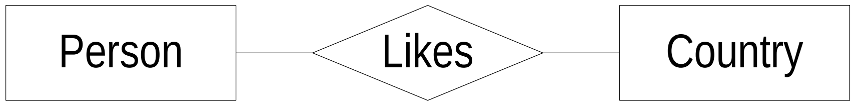
if) and) are in the relationship then
and
if) and) are in the relationship then

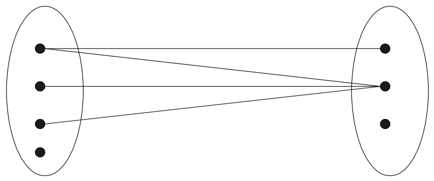
Many-To-Many Relationships

- There are no restriction beyond its being a relationship

Binary Relationships and Their Functionality

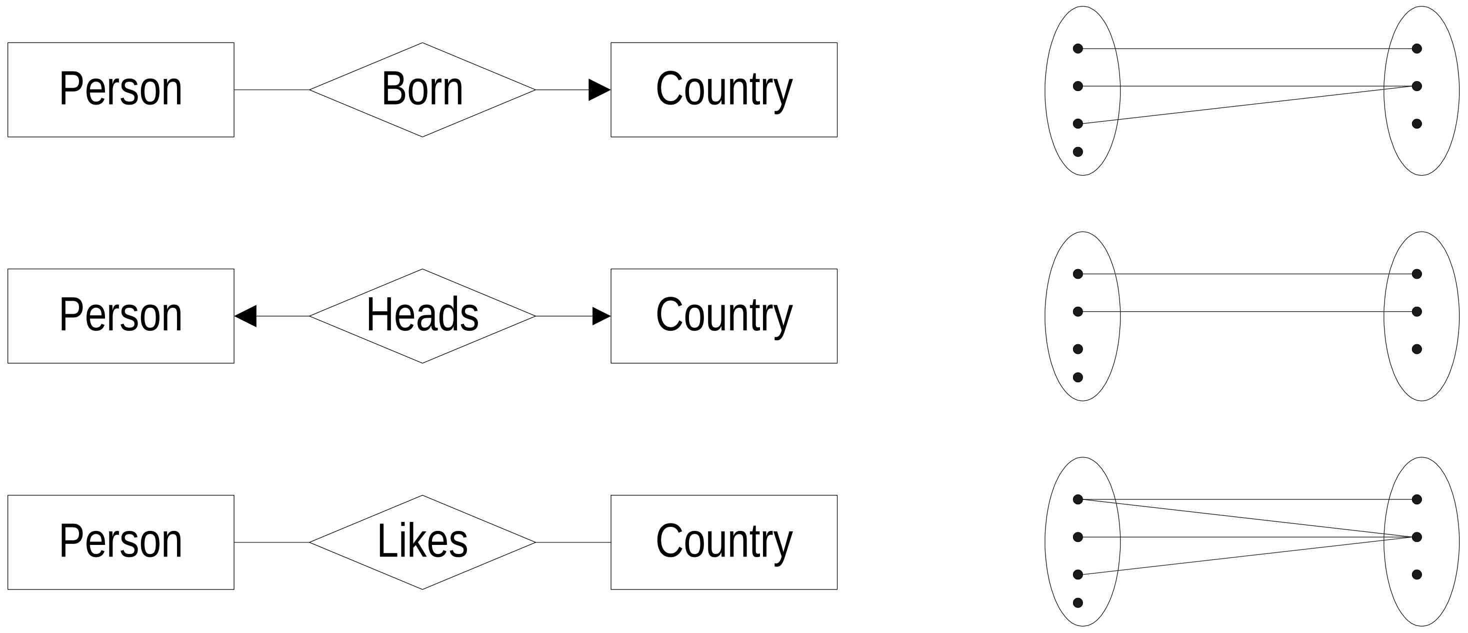
- A relationship is called ***many to many*** between A and B, if there are no restrictions.
  - Example: R is “Likes”





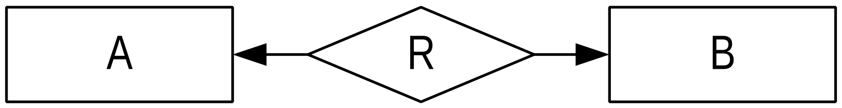
Binary Relationships and Their Functionality

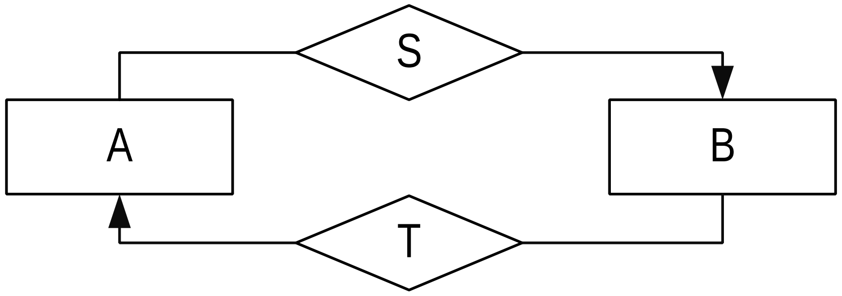
- We have in effect considered the concepts of partial functions of one variable.
  - The first two examples were ***partial functions*** (to remind, “partial” means that the function does not have to be defined on all the elements of its domain)
  - The last example was not a function
- Pictorially, functionality for binary relationships can be shown by drawing an arc head in the direction to the “one”



Classroom Exercise

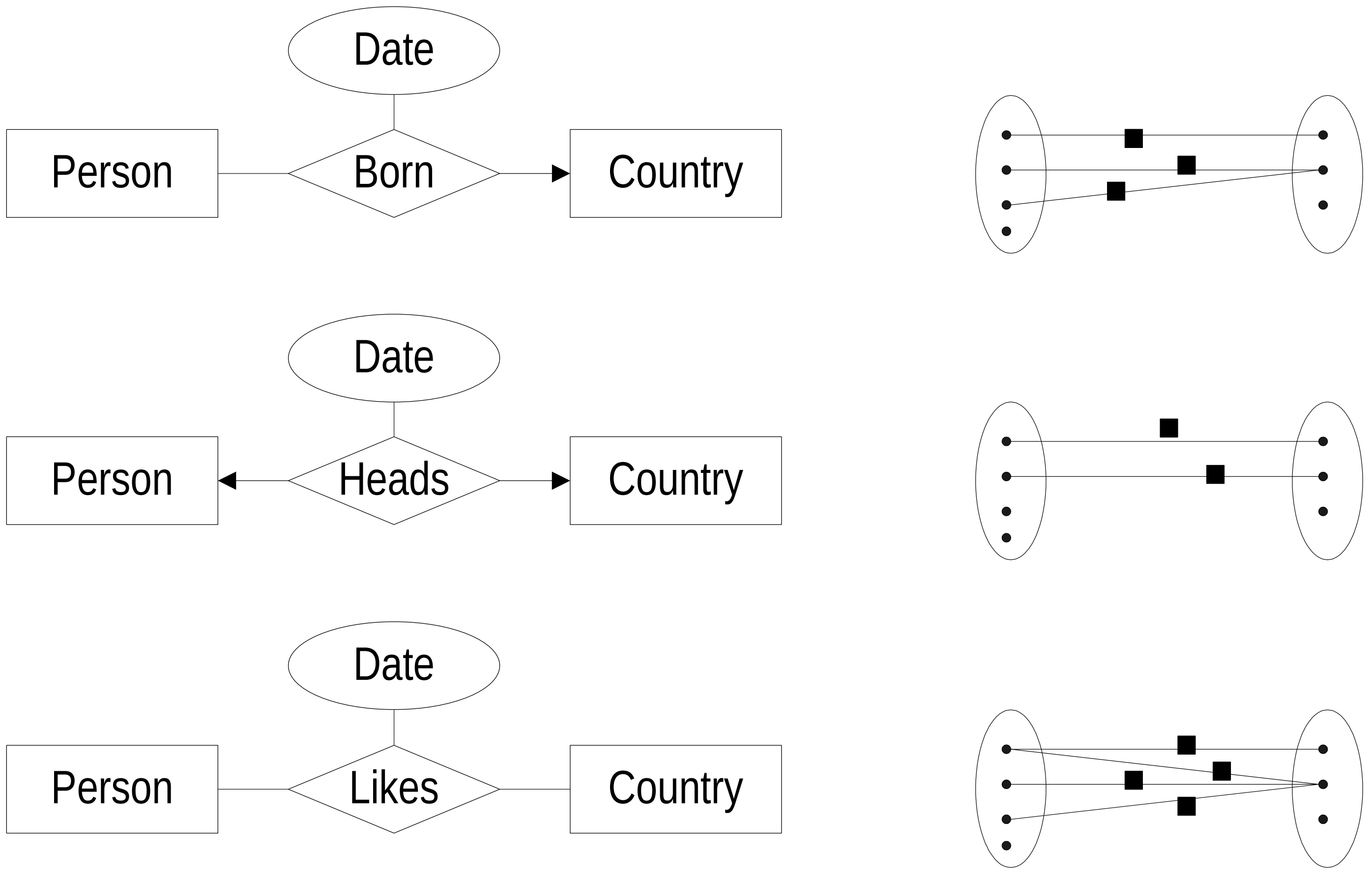
- What is the difference between these two diagrams, if any, ignoring the names of the relationships?





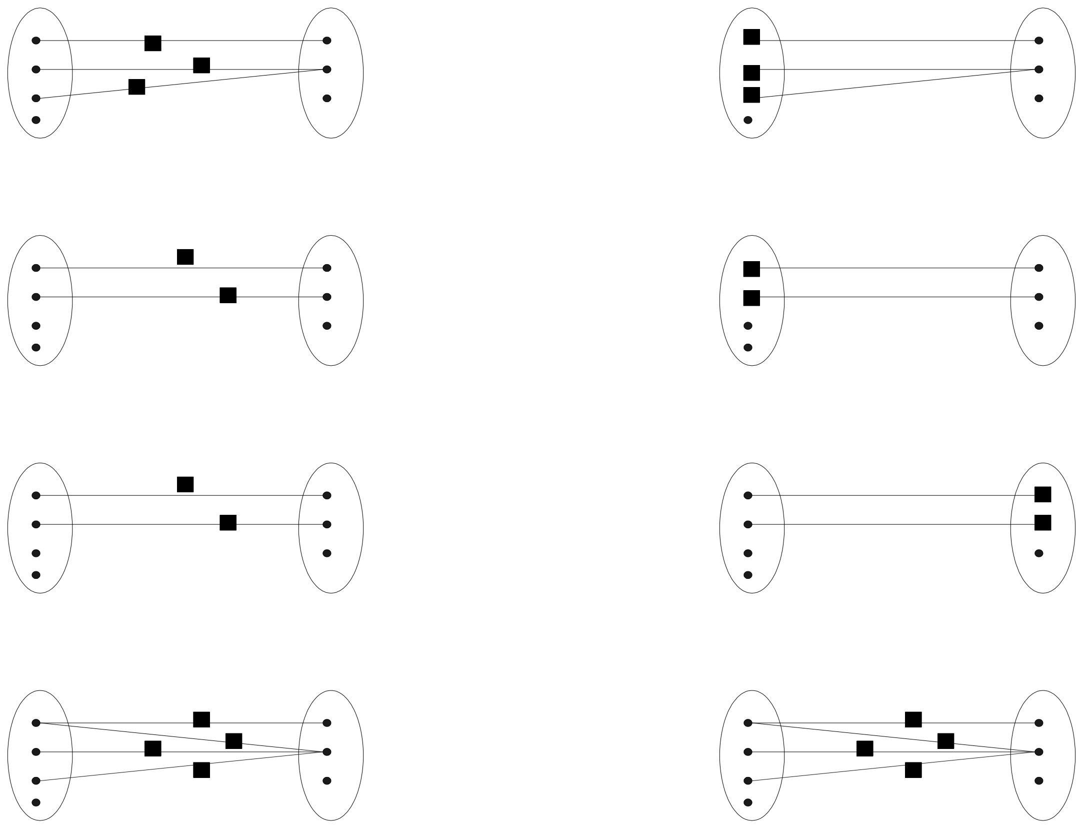
Binary Relationships and Their Functionality

- How about the properties of a relationship? Let's look at an example
- Date: when a person and a country are in a relationship this attribute is the first day when they entered into the relationship (Date marked with a black square)



Binary Relationships and Their Functionality

- Can make Date in some cases a property of an entity
  - In Born “slide” the Date to the Person, but not the Country
  - In Heads “slide” the Date to either the Person or the Country (but not for both, as this would be redundant)
- Cannot “slide” the Date to either “Liker” or “Liked”
- Can “slide” if no two squares end up in the same entity



Binary Relationships and Their Functionality

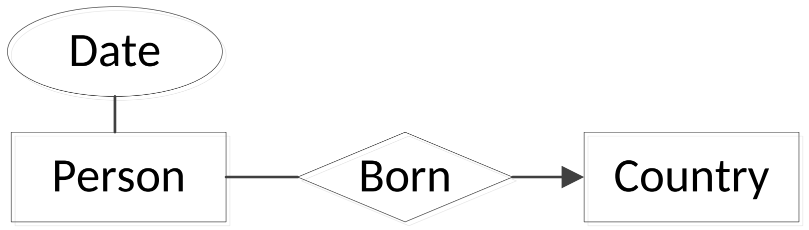
- If the relationship is many-to-one, then a property of the relationship can be attributed to the "many" side

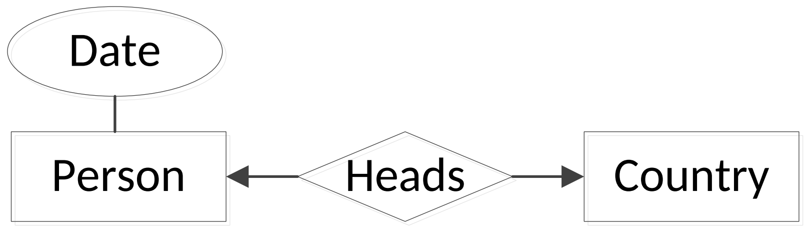
- More generally, we could say that if the relationship is many-to-one then the property of the relationship can be attributed to the side "opposite" to the "one" side

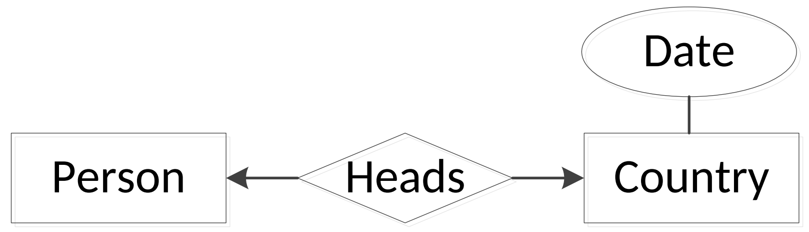
- The following can be done if the relationship is one-to-one
- A property of the relationship can be "attributed" to any of the two sides
  - Each of these two sides is "opposite" to a "one" side

Alternate Designs

- Entities “inheriting” attributes of relationships when the relationships are not many to many

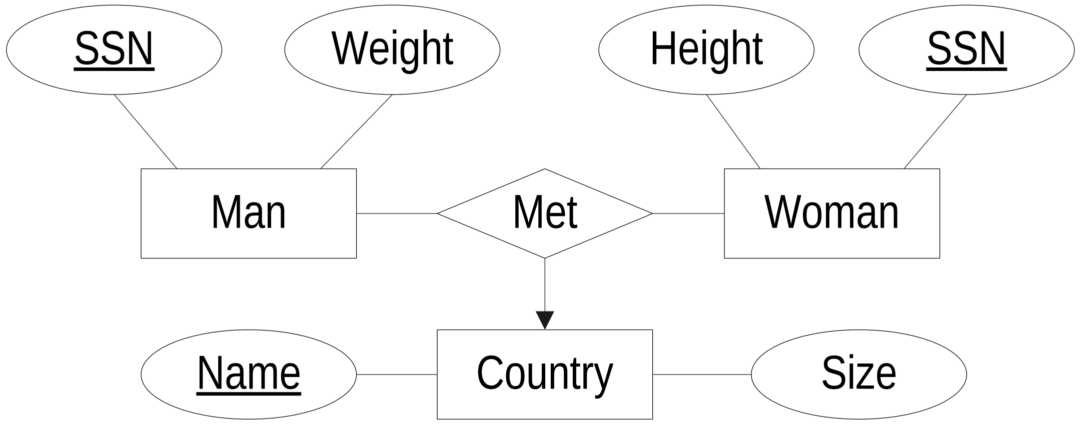






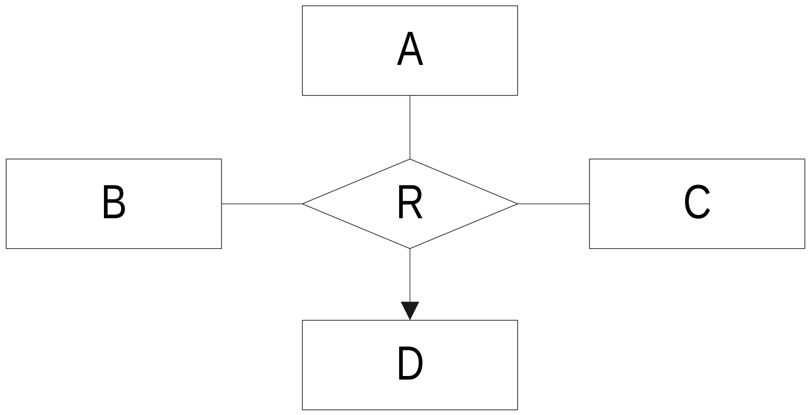
Non-Binary Many-To-One Relationships

- There could also be relationships that are not binary but still many-to-one
- In the example below, a man and a woman could have met in at most one country
- This is indicated by an arrow



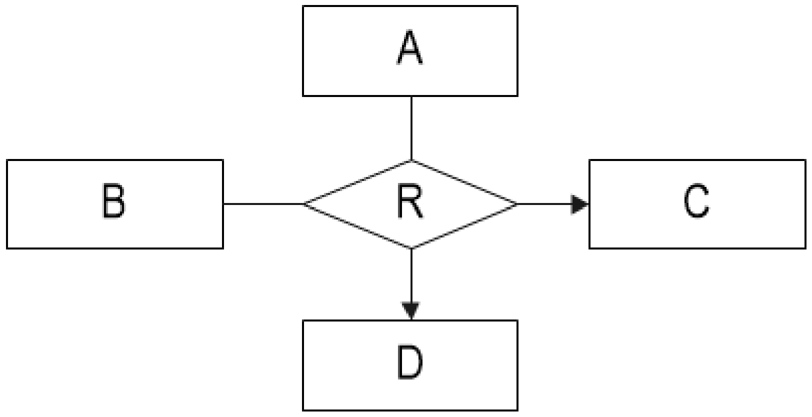
Non-Binary Many-To-One Relationships

- Below an example of modeling when for every triple of entities (a,b,c) from A, B, and C, there is at most one entity d in D



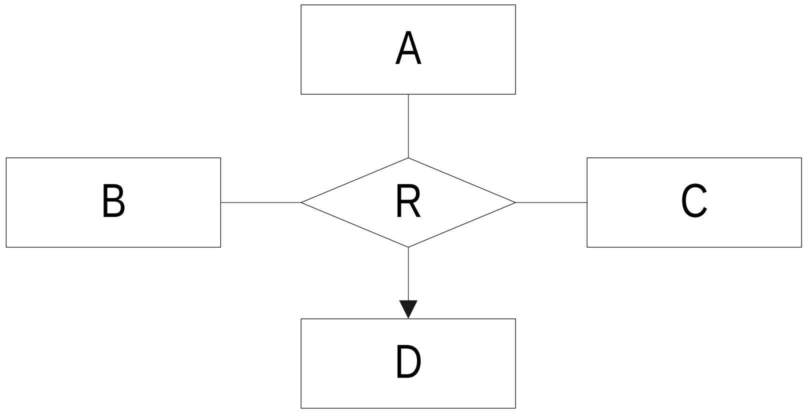
Non-Binary Many-To-One Relationships

- However, cannot model using ER diagrams more complex cases such as the case where for every pair of entities (a,b) in A and B there is at most one entity in C and at most one entity in D.
  - Also, that's less important as we will see later
- It is important to know about the two partial functions above, but the diagram below is ***not*** permitted in standard ER diagrams (though it could have been)
- It would be confusing as without annotations it is not clear from it that, for example it is not the case that
  - For every (a,c) whatever b is there is at most one d
  - For every (b,d) whatever a is there is at most one c



Non-Binary Many-To-One Relationships

- Assume now that for every pair of entities (a,b) from A and B, there is at most one entity d in D.
- Then we can draw



because, of course, it is also the case that for every triple of entities (a,b,c) from A, B, and C, there is at most one entity d in D

- But ***we must add a text annotation*** stating that the relationship is just from A and B

Arrows From Relationships To Entities

- Following on the previous discussions we allow at most one arrow “coming out” of any relationship
- Following on the example of the Heads relationship earlier

  There is one exception: A binary one-to-one relationship has 2 arrows coming out of it
- We will discuss this case in greater detail in the unit dealing with relational implementation

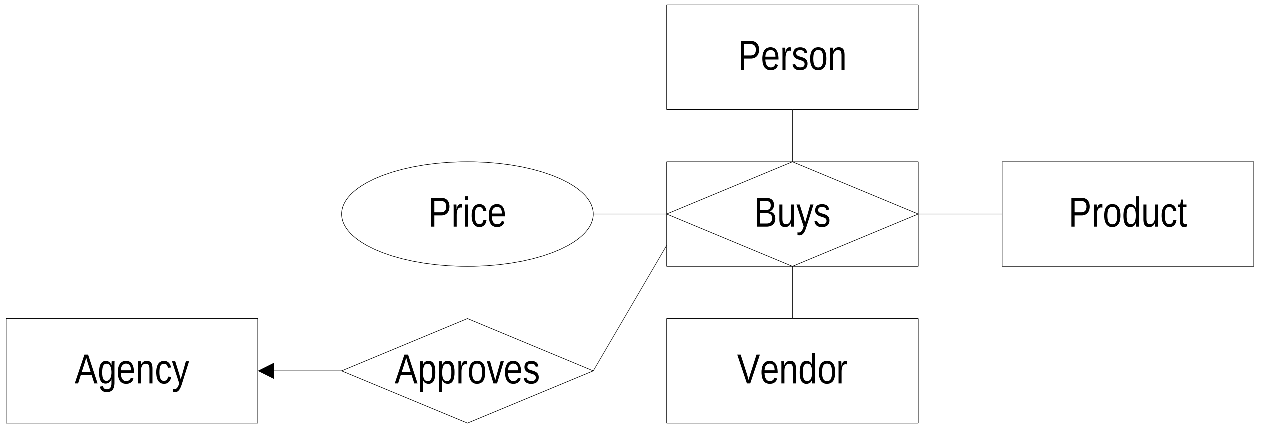
Aggregation

Aggregation: Relationships as Entities

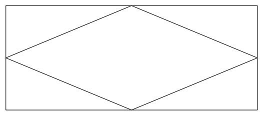
- It is sometimes natural to consider relationships as if they were entities.
- This will allow us to let relationships to be treated as entities, e.g., participate in other “higher order” relationships
- Here each instance of Buys needs to be approved by (at most) one agency
- Relationship is “made into” an entity by putting it into a ***rectangle***; note ***that the edge between Buys and Approves touches the Buys rectangle but not the Buys diamond***, to make sure that we won’t get confused

Aggregation: Relationships as Entities

- Relationship is "made into" an entity by putting it into a ***rectangle***; note ***that the edge between Buys and Approves touches the Buys rectangle but not the Buys diamond***, to make sure that we won't get confused



- Note: Sometimes the symbol of diamond in a rectangle is used to denote a so-called, associative entity



Weak Entities

Strong and Weak Entities

- We have two entity sets
  - Man
  - Woman
- Woman has a single attribute, SSN
- Let us defer the discussion of the attributes of Man
- A woman has 5 sons, among them Aarav and Wei, neither of the two is her eldest son and she writes the following in her will:

My SSN is 123-45-6789 and I leave $100 to my eldest son and $200 to my son Aarav and $300 to my son Wei …

- How do we identify these 3 men from among all the men in the universe? Let's defer the answer for now

Strong and Weak Entities

- A ***strong entity*** (entity set): Its elements ***can be identified*** by the values of their attributes, that is, a (primary) key can be made of its attributes

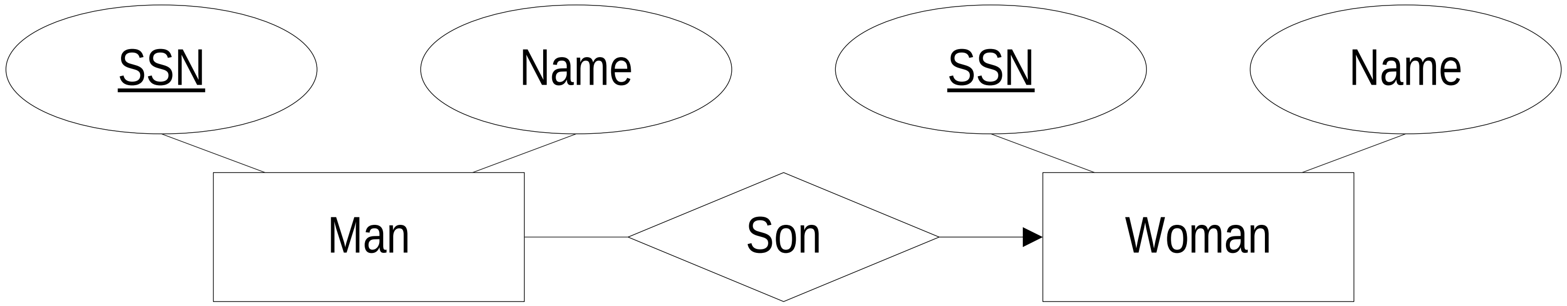
  Without saying that, we assumed only such entities so far

- ***A weak entity*** (entity set): Its elements ***cannot be identified*** by the values of their attributes: that is, no primary key can be made from its own attributes

  Such entities can be identified by a combination of their attributes and the relationship they have with an entity in another entity set

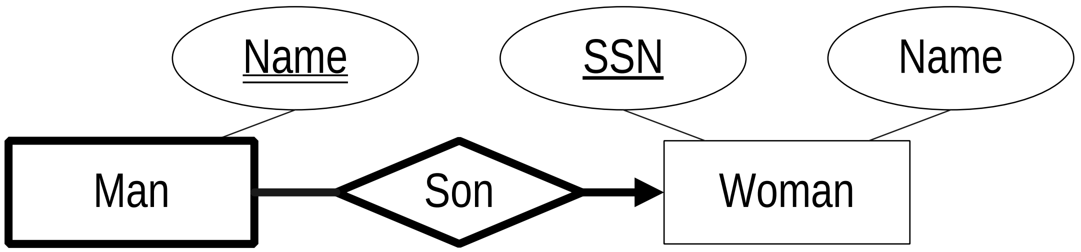
Man as a Strong Entity

- Most entities are ***strong***: a specific entity can be distinguished from other entities based on the values of its attributes
- We assume that every person has his/her own SSN
- Woman is a strong entity as we can identify a specific woman based on her attributes. She has a primary key: her own SSN
- Man is a strong entity as we can identify a specific man based on his attributes. He has a primary key: his own SSN



Man as a Weak Entity

- We assume that women are given SSNs
- Men are not given SSNs; they ***have first names only, but for each we know who the mother*** is (that is, we know the SSN of the man's mother)
- Man is a ***weak*** entity as we cannot identify a specific man based on his own attributes
- Many women could have a son named Bob, so there are many men named Bob
- However, if a woman ***never gives a specific personal name to more than one of her sons***, a man can be identified by his name and by his mother's SSN
- ***Note double underline and thick lines indicating the above***



Man as a Weak Entity

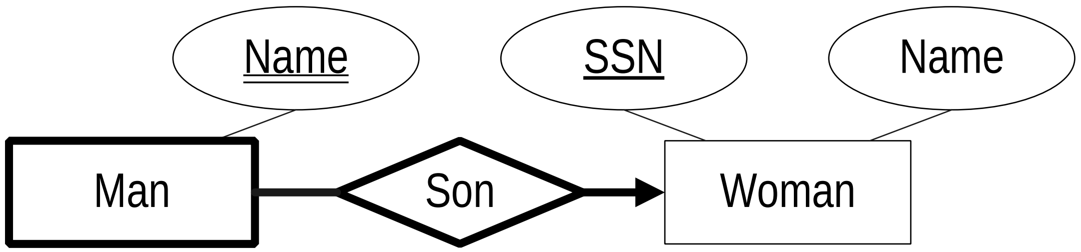
- We could have the following situation of 2 mothers: one with 2 sons, and one with 3 sons, when we gave people also heights in inches (just to have additional attributes that are not necessary for identification)

- SSN: 070-43-1234, height: 65
  - Name: Bob, height 35
  - Name: Dao-I, height 35

- SSN: 056-35-4321, height 68
  - Name: Bob, height 35
  - Name: Mohammed, height 45
  - Name: Vijay, height 84

Man as a Weak Entity

- Assume that a woman does not have more than one son with a specific first name
- Name becomes a ***discriminator; a discriminator cannot be NULL***
- Man can be identified by the combination of:
  - The Woman to whom he is related under the Son relationship. This is indicated by thick lines around Son (it is weak). Thick line connecting Man to Son indicates the relationship is total on Man (every Man participates) and it used for identification
  - His Name. The Name is now a ***discriminator;* this is indicated by double underline**
- We add an arrowhead for consistency, although we know that semantically the relationship must be many-to-one

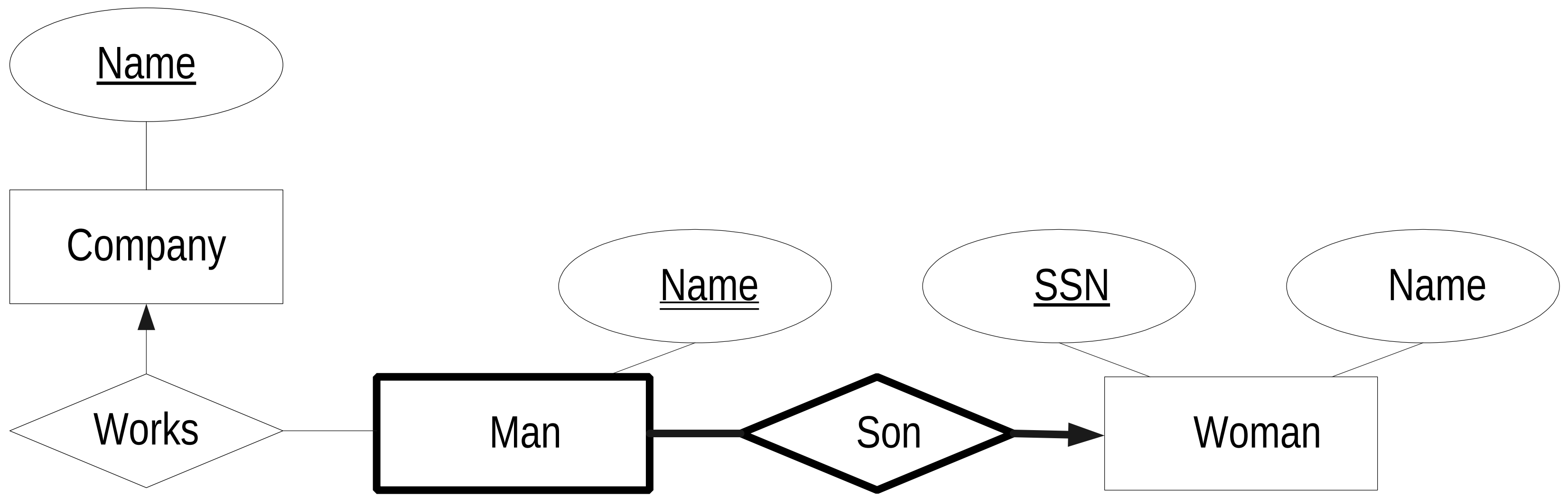


Man as a Weak Entity

- ***Discriminator*** is used to distinguish among Men related to the Woman
- It is always known, similarly to the primary key of strong entities
- This is the first meaning of discriminator
- There is another meaning, which we will see when we study the ISA relationship
- The term is, unfortunately, overloaded, as the two meaning are not related
- Note that the term "discriminate" has no negative connotations per se
    - "[T]o recognize that there is a difference between people or things; to show a difference between people or things". From https://www.oxfordlearnersdictionaries.com/us/definition/american_english/discriminate#:~:text=1%5Bintransitive%2C%20transitive%5D%20to,discriminate%20(between%20A%20and%20B)

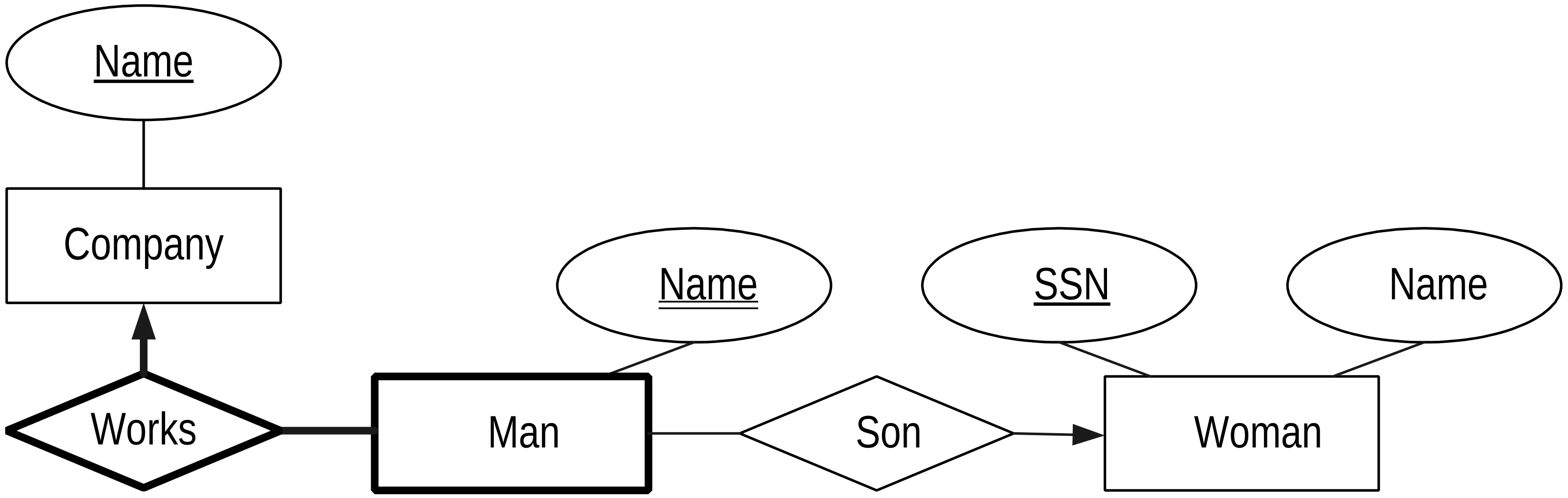
Man as a Weak Entity

- We need to specify for a weak entity **through which relationship it is identified; this is done by using thick lines**
- Otherwise, we do not know whether Man is identified through Son or through Works



Man as a Weak Entity

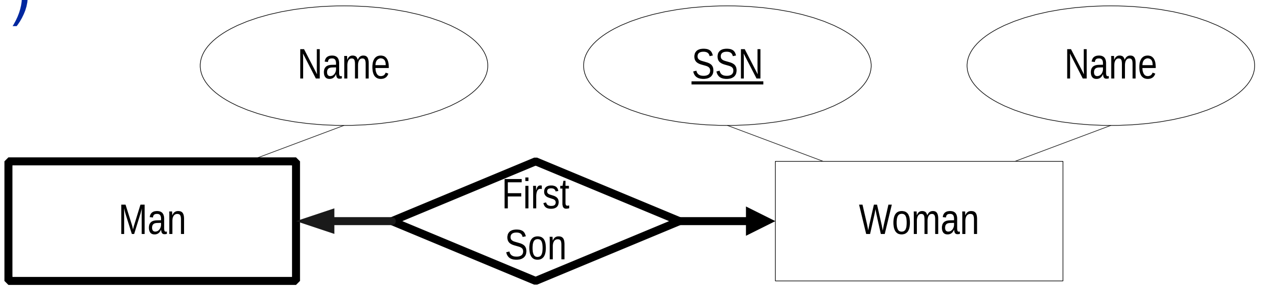
- We need to specify for a weak entity **through which relationship it is identified; this is done by using thick lines**
- Otherwise, we do not know whether Man is identified through Son or through Works



- Thick lines for the relationship are sufficient; not necessary to have thick lines in other parts of the diagram

Man as a Weak Entity

- Sometimes a discriminator is not needed
- We are only interested in men who happen to be first sons of women
- Every Woman has at most one First Son
- So, we do not need to have Name for Man (if we do not want to store it, but if we do store it, ***it is not a discriminator***)



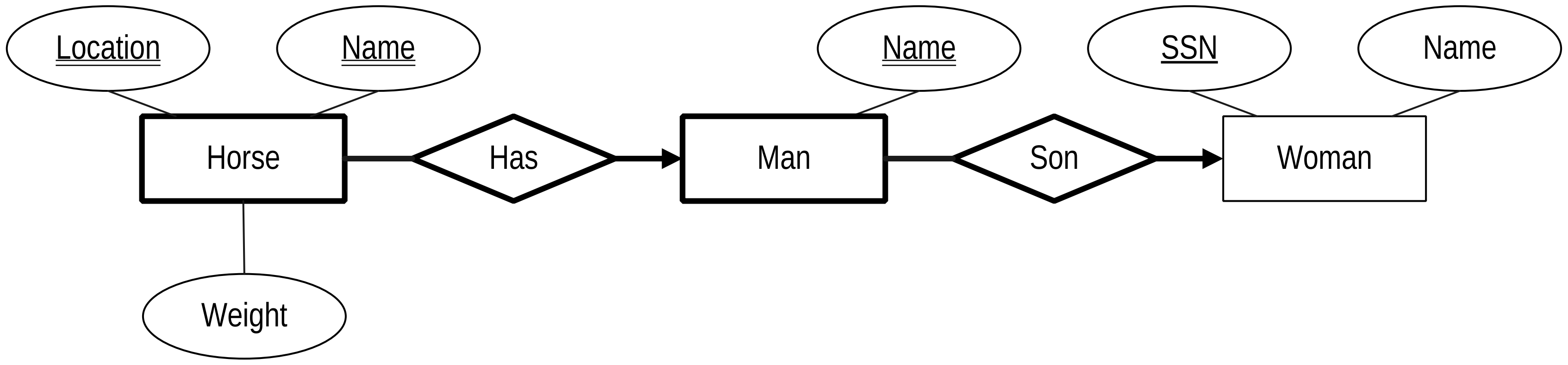
- Note an arrowhead to the left: each Woman has at most one First Son
  - This has nothing to do with Man being a weak entity in our design; it is just a semantic fact about the world
  - This is a one-to-one mapping

Man as a Weak Entity

- In general, more than one attribute may be needed as a discriminator
- For example, let us say that man has both first name and middle name
- A mother may give two sons the same first name or the same middle name
- A mother will never give two sons the same first name and the same middle name
- The pair (first name, middle name) together form a discriminator

From Weaker to Stronger

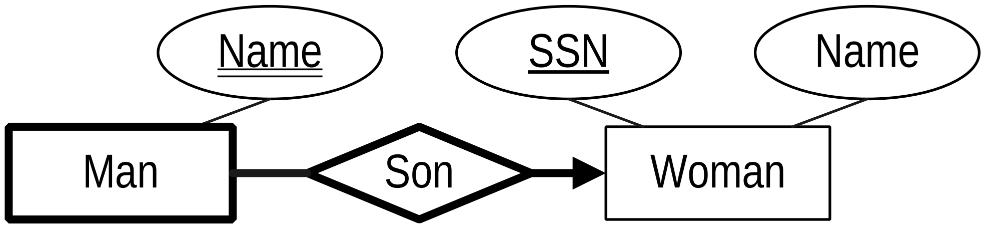
- There can be several levels of “weakness”
- Here we can say that a horse named Speedy located in Saratoga Springs belongs to Bob, whose mother is a woman with SSN 072-45-9867



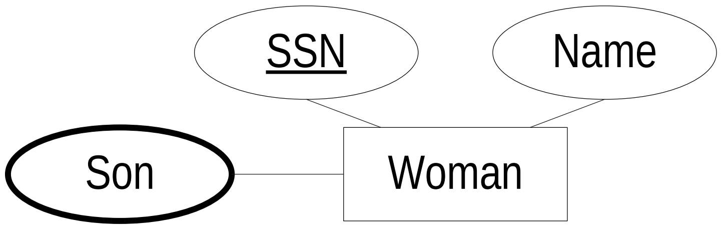
- A woman can have several sons, each of whom can have several horses

Weak Entity or Multivalued Attribute?

- We had



- If nothing in the description of the application leads us to believe that Man could have its own attributes or could participate in relationships (other than with his mother), it might be more natural to implement the fragment of the database using a multivalued attribute
  - **That's what you should do in all your relevant assignments**



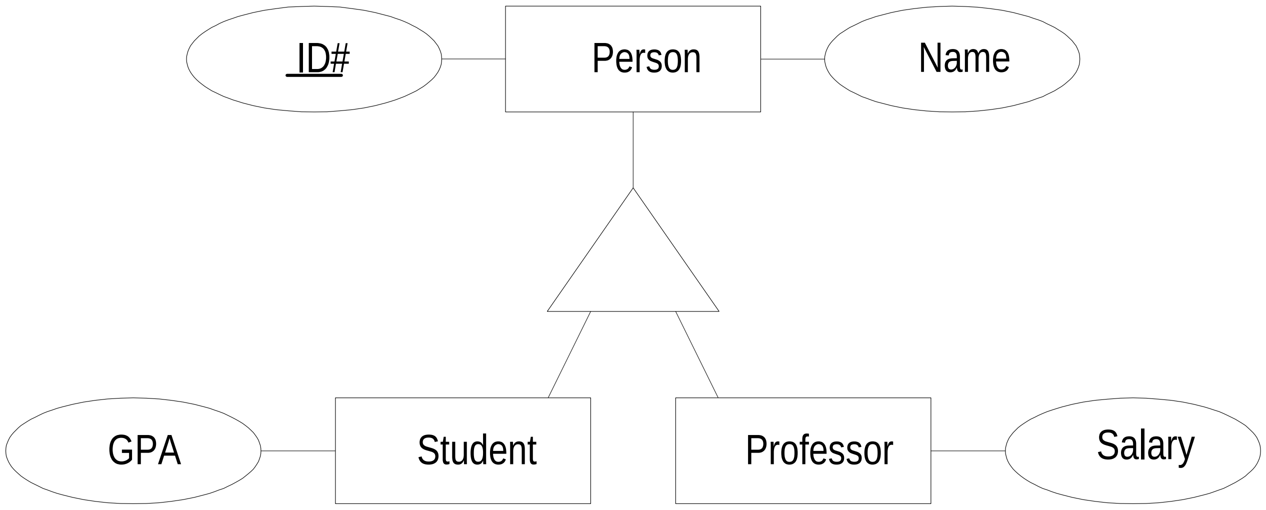
Hierarchy (Class/Subclass)
ISA

The ISA Relationship

- For certain purposes, we consider subsets/subclasses of an entity set
- The subset relationship between the set and its subset is called ***ISA***, meaning "***is a***"
- Elements of the subset, of course, have all the attributes and relationships as the elements of the set: they are in the "original" entity set
- In addition, they may participate in relationships and have attributes that make sense for them
  - But those relationships and attributes do not make sense for every entity in the "original" entity set
- The set and the subsets are sometimes referred to as ***class*** and ***subclasses***
- ISA is indicated by a triangle; some write ISA in the triangle
- The elements of the subset are weak entities, as we will note next

The ISA Relationship

- Example: A subset that has an attribute that the original set does not have
- We look at all the persons associated with a university
- Some of the persons happen to be professors and some of the persons happen to be students
  - Student is a subset of Person
  - Professor is a subset of Person



- Student and Professor “inherit” ID# as their primary keys

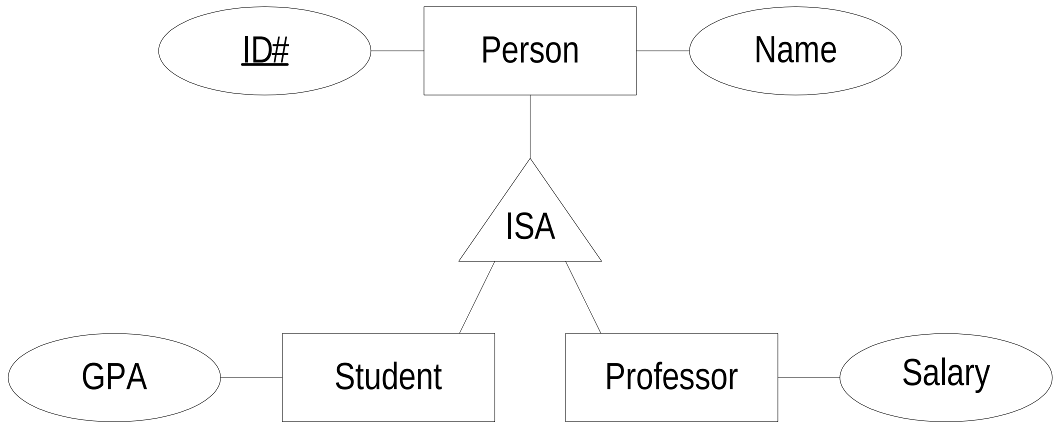
ISA Is A Set Of One-To-One Functions

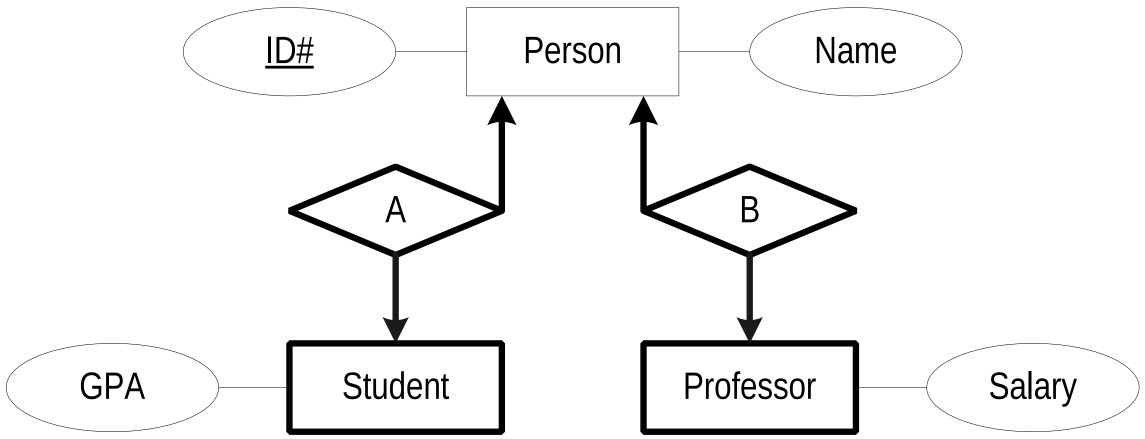
- In our example there is one-to-one function between Person and Student
- Person and Student can be thought of informally as two instantiations/roles of the same entity

- In our example there is one-to-one function between Person and Professor
- Person and Professor can be thought of informally as two instantiations/roles of the same entity

- The only permitted primary keys of Student and Professor are the ID#’s of the associated entities in Person

It Is Possible to Represent Hierarchies Using "Standard" Relationships





- But not a good idea as less clear: **don't do that**
- We put "dummy" names A and B to follow our convention that relationships have names

Discriminators

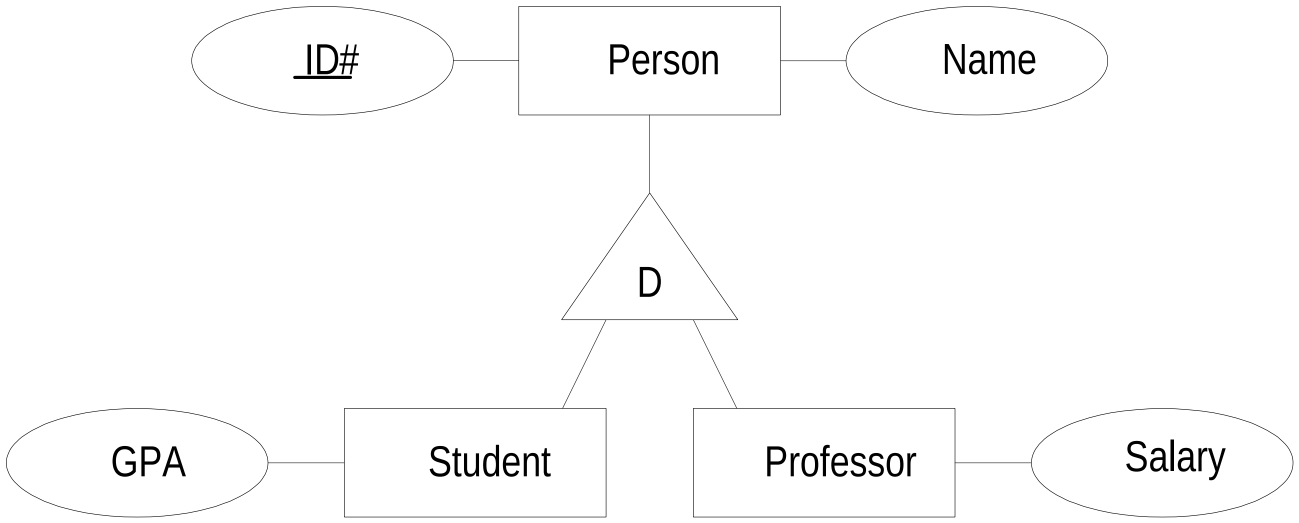
- Professor is a kind of weak entity because it cannot be identified by its own attributes (here: Salary)
- Student is a kind of weak entity because it cannot be identified by its own attributes (here: GPA)
- They do not have discriminators, they are identified by the primary key of the strong entity (Person)
- We do not need to put Student and Professor in a rectangle with thick lines as being “under” ISA automatically tells us that they are weak entities identified by the primary key of the strong entity
- There may be an attribute of Person (called ***discriminator***) that can be used to determine who is a Student and who is a Professor
  - For a fake example: Professor names start with a A–M and Student names start with N–Z (but note advisement in our department for a real example)
- Note: different meaning for discriminator than in weak entities

The ISA Relationship

- A person associated with the university (and therefore in our database) can be in general
  - Only a professor
  - Only a student
  - Both a professor and a student
  - Neither a professor nor a student
- A specific ISA could be
  - ***Disjoint***: no entity could be in more than one subclass
  - ***Overlapping***: an entity could be in more than one subclass
  - ***Total*** (also called ***Complete***): every entity must be in at least one subclass
  - ***Partial***: an entity does not have to be in any subclass
- If nothing stated, then no restriction
  - “Overlapping” is not a restriction
  - “Partial” is not a restriction

The ISA Relationship (Example)

- Some Persons can be Professors
- Some Persons can be Students
- Some Persons can be neither Professors nor Students
- No Person can be both a Professor and a Student

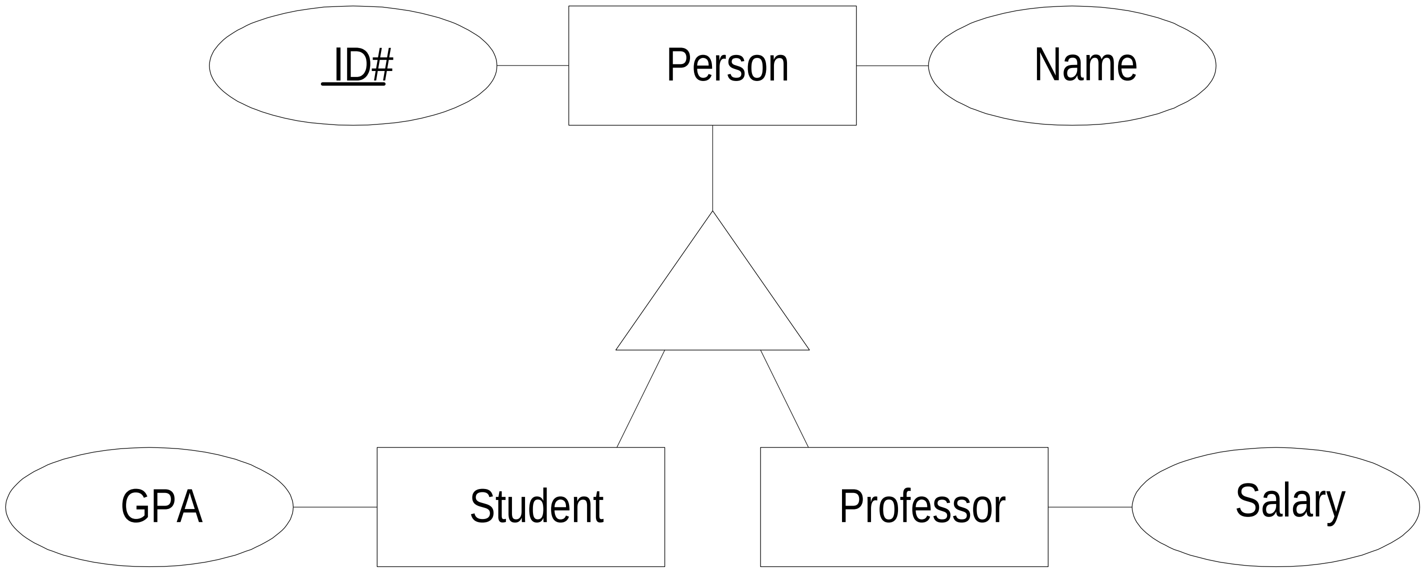


- We do not put P in the triangle as that is not a restriction

- If applicable, can put both D and T in the triangle

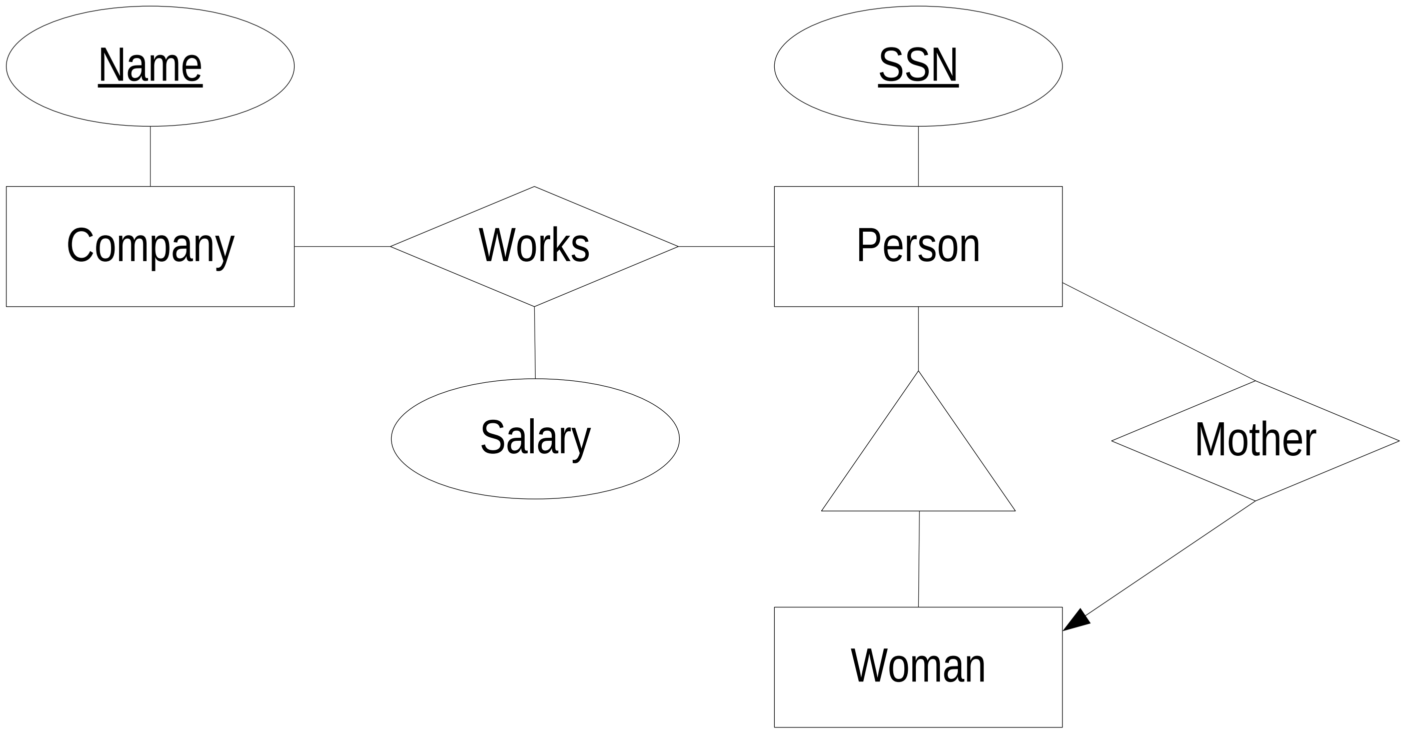
The ISA Relationship (Example)

- If there are no constrains on the ISA relationship leave the triangle empty
- To repeat, note that the following are not constraints
  - Partial (does not have to be total)
  - Overlapping (does not have to be disjoint)
  - **Therefore, do not state them in your assignments**



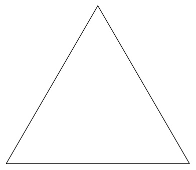
The ISA Relationship

- Example: subsets participating in relationships modeling the assumed semantics more clearly (every person has at most one woman who is the biological mother) for people in the database

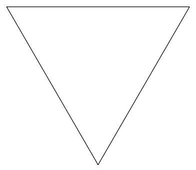


The ISA Relationship

- ISA is really a superclass/subclass relationship
- ISA could be ***generalization***: a superset is made from "more basic" sets
- ISA could be ***specialization***: subsets are made from the "more basic" set
- This information could be added in annotations, if desired
- Sometimes the "orientation" of a triangle is used to distinguish between specialization and generalization
  - We will not do that

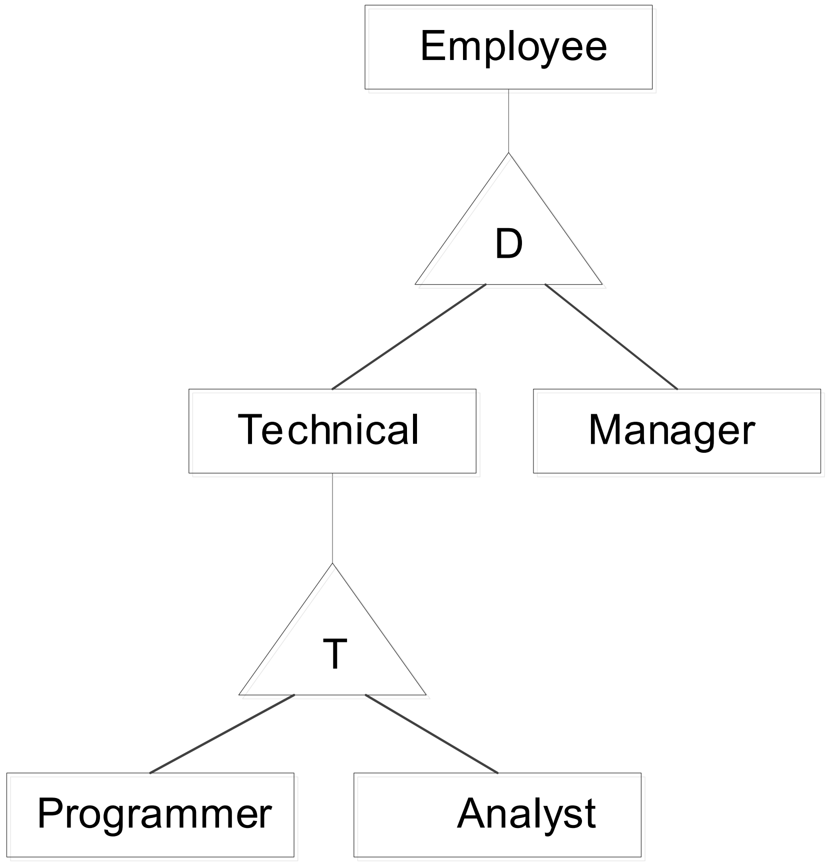


Generalization



Specialization

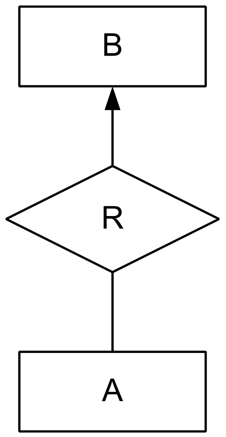
A Sketch of a More Complex ER Diagram With Hierarchical ISAs

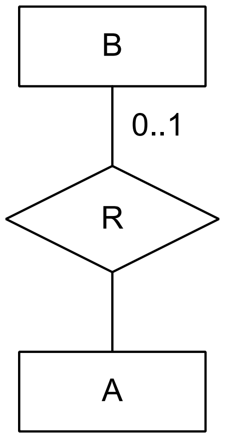


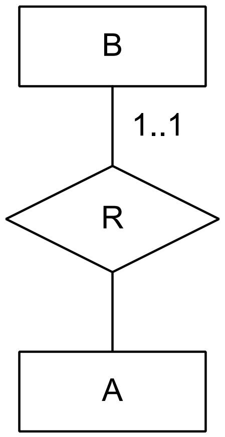
Cardinality Constraints

Cardinality Constraints

- We can generalize the arrow notation
- Best explained by examples







Starting from an element of A we end up in 0 or 1 elements of B

Starting from an element of A we end up in between 0 or 1 elements of B

Starting from an element of A we end up in between 1 and 1 elements of B

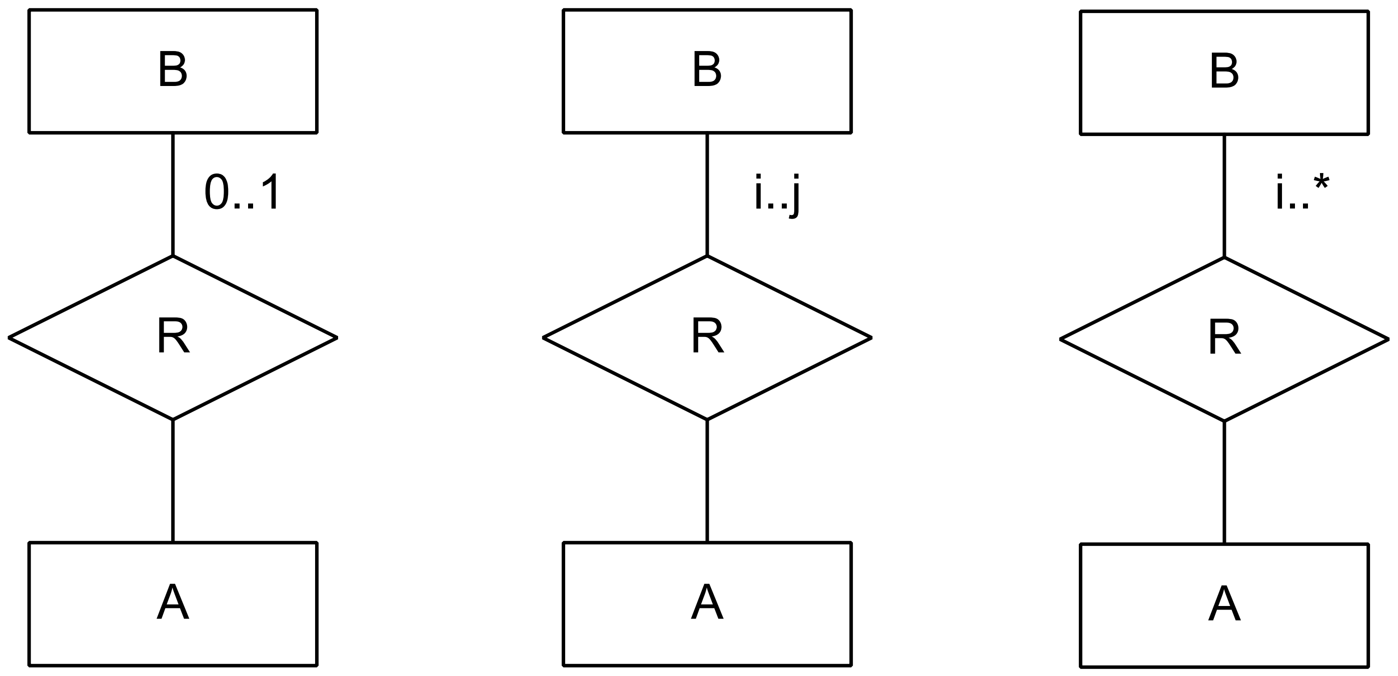
The function is partial unless an annotation states: total

The function is partial

The function is total

Cardinality Constraints

- Can specify the range i..j, meaning between i and j
- An asterisk, *, stands for unbounded



Cardinality Constraints

- Further generalizations are possible

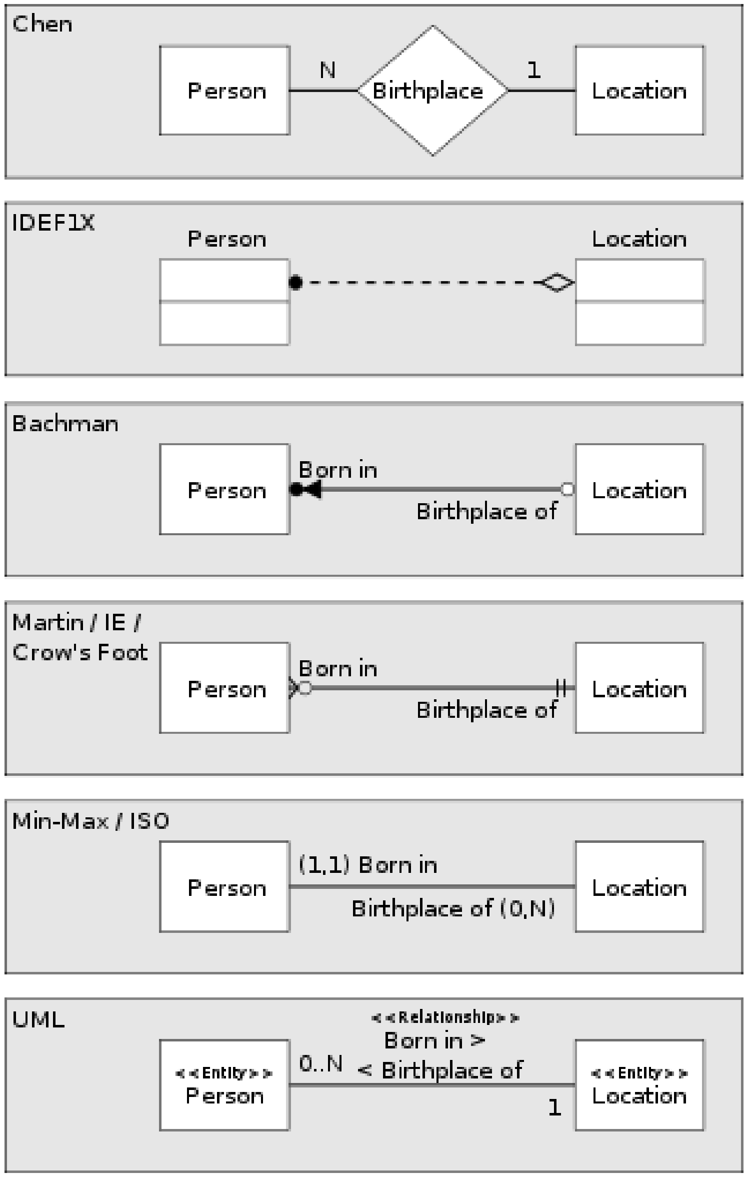
- The cardinality notation allows for a richer set of constraints
- The arrow notation is a special case

- However, the arrow notation is preferred and that's what we will use unless stated otherwise

- We will discuss that further in the unit dealing with relational implementations

Many "Standards" Just For Binary Many-to-One

- https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model



Software for Modeling

Drawing Software For ER Diagrams

- Any software that can produce the shapes we need and connect them appropriately would be fine
  - We could have used the drawing shapes in PowerPoint
- We will use draw.io at https://app.diagrams.net/
  - It is web-based so works for all operating systems; you may be able to install local copy if you like from https://www.diagrams.net/
  - It is easy to use and will let everybody use the same tool for the assignments
  - It has the shapes we need though a slight a variant from what I produced for the lectures
  - We will see how to use it later

Case Study

A Case Study

- Next, we will go through a relatively large example to make sure we know how to use ER diagrams
- We have a largish fragment of a large application to make sure we understand all the points
- The fragment has been constructed so it exhibits interesting and important capabilities of modeling
- It will also review the concepts we have studied earlier

- It is chosen based on its suitability to practice modeling using the power of ER diagrams
- It will also introduce some points, to be discussed later on how to design good relational databases

- We will use the standard, arrow notation, which is understood by all

Our Application

- We are supposed to design a database for a university
- We will look at a small fragment of the application and will model it as an entity relationship diagram annotated with comments, as needed to express additional features
- But it is still a reasonable “small” database
- In fact, larger than what is typically discussed in database courses, but more realistic for modeling real applications
- I have a seen a database used by a startup to manage what’s done by it and it is roughly only twice as big as our database, and it is not as sophisticated as ours

Our Application

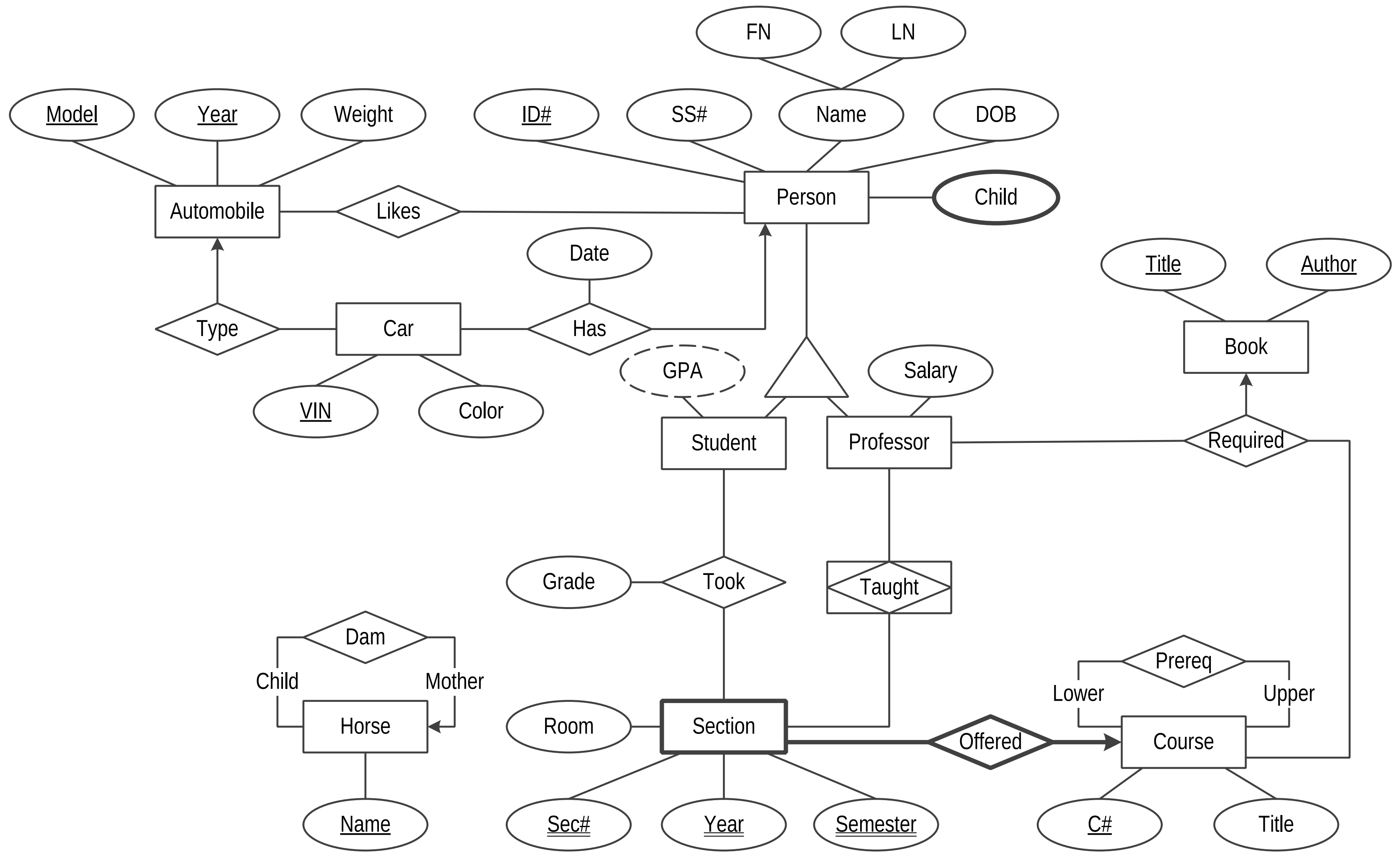
- Our understanding of the application will be described in a narrative form
- While we do this, we construct the ER diagram
- For ease of exposition (technical reasons only: limitations of the projection equipment) we look at the resulting ER diagram and construct it in pieces
- We will pick some syntax for annotations, as there is no commonly accepted standard
- One may try and write the annotations on the diagram itself using appropriate phrasing, but this will make our example too cluttered

Our Workflow

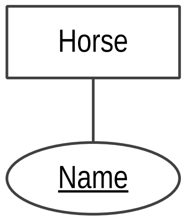
- For each component we may have some or all of the following
  1. Description in words, although we have it already in the client's narrative
  2. The ER diagram with the new component added
  3. The additional annotations required to express all that is not expressible by the diagram and only that. If the slide is empty, there are no annotations
  4. Comments for us, but not a part of the design
- We will not list the domains of the attributes, as this is trivial to do and not interesting, but in a real application, this is needed and would be put in annotations
- Unless stated or implied otherwise, the value of the attribute does not have to be known
  - ***Note, however, that the value of an attribute in a primary key or in a discriminator for weak entities is always known so do not say that***

Building the ER Diagram

- We describe the application in stages, ultimately getting the following diagram with additional annotations



Our ER Diagram So Far

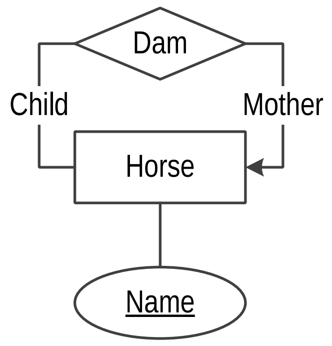


Annotations

Comments

- We must not say/write that Name in Horse is its primary key
- We see that from the diagram

Our ER Diagram So Far

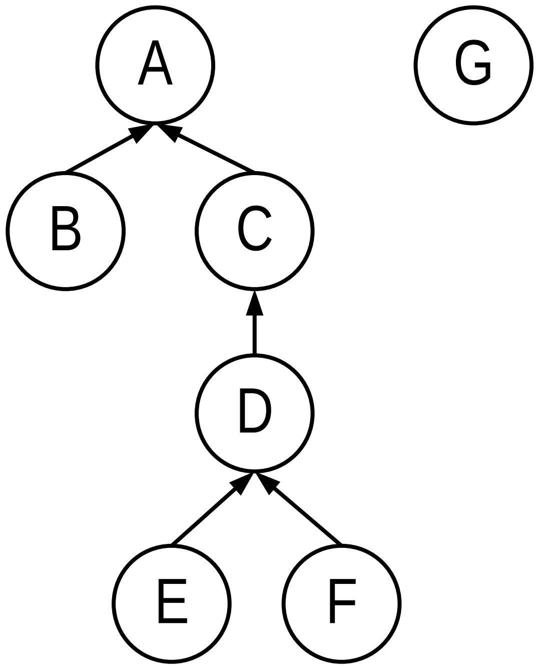


Annotations

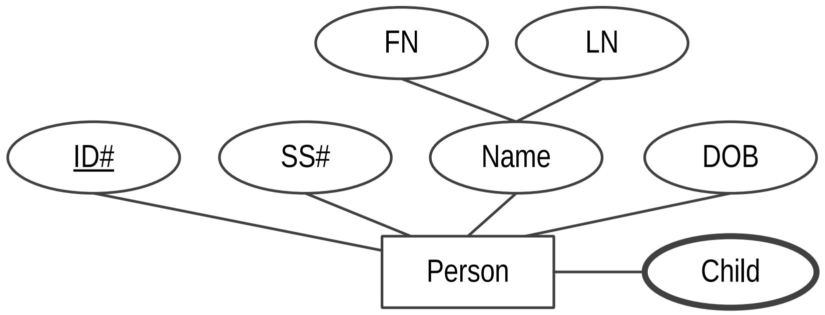
- The graph describing the Dam relationship is a forest of rooted trees

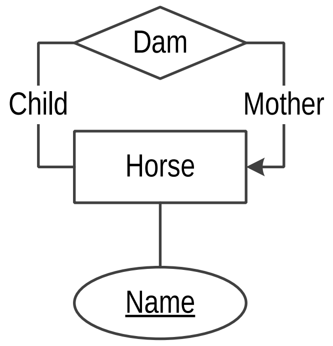
Comments

- Dam (mother of a Horse) is a partial function
- We are not told that it is total, so we have the default case of partial
- Example of Dam



Our ER Diagram So Far





Annotations

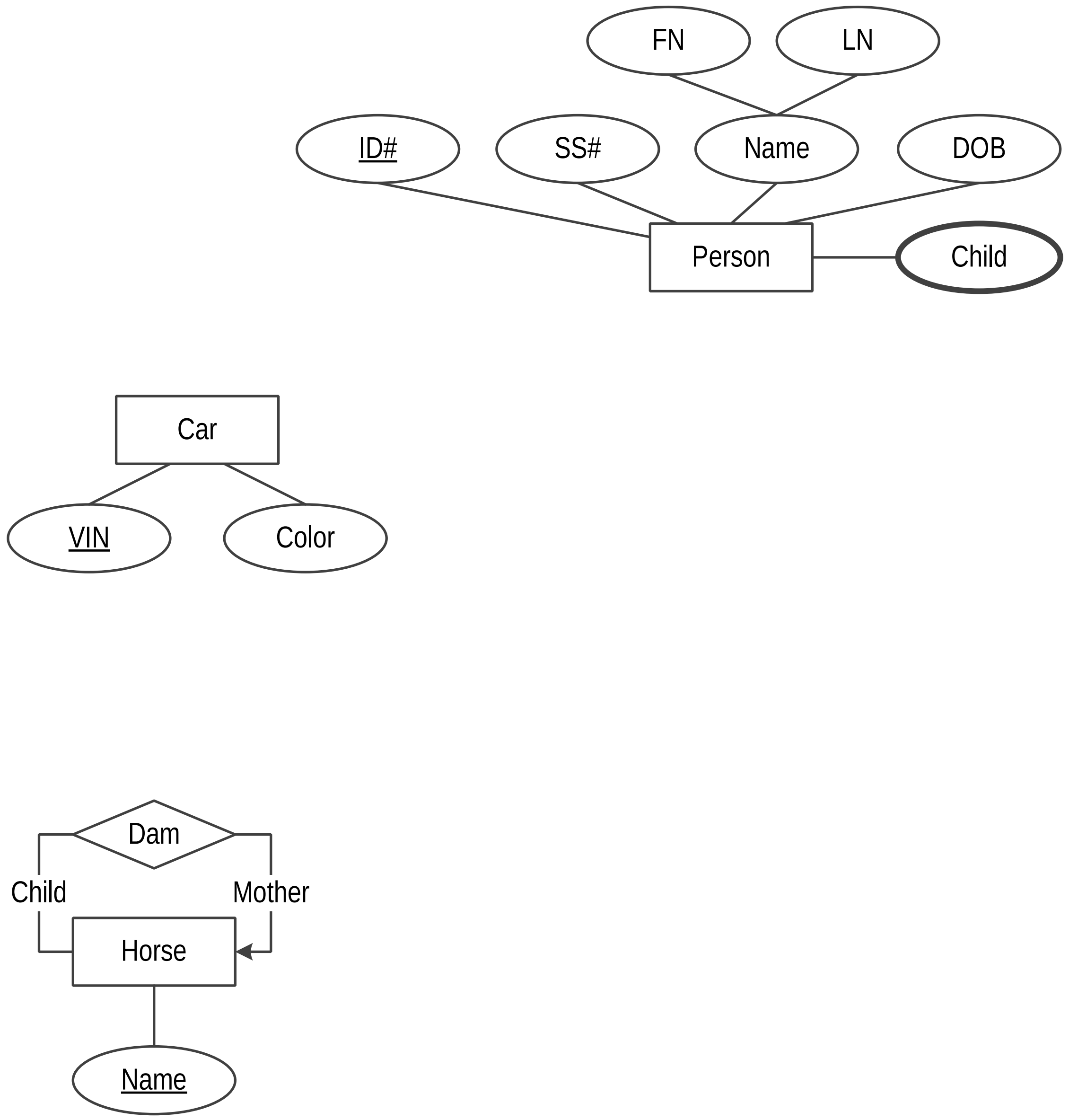
- LN, SS#, DOB in Person are always known
- No two Persons can have the same SS#

Comments

- It is very important to specify what we did for SS#
- Semantically, SS# could have been chosen to be the primary key
- The DBMS must enforce that

- Note that because in our specifications in the narrative, Child did not have any attributes and did not participate in any relationship, we made Child to be a multivalued attribute and not an entity with a relationship to a Person
- We also assumed that all the Children of a Person are different, that is a Person does not give the same name/label to two different children

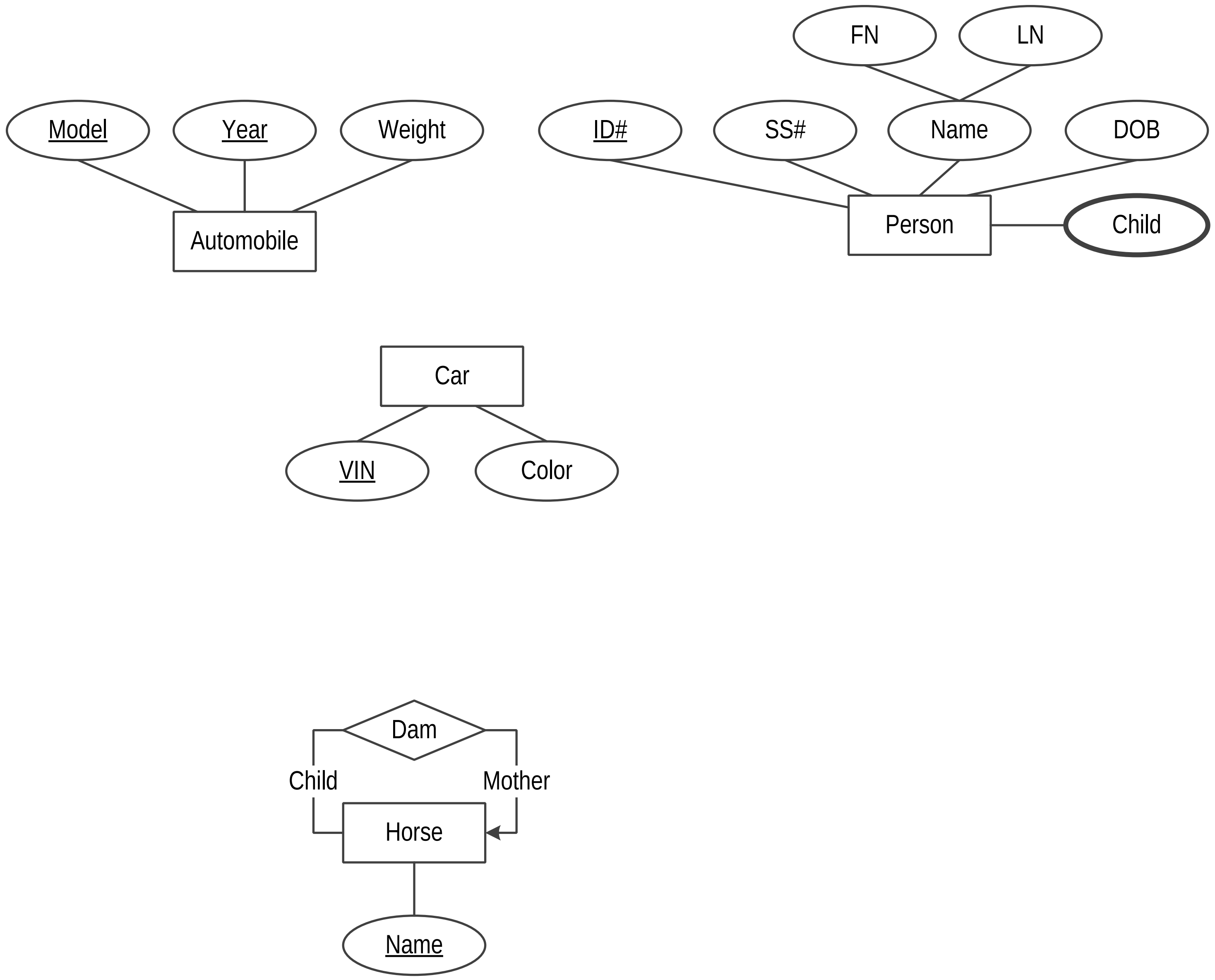
Our ER Diagram So Far



Annotations

- Color in Car is always known

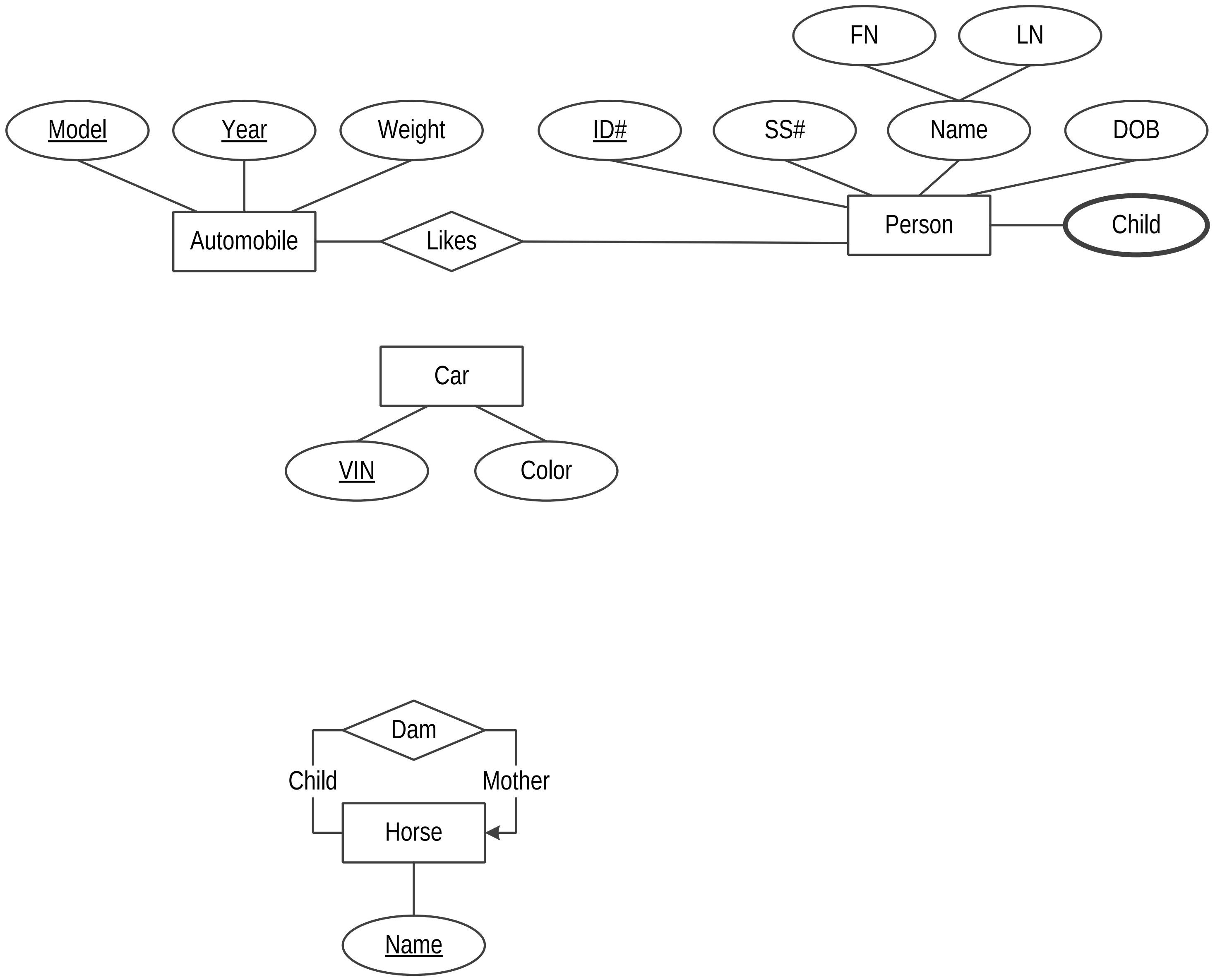
Our ER Diagram So Far



Annotations

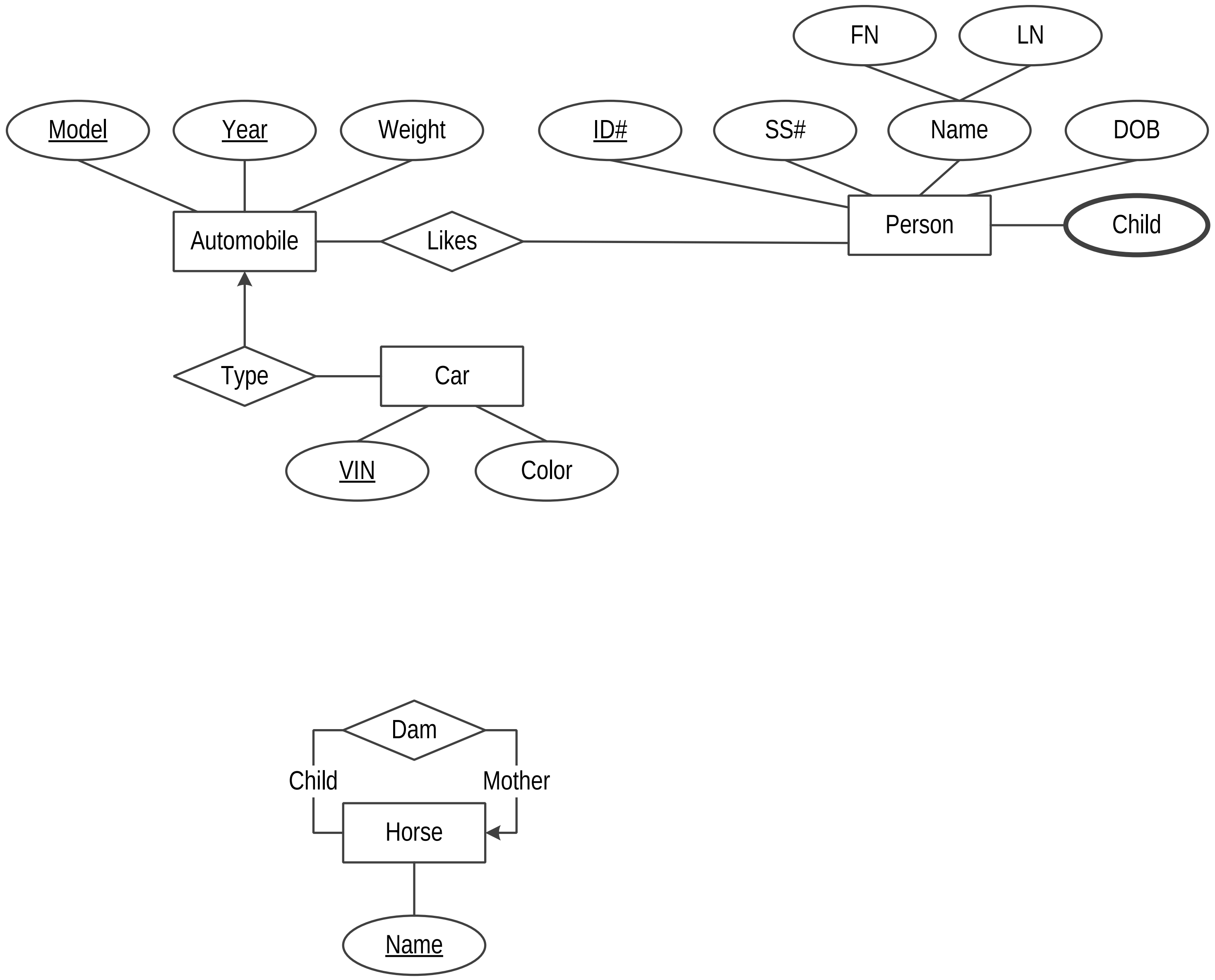
- Weight in Automobile is always known

Our ER Diagram So Far



Annotations

Our ER Diagram So Far



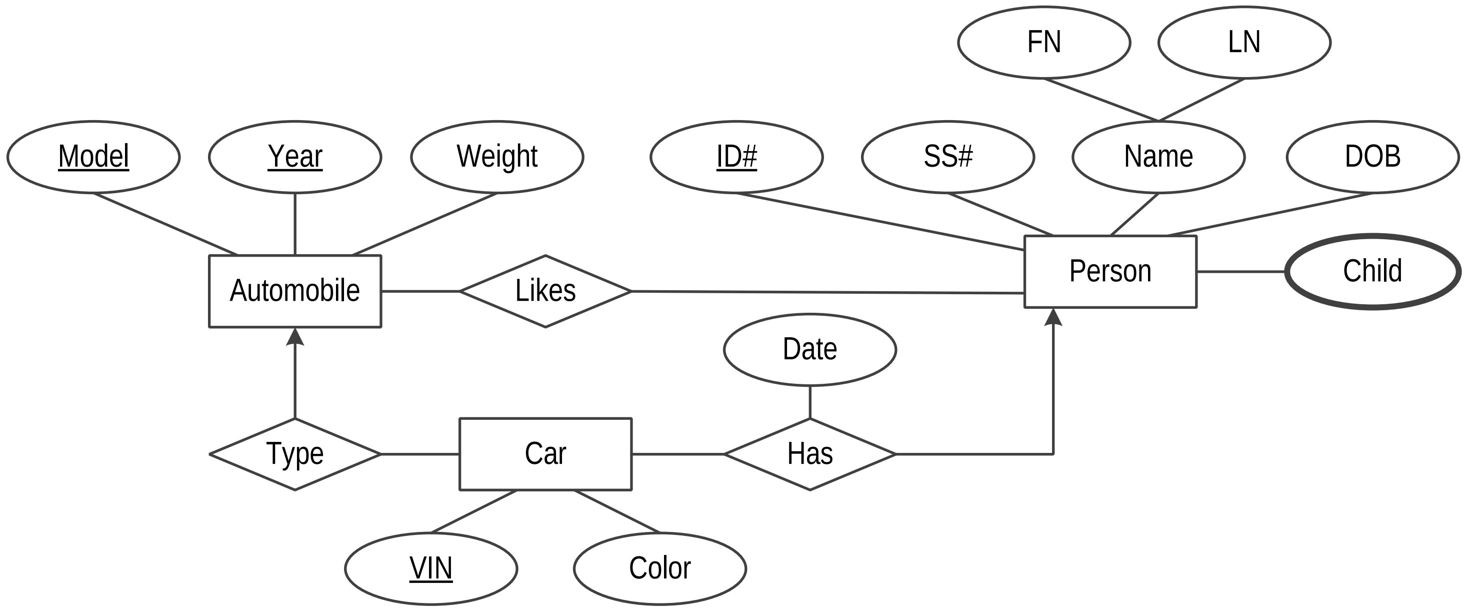
Annotations

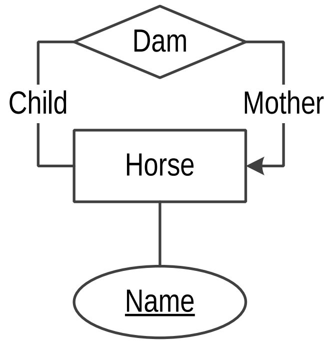
- Type is total

Comments

- An entity in Car is an object (in which people can ride) and it is on a road (or parking lot, etc.)
- An entity in Automobile is a catalog entry, say a page in a book describing all automobiles
- Type tells us for each physical car what is the automobile catalog entry of which it is an instantiation
    - Each car is an instantiation of an exactly one catalog entry
- So the entities in Car are not a subset of the entities in Automobile.
- That's why we do not have ISA here
- We will see ISA later
- Not every Automobile is necessarily in the range of the relationship
- There may be elements in Automobile for which no Car exists

Our ER Diagram So Far

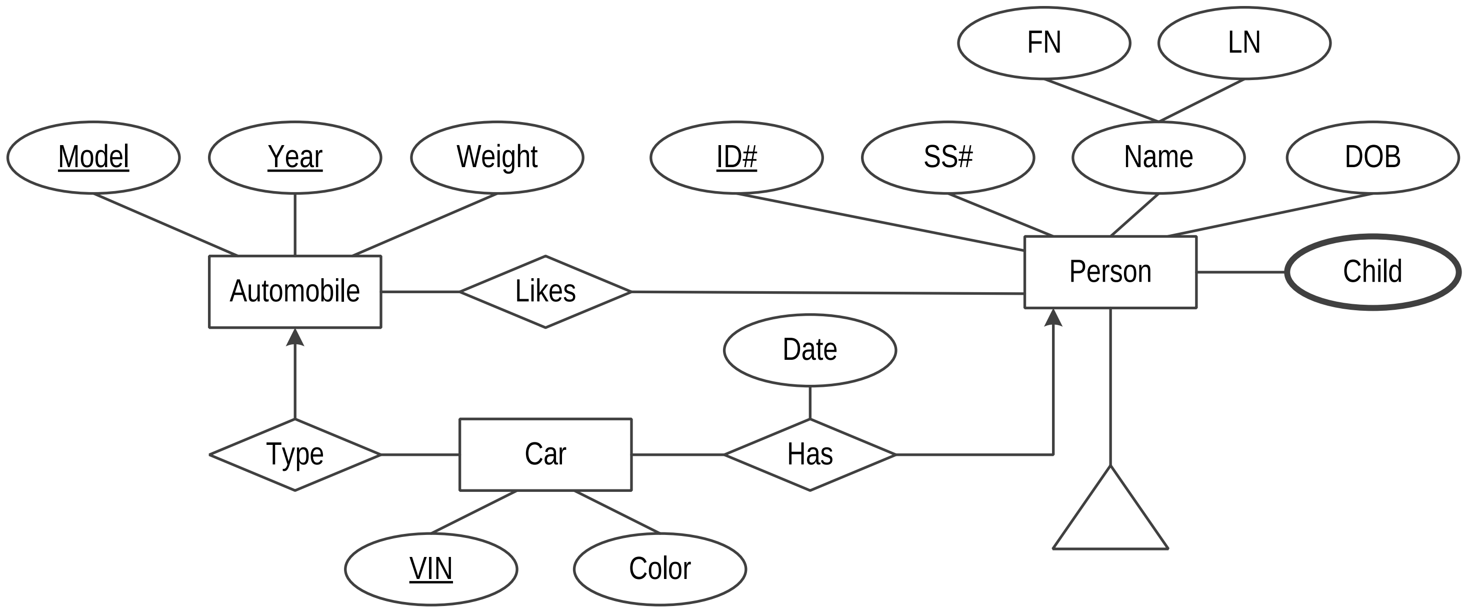


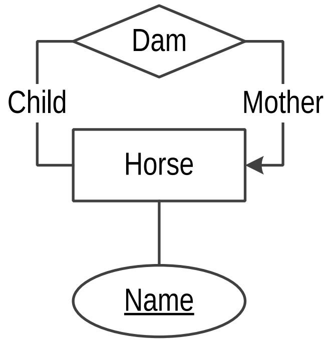


Annotations

- Every Person Has at least 2 Cars

Our ER Diagram So Far



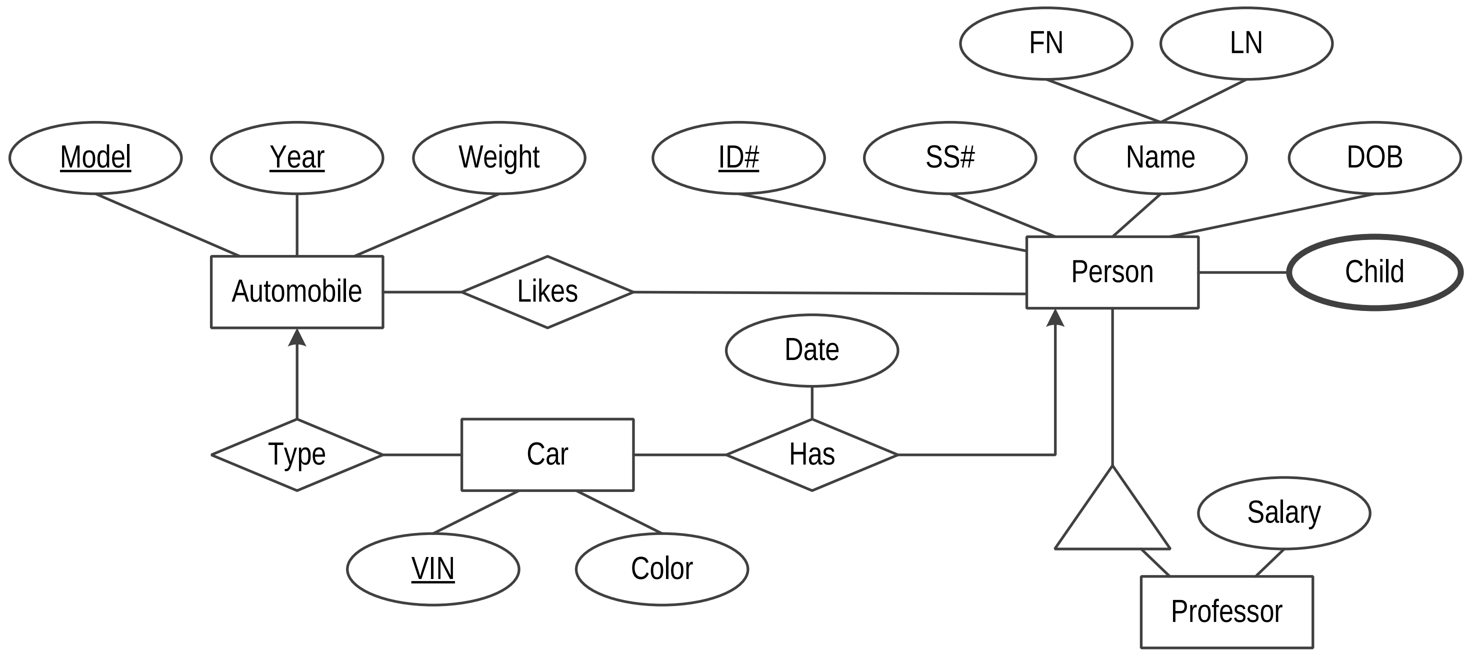


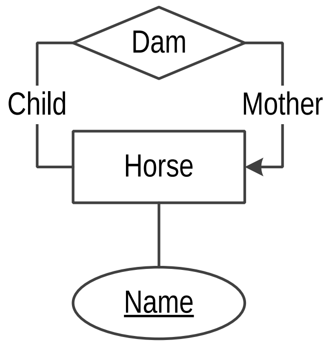
Annotations

Comments

- We do not have any letters (D or T) in the triangle and therefore there are no restrictions on the hierarchy
- So, it could be partial, and it could be overlapping

Our ER Diagram So Far





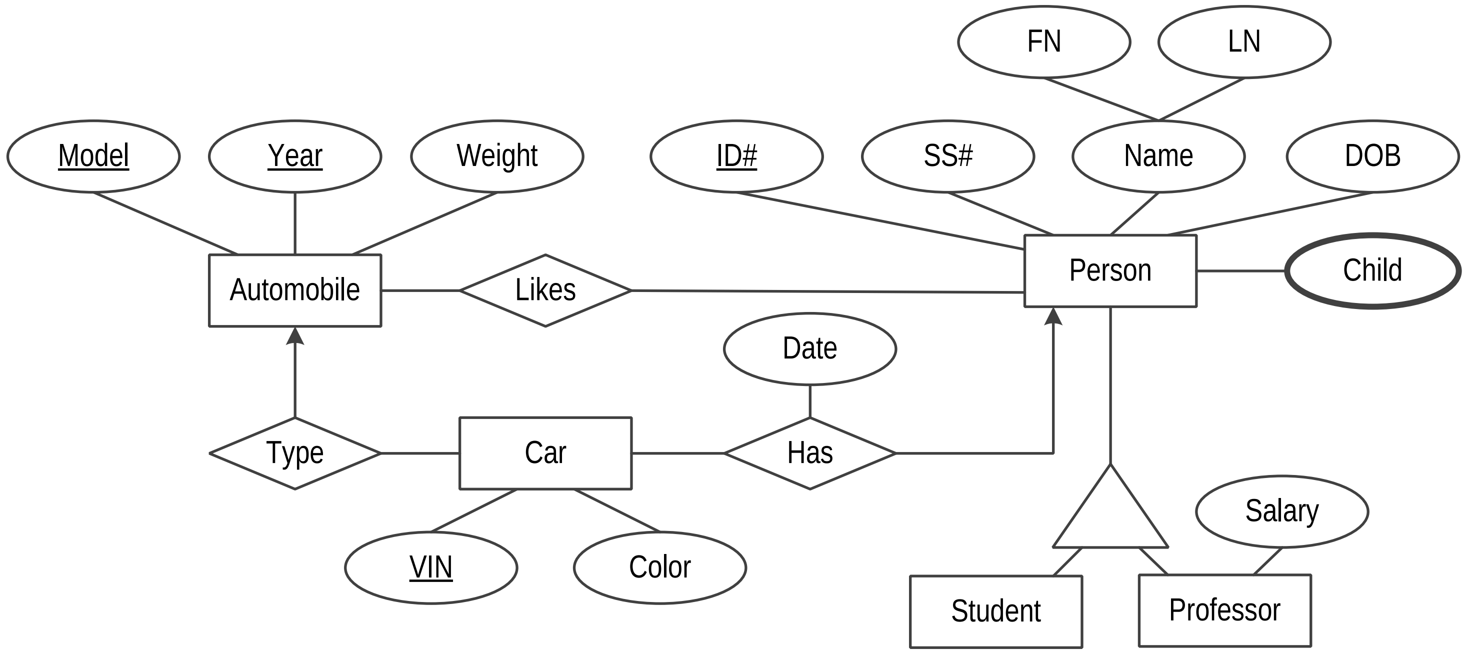
Annotations

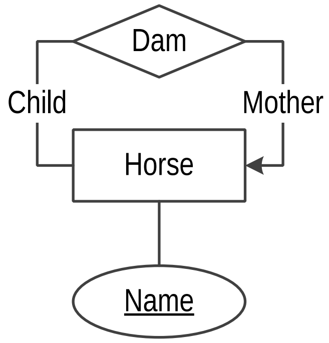
- Salary is always known

Comments

- Professor is a weak entity
  - This is generally true for related subclass/class
  - It is identified through a relationship with Person
  - You may think of a Professor as being an "alias" for some person: "Split personality"
- Note that there is a binary one-to-one mapping/function between Professor and Person and
  - It is total from Professor to Person
  - This is generally true for related subclass/class

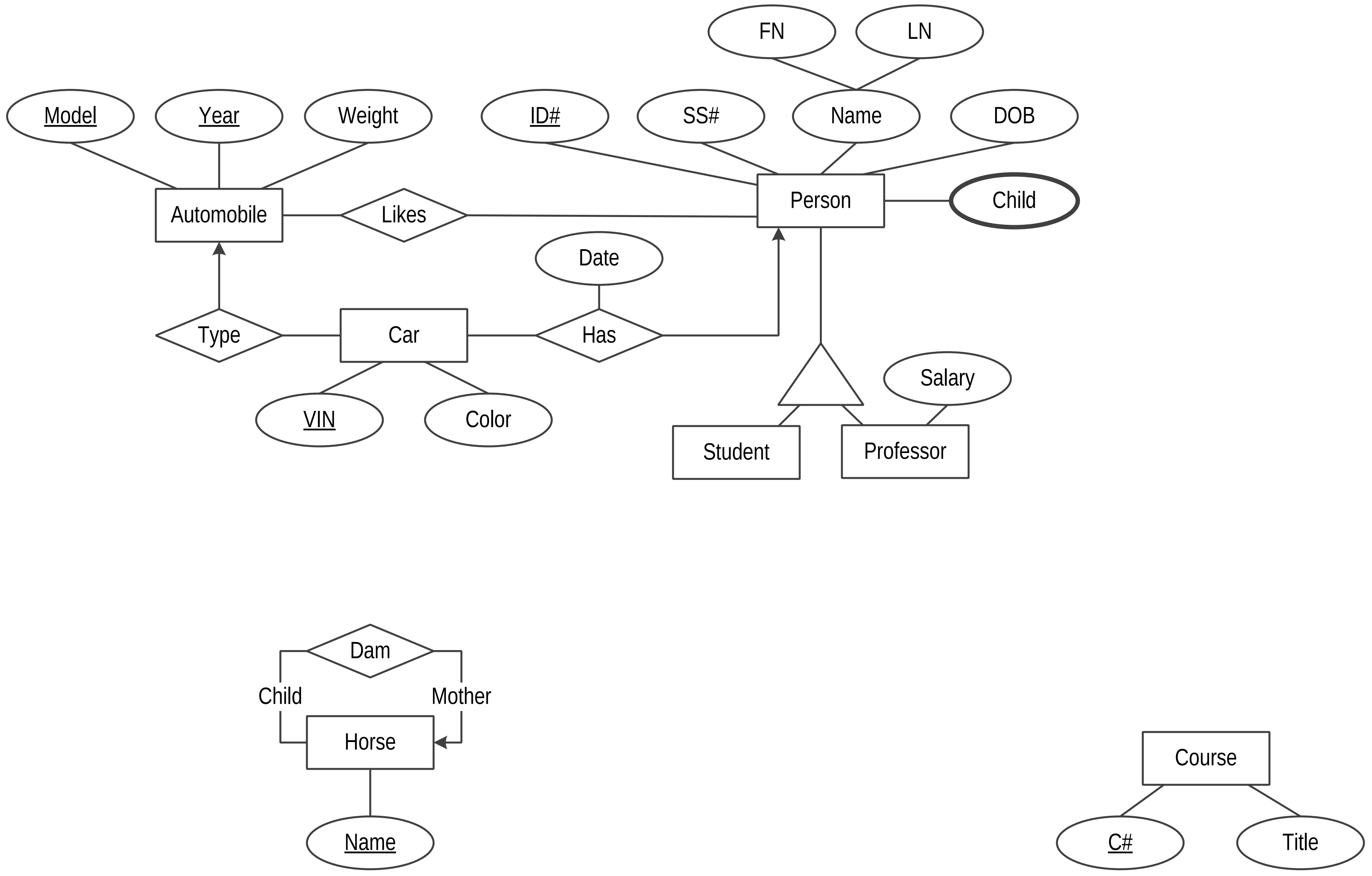
Our ER Diagram So Far





Annotations

Our ER Diagram So Far



Annotations

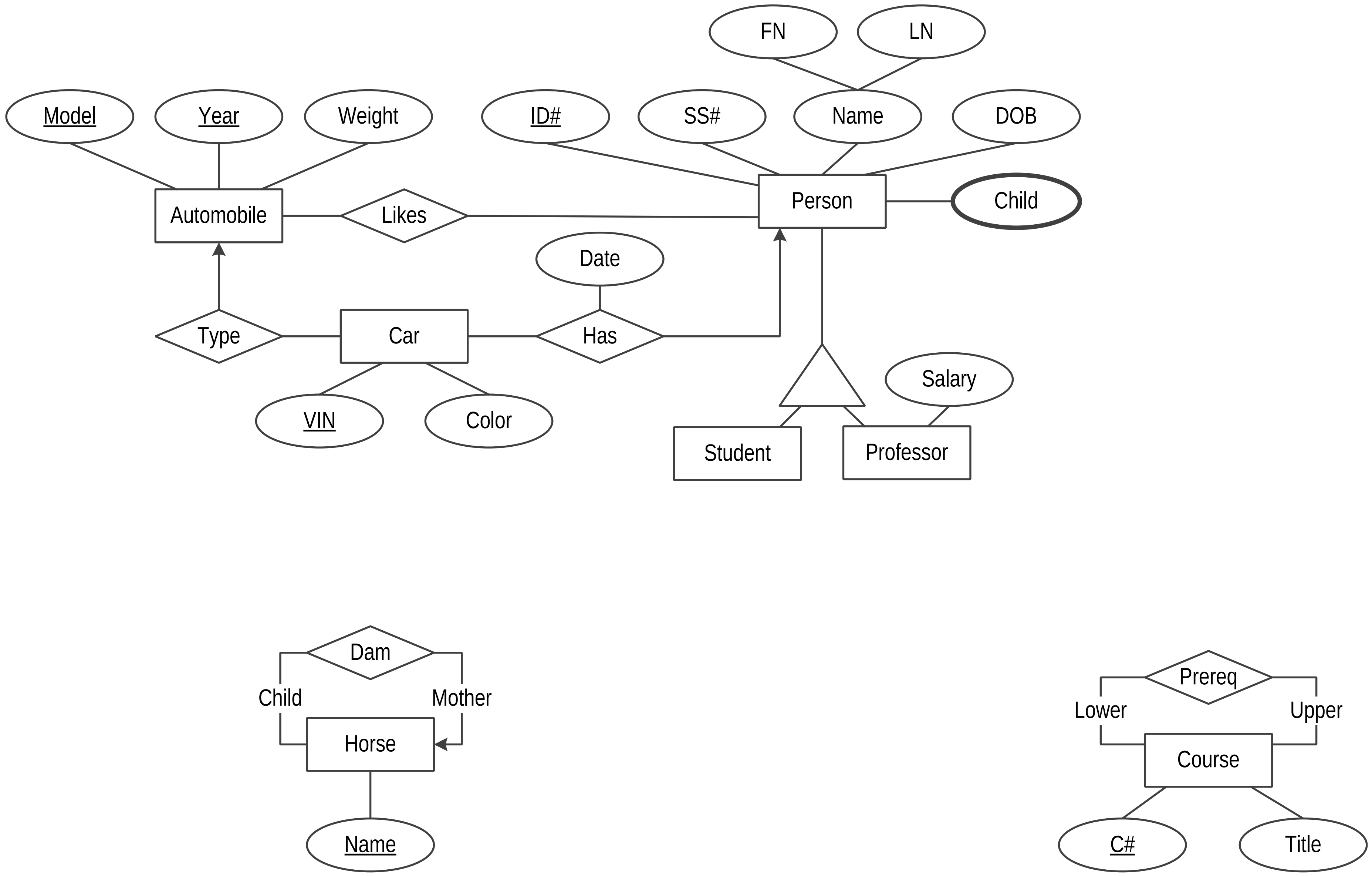
- Title in Course is always known

Comments

- Course is a catalog entry appearing in the bulletin
  - Not a particular offering of a course
  - Example: CSCI-GA.2433 (which is a C#), not its current offering

- In practice, we may need an attribute specifying the number of points, but let's assume that it is always 3, so we do not have the attribute

Our ER Diagram So Far



Annotations

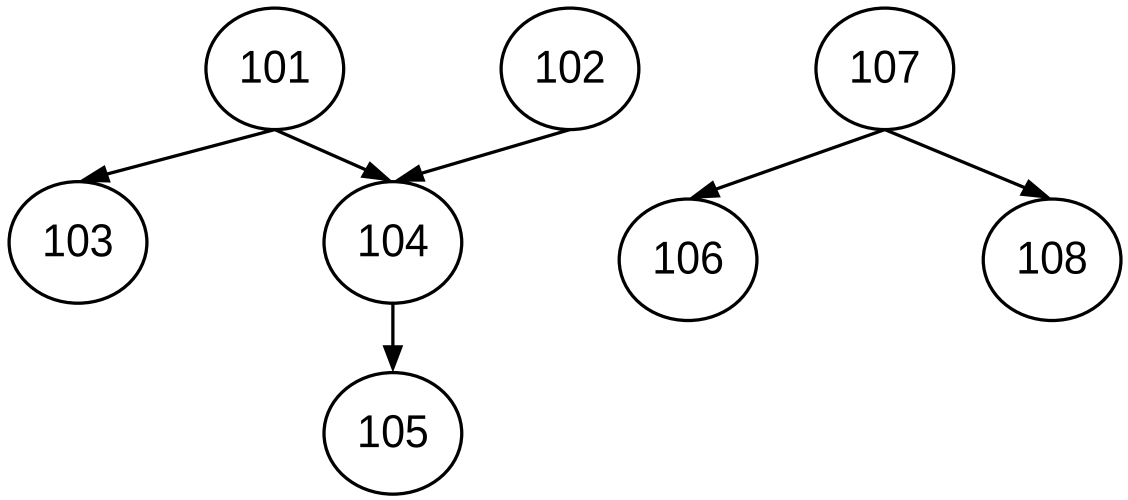
- Prereq is a DAG (Directed Acyclic Graph)

Comments

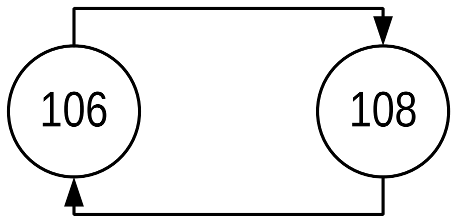
- We have a directed graph on courses, telling us prerequisites for each course, if any
  - To take an “upper” course every “lower” course related to it must have been taken previously
  - We needed the roles Lower and Upper for clarity
  - Note how we model well that prerequisites are not between offerings of a course but catalog entries of courses
  - Note however, that we have no tools to indicate in the ER diagram that Prereq must not have cycles

Comments

- Example of Prereq: lower-level Courses before upper-level courses
- As there is an arc (directed edge) from 101 to 103, 101 is a prerequisite for 103



- Non-example of Prereq



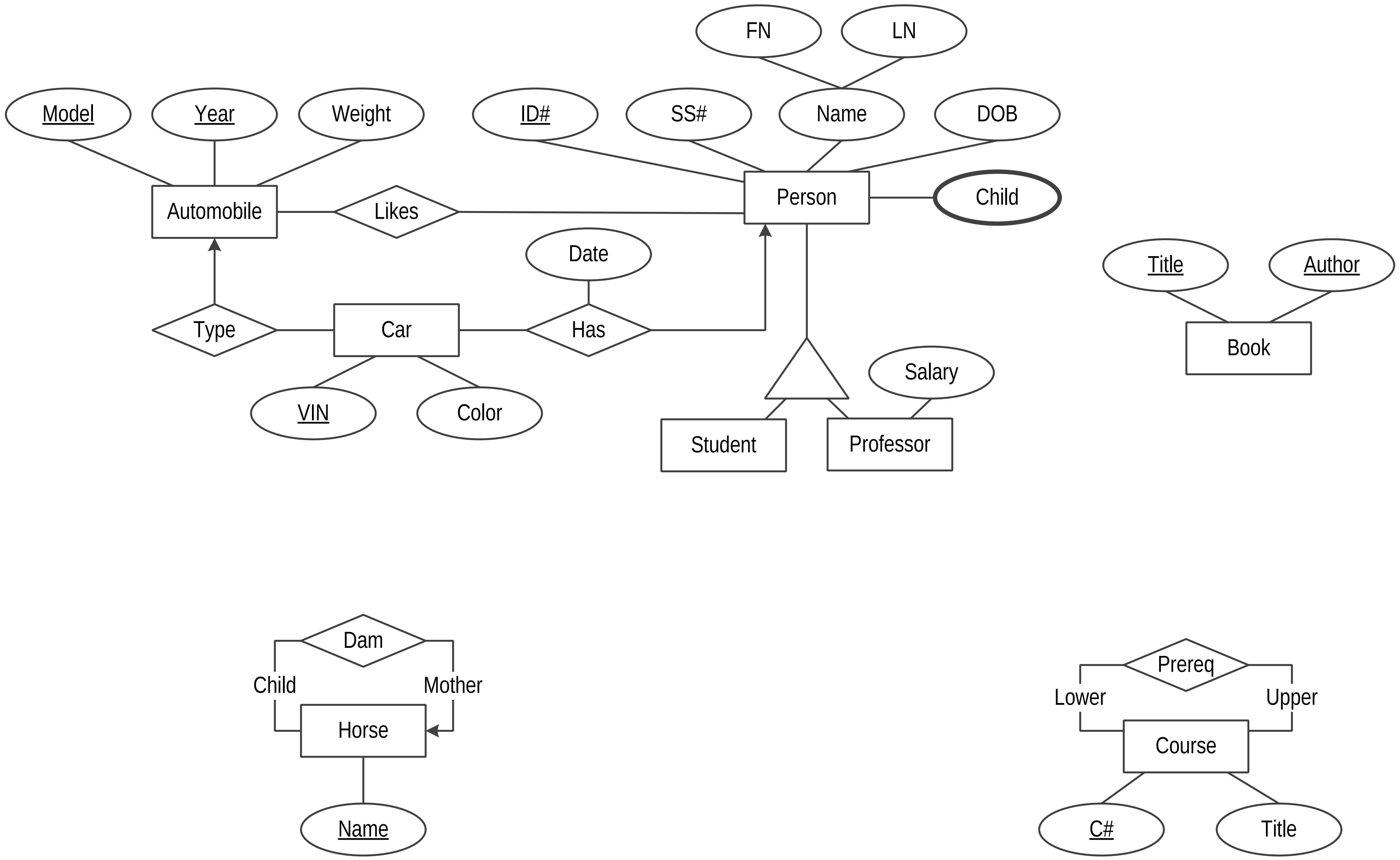
Comments

- Ultimately, we will store (most) relationships as tables
- So, comparing to our earlier example for Likes, Prereq instance could be

Prereq	Lower	Upper
	101	103
	101	104
	102	104
	104	105
	107	106
	107	108

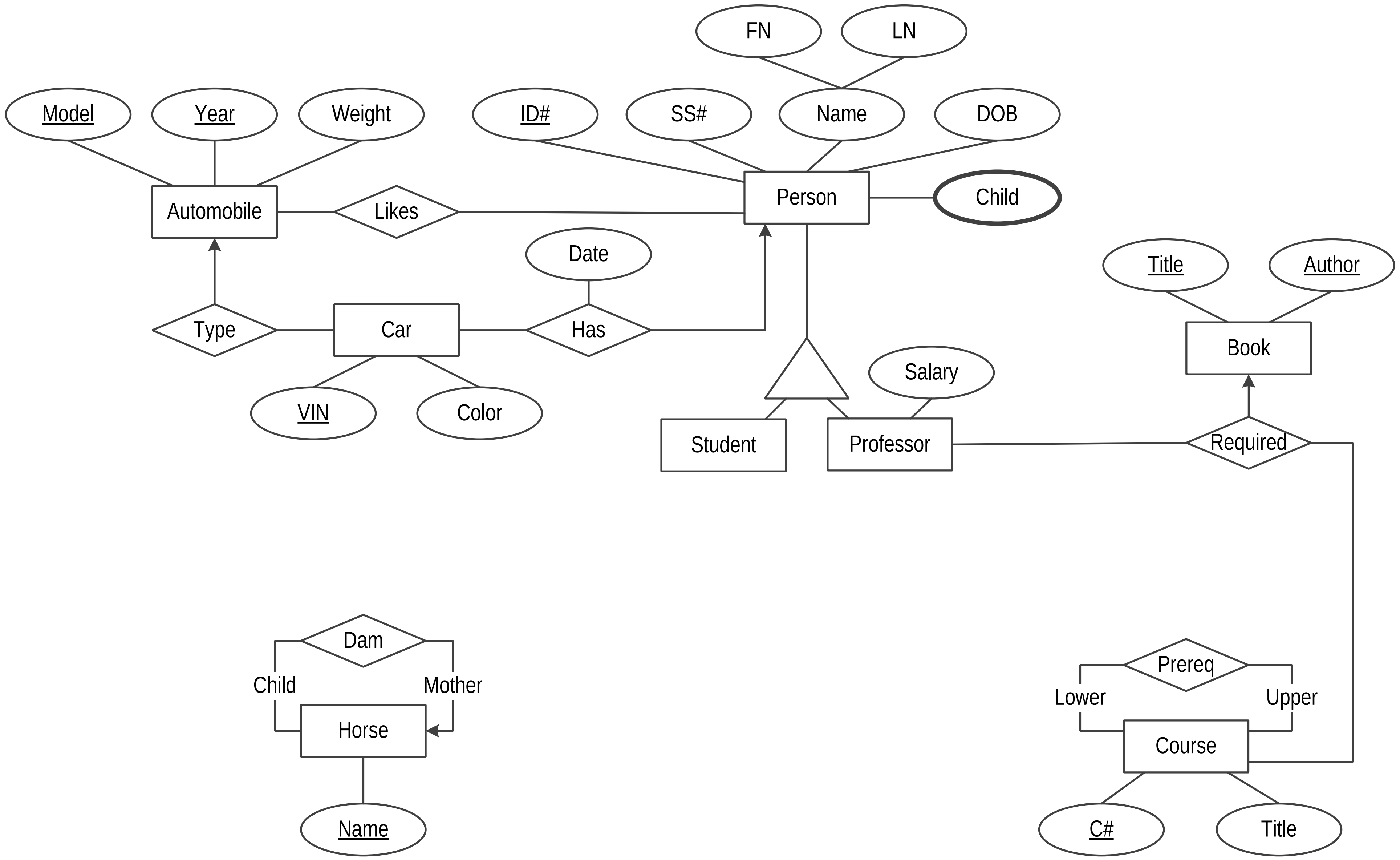
- We identify the “participating” entities using their primary keys, but rename the columns using roles
- In each row, the course under Lower is prerequisite for the course under Upper
- So, looking at the table we see that 101 is a prerequisite for 104 but 104 is not a prerequisite for 101

Our ER Diagram So Far



Annotations

Our ER Diagram So Far

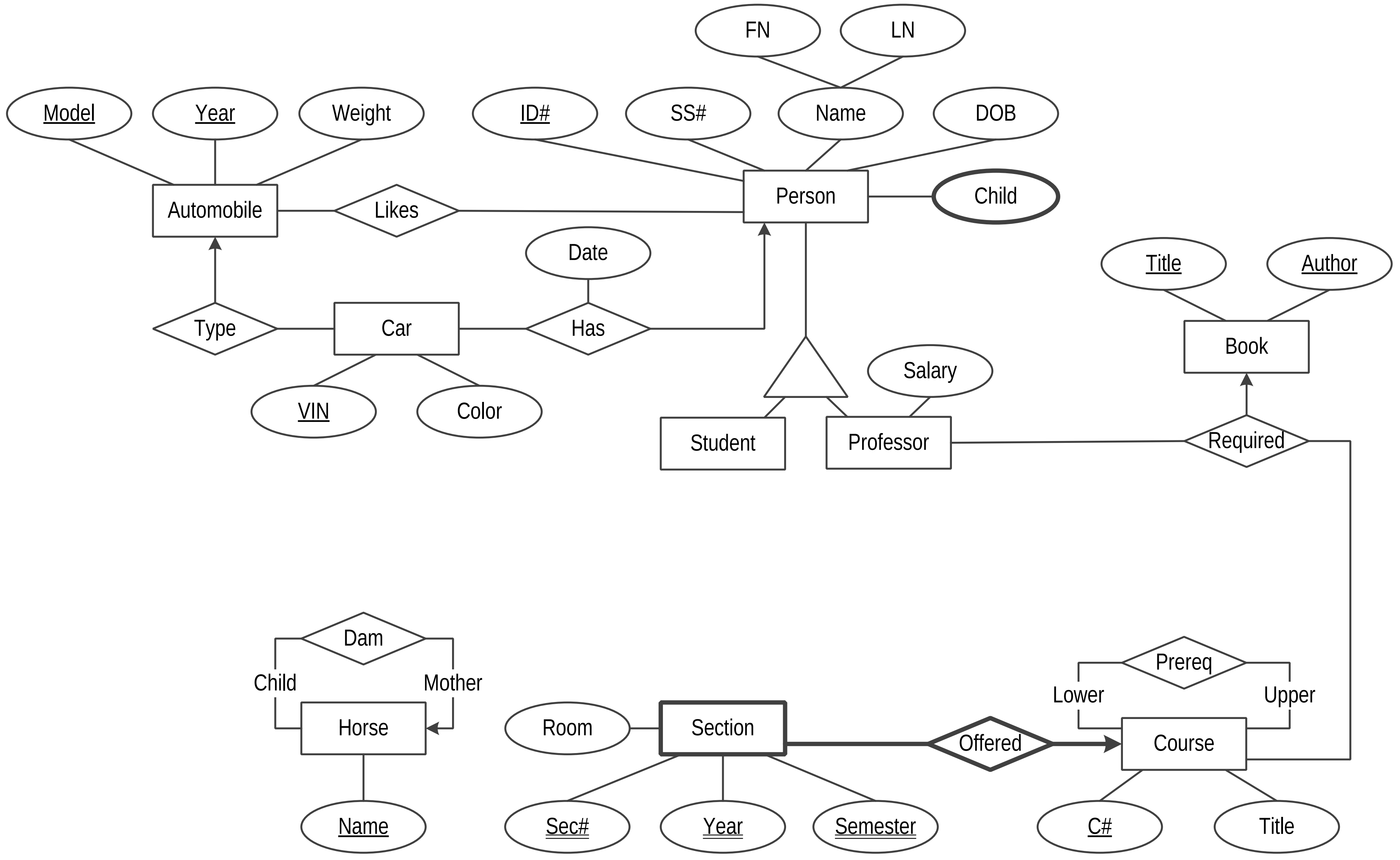


Annotations

Comments

- A Professor may specify at most one Book for a Course

Our ER Diagram So Far



Annotations

- Each Course has at least 1 Section related to it through Offered

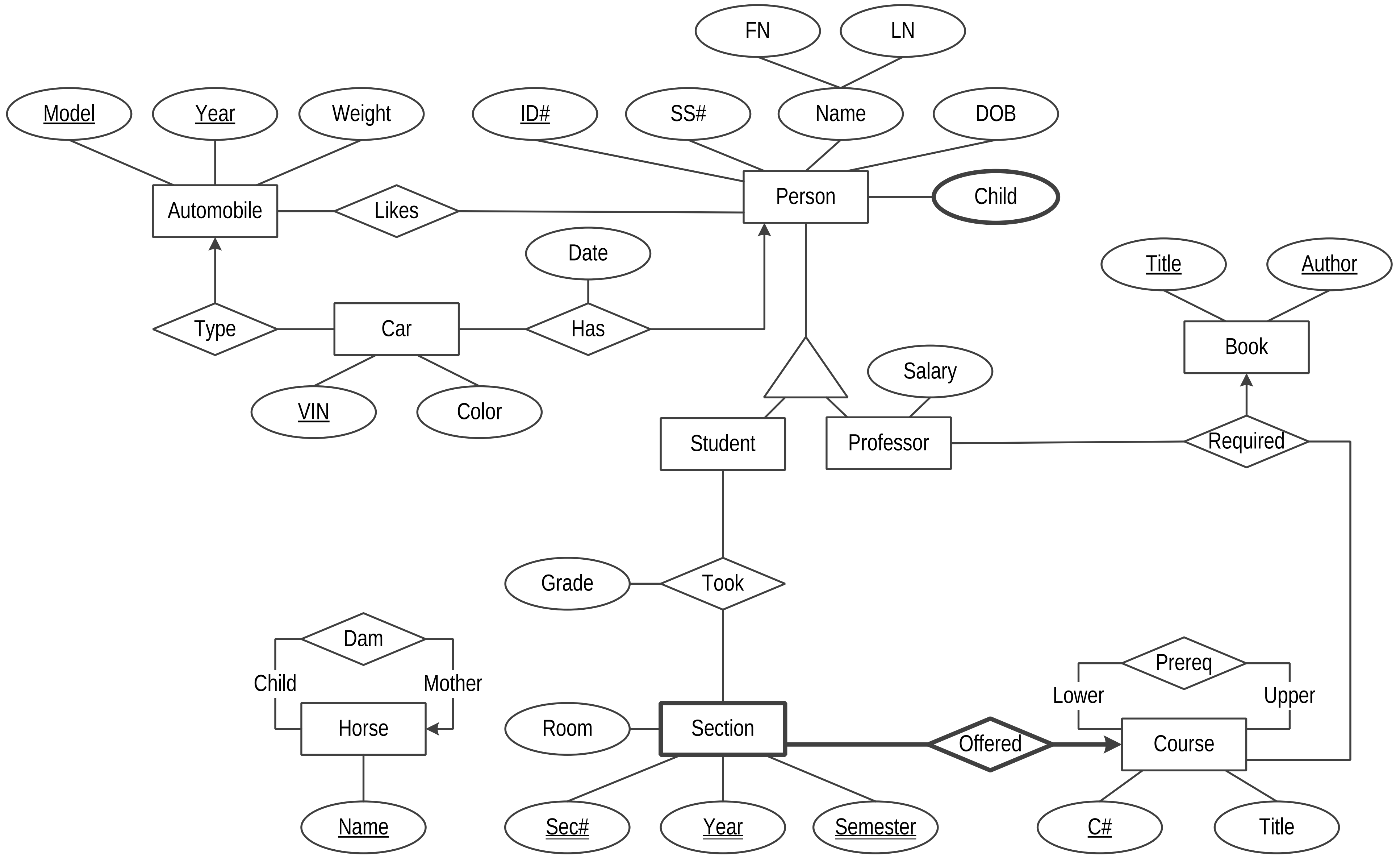
Comments

- Section is a weak entity set identified (partially) through a relationship Offered with Course
  - Partially, because it also has discriminant attributes
- Section and Offered need to be handled together

- One of our current sections (we have 3) can be identified at NYU by: CSCI-GA.2433, 2021, Fall, 001
- This is a primary key, so it seems to be a strong entity
- But it is ultimately a table in a relational database, so it is not completely relevant to the current unit

- In the ER diagram, this Section is identified by
  - Being related to Course identified by CSCI-GA.2433 (primary key of the strong entity Course)
  - Having attribute values of 2021, Fall, 001 (discriminants for Section)

Our ER Diagram So Far



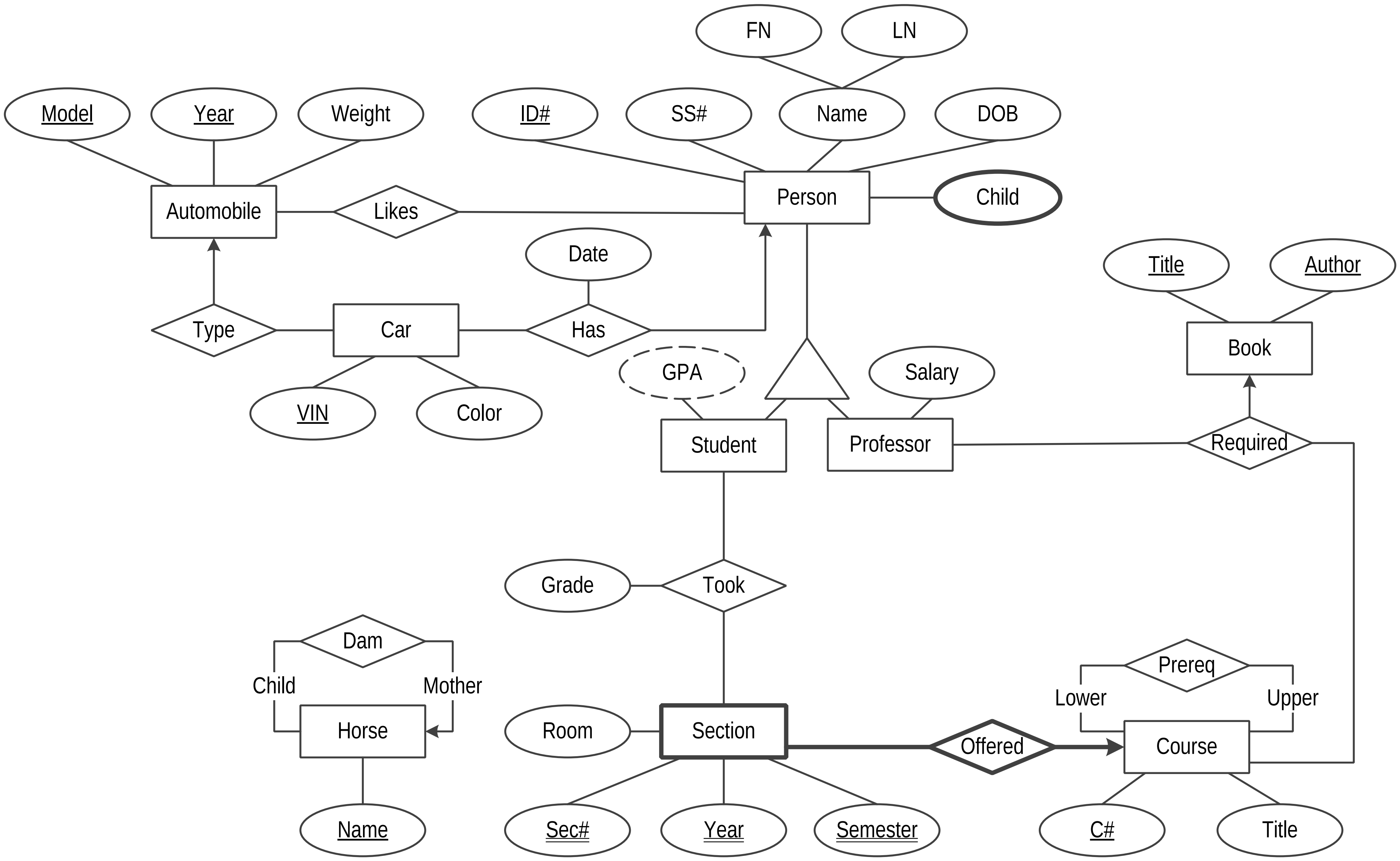
Annotations

- Grade is a number between 0.0 and 4.0
- A Section has between 3 and 50 Students

Comments

- Students register for sections and not for courses

Our ER Diagram So Far



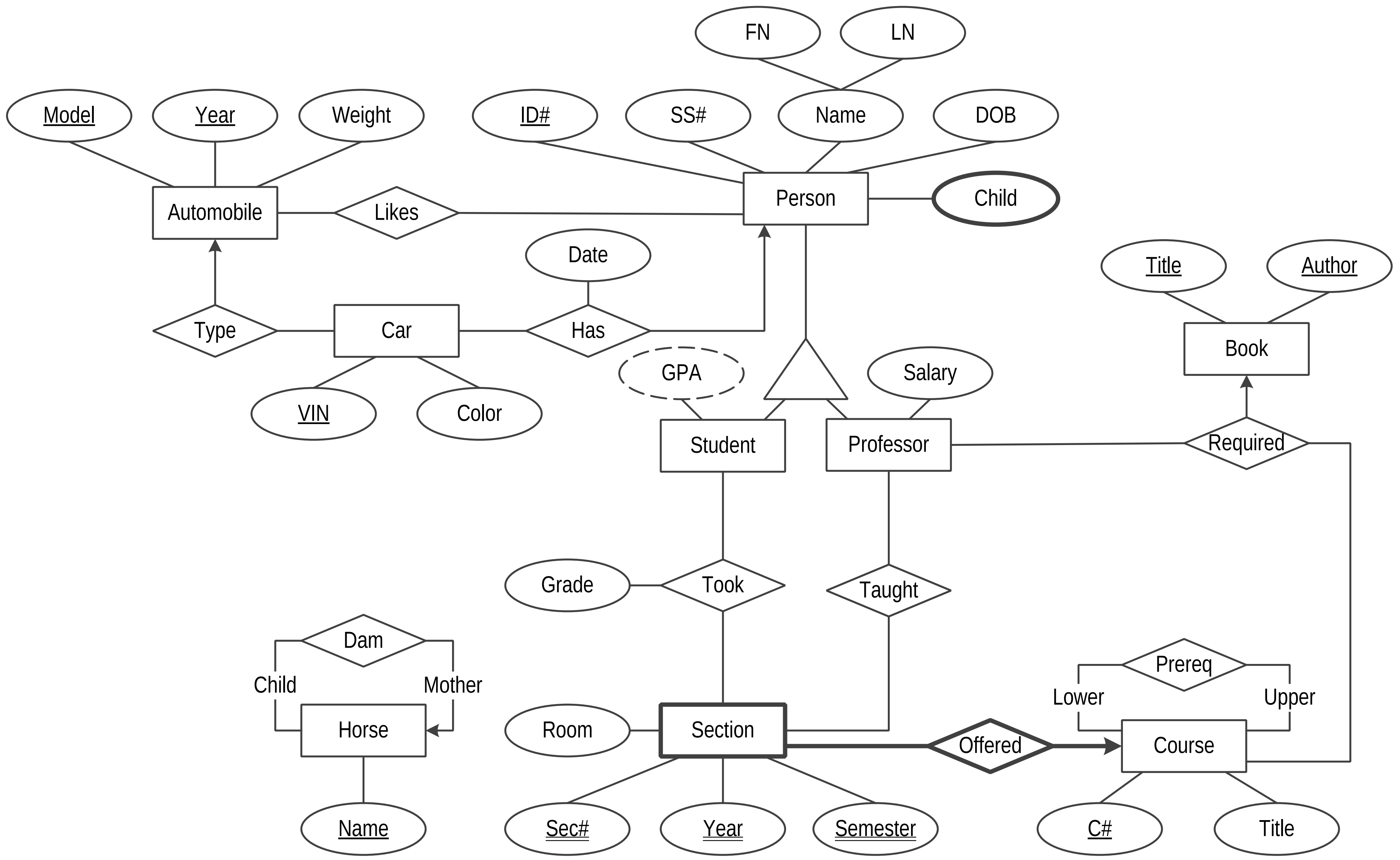
Annotations

- GPA: Computed Attribute for Student by adding all the known numeric Grades, dividing by the number of Sections that the Student Took and got a numeric Grade

Comments

- In our application, all the sections have the same number of points

Our ER Diagram So Far

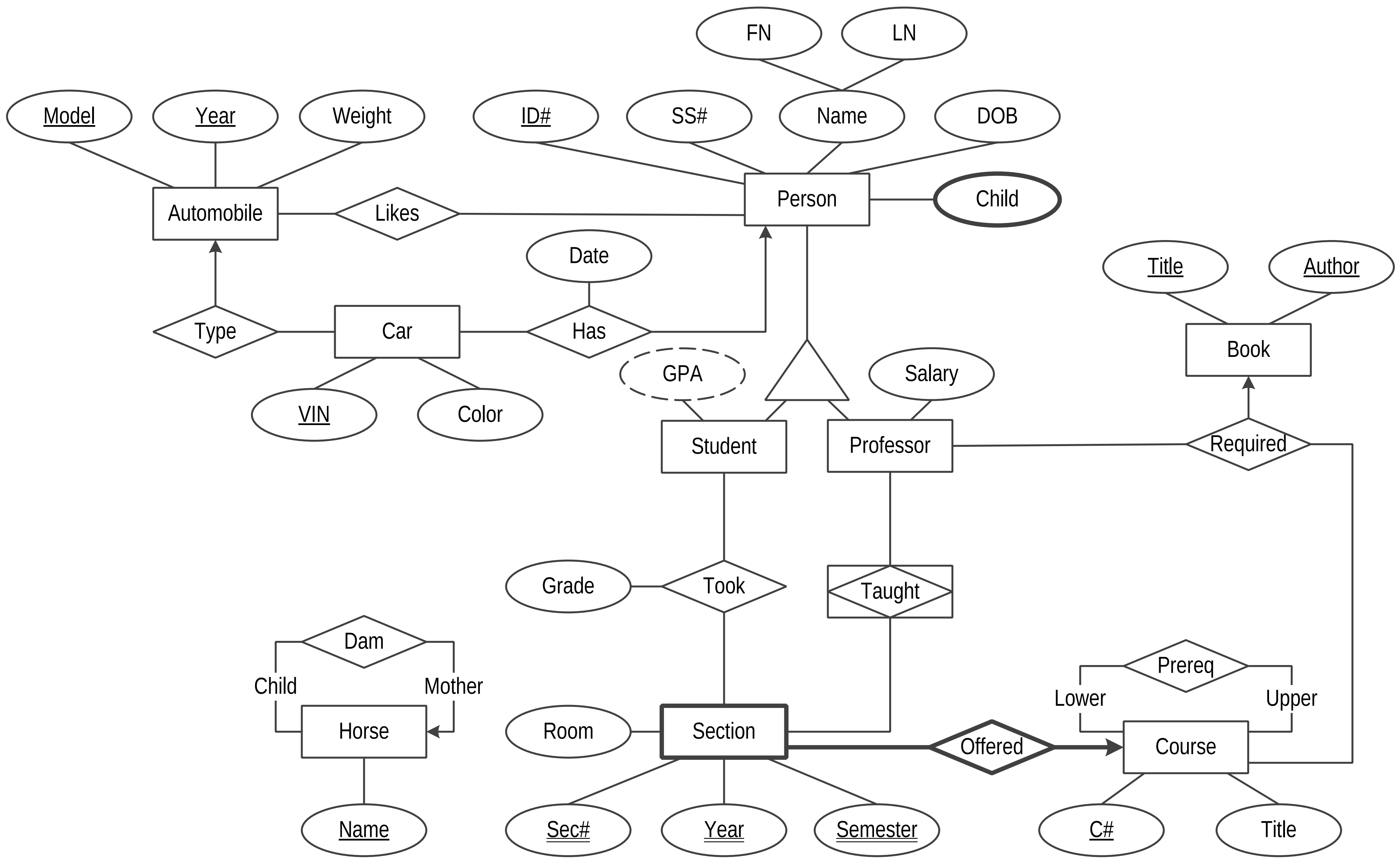


Annotations

Comments

- Professor teach Sections and not Courses

Our ER Diagram So Far

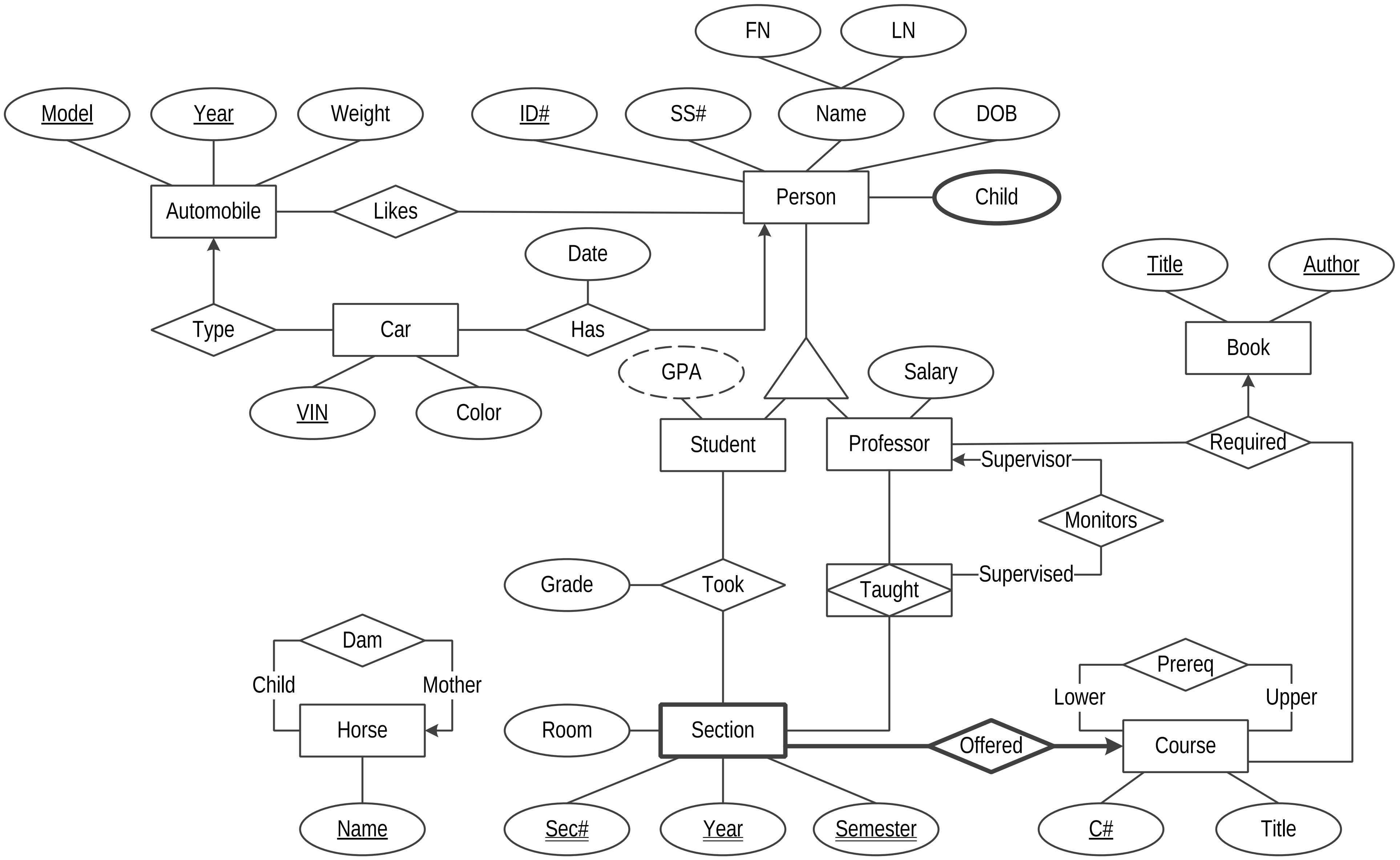


Annotations

Comments

- We want to think of Taught as an entity because we want it to participate in a relationship

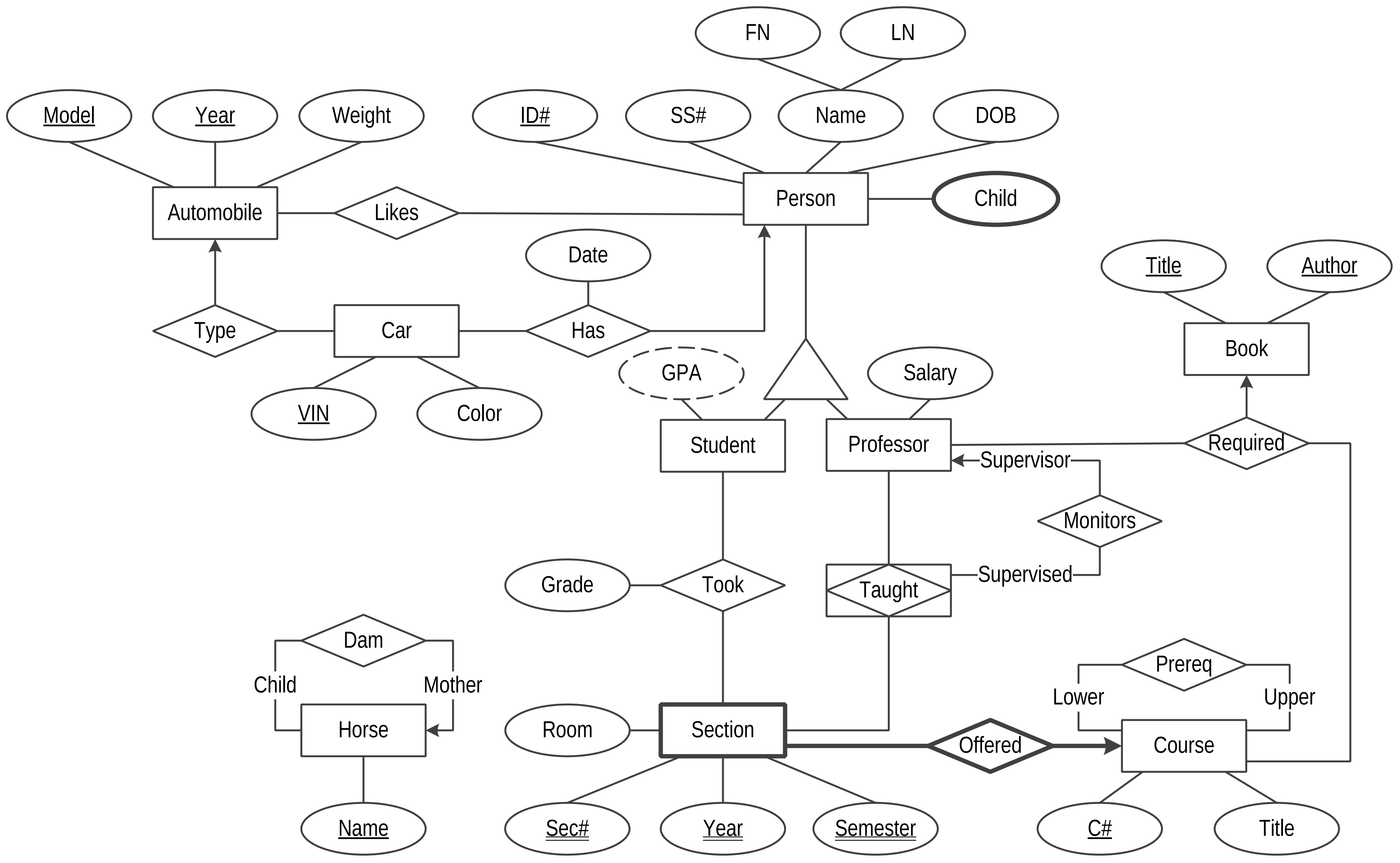
Our ER Diagram So Far



Annotations

- Supervisor and Supervised cannot be the same Professor

Our Final ER Diagram



ER Diagram: Annotations

- Collect all the annotations we have produced so far

- Annotations are needed so that whoever converts an ER diagram + annotations into a relational schema + annotations can account for all the relevant constraints imposed on the application

Annotations For The ER Diagram

- The graph describing the Dam relationship is a forest of rooted trees
- LN, SS#, DOB in Person are always known
- No two Persons can have the same SS#
- Color in Car is always known
- Weight in Automobile is always known
- Type is total
- Every Person Has at least 2 Cars
- Salary is always known
- Title in Course is always known
- Prereq is a DAG (Directed Acyclic Graph)
- Each Course has at least 1 Section related to it through Offered
- Grade is a number between 0.0 and 4.0

Annotations For The ER Diagram

- A Section has between 3 and 50 Students
- GPA: Computed Attribute for Student by adding all the known numeric Grades, dividing by the number of Sections that the Student Took in which the Grades are known
- Supervisor and Supervised cannot be the same Professor

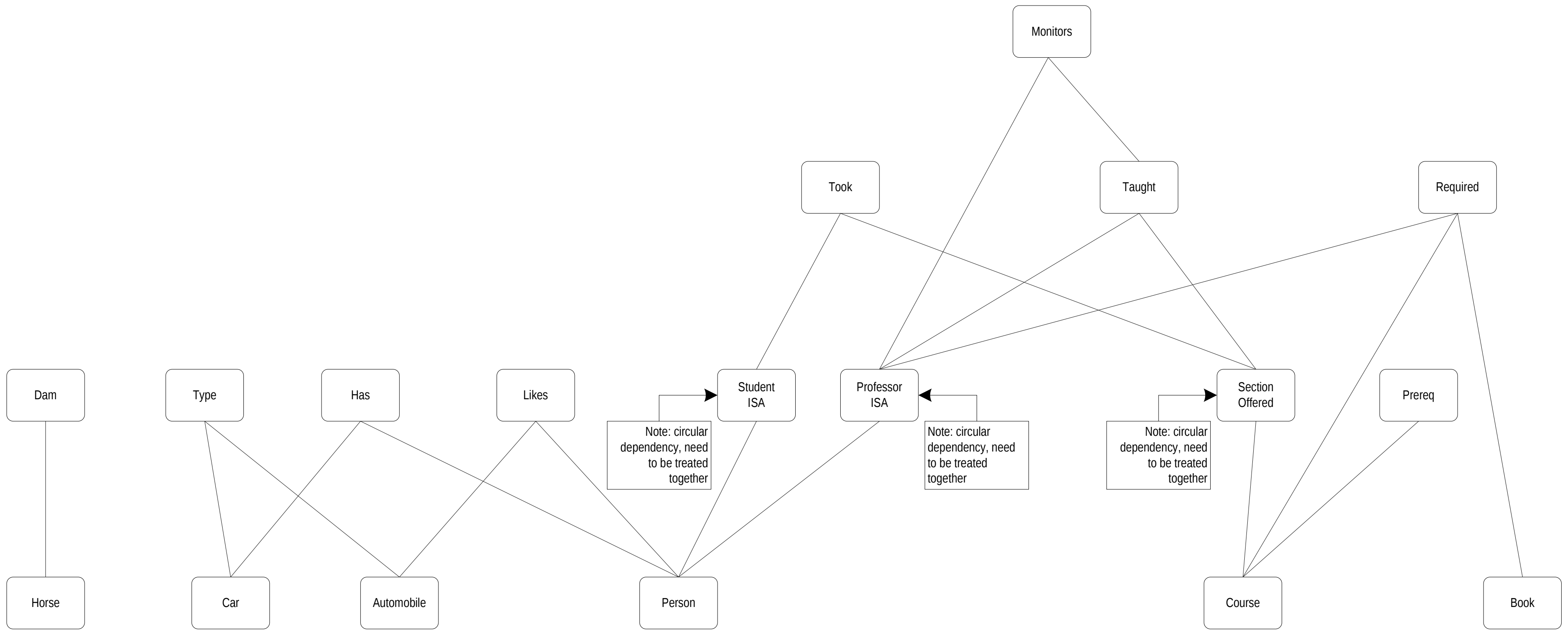
Comments

- Good design: few annotations

Hierarchy for Our ER Diagram

- There is a natural hierarchy for our ER diagram
- It shows us going from bottom to top how the ER diagram was constructed
- Section and Offered have to constructed together as there is a circular dependency between them
- Similar issue comes up when dealing with ISA

Hierarchy for Our ER Diagram


Monitors
Took
Taught
Required
Dam
Type
Has
Likes
Student ISA
Professor ISA
Section Offered
Prereq
Note: circular dependency, need to be treated together
Note: circular dependency, need to be treated together
Note: circular dependency, need to be treated together
Horse
Car
Automobile
Person
Course
Book

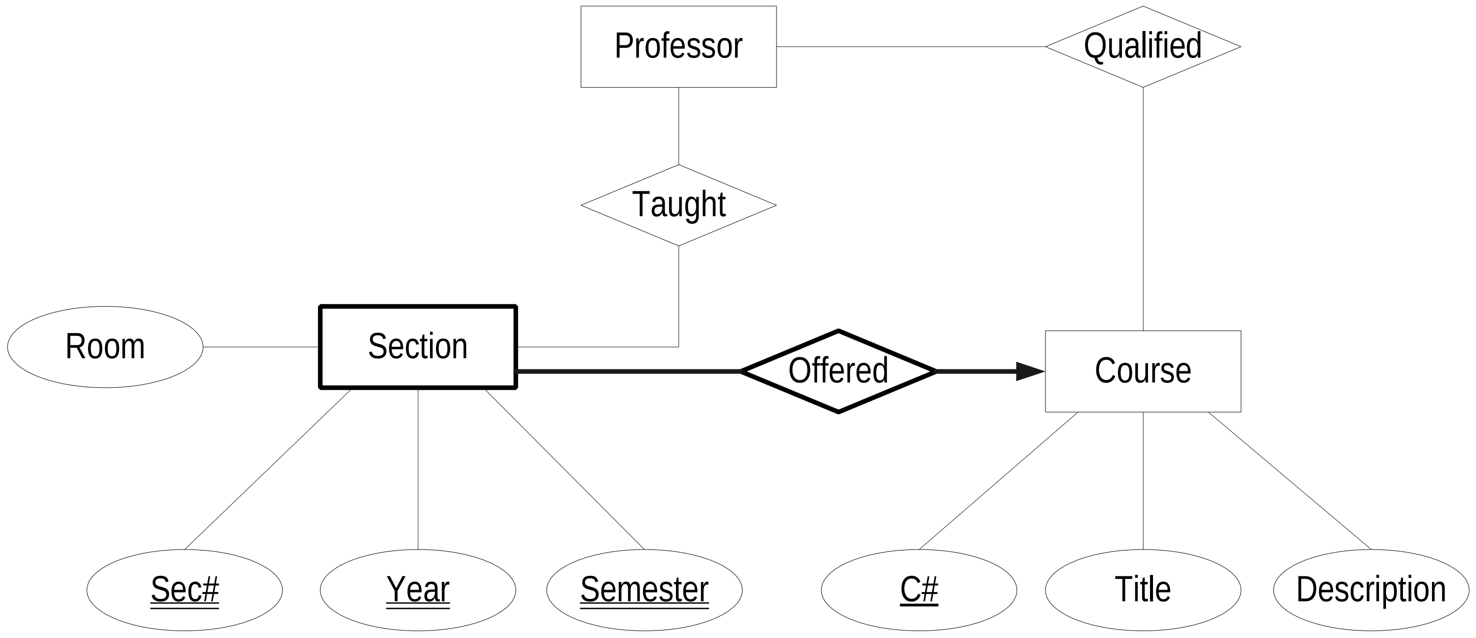
A Few Issues Of Importance

Some Additional Issues

- We will go over a few issues that could not be conveniently covered previously without distracting from an orderly development

Some Constraints Are Difficult to Specify

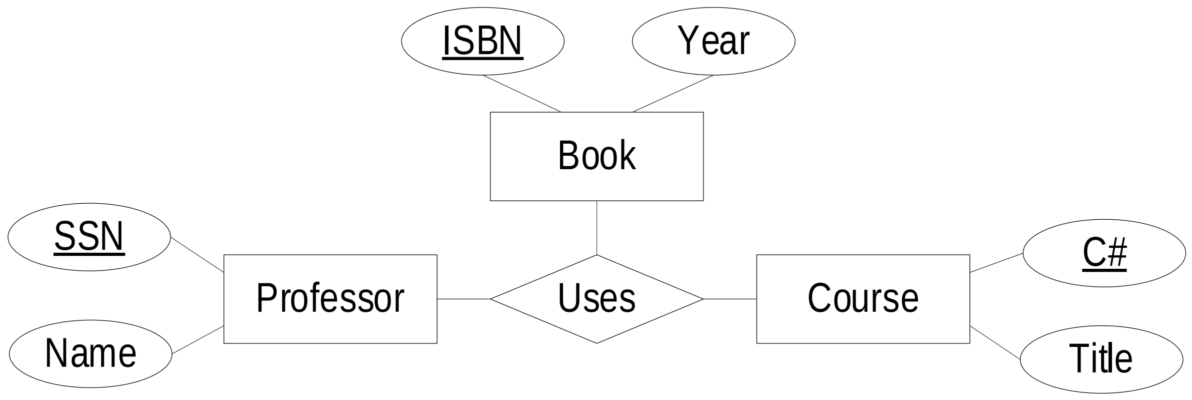
- Imagine that we also have relationship Qualified between Professor and Course specifying which professors are qualified to teach which courses
- We probably use words and not diagrams to say that only a qualified professor can teach a course



Many-To-One Non-Binary Relationships

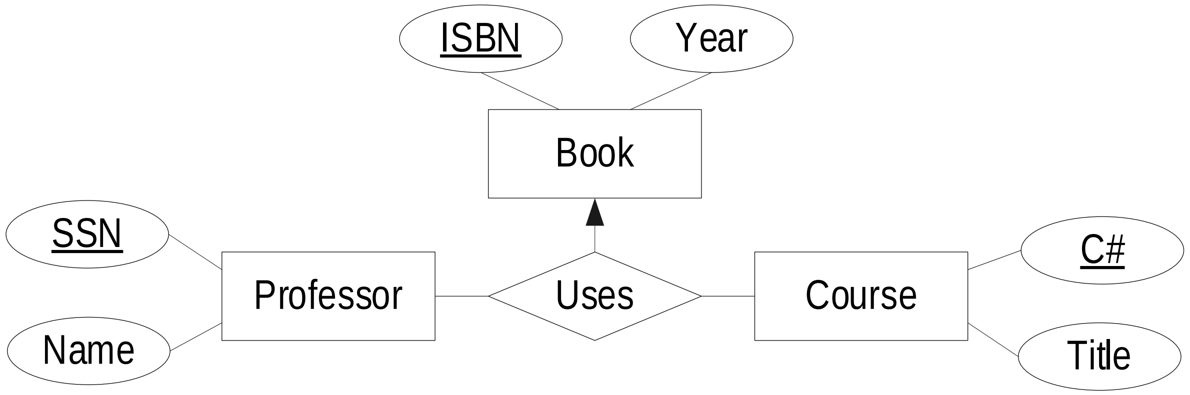
Elaboration on Many-To-One Relationships

- We will look at an example like a small part of our large ER diagram
- We want to specify that professors use books in courses
- Generally, there are no constraints and
  - A book can be used by any professor in any course as specified in an instance of the database



Elaboration on Many-To-One Relationships

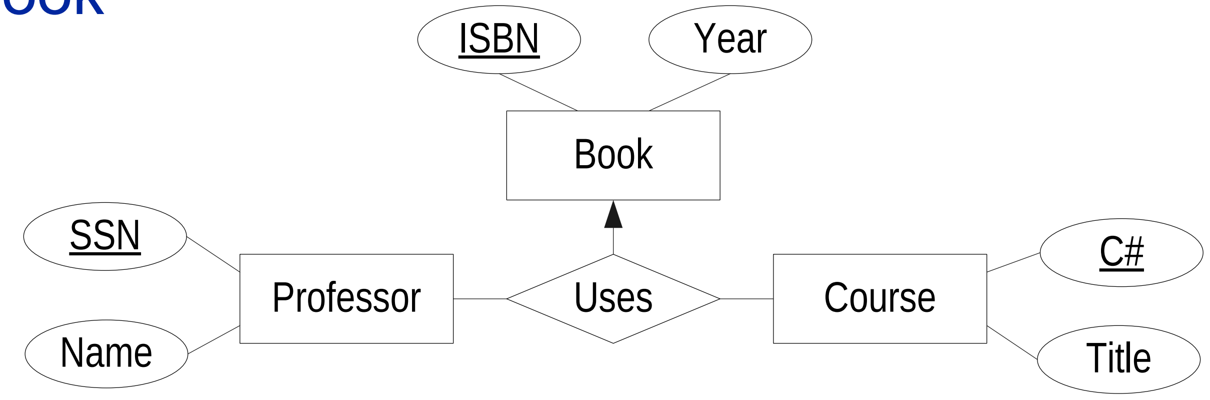
- We want to specify that professors use books in courses
- There is a constraint
  - A professor can use at most one book in a course
- Therefore, Book is a (partial) function of both Professor and Course
  - Given a pair (p,c), where p in Professor and c in Course there is at most one b in Book
  - As different Professors can Use different Books in the same Course, but each Professor can Use at most 1 Book in a Course



  - No need for annotation as the constraint follows from the arrow

Elaboration on Many-To-One Relationships

- We want to specify that professors use books in courses
- There is a constraint
  - At most 1, specific Book can be used in a Course, no matter who the Professor is
- Therefore, Book is a (partial) function of Course (only)
  - Given c in Course there is at most one b in Book
- ***We need to annotate the diagram below*** stating that in the ternary relationship (Professor, Course, Book) the many-to-one relationship (arrow) is between Course and Book, as otherwise we will think it is between (Professor, Course) and Book



- Class exercise: This is a bad design; what is better?

Key Ideas

Key Ideas

- ER diagrams
- Entity and Entity Set
- Attribute
  - Base
  - Derived
  - Simple
  - Composite
  - Singlevalued
  - Multivalued
- Key
- Candidate Key
- Primary Key
- UNIQUE

Key Ideas

- Relationship
- Binary relationship and its functionality
- Non-binary relationship
- Relationship with attributes
- Aggregation
- Strong and weak entities
- Discriminator for weak entities
- ISA
  - Disjoint
  - Overlapping
  - Total
  - Partial
  - Specialization
  - Generalization
  - Discriminator to specify the subsets

Key Ideas

- General cardinality constraints
- Using drawing software
- Case study of modeling
- Elaboration on many-to-one relationships (partial functions)

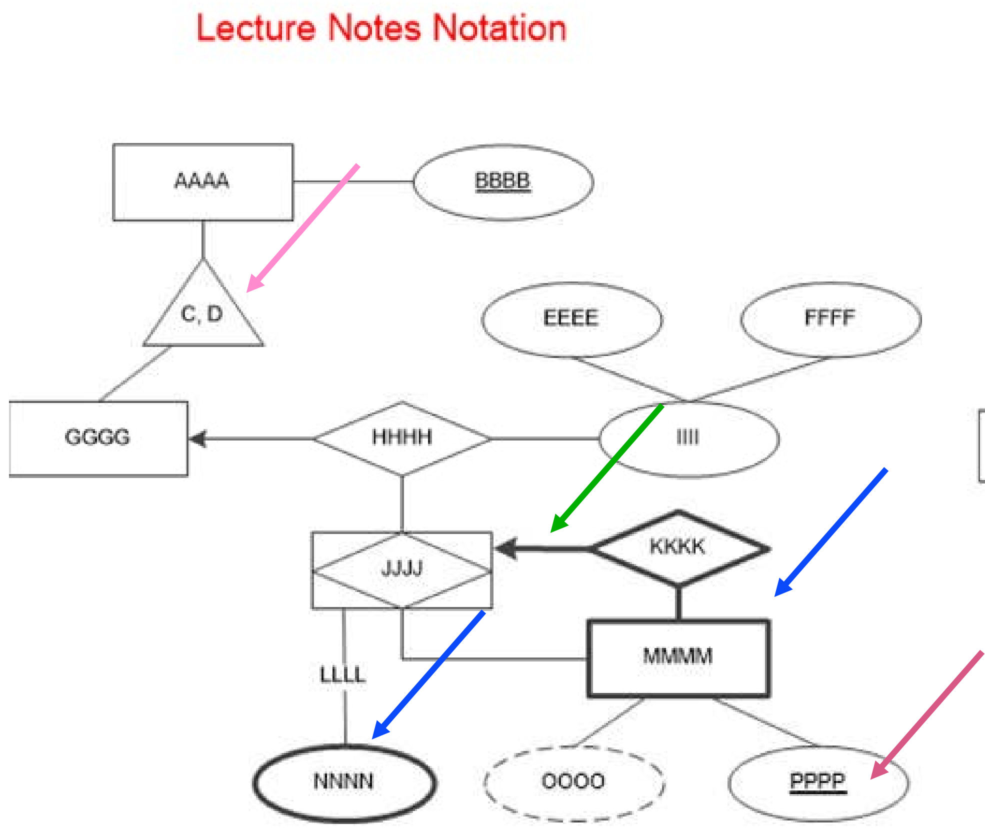
Some Points to Remember for Homework

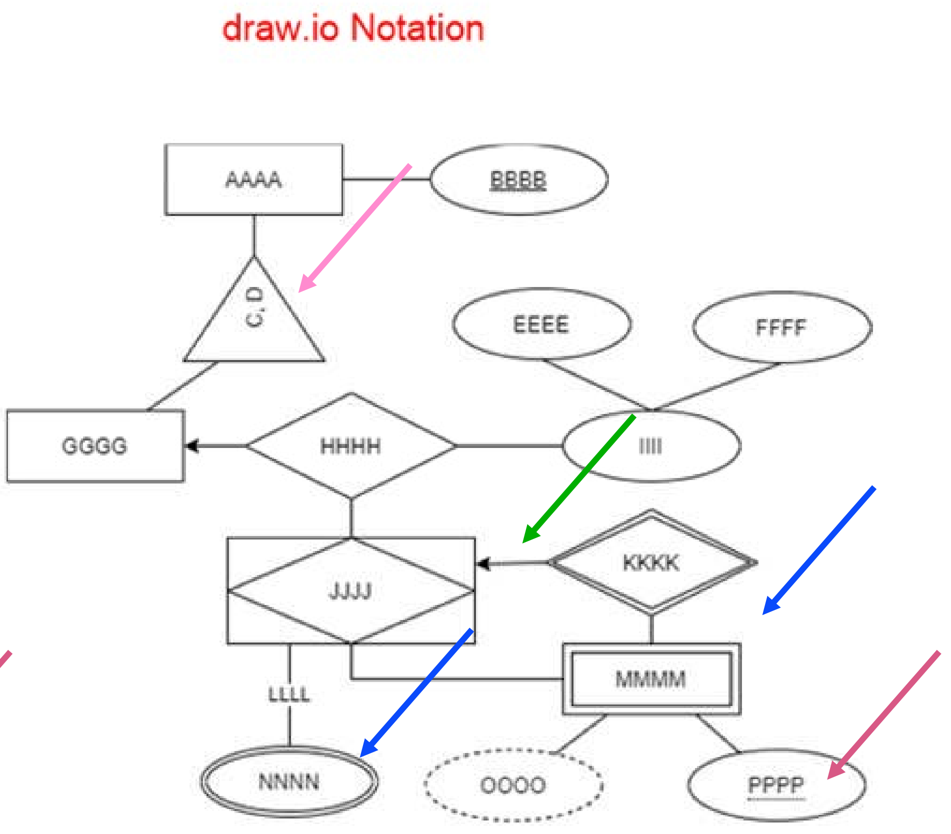
- I list a some of the points covered in the notes that are worth remembering while doing the homework
  - They are phrased informally
- If an entity/attribute does not have its own attributes and does not participate in relationships perhaps it should not be an entity but only an attribute
- If an entity/attribute has attributes but does not participate in relationships perhaps it should be a composite attribute
- A derived attribute should typically not be stored
- Do not specify hierarchies using explicit weak entities
- A relationship can have at most one arrow “coming out” of it (unless binary one-to-one)
  - This, by itself, indicates that the relationship is many-to-one from into a specified entity from all the other entities
  - Additional constraints on such a relationship, if any, may need to be specified by annotations

Remember for the Homework

- Specify in annotations all the constraints that cannot be specified by the diagram ***and only such constraints***

Notation: Lectures vs. draw.io





Differences in changing from Lecture Notes to draw.io

- Replace thick lines borders by double lines borders
- Replace thick lines by normal lines
- Replace double underline by dotted underline
- Replace horizontal text by vertical text in a triangle