Database Systems

Session 3 – Main Theme

Enterprise Data Modeling
Using The Entity/Relationship Model

Dr. Jean-Claude Franchitti

New York University
Computer Science Department
Courant Institute of Mathematical Sciences

Presentation material partially based on textbook slides
Fundamentals of Database Systems (6th Edition)
by Ramez Elmasri and Shamkant Navathe

Agenda

➡ 1 **Session Overview**

2 **Enterprise Data Modeling Using the ER Model**

3 **Using the Enhanced Entity-Relationship Model**

4 **Case Study**

5 **Summary and Conclusion**

Session Agenda

- Session Overview
- Enterprise Data Modeling Using the ER Model
- Using the Extended ER Model
- Case Study
- Summary & Conclusion

What is the class about?

- Course description and syllabus:
  - » http://www.nyu.edu/classes/jcf/CSCI-GA.2433-001
  - » http://cs.nyu.edu/courses/fall11/CSCI-GA.2433-001/
- Textbooks:
  - » ***Fundamentals of Database Systems (6th Edition)***
    Ramez Elmasri and Shamkant Navathe
    Addition Wesley
    ISBN-10: 0-1360-8620-9, ISBN-13: 978-0136086208 6th Edition (04/10)

Icons / Metaphors

Information

Common Realization

Knowledge/Competency Pattern

Governance

Alignment

Solution Approach

Agenda

1 Session Overview

➡ 2 Enterprise Data Modeling Using the ER Model

3 Using the Enhanced Entity-Relationship Model

4 Case Study

5 Summary and Conclusion

Agenda

- Using High-Level Conceptual Data Models for Database Design
- A Sample Database Application
- Entity Types, Entity Sets, Attributes, and Keys
- Relationship Types, Relationship Sets, Roles, and Structural Constraints
- Weak Entity Types
- Refining the ER Design for the COMPANY Database
- ER Diagrams, Naming Conventions, and Design Issues
- Example of Other Notation: UML Class Diagrams
- Relationship Types of Degree Higher than Two

Data Modeling Using the Entity-Relationship (ER) Model

- Entity-Relationship (ER) model
  - Popular high-level conceptual data model
- ER diagrams
  - Diagrammatic notation associated with the ER model
- Unified Modeling Language (UML)

Using High-Level Conceptual Data Models for Database Design (1/2)

- Requirements collection and analysis
  - Database designers interview prospective database users to understand and document data requirements
  - Result: data requirements
  - Functional requirements of the application
- Conceptual schema
  - Conceptual design
  - Description of data requirements
  - Includes detailed descriptions of the entity types, relationships, and constraints
  - Transformed from high-level data model into implementation data model

Using High-Level Conceptual Data Models for Database Design (2/2)

- Logical design or data model mapping
  - Result is a database schema in implementation data model of DBMS
- Physical design phase
  - Internal storage structures, file organizations, indexes, access paths, and physical design parameters for the database files specified

Purpose Of ER Model And Basic Concepts (1/2)

- ***Entity/relationship (ER) model*** provides a common, informal, and convenient method for communication between application end users (customers) and the Database Administrator to model the information's structure
- 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, frequently employs ***ER diagrams***, which are pictorial descriptions to visualize information's structure
- ER models are both surprisingly both ***simple*** and ***powerful***

Purpose Of ER Model And Basic Concepts (2/2)

- There are three basic concepts appearing in the original ER model, which has since been extended
  - We will present the model from more simple to more complex concepts, with examples on the way
- We will go beyond the original ER model, and cover most of ***Enhanced ER*** model
- 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 in the context of the Visio tool
  - 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

A Sample Database Application

- COMPANY
  - Employees, departments, and projects
  - Company is organized into departments
  - Department controls a number of projects
  - Employee: store each employee’s name, Social Security number, address, salary, sex (gender), and birth date
  - Keep track of the dependents of each employee

A Sample ER Diagram for the COMPANY Database

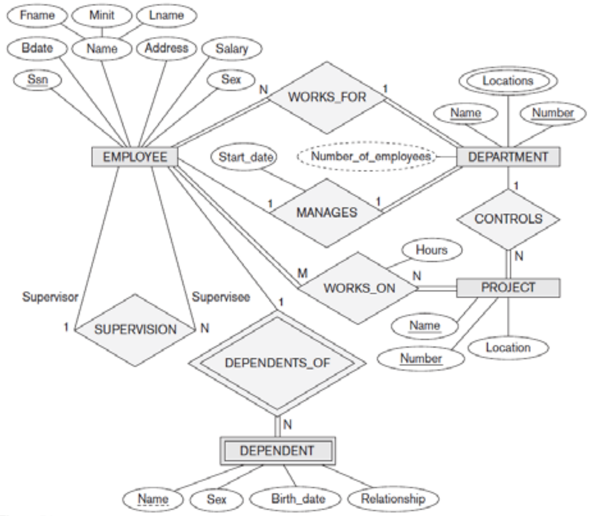


Figure 7.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter and is summarized in Figure 7.14.

Entity Types, Entity Sets, Attributes, and Keys

- ER model describes data as:
  - Entities
  - Relationships
  - Attributes

Entities and Attributes (1/2)

- Entity
  - Thing in real world with independent existence
- Attributes
  - Particular properties that describe entity
  - Types of attributes:
    - *Composite* versus *simple* (atomic) *attributes*
    - Single-valued versus multi-valued attributes
    - Stored versus derived attributes
    - NULL values
    - Complex attributes

Entities and Attributes (2/2)

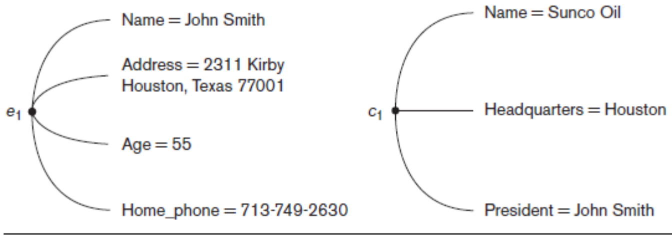


Figure 7.3
Two entities, EMPLOYEE e_1, and COMPANY c_1, and their attributes.

Entity Types, Entity Sets, Keys, and Value Sets (1/2)

- Entity type
  - Collection (or set) of entities that have the same attributes

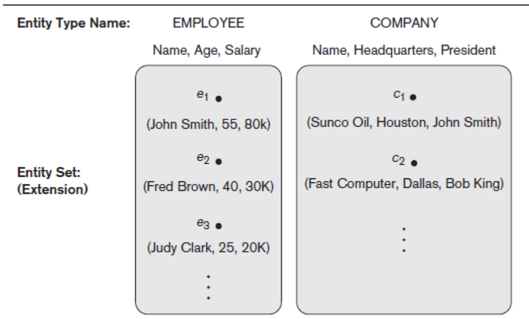
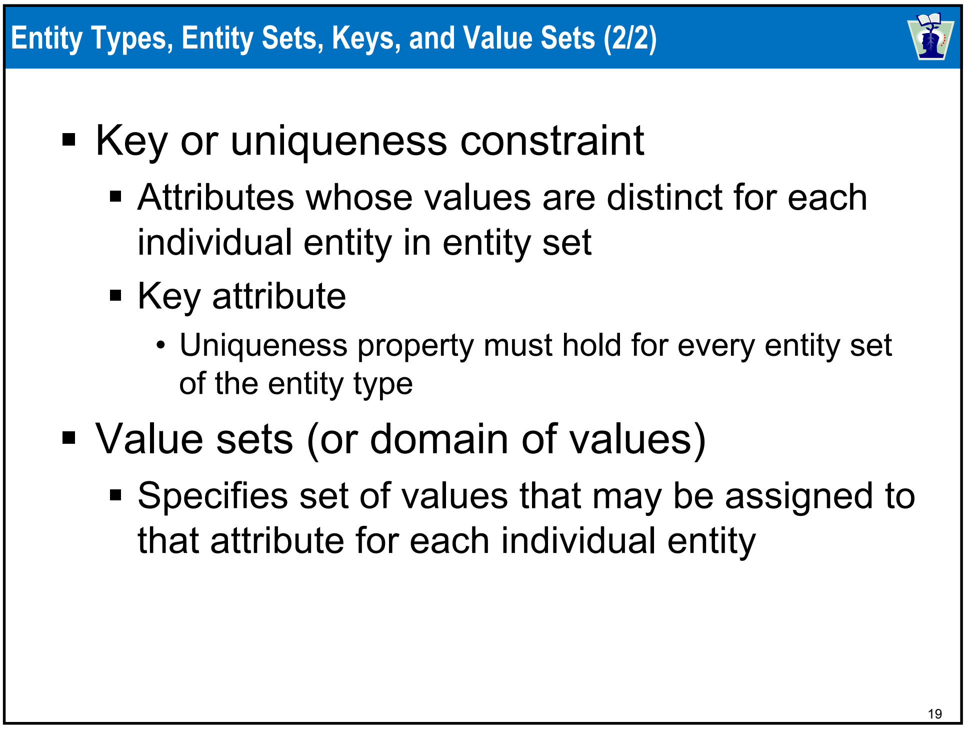
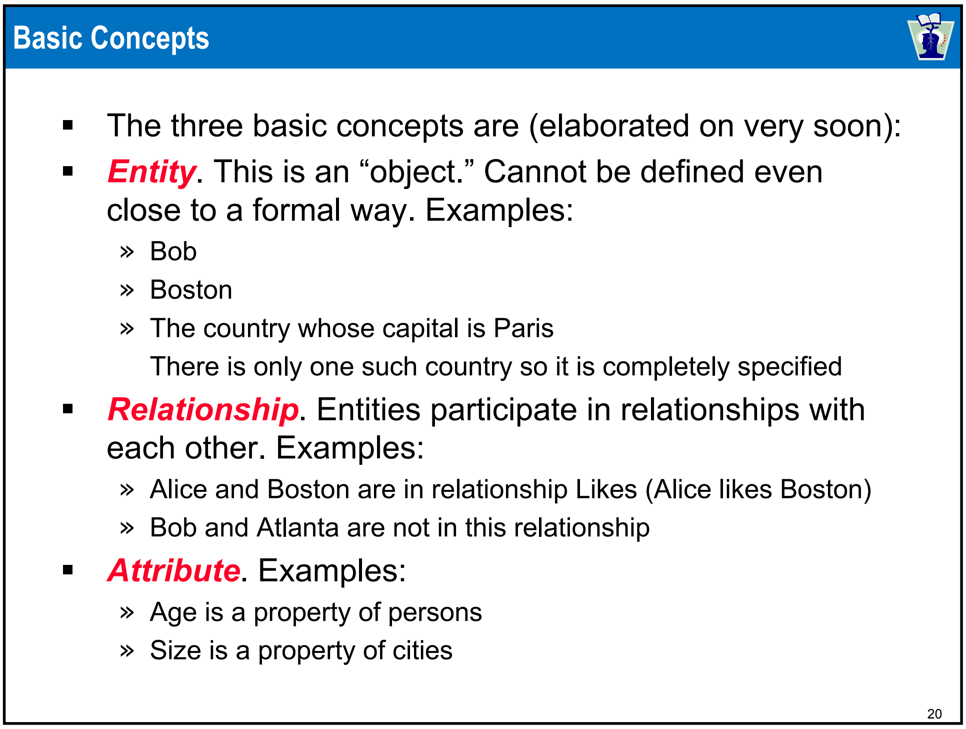


Figure 7.6
Two entity types, EMPLOYEE and COMPANY, and some member entities of each.



Entity Types, Entity Sets, Keys, and Value Sets (2/2)

- Key or uniqueness constraint
  - Attributes whose values are distinct for each individual entity in entity set
  - Key attribute
    - Uniqueness property must hold for every entity set of the entity type
- Value sets (or domain of values)
  - Specifies set of values that may be assigned to that attribute for each individual entity



Basic Concepts

- The three basic concepts are (elaborated on very soon):
- ***Entity***. This is an “object.” Cannot be defined even close to a formal way. Examples:
  - » Bob
  - » Boston
  - » The country whose capital is Paris
    There is only one such country so it is completely specified
- ***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***. Examples:
  - » Age is a property of persons
  - » Size is a property of cities

Entity And Entity Set (1/3)

- ***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 type is a little tricky sometimes
- Example. Do we partition people by sex or not?
  - » Sometimes makes sense (gave birth)

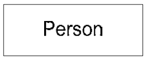
    This allows better enforcement of constraints. You could “automatically” make sure that only entities in the set of women, but not in the set of men can give birth
  - » Sometimes not (employment)

Entity And Entity Set (2/3)

- Example. When we say “the set of all Boeing airplanes,” is this
  - » The set of all models appearing 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 that are somehow related
- We will frequently use the term “entity” while actually referring to entity sets, unless this causes confusion

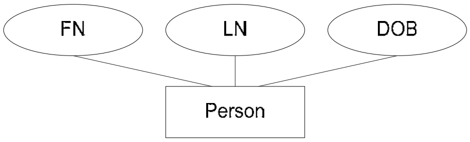
Entity And Entity Set (2/3)

- Pictorially, an entity set is denoted by a ***rectangle*** with its type written inside
- 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



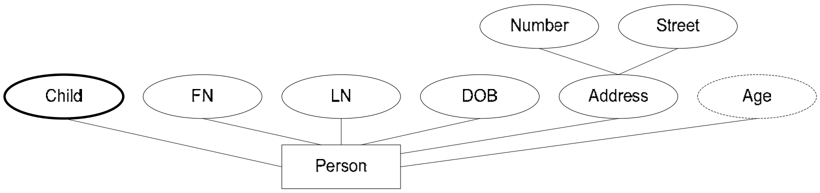
Attribute (1/3)

- An entity may have (and in general 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)
- All entities in an entity set have the same set of properties, though not generally 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.”



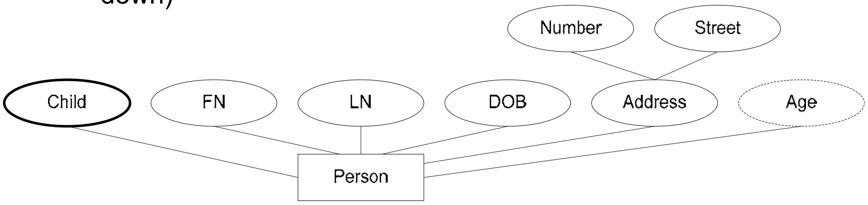
Attribute (2/3)

- Attributes can be
  - ***Base*** (such as DOB); or ***derived*** denoted by dashed ellipses (such as Age, derived from DOB and the current date)
  - ***Simple*** (such as DOB); or ***composite*** 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 unspecified in advance number of values denoted by thick-lined ellipses (such as Child; a person may have any number of children—we do not consider children as persons at this point, this means that they are not entities, just attributes of persons)



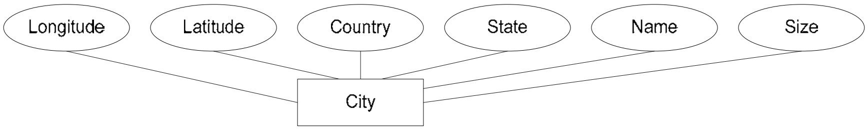
Attribute (3/3)

- To have a simple example of a person with attributes
  - Child: Bob
  - Child: Carol
  - FN: Alice
  - LN: Xie
  - DOB: 1980-01-01
  - Address.Number: 100
  - Address.Street: Mercer
  - Age: Current Date minus DOB specified in years (rounded down)



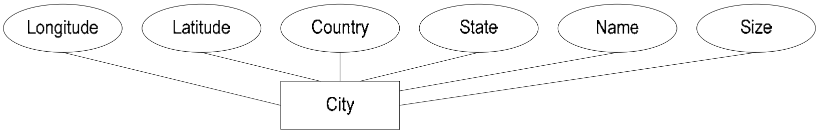
Keys (1/2)

- Most of the times, some subset (proper or not) of the attributes of an entity has the property that two different entities must differ on the values of these attributes
- This must hold for all conceivable entities in our database
- Such a set of attributes is called a ***superkey*** (“weak” superset of a key: either superset or equal)
- A minimal superkey is called a ***key*** (sometimes called a ***candidate key***).
  - » This means that no proper subset of it is itself a superkey



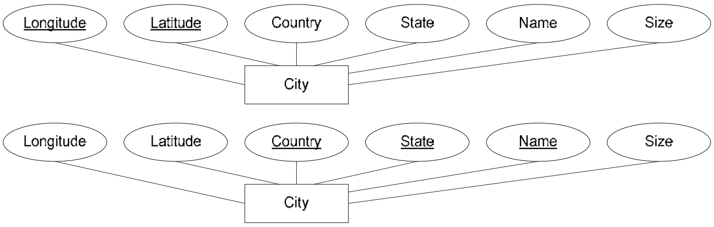
Keys (2/2)

- In our example:
  - » Longitude and Latitude (their values) identify (at most) one City, but only Longitude or only Latitude do not
  - » (Longitude, Latitude) form a superkey, which is also a key
  - » (Longitude, Latitude, Size, Name) form a superkey, which is not a key, because Size and Name are superflous
  - » (Country, State, Name) form another key (and also a superkey, as every key is a superkey)
- For simplicity, we assume that every country is divided into states and within a state the city name is unique



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 SQL the other keys, loosely speaking, are referred to using the keyword UNIQUE
- In the ER diagram, the attributes of the primary key are underlined
- So in our example, ***one*** of the two below:



Initial Conceptual Design of the COMPANY Database

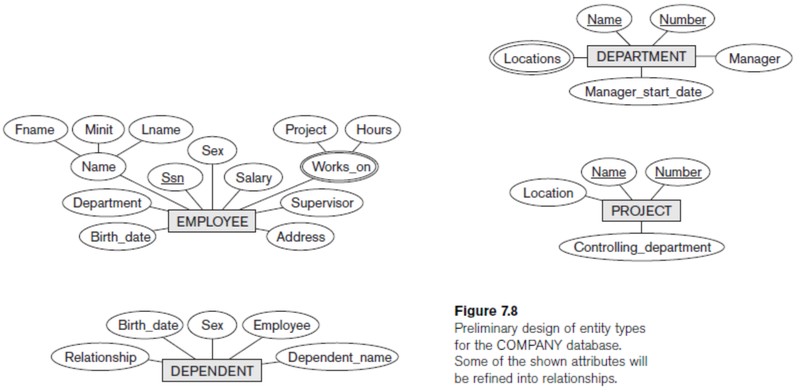


Figure 7.8
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

Relationship Types, Relationship Sets, Roles, and Structural Constraints

- Relationship
  - When an attribute of one entity type refers to another entity type
  - Represent references as relationships not attributes

Relationship Types, Sets, and Instances

- Relationship type *R* among *n* entity types E_1, E_2, ..., E_n
  - Defines a set of associations among entities from these entity types
- Relationship instances r_i
  - Each r_i associates n individual entities (e_1, e_2, ..., e_n)
  - Each entity e_j in r_i is a member of entity set E_j

Relationship Degree

- Degree of a relationship type
  - Number of participating entity types
  - Binary, ternary
- Relationships as attributes
  - Think of a binary relationship type in terms of attributes

Sample Ternary Relationship

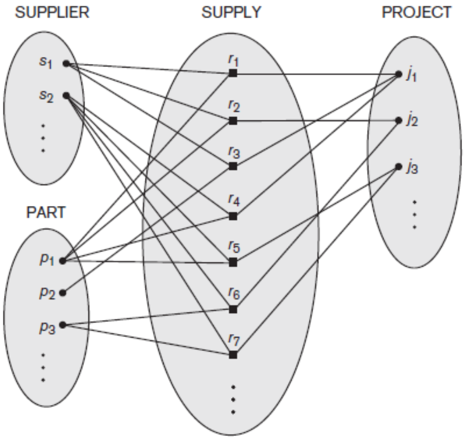


Figure 7.10
Some relationship instances in the SUPPLY ternary relationship set.

Role Names and Recursive Relationships

- Role names and recursive relationships
  - Role name signifies role that a participating entity plays in each relationship instance
- Recursive relationships
  - Same entity type participates more than once in a relationship type in different roles
  - Must specify role name

Sample Recursive Relationships

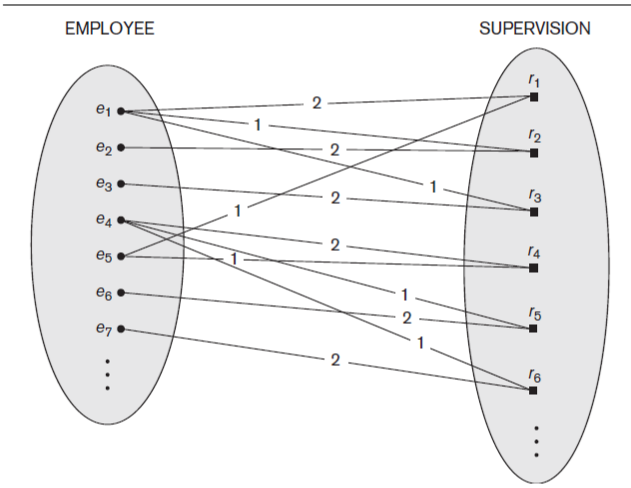


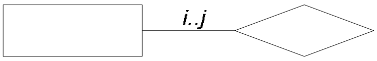
Figure 7.11
A recursive relationship SUPERVISION between EMPLOYEE in the *supervisor* role (1) and EMPLOYEE in the *subordinate* role (2).

Constraints on Binary Relationship Types

- Cardinality ratio for a binary relationship
  - Specifies maximum number of relationship instances that entity can participate in
- Participation constraint
  - Specifies whether existence of entity depends on its being related to another entity
  - Types: total and partial

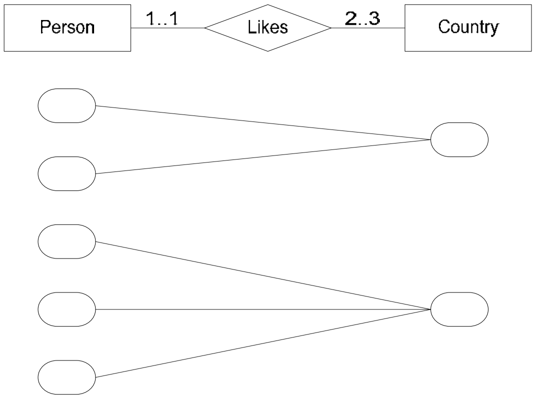
Cardinality Constraints (1/8)

- We can specify how many times each entity from some entity set can participate in some relationship, in every instance of the database
- In general we can say that
  - » This number is in the interval [*i*,*j*], $0 \le i \le j$, with *i* and *j* integers, denoted by i..j; or
  - » This number is at the interval [*i*, ∞), denoted by i..*
- 0..* means no constraint
- No constraint can also be indicated by not writing out anything



Cardinality Constraints (2/8)

- Every person likes exactly 1 country
- Every country is liked by 2 or 3 persons

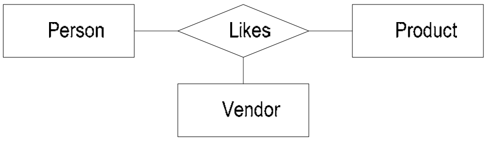


Cardinality Constraints (3/8)

- Returning to an old example without specifying which entities actually exist

Person
Vendor
Product

- We have a relationship: Likes
- A typical “participation” in a relationship would be that Joe, IBM, Computer participate in it



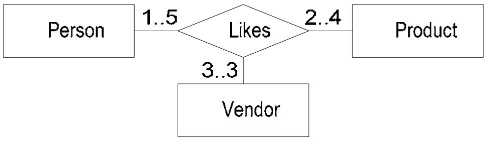
Cardinality Constraints (4/8)

Person	Vendor	Product

We want to specify cardinality constraints that every instance of the database needs to satisfy

- » Each person participates in between 1 and 5 relationships
- » Each vendor participates in between 3 and 3 (that is exactly 3) relationships
- » Each product participates in between 2 and 4 relationships

- This is indicated as follows:



Cardinality Constraints (5/8)

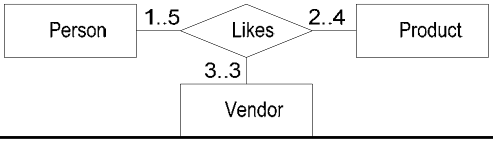
- A specific instance of the database

Person	Vendor	Product
Joe	IBM	computer
Tom	Apple	monitor
Max		

- If we have the following tuples in the relationship

Joe IBM computer
Tom Apple monitor
Max Apple computer
Max IBM monitor
Max IBM computer
Tom Apple computer

- Then, it is true that:

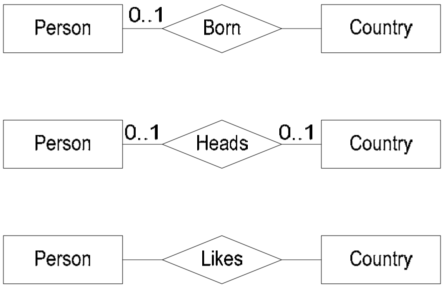


Cardinality Constraints (6/8)

- Let us confirm that our instance of Likes satisfies the required cardinality constraints
- Person: required between 1 and 5
  - Joe in 1
  - Tom in 2
  - Max in 3
- Product: required between 2 and 4
  - Monitor in 2
  - Computer in 4
- Vendor between 3 and 3
  - Apple in 3
  - IBM in 3
- Note that we do not have to have an entity for every possible permitted cardinality value
  - For example, there is no person participating in 4 or 5 tuples

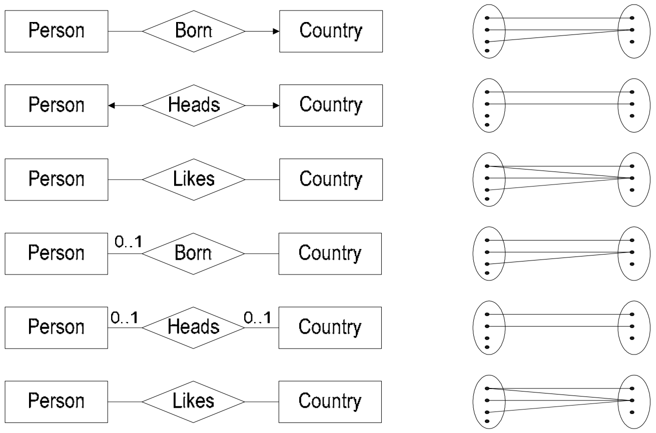
Cardinality Constraints (7/8)

- So we can also have, expressing exactly what we had before



Cardinality Constraints (8/8)

- Compare to previous notation



Attributes of Relationship Types

- Attributes of 1:1 or 1:N relationship types can be migrated to one entity type
- For a 1:N relationship type
  - Relationship attribute can be migrated only to entity type on N-side of relationship
- For M:N relationship types
  - Some attributes may be determined by combination of participating entities
  - Must be specified as relationship attributes

Weak Entity Types

- Do not have key attributes of their own
  - Identified by being related to specific entities from another entity type
- Identifying relationship
  - Relates a weak entity type to its owner
- Always has a total participation constraint

Introducing Relationship (1/2)

- Several entity sets (one or more) can participate in a ***relationship***
- Relationships are denoted by diamonds, to which the participating entities are "attached"
- A relationship could be binary, ternary, ….
- By convention, a capitalized verb in third person singular (e.g., Likes)
  - » We may not follow this convention, when inconvenient

Introducing Relationship (2/2)

- We will have some examples of relations
- We will use three entity sets, with entities in these sets listed below

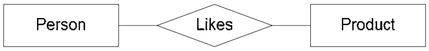
Person
Joe
Tom
Max
Mike
Kim

Vendor
IBM
Apple
Dell
HP

Product
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 (we omit article “a”):
  - Joe likes computer
  - Joe likes monitor
  - Tom likes computer
  - Max likes computer
- Note
  - Not every person has to Like a product
  - Not every product has to be Liked
  - A person can Like many products
  - A product can be Liked by many persons



More on Relationships (1/2)

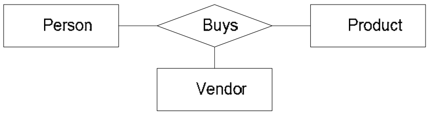
- Formally we say that R is a ***relationship*** among (not necessarily distinct) entity sets E_1, E_2, …, E_n if and only if R is a subset of $E_1 \times E_2 \times \ldots \times E_n$ (Cartesian product)
- In our example above:
  - » $n = 2$
  - » E_1 = {Joe, Tom, Max, Mike, Kim}
  - » E_2 = {computer,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

More on Relationships (2/2)

- 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

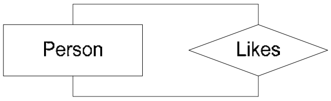
Ternary Relationship

- 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:
  - Joe buys computer from IBM
  - Joe buys computer from Dell
  - Tom buys computer from Dell
  - Tom buys monitor from Apple
  - Joe buys monitor from IBM
  - Max buys computer from IBM
  - Max buys monitor from Dell



Relationship With Non-Distinct Entity Sets (1/4)

- Let's look at Likes, listing all pairs of (*x*,*y*) where person *x* Likes person *y*
- Let us just state it informally
  - Joe likes Tom
  - Joe likes Max
  - Tom likes Max
  - Tom likes Mike
  - Tom likes Tom
  - Max likes Tom
- Note that pairs must be ordered to properly specify the relationship, Joe likes Tom, but Tom does not like Joe

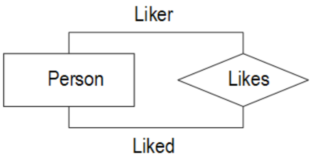


Relationship With Non-Distinct Entity Sets (2/4)

- Again:
  - Joe likes Tom
  - Joe likes Max
  - Tom likes Max
  - Tom likes Mike
  - Tom likes Tom
  - Max likes Tom
- 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 { (Joe,Tom), (Joe,Max), (Tom,Max), (Tom,Mike), (Tom,Tom), (Max,Tom) }
- Likes is an arbitrary directed graph in which persons serve as vertices and arcs specify who likes whom

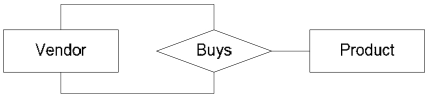
Relationship With Non-Distinct Entity Sets (3/4)

- 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 Joe likes Tom, then Joe is the “Liker” and Tom is the “Liked”
- Roles are explicitly listed in the diagram, but the semantics of they mean cannot be deduced from looking at the diagram only



Relationship With Non-Distinct Entity Sets (4/4)

- 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



Refining the ER Design for the COMPANY Database

- Change attributes that represent relationships into relationship types
- Determine cardinality ratio and participation constraint of each relationship type

ER Diagrams, Naming Conventions, and Design Issues

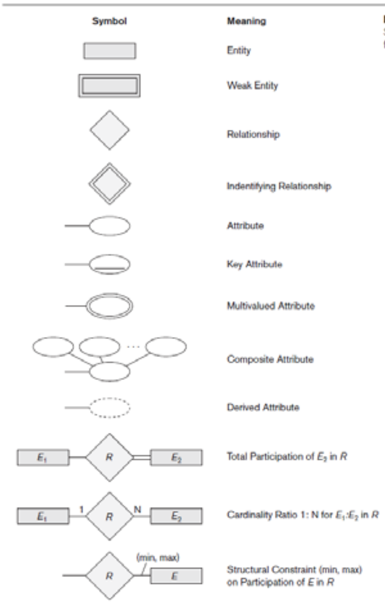
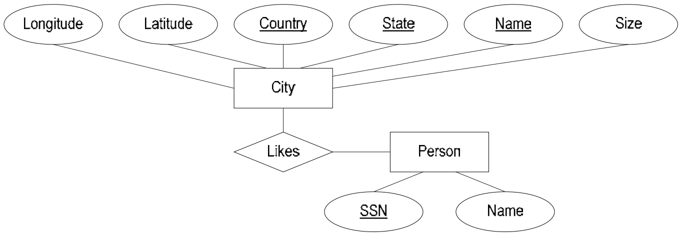

Symbol
Meaning
Entity
Weak Entity
Relationship
Indentifying Relationship
Attribute
Key Attribute
Multivalued Attribute
Composite Attribute
Derived Attribute
E_1
R
E_2
Total Participation of E_2 in R
E_1
1
R
N
E_2
Cardinality Ratio 1: N for E_1:E_2 in R
(min, max)
R
E
Structural Constraint (min, max) on Participation of E in R

Figure 7.14
Summary of the notation for 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):



Proper Naming of Schema Constructs

- Choose names that convey meanings attached to different constructs in schema
- Nouns give rise to entity type names
- Verbs indicate names of relationship types
- Choose binary relationship names to make ER diagram readable from left to right and from top to bottom

Design Choices for ER Conceptual Design

- Model concept first as an attribute
  - Refined into a relationship if attribute is a reference to another entity type
- Attribute that exists in several entity types may be elevated to an independent entity type
  - Can also be applied in the inverse

Alternative Notations for ER Diagrams

- Specify structural constraints on relationships
  - Replaces cardinality ratio (1:1, 1:N, M:N) and single/double line notation for participation constraints
  - Associate a pair of integer numbers (min, max) with each participation of an entity type *E* in a relationship type *R*, where 0 ≤ min ≤ max and max ≥ 1

Sample Use of Alternative Notations for the COMPANY Schema

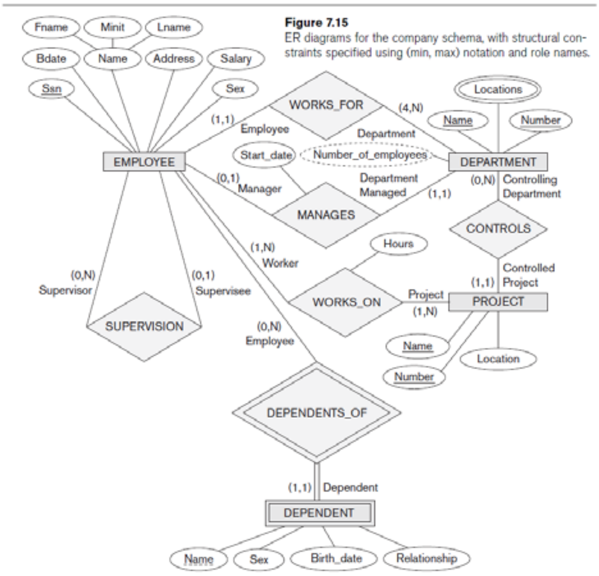


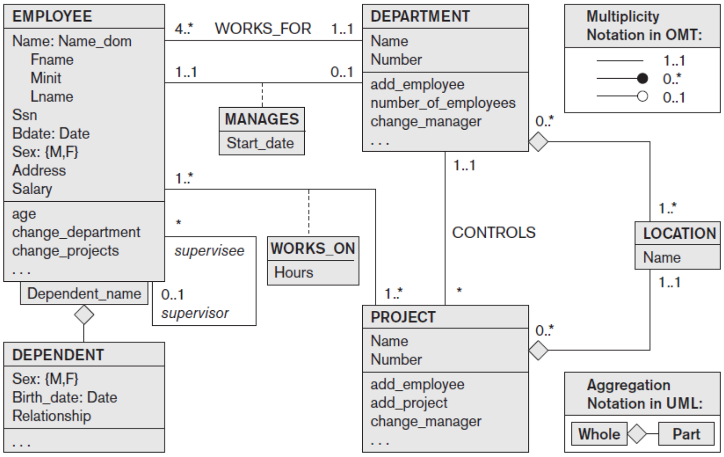
Figure 7.15
ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.

Example of Other Notation: UML Class Diagrams (1/3)

- UML methodology
  - Used extensively in software design
  - Many types of diagrams for various software design purposes
- UML class diagrams
  - Entity in ER corresponds to an object in UML

UML Class Diagram for the COMPANY Conceptual Schema

Figure 7.16
The COMPANY conceptual schema in UML class diagram notation.



Example of Other Notation: UML Class Diagrams (2/3)

- Class includes three sections:
  - Top section gives the class name
  - Middle section includes the attributes;
  - Last section includes operations that can be applied to individual objects
- Associations: relationship types
- Relationship instances: links
- Binary association
  - Represented as a line connecting participating classes
  - May optionally have a name
- Link attribute
  - Placed in a box connected to the association's line by a dashed line

Example of Other Notation: UML Class Diagrams (3/3)

- Multiplicities: min..max, asterisk (*) indicates no maximum limit on participation
- Types of relationships: association and aggregation
- Distinguish between unidirectional and bidirectional associations
- Model weak entities using qualified association

Relationship Types of Degree Higher than Two

- Degree of a relationship type
  - Number of participating entity types
- *Binary*
  - Relationship type of degree two
- *Ternary*
  - Relationship type of degree three

Choosing between Binary and Ternary (or Higher-Degree) Relationships

- Some database design tools permit only binary relationships
  - Ternary relationship must be represented as a weak entity type
  - No partial key and three identifying relationships
- Represent ternary relationship as a regular entity type
  - By introducing an artificial or surrogate key

Sample Ternary Relationships

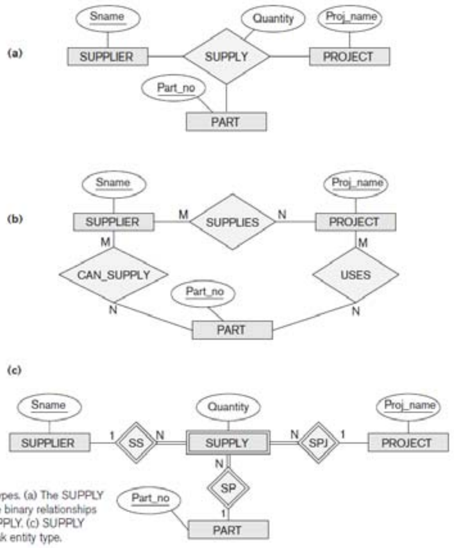


Figure 7.17
Ternary relationship types. (a) The SUPPLY relationship. (b) Three binary relationships not equivalent to SUPPLY. (c) SUPPLY represented as a weak entity type.

Constraints on Ternary (or Higher-Degree) Relationships

- Notations for specifying structural constraints on *n*-ary relationships
  - Should both be used if it is important to fully specify structural constraints

Further Refinements To The ER Model

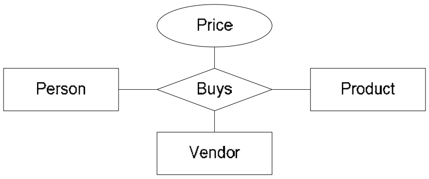
- There are further refinements to the model and associated diagrams that will be presented piecemeal in the following slides
- The previous modeling concepts and the ones that follow are needed for producing a data base design that models a particular application well
- The comprehensive case study provided at the end of this presentation puts it all together

Relationship With Attributes (1/2)

- 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
  - » A property of a product, because different vendors may sell the same product at different prices
  - » A property of a person, because different products may be bought by the same person at different prices

Relationship With Attributes (2/2)

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

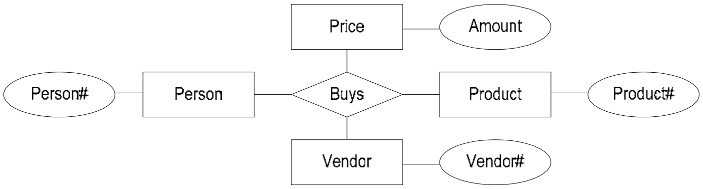


Entity Versus Attribute

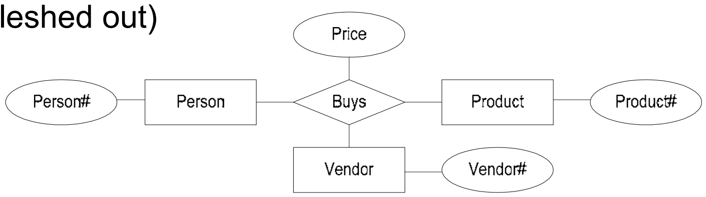
- Entities can model situations that attributes cannot model naturally
- Entities can
  - » Participate in relationships
  - » Have attributes
- Attributes cannot do any of these
- Let us look at a “fleshed out example” for possible alternative modeling of Buys

Other Choices For Modeling Buys (1/2)

- 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

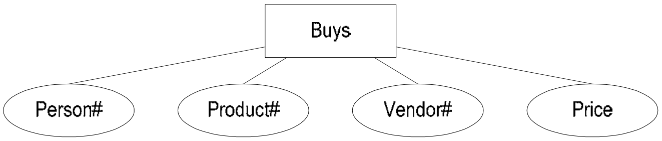


- We should probably have (as we had earlier less fleshed out)



Other Choices For Modeling Buys (2/2)

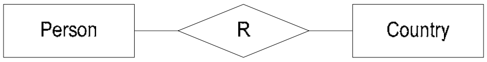
- Or should we just have this?



- Not if we want to model something about a person, such as the date of birth of a person or whom a person likes
- These require a person to have an attribute (date of birth) and 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

Binary Relationships And Their Functionality (1/8)

- 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



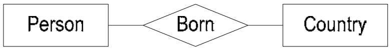
- We will be making some simple assumptions about persons and countries, which we list when relevant

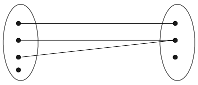
Binary Relationships And Their Functionality (2/8)

- Relationship R is called ***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 (may not in a country but in a ship

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





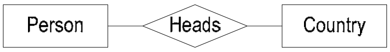
- The picture on the right describes the universe of four persons and three countries, with lines indicating which person was born in which country
  - We will have similar diagrams for other examples

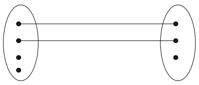
Binary Relationships And Their Functionality (3/8)

- The 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 related to it and for each element of B there exists at most one element of A related to it
  - » Example: R is Heads

    Each Person is a Head (President, Queen, etc.) of at most one country

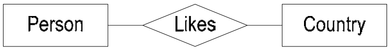
    Each country has at most one head (maybe the queen died and it is not clear who will be the monarch next)
- In other words, R is one to one, if and only ir
  - » R is many to one from A to B, and
  - » R is many to one from B to A

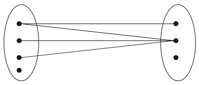




Binary Relationships And Their Functionality (4/8)

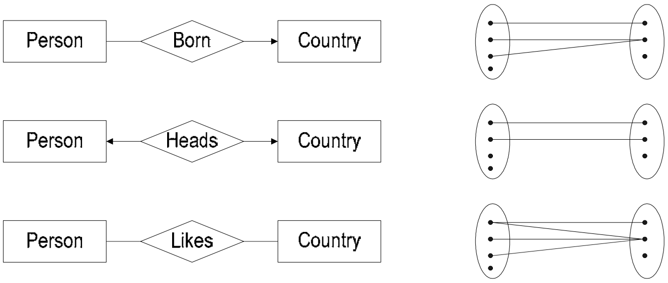
- The relationship is called ***many to many*** between A and B, if it is not many to one from A to B and it is not many to one from B to A
  - » Example: R is “likes”





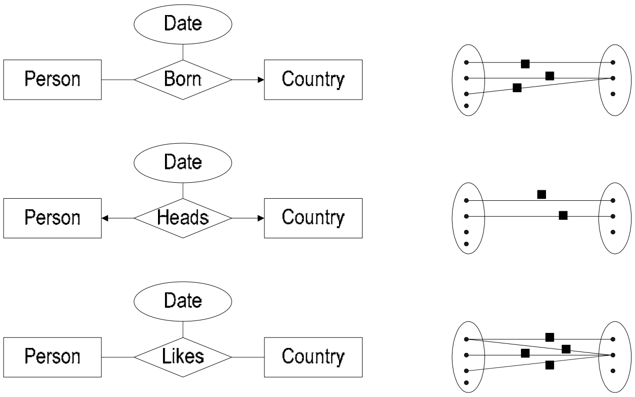
Binary Relationships And Their Functionality (5/8)

- We have in effect considered the concepts of partial functions of one variable.
  - The first two examples were ***partial functions***
  - 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"



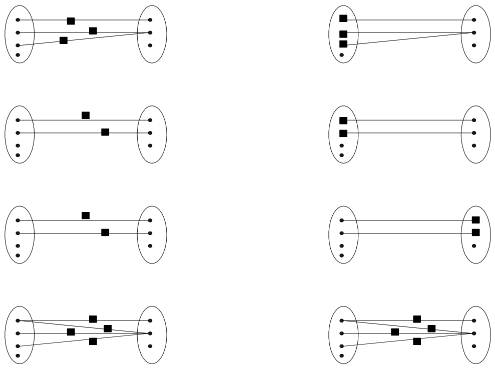
Binary Relationships And Their Functionality (6/8)

- How about properties of the relationship?
- Date: when the two people in a relationship first entered into the relationship (marked also with black square)



Binary Relationships And Their Functionality (7/8)

- Can make Date in some cases the property of an entity
  - » "Slide" the Date to the Person, but not the Country
  - » "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

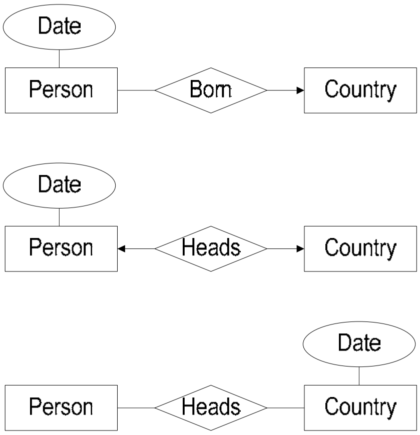


Binary Relationships And Their Functionality (8/8)

- This can be done if the relationship is many-to-one
  - » Then, the property of the relationship can be attributed to the "many" side
- This can be done if the relationship is one-to-one
  - » Then a property of the relationship can be "attributed" to any of the two sides

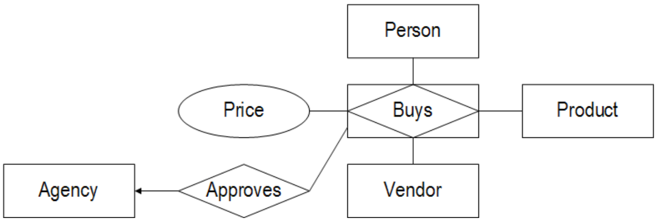
Alternate Designs

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



Aggregation: Relationships As Entities

- It is sometimes natural to consider relationships as if they were entities.
- This will allow us to let relationships participate in other “higher order” relationships
- Here each “contract” 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 we are not confused



Strong And Weak Entities

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

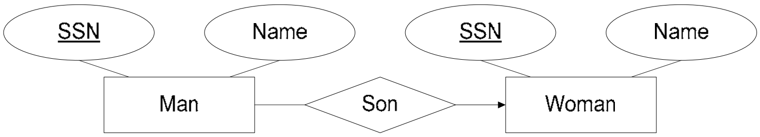
  Tacitly, we assumed only such entities so far

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

  They can be identified by a combination of their attributes and the relationship they have with another entity set

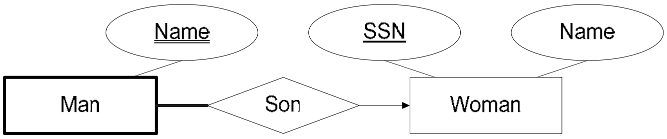
Man As A Strong Entity

- 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 (1/6)

- 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 the same name to more than one of her sons, a man can be identified by his name and by his mother's SSN



Man As A Weak Entity (2/6)

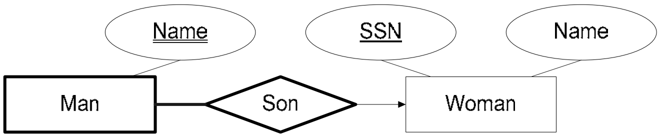
- We could have the following situation of two mothers: one with two sons, and one with three sons, when we gave people also heights in inches (just to have additional attributes that are not necessary for identification)

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

- SSN: 056-35-4321, height 68
  - Name: Bob, height 35
  - Name: Dave, height 45
  - Name: Victor, height 74

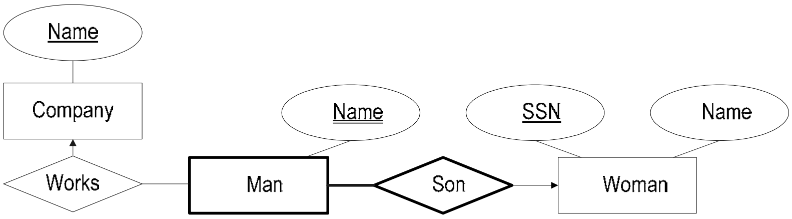
Man As A Weak Entity (3/6)

- Assuming that a woman does not have more than one son with the same name
- Name becomes a ***discriminant***
- Man can be identified by the combination of:
  - » The Woman to whom he is related under the Son relation. 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 used for identification
  - » His Name. His name is now a ***discriminant***; this is indicated by double underline



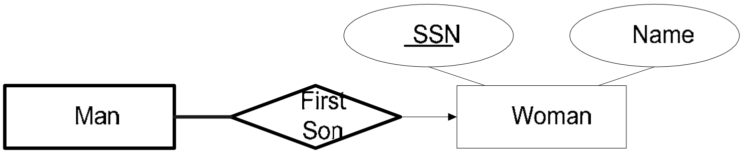
Man As A Weak Entity (4/6)

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



Man As A Weak Entity (5/6)

- Sometimes a discriminant 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 discriminant)

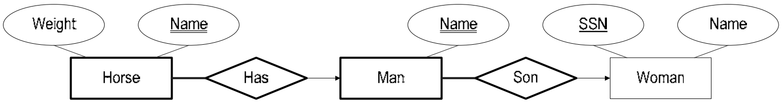


Man As A Weak Entity (6/6)

- In general, more than one attribute may be needed as a discriminant
- 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 discriminant

From Weaker To Stronger

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

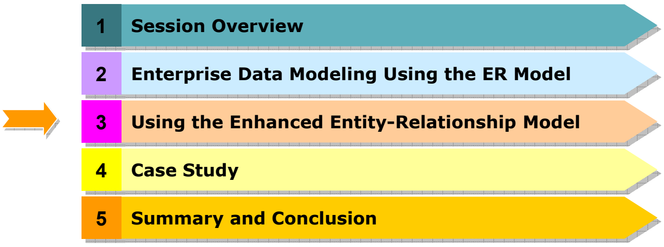


- A woman can have several sons, each of whom can have several horses

Summary

- Basic ER model concepts of entities and their attributes
  - Different types of attributes
  - Structural constraints on relationships
- ER diagrams represent E-R schemas
- UML class diagrams relate to ER modeling concepts

Agenda



1 Session Overview

2 Enterprise Data Modeling Using the ER Model

3 Using the Enhanced Entity-Relationship Model

4 Case Study

5 Summary and Conclusion

Agenda

- Subclasses, Superclasses, and Inheritance
- Specialization and Generalization
- Constraints and Characteristics of Specialization and Generalization Hierarchies
- Modeling of UNION Types Using Categories
- A Sample UNIVERSITY EER Schema, Design Choices, and Formal Definitions
- Example of Other Notation: Representing Specialization and Generalization in UML Class Diagrams
- Data Abstraction, Knowledge Representation, and Ontology Concepts

The Enhanced Entity-Relationship (EER) Model

- Enhanced ER (EER) model
  - Created to design more accurate database schemas
    - Reflect the data properties and constraints more precisely
  - More complex requirements than traditional applications

Subclasses, Superclasses, and Inheritance (1/3)

- EER model includes all modeling concepts of the ER model
- In addition, EER includes:
  - Subclasses and superclasses
  - Specialization and generalization
  - Category or union type
  - Attribute and relationship inheritance

Subclasses, Superclasses, and Inheritance (2/3)

- Enhanced ER or EER diagrams
  - Diagrammatic technique for displaying these concepts in an EER schema
- Subtype or subclass of an entity type
  - Subgroupings of entities that are meaningful
  - Represented explicitly because of their significance to the database application

Subclasses, Superclasses, and Inheritance (3/3)

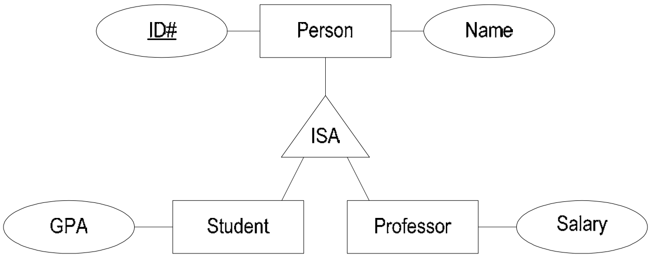
- Terms for relationship between a superclass and any one of its subclasses
  - Superclass/subclass
  - Supertype/subtype
  - Class/subclass relationship
- Type inheritance
  - Subclass entity inherits all attributes and relationships of superclass

The ISA Relationship (1/7)

- For certain purposes, we consider subsets 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 do not make sense for every entity in the “original” entity set
- ISA is indicated by a triangle
- The elements of the subset are weak entities, as we will note next

The ISA Relationship (2/7)

- 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



The ISA Relationship (3/7)

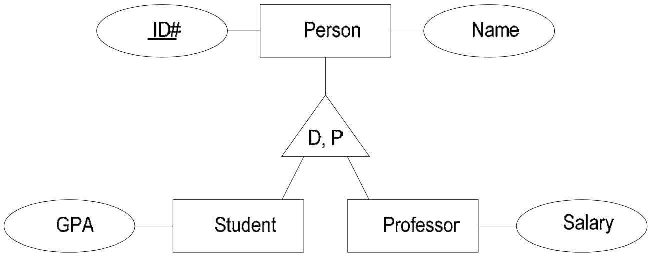
- Professor is a weak entity because it cannot be identified by its own attributes (here: Salary)
- Student is a weak entity because it cannot be identified by its own attributes (here: GPA)
- They do not have discriminants, nothing is needed to identify them in addition to the primary key of the strong entity (Person)
- The set and the subsets are sometimes referred to as class and subclasses

The ISA Relationship (4/7)

- A person associated with the university (and therefore in our database) can be
  - Only a professor
  - Only a student
  - Both a professor and a student
  - Neither a professor nor a student
- In general, ISA could be
  - ***Disjoint***: no entity could be in more than one subclass
  - ***Overlapping***: an entity could be in more than one subclass
  - ***Total***: every entity has to be in at least one subclass
  - ***Partial***: an entity does not have to be in any subclass
- This could be specified by replacing “ISA” in the diagram by an appropriate letter

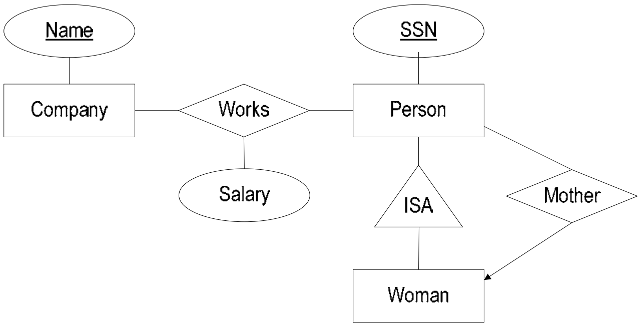
The ISA Relationship (5/7)

- Some persons are professors
- Some persons are students
- Some persons are neither professors nor students
- No person can be both a professor and a student



The ISA Relationship (6/7)

- Example: subsets participating in relationships modeling the assumed semantics more clearly (every person has one woman who is the legal mother)



The ISA Relationship (7/7)

- ISA is really a superclass/subclass relationship
- ISA could be ***specialization***: subsets are made out of the “more basic” set
- ISA could be ***generalization***: a superset is made of “more basic” sets
- Again, the diagram could be annotated to indicate this

Sample EER Diagram

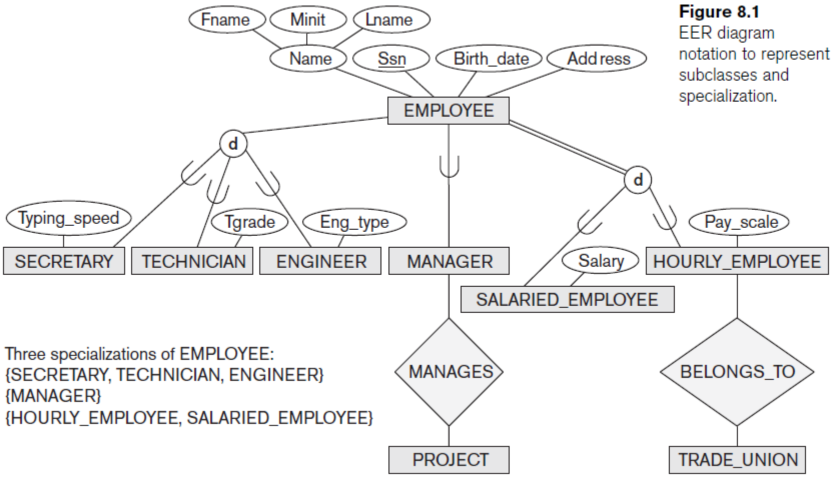


Figure 8.1
EER diagram notation to represent subclasses and specialization.

Three specializations of EMPLOYEE:
{SECRETARY, TECHNICIAN, ENGINEER}
{MANAGER}
{HOURLY_EMPLOYEE, SALARIED_EMPLOYEE}

Specialization and Generalization (1/2)

- Specialization
  - Process of defining a set of subclasses of an entity type
  - Defined on the basis of some distinguishing characteristic of the entities in the superclass
- Subclass can define:
  - Specific attributes
  - Specific relationship types

Instances of a Specialization

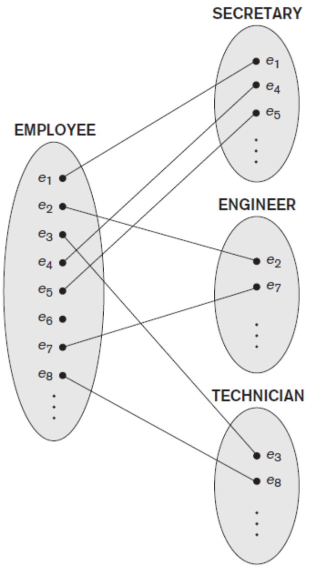


Figure 8.2
Instances of a specialization.

Specialization and Generalization (2/2)

- Certain attributes may apply to some but not all entities of the superclass
- Some relationship types may be participated in only by members of the subclass

Generalization

- Reverse process of abstraction
- Generalize into a single superclass
  - Original entity types are special subclasses
- Generalization
  - Process of defining a generalized entity type from the given entity types

Constraints and Characteristics of Specialization/Generalization Hierarchies

- Constraints that apply to a single specialization or a single generalization
- Differences between specialization/ generalization lattices and hierarchies

Constraints on Specialization and Generalization (1/2)

- May be several or one subclass
- Determine entity subtype:
  - Predicate-defined (or condition-defined) subclasses
  - Attribute-defined specialization
  - User-defined

Constraints on Specialization and Generalization (2/2)

- Disjointness constraint
  - Specifies that the subclasses of the specialization must be disjoint
- Completeness (or totalness) constraint
  - May be total or partial
- Disjointness and completeness constraints are independent

Specialization and Generalization Hierarchies and Lattices (1/2)

- Specialization hierarchy
  - Every subclass participates as a subclass in only one class/subclass relationship
  - Results in a tree structure or strict hierarchy
- Specialization lattice
  - Subclass can be a subclass in more than one class/subclass relationship

Sample Specialization Lattice

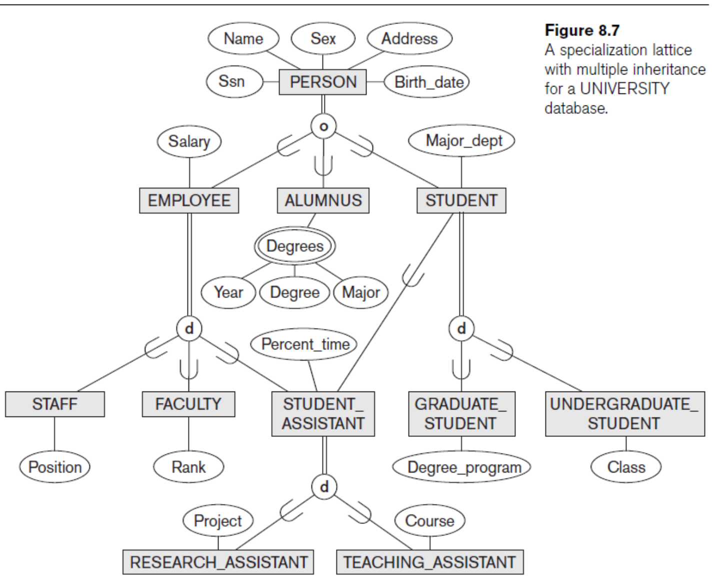


Figure 8.7
A specialization lattice with multiple inheritance for a UNIVERSITY database.

Specialization and Generalization Hierarchies and Lattices (2/2)

- Multiple inheritance
  - Subclass with more than one superclass
  - If attribute (or relationship) originating in the same superclass inherited more than once via different paths in lattice
    - Included only once in shared subclass
- Single inheritance
  - Some models and languages limited to single inheritance

Utilizing Specialization and Generalization in Refining Conceptual Schemas

- Specialization process
  - Start with entity type then define subclasses by successive specialization
  - Top-down conceptual refinement process
- Bottom-up conceptual synthesis
  - Involves generalization rather than specialization

Modeling of UNION Types Using Categories

- Union type or a category
- Represents a single superclass/subclass relationship with more than one superclass
- Subclass represents a collection of objects that is a subset of the UNION of distinct entity types
- Attribute inheritance works more selectively
- Category can be total or partial
- Some modeling methodologies do not have union types

A Sample UNIVERSITY EER Schema, Design Choices, and Formal Definitions

- The UNIVERSITY Database Example
  - UNIVERSITY database
    - Students and their majors
    - Transcripts, and registration
    - University’s course offerings

Sample EER Conceptual Schema for a University Database

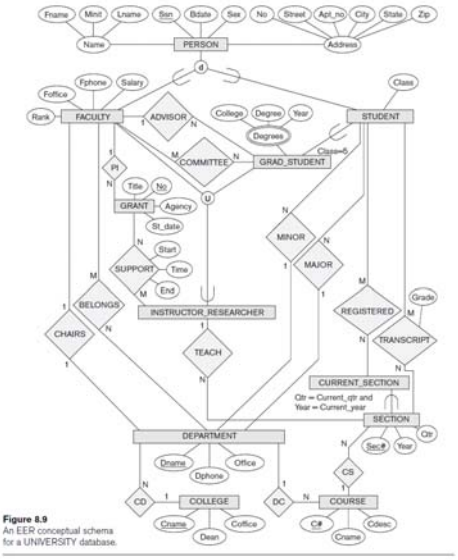


Figure 8.9
An EER conceptual schema for a UNIVERSITY database.

Design Choices for Specialization/Generalization (1/3)

- Many specializations and subclasses can be defined to make the conceptual model accurate
- If subclass has few specific attributes and no specific relationships
  - Can be merged into the superclass

Design Choices for Specialization/Generalization (2/3)

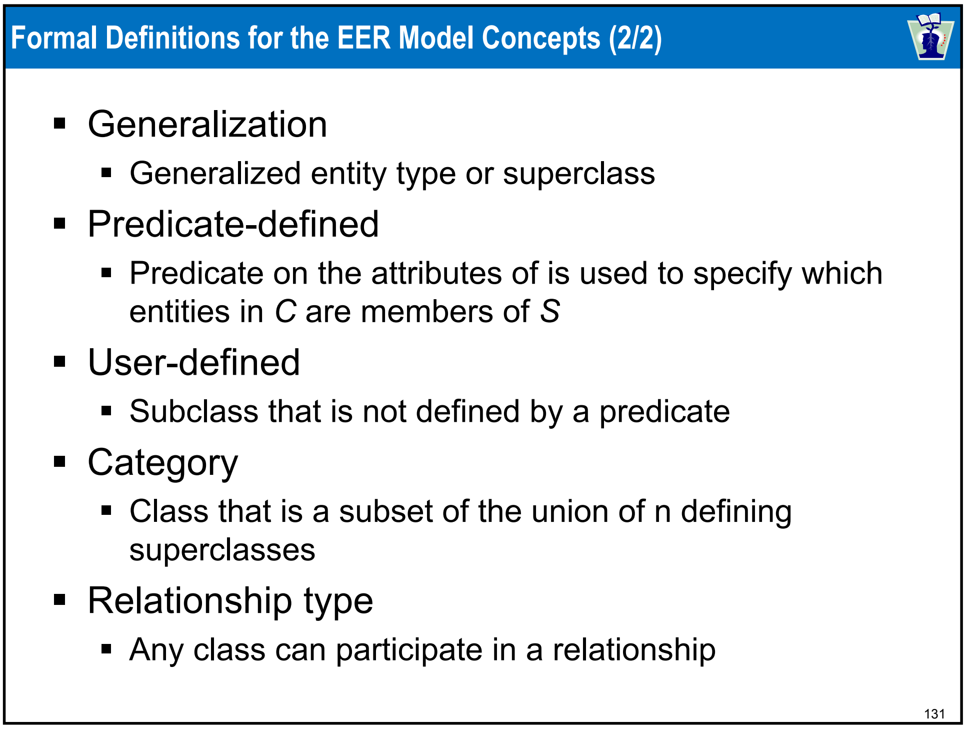
- If all the subclasses of a specialization/generalization have few specific attributes and no specific relationships
  - Can be merged into the superclass
  - Replace with one or more type attributes that specify the subclass or subclasses that each entity belongs to

Design Choices for Specialization/Generalization (3/3)

- Union types and categories should generally be avoided
- Choice of disjoint/overlapping and total/partial constraints on specialization/generalization
  - Driven by rules in miniworld being modeled

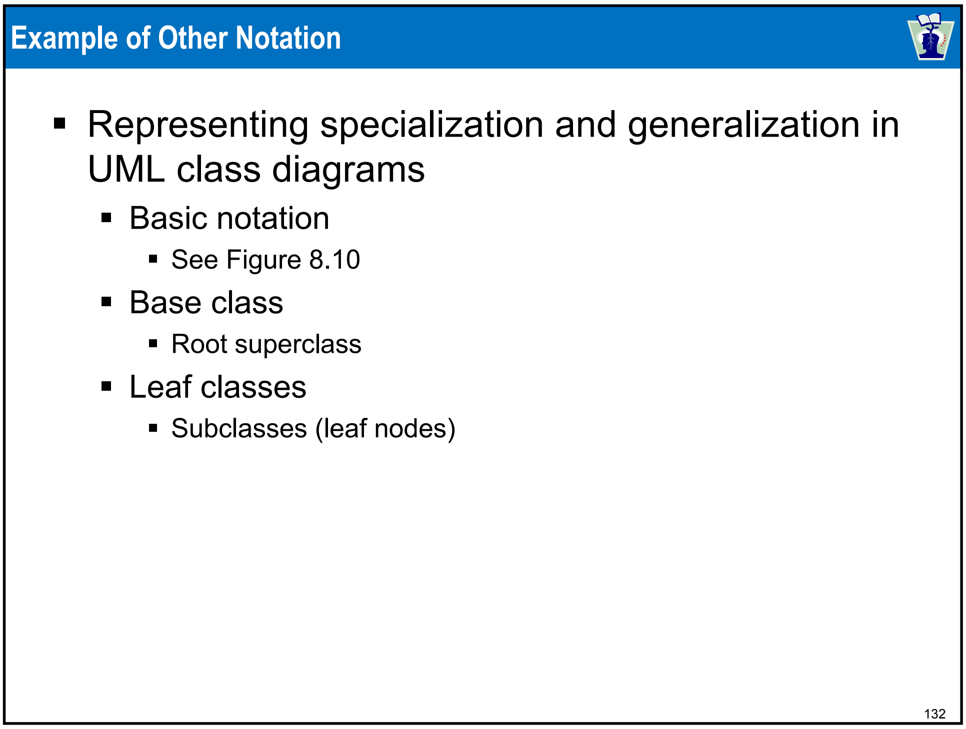
Formal Definitions for the EER Model Concepts (1/2)

- Class
  - Set or collection of entities
  - Includes any of the EER schema constructs of group entities
- Subclass
  - Class whose entities must always be a subset of the entities in another class
- Specialization
  - Set of subclasses that have same superclass



Formal Definitions for the EER Model Concepts (2/2)

- Generalization
  - Generalized entity type or superclass
- Predicate-defined
  - Predicate on the attributes of is used to specify which entities in *C* are members of *S*
- User-defined
  - Subclass that is not defined by a predicate
- Category
  - Class that is a subset of the union of n defining superclasses
- Relationship type
  - Any class can participate in a relationship



Example of Other Notation

- Representing specialization and generalization in UML class diagrams
  - Basic notation
    - See Figure 8.10
  - Base class
    - Root superclass
  - Leaf classes
    - Subclasses (leaf nodes)

Sample UML Class Diagram

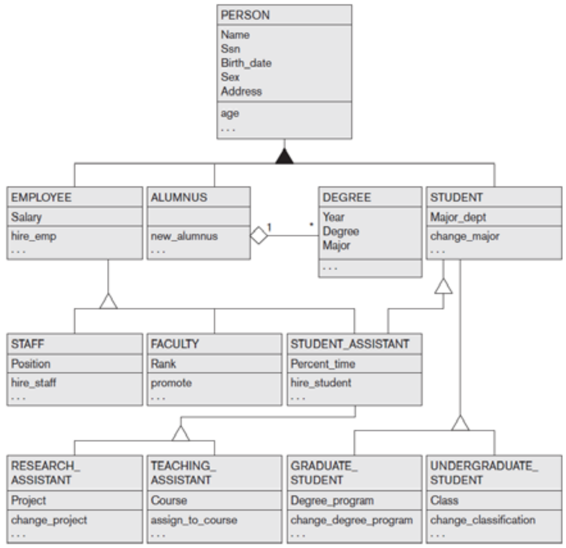


Figure 8.10
A UML class diagram corresponding to the EER diagram in Figure 8.7, illustrating UML notation for specialization/generalization.

Data Abstraction, Knowledge Representation, and Ontology Concepts

- Goal of knowledge representation (KR) techniques
  - Accurately model some domain of knowledge
  - Create an ontology that describes the concepts of the domain and how these concepts are interrelated
- Goals of KR are similar to those of semantic data models
  - Important similarities and differences

Classification and Instantiation (1/2)

- Classification
  - Systematically assigning similar objects/entities to object classes/entity types
- Instantiation
  - Inverse of classification
  - Generation and specific examination of distinct objects of a class

Classification and Instantiation (2/2)

- Exception objects
  - Differ in some respects from other objects of class
  - KR schemes allow such class properties
- One class can be an instance of another class (called a meta-class)
  - Cannot be represented directly in EER model

Identification

- Abstraction process
- Classes and objects are made uniquely identifiable by means of some identifier
- Needed at two levels
  - To distinguish among database objects and classes
  - To identify database objects and to relate them to their real-world counterparts

Specialization and Generalization

- Specialization
  - Classify a class of objects into more specialized subclasses
- Generalization
  - Generalize several classes into a higher-level abstract class
  - Includes the objects in all these classes

Aggregation and Association

- Aggregation
  - Abstraction concept for building composite objects from their component objects
- Association
  - Associate objects from several independent classes
- Main structural distinction
  - When an association instance is deleted
    - Participating objects may continue to exist

Sample EER Model Illustrating Aggregation (1/2)

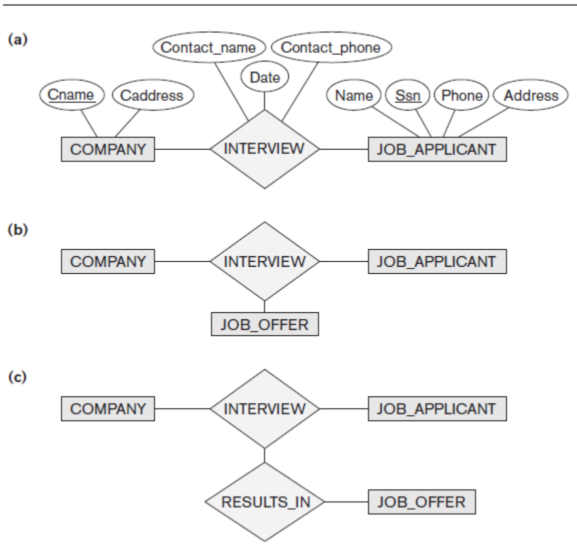


Figure 8.11
Aggregation. (a) The relationship type INTERVIEW. (b) Including JOB_OFFER in a ternary relationship type (incorrect). (c) Having the RESULTS_IN relationship participate in other relationships (not allowed in ER). (d) Using aggregation and a composite (molecular) object (generally not allowed in ER but allowed by some modeling tools). (e) Correct representation in ER.

Sample EER Model Illustrating Aggregation (2/2)

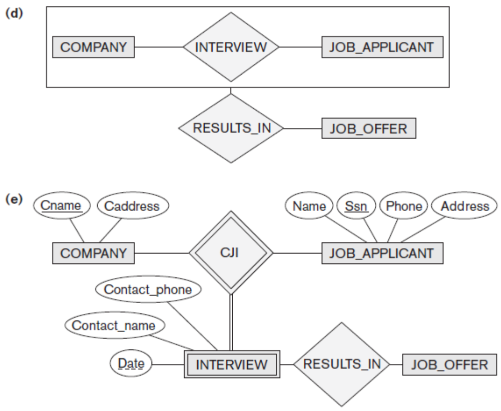


Figure 8.11
Aggregation. (a) The relationship type INTERVIEW. (b) Including JOB_OFFER in a ternary relationship type (incorrect). (c) Having the RESULTS_IN relationship participate in other relationships (not allowed in ER). (d) Using aggregation and a composite (molecular) object (generally not allowed in ER but allowed by some modeling tools). (e) Correct representation in ER.

Ontologies and the Semantic Web

- Documents contain less structure than database information does
- Semantic Web
  - Allow meaningful information exchange and search among machines
- Ontology
  - Specification of a conceptualization
- Specification
  - Language and vocabulary terms used to specify conceptualization

Summary

- Enhanced ER or EER model
  - Extensions to ER model that improve its representational capabilities
  - Subclass and its superclass
  - Category or union type
- Notation and terminology of UML for representing specialization and generalization

Agenda

1 Session Overview

2 Enterprise Data Modeling Using the ER Model

3 Using the Enhanced Entity-Relationship Model

4 Case Study

5 Summary and Conclusion

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 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 exercise various points, to be discussed later on how to design good relational databases

Our Application (1/2)

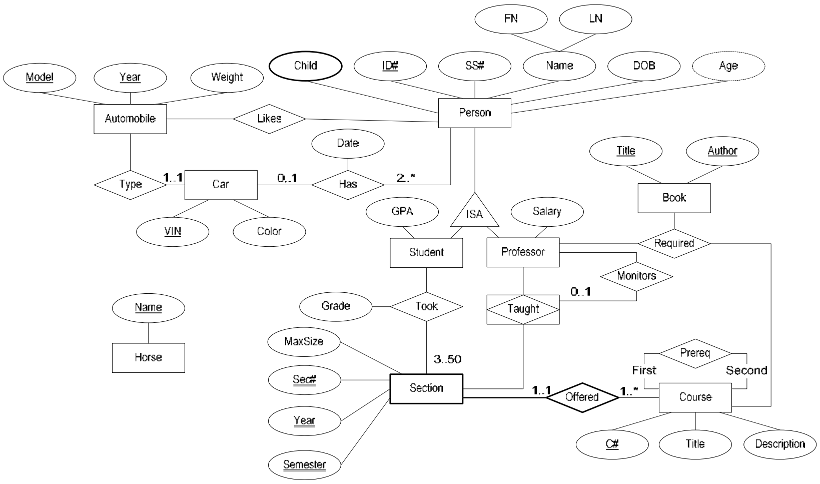
- We are supposed to design a database for a university
- We will look at a small fragment of the application and will model it as an entity relationship diagram annotated with comments, as needed to express additional features
- But it is still a reasonable "small" database
- In fact, much larger than what is commonly discussed in a course, but more realistic for modeling real applications

Our Application (2/2)

- 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 AV equipment) we look at the resulting ER diagram and construct it in pieces
- We will pick some syntax for annotations, as this is not standard

Building The ER Diagram

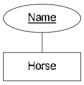
- We describe the application in stages, getting:



Horse

- ***Horse***; entity set
- Attributes:
  - » ***Name***
- Constraints
  - » Primary Key: Name

Our ER Diagram



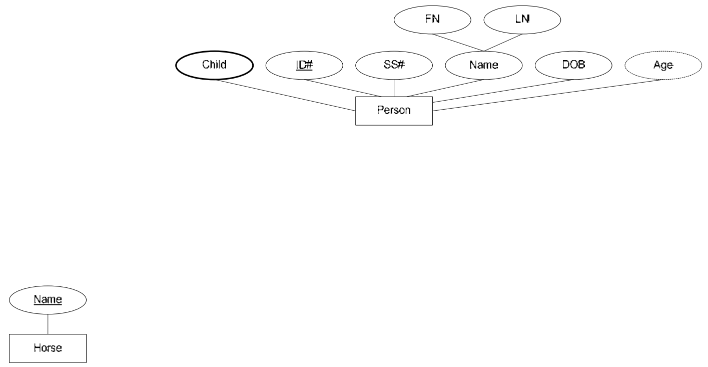
Horse

- We should specify what is the domain of each attribute, in this case, Name only
- We will generally not do it in our example, as there is nothing interesting in it
  - We could say that Name is an alphabetic string of at most 100 characters, for example

Person

- ***Person***; entity set
- Attributes:
  - ***Child****; a multivalued attribute*
  - ***ID#***
  - ***SS#***
  - ***Name****; composite attribute, consisting of*
    - ***FN***
    - ***LN***
  - ***DOB***
  - ***Age****; derived attribute* (we should state how it is computed)
- Constraints
  - Primary Key: ID#
  - Unique: SS#

Our ER Diagram



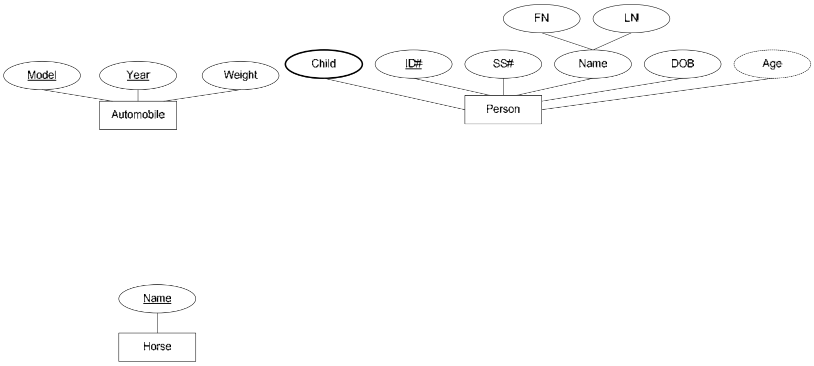
Person

- Since ID# is the primary key (consisting here of one attribute), we will consistently identify a person using the value of this attribute
- Since SS# is unique, no two persons will have the same SS#

Automobile

- ***Automobile***; entity set
- Attributes:
  - ***Model***
  - ***Year***
  - ***Weight***
- Constraints
  - Primary Key: Model,Year
- Note: Automobile is a “catalog entry”
  - It is not a specific “physical car”

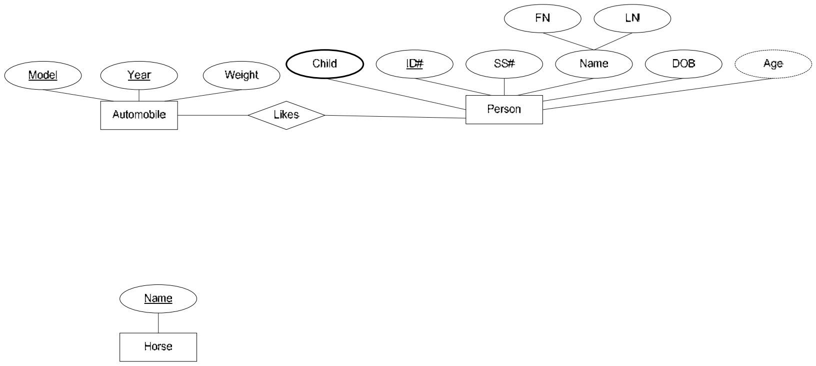
Our ER Diagram



Likes

- ***Likes***; relationship
- Relationship among/between:
  - » ***Person***
  - » ***Automobile***
- Attributes
- Constraints

Our ER Diagram



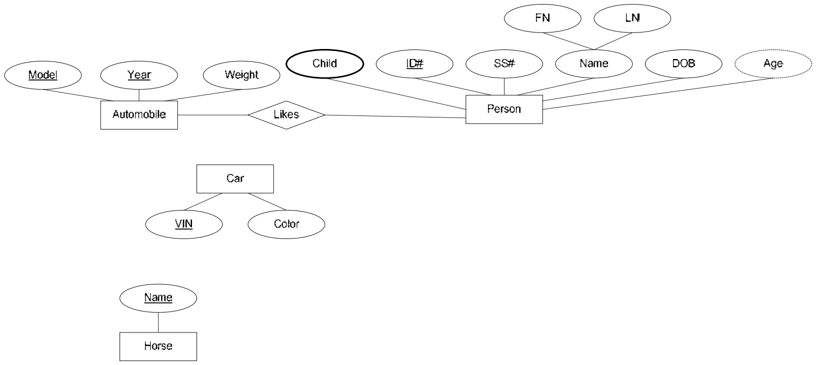
Likes

- This relationship has no attributes
- This relationship has no constraints
- This relationship is a general many-to-many relationship (as we have not said otherwise)
- This relationship does not have any cardinality constraints

Car

- ***Car***; entity set
- Attributes
  - ***VIN***
  - ***Color***
- Constraints
  - Primary Key: VIN
- Note: Car is a "physical entity"
  - VIN stands for "Vehicle Identification Number," which is like a Social Security Number for cars

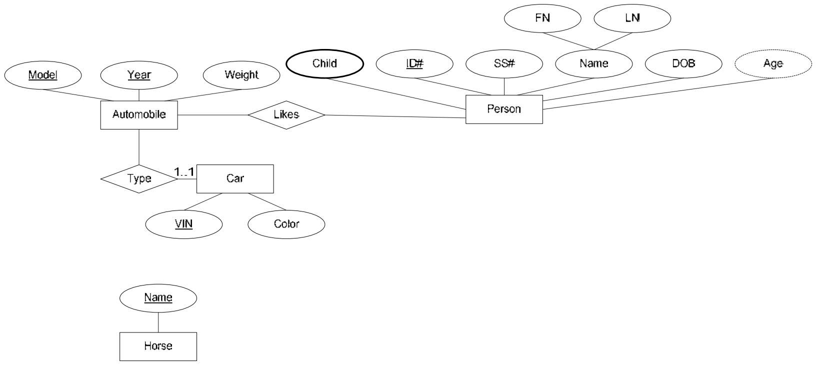
Our ER Diagram



Type

- ***Type***; relationship
- Relationship among/between:
  - ***Automobile***
  - ***Car***
- Attributes
- Constraints
  - Cardinality: 1..1 between Car and Type
- This tells us for each physical car what is the automobile catalog entry of which it is an instantiation
  - Each car is an instantiation of a exactly one catalog entry

Our ER Diagram



Type

- We see that the relationship Type is:
- Many to one from Car to Automobile
- It is total not partial

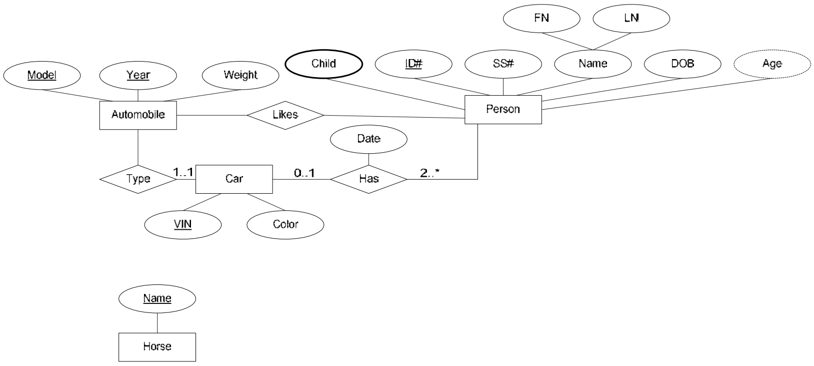
  In other words, it is a total function from Car to Automobile
- Not every Automobile is a “target”

  There may be elements in Automobile for which no Car exists

Has

- ***Has***; relationship
- Relationship among/between
  - » ***Person***
  - » ***Car***
- Attributes
  - » ***Date***
- Constraints
  - » Cardinality: 2..* between Person and Has
  - » Cardinality: 0..1 between Car and Has
- Date tells us when the person got the car
- Every person has at least two cars
- Every car can be had (owned) by at most one person
  - » Some cars may have been abandoned

Our ER Diagram



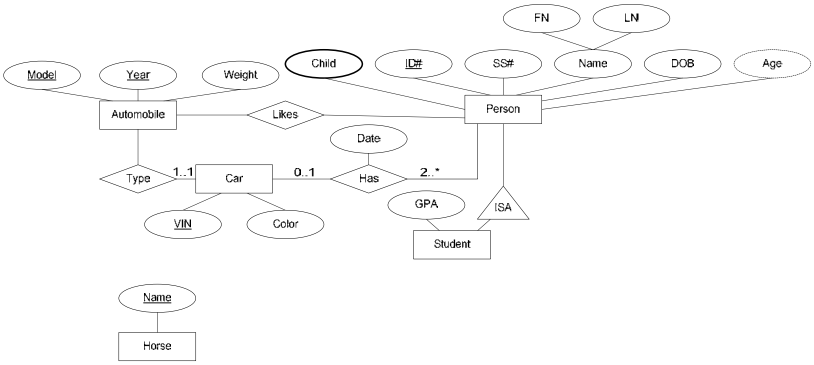
Has

- We see that Has is a partial function from Car to Person
- Every Person is a “target” in this function (in fact at least twice)

Student

- ***Student***; entity set
- Subclass of Person
- Attributes
  - ***GPA***
- Constraints
- Note that Student is a weak entity
  - It is identified through a person
  - You may think of a student as being an “alias” for some person
  - “Split personality”

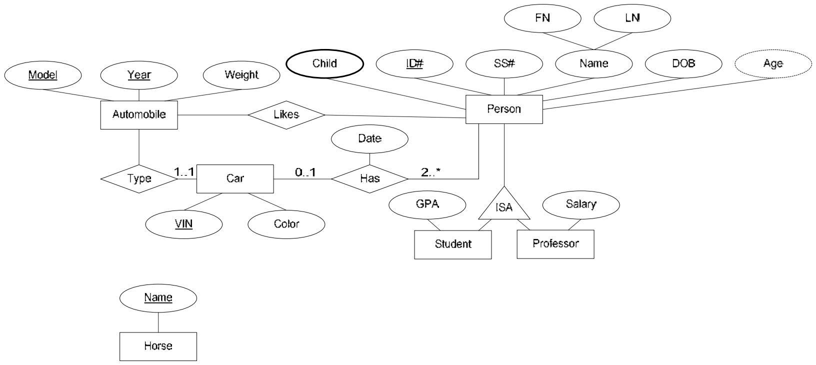
Our ER Diagram



Professor

- ***Professor***; entity set
- Subclass of Person
- Attributes
  - ***Salary***
- Constraints

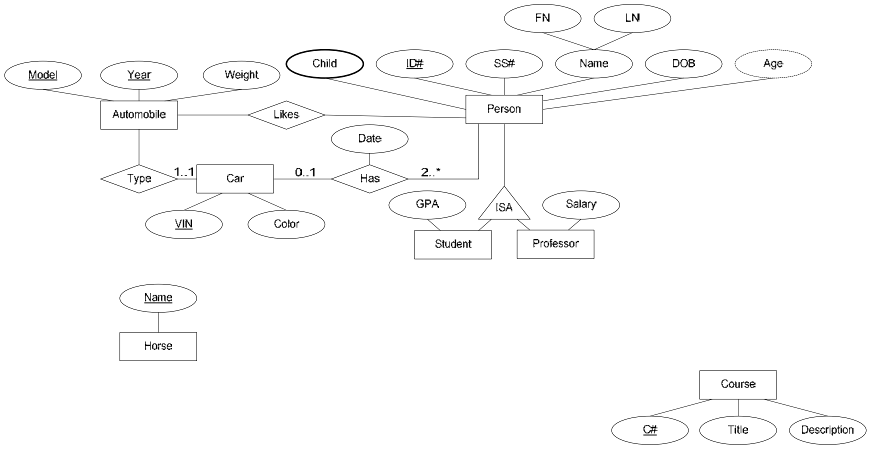
Our ER Diagram



Course

- ***Course***; entity set
- Attributes:
  - ***C#***
  - ***Title***
  - ***Description***
- Constraints
  - Primary Key: C#

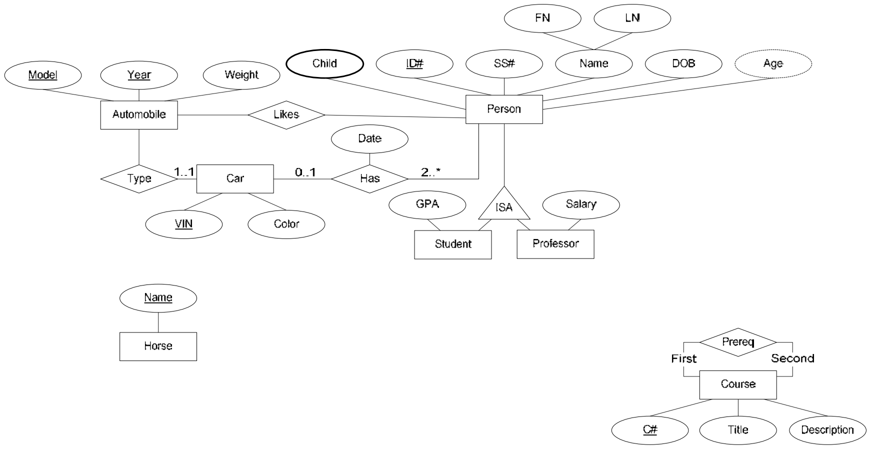
Our ER Diagram



Prereq

- ***Prereq***; relationship
- Relationship among/between:
  - ***Course***; role: First
  - ***Course***; role: Second
- Attributes
- Constraints
- We have a directed graph on courses, telling us prerequisites for each course, if any
  - To take “second” course every “first” course related to it must have been taken previously
  - We needed the roles first and second, to be clear
  - Note how we model well that prerequisites are not between offerings of a course but catalog entries of courses

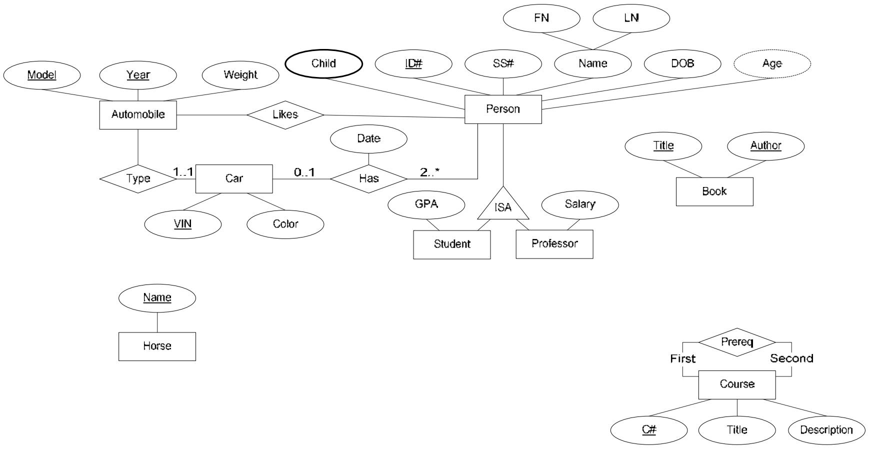
Our ER Diagram



Book

- ***Book***; entity set
- Attributes:
  - ***Author***
  - ***Title***
- Constraints
  - Primary Key: Author,Title

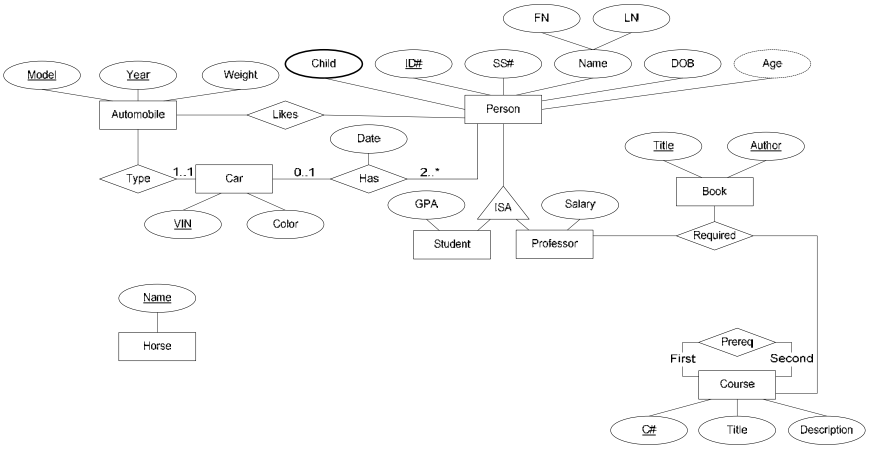
Our ER Diagram



Required

- ***Required***; relationship
- Relationship among/between:
  - **Professor**
  - **Course**
  - **Book**
- Attributes
- Constraints
- A professor specifies that a book is required for a course

Our ER Diagram



Required

- Note that there are no cardinality or other restrictions
- Any professor can require any book for any course and a book can be specified by different professors for the same course
- A book does not have to required for any course

Section

- ***Section***; entity set
- Attributes:
  - » ***Year***
  - » ***Semester***
  - » ***Sec#***
  - » ***MaxSize***
- Constraints
  - » Discriminant: Year, Semester, Sec#
  - » Identified through relationship Offered to Course
  - » Each Course has to have at least one Section (we do not put a course in a catalog unless it has been offered at least once)

Section

- Section is a weak entity
- It is related for the purpose of identification to a strong entity Course by a new relationship Offered
- It has a discriminant, so it is in fact identified by having the following specified

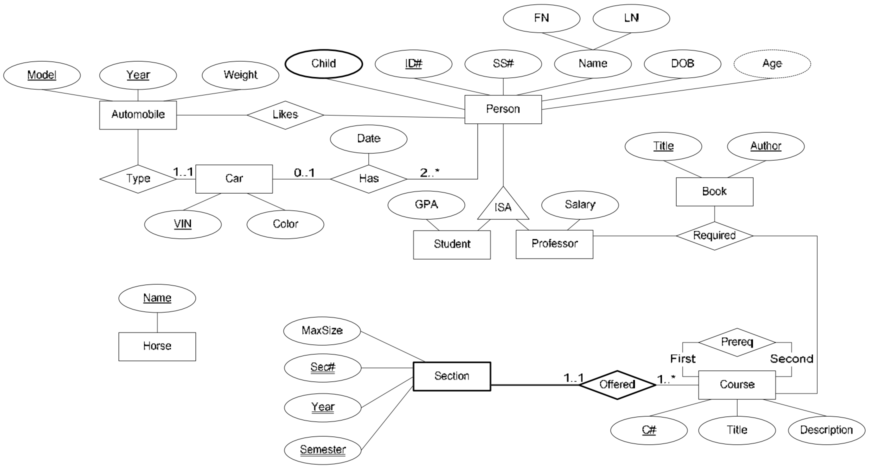
  C#, Year, Semester, Sec#
- Our current section is:

  G22.2433, 2011, Fall, 001

Offered

- ***Offered***; relationship
- Relationship among/between:
  - » **Course**
  - » **Section**
- Attributes
- Constraints
  - » Course has to be related to at least one section (see above)
  - » Section has to be related to exactly one course (this automatically follows from the fact that section is identified through exactly one course, so maybe we do not need to say this)

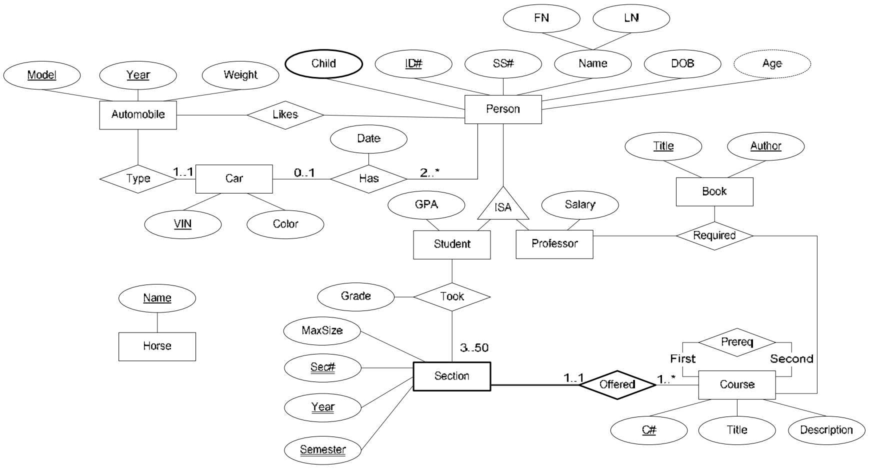
Our ER Diagram



Took

- ***Took***; relationship
- Relationship among/between
  - ***Student***
  - ***Section***
- Attributes
  - ***Grade***
- Constraints
  - Cardinality: 3..50 between Section and Took (this means that a section has between 3 and 50 students)

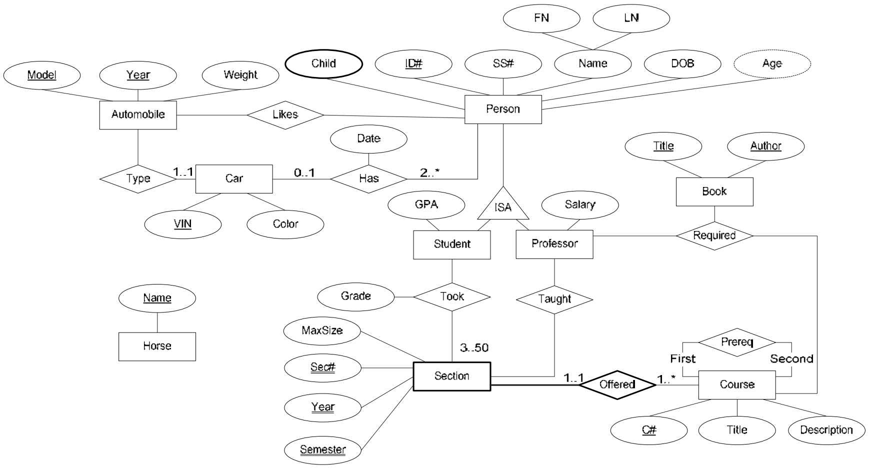
Our ER Diagram



Taught

- ***Taught***; relationship
- Relationship among/between
  - ***Professor***
  - ***Section***
- Attributes
- This tells us which professor teach which sections
  - Note there is no cardinality constraint: any number of professors, including zero professors can teach a section (no professor yet assigned, or hypothetical situation)
  - If we wanted, we could have put 1..* between Section and Taught to specify that at least one professor has to be assigned to each section

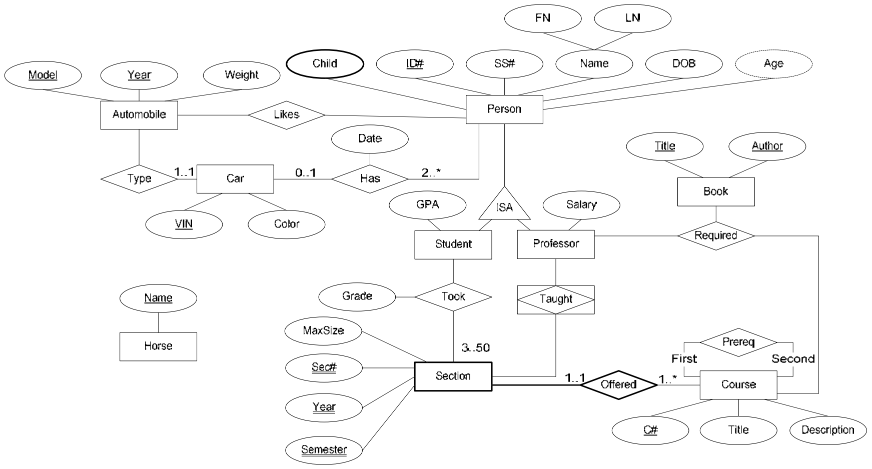
Our ER Diagram



Taught

- We want to think of Taught as an entity
  - » We will see soon why

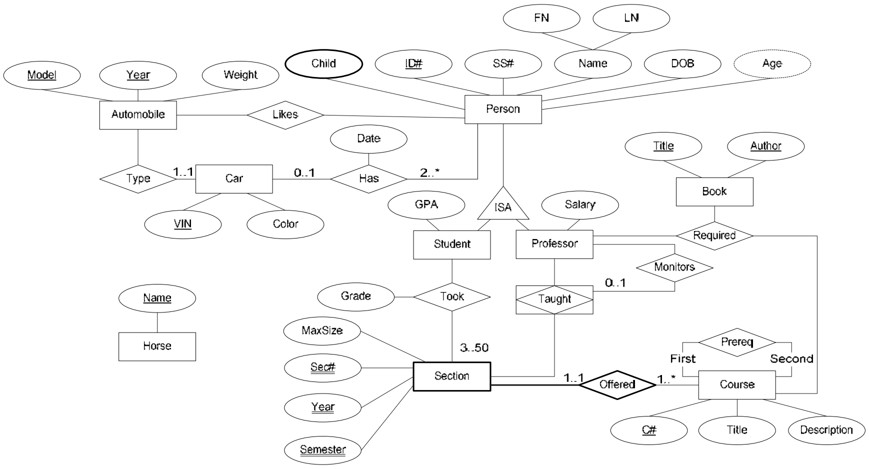
Our ER Diagram



Monitors

- ***Monitors***; relationship
- Relationship among/between
  - » ***Professor***
  - » ***Taught*** (considered as an entity)
- Attributes
- Constraints
  - » Cardinality: 0..1 between Taught and Professor
- This models the fact that Taught (really a teaching assignment) may need to be monitored by a professor and at most one professor is needed for such monitoring
  - » We are not saying whether the professor monitoring the assignment has to be different from the teaching professor in this assignment (but we could do it in SQL DDL, as we shall see later)

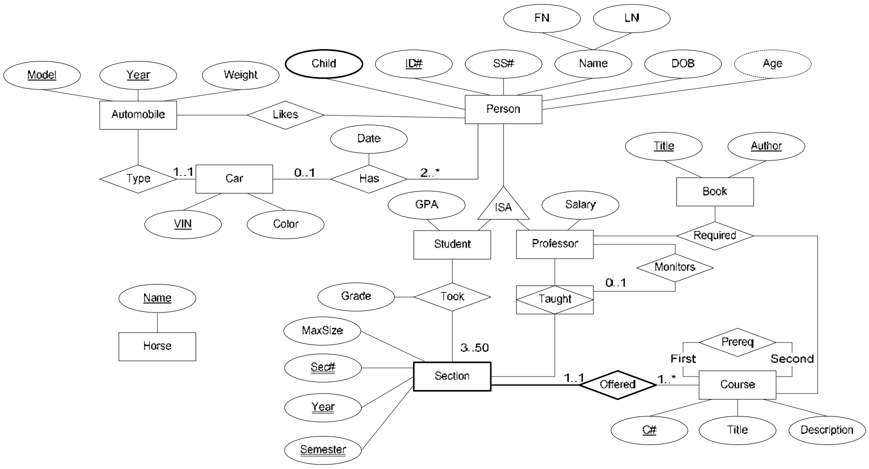
Our ER Diagram



What Can We Learn From The Diagram?

- Let’s look
- We will review everything we can learn just by looking at the diagram

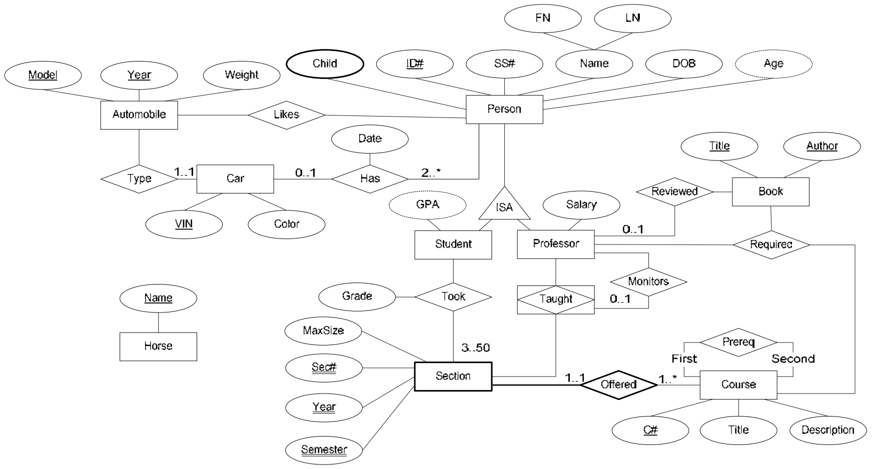
Our ER Diagram



GPA

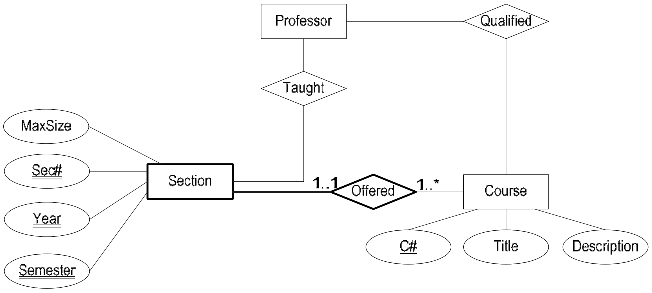
- We now observe that GPA should probably be modeled as a derived attribute, as it is computed from the student's grade history
- So, we may want to revise the diagram

Our ER Diagram



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



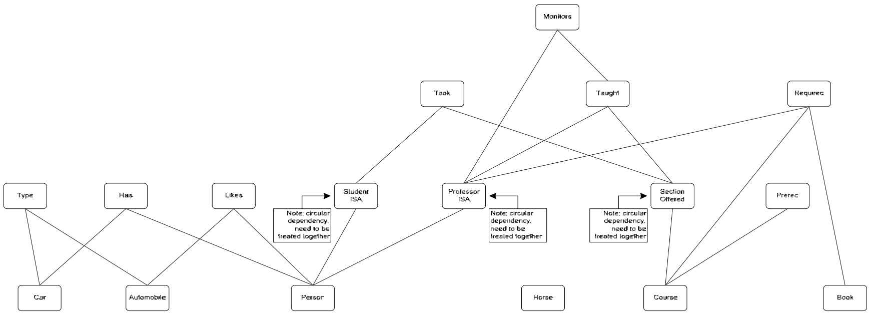
Annotate, Annotate, Annotate …

- An ER diagram should be annotated with all known constraints

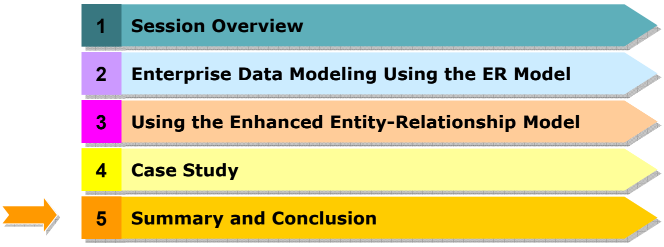
Hierarchy For Our ER Diagram

- 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
Requires
Type
Has
Likes
Note: circular dependency, need to be treated together
Student ISA
Professor ISA
Note: circular dependency, need to be treated together
Note: circular dependency, need to be treated together
Section Offered
Prerec
Car
Automobile
Person
Horse
Course
Book

Agenda



1 Session Overview

2 Enterprise Data Modeling Using the ER Model

3 Using the Enhanced Entity-Relationship Model

4 Case Study

5 Summary and Conclusion

Summary

- Basic ER model concepts of entities and their attributes
  - Different types of attributes
  - Structural constraints on relationships
- ER diagrams represent E-R schemas
- UML class diagrams relate to ER modeling concepts
- Enhanced ER or EER model
  - Extensions to ER model that improve its representational capabilities
  - Subclass and its superclass
  - Category or union type
- Notation and terminology of UML for representing specialization and generalization

Assignments & Readings

- Readings

  - Slides and Handouts posted on the course web site
  - Textbook: Chapters 7 & 8

- Assignment #2
  - Textbook exercises: TBA

- Project Framework Setup (ongoing)

Next Session: Practical Database Design

- We will learn how to take an ER diagram and convert it into a relational database
- We will learn how to specify such databases using CA’s ErWin or another tool
  - These tools allow you to produce a mapping of an ER model to a corresponding relational logical model
  - These tools are also used by Enterprise Data Architects to automatically generate SQL DDL code for various database systems

Any Questions?