

Topic 2: Database design

L19: Extended ER diagrams

Wolfgang Gatterbauer

CS3200 Database design (fa22)

<https://northeastern-datalab.github.io/cs3200/fa22s3/>

11/14/2022

Before class: Logistics ideas & suggestions

- Open Exam 2 discussion: any surprises?
 - Exam room: did CH103 work well? Academic integrity vs. space to spread
- I observed repeated computer problems during exams:
 - please test your setup before exams, possibly restart, switch off intensive processes
- Suggestion for exam 3: Gradescope uploads by students instead of instructor:
 - also cheat sheet separately
 - https://gradescope-static-assets.s3-us-west-2.amazonaws.com/help/submitting_hw_guide.pdf
- Why cheat sheets?
 - Preparing cheat sheets ("synthesizing") is a highly reflective exercise that helps you rethink what you learned in class. Class slides for learning not a reference manual.
 - What about repeating SDK notation on exam specification?
- Please treat in-class examples like homeworks: preparation for exams
- Please do use office hours to your advantage. We are here to help (and understand where problems are

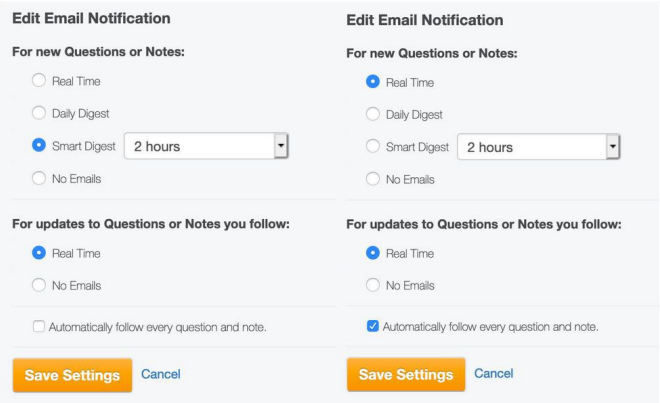
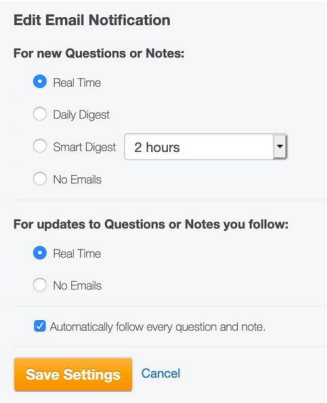
Class warm-up

- Class communication:
 - Several students did not see my Piazza announcements (e.g. posted L16 slides, example solutions to HW5). Reason: Piazza overload across different classes

Getting Help / Leaving Feedback / Piazza Tips

If at all possible, please reach out to us with your questions via Piazza (see link in Canvas navigation menu). That way whoever of us is at hand can answer your question first. Plus your peers can see the discussion and join the conversation. We use Piazza also as our main online message board from instructors to students. If I have updates to share, I will post them on Piazza. Thus I recommend you to automatically follow every new note.

→ Click on the arrow on the right upper corner from Piazza → Account/Email settings → Edit Email notifications:

Before:	After:
	

If you have any feedback on any aspect of this class whatsoever and prefer to remain anonymous, please use following [Anonymous feedback form](#). that only me the instructor can see. I will also ask you for a midterm feedback in the middle of this class to adjust the scope and pace of the class.

- Suggestion: Instructor makes announcements only via Canvas, Students via Piazza

- Literature cited below:
 - "... The groups of students who were doing best spontaneously formed study groups... Students who were not doing as well tended to do as the instructor suggested -- study two hours out of class for every hour in class -- but did it by themselves with little social support... even well-prepared students (high math SATs) are often disadvantaged by high school experiences that lead them to work alone... taught us that checking your homework with another student is cheating. "
- My goal with randomly assigned homework groups
 - help students (who are not social / don't have friends in class) get started
 - while avoiding people just splitting up the homework with their existing friends and then failing the exam

Class logistics: Breaking the "bamboo ceiling" (cp. glass ceiling)



Experiential Learning: We want you to be successful in real life. Thus, we like you to get out of your comfort zone to speak up and to contribute in class.

"'The loudest duck gets shot' is a Chinese proverb. 'The nail that sticks out gets hammered down' is a Japanese one. Its Western correlative: 'The squeaky wheel gets the grease.'" <http://nymag.com/news/features/asian-americans-2011-5/>

Many of these stereotypes and expectations have a basis in cultural misunderstandings. Some Asian Americans claim that they are raised with culture-specific values that affect perceptions of their workplace behaviour. For example, some report being taught from an early age to be self-effacing, reticent, respectful, and deferential towards authority.^{[3][31]} These values do not translate well into the American workplace, where Asian Americans' respectfulness can be misinterpreted as aloofness, arrogance, and inattentiveness.^[3] As a result, Asian Americans are less likely to be seen as having qualities that appeal to American employers, such as leadership, charisma and risk-taking, and are often passed over for promotions in spite of satisfactory or high job performance. Asian Americans are also less likely to aggressively network, self-promote, and speak up at work meetings with concerns and ideas when compared to their coworkers.^[3] Whilst Asian Americans who do are received negatively.^{[32][33]}

http://en.wikipedia.org/wiki/Bamboo_ceiling

Class logistics

- Class participation

▼ Participation & Logistics		5% of Total
	Honor code Due Sep 15 at 11:59pm	
	Class participation (will be entered at the end of the term) -/5 pts	
	Optional extra class contribution Due Dec 8 at 11:59pm	

Optional extra class contribution

Start Assignment

Due Dec 8 by 11:59pm Points 0 Submitting a file upload File Types pptx

Recall that part of the class grade depends on your class participation.

There are several ways to participate in class: e.g., asking questions that move the discussion forward (e.g. you ask a question that triggers me to update or improve or add a new example), answering questions (I ask a question, nobody answers for a while, then you raise your hand and make an attempt, even if wrong it is a great way to participate), pointing out errors in slides, participating on Piazza and asking questions or answering questions before I can. Asking me questions that made me add non-trivial clarifications, update examples to the class later. Bringing issues to my attention that affect all students and I am just not aware of. All of those are examples of great class participation and class contributions: all of those contribute to the overall discussion, interactive experience, lead to improvements for everyone, and thus help me help you and everyone. Great!

If by the end of the term you have so far not participated in class (e.g. never asked or answered questions in class or on Piazza), then there is no class participation. No class participation means 0 points.

For those that realize by December that this policy applies to them, I add here an optional assignment. Was there anything confusing in class for which you think you have a better way of explaining the concept? Anything were you were left confused but Google or a friend or a book came to the rescue and you wished we had used a similar example in class. Here you can submit a few PPTX slides that you think will be helpful for your peers in future variants of this class. If your suggestions make me reflect and consider adding them in a similar form in future classes, those can make partially up for currently no class participation.

Everybody can participate. It is completely optional.

Class logistics

PARTICIPATION

There are several ways to participate in class: e.g., asking questions that move the discussion forward (e.g. you ask a question that triggers me to update or improve or add a new example), answering questions (I ask a question, nobody answers for a while, then you raise your hand and make an attempt, even if wrong it is a great way to participate), pointing out errors in slides, participating on Piazza and asking questions or answering questions before I can. Asking me questions that make me add non-trivial clarifications or update examples to the class later. Bringing issues to my attention that affect all students and I am just not aware of. All of those are examples of great class participation and class contributions: all of those contribute to the overall discussion, interactive experience, lead to improvements for everyone, and thus help me help you and everyone. Great!

A rough guide for assigning participation points:

- never asking nor answering questions during class, never answering questions on Piazza = 0 points
- answering questions during class regularly (does not matter whether right or wrong) = 5 points
- answering questions carefully and regularly on Piazza before I can = 5 points
- asking regularly very insightful questions during class, pointing out misunderstandings that help me create better more illustrative examples \geq 5 points
- rest: somewhere in between

Please state your name before asking or answering questions in class, that will tremendously help me remember your names as soon as possible. Please also add photos to your profiles in Canvas and Piazza so I can more easily associate your class contributions.

Optional Extra Class Contribution

If by the end of the term you have not participated in class (e.g. never asked or answered questions in class or on Piazza), then there is no class participation. No class participation means 0 points.

For those that realize by December that this policy applies to them, I had added a completely optional assignment at the beginning of the term. Was there anything confusing in class for which you think you have a better way of explaining the concept? Anything were you were left confused but Google or a friend or a book came to the rescue and you wished we had used a similar example in class. Here you can submit a few PPTX slides that you think will be helpful for your peers in future variants of this class. If your suggestions make me reflect and consider adding them in a similar form in future classes, those can make partially up for currently no class participation.

A suggestion on how to best use class time!

- It is ok to make mistakes in class. Making mistakes in class is actually the best thing that can happen to you. You learn and will never make it again 😊
- From Ray Dalio's Principles (2017):
 - "Create a Culture in Which It Is Okay to Make Mistakes and Unacceptable Not to Learn from Them"
 - "Recognize that mistakes are a natural part of the evolutionary process."
 - "Don't feel bad about your mistakes or those of others. Love them!"



One reason why I don't post slides *before* lecture

From the preamble of one of the best physics books ever: „How to read this book“

The best way to use this book is NOT to simply read it or study it, but to read a question and STOP. Even close the book. Even put it away and THINK about the question. Only after you have formed a reasoned opinion should you read the solution. Why torture yourself thinking? Why jog? Why do push-ups?

If you are given a hammer with which to drive nails at the age of three you may think to yourself, “OK, nice.” But if you are given a hard rock with which to drive nails at the age of three, and at the age of four you are given a hammer, you think to yourself, “What a marvelous invention!” You see, you can't really appreciate the solution until you first appreciate the problem.

...

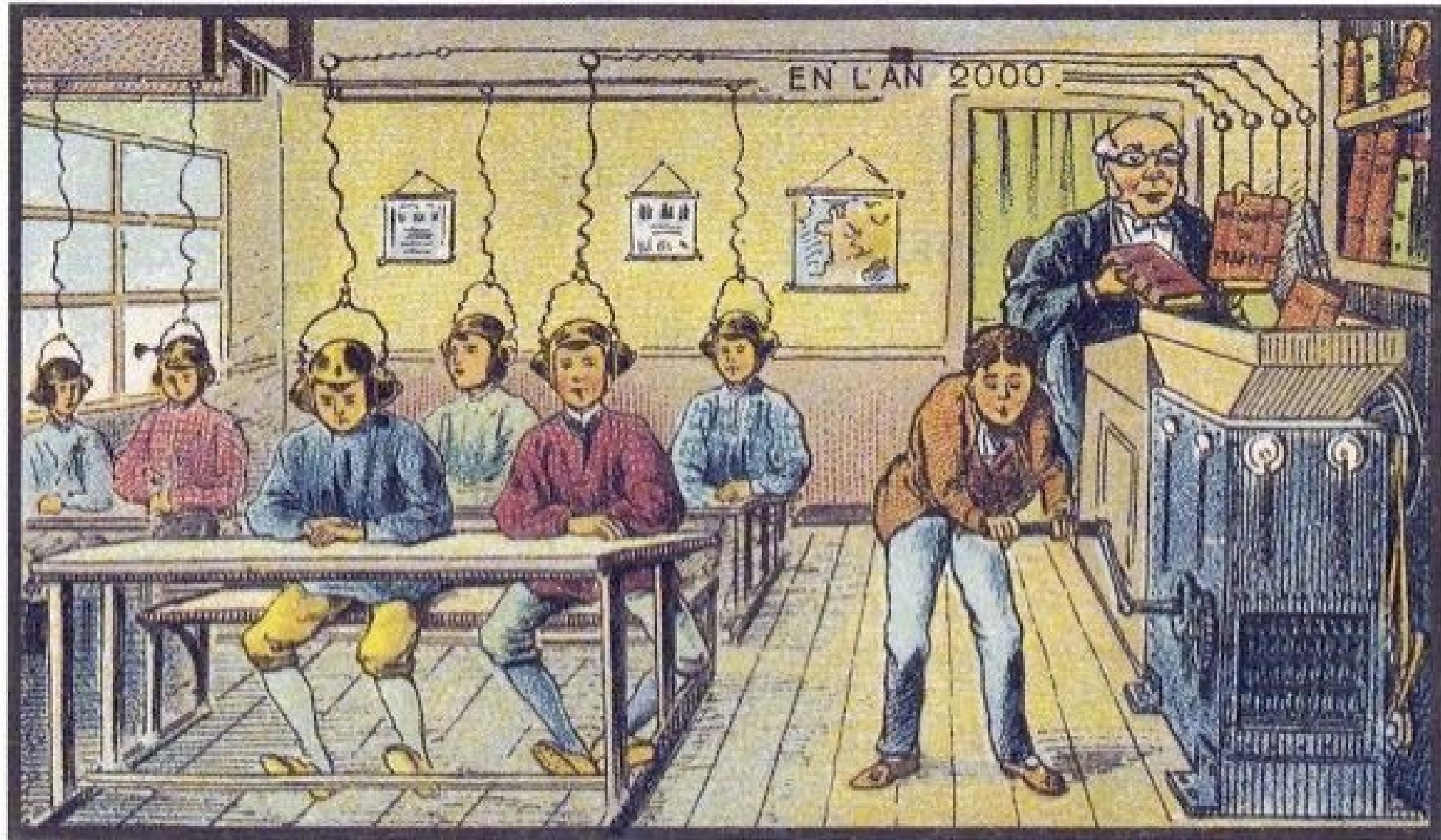
...

Let this book, then, be your guide to mental push-ups. Think carefully about the questions and their answers *before* you read the answers offered by the author. **You will find many answers don't turn out as you first expect. Does this mean you have no sense for physics? Not at all. Most questions were deliberately chosen to illustrate those aspects of physics which seem contrary to casual surmise. Revising ideas, even in the privacy of your own mind, is not painless work.** But in doing so you will revisit some of the problems that haunted the minds of Archimedes, Galileo, Newton, Maxwell, and Einstein.* The physics you cover here in hours took them centuries to master. Your hours of thinking will be a rewarding experience. Enjoy!

Lewis Epstein

The year 2000 imagined in 1900

Slide from L01



At School

The "Surfer Analogy" for time management

Slide from L01



- Please do use office hours to your advantage. We are here to help (and understand where problems are)

A quick primer on OOP (Object-Oriented Programming)

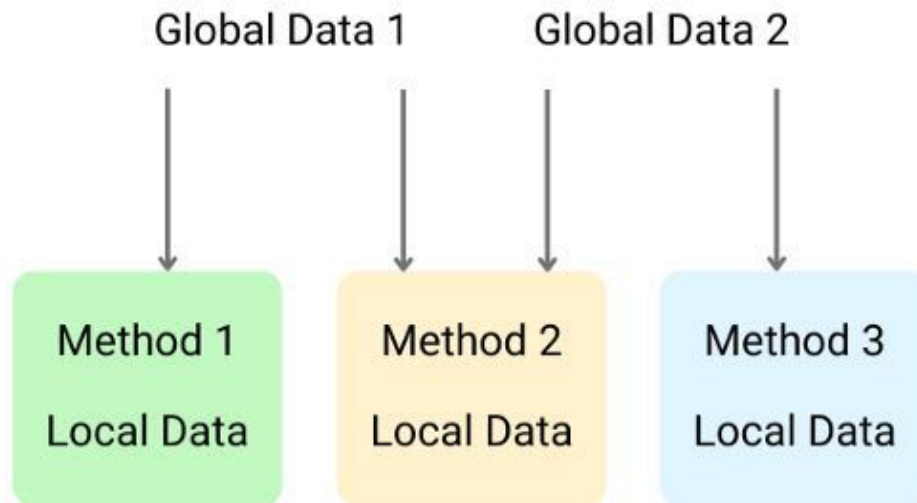
If you used Python, you have already used OOP

```
>>> a = 123
>>> type(a)
<class 'int'>
>>> b = "abc"
>>> type(b)
<class 'str'>
>>> c = [1, 2]
>>> type(c)
<class 'list'>
```

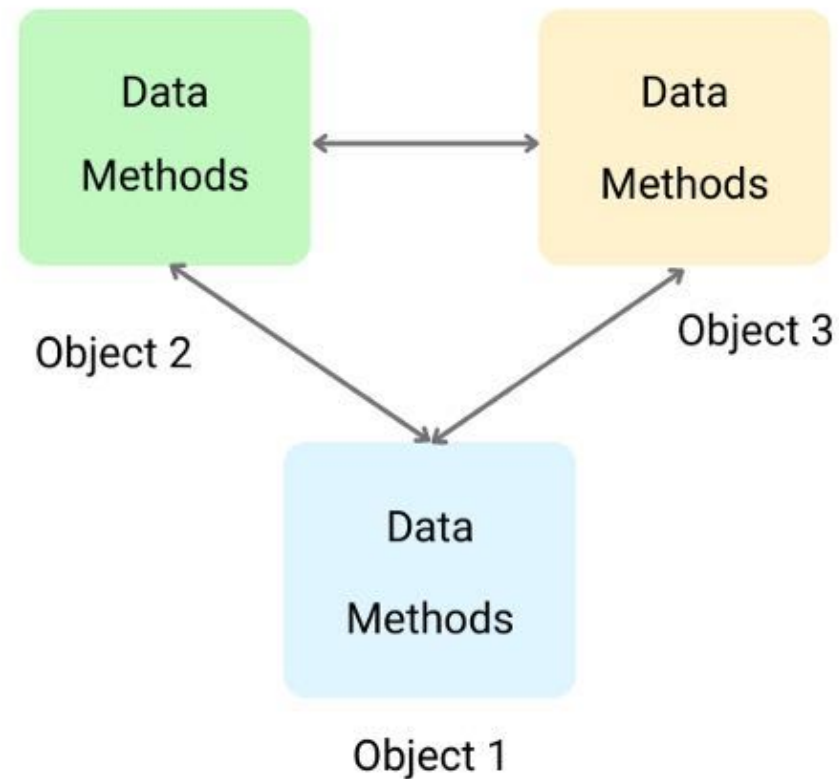
All values in
Python are
objects 😊

Procedural vs Object-Oriented Programming (OOP)

Procedural Programming



Object Oriented Programming



enjoyalgorithms.com

Key features of OOP

Encapsulation

Containing information in an object, exposing only selected information

Inheritance

Allows classes to inherit features of other classes.

OOPs features in C++

Abstraction

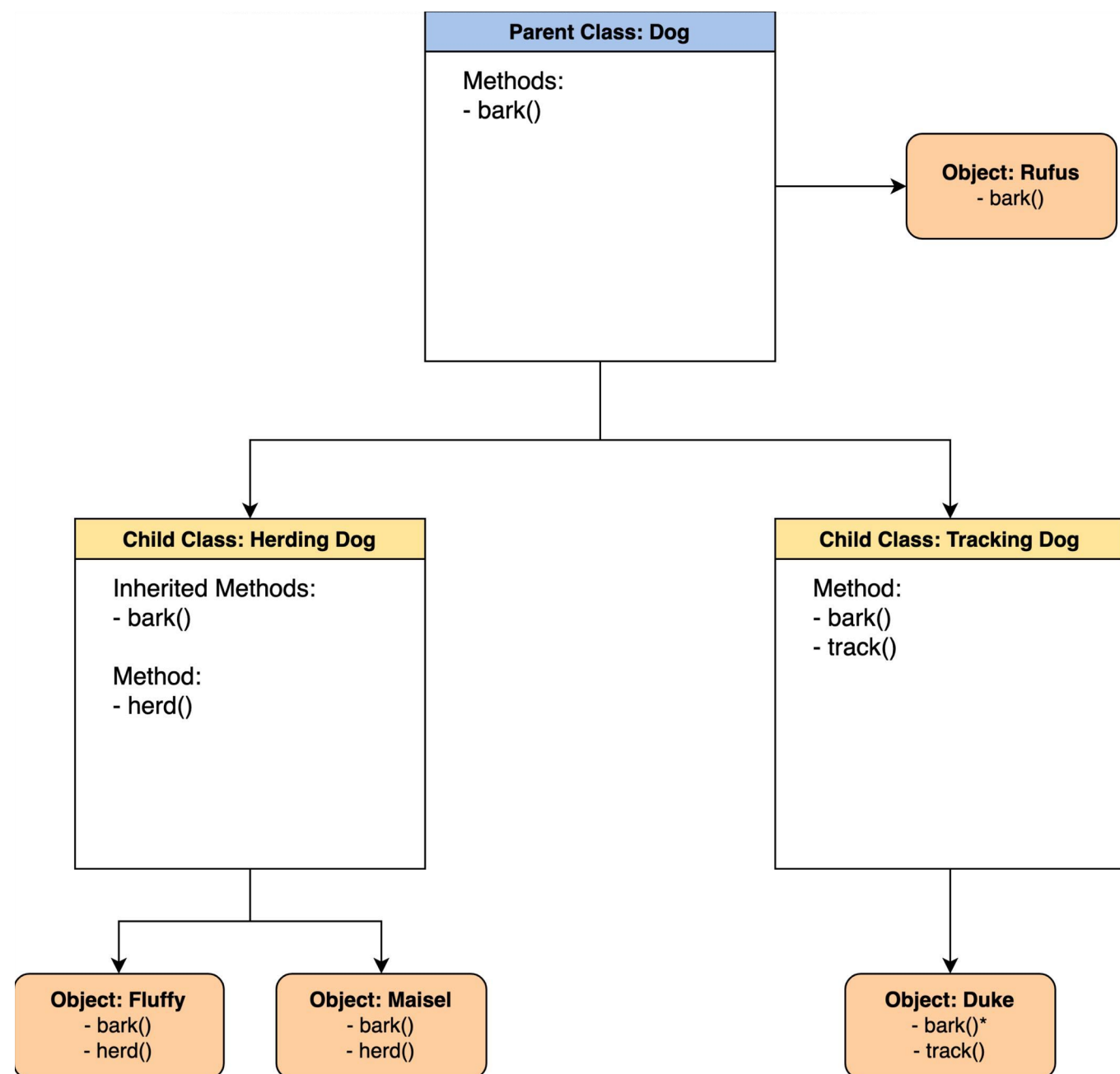
Exposing essential details and hiding implementation details of a class

Polymorphism

Allows us to perform a single action in different ways

enjoyalgorithms.com

Class vs Instance (= Object)



Inheritance in Python (Object-Oriented Programming)

Python

```
class Dog:
    species = "Canis familiaris"

    def __init__(self, name, age):
        self.name = name
        self.age = age

    def __str__(self):
        return f"{self.name} is {self.age} years old"

    def speak(self, sound):
        return f"{self.name} says {sound}"
```

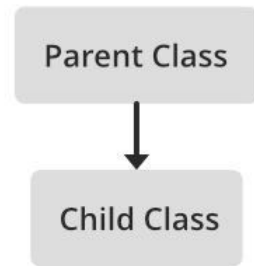
Python

```
class JackRussellTerrier(Dog):
    def speak(self, sound="Arf"):
        return f"{self.name} says {sound}"
```

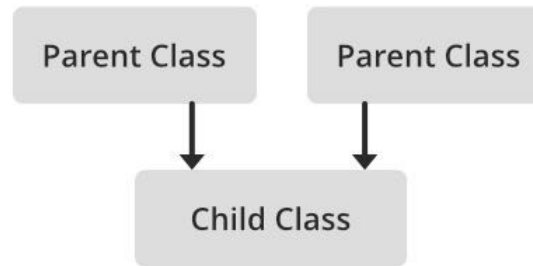
Python

```
>>> miles = JackRussellTerrier("Miles", 4)
>>> miles.speak()
'Miles says Arf'
```

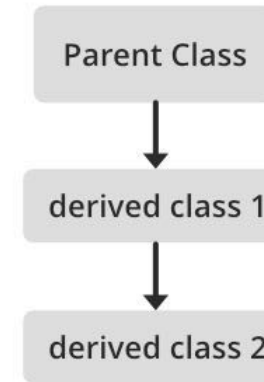
Inheritance in Object-Oriented Programming



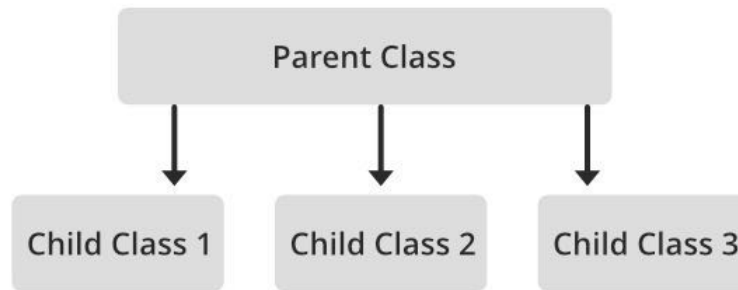
Single inheritance



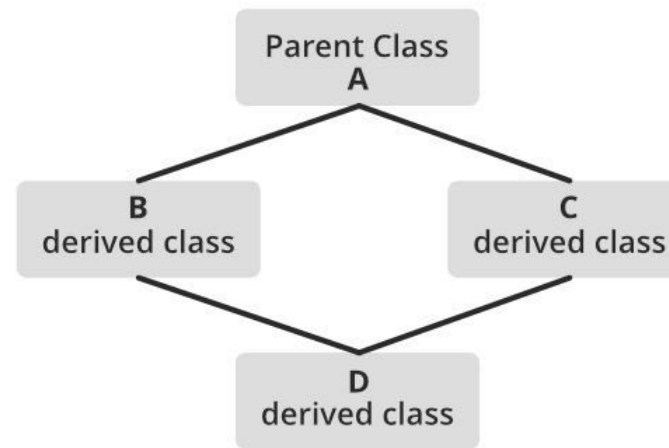
Multiple inheritance



Multilevel inheritance



Hierarchical inheritance

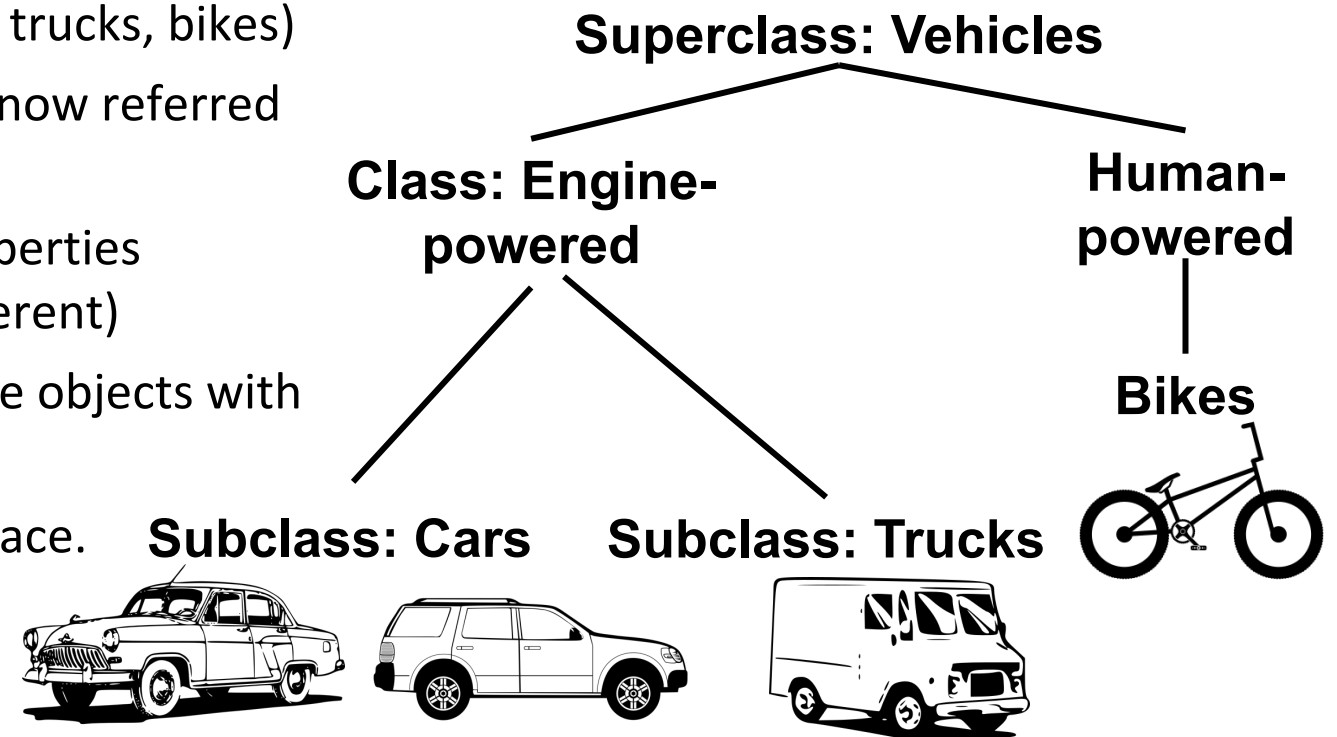


Hybrid Class

Inheritance in OOP and connection to Enhanced ERDs

- Objects are analogous to real-world objects (e.g. a vehicle). Compare to entities.
- **Objects have properties** (e.g. number of wheels, max speed)
- Related objects are grouped into **classes** (i.e. vehicles). Compare to entity sets.

- Can also be grouped into sub-classes (e.g. cars, trucks, bikes)
- **Subclass “inherits” properties** of parent class (now referred to as the **superclass**)
- Subclass can be modified to have different properties from parent class (i.e. they are similar, but different)
- Reduces code **duplication**: Coders can produce objects with reduced codebase.
- Future changes only need to be made in one place.



EER (Enhanced ER)

Subtypes in ER diagrams

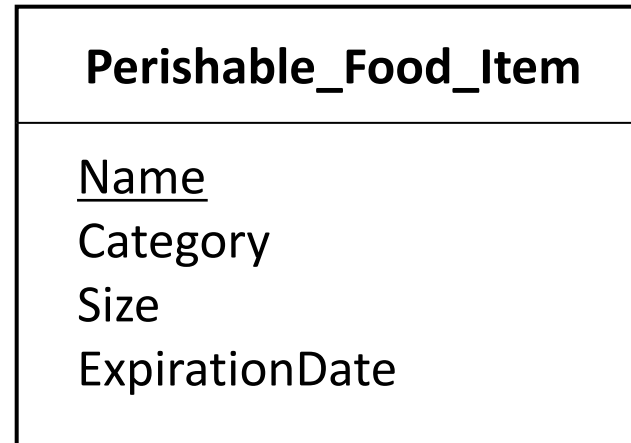
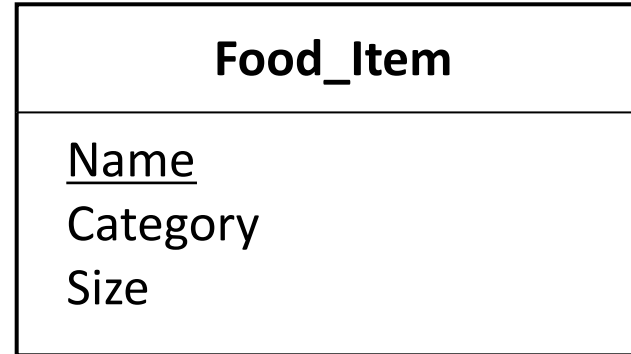
Common Modeling Dilemma

- You want to represent multiple things in your data model that are similar, but not exactly the same
 - Many shared attributes (usually also including keys)
 - Each specific thing has some unique attributes
- Two options with traditional E-R diagrams:
 1. Create a new entity type for each thing with its own attributes
 2. Create one generic entity type with all of the possible attributes for any of the things

Modeling Example

- Model food stocked by a grocery store
- Two basic types of food:
 - Food_Item
 - Name
 - Category
 - Size
 - Perishable_Food_Item
 - Name
 - Category
 - Size
 - ExpirationDate

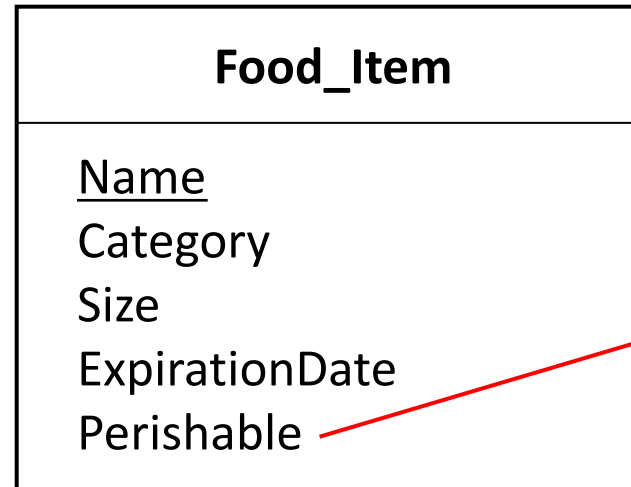
Option 1: Two Entity Types



Modeling Example

- Model food stocked by a grocery store
- Two basic types of food:
 - Food_Item
 - Name
 - Category
 - Size
 - Perishable_Food_Item
 - Name
 - Category
 - Size
 - ExpirationDate

Option 2: One Entity Type



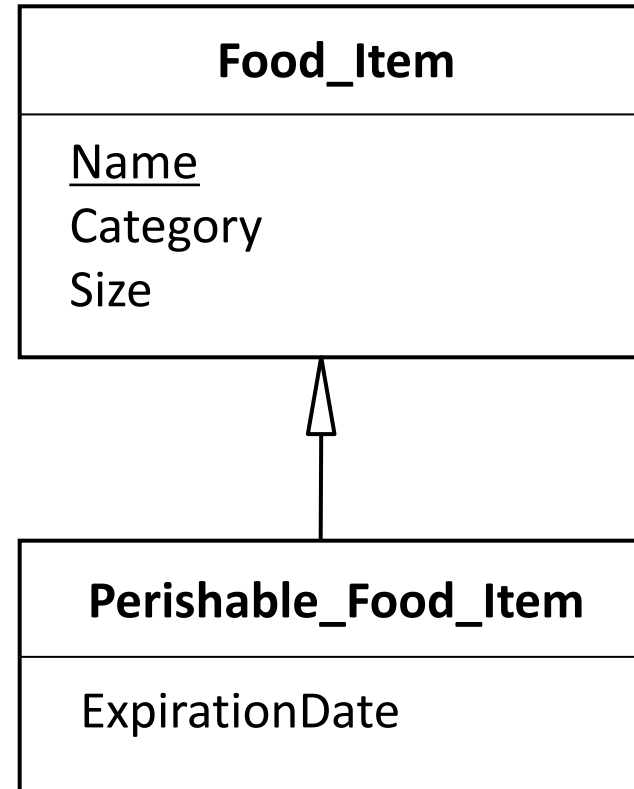
Optional attribute that is not relevant for many instances

Modeling Example: Supertypes and Subtypes

- Food_Item is a supertype
- Perishable_Food_Item is a subtype
- Expiration_Date is an attribute unique to the Perishable_Food_Item subtype

*Notice: we never repeat attributes in the subentities, not even the key!
-> That is Inheritance, and similar to subtyping in UML or OO design*

Option 3: Super/Sub Types



Generalization and Specialization

- Generalization: The process of defining a more general entity type from a set of more specialized entity types.
 - Also known as the bottom-up approach
- Specialization: The process of defining one or more subtypes of the supertype, and forming supertype/subtype relationships.
 - Also known as the top-down approach

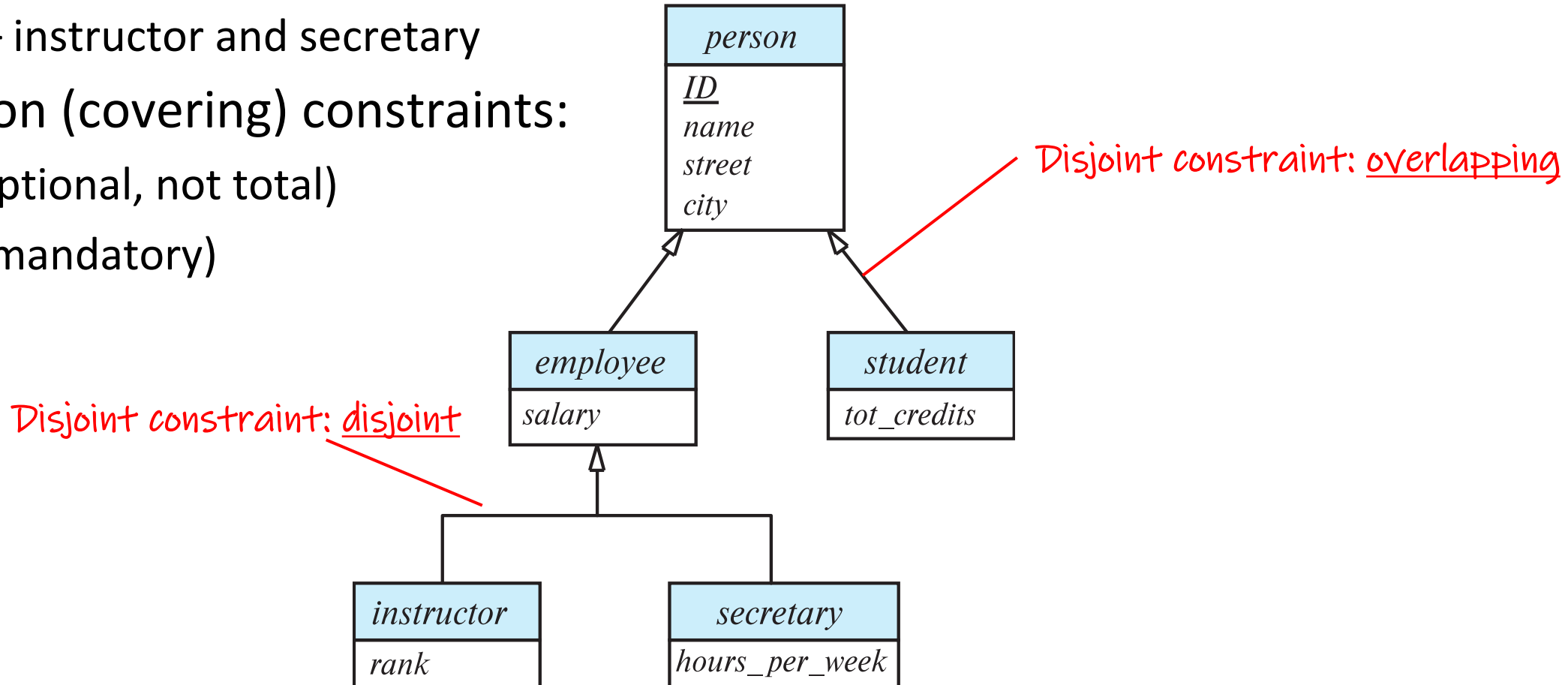
Specialization Example

1. Disjointness constraints:

- Overlapping – employee and student
- Disjoint – instructor and secretary

2. Participation (covering) constraints:

- partial (optional, not total)
- total (or mandatory)



Notations for specialization ("ISA relationship")

11/16/2022

Participation (or covering) constraint
(optional=partial | mandatory=total)

Partial-overlapping
optional

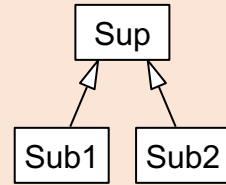
Partial-disjoint

Total-overlapping
mandatory

Total-disjoint

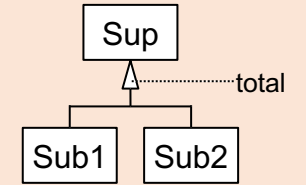
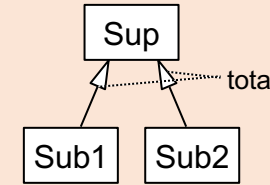
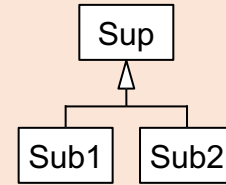
[Silberschatz+'20]

Please use the notation from our textbook even though I will use slides with various notation

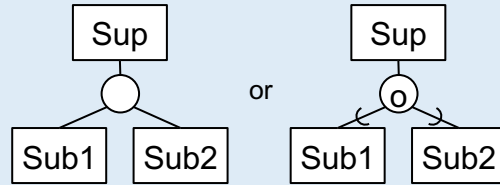


Super entity
(more generalized,
higher-level)

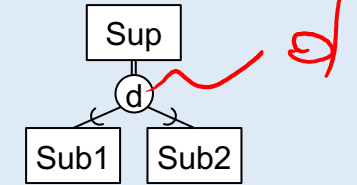
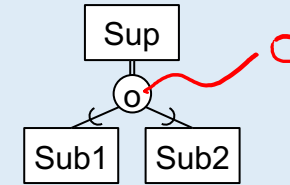
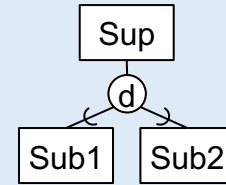
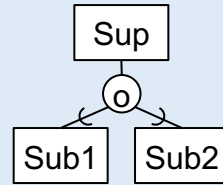
Sub entity
(more specialized,
lower-level)



[Elmasri+'15],
[Hoffer+'10]

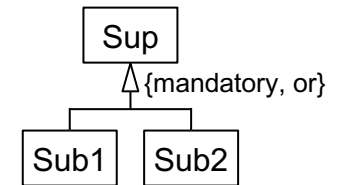
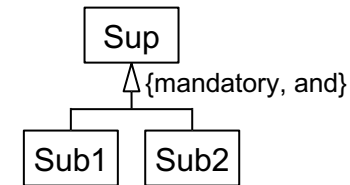
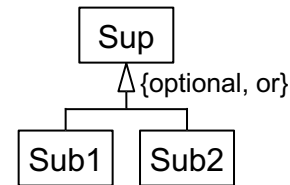
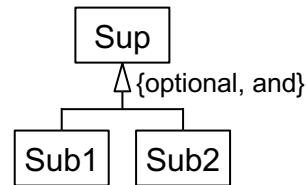


or



UML
[Connolly+'15]

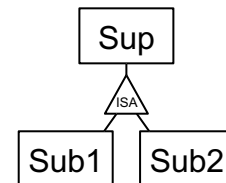
Overlap (or disjoint) constraints
(or=disjoint | and=overlapping)



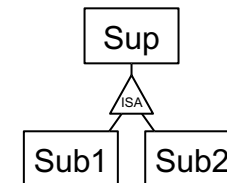
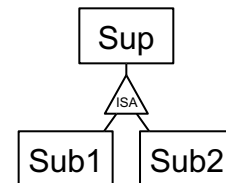
[Stanford book'03]

also by Gradiance

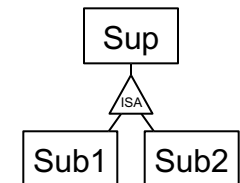
[Cow book'03]



Sub1 overlaps Sub2



Sub1 overlaps Sub2
Sub1 and Sub2 cover Sup



Sub1 and Sub2 cover Sup

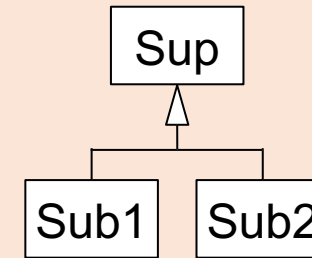
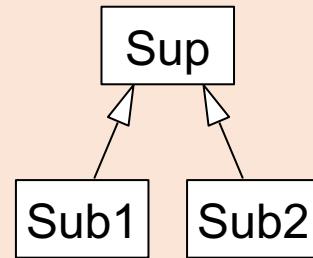
1. Disjointness Constraints

- Describes whether an instance of a supertype may simultaneously be a member of two (or more) subtypes

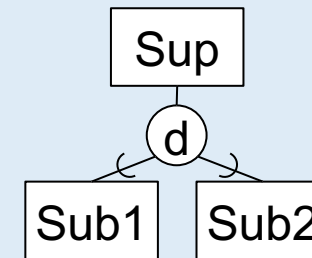
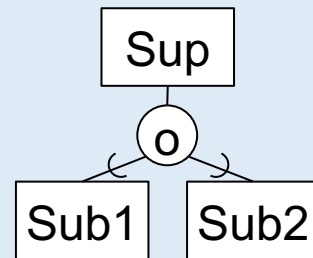
Overlap (nondisjoint) Rule: An instance of the supertype could be more than one of the subtypes

Disjoint Rule: An instance of the supertype can be only ONE of the subtypes

[Silberschatz+'20]



[Elmasri+'15],
[Hoffer+'10]



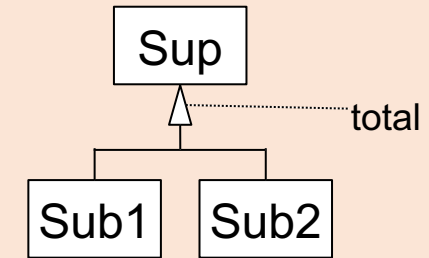
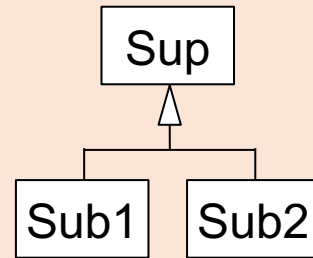
2. Participation Constraints

- Describes whether every instance of a superclass must also be a member of at least one subtype

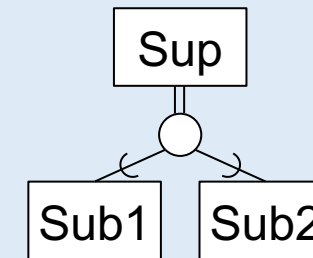
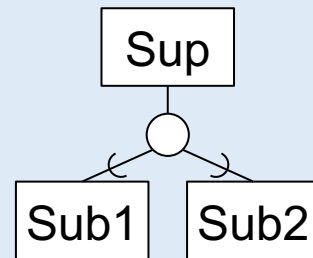
Partial (optional) specialization: no

Total (mandatory) specialization: yes

[Silberschatz+'20]



[Elmasri+'15],
[Hoffer+'10]





Exercise: Generalize With Sub/Super Types

- Three entity types: CAR, TRUCK, and MOTORCYCLE

Car
<u>vehicle id</u>
price
engine_displacement
Vehicle_name
make
model
no_of_passengers

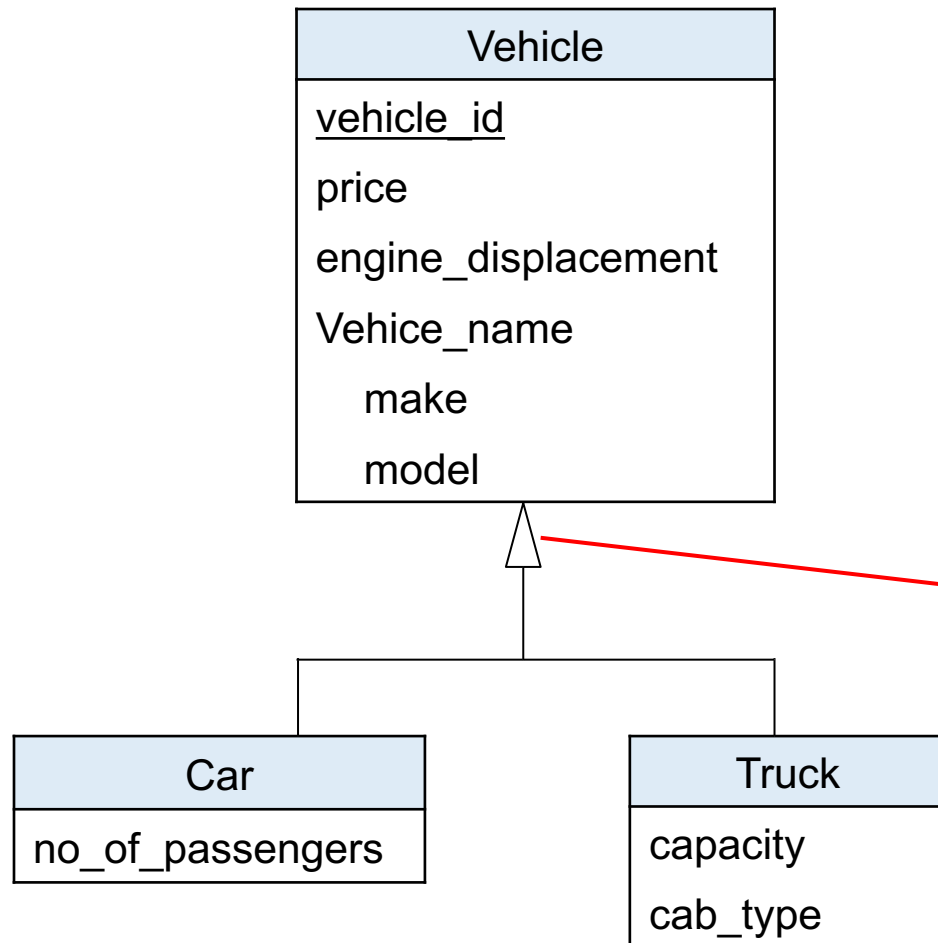
Truck
<u>vehicle id</u>
price
engine_displacement
Vehicle_name
make
model
capacity
cab_type

Motorcycle
<u>vehicle id</u>
price
engine_displacement
Vehicle_name
make
model

*How to use
subtyping to
create a more
general solution*

?

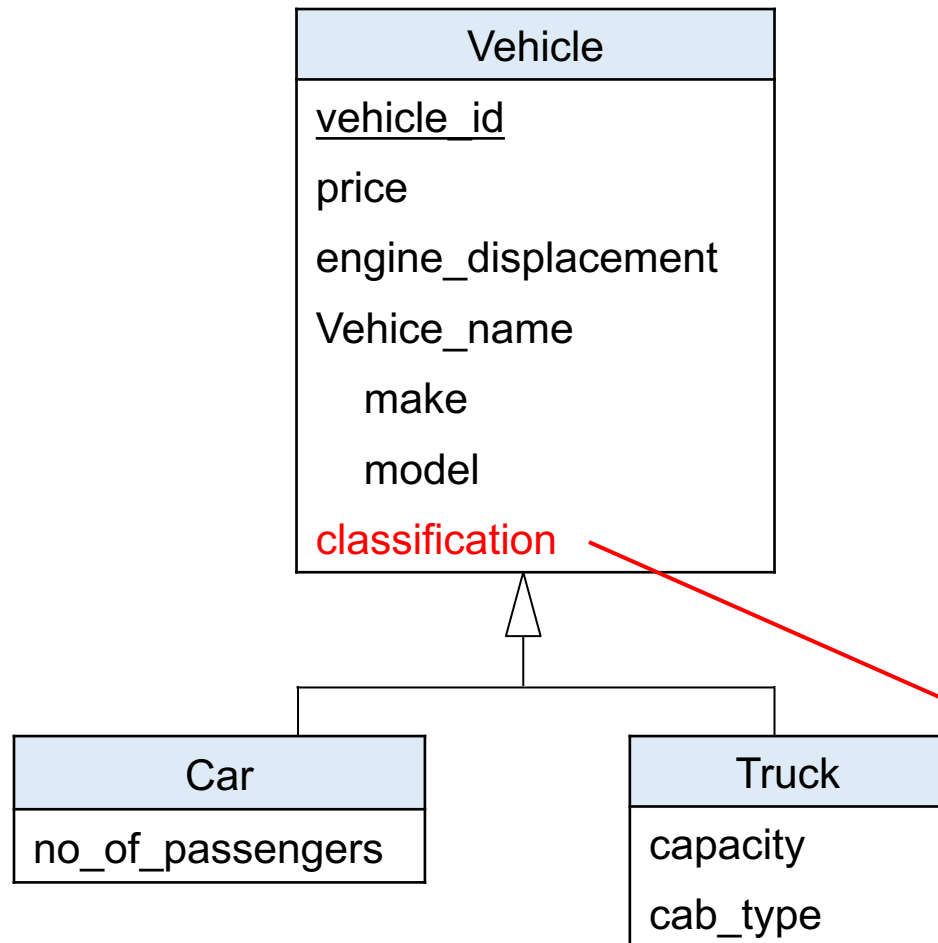
Solution: Generalize With Sub/Super Types



- Put the shared attributes in a supertype
- Note: no subtype for motorcycle necessary, since it has no unique attributes. Thus participation is partial
- A vehicle cannot be a car and a truck at the same time, thus a disjoint specialization

Partial (superclass does not need to be specialized), disjoint (can only be car or truck)

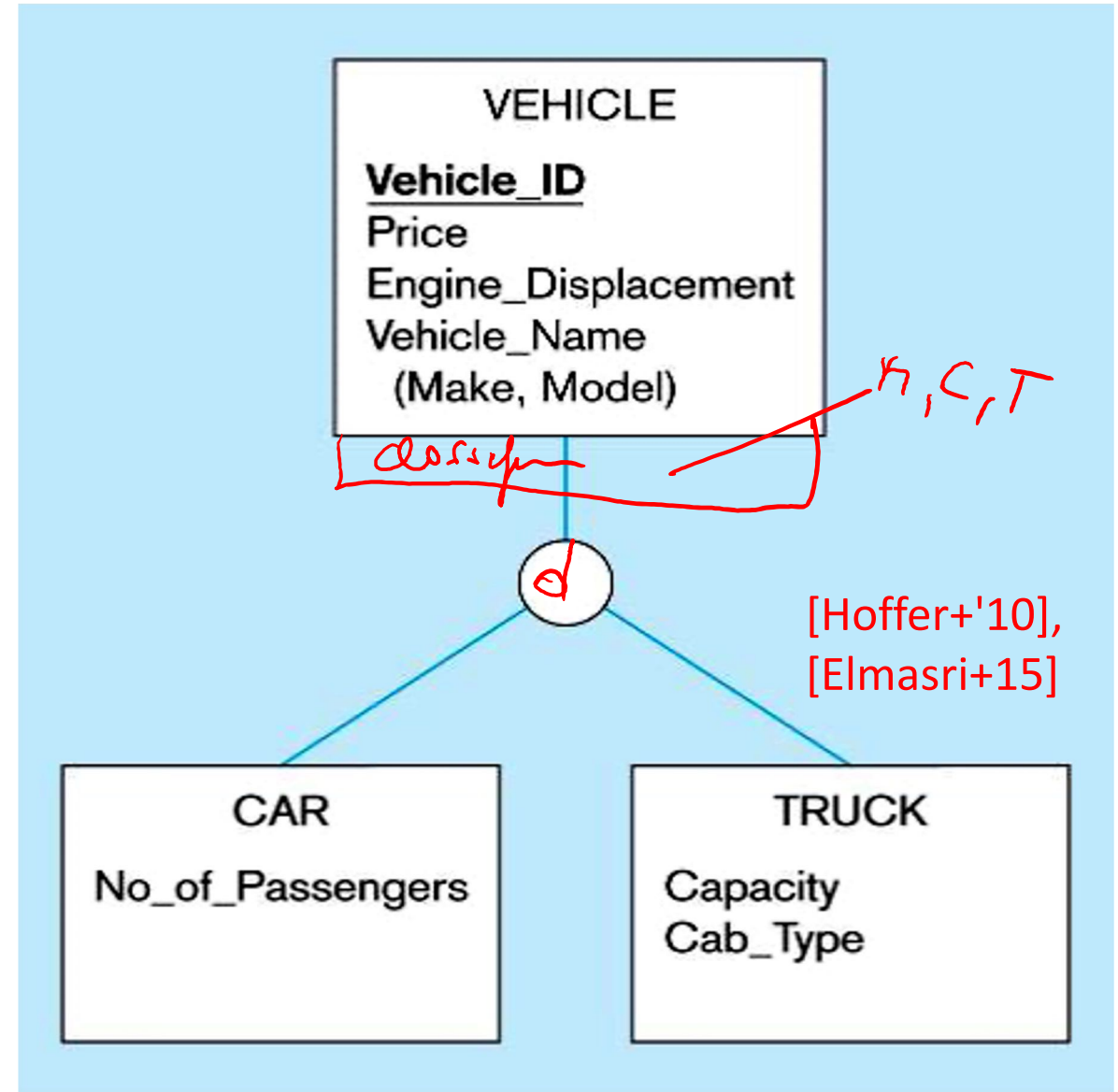
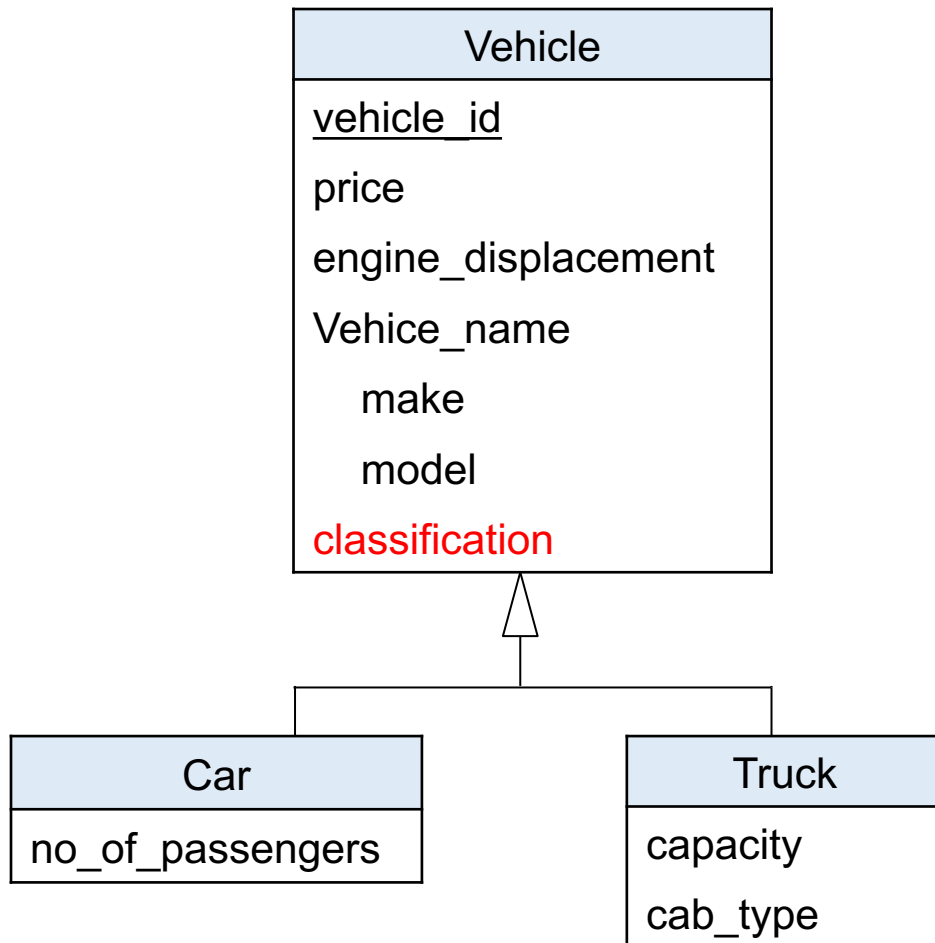
Solution: Generalize With Sub/Super Types



- Put the shared attributes in a supertype
- Note: no subtype for motorcycle necessary, since it has no unique attributes. Thus participation is partial
- A vehicle cannot be a car and a truck at the same time, thus a disjoint specialization

subtype discriminator with 3-valued domain: "M", "C", "T". Enforces that each vehicle (instance of superclass) can be only exactly one of those three.

Solution: Generalize With Sub/Super Types [Hoffer+'10]



Subtype Discriminators

- **Subtype discriminator**: An attribute of the supertype whose values determine the target subtype(s)
 - If disjoint: a simple attribute with alternative values to indicate the possible subtypes
 - if overlapping: a composite attribute whose subparts pertain to different subtypes. Each subpart contains a boolean value to indicate whether or not the instance belongs to the associated subtype



Exercise: Specialize with Sub/Super Types

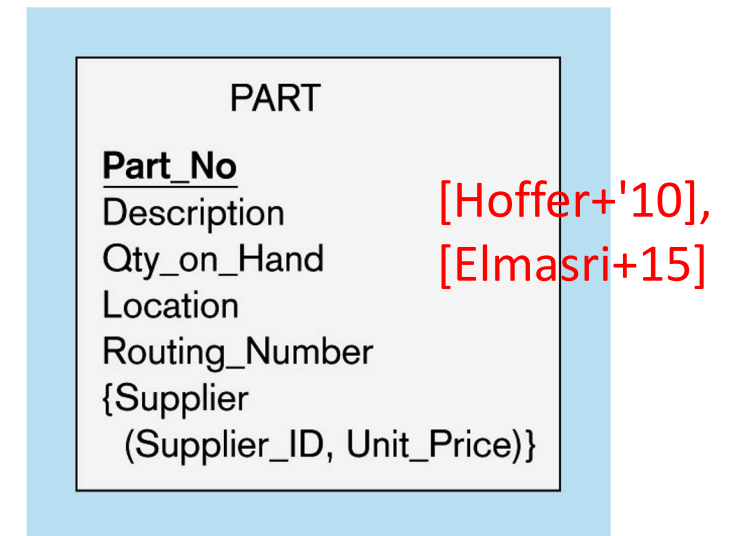
- Assume parts can be both manufactured and purchased (sometimes you make a screw, sometimes you purchase one)
- Assume certain attributes only apply to manufactured parts, others only to purchased parts

Part
<u>part no</u>
description
location
qty_on_hand
routing_number
{supplier
supplier_id
unit_price}

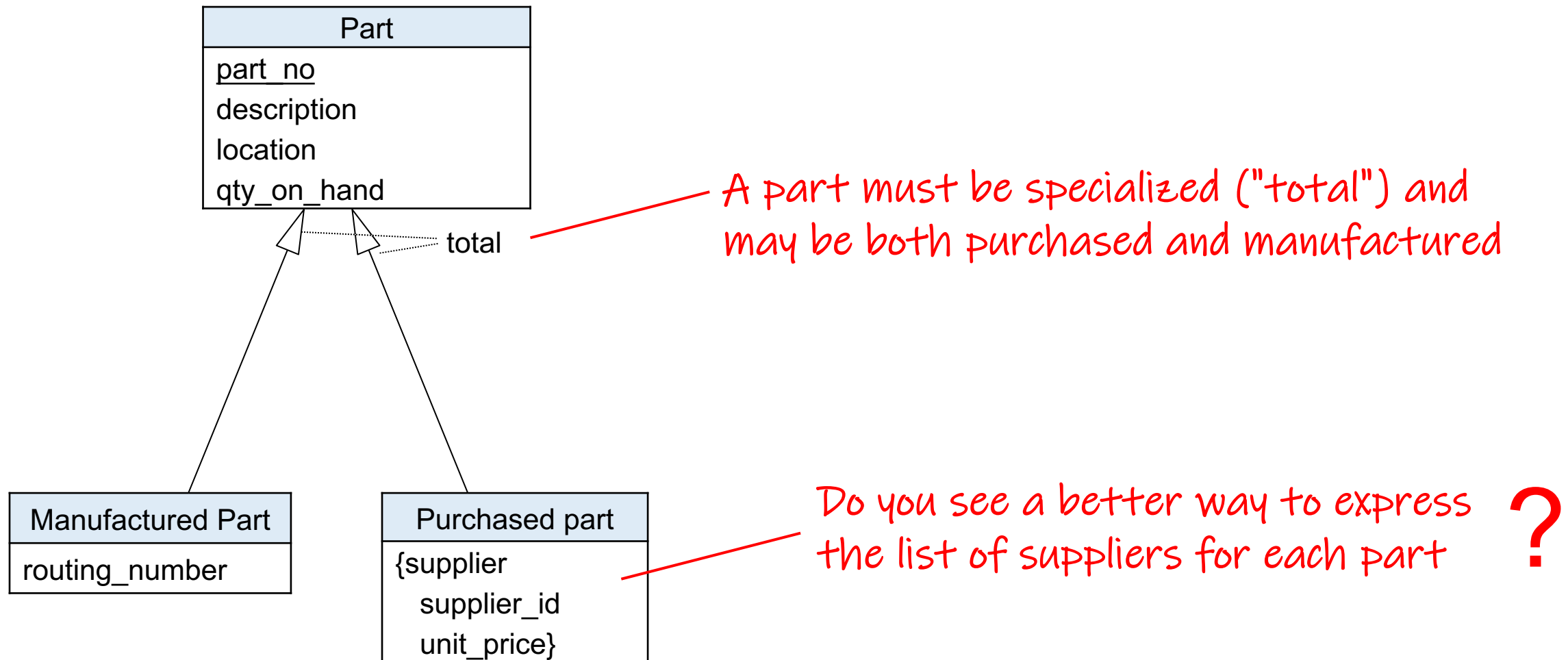
How to specialize with subtypes ?

Applies only to manufactured parts

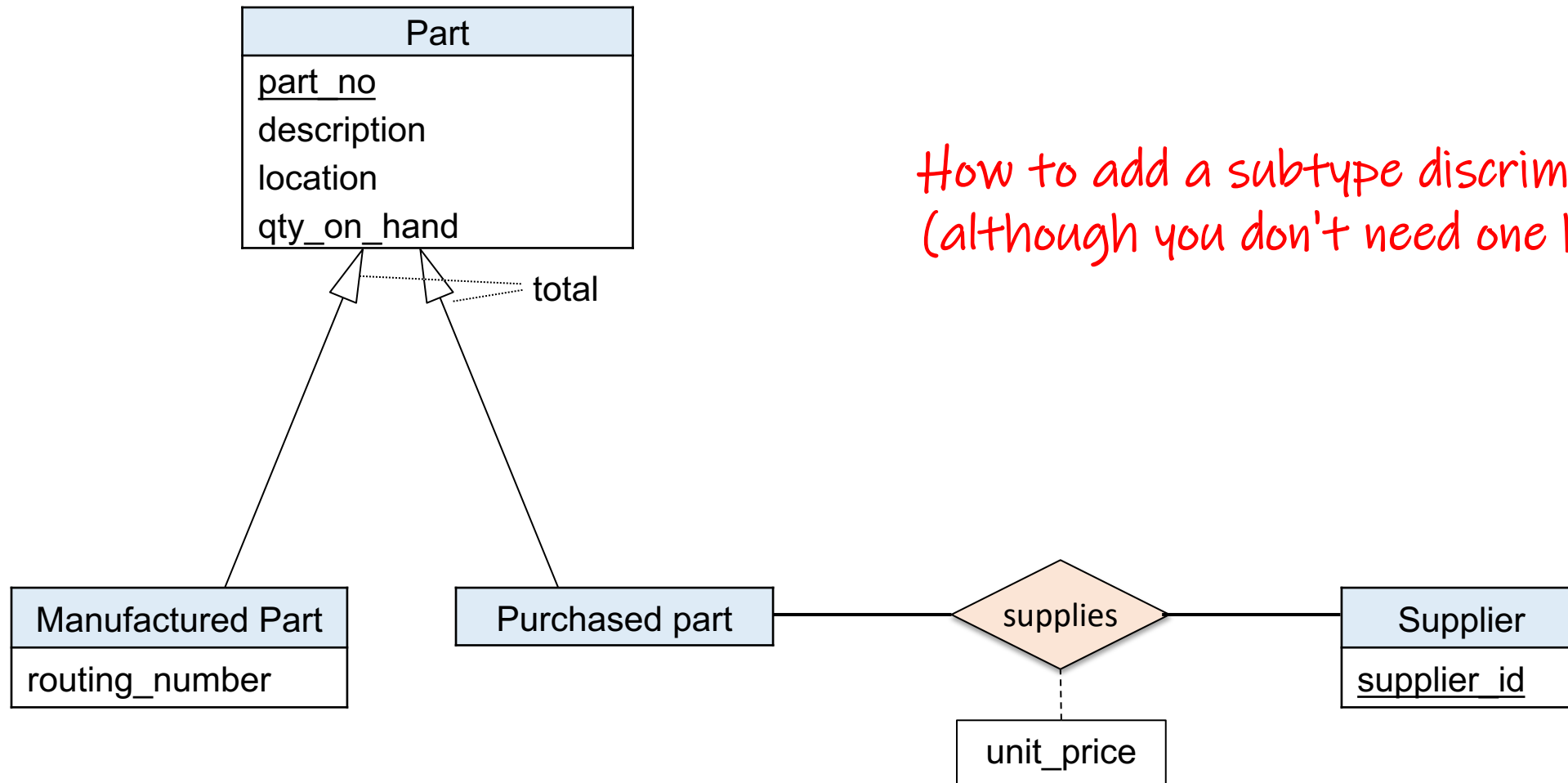
Applies only to purchased parts



Ex: Subtype Discriminator w/ Overlap Constraint



Ex: Subtype Discriminator w/ Overlap Constraint

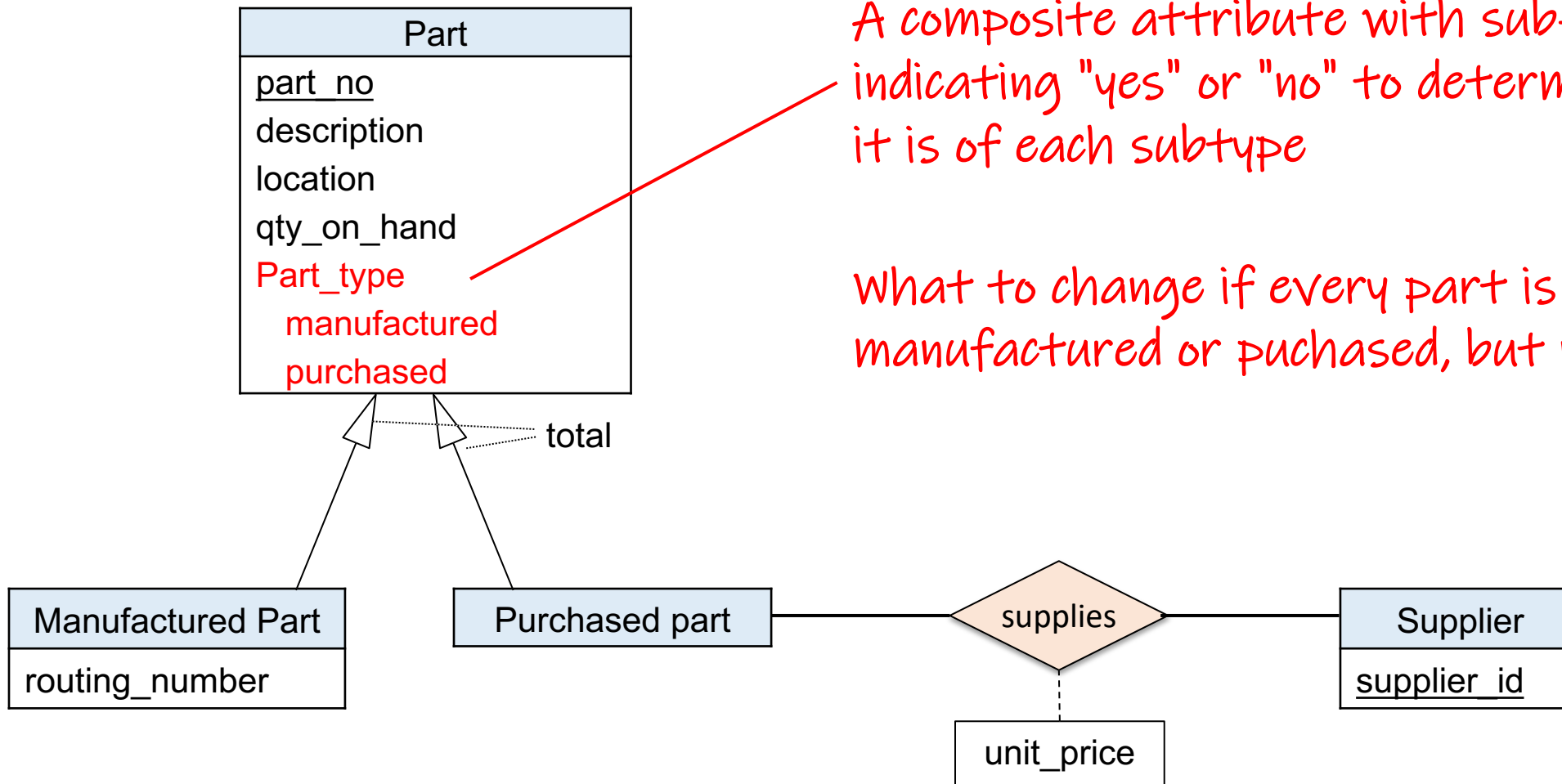


How to add a subtype discriminator
(although you don't need one here)

?

Replace multivalued attribute with a relationship to another entity

Ex: Subtype Discriminator w/ Overlap Constraint

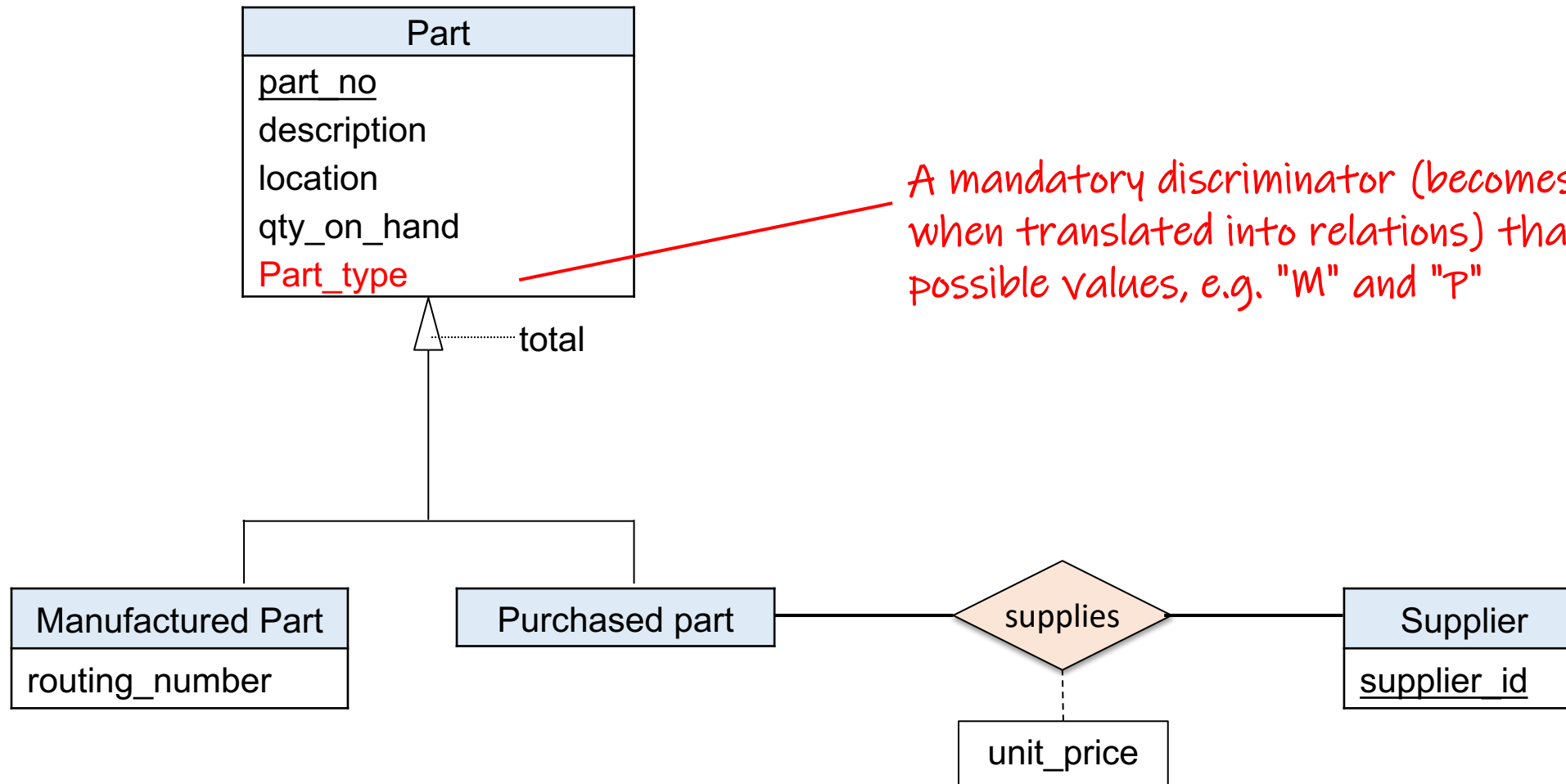


A composite attribute with sub-attributes indicating "yes" or "no" to determine whether it is of each subtype

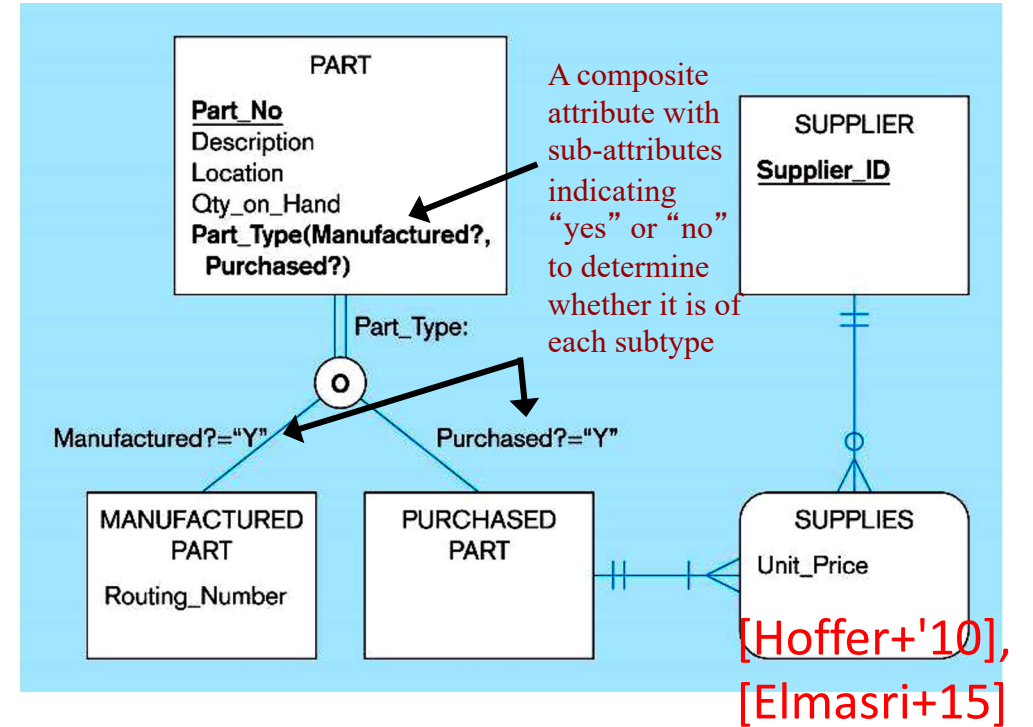
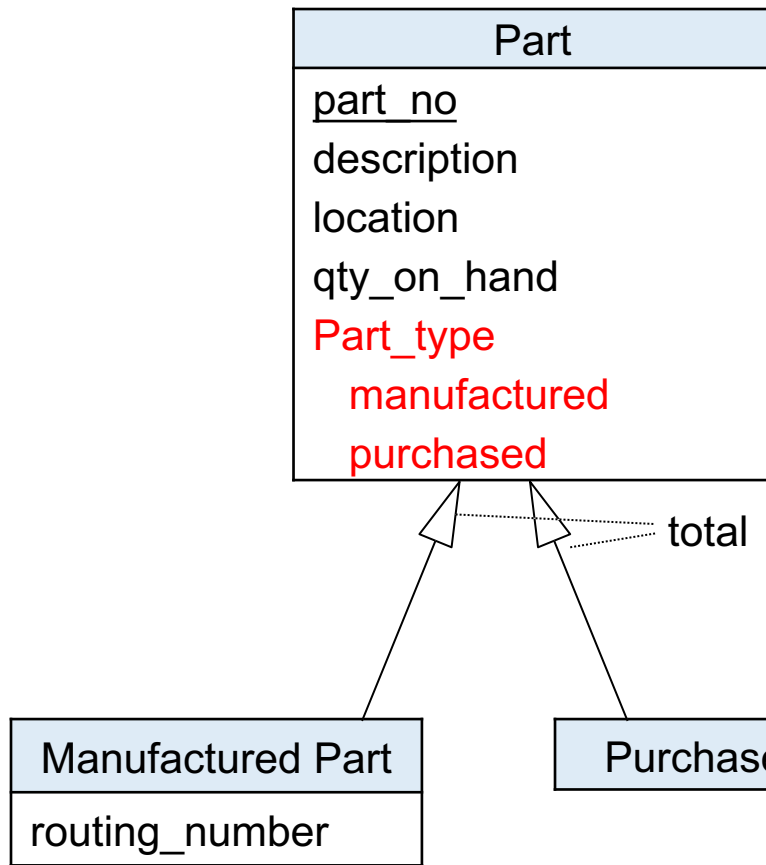
What to change if every part is manufactured or purchased, but not both ?

?

Ex: Subtype Discriminator w/ Overlap Constraint



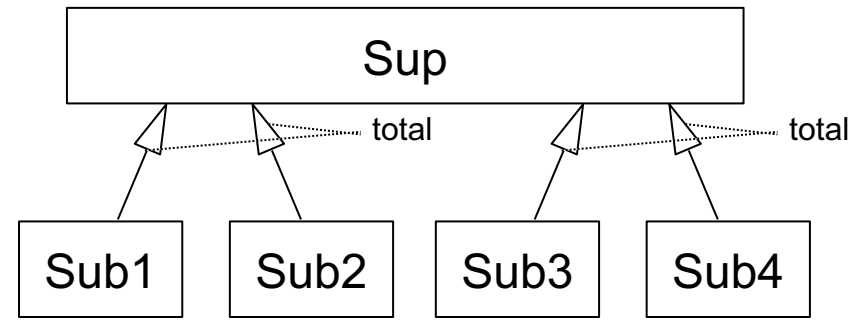
Ex: Subtype Discriminator w/ Overlap Constraint



Quiz: translate from SDK to Hoffer



[Silberschatz+'20]



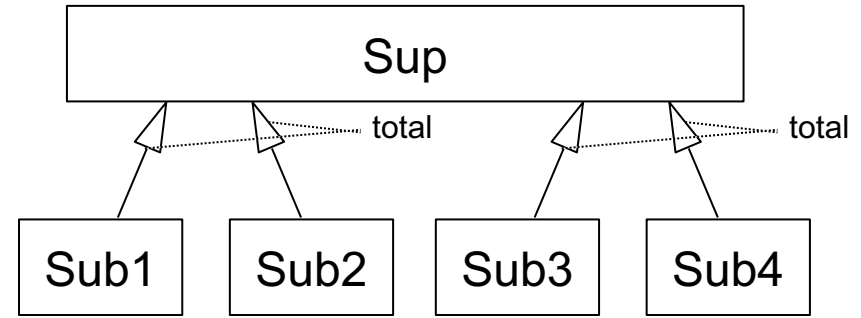
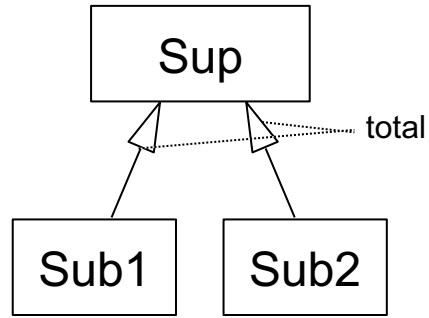
[Elmasri+'15],
[Hoffer+'10]



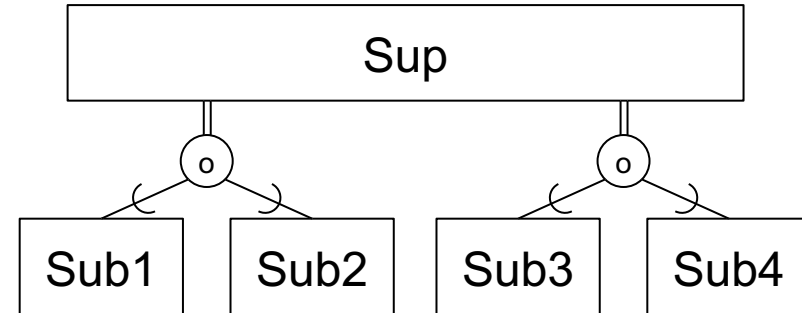
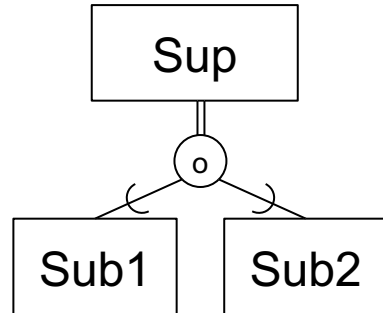
Quiz: translate from SDK to Hoffer



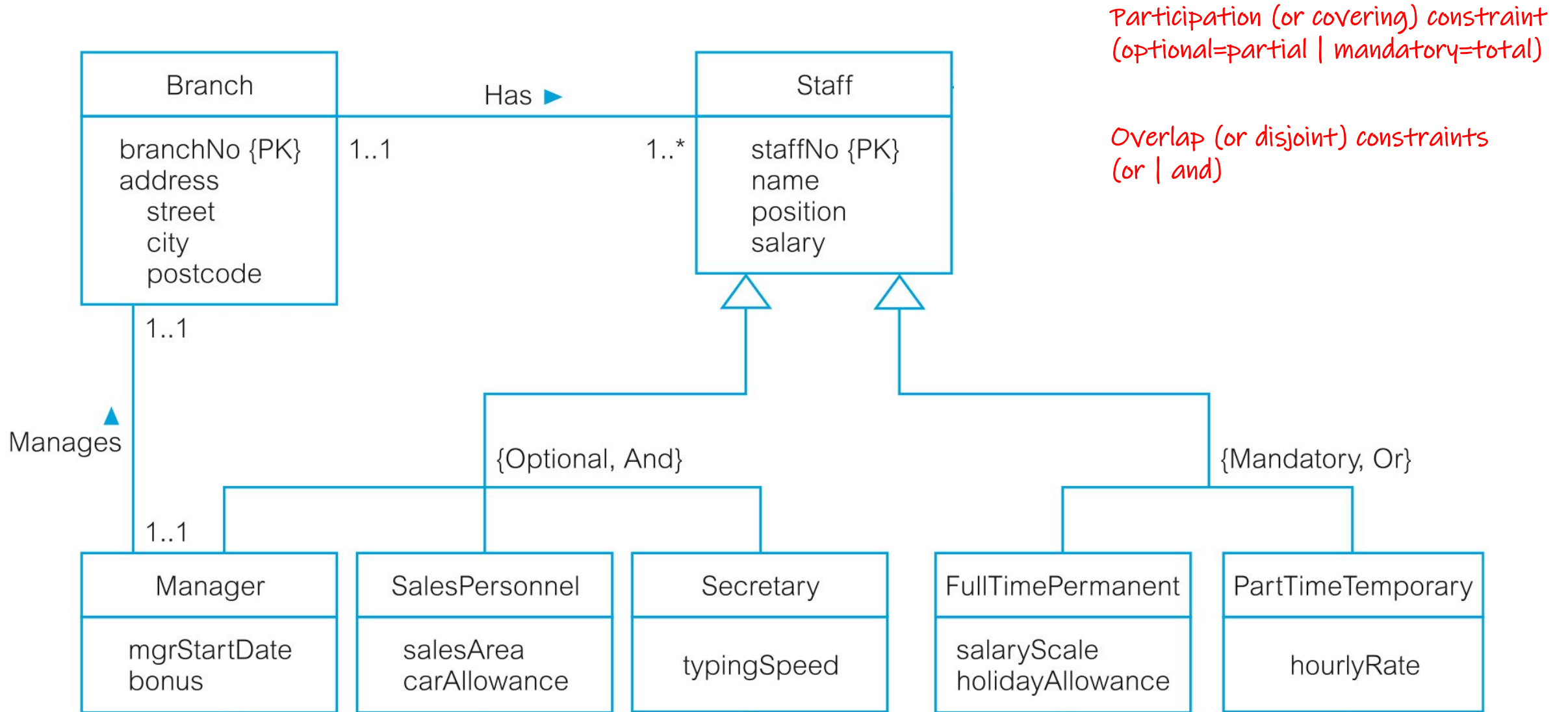
[Silberschatz+'20]



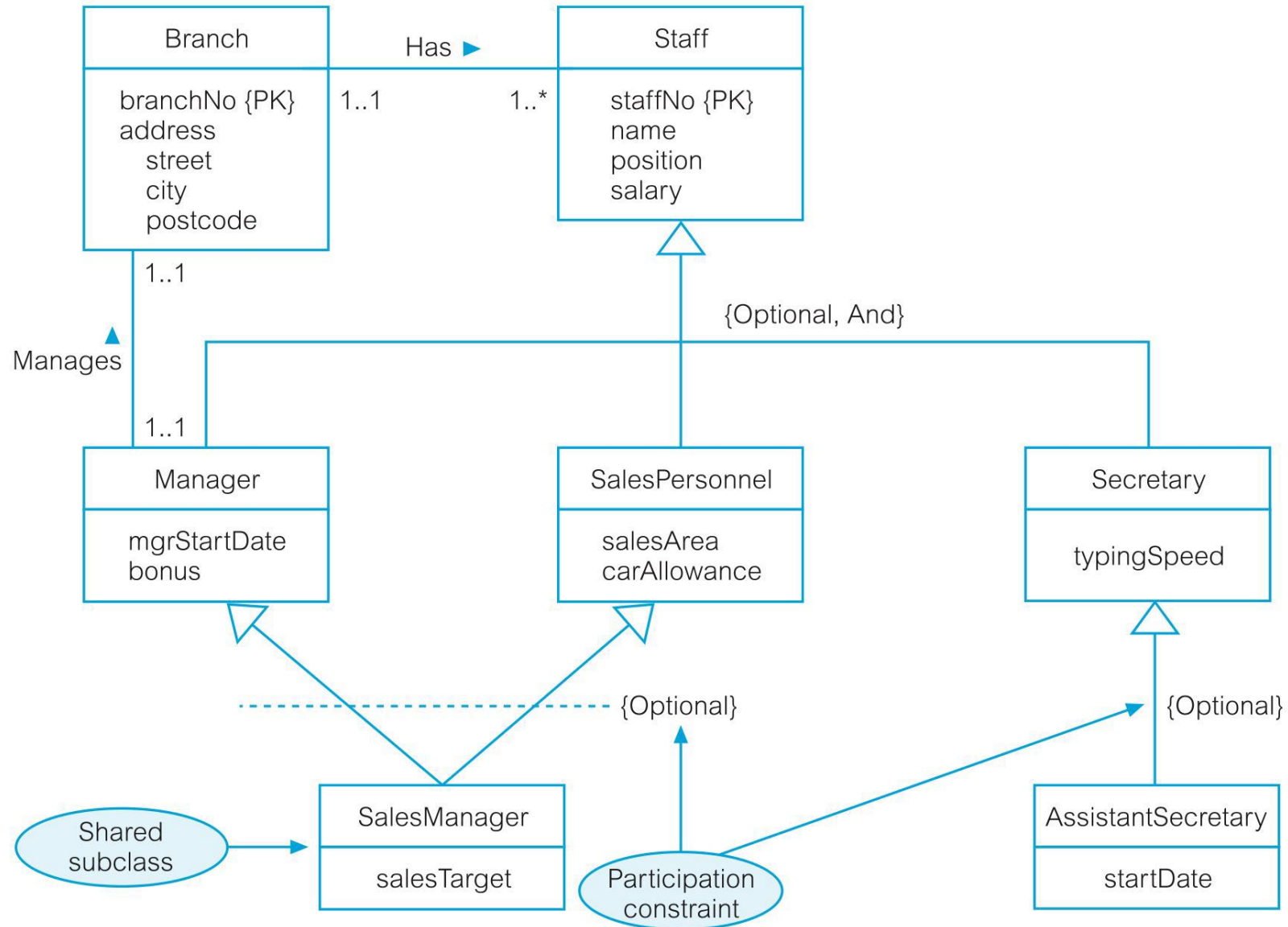
[Elmasri+'15],
[Hoffer+'10]



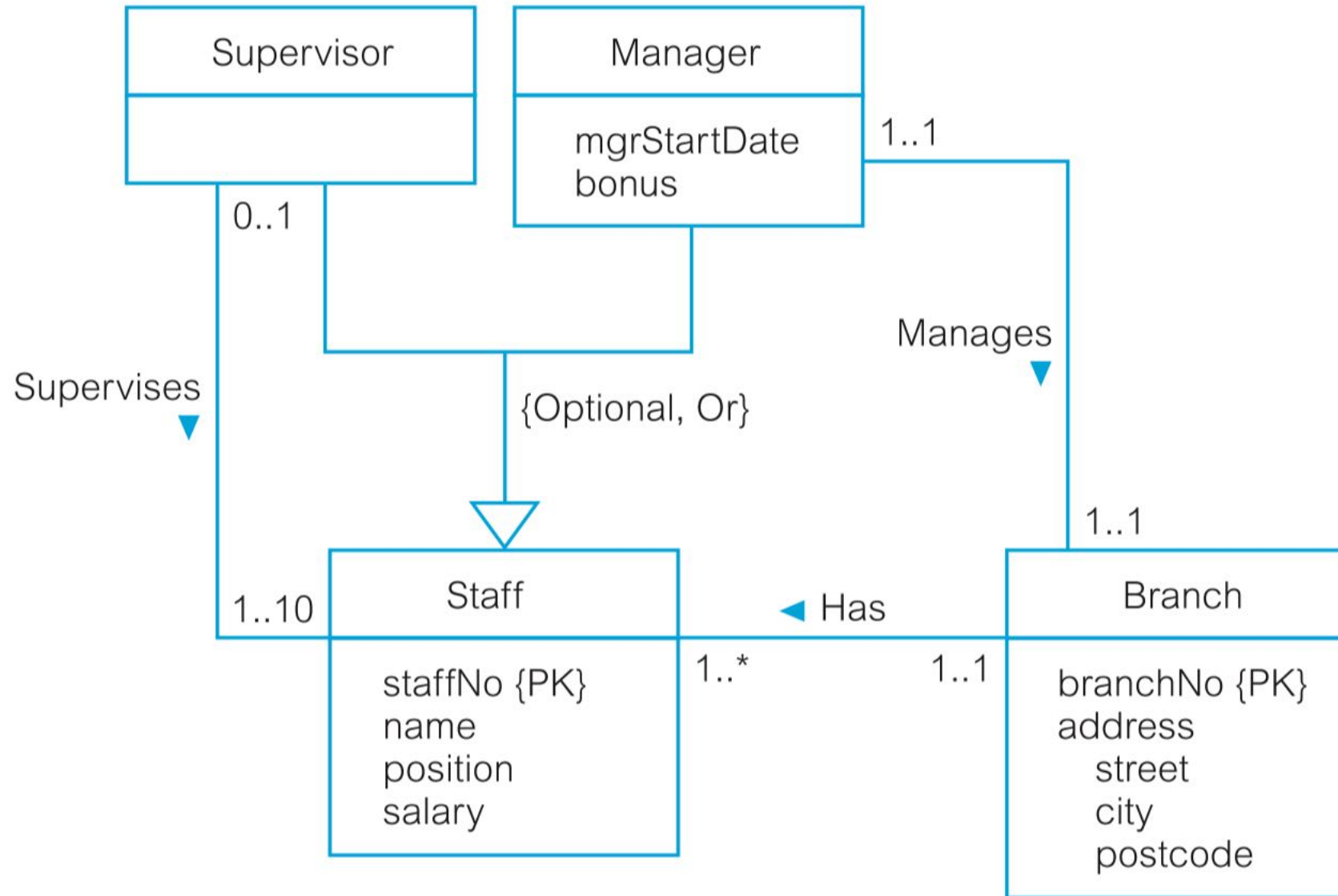
UML notation [Connolly+15]



UML notation [Connolly+15]



UML notation [Connolly+15]



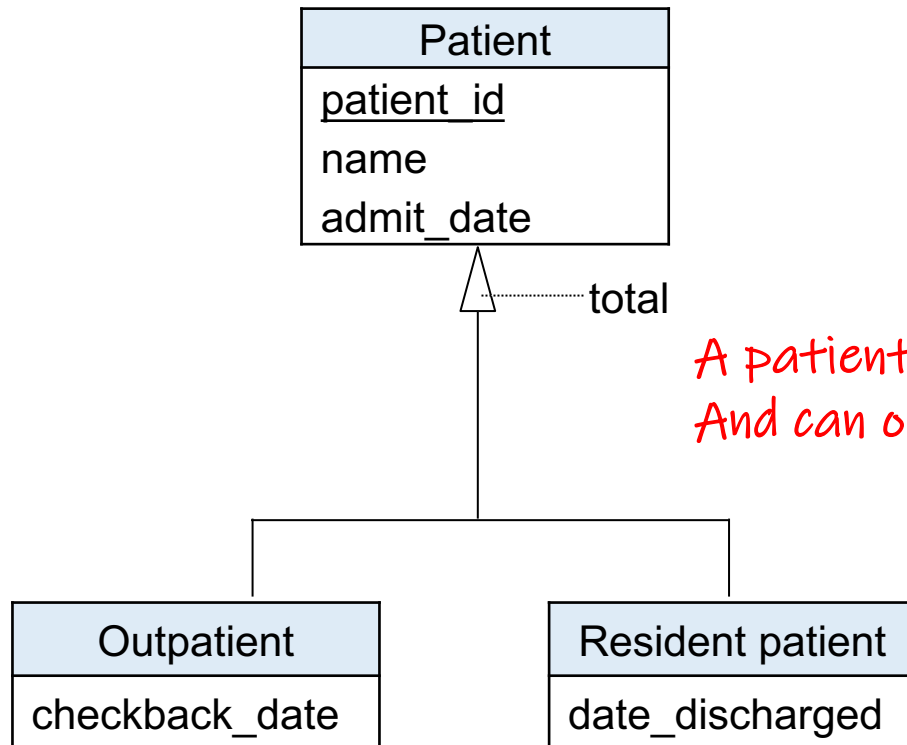
Example: Hospital



Model following situation as ERD

- Patients (id, name, admit date) are either outpatients or resident patients
- Both outpatients and resident patients are cared for by a single responsible physician (id, name)
- Each outpatients has a checkback date. Each resident patient has a discharge date and is assigned to exactly one bed (id)

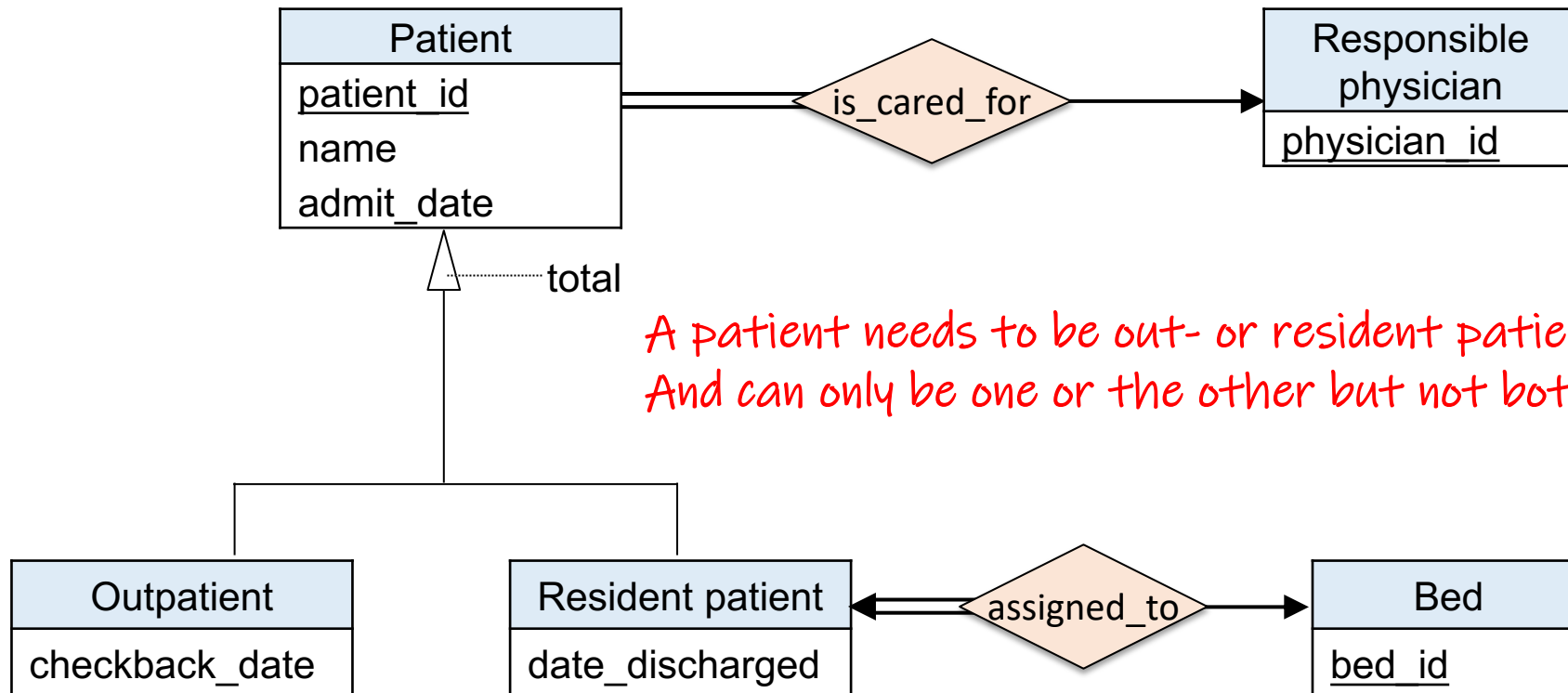
Supertype/Subtype Relationships in a Hospital



*A patient needs to be out- or resident patient (total, mandatory)
And can only be one or the other but not both (disjoint).*

Supertype/Subtype Relationships in a Hospital

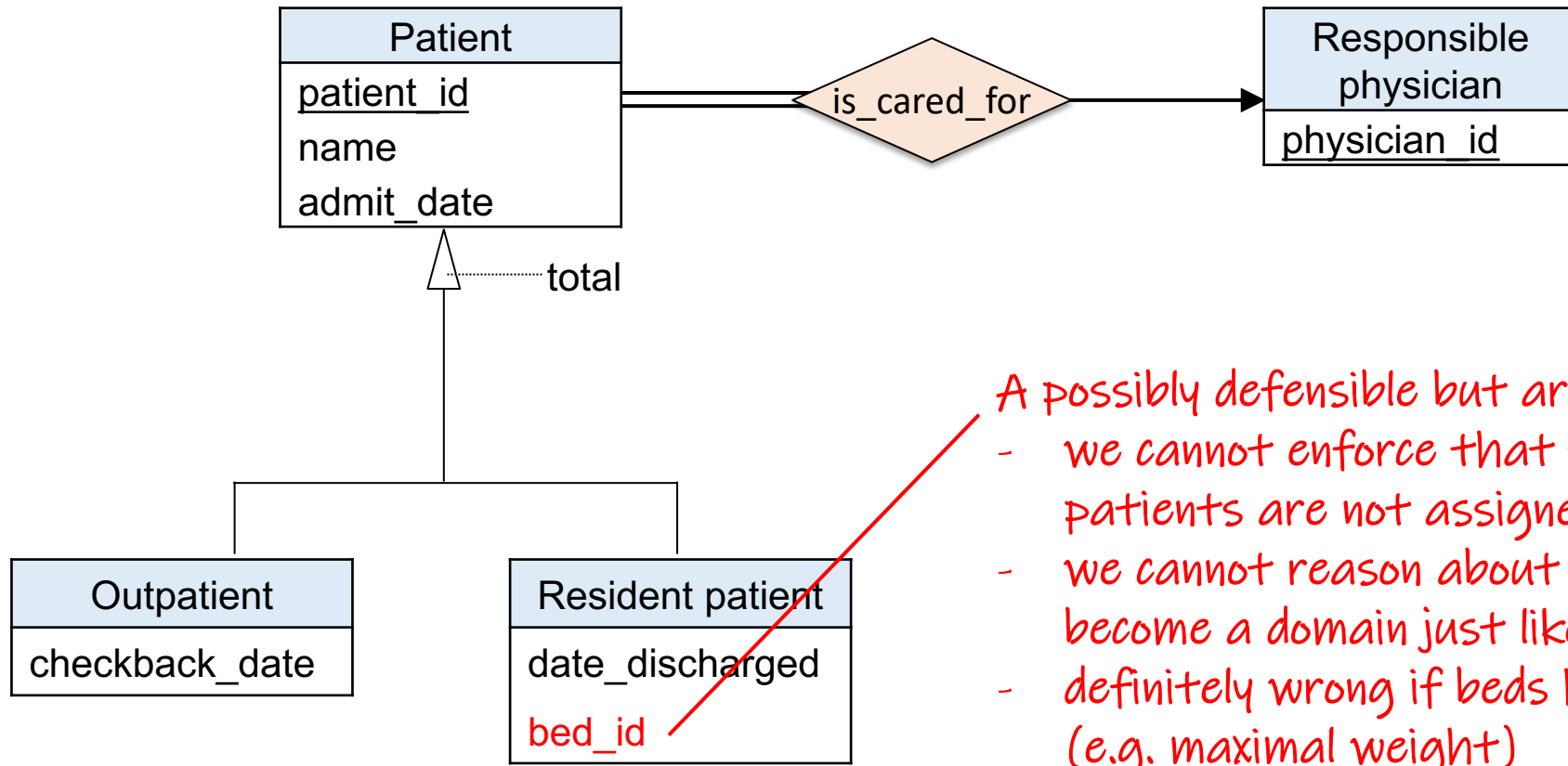
Relationships at the supertype level indicate that all subtypes will participate in the relationship: Both outpatients and resident patients are cared for by a responsible physician



*A patient needs to be out- or resident patient (total, mandatory)
And can only be one or the other but not both (disjoint).*

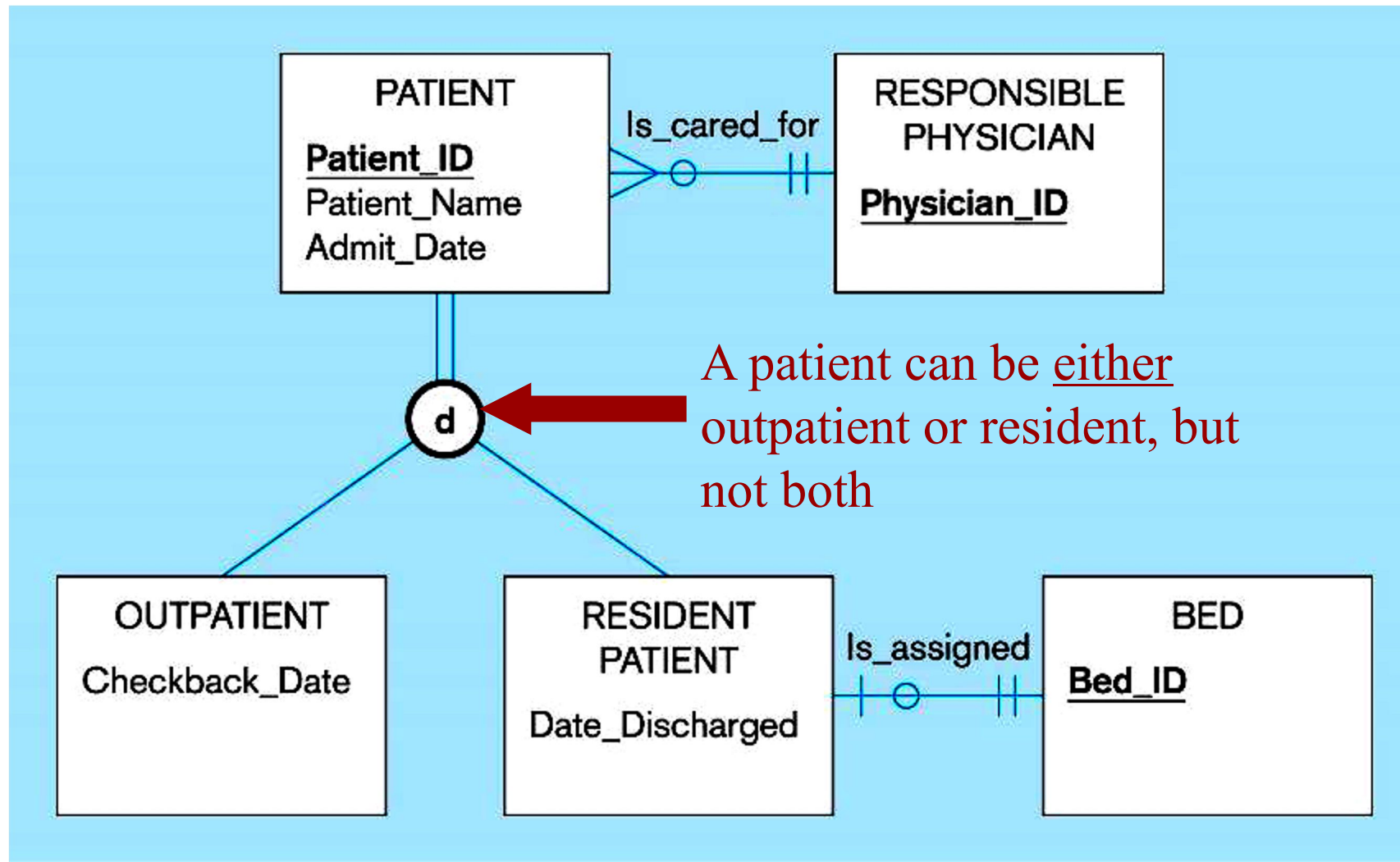
Only resident patients are assigned to a bed

Supertype/Subtype Relationships in a Hospital



- A possibly defensible but arguably inferior way to model.*
- we cannot enforce that two different resident patients are not assigned to the same bed
 - we cannot reason about the set of beds (the bed ids become a domain just like colors of cars)
 - definitely wrong if beds have additional attributes (e.g. maximal weight)

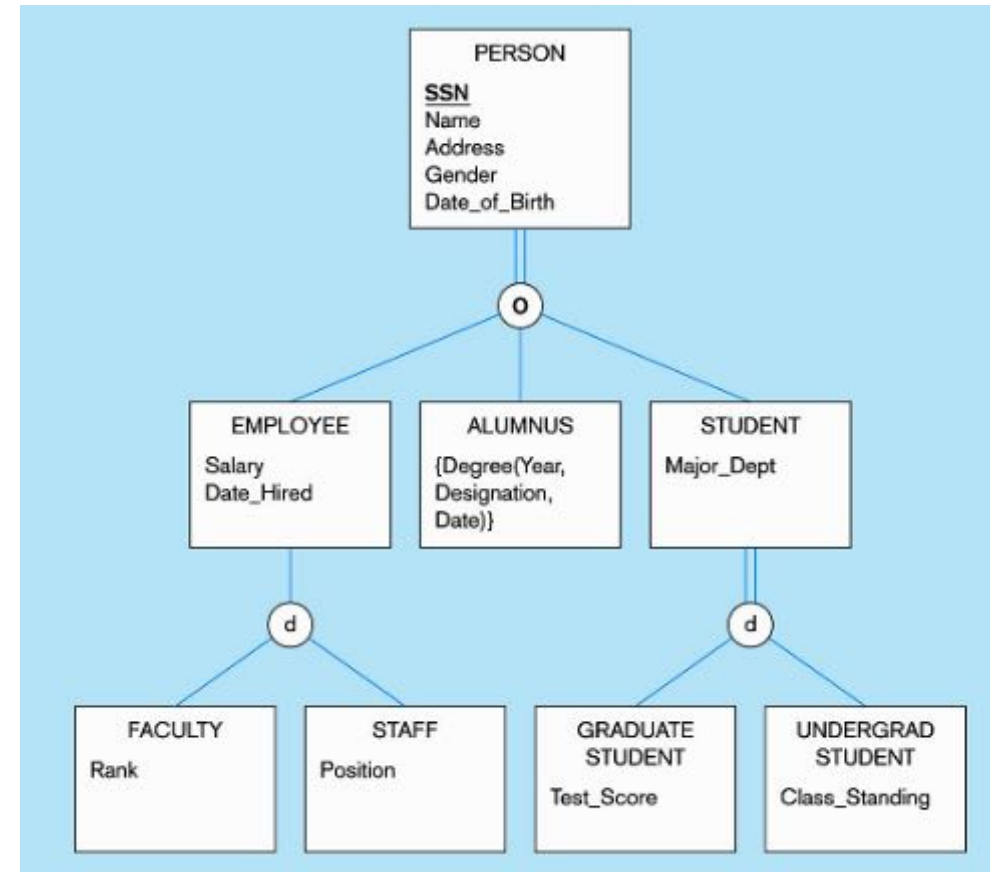
Supertype/Subtype Relationships in a Hospital



[Hoffer+'10],
[Elmasri+15]

Hierarchical Subtyping (1/2)

- An entity can be both a supertype and a subtype
- Subtyping relationships can be arbitrarily deep
 - But don't go crazy, they get confusing pretty quickly
 - Two levels of subtyping is usually the practical maximum (I tend to see only one level)



[Hoffer+'10],
[Elmasri+15]

Hierarchical Subtyping

- An entity can be both a supertype and a subtype
- Subtyping relationships can be arbitrarily deep
 - But don't go crazy, they get confusing pretty quickly
 - Two levels of subtyping is usually the practical maximum (I tend to see only one level)

1. Participation (Completeness Constraints)

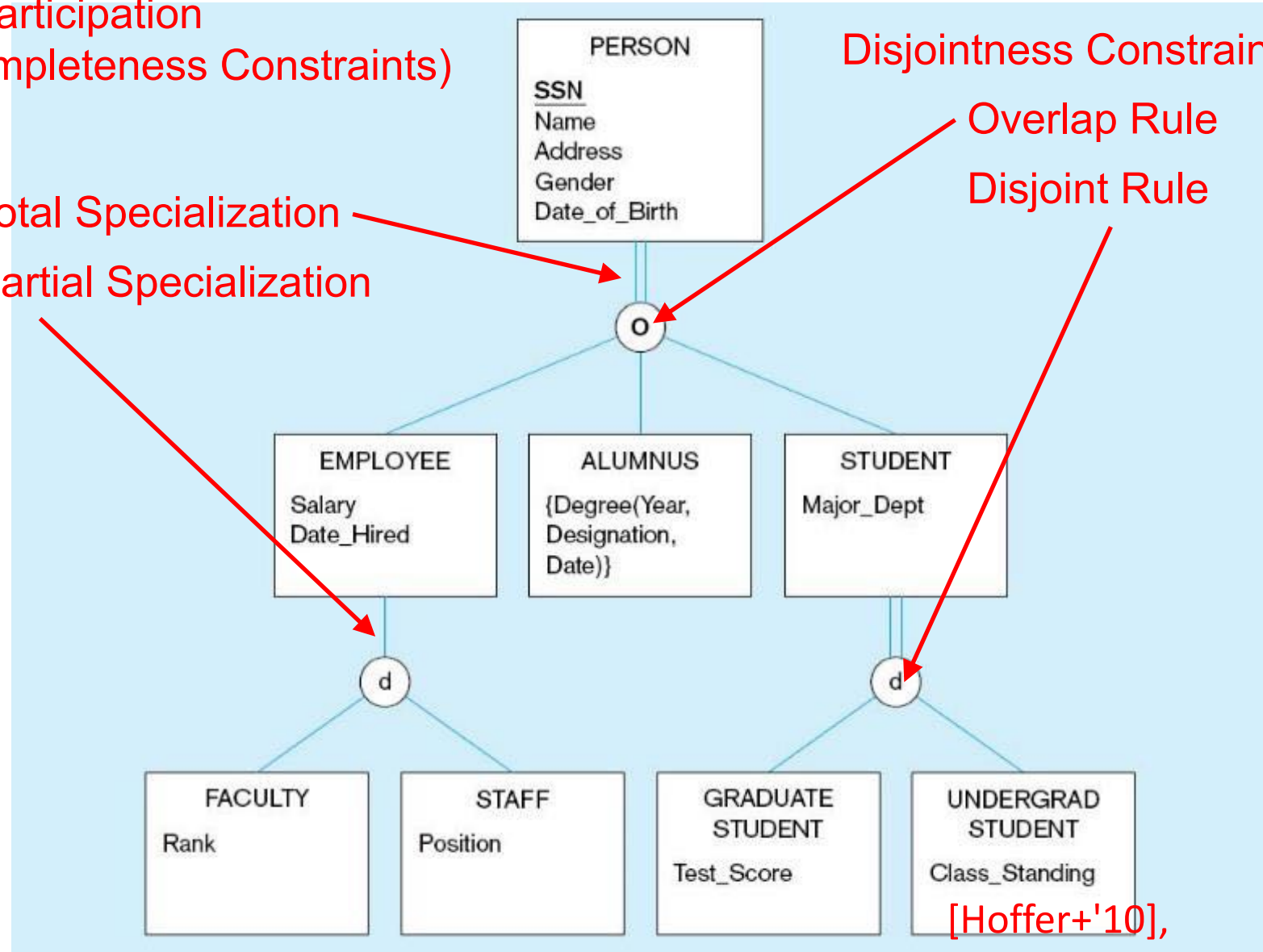
Total Specialization

Partial Specialization

Disjointness Constraints

Overlap Rule

Disjoint Rule



[Hoffer+'10],
[Elmasri+15]