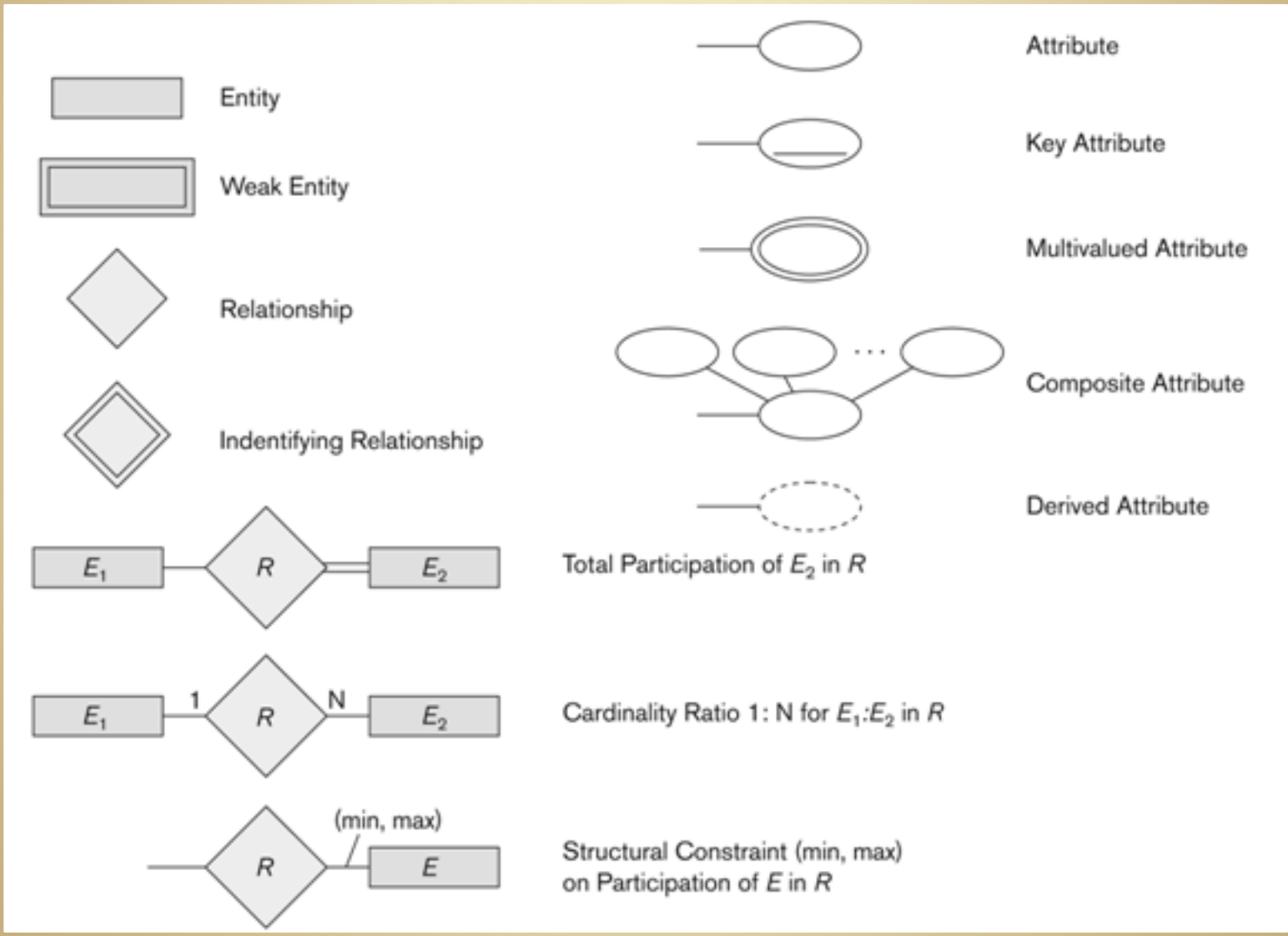
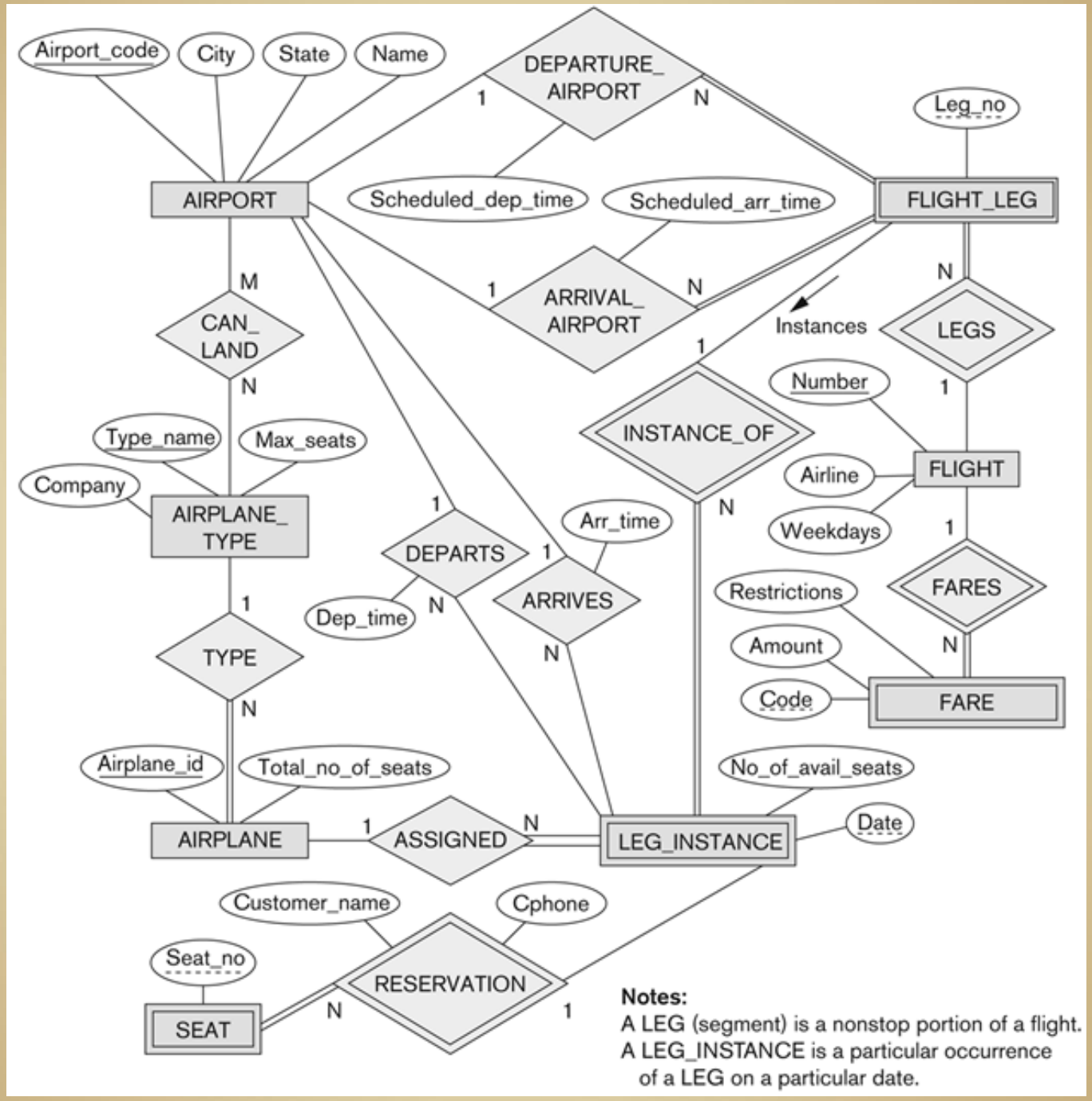
# Advanced Database Management Systems

## Lecture 3 – Sections 4.1-4.5
## Enhanced Entity-Relationship Diagrams

# ER Notation


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
1
N
Cardinality Ratio 1: N for $E_1$:$E_2$ in R
(min, max)
E
Structural Constraint (min, max)
on Participation of E in R

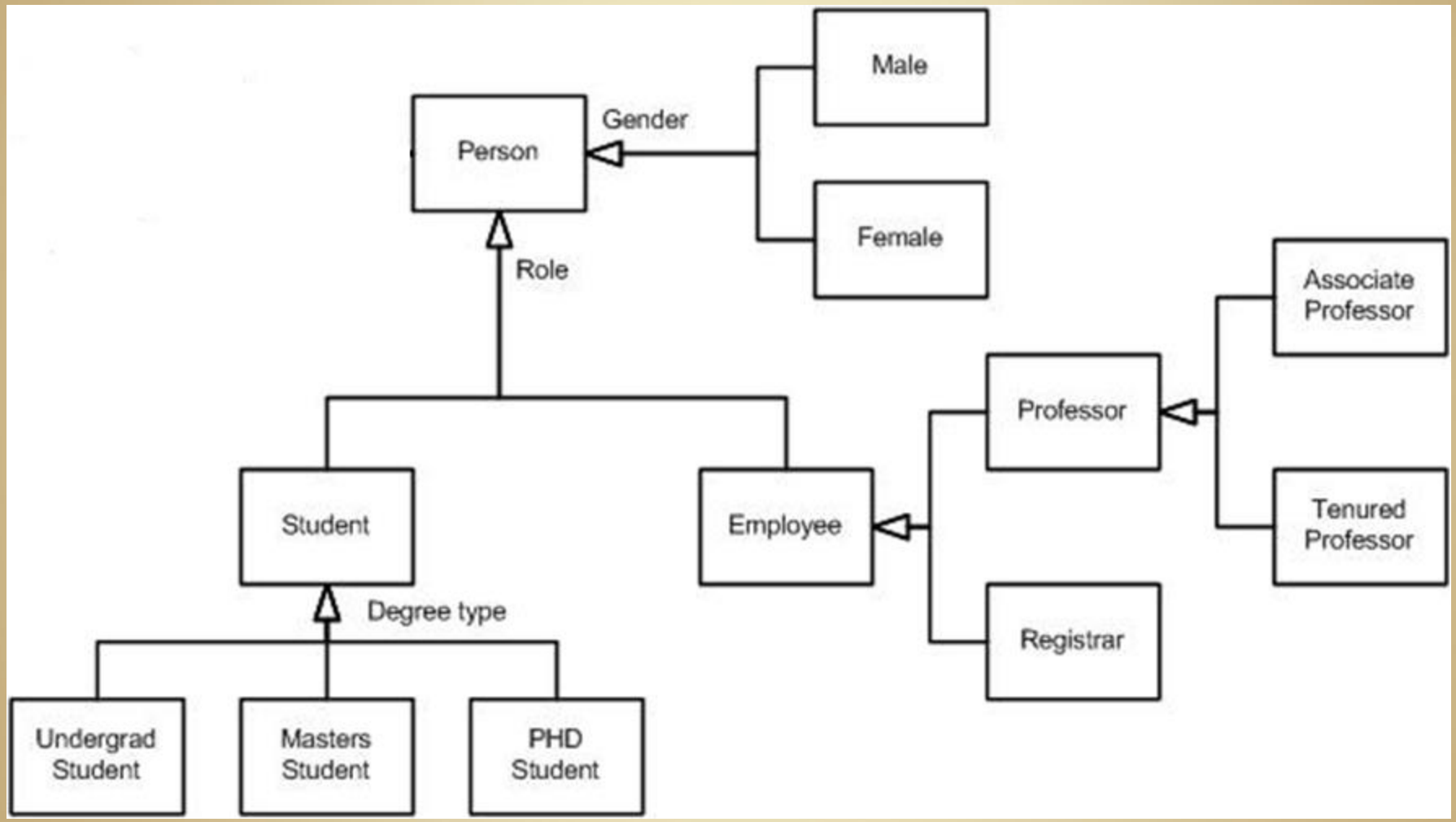
# REVIEW


Airport_code
City
State
Name
DEPARTURE_AIRPORT
1
N
Leg_no
AIRPORT
Scheduled_dep_time
Scheduled_arr_time
FLIGHT_LEG
M
1
ARRIVAL_AIRPORT
N
N
Instances
LEGS
CAN_LAND
N
1
Number
Type_name
Max_seats
INSTANCE_OF
Company
AIRPLANE_TYPE
1
Arr_time
Airline
FLIGHT
DEPARTS
1
Weekdays
1
Dep_time
N
ARRIVES
Restrictions
FARES
1
N
Amount
TYPE
N
N
Code
FARE
N
Airplane_id
Total_no_of_seats
No_of_avail_seats
AIRPLANE
1
ASSIGNED
N
LEG_INSTANCE
Date
Customer_name
Cphone
Seat_no
RESERVATION
SEAT
N
1
Notes:
A LEG (segment) is a nonstop portion of a flight.
A LEG_INSTANCE is a particular occurrence of a LEG on a particular date.

# Enhanced ER Model (EER)

- aka Extended Entity-Relationship Model
- adds *Inheritance*
  - indicates that one entity type is
    an extension of another entity type
  - often referred to as an IS-A relationship

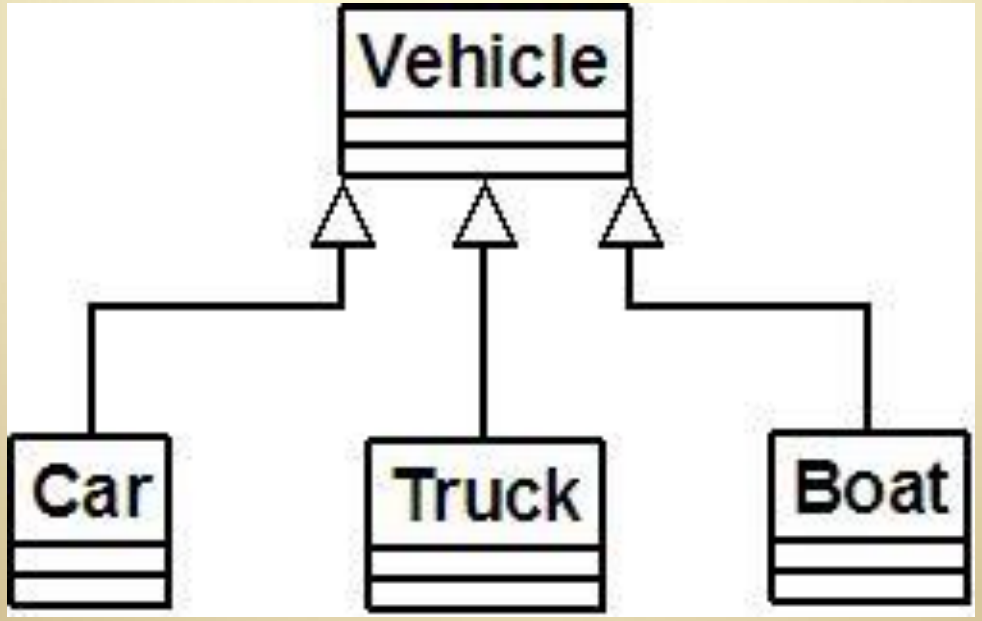
# Inheritance




http://www.agilemodeling.com/artifacts/classDiagram.htm

# Inheritance: UML

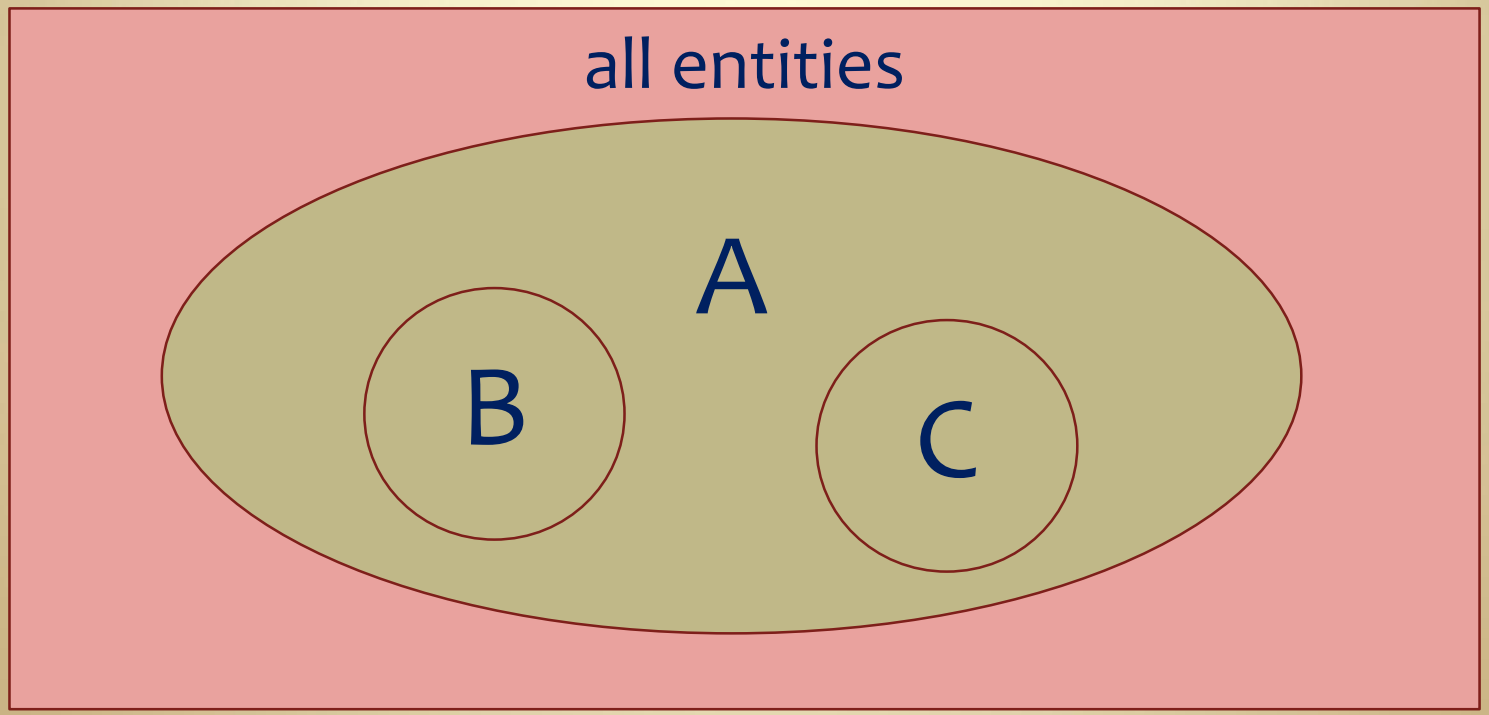
- *Inheritance* defines a subclass relationship
  - A subclass inherits all properties (members) of the superclass
- This is the perspective
  of most modern programming languages



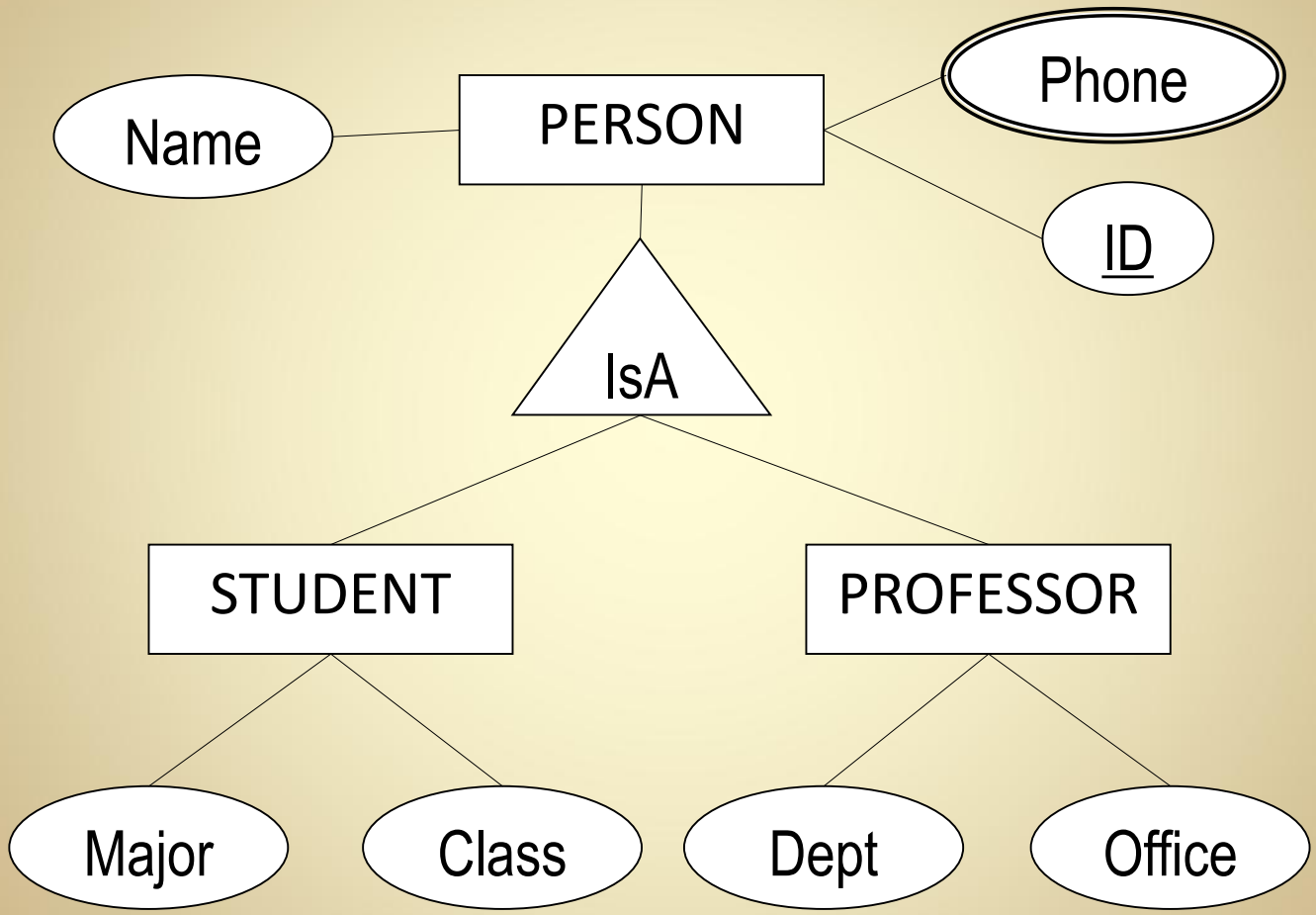
# Set Theoretic View of Inheritance

- $B \subset A$, $C \subset A$
- Every B is also an A
- Every C is also an A

everything true about a member of a set is also true about any member of its subsets.

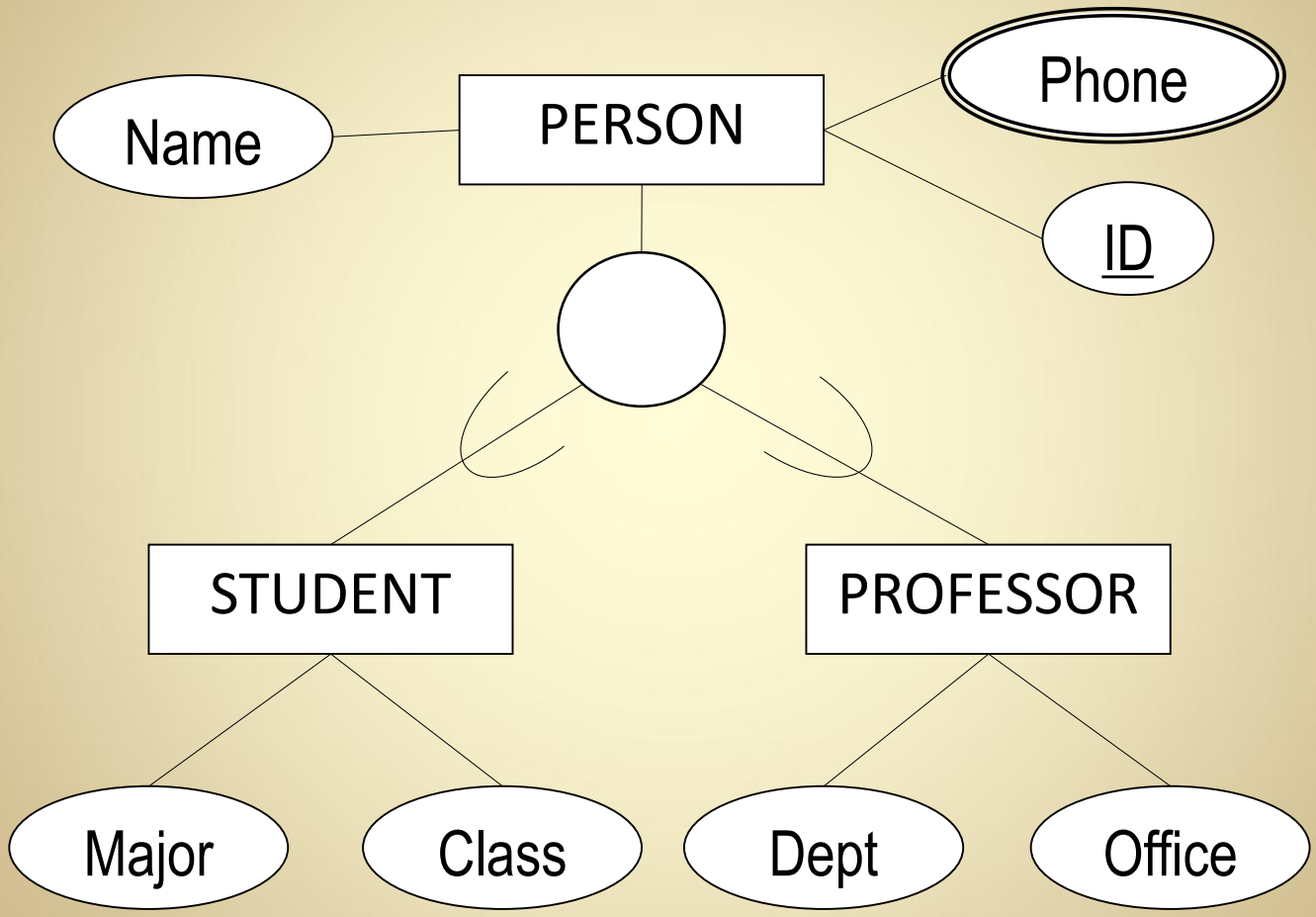


# EER IsA Notation



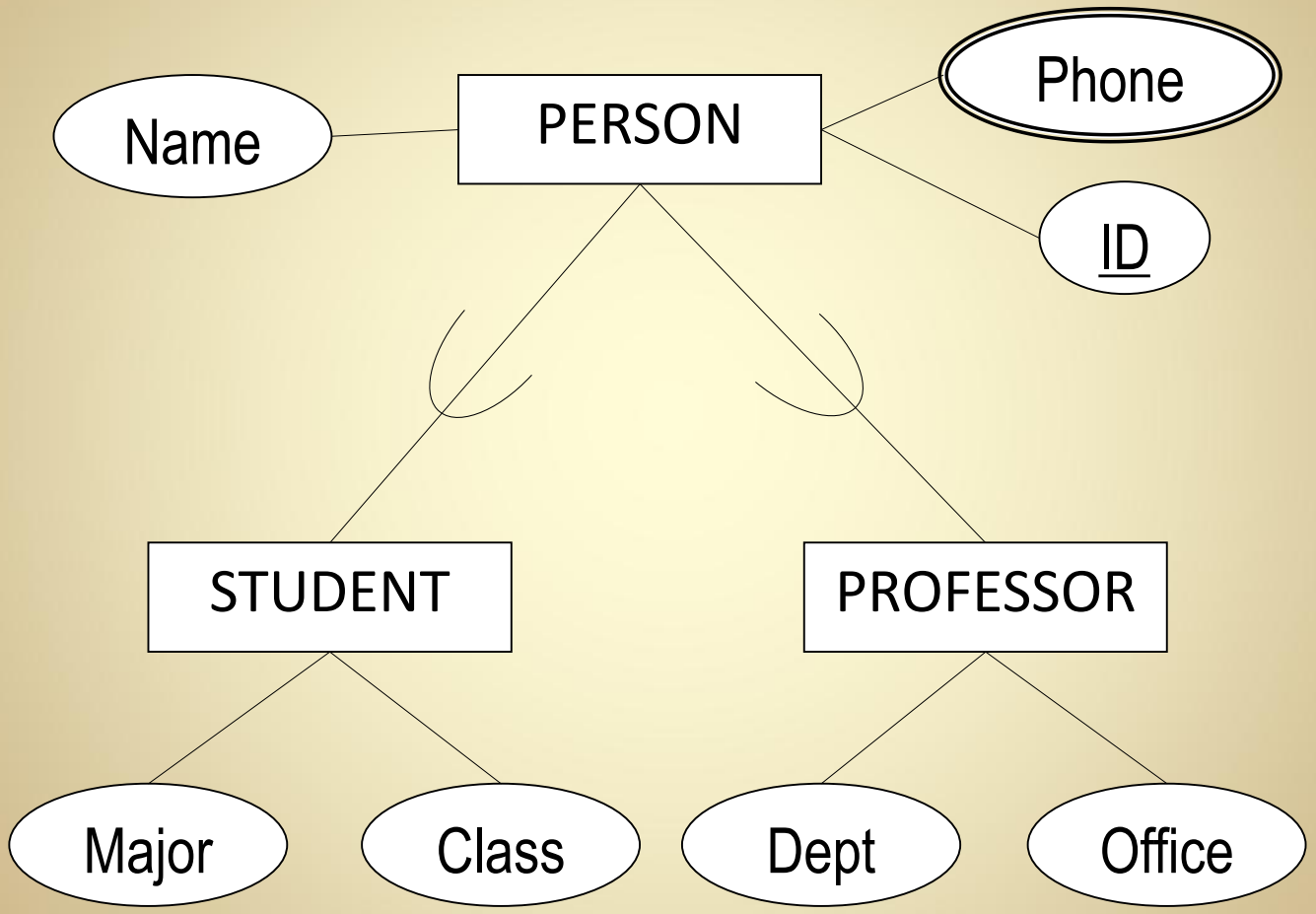

(not used in our textbook)

# EER Subset Notation




preferred notation: shows directionality of inheritance

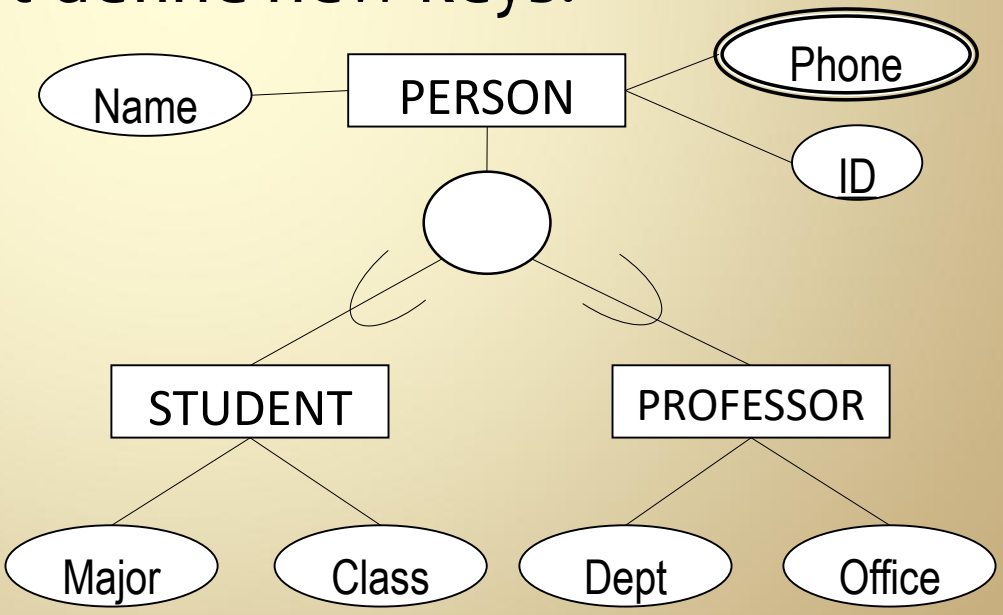
# EER Subset Notation (variant)




The circle may be omitted when not needed.

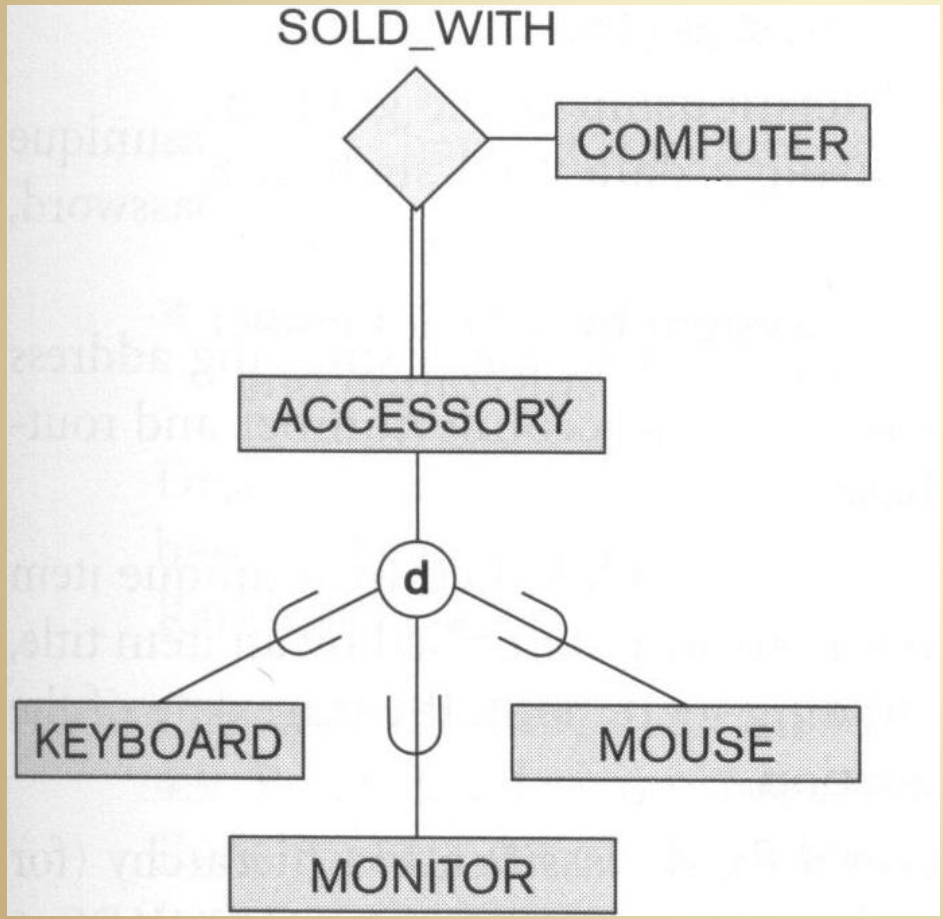
# Inheritance of Properties

- Student and professor entities have all attributes defined for a person, plus additional attributes
- Keys are also inherited. Subtypes should not define new keys.



# Inheritance of Relationships

- Relationships are also inherited by subtypes




Every mouse must be sold with a computer? TRUE

Every computer must be sold with a mouse? FALSE

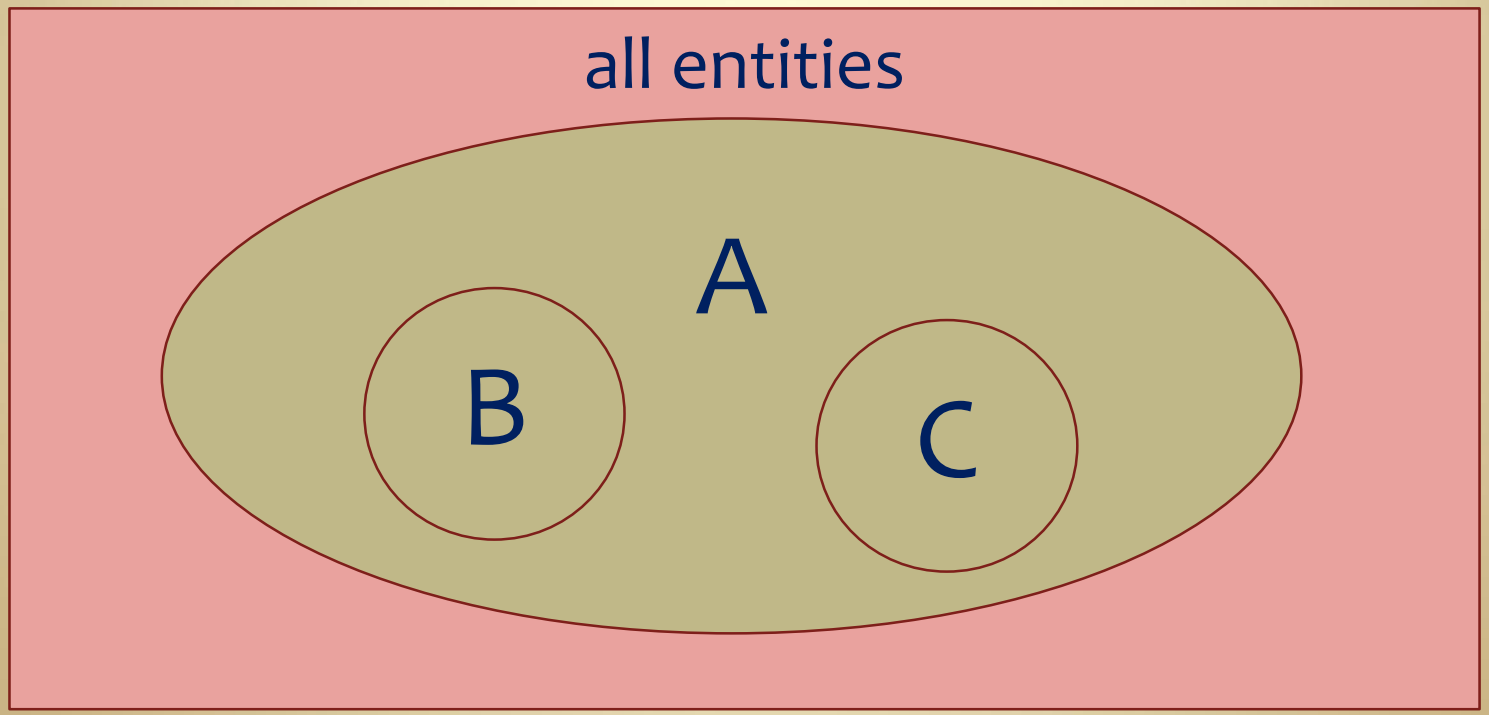
# Constraints on Inheritance

- *Disjointness*: an entity can be a member of at most one subtype
  - a person *may be* a student or *may* be a professor, but *not both*
- Covering: every entity of the supertype must also be a member of at least one subtype
  - every person *must be* a professor *or* a student

# Disjoint Subsets

- $B \subset A$, $C \subset A$
- $B \cap C = \emptyset$

no entity is in both B and C

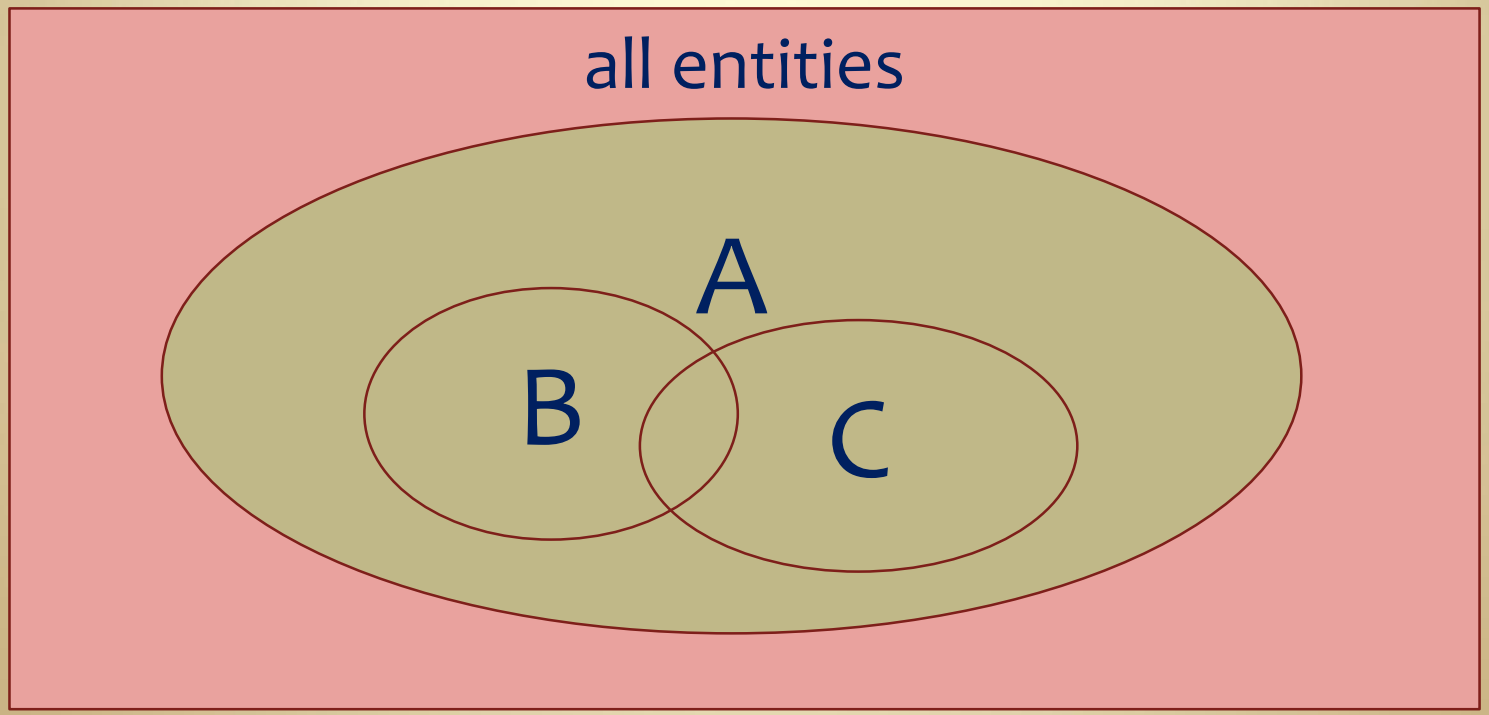


# Overlapping Subsets

- $B \subset A, C \subset A$
- ~~$B \cap C = \emptyset$~~

an entity may be in both B and C

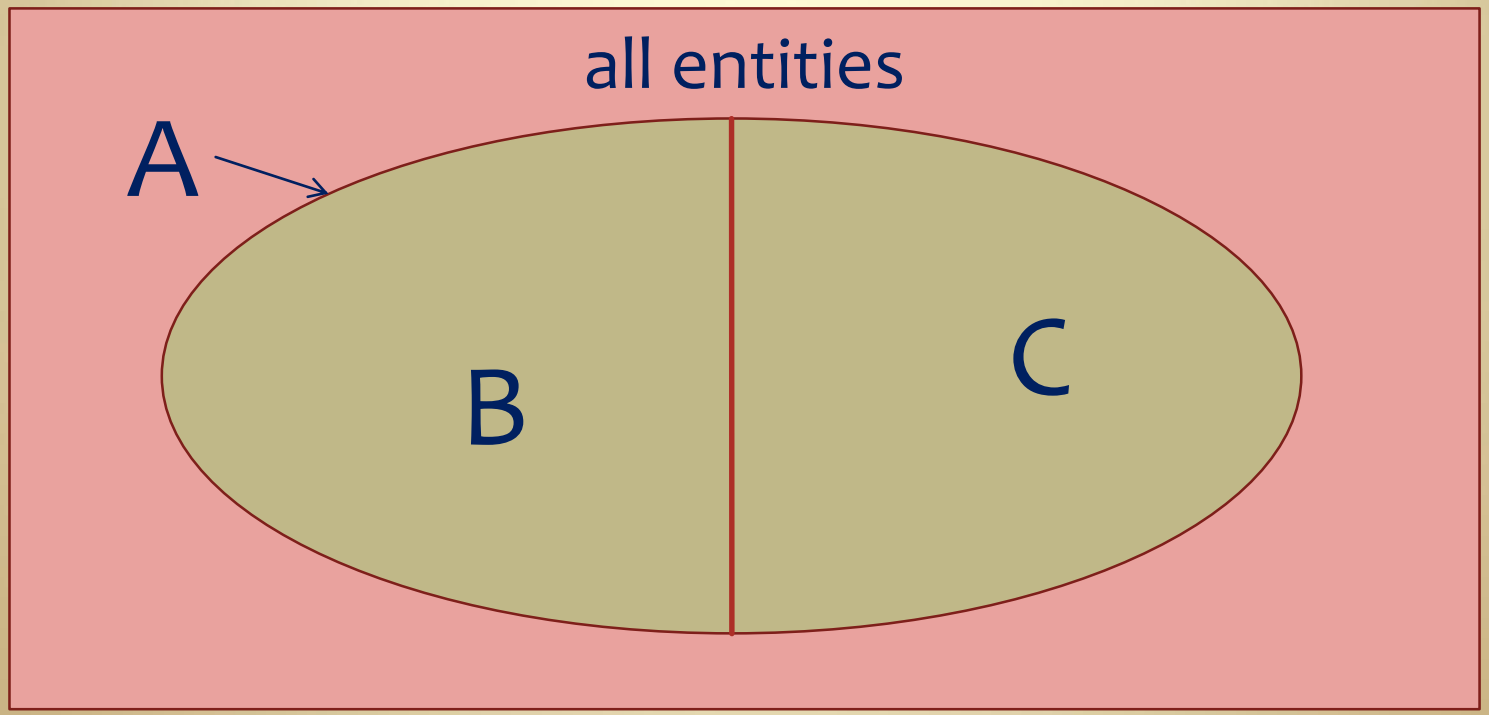
overlapping = non-disjoint



# Covering Subsets

- $B \subset A$, $C \subset A$
- $B \cup C = A$

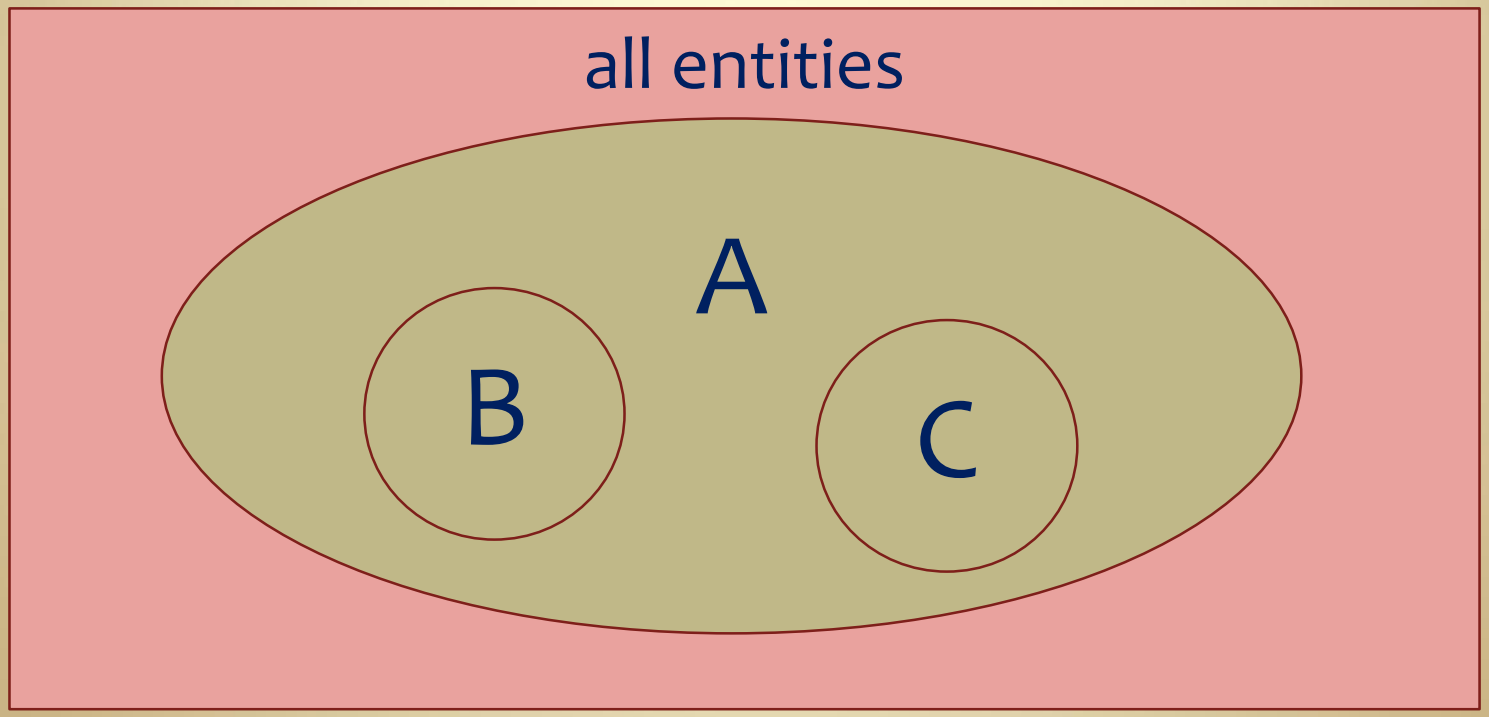
every entity in A is also in B or C



# Non-Covering Subsets

- $B \subset A, C \subset A$
- ~~$B \cup C = A$~~

some entities in A are not in B or C



# Inheritance Constraint Notation

- IsA (triangle) notation:

  write “disjoint” and/or “covering”
  next to the triangle

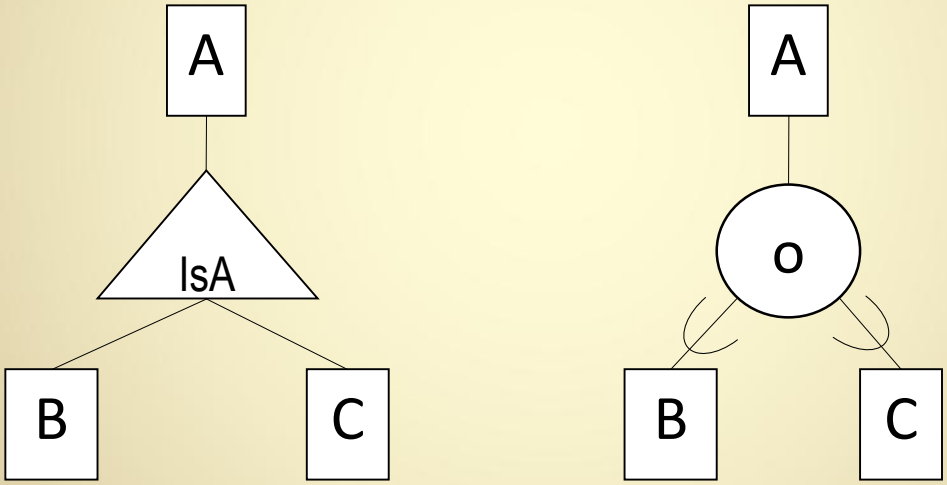
- Subset notation,

  ‘d’ in the circle → disjointness
  ‘o’ in the circle → no disjointness (overlap)

  required participation from supertype
  indicates a covering constraint

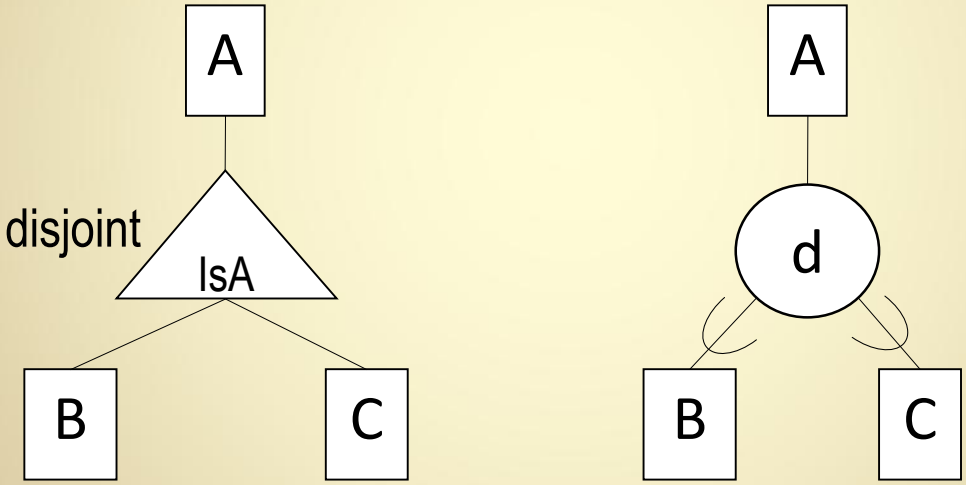
# Non-disjoint, Non-covering

- Every A can also be a B or a C, or both, or neither



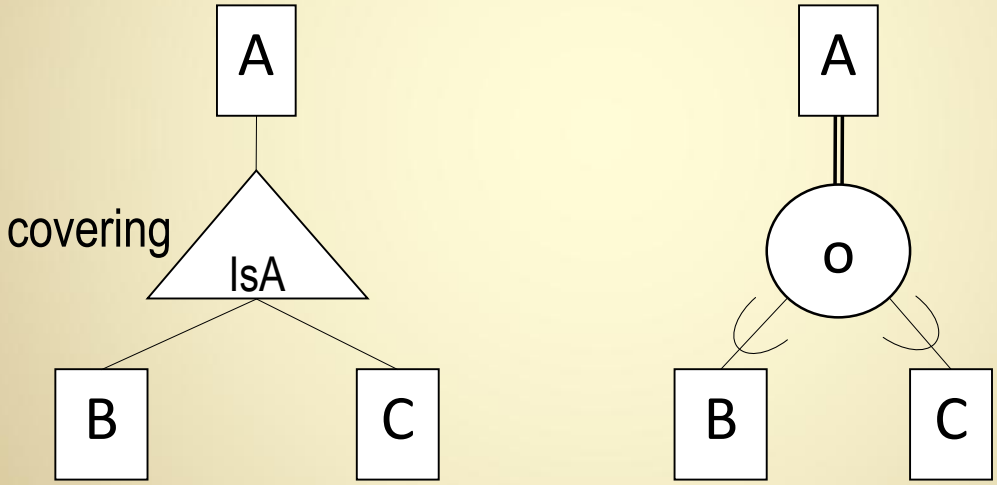
# Disjoint, Non-covering

- Every A can also be a B or a C or but not both



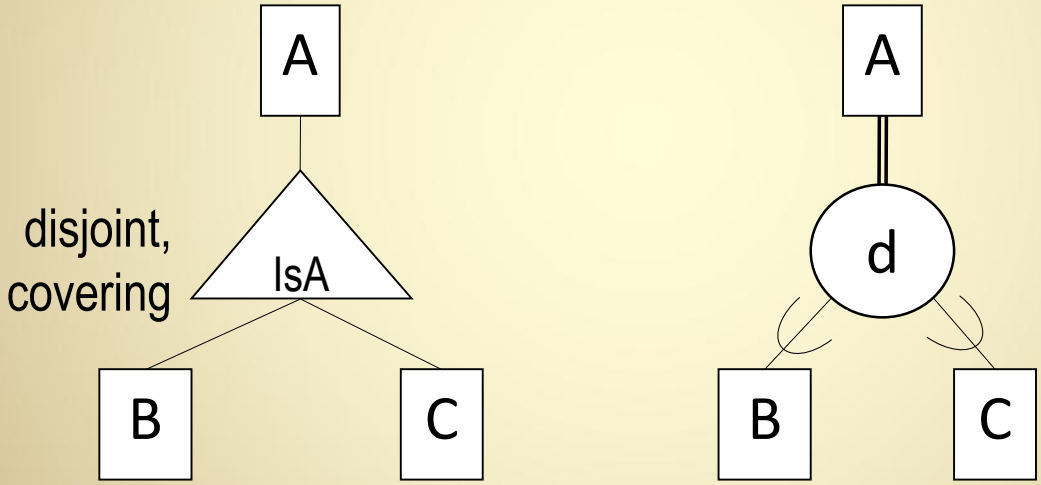
# Non-disjoint, Covering

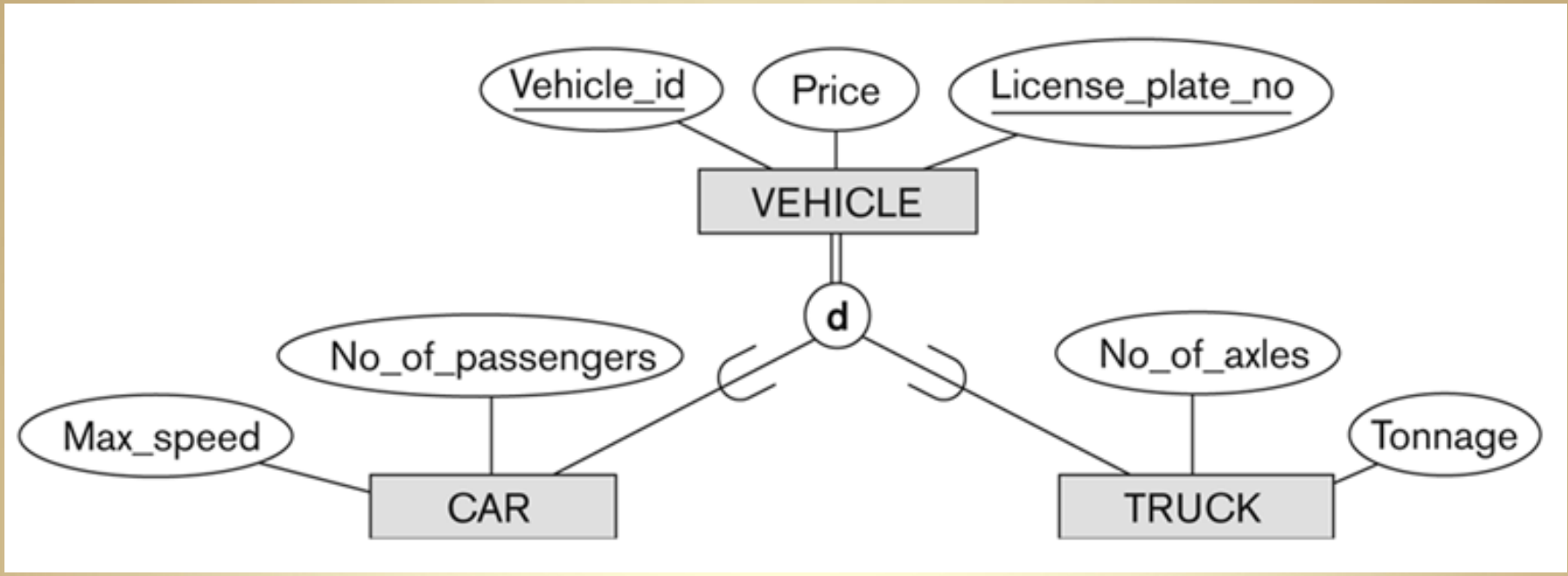
- Every A must be a B or a C or both



# Disjoint, Covering

- Every A must be a B or a C, but not both





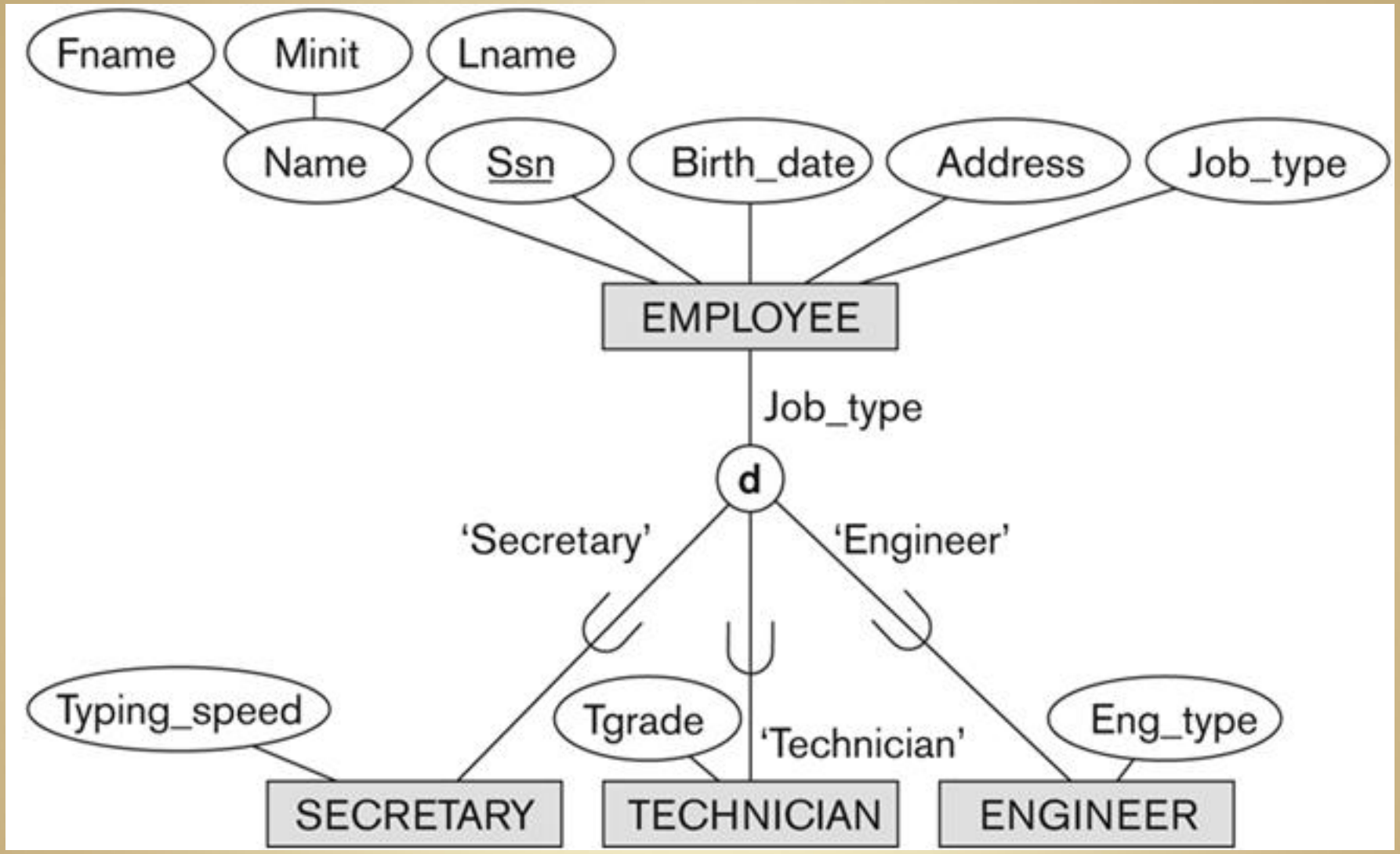

**disjoint, covering inheritance:**

every car is a vehicle

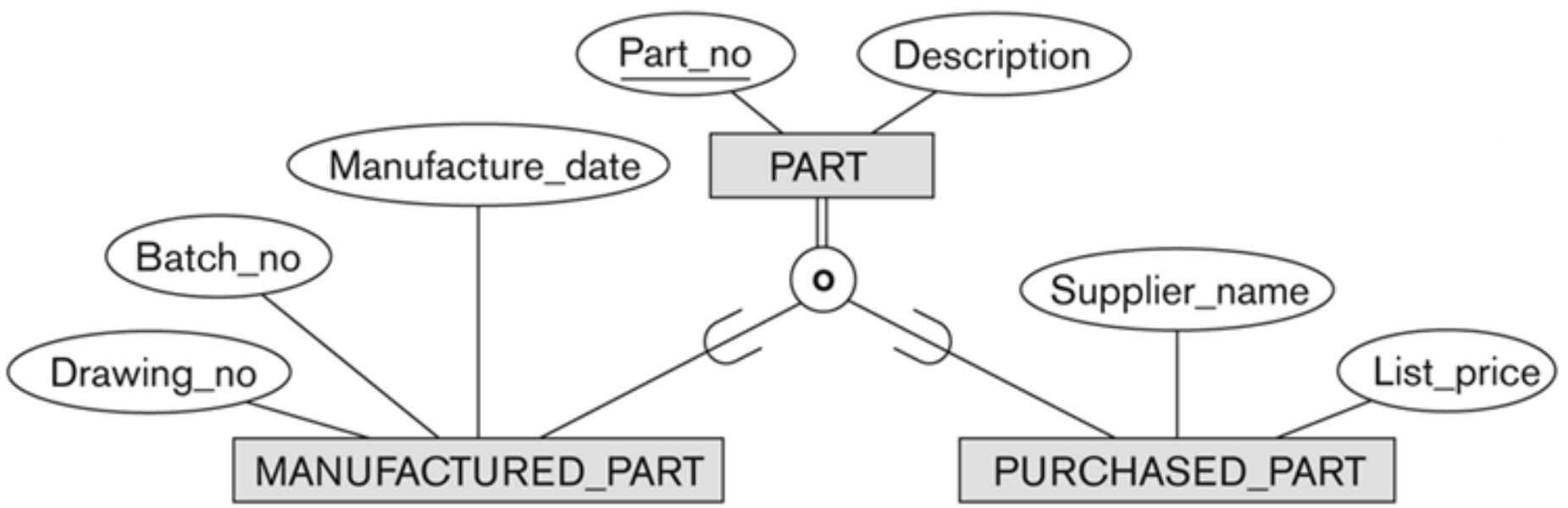
every truck is a vehicle

every vehicle is either a car or a truck

no vehicle is both a car and a truck

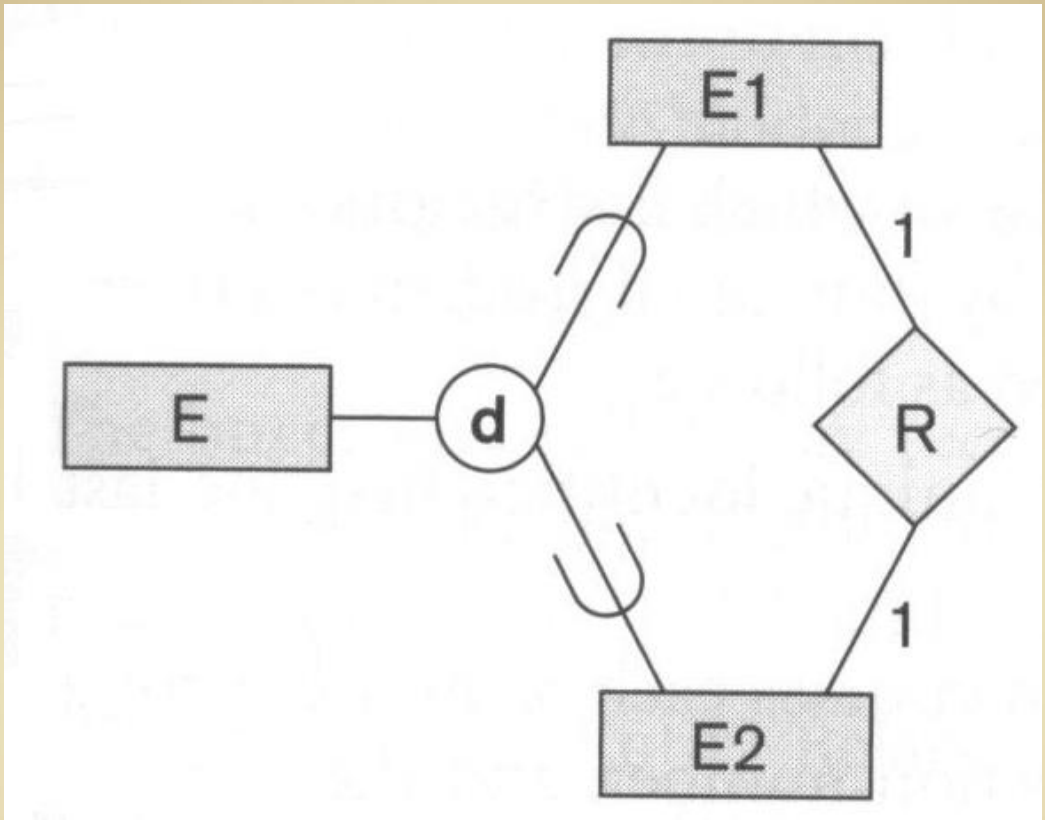



**disjoint, non-covering inheritance:**
employees may be secretaries, technicians or engineers, but not more than one of these




**non-disjoint, covering inheritance:**
every part is a manufactured part,
or a purchased part,
or both (a purchased, manufactured part)

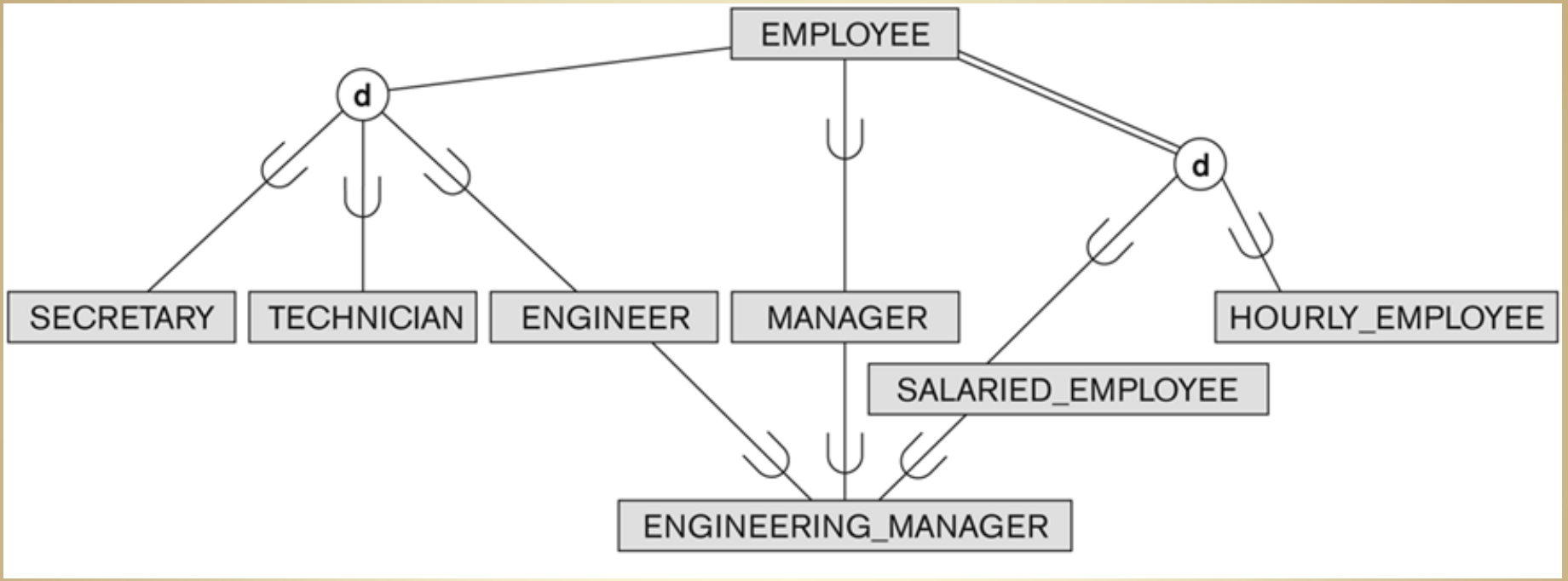
** How many attributes does a purchased, manufactured part have?
** How would we model this in UML (C++ or Java)?




** Interpret this schema.

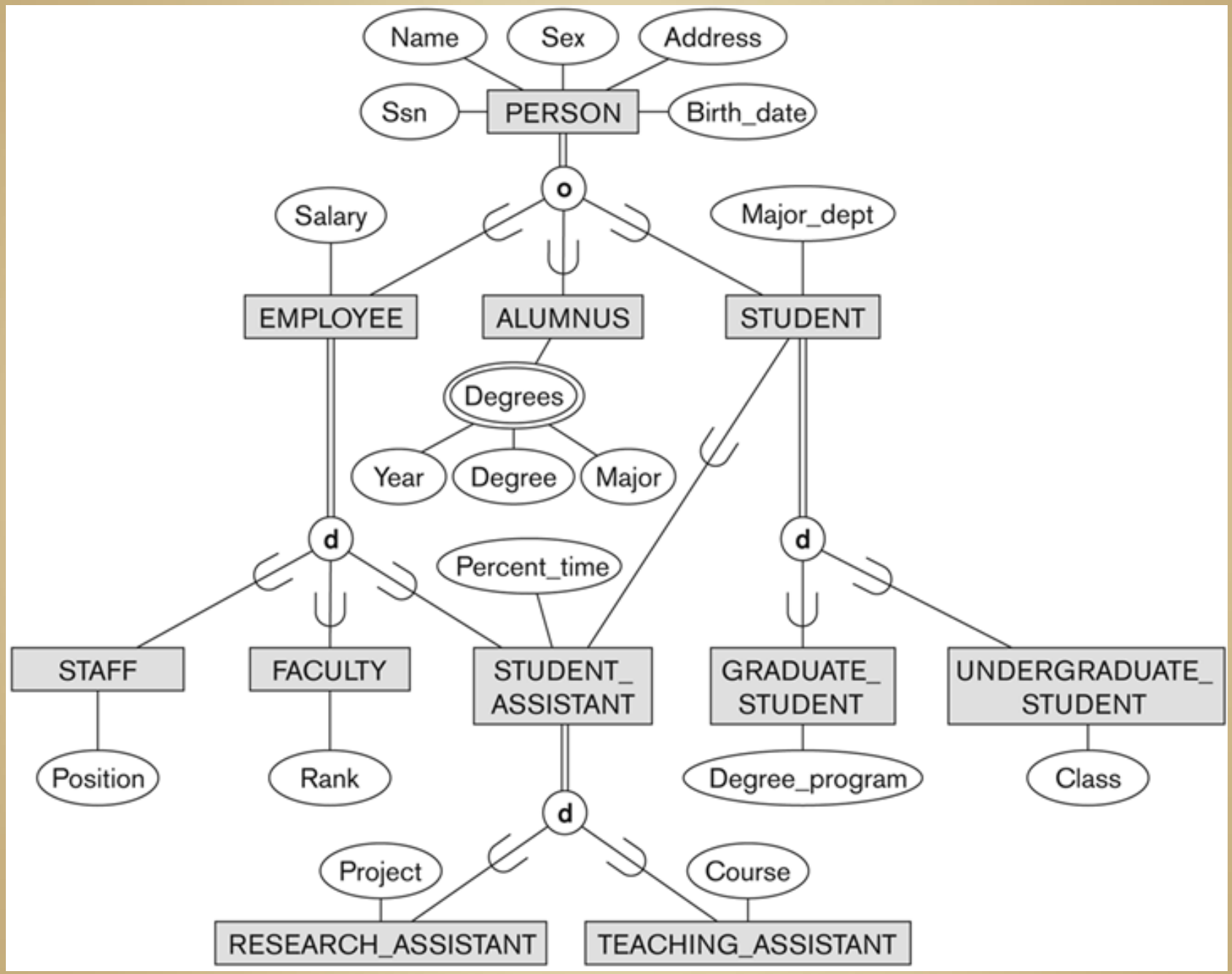
** Can you find a real-world example?

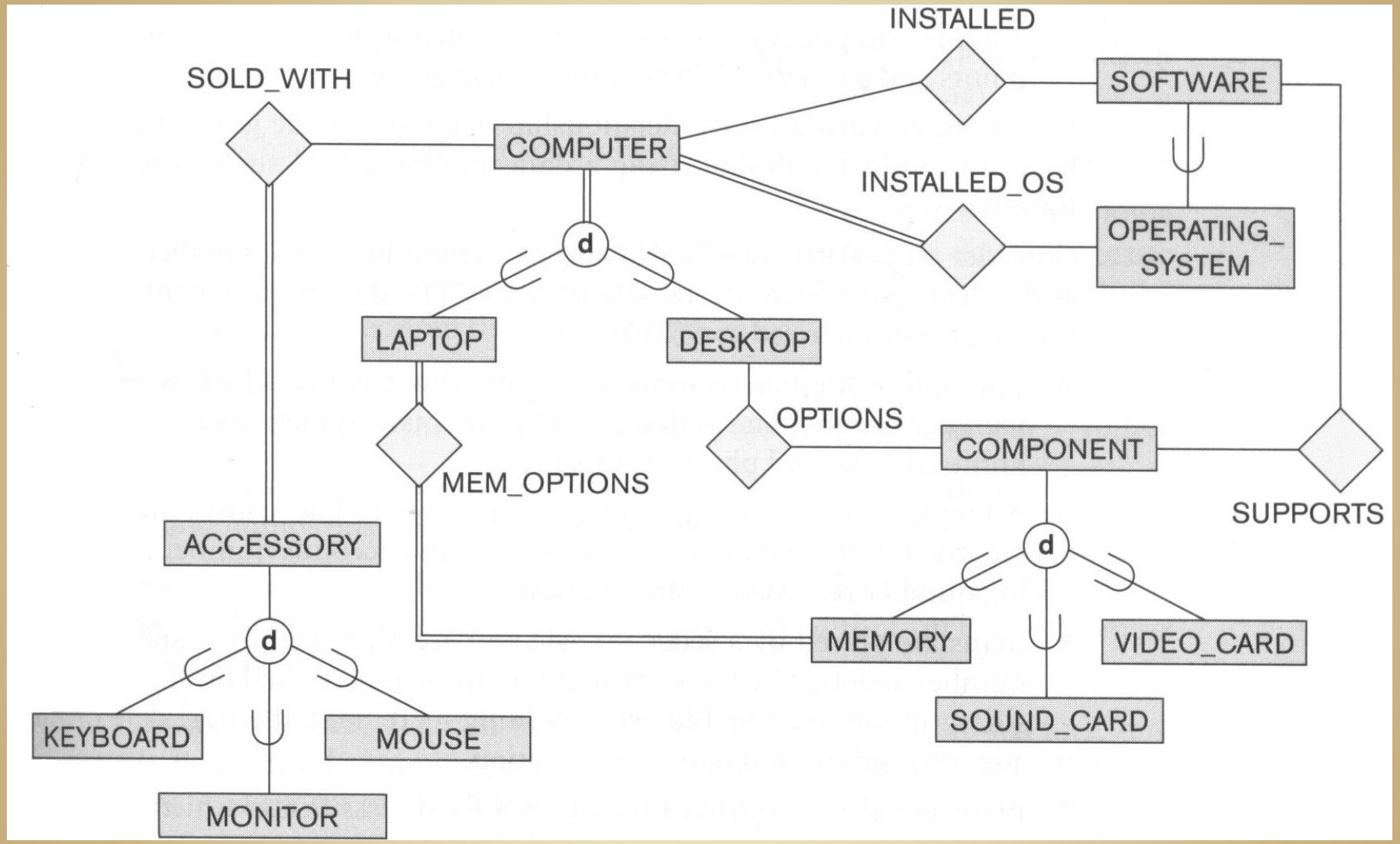
# Inheritance Lattice

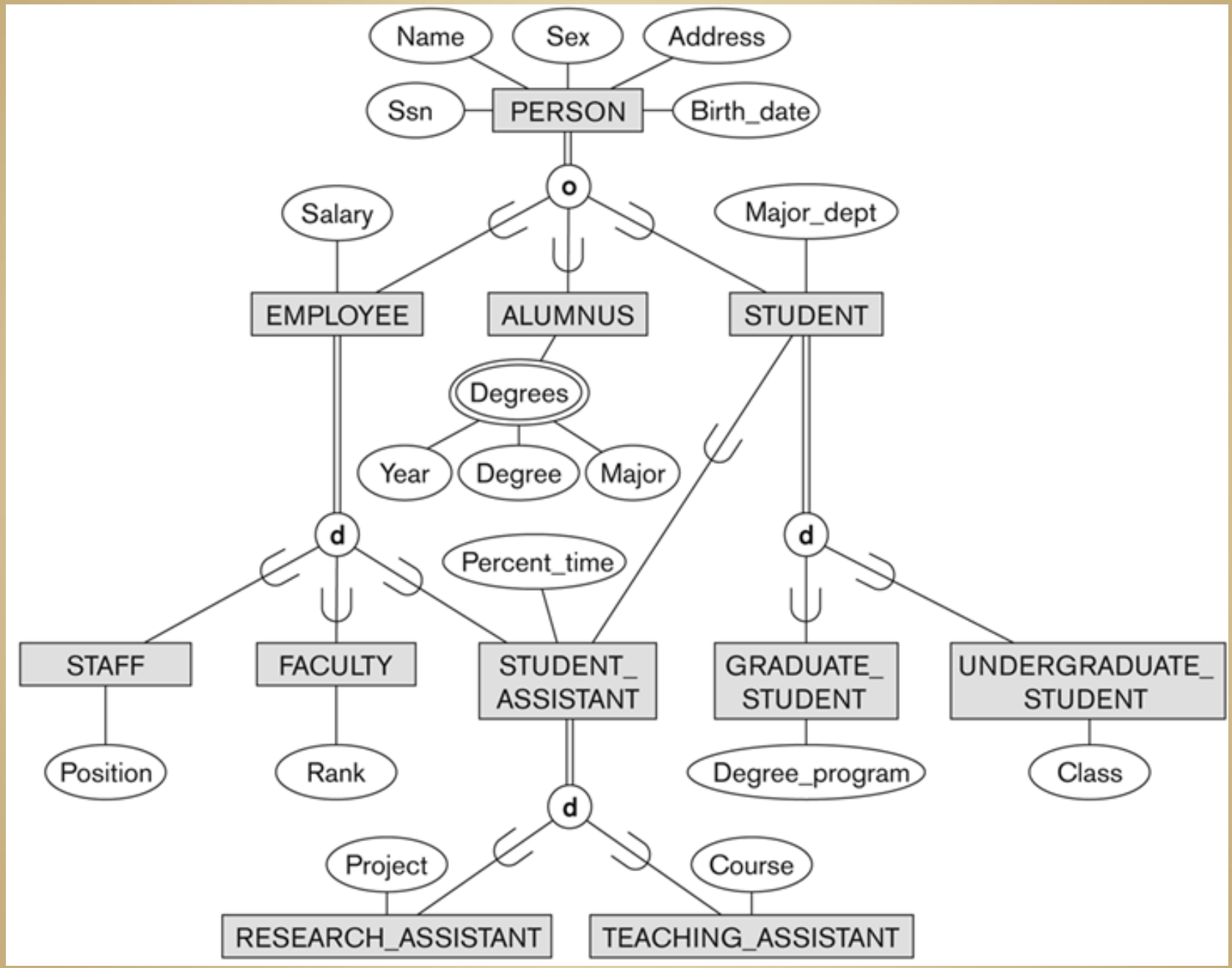



Multiple inheritance gives us a lattice, rather than a hierarchy

** Could we have engineering managers without defining the E_M class? (compare to previous example)

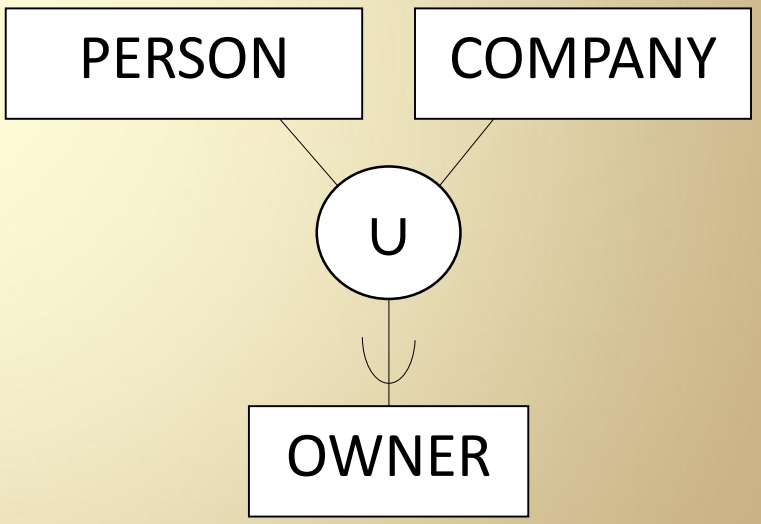

Name
Sex
Address
Ssn
PERSON
Birth_date
o
Salary
Major_dept
EMPLOYEE
ALUMNUS
STUDENT
Degrees
Year
Degree
Major
d
Percent_time
d
STAFF
FACULTY
STUDENT_ ASSISTANT
GRADUATE_ STUDENT
UNDERGRADUATE_ STUDENT
Position
Rank
Degree_program
Class
d
Project
Course
RESEARCH_ASSISTANT
TEACHING_ASSISTANT


INSTALLED
SOLD_WITH
SOFTWARE
COMPUTER
INSTALLED_OS
OPERATING_
SYSTEM
d
LAPTOP
DESKTOP
OPTIONS
COMPONENT
MEM_OPTIONS
SUPPORTS
ACCESSORY
d
d
MEMORY
VIDEO_CARD
KEYBOARD
MOUSE
SOUND_CARD
MONITOR


Name
Sex
Address
Ssn
PERSON
Birth_date
o
Salary
Major_dept
EMPLOYEE
ALUMNUS
STUDENT
Degrees
Year
Degree
Major
d
Percent_time
d
STAFF
FACULTY
STUDENT_ ASSISTANT
GRADUATE_ STUDENT
UNDERGRADUATE_ STUDENT
Position
Rank
Degree_program
Class
d
Project
Course
RESEARCH_ASSISTANT
TEACHING_ASSISTANT

# Unions

- Union defines a type as the union of other types
- OWNER = PERSON ∪ COMPANY
- OWNER is called a *union type* or *category*
  - OWNER is the subtype of the union of PERSON and COMPANY
- Not multiple inheritance
  - an OWNER does not need all the attributes from both PERSON and COMPANY



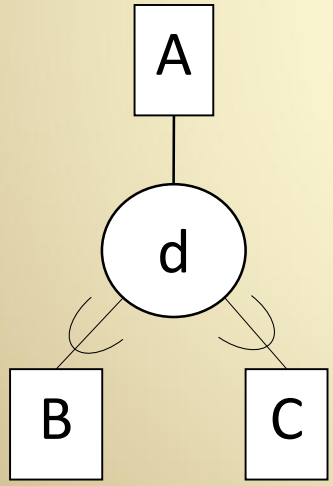
# Comparison

B and C are subtypes of A

$B \subset A$

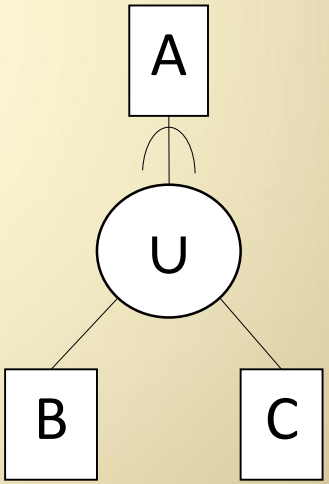
$B \subset A$

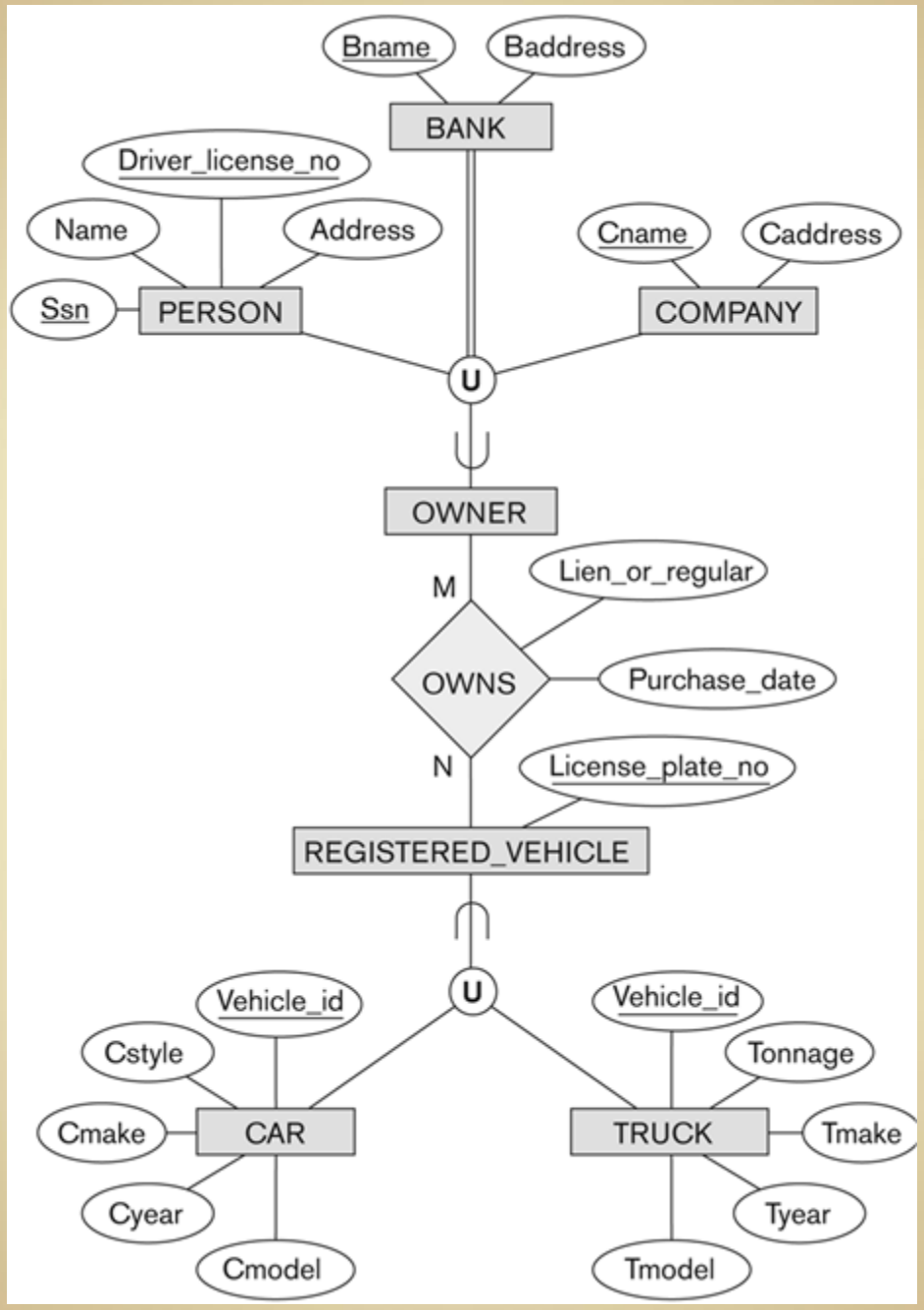
$B \cup C \subset A$

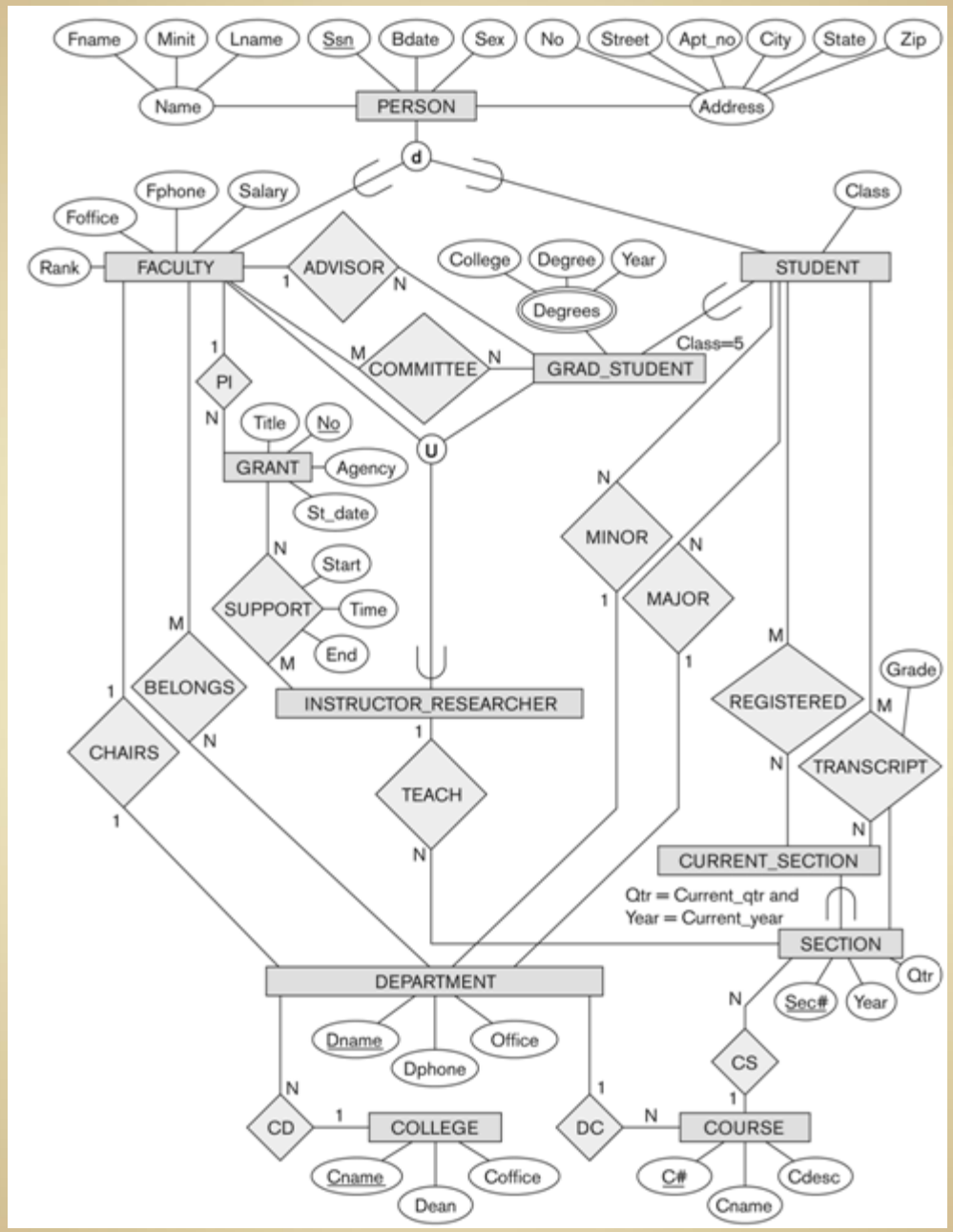


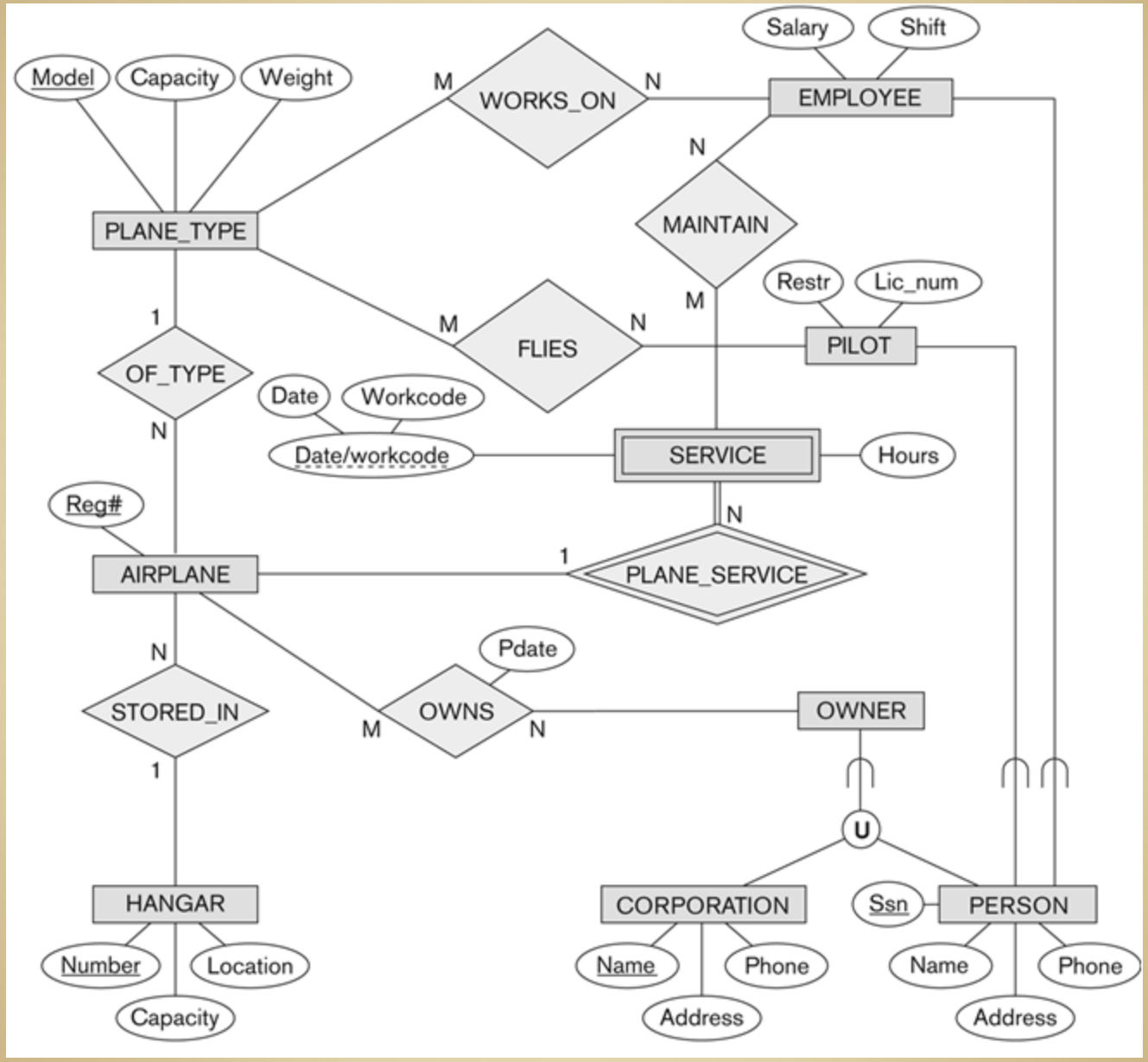

A is a subtype of B∪C

$A \subset B \cup C$




Bname
Baddress
BANK
Driver_license_no
Name
Address
Ssn
PERSON
Cname
Caddress
COMPANY
U
OWNER
M
Lien_or_regular
OWNS
Purchase_date
N
License_plate_no
REGISTERED_VEHICLE
U
Vehicle_id
Cstyle
Cmake
CAR
Cyear
Cmodel
Vehicle_id
Tonnage
TRUCK
Tmake
Tyear
Tmodel


Fname
Minit
Lname
Ssn
Bdate
Sex
No
Street
Apt_no
City
State
Zip
Name
PERSON
Address
d
Fphone
Salary
Foffice
Class
Rank
FACULTY
ADVISOR
College
Degree
Year
STUDENT
Degrees
Class=5
PI
COMMITTEE
GRAD_STUDENT
Title
No
U
GRANT
Agency
St_date
MINOR
MAJOR
Start
SUPPORT
Time
End
BELONGS
INSTRUCTOR_RESEARCHER
REGISTERED
Grade
CHAIRS
TEACH
TRANSCRIPT
CURRENT_SECTION
Qtr = Current_qtr and
Year = Current_year
SECTION
Qtr
DEPARTMENT
Sec#
Year
Dname
Office
Dphone
CS
CD
COLLEGE
DC
COURSE
Cname
Coffice
Dean
C#
Cdesc
Cname


Salary
Shift
Model
Capacity
Weight
M
WORKS_ON
N
EMPLOYEE
N
MAINTAIN
PLANE_TYPE
M
Restr
Lic_num
1
M
FLIES
N
PILOT
OF_TYPE
Date
Workcode
N
Date/workcode
SERVICE
Hours
Reg#
N
1
AIRPLANE
PLANE_SERVICE
N
Pdate
STORED_IN
M
OWNS
N
OWNER
1
U
HANGAR
CORPORATION
Ssn
PERSON
Number
Location
Name
Phone
Name
Phone
Capacity
Address
Address