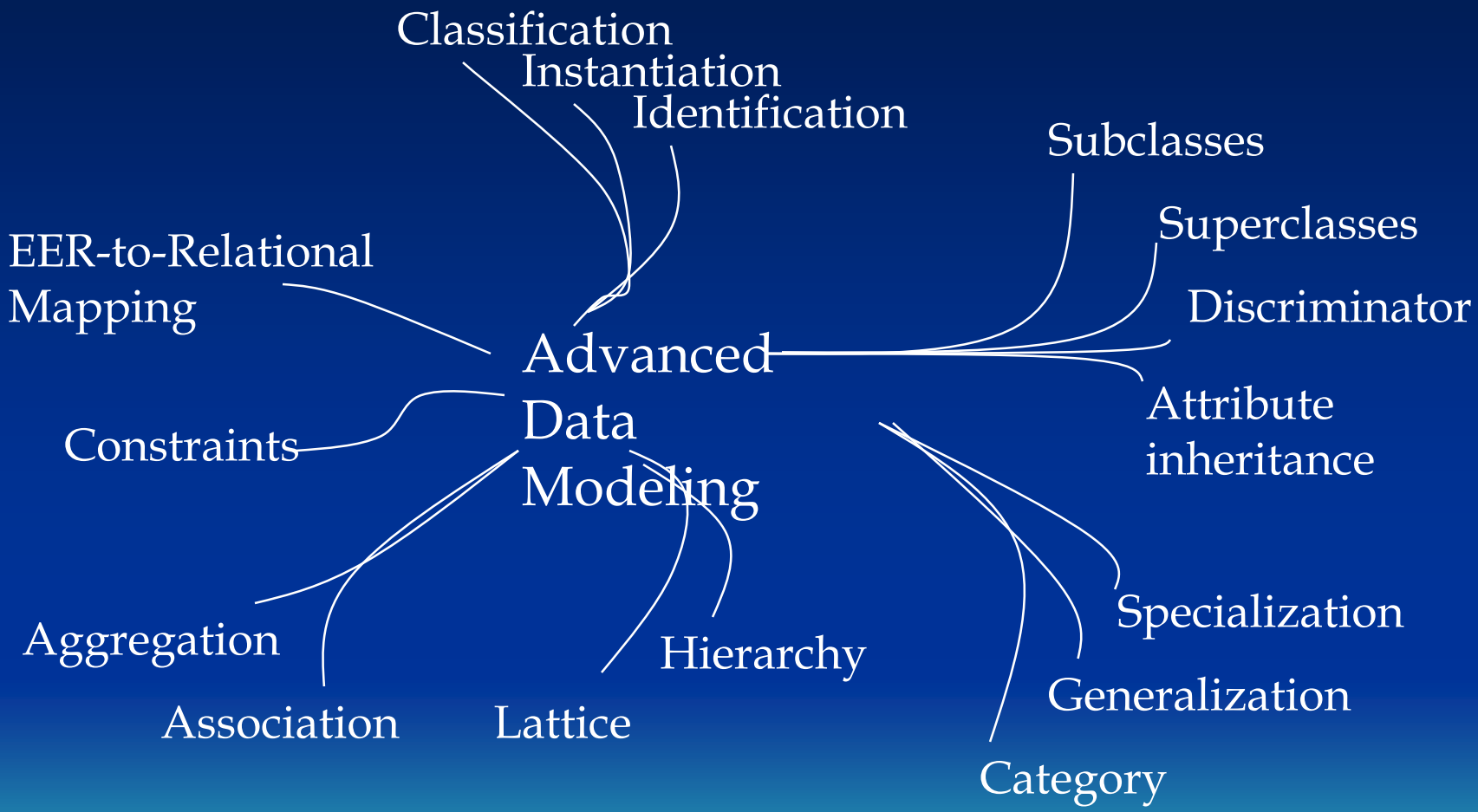
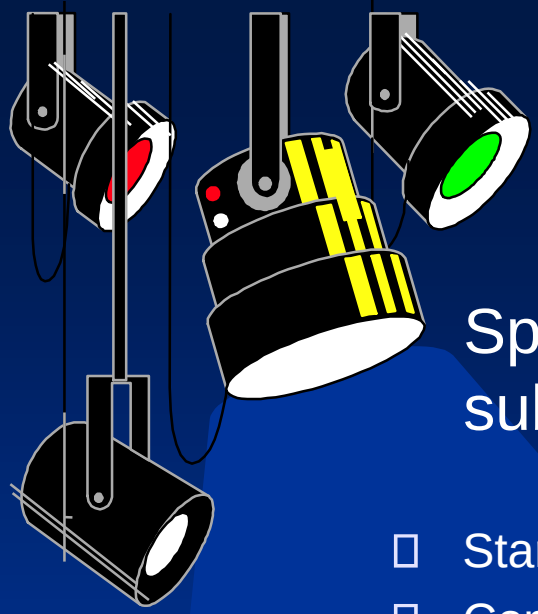
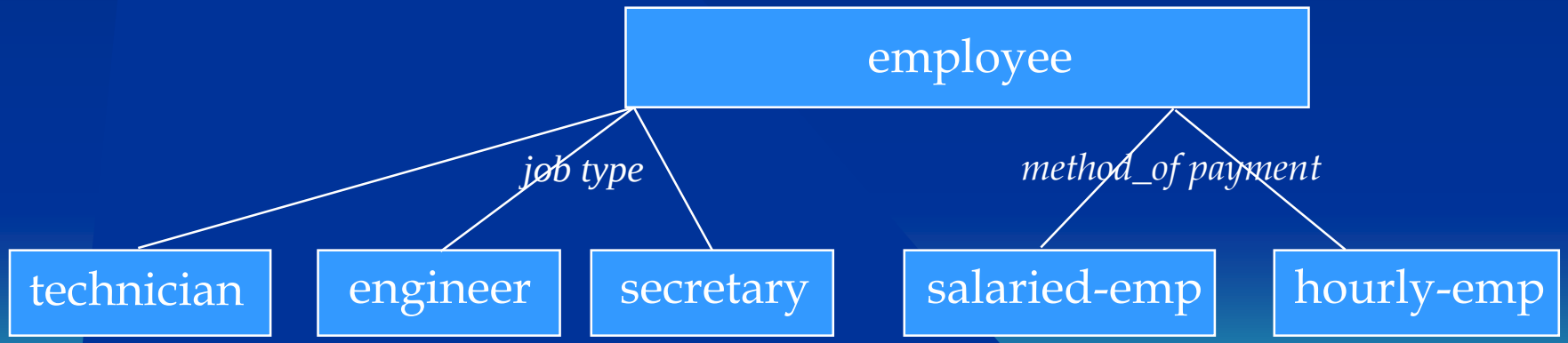

Classification
Instantiation
Identification
Subclasses
Superclasses
Discriminator
Attribute inheritance
EER-to-Relational Mapping
Advanced Data Modeling
Constraints
Aggregation
Association
Lattice
Hierarchy
Specialization
Generalization
Category



Specialization

Specialization is the process of defining a set of sub-entities of some entity type.

- Starting with Employee
- Consider the *Job Type* and *Method of payment* attributes
- We can specialize to create:



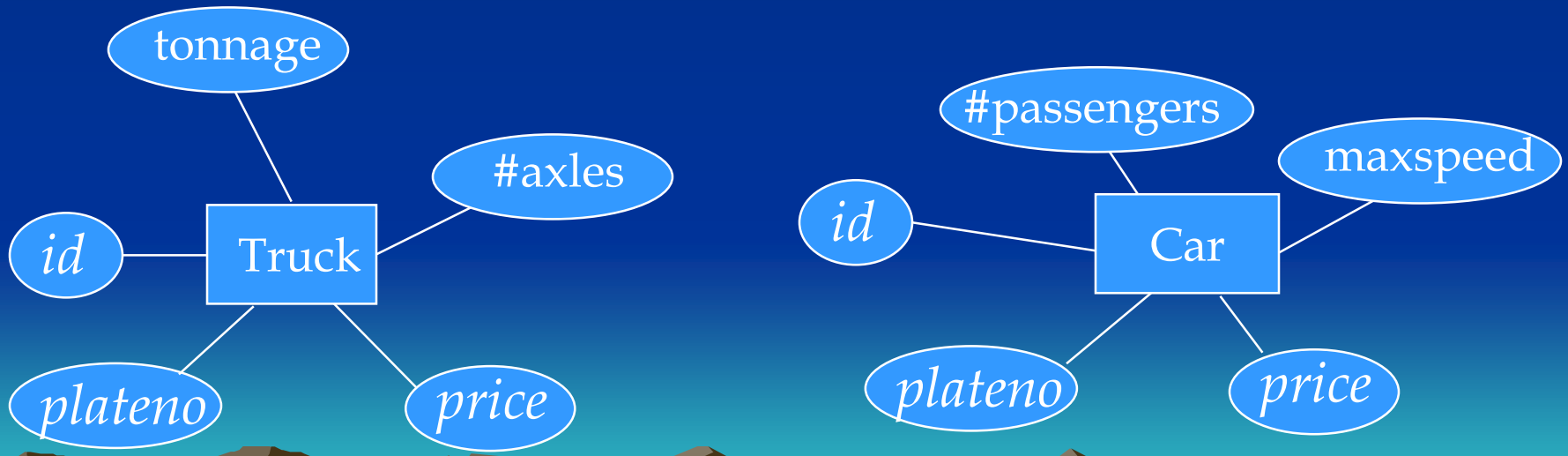
Specialization

- Allows us to define a set of subclasses of an entity
  - in other words, helps us focus on distinguishing characteristics
- Associate additional specific characteristics with each subclass
  - helps us define attributes of each subclass
- Establish other relationships for subclasses

Generalization

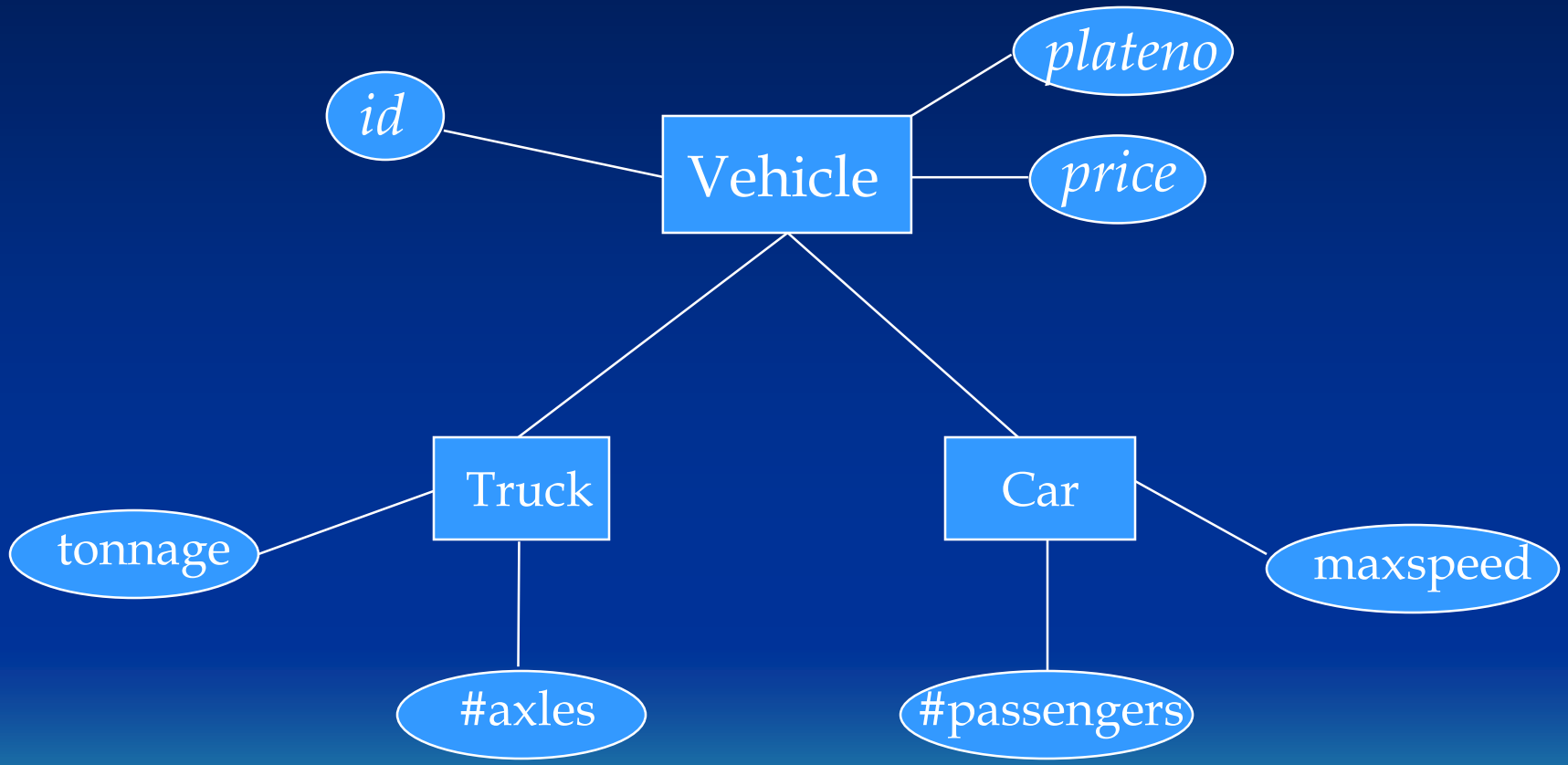
Generalization is the opposite approach/process of determining a supertype based on certain entities having common characteristics.

- reverse process of defining subclasses
- bottom up approach
- bring together common attributes from similar entity types, and suppress the differences (to form a superclass)
- example: suppose we begin with Cars and Trucks



Generalization

- we generalize to get Vehicle



Specialization

- The entity in the subclass (secretary, engineer, technician) is said to play a specialized role

- The 

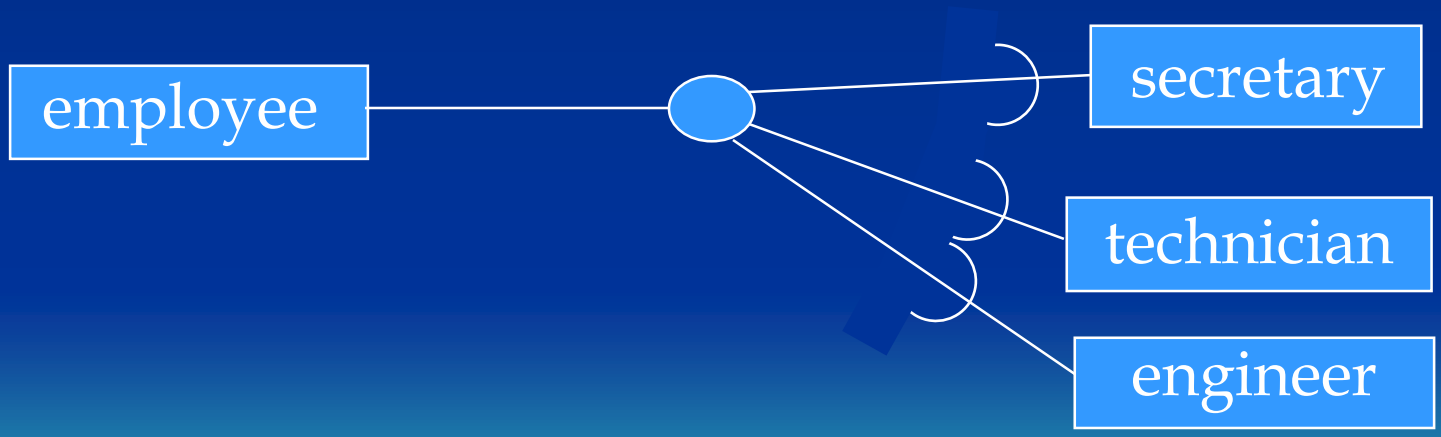
  is used when you have more than one subclass based on the same defining attribute (e.g., *Job type*)

- The class/subclass relationship is shown using:

EERD with Class/SubClass Relationship

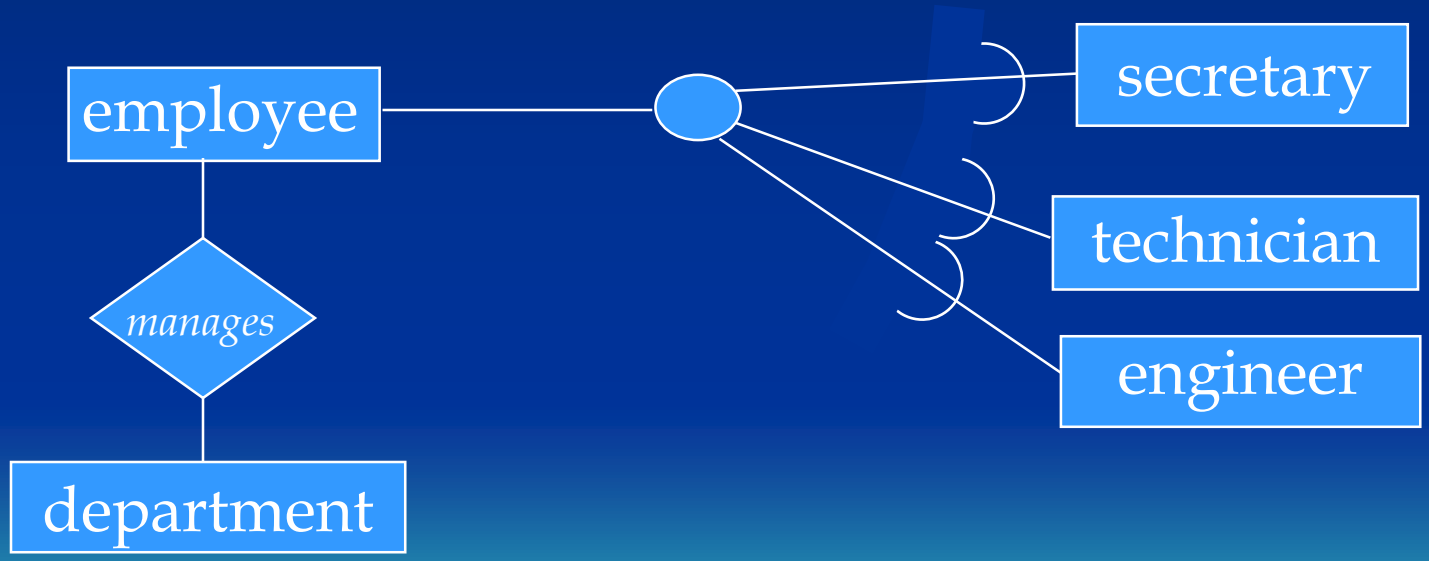
- similar to a 1:1 relationship except that it is between instances of the same entity type
- A 1:1 relationship is between two different entity types

- Example: employee/secretary/technician/engineer



EERD with Class/SubClass Relationship

- A 1:1 relationship is between two different entity types
  - Example: a *manages* relationship between employee and department



Constraints

- to determine when an entity will become a member of subclass:
  - Predicate-defined
    - system automatically enforces the constraint
    - e.g *JobType = 'Secretary'*
      - usually defined by an attribute value, a discriminator
  - User-defined
    - users decide the subclass for each entity
    - not automatically enforced

Disjointness Constraint

- Disjoint
  - an entity can be a member of at most one subclass of a specialization
  - Notation

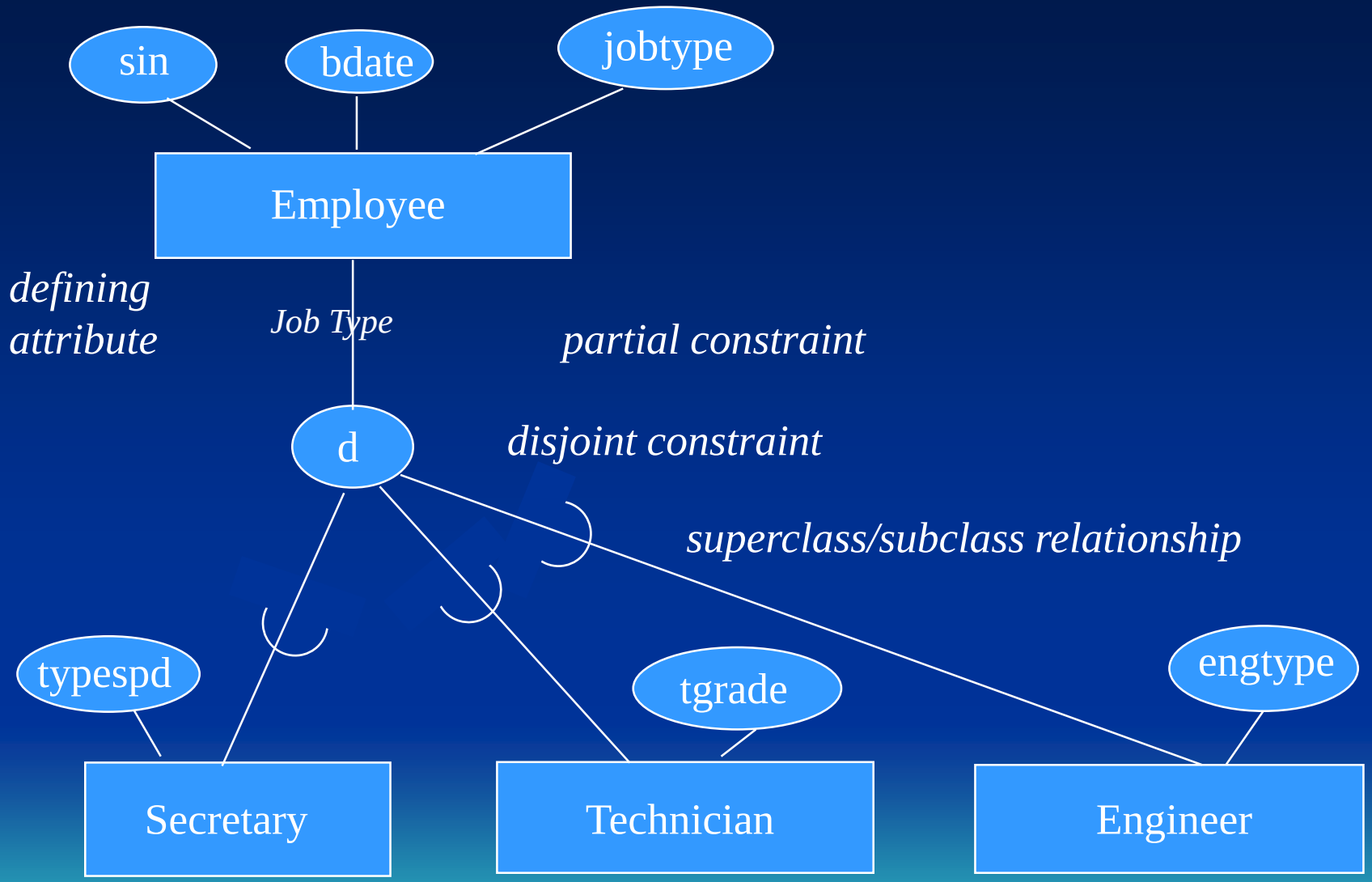
- Overlap
  - the same entity may belong to more than one subclass of a specialization
  - Notation

Completeness Constraint

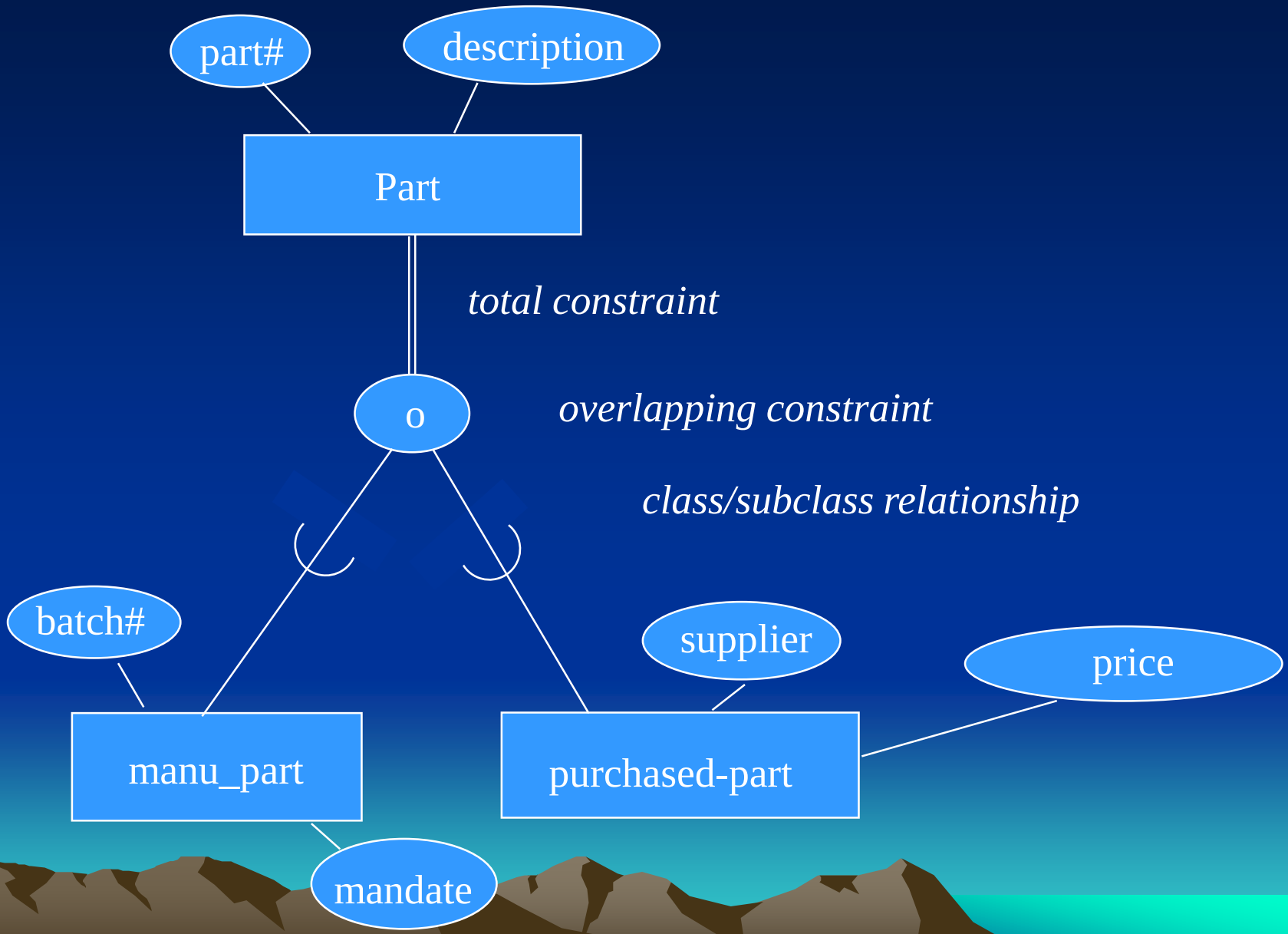
- Total Specialization
  - each entity of a superclass belongs to some subclass of a specialization
  - Notation

- Partial Specialization
  - every entity of a superclass need not belong to a subclass of a specialization
  - Notation

Putting concepts together



Putting concepts together



Insertion, Deletion Rule

- Deleting an entity from a superclass implies that it is automatically deleted from all subclasses it belongs to

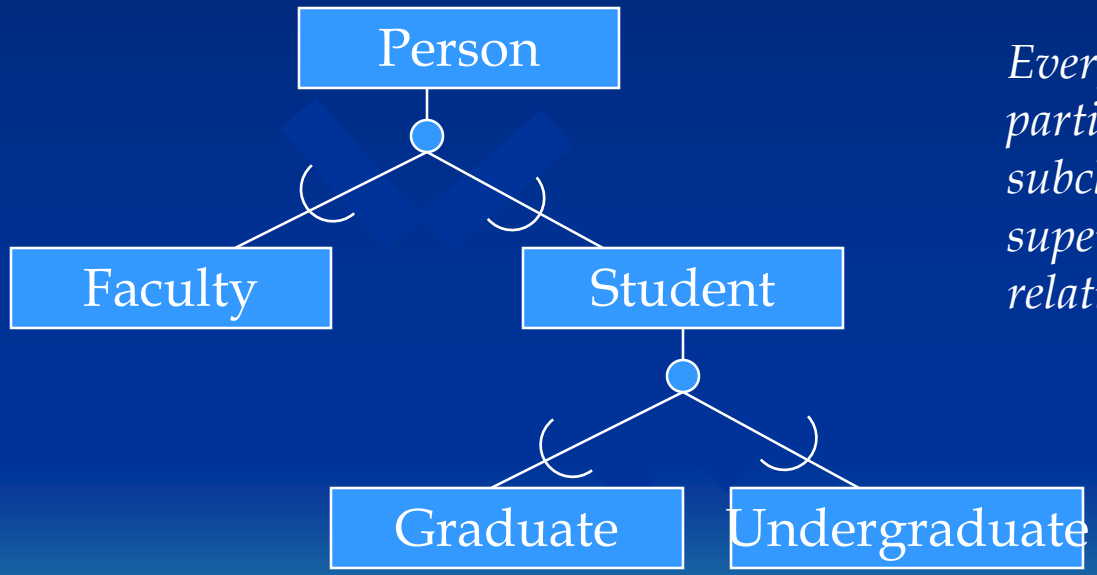
- Inserting an entity into a superclass implies that it is automatically inserted into all predicate-defined subclasses for which it satisfies the condition

  (Ex: *Job Type = 'secretary'*)

- Inserting an entity into a superclass of a total specialization implies that it is automatically inserted into some subclass

Hierarchy

- Hierarchy
  - where a subclass participates in only one class/subclass relationship

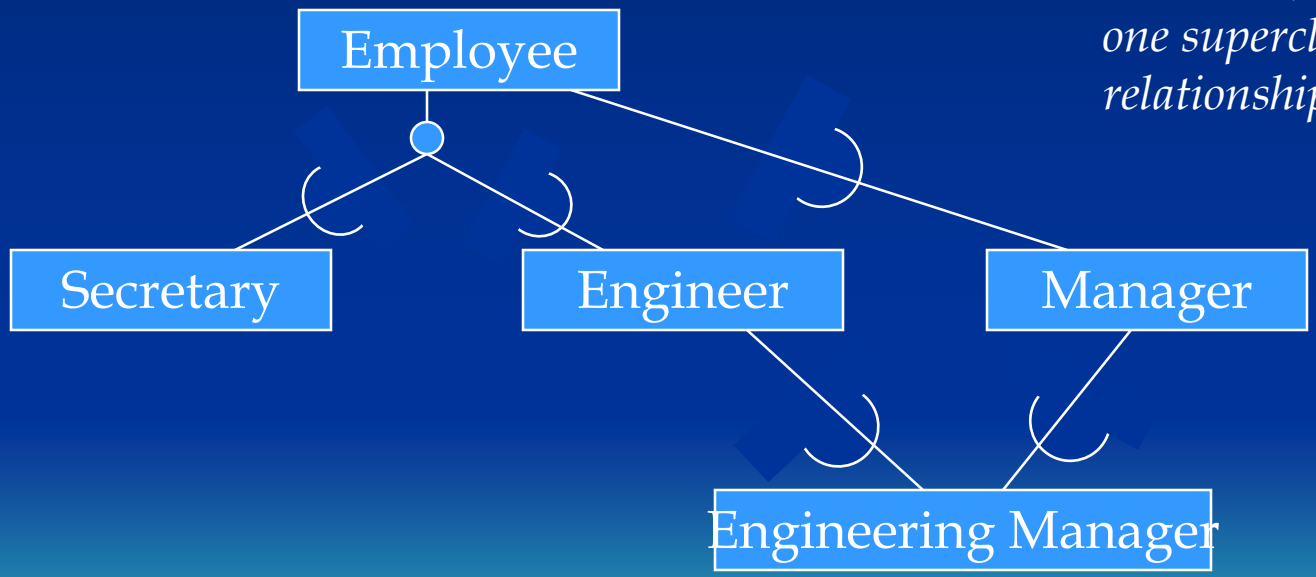


Every subclass participates, as a subclass, in exactly one superclass/subclass relationship

Lattice

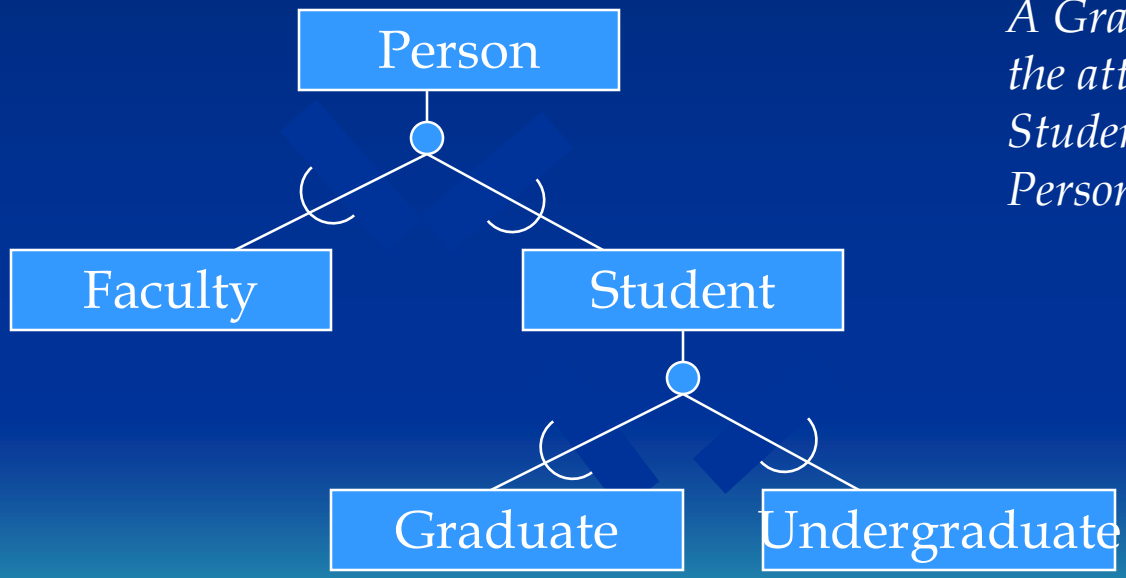
- Lattice
  - where a subclass participates in more than one class/subclass relationship

A subclass participates, as a subclass, in more than one superclass/subclass relationship



Attribute Inheritance

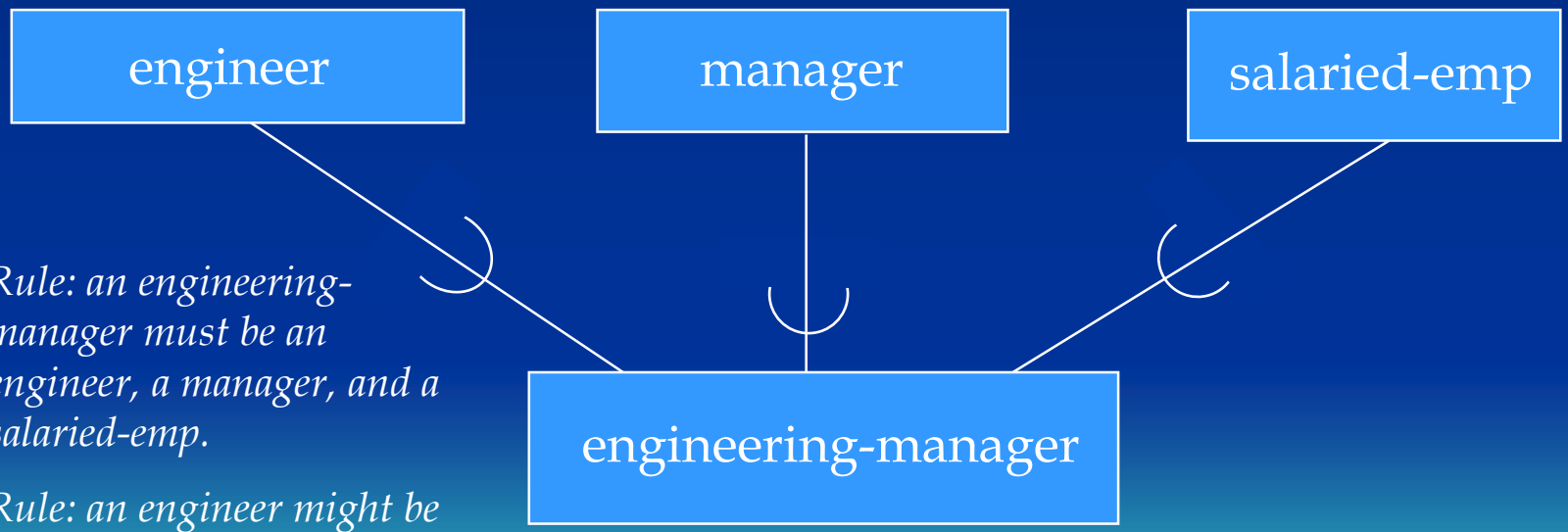
- Attribute Inheritance
  - A subclass inherits attributes not only from its direct superclass, but also from all its predecessor superclasses all the way to the root



A Graduate has all the attributes of a Student and a Person.

Shared Subclass

- Shared SubClass
  - a subclass with more than one superclass
  - leads to the concept of multiple inheritance: engineering manager inherits attributes of engineer, manager, and salaried employee

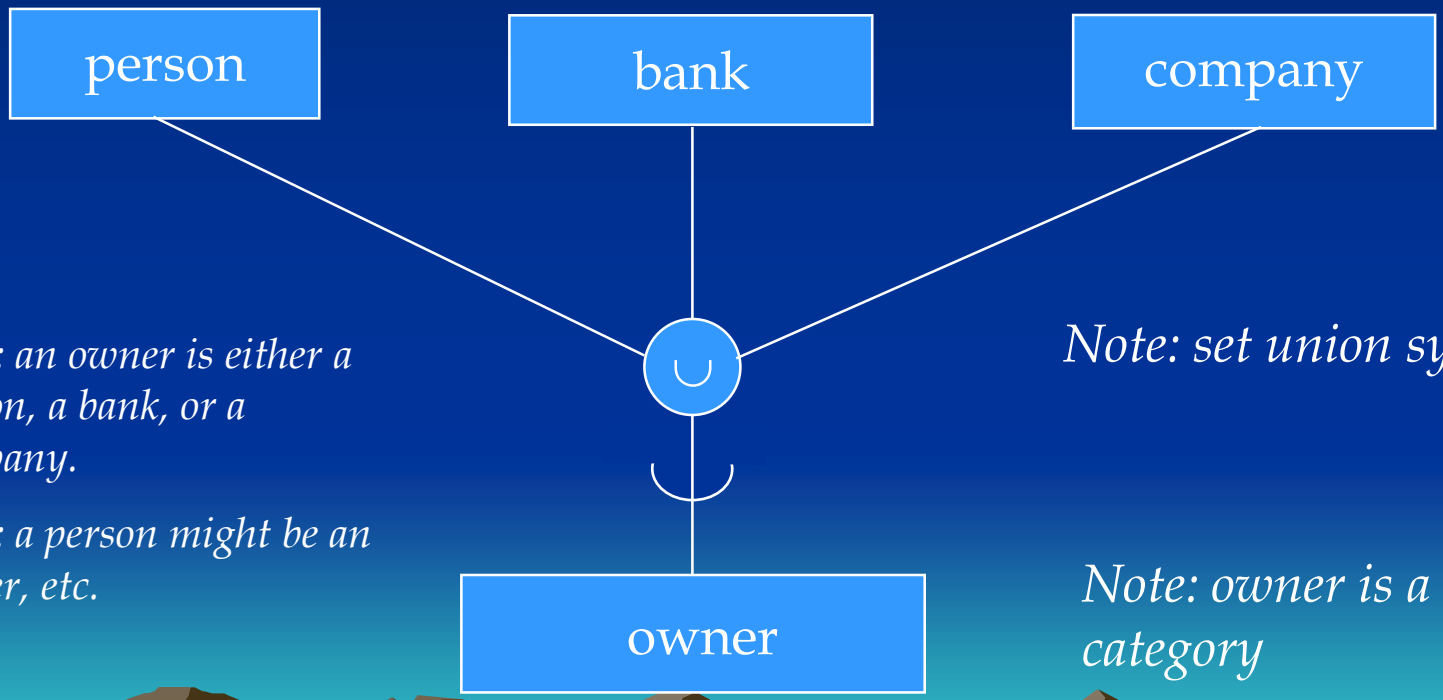


Rule: an engineering-manager must be an engineer, a manager, and a salaried-emp.

Rule: an engineer might be an engineering manager, etc.

Categories

- Models a single class/subclass with more than one super class of different entity types



Rule: an owner is either a person, a bank, or a company.

Rule: a person might be an owner, etc.

Note: set union symbol

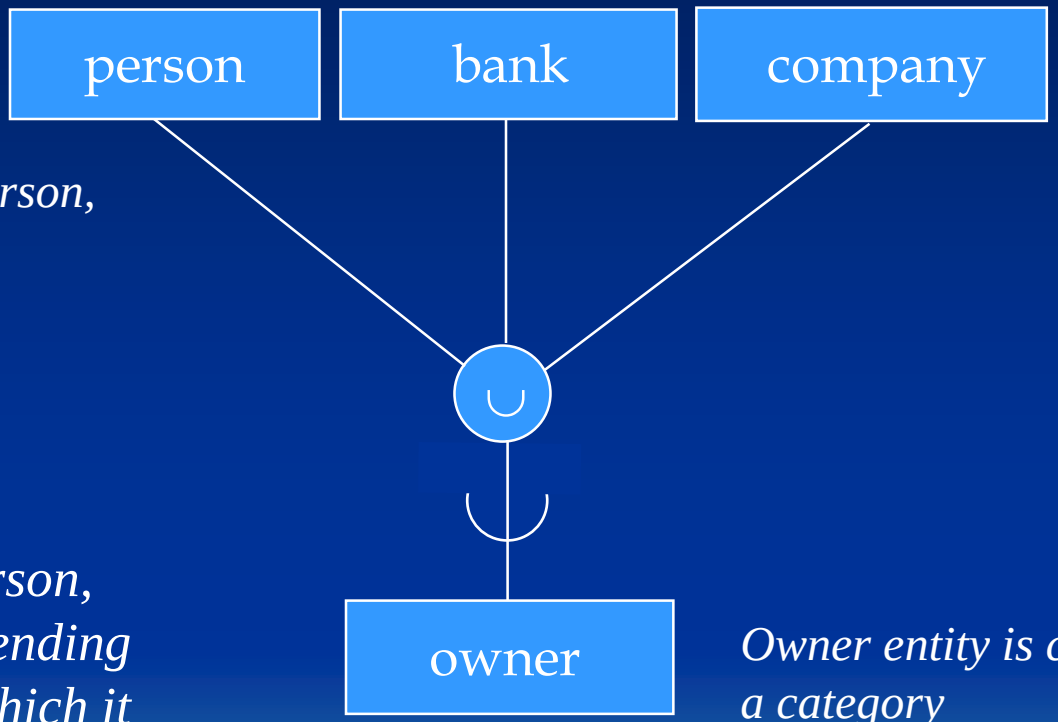
Note: owner is a category

Categories

A category has two or more superclasses that represent distinct entity types

Owner may be either a person, bank, or company

Person, Bank, and Company might have different keys



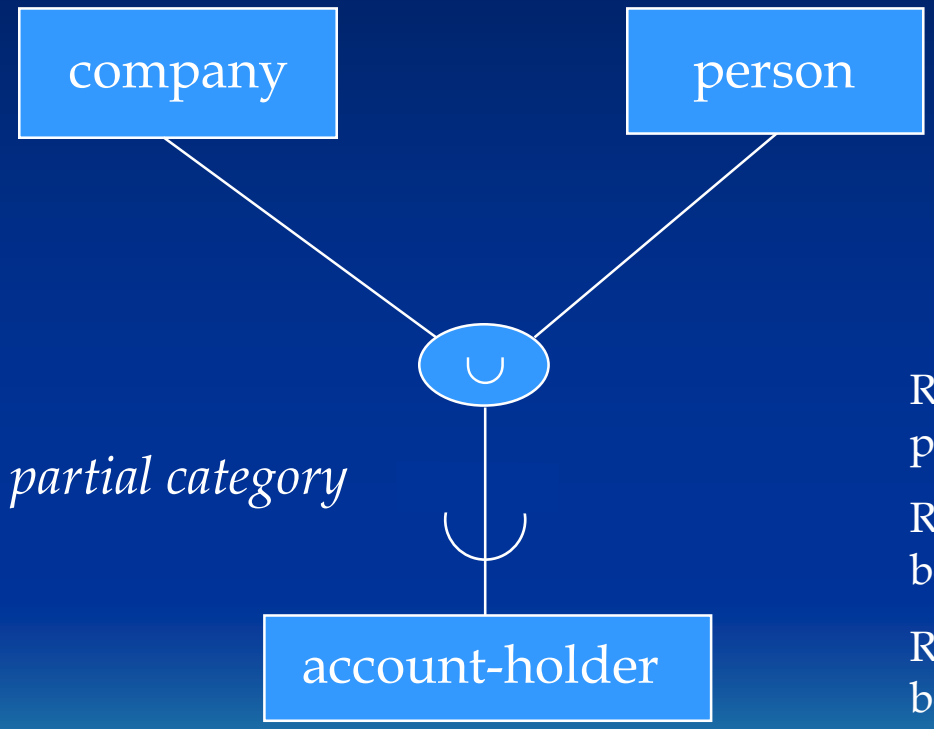
Each owner inherits the attributes of either person, bank or company depending on the superclass to which it belongs - selective inheritance

Owner entity is called a category

Attribute inheritance is selective for categories

Categories

- A category can be either total or partial



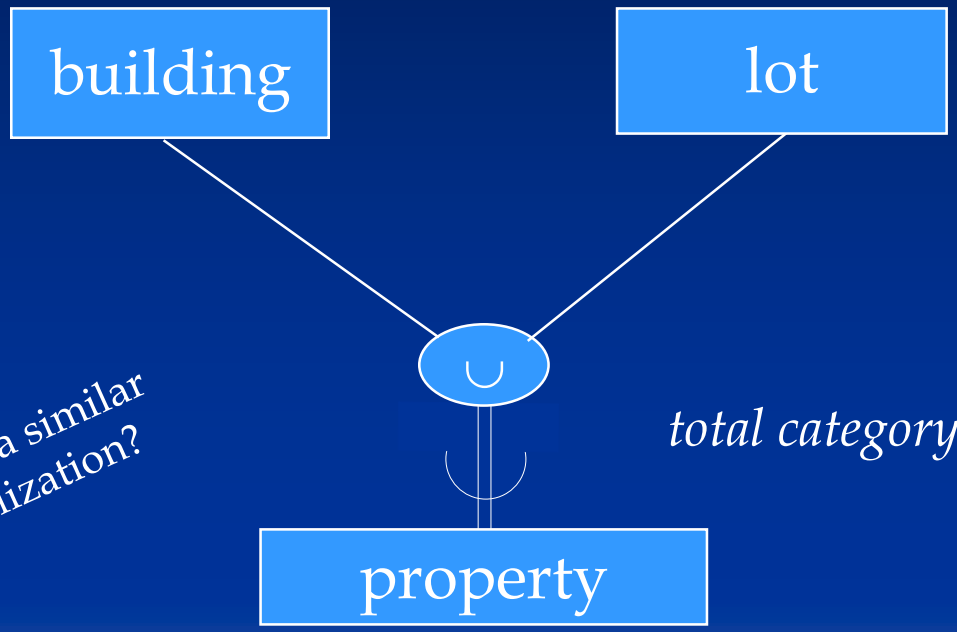
Rule: an account holder is either a person or a company.

Rule: a person may, or may not, be an account owner

Rule: a company may, or may not, be an account holder

Categories

- A category can be either total or partial



Is there a similar generalization?

total category

Rule: a property is either a building or a lot

Rule: a building is a property

Rule: a lot is a property

EER to relational

- Review 7-step algorithm in Section 9.1

1. Create a relation for each strong entity type
   - include all simple attributes
   - choose a primary key
2. Create a relation for each weak entity type
   - include primary key of owner (an FK)
   - PK becomes owner's PK + partial key
3. For each binary *1:1* relationship choose an entity and include the other's PK in it as a FK

4. For each binary *1:n* relationship, choose the *n*-side entity and include a FK wrt the other entity.

5. For each binary *M:N* relationship, create a relation for the relationship
   - include PKs of both participating entities and any attributes of the relationship
   - PK is the catenation of the participating entity PKs

6. For each multi-valued attribute create a new relation
   - include the PK attributes of the entity type
   - PK is the PK of the entity type and the multi-valued attribute

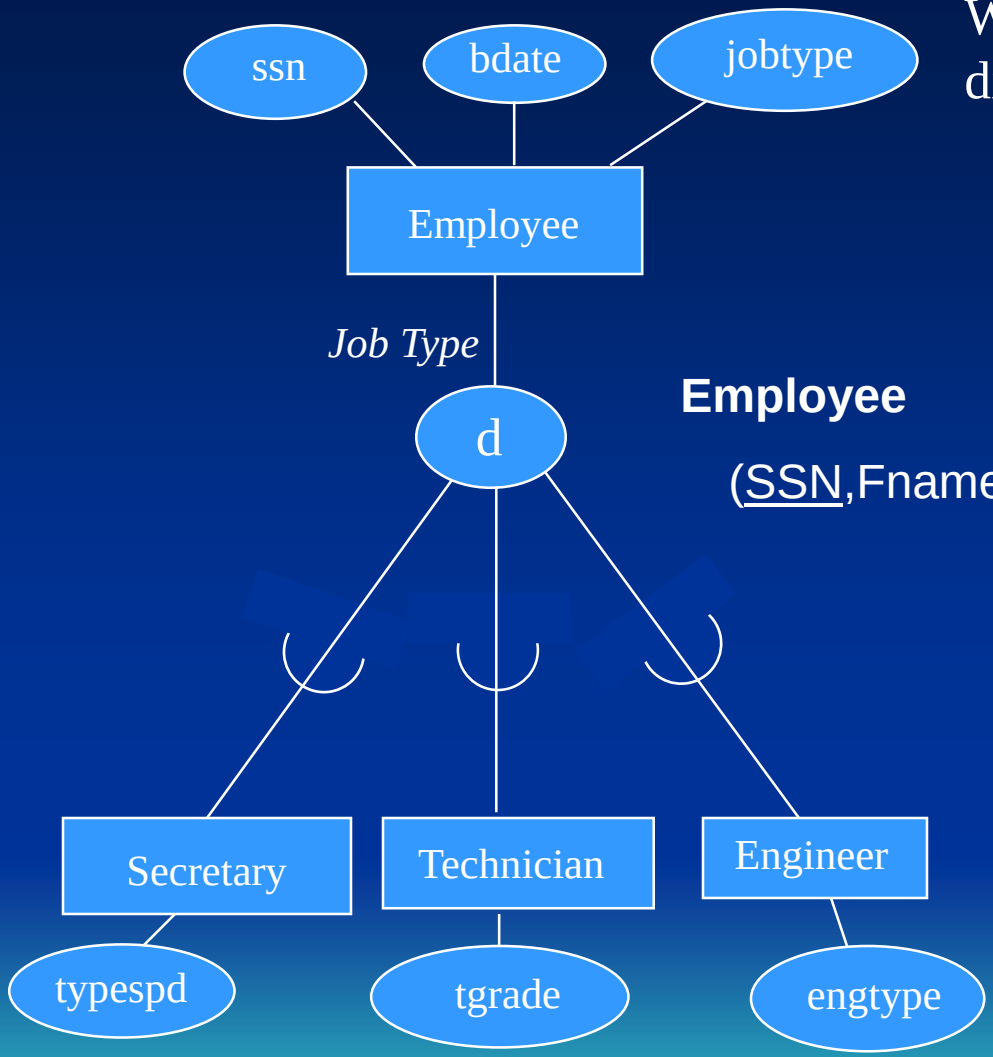
7. For each *n*-ary relationship, create a relation for the relationship
   - include PKs of all participating entities and any attributes of the relationship
   - PK may be the catenation of the participating entity PKs

   (depends on cardinalities)

- Step 8 Conversion of Subclass/Superclasses

- Option A
  - Create a table for the Superclass
  - Create a separate table for each subclass with primary key of superclass

EER to relational



Option A

Works for any kind of constraint: disjoint, overlapping, partial or total

Employee

(SSN,Fname,Minit,Lname,Bdate,Address,JobType)

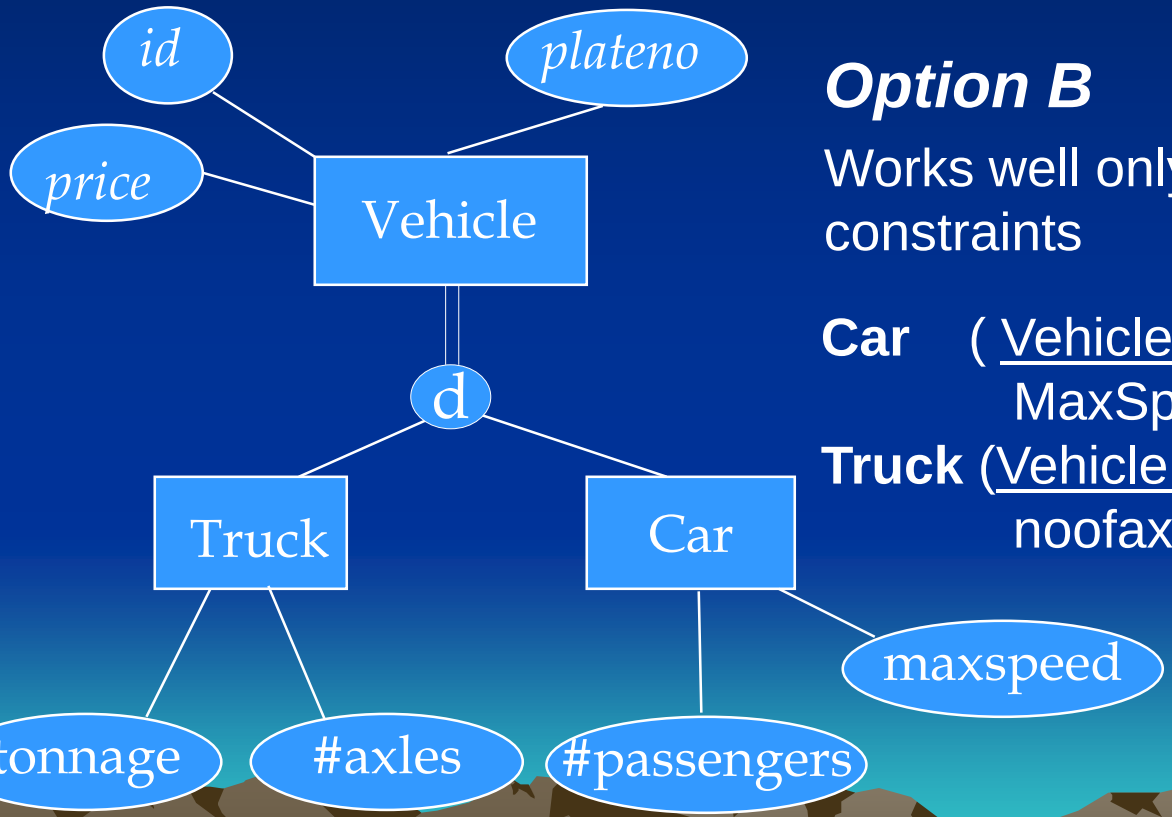
Secretary (SSN, typing Speed)

Technician (SSN, Tgrade)

Engineer (SSN, Engtype)

EER to relational

- Option B
  - Create tables for each subclass, but not for the superclass
  - Move all the attributes of the superclass and include them as attributes of each subclass



Option B

Works well only for disjoint and total constraints

Car (VehicleID, LicensePlateNo, Price, MaxSpeed, NoOfpassengers)

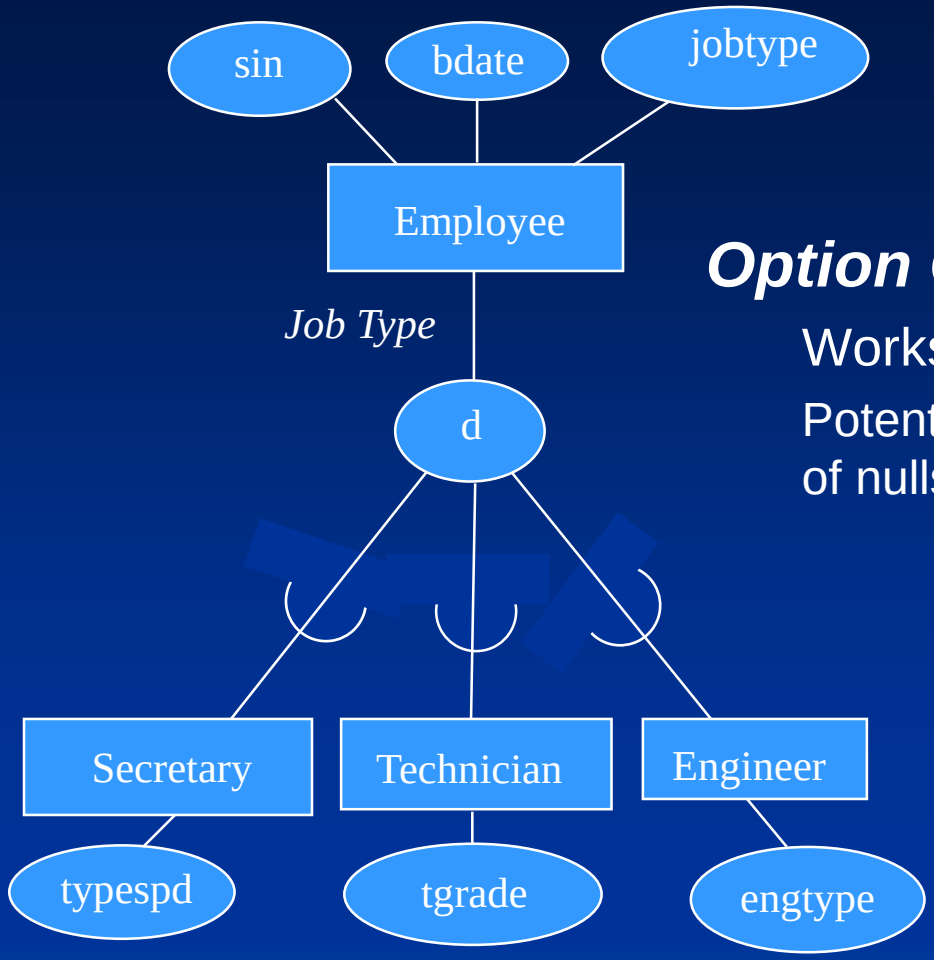
Truck (VehicleID, LicensePlateNo, Price, noofaxles, tonnage)

EER to relational

- ***Option C***
  - Create a single relation with attributes of all the subclasses with a single type attribute as a discriminator
  - Only for disjoint subclasses

- ***Option D***
  - Create a single relation with attributes of all the subclasses and include one flag per subclass
  - Only for overlapping subclasses

EER to relational



Option C

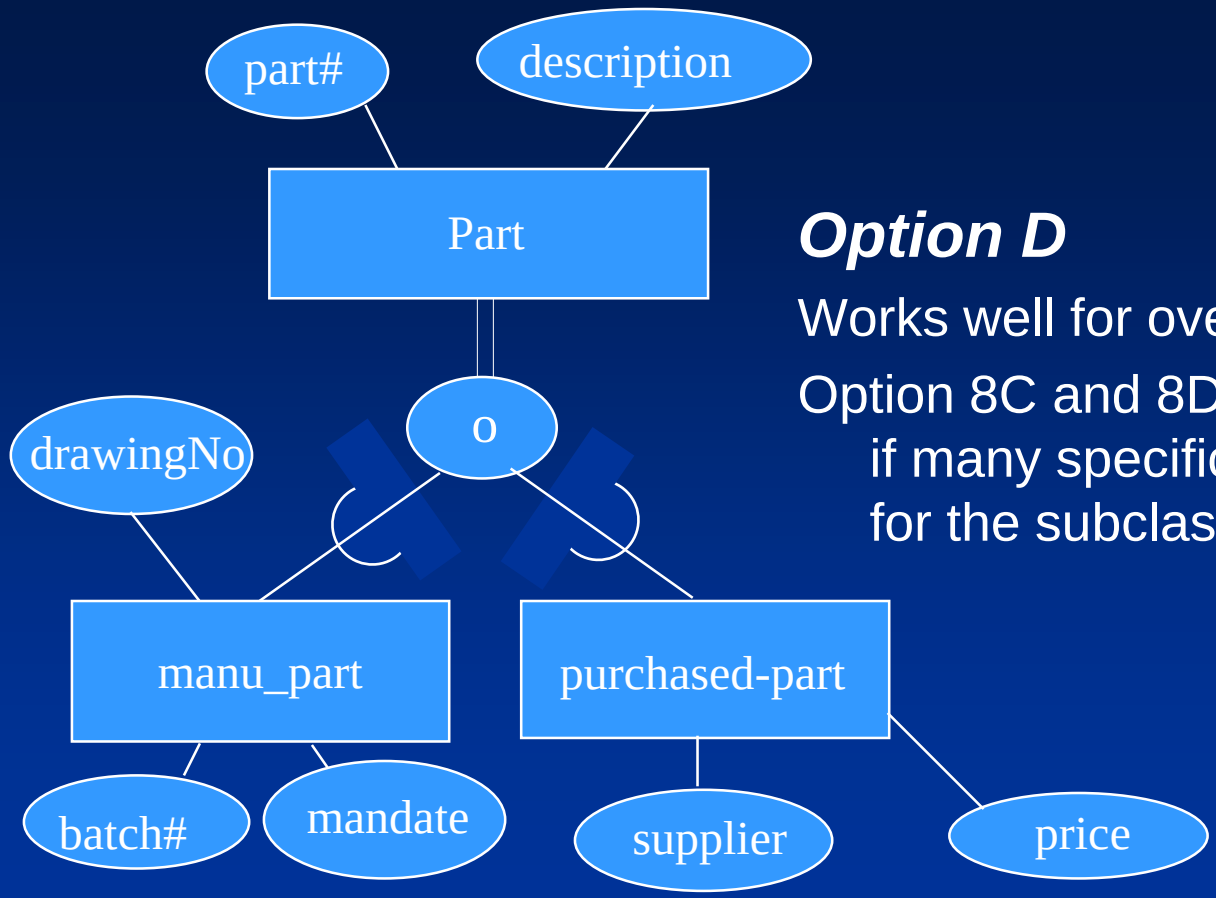
Works well for disjoint constraints

Potential for generating large number of nulls

Employee (SSN, bdate, Address, *JobType*, Typing Speed, Tgrade, EngType)

12345	…	…	**1**	…	…		
56463	…	…	2	…	…		
55554	…		3	…	…		

EER to relational



Option D

Works well for overlapping constraints

Option 8C and 8D are not recommended if many specific attributes are defined for the subclasses

Part (PartNo, Descr, *Mflag*, DrawingNo, ManDate, BatchNo, *Pflag*, SupName, ListPrice)

1	**screw**	**1**	**…**	**…**		**…**	**…**
2	bolt		…	…	1	…	…
3	nail	1	…	…	1		

EER to relational

- Mapping of Categories/Superclasses

  Category:

  - Generate a table for each entity type involved
  - Each table with different key
  - Specify a new key called surrogate key for the category

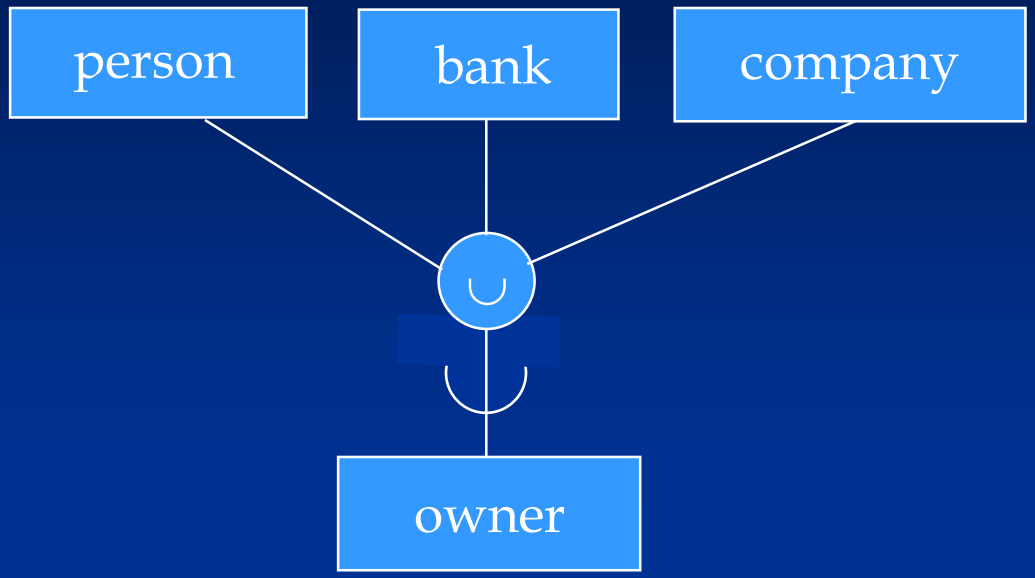
  Superclass

  - Generate a table for each entity type involved
  - All tables with the same keys

    No need of a surrogate key

EER to relational

- Categories - Superclasses with different keys



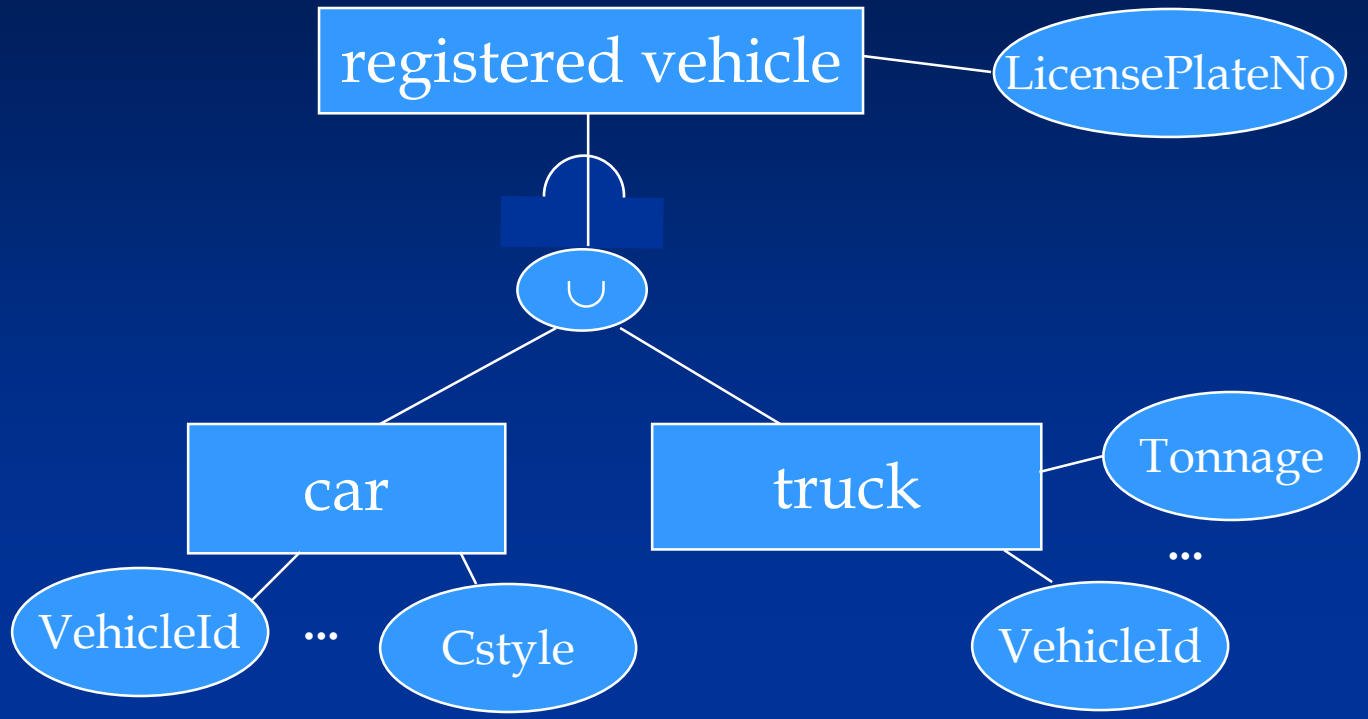
Person (SSN, DrLicNo, Name, Address, *Ownerid*)
Bank (Bname, BAddress, *Ownerid*)
Company (CName, CAddress, *Ownerid*)
Owner (*Ownerid*)

Note the Foreign Keys

Surrogate key

EER to relational

- Categories - Superclasses with the same keys



Registered Vehicle (*VehicleID*, LicensePlateNo,)

Car (*VehicleID* , Cstyle, CMake, CModel,CYear)

Truck (*VehicleID* , TMake, TModel,TYear, Tonnage)

Note there are no Foreign Keys

Classification and Instantiation

- Classification - Process of systematically assigning similar objects to object classes
  - from employee objects to Employee Class
- Instantiation - Refers to the generation and specific examination of distinct objects of a class
  - employee objects of the Employee Class

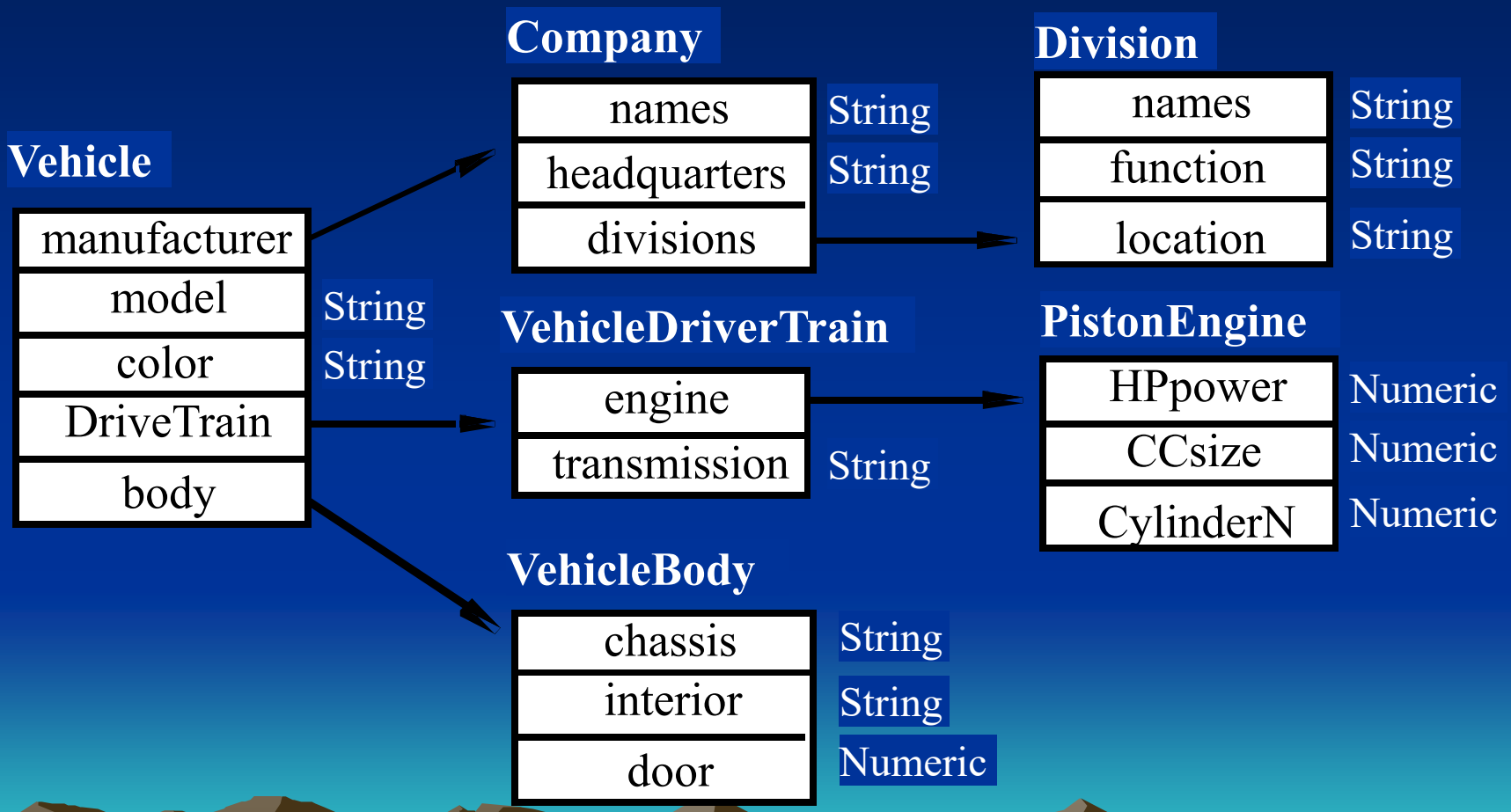
Opposites of one another

- Knowledge based systems (KR)
  - Classes can be an instance of another class (meta-classes)
- EERD
  - Only super/subclass association is possible

Aggregation and Association

- Aggregation - Concept of building composite objects from their components
  - EERD: Aggregating attributes into an entity
- Association - Associate objects from several independent classes
  - EERD: relationships between different entities
- Aggregation vs Association
  - Delete an aggregate object involves deleting its components
  - Deleting an association does not involve deleting its participating objects

Aggregation:



Aside: UML

Unified Modeling Language (1994+)

- Booch, Rumbaugh, Jacobson @ Rational Software
- for modeling systems
- concepts learned here apply to UML

Note: there are many modeling notations: Chen, IE, IDEF1X,UML