Fall 2019

CS 220
Database Systems

Lecture 9

Enhanced Entity Relationship Modelling

E-R Modeling Capabilities

Entity Relationship Models are normally adequate for building data models of traditional, administrative based database systems such as

Stock control

Product ordering

Customer invoicing.

Evolution of Databases

Designers of database applications have tried to design more accurate database schemas that reflect the data properties and constraints more precisely.

This was particularly important for newer applications of database technology, such as databases for

- Engineering design and manufacturing
- Telecommunications
- Complex software systems
- Geographic information systems (GISs)

Enhanced Entity-Relationship Modeling

Enhanced Entity-Relationship (EER) modeling is an extension of ER modeling to include object-oriented concepts such as:

- ➔ superclasses and subclasses
- ➔ specialization and generalization
- ➔ aggregation and composition

Superclasses & Subclasses

A **superclass** is a general class that is extended by one or more subclasses.

A **subclass** is a more specific class that extends a superclass by inheriting its methods and attributes and then adding its own methods and attributes.

Inheritance is the process of a subclass inheriting all the methods and attributes of a superclass.

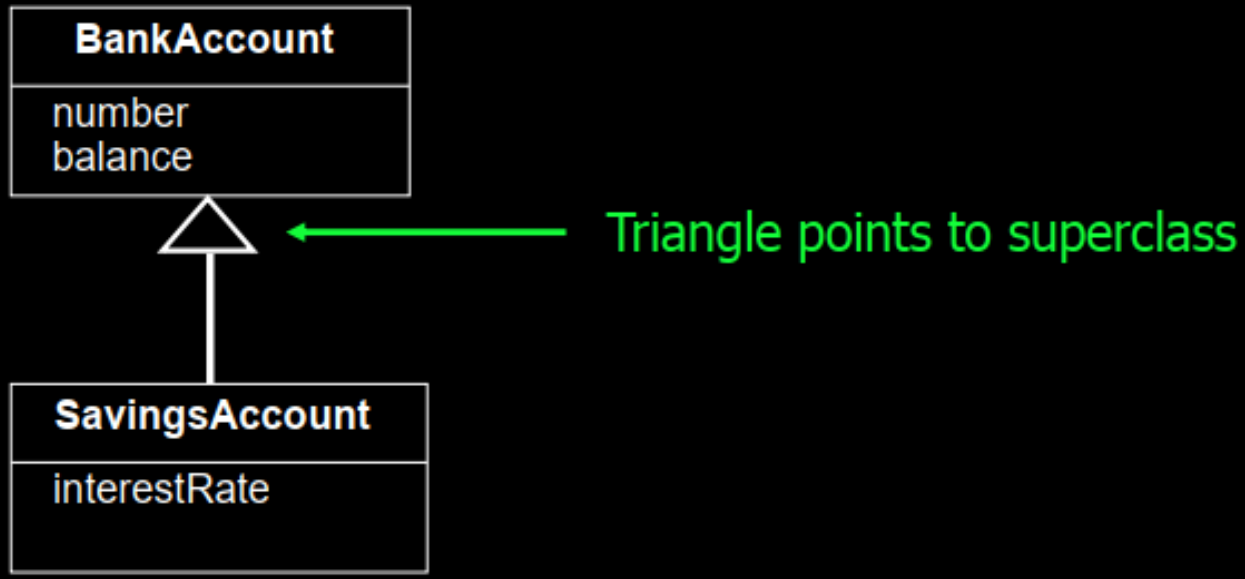
Subclasses

Entity type has numerous subgroupings or subtypes of its entities that are meaningful and need to be represented explicitly because of their significance to the database application

EMPLOYEE entity type may be distinguished further into

- Secretary, Engineer, Technician
- Manage, Director
- Salaried employee, Hourly employee

UML class diagram:



When to Use EER Modeling?

It is important to emphasize that many database projects do not need the object-oriented modeling features of EER modeling.

Remember the goal of conceptual modeling is to produce a model that is simple and easy to understand.

Do not introduce complicated subclass/superclass relationships if they are not needed.

Only use the EER modeling constructs if they offer a significant advantage over regular ER modeling

Example Employee Relation

eno	ename	bdate	title	salary	supereno	dno
E1	J. Doe	01-05-75	EE	30000	E2	null
E2	M. Smith	06-04-66	SA	50000	E5	D3
E3	A. Lee	07-05-66	ME	40000	E7	D2
E4	J. Miller	09-01-50	PR	20000	E6	D3
E5	B. Casey	12-25-71	SA	50000	E8	D3
E6	L. Chu	11-30-65	EE	30000	E7	D2
E7	R. Davis	09-08-77	ME	40000	E8	D1
E8	J. Jones	10-11-72	SA	50000	null	D1

title attribute indicates what job the employee does at the company. Consider if each job title had its own unique information that we would want to record such as:

programming language used (lang), DB used (db)

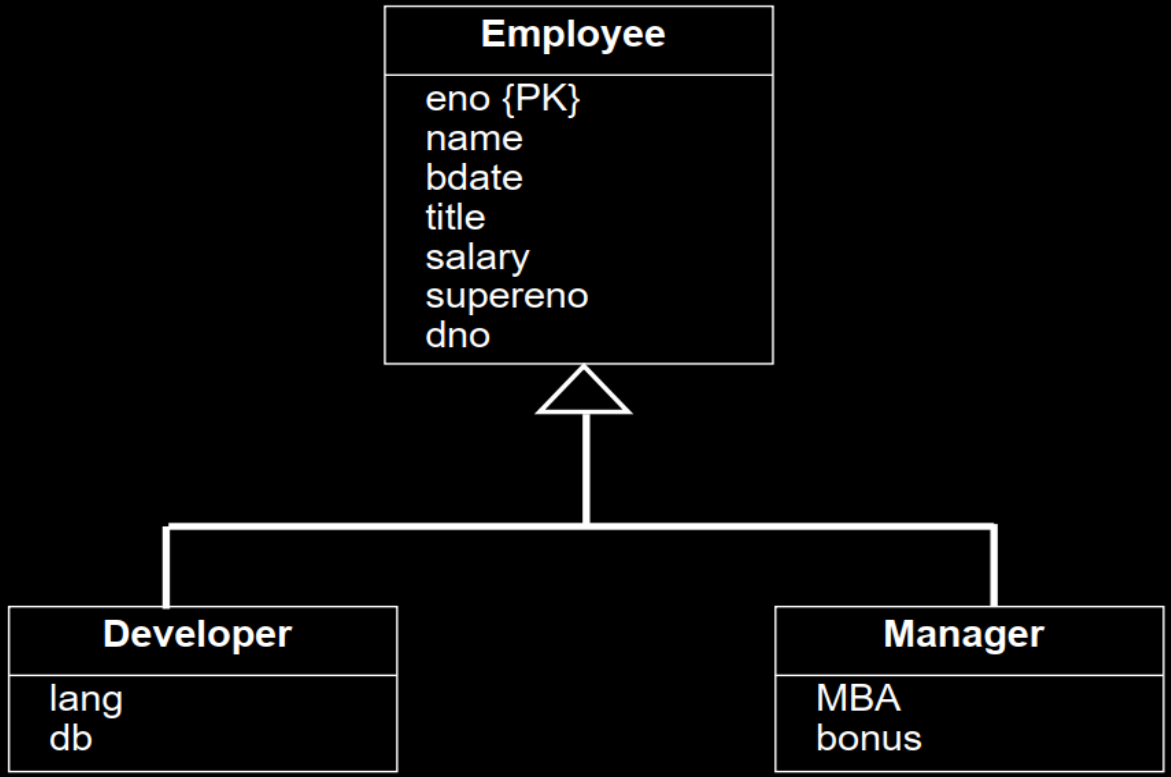
Example Employee Relation

We could represent all these attributes in a single relation:

eno	ename	bdate	title	salary	supereno	dno	lang	db	MBA	bonus
E1	J. Doe	01-05-75	EE	30000	E2		C++	MySQL		
E2	M. Smith	06-04-66	SA	50000	E5	D3			N	2000
E3	A. Lee	07-05-66	ME	40000	E7	D2			N	3000
E4	J. Miller	09-01-50	PR	20000	E6	D3	Java	Oracle		
E5	B. Casey	12-25-71	SA	50000	E8	D3			Y	4000
E6	L. Chu	11-30-65	EE	30000	E7	D2	C++	DB2		
E7	R. Davis	09-08-77	ME	40000	E8	D1			N	3000
E8	J. Jones	10-11-72	SA	50000		D1			Y	6000

Note the wasted space as attributes that do not apply to a particular subclass are NULL

Example Employee Relation



A better solution would be to make two subclasses of *Employee* called *Developer* and *Manager*

Example Employee Relation

Employee Relation

eno	ename	bdate	title	salary	supereno	dno
E1	J. Doe	01-05-75	EE	30000	E2	null
E2	M. Smith	06-04-66	SA	50000	E5	D3
E3	A. Lee	07-05-66	ME	40000	E7	D2
E4	J. Miller	09-01-50	PR	20000	E6	D3
E5	B. Casey	12-25-71	SA	50000	E8	D3
E6	L. Chu	11-30-65	EE	30000	E7	D2
E7	R. Davis	09-08-77	ME	40000	E8	D1
E8	J. Jones	10-11-72	SA	50000	null	D1

Manager Relation

eno	lang	db
E1	C++	MySQL
E4	Java	Oracle
E6	C++	DB2

Developer Relation

eno	MBA	bonus
E2	N	2000
E3	N	3000
E5	Y	4000
E7	N	3000
E8	Y	6000

Generalization & Specialization

Subclasses and superclasses are created by using either generalization or specialization.

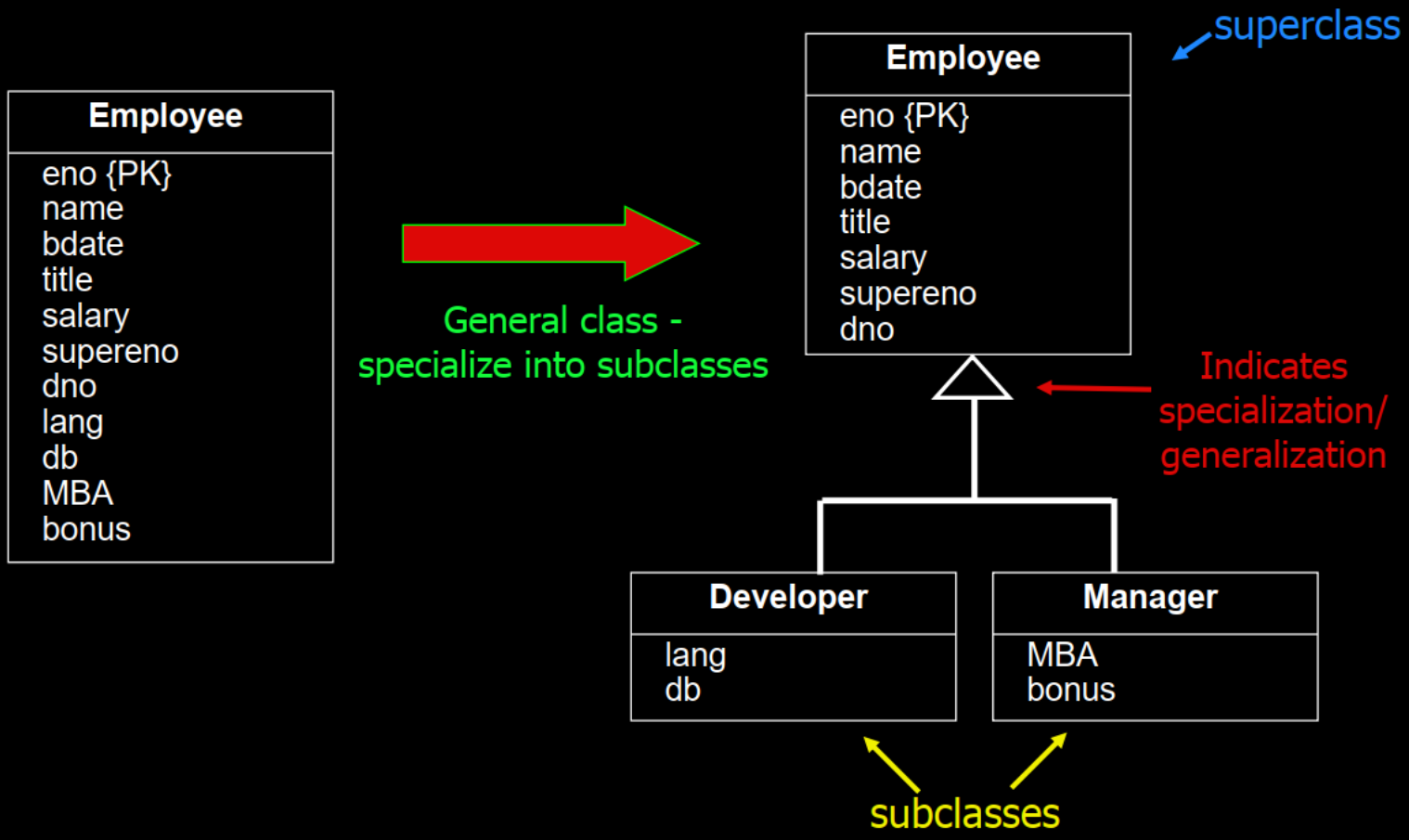
Specialization is the process of creating more specialized subclasses of an existing superclass.

- Top-down process: Start with a general class and then subdivide it into more specialized classes.
  - The specialized classes may contain their own attributes. Attributes common to all subclasses remain in the superclass.

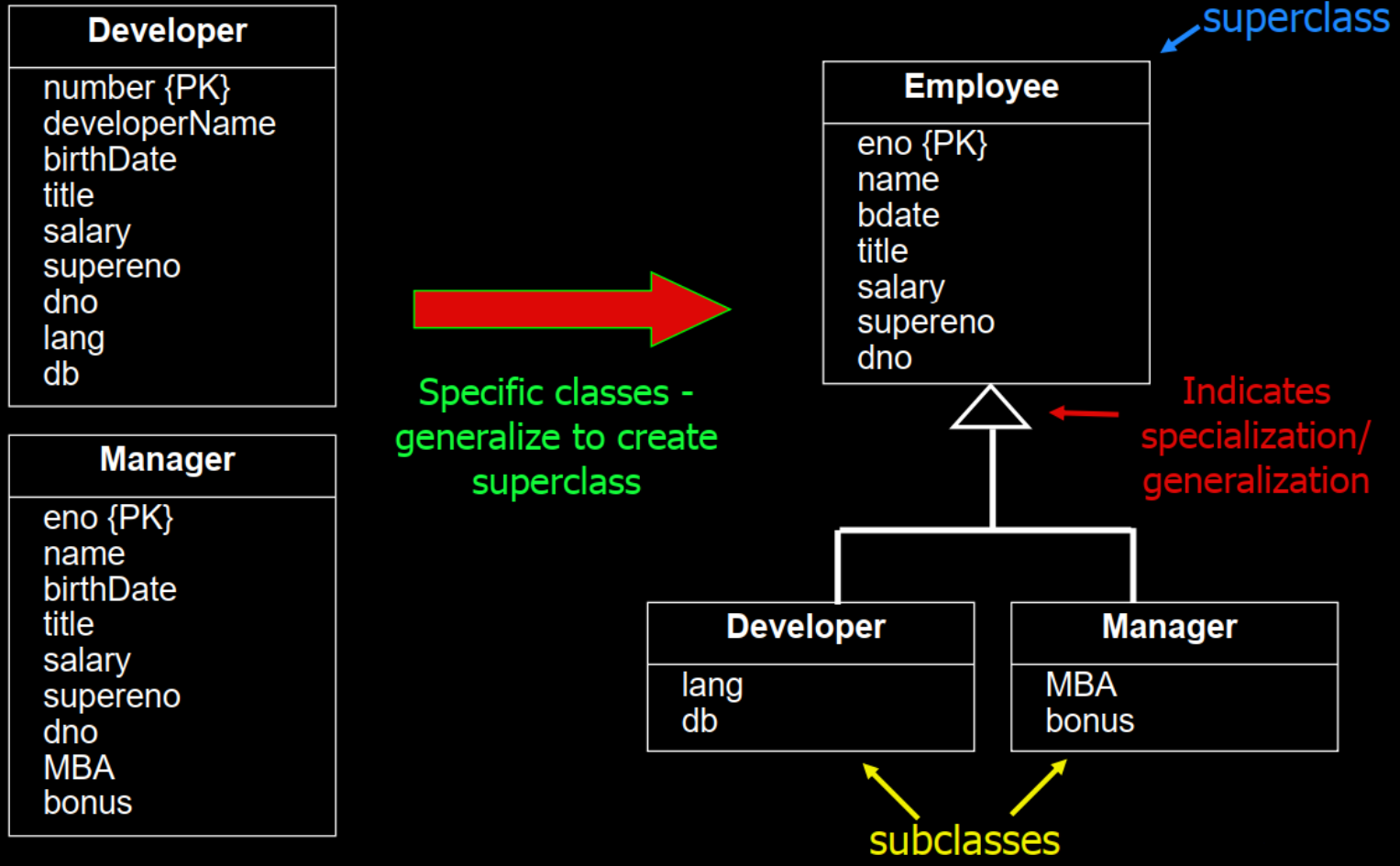
Generalization is the process of creating a more general superclass from existing subclasses.

- Bottom-up process: Start with specialized classes and try to determine a general class that contains the attributes common to all of them.

Specialization



Generalization



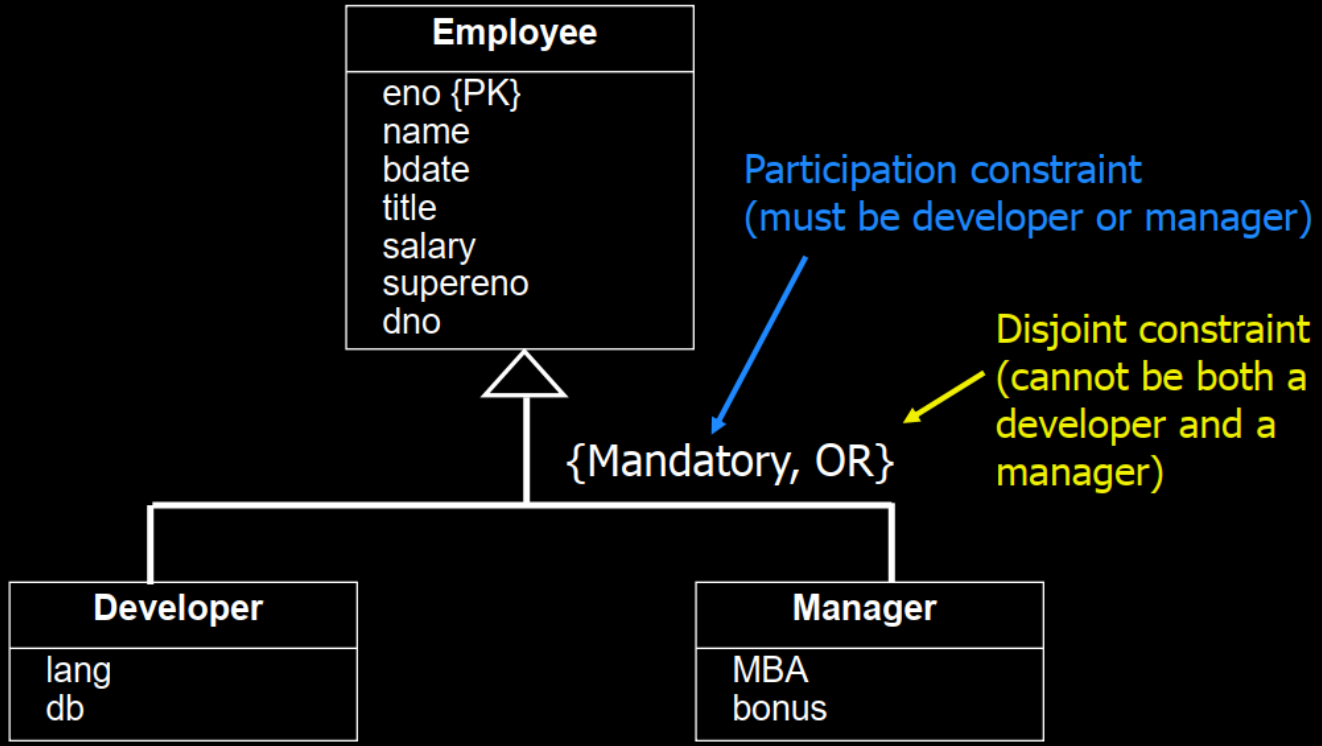
Constraints on Generalization & Specialization

There are two types of constraints associated with generalization and specialization:

- ***Participation constraint*** - determines if every member in a superclass must participate as a member of one of its subclasses.
  - ⇨ It may be `optional` for a superclass member to be a member of one of its subclasses, or it may be `mandatory` that a superclass member be a member of one of its subclasses.

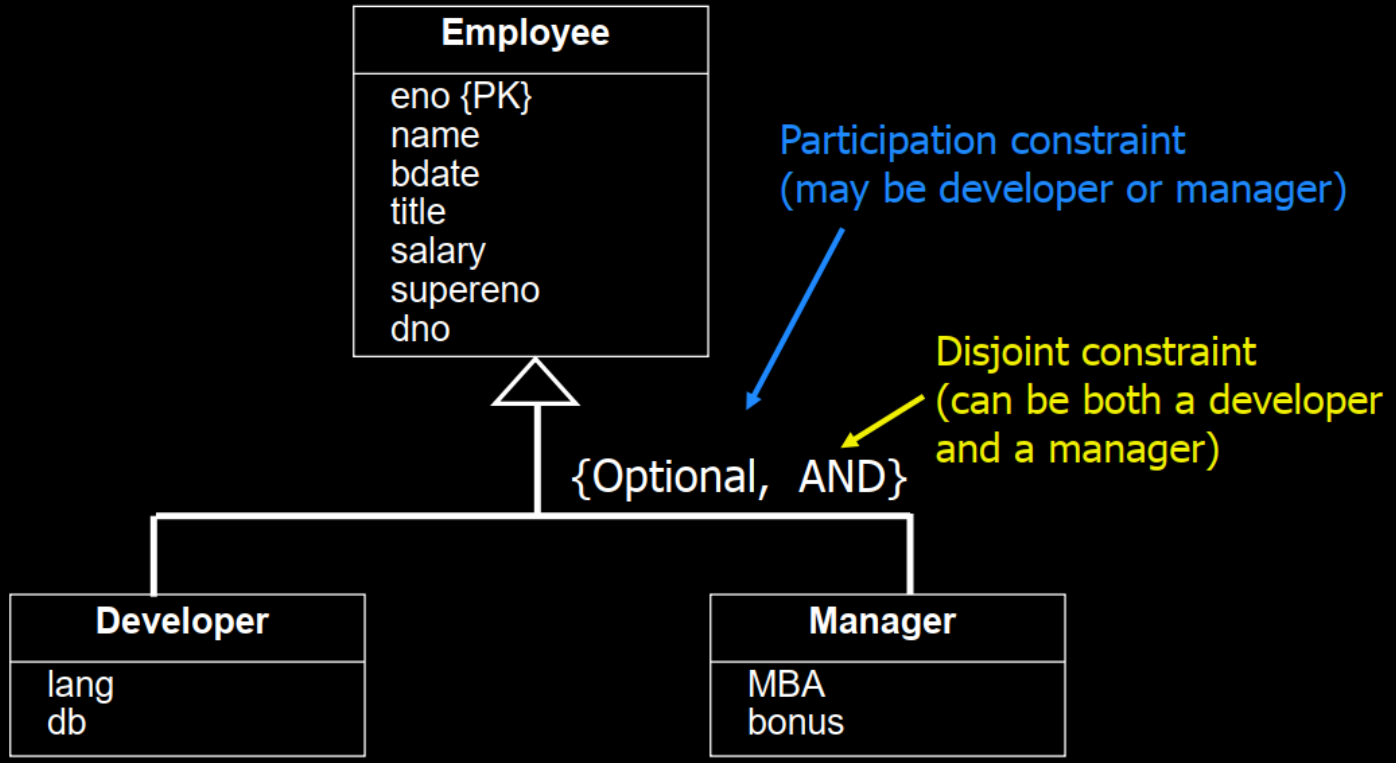
- ***Disjoint constraint*** - determines if a member of a superclass can be a member of one or more than one of its subclasses.
  - ⇨ If a superclass object may be a member of only one of its subclasses this is denoted by `OR` (subclasses are *disjoint*).
  - ⇨ Otherwise, `AND` is used to indicate that it may be in more than one of its subclasses.

Constraints Example



An employee must be either a **developer** or a **manager**, but *cannot be both*.

Constraints Example



An employee may specialize as a developer or manager.
An employee may be both a manager and developer.

Time to Think !!!

Note: What is the participation and the disjoint constraints for superclass `Employee` (with subclasses `Manager` and `Supervisor`) given these instances?

Relationship Constraints

- Minimum # of occurrences – called participation constraint in both cases
- Maximum # of occurrences – called cardinality constraint for relationships and disjoint constraint for subclasses

Combining Inheritance Constraints & Relationship Constraints

Possible combinations:

Subclass Constraints	Relationship Constraints
Optional, AND	0..*
Optional, OR	0..1
Mandatory, AND	1..*
Mandatory, OR	1..1

More on Generalization & Specialization

Predicate-defined constraints specify when an object participates in a subclass using a certain rule.

- For example, a subclass called `RichEmployees` can be defined with a membership predicate such as `salary >100000`.

Attribute-defined subclasses are a particular type of predicate-defined constraint where the value of an attribute(s) determines if an object is a member of a subclass.

- For example, the `title` field could be used as a *defining attribute* for the `Developer` and `Manager` subclasses.
  - `Emp` is in `Developer` if `title = 'EE' or 'PR'`
  - `Emp` is in `Manager` if `title = 'ME' or 'SA'`

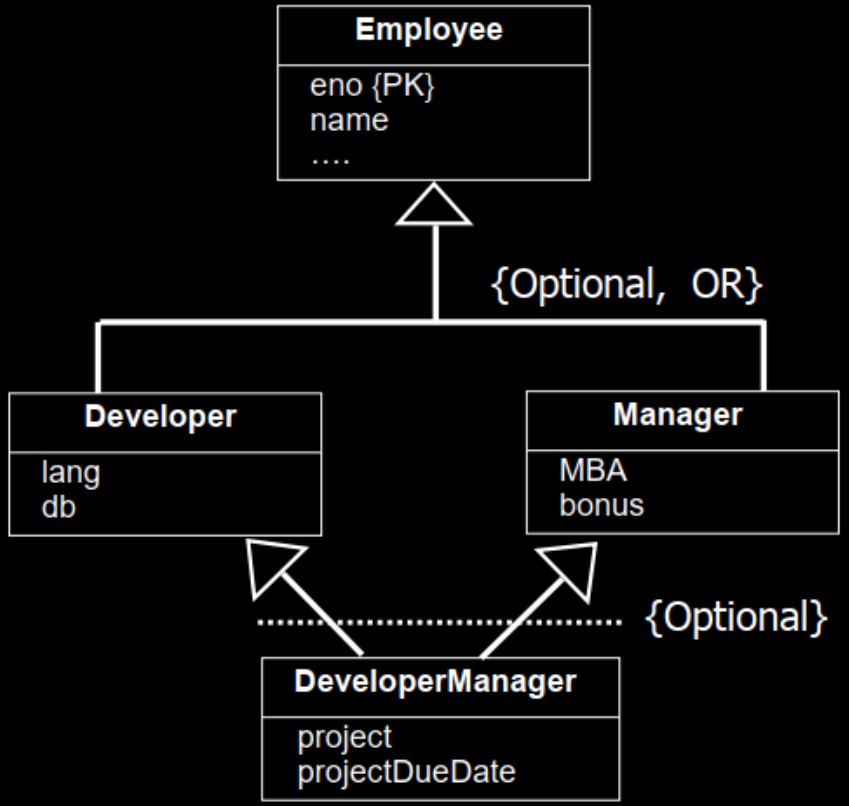
Multiple Inheritance

If each class only has one superclass, then the class diagram is said to be a ***specialization*** or ***type hierarchy***.

If a class may have more than one superclass, then the class diagram is said to be a ***specialization*** or ***type lattice***.

Although multiple inheritance is powerful, it should be avoided if possible.

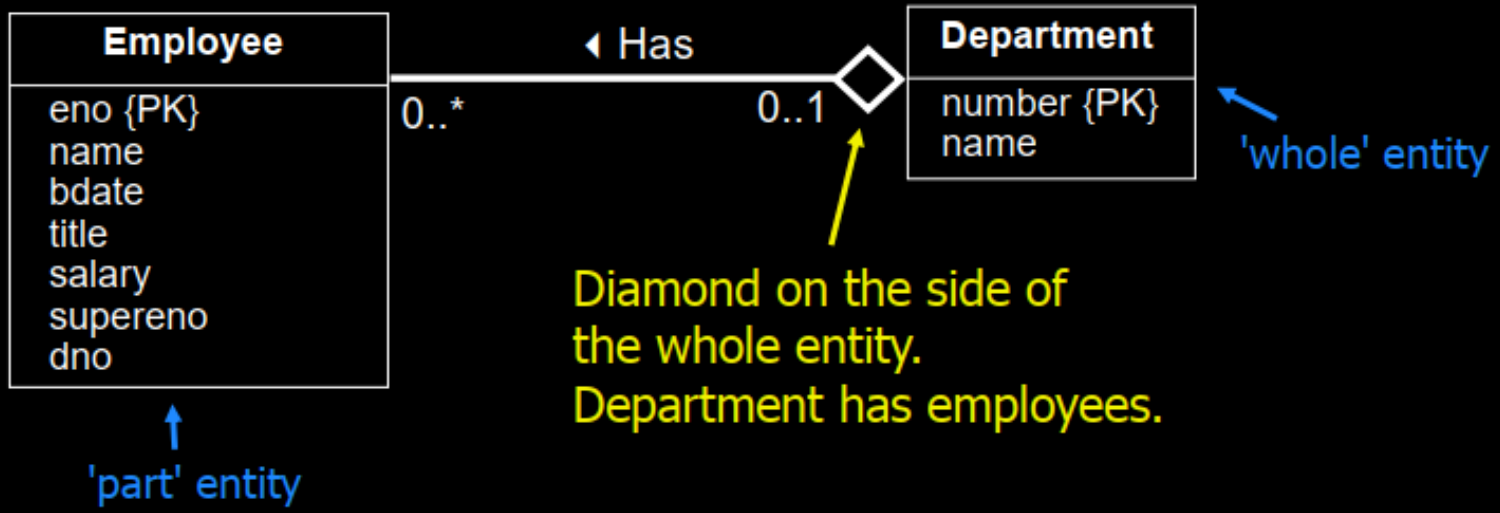
Multiple Inheritance



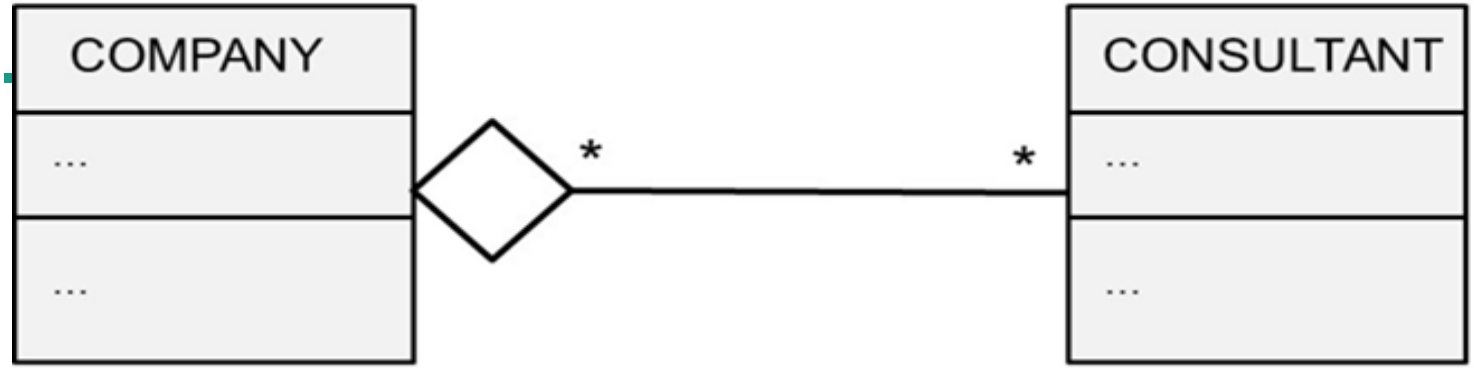
Aggregation

Aggregation represents a `'HAS-A'` or `'IS-PART-OF'` relationship between entity types. One entity type is the *whole*, the other is the *part*.

Example:



Aggregation



–part object can simultaneously belong to multiple composite objects

–maximum multiplicity at the composite side is undetermined

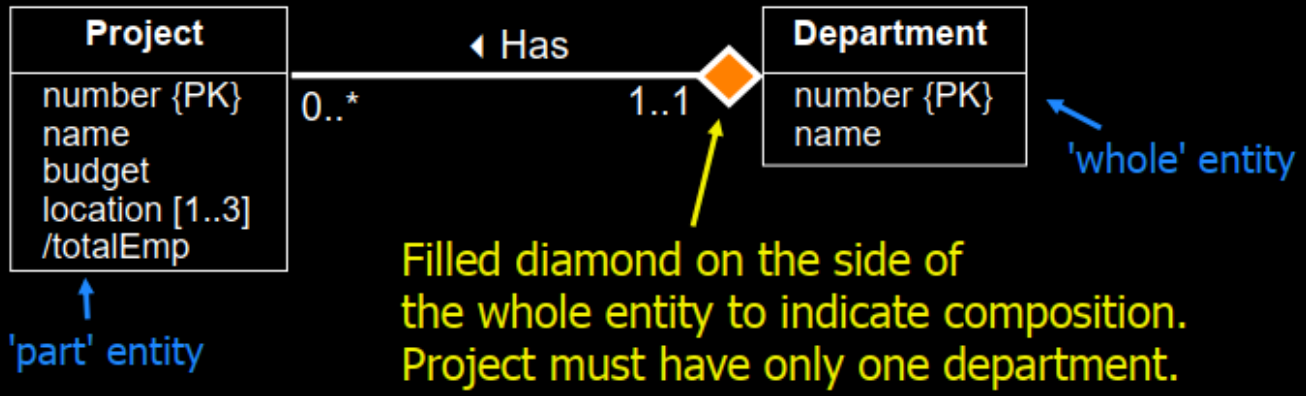
–part object can also occur without belonging to a composite object

–loose coupling

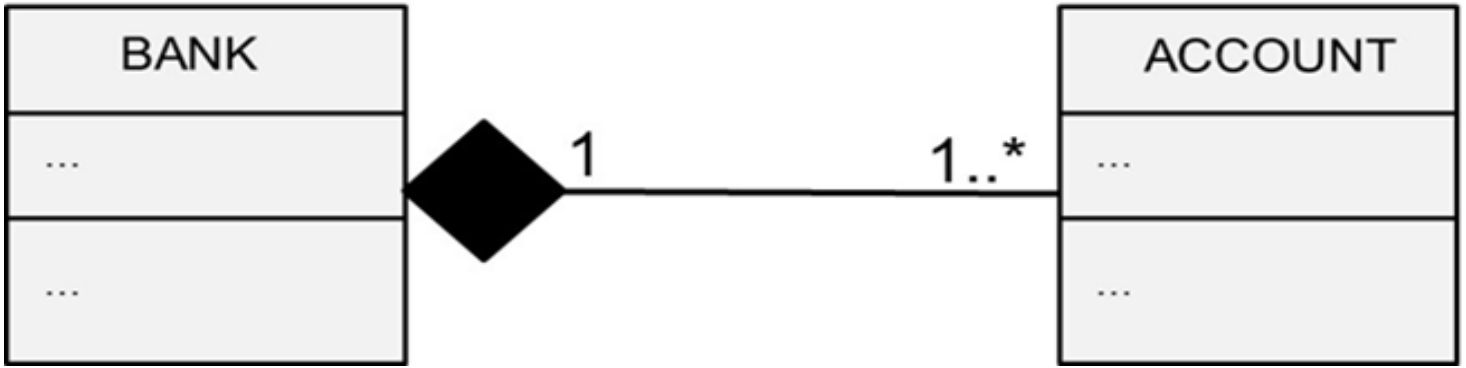
Composition

Composition is a stronger form of aggregation where the part cannot exist without its containing whole entity type and the part can only be part of one entity type.

Example:



Composition



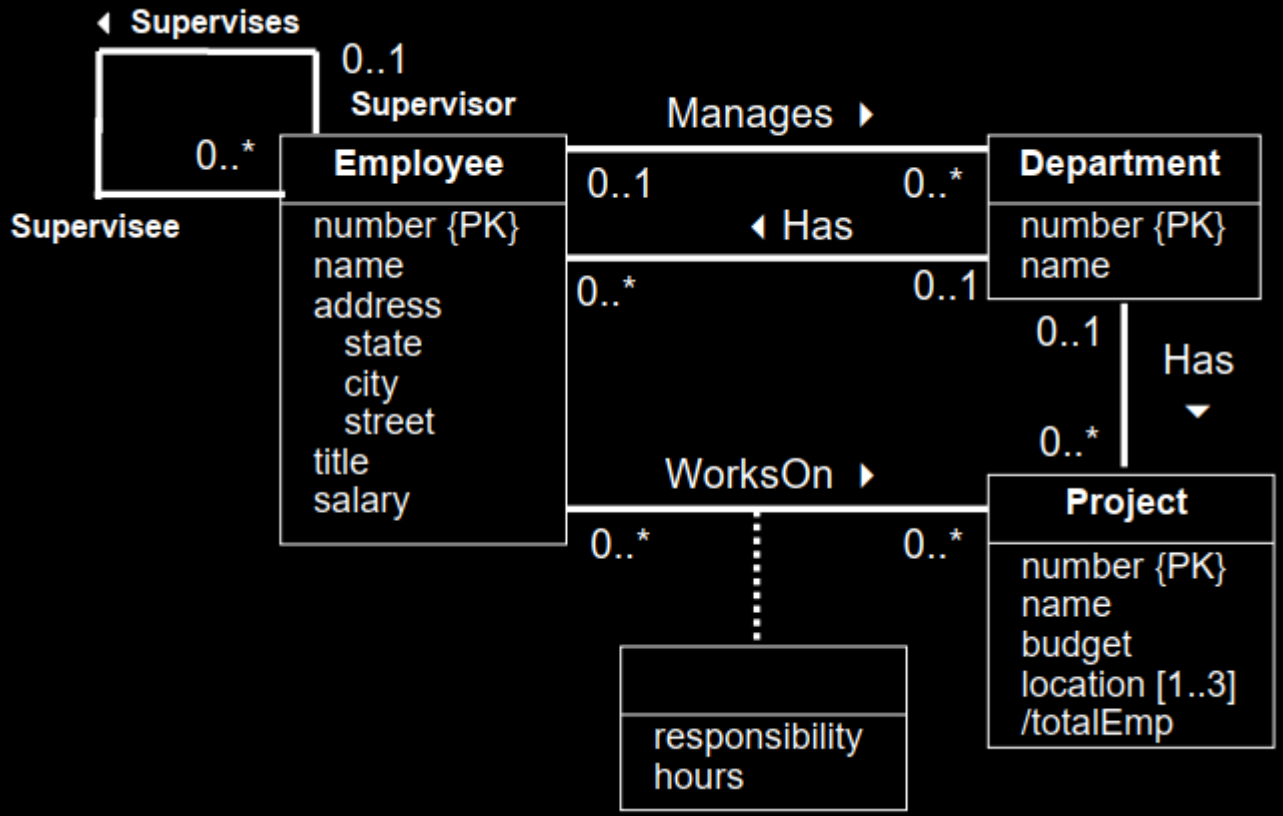
–the part object can only belong to one composite

–maximum multiplicity at the composite side is 1

–minimum multiplicity can be either 1 or 0

–tight coupling

ER Model - Example



EER Model - Example