Entrepôts de Données et Big-Data

Équipe Pédagogique

- Anne-Muriel Arigon - UM, LIRMM - Responsable UE
  (chifolleau@lirmm.fr)

- Federico Ulliana - INRIA, UM
  (federico.ulliana@inria.fr)

Special Guest :

- Christophe Menichetti - HPE (Hewlett Packard), IBM

Objectif du Module

vers le

BIG-DATA ANALYTICS

Plus concrètement, présenter

1. *les techniques de modélisation*
2. *les systèmes et technologies*

conçues pour l'analyse des données massives

Programme

1. Rappels

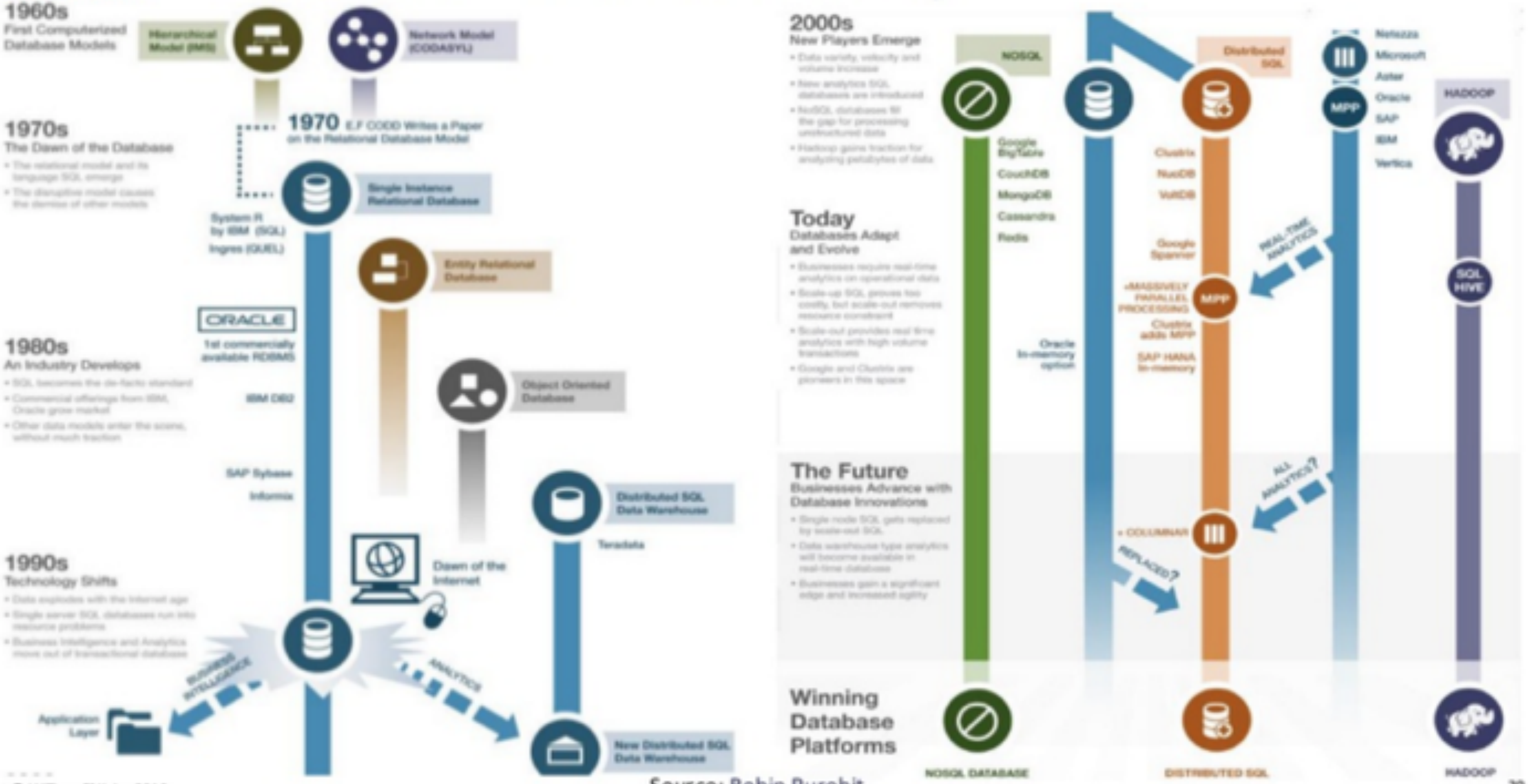
3. Préliminaires : stockage et optimisation

2. Entrepôts de Données

3. Hadoop/Map-Reduce

4. **Séminaires** : Défis, Cas d'Utilisation, Solutions Big-Data Analytics & IA/ML

Database Evolution History



1960s
First Computerized Database Models

1970s
The Dawn of the Database
- The relational model and its language SQL emerge
- The disruptive model causes the demise of other models

1980s
An Industry Develops
- SQL becomes the de-facto standard
- Commercial offerings from IBM, Oracle grow market
- Other data models enter the scene, without much traction

1990s
Technology Shifts
- Data explodes with the Internet age
- Single server SQL databases run into resource problems
- Business Intelligence and Analytics move out of transactional database

2000s
New Players Emerge
- Data variety, velocity and volume increase
- New analytics SQL databases are introduced
- NoSQL databases fill the gap for processing unstructured data
- Hadoop gains traction for analyzing petabytes of data

Today
Databases Adapt and Evolve
- Businesses require real-time analytics on operational data
- Scale-up SQL proves too costly, but scale-out removes resource constraint
- Scale-out provides real time analytics with high volume transactions
- Google and Clustrix are pioneers in this space

The Future
Businesses Advance with Database Innovations
- Single node SQL gets replaced by scale-out SQL
- Data warehouse type analytics will become available in real-time database
- Businesses gain a significant edge and increased agility

Winning Database Platforms

Source: Robin Purohit

MCC 40%TD + 60%Examen

- TD obligatoires
  - Rappels
  - Entrepôts D. 1

  TD facultatifs
  - Optimisation
  - Entrepôts D. 2
  - Map/Reduce

- Mini-projet
  – Rendu intermédiaire (document)
  – Rendu final (document, code, transparents)

- **Présence obligatoire** séminaires C. Menichetti

Divers

- TD
  - salles 36.207 (F. Ulliana) + 5.130 (Anne-Muriel Arigon/Julie Cailler)
  - éléments de correction dispensés en cours
  - binômes ou monômes (sans exceptions)

- Consultez la page Moodle dédiée à l'UE

- Access comptes Oracle *possible à distance* (SSH)
  - instructions sur Moodle

- Signaler les dysfonctionnements des serveurs Oracle
  - ENT --> Assistance --> Centre de Services --> Déclarer un Incident

- Nous écrire pour tout type de problème ou question
  - chifolleau@lirmm.fr,federico.ulliana@inria.fr

Objectifs Pour Aujourd’hui

- Rappels
  - UML
  - Modèle Relationnel
  - SQL
- Stockage
  - “the database game” (clickbait)
- TD
  - Modélisation UML, SQL (facile)
  - Rendu jeudi 21/09 avant 23h59

Rappels

Readings

These slides should be considered simply as "pointers" to the references below.

[BD-G] Bases de données,
Georges Gardarin, 5ème edition 2005
http://georges.gardarin.free.fr/Livre_BD_Contenu/XX-TotalBD.pdf

[ORA] Oracle® Database SQL Language
Reference 11g Release 1 (11.1) - 2013
http://docs.oracle.com/cd/B28359_01/server.111/b28286.pdf

[UML] Prolegomenes_uml.pdf

[UML2] UML2 : de l'apprentissage à la pratique

Relational Databases & UML

NB : *Assumed to be well known from L2/L3, we just recall basic topics.*

1. UML

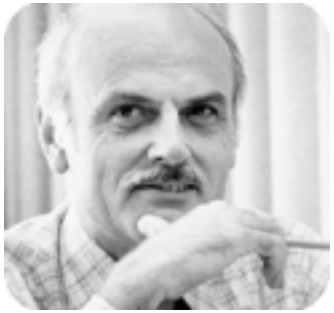
2. Relational Model

3. SQL

Levels of Modelling

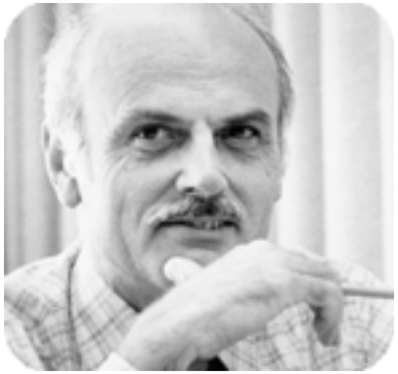
- Conceptual Model (UML, EA, Merise)
  - defines what the system contains
- Logical Model (Relational Model, Object Model, Graph)
  - Defines data structures and rules of the system
- Physical Model (SQL, OQL, XML)
  - defines how the system has to be implemented

Peter Chang
EA – 1976



Ted Codd
RM – 1970

Don Chamberlin
SQL – 1974



BASIC RELATIONAL MODEL THEORY

The Relational Model

[BD-G] chapter VI section 2.1

Everything is a relation

- **Person**(`Bob,42,Paris`)

  (can model entities)

- **LiveTogether**(`Alice,Bob,Lyon,2010`)

  (can model associations)

Relational Schema

[BD-G] chapter VI section 2.2

- A set of relations built on a set of attributes, with well defined domains.

Person(Name,Age,City)

Name : String   Age: Integer   City : {Lyon,Paris}

The Model VS. The Content

- The idea of representing data using relations is abstract, universal.

- But, as this data is originated from real world interactions (eg., trading, social), all forms of weak and strong correlations are found in it.

Functional Dependency

[BD-G] chapter VI section 3

A set of attributes ***A*** *determining a set of attributes* ***B***

(name, surname) ------------> birthday

determine

city ------------> (population, state)

determine

Key(s)

[BD-G] chapter VI section 3.1

minimal set of attributes determining a whole tuple

LiveTogether(Person1, Person2, City, Date)

Person1 , Person2 , Date --------> City

key determines

(ex) **LiveTogether**(`Alice,Bob,Lyon,2010`)

Strongly recommended in systems (efficiency, integrity)

Data dependencies were undesirable

- Except for **keys** and referential **integrity constraints**
  - beside these cases, they just bring redundancy
- Database **normalization** eliminated dependencies

Normal Forms

Normal-Forms are guidelines for modeling.

Normal forms came after the relational model.

Their definition is motivated by design mistakes.

So, let’s find the right place for the attributes !

Normal Forms : 2NF

[BD-G] chapter VI section 6.2

- **2NF** : non-key attributes fully-dependent from the key

FournisseurPiece(Name, Article, Address, Price) **NO**

(Article --/--> Address)

↓ (decomposition)

FournisseurPiece(Name, Article, Price)

Fournisseur(Name, Address)

Normal Forms : 3NF

[BD-G] chapter VI section 6.3

- 3NF : no dependencies between non-key attributes

Person(ID , Name, City, *CityPopulation*) **NO**

ID -----> City -----> CityPopulation

↓ (decomposition)

Person(ID , Name, City) **Place**(City, *CityPopulation)*

Normal Forms

[BD-G] chapter VI section 6.3

- 3NF respected by **most***"transactional-database"* you will find in **any** real world company
  - it allows to fix common data redundancy problems
  - also, every schema can be normalized in 3NF

- but sometimes only 2NF …
  - not (always) by mistake of database administrator
  - conscious choice : pay the price of redundancy (and update) to gain in performances (more later)

- 3NF = 2NF + no indirect dependencies (simplification)
- 2NF = 1NF + no partially dependent attributes (simplification)
- 1NF = ?

- Are there transactional databases which do not respect the 1NF ? (Non-first-normal-form N1NF)

Normal Forms

[BD-G] chapter VI section 6.3

- 3NF respected by **most***"transactional-database"* you will find in **any** real world company
  - it allows to fix common data redundancy problems
  - also, every schema can be normalized in 3NF

- but sometimes only 2NF …
  - not (always) by mistake of database administrator
  - conscious choice : pay the price of redundancy (and update) to gain in performances (more later)

- 3NF = 2NF + no indirect dependencies (simplification)
- 2NF = 1NF + no partially dependent attributes (simplification)
- 1NF = only atomic values in a tuple

- Are there transactional databases which do not respect the 1NF ? (Non-first-normal-form N1NF)
  - Yes ! Object-oriented, XML, many in the NoSQL family (using JSON as data model)

Normal Forms : Remarks

- 2NF & 3NF respected in practically any information system using a relational database

- Stronger normal forms (BCNF, 4NF, 5NF) are less employed (avoid rarer mistakes; not always achievable)

- Exceptions to normalizations are tolerated to save joins (at the price of redundancy)
  - **we will see this for datawarehouses**

SQL : SURVIVAL KIT

SQL

- Structured Query Language

- Declarative (logical) Language : tell what you want from relations, not what to do with them.
  - This is the main difference with C and Java, *not only* the fact that we deal with data.

- In SQL terminology, a relation is called “table”.

SQL

- DDL (Data Definition Language)
  - CREATE/ALTER structures (table, view, index)
- DML (Data Manipulation Language)
  - UPDATE/INSERT/DELETE content
- DQL (Data Query Language)
  - SELECT data

SQL

- The purpose of the following slides is just to recall you the basic syntax.

- Check the documentation for more details or advanced features !

Create Table

[BD-G] chapter VII section 2.1 and [ORA] section 16-6

```
CREATE TABLE Employee (

 id   NUMBER,
 name VARCHAR2(50),
 birthday DATE

)
```

Datatypes

[ORA] section 2-6

Why do we need datatypes ?

- To associate fixed set of properties to attributes.
- This improves the database:
  - coherence : type-checking operations
    - cannot sum two strings
  - efficiency : a datatype has its own best storage
    - BLOB vs integers

ORACLE Built-in Datatypes

- Character
- Numeric
- Date/Time
- Large Object

Complete list : see [ORA] table 2-1

Value Constraints

[ORA] section 8-4 and [BD-G] chapter VII section 2.1

- Why do we need constraints ?

- To restrict the values in a database and ensure the data integrity
  - ex : No employee without an ID

Not Null

[ORA] section 8-8

- Prohibits a database value from being null

```
CREATE TABLE Employee (

id         NUMBER NOT NULL,
name       VARCHAR2(50),
birthday   DATE

)
```

Unique

[ORA] section 8-9

- Prohibits multiple rows from having the same value (but allows them to be null)

```
CREATE TABLE Employee (

id        NUMBER UNIQUE,
name      VARCHAR2(50),
birthday  DATE

)
```

Primary Key

[ORA] section 8-9

- Combines a NOT NULL constraint and a UNIQUE constraint in a single declaration

```
CREATE TABLE Employee (

id          NUMBER PRIMARY KEY,
name        VARCHAR2(50),
birthday    DATE

)
```

Primary Key : Multiple Attributes

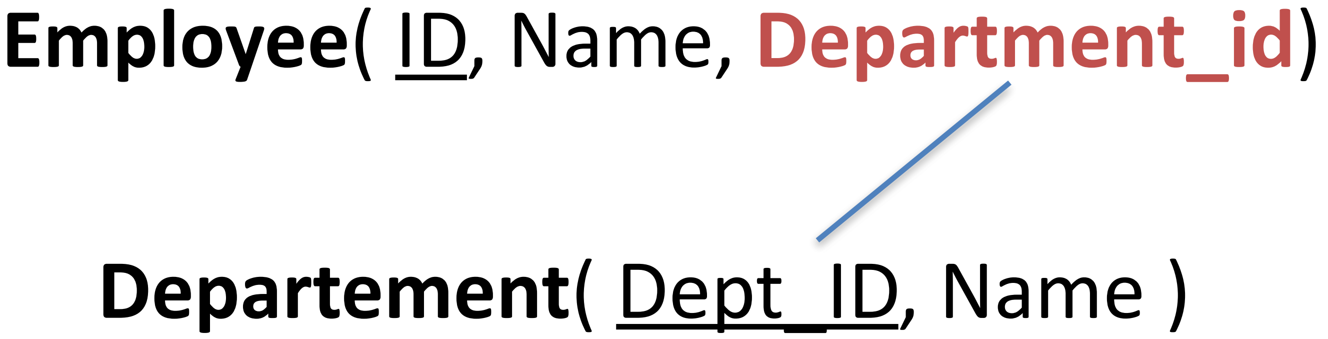
[ORA] section 8-20

- Combines a NOT NULL constraint and a unique constraint in a single declaration

```
CREATE TABLE Employee (
id        NUMBER,
name      VARCHAR2(50),
birthday  DATE,

PRIMARY KEY (name,birthday)
)
```

Foreign key

one (or more) ***attributes*** *which correspond to the* ***key*** *of another relation*



Employee(ID, Name, **Department_id**)

Departement(Dept_ID, Name)

Foreign Key

[ORA] section 8-10 and 8-21

Requires values in one table to match values in another table.

```
CREATE TABLE Employee (

      id            NUMBER,
      department    NUMBER

      FOREIGN KEY department
                     REFERENCES Dept(dept_id)
      )
```

Check

[ORA] section 8-10 and 8-22

Requires a value to satisfy with a specified condition

```
CREATE TABLE Employee (
      id             NUMBER,
      department     NUMBER,
      office         VARCHAR2(10)

      CHECK (
         office IN
            ('DALLAS','BOSTON’, 'PARIS','TOKYO')
      )
)
```

Oracle does not verify mutually exclusive conditions (eg. AGE>1 AND AGE<0)

ALTER TABLE

[BD-G] chapter VII section 6.2.4 and [ORA] section 12-2

- Add a new column
  - `ALTER TABLE Employee ADD (office VARCHAR2(20));`
- Modify an existing column
  - `ALTER TABLE Employee MODIFY (office NUMBER);`
- Define a default value for the new column
  - `ALTER TABLE Employee MODIFY office DEFAULT ‘Corridor’;`
- Drop a column
  - `ALTER TABLE Employee DROP (office);`

DELETE TRUNCATE DROP

[ORA] section 17-25 and 19-62 and 18-5

	removes		
	rows	table	rollback
DELETE	✓	✗	✓
TRUNCATE	✓	✗	✗
DROP	✓	✓	✗

- TRUNCATE = DROP + CREATE TABLE

INSERT

[BD-G] chapter VII section 4.1 [ORA] section 18-66 and 18-54

```
INSERT
INTO Employee
VALUES
       (‘Bob’,
        TO_DATE(
        ’03-OCT-1972',’DD-MON-YYYY')
        )
```

TO_DATE converts a character/numeric to a date

[ORA] 2-49/50

SELECT FROM

[BD-G] chapter VII section 3.1 [ORA] section 18-66 and 18-54 and 2-49

```
SELECT
   TO_CHAR(birthday,'MM-DD-YYYY')

FROM
   Employee
```

TO_CHAR converts a numeric to a character

[ORA] 2-287/288 and 5-292

SELECT FROM WHERE

[BD-G] chapter VII section 3.2 [ORA] section 2-50

```
SELECT
    name

FROM
    Employee

WHERE
    birthday >
        TO_DATE(’01-10-1970','DD-MM-YYYY’)
```

JOINS

[BD-G] chapter VII section 3.3

```
SELECT
   Employee.name, Department.name
FROM
   Employee, Department
WHERE
   Employee.dept = Department.id
```

JOINS

Employee

name	**dept**
Alice	dep1
Bob	dep2
Eddy	dep1

Department

id	**name**
dep1	Sales
dep2	Production

Emp_Join_Dep

Employee.name	**Department.name**
Alice	Sales
Bob	Production
Eddy	Sales

Group By

[BD-G] chapter VII section 3.7

```
SELECT
   dept, count(*) as N
FROM
   Employee
GROUP By
   dept
```

Employee

name	dept
Alice	dep1
Bob	dep2
Eddy	dep1

Agg_Emp

dept	N
dep1	2
dep2	1

Group by queries are at the essence of ANALYTICS.

Summing Up

- Relations
  - Functional dependencies, Normal-forms

- SQL
  - CREATE, INSERT, DELETE, SELECT, GROUP-BY

CONCEPTUAL MODELING WITH ENTITY-RELATIONSHIP DIAGRAMS

To minimize the number of graphical notations introduced across the various programs of this training, we take an alternative model.

CONCEPTUAL MODELING WITH ~~ENTITY-RELATIONSHIP DIAGRAMS~~

CONCEPTUAL MODELING WITH UML

UML (Unified Modeling Language)

- Universal **graphical** modeling language designed to model objects, associations, time events, system states
  - Main goal is to ease prototyping

- Good news : UML is a rich model and we can also use it to model data !

UML (Unified Modeling Language)

graphical

means :

- flexible
- error-prone
  - improper use of notation, wrong modelling
- no debugger !

UML : Plan

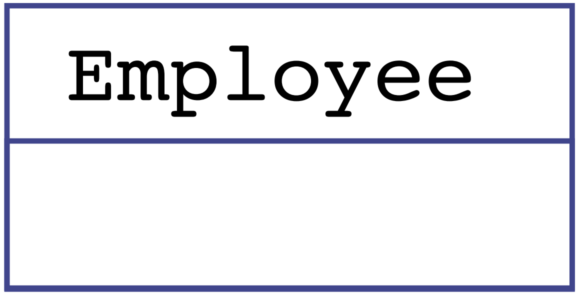
- UML Class Diagram
  - Basic constructs that can be used to model Relational Databases in UML
  - Real Object-oriented features

Class

[UML] section 3 [BD-G] chapter XVII section 2

- Set of elements sharing common properties
  - ex. peoples, animals, cars

- UML : draw a labelled box

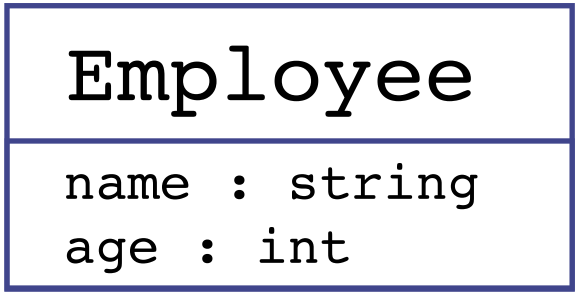


Class : Attributes

[UML] section 3 [BD-G] chapter XVII section 2

- Attributes denote the properties of class objects
  - They are usually typed

- Write the attributes below the class name

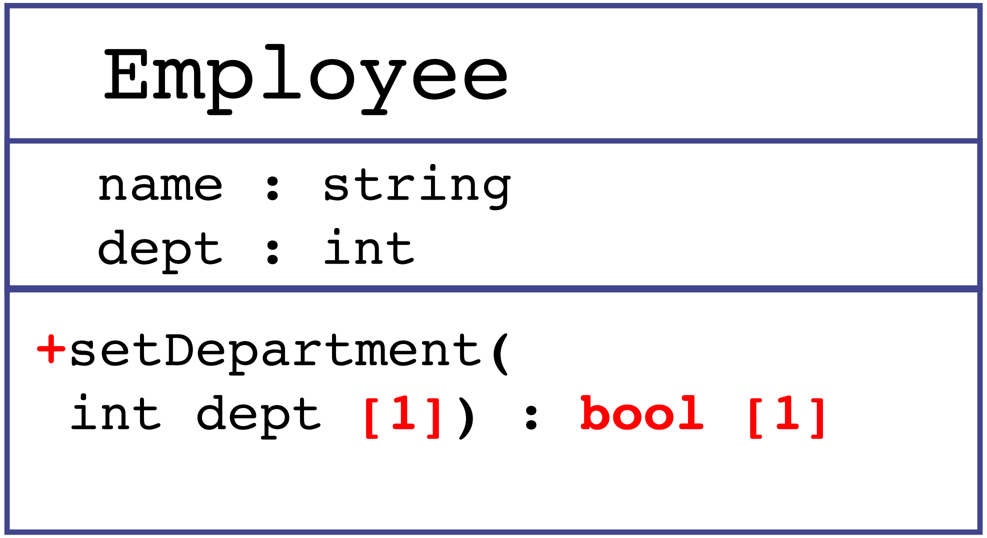


Employee

name : string
age : int

Operations can specify

- Visibility (**+** public) (**-** private) (**#** protected)
- Return-type (*optional*; can be undefined)
- Multiplicity of parameter/return-type (*optional*)

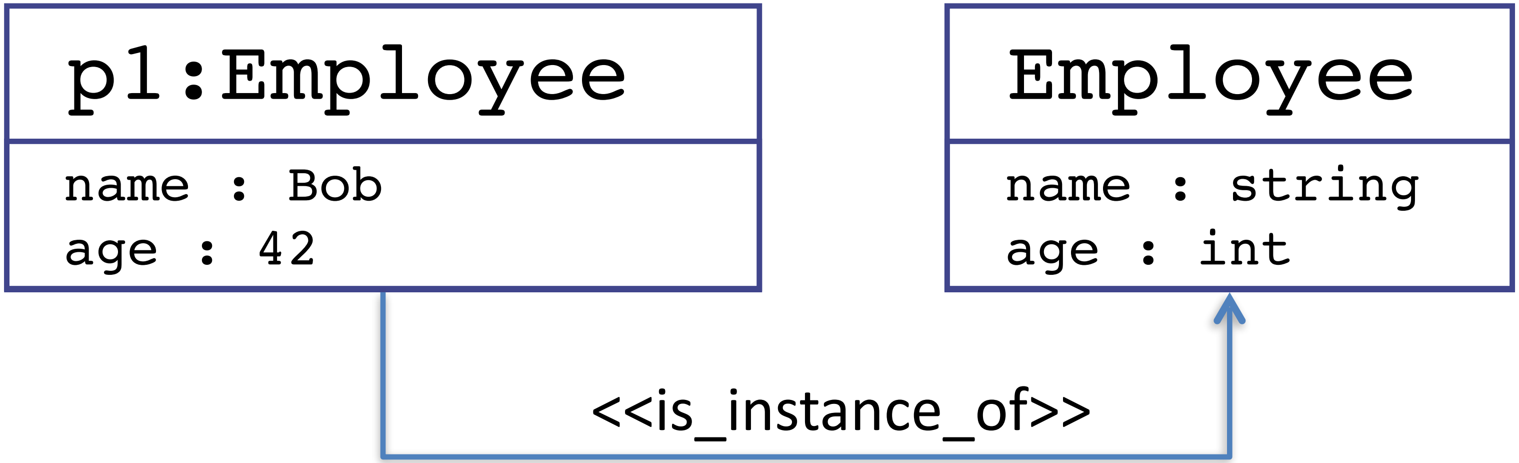


```
Employee

name : string
dept : int

+setDepartment(
 int dept [1]) : bool [1]
```

Instances

- The elements of a class
  - ex. the employee Bob

- Related to their class by a directed edge

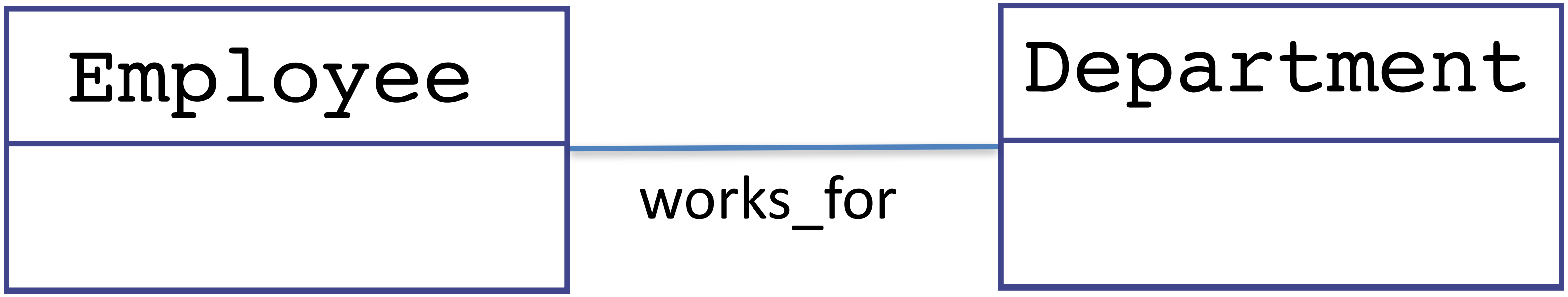


Binary Associations

[UML] section 4 [BD-G] chapter XVII section 2

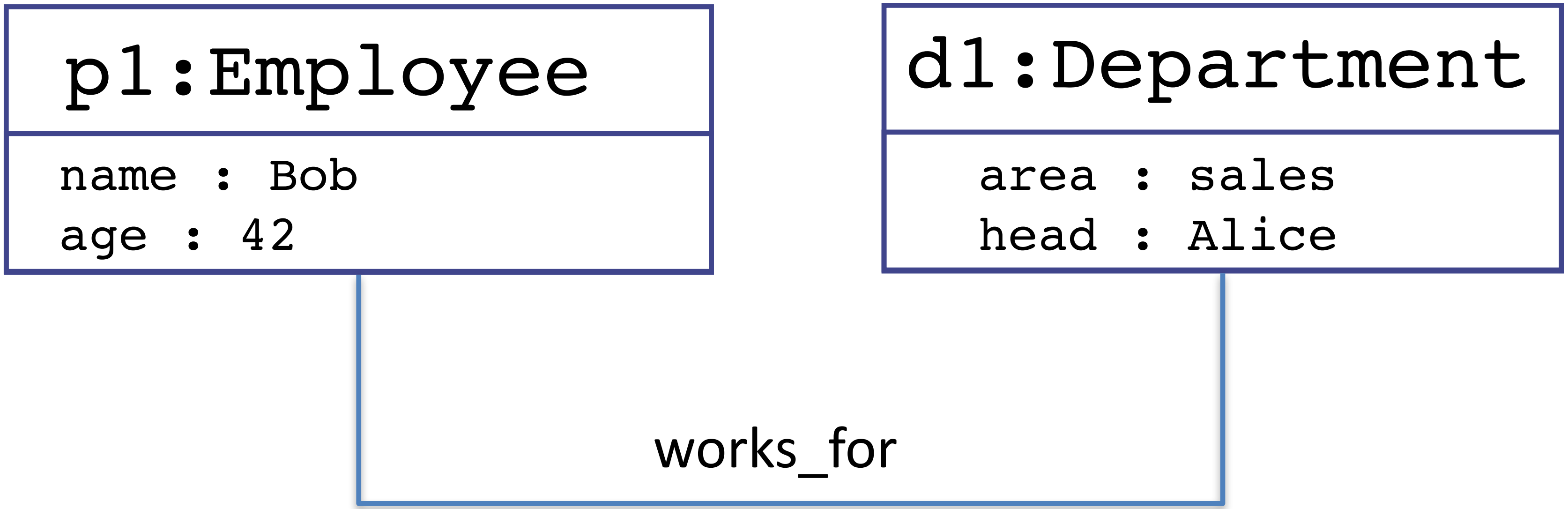
- General relationships between elements of two classes
  - ex. an employee works for a department
  - a concrete instance of association is called link

- Binary association : *undirected* edge between classes



Associations : Links

- Relationships between instances of classes



p1:Employee

name : Bob
age : 42

d1:Department

area : sales
head : Alice

works_for

Associations : a tricky notation

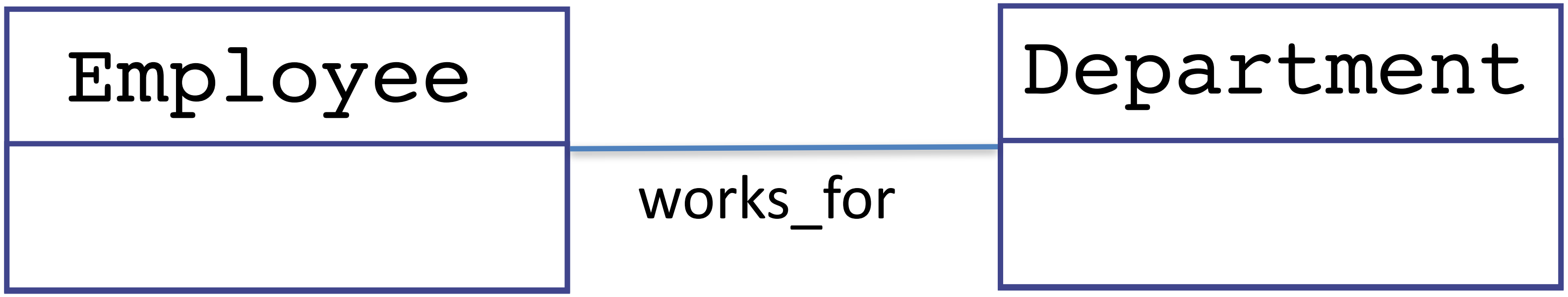
- Any association is specified by three things

1. Its name

1. The cardinality constraints of the class elements

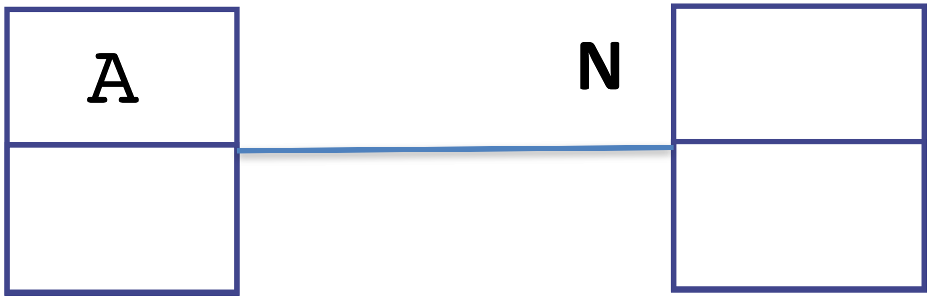
2. The role of the classes in the association (optional)

Associations : Name (1)



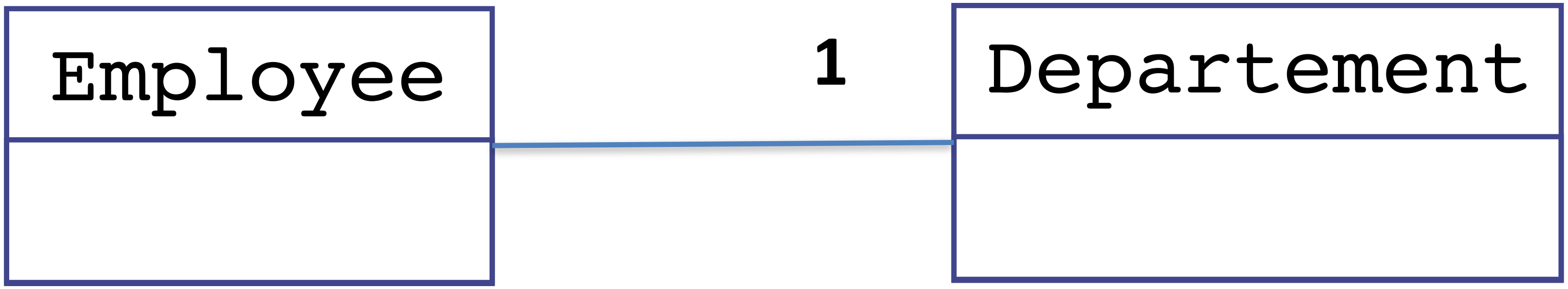
- The name is a label placed in the middle of the edge

Associations : Cardinality (2)



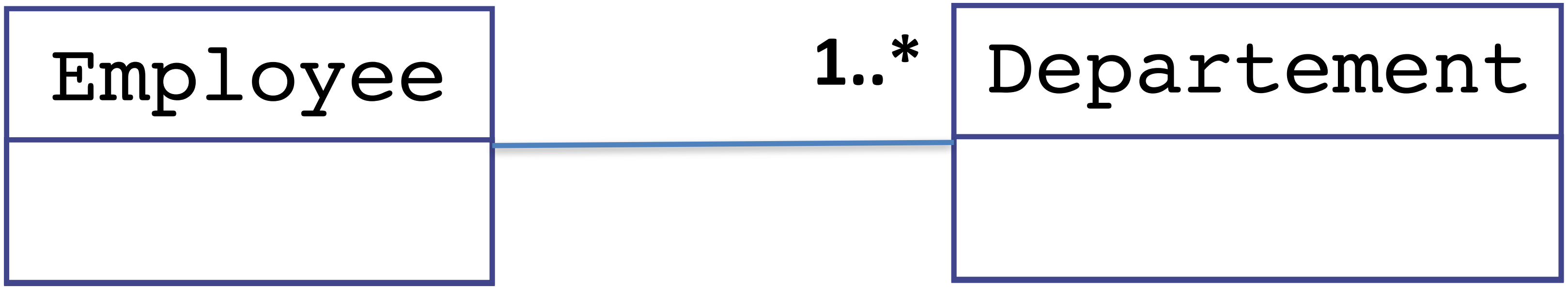
- This means that one instance of `A` participates in the association with **N** elements of the other class

Associations : Cardinality (2)



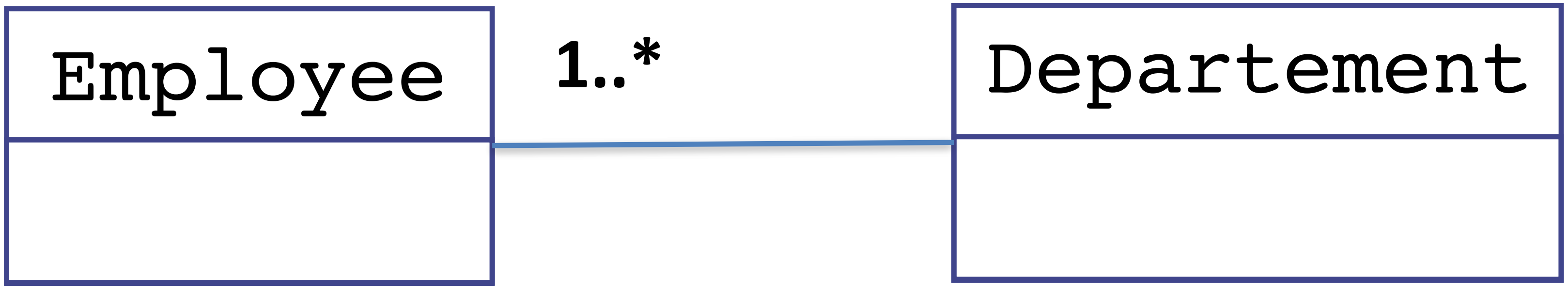
This means that an employee works for exactly **1** department

Associations : Cardinality (2)



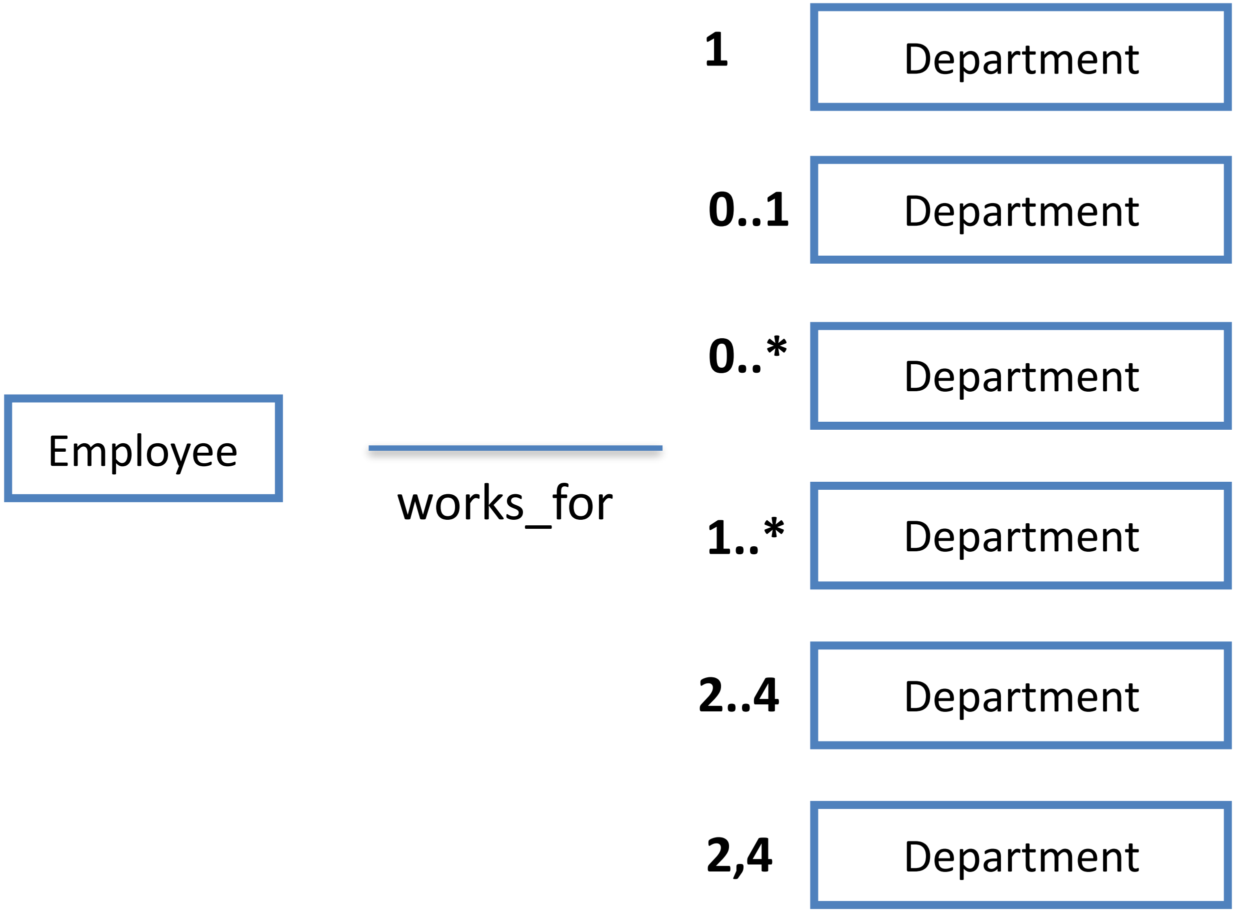
This means that an employee can work for more than **1** department (but at least one)

Associations : Cardinality (2)



This means that a department has at least one employee, with no upper-limit.

Cardinality Specification

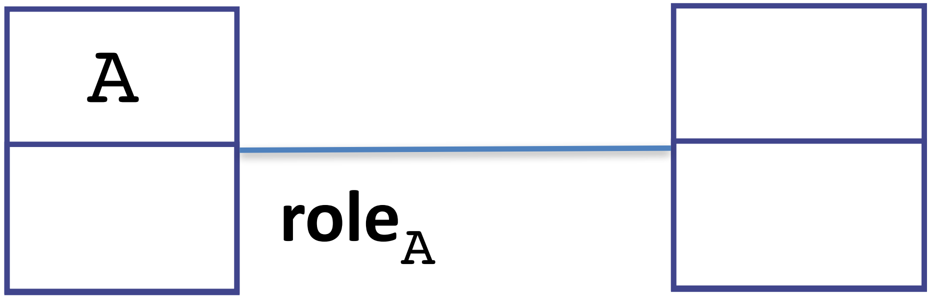


Employee

works_for

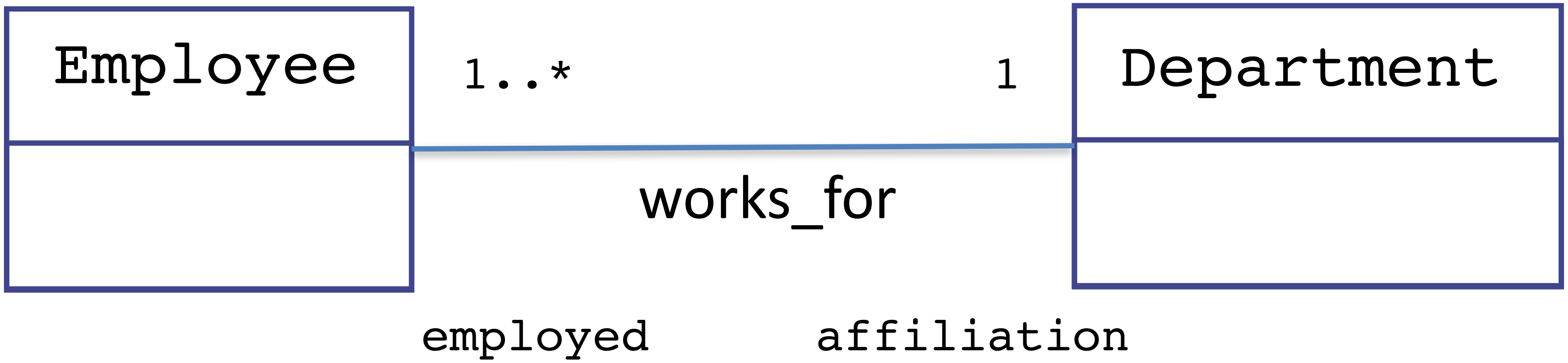
1	Department	exactly one
0..1	Department	zero or one
0..*	Department	zero or N (default)
1..*	Department	at least one
2..4	Department	from two to four
2,4	Department	two or four

Associations : Roles (2)

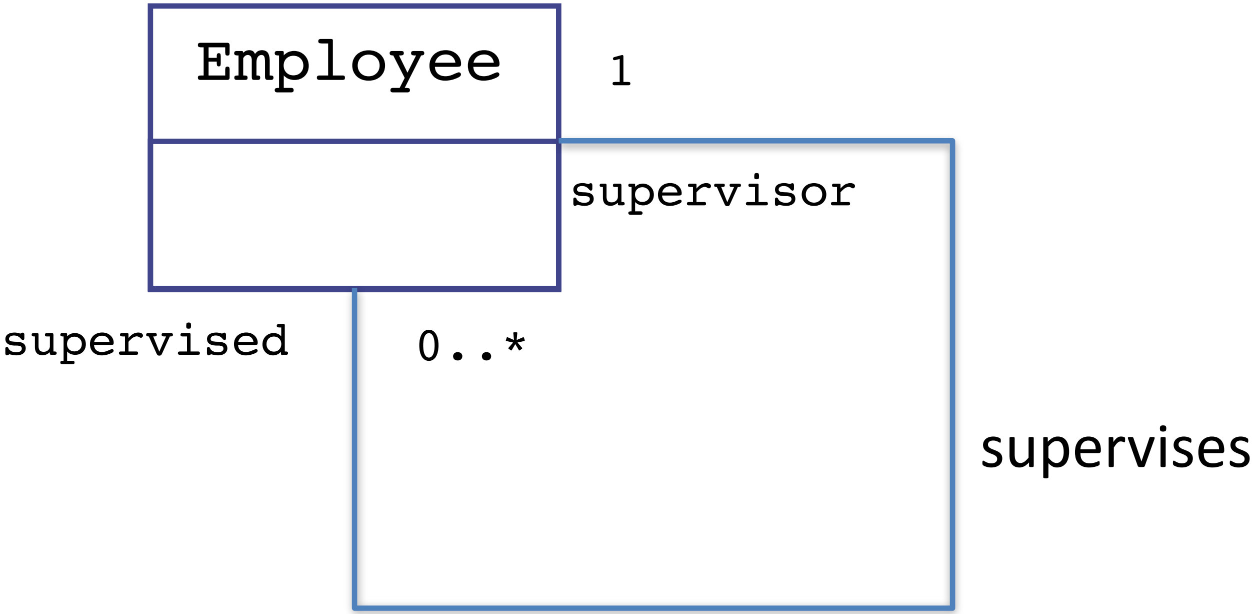


- This means that in the association an instance of `A` plays **role**$_{\mathtt{A}}$

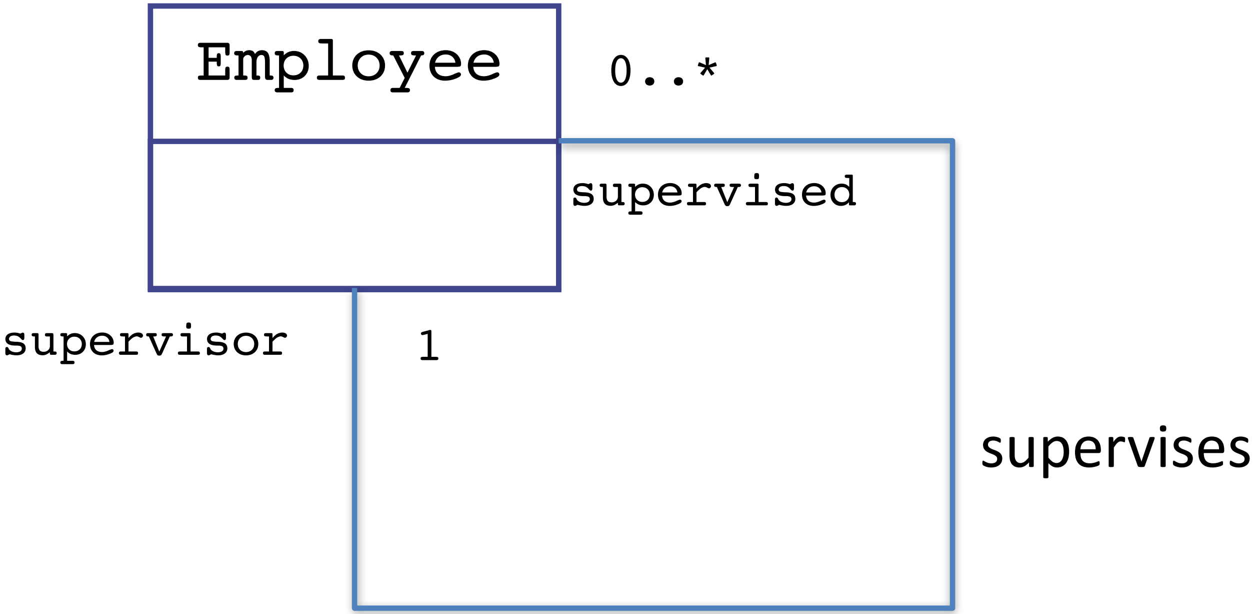
Putting everything together



Reflexive Associations

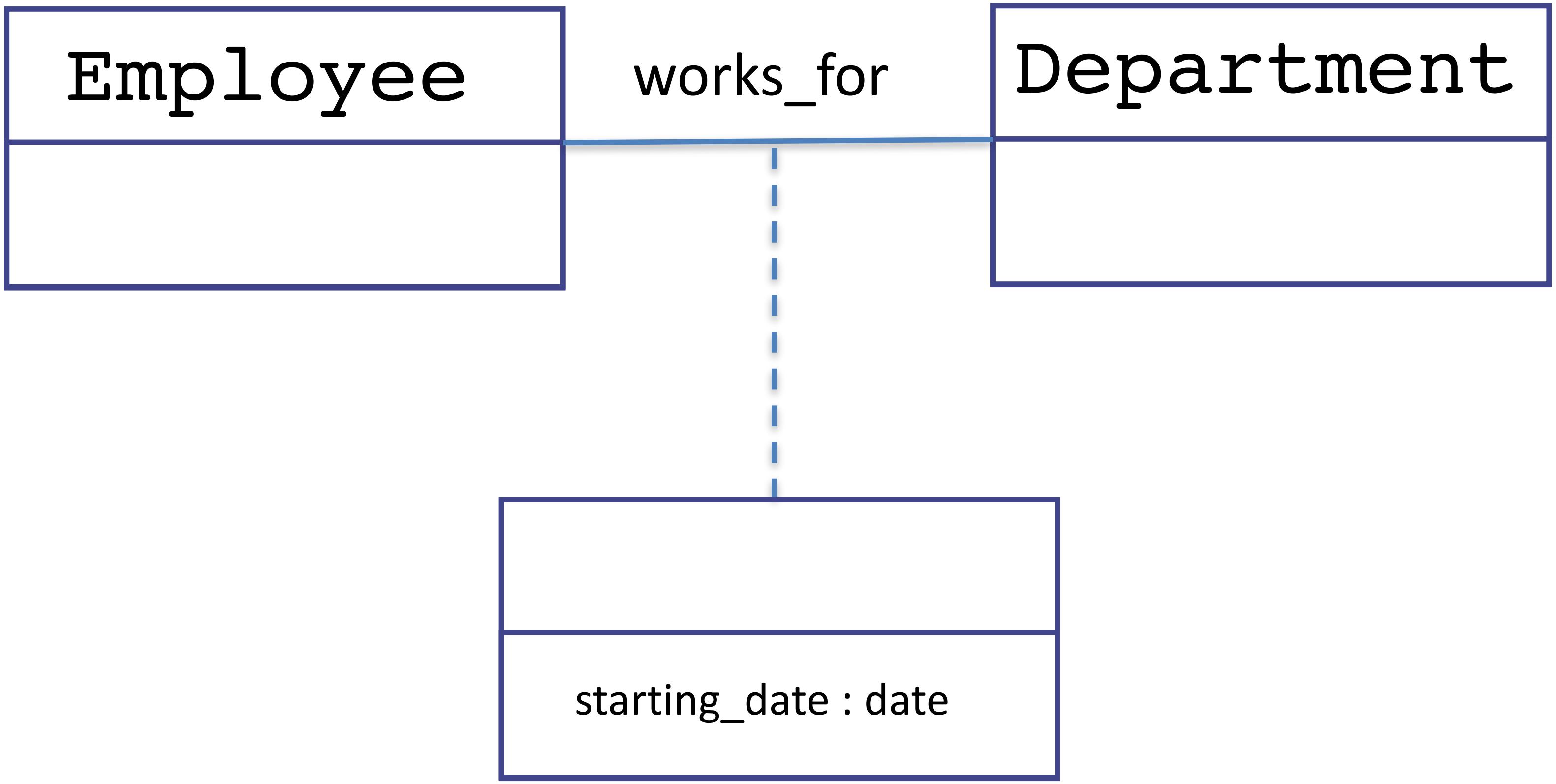


Equivalent Formulation



Notation for Attributes in Associations

[UML] section 4 [BD-G] chapter XVII section 2



Associations, Associations, Associations…

Often, a relationship between two entities combines:

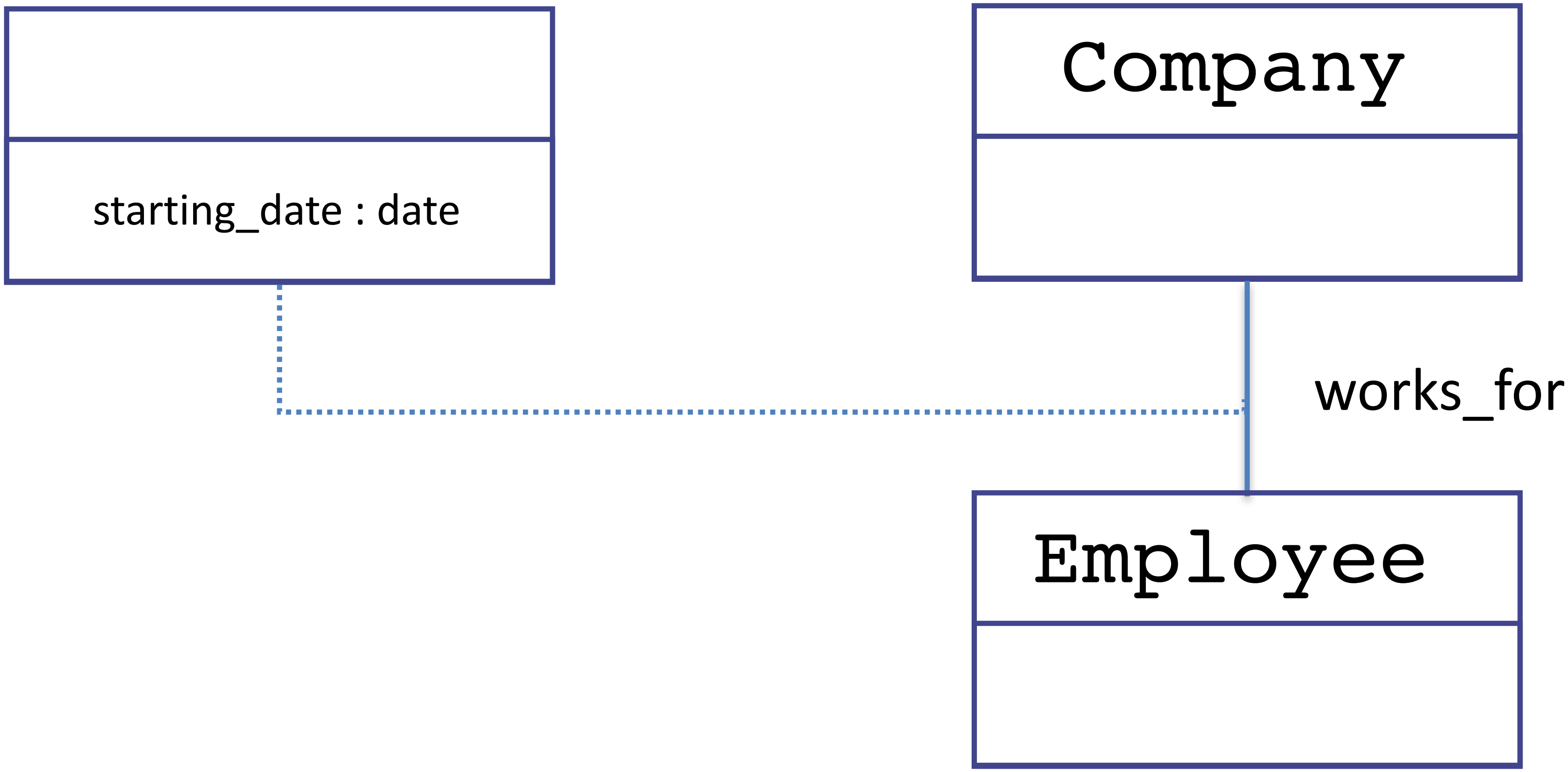
- ***association*** features
  - the fact that two or more things are linked
- ***representation*** features
  - the details about this association

- Ex: a working contract can be seen
  - as a relationship
    (an association between an employee and a company)
  - as an entity (representation of a legal concept)

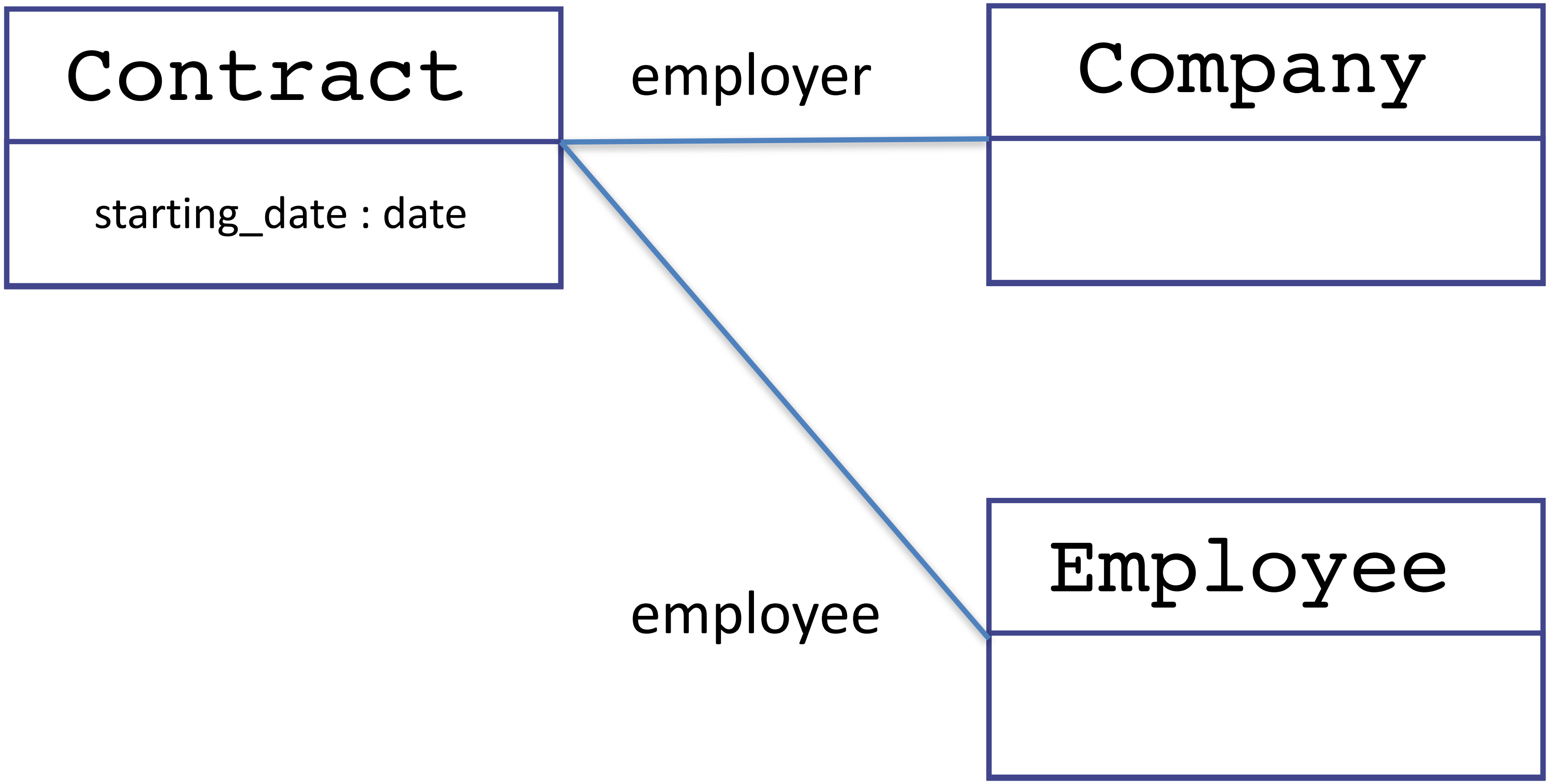
This duality is the source of most design problems !

- Recognize your own modelling choices !

Choice 1



Choice 2



Contract

starting_date : date

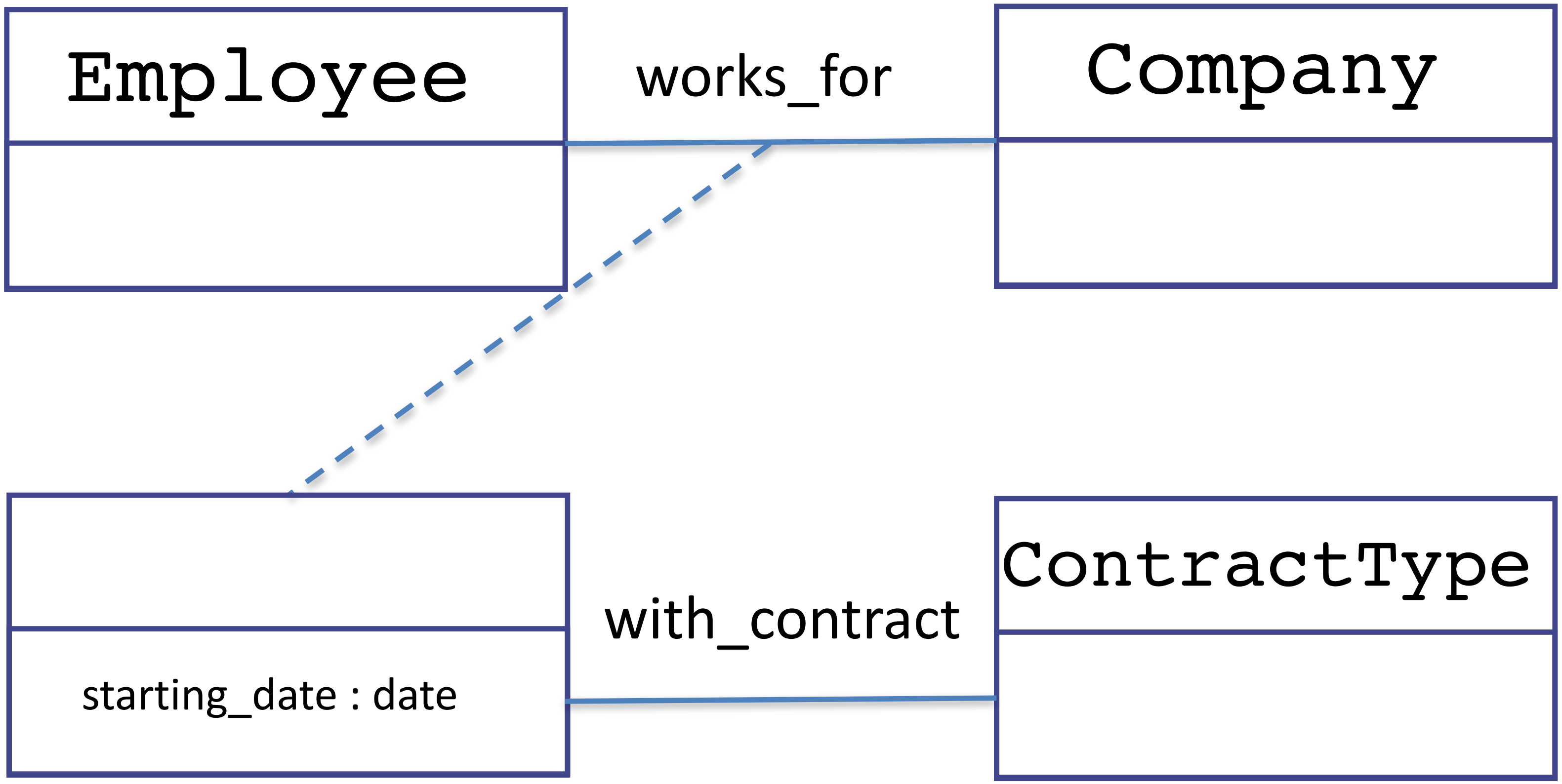
employer

Company

employee

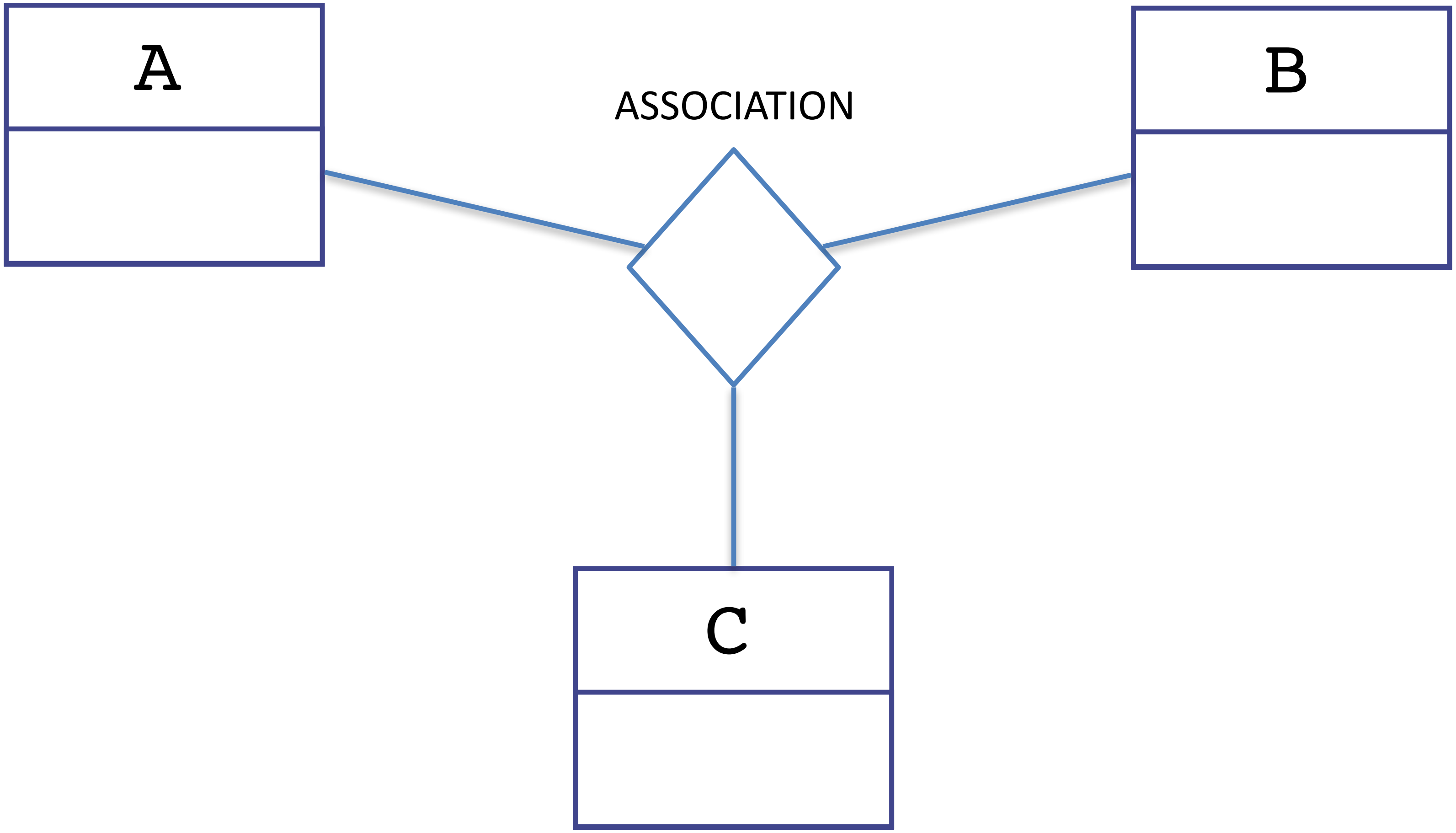
Employee

Associations Participating in Other Associations



Notation for 3-ary Associations

[UML] section 4.1 [UML2] section [3.3.4]



Summing Up

UML for Relational Databases

[UML] section 4 [BD-G] chapter XVII section 2

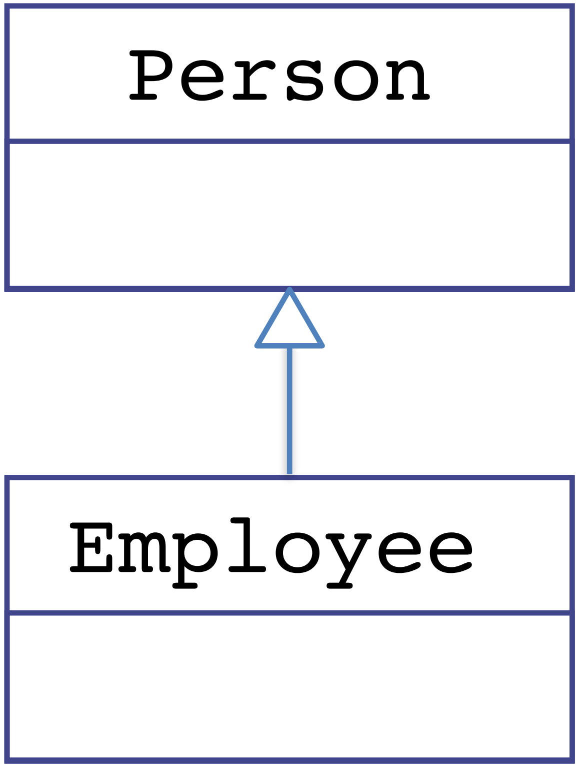
- Basic UML
  - Classes, binary associations, n-ary associations
  - Can model relational databases
- The construct of the language we have seen so far are enough to model relational databases

SubClass

[UML] section 5 [BD-G] chapter XVII section 2.2

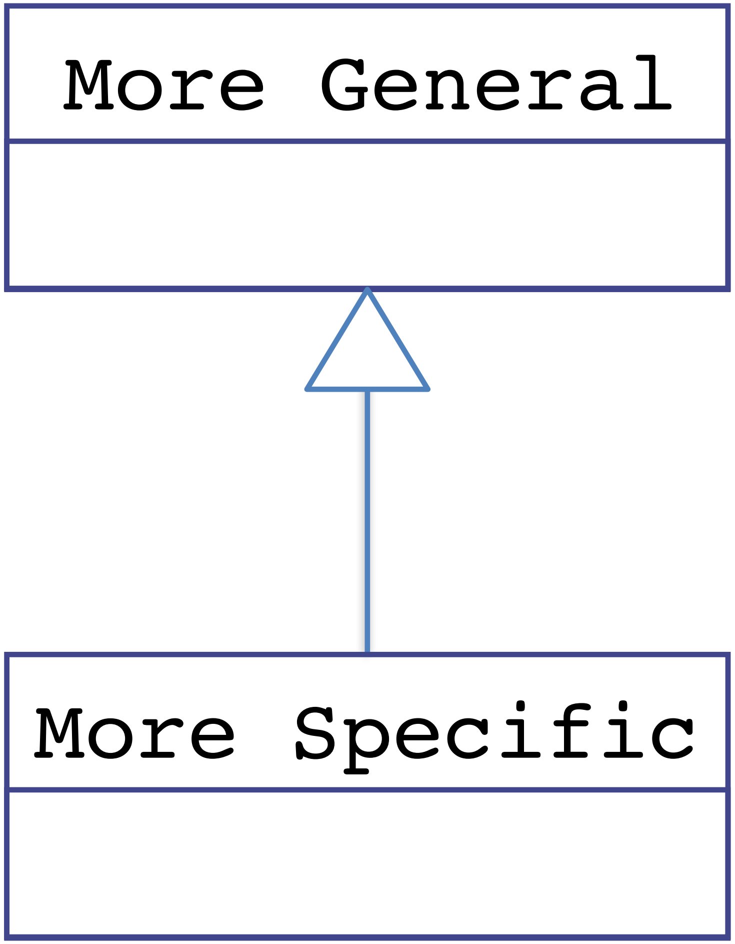
- A subset of the instances of a Class
  - ex. every employee is a person

- A subclass inherits all superclass properties or redefines them.



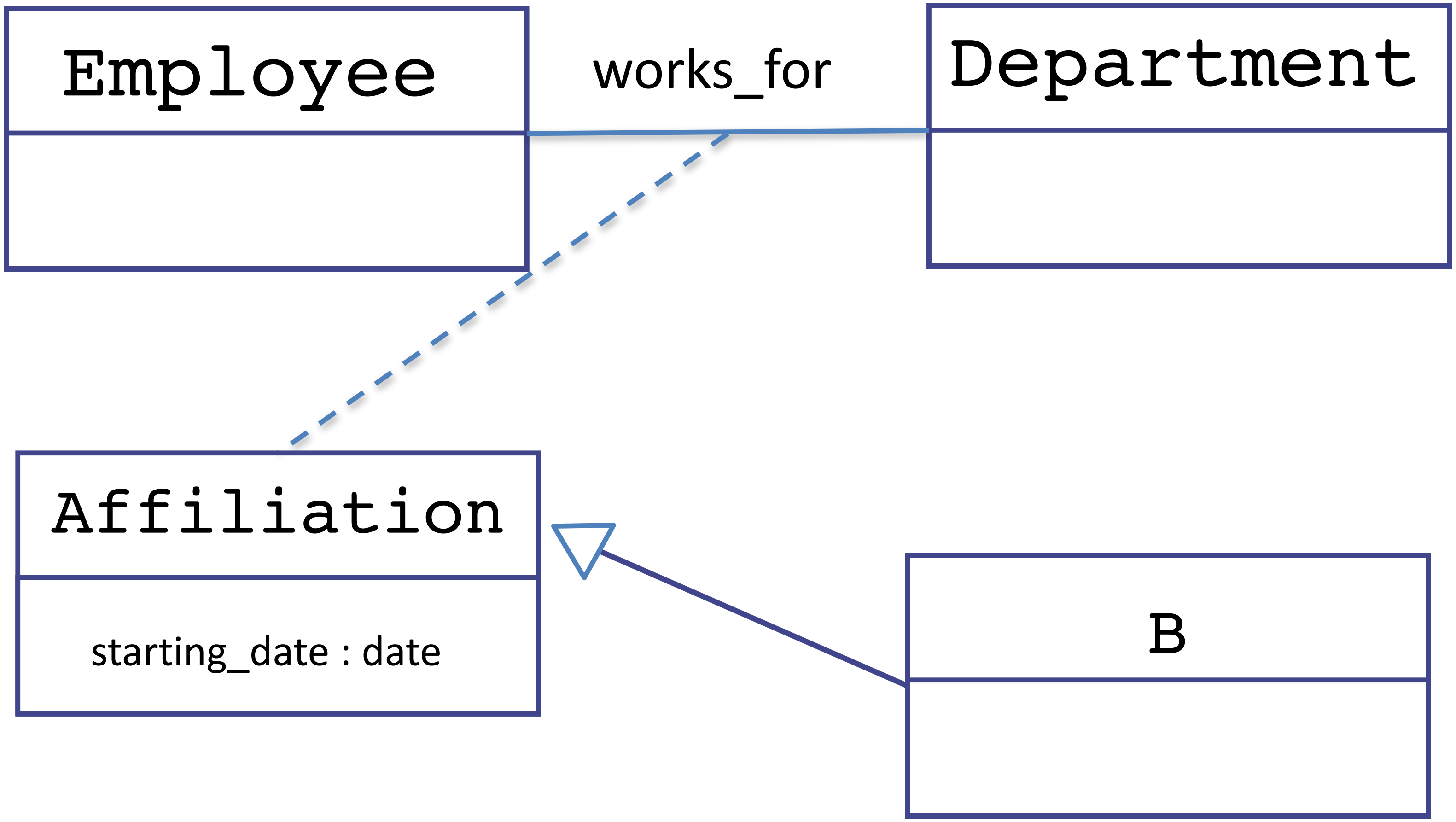
SubClass = Generalization/Specialization

[UML] section 5

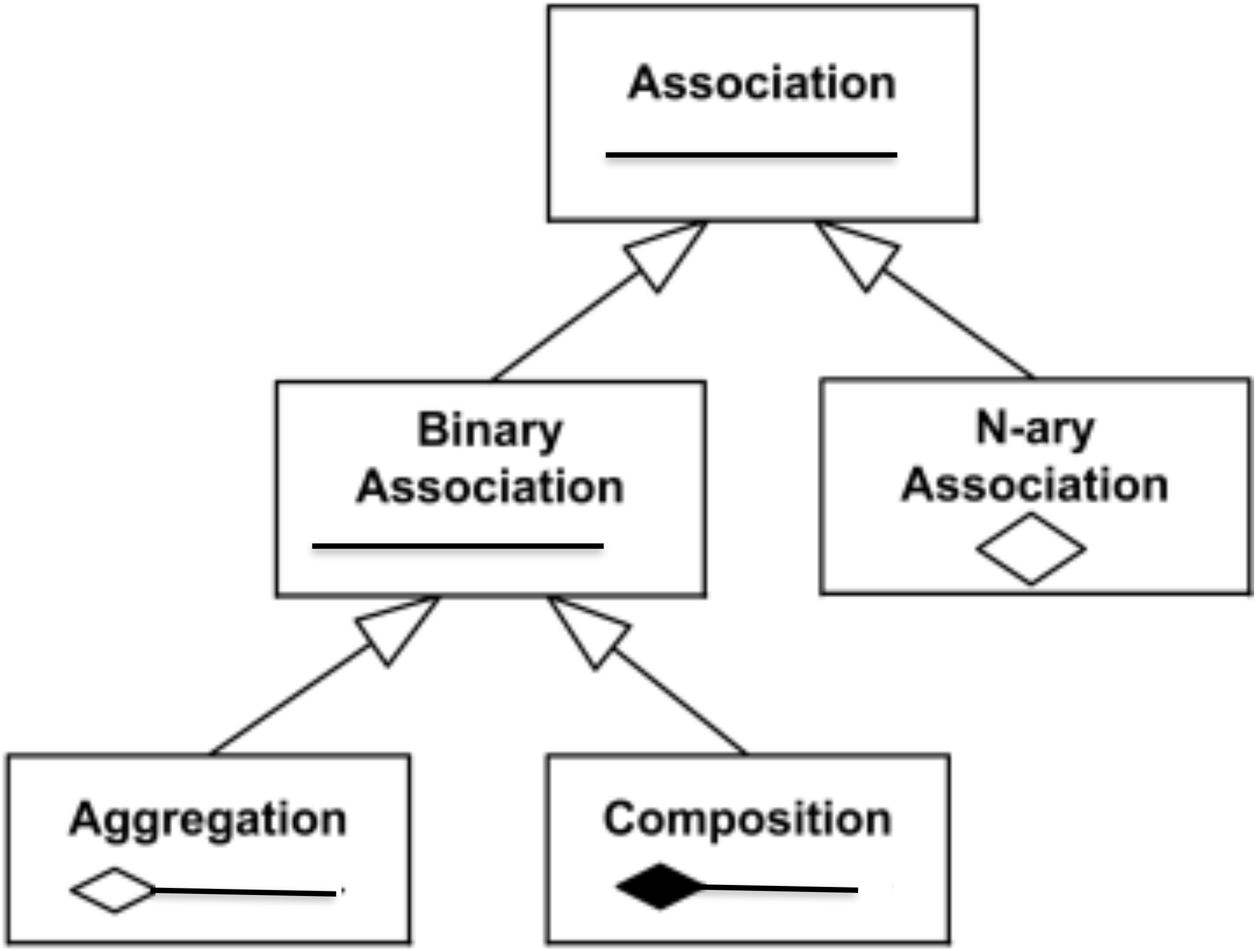


Association modeled by a Class

[UML] section 4.3



- Consequence : such class can have proper attributes, operations and also other associations


Association
Binary Association
N-ary Association
Aggregation
Composition

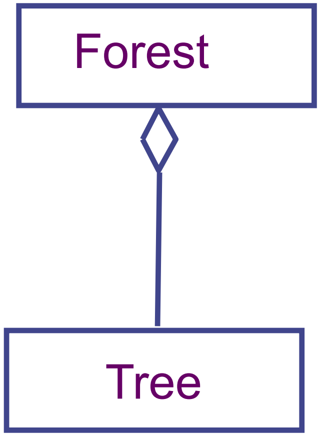
Part-Of (binary) Association

- “A forest is made of trees”

Aggregation

[BD-G] chapter XVII section 2.2

- Consequence 1 : a tree can exist even without a forest



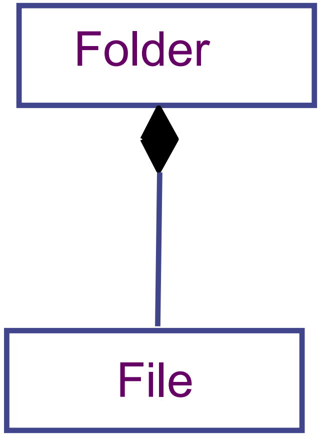
Part-Of (binary) Association

- “A folder is made of files”

Composition

[BD-G] chapter XVII section 2.2

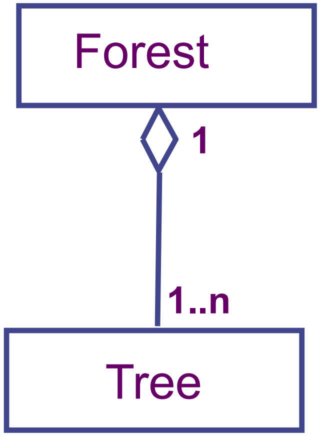
- Consequence 1 : a file cannot exists without a folder



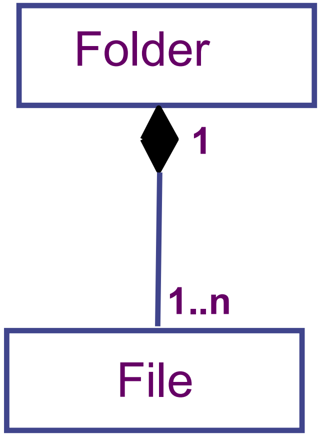
Aggregation Composition

[BD-G] chapter XVII section 2.2

- Consequence 1 : a tree can exist even without a forest

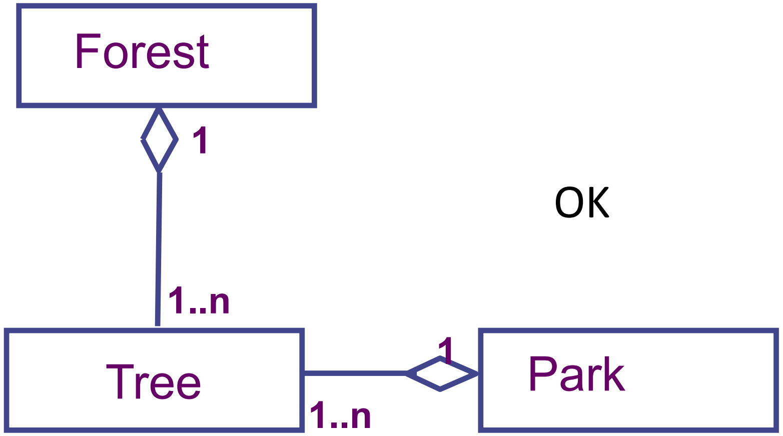


- Consequence 1 : a file cannot exists without a folder



Aggregation

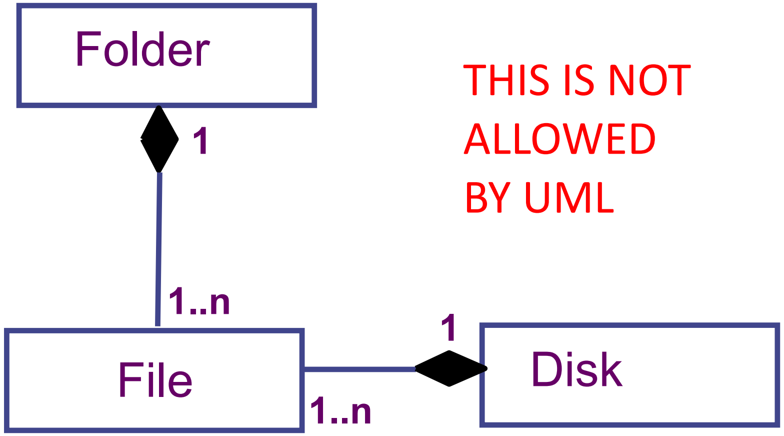
- Consequence 2 : a tree can be part of both a forest and a park



OK

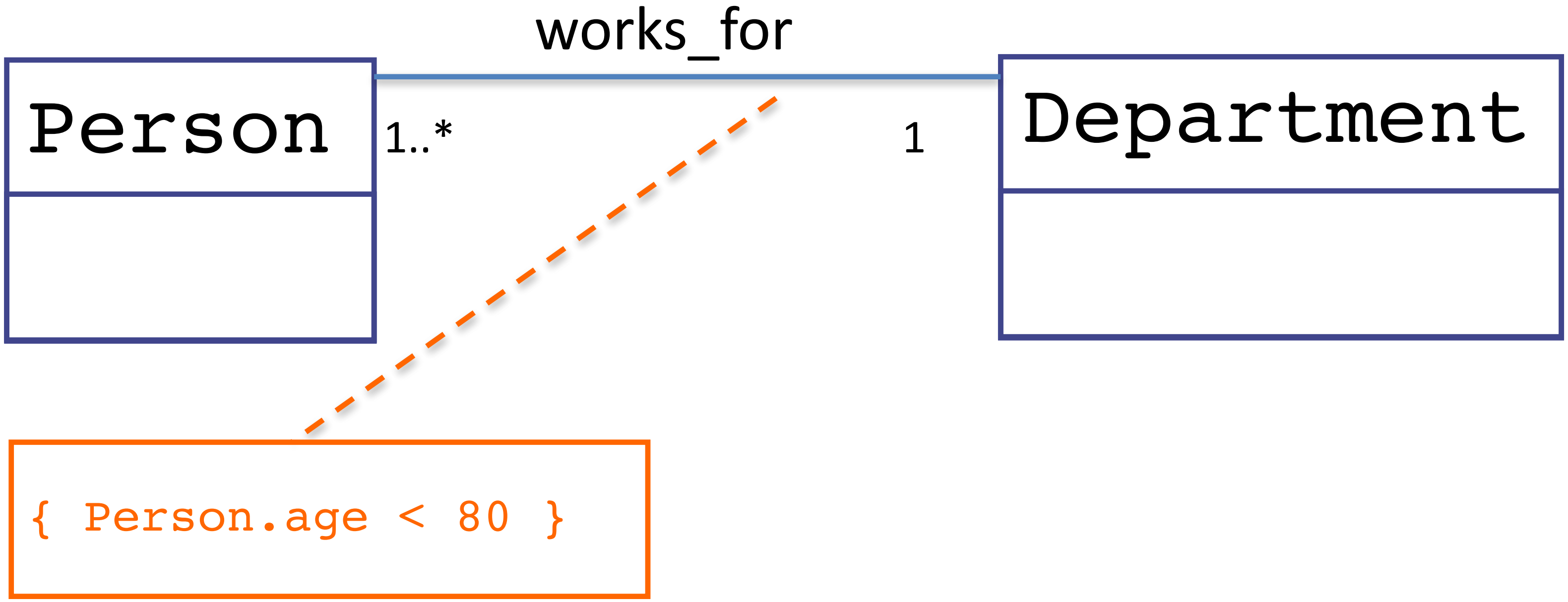
Composition

- Consequence 1 : a file cannot exists without a folder



THIS IS NOT ALLOWED BY UML

Constraints on Associations



- Consequence : only people whose age is less than 80 can participate in the association

Summing Up

- Instances/Links/Operations
- Subclasses
- Aggregation and Composition