Bases de Données Avancées (BDA)

Federico Ulliana
UM, LIRMM & INRIA GraphIK

ulliana@lirmm.fr

BDA – Intervenants

- Anne-Muriel Chifolleau – UM, LIRMM - Resp.
  (chifolleau@lirmm.fr)
- Federico Ulliana – UM, LIRMM, INRIA - Resp.
  (ulliana@lirmm.fr)
- Hatim Chadhi – Chargé de TP
- Christophe Menichetti (externe)

BDA - Programme

Objectifs : comprendre des nouveaux modèles de données et expérimenter avec la technologie associée.

1. Rappel Relationnel
2. Optimisation de requête

2. BD Objets et Objet-Relationnel

3. Modèles multidimensionnel et entrepôts de données
4. Vue “entreprise” sur la “Business Intelligence”
5. Hadoop Map/Reduce

BDA : Objectifs du Cours

- Maitriser la modelisation de données Objet et Multidimensionnelle
- Expérimenter avec la technologie ORACLE
  - +2 lignes dans votre CV.

MCC

- **50%** Partie Pratique (1 session)
  - (1) TD/TP Rappels (groupes de 2 ou 3)
  - (1) Optimisation (groupes de 2 ou 3)
  - (2) Mini-projet Objet-Relationnel (groupes de 4)
  - (3) Mini-projet Entrepôts de données (groupes de 4)

- **50%** Contrôle terminal (2 sessions)

- **Présence** au séminaire = 1pt dans la note.

Divers

- Information sur l'UE dans le Moodle
  - mdp **bda1718**

- TP : éléments de correction dispensés en cours

- *Accueil étudiants* le lundi à 16h au LIRMM (rdv par mail)

Divers

- 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 , ulliana@lirmm.fr

Prelude to BDA

Relational Databases & UML

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. Relational Model

2. SQL

3. UML

BASIC RELATIONAL 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 clearly independent from the data to store.

- 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)

key Person1 , Person2 , Date --------> City
determines

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

Strongly recommended in systems (efficiency, coherence)

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.

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 : 3NF

[BD-G] chapter VI section 6.3

- This normal form is respected by **most** *"transactional-database"* you will find in **any** real world company
  - it allows to solve the problem of data redundancy
  - also, every schema can be normalized in 3NF

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

Create Table

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

```
CREATE TABLE Employee (

 id    NUMBER,
 name VARCHAR2(50),
 birthday DATE

)
```

Oracle Built-in 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 being 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

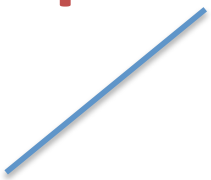
```
CREATE TABLE Employee (
id          NUMBER,
name        VARCHAR2(50),
birthday    DATE,

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

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

Summing Up

- Relations
  - Dependencies, Keys and Normal-forms
- SQL
  - CREATE, INSERT, DELETE, SELECT, GROUP-BY

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 : Plan

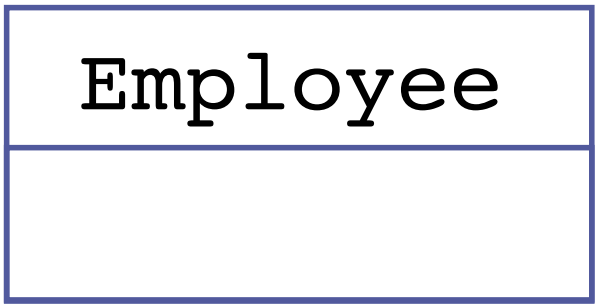
- UML Class Diagram
  - Basic constructs that can be used to model Relational Databases in UML
  - Real Object-oriented features

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

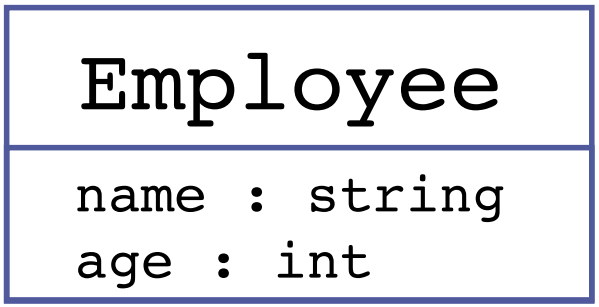


Class : Attributes

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

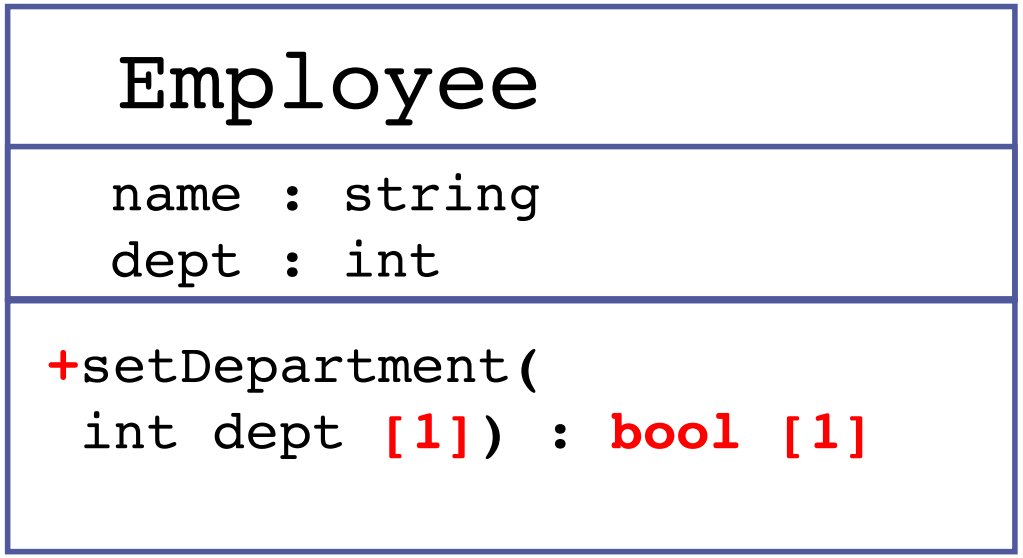
- Attributes denote the properties of class objects
  - They are usually typed

- Write the attributes below the class name



Operations can specify

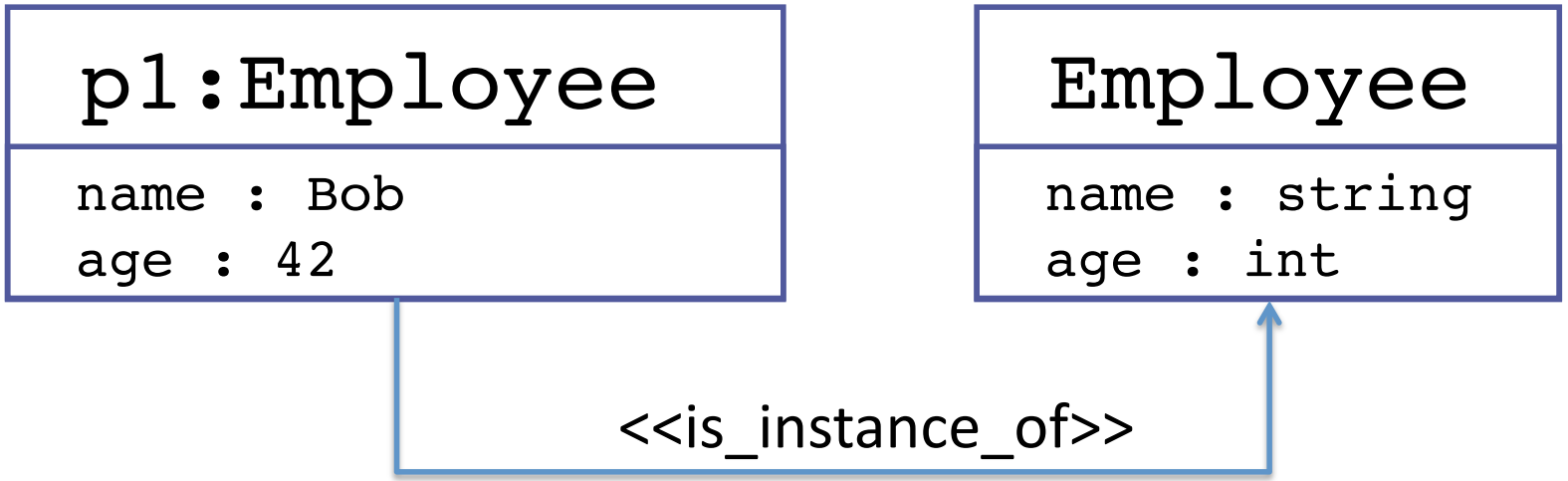
- Visibility (**+** public) (**-** private) (**#** protected)
- Return-type (*optional*; can be undefined)
- Multiplicity of parameter/return-type (*optional*)



Instances

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

- Related to their class by a directed edge

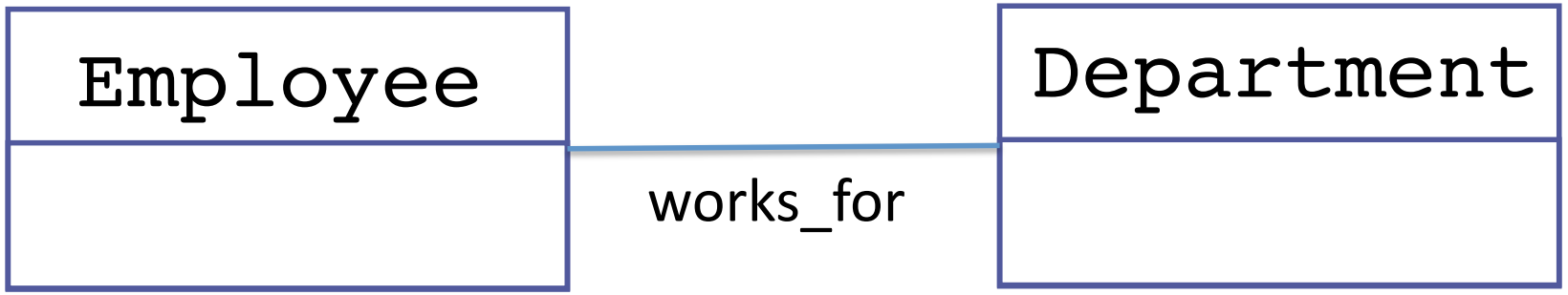


Binary Associations

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

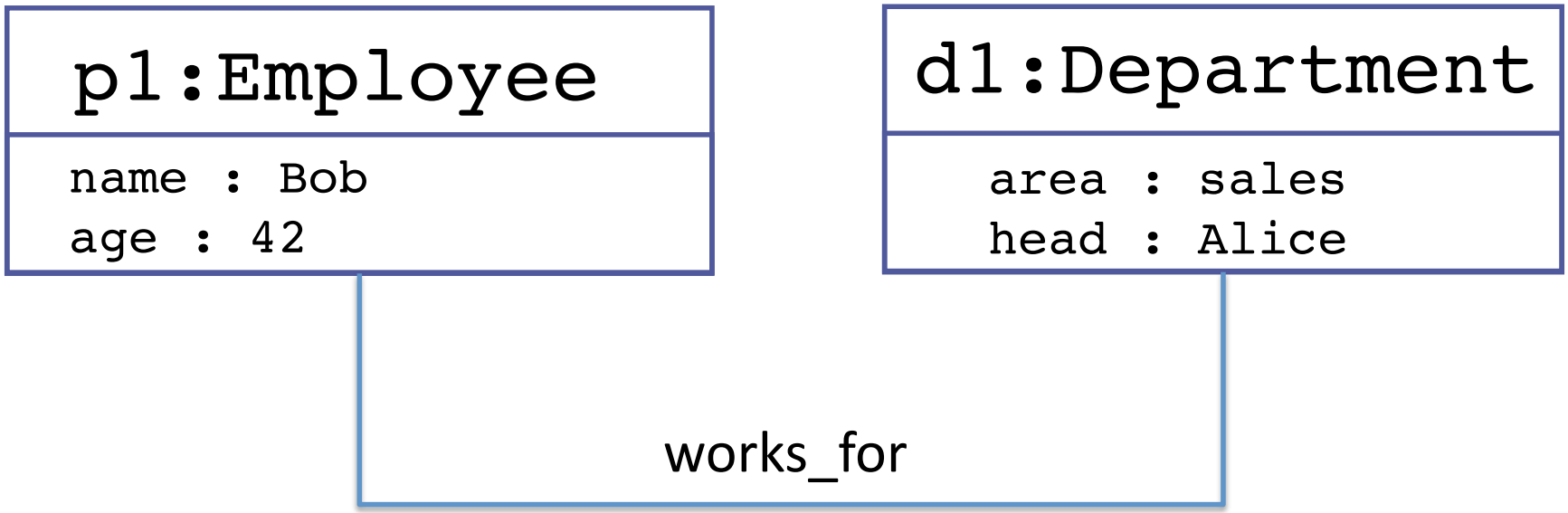
- General relationships between elements of two classes
  - ex. an employee works for a department
  - a concrete instance of association is called link

- Binary association : *undirected* edge between classes



Associations : Links

- Relationships between instances of classes



Associations : a tricky notation

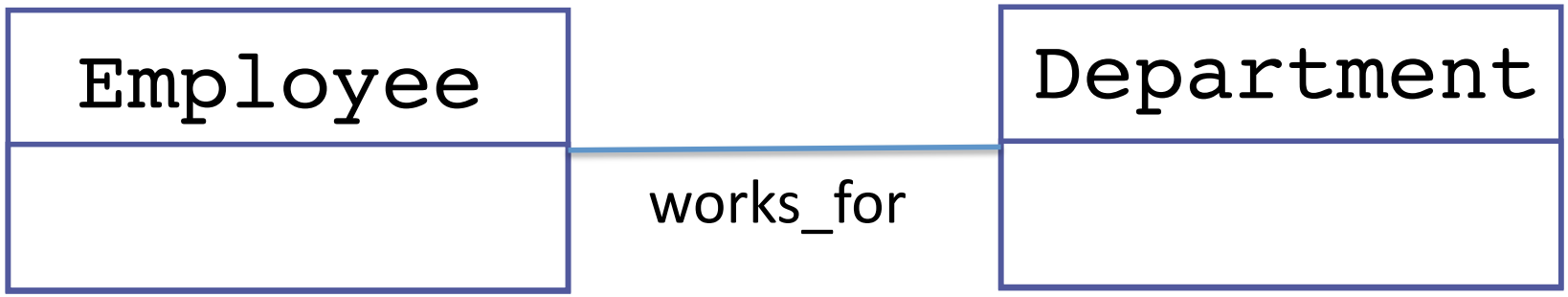
- Any association is specified by three things

1. Its name

2. The cardinality constraints of the class elements

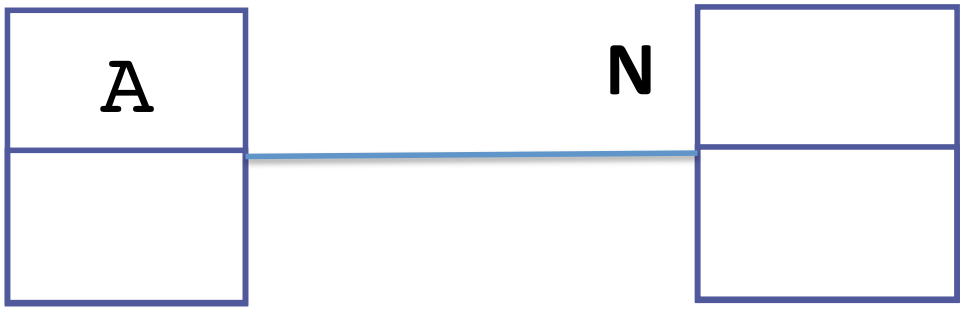
3. The role of the classes in the association (optional)

Associations : Name (1)



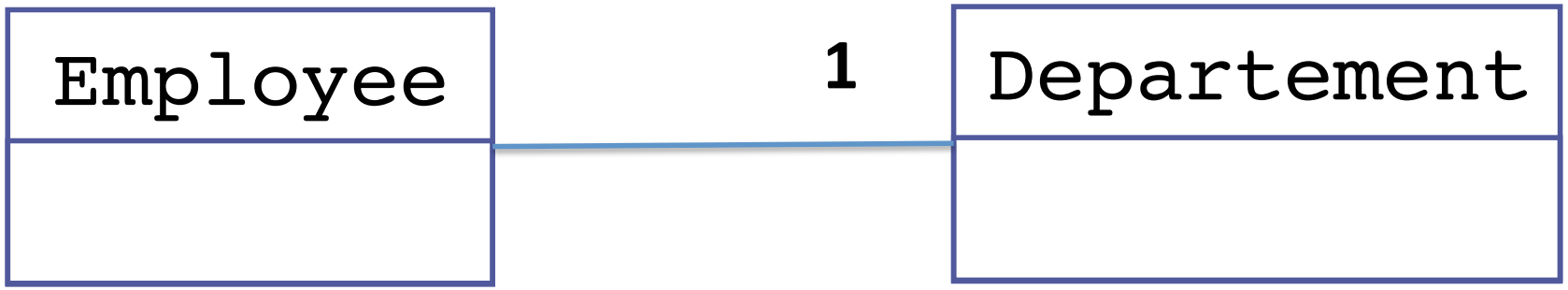
- The name is a label placed in the middle of the edge

Associations : Cardinality (2)



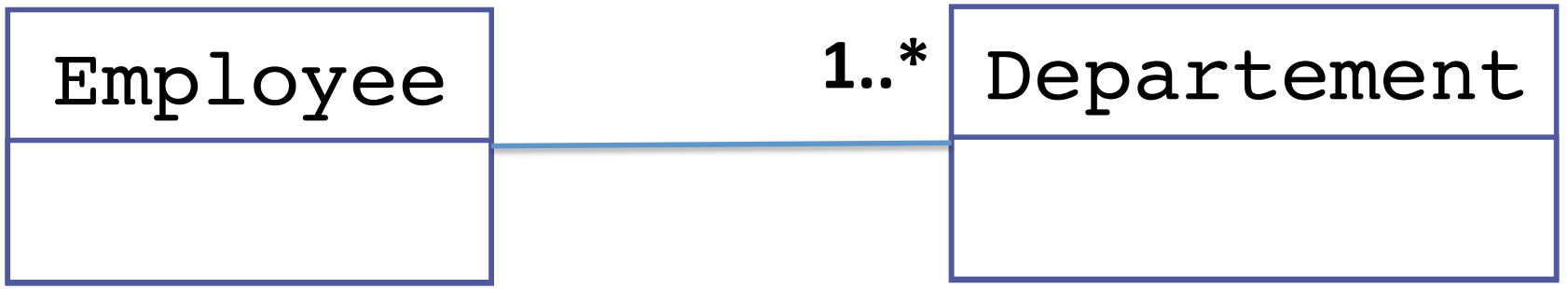
- This means that one instance of A participates in the association with **N** elements of the other class

Associations : Cardinality (2)



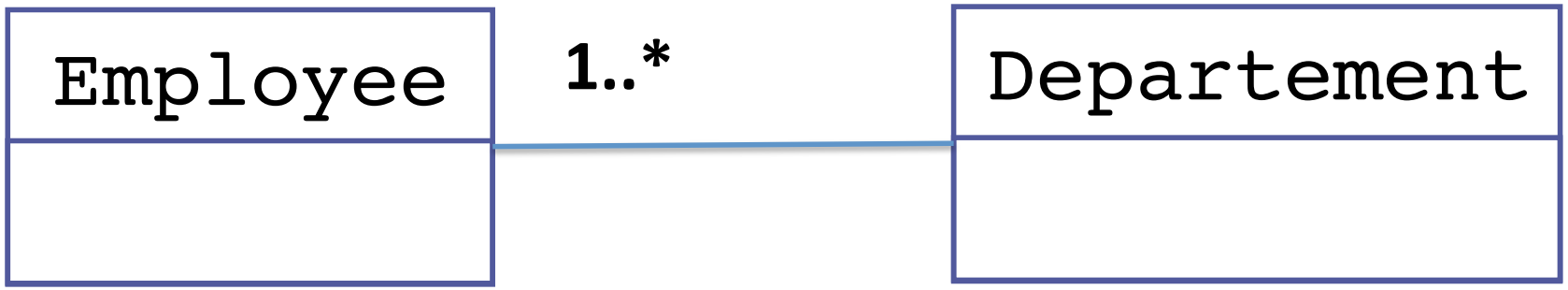
This means that an employee works for
exactly **1** department

Associations : Cardinality (2)



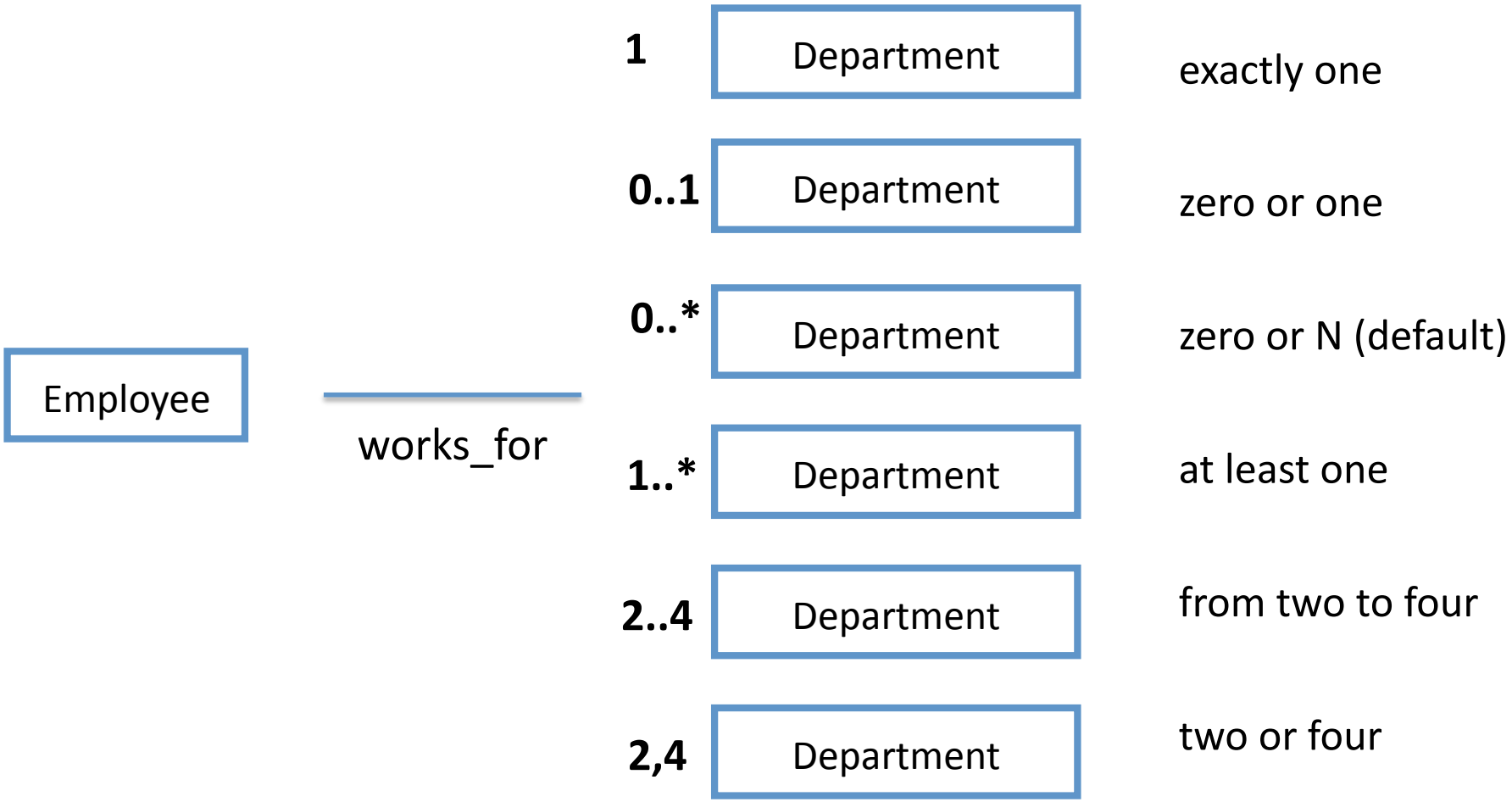
This means that an employee can work for
more than **1** department (but at least one)

Associations : Cardinality (2)

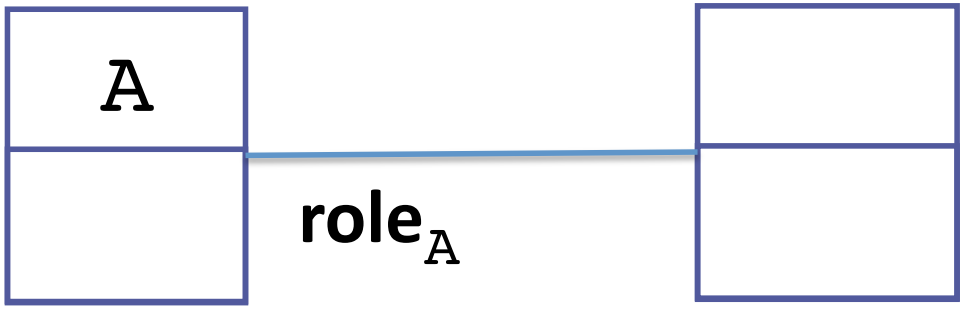


This means that a department has at least one employee, with no upper-limit.

Cardinality Specification

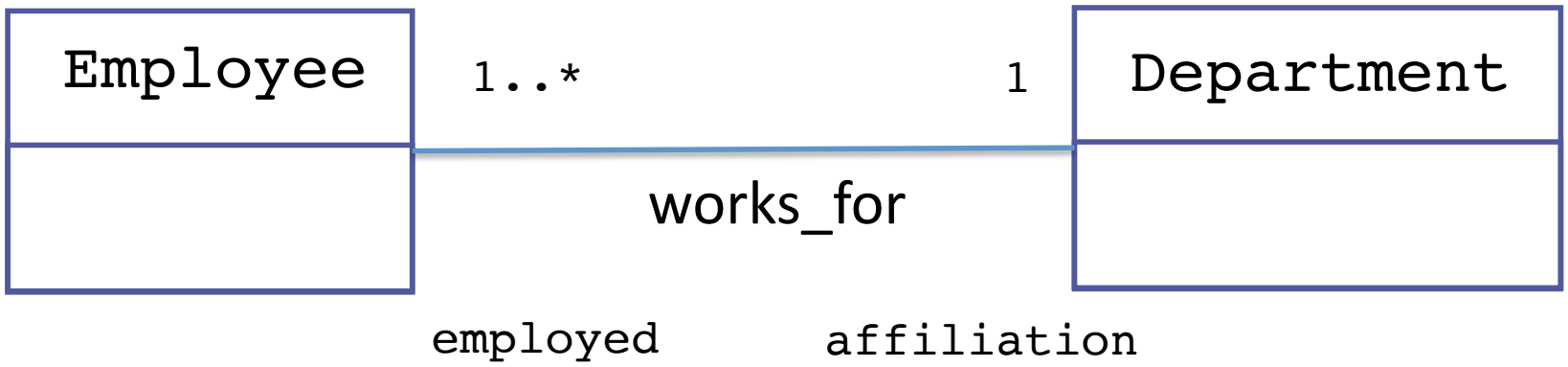


Associations : Roles (2)

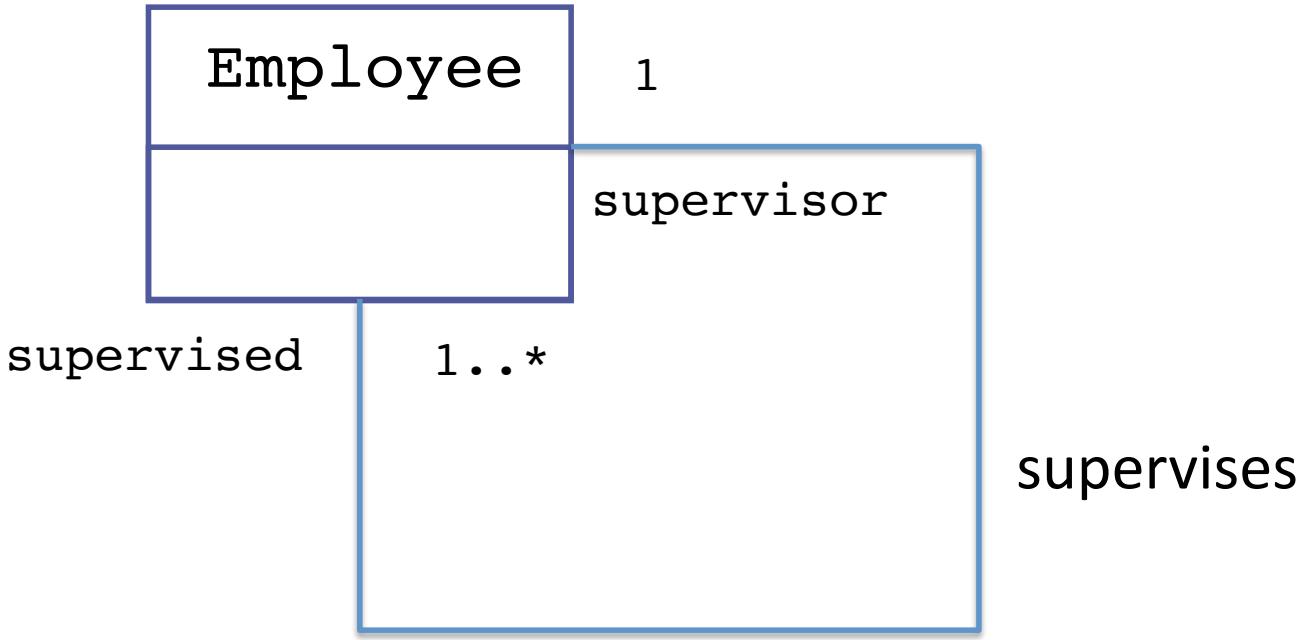


- This means that in the association an instance of A plays **role**$_A$

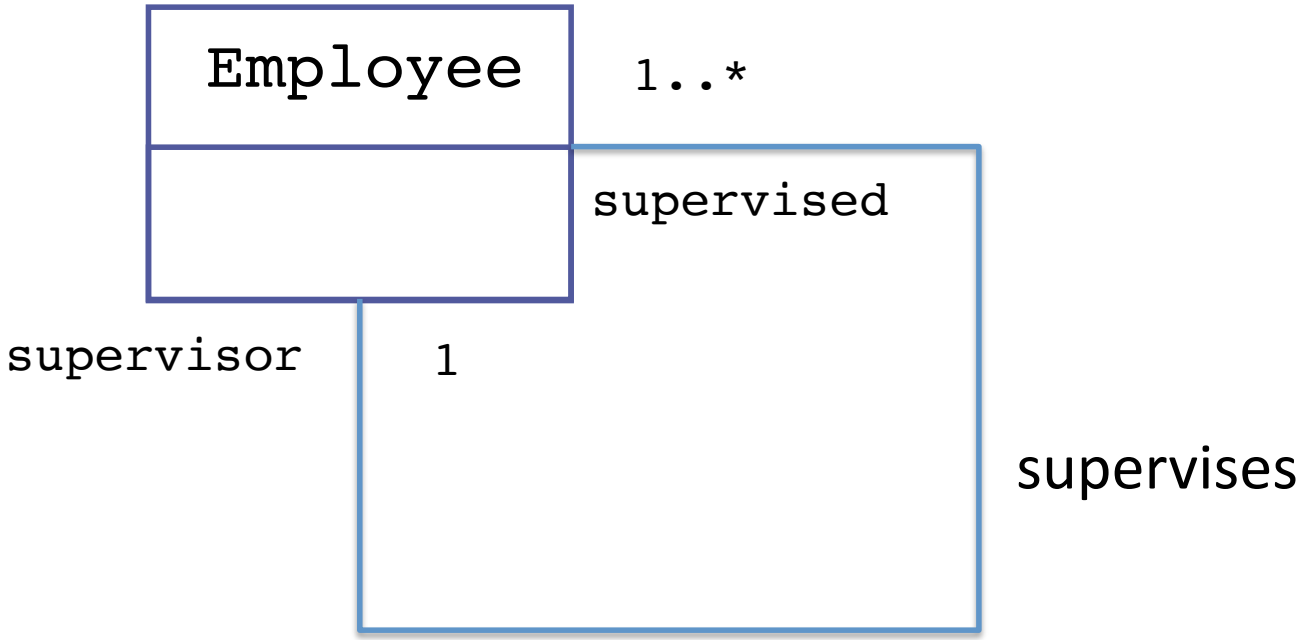
Putting everything together



Reflexive Associations

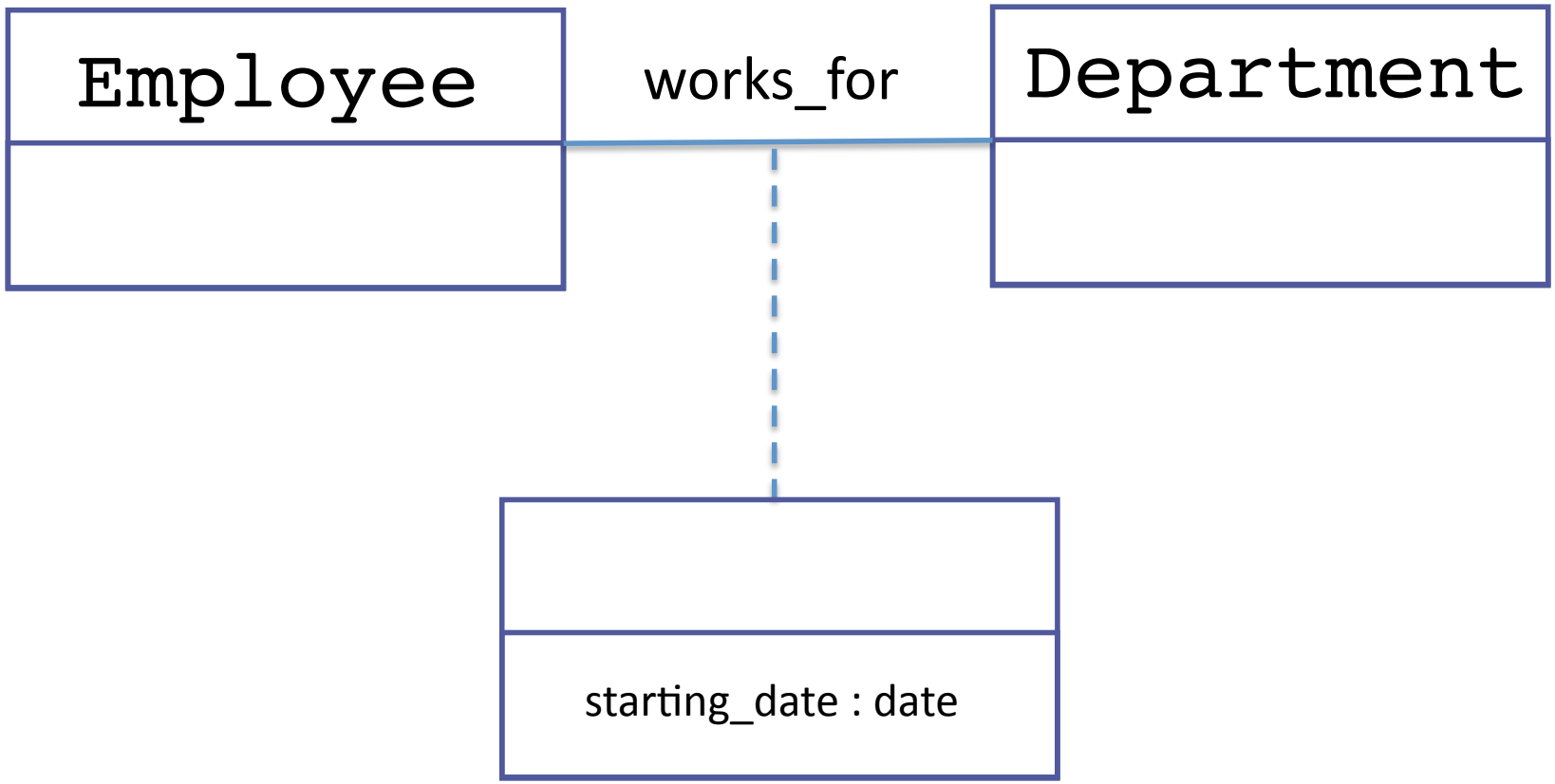


Equivalent Formulation



Notation for Attributes in Associations

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



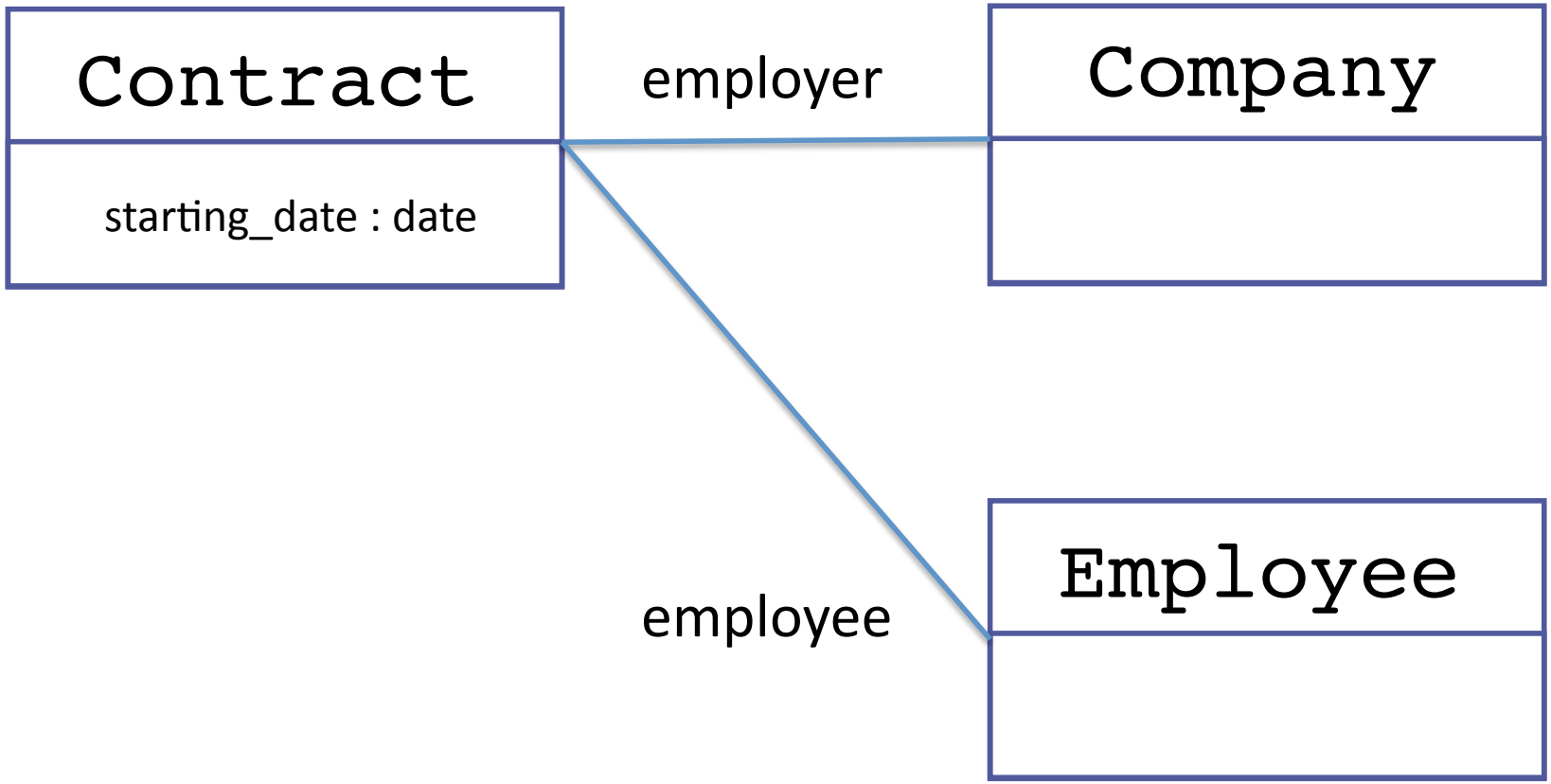
Why Associations can be Tricky

Often, a relatioship between two entities combines:

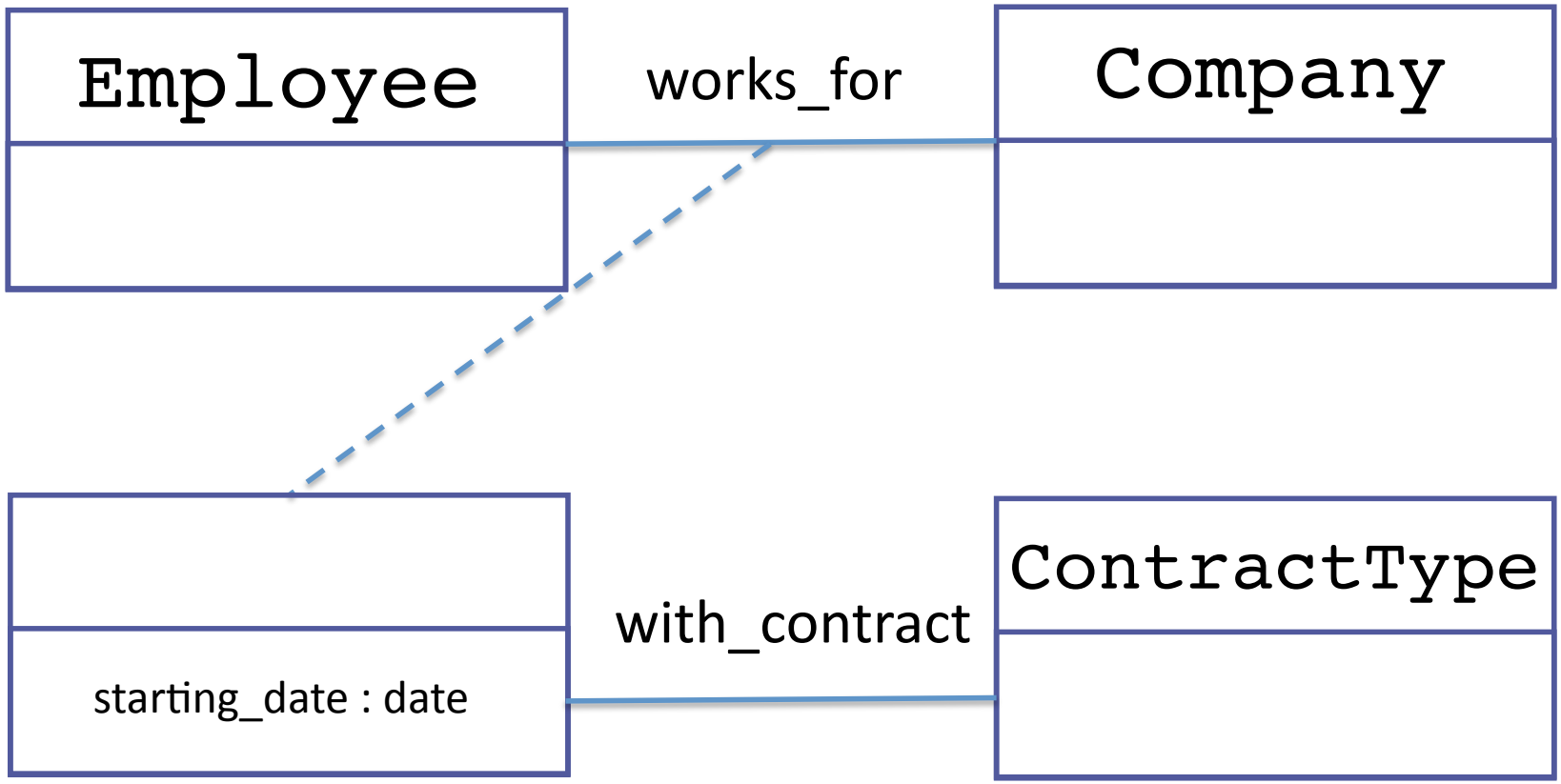
- ***association*** features
  - the fact that two or more things are linked
- **representation** features
  - the details about this association

- Ex: a working contract can be seen as a relationship (an association between an employee and a company) or as an entity (representation of a legal concept)
  - This duality is the source of all design problems !
    - Recognize your own modelling choices !

Why Associations can be Tricky

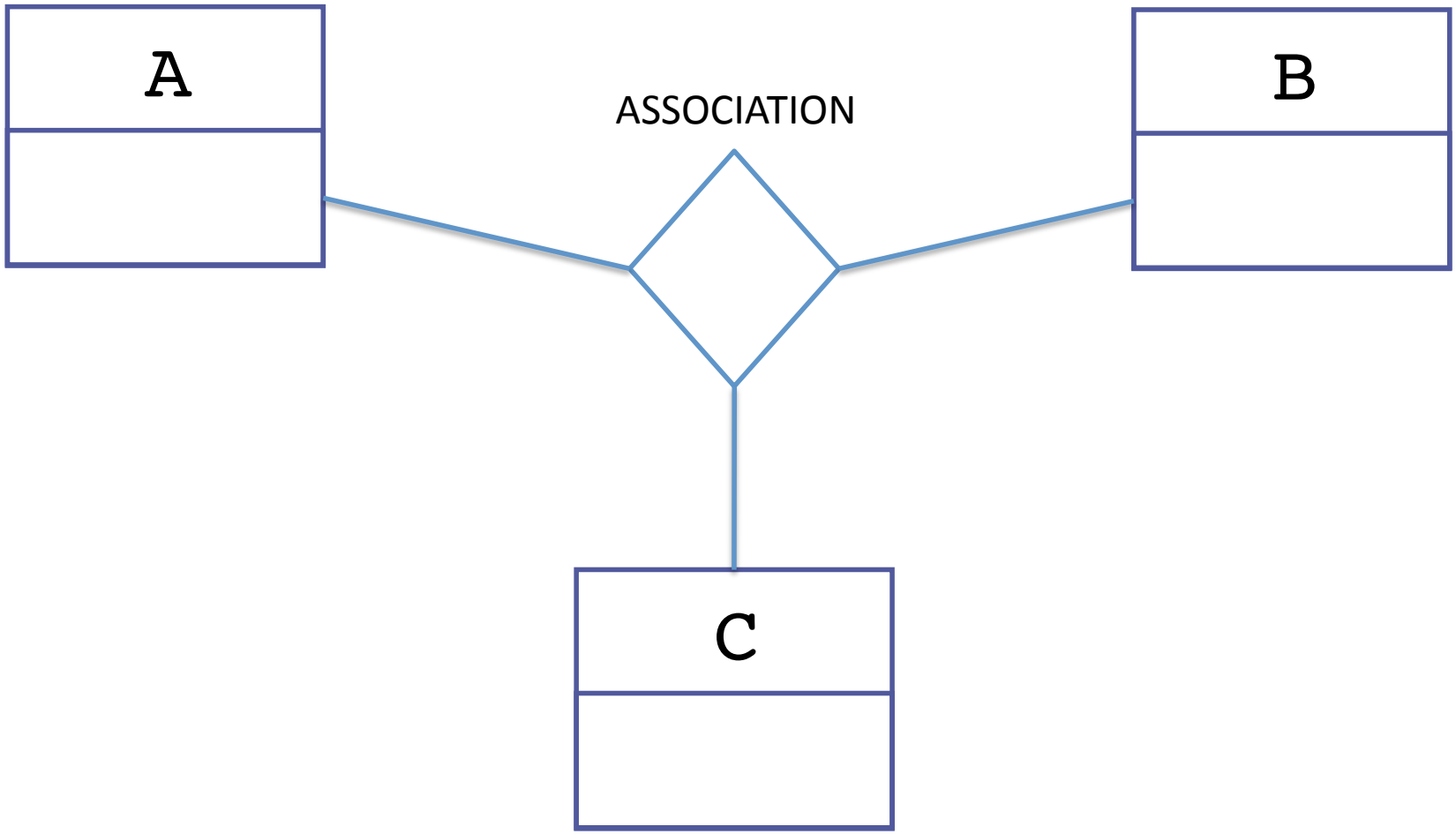


Associations Participating in Other Associations



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

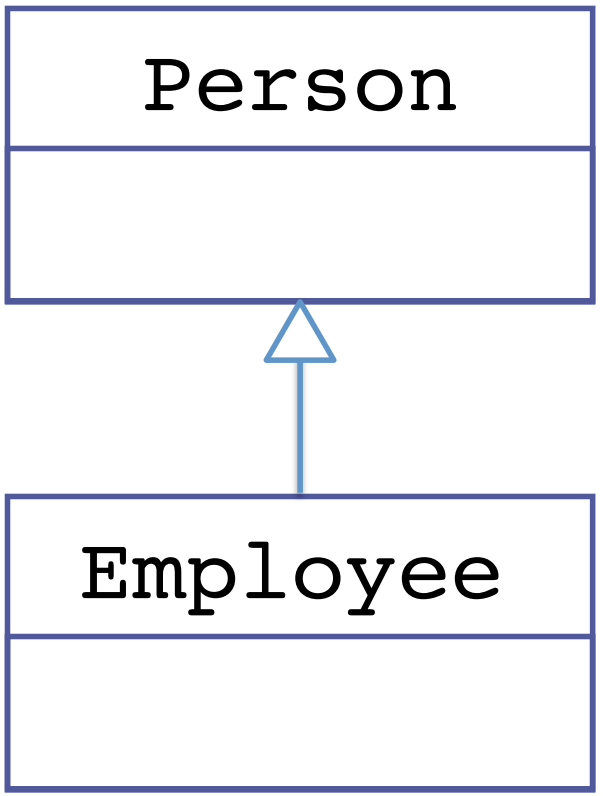
- The construct of the language we have seen so far are enough to model relational databases

SubClass

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

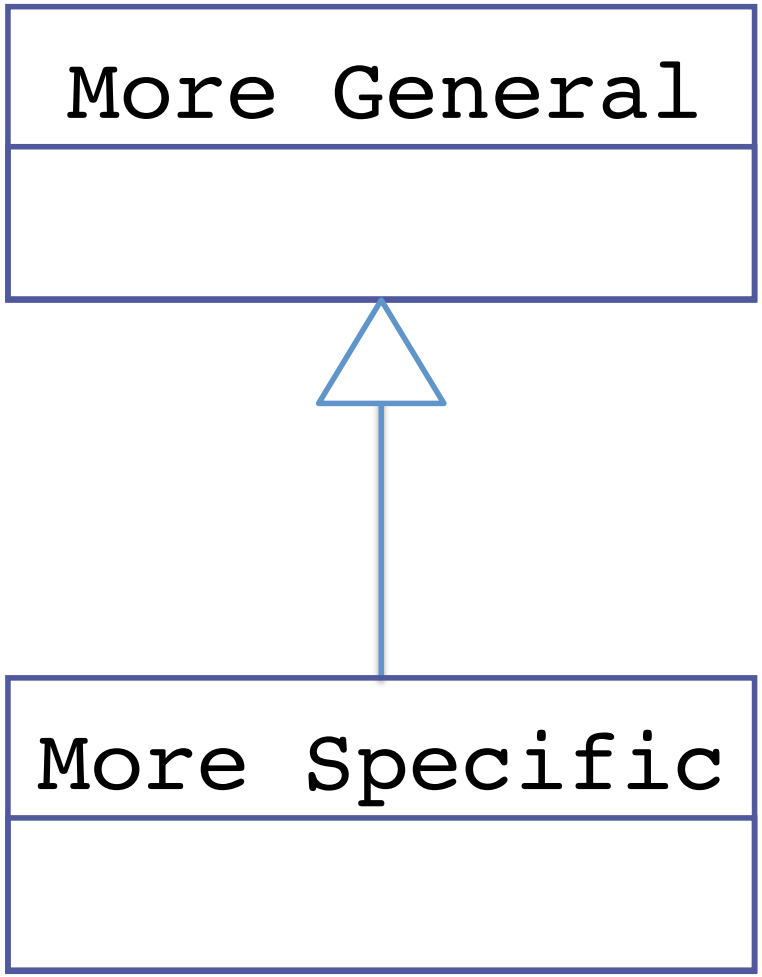
- A subset of the instances of a Class
  - ex. every employee is a person

- A subclass inherits all superclass properties or redefines them.



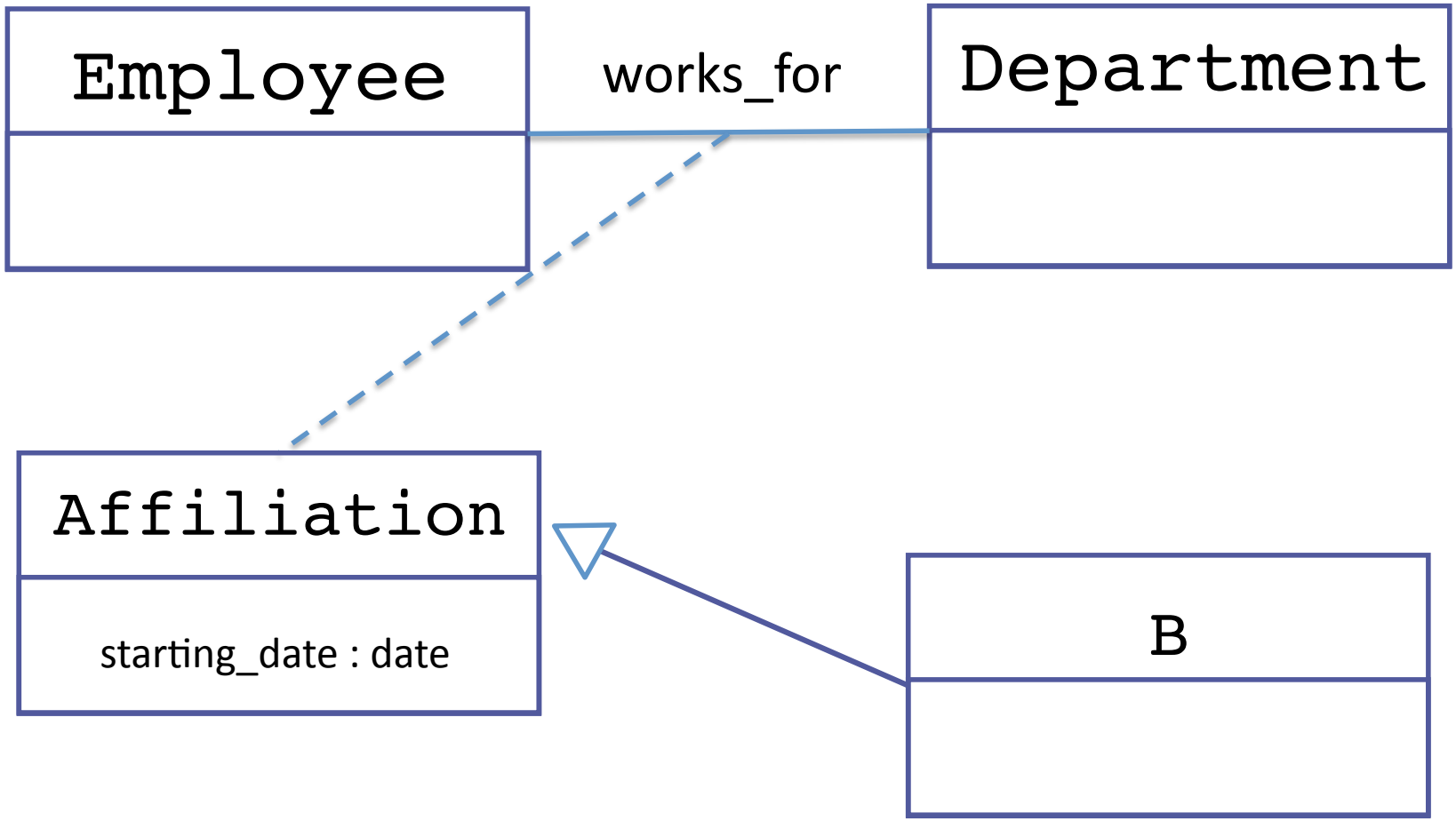
SubClass = Generalization/Specialization

[UML] section 5

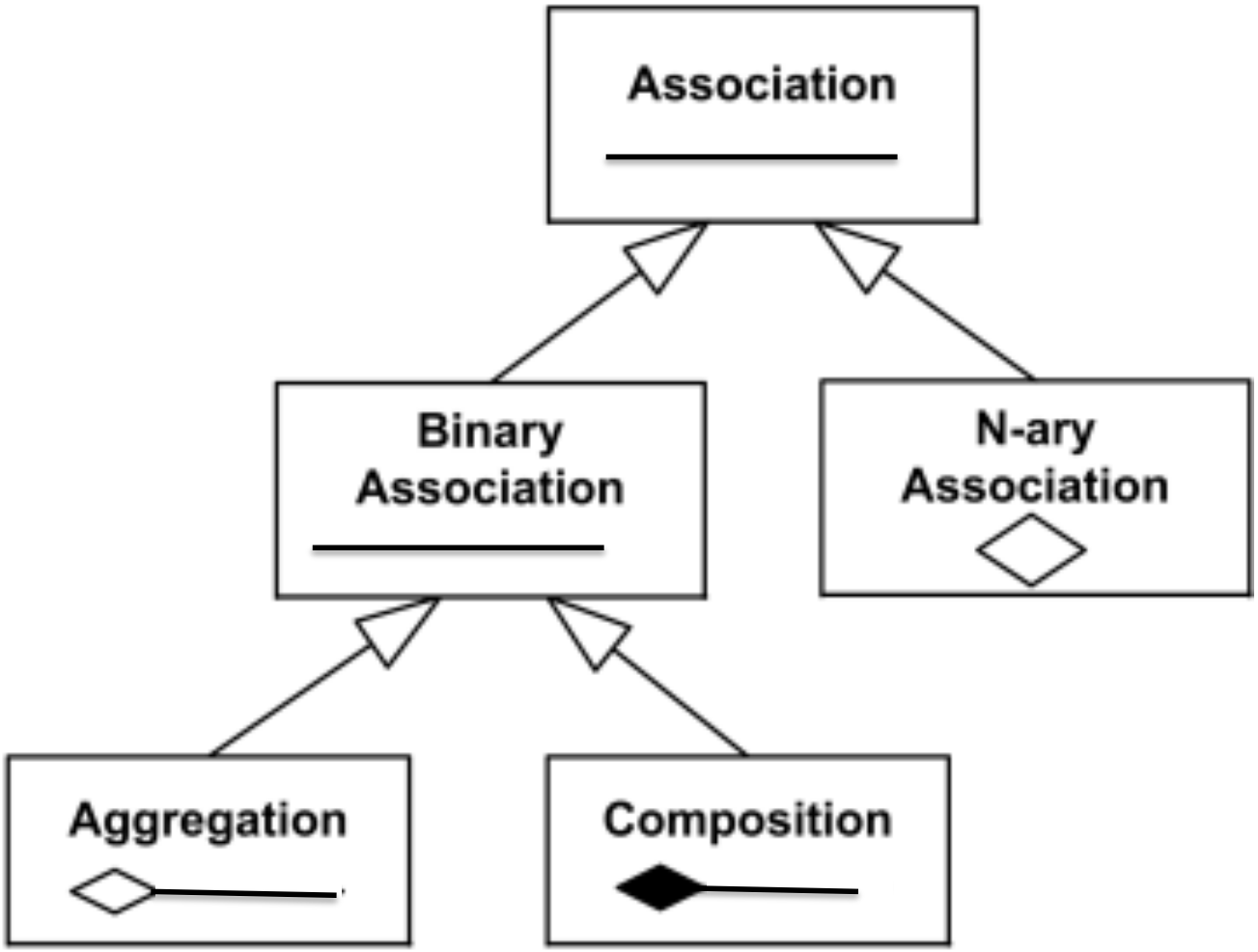


Association modeled by a Class

[UML] section 4.3



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


Association
Binary Association
N-ary Association
Aggregation
Composition

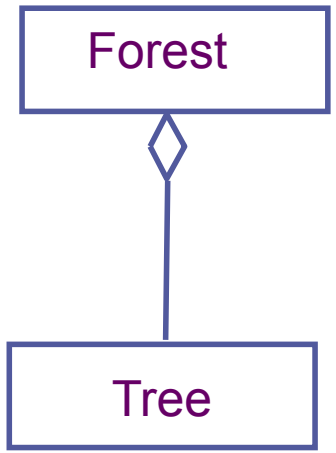
Part-Of (binary) Association

- “A forest is made of trees”

Aggregation

[BD-G] chapter XVII section 2.2

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



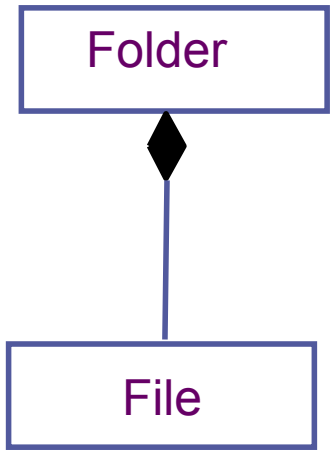
Part-Of (binary) Association

- “A folder is made of files”

Composition

[BD-G] chapter XVII section 2.2

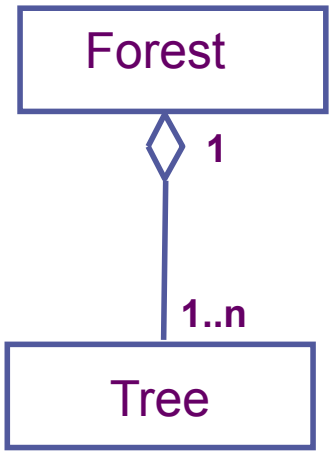
- Consequence 1 : a file cannot exists without a folder



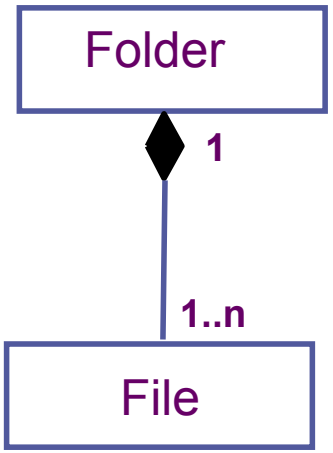
Aggregation Composition

[BD-G] chapter XVII section 2.2

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

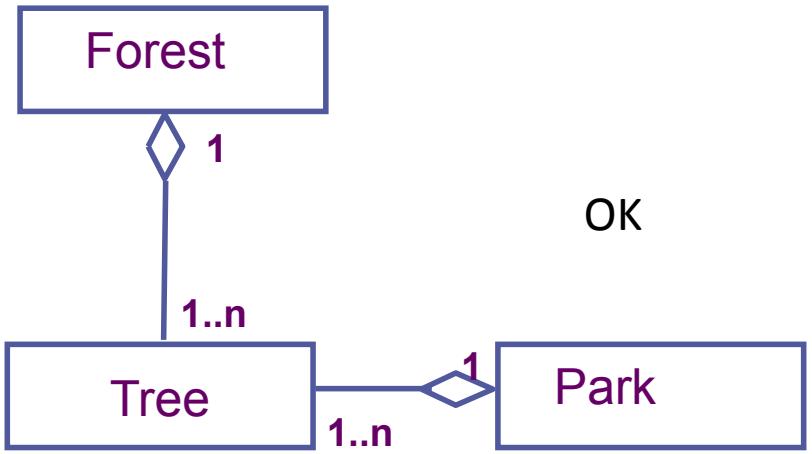


- Consequence 1 : a file <u>cannot</u> exists without a folder



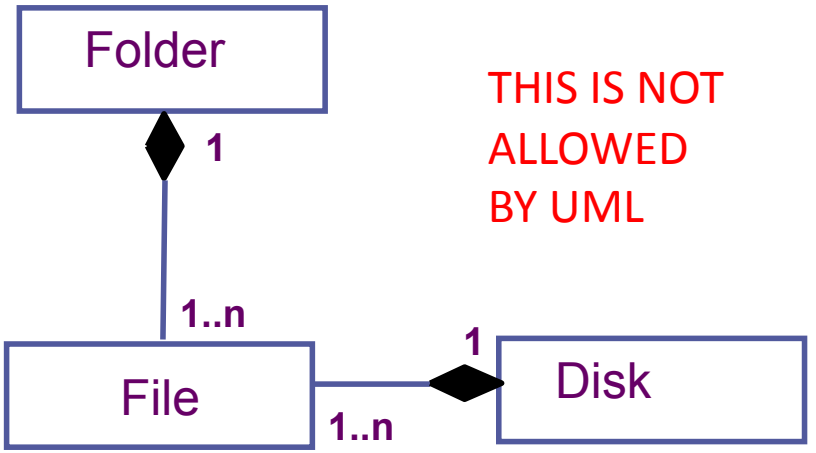
Aggregation

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



Composition

- Consequence 1 : a file <u>cannot</u> exists without a folder



CONSTRAINTS

Fine-grain Modelization : Constraints

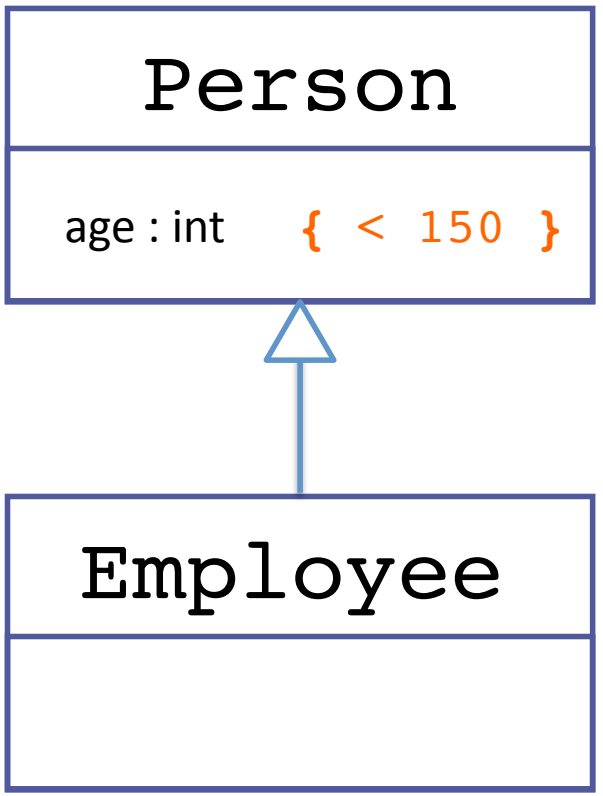
[UML] section 6

- Conditions/Restrictions on classes, attributes and associations that must hold at any time.
- Syntax `{ this is a constraint }`
- OCL (object constraint language) or other

Attribute constraints

- Constraints are inherited

- Consequence : the age of an employee is less than 150

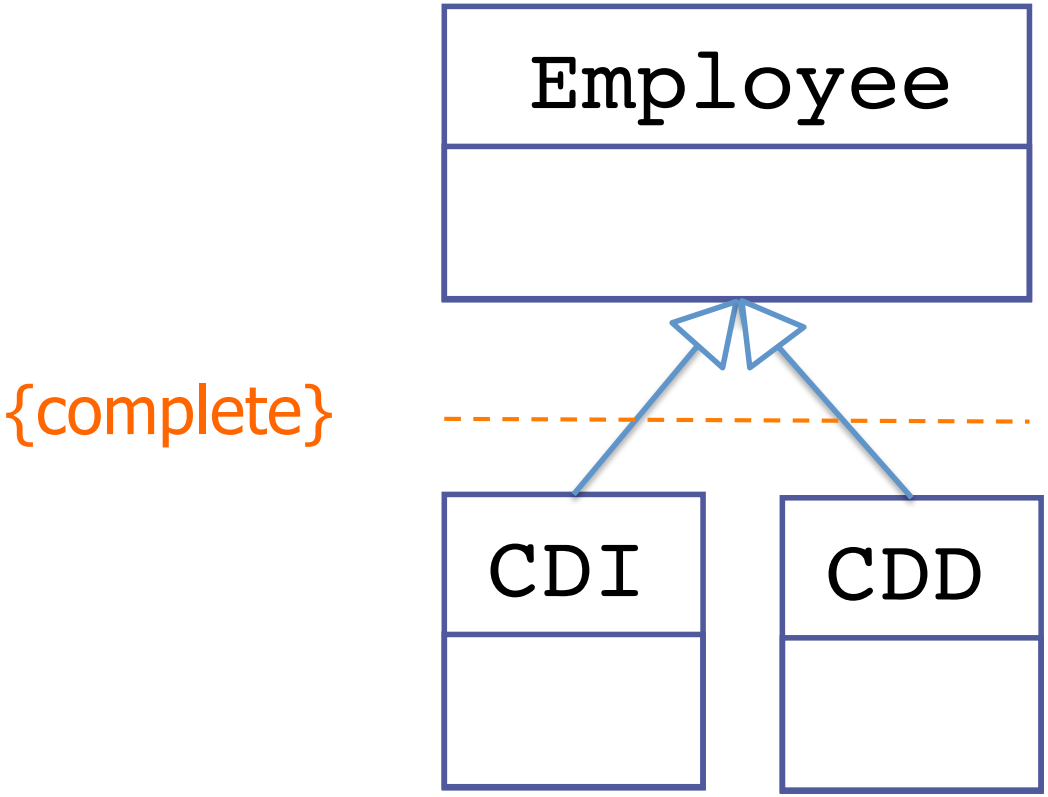


Constraints on Subclasses

[UML] section 5.2

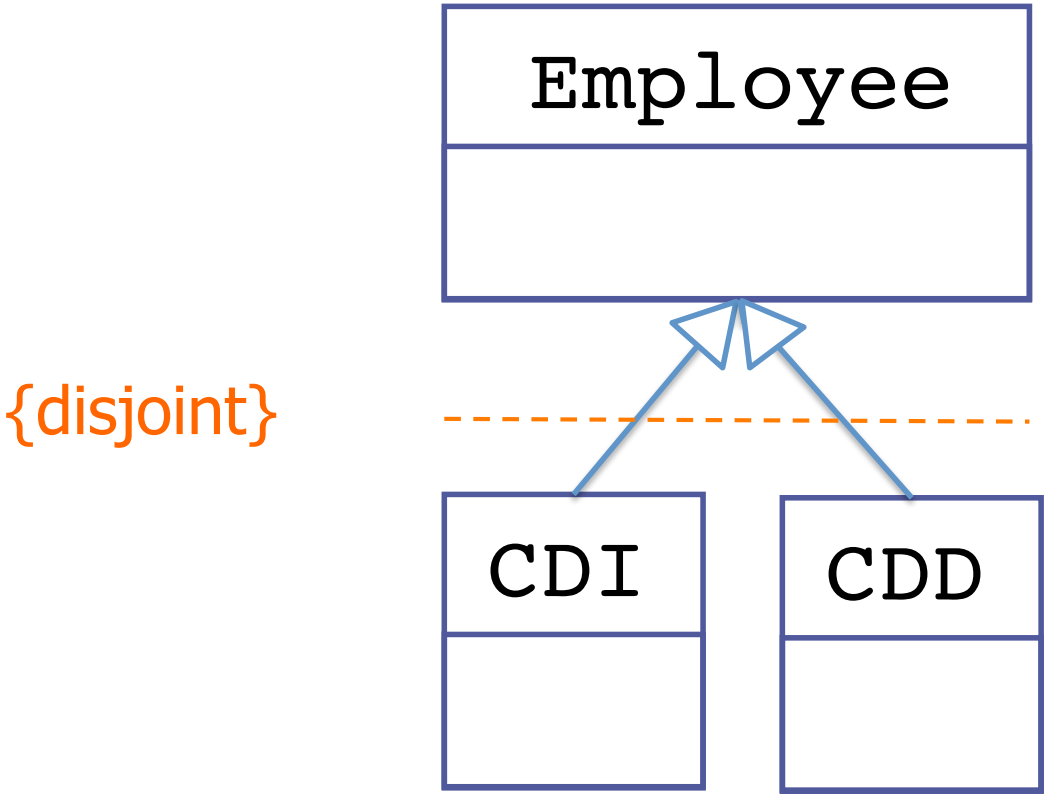
- Complete / Incomplete
- Disjoint / Overlapping

Complete / Incomplete



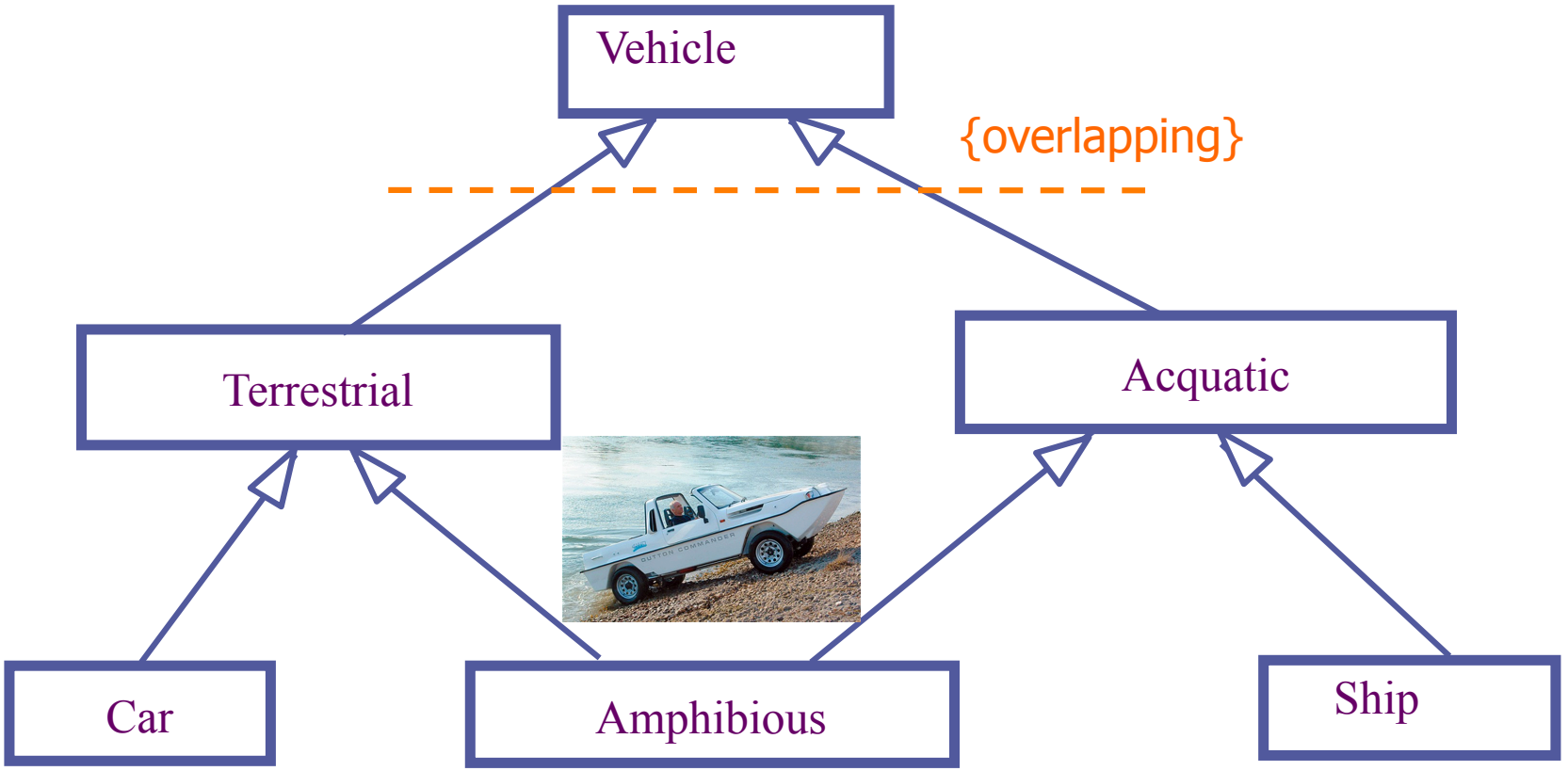
- Consequence : an employee is either CDI, or CDD, or both

Disjoint / Overlapping



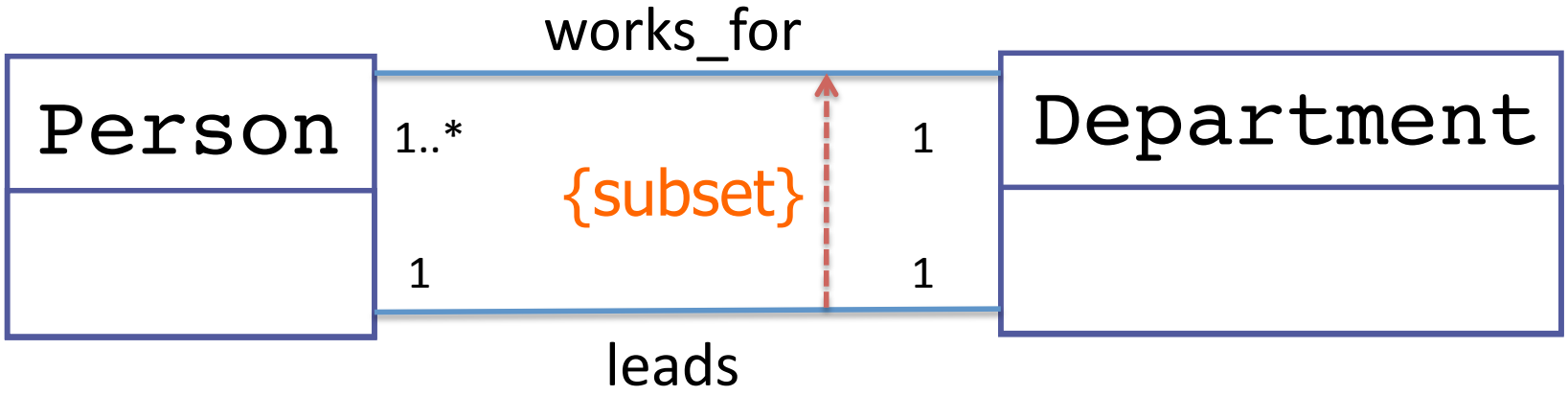
- Consequence : an employee is either CDI, or CDD, ~~or both~~

Disjoint / Overlapping



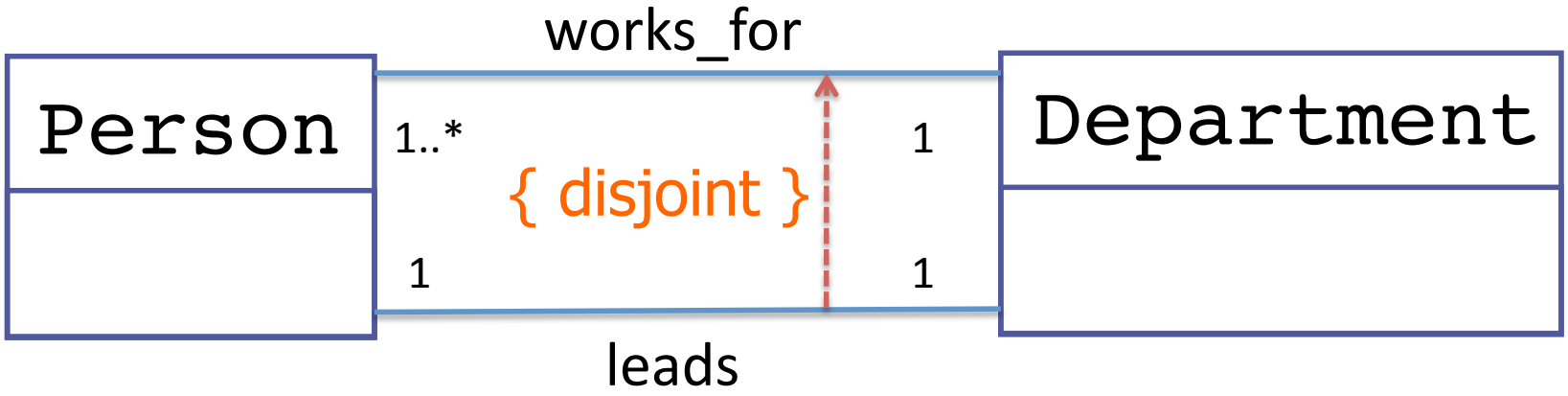
- Consequence : some terrestrial vehicle can also be an acquatic vehicle

Constraints on Associations



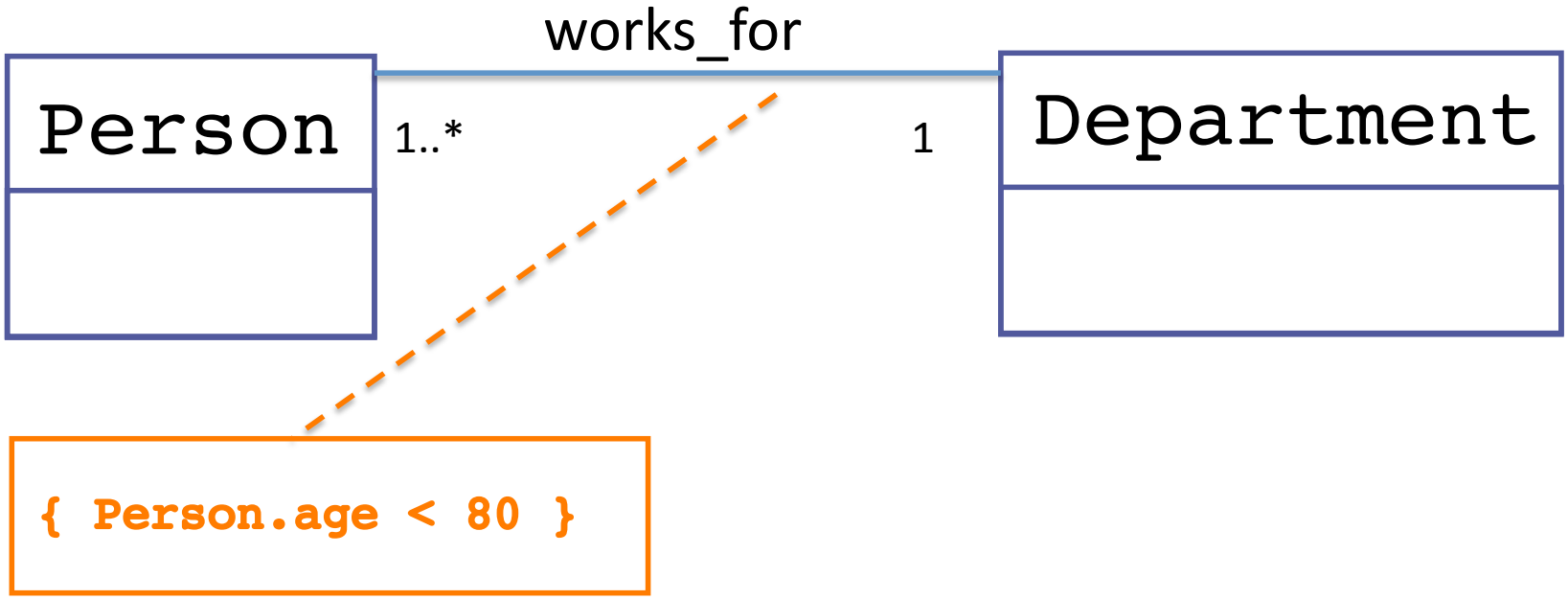
- Consequence : every leads association is also a works_for association

Constraints on Associations



- Consequence : a person cannot lead and work for the same department

Constraints on Associations



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

Summing Up

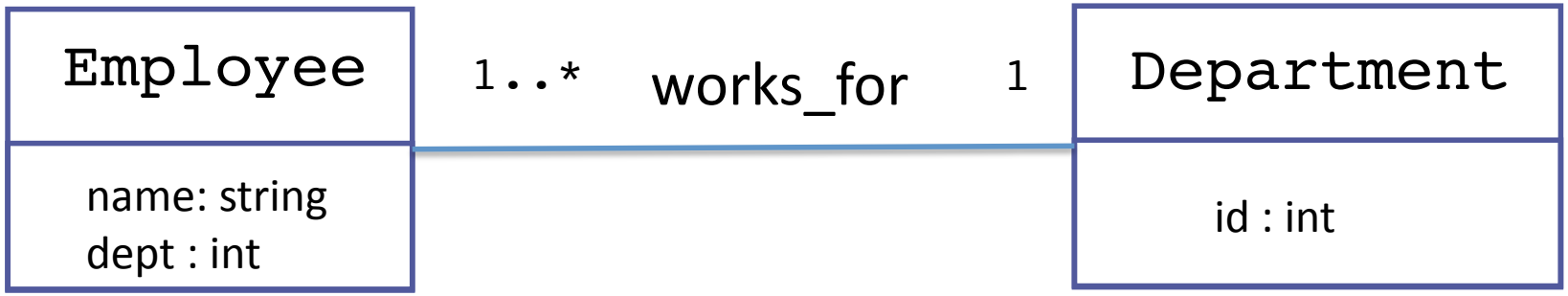
- Instances/Links/Operations
- Subclasses
- Aggregation and Composition
- Constraints

RELATIONAL IMPLEMENTATION OF UML DIAGRAMS

Levels fo Modelling

- Conceptual Model (UML;EA)
- Logical Model (Relational, Object, Graph)
- Physical Model (SQL, OQL, XML)

Associations 1:N



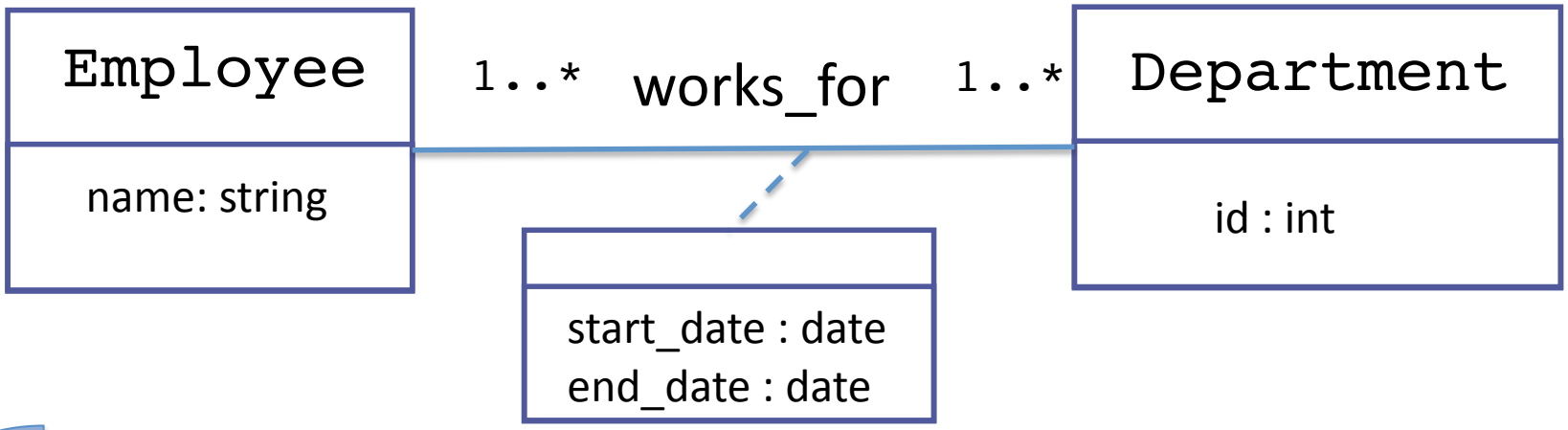
Employee

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

Department

id	**name**
dep1	Sales
dep2	Production

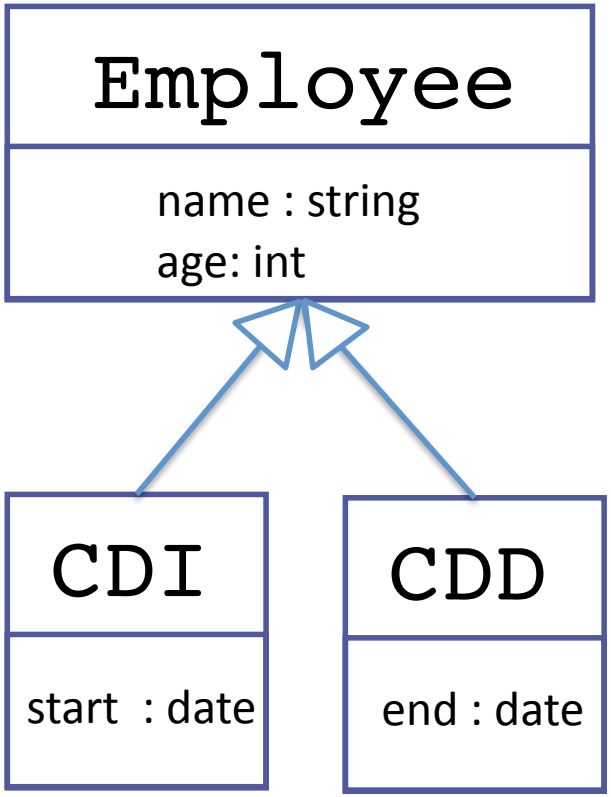
Associations N:M



Works_for

name	id	start date	end date
Alice	dep1	01/12/15	31/12/15
Bob	dep2	02/08/15	03/12/15
Eddy	dep1	14/08/15	31/12/15

Subclasses



- Three solutions

(1) Fix Top-Class and Copy Bottom-attributes

Employee

name	start	end	age
Alice	null	1/1/15	20
Bob	3/4/10	null	42
Eddy	5/8/09	null	28

- Redundancy ↗
  - lots of null values
- Flexibility ↘
  - Add colums for new subclasses
- Query Cost ↘ (no joins)

(2) Keep Bottom-Classes ; Copy Top-Class Attributes

CDI

name	start	age
Bob	3/4/08	42
Eddy	5/8/09	28

CDD

name	end	age
Alice	1/1/15	20

- Redundancy ↘
- Flexibility ↗
  - Add tables for new subclasses
- Query Cost ↗
  - but only when subclasses are joined or put in union

(3) : Keep all-classes ; Copy Top-key

CDI

name	**start**
Bob	3/4/10
Eddy	5/8/09

CDD

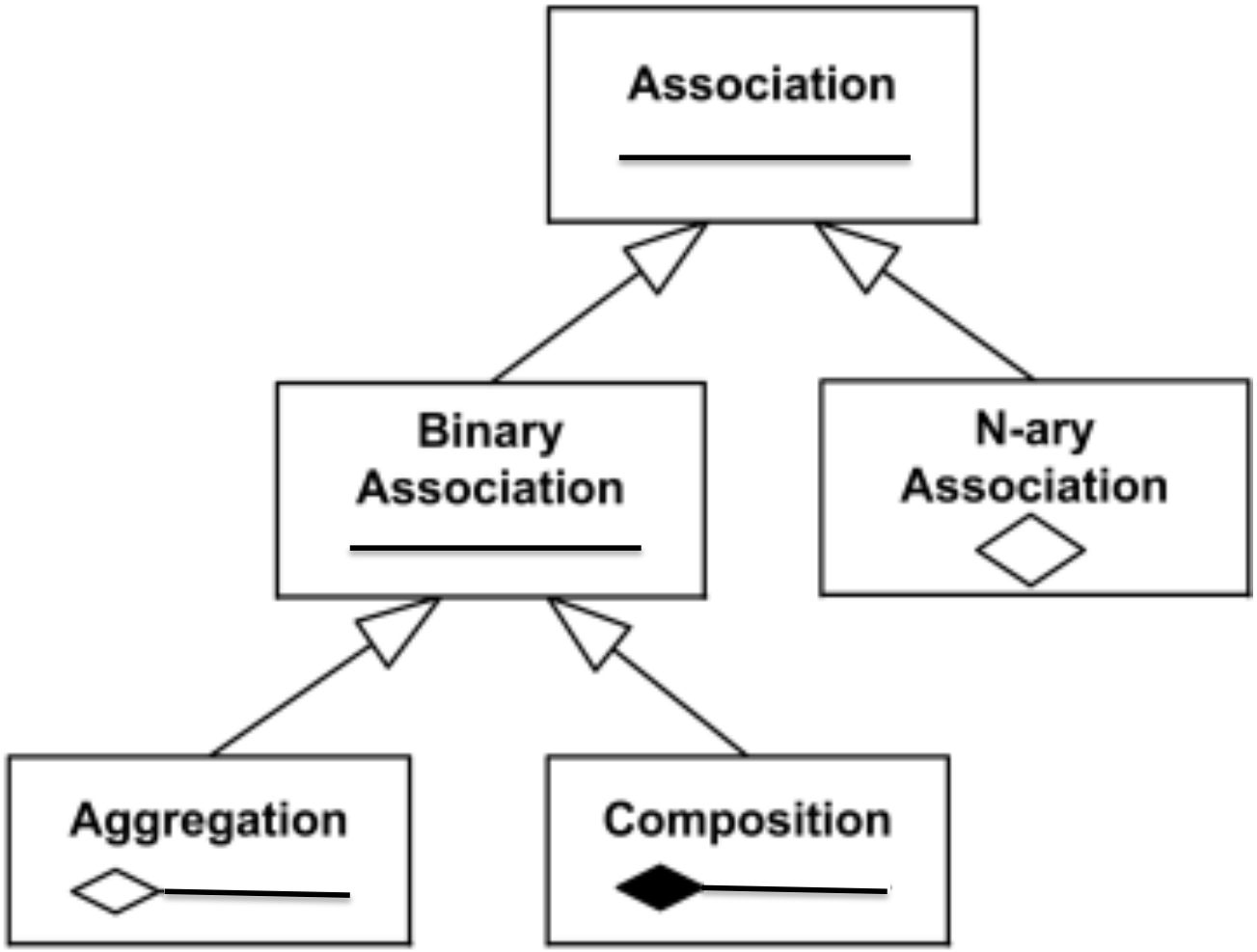
name	**end**
Alice	1/1/15

Person

name	**age**
Alice	20
Bob	42
Eddy	28

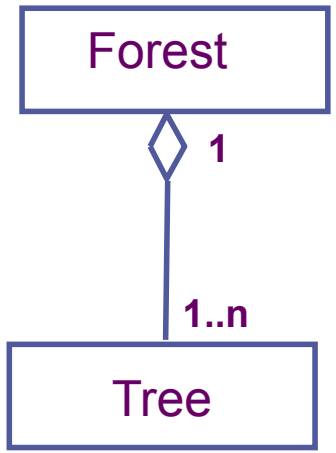
- Redondance ↘
- Flexibility ↗
- Query Cost ↗
  - a lot of joins between subclass/superclass

Back on Associations



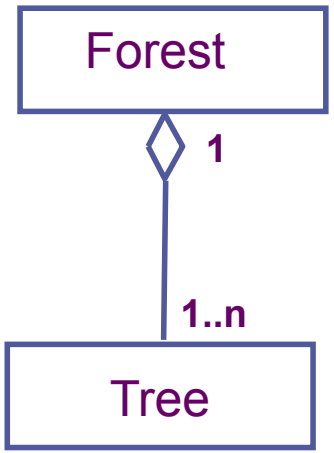
Aggregation

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



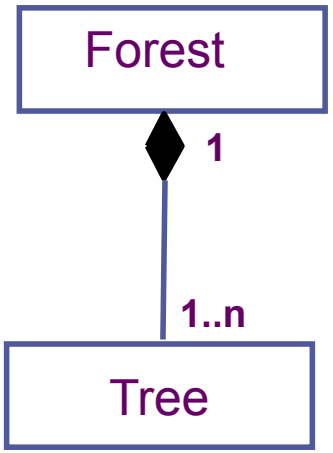
Aggregation

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



Composition

- Consequence 1 : a tree cannot exists without a forest



Translating Aggregation

Forest

id
1
2

Tree

name	forest
T	1
S	*null*


```
CREATE TABLE Tree (
name   : varchar2(50)  PRIMARY KEY,
forest : number        FOREIGN KEY
                  REFERENCES Forest(id) )
```

Translating Composition (1)

Forest

id
1
2

Tree

name	forest
T	1
S	2

```
CREATE TABLE Tree (
name   : varchar2(50)  PRIMARY KEY,
forest : number   NOT NULL  FOREIGN KEY
                  REFERENCES Forest(id) )
```

Translating Composition (2)

Forest

id
1
2

Tree

name	forest
T	1
S	2

```
CREATE TABLE Tree (
name   : varchar2(50)  PRIMARY KEY,
forest : number   NOT NULL  FOREIGN KEY
                  REFERENCES Forest(id)
                  ON DELETE CASCADE
                  ON UPDATE CASCADE      )
```

Translating Composition (3)

Forest

id
1
2

Tree

name	forest
T	1
S	2

```
CREATE TABLE Tree (
name   : varchar2(50)  PRIMARY KEY,
forest : number   NOT NULL  FOREIGN KEY
                  REFERENCES Forest(id)
                  ON DELETE CASCADE
                  ON UPDATE CASCADE      )
```

Translating Composition (3)

Forest

id
1
✗

Tree

name	forest
T	1
✗	✗

```
CREATE TABLE Tree (
name   : varchar2(50)  PRIMARY KEY,
forest : number   NOT NULL  FOREIGN KEY
                  REFERENCES Forest(id)
                  ON DELETE CASCADE
                  ON UPDATE CASCADE      )
```

Translating Composition (4)

Forest

id
1

Tree

name	forest
T	1

```
CREATE TABLE Tree (
name   : varchar2(50)  PRIMARY KEY,
forest : number   NOT NULL  FOREIGN KEY
                  REFERENCES Forest(id)
                  ON DELETE CASCADE
                  ON UPDATE CASCADE      )
```

Translating Composition (5)

Forest

id
333

Tree

name	forest
T	1

```
CREATE TABLE Tree (
name   : varchar2(50)  PRIMARY KEY,
forest : number   NOT NULL  FOREIGN KEY
                  REFERENCES Forest(id)
                  ON DELETE CASCADE
                  ON UPDATE CASCADE      )
```

Translating Composition (5)

Forest

id
333

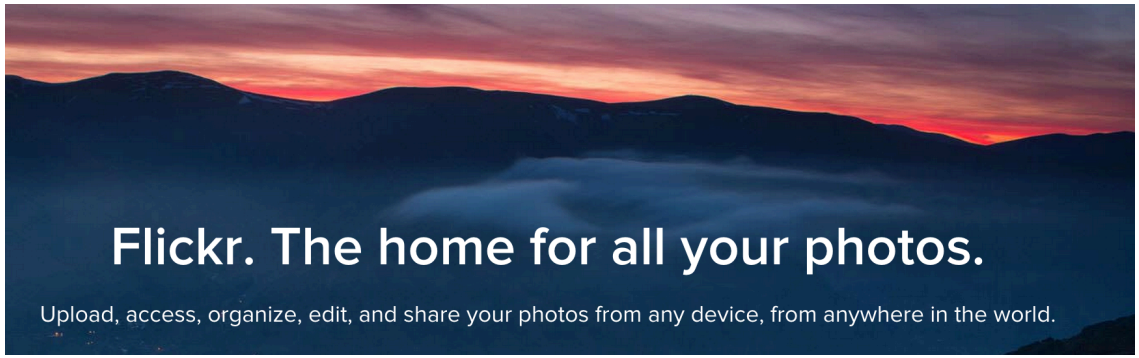
Tree

name	forest
T	333

```
CREATE TABLE Tree (
name   : varchar2(50)  PRIMARY KEY,
forest : number   NOT NULL  FOREIGN KEY
                  REFERENCES Forest(id)
                  ON DELETE CASCADE
                  ON UPDATE CASCADE      )
```

TP

- Tâche 0 : Overture des comptes Oracle
  - instructions sur Moodle



- Étude de la base de données de flickr
  1. Modèle UML et Relationnel
  2. SQL, Mises à jour, Requêtes

Readings

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

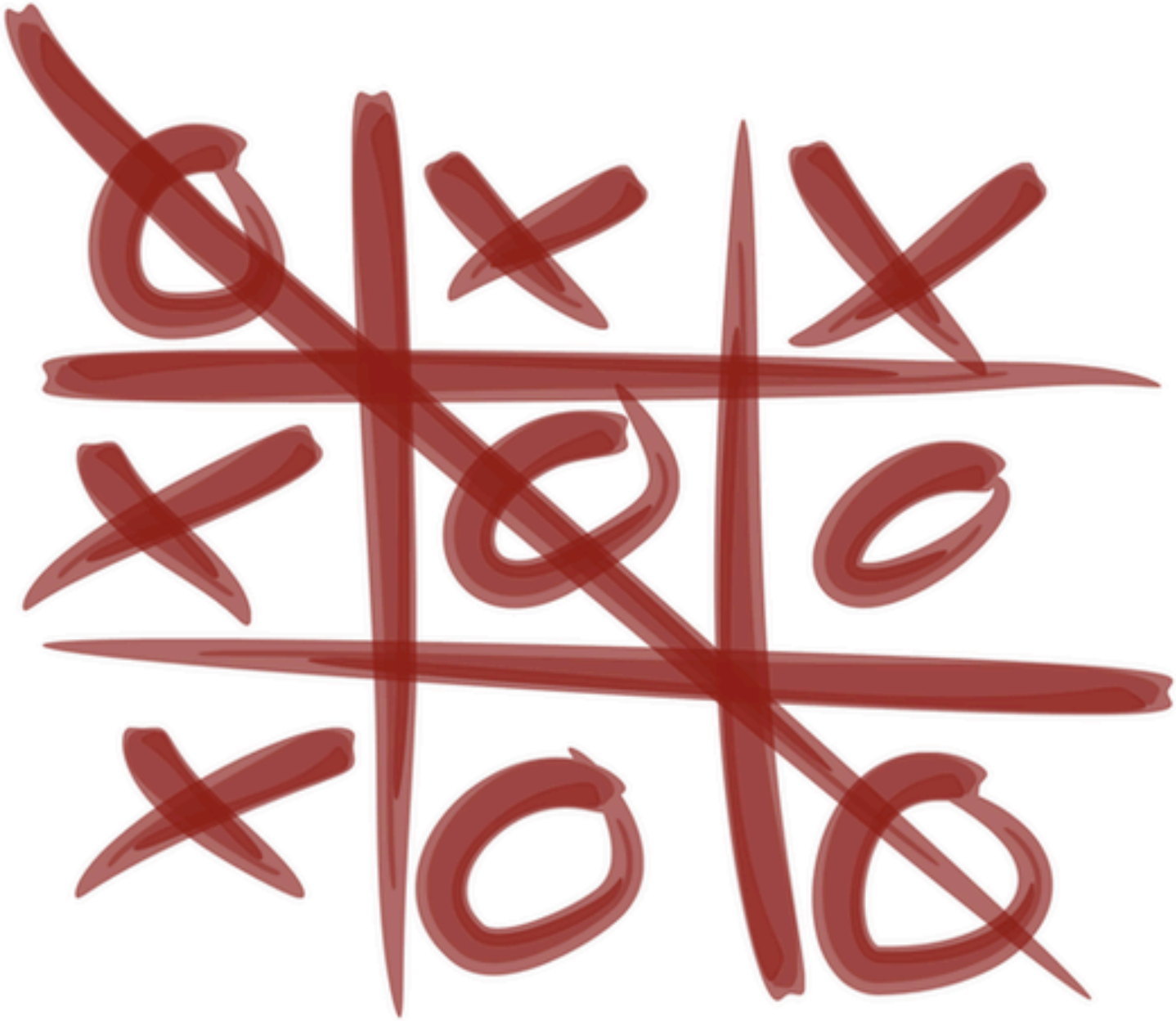
[BD-G] Bases de données,
Georges Gardarin, 5ème edition 2005
http://georges.gardarin.free.fr/Livre_BD_Contenu/XX-TotalBD.pdf

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

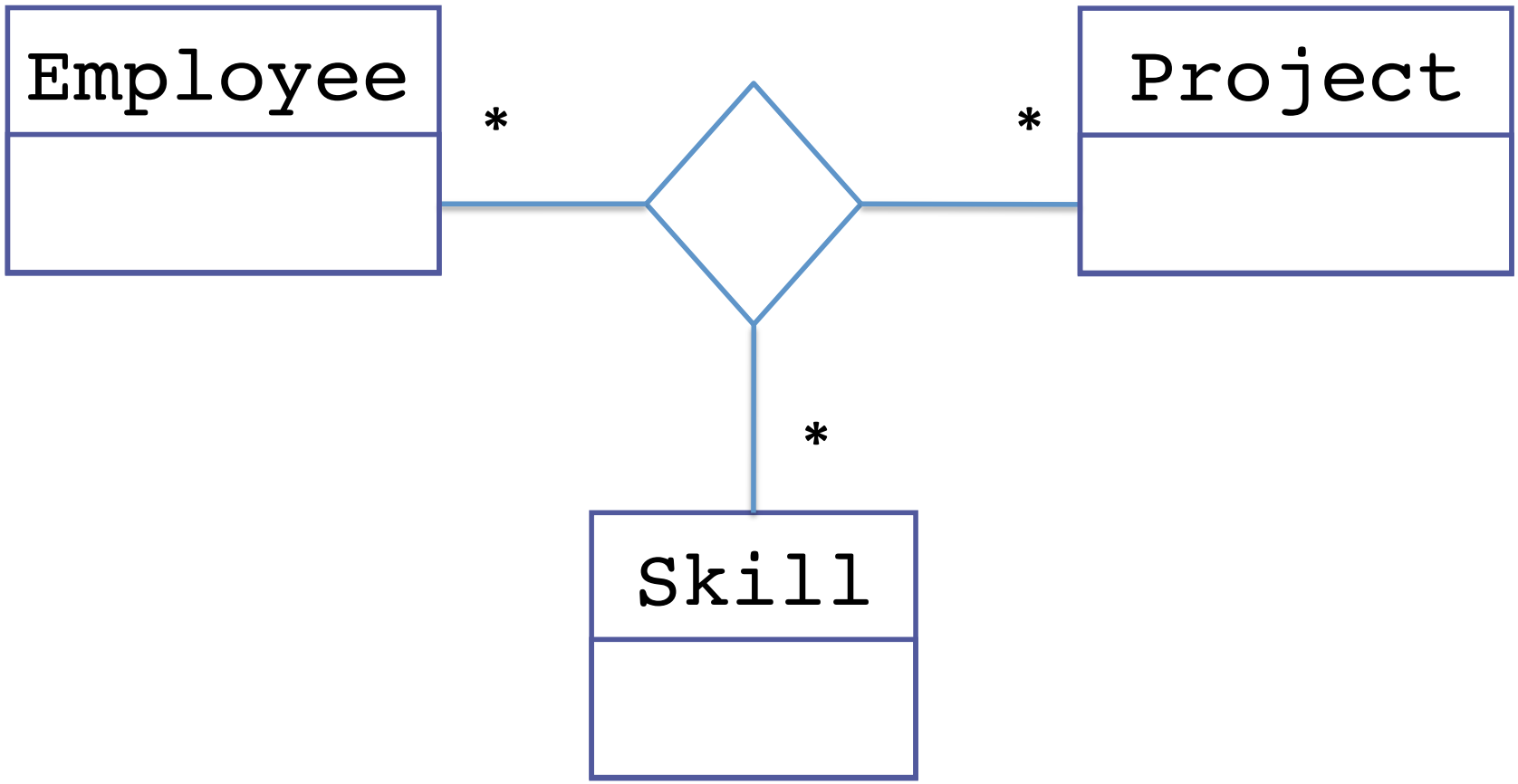
[UML] Prolegomenes_uml.pdf

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

Annexe – Les relations ternaires



Employee can participate to projects to which they bring some skills



What about the Cardinalities ?

[UML2] section [3.3.4]

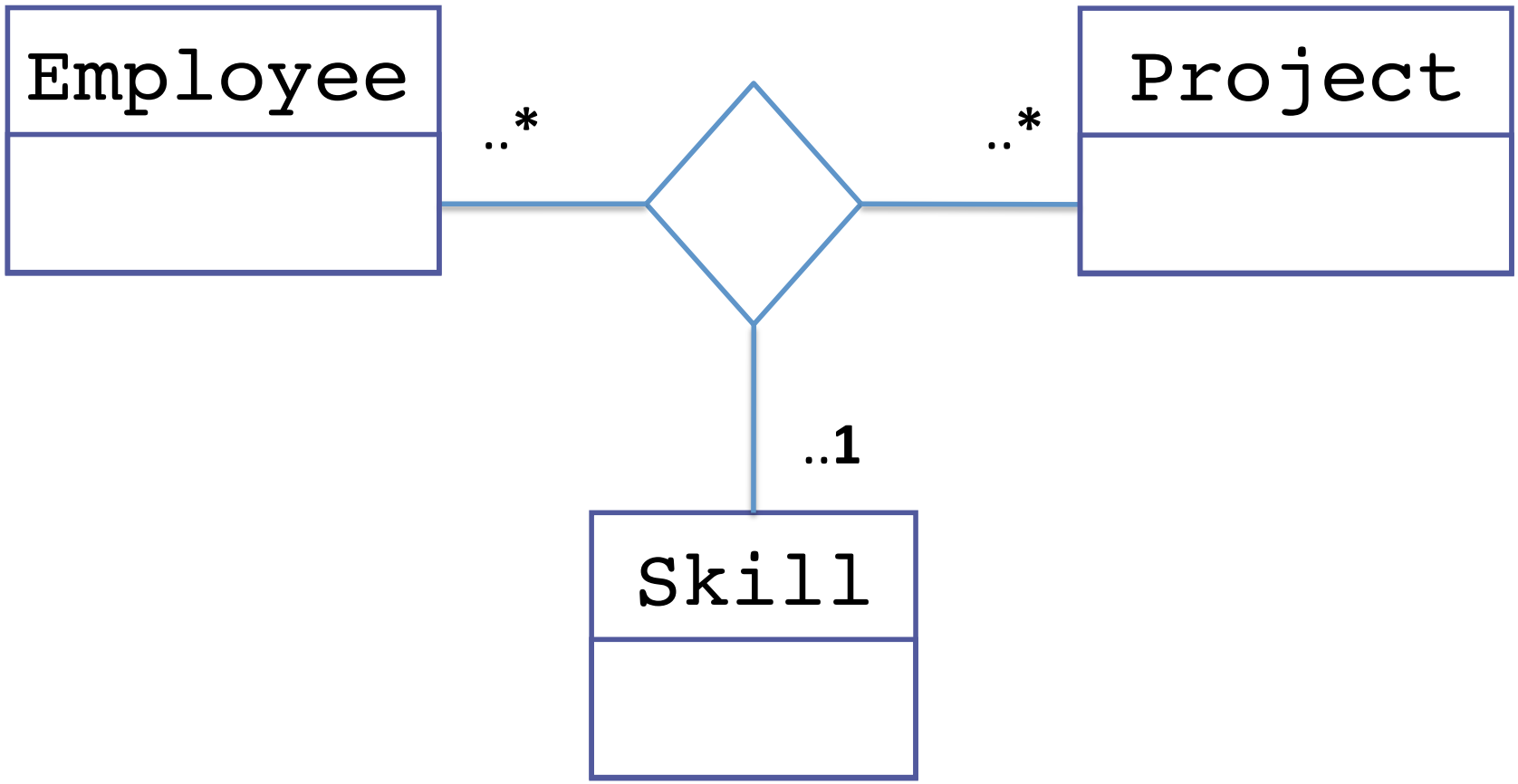
min .. max

What about the Cardinalities ?

[UML2] section [3.3.4]

.. **max**

An employee can participate in a project with **at most one** skill



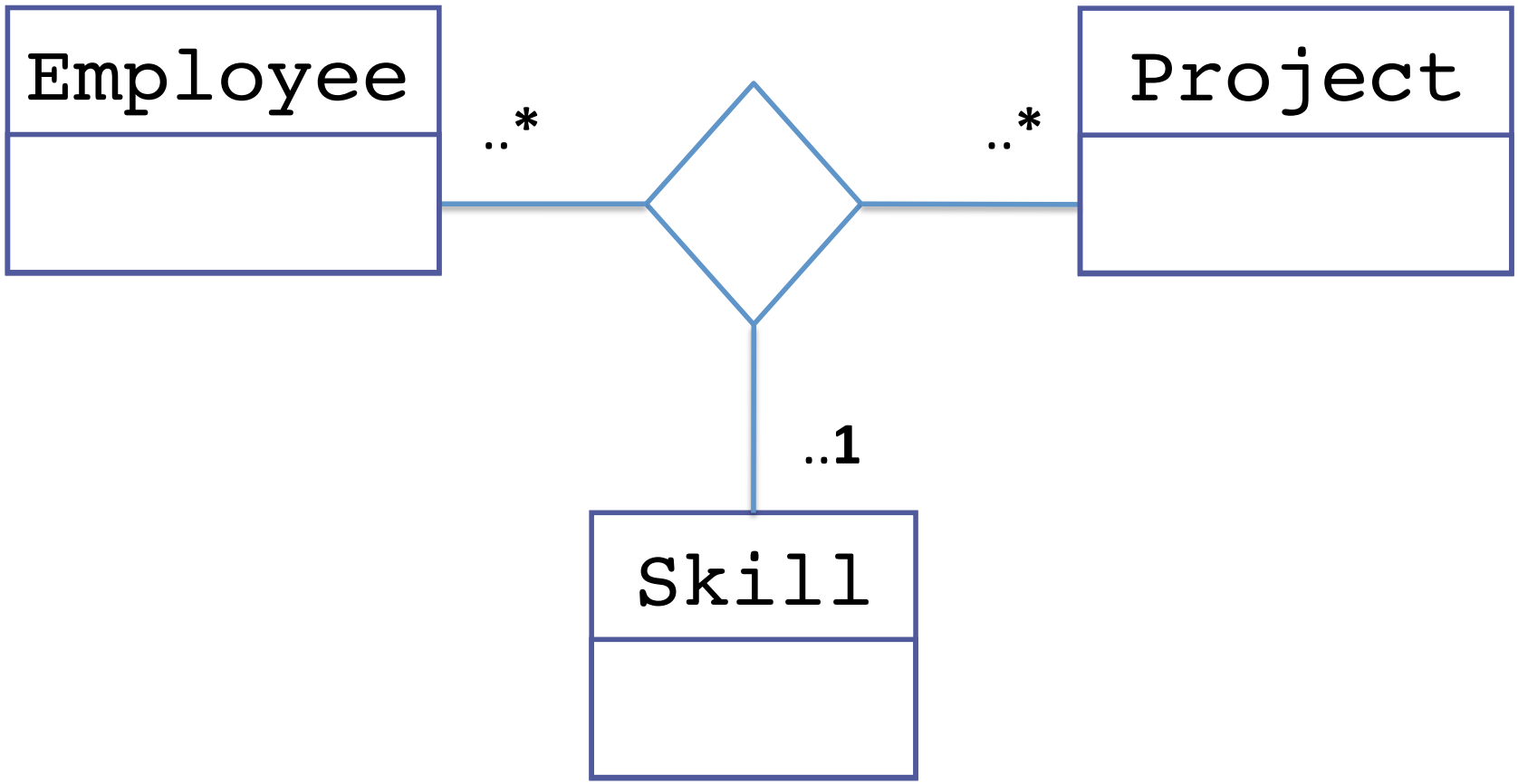
FD : Employee, Project → Skill

The **max**-multiplicity

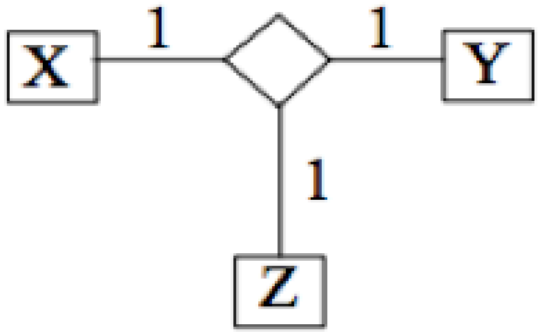
- Potential number of values at an end of the association, when the values at the other ends are fixed

- Association ends with max-multiplicity 1 are *functionally dependent* from all other ends.

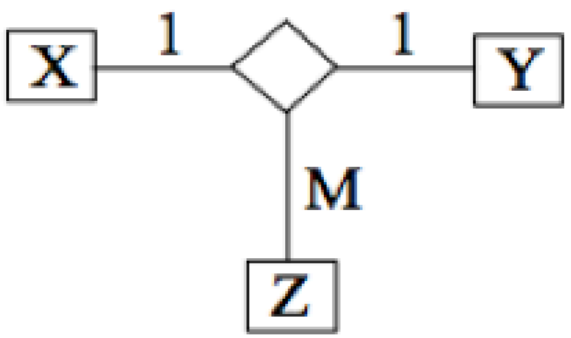
An employee can participate in a project with **at most one** skill



Particular cases of max-multiplicity



FD's: XY -> Z
XZ -> Y
YZ -> X



FD's: XZ -> Y
YZ -> X

Warning: different modelization is probably viable.

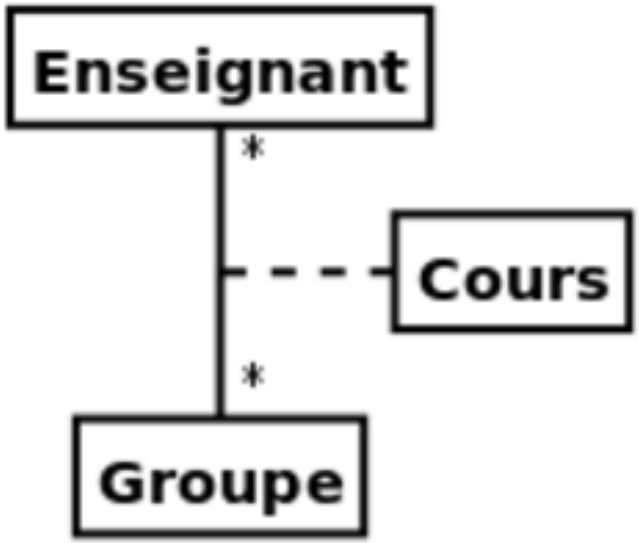
Almost-Equivalences between class-associations and ternary-relations

[UML2] section [3.3.7-e]

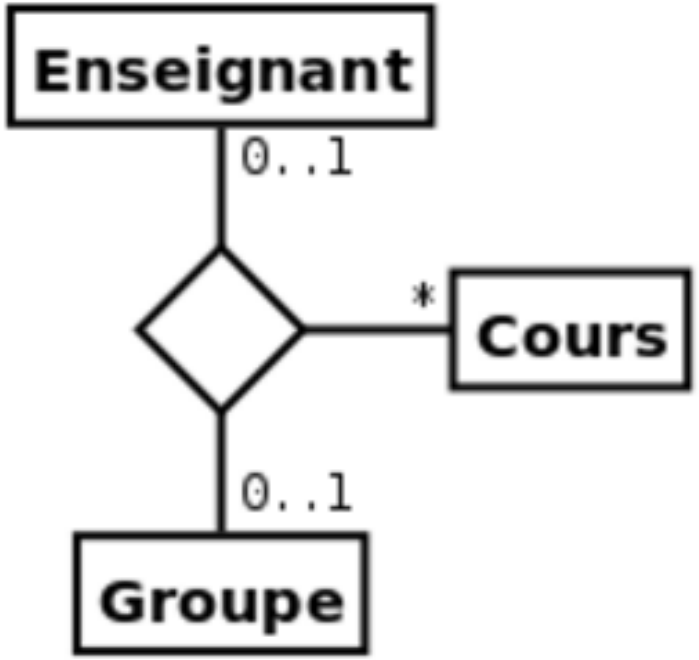
Figure 3.15 : Deux modélisations modélisant la même information.

Can a Cours have no participants ?

[UML2] section [3.3.7-f]



NO



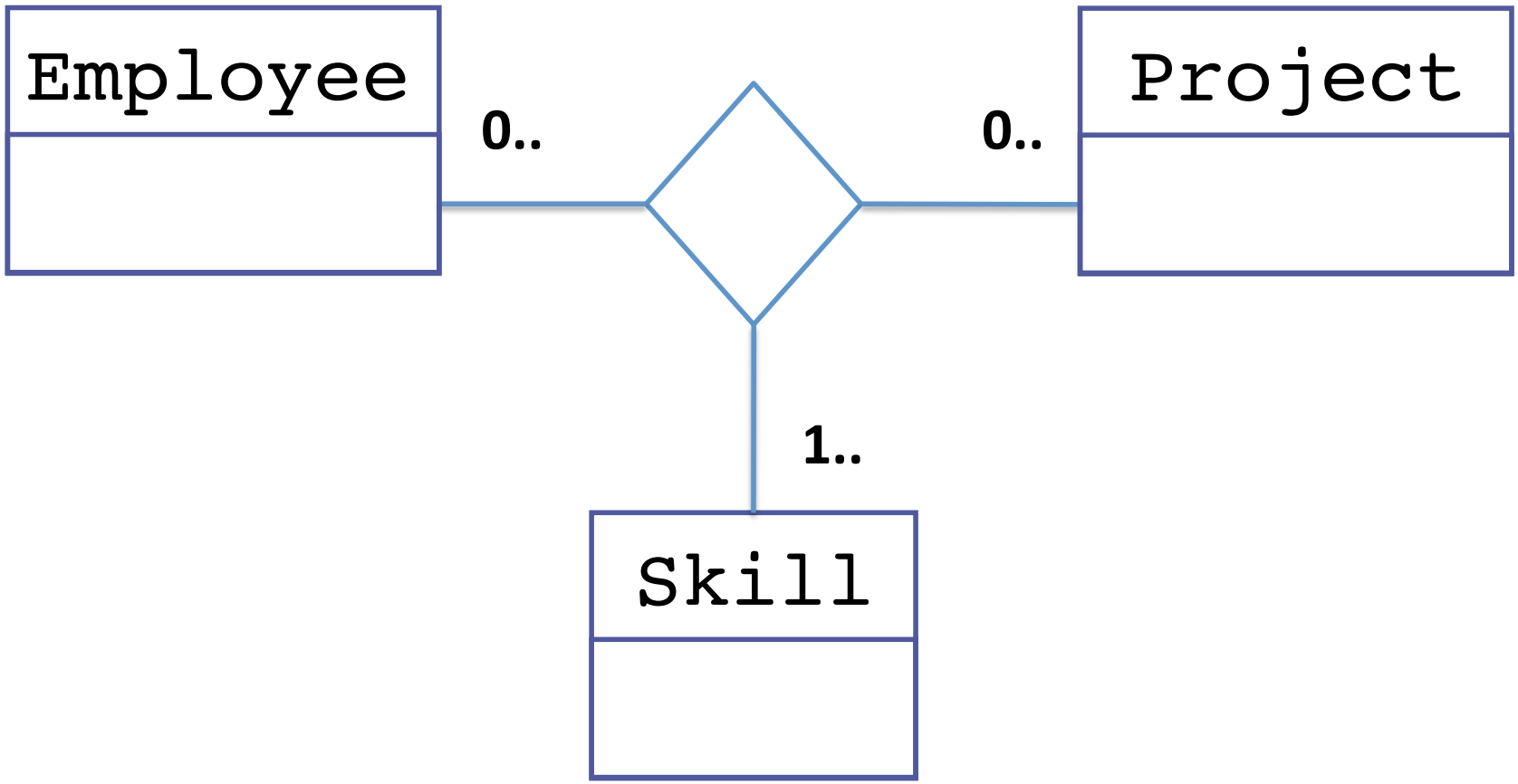
YES

What about the Cardinalities ?

[UML2] section [3.3.4]

min ..

Every employee must participate in **every** project with **every** skill

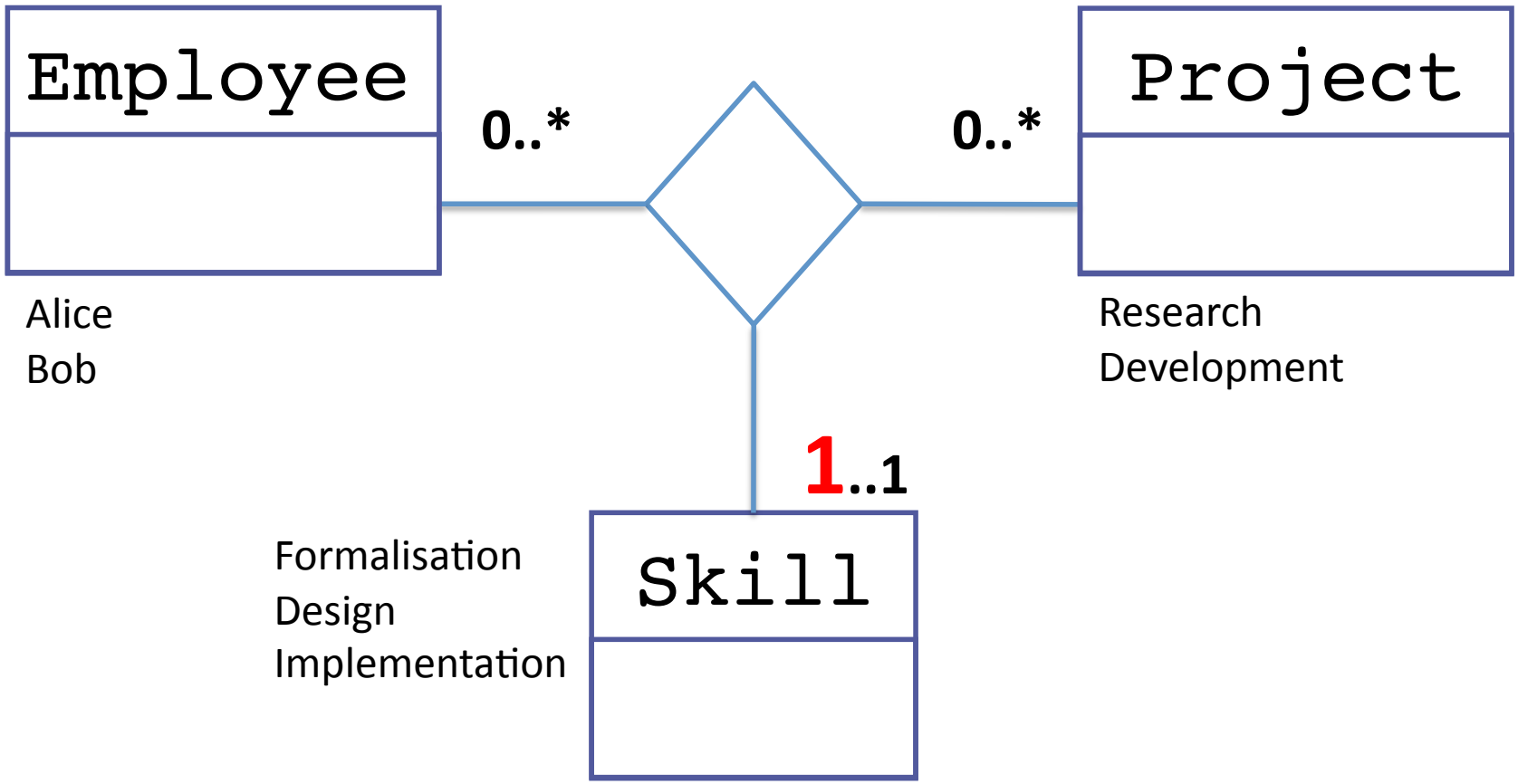


The **min**-multiplicity

- For n-ary associations, it is typically 0.

- If the **min**-multiplicity of the end of an n-ary association is 1
  - Then : one instance of the association **must exist** for **every possible combination** of the values at the the other ends.
  - Paradox called «Bouncing effect of the one»

Every employee must participate in **every** project with **every** skill



This is what is expected now from the association, because of **min**-mult. = 1

Alice	Research	Formalisation
Alice	Research	Design
Alice	Research	Implementation
Alice	Development	Formalisation
Alice	Development	Design
Alice	Development	Implementation
Bob	Research	Formalisation
Bob	Research	Design
Bob	Research	Implementation
Bob	Development	Formalisation
Bob	Development	Design
Bob	Development	Implementation