INF5180

Conception et exploitation d'une base de données

Zied Zaier, PhD
Département d'informatique
Université du Québec à Montréal

Cours 1

LE MODÈLE CONCEPTUEL DE DONNÉES

DANS CE DOCUMENT, LE CONTENU EST ADAPTÉ DE ROBERT GODIN - SYSTÈMES DE GESTION DE BASES DE DONNÉES PAR L'EXEMPLE – LOZE-DION (2012)

Sommaire

I. Processus de conception d'une base de données
II. Planification
III. Analyse: Modèle conceptuel de données
IV. Diagramme de Classes UML

Concepts de base

- Processus de Conception d'une BD:
  1. planification
  2. analyse: modèle conceptuel de données
  3. diagramme de classe UML
  4. Modèle entité-association

Processus de conception de BD

- Le cycle de vie d'un système d'information:
  - présente un cadre conceptuel permettant d'organiser le processus de développement d'un système d'information
  - décompose le processus en sous processus plus simple
  - Cas des base de données
  - Représentation graphique très privilégiées

Processus de conception de BD

1. étude d'opportunité de développer un SI, en fonction des besoins de l'organisation

document des exigences logicielles - spécification de haut niveau, analyse coûts/bénéfices, risques, échéancier

2. Produit une spécification détaillée du système à développer

Indépendant des technologies

3. Solution informatique au problème

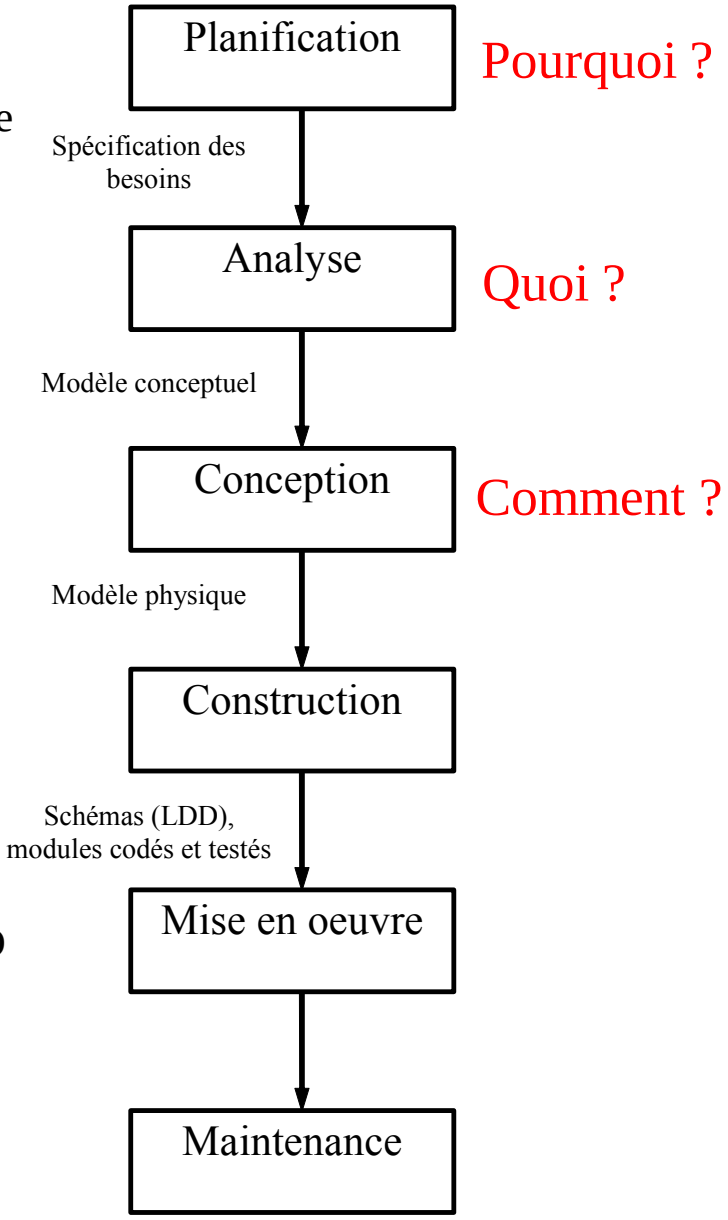
Considérer les technologies

4. Élabore le code des schémas de la BD

Instancier la BD

5. Configuration des différents paramètres du SGBD

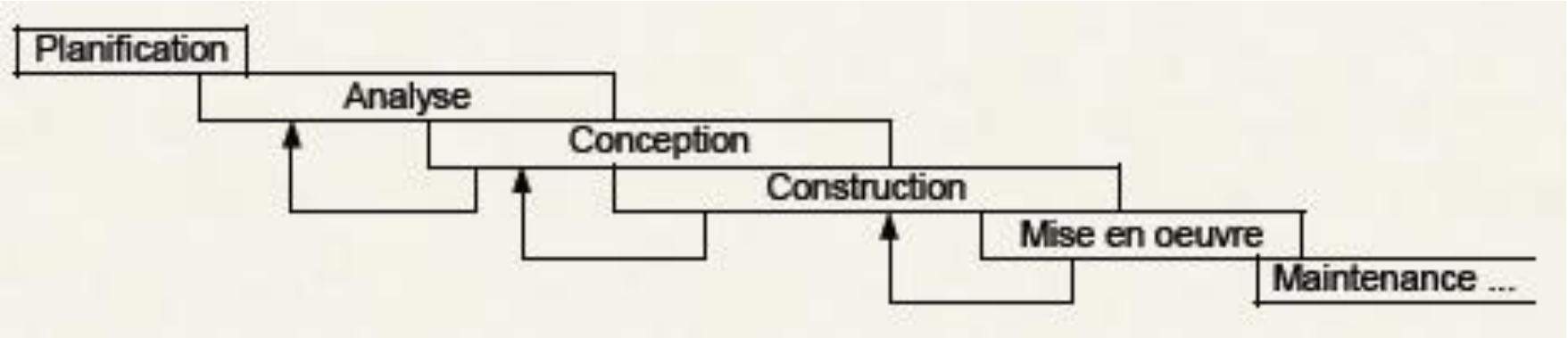
6.Performances: surveillance du système, correction des erreurs, etc.



Processus de développement

- *Cycle de vie en cascade*
- *Cycle de vie itératif*
- *…*

Livraison en phases



Planification

- Pourquoi développer un système ?
- *Étude d'opportunité*
  - risques
  - coûts
  - bénéfices
- Résultat: *Document des exigences logicielles*
  - spécification de haut niveau du système
  - *diagramme de contexte*
    - UML : diagramme des cas d ’utilisation

Entités

- Entité = “chose” ou objet
- Ensemble d’entités = collection d’entités similaires
  - Comparable à une classe dans le modèle orienté objet
- Attribut = propriété d’un ensemble d’entités
  - En général, toutes les entités d’un même ensemble possèdent les mêmes propriétés
  - Défini par des types de base (ex: entier, caractère)

Étude de cas : *SyLeRat*

- Développement d'un système d’information pour la bibliothèque *LeRat*
  - gestion des collections
  - service de prêt
  - suivi des retards
  - service de repérage documentaire
  - alimenté par *SystèmeAcquisitions (gestion des nouvelles acquisitions)*

Acteurs et cas d'utilisation

- *Approche pour spécifier ce que le système doit faire:*
  - *Méthode des Cas d'utilisation* (*use cases*) Jacobson (92)
    - Décrit l'interface au système d'un point de vue de son utilisation par les acteurs
  - *Acteur*
    - entité externe qui interagit avec le système
    - représente une catégorie d'utilisateurs et non pas un utilisateur physique; aussi pas juste un humain
    - acteur primaire, acteur secondaire

Diagramme de contexte de *SyLeRat en UML*

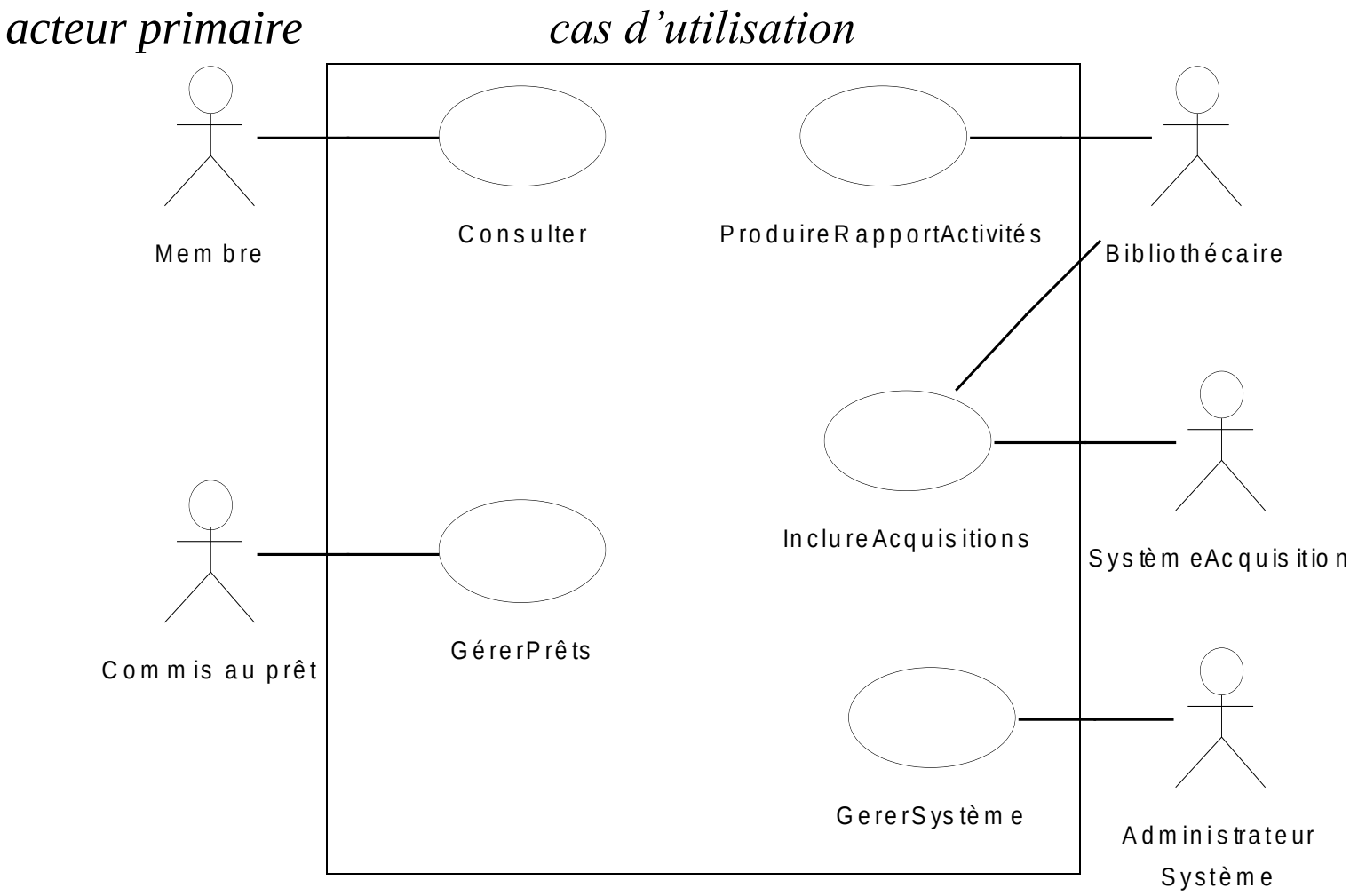


Diagramme de contexte de SyLeRat

- Fréquemment utilisé dans le document des exigences logicielles
- Documentation complémentaire possible des cas d’utilisation

Documentation d'accompagnement pour le cas d'utilisation *GérerPrêt*

Nom: GérerPrêts
Description courte : Gérer les prêts.
Type: Interactif
Description: Ce cas d'utilisation est déclenché par le commis au prêt suite à une requête d'un membre ou d'un employé. Il lui permet d'enregistrer un prêt ou un retour, de consulter les prêts, de gérer les données d'identification des membres, i.e. l'identificateur d'utilisateur de sa carte de membre, le mot de passe du membre, son nom, prénom et le numéro de téléphone de sa résidence. Il permet aussi de produire un rapport des retards. Lors d'un prêt ou d'un retour, l'identificateur d'utilisateur et l'identificateur de l'exemplaire peuvent être saisis en utilisant un lecteur optique ou manuellement.
Règles du domaine d'application:

1. La durée maximale d'un prêt est fixée à 7 jours pour un membre.
2. Le nombre maximal d'emprunts est fixé à cinq pour un membre.
3. Il est interdit d'effectuer un prêt lorsqu'un membre a un retard.
4. Les contraintes précédentes ne s'appliquent pas aux emprunts effectués par les employés.

Exigence de performance : Le temps d'attente de la validation de l'identificateur de l'utilisateur et de la vérification des conditions requises pour un emprunt doit être inférieur à 1 seconde.

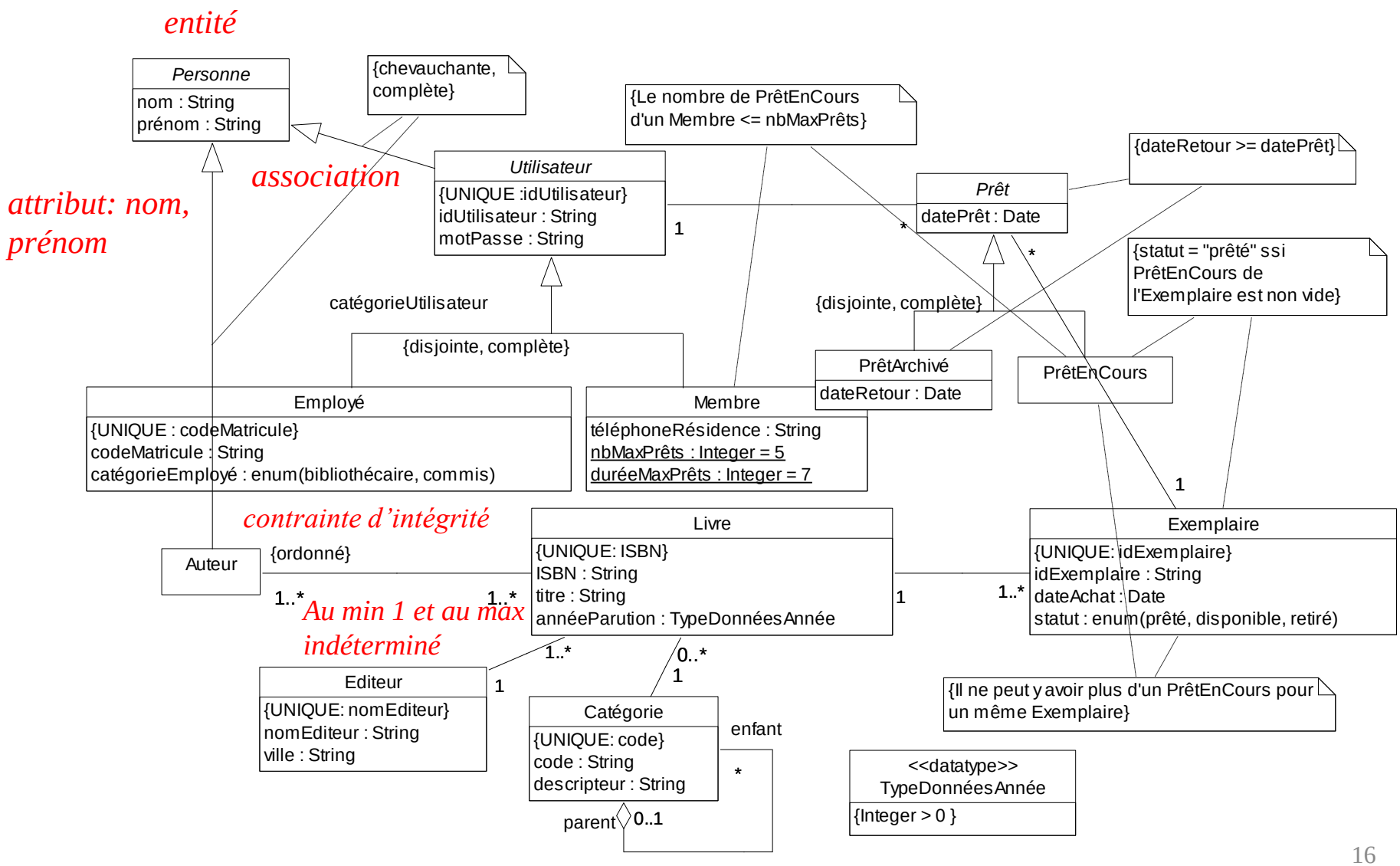
Exigence de sécurité : Le commis doit être autorisé à l'aide de son identificateur d'utilisateur et de son mot de passe.

Analyse : modèle conceptuel de données

- *Modèle conceptuel de données* :
  - représentation abstraite des informations à placer dans la base de données qui est indépendante de la technologie utilisée pour l'implémentation

- ~Données persistantes du *Platform Independent Model* (PIM) de *Model Driven Architecture* (MDA) de l'OMG

Exemple de MC pour SyLeRat



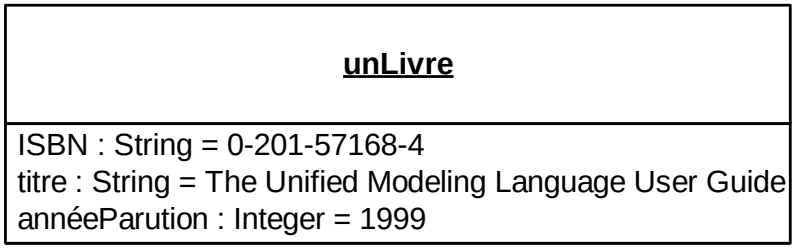
Représentation du modèle conceptuel

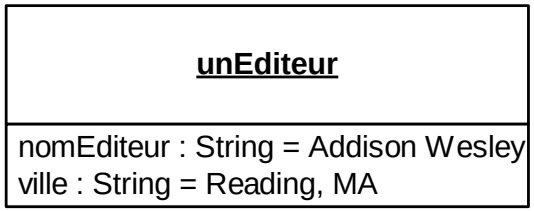
- Formalisme entité/association (Chen, 76)
  - diverses extensions
- Modèles sémantiques
  - graphes conceptuels (Sowa), SDM, ...
- UML
  - ~ *entité/association++*
  - *diagramme de structure statique* (*diagrammes de classes*)

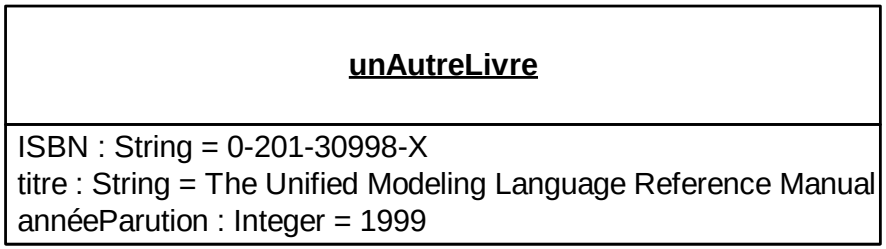
Modélisation Objet: Notion d'objet et de classe

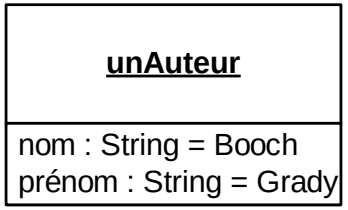
- ***Objet (instance d'une classe)***
  - significatif pour le domaine d'application
  - caractérisé par
    - *identité*
    - *état*
    - *comportement*
- ***Attribut* (*variable membre, variable d'instance*)**
  - contenant pour une valeur
  - Les valeurs des attributs d’un objet représentent *son état*

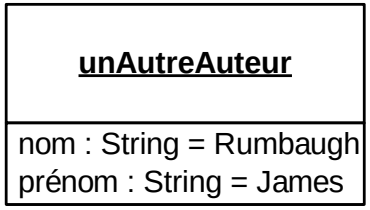
Représentation d ’un objet en UML









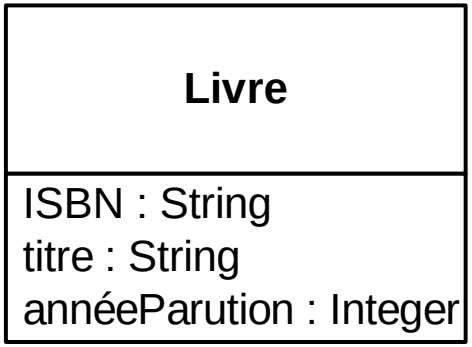


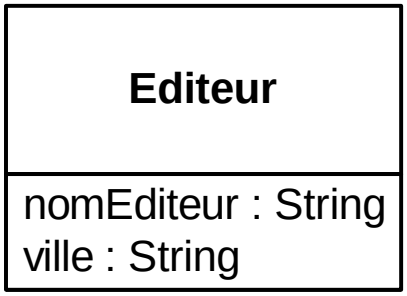
*Le MC vise a représenter les **classes** d’objets.*

Classe

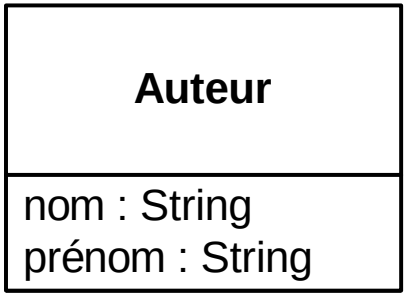
- Abstraction
- Regroupe les caractéristiques communes à un ensemble d'objets
  - attributs
  - associations
  - opérations (non considérées dans un premier temps)

Représentation d ’une classe en UML





Abstraction correspondant à la structure commune à l’ensemble des livres



Le nom de classe n’est pas souligné

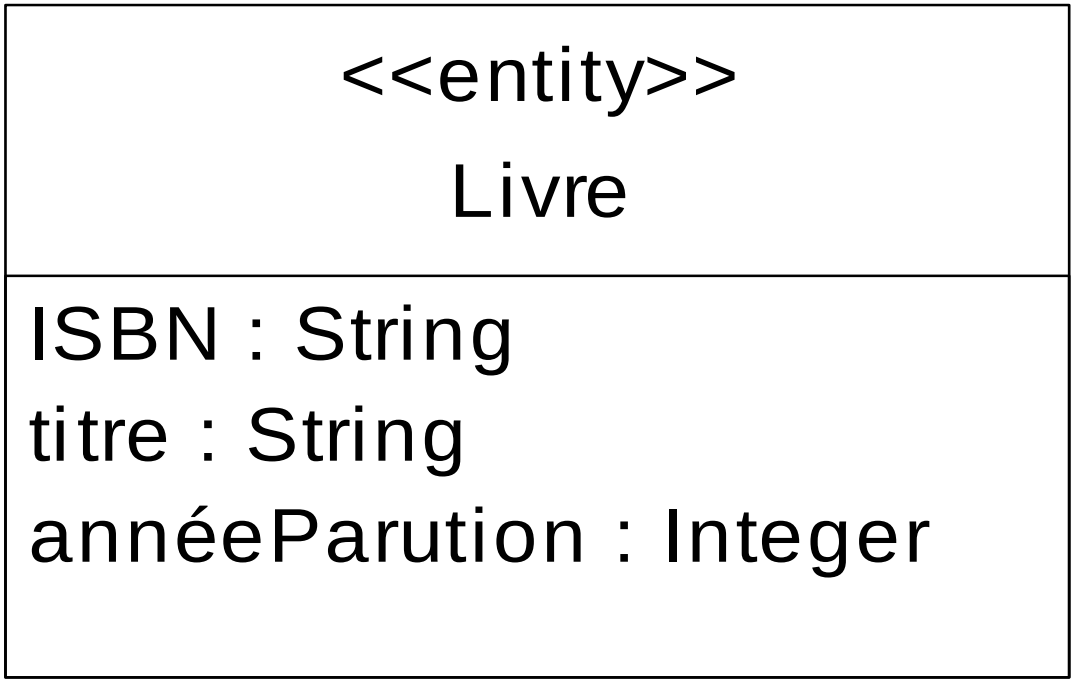
Classe : ‘contenant + contenu’

- ***Intention* (*intent*) d'une classe**
  - propriétés communes (attributs, associations et opérations)
- ***Extension* (*extent*) d'une classe**
  - ensemble des objets correspondant à la classe
    - extension représentée par un objet ?
- *Classe = intention + extension*

Terminologie

- *Objet*
  - *instance*, *occurrence*, *entité*
- *Classe à l'étape d'analyse*
  - abstraction
  - pas toujours une classe d 'implémentation
  - *concept*, *entité*, *type* (*stéréotype UML*)
  - stéréotype « *entity* » pour données persistantes du domaine d 'application
    - acteur est une classe UML de stéréotype « actor »
  - valeur étiqueté {*persistent*}

Stéréotype UML

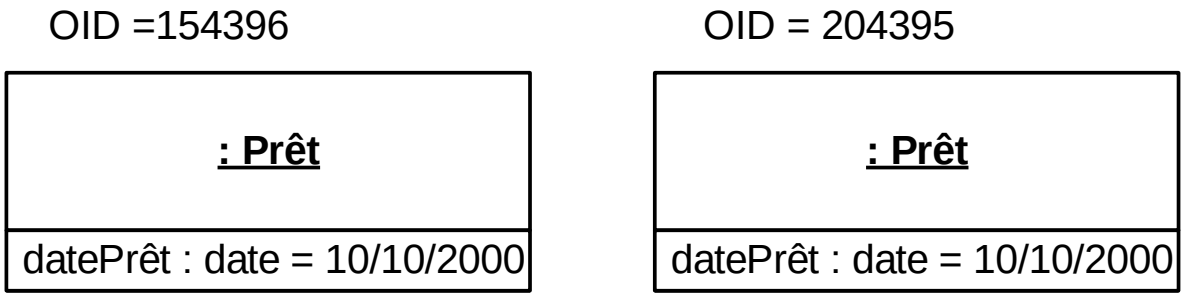


Identifiant d'objet (OID, *object identifier*)

- *Identificateur unique implicite associé à un objet*
- Mécanisme d ’identification
  - pas deux objets avec le même OID
- Implicite
  - non visible
  - réalisation traitée à la conception
- Mécanisme de référence
- Traitement de ce mécanisme d’objet à l’étape *conception*

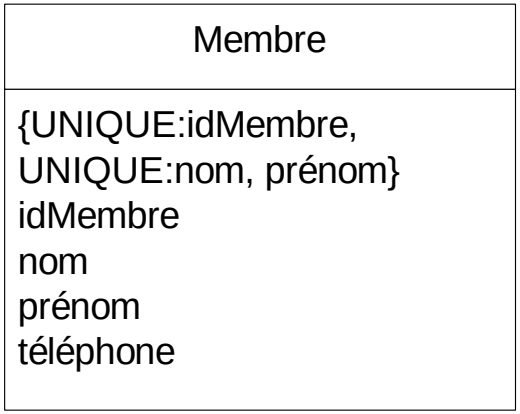
Pas besoin d ’identificateur explicite !

- Par opposition au relationnel



Identifiant naturel (ou *clé* «*key*») pour une classe

- Ensemble d'attributs minimal qui identifie chacun des objets de manière unique
  - ~clé candidate du relationnel
- Représentation par une contrainte UML ({…})



Syntaxe générale pour la spécification des attributs en UML

- [*visibilité*] *nom* [*multiplicité*] [*: type*] [*= valeurInitiale*] [*{propriétés}*
  - *visibilité* peut être :
    - \+ publique
    - # protégé
    - \- privé
  - *nom* de l'attribut
  - *multiplicité* ([1..1] par défaut) : contrainte sur la cardinalité
    - ✓ *téléphone*[1..2]: *String*
    - ✓ *adresse* [0..1]: *String*
    - ✓ *auteurs* [1..*]: *String*

permet un Min 1 et un MAX indéterminé d'auteurs

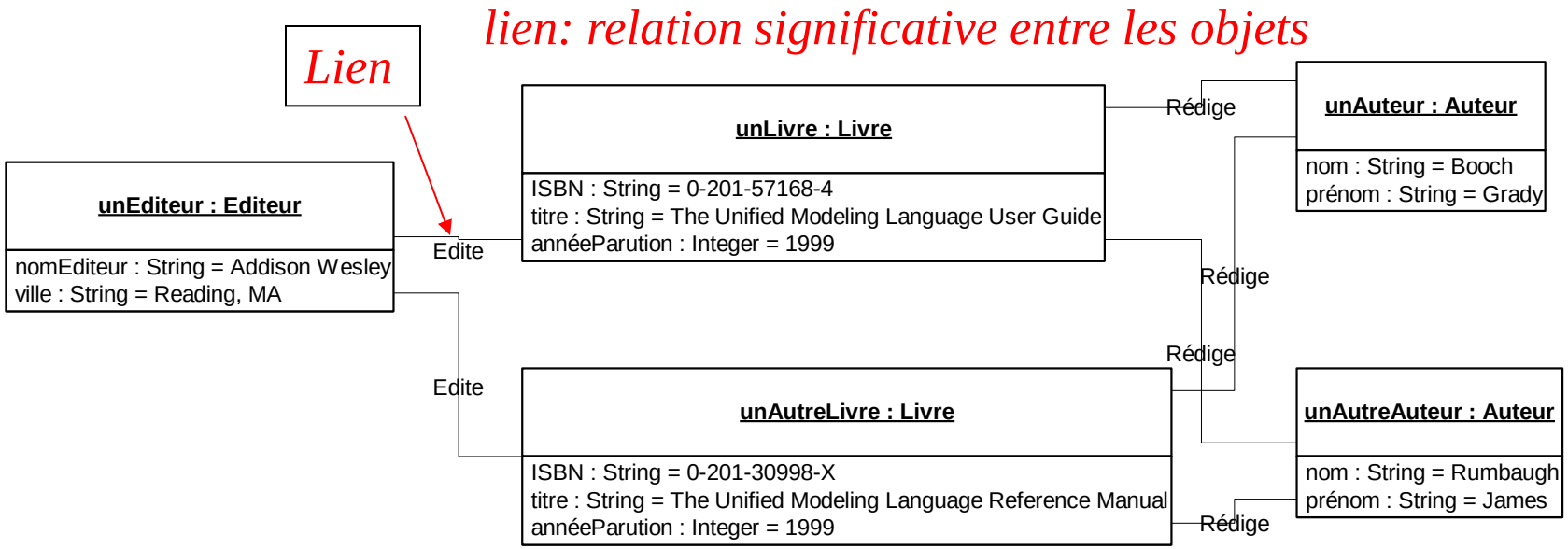
Syntaxe pour attributs (suite)

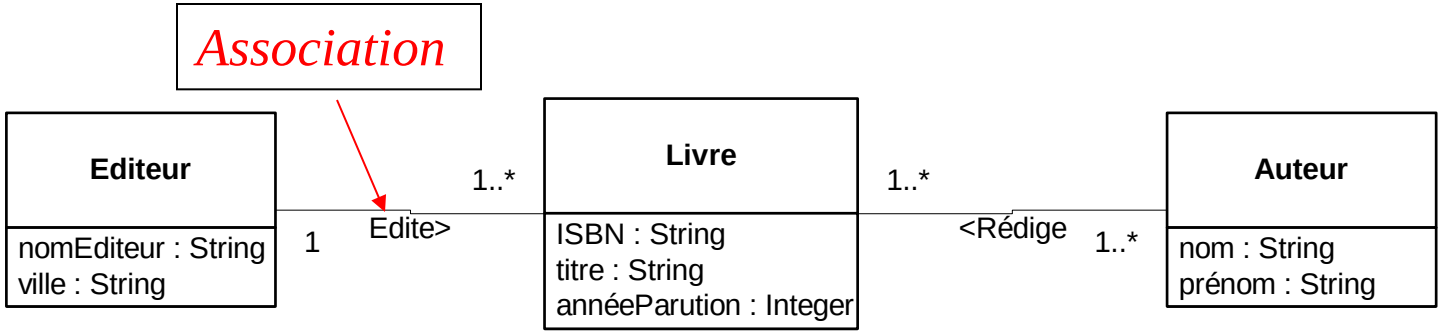
- [*visibilité*] *nom* [*multiplicité*] [*: type*] [*= valeurInitiale*] [*{propriétés}*]
  - *type*
    - *OCL ('Object Constraint Language') → UML*
      - *Boolean*, *Integer*, *Real*, *String*, *enum*{*valeur*1,…, *valeur*n}
    - types de la plate-forme visée
      - *Types de JAVA, C++, SQL,..*
    - type non pré-défini
      - classe de stéréotype «*datatype*»
    - *Type de données~domaine* en modélisation conceptuelle

Syntaxe pour attributs (suite)

- [*visibilité*] *nom* [*multiplicité*] [*: type*] [*= valeurInitiale*] [*{propriétés}*]
  - *valeurInitiale*
    - à la création de l ’objet
  - *propriétés*
    - Prédéfinies sur l’*attribut* :
      - *Modifiable - changeable* (par défaut)
      - *Fixe - frozen*
      - *insertionSeulement – addOnly* (pour multiplicité >1)
  - *portée* (distinguer l’attribut de classe de celui de l’objet)
    - souligner attribut de classe (Rational Rose :$ avant l’attribut de classe)

Notion de lien et d'association binaire



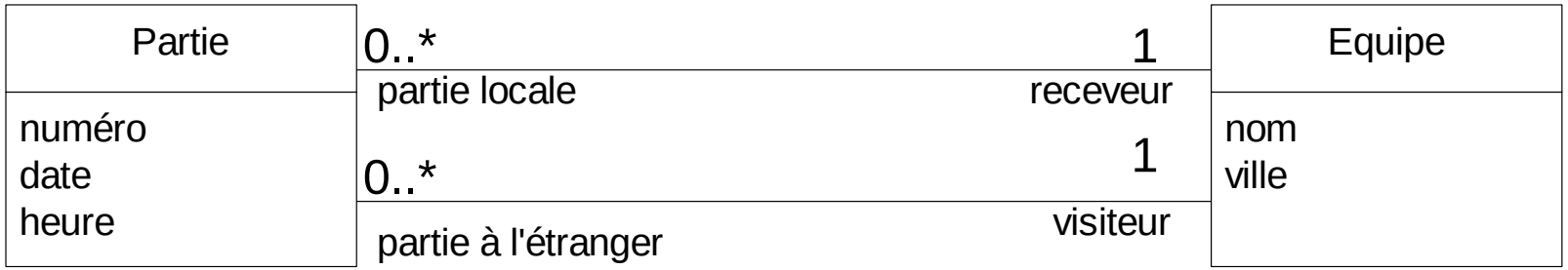


Association entre classes: abstraction qui correspond à un ensemble de liens qui ont une sémantique commune

Rôles et multiplicités

- *Nom de rôle*

Rôle: une des extrémités de l'association

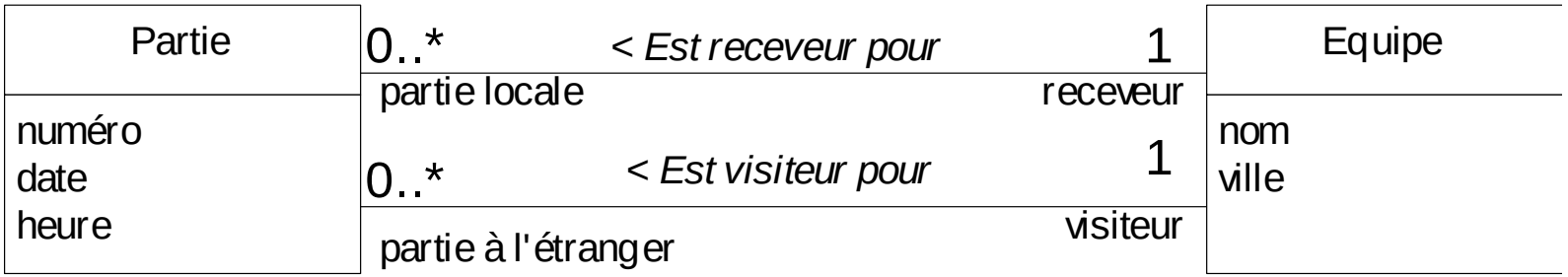


deux associations différentes

- Notation de multiplicité
  - 1..1, 1
  - 0..1
  - 0..*, *
  - 1..*
  - n
  - n..m
  - Liste

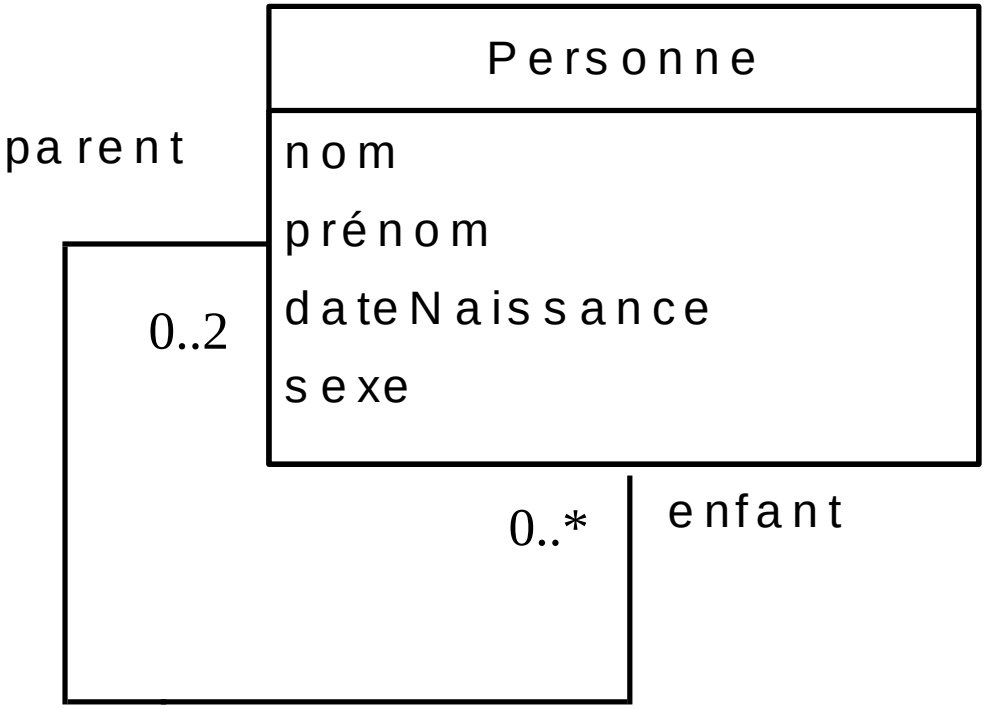
Rôles et multiplicités

- Exemple avec nom de rôle et d ’association



deux associations différentes

Association réflexive

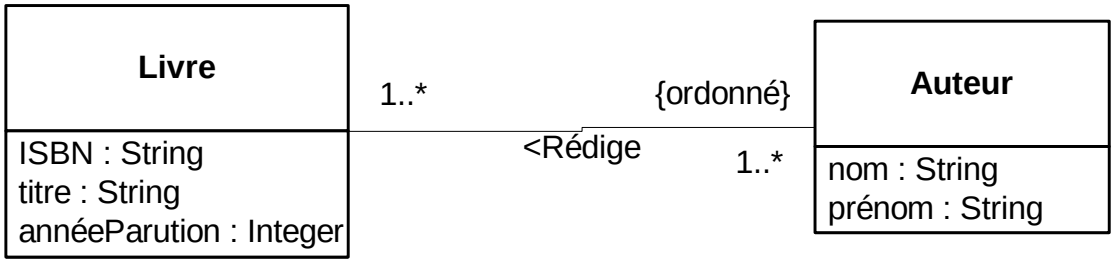


Une personne a exactement 0..2 parents
peut avoir un nombre quelconque (dont 0) d'enfants

Contraintes pré-définies pour les associations

- *Ordonné* (*ordered*)

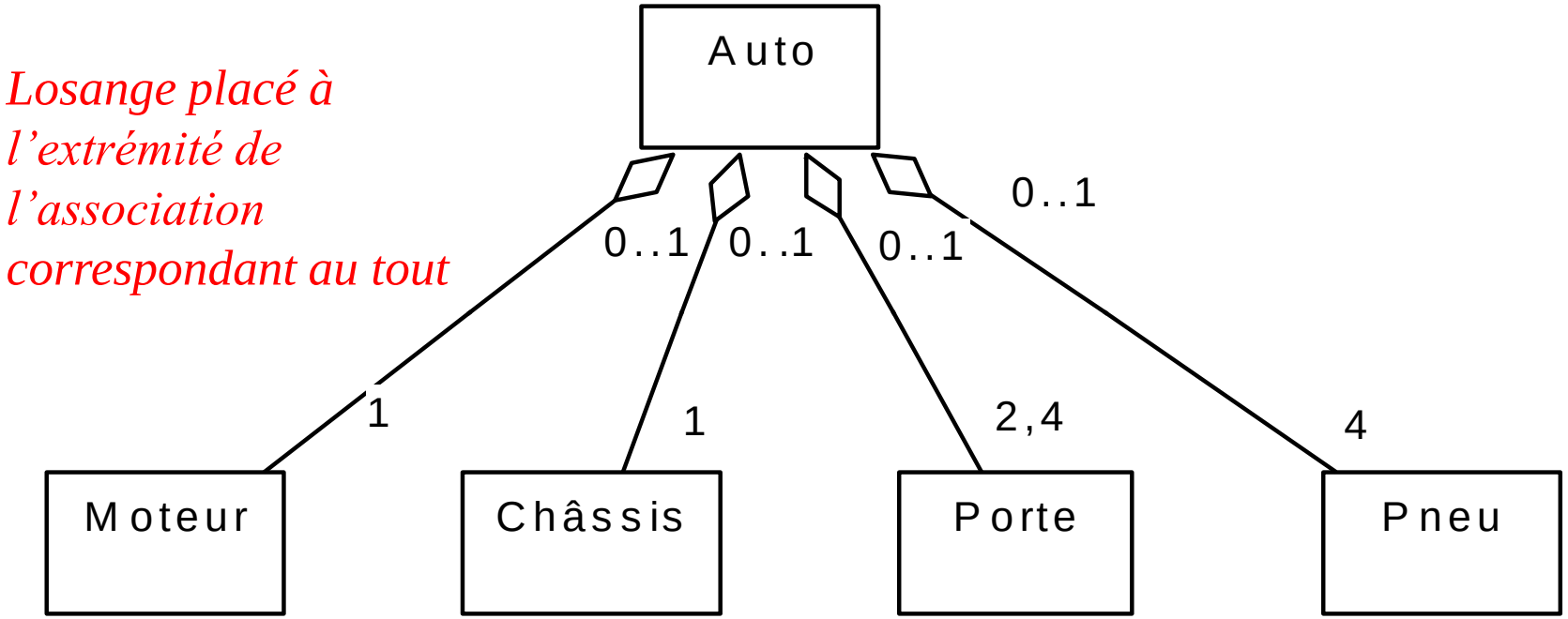
Les auteurs d'un livre sont ordonnés



- *Modifiable* (*changeable*)
- *InsertionSeulement* (*addOnly*)
- *Fixe* (*frozen*)
- *Exclusives*
  - entre deux associations ou plus
  - un objet ne participe qu'à une seule des associations; ex: étudiant sous-gradué - gradué

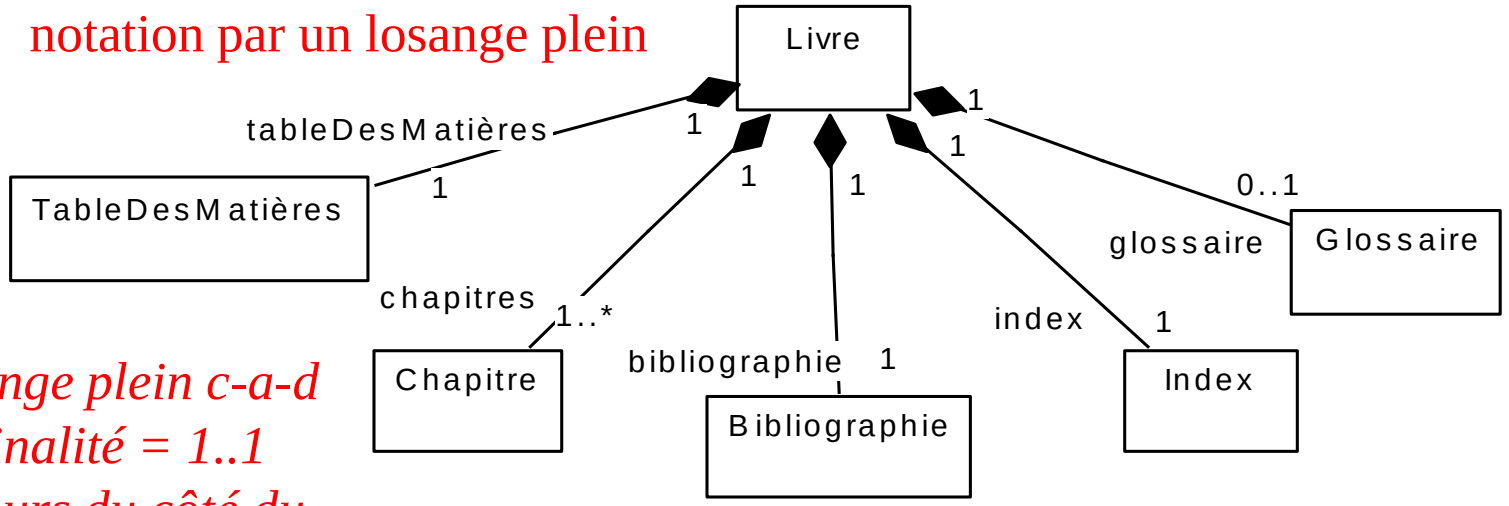
Agrégation

- Cas particulier d ’association
  - Association binaire entre *un tout et ses parties*



Composition

notation par un losange plein



Losange plein c-a-d cardinalité = 1..1 toujours du côté du composant

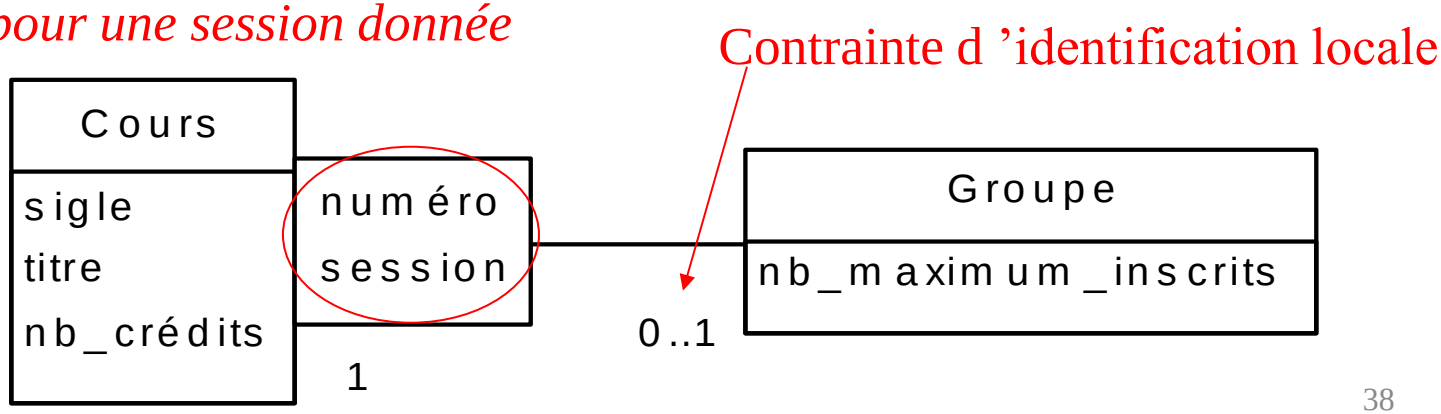
Livre
tableDesMatières : TableDesMatières chapitres[1..*] : Chapitre bibliographie : Bibliographie index : Index glossaire[0..1] : Glossaire

Associations qualifiées

- Permet de:
  - préciser comment L'ensemble des objets liés à un objet particlulier est partitionné par rapport aux valeurs d'un ensemble d'attributs
  - représenter une contrainte d'identification locale au contexte d'une association.

Précise comment un groupe est relié à un cours : *par un numéro et une session*

un Groupe est identifié par un numéro de groupe unique pour une session donnée

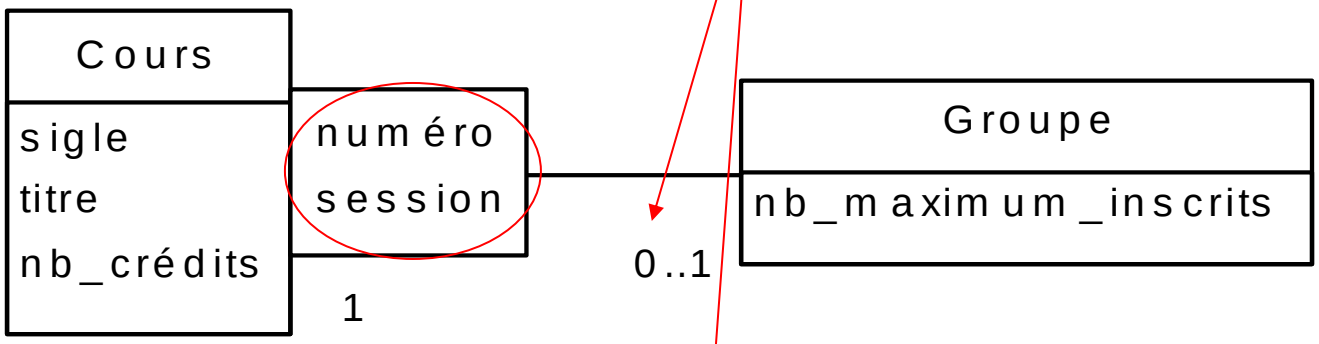


Associations qualifiées

- Partition des objets associés par rapport aux valeurs d'un ensemble d'attributs

Contraintes d'identification locale

Représentation avec qualification



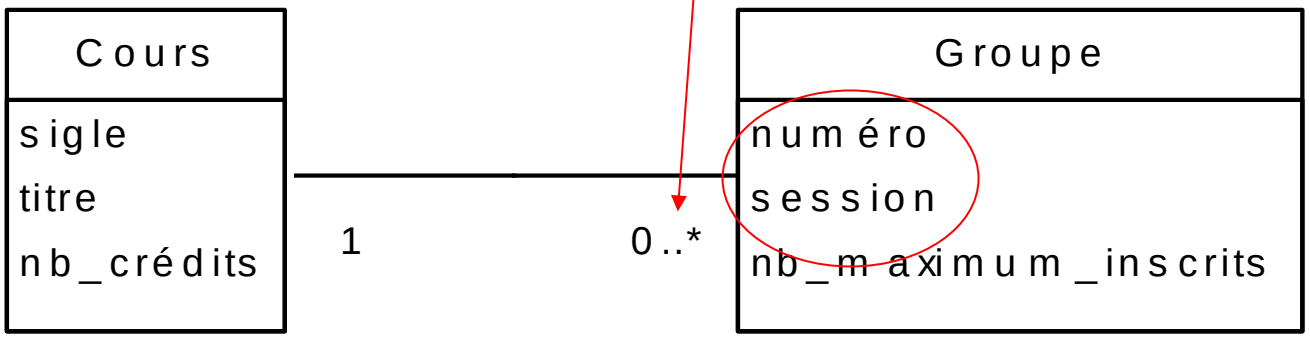
Un Groupe es lié à un Cours, pour un numéro de groupe et une session donnée

Groupe est identifié par un numéro de groupe unique pour une session donnée

- Précise comment un groupe est relié à un cours : par un numéro et une session

Représentation sans qualification

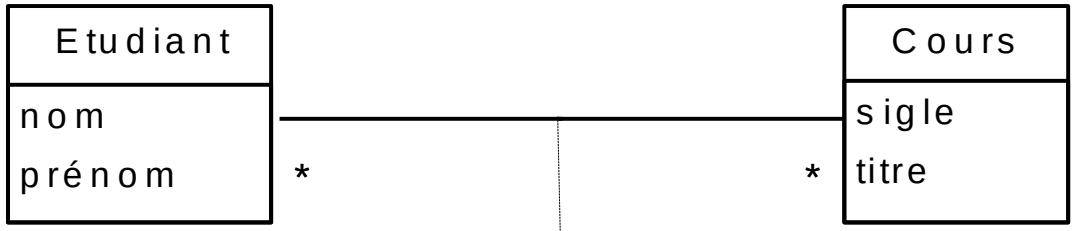
{UNIQUE : Cours, numéro, session}



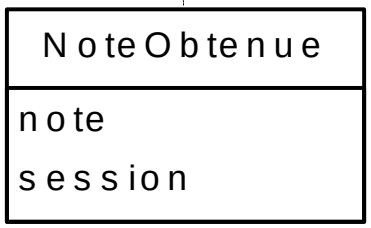
une contrainte UML est ajoutée

Classes associatives

- Données spécifiques à l ’association



Incorrect si plusieurs notes pour un *Etudiant* et un *Cours*



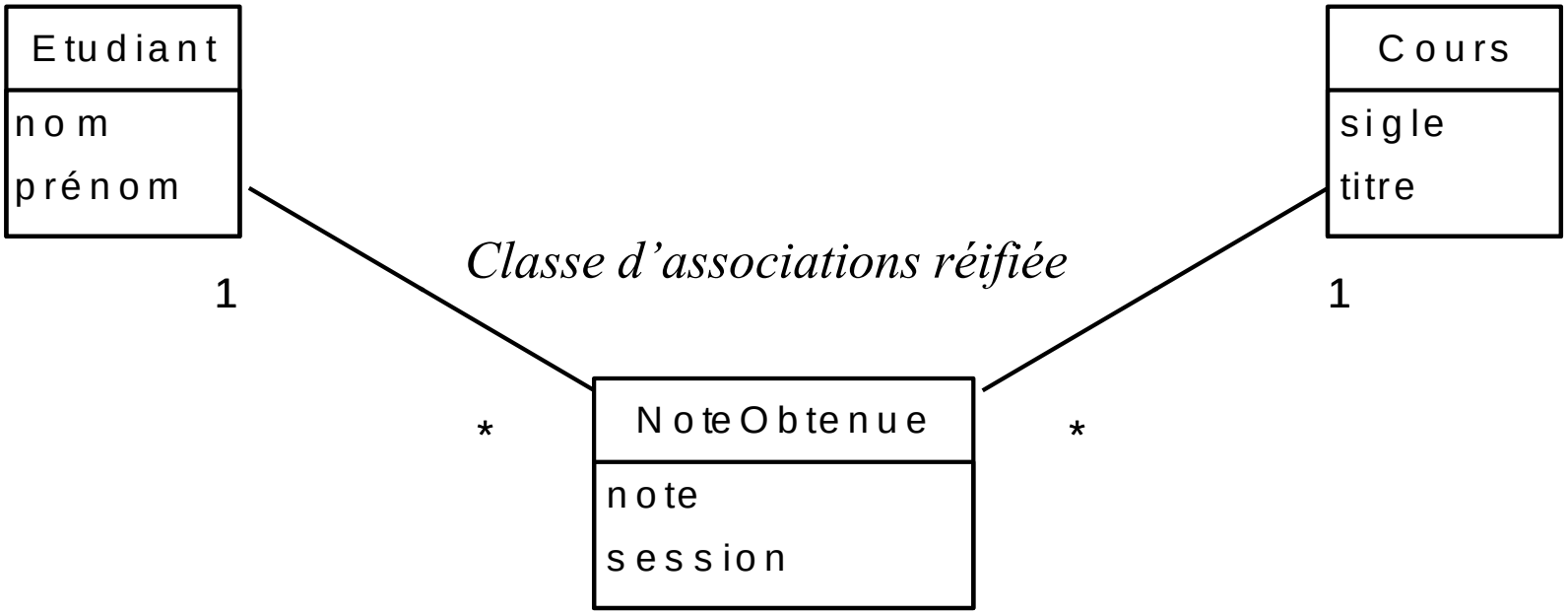
classe associative

un et un seul objet de la classe associative pour chaque lien de l’association

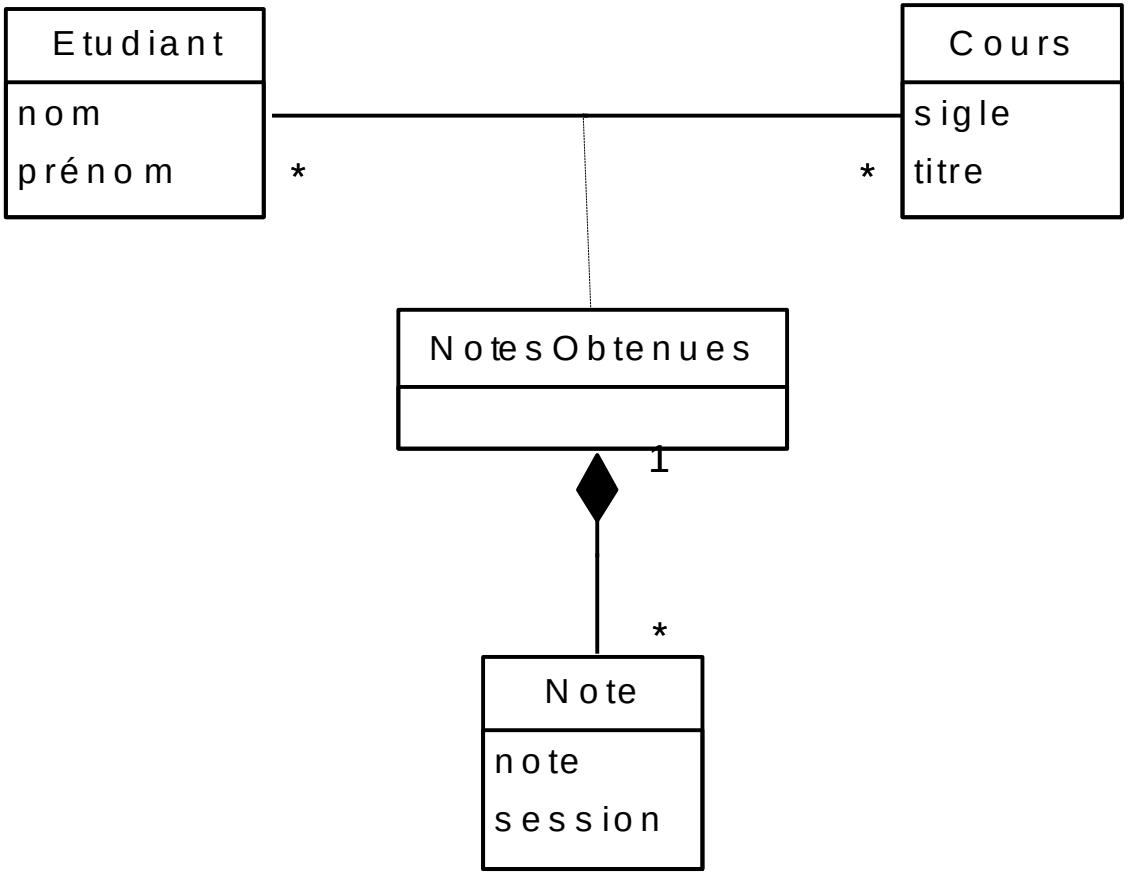
données caractéristiques au lien entre un Etudiant et le Cours suivis

Réification de l ’association

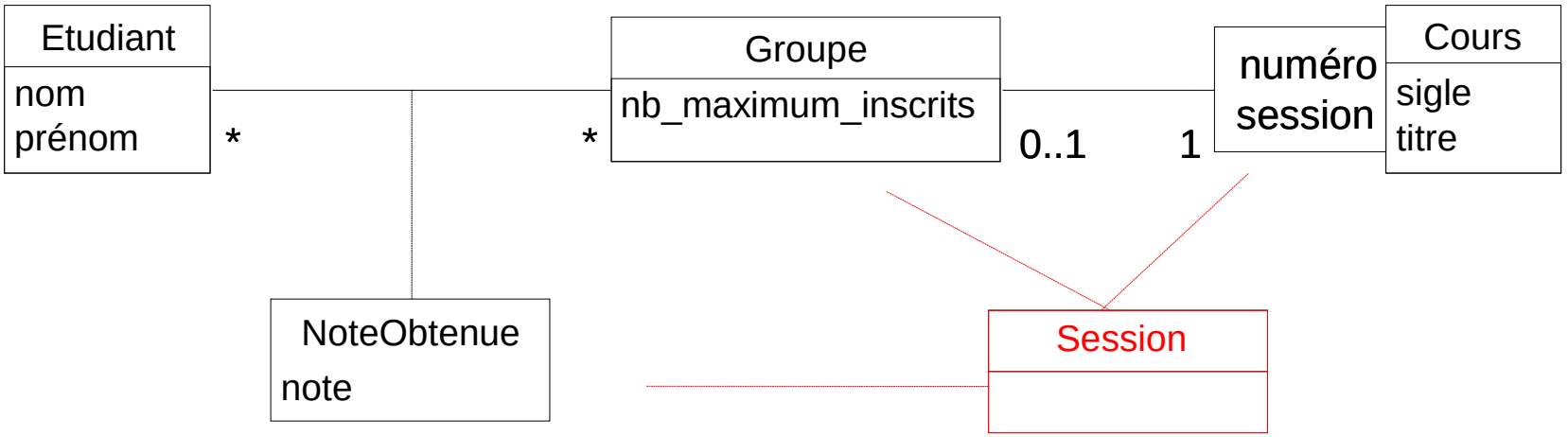
- Plusieurs notes pour un *Etudiant* et un *Cours*



Autre solution : classe associative + agrégation



Solution avec classe *Groupe*



Créer une classe Session ?

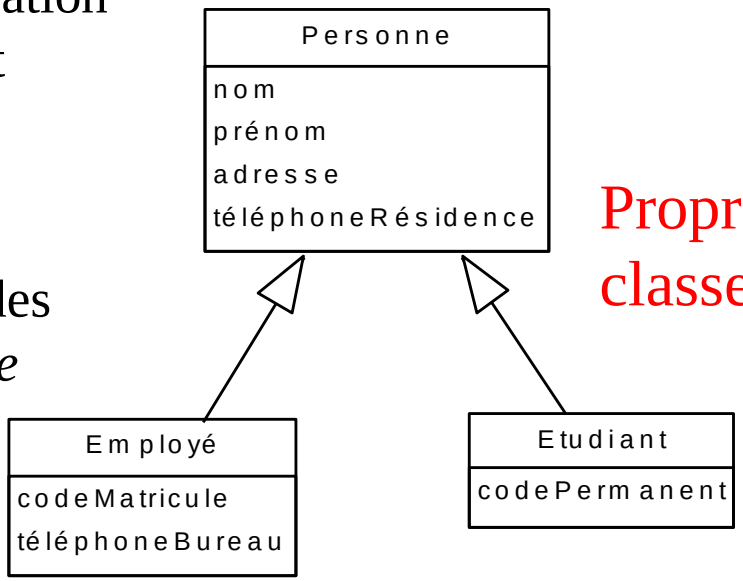
La généralisation/ spécialisation

- Généralisation
  - permet de faire ressortir les propriétés communes (attributs, associations, opérations) et les différences entre les classes.
- Spécialisation est l'inverse
- Héritage
  - Une sous-classe hérite des propriétés de la superclasse

La généralisation/ spécialisation

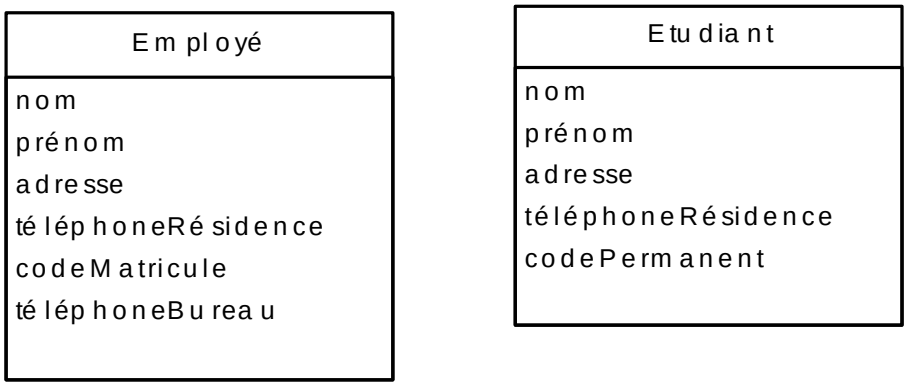
Personne est une généralisation (superclasse) d'*Employé* et *Etudiant*

Employé et *Étudiant* sont des spécialisations de *Personne*



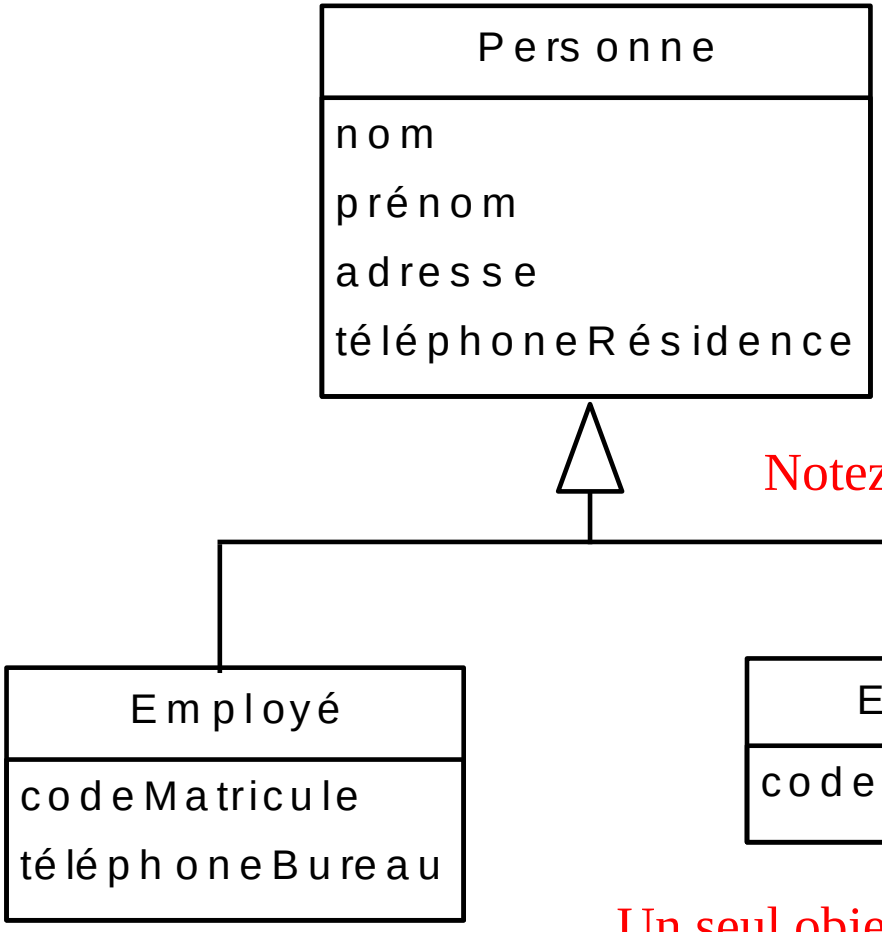
Propriétés communes : classe plus générale

Classe équivalente sans généralisation



Héritage

Notation multi-segments



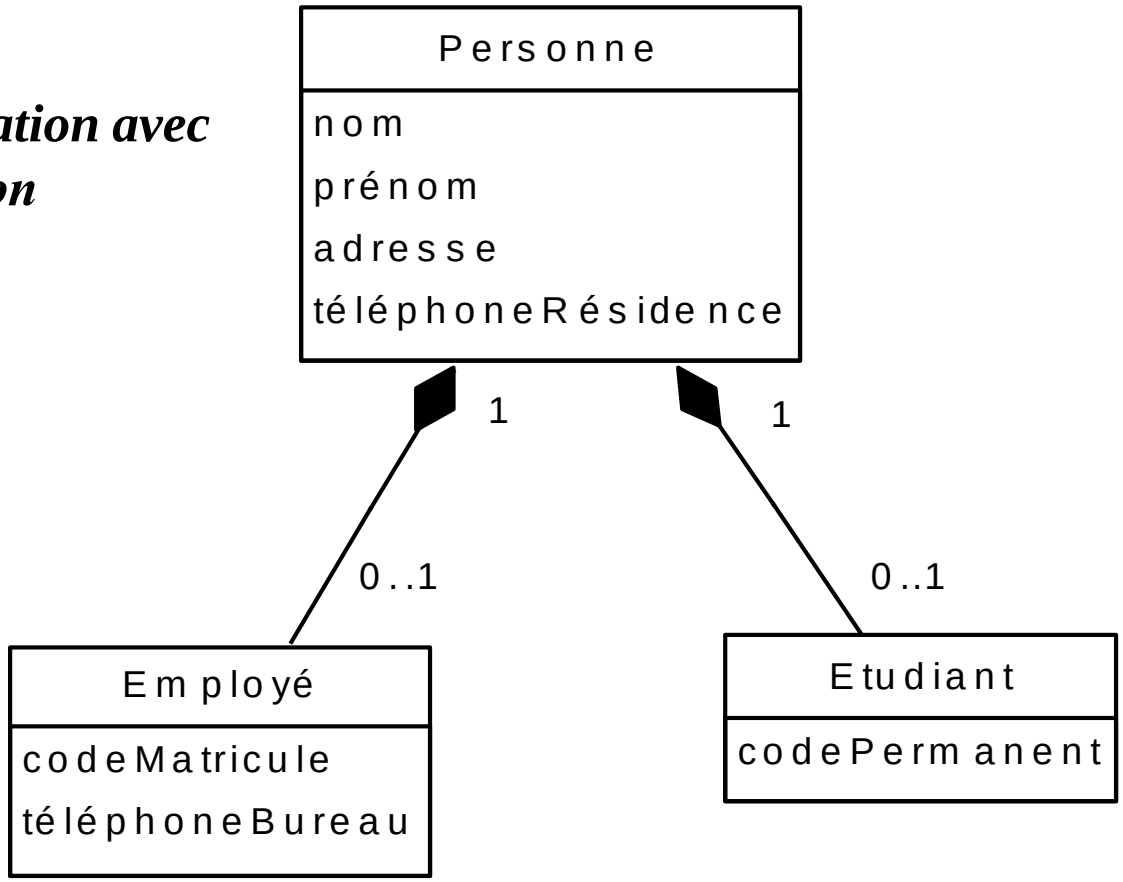
Personne est une généralisation (superclasse) d'*Employé* et *Etudiant*

Notez : pas de multiplicité

Un seul objet : étudiant (personne)

Mise en facteur par délégation ?

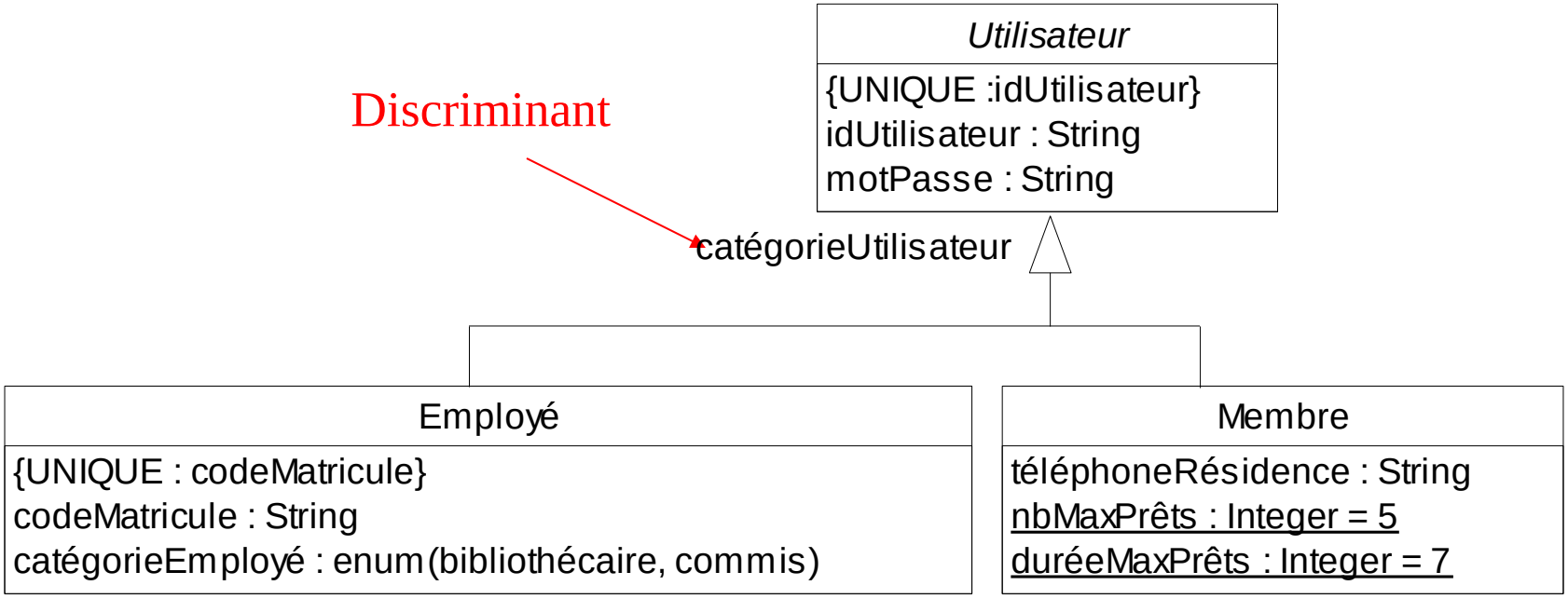
Représentation avec l'agrégation



Un employé est représenté avec deux objets: un objet de la classe Employé et un objet de la classe Personne

Discriminant

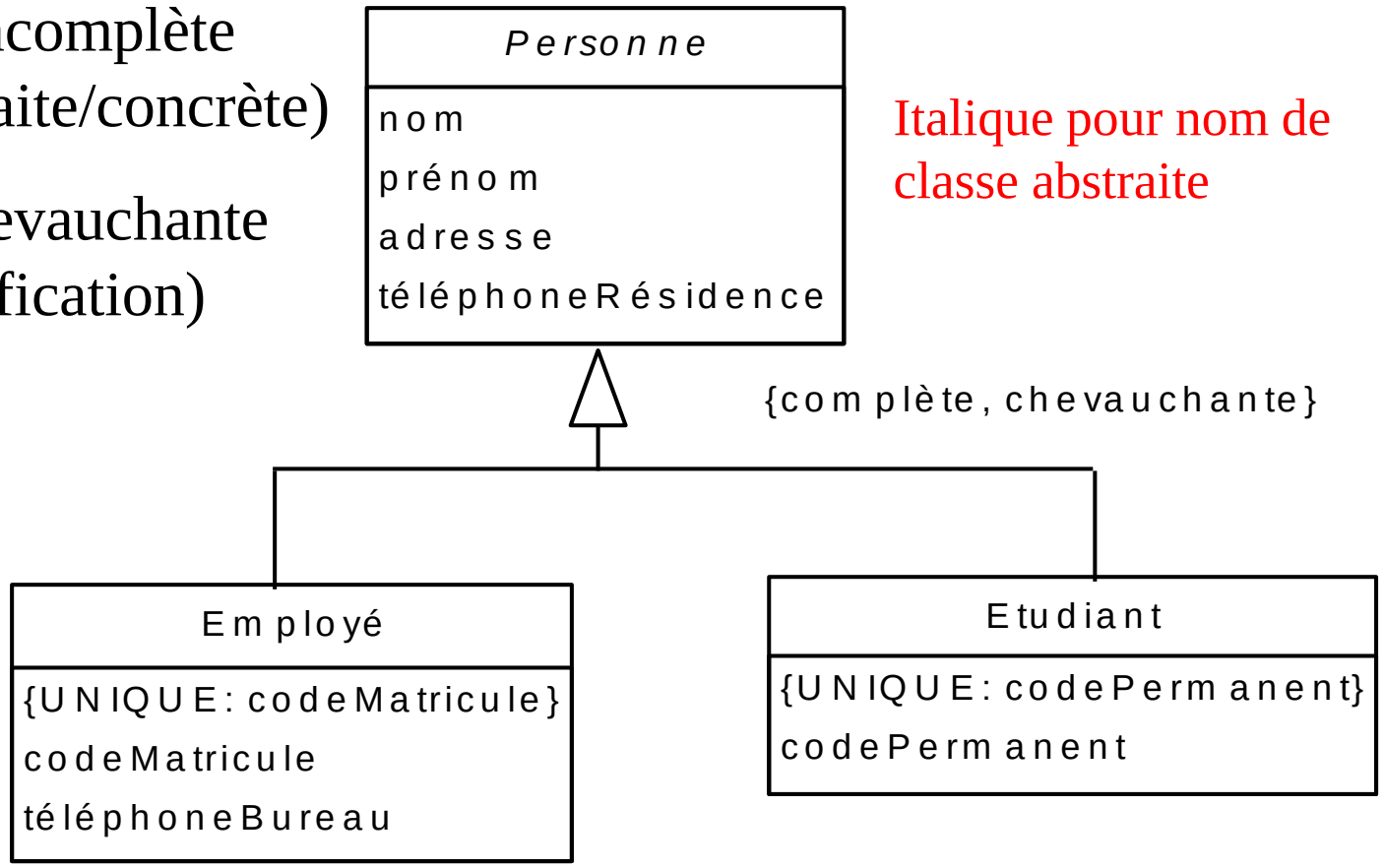
Le discriminant indique le critère de partitionnement des objets de la *superclasse* en *sous-classes*



Contraintes pré-définies pour la généralisation

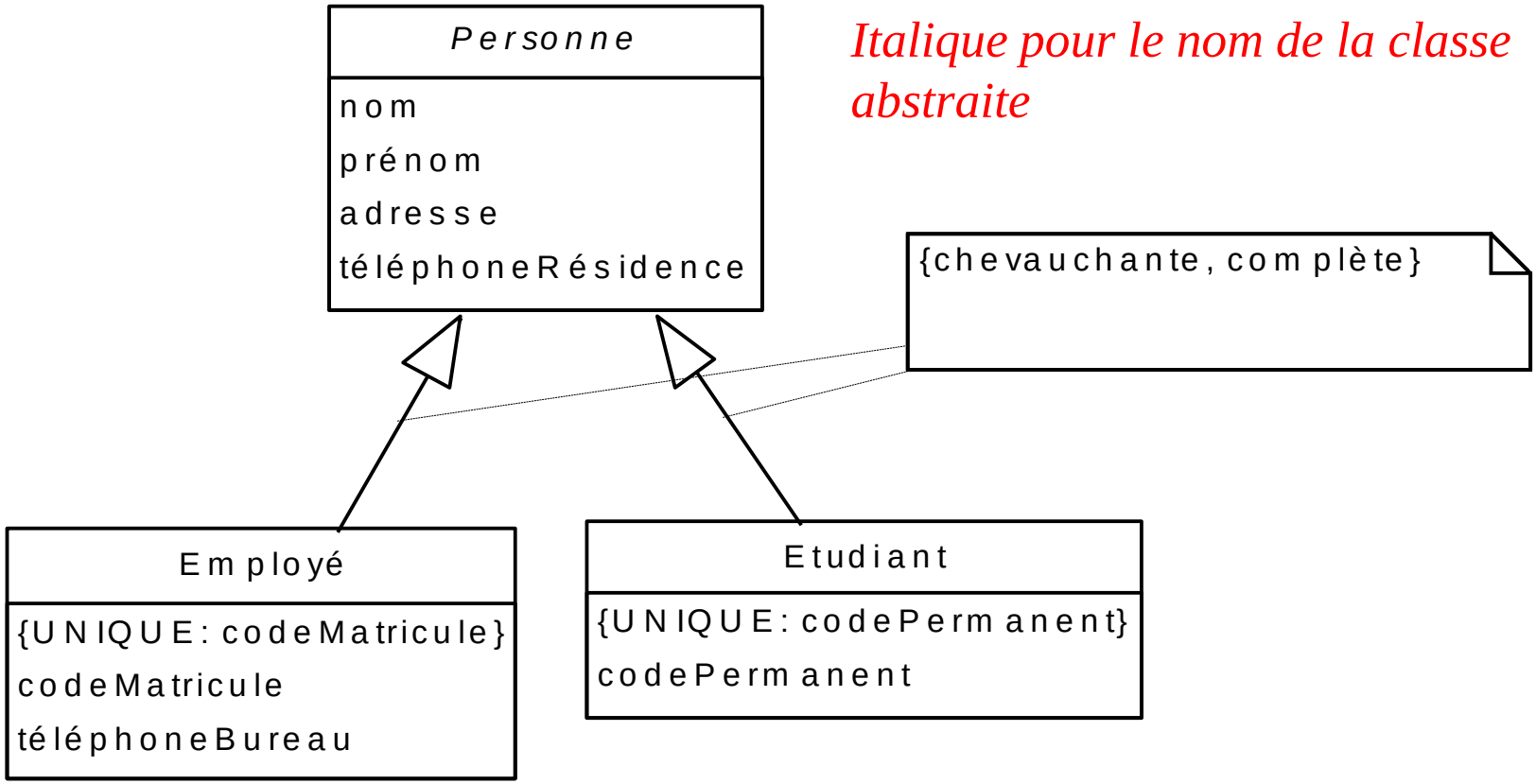
Complète /incomplète
(classe abstraite/concrète)

Disjointe/chevauchante
(multi-classification)



Italique pour nom de classe abstraite

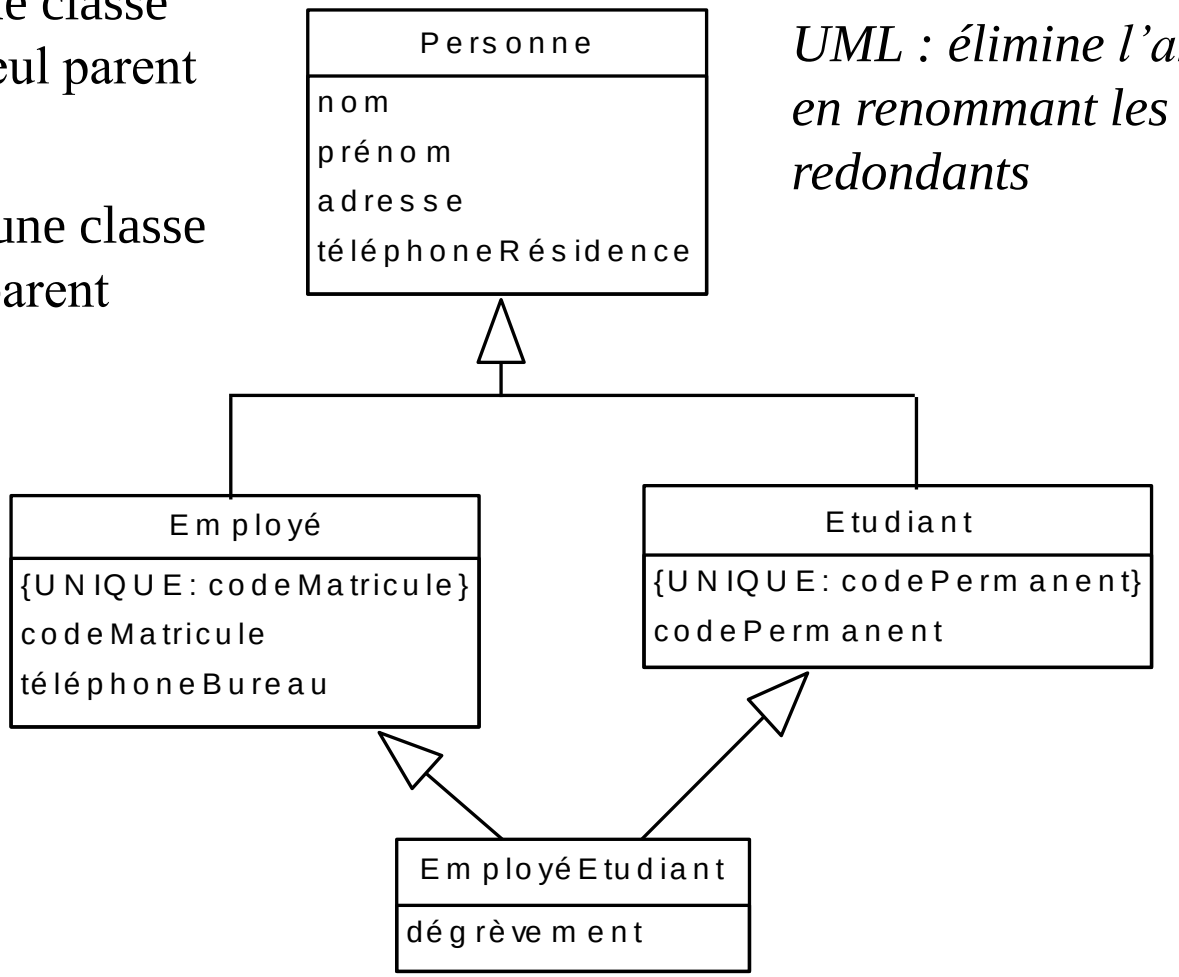
Notation alternative par une *note* UML



Héritage multiple

Héritage simple: une classe ne possède qu'un seul parent

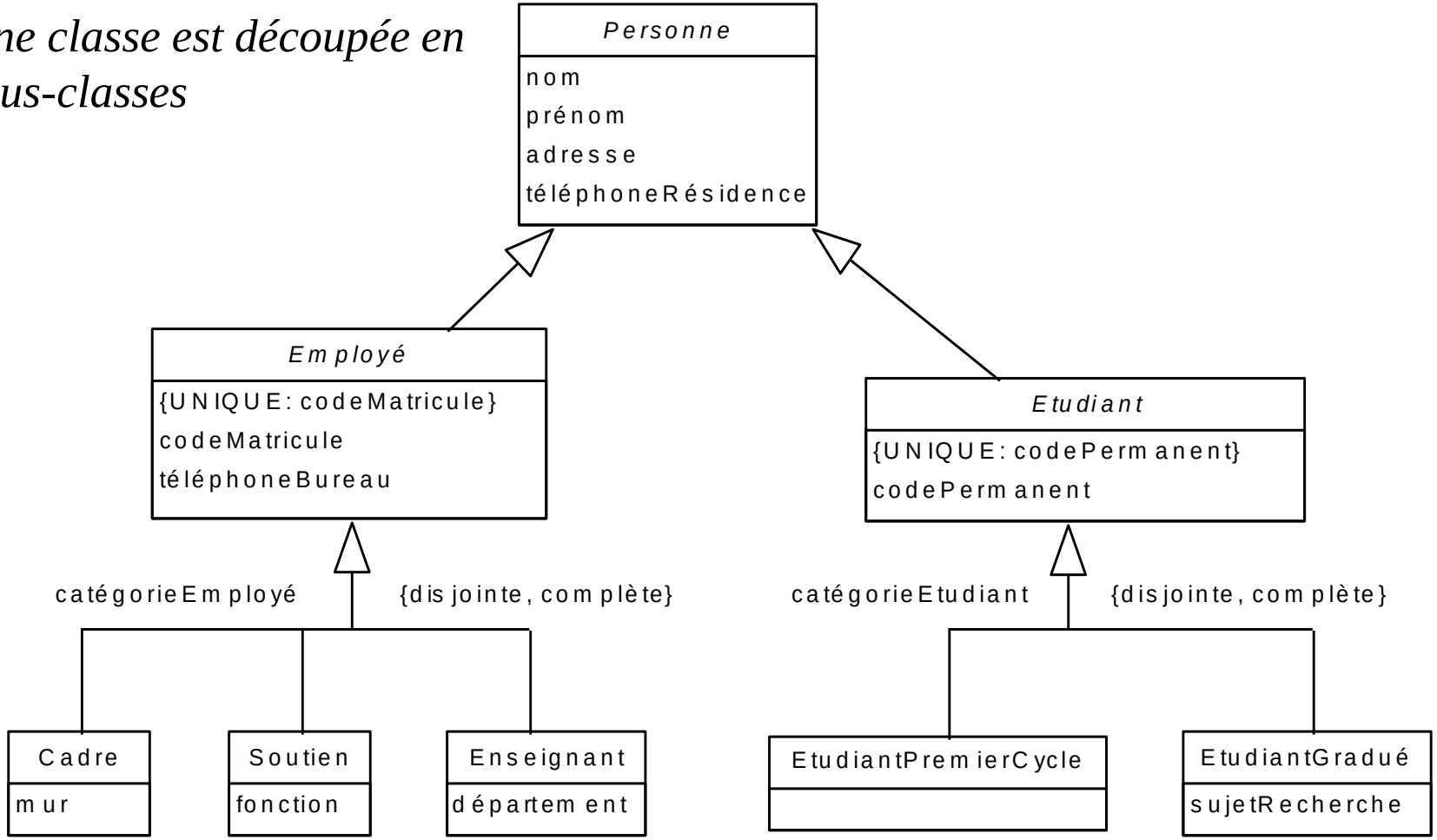
Héritage multiple: une classe possède plus d'un parent



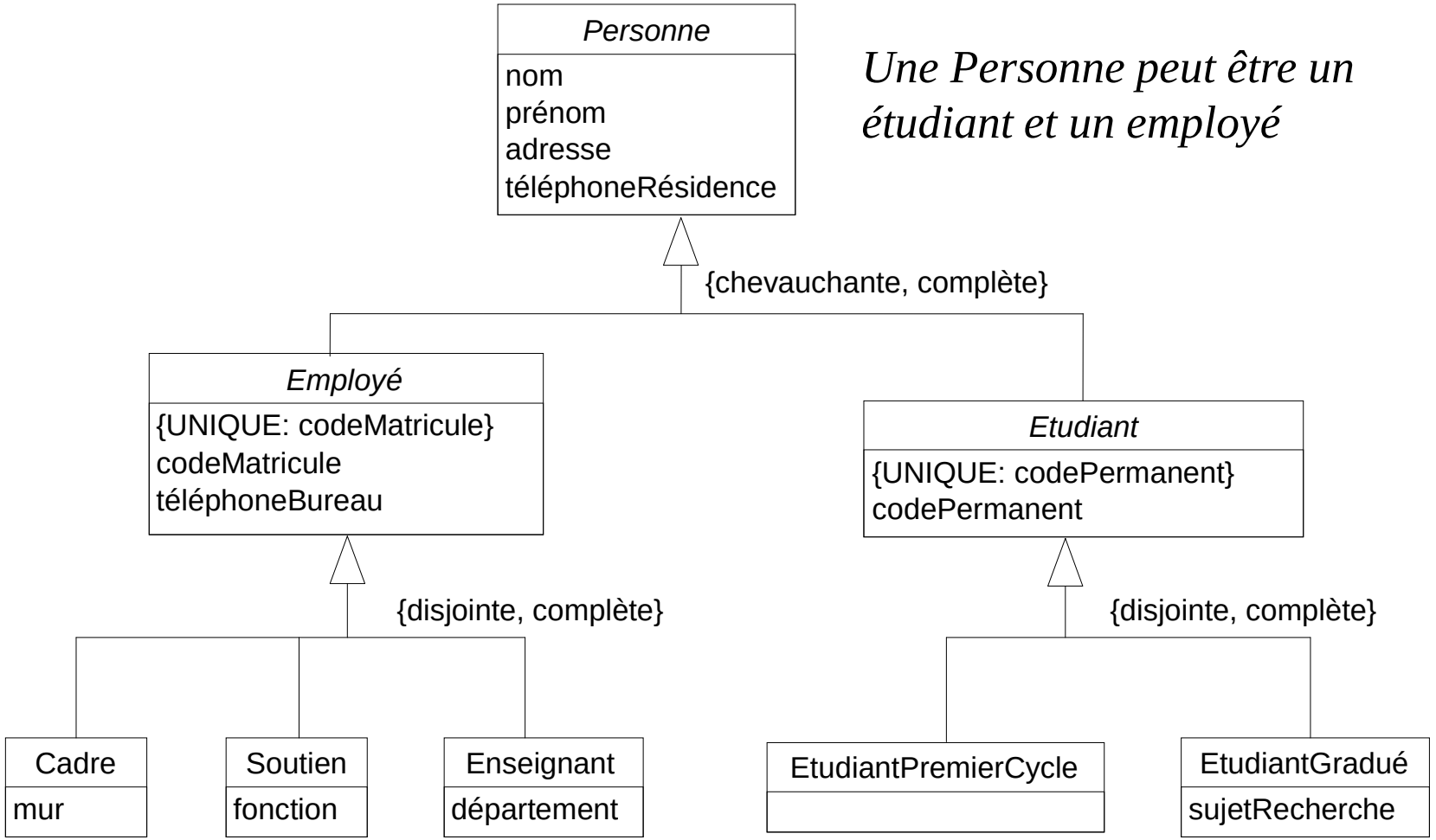
UML : élimine l'ambiguïté, en renommant les attributs redondants

Multi-classification et héritage multiple

Une classe est découpée en sous-classes



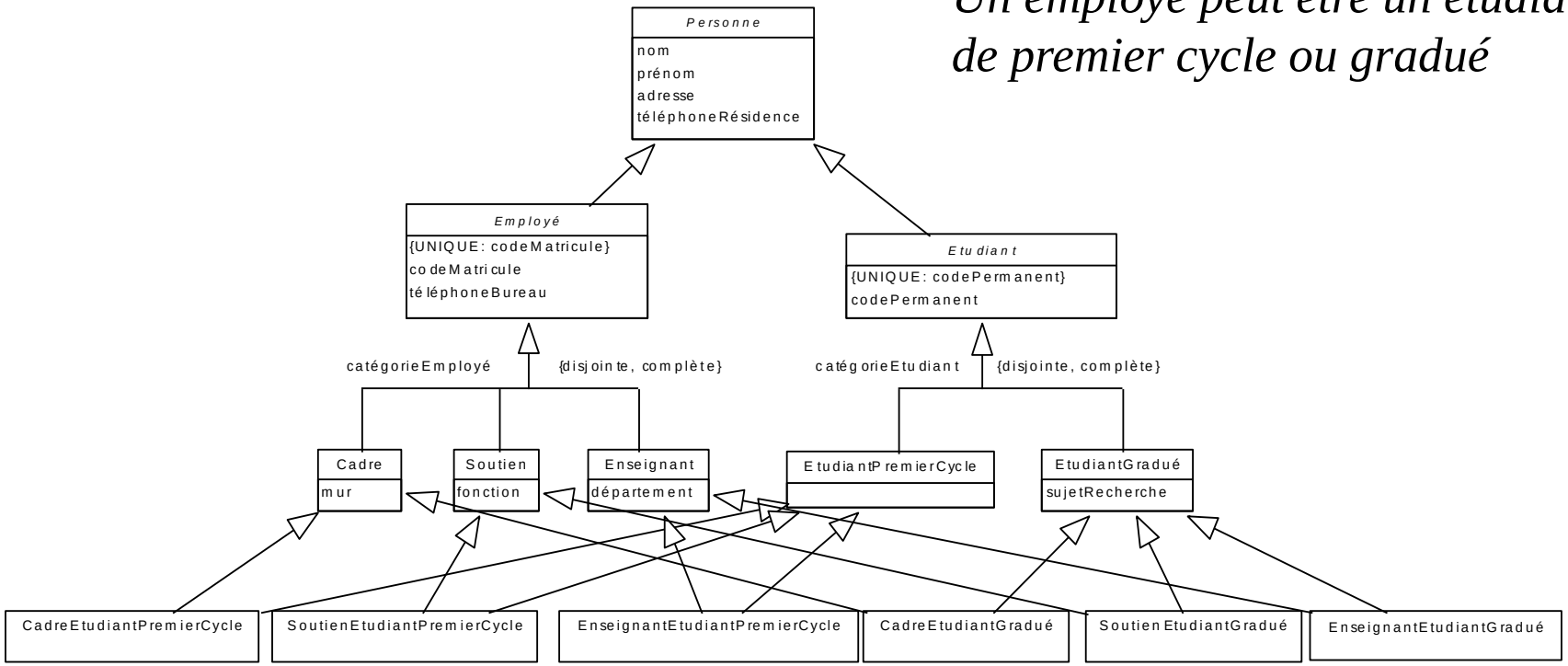
Multi-classification (suite)



Une Personne peut être un étudiant et un employé

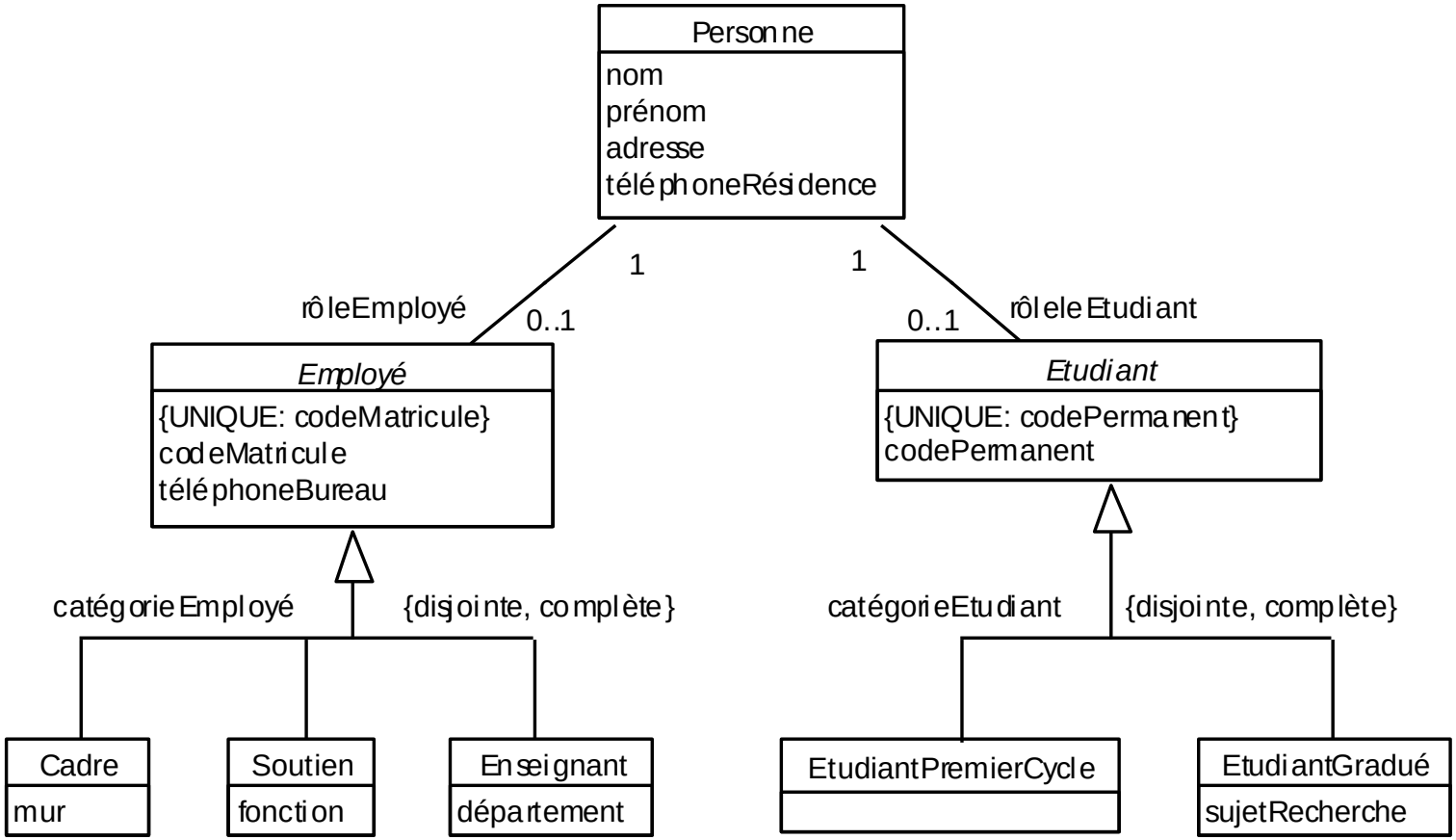
Sous-classes de jointure?

Un employé peut être un étudiant de premier cycle ou gradué



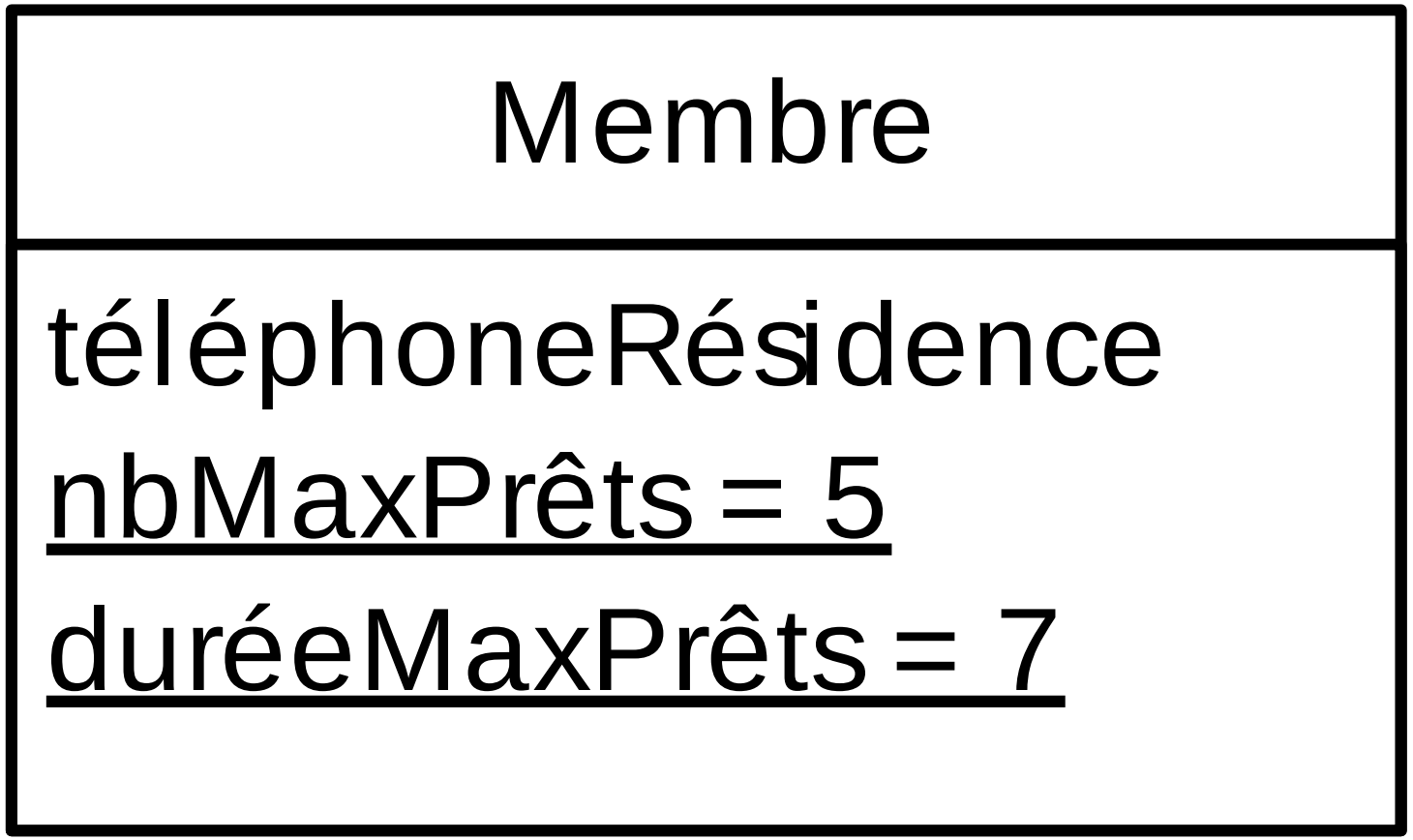
Sous-classes de jointure

Modélisation par rôle



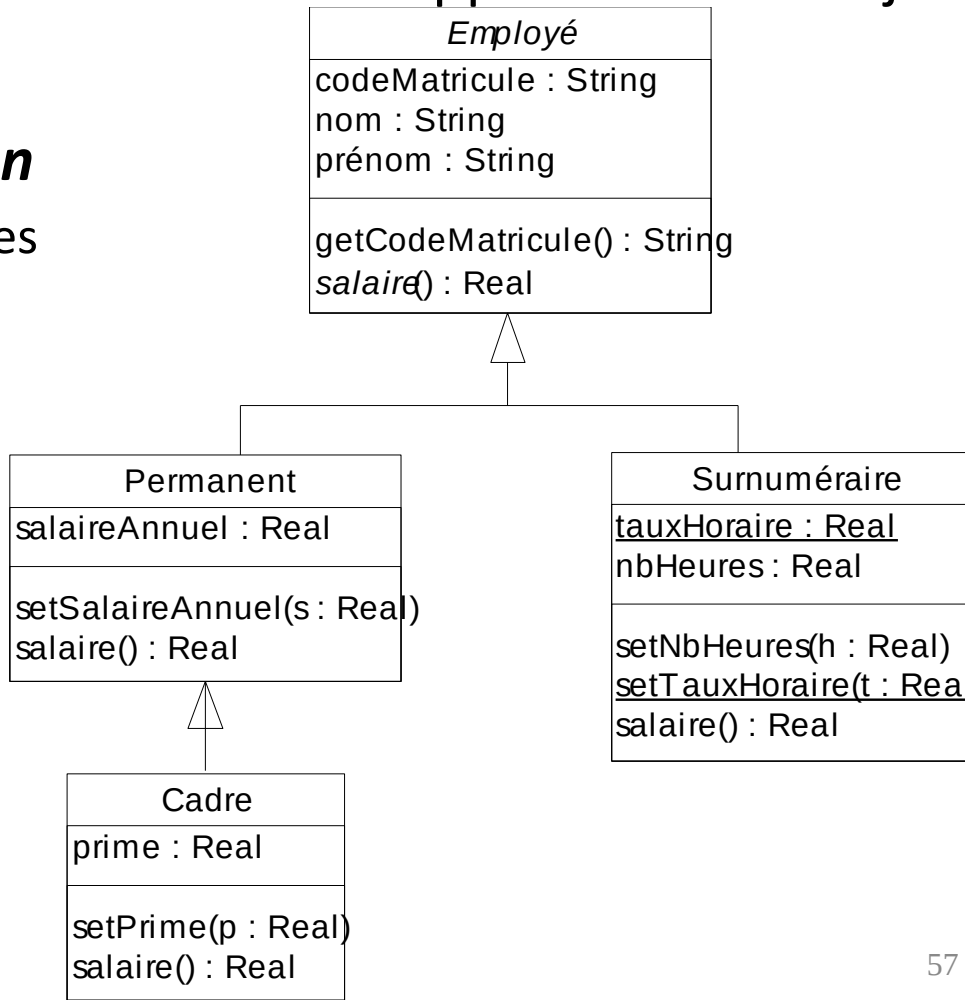
Attribut de classe

valeur fixe et unique pour tous les objets de la classe



Opérations

- Une opération représente un traitement applicable à un Objet
- ***Signature d'une opération***
  - nom et type des paramètres



Syntaxe générale pour la spécification des opérations en UML

- [«*stéréotype*»][*visibilité*] *nom* [(*listeParamètres*)] [: *typeRetour*] [*{propriétés}*]
  - *visibilité* peut être :
    - \+ publique
    - # protégé
    - \- privé
  - *nom* de l ’opération
  - *listeParamètres*
    - syntaxe d ’un paramètre
    - [*direction*] *nomParamètre* : *typeParamètre* [= *valeurDeDéfaut*]
      - *direction* (in, out ou inout)

Syntaxe pour opérations (suite)

- [«*stéréotype*»][*visibilité*] *nom* [(*listeParamètres*)] [: *typeRetour*] [*{propriétés}*]
  - *typeRetour*
    - optionnel
  - *propriétés*
    - *'user defined'*
    - *ex: « valeurs des attributs de l'objet non modifiable »*
  - *portée*
    - souligner opération de classe (*Rational Rose* : $)
  - *abstraite*
    - en italique (sans implémentation au niveau de la classe, ex. Implémentation spécifiée au niveau des sous classes)

Interface

- Interface d'un objet / d'une classe :
  - L'ensemble des opérations publiques d'un objet / des objets d'une classe
  - Inclue les opérations héritées des classes plus générales

Définitions

- ***Méthode***
  - une implémentation d'une opération
- ***Polymorphisme***
  - même signature d'opération
  - méthodes distinctes pour des classes distinctes
- ***Surcharge* (« *overloading* »)**
  - même nom avec signatures différentes
- ***Redéfinition***
  - même nom, même signature
  - dans une hiérarchie de sous-classes

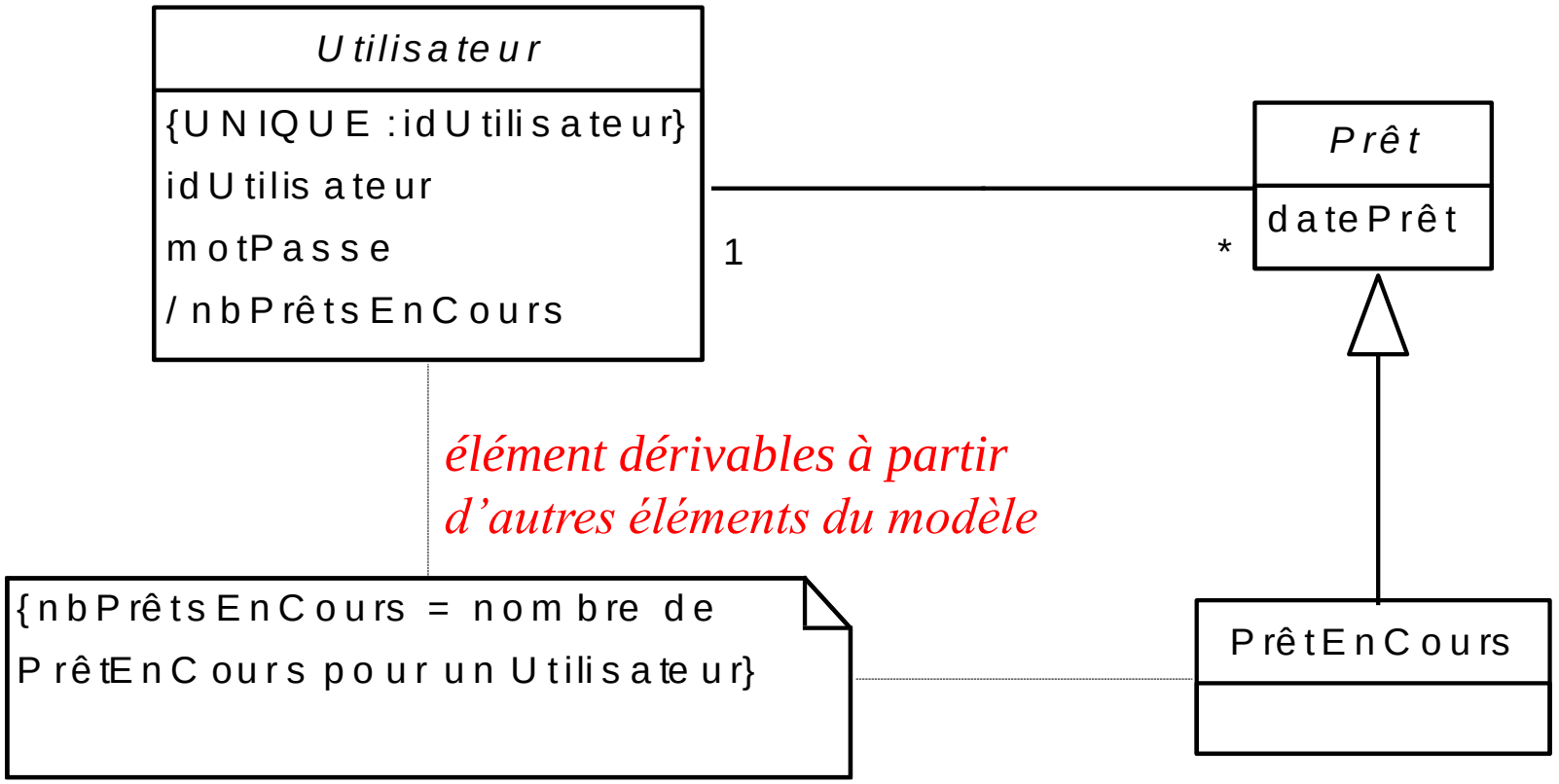
Catégories d ’opérations

- *Constructeur :* crée et initialise l’état d’un nouvel objet de la classe
- *Modifieur :* modifie l’état d’un objet
- *Lecteur :* retourne des valeurs d’attributs d’un objet
- …

Spécification de contraintes

- Entre { }
- A proximité de l ’élément concerné
  - après spécification d ’un attribut
  - avant un ensemble d ’attributs
- Note reliée aux éléments
- Près d ’un trait pointillé
- Près d ’une flèche pointillée (modèle relationnel)
- Syntaxe
  - langue naturelle
  - Langage formel: *OCL* (*object constraint langage*) - version 1.1 d ’UML

Éléments dérivés



Utilisateur

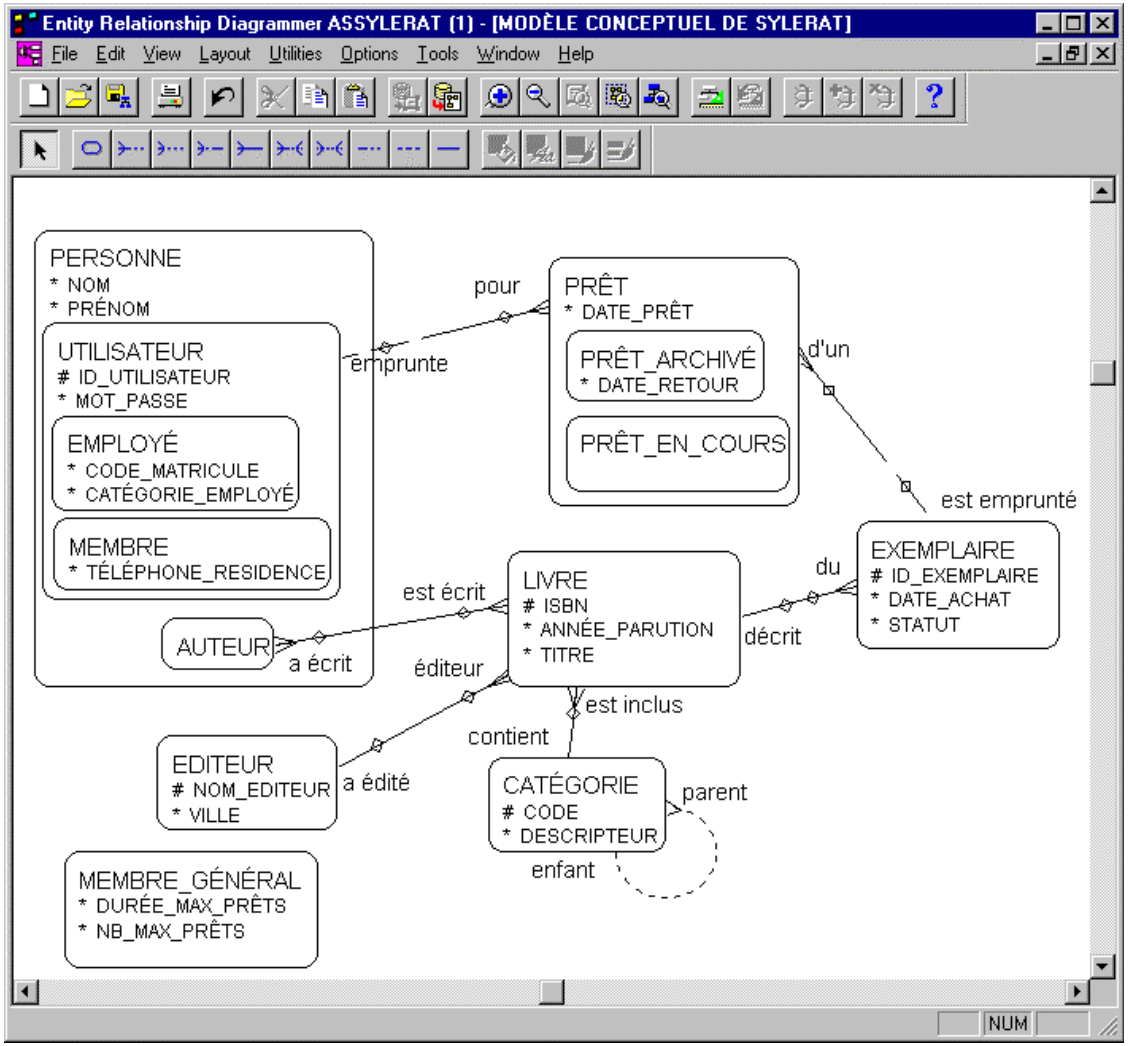
```
self.nbPrêtsEnCours = self.prêtEnCours -> size
```

nbPrêtEnCours est calculé en comptant le nombre de liens avec les objets de la classe *PrêtEnCours*

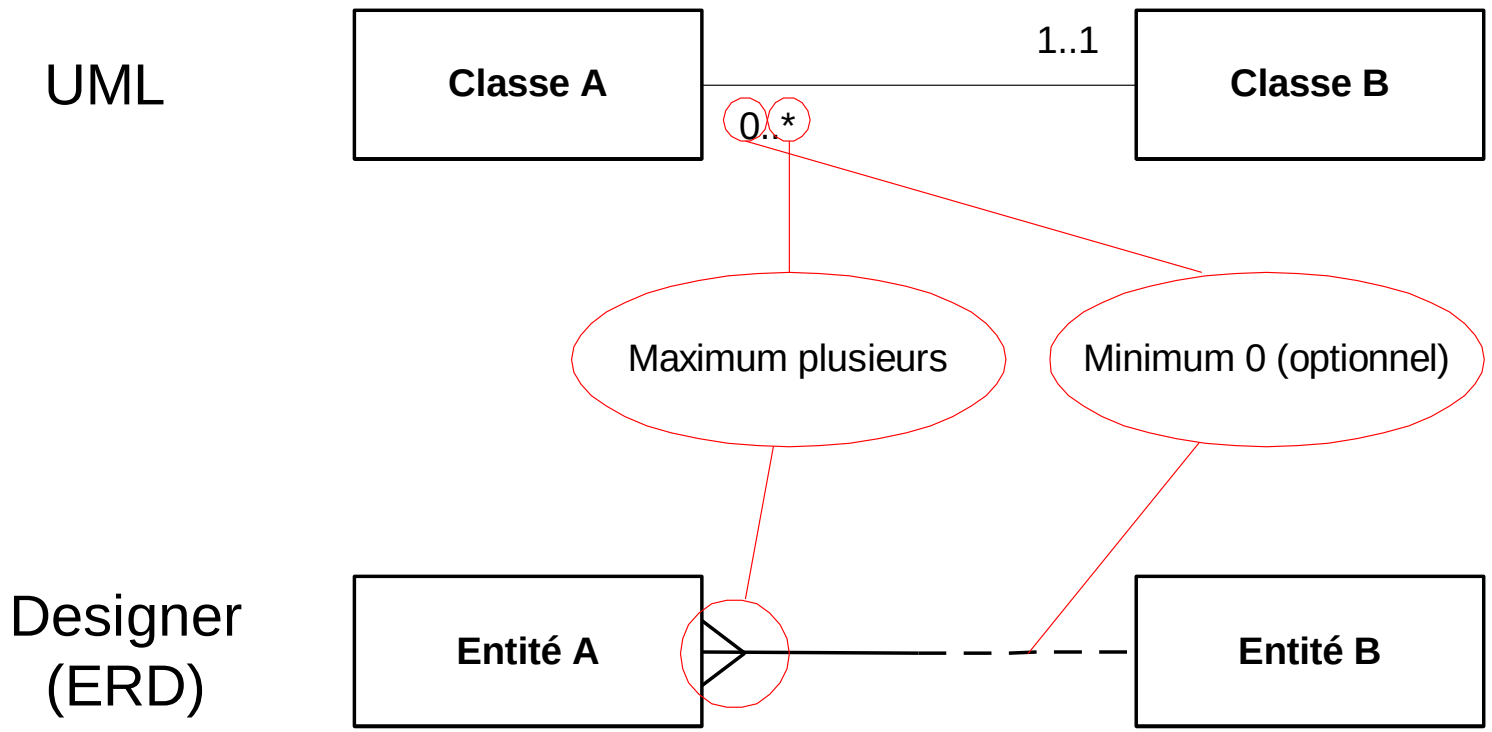
Notation pour les diagrammes de classe UML

- Classe
- Relation de généralisation/spécialisation
- Association binaire
- Agrégation
- Composition
- Association qualifiée
- Classe associative
- Élément dérivable
- Contrainte d’intégrité

Modèle entité-association : ERD de Oracle Designer



Notation des multiplicités



Résumé des concepts

Stéréotype (UML)	« *actor* » (use case), « *entity* » (classe), « *datatype* » (classe typée – pour types complexes), « 'any' » (user defined)
Attributs	Variables d'une classe; valeurs d'un objets = état; attribut de classe
Classe	Entité données persistantes; structure commune d'un ensemble d'objets - superclasse, sous-classe, abstraite
Associations (nom, rôles)	Relation entre classes; liens entre objets (1 seul lien entre 2 objets) - réflexive, binaire, n-aire - agrégation, composition, qualifiée, classe, réifiée, hiérarchique
Contraintes associations	{modifiable}, {ajout}, {fixe}, {exclusive}, {ordonnée}
Contraintes spécialisations	{(disjointe/chevauchante)}, {(complète/incomplète)}; discriminant
Délégation	Factorisation par composition
Héritage multiple	Plusieurs parents; multiclassification et sous-Héritage de jointure
Opérations	Comportement des objets / classes; signature; surcharge; redéfinition; - constructeurs; lecteurs; modifieurs (encapsulation)
Polymorphisme	Capacité d'exécuter une même opération de façon différente sur des objets différents (héritant d'une classe commune).