Modélisation

© Olivier Caron

Polytech'Lille

Polytech'Lille

Méthodologie et Modélisation

√ Méthodologie : démarche de conception

Méthodologie et Modélisation

- √ Méthodologie : démarche de conception
- √ Modélisation : description structurelle

Méthodologie et Modélisation

- √ Méthodologie : démarche de conception
- √ Modélisation : description structurelle
- √ Certaines méthodes sont liées à une notation : OMT, Merise (MCD)

Méthodologie et Modélisation

- √ Méthodologie : démarche de conception
- √ Modélisation : description structurelle
- √ Certaines méthodes sont liées à une notation : OMT, Merise (MCD)
- √ La **notation** UML indépendante de toute méthodologie

Le modèle Entité-Association

√ Issu des travaux de Chen (US) et Européens

Le modèle Entité-Association

- √ Issu des travaux de Chen (US) et Européens
- √ Objectifs :
  - Puissance de représentation

Le modèle Entité-Association

- √ Issu des travaux de Chen (US) et Européens
- √ Objectifs :
  - Puissance de représentation
  - Stabilité et flexibilité : un **ajout** de donnée ne doit pas remettre en cause le schéma.

Le modèle Entité-Association

- √ Issu des travaux de Chen (US) et Européens
- √ Objectifs :
  - Puissance de représentation
  - Stabilité et flexibilité : un **ajout** de donnée ne doit pas remettre en cause le schéma.
  - Simplicité : facilité de compréhension et d'utilisation

Le modèle Entité-Association

- √ Issu des travaux de Chen (US) et Européens
- √ Objectifs :
  - Puissance de représentation
  - Stabilité et flexibilité : un **ajout** de donnée ne doit pas remettre en cause le schéma.
  - Simplicité : facilité de compréhension et d'utilisation
  - Indépendance par rapport à l'implémentation cible (SGBDR, fichiers, programmation. . .)

Le modèle Entité-Association

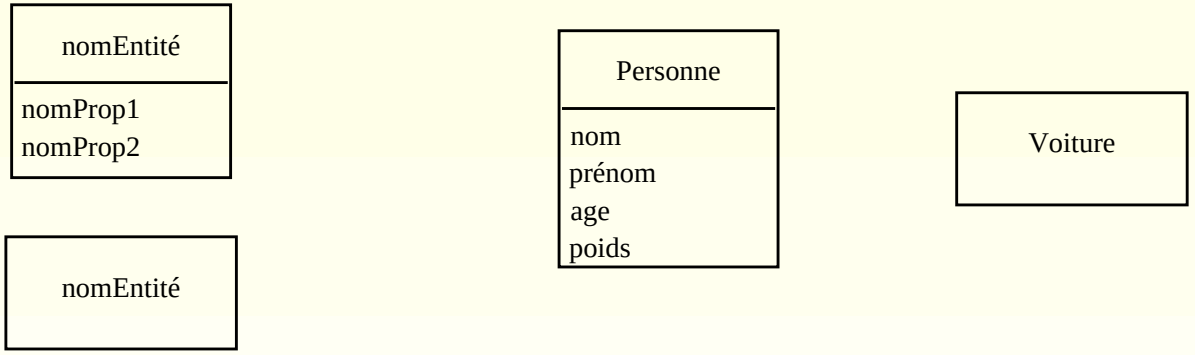
- √ Issu des travaux de Chen (US) et Européens
- √ Objectifs :
  - Puissance de représentation
  - Stabilité et flexibilité : un **ajout** de donnée ne doit pas remettre en cause le schéma.
  - Simplicité : facilité de compréhension et d'utilisation
  - Indépendance par rapport à l'implémentation cible (SGBDR, fichiers, programmation. . .)
- √ Plusieurs réalisations : MCD (Merise) , OMT (+méthodologie), UML,. . .

L'entité

Définition 1. Entité : *c'est un objet discernable d'autres objets comme, par exemple, une personne, une voiture mais qui peut être aussi un concept ou une grandeur abstraite. L'entité est définie par une liste de propriétés qui la caractérisent. Celles-ci constituent le plus petit élément d'information ayant un sens par lui-même.*

L'entité

Définition 1. Entité : *c'est un objet discernable d'autres objets comme, par exemple, une personne, une voiture mais qui peut être aussi un concept ou une grandeur abstraite. L'entité est définie par une liste de propriétés qui la caractérisent. Celles-ci constituent le plus petit élement d'information ayant un sens par lui-même.*



Classe d'entité

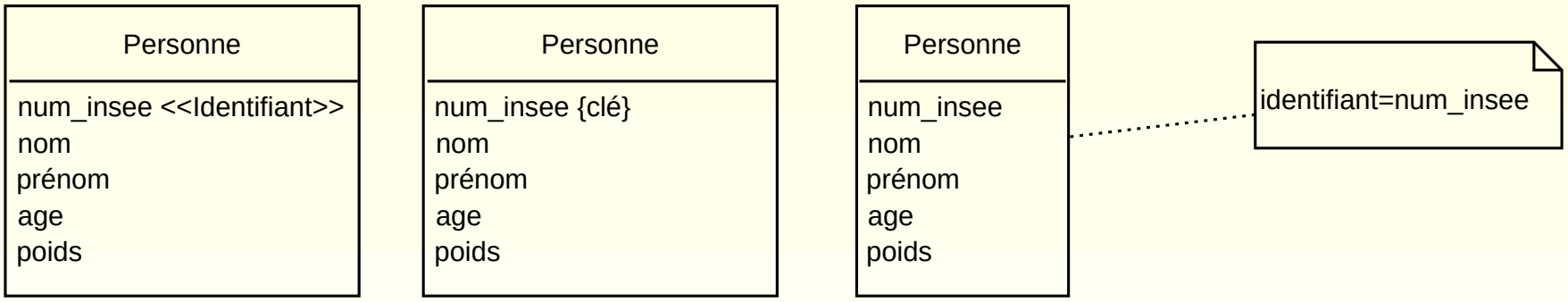
Définition 2. Classe d'entité : *c'est l'ensemble des entités de même type qu'il est possible de définir au cours du temps.*

Exemple : le `personnel`, classe d'entité de l'entité `employé`

On parle également d'*instances*

Identifiant

- √ Parmi les propriétés d'une entité, l'une doit permettre **d'identifier** une occurrence d'une entité parmi toutes les autres.
- √ On parle également de clé.
- √ Exemple : n°sécurité sociale d'une personne (Attention CNIL)
- √ Représentation graphique UML (contrainte ou stéréotype, notion de profil)



Terminologie UML

√ UML ainsi que les différents ateliers UML ont une terminologie différente :

- Entité → Classe
- Propriété → Attribut
- Identifiant → n'existe pas par défaut en UML

Association

Définition 3. *C'est un lien logique entre l'ensemble des entités appartenant à plusieurs classes différentes.*

- √ Souvent perçue comme une action menée vis à vis des entités (et se traduit alors par un verbe).
- √ La mise en relation d'entités peut faire apparaître des propriétés qui n'appartiennent en propre à aucune des entités. On distingue :
  - ▶ Les associations binaires relient les différentes instances de deux classes d'entité (imposé par OMT et ODMG)

Association

Définition 3. *C'est un lien logique entre l'ensemble des entités appartenant à plusieurs classes différentes.*

- √ Souvent perçue comme une action menée vis à vis des entités (et se traduit alors par un verbe).
- √ La mise en relation d'entités peut faire apparaître des propriétés qui n'appartiennent en propre à aucune des entités. On distingue :
  - ▶ Les associations binaires relient les différentes instances de deux classes d'entité (imposé par OMT et ODMG)
  - ▶ Les associations n-aire relient les instances de n-classes d'entité.

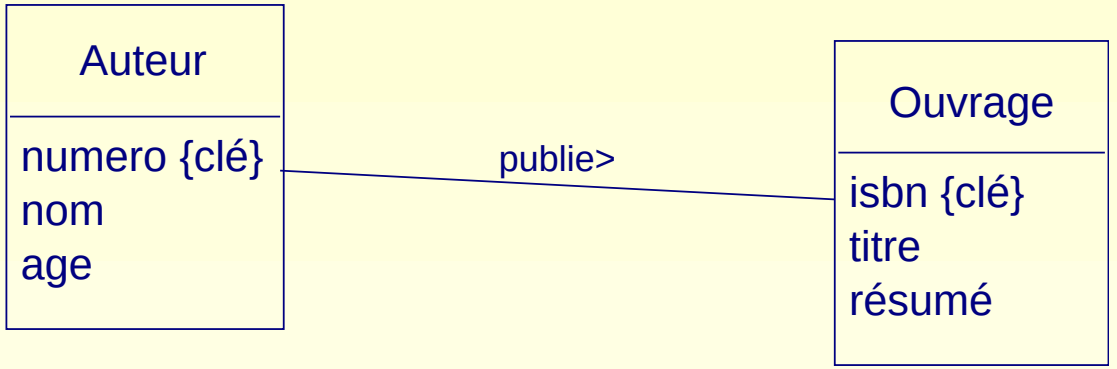
Association

Définition 3. *C'est un lien logique entre l'ensemble des entités appartenant à plusieurs classes différentes.*

- √ Souvent perçue comme une action menée vis à vis des entités (et se traduit alors par un verbe).
- √ La mise en relation d'entités peut faire apparaître des propriétés qui n'appartiennent en propre à aucune des entités. On distingue :
  - ▶ Les associations binaires relient les différentes instances de deux classes d'entité (imposé par OMT et ODMG)
  - ▶ Les associations n-aire relient les instances de n-classes d'entité.
  - ▶ Les associations réflexives relient une instance d'une classe d'entité à d'autres instances d'une même classe.

Association simple

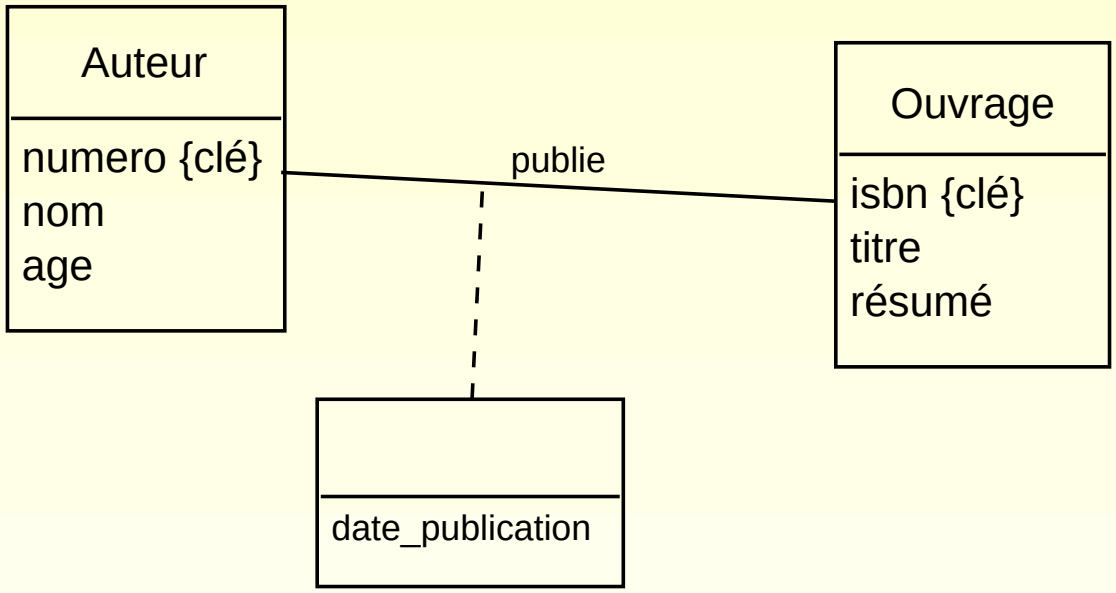
√ Représentation graphique UML :



√ Indiquez le sens de lecture du nom de l'asociation

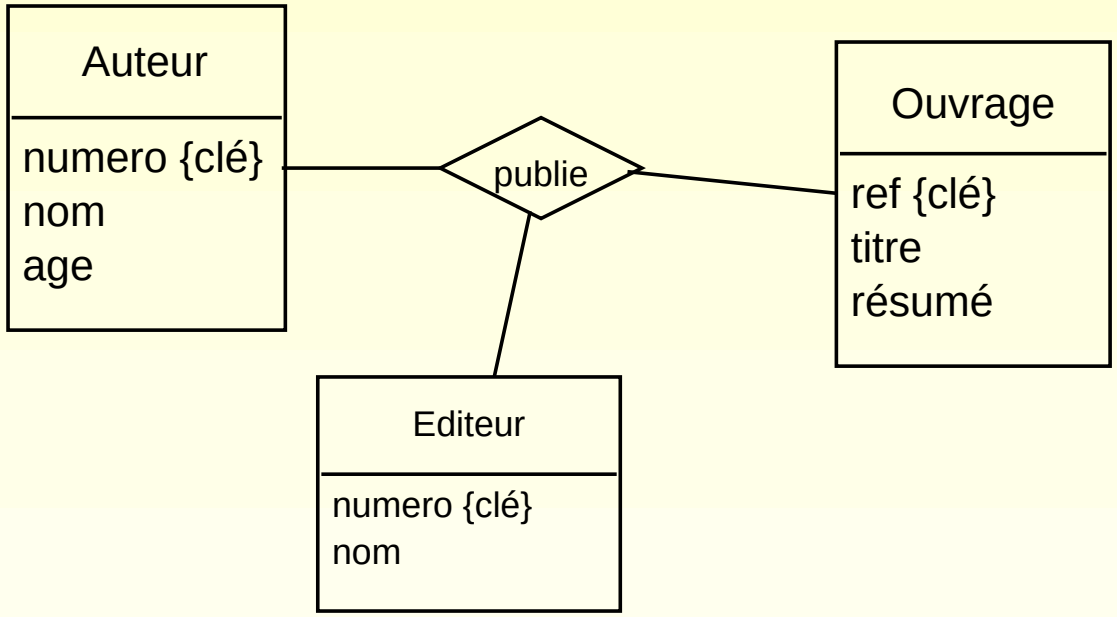
Propriétés d'association

√ Représentation graphique UML :



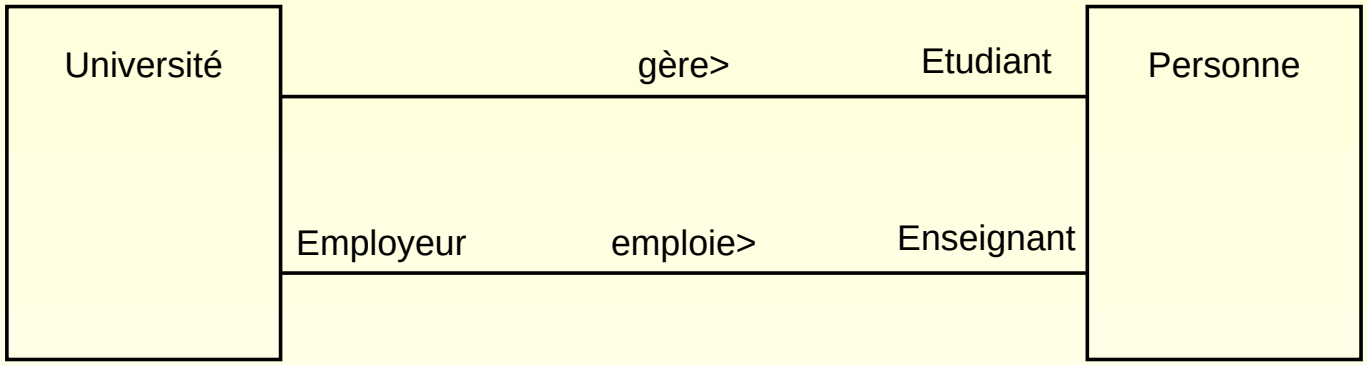
Association n-aire

√ Représentation graphique UML :



Les rôles d'une association

- √ Cette notion n'existe pas avec MERISE
- √ Représentation graphique UML :



L'agrégation

√ Forme particulière d'association (inconnue avec MERISE)

L'agrégation

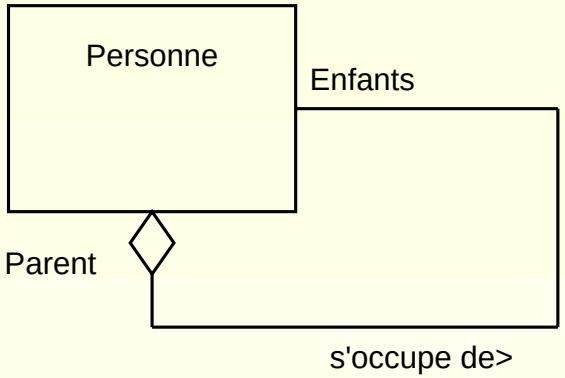
- √ Forme particulière d'association (inconnue avec MERISE)
- √ Relation de type maître-esclaves

L'agrégation

- √ Forme particulière d'association (inconnue avec MERISE)
- √ Relation de type maître-esclaves
- √ Une des classes joue un rôle plus important que l'autre dans la relation

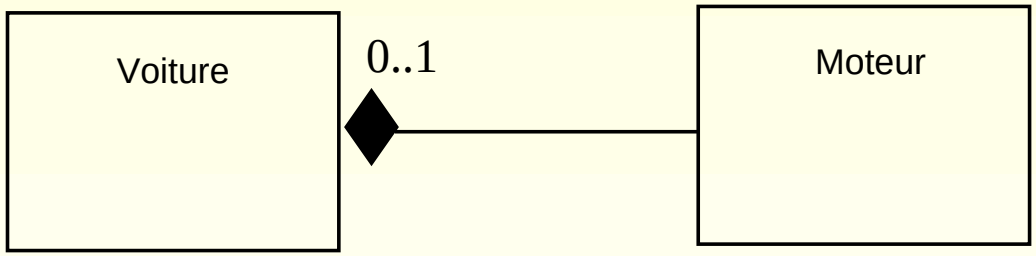
L'agrégation

- √ Forme particulière d'association (inconnue avec MERISE)
- √ Relation de type maître-esclaves
- √ Une des classes joue un rôle plus important que l'autre dans la relation
- √ Représentation graphique UML :



La composition

- √ Forme particulière d'association (inconnue avec MERISE)
- √ Relation de type maître-esclaves
- √ Il ne peut y avoir qu'un seul maître.
- √ Classes composites physiquement contenus par l'agrégat (cycle de vie)
- √ Représentation graphique UML :



La navigation

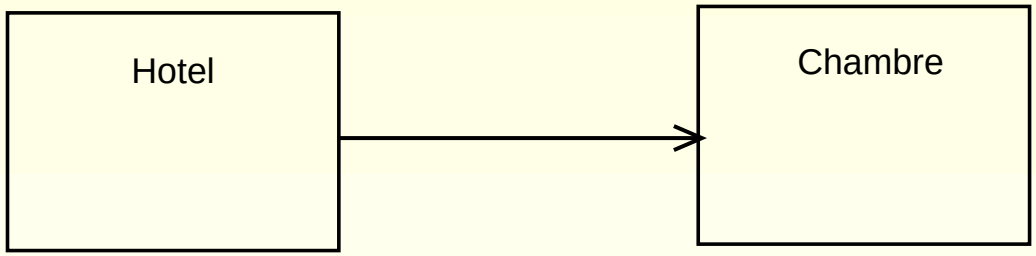
√ Forme particulière d'association (inconnue avec MERISE)

La navigation

- √ Forme particulière d'association (inconnue avec MERISE)
- √ Permet de se déplacer dans le modèle

La navigation

- √ Forme particulière d'association (inconnue avec MERISE)
- √ Permet de se déplacer dans le modèle
- √ Correspond aux différents scénarios (dynamique)
- √ Par défaut, les associations sont navigables dans les deux directions
- √ Représentation graphique UML :



Cardinalités d'une association

√ Elles précisent les nombres minimum et maximum d'occurrences d'une entité pouvant être impliquées dans les occurrences d'association.

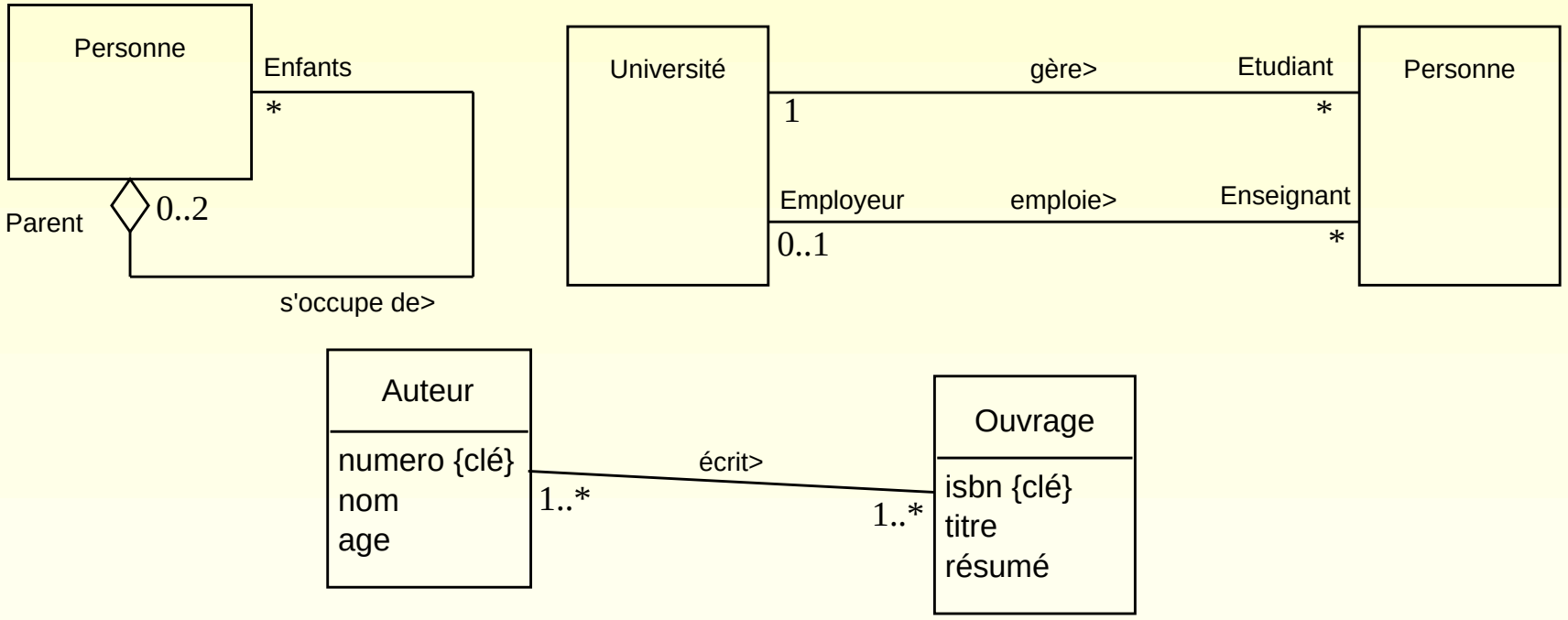
Cardinalités d'une association

- √ Elles précisent les nombres minimum et maximum d'occurrences d'une entité pouvant être impliquées dans les occurrences d'association.
- √ définie au niveau de chaque extrémité de l'association (Attention, UML inverse de MCD Merise)
- √ Conventions d'affichage :

représentation	signification
1	un et un seul
0..1	zéro et un
M..N	de M à N
*	de zéro à plusieurs
0..*	de zéro à plusieurs
1..*	de un à plusieurs

Cardinalités

√ Exemples UML



UML, c'est aussi . . .

√ Des types pour les propriétés (attributs)

UML, c'est aussi . . .

- √ Des types pour les propriétés (attributs)
- √ Des paquetages

UML, c'est aussi . . .

- √ Des types pour les propriétés (attributs)
- √ Des paquetages
- √ Des droits d'accès (public, privé, etc)

UML, c'est aussi . . .

- √ Des types pour les propriétés (attributs)
- √ Des paquetages
- √ Des droits d'accès (public, privé, etc)
- √ De l'héritage

UML, c'est aussi . . .

- √ Des types pour les propriétés (attributs)
- √ Des paquetages
- √ Des droits d'accès (public, privé, etc)
- √ De l'héritage
- √ D'autres diagrammes (cf cours Analyse et dynamique)

Premier bilan UML

√ Outil simple, intuitif

Premier bilan UML

- √ Outil simple, intuitif
- √ Outil **non formel**

Premier bilan UML

- √ Outil simple, intuitif
- √ Outil **non formel**
  - Avantage : autorise un processus itératif, on affine le schéma à chaque étape.

Premier bilan UML

- √ Outil simple, intuitif
- √ Outil **non formel**
  - Avantage : autorise un processus itératif, on affine le schéma à chaque étape.
  - Inconvénient : est-ce que le schéma décrit est suffisant ?, est-il valide ?

Premier bilan UML

- √ Outil simple, intuitif
- √ Outil **non formel**
  - ▶ Avantage : autorise un processus itératif, on affine le schéma à chaque étape.
  - ▶ Inconvénient : est-ce que le schéma décrit est suffisant ?, est-il valide ?
- √ Une solution : le langage OCL (Object Constraint Language)

Le langage OCL (1/3)

√ Langage formel, textuel et normalisé (OMG).

Le langage OCL (1/3)

- √ Langage formel, textuel et normalisé (OMG).
- √ Des parsers OCL existent

Le langage OCL (1/3)

- √ Langage formel, textuel et normalisé (OMG).
- √ Des parsers OCL existent
- √ Complémentaire à la notation UML

Le langage OCL (1/3)

- √ Langage formel, textuel et normalisé (OMG).
- √ Des parsers OCL existent
- √ Complémentaire à la notation UML
- √ Permet de spécifier des contraintes sur tout schéma UML

Le langage OCL (1/3)

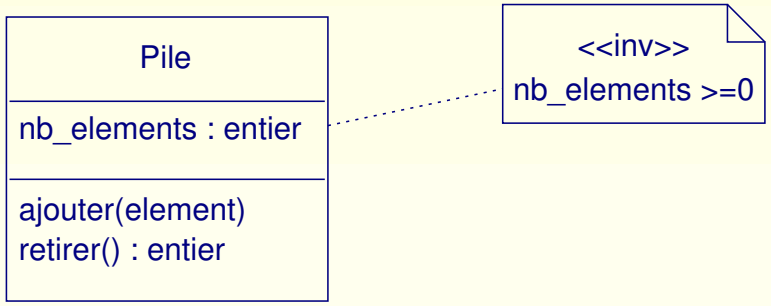
- √ Langage formel, textuel et normalisé (OMG).
- √ Des parsers OCL existent
- √ Complémentaire à la notation UML
- √ Permet de spécifier des contraintes sur tout schéma UML
- √ UML est lui-même spécifié avec OCL (méta-modèle).

Le langage OCL (1/3)

- √ Langage formel, textuel et normalisé (OMG).
- √ Des parsers OCL existent
- √ Complémentaire à la notation UML
- √ Permet de spécifier des contraintes sur tout schéma UML
- √ UML est lui-même spécifié avec OCL (méta-modèle).

Le langage OCL (2/3)

- √ Lié à un *contexte* : précise le type de l'instance auquel la contrainte se rapporte
- √ Les *stéréotypes suivants* sont applicables à une contrainte : `inv, pre, post`
- √ Exemple :



context Pile inv :
nb_elements >=0

Le langage OCL (3/3)

√ Parcours simple d'un schéma UML (via les associations)

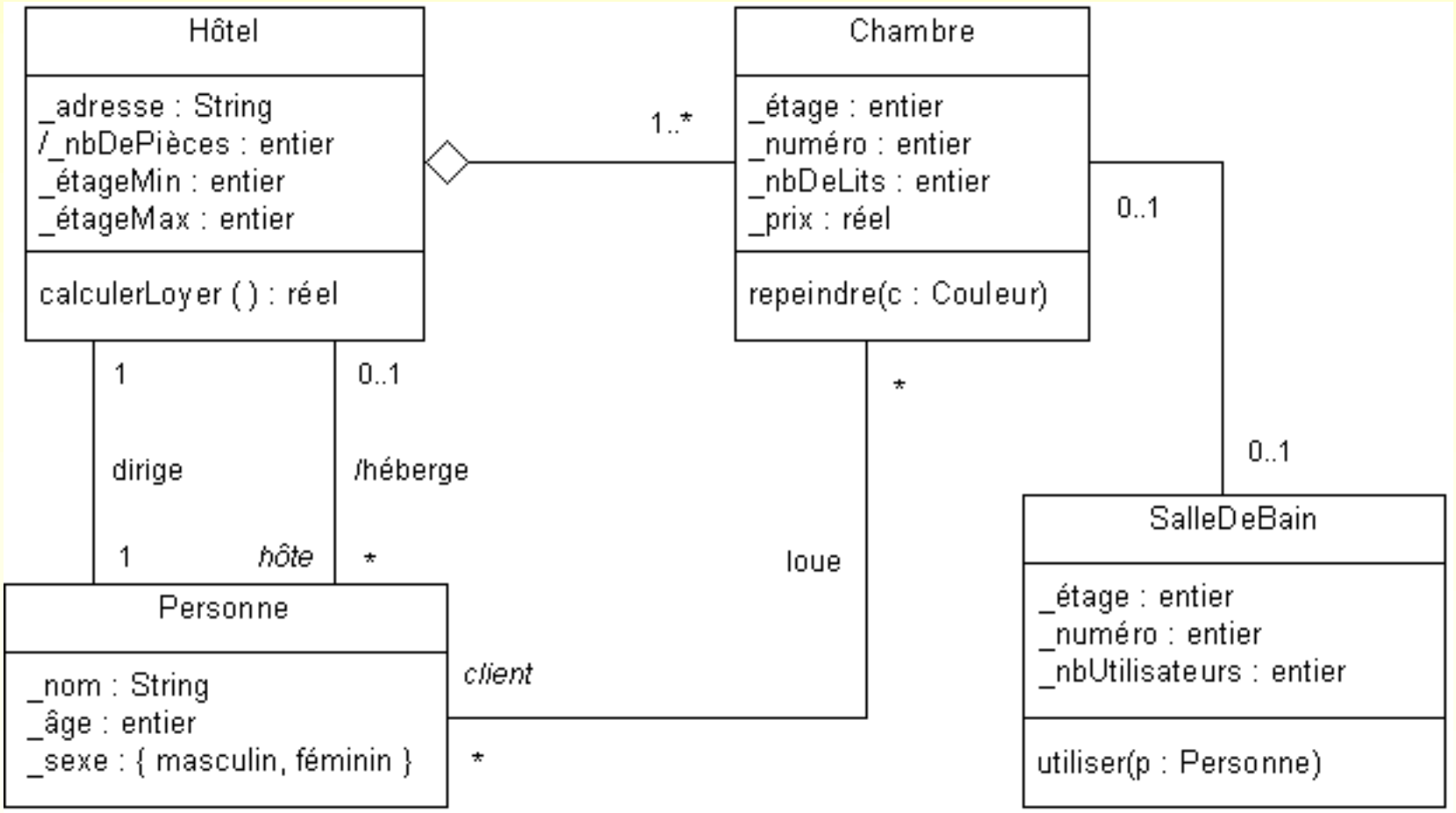
Le langage OCL (3/3)

- √ Parcours simple d'un schéma UML (via les associations)
- √ Expressions arithmétiques, booléennes

Le langage OCL (3/3)

- √ Parcours simple d'un schéma UML (via les associations)
- √ Expressions arithmétiques, booléennes
- √ Conformité des types (`oclIsTypeOf()`)
- √ Expressions ensemblistes (`collect`, `select`, `forAll`,...)

Exemple OCL, source JOOP 1999, uml.free.fr



Contraintes OCL (1/3)

√ Un hôtel ne contient jamais d'étage numéro 13 (superstition oblige).

```
context Chambre inv:
self._étage <> 13
```

√ Le nombre de personnes par chambre doit être inférieur ou égal au nombre de lits dans la chambre louée. Les enfants (accompagnés) de moins de 4 ans ne "comptent pas" dans cette règle de calcul (à hauteur d'un enfant de moins de 4 ans maximum par chambre).

```
context Chambre inv:
  client->size <= _nbDeLits or
  (client->size = _nbDeLits + 1 and
                  client->exists(p : Personne | p._âge < 4))
```

Contraintes OCL (2/3)

√ L'étage de chaque chambre est compris entre le premier et le dernier étage de l'hôtel.

```
context Hôtel inv:
 self.chambre->forAll(c : Chambre | c._étage <= self._étageMax and
                                    c._étage >= self._étageMin)
```

√ Chaque étage possède au moins une chambre (sauf l'étage 13, qui n'existe pas...).

```
context Hôtel inv:
 Sequence{_étageMin.._étageMax}->forAll(i : Integer |
   if i <> 13 then
     self.chambre->select(c : Chambre | c._étage = i)->notEmpty)
   endif)
```

Contraintes OCL (3/3)

- √ On ne peut repeindre une chambre que si elle n'est pas louée. Une fois repeinte, une chambre coûte 10 pour cent de plus.

```
context Chambre::repeindre(c : Couleur)
pre: client->isEmpty
post: _prix = _prix@pre * 1.1
```

- √ Une salle de bain privative ne peut être utilisée que par les personnes qui louent la chambre contenant la salle de bain et une salle de bain sur le palier ne peut être utilisée que par les clients qui logent sur le même palier.

```
context SalleDeBain::utiliser(p : Personne)
pre: if chambre->notEmpty then chambre.client->includes(p)
     else p.chambre._étage = self._étage
     endif
post: _nbUtilisateurs = _nbUtilisateurs@pre + 1
```

Schéma conceptuel UML et bases de données relationnelles

- √ Tout n'est pas obligatoire
- √ Trouver le bon compromis !
- √ Surcharge du schéma, difficulté de lecture
- √ Tout n'est pas possible facilement (exemple : héritage)
- √ La solution : définir un profil UML

Qu'est ce qu'un profil ? (1/2)

√ Mécanismes d'extension standard, *annoter* un schéma.

Qu'est ce qu'un profil ? (1/2)

- √ Mécanismes d'extension standard, *annoter* un schéma.
- √ Compatibilité UML, peut être lu par des outils du commerce

Qu'est ce qu'un profil ? (1/2)

- √ Mécanismes d'extension standard, *annoter* un schéma.
- √ Compatibilité UML, peut être lu par des outils du commerce
- √ Notion de Méta-modèle

Qu'est ce qu'un profil ? (1/2)

- √ Mécanismes d'extension standard, *annoter* un schéma.
- √ Compatibilité UML, peut être lu par des outils du commerce
- √ Notion de Méta-modèle
- √ Stéréotypes, valeurs marquées, contraintes

Qu'est ce qu'un profil ? (1/2)

- √ Mécanismes d'extension standard, *annoter* un schéma.
- √ Compatibilité UML, peut être lu par des outils du commerce
- √ Notion de Méta-modèle
- √ Stéréotypes, valeurs marquées, contraintes
- √ Un profil correspond à un ensemble d'extensions et des règles d'utilisation de ces extensions.

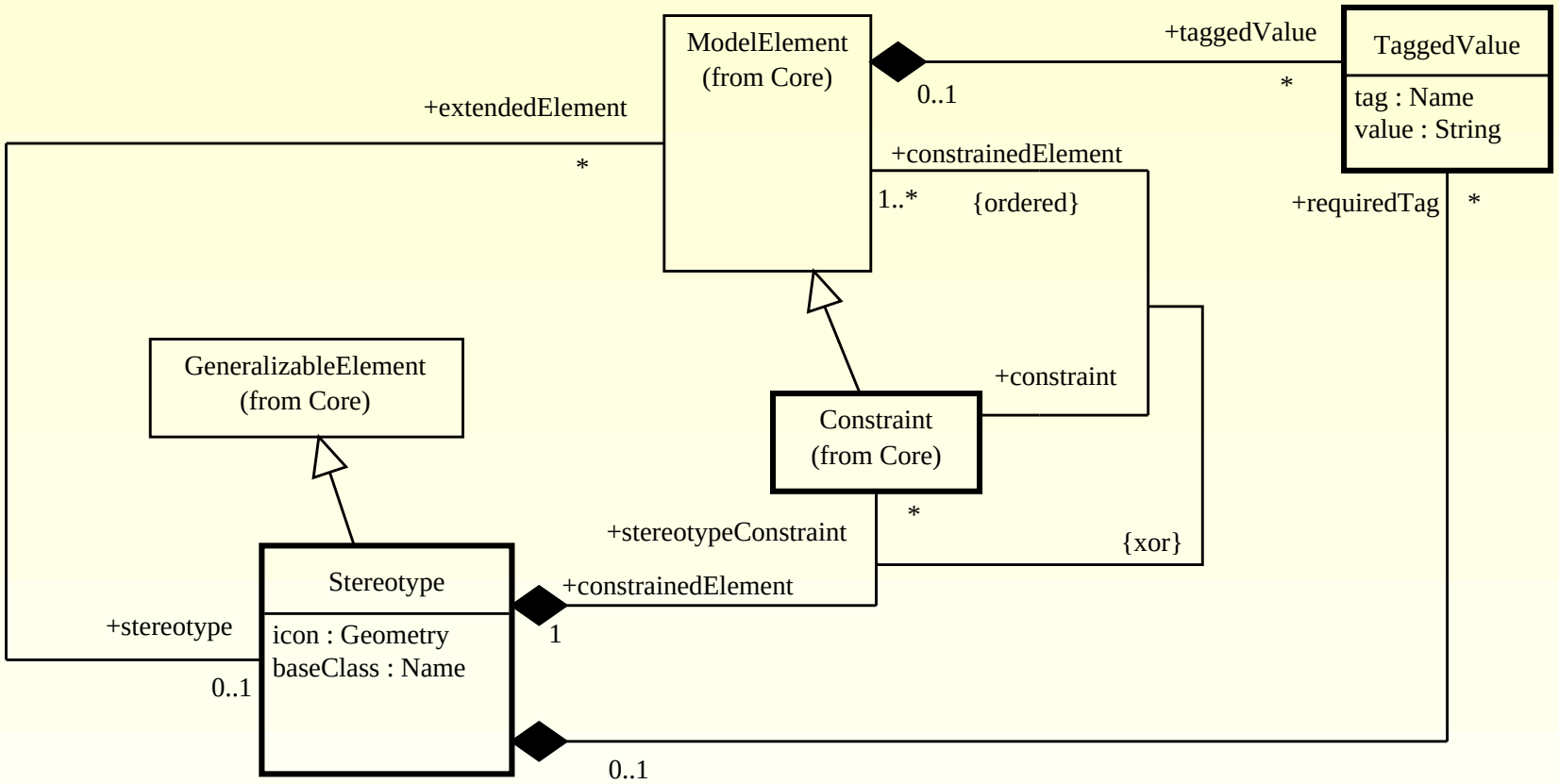
Qu'est ce qu'un profil ? (1/2)

- √ Mécanismes d'extension standard, *annoter* un schéma.
- √ Compatibilité UML, peut être lu par des outils du commerce
- √ Notion de Méta-modèle
- √ Stéréotypes, valeurs marquées, contraintes
- √ Un profil correspond à un ensemble d'extensions et des règles d'utilisation de ces extensions.
- √ Des stéréotypes standards existent : abstract, interface,. . .

Qu'est ce qu'un profil ? (1/2)

- √ Mécanismes d'extension standard, *annoter* un schéma.
- √ Compatibilité UML, peut être lu par des outils du commerce
- √ Notion de Méta-modèle
- √ Stéréotypes, valeurs marquées, contraintes
- √ Un profil correspond à un ensemble d'extensions et des règles d'utilisation de ces extensions.
- √ Des stéréotypes standards existent : abstract, interface,. . .
- √ Des profils standards existent : profil Java, profil EJB, . . .

Qu'est-ce qu'un profil ? (2/2)



Un profil Bases de Données Relationnelles

- √ Le profil standard BDR n'existe pas.
- √ Une ébauche de solution :
  ajout de un stéréotype`Identifiant` (Attribute)
  ajout de contraintes, exemple :

```
// chaque entite (classe) doit posséder au moins une clé.
context Class  :
self.typedFeature->
  select(extensionStereotype.Name=Identifiant).size()>=1
```

Avantages d'un profil

- √ Le schéma UML dispose d'informations supplémentaires liés au domaine.
- √ L'outil peut vérifier la cohérence du schéma
- √ L'outil peut générer du code

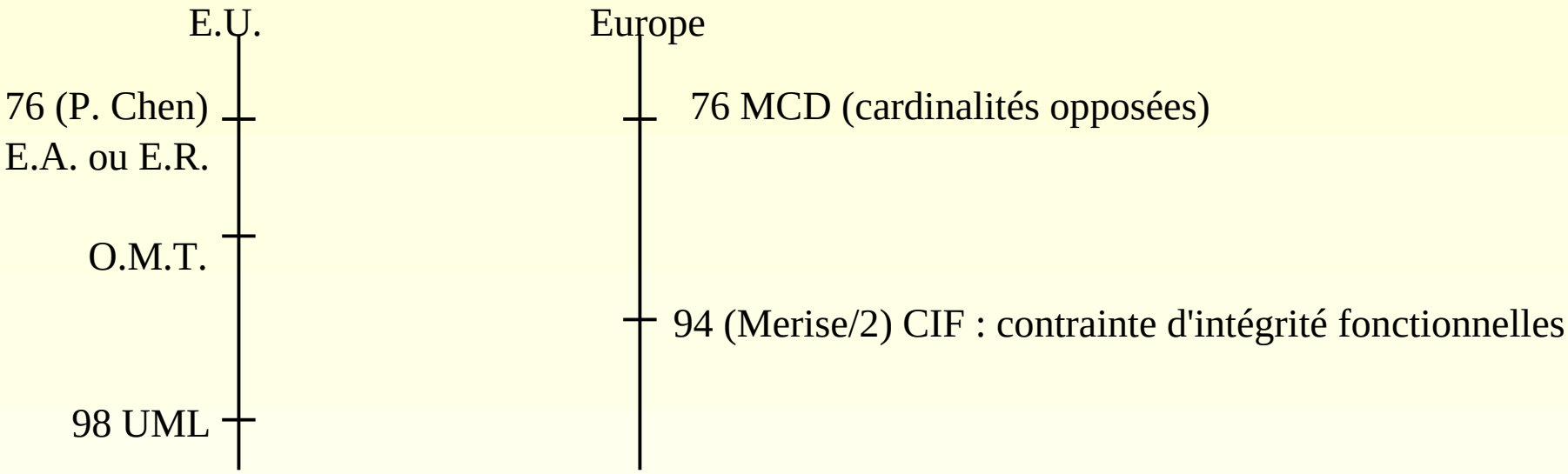
Quelques outils UML

- √ Le plus répandu : Rational Rose (Windows, Solaris)
- √ Softeam Objecteering (`www.softeam.fr`) avec Profile Builder
- √ Argo UML (toute plate-forme (Java)), gratuit
- √ Un éditeur simple : TCM (gratuit)

```
export TCM_HOME=/usr/local/tcm-2.01
export PATH=${PATH}:${TCM_HOME}/bin
tcm &
```

Apports Sémantiques

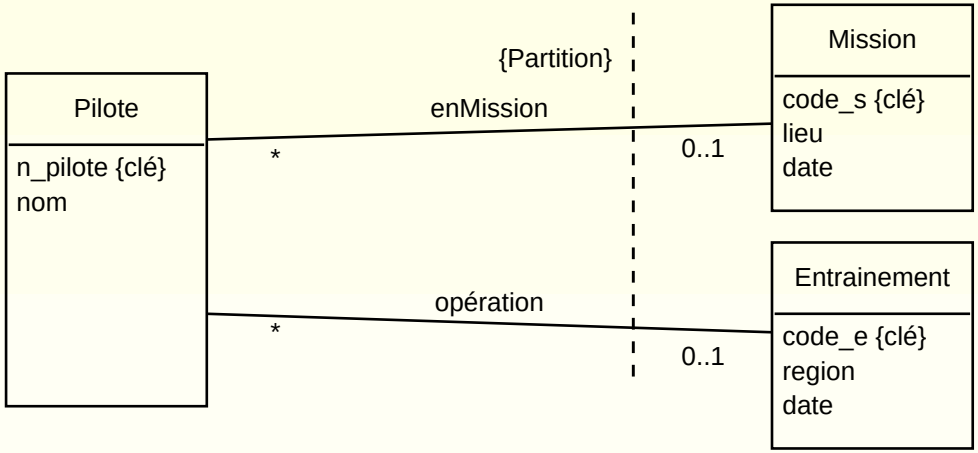
- √ Favoriser la compréhension du schéma
- √ Ces notions existent dans Merise (Merise 2 - 1994)



Les contraintes d'intégrité fonctionnelles (1/5)
Les contraintes de partition

Définition 4. *Toutes les occurrences d'une entité participent à l'une des deux associations, mais pas aux deux, ni à aucune des deux*

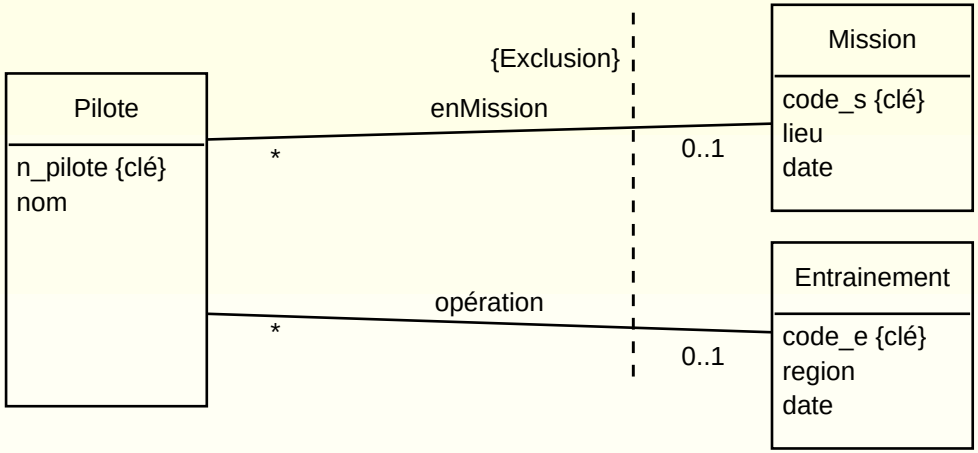
Exemple : des pilotes sont en mission, ou alors en opération d'entrainement, jamais au repos !



Les contraintes d'intégrité fonctionnelles (2/5)
Les contraintes d'exclusion

Définition 5. *Toutes les occurrences d'une entité peuvent participer à l'une des deux associations, mais pas aux deux à la fois*

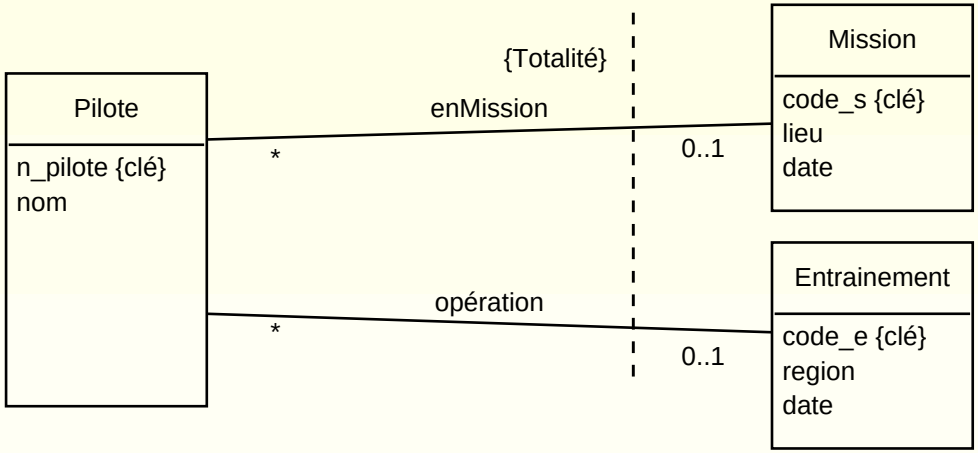
Exemple : des pilotes sont en mission, ou alors en opération d'entrainement, peut être au repos !



Les contraintes d'intégrité fonctionnelles (3/5)
Les contraintes de totalité

Définition 6. *Toutes les occurrences d'une entité participent au moins à une association*

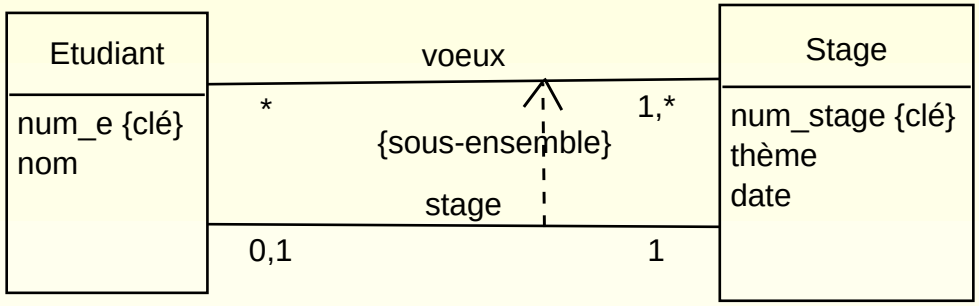
Exemple : un pilote peut être affecté à une mission et à un entrainement, tous les pilotes participent au moins à l'une des associations



Les contraintes d'intégrité fonctionnelles (4/5)
Les contraintes d'inclusion

Définition 7. *Toutes les occurrences d'une association doivent être incluses dans les occurrences d'une autre association*

Exemple : Un étudiant est assuré d'effectuer un stage qu'il a voulu ! :-)



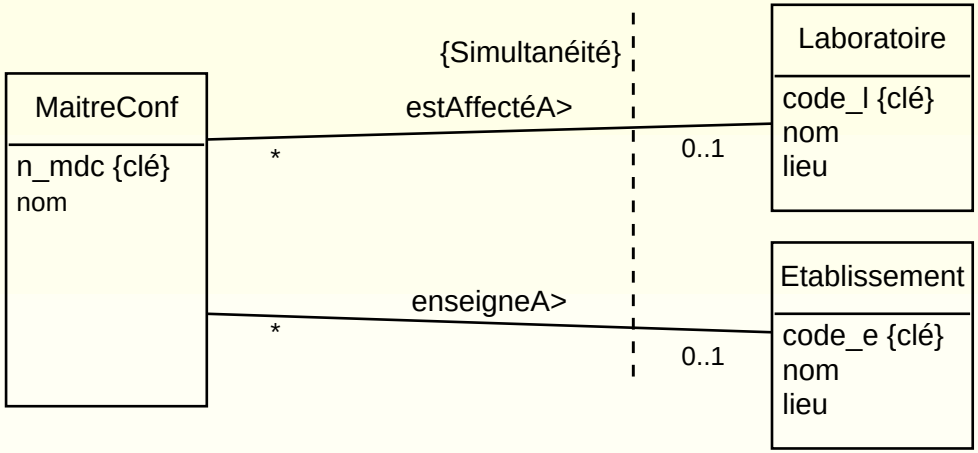
Les contraintes d'intégrité fonctionnelles (5/5)
Les contraintes de simultanéité

Définition 8. *Si une occurrence d'une entité participe à l'une des deux associations, elle participe alors également à l'autre.*

Exemple : Un maître de conférences est affecté à un établissement d'enseignement et à un laboratoire de recherche

REMARQUE:

Cette contrainte n'est pas standard UML

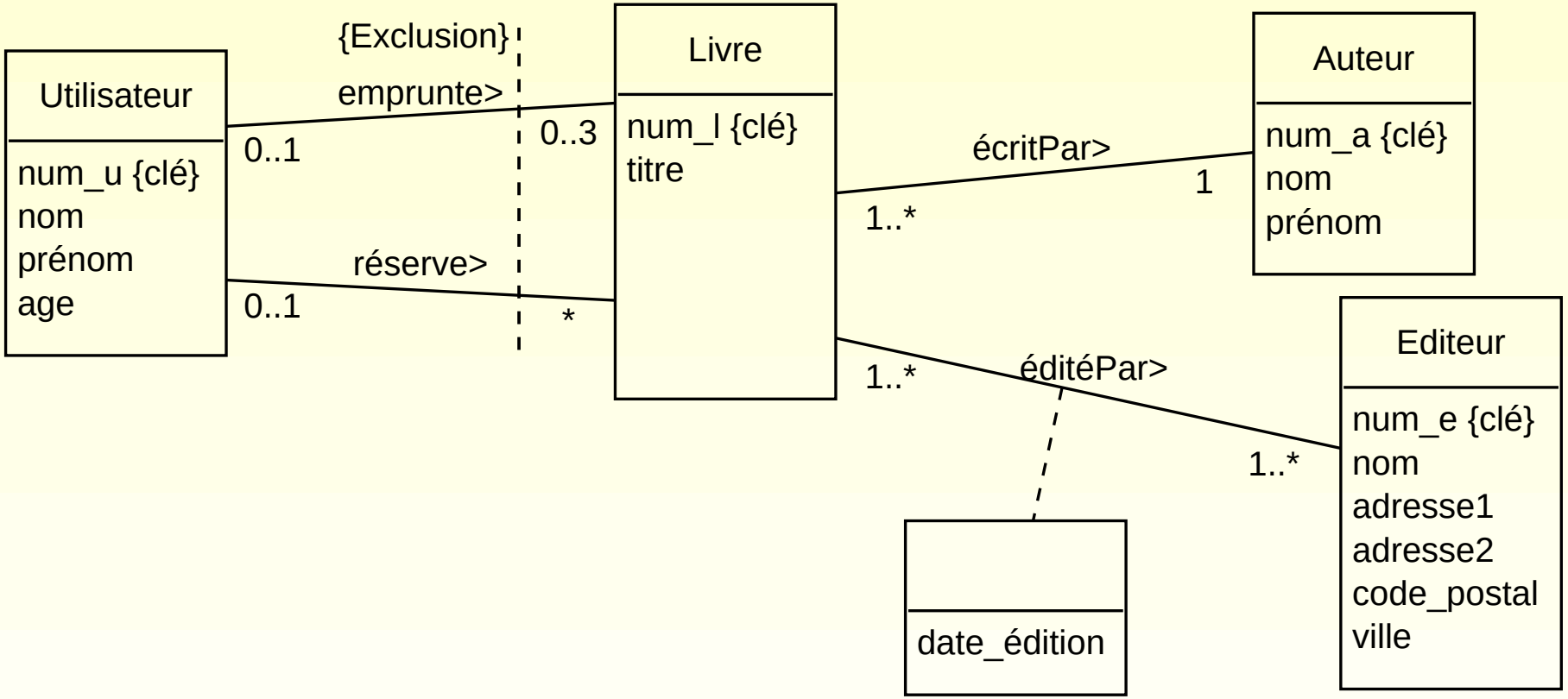


Problème classique - Gestion d'une bibliothèque

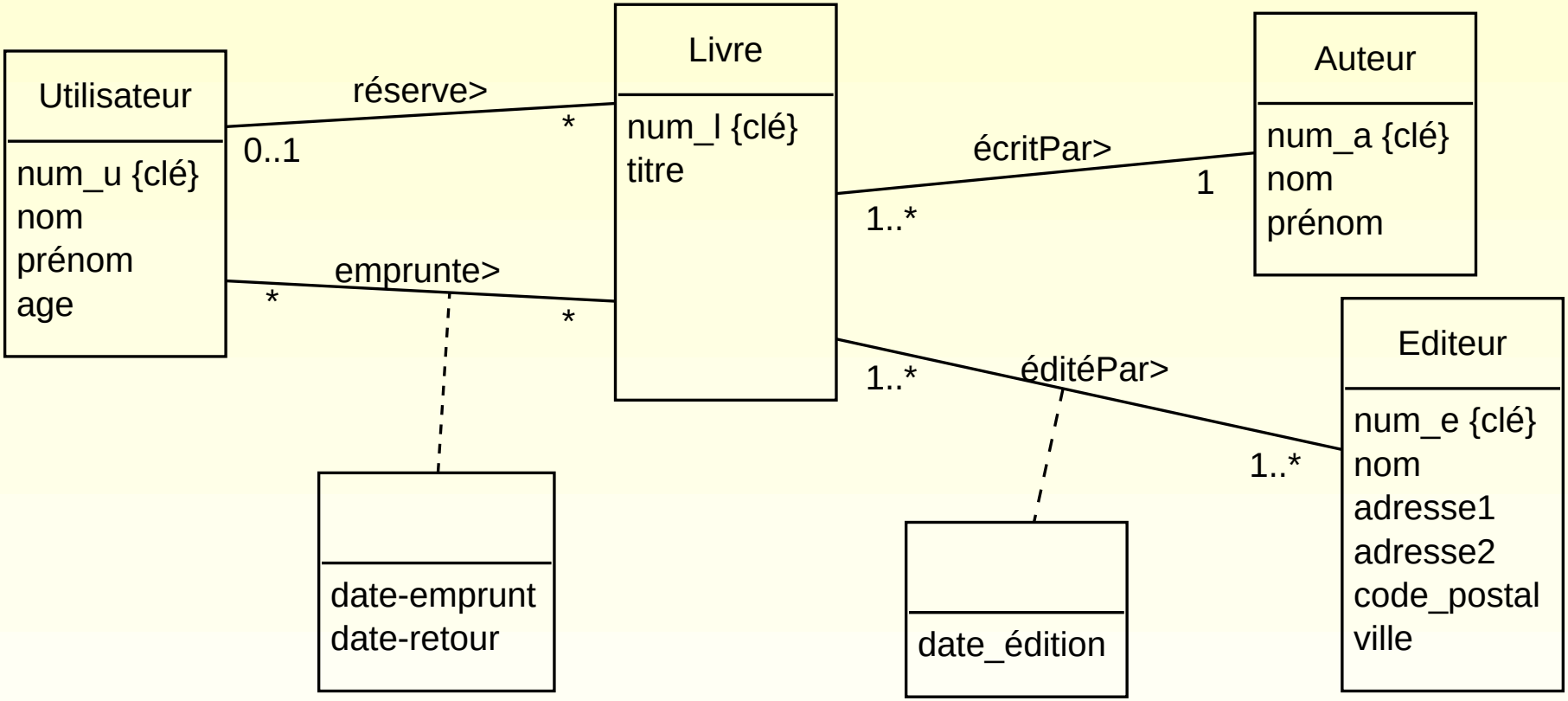
Soit le cahier des charges suivants *volontairement flou* pour gérer une bibliothèque.
Un livre est caractérisé par son titre, son auteur, ses éditeurs. Pour chaque livre édité, on veut connaître la date d'édition.
Un auteur est caractérisé par son nom, prénom.
Un éditeur est caractérisé par son nom et son adresse.
Un utilisateur est caractérisé par son nom, prénom et age.
Le but de cette gestion est de gérer les emprunts de livres (maximum 3 livres par utilisateur) et la possibilité de réserver des livres.

$\Longrightarrow$ Proposez un schéma conceptuel

Une solution possible



Autre solution - gestion de l'historique des emprunts



Une dernière solution, plus réaliste