Modélisation UML pour les Bases de données

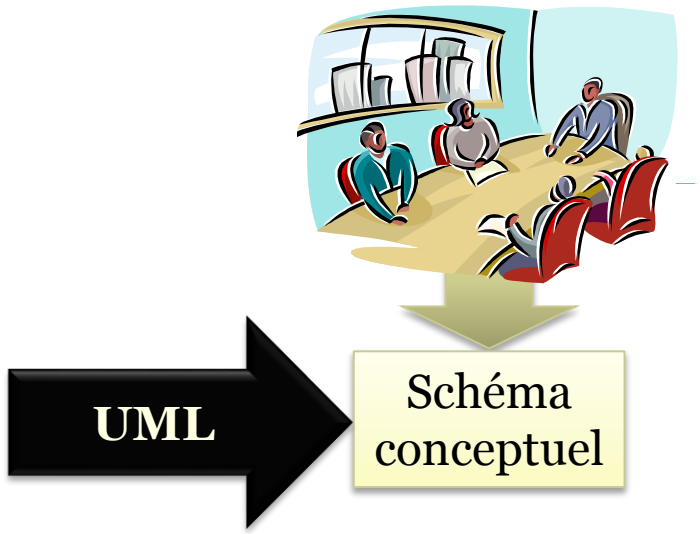
Avertissement

- Nous utilisons dans ce cours le langage UML, nous aborderons cette méthode que du point vue modélisation des bases de données, par conséquent les principaux concepts fondateurs (modélisation objet) de cette méthode seront occultés où du moins survolés.

Plan de cours

- Conception des BDD : les différents niveaux
- Pourquoi modéliser ?
- Présentation du formalisme UML
- Bien modéliser

Conception des BDD : les différents niveaux

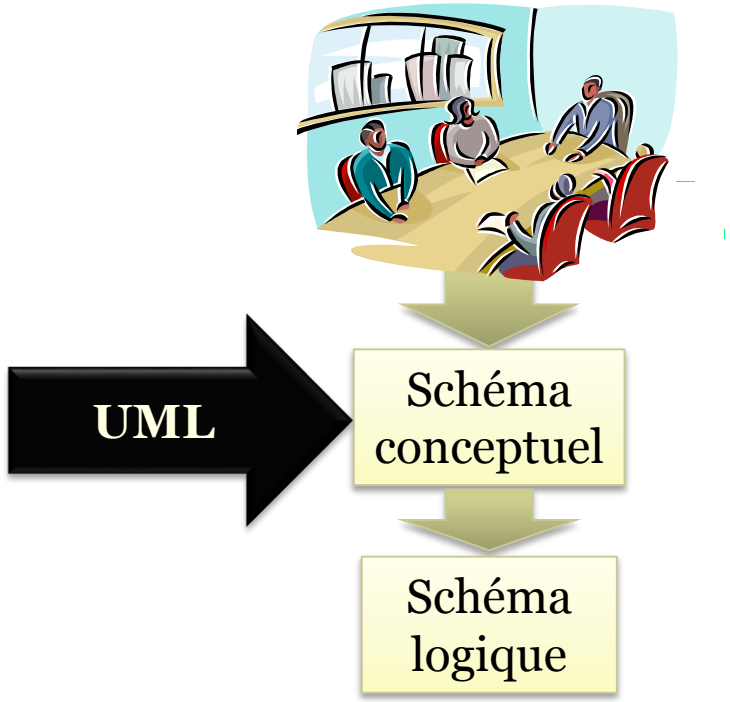


1. Le niveau conceptuel

C'est le niveau d'abstraction le plus élevé.
On cherche à structurer les données pérennes du domaine d'étude de notre application.

Comment seront stockées les données, sur quel type machine, sous quel de système d'exploitation et sur quel SGBD seront implémenté les données nous importe pas dans cette partie.

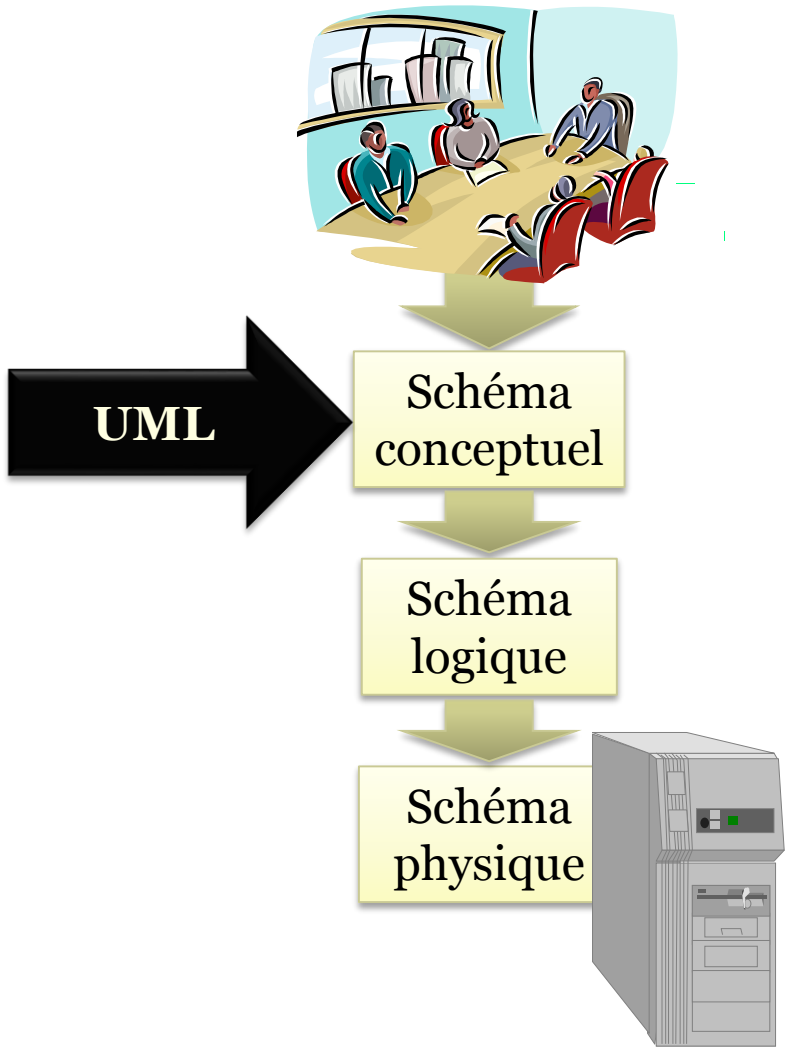
Conception des BDD : les différents niveaux



1. Le niveau conceptuel
2. Le niveau logique

On va définir les tables nécessaires à la prise en charge des besoins de stockage de notre application tout en préservant la pérennité et l'intégrité des données. Cette étape est directement liée à la précédente (niveau conceptuel)

Conception des BDD : les différents niveaux



1. Le niveau conceptuel
2. Le niveau logique
3. Le niveau Physique

Là on est au plus prés de la machine, le choix du SGBD est effectué et l'implémentation sur une machine réalisé.

Notre base de données est prête à *"vivre"*

Conception des BDD : les différents niveaux

UML

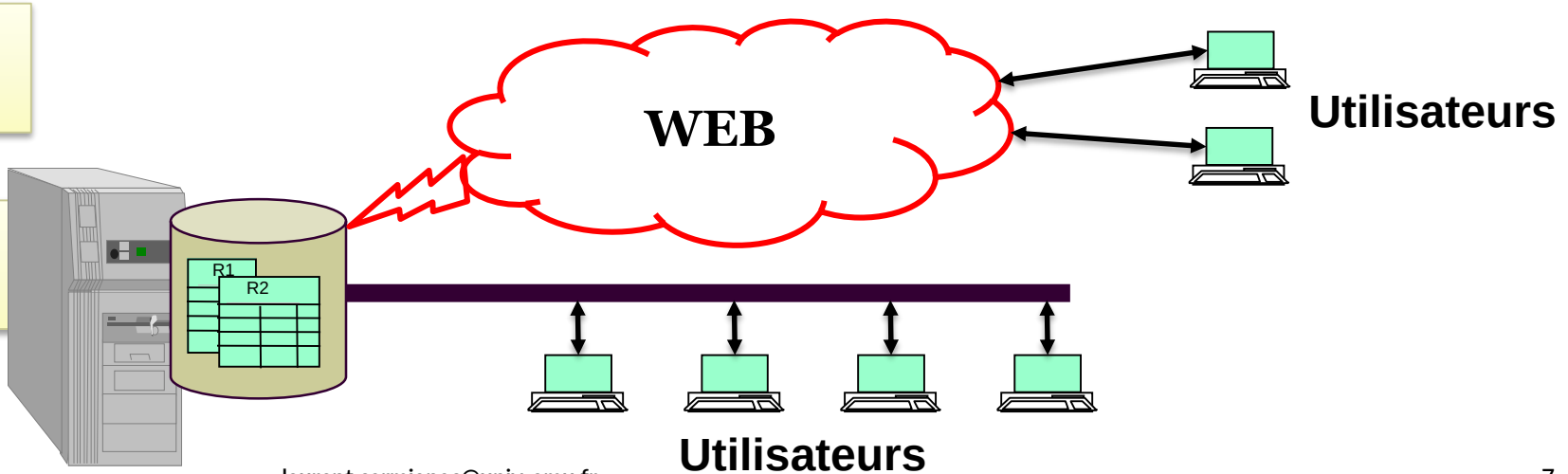
Schéma conceptuel

Schéma logique

Schéma physique

1. Le niveau conceptuel
2. Le niveau logique
3. Le niveau Physique
4. Le niveau externe

Il s'agit de la présentation des données aux utilisateurs via des applications du marché ou développées spécialement pour nos besoins

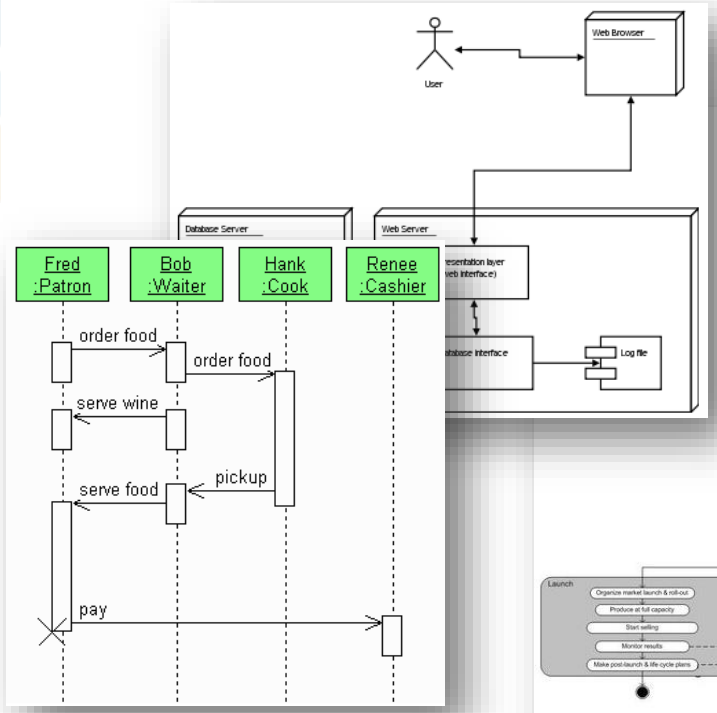


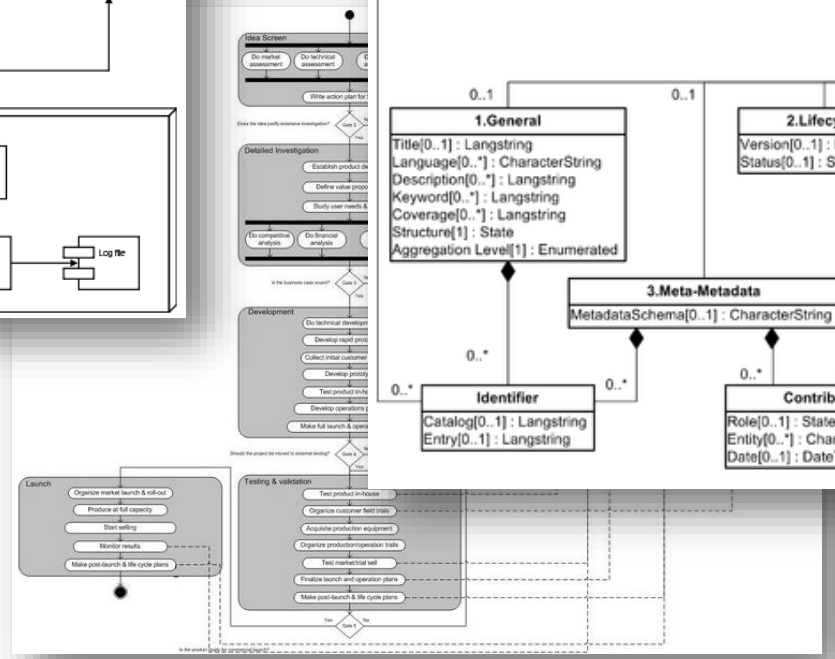
Pourquoi modéliser ?

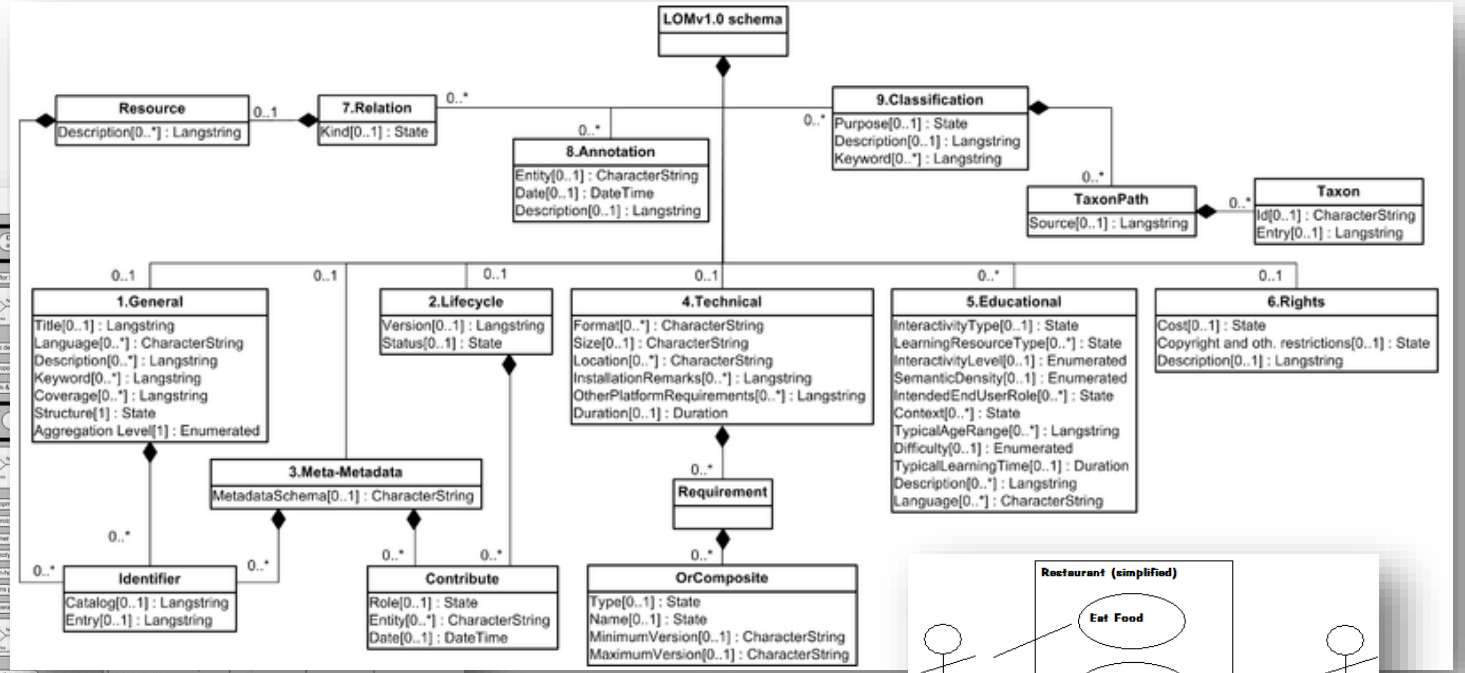
- Etape primordiale dans la création d'une application et dans sa réalisation
  - Elle permet de structurer ses idées et de les documenter,
- La modélisation permet de se concentrer sur les objectifs, les besoins de l'entreprise et de ses acteurs
  - Elle permet de :
    - Minimiser la complexité,
    - Maximiser l'évolutivité,
    - Augmenter la robustesse
- Pour modéliser, il est important d'établir et de fixer une notation.
    - Nous utiliserons le langage UML (Unified Modeling Language)
    - UML est destiné à la modélisation objet et couvre toutes les phases d'un projet, il nous permettra de spécifier et de structurer les données pérennes des systèmes d'information

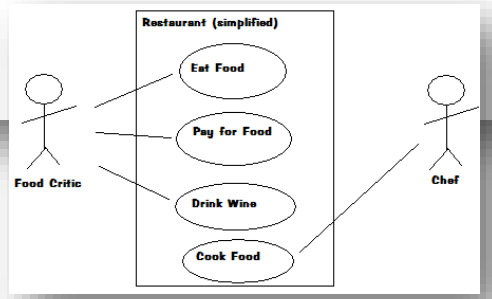
Présentation du formalisme UML

- UML comporte plus d'une dizaine de diagrammes standards qui se répartissent en deux catégories.









Présentation du formalisme UML

- Les diagrammes structurels ou statiques

  - Diagrammes de classe (*Class diagram*)
  - Diagrammes d'objet (*Object diagram*)
  - Diagrammes de composant (*Component diagram*)
  - Diagrammes de déploiement (*Deployment diagram*)
  - Diagrammes de paquetage (*Package diagram*)
  - Diagramme de structure composite (*Composite structure diagram*)

Présentation du formalisme UML

- Les diagrammes comportementaux ou diagrammes dynamiques

- Diagrammes de cas d'utilisation (*Use case diagram*)
- Diagrammes d'activité (*Activity diagram*)
- Diagrammes d'état-transition (*State machine diagram*)
- Diagrammes d'interaction (*Interaction diagram*)
  - Diagrammes de séquence (*Sequence diagram*)
  - Diagrammes de communication (*Communication diagram*)
  - Diagramme global d'interaction (*Interaction overview diagram*)
  - Diagramme de temps (*Timing diagram*)

Avertissement

- Dans cette partie nous nous intéresserons uniquement au diagramme de classes qui permettra de modéliser la structure et les liens entre les données des systèmes d'information

Présentation du formalisme UML

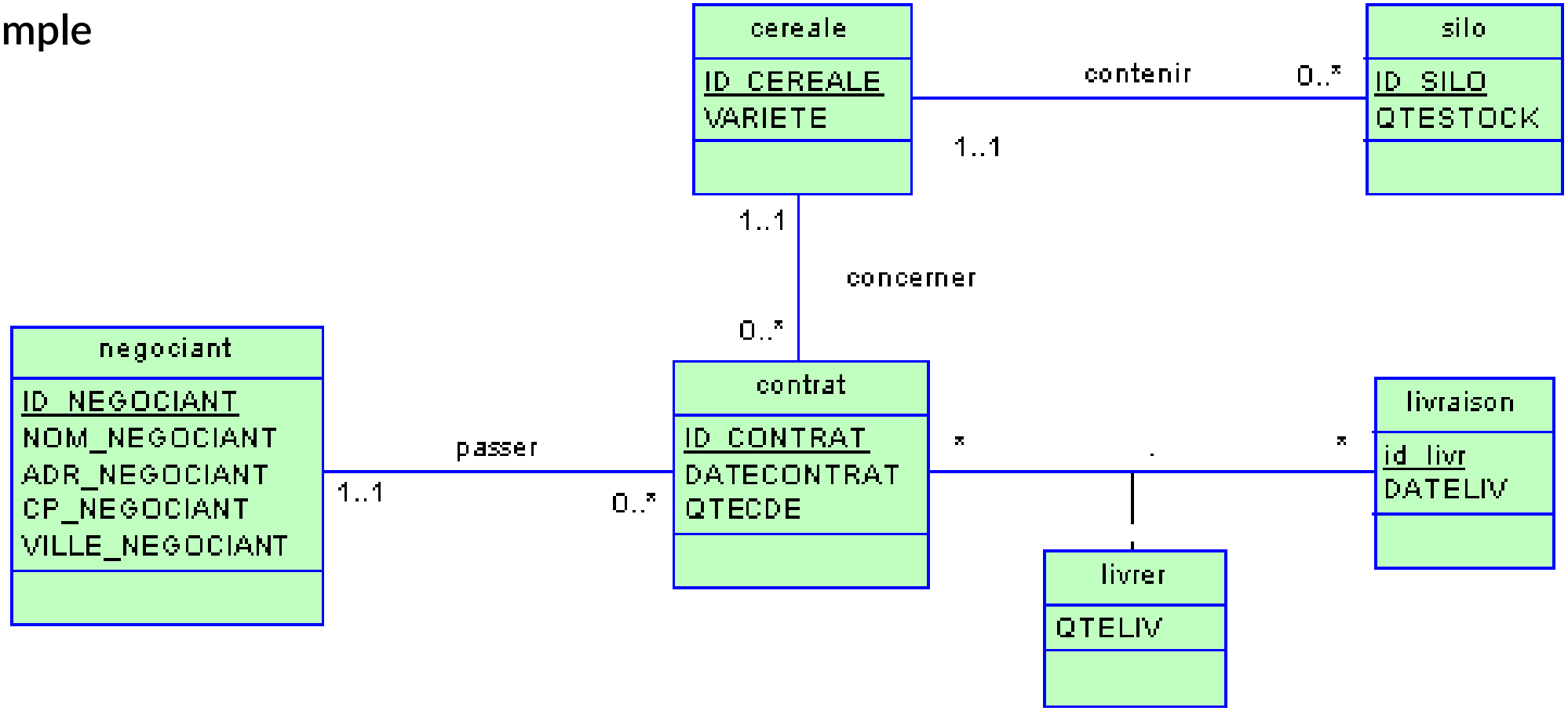
- Diagramme de classes pour les BDD
  - C'est la représentation de l'ensemble des données persistantes du système d'information étudié, sans tenir compte des aspects techniques, économiques, de mémorisation, d'accès et sans se référer aux conditions d'utilisation par tel ou tel traitement.
  - Il est composé de classes et d'associations

Présentation du formalisme UML

- Diagramme de classes
  - Objectif du diagramme de classes
    - Il consiste à identifier, à décrire des informations du domaine étudié et à identifier les liens qui existent entres elles.
  - Démarche du diagramme de classes
    - On distingue deux attitudes, correspondant en fait à la connaissance du domaine étudié :
      - une démarche déductive qui s'appuie sur l'existence préalable d'une liste d'informations à structurer ; le cahier des charges est décortiqué en informations élémentaires ;
      - Une démarche inductive qui cherche à mettre rapidement en évidence les différents concepts évoqués dans le cahier des charges, puis à les décrire par des informations.

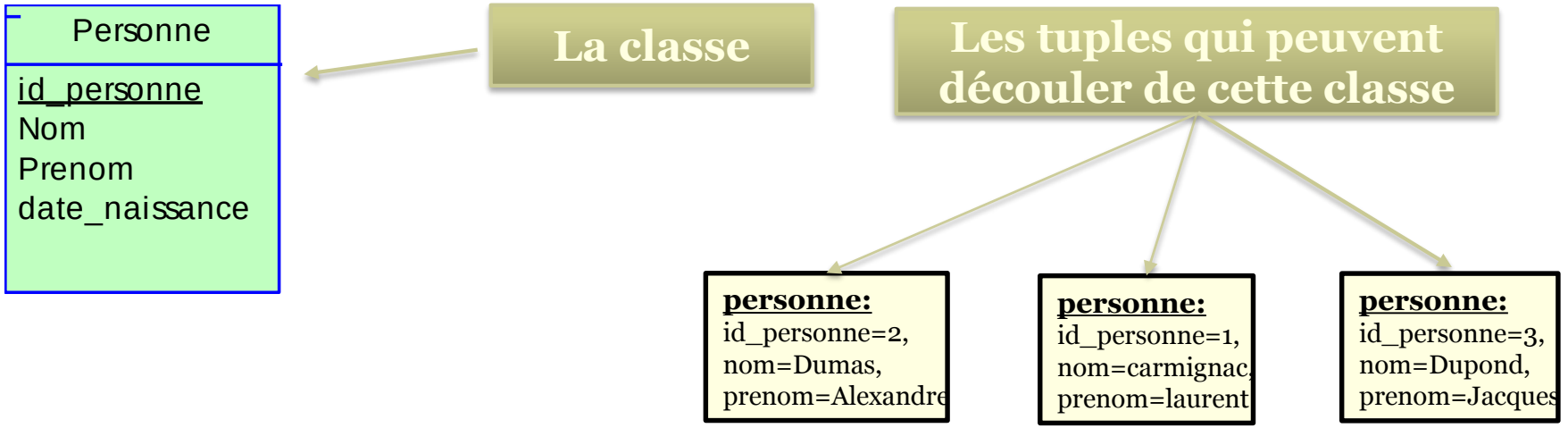
Présentation du formalisme UML

- Exemple



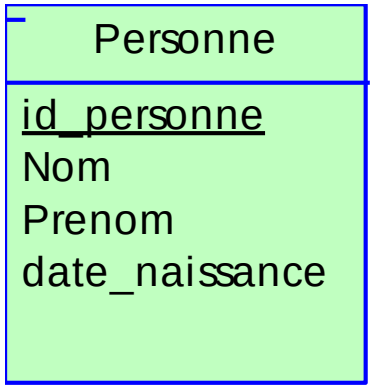
Présentation du formalisme UML

- Classe
  - Description "statique" d'un ensemble d'objets (ou tuples) concrets ou abstraits de même nature
  - Une classe est décrite par ses attributs, méthodes *(n'apparaissent pas dans la conception des bdd)* et contraintes
  - Exemples : Etudiant, Employe, Produit, …
  - Elle permet de définir des tuples (ou enregistrements) possédant ces caractéristiques.



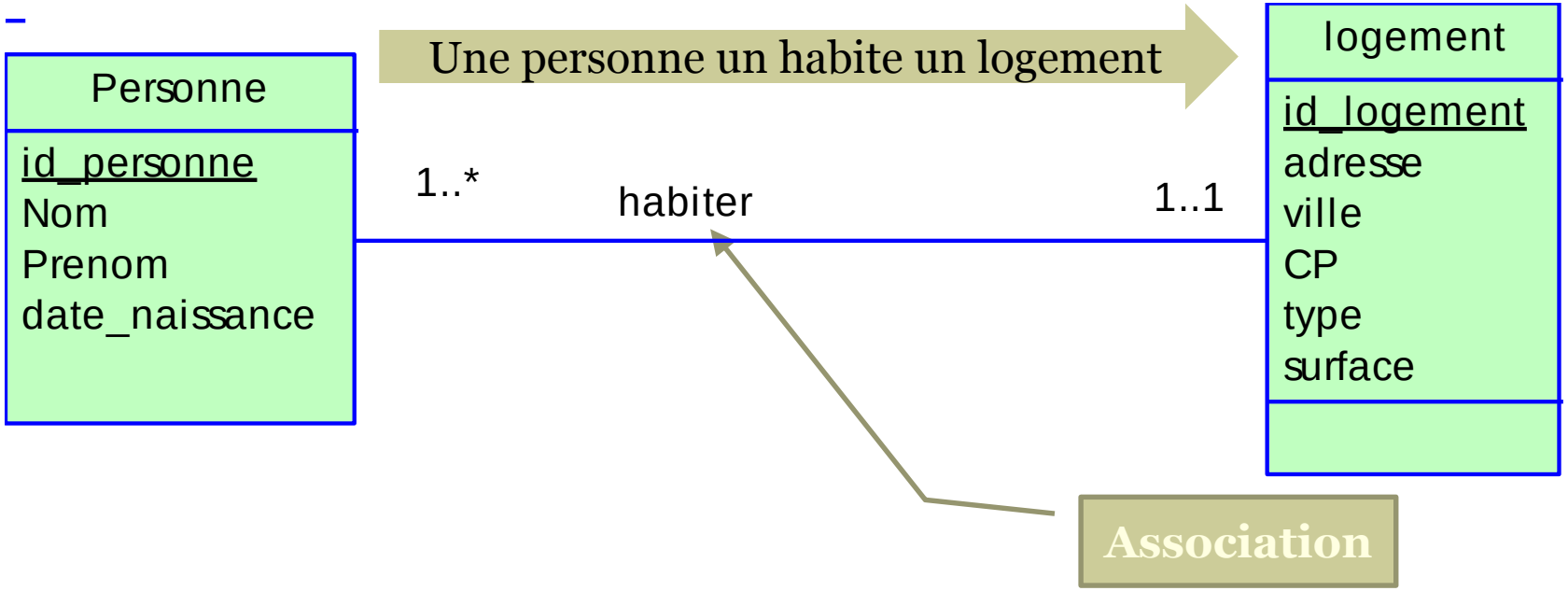
Présentation du formalisme UML

- Les classes (entité)
  - Clé primaire d'une entité -la base de la relation-
    - C'est un attribut (ou un ensemble d'attributs) qui permet de distinguer un élément de l'entité de manière unique et sans aucune ambiguïté par rapport à l'ensemble des autres éléments.
    - Cette notion de clé primaire est utilisée uniquement pour la modélisation des BDD, UML ne prévoit pas de formalisme particulier pour distinguer un attribut clé primaire, nous soulignerons l'attribut clé primaire dans la classe.
    - Par exemple la clé primaire de la classe "*Personne*" pourrait être le nom. Mais comme le cas d'homonymie est assez fréquent, alors cet attribut constitue une mauvaise clé en général. On pourrait lui attribuer une clé numérique (auto incrémentée) ou son N° de Sécurité Sociale.



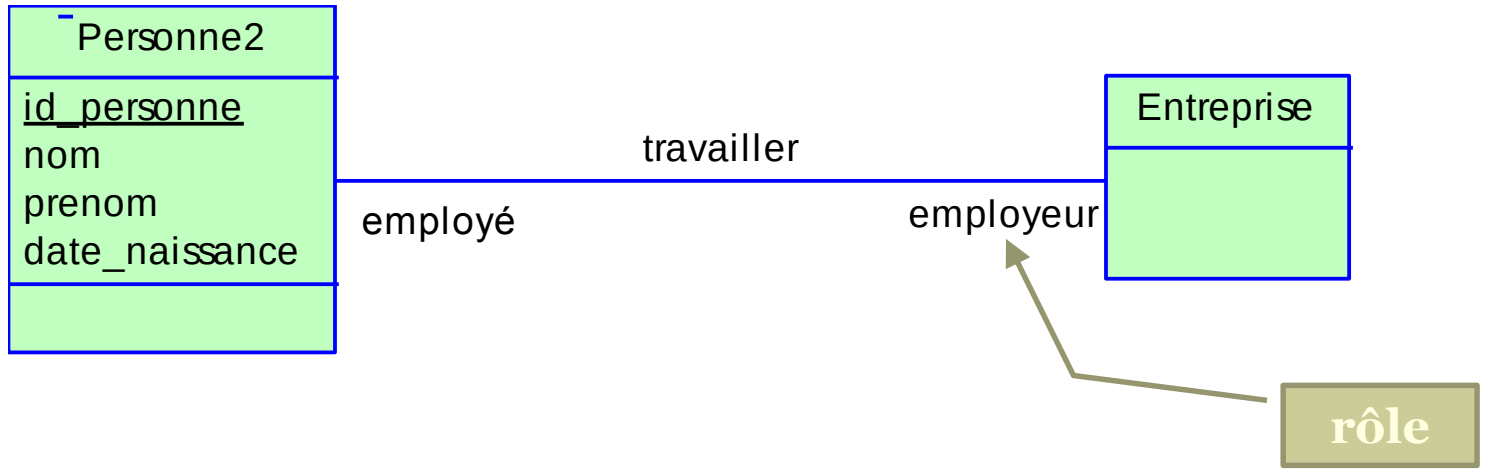
Présentation du formalisme UML

- Les associations
  - Les associations déterminent les interactions entres les classes, le nom de l'association spécifie cette interaction. Elles sont souvent explicites dans l'expression des besoins, elles correspondent la plupart du temps à des verbes.



Présentation du formalisme UML

- Les associations
  - Pour être plus précis on peut recourir aux rôles.

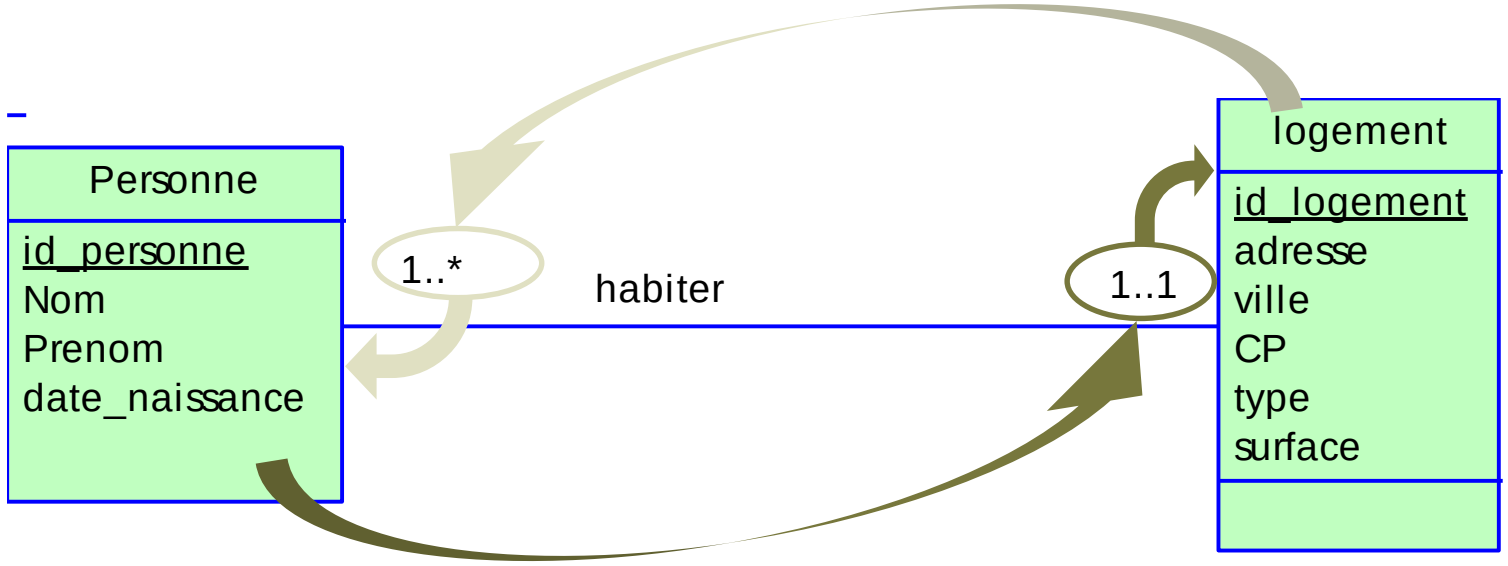


  - Les personnes ont un rôle d'employé d'une entreprise dans une relation de travail
  - Les entreprises ont un rôle d'employeur des personnes dans une relation de travail

Présentation du formalisme UML

- Cardinalité des associations
  - Les cardinalités, au sens arithmétique du terme, permettent de dénombrer les éléments de la classe d'arrivée en relation avec un élément de la classe de départ, et vice versa.

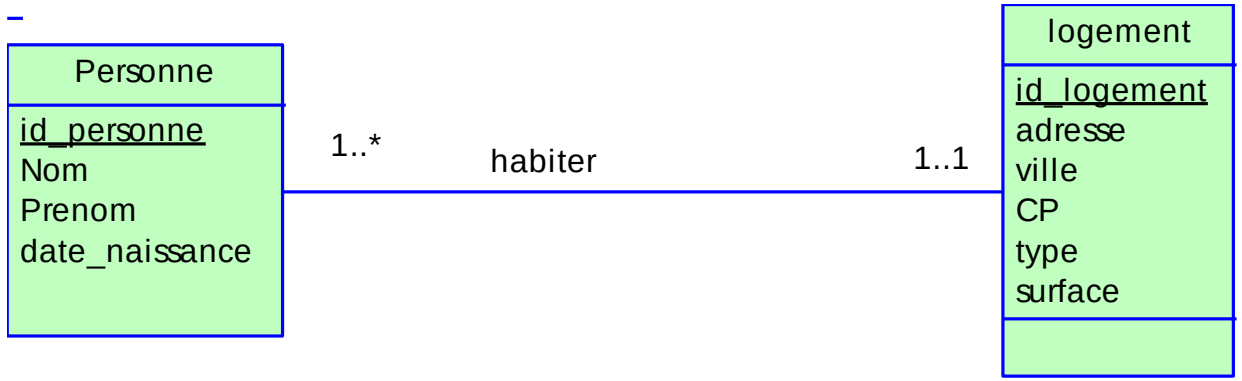
<< 1 logement est occupé par 1 ou plusieurs (*) personnes

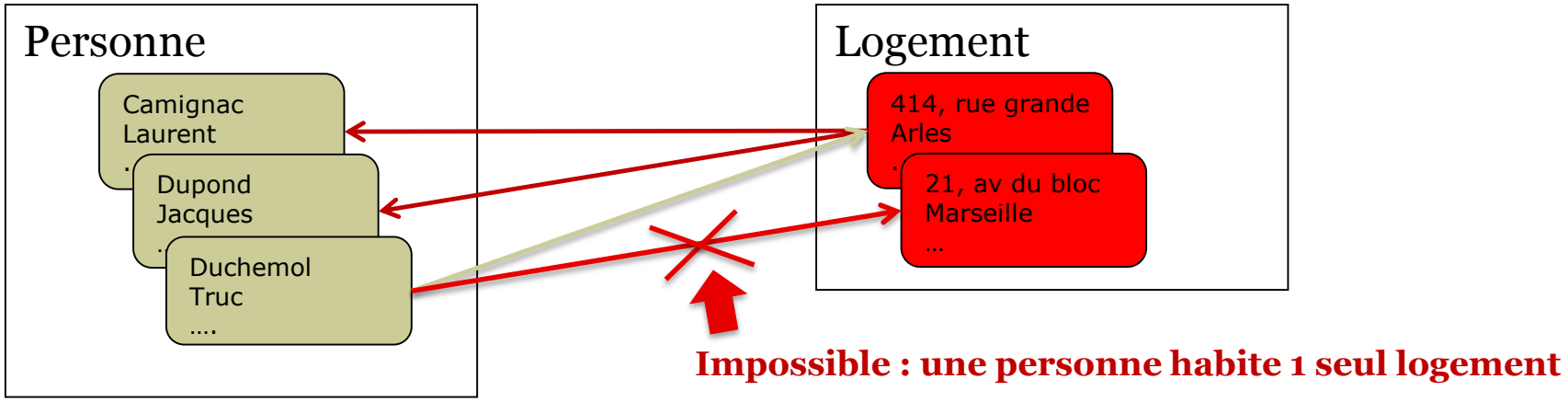


1 personne habite 1 et 1 seul logement >>

Présentation du formalisme UML

- Cardinalité des associations
  - Les liens possibles





Présentation du formalisme UML

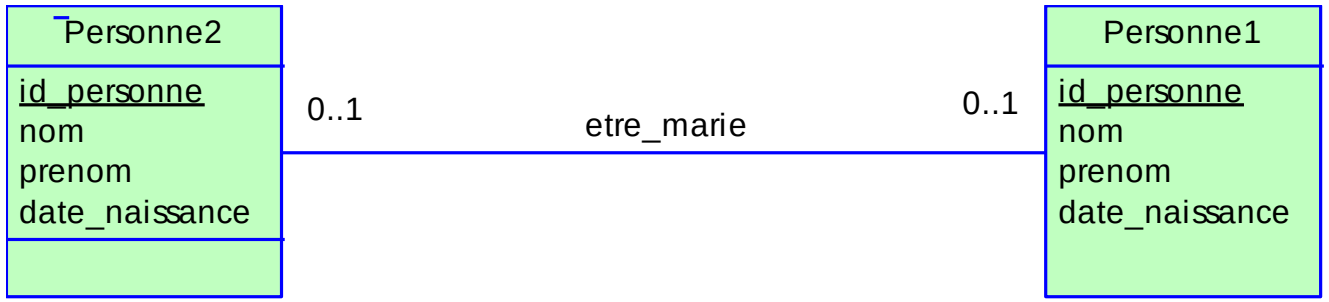
- Cardinalité des associations
  - Les cardinalités possibles

    - 0..1 : 0 minimum à 1 maximum
    - 0..* : 0 minimum à plusieurs maximum
    - 1..1 : strictement 1
    - 1..* : 1 minimum à plusieurs maximum
    - * : plusieurs (plus que 0)

    - 3 : obligatoirement 3
    - 1..4 : 1 minimum à 4 maximum
    - etc

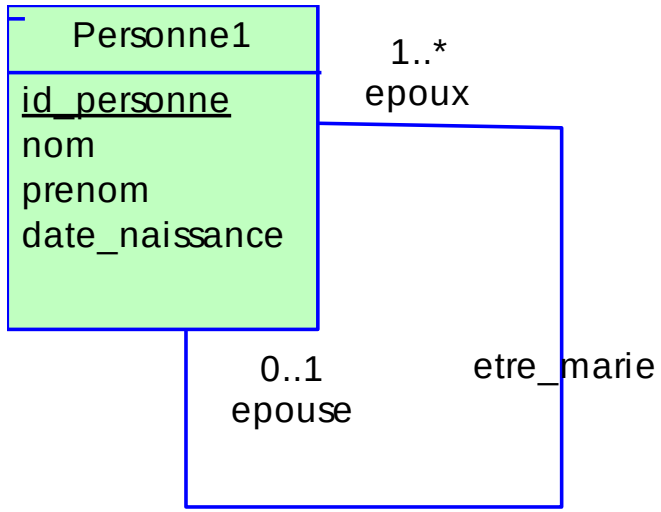
Présentation du formalisme UML

- Associations réflexive
  - Imaginons que vous vouliez modéliser le fait qu'une personne soit marié à une autre personne, la première chose qui vous vient à l'esprit est de créer deux classes et de les relier ensemble
    - Il y a un problème de redondance de classe, en effet nous avons deux classes qui ont les mêmes propriétés, mais alors comment faire?



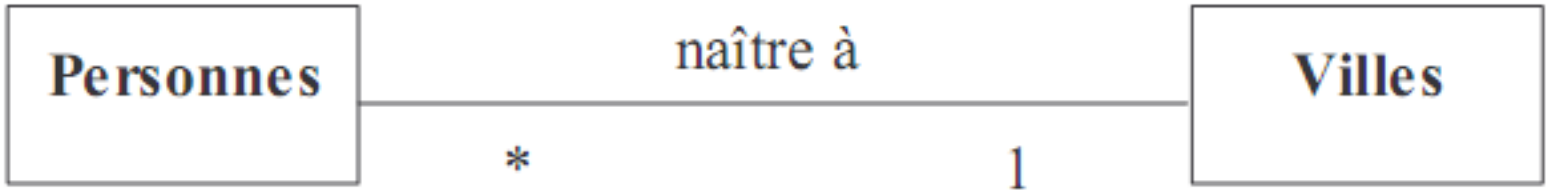
Présentation du formalisme UML

- Associations réflexive
  - Nous utiliserons l'association réflexive qui permet de mettre en relation une classe avec elle même.
    - L'utilisation de rôle est alors fortement conseillé



Présentation du formalisme UML

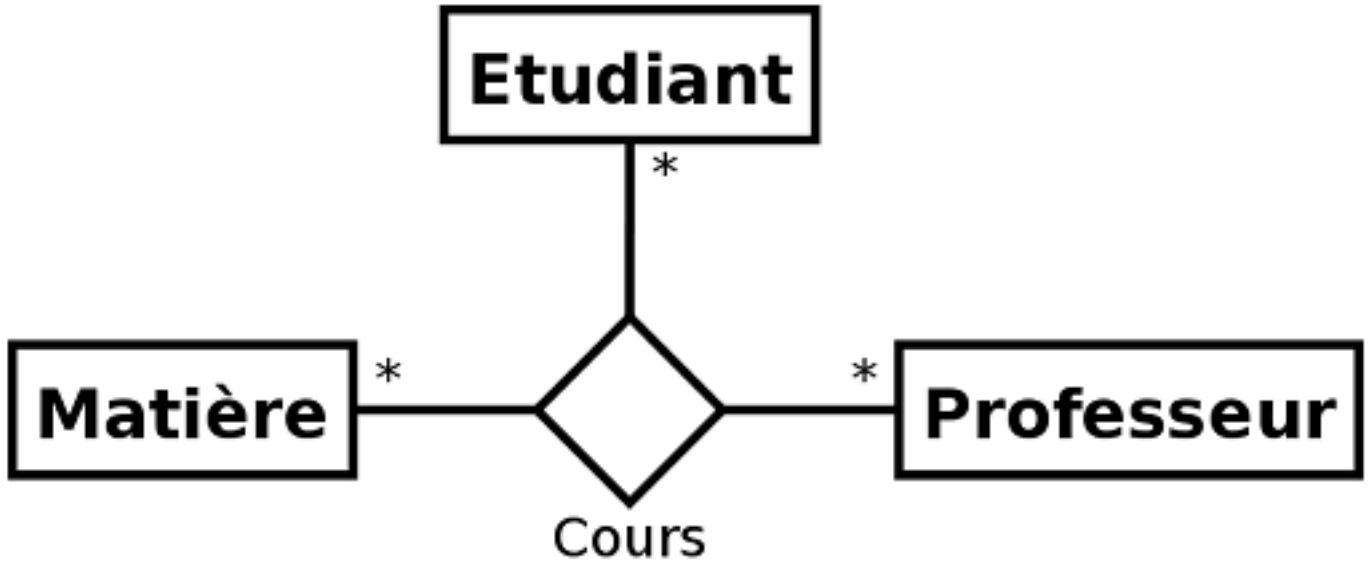
- Les associations binaires, ternaires….
  - On définit l'arité[1] de l'association comme étant le nombre de classes participant à l'association. Nous n'avons vu pour le moment que des associations binaires.



[1] *Wikipédia :En mathématiques, l'arité d'une fonction ou d'un opérateur est le nombre d'arguments ou d'opérandes qu'elle requiert.*

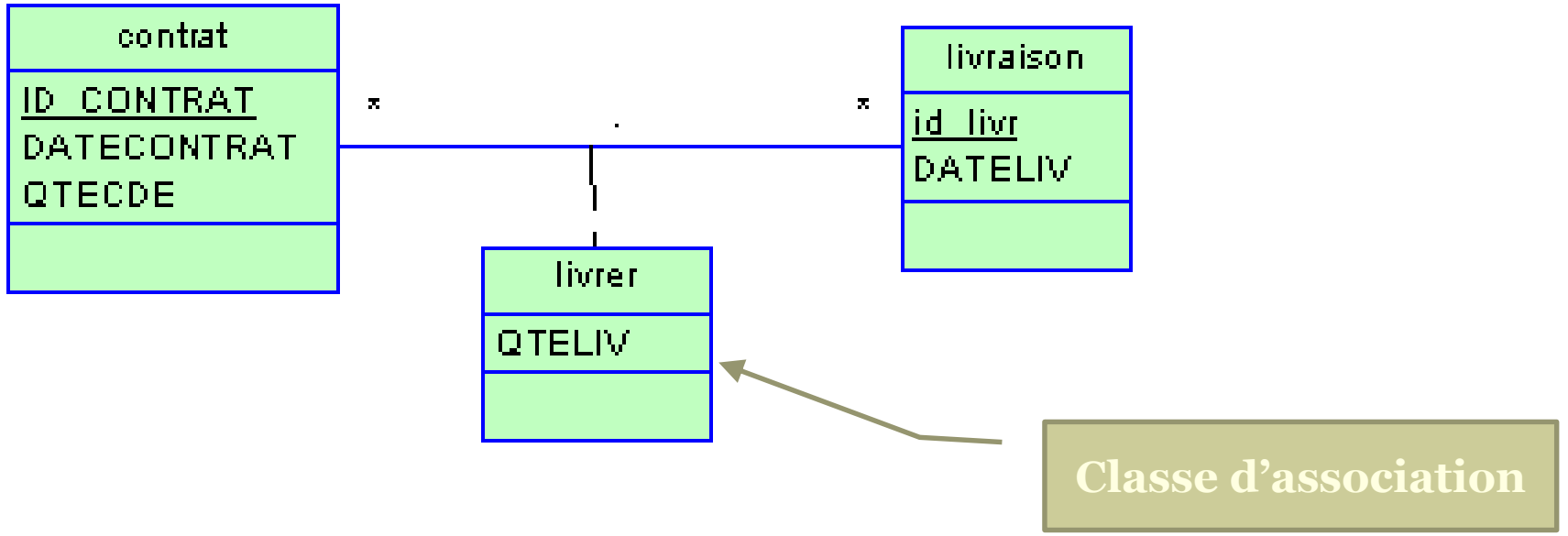
Présentation du formalisme UML

- Les associations binaires, ternaires….
  - Exemple d'association ternaire



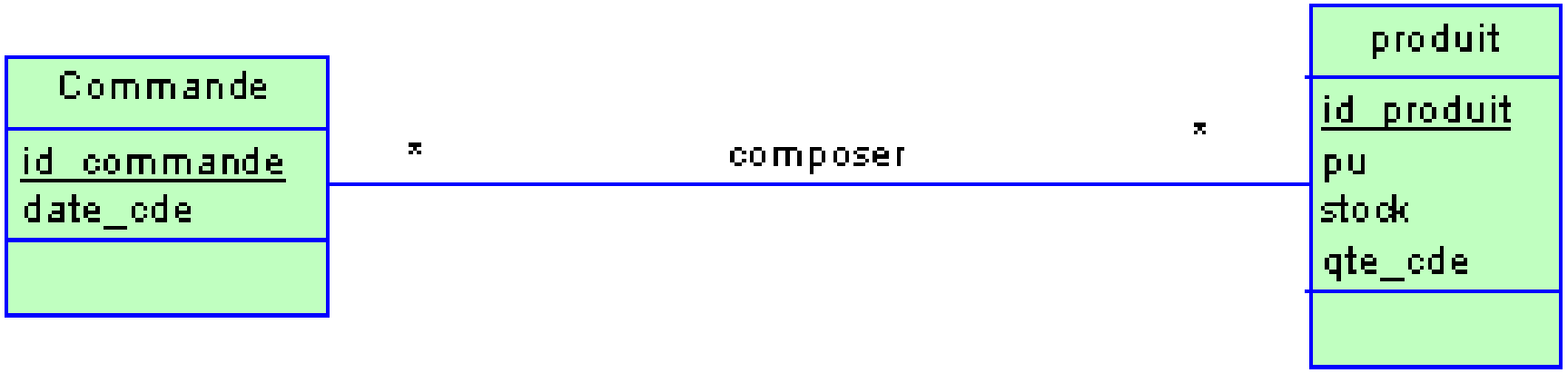
Présentation du formalisme UML

- Les classes d'associations
  - Il est parfois utile de pouvoir ajouter des informations dans une association, par exemple lorsque l'état d'un attribut dépend des deux classes que l'association lie.
  - Ces informations seront représentées dans une classe d'association directement reliée à l'association par un trait pointillé



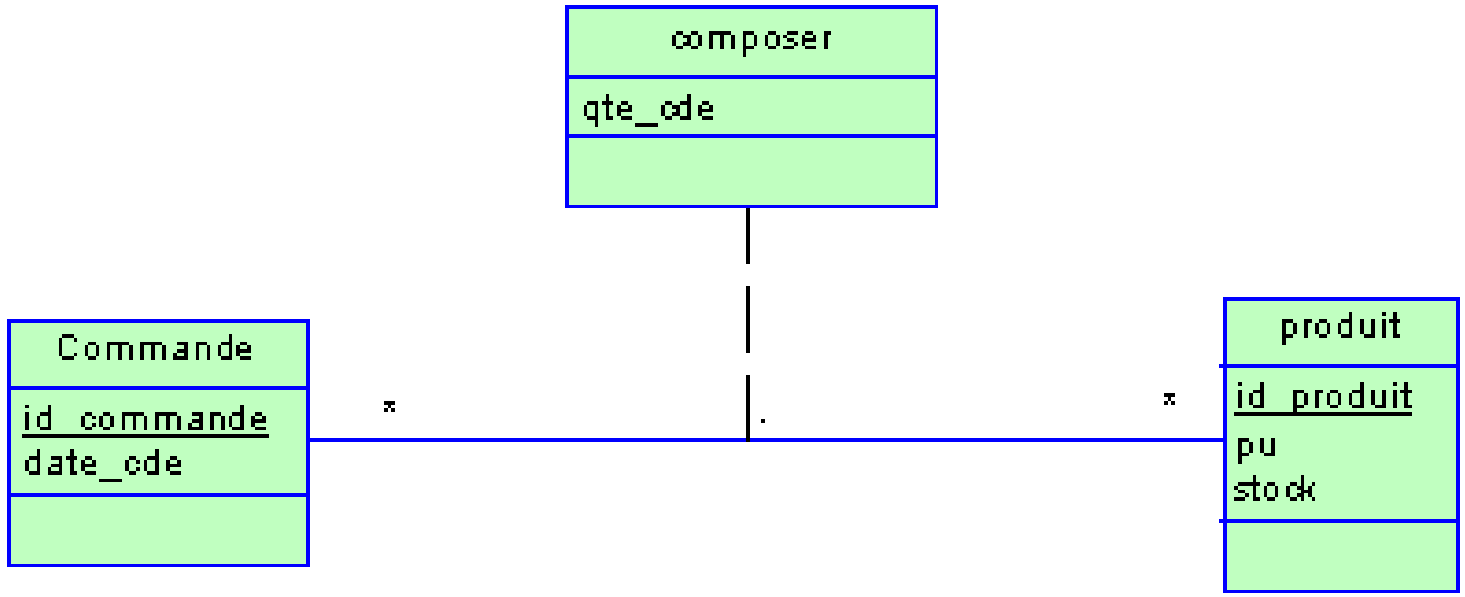
Présentation du formalisme UML

- Les classes d'associations
  - Prenons l'exemple de la quantité d'un produit dans une commande.
    - Dans ce cas la quantité commandée (qte_cde) ne dépend que du produit, cela revient à avoir une quantité identique du même produit dans toutes les commandes, un peu gênant non?



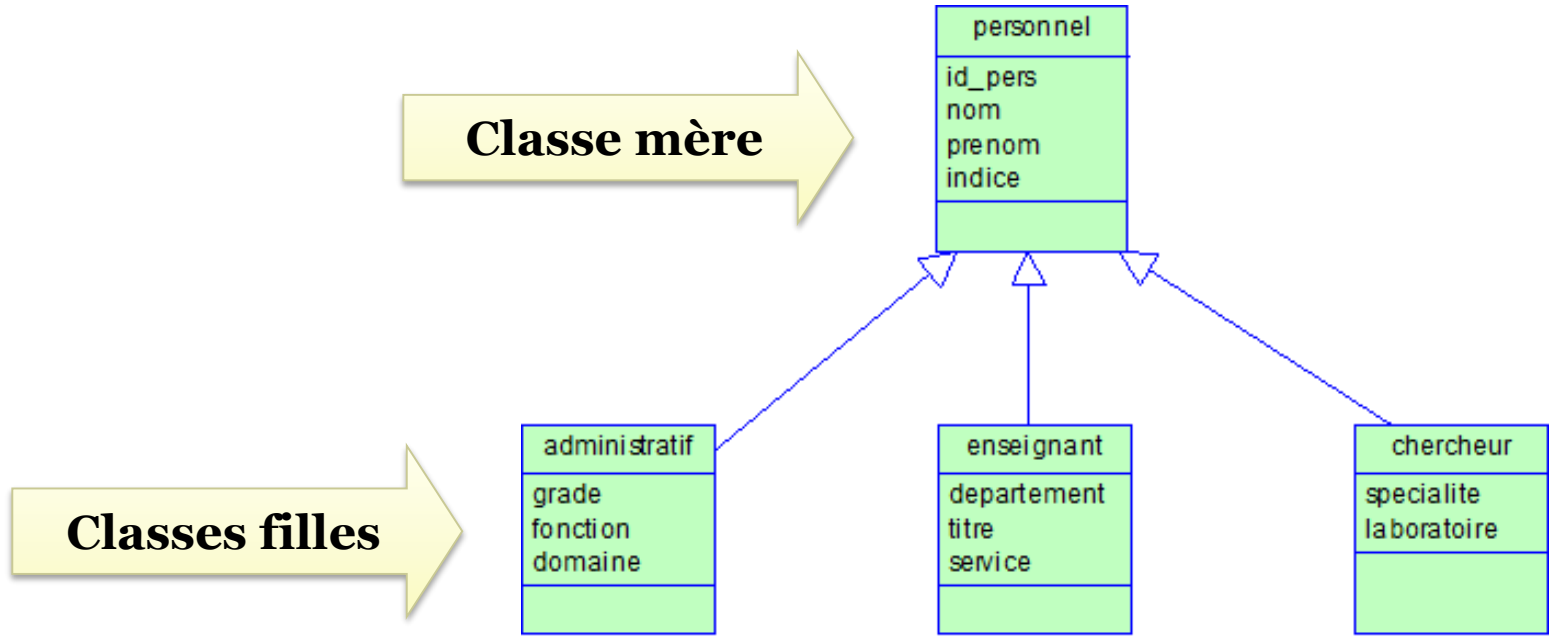
Présentation du formalisme UML

- Les classes d'associations
  - Prenons l'exemple de la quantité d'un produit dans une commande.
    - La solution consiste à déplacer cet attribut dans une classe d'association, de cette manière la quantité commandée dépend de la commande et du produit.



Généralisation et spécialisation

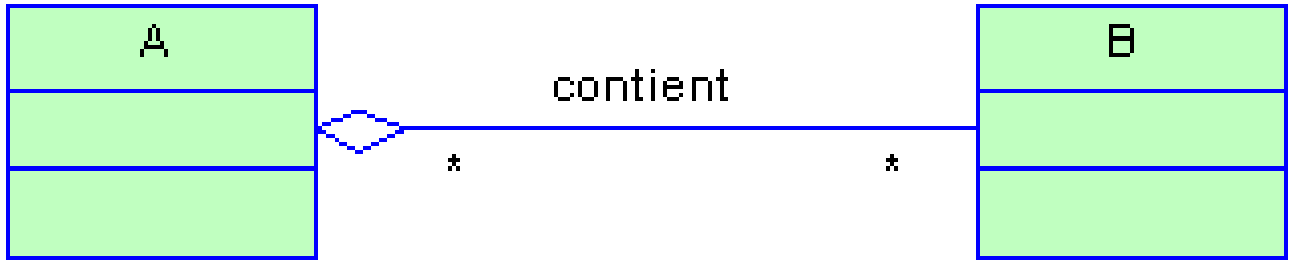
- Notion d'héritage
  - Permet de regrouper dans une classe générique (classe mére) les caractéristiques communes à un ensemble de classes et de mettre les caractéristiques spécifiques dans des classes spécifiques(classes filles)



  - Les classes filles contiennent les caractéristiques de leurs classes mères plus leurs propres caractéristiques

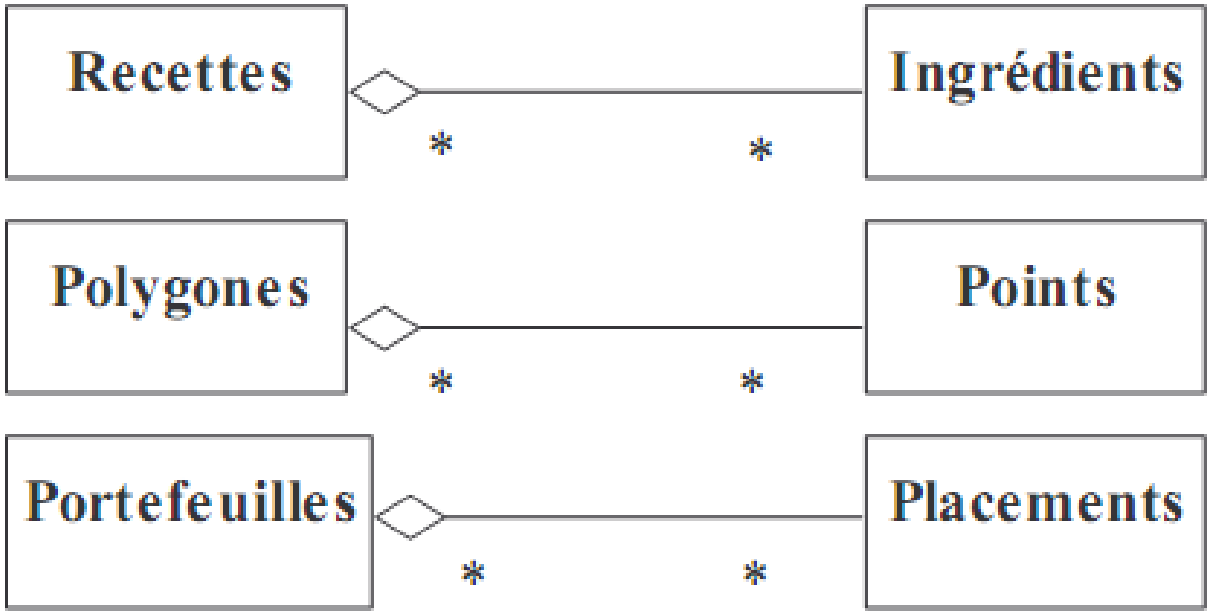
Présentation du formalisme UML

- Agrégation et composition
  - L'agrégation définit une association non symétrique, car les classes associées ne sont plus complètements indépendantes, les phrases suivantes marquent cette dissymétrie :
    - A est formé de B ;
    - A préexiste à B ;
    - A contient B
  - L'agrégation est spécifié par un losange du côté de la classe qui joue le rôle de maitre dans l'association *(A dans les phrases précédentes)*



Présentation du formalisme UML

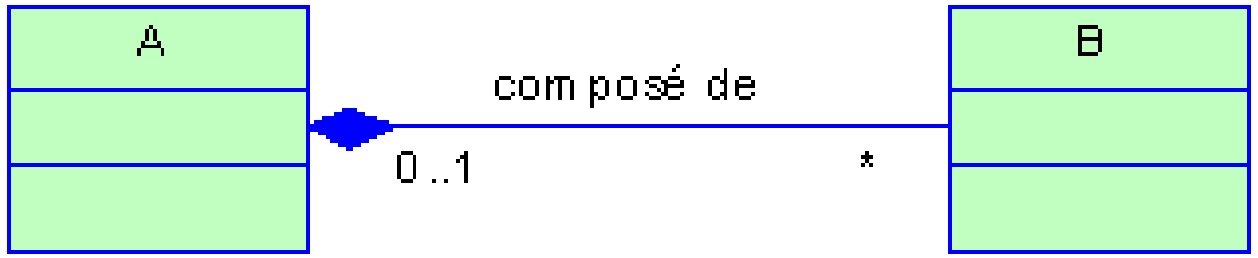
- Agrégation et composition
  - quelques exemples d'agrégations



Présentation du formalisme UML

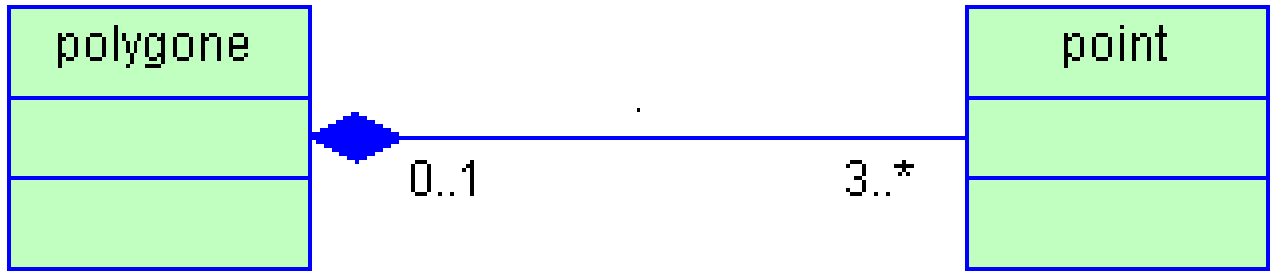
- Agrégation et composition
  - La composition
    - La composition est une agrégation forte,
    - Si B n'est lié qu'à un objet de A alors l'agrégation est une composition

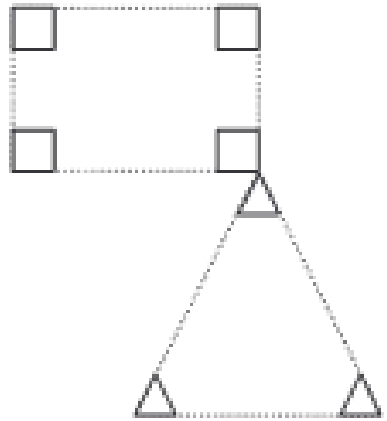
B n'existe pas sans A



Présentation du formalisme UML

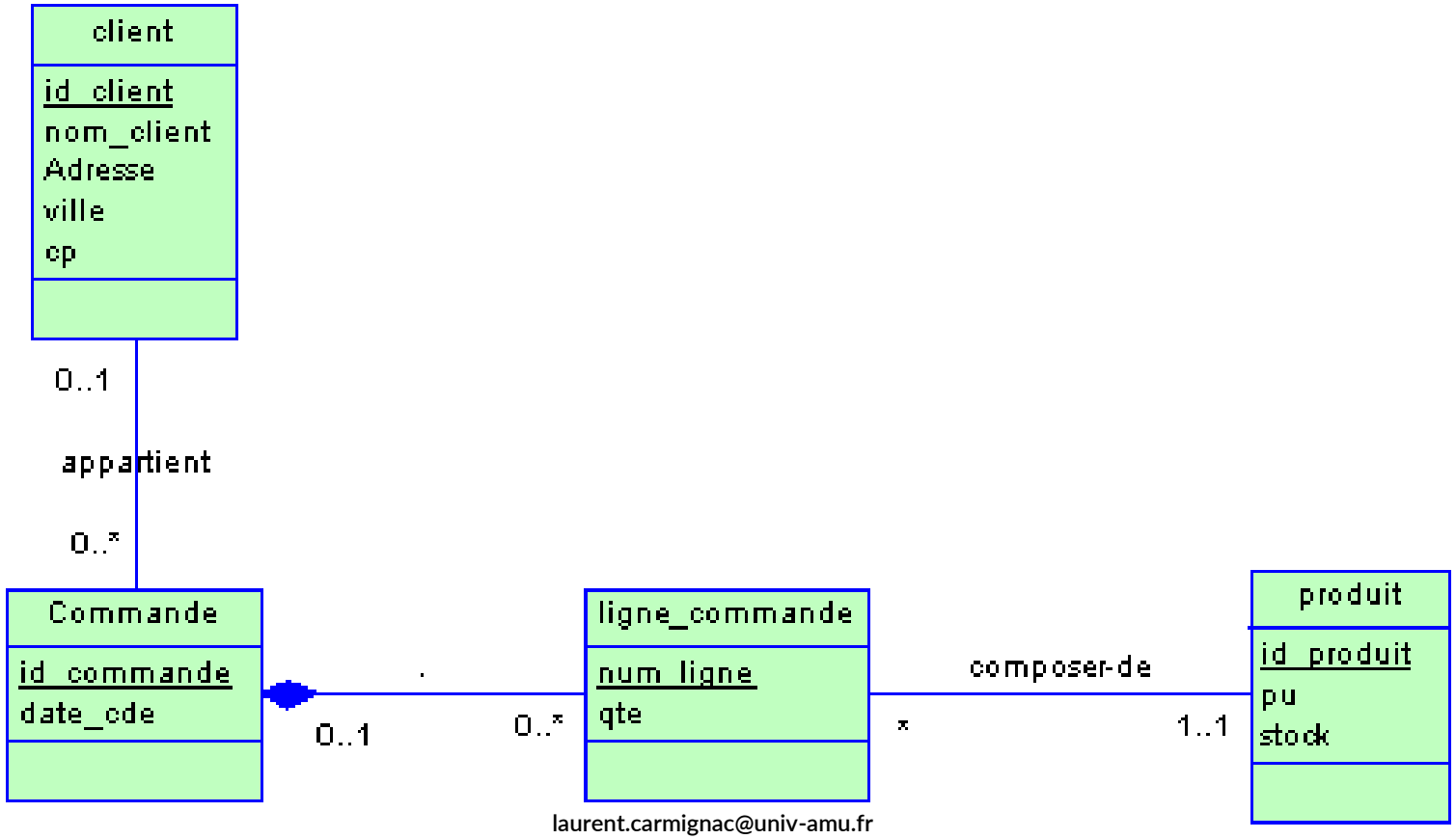
- Agrégation et composition
  - Quelques exemples de composition
    - Un polygone est composé d'au moins 3 points
    - Un point appartient à un seul polygone
      - Si deux polygones partagent un sommet il existe alors deux points différents qui ont les mêmes coordonnées





Présentation du formalisme UML

- Agrégation et composition
  - Quelques exemples de composition



Présentation du formalisme UML

- Une fois ce formalisme assimilé par vos méninges, il va falloir se donner quelques règles de bonne conduite pour bien modéliser

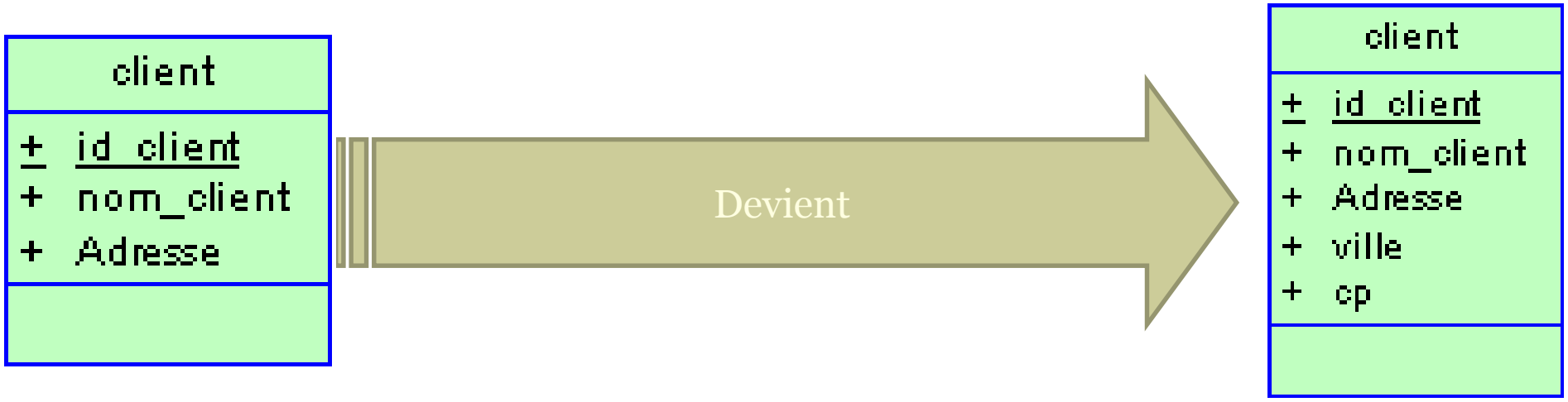
Bien modéliser

- Bien modéliser c'est :
  - éviter une grande partie des sources d'incohérences.
  - rendre possible son implémentation dans un SGBD
  - éviter les redondances.

Un bon modèle doit respecter certaines règles telles que :

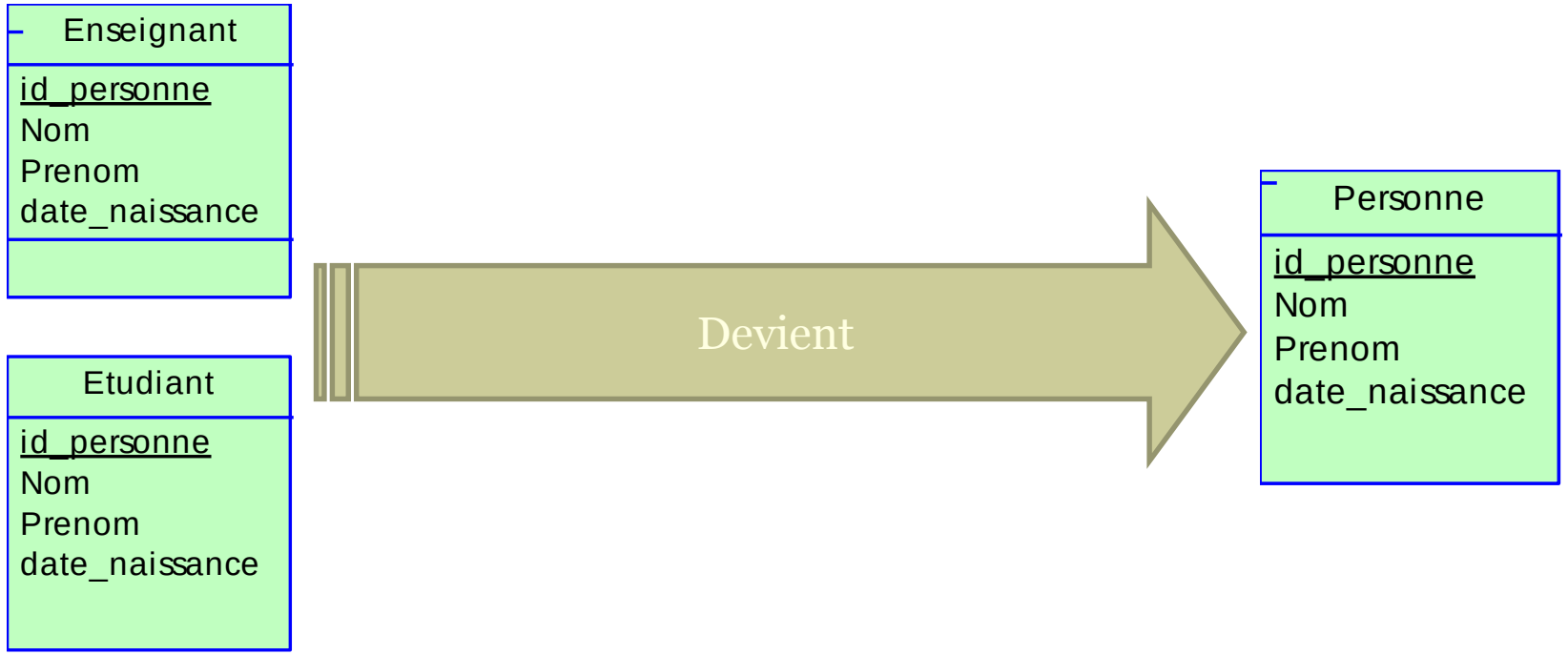
Bien modéliser

- Un attribut ne peut contenir qu'une seule information:
  - *Ici Adresse contient la rue, la ville et le code postale, il faut donc décomposer l'adresse :*



Bien modéliser

- Eviter les classes qui ont les mêmes caractéristiques:
  - *Ici enseignant et étudiant on les mêmes caractéristiques (mêmes attributs), il faut regrouper ces deux classes en une.*



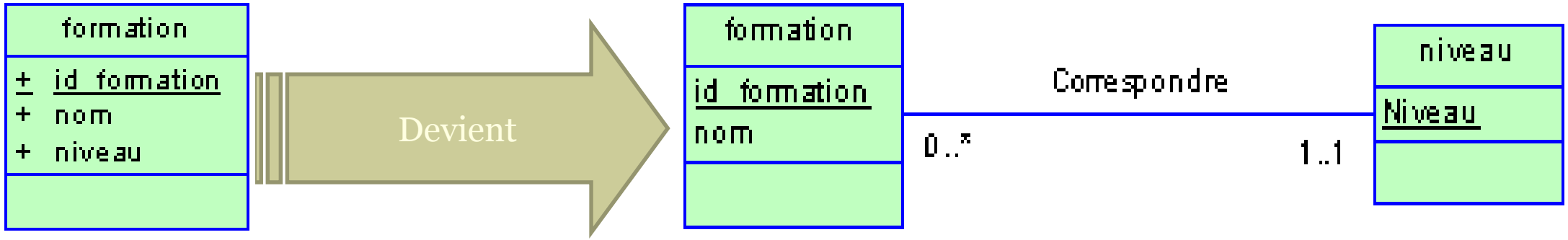
Bien modéliser

- Les attributs qui peuvent être calculés ou déterminés par d'autres attributs ne doivent pas apparaître dans une classe

  - *Ici le montant de la classe commande peut être calculé, il doit disparaitre*

Bien modéliser

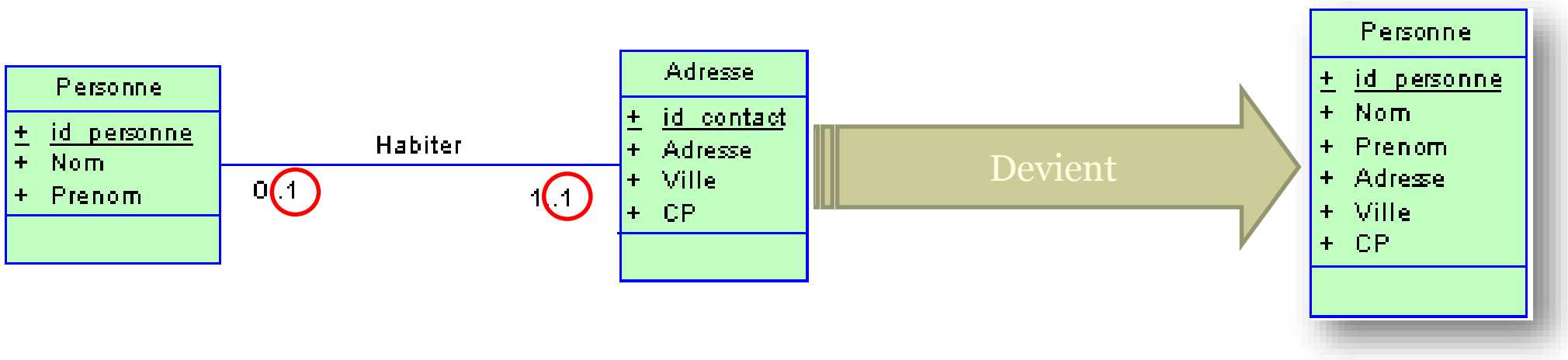
- Un attribut correspondant à un type énuméré est généralement avantageusement remplacé par une classe
  - Par exemple dans la classe formation, le niveau peut prendre les valeurs : *débutant*, *moyen*, *confirmé et expert*



Bien modéliser

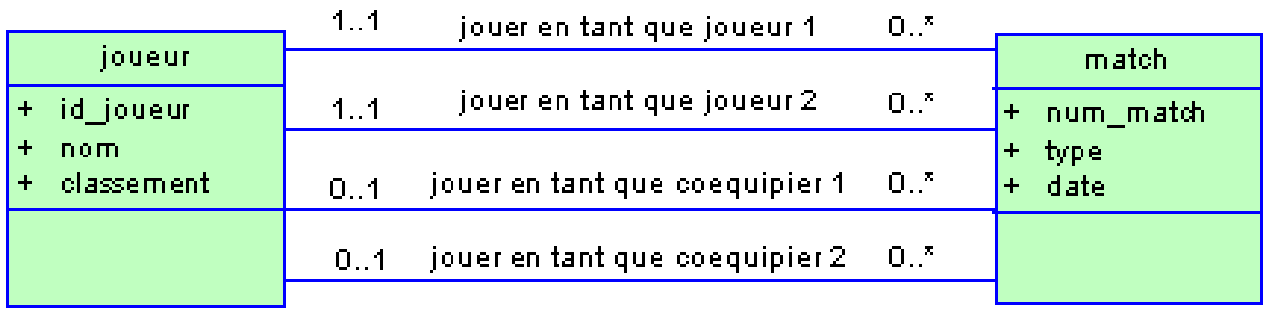
- Si les cardinalités sur une association binaire sont de type X,1 sur chaque extrémité l'association n'a pas lieu d'être

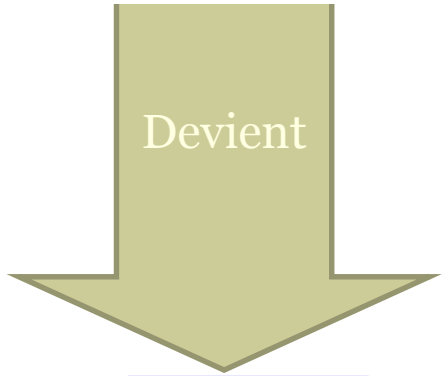
  - *Une personne n'a qu'une seule adresse et l'adresse ne concerne qu'une seule personne, les attributs de la classe adresse sont donc à déplacer dans la classe Personne et la classe adresse disparait*

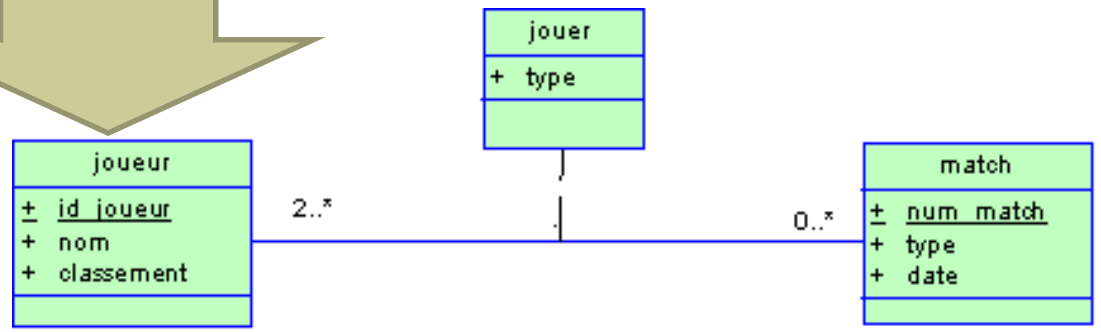


Bien modéliser

- Il faut factoriser les associations quand c'est possible.

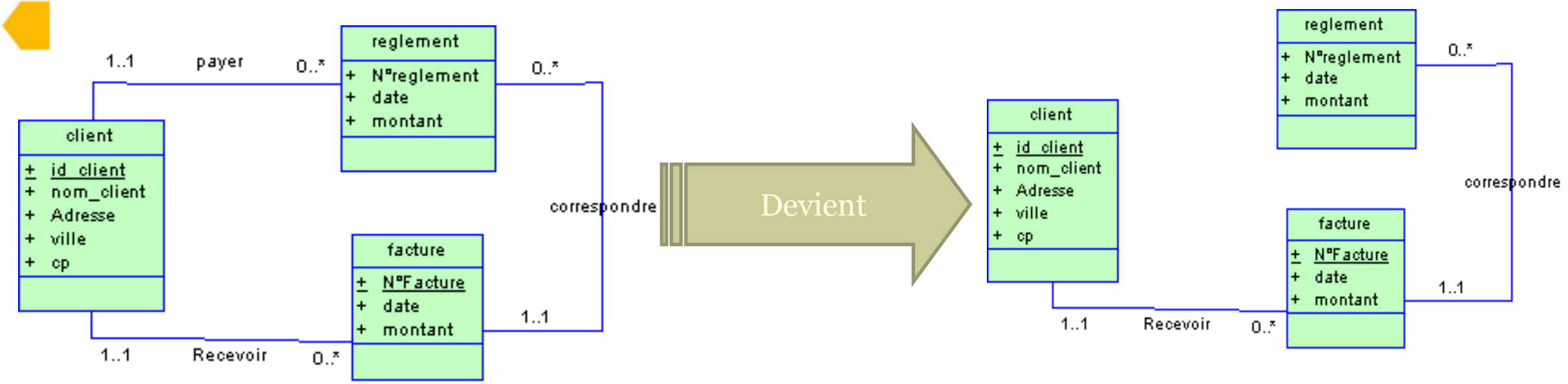






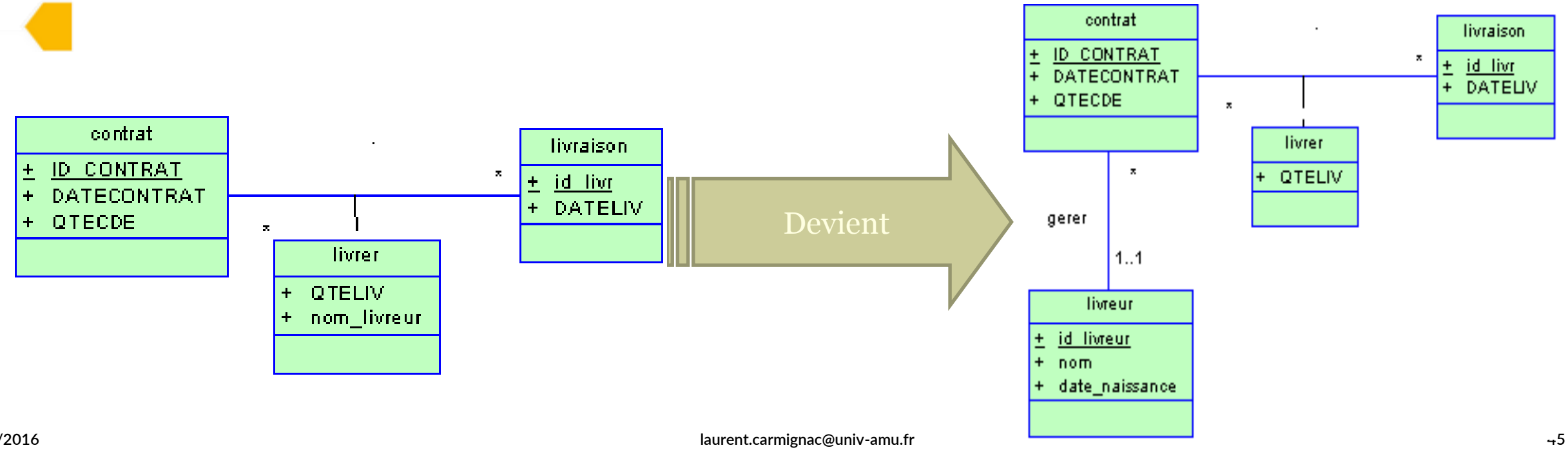
Bien modéliser

- Il faut veiller à éviter les associations redondantes.
  - *En effet, s'il existe deux chemins pour se rendre d'une classe à une autre, alors ces deux chemins doivent avoir deux significations distinctes. Dans le cas contraire, il faut supprimer le chemin le plus court puisqu'il est déductible des autres chemins.*



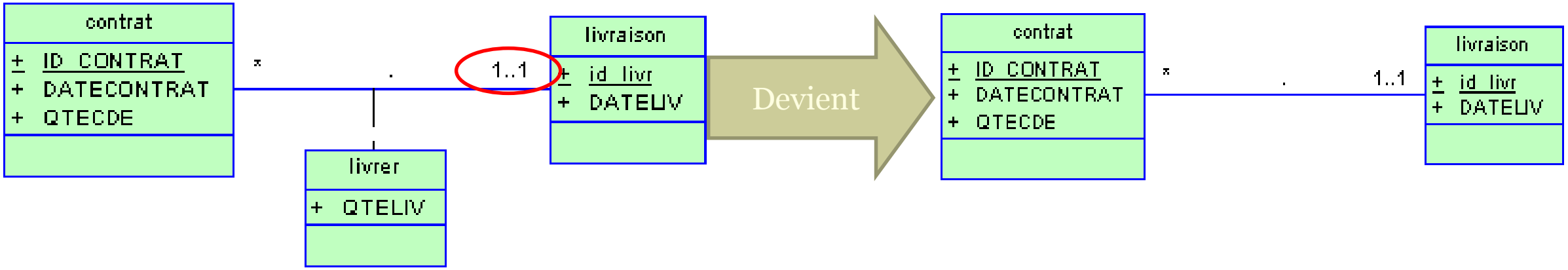
Bien modéliser

- Les attributs d'une classe d'association doivent dépendre directement des deux classes qu'elle lie
  - Dans notre exemple on suppose qu'un contrat est livré par un seul livreur il ne dépend donc que du contrat, l'attribut *nom_livreur* peut passer dans la classe *contrat ou devenir un classe*



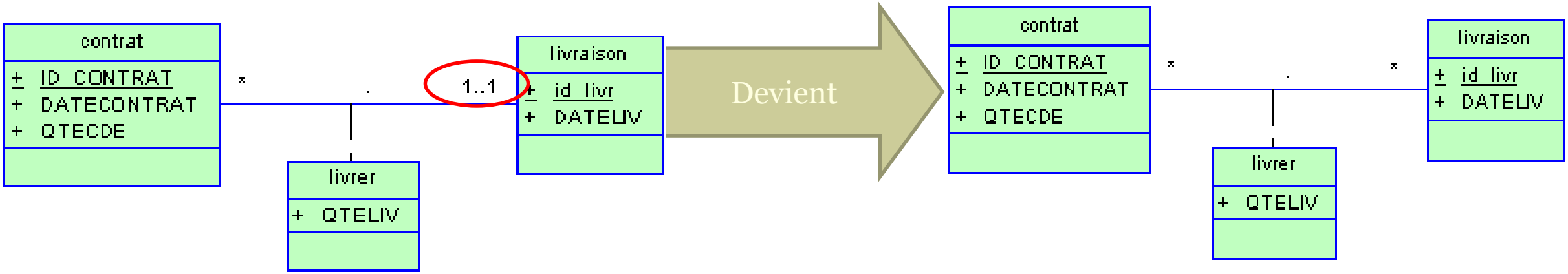
Bien modéliser

- La cardinalités sur une association qui fait intervenir une classe d'association ne peuvent être de type X,1
  - *1er cas : Si un contrat est livré en une seule fois alors la classe d'association livrer est inutile et doit disparaître, la quantité livrée (QTELIV) ne dépendra que de la livraison, en l'occurrence cette attribut n'est plus utile puisque la quantité livrée correspond à la quantité commandée !*



Bien modéliser

- La cardinalités sur une association qui fait intervenir une classe d'association ne peuvent être de type X,1
  - *2ème cas : Si un contrat peut-être livré en plusieurs fois alors il faut changer la cardinalité 1..1 en **



FIN