BUT Informatique
1A - Semestre 1
Introduction aux bases de données
(R1.05)

R. Fleurquin

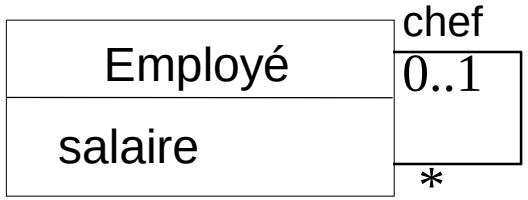
Chapitre 9

Diagramme de classes : le contraintes

Pourquoi des contraintes textuelles annexes?

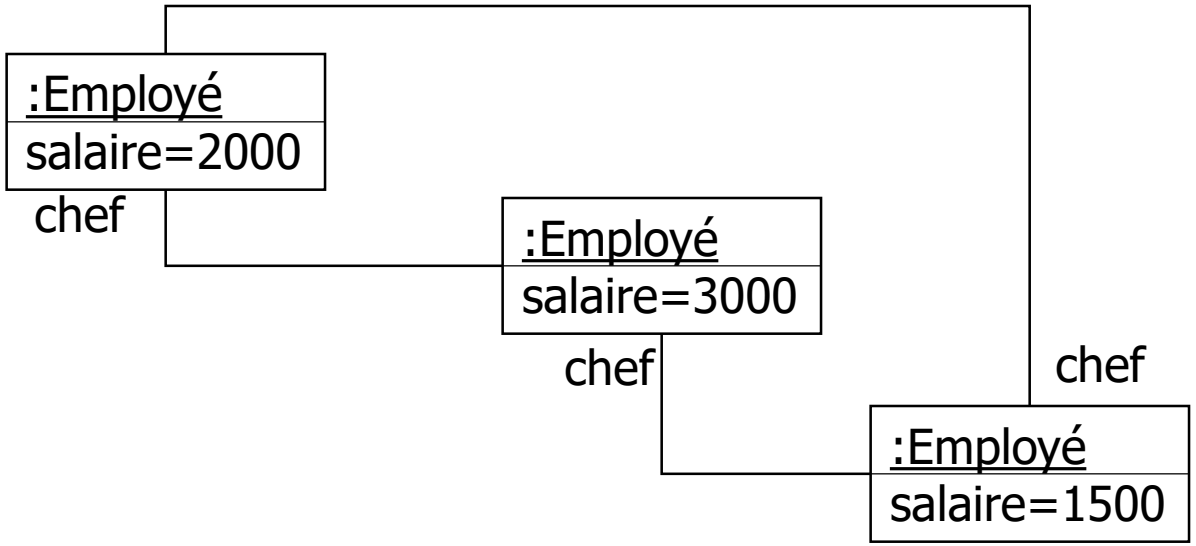
- Il est fréquent de ne pouvoir exprimer toutes les informations d’un domaine à l ’aide des seuls outils langagiers du diagramme de classes UML.
  - Le diagramme de classes peut se révéler trop permissif et permettre la génération de diagrammes d’objets qui n’ont pas de sens dans le domaine modélisé.

Exemple 1

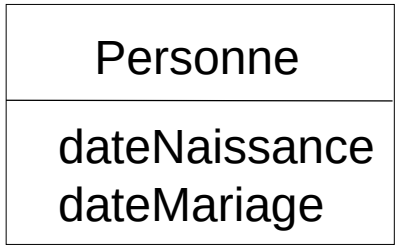


Avec ce diagramme de classes un employé peut avoir comme supérieur l'un de ses subalternes!

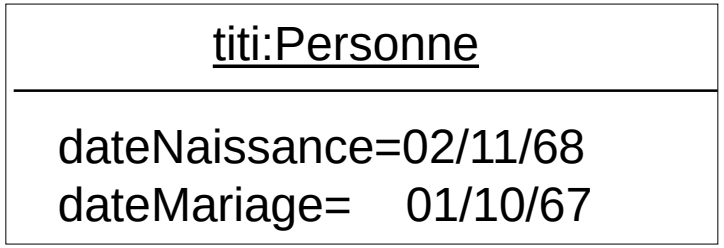
Le salaire d'un subalterne peut être supérieur à celui de son chef!



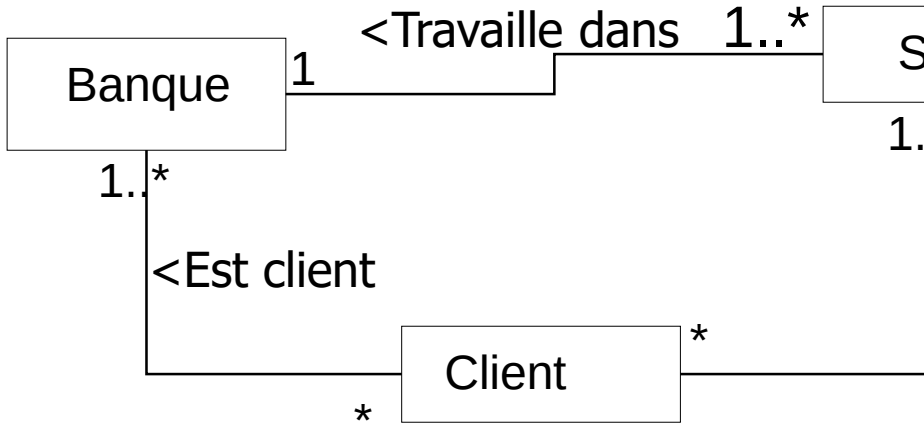
Exemple 2



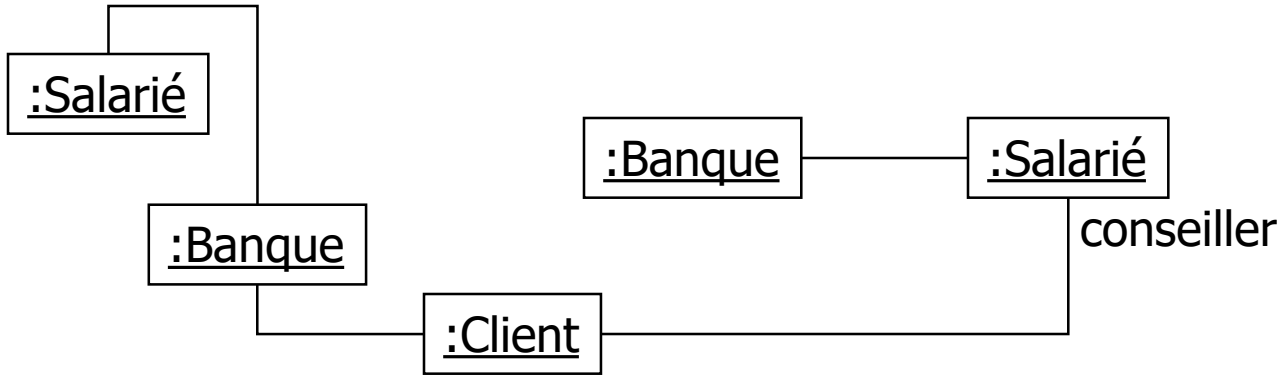
Une personne peut s'être mariée avant d'être née!



Exemple 3 : un « vrai » cycle

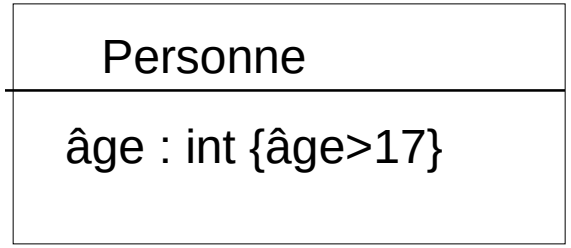


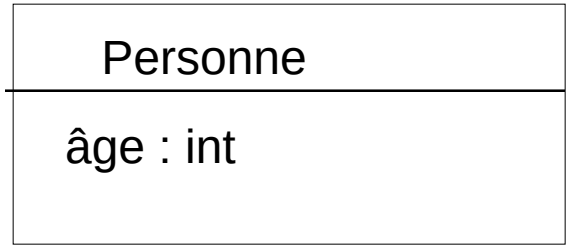
Un client peut être conseillé par un salarié d'une banque dans laquelle il n'est pas client!



- Il est donc souvent nécessaire pour modéliser fidèlement un domaine d'adjoindre aux diagrammes de classes des contraintes qui vont restreindre les diagrammes d'objets instanciables.
- Les contraintes de vos diagrammes comptent autant en partiel que le diagramme de classes UML.

- Les contraintes peuvent se représenter dans un diagramme :
  - près des éléments concernés sur le diagramme entre {…} en usant de contraintes UML prédéfinies par ce standard (XOR, subsets, etc.)
  - en annexe du diagramme en précisant leur contexte (préférable) sous la forme de textes formels ou non.
  - Il existe un langage UML formel pour écrire certaines de ces contraintes annexes : **OCL**
    - Ce langage n'est pas enseigné en DUT
    - Il est peu utilisé en pratique dans l'industrie car compliqué à manipuler et parfois très difficile à lire.
    - La preuve en est que même le standard UML qui utilise OCL pour sa définition rigoureuse et qui a été rédigé par les meilleurs experts du monde comprend des dizaines de fautes (des codes OCL ne compilent pas!!)





Personne
âge>17

A l’IUT de vannes nous ne ferons l’usage que de :

- certaines des contraintes UML prédéfinies directement dans le diagramme de classes

+

- Et très majoritairement de contrainte textuelles écrites en annexe du diagramme en bon français.

- Il y a deux grands types de contraintes :
  - Les contraintes statiques (portent sur la structure des diagrammes d'objets compatibles)
  - Les contraintes dynamiques (portent sur le comportement des objets, **on ne traite pas de cela en R1.05**!)
- Ces contraintes peuvent être :
  - intra-classe (la contrainte ne concerne que les propriétés d'une seule classe)
  - inter-classes (la contrainte concerne plusieurs classes)

- En UML certaines contraintes statiques très fréquentes sont prédéfinies :
  - ReadOnly, XOR, SUBSETS, COMPLETE, DISJOINT, Dérivabilité des attributs et des associations, etc.

A l'IUT de Vannes nous n'userons que d'un sous ensemble de ces contraintes prédéfinies et nous privilégierons les contraintes écrites en français en annexe.

Dans vos diagrammes vous aurez le droit d'user uniquement de :

- Abstract (pour déclarer une classe non instanciable, cf. cours sur la généralisation)
- Subsets (pour indiquer une inclusion entre deux associations)
- Xor (pour indiquer la valuation d'une et d'une seule association parmi 2)
- Attribut et association dérivé (« / »)
- Une notation propriétaire IUT de vannes pour les identifiants (1), (2)…

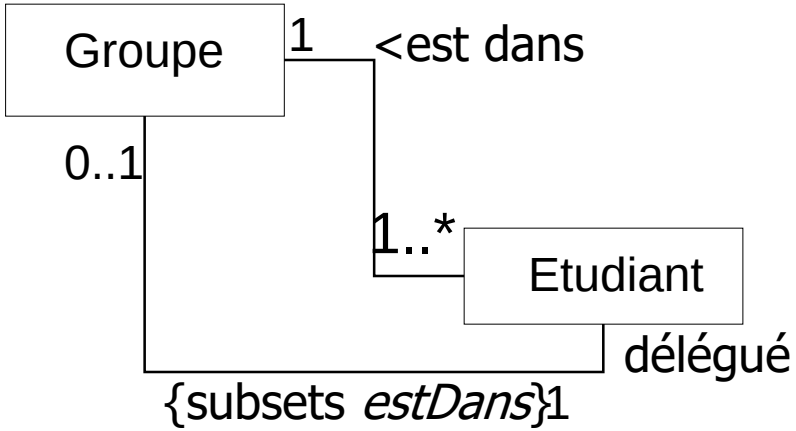
Tout le reste est en contrainte annexe!

- A l ’IUT on usera pour les contraintes en annexe de textes rédigés en (bon) français.
- La syntaxe d’une contrainte sera toujours :

Nom des classes et/ou associations concernées par la contrainte

- Contraintes A
- Contraintes B…

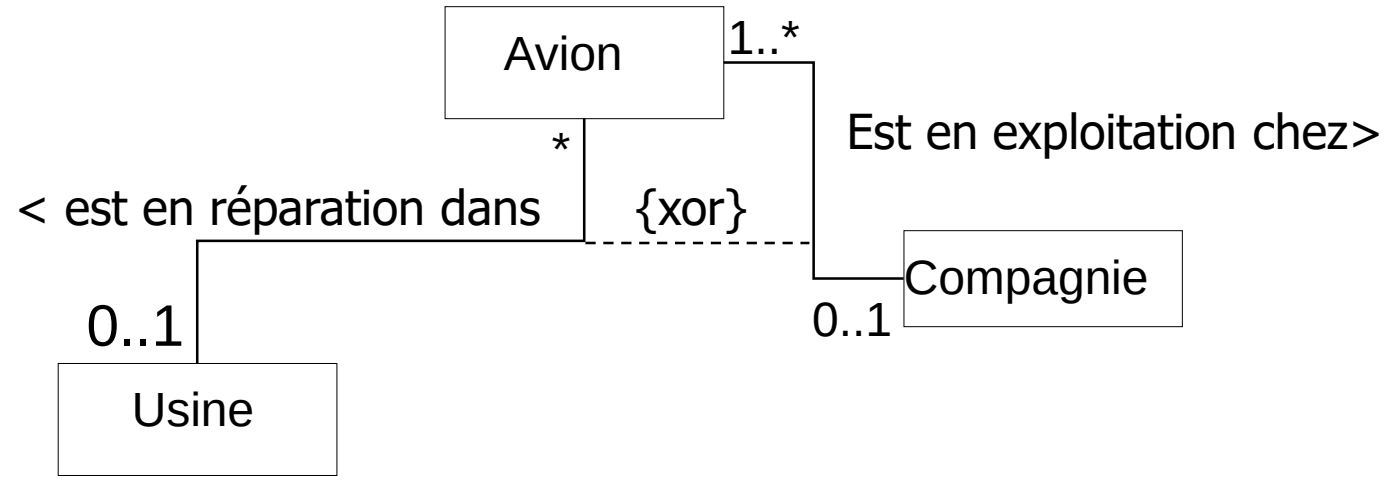
Le cas de subsets



Délégué $\subseteq$ Est_dans

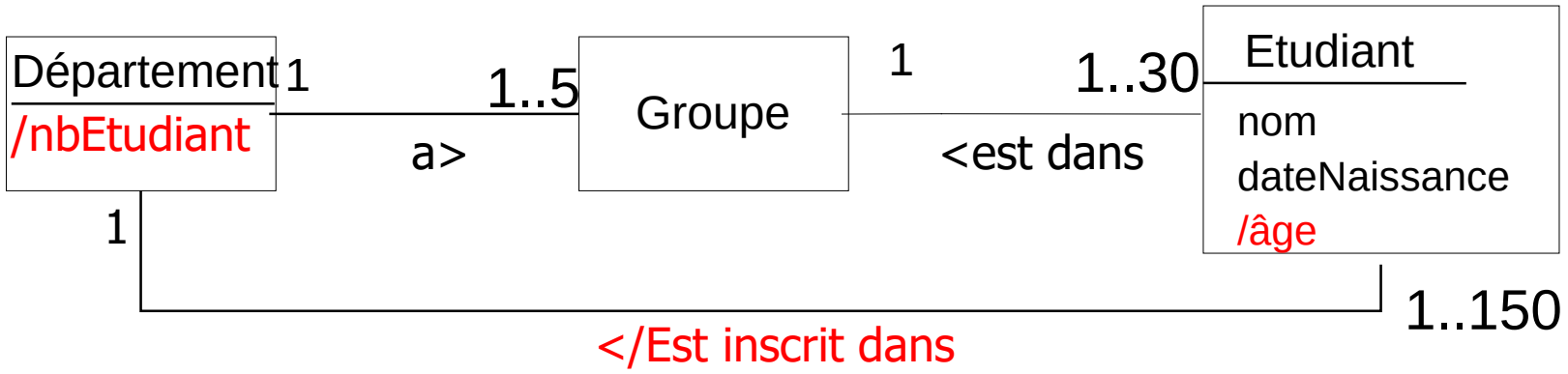
« Un délégué est nécessairement du groupe »

Le cas du XOR

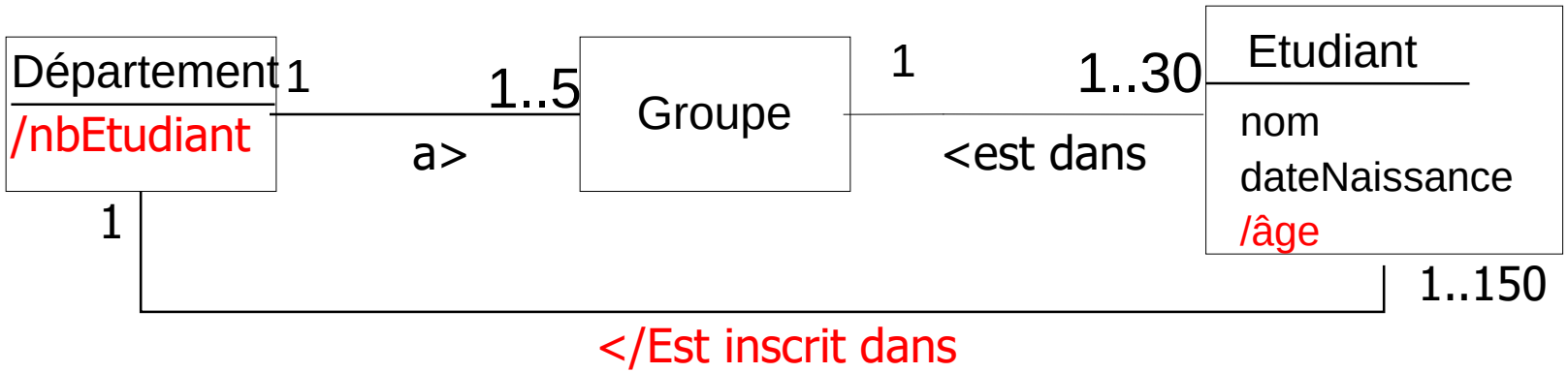


Un avion est soit à l'usine, soit en exploitation dans une compagnie mais jamais dans les deux (c'est le ou exclusif bien connu)

- Un attribut ou une association est dit dérivable s'il peut s'obtenir (être calculé) à l'aide d'autres informations présentes dans le diagramme. Il se note précédé d'un « / »



- Les attributs et associations dérivés doivent être laissés sur le diagramme car ils manifestent une information que l'on souhaite gérer.
- Par contre, il convient :
  - de décorer du « / » les attributs et associations concernés
  - d'indiquer précisément de quelle façon sont calculés ces éléments en contrainte annexe de votre diagramme



Département
- nbEtudiant = somme des étudiants de chacun des groupes liés par « a »

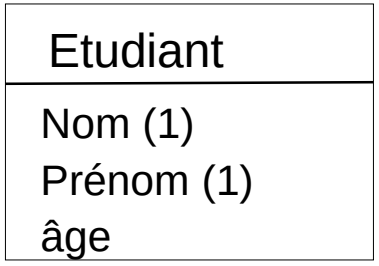
Etudiant
- âge = date courante - dateNaissance

/Est inscrit dans

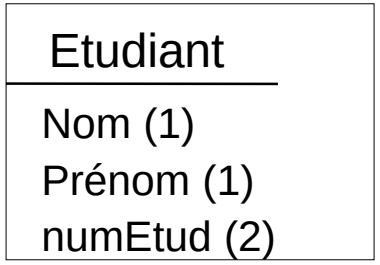
Un étudiant est inscrit dans un département s ’il est dans un groupe appartenant à ce département.

Contraintes de domaines et d'identifiant

- Pour le domaine fini (convention IUT Vannes)
  - `dom(couleur)={R,V,B}`
- Pour le caractère identifiant (convention IUT Vannes) de un ou plusieurs attribut `(1) (identifiant primaire)`, `(2)` (identifiant secondaire) ou `(X)` le `X` pour numéroter les identifiants potentiels multiples



(Nom,prénom) est un identifiant pour les étudiants

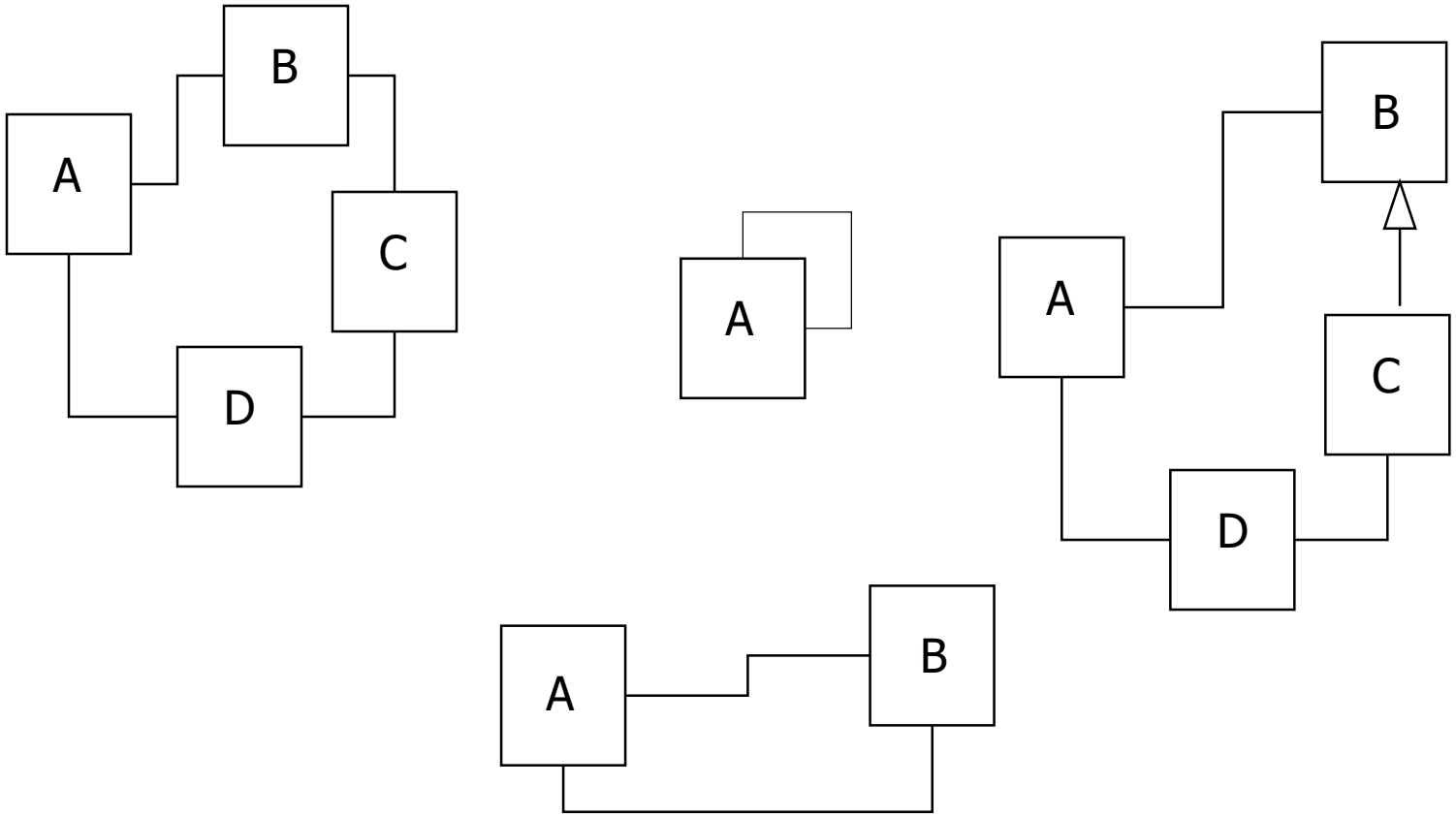


(Nom,prénom) est un identifiant primaire ET (numEtud) seul l'est également (secondaire)

Les contraintes de cycle

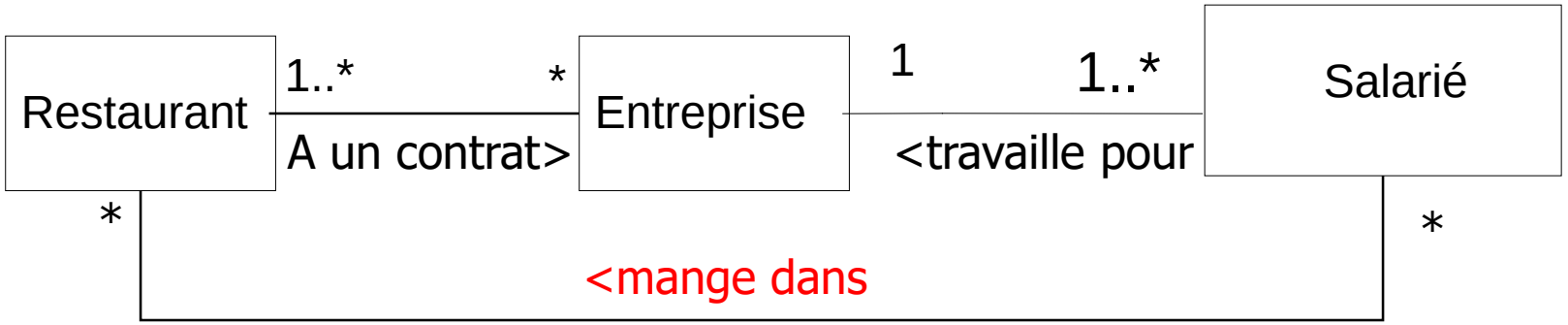
- Des contraintes très fréquentes en modélisation de domaine sont celles à ajouter aux « vrais cycles non simplifiables ».
- Un cycle correspond à la présence, dans le graphe des classes, d'un parcours traversant des associations et des généralisations menant d'une classe vers elle même

Quelques exemples de cycle

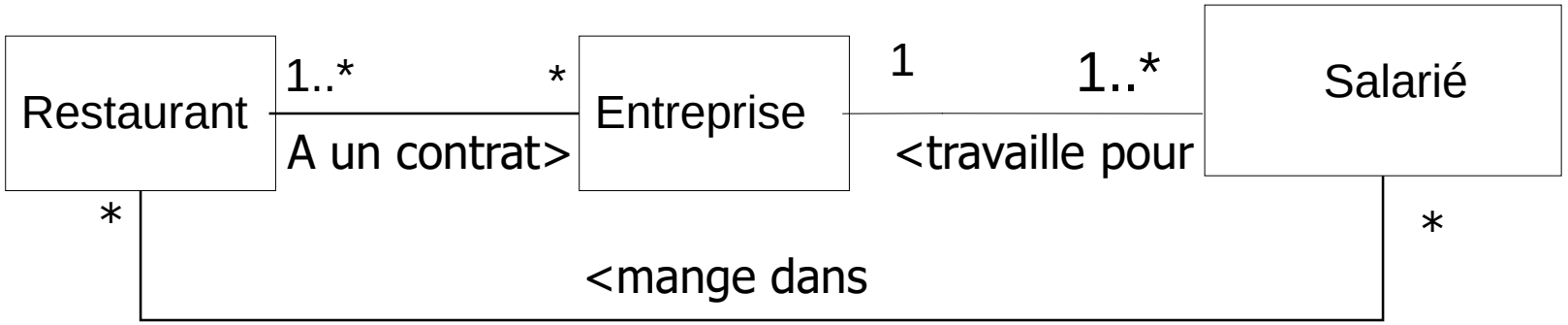


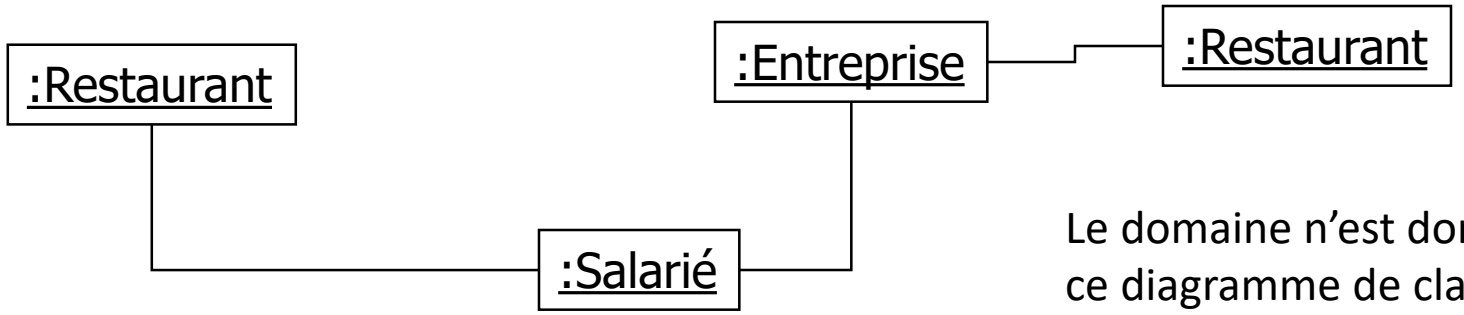
- Il y a trois sortes de cycles :
  - Les VRAIs CYCLES
    - Simplifiable
    - Non Simplifiable
  - Les FAUX CYCLES

- Un vrai cycle est un cycle dans lequel toutes les associations sont sémantiquement liées dans le domaine.



Or, dans ce domaine un salarié ne peut manger que dans un restaurant avec lequel son entreprise a un contrat…
Ici ce diagramme ne permet pas d'assurer cette contrainte

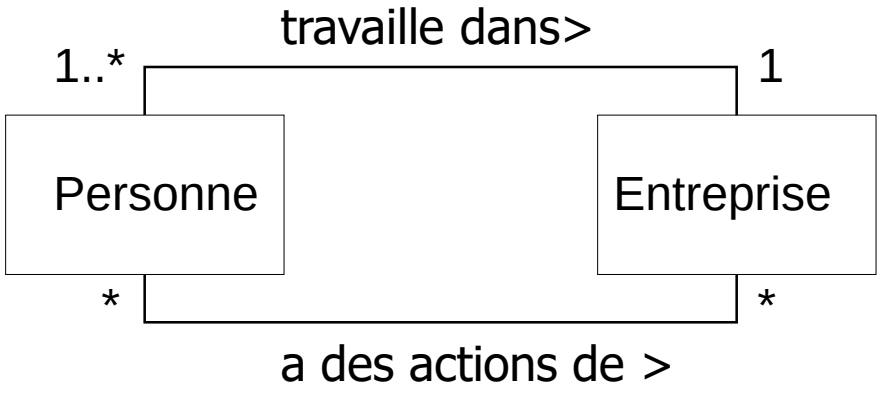




Le domaine n'est donc pas respecté avec ce diagramme de classes!

Les faux cycles

C’est quand certaines de associations du cycle sont sémantiquement indépendantes

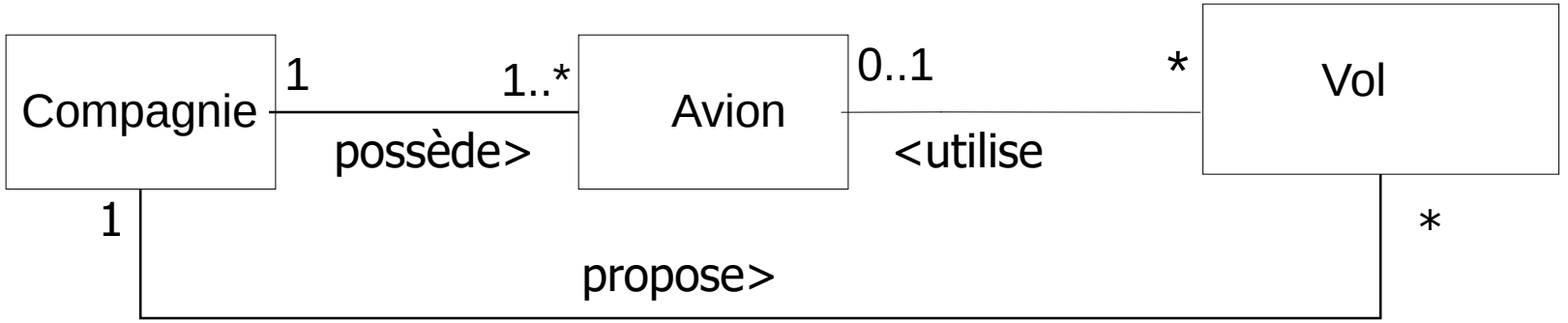


Si dans le domaine modélisé une personne peut avoir des actions de n’importe laquelle des entreprises, les deux associations sont finalement indépendantes. Toutes les situations sont possibles. Le cycle ne pose aucun problème! C’est un faux cycle.

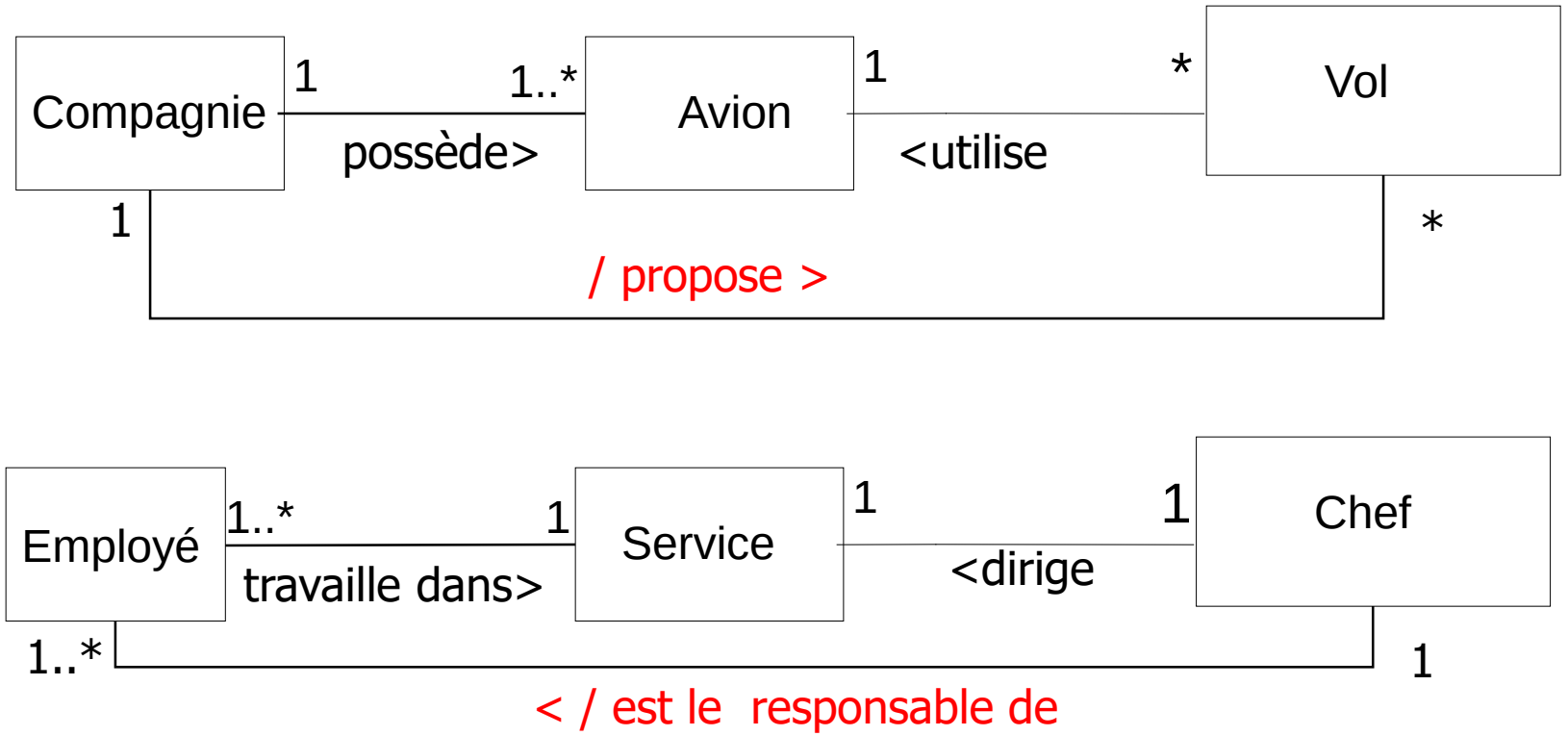
Certains « vrais cycles » sont simplifiables

- Un vrai cycle permet souvent la génération d'instances non conformes au domaine

- Un vrai cycle est simplifiable si l'une au moins des associations participantes est dérivable.
- Note : il se peut même que plusieurs d'entre elles le soient. Il suffit alors d'en choisir une…



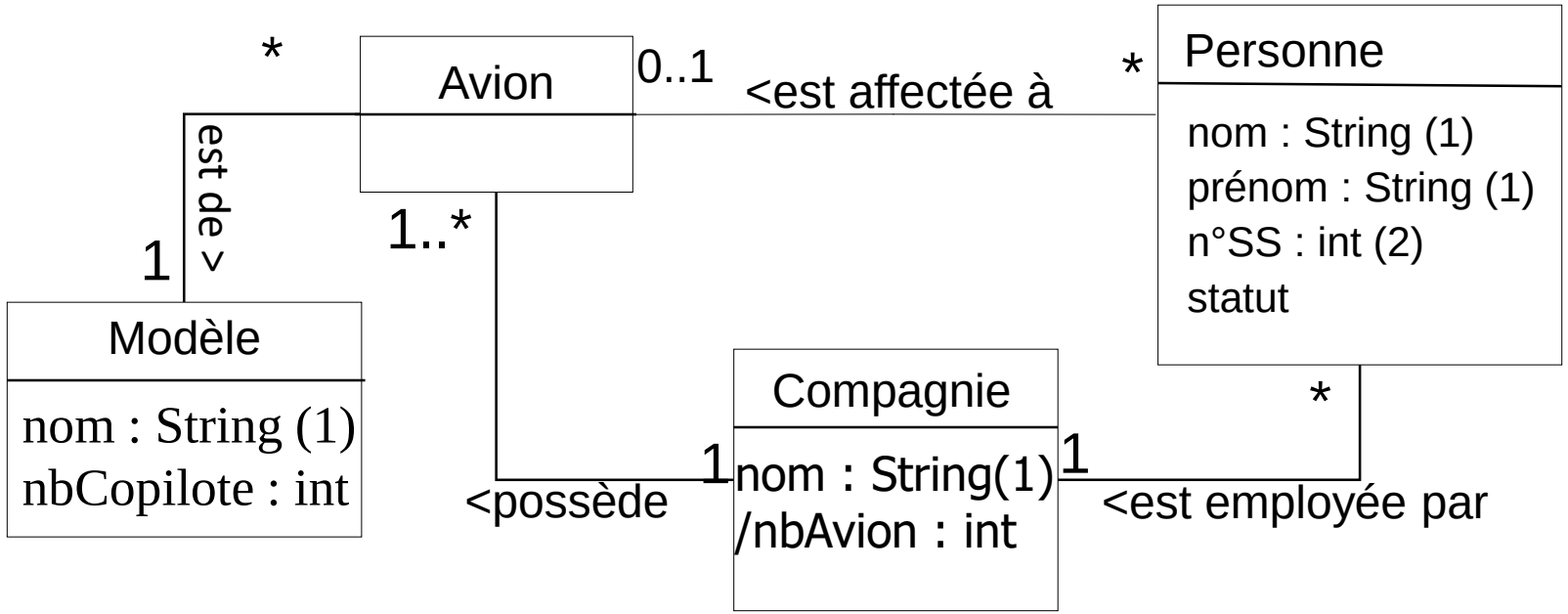
Voici un « vrai cycle » non simplifiable. Aucune des associations de ce cycle ne peut être ôter sans perdre de l'information (essayez de les supprimer une à une pour vous en convaincre)!



2 Vrais cycle simplifiables
(le « / » manifeste la dérivabilité de l'association)

- On résout le problème des vrais cycles en :
  - Soit notant dérivable l'une des associations dérivables d'un cycle simplifiable
  - Soit rajoutant une contrainte textuelle en annexe à un vrai cycle non simplifiable

Exemple



Personne
- dom(statut)={Pilote, Copilote, Stewart, Hôtesse}

Modèle- Avion – Personne
- un avion a 1 personne de statut pilote et autant de personne de statut copilote affectées que le nécessite son modèle

Compagnie
- nbAvion = le nombre des avions liés par l'association « possède »

Avion – Personne - Compagnie

Une personne ne peut être affectée qu'à un avion de sa compagnie.