Base de données: Modélisation orienté objet

Hamida LAGRAA

Diagramme de Classes UML

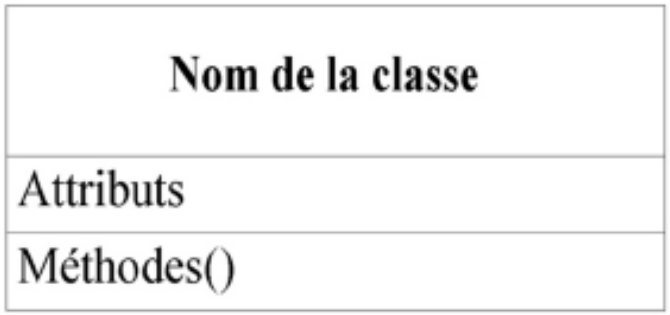
Diagrammes de classes UML

Classe

Définition

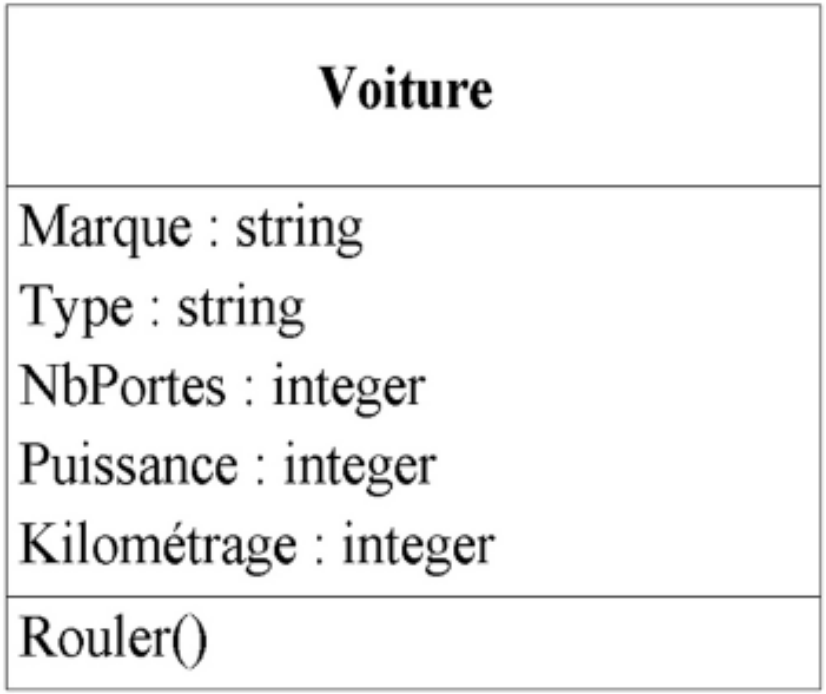
- Une classe est un type abstrait caractérisé par des propriétés (attributs et méthodes) communes à un ensemble d'objets et permettant de créer des instances de ces objets, ayant ces propriétés.

Syntaxe



Diagrammes de classes UML (suite)

- Classe
  - Exemple: La classe voiture



Diagrammes de classes UML (suite)

Classe

Exemple: instance de la classe voiture

- L'objet V1 est une instance de la classe Voiture.
- V1 : Voiture
  - Marque : 'Citroën'
  - Type : 'ZX'
  - Portes : 5
  - Puissance : 6
  - Kilométrage : 300000

Diagrammes de classes UML (suite)

Attribut

Définition

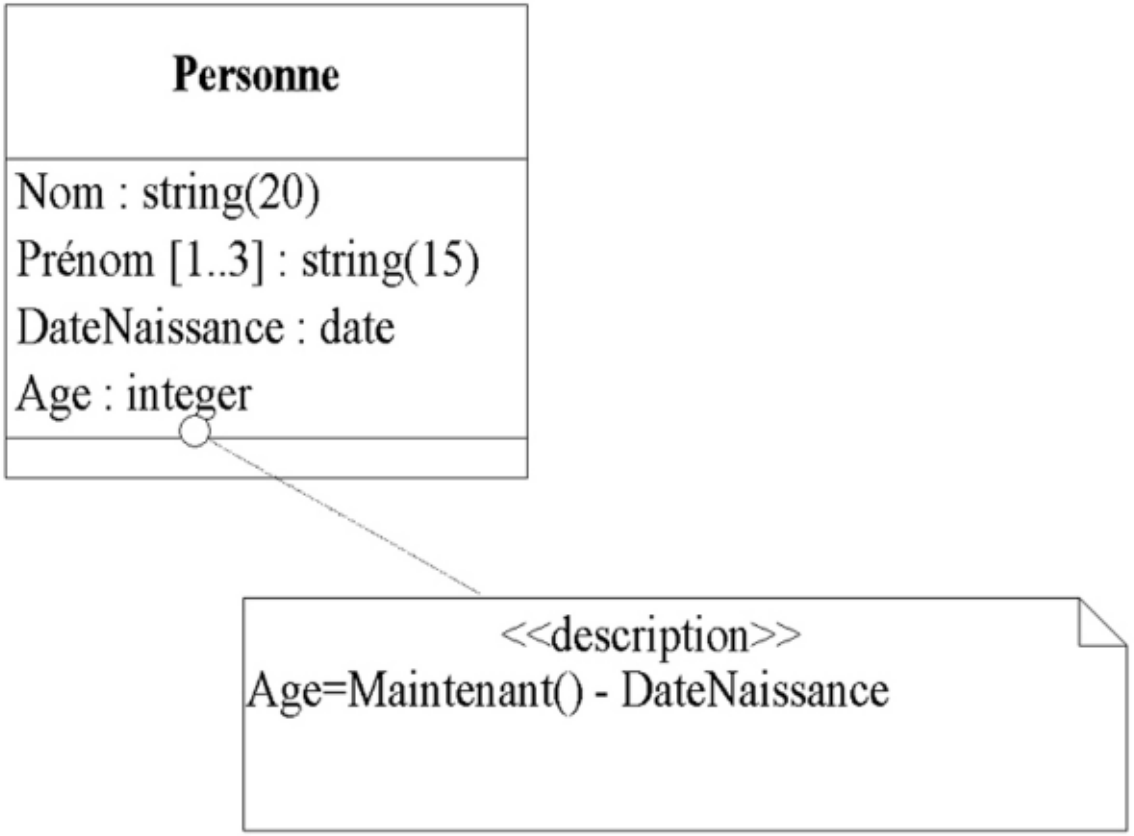
- Un attribut est une information élémentaire qui caractérise une classe et dont la valeur dépend de l'objet instancié.

Remarque

- **Un attribut est typé :**Le domaine des valeurs que peut prendre l'attribut est fixé a priori.
- **Un attribut peut être multivalué:** Il peut prendre plusieurs valeurs distinctes dans son domaine.
- **Un attribut peut être dérivé:** Sa valeur alors est une fonction sur d'autres attributs de la classe (il peut donc aussi être représenté comme une méthode, et c'est en général préférable)

Diagrammes de classes UML (suite)

- Attribut
  - Exemple : la classe Personne



Diagrammes de classes UML (suite)

Héritage

Définition

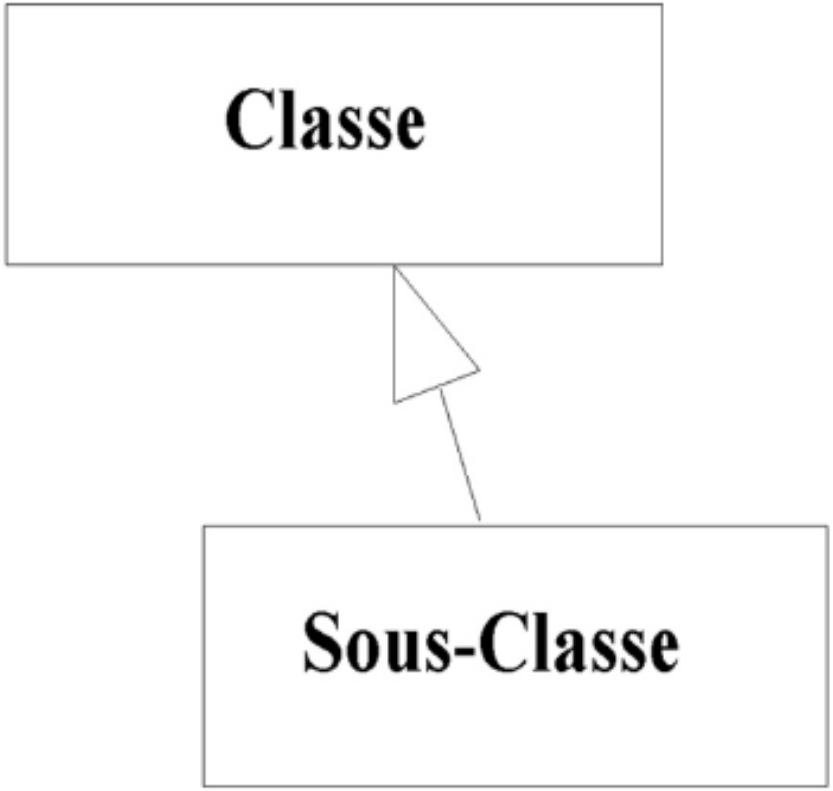
- L'héritage est une relation entre deux classes permettant d'exprimer que l'une est plus générale que l'autre.
- L'héritage implique une transmission automatique des propriétés (attributs et méthodes) d'une classe A à une classe A'.
- Dire que A' hérite de A équivaut à dire que A' est une sous-classe de A. On peut également dire que A est une généralisation de A' et que A' est une spécialisation de A.

Remarque

- L'héritage permet de représenter la relation « est-un » entre deux objets.
- l'héritage a un avantage pratique, celui de factoriser la définition de propriétés identiques pour des classes proches.

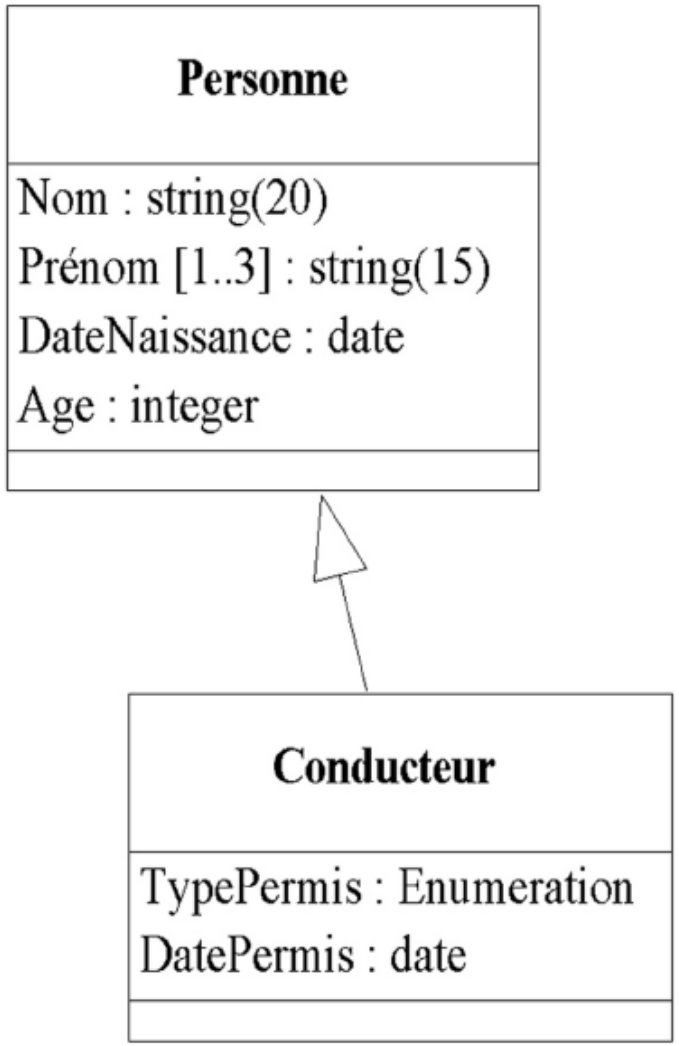
Diagrammes de classes UML (suite)

- Héritage
  - Syntaxe



Diagrammes de classes UML (suite)

- Héritage
  - Exemple:
    - ✓ la classe Conducteur



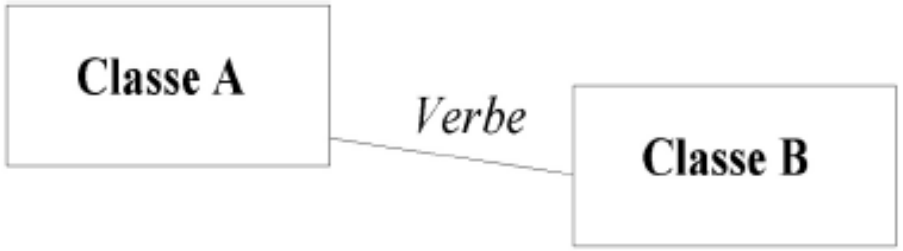
Diagrammes de classes UML (suite)

Association

Définition

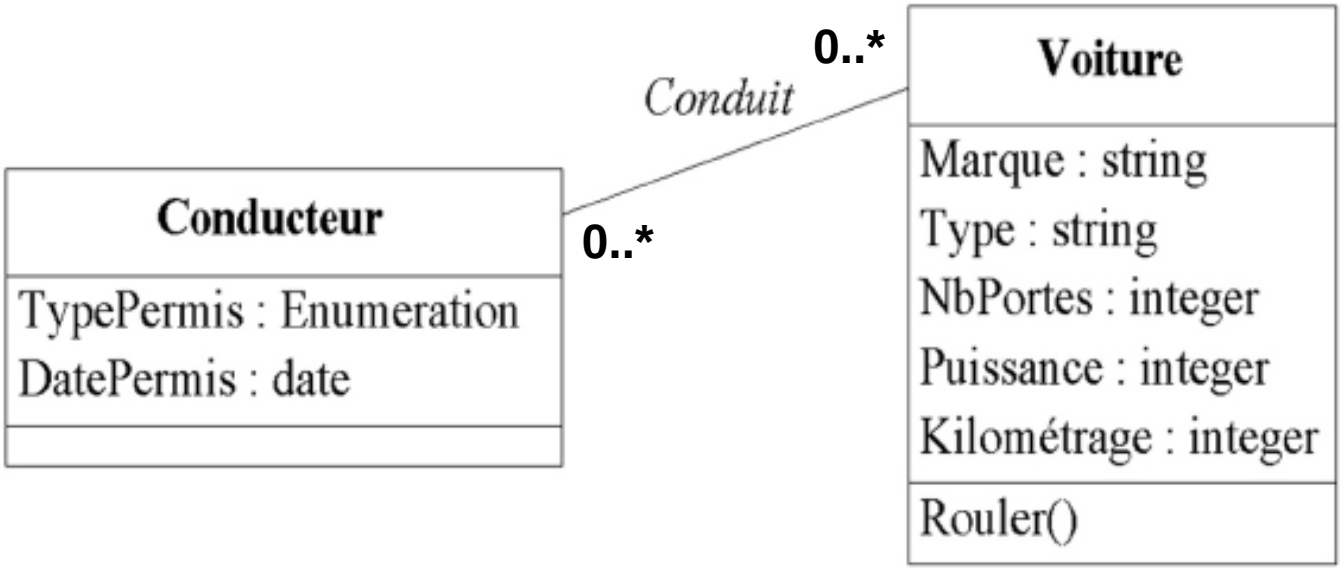
- Une association est une relation logique entre deux classes (association binaire) ou plus (association n-aire) qui définit un ensemble de liens entre les objets de ces classes.
- Une association est nommée, généralement par un verbe.
- Une association peut avoir des propriétés (à l'instar d'une classe).
- Une association définit le nombre minimum et maximum d'instances autorisée dans la relation (on parle de cardinalité).

Notation



Diagrammes de classes UML (suite)

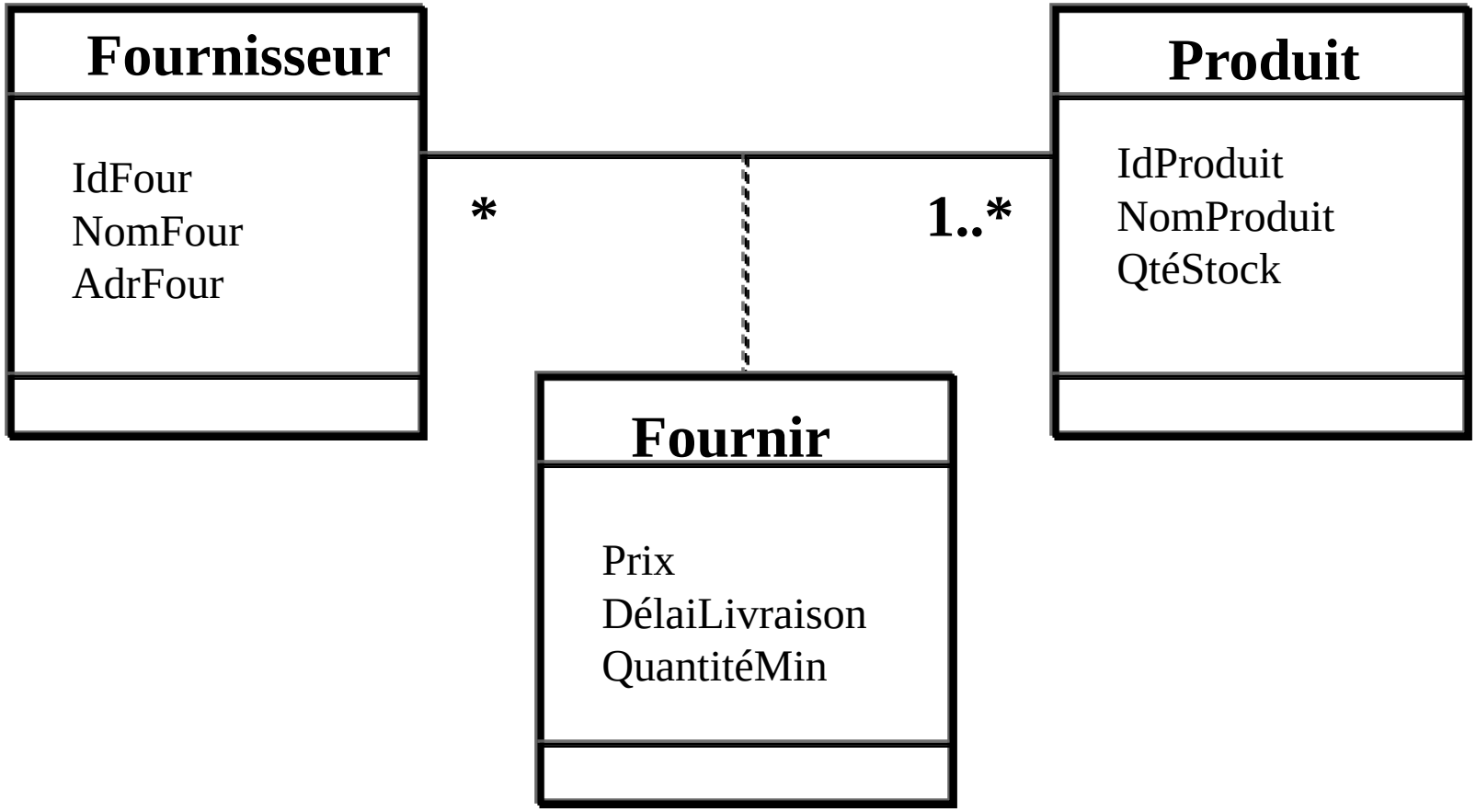
- Association
  - Exemple



Diagrammes de classes UML (suite)

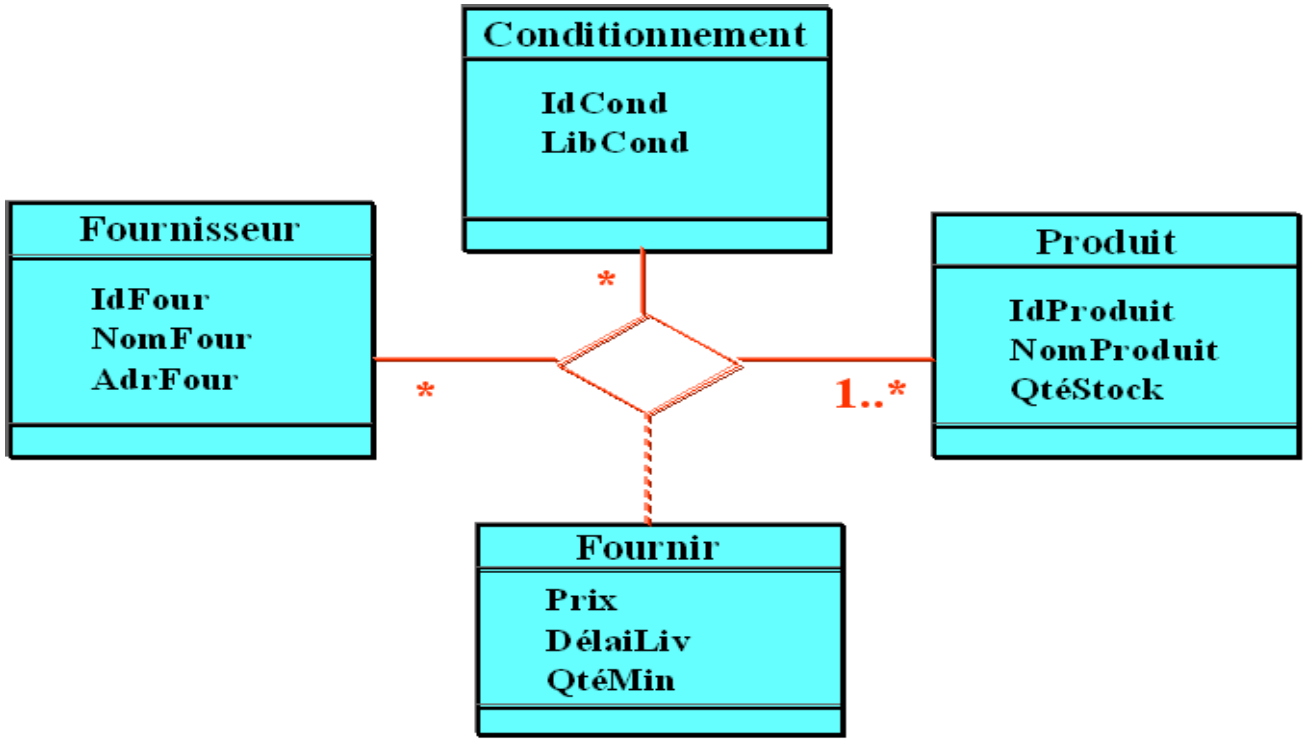
Association

Exemple : association avec attributs



Diagrammes de classes UML (suite)

- Association n-aire
  - Exemple :



Diagrammes de classes UML (suite)

Cardinalité d'une association

Définition

- La cardinalité d'une association permet de représenter le nombre minimum et maximum d'instances qui sont autorisées à participer à la relation.
- La cardinalité est définie pour les deux sens de la relation.

Syntaxe

- Si min_a (resp. max_a) est le nombre minimum (resp. maximum) d'instances de la classe A autorisées à participer à l'association, on note sur la relation, à côté de la classe A : $min_a..max_a$.
- Si le nombre maximum est indéterminé, on note n ou *.

Diagrammes de classes UML (suite)

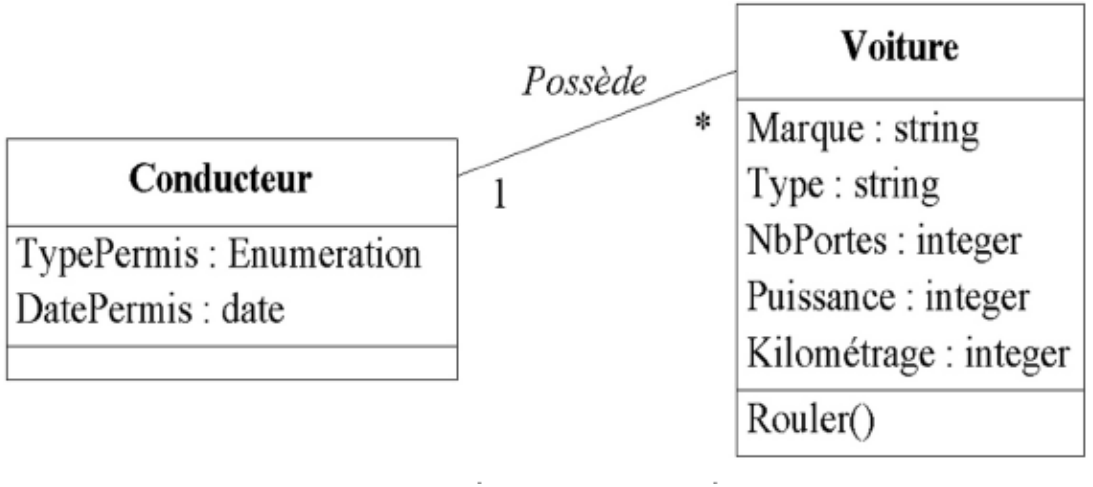
Cardinalité d'une association

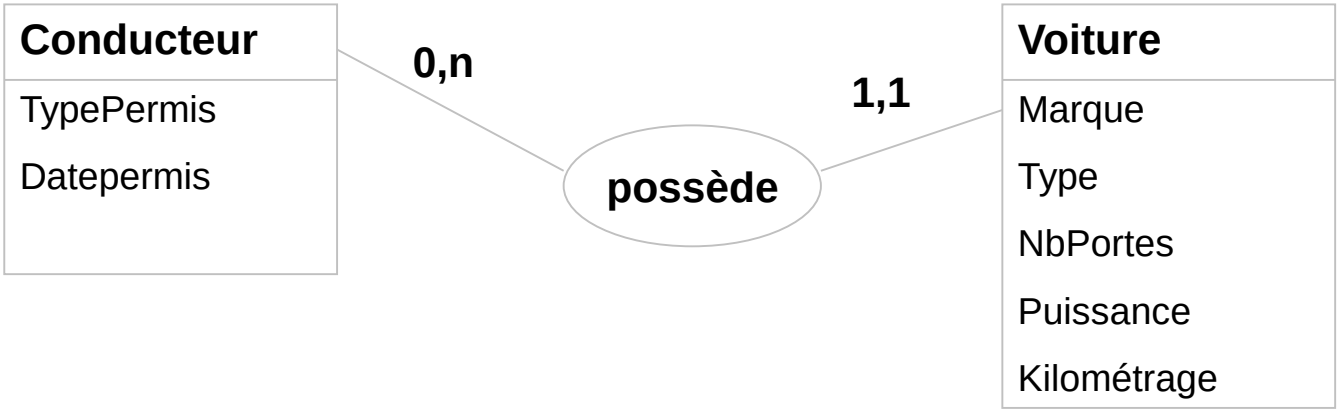
- Remarque : Les cardinalité les plus courantes sont :
  - ✓ 0..1 (optionnel)
  - ✓ 1..1 ou 1 (un)
  - ✓ 0..n ou 0..* ou * (plusieurs)
  - ✓ 1..n ou 1..* (obligatoire)
- Attention
  - ✓ La notation de la cardinalité en UML est opposée à celle adopté en E-A.
    - En E-A, on note à gauche(resp. à droite) de l'association le nombre d'instances de la classe de gauche (resp. de droite) autorisées dans l'association.
    - En UML, on note à gauche (resp. à droite) le nombre d'instances de la classe de droite (resp. de gauche) autorisées dans l'association.

Diagrammes de classes UML (suite)

Cardinalité d'une association

Exemple

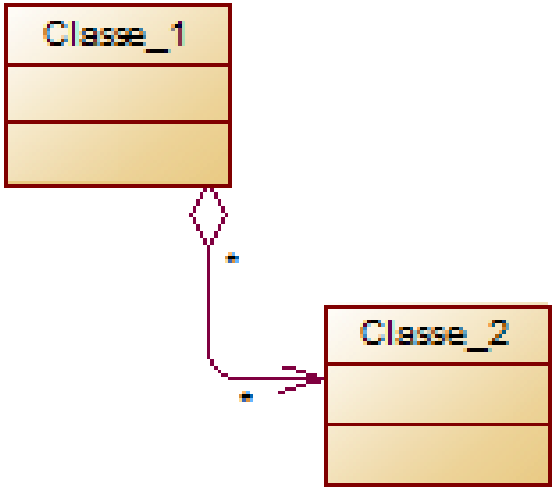




Diagrammes de classes UML (compléments)

Agrégation

- Association particulière utilisée pour préciser une relation tout/partie (ou ensemble/element)
- Exemple

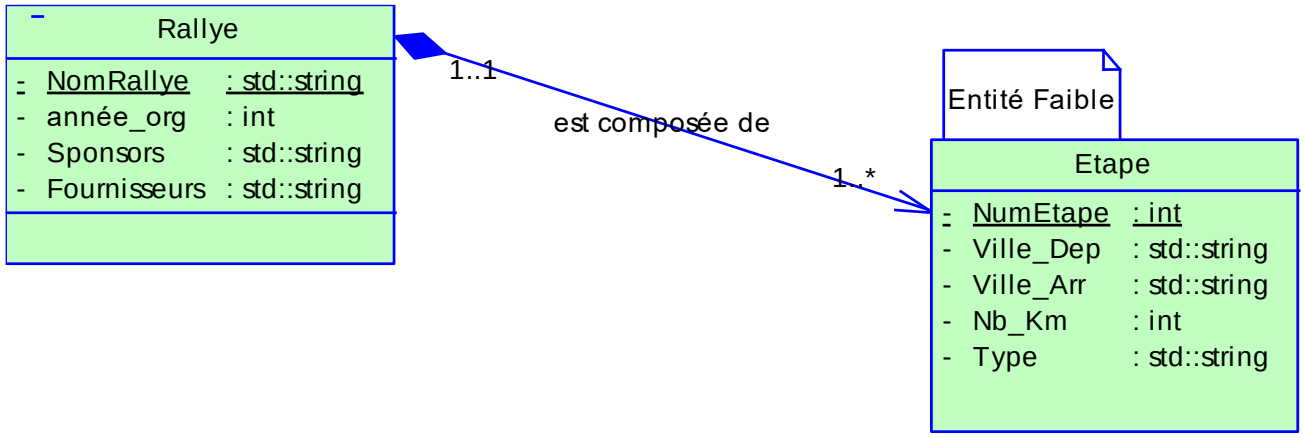


- **Attention**
  - ✓ **Il n'y a pas la relation de dépendance forte (exprimée par la composition)**.

Diagrammes de classes UML (suite)

Entités faibles (COMPOSITION)

Exemple



- L'entité Etape est complètement dépendante de l'entité Rallye et sa clé locale (NumEtape) n'est pas suffisante à l'identifier de façon absolue.

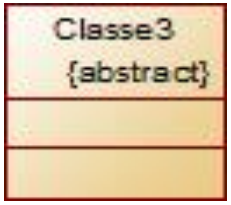
Attention

- Le repérage des entités de type faible est très important dans le processus de modélisation, il permet de réfléchir à la meilleure façon d'identifier de façon unique les entités et donc de trouver les meilleures clés.

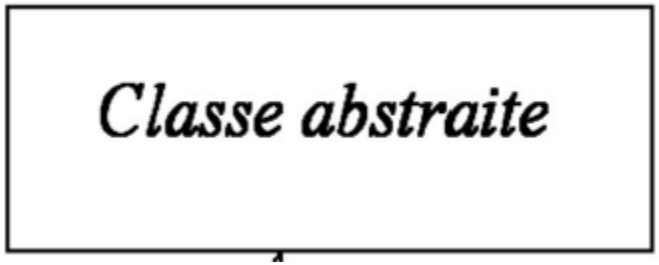
Diagrammes de classes UML (compléments)

Classe Abstraite

- C'est une classe non instanciable
- Elle est toujours héritée
- Exemple



- Ou



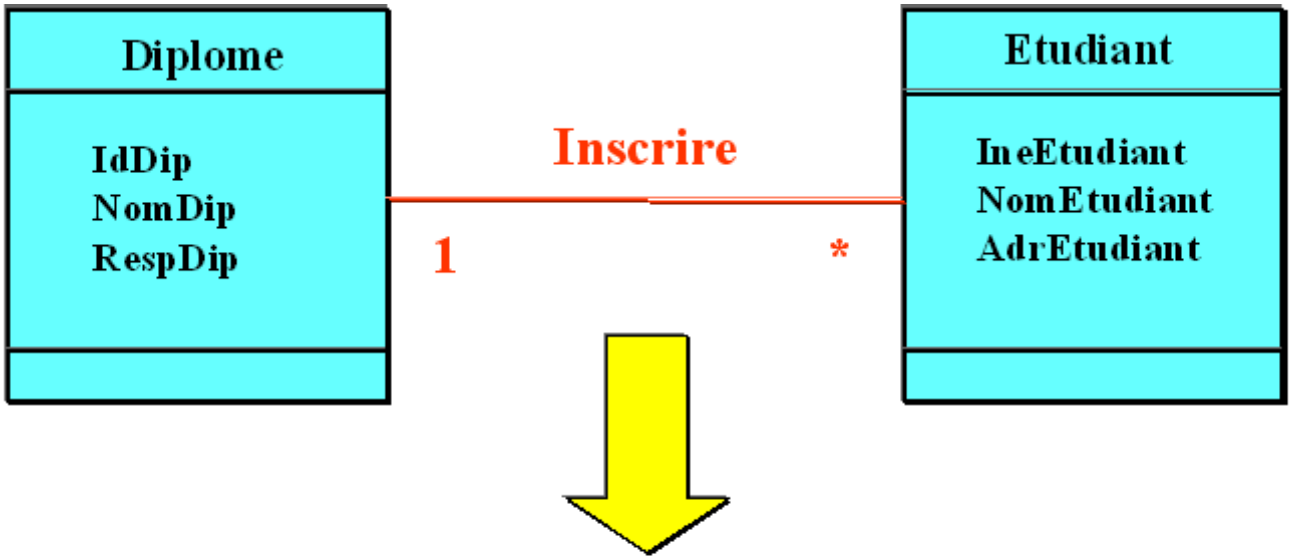
Passage DC vers relationnel

Le Passage DC vers Relationnel

- **Règle n°1 : Classes normales**
  - Chaque classe devient une relation
  - L'identifiant de la classe devient la clé primaire de la relation
- **Règle n°2 : Classes d'Associations 1-N (Mère-Fille)**
  - Cette classe disparaît
  - La clé de la relation mère glisse dans la relation fille comme Clé Étrangère
-

Le Passage DC vers Relationnel (suite)

- Exemple 1

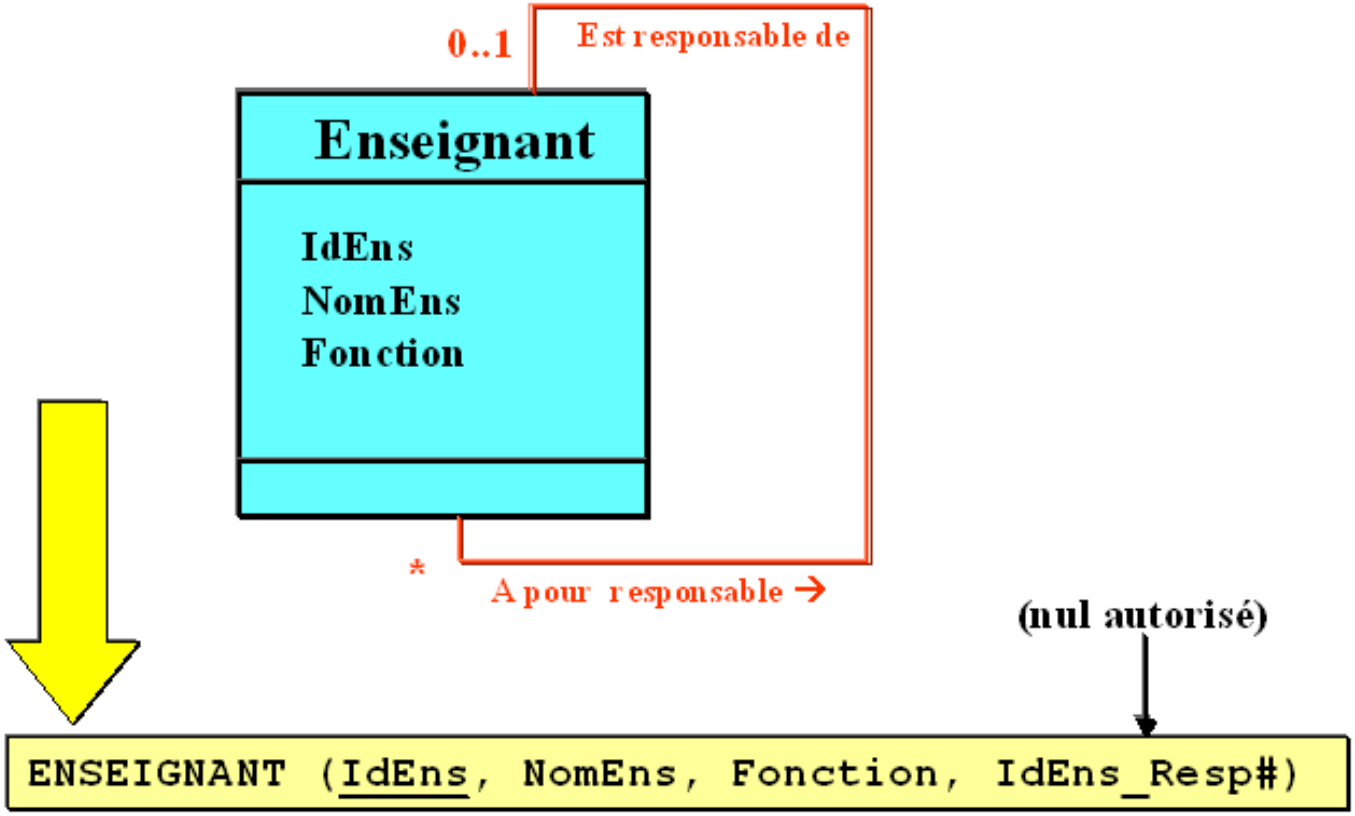


DIPLÔME (IdDip, NomDip, RespDip)

ETUDIANT (IneEtudiant, NomEtudiant, AdrEtudiant, IdDip#)

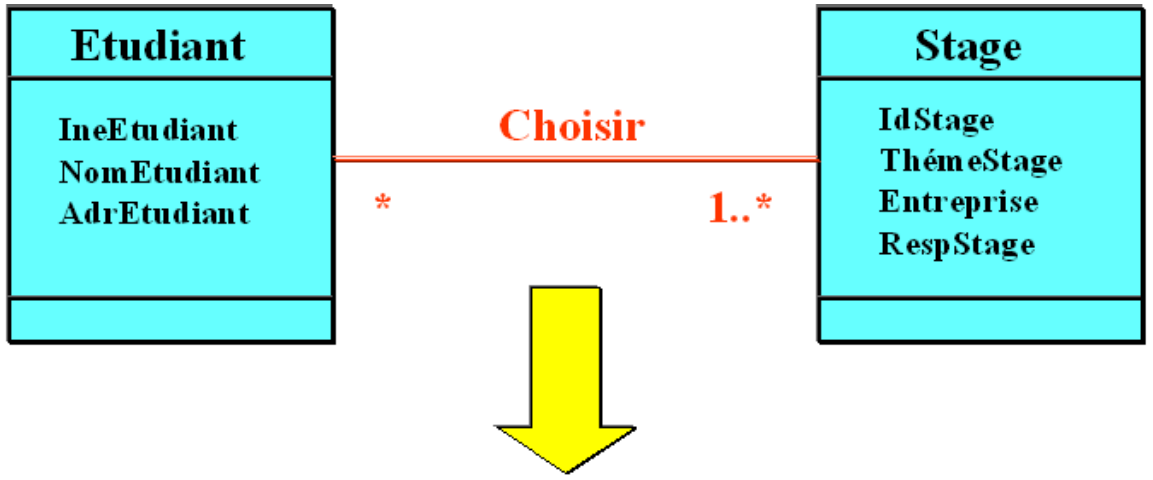
Le Passage DC vers Relationnel (suite)

- Exemple 2



Le Passage DC vers Relationnel (suite)

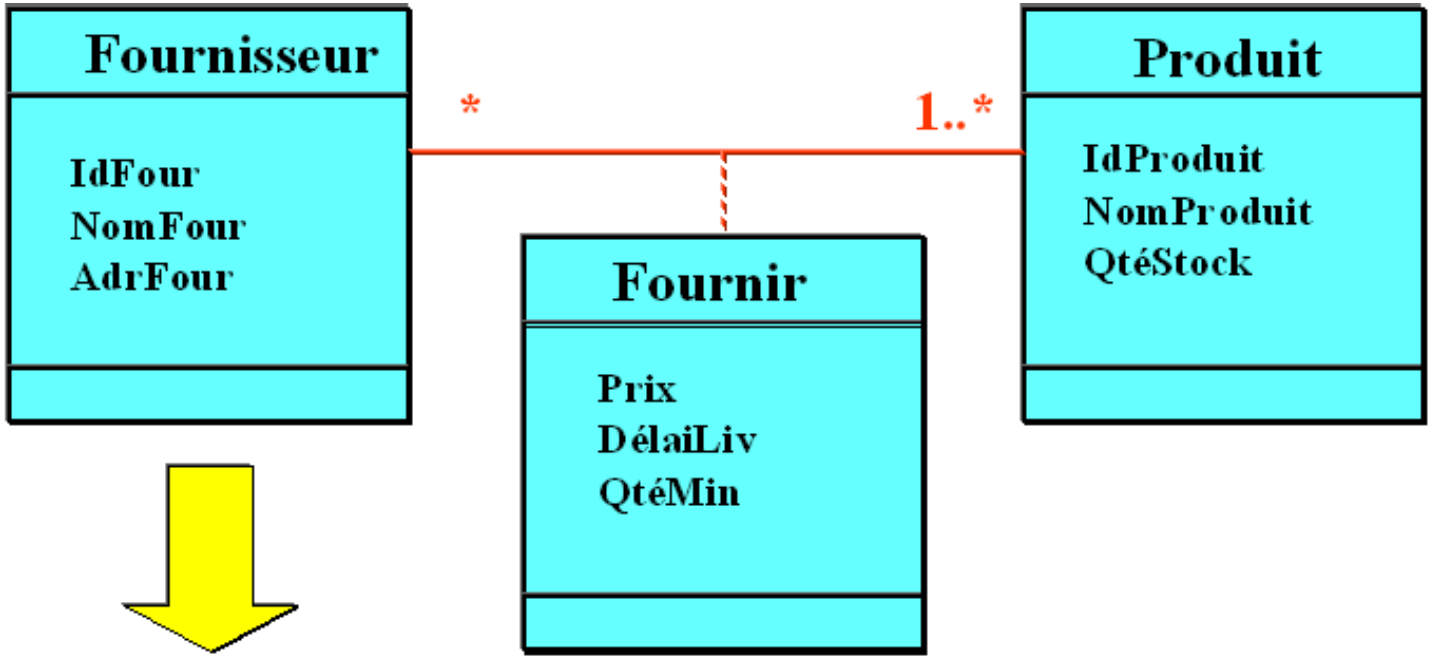
- **Règle n°3 : Classes d'Associations N-M (et *n*-aires)**
  - Cette classe devient une relation
  - La clé primaire est composée des clés associées (clé primaire composée)
- Exemple 1




```
ETUDIANT (IneEtudiant, NomEtudiant, AdrEtudiant)
CHOISIR (IneEtudiant#, IdStage# )
STAGE (IdStage, Thèmestage, Entreprise, RespStage)
```

Le Passage DC vers Relationnel (suite)

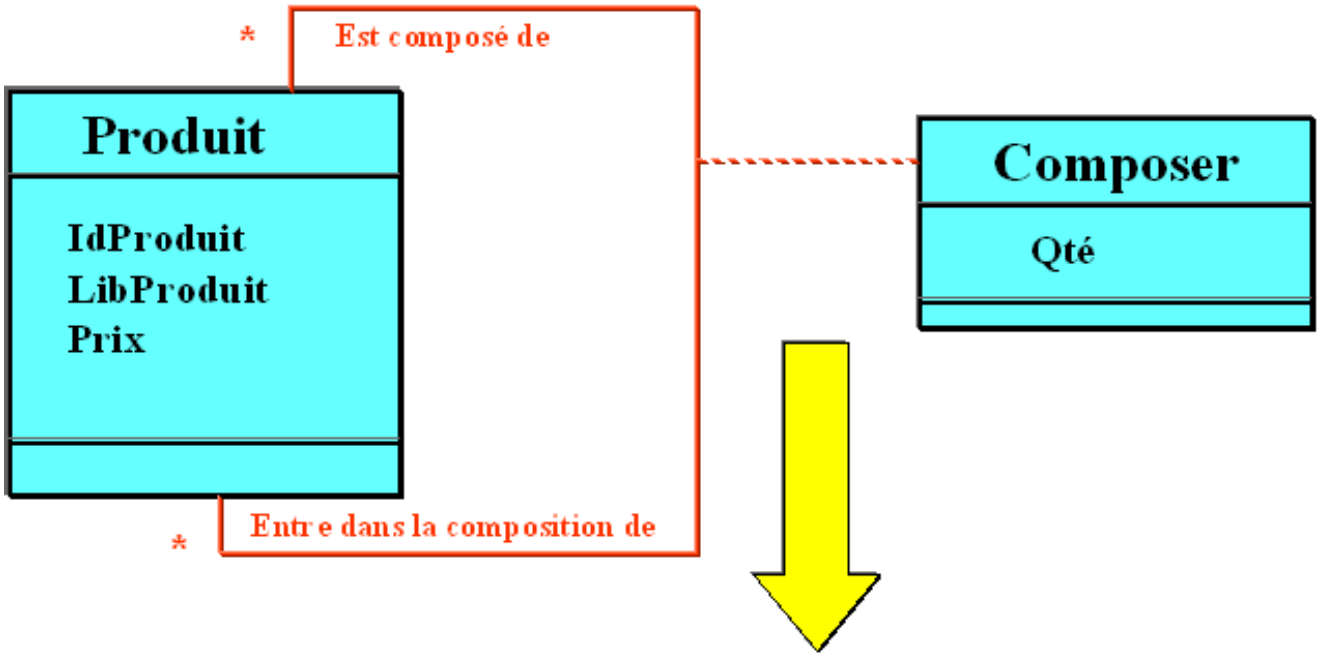
- Exemple 2




```
FOURNISSEUR(IdFour, NomFour, AdrFour)
FOURNIR (IdFour #, IdProduit#, Prix, DélaiLiv, QtéMin )
PRODUIT (IdProduit, NomProduit, QtéStock)
```

Le Passage DC vers Relationnel (suite)

- Exemple 3

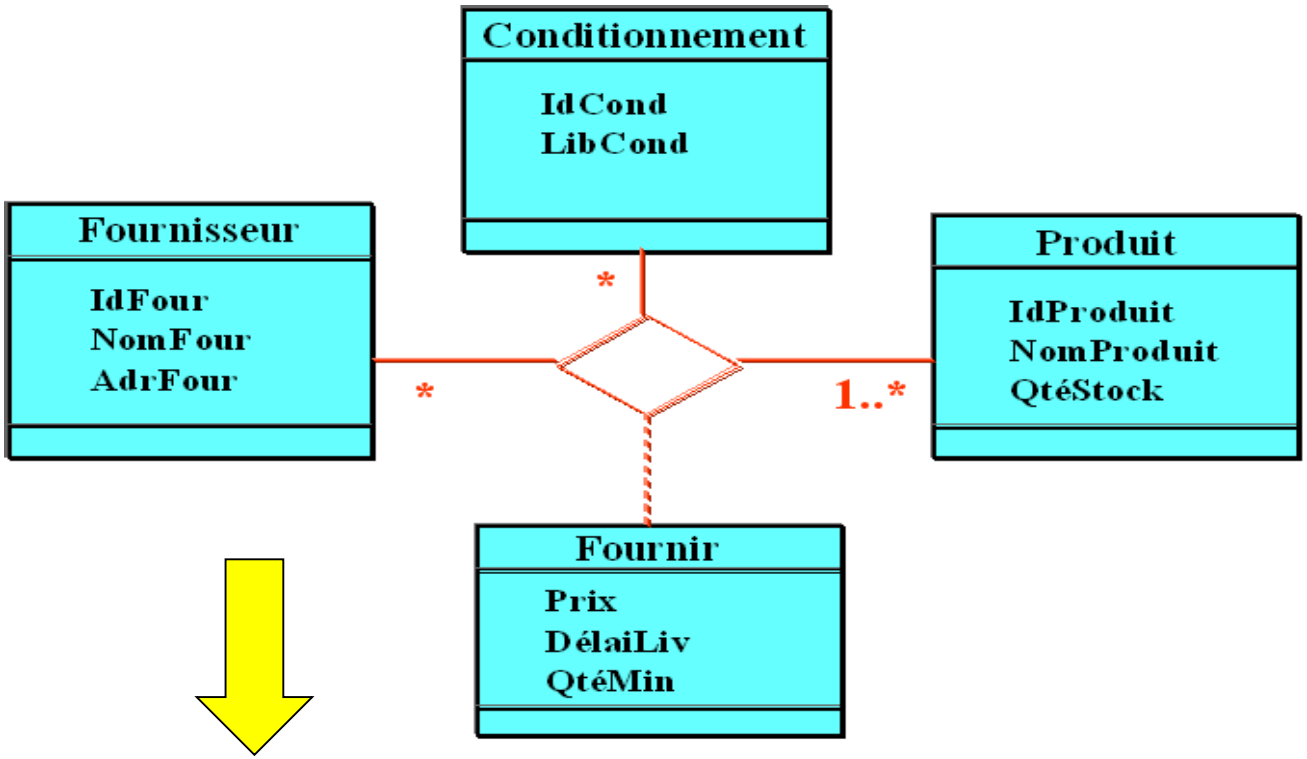



```
PRODUIT (IdProduit, LibProduit, Prix)

COMPOSER (IdProduitComposé#, IdProduitComposant#, Qté)
```

Le Passage DC vers Relationnel (suite)

- Exemple 4

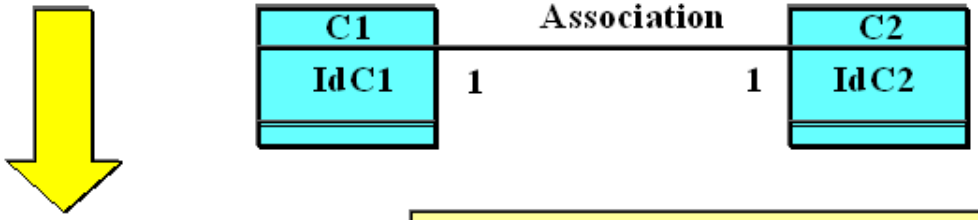



```
FOURNISSEUR(IdFour, NomFour, AdrFour)
PRODUIT (IdProduit, NomProduit, QtéStock)
CONDITIONNEMENT (IdCond, LibCond)
FOURNIR (IdFour #, IdProduit#, IdCond#,
                    Prix, DélaiLiv, QtéMin )
```

Le Passage DC vers Relationnel (suite)

Règle n°4 : Classes d'Associations 1-1

- Plusieurs cas selon les valeurs minimums mais aussi selon que le SGBD accepte les valeurs null ou pas
- Exemple



0,1----------1,1

```
C1 (IdC1, ……., IdC2#)
C2 (IdC2, ……..)
```

1,1----------0,1

```
C1 (IdC1, ……)
C2 (IdC2, ……., IdC1#)
```

0,1----------0,1

```
C1 (IdC1, ……)
C2 (IdC2, ……..)
Association (IdC1#, IdC2#)
```

Le Passage DC vers Relationnel (suite)

- Transformations des attributs et méthodes
  - **Attributs composites**
    - ✓ Pour chaque attribut composite C, comprenant N sous-attributs, d'une classe E (idem pour lesassociations), on crée N attributs correspondants à C sur la relation RE correspondant à E.
  - **Attributs multivalués**
    - ✓ Pour chaque attribut multivalué M d'une classe E (idem pour les associations), on crée une nouvelle relation RM qui comprend un attribut monovalué correspondant à M plus la clé de RE (relation représentant E). La clé de RM est la concaténation des deux attributs.
  - **Attributs dérivés et méthodes**
    - ✓ On ne représente pas en général les attributs dérivés ni les méthodes dans le modèle relationnel, ils seront calculés dynamiquement soit par des procédures internes à la BD (procédures stockées), soit par des procédures au niveau applicatif.

Le Passage DC vers Relationnel (suite)

- Transformations des compositions
  - Une composition est transformée comme une association 1:N, mais on ajoute à la clé de la classe partie (dite clé locale) la clé étrangère vers la classe composite.
  - Soit la composition entre la classe composite C et la classe partie P (représentés par les relations RC et RP respectivement) on inclut dans la définition de RP comme clé étrangère la clé de RC. La clé de RP est redéfinie comme la concaténation de la clé de P (clé locale) avec la clé étrangère vers RC.
  - **Remarque : Composition et entités faibles en E-A**
    - ✓ Une composition est transformée selon les mêmes principes qu'une entité faible en E-A.

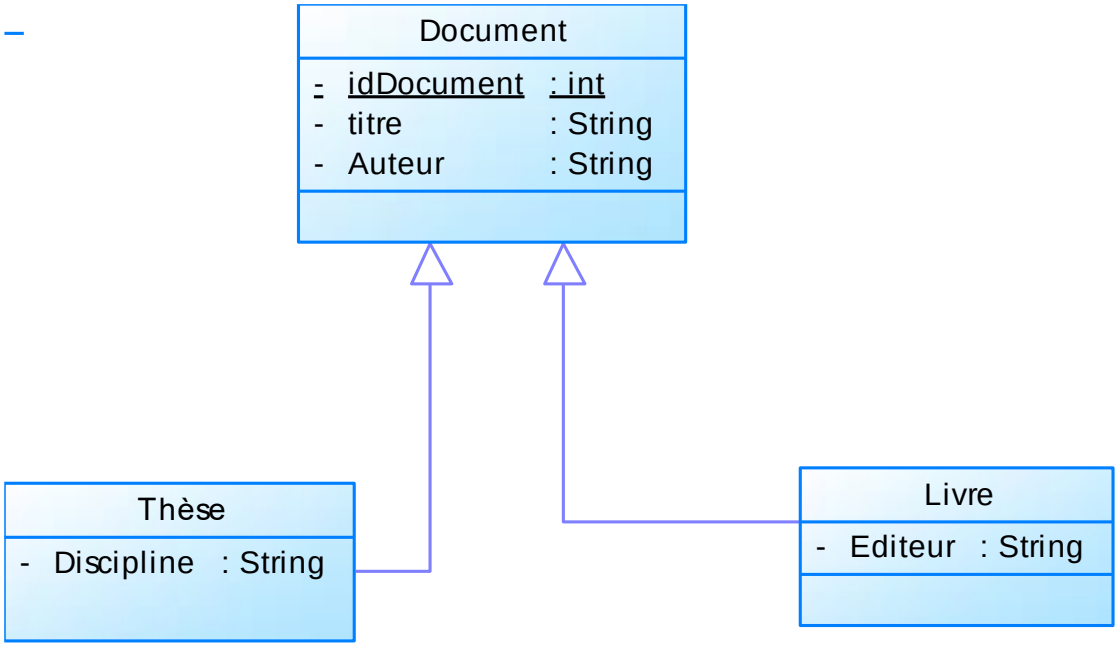
Le Passage DC vers Relationnel (suite)

Transformation de la relation d'héritage

- Le modèle relationnel ne permet pas de représenter directement une relation d'héritage, puisque que seuls les concepts de relation et de référence existent dans le modèle. Il faut donc appauvrir le modèle conceptuel pour qu'il puisse être représenté selon un schéma relationnel.
- Trois solutions existent pour transformer une relation d'héritage :
  1. Représenter l'héritage par une référence entre la super classe et la sous classe.
  2. Représenter uniquement les sous classes par des relations.
  3. Représenter uniquement la super classe par une relation.

Le Passage DC vers Relationnel (suite)

- Exemple



Le Passage DC vers Relationnel (suite)

- Héritage représenté par une référence
  - Chaque classe, super-classe ou sous-classe est représentée par une relation. La clé primaire de la super-classe est utilisée pour identifier chacune des sous-classes (cette clé étant pour chaque sous-classe à la fois la clé primaire et une clé étrangère vers la super-classe).
  - Exemple:

Document(**id_document**, Titre:Chaîne, Auteur:Chaîne)
These((**id_documen#,** Discipline:Chaîne)
Livre((**id_document#,** Editeur:Chaîne)

Le Passage DC vers Relationnel (suite)

- Héritage absorbé par les sous classes
  - Chaque sous-classe est représentée par une relation, mais la super-classe n'est pas représentée par une relation. Tous les attributs de la super-classe sont répétés au niveau de chaque sous-classe.
  - C'est la clé primaire héritée de la super-classe qui devient la clé primaire de chaque sous-classe.
  - Exemple:

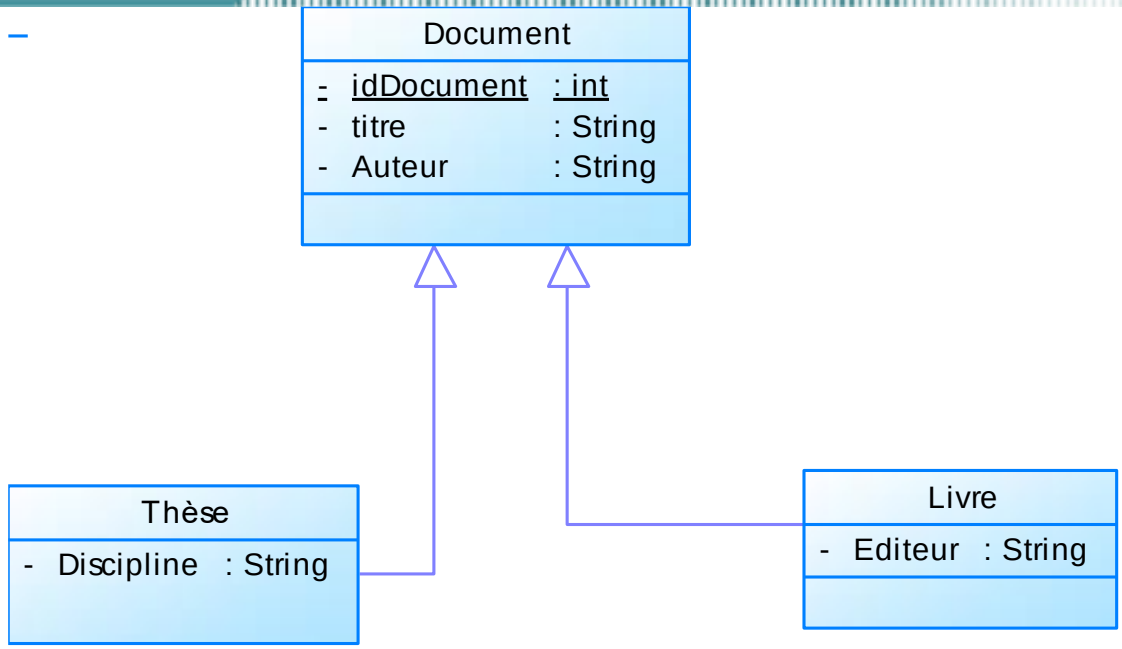
These(**id_document,** Titre, Discipline:Chaîne, Auteur:Chaîne)
Livre(**id_document,**Titre, Editeur:Chaîne, Auteur:Chaîne)

Le Passage DC vers Relationnel (suite)

- Héritage absorbé par la super classe
  - La super-classe est représentée par une relation, mais les sous-classes ne sont pas représentés par des relations. Tous les attributs de chaque sous-classe sont réintégrés au niveau de la super-classe.
  - Un attribut supplémentaire, dit de discrimination, est ajouté à la super-classe, afin de distinguer les tuples des différentes sous-classes. Cet attribut est de type énumération et a pour valeurs possibles les noms des différentes sous-classes.
  - Exemple:

Document(**id_document,**Titre, Discipline:Chaîne, Editeur:Chaîne, Auteur:Chaîne,Type:{These|Livre})

Le Passage DC vers Relationnel (suite)



Remarque : Type d'héritage

Si une thèse peut également être un livre (et dans la réalité c'est bien le cas puisqu'une thèse peut-être publiée), alors l'héritage n'est pas exclusif puisque un même document peut être une thèse *et un livre.*

L'héritage n'est pas non plus complet, puisque l'on observe que les thèses et les livres ont des attributs qui ne sont pas communs (la discipline pour la thèse et l'éditeur pour le livre).

S'il n'existe pas de document qui ne soit ni thèse ni livre alors la classe mère est abstraite

Le Passage DC vers Relationnel (suite)

Transformation	Héritage complet	Héritage exclusif	Ni complet Ni exclusif
Par références	non	non	Oui (surtout si la super classe n'est pas abstraite)
Absorbé par les sous classes	non	Oui (surtout si la super classe est abstraite)	non
Absorbé par la super classe	Oui (surtout si la super classe n'est pas abstraite)	non	non