Introduction à la modélisation conceptuelle de données avec UML

Table des matières

Cours

La **modélisation conceptuelle** est l'étape **fondatrice** du processus de conception de *BD∗*. Elle consiste à abstraire le problème réel posé pour en faire une reformulation qui trouvera une solution dans le cadre technologique d'un *SGBD∗*.

Si le modèle dominant en conception de bases de données a longtemps été le modèle *E-A∗*, le modèle *UML∗* se généralise de plus en plus. Nous proposons ici une introduction au diagramme de classes à travers la représentation de classes et d'associations simples (il existe d'autres diagrammes UML, par exemple le diagramme de cas, et d'autres primitives de représentation dans le diagramme de classe, par exemple l'héritage).

1. Notion de modèle

Objectifs

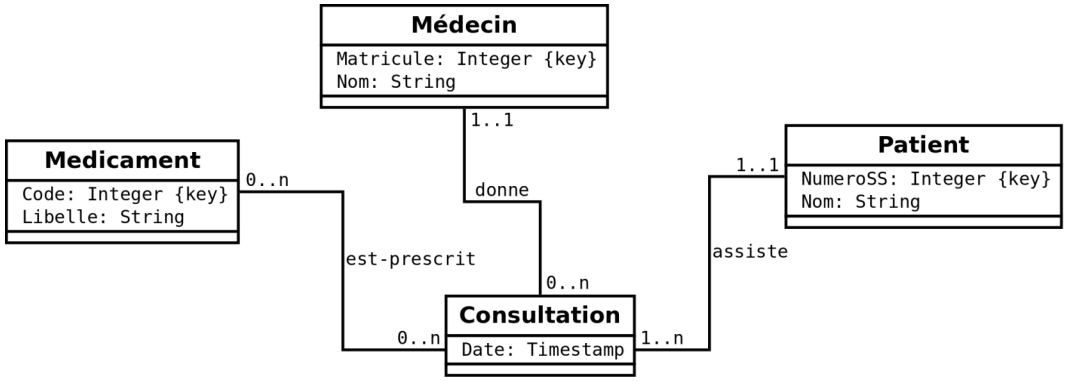
Savoir ce qu'est un modèle

Savoir ce qu'est le langage UML

1.1. Exercice : Centre médical

[10 min]

Soit le modèle conceptuel suivant représentant des visites dans un centre médical. Quelles sont les assertions vraies selon ce schéma ?



- ☐ Un patient peut effectuer plusieurs visites.
- ☐ Tous les patients ont effectué au moins une consultation.
- ☐ Un médecin peut recevoir plusieurs patients pendant la même consultation.
- ☐ Un médecin peut prescrire plusieurs médicaments lors d'une même consultation.
- ☐ Deux médecins différents peuvent prescrire le même médicament.

1.2. Qu'est ce qu'un modèle ?

Modèle *Définition*

« Modeling, in the broadest sense, is the cost-effective use of something in place of something else for some cognitive purpose. It allows us to use something that is simpler, safer or cheaper than reality instead of reality for some purpose. A model represents reality for the given purpose; the model is an abstraction of reality in the sense that it cannot represent all aspects of reality. » (*Rothenberg, 1989*∗, cité par *Arribe, 2014*∗)

« Système physique, mathématique ou logique représentant les structures essentielles d'une réalité et capable à son niveau d'en expliquer ou d'en reproduire dynamiquement le fonctionnement. » (TLFi)

Modèle *Fondamental*

Un modèle est une représentation simplifiée de la réalité en vue de réaliser quelque chose.

1.3. Qu'est ce qu'un modèle en informatique ?

Modèle informatique *Définition*

Un modèle informatique est une représentation simplifiée de la réalité en vue de réaliser un traitement avec un ordinateur.

Numérisation et abstraction : Toute information numérique a été codée selon un modèle donné *Complément*

« Tout numérisation est une représentation de la réalité sous la forme d'une modélisation numérique. Cette modélisation procède d'une abstraction au sens où c'est une séparation d'avec le réel, au sens où c'est une construction destinée à la manipulation (algorithmique en l'occurrence) et au sens où c'est une simplification de la réalité. »

http://aswemay.fr/co/tropism-pres.html

1.4. Qu'est ce qu'un bon modèle ?

Attention

Un modèle est une abstraction, une simplification de la réalité, ce n'est pas la réalité, il n'est jamais complètement fidèle par construction.

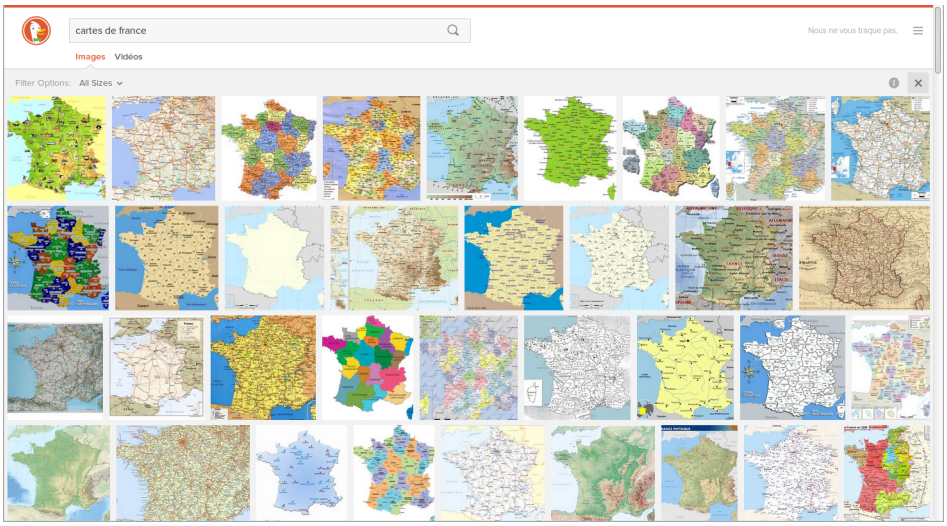
Le seul modèle complètement fidèle à la réalité est la réalité elle-même, et ce n'est donc pas un modèle.

La carte et le territoire *Exemple*

Une carte est un modèle d'un territoire. Elle est une représentation simplifiée destiné à un usage particulier :

- randonner à pied, en vélo ;
- se diriger en voiture sur des grands axes, sur des axes secondaires ;
- voler en avion de tourisme, en avion de ligne ;
- naviguer sur fleuve, sur mer ;
- étudier les frontières d'une région, d'un pays, de la terre ;

- étudier la démographie, l'économie... ;
- ...



Fondamental

À partir de cet exemple on notera que :

1. **Un modèle est orienté par un usage.**

   Chacune de ces cartes est très différente selon ce que l'on veut faire.

2. **Un modèle ne cherche pas à être proche de la réalité.**

   Chacune de ces cartes est très différente de la réalité qu'elle représente.

3. **Un modèle adresse un niveau d'information qui existe mais qui n'est pas accessible dans la réalité.**

   Chacune de ces cartes permet quelque chose que ne permet pas l'accès direct à la réalité.

Le rasoir d'Ockham : Entre deux modèles donnés le meilleur modèle est-il toujours le plus fourni ?

Méthode

La méthode de raisonnement connue sous le nom de rasoir d'Ockham (du nom du philosophe éponyme) consiste à préférer les solutions les plus simples aux plus complexes, lorsqu'elles semblent permettre également de résoudre un problème donné ; entre deux théories équivalentes, toujours préférer la plus simple.

Ce principe s'applique très bien à la modélisation : étant donné un objectif et plusieurs modèles possibles, il ne faut pas choisir a priori celui qui représente le plus de choses, mais préférer le plus simple dès qu'il couvre le besoin.

C'est un principe d'économie (il coûte moins cher à produire) et d'efficacité (car les éléments inutiles du modèle plus fourni nuiront à l'efficacité de la tâche).

Exemple

Ainsi, pour naviguer en voiture, il est plus simple de ne **pas** avoir sur la carte les chemins de randonnées qui ne sont pas praticables en voiture.

2. Introduction au diagramme de classes UML : classes et associations

Objectifs

Savoir faire un modèle conceptuel.

Savoir interpréter un modèle conceptuel.

2.1. Exercice : Lab I

Description du problème

[15 min]

Un laboratoire souhaite gérer les médicaments qu'il conçoit.

- Un médicament est décrit par un nom, qui permet de l'identifier. En effet il n'existe pas deux médicaments avec le même nom. Un médicament comporte une description courte en français, ainsi qu'une description longue en latin. On gère aussi le conditionnement du médicament, c'est à dire le nombre de pilules par boîte (qui est un nombre entier).
- À chaque médicament on associe une liste de contre-indications, généralement plusieurs, parfois aucune. Une contre-indication comporte un code unique qui l'identifie, ainsi qu'une description. Une contre-indication est toujours associée à un et un seul médicament.

Exemple de données

Afin de matérialiser notre base de données, nous obtenons les descriptions suivantes :

- Le **Chourix** a pour description courte *« Médicament contre la chute des choux »* et pour description longue *« Vivamus fermentum semper porta. Nunc diam velit, adipiscing ut tristique vitae, sagittis vel odio. Maecenas convallis ullamcorper ultricies. Curabitur ornare. »*. Il est conditionné en boîte de 13.

  Ses contre-indications sont :

  - CI1 : Ne jamais prendre après minuit.
  - CI2 : Ne jamais mettre en contact avec de l'eau.
- Le **Tropas** a pour description courte *« Médicament contre les dysfonctionnements intellectuels »* et pour description longue *« Suspendisse lectus leo, consectetur in tempor sit amet, placerat quis neque. Etiam luctus porttitor lorem, sed suscipit est rutrum non. »*. Il est conditionné en boîte de 42.

  Ses contre-indications sont :

  - CI3 : Garder à l'abri de la lumière du soleil

Question 1

Réaliser le modèle conceptuel de données en UML du problème.

Question 2

Étendre le modèle conceptuel UML afin d'ajouter la gestion des composants. Un composant est identifié par un code unique et possède un intitulé. Tout médicament possède au moins un composant, souvent plusieurs. Tout composant peut intervenir dans la fabrication de plusieurs médicaments. Il existe des composants qui ne sont pas utilisés pour fabriquer des médicaments et que l'on veut quand même gérer.

2.2. Présentation d'UML

UML∗ est un langage de représentation destiné en particulier à la modélisation objet. UML est devenu une norme *OMG*∗ en 1997.

UML propose un formalisme qui impose de "penser objet" et permet de rester indépendant d'un langage de programmation donné. Pour ce faire, UML normalise les concepts de l'objet (énumération et définition exhaustive des concepts) ainsi que leur notation graphique. Il peut donc être utilisé comme un moyen de communication entre les étapes de spécification conceptuelle et les étapes de spécifications techniques.

Diagramme de classe — ***Fondamental***

Le diagramme de classes est un sous ensemble d'UML qui s'attache à la description statique d'un modèle de données représentées par des classes d'objets.

Remarque

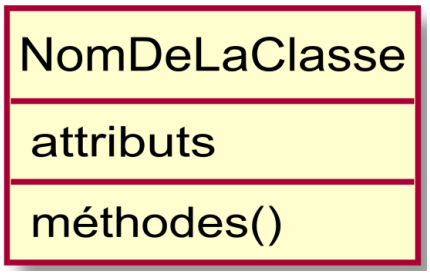
Dans le domaine des bases de données, UML peut être utilisé à la place du modèle *E-A*∗ pour modéliser le domaine. De la même façon, un schéma conceptuel UML peut alors être traduit en schéma logique (relationnel ou relationnel-objet typiquement).

2.3. Classes

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

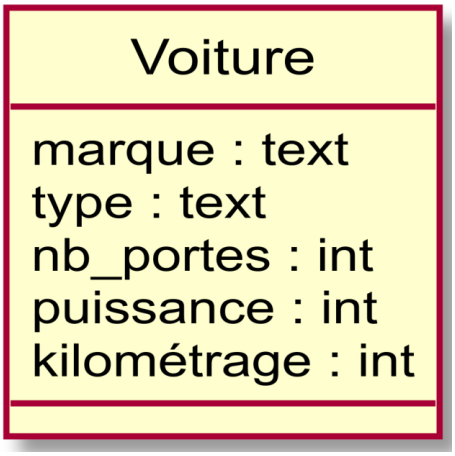


Représentation UML d'une classe

Exemple

La classe Voiture

Exemple



La classe Voiture (avec attributs)

Une instance de la classe Voiture

Exemple

L'objet V1 est une instance de la classe Voiture.

V1 : Voiture

- Marque : 'Citroën'
- Type : 'Visa'
- Portes : 5
- Puissance : 60
- Kilométrage : 300000

Complément

La modélisation sous forme de diagramme de classes est une modélisation statique, qui met en exergue la structure d'un modèle, mais ne rend pas compte de son évolution temporelle. UML propose d'autres types de diagrammes pour traiter, notamment, de ces aspects.

2.4. Attributs

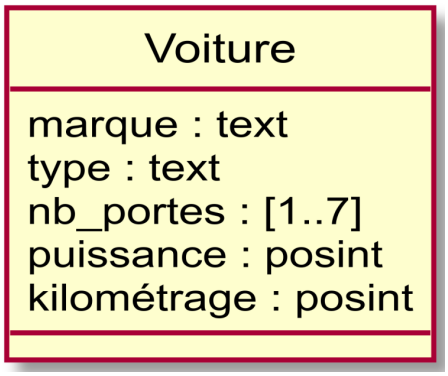
Attribut

Définition

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

Un attribut est typé : Le domaine des valeurs que peut prendre l'attribut est fixé a priori.

Exemple



La classe Voiture (avec attributs et types précis)

Types *Syntaxe*

Le typage des attributs peut se faire dans une syntaxe libre, il n'est exprimé que pour :

- consigner des contraintes,
- expliciter la nature de l'attribut.

On peut utiliser une terminologie française, anglais, inventée, liée à un langage, etc. On veillera néanmoins à conserver une terminologie cohérente.

Précision des types *Conseil*

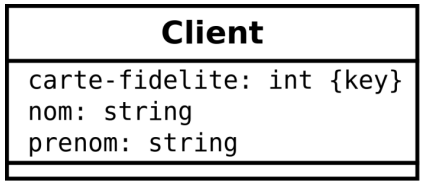
On conseille d'adopter sur le schéma un typage précis qui rend compte de ce qui est exprimé dans l'analyse des besoins. Si l'on sait qu'une voiture ne peut pas avoir plus de sept portes, on le consigne dès l'UML dans le typage.

2.5. Repérage des clés

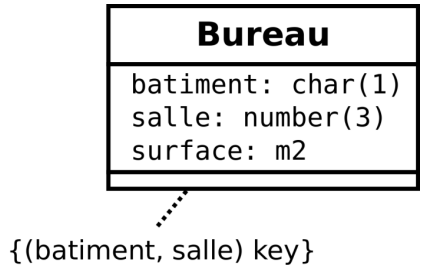
Un attribut ou un groupe d'attributs peut être annoté comme étant **clé** s'il permet d'identifier de façon unique un objet de la classe.

On ajoute le symbole `{key}` à côté du ou des attributs concernés.

Exemple



Clé en UML



Clé composée de deux attributs

Méthode

Le repérage des clés n'est pas systématique en UML (la définition des clés se fera essentiellement au niveau logique). On cherchera néanmoins à repérer les clés rendues évidentes par la phase de clarification.

Une classe peut très bien ne pas avoir de clé repérée en UML.

Attention

On n'ajoutera jamais de *clé artificielle*∗ au niveau du MCD. Si aucune clé n'est évidente, on laisse la classe sans clé.

Notation {unique}

Remarque

Les notation `{key}` et `{unique}` sont équivalente en UML}

Attribut souligné et #

Remarque

On trouvera dans ce cours des exemples d'attributs soulignés ou précédés de # pour exprimer l'unicité. Ce n'est pas une pratique standard et la notation `{key}` devrait lui être substituée.

- Un attribut souligné est normalement un attribut de classe, ou *static*, en UML,
- Un attribut précédé de # est normalement un attribut protégé en UML.

Mais les concepts d'attribut de classe et d'attribut protégé ne sont pas utilisés dans le cadre des bases de données.

2.6. Méthodes

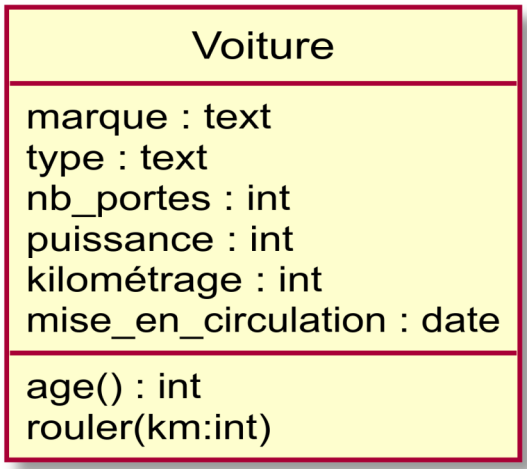
Méthode

Définition

Une méthode (ou opération) est une fonction associée à une classe d'objet qui permet d'agir sur les objets de la classe ou qui permet à ces objets de renvoyer des valeurs (calculées en fonction de paramètres).

Syntaxe

Les méthodes se notent sous les attributs.



La classe voiture (avec méthodes)

Méthodes et modélisation de BD *Méthode*

Pour la modélisation des bases de données, les méthodes sont surtout utilisées pour :

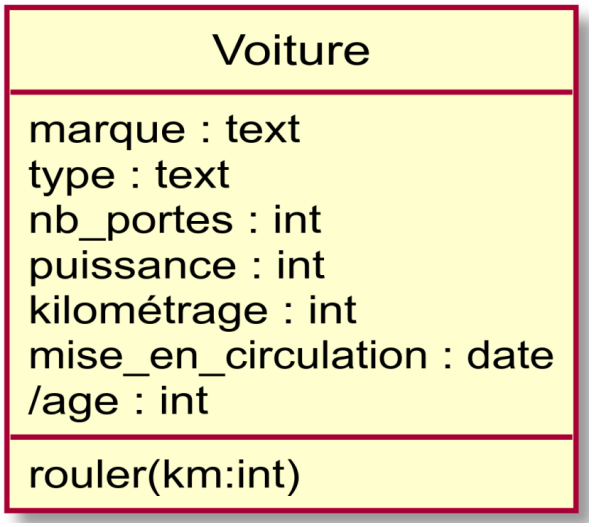
- représenter des données calculées : exemple de l'age calculé à partir de la date de mise en circulation ;
- ou pour mettre en exergue des fonctions importantes du système cible : exemple de la fonction rouler qui insiste sur la fonction de mise à jour du kilométrage (qui devra être particulièrement considérée dans l'application cible).

Seules les méthodes les plus importantes sont représentées (l'approche est moins systématique qu'en modélisation objet par exemple).

Attributs dérivés *Complément*

La notation d'attributs dérivés (importée du modèle E-A en UML) consiste à faire précédé d'un / un nom d'attribut pour montrer que cet attribut est calculé à partir d'autres attributs de la même classe.

C'est donc une alternative aux méthodes dans ce cas. Mais en UML on préfère l'usage de méthodes aux attributs dérivés.



La classe voiture (avec attribut dérivé)

Complément

Transformation des méthodes par des vues (cf. p.24)

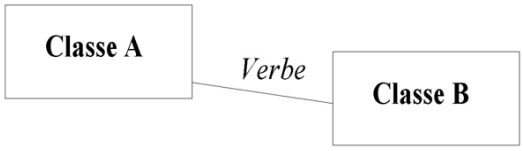
2.7. Associations

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).

Syntaxe



Notation de l'association en UML

Attention

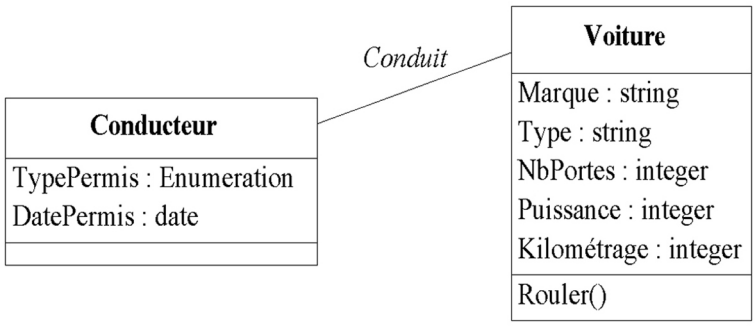
Le nom de l'association (verbe qui la décrit) est obligatoire, au même titre que le nom d'une classe ou d'un attribut.

Remarque

Une association est généralement bidirectionnelle (c'est à dire qu'elle peut se lire dans les deux sens). Les associations qui ne respectent pas cette propriété sont dites unidirectionnelles ou à navigation restreinte.

L'association Conduit

Exemple



Représentation d'association en UML

L'association Conduit entre les classes Conducteur et Voiture exprime que les conducteurs conduisent des voitures.

Voir aussi

Complément

- *Cardinalité* *(cf. p.12)*
- *Explicitation des associations* *(cf. p.24)*
- *Associations ternaires* *(cf. p.25)*
- *Contraintes standardisées sur les associations* *(cf. p.25)*

2.8. Cardinalité

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 *.

Attention

La notation de la cardinalité en UML est opposée à celle adoptée en E-A. En UML on note à gauche (resp. à droite) le nombre d'instances de la classe de gauche (resp. de droite) autorisées dans l'association. En E-A, on note à gauche (resp. à droite) le nombre d'instances de la classe de droite (resp. de gauche) autorisées dans l'association.

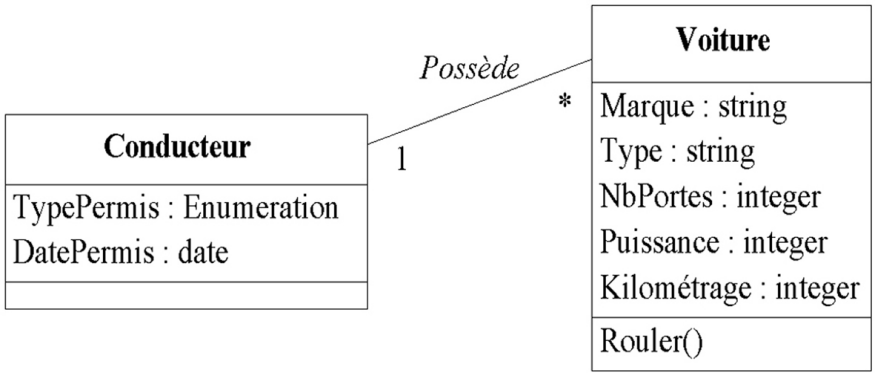
Remarque

Les cardinalités les plus courantes sont :

- 0..1 (optionnel)
- 1..1 ou 1 (un)
- 0..n ou 0..* ou * (plusieurs)
- 1..n ou 1..* (obligatoire)

La cardinalité de l'association Possède

Exemple



Représentation de cardinalité en UML

Ici un conducteur peut posséder plusieurs voitures (y compris aucune) et une voiture n'est possédée que par un seul conducteur.

Terminologie

Fondamental

- On appelle association 1:1 les associations de type :
  - 0..1:0..1
  - 0..1:1..1
  - 1..1:0..1
  - 1..1:1..1

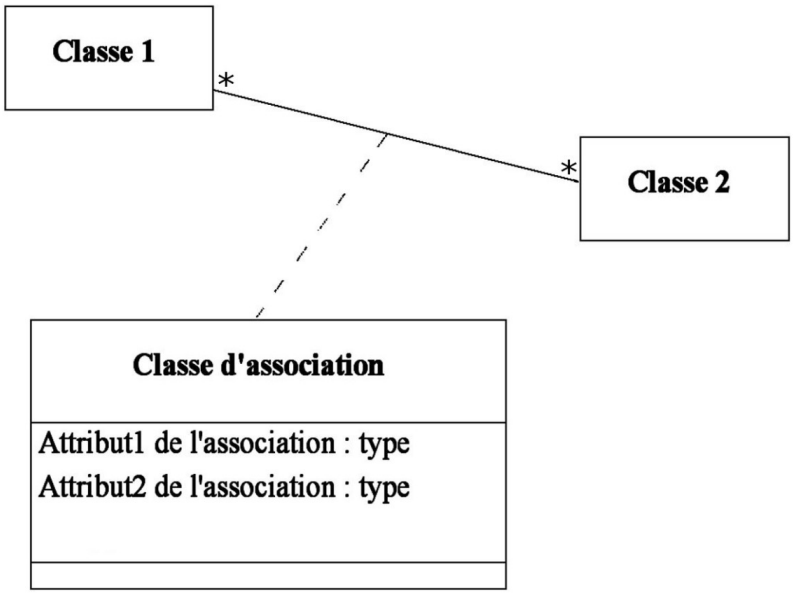
- On appelle association 1:N les associations de type :
  - 0..1:0..N
  - 0..1:1..N
  - 1..1:0..N
  - 1..1:1..N
- On appelle association N:M (ou M:N) les associations de type :
  - 0..N:0..N
  - 0..N:1..N
  - 1..N:0..N
  - 1..N:1..N

2.9. Classe d'association

Classe d'association *Définition*

On utilise la notation des classes d'association lorsque l'on souhaite ajouter des propriétés à une association.

Notation d'une classe d'association en UML *Syntaxe*



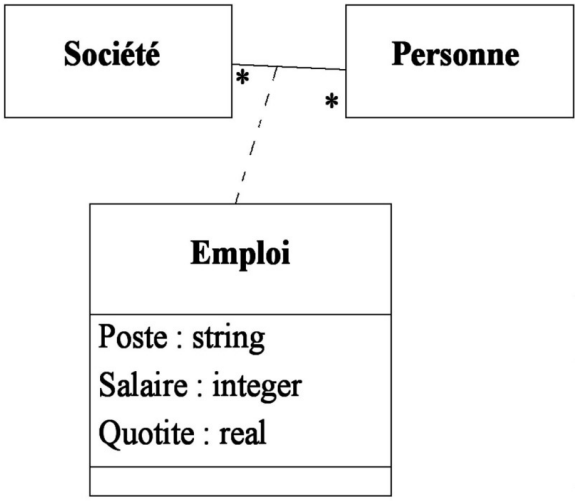
Notation d'une classe d'association en UML

Méthode

On réserve en général les classes d'association aux associations N:M.

Il est toujours possible de réduire une classe d'association sur une association 1:N en migrant ses attributs sur la classe côté N, et c'est en général plus lisible ainsi.

Exemple de classe d'association *Exemple*



Emplois

Conseil

Selon le standard UML une classe d'association est une classe et à ce titre elle peut être mobilisée dans d'autres associations ou dans des héritages. Nous déconseillons néanmoins ces notations qui ont tendance à complexifier la lecture et la transformation du diagramme.

Nous conseillons donc de ne jamais associer une classe d'association.

3. Quels outils pour faire de l'UML ?

3.1. Outils pour l'élaboration (papier/crayon)

Fondamental

L'élaboration d'un schéma UML est d'abord un outil permettant de chercher une solution, il est donc **indispensable** qu'il soit facile de le modifier de changer d'avis, et que sa manipulation ne soit pas parasitée par de la manipulation informatique.

Papier/crayon *Conseil*

Pour élaborer un schéma UML la meilleure méthode est de se doter :

- d'une grande feuille (au minimum un A3 pour un petit projet de quelques classes) ;
- d'un crayon de papier et d'une gomme

On peut aussi travailler au tableau (noir, blanc, numérique...).

Attention

Un diagramme UML, même pour l'élaboration doit rester parfaitement visible, c'est un outil **graphique** et c'est un outil de **dialogue** s'il est mal lisible on fera des erreurs et il remplira mal son rôle.

Éditeur graphique *Conseil*

Si l'on est doté d'une bonne expérience sur un outil **graphique** alors il est possible de l'utiliser à la place du papier crayon, mais :

- il faut bien connaître l'outil pour que son attention soit portée sur la conception et non sur le fonctionnement de l'outil ;
- il faut que l'outil soit totalement souple pour permettre une expression graphique non contrainte (typiquement certains éditeurs contraignent la création afin de rester en mesure de faire de la génération automatique de code, ce qui est hors-sujet dans notre cas).

3.2. Dia

Dia est un petit outil graphique libre et multi-plate-forme.

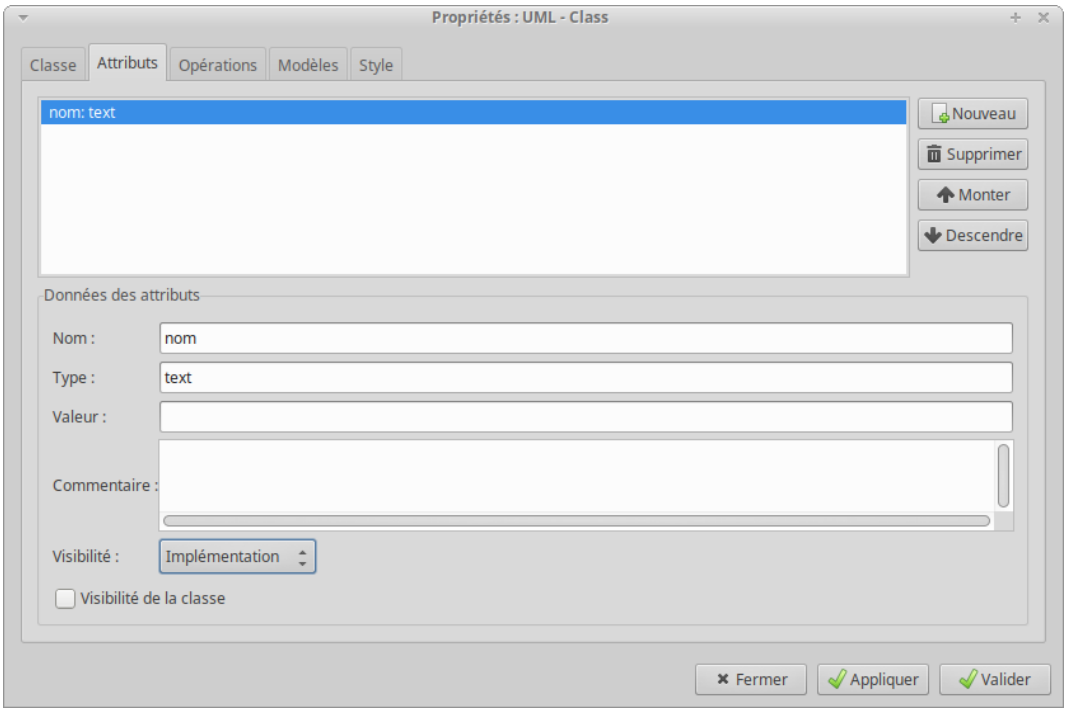
- https://fr.wikipedia.org/wiki/Dia_(logiciel)
- http://dia-installer.de/

Il présente pas mal de défauts, mais il est assez facile à prendre en main et il est possible d'utiliser le mode graphique pour dessiner ce qui n'est pas prévu dans le module UML.

Ne pas afficher la visibilité des attributs *Méthode*

Dia fixe par défaut la visibilité de tous les attributs à la valeur "Public", ce qui a comme conséquence de faire précéder l'attribut d'un caractère "+".

C'est un concept qu'on utilise pas en base de données relationnelle, cela surcharge donc inutilement tous les diagrammes. Il faut donc systématiquement modifier cette propriété.



Mettre la visibilité des attributs à la valeur "implémentation"

Export en PNG *Méthode*

Utiliser la fonction `Fichier > Export` pour transformer votre diagramme en fichier PNG.

Choisissez une résolution assez élevée si vous visez une impression papier.

3.3. Outils pour la documentation (éditeur)

Pour la documentation de la conception d'une base de données, on est souvent amené à livrer le MCD en UML.

Diagramme papier/crayon propre *Conseil*

On peut tout à fait livrer un UML fait à la main sur papier.

On veillera néanmoins à ce que le diagramme soit très propre et très clair et à ce que la numérisation soit de qualité (on préférera l'usage d'un scanner à celui d'un ordiphone, et si on prend une photo ce sera une photo de qualité, plane et parfaitement recadrée).

Attention

Le diagramme UML livré en documentation ne doit pas ressembler à un brouillon, mais bien à un document de référence.

PlantUML *Conseil*

On conseille pour la documentation l'usage du logiciel PlantUML.

Combiné avec un dépôt Git il permet d'historiser la dynamique de conception.

3.4. PlantUML

PlantUML est un outil libre et multi-plate-forme qui permet de créer des diagrammes UML à partir d'un langage textuel de description.

- http://plantuml.com/fr/class-diagram (*PDF (cf. PlantUML_Language_Reference_Guide_fr_diagramme_de_classe)*)
- https://en.wikipedia.org/wiki/PlantUML

Exemple

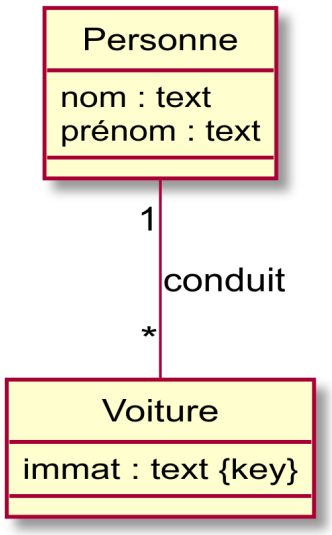
```
@startuml
hide circle

class Voiture {
  immat : text {key}
}

class Personne {
  nom : text
  prénom : text
}

Personne "1" -- "*" Voiture  : conduit

@enduml
```



hide circle

Méthode

La commande `hide circle` permet de supprimer des éléments de syntaxe propre à certaines représentations objet, non utilisés en base de données.

Notation des clés composées

Méthode

```
class Employé {
  nom : text
  prénom : text
  ddn : date
  date_embauche : date
  quotite : pourcentage
}
note left : (nom, prénom, ddn) unique
```

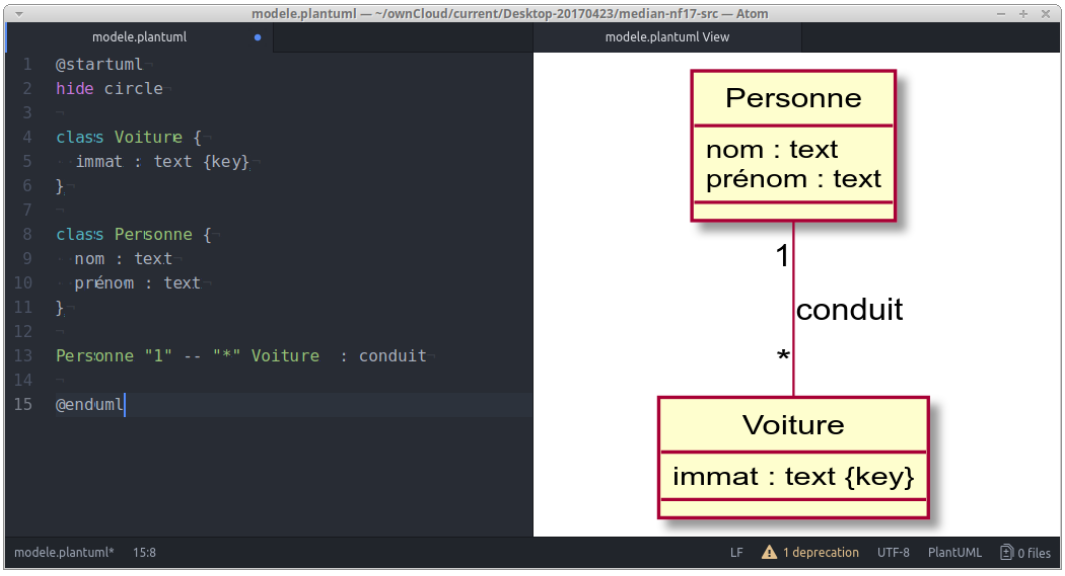
Atom et PlantUML

PlantUML peut-être intégré à Atom avec les plugin `plantuml-viewer` et `language-plantuml`.

```
apm install plantuml-viewer language-plantuml
```

PlantUML dans Atom

Exemple



Positionnement vertical / horizontal

Remarque

Notez la différence entre :

- `-` : à côté (un seul tiret)
- `--` : en dessous (deux tirets ou plus)

L'ordre de déclaration des classes est pris en compte.

Ajustement de la longueur des associations avec PlantUML

Complément

- Horizontal : `Classe1 - Classe2 : "  Association  "` (on ajoute des espaces)
- Vertical : `Classe1 ------ Classe2 : "Association"` (on ajoute des tirets)

Ajustement du positionnement avec PlantUML

Complément

L'usage d'association `[hidden]` permet d'améliorer un peu la présentation d'un diagramme PlantUML (à utiliser avec parcimonie).

La bonne solution est plutôt l'usage de **package**.

```
@startuml

class Alice
class Bob
class Charlie
class Dan
class Eugene
class Foobar

Bob -[hidden] Alice : ré-ordonnancement
Alice -[hidden] Charlie : éloignement horizontal_________________________
Dan --[hidden] Eugene : positionnement vertical
Alice --------[hidden] Foobar : éloignement vertical

@enduml
```

Contournements pour exprimer des contraintes d'association avec PlantUML

Complément

```
Alice -- Bob : Assoc1
note right on link
{XOR}
end note
Alice -- Bob : Assoc2
```

```
Alice -- Bob : Assoc1....{XOR}....
Alice -- Bob : Assoc2
```

```
Alice -- Bob : Assoc1
Alice -- Bob : Assoc2
note "{XOR}" as N #white
(Alice,Bob) .. N
N .. (Alice,Bob)
```

Commentaires

Complément

Toute ligne qui commence par un `'` est un commentaire, non interprété par le moteur.

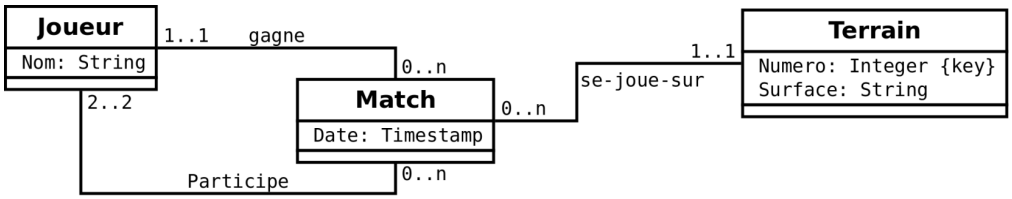
Exercices

1. Exercice : Lire l'UML

[30 min]

Tennis

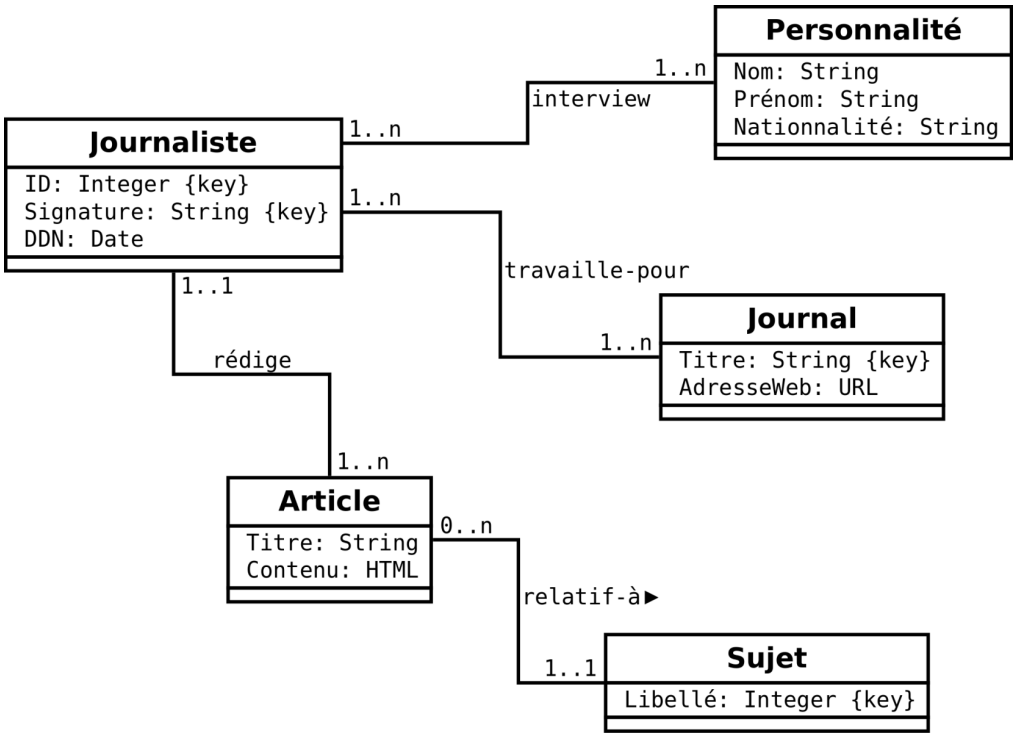
Le schéma suivant représente les rencontres lors d'un tournoi de tennis. Quelles sont les assertions vraies selon ce schéma ?



- ☐ On peut jouer des matchs de double.
- ☐ Un joueur peut gagner un match sans y avoir participé.
- ☐ Il peut y avoir deux matchs sur le même terrain à la même heure.
- ☐ Connaissant un joueur, on peut savoir sur quels terrains il a joué.

Journal

Voici le schéma conceptuel du système d'information (très simplifié) d'un quotidien. Quelles sont les assertions vraies selon ce schéma ?

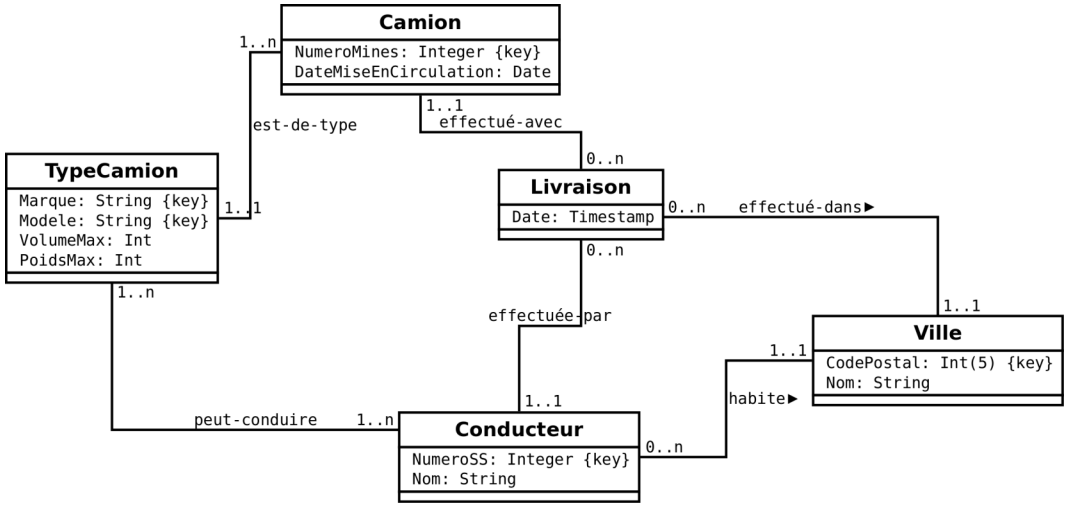


- ☐ Un article peut être rédigé par plusieurs journalistes.
- ☐ Un article peut être publié plusieurs fois dans le même journal.
- ☐ Un article peut être publié dans un journal par un journaliste qui ne travaille pas pour ce journal.
- ☐ Il peut y avoir plusieurs articles sur le même sujet.
- ☐ Un journaliste peut interviewer une personnalité sans faire d'article à ce propos.

Logistique

Une société de transport routier veut installer un système d'information pour rendre plus efficace sa logistique. Embauché au service informatique de cette compagnie, vous êtes donc chargé de reprendre le travail déjà effectué (c'est à dire le schéma suivant).

Quelles sont les assertions vraies selon ce schéma ?



- ☐ Un conducteur peut conduire plusieurs camions.
- ☐ Un conducteur peut conduire un camion sans y être autorisé.
- ☐ Il peut y avoir plusieurs conducteurs pour le même camion.
- ☐ Un conducteur peut livrer sa propre ville

2. Exercice : Cours et intervenants

[20 min]

On souhaite réaliser une base de données pour gérer les cours dispensés dans une école d'ingénieur, ainsi que les personnes qui interviennent dans ces cours.

Chaque cours est identifié par une année et un numéro. Chaque cours a donc un numéro unique localement à chaque année. Un cours possède un titre et un type ('C' pour Cours, 'TD' ou 'TP'). Un cours possède également une date de début, et une date de fin, qui est toujours de 5 jours après la date de début.

Chaque intervenant est identifié par son nom (deux intervenants ne peuvent pas avoir le même nom). Il a un prénom, un bureau, un numéro de téléphone et des spécialités. Un bureau est défini par un centre ('R' pour Royallieu, 'BF' pour Benjamin Franklin et 'PG' pour Pierre Guillaumat), un bâtiment (une lettre de A à Z), et un numéro (inférieur à 1000). Les spécialités sont des couples de chaînes de caractères désignant un domaine (par exemple 'BD') et une spécialité (par exemple 'SGBDRO').

Chaque cours est donné par un unique intervenant.

Voici un exemple : Le cours 'Machines universelles', n°21 de l'année 2014 est donné par Alan Turing entre le 05/01/2014 et le 10/01/2014. C'est un cours de type 'C'. Alan Turing a le bureau 666 au bâtiment X de PG. Il a les numéros de téléphone 0666666666 et 0766666666. Il possède les spécialités suivantes :

- Domaine : Mathématique ; Spécialité : Cryptographie
- Domaine : Informatique ; Spécialité : Algorithmie
- Domaine : Informatique ; Spécialité : Intelligence Artificielle

Question

Réaliser le modèle UML de la base de données. Préciser les clés et les types des attributs.

3. Exercice : Gestion méthodique du personnel

[30 minutes]

Le service de gestion du personnel d'une entreprise désire s'équiper d'un outil lui permettant de gérer informatiquement ses employés, leurs salaires et leurs congés. Les spécifications suivantes ont pu être mises en exergue par une analyse des besoins réalisée auprès des utilisateurs du service du personnel.

Généralités :

- tout employé est identifié par un nom, un prénom et une date de naissance ;
- tout employé remplit une fonction et appartient à un service ;
- pour chaque employé on gère la date d'embauche et la quotité (c'est à dire le pourcentage de temps travaillé par rapport au temps plein, en cas de travail à temps partiel).

Gestion des salaires :

- pour chaque employé on gère l'historique de ses salaires dans l'entreprise, chaque salaire étant affecté à une période de temps ;
- un salaire est composé d'un salaire brut, de charges patronales et de charges salariales ;
- on cherchera à partir des ces données à obtenir également le salaire chargé (brut + charges patronales), et le salaire net (brut - charges salariales), et ce en particulier pour le salaire en cours (celui que touche actuellement le salarié).

Gestion des congés :

- pour chaque employé, on mémorise chaque congé pris (posant qu'un congé concerne toujours une ou plusieurs journées entières) ;
- chaque employé a le droit aux jours de congés suivants :
    - 25 jours (pour une quotité de 1) et 25 x quotité sinon,
    - chaque fonction ouvre les droits à un certain nombre de jours de RTT,
    - chaque service ouvre les droits à un certain nombre de jours de RTT,
    - chaque tranche de 5 ans passés dans l'entreprise donne droit à 1 jour supplémentaire,
    - les employés de plus de 40 ans ont un jour supplémentaire, et ceux de plus de 50 ans deux.
- pour chaque employé on cherchera à connaître le nombre total de jours de congés autorisés, le nombre de jours pris et le nombre de jours restants sur l'année en cours.

Question

Réaliser le diagramme de classes permettant de modéliser ce problème.

Indice :

On fera apparaître les méthodes pertinentes étant donné le problème posé.

Méthodes (cf. p.10)

Contenus annexes

1. Transformation des méthodes par des vues

Méthode

Lorsqu'une méthode est spécifiée sur un diagramme UML pour une classe C, si cette méthode est une fonction relationnelle (elle renvoie une unique valeur et elle peut être résolue par une requête SQL), alors on crée une vue qui reprend les attributs de la classe C et ajoute des colonnes calculées pour les méthodes.

Attributs dérivés *Remarque*

Les attributs dérivés étant apparentés à des méthodes, ils peuvent également être gérés par des vues.

2. Explicitation des associations (sens de lecture et rôle)

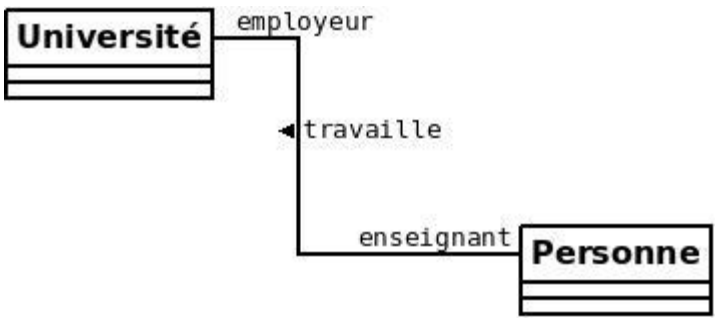
Sens de lecture *Syntaxe*

Il est possible d'ajouter le sens de lecture du verbe caractérisant l'association sur un diagramme de classe UML, afin d'en faciliter la lecture. On ajoute pour cela un signe < ou > (ou un triangle noir) à côté du nom de l'association

Rôle *Syntaxe*

Il est possible de préciser le rôle joué par une ou plusieurs des classes composant une association afin d'en faciliter la compréhension. On ajoute pour cela ce rôle à côté de la classe concernée (parfois dans un petit encadré collé au trait de l'association.

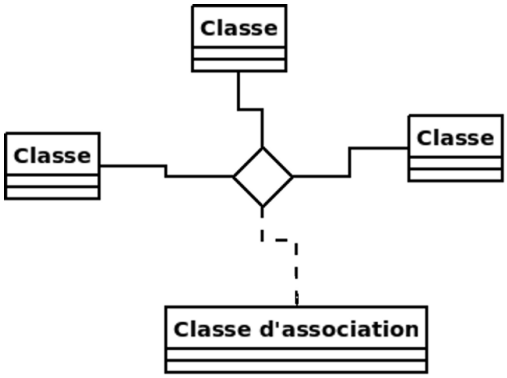
Exemple



Rôle et sens de lecture sur une association

3. Associations ternaires

Syntaxe



Notation d'une association ternaire

Ne pas abuser des associations ternaires *Conseil*

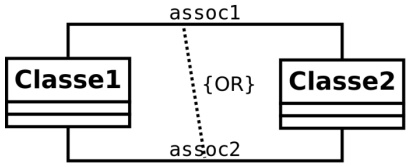
Il est toujours possible de réécrire une association ternaire avec trois associations binaires, en transformant l'association en classe.

Pas de degré supérieur à 3 *Conseil*

En pratique on n'utilise jamais en UML d'association de degré supérieur à 3.

4. Contraintes standardisées sur les associations

Exemple de notation *Exemple*

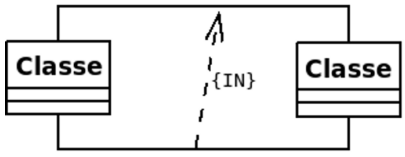


Contrainte OR entre deux associations

Classe1	Classe2	IN(C1→C2)	IN(C2→C1)	AND ou S	X	OR ou T	XOR ou XT
0	0	0	0	0	1	0	0
0	1	1	0	0	1	1	1
1	0	0	1	0	1	1	1
1	1	1	1	1	0	1	0

Inclusion {I}, également notée {Subset} ou {IN} *Définition*

Si l'association inclue est instanciée, l'autre doit l'être aussi, la contrainte d'inclusion a un sens, représenté par une flèche.



Contrainte orientée IN

Simultanéité {S}, également notée {=} ou {AND} *Définition*

Si une association est instanciée, l'autre doit l'être aussi.

La simultanéité est équivalente à une double inclusion.

Exclusion {X} *Définition*

Les deux associations ne peuvent être instanciées en même temps.

Totalité {T}, également notée {OR} *Définition*

Au moins une des deux associations doit être instanciée.

Partition {XT}, également notée {+} ou {XOR} ou {P} *Définition*

Exactement une des deux associations doit être instanciée.

Glossaire

Clé artificielle (surrogate key)

En relationnel une clé artificielle est un attribut artificiel (qui ne représente aucune donnée) ajouté à la relation pour servir de clé primaire. On mobilise cette technique lorsque l'on n'a pas pu ou voulu choisir une clé naturelle pour clé primaire.

Abréviations

BD : Base de Données
E-A : Entité-Association
OMG : Object Management Group
SGBD : Système de Gestion de Bases de Données
UML : Unified Modeling Language

Bibliographie

Arribe Thibaut. 2014. *Conception des chaînes éditoriales : documentariser l'activité et structurer le graphe documentaire pour améliorer la maîtrise de la rééditorialisation*. Université de Technologie de Compiègne, Mémoire de Doctorat. http://ics.utc.fr/~tha.

Rothenberg Jeff, Widman Lawrence E, Loparo Kenneth A, Nielsen Norman R. 1989. *The nature of modeling*. Rand. vol.3027.

Index