

Introduction au passage UML-Relationnel : classes et associations

*stph.scenari-community.org/bdd
rel2.pdf*



Stéphane Crozat

Table des matières



I - Cours	3
1. Transformation des classes et attributs	3
1.1. Transformation des classes	3
1.2. Transformation des attributs	3
1.3. Transformation des attributs dérivés et méthodes	5
2. Transformation des associations	6
2.1. Transformation des associations 1:N	6
2.2. Transformation des associations N:M	6
2.3. Transformation des associations 1:1 (approche simplifiée)	7
2.4. Transformation des classes d'association	7
II - Exercices	9
1. Exercice : Lab I+	9
2. Exercice : Usine de production	10
III - Devoirs	12
1. Exercice : Tourisme	12
Contenus annexes	13
Solutions des exercices	19

Cours

I

Afin de pouvoir implémenter une base de données, il faut pouvoir traduire le modèle conceptuel en modèle logique. Cela signifie qu'il faut pouvoir convertir un modèle UML en modèle relationnel. Les modèles conceptuels sont suffisamment formels pour que ce passage soit systématisé dans la plupart des cas.

1. Transformation des classes et attributs

Objectifs

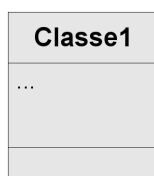
Savoir faire le passage d'un schéma conceptuel UML à un schéma relationnel pour les cas simples.

1.1. Transformation des classes

Méthode : Classe

Pour chaque classe non abstraite,

- on crée une relation dont le schéma est celui de la classe ;
- la clé primaire de cette relation est une des clés de la classe.



Classe

```
Classe1(...)
```

Remarque

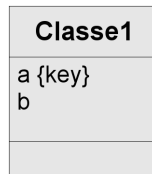
Les classes abstraites sont ignorées à ce stade, et n'étant pas instanciables, ne donnent généralement pas lieu à la création de relation.

1.2. Transformation des attributs

Méthode : Attributs simples

Pour chaque attribut élémentaire et monovalué d'une classe,

- on crée un attribut correspondant.



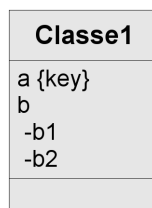
Attribut

```
Classe1 (#a, b)
```

Méthode : Attributs composites

Pour chaque attribut composite comprenant N sous-attributs d'une classe,

- on crée N attributs correspondants,
- dont les noms sont la concaténation du nom de l'attribut composite avec celui du sous-attribut.



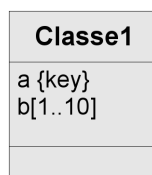
Attribut composé

```
Classe1 (#a, b_b1, b_b2)
```

Méthode : Attributs multivalués

Pour chaque attribut multivalué b d'une classe C,

- on crée une nouvelle relation RB,
- qui comprend un attribut monovalué correspondant à b,
- plus la clé de la relation représentant C ;
- la clé de RB est la concaténation des deux attributs.



Attribut multivalué

```
Classe1 (#a)
```

```
RB (#b, #a->Classe1)
```

Méthode : Attributs multivalués (méthode alternative)

Dans le cas où le nombre maximum de b est fini, et petit, on peut également adopter la transformation suivante : `Classe1 (#a, b1, b2, b3, b4, b5, b6, b7, b8, b9, b10)`.

Si le nombre d'attributs est infini (`b[1..*]`) c'est impossible, s'il est trop grand ce n'est pas souhaitable.

Méthode : Attributs composés multivalués

On combine les règles énoncées pour les attributs composés et pour les attributs multivalués.

Classe1
a {key}
b [0..N]
-b1
-b2

Attribut composé multivalué

Classe1 (#a)

RB (#b_b1, #b_b2, #a=>Classe1)

Rappel : Voir aussi

Transformation des compositions (cf. p.13)

1.3. Transformation des attributs dérivés et méthodes

Méthode : 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.

Classe1
a {key}
\b
m()

Attribut dérivé et méthodes

Classe1 (#a)

Complément : Attribut dérivé stockés

On peut décider (pour des raisons de performance essentiellement) de représenter l'attribut dérivé ou la méthode comme s'il s'agissait d'un attribut simple, mais il sera nécessaire dans ce cas d'ajouter des mécanismes de validation de contraintes dynamiques (avec des *triggers* par exemple) pour assurer que la valeur stockée évolue en même temps que les attributs sur lesquels le calcul dérivé porte.

Notons qu'introduire un attribut dérivé ou un résultat de méthode dans le modèle relationnel équivaut à introduire de la redondance, ce qui est en général déconseillé, et ce qui doit être dans tous les cas contrôlé.

2. Transformation des associations

Objectifs

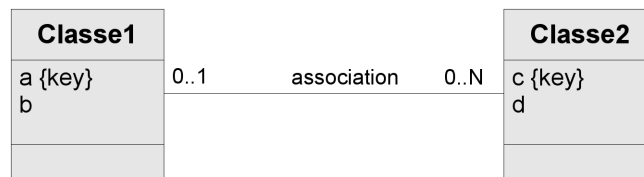
Savoir faire le passage d'un schéma conceptuel UML à un schéma relationnel pour les cas simples.

2.1. Transformation des associations 1:N

Méthode

Pour chaque association binaire de type 1:N :

- on ajoute à la relation côté N une clé étrangère vers la relation côté 1.



Association 1:N

Classe1 (#a ,b)

Classe2 (#c ,d ,a=>Classe1)

Complément

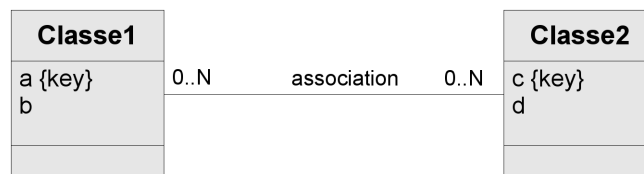
Contrainte de cardinalité minimale 1 dans les associations 1:N (cf. p.14)

2.2. Transformation des associations N:M

Méthode

Pour chaque association binaire de type M:N :

- on crée une nouvelle relation,
- composée de clés étrangères vers chaque relation associée,
- et dont la clé primaire est la concaténation de ces clés étrangères.




Association N:M

Classe1 (#a ,b)

Classe2 (#c ,d)

Assoc (#a=>Classe1 ,#c=>Classe2)

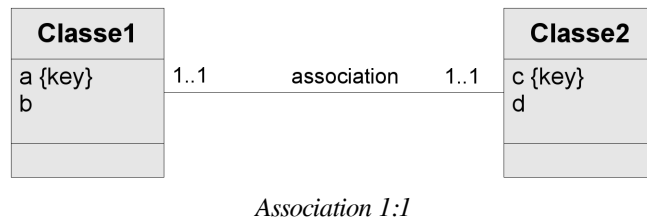
 **Complément**

Contrainte de cardinalité minimale 1 dans les associations N:M (cf. p.16)

2.3. Transformation des associations 1:1 (approche simplifiée)

 **Méthode**

La solution la plus simple et la plus générale pour transformer une association 1:1 consiste à traiter cette association 1:1 comme une association 1:N, puis à ajouter une contrainte UNIQUE sur la clé étrangère pour limiter la cardinalité maximale à 1.



Classe1(#a,b,c=>Classe2) avec c UNIQUE

Classe2(#c,d)


ou

Classe1(#a,b)

Classe2(#c,d,a=>Classe1) avec a UNIQUE

 **Remarque**

Il existe toujours deux solutions selon que l'on choisit une ou l'autre relation pour accueillir la clé étrangère. Selon la cardinalité minimale, un des deux choix peut être plus pertinent.

 **Complément**

Il est parfois possible de choisir de fusionner les deux classes au sein d'une seule relation plutôt que d'opter pour une clé étrangère.

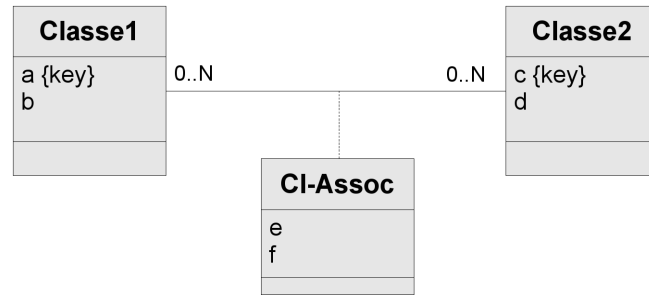
 **Complément**

Transformation des associations 1:1 (approche générale) (cf. p.17)

2.4. Transformation des classes d'association

 **Méthode : Classe d'association N:M**

Les attributs de la classe d'association sont ajoutés à la relation issue de l'association N:M.



Classe association (N:M)

```
Classe1 (#a , b)
```

```
Classe2 (#c , d)
```

```
Assoc (#a=>Classe1 , #c=>Classe2 , e , f)
```

Complément : Classe d'association 1:N

Les attributs de la classe d'association sont ajoutés à la relation issue de la classe côté N.

Complément : Classe d'association 1:1

Les attributs de la classe d'association sont ajoutés à la relation qui a été choisie pour recevoir la clé étrangère.
Si les deux classes ont été fusionnées en une seule relation, les attributs sont ajoutés à celle-ci.

Exercices


 II

1. Exercice : Lab I+

Description du problème

[30 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

[solution n°1 p.19]

Réaliser le modèle conceptuel de données en UML du problème.

Question 2

[solution n°2 p.19]

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.

Question 3

[solution n°3 p.19]

En mobilisant les règles adéquates, proposer un modèle logique de données correspondant en relationnel. Le repérage des domaines et des clés est obligatoire.

Question 4

[solution n°4 p.19]

Dessiner des tableaux remplis avec les données fournies en exemple, afin de montrer que le modèle fonctionne selon le besoin exprimé initialement. On pourra mettre le premier mot seulement des descriptions pour gagner du temps.

2. Exercice : Usine de production

[30 minutes]

Une usine cherche à modéliser sa production de véhicules et de moteurs :

- Les véhicules sont identifiés par un numéro d'immatriculation alphanumérique et caractérisés par une couleur, dont la dénomination est une chaîne de caractères. Chaque véhicule peut comporter un unique moteur et/ou un nombre quelconque de pneus.
- Chaque moteur est monté sur un et un seul véhicule et est identifié par un numéro de série. Un moteur est caractérisé par une puissance, en chevaux.
- Tout pneu est monté sur un unique véhicule et est identifié par un numéro de série. Sa position est définie localement sur ce véhicule et par rapport à l'essieu : *Dn* pour les pneus situés sur la droite de l'essieu et *Gn* pour les pneus situés à gauche ; *n* représentant le numéro de l'essieu (1 pour celui situé devant, 2 pour la deuxième rangée, etc.). Un pneu est caractérisé par un diamètre et une largeur en pouces.
- Les moteurs, les pneus et les véhicules sont fabriqués sous une marque. Les mêmes marques peuvent fabriquer indifféremment des moteurs, des pneus et/ou des véhicules, et un véhicule d'une certaine marque peut comporter un moteur et/ou des pneus de marque différente.

Question 1

[solution n°5 p.20]

Réaliser le modèle UML de ce problème en faisant apparaître les domaines et les clés.

Question 2

[solution n°6 p.20]

Réaliser le passage au modèle relationnel, en faisant apparaître les clés primaires, candidates et étrangères.

Question 3

[solution n°7 p.21]

Dessiner les tableaux correspondant aux relations du modèle. Instancier au minimum deux véhicules et quatre marques.

Question 4

[solution n°8 p.21]

Donner quatre exemples d'enregistrements qui seront refusés - étant données les données déjà insérées - pour quatre raisons différentes :

1. contrainte de clé sur une clé primaire
2. contrainte de clé sur une clé candidate
3. contrainte d'intégrité référentielle
4. contrainte de non nullité

Devoirs

III

1. Exercice : Tourisme

[45 min]

Une commune veut mieux tenir à jour et représenter les offres disponibles en terme de tourisme, et cherche ainsi à en établir une base de données.

Elle a répertorié :

- la liste de ses sites touristiques : leur nom, et leur ancienneté. Chaque site propose différentes activités (visite, visite guidée, concerts, ateliers, etc.).
- la liste de ses hôtels : leur nom, leur adresse et leur nombre d'étoiles.
- la liste de ses transports en commun, permettant de se déplacer entre les hôtels et les sites touristiques (on ne considérera ici que les bus). Un arrêt de bus peut desservir plusieurs hôtels et sites touristiques, et il est identifié par son numéro de ligne et son horaire.
- la liste de ses restaurants : un restaurant se trouve dans un hôtel ou sur un site touristique. Les restaurants possèdent un nom, un numéro de téléphone, et un type de cuisine (traditionnel, *fast food*, pizzeria, etc.).

Question 1

Établir un diagramme de classe UML pour cette base de données.

Question 2

Proposer un modèle relationnel cohérent avec le diagramme de classe présenté.

Question 3

Instancier la base de données en dessinant des tableaux avec des valeurs représentatives de ce qui est permis par le modèle.

Question 4

Proposer des exemples caractéristiques de données qui seront refusées grâce aux contraintes posées par le modèle.

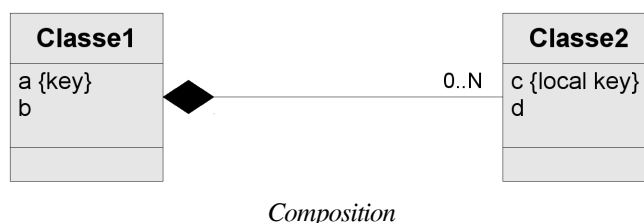
Contenus annexes

> Transformation des compositions

Méthode

Une composition

- est transformée comme une association 1:N,
- puis on ajoute à la clé de la classe partie (dite clé locale) la clé étrangère vers la classe composite pour construire une clé primaire composée.



Classe1(#a, b)

Classe2(#c, #a=>Classe1, d)

Remarque : Clé locale

Pour identifier une classe partie dans une composition, on utilise une clé locale concaténée à la clé étrangère vers la classe composite, afin d'exprimer la dépendance entre les deux classes.

Si une clé naturelle globale permet d'identifier de façon unique une partie indépendamment du tout, on préférera la conserver comme clé candidate plutôt que de la prendre pour clé primaire.

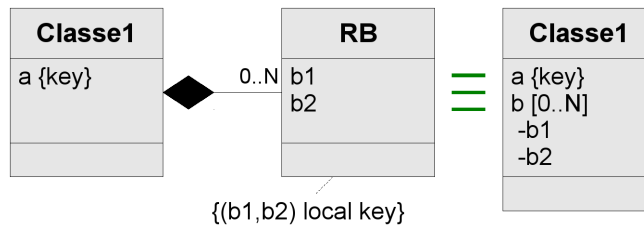
Si on la choisit comme clé primaire cela revient à avoir transformé la composition en agrégation, en redonnant une vie propre aux objets composants.

Complément : Composition et entités faibles en E-A

Une composition est transformée selon les mêmes principes qu'une entité faible en E-A.

Complément : Attributs multivalués et composés

La transformation d'un attribut composé multivalué donne un résultat équivalent à la transformation d'une composition.

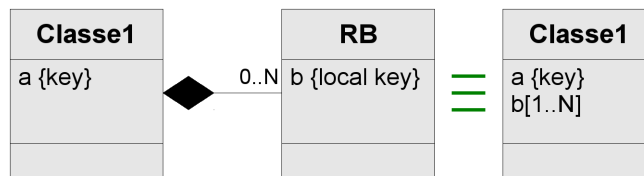


Composition et attribut composé multivalué

Classe1 (#a)

RB (#b_b1, #b_b2, #a=>Classe1)

La transformation d'une composition avec un seul attribut pour la classe composante donne un résultat équivalent à la transformation d'un attribut multivalué.



Composition et attribut multivalué

Classe1 (#a)

RB (#b, #a=>Classe1)

◆ Rappel : Voir aussi

Transformation des attributs (cf. p.3)

> **Contrainte de cardinalité minimale 1 dans les associations 1:N**

◆ Rappel

Transformation des associations 1:N (cf. p.6)

✂ Méthode



Association 1:N

- Si la cardinalité est exactement 1 (1..1) côté 1, alors on ajoutera une contrainte de non nullité sur la clé étrangère,
- si la cardinalité est au moins 1 (1..N) côté N, on ajoutera une contrainte d'existence de tuples référençant pour chaque tuple de la relation référencée.

R1 (#a, b)

R2 (#c, d, a=>R1)

Contraintes : a NOT NULL et PROJECTION(R1, a) = PROJECTION(R2, a)

On veut que tous les éléments de R1 soient référencés au moins une fois dans R2, donc il n'existe pas de tuple de R1 qui ne soit pas référencé par R2.a, donc $PROJECTION(R1, a) = PROJECTION(R2, a)$.

#a	b
1	Lorem
2	Ipsum

R1

#c	b	a=>R1
a	Sed	1
b	Ut	2
c	Perspiciatis	2

R2

 Complément : Cas général : $PROJECTION(R1, a) = PROJECTION(R2, a)$


Si la cardinalité côté 1 est 0..1 alors il peut exister la valeur NULL dans R2.a et donc la contrainte devient : $PROJECTION(Classe1, a) \subseteq PROJECTION(Classe2, a)$.

#a	b
1	Lorem
2	Ipsum

R1

#c	b	a=>R1
a	Sed	1
b	Ut	2
c	Perspiciatis	2
d	Unde	NULL

R2

 Rappel : La projection élimine les doublons

Projection (cf. p.15)

> Projection

Définition : Projection

La projection est une opération unaire (c'est à dire portant sur une seule relation). La projection de R1 sur une partie de ses attributs {A1, A2, ...} produit une relation R2 dont le schéma est restreint aux attributs mentionnés en opérande, comportant les mêmes tuples que R1, et dont les doublons sont éliminés.

Syntaxe

$R = \text{Projection} (R1, a1, a2, \dots)$

Exemple

Soit la relation suivante : *Personne* (nom, prénom, age)

nom	prénom	age
Dupont	Pierre	20
Durand	Jean	30

Personne

Soit l'opération : $R = \text{Projection} (\text{Personne}, \text{nom}, \text{age})$

On obtient alors la relation R composée des tuples suivants :

nom	age
Dupont	20
Durand	30

R

Remarque : La projection élimine les doublons

Après suppression d'une partie des attributs du schéma, la relation peut comporter des doublons. Étant donné que l'on ne pourrait plus identifier ces doublons les uns par rapport aux autres, la seule solution sensée est donc de considérer que deux doublons sont équivalents, et donc de n'en garder qu'un seul dans la relation résultante.

Complément : Syntaxes alternatives

$R = (R1, a1, a2, \dots)$

$R = a1, a2, \dots (R1)$

> Contrainte de cardinalité minimale 1 dans les associations N:M

◆ Rappel

Transformation des associations N:M (cf. p.6)

✂ Méthode

Si la cardinalité est au moins 1 (1..N) d'un côté et/ou de l'autre, alors des contraintes d'existence simultanée de tuples devront être ajoutées.



Association N:M

$R1(\#a, b)$

$R2(\#c, d)$

$A(\#a \Rightarrow R1, \#c \Rightarrow R2)$

Contraintes : $PROJECTION(R1, a) = PROJECTION(A, a)$ AND $PROJECTION(R2, c) = PROJECTION(A, c)$

🔍 Remarque

Si la cardinalité est $0..N : 1..N$ seule une des deux contraintes est exprimée.

◆ Rappel : La projection élimine les doublons

Projection (cf. p.)

> Transformation des associations 1:1 (approche générale)

Il existe deux solutions pour transformer une association 1:1 :

- Avec deux relations : on traite l'association 1:1 comme une association 1:N, puis l'on ajoute une contrainte UNIQUE sur la clé étrangère pour limiter la cardinalité maximale à 1 ;
- Avec une seule relation : on fusionne les deux classes en une seule relation.



Association 1:1

Méthode : Avec deux relations (clé étrangère)

- Une des deux relations est choisie pour porter la clé étrangère ;
- on ajoute les contraintes : UNIQUE ou KEY (clé candidate) sur la clé étrangère ; et si nécessaire une contrainte imposant l'instanciation simultanée des deux relations.

Classe1(#a,b,c=>Classe2) avec c UNIQUE ou KEY

Classe2(#c,d)

Contrainte (éventuellement) : PROJ(Classe1,c)=PROJ(Classe2,c)

ou

Classe1(#a,b)

Classe2(#c,d,a=>Classe1) avec a UNIQUE ou KEY

Contrainte (éventuellement) : PROJ(Classe1,a)=PROJ(Classe2,a)

Méthode : Avec une relation (fusion)

- On crée une seule relation contenant l'ensemble des attributs des deux classes ;
- on choisit une clé parmi les clés candidates.

Classe12(#a,b,c,d) avec c UNIQUE ou KEY

ou

Classe21(#c,d,a,b) avec a UNIQUE ou KEY

Remarque : Fusion des relations dans le cas de la traduction de l'association 1:1

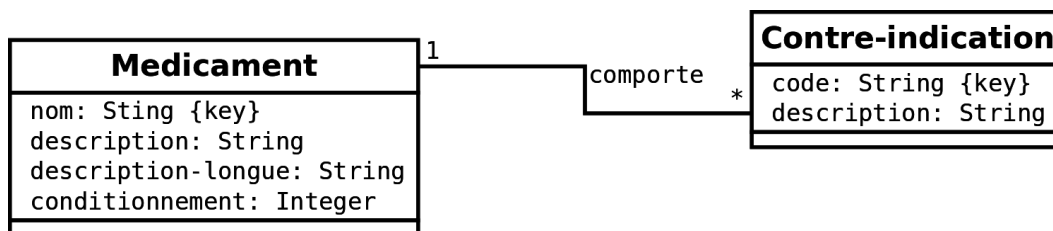
Ce choix entre les deux méthodes sera conduit par une appréciation du rapport entre :

- La complexité introduite par le fait d'avoir deux relations là où une suffit
- La pertinence de la séparation des deux relations d'un point de vue sémantique
- Les pertes de performance dues à l'éclatement des relations
- Les pertes de performance dues au fait d'avoir une grande relation
- Les questions de sécurité et de sûreté factorisées ou non au niveau des deux relations
- ...

Solutions des exercices

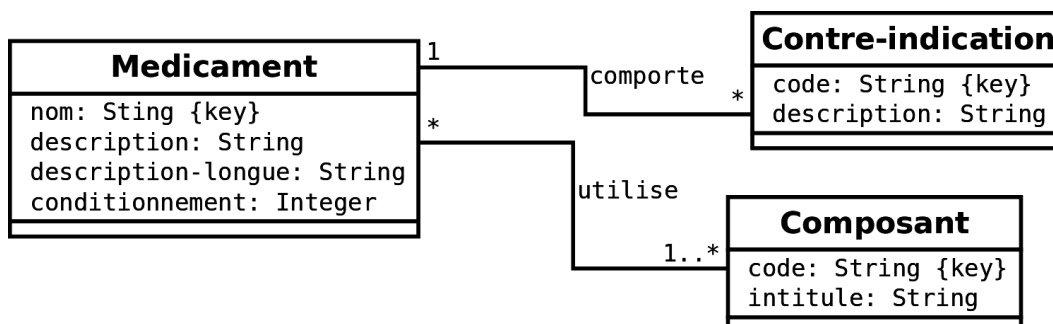
> Solution n°1

Exercice p. 9



> Solution n°2

Exercice p. 9



> Solution n°3

Exercice p. 10

```

1 Medicament (#nom:vvarchar, description:vvarchar, description_longue:vvarchar,
  conditionnement:number)
2 Contre_indication(#code:vvarchar, description:vvarchar, medicament=>Medicament)
3 Composant (#code:vvarchar, intitule:vvarchar)
4 Composition (#medicament=>Medicament, #composant=>Composant)
  
```

> Solution n°4

Exercice p. 10

Medicament

#nom	description	desc_longue	cond.
Chourix	Médicament contre la chute des choux	Vivamus...	11
Tropas	Médicament contre les dys...	Suspendisse...	42

Contre_indication

#code	description	medicament=>Medicament
CI1	Ne jamais prendre après minuit	Chourix
CI2	Ne jamais mettre en contact avec l'eau.	Chourix
CI3	Garder à l'abri de la lumière du soleil	Tropas

Composant

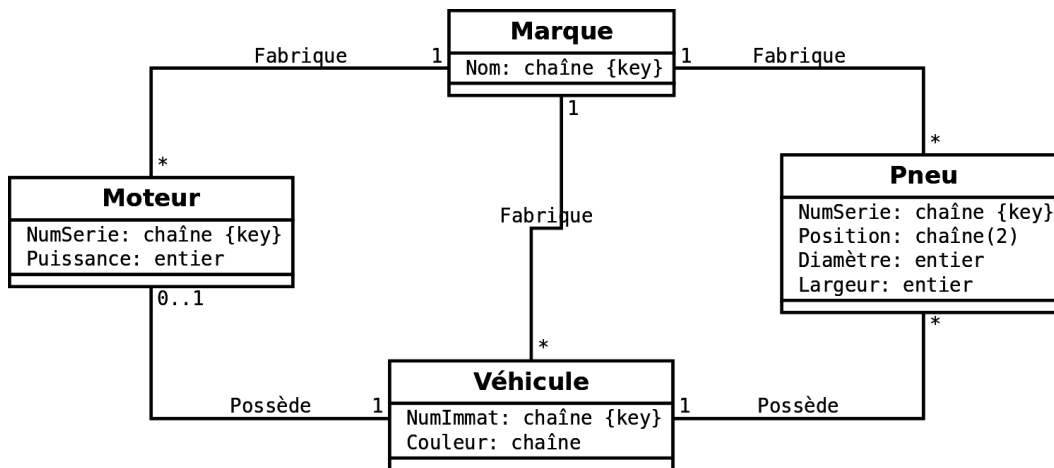
#code	intitule
HG79	Vif-argent allégé
HG81	Vif-argent alourdi
SN50	Pur étain

Composition

#medicament=>Medicament	#composant=>Composan
Chourix	HG79
Chourix	SN50
Tropas	HG79

> **Solution n°5**

Exercice p. 10

> **Solution n°6**

Exercice p. 10

```

1 Marque (#Nom:string)
2 Véhicule (#NumImmat:string, Couleur:string, Marque=>Marque) avec Marque NOT NULL
3 Moteur (#NumSerie:string, NumImmat=>Véhicule, Puissance:integer, Marque=>Marque)
  avec NumImmat KEY et Marque NOT NULL
4 Pneu (#NumSerie:string, Position:string(2), Diamètre:integer, Largeur:integer,
  Marque=>Marque, NumImmat=>Véhicule) avec Marque NOT NULL et NumImmat NOT NULL
  
```

⚠ **Attention**

Moteur.NumImmat est une *clé*, car l'association Moteur-Véhicule est de cardinalité 1:1.

 **Complément**

Si l'on fait l'hypothèse qu'il peut y avoir plusieurs pneus du côté d'un essieu (comme sur un camion), alors Position n'est pas une clé locale, il peut exister plusieurs fois un pneu D1 sur un véhicule par exemple.

En revanche si l'on fait l'hypothèse inverse (comme sur une voiture), alors Position est une clé locale, ce que l'on exprime par : (NumImmat, Position) KEY

> **Solution n°7**

Exercice p. 10

Marque					
#nom					
Peugeot					
Tesla					
Airbus					
Picardair					

Véhicule		
#numimmat	couleur	marque=>Marque
AA 122 SD	Rouge	Peugeot
ILOVEUTC		Tesla

Moteur		
#numserie	numimmat (k)	marque=>Marque
1425412	AA 122 SD	Tesla

Pneu					
#numserie	position	diamètre	largeur	marque=>Marque	numimmat (k)
KJGHYUN	A1	16	205	Picardair	AA 122 SD
KJGHYUM				Picardair	AA 122 SD
KJGHYUO	B2			Tesla	ILOVEUTC

> **Solution n°8**

Exercice p. 11

1. Ajouter [Peugeot] dans Marque (il existe déjà une marque Peugeot)
2. Ajouter [1425413, AA 122 SD, Airbus] dans Moteur (la voiture AA 122 SD a déjà un moteur)
3. Ajouter [1425414, AK 123 X, Airbus] dans Moteur (la voiture AK 123 X n'existe pas)
4. Ajouter [1425415, NULL, Airbus] dans Moteur (les moteurs sont nécessairement montés sur une voiture)