Introduction au passage UML-Relationnel : classes et associations

Table des matières

Cours

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

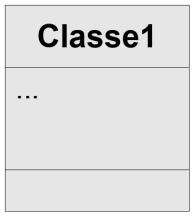
Savoir faire le passage d'un schéma conceptuel UML à un schéma relationnel pour les cas simples.

1.1. Transformation des classes

Classe *Méthode*

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

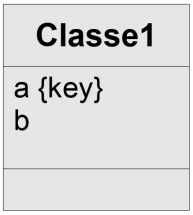
Les classes abstraites sont ignorées à ce stade, et n'étant pas instanciables, ne donnent généralement pas lieu à la création de relation.

1.2. Transformation des attributs

Méthode

Pour chaque attribut élémentaire et monovalué d'une classe,

- on crée un attribut correspondant dans la relation correspondant à la classe.



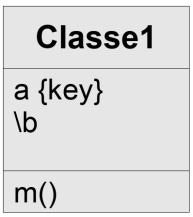
Attribut

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

1.3. Transformation des attributs dérivés et méthodes

Attributs dérivés et méthodes ***Méthode***

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.



Attribut dérivé et méthodes

```
Classe1(#a)
```

Attribut dérivé stockés ***Complément***

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

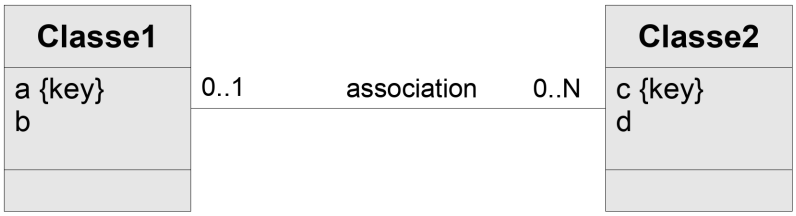
Savoir faire le passage d'un schéma conceptuel UML à un schéma relationnel pour les cas simples.

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.10)*

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)
```

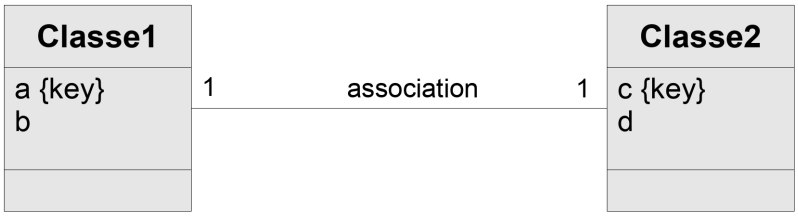
Complément

Contrainte de cardinalité minimale 1 dans les associations N:M *(cf. p.12)*

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.



Association 1:1

```
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.

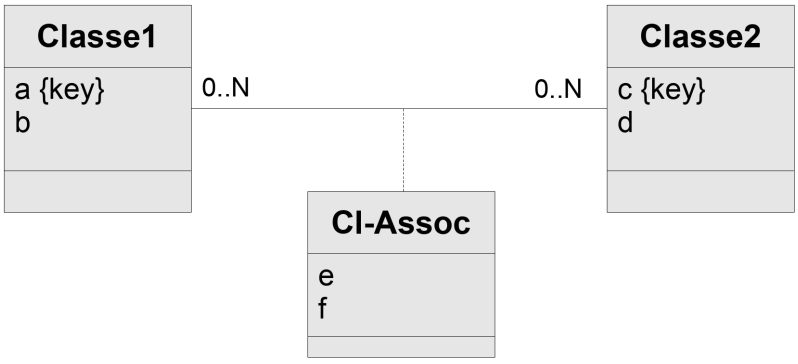
Complément

Transformation des associations 1:1 (approche générale) *(cf. p.13)*

2.4. Transformation des classes d'association

Classe d'association N:M **Méthode**

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



Classe assocation (N:M)

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

Classe d'association 1:N *Complément*

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

Classe d'association 1:1 *Complément*

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

1. Exercice : Lab I+

Description du problème

[45 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.

Question 3

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

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 situé 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

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

Question 2

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

Question 3

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

Question 4

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

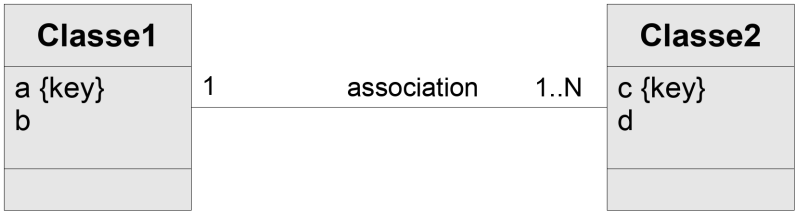
Contenus annexes

1. Contrainte de cardinalité minimale 1 dans les associations 1:N

Rappel

Transformation des associations 1:N *(cf. p.4)*

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 : `R2.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).`

R1

#a	b
1	Lorem
2	Ipsum

R2

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

Cas général : PROJECTION(R1,a) ⊆ PROJECTION(R2,a) *Complément*

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) ⊆ PROJECTION(Classe2,a).`

R1

#a	b
1	Lorem
2	Ipsum

R2

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

La projection élimine les doublons *Rappel*

Projection *(cf. p.11)*

2. Projection

Projection *Définition*

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 = Projection (R1, a1, a2, ...)
```

Exemple

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

Personne

nom	prénom	age
Dupont	Pierre	20
Durand	Jean	30

Soit l'opération : `R = Projection (Personne, nom, age)`

On obtient alors la relation R composée des tuples suivants :

R

nom	age
Dupont	20
Durand	30

La projection élimine les doublons

Remarque

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.

Syntaxes alternatives

Complément

`R = π (R1, a1, a2, ...)`

$R = \pi_{a1,a2,...}(R1)$

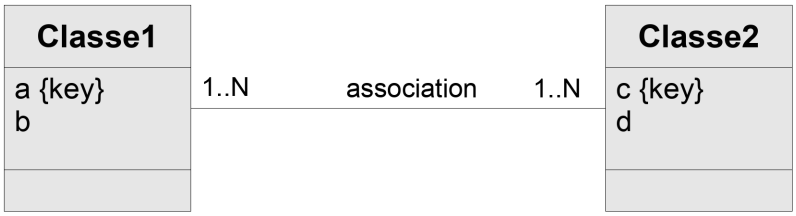
3. Contrainte de cardinalité minimale 1 dans les associations N:M

Rappel

Transformation des associations N:M *(cf. p.5)*

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=>R1,#c=>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.

La projection élimine les doublons ***Rappel***

Projection *(cf. p.11)*

4. 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

Avec deux relations (clé étrangère) ***Méthode***

- 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)`

ou

`Classe1(#a,b)`

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

Avec une relation (fusion) *Méthode*

- On créé 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
```

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

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 ou 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
- ...

Complément

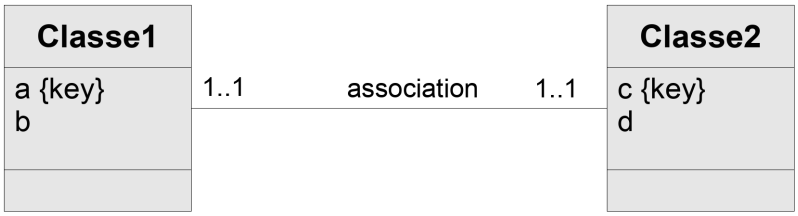
Dans le cas d'une association 1..1:1..1 il faudra ajouter une contrainte complémentaire.

Transformation des associations 1..1:1..1 (cf. p.14)

5. Transformation des associations 1..1:1..1

Association 1..1:1..1 *Méthode*

- Le plus souvent c'est méthode par **fusion** des relations qui est la plus adaptée à ce cas.
- Lorsqu'elle ne l'est pas, et que l'on choisit deux relations il faut ajouter une contrainte dynamique qui contrôlera que les deux relations sont bien toujours instanciées ensembles. Notons que la clé étrangère peut être choisie comme clé primaire.



Association 1:1

```
Classe12(#a,b,c,d) avec c KEY
```

ou

```
Classe21(#c,d,a,b) avec a KEY
```

ou

```
Classe1(#a,b,c=>Classe2) avec c KEY
Classe2(#c,d)
```

Contrainte : `PROJECTION(Classe1,c) = PROJECTION(Classe2,c)`

ou

```
Classe1(#a,b)
Classe2(#c,d,a=>Classe1) avec a KEY
```

Contrainte : `PROJECTION(Classe1,a) = PROJECTION(Classe2,a)`

Contrainte dynamique ***Complément***

La contrainte dynamique exprimée ici suit le même principe que pour les association 1:N.*Contrainte de cardinalité minimale 1 dans les associations 1:N* *(cf. p.10)*

Index