Université de La Rochelle
IUT de La Rochelle
Département Informatique
BD2 - UE 21 - M2106
Responsable : J. Malki

TD-08
Transformation
Modèle du domaine
vers un
Schéma Relationnel

Objectif

Passage d'un modèle de domaine (diagramme de classes UML) vers un schéma relationnel équivalent. On aborde aussi la rétro-conception, à savoir le passage d'un schéma relationnel vers un diagramme de classes. La méthode de transformation abordée dans ce travail n'est pas la seule. Il existe d'autres solutions de transformation, mais ces règles sont les plus simples et les plus opérationnelles.

1 Transformation des classes

Règle R1 :

Chaque classe du diagramme de classes UML devient une relation. Il faut choisir un attribut identifiant de la classe pouvant jouer le rôle de clé dans la relation. Si aucun attribut ne convient en tant qu'identifiant, il faut en ajouter un de telle sorte que la relation dispose d'une clé primaire (les AGL, type Modelio, proposent l'ajout de tels attributs).

1. Appliquer la règle R1 à l'extrait du diagramme de classes de la figure 1.

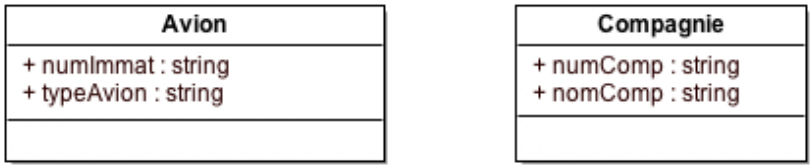

Figure 1: Extrait du domaine : 2 classes

2 Transformation des associations

Les règles de transformation que nous allons étudier dépendent des multiplicités maximales des associations. On distingue trois familles d'associations :

- un-à-plusieurs ;
- plusieurs-à-plusieurs ou classes associations, et n-aires ;
- un-à-un.

2.1 Transformation des associations : un-à-plusieurs (ou individu-population)

Règle R2 :

Il faut ajouter un attribut clé étrangère (et exprimer la dépendance de référence correspondante) dans la relation « individu » de l'association. L'attribut peut porter le nom de la clé primaire de la relation « population » de l'association ; il peut aussi ? c'est préférable ? porter un nom plus riche sémantiquement faisant référence au rôle joué par une instance de la classe « population » dans ses liens avec les instances de la classe « individu » à travers l'association.

On peut se rappeler cette règle de la manière suivante : la clé de la relation « population » est recopiée dans la relation « individu » et y exprime le rôle joué par la « population » (et « dit » de quelle population l'individu fait partie).

2. Appliquer la règle R2 à l'extrait du diagramme de classes de la figure 2.

Figure 2: Cas d'une association : un-à-plusieurs (ou individu-population)

2.2 Transformation des associations : plusieurs-à-plusieurs (ou agent-emploi-ressources) et n-aires

Règle R3 :

La classe association « emploi » devient une relation dont la clé primaire est composée par la concaténation des identifiants des classes « agent » et « ressources » connectées à l'association. On donne une dépendance de référence pour chaque attribut formant la clé primaire de la relation issue de la classe association si la classe connectée dont il provient devient une relation en vertu de la règle R1. Les attributs de la classe association doivent êtres ajoutés à la nouvelle relation.

3. Appliquer les règles R1 et R3 aux deux extraits des diagrammes de classes (figures 3 et 4). Le rôle « propriétaire » intuité au paragraphe précédent ne correspondait pas à la réalité. L'analyse est un processus itératif qui augmente la compréhension que l'on a d'un domaine au fur et à mesure qu'on le modélise.

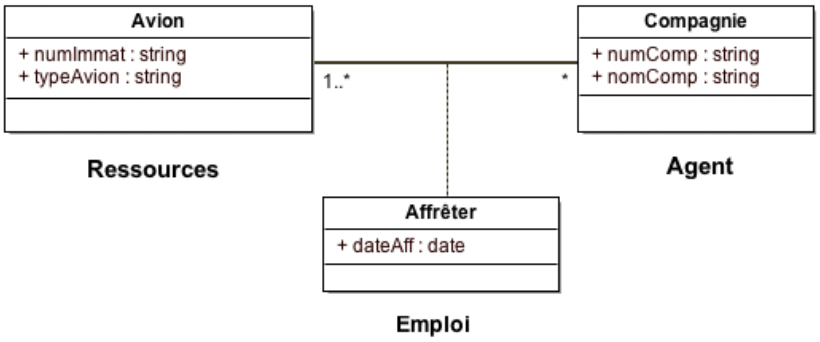

Figure 3: Cas d'une association plusieurs-à-plusieurs (ou agent-emploi-ressources)

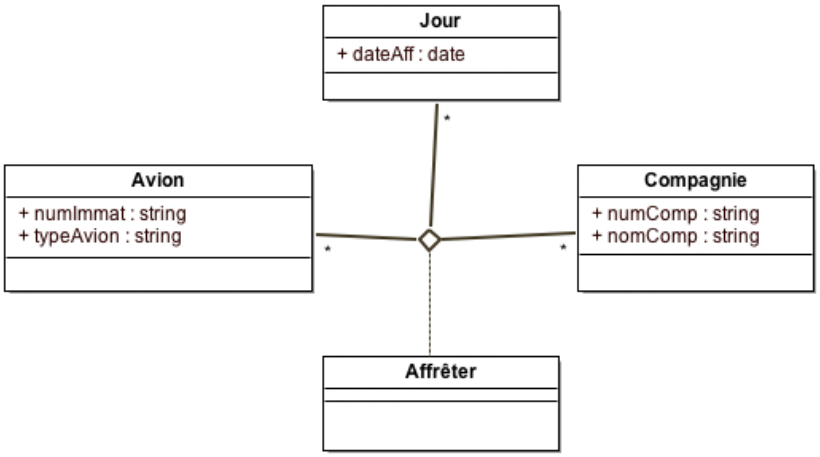

Figure 4: Cas d'une association n-aires

2.3 Transformation des associations : un-à-un

Règle R4 :

Il faut ajouter un attribut clé étrangère (et écrire la dépendance de référence correspondante) dans une relation dérivée de la classe ayant la multiplicité minimale égale à zéro. Il est, ici aussi préférable de faire porter à cet attribut un nom faisant référence au rôle joué par les instances de la classe où ses valeurs sont référencées (là où « il » est clé primaire). Si les deux multiplicités minimales sont à zéro, le choix est donné entre les deux relations dérivées de la règle R1. Si les deux multiplicités minimales sont à un, il est sans doute préférable de fusionner les deux classes en une seule. La règle R4 permet d'éviter les valeurs NULL dans la base de données.

4. Appliquer les règles R1 et R4 à l'extrait du diagramme de classes suivant (figure 5).

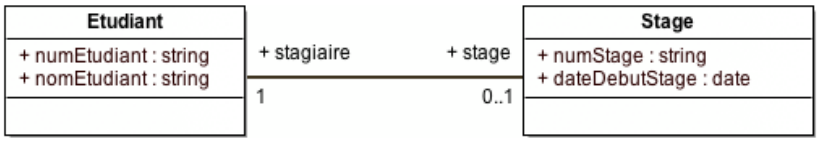

Figure 5: Cas d'une association : un-à-un

3 Transformation de l'héritage simple

Trois décompositions sont possibles pour traduire une association d'héritage en fonction des contraintes existantes :

- décomposition par distinction ;
- décomposition descendante (push-down). S'il existe une contrainte de totalité ou de partition sur l'association d'héritage ;
- décomposition ascendante (push-up).

3.1 Décomposition par distinction

Règle R5 :

Il faut transformer la sur-classe et chaque sous-classe en une relation. La clé primaire de la relation issue de la sur-classe migre dans la (les) relation(s) issue(s) de la (des) sous-classe(s) et devient à la fois clé primaire et clé étrangère pour laquelle on écrit la dépendance de référence.

5. Appliquer les règles R1 et R5 à l'extrait du diagramme de classes suivant (figure 6).

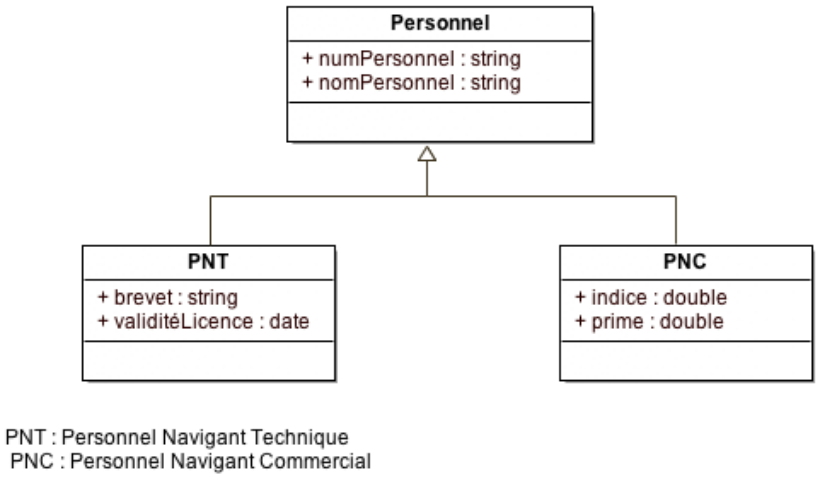

Figure 6: Cas d'un héritage simple

3.2 Décomposition descendante (push-down)

Règle R6 :

S'il existe une contrainte de totalité ou de partition sur l'association, il est possible de ne pas traduire la relation issue de la sur-classe. Il faut alors faire migrer tous ses attributs dans la (les) relation(s) issue(s) de la (des) sous-classe(s).

6. Appliquer les règles R1 et R6 à l'extrait du diagramme de classes suivant (figure 7).

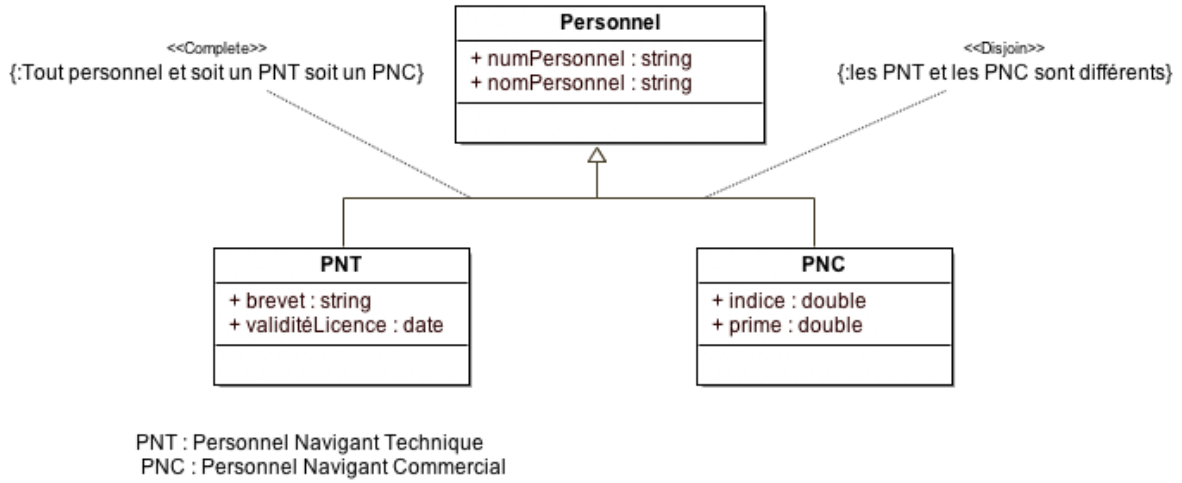

Figure 7: Cas d'un héritage contraint

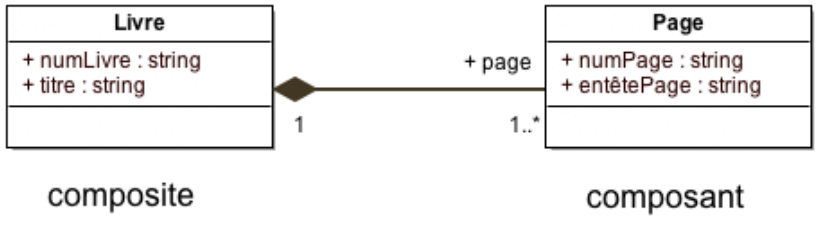

Figure 8: Cas d'une composition

3.3 Décomposition ascendante (push-up)

Règle R7 :

Les sous-classes ne donnent pas lieu à des relations. Il faut migrer dans la relation issue de la sur-classe tous les attributs issus des sous-classes. Il pourra donc, si l'association est disjointe, y avoir des attributs NULL, chaque nuplet ne verra ses attributs renseignés que pour ceux correspondant à sa sous-classe d'origine. On peut ajouter un attribut de traçabilité de la sous-classe d'origine. Le domaine de valeurs de cet attribut est l'ensemble des noms des sous-classes.

7. Appliquer les règles R1 et R7 à l'extrait du diagramme de classes (figure 6).

4 Transformation des agrégations et des compositions

Alors que l'agrégation UML se traduit au niveau logique comme une simple association, il n'en est pas de même pour la composition.

Règle R8 :

La clé primaire des relations déduites des classes composantes doit contenir l'identifiant de la classe composite (quelles que soient les multiplicités).

8. Appliquer les règles R1 et R8 à l'extrait du diagramme de classes suivant (figure 8).

5 Transformation d'un diagramme de classes

5.1 Cas : Gestion DT

1. Produire le schéma relationnel en appliquant les règles R1 à R8 à l'extrait du diagramme de classes suivant (figure 5.1).

2. Dossier : `GestionDT`

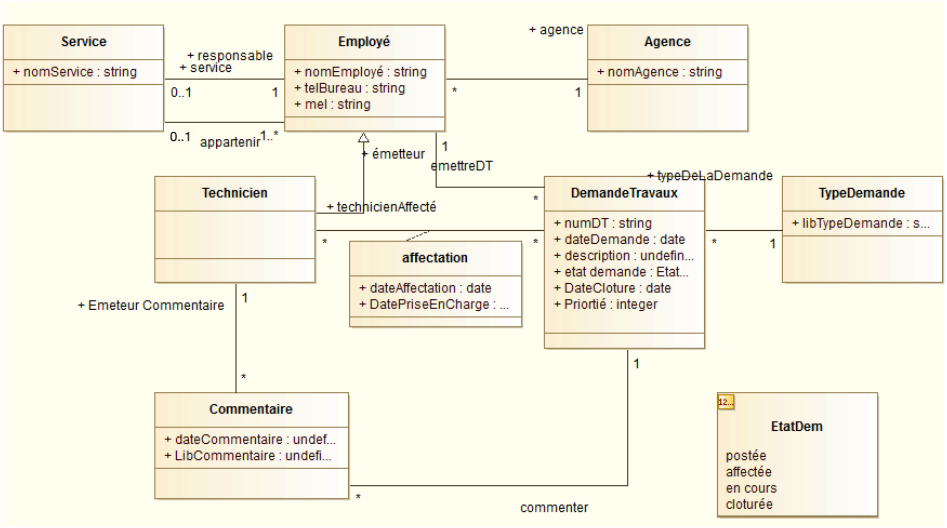

Figure 9: Modèle du domaine : Gestion DT

3. Fichier `ddl_gestionDT.sql` : fichier SQL contenant les ordres SQL de création des tables ;

4. Fichier `ddl_gestionDT_DR.sql` : fichier SQL contenant les ordres SQL de création des dépendances de références ;

5. Fichier `modele_insertion_gestionDT_DR.sql` : modèle d'insertion pour la base de données ;

6. Fichier `vide_contenu_base_gestionDT.sql` : suppression des nuplets des tables de la base de données.

7. Fichier `affiche_contenu_base_gestionDT.sql` : affiche le contenu des tables de la base de données.

8. Fichier `presentation_gestionDT.sql` : ordres SQLPLUS `COLUMN` pour la présentation du contenu des colonnes de la base de données de type `VARCHAR`.

5.2 Cas : Gestion Compte Bancaire

1. Produire le schéma relationnel en appliquant les règles R1 à R8 à l'extrait du diagramme de classes suivant (figure 5.1).

2. Dossier : `GestionBC`

3. Fichier `ddl_gestionBC.sql` : fichier SQL contenant les ordres SQL de création des tables ;

4. Fichier `ddl_gestionBC_DR.sql` : fichier SQL contenant les ordres SQL de création des dépendances de références ;

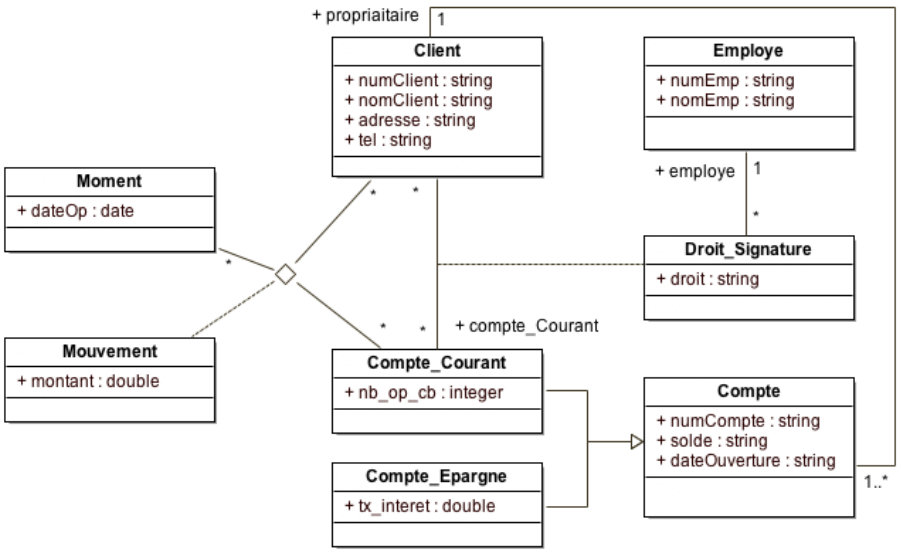

Figure 10: Modèle du domaine : Gestion Compte Bancaire

5. Fichier `modele_insertion_gestionBC_DR.sql` : modèle d'insertion pour la base de données ;
6. Fichier `vide_contenu_base_gestionBC.sql` : suppression des nuplets des tables de la base de données.
7. Fichier `affiche_contenu_base_gestionBC.sql` : affiche le contenu des tables de la base de données.
8. Fichier `presentation_gestionBC.sql` : ordres SQLPLUS `COLUMN` pour la présentation du contenu des colonnes de la base de données de type `VARCHAR`.

6 Devoir TP08

Déposer :

1. une archive du cas « Gestion DT » : `GestionDT.zip`
2. une archive du cas « Gestion BC » : `GestionBC.zip`