Ce document constitue l'annexe G de l'ouvrage "Bases de données", J-L Hainaut, Dunod, 2012

***Date de dernière modification** : 1/12/2011*

Annexe **G**

Conception logique d'une base de données relationnelle objet

La conception logique *relationnelle objet* tient compte des nouvelles structures qu'il est possible d'inclure dans les schémas logiques : TDU, hiérarchies de TDU et de tables, types array et row. Dans ce chapitre, on analyse la traduction des constructions d'un schéma conceptuel en schéma relationnel objet. On propose ensuite un plan de transformation systématique qu'on illustre sur un cas concret. Quelques commentaires sur l'approche relationnelle objet sont proposés.

Dans la première édition de cet ouvrage, la conception logique *relationnelle objet* faisait l'objet du chapitre 19. Ce dernier se présente dans les éditions ultérieures sous la forme de la présente annexe. Les particularités relatives aux constructions spécifiques au modèle relationnel objet sont désormais traitées brièvement à la fin du chapitre consacré à la conception logique *relationnelle*.

G.1 INTRODUCTION

Nous suivrons une approche similaire à celle du chapitre 18. Après avoir rappelé les principe des schémas relationnels objet[1], sous la forme d'un modèle logique étendant le modèle logique relationnel, nous examinerons les différentes manières de convertir les différentes constructions du modèle Entité-association en structures relationnelles objet. Nous déduirons de cette analyse un plan de transformation général. Par analogie avec la conception logique relationnelle, nous représenterons

la conception logique comme illustré à la figure G.1, suggérant que le processus est similaire pour tous les modèles, et que seule change la définition du modèle[2].

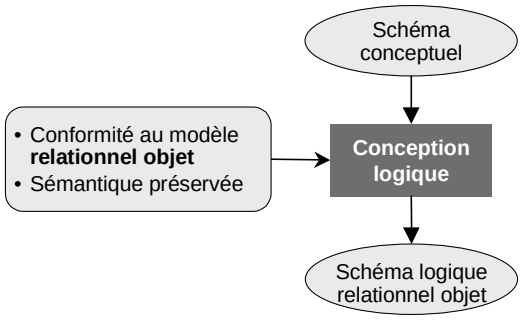

Figure G.1 - Le processus de conception logique selon le modèle cible *relationnel objet*

G.2 LE MODÈLE LOGIQUE RELATIONNEL OBJET

Notons avant tout qu'un schéma *relationnel* est aussi un schéma *relationnel objet*. Selon la description que nous en avons donnée à la section 9.6, le modèle de données relationnel objet introduit plusieurs constructions qui le rapprochent du modèle Entité-association. Par certains côtés, la conception logique relationnelle objet s'en trouvera facilitée, mais d'autre part, ces nouveaux concepts autorisent des variantes qui pourraient nous compliquer la tâche. Aux modèle relationnel de base, le modèle relationnel objet ajoute des constructions spécifiques, dont nous retiendrons les suivantes.

- *Le type de données complexe row.* Une colonne de ce type correspond à un attribut composé.
- *Le type de données complexe array.* Une colonne de ce type correspond à un attribut multivalué; cependant, la collection de valeurs d'une telle colonne pour une ligne ne forme pas un ensemble, mais bien un tableau. Comme déjà précisé (18.3.7/b), l'utilisation du constructeur array (tableau) pour représenter des *ensembles* de valeurs n'est que partiellement satisfaisante : un tableau est

1. Nous avons privilégié le terme *relationnel objet* au détriment de celui d'*objet relationnel* pour deux raisons. La première est historique : le modèle relationnel, bien après qu'il soit devenu la référence dans les SGBD, a été étendu par des fonctionnalités objet. Sur bien des points, son origine relationnelle reste prépondérante. Il est donc légitime de respecter cette chronologie par l'expression *relationnel objet*. La seconde est linguistique : la traduction correcte de l'expression anglaise standard *object relational* est bien *relationnel objet*, de même que *indexed sequential* se traduit par *séquentiel indexé* et non *indexé sequentiel* comme on le lit trop souvent.
2. C'est d'ailleurs ainsi que fonctionne l'atelier DB-MAIN : le moteur de transformation est unique, mais il est piloté par un *script de transformation*, qui est la traduction effective du plan de transformation spécifique au modèle logique cible. Des publications apparaissent actuellement sur la dérivation de plans de transformation à partir de la description d'un modèle cible.

d'une taille limitée, n'assure pas l'unicité des valeurs qu'il contient, impose un ordre des valeurs[3] et admet les valeurs manquantes. La norme SQL 2003 ajoute le constructeur multiset (multi-ensemble) qui lève les contraintes sur la taille, l'ordre et les valeurs manquantes. L'unicité n'est toujours pas garantie. L'introduction future du constructeur set[4] (ensemble) est en discussion.

- *Le type de données défini par l'utilisateur (TDU)*. Un TDU est utilisé comme type de colonne ou comme type de table. Lorsqu'il est exclusivement utilisé pour définir des colonnes, il correspond au TDU Entité-association (15.7.4). En revanche, le *type de table* n'a pas d'équivalent dans le modèle Entité-association, où il correspondrait à un *type de types d'entités*. Dans ce chapitre, nous ignorerons donc les TDU qui servent à définir des tables, mais nous en appliquerons toutes les caractéristiques aux tables qu'ils servent à définir.
- *La hiérarchie de TDU de tables*. Définit une hiérarchie *is-a* qui sera appliquée aux tables typées.
- *Les tables typées*. Nous retenons qu'une sous-table hérite des colonnes de sa surtable selon la structure des TDU de tables utilisés. On peut ainsi traduire directement les relations *is-a* de types d'entités. Cependant, cette traduction est limitée aux *hiérarchies simples* (un seul surtype par type d'entités) et à la contrainte de *disjonction*. On ne peut donc définir que les contraintes D et P.
- *Les identifiants d'objet*. L'identifiant d'objet sera un identifiant technique (system generated) sans signification, ou un identifiant significatif constitué d'une colonne identifiante de la table. Un identifiant primaire peut en outre être déclaré de manière traditionnelle. Il est possible de choisir l'identifiant primaire comme identifiant d'objet.
- *La colonne de référence*. Cette colonne, de même nature que l'identifiant d'objet, sert de clé étrangère système. elle jouera le même rôle que la clé étrangère standard.

Comme nous l'avons fait pour le modèle relationnel, nous déduisons de ce rappel le *modèle relationnel objet* de la figure G.2. Nous y avons cependant ajouté la contrainte référentielle totale (equ) et les contraintes d'existence, pour les mêmes raisons que pour le modèle relationnel pur. De même, nous conservons les attributs multivalués de type *set*, sachant qu'il faudra leur accorder une attention particulière. Tout schéma Entité-association qui respecte ces contraintes peut être qualifié de *relationnel objet*.

À l'inverse, un schéma **n'est pas relationnel objet** dès qu'on y trouve des constructions du modèle Entité-association qui ne sont pas reprises dans le modèle relationnel objet. Ces constructions non valides sont reprises dans la figure G.3.

3. L'ordre des valeurs n'est pas en elle-même une caractéristique négative. Le problème est que l'utilisateur (ou le programmeur) sera tenté d'utiliser cet ordre pour représenter une propriété absente du schéma conceptuel. Exemple : *le 1er numéro de téléphone est celui du secrétariat*.
4. Déjà présent en Oracle sous la forme de nested table.

Modèle relationnel objet

1. le schéma comporte des types d'entités (appelés *tables*),
2. un type d'entités peut être un sous-type d'un autre type d'entités ; les sous-types d'un type d'entités sont soumis à une contrainte de disjonction,
3. tout type d'entités possède au moins un attribut (appelé *colonne*),
4. un attribut est mono-valué ou multivalué (tableau), il est atomique ou composé; il est obligatoire ou facultatif,
5. les seules contraintes sont celles qui sont induites par les identifiants (*identifiant primaire* ou *secondaire*), les attributs de référence (*clés étrangères ref*) et les attribut de référence totaux (*clés étrangères equ*) ainsi que les contraintes d'existence (*coex, excl, at-least-1, exact-1, if*),
6. le schéma ne contient pas d'autres constructions,
7. les noms des objets respectent la syntaxe SQL.

Figure G.2 - Définition du modèle relationnel objet

Constructions non relationnelles objet

1. les ensembles de sous-types non disjoints,
2. les types d'entités sans attributs,
3. les types d'associations,
4. les contraintes autres que celles des identifiants, des attributs de référence (éventuellement totaux) et d'existence,
5. les noms d'objets non conformes à la syntaxe SQL.

Figure G.3 - Les constructions non relationnelles objet

G.3 TYPES D'ENTITÉS ET IDENTIFIANTS

Réglons deux questions préliminaires relatives à la traduction des types d'entités et des identifiants primaires.

G.3.1 Types d'entités

Un type d'entités peut être traduit en une table comme nous l'avons fait pour le modèle relationnel pur. Il peut aussi se traduire par un type de table et une table de ce type, ce qui permet de profiter des caractéristiques orientées objet de SQL3. Afin de simplifier le processus de conception logique et de favoriser l'évolution ultérieure de la base de données, on adopte la seconde approche. On suggère d'associer au type d'entités de nom **E** le type de tables de nom **T_E** et la table de nom **E**. Ces noms devront encore être rendus conformes à la syntaxe SQL.

G.3.2 Identifiants primaires

Cette question risque d'opposer les communautés *bases de données* et *programmation orientée objet*. La première postule que toute information est basée sur des valeurs (*value-based modeling*) tandis que la seconde se base sur la prévalence de l'objet sur les valeurs, dont le rôle se limite à définir les états d'un objet (*object-based modeling*). Proposons un compromis[5] selon lequel toute table est munie d'un identifiant d'objet, celui-ci étant composé de l'identifiant primaire lorsque la chose est raisonnable et d'une colonne technique sinon. On procède comme suit.

- *Type d'entités **E** doté d'un identifiant primaire constitué d'un seul attribut **A*** : la table **E** (accompagnée de son type) reçoit un identifiant primaire constitué de la colonne **A**. Un identifiant d'objet est défini sur base de la colonne **A**.
- *Type d'entités **E** doté d'un identifiant primaire constitué de plus d'un composant* ou dont l'unique composant n'est pas stable : l'identifiant primaire de **E** est traduit en identifiant primaire de la table **E**[6]. On ajoute un identifiant d'objet technique (system generated) de nom **OID_E**.
- *Type d'entités **E** sans identifiant primaire* : la table **E** reçoit un identifiant d'objet technique (system generated) de nom **OID_E**.

G.4 REPRÉSENTATION DES ATTRIBUTS COMPLEXES

Un schéma relationnel étant aussi un schéma relationnel objet, les attributs complexes peuvent être traités par les techniques qui ont été développées pour le modèle relationnel. Il peut cependant être intéressant de profiter des nouvelles constructions du modèle SQL3.

Les *attributs composés*, éventuellement multi-niveaux, sont directement exprimables dans le modèle relationnel objet sous forme du type row.

La question des *attributs multivalués* est un peu plus complexe. Là où les caractéristiques des tableaux sont inacceptables, on appliquera les techniques utilisées pour le modèle relationnel pur (section 18.3), basées sur la transformation de l'attribut multivalué en type d'entités par représentation des instances ou des valeurs[7].

On admet donc les règles suivantes, applicables à l'attribut complexe A du type d'entités E et illustrée à la figure G.4.

1. Un attribut composé A est représenté par une colonne row.
2. Un attribut multivalué A est représenté par une colonne array pour autant que le nombre d'éléments soit limité, que la notion d'ordre et de valeur manquante soit jugé sans effet négatif et que l'unicité des valeurs soit géré par une contrainte additionnelle[8].

5. Le compromis est, avec la bière et la bande dessinée, une des spécialités belges remarquables.
6. Nous tolérerons une exception dans les tables d'association pures (section G.5.3).
7. Au moment de faire le choix d'une représentation, on n'oubliera pas que la représentation sous forme d'une table s'avérera, avec le temps, plus évolutive que la traduction sous forme d'une colonne complexe.

3. Si les contraintes du type array sont jugées inacceptables, on transformera l'attribut multivalué A en un type d'entités par représentation des instances ou des valeurs[9].

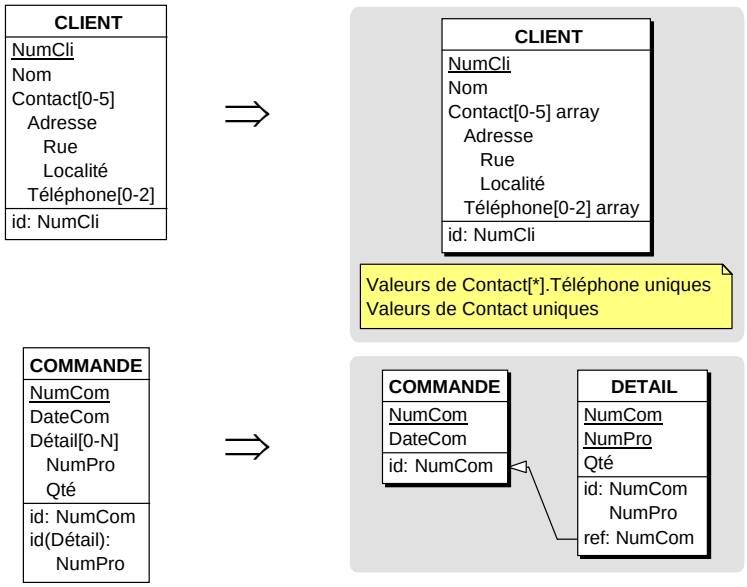

Figure G.4 - Représentation (partielle) d'attributs complexes

G.5 REPRÉSENTATION DES TYPES D'ASSOCIATIONS BINAIRES

Les techniques présentées dans les sections 13.4.1 à 13.4.4 sont bien sûr d'application. Un type d'associations fonctionnel se traduit par une clé étrangère tandis qu'un type d'associations N:N produit la définition d'une *table association* dont chaque ligne représentera une association, et de deux clés étrangères. Nous allons cependant approfondir la question pour deux raisons :

- d'une part, le modèle relationnel objet offre de nouvelles possibilités de représentation,
- d'autre part, les recommandations qu'on trouve dans la littérature suggèrent parfois des techniques qu'il est utile de mentionner et d'étudier.

Étudions d'un peu plus près ce que nous offre le modèle relationnel objet pour chacune des classes fonctionnelles.

8. Rappelons que le type nested table d'Oracle correspond à un ensemble, ce qui simplifie la traduction des attributs multivalués.
9. DB2 n'admet pas (encore) le type array. La technique proposée sera donc appliquée systématiquement.

G.5.1 Types d'associations un-à-plusieurs (1:N)

Examinons d'abord une des structures les plus classiques : un type d'associations 1:N dont chaque type d'entités possède un identifiant primaire (figure G.5). La traduction (a), traditionnelle, est celle qui vient immédiatement à l'esprit. On trouvera cependant assez fréquemment la traduction (b), qui utilise ne concept nouveau de *clé étrangère multivaluée* sous forme d'un tableau de valeurs de Matricule, dont la taille a été fixée arbitrairement à 100, et dont les valeurs doivent être uniques. On spécifie cette clé étrangère en indiquant que chaque valeur de Matricule (notée Matricule[*]) constitue une référence à la table EMPLOYE. Rappelons qu'il peut être intéressant de représenter un type d'associations 1:N par une table association (section 18.4). Pour des raisons de place, nous ne reprendrons pas cette technique dans la figure G.5.

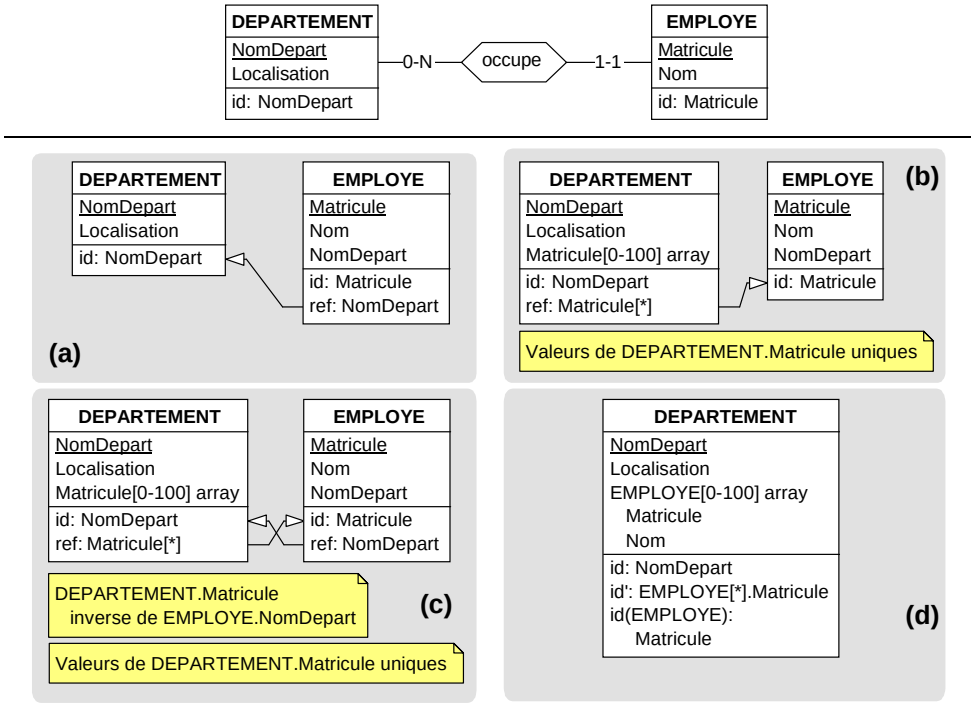

Figure G.5 - Expression d'un type d'associations 1:N

Le schéma (b) peut sembler étonnant à première vue, mais peut se comprendre comme suit. D'une part, l'usage d'une référence basée sur un identifiant d'objet offre des possibilités intéressantes, dont le mode de désignation navigationnel fléché "->" décrit à la section 9.6.6. D'autre part, l'approche orientée objet est par nature plus procédurale et navigationnelle que l'approche relationnelle, intrinsèquement déclarative (SQL permet de spécifier les propriétés des données à extraire en ignorant la manière d'y accéder). Le programmeur OO accède aux *employés* d'un *département* par une construction orientée, différente de celle de l'accès inverse (la

jointure SQL est au contraire symétrique). D'ailleurs, on observe fréquemment la combinaison des deux mécanismes (schéma (c)). Si on adopte cette dernière, il conviendra d'exprimer la contrainte de redondance[10] en indiquant que les deux clés étrangères sont *inverses* l'une de l'autre.

Le schéma (d) représente le type d'entités EMPLOYE sous la forme de la colonne complexe DEPARTEMENT.EMPLOYE. On remarque que l'identifiant d'EMPLOYE se traduit d'une manière particulièrement complexe (voir aussi figure 15.37).

Examinons ensuite une variante dans laquelle le type d'entités dépendant possède un *identifiant hybride* (figure G.6). La clé primaire d'EXEMPLAIRE étant multi-composants, on dote cette table d'un identifiant d'objet OID_Ex. Le schéma (a) utilise une clé étrangère traditionnelle et le schéma (b) inclut une clé étrangère multivaluée. La combinaison des deux n'est pas illustrée. Le schéma (c) intègre EXEMPLAIRE à OUVRAGE sous forme d'une colonne complexe. La taille du tableau a été limitée à 20.

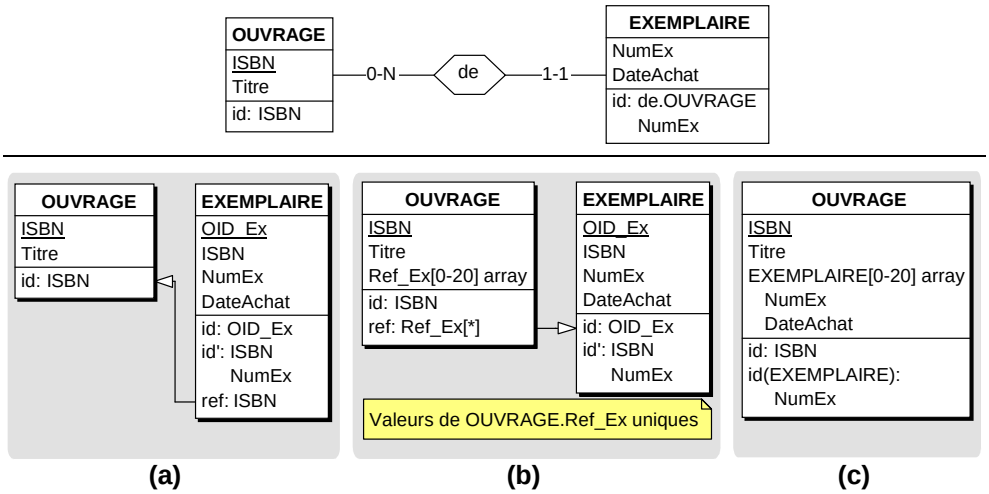

Figure G.6 - Expression d'un type d'associations 1:N en présence d'un identifiant hybride

Proposition. Sauf conditions exceptionnelle, on propose d'adopter la traduction relationnelle pure, sous la forme d'une clé étrangère (schéma a), selon les règles de la section 13.4.1.

G.5.2 Types d'associations un-à-un (1:1)

Un type d'association 1:1 se traite comme un cas particulier de la classe fonctionnelle 1:N. La figure G.7 reprend trois représentations standard. Le lecteur s'étonnera peut-être de la variante (c), qu'il jugera peu naturelle et qu'il serait tenté de remplacer par son inverse : une table SERVICE comportant une colonne row DIREC-

10. Cette contrainte existe dans certains SGBD OO purs sous la forme d'*inverse*.

TEUR. Cette solution n'est malheureusement pas possible, tout employé n'étant, dans la plupart des services, pas (encore) directeur. Il peut être intéressant de représenter un type d'associations 1:1 par une table association (section 18.4). Pour des raisons de place, nous ne reprendrons pas cette technique dans la figure G.7.

On note que la traduction (b) est fréquemment proposée comme l'expression naturelle relationnelle objet d'un type d'associations 1:1[11]. Il n'est pas rare dans ce cas que la contrainte d'inverse soit ignorée.

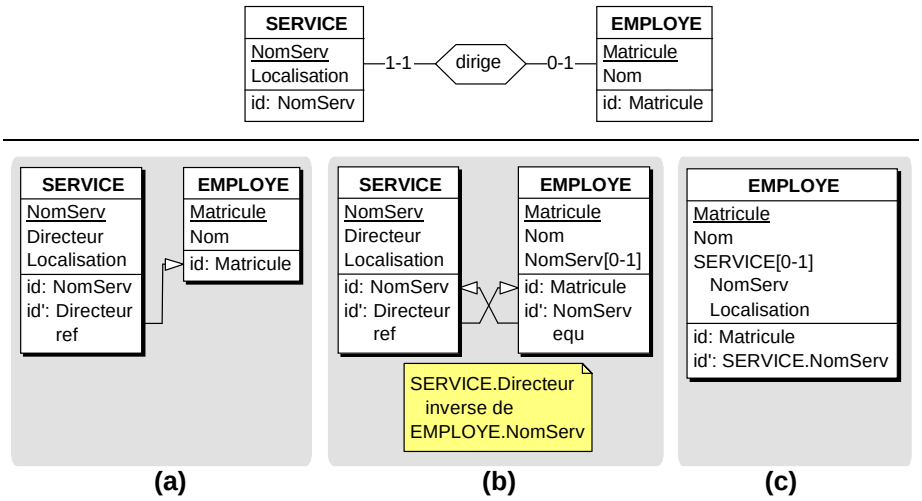

Figure G.7 - Expression d'un type d'associations 1:1

Proposition. Sauf conditions exceptionnelles, on propose d'adopter la traduction relationnelle pure (schéma a), sous la forme d'une clé étrangère identifiante, selon les règles de la section 13.4.2.

G.5.3 Types d'associations plusieurs-à-plusieurs (N:N)

L'expression relationnelle pure a été présentée à la figure 13.6 : une table association est constituée des deux clés étrangères qui référence les tables qui traduisent les types d'entités associés. Elle est reprise à la figure G.8 (a).

Les autres représentations sont basées sur le principe des clés étrangères multivaluées que nous avons étudiées. Les schémas (b) et (c) représentent respectivement les traductions par clés étrangères multivaluées asymétrique et symétrique.

Proposition. Sauf conditions exceptionnelles, on propose d'adopter la traduction relationnelle pure (schéma a), sous la forme d'une table d'association formée de clés étrangères, selon les règles de la section 13.4.3.

11. ... et même comme schéma relationnel pur, ce qui est absolument inutile, redondant et souvent mal géré.

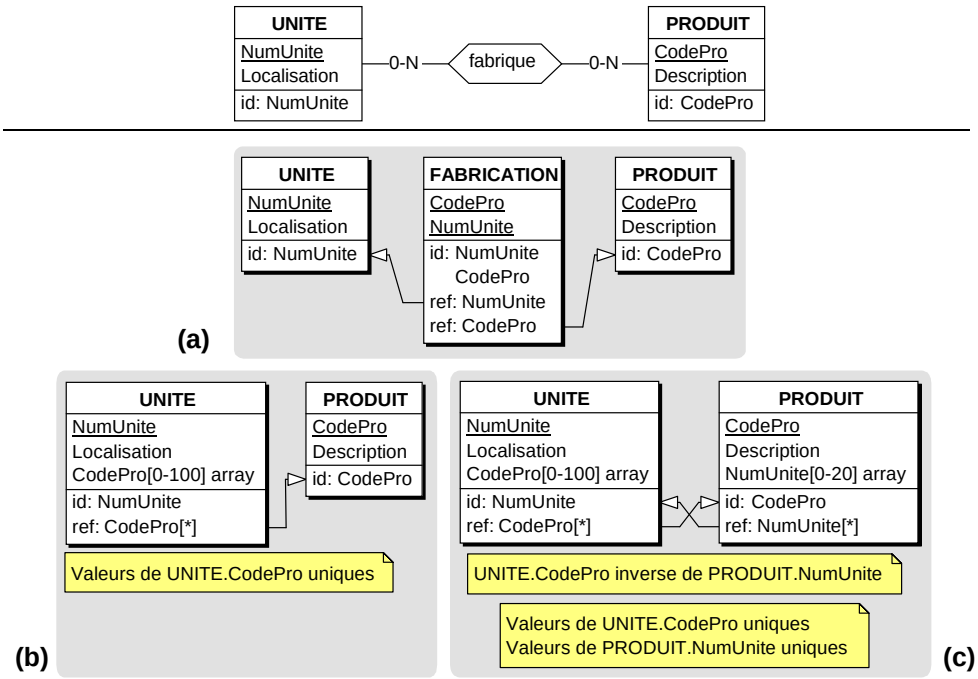

Figure G.8 - Expression d'un type d'associations N:N

G.6 REPRÉSENTATION DES TYPES D'ASSOCIATIONS COMPLEXES

Un type d'associations complexe (n-aire ou binaire doté d'attributs) se traite comme un type d'associations N:N. On part d'un schéma dans lequel le type d'associations est d'abord transformé en un *type d'entités association* accompagné de n types d'associations fonctionnels (schéma 18.15, gauche). On génère ensuite deux familles de traductions :

- *Expression symétrique* : les n types d'associations fonctionnels sont transformés en n clés étrangères (figure 18.15, droite).
- *Expressions asymétriques* : n-1 types d'associations fonctionnels sont transformés en n-1 clés étrangères (figure G.9, gauche) ; le dernier type d'associations permet l'intégration du type d'entités association (EMPRUNT) sous forme d'une colonne complexe (figure G.9, droite). Ce canevas permet de définir n schémas relationnels objet.

Les schémas de la famille asymétrique posent plusieurs problèmes. Le premier est leur asymétrie même : comme nous l'avons fait remarqué, la communauté objet apprécie les schémas qui incluent explicitement tous les mécanismes d'accès, fussent-ils redondants. Le schéma G.9 favorise les accès, à partir de la table EXEM-

PLAIRE, aux associations emprunte (colonne EMPRUNT) et aux lignes de PROJET et EMPRUNTEUR. Les accès au départ de PROJET et EMPRUNTEUR sont de toute évidence beaucoup plus laborieux. L'idée d'intégrer EMPRUNT sous forme d'une colonne complexe à PROJET et à EMPRUNTEUR comme nous l'avons fait pour EXEMPLAIRE doit être abandonnée : elle produirait un schéma pachydermique affecté d'une énorme redondance dont la gestion s'avérerait très complexe.

Le deuxième problème est celui de la complexité de la gestion des différentes contraintes imposées à la colonne EMPRUNT et qui garantissent l'équivalence avec le schéma conceptuel.

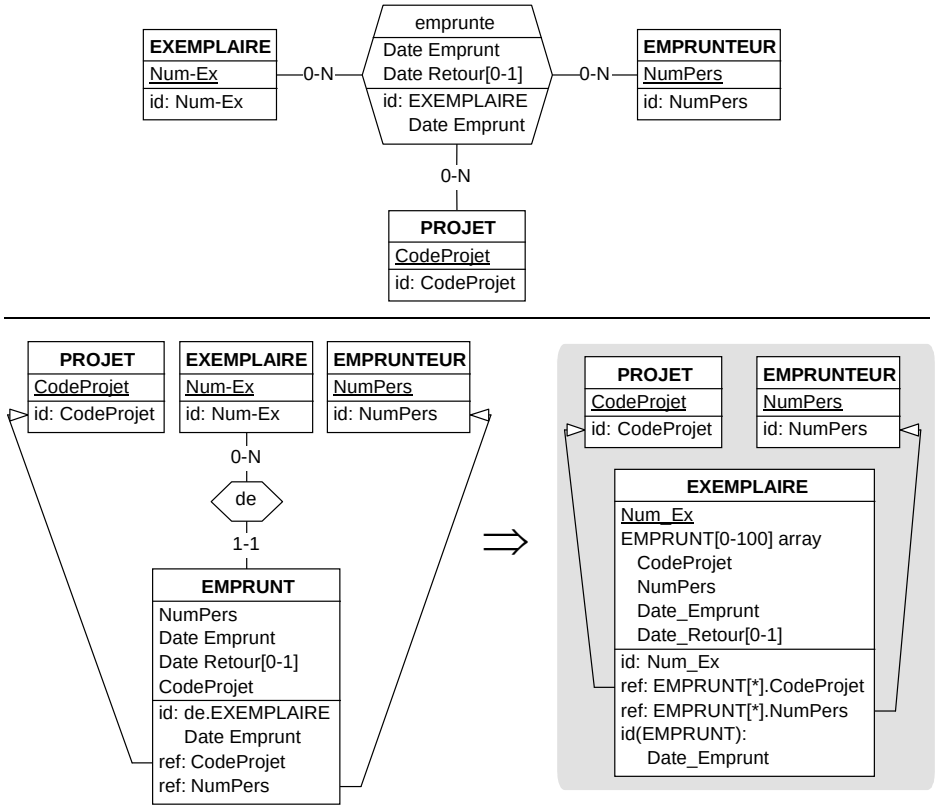

Figure G.9 - Expression asymétrique d'un type d'associations ternaire (solution non retenue)

Proposition. Sauf conditions exceptionnelles, on propose d'adopter la traduction relationnelle pure, sous la forme d'une table association munie de n clés étrangères, selon les règles de la section 18.5.

G.7 REPRÉSENTATION DES RELATIONS *IS-A*

La possibilité de définir une hiérarchie de TDU et de tables va nous permettre de représenter directement certaines relations *is-a* du schéma conceptuel sans devoir recourir à des techniques de transformation. Comme nous l'avons observé en 9.6.5, les hiérarchies de TDU sont simples[12] et munies d'une contrainte de disjonction. Les contraintes de sous-types sont limitées à D et T (et donc P). Nous devons donc examiner la traduction des *sous-types non disjoints* et des *hiérarchies multiples*.

G.7.1 Les sous-types non disjoints

Rendre disjoints des sous-types non disjoints est un problème que nous avons déjà rencontré et partiellement résolu dans le processus de normalisation (17.7.10). Nous avons observé que la transformation est très simple dans un cas de figure bien précis : les sous-types non disjoints sont au **nombre de 2** et sont des **feuilles** de la hiérarchie (eux-mêmes n'ont pas de sous-types). La transformation de disjonction des sous-types B et C est rappelée à la figure G.10, qui montre en outre une deuxième transformation, qualifiée d'*asymétrique*, dans laquelle soit C soit B, mais pas les deux, sont conservés.

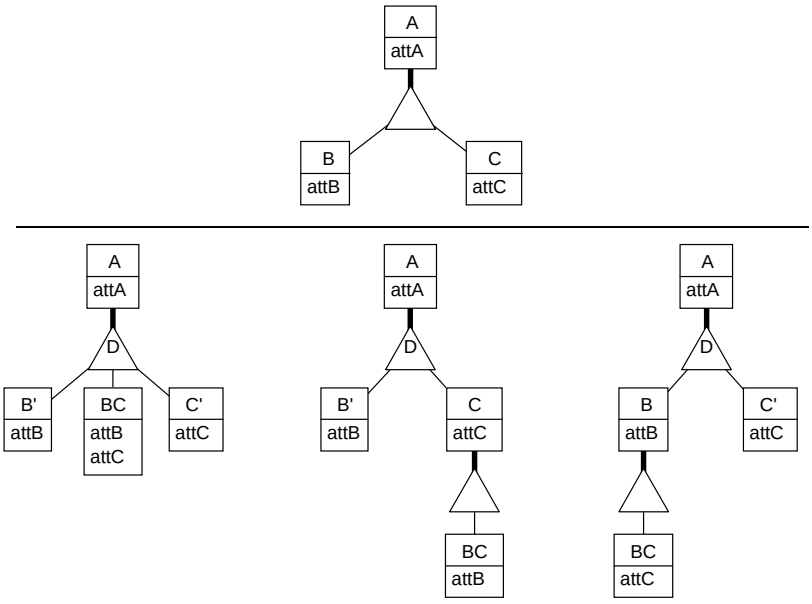

Figure G.10 - Disjonctions symétrique et asymétrique de deux sous-types

Si un des sous-types (**et un seul**) joue un rôle dans un type d'associations ou est soumis à des contraintes, on privilégiera la transformation asymétrique qui le

12. PostgreSQL admet cependant l'héritage multiple, ce qui en simplifie la conception logique.

préserve. Cependant, les choses se compliquent dans trois situations, que nous allons examiner plus en détail.

- les **deux sous-types** jouent un rôle ou sont soumis à des contraintes;
- le surtype possède **plus de deux** sous-types;
- un des sous-types B ou C n'est **pas une feuille** mais possède lui-même des sous-types.

a) Les deux sous-types jouent un rôle ou sont soumis à des contraintes

La disjonction asymétrique préserve un des sous-types et ses propriétés, tandis que l'autre subit un éclatement. Le rôle et les contraintes de ce dernier doivent donc être répartis entre ses fragments. La répartition d'un **rôle** s'effectue via un *rôle multi-type*. La figure G.11 montre deux approches : la première (a) préserve le sous-type D et répartit le rôle s.B en s.B' et s.BC; la seconde (b) se base sur une disjonction symétrique et répartit les deux rôles s.B et r.C de manière identique. Bien que d'apparence plus complexe, la seconde solution présente l'avantage de la **régularité**, qualité que nous avons mise en avant à la section 17.7.5 : *deux constructions similaires se traduisent de manière similaire*.

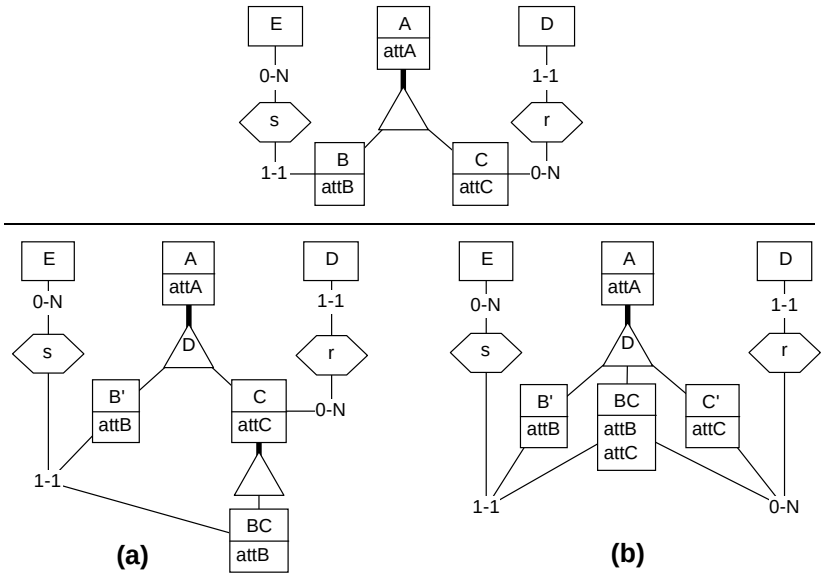

Figure G.11 - Transformation des rôles joués par les sous-types

La traduction d'une **contrainte** affectant un type d'entités B suite à l'éclatement de ce dernier peut s'avérer délicate. Alors qu'une *contrainte locale* (de domaine ou de valeurs par exemple) ou une *contrainte d'existence* sera simplement dupliquée pour chaque fragment, il n'en va pas de même des identifiants et des dépendances fonctionnelles, qui sont des *contraintes globales* qui portent sur la population entière de B. Un identifiant portant sur B dans la figure G.11 ne se traduit qu'incomplètement

par un identifiant sur B' et un identifiant sur BC. Encore faut-il garantir qu'aucune valeur de cet identifiant n'est commune à ces deux fragments. Nous avons étudié et traité cette contrainte distribuée à la section 18.6.3, figure 18.22. Si le résultat sous la forme d'une hiérarchie de TDU et de tables est jugée trop complexe, on transformera la structure par matérialisation (section 18.6.2).

b) Le surtype possède plus de 2 sous-types

La transformation de disjonction de n sous-types non disjoints génère $2^n - 1$ sous-types disjoints, soit 7 pour n = 3 et 15 pour n = 4. Le résultat devenant vite illisible, on admet que la transformation de disjonction n'est pas appropriée pour n > 2. On appliquera dans ce cas la transformation par matérialisation.

c) Un des sous-types sources n'est pas une feuille

Un des sous-types B et C possède lui-même un ou plusieurs sous-types. Il est impossible dès lors d'éclater le surtype de ces derniers[13], de sorte qu'on devra recourir à une transformation par matérialisation[14]. Il existe cependant une variante qui peut être traitée de manière fort élégante : les deux sous-types B et C ont un sous-type D en commun. Suite à la disjonction de B et C, D peut en effet être attaché au nouveau sous-type BC représentant l'intersection de B et C. La figure G.12 montre l'application de ce principe à une disjonction symétrique et à une disjonction non symétrique. Le choix de l'une ou l'autre dépendra de l'existence de rôles et de contraintes globales.

On observe que la traduction des hiérarchies *is-a* d'un schéma conceptuel quelconque dans le modèle relationnel objet peut conduire à un schéma logique complexe et irrégulier, c'est-à-dire peu lisible et difficile à maintenir et à faire évoluer. Nous devons donc élaborer des règles de traduction qui, à la fois, profitent des nouvelles fonctionnalités du modèle, restent simples à appliquer et produisent des schémas le plus réguliers possible. Considérons une construction formée d'un type d'entités A et de l'ensemble S de ses sous-types directs {B, C, ...}.

1. *S est soumis à une contrainte de disjonction*. La construction est conservée. Elle se traduit directement en une hiérarchie de TDU et de tables.
2. *S n'est pas soumis à une contrainte de disjonction mais comprend plus de deux sous-types*. On applique la transformation relationnelle par matérialisation.
3. *S n'est pas soumis à une contrainte de disjonction, il comprend deux sous-types, ceux-ci sont des feuilles dans la hiérarchie, ne jouent aucun rôle propre et ne sont soumis à aucune contrainte globale propre*. On applique la transformation de disjonction symétrique.

---

13. À noter que le concept de surtype constitué de l'*union de plusieurs types d'entités* a été proposé sous le nom de *catégorie* dans un des modèles conceptuels publiés dans la littérature : voir [Elmasri, 2006] par exemple.
14. La disjonction asymétrique n'est malheureusement pas possible dans un modèle qui n'offre pas le concept de relation *is-a* à *répartition multiple* (section 15.9.6).

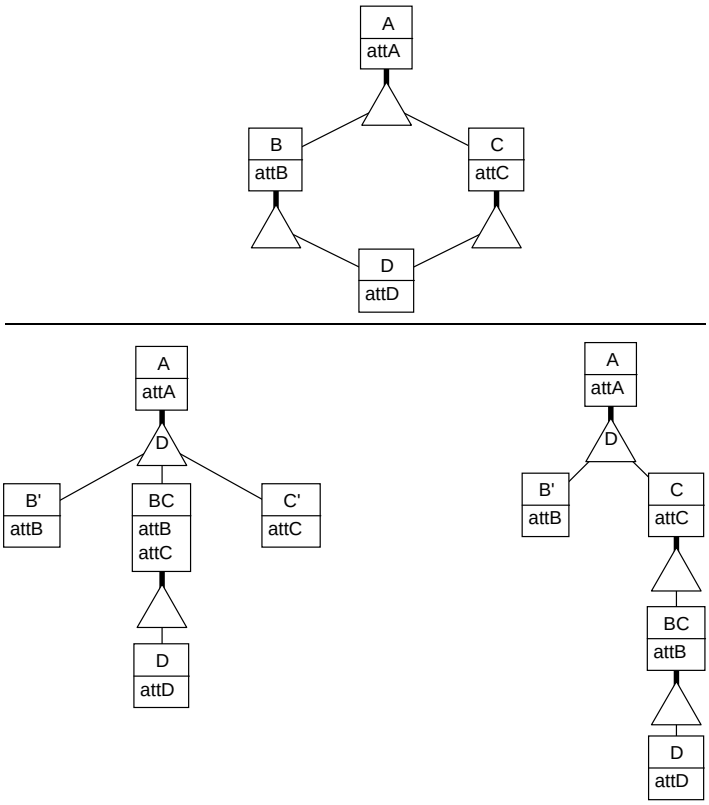

Figure G.12 - Il est possible de préserver un sous-type D commun aux sous-types B et C rendus disjoints

4. *S n'est pas soumis à une contrainte de disjonction, il comprend deux sous-types, ceux-ci sont des feuilles dans la hiérarchie, seul l'un d'entre eux joue un rôle propre et/ou est soumis à une contrainte globale propre*. On applique la transformation de disjonction asymétrique qui favorise le dernier sous-type.
5. *S n'est pas soumis à une contrainte de disjonction, il comprend deux sous-types, ceux-ci sont des feuilles dans la hiérarchie, ils jouent tous deux un rôle propre et/ou sont soumis à une contrainte globale propre*. On applique la transformation de disjonction symétrique ou, si on juge le résultat trop complexe, la transformation relationnelle par matérialisation.
6. *S n'est pas soumis à une contrainte de disjonction, il comprend deux sous-types, ceux-ci ont un et un seul sous-type commun D, ils ne jouent aucun rôle propre et ne sont soumis à aucune contrainte globale propre*. On applique la transformation de disjonction symétrique aux sous-types. D est migré vers le sous-type d'intersection BC.
7. *S n'est pas soumis à une contrainte de disjonction, il comprend deux sous-types, ceux-ci ont un et un seul sous-type commun D, l'un d'eux joue un rôle propre ou est soumis à une contrainte globale propre*. On applique la

transformation de disjonction asymétrique aux sous-types. D est migré vers le sous-type d'intersection BC.

8. Dans tous les autres cas, les relations *is-a* sont transformées, de préférence par matérialisation, conformément aux règles élaborées pour le modèle relationnel pur (section 18.6).

La figure G.13 résume l'application de ces règles.

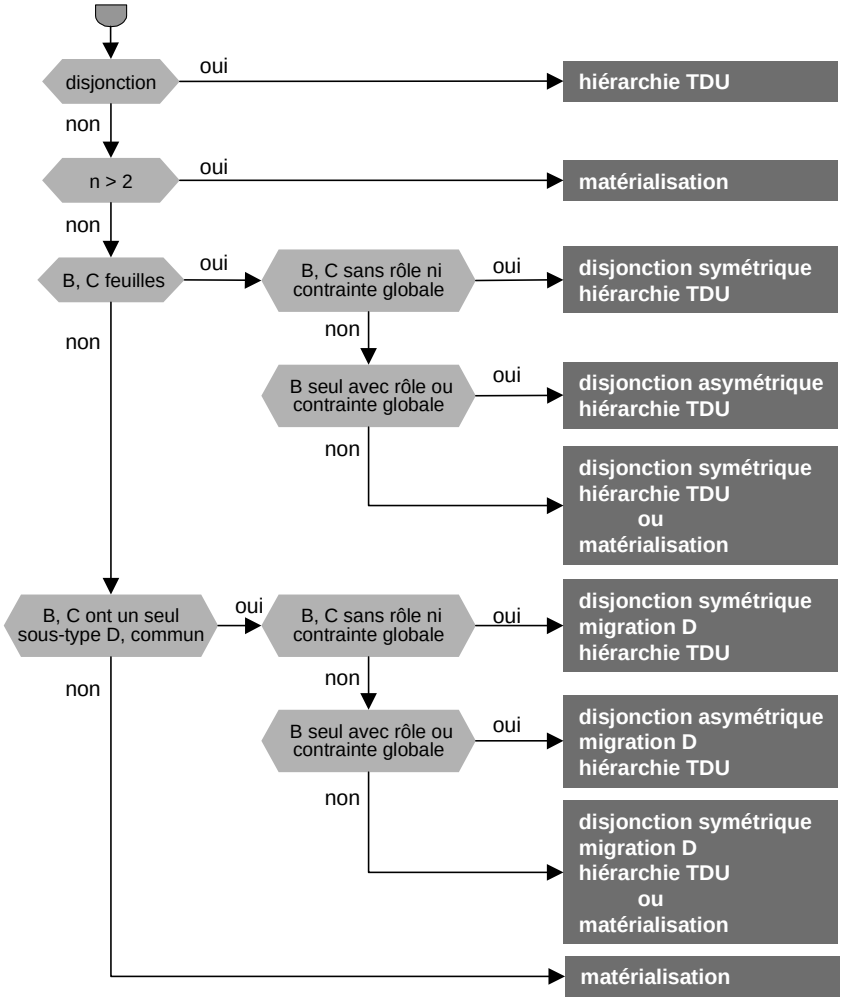

Figure G.13 - Règles de transformation d'une hiérarchie *is-a* simple

G.7.2 Les hiérarchies multiples

Dans cette analyse, nous faisons l'hypothèse que la question des sous-types disjoints a déjà été traitée. Lorsqu'un sous-type possède deux surtypes, la relation avec un des surtypes est transformée par matérialisation (section 18.6). Le choix de la relation *is-*

a à sacrifier peut être guidé par des considérations telles que la minimisation du nombre de transformations par matérialisation ou la préservation des hiérarchies sémantiquement les plus importantes.

G.7.3 Les trois techniques de base

En cas de nécessité, nous avons privilégié la technique de transformation par *matérialisation* sur base de considérations valides pour le modèle relationnel pur. On notera cependant que la transformation par héritage ascendant peut être envisagée dans la mesure où les attributs composés sont désormais admis dans le modèle relationnel objet. On évite ainsi les contraintes de coexistence et d'implication complexes rencontrées aux section 18.6.4 et 18.3.2. En revanche, le modèle relationnel objet ne favorise pas particulièrement la technique par héritage descendant.

G.8 COMPLÉMENTS

Les types d'associations avec rôle multi-types, les types d'association de composition et de matérialisation se traitent comme décrit pour la conception logique relationnelle. Certains auteurs suggèrent de traduire les types d'associations de composition dans lesquels les composants sont exclusivement liés à leur composé (rôle de cardinalité [1-1]) par un attribut complexe (voir 16.11, note de bas de page 7). De la sorte, les données relatives aux composants sont elles-mêmes des composants des données du composé, ce qui ne favorise pas l'évolutivité.

G.9 TRANSFORMATION D'UN SCHÉMA CONCEPTUEL

De la même manière que nous l'avons fait pour la conception logique relationnelle, nous pouvons désormais élaborer un **plan de transformation** spécifique au modèle relationnel objet. Le choix des techniques le plus appropriées pour chaque construction non conforme ayant déjà été détaillé dans les sections qui précèdent, nous en ferons un rappel rapide avant de proposer un plan de transformation (figure G.14).

G.9.1 Choix des transformations privilégiées

On reprend les principales constructions Entité-association pour leur appliquer les transformations adéquates.

a) Traitement des relation is-a

Dans un premier temps, toutes les relations *is-a* munies d'une contrainte de disjonction sont conservées.

Les relations *is-a* sans contrainte de disjonction sont transformées selon les règles de la figure G.13.

b) Traitement des hiérarchies is-a multiples

Si, à ce stade, le schéma comprend des types d'entités qui possèdent plus d'un surtype (hiérarchies multiples), on transforme les relations *is-a* surnuméraires à l'aide des techniques relationnelles (section 18.6) de manière à ne plus conserver qu'un surtype par sous-type. Le choix du surtype à sacrifier s'effectue sur base des critères évoqués en G.7.2. En l'absence d'indication contraire, on choisira la transformation par matérialisation sous forme de types d'associations 1:1.

c) Traitement des attributs multivalués

Si le type array est approprié, on l'utilisera pour représenter l'attribut multivalué. En particulier, si le nombre maximum de valeurs est connu et limité, si la présence de valeurs manquantes est tolérée[15], si l'existence d'un ordre sur les valeurs ne pose pas de problème, alors on peut admettre l'utilisation du type array. Même si ces conditions ne sont pas réunies, il est possible d'associer à l'UDT de la table des méthodes permettant d'émuler des ensembles à l'aide de tableaux.

Si les tableaux ne conviennent pas, on transformera l'attribut multivalué en un type d'entités et un type d'associations fonctionnel comme pour la conception logique relationnelle.

d) Traitement des types d'associations N:N et complexes

Les types d'associations N:N ou complexes sont transformés de la même manière que pour la conception logique relationnelle : type d'entités association et types d'associations fonctionnels.

e) Traitement des rôles multi-types

La transformation des hiérarchies *is-a* sans disjonction peut produire des rôles multi-types. Ceux-ci seront transformés selon la procédure décrite en 18.5.2, qui produit des types d'associations classiques.

f) Traitement des identifiants

A chaque type d'entités est associé un identifiant d'objet, propre ou hérité, si possible de même composition que l'identifiant primaire. Si ce dernier ne peut jouer ce rôle (multi-composant, instable), alors on introduit un identifiant technique. L'identifiant d'objet est facultatif pour les types d'entités associations.

g) Traitement des types d'associations fonctionnels

Les types d'associations binaires sont transformés en clés étrangères de la même manière que pour la conception logique relationnelle. L'identifiant cible est l'identifiant d'objet. Les problèmes d'identifiants non encore résolus ou manquants rencontrés lors de la conception logique relationnelle ne se posent pas ici en raison de la présence des identifiants d'objet.

15. L'amateur appréciera l'oxymore !

G.9.2 Construction du plan de transformation

À partir des solutions partielles qu'on vient d'adopter, on peut proposer le plan de transformation suivant, destiné à produire un schéma relationnel objet conforme à un schéma conceptuel quelconque. On notera que le problème des identifiants facultatifs n'est pas traité, les SGBD modernes gérant ces constructions de manière adéquate.

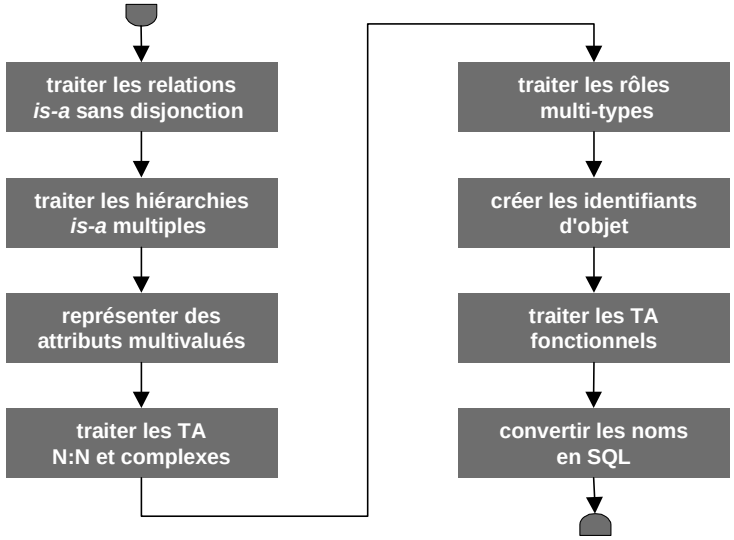

Figure G.14 - Un plan de transformation pour la production d'un schéma logique relationnel objet

G.10 CONCEPTION LOGIQUE : UN EXEMPLE

Nous illustrerons les principes de conception logique en les appliquant au schéma de la figure 15.52. Quelques observations :

- les relations *is-a* sont conformes (disjonction),
- on admet la représentation des attributs multivaluée par des tableaux (array),
- un identifiant technique est ajouté aux types d'entités AUTEUR et EXEMPLAIRE,
- on traite ensuite les types d'associations et les noms.

Le schéma de la figure G.15 montre l'état du schéma au moment où les identifiants d'objets viennent d'être créés. La figure G.16 représente le schéma logique relationnel objet. Le lecteur complétera ce schéma par les annotations exprimant les contraintes additionnelles.

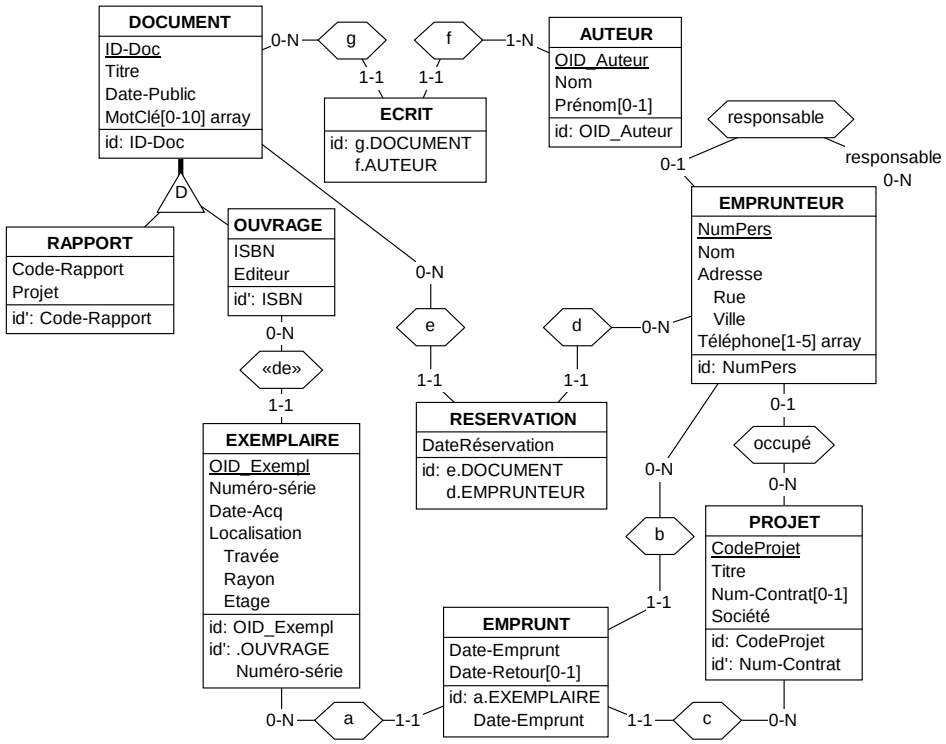

Figure G.15 - Schéma logique : stade intermédiaire

G.11 QUE RETENIR ?

Par opposition au schéma relationnel, un schéma **relationnel objet** peut comporter des **relations *is-a***, des **attributs multivalués** et des **attributs composés**. Certains objets conceptuels vont donc se traduire directement, sans transformation, en constructions relationnelles objet. Cette traduction est cependant bridée par les **limitations** affectant les nouvelles constructions : les attributs multivalués sont des tableaux, les relations *is-a* sont soumises à une contrainte de disjonction et les hiérarchies *is-a* sont simples.

La conception logique relationnelle objet se base sur les principes suivants :

1. Chaque type d'entité est représenté par un TDU et par une table de ce TDU. Chaque type d'entités source reçoit un identifiant d'objet.
2. Les relations *is-a* soumises à une contrainte de disjonction sont conservées. En cas de hiérarchie multiple, seul un surtype est conservé. Les autres relations *is-a* sont traduites par matérialisation. Les sous-types non disjoints sont transformés par disjonction symétrique ou asymétrique, ou par matérialisation.

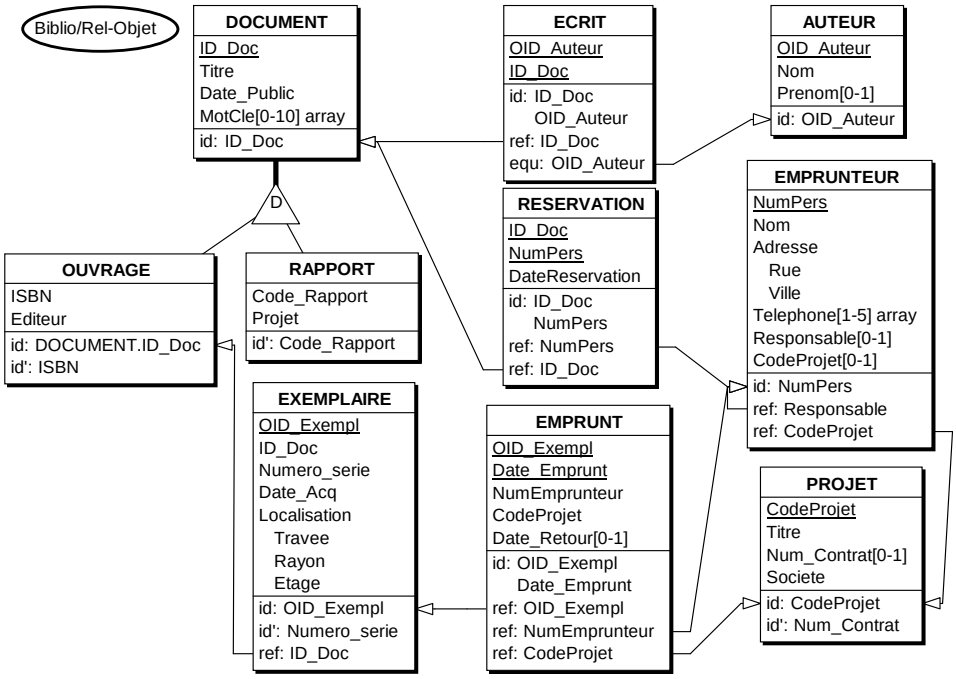

Figure G.16 - Schéma logique relationnel objet.

3. Les types d'associations fonctionnels sont traduits sous forme d'une clé étrangère comme dans la conception logique relationnelle.
4. Les types d'associations N:N et complexes sont traduits sous forme d'une table d'associations et de clés étrangères comme dans la conception logique relationnelle.
5. Les attributs multivalués sont traduits en tableaux ou, lorsque cette forme ne convient pas, sous forme d'une table comme dans la conception logique relationnelle.
6. Les noms des objets du schéma sont rendus conformes à la syntaxe SQL.

Il existe plusieurs manières de produire un schéma relationnel objet. La méthode retenue dans ce chapitre favorise la régularité et la lisibilité du schéma. On rappelle enfin qu'un schéma relationnel est également un schéma relationnel objet.

G.12 POUR EN SAVOIR PLUS

Il faut observer que la littérature sur la conception logique relationnelle objet est beaucoup plus discrète que celle qui concerne le modèle relationnel pur. L'introduction récente de cette technologie, sa complexité, son absence de normalisation en pratique[16], l'immaturité de ses implémentations, son adoption plutôt tiède par les

praticiens et surtout les difficultés que nous n'avons fait qu'aborder dans ce chapitre sont des raisons assez convaincantes. La littérature scientifique elle-même reste prudente. Trois références destinées au lecteur motivé : [Markos, 2001], [Mok, 2007] et [Mork, 2007]. On mentionnera enfin l'effort courageux de vulgarisation de la référence [Soutou, 2007], dans laquelle on trouvera des techniques assez similaires à celles du présent ouvrage et leur comparaison.

Le lecteur cultivé pourrait s'étonner de la complexité des procédures décrites dans ce chapitre. Il pourrait, à juste titre, faire remarquer que l'autres ouvrages sur le même sujet ne s'embarrassent guère de subtilités telles que celles qui ont été détaillées dans l'analyse des sous-types non disjoints. Nous ferons remarquer à ce lecteur que (1) ignorer un problème est une façon facile mais indigne de (ne pas) le résoudre et que (2) c'est se priver de bien des plaisirs que de chercher à éviter les défis que nous posent les technologies d'aujourd'hui.

Le modèle relationnel objet autorise de nombreuses variantes de traduction, qui se distinguent par leurs propriétés de performance ou de possibilités d'accès. Les raisonnements de conception logique se compliquent donc aussi par la prise en compte de critères d'efficacité et de programmabilité que nous voudrions rejeter au niveau de la conception physique. Nous n'avons pas rencontré de tels problèmes pour la conception logique relationnelle. Les choix de traduction que nous avons fait dans le présent chapitre tentent de minimiser cette variabilité.

Certains experts estiment que l'introduction des concepts d'objet dans le modèle relationnel n'est pas une réussite : *too short too late.* Non seulement la fusion de deux approches, a priori antagonistes, conduit-elle à un résultat complexe, hétérogène et irrégulier, mais le modèle relationnel objet lui-même risque bien de se faire dépasser par les *frameworks* de développement (EJB, Hibernate, Ruby on Rails, et ADO.NET par exemple), qui offrent au programmeur la possibilité de développer des classes qui modélisent directement les objets métiers tout en produisant ou gérant la correspondance objet/BD de manière automatique[17]. Pour un nombre croissant de programmeurs Java, la base de données est devenue un service aussi opaque que le stack TCP/IP de leur station de travail.

16. On comparera par exemple Oracle 11 avec les normes SQL3 (SQL:1999) et SQL 2003.
17. Ce pont entre Java/C# et la BD, dit ORM (*Object-Relational Mapping*), est principalement établi aujourd'hui vers les bases de données relationnelles. Ce qui confirme notre remarque.

G.13 EXERCICES

G.1 Proposer un schéma relationnel objet pour le schéma conceptuel de la figure G.17.

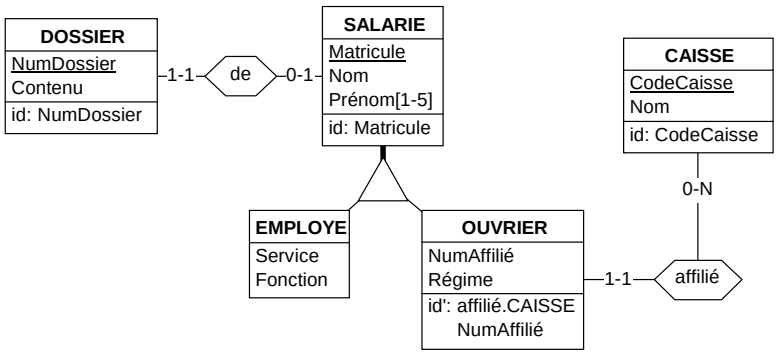

Figure G.17 - Le monde du travail

G.2 Transformer la hiérarchie *is-a* de la figure G.18 de manière à la rendre conforme au modèle relationnel objet.

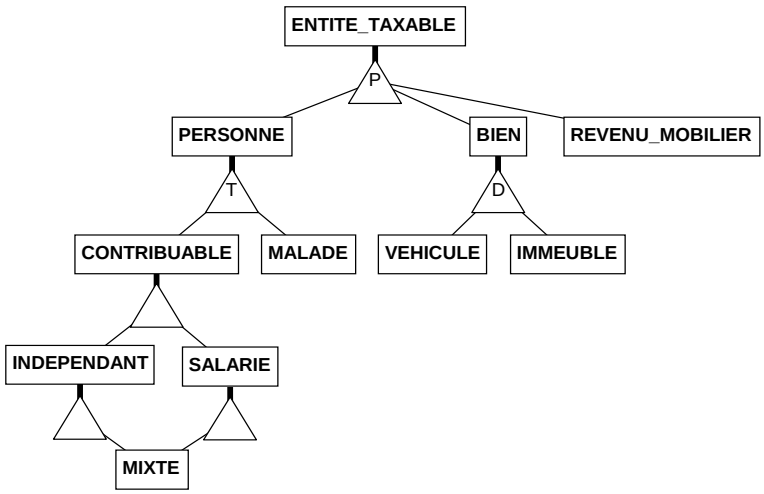

Figure G.18 - Univers économique

G.3 Proposer un schéma relationnel objet pour le schéma conceptuel de la figure G.19.

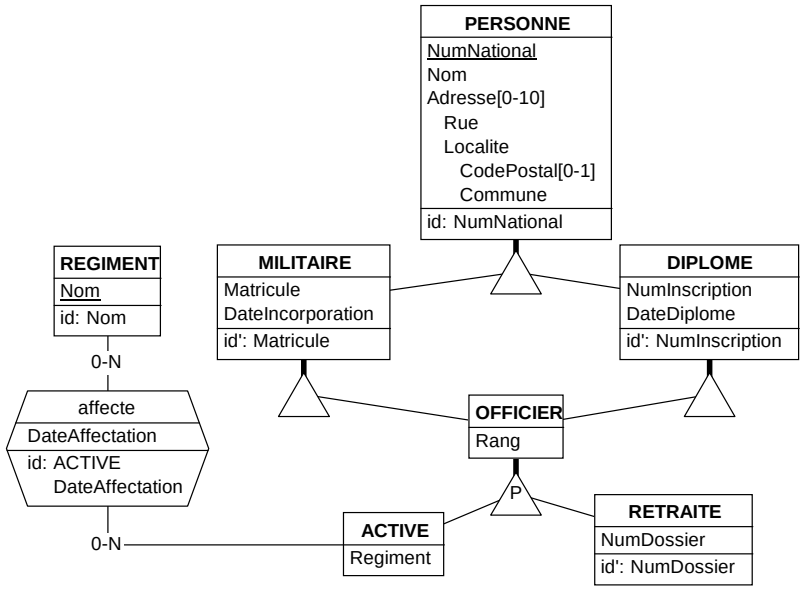

Figure G.19 - Corps d'armée

G.4 Transformer le schéma de la figure G.20 en schéma relationnel objet.

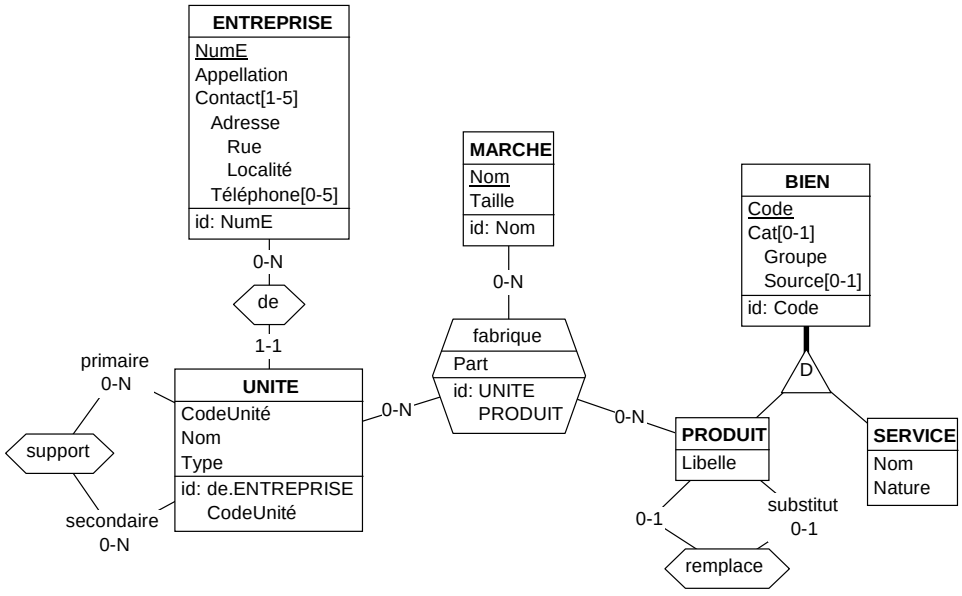

Figure G.20 - Fabrication

G.5 Développer un plan de transformation relationnel objet pour les diagrammes de classes UML.

Exercices corrigés

- **Exercice G.1**. Le principal problème de représentation est celui de la hiérarchie *is-a* sans contrainte de disjonction. On adopte dans un premier temps la transformation de *disjonction asymétrique*, ce qui donne le schéma G.21.

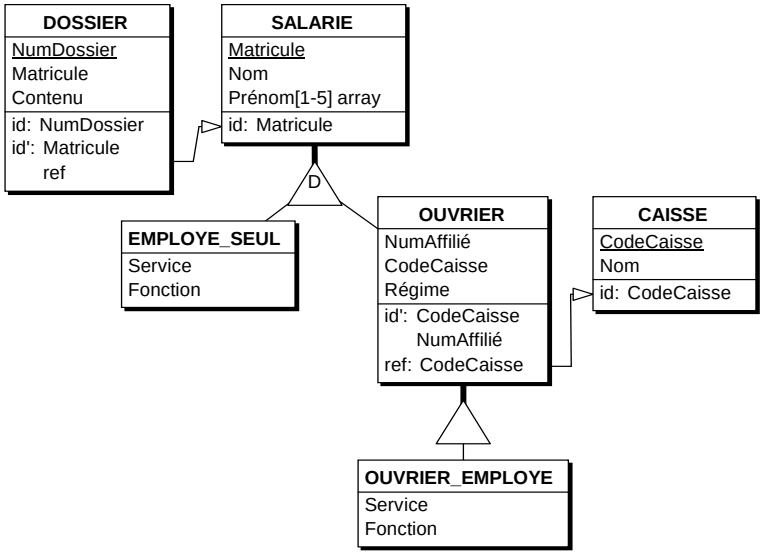

Figure G.21 - Le monde du travail, version relationnelle objet (1)

Par une *disjonction symétrique*, on obtient le schéma G.22, dans lequel il conviendrait d'ajouter une contrainte d'identifiant secondaire global pour les tables OUVRIER_EMPLOYE et EMPLOYE_SEUL.

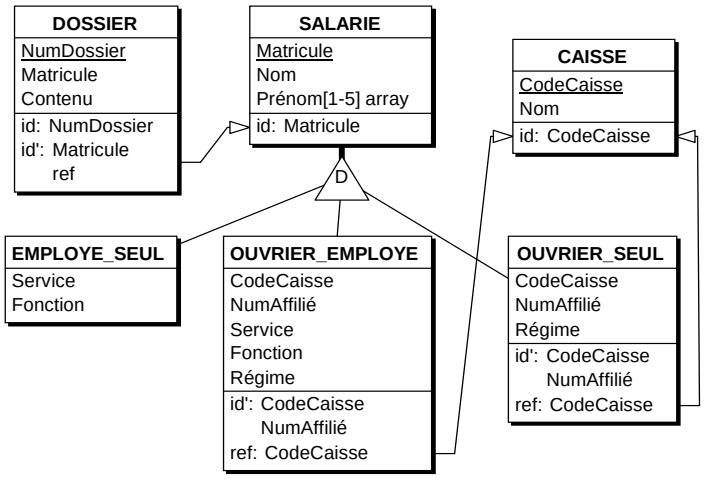

Figure G.22 - Le monde du travail, version relationnelle objet (2)

• **Exercice G.2**. La sous-hiérarchie (ENTITE_TAXABLE (PERSONNE, BIEN (VEHICULE, IMMEUBLE), REVENU_MOBILIER)) est conforme. La sous-hiérarchie de racine CONTRIBUABLE ne l'est pas, mais est aisée à transformer par *disjonction symétrique*. Le problème principal est la sous-hiérarchie non disjointe issue de PERSONNE. On la transforme par matérialisation (schéma G.23).

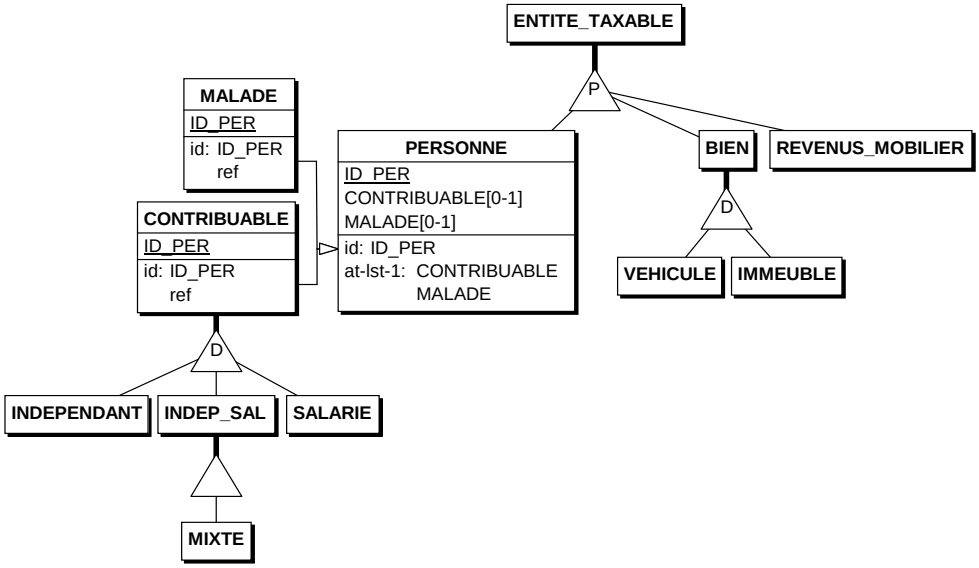

Figure G.23 - Univers économique, version relationnelle objet

• **Exercice G.3**. Malgré la hiérarchie sans disjonction et l'héritage multiple, ce schéma se traduit de manière assez naturelle, grâce à l'unique sous-type de MILITAIRE et DIPLOME. On convertit les sous-types de PERSONNE par disjonction symétrique, de manière à obtenir le schéma G.24.

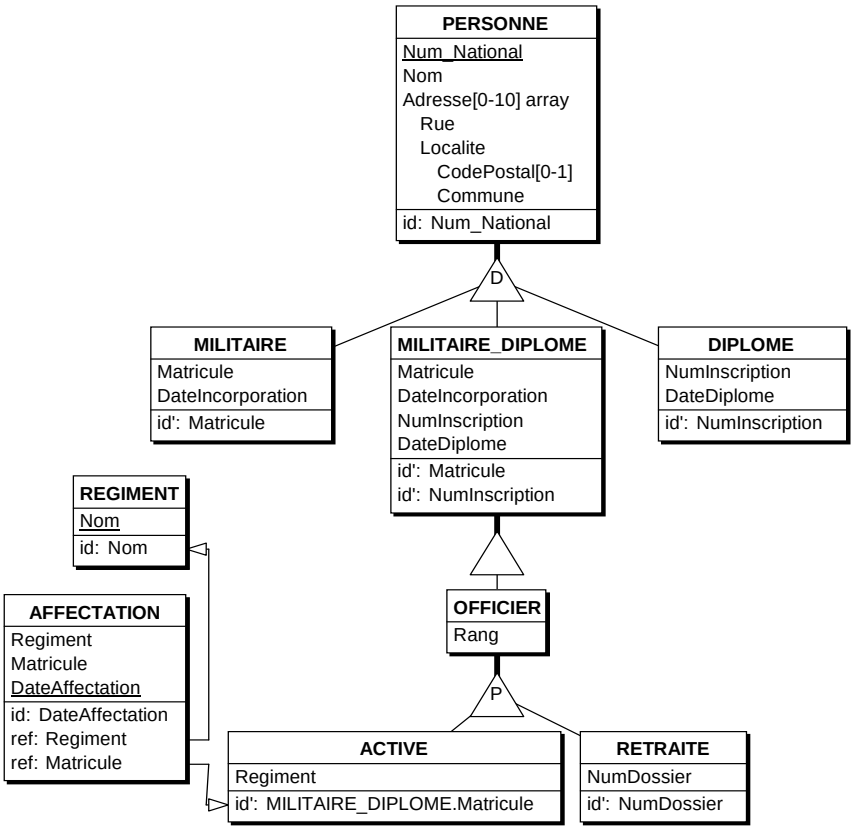

Figure G.24 - Corps d'armée, version relationnelle objet