Date de dernière modification *: 15/06/2015*

Annexe **25**

Transformations de schémas

Cette annexe décrit le concept de transformation de schéma, qui a été largement utilisé dans la troisième partie de l'ouvrage. Elle répertorie également, classées selon le type d'objet source, les principales transformations utilisées dans cet ouvrage.

A25.1INTRODUCTION

La dérivation d'un schéma à partir d'un autre peut être réalisée au moyen d'opérateurs dénommés *transformations de schéma*. Typiquement, un tel opérateur remplace une construction d'un schéma par une autre. Par exemple, un attribut peut être transformé en un type d'entités, une série d'attributs en un attribut composé, ou encore un type d'associations en un clé étrangère.

C'est ainsi que nous avons préparé un schéma conceptuel à l'enrichissement incrémental (analyse conceptuelle), que nous avons normalisé un schéma conceptuel ou relationnel, mis en conformité deux schémas avant leur intégration, produit un schéma relationnel à partir d'un schéma conceptuel (conception logique) et reconstruit un schéma conceptuel d'une base de données existante (rétro-ingénierie).

Dans cette annexe, on analyse sommairement les caractéristiques générales des transformations de schémas. On présente une classe de transformations remarquables, les opérateurs de mutation, puis on répertorie, sans chercher l'exhaustivité, les principales transformations utiles, classées selon la nature de l'objet transformé : attribut, type d'entités, relation *is-a*, type d'associations, contraintes. A l'occasion, mais uniquement à titre d'illustration, nous suggérons des usages appropriés pour l'une ou l'autre transformation.

A25.2CARACTÉRISTIQUES GÉNÉRALES DES TRANSFORMATIONS

Nous analyserons les différentes composantes d'une transformation, en mettant en évidence les objets qui disparaissent, ceux qui apparaissent et ceux qui sont conservés. Nous observerons les deux niveaux de transformation : schéma et instances. Enfin, nous discuterons la question de la préservation de l'information et nous proposerons un critère relatif à cette propriété. Cette section s'inspire de la référence [Hainaut, 2006], à laquelle nous renvoyons le lecteur pour plus de détails.

A25.2.1Anatomie d'une transformation

En toute généralité, une transformation **T** est un opérateur qui remplace une construction source **C** (un ensemble d'objets) d'un schéma **S1** par une autre construction **C'**, dite cible. On désignera par **S2** le schéma ainsi modifié. On peut écrire :

C' = T(C)

La figure A25.1 reprend une transformation que nous avons rencontrée à de multiples reprises : la transformation d'un attribut d'un type d'entités en type d'entités. L'attribut NumFournisseur de PRODUIT (**C**) du schéma de gauche est remplacé par une construction complexe (**C'**) dans le schéma de droite : type d'entités FOURNISSEUR, attribut NumFournisseur, identifiant {NumFournisseur}, type d'associations de et ses deux rôles. Comparé au schéma source, le schéma cible :

- a perdu l'attribut NumFournisseur de PRODUIT,
- a gagné le type d'entités FOURNISSEUR, son attribut NumFournisseur et son identifiant {NumFournisseur}, le type d'associations de et ses deux rôles,
- a conservé le type d'entités PRODUIT.

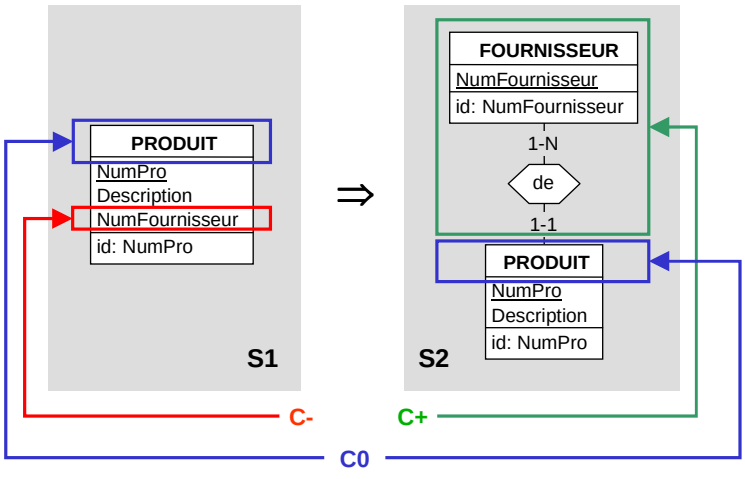


Figure A25.1 - Une transformation classique

Il est important de mentionner le type d'entités PRODUIT bien qu'il reste inchangé durant la transformation. Il y joue en effet un rôle passif mais important : parent de NumFournisseur dans le schéma source et rôle de.PRODUIT dans le schéma cible. Les autres composants du schéma source (attributs NumPro et Description, identifiant {NumPro}) ont également été conservés mais, comme ils ne jouent aucun rôle dans la transformation (ils appartiennent à **S1** mais pas à **C**), nous pouvons les ignorer.

Si nous désignons par **C-** les composants de **C** absents de **C'**, **C+** les composants de **C'** absents de **C** et **C0** les composants communs à **C** et **C'**, nous pouvons écrire :

$$\mathbf{C} = \mathbf{C0} \cup \mathbf{C-}$$

$$\mathbf{C'} = \mathbf{C0} \cup \mathbf{C+}$$

A25.2.2Description d'une transformation

En fait, **C** et **C'** sont des *classes de constructions* et non des instances qui seraient spécifiques aux schémas **S1** et **S2**, ce qui ne présenterait aucun intérêt. Dans notre exemple, **C** désigne *tout attribut monovalué de niveau 1 d'un type d'entités quelconque* et non NumFournisseur en particulier.

Il existe plusieurs méthodes de description d'une transformation. L'une consiste à fournir un prédicat structurel **P** décrivant les constructions de la classe **C** ainsi qu'une séquence d'opérations élémentaires à appliquer à une construction **C** : *supprimer un objet, ajouter un objet, modifier une propriété d'un objet*. Une autre méthode, qui sera adoptée dans cette annexe, consiste à préciser, par un prédicat structurel, la précondition minimale **P** que toute instance de **C** doit satisfaire pour pouvoir faire l'objet de **T**. De même, on précise, toujours par un prédicat structurel, la postcondition maximale **Q** que satisfait toute instance de **C'** résultant de l'application de **T** à l'instance de **C**. La transformation **T** s'écrira donc aussi **<P,Q>**.

Sans rentrer dans le détail, on peut considérer un **prédicat structurel** comme un ensemble d'assertions du type : **P** ≡ "*il existe un attribut **a** de nom **A1**, de cardinalité **[i-1]**, de type **t**, associé au type d'entités **e**, de nom **A***". Ce prédicat énumère (et nomme) les composants utiles de **C** et précise les propriétés qu'ils doivent satisfaire. Si un même nom (tel que **e** par exemple) apparaît dans **P** et **Q**, alors il y désigne le même objet (lequel appartient donc à **C0**). Dans la suite, nous nous contenterons d'exprimer ces prédicats structurels sous une forme graphique.

Pour certaines transformations, la construction **C** (ou au moins **C-**) est vide : tel est le cas de l'ajout d'un type d'associations au schéma. Pour d'autres, c'est la construction **C'** (ou au moins **C+**) qui est vide, comme dans la suppression d'un attribut. Les transformations qui nous intéressent ont pour principale caractéristique de garantir *une certaine équivalence* des schémas **S1** et **S2**.

A chaque transformation **T** est associée une inverse **T'** dont l'effet est d'annuler celui de **T** sur le schéma :

$$\mathbf{C} = \mathbf{T'(C')}$$

Ainsi, la transformation de la figure 1 admet une inverse qui consiste à transformer en attribut un type d'entités qui possède certaines propriétés.

Attention, le fait qu'une transformation ait une inverse ne signifie pas qu'elle préserve l'information de la construction **C**. Pour comprendre cette propriété de préservation, il faut étudier la manière dont les instances de **C** et **C'** sont traitées lors de la transformation **T**.

Considérons une transformation **t**, associée à **T**, qui, appliquée à toute instance **c** de **C**, produit une instance **c'** de **C'** (figure A25.2). Intuitivement, **T** explique comment restructurer le schéma tandis que **t** précise la manière de convertir des données de **C** pour les rendre conformes à **C'**. **T** est la transformation de schéma et **t** la transformation d'instance[1]. On a donc :

C' = T(C)

c' = t(c)

Pour résumer, une transformation Σ est définie par le couple **<T,t>**, qu'on peut réécrire **<P, Q, t>**. Dans certaines classes de transformations, il est possible d'associer à **t** une transformation inverse **t'**, qui opére la conversion inverse de **t**.

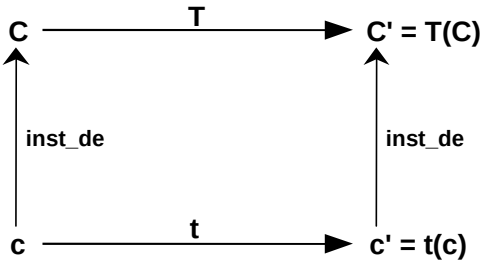


Figure A25.2 - Les composantes **T** et **t** d'une transformation

A25.3PRÉSERVATION DE LA SÉMANTIQUE

L'une des propriétés les plus intéressantes des transformations est leur capacité à préserver l'information du schéma source. Une transformation qui jouit de cette propriété est dite *à sémantique constante* ou *réversible*. Dans ce cas, les constructions **C** et **C'** décrivent exactement, bien que sous des formes différentes, les mêmes types de faits du domaine d'applications. Cette notion étant malaisée à définir formellement et en toute généralité relativement complexe, nous limiterons sa définition à celle de la *réversibilité symétrique* décrite dans [Hainaut, 2006]. Une transformation Σ = **<T, t>** = **<P, Q, t>** est dite **réversible** si, simultanément :

1. elle admet une transformation inverse Σ' = **<T',t'>** = **<P', Q', t'>**,
2. pour toute construction **C** telle que **P**(**C**) [2] et pour toute instance **c** de **C**, **C** = **T'**(**T**(**C**)) et **c** = **t'**(**t**(**c**)),

1. **T** correspond à la *syntaxe* de la transformation et **t** à sa *sémantique*.
2. L'expression **P**(**C**) s'interprète comme : *la construction **C** vérifie le prédicat structurel **P**.*

3. pour toute construction **C'** telle que **P'**(**C'**) et pour toute instance **c'** de **C'**, **C'** = **T**(**T'**(**C'**)) et **c'** = **t**(**t'**(**c'**)).

La propriété 2 impose que l'application à **C** (qui satisfait **P**) de **T** puis de l'application de **T'** au résultat ainsi obtenu regénère **C** et qu'il en est de même pour toute instance **c** de **C**. La propriété 3 précise que cette préservation est aussi d'application dans l'autre sens[3] : l'application de **T'** puis de **T** à **C'** (qui vérifie **P'**) préserve **C'**. En d'autres termes, en dehors de tout autre critère, le choix de l'une ou l'autre des constructions **C** et **C'** est indifférent. Il est en effet possible, à tout moment, de remplacer l'une par l'autre, y compris leurs instances, sans perte d'information. Sauf mention contraire, nous ne considérerons dans cette section que des transformations réversibles au sens que nous venons de décrire.

A25.4CATALOGUE DE TRANSFORMATIONS

Cette annexe a principalement pour but de répertorier les transformations les plus communes dans les différents processus étudiés dans cet ouvrage. Un tel répertoire peut être organisé de plusieurs manières : selon leur degré d'utilité décroissant, selon les objectifs à atteindre ou selon de type d'objets auquel elles s'appliquent. Nous adopterons la troisième approche, qui apparaît comme la plus générale. Cependant, avant la présentation systématique des transformations, nous décrirons une classe très importante d'opérateurs: les *transformations de mutation*.

Nous présenterons les transformations sous forme de schémas génériques dont les objets reçoivent des noms abstraits. Le schéma de gauche représente graphiquement le prédicat **P** et celui de droite la prédicat **Q**. La plupart des transformations peuvent se lire dans les deux sens. On attribue aux objets intervenant dans les constructions **C** et **C'** des noms génériques : lettres majuscules (**A**, **B**, **C**, **EA2**, ... pour les types d'entités, lettres minuscules **r**, **s**, **t**, ... pour les types d'associations et lettres majuscules indicées **A1**, **A21**, **B1**, ... pour les attributs. Un même nom attribué à des objets de même genre (type d'entités, type d'associations, attribut) dans les deux schémas d'une transformation indique que ces objets sont les mêmes (ils appartiennent à la fraction **C0** des constructions). En ce qui concerne les cardinalités des attributs et des rôles, les valeurs valides sont à interpréter telles quelles (sauf mention contraire, [**1-N**] signifie exactement [**1-N**]) tandis que les valeurs symboliques [**a-b**] ou [**c-d**] peuvent être remplacées par n'importe quelles valeurs valides.

Une transformation spécifique est obtenue par substitution de noms réels à ces noms génériques et de valeurs réelles à ces valeurs symboliques.

La spécification complète d'une transformation peut être très complexe. Par exemple, lors de la transformation d'un attribut en type d'entités (figure A25.1) il faudrait indiquer avec précision ce qu'il advient de cet attribut :

- lorsqu'il est facultatif
- lorsqu'il est multivalué selon [0-N] ou [1-N]

3. D'ailleurs, les deux transformations sont liées : Σ' = **<Q,P,t'>**.

- lorsqu'il est multivalué selon [0-5] ou [1-5]
- lorsqu'il est multivalué selon [3-12]
- lorsqu'il est composé
- lorsqu'il est composé et qu'un de ses composants est soumis à une contrainte
- lorsqu'il possède une annotation sémantique exprimant sa sémantique externe
- lorsqu'il est l'unique composant d'un identifiant
- lorsqu'il est un des composants d'un identifiant
- lorsqu'il est un composant de plusieurs identifiant
- lorsqu'il participe à une contrainte d'existence,
- lorsqu'il participe à une contrainte additionnelle
- lorsqu'il est dérivable
- lorsqu'il est hérité d'un surtype
- lorsqu'il est hérité d'un surtype et qu'il est soumis à une contrainte dans le sous-type
- lorsqu'il est utilisé comme argument d'une opération
- etc.

Nous envisagerons pour certaines transformations certaines variantes parmi celles qui sont évoquées ci-dessus mais nous laisserons au lecteur le soin d'étudier la prise en compte des autres aspects. Nous limiterons la présentation aux grands principes des familles de transformations sans tenter de fournir une spécification complète des prédicats P et Q (sauf dans de rares cas).

Cette annexe décrit les transformations suivantes.

Les transformations

A25.5TRANSFORMATIONS DE MUTATION

Une transformation de mutation change le *genre* d'un objet : un attribut est transformé en type d'entités ou en type d'associations, un type d'entités en attribut ou en type d'associations, un type d'associations en type d'entités ou en attribut (figure A25.3).

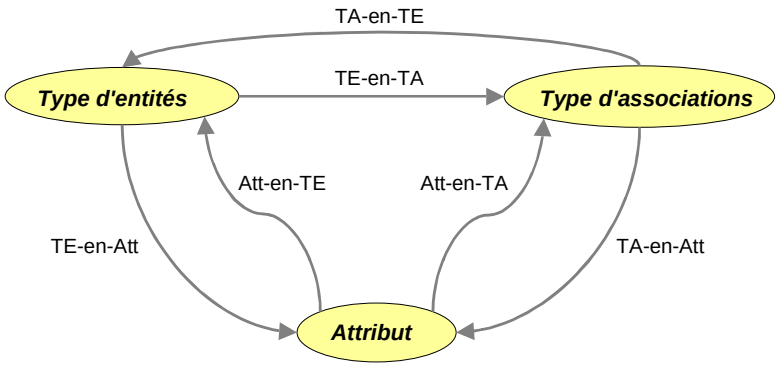


Figure A25.3 - Graphe des six transformations de mutation de base

On définit ainsi trois couples de transformations inverses (figure A25.4). Toutes ces transformations sont réversibles. Il existe plusieurs manières de changer le genre d'un objet mais celles que nous présentons ci-après sont les plus courantes. Nous reviendrons plus en détail sur certaines d'entre elles dans la suite de l'exposé.

La troisième transformation est appelée *transformation d'un attribut en type d'entités par* ***représentation des valeurs***. En effet, chaque entité EA2 représente une valeur distincte de l'attribut source A2. A ces trois couples de transformations nous en ajouterons un quatrième (figure A25.5), qui dérive des précédents, mais que nous avons déjà décrit et comparé, notamment au chapitre 12 (section 12.5.6). Cette transformation s'obtient en deux étapes à partir du schéma inférieur droit de la figure A25.4 :

1. le type d'associations **r** est transformé en type d'entités **RA2**,
2. le type d'entités **EA2** est transformé en attribut **A2'** de **RA2**.

Elle est appelée *transformation d'un attribut en type d'entités par* ***représentation des instances***. Chaque entité RA2 représente une instance de A2 (deux instances peuvent avoir la même valeur).

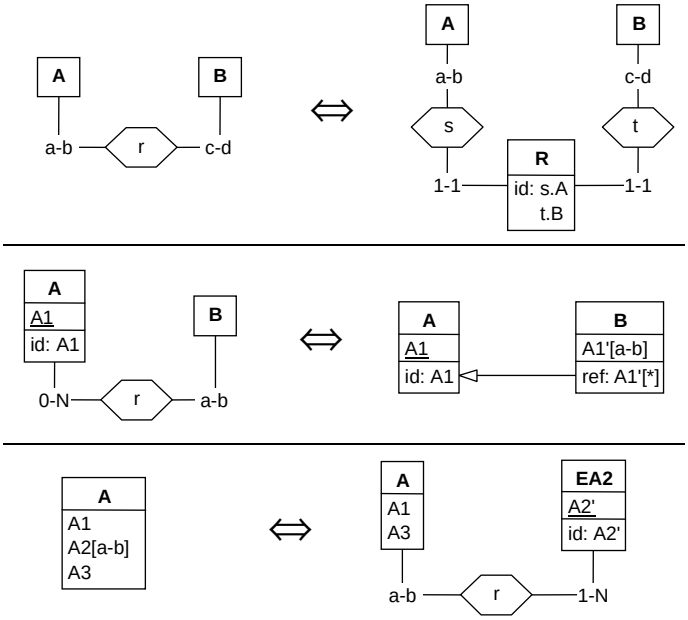


Figure A25.4 - Les six transformations de mutation de base

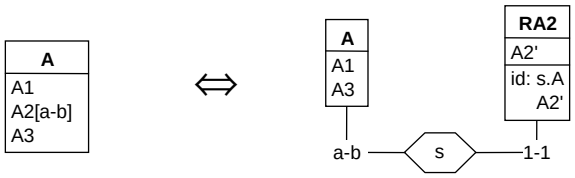


Figure A25.5 - Un couple supplémentaire de transformations de mutation

A25.6TRANSFORMATION D'ATTRIBUTS D'UN TYPE D'ENTITÉS

Cette famille de transformations est principalement basée sur la mutation Att-en-TE (figures A25.4 et A25.5). Nous y ajouterons des opérateurs de moindre qualité (mais d'égale importance) tels que décomposition, instanciation et agrégation. Ces transformations seront spécialisées successivement pour :

1. les attributs atomiques monovalués
2. les attributs atomiques multivalués
3. les attributs composés monovalués
4. les attributs composés multivalués
5. les attributs monovalués facultatifs
6. les groupes d'attributs

7. les attributs formant une clé étrangère
8. les attributs objets
9. les attributs multivalués non ensemblistes
10. les attributs définis sur un TDU (ou domaine)

Certaines transformations vont apparaître dans plusieurs sections de cette nomenclature, ce qui témoigne de leur puissance. Par exemple, la transformation d'un attribut en type d'entités sera utilisée pour résoudre des problèmes relatifs à tous les attributs, qu'ils soient obligatoires ou facultatifs, monovalués ou multivalués, atomiques ou composés.

A25.6.1Transformation d'un attribut atomique monovalué

Nous pouvons d'abord appliquer les deux transformations de mutation d'un attribut en type d'entités (figures A25.4/3 et A25.5). Nos présenterons ensuite deux opérateurs de décomposition.

a) Transformation en TE par représentation des valeurs

On envisage trois configurations fréquentes (figure A25.6) : l'attribut A2 est obligatoire, A2 est facultatif et A2 constitue un identifiant de A. Cette transformation n'est applicable qu'à un attribut de niveau 1.

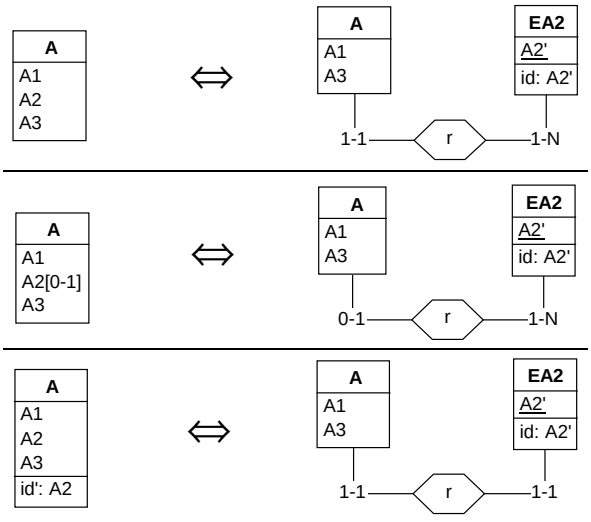


Figure A25.6 - Transformation d'un attribut monovalué en type d'entités par représentation des valeurs

b) Transformation en TE par représentation des instances

Ici encore, on envisage trois configurations fréquentes (figure A25.7) : l'attribut A2 est obligatoire, A2 est facultatif et A2 constitue un identifiant de A. Cette transformation n'est applicable qu'à un attribut de niveau 1.

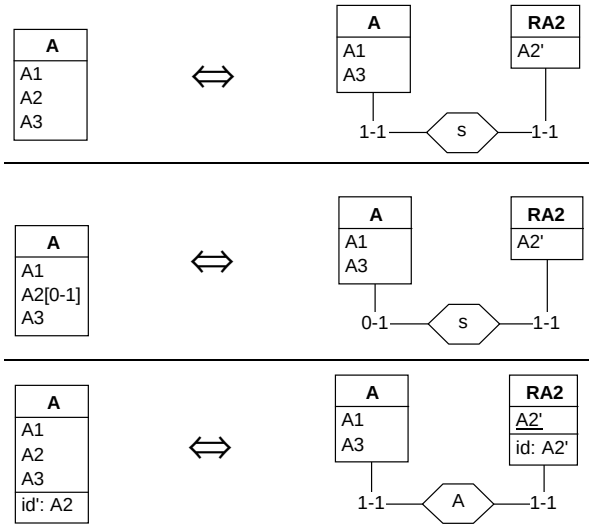


Figure A25.7 - Transformation d'un attribut monovalué en TA par représentation des instances

c) Décomposition en attribut composé

Cette transformation est l'inverse de celle *d'un attribut composé par concaténation* (section A25.6.3-d). Elle consiste à imposer à l'attribut une décomposition en sous-attributs de tailles et types éventuellement différents. On l'utilisera presqu'exclusivement en rétro-ingénierie.

d) Décomposition en attribut multivalué

Cette transformation est l'inverse de celle *d'un attribut multivalué par concaténation* (section A25.6.2-d). Elle consiste à imposer à l'attribut une décomposition en composants de taille et type identiques, formant ainsi un tableau de valeurs. On la rencontrera fréquemment en rétro-ingénierie. Elle servira également au niveau physique au traitement d'attributs de grande longueur, dont les valeurs sont ainsi tronçonnées en une suite de segments de taille fixe raisonnable, plus aisés à manipuler[4]. Les types CLOB et BLOB en SQL sont souvent traités de cette manière.

4. L'attribut multivalué liste est ensuite transformé en un type d'entités, qui se traduit physiquement en une chaîne d'enregistrements de taille fixe ou variable limitée.

A25.6.2Transformation d'un attribut atomique multivalué

Ici encore, nous appliquerons d'abord les deux transformations de mutation d'un attribut en type d'entités (réservées aux attributs de niveau 1). Nous leur ajouterons deux transformations typiques des processus d'optimisation logique et physique.

a) Transformation en TE par représentation des valeurs

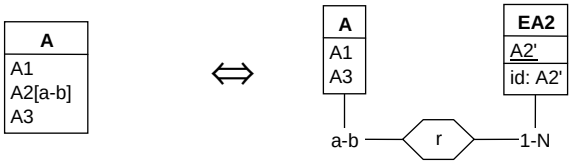


Figure A25.8 - Transformation d'un attribut multivalué en TE par représentation des valeurs

b) Transformation en TE par représentation des instances

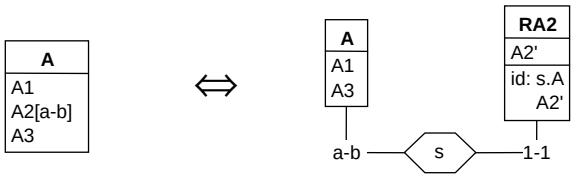


Figure A25.9 - Transformation d'un attribut multivalué en TE par représentation des instances

c) Transformation d'un attribut multi-ensemble

En principe, avant transformation d'un attribut non ensembliste, on procédera à une transformation ensembliste selon les opérateurs de la section A25.6.9. Il existe cependant une variante spécifique aux attributs multi-ensembles qui permet d'exprimer directement un tel attribut sous la forme d'un type d'entités (figure A25.10).

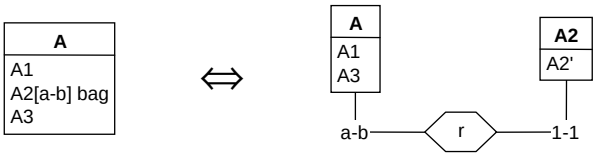


Figure A25.10 - Transformation en TE d'un attribut multi-ensemble

d) Transformation par instanciation

Chaque valeur de l'attribut multivalué de cardinalité [**a-b**] devient un attribut monovalué autonome. Un nombre **a** de ces attributs sont déclarés obligatoires, les autres étant facultatifs. *Strictu senso*, cette transformation crée une variante de *tableau* : unicité non garantie, structure d'ordre sur les attributs, pas d'obligation de regrouper les attributs sans valeur en fin de liste[5]. La transformation n'est réversible que lorsque l'attribut source A2 est un tableau[6]. Il est recommandé d'éviter cette transformation dans les approches de conception classiques.

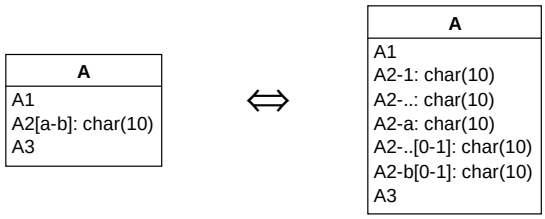


Figure A25.11 - Transformation d'un attribut multivalué par instanciation

e) Transformation par concaténation

Les valeurs de l'attribut sont concaténées dans un certain ordre de manière à constituer une unique valeur. La longueur de cette dernière est en principe **b** fois celle de l'attribut source mais dépend en fait de la manière dont les composants de la concaténation sont organisés (format fixe, séparateur, etc.) De même que la transformation par instanciation, et pour des raisons similaires, cette transformation crée une variante de *tableau*. Il est également recommandé de l'éviter dans les approches de conception classiques.

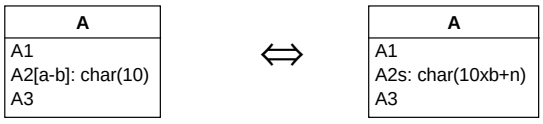


Figure A25.12 - Transformation d'un attribut multivalué par concaténation

A25.6.3Transformation d'un attribut composé monovalué

Un attribut composé monovalué peut être transformé en type d'entités, peut être décomposé et peut être concaténé.

---

5. En toute généralité, les valeurs d'un tableau ne sont pas soumises à une contrainte d'unicité, sont ordonnées (selon l'indice de leur cellule) et n'occupent pas nécessairement des cellules contiguës (existence de "trous" dans la séquence des cellules).
6. Cette affirmation est presqu'exacte ! Le schéma cible fait une hypothèse supplémentaire sur les cellules obligatoires (qui occupent les premières cellules du tableau) que le schéma source n'impose pas.

a) Transformation en TE

Un attribut composé peut se traiter comme un attribut monovalué atomique. Cependant, il est intéressant de procéder ensuite à une désagrégation de l'attribut déplacé, la forme composée n'ayant plus beaucoup d'intérêt après déplacement (figures A25.13 et A25.14). Ces transformations ne sont applicables qu'aux attributs de niveau 1.

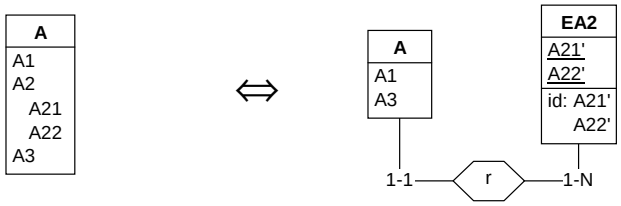


Figure A25.13 - Transformation d'un attribut composé en TE par représentation des valeurs

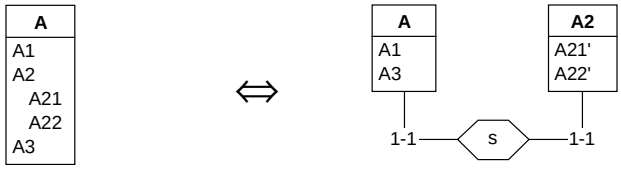


Figure A25.14 - Transformation d'un attribut composé en TE par représentation des instances

b) Transformation par désagrégation

La transformation la plus simple consiste à remplacer l'attribut composé par ses composants. L'attribut composé d'origine est évoqué par un préfixe commun ajouté aux noms des composants (figure A25.15).

Figure A25.15 - Décomposition d'un attribut composé monovalué

Lorsque l'attribut source est obligatoire, les nouveaux attributs conservent leur cardinalité d'origine (figure A25.16).

Lorsque l'attribut source est facultatif et que ses composants sont tous obligatoires, les nouveaux attributs sont facultatifs et sont soumis à une contrainte de coexistence (figure A25.17).

Figure A25.16 - Décomposition d'un attribut composé monovalué obligatoire

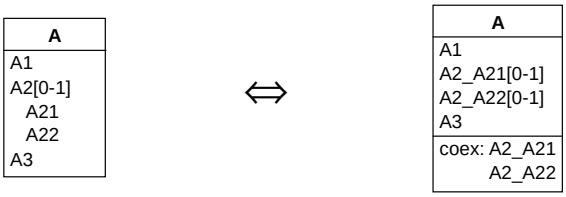


Figure A25.17 - Décomposition d'un attribut composé monovalué facultatif

Lorsque l'attribut source est facultatif et que certains de ses composants sont obligatoires alors que les autres sont facultatifs, la situation est plus complexe. Les nouveaux attributs sont facultatifs. Ceux qui étaient obligatoires sont soumis à une contrainte de coexistence. Chacun de ceux qui étaient facultatifs est soumis à une contrainte d'implication dont le deuxième membre est un des composants anciennement obligatoires (figure A25.18). Dans un tel cas, il sera sans doute préférable de transformer l'attribut composé en type d'entités.

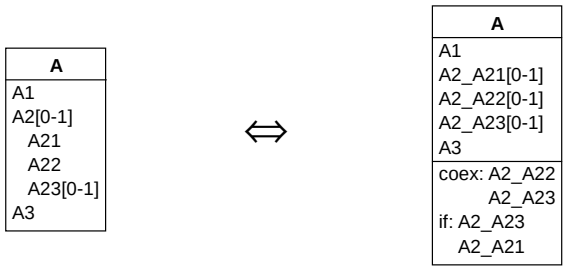


Figure A25.18 - Décomposition d'un attribut composé monovalué facultatif - Statut des composants facultatifs

Le cas d'un attribut composé facultatif dont tous les composants sont facultatifs ne peut pas être transformé par désagrégation (figure A25.19).

Figure A25.19 - Un attribut composé facultatif dont tous les composants sont facultatifs ne peut être décomposé

c) Transformation par concaténation

L'attribut composé est remplacé par la concaténation de ses composants. On suggère d'éviter cette transformation dans les approches de conception classiques. Elle n'est en effet pas réversible.

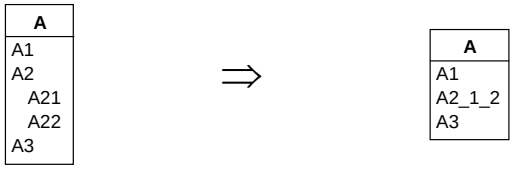


Figure A25.20 - transformation d'un attribut monovalué composé par concaténation

A25.6.4Transformation d'un attribut multivalué composé (complexe)

Un attribut à la fois multivalué et composé se traitera généralement comme un attribut multivalué. Il peut également, dans certains cas, être traité par désagrégation.

a) Transformation en TE

L'attribut est transformé en un type d'entités, soit par représentation des valeurs, soit par représentation des instances. Dans le nouveau type d'entités ainsi produit, l'attribut sera généralement décomposé (figures A25.21 et A25.22). Cette transformation n'est applicable qu'à un attribut de niveau 1.

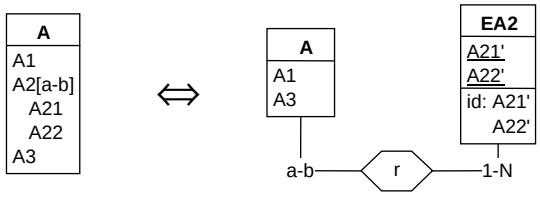


Figure A25.21 - Transformation d'un attribut complexe en TE par représentation des valeurs

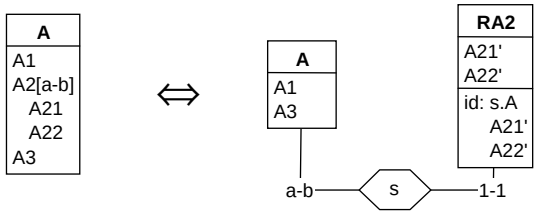


Figure A25.22 - Transformation d'un attribut complexe en TE par représentation des instances

b) Transformation par désagrégation

Il est possible de désagréger un attribut complexe pour autant qu'il soit du type *liste* ou *tableau*. Les composants de l'attribut source deviennent multivalués et se substituent à ce dernier (figure A25.23). Les éléments de même rang des attributs cibles permettent la reconstitution de l'élément de ce rang de l'attribut source.

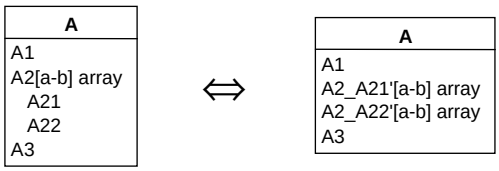


Figure A25.23 - Transformation d'un attribut complexe par désagrégation

A25.6.5Transformation d'un attribut monovalué facultatif

Il est parfois utile d'éviter les attribut facultatifs. On transformera un attribut monovalué facultatif soit en type d'entités soit par extension de son domaine.

a) Transformation en TE

La transformation d'un attribut monovalué en type d'entités a été étudié en A25.6.1/ a pour la représentation des valeurs (figure A25.24) et A25.6.1/b pour la représentation des instances. Cette transformation n'est applicable qu'aux attributs de niveau 1.

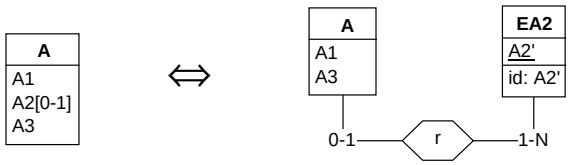


Figure A25.24 - Transformation d'un attribut facultatif en TE

b) Transformation par extension du domaine

Cette technique consiste à rendre l'attribut obligatoire mais à étendre son domaine de valeurs à une valeur conventionnelle indiquant l'absence de valeur (ω dans la figure A25.25).

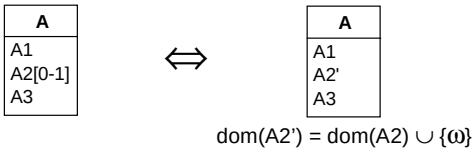


Figure A25.25 - Représentation de l'absence de valeur par une valeur conventionnelle (ω)

A25.6.6Transformation d'un groupe d'attributs

Un groupe d'attributs, dits *attributs sériels*, présentant une certaine similitude de nom, de type et/ou de longueur, peut faire l'objet d'une restructuration. On propose deux techniques.

a) Transformation par agrégation

Un groupe d'attributs de tailles et de types éventuellement distincts peuvent être rassemblés en un attribut composé. Le nom de celui-ci peut s'inspirer d'un fragment commun aux attributs du groupe source (figure A25.26).

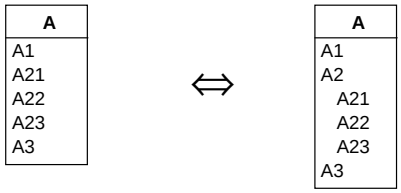


Figure A25.26 - Transformation d'attributs sériels par agrégation

b) Transformation en attribut multivalué

Un groupe d'attributs de tailles et de types identiques (ou compatibles) peut être restructuré en un attribut multivalué. En toute rigueur, cet attribut est du type *tableau*. Le nom de celui-ci peut s'inspirer d'un fragment commun aux attributs du groupe source (figure A25.27/haut).

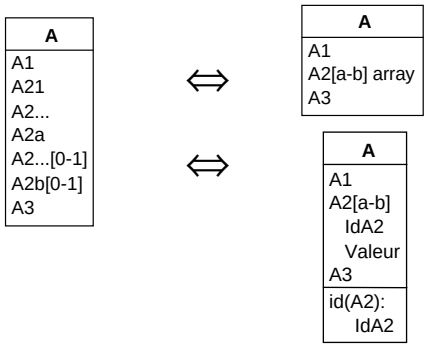


Figure A25.27 - Deux transformations d'attributs sériels en attribut multivalué

Fréquemment, le nom des attributs du groupe fournit une information importante d'identification des différents membres de ce groupe. *Exemple* : Dépenses_Janvier, Dépenses_Février, ..., Dépenses_Décembre. Une partie du nom agit comme une

dimension, temporelle dans notre exemple. Dans ce cas, il est nécessaire de conserver cette information. On peut alors proposer la transformation de la figure A25.27/bas. Le domaine de valeurs de l'attribut IdA2 est formé des parties variables des noms A21, ..., A2b du schéma source. Dans l'exemple qui vient d'être cité, IdA2 pourrait s'appeler Mois et son domaine serait {Janvier, Fevrier, ..., Decembre}.

A25.6.7Transformation d'attributs en TA

La section A25.10.1 étudie une famille de transformations (réversibles) de types d'associations binaires en clés étrangères. Transformer un groupe d'attributs constituant une clé étrangère revient à appliquer les transformations inverses (figure A25.28). Ces transformations ne concernent que les attributs de niveau 1. On rencontrera surtout cette transformation en rétro-ingénierie.

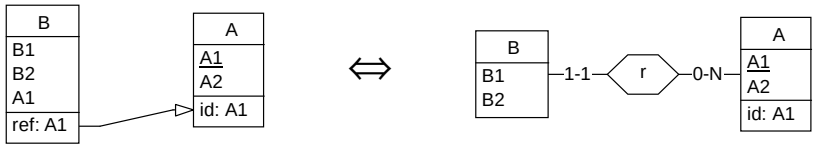


Figure A25.28 - Transformation d'un attribut constituant une clé étrangère en TA

Des contraintes additionnelles sont imposées (dans la précondition **P**). En particulier, aucun composant de la clé étrangère source ne peut apparaître dans une autre contrainte, à l'exception du cas suivant : la clé étrangère est un identifiant ou une partie d'un identifiant. A titre d'exemple, aucune des deux clés étrangères {A1, A2} et {A2, A3} d'un même type d'entités ne peut être transformée selon ces techniques (figure A25.29).

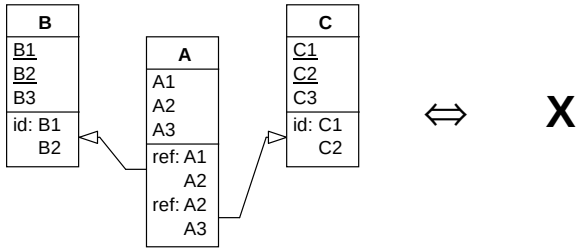


Figure A25.29 - Certains patterns de clés étrangères ne peuvent être transformés simplement en TA

A25.6.8Transformation d'un attribut objet en TA

Un *attribut objet* a pour domaine de valeurs une classe d'objets (soit en toute généralité un type d'entités). Sa valeur est donc une entité, ou une collection d'entités, du type utilisé comme domaine. La transformation remplace cette construction, typique

des schémas orientés objet classiques, voire des schémas XML, par un type d'associations équivalent. Les diverses variantes selon les propriétés de l'attribut source sont les mêmes que celles des attributs standard (section A25.6.7).

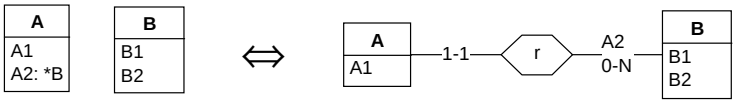


Figure A25.30 - Transformation d'un attribut-objet en type d'associations

A25.6.9Conversion d'un attribut multivalué non-ensembliste

Les attributs non-ensemblistes se trouveront le plus souvent dans les schémas logiques, bien qu'on puisse en trouver à l'occasion dans un schéma conceptuel (les prénoms d'une personne se représentent par une liste de valeurs distinctes plutôt que par un ensemble de valeurs).

a) Transformation d'un attribut multi-ensemble

Un multi-ensemble (*bag* ou *multiset*) est une collection non ordonnées de valeurs non nécessairement distinctes. Chaque valeur de A2' représente une valeur distincte (Value) de A2 ainsi que sa multiplicité (Multiplicity) (figure A25.31).

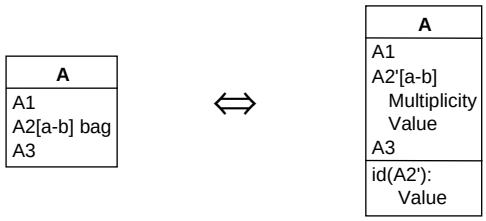


Figure A25.31 - Transformation ensembliste d'un attribut multi-ensemble

b) Transformation d'un attribut liste

Une liste (*list*) est une collection ordonnée de valeurs non nécessairement distinctes. On les représente pas un attribut ensembliste complexe dont les valeurs (Value) sont identifiées par un numéro de séquence (Sequence) (figure A25.32).

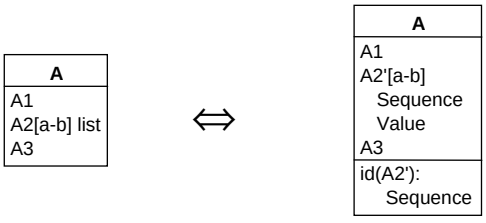


Figure A25.32 - Transformation ensembliste d'un attribut liste

c) Transformation d'un attribut liste unique

Une liste unique (*unique list*) est une collection ordonnée de valeurs distinctes. Leur représentation est similaire à celle des listes simples, à laquelle on ajoute l'identifiant d'attribut {Value} (figure A25.33).

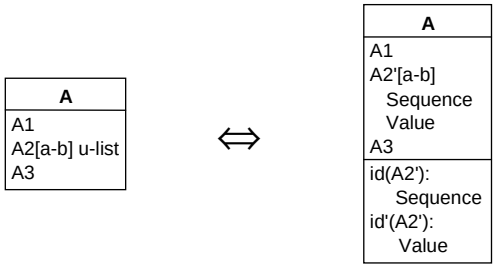


Figure A25.33 - Transformation ensembliste d'un attribut liste unique

d) Transformation d'un attribut tableau

Un tableau (*array*) est une structure de stockage comportant un nombre fixe (ou variable, ce que nous ignorerons ici) de cellules indicées pouvant contenir chacune une valeur. Les valeurs ne sont pas nécessairement distinctes. Certaines cellules peuvent ne pas contenir de valeur. Leur représentation est celle d'un attribut ensembliste complexe modélisant les cellules. Chaque cellule contient (ou non) une valeur (Value) et est identifiée par une valeur d'indice (Index) (figure A25.34).

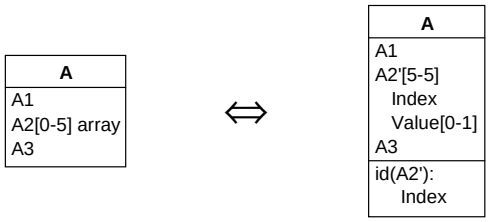


Figure A25.34 - Transformation ensembliste d'un attribut tableau

e) Transformation d'un attribut tableau unique

A la représentation des tableaux, on ajoute un identifiant {Value} (figure A25.35). Nous avons introduit une cardinalité générique [a-b] pour illustrer une contrainte qui apparaît lorsque a > 0. Dans cas, il faut vérifier qu'au moins a cellules contiennent une valeur.

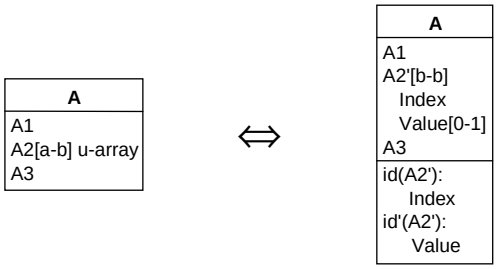


Figure A25.35 - Transformation ensembliste d'un attribut tableau unique

A25.6.10Matérialisation d'un domaine utilisateur (TDU)

Il s'agit d'une transformation technique qui sera utile pour satisfaire les modèles qui ne possède pas le concept de TDU complexe (SQL2 par exemple). On convient de ne pas la considérer comme parfaitement réversible.

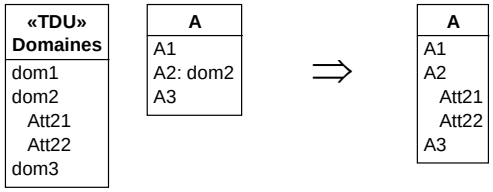


Figure A25.36 - Matérialisation du TDU d'un attribut

A25.7TRANSFORMATION D'ATTRIBUTS D'UN TYPE D'ASSOCIATIONS

Les attributs d'un type d'associations peuvent également faire l'objet de transformations. Un attribut obligatoire monovalué (éventuellement du type *objet*) peut être transformé en rôle d'un nouveau type d'entités ou d'un type d'entités existant. Certaines transformations exigeront la conversion préalable du type d'associations parent en type d'entités.

A25.7.1Transformation d'un attribut monovalué obligatoire en TE

L'attribut source transformé en un rôle joué par un nouveau type d'entités qui représente les valeurs de cet attribut (figure A25.37). Celui-ci est monovalué et obligatoire (un rôle peut être facultatif pour son type d'entités mais il est naturellement obligatoire pour son type d'associations). Il peut être composé et participer à un identifiant du type d'associations, auquel cas il y sera remplacé par le nouveau rôle.

Cette transformation n'est pas primitive. Elle résulte de la composition de (1) la transformation de r en type d'entités ER, (2) la transformation de A2' de ER en type d'entités EA2 par représentation des valeurs, (3) la transformation de ER en type d'associations r'.

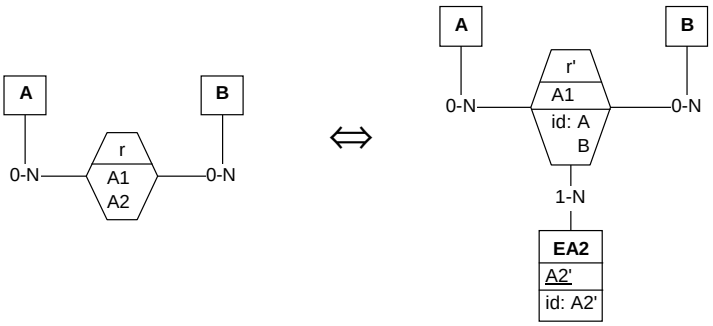


Figure A25.37 - Transformation d'un attribut d'un TA en TE par représentation des valeurs

La transformation en type d'entités admet une variante basée sur la représentation des instances de l'attribut (figure A25.38). Elle n'est pas non plus primitive, puisqu'elle peut être construite par la composition de transformations existantes.

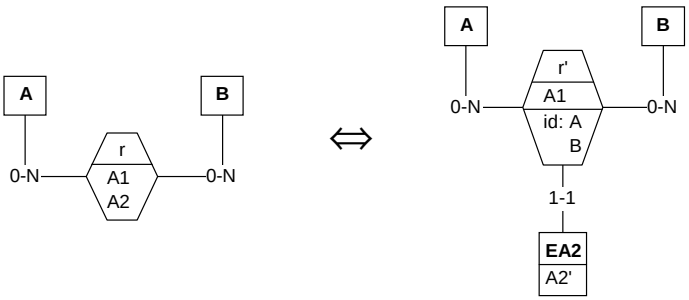


Figure A25.38 - Transformation d'un attribut d'un TA en TE par représentation des instances

A25.7.2 Transformation d'un attribut objet monovalué obligatoire en rôle

Il s'agit de l'inverse de la transformation A25.10.7. Elle exprime un attribut objet monovalué d'un TA sous la forme d'un rôle supplémentaire. On l'utilisera notamment lors de la rétro-ingénierie d'un schéma orienté objet.

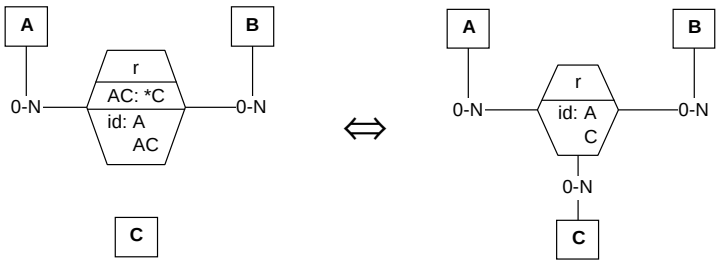


Figure A25.39 - Transformation d'un attribut objet d'un TA en rôle

A25.7.3Transformation d'une clé étrangère en rôle

Cette transformation exprime une clé étrangère d'un TA sous la forme d'un rôle supplémentaire. Le concept de clé étrangère au départ d'un TA est atypique mais peut se rencontrer en rétro-ingénierie lorsque cette clé étrangère est implicite. On utilisera surtout cette transformation dans le processus de conceptualisation en rétro-ingénierie.

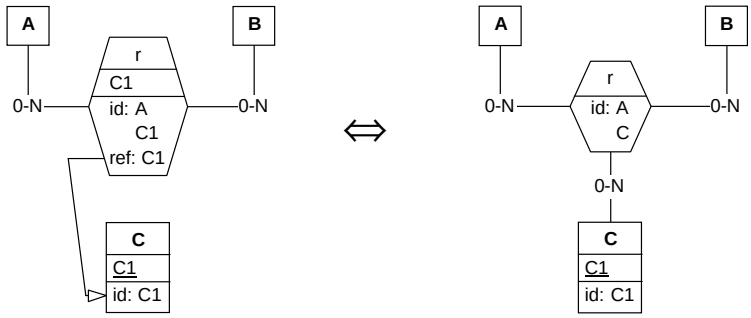


Figure A25.40 - Transformation d'une clé étrangère en rôle

A25.7.4Transformation des autres catégories d'attributs

Tous les attributs, et en particulier ceux qui sont facultatifs et/ou multivalués, peuvent faire l'objet de transformations locales (instanciation, concaténation, extension du domaine, matérialisation du domaine, attributs non-ensemblistes, attributs sériels) tout comme les attributs d'un type d'entités. Les transformations qui entraînent la création d'un type d'entités ou d'un type d'associations ne pourront être appliquées à l'attribut source qu'après transformation du type d'associations parent en type d'entités.

A25.8TRANSFORMATIONS DE TYPES D'ENTITÉS

Un type d'entités peut être transformé en type d'associations, en attribut, en plusieurs types d'entités ou encore être intégré à un autre type d'entités. On étudiera successivement :

1. les transformation d'un TE en type d'associations
2. les transformation d'un TE en attribut
3. la décomposition d'un TE
4. la fusion de TE
5. la factorisation de composants de TE
6. l'assignation d'un identifiant technique à un TE

A25.8.1Transformation d'un TE en TA

Sous certaines conditions, un type d'entités ER peut être transformé en un type d'associations r. On qualifie ER de *type d'entités association* s'il vérifie les principales préconditions suivantes :

1. ER joue au moins deux rôles, tous monotypes et appartenant à des types d'associations fonctionnels non cycliques,
2. chacun des rôles de ER est de cardinalité [1-1],
3. chacun des identifiants explicites ou implicites de ER comporte au moins un rôle opposé,
4. RE n'appartient pas à une hiérarchie *is-a*.

Le type d'associations cible r hérite des rôles sources opposés à ER; ses attributs sont ceux de ER; ses identifiants sont ceux de ER; ceux de ses identifiants qui satisfont les caractéristiques des identifiants implicites ne sont pas déclarés (figure A25.41). Cette transformation est l'inverse de la transformation d'un type d'associations en type d'entités (section A25.10.2).

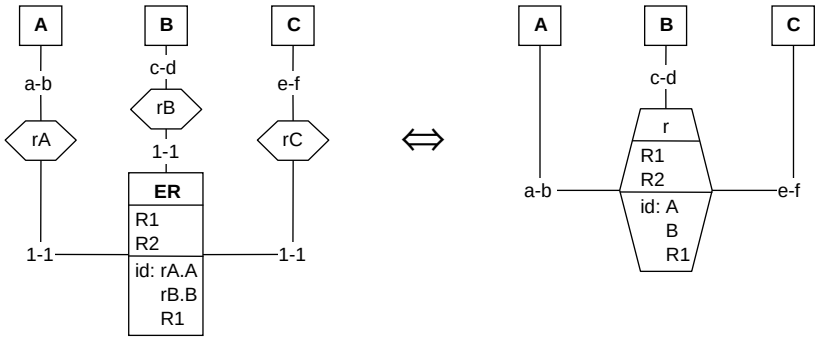


Figure A25.41 - Transformation d'un type d'entités association en TA

Il existe une transformation plus complexe mais appartenant à la même famille. Elle est applicable dans les conditions suivantes :

1. ER joue au moins deux rôles, tous monotypes et appartenant à des types d'associations fonctionnels non cycliques,
2. chacun des rôles de ER appartenant à ces types d'associations fonctionnels est de cardinalité [1-1],
3. ER joue en outre un (et un seul) rôle de cardinalité [1-N][7] dans un type d'associations rr,
4. l'unique identifiant de ER est formé des rôles opposés des types d'associations fonctionnels,
5. ER ne possède pas d'attribut,
6. ER n'apparaît dans aucune hiérarchie *is-a*.

La transformation s'explique comme suit (figure A25.42). Dans le schéma source, il existe une bijection entre ER et le produit cartésien A x B. Si on remplace dans rr chaque entité ER par le couple de A x B qui lui correspond, on obtient le type d'associations rr'.[8]

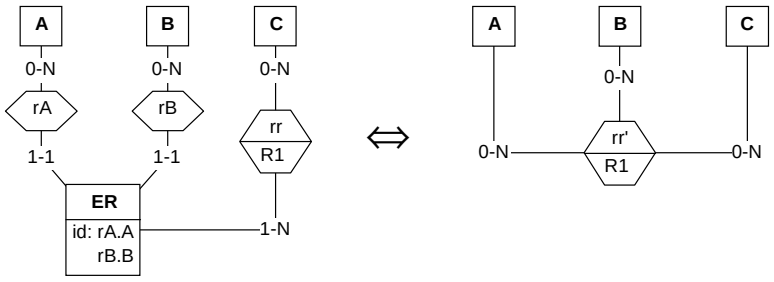


Figure A25.42 - Une transformation atypique d'un TE en TA

Il est intéressant de noter qu'en réalité rr' n'est pas l'équivalent de ER mais plutôt de rr dans le schéma source.

A25.8.2Transformation d'un TE en attribut

Sous certaines conditions, un type d'entités EA2 peut être transformé en attribut A2. On qualifie EA2 de *type d'entités attribut* s'il vérifie les principales préconditions suivantes :

1. EA2 joue un et un seul rôle dans un type d'associations r binaire non cyclique,
2. ce rôle est obligatoire,

7. Le cas [1-1] n'est pas illégal, mais il correspond alors à la transformation standard de la figure A25.41.
8. Cette transformation a été décrite et sa réversibilité démontrée dans [Hainaut, 1996].

3. EA2 possède au moins un attribut,
4. EA2 possède au plus un identifiant,
5. si EA2 possède un identifiant, celui-ci est constitué de tous les attributs de EA2; il peut en outre comprendre le rôle opposé à EA2 dans r,
6. si EA2 ne possède pas d'identifiant, alors son rôle dans r est de cardinalité [1-1],
7. si l'identifiant de EA2 comprend un rôle, alors ce rôle est de cardinalité [a-b] avec b > 1 et le rôle joué par EA2 est de cardinalité [1-1],
8. ER n'apparaît dans aucune hiérarchie *is-a*.

a) Le type d'entités attribut ne possède pas d'identifiant

La transformation produit un attribut monovalué ou un multi-ensemble selon la cardinalité du rôle opposé à EA2 (figure A25.43). Il est à noter que ce pattern impose la cardinalité [1-1] au rôle joué par EA2. Dans le cas contraire, l'identité des entités EA2 serait perdue dans la transformation, laquelle ne serait dès lors plus réversible.

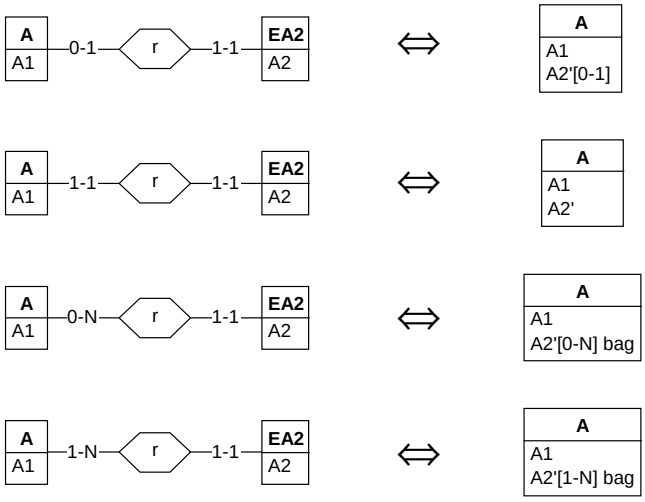


Figure A25.43 - Transformation en attribut d'un TE sans identifiant

b) Le type d'entités attribut possède un identifiant constitué d'un attribut

Lorsque le type d'entités attribut possède un identifiant constitué d'un et un seul attribut (identifiant non hybride), le schéma cible dépend de la classe fonctionnelle du type d'associations r. Les quatre classes fonctionnelles sont reprises aux figures A25.44, A25.45, A25.46 et A25.47.

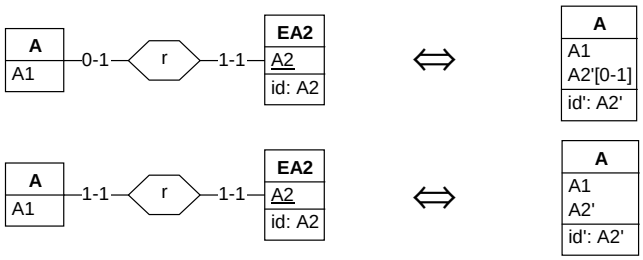


Figure A25.44 - Transformation en attribut d'un TE avec identifiant non hybride - type d'associations 1:1

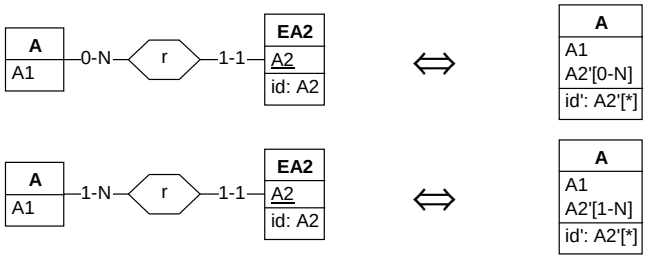


Figure A25.45 - Transformation en attribut d'un TE avec identifiant non hybride - type d'associations 1:N

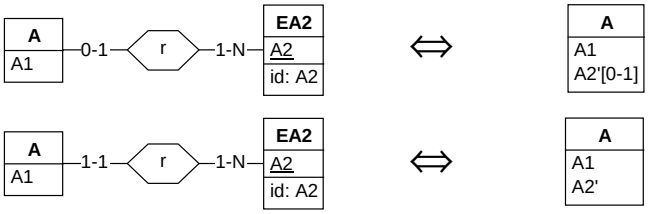


Figure A25.46 - Transformation en attribut d'un TE avec identifiant non hybride - type d'associations N:1

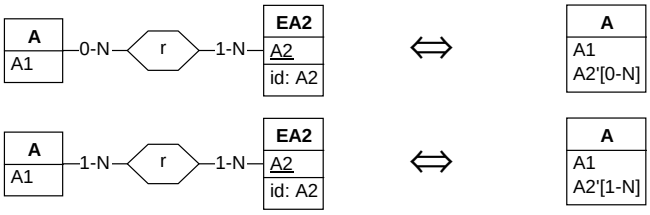


Figure A25.47 - Transformation en attribut d'un TE avec identifiant non hybride - type d'associations N:N

c) Le type d'entités attribut possède un identifiant hybride

Le type d'entités joue un rôle de cardinalité [1-1] et le rôle opposé est de cardinalité [0-N] ou [1-N] (r est de classe fonctionnelle 1:N). EA2 se transforme en un attribut multivalué identifiant de A (figure A25.48). Ces transformations peuvent être regroupées sous le pattern suivant : *le type d'entités joue un rôle de cardinalité [1-1] et le rôle opposé est de cardinalité [a-b] où b > 1; l'attribut cible A2' hérite de cette cardinalité*.

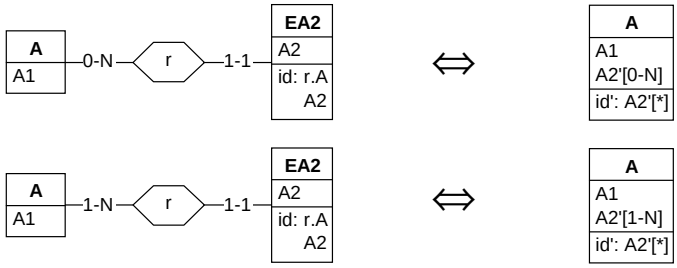


Figure A25.48 - Transformation en attribut d'un TE avec identifiant hybride

d) Le TE possède plus d'un attribut

Ces attributs sont collectivement soumis aux mêmes contraintes que l'attribut A2 de EA2 des cas précédents et jouent le même rôle. Les transformations produisent des résultats similaires, le nouvel attribut étant composé des attributs sources.

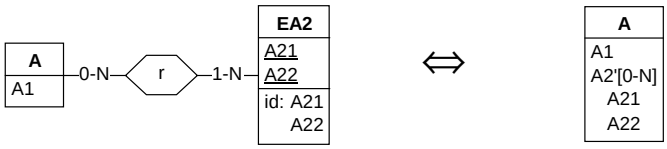


Figure A25.49 - Transformation en attribut d'un TE avec identifiant multi-attributs

A25.8.3Décomposition d'un TE

Cette famille de transformation transforme un type d'entités en plusieurs types d'entités. On y trouve les opérateurs de normalisation et de partitionnement.

a) Normalisation relationnelle d'un TE

La décomposition d'un type d'entités dont les composants (attributs et rôles) sont soumis à des dépendances fonctionnelles anormales (dont le déterminant n'est pas un identifiant) suit les raisonnements et les règles de restructuration développés dans la théorie relationnelle (chapitre 3).

La figure A25.50 illustre le processus de décomposition comme l'extraction des attributs soumis à une dépendance anormale (A3, A4) sous la forme d'un nouveau

type d'entités EA3 recueillant tous les *composants litigieux* et dont l'identifiant est le déterminant de la dépendance incriminée. Un type d'associations N:1 connecte les deux types d'entités.

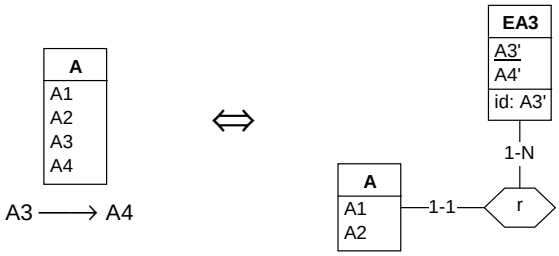


Figure A25.50 - Décomposition de normalisation d'un TE - Déterminant simple

Tant le déterminant que le déterminé de la dépendance anormale peuvent comprendre des rôles opposés au type d'entités source. La figure A25.51 montre la normalisation dans le cas où le déterminant comporte un rôle. L'identifiant du nouveau type d'entités est par conséquent hybride.

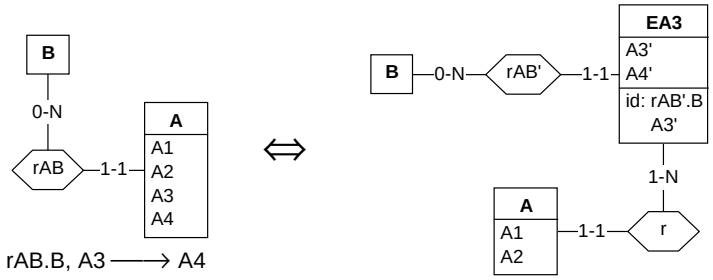


Figure A25.51 - Décomposition de normalisation d'un TE - Déterminant complexe

b) Partitionnement vertical

Le partitionnement vertical d'un type d'entités consiste à extraire un sous-ensemble de ses composants (attributs et rôles) et à les déplacer dans un autre type d'entités connecté au premier un type d'associations 1:1 (Figure A25.52). Cette transformation, à l'inverse de la précédente, ne résout pas les éventuels problèmes de normalisation. Elle est principalement utilisée en vue de l'optimisation d'un schéma logique ou encore pour isoler des composants sémantiquement homogènes d'un type d'entités trop complexe.

Si les composants déplacés sont tous facultatifs et soumis à une contrainte de coexistence, alors ils deviennent obligatoires dans le schéma cible et le nouveau rôle rAB.A est de cardinalité [0-1] (figure A25.53).

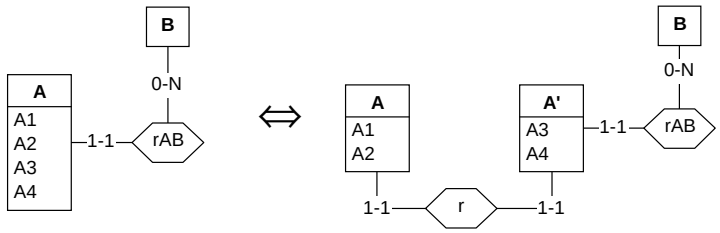


Figure A25.52 - Partitionnement vertical des composants d'un TE (1)

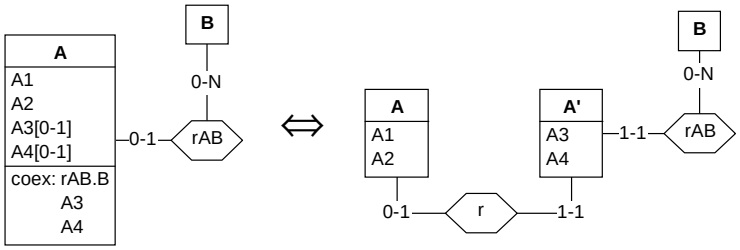


Figure A25.53 - Partitionnement vertical des composants d'un TE (2)

c) Partitionnement horizontal

Le partitionnement horizontal consiste à remplacer un type d'entités par plusieurs types d'entités de même structure, chacun d'eux accueillant un sous-ensemble de la population du type d'entités source (figure A25.54)[9]. Cette transformation est principalement utilisée pour optimiser un schéma physique.

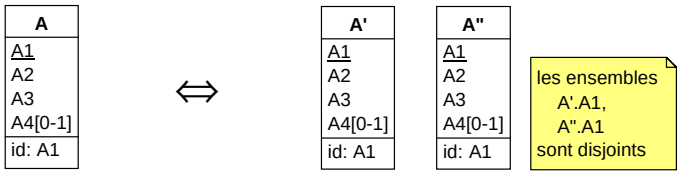


Figure A25.54 - Partitionnement horizontal de la population d'un TE (1)

Cette transformation peut cependant avoir une utilité au niveau conceptuel et logique pour clarifier l'existence de sous-population dotées de propriétés différentes. C'est ce qu'illustre la figure A25.55, qui montre en outre que le partitionnement peut être vu comme la composition de deux transformations standard.

9. On résistera à la tentation de connecter ces types d'entités par un type d'associations 1:1 !

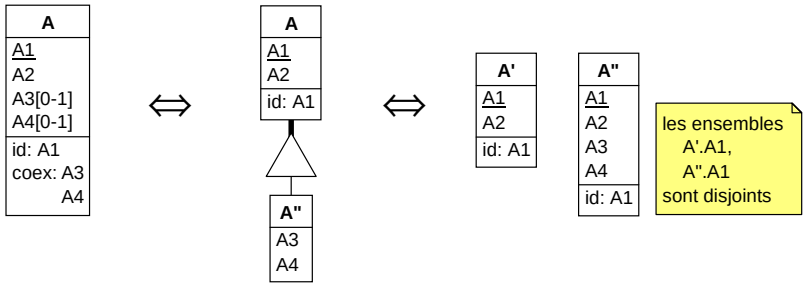


Figure A25.55 - Partitionnement horizontal de la population d'un TE (2)

A25.8.4Fusion de types d'entités

Ces transformations permettent de remplacer plusieurs types d'entités par un seul.

a) Fusion de dénormalisation

La dénormalisation d'un schéma est une transformation d'optimisation. Elle constitue l'inverse de la transformation de normalisation (A25.8.3/a). On introduit de la sorte une redondance interne conduisant à diverses anomalie de mise à jour des données.

b) Fusion bijective

Cette transformation apparaît comme l'inverse d'un partitionnement vertical (E8.3/b). On distingue généralement le type d'entités absorbant, qui conserve son identité dans la transformation, et le type d'entités absorbé, dont les composants sont migrés vers le premier (dans la figure A25.56, ces rôles sont indistincts). Elle est applicable lorsque deux types d'entités sont connectés par un type d'associations 1:1. Ce type d'associations est obligatoire ou facultatif. Dans ce dernier cas, la transformation peut induire une contrainte de coexistence dans le type d'entités absorbant. Dans certains cas simplifiés, cet opérateur correspond à la transformation d'un *type d'entités attribut* en attribut.

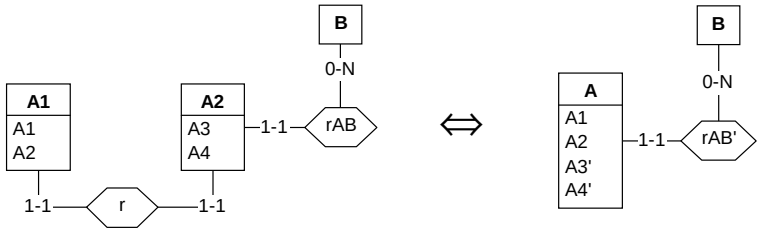


Figure A25.56 - Fusion de deux TE

c) Union de TE

Cette transformation est l'inverse d'un partitionnement horizontal (E8.3/c). Elle est applicable lorsque deux types d'entités ont le même schéma et que leurs populations sont disjointes.

A25.8.5Factorisation de composants de TE

L'idée de cette transformation est intuitive mais sa description complète est relativement complexe. Elle correspond en fait une famille de transformations.

Dans le cas le plus simple, deux types d'entités présentent des attributs qu'on pourrait qualifier de *communs* (même nom, même type, sémantique externe identique). On convient de définir un nouveau type d'entités qui regroupe les attributs communs et qui est déclaré surtype des types d'entités sources (figure A25.57). On impose une contrainte de totalité, mais l'éventuelle contrainte de disjonction dépendra de la sémantique des types d'entités.

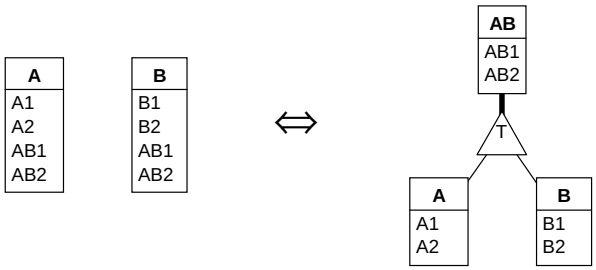


Figure A25.57 - Factorisation des éléments communs à deux TE

En toute généralité, le schéma source contient un nombre quelconque de types d'entités à traiter. En outre, les composants communs sont des attributs, des rôles de types d'associations et des contraintes. La factorisation dans ce cas produira le plus souvent une hiérarchie *is-a* à plus de deux niveaux et des surtypes multiples. La figure A25.58 en fournit un exemple élémentaire.

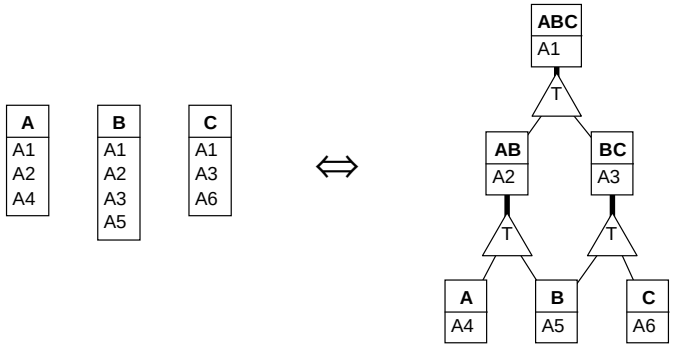


Figure A25.58 - Factorisation de plusieurs TE

Cet opérateur est l'inverse de la transformation de relations *is-a* par héritage descendant (A25.9.2).

La construction de cette hiérarchie relève d'un processus appelé FCA (Formal Concept Analysis). Cette hiérarchie est également un cas particulier de *treillis de Galois*. On en trouvera une application non triviale et des références à l'annexe F.

A25.8.6 Assignation d'un identifiant technique à un TE

Cette transformation ajoute au type d'entités un attribut technique sans signification et en fait l'identifiant primaire. Si le type d'entités source dispose déjà d'un identifiant primaire, ce dernier est repris mais sous statut de *secondaire*. Il s'agit d'une transformation technique visant plusieurs objectifs : attribuer un identifiant à un type d'entités qui n'en possède pas, substituer un identifiant simple à un identifiant jugé trop complexe ou trop long, attribuer aux types d'entités des identifiants non significatifs dans une perspective orientée objet (*object id*). On l'utilisera principalement lors de la conception logique.

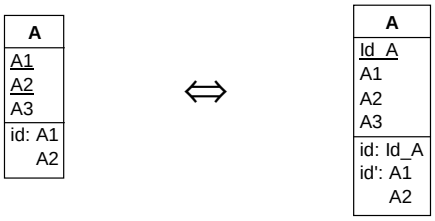


Figure A25.59 - Assignation d'un identifiant technique à un TE

A25.9 TRANSFORMATION DE RELATIONS *IS-A*

Cette famille relativement étendue de transformations comprend des opérateurs de restructuration de hiérarchies *is-a* ainsi que des opérateurs qui font disparaître et qui créent ces hiérarchies. On les utilisera dans tous les processus d'ingénierie des bases de données. Nous ne donnerons de ces transformations, souvent complexes, que les principes généraux. Nous étudierons successivement :

1. la transformation par matérialisation en TA
2. la transformation par héritage descendant
3. la transformation par héritage ascendant
4. la traduction des autres aspects
5. la disjonction de sous-types
6. la couverture d'un surtype
7. le partitionnement d'un surtype
8. les transformations d'une répartition multiple

9. les transformations d'une hiérarchie à surtypes multiples
10. les autres transformations

A25.9.1Transformation par matérialisation en TA

On considère un schéma source constitué d'un surtype et de ses sous-types. Le schéma cible **représente à la fois le surtype et ses sous-types**. Les relations *is-a* sont représentées par des types d'associations 1:1. Les contraintes de sous-types sont représentées par des contraintes d'existence (figure A25.60).

Cette transformation est celle qui préserve le mieux la topologie du schéma source et offre une évolutivité maximale mais l'éventuelle traduction relationnelle ultérieure des contraintes d'existence posera problème (voir section 18.6.2). Cette transformation ainsi que les deux suivantes sont largement utilisées en conception logique relationnelle mais aussi relationnelle objet. En effet, les contraintes qu'impose le modèle relationnel objet sur les relations *is-a* obligent dans certains cas de figure à recourir à la transformation de ces relations.

Ces opérateurs sont décrits plus en détail dans les références [Wagner 1989], [Hainaut, 1996] et [Da Silva, 2000].

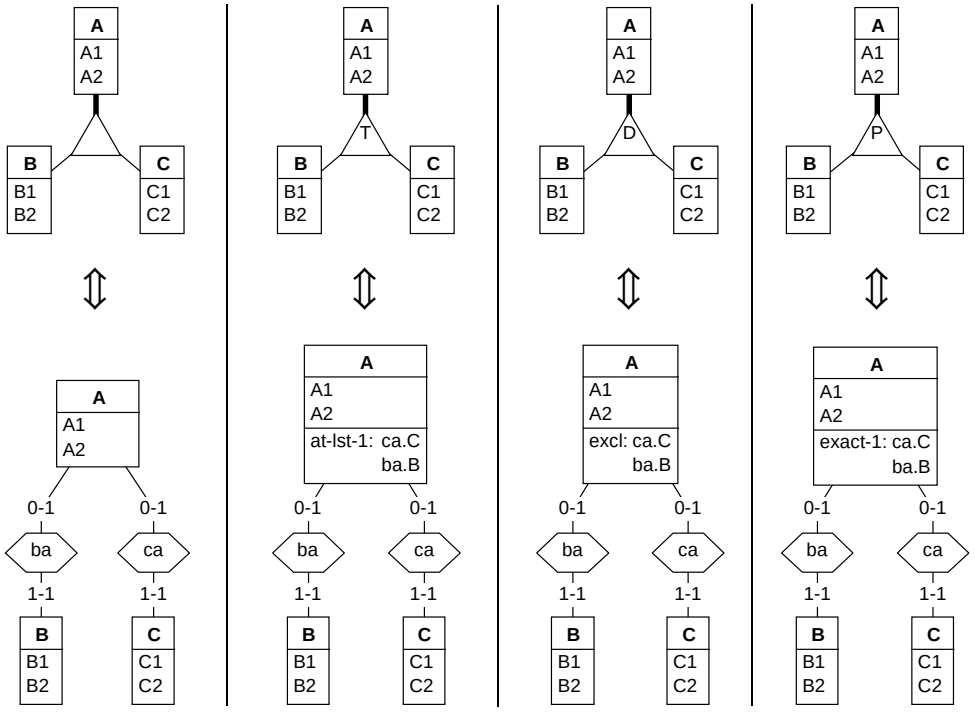


Figure A25.60 - Transformation de relations *is-a* par matérialisation

A25.9.2Transformation par héritage descendant

On considère un schéma constitué d'un surtype et de ses sous-types. Le schéma cible **ne représente que les sous-types**, lesquels sont complétés des composants de leur surtype (figure A25.61). Lorsque le recouvrement n'est pas total, on considère un sous-type qui recueille les entités du surtype qui n'appartiennent à aucun sous-type (transformation de *couverture d'un surtype*, section A25.9.6).

Cette transformation est peu appropriée aux structures dont le surtype comporte des composants importants tels que des identifiants, des rôles et des contraintes. La distribution de ces composants parmi les sous-types pose en effet des problèmes d'expression non triviaux (section 18.6.3).

En outre, la non disjonction des sous-types entraîne des difficultés importantes relatives à des redondances internes qui apparaissent dans les structures cibles. On observe également que cette transformation repose sur une contrainte de totalité, puisque, lorsqu'elle n'existe pas, on la récrée sous la forme d'un sous-type complément A'. En résumé, cette transformation n'est réellement applicable que dans une structure soumise à une contrainte de **partition**.

On se reportera à la section 18.6.3. pour plus de détails sur cette transformation. On y décrit en particulier un compromis permettant de résoudre simplement le problème des identifiants du surtype.

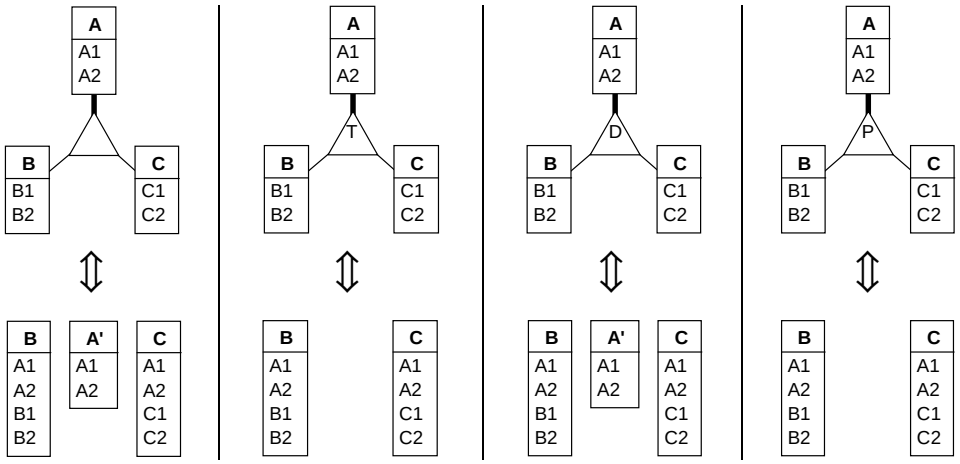


Figure A25.61 - Transformation de relations *is-a* par héritage descendant

A25.9.3Transformation par héritage ascendant

On considère un schéma constitué d'un surtype et de ses sous-types. Le schéma cible **ne représente que le surtype**, lequel est complété des composants de chacun des sous-types (figure A25.62). Les attributs d'un sous-type sont rassemblés sous la forme d'un attribut composé facultatif. Les rôles des sous-types deviennent facultatifs et font l'objet d'une contrainte de coexistence avec l'attribut composé propre à

son sous-type. Les contraintes des sous-types sont traduites en contraintes d'existence imposées aux attributs composés propres aux sous-types.

Cette transformation produit un schéma cible peu naturel et peu lisible dans lequel les surtypes paraissent hypertrophiés. En revanche, contrairement aux deux autres transformations, la traduction complète du schéma cible en structures relationnelles n'exigera que des techniques légères telles que des *checks*.

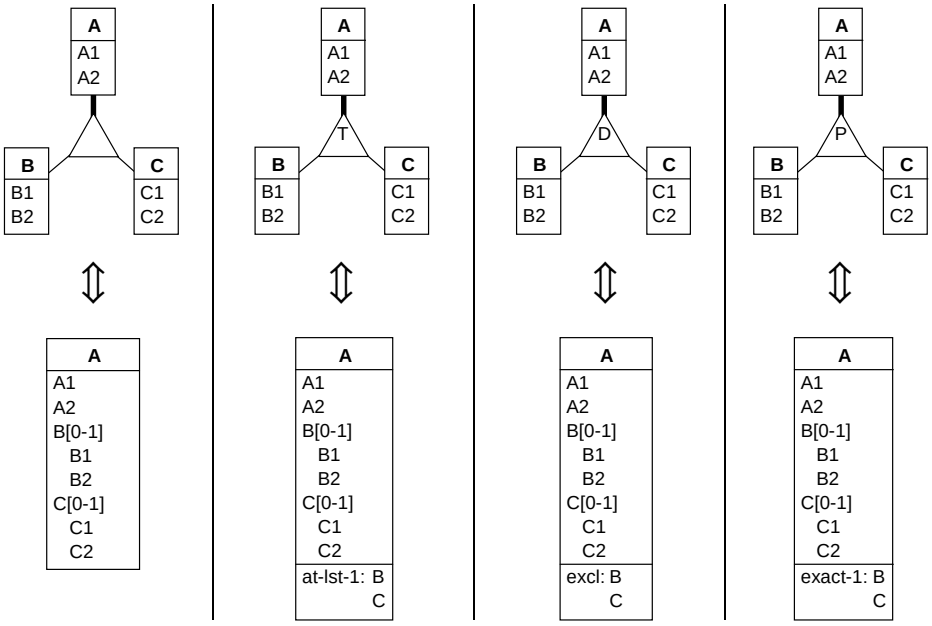


Figure A25.62 - Transformation de relations *is-a* par héritage ascendant

A25.9.4Traduction des autres aspects

Dans chacune des transformations précédentes, nous n'avons étudié que le sort des attributs. Il faut en outre considérer la manière dont sont traités les identifiants, les rôles et les contraintes pour le surtype et les sous-types. Ces aspects sont discutés dans [Hainaut, 1996].

A25.9.5Disjonction de sous-types

L'ensemble des sous-types d'un surtype est transformé de sorte que ces sous-types soient disjoints. Cet opérateur est nécessaire en conception logique basée sur un modèle qui admet les relations *is-a* mais impose la contrainte de disjonction. Tel est le cas de SQL3, C++ et Java par exemple. On trouvera une discussion relative à cette transformation dans le cadre de SQL3 à la section G.7.1.

Il s'agit plus généralement d'une famille de transformations. Elle s'avère plus complexe qu'elle pourrait paraître à première vue. Nous traiterons principalement le

cas de structures à deux sous-types. Les sections suivantes n'abordent ces transformations que de manière superficielle. Leur approfondissement dans des cas particuliers est laissée à l'initiative du lecteur.

a) Disjonction de deux sous-types

La transformation d'un ensemble de deux sous-types non disjoints en un ensemble de sous-types disjoints admet deux variantes : symétrique et asymétrique. On considère pour l'instant que ces sous-types n'ont pas eux-mêmes de sous-types.

La transformation symétrique (figure A25.63) remplace l'ensemble des deux sous-types par les trois types suivants : B' qui représente les entités B qui n'appartiennent pas à C, C' qui représente les entités C qui n'appartiennent pas à B, et BC qui regroupe les entités d'intersection appartenant à la fois à B et C. Les propriétés des populations des types d'entités du schéma cible sont les suivantes :

B' = B - C
C' = C - B
BC = B∩C

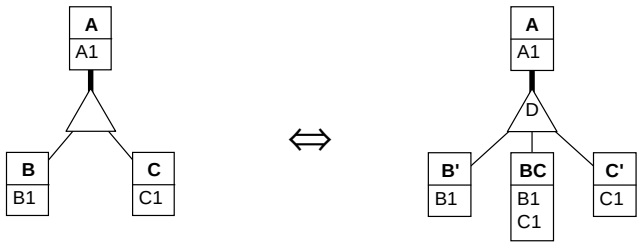


Figure A25.63 - Transformation de disjonction symétrique de deux sous-types

La transformation asymétrique préserve un des sous-types B ou C, qui est repris dans le schéma cible, tandis qu'elle décompose l'autre (figure A25.64). Le sous-type préservé admet le sous-type BC du schéma précédent.

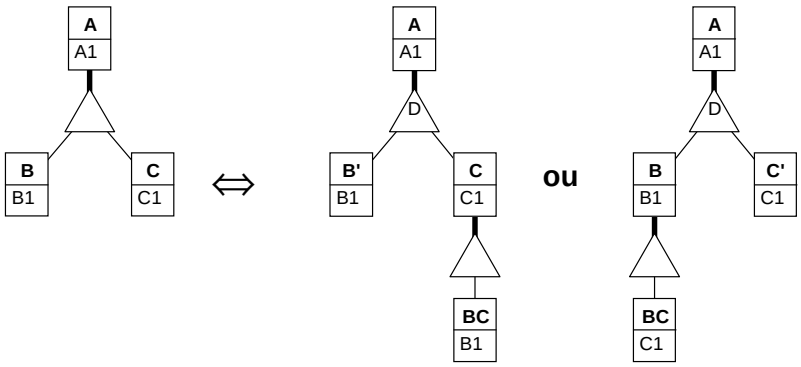


Figure A25.64 - Transformation de disjonction asymétrique de deux sous-types

Les propriétés des populations du schéma cible sont les suivantes :

B' = B - C	**ou**	C' = C - B
BC = B∩C		BC = B∩C

b) Traitement des sous-types communs aux deux sous-types

Lorsque les deux sous-types B et C du schéma source possèdent en commun un ou plusieurs sous-types (D et/ou E par exemple), et que ces sous-types communs n'appartiennent qu'à B et C, ils sont transférés collectivement vers le type d'entités d'intersection BC (figures A25.65 pour la transformation symétrique et A25.66 pour les transformations asymétriques).

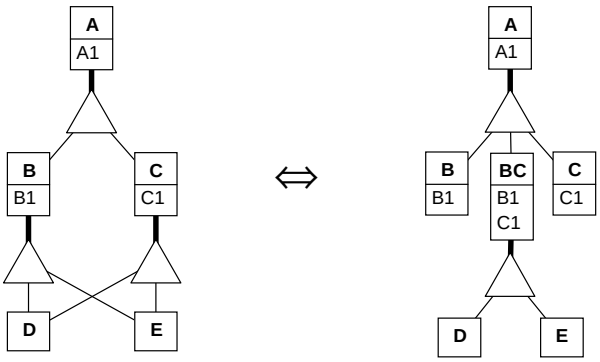


Figure A25.65 - Transformation de disjonction - Traitement des sous-types communs aux deux sous-types (cas symétrique)

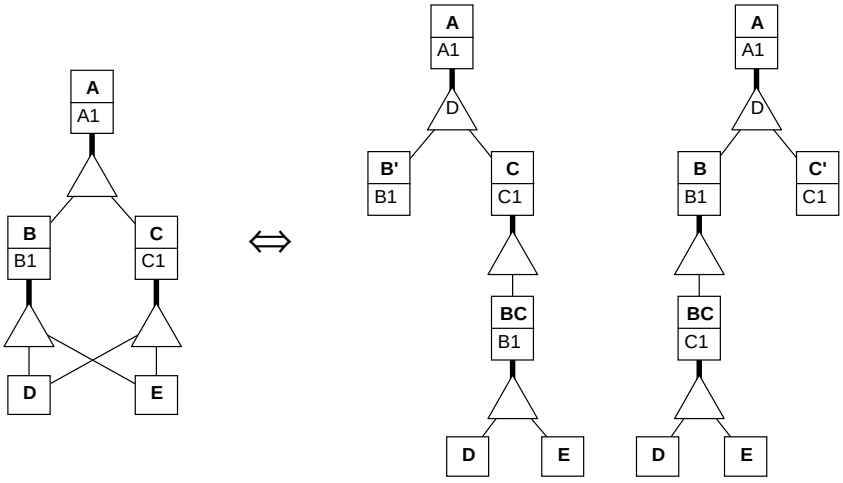


Figure A25.66 - Transformation de disjonction - Traitement des sous-types communs aux deux sous-types (cas asymétrique)

c) Traitement des sous-types d'un des sous-types

La propagation des sous-types (E et D, figure A25.67) d'un seul des sous-types (B par exemple) dans la transformation de disjonction est compliquée par le fait que leur surtype (B) est éclaté en deux types disjoints B' et BC.

Un examen superficiel pourrait conduire à une suggestion simple : puisque qu'il existe une transformation asymétrique qui préserve B, il suffit de déclarer B surtype de E et D dans le schéma transformé (schéma de droite de la A25.64). Malheureusement, E et D y seraient en conflit avec l'autre sous-type BC de B, avec lequel il ne sont pas disjoints. On obtiendrait ainsi un schéma non conforme et complexe, particulièrement lorsque E et D sont disjoints.

La figure A25.67 suggère une autre technique de résolution, par dédoublement de la hiérarchie issue de B entre B' et BC :

E = B'E ∪ BCE

D = B'D ∪ BCD

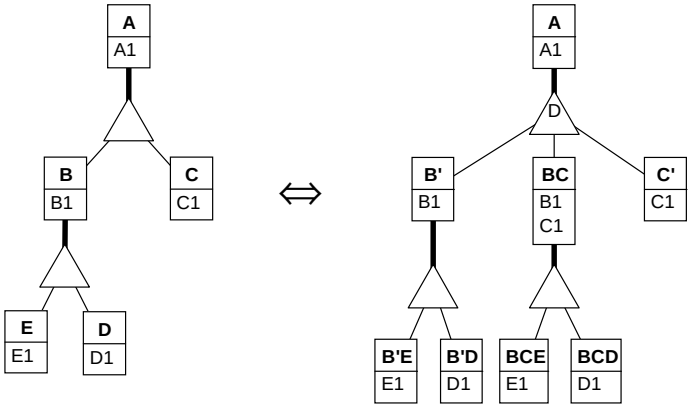


Figure A25.67 - Transformation de disjonction - Traitement des sous-types d'un des sous-types

d) Transformation des autres composants des sous-types

Dans les transformations qui précèdent, on n'a étudié que la propagation des attributs des sous-types transformés. Il faudrait de la même manière analyser la propagation des rôles et les contraintes du schéma source. La section G.7 abordent certaines de ces questions.

e) Disjonction de plus de deux sous-types

La figure A25.68 montre l'extension de la disjonction symétrique à une collection de trois sous-types. En toute généralité, étant donné **n** sous-types non-disjoints, la

transformation produit 2^n - 1 sous-types disjoints, structure qui devient pratiquement illisible et ingérable au-delà de 3.

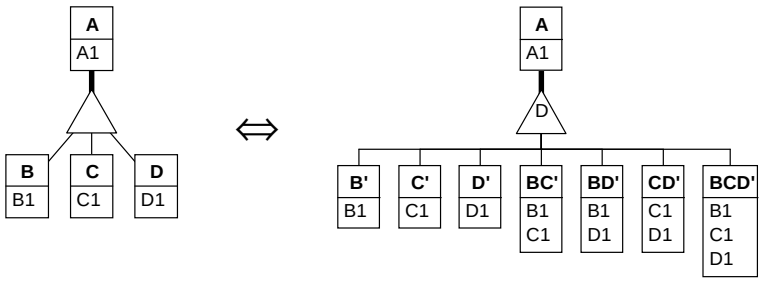


Figure A25.68 - Disjonction de plus de deux sous-types

Il est possible d'améliorer la présentation de cet ensemble par une hiérarchisation suggérée par l'observation de la répartition des entités illustrée par la figure A25.69.

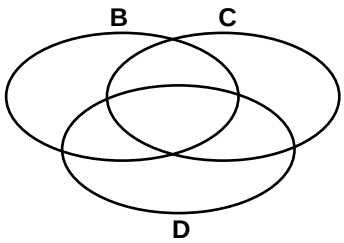


Figure A25.69 - Distribution des entités B, C et D

En regroupant les entités qui appartiennent à plus d'un sous-type dans un type de nom BCD, on obtient le schéma de la figure A25.70. cette approche est généralisable à plus de trois sous-types mais le résultats est de plus en plus complexe puisque chaque surtype possède **n** + 1 sous-types et que le nombre total de sous-types feuilles reste 2^n - 1.

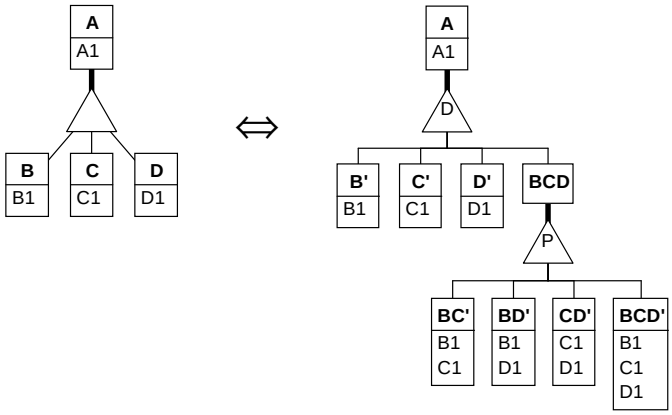


Figure A25.70 - Restructuration de l'ensemble des sous-types disjoints

f) Disjonction dans les hiérarchies multiples

Dans le cas général, les sous-types à disjoindre ont eux-même des sous-types, en partie communs et en partie en propres. Dans l'exemple de la figure A25.71, schéma de gauche, les sous-types à disjoindre A1 et A2 ont des sous-types en commun (C) et des sous-types en propre (B pour A1 et D pour A2). On fait l'hypothèse que les sous-types de A1 sont disjoints et qu'il en est de même des sous-types de A2. En revanche, en toute généralité, B et D ne sont pas disjoints. La figure A25.71 permet d'identifier les relations entre les populations de ces types d'entités dans le cas général.

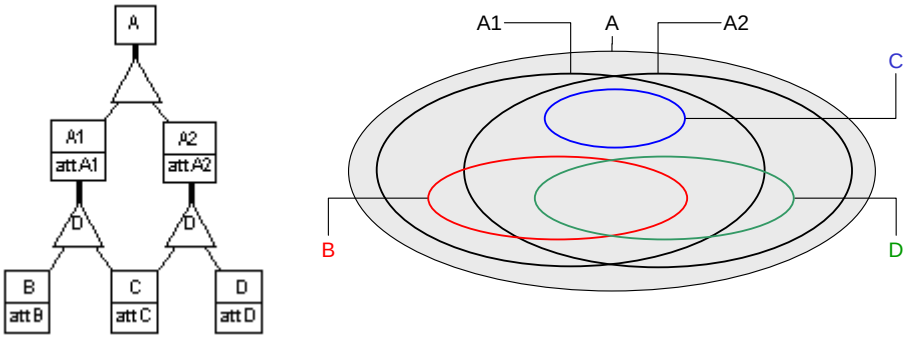


Figure A25.71 - Relations entre les populations des types d'entités A1, A2, B, C et D

On applique d'abord une transformation de disjonction symétrique à A1 et A2, conduisant à A1', A12 et A2' :

A1' = A1 - A2
A12 = A1 ∩ A2
A2' = A2 - A1

Ensuite, on répartit les sous-types B, C et D en types d'entités disjoints. C est bien entendu préservé comme sous-type de A12, comme nous l'avons observé plus haut, mais B et D doivent être répartis entre BA1 sous-type de A1', BA12D sous-type de A12 et DA2 sous-type de A2' (figure A25.72) :

BA1 = B - A2
BA12D = (B ∪ D) ∩ A12
DA2 = D - A1

Les entités de BA12D regroupent les entités du type A12 qui sont aussi des types B et/ou D, ce qui justifie le caractère facultatif de ses attributs et la contrainte at-least-1. Il reste à disjoindre les sous-ensembles de B et D au sein de BA12D, sous la forme des sous-types B12D, BD et D12B de BA12D :

B12D = BA12D - D
BD = B ∩ D

D12B = BA12D - B

Le schéma de droite de la figure A25.73 montre le résultat final. Il est possible de réduire le schéma d'un niveau en remplaçant BA12D par ses sous-types. Une autre approche sera développée à la section A25.9.9.

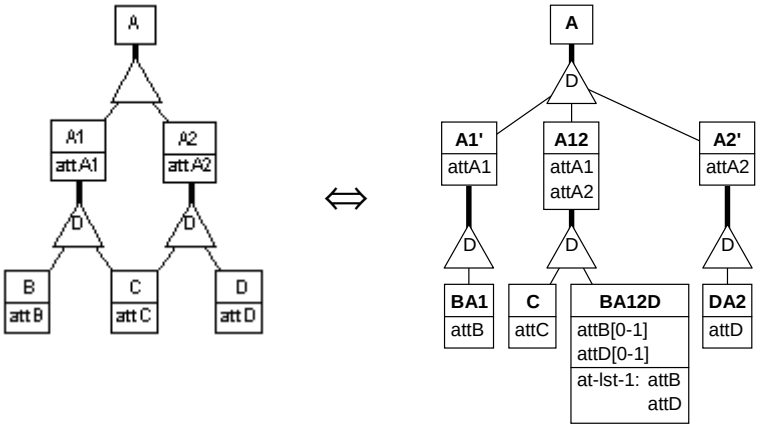


Figure A25.72 - Distribution des sous-types B, C et D lors de la disjonction de A1 et A2 (stade intermédiaire)

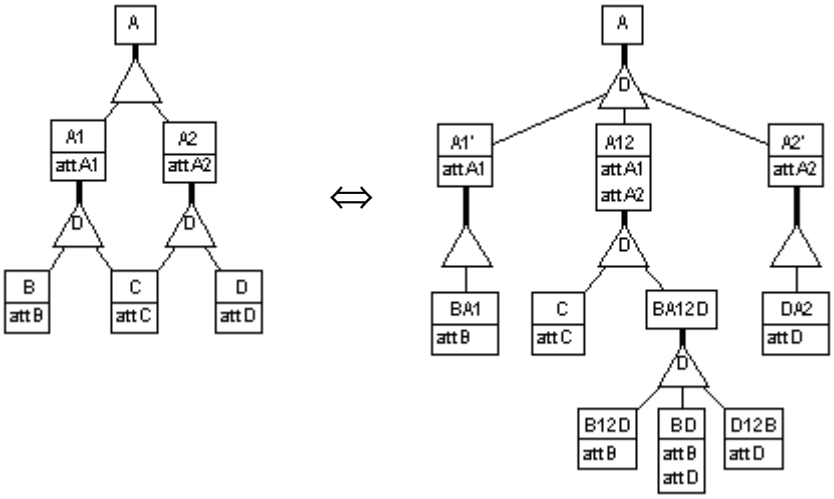


Figure A25.73 - Distribution des sous-types B, C et D lors de la disjonction de A1 et A2 stade final)

A25.9.6Couverture d'un surtype

Lorsque l'ensemble des sous-types n'est pas soumis à une contrainte de totalité, il est possible que des entités du surtype n'appartiennent à aucun sous-type. On peut

demander de compléter cet ensemble de manière à imposer cette contrainte. On fait ainsi apparaître un nouveau sous-type A' sans composants propres (figure A25.74). Attention cependant à l'asymétrie des sous-types en cas de non-disjonction (schéma supérieurs) : B et C restent non disjoints mais A' est disjoint de B et C, ce qui correspond à un schéma non standard (*disjonction partielle*), non conforme au modèle Entité-association présenté dans cet ouvrage. Il sera préférable d'appliquer préalablement une transformation de disjonction à B et C. Lorsque les sous-types sources sont disjoints, le résultat est une partition (schémas inférieurs).

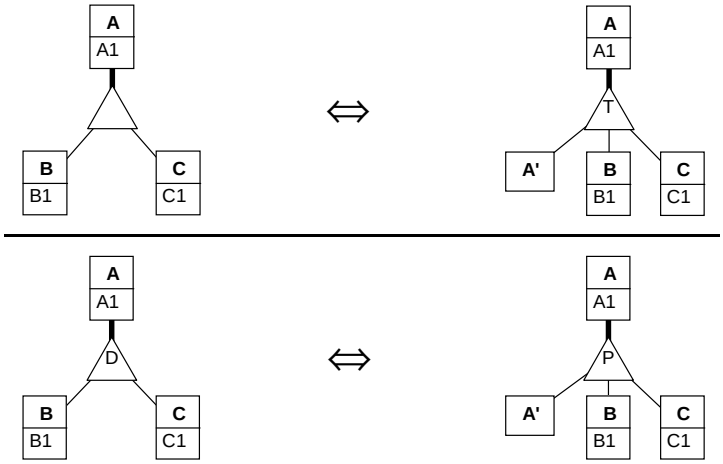


Figure A25.74 - Couverture d'un surtype

A25.9.7Partitionnement d'un surtype

Cette transformation est la composition de la disjonction des sous-types et de la couverture du surtype. Elle transforme en partition un ensemble quelconque de sous-types relatifs à un même surtype. Il existe plusieurs schémas cibles selon qu'on a utilisé une disjonction symétrique (figure A25.75) ou asymétrique. Comme nous l'avons observé précédemment, cette transformation n'est guère praticable pour des structures de plus de 3 sous-types, à moins de hiérarchiser l'ensemble de ces sous-types (figure A25.70).

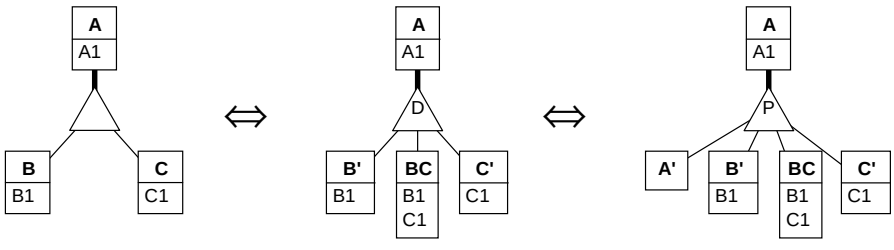


Figure A25.75 - Partitionnement d'un surtype

A25.9.8Transformation d'une répartition multiple

Une répartition multiple d'un surtype offre plus d'une manière de classer les entités de ce surtype (figure A25.76/gauche). Il n'existe pas de transformation de répartition multiple en une répartition unique qui soit totalement satisfaisante. On propose quatre techniques permettant de gérer **n** répartitions (**n** > 1) : le *dédoublement du surtype*, le *clonage du surtype*, la *hiérarchisation des répartitions* et le *croisement des répartitions*. Nous décrirons ces transformation pour deux répartitions. Elles se généralisent sans difficulté à plus de deux répartitions.

Le **dédoublement du surtype** (figure A25.76) consiste à attacher en bijection au surtype **n**-1 types d'entités sans propriétés (ni attribut, ni rôle, ni contrainte). Le surtype ainsi que ses doublures reçoivent chacun une répartition. Les mécanismes d'héritage sont propres à cette structure et doivent donc être, partiellement du moins, gérés de manière procédurale (l'héritage de A vers D et E doit être simulé). Une variante consiste à dupliquer les propriétés de A dans A'. Dans ce cas, les mécanismes d'héritage standard sont d'application mais c'est la redondance A/A' qu'il faut contrôler de manière procédurale.

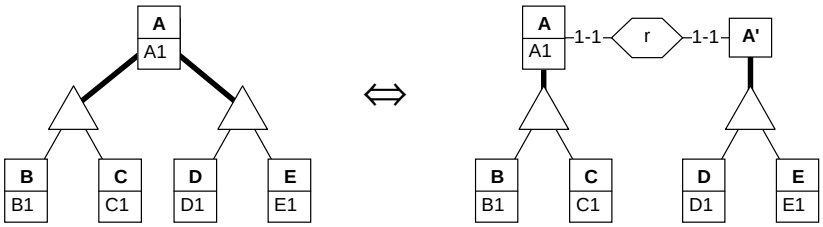


Figure A25.76 - Transformation d'une répartition multiple par dédoublement du surtype

Le **clonage du surtype** (figure A25.77) consiste à définir **n** sous-types du surtype source, à raison d'un par répartition. Chaque clone est à son tour surtype d'une des répartitions. Chaque sous-type clone possède la même population que le surtype : A' = A" = A. Cette propriété étant inconnue dans le modèle Entité-association, il faudra la gérer de manière procédurale.

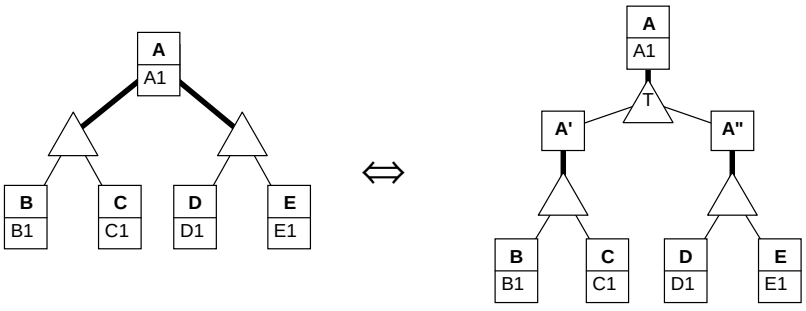


Figure A25.77 - Transformation d'une répartition multiple par clonage du surtype

La transformation par **hiérarchisation** (figure A25.78) consiste à subordonner une répartition à l'autre. Une des répartitions (idéalement une partition) est conservée alors que la seconde est reproduite sous chacun des sous-types de la première. On procédera au préalable, si nécessaire, à la transformation en une partition de la première répartition. Par exemple, le nouveau sous-type BD représente les entités appartenant à la fois aux types B et D du schéma source. La transformation par hiérarchisation préserve une des répartitions mais duplique l'autre. Le schéma final implique des redondances dans les composants des sous-types dupliqués : les propriétés de D sont dupliquées dans BD et CD. Il n'existe cependant pas de redondance au niveau des instances.

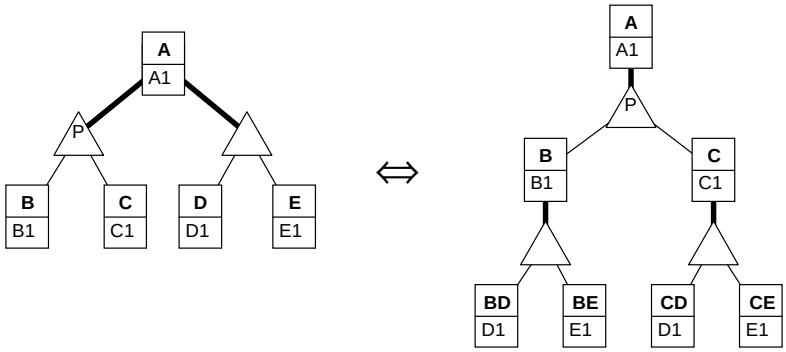


Figure A25.78 - Transformation d'une répartition multiple par hiérarchisation des répartitions

La transformation par **croisement des répartitions** (figure A25.79) remplace les répartitions multiples par une seule répartition dont les sous-types sont obtenus par le produit cartésien des **n** ensembles de sous-types. Le nouveau sous-type BD représente les entités appartenant à la fois aux types B et D du schéma source. Cette transformation exige que chaque répartition soit une partition (il est possible de supprimer cette contrainte mais au prix d'un grande complexité de schéma résultant). On procédera donc au préalable, si nécessaire, à la transformation de chaque ensemble de sous-types en une partition. Le schéma final implique des redondances dans les composants des sous-types : les propriétés de B sont dupliquées dans BD et BA25. Il n'existe cependant pas de redondance au niveau des instances. Cette transformation n'est envisageable que si le fragment de schéma source ne comprend qu'un petit nombre de sous-types. On observera que cet opérateur n'est pas primitif. Il s'obtient par application de la transformation des relations *is-a* par héritage descendant (section A25.9.2) aux sous-types BD, BE, CD et CE de la figure A25.78.

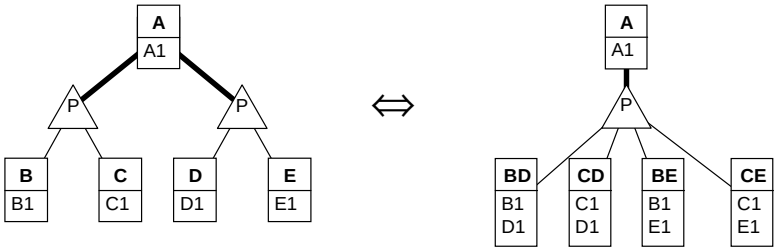


Figure A25.79 - Transformation d'une répartition multiple par croisement des répartitions

A25.9.9Transformation d'une hiérarchie à surtypes multiples

L'élimination du pattern de surtypes multiples va généralement de pair avec la disjonction des sous-types. En effet, ce pattern n'est possible qu'en présence de sous-types non disjoints. Nous décrirons ci-dessous deux techniques : la disjonction des surtypes et leur fusion.

a) Disjonction des sous-types

La technique de disjonction de sous-types dans le contexte d'une hiérarchie complexe a été détaillée à la section 9.5, alinea *f*. Elle conduit à éliminer la structure de surtypes multiples et peut donc être utilisée dans ce but.

b) Fusion des surtypes

Les surtypes multiples (A1 et A2 dans la figure A25.80) dépendent d'un surtype commun (ici A) autorisant leur recouvrement (non disjonction). On suppose que A est le surtype direct de A1 et A2. Si tel n'est pas le cas, on fusionnera également les parents intermédiaires.

La technique de fusion consiste à remplacer les surtypes par un unique type d'entités A1A2 regroupant les entités des types A1 et A2 (figure A25.80, schéma de droite) :

$A1A2 = A1 \cup A2$

A1A2 admet deux sous-types disjoints, C et BD, lequel regroupe les entités des types B et D ($BD = B \cup D$). B et D étant non disjoints, on leur appliquera si nécessaire une transformation de disjonction. Pour simplifier l'expression des contraintes, nous ajoutons à A1A2 un attribut indicateur pour chaque sous-type, soit A1 et A2.

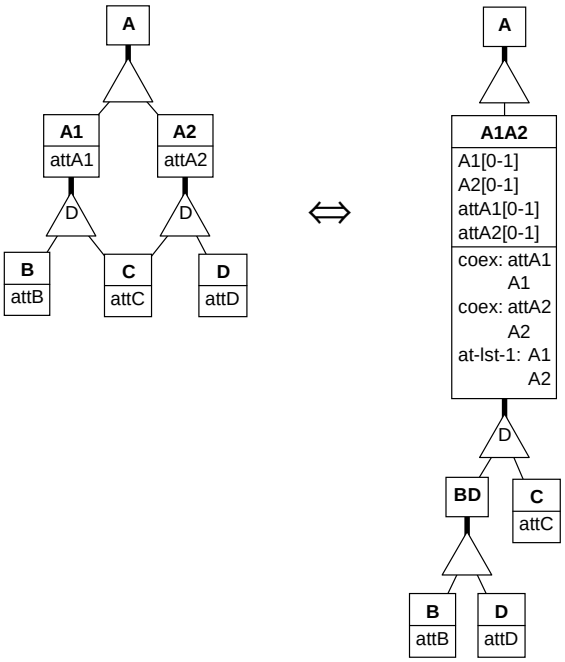


Figure A25.80 - Réduction de surtypes multiples par fusion

A25.9.10 Autres transformations

Les hiérarchies *is-a* constituent des structures riches et expressives. Elles font l'objet d'autres transformations que celles qui ont été présentées dans cette section. Certaines préservent la sémantique tandis que d'autres l'enrichissent en modifiant l'interprétation des structures. Citons quelques exemples :

- insertion d'un type d'entités dans une hiérarchie
- élimination d'un type intermédiaire (rattachement des *enfants* à leurs *grands parents*)
- introduction d'un type ou plusieurs types intermédiaire(s) entre un *parent* et ses *enfants*
- regroupement de deux hiérarchies indépendantes
- élimination d'un sous-type terminal sans composants dans une partition
- transformation de sous-types terminaux sans composants par un attribut indicateur de sous-type (et inverse).

A25.10TRANSFORMATION DE TYPES D'ASSOCIATIONS

Un type d'associations peut être transformé en attributs (formant une clé étrangère), en type d'entités ou encore être décomposé en plusieurs types d'associations plus simples. Nous étudierons successivement :

1. les transformations d'un TA binaire en clé étrangère
2. les transformations d'un TA en TE
3. la transformation de TA 1:1 en relations *is-a*
4. les transformation d'un rôle multitypes
5. la transformation d'un TA binaire en attribut objet
6. la transformation d'un attribut (objet) en un rôle supplémentaire
7. la réduction du degré d'un TA n-aire
8. les décompositions d'un TA n-aire

A25.10.1Transformation d'un TA binaire en clé étrangère

On détaille ci-dessous les principaux cas de figure de la transformation de mutation d'un *type d'associations en attributs*, lesquels forment une clé étrangère. On utilisera principalement ces transformations dans les processus de conception logique. On notera que cette famille de transformation peut produire des clés étrangères multivaluées. On étudiera successivement la transformation des types d'associations de classes fonctionnelles 1:1, N:1, 1:N et N:N. Le type d'entités cible de la clé étrangère produite doit disposer d'un identifiant (primaire de préférence) constitué exclusivement d'attributs. Il sera dans certains cas nécessaire d'assigner un identifiant technique à ce type d'entités.
Il aurait été possible de simplifier cette étude par l'usage de transformations génériques, mais, en raison de l'importance de ces transformations, nous avons préféré en présenter le détail.

a) Types d'associations 1:1

Un type d'associations 1:1 se transforme en une *clé étrangère mono-valuée identifiante* (figure A25.81). Cette clé est obligatoire ou facultative, simple ou totale.

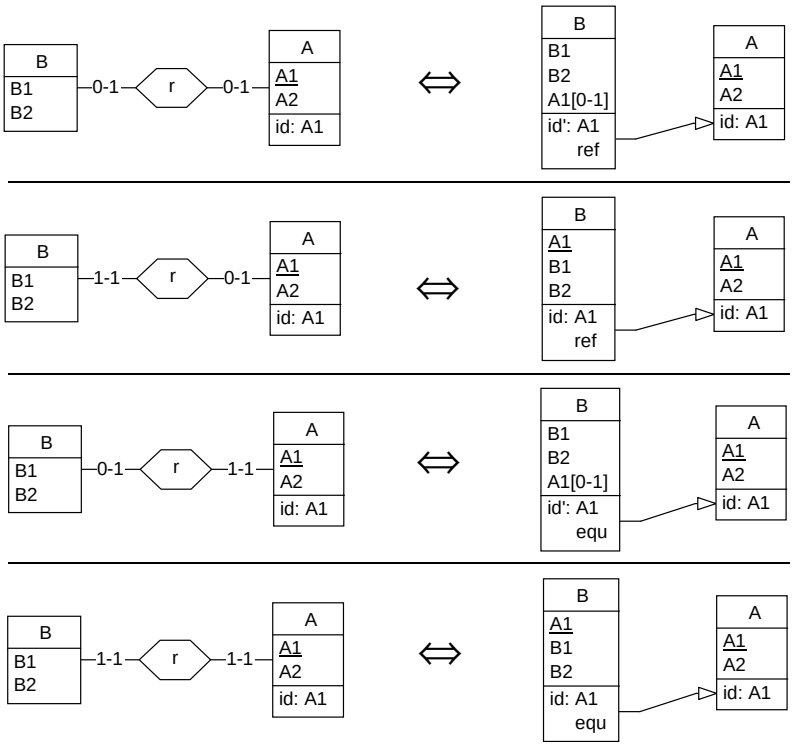


Figure A25.81 - Transformation d'un type d'associations 1:1

b) Types d'associations N:1

Un type d'associations N:1 se transforme en une *clé étrangère mono-valuée non identifiante* (figure A25.82). Cette clé est obligatoire ou facultative, simple ou totale.

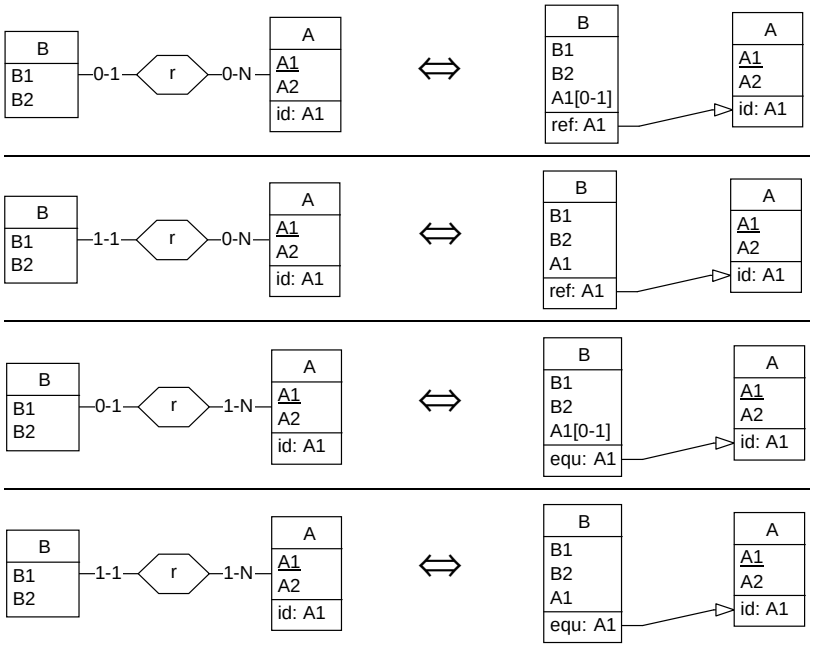
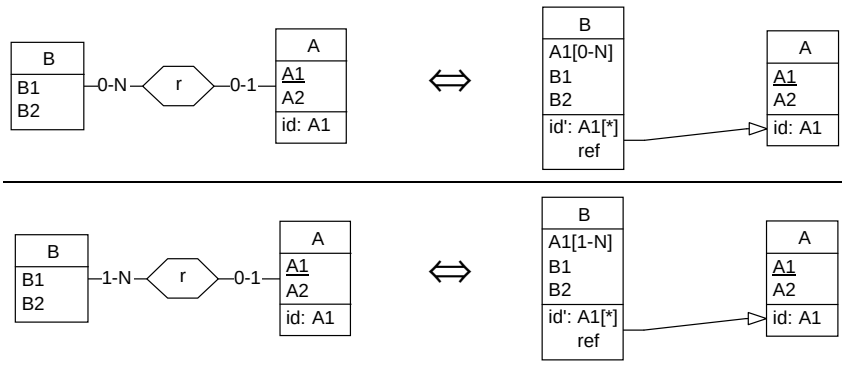


Figure A25.82 - Transformation d'un type d'associations N:1

c) Types d'associations 1:N

Un type d'associations 1:N se transforme en une *clé étrangère multivaluée identifiante* (figure A25.83). Cette clé est obligatoire ou facultative, simple ou totale.

Si l'identifiant de A est constitué de plusieurs attributs, les composants de la clé étrangère sont regroupés sous la forme d'un attribut composé. La cardinalité maximum du rôle joué par B, notée **N**, peut être toute valeur supérieure à 1.



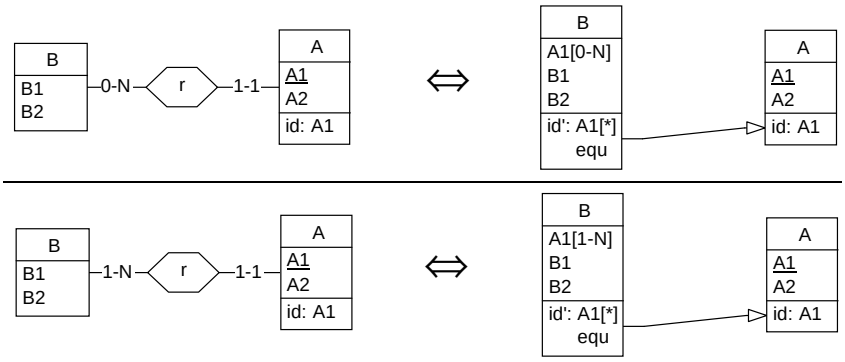


Figure A25.83 - Transformation d'un type d'associations 1:N

d) Types d'associations N:N

Un type d'associations N:N se transforme en une *clé étrangère multivaluée non identifiante* (figure A25.84). Cette clé est obligatoire ou facultative, simple ou totale.

Si l'identifiant de A est constitué de plusieurs attributs, les composants de la clé étrangère sont regroupés sous la forme d'un attribut composé. La cardinalité maximum du rôle joué par B, notée **N**, peut être toute valeur supérieure à 1.

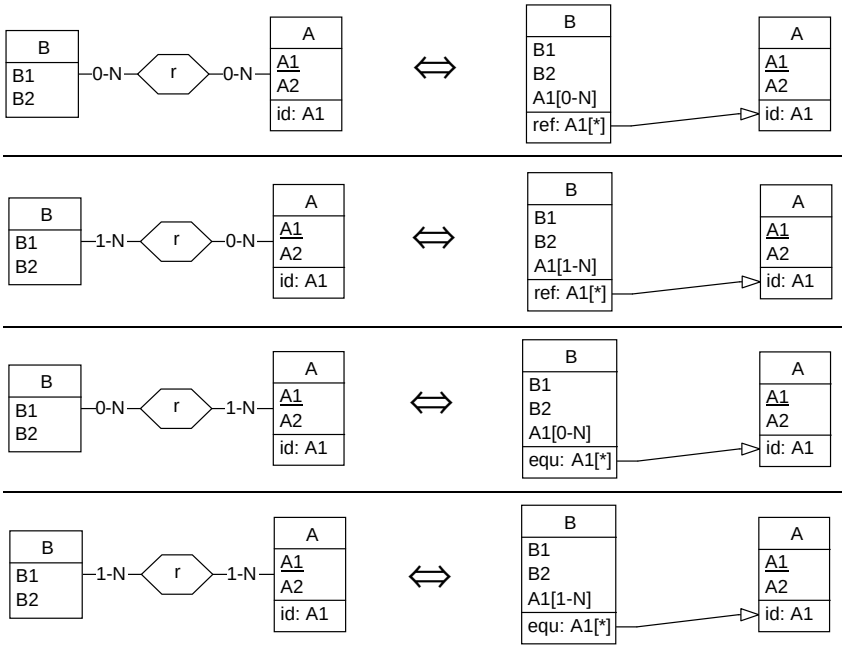


Figure A25.84 - Transformation d'un type d'associations N:N

A25.10.2Transformation d'un TA en TE

Cette famille d'opérateurs réalise une mutation du type d'associations en un type d'entités. Elle admet deux variantes. La première, la plus classique, représente chaque association par une entité. La seconde, moins connue, exploite une forme particulière de décomposition d'un type d'associations par réduction de son degré. Elle sera décrite à la section A25.10.9.

La figure A25.85 illustre la mutation classique. Elle montre la propagation des cardinalités, des attributs et des identifiants.

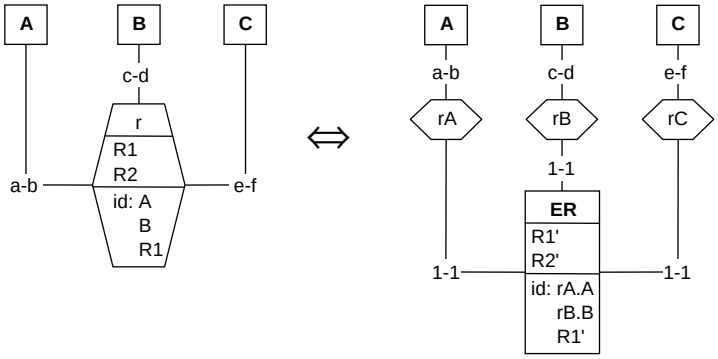


Figure A25.85 - Transformation standard d'un type d'association en TE

A25.10.3Transformation de TA 1:1 en relations *is-a*

Cet opérateur est l'inverse de la transformation de relations *is-a* par matérialisation (section A25.9.1). Des préconditions existent sur les contraintes de cardinalités des types d'associations et sur les contraintes d'existence. La figure A25.86 montre un exemple de transformation conduisant à une partition des sous-types.

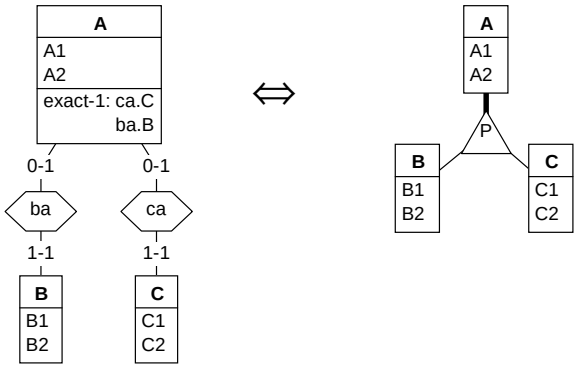


Figure A25.86 - Transformation de types d'associations 1:1 en relations *is-a*

A25.10.4Remarque sur la transformation de TA 1:1

Nous avons rencontré plusieurs transformations applicables à des types d'associations 1:1. Il est utile de rappeler que le choix d'une transformation est souvent guidé par des considérations sémantiques. A titre d'exemple, un type d'associations 1:1 entre les types d'entités :

- VEHICULE et UTILITAIRE se traduira probablement par une relation *is-a*,
- PATIENT et STATISTIQUES pourra se traduire par une fusion bijective,
- DELEGUE et VEHICULE sera généralement conservé tel quel.

A25.10.5Transformation d'un rôle multitypes

Un rôle multitypes est joué non par un type d'entités, comme un rôle ordinaire, mais par plusieurs types d'entités (figure A25.87/gauche). La transformation la plus évidente consiste à partitionner les associations r selon le type d'entités du rôle multitypes. On produit ainsi autant de types d'associations simples que le rôle couvre de types d'entités (schéma central). Si nécessaire, on peut ensuite appliquer une transformation de factorisation (section A25.8.5), ce qui donne le schéma de droite.

Si le rôle r.A du schéma source est de cardinalité [1-b], avec b > 1, les rôles r1.A et r2.A du schéma central sont tous de cardinalité [0-b]. Si le rôle r.A du schéma source est de cardinalité [0-b], avec b ≥ 1, la contrainte d'existence du schéma central tombe et les rôles r1.A et r2.A du schéma central sont de cardinalité [0-b]. Les rôles r.A et r'.A sont de même cardinalité. La cardinalité du rôle r.b est transmise sans modification aux rôles r1.B1, r2.B2 et r'.B12.

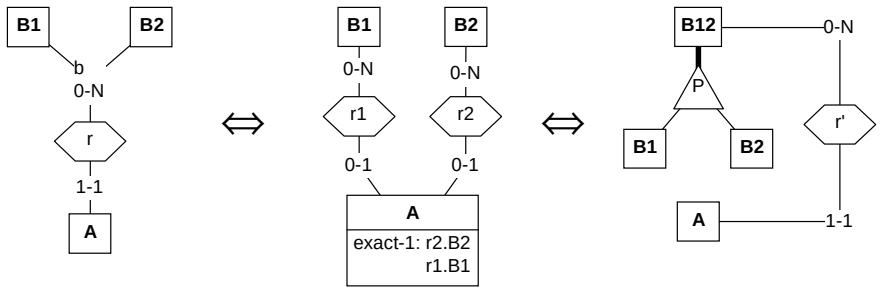


Figure A25.87 - Transformation d'un rôle multitypes

A25.10.6Transformation d'un TA binaire en attribut objet

Cette transformation est l'inverse de la transformation A25.6.8. Elle sera plus particulièrement utilisée dans le développement de bases de données orientées objet. Les différentes variantes selon les cardinalités du type d'associations se traitent de la même manière que pour les clés étrangères (section A25.10.1).

A25.10.7Transformation d'un rôle en attribut objet

Il s'agit de l'inverse de la transformation A25.7.2. Elle permet de réduire (en apparence du moins) le degré d'un TA et est applicable à tout TA de degré supérieur à 2. On l'utilisera notamment lors de la traduction d'un schéma conceptuel dans un modèle orienté objet.

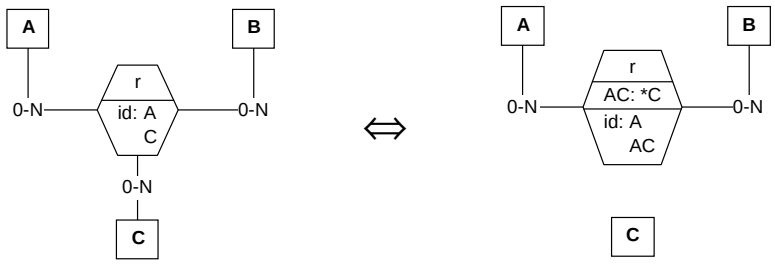


Figure A25.88 - Transformation d'un rôle en attribut objet

A25.10.8Transformation d'un rôle en clé étrangère

Il s'agit de l'inverse de la transformation A25.7.3. Elle permet de réduire (en apparence du moins) le degré d'un TA et est applicable à tout TA de degré supérieur à 2. Le concept de clé étrangère au départ d'un TA étant atypique, cette transformation doit être considérée comme anecdotique.

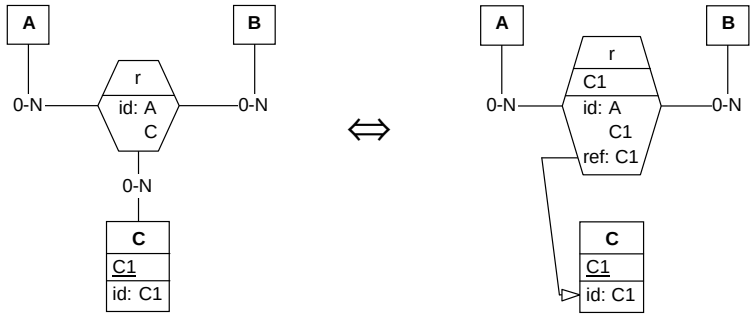


Figure A25.89 - Transformation d'un rôle en clé étrangère

A25.10.9Réduction du degré d'un TA n-aire

Il est possible de réduire un type d'associations r de degré **n** en

1. un type d'associations r' de degré **n** - **k** + 1,
2. **k** types d'associations fonctionnels,
3. un nouveau type d'entités,

k, tel que $1 \leq \mathbf{k} \leq \mathbf{n}$, est un paramètre de la transformation.

Cet opérateur, décrit plus en détail dans [Hainaut, 1991], regroupe plusieurs transformations classiques et moins classiques.

Son principe est assez intuitif. Considérons, dans le schéma de gauche de la figure A25.90, un sous-ensemble quelconque de **k** rôles de r. Soit par exemple {r.A, r.B} cet ensemble (**k** = 2). L'ensemble de tous les couples (a,b) de A x B présents dans les instances de r s'exprime par la projection r[A,B].

Représentons chacun de ces couples par une entité d'un nouveau type nommé AB. Nous pouvons dès lors substituer AB aux deux rôles r.A et r.B dans r, ce qui nous donne un nouveau type d'associations r' de degré n - k - 1, soit ici 3, entre AB, C et D.

Il nous faut encore définir les entités de AB comme représentant des couples de A x B, ce qu'on réalise par les **k** = 2 types d'associations fonctionnels rA et rB et par l'identifiant {rA.A, rB.B} de AB.

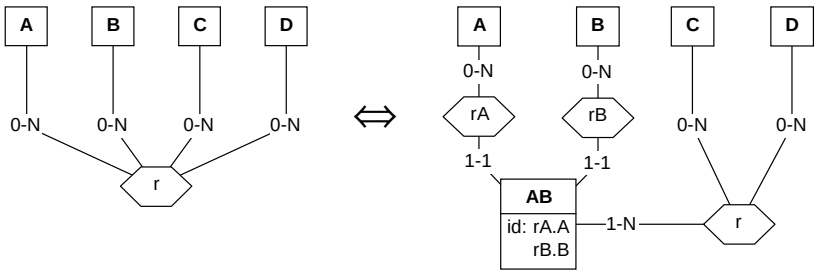


Figure A25.90 - Réduction du degré d'un TA (**k** = 2)

Lorsque **k** = 1, le nouveau type d'entités (soit A') représente toutes les entités actives du rôle sélectionné (r[A]). Dans ce cas limite, le degré de r est conservé (**n** - **1** + 1). Le type d'associations fonctionnel rA est 1:1 (il constitue un identifiant implicite de A'). Il peut ensuite, si nécessaire, être transformé en une relation *is-a* (figure A25.91)

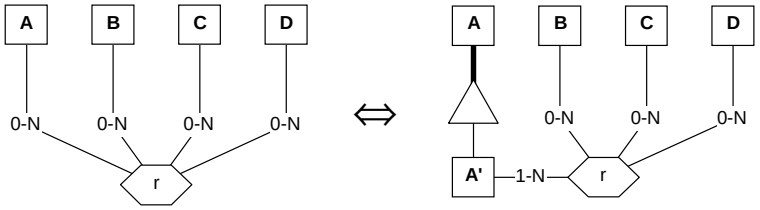


Figure A25.91 - Réduction du degré d'un TA (**k** = 1)

Considérons l'autre cas limite : **k** = **n**. Dans ce cas, le nouveau type d'entités ABCD représente toutes les instances de r (r[A,B,C,D]). On introduit **n** types d'associations fonctionnels rA, ..., rD dont les rôles opposés forment l'identifiant de ABCD. r' est ici inexistant et on retrouve la transformation classique de la section A25.10.2, rappelée à la figure A25.92.

En pratique, l'identifiant minimal du nouveau type d'entités dépend des identifiants de r.

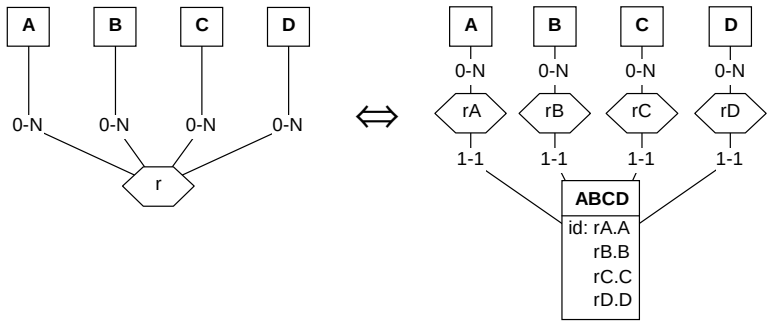


Figure A25.92 - Réduction du degré d'un TA (**k** = **n**)

A25.10.10Décomposition d'un TA n-aire

La décomposition d'un type d'associations remplace celui-ci par plusieurs types d'associations de degré inférieur. On considère deux opérateurs : la décomposition selon un rôle de cardinalité [i-1] et la décomposition de normalisation.

a) Décomposition selon un rôle de cardinalité [i-1]

Un type d'associations dont un rôle est de cardinalité [0-1] ou [1-1] peut être décomposé selon ce rôle. Celui-ci est un effet un déterminant de chacun des autres rôles. Cette transformation et sa justification ont été présentées à la section 15.18.9.

La figure A25.93 présente cette transformation selon les deux configurations de cardinalité du rôle r.B. Dans le cas d'un rôle de cardinalité [0-1], il est possible d'abandonner la contrainte de coexistence au profit d'une relation *is-a* (section A25.11.1).

b) Normalisation relationnelle d'un TA

On a montré (section 15.18) qu'un type d'associations pouvait être représenté par une relation constituée de la représentation de ses rôles et de ses attributs. Cette relation peut être le siège de dépendances anormales, auquel cas il convient de la normaliser par décomposition.

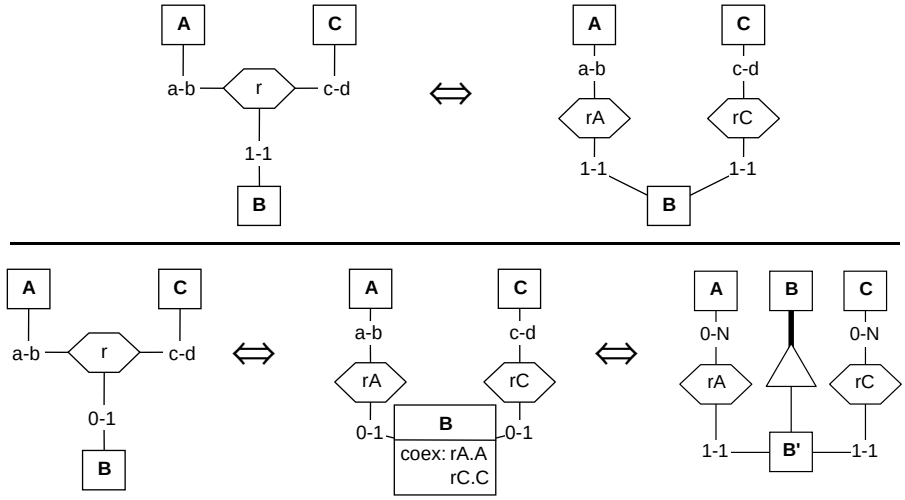


Figure A25.93 - Décomposition selon un rôle de cardinalité [i-1]

Le cas le plus simple est celui d'un type d'associations n-aire sans attributs, dont la normalisation a été discutée à la section 17.7.8. Il est repris à la figure A25.94. Lorsque le rôle déterminant de la dépendance anormale (r.B) est facultatif, le schéma cible comporte une contrainte de coexistence, qui peut ensuite être éliminée, par exemple par la définition d'un sous-type (B') selon les transformations de la section A25.11.1.

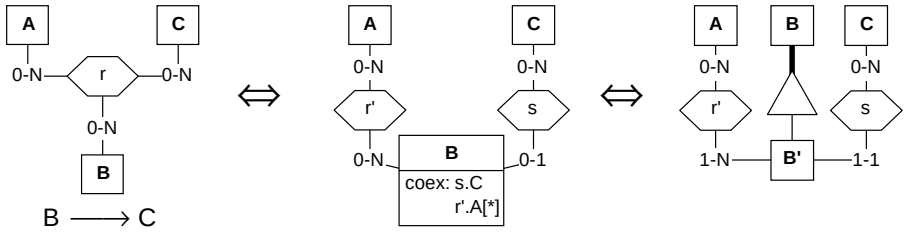


Figure A25.94 - Normalisation relationnelle d'un TA

La présence d'attributs ne pose pas de problème particulier, la procédure de normalisation relationnelle s'appliquant telle quelle (figure A25.95). Cependant, les configurations de dépendances fonctionnelles dans lesquelles le déterminant est exclusivement constitué d'attributs ne sont pas valide.

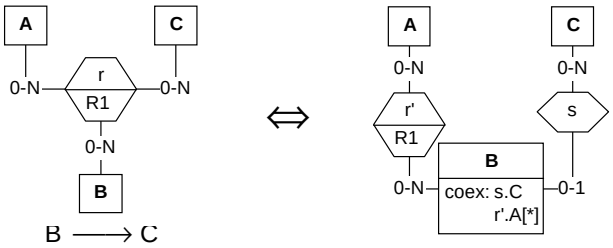


Figure A25.95 - Normalisation relationnelle d'un TA en présence d'attributs

A25.11TRANSFORMATIONS RELATIVES AUX CONTRAINTES

Ces transformations ont pour effet de remplacer une contrainte par une structure équivalente. On les utilise en particulier dans le processus de normalisation et en rétro-ingénierie.

A25.11.1Contrainte de coexistence

Un groupe d'attributs soumis à une contrainte de coexistence peut faire l'objet d'une transformation d'agrégation en un attribut composé facultatif. Ce dernier peut, si nécessaire, être transformé en un type d'entités, lequel peut ensuite, selon la sémantique qui lui est assignée, être déclaré sous-type du type d'entités source (figure A25.96).

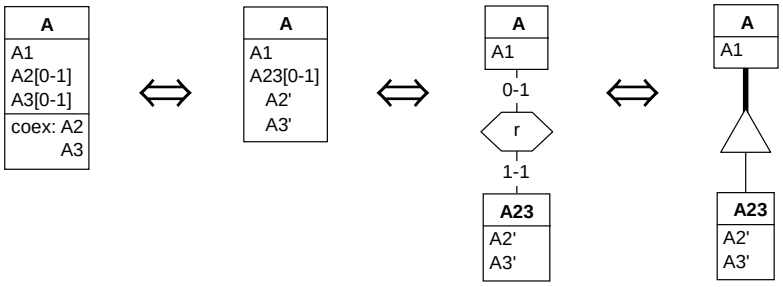


Figure A25.96 - Transformations successives des composants d'une contrainte de coexistence

Ces transformations restent d'application lorsque la contrainte implique un ou plusieurs rôles (figure A25.97).

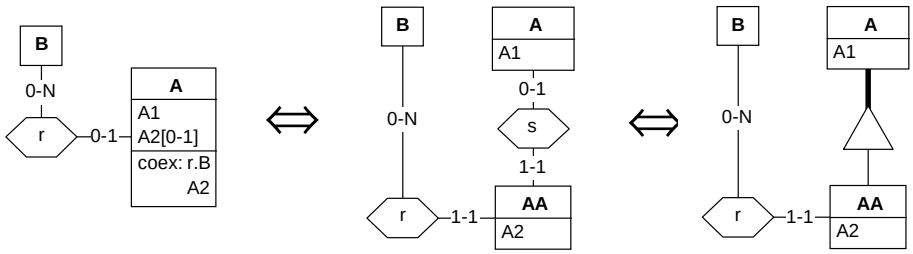


Figure A25.97 - Transformations successives des composants d'une contrainte de coexistence impliquant des rôles

A25.11.2Contrainte d'implication

Le traitement d'une contrainte d'implication (if) est similaire à celui de la contrainte de coexistence (figure A25.98). Il en est de même des contraintes impliquant des rôles.

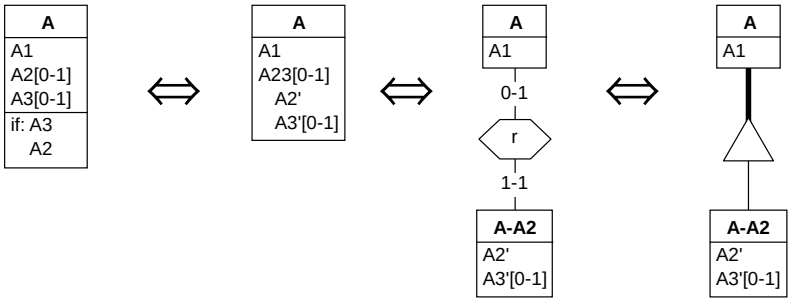


Figure A25.98 - Transformations successives des composants d'une contrainte d'implication

A25.12POUR EN SAVOIR PLUS

Le concept de transformation, par ailleurs très ancien [Navathe, 1980], fait partie intégrante du domaine des bases de données. En effet, depuis près de 40 ans, la plupart des processus d'ingénierie de bases de données, et en particulier la normalisation conceptuelle, la conception logique, l'optimisation et la rétro-ingénierie, ont été modélisés comme des chaînes de transformations. Les techniques transformationnelles constituent le moteur des approches dites *dirigées par les modèles*, ou MDE, selon la terminologie de l'OMG. L'idée de spécifier de manière rigoureuse un composant d'un système informatique dans un langage abstrait, puis de transformer cette spécification par des règles de conversion vers du code exécutable est le rêve de l'informaticien depuis la nuit des temps (estimée au début des années 60 pour cette question).

L'approche suivie dans cette annexe se base sur un ensemble de travaux dont les références sont reprises dans la section A25.13 et dont la synthèse a été publiée dans [Hainaut, 2006].

A25.13RÉFÉRENCES ET BIBLIOGRAPHIE

Alves, T.L., Silva, P.F., Visser, J., Oliveira, J.N., Strategic Term Rewriting and Its Application to a Vdm-SL to SQL Conversion, in *Proc. FM 2005*, LNCS, No 3582, Springer-Verlag. (2005) 399-414

Baader, F., Horrocks, I., and Sattler, U. Description logics. In Staab, S. and Studer, R. (Ed.), *Handbook on Ontologies*, International Handbooks on Information Systems, pages 3-28. Springer, (2004).

Balzer, R. Transformational implementation : An example. *IEEE TSE*, Vol. SE-7(1). (1981)

Batini, C., Ceri, S., & Navathe, S., B. *Conceptual Database Design*, Benjamin/Cummings. (1992)

Batini, C., Di Battista, G., Santucci, G. Structuring Primitives for a Dictionary of Entity Relationship Data Schemas, *IEEE TSE*, Vol. 19, No. 4. (1993)

Bolois, G., & Robillard, P. Transformations in Reengineering Techniques. *Proc. of the 4th Reengineering Forum "Reengineering in Practice"*, Victoria, Canada. (1994)

Boyd, M., McBrien. Towards a Semi-Automated Approach to Intermodel Transformation, In *Proceedings of EMMSAD'04,Volume 1, CAiSE Workshop Proceedings*, Riga Technical University. (2004) 175-188

Casanova, M., A., Amaral De Sa. Mapping uninterpreted Schemes into Entity-Relationship diagrams : two applications to conceptual schema design. *IBM J. Res. & Develop.*, 28(1). (1984)

Clève, A., Henrard, J., Hainaut, J-L. Co-transformations in Information System Reengineering, in *Proc. of WCRE'04/ATEM-04*, (2004)

Darwen, H., Date, C., J. Relation-valued Attributes, in Date, C., J., Darwen, H., *Relational Database Writings* 1989-1991, Addison-Wesley (1993)

D'Atri, A., & Sacca, D. Equivalence and Mapping of Database Schemes, *Proc. 10th VLDB conf.*, Singapore. (1984)

Estiévenart, F., François, A., Henrard, J., Hainaut, J-L. Web Site Engineering. *Proc. of the 5th International Workshop on Web Site Evolution*, Amsterdam, Sept. 2003, IEEE CS Press. (2003)

Fagin, R. Multivalued dependencies and a new normal form for relational databases, *ACM TODS*, 2(3). (1977)

Fikas, S., F. Automating the transformational development of software, *IEEE TSE*, Vol. SE-11. (1985)

Hainaut, J-L. Theoretical and practical tools for database design, in *Proc. of the Very Large Databases Conf.*, pp. 216-224, September, IEEE Computer Society Press. (1981)

Hainaut, J.-L. A Generic Entity-Relationship Model. *Proc. of the IFIP WG 8.1 Conf. on Information System Concepts: an in-depth analysis*, North-Holland. (1989)

Hainaut, J-L. Entity-generating Schema Transformations for Entity-Relationship Models, in *Proc. of the 10th Entity-Relationship Approach*, San Mateo (CA), 1991, North-Holland. (1992)

Hainaut, J-L., Chandelon M., Tonneau C., & Joris M. (1993). Contribution to a Theory of Database Reverse Engineering. *Proc. of the IEEE Working Conf. on Reverse Engineering*, Baltimore, May 1993, IEEE Computer Society Press.

Hainaut, J-L, Chandelon M., Tonneau C., Joris M. Transformational techniques for database reverse engineering. *Proc. of the 12th Int. Conf. on ER Approach*, Arlington-Dallas, ER Institute (and LNCS Springer-Verlag in 1994). (1993)

Hainaut, J-L. *Transformation-based database engineering*. Tutorial notes, *VLDB'95*, Zürich, Switzerland, (1995) (available at http://www.info.fundp.ac.be/libd).

Hainaut, J-L. Specification preservation in schema transformations - application to semantics and statistics, *Data & Knowledge Engineering*, 11(1). (1996)

Hainaut, J-L., Henrard, J., Hick, J-M., Roland, D., Englebert, V. Database Design Recovery, in *Proc. of the 8th Conf. on Advanced Information Systems Engineering* (CAiSE²96), Springer-Verlag (1996)

Hainaut, J.-L., Hick, J.-M., Englebert, V., Henrard, J., Roland, D. Understanding implementations of IS-A Relations, in *Proc. of the conference on the ER Approach*, Cottbus, Oct. 1996, LNCS, Springer-Verlag (1996).

Hainaut, J-L. Transformation-based Database Engineering. In: [van Bommel, 2005]. (2005) 1–28

Hainaut, J-L., The transformational approach to database engineering, in *Generative and Transformational Techniques in Software Engineering*. Ralf Lämmel, João Saraiva, Joost Visser, eds, LNCS 4143, Springer-Verlag, pp. 89-138, 2006

Halpin, T., A., & Proper, H., A. Database schema transformation and optimization. *Proc. of the 14th Int. Conf. on ER/OO Modelling* (ERA). (1995)

Henrard, J., Hick, J-M. Thiran, Ph., Hainaut, J-L. Strategies for Data Reengineering, in *Proc. of WCRE'02*, IEEE Computer Society Press. (2002)

Hick, J-M., Hainaut, J-L. Strategy for Database Application Evolution: the DB-MAIN Approach, in *Proc. ER'2003 conference*, Chicago, Oct. 2003, LNCS Springer-Verlag. (2003)

Jajodia, S., Ng, P., A., & Springsteel, F., N. The problem of Equivalence for Entity-Relationship Diagrams, *IEEE Trans. on Soft. Eng.*, SE-9(5). (1983)

Kobayashi, I. Losslessness and Semantic Correctness of Database Schema Transformation : another look of Schema Equivalence, *Information Systems*, 11(1). (1986) 41-59

Lämmel, R. Coupled Software Transformations (Extended Abstract), In *Proc. First International Workshop on Software Evolution Transformations (SET 2004)*. (2004) [http://banff.cs.queensu.ca/set2004/set2004_proceedings_acrobat4.pdf]

Levene, M. *The Nested Universal Relation Database Model*, LNCS 595, Springer-Verlag. (1992)

Lien, Y., E. On the equivalence of database models, *JACM*, 29(2). (1982)

Ling, T., W. External schemas of Entity-Relationship based DBMS, in *Proc. of Entity-Relationship Approach : a Bridge to the User*, North-Holland. (1989)

McBrien P., & Poulovassilis, A. Data integration by bi-directional schema transformation rules, *Proc 19th International Conference on Data Engineering* (ICDE'03), IEEE Computer Society Press. (2003)

Motro, Superviews: Virtual integration of Multiple Databases, *IEEE Trans. on Soft. Eng*. SE-13, 7, (1987)

Navathe, S., B. Schema Analysis for Database Restructuring, *ACM TODS*, 5(2), June 1980. (1980)

Partsch, H., & Steinbrüggen, R. Program Transformation Systems. *Computing Surveys*, 15(3). (1983)

Poole, J. Model-Driven Architecture : Vision, Standards And Emerging Technologies. in *Proc. of ECOOP 2001*, Workshop on Metamodeling and Adaptive Object Models, (2001)

Rauh, O., & Stickel, E. Standard Transformations for the Normalization of ER Schemata. *Proc. of the CAiSE•95 Conf.*, Jyväskylä, Finland, LNCS, Springer-Verlag. (1995)

Roland, D. *Database engineering process modelling*, PHD Thesis, University of Namur. *http://www.info.fundp.ac.be/~dbm/publication/2003/these-dro.pdf* (2003)

Rosenthal, A., & Reiner, D. Theoretically sound transformations for practical database design. *Proc. of Entity-Relationship Approach*. (1988)

Rosenthal, & A., Reiner, D. Tools and Transformations - Rigourous and Otherwise - for Practical Database Design, *ACM TODS*, 19(2). (1994)

Schek, H-J., Scholl, M., H. () The relational model with relation-valued attributes, *Information Systems*, 11. (1986) 137-147

Thalheim, B. *Entity-Relationship Modeling: Foundation of Database Technology*. Springer-Verlag, (2000)

Thiran, Ph., Hainaut, J-L. Wrapper Development for Legacy Data Reuse. *Proc. of WCRE'01*, IEEE Computer Society Press. (2001)

Thiran, Ph., Estiévenart, F., Hainaut, J-L., Houben, G-J, A Generic Framework for Extracting XML Data from Legacy Databases, in *Journal of Web Engineering*, Rinton Press, (2005)

van Bommel, P. (Ed.). *Transformation of Knowledge, Information and Data: Theory and Applications*, Information Science Publ., Hershey. (2005)

van Griethuysen, J.J., (Ed.). Concepts and Terminology for the Conceptual Schema and the Information Base. Publ. nr. ISO/TC97/SC5-N695. (1982)