Modélisation des données et notation UML

Bruno Crémilleux avec la contribution de Jacques Madelaine
Département informatique de l'Université de Caen Basse-Normandie

18 octobre 2011

Table des matières

1 Introduction

1.1 Pourquoi une modélisation des données ?

Il est bien connu qu'avant d'entreprendre la réalisation informatique d'un "problème", il est nécessaire de réfléchir aux tenants et aboutissants du "système" à réaliser : il s'agit de passer du monde réel, complexe et confus, au monde informatique où les structures et les propriétés des objets doivent être identifiées (cf. figure 1). Cette tâche classique, entre autres en génie logiciel, est également essentielle dans la conception du schéma d'une base de données.

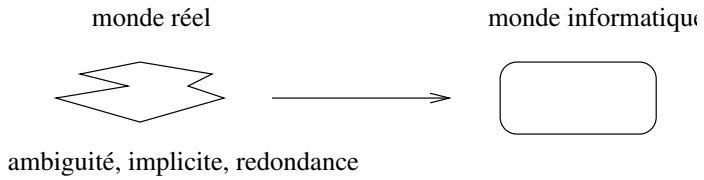

FIGURE 1 – Passage du monde réel au monde informatique

Cette phase de conception nécessite de nombreux choix qui auront des répercussions importantes dans la suite. De façon pragmatique, citons quelques problèmes concrets pouvant survenir lors de la réalisation d'une base de données qui montrent qu'il n'est pas aisé de déterminer les informations pertinentes à introduire dans celle-ci.

- **Au niveau technique.**
  - Il est difficile de déterminer les données à conserver. Qu'elles sont les données nécessaires pour construire un système décisionnel pour améliorer des ventes de voitures ? L'historique des ventes, les caractéristiques des véhicules, les ventes des concurrents, mais aussi, pourquoi pas, des caractéristiques sur la population afin de produire des profils de clientèle.
  - Des conversions sont parfois nécessaires. Un relevé géographique en Angleterre pourra indiquer des distances en miles, un système d'information géographique français utilisera probablement le kilomètre comme unité.
- **Au niveau humain.**
  - Toutes les personnes concernées par la gestion des données n'ont pas les mêmes besoins, elles n'ont pas forcément la même vision des données. Il est parfois nécessaire de gérer des intérêts contradictoires. Considérons par exemple la gestion des emplois du temps dans une école : les enseignants veulent savoir quand ont lieu leurs cours alors que le responsable des locaux souhaite savoir quelles sont les salles occupées et à quel moment. Les mêmes données sont ici exploitées avec des points de vue différents.
  - Certains répugnent parfois à partager leur savoir ou l'information, car, d'une certaine manière, un tel partage leur fait perdre une parcelle de pouvoir

Pour cela, il existe des méthodes (dites d'*analyse*) qui permettent de structurer et de présenter de manière abstraite le travail. La phase d'analyse est importante, c'est elle qui sera validée par les utilisateurs avant la mise en œuvre du système concret.

1.2 Démarche d'analyse dans la conception de schéma de BD

Traditionnellement, la démarche de conception s'effectue par abstractions successives en allant du problème à l'implantation du schéma dans le SGBD. Elle comporte 5 étapes :

1. **Perception du monde réel et capture des besoins.** C'est l'étude des problèmes des utilisateurs et de leurs besoins. Comme il est généralement trop compliqué d'aborder un problème dans son ensemble, les concepteurs réalisent des études de cas partiels. Le résultat est un ensemble de vues ou schémas externes.
2. **Élaboration du schéma conceptuel.** C'est l'intégration, dans un schéma global complet, non redondant et cohérent, de l'ensemble des schémas externes obtenus à l'étape précédente.

3. **Conception du schéma logique.** Transformation du schéma conceptuel en structures de données supportées par le système choisi (par exemple, passage à des tables avec un SGBD relationnel). Comme nous le verrons au paragraphe 3, cette étape peut être automatisée.
4. **Affinement du schéma logique.** Il s'agit de savoir si le schéma obtenue est un "bon" schéma (par exemple, sans redondance de données). Pour cela, le modèle relationnel s'appuie sur la théorie de la normalisation.
5. **Élaboration du schéma physique.** C'est le choix des bonnes structures physiques (groupement de tables, index,...) pour optimiser les performances. Ce travail nécessite la prise en compte des transactions.

La première étape relève plus du génie logiciel, voire de l'économie ou de la psychologie. Le paragraphe 2 détaille l'étape 2 et le 3 l'étape 3. On note que :

- la participation de l'utilisateur diminue à mesure qu'on avance dans les étapes de 1. à 5.
- la participation relative de l'informaticien augmente (surtout du point de vue technique) à mesure qu'on avance dans les étapes de 1. à 5.

2 Conception du schéma conceptuel

2.1 Modélisation avec entités et associations

2.1.1 Entités, attributs et associations

Le modèle entité/association (ayant pour origine les travaux de Chen) propose une modélisation du monde réel sous forme d'un ensemble d'"objets" (agrégation de données élémentaires), appelés entités et reliés par des associations.

Entités.

Intuitivement, une *entité* est un objet du monde réel, discernable parmi d'autres objets de la modélisation. Par exemple, un client, un compte bancaire, une voiture, un vin. Une entité peut aussi être "abstraite" (par exemple, une période de l'année). Plus précisément, on distingue :

- *une classe d'entités* qui regroupe un ensemble d'entités constitué par des données de même type. Par exemple la classe d'entités Client qui regroupe toutes les personnes ayant un compte en banque dans une modélisation d'une agence bancaire,
- *une occurrence d'entité* : en continuant l'exemple précédent, Jean, qui habite place du théâtre à Louvigny est une occurrence d'entité si l'agence bancaire a un unique client nommé Jean.

La notion de classe permet de spécifier des propriétés communes à un ensemble d'occurrences. Une occurrence vérifie les propriétés de la classe à laquelle elle appartient. Dans la suite, lorsqu'il n'y a pas de risque de confusion entre "classe d'entités" et "occurrence d'entité", nous emploierons le terme "entité".

Les classes d'entités ne sont pas forcément disjointes. On peut définir la classe des employés de la banque et la classe de ses clients à partir d'un ensemble de personnes. Une occurrence d'entité personne peut appartenir à l'une, aux deux, ou à aucune de ces 2 classes.

Attributs.

Une entité est caractérisée par un ensemble d'*attributs* définissant ses propriétés. Par exemple, une entité de classe Client est définie par un nom et une adresse. Pour chaque attribut, il existe un ensemble de valeurs autorisées (le domaine de l'attribut).

Associations.

Une *association* correspond à un lien logique entre deux entités ou plus. Il existe par exemple une association entre une classe d'entités Client et une classe d'entités Compte pour signifier que

des clients possèdent des comptes bancaires. Une association est souvent binaire (elle porte sur deux classes d'entités), mais elle peut être aussi ternaire ou plus généralement n-aire.

Comme pour les entités, on distingue une classe d'associations d'une occurrence d'association. Tout comme pour les entités, lorsqu'il n'y a pas de risque de confusion, nous emploierons le terme "association".

Une association peut posséder des attributs. Par exemple, "date" peut être un attribut de la classe d'associations reliant Compte à Client : l'attribut "date" indique la date la plus récente à laquelle le mode de fonctionnement du compte a été modifié (par exemple, ajout ou suppression d'un titulaire).

Choix du vocabulaire.

Pour clarifier la lecture des diagrammes (cf. paragraphe 2.2 et 2.3), le choix du vocabulaire repose sur la règle suivante :

- *noms* pour les entités,
- *verbes* pour les associations.

2.1.2 Cardinalités

A l'intérieur d'une classe d'associations, une occurrence d'entité peut être associée à aucune, une ou plusieurs entités d'une autre classe d'entités. Les *cardinalités* précisent ce point.

Cardinalités d'association pour une association binaire : l'extrémité d'une association est appelée "rôle" (le rôle décrit comment une classe voit une autre classe à travers l'association, cf. paragraphe 2.2). Chaque rôle d'une association porte une indication de multiplicité qui montre combien d'occurrences d'association de la classe d'entités considérée peuvent être liées à une occurrence de l'autre classe d'entités.

UML (cf. paragraphe 2.3) propose les notations indiquées au tableau 1.

1	un et un seul
0..1	zéro ou un
*	de zéro à plusieurs
0..*	de zéro à plusieurs
1..*	de un à plusieurs
m..n	de m à n (entiers naturels)

TABLE 1 – Notation des cardinalités d'associations

Une cardinalité se lit dans le sens entité vers association. Avec des associations binaires, cela revient à indiquer le nombre d'occurrences de l'autre entité pour une occurrence de la classe d'entités à laquelle est attachée la cardinalité du rôle.

La figure 2 illustre des exemples de cardinalités d'associations binaires entre deux classes d'entités E1 et E2 (en supposant que les exemples d'associations de la figure soient représentatifs de la réalité). Un trait représente une occurrence d'association. Les cardinalités réelles d'une association (celles observées dans le "système" réel) dépendent de l'environnement auquel s'applique la relation.

La détermination des valeurs de multiplicité est importante : lors de la phase de réalisation, il faudra choisir des structures de données (piles, files, ensembles,...) implantant les collections correspondant aux cardinalités. La surestimation des valeurs de multiplicité conduit à un sur-coût en taille de stockage et en vitesse de recherche. De même, une multiplicité égale à 0 implique que les opérations doivent contenir du code pour tester la présence ou l'absence de liens entre occurrences.

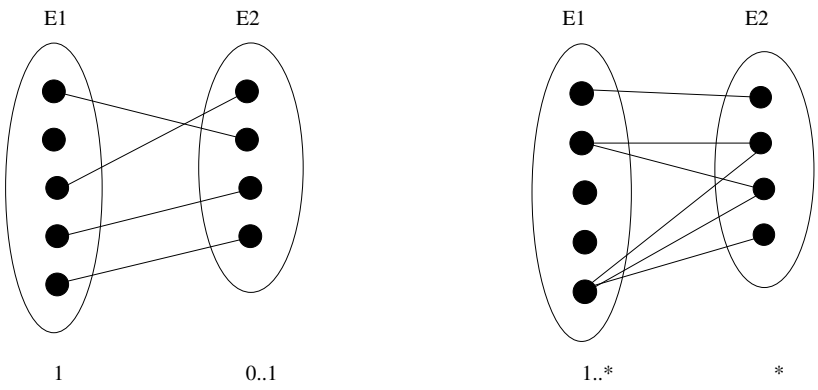

FIGURE 2 – Exemples de cardinalités

2.2 Notation en UML

Il existe différentes représentations graphiques d'une modélisation par entités et associations. Nous présentons ici celle utilisée dans le langage universel de modélisation UML (voir aussi le paragraphe 2.3 pour de plus amples détails sur UML).

Une classe d'entités est représentée par un rectangle (nommé par le nom de la classe d'entités) contenant les éventuels attributs, les associations par des traits simples. Le nom de l'association (normalement écrit en italique) apparaît au-dessus du trait (en cas d'ambiguïté, le symbole > précise le sens de lecture). Si une association possède des attributs, ceux-ci sont indiqués dans un rectangle accroché par un trait en pointillé à l'association.

Voici quelques exemples :

L'exemple bancaire : on suppose ici qu'un compte est toujours possédé par un seul client et un client peut détenir plusieurs comptes et toujours au moins un. Un client est ici caractérisé par un numéro, son nom et son adresse, un compte par son numéro et son solde.

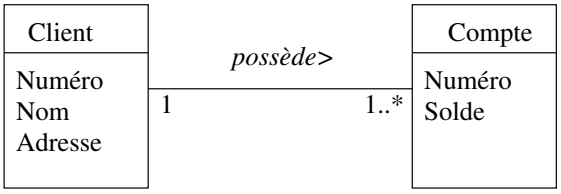

FIGURE 3 – Exemple d'association.

Personnes et voitures : pour illustrer qu'une association peut posséder des attributs (les traits pointillés sont ici appelés "lien d'attribut"). A chaque occurrence d'association, il correspond une date et un prix d'achat. Les cardinalités indiquent qu'une personne peut posséder de 0 à plusieurs voitures et qu'une voiture est possédée par au moins une personne.

Importance de nommer les rôles. L'extrémité d'une association est appelée "rôle". Le rôle décrit comment une classe se situe par rapport à une autre classe à travers une association. Le nom du rôle est placé près de l'extrémité. Le rôle permet d'exprimer des contraintes et aide la lecture du schéma (ce qui est d'autant plus important lorsque les classes d'entités de l'association ne sont pas disjointes ou qu'une même classe d'entités participe plusieurs fois à l'association). Par exemple, si on

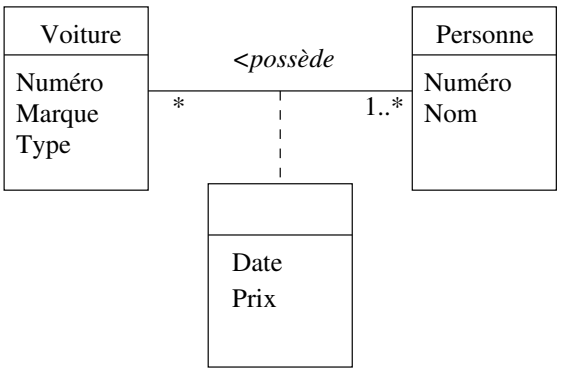

FIGURE 4 – Exemple d'association possédant un attribut.

considère la classe d'entités "Personne" et l'association de filiation parents/enfants, les cardinalités sont distinctes suivant qu'on considère les parents ou les enfants d'une personne (cf. figure 5). Ce type d'association est appelée association réflexive.

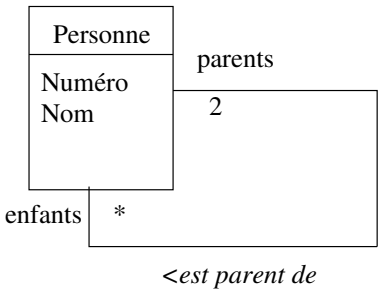

FIGURE 5 – Exemple d'association avec indication des noms des rôles.

Il en va de même pour l'association reliant un employé à son directeur, (en supposant qu'un directeur "dirige" des employés et qu'un employé "est subordonné à" un directeur) comme le montre la figure 6.

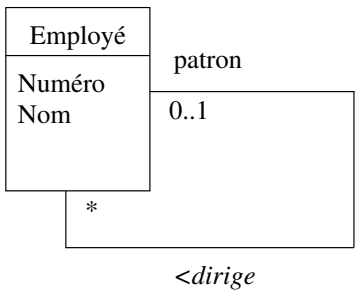

FIGURE 6 – Exemple d'association avec indication des noms des rôles.

Exemple d'association ternaire : association réunissant un étudiant et une machine qu'il utilise pour réaliser un projet. Une association ternaire est représentée par un losange où convergent les traits associatifs et, le cas échéant, les attributs de l'association. Pour une classe d'entités E, la cardinalité s'obtient en comptant le nombre de liens d'associations entre E et les autres entités de

l'association prises rassemblées.

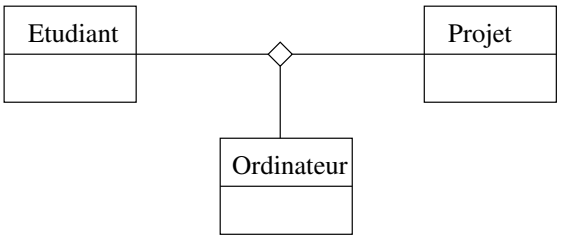

FIGURE 7 – Exemple d'association ternaire.

Une classe d'association ternaire peut se représenter par une classe associative (i.e. promouvoir l'association au rang de classe), et en ajoutant une contrainte exprimant que les branches de l'association s'instancient simultanément.

Exemple d'association étant elle-même source d'association : si les données d'une classe d'associations participent elle-même à une association, on nomme ces données. On a alors une véritable entité associative possédant une valeur pour chaque occurrence de l'association. Dans l'exemple de la figure 8, l'association *<travaille* correspond aussi à la classe d'entités "Emploi". Celle-ci participe à l'association *<dirige* : cette association indique que des emplois sont liés à d'autres emplois suivant le lien "*<dirige*. Cette association est indépendante des personnes "concrètes" : un emploi est un emploi de "patron" à cause de la nature de l'emploi et non pas à cause de la personne. Un emploi vu comme un emploi de patron dirige de zéro à plusieurs emplois. Un emploi vu comme un emploi sous la direction d'un patron est dirigé par 1 patron ou aucun (pour les emplois de patron par exemple).

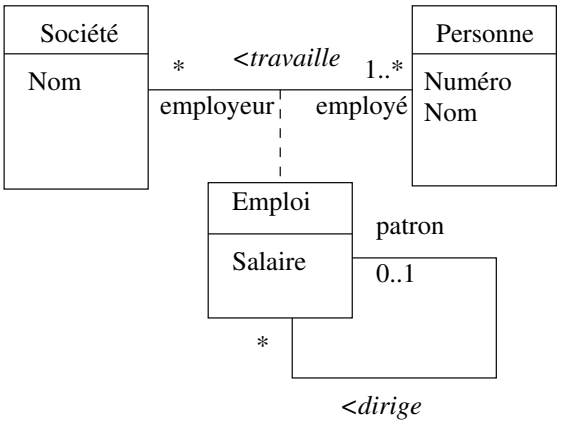

FIGURE 8 – Exemple d'association elle même source d'association.

Agrégation. Une agrégation est une forme particulière d'association : elle représente une association non symétrique dans laquelle une des extrémités joue un rôle prédominant (correspondant à l'entité "composite") par rapport à l'autre extrémité (appelée entité "composant"), mais les deux entités associées conservent une certaine autonomie. L'agrégation se représente par un losange creux situé à côté de l'agrégat (entité contenant le composant). Par exemple, un ordinateur se compose d'une unité centrale, d'un clavier et d'un écran (cf. figure 9).

La figure 10 schématise l'association *propriétaire>* : de par cette relation de "propriété", l'entité "Personne" joue un rôle prédominant. Une agrégation n'impose pas de contrainte particulière sur

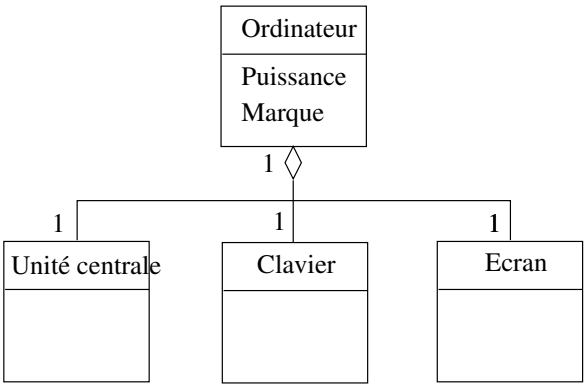

FIGURE 9 – Exemple d'agrégation.

les cardinalités : à la figure 10, le 1..* montre que des personnes peuvent être copropriétaires des mêmes immeubles.

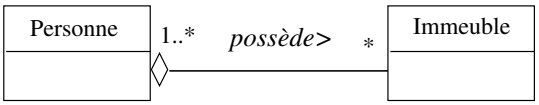

FIGURE 10 – Exemple d'agrégation avec agrégat multiple

Composition. La composition est un cas particulier d'agrégation : on l'utilise lorsqu'il y a une forte dépendance composant / composite et une coïncidence des durées de vie des parties et du tout. La composition se représente par un losange de couleur noire (mais d'autres notations où les composites sont représentés par des rectangles imbriqués dans le composant sont possibles). Avec la composition, la multiplicité de l'agrégat ne peut être que 0 ou 1 (pas de partage) et ne peut pas changer (liens fixes). La figure 11 représente une fenêtre du bâtiment (et aussi une fenêtre informatique...).

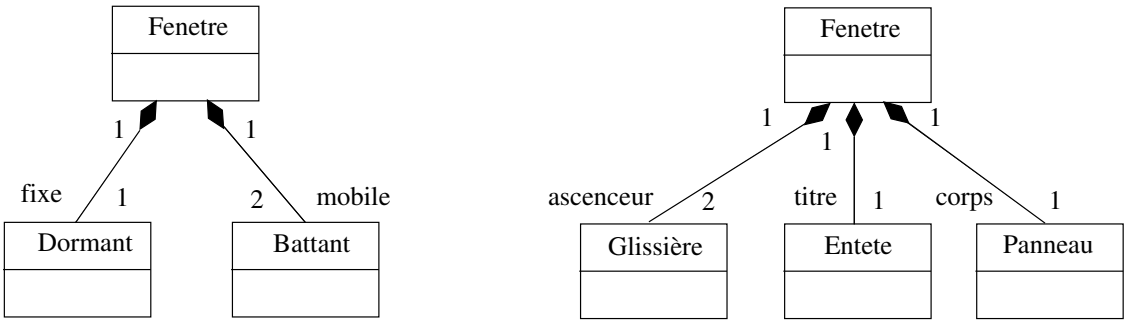

FIGURE 11 – Exemple de composition

La figure 12 schématise une voiture et ses deux sièges avant, modélisés sous la forme de composite.

On utilise aussi la composition lorsqu'un élément qui est a priori un attribut de l'entité agrégat est représenté comme une entité à part entière (c'est le cas pour "Puissance" et "Marque" à la figure 13 qui sont des attributs de l'agrégat à la figure 9). "Marque" ne représente pas ici l'entreprise

FIGURE 12 – Exemple de composition

qui produit l'ordinateur (dans ce cas, le lien serait une association, cette entreprise pouvant perdurer après la fin de l'existence de l'odinateur) mais à "l'étiquette" par exemple collée sur l'ordinateur et qui indique sa marque. On effectue cette opération par exemple lorsque l'attribut transformé en entité participe à d'autres relations dans le modèle.

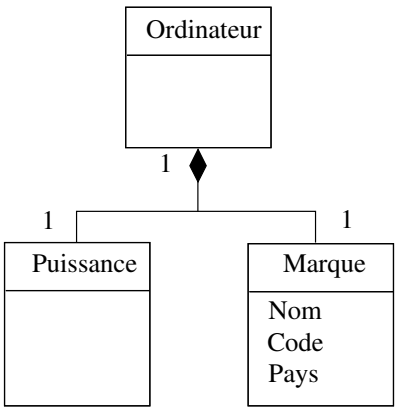

FIGURE 13 – Exemple de composition

Généralisation. La généralisation permet de factoriser dans une entité des attributs et/ou opérations (les opérations en UML sont définies au paragraphe 2.3). La généralisation n'est pas une association mais une relation de classification entre une classe générale (ou classe générique ou encore classe de base) et des classes plus spécifiques. Par exemple, une "personne" est un concept plus général qu'un "client". Un "étudiant" est un concept plus spécifique qu'une "personne". Une classe spécifique peut contenir des informations qui lui sont propres : par exemple, un étudiant est inscrit dans une filière. La généralisation signifie *est un* ou *est une sorte de* (certains auteurs réservent le lien *est un* à l'appartenance d'une occurrence à une classe). La figure 14 indique un exemple de généralisation : celle-ci se représente par un triangle dont la pointe est orientée vers la classe la plus générale (à la figure 14, les flèches sont agrégées en une seule, elles peuvent aussi être représentées de manière indépendante : présence d'une flèche entre chaque classe spécifique et la classe générique).

Les attributs et les opérations définies dans la classe générique sont héritées dans les classes spécifiques : un client hérite des attributs "Nom" et "Adresse" de la classe d'entités "Personne".

Une classe peut être reliée à plusieurs classes génériques : la généralisation est dite multiple (cf. figure 15). Les classes génériques n'ont pas forcément un ancêtre commun.

L'exemple de la figure 15 montre qu'une classe peut être spécialisée simultanément sur plusieurs critères (ici, "motorisation" et "milieu"). Ces critères sont indiqués sur le diagramme.

Des contraintes notées entre accolades (et portées sur le diagramme de façon attachées aux généralisations agrégées ou associées à une ligne pointillée reliant les relations de généralisation concernées) peuvent préciser la classification :

– *chevauchement ou inclusif :* une occurrence d'une classe générique peut être occurrence de

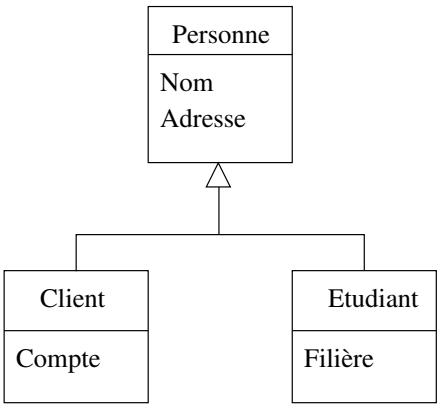

FIGURE 14 – Exemple de généralisation.

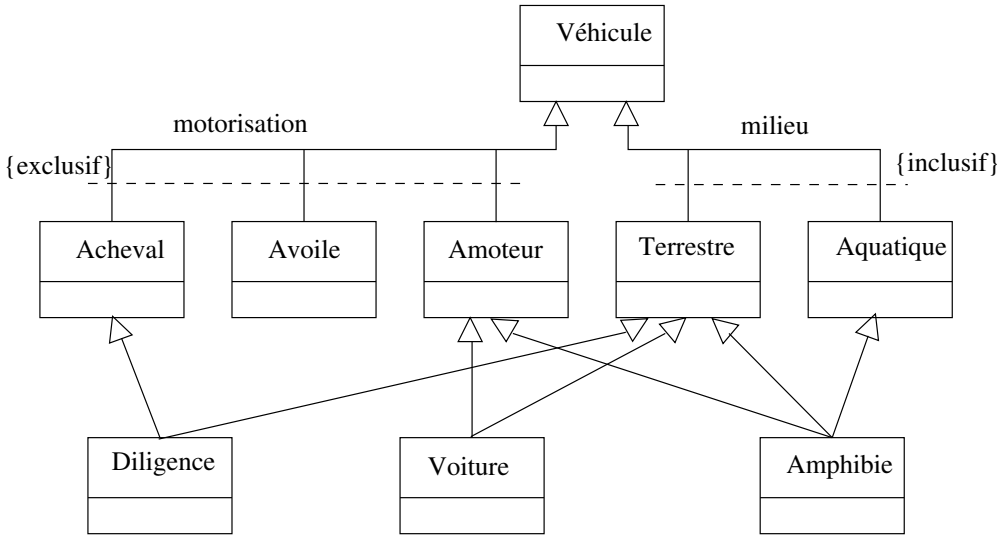

FIGURE 15 – Exemple de généralisation avec héritage multiple.

plusieurs classes spécifiques. Par exemple, un véhicule amphibie est occurrence à la fois de "Terrestre" et "Aquatique".
- *disjoint ou exclusif :* une occurrence d'une classe générique ne peut être occurrence que d'une seule de ses classes spécifiques (par exemple, on peut supposer que toute motorisation de véhicule est soit "A cheval", soit "A voile" ou soit "A moteur", mais que plusieurs types de motorisation ne sont pas possibles pour un même véhicule). C'est la contrainte par défaut. Cette contrainte interdit l'héritage multiple.
- *complet :* indique que toutes les classes spécifiques ont été ajoutées et on ne peut pas en introduire de nouvelles.
- *incomplet :* autorise l'ajout de nouvelles classes spécifiques. C'est la contrainte par défaut.

Contraintes. Enfin, de façon générale, des contraintes peuvent être définies sur une association. Les contraintes se notent entre accolades.

- *La contrainte ordonnée :* placée sur un rôle, elle spécifie qu'une relation d'ordre décrit les occurrences. Le modèle ne spécifie pas comment les éléments sont ordonnés, mais seulement que l'ordre doit être maintenu durant l'ajout ou la suppression d'occurrences par exemple. La figure 16 indique que la collection des comptes d'une personne est ordonnée.

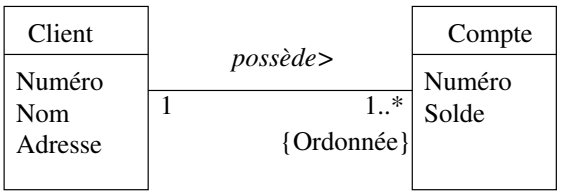

FIGURE 16 – Exemple d'utilisation de la contrainte {Ordonnée}.

- *La contrainte sous-ensemble :* elle indique qu'une collection d'occurrences est incluse dans une autre collection.
- *La contrainte ou exclusif :* elle précise que, pour une occurrence donnée, une seule association parmi un groupe d'associations est valide. La figure 17 montre qu'un compte appartient soit à une personne, soit à une société. L'ambiguïté introduite par le "ou exclusif" pour rechercher une occurrence associée empêche la notation des cardinalités.

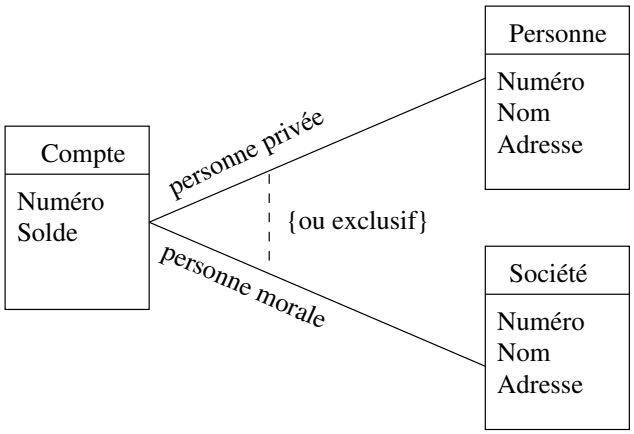

FIGURE 17 – Exemple d'utilisation du ou exclusif.

2.3 Quelques mots sur UML

La notation UML (pour *Unified Modelling Language for Object-Oriented Development*) est un langage de modélisation objet (et non pas une méthode objet) dont la première version date de 1996. UML est une norme de l'OMG (*Object Management Group*) qui est un consortium des principaux constructeurs et éditeurs de logiciels. La notation UML se veut intuitive, homogène, cohérente (élimination des symboles embrouillés, redondants ou superflus) et d'une sémantique précise : tout cela doit faciliter les échanges entre les différents intervenants. UML ne cherche pas la spécification à outrance : en cas de besoin, des précisons peuvent être apportées par des mécanismes d'extension et/ou des commentaires en texte libre.

UML définit différents modèles pour représenter les points de vue de la modélisation :

- *modèle des classes :* structure statique en termes de classes et de relations,
- *modèle des états :* comportement dynamique des objets,
- *modèle des cas d'utilisation :* fonctions du système du point de vue des utilisateurs,
- *modèle d'interaction :* représentation des scénarios et des flots de messages,
- *modèle de réalisation :* représentation des unités de travail,
- *modèle de déploiement :* répartition des processus sur les dispositifs matériels.

Ces modèles sont manipulés grâce à des diagrammes, ceux-ci pouvant correspondre à des vues complètes ou partielles des diagrammes. Il existe 9 sortes de diagrammes.

- *diagramme des classes :* structure statique,
- *diagramme des états/transitions :* comportement d'une classe en terme d'états,
- *diagramme d'objets :* représentation des objets (des occurrences des classes) et de leur relations, ils correspondent à des diagrammes de collaboration simplifiés (sans envoi de message),
- *diagramme de collaboration :* représentation des interactions entre objets sous forme de messages envoyés. Ils insistent plus particulièrement sur la structure spatiale statique qui permet la mise en collaboration d'un groupe d'objets.
- *diagramme des cas d'utilisation,*
- *diagramme d'activités :* représentation du comportement d'une opération en terme d'actions,
- *diagramme de séquences :* représentation temporelle des objets et de leurs interactions,
- *diagramme de composants :* représentation des composants physiques d'une application,
- *diagramme de déploiement :* représentation du déploiement des composants sur les dispositifs matériels.

Pour définir un modèle, on doit disposer d'un modèle de ce modèle (ou *meta-modèle*). Le meta-modèle de UML est UML lui-même.

Rappelons qu'une classe est une extension du concept d'entité avec des opérations comme le montre la figure 18.

Les attributs et les opérations sont précédés d'un niveau de visibilité :

- $+$: pour public : élément visible à tous les clients de la classe,
- $\#$: pour protégé : élément visible aux sous-classes de la classe,
- - : pour privé : élément visible à la classe seule.

Dans une description en langage naturel, les classes correspondent souvent à des noms. A partir des objets, on abstrait pour découvrir leurs propriétés, attributs et méthodes. Une réflexion sur le cycle de vie de l'objet et sur ses collaborations avec les autres objets permet de préciser les méthodes et les attributs manipulés par les méthodes.

UML définit un petit nombre de mécanismes communs qui assurent l'intégrité de la notation :

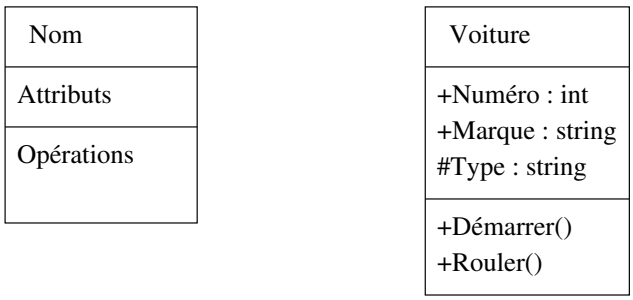

FIGURE 18 – Représentation d'une classe en UML.

– *stéréotypes :* un élément spécialisé par un stéréotype permet de définir une classe supplémentaire du meta-modèle et il est noté entre guillemets. Chaque élément a au plus un stéréotype. Par exemple, la figure 19 montre qu'un acteur peut être vu comme un stéréotype particulier d'une classe appelée "Acteur". Un certain nombre de stéréotypes sont prédéfinis en UML ("signal" pour déclencher une transaction dans un automate, "metaclasse" : la classe d'une classe,...). Le stéréotype rend possible l'identification d'une typologie de classes. Ils permettent une réelle capacité d'adaptation et d'évolution de la notation pour prendre en compte les particularités des différentes situations à modéliser.

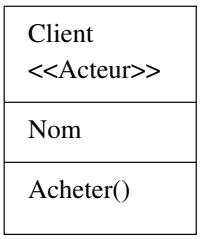

FIGURE 19 – Exemple d'une classe stéréotypée.

– *étiquettes :* paire (nom, valeur) : pour les propriétés d'un élément de modélisation, les propriétés permettent l'extension des attributs.
– *notes :* un simple commentaire.
– *contraintes :* relation sémantique quelconque entre éléments (exprimées en langage naturel, pseudo-code,...)
– *relation de dépendance :* relation d'utilisation unidirectionnelle entre deux éléments de modélisation. Les notes et contraintes peuvent être source d'une relation de dépendance.
– *dichotomie type/instance et type/classe :*
  – dans type/instance : le type dénote l'essence de l'élément et l'instance est une manifestation du type,
  – dans type/classe : le type est la spécification d'un élément, la réalisation de cette spécification est fournie par la classe.

Paquetage.
Les paquetages sont un mécanisme général pour la partition des modèles et le regroupement des éléments. Chaque paquetage est représenté graphiquement par un dossier (un rectangle avec un petit rectangle en haut à gauche). Il correspond à un sous-ensemble du modèle et contient, selon le modèle, des classes, des objets, des relations, des composants ou des nœuds ainsi que les diagrammes associés.

La décomposition en paquetage n'est pas fonctionnelle mais correspond à un critère logique du système à construire (par exemple, pour le système d'une entreprise, il y a aura le paquetage

“référentiel”, “domaine client”, “domaine fournisseur”). La forme générale du système est exprimée par la hiérarchie de paquetages et par le réseau de relations de dépendance entre paquetages. Un paquetage peut contenir d'autres paquetages.

Une relation de dépendance (schématisée par une flèche en pointillé orientée du paquetage Client vers le paquetage Fournisseur, cf. figure 20) entre deux paquetages signifie qu'au moins une classe du paquetage Client utilise les services offerts par au moins une classe du paquetage Fournisseur.

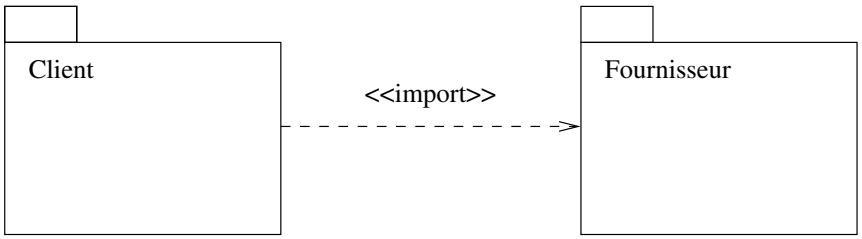

FIGURE 20 – Relations de dépendance entre deux paquetages.

La figure 21 montre un exemple de paquetage correspondant à un schéma externe en UML.

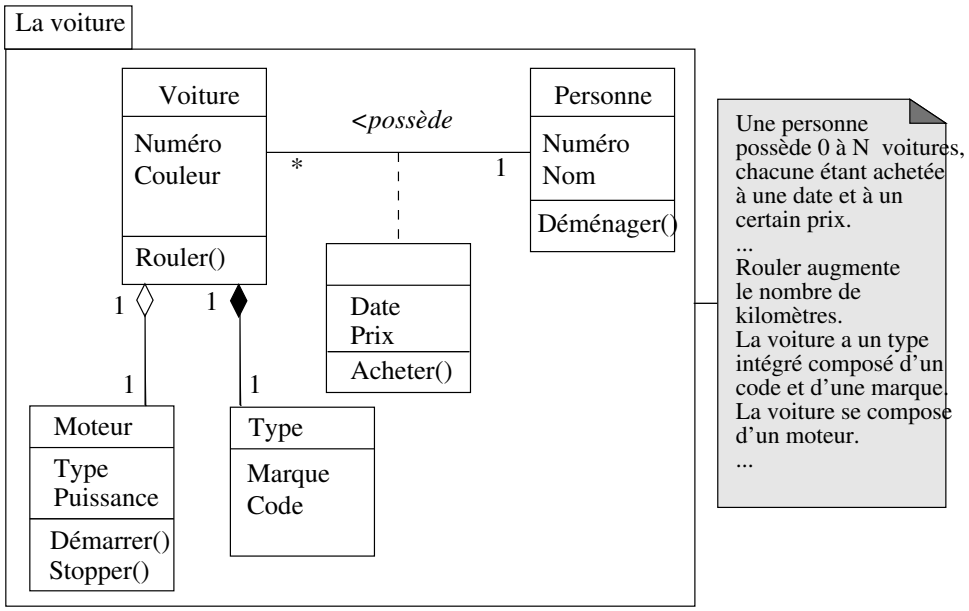

FIGURE 21 – Exemple de schéma externe en UML.

2.4 Conclusion : intégration de schémas externes

Dans la pratique, les différents schémas externes (ou vues, ou encore paquetages) représentent des sous-ensembles du schéma conceptuel de la base de données. Il est ensuite nécessaire d'intégrer ces différentes parties en un schéma externe global.

Des conflits peuvent provenir sur les noms d'entités et d'associations, de leur définition (inclusion, intersection non vide, noms différents pour classes équivalentes,...), des structures (attributs manquants, associations et/ou entités partagées par plusieurs schémas et donc à regrouper,...), de contraintes de cardinalités, etc.

Une méthode pour isoler les conflits et la construction d'un dictionnaire de noms, voire d'une ontologie spécifique au domaine (c'est-à-dire d'une définition complète des concepts avec leurs relations sémantiques ramenés à un référentiel unique afin de mesurer la distance et le recouvrement entre eux).

3 Du schéma conceptuel au schéma logique : passage au relationnel

Le passage du schéma conceptuel exprimé sous la forme d'un diagramme représentant les entités et les associations (comme par exemple le diagramme de classes en UML) au modèle relationnel est aisé et peut être automatisé. Pour ce passage, on dispose d'un principe général et d'un principe spécifique suivant les cardinalités de l'association. Ces principes respectent les règles de normalisation (autrement dit, le schéma relationnel obtenu doit être, si la modélisation est correcte, en 4ème forme normale).

Précisons qu'une autre manière de concevoir un schéma relationnel est celui de la normalisation : cette dernière démarche repose sur les décompositions successives d'une relation universelle.

3.1 Principe général

Les deux règles suivantes transforment un diagramme en schéma de bases de données :

- pour chaque entité, construire une relation qui a pour attributs ceux de l'entité,
- pour chaque association, construire une relation qui a pour attributs ceux de l'association ainsi qu'une clé de chaque entité participant à l'association.

Les relations issues des associations permettent de lier les informations provenant des entités.

Considérons le diagramme de la figure 22 : un client peut posséder plusieurs comptes et les comptes-joints sont autorisés. L'attribut Date indique la date où un client devient titulaire ou co-titulaire d'un compte (cette date ne correspond pas forcément à la date d'ouverture du compte et n'est pas un attribut de Compte, car, pour un compte-joint, des clients peuvent être ajoutés au cours de la durée de vie du compte).

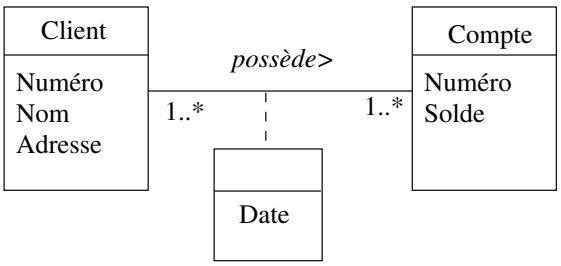

FIGURE 22 – Association avec multiplicité * pour chaque rôle.

On obtient le schéma suivant :
CLIENT (Numéro, Nom, Adresse)
COMPTE (Numéro, Solde)
POSSEDE (NuméroClient, NuméroCompte, Date)

Attention, (NuméroClient, NuméroCompte) est une clé de POSSEDE uniquement parce qu'une seule date est conservée. Si plusieurs dates étaient possibles, l'attribut Date devrait être introduit dans la clé. De façon générale, la réunion des clés des entités associées n'est pas une clé de l'association.

Agrégations et compositions étant des associations, celles-ci suivent ces règles. Cependant, une composition possèdant une cardinalité égale à 1, on utilise plutôt le principe spécifique (cf. section 3.2) pour une composition pour passer au relationnel.

3.2 Principe spécifique

Si dans une association une classe d'entités (notée E) a un rôle avec une multiplicité égale à 1, alors la relation issue de l'association peut être supprimée et remplacée en introduisant les attributs de l'association et une clé de E dans les relations issues des entités de l'association autres que E.

Supposons maintenant qu'un compte appartient à un et un seul client (cf. figure 23). Remarquez que la date où un client devient titulaire du compte est maintenant un attribut de Compte, c'est la date d'ouverture du compte (si le titulaire du compte change, avec cette modélisation, le compte est clôturé et un autre est ouvert, ou alors il n'est pas possible de conserver la date d'ouverture du compte).

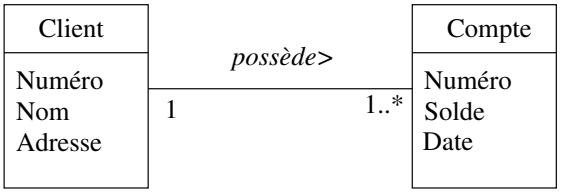

FIGURE 23 – Association avec multiplicité 1 pour un des rôles.

On obtient le schéma suivant :
CLIENT (Numéro, Nom, Adresse)
COMPTE (Numéro, Solde, Date, NuméroClient)

Ce principe peut être poussé plus loin si tous les rôles de l'association ont une multiplicité égale à 1, alors, une seule relation suffit comme le montre l'exemple de la figure 24 (un compte appartient à un seul client et un client possède un seul compte) :

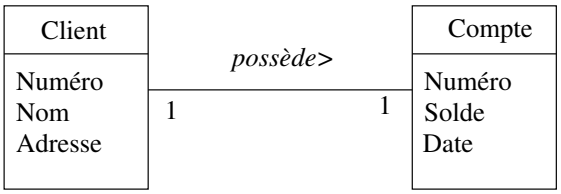

FIGURE 24 – Association avec multiplicité 1 pour tous les rôles.

On obtient le schéma suivant :
CLIENT-COMPTE (NuméroClient, Nom, Adresse, NuméroCompte, Solde, Date)
avec deux clés : (NuméroClient) et (NuméroCompte).

Remarques :

1. Cette diminution du nombre de relations n'est pas conseillée si la cardinalité minimale d'un rôle est égale à 0. En effet, dans cette situation, des valeurs nulles seront présentes pour les occurrences de l'association non liées à l'entité. Par exemple, la figure 25 autorise la présence de compte sans client (ce qui est peu vraisemblable dans la réalité...) et il y a au plus un client titulaire d'un compte :
   Si on utilise le schéma :
   CLIENT (Numéro, Nom, Adresse)

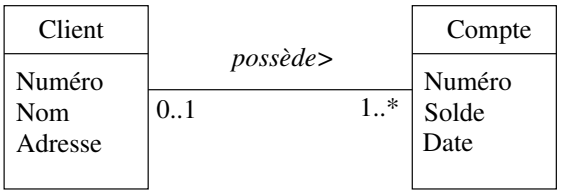

FIGURE 25 – Association avec multiplicité 0..1 pour un rôle.

COMPTE (Numéro, Solde, Date, NuméroClient)
alors il y aura présence d'une valeur nulle pour NuméroClient pour un compte sans titulaire.

2. Dans certaines situations, il est nécessaire de s'interroger sur l'opportunité de rassembler en une seule relation ce que le concepteur du diagramme avait pris soin de séparer.

3.3 Cas de la généralisation

Si on dispose d'un SGBD relationnel étendu objet (comme par exemple Postgres), la généralisation s'implante directement par un héritage entre tables. Sinon, une traduction de la généralisation dans le modèle relationnel doit être effectuée. La stratégie la plus classique est de créer une relation pour chaque classe d'entités (cela suit le principe général du paragraphe 3.1). Cependant, comme il n'y a pas d'héritage entre relations dans le modèle relationnel, il est nécessaire d'introduire une clé de la relation issue de la classe générique dans chaque relation issue des classes spécifiques pour passer de l'une aux autres (cela correspond au principe du paragraphe 3.2 car à une occurrence d'un spécifique il correspond une seule occurrence du générique). À cause de l'héritage multiple, un spécifique peut être rattaché à plusieurs génériques. Cette méthode revient à implanter l'héritage sous forme d'associations et ainsi à "perdre" la notion d'héritage : lors des requêtes, il faudra explicitement indiquer la clé du générique pour accéder à l'information détenue par celui-ci.

Une autre stratégie est de supprimer au niveau du schéma relationnel la relation issue du générique. Dans ce cas, les attributs du générique sont "importés" dans chaque spécifique. Cette stratégie permet de disposer dans une unique table de toute l'information liée à chaque spécifique sans effectuer une jointure entre le spécifique et son générique mais elle oblige à effectuer une "union" sur les tables des spécifiques pour une requête qui s'adresse aux informations relevant du générique. De plus, deux autres inconvénients sont possibles :

- lorsque les classes spécifiques ne sont pas exclusives (par exemple, si un véhicule est à la fois "terrestre" et "aquatique" (cf. exemple de la figure 15)), les attributs de la classe générique (ici Véhicule) sont dupliqués dans les relations issues des classes spécifiques ("terrestre" et "aquatique") pour les véhicules "terrestre" et "aquatique".
- si un objet appartient à la classe générique mais n'est spécialisé dans aucune classe spécifique (existence par exemple d'un véhicule dont on ne connait pas la motorisation ou le milieu, cf. figure 15), on ne sait pas dans quelle relation introduire les valeurs de cet objet.