Dauphine | PSL UNIVERSITÉ PARIS | MIDO MATHÉMATIQUES ET INFORMATIQUE DE LA DÉCISION ET DES ORGANISATIONS

Licences Mathématiques et Informatique
3ème année - Formation Initiale et par Apprentissage

Bases de données relationnelles

Polycopié de cours - Passage au relationnel

Maude Manouvrier

Table des matières

Chapitre 7

Modélisation Entité/Association et UML et Passage au relationnel

7.1 Modélisation Entité/Association et UML

La construction d'une base de données commence par la construction du schéma conceptuel c'est-à-dire par l'analyse de l'information que doit contenir la base et des relations qui existent entre les différentes informations de la base [21]. Le modèle le plus connu pour modéliser une base de données relationnelle est le **modèle Entité/Association** (*Entity-Relationship* [3] ou E/R en anglais).

Il est à noter que n'importe quelle autre méthode de conception [1] (Merise, OMT, UML, etc.) peut être utilisée pour représenter graphiquement le schéma conceptuel d'une base de données relationnelle.

Nous présentons, dans ce document, le modèle Entité-Association et le modèle UML.

7.1.1 Notions d'entité ou d'instance de classes

NB : Lorsqu'ils sont introduits, les termes présentés en gras font référence aux concepts au modèle entité-association. Les termes soulignés font référence aux concepts du modèle UML. Lorsque le terme est souligné et en gras, il est commun aux deux modèles. Pour bien comprendre la différence entre ces deux modèles, il est possible de consulter l'ouvrage [20] qui présente la différence entre Merise et UML dans le domaine des bases de données.

Une **entité** ou une instance de classe (également appelé objet) est un élément ou un objet concret (par exemple un produit fabriqué par une usine) ou abstrait (la livraison d'une commande) du monde réel qui existe et que l'on peut distinguer des autres.

Les entités ou les instances de classe de même type sont organisées en **ensembles d'entités** ou en classes. Un ensemble d'entités ou une classe regroupent des entités/instances ayant des propriétés similaires, par exemple l'ensemble des personnes ayant un compte dans une banque.

Une entité/instance est représentée par un ensemble d'**attributs** qui la décrivent. Par exemple une voiture est décrite par son numéro d'immatriculation, sa couleur, sa marque, son année de fabrication etc. Chaque attribut a un **domaine** qui correspond à l'ensemble des valeurs qu'il peut prendre.

1. Ce terme est employé de manière abusive, UML est un langage ...

L'ensemble des valeurs des attributs d'une entité/instance la distingue de toutes les autres entités/instance du même ensemble d'entités / de la même classe. En effet, toutes les entités d'un même ensemble ou les instances d'une même classe sont tous différents. Un sous-ensemble des attributs permet d'identifier l'entité/l'instance de manière unique (de la distinguer des autres entités/instances). Par exemple le *numéro d'immatriculation* permet d'identifier une entité/instance "Voiture", le *titre* et l'*année de sortie* permettent d'identifier une entité/instance de "Film". On appelle ce sous-ensemble, l'**identificateur** de l'entité/instance. En UML, en général on ne représente pas l'identificateur, il est implicite. Pour faciliter le passage au relationnel, il sera néanmoins représenté dans les schéma de modélisation du cours et des exercices (en gras ou souligné [2]).

7.1.2 Association

Une **association** correspond à l'existence d'un lien entre entités/instances. Par exemple, un livraison va concerner le produit de telle ou telle entreprise. Les associations de même type sont regroupées au sein d'un **ensemble d'associations**. Une association peut avoir des attributs.

La **cardinalité**/multiplicité d'une association est le nombre d'entités/instances que l'association relie. Une cardinalité, en Entité/Association, est notée *(Min :Max)* ou *(Min,Max)* et peut prendre les valeurs suivantes par exemple : *0 :1* ou *1 :1* ou *3 :8* ou *3 :3* ou *1 :N* avec la lettre N représentant 'plusieurs'.
Une multiplicité en UML est notée *(Min ... Max)*, et la valeur n'apparaît qu'une seule fois quand les bornes Min et Max sont égales. Le symbole $*$ correspond à zéro ou plusieurs. On peut avoir par exemple les valeurs suivantes : *0..1* ou *1* ou *0..** ou *3..8* ou *3* ou ***.

Dans la suite de la section, on utilise l'exemple de base de données de [16]. Cette base de données gère une université. Elle doit, en particulier, contenir de l'information sur les enseignants (*Enseignant* dans la base exemple), sur les départements (*Departement* dans la base exemple), les matières enseignées (*Cours* dans la base exemple), les salles (*Salle-de-Cours* dans la base exemple).

La représentation entité/association standard et la représentation Merise sont présentées sur la figure 7.1 selon 2 formalisme : celui de P.Chen que l'on trouve dans certains ouvrages et le format Merise, plus employé et que l'on utilisera dans le cours et les exercices. Cette figure représente deux ensemble d'entités, les enseignants et les départements et l'association qui existe entre enseignants et départements : un enseignant appartient à un et un seul département, un département contient 0 à N enseignants.

La figure 7.2 modélise les mêmes concepts en UML. **Il est à noter que les cardinalités dans le formalisme de P. Chen et les multiplictés en UML sont représentées dans le même sens, mais dans le sens opposé en Entité association au format Merise**.

2. Attention, il s'agit d'une petite liberté prise avec UML, les attributs soulignés ici ne correspondent pas à des attributs dérivés mais aux identificateurs (pour ne pas les oublier lors du passage au relationnel ! !).

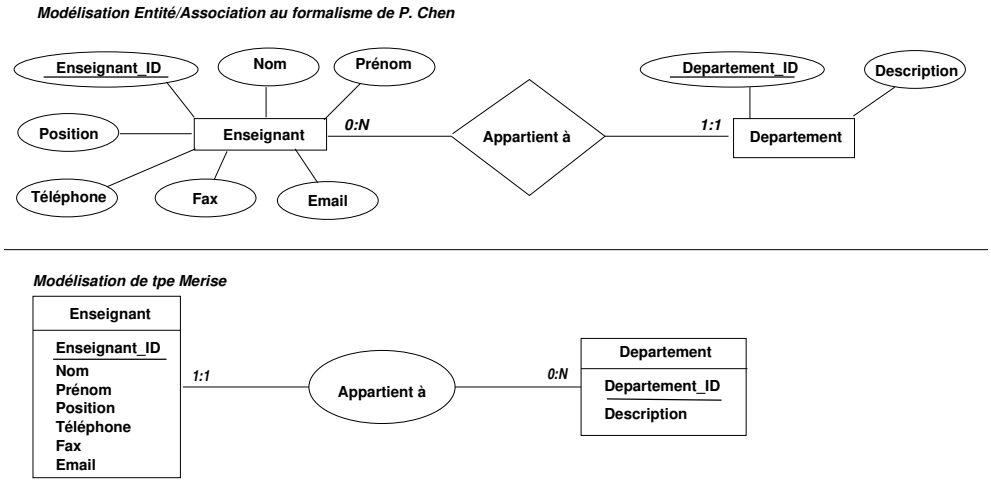

FIGURE 7.1 – Un exemple de modélisation E/A au formalisme de P. Chen et au formalisme Merise.

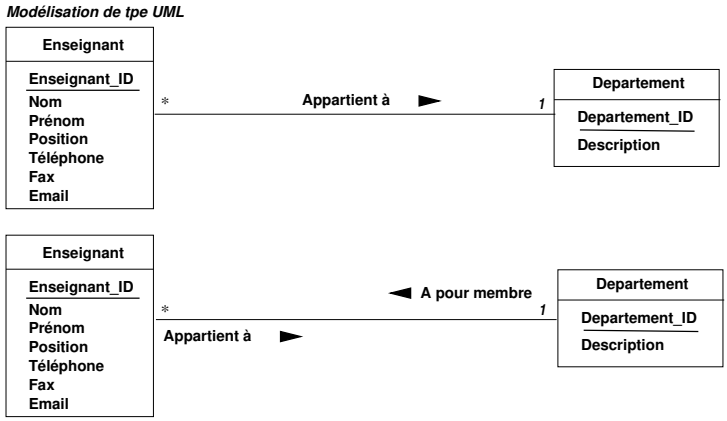

FIGURE 7.2 – Un exemple de modélisation UML.

Il existe des **associations n-aires**, où n représente le nombre d'ensembles d'entités ou classes associé(e)s. La figure 7.3 représente une association ternaire : une matière est enseignée par un enseignant dans une salle donnée. L'association a en plus des attributs qui indique le semestre et l'année pendant lesquels le cours a été fait.

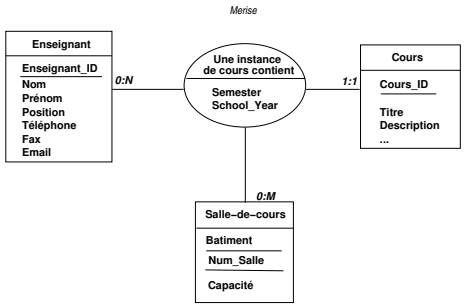

FIGURE 7.3 – Un exemple d'association ternaire.

Cependant, il est préférable d'éviter les associations n-aires, car par exemple les cardinalités ne sont pas toujours évidentes à trouver. Une association n-aire peut-être convertie en plusieurs associations binaires. Par exemple, l'association ternaire de la figure 7.3 peut être convertie en trois associations binaires, en introduisant un ensemble d'entités *Seance*. Les figures 7.4 et 7.5 représentent des exemples de conversion.

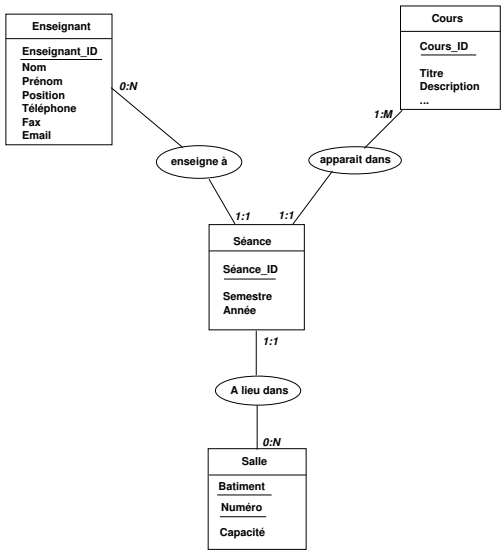

FIGURE 7.4 – La conversion de l'association tertiaire de la figure 7.3 en associations binaires.

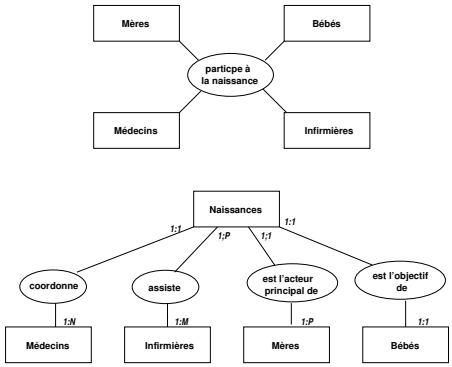

FIGURE 7.5 – Deux représentations des naissances [21].

En UML [20], les associations n-aires sont peu utilisées. On utilisera plutôt une classe-association (voir la section *Agrégation* ci-après).

7.1.3 Agrégation

Dans la modélisation Entité/Association, l'**agrégation** est un mécanisme par lequel des associations sont traitées comme des ensembles d'entités de plus haut niveau. L'association ternaire de la figure 7.3 peut également être représentée par une agrégation, comme celle représentée sur la figure 7.6 [3].

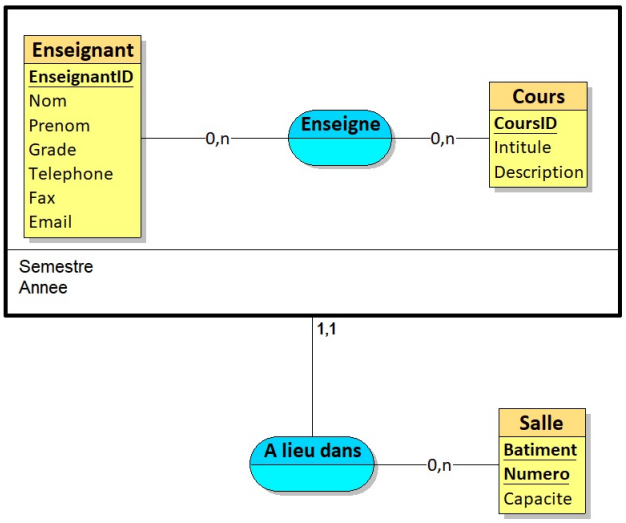

FIGURE 7.6 – Un exemple d'agrégation en modélisation E/A.

En UML, on utilise le concepts de classe-association, tel qu'il est représenté sur la figure 7.7, qui représente une modélisation équivalente à celle de la figure 7.6.

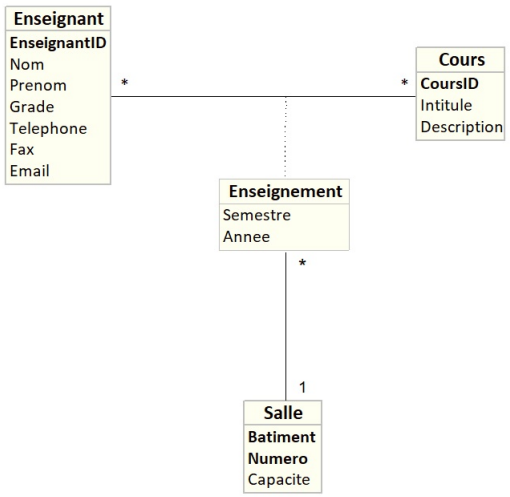

FIGURE 7.7 – Un exemple de classe-association en UML.

3. Figure réalisée sous le logiciel `https://www.looping-mcd.fr/`

7.1.4 Généralisation/Spécialisation ou Héritage

Dans la modélisation Entité/Association peut considérer des sous-ensembles d'entités :

- Soit les entités de ce sous-ensemble forment un recouvrement de l'ensemble d'entités. Par exemple l'entité *Personne* dans la figure 7.8 est un recouvrement des entités *Etudiant* et *Enseignant*. On parle alors de **Généralisation** en modélisation Entité/Association. La même notion est appelée est appelée **Héritage** en UML (voir la figure 7.9).
- Soit les entités de ce sous-ensemble ont des propriétés (attribut ou association) particulières. Dans l'exemple, un *Enseigant* a un grade et un *Etudiant* est caractérisé par une adresse et une année d'entrée à l'Université. C'est la **spécialisation**.

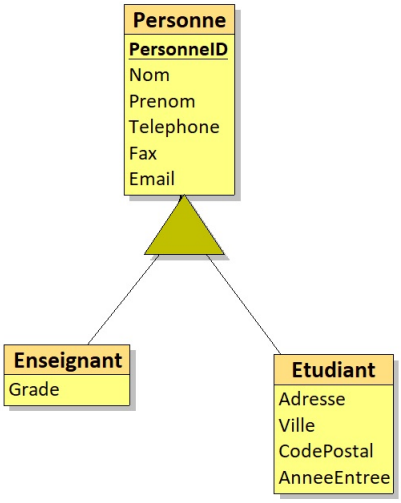

FIGURE 7.8 – Un exemple de généralisation/spécialisation en E/A.

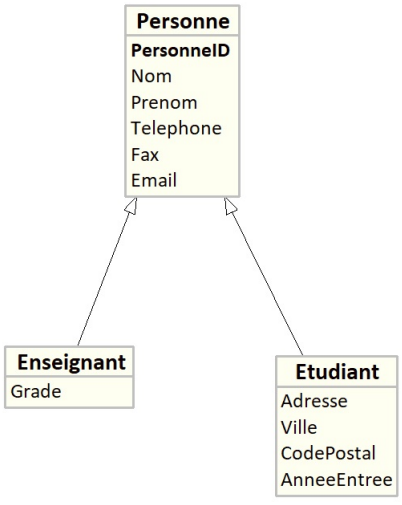

FIGURE 7.9 – Un exemple d'héritage en UML équivalent à la modélisation E/A de la figure précédente.

Pour représenter une généralisation/spécialisation, on utilise une association spéciale, l'association en triangle (parfois nommée **Is_a**). Les entités spécialisées héritent des attributs de l'entité qui les généralise. Un exemple est donné sur la figure 7.8.

Dans la modélisation UML, le principe est le même, la représentation diffère un peu comme présenté sur la Figure 7.9. On parle dans ce cas de classe mère ou super-classe et de classes filles, classes dérivées ou sous-classes.

7.1.5 Ensemble d'entités faibles en E/A ou association qualifiée en UML

En modélisation E/A, **entité faible** est une entité qui doit être associée à une autre entité (dite **forte**) pour exister (pour être identifier). Par exemple, si on gère des salle de cours et des bâtiment, un même numéro de salle peut apparaître dans différents bâtiments (ex. la salle 302 - à Dauphine il y a la salle 302 du bâtiment B, du bâtiment P et du bâtiment A). Un ensemble d'entités faibles ainsi que l'association qui la relie à son ensemble d'entités fortes sont représentés comme des ensemble d'entités et des associations classiques en doublant le contour. Un exemple est donné sur la figure 7.10.

En UML, ce concept n'existe pas, mais il est possible d'utiliser une association qualifiée. Cette association indique que l'identificateur *Numéro* de *Salle* doit être associé à l'identificateur de *Bâtiment* pour permettre d'identifier une salle. Un exemple est donné sur de la figure 7.11.

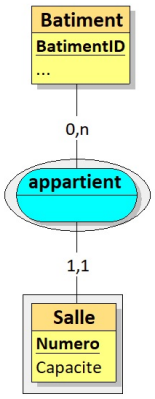

FIGURE 7.10 – Un exemple d'entité faible en modélisation E/A.

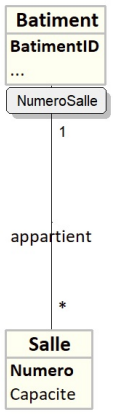

FIGURE 7.11 – Un exemple d'entité faible en modélisation E/A et d'association qualifiée en UML.

7.1.6 Dépendances fonctionnelles

Il existe des contrainte exprimant des liens de déduction entre attributs, les dépendances fonctionnelles. Un attribut (ou un groupe d'attributs) Y **dépend fonctionnellement** d'un attribut (ou groupe d'attributs) X si, étant donné une valeur de X, il lui correspond une valeur unique de Y (quel que soit l'instant considéré) [9]. On note une dépendance fonctionnelle par `X->Y`, signifiant que Y dépend fonctionnellement de X ou que X détermine fonctionnellement Y. Cela signifie que à une valeur de X est associée une et une seule valeur Y ou encore que si on connaît la valeur de X, alors on peut en déduire celle de Y.

Les dépendances fonctionnelles doivent être déclarées au niveau du schéma conceptuel.

Prenons par exemple un ensemble d'entité voiture [9] dont les attributs sont : l'immatriculation (IMA), la marque ($MARQUE$), le modèle ($MODELE$) (ex. C3), la puissance ($PUISS$) et la couleur ($COUL$). On a les dépendances fonctionnelles suivantes :

```
IMA -> MARQUE, MODELE, PUISS, COUL
MODELE -> MARQUE
(MODELE,TYPE) -> PUISS
```

Ces dépendances fonctionnelles signifient :
- Qu'à un numéro d'immatriculation est associé une et une seule valeur des autres attributs.
- Que toutes les voitures du même modèle sont de la même marque. Par exemple, toutes les voitures de modèle C3 sont de la marque Citroën.
- Que toutes les voitures du même modèle et du même type ont toutes les mêmes puissances.

Nous reviendrons par la suite sur les dépendances fonctionnelles dans un prochain chapitre.

7.2 Passage du modèle E/A ou UML au modèle relationnel

La table ci-dessous représente la différence de vocabulaire dans les 2 modèles présentés précédemment.

Table 7.1 – Comparaison des modèles E/A et UML.

Modèle E/A	**UML**
ensemble d'entités	classe
entité	objet
ensemble d'entités faibles	association qualifiée
ensemble d'associations	association
généralisation	classe mère/superclasse
spécialisation	classe fille/classe dérivée/sous-classe
agrégation	classe association
cardinalité	multiplicité

La table suivante présente la différence de vocabulaire avec le modèle relationnel. Cette section présente comment passer d'un schéma en E/A ou UML en un modèle relationnel.

TABLE 7.2 – Comparaison du vocabulaire des modèles E/A et relationnel.

Modèle E/A ou UML	**Modèle relationnel**
ensemble d'entités/classe	relation
entité/objet	nuplet
attribut	attribut
ensemble d'entités faibles/association qualifiée	relation
associations	relation
identifiant	clé primaire

Le passage du modèle E/A ou UML vers le modèle relationnel se fait par règles. Il est préférable de réfléchir à ce que signifie le modèle E/A ou UML et le modèle relationnel, plutôt que d'apprendre ces règles par cœur.

Le passage au relationnel se fait généralement en 2 étapes (qui peuvent être fusionnées, une fois que l'on a compris le principe) :

1. **Transformation des ensembles d'entités ou classes :** à chaque ensemble d'entités ou classes correspond une relation avec les mêmes attributs.

   Il faut faire attention aux ensembles d'entités faibles ou association qualifiée et aux généralisations/spécialisations ou hiérarchies de classes.

2. **Transformation des associations. Cette étape peut entraîner, en fonction des cardinalités/multiplicités :**
   (a) La modification de relations créées à l'étape 1.
   (b) La création de nouvelles relations.

Dans la suite, par convention, on soulignera les clés primaires des relations et les clés étrangères seront représentées par un #.

7.2.1 Règles portant sur la transformation des ensembles d'entités/classes

1. Pour chaque ensemble d'entités/classe E, on crée une relation R ayant les mêmes attributs que ceux de l'ensemble d'entités/classe. L'identifiant de E est la **clé primaire** de R.
2. Pour chaque ensemble d'entités faibles/classes ayant une association qualifié E, on crée une relation R qui comprend tous les attributs de E auxquels on ajoute l'identificateur de l'ensemble d'entités fortes/classe associé(e).
   Par exemple, après passage au relationnel des modèles des figures 7.10 et 7.11, la relation *Salle* aura comme clé primaire le couple $(Numero, \#BatimentID)$:
   `Salle(Numero,#BatimentID,Capacite)`.
3. Dans le cas de généralisation et de spécialisation/héritage, l'ensemble d'entités général/la classe mère E est représenté(e) par une relation R. Chaque ensemble d'entités E_i spécialisé/classe fille est représenté(e) par une relation R_i dans laquelle la clé primaire est de même domaine que l'identifiant de E et est une clé étrangère faisant référence à la clé primaire de E.
   Par exemple dans le cas des *Enseignant* et des *Etudiants* on aura :
   `Personen(PersonneID,Nom, Prenom, ...)`
   `Enseignant(#EnseignantID,Grade)`

`Etudiant(#EtudiantID,Adresse, Ville, CodePostal,AnneeEntree)`.
`EnseignantID` et `EtudiantID` sont clés primaires et clés étrangères, et font référence à la clé primaire de `Personne`.

7.2.2 Règles portant sur la transformation des ensembles d'associations

1. Pour chaque ensemble d'associations binaire A entre les ensemble d'entités/classes S et T (représenté(e)s respectivement par les relations RS et RT) tel que la cardinalité/multiplicité est $1,1$ dans le sens S vers T et $1,n$ ou $0,n$ (1..$*$ ou $*$ en UML) dans le sens T vers S, on inclut dans la définition de RS l'identifiant de S. Les attributs de A sont ajoutées à la définition de S.

   Par exemple, après passage au relationnel des schéma de modélisation des figures 7.12 et 7.13, le modèle relationnel contient deux relations :
   `Automobiliste(AutomobilisteId, Nom, Prénom)`
   `Voiture(Immatriculation, Marque, Type, Puissance, Année, #AutomobilisteId)` avec `#AutomobilisteId` une clé étrangère faisant référence à la clé primaire de `Automobiliste`. On aurait pu appeler cet attribut `ProprioID`.

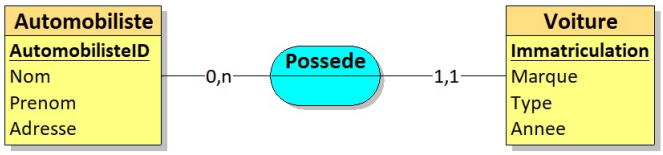

FIGURE 7.12 – Des exemples d'associations avec une cardinalité *0,n 1,1* entre une personne et une voiture en modélisation Entité/Association.

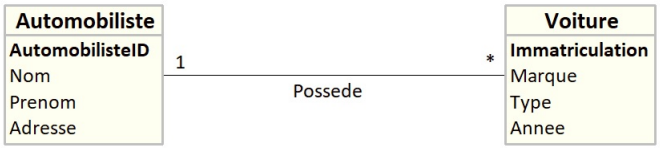

FIGURE 7.13 – Des exemples d'associations avec une multiplicité *1 ** entre une personne et une voiture en UML.

2. Pour chaque ensemble d'associations A de cardinalités $1,n$ ou $0,n$ (1..$*$ ou $*$ en UML) de chaque côté, on crée une nouvelle relation RA pour représenter l'association A. On met dans RA, les identificateurs de toutes les ensembles d'entités/classes participant à A. La clé primaire de RA est la concaténation de identificateurs. On ajoute également à RA tous les attributs définis sur A.

   Par exemple, dans les figures 7.14 et 7.15, le troisième ensemble d'associations (en bas) est transformé en trois relations :
   `Automobiliste(AutomobilisteId, Nom, Prénom)`
   `Voiture(Immatriculation, Marque, Type, Puissance, Année)`
   `Location(#AutomobilisteId, #Immatriculation, DateDebut, DateFin)`

ou `Location(LocID, #AutomobilisteId, #Immatriculation, DateDebut, DateFin)` avec une contrainte d'unicité sur le triplet (#AutomobilisteId,#Immatriculation,DateDebut).

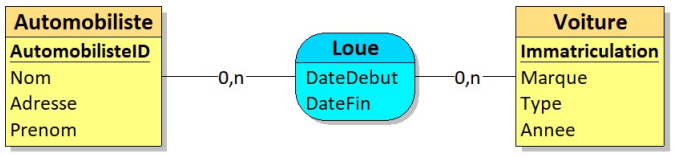

FIGURE 7.14 – Des exemples d'associations avec une cardinalité *0,n 1,1* entre une personne et une voiture en modélisation Entité/Association.

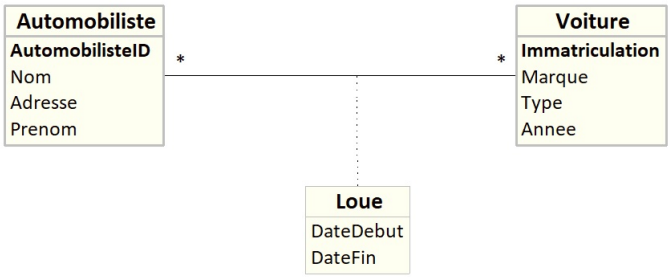

FIGURE 7.15 – Des exemples d'associations avec une multiplicité *1 ** entre une personne et une voiture en UML.

7.2.3 Dépendances fonctionnelles

Le passage au relationnel peut ajouter des dépendances fonctionnelles. Il ne faut pas oublier de définir les dépendances fonctionnelles de chaque relation. Une dépendance fonctionnelle ne porte que sur les attributs de la relation (pas entre attributs de différentes relations).

Par exemple, lorsque l'on a le schéma :
`Automobiliste(AutomobilisteId, Nom, Prénom)`
`Voiture(Immatriculation, Marque, Type, Puissance, Année, #ProprioID)` avec `#ProprioID` une clé étrangère faisant référence à la clé primaire de `Automobiliste`.

On a comme dépendances fonctionnelles associées à la relation `Voiture` : toutes les dépendances fonctionnelles associées à l'ensemble d'entités/classe `Voiture` à laquelle s'ajoute la dépendance fonctionnelle : $Immatriculation \longrightarrow \#ProprioID$ indiquant qu'une voiture a un et un seul propriétaire. La dépendance inverse en revanche n'est pas vraie, car un propriétaire peut posséder plusieurs voitures.

Bibliographie

[1] D. Austin, *Using Oracle8^{TM}*, Simple Solutions - Essential Skills, QUE, 1998, ISBN : 0-7897-1653-4

[2] R. Chapuis, *Oracle 8*, Editions Dunes et Laser, 1998, ISBN : 2-913010-07-5

[3] P. Chen, *The Entity-Relationship Model–Toward a Unified View of Data*, ACM Transactions on Database Systems, Vol. 1, No. 1, March 1976, Pages 9-36, http ://www.csc.lsu.edu/∼chen/pdf/erd.pdf

[4] T. Connolly, C. Begg et A. Strachan, *Database Systems - A pratical Approach to Design, Implementation and Management*, Addison-Wesley, 1998, ISBN : 0-201-34287-1, disponible à la BU 055.7 CON

[5] C.J. Date, *Introduction aux bases de données*, 6ème édition, Thomson Publishing, 1998, ISBN : 2-84180-964-1, disponible à la BU 005.7 DAT

[6] R. Elamsri et S.B. Navathe, *Fundamentals of Database Systems*, 3ème édition, Addison Wesley- disponible à la BU 005.7 ELM

[7] P. Delmal, *SQL2 - Application à Oracle, Access et RDB*, 2ème édition, Bibliothèque des Universités - Informatique, De Boeck Université, 1988, ISBN : 2-8041-2995-0,disponible à la BU 005.74 SQL

[8] S. Feuerstein, B. Pribyl et C. Dawes, *Oracle PL/SQL - précis et concis*, O'Reilly, 2000, ISBN : 2-84177-108-3

[9] G. Gardarin, *Bases de Données - objet & relationnel*, Eyrolles, 1999, ISBN : 2-212-09060-9, disponible à la BU 005.74 GAR

[10] R. Grin, *Introduction aux bases de données, modèle relationnel*, Université Sophia-Antipolis, jan. 2000

[11] R. Grin, *Langage SQL*, Université Sophia-Antipolis, jan. 2000

[12] G. Gardarin et O. Gardarin *Le Client-Serveur*, Eyrolles, 1999, disponible à la BU

[13] H. Garcia-Molina, J.D. Ulmann et J. Widow, *Database SYstem Implementation*, Prentice Hall, 2000, ISBN :0-13-040264-8, disponible à la BU 005.7 GAR

[14] H. Garcia-Molina, J.D. Ulmann et J. Widow, *Database Systems - The Complete Book*, Prentice Hall, 2002, ISBN :0-13-031995-3

[15] S. Krakowiak, *Gestion Physique des données*, Ecole Thématique "Jeunes Chercheurs" en Base de Données, Volume 2, Port Camargue, mars 1997

[16] D. Lockman, *Oracle8^{TM} Dévéloppement de bases de données*, Le programmeur - Formation en 21 jours, Editions Simon et Schuster Macmillan (S&SM), 1997, ISBN : 2-7440-0350-6, disponible à la BU 005.74 ORA

[17] P.J. Pratt, *Initiation à SQL - Cours et exercices corrigés*, Eyrolles, 2001, ISBN : 2-212-09285-7

[18] R. Ramakrishnan et J. Gehrke, *Database Management Systems*, Second Edition ; McGraw-Hill, 2000, ISBN : 0-07-232206-3, disponible à la BU 055.7 RAM

[19] A. Silberschatz, H.F. Korth et S. Sudarshan, *Database System Concepts*, Third Edition, McGraw-Hill, 1996, ISBN : 0-07-114810-8, disponible à la BU 005.7 DAT

[20] C. Soutou, *De UML à SQL - Conception de bases de données*, Eyrolles, 2002, ISBN : 2-212-11098-7

[21] J.D. Ullman et J. Widom, *A first Course in Database Systems*, Prentice Hall, 1997, ISBN : 0-13-887647-9, disponible à la BU 005.7 ULL