MERISE

Sommaire

1. INTRODUCTION

L'objectif de ce document est d'initier le lecteur au formalisme de modélisation des données tel que le définit la méthode MERISE.

Le besoin de méthodes

La conception d'un système d'information n'est pas évidente car il faut réfléchir à l'ensemble de l'organisation que l'on doit mettre en place. La phase de conception nécessite des méthodes permettant de mettre en place un modèle sur lequel on va s'appuyer. La modélisation consiste à créer une représentation virtuelle d'une réalité de telle façon à faire ressortir les points auxquels on s'intéresse.

Ce type de méthode est appelée *analyse*. Il existe plusieurs méthodes d'analyse, la méthode la plus utilisée en France étant la méthode **MERISE**.

Présentation de la méthode MERISE

MERISE est une méthode de conception, de développement et de réalisation de projets informatiques. Le but de cette méthode est d'arriver à concevoir un système d'information. La méthode MERISE est basé sur la séparation des données et des traitements à effectuer en plusieurs modèles conceptuels et physiques.

La séparation des données et des traitements assure une longévité au modèle. En effet, l'agencement des données n'a pas à être souvent remanié, tandis que les traitements le sont plus fréquemment.

Le modèle entité-association est apparu dans les travaux des chercheurs, entre 1972 et 1975 lors des travaux du français MOULIN puis de TARDIEU, TEBOUL... etc. Il a été rendu célèbre dans le monde entier par l'américain Peter CHEN, à la suite d'une publication intitulée "The Entity-Relationshionship Model" (ACM, Transaction on Database Systems, 1976).

La méthode MERISE date de 1978-1979, et fait suite à une consultation nationale lancée en 1977 par le ministère de l'Industrie dans le but de choisir des sociétés de conseil en informatique afin de définir une méthode de conception de systèmes d'information. Les deux principales sociétés ayant mis au point cette méthode sont le CTI (Centre Technique d'Informatique) chargé de gérer le projet, et le CETE (Centre d'Etudes Techniques de l'Equipement) implanté à Aix-en-provence.

Il existe des logiciels permettant de construire des schémas entités-associations et d'en analyser les conséquences logiques, puis de construire les tables associées aux modèles de manière entièrement automatique. Ces logiciels sont appelés AGL (atelier de génie logiciel) ou CASE suivant leur puissance. Les logiciels TRAMIS, AMC*Designor, SELECT... en sont des exemples.

A ce jour tous les spécialistes français et/ou latins du domaine de l'analyse orientée base de données se servent de ce modèle comme outil de communication des applications SGBDR. Il est présent de manière transparente ou plus visible, dans la plupart des logiciels de construction d'applications de bases de données comme ACCESS, PARADOX, ORACLE, SQL Server, Informix, Ingres, Sybase… Il n'est en revanche pas adapté aux bases de données purement objet comme O2 de Ardent Software… même si l'on admet la nouvelle dérive de MERISE orientée objet !

MERISE définit trois niveaux de description du système d'information :

- le niveau conceptuel,
- le niveau organisationnel,
- le niveau physique.

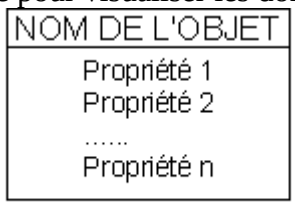
La représentation distincte des données et des traitements selon les 3 niveaux évoqués, conduit à l'élaboration de six modèles, mais ce document ne s'intéresse qu'à la modélisation des données au niveau conceptuel.
Le niveau conceptuel décrit les choix de gestion adoptés par l'entreprise. Schématiquement, ce niveau de description répond à la question « quoi ? », c'est-à-dire « que veut-on faire qui reste vrai quelles que soient les solutions d'organisation et les solutions techniques à mettre en œuvre ? ».
Ce document synthétise les différentes étapes nécessaires à la réalisation du modèle conceptuel des données (MCD). Il aborde ensuite les aspects liés à la normalisation qui permet la conception de bases de données cohérentes.

2. CONCEPTS DE BASE

Le formalisme utilisé pour décrire un MCD est celui du modèle entité-association. La représentation de ce formalisme s'appuie sur trois concepts de base :

- l'objet ou entité,
- l'association,
- la propriété.

L'objet est une entité ayant une existence propre. L'association est un lien ou relation entre objets sans existence propre. La propriété est la plus petite donnée d'information décrivant un objet ou une association.
La représentation graphique utilisée pour visualiser les données est la suivante :



Une propriété ne doit être présente que sur un seul objet ou une seule association.
Les ambiguités liées à la polysémie (un même nom de propriété désignant deux notions différentes) doivent être levées en nommant de façon précise les propriétés.
Les ambiguités liées à la synonymie (des noms différents de propriétés désignant une même notion) doivent être éliminées.
Un objet possède au moins une propriété.
Une association peut n'avoir aucune propriété.

3. AUTRES CONCEPTS

3.1. OCCURRENCE

L'occurrence d'une propriété est l'une des valeurs que peut prendre cette propriété.
L'occurrence d'un objet est l'un des ensembles d'occurrences de ses propriétés.
L'occurrence d'une association est l'une des liaisons entre occurrences d'objets participant à l'association.

Les propriétés d'un objet doivent avoir une occurrence quelle que soit l'occurrence de l'objet. Dans le cas contraire, il est nécessaire de créer un autre objet portant cette propriété.

Lorsqu'une propriété d'un objet peut avoir plusieurs occurrences pour une occurrence de l'objet, il faut :

- soit créer autant de propriétés sur cet objet qu'il y a de possibilités de valeurs pour cette propriété,
- soit créer un autre objet portant cette propriété et lier ce nouvel objet par une association avec l'objet initial.

De la même manière, lorqu'une propriété d'une association peut avoir plusieurs occurrences pour une occurrence de l'association, il faut :

- soit créer autant de propriétés sur cette association qu'il y a de possibilités de valeurs pour cette propriété,
- soit créer un autre objet portant cette propriété et mettre ce nouvel objet en liaison avec cette association.

3.2. IDENTIFIANT

L'identifiant d'un objet est la ou les propriétés permettant de déterminer de façon unique chacune des occurrences de l'objet. La valeur de l'identifiant doit être différente pour chaque occurrence de l'objet.

Pour repérer l'identifiant dans la représentation graphique d'un objet, le ou les propriétés constituant l'identifiant sont précédées d'un symbole (# ou *).

L'identifiant d'une association est la concaténation des identifiants des objets reliés.

3.3. DIMENSION D'UNE ASSOCIATION

La dimension d'une association est le nombre d'objets intervenant dans cette association.

Une association réflexive (de dimension 1) relie un objet à lui même.

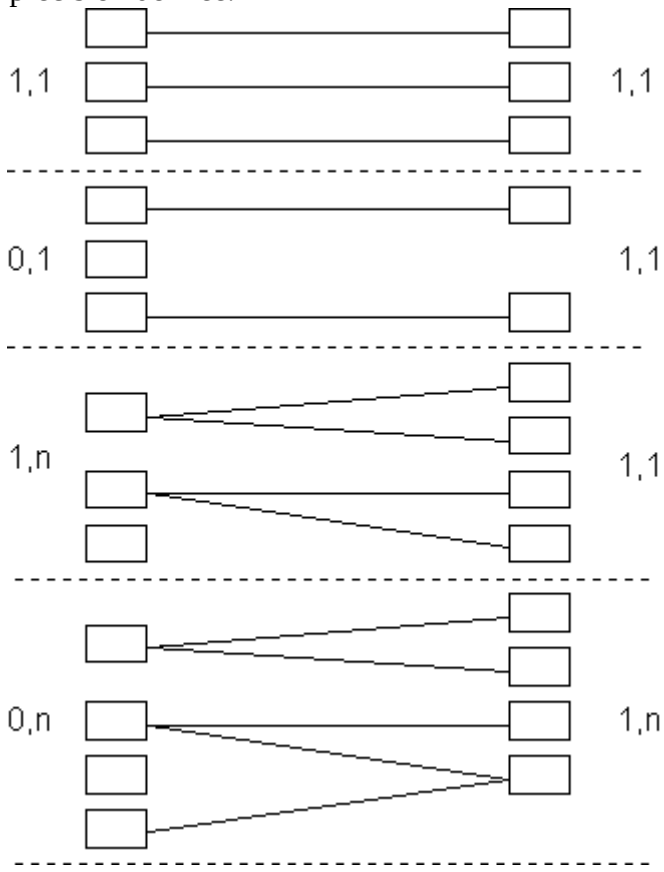
3.4. CARDINALITÉ

Les cardinalités d'un objet dans une association désignent le nombre minimum (0 ou 1) et le nombre maximum (1 ou n) de liens qu'il existe entre une occurrence de l'objet et une occurrence de l'association.

Une valeur minimum à 0 signifie qu'au moins une occurrence de l'objet n'est pas liée à l'association.
Une valeur minimum à 1 signifie que toutes les occurrences de l'objet sont liées à l'association.
Une valeur maximum à 1 signifie qu'aucune occurrence de l'objet n'est liée plus d'une fois à l'association.
Une valeur maximum à n signifie qu'au moins une occurrence de l'objet est liée plusieurs fois à l'association.

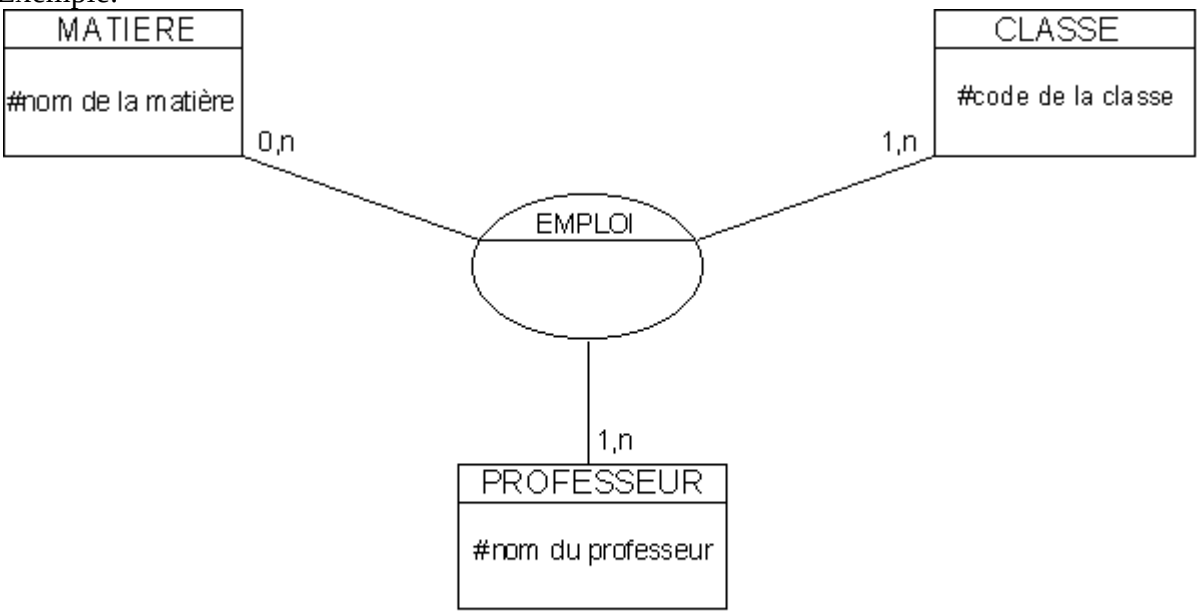
Á partir de ces valeurs minimum et maximum possibles, il existe quatre types de cardinalité :

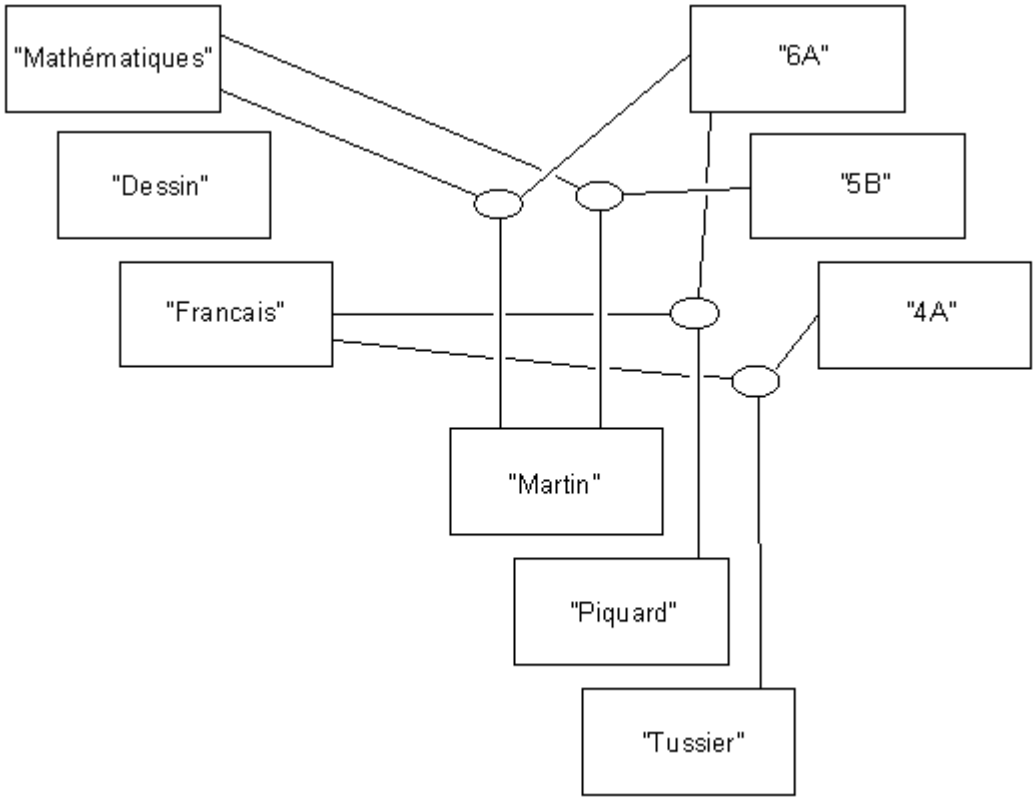
- (0,1) : une occurrence de l'objet n'est jamais liée plus d'une fois à l'association.
- (1,1) : une occurrence de l'objet est toujours liée une et une seule fois à l'association.
- (1,n) : une occurrence de l'objet est toujours liée au moins une fois à l'association.
- (0,n) : aucune précision donnée.



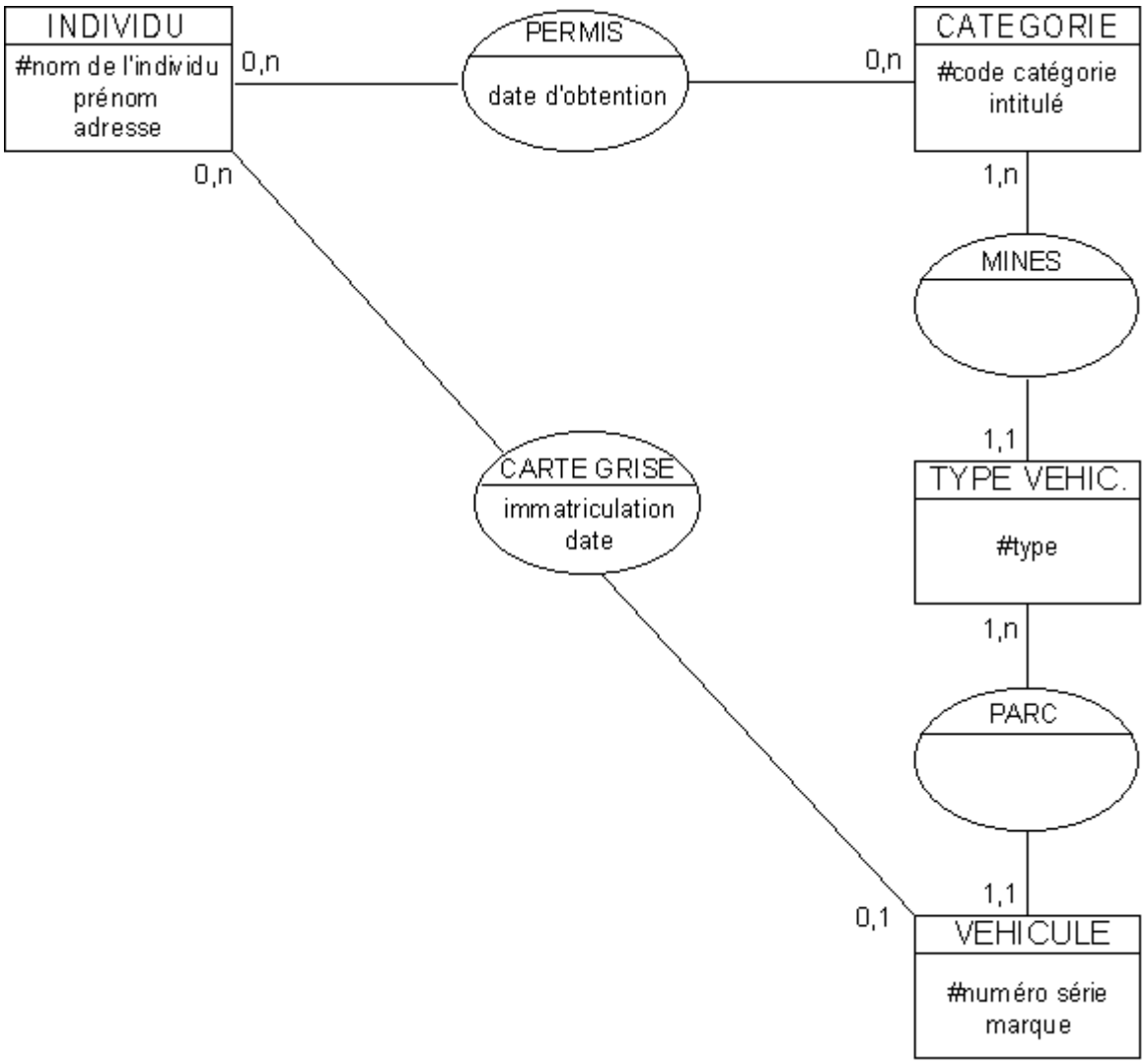
Une représentation schématique des liens entre les occurrences d'objets peut aider à déterminer les cardinalités d'une association.

Exemple:


MATIERE
#nom de la matière
0,n
CLASSE
#code de la classe
1,n
EMPLOI
1,n
PROFESSEUR
#nom du professeur


"Mathématiques"
"Dessin"
"Francais"
"6A"
"5B"
"4A"
"Martin"
"Piquard"
"Tussier"

4. EXEMPLE DE MCD



5. DÉPENDANCES

5.1. RAPPEL DU LANGAGE ALGÉBRIQUE

Une relation n-aire définie sur des ensembles D1, D2, ..., Dn est un sous-ensemble $\forall i, d_i \in D_i$ du produit cartésien D1 x D2 x ... x Dn. Une relation n-aire est un ensemble de n-uplets <d1, d2, ..., dn> oú .
Chacun des ensembles Di est appelé attribut de la relation. La relation est notée R(D1, D2, ..., Dn).

Dans le cadre d'une définition d'un modèle de données, les langages d'interrogation pour la recherche de données peuvent être scindés en deux classes :

- les langages algébriques,
- les langages prédicatifs.

Le principe d'un langage algébrique est de considérer que l'information à sélectionner peut s'exprimer sous forme d'une relation obtenue par applications successives d'opérateurs dont les opérandes sont les relations de base.
Ces opérateurs sont décrits dans la suite de ce paragraphe. Les langages prédicatifs ne sont pas abordés dans ce document.

Par convention, les premières lettres de l'alphabet sont utilisées pour désigner les attributs et les dernières lettres pour désigner les ensembles d'attributs.

5.1.1. Projection

La projection R[X] d'une relation n-aire R(X, Y) est définie par :
$< x > \in R[X] \Leftrightarrow \exists y \; tel \; que < x, y > \in R$

5.1.2. Anti-projection

L'anti-projection R]X[d'une relation n-aire R(X, Y) est définie par :
$< x > \in R]X[\Leftrightarrow \forall y, < x, y > \in R$

5.1.3. Produit cartésien

Le produit cartésien R x S de deux relations n-aires R(X) et S(Y) est définie par :
$< x, y > \in R \times S \Leftrightarrow < x > \in R \; et < y > \in S$

5.1.4. Jointure

La jointure (ou produit) R * S de deux relations n-aires R(X, Y) et S(Y, Z) est définie par :
$< x, y, z > \in R * S \Leftrightarrow < x, y > \in R \; et < y, z > \in S$

5.1.5. Thêta-jointure

La Thêta-jointure (ou Thêta-produit) $R *[A \Theta B] S$ de deux relations n-aires R(X, A) et S(B, Y) est définie par (Thêta est un opérateur de comparaison entre A et B) :

$< x, a, b, y > \in R *[A \Theta B] S \Leftrightarrow < x, a > \in R \; et < b, y > \in S \; et \; a \; \Theta \; b \; est \; vrai$

5.1.6. Sélection

La sélection R{F} de la relation n-aire R(X) par la formule logique F applicable sur les n-uplets de R, est définie par :

$< x > \in R \{F\} \Leftrightarrow < x > \in R(X) \; et \; F(x) \; est \; vrai$

La formule F est construite à partir des opérateurs de la logique : conjonction, disjonction, négation, égalité, inférieur, supérieur, différent.

5.1.7. Division

La division $R [A \div B] S$ de deux relations n-aires R(X, A) et S(B, Y) pour lesquelles les attributs A et B sont compatibles, est définie par :

$< x > \in R [A \div B] S \Leftrightarrow \forall b \in S[B], < x, b > \in R$

5.1.8. Dépendance fonctionnelle

Une dépendance fonctionnelle X $\rightarrow$ Y entre X et Y dans une relation n-aire R(X, Y, Z) est définie par :

$< x, y_1, z_1 > \in R \; et < x, y_2, z_2 > \in R \Rightarrow y_1 = y_2$

5.1.9. Décomposition

Une relation R(X1, X2, ..., Xn) est décomposable selon (X1, X2), (X2, X3, X4), ..., (Xn-3, Xn-2, Xn-1), (Xn-1, Xn) si

R = R[X1, X2] * R[X2, X3, X4] * ... * R[Xn-3, Xn-2, Xn-1] * R[Xn-1, Xn].

5.2. DÉPENDANCE FONCTIONNELLE (DF) ET CONTRAINTE FONCTIONNELLE (CIF)

5.2.1. Dépendance fonctionnelle entre propriétés d'un objet ou d'une association

Une propriété ou un ensemble de propriétés P2 dépend fonctionnellement d'une propriété ou d'un ensemble de propriétés P1, si la connaissance de la valeur de P1 détermine une et une seule valeur de P2.

Cette dépendance est notée P1 $\rightarrow$ P2 et se lit « P2 dépend fonctionnellement de P1 » ou « P1 détermine P2 par DF ».

Un identifiant détermine par DF toutes les autres propriétés de l'objet ou de l'association.

Une DF P1 $\rightarrow$ P2 est dite élémentaire si aucune partie de l'ensemble de propriétés P1 ne détermine la propriété ou l'ensemble de propriétés P2.

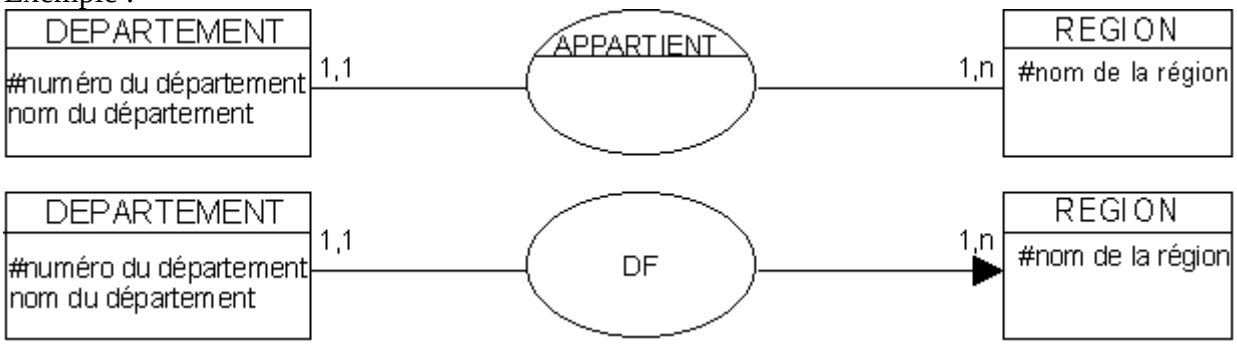
Les DF sont régies par la règle mathématique de transitivité : si P1 $\rightarrow$ P2 et P2 $\rightarrow$ P3 alors P1 $\rightarrow$ P3.

Une DF P1 $\rightarrow$ P2 est dite directe s'il n'y a pas de transitivité P1 $\rightarrow$ P3 et P3 $\rightarrow$ P2.

5.2.2. Dépendance fonctionnelle entre objets

Il existe une DF O1 $\rightarrow$ O2 entre deux objets O1 et O2 liés par une association, si chaque occurrence de O1 n'est associée qu'à au plus une occurrence de O2. En d'autres termes, la présence des cardinalités (0,1) ou (1,1) sur un objet, révèle une DF vers l'autre objet.
O1 est l'objet source de la DF et O2 l'objet but.
La DF est dite forte pour une cardinalité (1,1) et faible pour une cardinalité (0,1).

Exemple :



Une DF forte n'est pas porteuse de propriétés. Si une DF forte est porteuse de propriétés, alors ces propriétés doivent être déplacées sur l'objet source de la DF.
Une DF révèle une relation entre objets de type « est un ».

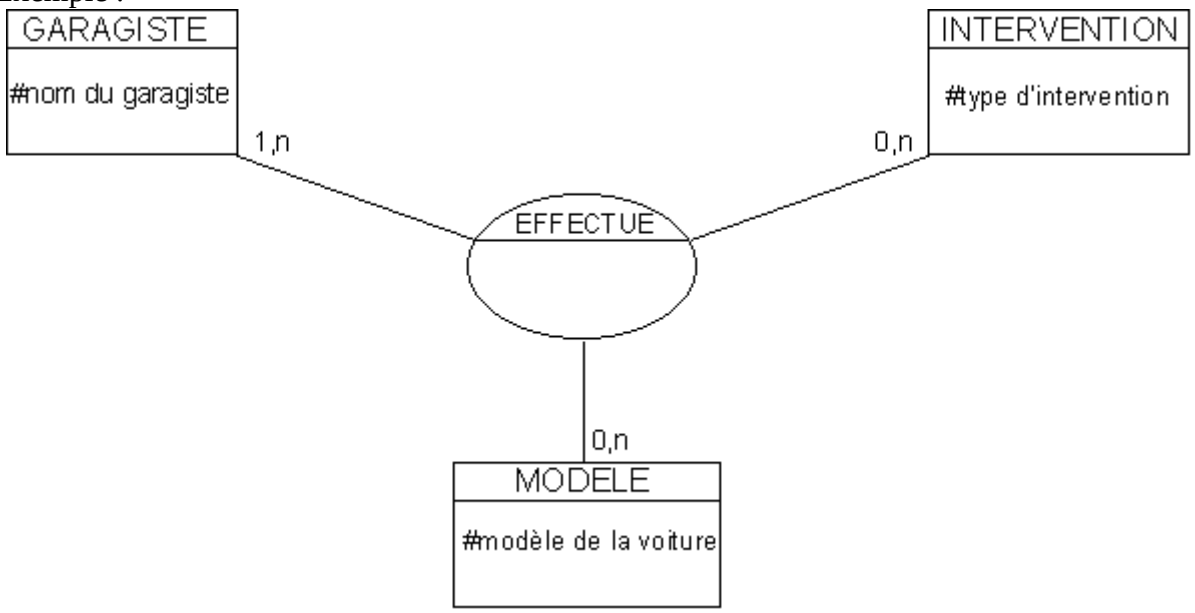
5.2.3. Contrainte d'intégrité fonctionnelle

Une contrainte d'intégrité fonctionnelle sur un objet signifie que cet objet est totalement identifié par la connaissance d'un ou plusieurs autres objets au sein d'une même association.

5.3. DÉPENDANCE MULTIVALUÉE (DM)

étant donné une association A(P) et X, Y des ensembles de propriétés inclus dans P, il existe une DM entre les ensembles de propriétés X, Y lorsque :
$< x, y_1, z_1 > \in A \; et < x, y_2, z_2 > \in A \Rightarrow < x, y_1, z_2 > \in A \; et < x, y_2, z_1 > \in A$
Cette dépendance est notée X $\rightarrow\!\!\!\rightarrow$ Y.
Remarque : $X \rightarrow\!\!\!\rightarrow Y \Rightarrow X \rightarrow\!\!\!\rightarrow P - (X \cup Y)$.

Exemple :



Les données de l'association EFFECTUE peuvent être :

garagiste	intervention	modèle
Martin	électricité	Citroën
Piquard	Carrosserie	Ford
Martin	Mécanique	Renault
Piquard	Carrosserie	Fiat
Tussier	Dépannage	Peugeot
Piquard	Alarme	Fiat
Martin	électricité	Renault
Piquard	Alarme	Ford
Martin	Mécanique	Citroën

Il existe une DM garagiste $\rightarrow\rightarrow$ intervention et une DM garagiste $\rightarrow\rightarrow$ modèle s'expliquant par le fait qu'un garagiste qui effectue un ensemble de types d'intervention pour un ensemble de modèles de voiture, est capable d'effectuer chacun de ces types d'intervention sur chacun de ces modèles de voiture.

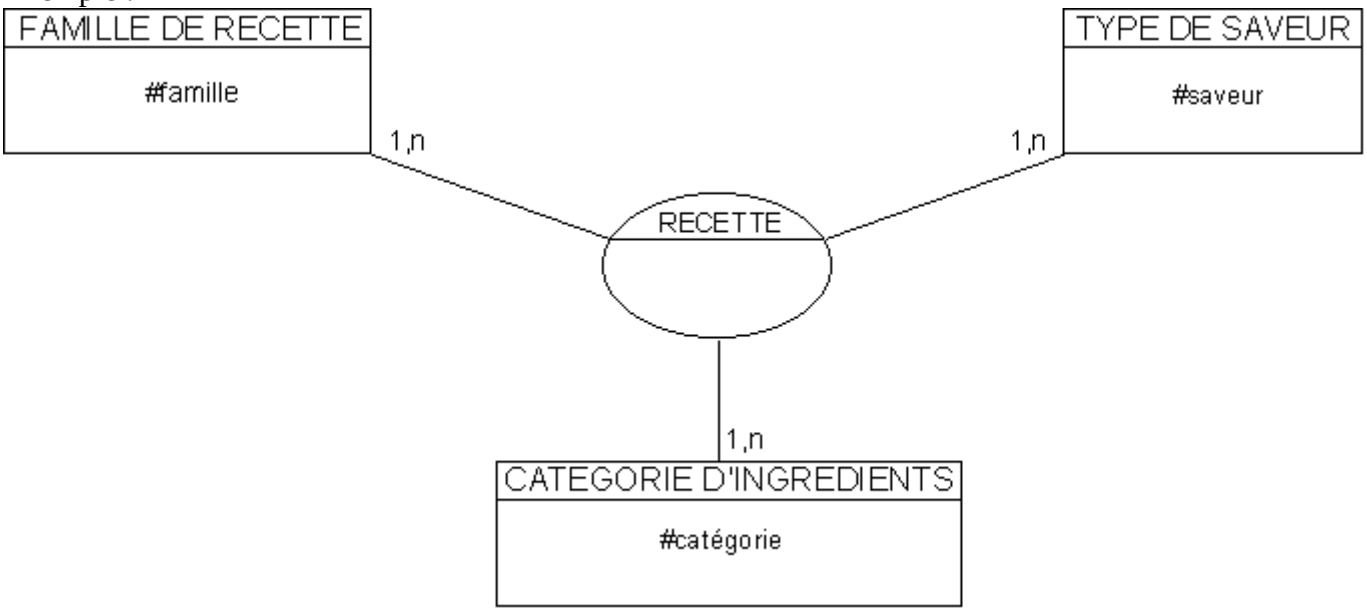
5.4. DÉPENDANCE DE JOINTURE (DJ)

étant donnée une association A(P) et X1, X2, ..., Xn des ensembles de propriétés dont l'union est P' tels que $P' \subset P, P' = \bigcup_{i=1}^{n} X_i \; et \; [X_i \cap X_{i+1} \neq \emptyset]_{i=1}^{n-1}$, il existe une DJ lorsque :
R = R[X1] * R[X2] * ... * R[Xn-1] * R[Xn].

Cette dépendance est notée *[X1][X2]...[Xn-1][Xn].

Si $P - P' \neq \emptyset$, la dépendance est dite partielle.
Si P = P', la dépendance est dite totale.

Exemple :



Les données de l'association RECETTE peuvent être :

famille	saveur	catégorie
Viandes	Sucré	Fruits
Viandes	Sucré	Légumes
Desserts	Sucré	Fruits
Desserts	Acide	Fruits

Ce cas correspond à X1 = (famille, saveur), X2 = (saveur, catégorie) et X3 = (catégorie, famille).
La projection RECETTE[X1] donne :

famille	saveur
Viandes	Sucré
Desserts	Sucré
Desserts	Acide

La projection RECETTE[X2] donne :

saveur	catégorie
Sucré	Fruits
Sucré	Légumes
Acide	Fruits

La jointure RECETTE[X1] * RECETTE[X2] donne :

famille	saveur	catégorie
Viandes	Sucré	Fruits
Viandes	Sucré	Légumes
Desserts	Sucré	Fruits
Desserts	Sucré	Légumes
Desserts	Acide	Fruits

La projection RECETTE[X3] donne :

catégorie	famille
Fruits	Viandes
Légumes	Viandes
Fruits	Desserts

La jointure RECETTE[X1] * RECETTE[X2] * RECETTE[X3] redonne l'association RECETTE.

Cette dépendance indique qu'il ne suffit pas qu'une famille de recettes ait une saveur dominante et que cette saveur corresponde à une catégorie d'ingrédients, pour qu'une recette à base de cet ingrédient et de saveur spécifiée, soit fabriquée.

Un dessert possède une saveur sucrée, une saveur sucrée peut correspondre à un légume mais un dessert sucré à base de légumes n'est pas fabriqué car il n'y a pas de légumes dans un dessert.

6. FORMES NORMALES

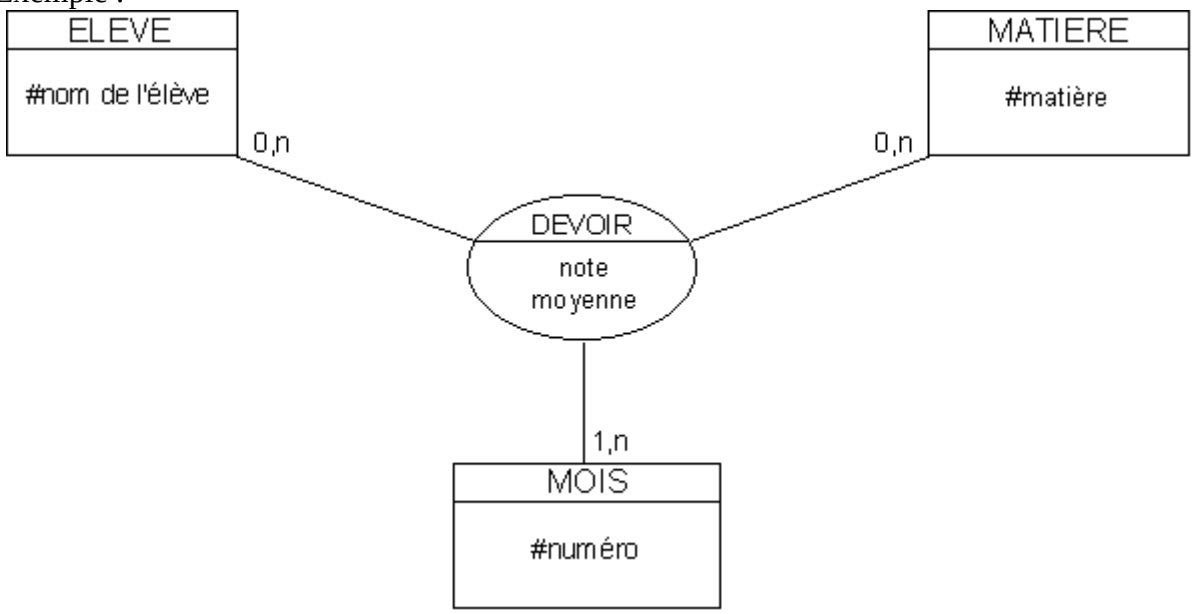
6.1. PREMIÈRE FORME NORMALE (1FN)

Un objet ou une association est en 1FN s'il possède un identifiant et si aucune propriété n'est à multiples valeurs.

6.2. DEUXIÈME FORME NORMALE (2FN)

Un objet ou une association est en 2FN s'il est en 1FN et si toutes les DF entre ses propriétés sont élémentaires.

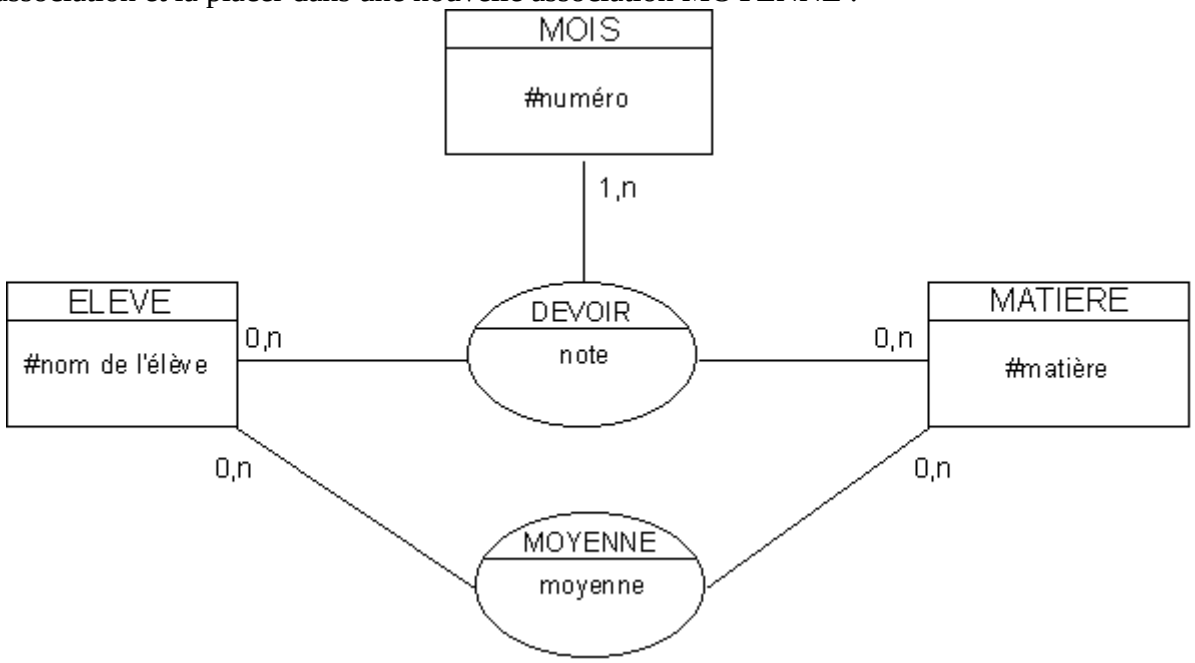
Exemple :



L'association DEVOIR est en 1FN mais pas en 2FN car :
#(nom, matière, numéro) $\rightarrow$ note,
#(nom, matière, numéro) $\rightarrow$ moyenne,
(nom, matière) $\rightarrow$ moyenne.

La deuxième DF n'est pas élémentaire.

Avec ce modèle, l'occurrence de la propriété « moyenne » sera répétée pour chacun des mois.
Pour que l'association DEVOIR soit en 2FN, il faut retirer la propriété « moyenne » de cette association et la placer dans une nouvelle association MOYENNE :

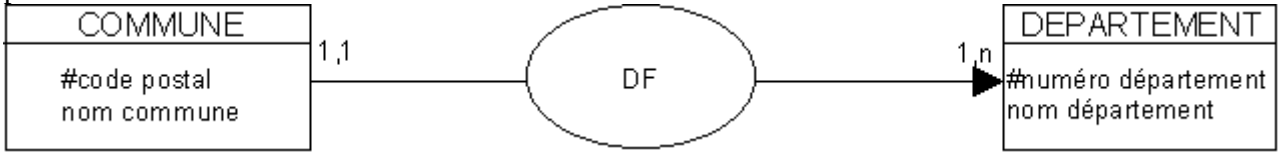


6.3. TROISIÈME FORME NORMALE (3FN)
Un objet ou une association est en 3FN s'il est en 2FN et si toutes les DF entre ses propriétés sont directes.
Exemple :

L'objet COMMUNE est en 2FN mais pas en 3FN car :
#code postal $\rightarrow$ nom commune,
#code postal $\rightarrow$ numéro département,
#code postal $\rightarrow$ nom département,
#numéro département $\rightarrow$ nom département.

La troisième DF n'est pas directe.

Avec ce modèle, la même occurrence de la propriété « nom département » sera répétée pour chaque commune appartenant au même département.
Pour que l'objet COMMUNE soit en 3FN, il faut créer un nouvel objet DEPARTEMENT pour refléter cette DF :

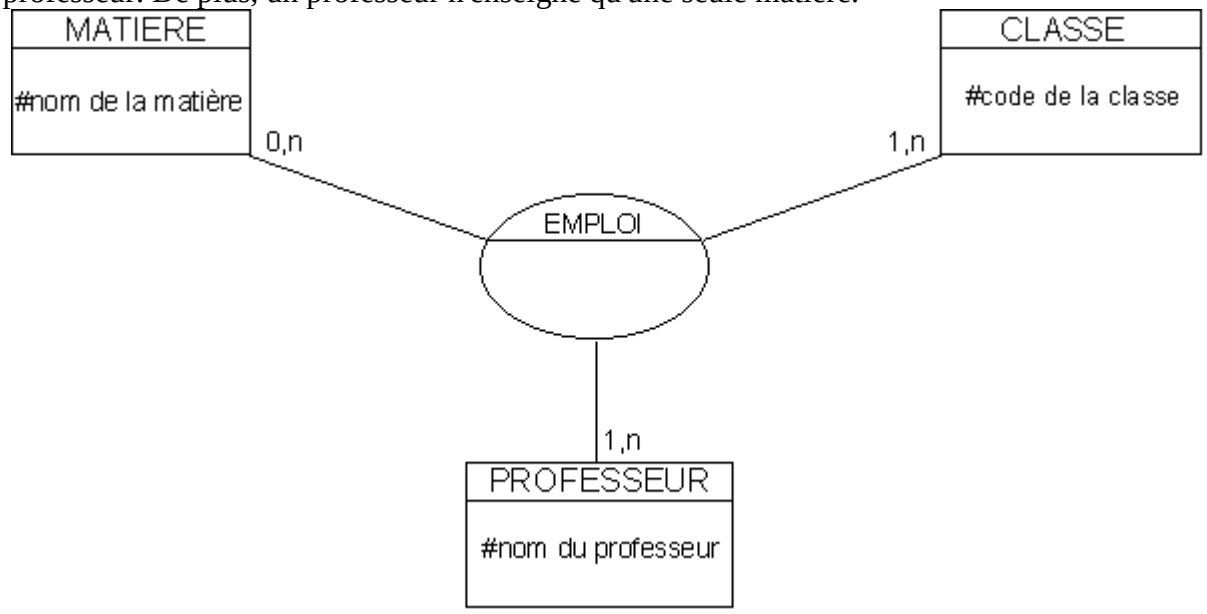


Ce modèle permet de plus de gérer l'ensemble des départements, y compris ceux qui ne sont pas rattachés à une commune.

6.4. FORME NORMALE DE BOYCE-CODD (BCFN)

Un objet ou une association est en BCFN s'il est en 3FN et si quand X $\rightarrow$ Y ($Y \not\subset X$) est vérifié alors X contient un identifiant de l'objet ou de l'association.

Exemple :
On suppose qu'une matière n'est enseignée qu'une seule fois dans une classe et que par un seul professeur. De plus, un professeur n'enseigne qu'une seule matière.

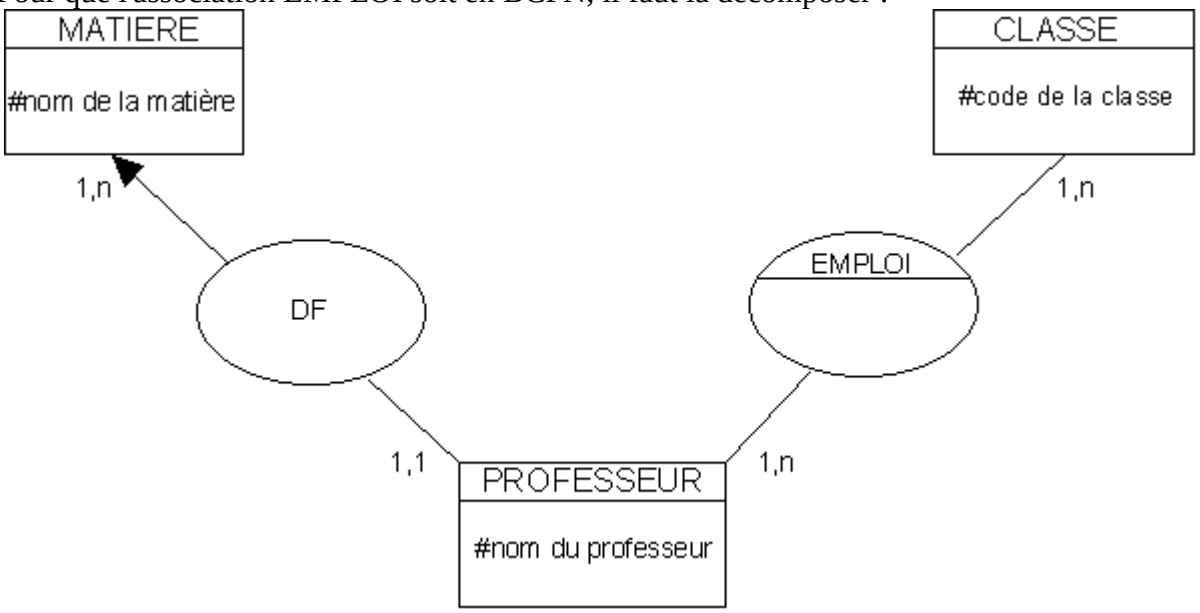


L'association EMPLOI est en 3FN mais pas en BCFN car :
(matière, classe) $\rightarrow$ professeur
professeur $\rightarrow$ matière

Il existe donc des DF dont la source ne contient pas l'identifiant de l'association.

Ce modèle n'interdit pas l'existence des occurrences du type <« Mathématiques », « 6A », « Martin »> et
<« Mathématiques », « 6A », « Piquard »>, et ceci est en contradiction avec la première contrainte.

Pour que l'association EMPLOI soit en BCFN, il faut la décomposer :

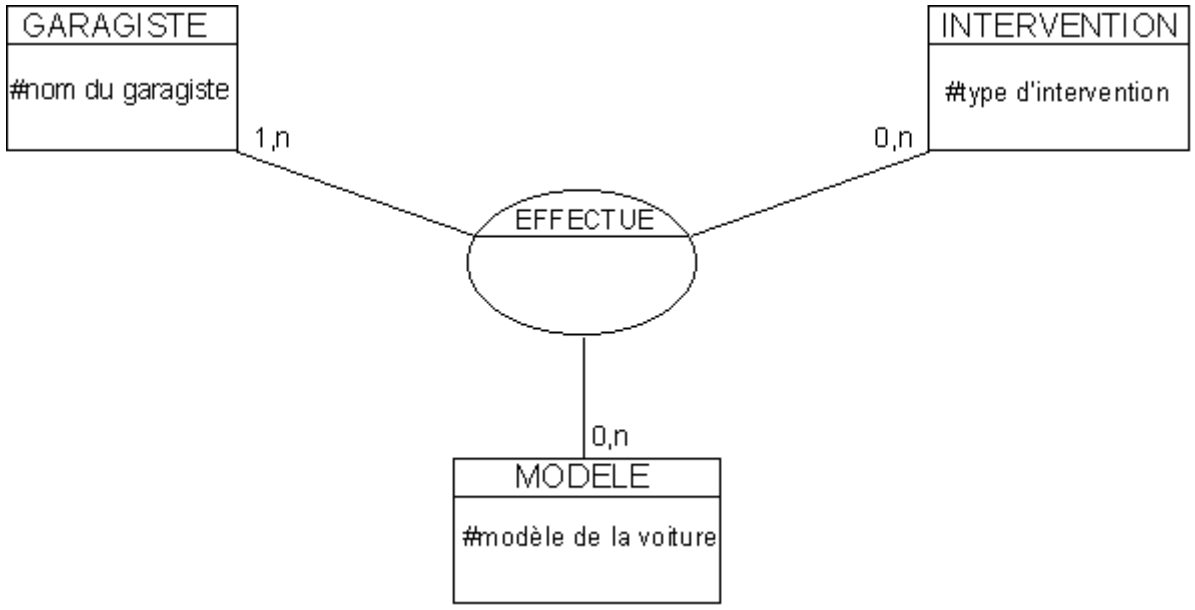


6.5. QUATRIÈME FORME NORMALE (4FN)

Une association est en 4FN si elle est en BCFN et si elle ne possède pas de DM ou bien, $X \rightarrow\!\!\!\!\rightarrow Y$ étant la DM, il doit exister une propriété A telle que $X \rightarrow A$ soit verifiée.

Exemple :

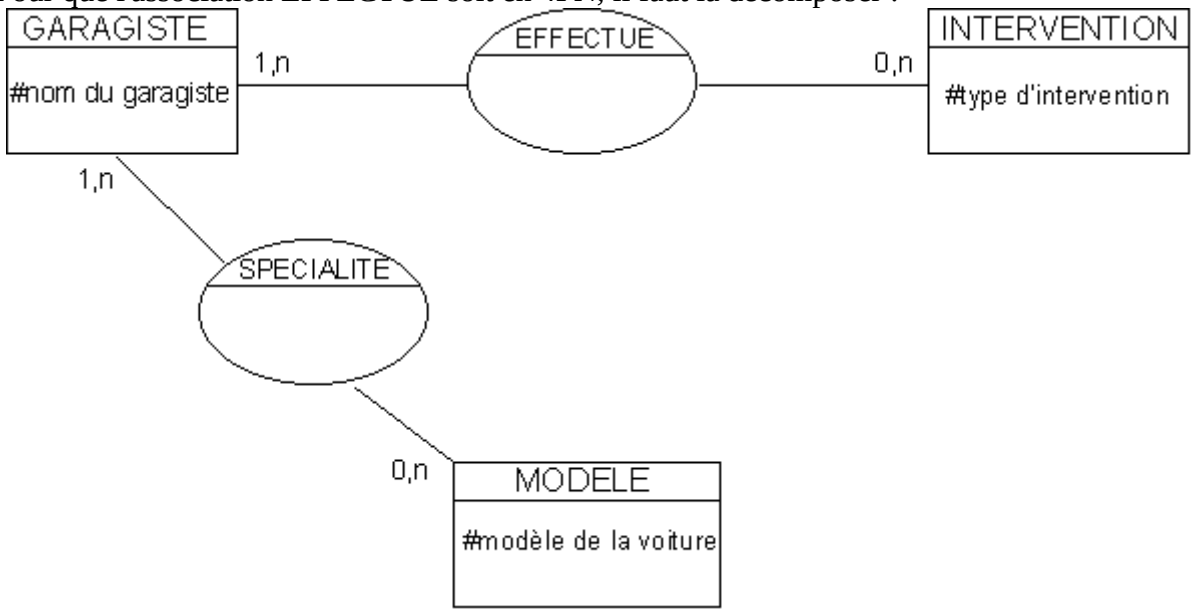
On suppose qu'un garagiste qui effectue un ensemble de types d'interventions pour un ensemble de modèle de voiture, est capable d'effectuer chacun de ces types d'interventions sur chacun de ces modèles de voitures.



L'association EFFECTUE est en BCFN mais pas en 4FN car il existe une DM (voir au paragraphe 5.3) et, puisque cette association n'est pas porteuse d'une propriété, il n'existe pas de DF.

Ce modèle présente des difficultés de mise à jour. Si l'occurrence <« Tussier », « Pare-brise », « Seat »> est ajoutée, il faut ajouter aussi les occurrences <« Tussier », « Dépannage », « Seat »> et <« Tussier », « Pare-brise », « Peugeot »>.

Pour que l'association EFFECTUE soit en 4FN, il faut la décomposer :

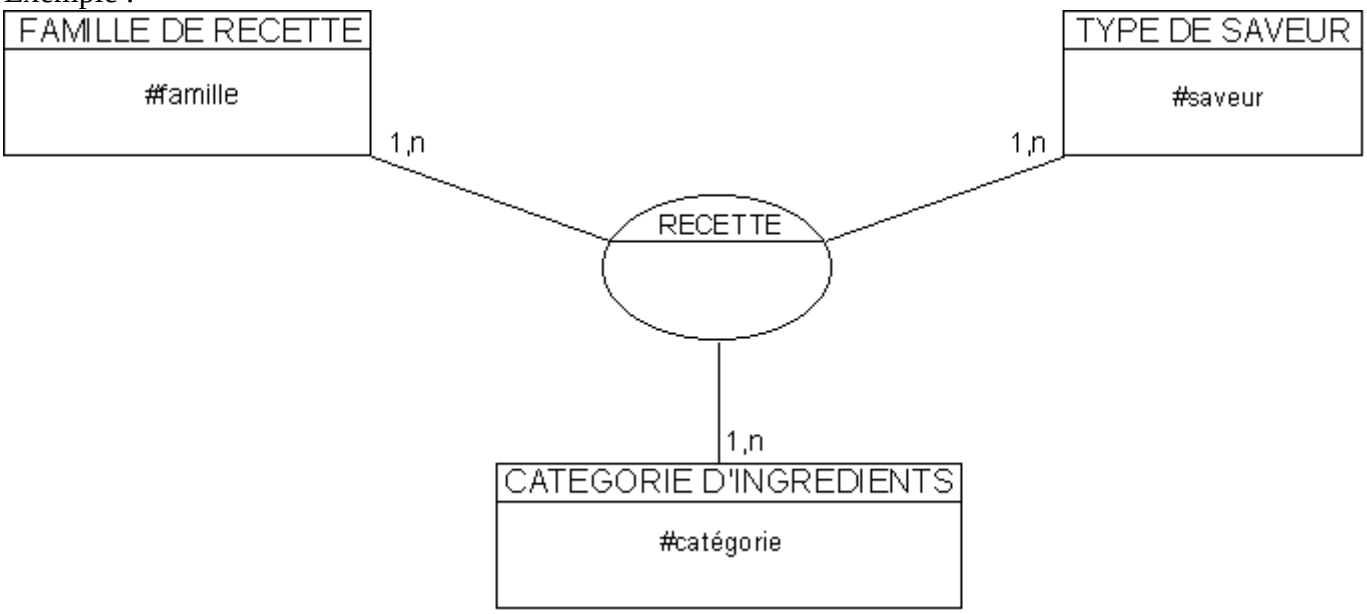


Si l'association EFFECTUE avait été porteuse d'une propriété « temps de réparation », elle aurait été en 4FN puisque la DF #(garagiste, intervention, modèle) $\rightarrow$ temps aurait existé.

6.6. CINQUIÈME FORME NORMALE (5FN)

Une association est en 5FN si elle est en 4FN et si elle ne possède pas de DJ ou bien, *[X1]…[Xn] étant la DJ, il doit exister une propriété A telle que X $\rightarrow$ A soit verifiée.

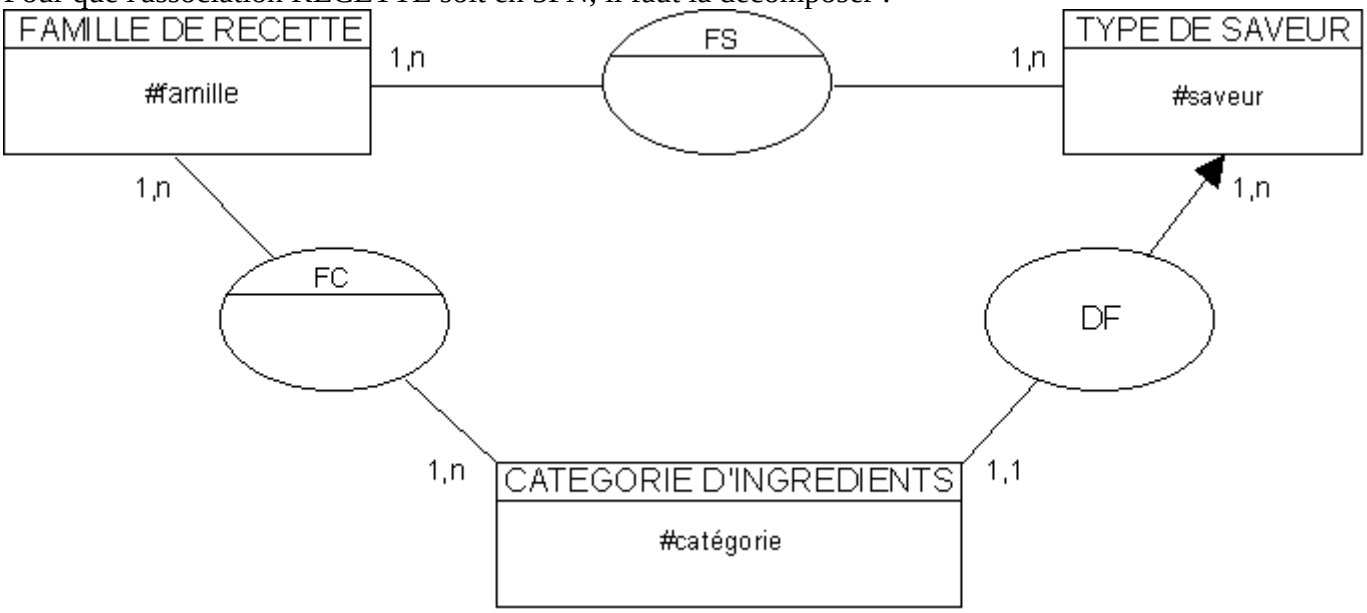
Exemple :



L'association RECETTE est en 4FN mais pas en 5FN car il existe une DJ (voir au paragraphe 5.4) et, puisque cette association n'est pas porteuse d'une propriété, il n'existe pas de DF.

Ce modèle présente des difficultés de mise à jour. Si l'occurrence <« Viandes », « Acide », « Vins »> est ajoutée, il faut également ajouter l'occurrence <« Viandes », « Acide », « Fruits »>.

Pour que l'association RECETTE soit en 5FN, il faut la décomposer :



7. ÉLABORATION D'UN MCD

Soit l'énoncé suivant :

La progression d'une recette donne les différents stades qui permettent la réalisation du travail. Cette progression est un enchaînement successif et logique des diverses manipulations qui concourent à l'aboutissement du plat à réaliser. Chacune de ces manipulations fait apparaître un temps d'exécution ou de cuisson. L'application de ces manipulations dans l'ordre spécifié conduit à la réalisation de la recette. Les termes techniques de base permettent de définir la manipulation à effectuer pour chacune des phases de la progression : décanter, émincer, éplucher, flamber, hacher, peler, trier, rôtir, … Les termes techniques employés dans une progression supposent l'utilisation du matériel adéquate. Le matériel est regroupé par catégorie : couteaux, plaques, ustensiles de cuisson, robots, … Un même matériel peut participer à plusieurs phases de la progression. Plusieurs matériels peuvent intervenir dans une même phase de la progression. Une quantité est indiquée pour chaque matériel utilisé dans une manipulation. Une recette est constituée à partir d'une liste d'ingrédients auxquels sont associés une quantité à utiliser. Les ingrédients sont répartis par type : légumes, fruits, poissons, boucherie, charcuterie, épicerie, boulangerie, cave, … Un ingrédient n'est répertorié que dans un seul de ces types. Chaque ingrédient possède une saveur. Les recettes sont regroupées par famille : hors d'œuvre, potages, poissons, entrées, rôtis, entremets, oeufs, viandes, … Une recette n'est repertoriée que dans une seule de ces familles.

Il s'agit d'élaborer le MCD correspondant à cet énoncé.

La lecture de ce texte permet de dégager les propriétés suivantes :

- type d'ingrédient,
- ingrédient,
- saveur,
- famille de recette,
- recette,
- catégorie de matériel,
- matériel,
- terme technique,
- quantité d'ingrédient,
- temps d'exécution ou de cuisson,
- numéro d'ordre de la manipulation dans la progression,
- quantité de matériel.

D'autres propriétés peuvent être éliminées car elles sont synonymes de celles précédemment citées :

- progression,
- stade,
- phase,
- manipulation,
- enchaînement de manipulations.

Une progression n'est autre qu'une recette. Un stade ou une phase dans une progression correspond à un terme technique auquel a été rajouté un numéro d'ordre. Une manipulation est un terme technique. Un enchaînement de manipulations est une progression.

La notion de quantité employée dans cet énoncé est ambiguë (polysémie) puisqu'elle fait référence à la quantité d'ingrédients et à la quantité de matériels. Cette dernière propriété est renommée :

- nombre de spécimens.

Á partir de ces propriétés une liste de DF peut être faite :

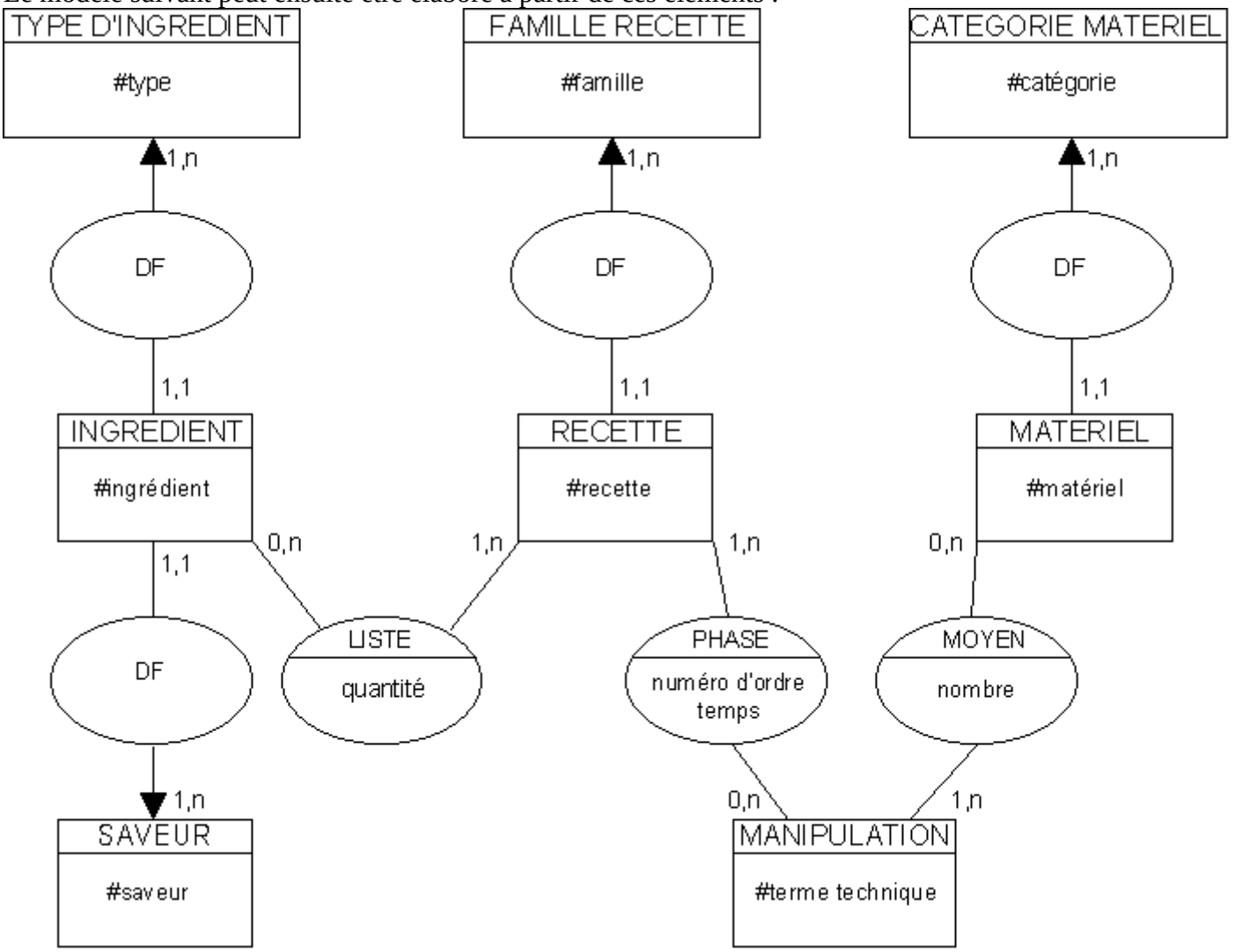
- ingrédient $\rightarrow$ type,
- recette $\rightarrow$ famille,
- matériel $\rightarrow$ catégorie,
- ingrédient $\rightarrow$ saveur,
- (recette, terme technique) $\rightarrow$ numéro d'ordre,
- (recette, terme technique) $\rightarrow$ temps d'exécution,
- (recette, ingrédient) $\rightarrow$ quantité d'ingrédient.

Il manque toutefois des informations sémantiques à cet énoncé, en particulier pour le nombre de spécimens de matériel à utiliser dans une manipulation. Ce nombre est-il dépendant de la recette ou bien est-il toujours le même à partir du moment oú la manipulation apparait dans une recette ?
Si l'on suppose que ce nombre est indépendant de la recette, la DF suivante est reconnue :
(matériel, terme technique) nombre de spécimens.

Les propriétés et les DF associées permettent de concevoir les objets suivants :

- type d'ingrédient,
- ingrédient,
- famille de recette,
- recette,
- catégorie de matériel,
- matériel,
- manipulation,
- saveur.

Le modèle suivant peut ensuite être élaboré à partir de ces éléments :



Ce modèle est en 5FN. Il présente tout de même une anomalie sur le numéro d'ordre dans l'association PHASE. En effet, rien n'indique dans ce schéma que la numérotation est séquentielle.

8. CONTRAINTES SÉMANTIQUES

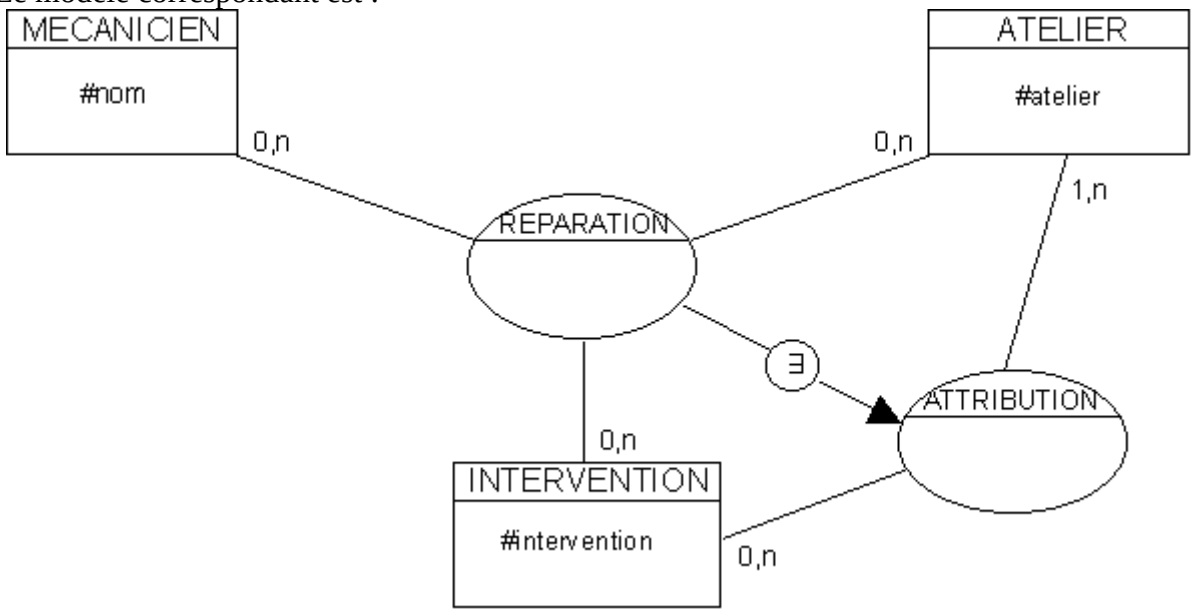
8.1. CONTRAINTES ENTRE ASSOCIATIONS

8.1.1. Contrainte d'existence

Ce type de contrainte fait dépendre l'existence d'une occurrence d'une association entre des occurrences d'objets, de l'existence d'une occurrence d'une autre association reliant ces objets ou une partie de ces objets.

Exemple :
Un mécanicien n'effectue des réparations sur des voitures qu'à l'atelier qui correspond au type d'intervention à réaliser.

Le modèle correspondant est :

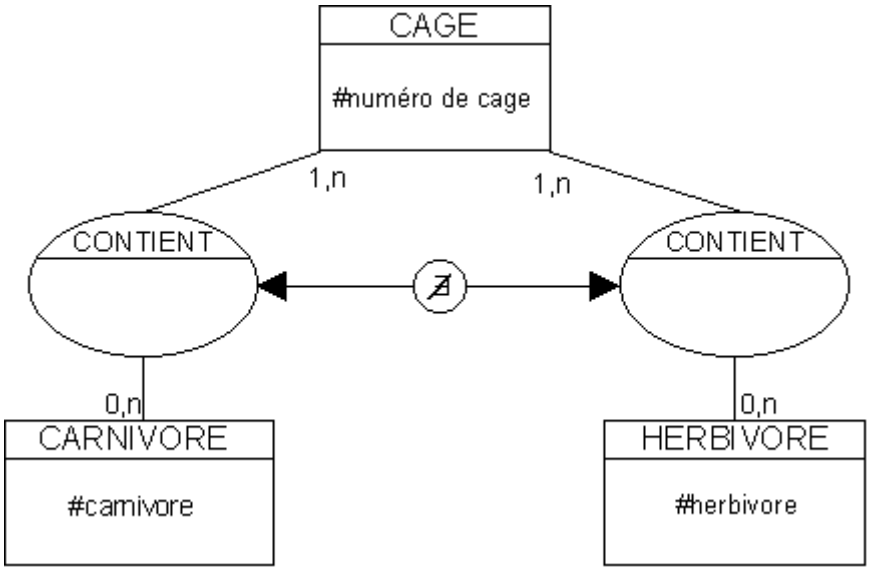


Il faut pouvoir indiquer dans ce modèle que l'occurrence <« Martin », « A01 », «Carrosserie»> n'est possible pour l'association REPARATION que si l'occurrence <« A01 », «Carrosserie»> existe pour l'association ATTRIBUTION.

8.1.2. Contrainte de non-existence

C'est l'inverse du type de contrainte précédent. L'existence d'une occurrence d'une association entre des occurrences d'objets doit dépendre de la non-existence d'une occurrence d'une association qui pourrait relier ces objets ou une partie de ces objets.

Exemple :

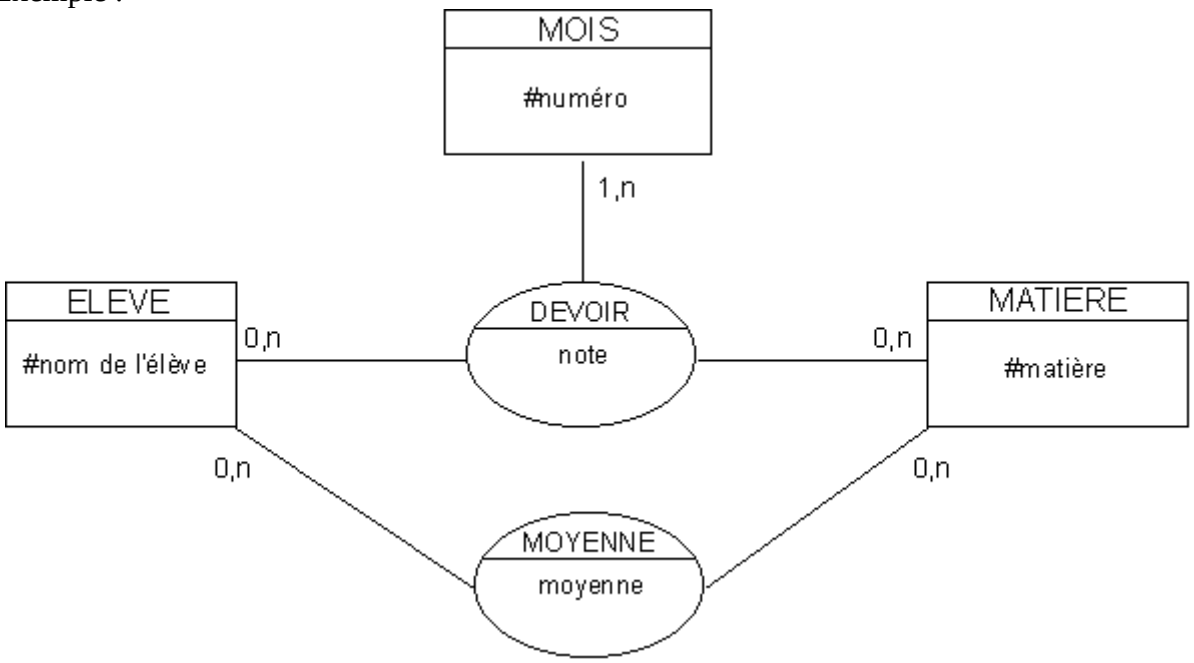


Un livre peut être un roman ou une étude mais pas les deux.

8.2. CONTRAINTES ENTRE PROPRIÉTÉS

Ces contraintes correspondent à des règles de gestion imposées par le système d'information. Elles ne sont en général pas indiquées sur la représentation graphique du modèle mais font plutôt partie des annexes.
Leur description n'est pas formalisée. Il ne s'agit pour ces contraintes que de faire apparaitre sous une forme syntaxique quelconque, une règle de calcul importante.

Exemple :



La moyenne d'un élève dans une matière est obtenue en divisant par le nombre de notes obtenues par cet élève dans cette matière, la somme des notes obtenues par cet élève dans cette matière.
Cette contrainte s 'exprime par :

$Pour\ e \in ELEVE\ et\ m \in MATIERE,$

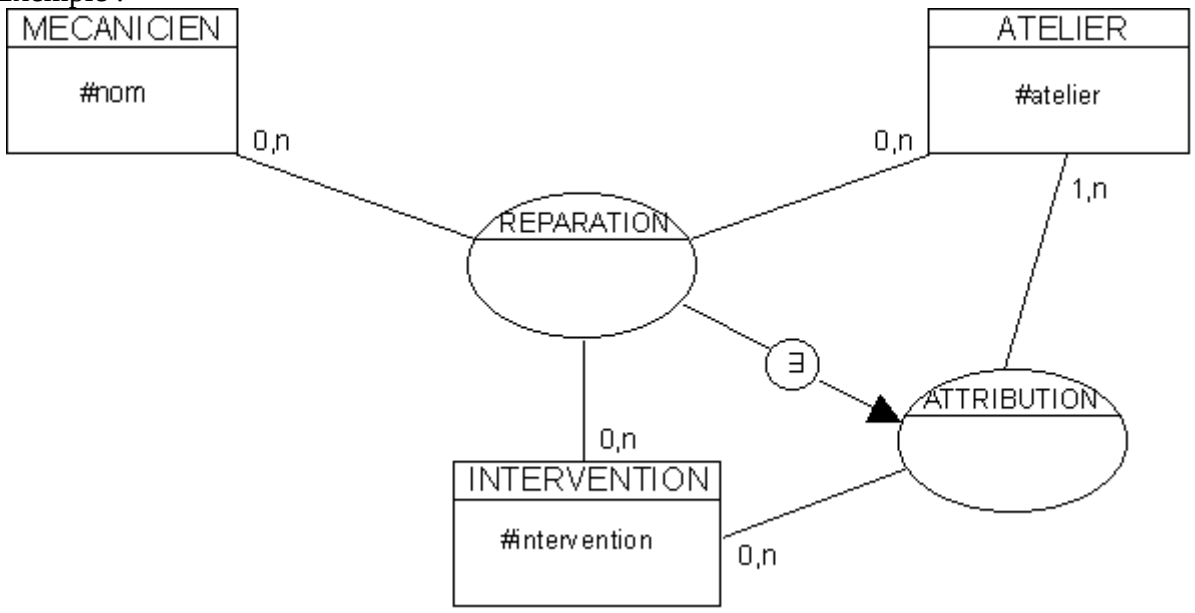
$[< e, m >.moyenne\ +=\ < e, m, n >.note\ et\ i\ +=\ 1]_{\forall\ <e,m,n>\ \in DEVOIR}$

$< e, m >.moyenne\ /=\ i$

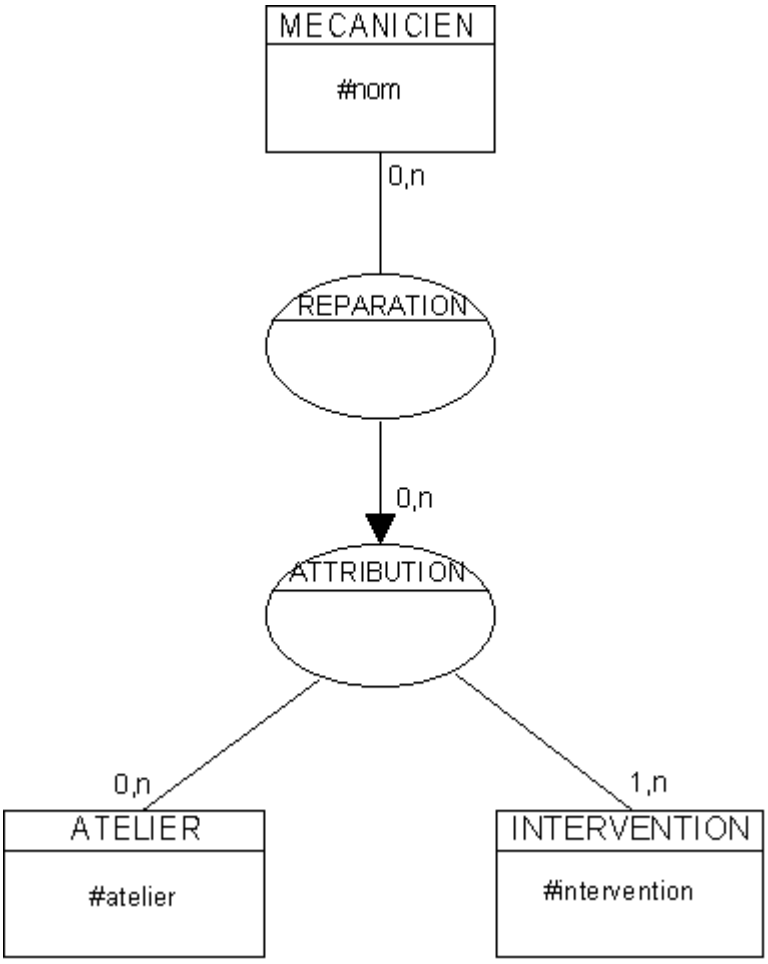
8.3. ASSOCIATION D'ASSOCIATIONS

Le concept d'association tel qu'il a été défini au paragraphe 2, est modifié pour prendre en compte le cas oú des contraintes apparaissent entre des associations d'objets.
Cette extension au concept initial permet d'exprimer différemment les contraintes entre associations vues au paragraphe 8.1.
Ce principe est limité au cas oú l'identifiant de l'une des associations est contenu dans l'identifiant de l'autre association.

Exemple :



La contrainte d'existence entre les deux associations REPARATION et ATTRIBUTION peut être remplacé par une association d'associations :



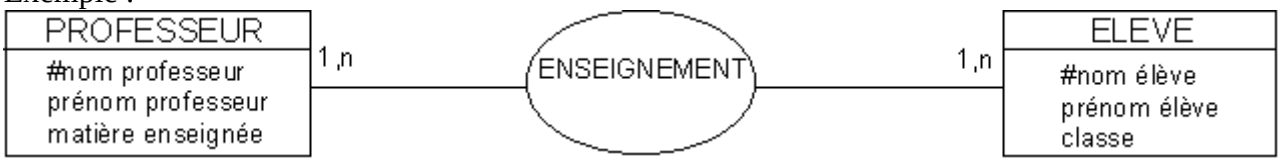
L'orientation du lien entre REPARATION et ATTRIBUTION précise quelle est l'association dont l'identifiant est contenu dans l'identifiant de l'autre association : l'association REPARATION est identifiée par #(mécanicien, ATTRIBUTION) c'est-à-dire #(mécanicien, intervention, atelier).

La cardinalité 0,n sur le lien REPARATION-ATTRIBUTION permet de spécifier que certaines occurrences de l'association ATTRIBUTION ne sont pas liées à une occurrence de l'association REPARATION.
En d'autre terme, un atelier peut être spécialisé dans une intervention sans qu'il y ait un garagiste qui y travaille.

9. TRADUCTION EN SCHÉMA RELATIONNEL

Chaque propriété d'un objet ou d'une association devient un attribut d'une relation.
Chaque objet devient une relation dont la clé est l'identifiant de l'objet.
Pour une DF forte, l'identifiant de l'objet but est ajouté à la relation définie à partir de l'objet source.
Une association devient une relation dont la clé est l'identifiant de l'association et à laquelle ont été ajoutés tous les attributs issus des propriétés de l'association.

Exemple :



Le schéma relationnel correspondant à ce modèle est le suivant :

individu(#nom, prénom, adresse)
catégorie(#code_catégorie, intitulé)
permis(#nom, #code_catégorie, date_obtention)
type_vehicule(#type, code)
véhicule(#numéro_série, marque, type)
carte_grise(#nom, #numéro_série, immatriculation, date)

10. EXTENSION POUR LA CONCEPTION PAR OBJETS

10.1. RAPPEL SUR LA CONCEPTION ORIENTÉE OBJETS

La conception orientée objets est une méthode qui conduit à l'élaboration d'une architecture basée sur les objets manipulés par le système étudié.

Cette méthode est un processus de conception ascendante. Elle favorise l'élaboration de composants logiciels autonomes et cohérents pouvant être utilisés pour constituer de nouveaux systèmes. Ce processus de conception conduit à l'élaboration d'un système modulaire sous réserve que les principes de modularité soient respectées.

La méthode ascendante est à opposer à la méthode descendante dont le principe est la décomposition d'un système en unités fonctionnelles fondées sur les traitements. Cette méthode ne permet pas de constituer des éléments logiciels réutilisables.

Faire la différence entre ces deux approches c'est faire la distinction entre les deux questions : « sur quoi agit le système ? » et « que fait le système ? ».

10.1.1. Abstraction de données

Une abstraction de données est un principe qui permet de séparer le comportement de la structure interne d'un objet.
Le comportement d'un objet est défini par les services que cet objet peut offrir à l'extérieur et les opérations qu'il peut effectuer.
La structure interne de l'objet et la mise en œuvre du comportement sont cachées de l'extérieur par encapsulation. L'encapsulation occulte les détails physiques par masquage de l'information.

10.1.2. Classes

Une classe est la description d'un ensemble d'objets ayant des propriétés communes. Chaque classe peut être membre d'une hiérarchie de classes. Cette hiérarchie relie les classes par des relations de nature différentes.
L'interface d'une classe donne une vue externe de la classe c'est à dire une déclaration des opérations applicables sur les objets de cette classe. Elle permet donc de savoir ce que peuvent offrir ou faire les objets de cette classe sans détails de mise en œuvre.
L'interface d'une classe est composée de primitives : attribut (composant ou propriété d'un objet), procédure (action sur l'état d'un objet) ou fonction (retour d'une valeur calculée sur l'état d'un objet). L'état d'un objet correspond aux valeurs courantes de chacune de ses propriétés.

10.1.3. Relations entre les classes

Utilisation
Cette relation traduit une collaboration de type client/fournisseur. Cela signifie qu'une classe utilise une autre classe pour réaliser le comportement de ses objets.

Agrégation
Cette relation traduit une dépendance de type composée/composantes. Elle indique qu'une classe est responsable des objets d'une autre classe. Au sens usuel, elle peut se traduire par &"171; contient » ou « est composée de ».

Association
Cette relation traduit un lien bidirectionnel représentant une dépendance sémantiques entre objets.

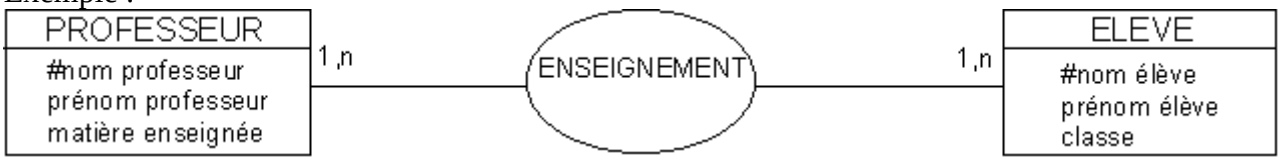
Héritage
Cette relation traduit un partage de comportement et de propriétés entre classes. Elle exprime une généralisation / spécialisation entre classes c'est à dire le fait qu'une classe spécialise une autre plus générale. Elle peut se traduire usuellement par « est un ».

Instanciation
Cette relation traduit une génération d'objets d'une classe par une autre classe.

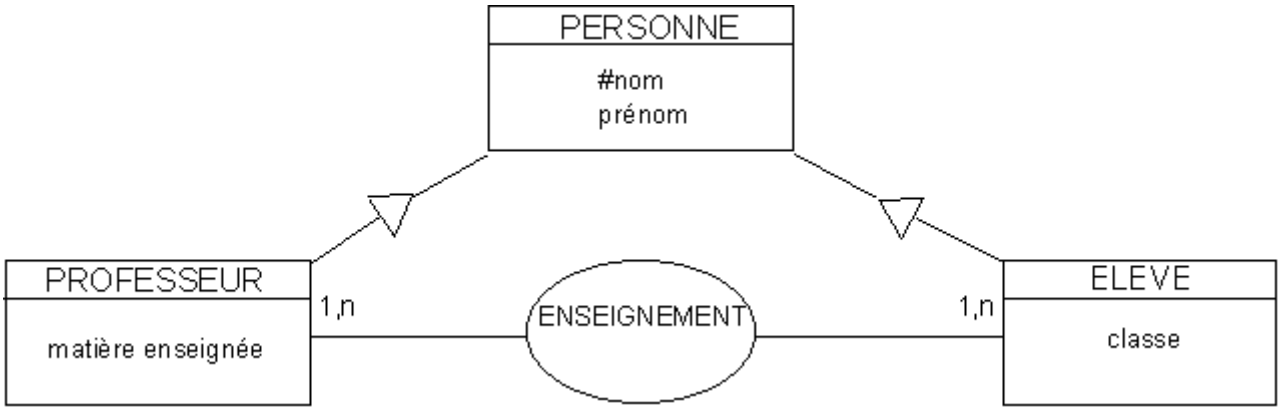
10.2. GÉNÉRALISATION / SPÉCIALISATION
La relation d'héritage définie entre classes est introduite dans le modèle conceptuel par généralisation ou spécialisation des objets.

La première méthode consiste à spécialiser un objet en créant un ou plusieurs objets spécialisés.
La seconde méthode est inverse. Elle consiste à généraliser un ou plusieurs objets en créant un objet générique. L'objet générique porte toutes les propriétés communes aux objets spécialisés.

Exemple :

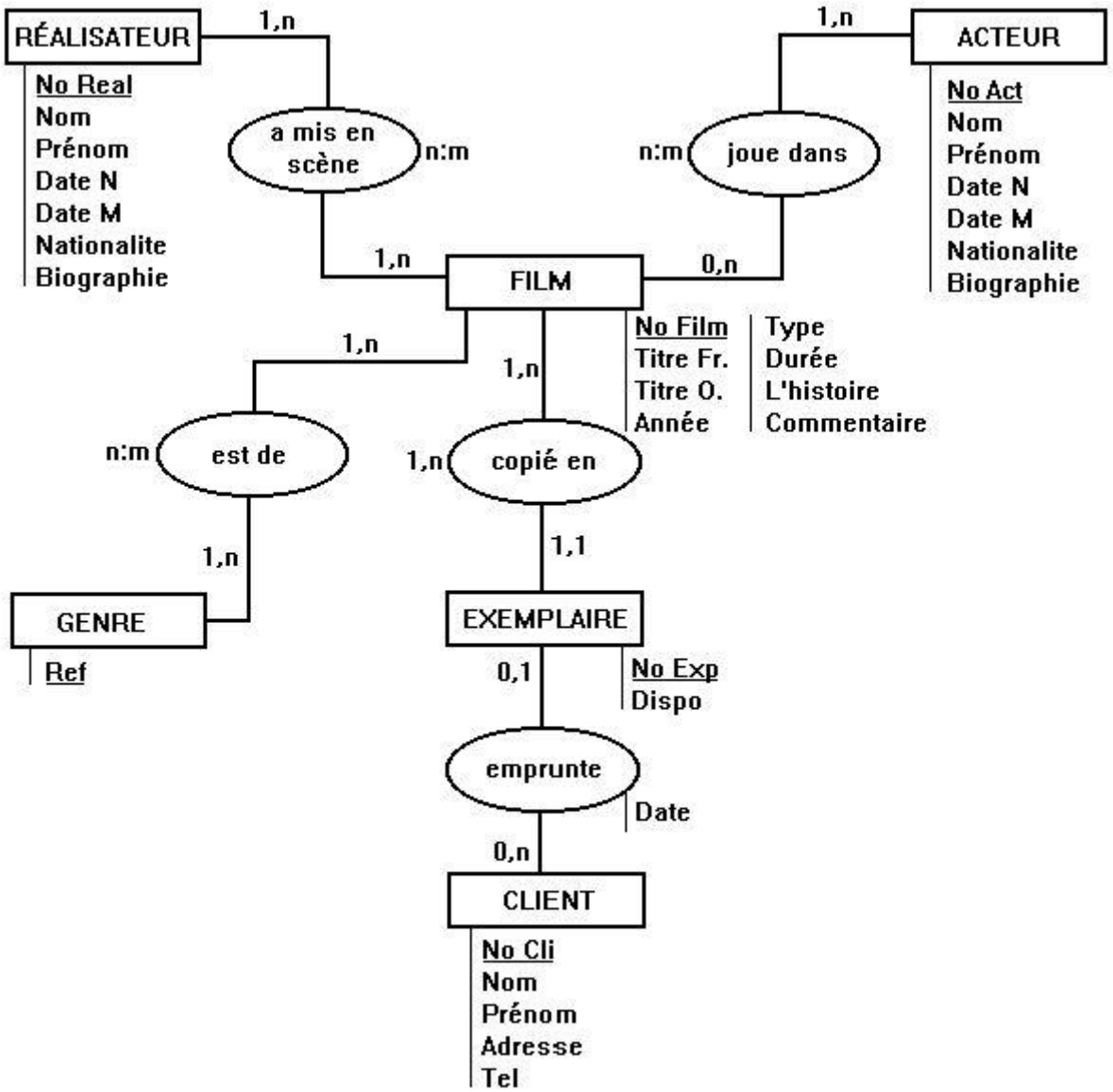


Il est possible de créer un objet générique PERSONNE à partir des objets PROFESSEUR et ELEVE :



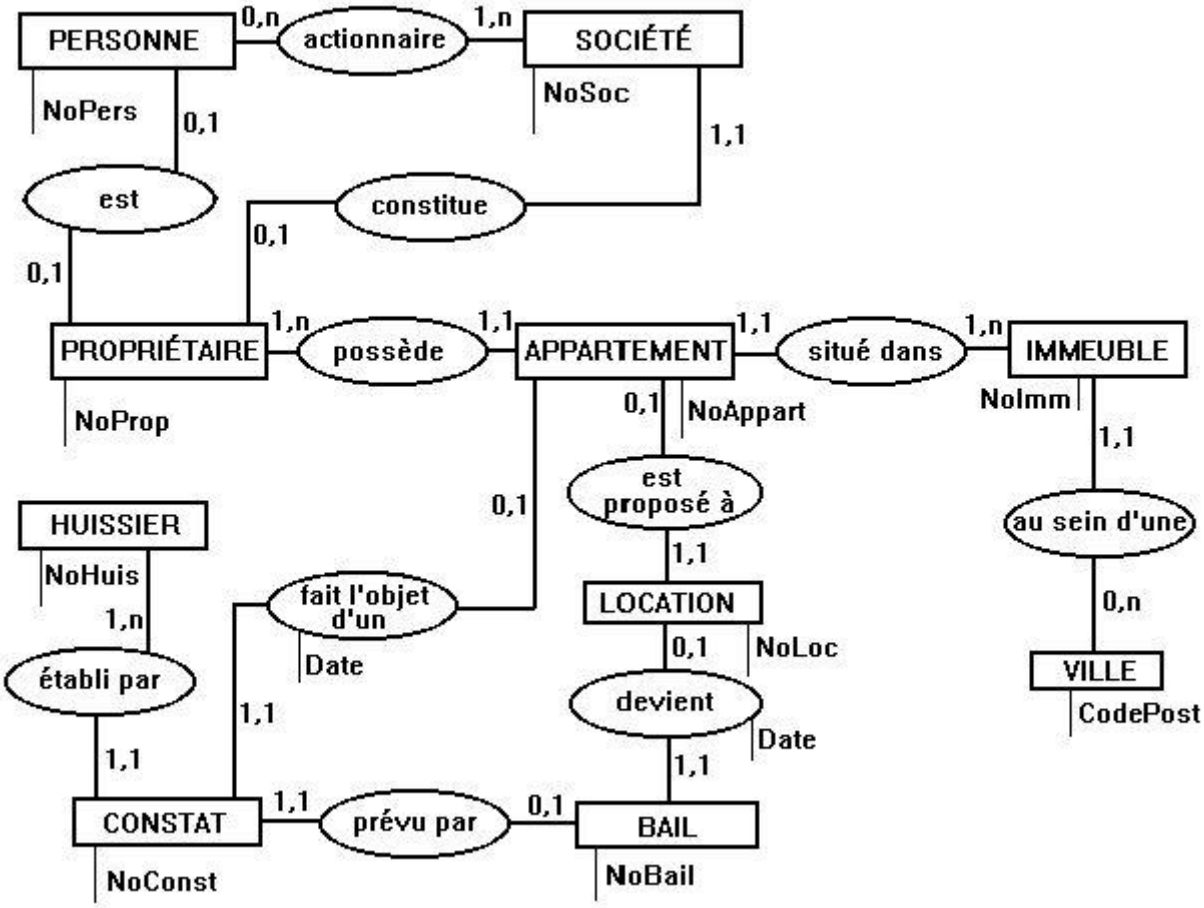
11 EXEMPLES DE MCD

11.1 Agence de location de films vidéo



Lors de la réalisation de la base de données, les entités RÉALISATEUR et ACTEUR peuvent être regroupées en une seule table car il y a un nombre non négligeable de réalisateurs qui sont acteurs, et vice-versa. Dans ce cas, un champ d'un seul caractère permettra de faire la différence entre un réalisateur pur, un acteur pur et un acteur réalisateur.
Notez aussi les cardinalités entre les entités ACTEUR et FILM, en effet, un film d'animation ne possède aucun acteur.
Dans l'entité EXEMPLAIRE figure un attribut "dispo" permettant de savoir si l'exemplaire n°X d'un film est disponible ou en cours d'emprunt.

11.2 Location d'appartements pour une agence immobilière



NOTA : pour simplifier l'écriture, ne figurent dans ce schéma que les attributs clefs ou les attributs d'associations.

12.Passage du schéma entité-association (MCD) à la construction des tables : Modèle Physique des Données (MPD)

Ce que nous venons de voir concerne l'analyse conceptuelle des données, c'est à dire un niveau d'analyse qui s'affranchi de toutes les contraintes de la base de données sur lequel va reposer l'application. Une fois décrit sous forme graphique, ce modèle est couramment appelé MCD pour "Modèle Conceptuel des Données".

Dès lors, tout MCD peut être tansformé en un MPD ("Modèle Physique des Données") c'est à dire un modèle directement exploitable par la base de données que vous voulez utiliser...

Tout l'intérêt de cet outil d'analyse est de permettre de modéliser plus aisément les relations existant entre les entités et d'automatiser le passage du schéma muni d'attributs aux tables de la base de données pourvues de leurs champs.

Voici maintenant les règles de base nécessaire à une bonne automatisation du passage du MCD au MPD :

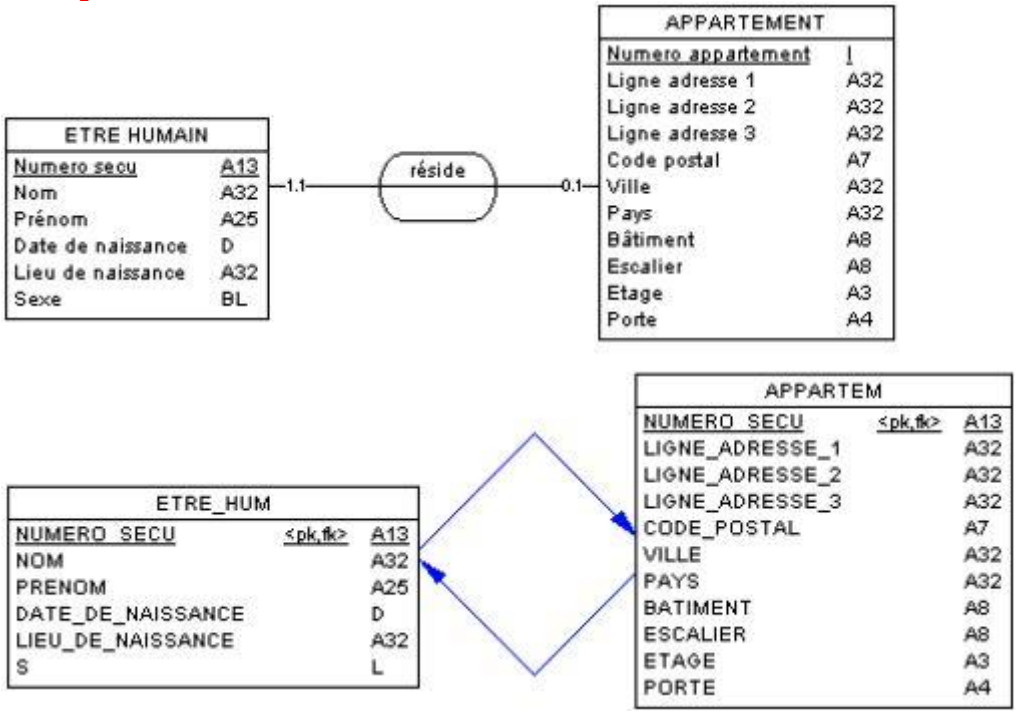
12.1 Transformation des entités

Règle n°1 : toute entité doit être représentée par une table.

12.1.1 Relations de type 1:1

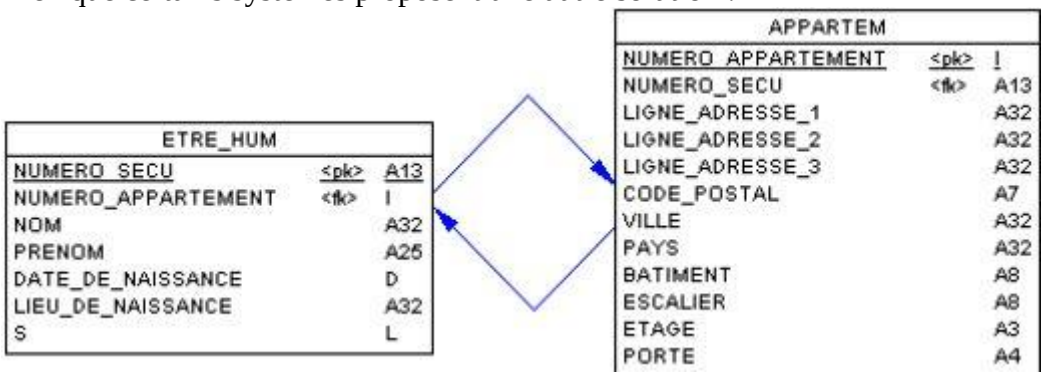
Règle n°2 : Dans le cas d'entités reliées par des associations de type 1:1, les tables doivent avoir la même clef.

Exemple :



NOTA : on aurait pu choisir l'autre clef comme clef commune, à savoir le n° d'appartement.

Bien que certains systèmes proposent une autre solution :

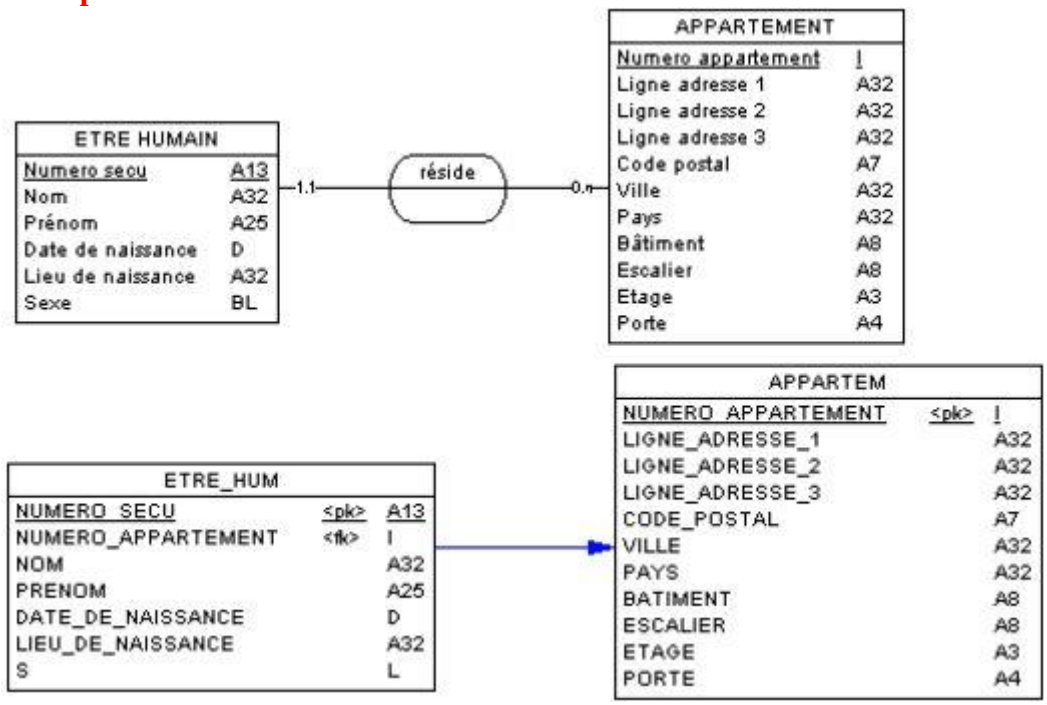


Dans ce cas une étude approfondie de la solution à adopter est nécessaire, mais ce type de relation est en général assez rare et peu performante...

12.1.2 Relations de type 1:n

Règle n°3 : Dans le cas d'entités reliées par des associations de type 1:n, chaque table possède sa propre clef, mais la clef de l'entité côté 0,n (ou 1,n) migre vers la table côté 0,1 (ou 1,1) et devient une clef étrangère (index secondaire).

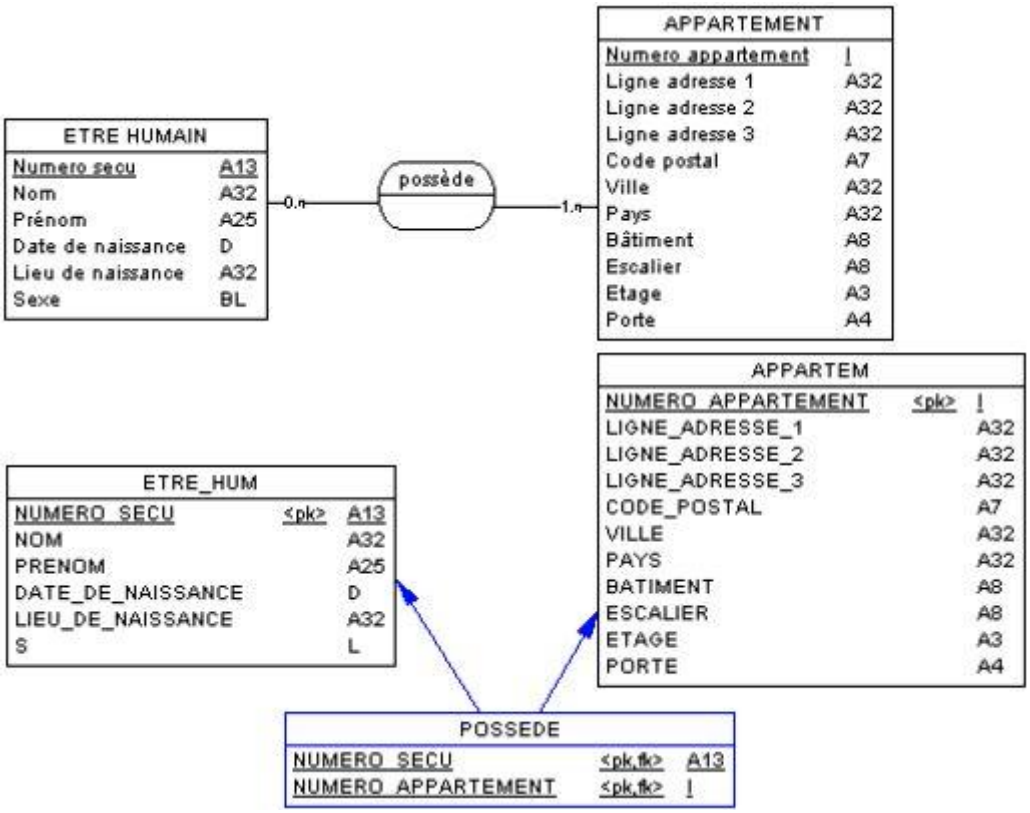
Exemple :



12.1.3 Relations de type n:m

Règle n°4 : Dans le cas d'entités reliées par des associations de type n:m, une table intermédiaire dite table de jointure, doit être créée, et doit posséder comme clef primaire une conjonction des clefs primaires des deux tables pour lesquelles elle sert de jointure.

Exemple :

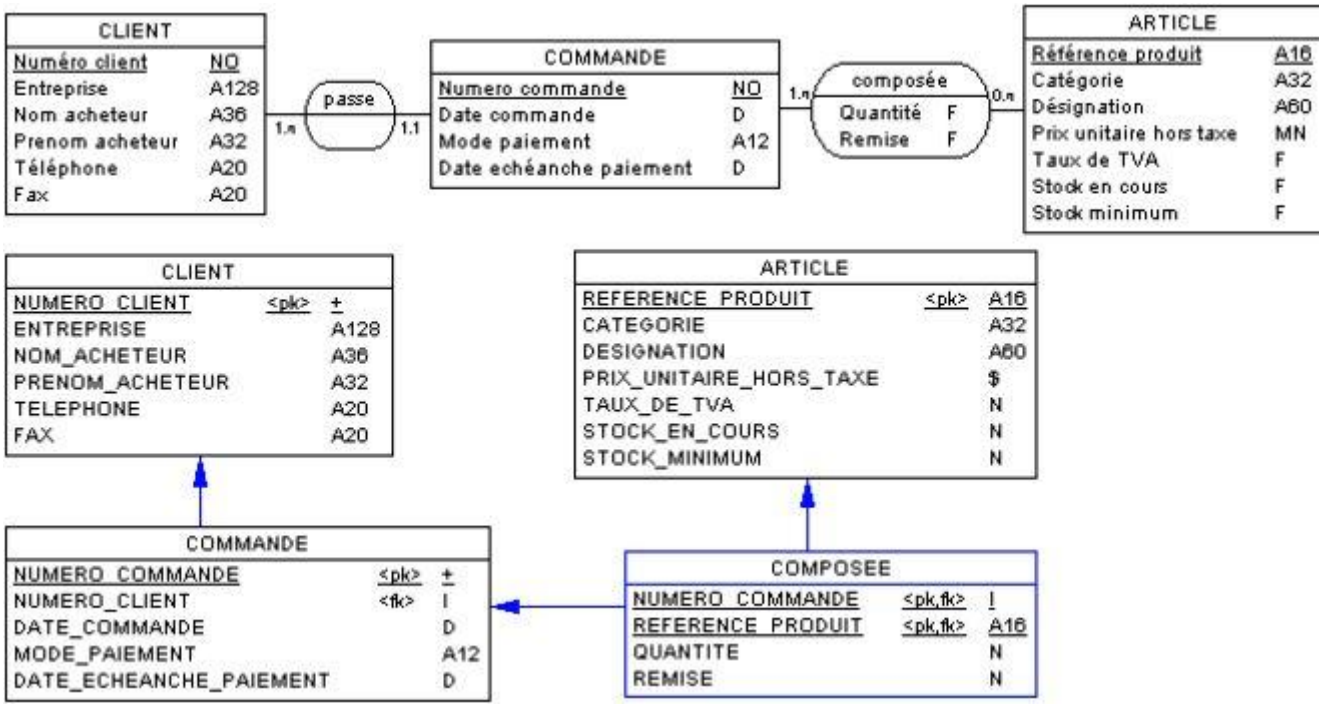


12.2 Ou placer les attributs d'association ?

Règle n°5 : Cas des associations pourvues d'au moins un attribut :

- **si le type de relation est n:m, alors les attributs de l'association deviennent des attributs de la table de jointure.**
- **si le type de relation est 1:n, il convient de faire glisser les attributs vers l'entités pourvue des cardinalités 1:1.**
- **si le type de relation est 1:1, il convient de faire glisser les attributs vers l'une ou l'autre des entités.**

Exemple : Pour synthétiser toutes ces règles, voici un exemple de modélisation d'une application. En l'occurrence il s'agit d'un service commercial désirant modéliser les commandes de ses clients.



BIBLIOGRAPHIE

- Bases de Données et Modèles de Calcul - Outils et Méthodes pour l'Utilisateur - Jean-Luc HAINAUT - InterEditions 1994
- Conception de bases de données : du schéma conceptuel au schéma physique - GALACSI - Dunod 1989
- La méthode MERISE, principes et outils (Tome 1 & 2) - TARDIEU, ROCHFELD, COLLETTI - Les éditions d'organisation 1986
- Modélisation dans la conception des systèmes d'information - ACSIOME - Masson 1990
- Apprendre et pratiquer MERISE - J. GABAY - Masson 1993
- MERISE, vers une modélisation orientée objet – José MOREJON – Les Editions d'Organisation 1994
- Maîtriser les bases de données – Georges GARDARIN – Eyrolles 1993
- Introduction aux bases de données, 6e édition– Chris J. DATE – Thomson International Publishing 1998
- Concepts fondamentaux de l'informatique – Alfred AHO, Jeffrey ULLMAN – Dunod 1993
- AMC*Designor, mise en œuvre de MERISE – Gilles GUEDJ – Eyrolles 1996