Chapitre IV :

Modélisation Conceptuelle des Données

Université Boumerdès/Faculté des Sciences/Département Informatique

Cours Licence 2 – Informatique et Tecweb

Par Ahmed AIT BOUZIAD

Décembre 2015

1 INTRODUCTION

Comme nous l'avons introduit dans les chapitres précédents, dans un SI en fonctionnement, les deux composants les plus importants sont les données et les traitements qui apparaissent intimement liées surtout du point de vue de l'utilisateur. Dans ce chapitre, nous ne nous intéresserons à la période conceptuelle des données du cycle de vie d'un SI.

Les méthodes de développement de SI proposent de décrire les données d'un domaine en utilisant des modèles ou des schémas en utilisant des formalismes (langage) spécifiques.

L'ensemble des informations utilisées, échangées constitue l'univers du discours du domaine. Dans cet univers du discours, on fait référence à des objets concrets (étudiant) ou abstraits (cours) et à des associations (ou relations) entre ces objets (*inscription* d'un étudiant à un cours). L'objectif de cette modélisation de données est d'identifier, de décrire par des informations et de modéliser ces objets et associations.

Dans ce cours nous allons étudier l'un des modèles le plus important proposé par la méthode MERISE introduite dans le chapitre précédent. Il s'agit du Modèle Conceptuel de Données qui utilise le formalisme dit « Entité-relation ». En dehors du contexte de la méthode, il a été reconnu internationalement (par l'ISO) et fait l'objet de nombreux développements.

Comme nous l'avons vu dans le chapitre précédent, MERISE est une méthode de développement de SI qui intègre dans la période de conception de sa démarche une approche par niveau d'abstraction basée le principe de la séparation des données et des traitements.

Elle possède, aussi bien pour les données que pour les traitements, un certain nombre de **modèles** qui sont répartis sur ses 4 niveaux d'abstraction :

- Le niveau **conceptuel**,
- Le niveau **organisationnel**,
- Le niveau **logique**,
- Le niveau **physique**.

Ainsi, dans ce chapitre, nous ne nous intéresserons qu'au niveau conceptuel de MERISE et plus particulièrement au modèle conceptuel de données (MCD) utilisé lors de la conception d'une base pour les données. Nous verrons comment élaborer MCD qui est une représentation graphique et structurée des informations mémorisées par un SI. Il est basé sur deux notions principales : les **entités** et les **associations**, d'où sa seconde appellation : le **modèle Entité/Association (E/A)**.

A la fin de ce chapitre nous introduirons le « *diagramme de classe UML* » ; qui est un autre modèle de conception de données introduit dans le cadre de méthode objets telle que UP, RUP, Scrum et XP. Ensuite, nous le comparerons au MCD de MERISE.

2 PROBLEMATIQUE DU MODELE CONCEPTUEL DE DONNEES (MCD)

Le modèle conceptuel de données (MCD) est la représentation de l'ensemble des données du domaine étudié qui ne tient compte ni des aspects techniques de mémorisation et d'accès et ni des conditions d'utilisation par tel ou tel traitement.

Dans un SI, l'ensemble des informations utilisées constitue l'univers du discours du domaine. Dans cet univers du discours, on fait référence à des objets concrets (*Professeur*) ou abstraits (*Matière*) et à des associations entre ces objets (Un professeur *enseigne* une matière). L'objectif du modèle conceptuel de données est :

1. d'identifier ces objets et associations,
2. de les décrire par des informations,
3. et enfin de les modéliser dans un formalisme (ou langage) donnée.

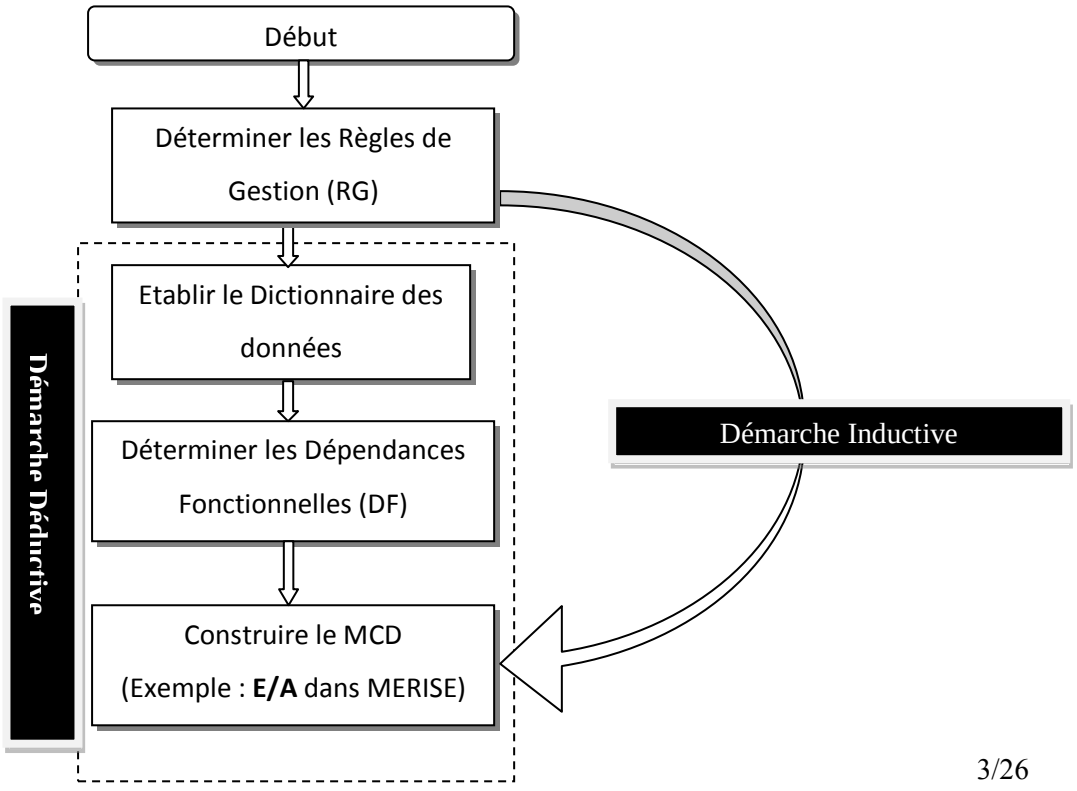
Pour l'élaboration d'un MCD, on distingue deux façons de faire basées essentiellement sur le discours (parlé ou écrit) de l'utilisateur ou du gestionnaire:

Une démarche dite déductive qui s'appuie sur l'existence préalable d'une liste d'informations à structurer. Dans ce cas le discours est décortiqué en informations élémentaires avant de concevoir le MCD.

Une démarche inductive qui cherche à mettre rapidement en évidence les différents concepts évoqués dans le discours, puis à les décrire par des informations. Autrement dit, le concepteur peut directement, à l'aide du formalisme choisi, construire le MCD.

Remarque : la démarche déductive est plus lourde à mettre en œuvre, et donc s'apprête moins par exemple en étude préalable confrontée à l'exigence d'une durée d'étude relativement courte (voir le chapitre précédent).

La figure suivante montre les étapes d'élaboration d'un MCD que nous développerons dans ce chapitre.



3 LES REGLES DE GESTION

Avant de se lancer dans la création d'un modèle entités et associations, il faut recueillir les besoins des futurs utilisateurs de l'application. Et à partir de ces besoins, on doit être en mesure d'établir les règles de gestion des données à conserver.

Prenons l'exemple d'un projet d'informatisation de la gestion de la scolarité d'une faculté pour une année d'étude donnée. Les données disponibles dans un tel environnement concernent les professeurs, les étudiants, les modules enseignés, les notes et les salles. On lui fixe les règles de gestion suivantes :

1. Chaque professeur est caractérisé par son code, son nom, son prénom, son adresse, son grade et son nombre d'heures d'enseignement selon son grade.
2. Un professeur peut enseigner plusieurs modules.
3. Chaque module est caractérisé par son code et son libellé (ou nom).
4. Dans chaque salle et lors d'une séance, un module est programmé pour un enseignant sous forme de CT, TD ou TP.
5. Une séance est caractérisée par un code, un jour de semaine, une heure de début et une heure de fin.
6. Chaque étudiant est caractérisé par son matricule, son nom, son prénom, sa date de naissance et son adresse.
7. Tout Domaine de formation est caractérisé par un code, un libellé. Exemple: MI, ST, STH, SM.
8. Plusieurs filières sont proposées dans chaque domaine, chacune caractérisée par son code et son libellé.
9. Un ou plusieurs tronc-communs peuvent-être proposés dans un domaine avant la spécialisation.
10. Un tronc-commun est caractérisé par son code et son libellé. Exemple : MI
11. Chaque Tronc-Commun est constitué d'un certain nombre de modules dotés d'un coefficient que l'étudiant doit suivre.
12. Chaque étudiant s'inscrit à un domaine, qui propose en première année d'étude soit un Troc-commun soit directement la première année d'étude d'une spécialité de licence. Exemple : Tronc-commun MI du domaine MI.
13. Plusieurs spécialités de licence et de master sont proposées dans chaque filière, caractérisées chacune par son code, son libellé et le nombre d'années d'étude nécessaires (3 ou 5).
14. Chaque année d'étude de spécialité est constituée de modules dotés d'un coefficient que l'étudiant doit suivre.
15. Une note par type d'évaluation est obtenue par un étudiant dans un module. Le type d'évaluation est caractérisé par son code et son libellé : CTL pour Contrôle Continue, EXA pour examen et RAT pour rattrapage.
16. Une moyenne par module est calculé pour chaque étudiants à partir des notes obtenues : Moyenne = (CTL + 2 * max(EXA, RAT)) / 3

Ces règles sont parfois données mais on peut être amené à les établir soi-même dans deux cas :

• Vous êtes à la fois maîtrise d'œuvre (MOE) et maîtrise d'ouvrage (MOA), et vous développez une application pour votre compte et/ou selon vos propres directives.

• ou bien, **Ce qui arrive le plus souvent**, les futurs utilisateurs de votre projet n'ont pas été en mesure de vous fournir ces règles avec suffisamment de précision ; c'est pourquoi vous devrez les interroger afin d'établir vous même ces règles.

Les règles de gestion expriment les contraintes d'intégrité du MCD. Elles représentent les lois du domaine réel qu'on veut modéliser.

4 DICTIONNAIRE DE DONNEES

Comme pour les règles de gestion, la base de l'identification de toutes les informations circulant dans le domaine c'est les *l'interview* et la *collecte de documents*. On peut centraliser toutes ces informations et les règles de gestion au sein d'un même document appelé **dictionnaire des données**.

Le dictionnaire des données est un document qui regroupe toutes les données que vous aurez à conserver dans votre future base de données et qui figureront donc d'abord dans le MCD. La description des données dans le dictionnaire peut être plus ou moins élaborée selon le degré de précision souhaité. Il pourrait se présenté comme suit :

Num	**Nom de la donnée**	**Signification**	**Type :** A/AN/N/D/L	**Longueur**	**Nature :** E/CO/CA	**Règle de Calcul**	**Règle de gestion (Intégrité)**	**Documents**

Nom de la donnée :

Décrit le nom ou le code de la donnée, par exemple : **Matricule** pour Le matricule de l'étudiant.

Signification :

Permet de décrire de manière très succincte la signification de la donnée.

Type :

C'est le type de la donnée en cours de description. Voici les types génériques reconnus :

- Alphabétique (**A**) : les données sont constituées que de caractères.
- Alphanumérique (**AN**) : dans ce cas les données sont constituées de caractères et de nombres.
- Numérique (**N**) : les nombres.
- Date (**D**)
- Logique (**L**) : Vrai ou Faux (ou 1/0)

Longueur :

Dans cette colonne on indique la longueur exacte ou estimée de la donnée.

Nature :

Une données est soit : Elémentaire (E), Calculé (CA) ou Concaténée (CO).

Règle de calcul :

C'est la formule de calcul permettant d'obtenir une donnée calculée.

Règle de gestion ou d'intégrité :

Indique la règle de gestion ou d'intégrité relative à la forme éventuelle de la donnée

Document :

Indique le document dans lequel la donnée à été trouvée.

4.1 Validation du dictionnaire de données

Pour chaque donnée identifiée, avant de l'ajouter dans le dictionnaire, il faudrait vérifier si elle respecte les trois conditions suivantes :

1. La donnée n'est pas déjà répertoriée : Par exemple le matricule de l'étudiant peut apparaitre dans différents documents étudiés et pour autant, il faut faire attention à ne le reporter dans le dictionnaire de données qu'une seule fois.
2. Elle n'a pas de **synonyme** déjà répertoriée dans le dictionnaire : Par exemple le code de l'étudiant et son matricule. Dans ce cas on doit retenir l'un des deux noms uniquement.
3. Elle n'est pas un **homonyme** : le nom de la donnée existe déjà mais est associée à une signification différente. Exemple : le nom est associé à l'étudiant et au professeur. Dans ce cas, on doit lever l'ambigüité en modifiant les noms des données. Par exemple : NomEtud et NomProf.

Ainsi, le dictionnaire de données sera validé s'il est sans redondance, sans synonyme et sans homonyme.

4.2 Exemple de dictionnaire de données

Reprenons l'exemple d'informatisation de la scolarité. Voici ce que pourrait être son dictionnaire de donnée :

Num	Nom de la donnée	Signification	Type: A/AN/ N/D/L	Longueur	Nature : E/CO/CA	Règle de Calcul	Règle de gestion (Intégrité)	Document Source
1	CodProf	Code professeur	AN	5	E		Une lettre + 3 chiffres	RG1
2	NomProf	Nom professeur	A	30	E			RG1
3	PreProf	Prénom professeur	A	30	E			RG1
4	AdrProf	Adresse professeur	AN	80	CO		N°Rue+Nom rue +Ville	RG1
5	Grade	Grade professeur	A	2	E		MA ou MC ou PR	RG1
6	NbrH	Nombre heure enseignement hebdomadaire	N	2	E		0<= NbrH<=12	RG1
7	CodMod	Code module	AN	4	E			RG3
8	LibMod	Libellé module	A	30	E			RG3
9	CodSalle	Code Salle	AN	5	E			RG4
10	Design	Désignation Salle	AN	30	E			RG4
11	CodSea	Code séance	AN	2	E			RG5
12	JouSea	Jour de la séance	A	10	E		Sam,Dim,.. ,Jeu	RG5
13	HeuDebSea	Heure début séance	D	5	E		Forme :HH:MM	RG5
14	HeurFinSea	Heure fin séance	D	5	E		Forme : HH:MM	R5

15	Matricule	Matricule étudiant	AN	6	E		Une lettre + AA + NNN	RG6
16	NomEtud	Nom étudiant	A	30	E			RG6
17	PreEtud	Prénom étudiant	A	30	E			RG6
18	DateNais	Date Naissance étudiant	D				Forme JJMMAA où JJ : 01 à 30 MM : 01 à 12 AA : 00 à 99	RG6
19	AdrEtud	Adresse étudiant	AN	80	CO		N°Rue+Nom rue +Ville	RG6
20	CodDom	Code domaine	A	3	E			RG7
21	LibDom	Libellé domaine	A	30	E			RG7
22	CodFil	Code filière	A	5	E			RG8
23	LibFil	Libellé filière	A	30	E			RG8
24	CodTC	Code tronc commun	A	5	E			RG10
25	LibTC	Libellé tronc commun	A	30	E			RG10
26	Coef	Coefficient d'un module rattaché à une formation	N	1	E			RG11
27	CodSpec	Code spécialité	A	6	E			RG13
28	TypeSpec	Type spécialité	A	1	E		L : Licence, M : Master	RG13
29	LibSpec	Libellé spécialité	A	30	E			RG13
29	NbrAn	Nombre années d'étude de la spécialité	N	1	E		3 ou 5	RG13
30	AnEtud	Année d'étude	N	1	E		1,2,3 en L et 1,2 en M	RG14
31	CodEval	Code évaluation	A	3	E		CTL,EXA,RAT	RG15
32	LibEval	Libellé évaluation	A	15	E			RG15
33	Note	Une note de l'étudiant	N	5	E		Forme 99,99	RG15
34	MoyMod	Moyenne dans un module	N	5	CA	(CTL + 2*max(EXA,RAT))/ 3	Forme 99,99	RG16
35	TypeEns	Type enseignement	A	2	E		CT, TD, TP	RG4

5 LES DEPENDANCES FONCTIONNELLES

L'analyse des données commence par la recherche des liens existants entre chaque donnée. Ces liens sont exprimés en utilisant la notion de dépendance fonctionnelle (df) entre données. Cette démarche de recherche des df constitue l'étape centrale de l'analyse. Si elle est négligée c'est toute la suite de l'étude qui en subira les conséquences.

5.1 Quelques définitions

5.1.1 Dépendance Fonctionnelle (DF) entre deux données

On dira que deux données **a** et **b** sont reliées par une dépendance fonctionnelle, si la connaissance d'une valeur de **a** nous permet de connaitre une et une seule valeur de **b**.

On écrira : **a** ⟶ **b**

On peut lire cette df comme suit :

- **b** dépend fonctionnellement de **a**
- ou bien, **a** détermine **b**

Exemple :

La connaissance de la valeur du matricule de l'étudiant nous permet de connaître sans ambiguïté la valeur d'un et un seul nom d'étudiant. Autrement dit, si on connaît le matricule d'un étudiant alors on doit pouvoir connaître son nom et celui-ci sera unique :

Matricule ⟶ NomEtud

La réciproque est fausse. En effet, le nom de l'étudiant ne permet pas de déterminer son matricule, puisque plusieurs étudiants peuvent avoir le même nom mais des matricules différents. Ainsi, la df suivante est fausse :

NonEtud ⟶ Matricule

Note : la partie gauche de la df (ici Matricule) est appelée **source** de la df. Sa partie droite est appelée **but** de la df (ici NomEtud).

5.1.2 Dépendance fonctionnelle composée

Une DF qui comporte plusieurs données dans sa source ou son but est dite **composée**.

Exemple : Matricule + CodMod ⟶ MoyMod

Le matricule seul ne suffit pas à déterminer la moyenne d'un étudiant dans un module. Puisque un étudiant possède plusieurs moyennes. De plus, le code module ne suffit pas non plus, puisque un module peut concerner plusieurs étudiants et donc plusieurs Moyennes.

Par contre, si on admet qu'un étudiant ne peut suivre un module qu'une seule fois dans son cursus, alors la connaissance d'un matricule est d'un module détermine celle de la moyenne dans le module.

Dans la suite, on utilisera une virgule à la place du + dans la df :

Matricule , CodMod ⟶ MoyMod

5.1.3 Dépendance fonctionnelle élémentaire

Une DF A ⟶ B est dite **élémentaire** dans deux cas :

Cas 1 : A est réduite à une seule donnée.

Cas 2 : A est composée de plusieurs données et

Il n'existe pas une partie C de A qui détermine B (ie. C → B)

Exemples :

1. La df **« Matricule,NomEtud → AdrEtud »** n'est pas élémentaire puisque la connaissance du matricule suffit pour déterminer l'adresse de l'étudiant : **Matricule → AdrEtud**
2. La df **« Matricule → AdrEtud »** est élémentaire.
3. La df **« Matricule,CodMod → MoyMod »** est élémentaire.

5.2 Propriétés des dépendances fonctionnelles

Soit A, B, C et D des sous-ensembles des données du dictionnaire de données.

- **Réflexivité**

Pour toute partie B de A, on a : A → B

Cas particulier : A → A

- **Transitivité**

Si A → B et B → C alors A→ C

Exemple : **CodProf → Grade** et **Grade → NbrH** alors **CodProf → NbrH**

- **Augmentation**

Si A → B alors A,C → B pour tout C

Exemple : **Matricule,NomEtud → DateNaiss**

Note : les trois propriétés ci-dessus sont connues sous le nom d'axiomes d'Amstrong.

Trois autres propriétés connues peuvent êtres déduites de ces axiomes :

- **Union (ou Additivité)**

Si A → B et A → C alors A → B,C

Exemple : Matricule → NomEtud et Matricule → PreEtud

Alors Matricule → NomEtud,PreEtud

- **Pseudo-transitivité**

Si A → B et B,C→ D alors A,C → D

Exemple : Matricule → CodDom et CodDom, CodMod → Coefficient

Alors Matricule, CodMod → Coefficient

- **Décomposition (ou Projection)**

Si A → B,C alors A → B et A → C

Exemple : Matricule → NomEtud , PreEtud

Alors Matricule → NomEtud et Matricule → PreEtud

Exercice : Montrer ces trois dernières propriétés à partir des axiomes d'Amtrong.

5.3 Autres définitions

Grâce aux propriétés ci-dessus, on introduit deux types de df supplémentaires.

5.3.1 Dépendance fonctionnelle élémentaire directe

Une df élémentaire **A → B** est dite **directe** s'il n'existe aucun groupe d'attributs **C** tel que :

1. **A → C**

et

2. **C → B**

Autrement dit **B** ne dépend pas transitivement de **A**.

Exemple : Soit les trois df suivantes :

1. Matricule → CodDom
2. CodDom → LibDom
3. Maricule → LibDom

Les deux premières df sont élémentaires et directes, mais la troisième ne l'est pas puisqu'on peut l'obtenir par transitivité grâce aux deux premières :

Matricule → CodDom → LibDom

5.3.2 Dépendance fonctionnelle canonique

Une df est dite **A → B** est dite **canonique** si **B** est réduit à une seule donnée.

Exemples : les df suivantes sont canoniques

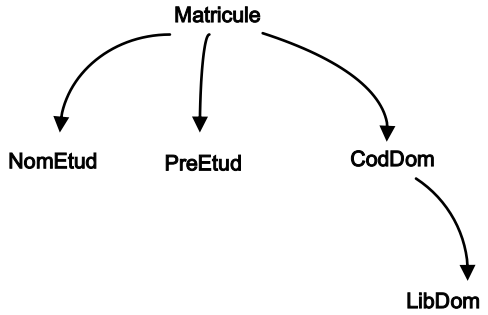
1. **Matricule → CodDom**
2. **CodDom, CodMod → Coefficient**

Note : Toute df qui n'est pas canonique peut être transformé en un ensemble de df canonique en utilisant la règle de décomposition.

5.4 Graphe de dépendances fonctionnelles

Le graphe de dépendances fonctionnelles (GDF) permet une représentation spatiale de l'ensemble des df identifiées et montre ce que sera le futur modèle conceptuel de données.

Voici un exemple de GDF :



5.5 Méthodologie d'élaboration des dépendances fonctionnelles

La liste des df est élaborée en examinant le dictionnaire de données (DD) et les règles de gestion. Les étapes à suivre sont :

1. Pour la recherche des df, on exclut les données calculées. Autrement dit, les df ne concernent que les données non déduites.
2. Rechercher les df formées par *deux* données (ou deux propriétés) uniquement, une pour la source et une autre pour le but. Elles sont donc forcément *élémentaires*. Et ne retenir que les df *direct.*
3. Traiter les données ou propriétés **isolées** qui ne sont pas impliquées dans les df précédentes. Deux cas possibles :
   (a) Soit en créant une nouvelle donnée permettant de déterminer une ou plusieurs données isolées. Ce qui permet de regrouper des données isolés
   (b) Soit en composant des données pour déterminer les données isolées. Ces nouvelles df sont *élémentaires et composées*.
4. Si le graphe obtenu comporte des cycles, on élimine cette anomalie en supprimant une df.

5. On élimine toutes les df qui peuvent être déduite à partir d'autres df grâce aux propriétés des df. On ne gardera qu'un ensemble minimal de df ; celles qui sont nécessaires.

5.6 Exemple d'élaboration des dépendances fonctionnelles

Reprenons l'exemple d'informatisation de la scolarité.

Etape 1 : on exclut les données calculées.

C'est le cas de la moyenne de l'étudiant dans un module **MoyMod** qui est le résultat de calcul à partir des notes de contrôle, de l'examen et d'un éventuel rattrapage. Ces informations sont utiles pour le développeur de l'application qui plus tard mettra en œuvre les procédures de calculs. Dans le cycle de développement de Merise cette donnée est déduite et non stockable, elles ne sont donc pas concernée par la suite du processus de conception.

Etape 2 : Recherche des df élémentaires directs formées de deux propriétés uniquement.

a) A la lecture du dictionnaire de données nous pouvons déduire plusieurs groupes d'informations distinctes :

❖ **Groupe d'informations caractérisant un professeur et les df les reliant :**

Il est facile d'extraire du DD les données suivantes qui participeraient à la caractérisation d'un professeur : CodProf, NomProf, PreProf, AdrProf, Grade, NbrH

Pour déduire les df pouvant les relier, posons-nous le type de questions suivante : « Quant je connais le code du professeur (CodProf), est-ce-que je connais son nom (NomProf) et de façon unique ? »

Si la réponse et « oui » alors on en déduit la df élémentaire suivante : CodProf → NomProf

En procédant da la même manière, on peut trouver l'ensemble de df élémentaires suivante :

CodProf → PreProf

CodProf → AdrProf

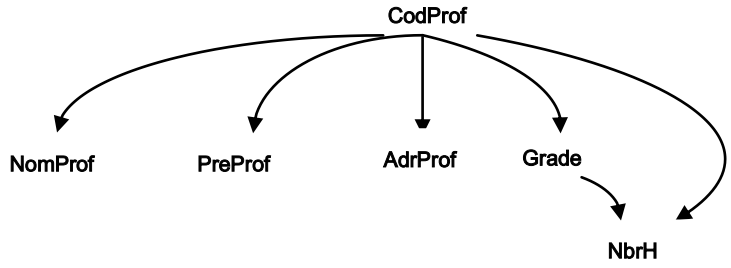
CodProf → Grade

CodProf → NbrH

Remarque : Ces df auraient pu s'écrire de la façon suivante :

CodProf → NomProf, PreProf, AdrProf, Grade, NbrH

Cette liste de df peut se visualiser par un graphe comme suit :



- **Groupe d'informations caractérisant un module et les df les reliant :**

On procédant de la même façon précédemment, on arrive à la liste suivante :
CodMod → libMod

- **Groupe d'informations caractérisant une salle et les df les reliant :**

CodSalle → Design

- **Groupe d'informations caractérisant une séance et les df les reliant :**

CodSea → JouSea, HeuDebSea, HeuFinSea

- **Groupe d'informations caractérisant un étudiant et les df les reliant :**

Matricule → NomEtud, PreEtud, DateNAiss, AdrEtud

- **Groupe d'informations caractérisant un module et les df les reliant :**

CodDom → libDom

- **Groupe d'informations caractérisant une filière et les df les reliant :**

CodFil → LibFil

- **Groupe d'informations caractérisant un tronc commun et les df les reliant :**

CodTC → LibTC

- **Groupe d'informations caractérisant une spécialité et les df les reliant :**

CodSpec → TypeSpec, LibSpec, NbrAn

- **Groupe d'informations caractérisant une évaluation et les df les reliant :**

CodEval → LibEval

b) <u>Cherchons les df élémentaires entre les groupes identifiés :</u>

Utilisons les règles de gestions.

- La règle de gestion 8 (RG8) nous permet de déduire la df suivante :

CodFil → CodDom

- La règle de gestion 9 (RG9) nous permet de déduire la df suivante :

CodTC → CodDom

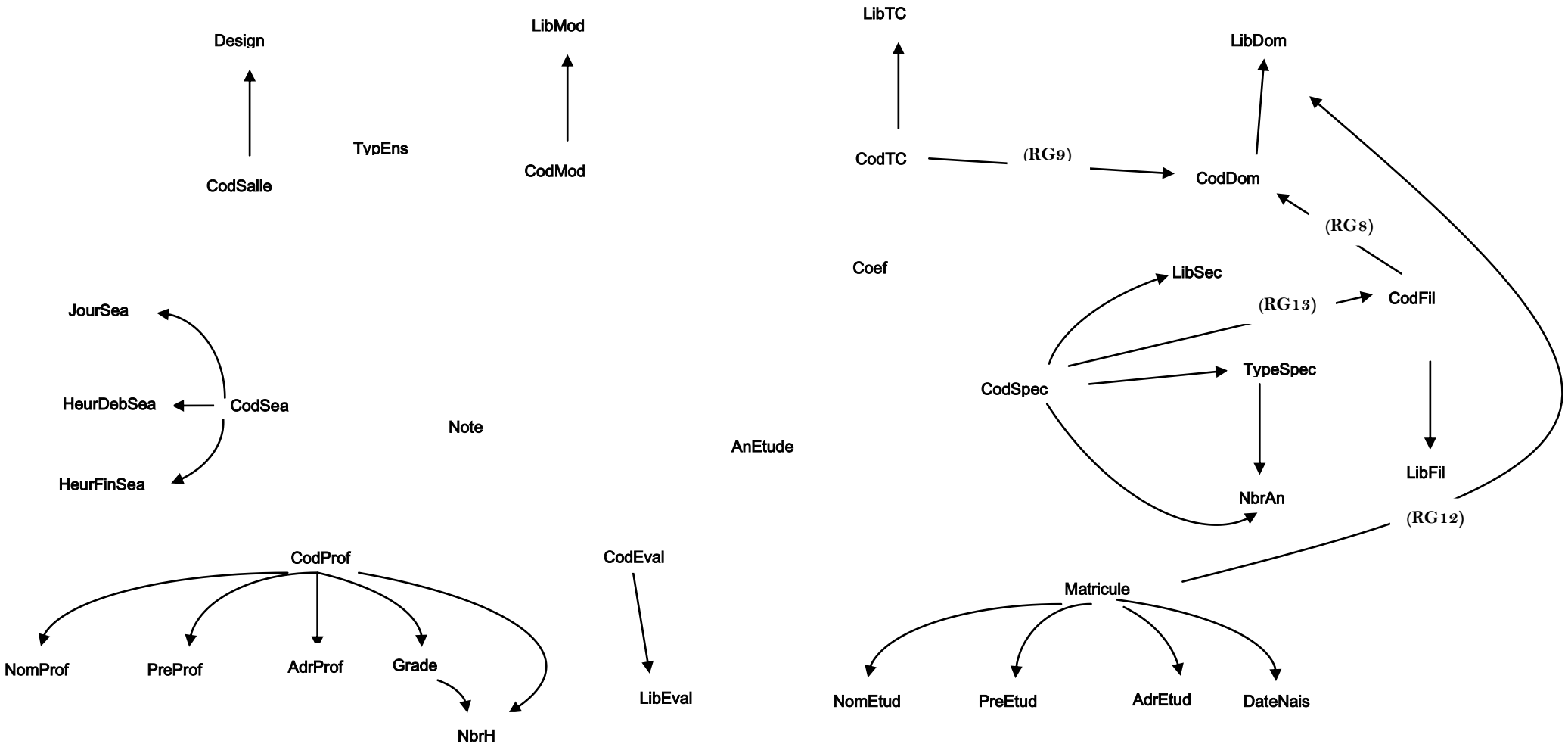
- La règle de gestion 12 (RG12) nous permet de déduire la df suivante :

Matricule → CodDom

- La règle de gestion 13 (RG13) nous permet de déduire la df suivante :

CodSpec → CodFil

c) <u>Le graphe des dépendances fonctionnelles (Etape2) :</u>


Design
LibMod
TypEns
CodSalle
CodMod
LibTC
LibDom
CodTC
(RG9)
CodDom
(RG8)
Coef
LibSec
JourSea
(RG13)
CodFil
HeurDebSea
CodSea
CodSpec
TypeSpec
Note
AnEtude
HeurFinSea
LibFil
NbrAn
(RG12)
CodProf
CodEval
Matricule
NomProf
PreProf
AdrProf
Grade
LibEval
NomEtud
PreEtud
AdrEtud
DateNais
NbrH

Etape 3 :

- Considérons la donnée isolée « *Type enseignement* » (**TypeEns**)

  Utilisons la règle de gestion n°4 (RG4) pour trouver la df composée :

  Est-ce que la connaissance de la salle nous permet de connaître de façon unique le type d'enseignement ? Non, puisque la RG4 suggère qu'il peut se dérouler plusieurs type d'enseignement dans une salle si plusieurs séances y sont prévues. Nous déduisons que cette donnée TypEns fait partie d'une dépendance fonctionnelle composée. En effet, si en plus nous connaissons la séance (jour, heure début et heure fin), alors la réponse est oui !

  Voici donc la df composée pour cette donnée isolée :

  CodSalle, CodSea → TypEns

  De plus, la connaissance de la salle et de la séance nous permettent de connaître en plus le module qui y donné et le professeur qui l'assure. Ainsi, nous avons les df suivantes :

  CodSalle, CodSea → TypEns, CodProf, CodMod

- En procédant de la même façon, nous déduisons les df composées suivantes :

A partir de la RG11 :

CodMod, CodTC → Coef

A partir de la RG12 :

Matricule, AnEtud → CodTC, CodSpec

A partir de la RG14 :

AnEtud, CodSpec, CodMod → Coef

A partir de la RG15 :

Matricule, CodMod, CodEval → Note

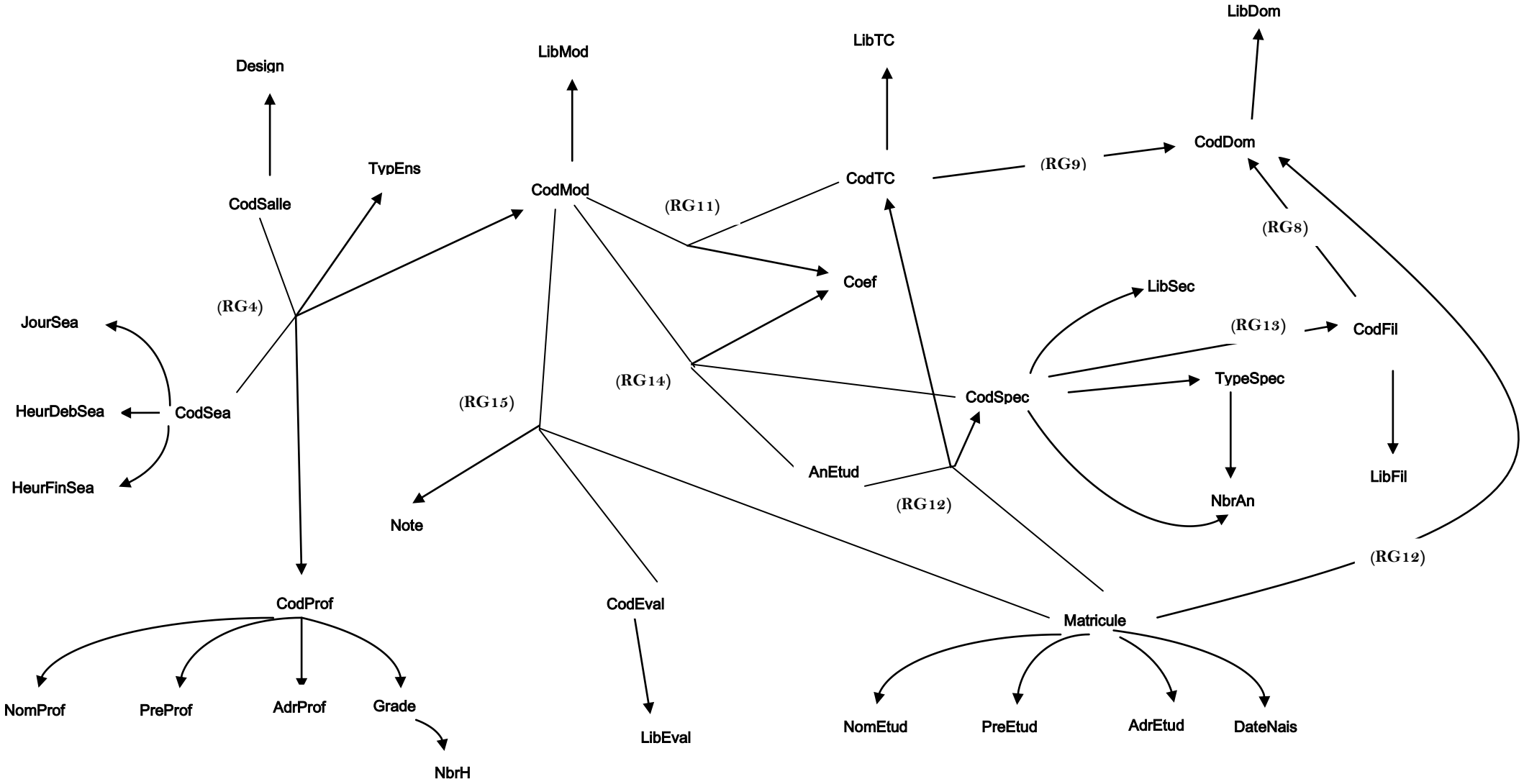
Etape 4 : Elimination de cycle.

Il n'y a pas de cycles dans ce graphe.

Etape 5 : Elimination des df déduites

La df CodProf → NbrH est à éliminée puisqu'elle peut être obtenue par transitivité.

A la fin de ces cinq étapes, on obtient le graphe des df ci-dessous : (Les RG à l'origine des df composées sont montrées dans ce graphe)


LibDom
LibTC
LibMod
Design
CodDom
TypEns
CodTC
(RG9)
CodMod
CodSalle
(RG11)
(RG8)
Coef
LibSec
JourSea
(RG4)
(RG13)
CodFil
HeurDebSea
CodSea
(RG14)
TypeSpec
(RG15)
CodSpec
HeurFinSea
AnEtud
LibFil
NbrAn
Note
(RG12)
(RG12)
CodProf
CodEval
Matricule
NomProf
PreProf
AdrProf
Grade
NomEtud
PreEtud
AdrEtud
DateNais
NbrH
LibEval

6 LE MODELE CONCEPTUEL DES DONNEES

Le formalisme de description des données utilisé dans Merise au niveau conceptuel est le formalisme **Entité-Association** (E/A) proposé par *Chen* en 1976. Grace à sa simplicité et la représentation graphique qu'il possède, il reste à ce jour l'un des plus utilisé dans la conception des systèmes d'information.

Sa diffusion lui a valu plusieurs appellations : formalisme individuel, formalisme entité-relation.

Le formalisme E/A comporte quatre concepts de base que nous détaillons dans cette partie:

- ✓ Deux concepts sont *structuraux* : **Entité** et **Association**;
- ✓ Un concept *descriptif* pour une entité ou une association: **Propriété**;
- ✓ Un concept qui *qualifie* la liaison entre une entité et une association : **cardinalité**.

6.1 Concepts de base du formalisme E/A

Dans cette partie nous introduisons les différents concepts du formalisme E/A en les illustrons à l'aide de l'exemple du SI précédent : scolarité d'une faculté.

6.1.1 Propriété

Définition (propriété) : Une propriété représente une information de base du SI. C'est le plus petit élément logique géré par le SI.

Exemple : Dans notre exemple de la scolarité, un étudiant possède les propriétés suivantes :

- ✓ un matricule,
- ✓ un nom,
- ✓ un prénom,
- ✓ une adresse,
- ✓ etc.

Ce sont donc des informations élémentaires et essentielles au bon fonctionnement du SI.

Note :

1. Ces propriétés sont souvent appelées attributs ou caractéristiques.
2. Les données identifiées dans un dictionnaire de données décrivent les propriétés. Cela étant dit, au niveau conceptuel, on n'utilisera pas la terminologie de données avec tout ce qu'elle induit comme informations telles que leurs types, natures, taille, etc. Car le niveau de ces informations est trop proche de la définition du système physique.

Règles :

1. La propriété est l'élément descriptif de l'entité type ou de la relation type.
2. Une propriété est obligatoirement rattachée à une entité à une relation.
3. Une propriété est unique dans un modèle conceptuel et ne peut être rattachée qu'à un seul concept (entité ou association).

Définition (propriété dite composée) : c'est une propriété dont la signification ou la description dépend d'autres informations.

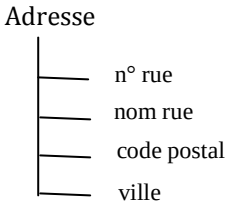
Exemples :

1. Matricule Etudiant : sexe + année 1ére inscription + n° ordre
2. Adresse : n° rue, nom rue, code postal, ville

Modélisation des propriétés composées :

Deux manières de modélisation sont proposées pour les propriétés composées :

1. Considérer cette propriété "construite" comme une propriété normale qui suit les mêmes règles que toute propriété, en précisant toutefois la règle de sa construction. Dans ce cas les informations composantes ne sont pas modélisées (pas représentées dans le modèle).
2. La définir en tant que *propriété composée* dont la composition est modélisé sous forme de propriétés composantes. Dans ce cas, chaque fraction de valeur d'une propriété composée peut être exprimée par la valeur de chaque propriété composante. Dans ce cas l'adresse par exemple sera modélisée comme suit :

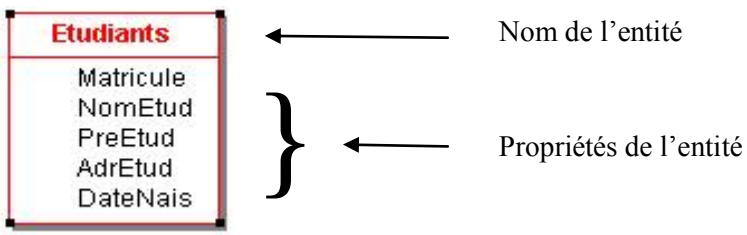


6.1.2 Entité

Définition (**entité**) : c'est une représentation d'un objet matériel ou immatériel de l'univers du SI. Elle est décrite par un ensemble de propriétés.

Exemple : un étudiant est représenté par une entité « Etudiants » décrite par les propriétés caractérisant un étudiants : Matricule, nom de l'étudiant, son prénom, son adresse et sa date de naissance.

Dans le formalisme E/A, le symbolisme retenu pour représenter une entité et ces propriétés appliqué à notre exemple est le suivant :

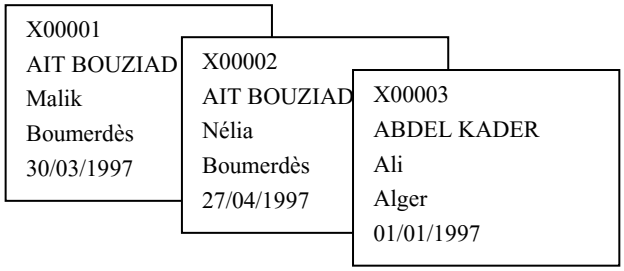


Définition : (**Occurrence d'entité**)

Chaque élément de l'ensemble représenté par une entité est appelé *occurrences d'entité.*

Une entité peut n'avoir aucune, une ou plusieurs occurrences. Pour illustrer ce terme d'«occurrence», voici un exemple de plusieurs occurrences de l'entité *Etudiants :*

Exemples d'occurrences :



Définition (*Identifiant*) :

On doit pouvoir faire référence distinctement à chaque occurrence de l'entité. Pour cela l'entité doit être dotée d'une propriété particulière appelée *identifiant.*

C'est une propriété qui doit posséder des occurrences unique et telle que, à une occurrence de l'identifiant, corresponde une seule occurrence de l'entité. Autrement dit, un identifiant permet de connaître de façon sûre et unique l'ensemble des propriétés qui décrivent l'entité

Remarque : Dans un graphe de dépendances fonctionnelle, un identifiant représente une propriété qui doit être la source de dépendances fonctionnelles pour toutes les autres propriétés de l'entité.

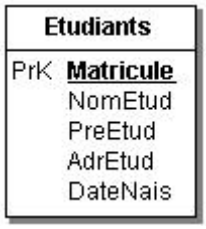
Symbolisme d'un identifiant : Dans le formalisme E/A, parmi les propriétés d'une entité, celle qui joue le rôle de l'identifiant est signalée simplement en la *soulignant.*

Propriétés d'un identifiant :

1. *univalué* : à une occurrence d'entité correspond une seule valeur pour un identifiant donné;
2. *discriminant* : à une valeur de l'identifiant correspond une seule occurrence de l'entité;
3. *stable* : la valeur d'un identifiant doit est conservée jusqu'à la destruction de l'occurrence de son entité;
4. *minimal* : pour un identifiant composé, la suppression d'un de ces composants lui ferait perdre son caractère discriminant.

Note : Certaines entités peuvent avoir plusieurs propriétés possédant les propriétés d'identifiant; celle sélectionnée par le concepteur est qualifiée d'identifiant *primaire* les autres sont qualifiés d'identifiants *alternatifs*. La notion de *clé* est parfois utilisée pour désigner un identifiant

Exemple d'identifiant: Il est facile de constater que la propriété matricule dans l'entité étudiant peut jouer le rôle d'un identifiant. Voici le schéma modifié de l'entité **Etudiants** : (l'outil JMerise utilise le préfixe Prk, acronyme de 'Primary Key' en anglais, comme commentaire indiquant l'identifiant)



Types d'identifiant : Le concepteur doit être prudent lors du choix d'un identifiant pour une entité. Plusieurs démarches sont possibles pour déterminer l'identifiant, soit :

1. une propriété « naturelle », par exemple le nom d'un pays pour l'entité pays;
2. une propriété « artificielle », inventée par le concepteur pour identifier l'entité qu'il vient de concevoir (matricule, code module, code spécialité, etc.);
3. une propriété composée en s'assurant que qu'elle ne générera pas de doublons; on parle alors d'identifiant composé; par exemple « nom + prénom + date et lieu de naissance »;
4. un identifiant relatif, par exemple n° ouvrage + n° exemplaire, pour identifier les exemplaires d'un ouvrage.

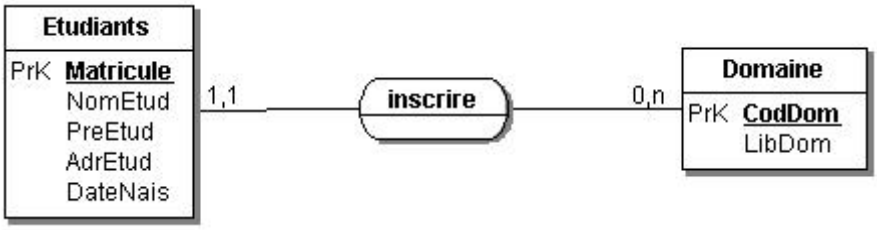
6.1.3 Association (ou Relation)

Définition (Association) : Une association définit un lien sémantique entre une ou plusieurs entités.

Le formalisme (E/A) d'une association est le suivant :

Remarque : Généralement le nom de l'association est un verbe définissant le lien entre les entités qui y sont reliées.

Exemple : Un étudiant à son entrée à l'université *s'inscrit* à un domaine pour sa formation. Si nous analysons cette phrase, on distingue les deux entités « Etudiants » et « Domaine » et un verbe « inscrire » qui met en évidence un lien entre étudiants et domaine. Dans le formalisme E/A cette phrase se traduit comme suit (nous reviendrons pus loin sur les cardinalités (1,1) et (0,n) utilisées ici) :



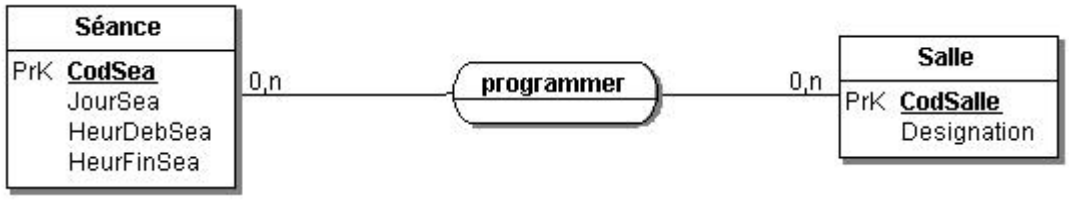
Définitions :

1. On appelle **dimension** d'une association le nombre d'entités qu'elle lie.
2. Et **collection** la liste de ces entités.

Identifiant d'une association :

- ✓ Une relation type n'a pas d'identifiant propre.
- ✓ L'occurrence d'une association est déterminée par les occurrences des entités de sa collection. Autrement dit, à une combinaison d'occurrences d'entités composant la collection d'une association, il ne peut y avoir au plus qu'une occurrence de cette association.
- ✓ L'identification d'une association est représentée par la conjonction des identifiants des entités de sa collection.

Exemple :

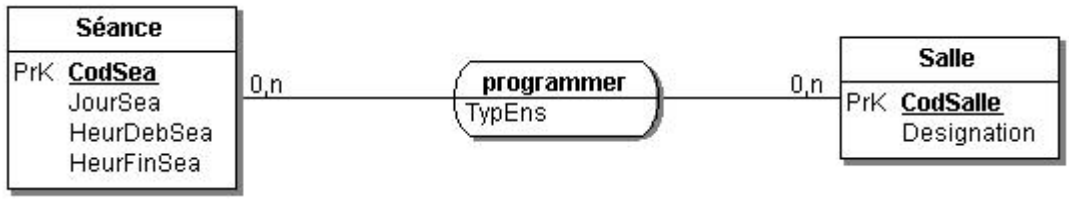


Dans cet exemple, l'association « programmer » est identifiée par la concaténation d'une valeur de *CodSea* de « Séance » et d'une valeur *CodSalle* dune « Salle ».

Une occurrence d'association ne peut exister que reliée à une occurrence de chacune des entités de sa collection.

Définition (Associations porteuses) : Une association est dite porteuse lorsqu'elle contient des propriétés

Exemple : dans notre exemple, nous savons lors de l'identification des dépendances fonctionnelles que si nous connaissons une (occurrence) salle et une (occurrence) séance alors nous connaissons le type d'enseignement qui est y programmée. Ainsi nous pouvons ajouter cette nouvelle propriété « TypEns » à l'association « programmer ». Voici comment la représenter dans le formalisme E/A.



Note : La définition de liens entre entités permet de traduire une partie des règles de gestion qui n'ont pas été satisfaites par la simple définition des entités. Par exemple, l'association «inscrire» vu ci-dessus traduit les deux règles de gestion suivantes :

• Un étudiant à son arrivée à l'université, il s'inscrit à **un et un seul** domaine,

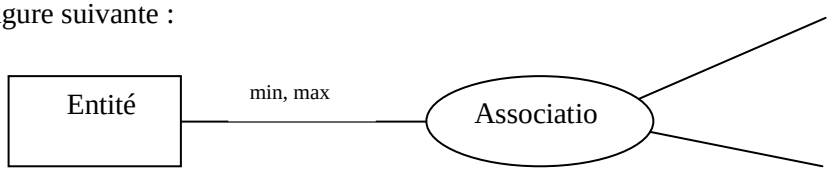
• Dans un domaine, sont inscrits **aucun, un ou plusieurs** étudiants.

Remarquons, que cette association est caractérisée par ces annotations **1,1** et **0,N** qui nous ont permis de définir les règles de gestions précédentes. Ces annotations sont appelées les **cardinalités**.

Définition (Cardinalités) :

Les cardinalités (min, max) d'une entité par apport à une association à laquelle elle participe, expriment le nombre de fois au minimum et au maximum ou une occurrence de cette entité participe aux occurrences de l'association.

Ces cardinalités min et max sont représentées dans le formalisme E/A sur le lien « entité, association » comme dans la figure suivante :



Exemple : Si on reprend l'exemple précédent, pour trouver les cardinalités (min, max) entre l'entité « Séance » et l'association « programmer », on doit ce posé les questions suivantes :

- Combien de fois au *minimum* une séance peut-elle être programmée dans les salles ? ou bien, étant donnée une occurrence de séance, dans combien de salles au *minimum* peut-on la programmer ?
- Combien de fois au *maximum* une séance peut-elle être programmée dans les salles ? ou bien, étant donnée une occurrence de séance, dans combien de salle au maximum peut-on la programmer ?

A la première question, nous pouvons répondre qu'une séance peut ne pas être programmée (ou utilisée) dans aucune salle. Ca pourrait être le cas d'une sixième séance; prévue mais pas toujours utilisée. Ainsi dans ce cas la cardinalité minimum est **0.**

A la deuxième question, nous pouvons répondre qu'une séance peut être programmée (utilisée) dans plusieurs salles donc **n.**

Valeurs des cardinalités :

Les cardinalités les plus répandues sont les suivantes :

min=0 (participation optionnelle) : certaines occurrences de l'entité ne participent pas à l'association ().

min=1 (participation obligatoire) : toute occurrence de l'entité participe au moins une fois aux occurrences de l'association.

max=1 (unicité de participation) : quand une occurrence de l'entité participe à l'association, elle n'y participe au plus qu'une fois.

max=n (multiplicité de participation): quand une occurrence de l'entité participe à l'association, elle peut y participer plusieurs fois.

Note :

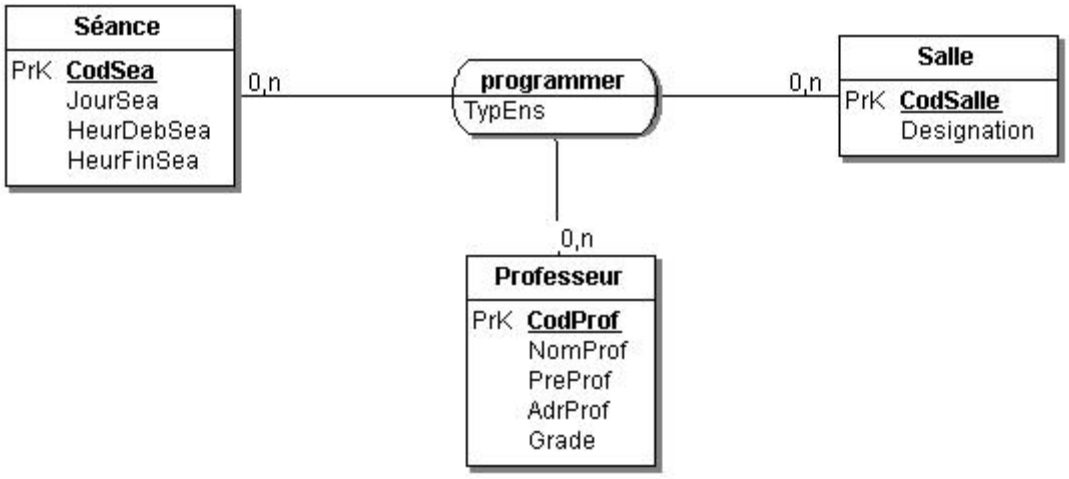
1. Théoriquement, des valeurs quelconques peuvent être affectées à ces cardinalités. En effet, on peut rencontrer des règles de gestion imposant des cardinalités avec des valeurs particulières, mais cela reste assez exceptionnel et la présence de ces cardinalités imposera l'implantation de traitements supplémentaires.
2. Toute association binaire avec cardinalité (1,1) ne peut être porteuse de propriété. En effet une telle propriété migre alors obligatoirement dans l'entité portant cette cardinalité (1,1).

Association n-aire :

En théorie la dimension d'une association est non limitée. Il est donc possible d'exprimer des associations plus que binaires (n-aires). Cependant en pratique, une grande proportion des associations modélisées est binaire ; on ne rencontre pas d'association de dimension supérieure à 7. En effet, il est admis que le cerveau humain ne peut visualiser, manipuler ou mémoriser simultanément que 6 ou 7 éléments distincts (mémoire à court terme).

Exemple :

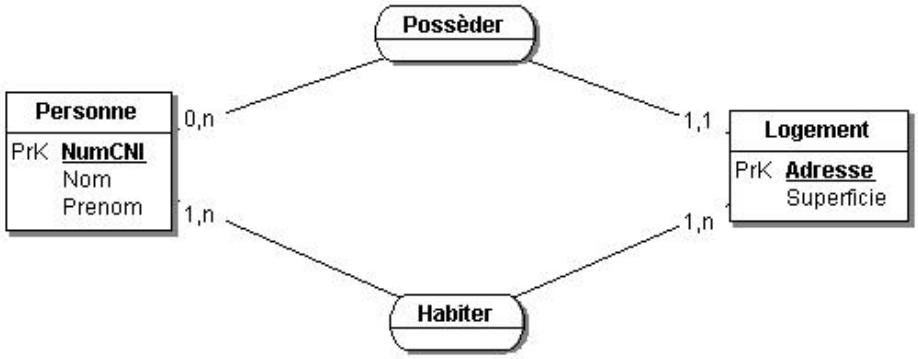
Si nous désirons connaître le professeur programmé dans une salle à une séance donnée, nous ajoutons un nouveau lien entre l'association « programmer » et l'entité « Professeur ». L'association « programmer » devient ternaire (de dimension 3) :



Partage de collection :

Plusieurs associations peuvent partager la même collection d'entités. Il s'agit associations de significations différentes entre deux ou plusieurs entités.

Exemple :



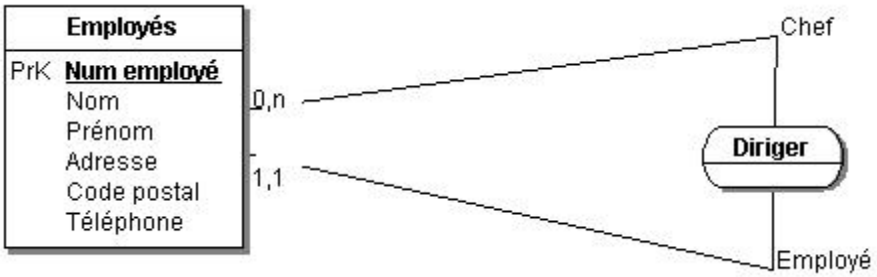
Association réflexive :

C'est une association qui lie une entité avec elle-même. Elle exprime une relation entre les occurrences d'une même entité.

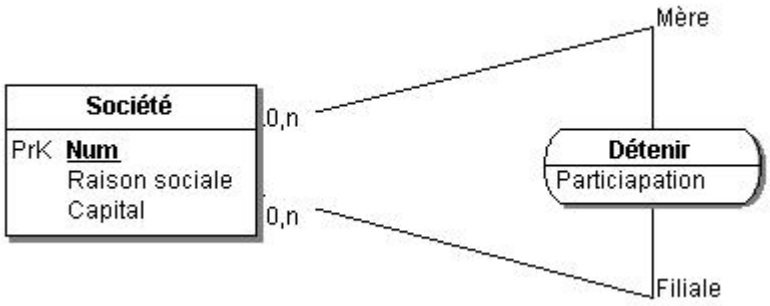
Règle : Il faut préciser sur chacun des deux liens le rôle joué par chacune des occurrences d'entité en relation.

Exemples :

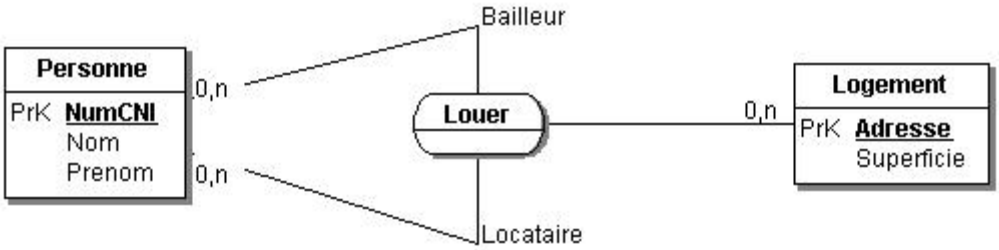
1. On désire modéliser le fait qu'un employé peut diriger d'autres employés.



2. Une société peut-être la mère avec d'autres sociétés de plusieurs filiales avec une certaine participation dans chacune d'entre elles.



3. Les termes bailleur et locataire précisent le rôle joué par chacune des occurrences d'entités personne dans l'association « louer ».



6.2 Extension du MCD