Méthode systémique : Merise (Partie 1)

Table of Contents

MERISE (Méthode d’Etude et de Réalisation pour Système d’Entreprise**)**

- MERISE (Méthode d’Etude et de Réalisation pour Système d’Entreprise) est une méthode d'analyse, de conception et de gestion de projet informatique.

- Elle née dans les années 70, développée initialement par Hubert Tardieu. Elle fut ensuite mise en avant dans les années 80, à la demande du Ministère de l'Industrie qui souhaitait une méthode de conception des SI.

- Cette méthode reste adaptée pour la gestion des projets internes aux organisations, se limitant à un domaine précis.

MERISE (Méthode d’Etude et de Réalisation pour Système d’Entreprise**)**

V.1 Introduction

MERISE est une méthode d'analyse et de conception des SI basée sur le principe de la séparation des données et des traitements.

Elle possède un certain nombre de modèles (ou schémas) qui sont répartis sur 3 niveaux :

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

1) **Niveau Conceptuel** :

Représentation des informations (les données et les opérations ou les traitements) sous forme d'un schéma ou modèle indépendamment de la solution et sans tenir compte de l'organisation de l'organisme.

On parlera de modèle conceptuel de données et de modèle conceptuel de traitements (**MCD** et **MCT**).

Le **MCD**: **Description** des **données** et des **relations** en termes de:

- Entité ou Individu
- Relation ou Association
- Propriétés ou d'Attributs

Le **MCT**: **Description** de la **partie dynamique** du ***SI*** en termes de:

- Processus
- Opérations

2) Niveau Organisationnel ou Logique :

son rôle consiste à définir l'organisation qu'il est souhaitable de mettre en place.

On y précisera les postes de travail, la chronologie des opérations et l'emploi des bases de données.

On parlera de modèle logique de données et de modèle organisationnel de traitements (**MLD** et **MOT**).

Le **MLD**:

- Le modèle «CODASYL» si une orientation base de données réseau est choisie
- Le modèle «relationnel» si une orientation base de données relationnelle est choisie
- Le modèle «hiérarchique»

Le **MOT**:

Permet de représenter par procédure les phases et les taches effectuées par chaque poste de travail

3) Le niveau physique (technique ou opérationnel)

Il consiste à apporter des solutions techniques au problème. Il consiste à se poser la question Comment ?

- Du point de vue des données on effectue des choix sur les méthodes de stockage et d'accès (fichiers physiques).
- Pour les traitements automatiques on étudie le découpage en programmes.

Niveau physique ou opérationnel :

- Représentation des **données** en mémoire sous forme des fichiers (DBase, RBase) ou sous forme de tables (Access, Oracle, MySQL, MS SQL…) et
- Représentation de **traitements** (écrans, les menus, les états, programmation, …) en tenant compte des moyens de l'organisme en termes de matériels et logiciels.

MERISE (Méthode d'Etude et de Réalisation pour Système d'Entreprise**)**

Chaque niveau doit être respecté la séparation des données du traitement, cela peut se résumer :

Les trois niveaux sont regroupés dans le tableau ci-dessous :

Niveaux	Données	Traitements
Conceptuel	**MCD** Modèle conceptuel des données	**MCT** Modèle conceptuel des traitements
Organisationnel ou logique	**MLD** Modèle logique des données	**MOT** Modèle organisationnel des traitements
Physique	**MPD** Modèle physique des données	**MOPT** Modèle opérationnel des Traitements

MERISE (Méthode d'Etude et de Réalisation pour Système d'Entreprise**)**

Exemples de niveaux d'abstraction

- **Conceptuel**

  Le client effectue une demande de service à la compagnie pour assurer son véhicule.

  Cette dernière lui propose un devis

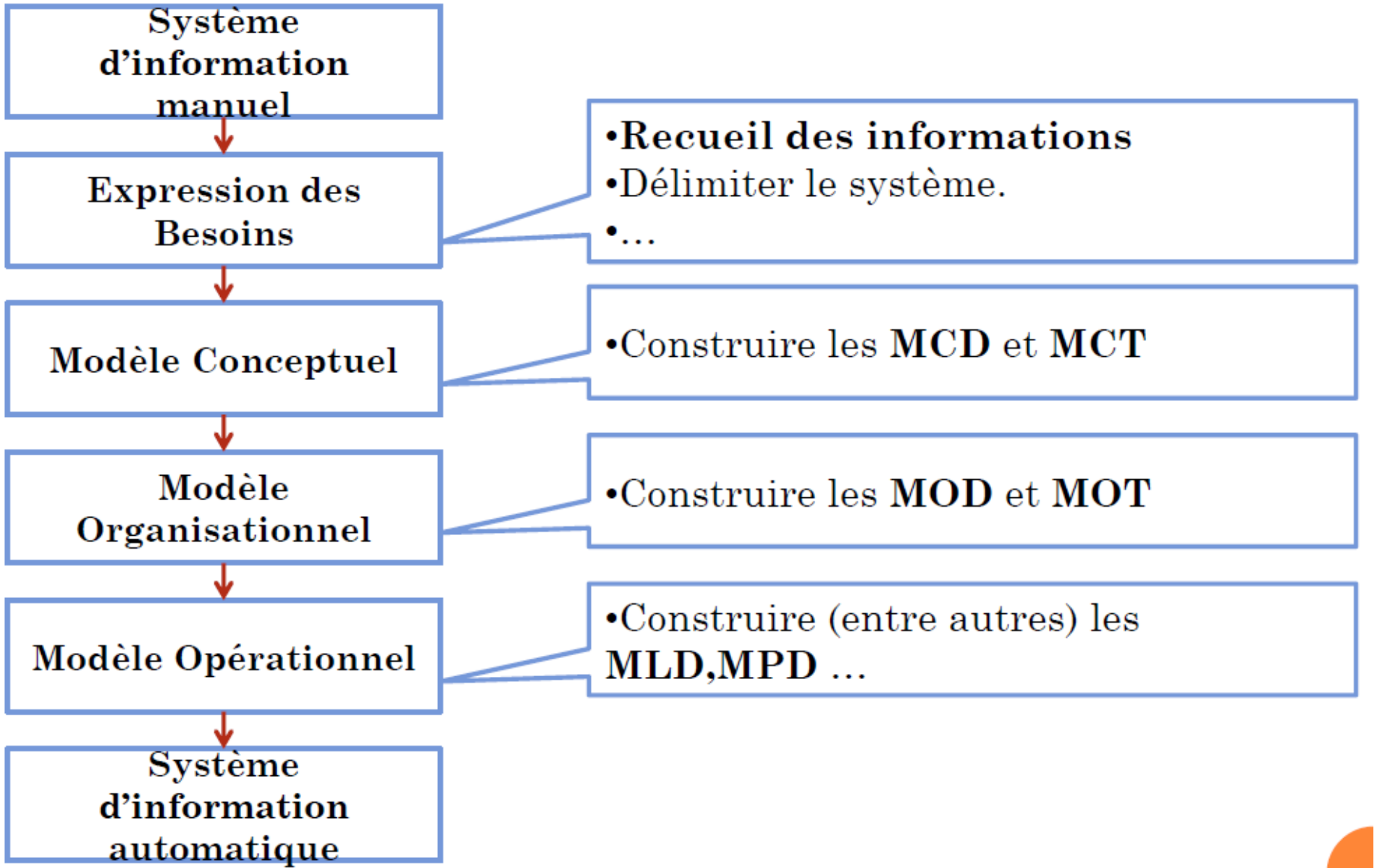
- **Organisationnel**

  Un client effectue une demande de service à l'agence de son choix, par courrier, pour assurer un véhicule. Un agent de service concerné, si le client est fiable (consultation d'un fichier central inter assurances), prend contact par téléphone pour une visite à domicile (après 17 heures) afin d'examiner plus précisément ses besoins et établir un devis

- **Physique**

  Le fichier central inter assurances est accessible par internet. Les agences sont connectées au siège de la compagnie par liaison ADSL. Chaque agence dispose de micro ordinateurs de type PC et peut traiter ses données en local grâce au SGBD Access

V.2 MERISE…CYCLE D'ABSTRACTION



Démarche à suivre

La méthode de merise propose à tout analyste, dans la conception d'un système d'information automatisable, la démarche suivante :

1- Etude de l'existant : temps consacré ≈ 50%.

2- {MCD} + {MCT, MOT} : temps consacré ≈ 25%.

3- Validation
4- MLD

} : Temps consacré ≈ 10%

5- MPD et Mopt : temps consacré ≈ 15%

Présentation des concepts et du formalisme du MCD

L’objectif d’une modélisation des données, est d’atteindre une Base de Donnée physique (*modèle interne*) contenant des informations relatives à l’activité de l’organisation.

Le modèle conceptuel des données (MCD) a pour but de représenter de façon structurée les données qui seront utilisées par le système d'information.

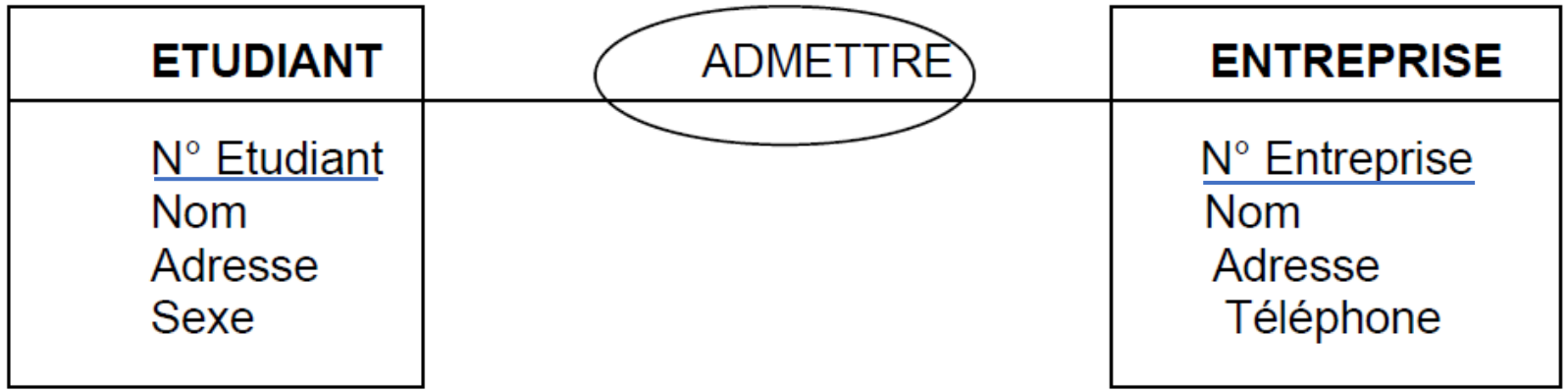
Comment modéliser les donnes

Dans MERISE, la représentation des données fait référence au modèle Entité-Association ou modèle individuel proche du langage naturel

Le modèle Entité-Association consiste à organiser les données en entité et association (relation)

Exemple:

Soit la phrase : Les étudiants sont admis dans les entreprises (dans le cadre des stages académiques)



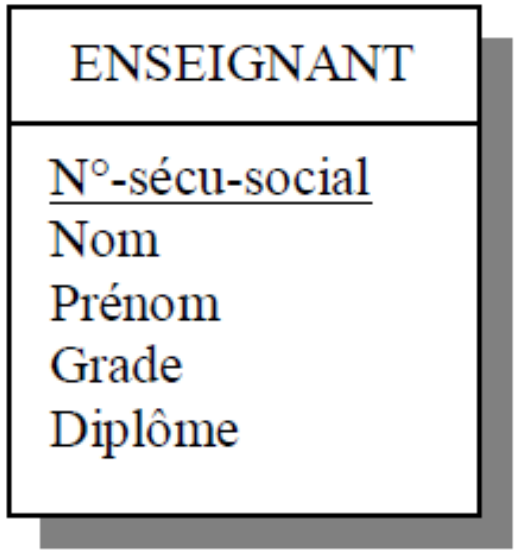
Présentation des concepts et du formalisme du MCD

Ensemble de concepts et de règles qui permettent d'exprimer le modèle conceptuel de données (MCD) et les modèles externes.

Concepts de base

- Entité : Une entité représente un objet du SI (acteur, document, concept, …),
- Une propriété : appelée aussi caractéristique ou attribut. Elle décrit un individu ou une relation.

Exemple :



- Une clé est **toujours soulignée** et en **tête** de liste des propriétés

Présentation des concepts et du formalisme du MCD

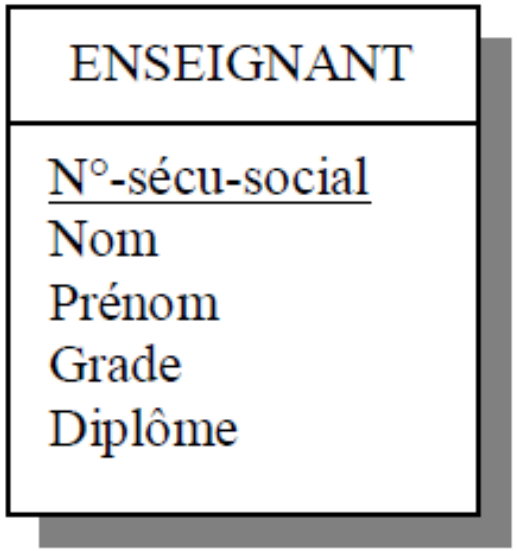
Concepts de base

❖ Identifiant (Clé)

Cette propriété (ou ensemble de propriétés) doit être :

- Non-nul (doit exister pour toutes les occurrences)
- Unique (chaque valeur est donnée une et une seule fois)
- Stable (ne doit pas changer dans le temps)

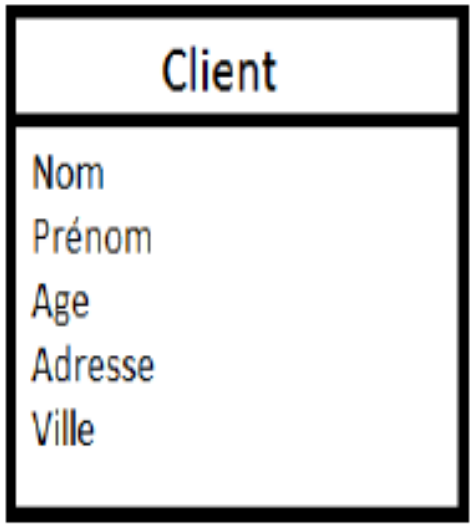
Exemple :



❖ Une clé est **toujours soulignée** et en **tête** de liste des propriétés

Concepts de base

- Une occurrence (un enregistrement) d'une entité



Exemples d'occurrences d'une entité: Client

Client	
Nom	: Amin
Prénom	: Jamil
Age	: 33
Adresse	: Rue Farah
Ville	: Nador

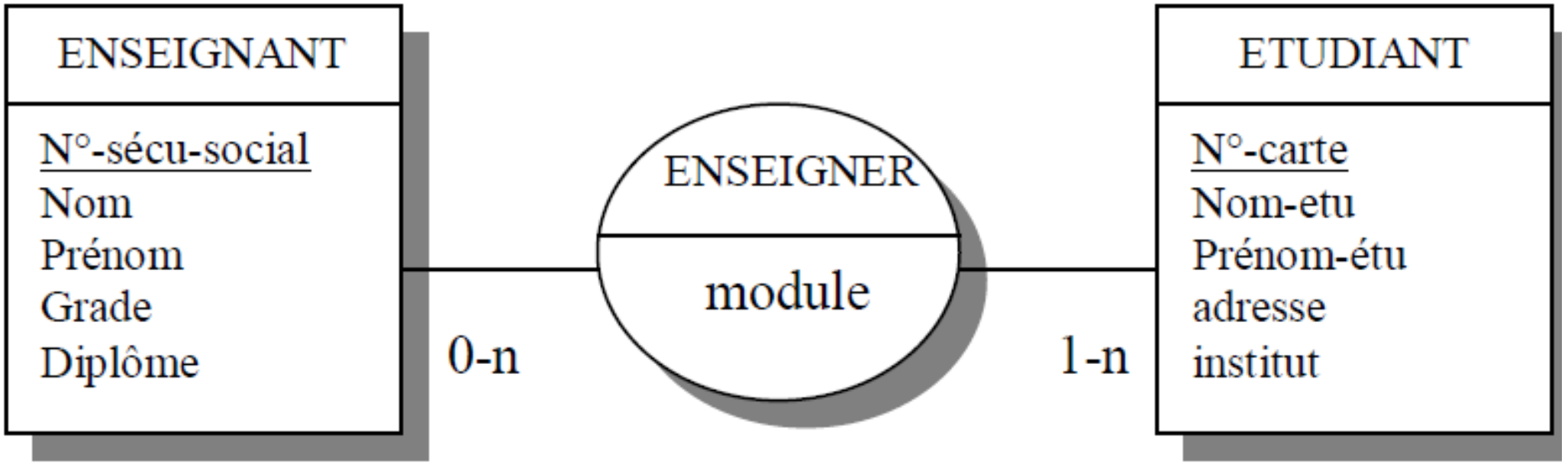
Client	
Nom	: Mohamed
Prénom	: Salim
Age	: 45
Adresse	: Rue M5
Ville	Oujda

Client	
Nom	: Tarik
Prénom	: Tribek
Age	: 26
Adresse	: Rue Maarif
Ville	: Casa

Associations (relations)

- **Relation** est une association entre des entités (objets).
- Une **association** (**relation**) peut avoir des **propriétés**

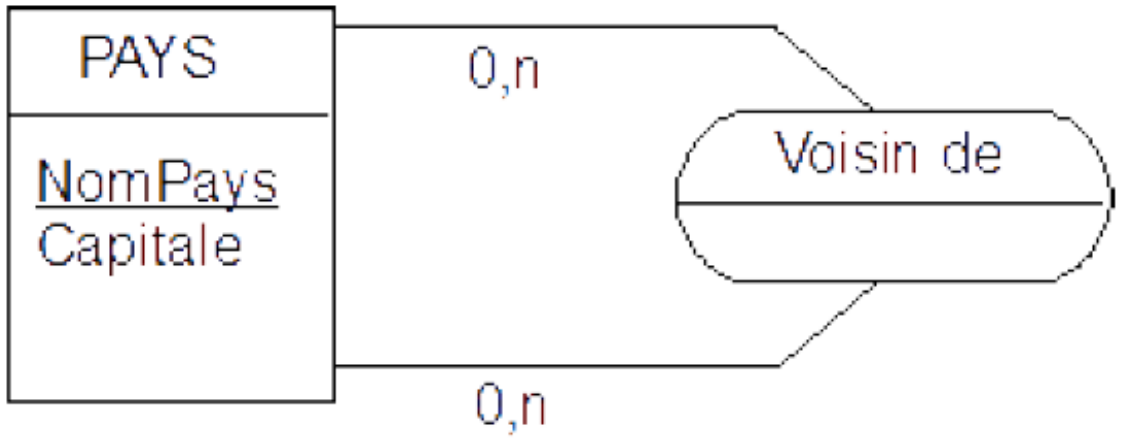
Exemple :



Types de relation:

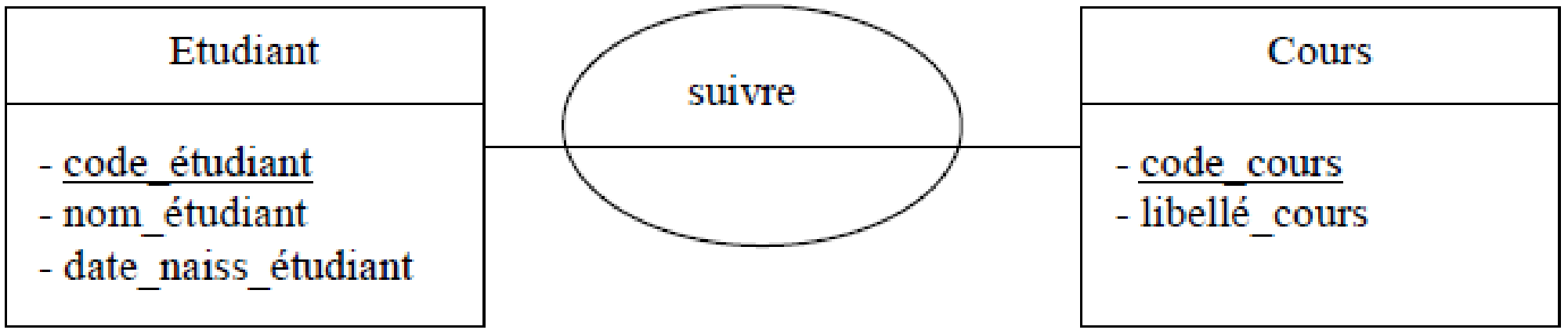
Associations réflexives

Une association qui lie une entité à elle-même est dite **unaire** (réflexive).



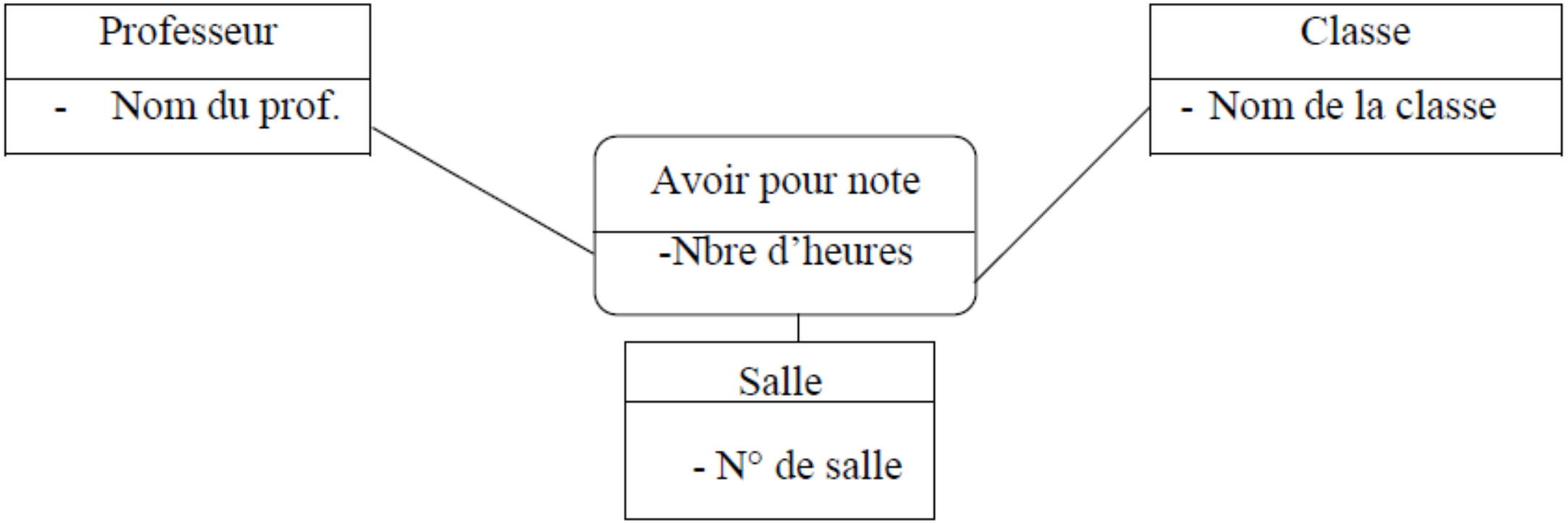
Types de relation:

Relation de dimension 2



Types de relation:

Relation de dimension 3

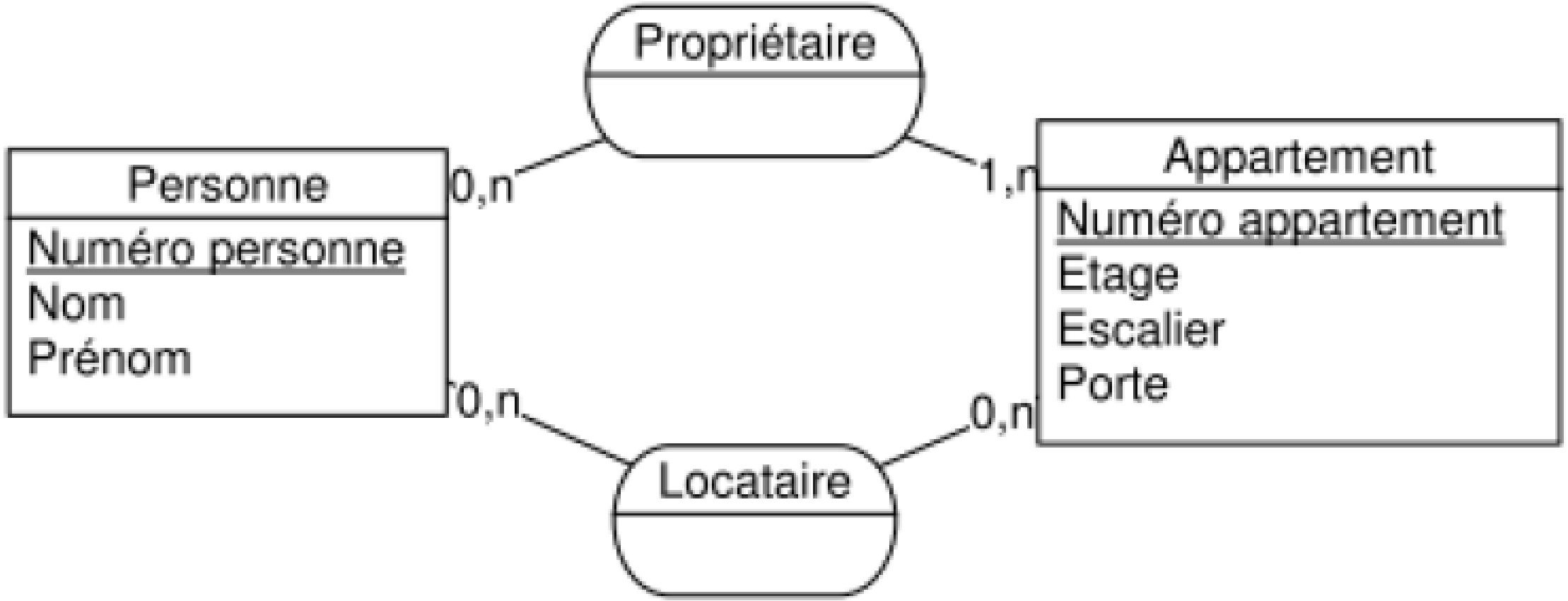


Remarques :

- ❑ En pratique il est déconseillé de dépasser la dimension 4 car ceci posera des problèmes lors du passage aux niveaux logique et physique.
- ❑ Il est souhaitable de décomposer les relations complexes par les méthodes de décomposition qu'on verne pas la suite.

❖ Associations (relations)

Il peut y avoir plusieurs associations type liant les mêmes entités si la sémantique est différente



Contraintes fonctionnelles

Cardinalités

Cardinalités d’un objet

- Les cardinalités d’un objet dans une relation mesurent le minimum et le maximum de participation de l’ objet à la relation.

- Càd, est l’ensemble des occurrences de cet objet.

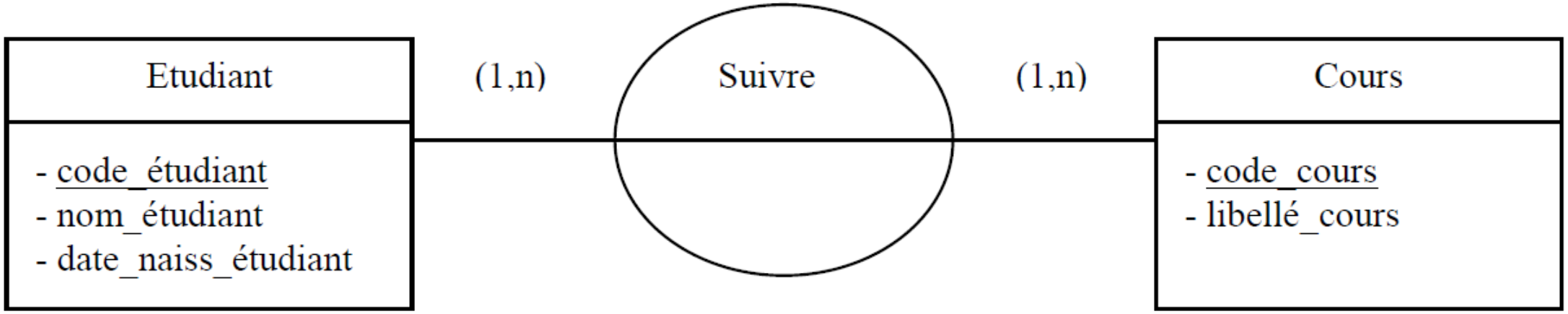
Types de cardinalité

- (0,1) : Une occurrence de l’objet participe au plus 1 fois à la relation.
- (1,1) : Une occurrence de l’objet participe toujours 1 et 1 seule fois à la relation.
- (1,n) : Une occurrence de l’objet participe au moins 1 fois à la relation.
- (0,n) : Aucune précision quant à la participation des occurrences de l’objet à la relation.

Contraintes fonctionnelles

Cardinalités

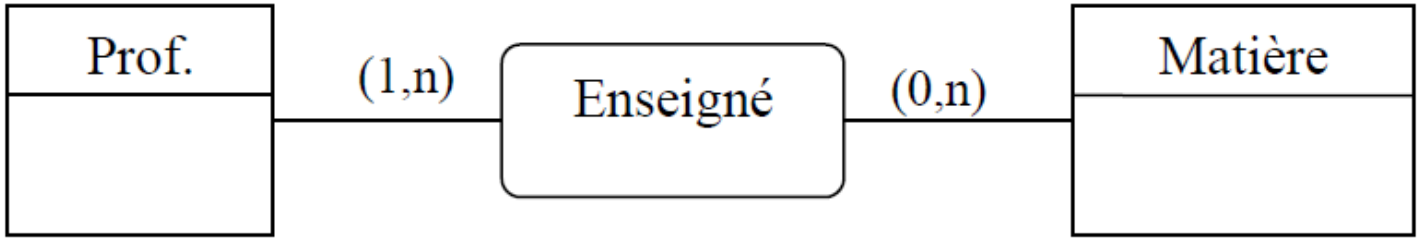
Exemple

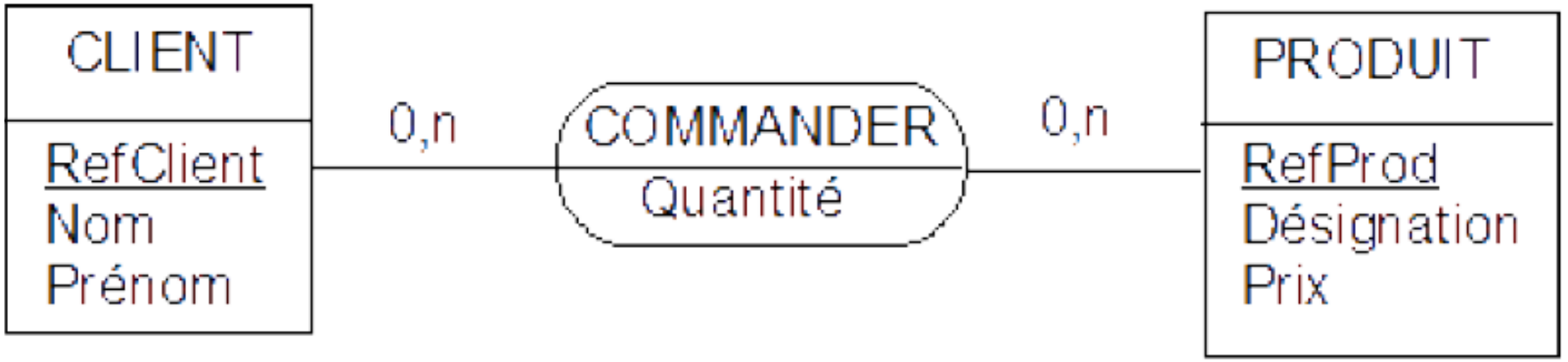


- Un étudiant suit **1** ou **plusieurs** cours ;
- Un cours est suivi par **1** ou **plusieurs** étudiants.

Contraintes fonctionnelles

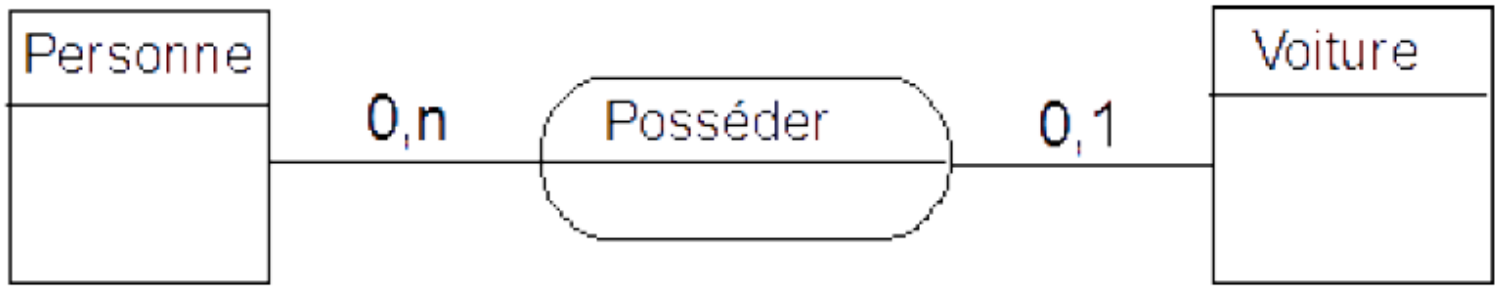
Cardinalités

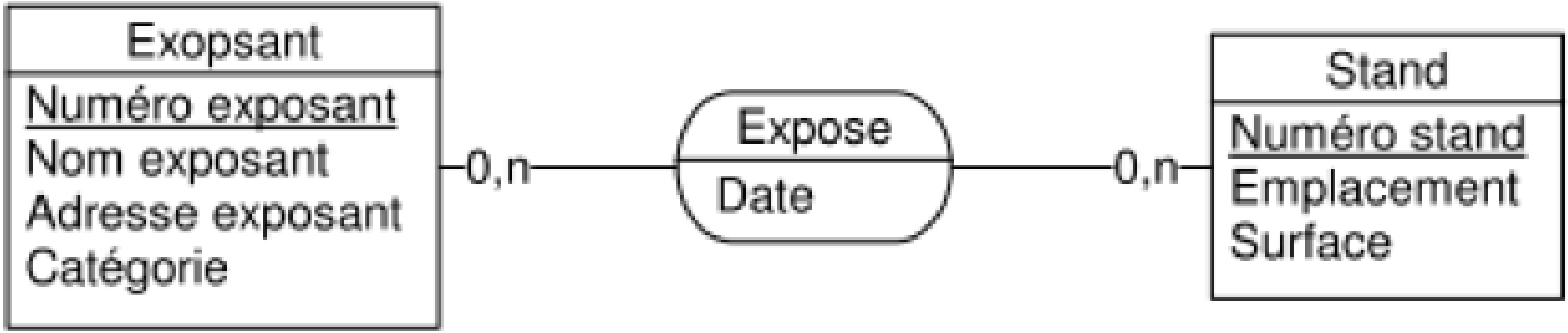




Contraintes fonctionnelles

Cardinalités





Les règles de gestion

Les règles de gestion du MCD précisent les contraintes qui doivent être respectées par le modèle.

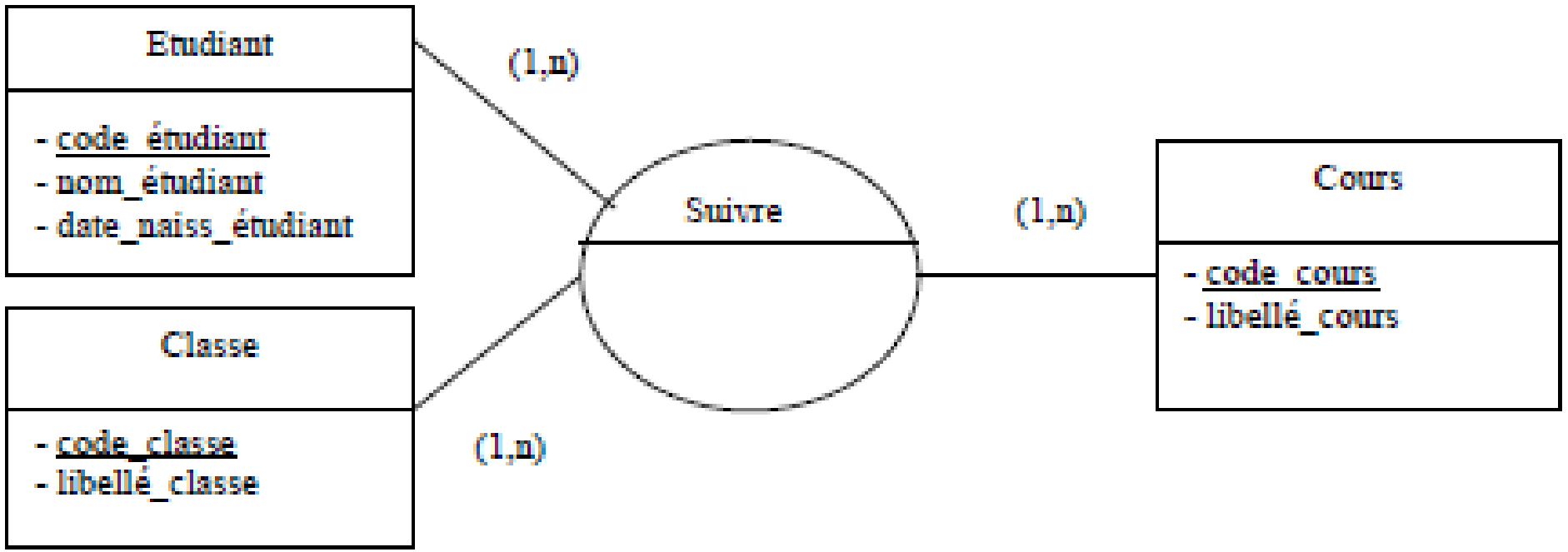
Exemple:

Règle de gestion 1 : tout étudiant d’une classe suit au moins un cours

Règle de gestion 2 : tout cours est suivi dans au moins une classe

Règle de gestion 3 : tout étudiant d’une classe à l’autre peut suivre différents cours.

Les règles de gestion



Les règles de gestion

Les règles de gestion expriment les contraintes d'intégrité qui représentent les lois de l'univers réel du SI.

- Une contrainte d'intégrité porte sur une propriété, une entité ou une relation. Elle peut être **dynamique** ou **statique**.
- Les contraintes d'intégrité statiques expriment souvent des règles de calcul.

Exemple :

- $date\_naiss < date\_décès$;
- individu ne peut être homme et femme.

Les dépendances fonctionnelles

On dit qu'une propriété B est dépendant fonctionnelle d'une autre propriété A, si la connaissance de la valeur de A détermine d'une manière sure (et unique) une valeur de B.

Et on note $A \rightarrow B$ (elle se lit "$\mathrm{A\ détermine\ B}$")

Exemple 01:

Soit les propriétés suivantes : Code_Etudiant, Nom_Etudiant, Prenom_Etudiant, Date de Naissance_Etudiant, Code_Module, Intitulé du Module, Coefficient_Module, Crédits Module

Les dépendances fonctionnelles

Exemple 01:

Nous pouvons définir les dépendances fonctionnelles suivantes:

Premier sous-ensemble:

- Code Etudiant → Nom Etudiant
- Code Etudiant → Prénom Etudiant
- Code Etudiant → Date de Naissance Etudiant

❖ Si on connait le **code de l'étudiant**, nous pouvons connaitre d'une manière sûre le **nom**, le **prénom** et la **date de naissance** de **l'étudiant**.

Les dépendances fonctionnelles

Exemple 01:

Deuxième sous-ensemble

- Code Module → Intitule Module
- Code Module → Coefficient Module
- Code Module → Crédits Module

❖ Si on connait le **code du module**, nous pouvons connaitre d'une **manière sûre l'intitule du module**, son **coefficient** et ses **crédits**.

❖ Il n'y a aucune dépendance fonctionnelle entre les deux sous-ensembles.

Les dépendances fonctionnelles

Exemple 02:

Reprenons l'ensemble des dépendances fonctionnelles de l'exemple précèdent, on ajoute la dépendance suivante:

- Code Module, Code Etudiant → Note

❖ Si on connait le **code du module** et le **code de l'étudiant**, alors on peut connaitre la **note obtenue** par l'étudiant dans ce module.

Graphe des Dépendances Fonctionnelles (*GDF*)

L’ensemble des dépendances fonctionnelles d’une application peut être représente par une arborescence appelée graphe des dépendances fonctionnelles.

Graphe des Dépendances Fonctionnelles (*GDF*)

Exemple

Si on reprend l'exemple 02 (extension de l'exemple 01) cite ci-dessus, le GDF sera comme suit :

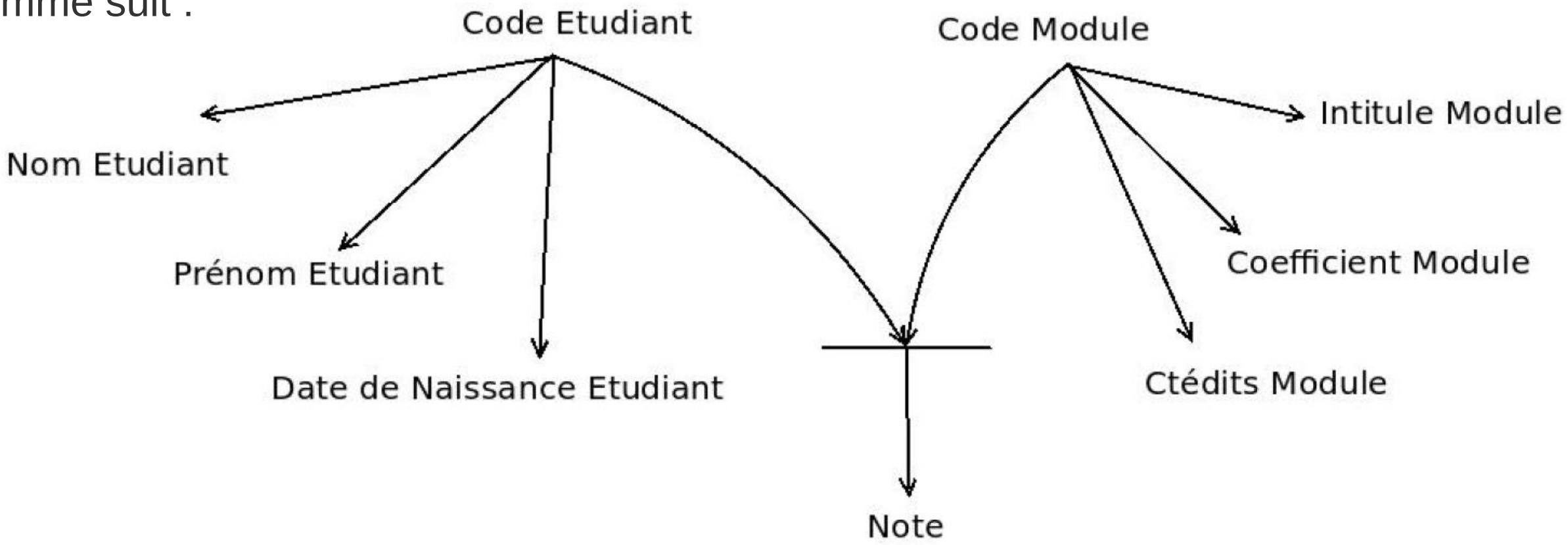


Figure 1. Graphe des Dépendances Fonctionnelles de l'exemple 02

Propriétés des Dépendances Fonctionnelles

1) **Réflexivité :**

$$Si\ Y\ est\ un\ sous-ensemble\ de\ X\ alors\ X \rightarrow Y$$

2) **Augmentation :**

$$Si\ X \rightarrow Y\ alors\ XZ \rightarrow YZ$$

3) **Transitivité :**

$$Si\ X \rightarrow Y\ et\ Y \rightarrow Z\ alors\ X \rightarrow Z$$

4) **Union :**

$$Si\ X \rightarrow Y\ et\ X \rightarrow Z\ alors\ X \rightarrow YZ$$

Propriétés des Dépendances Fonctionnelles

Pseudo-transitivite :

- Si $X \rightarrow Y$ et $WY \rightarrow Z$ alors $WX \rightarrow Z$

Decomposition :

- Si $X \rightarrow Y$ et Z est un sous-ensemble de Y alors $X \rightarrow Z$

La couverture minimale

La couverture minimale est l’ensemble des dépendances ou chaque dépendance fonctionnelle ne peut pas être déduite a partir des autres dépendances fonctionnelles.

Normalisation du modèle

La normalisation du modèle permet d'enlever toute anomalie qui peut rendre le modèle incorrect (difficile, ou même impossible, à implémenter et à gérer).

- ❖ L'anomalie la plus courante est la redondance de l'information.

La redondance dans le modèle peut engendrer :

- ❑ **Des erreurs lors de mise à jour**: puisque l'information est répétée, il est possible qu'au moment de mise à jour, quelques copies soient mises à jour tandis que d'autres ne le seront pas (garde l'ancienne valeur).
- ❑ **Lectures incohérentes**: si des erreurs de mise à jour sont survenues, il est possible de lire des valeurs différentes (incohérentes) pour la même information.

La validation de l'MCD par les formes normales (Normalisation du modèle)

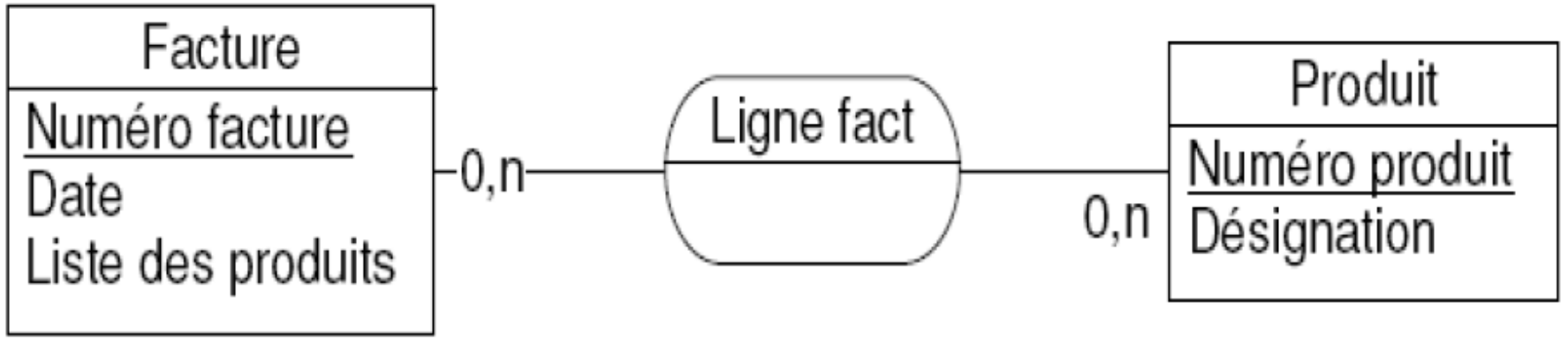
Règles de vérification appelées formes normales

1) **Première forme normale (1FN)**

- Toutes les entités et les associations possèdent un identifiant (clé).
- Aucune propriété n’est à valeurs multiples (propriétés atomiques ou élémentaires)

Exemple

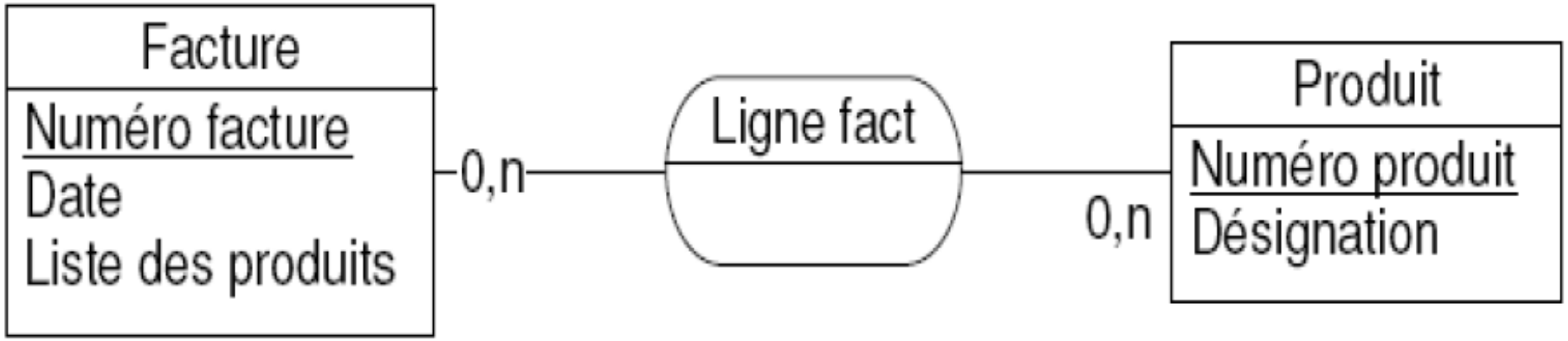
Ici, liste des produits n’est pas atomique, c’est une liste



La validation de l'MCD par les formes normales (Normalisation du modèle)

1) Première forme normale (1FN)

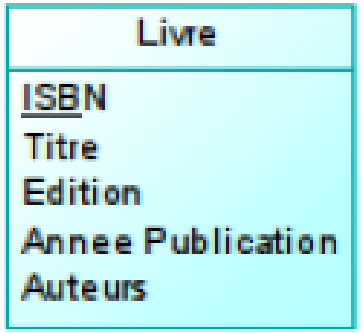
Exemple



La validation de l'MCD par les formes normales (Normalisation du modèle)

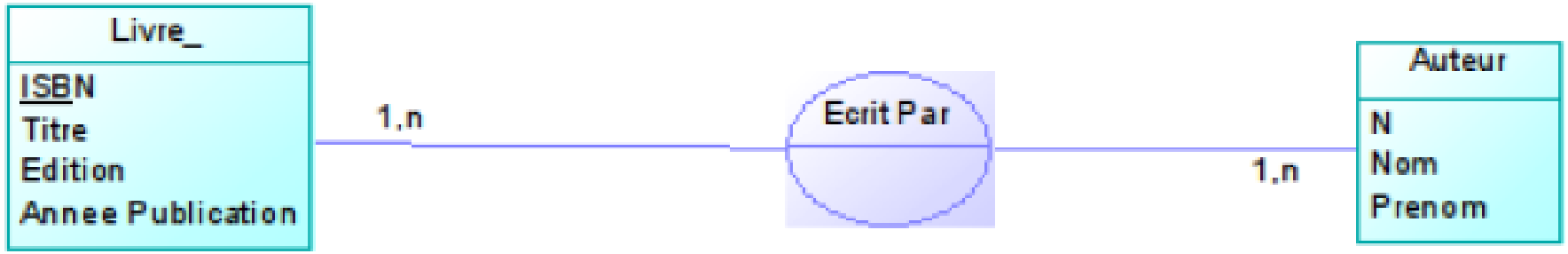
1) Première forme normale (1FN)

Exemple: Soit le modèle suivant :



- Si on considère qu'un livre peut avoir plusieurs auteurs, le modèle deviendra non normalise (ne respecte pas la première forme normale).
- Pour le normaliser, la propriété "Auteur" doit être extraite comme une entité.

Le modèle devient alors :



2) Deuxième forme normale (2FN)

- ❑ Le modèle doit être en 1FN
- ❑ Toutes les propriétés doivent dépendre de toute la clé et non pas d’une partie de la clé.

2) Deuxième forme normale (2FN)

Exemple

Soit le modèle suivant :

- Il est clair que la propriété "Nombre étages" dépend du "N° Bloc", c'est une propriété du bloc et est complètement indépendante de la notion d'appartement.
- Ainsi, ce modèle ne respecte pas la 2FN.

2) Deuxième forme normale (2FN)

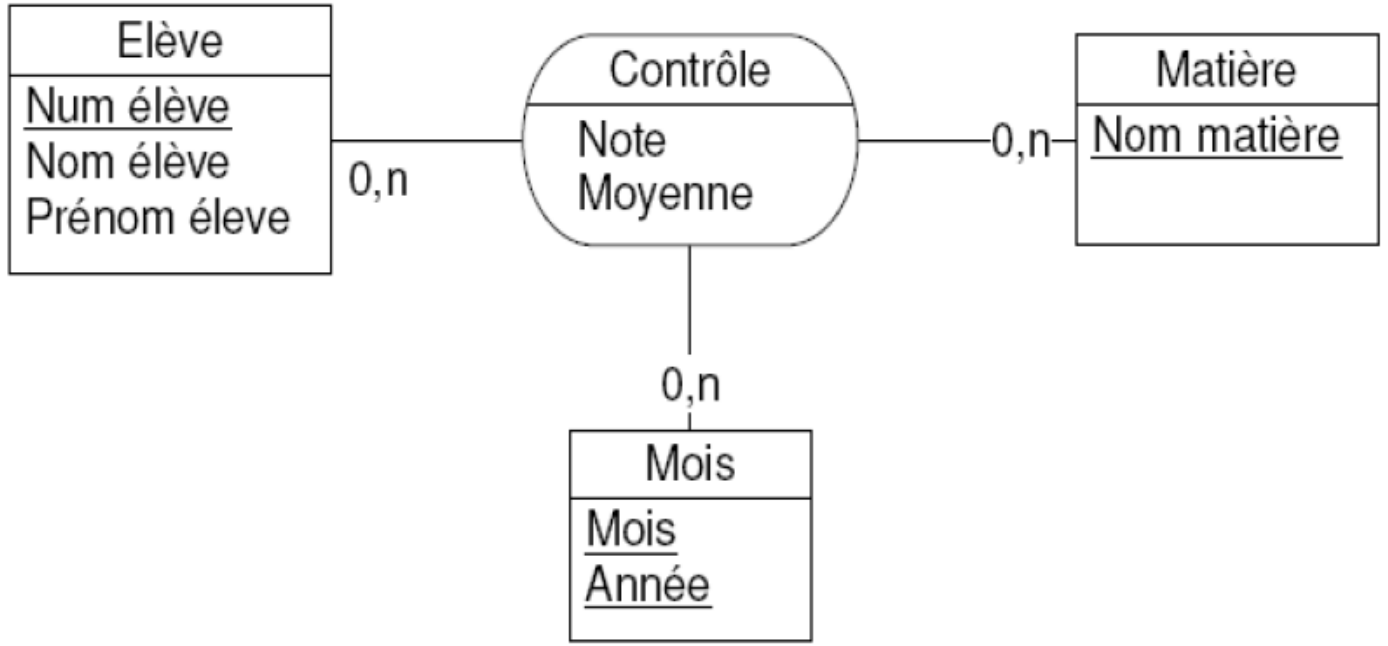
Pour le normaliser, la Dépendance Fonctionnelle qui cause problème doit être représentée par une entité séparée.

Modèle en 2FN devient alors :

- si la clé de l'entité est élémentaire (une seule propriété) alors l'entité respecte la 2FN automatiquement.

2) Deuxième forme normale (2FN)

Exemple:



Ici, d'après le schéma on a $NumEleve, NomMatière, Mois, Année \rightarrow Moyenne$

Ou, $NumEleve$ et $NomMatière$ suffisent

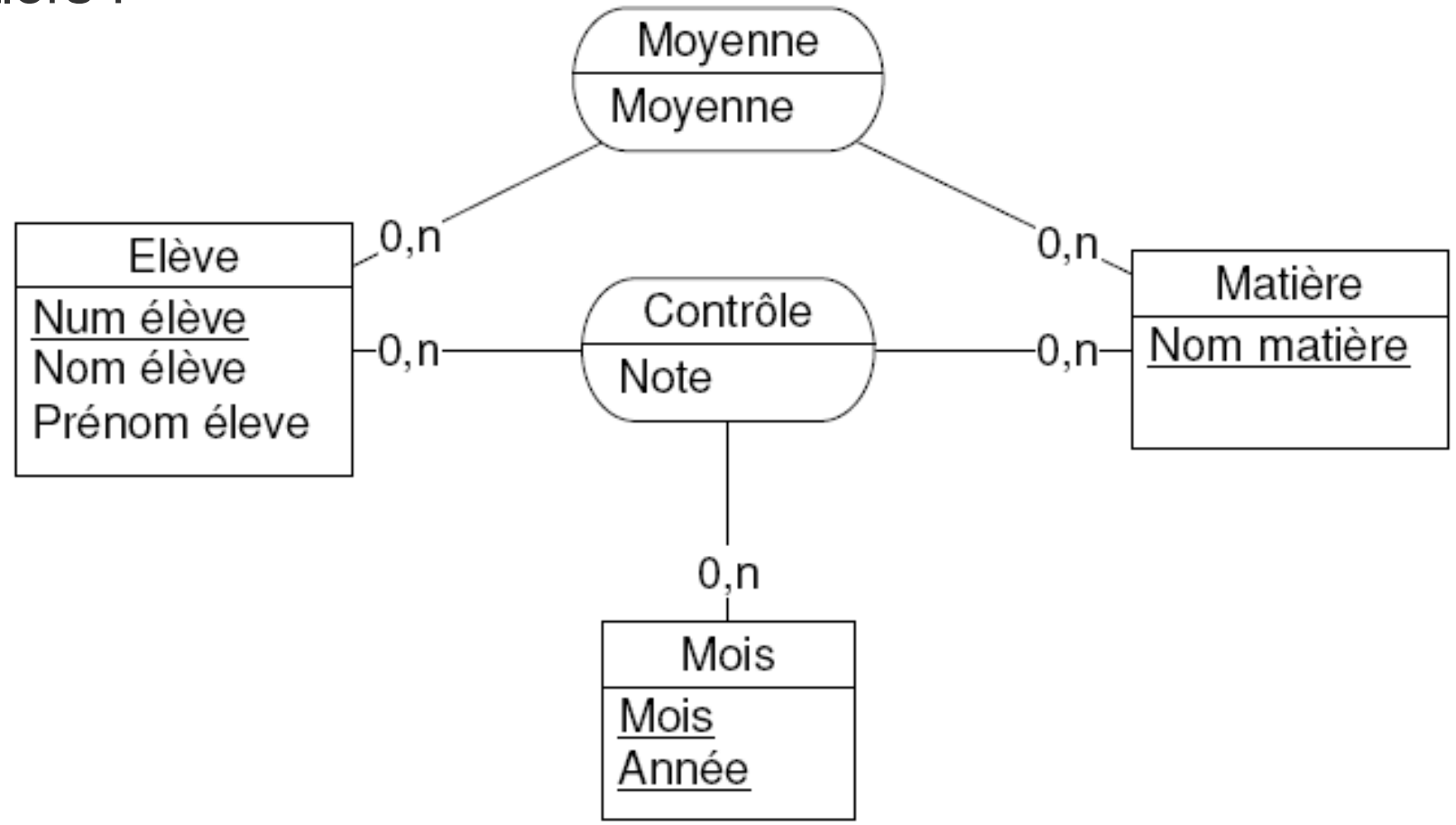
- Ainsi, ce modèle ne respecte pas la 2FN.

2) Deuxième forme normale (2FN)

Exemple:

Pour le normaliser, la Dépendance Fonctionnelle qui cause problème doit être représentée par une nouvelle association entre les deux entités suffisantes.

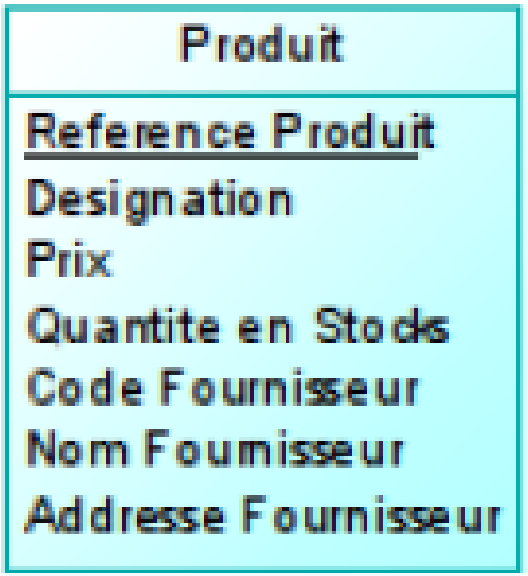
Modèle en 2FN devient alors :



3) La troisième forme normale (3FN)

- Le modèle doit être en 2FN,
- Tout attribut n'appartenant pas à la clé ne dépend pas d'un attribut non clé.

Exemple:



Les propriétés **Nom_Fournisseur** et **Adresse_Frounisseur** dépendent de l'attribut **non-clé** $\boldsymbol{Code\_Fournisseur}$.

En effet, les dépendances sont clairement :

- $\mathbf{Reference\_Produit} \rightarrow \mathbf{Code\_Frounisseur}$
  (si on connait le produit, on peut identifier d'où il a été acheté),

- $\boldsymbol{Code\_Fournisseur} \rightarrow \boldsymbol{Nom\_Fournisseur, Adresse\_Fournisseur}$
  (si on connait le code du fournisseur, on peut connaitre son nom et son adresse).

3) La troisième forme normale (3FN)

Exemple:

Pour le normaliser, les dépendances fonctionnelles qui provoquent l'erreur doivent être représentés par une nouvelle entité. Le modèle devient alors :