Analyse Merise : procédure simplifiée

1. Création du Dictionnaire de données
2. Création du Modèle Conceptuel de données (MCD)
3. Création du Modèle Logique de données (MLD)

Analyse Merise : implémentation rapide

1. Dictionnaire de données

- Description **abstraite** du système d'information dans un schéma

2. MCD (Modèle Conceptuel de données)

Éléments : **Classes d'Entités, Classes d'Associations, Cardinalités**

- Description **abstraite** du système d'information dans un schéma
- Ne tient pas compte du SGBD et du langage de programmation utilisé pour y accéder

3. MLD (Modèle Logique de données)

Eléments : **Tables** et **Liens entre les tables**

- Description du système d'information **compréhensible par un SGBD (logiciel)**
- Appelé aussi **modèle Relationnel**
- Précède à la réalisation physique dans un SGBD (ex: Oracle, SQL Server, Access, NexusDB...)

1. Dictionnaire des données

- Le première pas, on le fait avant la création du MCD

- **Inventaire** des données du domaine de la BD

Qu'est- ce qu'il faut stocker dans la BD? Quel est le type de chaque donnée? Au départ on aura un tas d'information sans catégoriser…

Ex: BD d'une bibliothèque: livres, titres, date d'emprunt, auteur, nombre de pages, nationalité de l'auteur, etc…

2. MCD : Implémentation

2.1. Définir les Clases d'Entités
2.2. Définir les Classe d'Association
2.3. Définir les Cardinalités

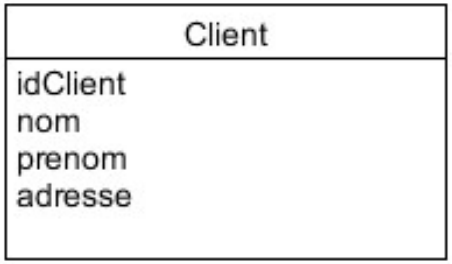
2.1. Définir les Classes d'Entités

Entité: tout objet concret ou abstrait ayant une existence propre

Ex : le client «Jean Dupont »

Classe d'entité (ou entité-type): des catégories pour le entités décrites par les mêmes propriétés (=attributs)

Ex : la classe qui sert de modèle à tous les clients, dont «Dupont» est une occurrence (ou instance)



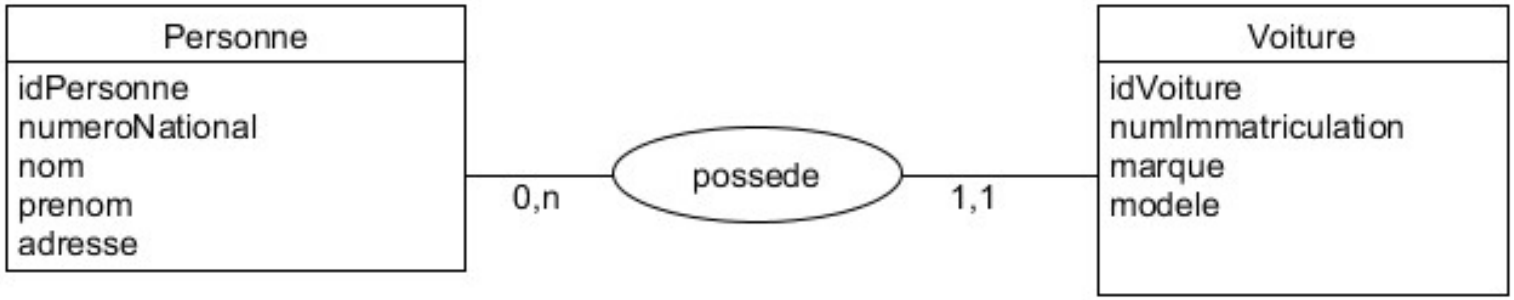
En plus des propriétés de base, créez par système une propriété **identifiant.** Vous pouvez choisir une propriété dont la valeur ne se répète jamais (ex: numéro national) mais pour faciliter l'implémentation on rajoutera toujours un idNomClasseEntite Ceci sera un code numérique dans le système de gestion de la BD qui ne se répètera jamais (ici **idClient**)

2.2. Définir les Classes d'Association

Association : lien entre deux entités

Ex: le lien entre la Personne Dupont et la voiture immatriculée 1367BS

Classe d'association (ou association-type) : regroupe toutes les associations constituées des mêmes types d'entités jouant le même rôle dans l'association

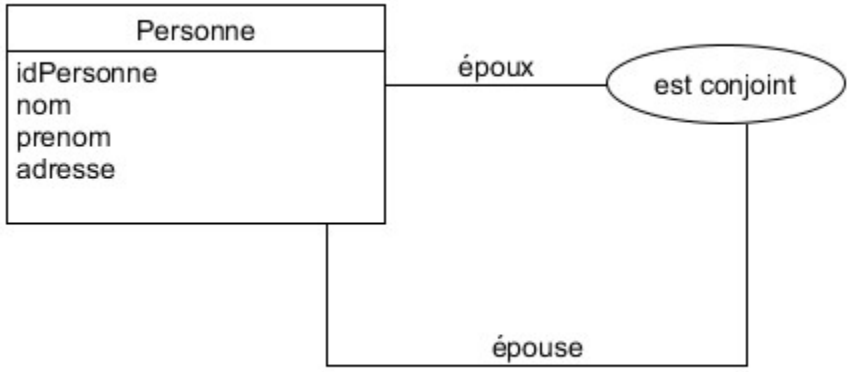


- On peut avoir **plusieurs** classes d'**associations sur** les **mêmes classes d'entités**.

  Ex: **achète** (CLIENT, VOITURE)
  et **conduit** (CLIENT, VOITURE)

- On peut avoir une classe **d'association sur une seule classe** d'entités (on parle d'**association 'réflexive').** On ajoute souvent dans ce cas des noms de **rôles** pour distinguer les deux occurrences.

  Ex: **estConjoint** (PERSONNE, PERSONNE)



2.3. Définir les Cardinalités

Cardinalités: valeurs qui indiquent, pour une entité d'une classe, le nombre minimal et maximal d'entités de l'autre classe avec lesquelles cette entité peut être en relation

Les cardinalités sont l'élément essentiel pour définir la sémantique (signification) des données car elles détermineront le nombre et la structure des tables de la BD

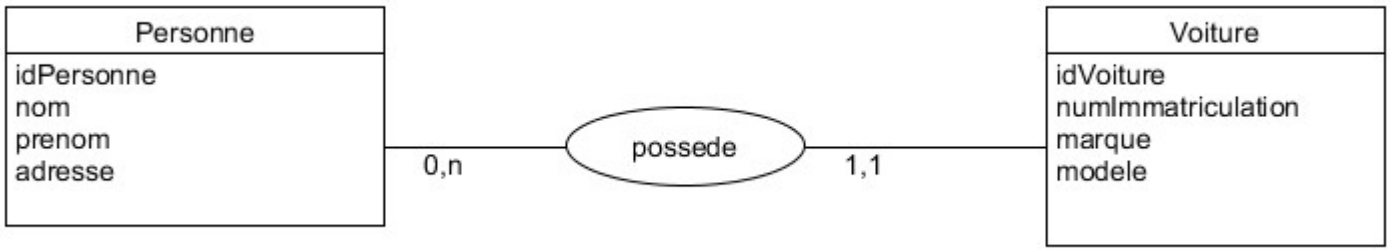
En gros, elles déterminent le nombre minimal et maximal d'entités qui peuvent être liées entre deux classes d'entités

Ex: possède (PERSONNE [0,n], VOITURE [1,1])

Dans un système informatique d'un concessionnaire :

- une **Personne** (client enregistré dans la BD qui n'a pas encore acheté) possède au minimum 0 voitures et au maximum plusieurs voitures (indiqué par **n**)

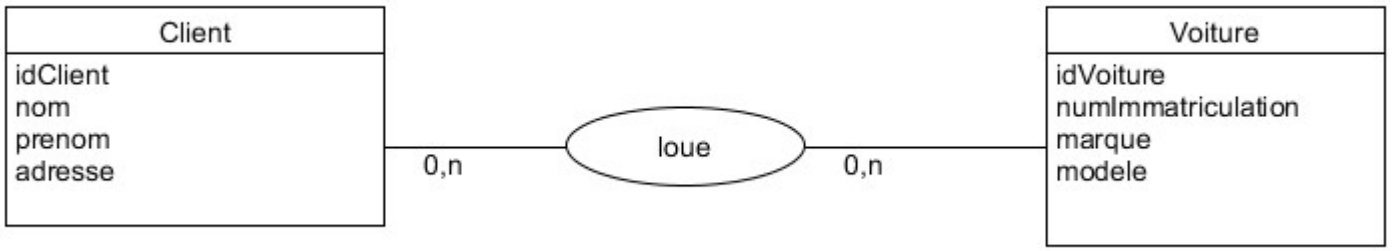
- une **Voiture** "est possédée" par 1 personne au minimum et (considérons le cas d'un seul propriétaire) par 1 personne au maximum



Ex: loue (PERSONNE [0,n], VOITURE [0,n])

Une personne loue de 0 à n voitures; une voiture est louée au moins par 1 et au maximum par plusieurs (n) personnes.

Une cardinalité minimale 0 implique qu'on peut considérer une occurrence d'entité sans occurrence d'association (ex: la personne existe dans la base de données même si elle n'a pas loué de voiture)

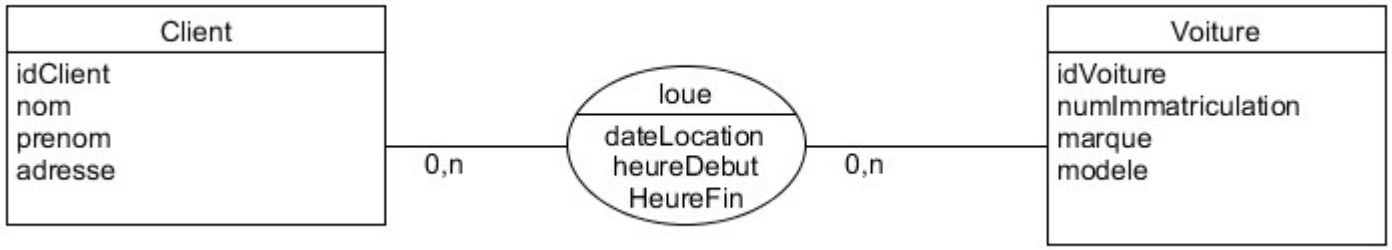


Attributs d'association

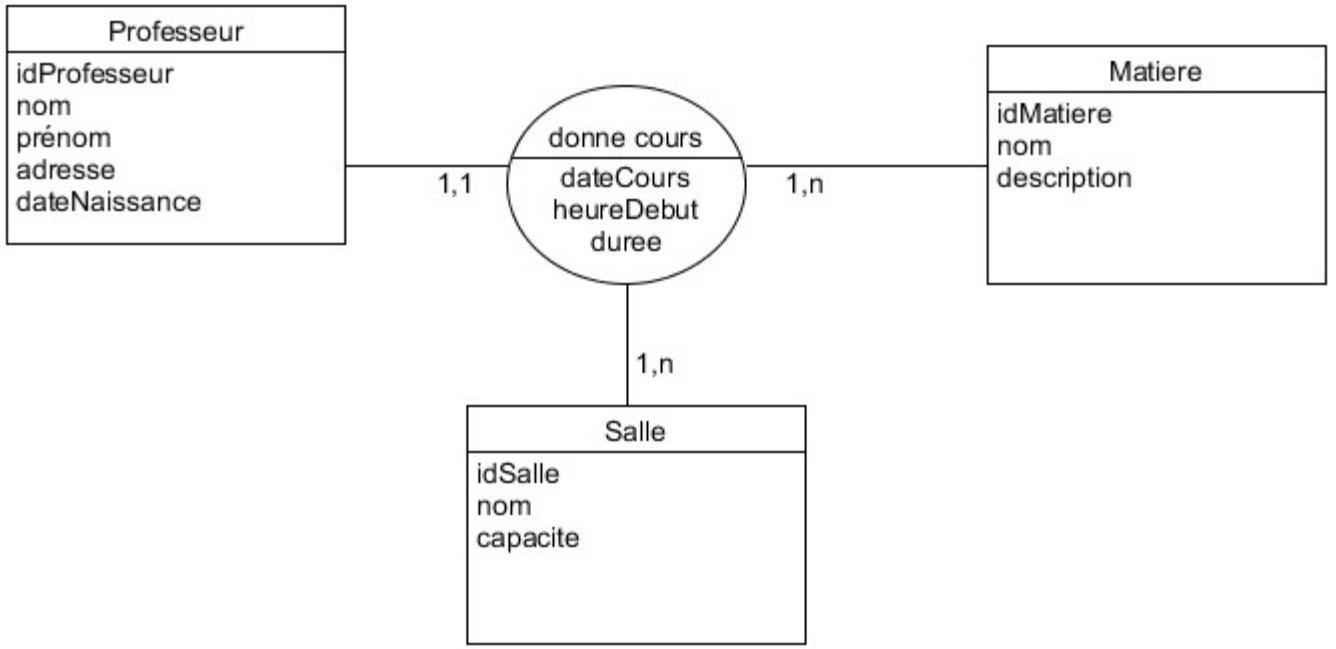
Une classe d'association peut avoir de propriétés. Pour savoir si une propriété doit se trouver dans la classe d'associations, posez vous cette question :

Est-ce que la propriété a besoin d'une entité de chaque côté pour exister? Si c'est le cas, il s'agit d'une propriété de la classe d'association et pas d'une des classes d'entités

Ex: dans cet exemple, une date de location ne peut pas se trouver ni dans Client (quelle voiture il loue?) ni dans Voiture (qui l'a loué?)



- On peut avoir une classe d'association définie **sur plus de deux** classes d'entités (association n-aire ou « d'arité » n ou de dimension n ou à « n pattes »).



Attention: les arités élevées sont rares. Elle dénotent souvent des faiblesses dans l'analyse.

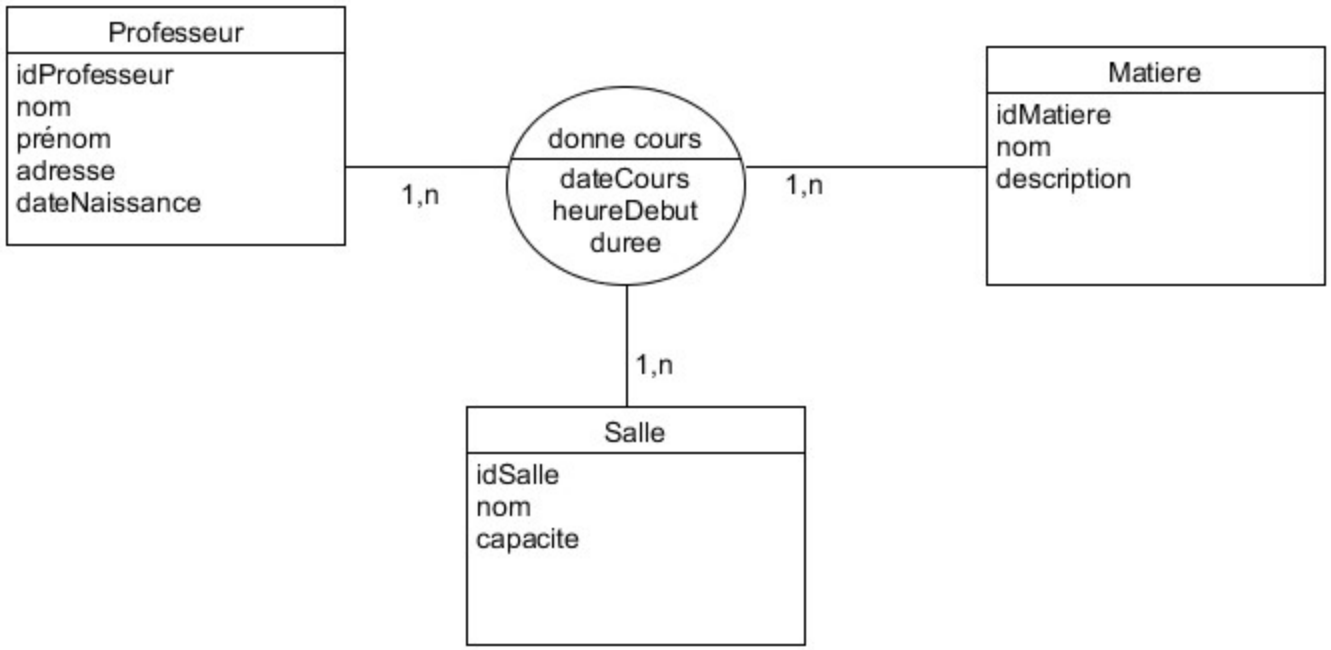
arité 2 : 80%
arité 3 : <20%
arité > 3 : infime…

donneCours (MATIERE [1,n], SALLE [1,n], PROFESSEUR[1,1])

Un **professeur dans une salle** donne cours d'une à plusieurs matières (cardinalité du côté Matière)

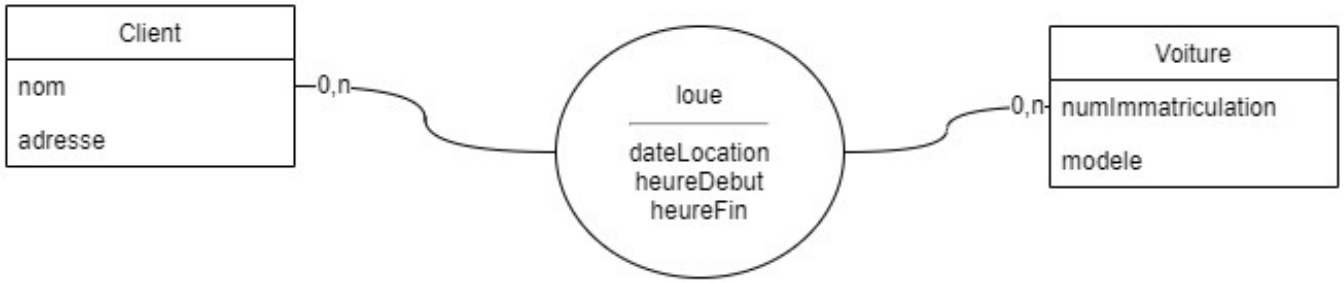
Un **professeur, pour une matière**, donne cours dans plusieurs salles (cardinalité du côté Classe)

Une **matière dans une salle** est donnée par un plusieurs professeurs (cardinalité du côté Professeur)



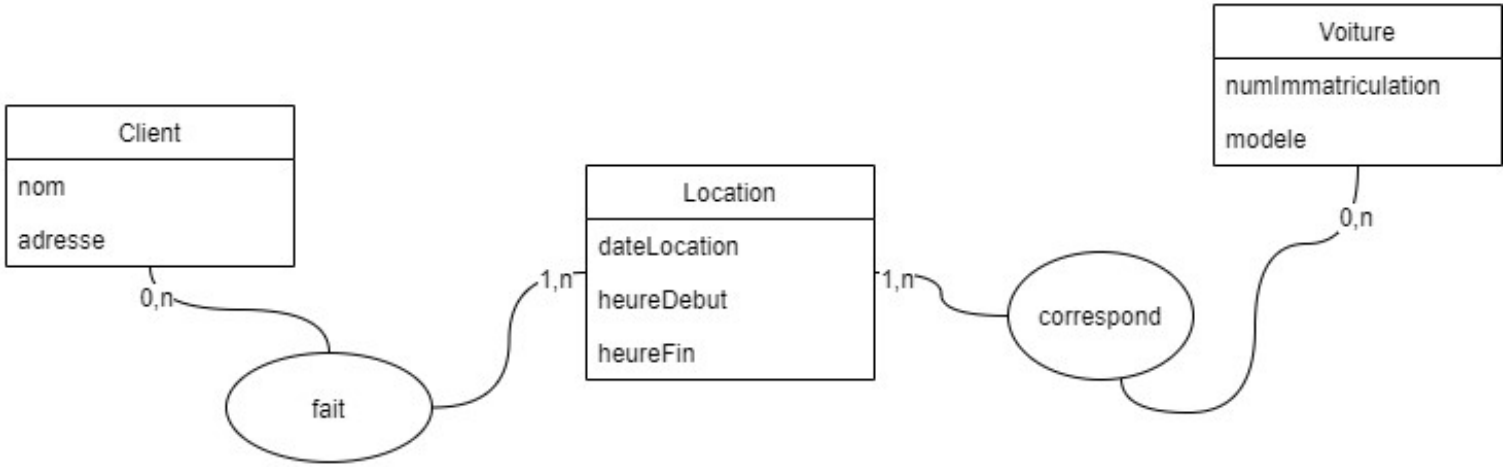
Choix entre Classe d'Association et une nouvelle Classe d'Entité

1) Solution avec classe d'association



Dans cette première solution la location est juste un lien entre Client et Voiture

2) Solution avec classe d'entité



Dans cette solution on les gère les Locations en tant que telles. **Les deux solutions génèrent le même MLD** (avec les cardinalités ci-dessous), mais le deuxième est plus souple (ex: dans le premier schéma est impossible de faire de modifications pour qu'une Location puisse contenir plusieurs Voitures et obtenir un nouveau MLD)

Pour une situation donnée, **il n'existe pas une «solution» unique.**

Le « bon modèle » est celui qui est **accepté par les personnes concernées par le projet.**

3. MLD : concept de BD relationnelle

- Une BD relationnelle est composée de **tables** structurés en **colonnes** (attributs) et **lignes** (enregistrements)

- Les **classes d'entités du MCD** (ex: classe Clients, Comptes) **deviennent tables dans le MLD**

- Dans certains cas, **une classe d'association du MCD se transformera aussi en une table**

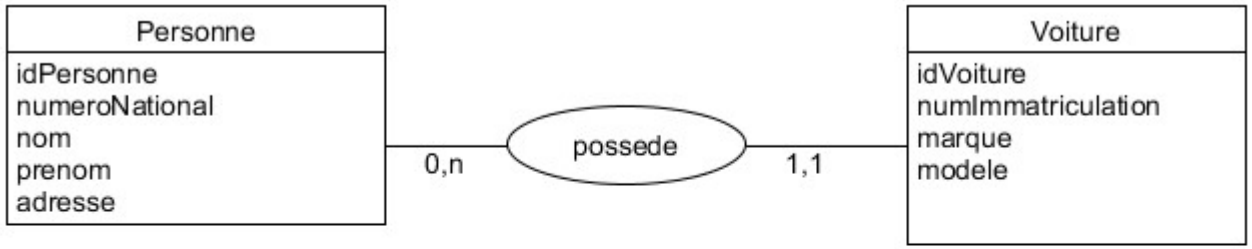
3. MLD : Implémentation

1. **Créer une table pour chaque classe d'entité. Chaque table doit avoir une clé primaire** (rajouter un champ si nécessaire)

2. **Transformer** les classes d'associations selon ses **cardinalités**

Aperçu du passage du MCD au MLD

MCD



MLD

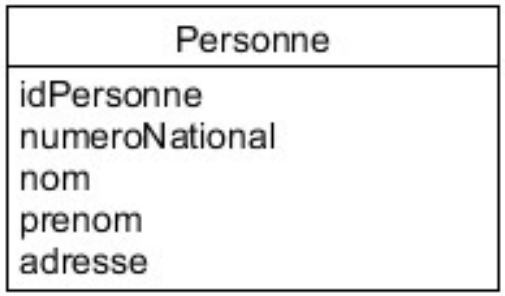
Les Classes d'Entités se transforment en Tables

3.1. Créer une table pour chaque entité

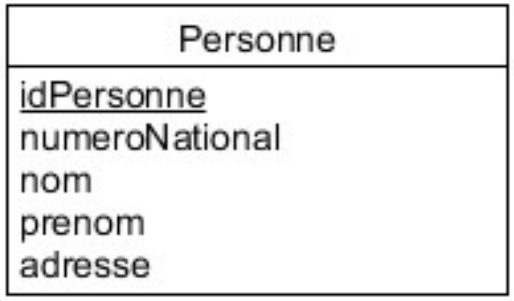
On doit créer un tableau à partir de chaque Classe d'Entité du MCD

Chaque table doit avoir une **clé primaire:** propriété **dont la valeur ne se répète jamais et n'est jamais vide (l'identifiant!)**

MCD



MLD



Ici on aurait pu utiliser comme clé le numéro national, mais par convenance on utilisera toujours un **idNomClasseEntité**

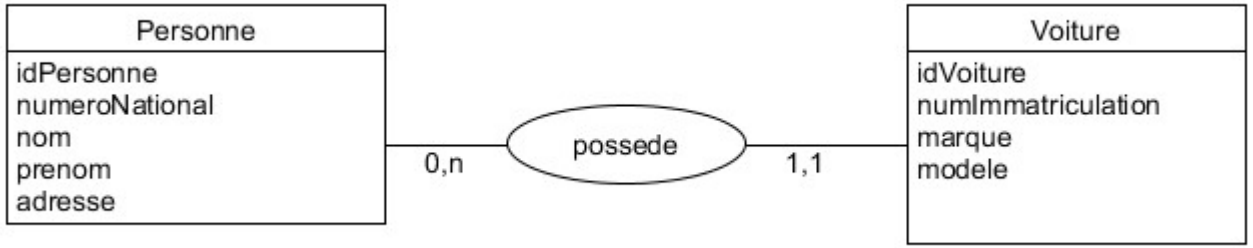
3.2. Transformer les Classes d'Association (I)

Association d'un a plusieurs (1:n ou n:1)

Dans les classes d'associations d'un a plusieurs on aura toujours une cardinalité max **1** d'un côté et **n** de l'autre (peu importe l'ordre, droite ou gauche).

Transformation : La clé primaire du coté max. **n** est rajoutée à la table du coté max. **1**, où elle sera clé étrangère

MCD



MLD

3.2. Transformer les Classes d'Association (II)

Association de plusieurs a plusieurs (n:n)

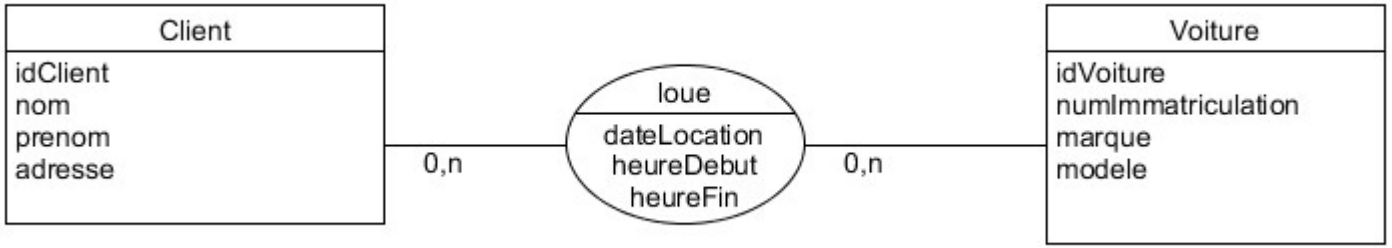
Création d'une nouvelle table dont **la clé primaire est :**

a) **Une nouvelle propriété** (le choix dans ce cours)

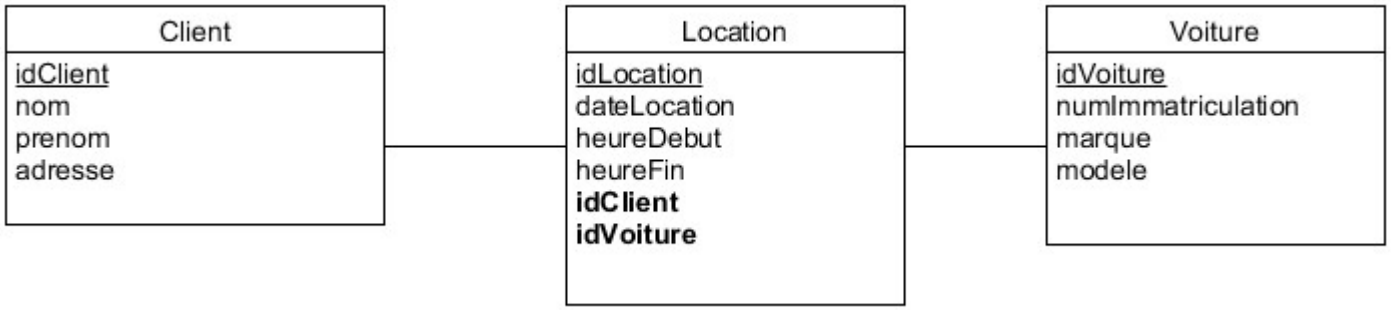
b) **L'ensemble des clés primaires des deux tables**

Les éventuelles propriétés de l'association deviennent les attributs de cette table.

MCD



MLD



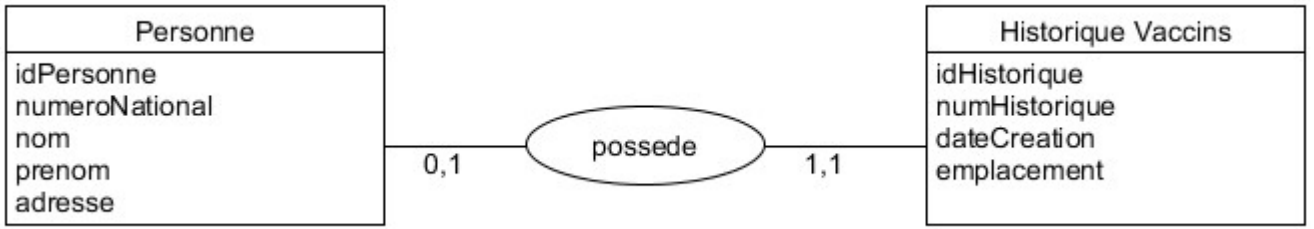
3.2. Transformer les Classes d'Association (III)

Association d'un a un

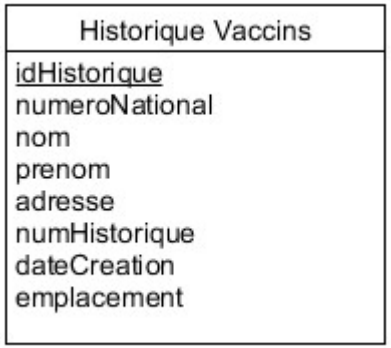
Dans les classes d'associations d'un a un on aura toujours une cardinalité max **1** des deux côtés

Transformation : On fusionne les tableaux et on crée une seule clé primaire.

MCD



MLD

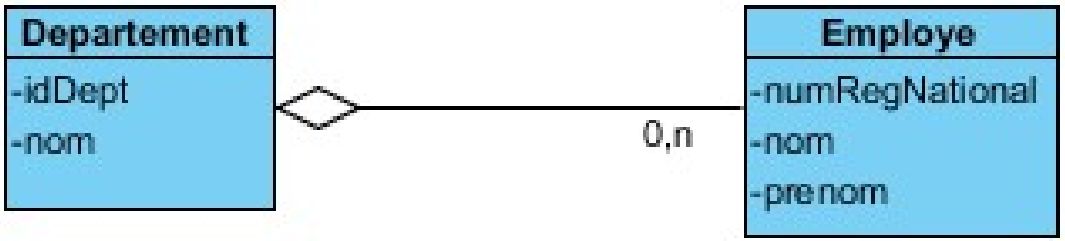


On peut aussi agir de la même manière qu'avec les relations 1-n pour conserver les deux tableaux (une des clés sera rajoutée à l'autre tableau)

Agrégation et composition: passage du MCD au MLD

Ex: Departement et Employes (agrégation)

MCD



MLD

Agrégation: Les Employes continuent à exister si le Departement est eliminé)

Composition: Dans une **composition**, on devrait effacer les employés de la table Employé quand on efface son Département

* Pour définir la composition au niveau du logiciel on peut créer une règle dans le SGBD: pour chaque enregistrement élimine de la table principale (Département), tous les enregistrements associés de la table secondaire (Employé) seront éliminés

Vérification et Normalisation

Contrôler la qualité du modèle vis-à-vis:

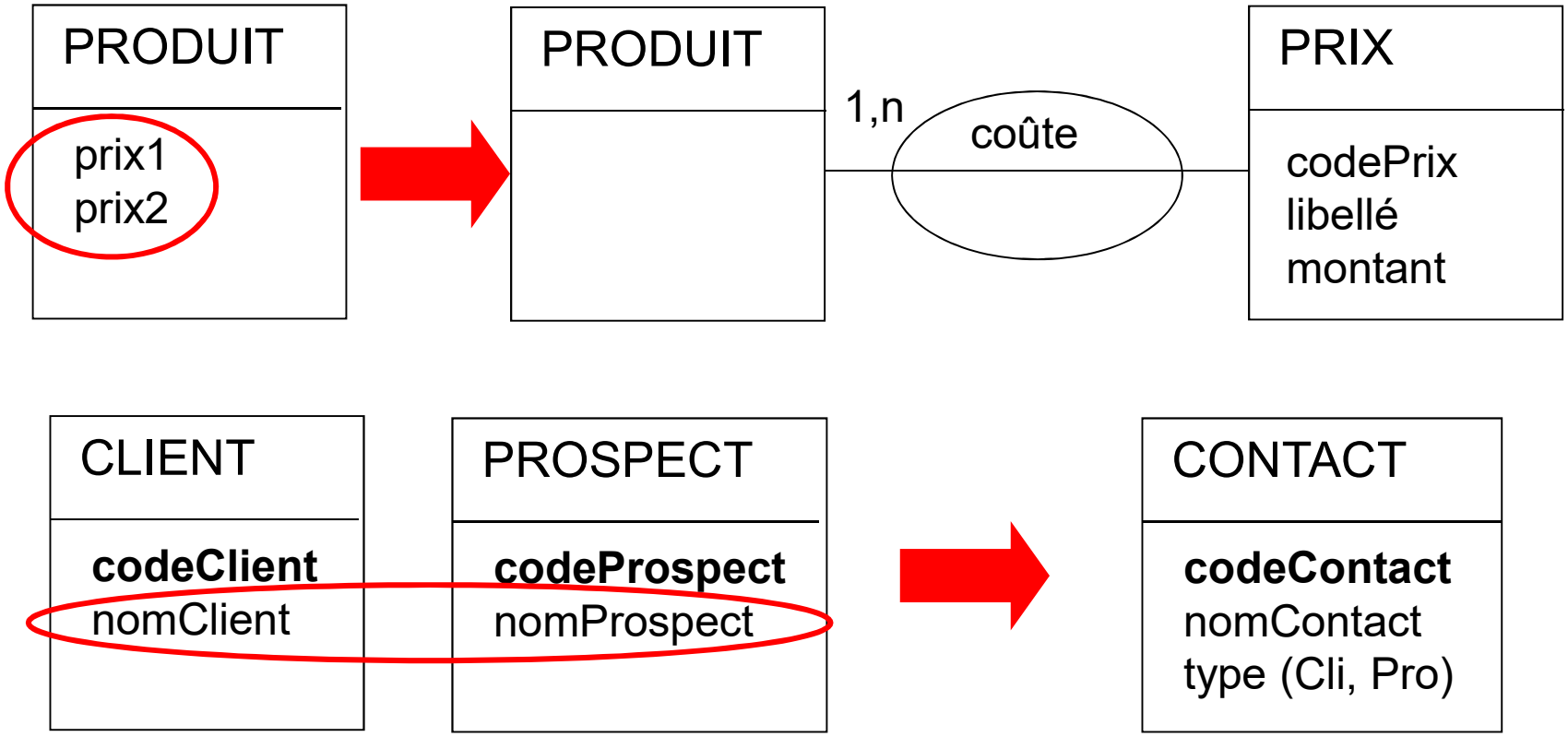
- des **fondements** du modèle d'une part (**règles de vérification**),
- de la **redondance de données** d'autre part (**règles de normalisation**)

Permet de détecter certaines incohérences dans la construction des modèles.

1. Règle Général des Classes Entités

- Toute propriété doit apparaître une seule fois dans un modèle.

  Il faut **éliminer les groupes répétés et la redondance des propriétés** dans la même classe d'entité (avec des noms différents) ou dans des classe d'entité distinctes



Note: cet exemple pourrait être modélisé en utilisant l'héritage au lieu de l'attribut « Type »

2. D'autres rrègles sur les classes d'entités

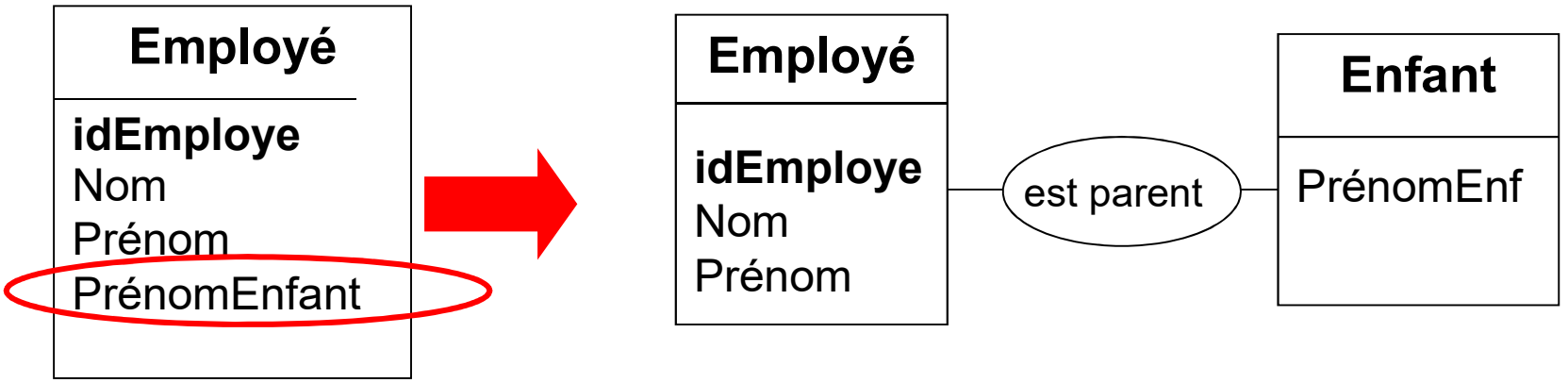
- **Règle de l'identifiant**

  Toutes les entités ont un identifiant (pas encore une clé primaire!)

- **Règle de vérification des entités**

  Pour une occurrence d'une entité, chaque propriété ne prend qu'une seule valeur: notre modèle se trouvera alors dans la **1ere forme normale**

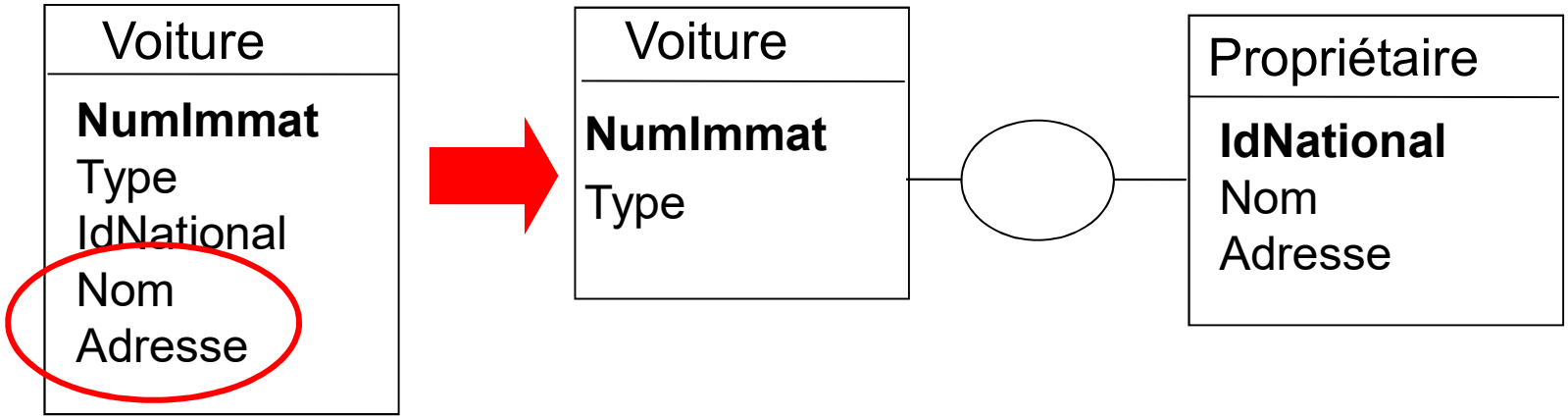
Ex: Un employé « est parent » de plusieurs enfants



On ne mettra jamais plusieurs noms d'enfant séparés par virgules dans le champ PrénomEnf!

- **Règles de normalisation sur les entités**

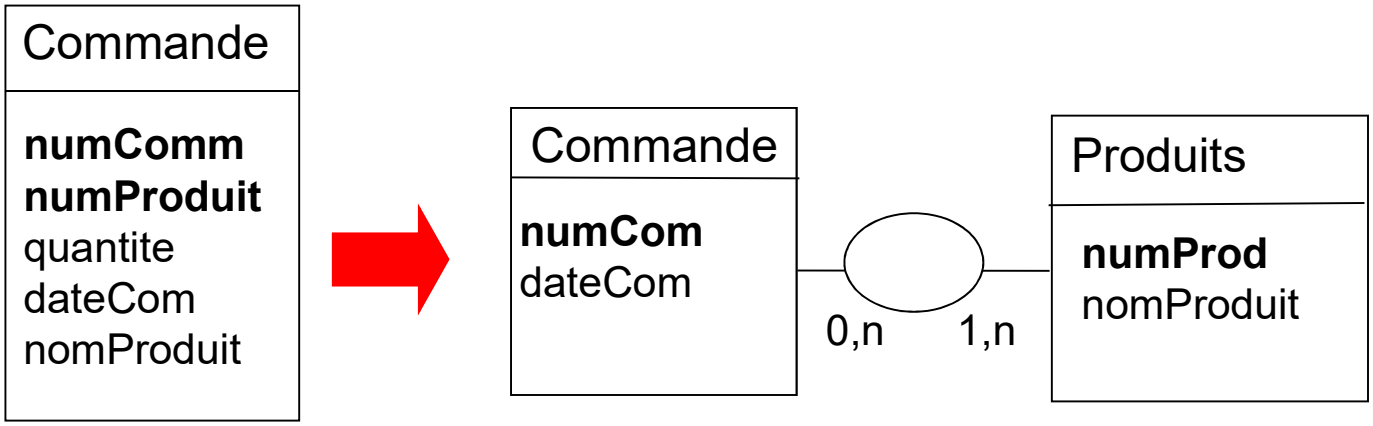
a) Toutes les propriétés de l'entité dépendent fonctionnellement de l'identifiant **et uniquement de l'identifiant: le modèle se trouve dans la 2ème forme normale**



Ici, Nom et Adresse dépendent de *IdNational* et pas de l'identifiant

Solution: on décompose l'entité en deux autres entités qui respectent la règle

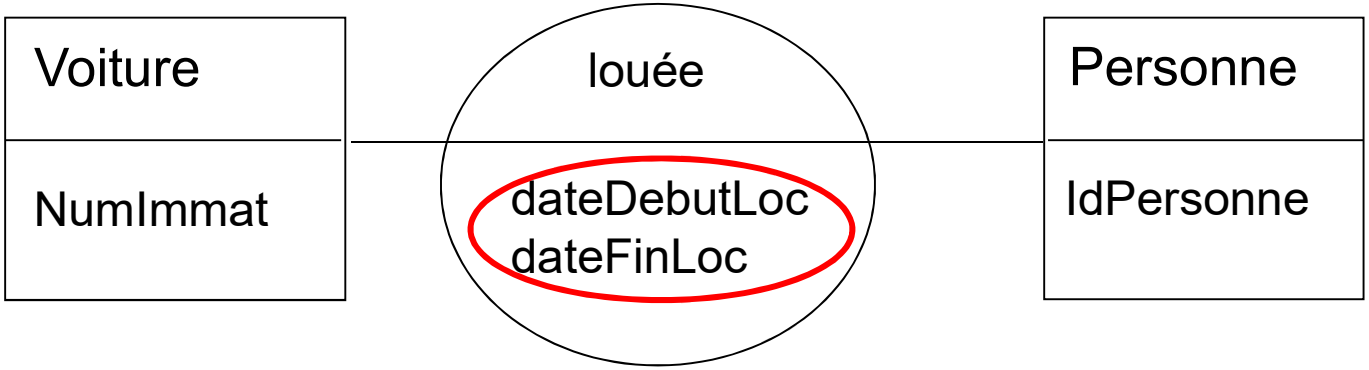
b) Les propriétés de l'entité ne peuvent pas être déterminées par une partie de l'identifiant: le modèle se trouve dans la **3ème forme normale**



numCom → dateCom contredit la règle car ***numComm*** est uniquement une partie de l'identifiant (pas l'identifiant dans sa totalité) et il détermine par lui méme les proprietés dateCom

Solution: On décompose l'entité Commande en deux entités. Ces règles correspondent aux 2FN et 3FN du modèle relationnel (dépendance pleine et directe des clés).

3. Règles sur les associations



- **Règle de vérification des associations**

Pour chaque lien d'association, chaque propriété ne prend qu'une seule valeur (ex: pas plusieurs dates **dans l'association**).

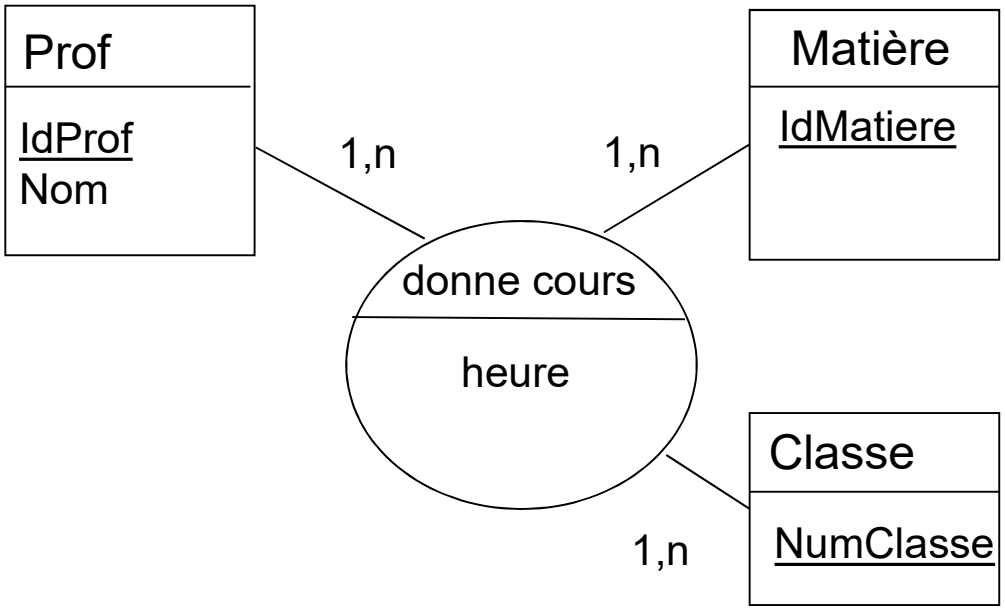
- **Règle de normalisation sur les propriétés des associations**

Toutes les propriétés de l'association doivent dépendre des identifiants des entités qui se trouvent dans les extrêmes de l'association, et uniquement d'eux.

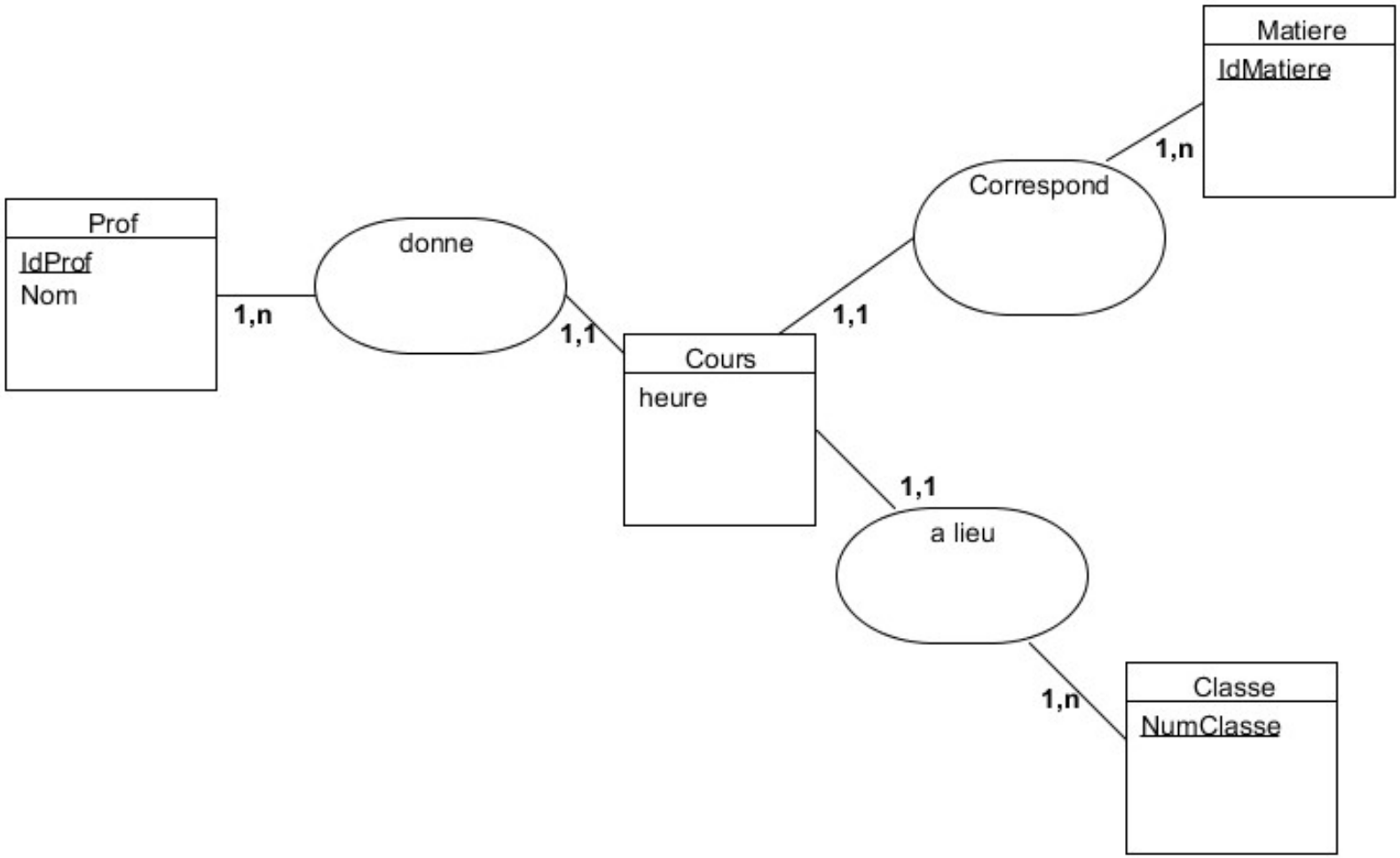
Ex: Si on remarque que dans l'association "autorisée" la datePermis dépend uniquement de l'entité Personne et pas de Voiture on déplacera cet attribut vers Personne et on l'enlèvera de l'association.

La décomposition des associations n-aires

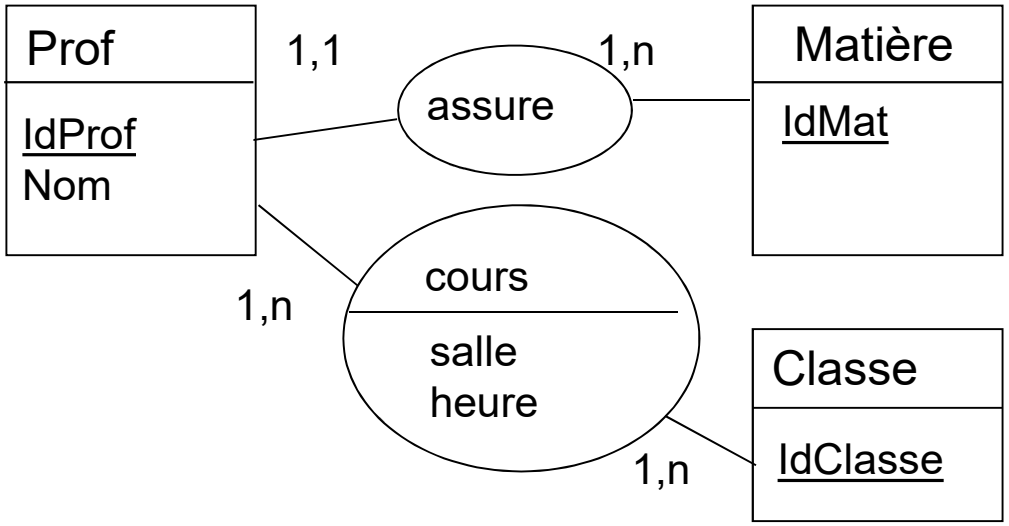
Il faut **réduire au minimum le nombre d'associations d'arité > 2**. Si on observe **une dépendance entre deux identifiants, on peut décomposer** l'association n-aire



Nous pouvons aussi **transformer les associations ternaires en binaires**. Pour ce faire on a besoin de créer une éntité artificiel (ex: Cours) qui établira le lien entre les trois entités (voir notes sur la correspondande dans le MLD pour mieux comprendre)

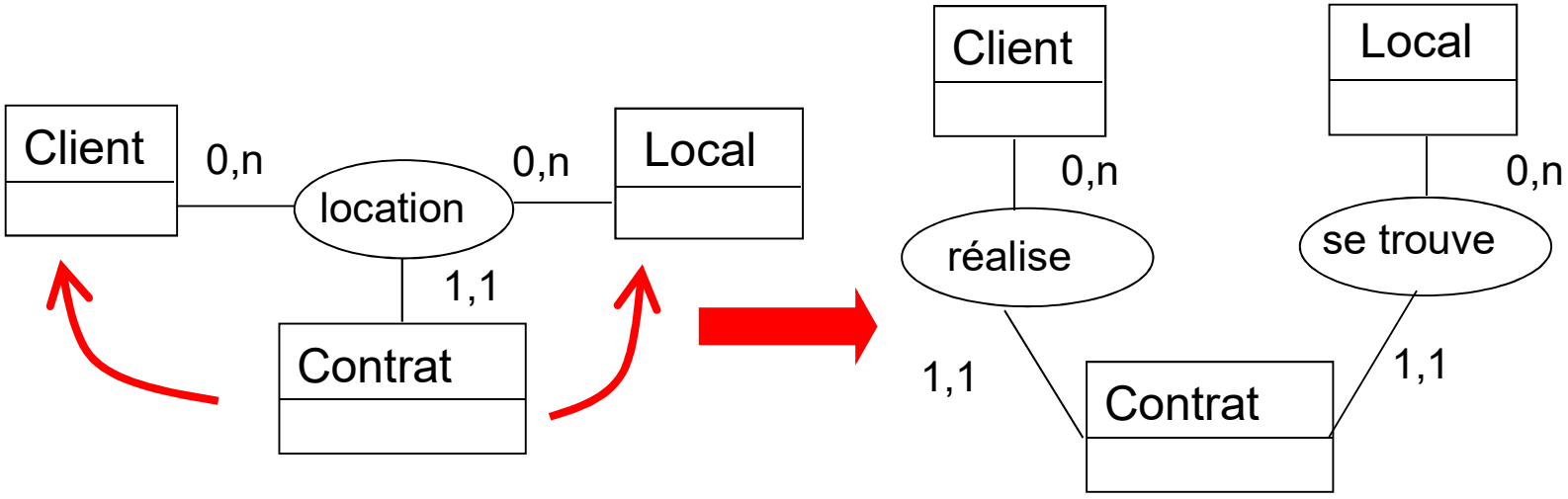


Une éventuelle dépendance *IdProf* → *IdMat* (ex: si un prof donne cours uniquement d'une matière) conduit à la décomposition suivante:



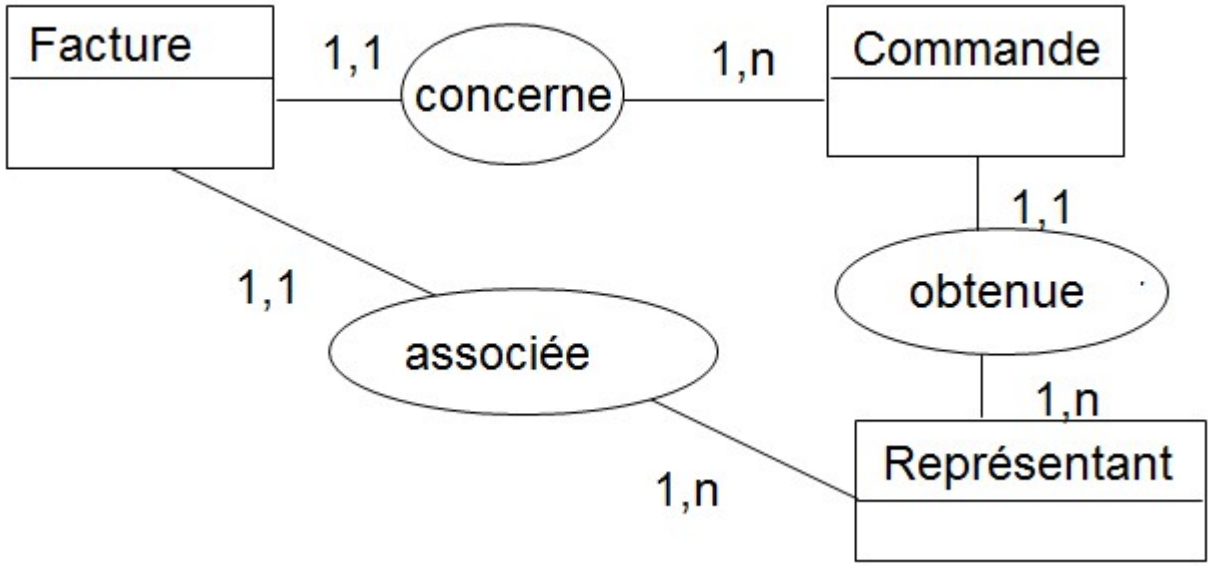
C'est plus facile quand il une patte a une cardinalité 1,1.

Exemple: un client et un local sont associés par un seul contrat



- **La suppression des associations transitives**

Toute association pouvant être obtenue par transitivité de n autres associations peut être supprimée. La transitivité s'évalue en fonction de la **signification** des associations.



On supprime l'association *associée*, car elle peut être obtenue par transitivité sur les associations *concerne* et *obtenue*

c) Quelques contraintes d'intégrité importantes

Des propriétés qui doivent être vérifiées par les données

- **Contraintes intégrées au modèle E/A**

1) Contrainte d'identifiant

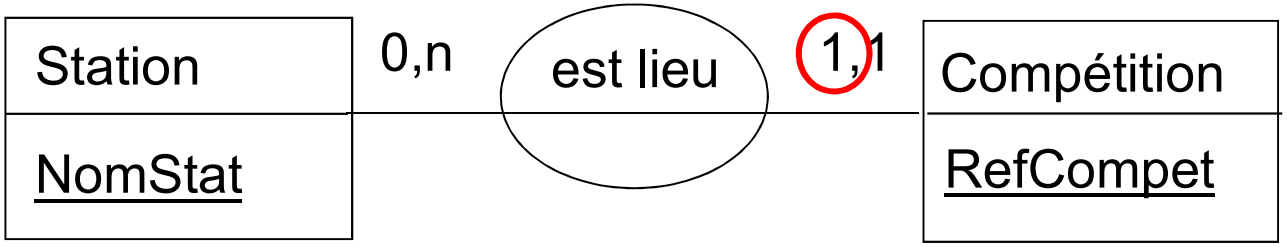
Les valeurs prises par l'identifiant sont uniques (dans le temps) et toujours définies.

Ex : identifiant de l'entité PERSONNE

- nom + prénom pas suffisant
- n° téléphone pas stable dans le temps

2) Contraintes de cardinalité

Les cardinalités portées par les entités membres d'association imposent des nombres minimales et maximales d'occurrence dans l'association.



Une compétition n'existe pas sans la station!

Ça implique que:

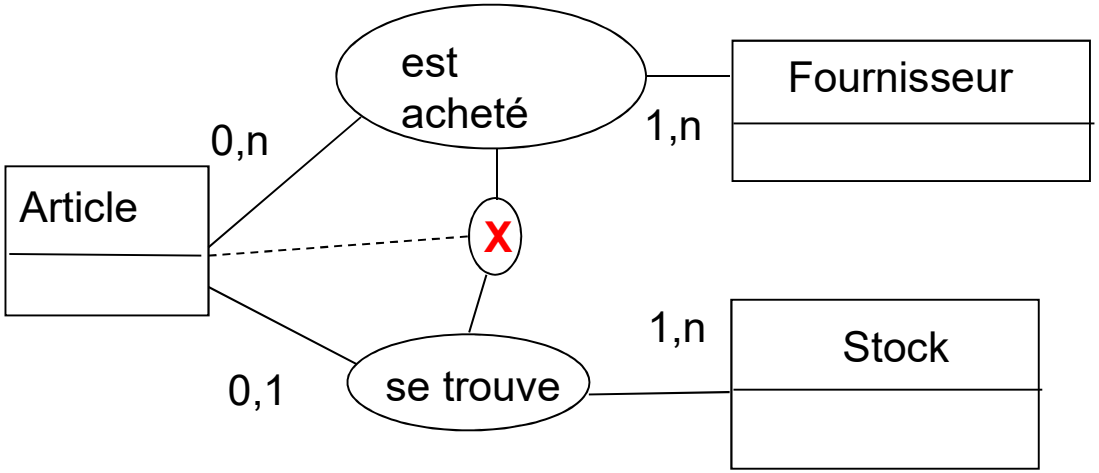
Une cardinalité mini de 1 rend l'existence d'une occurrence d'entité dépendante de l'existence d'une occurrence d'une autre entité

Une station peut exister de manière indépendante de toute compétition.

- **Contraintes extensions du modèle E/A**

1) Exclusivité de participation d'une entité à plusieurs associations

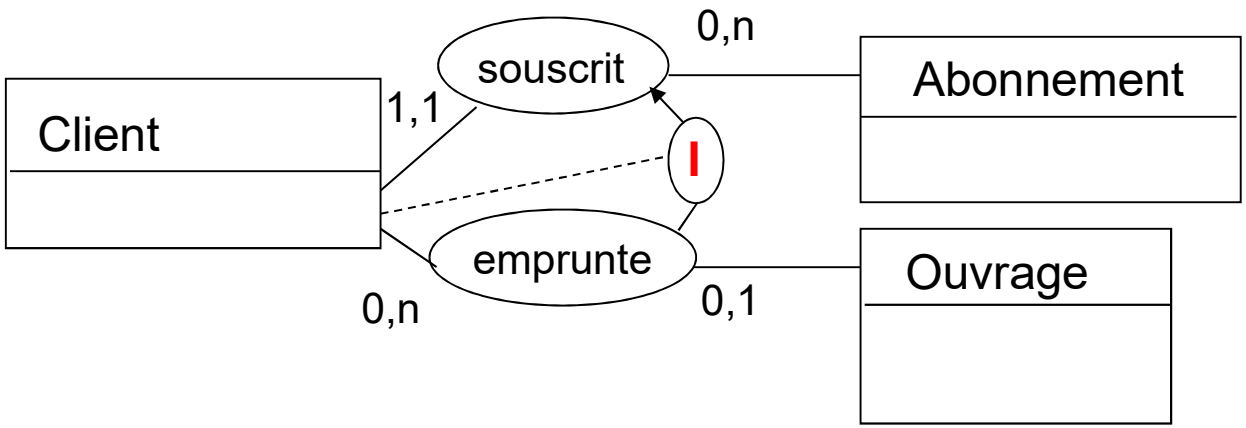
Si l'entité E participe à l'association A1, elle ne peut participer à l'association A2.



Un Article concret est soit acheté auprès d'un fournisseur, soit figure dans le Stock

2) Inclusion de participation d'une entité à plusieurs associations

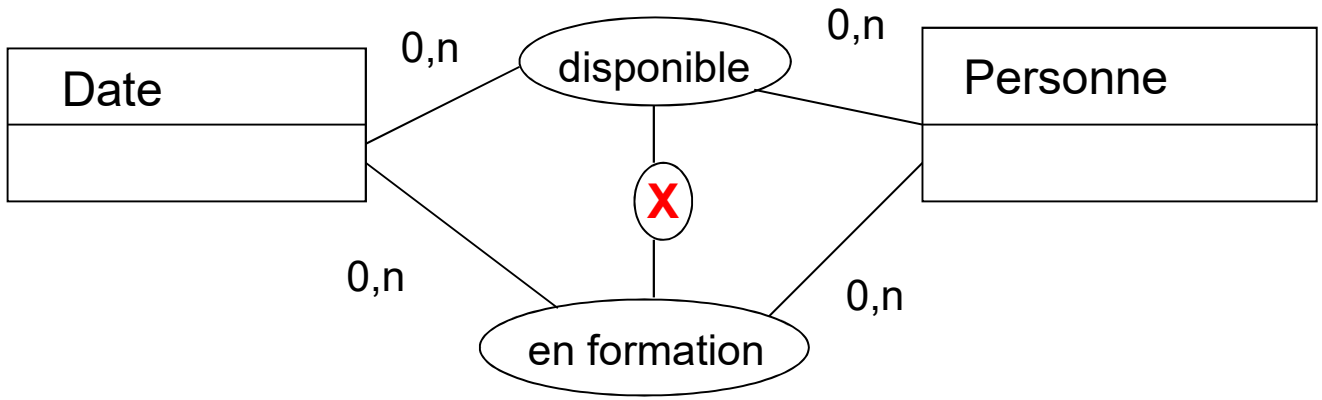
La participation d'une entité E à une association A1 implique sa participation à l'association A2.



La participation de *client* dans l'association *emprunte* implique sa participation à l'association *souscrit*.

3) Exclusion de participation entre associations

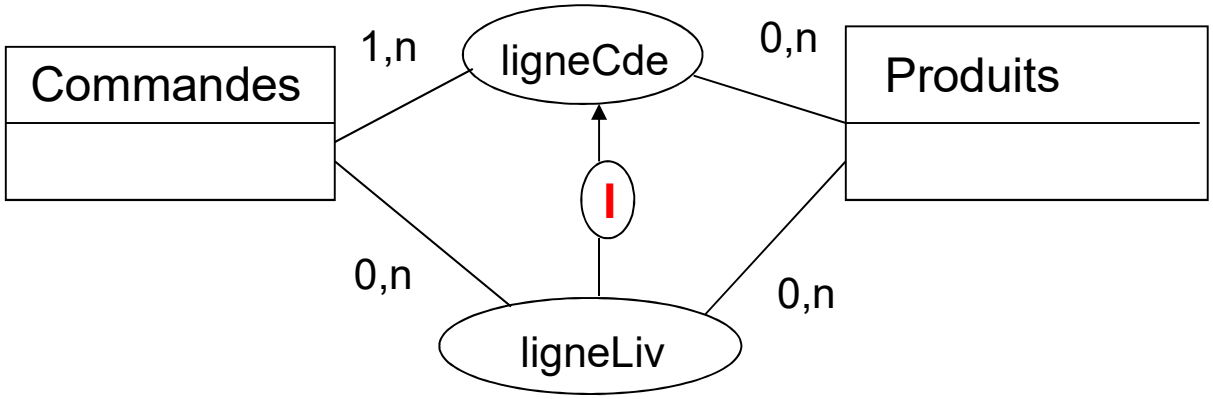
Il y a exclusion de participation entre associations si la participation des entités à l'association A1 exclut leur participation à l'association A2.



Une personne à une même date ne peut pas figurer simultanément dans les deux associations: *disponible* et *en formation*.

4) Inclusion de participation entre associations

Il y a inclusion de participation entre associations **si la participation des entités à l'association A1 implique leur participation à l'association A2**.



Tout couple Commande-Produits figurant dans l'association *ligneLiv* doit figurer dans l'association *ligneCde*

Le problème va être de vérifier toutes ces contraintes dans les programmes qui mettent à jour les données!