Plan du cours SGBD3

- Période A:
  - Les Extensions MERISE (MCD)
  - Les Règles de transformation DC UML -> BD
  - Les contraintes OCL
- Période B
  - Les Triggers

Modalités d'évaluation du cours SGBD3

- Période A:
  - DST de 3 heures ; coefficient 2
- Période B
  - TP noté; coefficient 1

LES EXTENSIONS MERISE: *Plan*

- 1) EXTENSIONS DU MODELE CONCEPTUEL DE DONNEES
  - Généralisation et Spécialisation
  - Les contraintes d'intégrité fonctionnelles (CIF)
  - L'identification relative
  - Les Contraintes ensemblistes sur associations
  - Les Contraintes de stabilité
  - Contraintes sur l'heritage
  - Exemple récapitulatif
  - L'historisation
- 2) PASSAGE AU MODELE LOGIQUE OU A L'ALGEBRE RELATIONNELLE
- 3) CONCLUSION

LES EXTENSIONS MERISE
Généralisation et Spécialisation

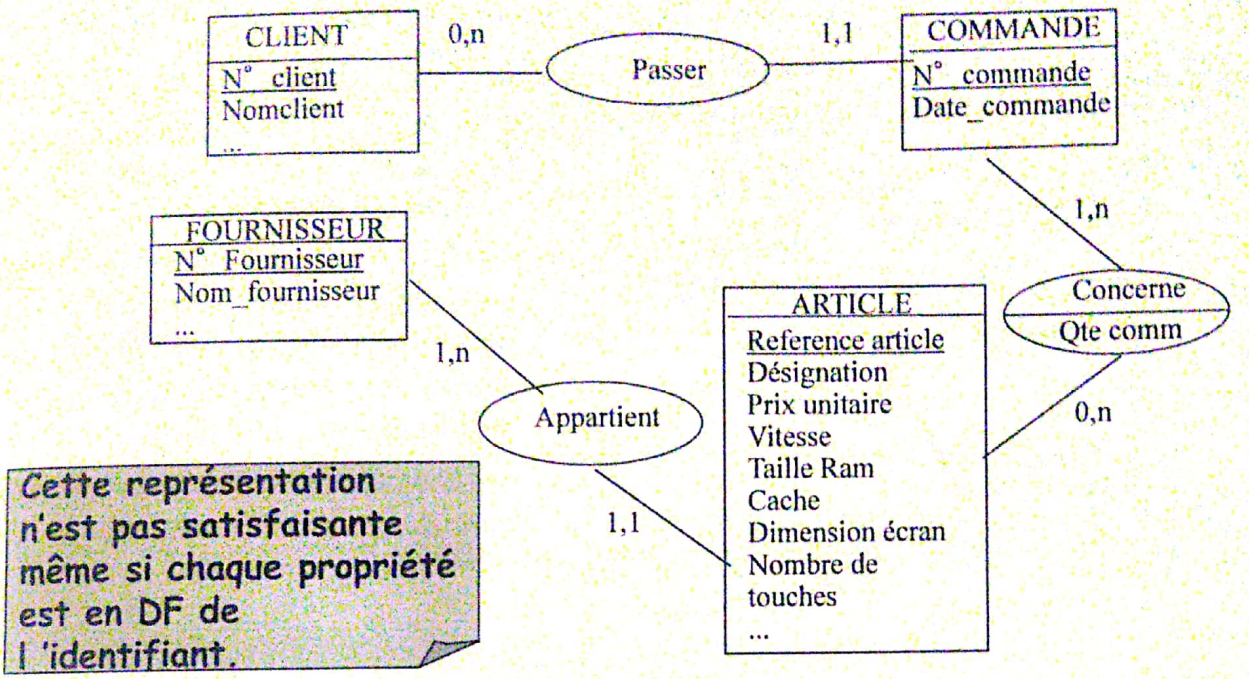
Exemple (1)

- Une entreprise vend du matériel informatique. Les articles vendus peuvent être des UC, des périphériques ou des combinaisons de plusieurs articles.
- Certaines **propriétés** définissant un article sont communes aux UC et aux périphériques : *la référence, le prix unitaire*, etc.
- Chaque type d'articles possède des caractéristiques propres. *La vitesse du processeur* pour l'UC, *le nombre de touches* pour le clavier, etc.

EXTENSIONS: *Généralisation et Spécialisation*

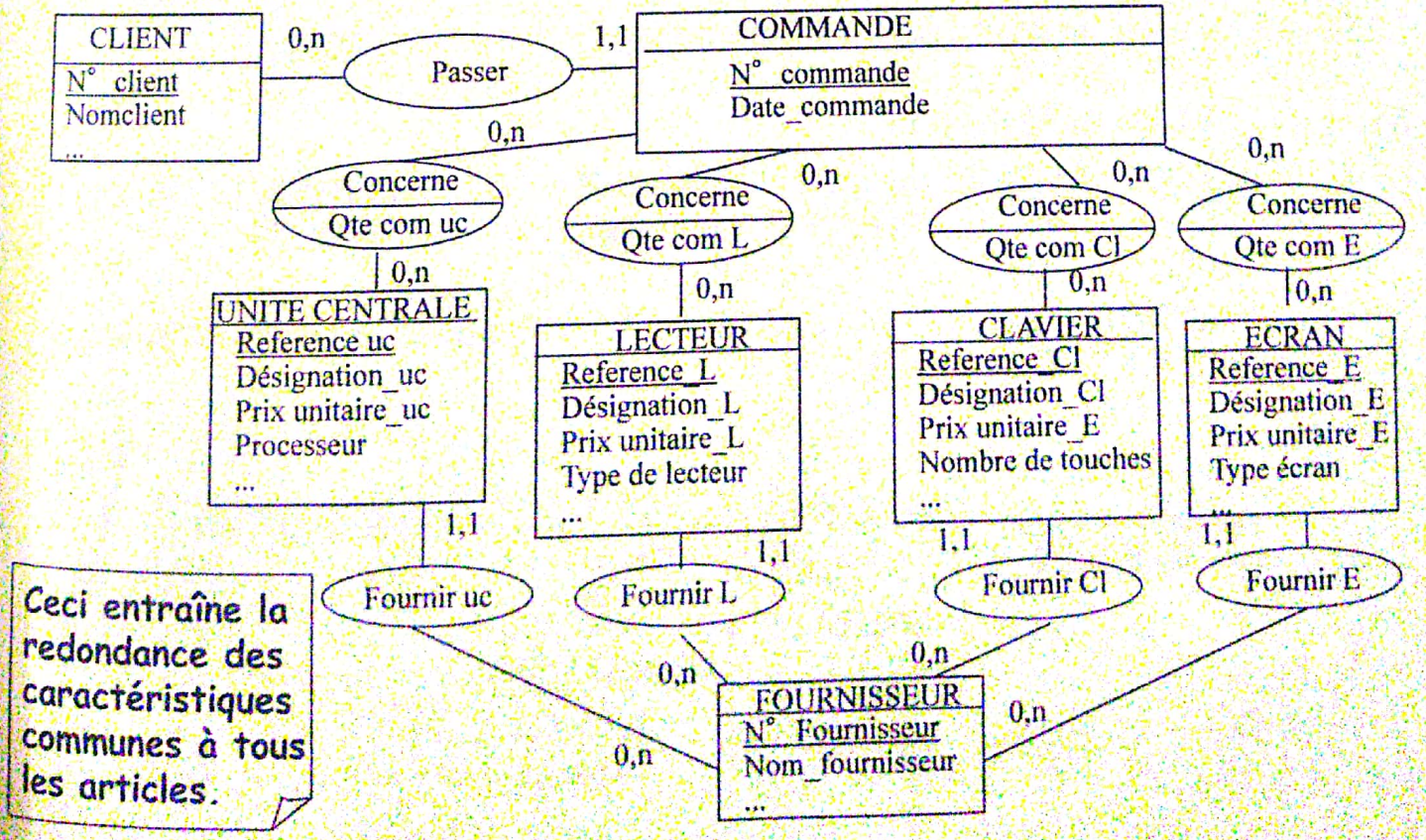
Exemple (2)

- Voici une représentation possible:

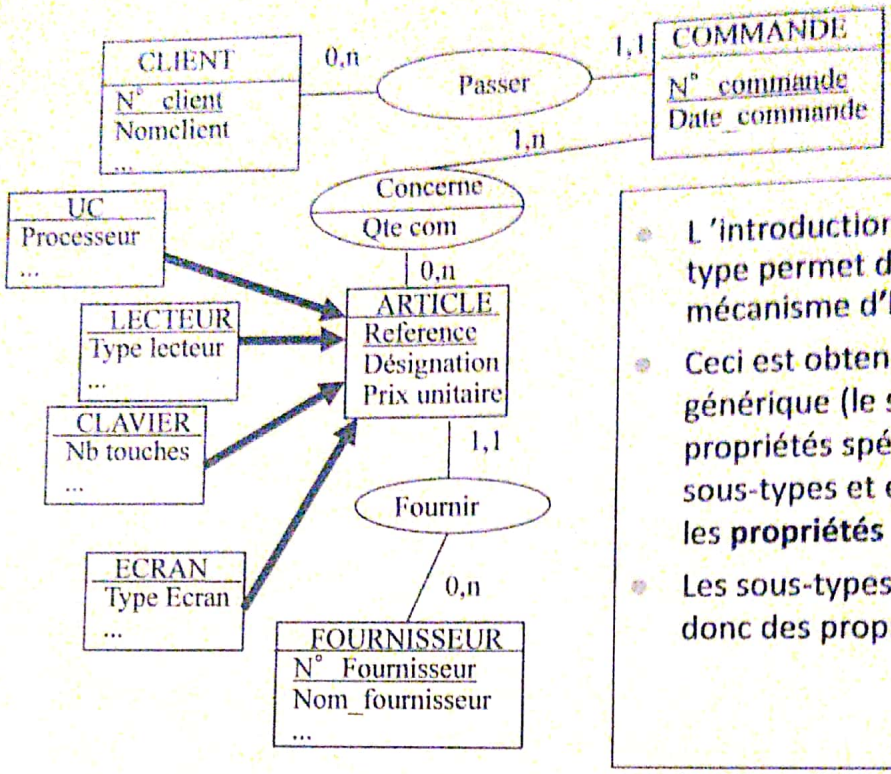


EXTENSIONS: *Généralisation et Spécialisation*

Exemple (3): **Spécialisons chaque groupe de propriétés:**



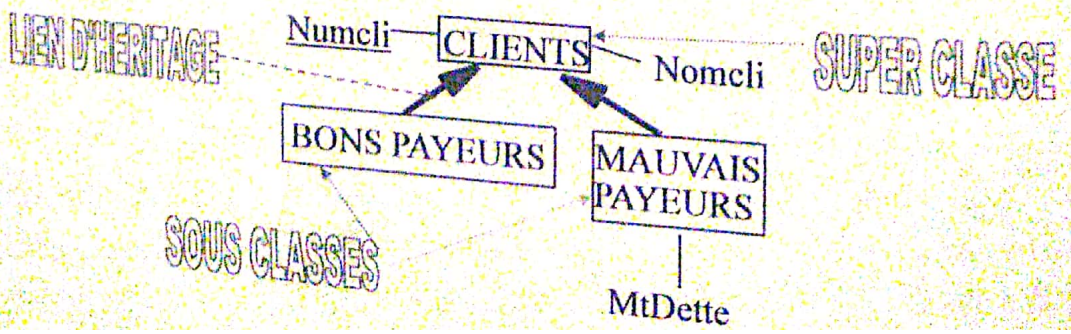
EXTENSIONS: *Généralisation et Spécialisation*



- L 'introduction de la notion de sous-type permet de tenir compte du mécanisme d'**héritage**.
- Ceci est obtenu est sortant de l'objet générique (le super-type) les propriétés spécifiques à chacun des sous-types et en ne conservant que les **propriétés communes**.
- Les sous-types héritent du super-type, donc des propriétés communes.

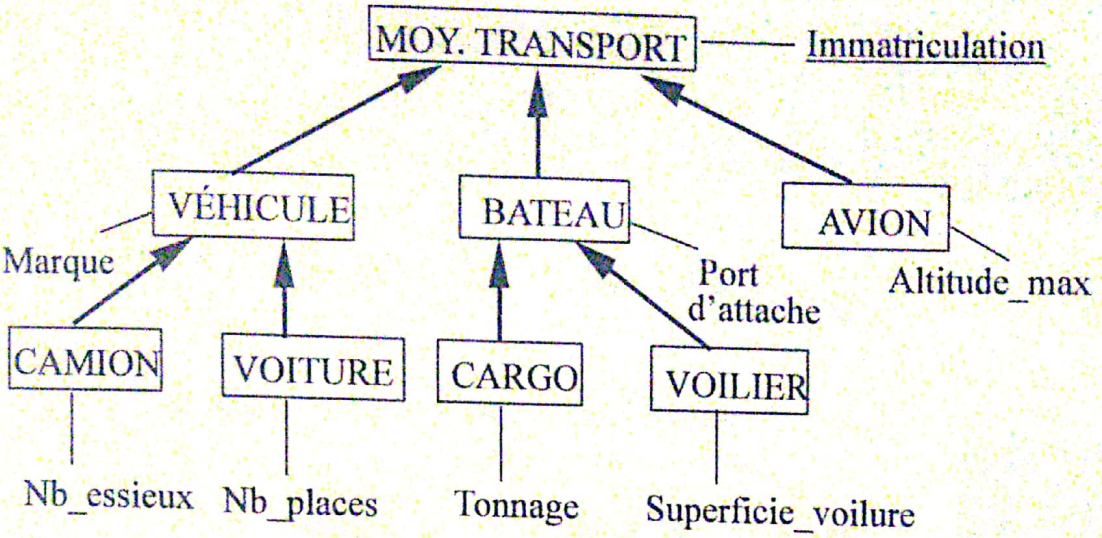
EXTENSIONS: *Généralisation et Spécialisation*

- **SOUS TYPE D'ENTITES**
  - LORSQU'UN SOUS ENSEMBLE D'ENTITES POSSEDE DES INFORMATIONS SPECIFIQUES, IL EST POSSIBLE DE CRÉER UNE SOUS CLASSE D'ENTITES;
  - TOUTES LES SOUS CLASSES **HERITENT DES ATTTRIBUTS** DE LA CLASSE MERE, ELLES PEUVENT AVOIR DES ATTRIBUTS SPECIFIQUES;
  - **L'IDENTIFIANT** DE LA CLASSE MERE EST AUSSI UN IDENTIFIANT DE SES SOUS-CLASSES.



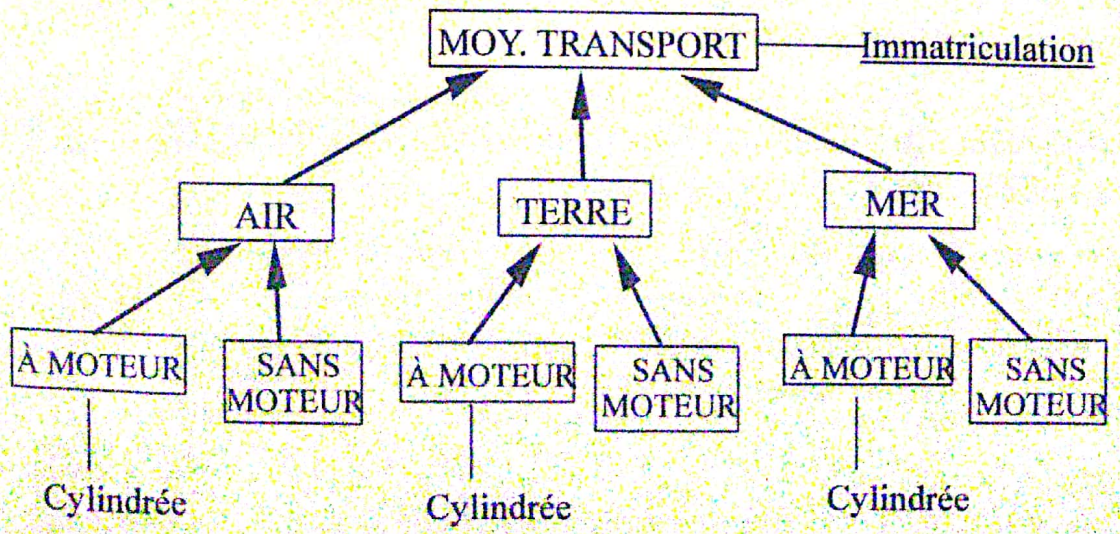
EXTENSIONS: *Généralisation et Spécialisation*

HIERARCHIE D'HERITAGE: EXEMPLE



EXTENSIONS: *Généralisation et Spécialisation*

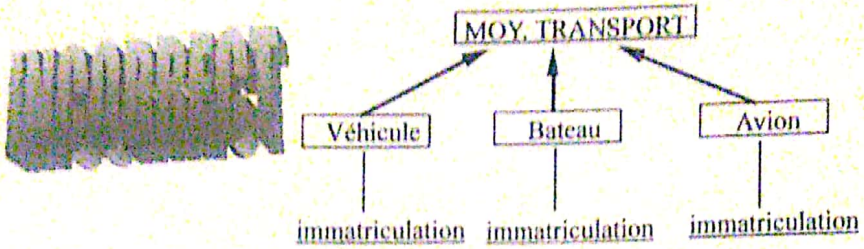
REDONDANCE DANS UNE HIERARCHIE D'HERITAGE

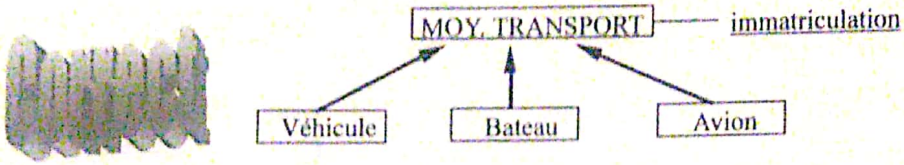


EXTENSIONS: *Généralisation et Spécialisation*

- PROBLEMES LIES A LA CONCEPTION

  PLACEMENT D'UN ATTRIBUT DANS UNE HIERARCHIE DE SOUS CLASSES D'ENTITES



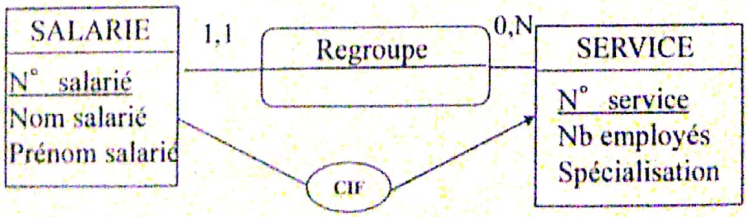


EXTENSIONS: LA CIF

- **CIF: CONTRAINTE D'INTEGRITE FONCTIONNELLE**

  ELLE EST DEFINIE SUR UNE ASSOCIATION ET REPRESENTE LE FAIT QUE L'UNE DES ENTITES DE SA COLLECTION EST IDENTIFIEE SANS AUCUN DOUTE PAR LA CONNAISSANCE D'UNE OU PLUSIEURS AUTRES.

  UNE ASSOCIATION BINAIRE AYANT DES CARDINALITES (0,1) OU (1,1) EST UNE CIF.

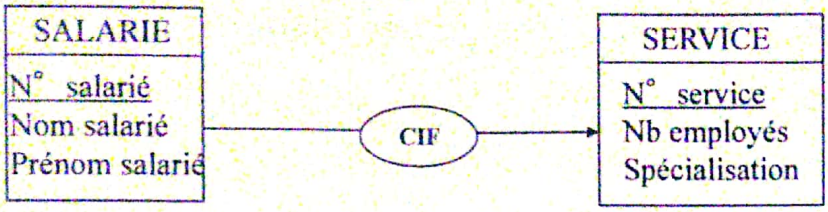


DE LA CONNAISSANCE DU "N° SALARIE", ON PEUT DEDUIRE LE "N° DE SERVICE" AUQUEL IL APPARTIENT

EXTENSIONS: LA CIF

- **CIF: CONTRAINTE D'INTEGRITE FONCTIONNELLE**

  SI L'ASSOCIATION EST VIDE (i.e. N'A PAS DE PROPRIETES) ET QU'IL N'EXISTE PAS D'AUTRE ASSOCIATION ENTRE LES DEUX ENTITES, ON PEUT REMPLACER L'ASSOCIATION PAR UNE CIF.

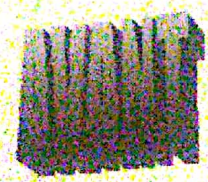


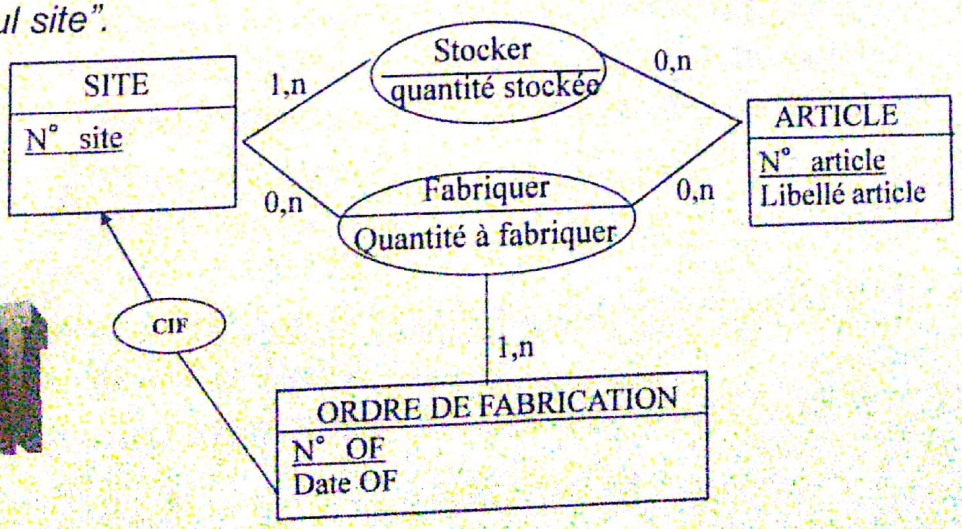
EXTENSIONS: LA CIF

- **CIF: CONTRAINTE D'INTEGRITE FONCTIONNELLE**

  LES CIF PERMETTENT DE SIMPLIFIER LES ASSOCIATIONS DE DIMENSION SUPERIEURE A 2.

Exemple: L'entreprise industrielle où: "Un ordre de fabrication ne concerne qu'un seul site".

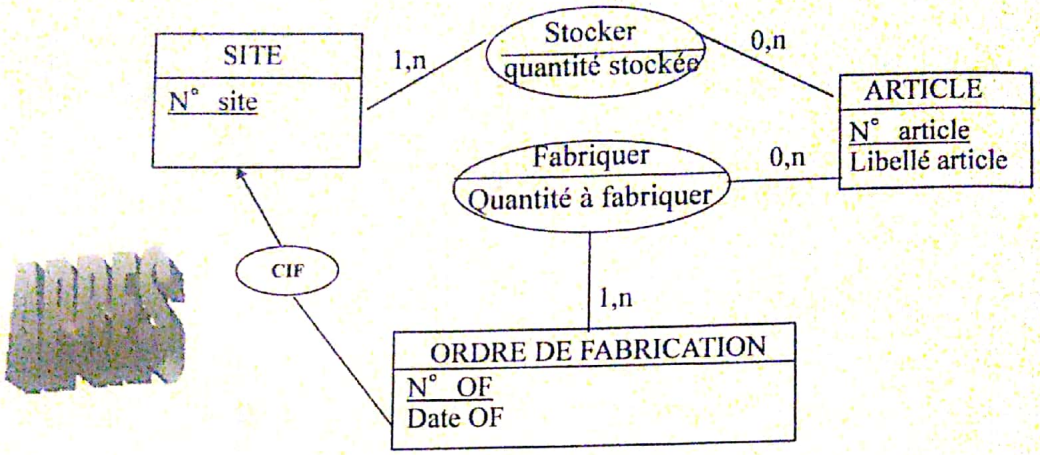




EXTENSIONS: LA CIF

- **CIF: CONTRAINTE D'INTEGRITE FONCTIONNELLE**

Exemple (suite): L'entreprise industrielle où: "Un ordre de fabrication ne concerne qu'un seul site".

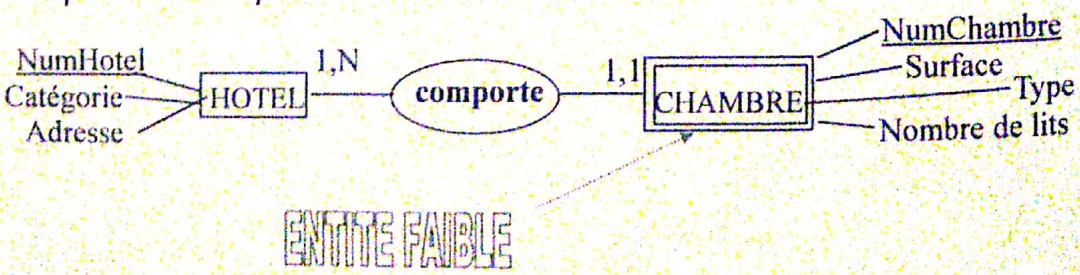


EXTENSIONS: *IDENTIFICATION RELATIVE*

- **CLASSES D'ENTITES FAIBLES**

UNE CLASSE D'ENTITES FAIBLES EST UN ENSEMBLE D'ENTITES FAIBLES DE MEME TYPE DEFINI PAR RAPPORT A DES ENTITES DE MEME TYPE.

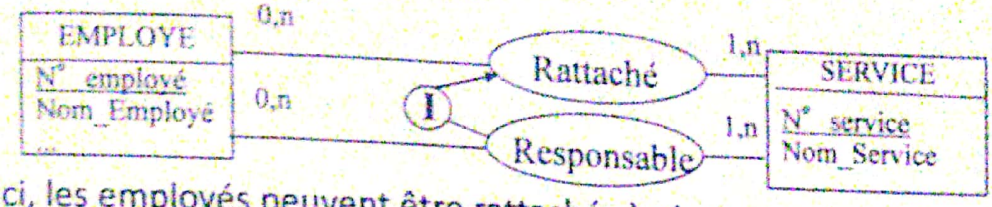
Exemple: Dans un hôtel, une chambre est identifiée de manière unique par son numéro. Si l'on gère plusieurs hôtels, le n° de chambre ne suffit plus pour identifier de manière unique une chambre parmi toutes les chambres de tous les hôtels. La classe d'entité ***chambre*** *est alors représentée par une classe d'entités faibles.*



EXTENSIONS: *LES CONTRAINTES ENSEMBLISTES*

La contrainte d'inclusion

- ***La contrainte d'inclusion exprime que l'ensemble des occurrences d'une association est comprise dans l'ensemble des occurrences d'une autre.***

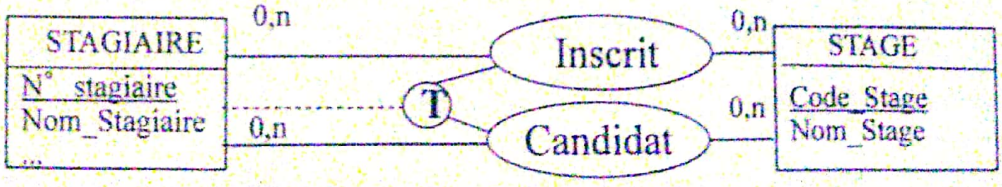


- Ici, les employés peuvent être rattachés à plusieurs services.
- Les services peuvent être mis sous la responsabilité d'un ou plusieurs employés.
- **Un employé ne peut être responsable que de services auxquels il appartient.**
- Tout couple (*employé, service*) participant à *Responsable* doit figurer parmi les couples (*employé, service*) de *Rattaché*.

EXTENSIONS: *LES CONTRAINTES ENSEMBLISTES*

La contrainte de totalité

- ***La totalité précise que toutes les occurrences d'une entité impliquée dans deux associations ou plus sont présentes dans au moins l'une d'entre elles.***

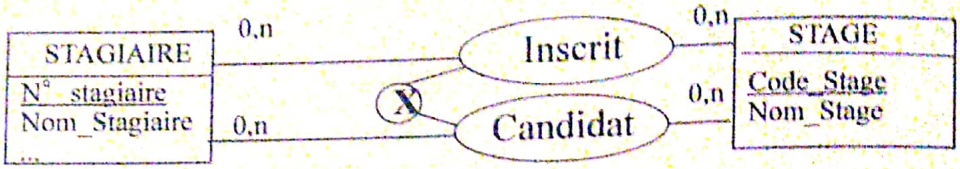


- Ici, l'entité STAGIAIRE est choisie comme pivot de la contrainte.
- Cette contrainte s'appelle aussi **OU Inclusif** ou **Couverture.**
- **La contrainte impose que STAGIAIRE participe au moins une fois à l'une des deux associations.**
- Le contrôle de couverture s'effectue par rapport au pivot, ici STAGIAIRE et non STAGE.

EXTENSIONS: *LES CONTRAINTES ENSEMBLISTES*

La contrainte d'exclusion (1)

- ***Elle interdit qu'une occurrence d'une entité impliquée dans deux associations ou plus soit présente dans 2 d'entre elles.***

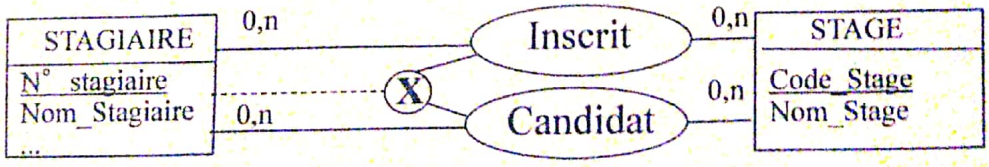


- Ici, un STAGIAIRE ne peut être à la fois candidat et inscrit dans une même formation.
- Le pivot implicite est le couple (*stagiaire, stage*), la vérification d 'exclusion se fait donc sur les couples (*stagiaire, stage*) dont une occurrence ne peut se trouver à la fois dans *candidat* et dans *inscrit*.

EXTENSIONS: *LES CONTRAINTES ENSEMBLISTES*

La contrainte d'exclusion (2)

- ***C'est une contrainte symétrique, comme la contrainte de totalité.***
- ***Il est possible de préciser son pivot.***



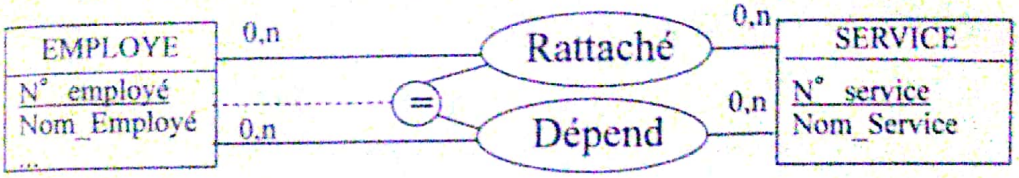
- Ici, un STAGIAIRE ne peut être à la fois candidat et inscrit quelque soit le stage concerné.
- **Règle de gestion**: un stagiaire est soit candidat à l'un quelconque des choix, soit inscrit, mais pas les deux à la fois.

- *Remarque: Cette contrainte peut aussi être notée par* ***E*** *(Exclusion)*

EXTENSIONS: *LES CONTRAINTES ENSEMBLISTES*

La contrainte d'égalité

- ***C'est la combinaison de deux inclusions symétriques, l'ensemble des valeurs du pivot participant à une des associations contraintes devant être inclus dans l'ensemble des valeurs participant à l'autre et réciproquement.***



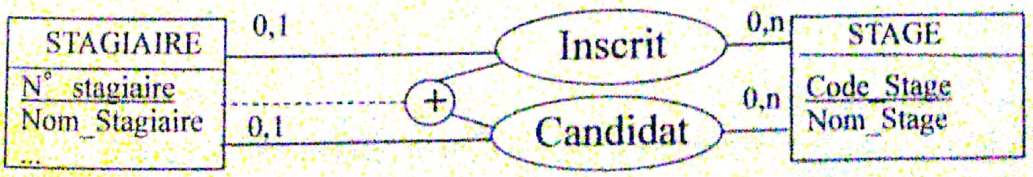
- Ici, l'ensemble des occurrence de *EMPLOYE* (le pivot) participant à l'association *Rattaché* doivent participer à l'association *Dépend* et réciproquement.

- *Remarque: Cette contrainte peut aussi être notée par **S** (Simultanéité)*

EXTENSIONS: *LES CONTRAINTES ENSEMBLISTES*

La contrainte de OU exclusif

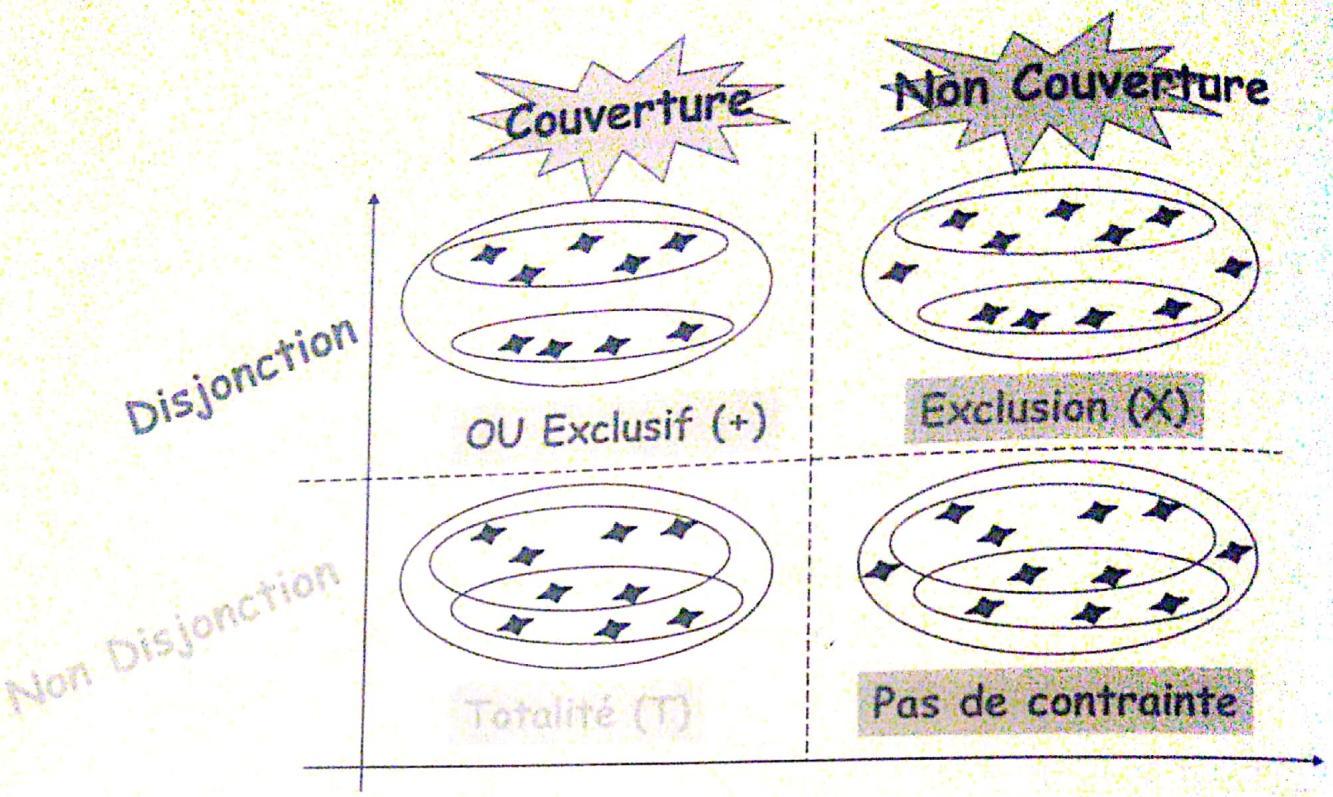
- ***Elle combine une totalité et une exclusion et revient donc à vérifier que la jointure entre les relations impliquées soit vide.***



- **Totalité**: un *stagiaire* est au moins *candidat* ou *inscrit*.
- **Exclusion**: un *stagiaire* ne peut être à la fois *candidat* et *inscrit*.
- **OU exclusif**: un *stagiaire* est soit *candidat*, soit *inscrit* mais pas les deux à la fois.

- *Remarque: Cette contrainte est parfois notée **O** (ou) au lieu de **+**.*

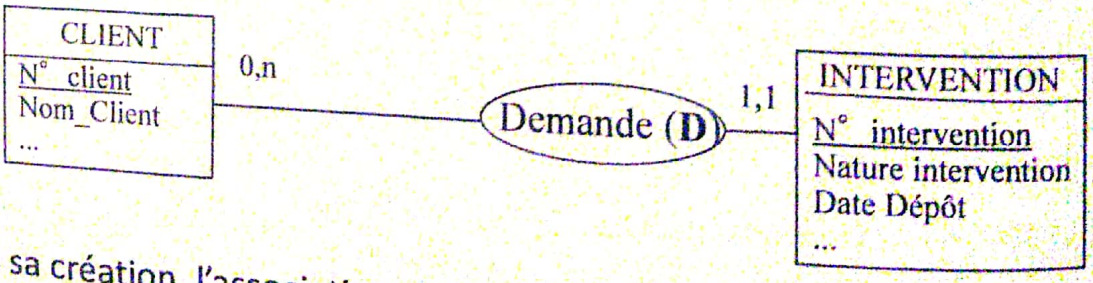
EXTENSIONS: *Bilan sur les contraintes ensemblistes*



EXTENSIONS: Les contraintes de stabilité

SUR UNE RELATION

- Une relation est dite permanente (P) ou définitive (D) si ces occurrences ne peuvent être ni modifiées ni détruites tant qu'existent les occurrences des objets qu'elle relie.

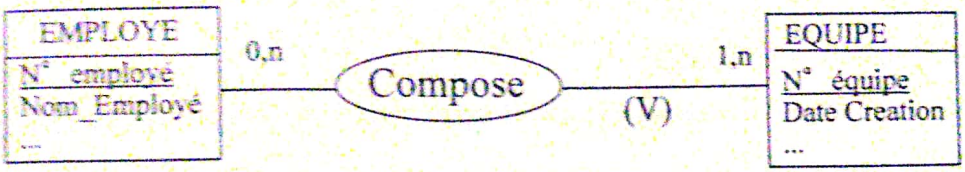


- Dès sa création, l'association *Demande* devient immuable. Une intervention ne pouvant exister que parce qu'elle est reliée à un client donné.

EXTENSIONS: Les contraintes de stabilité

SUR UNE PATTE

- On appelle contrainte de verrouillage (V) la contrainte de stabilité sur une patte. La patte est dite verrouillée si toutes les occurrences de l'association dans lesquels intervient une occurrence de l'entité doivent être créées en même temps que l'occurrence de cette entité.

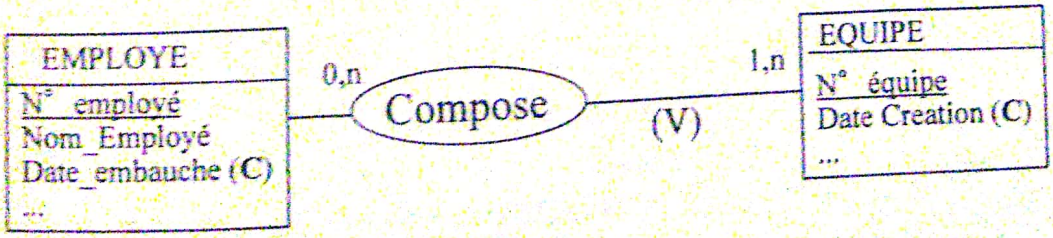


- Cette *équipe* n'existe qu'en ce qu'elle est le regroupement de *personnes* précises. Si l 'on modifie sa composition, ce n'est plus la même équipe. On crée donc en même temps *N°équipe* et les occurrences de *Compose* dans laquelle il intervient. Après création ces occurrences ne sont plus modifiables.

EXTENSIONS: Les contraintes de stabilité

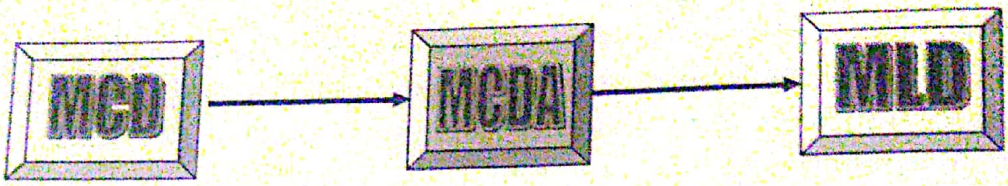
SUR UNE PROPRIETE

- Une propriété peut être déclarée constante (C). Etant donné une occurrence de l'entité ou de l'association qu'elle qualifie, la valeur correspondante de cette propriété ne peut être modifiée.
- L 'identifiant est par définition constant.



- *Date d'embauche* dans EMPLOYE, et
- *Date de création* pour EQUIPE ne peuvent varier au cours de la vie des entités qu'elles qualifient.

EXTENSIONS: MCD Analytique



MCDA = MCD Brut + contraintes + sous-types.

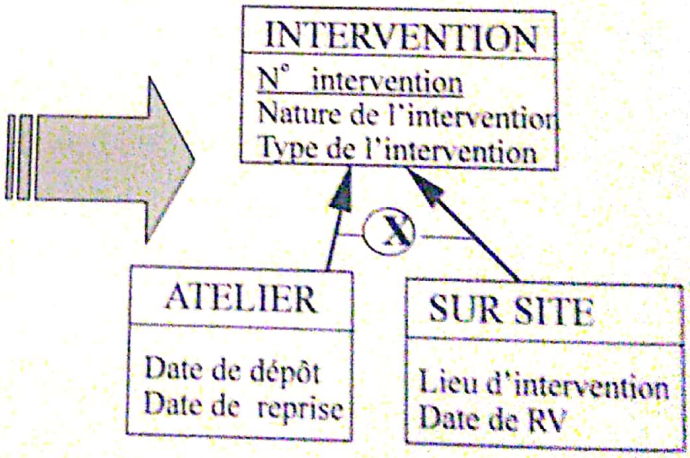
Contraintes sur l'héritage

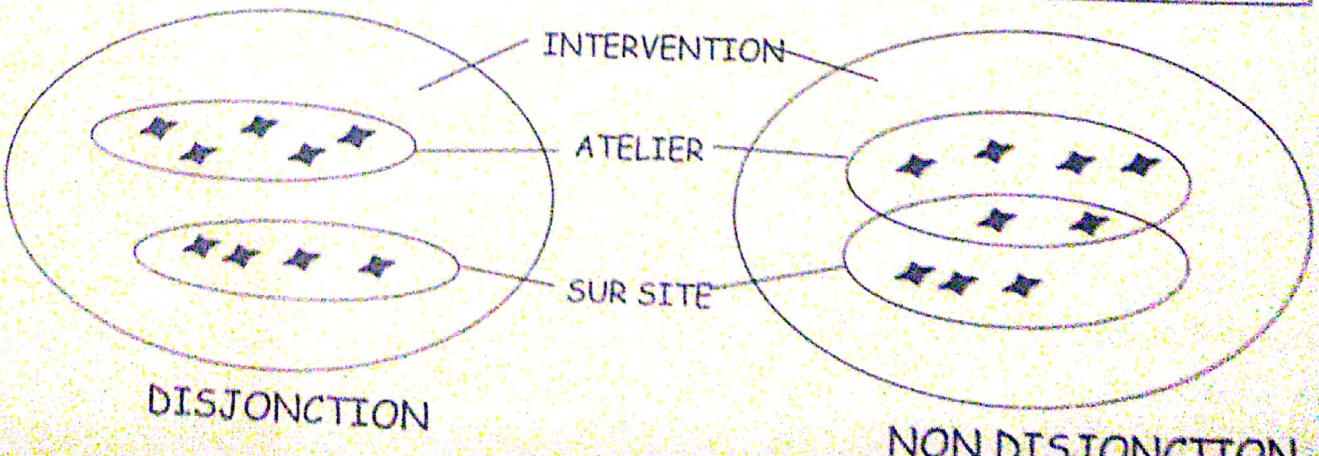
- DISJONCTION: Une occurrence de l'entité générique appartient au plus à l'un des sous-types.
- COUVERTURE: Une occurrence du super-type appartient au moins à l 'un des sous-types.

EXTENSIONS: Contraintes sur l'héritage

Contrainte de Disjonction

- L'intervention doit se dérouler entièrement en atelier ou sur site. Si elle est mixte, on distinguera deux interventions avec deux numéros différents.

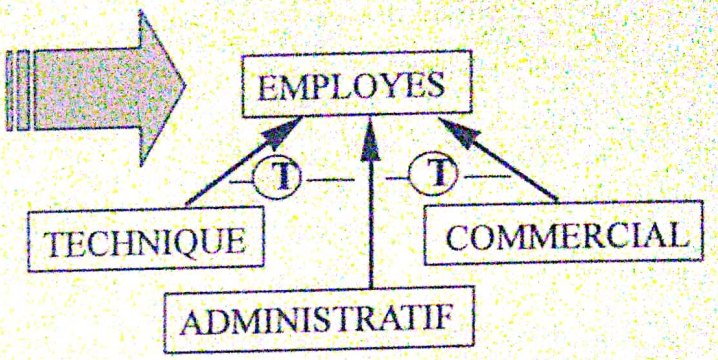


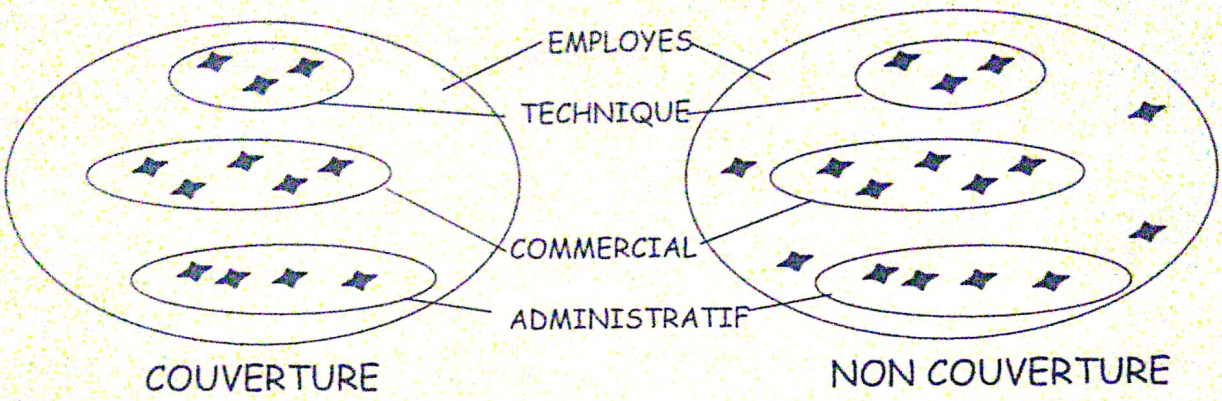


EXTENSIONS: Contraintes sur l'héritage

Contrainte de Couverture

- Cette représentation des employés est correcte si on ne trouve que ces 3 types d'employés, et si l'on ne peut appartenir à deux catégories à la fois.



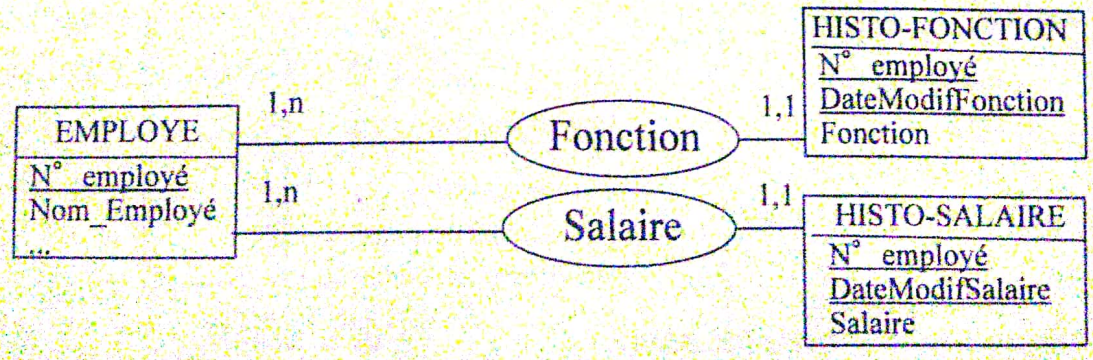


EXTENSIONS: Historisation

Objectif:

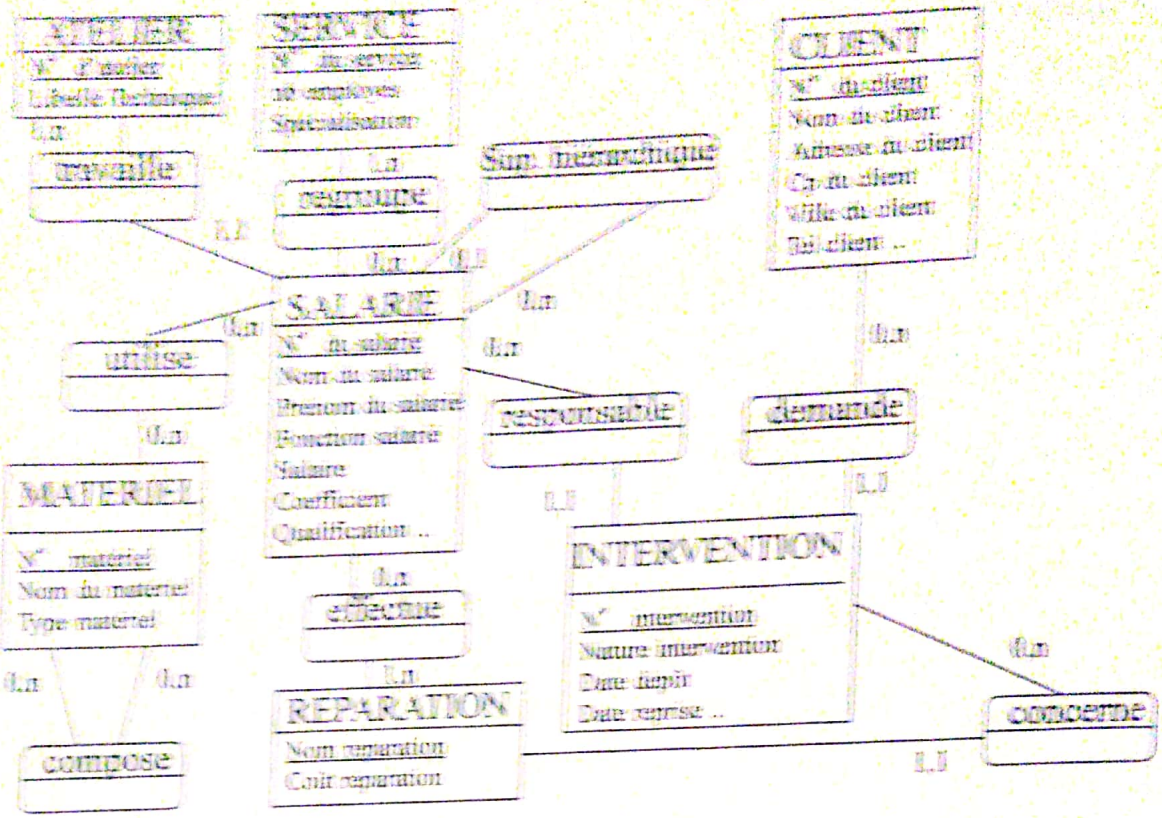
- Pouvoir conserver les différents états successifs d'une propriété non constante.

Exemple:

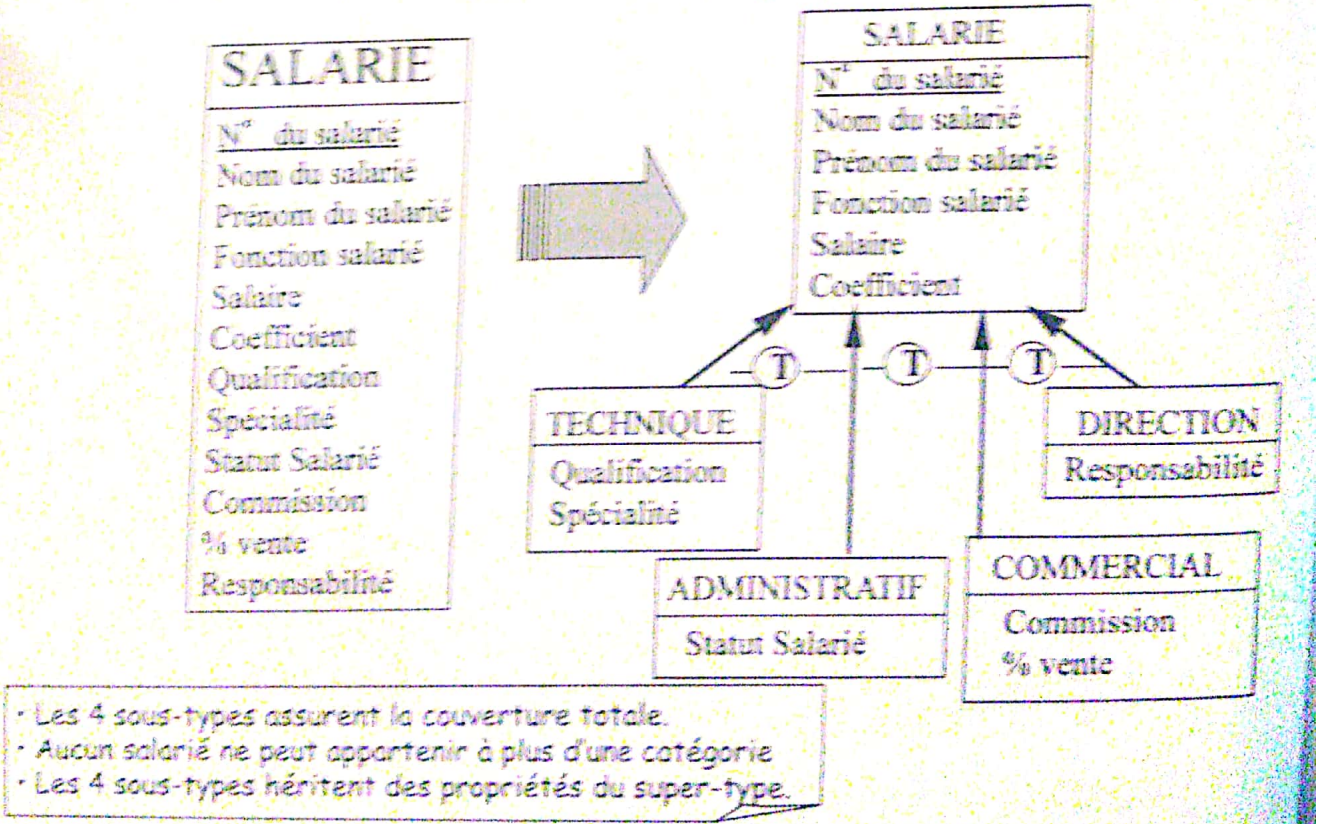


***Remarque**: Si la **fonction** et le **salaire** varient **simultanément**, on ne crée qu'une seule entité **Historisation**.*

EXTENSIONS: Exemple de construction du MCDA (1)

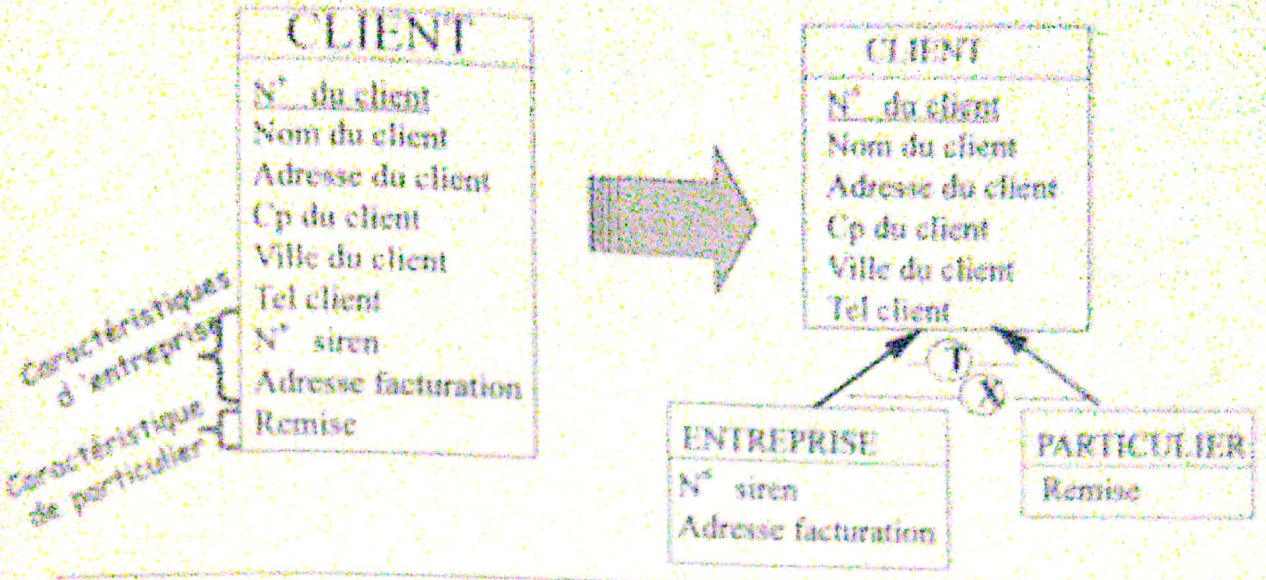


EXTENSIONS: Exemple de construction du MCDA (2)



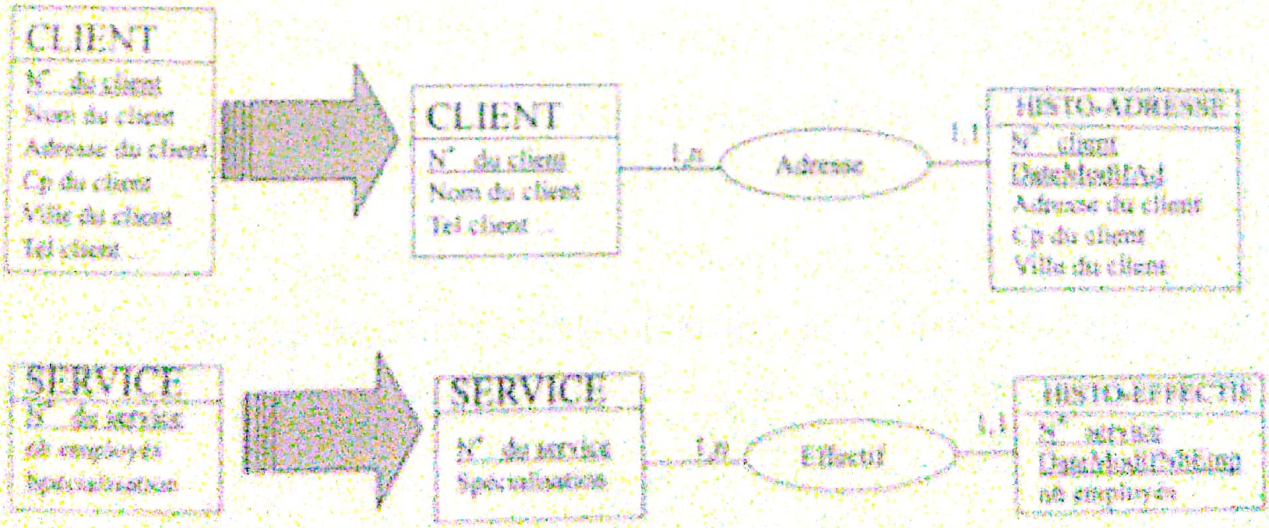
- Les 4 sous-types assurent la couverture totale.
- Aucun salarié ne peut appartenir à plus d'une catégorie
- Les 4 sous-types héritent des propriétés du super-type.

EXTENSIONS: Exemple de construction du MCDA (3)



- Règle de couverture: un client appartient obligatoirement à la catégorie ENTREPRISE ou PARTICULIER.
- Règle de disjonction: un client ne peut appartenir qu'à l'un ou l'autre des sous-types.

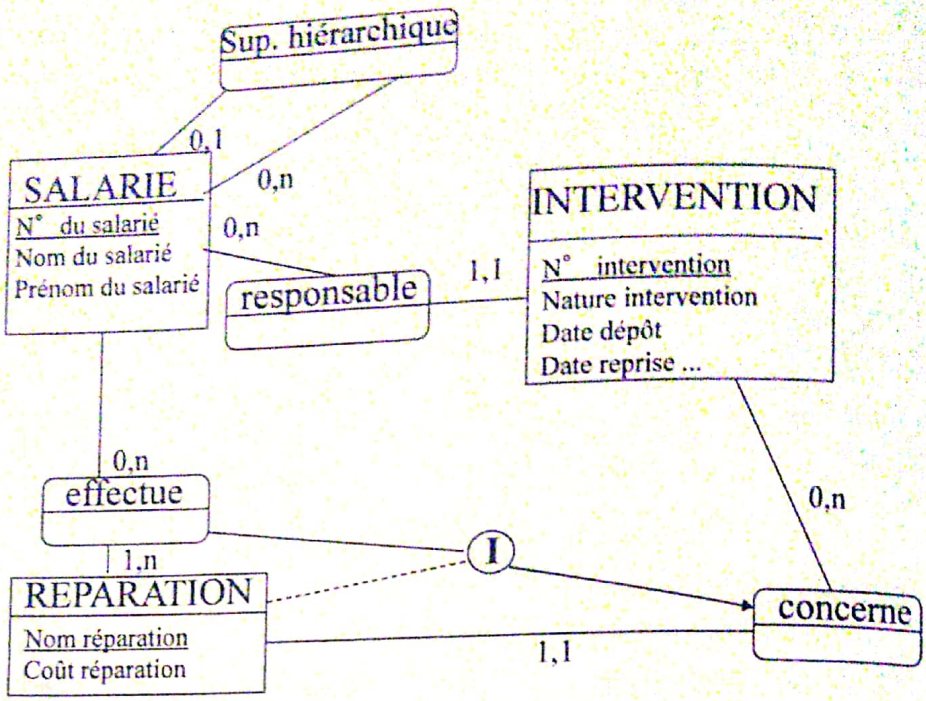
EXTENSIONS: Exemple de construction du MCDA (4)



[illegible]

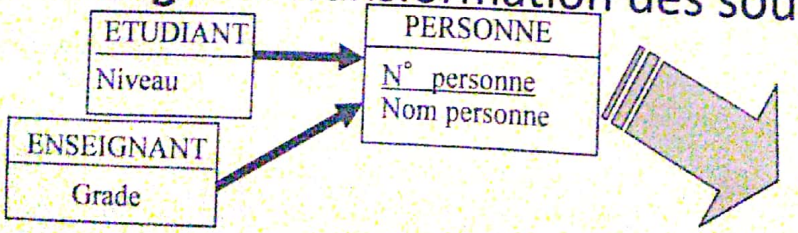
Après cela, on pose les contraintes ...

EXTENSIONS: Exemple de construction du MCDA (5)



EXTENSIONS: Passage au MLD (1)

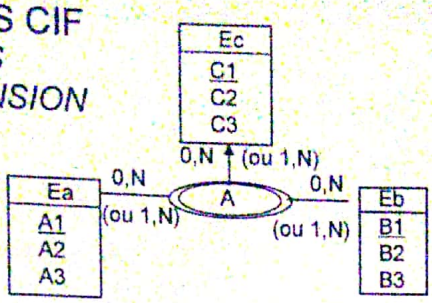
- **Règle 1**: Transformation des entités : **IDEM**
- **Règle 2**: Transformation des Associations à cardinalités multiples: **IDEM**
- **Règle 3**: Transformation des Associations comportant une cardinalité maximale à 1: **IDEM**
- **Règle 4**: Transformation des sous-types d'entité



PERSONNE (N° PERSONNE, NOM PERSONNE)
ETUDIANT (N° PERSONNE, NIVEAU)
ENSEIGNANT (N° PERSONNE, GRADE)

EXTENSIONS:
Passage au MLD (2)

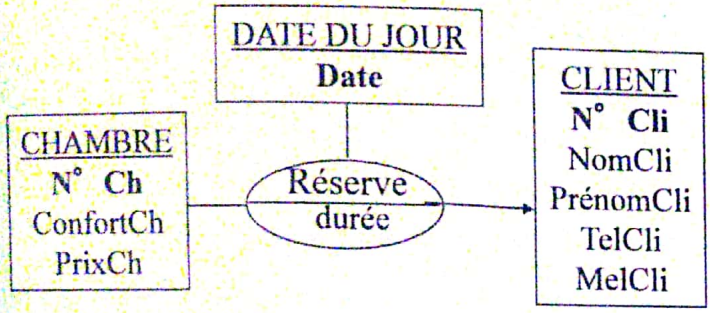
- **Règle 5** :
  TRANSFORMATION DES CIF
  DANS LES ASSOCIATIONS TERNAIRES OU DE DIMENSION SUPERIEURE



⇒ Ea(A1,A2,A3)
Eb(B1,B2,B3)
Ec(C1,C2,C3)
A(B1,A1,C1)

C1 ne fait pas partie de la clé dans A

- EXEMPLE

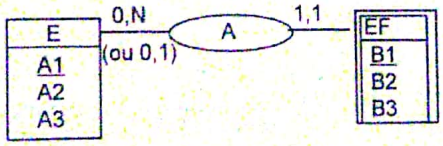


Chambre (**N° Ch**, ConfortCh,PrixCh)
Client(**N° cli**, NomCli, PrénomCli, TelCli, MelCli
Réserve (#**N° ch, Date**, #N° Cli, Durée)

#N° Cli ne fait pas partie de la clé de RESERVE

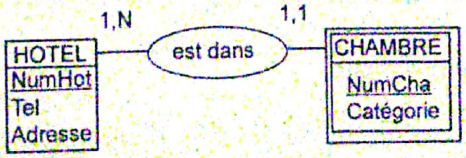
EXTENSIONS:
Passage au MLD (2)

- **Règle 6** :
  TRANSFORMATION DES CLASSES D'ENTITES FAIBLES



⇒ E(A1,A2,A3)
EF(A1,B1,B2,B3)

- EXEMPLE

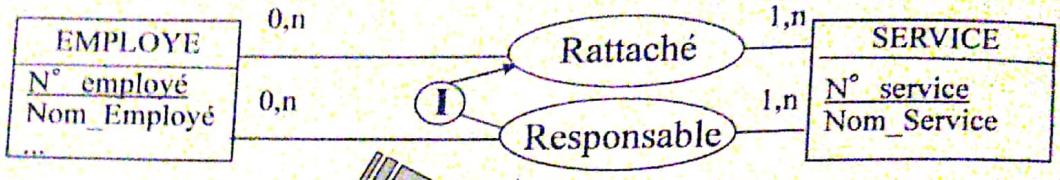


⇒ HOTEL(NumHot, Tel, Adresse)
CHAMBRE(NumHot, NumCha, Catégorie)

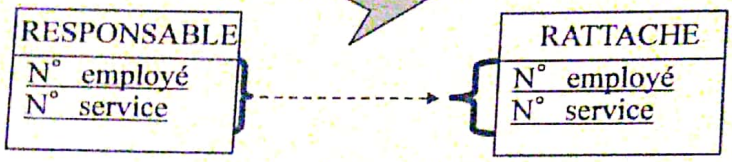
EXTENSIONS:
Passage au MLD (3)

- **Règle 7**: Prise en compte de la contrainte d'inclusion

MCD



MLD



- **On vérifie que :** Algèbre relationelle

Project(RESPONSABLE, N° employé, N° service) ⊂ Project(RATTACHE, N° employé,N° service)

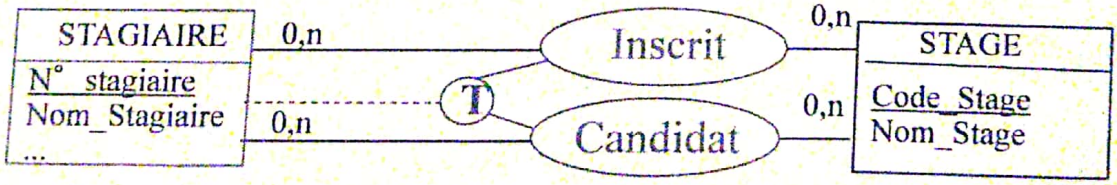
Le couple (N° employé, N° service) de la table RESPONSABLE ne pourra prendre que des valeurs également attribuées à ce couple dans la table RATTACHE.

EXTENSIONS:
Passage au MLD (4)

- **Règle 8**: Prise en compte de la contrainte de totalité

  La vérification se fait en utilisant la projection et l'union.

- **Exemple:**



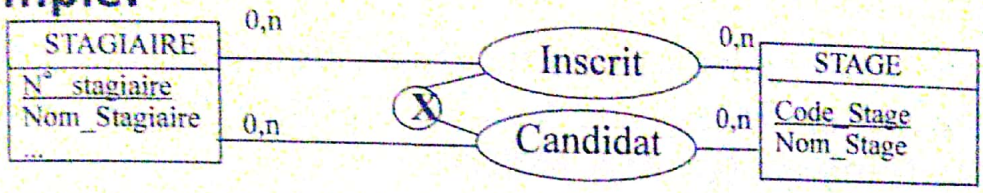
- Les ASSOCIATIONS en jeu sont Inscrit et Candidat
- Le Pivot est STAGIAIRE (clé: N°Stagiaire) => Projection sur N°Stagiaire

On vérifie que :

Project(STAGIAIRE, N°Stagiaire) ⊂
∪(Project(INSCRIT, N°Stagiaire) , Project(CANDIDAT, N°Stagiaire))

EXTENSIONS: Passage au MLD (5)

- **Règle 9**: Prise en compte de la contrainte d'exclusion
  - Les associations liées par une contrainte d'exclusion vont être transformées en relations.
  - L'intersection de la projection de ces relations sur leur clé doit être vide.
- **Exemple:**

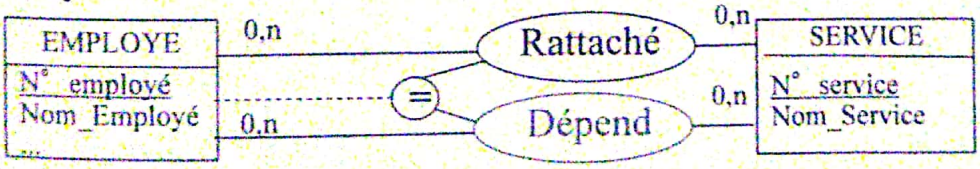


- **On vérifie que :**

Project(INSCRIT, (N°Stagiaire, Code Stage))
∩ Project(CANDIDAT, (N°Stagiaire, Code Stage)) = Ø

EXTENSIONS: Passage au MLD (6)

- **Règle 10**: Prise en compte de la contrainte d'égalité
  - Équivaut à une inclusion réciproque.
  - On vérifie l'inclusion dans les deux sens
- **Exemple:**



- **On vérifie que :**

Project(DEPEND, N° Employé) ⊂ Project(RATTACHE, N° Employé)
ET
Project(RATTACHE, N° Employé) ⊂ Project(DEPEND, N° Employé)

EXTENSIONS: Conclusion

- Les extensions présentées rapprochent MERISE des modèles orientés objet
  - Spécialisation **Versus** Héritage
  - Entités faibles **Versus** ≈ Composition
- Elles permettent une transformation en un schéma relationnel normalisé *au sens de la 3eme forme normale*
- Les extensions sur les modèles des données ont été davantage intégrées que les extensions sur les traitements (cf. AGL MEGA).
- MERISE OO versus UML...

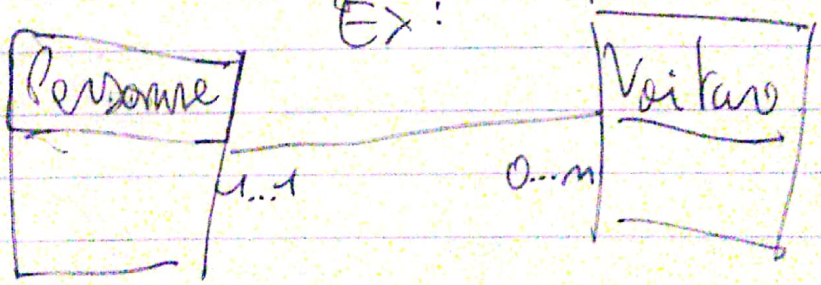
Chaque classe → relation

si manque d'identificateur d'objet dans le diagramme de classe on ajoute une clé.

REGLE 1: existence de la cardinalité x..1 d'un côté de l'association:

- Chaque classe → une table
- Chaque attribut de la classe se transforme en un champ de table relationnelle
- L'identifiant de la classe associée à la cardinalité ?..1 devient la clé la clé étrangère de l'autre classe.

Ex:



DC

Personne (ID Personne, ...)

Voiture (ID Voiture, ..., Propriétaire)

DEA

CIR et clé étrangère

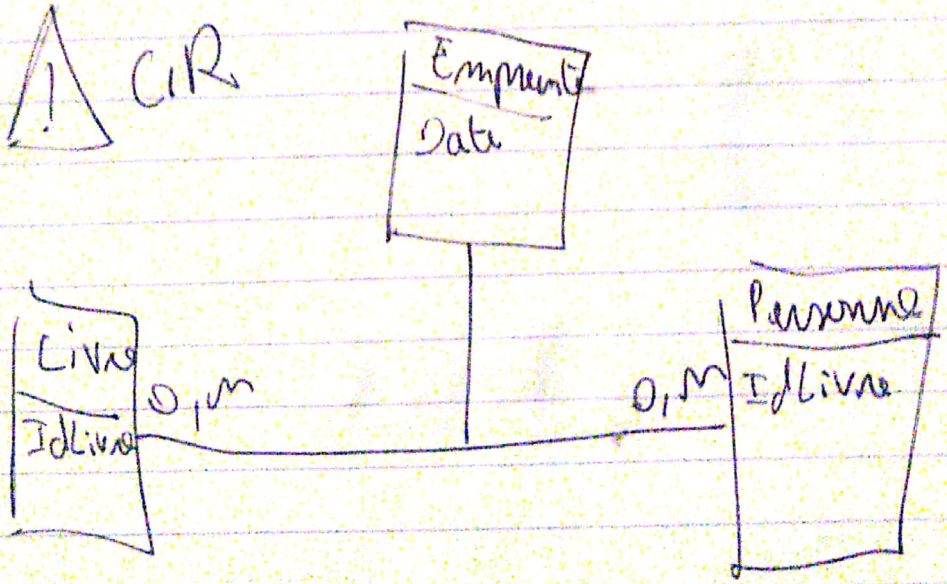
La clé étrangère doit être inclus dans la clé primaire.

Tous les ID qui sont en clé étrangère doivent tous être des clé primaires dans la table référence.

REGLE 2 : cardinalité 1...n

- Chaque classe devient une table
Chaque attribut de classe devient un champs de la table.

- L'association devient une table.
ID : ID des 2 objets + les attributs de l'association.

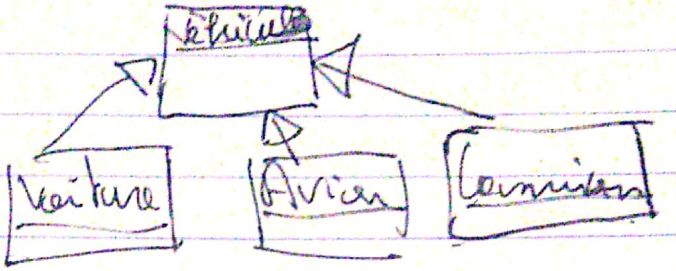


↳ Livre (IdLivre, ...)
Personne (IdPers, ...)
Emprunts (IdEmprunt, IdLivre, IdPers, ...)

REGLE 3 : Présence de généralisation

Méthode Ascendante :

Une table, avec tous attributs des classe + un attribut pour distinguer les types d'objet.



VEHICULE (IdVehicule, TypeVehicule, ...)

Méthode Distinction

On crée une table pour chaque spécialisation. Elle comprend les attributs communs et les attributs propres.

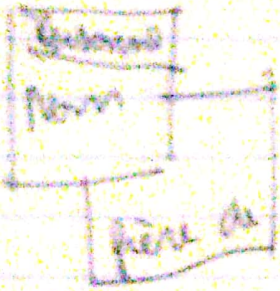
Méthode Descendante

Une classe de la généralisation avec les attributs communs.
Pour chaque spécialisation qui référence la table générale et possède les attributs propres.

[RÈGLE 4] Relation réflexive

Père de (Nom fils, Nom père)

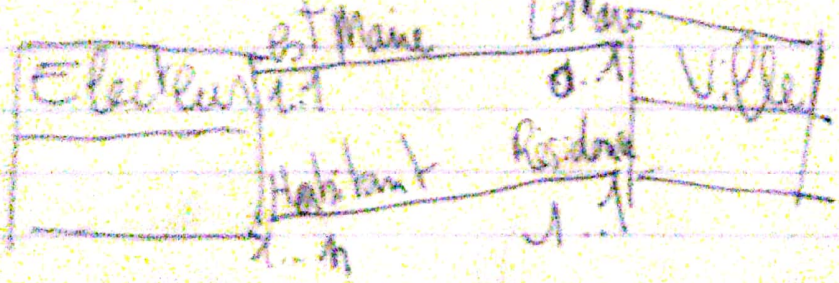
[RÈGLE 5] Relation réflexive symétrique



Personne (Nom)

Frère (nom, nom).

[RÈGLE 6] : circuit



On devrait faire: Electeur (ElecteurID, ..., Ville (VilleID, ..., Le maire))

Il faut supprimer une dépendance:

Electeur (ElecteurID, ..., résidence)

Ville (VilleID, ..., Le maire)

RÈGLE 7 Agrégation :

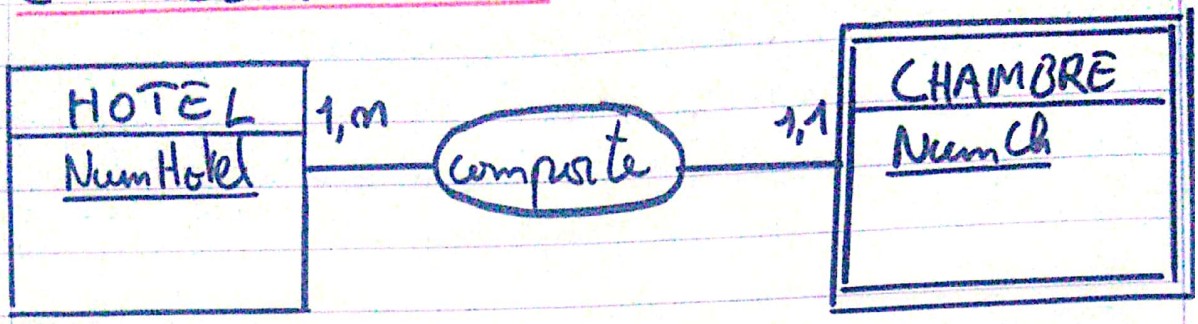
- Comme les associations classiques.

RÈGLE 8 Composition :

- Comme une association 1:n
- On ajoute à la classe la clé de la classe composite.

SGBD 3

ENTITÉS FAIBLES



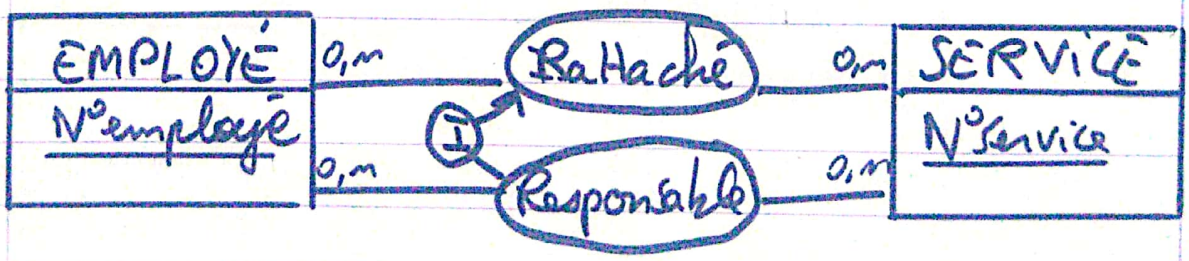
→ Le Numéro d'une chambre ne suffit pas à l'identifier car on gère plusieurs hôtels.

CONTRAINTES ENSEMBLISTES

INCLUSION

→ L'ensemble des occurences d'une association est compris dans l'ensemble des occurences d'une autre

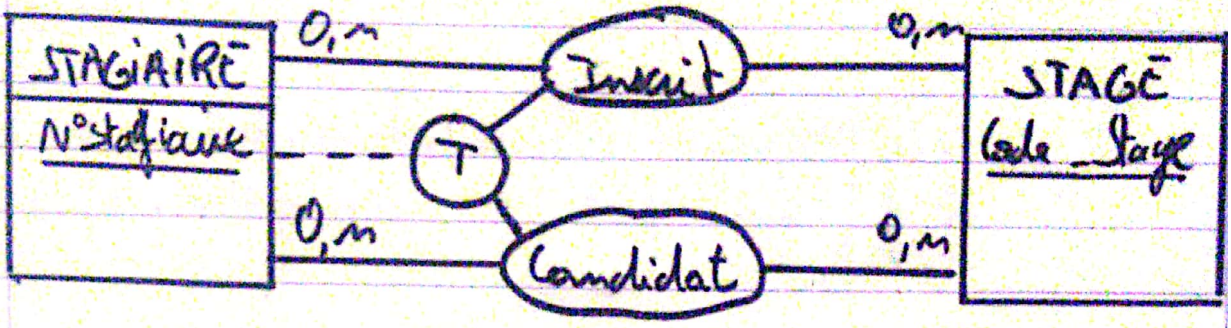
"Un employé ne peut être responsable que des services auxquels il appartient"



→ Tout les couples (employé, service) participant à Responsable doivent figurer dans les couple (employé, service) participant à Rattaché.

TOTALITÉ

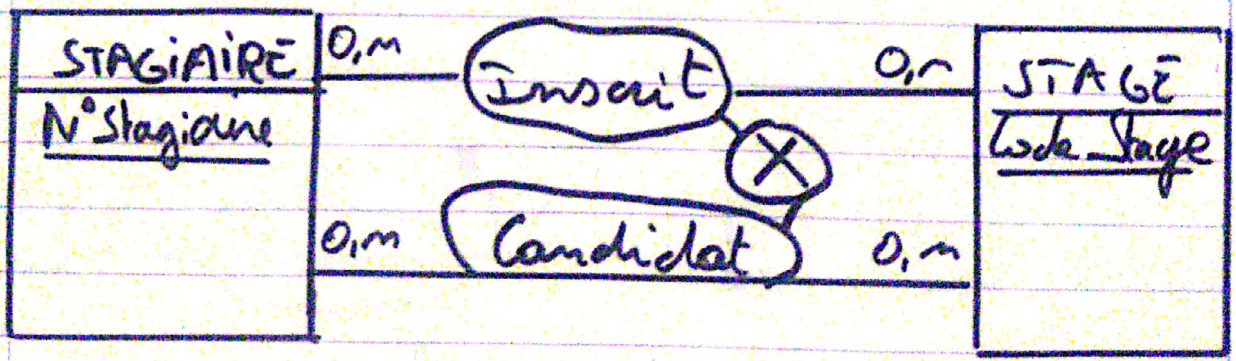
→ Toutes les occurence d'une entité impliquée dans deux associations ou plus sont présentes dans l'une d'entre elles.



→ Stagiaire participe au moins une fois à l'une des deux associations.
(Stagiaire est le pivot)

EXCLUSION I

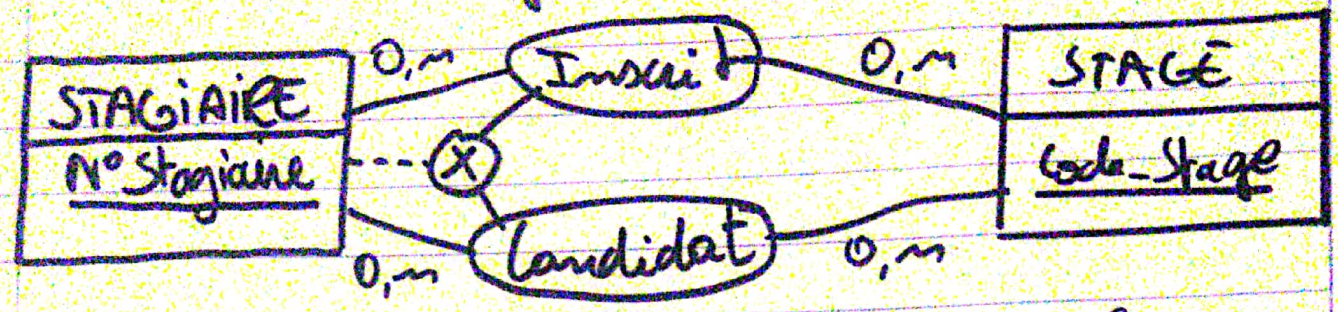
→ Interdit qu'une occurence d'une entité impliquée dans 2 association ou plus soit presente dans 2 d'entre elles.



→ Un Stagiaire ne peut être candidat et inscrit dans la même formation.

EXCLUSION II

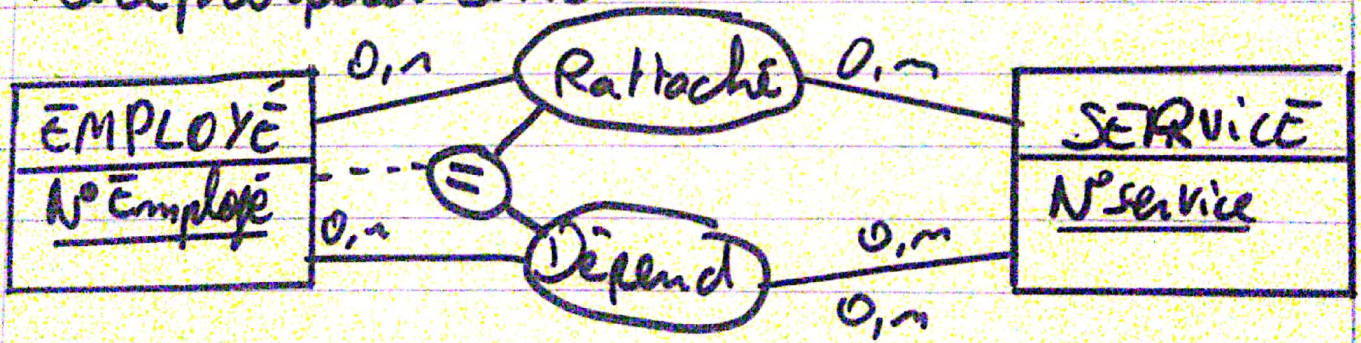
→ Exclusion symétrique, avec Pivot



→ Un stagiaire ne peut être à la fois candidat et inscrit quelque soit le Stage.

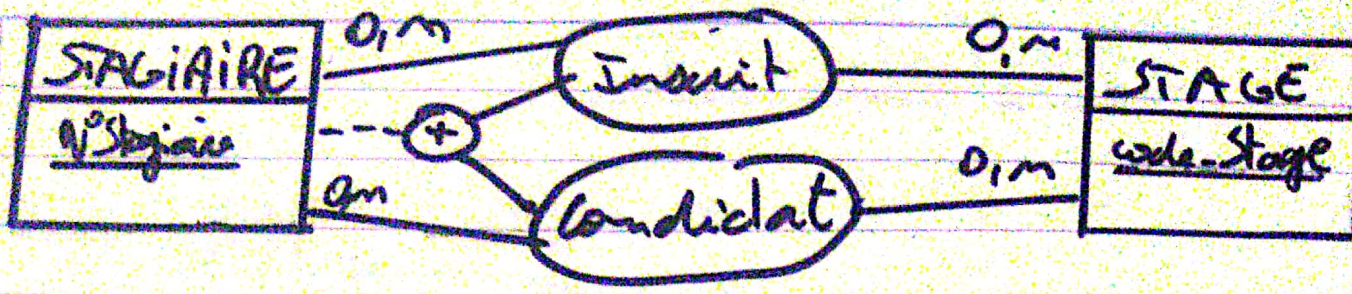
ÉGALITÉ

→ L'ensemble des valeurs du pivot qui sont dans une association doivent être inclues dans l'ensemble des valeurs participant à l'autre association et réciproquement.



OU EXCLUSIT

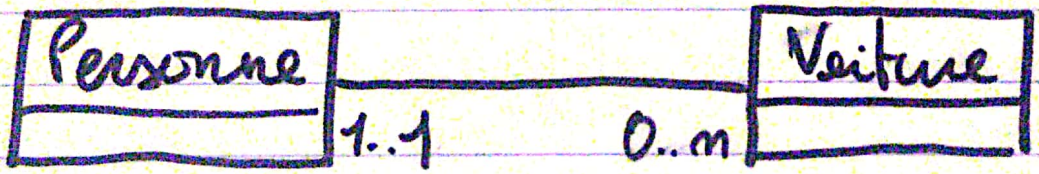
→ Une valeur du pivot doit participer être présente soit dans une des associations auxquelles elle participe mais pas les deux



CONVERSION DC -> MLD

REGLE 1 : Cardinalité X...1

- Chaque classe devient une table
- Chaque Attribut de la classe devient un champs de la table
- L'identifiant de la classe associée à la cardinalité X...1 devient la clé étrangère de l'autre classe



Personne (idPersonne, ...)

Voiture (idVoiture, ..., #propriétaire)

OCL : Object Constraint Language

↳ exprime des règles de cohérence de données

↳ Le diagramme de classe ne permet pas d'exprimer tous les types de règles

↳ OCL est une extension de UML

Deux types de contrainte :

→ Statiques : multiplicité et contraintes de l'héritage

→ Dynamique : expressions logiques dans le langage OCL

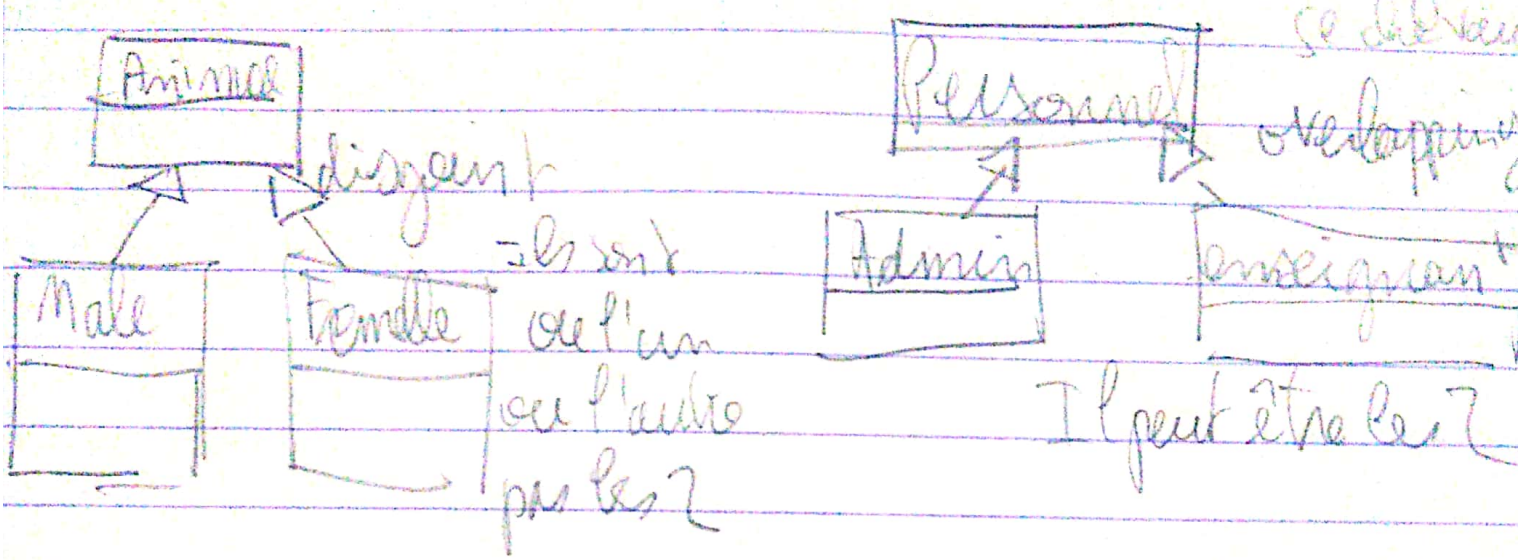
Statiques :

- {ordered} : Les objets de l'association sont ordonnée
- {frozen} : L'association n'est pas modifiable
- {addOnly} : On ne peut qu'ajouter des objets à l'association (pas de

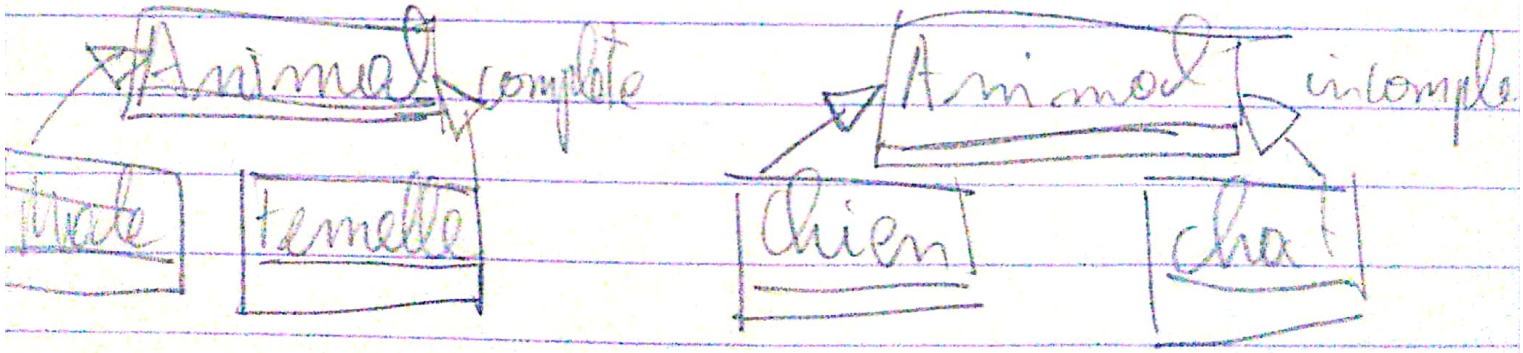
suppression d'objets).

- Disjoint / Overlapping

Les instances des sous classes sont disjointes (ou chevauchantes)



- Complete / Incomplete : toute instance de la superclasse est/n'est pas obligatoirement une instance d'une/plusieurs une des sous classes



⇒ Ces 2 aires peuvent être combinées

Dynamiques

- OCL : pseudo maths et pseudo Prog
- Logique propositionnelle
- A partir d'un objet

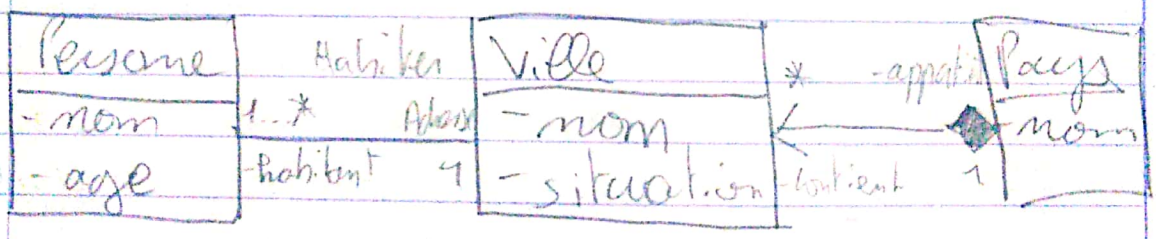
Syntaxe :

Contrainte = Contexte + invariant

↓ Objet concerné par la contrainte

↓ Expr logique de la contrainte

Self = this (désigne l'objet)
Utilisation des opérateurs logiques (ET, OU, IMPLIQUE)



Pour un objet de la classe Personne
self : objet de la classe personne
self.nom : nom d'une personne
self.adresse : référence la ville où habite la personne
self.adresse.appartient pays où habite la personne
Pour un objet de la classe
self.nom : nom de la ville
self.habitants : liste des objets Personnes

Qui habitent dans la ville

-> Contrainte simple sur les attributs :

"Un étudiant a au moins 16 ans"

context = Etudiant
invariant = self.age >= 16

"Un étudiant en App a au moins 20 ans"

ou

context = Etudiant en App
invariant = self.age >= 20

"si le sexe est féminin alors le titre est madame, si le sexe est masculin alors le titre est monsieur"

self.sexe = féminin IMPLIES self.titre = madame
self.sexe = masculin IMPLIE > self.titre = monsieur.

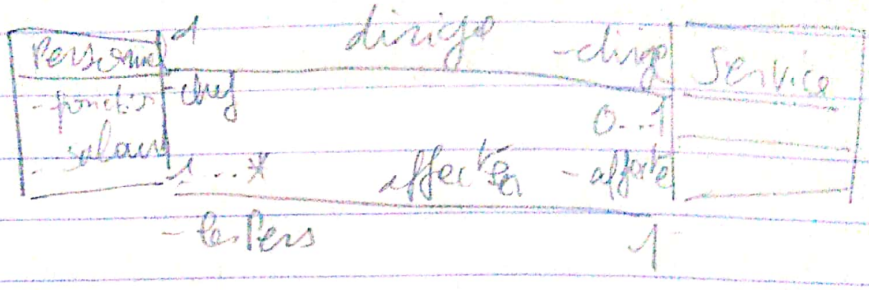
-> Contraintes sur les associations simples

"Employés touchent salaire supérieur au salaire min fixé par leur société"

context = salarié
invariant = self.salaire > self.employeur.salaire min

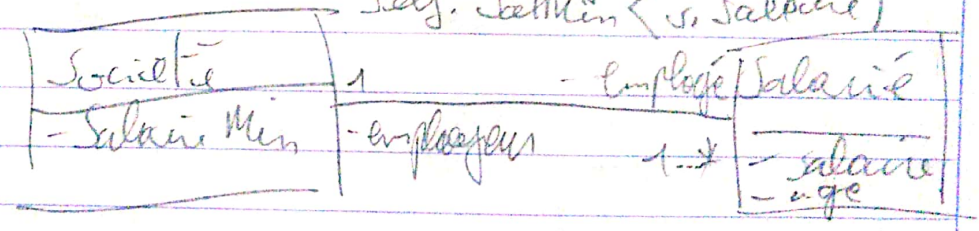
Société	1	employé	salarié
salaire min	employeur	1..*	salaire

"Les services sont dirigés par des cadres"
context : Service
invariant :



→ Contraintes sur les associations multiples

"Le salaire minimum d'une société doit être inférieur au salaire de tous les employés"
context : Société
invariant : self.employé.forAll(s : Salarié | self.salMin < s.salaire)



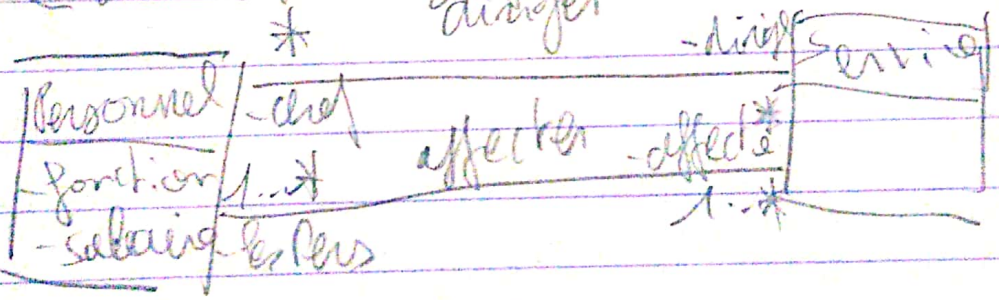
Primitives sur les collections

isEmpty : vrai si collection est vide
size : nb élément collection
forAll : vrai si tous les éléments vérifient la contrainte spécifiée
exists : vrai si au moins 1 élément respecte la contrainte spécifiée
includes : vrai si la collection contient l'élément spécifié

includesAll : vrai si contient tous les éléments spécifiés.
union/intersection : retourne union/intersection avec autre collection
select : retourne le sous ensemble d'objet donc les éléments vérifient la condition spécifiée
reject : idem mais garde que les éléments ne vérif pas contrainte

→ Exemple : isEmpty et size

"Les techniciens ne peuvent diriger un service"
context : Personnel
invariant : self.fonction=tech IMPLIES self.diriger->



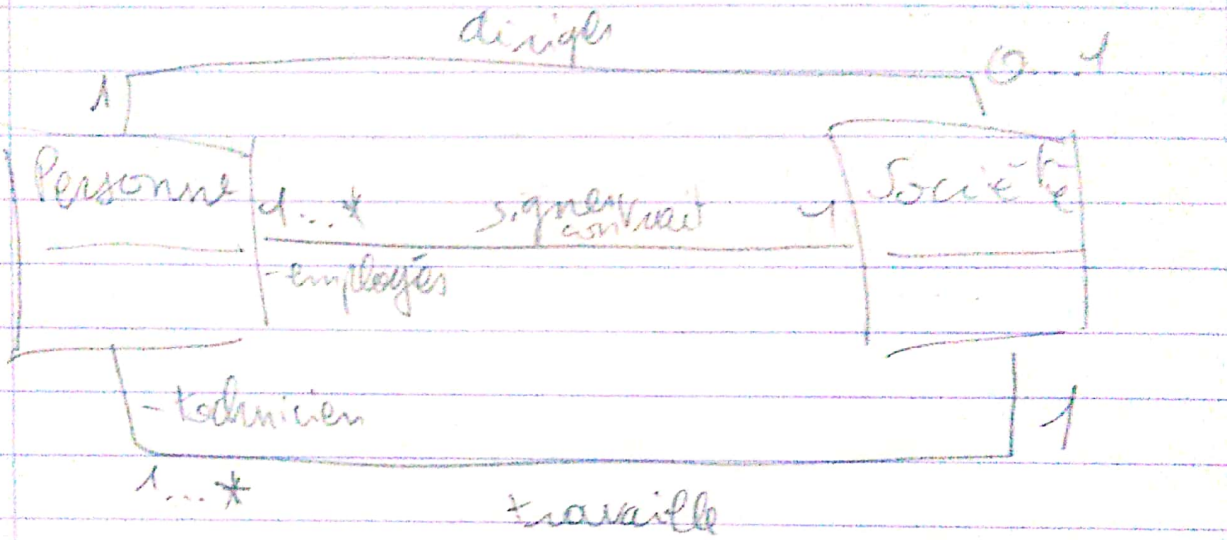
"Les cadres sont affectés à 1 seul service et les tech à 3"

context : Personnel
invariant : self.fonction = cadre IMPLIES
self.diriger.size = 1
self.fonction = tech IMPLIES
self.affecté = 3

Sem 6 — C.D.A — Cours 6

"Chaque salarié emploie au moins un salarié senior"

contexte : Société
invariant : self.employés.exists
(s : Salarié | s.age > 50)

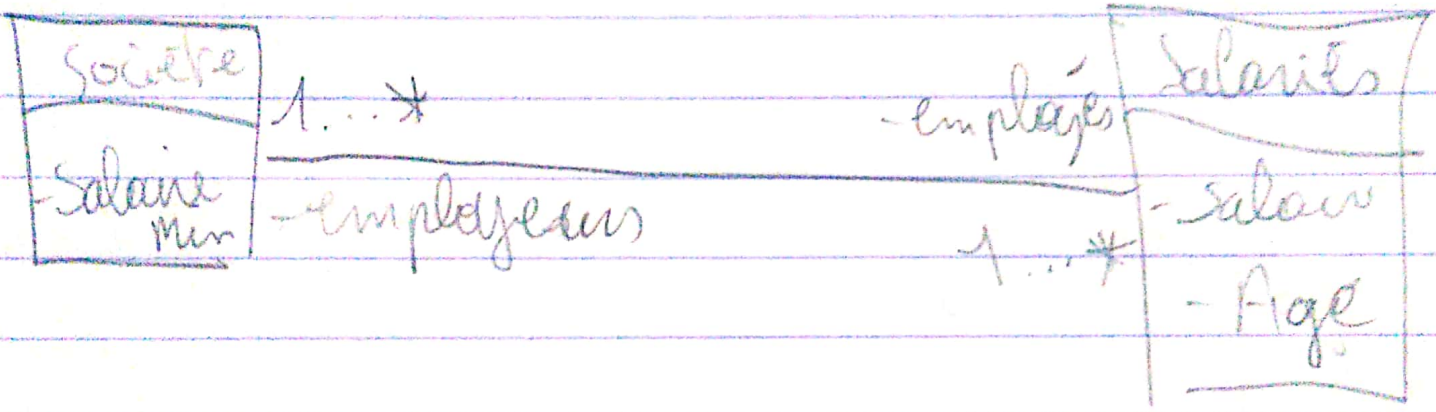


- "Les sociétés signent obligatoirement un contrat avec leur dirigeant"
contexte : société
invariant : self.employés.includes
(self.directeur)

- "Les sociétés signent obligatoirement un contrat avec leur techs"
contexte : société
invariant : self.employé.includes(self.techniciens)

- " Chaque société emploie au moins 10 salariés séniors")

context : Société
invariant : [self.employe->select (s : Salarié | s.age > 50)]->size() > 10



- "Une société n'emploie que des salariés femme

context : Société
invariant : [self.employé->reject(s : Salarié | s.age < 35)]->isEmpty