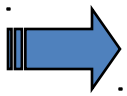
LA METHODE MERISE

Objectif

Informatiser un système d'information

- **Permet le passage d'un système non informatisé à un système d'information automatisé**
- **C'est une méthode de conception et de développement des systèmes d'informations**



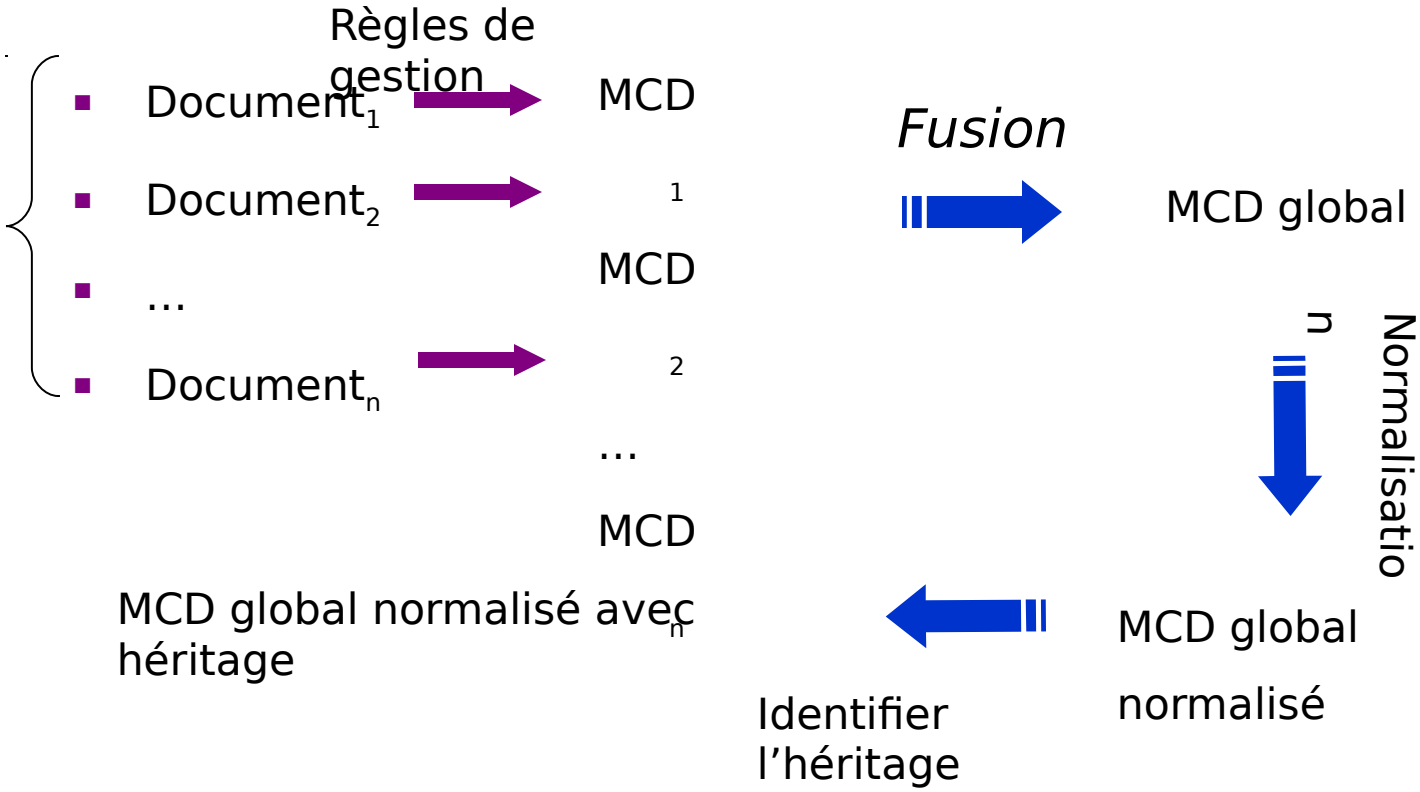
Deux versions de MERISE

- **MERISE 1 : Conception et développement des bases de données relationnelles**
- **MERISE 2 : Conception et développement des bases de données avec l'aspect Orienté Objet : Notion d'héritage,…**

Les niveaux de la méthode MERISE

Niveau / Aspect	Donnée	Traitement
Conceptuel	Modèle Conceptuel de Données	Modèle Conceptuel de Traitement
Logique	Modèle Logique de Données	Modèle Organisationnel de Traitement
Physique	Modèle Physique de Données	Modèle Opérationnel de Traitement

Construction du MCD global



MCD – Rappels

- **Propriété :** donnée que l'on perçoit sur une entité ou association

➡ **3 types de propriétés**

- **Calculée** : Calculée en fonction des autres propriétés : Moyenne_générale=∑note*Coef/ ∑Coef
- **Concaténée** : Composée de plusieurs propriétés : Étudiant = (nom , prénom)
- **Élémentaire** : Ni calculée, ni composée : nom , prénom , note , coefficient.

➡ Les propriétés calculées doivent être éliminées du MCD pour réduire l'espace mémoire de stockage

➡ Les propriétés concaténées doivent être éliminées du MCD pour réduire le temps de recherche des informations

MCD – Rappels

- **entité :** ensemble de propriétés de mêmes caractéristiques

→ Matière = **(**CodeMatière**,** NomMatière**)**

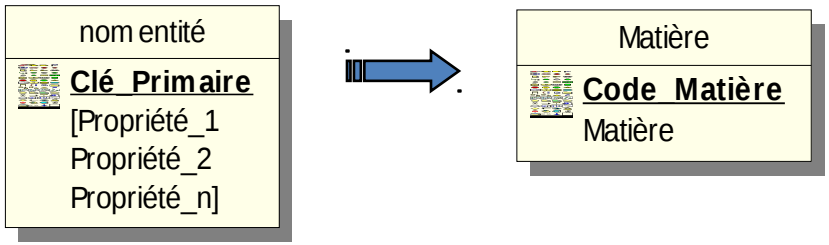
Nom de l'entité

Les deux propriétés de l'entité

- **Clé primaire d'une entité :** Groupe minimum de propriétés identifiant le reste.

Exemple : la clé primaire de l'entité Matière est CodeMatière

Formalisme d'une entité

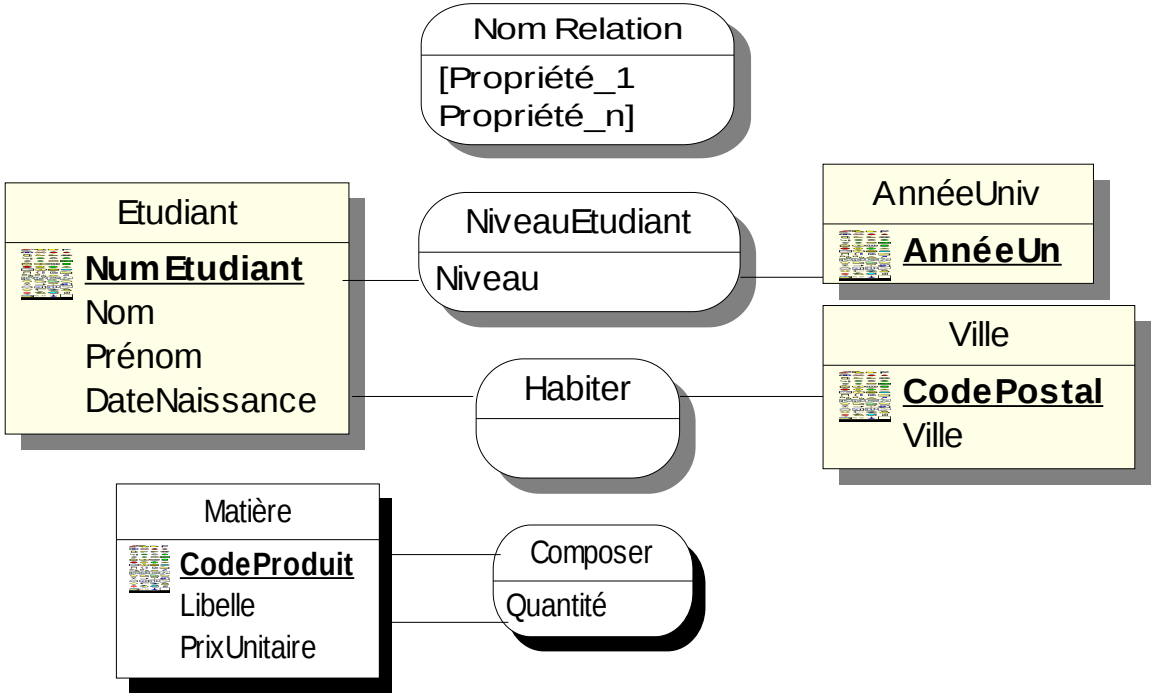


MCD – Rappels

- **Relation :** association de 2 ou plusieurs entités

⇒ Une relation peut ne pas contenir de propriétés

Formalisme d'une relation :



Exemple (Modèle Entité/Relation)

Cas particulier :

Relation qui associe une seule entité

MCD – Rappels

- **Clé primaire d'une Relation :** Composition des clés primaires des entités formant cette association.

  La clé primaire d'une relation n'est définie que dans le cas où l'association contient au moins une propriété

Exemple :

- la clé primaire de la relation NiveauEtudiant est (NumEtudiant , AnnéeUn)
- la relation Habiter n'a pas de clé.

MCD – Rappels

- **Dépendance fonctionnelle :**

$Propriété1 \xrightarrow{DF} Propriété2$ si la valeur de la 1ère entraine celle de la 2ème.

Exemple : $NumEtud , Nom \xrightarrow{DF} Prénom$ et $NumEtud \xrightarrow{DF} Nom , Prénom$

- **Dépen. Fonc. Élémentaire :**

$P_1 \xrightarrow{DF} P_2$ si $P_1 \xrightarrow{DF} P_2$ et aucune partie stricte de P_1 n'identifie P_2

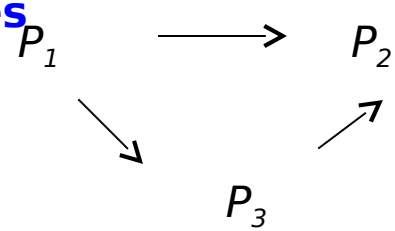
Exemple : $NumEtud , Nom \xrightarrow{DF} Prénom$ et $NumEtud \xrightarrow{DF} Nom , Prénom$

MCD – Rappels

- **Dépen. Fonc. Élém. Directe :**

  $P_1 \rightarrow P_2$ directement si elle n'existe aucune propriété P_3 telle que $P_1 \rightarrow P_3$ et $P_3 \rightarrow P_2$

⇒ **Pas de transitivités entre deux propriétés**



Exemple : *Les deux dépendances NumEtud → Prénom et NumEtud → Nom sont directes : ni Nom → Prénom , ni Prénom → Nom*

⇒ **La conception d'une BD dont les dépendances entre les propriétés reliées par une DFED permet de réduire l'espace mémoire de stockage.**

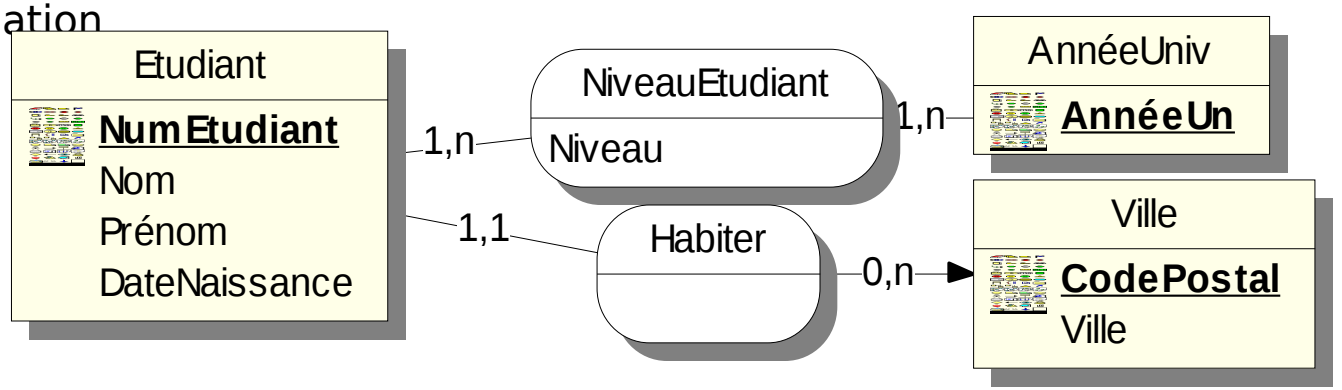
MCD – Rappels

- **Cardinalités :** Les Cardinalités d'une entité à travers une relation est le nombre d'occurrences de cette association trouvé pour une occurrence de l'entité.

Cardinalités maximales : le nombre maximum d'occurrences de la relation

Cardinalités minimales : le nombre minimum d'occurrences de la relation

Exemple :



0 ≤ Cardinalités minimales ≤ 1 et : 1 ≤ Cardinalités maximales ≤ n

MCD – Rappels

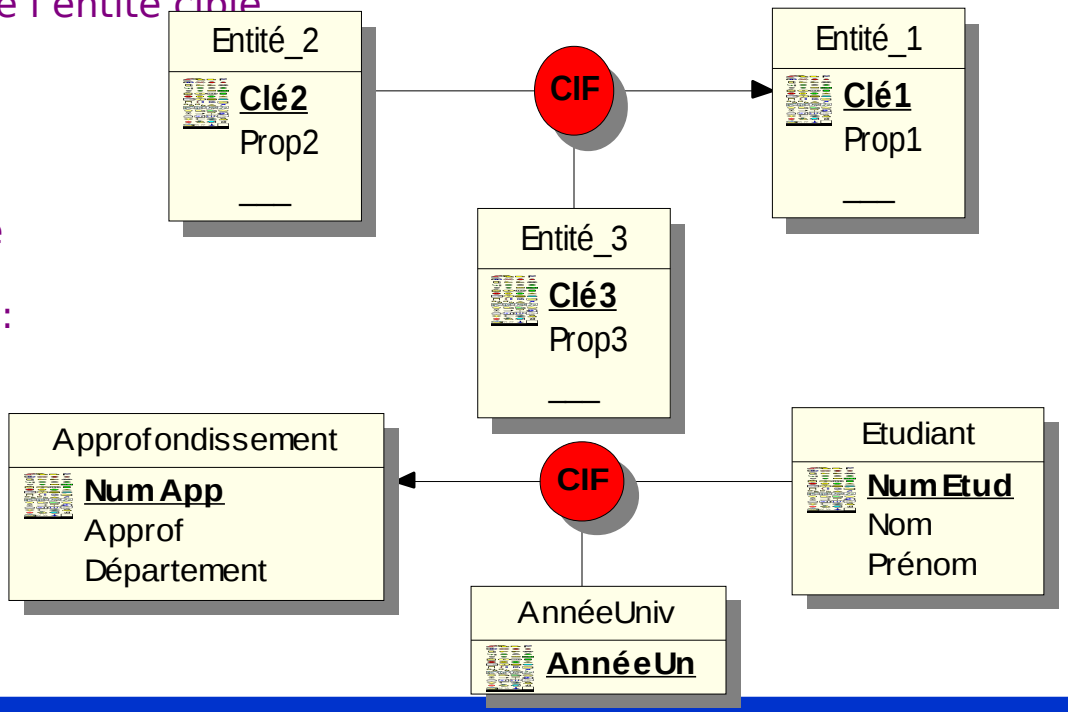
- **CIF :** La Contrainte d'Intégrité Fonctionnelle formalise une DF entre une ou plusieurs entités dites origines et une entité dite cible.

Pour toute occurrence des entités origines correspond au plus une occurrence de l'entité cible

Formalisme d'une CIF :

- Entité_1 : Cible
- Entité_2 et _3 : origines

Exemple :



MCD – Étapes de Construction

- **Étape 1 : Établissement de La liste des propriétés.**

  - Établir la liste à partir de chaque document recueillis.

  ⇒ La propriété apparaît sous deux formes dans un document :

  Valeur⇒ interpréter la valeur pour identifier la propriété.
  Exemple : Emploi du temps
  4ème année ⇒ *Niveau*

  Propriété : valeur
  Exemple : Étudiant : X Y
  ⇒ *Étudiant*

  - Éliminer les synonymes et régler les polysèmes.

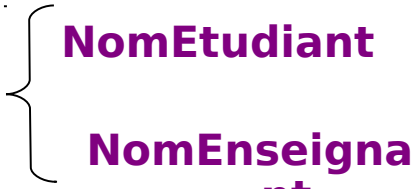
MCD – Étapes de Construction

- **Synonymes : deux ou plusieurs propriétés ayant la même signification**
- **Polysème : Une propriété qui se trouve dans le même document ou autre avec plusieurs sens.**

Exemple :

2 synonymes : on ne garde que l'un des deux

Nom



On précise le sens à la propriété polysème

MCD – Étapes de Construction

- **Étape 2 : Établissement du dictionnaire de données**

Nom abrégé	Nom détaillé	Nature	Type	Taille (en octet)	Remarques
….	….	….	….	….	….

- **Nom abrégé** de la propriété
- **Nom détaillé** de la propriété
- **Sa nature : E** (Élémentaire), **CA** (Calculée) et **CO** (Concaténée)
- **Type : N** (Numérique), **A** (Alphabétique), **AN** (AlphaNum.), **Date, Image, OLE, …**
- **Remarque :** si la propriété est calculée, on écrit l'expression du calcul; si elle est concaténée, on écrit l'expression de décomposition.

MCD – Étapes de Construction

- **Étape 3 : Établissement du Graphe de Dép. Fonct. Élém. (GDF)**

2 Méthodes

- Établir le GDF d'un document et fusionner les MCDs par la suite
- Établir le GDF global pour établir un MCD global directement

Il faut éliminer les transitivités du GDF pour rendre les dépendances fonctionnelles élémentaires des DFED.

MCD – Étapes de Construction

- **Étape 3 :**

Ex

Emploi du Temps (4ème année) *An.Un : ../..*

Semestre 2 - Semaine 25

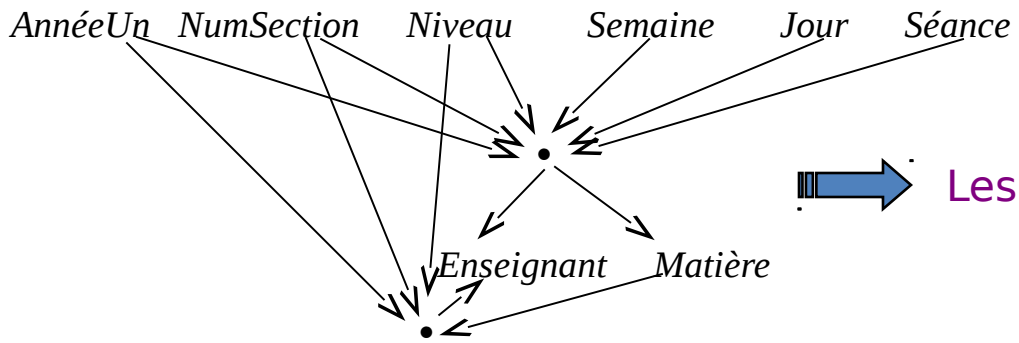
8H	*10*	*12*	*14….*	*18H*
	S1 MSD – M. X….			
Lundi				
	S2			
….				

Règles de gestion

- Les étudiants sont répartis en deux sections
- Une matière est enseignée au cours d'une année à une section par un seul enseignant
- L'emploi du temps peut être changé d'une semaine à une autre

MCD – Étapes de Construction

- **GDF**

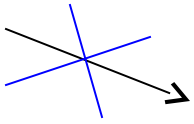


Les dépendances sont Élémentaires

- **Éliminer les transitivités**

AnnéeUn , *NumSection* , *Niveau* , Semaine , Jour , Séance → *Matière*

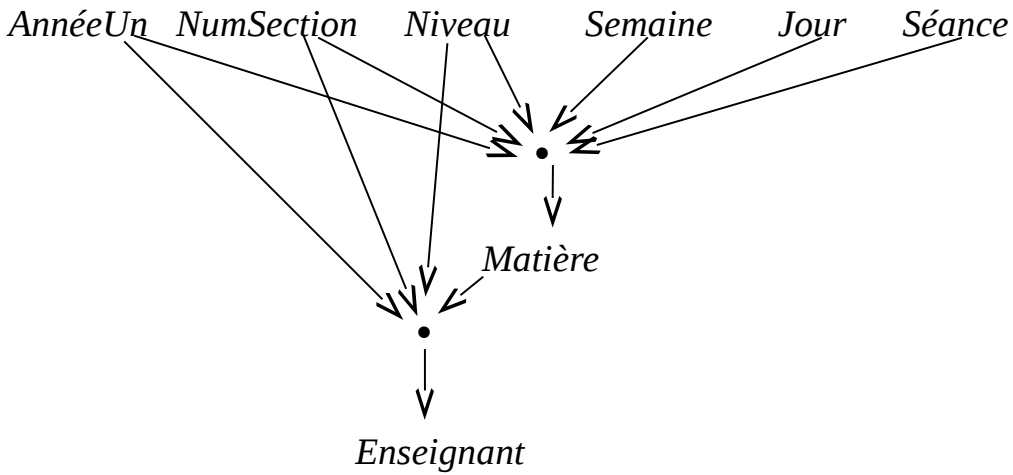
AnnéeUn , *NumSection* , *Niveau* , Semaine , Jour , Séance → *Matière* , *AnnéeUn* , *NumSection* , *Niveau*



Enseignant

MCD – Étapes de Construction

- **Graphe de dépendances élémentaire directes**



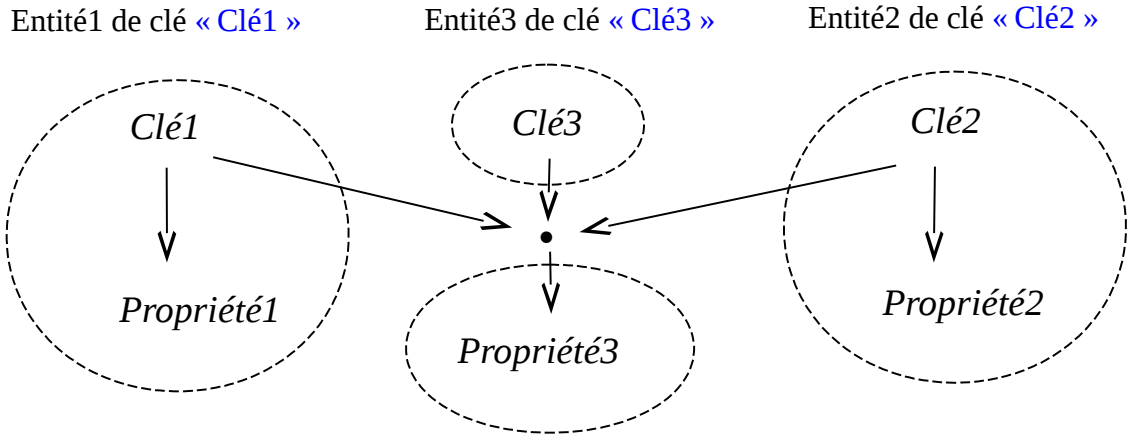
GDF Directe : graphe sans transitivités

MCD – Étapes de Construction

- **Étape 4 : Établissement du MCD**

Un **MCD** est formé des **entités** et des **relations ⇒ deux règles**

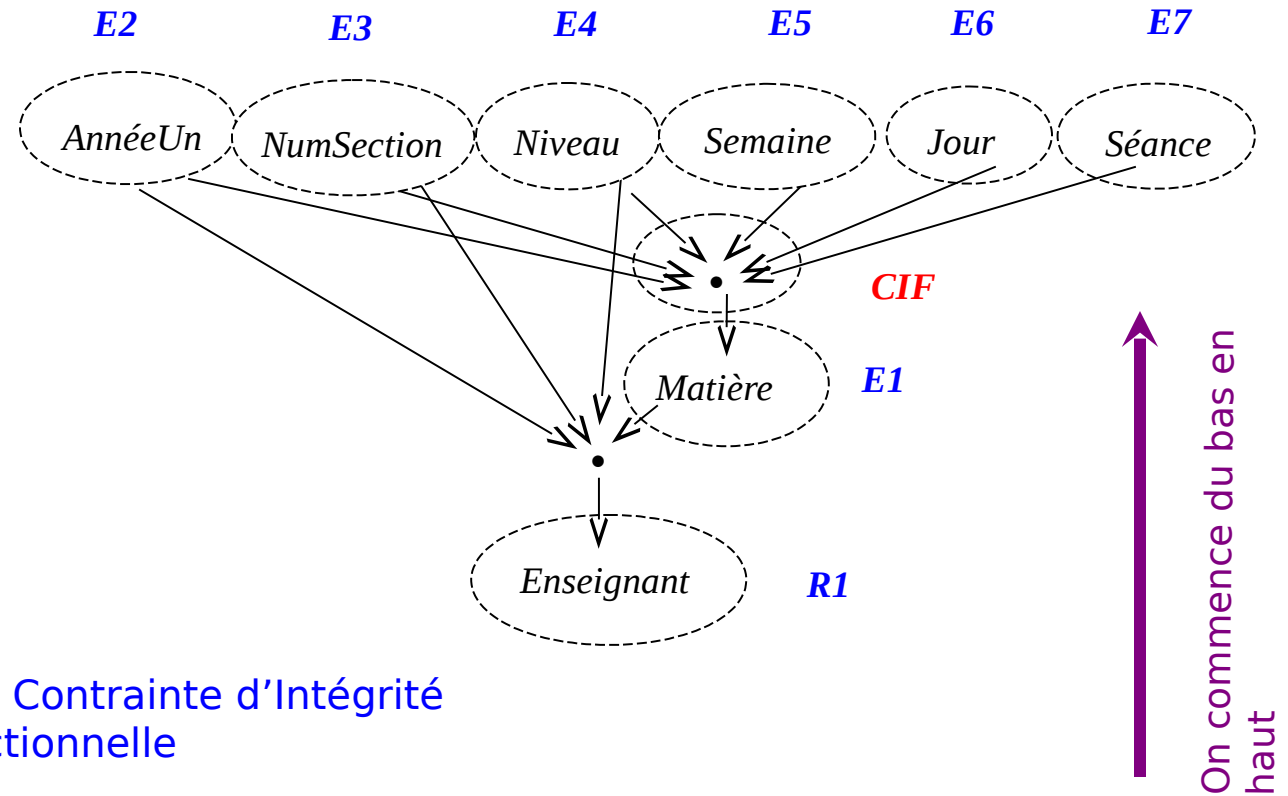
- Les propriétés qui dépendent d'une seule propriété forment une entité
- Les propriétés qui dépendent d'une propriété composée des « clés des entités » forment une relation qui associe ces entités



Relation1 de clé (Clé1 , Clé2 , Clé3)

MCD – Étapes de Construction

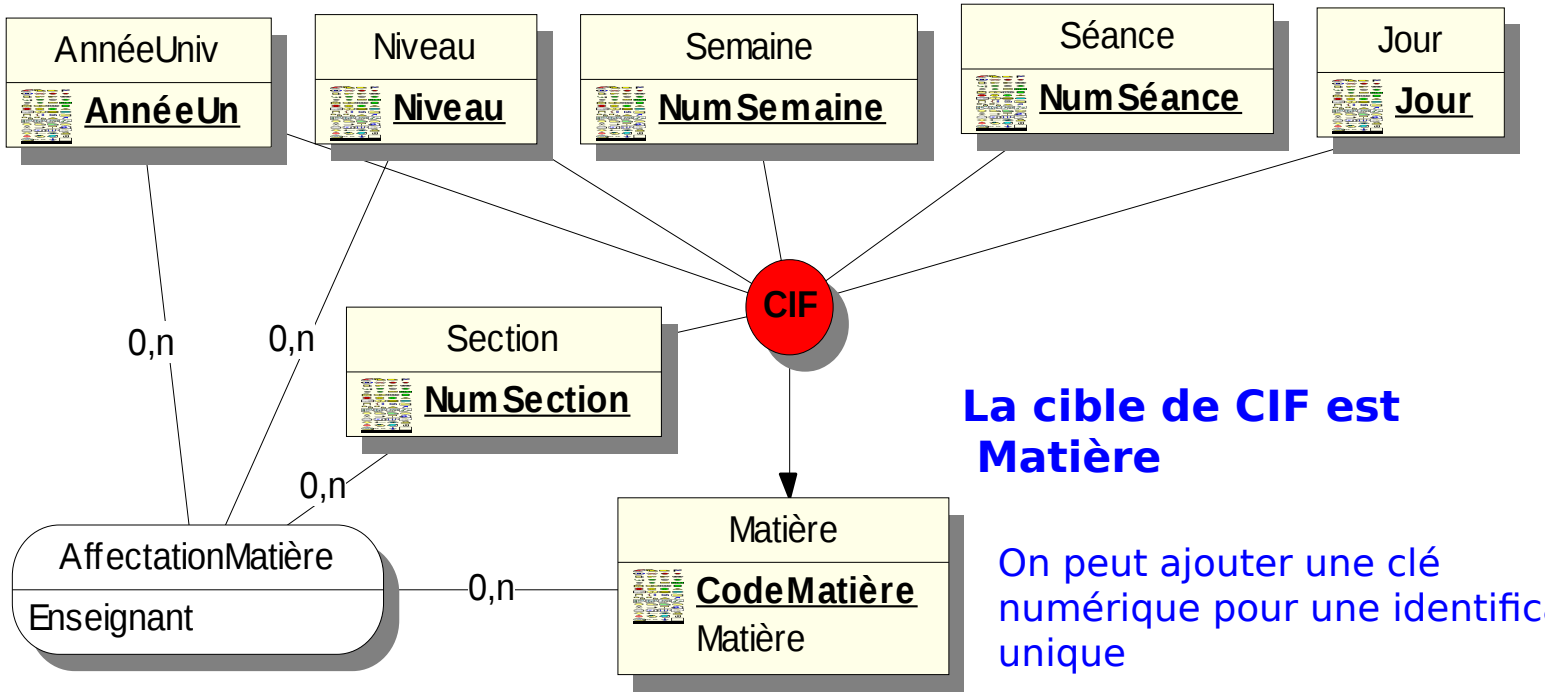
- **Étape 4 : Exemple**



CIF : Contrainte d'Intégrité Fonctionnelle

MCD – Étapes de Construction

- **Étape 4 : Exemple**



La cible de CIF est Matière

On peut ajouter une clé numérique pour une identification unique

MCD – Règles de la fusion

- **La fusion de plusieurs MCD revient à fusionner deux par deux**

➡ Étudier Les règles de la fusion de deux MCD

➡ Chaque MCD est constitué d'un ensembles d'entités et de relations :

MCD1 = {E1/R1}

MCD2 = {E2/R2}

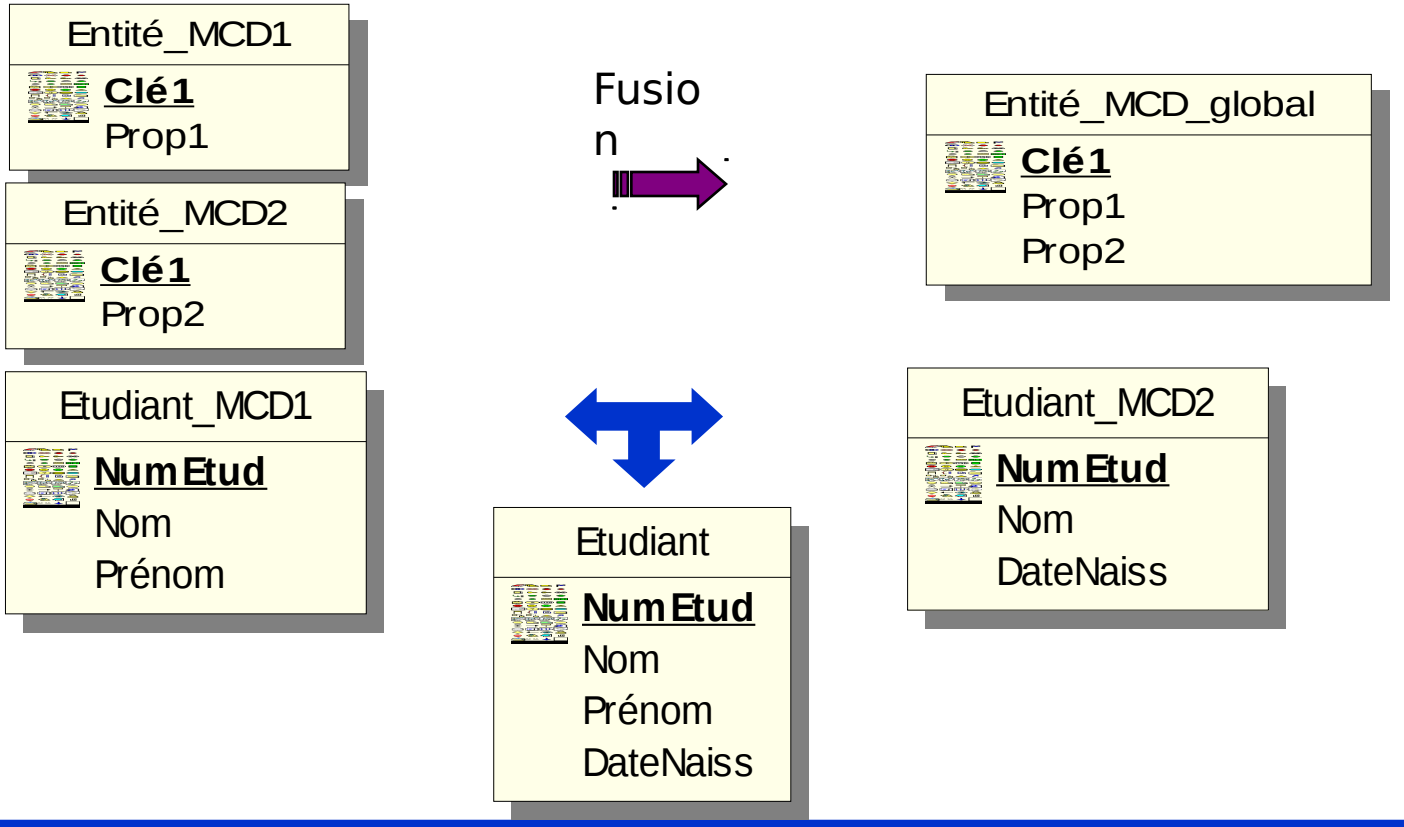
➡ On spécifie trois cas :

- Il existe une entité commune entre les 2 MCDs
- Il existe une relation commune entre les 2 MCDs
- Il existe une propriété commune entre les 2 MCDs

MCD – Règles de la fusion

- **Cas 1 : Entité Commune entre les 2 MCDs**

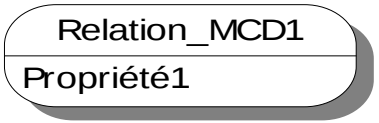
Exemple

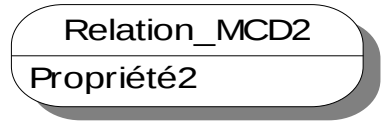


MCD – Règles de la fusion

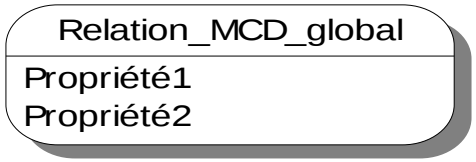
- **Cas 2 : Relation Commune entre les 2 MCDs**

Elle relie les mêmes entités dans les deux MCDs



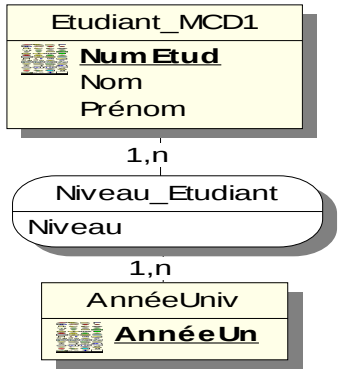


Fusion

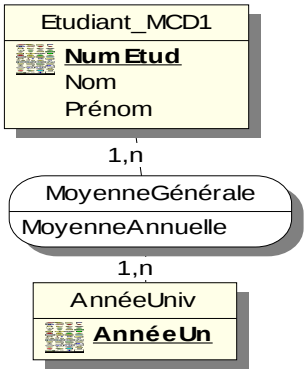


Exemple

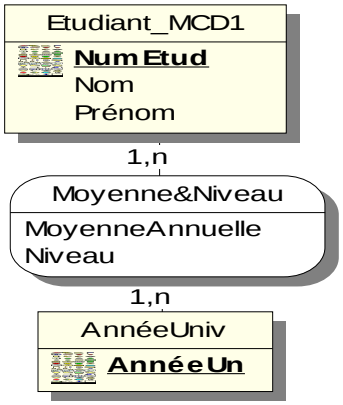
MCD 1



MCD 2



MCD global



MCD – Règles de la fusion

- **Cas 3 : Propriété Commune PC entre les 2 MCDs**

On spécifie Quatre cas :

- PC ∈ E1(MCD1) et PC ∈ E2(MCD2)
- PC ∈ E1(MCD1) et PC ∈ R2(MCD2)
- PC ∈ R1(MCD1) et PC ∈ E2(MCD2)
- PC ∈ R1(MCD1) et PC ∈ R2(MCD2)

} **Même cas**

On traitera les trois cas :

- PC ∈ E1(MCD1) et PC ∈ E2(MCD2)
- PC ∈ E1(MCD1) et PC ∈ R2(MCD2)
- PC ∈ R1(MCD1) et PC ∈ R2(MCD2)

MCD – Règles de la fusion

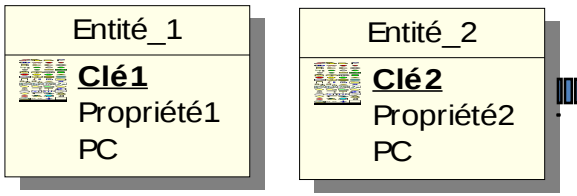
- PC ∈ E1(MCD1) et PC ∈ E2(MCD2)

On distingue deux cas :

- PC a les caractéristiques **des deux entités à la fois**
- PC a les caractéristiques **d'une seule entité** supposée E1

- PC a les caractéristiques de **E1 et E2 à la fois**

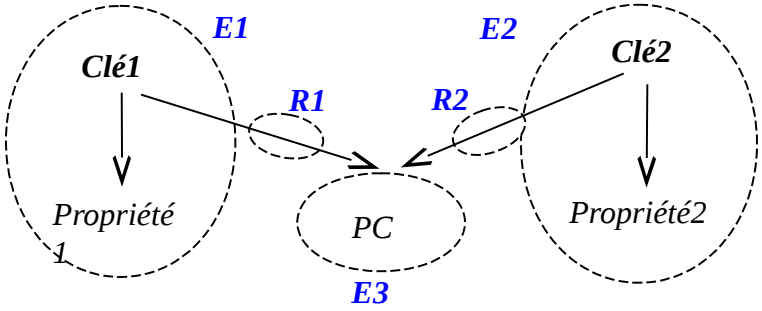
Formalisme :

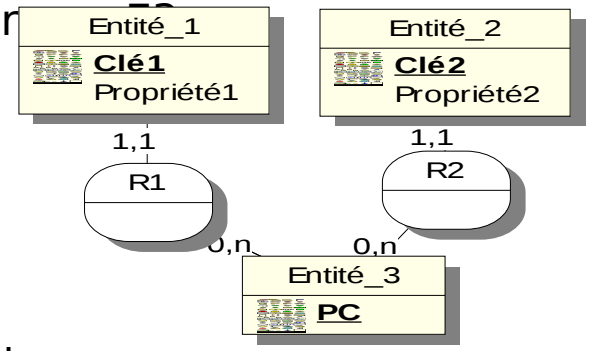


.- On ne peut favoriser ni E1 ni E2.

- On doit éliminer la PC des deux entités

- Créer

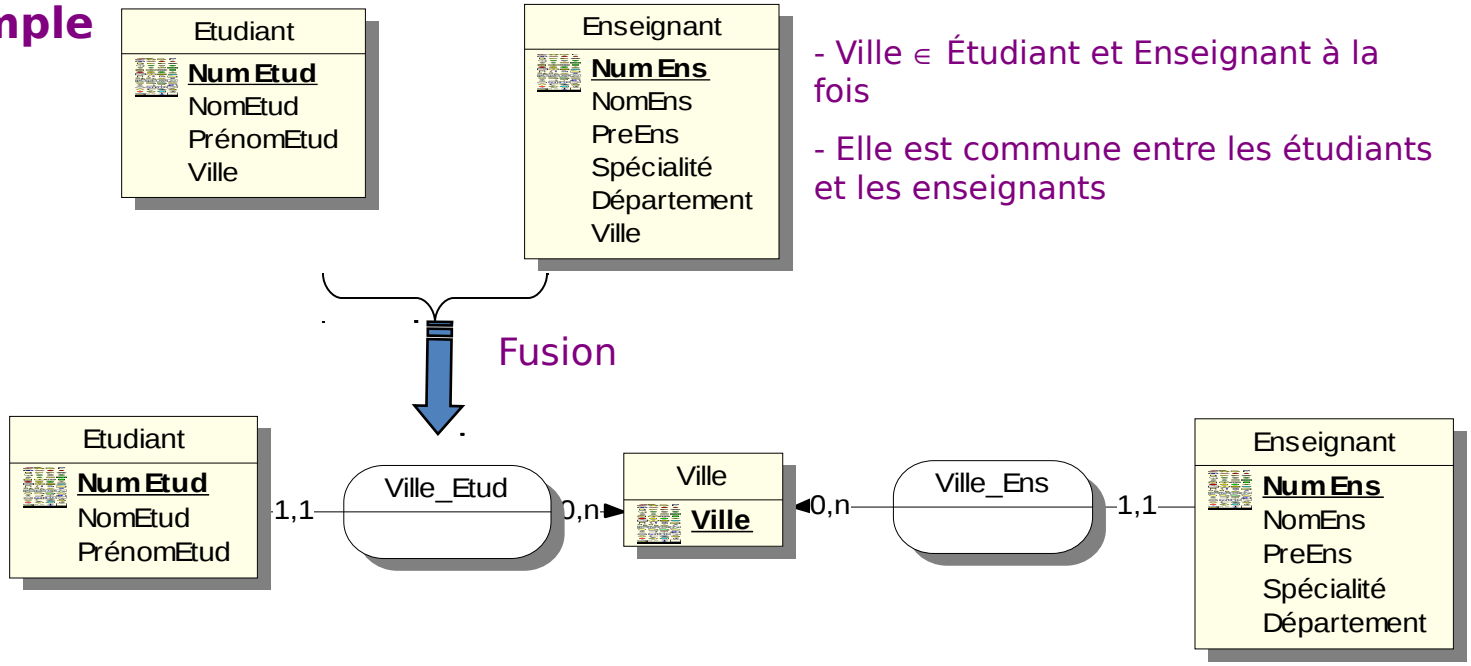




- Créer deux relations
- Cardinalités = 1,1 du côté E1 et E2

MCD – Règles de la fusion

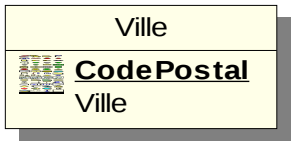
Exemple :



- Ville ∈ Étudiant et Enseignant à la fois

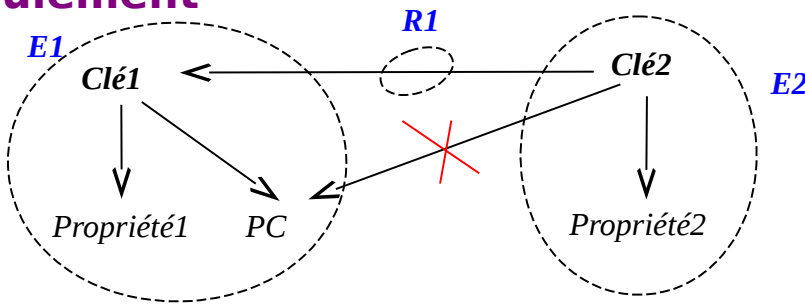
- Elle est commune entre les étudiants et les enseignants

- Le type de propriété Ville est « Alphabétique » → Ajouter une Clé Numérique : CodePostal pour une identification unique



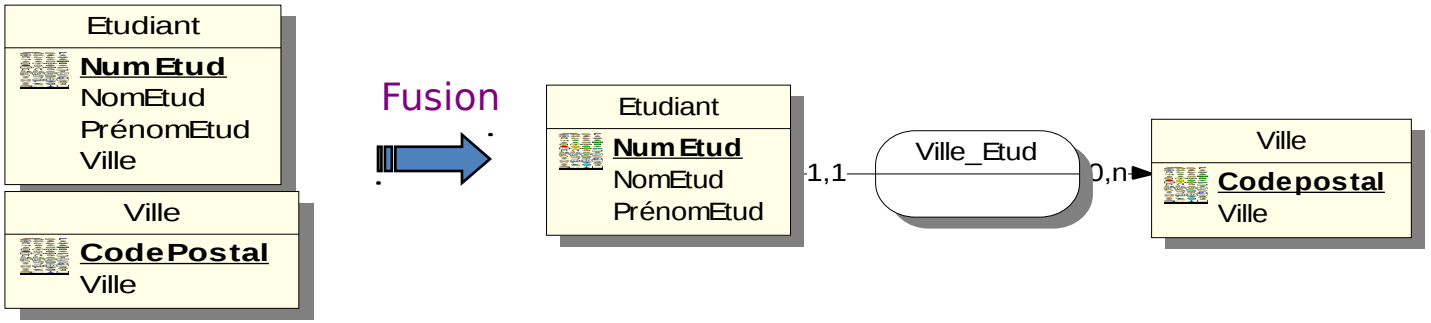
MCD – Règles de la fusion

- PC a les caractéristiques de **E1 seulement**



On ne éliminer la PC de **E2.**

Exemple :

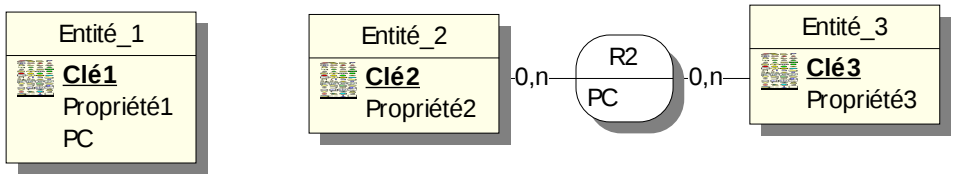


La propriété *ville* a les caractéristiques de l'entité ***Ville*** **seulement**

MCD – Règles de la fusion

- PC ∈ E1(MCD1) et PC ∈ R2(MCD2)

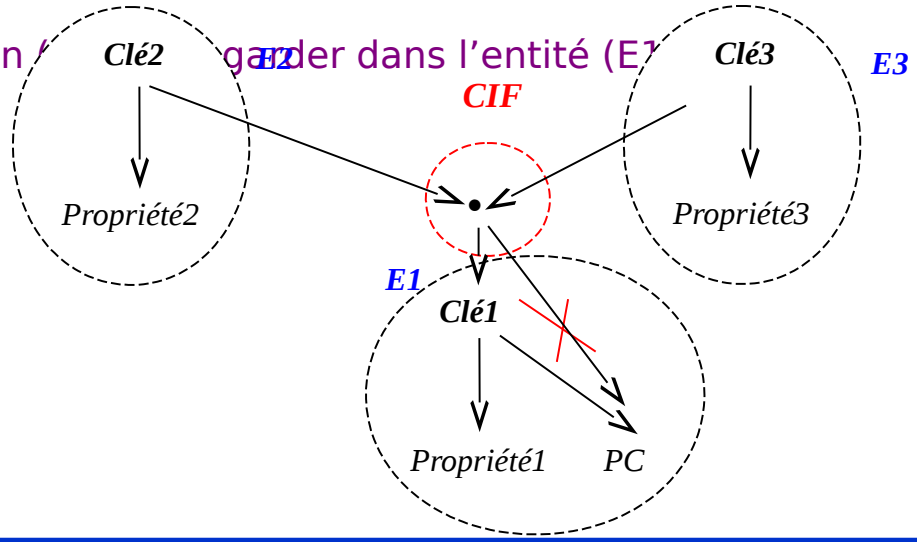
Formalisme :



• Les trois entités sont distinctes deux à deux → les trois clés sont différentes

• Les cardinalités maximales de E2 et E3 **doivent être nécessairement n**

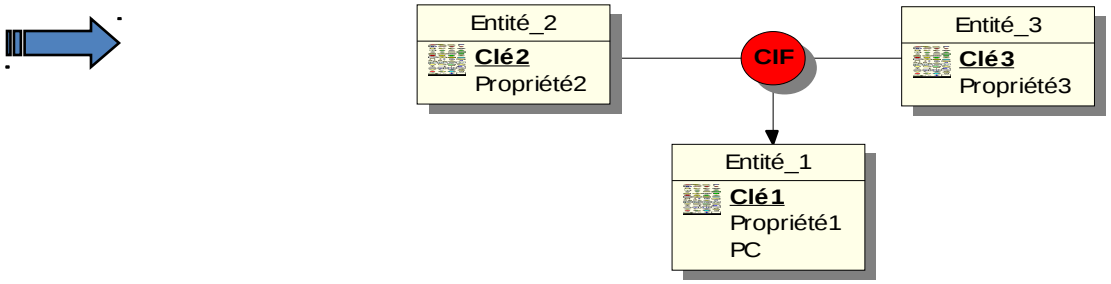
• On doit éliminer la PC de la relation / garder dans l'entité (E1



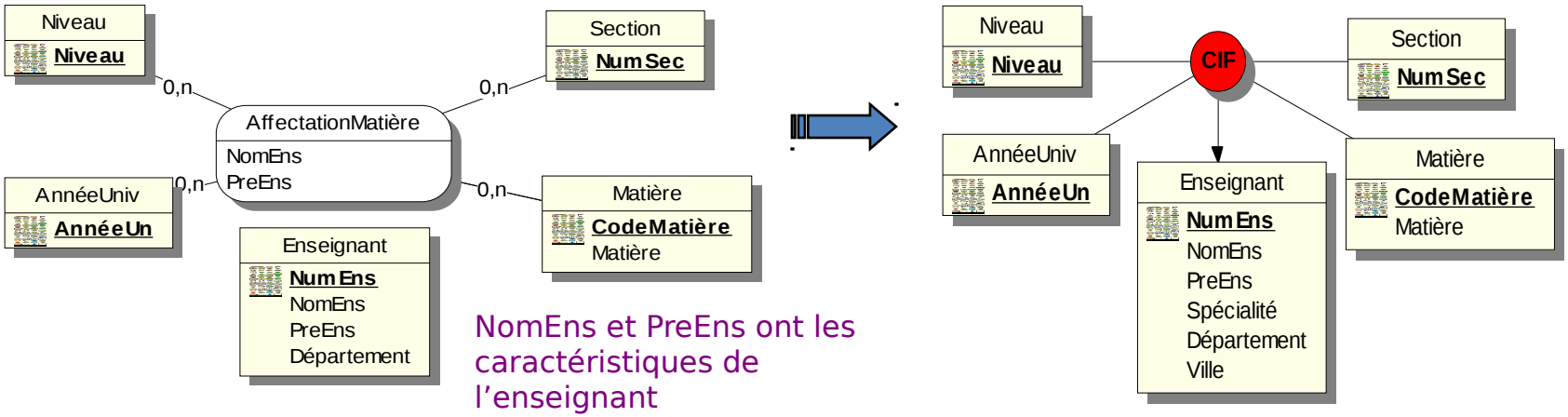
• Si la PC a les caractéristiques de E1 (**Clé2 , Clé3 → Clé1**), on obtient :

MCD – Règles de la fusion

Dans ce cas on élimine la R2 et on rajoute une CIF avec origine (E2 , E3) et cible E1



Exemple :



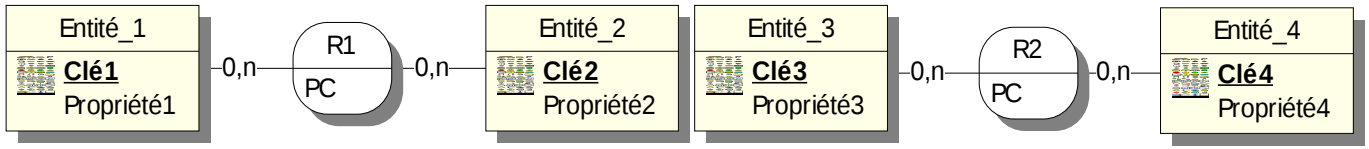
NomEns et PreEns ont les caractéristiques de l'enseignant

MCD – Règles de la fusion

- PC ∈ R1(MCD1) et PC ∈ R2(MCD2)

⇒
  - Chaque relation associe au moins **deux** relations
  - Les cardinalités maximales égales à n de chaque côté

Formalisme :



⇒ On distingue deux cas :

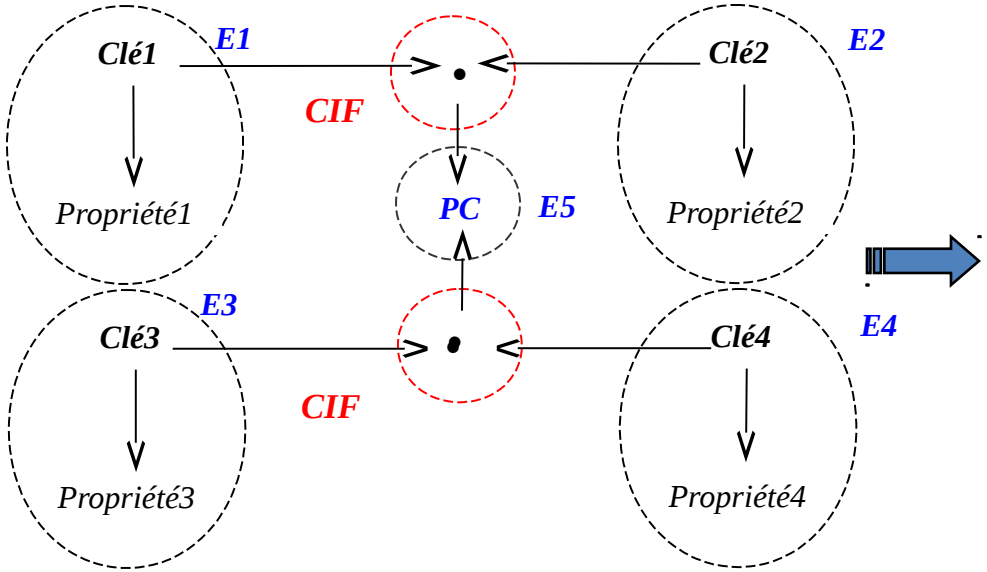
- Les entité **Ei** sont **distinctes deux à deux : ($Clé_i \neq Clé_j \ \forall \ i \neq j$)**

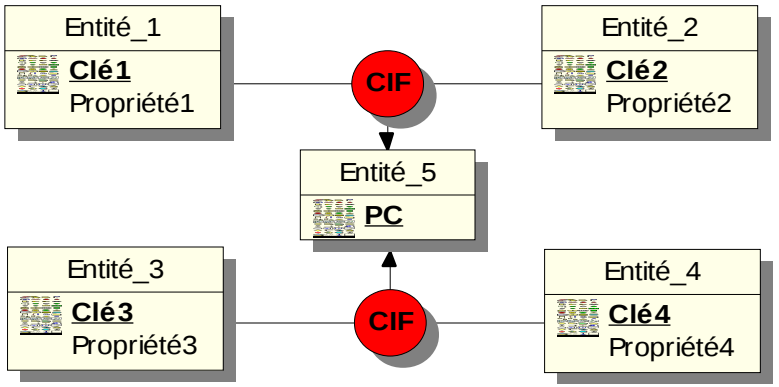
- Il existe une entité commune entre les deux relations (par exemple **E1=E3)**

MCD – Règles de la fusion

- Les entités **Ei** sont **distinctes deux à deux**

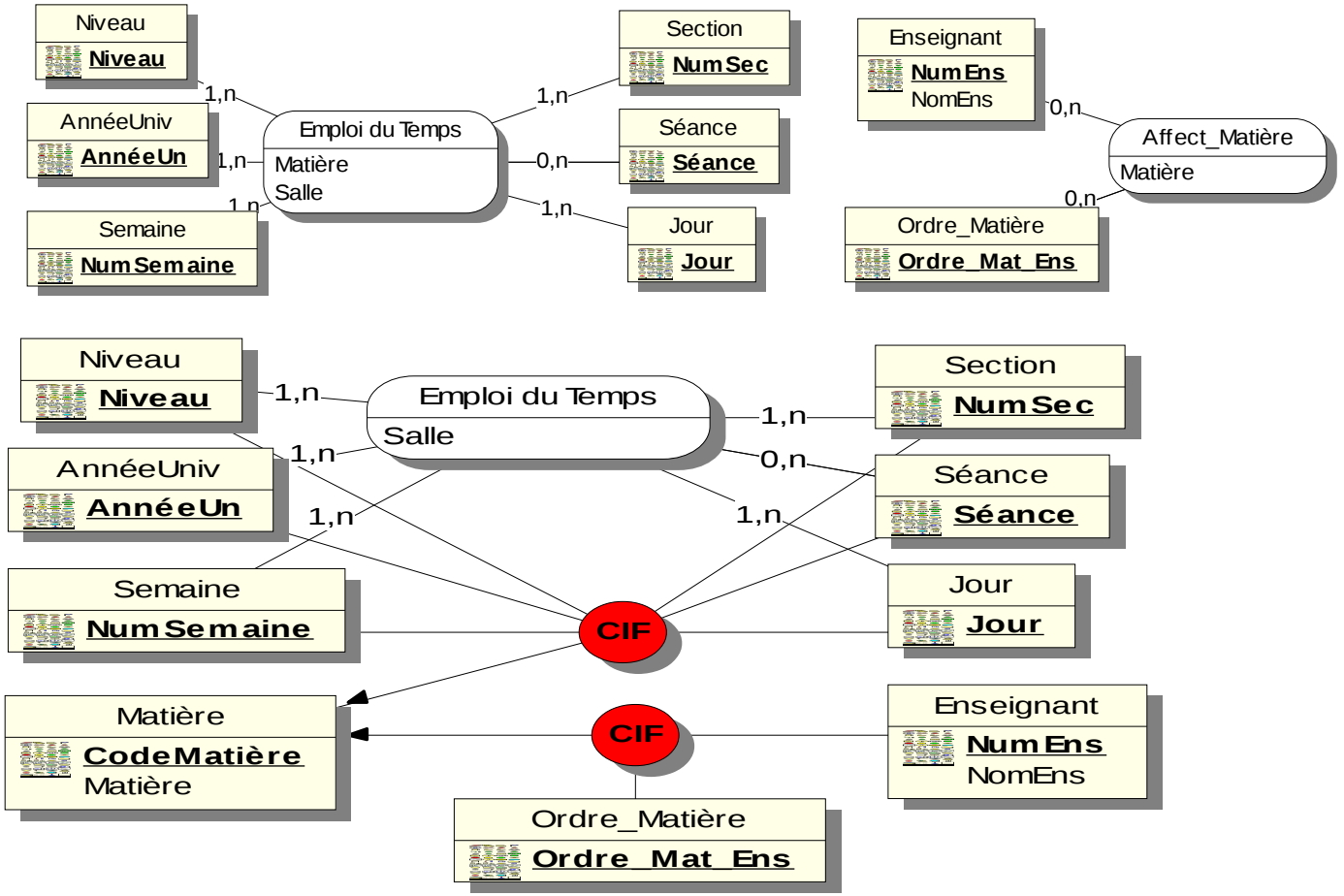
- On ne peut favoriser ni la première relation, ni la deuxième.
- On doit donc **éliminer** la **PC** des deux relations et la mètre dans une nouvelle entité ⇒ **Remplacer** les deux relations R1 et R2 par **deux CIF**.





MCD – Règles de la fusion

Exemple :



MCD – Règles de la fusion

- Il existe une entité commune entre les deux relations (par exemple **E1=E3)**

- **Clé1 = Clé3**
- **Elle doit exister une relation entre Clé2 et Clé4 ⇒**

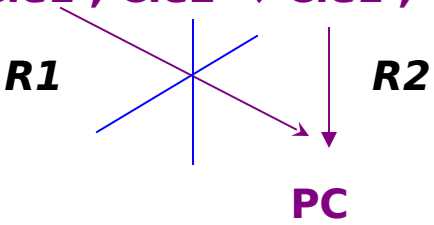
Il suffit d'étudier un seul cas : Clé2 → Clé4

Clé2 → Clé4 ⇒ Clé1 , Clé2 → Clé1 , Clé4

- **Clé2 → Clé4**

Ou

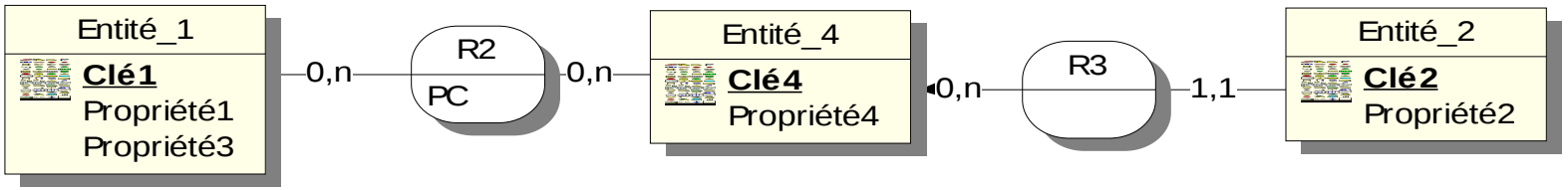
- **Clé4 → Clé2**



- **Ajouter une relation R3 entre E2 et E4 avec cardinalité 1,1 du côté E2**
- **Éliminer la relation R1 et garder la relation R2**

MCD – Règles de la fusion

Formalisme :



Dans le cas où :

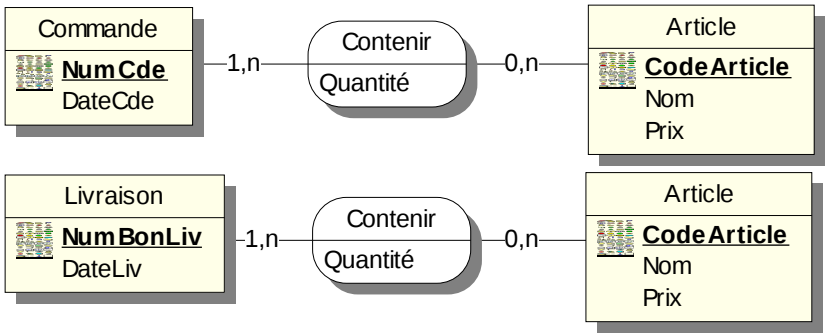
- **la relation R1 associe E1 , $(Ei_1)_{1 <i1 \leq n1}$**
- **la relation R2 associe E1 , $(Ei_2)_{1 <i2 \leq n2}$**

Si les $(Ei_1)_{1 <i1 \leq n1} \rightarrow (Ei_2)_{1 <i2 \leq n2}$

- **On élimine R1 et on garde R2**
- **On ajoute des CIF entre (Ei_1) et chaque entité du groupe (Ei_2) avec origine toute entité de (Ei_1) et cible toute entité de (Ei_2)**

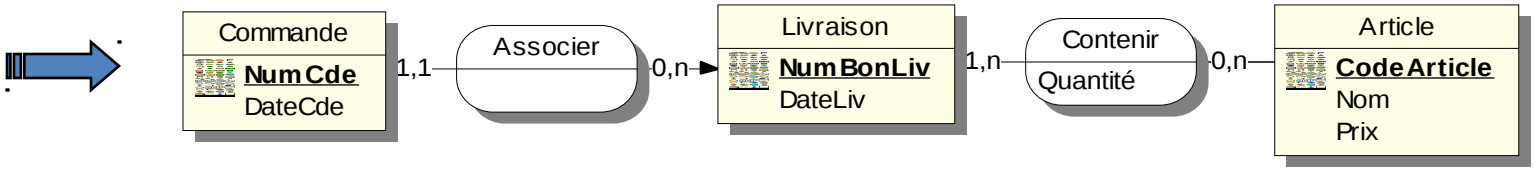
MCD – Règles de la fusion

Exemple :



- Une Commande est associée à **une** livraison ⇒ *NumCde* → *NumBonLiv*

- *Une* livraison est associée à plusieurs Commandes

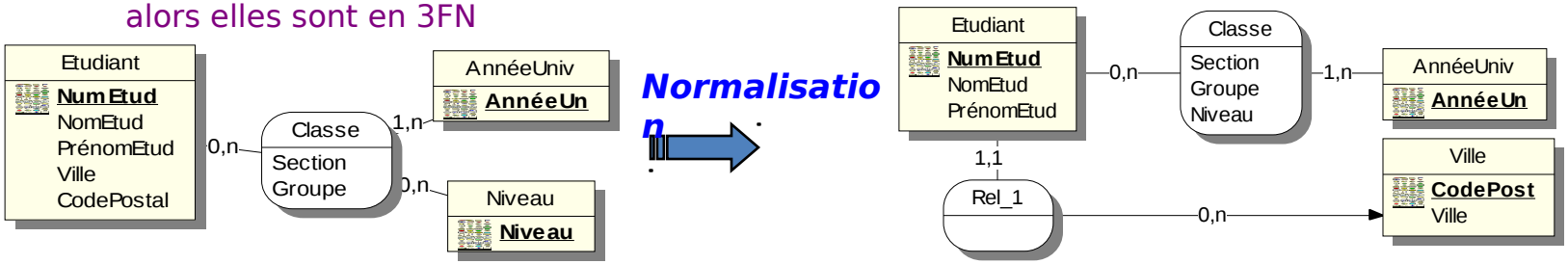


MCD – La normalisation

- Une entité ou relation est en première forme normale (respectivement deuxième et troisième FN) si toute ses propriétés sont reliées avec sa clé par une DF (respectivement DFE , DFED)

- Le MCD est en première FN (respectivement 2FN, 3FN) si toutes ses entités et ses relations sont en 1FN (respectivement 2FN, 3FN)

  - Si l'entité E1 est en 1FN alors elle est en 2FN (La clé est élémentaire)
  - Si l'entité E1 ou la relation R1 est en 2FN et contiennent seulement une seule propriété, alors elles sont en 3FN



  - L'entité Étudiant est en **2FN** mais elle n'est pas en 3FN (La **ville** ne dépend pas **directement** de **NumEtud**)
  - AnnéeUniv et Niveau sont en **3FN**
  - Classe est en **1FN** et non pas en 2FN (la *section et le groupe* dépend **seulement** de *NumEtud et AnnéeUn*)
  - ⇒ le MCD est en **1FN** : il faut rendre **Étudiant** et **Classe** en 3FN

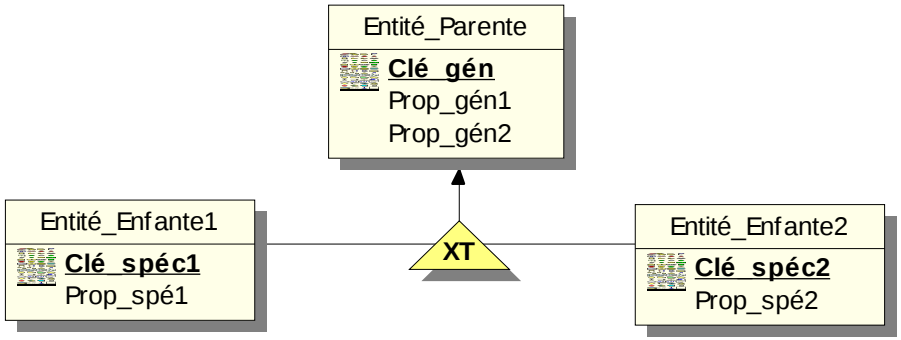
MCD – L'Héritage

- Une entité **ES** hérite d'une deuxième entité **EG** si ES est caractérisée à la fois par ses propriétés et celles d'elle même.

⇒ EG est appelée l'Entité générique ou l'entité Parente

⇒ ES est appelée l'entité spécialisée ou l'entité Enfante

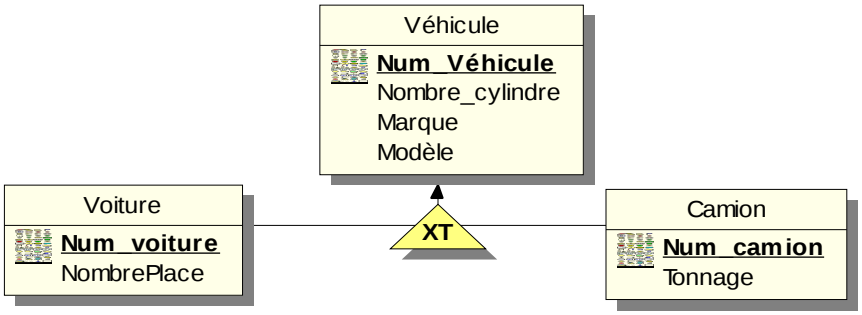
Formalisme :



- Les deux entités enfantes héritent de l'entité Parente les propriétés **prop_gén1** et **prop_gén2**
- **prop_gén1** et **prop_gén2** sont appelées les propriétés génériques
- **prop_spé1** et **prop_spé2** sont appelées les propriétés spécialisées

MCD – L'Héritage

Exemple :

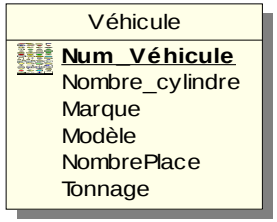


Apport de l'héritage :

Absence de l'héritage ⇒ Éliminer les entités spécialisées et garder une seule entité (l'entité générique)

⇒ s'il s'agit d'un camion, le nombre de place ne sera pas saisie et s'il s'agit d'une voiture le tonnage ne le sera pas également

⇒ Espace mémoire **perdu (non exploité)**



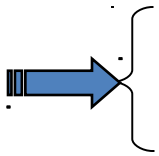
Le Modèle Conceptuel de Traitement

- **Le MCT permet de traiter la dynamique du système d'information : représente l'activité du système d'information en terme de traitements qui doivent être réalisés.**
- **Il décrit les actions à exécuter sur les données pour obtenir les résultats**

- **Acteur :** Personne morale ou physique intervenant dans le SI

Exemple :

Acteurs du SI de l'ENSAM : Étudiant, Enseignant, Service Scolarité, Service Bibliothèque, Service Achat, Fournisseur, etc.



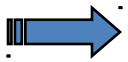
Acteurs Internes : Enseignant, Service Scolarité, Service Bibliothèque, Service Achat

Acteurs Externes : Étudiant, Fournisseur,…

- **Événement :** changement interne ou externe du SI

Exemple : Demande d'inscription, Étudiant inscrit, demande Emprunt livre, livre emprunté, etc.

MCT



Événement Interne : Étudiant inscrit, livre emprunté, etc.

Événement Externe : Demande d'inscription, demande Emprunt livre, etc.

- **Opération :** Un ensemble d'actions exécutées par le système suite à la présence d'un ou plusieurs événements

Exemple :

Opération 1 : Inscription Étudiant

Suite à une *demande d'inscription (événement_externe : entré de l'opération)*, l'étudiant sera inscrit*(événement_interne : sortie de l'opération)* s'il figure sur la liste des admis.

Opération 2 : Emprunt du livre

Suite à une *demande d'emprunt du livre (événement_externe : entrée de l'opération) par un étudiant, le livre sera emprunté (événement_interne : sortie de l'opération) si le nombre du livre pris ne dépasse pas deux*

MCT

- **Processus :** est un *ensemble d'opérations* enchaînées traitant des actions homogènes

Exemple :

Processus Absence :

Comporte l'opération saisie d'absence, contrôle d'absence, …

Processus Note :

Comporte l'opération Saisie des notes, Traitement des notes, …

- **Synchronisation :** Condition booléenne sur les événements qui déclenchent une opération.

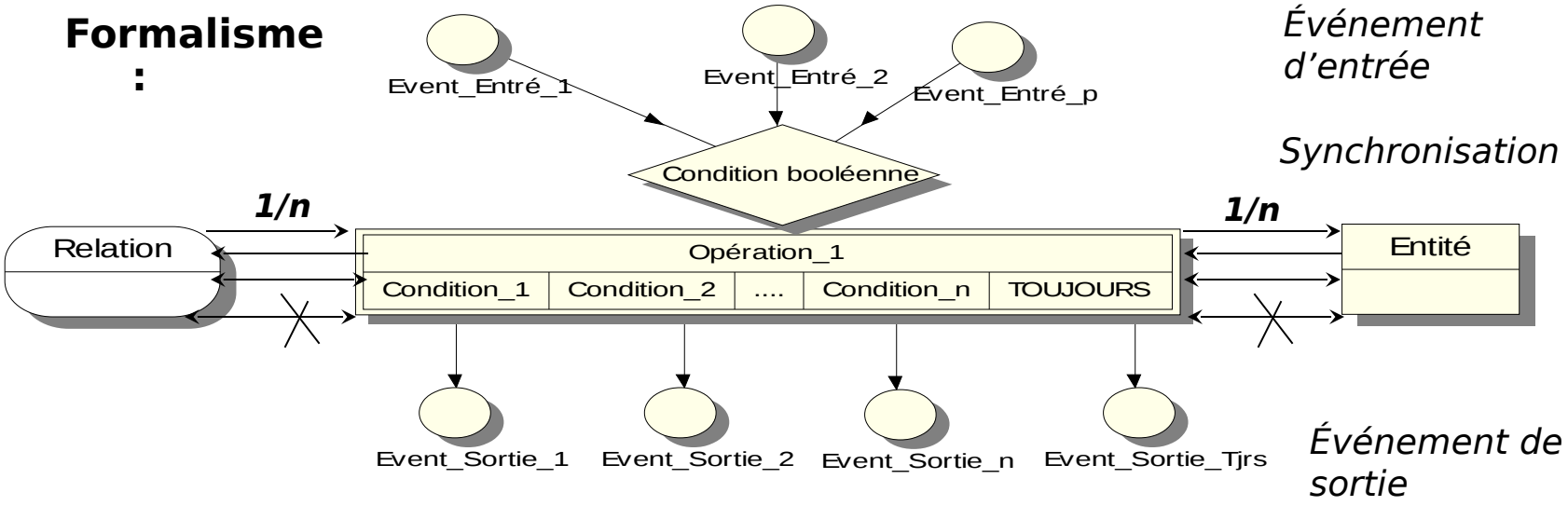
⇒ S'il s'agit d'un seul événement en entrée, aucune condition ne sera mentionnée

⇒ Sinon les événements doivent être reliés par les opérateurs logiques OU, ET, NON.

Exemple : L'opération *Emprunt_Livre* n'aura lieu que si l'étudiant est inscrit *(Événement 1 d'entrée à l'opération)* ***et*** *qu'il demande l'emprunt du livre (Événement 2 d'entrée à l'opération)*

MCT

Formalisme :

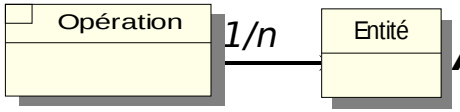


Événement d'entrée

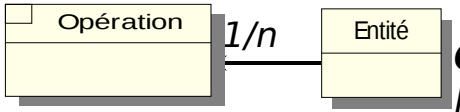
Synchronisation

Événement de sortie

-Condition_***i :*** Condition d'émission de résultats (événement de sortie)

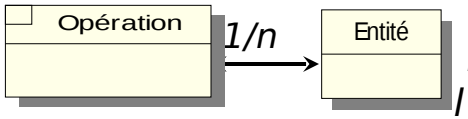


Ajout *d'un ou plusieurs occurrences à l'entité*

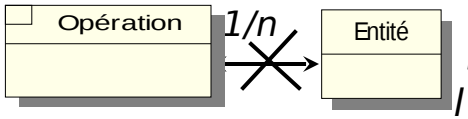


Consultation *d'un ou plusieurs occurrences de l'entité*

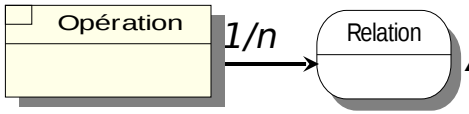
MCT



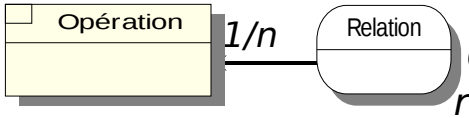
Modification *d'un ou plusieurs occurrences de l'entité*



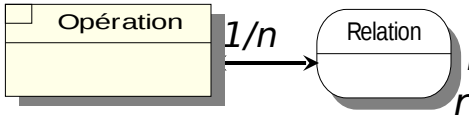
Suppression *d'un ou plusieurs occurrences de l'entité*



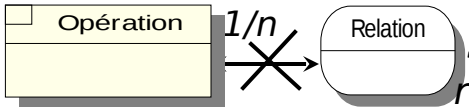
Ajout *d'un ou plusieurs occurrences à la relation*



Consultation *d'un ou plusieurs occurrences de la relation*



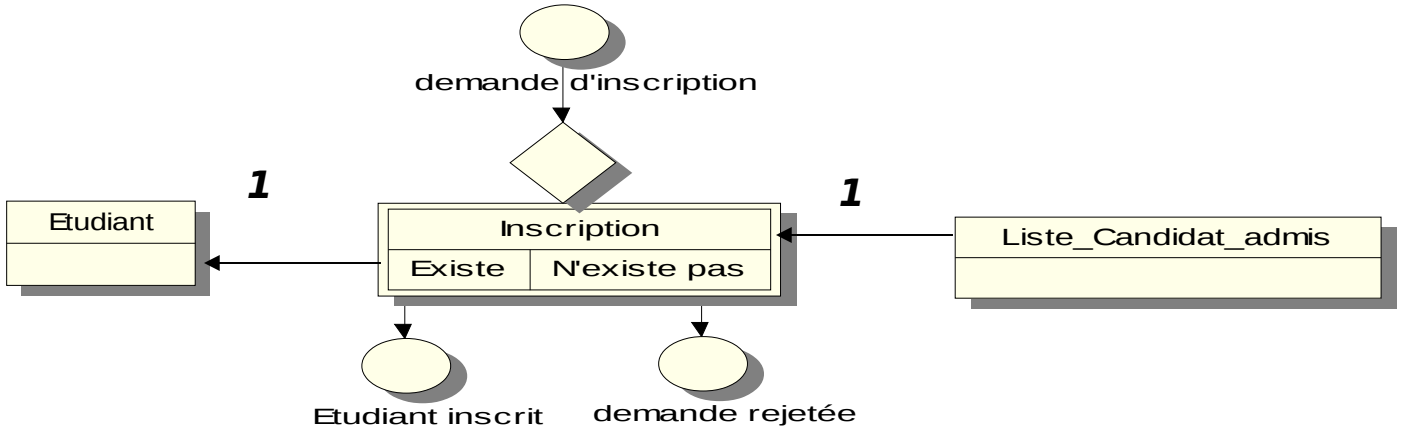
Modification *d'un ou plusieurs occurrences de la relation*



Suppression *d'un ou plusieurs occurrences de la relation*

MCT

Exemple :



Existe : Étudiant existe sur la liste des admis.

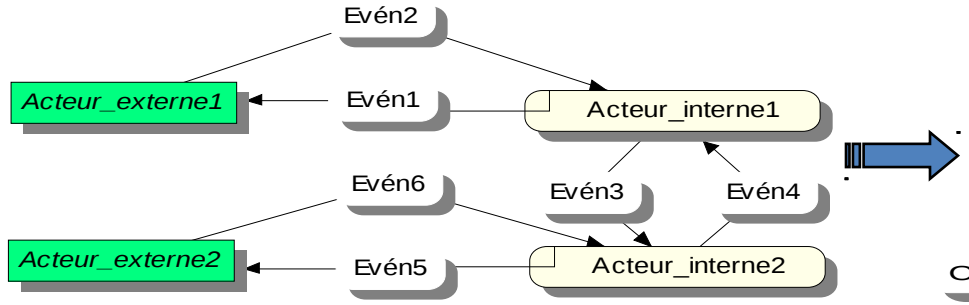
Suite à une *demande d'inscription* de l'étudiant, on vérifie (on consulte) son admission à partir de l'entité *liste des admis*. Si l'étudiant figure sur la liste, on l'inscrit.

MCT – Étapes de Construction

Étape 1 : Diagramme des flux de données

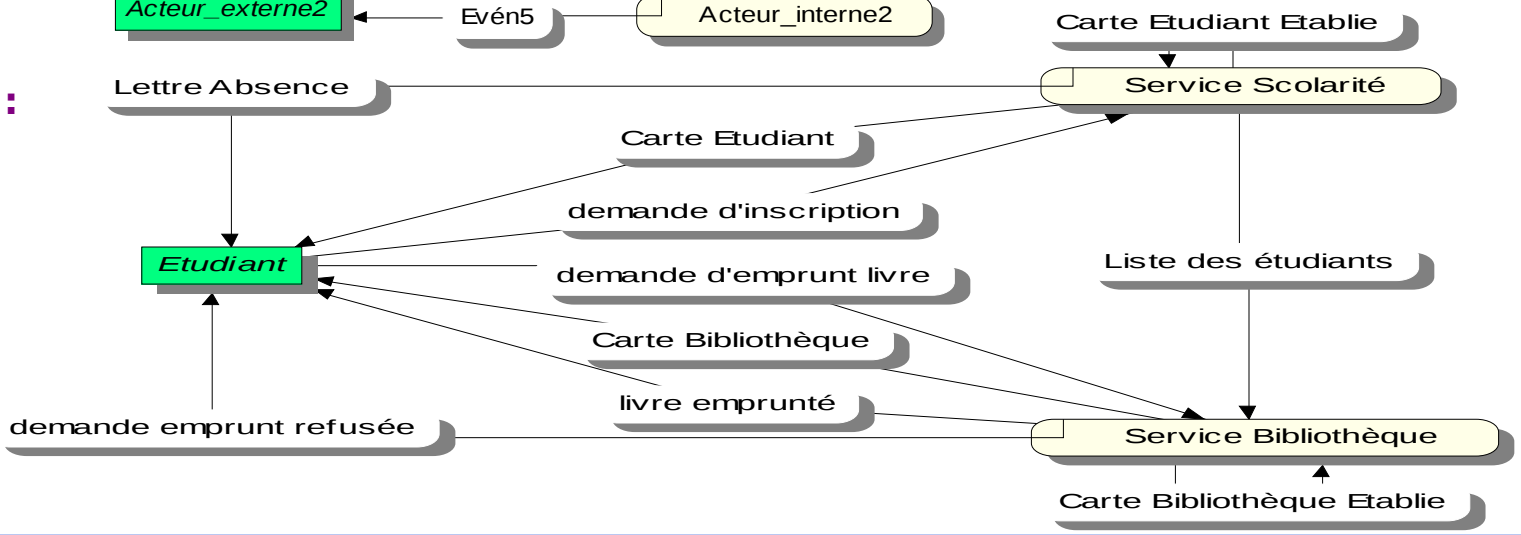
Le diagramme des flux de données représente les échanges de flux entre les acteurs

Formalisme :



. Il n'existe **aucun** flux entre les acteurs **externes**

Exemple :

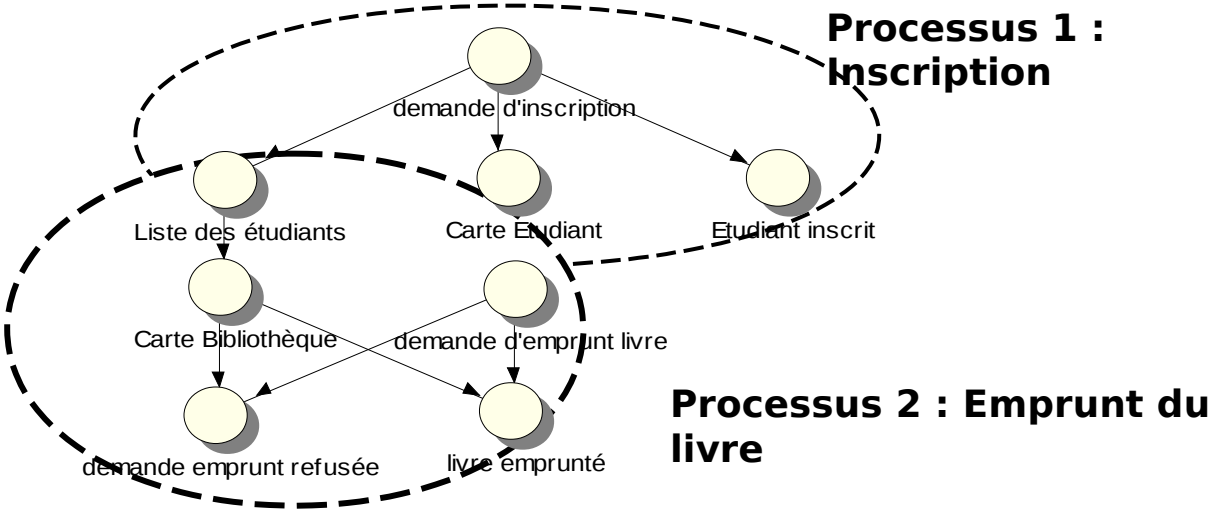


MCT – Étapes de Construction

Étape 2 : Diagramme de dépendance des documents et identification des processus

Le diagramme de dépendance représente l'enchaînement des événements dans l'axe du temps

Exemple :



Liste des étudiants est un événement commun entre les deux processus

MCT – Étapes de Construction

Étape 3 : Établissement du MCT par processus

⇒Pour chaque processus P_i, on établit le MCT associé à base des *règles organisationnelles*

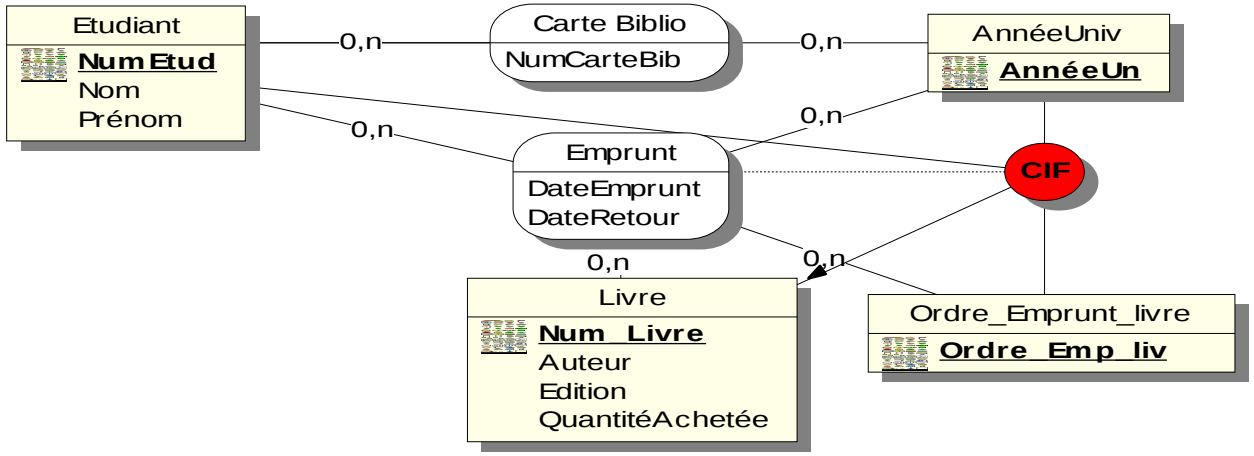
Exemple : MCT (Emprunt livre)

Règles organisationnelles :

R1 : Chaque étudiant a le droit d'emprunt au plus deux livres

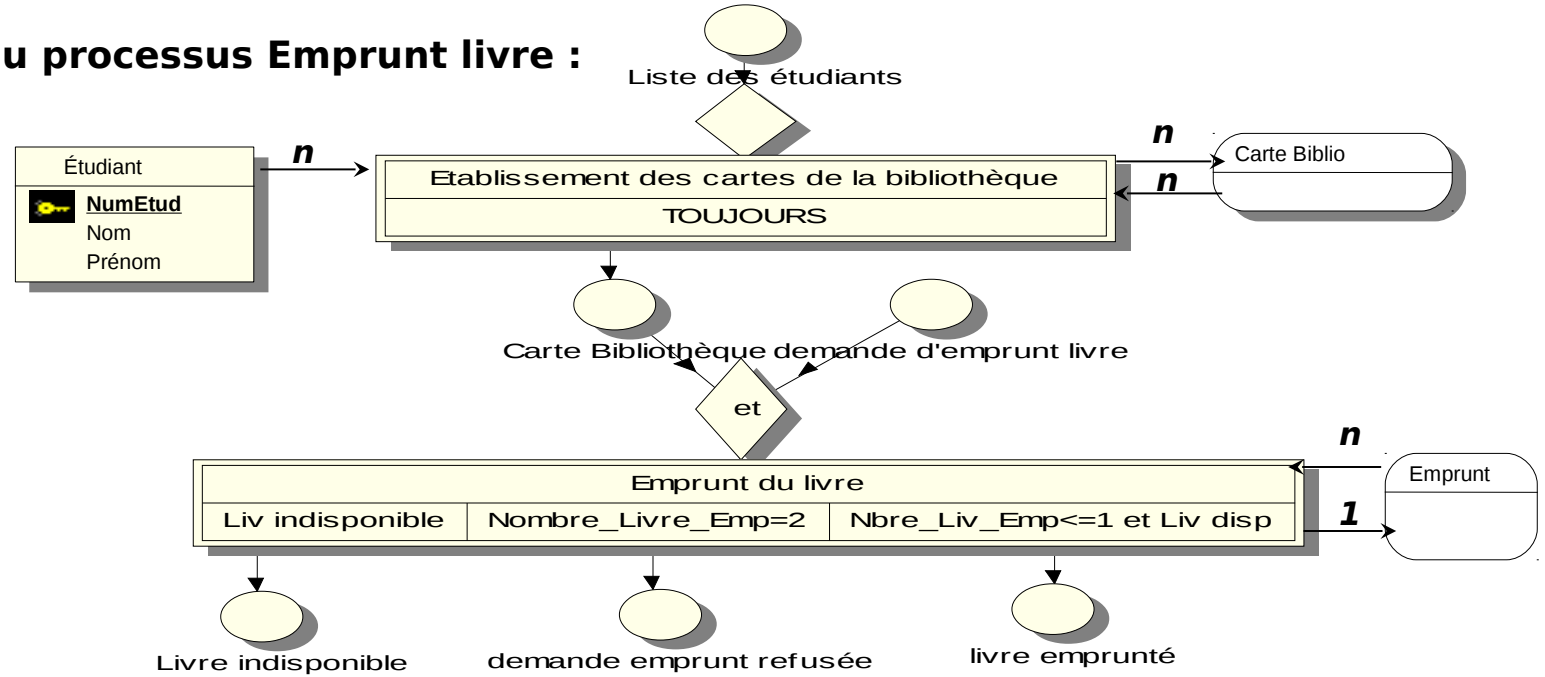
R2 : Le livre doit être rendu au plus une semaine à partir de la date d'emprunt

R3 : Si l'étudiant n'a pas rendu le livre dans le délai, il n'aura pas le droit d'emprunter pendant une semaine



MCT – Étapes de Construction

MCT du processus Emprunt livre :



- Pour imprimer les cartes biblio (contenant les informations nom, prénom, N°carte,..) on **consulte** l'entité *Étudiant* et la relation *Carte Biblio*

- Pour savoir le nombre de livres empruntés par un étudiant, on vérifie dans la relation *Emprunt* les occurrences *dont la date du retour* n'est pas remplie; si le nombre emprunté est inférieur ou égale à un et si le livre est disponible, il sera emprunté.

MCT – Validation du MCD - MCT

Validation du modèle conceptuel :

Elle Consiste à Valider le MCD et le MCT: Assurer la **cohérence** entre la conception les **données**

et celle des **traitements**.

• ***Sens MCD → MCT*** *:* s'assurer que chaque entité, relation ou propriété du MCD global est utilisée au moins par une opération d'un MCT parmi les MCTs qui existent.

⇒ Les propriétés doivent figurer dans une règle de calcul *(DateEmprunt= Date()),* ou dans une condition de déclenchement de résultats *(Livre indisponible)*

⇒ *Dans l'exemple du processus emprunt, le sens MCD – MCT est* ***validé .***

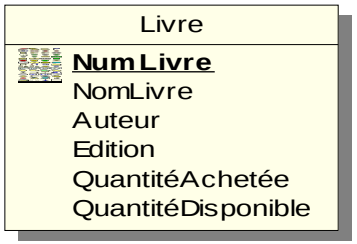
• ***Sens MCT → MCD*** *:* vérifier que chaque entité, relation ou propriété utilisée dans **chaque** MCT figure dans le MCD global.

⇒ Les propriétés à vérifier apparaissent dans la condition de synchronisation ou la condition de sortie.

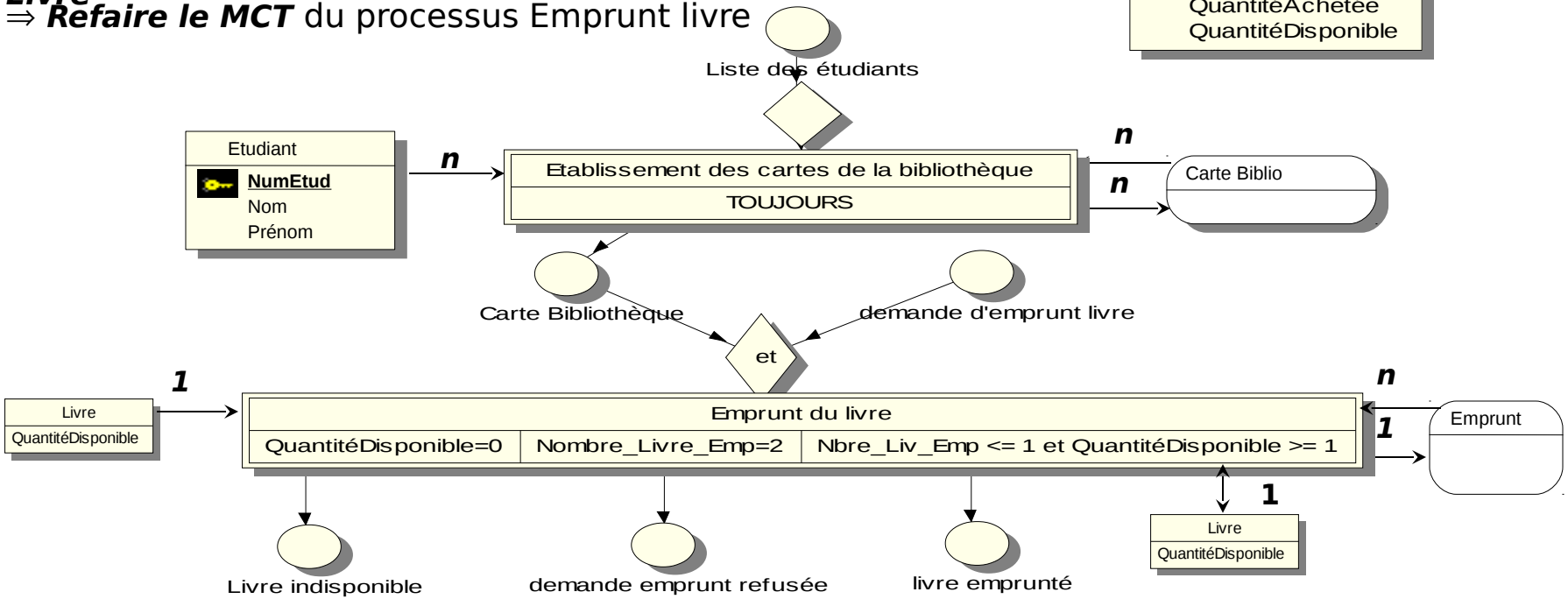
⇒ *Le sens MCT – MCD du processus emprunt est* ***non validé : la propriété***

MCT – Validation du MCD - MCT

⇒ *Ajouter la propriété* ***Quantité_Livre_disponible*** *à l'entité* ***Livre***



⇒ ***Refaire le MCT*** du processus Emprunt livre

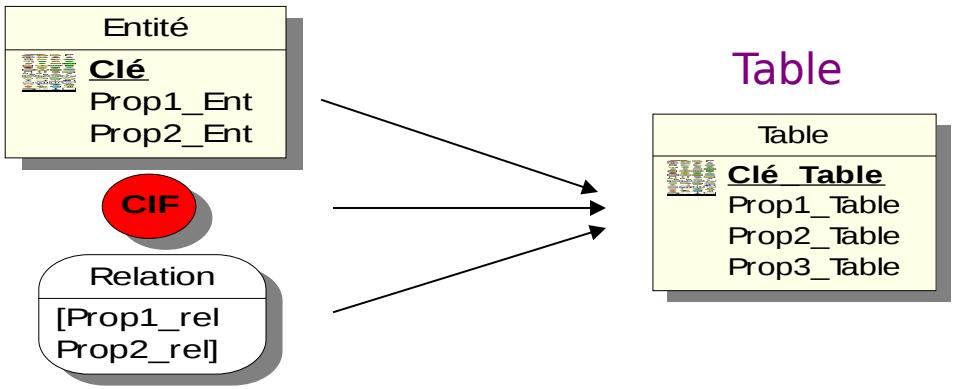


MLD

- Un MLD est un Modèle relationnel qui présente la base de données (Tables et jointures) à créer sous un système de gestion des bases de données SGBD
- Table : un ensemble d'attributs (champs) de même caractéristiques
- Le MLD est déduit à partir du MCD

MCD = {Entité , Relation , CIF}

Trois règles de passages :



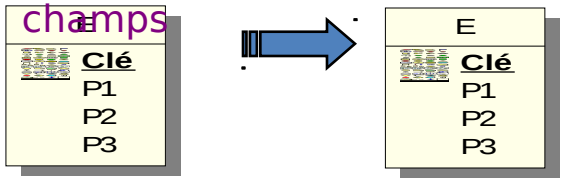
MLD – Passage du MCD sans héritage

- **Entité :**

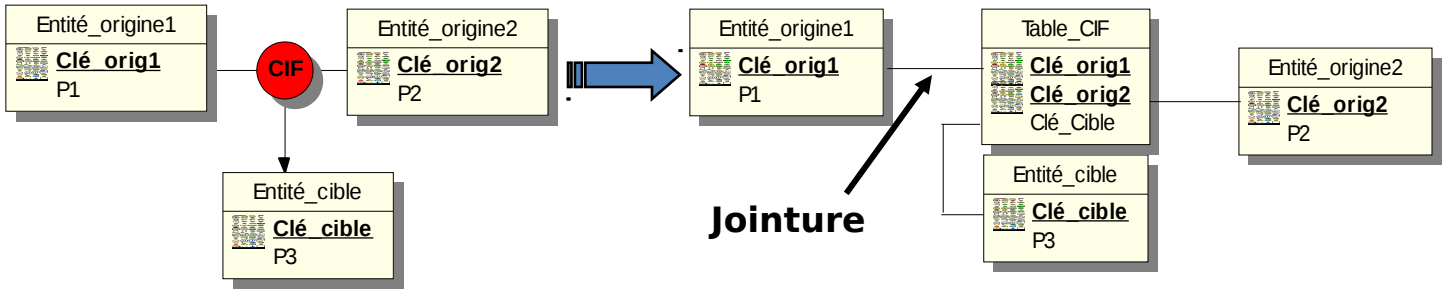
L'entité **E** devient une table **T** où ses attributs et sa clé sont respectivement les propriétés et la clé de **E**

Entité : ont le même nom et les mêmes champs

Table



- **CIF :**

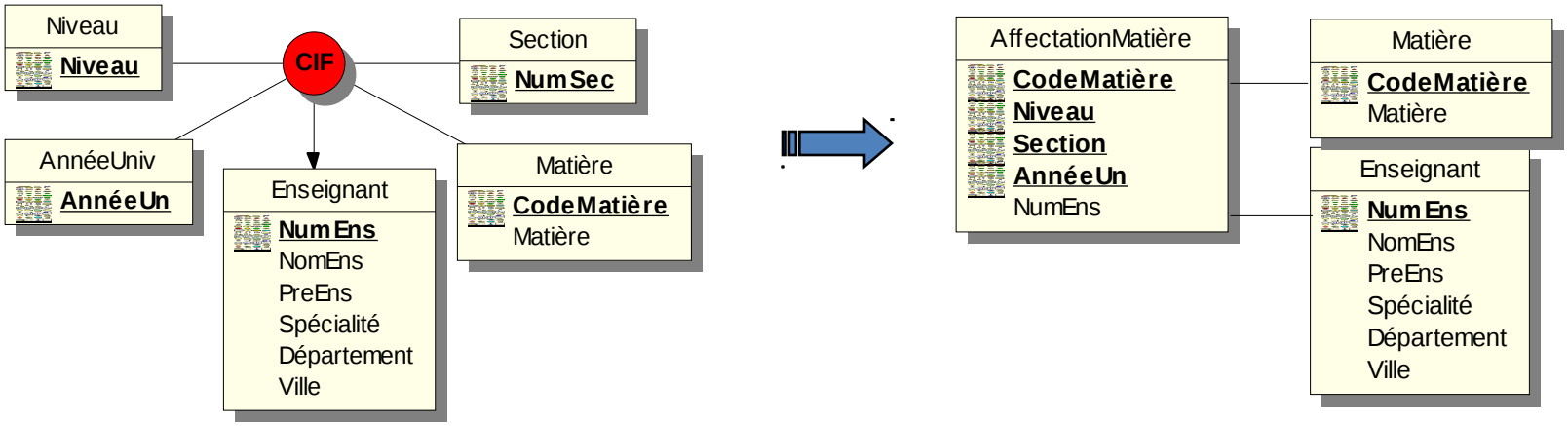


- La clé_Cible est **primaire** dans la table Entité_Cible et **étrangère** dans la table Table_CIF

- La clé de la table *Table_CIF* est composée de **Clé_orig1** et **Clé_orig2**

MLD – Passage du MCD sans héritage

- **Exemple :**



Les entités qui ne contiennent que la clé doivent être éliminées du MLD

MLD – Passage du MCD sans héritage

- **Relation :**

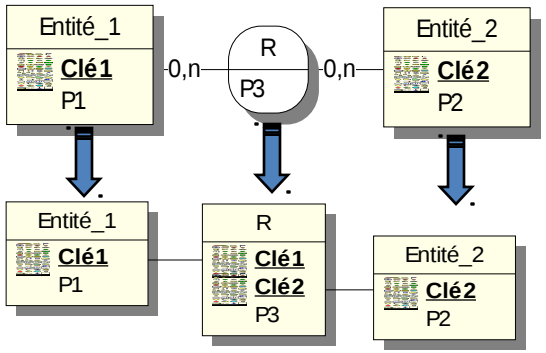
Trois cas :

- Cardinalités Maximales égales n du côté E1 **et** E2
- Cardinalités Maximales égales n d'un seul côté (E1 **ou** E2)
- Cardinalités Maximales égales 1 du côté E1 et E2 **(Exclu : E1 et E2 doivent être la même entité)**

On ne traitera que les deux premiers cas :

*** Cardinalités Maximales égales n du côté E1 et E2**

Formalisme :

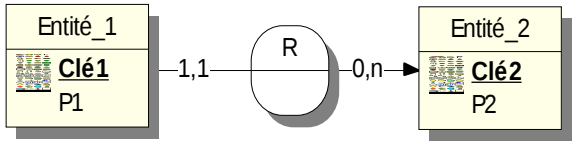


La relation R doit forcement contenir au moins une propriété

MLD – Passage du MCD sans héritage

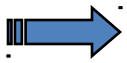
* Cardinalités Maximales égales n d'un seul côté (E2)

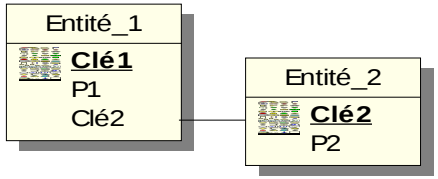
Formalisme :



La relation R doit être vide

Les cardinalités **1,1** du côté **E1** entraîne **l'élimination de la relation R** et l'introduction de la clé **étrangère Clé2** dans la table **Entité_1**

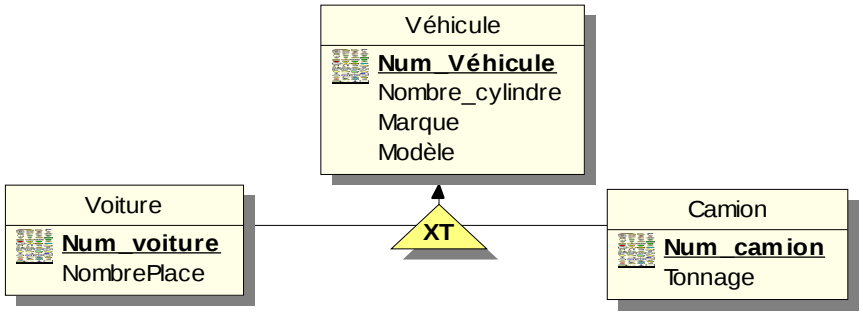




MLD – Passage du MCD avec héritage

*** Cas d'une entité**

Exemple :



Deux cas :

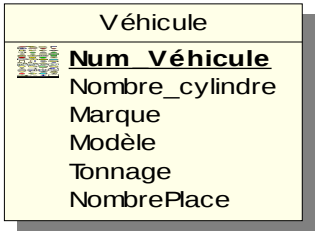
- **Favoriser la généralisation :** Éliminer les entités enfantes (spécialisées)

- **Favoriser la spécialisation :** Éliminer les entités spécialisées

MLD – Passage du MCD avec héritage

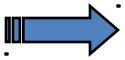
- **Favoriser la généralisation :**

⇒ **Garder** l'entité **générique et** éliminer les entités **spécialisées**



- **Favoriser la spécialisation :**

⇒ **Garder à la fois** l'entité **générique et** les entités **spécifiques**



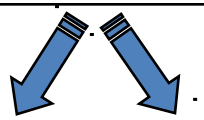
Deux cas :

- **Valeurs des clés spécialisées sont héritées de la clé générique**
- **Valeurs des clés spécialisées ne sont pas héritées de la clé générique**

MLD – Passage du MCD avec héritage

- **Valeurs des clés spécialisées sont héritées de la clé générique**

Num_Véhicule	Nombre_cylindre	Marque	Modèle
1	*4*	*Renault*	*R19*
2	*6*	*SCANIA*	*R144 LA 4X2 NA*
3	*5*	*Peugeot*	*506*
4	*6*	*SCANIA*	*R 124 LA 4X2 NA*

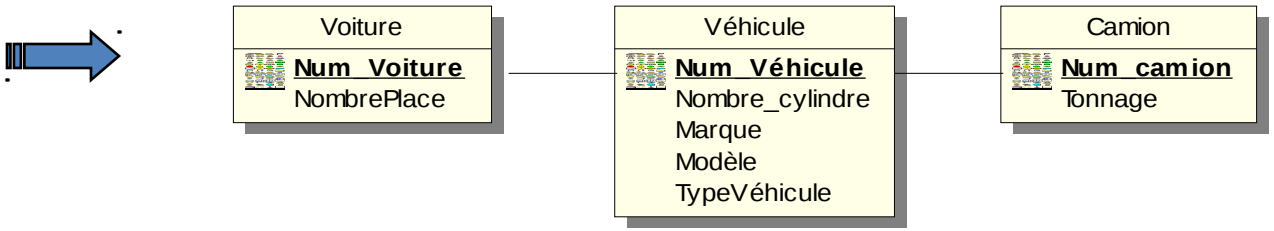


Num_Voiture	NombrePlace
1	*5*
3	*6*

Num_Camion	Tonnage (t)
2	*26*
4	*19*

MLD – Passage du MCD avec héritage

- **Valeurs des clés spécialisées sont héritées de la clé générique**

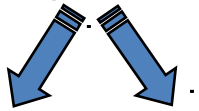


- **On relie les tables spécialisées avec la table générique via les clés**
- **On rajoute une propriété TypeVéhicule à la table générale pour accéder directement à la bonne table spécialisée : Réduire le temps de recherche des informations spécialisées**

MLD – Passage du MCD avec héritage

- **Valeurs des clés spécialisées ne sont pas héritées de la clé gé**

Num_Véhicule	Nombre_cylindre	Marque	Modèle
1	*4*	*Renault*	*R19*
2	*6*	*SCANIA*	*R144 LA 4X2 NA*
3	*5*	*Peugeot*	*506*
4	*6*	*SCANIA*	*R 124 LA 4X2 NA*



Num_Voiture	NombrePlace
1	*5*
2	*6*

Num_Camion	Tonnage (t)
1	*26*
2	*19*

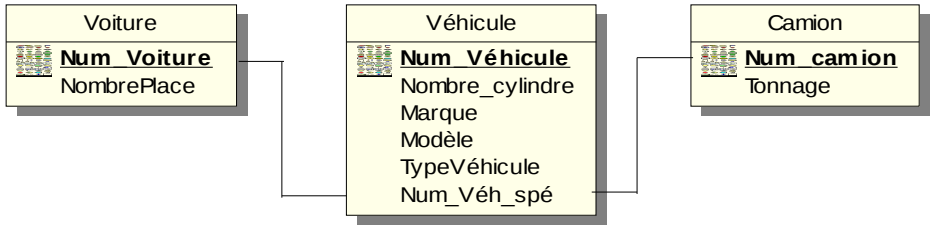
Deux cas :

- **introduire la clé spécialisée dans la table générique**
- **introduire la clé générique dans chaque table spécialisée**

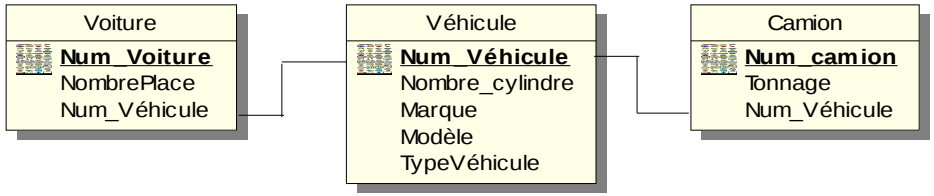
MLD – Passage du MCD avec héritage

- **introduire la clé spécialisée dans la table générique**

⇒ introduire une propriété ***typeVéhicule*** pour distinguer le type de la clé spécialisée introduite dans la table générique.



- **introduire la clé générique dans toutes les tables spécialisées**



Modèle Organisationnel des Traitements

- Le MOT décrit à partir des activités fonctionnelles du MCT, les modes d'organisation concernant les moyens mis en oeuvre, la répartition et les modalités de travail.

Répond aux questions :

- **Qui fait quoi** : Acteurs et ressources
- **Où ça se fait** : Lieu (Poste de travail)
- **Quand ?** : Temps

- **Procédure :** Un ensemble d'actions exécutées par le système suite à la présence d'un ou plusieurs événements;

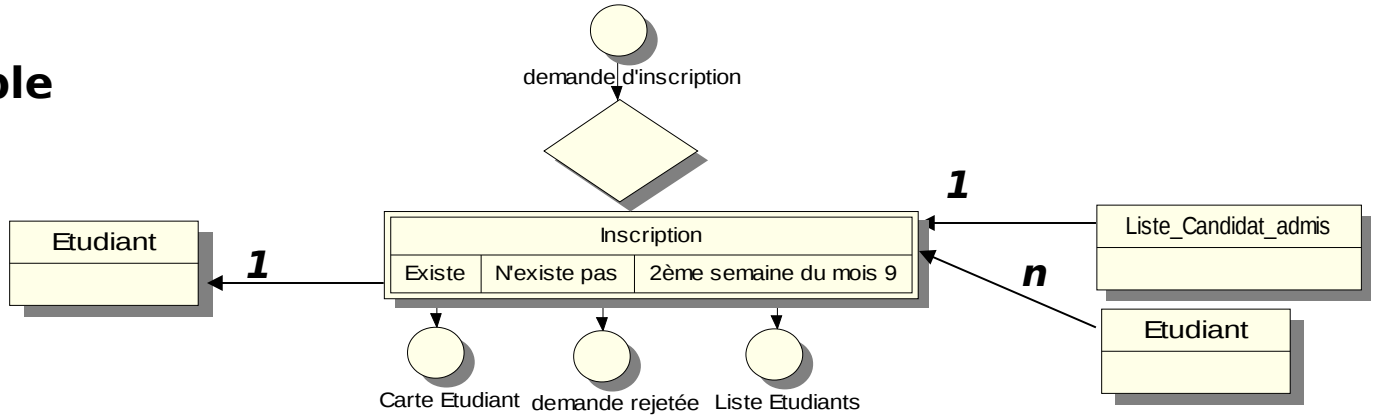
⇒ Elle peut être décomposée en plusieurs **phases** en cous où :

- les périodicités de certaines traitements sont différentes
- Une contrainte d'organisation est présentée.

- **Phase :** Ensemble de traitement dont l'enchaînement non interrompue

MOT

Exemple :



La prcédure *Inscription* comporte **deux phases** :

-Phase 1 : Saisie (inscription)

-Phase 2 : Edition de la liste des étudiants

Les deux phases ont des périodicités différentes : La 1ère phase peut durer jusqu'à 3 mois (Juillet –Septembre) parcontre la 2ème sera effectuée le 15 septembre.

MOT

- **Type d'automatisation :** détermine le type d'automatisation des traitements.

⇒ **Trois** types:

• Traitement Manuel ***(M)*** : non informatisé

• Traitement Automatique ***(A)*** ou « en Temps Réel ***TR***» : informatisé à réponse immédiate

• Traitement Conversationnel ***(C)*** ou « en Temps Différée ***TD***» : informatisé à réponse différée

La phase Saisie est du type Automatique

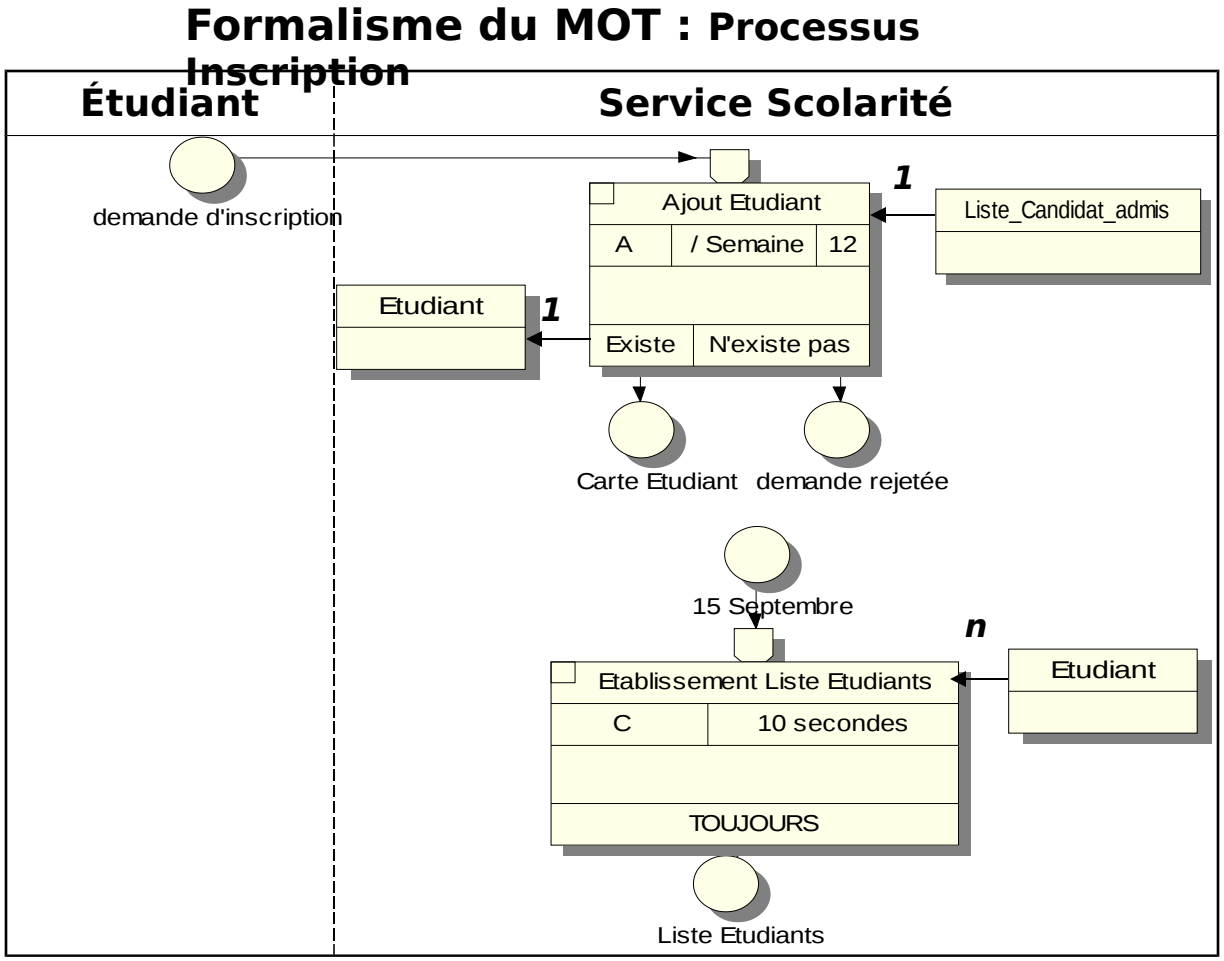
- La phase Impression est du type Conversationnel

- **Période de déroulement :** La période de temps pendant laquelle on exécute la phase

- La période de déroulement de la phase Saisie est du 3 mois

- La période de déroulement de la phase Impression est 10 secondes.

MOT


Formalisme du MOT : Processus Inscription
Étudiant
Service Scolarité
demande d'inscription
Ajout Etudiant
A
/ Semaine
12
1
Liste_Candidat_admis
Etudiant
1
Existe
N'existe pas
Carte Etudiant
demande rejetée
15 Septembre
n
Etablissement Liste Etudiants
C
10 secondes
TOUJOURS
Etudiant
Liste Etudiants

MOT

- **Description des traitements :**

⇒ Prendre en considération les **contraintes techniques** liées à la réalisation des logiciels :

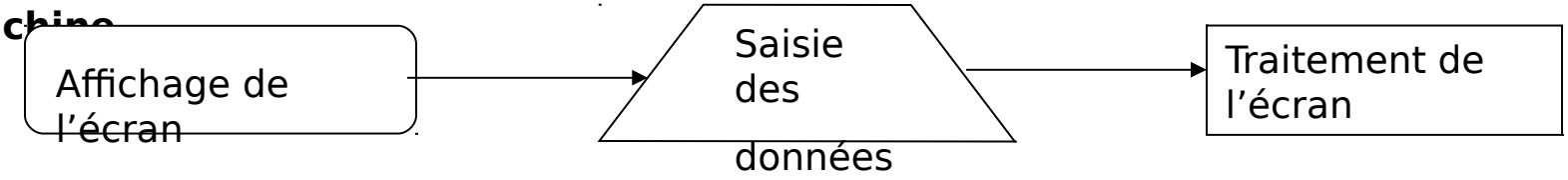
- Le type de poste de travail : caractéristique
- Les serveurs de données et de traitements : Serveur Primaire, secondaire, …
- Les outils de développement : VB, Php, ASP, Java, …
- Le SGBD : Choix (Access, MySql, SQL Server, Oracle, …) en fonction des performances désirées
- Les réseaux de l'entreprise : pas de réseaux, LAN, WAN ou Internationaux

⇒ **décrire les phases** :

- **Phase A (TR)** : Écrans et leurs enchaînement, Traitement associé à chaque écran, États possibles d'impression
- **Phase C (TD)** : États, Traitements associés.

MOT

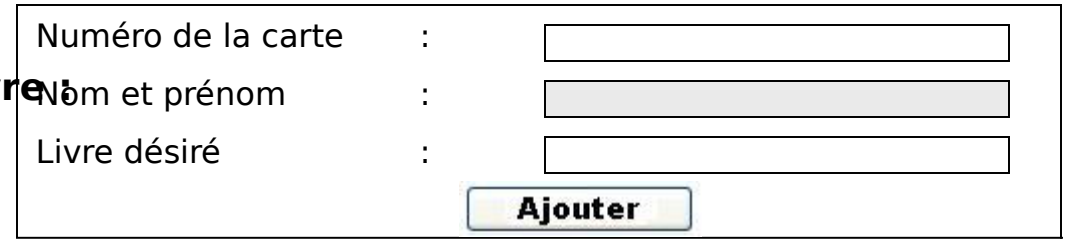
- **Phase A (TR)** : **Répartition Homme / Machine**



  - Définir les contrôles : contrôle de type, domaine, champs obligatoires, ...
  - Définir les règles du calcul
  - Édition : Impressions

Exemple :

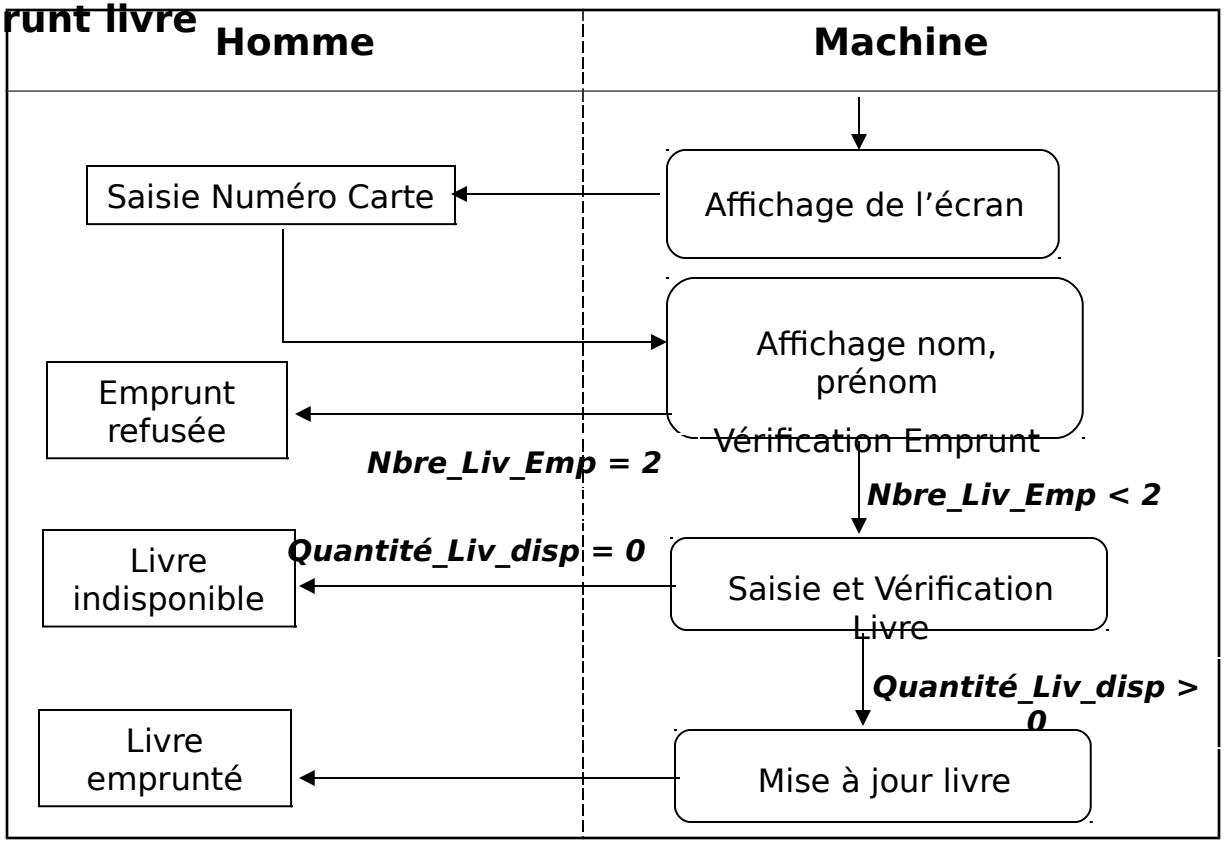
Écran de la tâche Emprunt livre :



- Le nom sera affiché suite à la sélection du Numéro de la carte.
- Suite à la saisie de la carte, on contrôle tout d'abord le droit d'emprunt d'un livre, puis en suite la disponibilité du livre dans la bibliothèque.

MOT

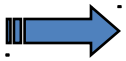
- **Phase A (TR)** : **Répartition Homme / Machine** : **Écran de la tâche Emprunt livre**



Modèle Physique de Données

⇒ La création du MLD sous un SGBD :

- **Base de données**
- **Tables**
- **Jointures**
- **Vues**
- **Index, Cluster**
- **Droit d'accès**
- **Journal de Transaction : sauvegarde les transactions effectuées sur la BD.**



Appel au **Langage de Définition des Données** de l'**SQL** *(Structed Query Language)*

On Prendra l'SQL de SQL SERVER comme exemple.

MPD - LDD

➢ Base de données :

• Création :

CREATE DATABASE *nom_BD* **[ON [** < *fichier* > **] [LOG ON** { < *fichier* >} **] [FOR LOAD]**

•[] : Optionnel

•ON : Permet de définir explicitement le fichier BD et le fichier de transaction

•LOG ON : définit le fichier de transaction

•FOR LOAD : Assure une compatibilité avec les version antérieurs du SGBD

•< fichier > : ([NAME = *nom_logique_fichier* **,] FILENAME = *'nom_phys_fichier'***

[, SIZE = ***Taille*** **] [, MAXSIZE = {** ***taille_Max*** **| UNLIMITED }])**

•Taille : en MO

•UNLIMITED : Taille illimitée.

Exemple :

CREATE DATABASE *ENSAM* **ON** (**NAME** = '*ENSAM* ', **FILENAME** = '*C:\Program Files\Microsoft SQL Server\MSSQL\data\ ENSAM.mdf*' , **SIZE** = *14*) **LOG ON** (**NAME** = '*ENSAM* ', **FILENAME** = '*C:\Program Files\Microsoft SQL Server\MSSQL\data\ ENSAM.ldf*' , **SIZE** = *2*)

MPD - LDD

➢ Tables et Jointure :

Création :

CREATE TABLE *nom_table*

({<*définition colonne*>|*nom_col* ***AS*** *expression_col_calculée* | <*contrainte_table*>**}|[{ PRIMARY KEY|**

UNIQUE }*[,...n]* **)**

- < *définition colonne* >::= **{ *nom_col*** *type_donnée* **}[DEFAULT *val_défaut*] [[NULL|NOT NULL]||**

[{ PRIMARY KEY | UNIQUE }[CLUSTERED | NONCLUSTERED]

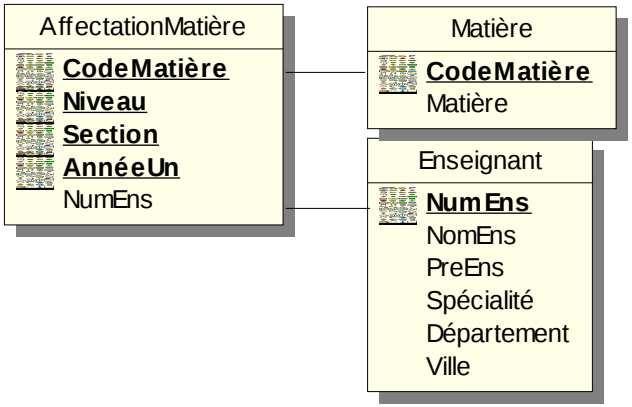
- *type_donnée* **: Numeric, int, smallint, bigint, Char(taille), Decimal, Date, …**
- *expression_col_calculée* **:** expression du calcul de la colonne calculée.
- **CLUSTERED :** Indexée ou non indexée
- < *contrainte_table* >: **[CONSTRAINT** nom_*contrainte***]**

  {[{PRIMARY KEY | UNIQUE} [CLUSTERED | NONCLUSTERED]{(*col* [ASC | DESC][,...*n*])}]

  |FOREIGN KEY [(*col* [,...*n*])] **REFERENCES** *ref_table* **[(** *ref_column* [,...*n*] **)]}**
- **Primary Key :** définit une clé **composée** s'elle est mentionnée dans < *contrainte_table* >
- **ASC | DESC :** Ordre de tri de la propriété dans l'index
- **FOREIGN KEY :** définit une clé étrangère référenciée dans la table dont elle **identifie** .

MPD - LDD

- **Exemple :**



CREATE TABLE *Matière* (*NumMatière* smallint NOT NULL **PRIMARY KEY CLUSTERED** , *Matière* char (30) NOT NULL)

CREATE TABLE *Enseignant* (*NumEns* smallint NOT NULL **PRIMARY KEY CLUSTERED** , *NomEns* char (20) NOT NULL , *PreEns* char (20) NOT NULL , *Spécialité* char (30) NOT NULL , *Département* char (5) NOT NULL ,*Ville* char (20) NULL)

CREATE TABLE *AffectationMatière* (*NumMatière* smallint NOT NULL , *AnnéeUn* char (5) NOT NULL , *Section* char (1) NOT NULL , *Niveau* smallint NOT NULL , *NumEns* smallint NOT NULL , **CONSTRAINT** clé_AffectationMatière **PRIMARY KEY CLUSTERED** *(NumMatière , AnnéeUn , Section , Niveau)* , **CONSTRAINT** *clé_étra1_AffectationMatière_Enseignant* **FOREIGN KEY** *(NumEns)* **REFERENCES** *Enseignant (NumEns)* , **CONSTRAINT** *clé_etra2_AffectationMatière_Matière* **FOREIGN KEY** *(NumMatière)* **REFERENCES** *Matière (NumMatière)))* ON [PRIMARY]

MPD - LDD

- **Index :**

⇒ Permet d'ordonner les valeurs d'une ou de plusieurs colonnes dans une **table** ou **vue** de la de base de données

⇒ Accès **rapide** à l'information spécifique dans une **table** ou **vue** de la base de données.

⇒ par exemple la colonne du NomEtudiant de la table Etudiant : si on recherche un étudiant par son nom, l'index permet d'obtenir le renseignement plus rapidement que on effectue la recherche dans toutes les lignes de la table.

Création

CREATE [UNIQUE] [CLUSTERED | NONCLUSTERED] INDEX nom_*index* ON { *nom_table* | *nom_vue* } **(** *colonne* **[ASC | DESC] [,...*n*])**

• **UNIQUE :** Crée un index unique dans lequel deux lignes ne peuvent pas avoir la même valeur d'index. Un index ordonné en clusters sur une **vue** doit être UNIQUE.

• **CLUSTERED :** Crée un objet dans lequel l'ordre physique des lignes est **identique** à l'ordre indexé des lignes.

Exemple

CREATE UNIQUE CLUSTERED INDEX *index_Matière* ON *Matière* **(***NumMatière* **ASC)**

⇔ **ALTER TABLE** *Matière* ***ADD CONSTRAINT*** *index_Matière* **PRIMARY KEY CLUSTERED (***NumMatière* **ASC)**

MPD - LDD

- **Droit Accès : Grant**

⇒ Crée une entrée dans le système de sécurité qui permet à un **utilisateur** de la base de données courante de travailler avec les données de la base ou d'exécuter des instructions SQL particulières.

• **Autorisations sur une instruction :**

GRANT { ALL | *instruction_LDD* [**,**...*n*] } **TO** *compte_utilisateur* [**,**...*n*]

• **Autorisations sur un objet :**

GRANT { ALL | *Instruction_LMD* [**,**...*n*] }
{ [**(** *colonne* [**,**...*n*] **)**] ON { *table* | *vue* } | **ON** { *table* | *vue* } [**(** *colonne* [**,**...*n*] **)** }
TO *compte_utilisateur* [**,**...*n*]

• **ALL :** Autoriser tous

• **Instruction_LDD :** Create [ALTER | DROP] Database, View, Index, Table**, ..**

TO : indique le compte utilisateur

• ***Instruction_LMD* :** *SELECT, INSERT, UPDATE, DELETE*

MPD - LDD

GRANT :

Exemple

- **GRANT ALL TO ALAMI**
- **GRANT CREATE TABLE TO Service_Informatique**
- **GRANT INSERT, UPDATE, DELETE ON LIVRE,EMPRUNT TO Service_Biblio**
- **GRANT SELECT ON LIVRE,EMPRUNT TO ETUDIANT,ENSEIGNANT**
- **GRANT INSERT, UPDATE, DELETE ON ETUDIANT,ABSENCE,NOTE,MATIERE TO Service_Scolarité**
- **GRANT ALL SELECT TO public // permettre au publique de consulter tous**

Modèle Opérationnel des Traitements

⇒ La manipulation des données sous un SGBD à l'aide du langage de manipulation de données

⇒ Création des formulaires (écrans), états d'impression, ...

- **Langage de Manipulation de Données**
  - **SELECT : Afficher les enregistrements à partir d'une table**
  - **INSERT : insérer les enregistrements dans une table**
  - **DELETE : supprimer les enregistrements d'une table**
  - **UPDATE : Modifier les enregistrements d'une table**

MOPT - LMD

- **SELECT :**

SELECT [TOP n**] [DISTINCT|ALL]** <*select_list*> **[INTO** *Nom_table_insertion***] FROM** <*liste_table*>
[WHERE < *condition*>**]**
[GROUP BY <*groupby_clause*>
[Having < *condition*>**]**
[ORDER BY *expression_tri* **[ASC | DESC]** [,...n]

- **TOP n :** Affiche les n premiers enregistrements
- **Distinct :** Affiche les enregistrements distincts
- **<select_list> :**
  - → **fonction :** count|sum|Min|Max|Avg ...
  - → *** ou Nom_table.* :** tous les champs de la table
  - → **champ[,..n]**
  - → **Nom_table.champ[,..n]**
  - → **Alias_table.champ[,..n]**
- **INTO Nom_table_insertion :** Crée une nouvelle table *Nom_table_insertion* *où seront* ***copier*** *les données*
- **<liste_table> :**
  - → **Nom_table[,...n]**
  - → **Nom_table Alias [,...n]**

MOPT - LMD

- **SELECT : Projection**

- **WHERE** < *condition*> **: limite** l'affichage des enregistrements qui **vérifient** la condition
  →**NOT <expression>**
  →**<expression1> OR|AND <expression2>**
  →**<expression> IS [NOT] NULL**
  →**<expression> [NOT] LIKE *string_expression***
  →**<expression> [NOT] BETWEEN *expression1* AND *expression2***
  →**<*expression1*> { = | < > | ! = | > | > = | ! > | < | < = | ! < } <*expression2*>**
  →**<*expression* [NOT] IN (*Requête_SQL* | *valeur[,…n]*)**
  →**<*expression { = | < > | ! = | > | > = | ! > | < | < = | ! < } (Requête_SQL)***
  → ***[NOT] EXISTS (Requête_SQL)***

- **GROUP BY** *<groupby_clause> : Indique les groupes dans lesquels les lignes de sortie doivent être placées*
  →**colonne[,…n]**
- **Having** < *condition> :* critère de recherche pour un groupement ou une fonction d'agrégation.
  → HAVING est habituellement utilisé avec la clause GROUP BY.
  →Si GROUP BY n'est pas utilisé, HAVING se comporte comme une clause WHERE.
- **ORDER BY** *expression_tri* **[ASC | DESC] :** définit le tri
  → *expression_tri* **:** colonne|Numéro_Apparition_colonne_dans_SELECT

MOPT - LMD

- **SELECT :**

E

Table Étudiant

N_INSC	NOM	PRENOM	DATE_NAISS	ADRESSE	CODE_POST
1	A	Mohammed	12/10/1986	1 Rue A1	50000
2	B	Fatima	10/01/1985	2 Bd A2	50000
3	A	Ali	01/01/1987	3 Rue A3	10000
4	C	Mohammed	28/03/1984	4 Rue A4	50000
5	D	Omar	21/03/1984	5 Rue A5	20000

Table Ville

CODE_POST	VILLE
10000	Rabat
20000	Oujda
50000	Meknès

Table Livre

NUM_LIV	LIVRE	AUTEUR	QT ACHETE	QT STOCK
1	L1	Masson	3	2
2	L2	Dunod	5	5
3	L3	Eyrolles	4	4

Table Emprunt

NUM_LIV	NUM_EMPR	N_INSC	DATE_EMPR	DATE_RETOUR
1	1	1	19/9/2004	25/09/2004
2	1	1	19/10/2004	NULL
1	2	2	15/10/2004	NULL

MOPT - LMD

- **SELECT : Exemple de requêtes**
  - **Afficher tous les étudiants ?**

→*SELECT * FROM Etudiant* ⇔*SELECT ALL * FROM Etudiant*
⇔*SELECT Etudiant.* FROM Etudiant*
⇔*SELECT* N_INSC, NOM, PRENOM, DATE_NAISS, ADRESSE, CODE_POST *FROM Etudiant*
⇔*SELECT* Etudiant.N_INSC, Etudiant.NOM, Etudiant.PRENOM, Etudiant.DATE_NAISS, Etudiant.ADRESSE, Etudiant.CODE_POST *FROM Etudiant*
⇔*SELECT* E.N_INSC, E.NOM, E.PRENOM, E.DATE_NAISS, E.ADRESSE, E.CODE_POST *FROM Etudiant E*

N_INSC	NOM	PRENOM	DATE_NAISS	ADRESSE	CODE_POST
1	A	Mohammed	12/10/1986	1 Rue A1	50000
2	B	Fatima	10/01/1985	2 Bd A2	50000
3	A	Ali	01/01/1987	3 Rue A3	10000
4	C	Mohammed	28/03/1984	4 Rue A4	50000
5	D	Omar	21/03/1984	5 Rue A5	20000

- **Afficher le premier étudiant de la liste ?**

→***SELECT TOP 1 * FROM***

N_INSC	NOM	PRENOM	DATE_NAISS	ADRESSE	CODE_POST
1	A	Mohammed	12/10/1986	1 Rue A1	50000

MOPT - LMD

- **Les fonctions d'agrégation**

Fonction	Description	Exemple	Résultat
Count(*) Count(Distinct champ)	Nombre d'enregistre.	SELECT count(*) FROM *Etudiant* SELECT count(Distinct *PRENOM*) FROM *Etudiant* SELECT count(*) FROM *Etudiant GROUP BY PRENOM* SELECT PRENOM FROM *Etudiant GROUP BY PRENOM HAVING Count(*)=2*	**5** **4** **2-1-1-1** Mohammed
Avg(Champ Numérique)	Moyenne	SELECT avg([QT ACHETE]) FROM *Livre* SELECT avg([QT ACHETE]) FROM *Livre GROUP BY LIVRE*	**4** **3-5-4**
Sum(Champ Numérique)	Somme	SELECT Sum([QT STOCK]) FROM *Livre*	**11**
Min(Champ Numérique)	Minimum	SELECT LIVRE FROM *Livre* WHERE [QT ACHETE]=(SELECT Min([QT ACHETE]) FROM *Livre*)	**L1**
Max(Champ Numérique)	Maximum	SELECT LIVRE FROM *Livre* WHERE NUM_LIV **IN** (SELECT NUM_LIV FROM *Emprunt* ***E*** WHERE (SELECT COUNT(*) FROM *Emprunt* WHERE DATE_RET=NULL AND NUM_LIV= **E.**NUM_LIV)= (SELECT **TOP 1** COUNT(*) FROM *Emprunt* WHERE DATE_RET=NULL GROUP BY NUM_LIV ORDER BY 1 DESC))	**L2-L1**

MOPT - LMD

- **Les fonctions Date**

Fonction/SGBD	ACCESS	SQL SERVER	MYSQL	ORACLE	Exemple
Date et Heure courante	Date(): *partie date*	GETDATE()	NOW()	NOW()	2004-05-01 19:52:17
Ajout d'intervalle à la date	DATEADD	DATEADD	DATE_ADD/ADD DATE/DATE_SUB	DATEADD	DATEADD('jj',2, GetDate())=dATE_ADD(NOW(), INTERVAL 2 DAY)= date_courante+2jours
Retrait en intervalle de 2 dates	DATEDIFF	DATEDIFF	DATEDIFF	DATEDIFF	DateDiff("m",'12/02/2004', '0205/2004)=3
Partie d'une date	DATEPART	DATEPART	DayOfMonth/ Year/Month	DATEPART	DatePart("jj",'12/02/2004') =12 Year('2004-12-01')=2004
Formatage d'une date (Anglais)			DATE_FORMAT		DATE_FORMAT('2004-05-10', '%W %d %M %Y')= Monday 10 May 2004 DATE_FORMAT('2004-05-10', '%d/%m/ %Y')=10/05/2004

- DATEADD*(type,nombre, date)* - DATE_ADD/ADDDATE *(date,INTERVAL nombre type)*

→Type : Année (aa/yy/aaaa/yyyy) ,Mois (m),jour(jj/dd/Day),heure(hh),minute(mm),seconde(ss)

- DATE_SUB*(date,INTERVAL* ***nombre*** *type)* = DATE_ADD*(date,INTERVAL* ***-nombre*** *type)*

MOPT - LMD

- **SELECT : Les fonctions Date**

- DATEDIFF :
  - *ACCESS, ORACLE, SQLSERVER : (type,date_début,date_fin)*
  - *MYSQL : (date_début,date_fin) et la différence est calculée en jour*
- DATEPART *(type , date)*
- *DATE_FORMAT(date,format) .*

Format	Description
%Y	Année 4 chiffres
%y	Année 2 chiffres
%M	Mois : January .. December
%c	Numéro du Mois : 1 .. 12
%W	Jour : Sunday .. Saturday
%j	Numéro du jour du mois : 1 .. 366
%d (%e)	Numéro du jour du mois : 1 .. 31
%h (%H)	Heure : 1 .. 12 (00 .. 23)
%i	Minute : 0 .. 59
%s	Seconde : 0 .. 59

MOPT - LMD

- **SELECT : Les fonctions chaîne de caractères**

Fonction/SGBD	ACCESS	SQL SERVER	MYSQL	ORACLE	Exemple
Concaténation	+	+	CONCAT	+	‘a’+’b’ = ‘ab’
Code Ascii d’un symbole	ASCII	ASCII	ASCII	ASCII	ASCII(‘A’)=65
Symbole associé à un code Ascii	CHAR	CHAR	CHAR	CHAR	CHAR(65)=‘A’
Longueur d’une chaîne	LENGTH	LENGTH	LENGTH	LENGTH	LENGTH(‘ab’)=2
Renversement d’une chaîne		REVERSE	REVERSE	REVERSE	REVERSE(‘ab’)=‘ba’
Position d’une sous chaîne		CHARINDEX PATINDEX	INSTR LOCATE	INSTR	CHARINDEX(‘ab’,’cgabh’)=3 INSTR(’cgabh’,‘ab’)=3 PATINDEX (‘**%**ab**%**’,’cabg’)=2 LOCATE(‘ab’,’cabg’)=2
Remplacement d’une chaîne		REPLACE	REPLACE	REPLACE	REPLACE('ahghp','h','rr')=‘abrr grrp’
Sous chaîne gauche	LEFT	LEFT	LEFT	LEFT	LEFT(‘abcdef’,3)=‘abc’
Sous chaîne droite	RIGHT	RIGHT	RIGHT	RIGHT	RIGHT((‘abcdef’,3)=‘def’

-SUBSTRING(Chaîne , position_début,Nombre_caractère) = sous chaîne

-REPLACE (Chaîne1, Chaîne2, Chaîne3) : Remplace dans la *chaîne1* la *chaîne2* par la *chaîne3*

MOPT - LMD

- **SELECT : Les fonctions chaîne de caractères**

Fonction/SGBD	ACCESS	SQL SERVER	MYSQL	ORACLE	Exemple
Extraction d'une sous chaîne	SUBSTRING	SUBSTRING	SUBSTRING MID	SUBSTR	SUBSTRING('abcdef',2,3)='bcd'
Éliminer les espaces à gauche	LTRIM	LTRIM	LTRIM	LTRIM	LTRIM(' ab')='ab'
Éliminer les espaces à droite	RTRIM	RTRIM	RTRIM	RTRIM	RTRIM(' ab ')=' ab'
Mettre une chaîne en Majuscule	UPPER	UPPER	UPPER	UPPER	UPPER('abC')='ABC'
Mettre une chaîne en Minuscule	LOWER	LOWER	LOWER	LOWER	LOWER('ABc')='abc'
La sous chaîne gauche	LEFT	LEFT	LEFT	LEFT	LEFT('abcd',2)='ab'
La sous chaîne droite	RIGHT	RIGHT	RIGHT	RIGHT	RIGHT('abcd',2)='cd'
Générer les espaces	SPACE	SPACE	SPACE	SPACE	SPACE(*3*)=' '
Répéter chaîne		REPLICATE	REPEAT		REPLICATE('ab',2)='abab'

MOPT - LMD

- **SELECT – INTO : Exemple**
  - **Créer la table NOM_PRE_Etud qui contient les deux champs NOM et PRENOM par ordre alphabét**

    →*SELECT NOM, PRENOM INTO* **NOM_PRE_Etud** *FROM Etudiant*

Table NOM_PRE_Etud

NOM	PRENOM
A	Mohammed
B	Fatima
A	Ali
C	Mohammed
D	Omar

- **SELECT – Produit Cartésien et Restriction (WHERE) :**
  - **Afficher les étudiants qui habitent à Meknès**

→*SELECT* NOM, PRENOM *FROM Etudiant, Ville* WHERE VILLE='Meknès'

NOM	PRENOM
A	Mohammed
B	Fatima
A	Ali
C	Mohammed
D	Omar

➡ Faux

MOPT - LMD

- **SELECT – Jointure :**
  - **Afficher les étudiants qui habitent à Meknès**

- ***Requêtes non imbriquées***

→*SELECT* NOM, PRENOM *FROM Etudiant, Ville* WHERE Etudiant.CODE_POST=Ville.CODE_POST **AND** VILLE='Meknès'

⇔*SELECT* NOM, PRENOM *FROM Etudiant **E**, Ville **V*** WHERE **E.**CODE_POST=**V.**CODE_POST **AND** VILLE='Meknès'

- ***Requêtes imbriquées***

⇔*SELECT* NOM, PRENOM *FROM Etudiant* WHERE CODE_POST=(*SELECT* CODE_POST *FROM Ville* WHERE VILLE='Meknès')

NOM	PRENOM
A	Mohammed
B	Fatima
C	Mohammed

- **SELECT – Union et Intersection (OR|AND):**
  - **Afficher les étudiants qui habitent à Meknès et Oujda**

- ***Requêtes non imbriquées***

→*SELECT* NOM, PRENOM *FROM Etudiant **E**, Ville **V*** WHERE **E.**CODE_POST=**V.**CODE_POST **AND** VILLE **IN** ('Meknès',' Oujda')

⇔*SELECT* NOM, PRENOM *FROM Etudiant **E**, Ville **V*** WHERE **E.**CODE_POST=**V.**CODE_POST **AND (**VILLE ='Meknès' **OR** VILLE ='Oujda')

MOPT - LMD

- **SELECT – Union et Intersection (OR| AND):**
- ***Requêtes imbriquées***

→*SELECT* NOM, PRENOM *FROM Etudiant* WHERE CODE_POST **IN** (*SELECT* CODE_POST FROM *Ville* WHERE VILLE **IN** (‘Meknès’,’ Oujda’))

⇔ *SELECT* NOM, PRENOM *FROM Etudiant* WHERE CODE_POST **IN** (*SELECT* CODE_POST FROM *Ville* WHERE **(**VILLE =‘Meknès’) **OR (**VILLE =’ Oujda’)

NOM	PRENOM
A	Mohammed
B	Fatima
C	Mohammed
D	Omar

- **SELECT – Différence (NOT IN):**
  - **Afficher les étudiants qui n’habitent pas à Meknès et Oujda**

→*SELECT* NOM, PRENOM *FROM Etudiant* WHERE CODE_POST **NOT IN** (*SELECT* CODE_POST FROM *Ville* WHERE VILLE **IN** (‘Meknès’,’ Oujda’))

NOM	PRENOM
A	Ali

MOPT - LMD

- **SELECT – Restriction (IS NULL) :**
  - **Afficher les livres non rendus**

- ***Requêtes non imbriquées***

→*SELECT* NOM_LIVRE *FROM Livre* ***L*** *, Emprunt* ***E*** WHERE **L.**NUM_LIV=**E.**NUM_LIV **AND** NUM_LIV **IS NULL**

- ***Requêtes non imbriquées***

→ *SELECT* NOM_LIVRE *FROM Livre* ***L*** WHERE *(SELECT* NOM_LIVRE *FROM Emprunt* WHERE NUM_LIV=**L.**NUM_LIV) **IS NULL**

NOM_LIV
L1
L2

- **SELECT – Restriction (NOT LIKE) :**
  - **Afficher les étudiants dont le prénom commence par l'aplphabet 'M' triés par ordre alphabétiq**

→*SELECT* NOM, PRENOM *FROM Etudiant* WHERE PRENOM **LIKE** 'M*' **ORDER BY 1 , 2**

NOM	PRENOM
A	Mohammed
C	Mohammed

MOPT - LMD

- **SELECT – Restriction (BETWEEN) :**
  - **Afficher les livres rendus entre le mois 9 et 10 de l'année en cours**

- ***Requêtes non imbriquées***

→*SELECT* NOM_LIVRE *FROM Livre* ***L*** , *Emprunt* ***E*** WHERE **L.**NUM_LIV=**E.**NUM_LIV **AND** DATEPART**(** *'m'*,DATE_RET) **BETWEEN** 9 **AND** 10) **AND** DATEPART**(** *'yyyy'*, DATE_RET)= *GETDATE()*

⇔ *SELECT* NOM_LIVRE *FROM Livre* ***L*** , *Emprunt* ***E*** WHERE **L.**NUM_LIV=**E.**NUM_LIV **AND** **(**DATEPART**(** *'m'*,DATE_RET) **>=**9) **AND** **(**DATEPART**(** *'m'*,DATE_RET) **<=**10) **AND** DATEPART**(** *'yyyy'*, DATE_RET)= *GETDATE()* .

NOM_LIV
L1

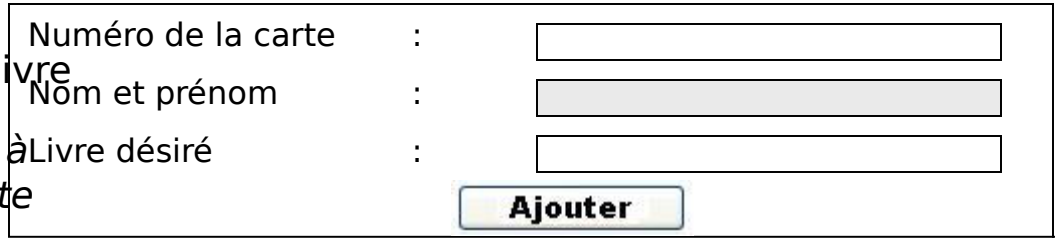
- **SELECT – Restriction (EXISTS) :**
  - **Afficher les étudiants qui n'ont pas empruntés au moins un livre**

→*SELECT* NOM, PRENOM *FROM Etudiant* **E** WHERE **NOT EXISTS** *(SELECT* NUM_LIVRE *FROM Emprunt* WHERE NUM_ETUD=E.NUM_ETUD)

NOM	PRENOM
A	Ali
C	Mohammed
D	Omar

MOPT - LMD

Exemple Écran de la tâche Emprunt livre :



- *Afficher Nom et prénom suite à La saisie du Numéro de la carte*

Code Php

```
$dbhost = "localhost";
$usebdd = "scolarité";
$user= "root";
$password= "";
$connexion = mysql_connect ("$dbhost","$user","$password");
$db = mysql_select_db("$usebdd", $connexion);
$requete="select nomEtud+preEtud as Nom_pre FROM Etudiant E , Carte_Biblio C Where
E.NumEtud=C.NumEtud and Num_Carte='$ Num_Carte '";
$Resultat=mysql_query($requete,$connexion);
$tableau=mysql_fetch_array($resultat);
mysql_close($connexion);
$Nom_pre=$tableau[0];
echo <strong>Nom et Prénom  </strong>  <input type=text name=nom_pre value=$nom_pre
disabled=disabled>
```

MOPT - LMD

- **Vue**

CREATE VIEW *nom_vue* [**(***colonne* [**,**...*n*] **)**] **AS** instruction_*select*

Exemple :

CREATE VIEW essai_vue(nom, prenom,age) AS
SELECT nom, prénom, datediff('yy' , datenaiss , getdate())) FROM Etudiant

⇒ *CREATE VIEW* n'est pas supportée par les version < 5.0 de Mysql

Exemple : *- L'ajout d'un enregistrement à la table Etudiant entraîne la màj automatique de la vue*

Numet	***Nom***	***Prenom***	***Datenaiss***	***…***
1	n1	p1	12/10/1975	…

Nom	***Prenom***	***Age***
n1	p1	29

MOPT - LMD

- **INSERT :**

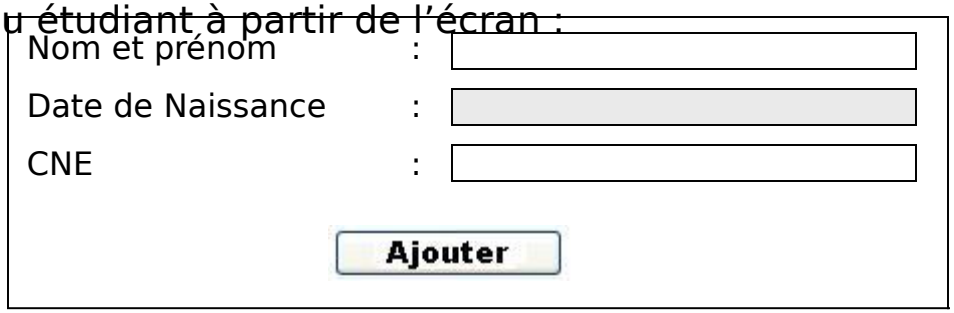
INSERT [INTO] *table_vue* **{** [(*Liste_colonne*)] **Values**

(*Liste_**Valeurs***)**

|(*Liste_colonne*) *Requête_SELECT* **}**

Exemple :

Ajout d'un nouveau étudiant à partir de l'écran :

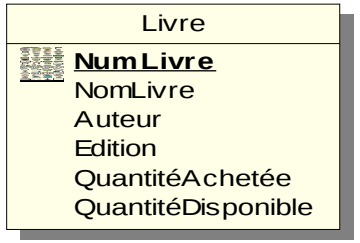


mysql_query("INSERT INTO *Etudiant* (NumEtud, Nom_Pre, DateNaiss , CNE)
Select count(*)+1, $Nom_pre , $DateNaiss , $CNE") FROM Etudiant")

MOPT - LMD

- **UPDATE :**

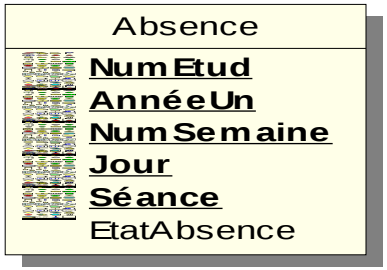
UPDATE *Nom_table*
SET *champ1 = Valeur[,...n]* **WHERE** <*condition*>



Exemple :

Emprunter un livre ⇒ décrémenter de un la Quantité Disponible dans la bibliothèque de ce livre

Update Livre
SET QuantitéDisponible = QuantitéDisponible-1 WHERE Livre = $Livre



- **DELETE :**

DELETE FROM <*liste_table*> **WHERE** <*condition*>

Exemple :

Éliminer les étudiants qui ont absentés plus de 200 séances au cours d'une année universitaire

DELETE FROM *Etudiant E* **WHERE (SELECT** *Count(*)* **FROM** *Absence* **WHERE**