MLD : **M**odèle **L**ogique des **D**onnées

Plan du cours

- Introduction
- Démarche de construction d'un MLD relationnel
- Rappel des concepts du modèle relationnel
- Règles de transformation d'un MCD en un MLD relationnel

Introduction

- La modélisation logique des données est une description de la représentation des données sous un formalisme compatible avec un type de SGBD.

- La modélisation logique des données s ’inscrit dans une démarche générale de construction d’une base de données

  - Modélisation conceptuelle
  - Modélisation organisationnelle (uniquement pour MERISE 2)
  - **Modélisation logique**
  - Modélisation physique

Introduction

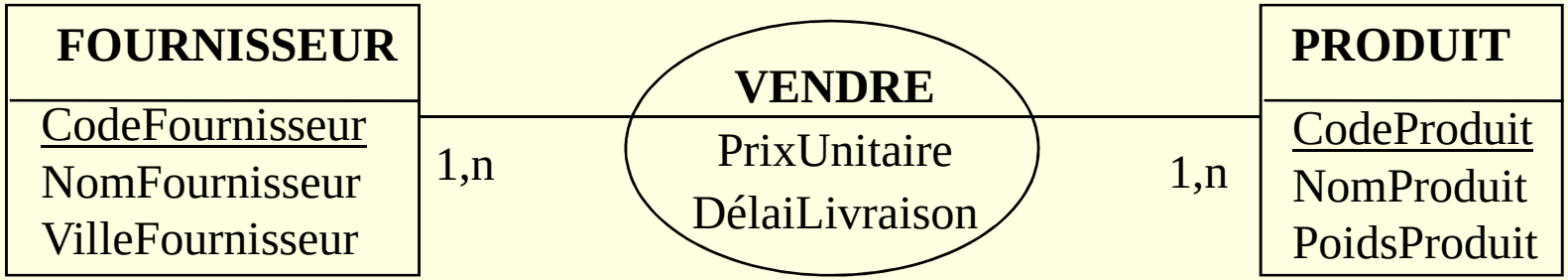
- Deux types de modèles logiques
  - Modèle navigationnel
    - Modèle hiérarchique
    - Modèle réseau
  - Modèle relationnel

Remarque : quand on conçoit une **base de donnée orientée objet**, notre point de départ est un **modèle conceptuel objet**.

Introduction

- Modèle hiérarchique
  - Liens entre ensembles de données = relations père-fils
  - Exemple de SGBD : IMS
- Modèle réseau ou CODASYL
  - Liens entre ensembles de données = relations père-fils + un fils peut avoir +ieurs pères
  - Exemple de SGBD : IDMS
- Modèle relationnel
  - Représentation tabulaire des ensemble de données
  - Liens entre ensembles de données = clés étrangères
  - exemple de SGBD : ORACLE

Exemple de passage



Chaque fournisseur vend des produits sous certaines conditions

Exemple de passage

Schéma Hiérarchique

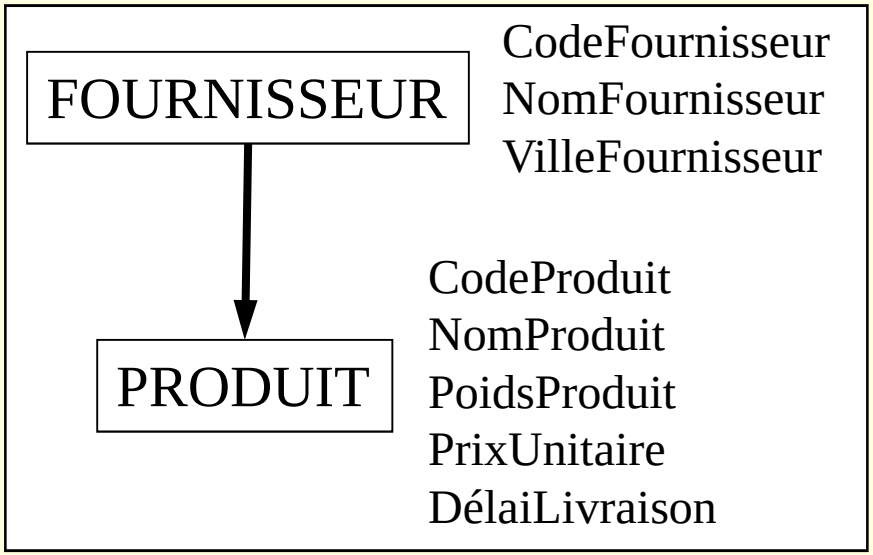


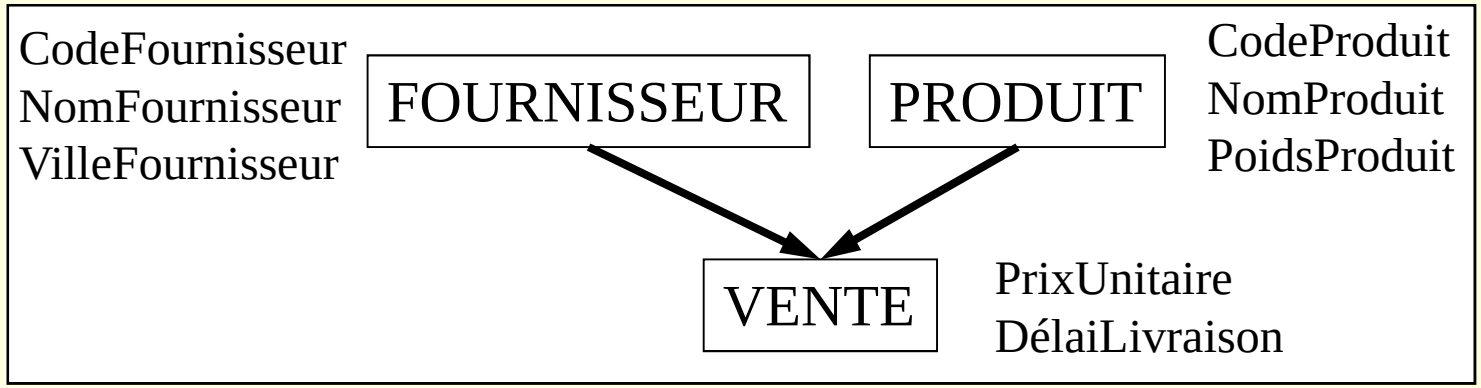
Schéma Relationnel

FOURNISSEUR (CodeFournisseur, NomFournisseur, VilleFournisseur)

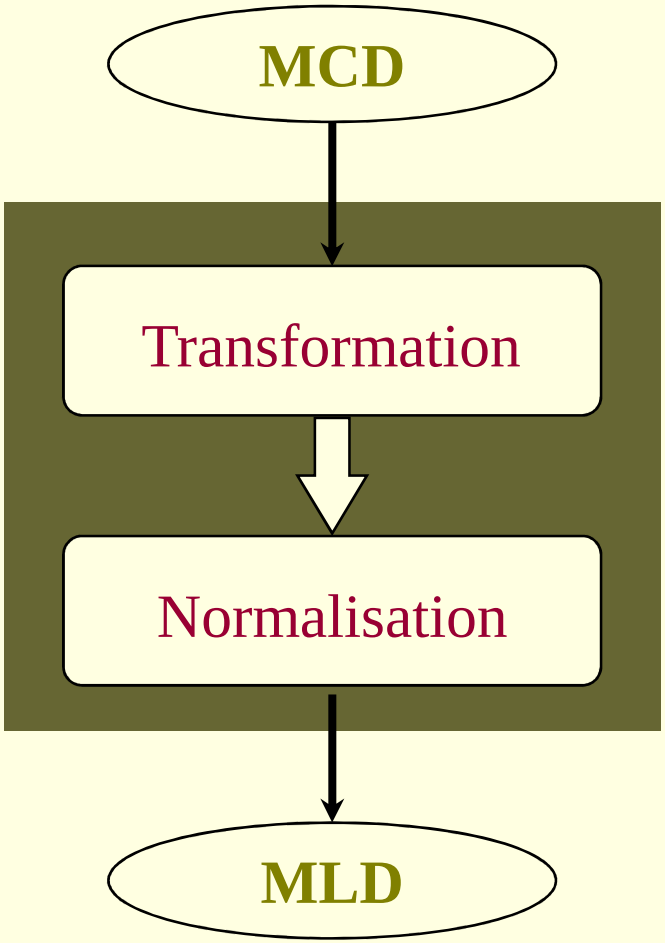
PRODUIT (CodeProduit, NomProduit, PoidsProduit)

COND-VENTE (CodeFournisseur, CodeProduit, PrixUnitaire, DélaiLivraison)

Schéma Réseau



Conception d'un MLD Relationnel



➡ Utilisation de règles de passage

➡ Vérification et décomposition si nécessaire

Règles de transformation d'un MCD en un MLD relationnel

- Règle de transformation de propriété type
- Règles de transformation d'entité type
- Règles de transformation des hiérarchies de généralisations
- Règles de transformation d'association type
- Règles de transformation des contraintes inter-associations
- Règles de transformation des historisations

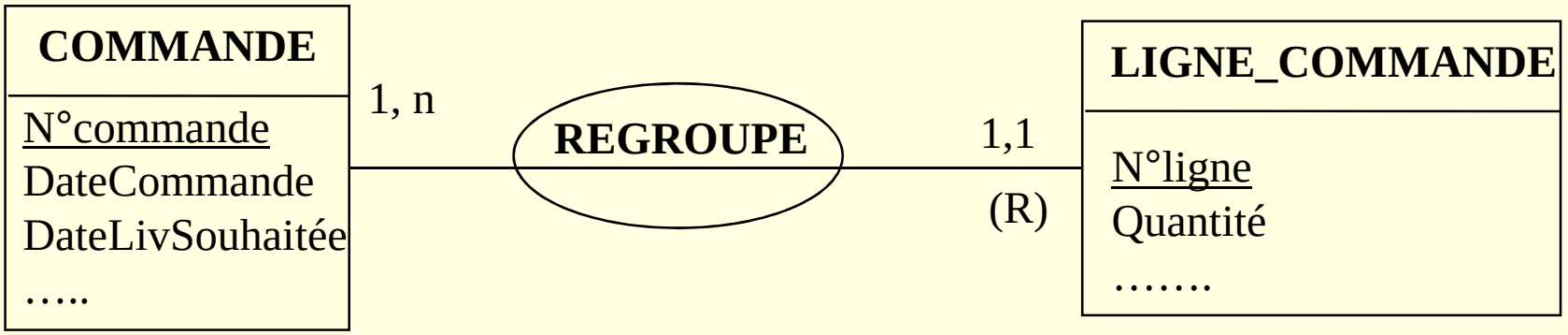
Règles de transformation de propriété type

- Toute propriété d'une entité type ou d 'une association type devient un attribut d'une relation
- Tout groupe de propriété type identifiant une entité est clé primaire de la relation représentant l'entité.
- Tout groupe de propriété type susceptible de jouer de rôle d'identifiant d'une entité est clé candidate de la relation représentant l'entité.
- Toute propriété type d'une entité ou d'une relation conserve son type et ses contraintes lors de la transformation

Règles de transformation d’entité-type

- Une entité type d’un MCD devient une relation du MLD.
- L’identifiant naturel ou artificiel d ’une entité type est clé primaire de la relation qui lui est associée.
- L’identifiant relatif d ’une entité type fait partie de la clé primaire de la relation qui lui est associée. Cette clé est aussi constituée de l’identifiant de l’entité forte dont dépend l’entité faible.

Exemple



COMMANDE (N°commande, DateCommande, DateLivSouhaitée, …..)

LIGNE_COMMANDE (N°commande, N°Ligne, Quantité, ……)

Exemple

Côte est second identifiant

LIVRE (N°Isbn, Titre, …..)

EXEMPLAIRE (N°Isbn, N°Exemplaire, Cote, Etat, ……)

Côte est clé candidate dans Exemplaire

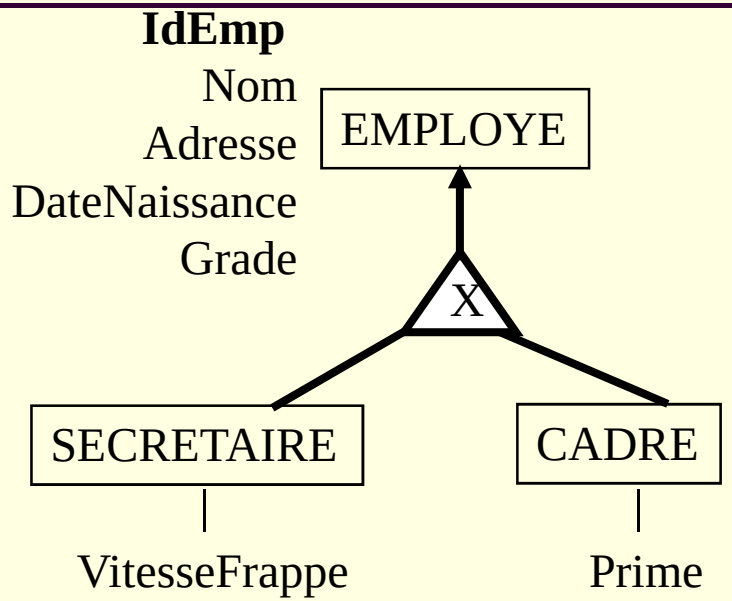
Règles de transformation des hiérarchies de généralisations

- Plusieurs solutions possibles
  - Garder les spécialisations et la généralisation (Cas 1)
    - avec contrainte T
    - avec contrainte X
    - avec contrainte TX
    - aucune contrainte
  - Garder uniquement les spécialisations
    - avec contrainte T
    - avec TX
  - Garder uniquement la généralisation
    - avec contrainte T
    - avec contrainte X
    - avec contrainte TX
    - aucune contrainte

Cas 1 : Garder les spécialisations et la généralisation

- Deux sous-cas
  - Décider de ne pas reproduire les propriétés héritées dans les spécialisations (cas 1.1)
    - Conserver dans la généralisation toutes les occurrences qu'elle représente
  - Décider de reproduire les propriétés héritées dans les spécialisations (cas 1.2)
    - Conserver dans la généralisation uniquement les occurrences qui ne se trouvent pas dans les spécialisations

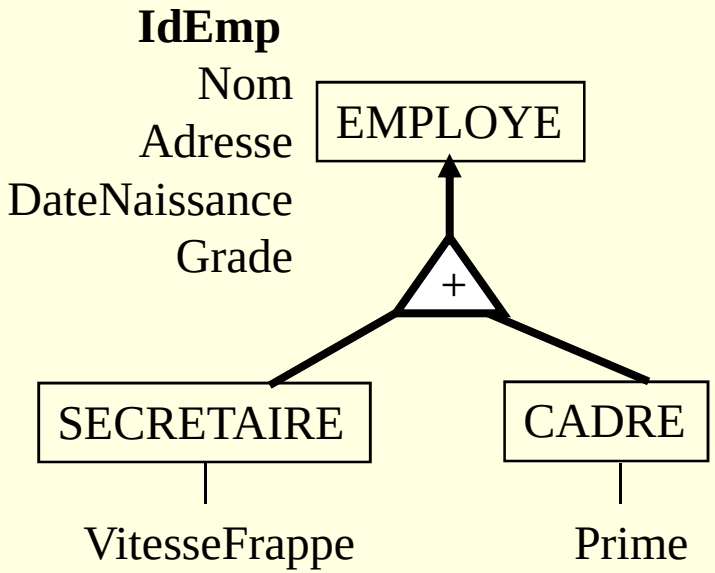
Cas 1 : garder les spécialisations et la généralisation (cas 1.1)



EMPLOYE (IdEmp, Nom, Adresse, DateNaissance, Grade)
SECRETAIRE (IdEmp, VitesseFrappe)
CADRE (IdEmp, Prime)

IdEmp de SECRETAIRE fait référence à EMPLOYE
IdEmp de CADRE fait référence à EMPLOYE
Π_{Idemp} SECRETAIRE $\cap \Pi_{Idemp}$ CADRE = $\varnothing$

Cas 1 : garder les spécialisations et la généralisation (cas 1.2)



AUTRE_EMPLOYE (IdEmp, Nom, Adresse, DateNaissance, Grade)
SECRETAIRE (IdEmp, Nom, Adresse, DateNaissance, Grade, VitesseFrappe)
CADRE (IdEmp, Nom, Adresse, DateNaissance, Grade, Prime)

Vue EMPLOYE = SECRETAIRE $\cup$ CADRE $\cup$ AUTRE_EMPLOYE

Π_{Idemp} SECRETAIRE $\cap$ Π_{Idemp} CADRE = $\varnothing$

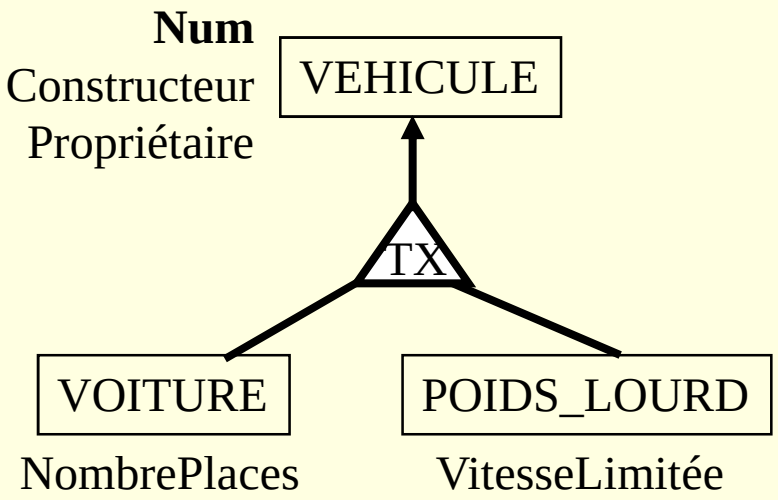
Π_{Idemp} AUTRE_EMPLOYE $\cap$ Π_{Idemp} CADRE = $\varnothing$

Π_{Idemp} AUTRE_EMPLOYE $\cap$ Π_{Idemp} SECRETAIRE = $\varnothing$

Cas 2 : Garder uniquement les spécialisations

- Cas intéressant si contrainte de totalité
- Il faut reproduire toutes les propriétés hérités dans les spécialisations
- Dans le cas d'absence de contraintes d'exclusion on peut vouloir représenter les occurrences communes aux spécialisations dans une autre relation.

Cas 2 : Garder uniquement les spécialisations



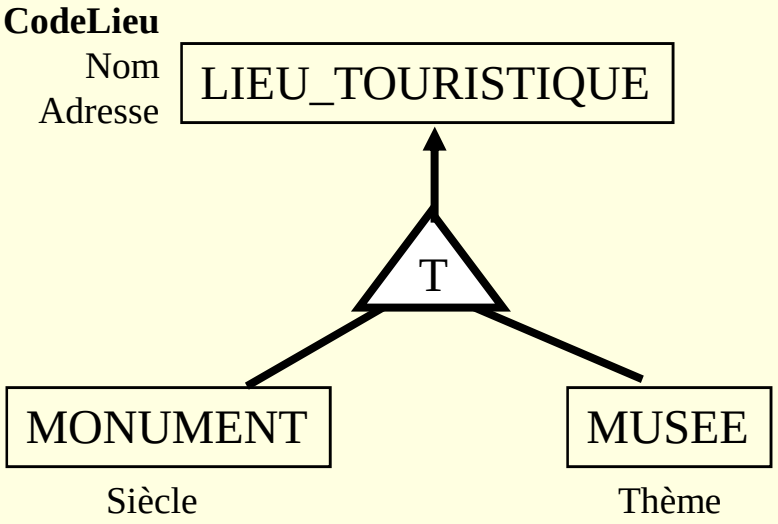
VOITURE (Num, Constructeur, Propriétaire, NomPlaces)

POIDS_LOURD (Num, Constructeur, Propriétaire, VitesseLimiteé)

Vue VEHICULE = VOITURE $\cup$ POIDS_LOURD

Π_{Num} VOITURE $\cap \Pi_{Num}$ POIDS_LOURD = $\varnothing$

Cas 2 : garder uniquement les spécialisations



MONUMENT (CodeLieu, Nom, Adresse, Siècle)

MUSEE (CodeLieu, Nom, Adresse, Thème)

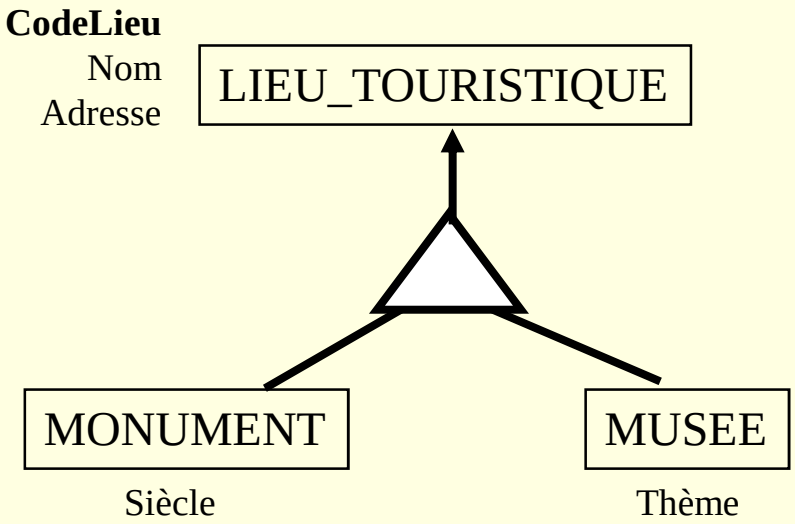
MONUMENT_MUSEE (CodeLieu, Nom, Adresse, Siècle, Thème)

Vue LIEU_TOURISTIQUE = ensemble des occurrences des trois relations

Cas 3 : garder uniquement la généralisation

- Remonter les propriétés héritées dans la généralisation
- Remplacer la sémantique de l'héritage par les contraintes d'existence
- Cas inintéressant dans le cas où le nombre d 'occurrences des spécialisations est élevé

Cas 3 : garder uniquement la généralisation



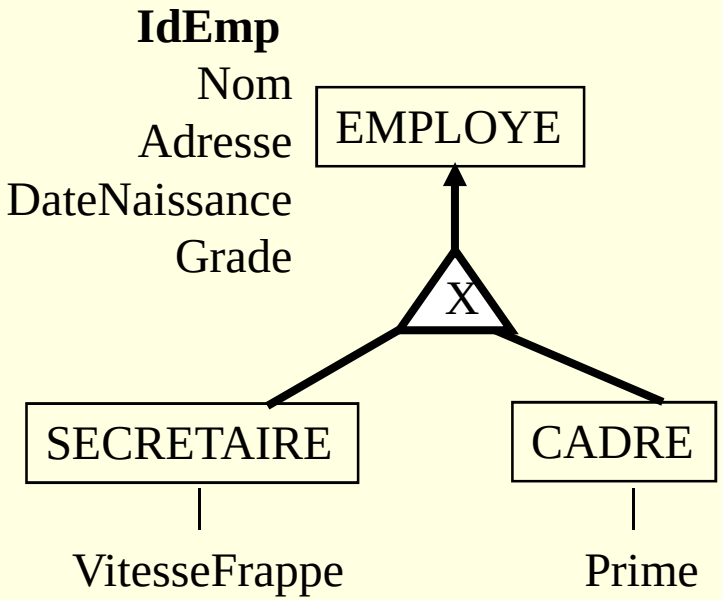
LIEU_TOURISTIQUE (CodeLieu, Nom, Adresse, Siècle, Thème)

Dans LIEU_TOURISTIQUE, Siècle et Thème sont facultatifs

Vue MONUMENT = Ensemble des lieux touristiques tel que Siècle soit valué

Vue MUSEE = Ensemble des lieux touristiques tel que Thème soit valué

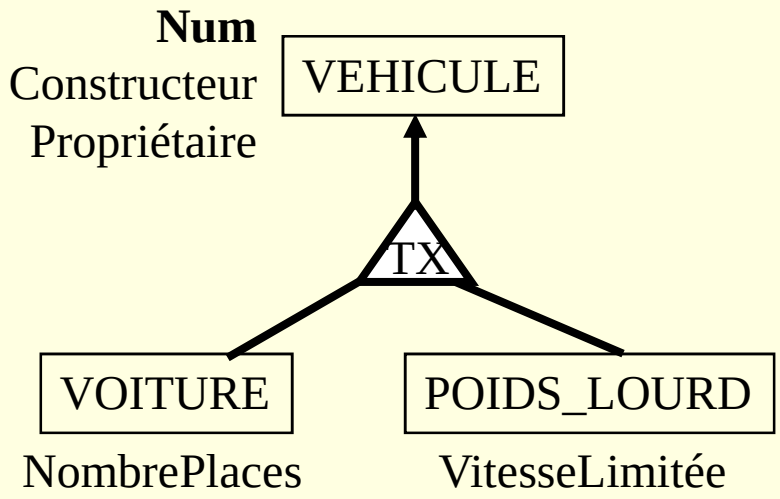
Cas 3 : garder uniquement la généralisation



EMPLOYE (IdEmp, Nom, Adresse, DateNaissance, Grade, VitesseFrappe, Prime)

Dans EMPLOYE, VitesseFrappe et Prime sont facultatifs.
Dans EMPLOYE, VitesseFrappe et Prime sont liés par une contrainte d'existence exclusive.
Vue SECRETAIRE = Ensemble des employés tel que VitesseFrappe soit valuée
Vue CADRE = Ensemble des employés tel que Prime soit valuée

Cas 3 : garder uniquement la généralisation



VEHICULE (Num, Constructeur, Propriétaire, NomPlaces, VitesseLimiteé)

Dans VEHICULE, NomPlaces et VitesseLimiteé sont facultatifs. Cependant au moins l'un des deux doit être renseigné (il n'y a pas de tuples où aucun des deux n'est renseigné).

Dans VEHICULE, NomPlaces et VitesseLimiteé sont liés par une contrainte d'existence exclusive.

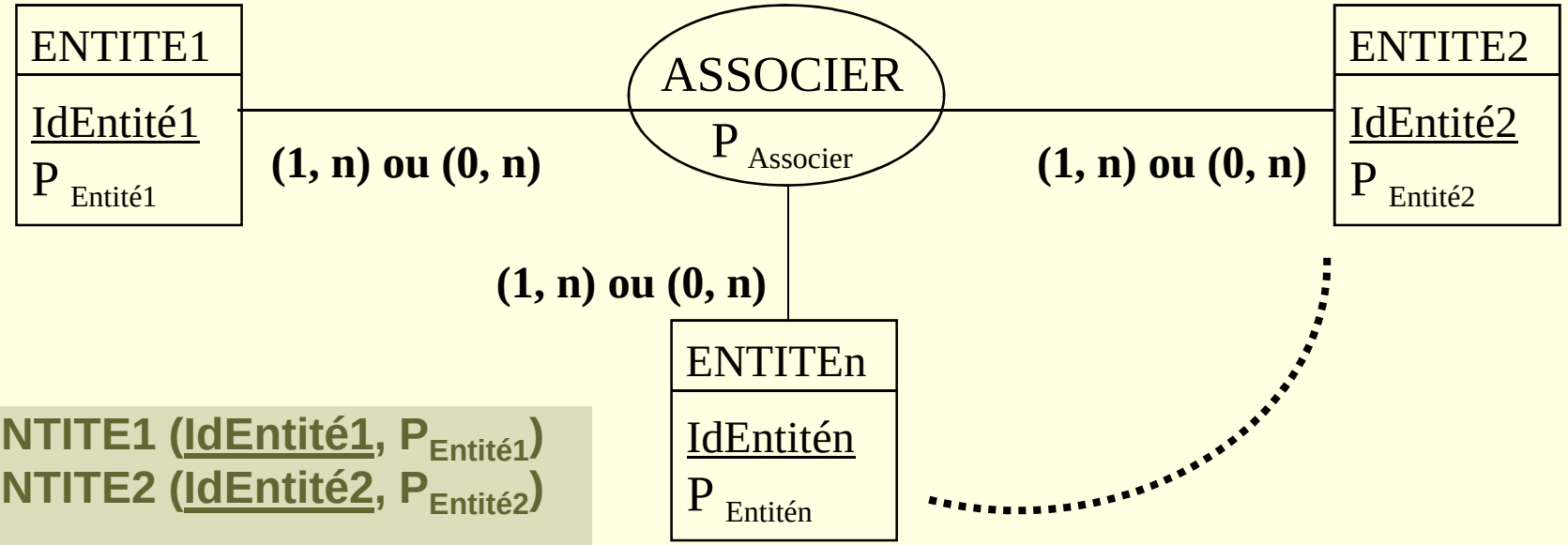
Vue VOITURE = Ensemble des véhicules tel que NomPlaces soit valué

Vue POIDS_LOURD = Ensemble des véhicules tel que VitesseLimitée soit valué

Règles de transformation d'association-type

- Plusieurs cas possibles :
  - Association n-aires (x_1, n) - …….- (x_p, n)
    - Binaire ou autre
  - Association binaire (x,y)-(1,1)
    - (0,n)-(1,1)
    - (1,n)-(1,1)
    - (0,1)-(1,1)
  - Association binaire (x, y) - (0, 1) $(x, y) \neq (1, 1)$
    - (0,n)-(0,1)
    - (1,n)-(0,1)
    - (0,1)-(0,1)

Règle de transformation d'association-type (x1, n) - ….. - (xp, n)



ENTITE1 (IdEntité1, $P_{Entité1}$)
ENTITE2 (IdEntité2, $P_{Entité2}$)
……
ENTITEn (IdEntitén, $P_{Entitén}$)

ASSOCIER (IdEntité1, IdEntité2, ……., IdEntitén, $P_{Associer}$)

IdEntité1 de ASSOCIER fait référence à Entité1
IdEntité2 de ASSOCIER fait référence à Entité2
IdEntité3 de ASSOCIER fait référence à Entité3

Exemple

FOURNISSEUR (CodeFournisseur, NomFournisseur, VilleFournisseur)

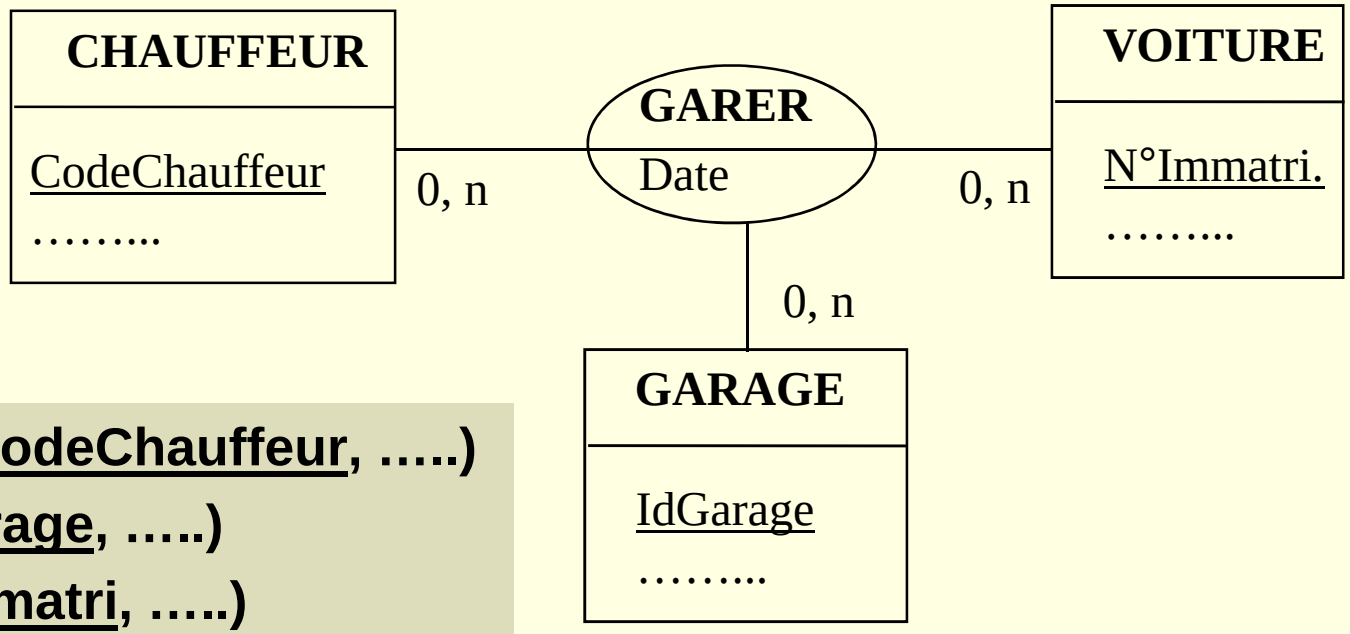
PRODUIT (CodeProduit, NomProduit, PoidsProduit)

COND-VENTE (CodeFournisseur, CodeProduit, PrixUnitaire, DélaiLivraison)

CodeFournisseur de COND-VENTE fait référence à FOURNISSEUR

CodeProduit de COND-VENTE fait référence à Produit

Exemple

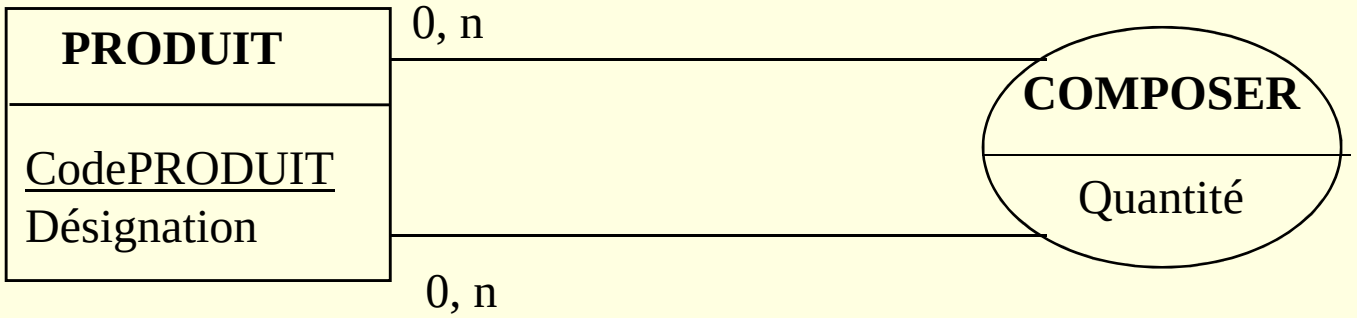


CHAUFFEUR (CodeChauffeur, …..)
GARAGE (IdGarage, …..)
VOITURE (N°Immatri, …..)

GARER (CodeChauffeur, IdGarage, N°Immatri, Date)

CodeChauffeur de GARER fait référence à CHAUFFEUR
IdGarage de GARER fait référence à GARAGE
N°Immatri de GARER fait référence à VOITURE

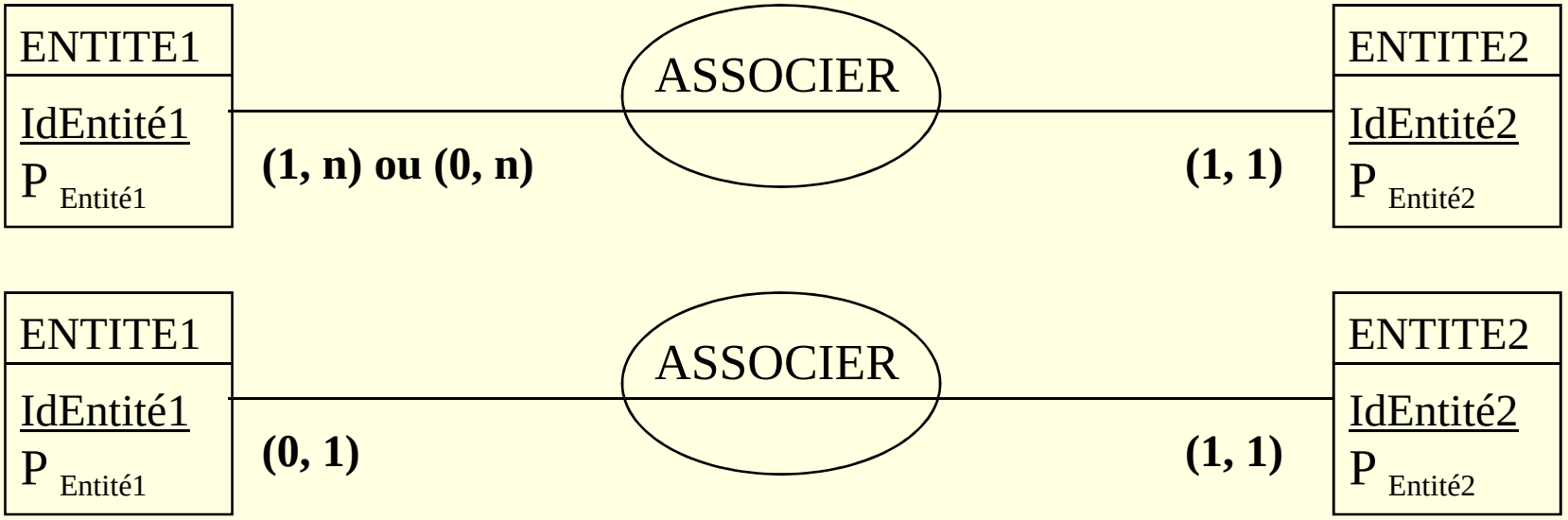
Exemple



PRODUIT (CodeProduit, Désignation)

COMPOSER (Composant, Composé, Quantité)
Composant de COMPOSER fait référence à PRODUIT
Composé de COMPOSER fait référence à PRODUIT

Règle de transformation d'association-type (x, y) - (1, 1)



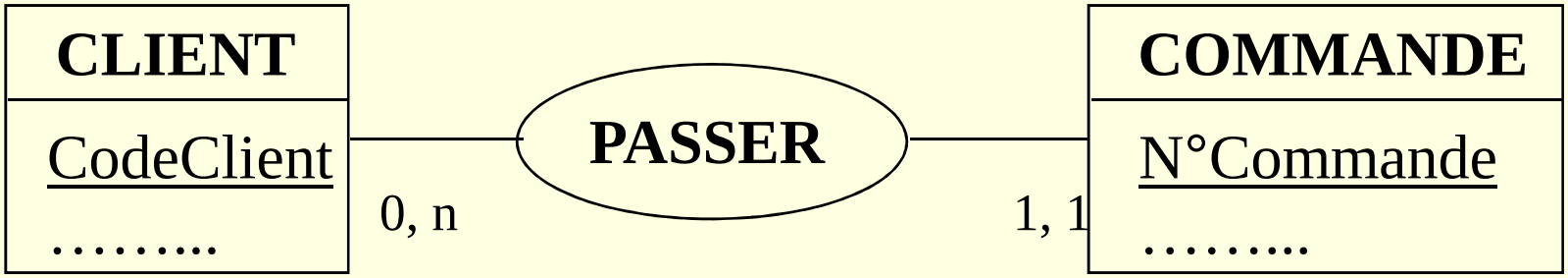
ENTITE1 (IdEntité1, $P_{Entité1}$)

ENTITE2 (IdEntité2, $P_{Entité2}$, IdEntité1)

IdEntité1 de ENTITE2 fait référence à ENTITE1

IdEntité1 de ENTITE2 est obligatoire

Exemple



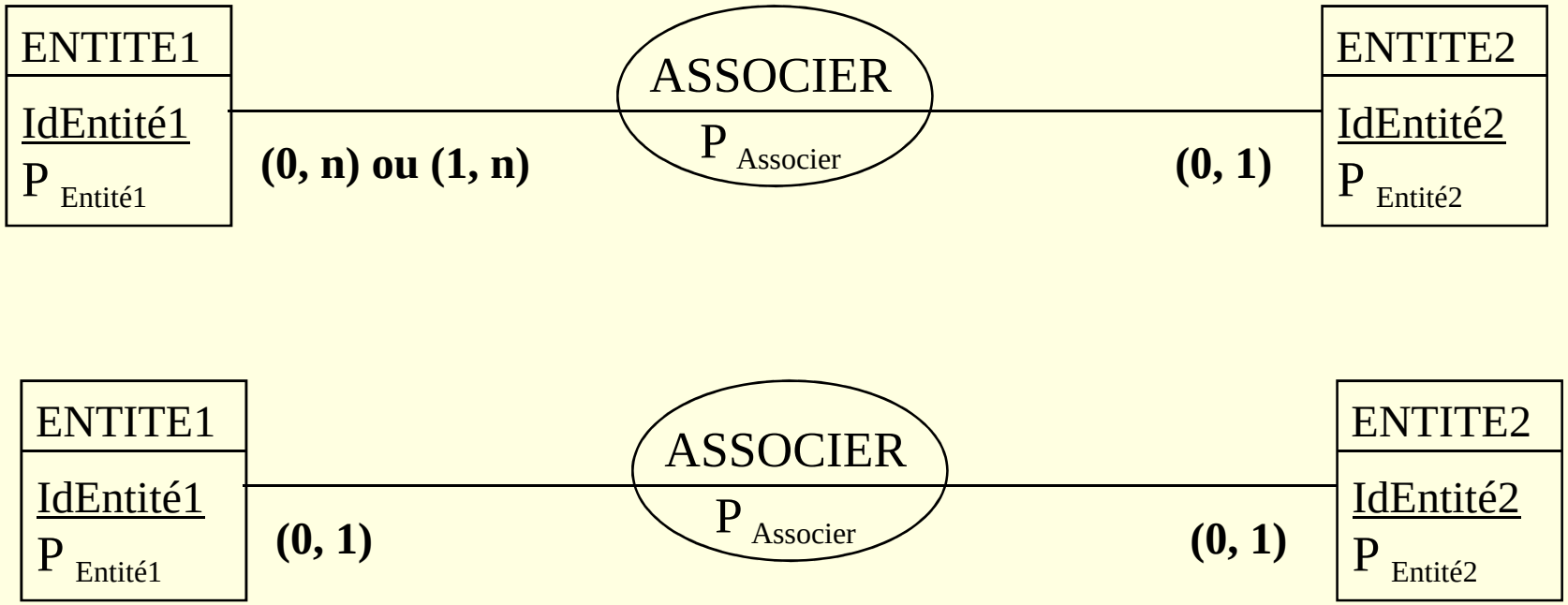
CLIENT (CodeClient, …..)

COMMANDE (N°Commande, ……, CodeClient)

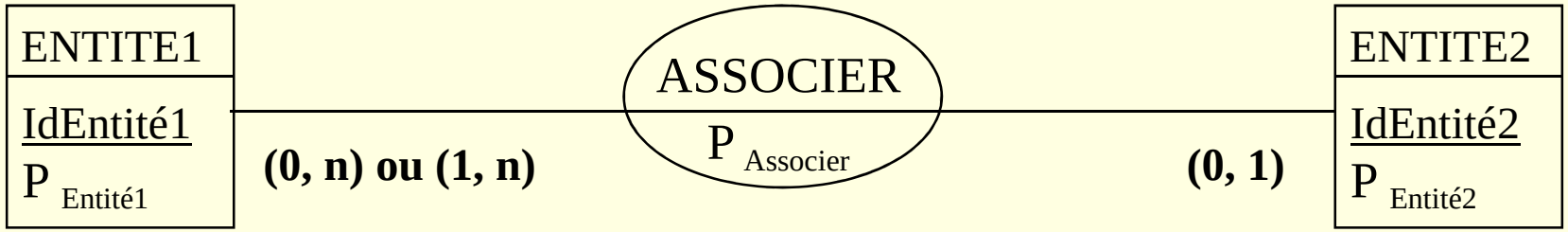
CodeClient dans COMMANDE fait référence à CLIENT

CodeClient dans COMMANDE est obligatoire

Règle de transformation d'association-type (x, y) - (0, 1)



Règle de transformation d'association-type (x, y) - (0, 1)



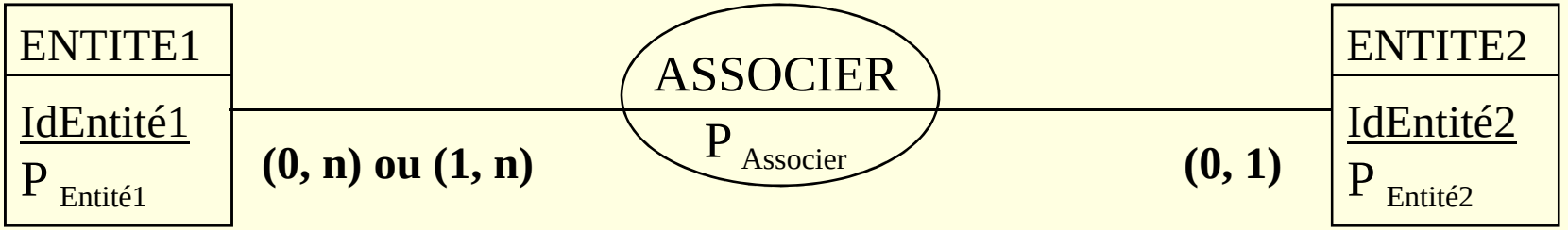
ENTITE1 (IdEntité1, $P_{Entité1}$)

ENTITE2 (IdEntité2, $P_{Entité2}$, IdEntité1, $P_{Associer}$)

IdEntité1 de ENTITE2 fait référence à ENTITE1

$P_{Associer}$ et IdEntité1 de ENTITE2 sont facultatifs mais ils coexistent dans ENTITE2

Règle de transformation d'association-type (x, y) - (0, 1)

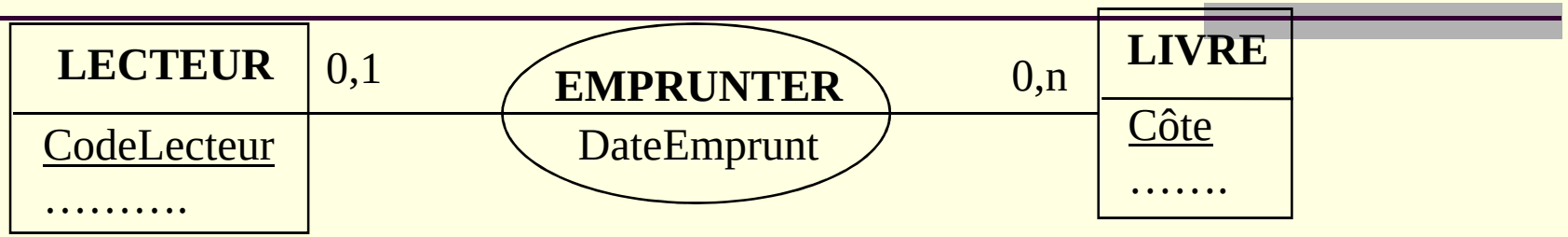


ENTITE1 (IdEntité1, $P_{Entité1}$)
ENTITE2 (IdEntité2, $P_{Entité2}$)

ASSOCIER (IdEntité2, IdEntité2, $P_{Associer}$)

IdEntité2 de ASSOCIER fait référence à ENTITE2
IdEntité1 de ASSOCIER fait référence à ENTITE1

Exemple



LIVRE (Côte, …….)

LECTEUR (CodeLecteur, …….., Côte, DateEmprunt)

Côte de LECTEUR fait référence à LIVRE

DateEmprunt et Côte sont valuée en même temps

Si nombre de lecteur non emprunteur est négligeable

LIVRE (Côte, …….)

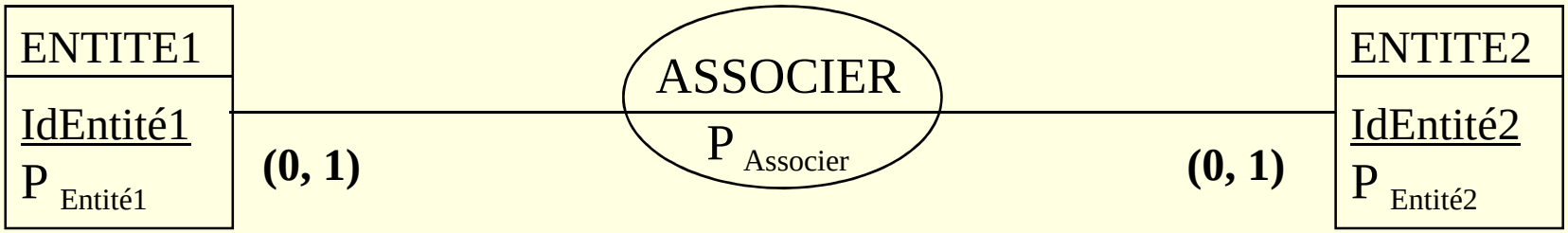
LECTEUR (CodeLecteur, …….)

EMPRUNTER (CodeLecteur, Côte, DateEmprunt)

Côte de EMPRUNTER fait référence à LIVRE

CodeLecteur de EMPRUNTER fait référence à LECTEUR

Règle de transformation d'association-type (x, y) - (0, 1)

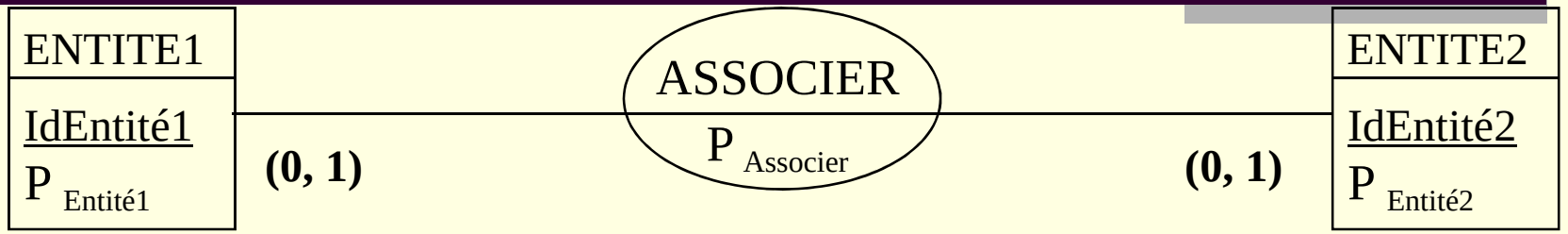


ENTITE1 (IdEntité1, $P_{Entité1}$, IdEntité2)
ENTITE2 (IdEntité2, $P_{Entité2}$, IdEntité1)

ASSOCIER (IdEntité2, IdEntité1, $P_{associer}$)

IdEntité2 de ASSOCIER fait référence à ENTITE2
IdEntité1 de ASSOCIER fait référence à ENTITE1

Règle de transformation d'association-type (x, y) - (0, 1)

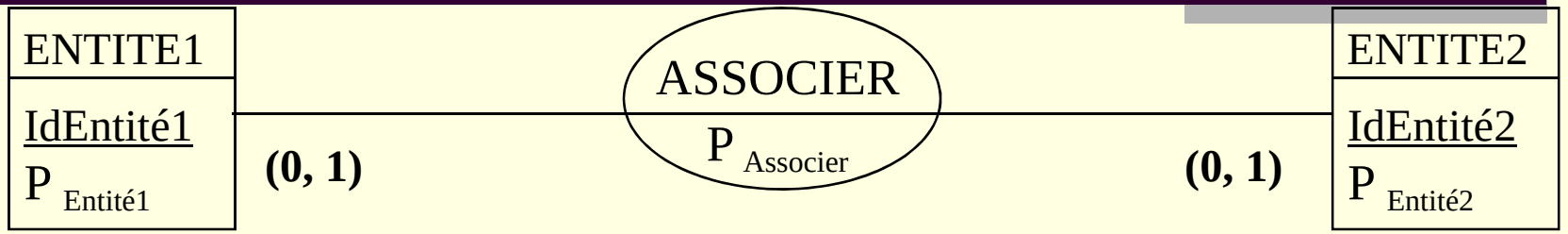


ENTITE1 (IdEntité1, $P_{Entité1}$)
ENTITE2 (IdEntité2, $P_{Entité2}$, IdEntité1, $P_{Associer}$)
IdEntité1 de ENTITE2 fait référence à ENTITE1
IdEntité1 et PAssocier de ENTITE2 coexistent

OU

ENTITE1 (IdEntité1, $P_{Entité1}$, IdEntité2, $P_{Associer}$)
ENTITE2 (IdEntité2, $P_{Entité2}$)
IdEntité2 de ENTITE1 fait référence à ENTITE2
IdEntité2 et PAssocier de ENTITE1 coexistent

Règle de transformation d'association-type (x, y) - (0, 1)



ENTITE1 (IdEntité1, $P_{Entité1}$, IdEntité2, $P_{Associer}$)

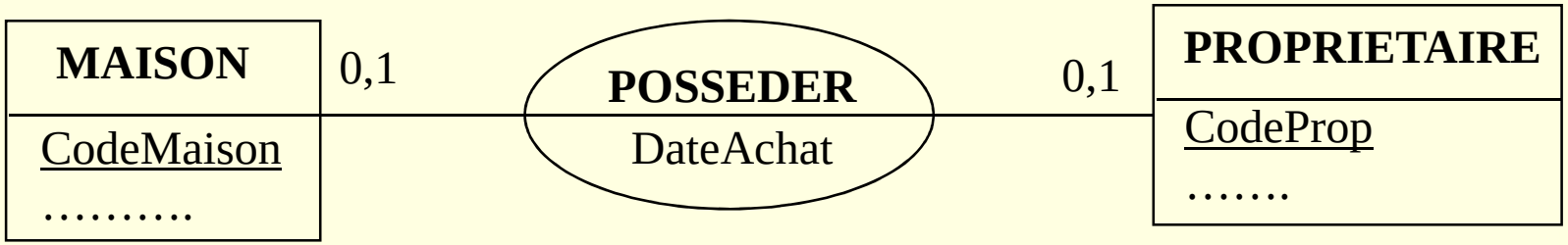
ENTITE2 (IdEntité2, $P_{Entité2}$, IdEntité1)

IdEntité2 de ENTITE1 fait référence à ENTITE2

IdEntité1 de ENTITE2 fait référence à ENTITE1

Si $P_{associer}$ est vide

Exemple



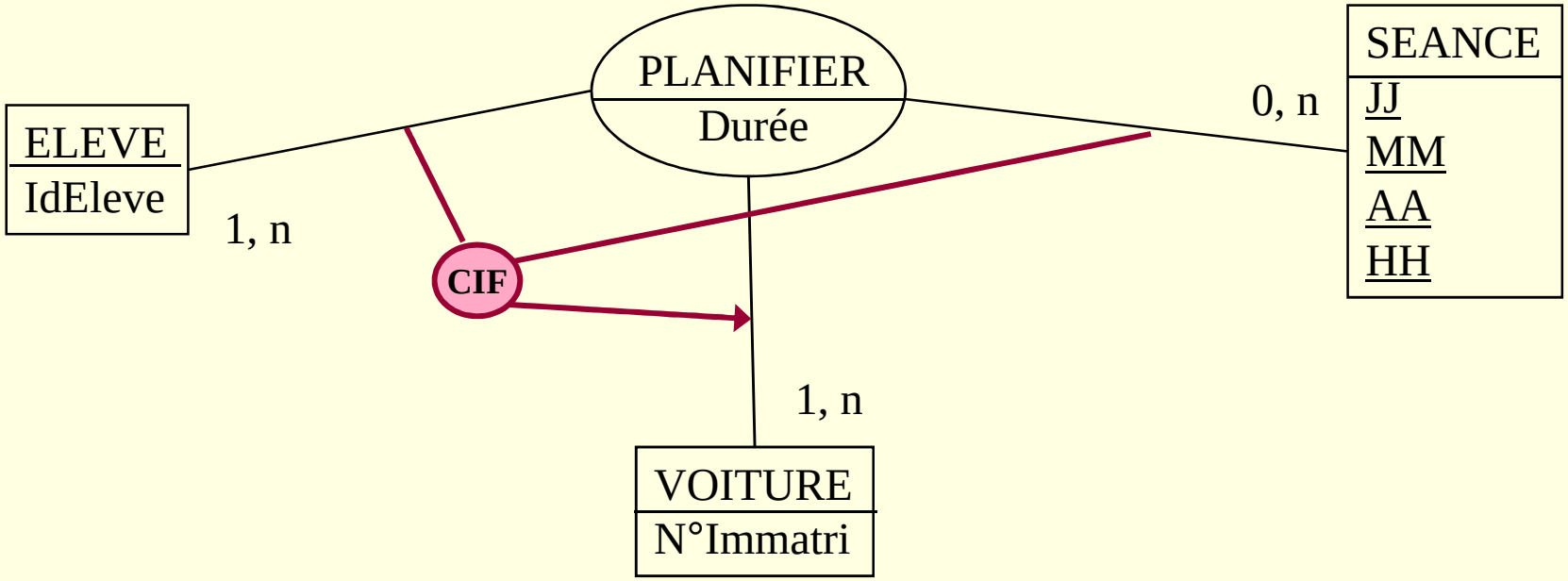
- Trois cas
  - assimiler le cas à un (0, n) (0, n)
  - assimiler le cas à un (1, 1) (0, 1)
    - référence pour une seule entité
    - référence pour les deux entités (ou mettre date d ’achat?)

Règles de transformation des contraintes inter-associations

- Les contraintes intra et inter associations se traduisent par des contraintes d'intégrité de la base de données

- Leur implémentation dépend des possibilités du système

Exemple 1a

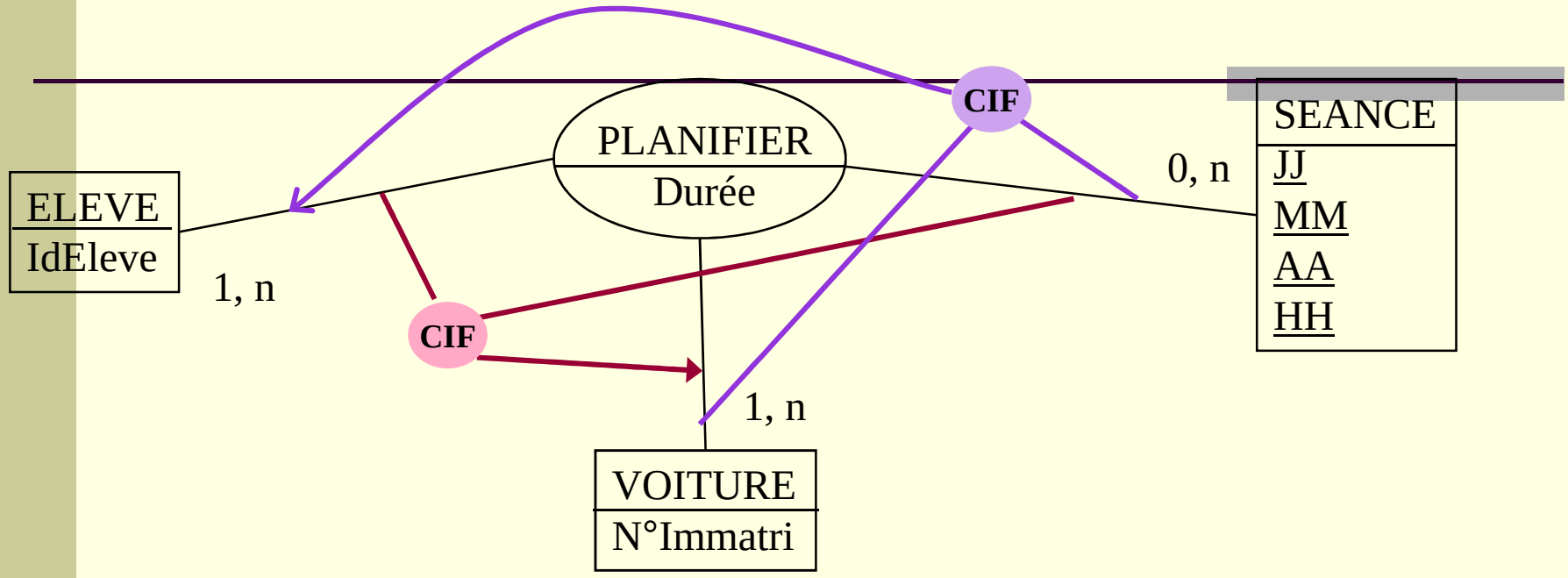


ELEVE (IdEleve, …….)

VOITURE (N°Immatri, ……..)

PLANIFIER (**IdEleve, JJ, MM, AA, HH**, N°Immatri, Durée)

Exemple 1b



ELEVE (IdEleve, …….)

VOITURE (N°Immatri, ……..)

PLANIFIER (**IdEleve, JJ, MM, AA, HH**,
N°Immatri, Durée)

{JJ, MM, AA, HH, N°Immatri} est clé candidate

OU

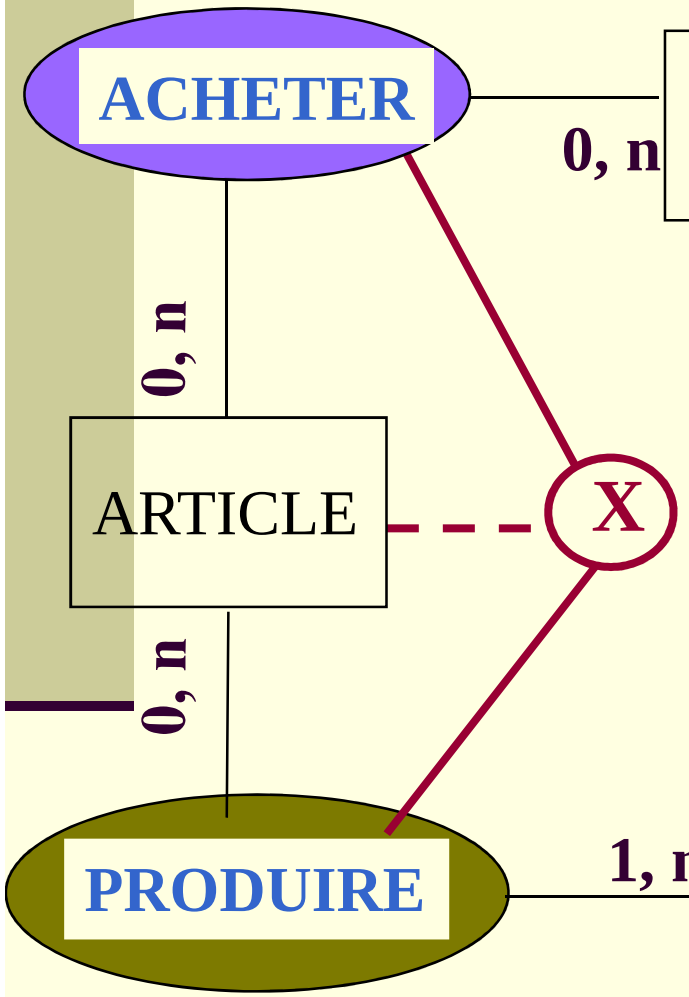
ELEVE (IdEleve, …….)

VOITURE (N°Immatri, ……..)

PLANIFIER (**N°Immatri, JJ, MM, AA, HH**,
IdEleve, Durée)

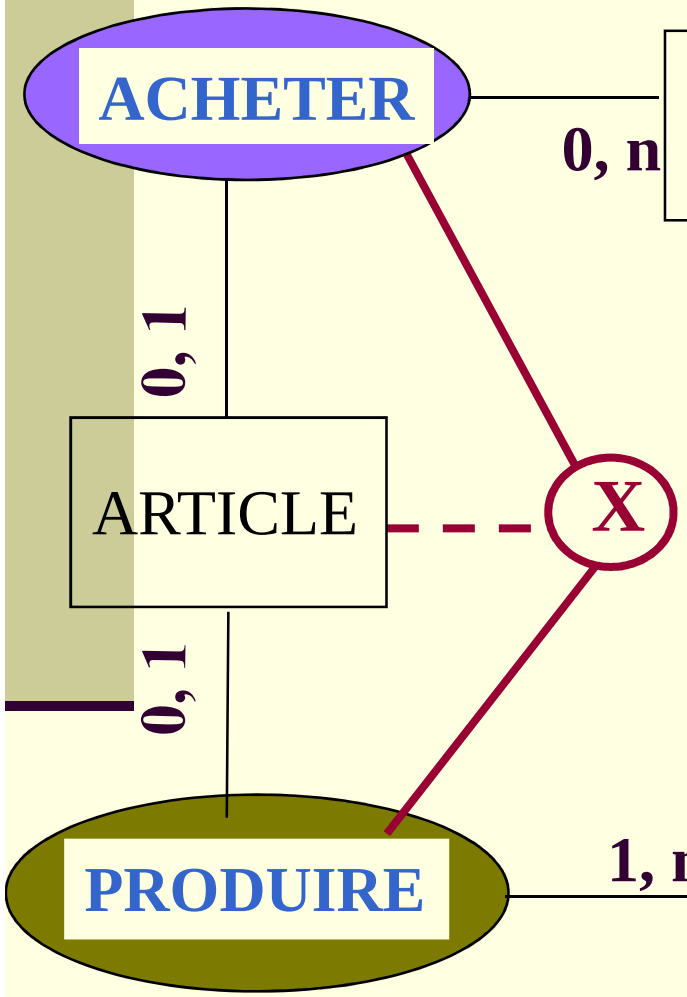
{JJ, MM, AA, HH, IdEleve} est clé candidate

Exemple 2a



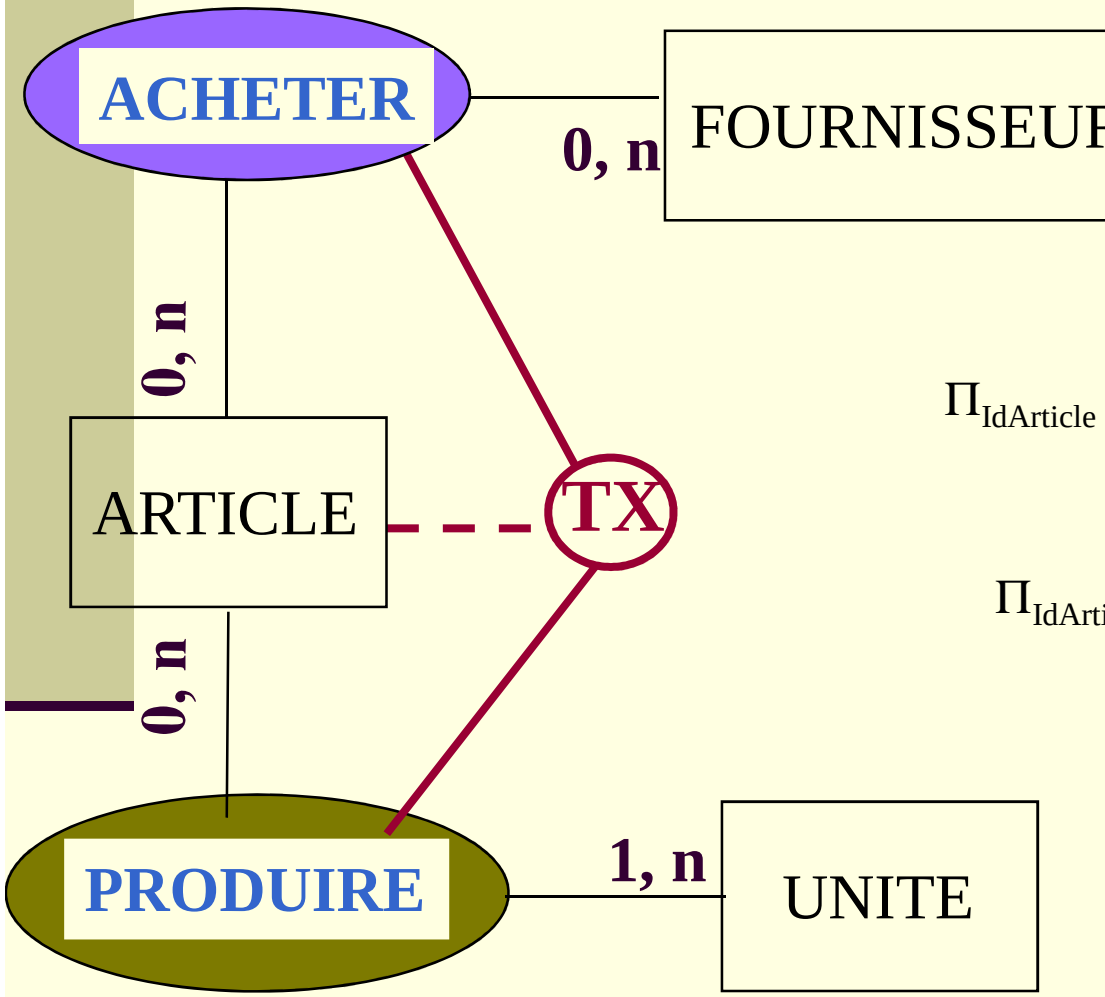
$\Pi_{IdArticle}$ ACHETER $\cap \Pi_{IdArticle}$ PRODUIRE = $\varnothing$

Exemple 2a bis



ARTICLE (id article, …., **idFournisseur**, **IdUnité**)
IdFournisseur de ARTICLE fait référence à FOURNISSEUR
idUnité de ARTICLE fait référence à UNITE
idFournisseur et idUnité ne coexistent pas

Exemple 3a

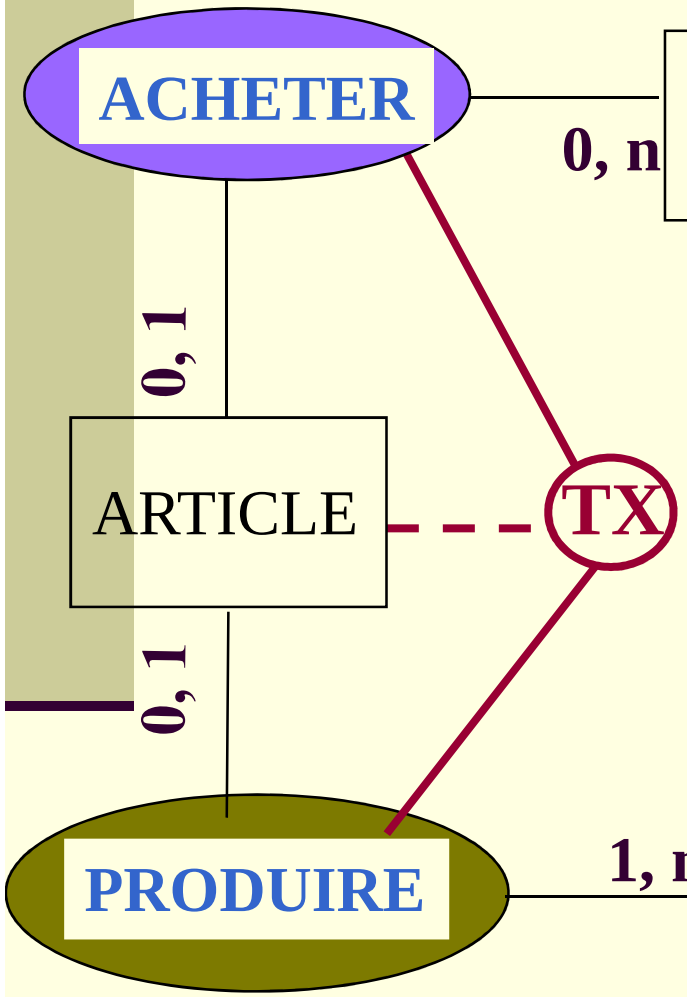


$$\Pi_{IdArticle}\ ACHETER\ \cap\ \Pi_{IdArticle}\ PRODUIRE = \varnothing$$

Et

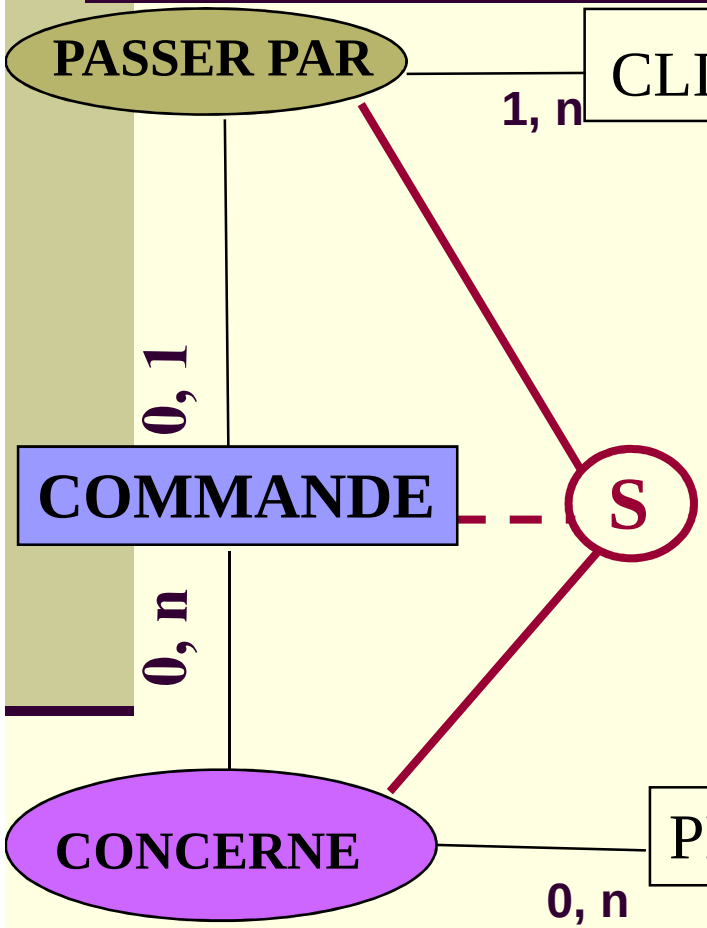
$$\Pi_{IdArticle}\ ACHETER\ \cup\ \Pi_{IdArticle}\ PRODUIRE = \Pi_{IdArticle}\ ARTICLE$$

Exemple 3a bis



ARTICLE (id article, …., **idFournisseur**, **IdUnité**)
IdFournisseur de ARTICLE fait référence à FOURNISSEUR
idUnité de ARTICLE fait référence à UNITE
idFournisseur et idUnité ne coexistent pas mais au moins l'un des deux doit être non null

Exemple 4



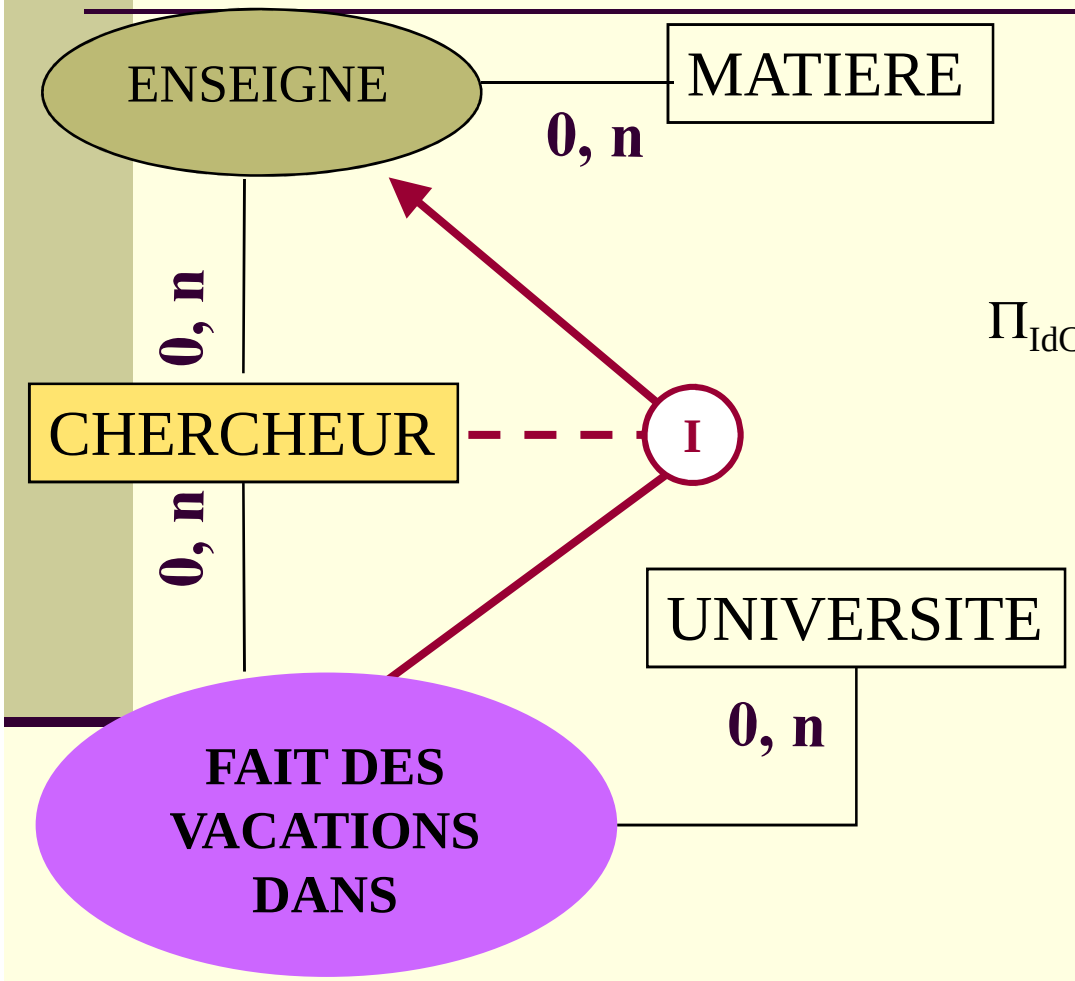
$\Pi_{IdCommande}$ PASSER_PAR = $\Pi_{IdCommande}$ CONCERNE

OU ENCORE

Ne pas créer PASSER_PAR. Mettre Une clé étrangère dans commande (idClient) qui fait référence à CLIENT et ajouter cette contrainte :

$\Pi_{IdCommande}$ ($\sigma_{IdClient<>null}$ (COMMANDE))
=
$\Pi_{IdCommande}$ CONCERNE

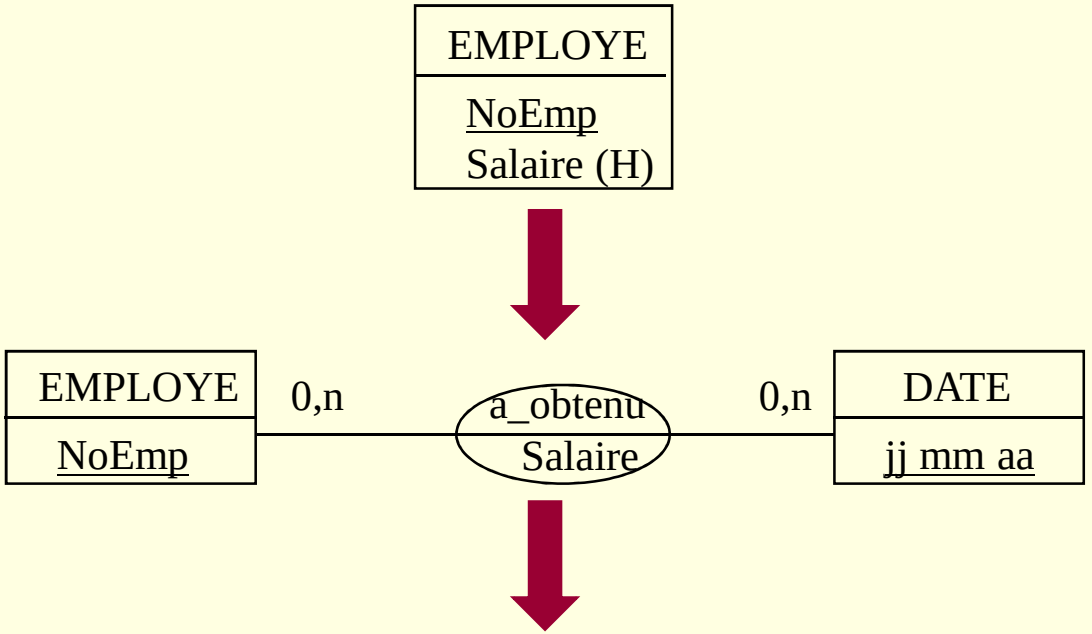
Exemple 5



$$\Pi_{IdChercheur}\ FAIRE\_VACATION\_DANS \subseteq \Pi_{IdChercheur}\ ENSEIGNER$$

Règles de transformation des historisations

- Pour les propriétés



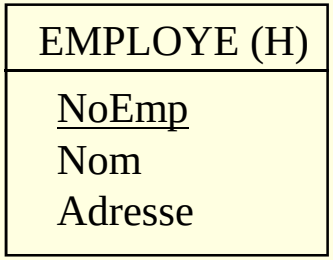
EMPLOYE (N°Emp, …….)

HISTORIQUE_SALAIRE (N°Emp, Date, Salaire)

N°Emp dans HISTORIQUE_SALAIRE fait référence à EMPLOYE

Règles de transformation des historisations

- Pour les entités ou associations

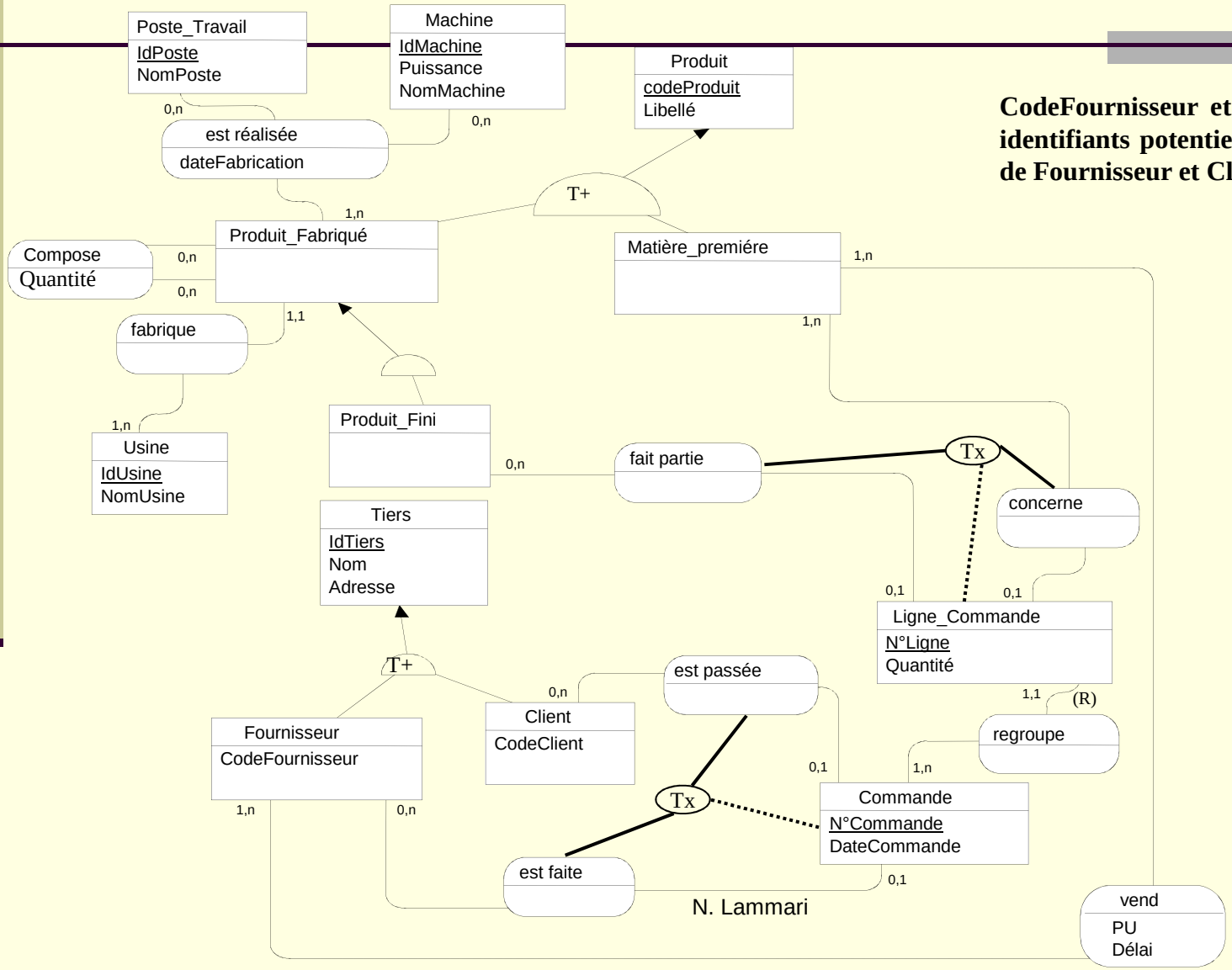


EMPLOYE (N°Emp, …….)

HISTORIQUE_EMPLOYE (N°Emp, Date, …..)

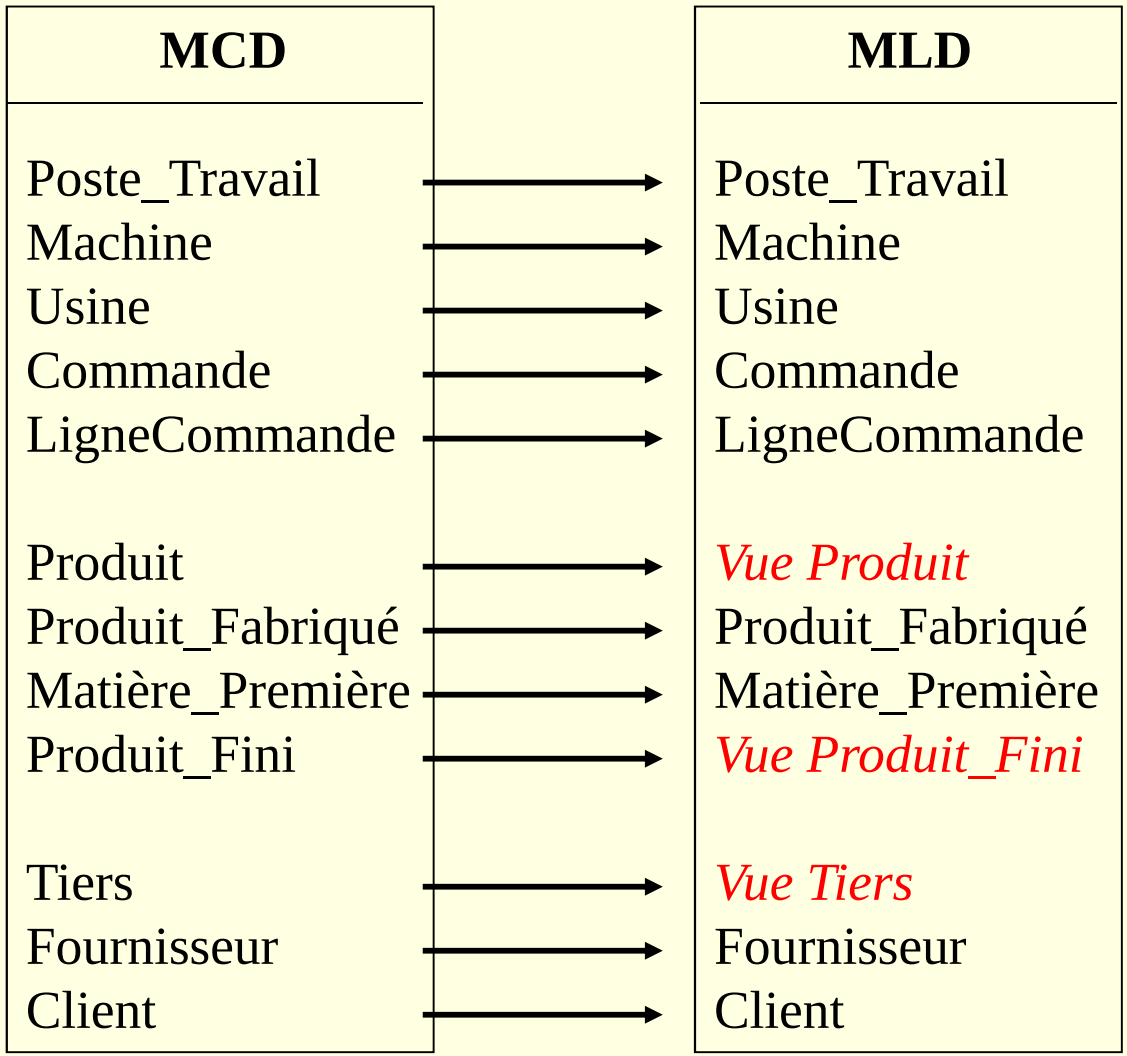
N°Emp dans HISTORIQUE_EMPLOYE fait référence à EMPLOYE

Exemple d'application



CodeFournisseur et CodeClient sont identifiants potentiels respectivement de Fournisseur et Client

N. Lammari


MCD
MLD
Poste_Travail
Machine
Usine
Commande
LigneCommande
Produit
Produit_Fabriqué
Matière_Première
Produit_Fini
Tiers
Fournisseur
Client
Poste_Travail
Machine
Usine
Commande
LigneCommande
Vue Produit
Produit_Fabriqué
Matière_Première
Vue Produit_Fini
Vue Tiers
Fournisseur
Client

Poste_Travail (IdPoste, NomPoste
Machine (IdMachine, Puissance, NomMachine
Usine (Id_Usine, NomUsine
Commande (N°Commande, DateCommande,
LigneCommande (N°Commande, N°ligne, Quantité

Produit_Fabriqué (CodeProduit, Libellé,
Matière_Premiére (CodeProduit, Libellé,

Fournisseur (IdTiers, Nom, Adresse, CodeFournisseur,
Client (IdTiers, Nom, Adresse, CodeClient

Poste_Travail (IdPoste, NomPoste)
Machine (IdMachine, Puissance, NomMachine)
Usine (Id_Usine, NomUsine)
Commande (N°Commande, DateCommande)
LigneCommande (*N°Commande*, N°ligne, Quantité, *CodeMatière*, *CodeProduitFini)*

Produit_Fabriqué (CodeProduit, Libellé, *IdUsine*, *Type)*
Matière_Premiére (CodeProduit, Libellé)

Fournisseur (IdTiers, Nom, Adresse, CodeFournisseur)
Client (IdTiers, Nom, Adresse, CodeClient)

Est_Réalisée (IdPoste, IdMachine, CodeProduit, DateFabrication)
Compose (Composant, Composé, Quantité)
Vend (CodeTiers, CodeProduit, PU, Délai)
Est_Passée (IdTiers, N°Commande)
Est_Faite (IdTiers, N°Commande)

CodeClient et Code Fournisseur sont clé candidates dans leurs relations respectives

IdPoste de Est_Réalisée fait reference à IdPoste de Poste_Travail
IdMachine de Est_Réalisée fait reference à IdMachine de Machine
CodeProduit de Est_Réalisée fait reference à CodeProduit de Produit_Fabriqué

Composant de Compose fait reference à CodeProduit de Produit_Fabriqué
Composé de Compose fait reference à CodeProduit de Produit_Fabriqué

IdUsine de produit_Fabriqué fait reference à Usine
IdUsine de produit_Fabriqué est obligatoire

CodeMatière de Ligne_Commande fait reference à Matière_Premiére

CodeProduitFini de ligne_Commande $\in \Pi_{CodeProduit}(\sigma_{Type = F}$ *(Produit_Fabriqué))*

CodeMatière ne coexiste pas avec CodeProduitFini (ils sont donc forcement facultatifs)

Type $\in$ *'{F, S}*

CodeTiers de Vend fait reference à Fournisseur
CodeProduit de Vend fait reference à Matière_Premiére

N°Commande de Ligne_Commande fait reference à Commande

IdTiers de Est_Passée fait reference à Client
N°Commande de Est_Passée fait reference à Commande

IdTiers de Est_Faite fait reference à Fournisseur
N°Commande de Est_Faite fait reference à Commande

$$\Pi_{N°Commande}(Est\_Faite) \cap \Pi_{N°Commande}(Est\_Passée) = \varnothing$$

$$\Pi_{N°Commande}(Est\_Faite) \cup \Pi_{N°Commande}(Est\_Passée) = \Pi_{N°Commande}(Commande)$$

Vue Produit =

$$\Pi_{CodeProduit,\ Libellé}(Produit\_Fabriqué) \cup \Pi_{CodeProduit,\ Libellé}(Matière\_Premiére)$$

Vue Produit_Fini = $\Pi_{CodeProduit,\ Libellé}(\sigma_{Type = F}\ (Produit\_Fabriqué))$

Vue Tiers =

$$\Pi_{IdTiers,\ Nom,\ Adresse}(Client) \cup \Pi_{IdTiers,\ Nom,\ Adresse}(Fournisseur)$$