

*Collection*  
*Ressources Informatiques*

# Merise

## Guide pratique

modélisation des données  
et des traitements,  
langage SQL

Jean Luc BAPTISTE

**Nouvelle  
Edition**

 INFORMATIQUE TECHNIQUE



# Merise

Guide pratique (nouvelle édition)

Jean-Luc BAPTISTE



## Résumé

Ce livre sur la **méthode Merise** s'adresse tout particulièrement aux étudiants en premier cycle d'informatique, aux étudiants en école de gestion et à toute personne souhaitant une **information simple, directe et pratique** sur la méthode Merise et sur le langage SQL.

Dans la partie sur la **méthode Merise**, vous découvrirez comment :

- Réaliser les différents modèles (modèles conceptuels, modèles logiques, modèles physiques) mais aussi les modèles spécifiques aux traitements (modèles conceptuels des traitements, modèles organisationnels des traitements...).
- Modéliser avec les extensions Merise/2.
- Comparer certains modèles Merise à certains diagrammes UML.

Le **langage SQL** est présenté de façon progressive et est illustré par de nombreux exemples. Dans cette partie vous apprendrez à :

- Manipuler, filtrer, trier, regrouper les données.
- Créer, modifier, supprimer des tables.
- Affecter ou enlever des droits à certains utilisateurs.

L'auteur n'a volontairement gardé que le **côté concret** de la méthode Merise et du langage SQL, pour permettre au lecteur une **immersion immédiate**. Il propose de **nombreux exercices** pour faciliter cette assimilation.

## L'auteur

Ancien responsable informatique et développeur d'applications stratégiques, **Jean-Luc Baptiste** a également créé et géré une société de service en informatique pendant plusieurs années. Il est aujourd'hui professeur certifié en informatique de gestion, en filière BTS, au Lycée Jean Lurçat de Perpignan. Très proche des interrogations des apprenants et fort de son expérience pédagogique et de ses compétences techniques, il propose au lecteur un ouvrage efficace pour s'initier à la méthode Merise et au langage SQL.

*Ce livre numérique a été conçu et est diffusé dans le respect des droits d'auteur. Toutes les marques citées ont été déposées par leur éditeur respectif. La loi du 11 Mars 1957 n'autorisant aux termes des alinéas 2 et 3 de l'article 41, d'une part, que les "copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective", et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, "toute représentation ou reproduction intégrale, ou partielle, faite sans le consentement de l'auteur ou de ses ayants droit ou ayant cause, est illicite" (alinéa 1er de l'article 40). Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles 425 et suivants du Code Pénal. Copyright Editions ENI*

# Introduction

Le Système d'Information, appelé aussi SI, est un ensemble de tâches complexes regroupées en modules spécialisés qui composent l'applicatif informatique : le logiciel.

Ces tâches complexes sont généralement un assemblage de tâches plus simples. Ces tâches simples sont les briques de base de l'applicatif.

Si nous devons créer une analogie nous pourrions dire que ces tâches simples sont comme les briques qu'un maçon assemble pour ériger une maison. Le logiciel, tout comme une maison, a besoin d'un plan de conception réalisée par un architecte. Une maison conçue sans plan risque de présenter, une fois finie, plus d'une erreur de conception. Il en est de même pour un logiciel. Le logiciel sans études préalables, construit sans méthodologie, risque de surprendre son utilisateur !

L'objet de ce livre est de vous présenter de façon pragmatique, simple et progressive la méthode Merise, son prolongement par Merise/2 et de la positionner par rapport au langage UML.

# Historique de la méthode Merise

**Merise** est un acronyme signifiant Méthode d'Étude et de Réalisation Informatique par les Sous-Ensembles ou pour les Systèmes d'Entreprise.

La méthode Merise a comme objectif d'aider, de guider les Ssii, dans leurs phases d'analyses, de conception et le développement de l'applicatif.

Nous devons la création, l'étude et la mise en place de cette méthode à une équipe de chercheurs et d'ingénieurs aixois (Jean-Louis le Moigne, Hubert Tardieu, Dominique Nancy, Henry Heckenroth, Daniel Pasco, Bernard Espinasse) qui en posèrent les bases dans le milieu des années 1970.

Le ministère de l'Industrie vit en cette méthode un excellent moyen pour standardiser et rationaliser les rapports existant entre les administrations et leurs sous-traitants. C'est pourquoi il finança quelque temps les recherches sur la méthode Merise. Le challenge était de pouvoir proposer des outils ou des méthodologies permettant aux donneurs d'ordres et aux développeurs de se comprendre et ainsi de mieux appréhender chacun de leur côté, avec leur propre culture professionnelle, l'ensemble du système d'information.

La méthode Merise présente comme avantage indéniable de permettre une définition claire et précise de l'ensemble du Système d'Information et d'en définir correctement le périmètre.

Cette méthode est actuellement enseignée aux étudiants se dirigeant vers des études informatiques, mais aussi aux étudiants voulant suivre des études comptables. Nous retrouvons là le besoin qui avait poussé le ministère de l'Industrie à investir dans cette méthode. En effet, dans les petites et moyennes entreprises qui n'ont souvent pas de service informatique c'est le comptable qui est l'interlocuteur privilégié entre l'entreprise et le prestataire de services informatiques.

Dominique Nancy, Henry Heckenroth rejoignirent Bernard Cohen créateur de la société Cecima distributrice du logiciel Win'Design qui permet de concevoir les différentes phases d'un projet piloté par la méthode Merise.

C'est avec ce logiciel que les différents schémas illustrant ce livre ont été réalisés.

Voici le lien internet vers la société Cecima où vous pourrez télécharger une version de démonstration de leur logiciel Win'Design.

<http://www.win-design.com/fr/index.htm>

# Présentation générale de la méthode Merise

La méthode Merise se caractérise par :

- une approche systémique en ayant une vue de l'entreprise en terme de systèmes ;
- une séparation des données (le côté statique) et des traitements (le côté dynamique) ;
- une approche par niveaux.

# La systémique

En anatomie, un système désigne un ensemble de parties similaires qui participent à une activité commune (système cardiaque, système digestif, système respiratoire, etc.). En science, le système peut servir pour définir des unités, par exemple le système métrique.

Selon le Robert, un système est un dispositif formé par la réunion d'éléments analogues.

La définition du Larousse semble plus explicite : « Combinaison de parties qui se coordonnent pour concourir à un résultat, de manière à former un ensemble ».

Tout système fonctionne en transformant des flux d'entrée en flux de sortie selon des processus plus ou moins complexes.

## 1. Les caractéristiques d'un système

Un système est un élément fini dont le périmètre est une frontière qui le sépare de son environnement.

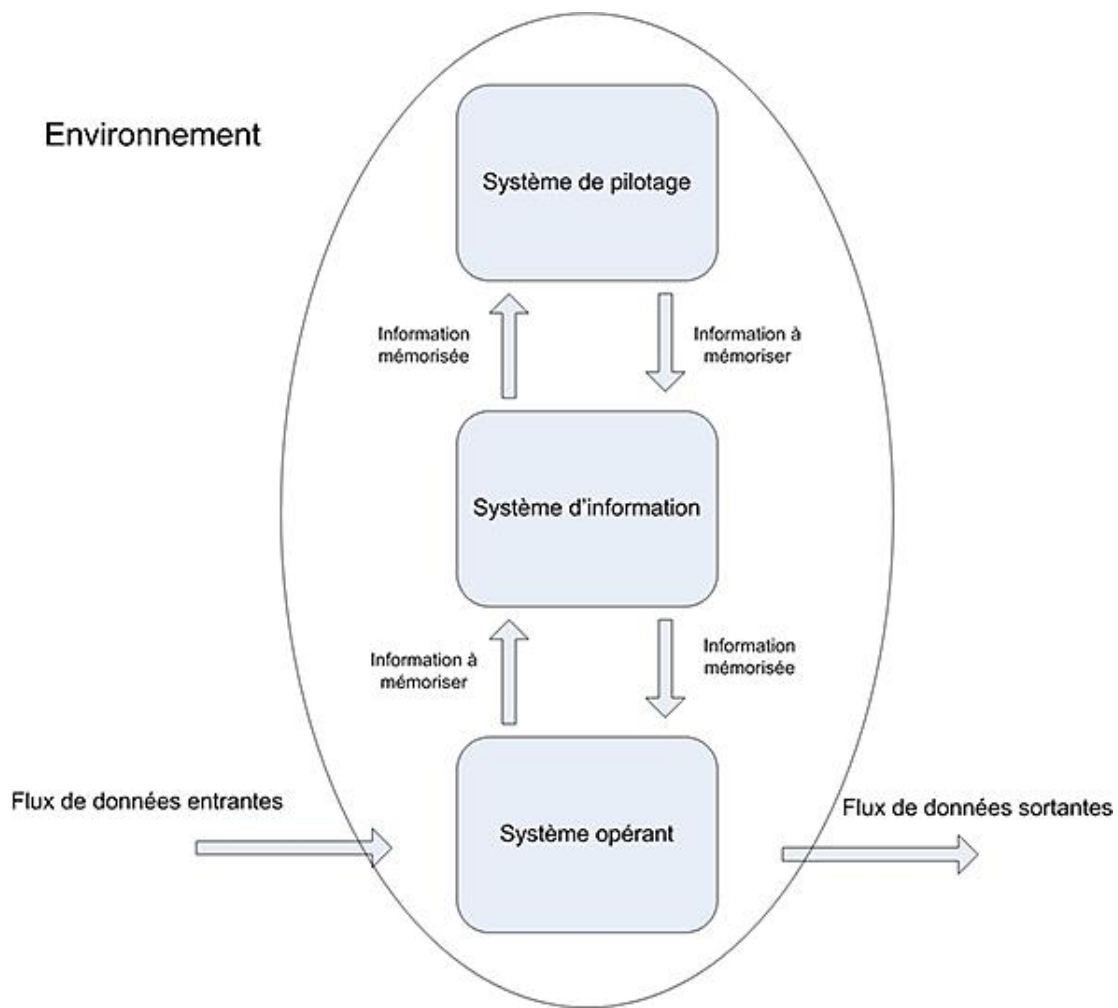
Il interagit avec son environnement grâce à des flux d'informations entrantes, qu'il va traiter et restituer à l'environnement sous forme de flux d'informations sortantes.

Le système va générer des informations qui rendent compte de son comportement à la fois au sein de l'environnement, mais aussi pour son propre compte. Un système communique.

Un système a besoin, pour prendre des décisions, de stocker et de traiter des informations.

## 2. La représentation schématique des systèmes de l'entreprise

Si nous reprenons l'analogie anatomique, et si nous comparons l'entreprise à un corps humain, nous pouvons réduire l'entreprise à un cerveau qui pilote, un muscle qui opère et des nerfs qui font transiter les informations. Voici un schéma simplifié qui en découle :



### **a. Le système de pilotage**

Le système de pilotage définit les missions et les objectifs, organise l'emploi des moyens, contrôle l'exécution des travaux. Il assigne des objectifs à l'organisation, analyse l'environnement et le fonctionnement interne à l'organisation, contrôle le système opérant. Il est relié aux autres systèmes par des flux d'informations internes.

### **b. Le système d'information**

Le système d'information est l'ensemble des ressources humaines, techniques et financières qui fournissent, utilisent, compilent, traitent et distribuent l'information de l'organisation. Il alimente l'organisation en informations d'origines diverses (internes ou externes). Il est la passerelle obligatoire pour toutes les informations de l'entreprise.

### **c. Le système opérant**

Le système opérant est l'ensemble des moyens humains, matériels, organisationnels qui exécutent les ordres du système de pilotage. Il assure le fonctionnement du système global, son activité est contrôlée par le système de pilotage.

# La séparation des données et des traitements

## 1. Les données (ou informations)

L'information est l'émission ou la réception de signaux oraux ou écrits, sonores, visuels ou multimédias dont le but est de déclencher les processus alimentant l'échange, base naturelle et indispensable de l'animation de l'organisation.

Les informations se recueillent à l'intérieur du domaine à étudier. La liste d'informations est constituée de plusieurs façons :

- l'interview ;
- l'étude des documents internes ;
- l'étude des documents externes.

### a. L'interview

Une des phases du recueil d'information est un entretien avec les différents acteurs de l'organisation. Cet entretien permet de définir le périmètre de l'applicatif futur. Les informations orales sont classées et regroupées en parties distinctes. Ainsi, les informations concernant l'enregistrement des données de l'organisation seront regroupées.

### b. L'étude des documents internes

Les documents internes (factures, bons de livraison, ordres de fabrication) recèlent des informations qui sont souvent omises lors des entretiens. Ces oublis sont dus au caractère automatique et récurrent de ces informations. Les personnes qui les manipulent au quotidien oublient souvent de les citer tant elles leur paraissent évidentes.

### c. L'étude des documents externes

L'étude des documents externes (factures des fournisseurs, bons de livraison fournisseurs...) tout comme l'étude des documents internes permet de découvrir des informations oubliées lors des interviews et de découvrir aussi quelques règles de gestion.

Pour ce recueil d'informations, il est nécessaire de respecter certaines règles pour éviter des erreurs futures.

Avant d'ajouter une information, il est impératif de s'assurer qu'elle n'est pas déjà présente. Par exemple, un numéro client peut apparaître sur un bon de livraison et sur une facture. Ce n'est pas la peine de le répertorier deux fois.

De même, une information peut être synonyme d'une autre. Par exemple sur le bon de livraison il apparaît « Code client » et sur la facture « Numéro Client ». Il est impératif de ne garder qu'une seule des deux informations.

## 2. Les différents types d'informations

### a. Les informations élémentaires et mémorisables

Les informations élémentaires sont des informations dont les valeurs **ne peuvent pas être inventées**, elles ne sont pas déductibles d'autres informations.

Par exemple, un nom de client ou sa raison sociale ne peuvent pas être inventés.

Une quantité commandée ne peut pas non plus être inventée.

Une information doit être **atomique**, c'est-à-dire non décomposable. Par exemple si l'information Adresse doit contenir « 36, rue de la paix 75000 Paris » celle-ci peut être décomposée en plusieurs informations élémentaires :

- Adresse ;
- Code postal ;



- Ville.

Chaque valeur prise par une information est appelée une **occurrence**. Par exemple, l'information Nom peut avoir les occurrences suivantes :

- Baptiste ;
- Durand.

## **b. Les informations calculées**

Les informations calculées sont déductibles des informations élémentaires.

Par exemple, le total d'une ligne de commande est le résultat de la multiplication du prix de vente hors taxe et de la quantité commandée.

## **c. Les traitements**

Ils sont collectés comme les informations via un processus d'interview et d'étude des documents.

Ils peuvent être de deux sortes :

- automatiques ;
- manuels.

Ils sont déclenchés par l'arrivée d'évènements.

La gestion des traitements sert à identifier les fonctionnalités selon une approche qui va du général au particulier et qui définit leur découpage et leur enchaînement.

# Une approche par niveaux

Pour la conception d'un SI, il est nécessaire de considérer quatre niveaux d'étude :

- Le niveau conceptuel.
- Le niveau organisationnel.
- Le niveau logique.
- Le niveau physique.

## 1. Le niveau conceptuel

Le niveau conceptuel consiste à concevoir le SI en faisant abstraction de toutes les contraintes techniques ou organisationnelles et cela tant au niveau des données que des traitements.

Le niveau conceptuel répond à la question Quoi ? (le quoi faire, avec quelles données).

Le formalisme Merise employé sera :

- Le **M**odèle **C**onceptuel des **D**onnées (MCD).
- Le **M**odèle **C**onceptuel des **T**raitements (MCT).

## 2. Le niveau organisationnel

Le niveau organisationnel a comme mission d'intégrer dans l'analyse les critères liés à l'organisation étudiée. Le niveau organisationnel fera préciser les notions de temporalité, de chronologie des opérations, d'unité de lieu, définira les postes de travail, l'accès aux bases de données...

Les questions posées, au niveau des traitements, sont :

- Qui ?
- Où ?
- Quand ?

Le formalisme Merise employé sera :

- Le **M**odèle **O**rganisationnel des **D**onnées (MOD).
- Le **M**odèle **O**rganisationnel des **T**raitements (MOT).

## 3. Le niveau logique

Le niveau logique est indépendant du matériel informatique, des langages de programmation ou de gestion des données. C'est la réponse à la question Avec quoi ?

Le formalisme sera :

- Le **M**odèle **L**ogique des **D**onnées (MLD).
- Le **M**odèle **L**ogique des **T**raitements (MLT).

## 4. Le niveau physique

Le niveau physique permet de définir l'organisation réelle (physique) des données. Il apporte les solutions techniques, par exemple sur les méthodes de stockage et d'accès à l'information. C'est la réponse au Comment ?

Le formalisme employé sera :

- Le **M**odèle **P**hysique des **D**onnées (MPD).
- Le **M**odèle **O**opérationnel et **p**hysique des **T**raitements (MOpT).

## 5. Tableau récapitulatif

Niveaux	Données	Traitements
Conceptuel	Modèle Conceptuel des Données	Modèle Conceptuel des Traitements
Organisationnel	Modèle Organisationnel des Données	Modèle Organisationnel des Traitements
Logique	Modèle Logique des Données	Modèle Logique des traitements
Physique	Modèle Physique des Données	Modèle Opérationnel et Physique des Traitements

## Les apports de Merise

La force de la méthode Merise est de structurer les besoins des décideurs de façon simple et compréhensible. Merise améliore la communication entre les différents acteurs du processus de développement. Cette méthode, grâce à ses modèles, encadre le projet et de ce fait protège les intervenants d'un possible développement hors sujet.

Suivre ce cheminement intellectuel peut aussi aider l'entreprise à mieux se connaître, mieux se comprendre et ainsi mieux communiquer.

Le projet Merise s'articule autour d'un schéma directeur qui détermine et planifie le projet et ses enchaînements.

# Des données aux dépendances fonctionnelles

Pour être traitées de manière informatisée, les données doivent être décrites dans un formalisme compris par le système informatique qui va les gérer. Voici les formats génériques utilisés :

- Le type alphabétique (rien que des caractères).
- Le type alphanumérique (des caractères, des chiffres...).
- Le type numérique (les nombres).
- Le type date.
- Le type logique (0-1, Vrai-Faux, Oui-Non).

Suite à l'interview et la collecte des documents il est nécessaire de centraliser toutes les informations et règles de gestions (calcul d'un taux de remise par exemple) au sein d'un document. Ce document se nomme le dictionnaire des données.

## 1. Le dictionnaire des données

Le dictionnaire des données est un document qui permet de recenser, de classer et de trier toutes les informations (les données) collectées lors des entretiens ou de l'étude des documents. Le dictionnaire peut être plus ou moins élaboré selon le niveau de granularité souhaité. En voici un exemple :

Nom de la donnée	Format	Longueur	Type		Règle de calcul	Règle de gestion	Document
			Élémentaire	Calculé			

### Nom de la donnée

Cette cellule recevra une donnée par exemple : Nom client.

### Format

Ici sera indiqué le format de la donnée, par exemple : alphabétique.

### Longueur

La longueur approximative ou exacte de la donnée sera indiquée, par exemple : 30.

### Type

Une croix sera inscrite dans la colonne pour indiquer si la donnée est élémentaire ou calculée.

### Règle de calcul

Ici sera indiquée de manière claire la formule ou le calcul nécessaire à appliquer pour obtenir la donnée.

### Règle de gestion

Dans cette zone sera indiquée, si nécessaire, la règle de gestion inhérente à la donnée.

### Document

La rubrique document permet de saisir le document dans lequel a été trouvée la donnée.

Voici ce que pourrait être le dictionnaire :

Nom de la donnée	Format	Longueur	Type		Règle de calcul	Règle de gestion	Document
			E	C			
Nom client	Alphabétique	30	X				Facture

Le nom est au format alphabétique, d'une longueur de 30 caractères, de type élémentaire, il n'y a aucune règle de gestion et le document dans lequel l'information a été trouvée est la facture.

➤ La longueur du champ nom a été définie aléatoirement à 30 caractères. Il faut toujours avoir à l'esprit que dans le doute il vaut mieux surdimensionner les tailles. Le proverbe qui s'adapte à la situation est « Qui peut le plus peut le moins ».

Exemple :

Suite à une demande d'un membre de notre famille, président d'une association, nous devons établir le dictionnaire des données de la gestion des adhérents.

Voici une représentation d'une fiche d'adhérent :

**Association des Palanges**

**Fiche Adhérent**

Numéro 66  
 Nom : BAPTISTE  
 Prénom : Jean-Luc  
 Adresse : Rue de la forêt  
 Code Postal : 12000  
 Ville : Rodez  
 Téléphone : 05-65-42-00-00  
 Mail : jeanluc.baptiste@btsig.org  
 Date d'adhésion : 20 décembre 2007

À la lecture de la fiche, nous pouvons déterminer la présence de neuf informations différentes :

- Le numéro de l'adhérent.
- Le nom.
- Le prénom.
- L'adresse.

- Le code postal.
- La ville.
- Le téléphone.
- Le mail.
- La date d'adhésion.

Voici le dictionnaire des données :

Nom	Format	Longueur	Type		Règle de calcul	Règle de gestion	Document
			E	C			
Numéro	Numérique		X				Fiche
Nom	Alphabétique	30	X				//
Prénom	Alphabétique	30	X				//
Adresse	Alphabétique	50	X				//
Code Postal	Alphanumérique	10	X				//
Ville	Alphabétique	50	X				//
Téléphone	Alphanumérique	15	X				//
Mail	Alphanumérique	50	X				//
Date d'adhésion	Date		X				//



Le code postal est alphanumérique et de taille 10. Certaines personnes peuvent considérer qu'il serait plus judicieux de placer le format en numérique. Or qu'est-ce qui prouve que dans certains pays la règle d'écriture des code postaux est identique à la règle française des 5 chiffres ? En effet, certains pays mélangent des chiffres et des lettres. Le format alphanumérique est le plus approprié dans ce cas-là. De manière générale, il est souhaitable de ne formater en numérique que les champs sur lesquels il va y avoir des calculs. Le raisonnement appliqué est le même pour le champ téléphone.

# Les dépendances fonctionnelles

Le rôle de l'établissement des dépendances fonctionnelles est de nous aider à comprendre les liens existants entre chaque donnée. Cette démarche de recherche des dépendances fonctionnelles est la pierre angulaire de toute l'analyse des données. En effet, cette activité étant la première dans l'élaboration de l'analyse, si elle est négligée c'est tout l'ensemble qui en subira les conséquences.

## Définition

Une donnée B dépend fonctionnellement (ou est en dépendance fonctionnelle) d'une donnée A lorsque la connaissance de la valeur de la donnée A nous permet la connaissance **d'une et au maximum une seule** valeur de la donnée B.

*Par exemple :*

*La connaissance de la valeur d'un numéro de client nous permet de connaître sans ambiguïté la valeur **d'un et d'un seul** nom de client.*

*Dans la fiche d'adhérent, l'adhérent numéro 1 a pour nom Baptiste.*

## Formalisme

Le formalisme de représentation d'une dépendance fonctionnelle est le suivant :

Numéro adhérent  $\rightarrow$  (Nom adhérent, prénom, adresse, code postal, ville, téléphone, mail, date d'adhésion)



Numéro adhérent sera appelé la clé de la relation ou clé primaire ou encore identifiant de la relation.

La partie gauche de la dépendance fonctionnelle (ici Numéro adhérent) est aussi appelée **source** de la dépendance fonctionnelle. La partie droite de la dépendance fonctionnelle est appelée le **but** de la dépendance fonctionnelle.

## 1. Dépendances fonctionnelles composées

Une dépendance fonctionnelle qui comporte plusieurs attributs est dite composée.

*Voici un exemple de dépendance fonctionnelle composée :*

*(Numéro Coureur, Numéro course)  $\rightarrow$  (temps)*

### Interprétation

Connaissant le numéro du coureur et le numéro de la course, nous connaissons de façon certaine le temps chronométré d'un coureur précis sur une course précise.

*Autre exemple :*

*(Code athlète, code sport)  $\rightarrow$  (année de pratique)*

### Interprétation

Connaissant le code de l'athlète et le code du sport nous pouvons connaître de façon sûre et unique le nombre d'années de pratique. Comme nous pouvons le constater la seule connaissance du code d'athlète ne nous permet pas de connaître le nombre d'années de pratique, de la même manière la seule connaissance du code du sport ne permet pas la connaissance pleine et entière des années de pratique. Structurellement, il est nécessaire d'avoir les deux informations : le code de l'athlète et le code du sport, pour pouvoir connaître les années de pratique d'un sport précis par un athlète précis.

## 2. Dépendance fonctionnelle élémentaire

Une dépendance fonctionnelle  $A \rightarrow B$  est élémentaire s'il n'existe pas une donnée C, sous-ensemble de A, décrivant une dépendance fonctionnelle de type  $C \rightarrow B$



Par exemple :

*RéférenceProduit* → *Désignation*

*NuméroCommande, RéférenceProduit* → *Quantité*

*NuméroCommande, RéférenceProduit* → *Désignation*

La première dépendance fonctionnelle est correcte car ayant deux rubriques elle est élémentaire.

La deuxième dépendance fonctionnelle est correcte également car la connaissance d'un numéro de commande et d'une référence produit nous permet de connaître la quantité commandé du produit. Elle est aussi élémentaire car c'est la connaissance du couple (*NuméroCommande, RéférenceProduit*) et pas seulement d'un des éléments qui permet la connaissance de la quantité.

La troisième dépendance fonctionnelle n'est pas élémentaire car il existe à l'intérieur d'elle *RéférenceProduit* → *Désignation* qui était déjà une dépendance fonctionnelle élémentaire. Pour connaître la *Désignation*, *NuméroCommande* est dans ce cas superflu.

### 3. Dépendance fonctionnelle élémentaire directe

On dit que la dépendance fonctionnelle  $A \rightarrow B$  est directe s'il n'existe aucun attribut C tel que l'on puisse avoir  $A \rightarrow C$  et  $C \rightarrow B$ . En d'autres termes, cela signifie que la dépendance fonctionnelle entre A et B ne peut pas être obtenue par transitivité.

Exemple :

*NumClasse* → *NumElève*

*NumEleve* → *NomElève*

*NumClasse* → *NomElève*

La troisième dépendance fonctionnelle n'est pas directe car nous pourrions écrire :

*NumClasse* → *NumElève* → *NomElève*

### 4. Méthodologie d'élaboration des dépendances fonctionnelles

L'élaboration des dépendances fonctionnelles est réalisée à l'aide du dictionnaire des données. La démarche consiste à rechercher :


- les dépendances fonctionnelles formées par deux rubriques, élémentaires et directes ;
- les dépendances fonctionnelles composées.

## Cas pratique

Monique, sa fille Rachel et son gendre Marc gèrent un camping dans les Pyrénées orientales. Le camping est ouvert du 1er juin au 30 septembre. Ils disposent de cinquante emplacements sur un terrain d'une superficie totale de quarante hectares.

Ils sont équipés d'un logiciel spécialisé dans la réservation des emplacements qui fonctionne très bien mais qui ne permet pas de gérer les achats de l'épicerie ou du bar selon leurs règles de gestion. En effet, les vacanciers ne payent leurs achats qu'à la fin de leur séjour. Concrètement, les achats sont inscrits manuellement sur une fiche bristol créée pour chaque famille de vacanciers. À la fin du séjour, les cumuls sont réalisés et une facture manuelle concernant les achats est établie. Les propriétaires du camping souhaiteraient disposer d'un logiciel permettant d'automatiser la création de la facture grâce à la saisie journalière des achats.

Voici une représentation de la fiche bristol :



# Camping de la source

Liste des Achats

Nom :  
Prénom :  
Adresse :  
Code Postal :  
Ville :  
Téléphone :

BAPTISTE  
Jean-Luc  
Rue de la forêt  
12000  
Rodez  
05-65-42-00-00

Date	Désignation	Qté	Prix	Total
14/7/08	Repas « Cargolade »	4	22	88
15/7/08	Café	1	1,20	1,20
15/7/08	Glace « Magnum »	2	2,10	4,20
16/7/08	Baguette	1	1,15	1,15
Total dû :				94,55

## Résolution du cas

À la lecture de l'énoncé, nous devons déterminer et séparer les informations mémorisables des informations décrivant le contexte.

Les prénoms des propriétaires du camping sont-ils des informations stockables ou des informations d'ordre général ? Si nous analysons la demande d'informatisation ces données ne font pas partie du système d'information.

Il en est de même pour les dates d'ouverture, de fermeture, le nombre d'emplacements ou la superficie du camping.

Il paraît évident que nous devons nous intéresser à l'élément de base, c'est-à-dire la fiche bristol. C'est elle qui contient les informations indispensables à l'élaboration de la facture finale.

Nous pouvons y trouver le nom de la famille, son adresse, la liste des articles achetés, leur prix unitaire, la quantité, le total. Il va être nécessaire de rajouter deux informations non présentes : le numéro du client et le code de l'article.

## 1. Dictionnaire des données

Voici un dictionnaire des données qui pourrait être élaboré suite à la lecture de l'énoncé :

Nom	Format	Longueur	Type		Règle de calcul	Règle de gestion	Document
			E	C			
NumCli	Numérique		X				Bristol
Nom	Alphabétique	30	X				//
Prénom	Alphabétique	30	X				//
Adresse	Alphabétique	50	X				//
Code Postal	Alphanumérique	10	X				//
Ville	Alphabétique	50	X				//
CodeArticle	Alphanumérique	15	X				//
Désignation	Alphabétique	50	X				//
PrixUnitaire	Numérique		X				//
Qté	Numérique		X				//
Date	Date		X				//
TotalLigne	Numérique			X	PrixUnitaire x Qté		//
TotalFacture	Numérique			X	Somme des TotalLigne		//

Le dictionnaire des données recense l'ensemble des informations. Comme nous pouvons le constater certaines informations seront déduites (ou calculées) en fonction d'informations élémentaires. C'est le cas du TotalLigne qui est le résultat de la multiplication du prix unitaire du produit et de sa quantité et du TotalFacture qui est la somme des TotalLigne. Ces deux informations sont utiles pour le développeur de l'application qui mettra en œuvre les procédures de calculs a posteriori. Dans le cycle de modélisation Merise ces deux informations sont des données déduites et non stockables, elles n'apparaîtront donc pas dans la suite du processus.

## 2. Détermination des dépendances fonctionnelles ou DF

À la lecture du dictionnaire nous pouvons déduire deux groupes d'informations distinctes. Un groupe caractérise les clients, l'autre les produits.

### Dépendances fonctionnelles pour les clients

Posons-nous la question :

« Quand je connais le numéro du client, est-ce que je connais de façon sûre et unique le nom du client ? ». Si la réponse est « oui » alors voici la transcription de la **DF** :

Numcli → Nom

Voici maintenant l'ensemble des DF élémentaires :

Numcli → Prénom

Numcli → Adresse

Numcli → Code Postal

Numcli → Ville

Dépendances fonctionnelles pour les articles :

CodeArticle → Désignation

CodeArticle → PrixUnitaire



Les DF auraient pu s'écrire de la façon suivante : Numcli → (Nom, Prénom, Adresse, Code Postal, Ville), CodeArticle → (Désignation, PrixUnitaire).

Intéressons-nous à la donnée Qté : est-ce que la connaissance du code de l'article nous permet de connaître de façon sûre et unique une quantité ?

Autrement dit :

Connaissant « 567Nut » nous pouvons connaître de façon sûre et unique la quantité « 4 » ?

Nous nous rendons compte que cette donnée Qté fait partie d'une dépendance fonctionnelle composée.

Voici une proposition :

(Numcli, CodeArticle, Date) → Qté

Et maintenant si nous nous posons la question :

« Connaissant le code du client, le code de l'article et la date d'achat puis-je connaître de façon sûre et unique la quantité achetée ? ».

Il est évident que la réponse est oui !

Voilà, nous venons de définir l'ensemble des dépendances fonctionnelles concernant notre cas.

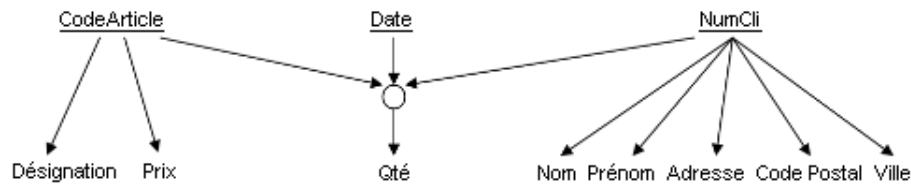
### Rappel

Les dépendances fonctionnelles ne concernent que les données non déduites. C'est pour cela que n'apparaissent pas les données concernant le total par ligne et le total global de la facture qui sont des informations déduites par calcul.

## 3. Graphe des dépendances fonctionnelles

Le graphe des dépendances fonctionnelles est une étape intéressante car il épure le dictionnaire en ne retenant que les données non déduites et élémentaires et il permet une représentation spatiale de ce que sera le futur modèle conceptuel des données.

Voici le graphe des dépendances fonctionnelles concernant le camping :



## 4. Matrice des dépendances fonctionnelles

Une autre façon de représenter les dépendances fonctionnelles est de créer une matrice. Cependant, cette représentation ne présente pas le même intérêt que le graphe, qui lui permet une vision plus graphique du futur modèle conceptuel des données.

Elle se présente sous forme d'un tableau ayant pour entrées l'ensemble des données du dictionnaire.

Les en-têtes de lignes sont les données sources des dépendances fonctionnelles

Les en-têtes de colonnes sont les données buts des dépendances fonctionnelles

Le tableau est parcouru colonne par colonne, et pour chaque colonne ligne par ligne.

À chaque étape la question suivante doit être posée : la donnée source est-elle en dépendance fonctionnelle avec la donnée but ? En cas de réponse positive, nous inscrivons un « 1 » dans la case d'intersection.

Exemple :

But		Sources											
		1	2	3	4	5	6	7	8	9	10	11	12
1	Numcli												
2	Nom	1											
3	Prénom	1											
4	Adresse	1											
5	Code Postal	1											
6	Ville	1											
7	CodeArticle												
8	Désignation							1					
9	Prix							1					
10	Date												
11	Qté												1
12	Numcli, CodeArticle, Date												

Une version simplifiée consiste à ne laisser que les colonnes sources ayant un « 1 » d'inscrit.

Exemple :

But		Sources											
		1	2	3	4	5	6	7	8	9	10	11	12

But		1	7	12
1	Numcli			
2	Nom	1		
3	Prénom	1		
4	Adresse	1		
5	Code Postal	1		
6	Ville	1		
7	CodeArticle			
8	Désignation		1	
9	Prix		1	
10	Date			
11	Qté			1
12	Numcli, CodeArticle, Date			

## Conclusion

Il est impératif de bien comprendre et bien maîtriser ces notions de dépendances fonctionnelles, car elles sont les fondations des modèles Merises qui vont suivre. Donc, il est nécessaire de passer du temps à bien les définir pour éviter les erreurs de conception plus tard.

# Introduction au Modèle Conceptuel des Données

Le Modèle Conceptuel des Données introduit la notion d'entités, de relations et de propriétés. Nous allons commencer par voir certains aspects « théoriques » avant de plonger dans la pratique. Il décrit de façon formelle les données utilisées par le système d'information. La représentation graphique, simple et accessible, permet à un non-informaticien de participer à son élaboration. Les éléments de base constituant un modèle conceptuel des données sont :

- les propriétés ;
- les entités ;
- les relations.

## 1. Les propriétés

Les propriétés sont les informations de base du système d'information.

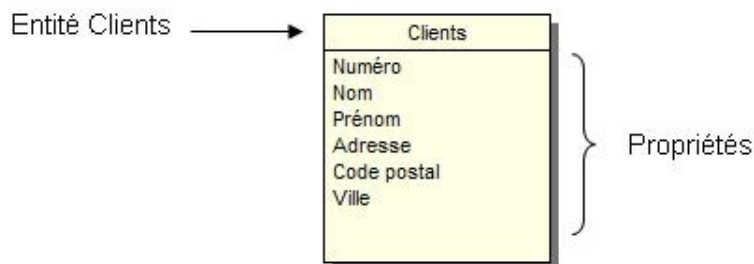
Un client possède un numéro de client, un nom, un prénom, habite à une adresse précise, etc. Ces informations élémentaires essentielles sont des propriétés.

Les propriétés disposent d'un type. Elles peuvent être numériques, représenter une date, leur longueur peut être aussi définie. Par exemple : le nom est une propriété de type alphabétique et de longueur 50, c'est-à-dire que la valeur saisie ne comportera aucun chiffre et ne dépassera pas cinquante caractères.

Les types ne sont pas décrits au niveau conceptuel, car ce niveau est trop proche de la définition du système physique. Nous y reviendrons plus tard.

## 2. Les entités ou objets

Comme il est aisé de le constater, les clients sont définis par certaines propriétés (numéro, nom, prénom...). Le fait de les regrouper amène naturellement à créer une entité Clients. Le symbolisme retenu est le suivant :



### a. L'identifiant

Une de ces propriétés a un rôle bien précis, c'est l'identifiant nommé aussi la clé.

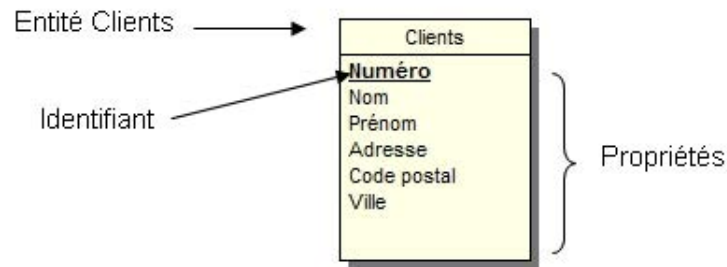
L'identifiant permet de connaître de façon sûre et unique l'ensemble des propriétés qui participent à l'entité. Par exemple, le fait de connaître la ville d'un client permet-il de connaître son nom ? La réponse est non. La connaissance du nom du client permet-elle de connaître sa ville ? La réponse est toujours non, car en cas d'homonymie la confusion entre un Durand Max et un Durand Raymond est totale.

Il faut donc trouver, ou inventer, une propriété qui lorsque sa valeur est connue permet la connaissance de l'ensemble des valeurs qui s'y rattachent de façon formelle.

Ainsi, lorsque le numéro du client est connu, son nom, son prénom et toutes les valeurs des autres propriétés qui s'y rattachent sont connues de façon sûre et unique.

Au niveau du formalisme, cette propriété se souligne. Voici le schéma modifié de l'entité **Clients**.





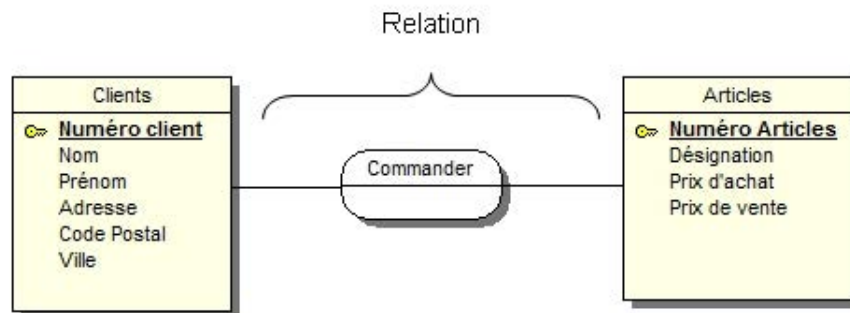
### 3. Les relations ou associations

Nous avons vu que les entités regroupaient un ensemble d'informations élémentaires. Les entités sont souvent liées entre elles.

*Par exemple :*

*Un client peut commander des articles.*

Si nous analysons cette phrase, on distingue deux entités (clients et articles) et un verbe (commander) qui indique un lien entre clients et articles. Formalisons cette phrase avec Merise.



Voilà la première étape, première car la lecture du schéma doit être améliorée en incorporant une notion importante : **les cardinalités**.

#### a. Les cardinalités

Elles expriment le nombre de fois ou l'occurrence d'une entité participe aux occurrences de la relation.

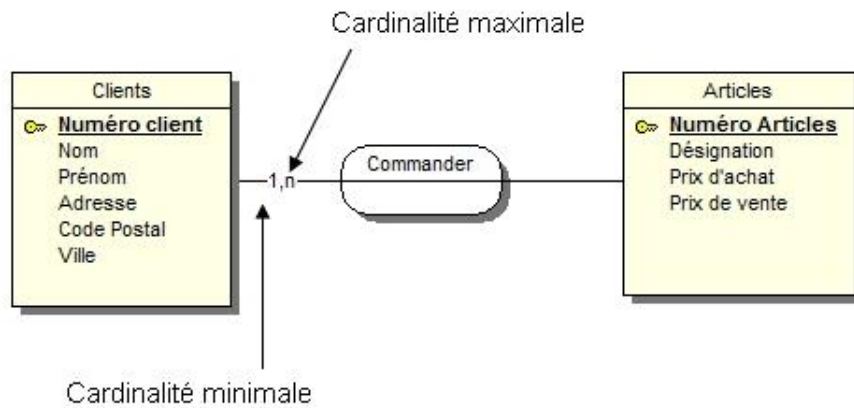
Dans notre exemple on peut se poser les questions suivantes :

- Combien de fois au minimum un client peut-il commander un article ?
- Combien de fois au maximum un client peut-il commander un article ?

À la première question, nous pouvons répondre qu'un client, pour être client, doit commander au moins un article.

À la deuxième question, nous pouvons répondre qu'un client peut commander plusieurs articles.

Voici comment symboliser cet état :



Le n représente la notion de « plusieurs » ; ici nous avons représenté le fait qu'un client peut commander un ou plusieurs articles. Il faut que nous nous posions les mêmes questions pour l'article :

- Combien de fois au minimum un article peut-il être commandé par un client ?
- Combien de fois au maximum un article peut-il être commandé par un client ?

Pour le minimum, nous pouvons l'interpréter de la façon suivante :

A-t-on des articles qui ne peuvent jamais être commandés ?

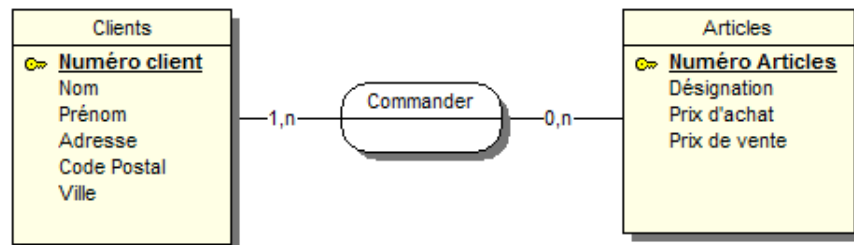
Si nous répondons oui dans ce cas la cardinalité minimale est 0.

Pour le maximum :

A-t-on des articles qui peuvent être commandés plusieurs fois ?

Nous pouvons espérer que oui, dans ce cas la cardinalité maximale sera n.

Voici le schéma finalisé :



### Définitions

La cardinalité minimale (0 ou 1) exprime le nombre de fois minimum qu'une occurrence d'une entité participe aux occurrences d'une relation.

La cardinalité maximale (1 ou n) exprime le nombre de fois maximal qu'une occurrence d'une entité participe aux occurrences de la relation.

➤ Si le maximum est connu, il faut inscrire sa valeur. Par exemple, si dans les règles de gestion le client n'a le droit de commander qu'un maximum de 3 articles en tout et pour tout, dans ce cas-là les cardinalités s'exprimeront de cette façon : 1,3.

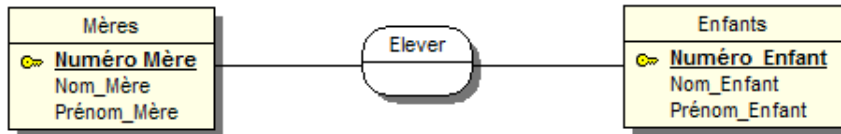
### Autre exemple :

Modélisons le fait qu'une mère élève des enfants.

Nous avons deux entités Mères et Enfants :



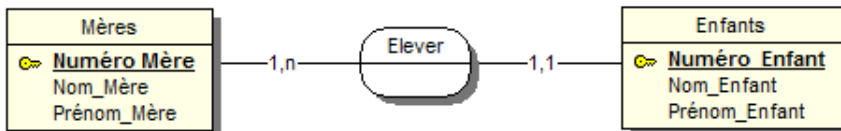
Une relation Elever :



Des cardinalités :

Une mère peut élever un ou plusieurs enfants.

Un enfant peut être élevé par une et une seule mère.



➤ Bien sûr, tout est question d'interprétation. Au sein d'une équipe de développement, il peut y avoir des divergences de point de vue. Pour les cardinalités, il faut être le plus logique possible, se référer aux règles de gestion édictées par le commanditaire de l'application et se rappeler la maxime suivante : "Qui peut le plus peut le moins".

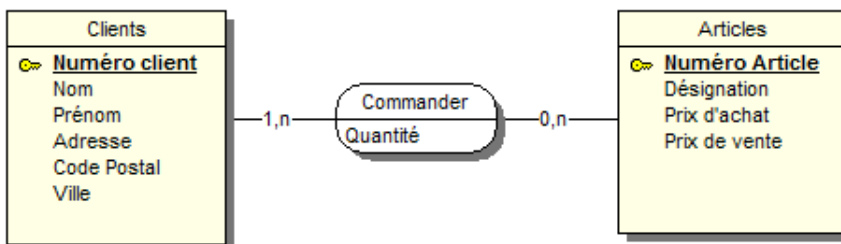
## b. Les relations porteuses

### Définition

Une relation est dite porteuse lorsqu'elle contient des propriétés.

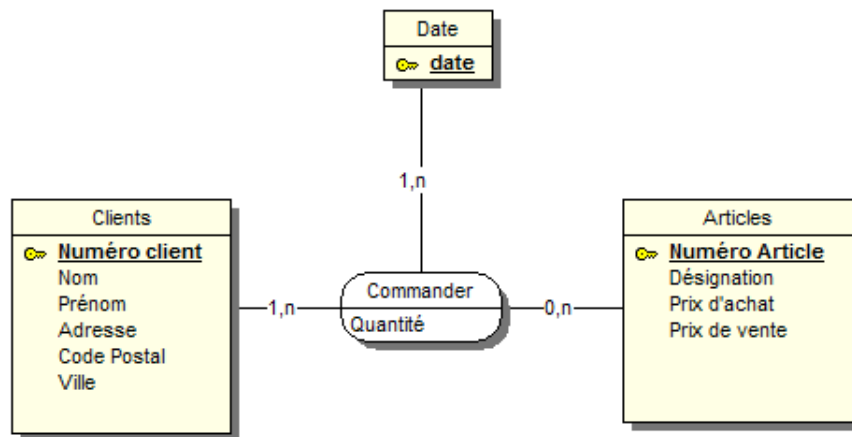
Imaginons que l'on veuille connaître la quantité d'articles commandés par clients, nous nous rendons compte qu'il faut utiliser une nouvelle propriété Quantité. Cette nouvelle propriété dépend de clients, d'articles ou des deux ? La bonne réponse est que Quantité dépend des deux entités.

Voici le modèle conceptuel correspondant :



Nous pouvons interpréter ce schéma de la façon suivante : Le client X a commandé la quantité Y d'articles Z. Si nous désirons connaître la date d'achat, il nous suffit de créer une entité Date à la relation Commander.

➤ Une relation faisant intervenir deux entités est dite **binaire**.



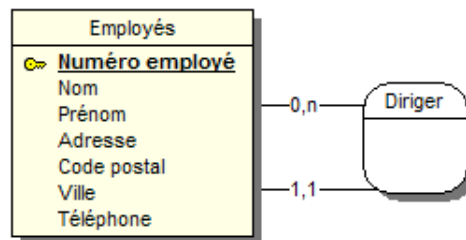
Une relation faisant intervenir trois entités est dite **ternaire**. Dans certains ouvrages elle est caractérisée par l'appellation « Tri-pattes ».

### c. Les relations réflexives

#### Définition

Une relation réflexive est une relation d'une entité sur elle-même.

Par exemple, on désire modéliser le fait qu'un employé peut diriger d'autres employés.



À la lecture de ce schéma, nous interprétons donc qu'un employé peut diriger zéro ou plusieurs personnes et qu'un employé est dirigé par un et un seul autre employé.

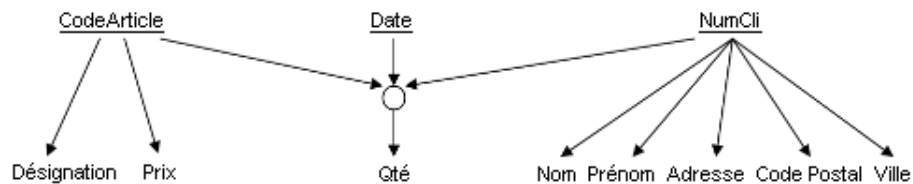
## 4. Règles d'usages

- Toute entité doit comporter un identifiant.
- Toutes les propriétés de l'entité dépendent fonctionnellement de l'identifiant. C'est-à-dire que connaissant la valeur de l'identifiant, nous connaissons de façon sûre et unique la valeur des propriétés associées. Si nous recherchons le client numéro 5, nous devons récupérer le nom et le prénom du client numéro 5 et pas ceux d'une autre personne.
- Le nom d'une propriété ne doit apparaître qu'une seule fois dans le modèle conceptuel des données. Si nous établissons une entité Clients et une nommée Prospects, nous ne devons pas retrouver la propriété Nom dans les deux entités. Il faut préférer la dénomination suivante Nom\_client et Nom\_prospect.
- Les propriétés résultantes d'un calcul ne doivent pas apparaître dans le modèle conceptuel des données.

#### Cas pratique

Reprenons le graphe concernant le cas du camping vu au chapitre précédent et réalisons le modèle conceptuel qui en découle.

## Le graphe



## Le MCD pas à pas

Commençons par déterminer les entités. Par rapport au graphe, nous pouvons remarquer trois sources de dépendances fonctionnelles :

- CodeArticle
- Date
- NumCli

Chacune de ces sources peut représenter une entité :

- Articles
- Date
- Clients

Voici la représentation graphique des entités :

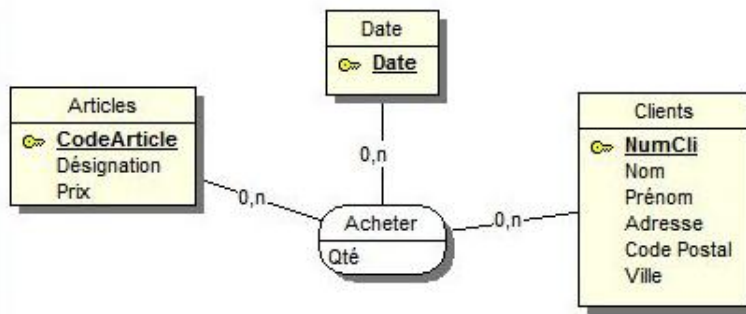


Renseignons maintenant les entités avec leurs propriétés respectives :

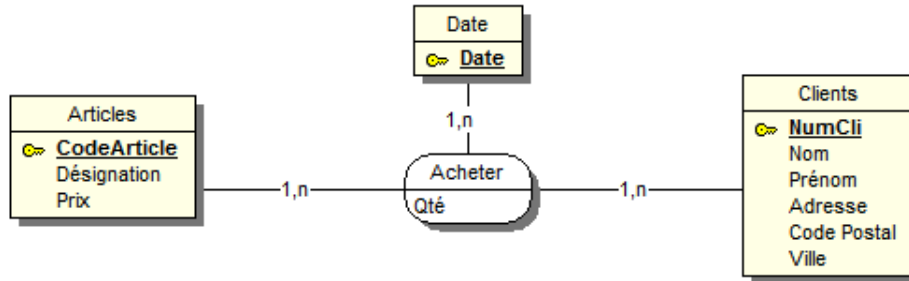


## Traçons les relations

Nous savons qu'une quantité d'articles est achetée par un client à une date donnée. Nous voyons qu'il existe une relation entre les trois entités. La voici modélisée :



Examinons les cardinalités. Sur le graphique ci-dessus nous pouvons interpréter qu'au minimum il ne se vend strictement rien. En pratique, nous pouvons imaginer qu'il ne se passe pas une journée sans que se produise la vente d'un article à un client. En voici le modèle conceptuel revu :



Maintenant, nous pouvons lire qu'au minimum un article est acheté dans une certaine quantité, par au minimum un client à une date donnée. Voilà qui nous rapproche plus de la réalité.

## 5. Notion d'entité forte et d'entité faible

### a. Entité forte

#### Définition

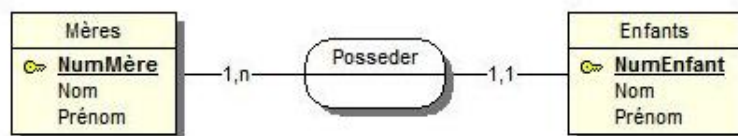
Une entité forte est une entité qui, disposant de son identifiant, peut être considérée de façon isolée.

### b. Entité faible

#### Définition

Une entité faible est une entité qui ne peut être considérée qu'en association avec une autre entité.

Voici un exemple :



Dans ce cas l'entité forte est l'entité Mères et l'entité faible est l'entité Enfants.

## 6. Notion de contrainte d'intégrité fonctionnelle

#### Définition

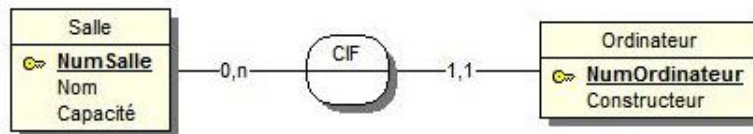
Une contrainte d'intégrité fonctionnelle (ou CIF) est définie par le fait qu'une des entités de l'association est complètement déterminée par la connaissance d'une ou de plusieurs entités participant à cette même association.

Par exemple :



Nous pouvons lire qu'une salle peut contenir zéro ou plusieurs ordinateurs et qu'un ordinateur existe dans une et une seule salle. Dans le cas d'une association **binaire** comme celle-ci, une contrainte d'intégrité fonctionnelle existe à partir du moment où une cardinalité de type **1,1** existe.

Certains auteurs proposent une écriture de ce type :



Mon conseil est de nommer votre relation de façon claire et compréhensible pour donner tout son sens à la lecture de votre modèle conceptuel.

## 7. Notion d'identifiant relatif

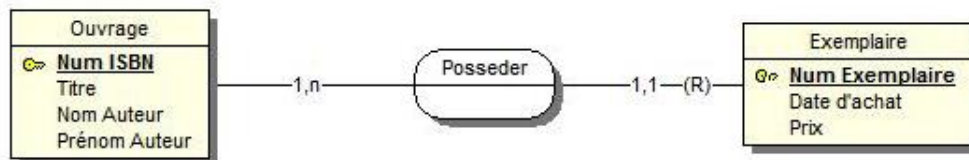


Les identifiants relatifs font partie des extensions Merise/2, je vous les présente volontairement dans ce chapitre.

Certaines entités ont une existence totalement dépendante d'autres entités. Dans ce cas nous avons recours à un identifiant relatif.

Par exemple :

*Nous devons gérer les exemplaires des livres commandés dans une bibliothèque :*



Nous pouvons considérer qu'une bibliothèque achète plusieurs exemplaires du même livre. Doit-elle numéroter les exemplaires dans l'ordre séquentiel croissant c'est-à-dire de 1 à 6 000 ou pour chaque ouvrage de 1 au nombre maximal de **cet ouvrage** présent dans la bibliothèque ?

Il paraît évident que la solution la plus judicieuse est la seconde proposition. Ainsi, la clé d'identification de l'exemplaire sera la concaténation du Num ISBN et du Num Exemplaire. L'identification d'un identifiant relatif se traduit par le symbole **(R)** à côté de la cardinalité.

Pour reconnaître un identifiant relatif, nous pouvons aussi nous poser la question « si je supprime l'ouvrage X de ma bibliothèque, dois-je supprimer tous les exemplaires ? ». La réponse est bien entendu oui. Nous pouvons donc dire que l'existence même de l'entité Exemplaire dépend de l'existence de l'entité Ouvrage.

Voici un autre exemple encore plus évident :

Vous êtes le responsable informatique d'un centre de formation possédant un campus sur lequel sont construits 3 bâtiments strictement identiques :

Ils possèdent 3 étages et chaque étage héberge 12 salles informatiques dans lesquelles peuvent être installés 10

ordinateurs.

Nous pouvons en conclure que dans chaque bâtiment, nous pouvons installer 360 ordinateurs et la capacité du campus est de  $3 \times 360$  ordinateurs soit 1080.

Imaginons le scénario suivant, vous recevez une dotation de 1080 ordinateurs à répartir dans toutes les salles. En outre, vous devez les numéroté pour les identifier.

Quelle solution choisissez-vous ?

Une solution serait de les numéroté de 1 à 1080.

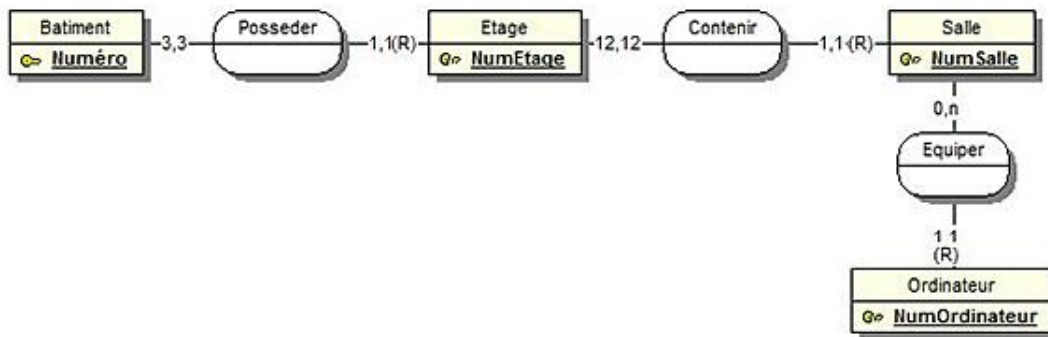
Une autre serait de rendre la numérotation relative à la salle.

Par exemple dans la salle 1 du premier étage du bâtiment A la numérotation serait de 1-1 à 1-10, dans la salle 2 la numérotation irait de 2-1 à 2-10.

Maintenant, il serait judicieux de rendre la salle relative à l'étage. La numérotation deviendrait E1-1-1, pour le premier ordinateur de la salle 1 de l'étage 1, ou E3-12-10 pour le dixième ordinateur de la salle 12 de l'étage 3.

Vous venez de comprendre que maintenant nous pourrions rendre les étages relatifs aux bâtiments et numéroté les ordinateurs de la façon suivante : BA-E3-12-10. Ainsi, nous savons que nous avons affaire à l'ordinateur 10 de la salle 12 de l'étage 3 du bâtiment A. Si les ordinateurs peuvent être changés de lieu rien qu'en regardant leur identifiant, vous pouvez les réinstaller dans leurs salles respectives.

Voici le modèle conceptuel découlant de cet exemple :



Imaginons que le centre de formation décide de vendre un bâtiment, tout disparaît en cascade !



# Conception d'un Modèle Conceptuel des Données pas à pas

Pas à pas, nous allons créer un Modèle Conceptuel des Données en partant d'un cas fictif.

Votre oncle, restaurateur, vous demande de lui réaliser un logiciel de gestion des commandes de repas. Voici les indications qu'il vous donne :

Il souhaite pouvoir gérer certaines informations concernant ses employés : nom, prénom, adresse complète, téléphone et diplômes.

Au niveau de la prise de commande, il souhaite savoir si elle porte sur le service de midi ou celui du soir et à quelle date elle a été passée.

Pour certains calculs statistiques, il souhaite aussi savoir quelle table a passé la commande et quel serveur l'a prise.

La carte du restaurant propose l'ensemble des plats d'entrées, principaux et desserts. Les menus proposés sont un assemblage des plats à la carte.

La carte des vins propose une sélection de vins qui sont stockés dans la cave du restaurant. Votre oncle désire connaître pour chaque bouteille son millésime, sa date d'achat, son prix d'achat et son prix de vente. Il voudrait saisir aussi pour chaque cru les informations concernant le viticulteur (nom, prénom, adresse complète, téléphone). À l'heure actuelle votre oncle, amoureux du vin, met sur chaque goulot de chaque bouteille une étiquette contenant le prix d'achat ainsi que la date d'achat. Votre système doit pouvoir remplacer ce traitement manuel.

Ensuite, certaines boissons comme les apéritifs, les digestifs, les sodas ou les cafés sont gérés de façon simpliste juste par leur libellé et leur prix de vente.

Chaque serveur prenant une commande saisit l'ensemble des informations sur un Pocket PC qui transmet la commande via Wifi sur un ordinateur central.

À l'aide de ces quelques informations basiques, essayons de réaliser le modèle conceptuel. Nous allons commencer par un dictionnaire des données simplifié.

## 1. Le dictionnaire des données

Nom	Format	Longueur	Type	
			E	C
Numéro employé	Numérique			
Nom employé	Alphabétique	30	X	
Prénom employé	Alphabétique	30	X	
Adresse employé	Alphabétique	40	X	
Code Postal employé	Alphanumérique	5	X	
Ville employé	Alphabétique	40	X	
Téléphone employé	Alphanumérique	15	X	
Diplômes des employés	Alphabétique	30	X	
Numéro viticulteur	Numérique		X	
Nom viticulteur	Alphabétique	30	X	
Prénom viticulteur	Alphabétique	30	X	
Adresse viticulteur	Alphabétique	40	X	
Code postal viticulteur	Alphanumérique	5	X	

Ville viticulteur	Alphabétique	40	X	
Téléphone viticulteur	Alphanumérique	15	X	
Numéro de la table	Numérique		X	
Capacité	Numérique		X	
Numéro de la boisson	Numérique		X	
Désignation de la boisson	Alphabétique	20	X	
Prix de vente de la boisson	Numérique		X	
Numéro du vin	Numérique		X	
Nom du vin	Alphabétique	30	X	
Millésime du vin	Numérique		X	
Prix de vente du vin	Numérique		X	
Numéro de la bouteille	Numérique		X	
Date d'achat de la bouteille	Date		X	
Prix d'achat de la bouteille	Numérique		X	
Numéro du menu	Numérique		X	
Libellé du menu	Alphabétique	30	X	
Prix de vente du menu	Numérique		X	
Numéro du plat	Numérique		X	
Libellé du plat	Alphabétique	30	X	
Prix de vente du plat	Numérique		X	
Numéro de type de plat	Numérique			
Désignation du type de plat (entrée, plat principal, dessert)	Alphabétique	20	X	
Type de service (midi ou soir)	Alphabétique	4	X	
Date de prise de commande	Date		X	
Numéro de la commande	Numérique		X	

Voici une ébauche du dictionnaire des données, ceci est notre base de travail. Si au fil de la conception ou de la réflexion certaines propriétés apparaissent, il suffira de les rajouter.

## 2. Les dépendances fonctionnelles

### a. Dépendances élémentaires

Numéro employé → (Nom employé, Prénom employé, Adresse employé, Code Postal employé, Ville employé, Téléphone employé).

Numéro viticulteur → (Nom viticulteur, Prénom viticulteur, Adresse viticulteur, Code postal viticulteur, Ville viticulteur, Téléphone viticulteur).

Numéro de la boisson → (Désignation de la boisson, Prix de vente de la boisson).

Numéro du vin → (Nom du vin, Millésime du vin, Prix de vente du vin).

Numéro de la bouteille → (Date d'achat de la bouteille, Prix d'achat de la bouteille).

Numéro du menu → (Libellé du menu, Prix de vente du menu).

Numéro du plat → (Libellé du plat, Prix de vente du plat).

Numéro de type de plat → Désignation du type de plat.

Numéro de la table → Capacité.

## **b. Dépendances isolées**

- Diplômes des employés.
- Type de service (midi ou soir).
- Date de prise de commande.
- Numéro de la commande.

### **Traitement des dépendances isolées**

Pour les diplômes, deux possibilités s'offrent à nous :

- Soit nous notons seulement un diplôme par employé.
- Soit nous notons tous les diplômes d'un employé.

Dans le premier cas, la résolution du problème est simple :

Numéro employé → Diplôme.

Dans le deuxième cas, nous pouvons gérer la possibilité d'avoir soit un diplôme soit plusieurs diplômes. Comme dit le proverbe : "Qui peut le plus peut le moins".

Il semble judicieux de pouvoir concevoir une analyse ne bloquant pas l'utilisateur. Nous allons donc utiliser la deuxième alternative en ajoutant même une propriété Date d'obtention :

(Numéro employé, diplôme) → Date d'obtention.

Pour la désignation du type de plat, nous pouvons être sûrs que lorsque l'on connaît un numéro de plat on connaît de façon sûre et unique un type de plat.

Numéro du plat → Type de plat.

Il nous reste :

- Type de service,
- Date de prise de commande,
- Numéro de la commande.

Nous pouvons ressentir que tout tourne autour d'un numéro de commande. En fait, une commande est prise par un serveur, concerne une table, est réalisée à une date donnée, sur un type de service et comprend des plats et/ou des menus et/ou des vins et/ou des boissons et comportera des quantités associées.

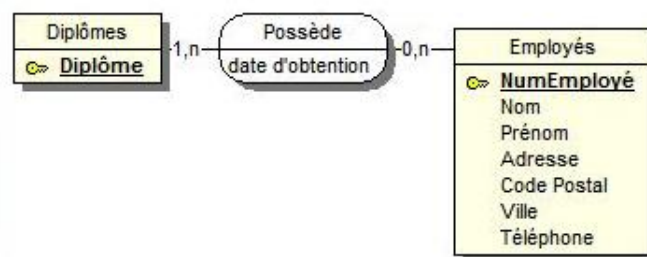
Nous pourrions essayer ceci :

Numéro de la commande → Date de prise de commande.  
 Numéro de la commande → Type de service.  
 Numéro de la commande → Numéro employé.  
 Numéro de la commande → Numéro de la table.  
 (Numéro de la commande, Numéro de la boisson) → Quantité.  
 (Numéro de la commande, Numéro du vin) → Quantité.  
 (Numéro de la commande, Numéro du plat) → Quantité.  
 (Numéro de la commande, Numéro du menu) → Quantité.

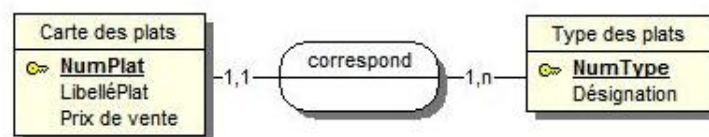
➤ La conception d'un modèle conceptuel est un processus dynamique, nous allons considérer maintenant que toutes nos dépendances fonctionnelles sont décrites et continuer à cheminer dans le projet en commençant à générer le MCD. De cette génération et réflexion, des liens peuvent apparaître, liens qui peuvent être absents de la première ébauche de dépendances fonctionnelles. Il suffit de les rajouter a posteriori.

### 3. Élaboration du Modèle Conceptuel des Données

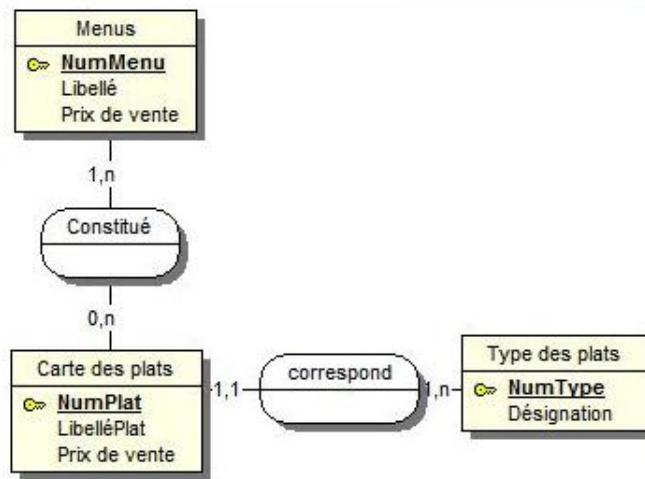
Maintenant, considérons que nos dépendances fonctionnelles sont décrites et commençons à générer pas à pas le Modèle Conceptuel des Données.



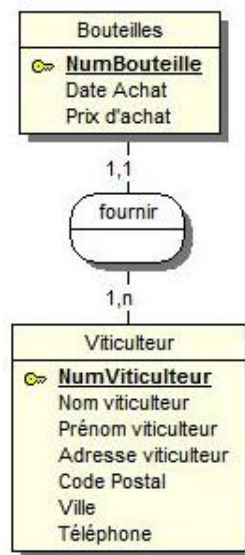
Nous pouvons dire qu'un employé peut posséder zéro ou plusieurs diplômes obtenus à une date précise, et qu'un diplôme peut être possédé par un ou plusieurs employés. Par exemple le bac pro cuisine est un diplôme pouvant être détenu par plusieurs employés.



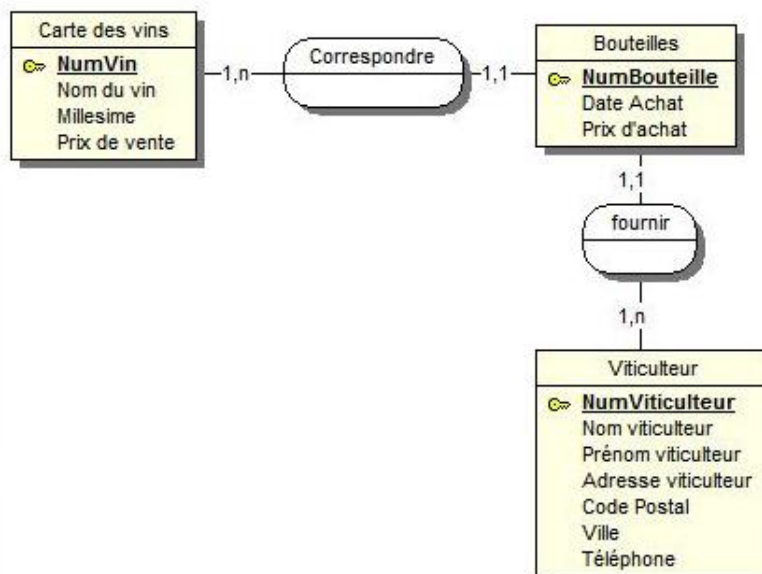
À un plat correspond un et un seul type de plat, c'est soit une entrée, soit un plat de résistance, soit un dessert. À l'inverse pour une entrée il peut y avoir un ou plusieurs plats différents.



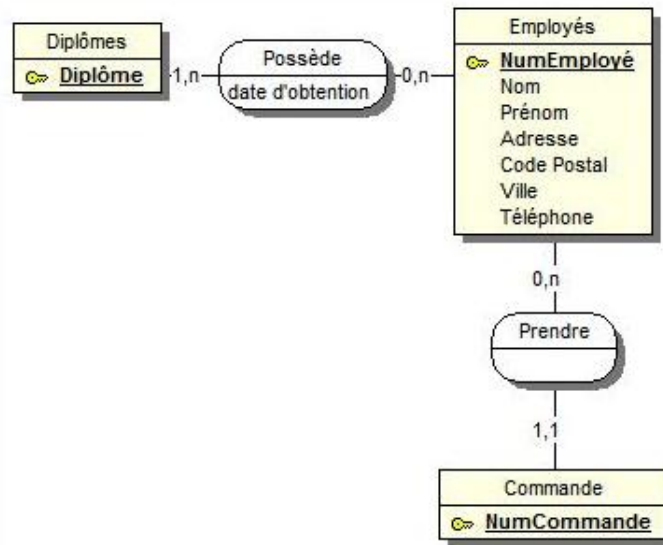
Nous voyons ici qu'un menu peut être constitué d'au minimum un plat et d'au maximum plusieurs. Par contre, un plat peut ne pas faire partie d'un menu.



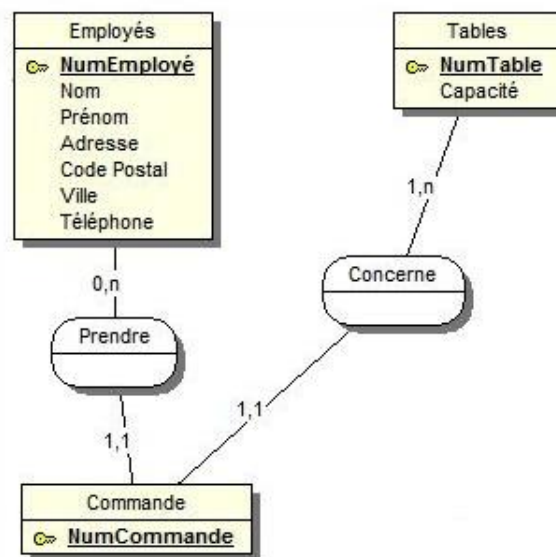
Une bouteille est fournie par un et un seul viticulteur et un viticulteur peut fournir une ou plusieurs bouteilles.



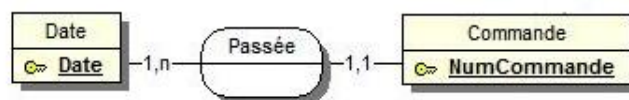
À un vin présent sur la carte des vins correspond une ou plusieurs bouteilles en stock et à une bouteille précise correspond une et une seule description de vin.



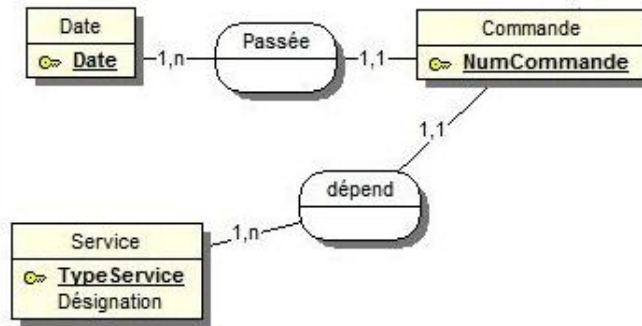
Nous pouvons considérer qu'un employé peut prendre zéro (le cuisinier par exemple) ou plusieurs commandes. Par contre, une commande est prise par un et un seul employé.



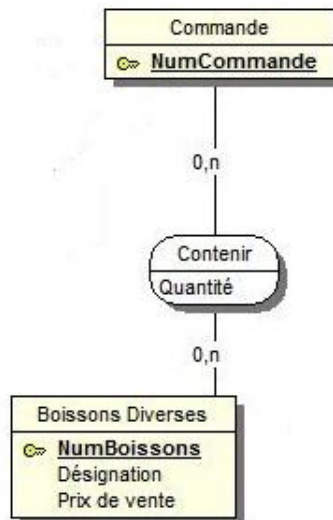
Une commande concerne une et une seule table et une table peut passer une ou plusieurs commandes.



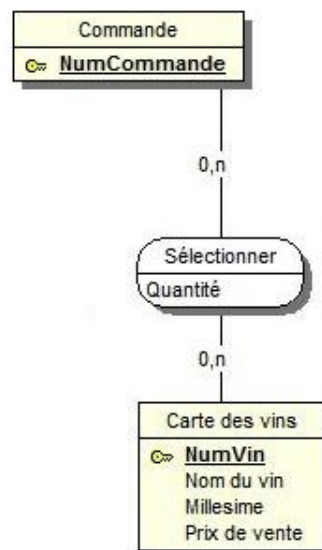
Une commande est passée à une et une seule date et à une date donnée il peut y avoir une ou plusieurs commandes passées.



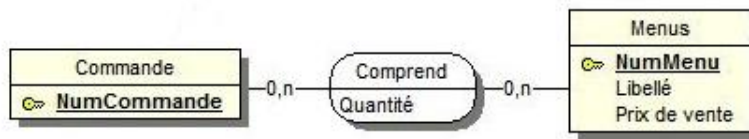
Une commande dépend d'un seul service (midi ou soir) et durant un service une ou plusieurs commandes peuvent être passées.



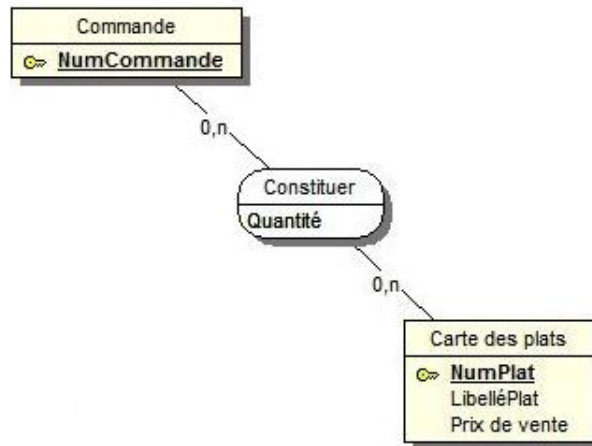
Une commande peut contenir zéro ou plusieurs boissons dans des quantités différentes et une boisson peut faire partie de zéro ou de plusieurs commandes dans des quantités différentes.



Une commande peut contenir zéro ou plusieurs vins dans des quantités différentes et un vin peut faire partie de zéro ou de plusieurs commandes dans des quantités différentes.

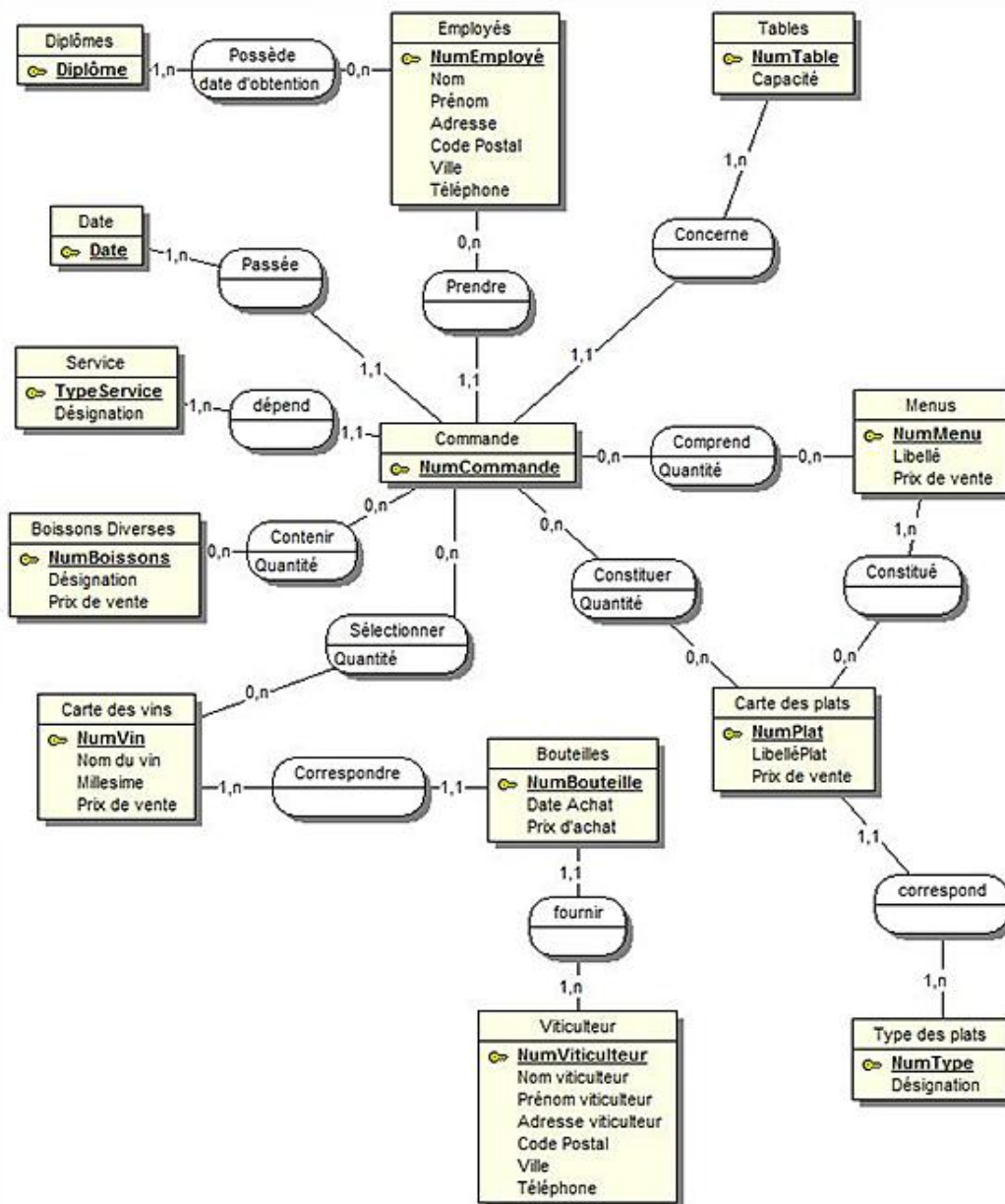


Une commande peut contenir zéro ou plusieurs menus dans des quantités différentes et un menu peut faire partie de zéro ou de plusieurs commandes dans des quantités différentes.



Une commande peut contenir zéro ou plusieurs plats dans des quantités différentes et un plat peut faire partie de zéro ou de plusieurs commandes dans des quantités différentes.





Voici la représentation globale du modèle conceptuel. Nous allons voir si nous pouvons améliorer certains points.

#### 4. Recherche d'identifiants relatifs

Si un viticulteur disparaît, les bouteilles qu'il vend vont disparaître aussi.

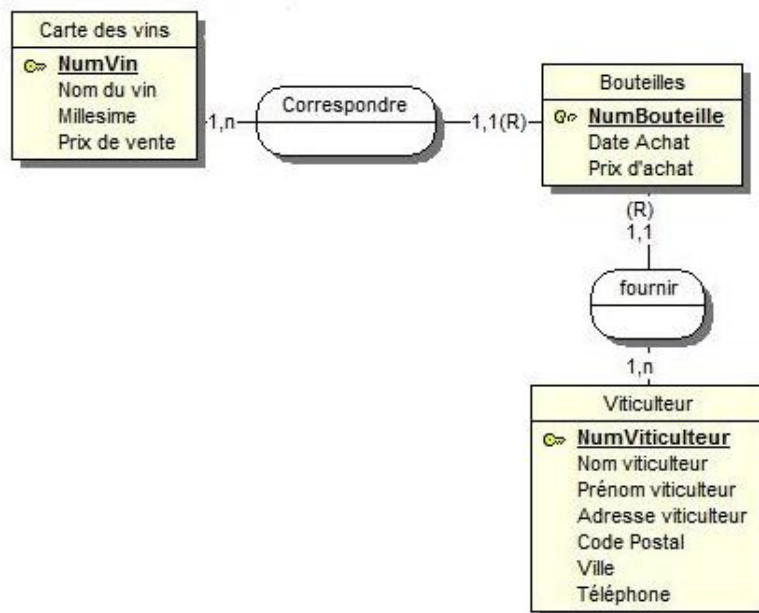
Si on supprime un vin de la carte des vins, les bouteilles afférentes à ce vin disparaissent aussi.

Si nous supprimons un type de plat, est-ce que la carte des plats disparaît ?

Ce n'est pas vraiment pertinent.

Nous considérerons qu'il n'y a pas d'identifiant relatif entre carte des plats et type des plats.

Modélisons ces modifications :



## Conclusion

Cette phase est l'une des plus importantes, il est évident qu'il faut y passer du temps. Il faut savoir aussi que plus vous pratiquerez les modèles conceptuels plus vous serez à l'aise.

N'hésitez pas à faire et refaire les exercices situés à la fin de ce livre pour vous perfectionner.

# Introduction au Modèle Logique des Données

Le **Modèle Logique des Données (MLD)** est la suite normale du processus Merise. Son but est de nous rapprocher au plus près du modèle physique. Pour cela, nous partons du **Modèle Conceptuel des Données** et nous lui enlevons les relations, mais pas n'importe comment, il faut en effet respecter certaines règles. Voici la procédure à suivre.

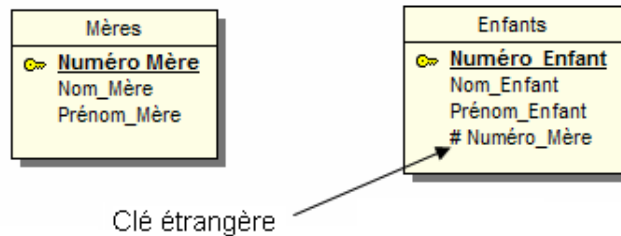
## 1. Cas (0, n), (1,1) ou (1,n), (0,1)

Voici un modèle conceptuel de départ :



Nous devons supprimer la relation Elever, cela se réalise de façon tout à fait mécanique. L'entité ayant la cardinalité de type 1,1 ou 0,1 absorbe l'identifiant de l'entité la plus forte (0, n ou 1, n). Cet identifiant est alors appelé **la clé étrangère**.

Voici le **Modèle Logique des Données** découlant du **Modèle conceptuel** précédent :



Nous pouvons l'illustrer par un cas concret.

Béatrice BAPTISTE a trois enfants : Amandine, Cédric, Sylvain.

Patricia AUGUY a deux enfants : Mathilde et Lucie.

Sandrine BAPTISTE a quatre enfants : Ophélie, Olivia, Lucie et Tom.

Voici le contenu du fichier de données Mères.

Mères	Numéro_Mère	Nom_Mère	Prénom_Mère
	1	BAPTISTE	Béatrice
	2	AUGUY	Patricia
	3	BAPTISTE	Sandrine
	...	...	...

Maintenant, illustrons le fichier de données Enfants.

Enfants	Numéro_Enfant	Nom_Enfant	Prénom_Enfant	#Numéro_Mère
	1	BAPTISTE	Amandine	1
	2	BAPTISTE	Cédric	1
	3	BAPTISTE	Sylvain	1

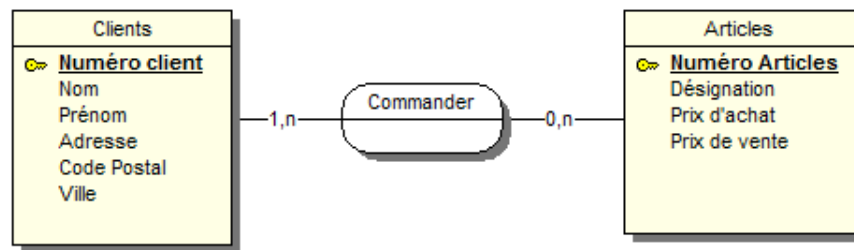
	4	AUGUY	Mathilde	2
	5	AUGUY	Lucie	2
	6	BAPTISTE	Ophélie	3
	7	BAPTISTE	Olivia	3
	8	BAPTISTE	Lucie	3
	9	BAPTISTE	Tom	3

Si nous désirons connaître les enfants de Patricia AUGUY, il nous faut concevoir une procédure qui réalise une recherche dans le fichier Mères et qui récupère l'identifiant de la mère recherchée. Ensuite, la procédure parcourt le fichier Enfants et chaque fois qu'elle rencontre l'identifiant de la mère dans la rubrique clé étrangère #Numéro\_Mère elle affiche la ligne correspondante.

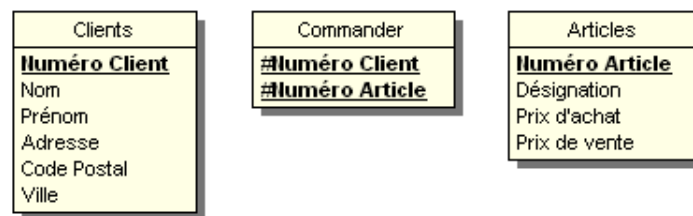
Comme nous le voyons, le principe général est très simple à comprendre.

## 2. Cas (0,n), (0,n) ou (1,n), (1,n)

Illustrons ce cas sur le Modèle Conceptuel des Données suivant :



Dans le cas où la cardinalité maximale est **n** de chaque côté de la relation, celle-ci se transforme en entité et absorbe les identifiants de chaque entité reliée. Les identifiants ainsi absorbés forment la nouvelle clé de l'entité. Cette nouvelle clé est donc formée par la concaténation des clés étrangères des entités reliées.



Voici la représentation virtuelle des fichiers de données :

Clients	Numéro Client	Nom	Prénom	....
	1	Cuvellier	Pierre	....
	2	Danjou	Isabelle	....
	3	Ardourel	Luc	....
	4	Perez	Jean-Christian	....
	....	....	....	....

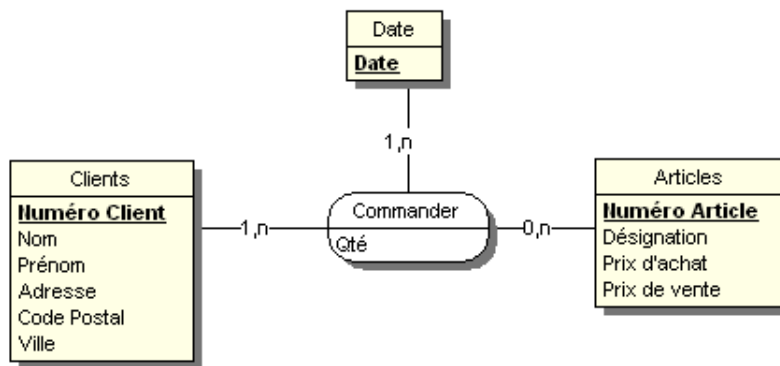
Articles	Numéro Article	Désignation	Prix d'achat	Prix de vente
----------	----------------	-------------	--------------	---------------

	1	Roquefort (300Gr)	8	12
	2	Livarot (500Gr)	7,8	11,90
	3	Perail de Brebis	5	9,4
	....	....	....	....

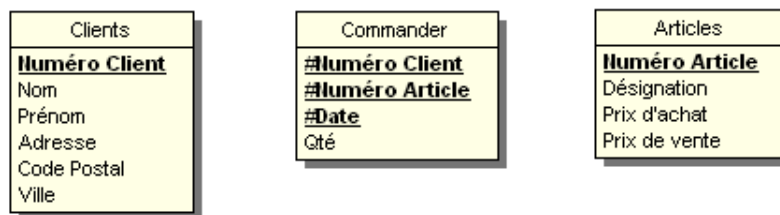
Commander	#Numéro Client	#Numéro Articles
	1	1
	2	2
	2	3
	....	....

➤ La concaténation des deux clés étrangères doit être unique. Les couples de clés sont donc (1,1), (2,2), (2,3) ... Nous arrivons donc à la conclusion suivante : le client 1 qui adore le Roquefort ne pourra pas en acheter plusieurs fois. Cette situation est anormale, je vous rappelle que « qui peut le plus peut le moins ».

Continuons à modifier le Modèle Conceptuel des Données.



Le Modèle Logique des Données en découlant sera :



La nouvelle représentation du fichier commande est maintenant ainsi :

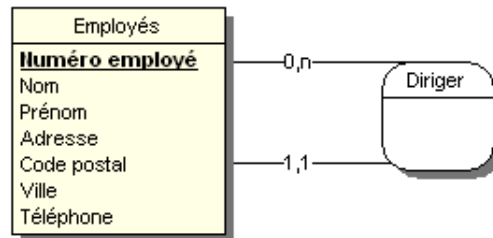
Commander	#Numéro Client	#Numéro Articles	Date	Qté
	1	1	12/08/2006	1
	1	2	12/08/2006	5
	1	1	13/08/2006	1

	2	2	12/08/2006	3
	2	2	13/08/2006	1
	....	....	....	....

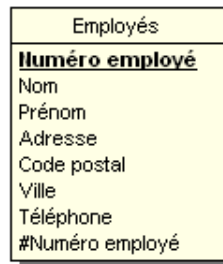
Nous avons apporté une solution à notre problème initial. Cependant, le même client ne pourra pas acheter deux fois le même fromage le même jour. Une solution élégante serait de concaténer la date et l'heure d'achat.

### 3. Modèle Logique des Données sur une relation réflexive

Reprenons ce Modèle Conceptuel des Données :



Les règles de passage du MCD au MLD s'appliquent toujours aussi mécaniquement. L'entité ayant la cardinalité la plus faible absorbe l'identifiant de l'entité reliée. Ici, nous n'avons qu'une seule entité, mais le principe est le même nous devons donc dupliquer l'identifiant Numéro employé.



Regardons la représentation du fichier de données.

Employés	Numéro employé	Nom	Prénom	...	#Numéro employé
	1	Collard	Marie-Claire	...	2
	2	Garcia	Patrick	...	
	3	Martinez	Manuel	...	2
	4	Savary	Thierry	...	2
	....	....	....	...	....

Nous observons bien que les employés sont dirigés par l'employé numéro 2.

### 4. Règles simples de passage du MCD au MLD

- L'entité qui possède la cardinalité maximale égale à 1, recevra l'identifiant ou les identifiants des entités ayant les cardinalités maximales les plus fortes.
- Les relations ayant toutes leurs entités reliées avec des cardinalités maximales supérieures à 1, se

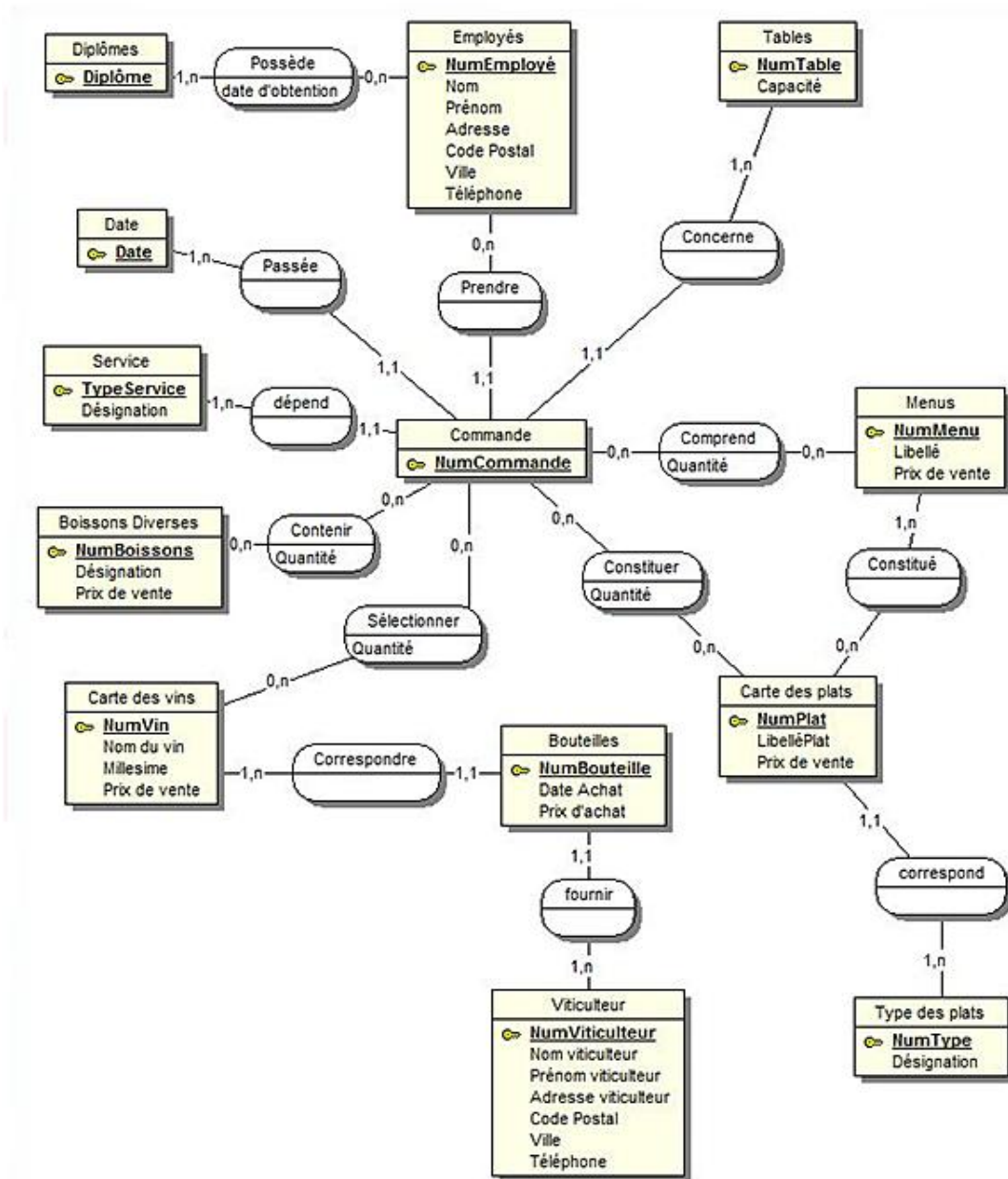
transformeront en entité en absorbant les identifiants des entités jointes.

- Toute relation porteuse de propriétés se transformera en entité et absorbera comme clé étrangère les identifiants des entités qui lui sont liées.

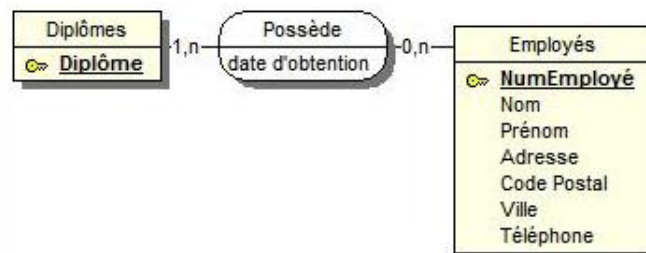


# Conception d'un Modèle Logique des Données pas à pas

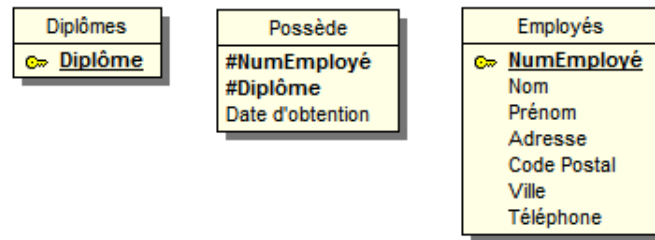
Appliquons ces règles au modèle conceptuel du chapitre précédent :



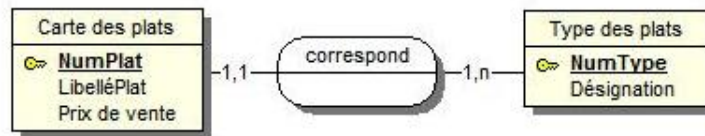
Reprenons au cas par cas et commençons par cet extrait du Modèle Conceptuel des Données :



Voici la traduction en Modèle Logique des Données :



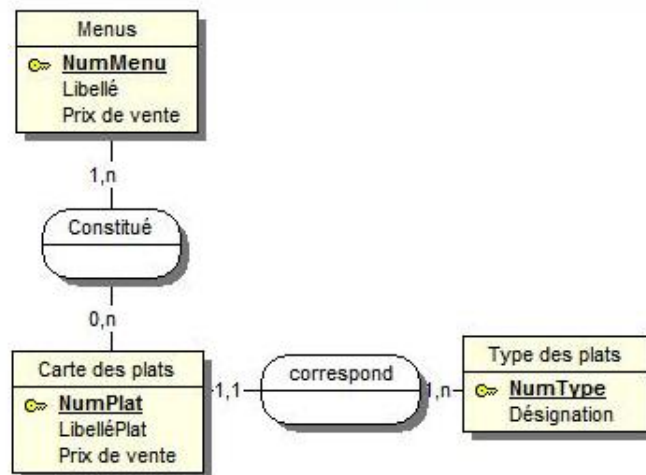
Comme nous pouvons le constater, une nouvelle entité est apparue (Possède). Cette entité contient trois propriétés, dont deux clés étrangères. Le nouvel identifiant de cette entité sera la concaténation des deux clés étrangères.



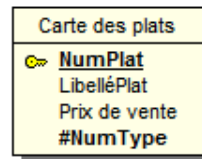
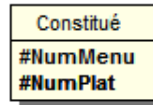
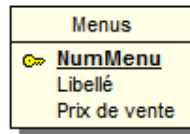
Ici, nous pouvons voir que la cardinalité (1,1) va nous indiquer l'entité qui va recevoir la clé étrangère.



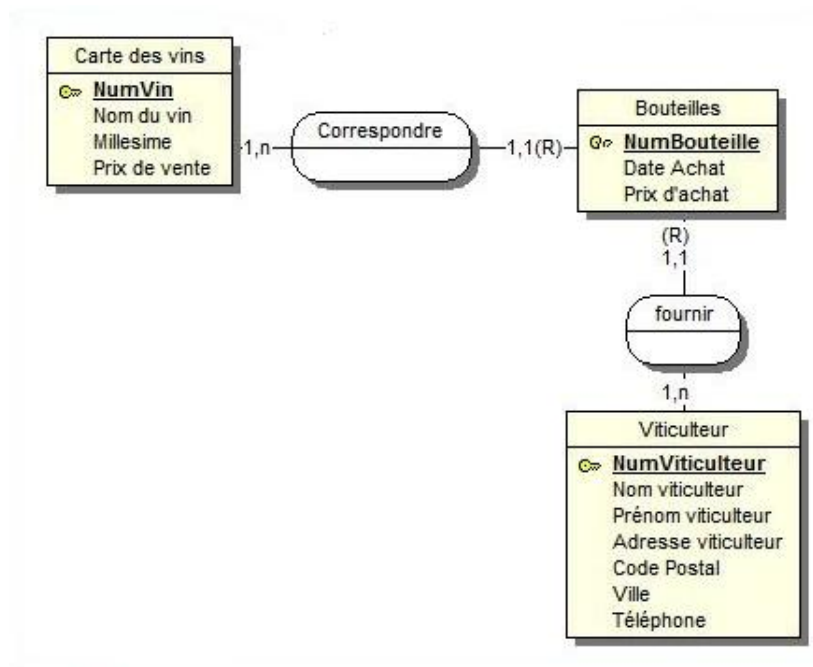
La propriété NumType va devenir clé étrangère dans l'entité Carte des Plats.



Cette partie de MCD n'est pas complexe à transposer en MLD :



Continuons le processus :



Ici, nous traitons le cas des identifiants relatifs :

Carte des vins
<b>NumVin</b> Nom du vin Millesime Prix de vente

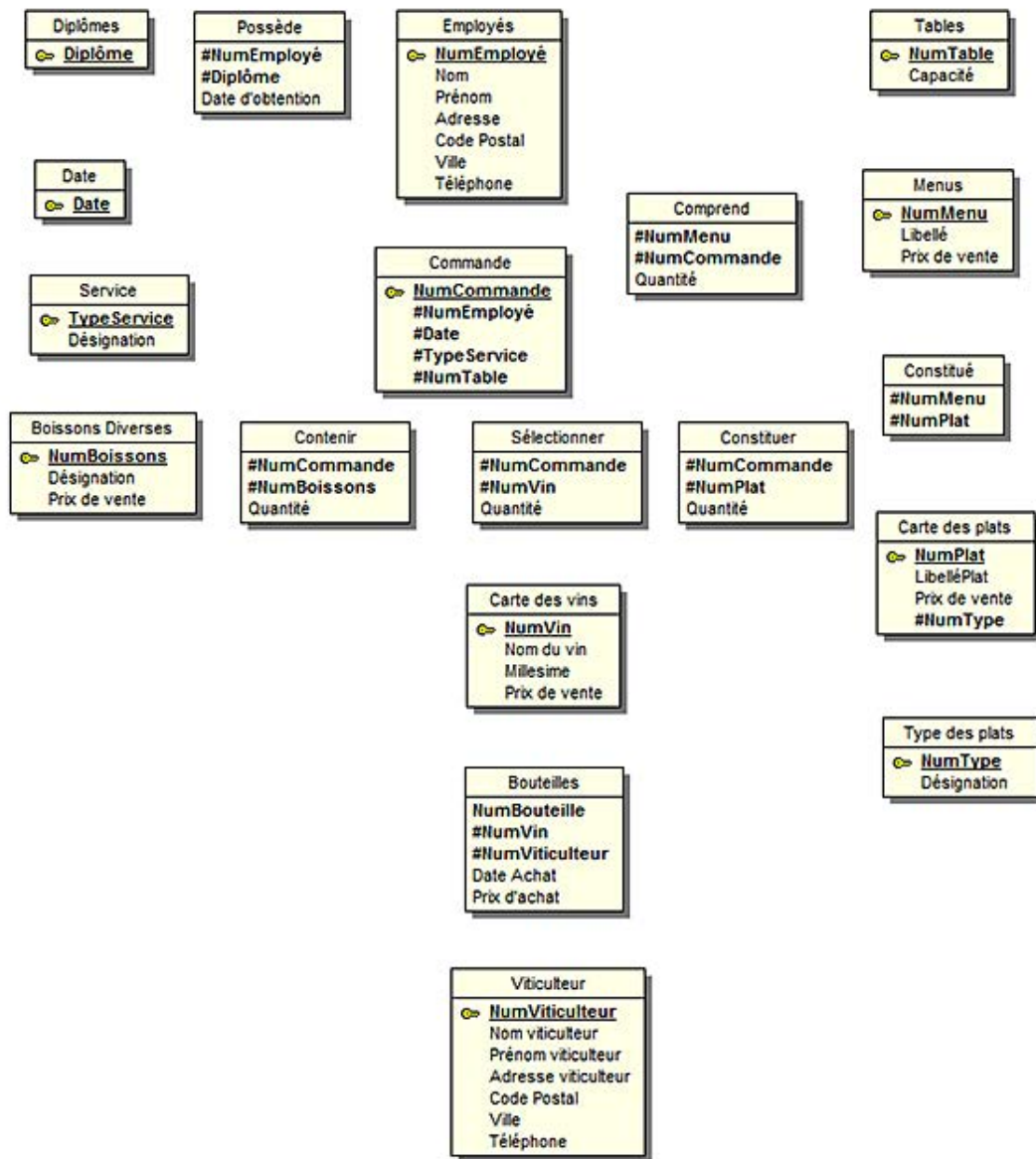
Bouteilles
<b>NumBouteille</b> <b>#NumVin</b> <b>#NumViticulteur</b> Date Achat Prix d'achat

Viticulteur
<b>NumViticulteur</b> Nom viticulteur Prénom viticulteur Adresse viticulteur Code Postal Ville Téléphone

La nouvelle clé identifiante de l'entité **Bouteilles** sera la concaténation des trois clés :

- NumBouteille
- #NumVin
- #NumViticulteur

Voici le Modèle logique finalisé :



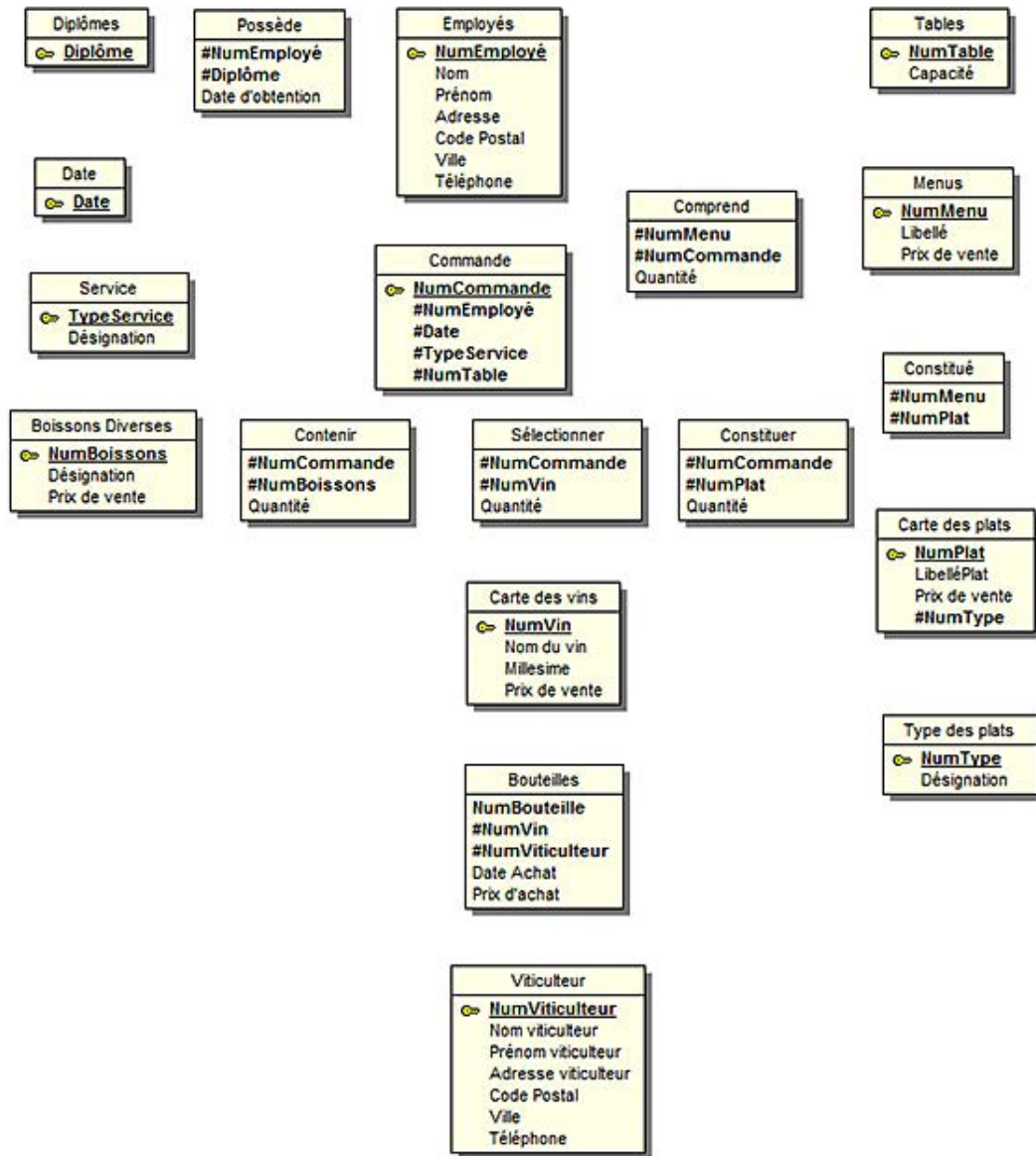
## Conclusion

Comme vous l'avez ressenti, le passage du modèle conceptuel au Modèle Logique des Données est purement mécanique, il suffit de respecter les quelques règles énoncées plus haut. Il n'y a plus de travail de conceptualisation ou de réflexion proprement dit. Lorsque nous réalisons un Modèle Logique des Données nous ne faisons que « *détruire* » un Modèle Conceptuel des Données pour recréer un autre modèle.

# Introduction au Modèle Physique des Données

Construire le Modèle Physique des Données consiste à transformer le Modèle Logique des Données en une suite de relations. Cette étape finalise le processus de traitement des données. L'implémentation des bases de données peut être réalisée de façon optimale.

Reprenons le modèle conceptuel précédent :



Voici les relations (ou schéma relationnel) du modèle physique qui en découlent :

Diplômes (**Diplomes**)

Possède (**#NumEmployé**, **#Diplôme**, Date d'obtention)

Employés (**NumEmployé**, Nom, Prénom, Adresse, Code Postal, Ville, Téléphone)

Tables (**NumTable**, Capacité)

Date (**Date**)

Service (**TypeService**, Désignation)

Boissons Diverses (**NumBoissons**, Désignation, Prix de vente)

Contenir (**#NumCommande**, **#NumBoissons**, Quantité)

Commande (**NumCommande**, **#Numemployé**, **#Date**, **#TypeService**, **#NumTable**)

Comprend (**#NumMenu**, **#NumCommande**, Quantité)

Menus (**NumMenu**, Libellé, Prix de vente)

Constitué (**#NumMenu**, **#NumPlat**)

Constituer (**#NumCommande**, **#NumPlat**, Quantité)

Sélectionner (**#NumCommande**, **#NumVin**, Quantité)

Carte des vins (**NumVin**, Nom du vin, Millesime, Prix de vente)

Carte des plats (**NumPlat**, LibelléPlat, Prix de vente, **#NumType**)

Type des plats (**NumType**, Désignation)

Bouteilles (**NumBouteille**, Date Achat, Prix d'achat, **# NumVin**, **#NumViticulteur**)

Viticulteur (**NumViticulteur**, Nom viticulteur, Prénom viticulteur, Adresse viticulteur, Code postal, Ville, Téléphone)

Comme vous le voyez, passer du modèle logique de données au modèle physique des données ne présente aucune difficulté. On abandonne juste la représentation graphique pour une représentation plus linéaire.



## Transcription SQL du modèle physique

Par exemple si nous devons porter notre modèle physique sur un **S**ystème de **G**estion de **B**ase de **D**onnées (**SGBD**) il suffirait d'écrire les requêtes SQL de création de tables correspondantes. En voici un exemple sur trois tables :

```
CREATE TABLE CARTE_DES_VINS
(
  NUMVIN INTEGER(2) NOT NULL ,
  NOM_DU_VIN CHAR(40) ,
  MILLESIME INTEGER(2) ,
  PRIX_DE_VENTE REAL(5,2)
,
  PRIMARY KEY (NUMVIN) CONSTRAINT PK_CARTE_DES_VINS
);

CREATE TABLE BOUTEILLES
(
  NUMVITICULTEUR INTEGER(2) NOT NULL ,
  NUMVIN INTEGER(2) NOT NULL ,
  NUMBOUTEILLE INTEGER(2) NOT NULL ,
  DATE_ACHAT DATE(8) ,
  PRIX_D_ACHAT REAL(5,2)
,
  PRIMARY KEY (NUMVITICULTEUR, NUMVIN, NUMBOUTEILLE) CONSTRAINT
PK_BOUTEILLES
);

CREATE TABLE VITICULTEUR
(
  NUMVITICULTEUR INTEGER(2) NOT NULL ,
  NOM_VITICULTEUR CHAR(20) ,
  PRÉNOM_VITICULTEUR CHAR(20) ,
  ADRESSE_VITICULTEUR CHAR(40) ,
  CODE_POSTAL CHAR(5) ,
  VILLE CHAR(40) ,
  TÉLÉPHONE CHAR(15)
,
  PRIMARY KEY (NUMVITICULTEUR) CONSTRAINT PK_VITICULTEUR
);
```

Nous pouvons remarquer l'expression de la clé primaire dans la table vin qui est la concaténation des trois identifiants : NUMVITICULTEUR, NUMVIN, NUMBOUTEILLE.

## Conclusion

Le Modèle Physique des Données est l'étape ultime dans le processus de gestion des données de la méthode Merise. Toute l'analyse ayant été réalisée en amont, l'essentiel du travail de réflexion ayant été encadré par le modèle conceptuel, le passage au modèle physique n'est qu'une simple formalité. Il peut être donné à un développeur pour qu'il puisse créer la base de données correspondante sur un serveur de base de données quelconque.

# Introduction aux formes normales

Pour être parfaites, les relations doivent respecter certaines règles. Cet ensemble de règles se nomme : **les formes normales**.

Cette théorie a été élaborée par E.F. Codd en 1970. Son objectif est d'éviter les anomalies dans les bases de données relationnelles :

- Problèmes de mise à jour.
- Suppression des redondances d'informations.
- Simplification de certaines contraintes d'intégrité.

Pour parfaire une base de données relationnelle, il est nécessaire de connaître les trois premières formes normales et la forme normale dite Boyce-Codd ; les suivantes ne sont que des extensions peu usitées.

## 1. 1FN - Première forme normale

### Définition

Une relation est en première forme normale si :

- Tous les attributs ne contiennent qu'une seule valeur atomique (non divisible).
- Les attributs ne contiennent pas de valeurs répétitives.

### Exemple :

Clients (NumCli, Nom, Prénom, Adresse, Téléphone)

Cette relation n'est pas en première forme normale, car **Adresse n'est pas atomique**. En effet voici une représentation d'un fichier ainsi décrit :

NumCli	Nom	Prénom	Adresse	Téléphone
1	Baptiste	Jean-Luc	25, rue de la forêt 12000 Rodez	0565420000
2	Auguy	Joel	Impasse des lys 15000 Aurillac	0471670000
3	Rascalou	André	2, rue droite 12000 Rodez	0565450000

Cette représentation si elle était mise en pratique générerait un accès aux données plus lent. Le simple fait de vouloir extraire les habitants d'une ville précise devra mettre en œuvre des procédures d'extraction de sous-chaînes sans fournir de garantie quant au résultat retourné.

Voici une représentation 1FN correcte :

Clients (NumCli, Nom, Prénom, Adresse, CodeP, Ville, Téléphone)

NumCli	Nom	Prénom	Adresse	Code Postal	Ville	Téléphone
1	Baptiste	Jean-Luc	25, rue de la forêt	12000	Rodez	0565420000
2	Auguy	Joel	Impasse des lys	15000	Aurillac	0471670000
3	Rascalou	André	2, rue droite	12000	Rodez	0565450000

Maintenant, récupérer les habitants d'une ville précise ne pose plus aucun problème, une simple requête SQL y parviendra de façon rapide et fiable.

## 2. 2FN - Deuxième forme normale

### Définition

Une relation est en deuxième forme normale si :

- Elle est en première forme normale.
- Si tous les attributs non-clés ne dépendent pas d'une partie de la clé primaire.

Autrement dit, toute propriété de la relation doit dépendre intégralement de toute la clé.

Par exemple :

*Commande (Numcli, CodeArticle, Date, Qté commandée, Désignation)*

Cette relation est-elle en première forme normale ? Oui.

Est-elle en deuxième forme normale ? Non, car la propriété Désignation ne dépend pas intégralement de la clé (Numcli, CodeArticle, Date).

Commandes :

NumCli	CodeArticle	Date	Qté commandée	Désignation
1	Art1	28/02/2009	5	Bocal d'un kilo de Tripoux
3	Art2	28/02/2009	9	Aligot congelé
5	Art3	28/02/2009	10	Cèpes séchés
6	Art41	28/02/2009	15	Bouteille de Marcillac rouge

Connaissant {1,Art1,28/02/2009} pouvons-nous connaître de façon sûre et unique « **Bocal d'un kilo de Tripoux** » ? La réponse est évidemment non ! « **Bocal d'un kilo de Tripoux** » ne dépend pas intégralement de la clé {1,Art1,28/02/2009}.

Voici comment corriger :

Commandes(Numcli, CodeArticle, date, Qté commandée) Articles(CodeArticle, Désignation)

NumCli	CodeArticle	Date	Qté commandée
1	Art1	28/02/2009	5
3	Art2	28/02/2009	9
5	Art3	28/02/2009	10
6	Art41	28/02/2009	15

Articles :

CodeArticle	Désignation
Art1	Bocal d'un kilo de Tripoux
Art2	Aligot congelé

Art3	Cèpes séchés
Art41	Bouteille de Marcillac rouge

### 3. 3FN - Troisième forme normale

#### Définition

Une relation est en troisième forme normale si :

- Elle est en deuxième forme normale.
- Si toutes les dépendances fonctionnelles par rapport à la clé sont directes (s'il n'y a pas de DF transitives entre les attributs non clé).

Autrement dit, tous les attributs n'appartenant pas à la clé ne dépendent pas d'un attribut non-clé.

#### Exemple :

La relation *Commande*(NuméroCommande, #CodeClient, Nom client, #RefArticle) est-elle en troisième forme normale ?

Est-elle en première forme normale ? Oui

Est-elle en deuxième forme normale ? Oui

Est-elle en troisième forme normale ? Non !

En effet Nom client dépend d'une propriété non clé : CodeClient

#### Commandes :

NuméroCommande	CodeClient	Nom client	RefArticle
1	C1	Baptiste	Art25
3	C5	Savary	Art20
5	C2	Martinez	Art10
6	C1	Baptiste	Art15

Voici comment corriger :

*Commande*(NuméroCommande, #CodeClient, #RefArticle) *Clients*(CodeClient, Nom client)

#### Commandes :

NuméroCommande	CodeClient	RefArticle
1	C1	Art25
3	C5	Art20
5	C2	Art10
6	C1	Art15

#### Clients :

CodeClient	Nom client
C1	Baptiste
C2	Martinez
C5	Savary

## 4. BCNF - Forme normale de Boyce - Codd

### Définition

Une relation est en forme normale de BOYCE-CODD (BCNF) si et seulement si :

- Elle est en troisième forme normale.
- Les seules dépendances fonctionnelles élémentaires qu'elle comporte sont celles dans lesquelles une clé détermine un attribut.

Considérons la relation Vaches(Race, Pays, Région) avec les dépendances fonctionnelles supposées :

Région → Pays

(Race, Pays) → Région

Race	Pays	Région
Aubrac	France	Auvergne
Salers	France	Auvergne
Limousine	France	Limousin
highland	Royaume-Uni	highlands

Cette relation est bien en troisième forme normale car aucun attribut non clé ne dépend d'une partie de la clé ou d'un attribut non clé. Cependant, on y trouve de nombreuses redondances, par exemple les deux premières lignes possèdent des pays et des régions identiques.

Afin d'éliminer ces redondances, Boyce et Codd ont introduit une forme normale qui porte leur nom (*Boyce Codd Normal Form/BCNF*) :

Une relation est en BCNF si et seulement si les seules dépendances fonctionnelles élémentaires sont celles dans lesquelles une clé détermine un attribut.

La relation Vaches pourra être décomposée en deux relations :

- Races (Race, Région)
- Régions (Région, Pays)

La dépendance fonctionnelle (Race, Pays) → Région est perdue, mais elle peut être recomposée par jointure.

Cette normalisation est très importante dans la pratique si l'on veut éviter de stocker des informations redondantes. On considère, en général, que la troisième forme normale est suffisante dans les cas courants.

## 5. 4FN - Quatrième forme normale

### Définition :

Une relation est en quatrième forme normale si et seulement si :

- Elle est en BCNF.
- Lorsqu'il existe une dépendance multivaluée élémentaire, celle-ci est unique.

Une relation en BCNF peut encore comporter des redondances.

Par exemple :

*L'étudiant (NumEtu) pratique une ou plusieurs langues et suit un ou plusieurs cours.*

Imaginons cette relation :

Etudiants (NumEtu, Langue, Cours)

Et les postulats suivants :

Il n'y a pas de dépendances fonctionnelles.

La clé est : (NumEtu, Langue, Cours).

La relation Etudiants est en troisième forme normale et en troisième forme normale Boyce-Codd.

Cependant Etudiants contient encore des redondances :

Etudiants	NumEtu	Langue	Cours
	1	Anglais	Mathématiques
	1	Anglais	Histoire
	2	Anglais	Economie
	2	Espagnol	Economie
	3	Espagnol	Economie
	3	Espagnol	Droit

L'étudiant numéro 1 ne connaît que l'anglais, mais suit 2 cours (les mathématiques et l'histoire). Il y a donc une redondance sur la langue.

L'étudiant numéro 2 connaît deux langues (l'anglais et l'espagnol) mais ne suit qu'un cours. Il y a redondance sur le cours.

Les dépendances fonctionnelles ne suffisent pas à définir toutes les dépendances entre les données.

### **Dépendances multivaluées**

Pour une valeur d'étudiant, on a toutes les valeurs possibles de langue et, pour chacune de ces valeurs, toutes les valeurs possibles de cours, mais langue et cours sont indépendantes entre elles : on dit que l'on a une dépendance multivaluée entre la colonne NumEtu et la colonne Langue et une dépendance multivaluée entre la colonne NumEtu et la colonne Cours.

On voit l'inconvénient de cette forme puisque, si l'on supprime une valeur possible de la colonne Cours, par exemple l'organisme de formation retire l'économie de son catalogue, il faut supprimer toutes les lignes où économie est inscrit.

Pour éviter ce genre de problèmes, il faut passer à la quatrième forme normale, qui se peut se définir ainsi :

Une relation est en quatrième forme normale si et seulement si les seules dépendances multivaluées élémentaires sont celles dans lesquelles une clé détermine la valeur d'une colonne.

Ici, les colonnes sur lesquelles portent des dépendances multivaluées font partie de la clé, donc la table n'est pas en 4FN et il faut la décomposer en deux tables :

Etudiants1(NumEtu, Langue)

Etudiant2(NumEtu, Cours)

## **6. 5FN - Cinquième forme normale**

Cette forme normale n'étant quasiment jamais utilisée, en voici juste la définition :

### **Définition**

Une association est en cinquième forme normale si et seulement si :

- Elle est en quatrième forme normale.
- Elle ne possède pas de dépendance de jointure.

La cinquième forme normale est une généralisation de la quatrième forme normale qui nécessite de prendre en compte les dépendances de jointure induites par la connaissance des clés d'une relation.



## Conclusion

L'étude des formes normales permet d'éviter certains pièges de conception risquant d'impacter la future base de données. Il est donc important que durant le processus de modélisation, un instant soit pris pour vérifier qu'il n'y a pas d'incohérences fonctionnelles.

# Introduction aux diagrammes des flux

Ce diagramme donne une vue d'ensemble (ou cartographie) de la circulation des informations (les flux) entre des acteurs internes ou externes qui participent à un domaine d'étude.

## 1. Définitions

### a. Domaine d'étude

Un domaine d'étude délimite le périmètre précis d'une ou de plusieurs activités au sein d'une organisation spécifique.

### b. L'acteur

L'acteur (interne ou externe au domaine d'étude) est un système actif intervenant dans le domaine d'étude au moyen des flux.

Un acteur peut représenter :

- Un intervenant extérieur à l'entreprise (fournisseurs, clients...).
- Un domaine de l'entreprise (le service du personnel, la comptabilité...).
- acheteurs, vendeurs, étudiants... variable selon les cas.

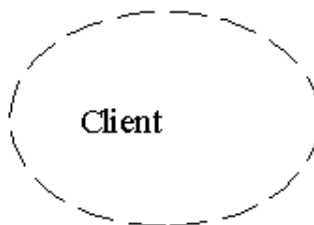
### c. Les flux

Les flux symbolisent un échange entre deux acteurs du système d'information étudié. Il est représenté par une flèche, porte un nom et peut, pour des soucis de lisibilité chronologique, être numéroté.

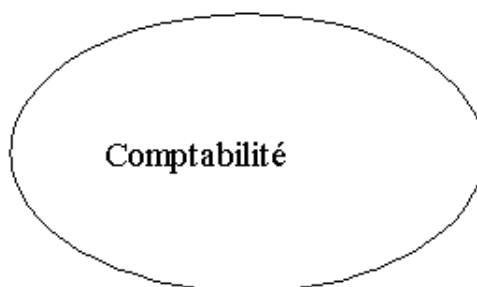
## 2. Représentation graphique des acteurs

Il n'existe pas à l'heure actuelle de normalisation des représentations des acteurs et des flux.

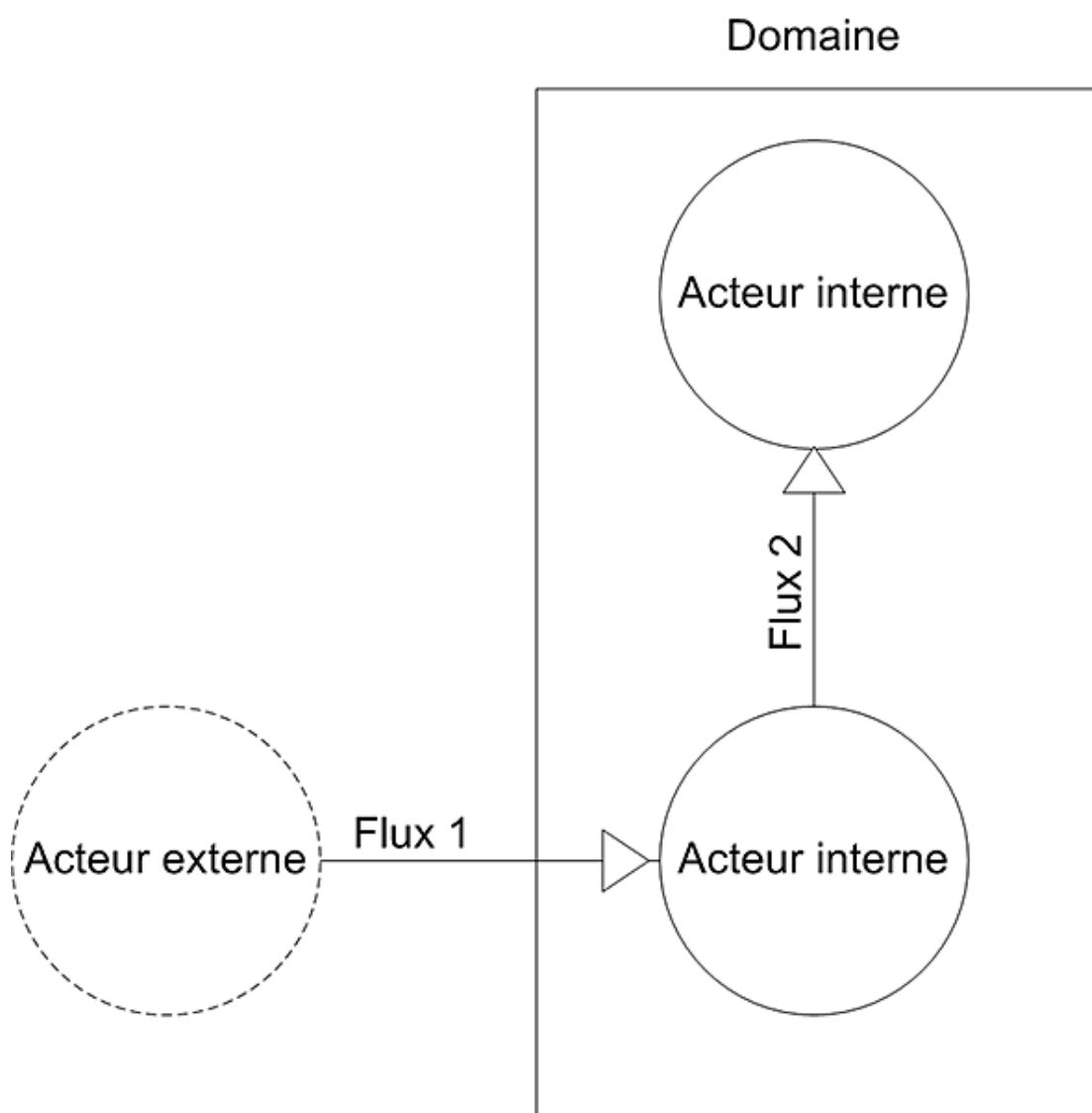
Dans certains cas les acteurs externes sont symbolisés de cette façon :



Les acteurs internes eux sont symbolisés de cette façon :



Voici une représentation plus complète d'un diagramme des flux :



# Conception d'un diagramme des flux pas à pas

Une agence de location de vélo veut informatiser la gestion des locations. Lorsqu'un client se présente à l'accueil, il précise le type de vélo désiré ainsi que la durée de location. L'accueil vérifie si, en fonction du stock disponible, la location est possible et donne la réponse au client. Si la location est possible, la facture est éditée et donnée au client. Celui-ci doit payer immédiatement. Le paiement et la facture sont ensuite transmis au service comptable. L'accueil transmet alors la demande au gestionnaire du parc. Ce dernier va préparer le vélo demandé et le mettre à disposition du client.

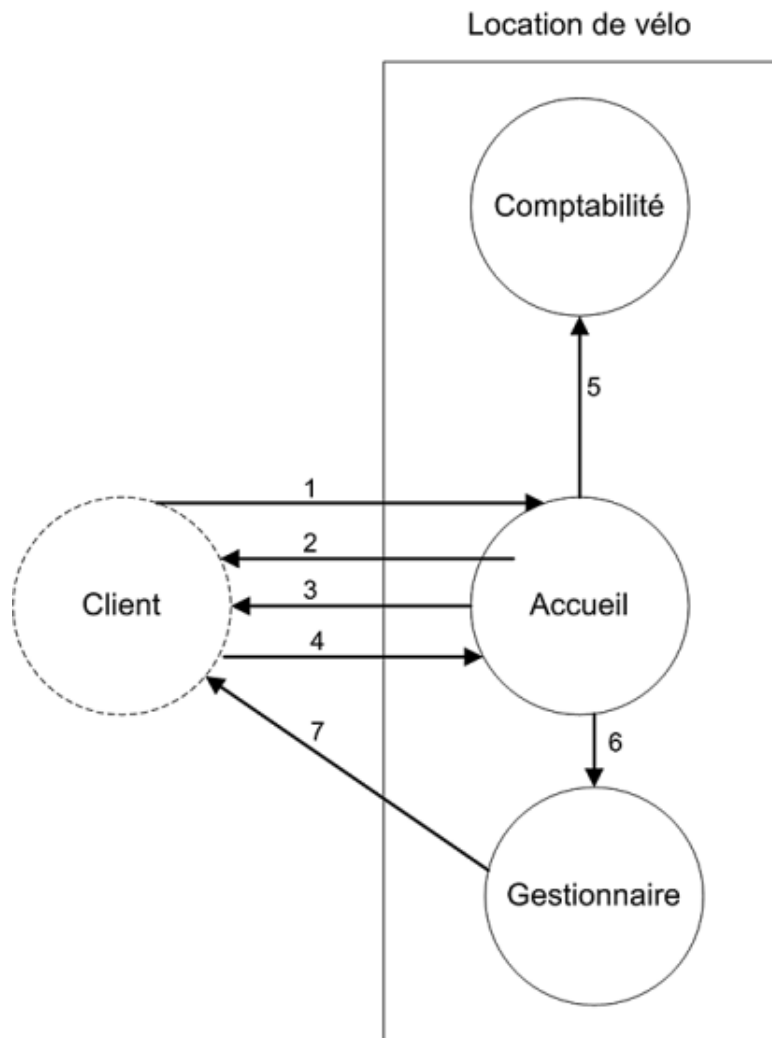
## 1. Identification des flux

- Le premier flux est la demande de location.
- Le deuxième flux est l'acceptation ou le refus de la location en fonction du stock disponible.

Dans le cas de l'acceptation :

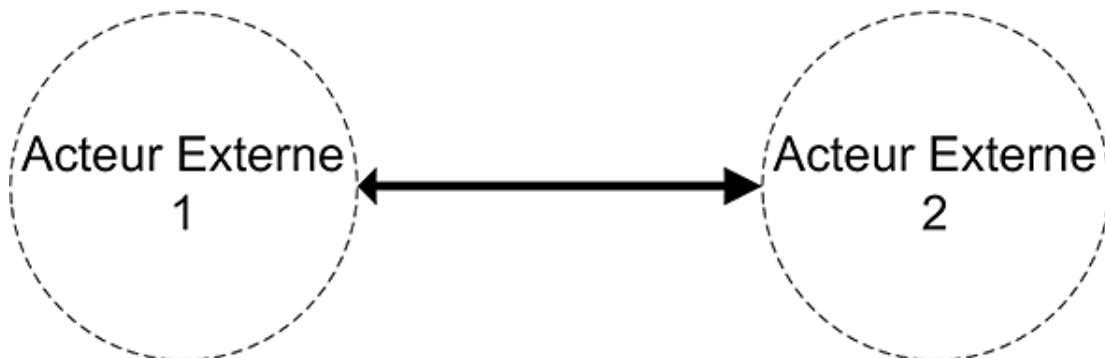
- Le troisième flux représentera l'édition de la facture.
- Le quatrième flux sera le paiement de la facture par le client.
- Le cinquième flux représente le passage de la facture et du paiement au service comptabilité.
- Le sixième flux est la transmission de la demande au gestionnaire du parc.
- Le septième et dernier flux est la remise du vélo au client par le gestionnaire.

## 2. Diagramme finalisé



### 3. Remarques et règles d'usages

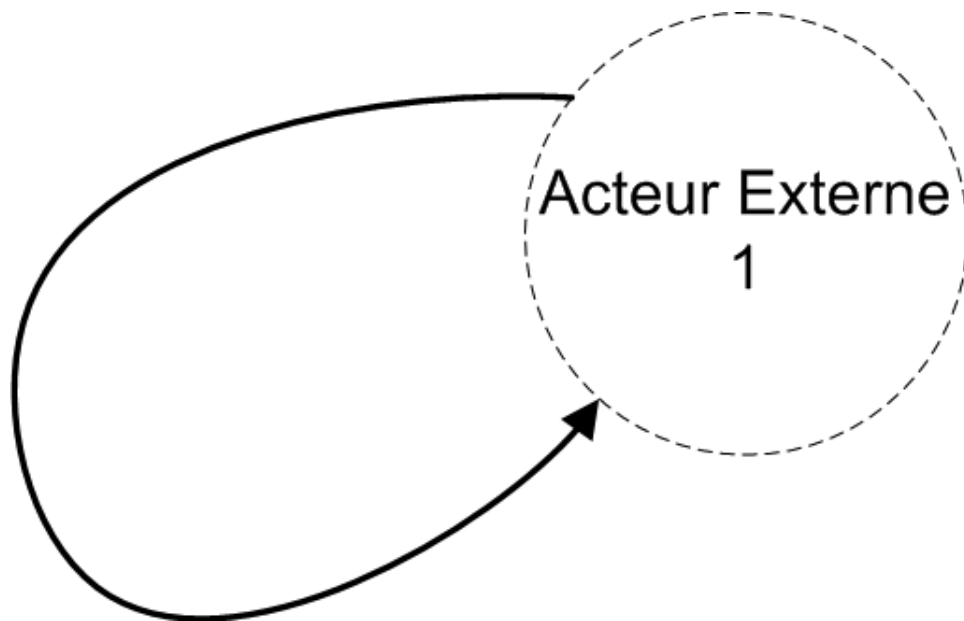
#### a. Un flux ne doit pas être bidirectionnel



Il ne doit pas exister entre deux acteurs (internes, externes ou identiques) de liens bidirectionnels. Il convient de noter deux flux distincts.

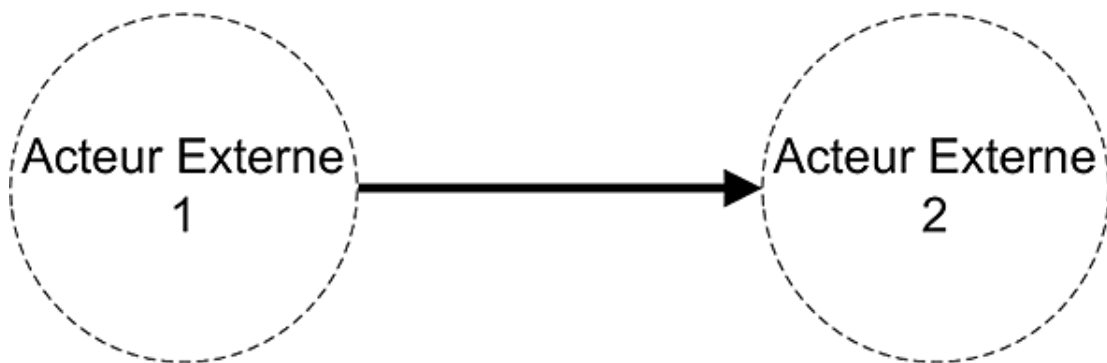
#### b. Le flux ne doit pas être réflexif

Un flux ne doit pas partir et revenir sur le même acteur (interne ou externe). Si cela est nécessaire, nous devons segmenter l'acteur.



**c. Pas de flux entre des acteurs externes**

Les flux entre les acteurs externes ne sont d'aucun intérêt dans l'étude du système.



## Le modèle de contexte

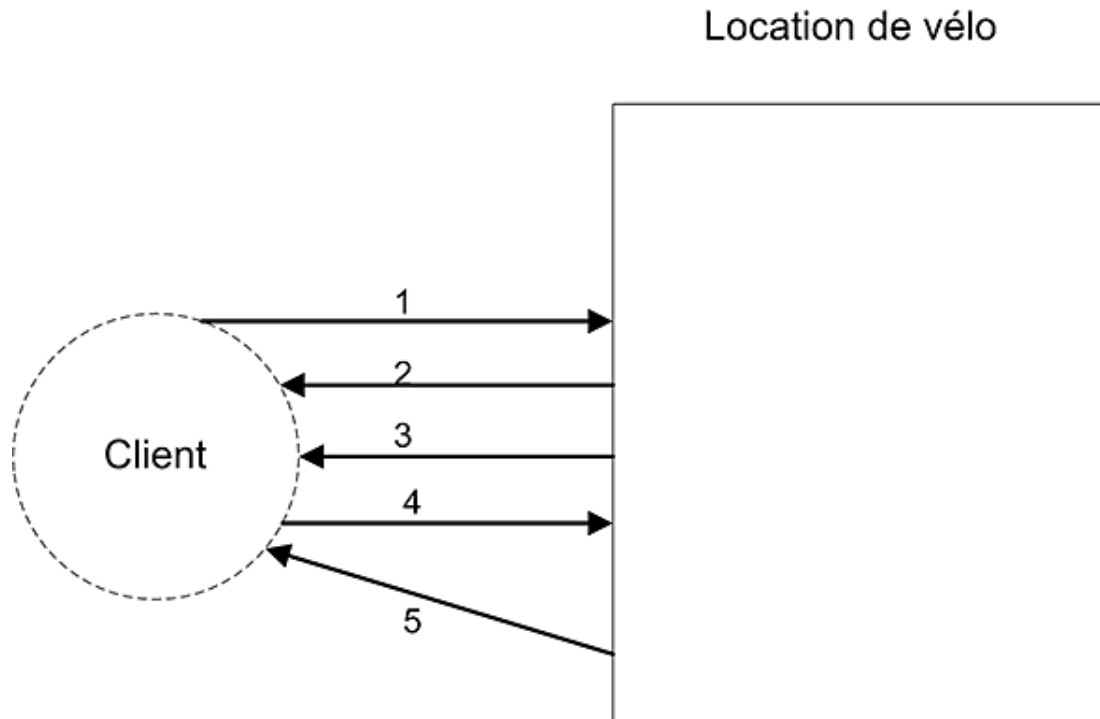
Le modèle est aussi appelé Modèle Conceptuel de Communication de niveau 0.



Il existe différentes dénominations des modèles de flux suivant le niveau d'observation.

Ce modèle ne gère pas les acteurs internes au domaine étudié, mais juste les échanges entre le domaine et les acteurs externes.

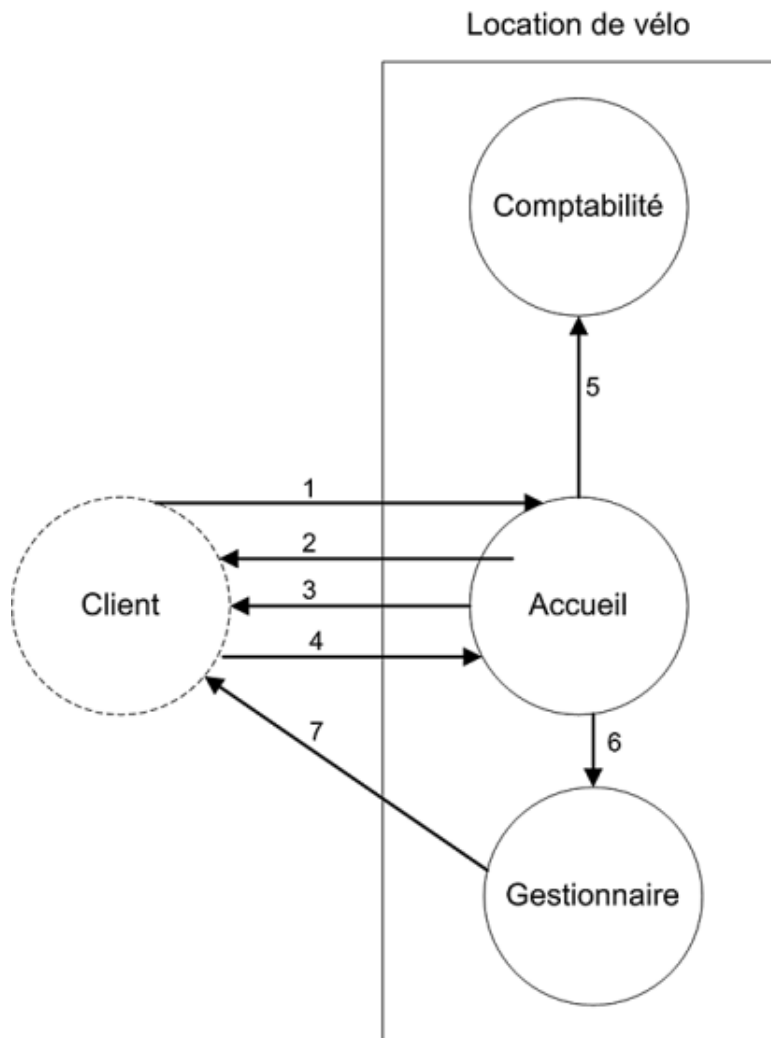
Par exemple :



### 1. Le Modèle de Flux Conceptuel (de niveau 1, de niveau N)

Les acteurs internes sont introduits dans le modèle, l'affinage successif des acteurs internes indique le niveau.

Par exemple voici un modèle de niveau 1 :

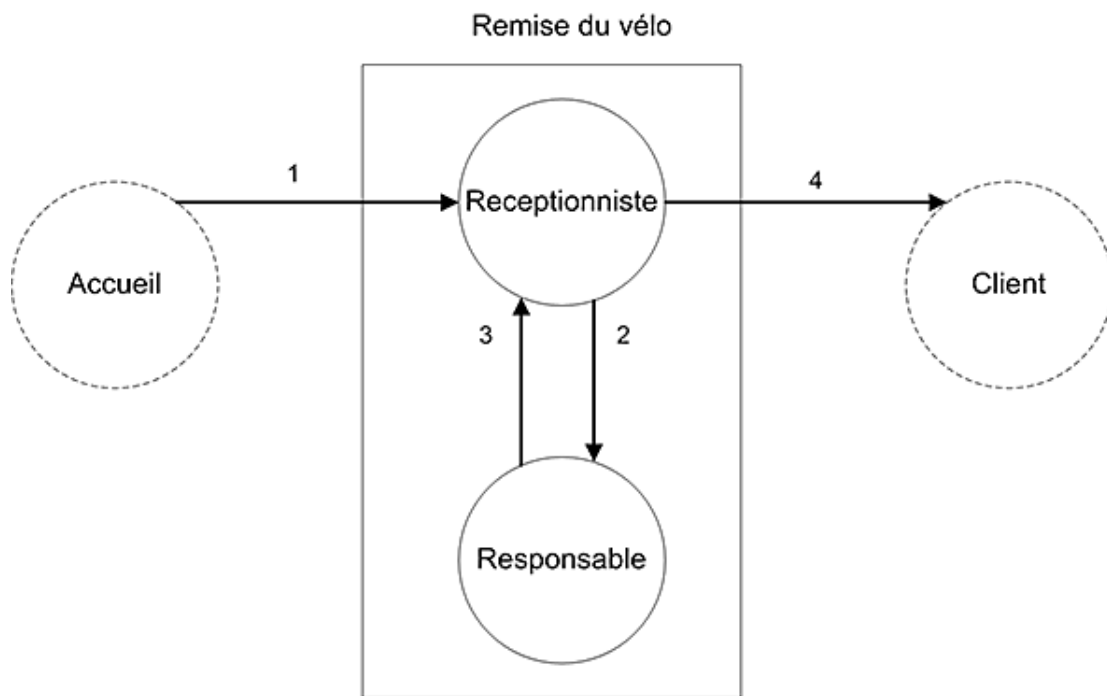


Voici un modèle de niveau 2 :

Imaginons que nous souhaitions « zoomer » sur le service gestionnaire en introduisant les nouvelles règles suivantes : le réceptionniste reçoit le bon de remise du vélo, il contrôle que le vélo est en état de fonctionnement. Si exceptionnellement ce n'est pas le cas, il demande à son supérieur l'autorisation de prêter un vélo de secours.

Voici le modèle de niveau 2 qui pourrait en découler :





## Conclusion

Les diagrammes des flux permettent d'avoir une vision claire du projet en amont. C'est souvent par un diagramme de contexte (modèle conceptuel de communication) qu'un projet Merise débute. Ce modèle permet une définition claire du système d'information et donne à l'intervenant externe une bonne idée de la structure de l'entreprise, de ses règles métiers et des différentes articulations interservices.

# Le Modèle Conceptuel des Traitements

## 1. Objectifs du Modèle Conceptuel des Traitements

Le Modèle Conceptuel des Traitements met en lumière les traitements effectués sur les données. Indépendamment de toute contrainte liée à l'organisation, le Modèle Conceptuel des Traitements répond à la question « Quoi ? ». Le Modèle Conceptuel des Traitements ne répond ni au comment, ni au quand, ni au qui, mais à Que souhaite-t-on obtenir ?

### a. Les événements

Le MCT est aussi appelé Modèle événement-résultat. L'arrivée d'un ou plusieurs événements va générer une opération qui va elle-même fournir un résultat. Selon leur origine on distingue les événements externes (exemple : la commande d'un client) et les événements internes générés par le système d'information (exemple : l'émission d'une facture).

Un événement est représenté de la façon suivante :

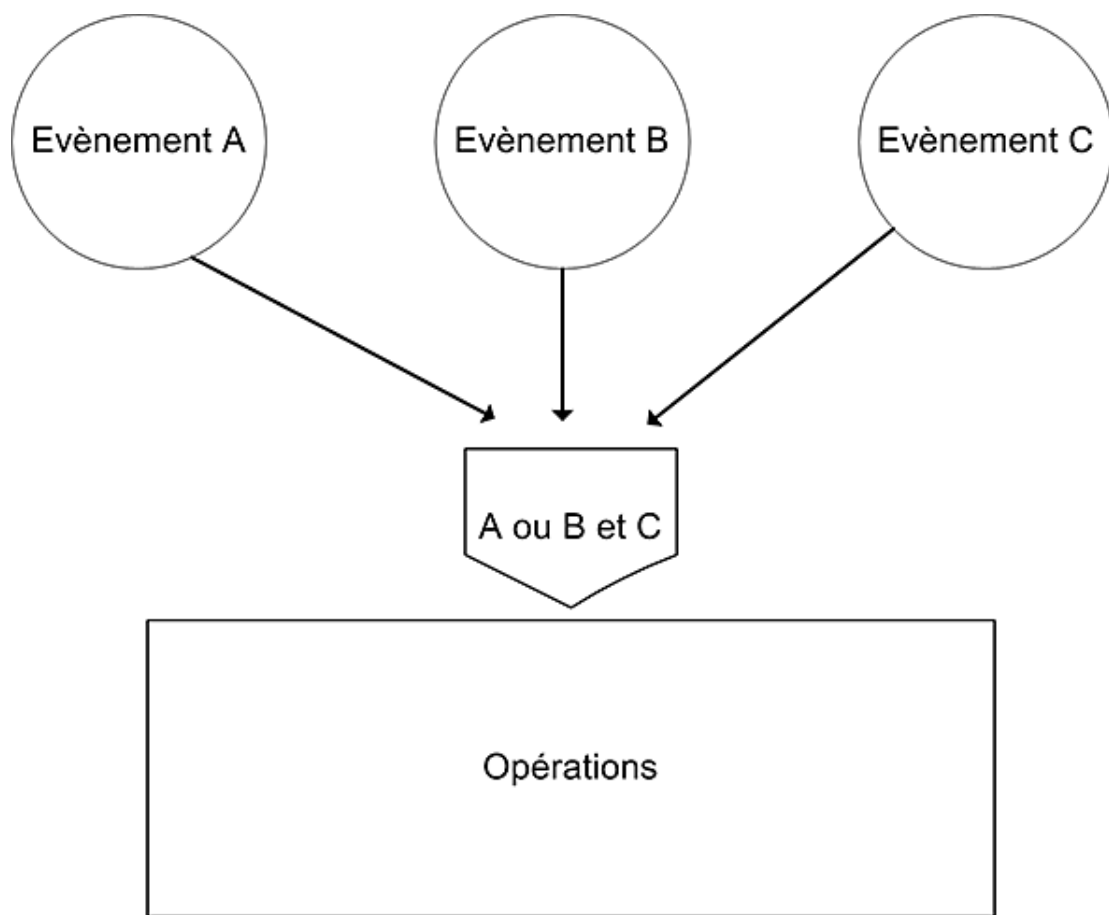


### b. Les opérations

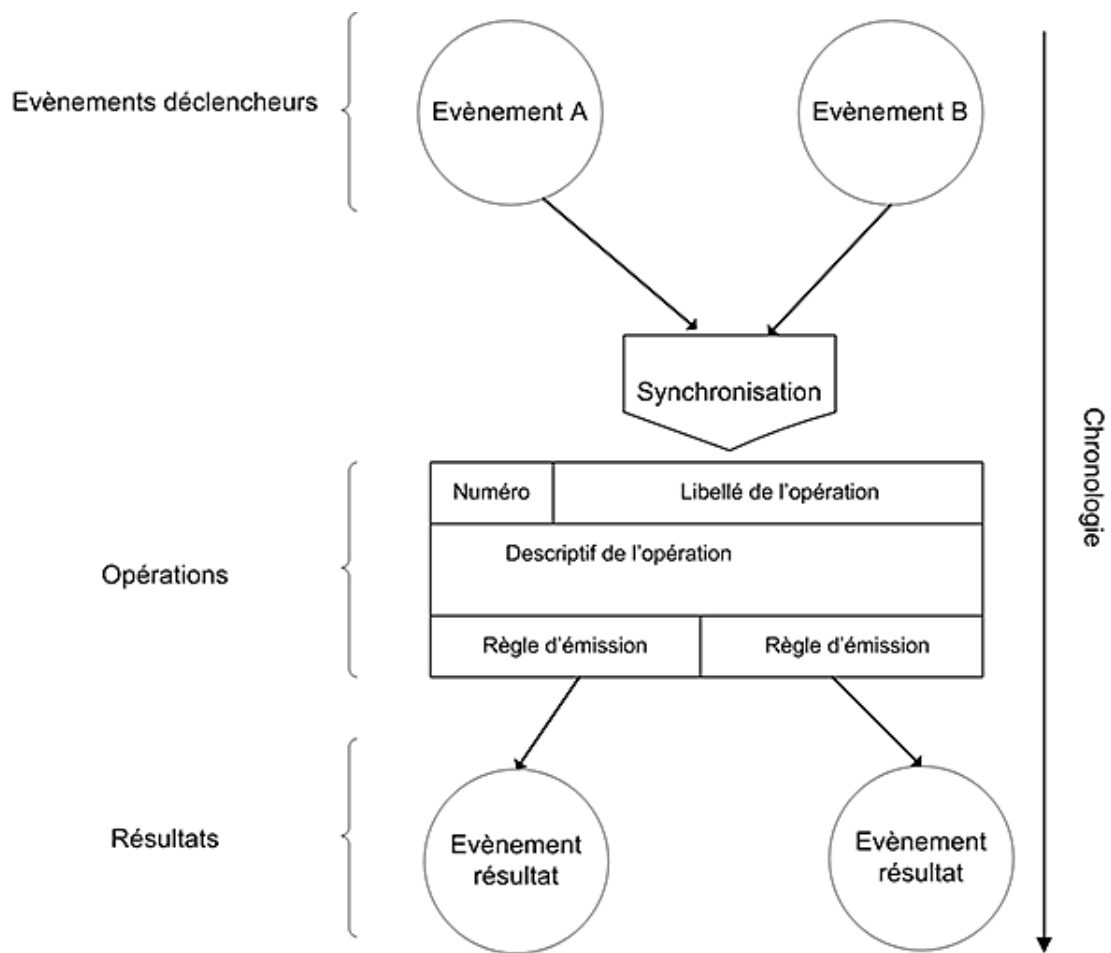
Une opération est une suite d'actions ininterrompibles. Pour trouver les opérations, on se sert du diagramme de flux conceptuel de niveau le plus bas et on décompose les activités en un ensemble d'opérations élémentaires.

### c. La synchronisation

La synchronisation agit au niveau des événements avec des opérateurs logiques : et, ou, non.



**d. Représentation schématique d'un Modèle Conceptuel des Traitements**



## Conception d'un Modèle Conceptuel des Traitements pas à pas

Nous allons nous entraîner sur un exemple :

Dans l'entreprise Baptiste&Co, voici comment sont traitées les commandes des clients et la facturation :

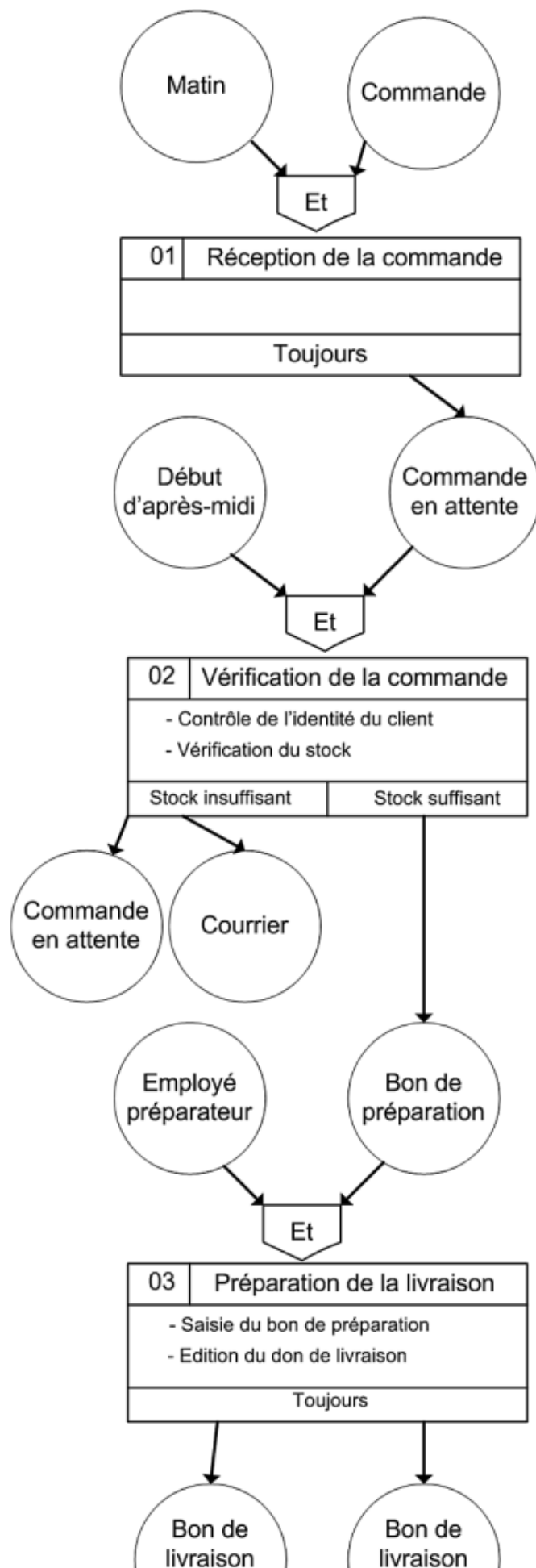
Les commandes des clients arrivent par courrier au service secrétariat, généralement le matin. En début d'après-midi, les commandes sont transmises au service de préparation des livraisons. Le responsable du service des livraisons vérifie l'identité du client et le stock pour les marchandises commandées.

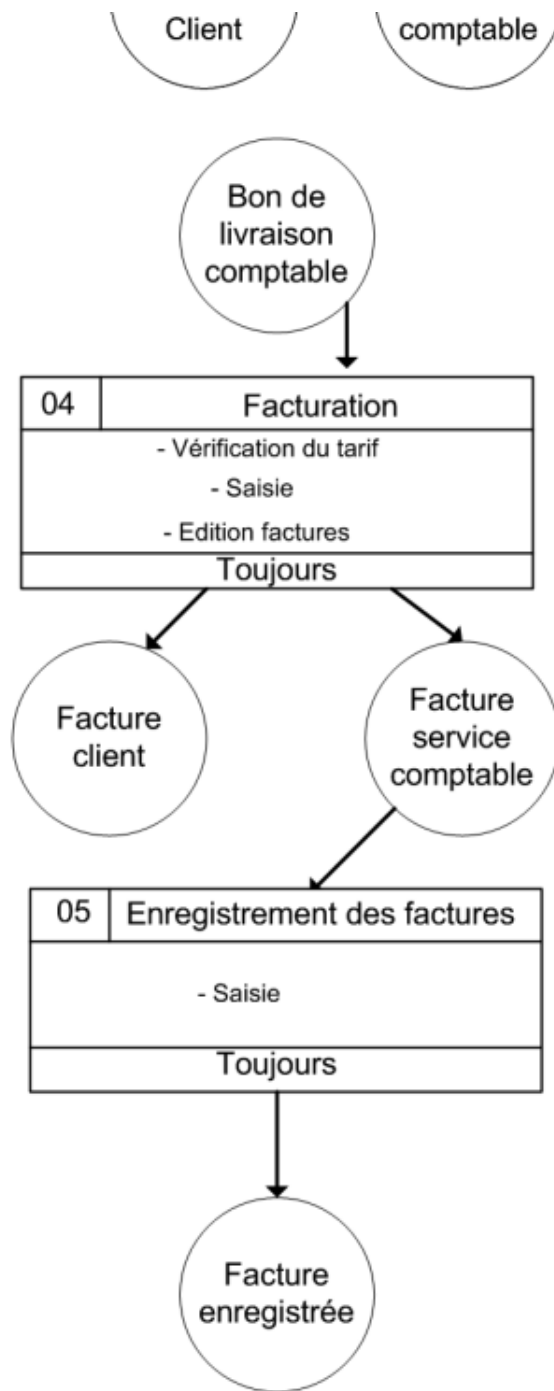
Si les stocks sont suffisants, un bon de préparation est rédigé, sinon il rédige un courrier au client pour l'avertir de l'absence d'un des produits et la commande est mise en attente.

Dans le cas où le stock est suffisant, un employé du service des livraisons prépare la livraison à l'aide du bon de préparation : il prélève et emballe les marchandises, ensuite il saisit les bons de préparation et édite en double exemplaire le bon de livraison dont un exemplaire est adressé au client en même temps que le colis, le deuxième exemplaire étant transmis au service comptable.

À partir du bon de livraison, un employé du service comptable saisit le numéro du bon, vérifie les tarifs et les conditions de règlement et édite la facture en double exemplaire : un exemplaire est adressé au client, l'autre est archivé en attente de comptabilisation.

En fin de semaine, un employé du service comptable récupère l'ensemble des factures en attente de comptabilisation ; pour chacune d'elle, l'employé saisit le numéro de facture et valide les données à l'écran. Après saisie, le grand livre est mis à jour.

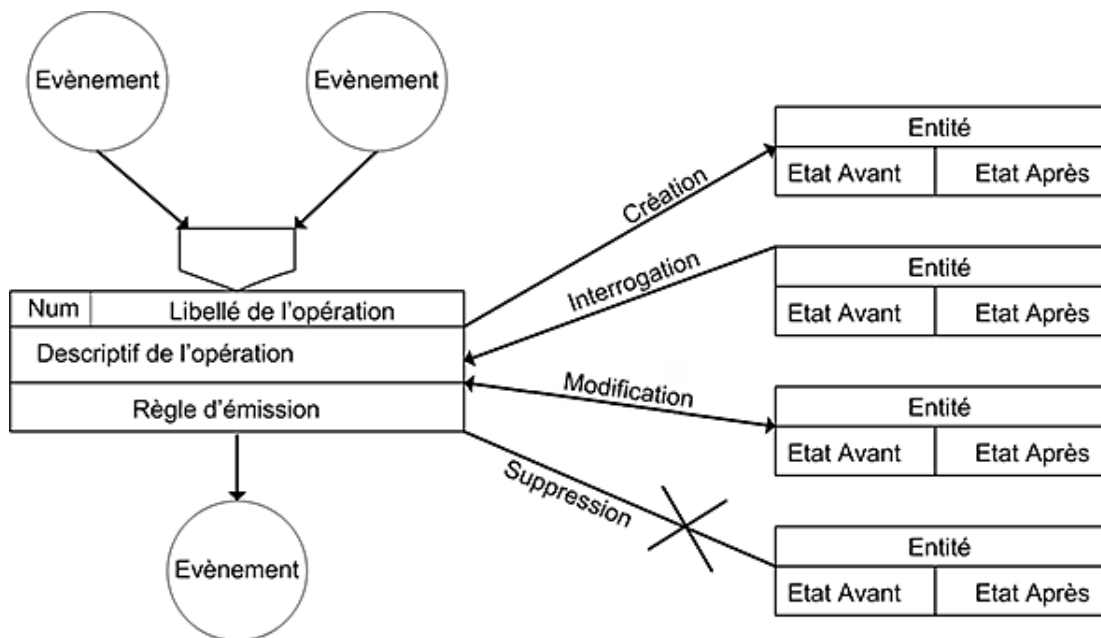






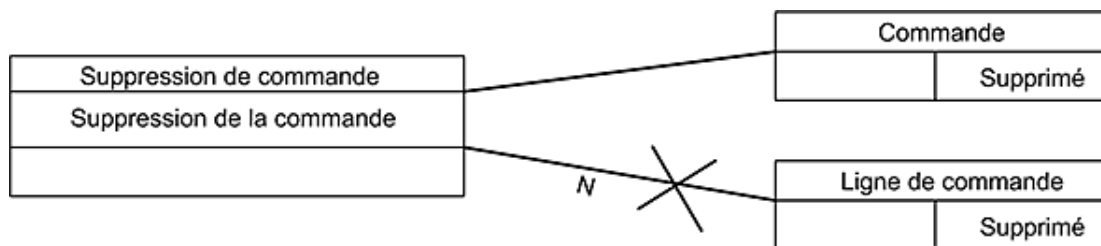
# Le Modèle Conceptuel des Traitements Analytiques

Le Modèle Conceptuel des Traitements Analytiques est un modèle introduit par Merise/2. Il consiste à s'intéresser aux interactions avec les fichiers. En voici une représentation schématique :



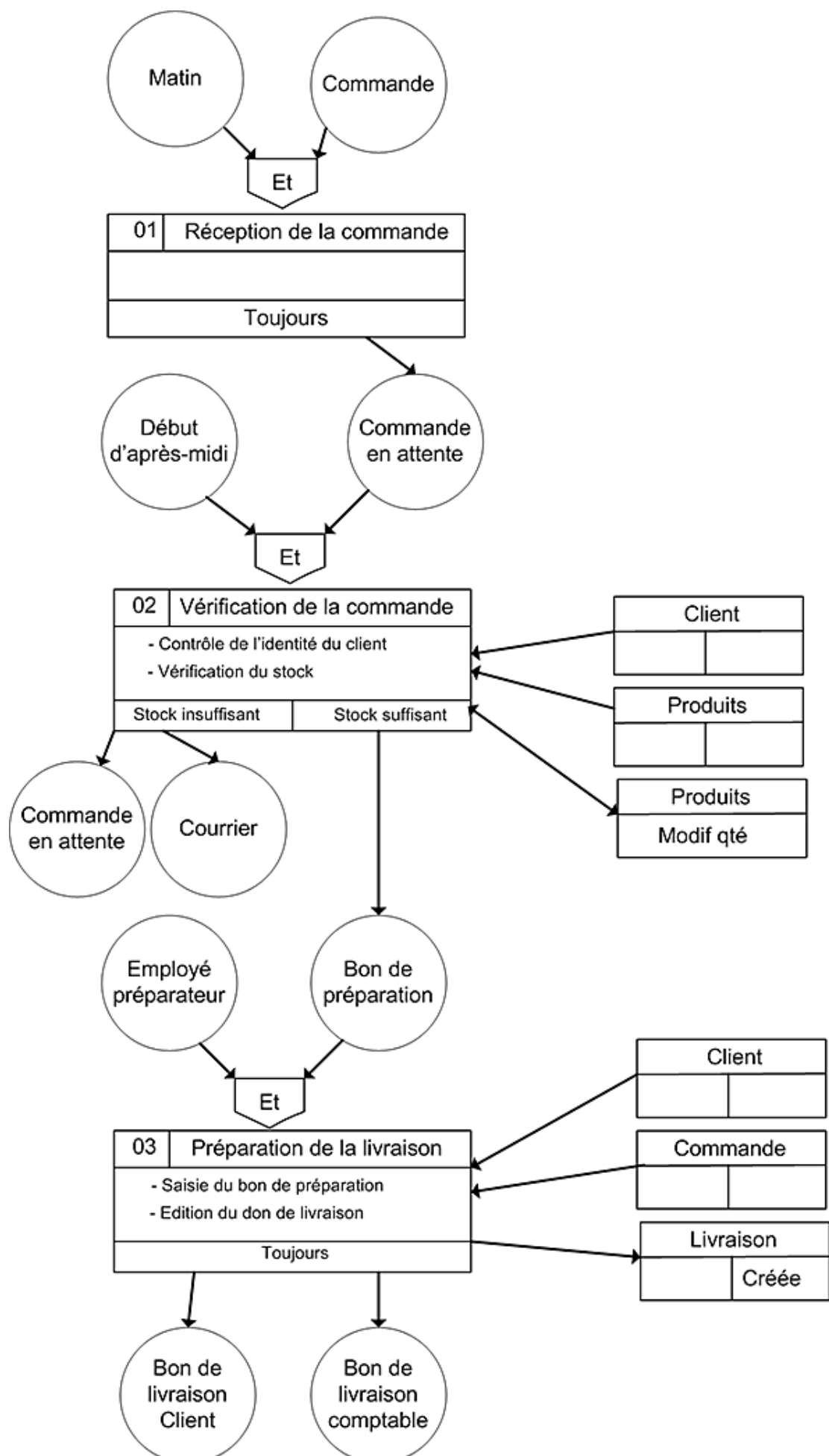
Une action peut agir sur plusieurs occurrences d'une même entité, dans ce cas on exprimera cette action itérative par la lettre N.

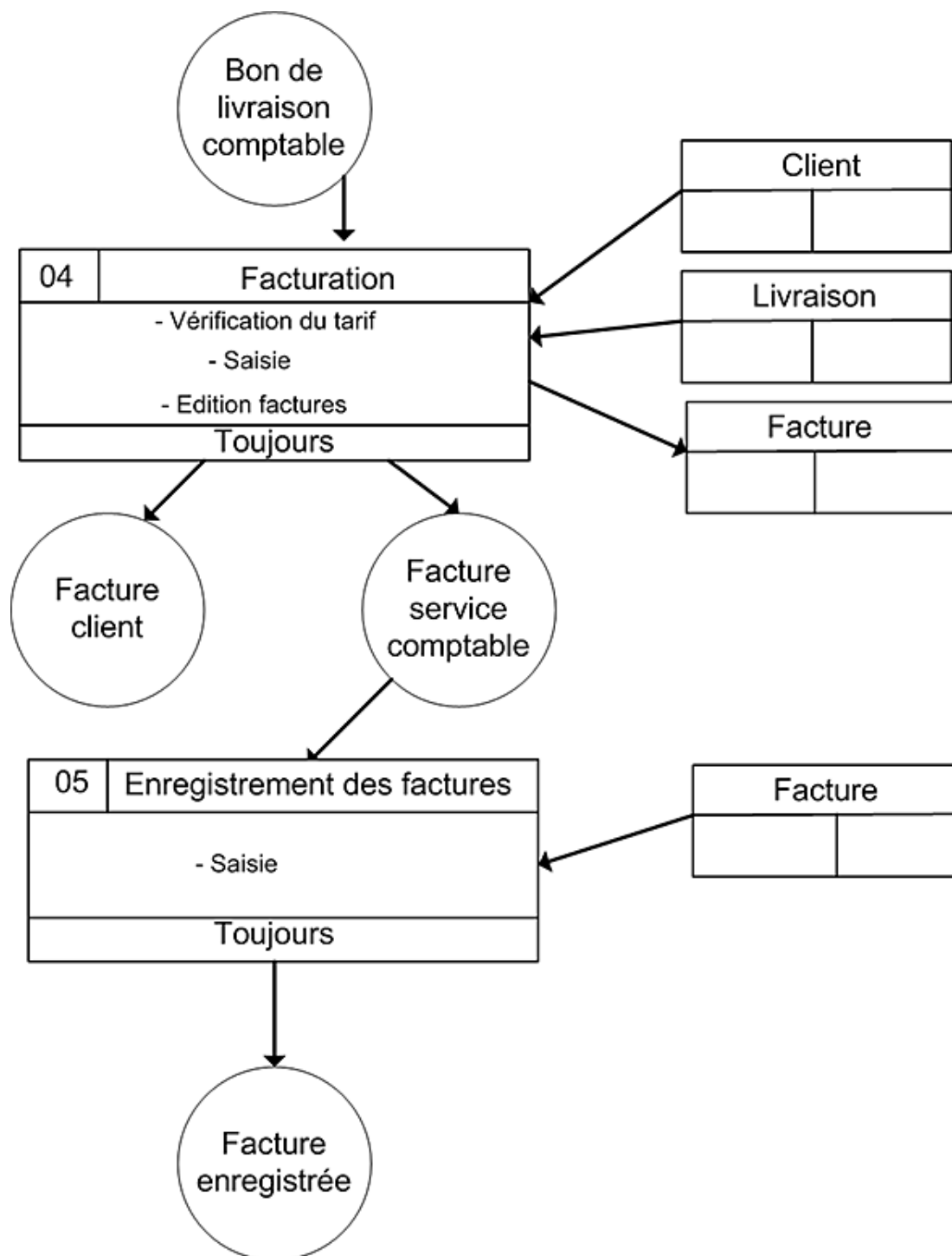
Par exemple :



Dans cet exemple nous constatons qu'il suffit de supprimer une ligne dans le fichier des commandes, mais une commande comprenant plusieurs lignes il est possible de devoir supprimer plusieurs lignes dans le fichier des lignes de commandes.

Appliquons le Modèle Conceptuel des Traitements Analytiques à l'exercice précédent :





## Conclusion

Ces modèles et surtout le Modèle Conceptuel des Traitements Analytiques donnent une représentation complète des traitements, car il fait apparaître les interactions avec les fichiers. L'équipe de développement possède ainsi une vision claire des procédures d'accès ou de modification des fichiers.

# Introduction au Modèle Organisationnel des Traitements

## 1. Objectifs

Il complète la description conceptuelle des traitements en intégrant tout ce qui est d'ordre organisationnel dans le domaine étudié.

Le Modèle Organisationnel des Traitements précise :

- Qui exécute les traitements et la nature des traitements :
  - Manuels,
  - Automatiques,
  - Semi-automatiques.
- Les lieux où sont exécutés les traitements (poste de travail, serveur...).
- Quand sont exécutés les traitements (notion de temporalité).

Le Modèle Organisationnel des Traitements est basé sur trois concepts principaux :

- L'événement.
- La phase ou procédure.
- Le résultat.

Il existe plusieurs représentations ou formalismes du Modèle Organisationnel des Traitements en voici un exemple :



# Conception d'un Modèle Organisationnel des Traitements pas à pas

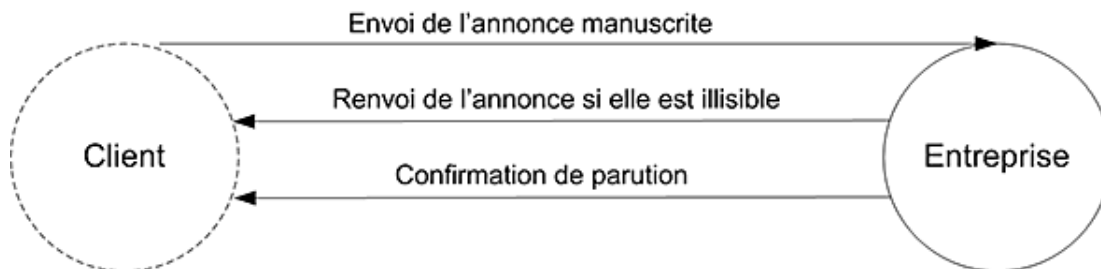
Pour avoir une vue d'ensemble de la méthodologie du traitement des données, nous allons dérouler un cas complet.

Dans l'entreprise Perpi2000 spécialisée dans la publication de petites annonces, les clients envoient de façon manuscrite le contenu de leur petite annonce recopié sur une grille découpée dans le journal.

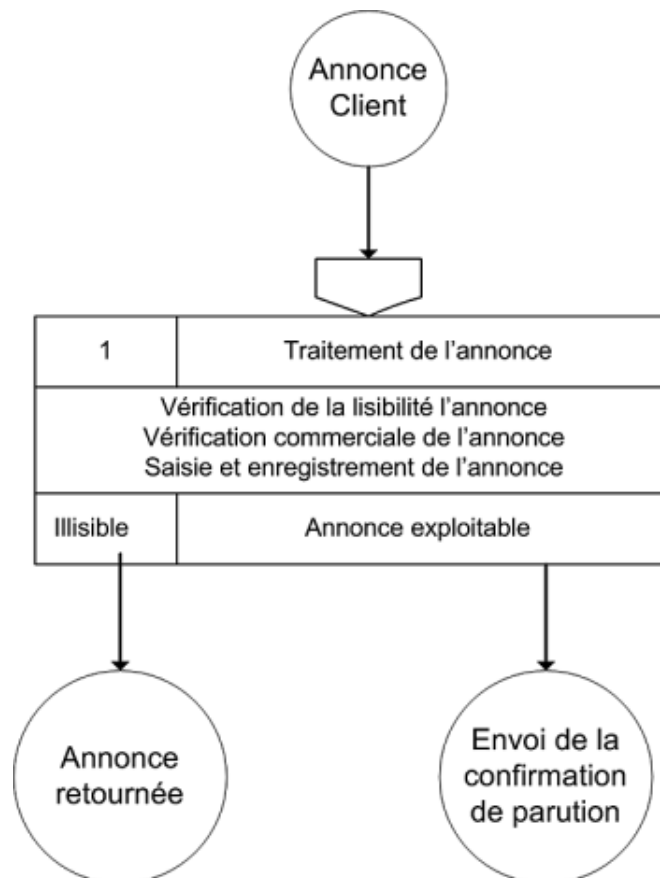
Dès que la petite annonce est arrivée au service secrétariat, un employé vérifie que l'annonce est lisible et correctement exploitable. Dans le cas où l'annonce est illisible, elle est retournée au client avec une lettre type annonçant que l'annonce est illisible.

Dans le cas où l'annonce est exploitable, elle est passée au service commercial qui vérifie la validité du montant dû et ensuite saisit et enregistre l'annonce. Une confirmation de parution est ensuite envoyée au client.

## 1. Le diagramme des flux (ou modèle conceptuel de communication)



## 2. Le Modèle Conceptuel des Traitements



## 3. Le Modèle Organisationnel des Traitements





## Conclusion

Le Modèle Organisationnel des Traitements permet d'avoir une vision claire et précise du déroulement des opérations et de leurs traitements.

# Introduction aux extensions Merise/2

## 1. Présentation

Pour s'adapter aux nouvelles technologies logicielles, notamment la percée des langages orientés objets et du langage UML, la méthode Merise a dû s'enrichir et évoluer.

Au cours de ce chapitre, nous allons étudier les éléments majeurs des extensions Merise, appelées aussi Merise/2.



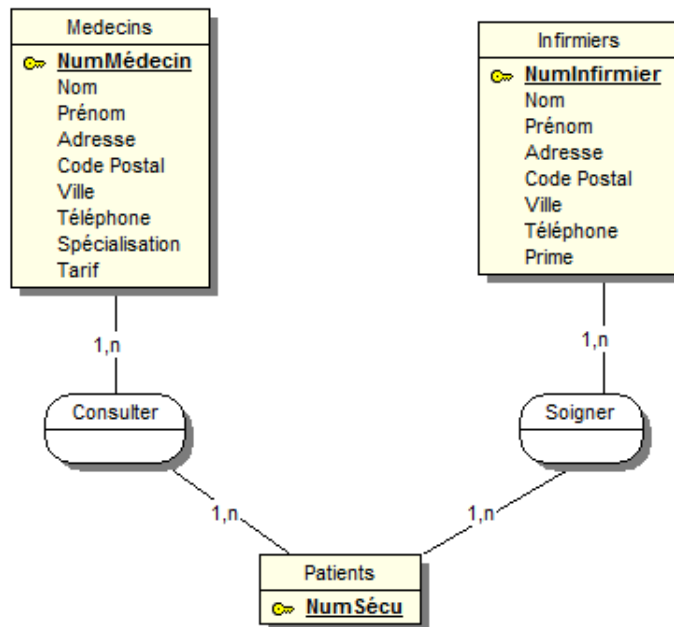
Les identifiants relatifs sont traités dans le chapitre Le Modèle Conceptuel des Données.

## 2. L'héritage (ou la généralisation - spécialisation)

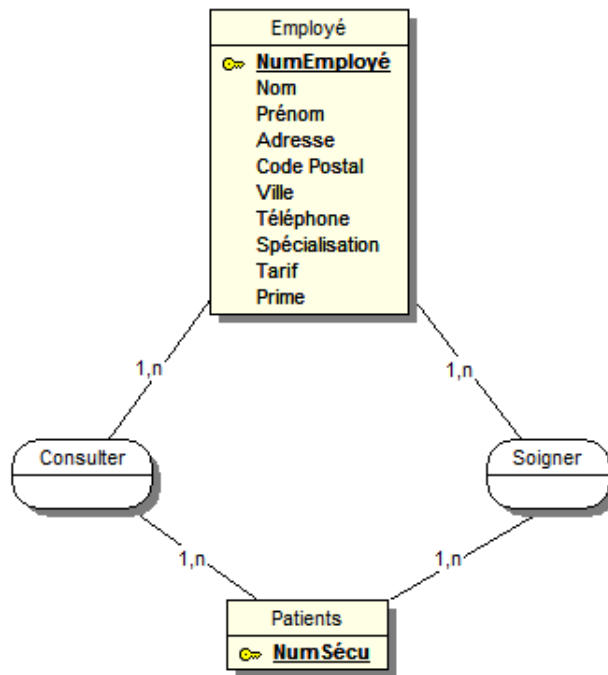
Commençons directement par un exemple.

- Une maison de santé reçoit des patients, deux types de personnel sont salariés : les médecins qui réalisent les consultations et les infirmiers administrent les soins.
- Les médecins ont une spécialisation (médecin du sport, gérontologue...) et un tarif à l'acte.
- Les infirmiers ont des primes d'astreintes.
- Les patients sont juste référencés par leur numéro de sécurité sociale.

Imaginons le MCD suivant :



Ce MCD pourrait aussi être représenté sous cette forme :



Ces deux solutions, correctes dans les faits, présentent des inconvénients structurels.

Si nous regardons le premier modèle conceptuel, nous pouvons voir qu'il y a des attributs dupliqués entre les deux entités médecins et infirmiers (le nom, le prénom, l'adresse...).

Si nous étudions le deuxième modèle conceptuel, nous nous rendons compte que certaines rubriques seront vides :

- Spécialisation et tarif pour les infirmiers.
- Prime pour les médecins.

Pour résoudre ces problèmes, nous allons utiliser **l'héritage**.

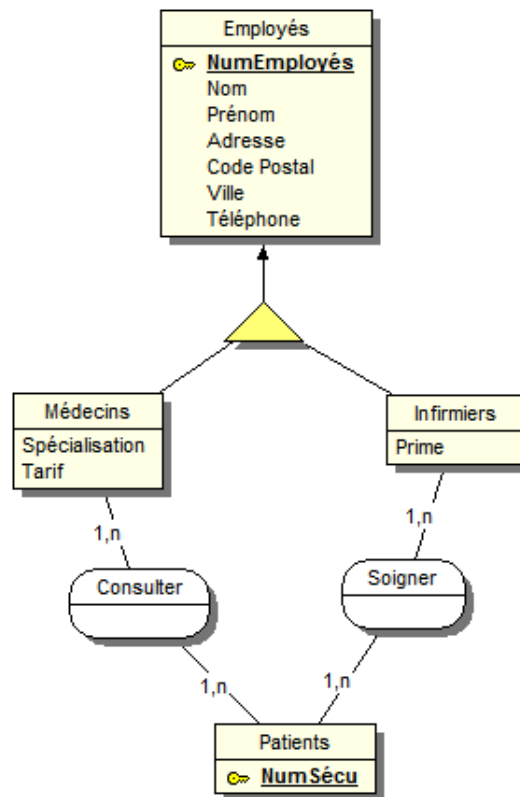
Le principe global de l'héritage est de factoriser les propriétés identiques dans une entité commune. Cette entité commune est aussi nommée **entité générique** ou **sur-type** d'entité.

Les propriétés spécifiques seront contenues dans une entité spécialisée nommée aussi **sous-type**.



Chaque sous-type hérite des propriétés et des associations du sur-type. La relation qui fait correspondre un sous-type à son sur-type est une **généralisation**. La relation inverse est une **spécialisation**.

Voici l'illustration du principe d'héritage :



Maintenant, tous les attributs communs entre Médecins et Infirmiers sont les propriétés de l'entité générique Employés.

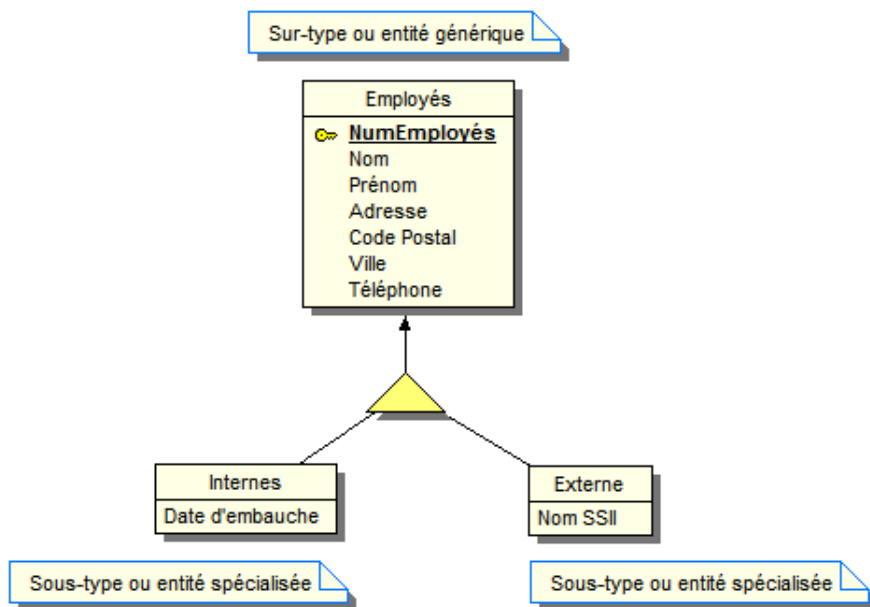
Médecins et Infirmiers sont des entités spécialisées d'Employés. Chacune aura comme propriétés ses propres attributs.

L'identifiant de Médecins et d'Infirmiers est celui d'Employés : les entités spécialisées héritant des propriétés de l'entité générique, on ne fait pas apparaître l'identifiant des entités spécialisées.

#### Autre exemple :

*Une multinationale gère ses salariés ainsi que des salariés de SSII.*

*Voici le modèle conceptuel qui en découle.*



# Contraintes ensemblistes

Les extensions du modèle entité-association permettent de représenter des contraintes sur des ensembles **d'occurrences d'entités ou d'associations**.



L'entité concernée par la contrainte est appelée le **pivot** de la contrainte.

L'ensemble des contraintes est formé à partir de deux contraintes de base :

- La contrainte de couverture.
- La contrainte de disjonction.

## 1. La contrainte de couverture

Toute occurrence de l'entité générique appartient au moins à l'un des sous-types.

## 2. La contrainte de disjonction

Toute occurrence de l'entité générique doit appartenir à un seul sous-type : les sous-types sont mutuellement exclusifs.

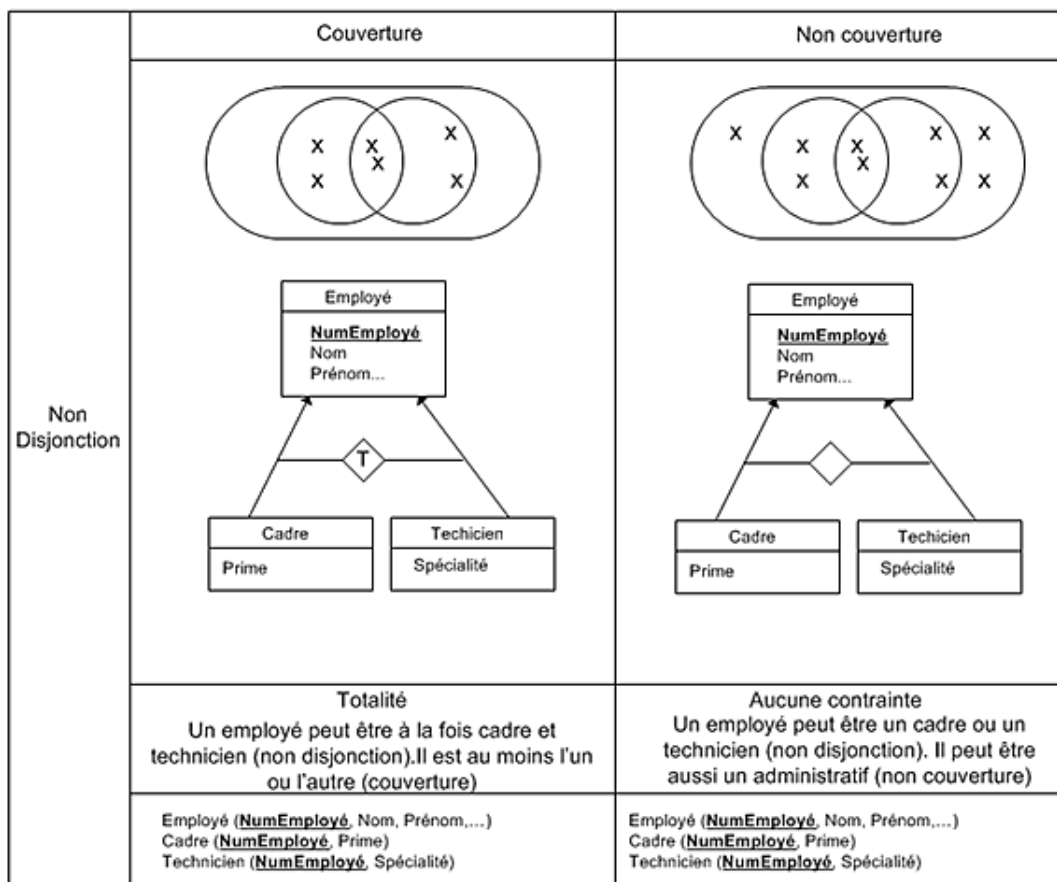
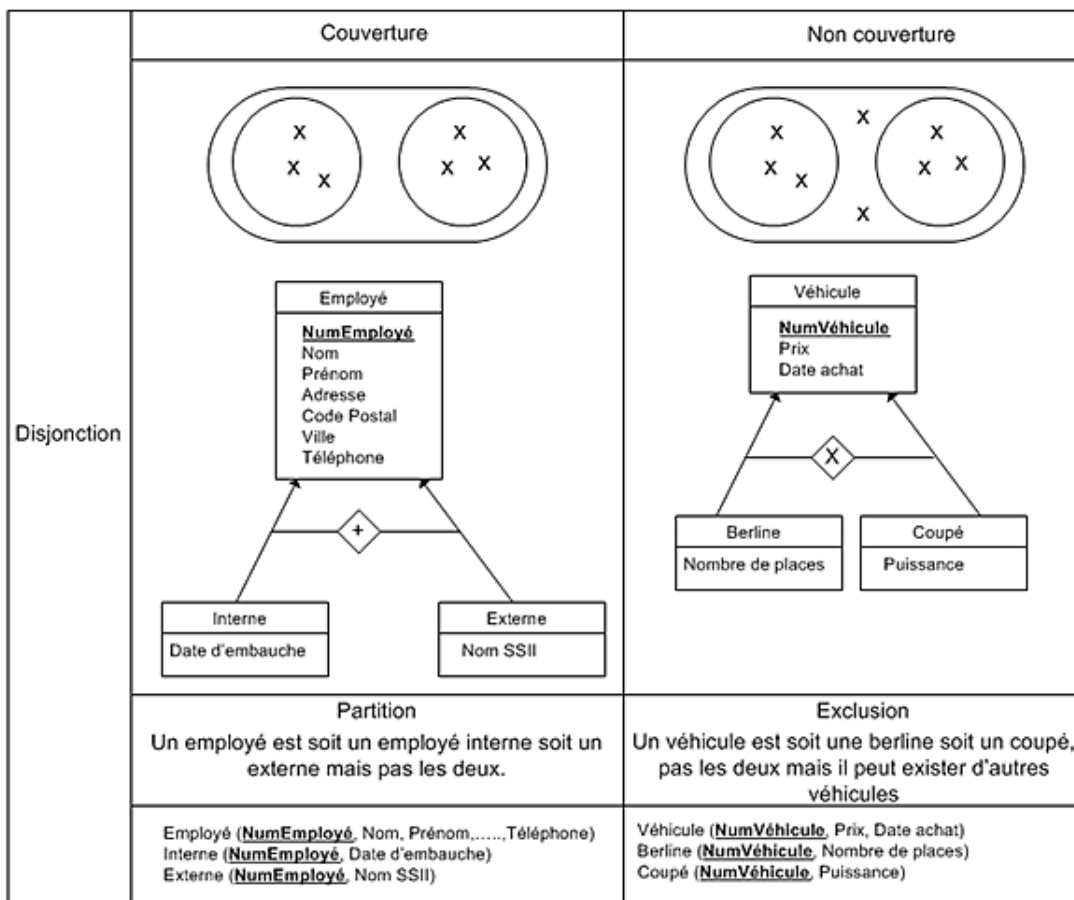
Exemple :

*Un employé est soit un employé interne soit un employé d'une SSII mais pas les deux.*

Si nous combinons ces deux types de contraintes, nous obtenons les **quatre cas** suivants :

- Une occurrence du sur-type n'appartient à **aucun** sous-type, à **un** sous-type ou à **plusieurs** sous-types. Il y a donc ni couverture ni disjonction. Dans ce cas-là, il n'y a pas de contrainte à formaliser.
- Une occurrence du sur-type appartient *toujours* à **un** sous-type, *éventuellement* à **plusieurs**. Dans ce cas il y a donc couverture, mais pas de disjonction. C'est formalisé par une contrainte de **totalité**, notée **T**.
- Une occurrence du sur-type appartient *toujours* à **un** sous-type et à **un** seul. Ici, nous avons l'expression d'une couverture et d'une disjonction. C'est formalisé par une contrainte de **partition** notée : **XT ou +**. De façon mnémonique nous pouvons retenir dans ce cas que c'est « l'un ou l'autre, mais pas les deux ».
- Une occurrence du sur-type appartient *éventuellement* à **un** et **un** seul sous-type. Il y a donc disjonction, mais pas de couverture. C'est formalisé par une contrainte **d'exclusion** : **X**. Nous pouvons l'exprimer par « l'un ou l'autre, mais pas les deux, ou aucun ».

Voici un récapitulatif :



## La formalisation des contraintes entre associations

Nous avons vu comment formaliser les contraintes entre entités, voyons maintenant comment formaliser des contraintes entre associations.

Nous retrouvons les quatre contraintes précédemment décrites pour les sous-types :

- La totalité : couverture + non-disjonction.
- La partition : couverture + disjonction.
- L'exclusion : non-couverture + disjonction.
- Aucune contrainte : non couverture + non disjonction.

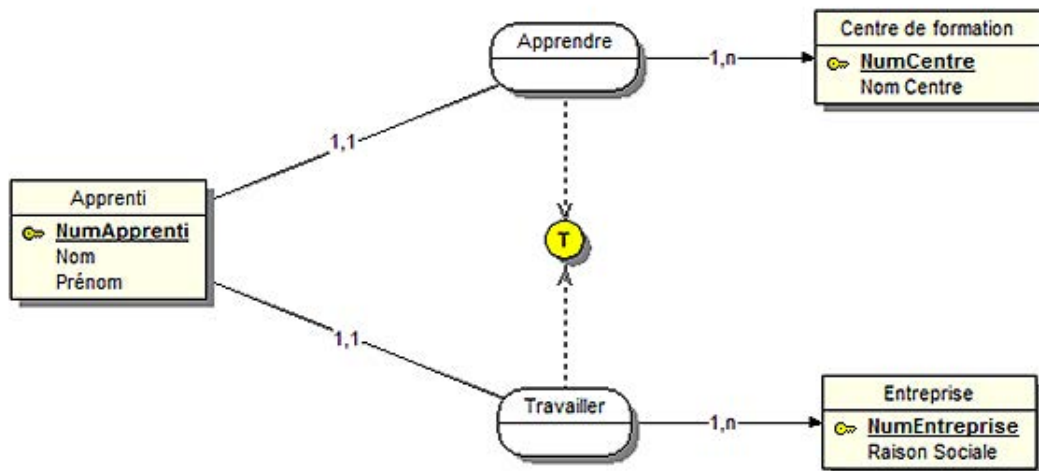
À ces contraintes s'ajoutent :

- Une contrainte d'égalité ou de simultanéité : toute occurrence qui participe à l'association A, participe également à l'association B ; se note = ou S ;
- Une contrainte d'inclusion : toutes les occurrences d'une association A sont également occurrences d'une association B ; se note I.

Étudions cela avec quelques exemples.

### 1. La totalité (couverture + non-disjonction)

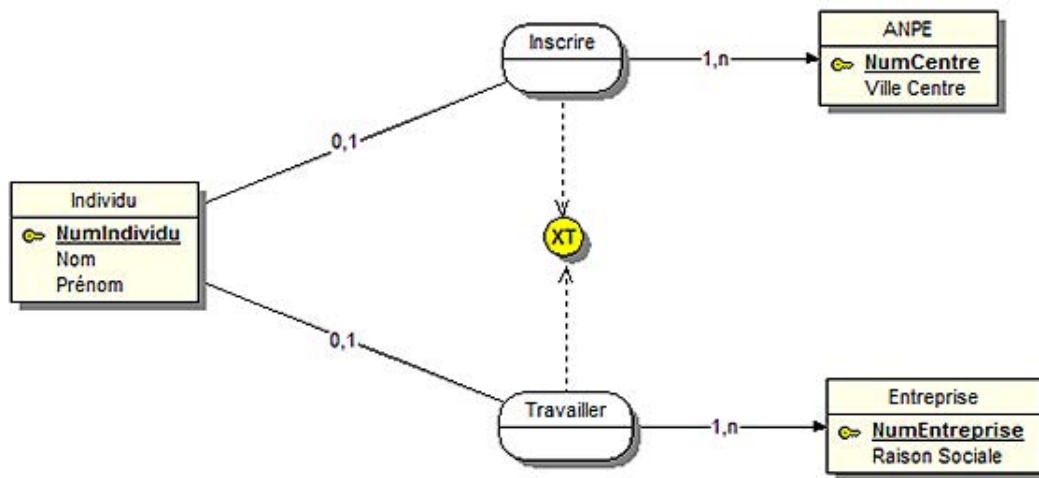
Prenons l'exemple d'un apprenti qui travaille dans une entreprise et se forme dans un centre de formation :



Si nous interprétons la contrainte d'intégrité nous voyons qu'un apprenti participe à au moins une des deux associations, car il y a couverture et non-disjonction. En fait, soit il travaille dans une entreprise, soit il étudie et apprend.

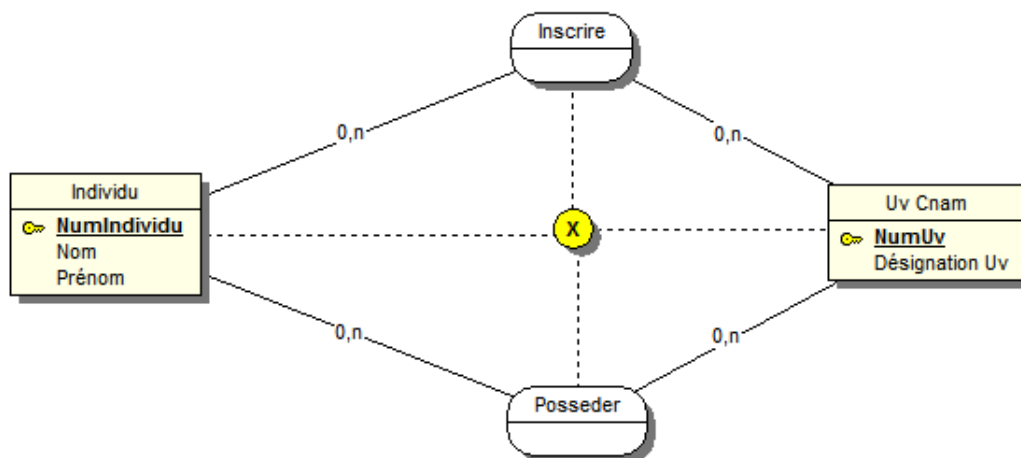
### 2. Partition (couverture + disjonction)





Un individu participe nécessairement et exclusivement à une des deux associations (couverture et disjonction). Soit une personne travaille, soit elle est au chômage.

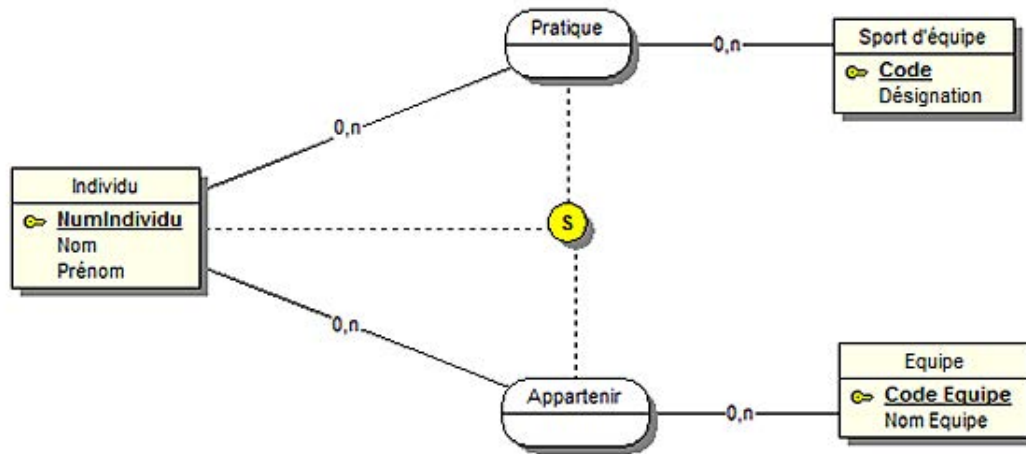
### 3. Exclusion (non-couverture + disjonction)



Ici le pivot implicite est entre les entités Individu et Uv Cnam. Nous pouvons interpréter ceci de la façon suivante : Un individu ne peut, à la fois être inscrit à une UV et la posséder.

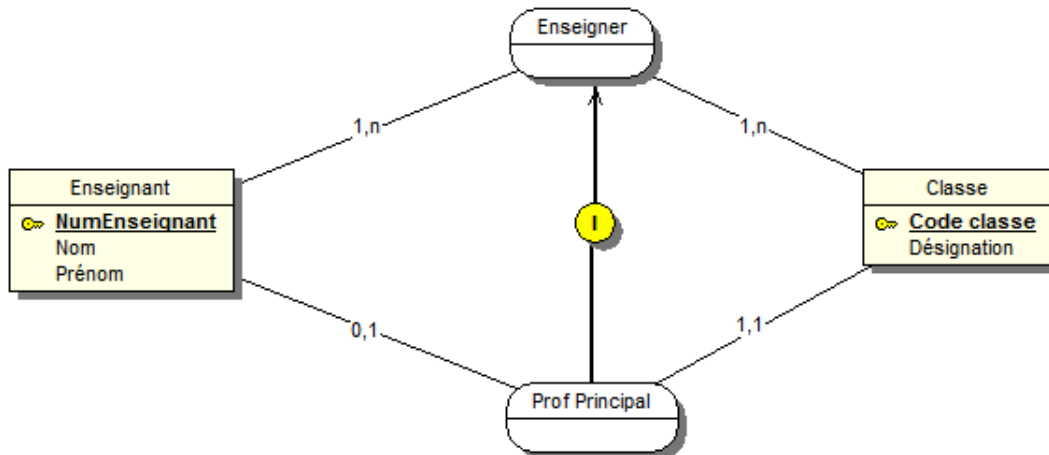
### 4. Égalité

Cette contrainte est également nommée contrainte de simultanéité.



Toute personne qui pratique un sport appartient à une équipe et vice versa. L'ensemble des occurrences du pivot est donc la même pour chacune des associations participant à la contrainte.

## 5. Inclusion



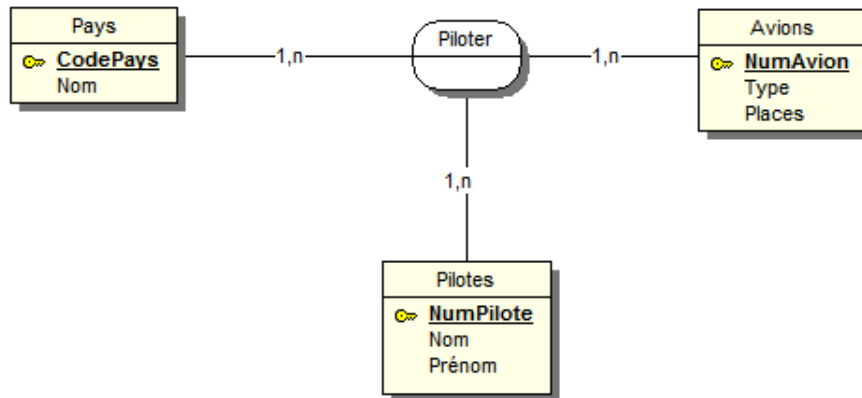
Elle traduit le fait que **toutes les occurrences d'une association sont également occurrences d'une autre.**

Dans notre cas, un enseignant ne peut être professeur principal que d'une classe où il enseigne.

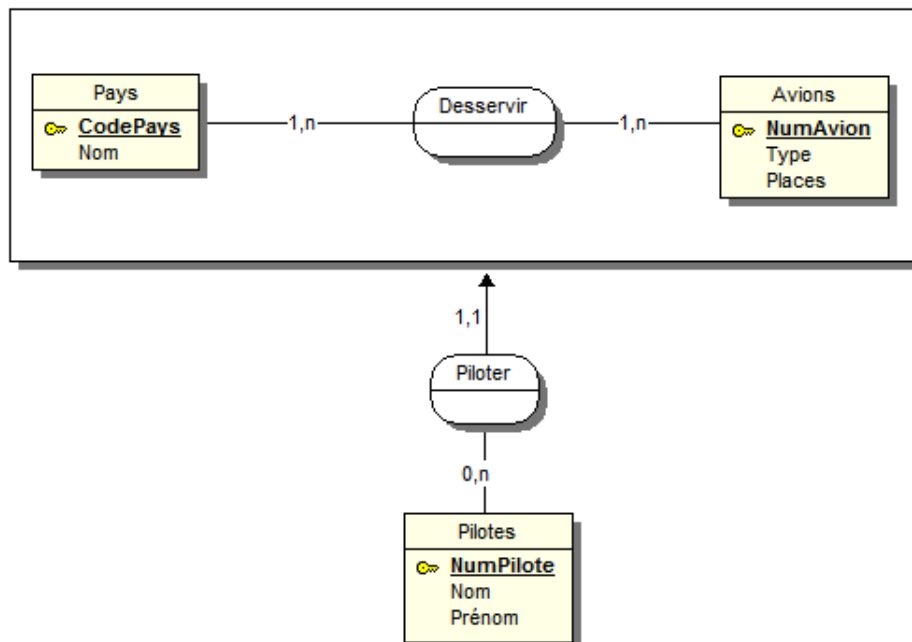
## Agrégation sur les associations (ou associations d'associations)

Prenons le cas suivant :

Dans la compagnie d'aviation « Vol'toujours », les pilotes font voler des avions dans différents pays. Mais un avion pour un pays donné n'est piloté que par un seul pilote. Cette règle de gestion est absurde, mais elle va nous permettre d'illustrer les agrégations.

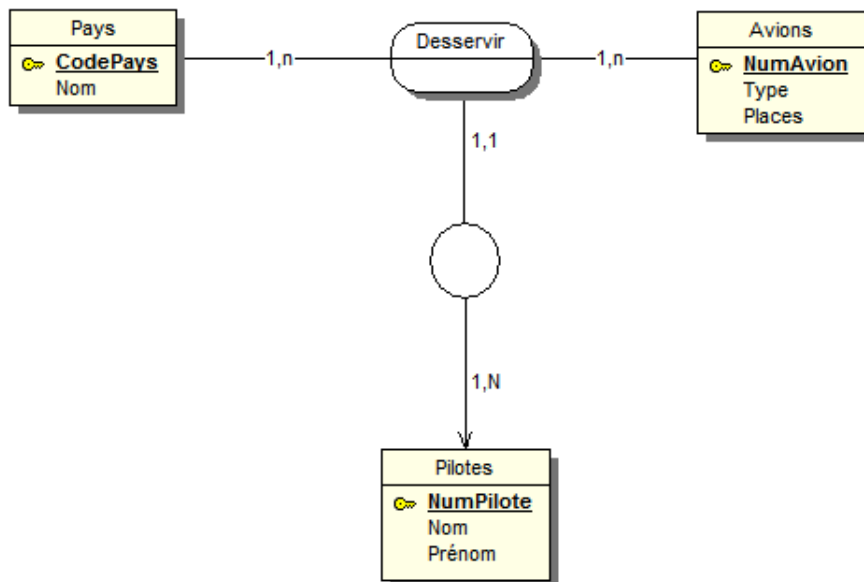
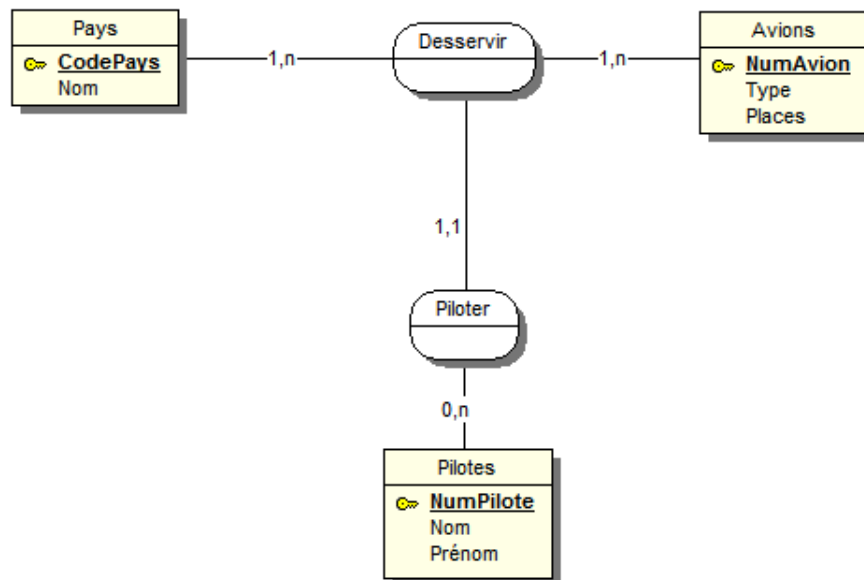


Voici un modèle conceptuel exprimant parfaitement le fait que les pilotes peuvent faire voler des avions dans différents pays. Mais la règle de gestion n'est pas intégrée et difficilement intégrable sans l'utilisation de l'agrégation. Voici une solution intégrant la représentation de l'agrégation :



L'agrégation consiste à percevoir comme un objet unique l'association de plusieurs autres objets. Ici ce qui est encadré représente un ensemble.

D'autres représentations peuvent être proposées, mais elles sont moins explicites que celle vue précédemment.



# Introduction au cycle de vie des objets

En évoluant Merise s'est inspiré de certains concepts des méthodologies dites orientées objets. Comme nous l'avons vu, le modèle entité-association a évolué pour prendre en considération, dans les modèles de données, le sous-typage d'entité, la spécialisation, la généralisation. La notion d'état (voir chapitre suivant sur la comparaison entre la méthode Merise et UML) est intégrée dans la modélisation des traitements avec l'élaboration des cycles de vie des objets (CVO).

## 1. Les objectifs de la gestion du cycle de vie d'un objet

Les objectifs du CVO sont de mettre en évidence l'ensemble des états remarquables d'un élément de gestion (une entité par exemple) au cours de son cycle de vie. Le CVO permet d'identifier les événements qui transforment les états, d'illustrer les interactions des changements d'états. Il permet d'affiner par son approche les règles de gestion du MCD et prépare la construction du MCTA.

La plupart des formalismes retenus dans les méthodes objets utilisent les concepts suivants :

- État de l'objet
- Événement
- Transition

### a. État de l'objet

L'état est un palier transitoire par lequel passe un objet au cours de son cycle de vie.

L'état peut correspondre à des choix de gestion. Par exemple, « En stock » ou « En commande » peuvent être les états d'un objet Produit.

Une occurrence de l'individu ne peut être dans plusieurs états à la fois.

### b. Événement

Un événement (externe, interne, ou temporel) déclenche la transition d'un état à un autre état.

### c. Transition

Passage de l'objet d'un état à un autre. Elle peut être soumise à une condition.

Formalisme et représentation graphique :

- Les états sont représentés par un carré.
- Les événements qui déclenchent le passage d'un état à l'autre sont représentés par un ovale.

Ces deux objets, l'état et l'événement, sont reliés par des flèches orientées représentant les transitions entre états, une flèche reliant un événement à un état ou réciproquement. Une même flèche ne peut relier deux états ou deux événements.

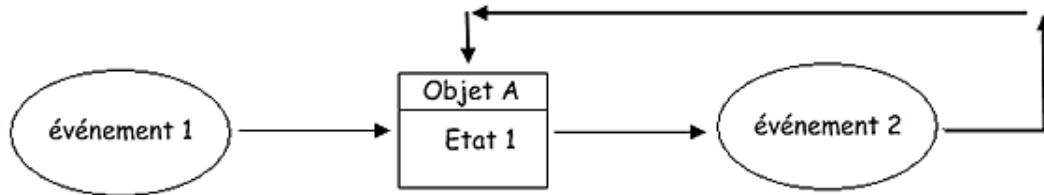


➤ Le lien entre l'état de l'objet A et l'événement 2 ne signifie pas qu'il y a un déclenchement de l'événement. Il signifie simplement que l'événement 2 est pris en compte quand il survient après mise en état 1 de l'objet A.

## 2. Mise en œuvre

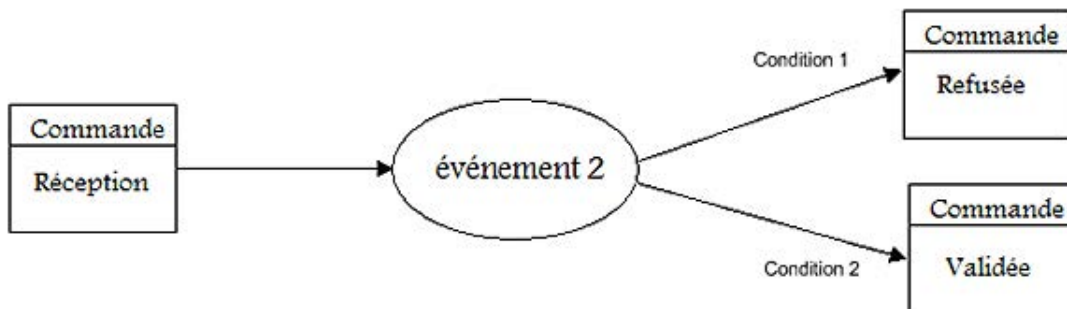
### a. L'itération

Un événement peut absorber un objet ayant un état défini et le restituer dans le même état sans qu'aucune modification n'ait eu lieu.



### b. La transition conditionnelle

La transition d'un état à l'autre peut être soumise à une condition.

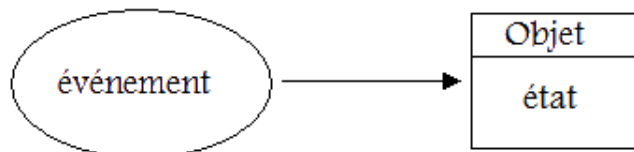


Condition 1 : L'acompte est insuffisant.

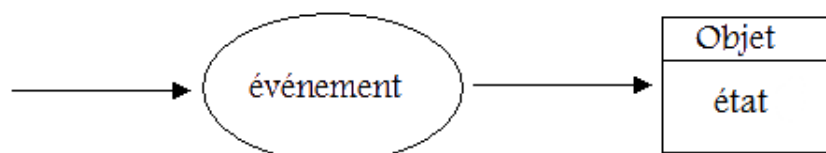
Condition 2 : L'acompte est correct.

### c. La création

Tout événement (généralement il y en a un seul par CVO) non but d'une flèche est l'événement de création qui fait passer l'objet dans tel ou tel état initial.



Mais si la création de l'objet se fait par un événement qui peut être également but d'une flèche, il faudra lui faire arriver une flèche supplémentaire sans origine pour indiquer qu'il s'agit de l'événement de création.



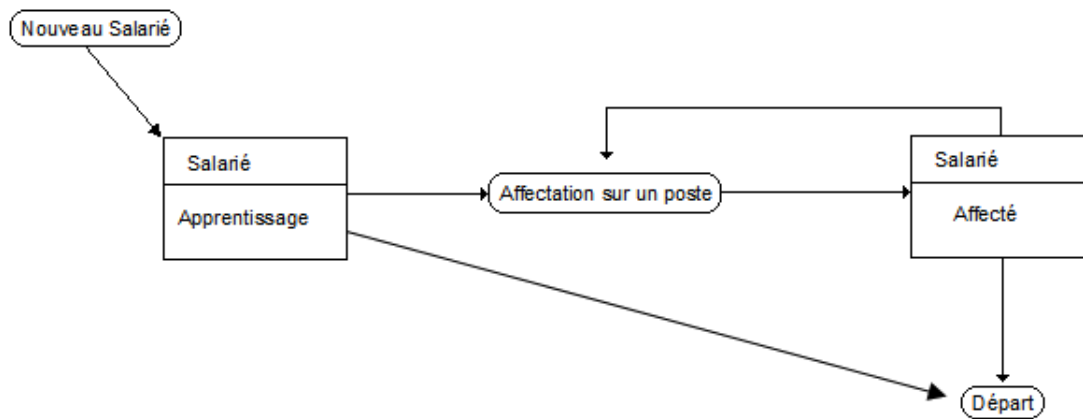
### d. La suppression

À l'inverse de la création, la suppression de l'objet se représente de la façon suivante :



## Conception d'un CVO pas à pas

Nous allons étudier un exemple simplifié de cycle de vie de l'objet salarié :



L'arrivée d'un nouveau salarié passe l'état de l'objet Salarié à Apprentissage. À la fin de l'apprentissage, il peut y avoir deux options :

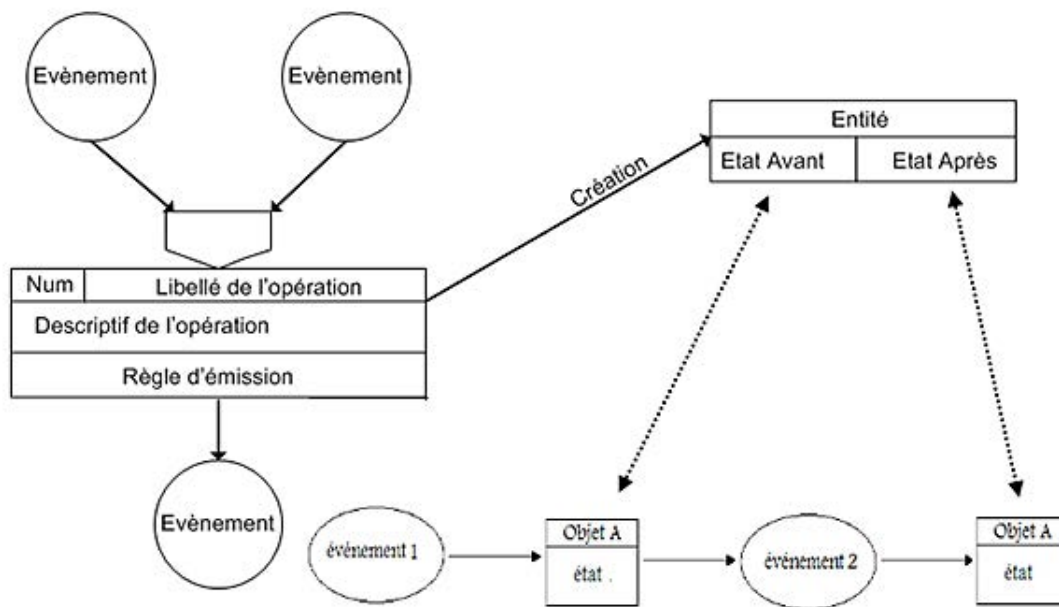
- Soit le salarié est affecté sur un poste.
- Soit le salarié s'en va de l'entreprise.

Un salarié peut avoir durant sa carrière plusieurs affectations ce qui explique l'itération il peut partir de l'entreprise.



## Positionnement du CVO par rapport au MCTA

Mise en œuvre du CVO en parallèle avec le MCTA :



Si l'on voulait mettre en lumière le cycle de vie d'un objet, nous pourrions le voir de cette façon. Sur le modèle conceptuel des traitements analytiques, une entité a un état avant et un état après. Sur le CVO, cela correspond à un état distinct de l'objet A.

## 1. Présentation d'UML

UML (*Unified Modeling Language* ou « langage de modélisation unifié ») est un métalangage de modélisation. Il est une synthèse de certaines méthodes de modélisation objet (OMT, Booch et OOSE).

UML a été normalisé en 1997 par l'OMG (*Object Management Group*). Son but est de formaliser les concepts orientés objet au travers de diagrammes. L'OMG a normalisé depuis novembre 2007 la version UML 2.1.2, et prépare la version 2.2.

Les premières versions d'UML proposaient neuf diagrammes spécifiques :

- **Le diagramme de cas d'utilisation** qui représente les relations entre les acteurs et les fonctionnalités du système.
- **Le diagramme de classes** est un ensemble d'éléments statiques qui montre la structure du modèle étudié.
- **Le diagramme d'objets (objet : instance d'une classe)** représente les objets et leurs interdépendances.
- **Le diagramme d'états/transitions** représente le cycle de vie des objets générés par une classe.
- **Le diagramme de composants** détaille les éléments logiciels (exécutables, fichiers...) et leurs dépendances.
- **Le diagramme de déploiement** montre la répartition physique des éléments matériels du système (processeurs, périphériques) et leurs connexions.
- **Le diagramme de séquence** détaille les messages échangés entre les acteurs et le système selon un ordre chronologique.
- **Le diagramme de collaboration** qui représente les messages échangés entre les objets. Il insiste plus particulièrement sur la notion organisationnelle.
- **Le diagramme d'activités** est une variante du diagramme états/transition qui représente le déclenchement d'événements selon certains états du système.

La version 2 d'UML propose treize diagrammes dépendants hiérarchiquement et se complétant.

### a. Les diagrammes statiques

- Diagramme de classes
- Diagramme d'objets
- Diagramme de composants
- Diagramme de déploiement
- Diagramme de paquetages
- Diagramme de structure composite

### b. Les diagrammes comportementaux

- Diagramme des cas d'utilisation


- Diagramme états-transitions
- Diagramme d'activité

### **c. Les diagrammes dynamiques**

- Diagramme de séquences
- Diagramme de communication
- Diagramme global d'interaction
- Diagramme de temps

# Merise par rapport à UML

---

 Ce chapitre ne développera pas le langage UML, juste les analogies entre certains diagrammes ou modèle que comportent UML et Merise. Il existe d'excellents ouvrages spécialisés sur UML, ce chapitre ne fera que mettre en parallèle Merise et UML en partant du postulat que le lecteur a quelques bases en UML.

---

Il y a les pro-Merise et les pro-UML. Une barrière infranchissable semble les séparer, pourtant l'association des deux pourrait être redoutable.

Merise est une méthode de conception de système informatique particulièrement efficace pour la représentation et la modélisation des bases de données.

Merise repose sur quelques principes fondamentaux :

- Une approche systémique.
- La séparation des données et des traitements.
- Une approche qui part du général vers le particulier.

## **Pour l'approche systémique :**

Merise découpe l'entreprise en trois sous-systèmes :

- Le système de décision.
- Le système d'information.
- Le système opérant.

Dans UML, l'approche par les cas d'utilisation peut constituer une approche systémique.

## **1. Une séparation des données et des traitements**

Comme nous l'avons vu la méthode Merise est caractérisée par une approche conjointe des données et des traitements qui est formalisée par des modèles spécifiques. Le côté statique du système est décrit par les modèles de données tandis que le côté dynamique est mis en œuvre par les modèles de traitement.

UML décrit comme Merise les données et les traitements, mais contrairement à Merise qui les sépare UML les associe. Cela est dû à l'approche objet d'UML, approche objet qui agrège les données et les traitements.

### **Du général vers le particulier**

Nous avons bien ressenti que la méthode Merise au travers de ses modèles partait d'une vision globale du système d'information qui était affinée au fur et à mesure de la progression dans la réalisation de la méthode.

UML pour sa part permet une approche aussi bien descendante qu'ascendante.

# Analogie Merise/UML

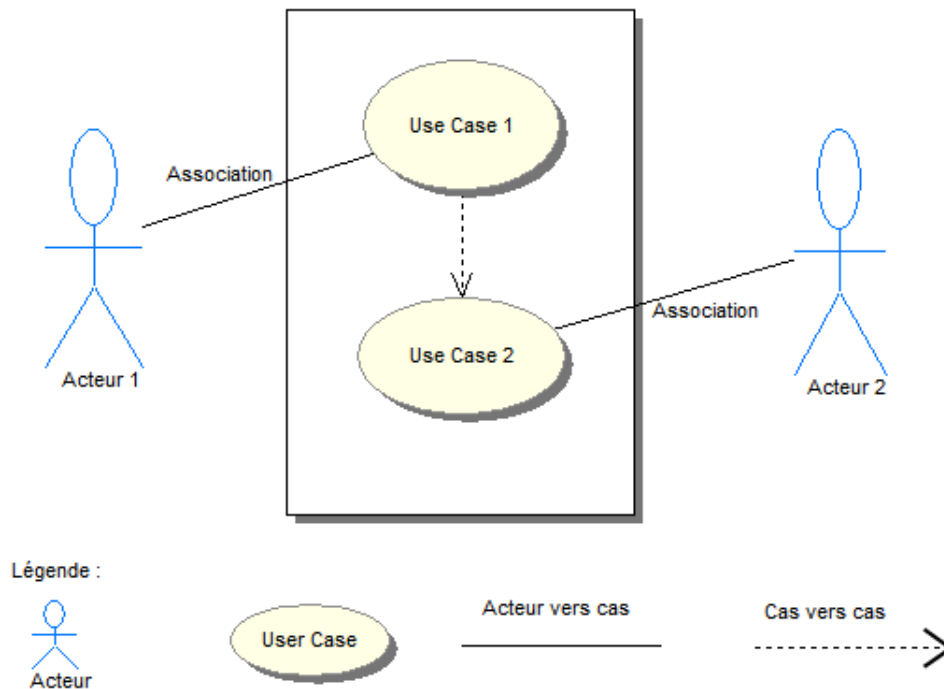
## 1. Modèle de contexte (diagramme des flux) - Diagramme des cas d'utilisation

Le modèle de contexte (ou diagramme des flux) de Merise est souvent utilisé au démarrage d'un projet car il permet de définir le périmètre du système d'information. Il utilise la notion d'acteurs internes et d'acteurs externes. Il est tentant de rapprocher le diagramme des flux du diagramme des cas d'utilisation. C'est partiellement faux, car le diagramme des cas d'utilisation traduit une relation entre l'acteur et le domaine dans une optique de collaboration. Selon la méthode Merise, les diagrammes des flux sont plutôt orientés vers la décomposition des fonctions. En théorie les diagrammes des flux devraient être rapprochés des diagrammes d'activité. Mais en pratique, comme il n'y a pas d'équivalent direct au modèle de contexte il est tentant de le rapprocher du diagramme des cas d'utilisation. Le diagramme des cas d'utilisation peut intervenir avant les diagrammes d'activité pour donner une vision globale des communications entre les acteurs et le domaine.

## 2. Le diagramme des cas d'utilisation

Aussi nommé Use Case, ce diagramme peut permettre de représenter les relations existantes entre les acteurs et le domaine étudié.

Exemple :



Reprenons un exemple déjà vu au chapitre concernant les diagrammes de flux : une agence de location de vélo veut informatiser la gestion des locations. Lorsqu'un client se présente à l'accueil, il précise le type de vélo désiré ainsi que la durée de la location. L'accueil vérifie si, en fonction du stock disponible, la location est possible et donne la réponse au client. Si la location est possible, la facture est éditée et donnée au client. Celui-ci doit payer immédiatement. Le paiement et la facture sont ensuite transmis au service comptable. L'accueil transmet alors la demande au gestionnaire du parc. Ce dernier va préparer le vélo demandé et le mettre à disposition du client.

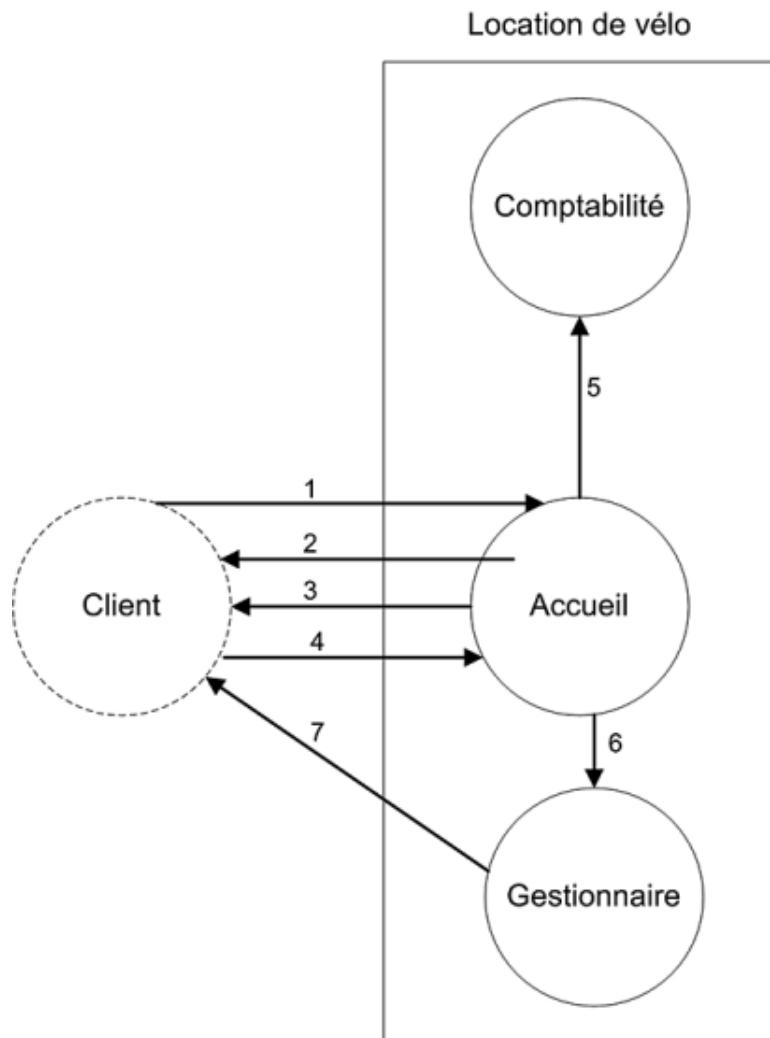
Identifions les flux :

- Le premier flux est la demande de location.
- Le deuxième flux est l'acceptation ou le refus de la location en fonction du stock disponible.

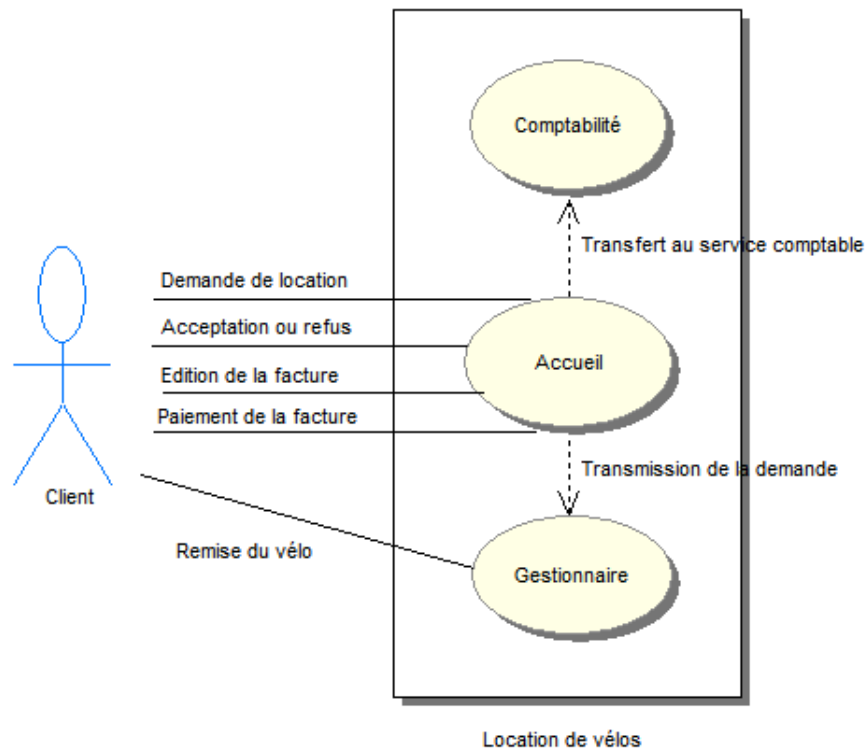
Dans le cas de l'acceptation :

- Le troisième flux représentera l'édition de la facture.
- Le quatrième flux sera le paiement de la facture par le client.
- Le cinquième flux représente le passage de la facture et du paiement au service comptabilité.
- Le sixième flux est la transmission de la demande au gestionnaire du parc.
- Le septième et dernier flux est la remise du vélo au client par le gestionnaire.

Le diagramme des flux Merise était le suivant :



Voici ce que pourrait être le diagramme des cas d'utilisation associé :



Comme vous le constatez, les analogies entre les graphiques sont flagrantes. Mais attention je le répète : il est faux de dire, d'un point de vue de puriste, que ces deux modèles sont le pendant l'un de l'autre. Ici on détourne un diagramme UML pour le faire correspondre à un diagramme Merise.

### 3. Modèle Conceptuel des Données/Diagramme de classes

Le MCD et le diagramme de classes partagent beaucoup de points communs. Les différences majeures apparaissent dans le côté objet d'UML. Cependant, au niveau du processus d'analyse, le diagramme de classes se rapproche plus du modèle logique des données. Le langage UML pour représenter une base de données ne passe pas d'un état correspondant à un MCD à un MLD. Il faut donc à l'analyste une bonne approche ou vision de la base de données pour la représenter sans faille en langage UML. Le découpage MCD, MLD apporte plus de sécurité, à ce niveau-là, qu'UML.

Voici quelques éléments de comparaison entre Merise et UML.

#### a. Les cardinalités

Les cardinalités s'expriment différemment dans le diagramme de classes. Elles sont inversées par rapport à la méthode Merise. Voici un tableau donnant des éléments de comparaison.

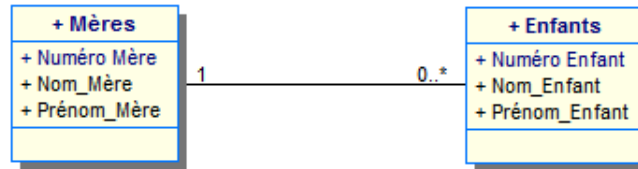
Merise	UML
0,1	0..1
1,1	1
0,n	0..* ou *
1,n	1..*

Voici au travers de quelques exemples simples les différences entre Merise et UML.

Voici un exemple Merise :



Voici comment le traiter en UML :



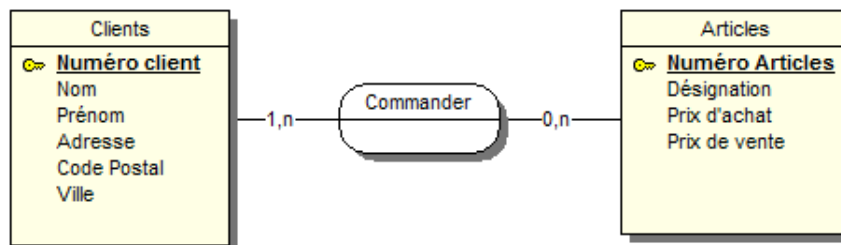
Cette solution modélise le MCD, comme vous le constatez les cardinalités ne se placent pas de manières identiques à Merise. Elles se lisent ainsi « Une mère peut avoir 0 ou plusieurs enfants ».

Une représentation du MCD serait la suivante :

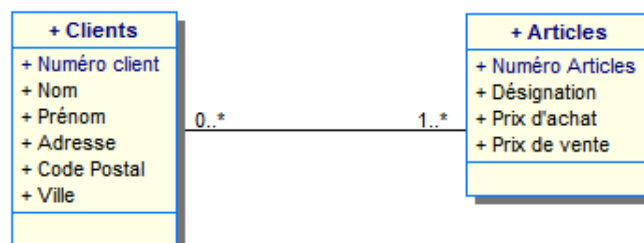


UML n'ayant pas la notion d'identifiants, les clés primaires sont les propriétés les plus hautes et les clés étrangères sont dans cet exemple précédées d'un #.

Voici le cas où les cardinalités sont toutes à n :

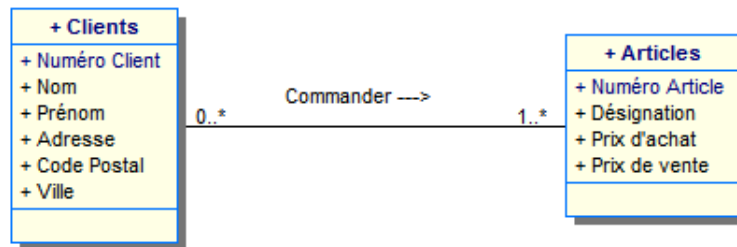


Voici la traduction UML du MCD :



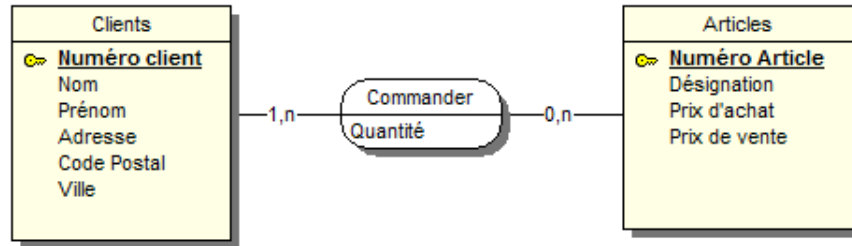
Pour aider la compréhension du modèle, il est conseillé de décrire la relation comme dans l'exemple ci-dessous :



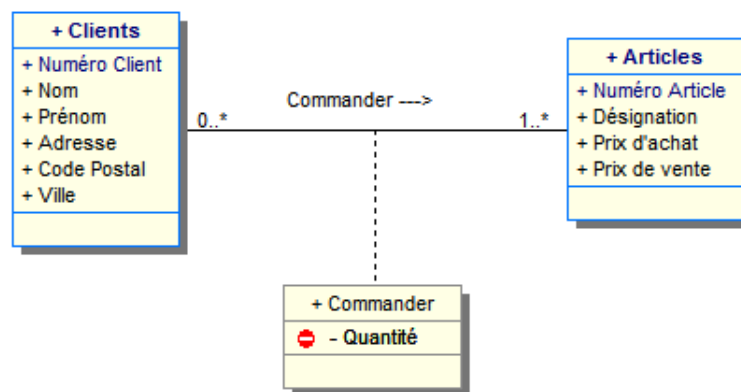


## b. Associations porteuses

Voici maintenant le cas où l'association est porteuse de propriétés :



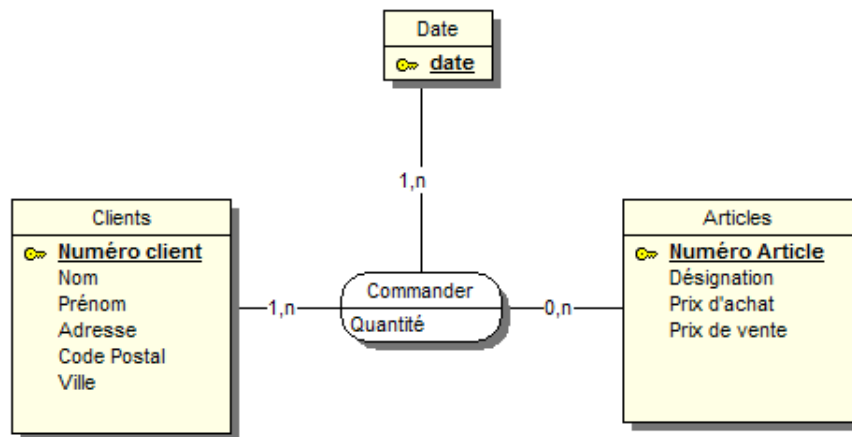
En UML cela donnerait le diagramme des classes suivant :



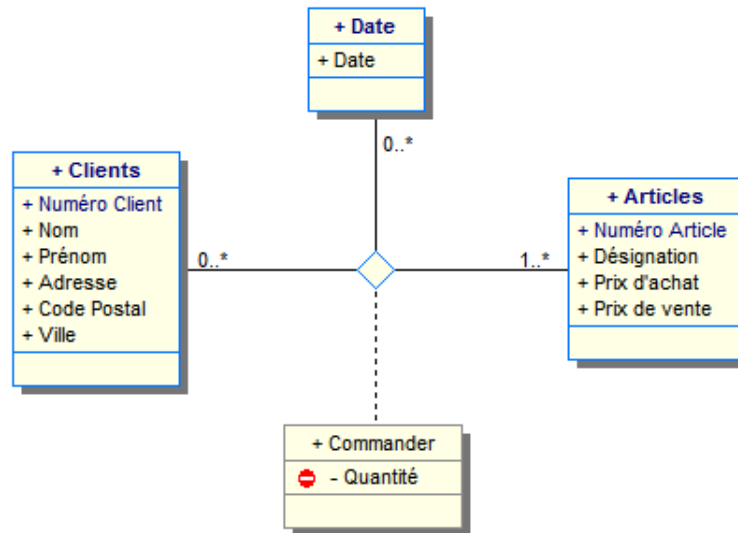
Le symbole sens interdit devant Quantité signifie que l'attribut est privé à la classe et seulement accessible par une méthode assesseur. Cette représentation provient du logiciel Win'Design et peut être différente selon les outils de modélisation (un cadenas avec Rational Rose par exemple).

## c. Associations ternaires

Passons à l'interprétation d'une association ternaire.

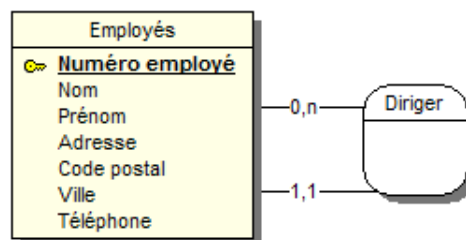


Voici le résultat en UML :

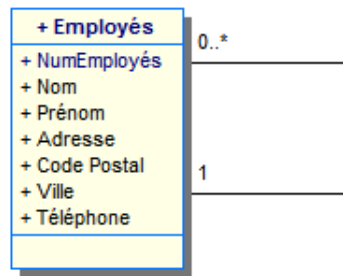


#### d. Représentation de la réflexivité

Voici maintenant le cas de l'association réflexive :

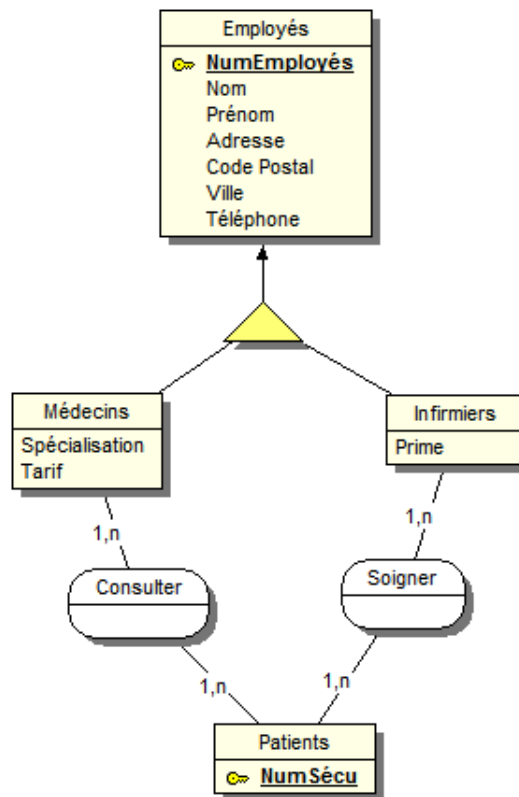


Le diagramme de classe correspondant est le suivant :

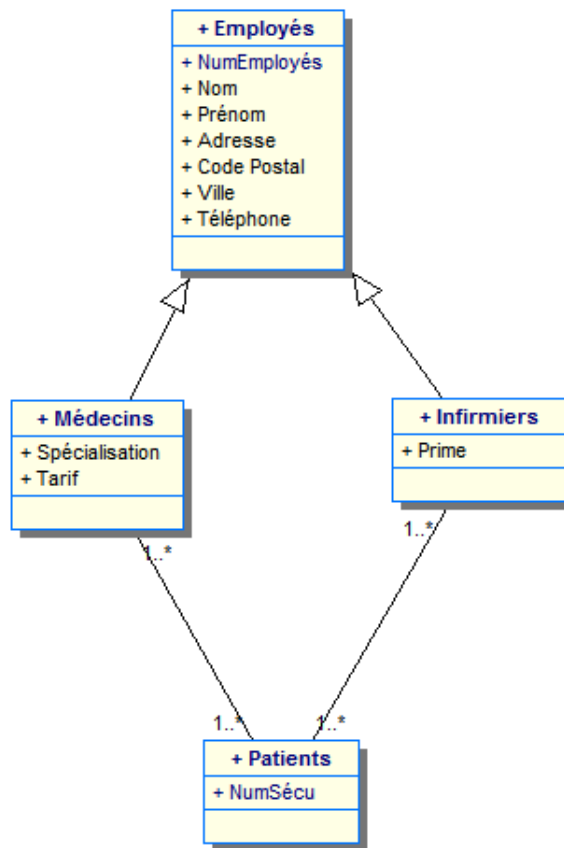


## e. L'héritage

La notion d'héritage ne pose aucun problème particulier. Si nous reprenons ce modèle Merise :



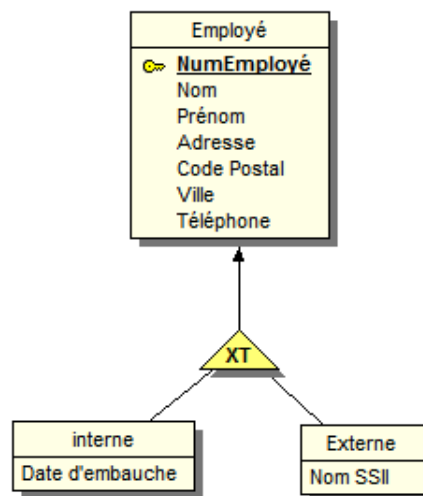
En UML la représentation de l'héritage (ou de la généralisation) est très explicite.



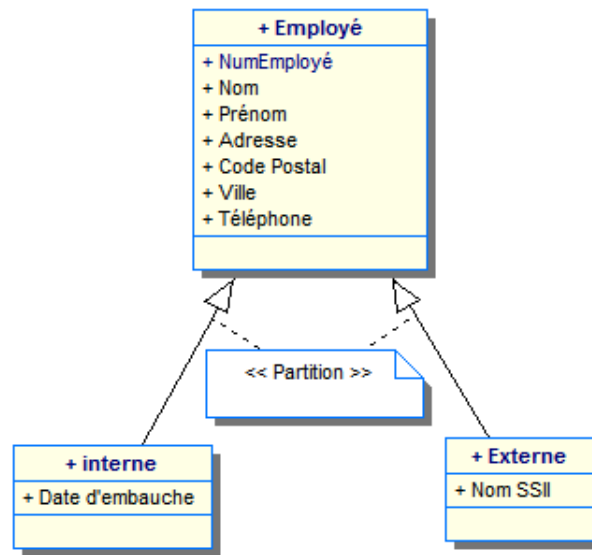
## 4. Les contraintes

### a. La contrainte de partition

Voyons maintenant comment exprimer les contraintes. Le graphique ci-dessous représente l'expression d'une contrainte de partition en Merise.

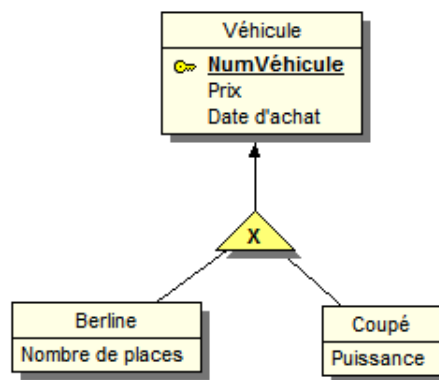


Un employé est soit un employé interne, soit un employé externe, mais pas les deux.  
Voici comment l'exprimer en UML.

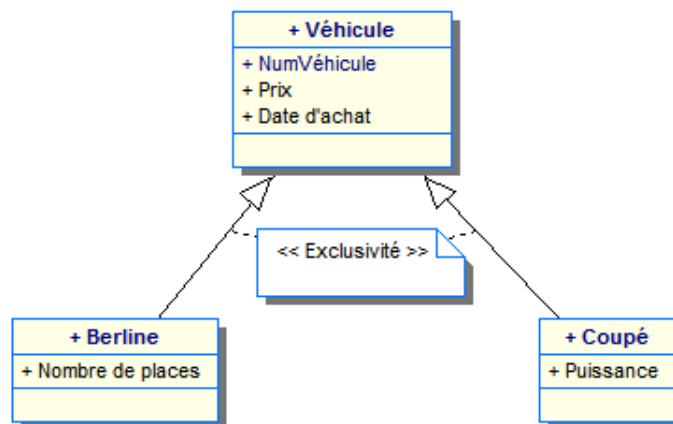


## b. L'exclusion

Un véhicule est soit une berline, soit un coupé mais pas les deux et il peut exister d'autres véhicules.

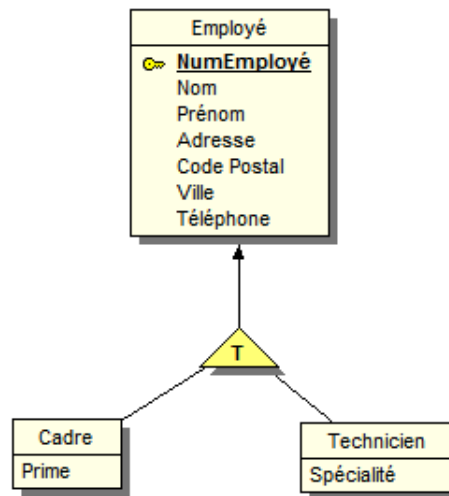


L'exclusion en UML :

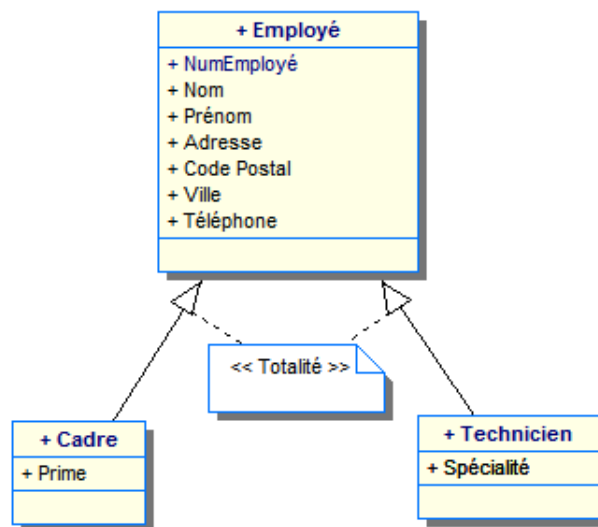


## c. La contrainte de totalité

Un employé peut être à la fois cadre et technicien. Il est au moins l'un ou l'autre.

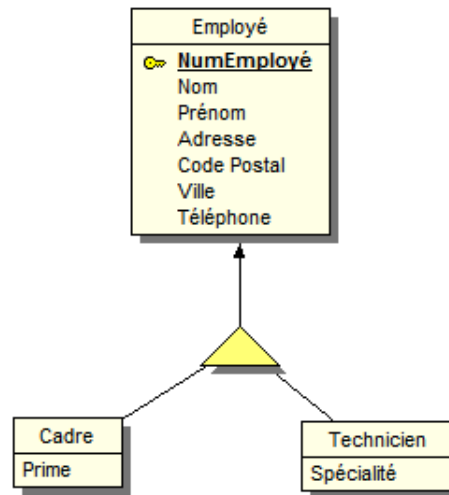


L'expression de la totalité en UML :

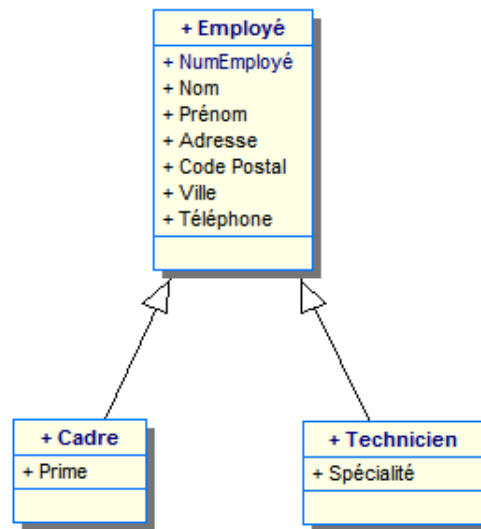


#### d. Aucune contrainte

Et enfin l'expression d'aucune contrainte en Merise :



Et en UML :



## 5. Le Modèle Conceptuel des Traitements

Dans un Modèle Conceptuel des Traitements sont décrits les concepts d'événements, de règles de gestion, de synchronisation et d'opérations qui réutilisent la notion de flux.

En UML, les événements internes et externes apparaissent au niveau des diagrammes de séquences ou d'activité. Les synchronisations sont décrites dans les diagrammes d'activité et d'états-transitions. Les opérations sont décrites dans les diagrammes d'activité.

Comme vous le voyez, il faut plusieurs diagrammes UML pour décrire un Modèle Conceptuel des Traitements.

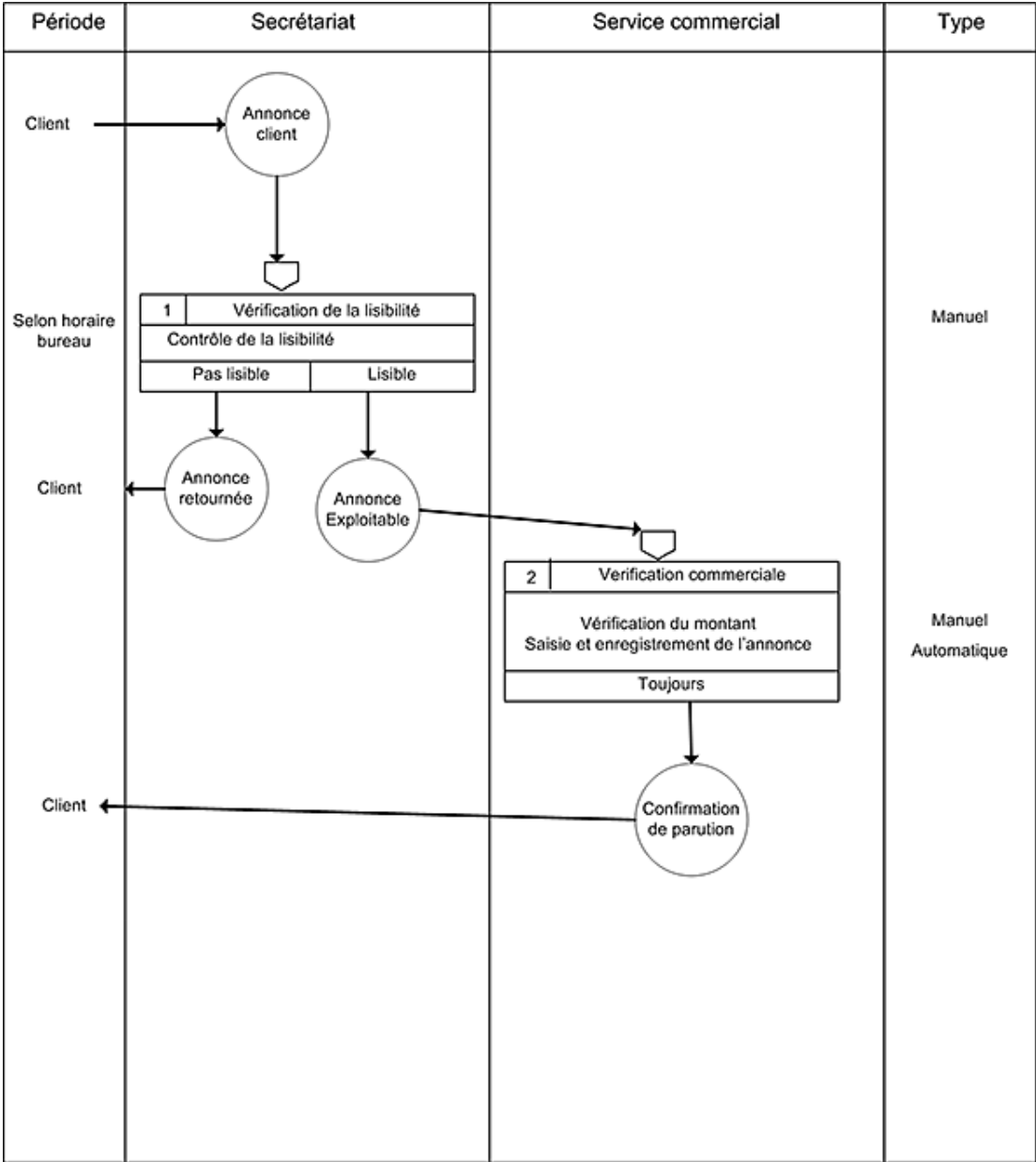
## 6. Le Modèle Organisationnel des Traitements

Dans un MOT, nous trouvons :

- les fonctions (règles de gestion),
- les acteurs,
- les flux.

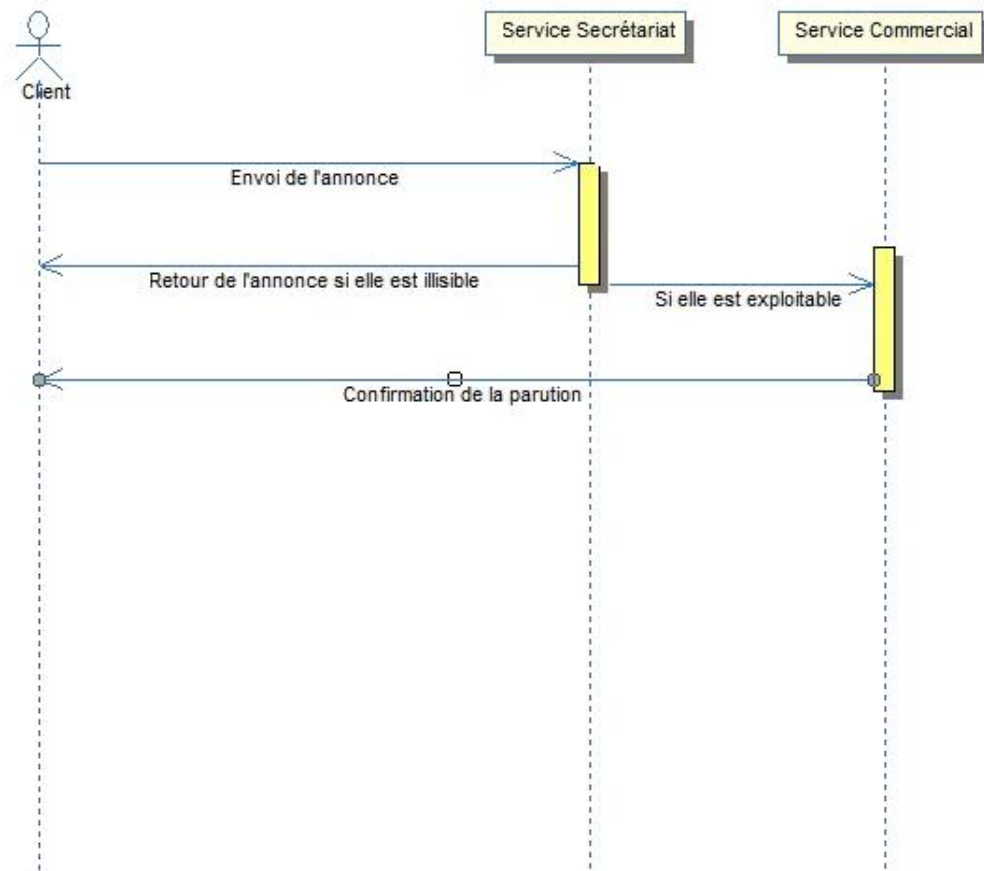
En UML ces aspects dynamiques peuvent se retrouver dans les diagrammes d'activité ou de séquence, mais Merise donnera toujours une vision plus globale du système d'information.

Traduisons en UML le MOT suivant :

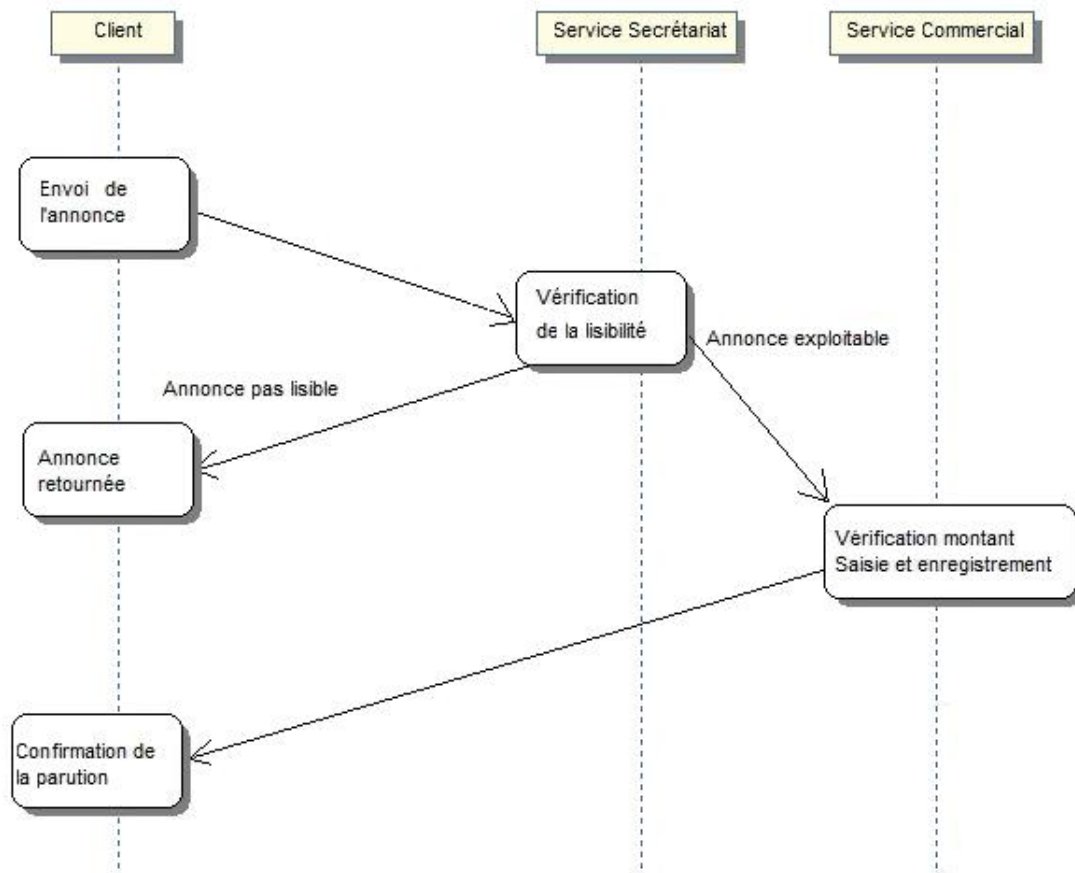


Voici une adaptation du MOT en diagramme de séquence :





Voici maintenant une représentation du diagramme d'activité :





# Présentation du langage SQL

Au chapitre Le modèle physique des données de ce livre, nous avons vu que le modèle physique nous permettait d'établir l'ensemble des relations (ou fichiers) constituant la base de données. Merise nous aide à la conception et à l'élaboration de la base de données, le langage SQL quant à lui nous aide à manipuler les données ou la structure de la base de données.

## 1. Historique

Le langage SQL a été élaboré dans les années 1970 d'après les théories d'un informaticien britannique : Edgar Frank Codd.

Le docteur Codd est considéré comme l'inventeur du modèle relationnel. Il travaillait au laboratoire de recherche d'IBM à San José en Californie. IBM mis du temps à croire en la théorie d'un langage d'interrogation des données structuré et ce sont des entreprises concurrentes telles Oracle qui les premières misèrent sur le langage SQL.

De par sa nature simple et presque naturelle, SQL ne tarda pas à devenir un standard de fait dans la manipulation des données.

Il fut rapidement normalisé, garantissant ainsi son indépendance vis-à-vis des systèmes de gestion de bases de données relationnelle (SGBDR). Ainsi, une requête SQL peut être portée sans modifications (ou alors mineures) de MySQL à Oracle ou d'Oracle à SQL Server.

Voici un tableau récapitulant les différentes versions successives du langage SQL :

Année	Appellation
1986	SQL-86 ou SQL-87
1989	SQL-89 ou SQL-1
1992	SQL-92 ou SQL2
1999	SQL-99 ou SQL3
2003	SQL:2003
2008	SQL:2008

Les versions apportant leurs lots de modifications il est important, avant d'utiliser l'une ou l'autre, de vérifier que le système de gestion de bases de données intègre la même version.

## 2. Structuration

Le langage SQL est composé de trois parties majeures et distinctes :

- Le DML (*Data Manipulation Language*) ou LMD (langage de manipulation des données) : le DML permet de consulter ou de modifier le contenu de la base de données.
- Le DDL (*Data Definition Language*) ou LDD (langage de définition des données) : le DDL permet de modifier la structure de la base de données.
- Le DCL (*Data Control Language*) ou LCD (langage de contrôle des données) : le DCL permet de gérer les privilèges, ou les différents droits des utilisateurs sur la base de données.

Voici les principaux ordres employés :

DML	DDL	DCL
SELECT	CREATE	GRANT

INSERT	ALTER	REVOKE
DELETE	RENAME	
UPDATE	DROP	

Une requête SQL peut être employée seule, dans un éditeur de requête fourni par le logiciel de gestion de bases de données, ou directement intégrée dans le langage de programmation. Par exemple, une requête peut être testée directement avec phpMyAdmin et être ensuite intégrée dans une procédure écrite en langage PHP pour interagir avec le gestionnaire de bases de données MySQL.

# Le langage de manipulation des données

Nous allons aborder dans cette partie les ordres du langage de manipulation de données. Nous allons commencer par l'ordre SELECT.

Pour les exemples, nous allons utiliser les trois fichiers tirés de ces modèles relationnels :

Clients(**NumCli**, Nom, Prénom, Adresse, Cp, Ville, Téléphone)

Achats(**#NumCli**, **#NumArt**, Date, Qté)

Articles(**NumArt**, Désignation, Catégorie, Prix)

Clients	NumCli	Nom	Prénom	Adresse	CP	Ville	Téléphone
	1	Auguy	Joel	1 rue droite	30000	Nîmes	0485957575
	2	Baptiste	Jean-Luc	7 rue courbe	12000	Rodez	0565428775
	3	Baptiste	Amandine	Avenue Foch	12000	Rodez	
	4	Collard	Marie-Claire	Rue d'Espagne	66000	Perpignan	
	5	Durand	Raymond	Rue des oliviers	30000	Nîmes	0475145425

Achats	NumCli	NumArt	Date	Qté
	1	1	30/01/2009	1
	1	5	30/01/2009	4
	4	3	29/01/2009	1
	4	2	30/01/2009	2
	5	2	01/02/2009	2

Articles	NumArt	Désignation	Catégorie	Prix
	1	Charlie Winston	Cd	12
	2	Caméra Café	Dvd	19
	3	WebCam	Informatique	24
	4	Graveur	Informatique	38
	5	Clé Usb 16G	Informatique	18

## 1. Sélection des données

La commande SELECT permet de réaliser une recherche d'informations selon certains critères.

Syntaxe :

```
SELECT [ALL / DISTINCT] nom_attribut1 [, nom_attribut2, .....]  
FROM nom_table1 [, nom_table2, ....]  
WHERE <condition de recherche> ;
```

L'option **ALL** est l'option par défaut qui permet de sélectionner l'ensemble des lignes satisfaisant à la condition de recherche.

L'option **DISTINCT** permet de ne conserver que des lignes distinctes, en éliminant les doublons.


La **liste des attributs** indique la liste des colonnes choisies, séparées par des virgules. Pour sélectionner l'ensemble des colonnes d'une table, il est possible d'utiliser l'option **\***.

La **liste des tables** indique l'ensemble des tables (séparées par des virgules) sur lesquelles portent les opérations.

La **condition de recherche** permet d'exprimer des critères de recherche complexes à l'aide d'opérateurs logiques et/ou de comparateurs arithmétiques.

**a. La projection**

La projection est une opération sur une relation **RELATION1** consistant à composer une relation **RELATION2** en enlevant à la relation initiale tous les attributs non mentionnés en opérande (aussi bien au niveau du schéma que des tuples) et en éliminant les tuples en double qui ne sont conservés qu'une seule fois.

 Un tuple représente une ligne dans un fichier.

Voyons ensemble quelques exemples :

*Afficher le contenu de la table client.*

```
SELECT *
FROM Clients;
```

Résultat renvoyé par la requête :

NumCli	Nom	Prénom	Adresse	CP	Ville	Téléphone
1	Auguy	Joel	1 rue droite	30000	Nîmes	0485957575
2	Baptiste	Jean-Luc	7 rue courbe	12000	Rodez	0565428775
3	Baptiste	Amandine	Avenue Foch	12000	Rodez	
4	Collard	Marie-Claire	Rue d'Espagne	66000	Perpignan	
5	Durand	Raymond	Rue des oliviers	30000	Nîmes	0475145425

*Afficher les noms et prénoms des clients.*

```
SELECT Nom, Prénom
FROM Clients;
```

Résultat retourné par la requête :

Nom	Prénom
Auguy	Joel
Baptiste	Jean-Luc
Baptiste	Amandine
Collard	Marie-Claire
Durand	Raymond

*Utilisation du mot clé DISTINCT.*

Activons cette requête :

```
SELECT Nom  
FROM Clients;
```

Comme le mot clé ALL est actif par défaut, le résultat retourné par la requête sera le suivant :

Nom
Auguy
Baptiste
Baptiste
Collard
Durand

Nous nous rendons compte que le nom Baptiste apparaît deux fois, ce qui dans notre cas n'est d'aucun intérêt.

Voici la requête corrigeant ce problème :

```
SELECT DISTINCT Nom  
FROM Clients;
```

Et le résultat retourné :

Nom
Auguy
Baptiste
Collard
Durand

Il est possible de modifier l'affichage d'un nom de colonne en utilisant le mot clé AS. Par exemple, nous souhaitons modifier l'affichage du Nom par Nom des clients.

Voici la requête :

```
SELECT DISTINCT Nom AS Nom des clients  
FROM Clients;
```

et le résultat retourné :

Nom des clients
Auguy
Baptiste
Collard
Durand

## b. La restriction

La restriction est une opération sur une relation RELATION1 produisant une relation RELATION2 de même schéma

mais comportant les seuls tuples qui vérifient la condition précisée en opérande.

Essayons les requêtes suivantes :

*Sélectionner les clients habitants Rodez.*

```
SELECT *  
FROM Clients  
WHERE Ville='Rodez';
```

*Lister les articles dont le prix est supérieur à 15 euros.*

```
SELECT *  
FROM Articles  
WHERE Prix>15;
```



Les apostrophes ne servent que pour différencier les valeurs alphanumériques des valeurs numériques.

Dans la clause WHERE les opérateurs suivants peuvent être utilisés :

>	Supérieur
>=	Supérieur ou égal
<	Inférieur
<=	Inférieur ou égal
=	égal
<>	Différent
AND	Et
OR	Ou
NOT	Pas
IS NULL	Valeur indéterminée
ALL	Tous
ANY	Au moins un
EXISTS	Existence

*Lister les articles dont le prix est compris entre 14 et 30 euros.*

```
SELECT *  
FROM Articles  
WHERE prix>14 AND prix <30;
```

Cette requête est correcte, mais manque d'élégance. Il existe d'autres mots clés permettant d'exprimer certaines contraintes.

IN
BETWEEN
LIKE



Reformulons la requête précédente pour la rendre plus élégante :

```
SELECT *  
FROM Articles  
WHERE prix BETWEEN 14 AND 30;
```

*Afficher les clients dont les noms sont Baptiste et la ville est Rodez*

```
SELECT *  
FROM Clients  
WHERE nom='Baptiste'  
AND ville='Rodez';
```

*Afficher les clients dont les noms sont Baptiste ou la ville est Rodez*

```
SELECT *  
FROM Clients  
WHERE nom='Baptiste'  
OR ville='Rodez';
```

*Afficher les clients dont les numéros de téléphone ont été saisis.*

```
SELECT *  
FROM Clients  
WHERE Téléphone IS NOT NULL;
```

*Afficher les clients dont les numéros de téléphone n'ont pas été saisis.*

```
SELECT *  
FROM Clients  
WHERE Téléphone IS NULL;
```

*Afficher les clients qui habitent les villes de Rodez, Aurillac ou Tarbes.*

```
SELECT *  
FROM Clients  
WHERE Ville IN ('Rodez', 'Aurillac', 'Tarbes');
```

*Afficher les clients dont les noms commencent par B.*

```
SELECT *  
FROM Clients  
WHERE Nom LIKE 'B%'
```

*Afficher les clients dont les noms ne commencent pas par B.*

```
SELECT *  
FROM Clients  
WHERE Nom NOT LIKE 'B%'
```

*Afficher les clients dont les noms finissent par B.*

```
SELECT *  
FROM Clients  
WHERE Nom LIKE '%B'
```

*Afficher les clients dont les noms contiennent un B.*

```
SELECT *
FROM Clients
WHERE Nom LIKE '%B%'
```

### c. Les tris

Avec SQL il est possible d'effectuer des tris selon différents critères grâce à la clause ORDER BY et aux mots clés ASC, DESC. Par défaut le tri est par ordre croissant.

*Afficher les noms et prénoms des clients triés par noms croissants.*

```
SELECT *
FROM Clients
ORDER BY Nom ASC;
```

*Afficher les noms et prénoms des clients triés par noms décroissants.*

```
SELECT *
FROM Clients
ORDER BY Nom DESC;
```

### d. Les jointures

La jointure est l'opération consistant à rapprocher selon une condition les tuples de deux relations RELATION1 et RELATION2 afin de former une troisième relation RELATION3 qui contient l'ensemble de tous les tuples obtenus en concaténant un tuple de RELATION1 et un tuple de RELATION2 vérifiant la condition de rapprochement.

*Afficher les noms et prénoms des clients ayant acheté un article le 30 février 2009.*

```
SELECT Nom, Prénom
FROM Clients, Achats
WHERE Clients.NumCli=Achats.Numcli
AND Achats.Date='30/02/2009';
```

*Afficher le nom, le prénom des clients ainsi que la quantité et désignation des produits achetés.*

```
SELECT Nom, Prénom, Qté, Désignation
FROM Clients, Achats, Articles
WHERE Clients.NumCli=Achats.Numcli
AND Achats.NumArt=Articles.NumArt;
```



Les noms de tables peuvent devenir fastidieux à saisir, on peut simplifier l'écriture en utilisant le mot clé AS.

*Afficher le nom, le prénom des clients ainsi que la quantité et désignation des produits achetés.*

```
SELECT Cl.Nom, Cl.Prénom, Ac.Qté, Ar.Désignation
FROM Clients AS Cl, Achats AS Ac, Articles AS Ar
WHERE Cl.NumCli=Ac.Numcli
AND Ac.NumArt=Ar.NumArt;
```

### e. Les fonctions statistiques

SQL offre cinq fonctions mathématiques standard :

Fonctions	Description

<b>AVG</b> (attribut)	calcule la moyenne des valeurs dans l'attribut.
<b>SUM</b> (attribut)	calcule la somme des valeurs dans l'attribut.
<b>MIN</b> (attribut)	détermine la plus petite valeur dans l'attribut.
<b>MAX</b> (attribut)	détermine la plus grande valeur dans l'attribut.
<b>COUNT</b> (attribut)	compte le nombre d'occurrences dans l'attribut.

*Calculer le prix moyen des articles.*

```
SELECT AVG(Prix) AS Prix Moyen
FROM Articles;
```

*Calculer le prix moyen des articles, afficher le prix minimum et le prix maximum.*

```
SELECT AVG(Prix) AS Prix Moyen, MIN(Prix), MAX(Prix)
FROM Articles;
```

*Afficher la somme de toutes les quantités achetées.*

```
SELECT SUM(Qté) AS Quantité
FROM Achats;
```

*Compter le nombre de catégories.*

```
SELECT COUNT(Catégorie) AS Nombre de catégorie
FROM Articles.
```

Le problème avec cette requête c'est qu'elle va retourner le nombre total de lignes dans la colonne catégorie (5 dans la table exemple), mais pas le nombre de catégories uniques (3 dans la table exemple). Voici comment corriger ce problème.

*Compter le nombre de catégories sans doublons.*

```
SELECT COUNT(DISTINCT Catégorie) AS Nombre de catégorie
FROM Articles.
```

*Faire la somme des quantités totales des achats réalisés pour l'article numéro 3.*

```
SELECT SUM(Qté) AS Quantité totale
FROM Achats
WHERE NumArt=3;
```

En dehors de ces fonctions, il est possible aussi d'effectuer des calculs dans les requêtes. Imaginons que nous désirions afficher les prix augmentés de 10%. Voici une façon de l'écrire :

```
SELECT Désignation, Prix*1,10 AS Prix augmenté
FROM Articles;
```



La valeur n'est pas modifiée dans le fichier, la valeur est calculée juste pour l'affichage.

## f. Les opérations portant sur des ensembles

L'algèbre relationnelle permet l'utilisation d'opérateurs ensemblistes. Il en existe trois :

- UNION,
- INTERSECT,
- EXCEPT.



Les mots clés INTERSECT et EXCEPT n'étant pas normalisés dans SQL, ils risquent de ne pas être fonctionnels dans le SGBD. Le mot clé EXCEPT peut être remplacé par NOT EXISTS et INTERSECT par EXISTS.

Le mot clé UNION retourne l'ensemble des tuples appartenant aux relations de la requête.

Par exemple :

*Afficher les numéros des articles dont le prix est supérieur à 20 euros ainsi que les numéros des articles achetés par le client numéro 1.*

```
SELECT NumArt
FROM Articles
WHERE Prix > 20
UNION
SELECT NumArt
FROM Achats
WHERE NumCli=1;
```

## g. Les regroupements

Les regroupements permettent de créer des sous-ensembles d'occurrences. Une seule ligne regroupe ainsi les valeurs identiques en fonction de l'attribut spécifié.

Imaginons que dans le fichier achat nous souhaitons avoir le total des prix par catégorie. Par exemple ceci :

Catégorie	Prix
Cd	12
Dvd	19
Informatique	80

La requête permettant ce regroupement s'exprime en utilisant la clause GROUP BY.

*Effectuer le regroupement des articles par catégorie et cumuler les prix.*

```
SELECT Catégorie, Sum(Prix) AS Prix cumulés
FROM Articles
GROUP BY Catégorie;
```

*Calculer le prix de vente moyen par catégorie.*

```
SELECT Catégorie, AVG(Prix) AS Prix moyen
FROM Articles
GROUP BY Catégorie;
```

*Pour chaque client, afficher le nombre d'achats et le montant cumulé des achats.*

```
SELECT Clients.Nom, Clients.Prénom, Count(*) AS Nbre,
Sum(Articles.prix*Achats.Qté) As Total
FROM Achats, Clients, Articles
WHERE Achats.NumCli=Clients.NumCli
AND Achats.NumArt=Articles.NumArt
```

```
GROUP BY Clients.Nom, Clients.Prénom;
```

*Calculer le prix de vente moyen des articles par catégorie, dont le prix de vente est inférieur ou égal à 15.*

```
SELECT Catégorie, AVG(Prix)
FROM Articles
WHERE Prix <= 15
GROUP BY Catégorie;
```

Lorsque l'on souhaite réaliser des restrictions sur la clause de regroupement, il faut utiliser la clause HAVING.

*Afficher les numéros de clients ayant plus de 1 achat.*

```
SELECT NumCli, count(*) AS Nbr de commande
FROM Achats
GROUP BY NumCli
HAVING COUNT(*)>1;
```

*Afficher les clients dont le prix moyen d'achat des articles est supérieur à 10 euros dans l'ordre croissant des noms.*

```
SELECT Clients.nom, AVG(Articles.prix)
FROM articles, Achats, Clients
WHERE Clients.NumCli = Achats.NumCli
AND Achats.NumArt=Articles.NumArt
GROUP BY Clients.nom
HAVING AVG(Articles.Prix)>10
ORDER BY Clients.nom;
```

## h. Les sous-requêtes

Les sous-requêtes, appelées aussi requêtes imbriquées, permettent de réaliser de façon élégante des traitements qui pourraient s'avérer fastidieux voir difficilement réalisable à l'aide de plusieurs requêtes simples. Pour simplifier, le rôle d'une sous-requête est d'envoyer le résultat de son traitement dans la requête principale. Voyons grâce à un exemple.

*Lister les achats du client « Auguy »*

```
SELECT NumArt, Date, Qté
FROM Achats
WHERE NumCli = (SELECT NumCli
                FROM Clients
                Where Nom = 'Auguy');
```

La sous-requête renvoie le numéro du client recherché à la requête principale.

Nous aurions pu, pour cet exemple traiter la requête de cette façon à l'aide d'une jointure :

```
SELECT Achats.NumArt, Achats.Date, Achats.Qté
FROM Achats, Clients
WHERE Clients.NumCli=Achats.NumCli
And Clients.Nom='Auguy' ;
```

Mais si nous compliquons les exemples, nous allons nous rendre compte que les sous-requêtes deviennent de bonnes alliées.

Par exemple :

*Nous désirons connaître les articles de prix supérieur au prix moyen de tous les articles.*

```
SELECT *
FROM Articles
WHERE Prix > (SELECT AVG(PRIX)
              FROM Articles);
```

La sous-requête calcule le prix moyen et le retourne à la requête principale qui peut ainsi effectuer son travail de recherche des prix supérieurs au prix moyen.

Maintenant, observons la puissance de sous-requêtes.

*Rechercher les clients habitant la même ville que le client numéro 2 et ayant acheté des articles de prix supérieur à 15 euros.*

```
SELECT Nom, Prénom
FROM Clients, Achats
WHERE Clients.Ville = (SELECT Ville
                      FROM Clients
                      WHERE NumCli=2)
AND Clients.NumCli=Achats.NumCli
AND Achats.NumArt IN (SELECT NumArt
                     FROM Articles
                     WHERE Prix>15);
```

Observons la deuxième sous-requête. Le mot clé IN permet de faire rechercher un numéro d'article dans un ensemble de numéros renvoyés par la sous-requête.

*Afficher les numéros de clients ayant acheté un produit en quantité supérieure à chacun des produits achetés par le client numéro 1.*

```
SELECT DISTINCT NumCli
FROM Achats
WHERE Qté > ALL
      (SELECT Qté
       FROM Achats
       WHERE NumCli=1);
```

Le mot clé ALL signifie que Qte va être testé avec l'ensemble des quantités renvoyées par la sous-requête.

*Donner la liste des articles dont le prix de vente est supérieur au prix de vente de tous les articles dont la catégorie est CD.*

```
SELECT *
FROM articles
WHERE Prix > ALL      (SELECT Prix
                      FROM articles
                      WHERE Catégorie= 'Cd');
```

Une autre façon d'écrire cette requête aurait pu être la suivante :

```
SELECT *
FROM articles
WHERE Prix > (SELECT MAX(Prix)
             FROM articles
             WHERE Catégorie= 'Cd');
```

### **Utilisation du mot clé EXISTS**

Le mot clé EXISTS permet de tester que la sous-requête renvoie un résultat :

*Afficher la liste de tous les clients si l'un d'eux habite Aurillac.*

```
SELECT NumCli, Nom, Prénom
FROM Clients
WHERE EXISTS (Select *
             FROM Clients
             WHERE Ville='Aurillac');
```

Ou l'inverse :

*Afficher la liste de tous les clients si aucun d'eux n'habite Aurillac.*

```
SELECT NumCli, Nom, Prénom
FROM Clients
WHERE NOT EXISTS (Select *
                  FROM Clients
                  WHERE Ville='Aurillac');
```

## 2. L'insertion des données

### a. Insertion simple

Le mot clé INSERT permet l'ajout d'enregistrements dans les fichiers.

Syntaxe :

```
INSERT INTO "nom de table" ("colonne 1", "colonne 2", ...)
VALUES ("valeur 1", "valeur 2", ...)
```

*Ajouter un article.*

```
INSERT INTO Articles(NumArt,Désignation,Catégorie,Prix)
VALUES(6,'Pocket Pc HP','Pda',175);
```

Une autre façon d'écrire la requête est la suivante :

```
INSERT INTO Articles
VALUES(6,'Pocket Pc HP','Pda',175);
```

Cette dernière requête est fonctionnelle, car l'ensemble des champs est renseigné. Le nombre de valeurs de la requête doit coïncider avec le nombre de champs de destination ; ainsi la requête suivante va retourner une erreur :

```
INSERT INTO Articles
VALUES(7,'Pocket Pc HP');
```

Lorsque certaines valeurs ne sont pas connues à l'exécution de la requête, il est possible de la compléter avec le mot clé NULL :

```
INSERT INTO Articles
VALUES(8,'Pocket Pc HP',NULL,NULL);
```

ou encore il est possible de tout préfixer :

```
INSERT INTO Articles( NumArt,Désignation)
VALUES(9,'Pocket Pc');
```

### b. Insertion en masse

Il est possible d'insérer dans une table un ensemble d'enregistrements. Par exemple, imaginons que nous souhaitons envoyer un mailing ciblé sur les clients habitant la ville de Rodez. Un fichier nommé mailing de même structure que le fichier Clients est créé. Voici la requête qui permet de copier les clients habitant Rodez dans le fichier mailing.

```
INSERT INTO mailing
SELECT *
FROM Clients
WHERE Ville = 'Rodez';
```

## 3. La modification des données

SQL permet la modification des données grâce au mot clé UPDATE.

Syntaxe :

```
UPDATE "nom de table"  
SET "colonne 1" = [nouvelle valeur]  
WHERE {condition}
```

*Augmenter tous les prix de 15 %.*

```
UPDATE Articles  
SET Prix = Prix*1.15
```

*Augmenter de 5 % les articles de la catégorie Informatique.*

```
UPDATE Articles  
SET Prix = Prix*1.15  
WHERE Catégorie='Informatique'
```

## 4. La suppression des données

La suppression des données se réalise avec le mot clé DELETE.

Syntaxe :

```
DELETE FROM "nom de table"  
WHERE {condition}
```

*Supprimer l'ensemble des lignes de la table clients.*

```
DELETE *  
FROM Clients;
```

*Supprimer les clients habitant Rodez.*

```
DELETE  
FROM Clients  
WHERE Ville='Rodez';
```



# Le langage de définition des données

SQL fournit des instructions permettant de créer, supprimer, modifier, renommer des fichiers. Ces instructions sont les suivantes :

- CREATE,
- DROP,
- ALTER,
- RENAME.

## 1. La création de tables

La création d'une table se réalise avec l'ordre CREATE.

Syntaxe :

```
CREATE TABLE "nom de table"  
( "colonne 1" "type de données pour la colonne 1",  
  "colonne 2" "type de données pour la colonne 2",  
  ... )
```

*Création de la table Articles.*

```
CREATE TABLE Articles  
  (NumArt INTEGER NOT NULL,  
   Désignation CHAR(60) NOT NULL,  
   Catégorie CHAR(30),  
   Prix INTEGER);
```

### a. Définition de la clé primaire

Dans la table Articles, la clé primaire est NumArt. Voici comment une clé primaire est définie avec SQL :

```
CREATE TABLE Articles  
  (NumArt INTEGER NOT NULL,  
   Désignation CHAR(60) NOT NULL,  
   Catégorie CHAR(30),  
   Prix INTEGER  
   Constraint C1 PRIMARY KEY (NumArt));
```

### b. Définition des clés étrangères

La table Achats contient deux clés étrangères :

- NumCli,
- NumArt.

Voici la requête de création de la table Achat :

```
CREATE TABLE Achats  
  (NumArt INTEGER NOT NULL,  
   NumArt INTEGER NOT NULL,  
   Date CHAR(10),
```

```
Qté INTEGER,  
CONSTRAINT FK1 FOREIGN KEY (NumArt) REFERENCES Articles (NumArt)  
CONSTRAINT FK2 FOREIGN KEY (NumCli) REFERENCES Clients (NumCli));
```

## 2. La suppression physique de tables

La suppression physique de tables se réalise avec l'ordre SQL DROP.

Par exemple :

*Suppression de la table mailing.*

```
DROP TABLE mailing;
```

## 3. Modification d'une structure de table

Il peut être nécessaire de modifier la structure d'une table par exemple pour ajouter un champ, redimensionner un champ ou supprimer un champ.

### a. Ajouter un champ

*Ajouter un champ Mail à la table Clients.*

```
ALTER TABLE Clients  
ADD Mail CHAR(40);
```

### b. Redimensionner un champ

*Agrandir le champ Mail de la table Clients.*

```
ALTER TABLE Clients  
MODIFY Mail CHAR(60);
```

### c. Supprimer un champ

*Supprimer le champ Mail de la table Clients.*

```
ALTER TABLE Clients  
DROP COLUMN Mail;
```

### d. Supprimer une clé sur une table existante

Il est possible de supprimer a posteriori des clés sur une table existante.

*Supprimer la clé sur le champ NumArt de la table Articles.*

```
ALTER TABLE Articles  
DROP CONSTRAINT C1;
```

*Ajouter une clé primaire sur le champ NumArt de la table Articles.*

```
ALTER TABLE Articles  
ADD CONSTRAINT C1 PRIMARY KEY (NumArt);
```

---

## 4. Renommer une table

Pour renommer une table, rien de plus simple.

*Renommer la table Articles en Produits.*

```
RENAME TABLE Articles TO Produits;
```

# Le langage de contrôle des données

Le langage de contrôle des données comprend deux ordres :

- GRANT,
- REVOKE.

GRANT permet de donner des droits à un utilisateur sur une base de données, REVOKE supprime des droits acquis.

## 1. L'ordre GRANT

Cet ordre utilise différentes options pour définir au mieux les droits.

Option	Explication
ALTER	Donne le droit de modifier la structure d'une table.
DELETE	Donne le droit de supprimer des enregistrements.
INSERT	Donne le droit d'insérer des enregistrements dans une table.
SELECT	Donne le droit d'exécuter des requêtes de sélection.
UPDATE	Donne le droit de modifier les données d'une table.
ALL	Donne tous les droits.

*Donner le droit à l'utilisateur Jean-Luc de modifier la structure de la table Clients.*

```
GRANT ALTER
  ON Clients
  TO Jean-Luc;
```

*Donner le droit à l'utilisateur Jean-Luc d'afficher le contenu de la table Clients.*

```
GRANT SELECT
  ON Clients
  TO Jean-Luc;
```

*Donner tous les droits à l'utilisateur Jean-Luc sur la table Client.*

```
GRANT ALL
  ON Clients
  TO Jean-Luc;
```

*Permettre à Jean-Luc de lire, modifier, insérer dans la table Clients.*

```
GRANT SELECT, UPDATE, INSERT
  ON Clients
  TO Jean-Luc;
```

*Permettre à Jean-Luc de modifier seulement les prix et les catégories de la table Articles.*

```
GRANT UPDATE(Catégorie, Prix)
  ON Articles
```

```
TO Jean-Luc;
```

*Permettre à tout le monde d'afficher le contenu de la table Clients.*

```
GRANT SELECT
  ON Clients
  TO PUBLIC;
```

*Donner le droit à Jean-Luc de faire des sélections dans la table Article et de pouvoir redistribuer ce droit à d'autres utilisateurs.*

```
GRANT SELECT
  ON Articles
  TO Jean-Luc
  WITH GRANT OPTION;
```

## 2. L'ordre REVOKE

L'ordre REVOKE permet de reprendre tous les droits accordés aux utilisateurs.

*Enlever le droit donné à Jean-Luc d'afficher la table Clients.*

```
REVOKE SELECT
  ON clients
  FROM Jean-Luc ;
```

*Enlever tous les droits donnés sur la table Clients à tous les utilisateurs.*

```
REVOKE ALL
  ON Clients
  FROM PUBLIC;
```

# Mise en pratique avec MySQL

Tous les exemples donnés plus haut peuvent être testés avec un vrai serveur de base de données.

Le serveur de base de données MySQL est un serveur réputé pour plusieurs raisons :

- Il existe sur de nombreuses plates-formes (Linux, Unix, Windows).
- Il supporte une grande montée en charge. Il est capable de gérer des entrepôts de données de plusieurs téraoctets.
- MySQL peut répondre aux demandes de performances les plus exigeantes. Il peut traiter un volume de requêtes s'exprimant en milliards de requêtes par jours.
- MySQL offre des fonctions de sécurité qui garantissent une très forte protection des données.
- MySQL est Open Source et gratuit en licence développement, payant en exploitation. Le coût de possession est très inférieur à ce que propose la concurrence.

Le site officiel français est : <http://www.mysql.fr>

Il existe des packs logiciels installant une suite logicielle permettant d'installer un serveur MySQL sans effort. Voici certains de ces packs :


- EasyPHP : <http://www.easyphp.org/>
- XAMPP : <http://www.apachefriends.org/en/xampp.html>
- WampServer : <http://www.wampserver.com/>

Ces différents packs intègrent le serveur Web Apache, le langage PHP et MySQL. Ils offrent aussi des outils de configuration et de gestion de MySQL comme phpMyAdmin.

## 1. Installation du serveur de base de données

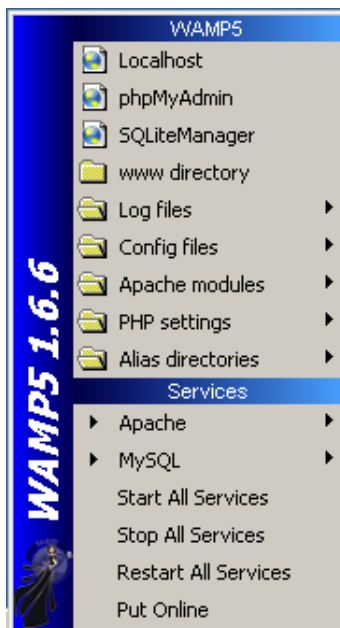
Le pack utilisé dans cet ouvrage sera WampServer.

Il n'y a aucune difficulté pour installer WampServer. Il suffit de télécharger le pack sur le site et de l'installer sur l'ordinateur qui fera office de serveur. En cas de difficultés reportez-vous à l'aide en ligne.

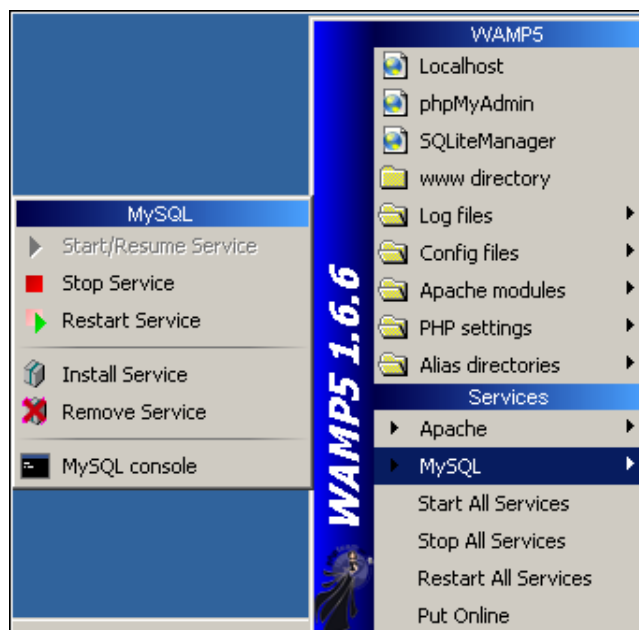
Une fois le pack WampServer installé, une icône spécifique doit apparaître dans la barre d'état du système (ou tray bar). Voici à quoi ressemble cette icône : 

En passant la souris dessus, une indication sur le nombre de services lancés apparaît. Il faut avoir impérativement le service Apache et le service MySQL lancés.

Si nous cliquons dessus avec le bouton **gauche** de la souris, ce menu apparaît :



Pour vérifier l'état des services, il suffit de cliquer sur MySQL par exemple.



L'état de MySQL est lancé et actif, car les seuls choix actifs sont la possibilité d'arrêter le service, de le faire redémarrer, d'installer le service, de le supprimer ou de lancer la console.

Il en est de même pour Apache :

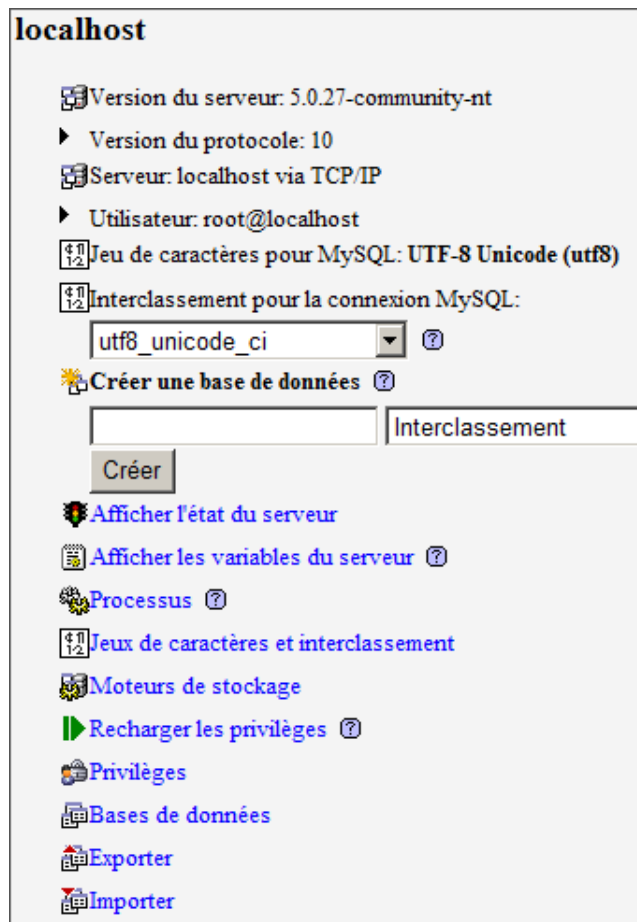


## 2. Création d'une base de données

Le pack étant installé, il faut créer à l'intérieur de MySQL un utilisateur et une base de données. Pour configurer le serveur de bases de données, il faut passer par un utilitaire qui se nomme phpMyAdmin. Il se situe dans la liste des outils de WampServer.

phpMyAdmin est un programme écrit en PHP qui s'exécute à l'intérieur d'un navigateur Web.

Voici une vue partielle de ce qui apparaît dans le navigateur Web :






Il est nécessaire de créer une base de données qui contiendra l'ensemble des fichiers de l'application. Cette base de données se nommera Exercices.

Pour la créer il suffit de renseigner le champ **Créer une base de données** comme ci-dessous :

Le fait de cliquer sur le bouton **Créer** ordonnera à MySQL de créer la base de données. L'ensemble des bases de données hébergées par le serveur MySQL sont visibles en cliquant sur la liste déroulante **Base de données** :

Le chiffre entre parenthèses indique le nombre de fichiers contenus dans la base. Il est visible qu'aucun fichier ne figure dans la base Exercices. Toutes les autres bases (sauf celle nommée test) sont des bases de travail nécessaire à MySQL. Il est impératif de ne pas modifier quoi que ce soit les concernant.

Pour créer un utilisateur, il suffit de cliquer sur le lien **Privilèges**  **Privilèges** .

Voici une vue partielle de la gestion des utilisateurs :

Au départ seul l'utilisateur nommé **root** existe. Ce compte est considéré comme le super utilisateur et il détient tous les droits sur tous les objets existant dans MySQL. Pour des raisons de sécurité, il est impératif de créer un utilisateur possédant seulement des droits sur la base de données nous intéressant : Exercices.

Pour créer l'utilisateur, il suffit de cliquer sur le lien **Ajouter un utilisateur**.


L'utilisateur se nommera **Admin** et aura comme mot de passe **Exercices**. L'exemple n'est pas excellent en terme de sécurité. Dans un cas réel, il faut employer des noms de compte moins génériques et surtout des mots de passe différents du login.




MySQL peut créer une base de données au nom de l'utilisateur, ou donner par défaut une liste de privilèges génériques. Ici nous allons laisser le choix par défaut **Aucune**.


Le fait de cliquer sur le lien **Tout cocher** activerait tous les droits possibles pour l'utilisateur Admin, nous n'allons pour l'instant donner **aucun droit** à cet utilisateur car sinon il aurait tous les droits cochés accordés sur toutes les bases de données.

Pour valider toutes ces actions, il suffit de cliquer sur le bouton **Exécuter**.

Comme le montre la capture d'écran ci-dessous, le nouvel utilisateur Admin a bien été ajouté à la liste des utilisateurs.


**Vue d'ensemble des utilisateurs**

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
	Utilisateur	Serveur	Mot de passe	Privilèges globaux 	"Grant"															
<input type="checkbox"/>	Admin	%	Oui	ALL PRIVILEGES	Oui															
<input type="checkbox"/>	root	localhost	Oui	ALL PRIVILEGES	Oui															


[Tout cocher](#) / [Tout décocher](#)

Sur la droite de la ligne concernant le compte Admin il existe une icône représentant un utilisateur avec un stylo. Cette icône permet de changer les privilèges. Nous allons cliquer dessus pour donner des droits à ce compte sur la base de données Exercices.

Privileges spécifiques à une base de données

Base de données Privileges "Grant" Privileges spécifiques à une table

aucune

Ajouter des privileges sur cette base de données: Entrez une valeur:

Entrez une valeur:  
information\\_schema  
mysql  
phpmyadmin  
**Exercices**  
test

Modifier le mot de passe

☐ aucun mot de passe

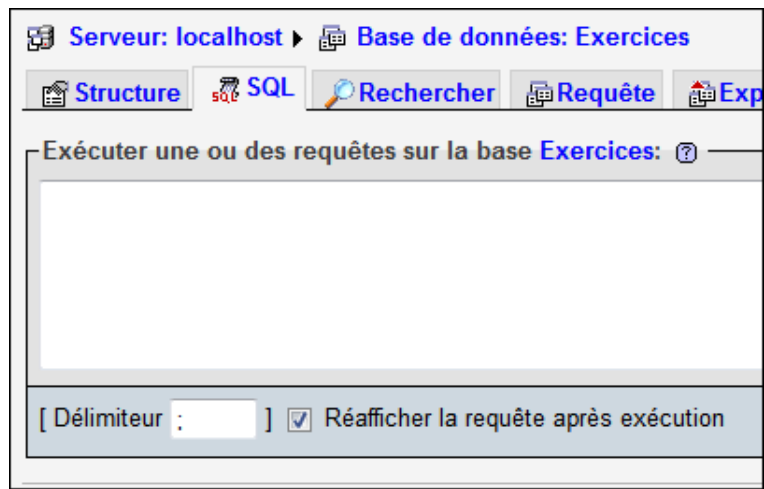
Dans la nouvelle fenêtre qui apparaît, il est possible d'affecter des droits seulement sur une base de données. Il suffit de sélectionner la base **Exercices** dans la liste déroulante. Le fait de choisir une base entraîne l'ouverture d'une nouvelle fenêtre.

Privileges spécifiques à une base de données ([Tout cocher](#) / [Tout décocher](#))

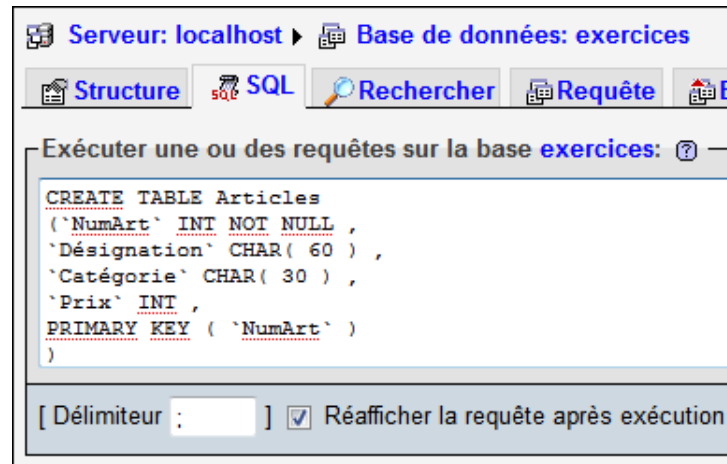
*Veillez noter que les noms de privileges sont exprimés en anglais*

Données	Structure	Administration
<input checked="" type="checkbox"/> SELECT	<input checked="" type="checkbox"/> CREATE	<input checked="" type="checkbox"/> GRANT
<input checked="" type="checkbox"/> INSERT	<input checked="" type="checkbox"/> ALTER	<input checked="" type="checkbox"/> LOCK TABLES
<input checked="" type="checkbox"/> UPDATE	<input checked="" type="checkbox"/> INDEX	<input checked="" type="checkbox"/> REFERENCES
<input checked="" type="checkbox"/> DELETE	<input checked="" type="checkbox"/> DROP	
	<input checked="" type="checkbox"/> CREATE TEMPORARY TABLES	
	<input checked="" type="checkbox"/> CREATE VIEW	
	<input checked="" type="checkbox"/> SHOW VIEW	
	<input checked="" type="checkbox"/> CREATE ROUTINE	
	<input checked="" type="checkbox"/> ALTER ROUTINE	
	<input checked="" type="checkbox"/> EXECUTE	

En cliquant sur l'onglet **SQL**, les différentes requêtes vont pouvoir être testées.



Par exemple, voici la requête de création de la table Articles.



Comme vous pouvez le constater, il peut être nécessaire de faire quelques « retouches » par rapport aux requêtes originelles. Ici, par exemple, les noms des champs doivent être entre apostrophes.

À partir de maintenant, vous pouvez vous amuser à tester les requêtes SQL vues précédemment dans le système de gestion de bases de données MySQL !

## Exercices applicatifs

Cette partie va nous permettre de travailler un peu plus avec la méthode Merise.

Les exercices sont résolus et certains points seront expliqués.

Comme toute méthode informatique doit répondre à quelques objectifs principaux, voici les conseils d'usages :

- Bien définir ce que l'utilisateur final veut informatiser et la faisabilité.
- Vérifier la cohérence de sa demande.
- Structurer les données à informatiser.
- Se rappeler que « qui peut le plus peut le moins ».
- Passer du temps sur l'analyse.

Pour bien définir l'expression des besoins de l'utilisateur final la méthodologie Merise nous est d'une grande aide. En effet, en raison de son caractère graphique et de sa sémantique simple, la méthode Merise peut être comprise par un non informaticien. Il est plus simple de faire valider un Modèle Conceptuel des Données qu'un brouillon de prise de notes. L'avantage de faire valider un modèle Merise est de pousser l'utilisateur final à réfléchir à l'ensemble de son système d'information.

La vérification de la cohérence de la demande consiste à bien délimiter le périmètre du système d'information et de ses sous-systèmes. Là encore Merise, grâce à ses modèles, va permettre un découpage fin des règles métier et mettre en évidence les interactions des traitements.

Bien structurer les données, c'est concevoir une ou des applications solides et capables de s'adapter facilement aux évolutions futures du système d'information.

Se rappeler que « qui peut le plus, peut le moins » doit permettre d'éviter les logiciels trop statiques, car conçu de façon rigide pour répondre à un besoin à un instant T et non en prenant en compte certaines évolutions possibles de la donnée. Par exemple dans une bibliothèque municipale le logiciel de gestion des ouvrages conçu pour répondre à la règle de gestion suivante : « Un emprunteur ne peut emprunter qu'au maximum 3 livres à la fois », bloque physiquement le prêt à trois livres sans laisser la possibilité de modifier cette règle dans le futur.

Passer du temps à l'analyse... Passer **beaucoup** de temps à l'analyse. Nombre de personnes foncent immédiatement dans la phase de codage et délaissent l'analyse. Or les reprises de code, les modifications, les errements dans les parties de programmes pour corriger une phase mal conçue sont générateurs de perte de temps.

Pour une société, reprendre un logiciel « mal ficelé » peut s'avérer coûter aussi cher que ce qu'elle l'a vendu.

# Premier exercice

## Énoncé

Un agriculteur, Monsieur Bousquet, fait de la vente directe de ses produits ou animaux qu'il élève. Il vend des lapins, des poules, des dindes, des veaux, des cochons. Selon la saison il vend aussi des légumes (choux, pommes de terre, carottes...) et des fruits (fraises, poires, pommes...). Il ne fait que de la vente directe. Suite à votre discussion, il ressort les informations suivantes.

À l'heure actuelle, les ventes sont inscrites sur trois cahiers distincts :

- Un pour les animaux.
- Un pour les fruits.
- Un pour les légumes.

Tout est vendu au kilo, les animaux sont pesés vivants avant d'être vendus.

Il souhaiterait un logiciel simple pour saisir les ventes journalières et pouvoir éditer un récapitulatif mensuel par type de vente (animaux, légumes et fruits) et par produit (poulets, lapins, poireaux, poires...) pour sa comptabilité.

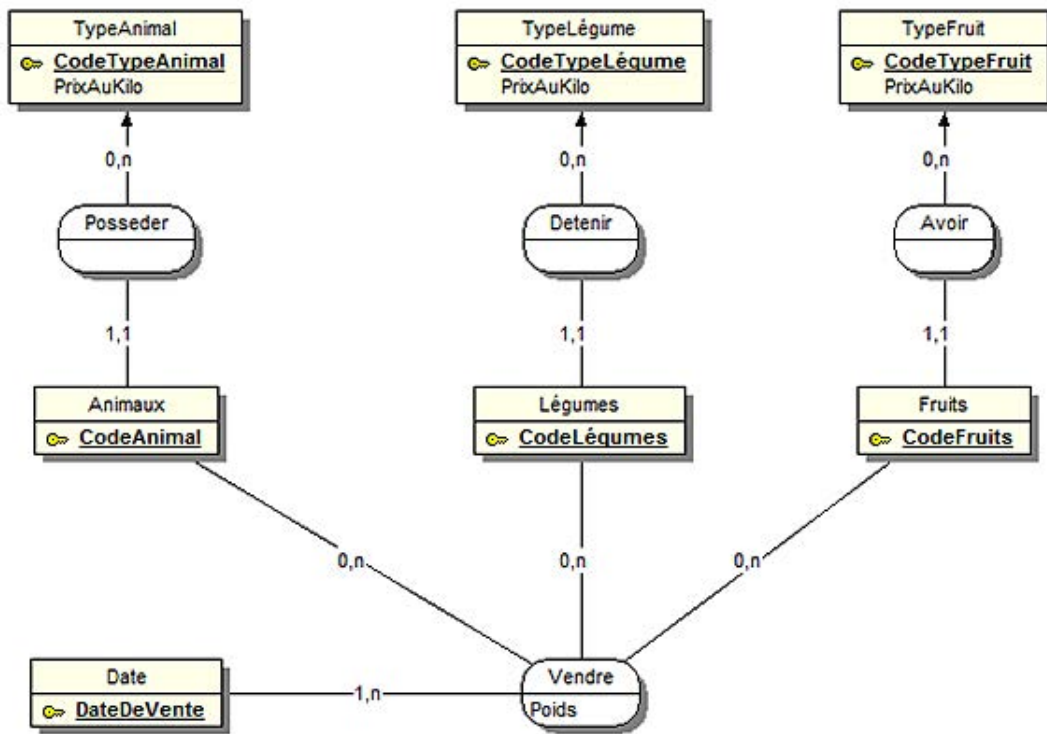
## Travail à faire

- Créer le Modèle Conceptuel des Données.
- Concevoir le Modèle Logique des Données.
- Finir par le Modèle Physique des Données.

## 1. Solutions

### a. Le Modèle Conceptuel des Données

Voici un premier modèle conceptuel des données, qui pourrait être réalisé. Attention ce modèle présente des imperfections structurelles qui vont pénaliser la performance, la maintenance et l'intégrité de l'applicatif. Le voici :



Au premier regard ce modèle peut sembler cohérent. Si nous y regardons de plus près et imaginons la structure physique de la table Vendre nous verrions ceci :

Vendre(#CodeAnimal,#CodeLégumes,#CodeFruits,#DateDeVente,Poids)

Imaginons le fichier :

#CodeAnimal	#CodeLégumes	#CodeFruits	#DateDeVente	Poids
1	NULL	NULL	06/01/09	2
<b>2</b>	<b>15</b>	<b>5</b>	<b>06/01/09</b>	<b>6</b>

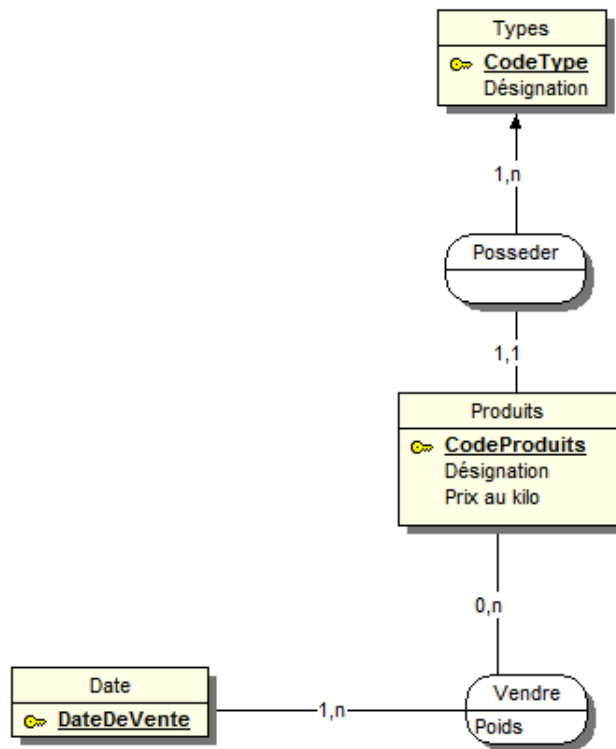
### Incohérences

La première ligne fait apparaître deux codes vides (NULL). Le fichier risque donc de contenir des cellules vides, ce qui va l'alourdir inutilement.

La ligne deux présente un inconvénient majeur, en effet à quel produit correspond le poids inscrit (6 kilos) :

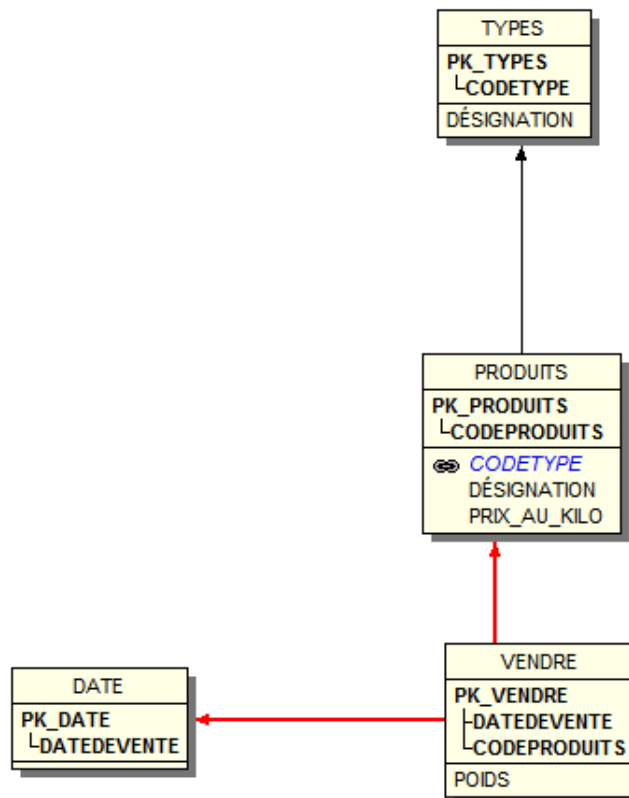
- L'animal de code 2 ?
- Le légume de code 15 ?
- Le fruit de code 5 ?

Voici une autre possibilité optimisée par rapport à la précédente :



## b. Le Modèle Logique des Données

Ce modèle semble plus cohérent, en voici le modèle logique tel que généré par Win'Désign :





### c. Le modèle relationnel

Et enfin, concevons le modèle relationnel :

Types(**Codetype**, Désignation)

Produits(**CodeProduits**, Désignation, Prix\_au\_kilo, **#Codetype**)

Date(**DateDeVente**)

Vendre(**#CodeProduits**, **#DateDeVente**, Poids)

Remplissons virtuellement la table Types.

CodeType	Désignation
1	Animaux
2	Fruits
3	Légumes

La table Produits pourrait ressembler à ceci :

CodeProduits	Désignation	Prix_au_Kilo	#Codetype
1	Lapin	7	1
2	Veau	11	1
3	Salade	1,2	3
4	Endives	11	3
5	Pommes	5	2
6	Noisettes sèches	15	2

Et la table Vendre :

CodeProduits	DateDeVente	Poids
1	07/01/09	2
1	08/01/09	1
6	08/01/09	0,5
2	08/01/09	4

Voilà, le modèle devient cohérent. Si l'on regarde la première ligne de la table Vendre, nous pouvons voir qu'il a été vendu un produit de code numéro 1 (un lapin à 7 euros le kilo de CodeType 1 c'est-à-dire Animaux), le 7 janvier 2009 pour 2 kilos.

Le total dû sera calculé donc il n'est pas nécessaire de le stocker.

Le logiciel développé suite à cette analyse présente toutes les informations nécessaires pour pouvoir répondre aux besoins de Monsieur Bousquet.

## Deuxième exercice

### Énoncé

Voici un modèle relationnel décrivant une nomenclature de conception d'un meuble. Le meuble est un ensemble composé de sous-ensembles et de composants divers. Un sous-ensemble est élaboré grâce à un assemblage de composants.

À partir de ce modèle relationnel, il vous est demandé de procéder à du *reverse engineering* ou en français de la **rétro-ingénierie**. C'est-à-dire de remonter jusqu'au modèle conceptuel en passant par le modèle logique des données.

### Modèle relationnel

Ensembles(**CodeEnsemble**, Désignation)

Sous-Ensembles(**CodeSousEnsemble**, Désignation, Longueur, Largeur, Hauteur, Prix\_Unitaire)

Composants(**CodeComposant**, Désignation, Prix\_Unitaire)

LienEnsSE(**#CodeEnsemble**, **#CodeSousEnsemble**, Qté)

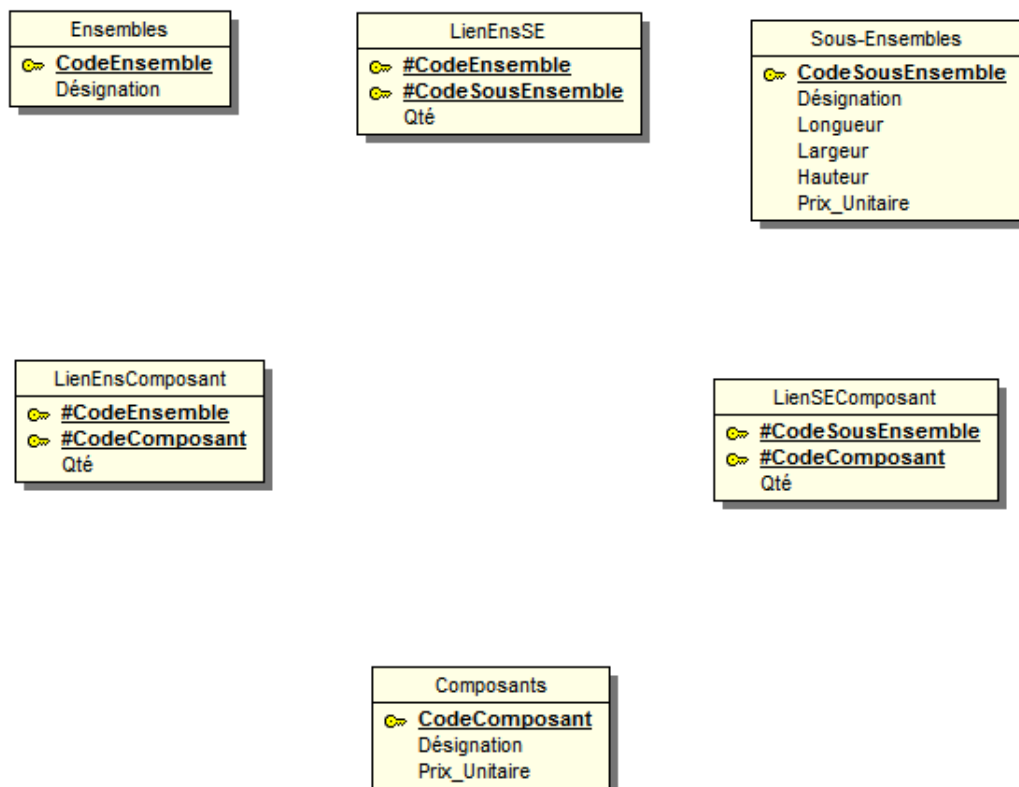
LienEnsComposant(**#CodeEnsemble**, **#CodeComposant**, Qté)

LienSEComposant(**#CodeSousEnsemble**, **#CodeComposant**, Qté)

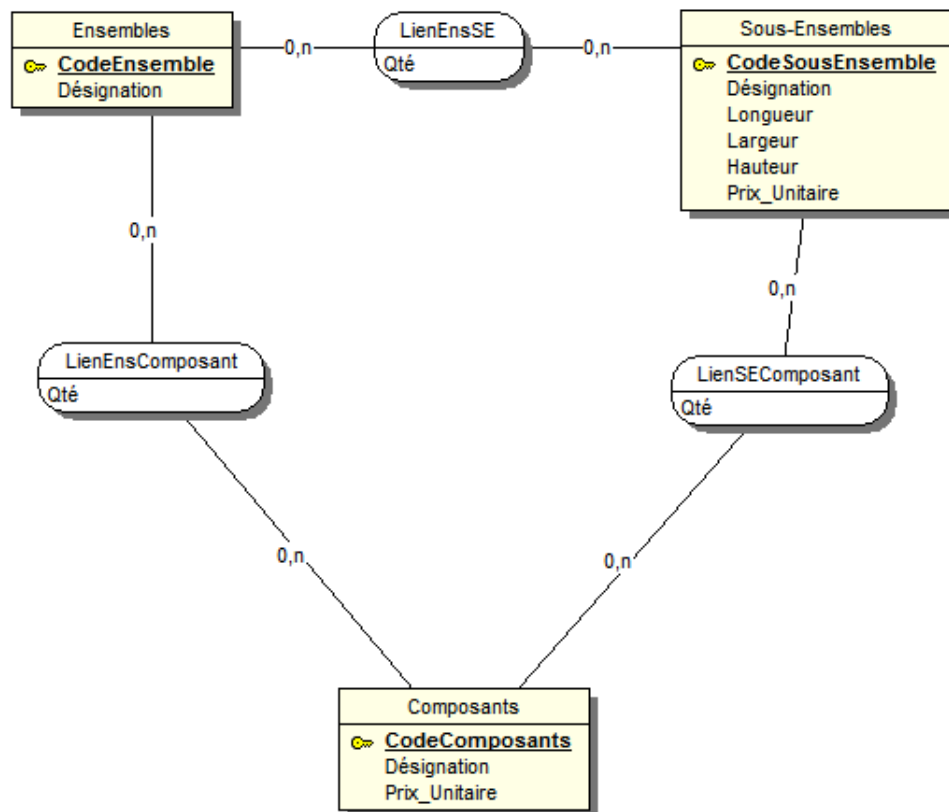
## 1. Solutions

### a. Le Modèle Logique des Données

Voici le modèle logique que l'on peut concevoir à partir du modèle relationnel :



### b. Le Modèle Conceptuel des Données



## Troisième exercice

### Énoncé

La nouvelle loi sur l'auto-entrepreneuriat vient d'être promulguée et vous vous dites que c'est peut-être le moment de vous mettre à votre compte.

Comme toutes les personnes de votre village font appel à vos services lorsqu'ils ont un problème informatique, vous êtes sûr que votre affaire va fonctionner.

Pour démarrer il vous faut un petit logiciel vous permettant de saisir vos interventions pour faciliter la tenue de votre comptabilité.

Ce logiciel permettra la saisie des coordonnées des clients et le matériel sur lequel vous êtes intervenu.

Vous décidez d'appliquer un prix horaire différent selon le type d'intervention (certaines réparations ou manipulation complexes doivent être facturées plus cher).

Pour certaines pannes vous vendrez le composant neuf. Le logiciel devra donc intégrer la vente de matériel inhérente à la réparation.

### Travail à faire

- Concevoir le dictionnaire des données simplifié.
- Concevoir le Modèle Conceptuel des Données.
- Concevoir le Modèle Logique des Données.
- Concevoir le Modèle Physique des Données.

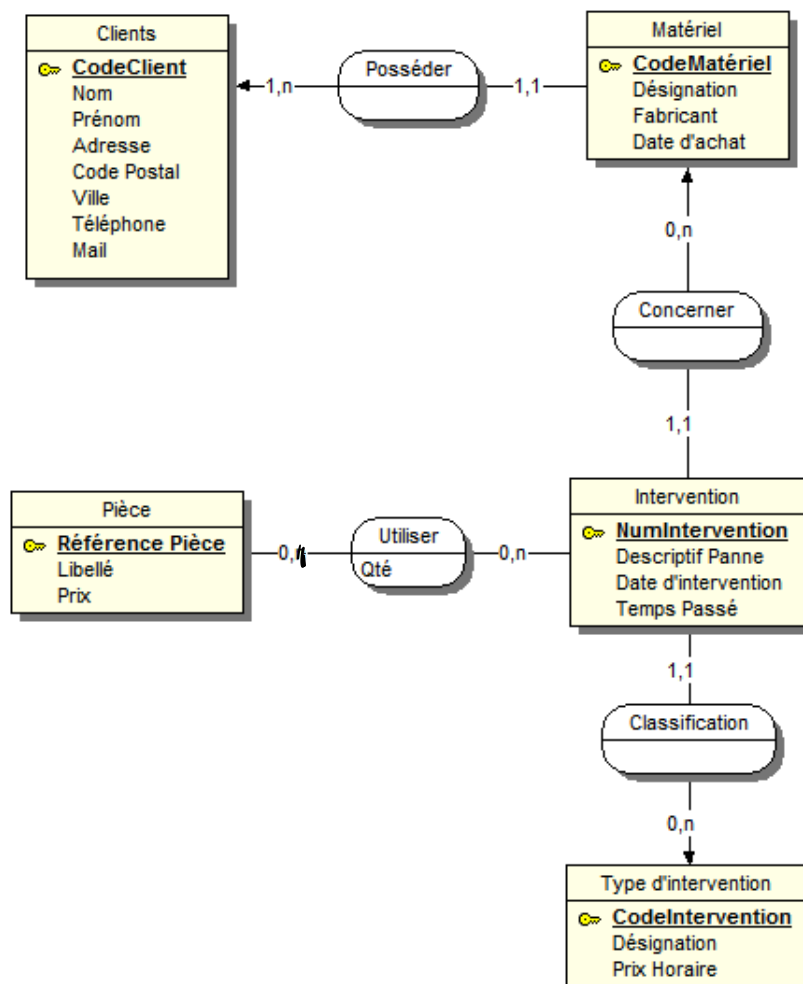
## 1. Solutions

### a. Dictionnaire des données simplifié

Nom de la donnée	Format	Longueur	Type
CodeClient	Alphanumérique	15	Élémentaire
Nom	Alphabétique	30	Élémentaire
Prénom	Alphabétique	30	Élémentaire
Adresse	Alphabétique	30	Élémentaire
Code Postal	Alphanumérique	6	Élémentaire
Ville	Alphabétique	30	Élémentaire
Téléphone	Alphanumérique	15	Élémentaire
Mail	Alphanumérique	30	Élémentaire
CodeMatériel	Alphanumérique	15	Élémentaire
Désignation	Alphanumérique	60	Élémentaire
Fabricant	Alphanumérique	60	Élémentaire

Date d'achat	Date		Élémentaire
NumIntervention	Alphanumérique	15	Élémentaire
Descriptif Panne	Alphanumérique	60	Élémentaire
Date d'intervention	Date		Élémentaire
Temps Passé	Numérique		Élémentaire
CodeIntervention	Alphanumérique	15	Élémentaire
Désignation	Alphanumérique	60	Élémentaire
Prix horaire	Numérique		Élémentaire
Référence Pièce	Alphanumérique	15	Élémentaire
Libellé	Alphanumérique	60	Élémentaire
Prix	Numérique		Élémentaire

## b. Le Modèle Conceptuel des Données



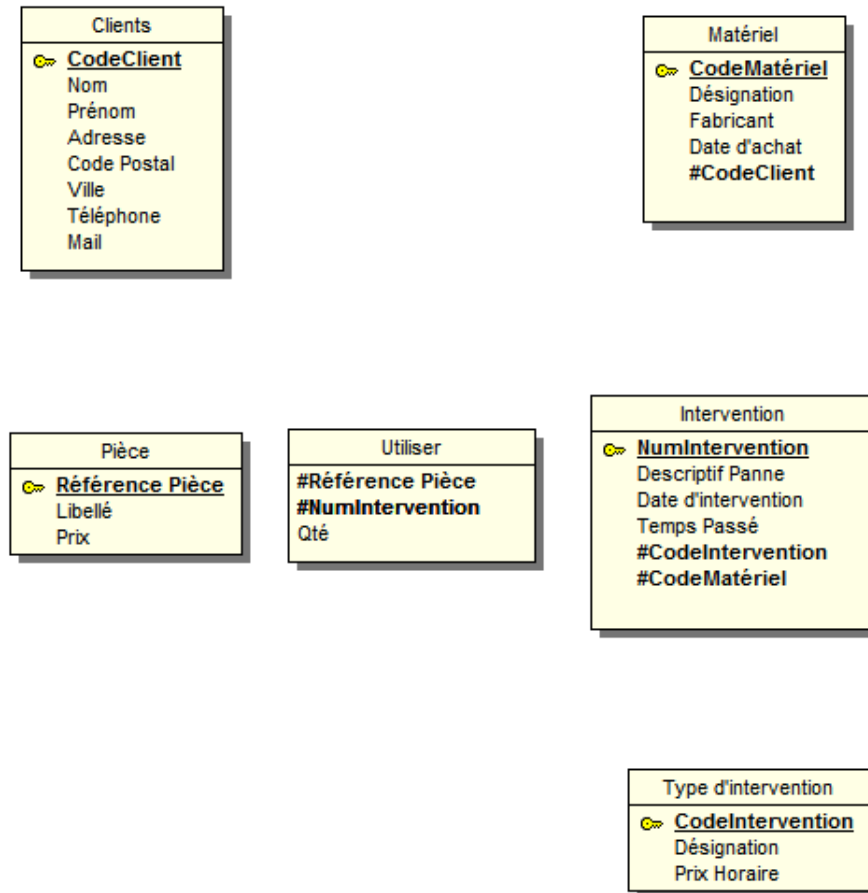
## Explications

Un client peut posséder un ou plusieurs matériels. Une intervention concerne un et un seul matériel et un matériel précis peut nécessiter zéro ou plusieurs interventions.

Une pièce détachée (par exemple un disque dur de référence DD001) peut être utilisée lors d'une intervention dans une quantité précise.

Une intervention est classifiée selon un et un seul type, à un type d'intervention précis peut correspondre zéro ou plusieurs interventions.

### c. Le Modèle Logique des Données



Le Modèle Logique des Données ne présente aucune difficulté. La relation porteuse **Utiliser** est transformée en entité. Les clés primaires des entités faisant partie de la relation (**Pièce et Intervention**) sont intégrées et deviennent les nouvelles clés étrangères de la relation **Utiliser**.

### d. Le modèle physique des données

Clients(CodeClient, Nom, Prénom, Adresse, Code Postal, Ville, Téléphone, Mail)

Matériel(CodeMatériel, Désignation, Fabricant, Date d'achat, #CodeClient)

Pièce(Référence Pièce, Libellé, Prix)

Utiliser(#Référence Pièce, #NumIntervention, Qté)

Intervention(NumIntervention, Descriptif Panne, Date d'intervention, Temps Passé, #CodeIntervention, #CodeMatériel)

Type d'intervention(CodeIntervention, Désignation, Prix Horaire)

## Quatrième exercice

Vous êtes missionné par un de vos amis qui exerce la profession d'agent immobilier pour lui réaliser un petit programme.

Il désire un logiciel dans lequel il peut inscrire son fichier des maisons, des propriétaires et des locataires.

### Règles de gestion

Une maison appartient à une ou plusieurs personnes.

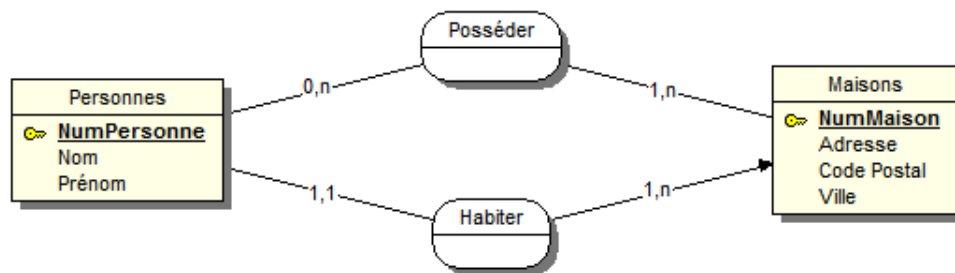
Une personne peut être propriétaire d'une maison et en louer une autre.

### Travail à faire

- Créer le Modèle Conceptuel des Données.
- Concevoir le Modèle Logique des Données.
- Finir par le Modèle Physique des Données.

## 1. Solutions

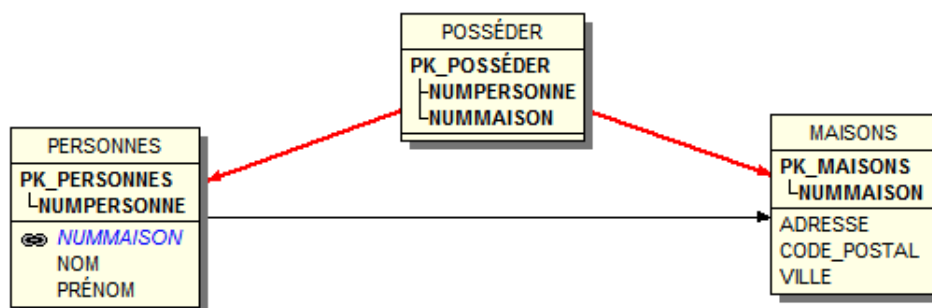
### a. Modèle Conceptuel des Données



Ici, nous voyons deux relations entre deux entités. Le traitement de ce cas de figure est classique. Les règles vues précédemment s'appliquent intégralement.

### b. Modèle Logique des Données

Voici une représentation Win'design. Une nouvelle entité est créée et une clé étrangère est glissée dans l'entité Personnes.



### c. Modèle Physique des Données

Voici le Modèle Physique des Données que l'on peut concevoir à partir du MLD précédent.

Personnes(**NumPersonne**, Nom, Prénom, **#NumMaison**)

Maisons(**NumMaison**, Adresse, Code Postal, Ville)

Posséder(**#NumPersonne**, **#NumMaison**)



## Cinquième exercice

ASSUR'AUTO, comme son nom l'indique, est une petite société d'assurance spécialisée dans les contrats d'assurance automobile. Malgré son envergure restreinte (elle dispose tout de même de plusieurs agences et plusieurs employés sur le territoire) elle assure aussi bien les véhicules de tourisme que les véhicules utilitaires.

Pour assurer un véhicule, son propriétaire, dont on enregistre le nom, le prénom, l'adresse et les coordonnées (téléphone, fax éventuel, e-mail...), doit fournir au conseiller de l'agence la carte grise du véhicule afin que l'on enregistre son type, sa marque, son numéro d'immatriculation, sa date de mise en circulation et sa puissance fiscale. S'il s'agit d'un véhicule de tourisme, on enregistre aussi le nombre de portes et de passagers autorisés, tandis que s'il s'agit d'un véhicule utilitaire on enregistre le poids à vide, le poids autorisé en charge, la longueur, la largeur, la hauteur.

Chaque contrat, établi à une certaine date, est référencé par un numéro de contrat et est d'une certaine catégorie : tous risques, au « tiers »...

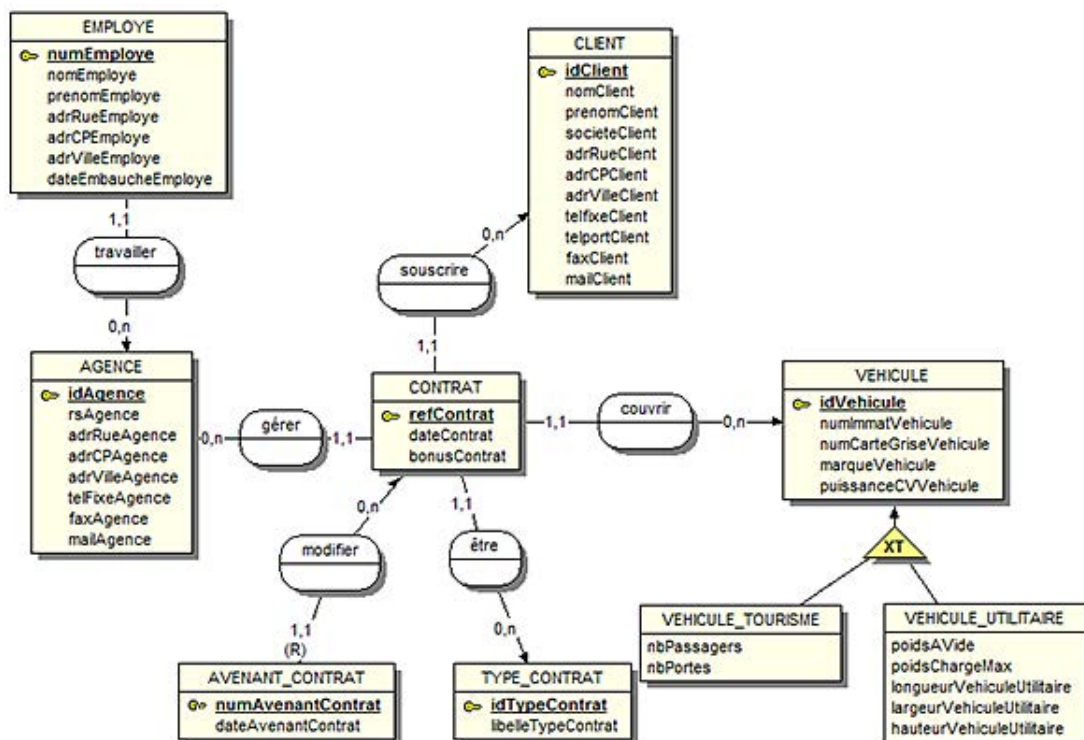
Le contrat est attaché à la personne, pas au véhicule : lorsqu'il y a changement de véhicule le propriétaire conserve le bonus ou le malus attaché à ce contrat qui est alors reporté sur le nouveau véhicule.

### Travail à faire

- Créer le Modèle Conceptuel des Données.
- Concevoir le Modèle Logique des Données.

## 1. Solutions

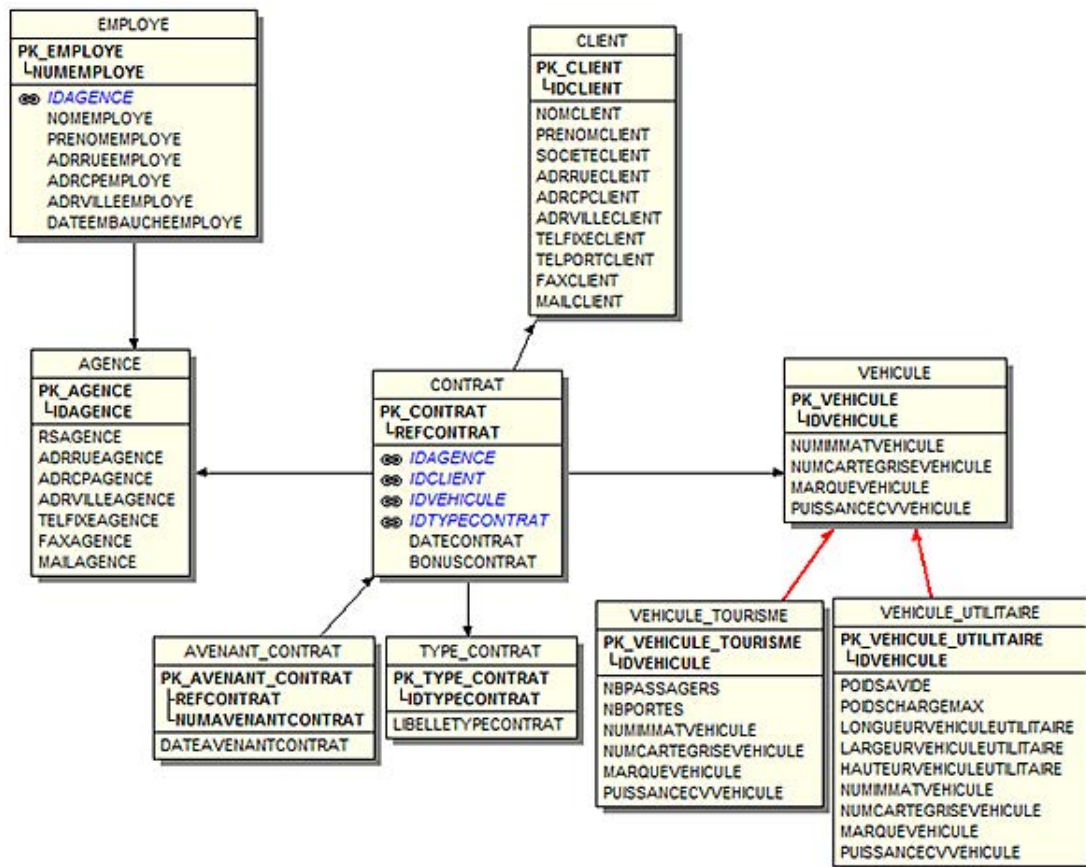
### a. Modèle Conceptuel des Données



Ici, nous pouvons appliquer l'héritage, comme nous pouvons le constater l'entité **Véhicule\_tourisme** hérite des propriétés de l'entité **Véhicule**. Il en est de même pour l'entité **Véhicule\_utilitaire**. XT nous indique qu'un véhicule est soit utilitaire, soit tourisme mais pas les deux.

### b. Modèle Physique des Données

Voici le Modèle Physique des Données qui découle du MCD précédent.



## Sixième exercice

L'entreprise XProd fabrique et commercialise divers produits. Ils sont identifiés par une référence propre à XProd et on enregistre une désignation (libellé court), un descriptif (libellé long) et un prix de vente catalogue unitaire hors taxes.

Dans la base de données elle gère deux types de produits :

- les produits qu'elle fabrique pour lesquels on enregistre le nombre moyen d'heures de main d'œuvre nécessaire à leur fabrication ;
- les produits dits « approvisionnés » parce qu'elle ne les fabrique pas : ils sont achetés à un ou plusieurs fournisseurs à un prix d'achat unitaire moyen.

Pour ne pas dépendre d'un fournisseur, enregistré par ses raison sociale, adresse, etc., pour chaque produit approvisionné l'entreprise a établi une liste de fournisseurs capables de livrer ce produit. Bien entendu pour un même produit chaque fournisseur peut avoir sa propre référence et un prix différent.

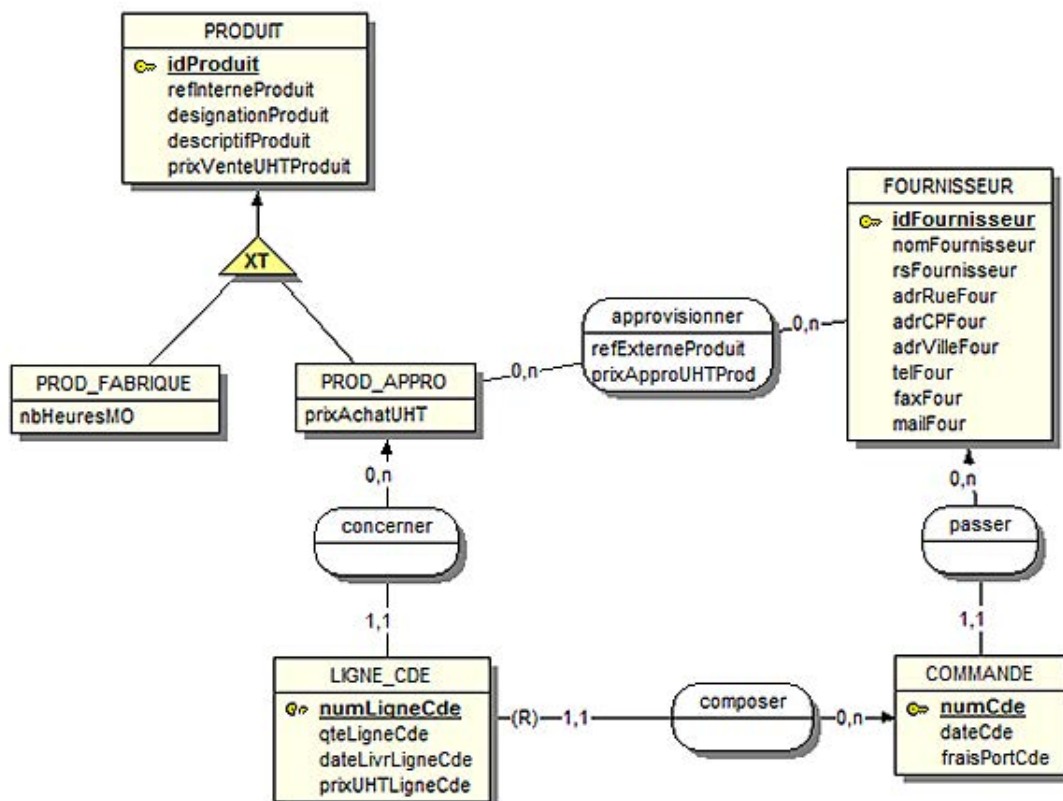
Lorsque XProd passe une commande à une certaine date à un fournisseur, elle essaie de grouper plusieurs lignes de commande : une par produit dans une certaine quantité avec sa date de livraison prévue, pour réduire les frais de livraison de la commande et essayer de négocier un prix d'achat unitaire inférieur au prix catalogue du fournisseur.

### Travail à faire

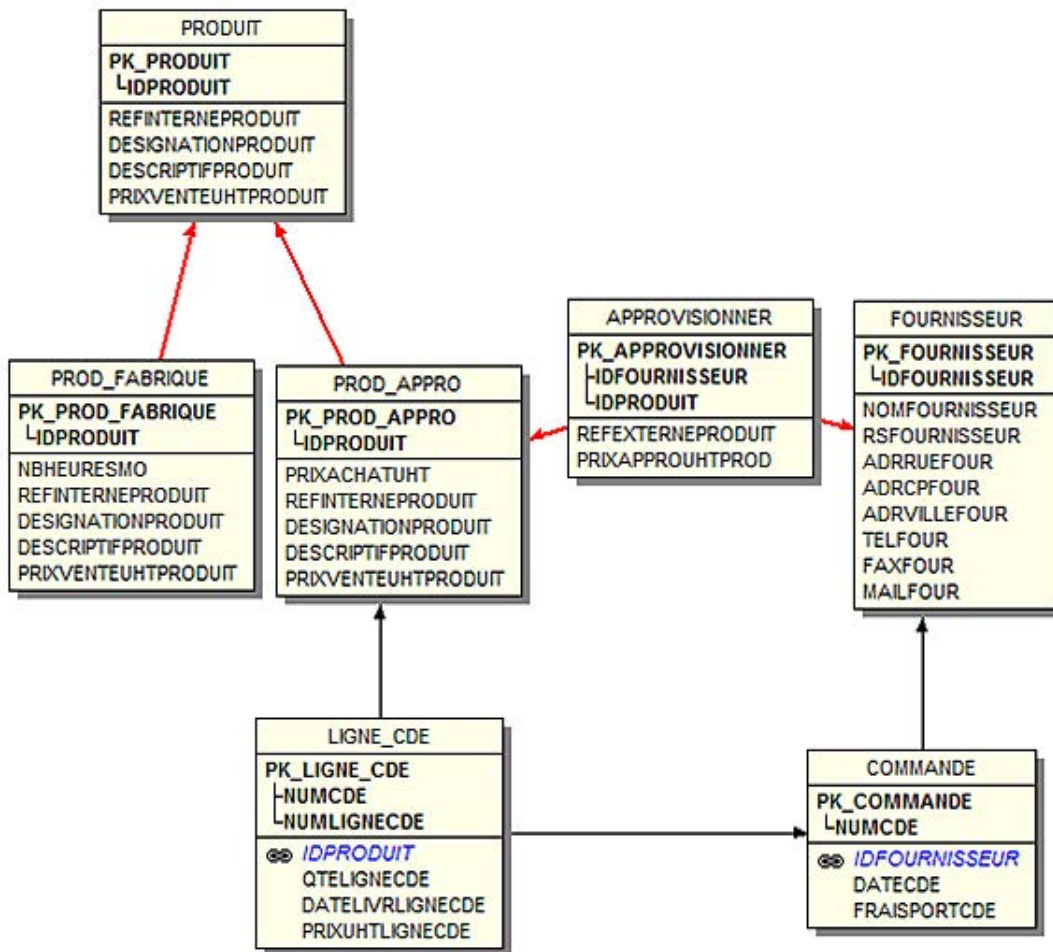
- Créer le Modèle Conceptuel des Données.
- Concevoir le Modèle Logique des Données.

## 1. Solutions

### a. Modèle Conceptuel des Données



## b. Modèle Logique des Données



## Septième exercice

Les fédérations de sport proposant des compétitions composées de plusieurs sports ou épreuves, comme le biathlon, triathlon et autre décathlon vous ont demandé d'analyser et de développer un logiciel générique pouvant gérer l'organisation de leurs compétitions. Voici quelques éléments vous permettant de commencer l'analyse.

Les sportifs s'inscrivent à une compétition. Lors de cette inscription on enregistre son nom, son prénom, son adresse et ses coordonnées téléphoniques, fax et e-mail. Il se voit attribuer un numéro de dossard dans cette compétition, qui servira aussi à retrouver son dossier d'inscription.

Attention : un sportif peut être licencié à la fédération via un club ou pas, les amateurs sont parfois autorisés à concourir. C'est pourquoi pour un sportif licencié on enregistre bien sûr son numéro de licence et son club, tandis que pour un sportif amateur on exigera seulement un certificat médical daté de moins de trois mois délivré par un médecin du sport pour des questions d'assurance.

Une compétition a lieu à une certaine date dans une certaine ville et porte éventuellement un libellé comme « Grand prix de printemps ». Chaque compétition est composée d'un certain nombre d'épreuves effectuées dans un certain ordre : pour certaines compétitions il y a d'abord une épreuve de 3 km de natation suivi de 50 km à bicyclette et enfin 20 km de course à pied ; pour d'autres cela commencera par une escalade d'un mur de niveau 3, continuera par une randonnée pédestre de 10 km et se terminera par un parcours en traîneau tiré par des chiens... Bref chaque épreuve est d'un certain type et il faut spécifier alors sa distance et les conditions de réalisation.

### Travail à faire :

Au niveau des données :

- Créer le dictionnaire des données.
- Créer le Modèle Conceptuel des Données.
- Concevoir le Modèle Logique des Données.
- Concevoir le Modèle Relationnel des Données.

Au niveau des traitements :

- Créer le modèle de contexte de niveau 0 concernant l'inscription d'un sportif licencié d'un club à une compétition.

Lors de l'inscription, un stand est mis en place avec plusieurs bénévoles. L'un d'eux est chargé de vérifier la licence et de réaliser la pré-inscription du sportif sur la compétition. Il communique les frais d'inscription au sportif. Une fois les frais d'inscription connus, le sportif les règle à une deuxième personne chargée de la comptabilité qui lui remet une facture acquittée. Avec cette facture acquittée, le sportif se déplace jusqu'à un troisième bénévole qui lui remet son numéro de dossard et transmet au premier bénévole un bon de validation.

- Créer le modèle de flux conceptuel de niveau 1, représentant le processus de validation de l'inscription à une compétition.
- Créer le modèle organisationnel des traitements.
- Créer une requête SQL qui liste l'ensemble des sportifs habitant Perpignan.

## 1. Solutions

### a. Le dictionnaire des données

Voici un dictionnaire des données très simplifié :

Nom de la donnée	Format	Longueur
Nom du sportif	Alphabétique	30

Prénom du sportif	Alphabétique	20
Adresse	Alphanumérique	60
Code postal	Alphanumérique	5
Ville	Alphanumérique	60
Téléphone	Alphanumérique	15
Fax	Alphanumérique	15
Mail	Alphanumérique	40
Numéro de licence	Alphanumérique	10
Club	Alphanumérique	60
Commentaire certificat médical	Alphanumérique	80
Date certificat médical	Date	8
Médecin traitant	Alphabétique	30
Date d'inscription	Date	8
Catégorie de l'inscription	Alphanumérique	20
Date compétition	Date	8
Libellé de la compétition	Alphanumérique	40
Ville de compétition	Alphanumérique	60
Distance de l'épreuve	Numérique	6
Descriptif de l'épreuve	Alphanumérique	60
Libellé du type de l'épreuve	Alphanumérique	80

Les champs téléphone, fax sont de type alphanumérique car les utilisateurs peuvent saisir des caractères séparateurs.

Le code postal aussi est de type alphanumérique car si en France nous utilisons 5 chiffres, il n'en est pas de même dans d'autres pays. De plus, retenir cette règle simple :

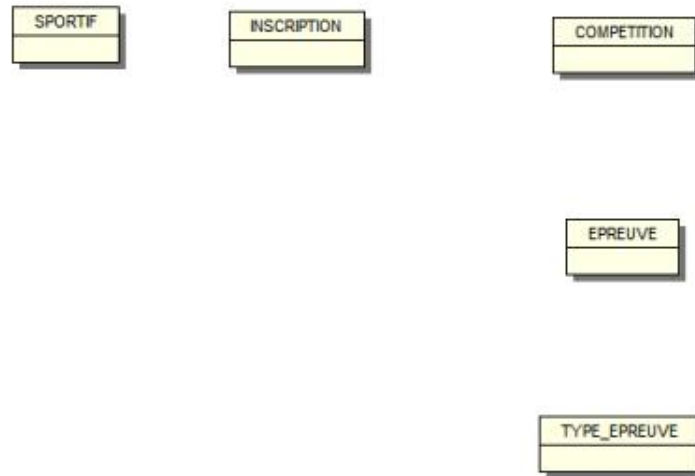
- Seuls les champs sur lesquels des calculs seront réalisés, doivent être déclarés en type numérique.

## **b. Modèle Conceptuel des Données**

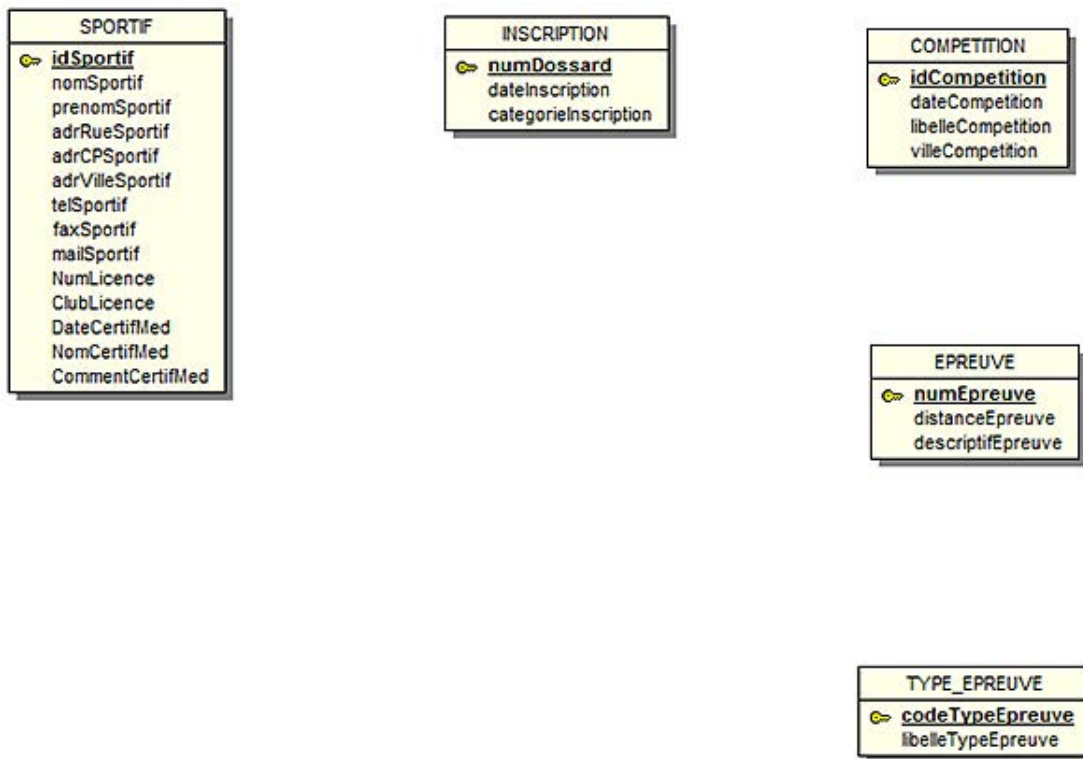
À la lecture du dictionnaire des données, nous pouvons dégager les entités suivantes :

- Sportif,
- Inscription,
- Compétition,
- Épreuve,

- Type de l'épreuve.

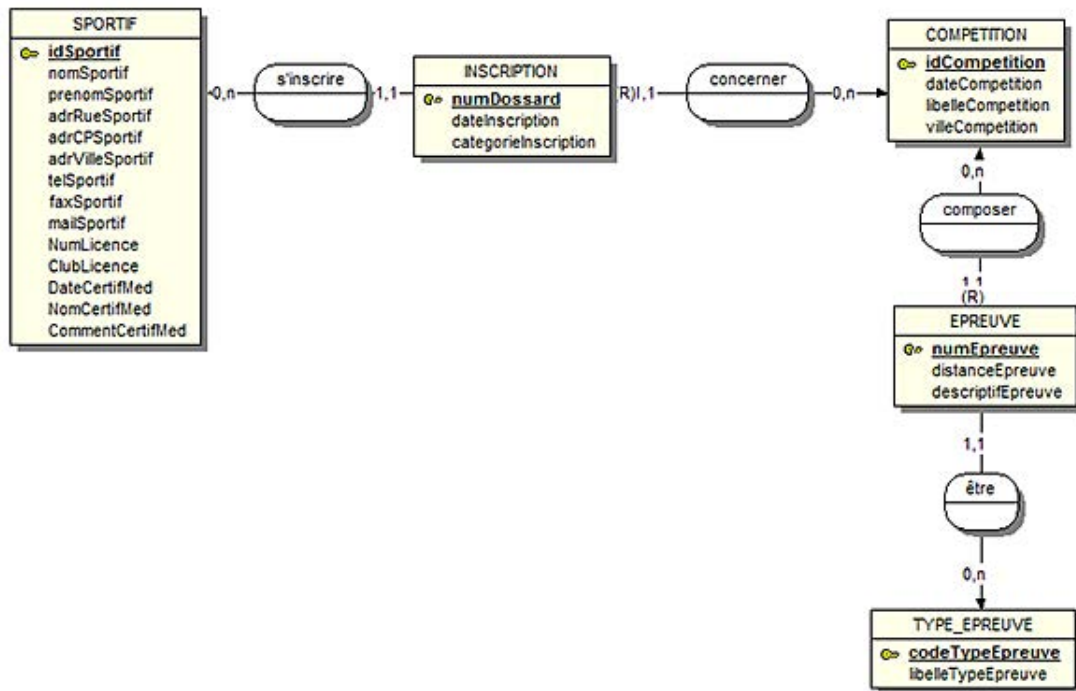


Voici maintenant les entités remplies avec leurs propriétés.



Intéressons-nous aux relations. Un sportif prend une inscription sur une compétition qui est composée d'épreuves d'un certain type.

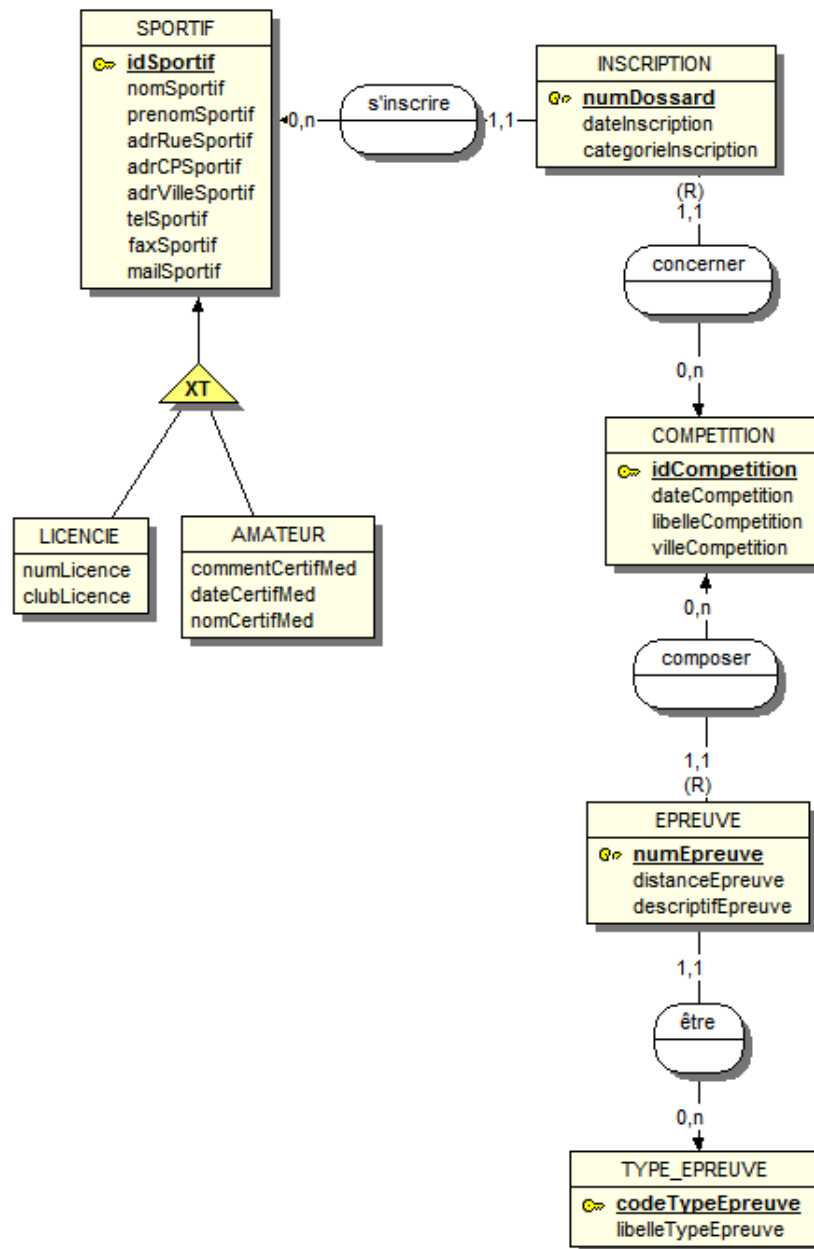




La présence d'identifiants relatifs indique que si une épreuve est supprimée, la compétition et l'inscription doivent être aussi supprimées.

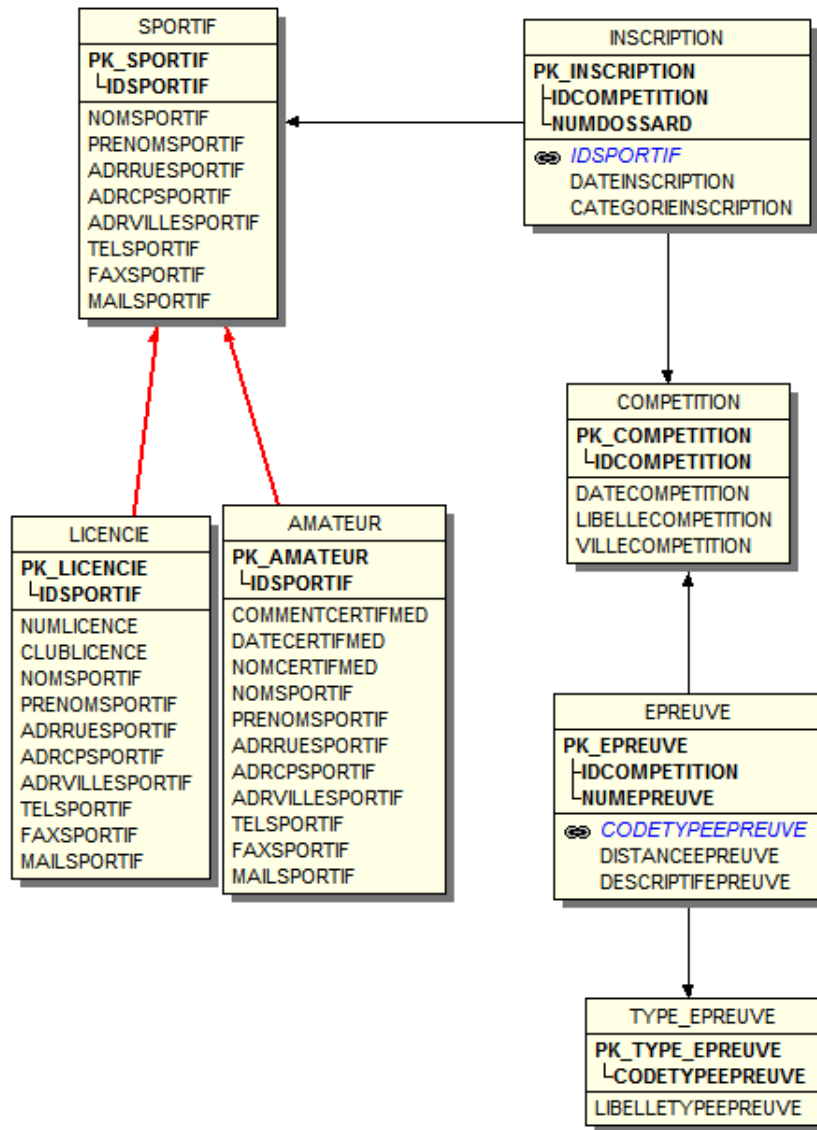
Si nous regardons attentivement ce modèle, nous pouvons aussi nous demander si nous ne pouvons pas spécialiser l'entité **sportif**. En effet, un sportif peut être soit licencié, soit amateur. Voici le modèle qui découle de cette remarque.





### c. Le Modèle Logique des Données

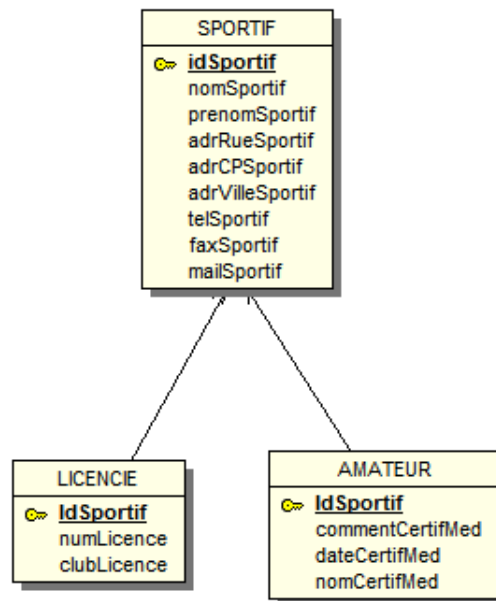
Le modèle logique qui en découle peut être le suivant :



Ce modèle a été généré par le logiciel Win'Design. Au niveau des entités Licencié et Amateur, nous avons plusieurs possibilités :

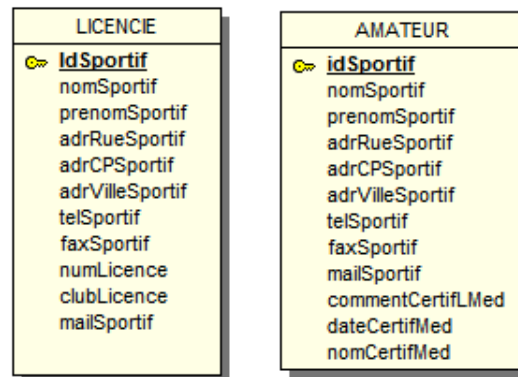
Soit, comme dans le schéma précédent, nous dupliquons la totalité de l'entité **Sur-type** dans les entités **Sous-types**. Cette solution entraîne des redondances d'informations, mais nous trouvons l'intégralité des informations du sportif dans chaque table, ce qui peut être un gage de rapidité d'accès à toutes les informations du sportif.

Soit nous ne dupliquons dans les entités « Sous-types » que l'identifiant de l'entité **Sur-type**.



Cette solution est la plus élégante, de plus l'espace nécessaire aux données est optimisé.

Soit nous ne conservons que les entités **Sous-types** après avoir dupliqué le contenu du **Sur-type**.



Cette solution évite de gérer la table Sportif, donc évite le doublonnage des données.

Soit nous passons toutes les propriétés des **Sous-Types** dans le **Sur-type**.



Cette solution, la plus simpliste, produira une table ayant un grand nombre de champs vides.

#### d. Le modèle relationnel des données

Sportif(**IdSportif**, nomSportif, prenomSportif, adrRueSportif, adrCPSportif, adrVilleSportif, telSportif, faxSportif, mailSportif)

Licencié(**IdSportif**, numLicence)

Amateur(**IdSportif**, commentCertifMed, dateCertifMed, NomCertifMed)

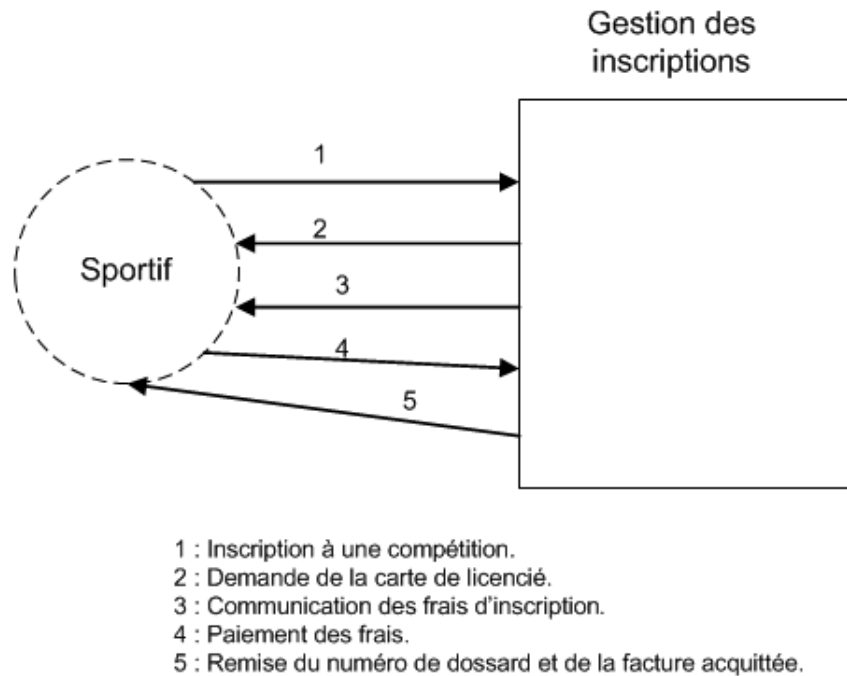
Inscription(**numDossard**, dateInscription, categorieInscription, **#IdSportif**, **#idCompétition**)

Compétition(**IdCompétition**, dateCompétition, libelleCompétition, villeCompétition)

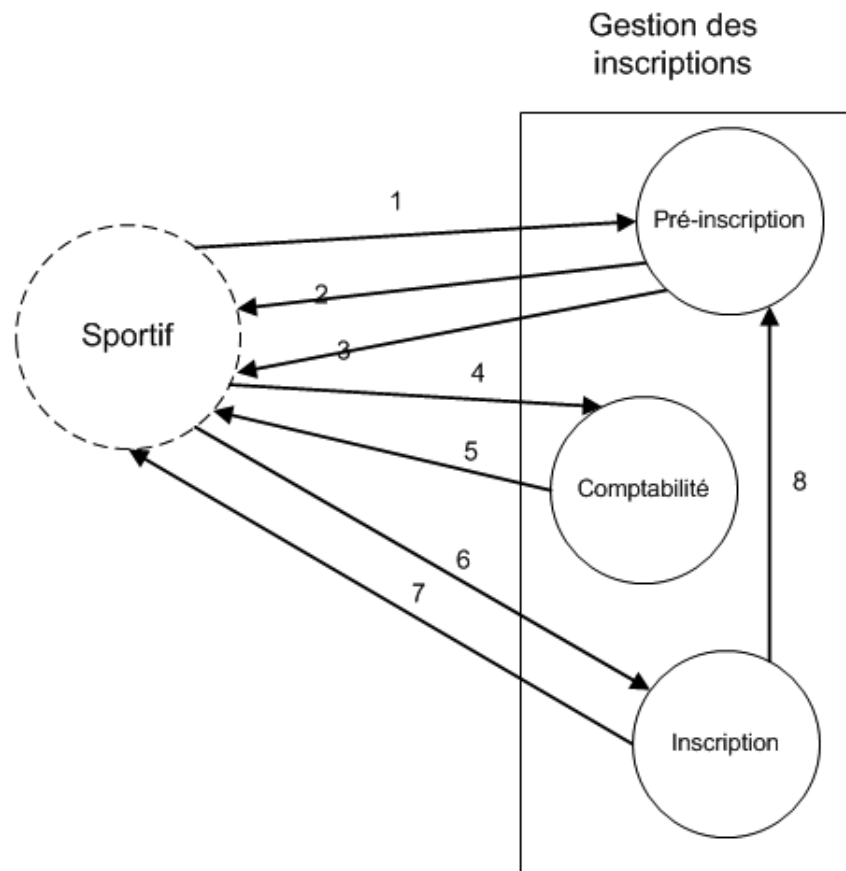
Epreuve(**numEpreuve**, distanceEpreuve, descriptifEpreuve, **#IdCompétition**, **#CodeTypeEpreuve**)

Type\_Epreuve(**CodeTypeEpreuve**, libelleTypeEpreuve)

#### e. Modèle de contexte de niveau 0

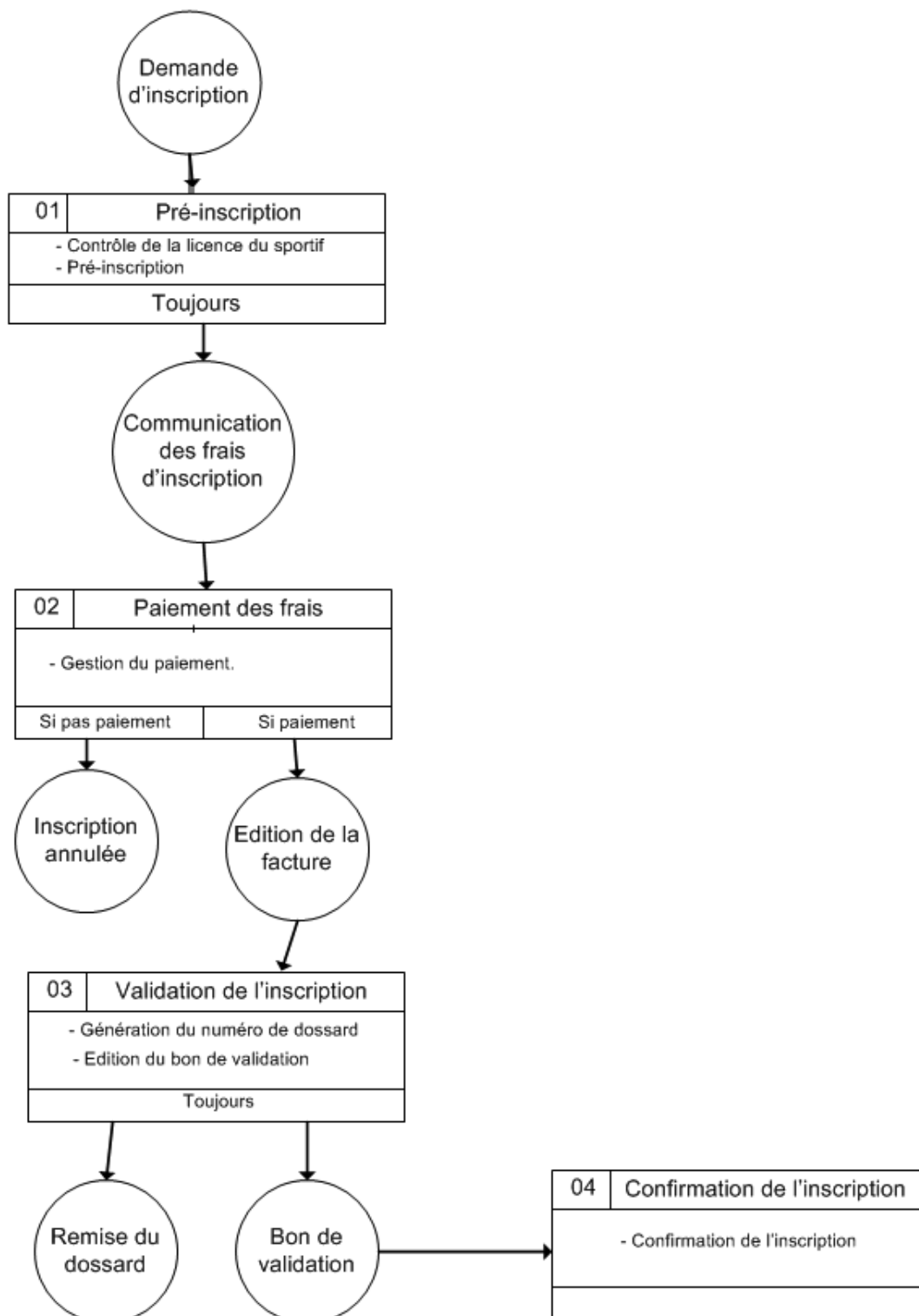


#### f. Le modèle de flux conceptuel de niveau 1



- 1 : Inscription à une compétition.
- 2 : Demande de la carte de licencié.
- 3 : Communication des frais d'inscription.
- 4 : Paiement des frais.
- 5 : Remise de la facture acquittée.
- 6 : Présentation de la facture acquittée.
- 7 : Remise du numéro de dossard.
- 8 : Création du bon de validation de l'inscription.

#### g. Le Modèle Organisationnel des Traitements



#### h. Requête SQL listant l'ensemble des sportifs habitant Perpignan

```

Select *From
Sportif
Where adrVilleSportif ='Perpignan';

```