Système d'information hospitaliers

Dr. BELAIDI Asma

Université de Tlemcen

Faculté de Technologie

Département Génie Biomédical

Email:*asma.belaidi@univ-tlemcen.dz*

Table des matières

I - **Présentation et étude des méthodes d'analyse** 3

1. Objectifs 3

2. Introduction 3

3. Le modèle entité-association 5

3.1. Introduction 5
3.2. Propriétés 5
3.3. Entités 6
3.4. Relations 7
3.5. Cardinalités 7
3.6. Règles d'usages 10
3.7. Exercices : Série de TD 03 10

4. Modèle Logique des données 11

4.1. Introduction 11
4.2. Passage du modèle conceptuel de données au modèle logique de données 11
4.3. Règle une 11
4.4. Règle deux : Relation 0,1 - 0,N ou 0,1 --- 1,N 12
4.5. Règle trois : Relation dont les cardinalités maximales sont supérieure à 1. 12
4.6. Associations n-aires 13
4.7. Associations réflexives 14
4.8. Règle quatre: cas particulier 14
4.9. Exercices : Série de TD 04 15

5. Normalisation d'une relation 16

5.1. Introduction 16
5.2. Première forme normale (1FN) 16
5.3. Deuxième forme normale (2FN) 16
5.4. Troisième forme normale (3FN) 17
5.5. Forme normale de Boyce-Codd (BCNF) 19
5.6. Quatrième forme normale (4FN) 20
5.7. Cinquième forme normale (5FN) 21
5.8. Quelques exemples 21
5.9. Exercices : Série de TD 05 24

6. Merise / UML 25

6.1. Introduction 25
6.2. L'héritage 26
6.3. La méthode MERISE 27
6.4. Le langage de modélisation UML 27
6.5. Analogie Merise/UML 31
6.6. Exercices : Série de TD 06 et 07 38

7. Conclusion 38

Présentation et étude des méthodes d'analyse

I

1. Objectifs

Ce chapitre s'articule autour de plusieurs objectifs clés dans le domaine de la modélisation et de la simulation des systèmes d'information. Nous explorerons les concepts et les techniques essentiels utilisés dans ce processus, en mettant particulièrement l'accent sur les normes de spécification, de conception et de documentation. En outre, nous examinerons en détail la méthode MERISE, en analysant ses objectifs spécifiques et en clarifiant ses différences par rapport à UML.

Ce chapitre vise à :

- Expliquer des concepts et des techniques utilisées pour la modélisation des système d'informations.
- Étudier les objectifs de la méthode MERISE.
- Extraire la différence entre la méthode MERISE et UML.
- Créer des modélisations des systèmes d'informations en utilisant les deux méthodes : MERSIE & UM

2. Introduction

L'analyse et la conception de systèmes informatiques sont des étapes cruciales dans le développement de logiciels. Elles visent à transformer des besoins fonctionnels en structures et processus informatiques clairs et efficaces. Deux méthodes largement utilisées pour atteindre cet objectif sont UML (Unified Modeling Language) et Merise.

UMLest un langage de modélisation standardisé et polyvalent, utilisé principalement pour spécifier, visualiser, construire et documenter les artefacts d'un système logiciel. Depuis son apparition à la fin des années 1990, UMLest devenu un outil essentiel pour les développeurs et les analystes, offrant une syntaxe graphique intuitive pour représenter divers aspects d'un système logiciel.

Merise, quant à elle, est une méthode d'analyse, de conception et de gestion de projet qui est particulièrement populaire en France et dans les pays francophones. Développée dans les années 1970 et 1980, Merise est structurée autour de trois phases principales : l'analyse des besoins, la conception des traitements et la gestion des données. Elle met l'accent sur une approche structurée et méthodique pour modéliser les informations et les traitements d'un système d'information.

Dans ce chapitre, nous explorerons en détail ces deux méthodes d'analyse. Nous examinerons leurs principes fondamentaux, leurs diagrammes de modélisation clés, ainsi que leurs applications pratiques dans le domaine du développement logiciel.

Enfin, nous aborderons les aspects essentiels de la transition entre UMLet Merise, soulignant comment ces méthodologies peuvent être intégrées pour optimiser la modélisation et la gestion des systèmes d'information complexes. À travers cette exploration, nous espérons offrir une compréhension approfondie et pratique des méthodes d'analyse UMLet Merise, afin d'équiper les professionnels du développement logiciel avec les outils nécessaires pour réussir dans leurs projets.

3. Le modèle entité-association

3.1. Introduction

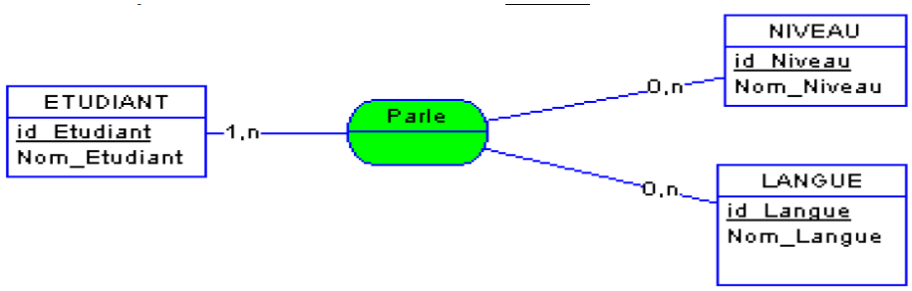
le modèle conceptuel de données MCD ou modèle entité-association E-A vise à fournir une base conceptuelle claire pour la modélisation des données dans le domaine des bases de données. Ce modèle est fondamental en informatique pour représenter les relations entre différentes entités au sein d'un système d'information. En utilisant des entités pour représenter des objets du monde réel (comme des personnes, des lieux ou des objets), et des associations pour représenter les relations entre ces entités, le modèle E-A offre une méthode visuelle et structurée pour organiser les données et définir leurs interactions.

En pratique, le modèle E-A est largement utilisé pour concevoir des bases de données relationnelles, où les entités deviennent des tables et les associations deviennent des relations entre ces tables. Cette approche permet de capturer de manière précise les relations complexes entre les données, tout en assurant une structure de données cohérente et normalisée.

En résumé, le modèle entité-association constitue un outil essentiel dans le développement de systèmes d'information, facilitant la représentation, l'analyse et la gestion des données à travers une approche intuitive et systématique.

Les éléments de base constituant un modèle E-A sont :

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



Exemple d'un modèle entité_association

3.2. Propriétés

La proprité est une information élémentaire conforme aux choix de gestion de l'entreprise, Les propriétés sont les informations de base du système d'information.

Exemples:

- référence d'un produit,
- adresse d'un client,
- quantité vendue d'un produit
- ,...

Les propriétés sont caractérisées par leur type, qui peut être numérique, une représentation de date, ou une longueur spécifique peut être définie. Par exemple, le nom est une propriété de type alphabétique avec une longueur maximale de 50 caractères.

3.3. Entités

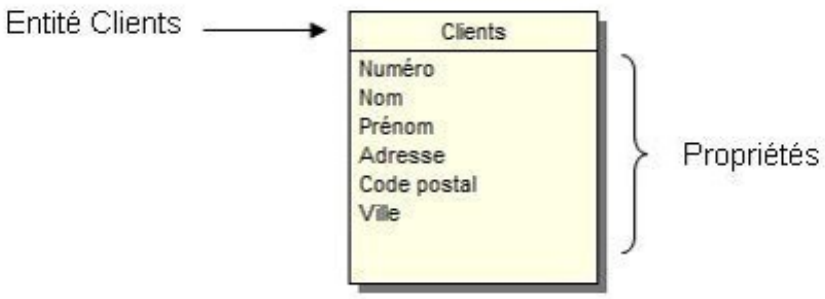
Une entité dans le modèle E-A représente un objet, une personne, un lieu ou un concept distinctif du monde réel, qui peut être identifiable et distingué par ses caractéristiques propres. Dans un contexte de modélisation de données, une entité est souvent associée à une table dans une base de données relationnelle. Chaque entité est caractérisée par des attributs qui décrivent ses propriétés et ses caractéristiques spécifiques.

L'entité joue un rôle central dans la modélisation des données car elle permet de structurer et d'organiser les informations relatives à des éléments spécifiques du domaine d'application. Par exemple, dans un système de gestion d'une bibliothèque, une entité pourrait être "Livre", avec des attributs tels que "Titre", "Auteur", "Date de publication", etc.

Exemple

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 :



Exemple de l'entité Client

L'identifiant

L'identifiant dans une entité du modèle entité-association (E-A) est un attribut spécial qui permet de distinguer de manière unique chaque occurrence de cette entité. Cet identifiant est souvent appelé "clé primaire" dans le contexte des bases de données relationnelles. Il sert à garantir que chaque ligne ou instance d'une entité dans la base de données est identifiable de manière exclusive.

Voici quelques points importants à propos de l'identifiant dans le modèle E-A :

1. **Unicité :** L'identifiant doit être unique pour chaque occurrence de l'entité. Cela signifie qu'aucune autre occurrence de la même entité ne peut avoir la même valeur pour son identifiant.
2. **Stabilité** : L'identifiant ne change généralement pas au cours de la durée de vie de l'entité. Il est stable et permet de référencer l'entité de manière constante.
3. **Utilisation :** L'identifiant est essentiel pour établir des relations entre différentes entités dans le modèle E-A. Par exemple, dans un système de gestion d'une bibliothèque, l'identifiant d'un livre pourrait être un numéro d'identification unique assigné à chaque livre.
4. **Clé primaire :** En bases de données relationnelles, l'identifiant est souvent défini comme la clé primaire de la table correspondante. Cela implique qu'il est utilisé pour indexer et accéder efficacement aux données de cette entité.

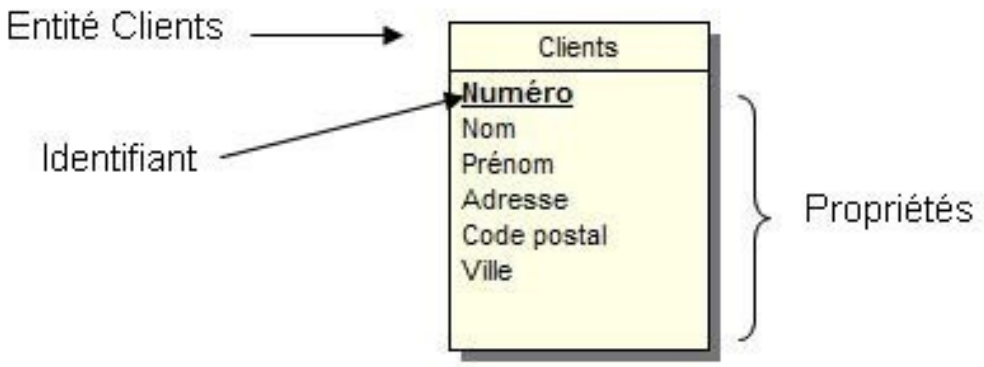
En résumé, dans le modèle E-A, l'identifiant d'une entité est un attribut crucial qui assure l'unicité et l'identification unique de chaque instance de cette entité au sein du système d'information.

Exemple

le fait de connaître la ville d'un client permet--il de connaître son nom ? La réponse est non. 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.

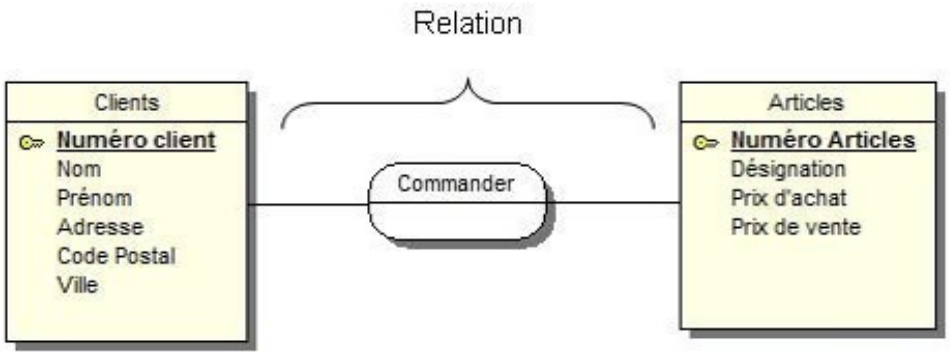
Au niveau du formalisme, cette propriété se souligne. Voici le schéma modifié de l'entité Clients.



Identifiant d'une entité

3.4. Relations

Une relation représente une association significative entre deux ou plusieurs entités. Elle exprime comment les entités sont connectées ou interagissent les unes avec les autres dans le contexte d'un système d'information.



Relation entre deux entité

3.5. Cardinalités

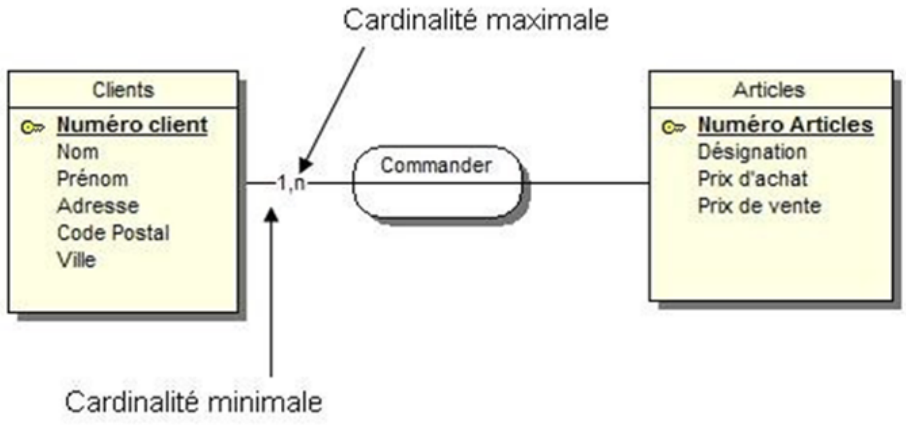
E -

Les cardinalités dans le modèle A définissent le nombre d'occurrences d'une entité qui peuvent être associées à un nombre donné d'occurrences d'une autre entité à travers une relation. Elles sont cruciales pour préciser la nature et la structure des associations entre les entités.

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.



Cardinalités dans une relation

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 interpréter cela comme suit :

Y a-t-il des articles qui ne peuvent jamais être commandés ? Si nous répondons oui, alors la cardinalité minimale est 0.

Pour le maximum : Y a-t-il des articles qui peuvent être commandés plusieurs fois ? Si nous répondons oui, alors la cardinalité maximale sera n.

Voici le schéma finalisé :

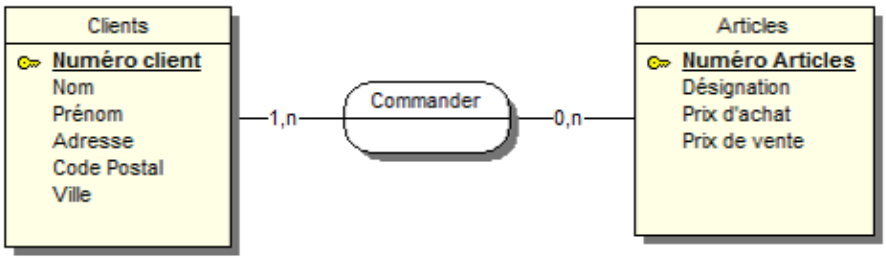


Schéma avec toutes les cardinalités

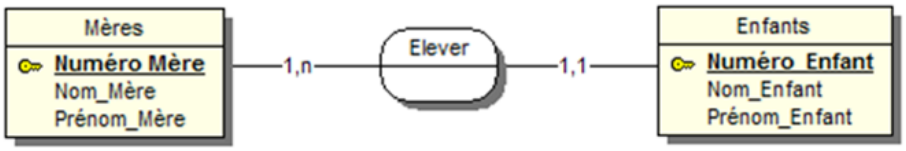
Définitions

1. La cardinalité minimale (0 ou 1) exprime le nombre de fois minimum qu'une occurrence d'une entité participe aux occurrences d'une relation.
2. 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 :



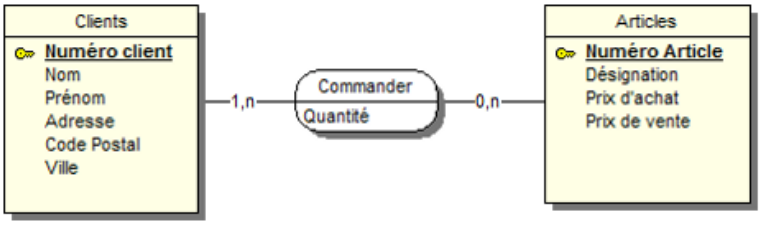
Exemple de modélisation

3.5.1. Les relations porteuses

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 :



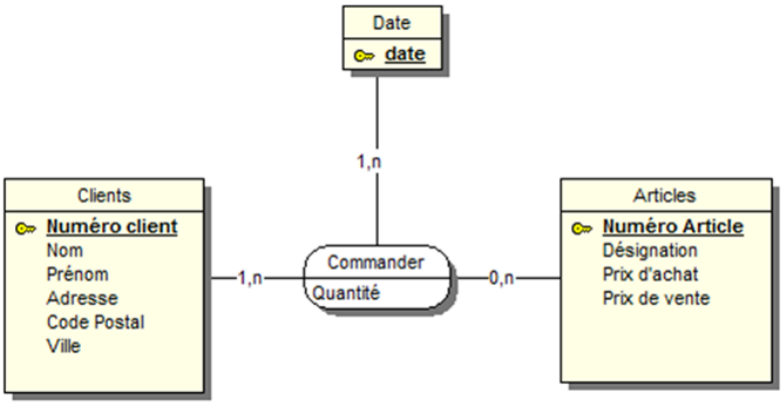
Relation porteuse

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** ».

Exemple :

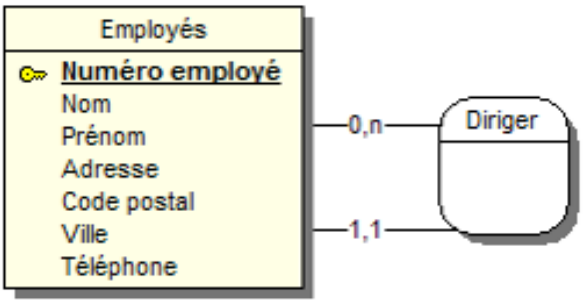


Exemple d'une relation ternaire

3.5.2. Relation réflexive

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.



Relation réflexive

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.

3.6. 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 Fournisseurs, nous ne devons pas retrouver la propriété Nom dans les deux entités. Il faut préférer la dénomination suivante Nom_client et Nom_fournisseur.
- Les propriétés résultantes d'un calcul ne doivent pas apparaître dans le modèle conceptuel des données.

3.7. Exercices : Série de TD 03

Ci-joint, vous trouverez la série de TD03 sur la modélisation des systèmes d'information à l'aide du modèle entité-association.

[cf. Série de TD03]

4. Modèle Logique des données

Le MCD possède plusieurs avantages comme : la clarté, la richesse, etc, mais ce formalisme ne peut pas être supporté par la machine. Ce qui nous oblige de passer par un autre modèle de données plus proche de la machine qui s'appelle modèle logique de données MLD.

4.1. Introduction

Le MLDreprésente la structure des données d'un système d'information sans se préoccuper des détails physiques de stockage. Il se situe entre le modèle conceptuel (qui décrit les entités, leurs attributs et leurs relations) et le modèle physique (qui spécifie comment les données sont concrètement stockées dans une base de données).

En utilisant des concepts tels que les tables, les colonnes et les relations, le modèle logique de données définit la façon dont les données seront organisées et interconnectées dans une base de données relationnelle. Il permet de traduire les concepts abstraits du modèle conceptuel en une structure de données concrète et exploitable.

Le MLD est lui aussi indépendant du matériel et du logiciel, il ne fait que prendre en compte l'organisation des données. Si l'organisation des données est relationnelle, alors le MLDest relationnel et devient le MLDR, ou Modèle Logique de Données Relationnelles.

Le MLDR a été inventé par Codd en 1970, et repose sur la théorie ensembliste, il consiste à décrire la structure de données utilisée sans faire référence à un langage de programmation. Il est dépendant du type de la base de données utilisée. (Relationnelle ou autre)

Nous traiterons dans cette partie du cours la formalisation du MLD appliquée à une base de données relationnelle, c'est-à-dire le MLDR

4.2. Passage du modèle conceptuel de données au modèle logique de données

Passer du modèle conceptuel de données au modèle logique de données implique de transformer la représentation abstraite des entités, attributs et relations en une structure plus concrète adaptée à une implémentation en base de données relationnelle.

4.3. Règle une

A- Les entités types du MCDsont converties en tables dans le MLD.

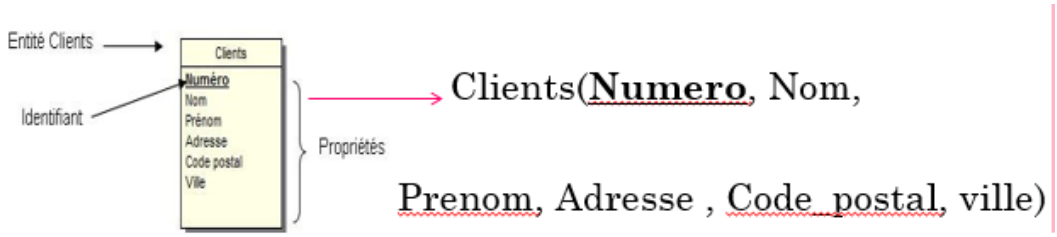
- une table est une structure tabulaire dont chaque ligne correspond à un enregistrement qui contient les données d'un objet enregistré et chaque colonne correspond à un attribut ou champ qui contient les propriétés de cet objet
- La valeur prise par un attribut pour un enregistrement donné est située à l'intersection ligne-colonne correspondant à enregistrement-attribut.
- la limite au nombre d'enregistrements est liée à l'espace de stockage.

B-L' identifiant devient la clé primaire de la relation.

- La clé primaire permet d'identifier de façon unique un enregistrement dans une table, sa valeur est unique obligatoirement non nulle.
- Dans la plupart des système de gestion de base de données relationnelles, le fait de définir une clé primaire donne lieu automatiquement à la création d'un index.

C- les propriétés deviennent des attributs de la tables (colonnes)

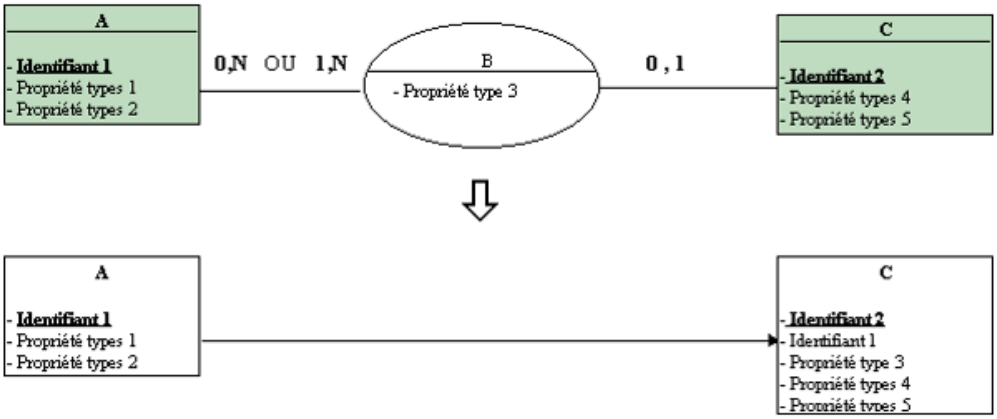
Exemple :



Passage du MCD au MLD

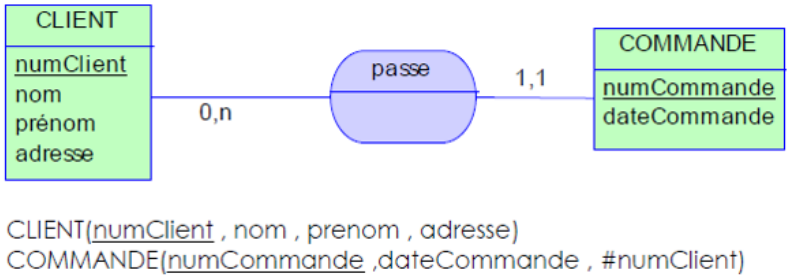
4.4. Règle deux : Relation 0,1 - 0,N ou 0,1 --- 1,N

Une association de type **1:N** est une association qui a les cardinalités maximales positionnées à «N» d'un côté de l'association et à « 1 » de l'autre côté. L'association est supprimée et ses propriétés types deviennent des rubriques de la table issue de l'entité qui a la cardinalité maximale **1**.



Règle deux

Exemple :

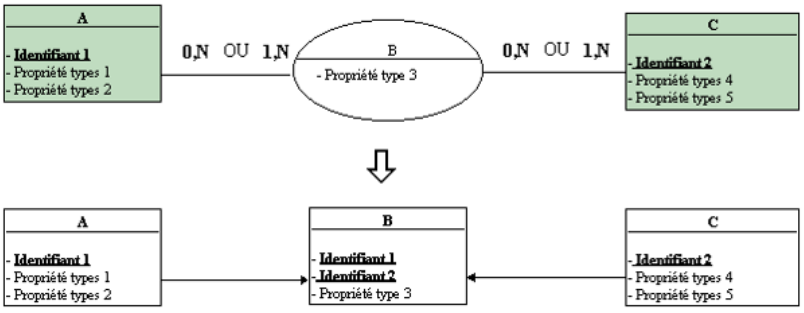


Exemple du passage de MCD au MLD

4.5. Règle trois : Relation dont les cardinalités maximales sont supérieure à 1.

Une association de type **N:N** (appelée aussi association père-père) est une association qui a les cardinalités maximales positionnées à «N» des 2 côtés de l'association). Elle se traduit par la création d'une **nouvelle** table dont la clé primaire est composée des clés étrangères référençant les relations correspondant aux entités liées par l'association.

Les éventuelles propriétés de l'association deviennent des attributs de la nouvelle table.



Règle trois du passage de MCD au MLD

Exemple : traduisez le MLD suivant en MLDR

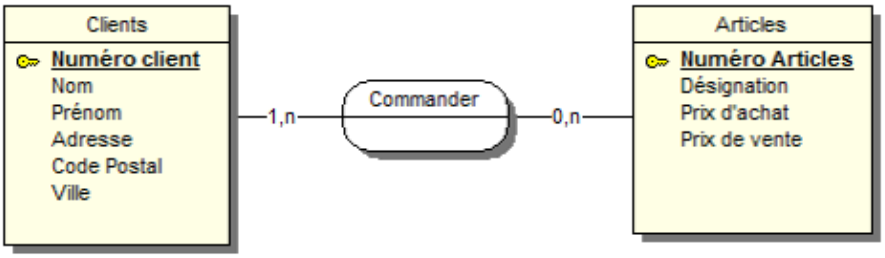


Schéma avec toutes les cardinalités

Le MLDR :

Clients(**Numero_client,** nom, prenom, adresse, code_postal, ville)

Articles(**Numero_article**, designation, prix d'achat, prix de vente)

Commander(**#Numero_client, #Numero_client)**

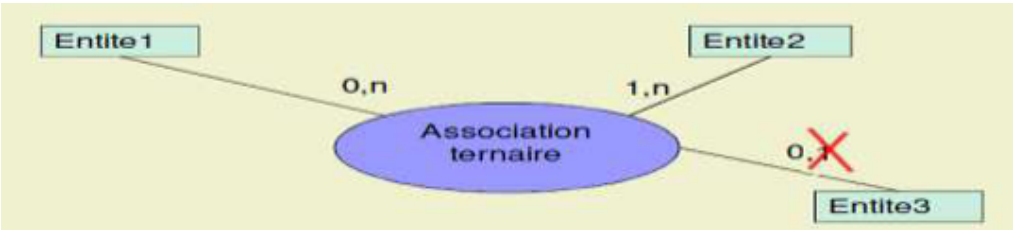
4.6. Associations n-aires

Les règles définies ci-dessus s'appliquent aux associations n-aires (n >=3) .

dans ce type d'association :

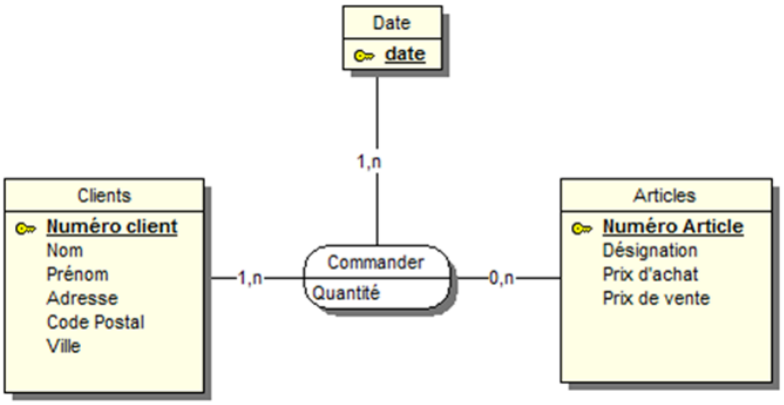
- Les cardinalités minimales peuvent être à 0 ou.
- Toutes les cardinalités maximales sont obligatoirement à n.

Si l'une des cardinalités maximales n'est pas à n, il y a une erreur de conception.



Association n-aires

Exemple : traduisez le MLD suivant en MLDR



Exemple d'une relation ternaire

Le MLDR :

Clients(**Numero_client**, nom, prenom, adresse, code_postal, ville)

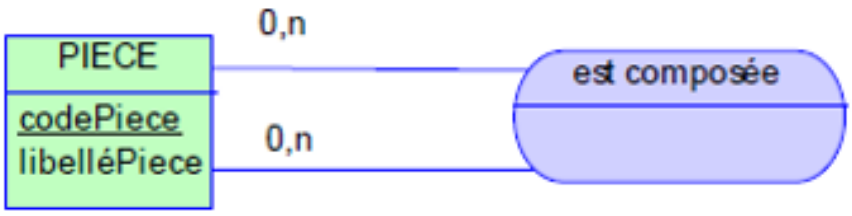
Articles(**Numero_article**, designation, prix d'achat, prix de vente)

Date(**date**)

Commander(**#Numero_client, #Numero_client, # date**, quantite)

4.7. Associations réflexives

Pour une association réflexive voila un exemple de traduction du MCD au MLDR



Association réflexive

Le MLDR :

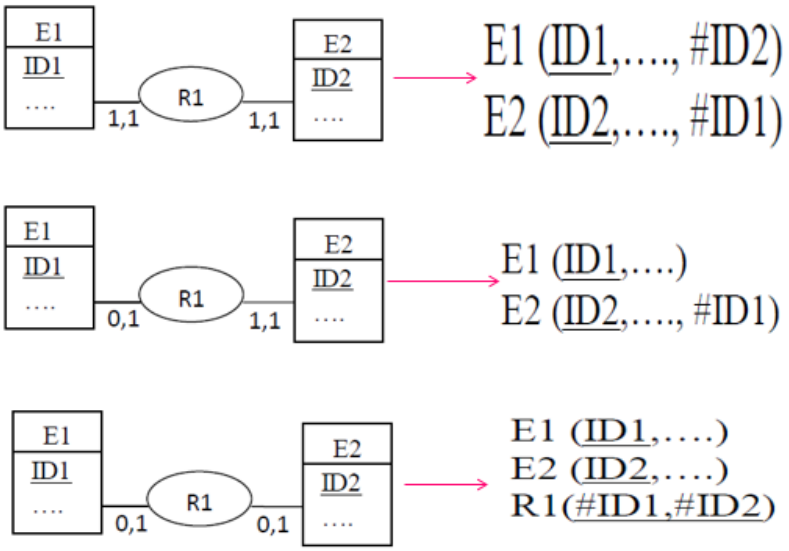
PIECE(**codePiece** ,libellePiece)

COMPOSITION(**#codepieceComposee , #codepieceComposante**)

4.8. Règle quatre: cas particulier

Une association de type 1:1 est une association qui a les cardinalités maximales positionnées à «1» des 2 côtés de l'association).

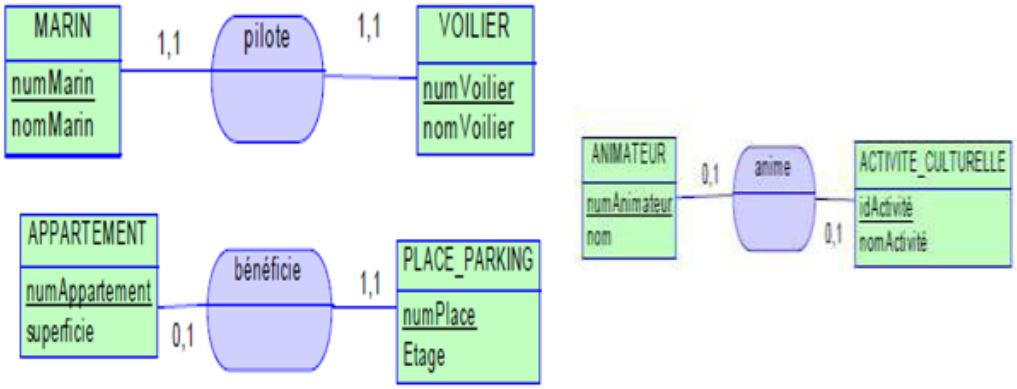
On distingue 3 cas possibles :



Règle 04

Exemple : Exemple des cas particuliers

Voici quelques cas particulier qui ont les cardinalités maximales 1 dans les deux cotés :



Exemple de cas particulier

4.9. Exercices : Série de TD 04

Ci-joint la série d'exercices TD04 comprenant des exercices sur la transition du Modèle Conceptuel de Données vers le Modèle Logique de Données Relationnelles.

[cf. Série TD04]

5. Normalisation d'une relation

5.1. Introduction

Pour être parfaites, les relations doivent respecter certaines règles. Cet ensemble de règles se nomme : les Formes Normales FN.

Cette théorie a été élaborée par 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.

Il existe 5 FN principales. Plus le niveau de normalisation est élevé, plus le modèle est exempte de redondances.

Pour parfaire une base de données relationnelle, il est nécessaire de connaître les trois premières FN et la forme normale dite Boyce-Codd ; les autres FN ne sont que des extensions peu utilisée

5.2. Première forme normale (1FN)

Une relation est en 1FN si tout attribut contient une valeur atomique (non divisible).

Exemple : Exemple 1FN

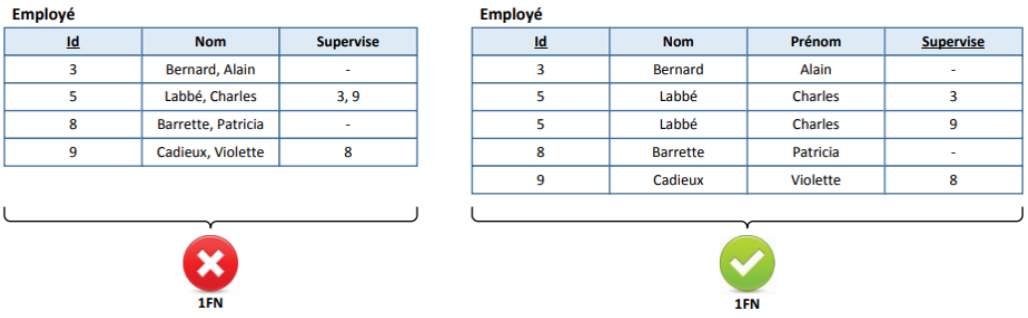
Clients (**NumCli**, Nom, Prénom, Adresse, Téléphone)

Cette relation n'est pas en première forme normale, car Adresse n'est pas atomique.

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)



Exemple 1FN

5.3. Deuxième forme normale (2FN)

Une relation est en 2FN Si :

- Elle est en 1FN.
- tous les attributs non clés sont complètement dépendants de la clé primaire.

Exemple : Exemple 2FN

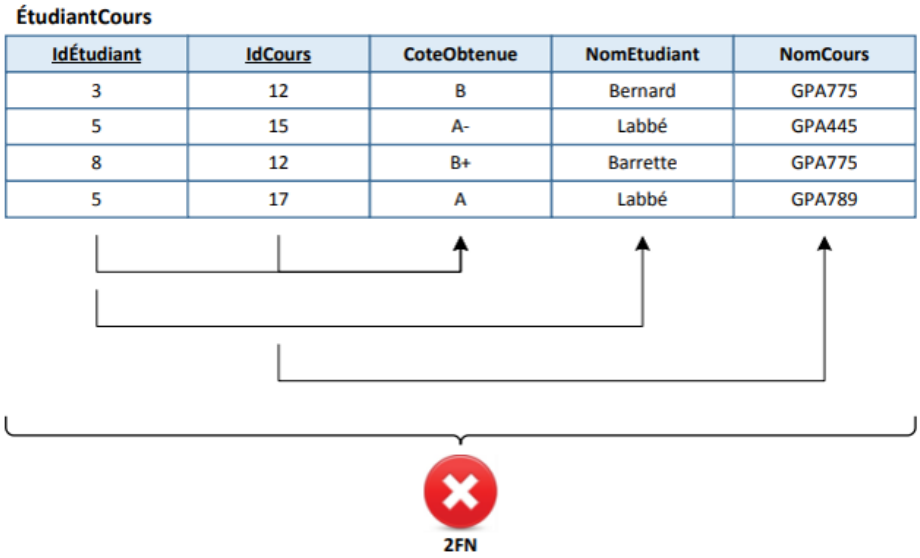
Commande (Numcli, CodeArticle, Date, Qté commandée, Désignation)

Cette relation est -elle en 1FN ? Oui.

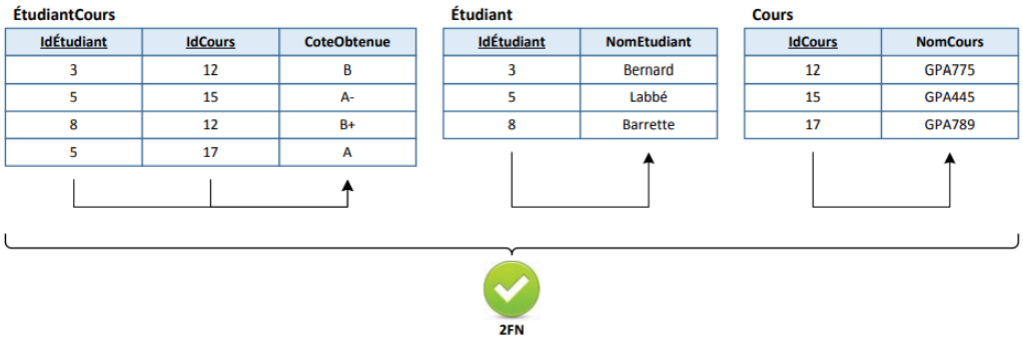
Est -elle en deuxième forme normale ? Non, car la propriété Désignation ne dépend pas intégralement de la clé (Numcli, CodeArticle, Date).

Voici comment corriger :

- Commandes(Numcli, CodeArticle, Date, Qté commandée)
- Articles(CodeArticle, Désignation)



Exemple 2FN



Solution exemple 2FN

5.4. Troisième forme normale (3FN)

Une relation est en 3FN Si :

- Elle est en 2FN.
- Tout attribut n'appartenant pas à une clé ne dépend pas d'un autre attribut non clé.

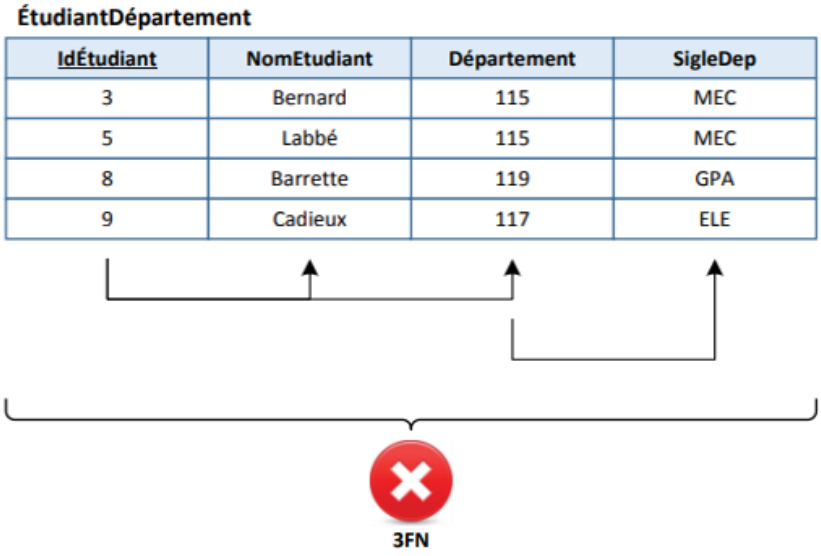
Exemple : Exemple1 3FN

Véhicule (**NV**, Type, Marque, Puissance, Couleur) cette relation n'est pas en 3FN car :

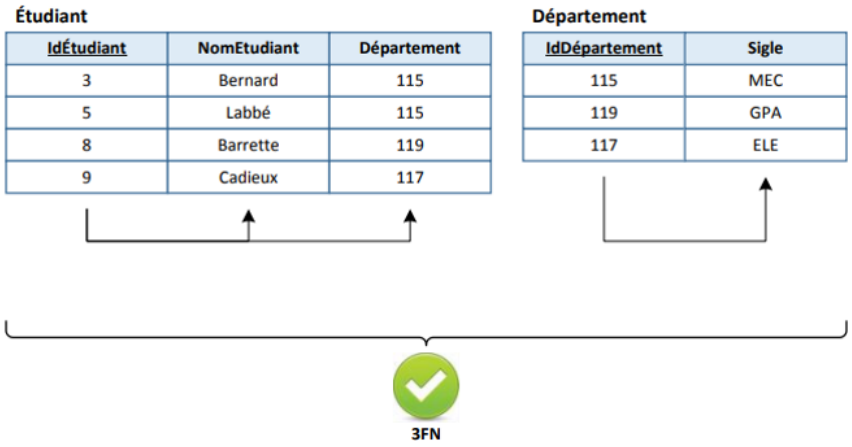
L'attribut non clé Type → Marque et Type → Puissance.

Cette relation peut être décomposée en deux relations

- Véhicule (NV, Type, Couleur)
- Modèle (Type, Marque, Puissance)



Exemple 3FN



Solution exemple 3FN

Exemple : Exemple2 3FN

La relation Commande(**NuméroCommande**, #CodeClient, Nom client, #RefArticle)

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

Voici comment corriger :

Commande(NuméroCommande, #CodeClient, #RefArticle)

Clients(CodeClient, Nom client)

5.5. Forme normale de Boyce-Codd (BCNF)

Boyce et Codd ont introduit une forme normale qui porte leur nom (Boyce Codd Normal Form/BCNF) :

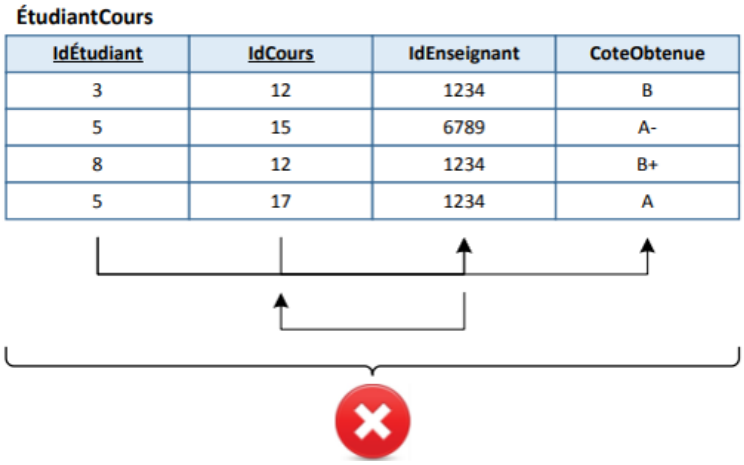
Une relation est en forme normale de Boyce-Codd Si :

1 - Elle est en 3FN.

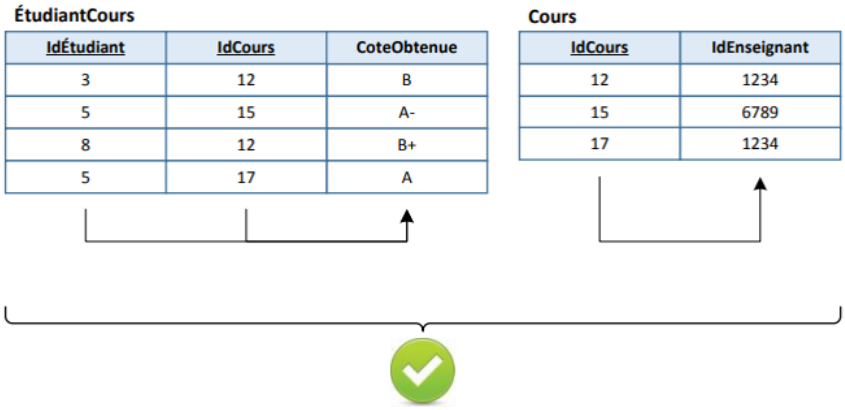
2 - les seules dépendances fonctionnelles élémentaires sont celles dans lesquelles une clé détermine un attribut. (Aucun attribut faisant partie de la clé dépend d'un attribut ne faisant pas partie de la clé)

Exemple : Exemple BCNF

Voici un exemple de la forme Boyce Codd



Exemple de la forme Boyce Codd



5.6. Quatrième forme normale (4FN)

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 dépendance à valeurs multiples existe toujours si deux attributs non liés dépendent du même attribut,

Exemple : Exemple 4FN

Le tableau suivant indique quels produits ont été commandés par client et à quel code postal ils doivent être livrés.

Commandes des clients par code postal		
N° Client	**N° Produit**	**Code Postal**
234	1-0023-D	12345
234	2-0023-D	12345
567	1-0023-D	56789
567	3-0023-D	56789
567	4-0023-D	56789
567	5-0023-D	56789

Exemple 4FN

Par exemple, le client portant le numéro de client 234 a commandé les articles 1-0023-D et 2-0023-D qui doivent être livrés à son adresse au code postal 12345. Pour le client 567, les articles 1-0023-D, 3-0023-D, 4-0023-D, 4-0023-D et 5-0023-D seront livrés sous le code postal 56789.

l'attribut numéro de produit et l'attribut code postal dépendent tous deux de l'attribut numéro client, mais ne sont pas liés l'un à l'autre.

L'inconvénient d'une telle conception de base de données est que chaque fois qu'un nouveau produit est ajouté à l'enregistrement du client, le code postal doit également être ajouté, ce qui entraîne des données redondantes.

Ces redondances peuvent être éliminées en convertissant le tableau en 4FN. Pour ce faire, vous devez diviser la table

Produit	
N° Client	N° Produit
234	1-0023-D
234	2-0023-D
567	1-0023-D
567	4-0023-D
567	5-0023-D

Lieu de livraison	
N° Client	Code Postal
234	12345
567	56789

Solution exemple 4FN

5.7. Cinquième forme normale (5FN)

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

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

Elle est en quatrième forme normale.

Le tableau ne peut pas être divisé davantage sans perdre des informations.

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.

la jointure est l'opération permettant d'associer plusieurs tables ou vues de la base par le biais d'un lien logique de données entre les différentes tables ou vues

Remarque

- Ces formes normales évitent principalement la redondance d'information, elles sont plus précises.
- En pratique la 3FN est suffisante. En effet, les autres formes sont coûteuses pour le système, ainsi une trop forte normalisation diminue fortement les performances

5.8. Quelques exemples

Voici un exemple illustrant les différentes formes normales.

Exemple : Exemple de normalisation

Voici une table non normalisée

Fact. N°.	Date	Client	N° Client	Adresse	N° inventaire	Produit	N°.Produit	No.	Prix
123	29/01/2018	John Public	11	35 Wood Lane, Springfield, ME 04487	1	Monitor	2-0023-D	10	$200
123	29/01/2018	John Public	11	35 Wood Lane, Springfield, ME 04487	2	Tapis de souris	4-0023-D	12	50¢
123	29/01/2018	John Public	11	35 Wood Lane, Springfield, ME 04487	3	Chaise	5-0023-D	1	$120
124	30/01/2018	Jane Doe	12	72 Windy Road, Springfield, ME 04487	1	Ordinateur	1-0023-D	2	$1,200
124	30/01/2018	Jane Doe	12	72 Windy Road, Springfield, ME 04487	2	Casque	3-0023-D	2	$75

Table non normalisée

Table normalisée avec 1FN

Fact. N°.	Date	Prénom	Nom	N°. Client	Adresse Postale	Ville	Etat	Code Postal	N° inventaire	Produit	N°.Produit	No.	Prix en €
123	01/29/2018	John	Public	11	35 Wood Lane	Springfield	ME	04487	1	Monitor	2-0023-D	10	200
123	01/29/2018	John	Public	11	35 Wood Lane	Springfield	ME	04487	2	Tapis de souris	4-0023-D	12	0.50
123	01/29/2018	John	Public	11	35 Wood Lane	Springfield	ME	04487	3	Chaise	5-0023-D	1	120
124	01/30/2018	Jane	Doe	12	72 Windy Road	Springfield	ME	04487	1	Ordinateur	1-0023-D	2	1200
124	01/30/2018	Jane	Doe	12	72 Windy Road	Springfield	ME	04487	2	Casque	3-0023-D	2	75

Table normalisée avec 1FN

Table normalisée avec 2FN

facture								
Fact. N°.	Date	Prénom	Nom	N° Client	Adresse Postale	ville	Etat	Code postal
123	29/01/2018	John	Public	11	35 Wood Lane	Springfield	ME	04487
124	30/01/2018	Jane	Doe	12	72 Windy Road	Springfield	ME	04487

Poste de facture					
Fact. N°.	N°. inventaire	Produit	N° Produit	No.	Prix en €
123	1	Monitor	2-0023-D	10	200
123	2	Tapis de souris	4-0023-D	12	0.50
123	3	Chaise	5-0023-D	1	120
124	1	Ordinateur	1-0023-D	2	1200
124	2	Casque	3-0023-D	2	75

Table normalisée avec 2FN

Table normalisée avec 3FN

Dans le tableau "Facture", le nom et le prénom, l'adresse, la ville, l'état et le code postal dépendent non seulement de la clé primaire (le numéro de facture), mais aussi du numéro du client.

Facture								
Fact. N°	**Date**	**Prénom**	**Nom**	**N° Client**	**Adresse Postale**	**Ville**	**Etat**	**Code postal**
123	29/01/2018	John	Public	11	35 Wood Lane	Springfield	ME	04487
124	29/01/2018	Jane	Doe	12	72 Windy Road	Springfield	ME	04487

Dans le tableau "Poste de facture", les attributs produits et prix dépendent non seulement de la clé primaire, dérivée du numéro de facture et du numéro de poste de facture, mais aussi du numéro de produit. Cette condition spécifique viole également la troisième forme normale.

Poste de Facture					
Fact. N°	**N° inventaire**	**Produit**	**N° Produit**	**No.**	**Prix en €**
123	1	Monitor	2-0023-D	10	200
123	2	Mousepad	4-0023-D	12	0.50
123	3	Chair	5-0023-D	1	120
124	1	Laptop	1-0023-D	2	1200
124	2	Headset	3-0023-D	2	75

Pour supprimer toutes les dépendances entre les attributs non clés, les attributs pertinents ont été déplacés dans des tables séparées, reliées entre elles par des clés étrangères. Il en résulte les quatre tableaux normalisés : "Facture", "Client", "Poste de Facture" et "Produit".

Facture

Fact. N°	Date	N° Client
123	01/29/2018	11
124	01/30/2018	12

Client

N° Client	Prénom	Nom	Adresse Postale	Ville	Etat	Code Postal
11	John	Public	35 Wood Lane	Springfield	ME	04487
12	Jane	Doe	72 Windy Road	Springfield	ME	04487

Poste de facture

Fact. N°	N° Inventaire	N° Produit	No.
123	1	2-0023-D	10
123	2	4-0023-D	12
123	3	5-0023-D	1
124	1	1-0023-D	2
124	2	3-0023-D	2

Produits

N° Produit	Produit	Prix en €
1-0023-D	Ordinateur	1200
2-0023-D	Monitor	200
3-0023-D	Casque	75
4-0023-D	Tapis de souris	0.50
5-0023-D	Chaise	120

Normalisation de l'exmple

5.9. Exercices : Série de TD 05

Voici une série d'exercices concernant la normalisation des relations.

[cf. Série de TD 05]

6. Merise / UML

6.1. Introduction

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

Exemple :

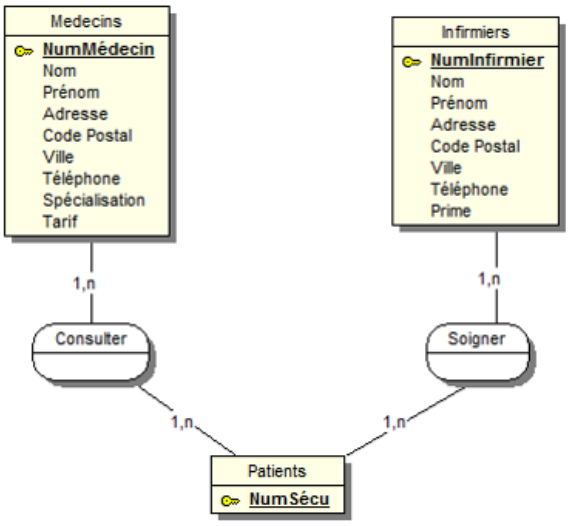
Une maison de santé reçoit des patients, deux types de personnel sont salariés : les médecins qui réalisent les consultations et les infirmiers administrent les soins.

Les médecins ont une spécialisation (médecin du sport, gynécologue,...) et un tarif à l'acte.

Les infirmiers ont des primes.

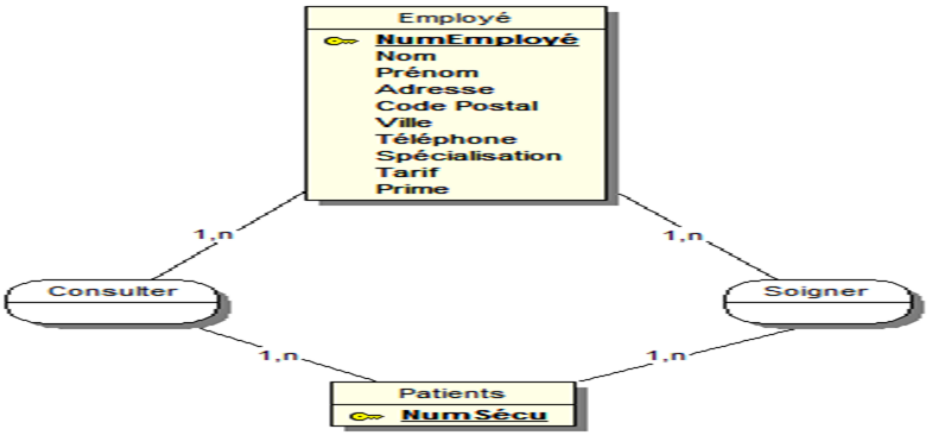
Les patients sont juste référencés par leur numéro de sécurité sociale.

Imaginons le MCDsuivant:



MCD d'une maison de santé

Une autre représentation du MCD:



Une autre représentation du MCD

Ces deux solutions, correctes, 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 (elle concerne les médecins).
- Prime pour les médecins.

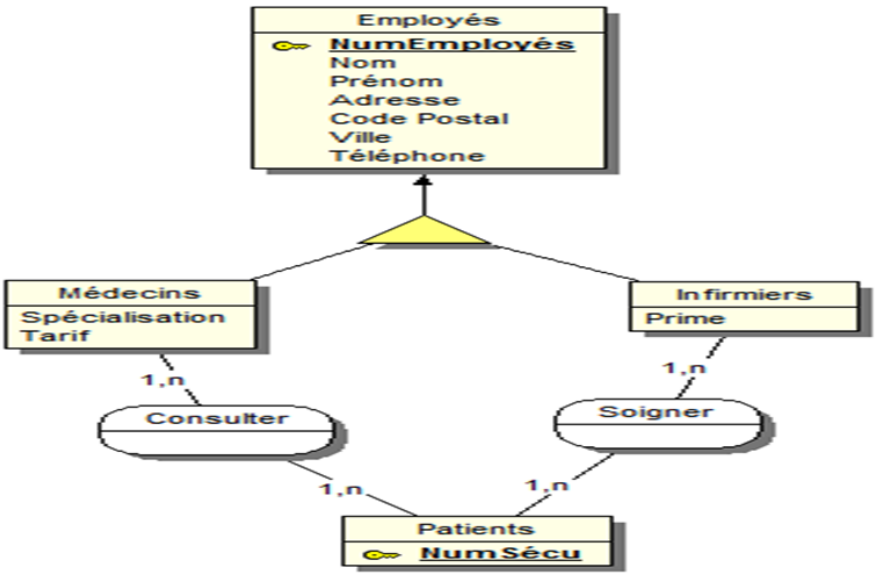
Pour résoudre ces problèmes, nous allons utiliser **l'héritage.**

6.2. 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.



Héritage

LeMLDR de cet exemple de l'héritage :

- Employé(**NumEmployes,** Nom, Prenom, Adresse, CodePostal, Ville, Telephone)
- Medecins(**#NumEmployes**, Specialisation, Tarif)
- Infirmiers(**#NumEmployes**, Prime)
- Patient(**NumSecu**)
- Consulter(**#NumEmployes, #NumSecu**)
- Soigner(**#NumEmployes, #NumSecu**)

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.

6.3. La méthode MERISE

Merise repose sur quelques principes fondamentaux :

Merise est une méthode d'analyse

- L'analyse est le moyen permettant de faire évoluer les systèmes d'information (mise en évidence de nouveaux besoins en informations, amélioration des procédures, des traitements...).
- L'analyse du SI repose sur la modélisation. Un modèle est une représentation simplifiée de la réalité : on ne retient que les éléments utiles et nécessaires au système d'information.
- Il existe plusieurs méthodes de modélisation: Merise, UML...
- MERISE permettait d'analyser les données et les traitements selon le MCD, MLD, MPD: consiste à implémenter le modèle dans le système de gestion de base de données, c'est-à-dire le traduire dans un langage de définition de données.)

La séparation des données et des traitements

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 qui met en lumière les traitements effectués sur les données c'est-à-dire les opérations qui sont réalisées en fonction d'événements

Une approche qui part du général vers le particulier

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.

6.4. Le langage de modélisation UML

UML « langage de modélisation unifié ») est un métalangage de modélisation, il a été normalisé en 1997 par l'OMG (Object Management Group). Son but est de formaliser les concepts orientés objet au travers des diagrammes.

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.

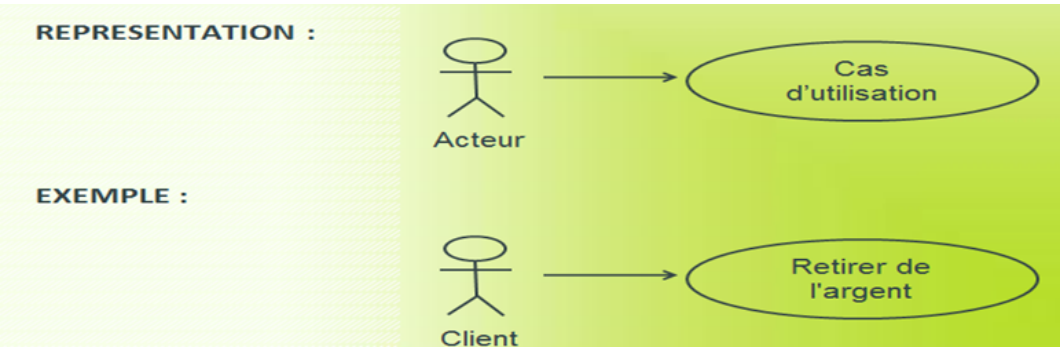


Diagramme de cas d'utilisation

- **Le diagramme de classes** est un ensemble d'éléments qui montre la structure du modèle étudié.

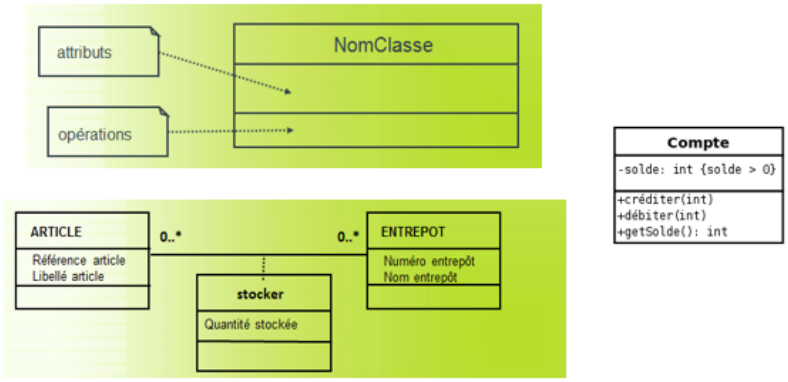


Diagramme de classe

- **Le diagramme d'objets** (objet : instance d'une classe) représente les objets et leurs interdépendances.

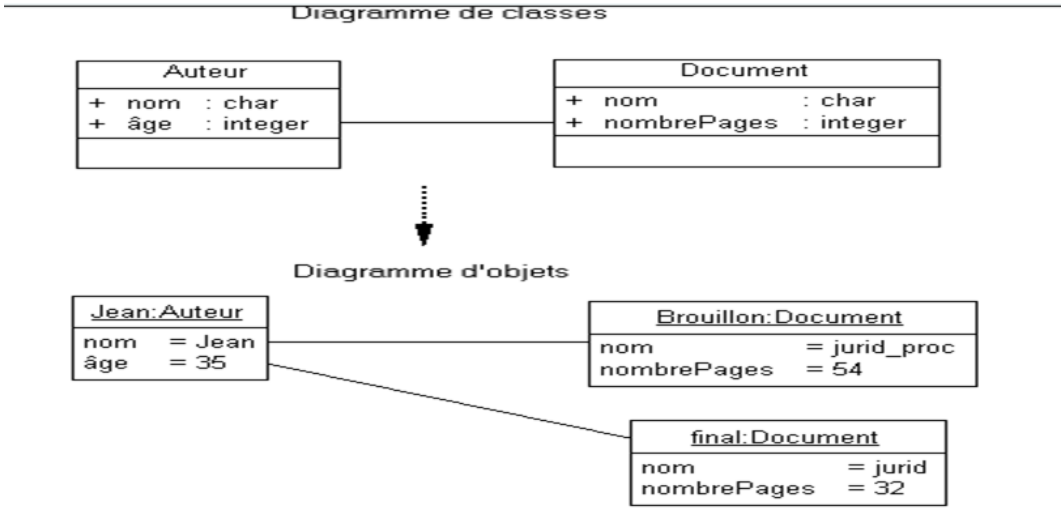


Diagramme d'objet

- **Le diagramme d'états/transitions** représente le cycle de vie des objets générés par une classe.

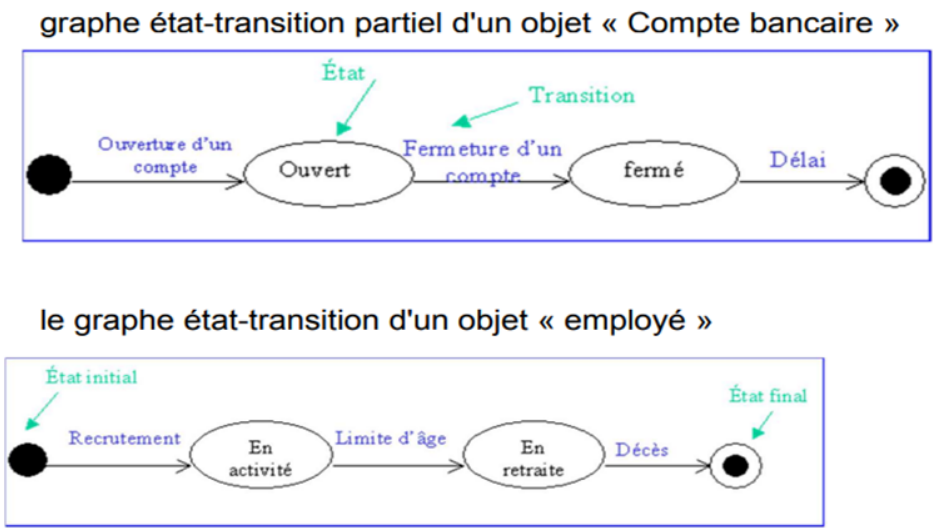


Diagramme d'état/transition

- **Le diagramme de composants** détaille les éléments logiciels (exécutables, fichiers...) et leurs dépendances.

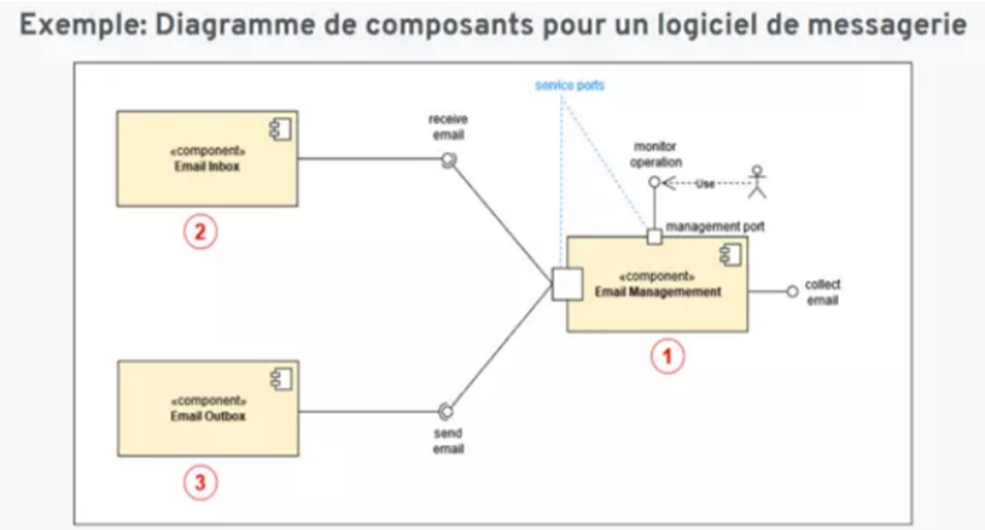


Diagramme de composants

- **Le diagramme de déploiement** montre la répartition physique des éléments matériels du système (processeurs, périphériques) et leurs connexions.

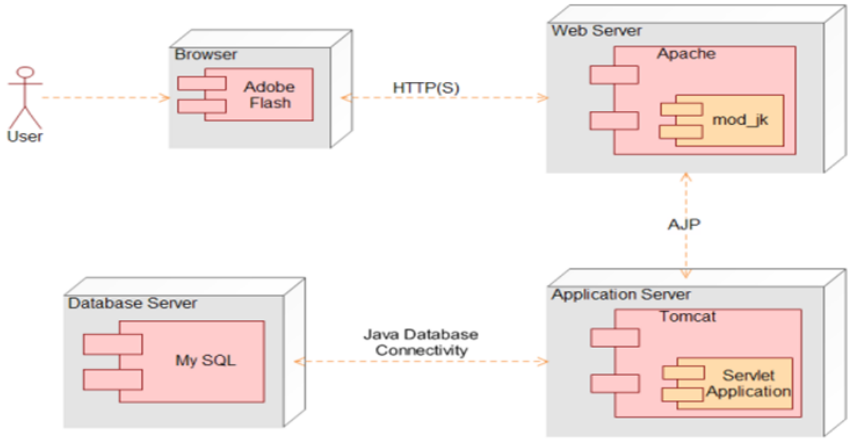


Diagramme de déploiement

- **Le diagramme de séquence** détaille les messages échangés entre les acteurs et le système selon un ordre chronologique.

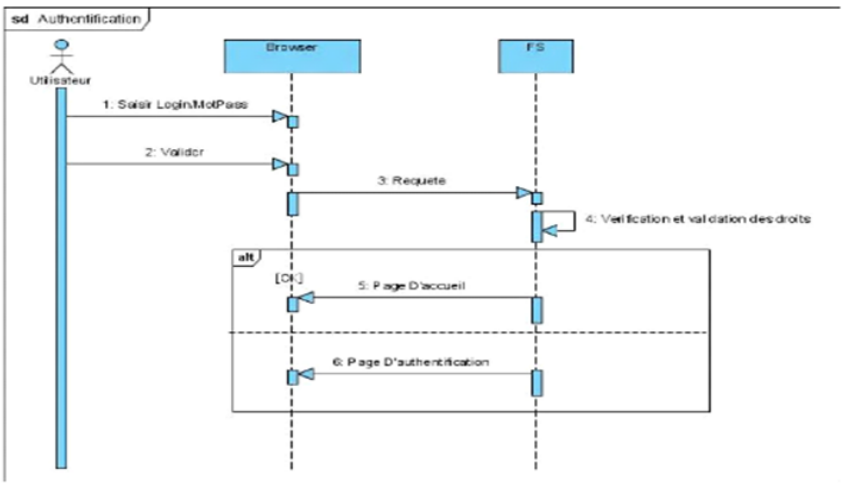


Diagramme de séquence

- **Le diagramme de collaboration** qui représente les messages échangés entre les objets.

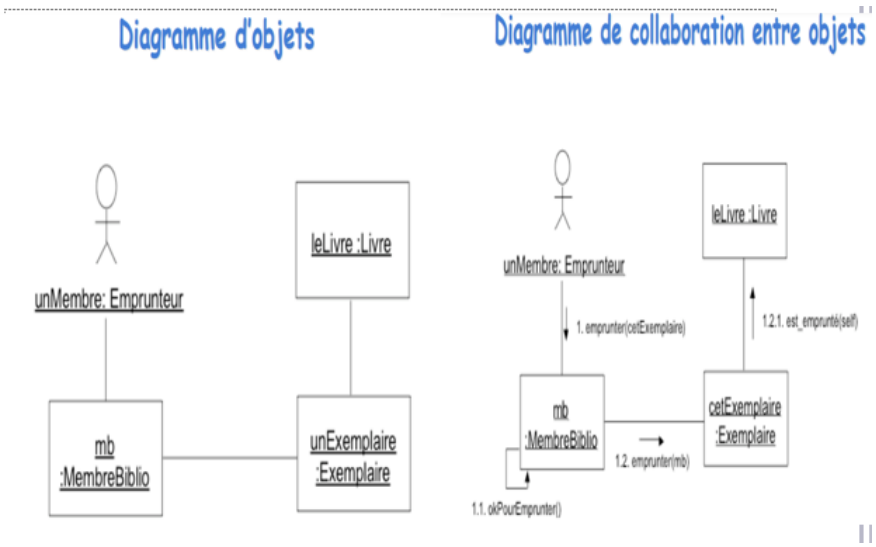


Diagramme de collaboration

- **Le diagramme d'activités** est une variante du diagramme états/transition qui représente le déclenchement d'évènements selon certains états du système.

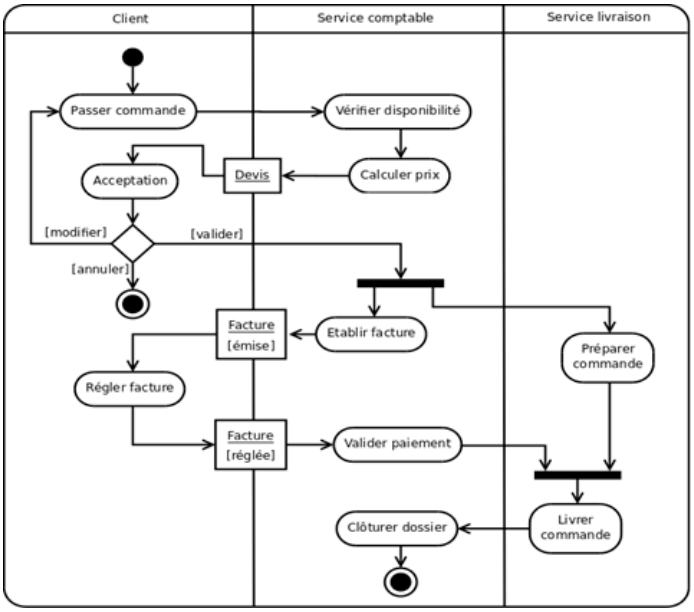


Diagramme d'activités

6.5. 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 ca 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
- 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 : Location des véhicules

Diagramme des flux MERISE

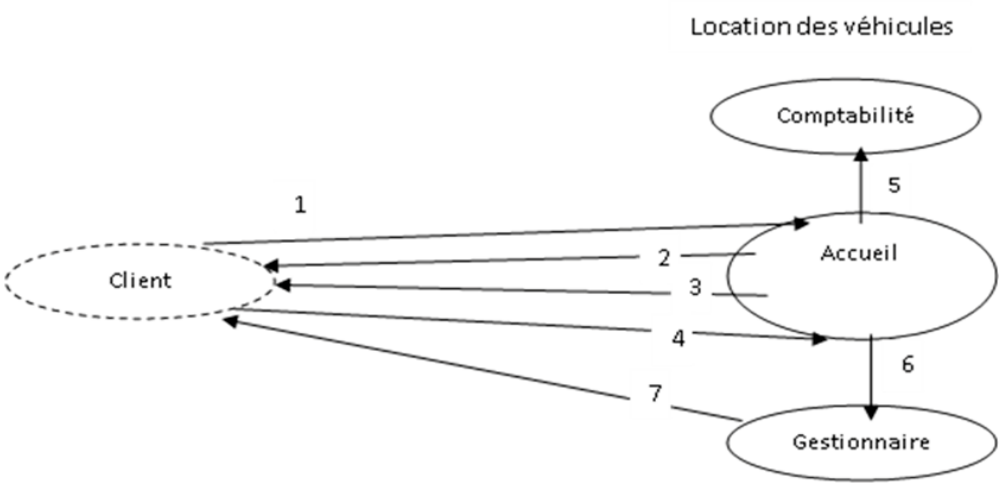


Diagramme de flux

N°	Description
1	Demande de location
2	Acceptation ou refus de la location en fonction du stock
3	L'édition de la facture
4	Paiement de la facture par le client
5	Passage de la facture et du paiement au service comptabilité.
6	Transmission de la demande au gestionnaire du parc
7	Remise du véhicule au client par le gestionnaire

Diagramme des cas d'utilisation UML

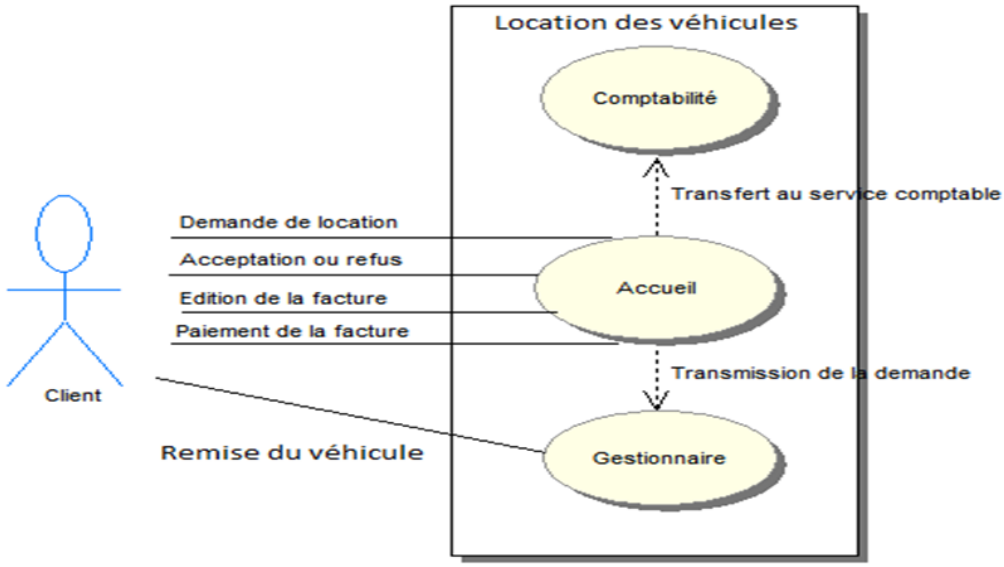


Diagramme de cas d'utilisation

Ces diagrammes montrent le passage de la méthode MERISE à la modélisation en utilisant UML

2-Modèle Conceptuel des Données/Diagramme de classes

Le MCD et le diagramme de classes partagent beaucoup de points communs. 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, MLDapporte plus de sécurité, à ce niveau là, qu'UML.

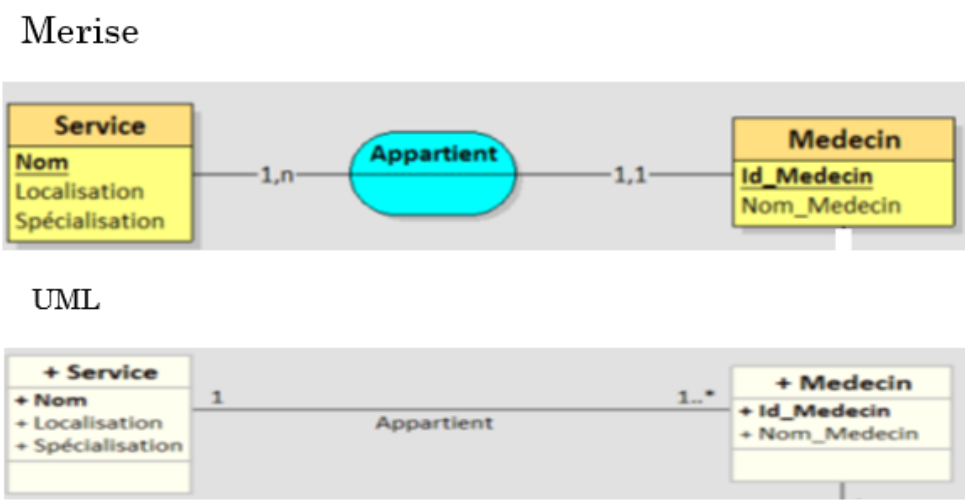
Voici quelques éléments de comparaison entre Merise et UML.

- Entité →Classe
- Attributs →Propriétés

- Identifiant →Propriété éventuellement en clé primaire
- 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.

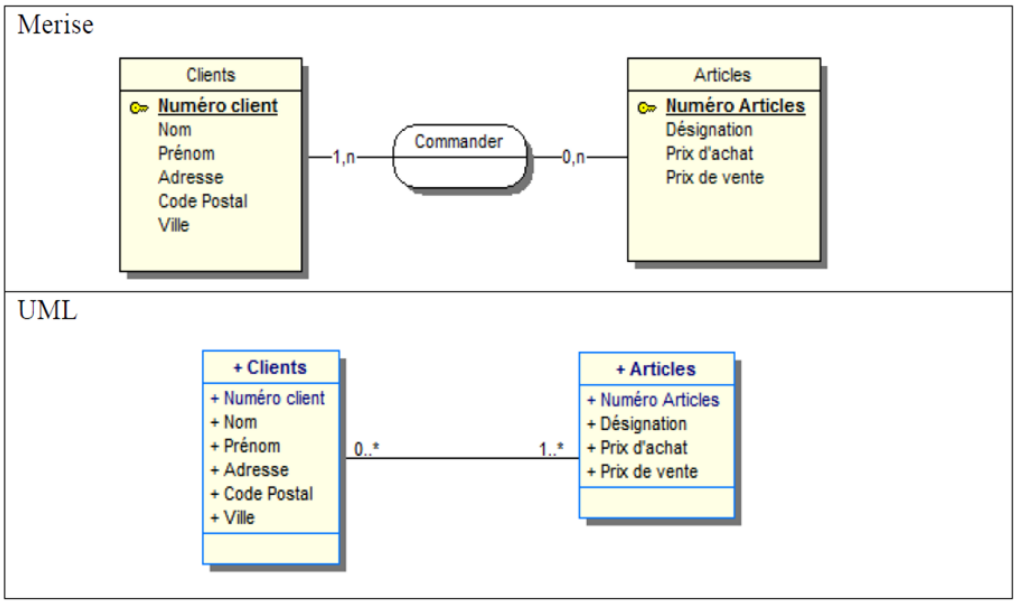
Merise	UML
0,1	0..1
1,1	1
0,n	0..* ou *
1,n	1..*

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



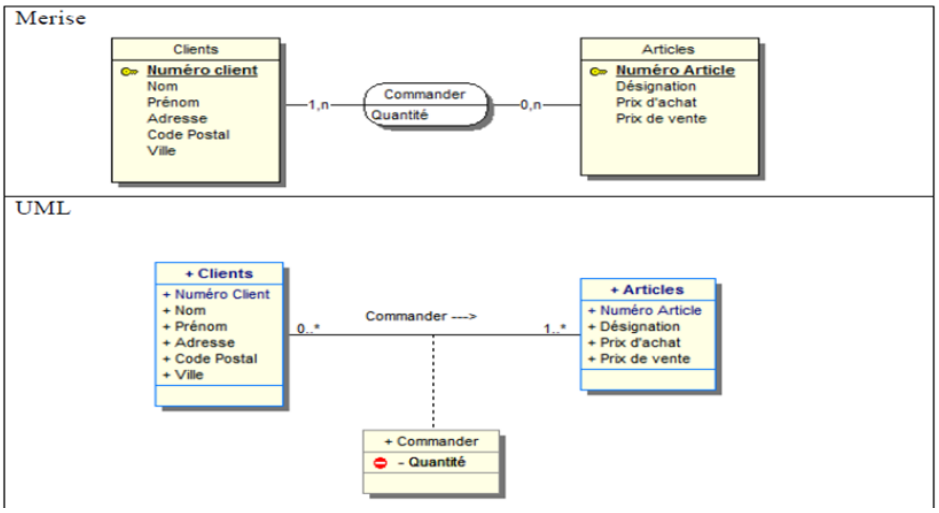
MERISE/UML

- **Association non porteuse**



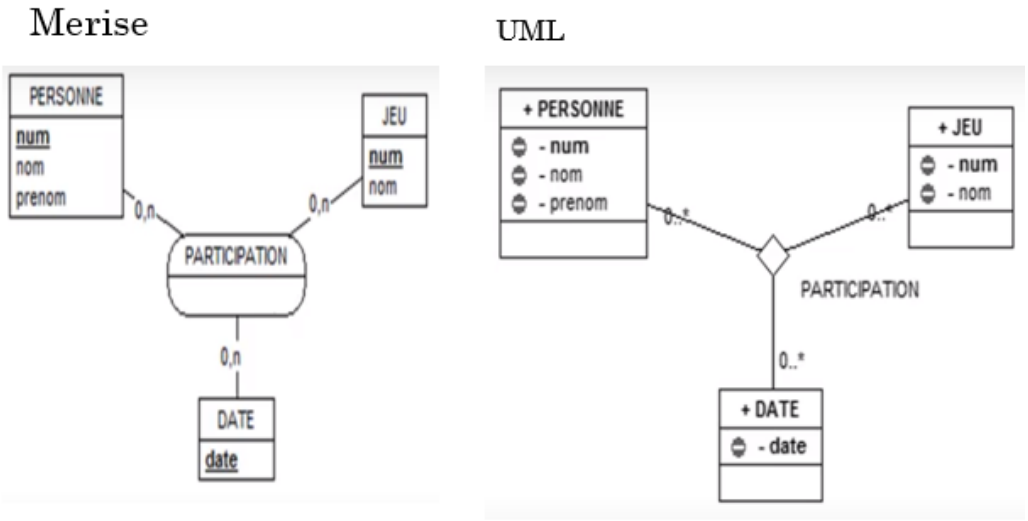
Merise/UML : Association non porteuse

- **Association porteuse**



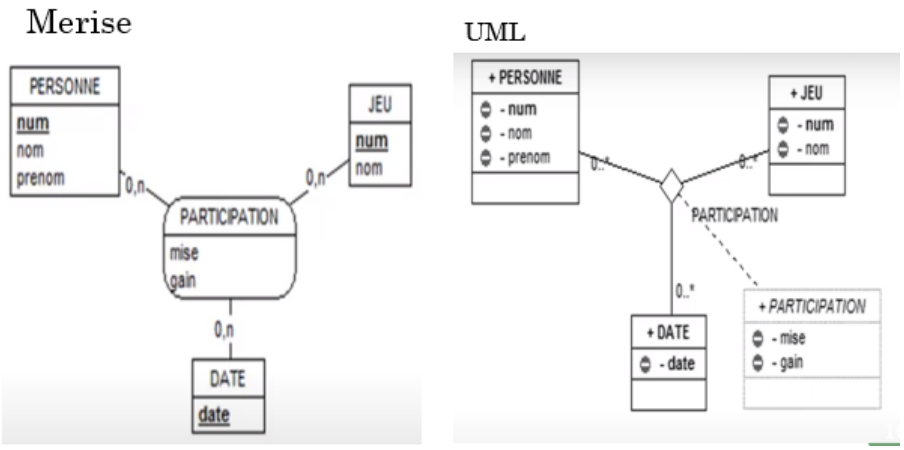
MERISE/UML : Association porteuse

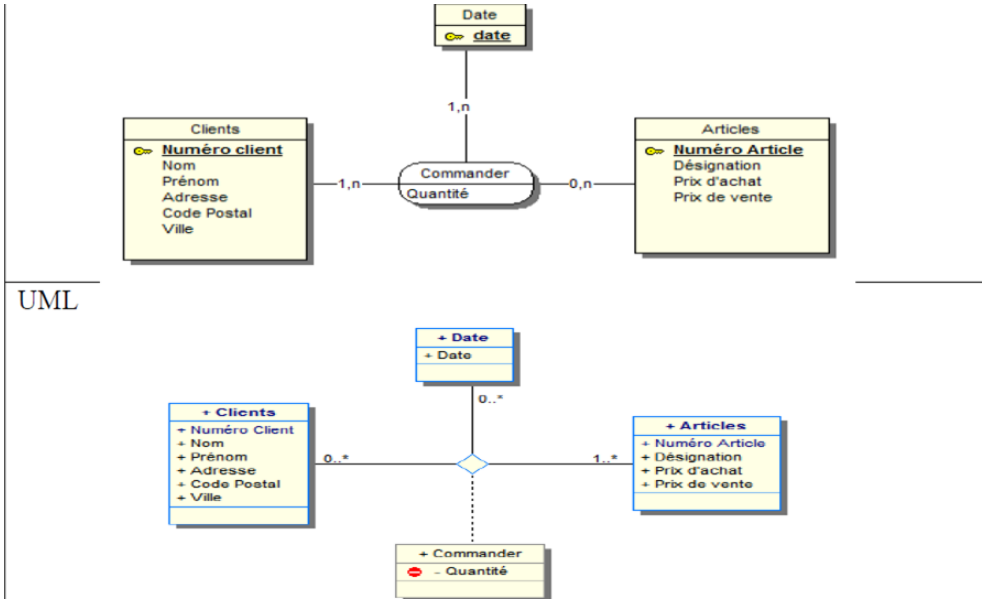
- **Association ternaires non porteuse**



MERISE/UML : Associations ternaires non porteuse

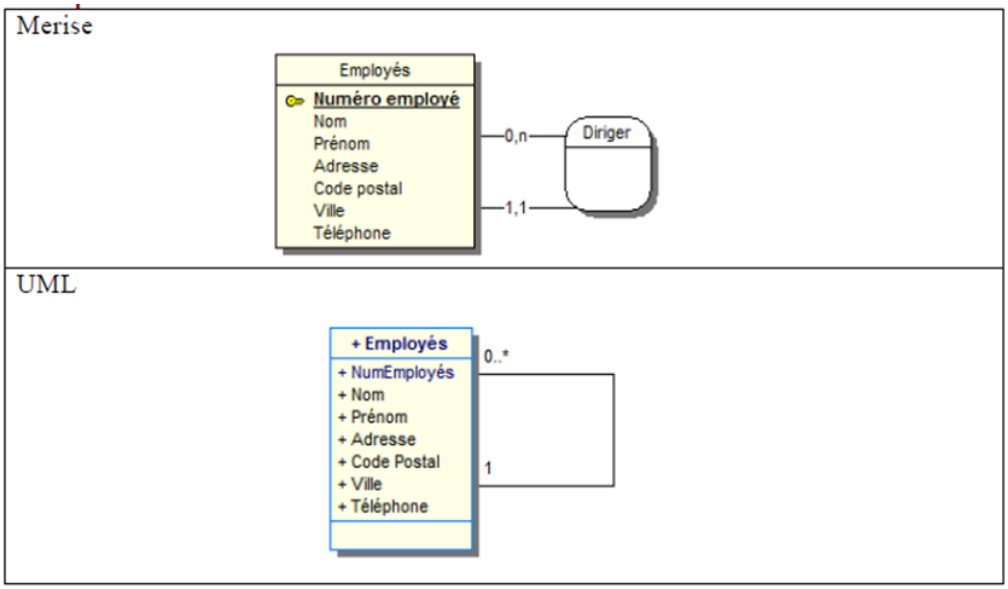
- **Association ternaires porteuse**





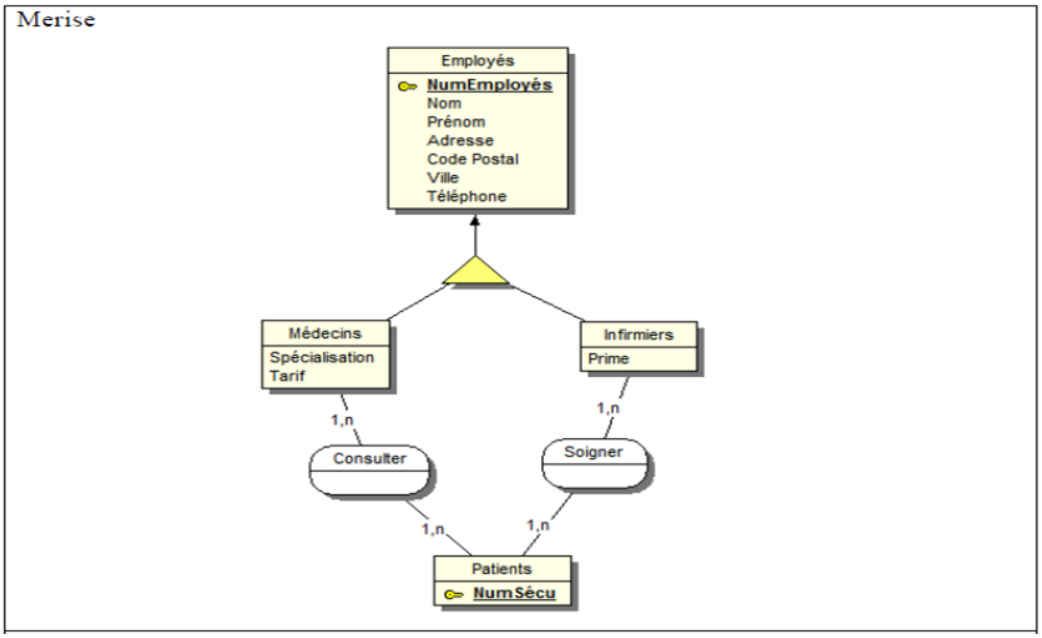
MERISE/UML

- **La réflexivité**

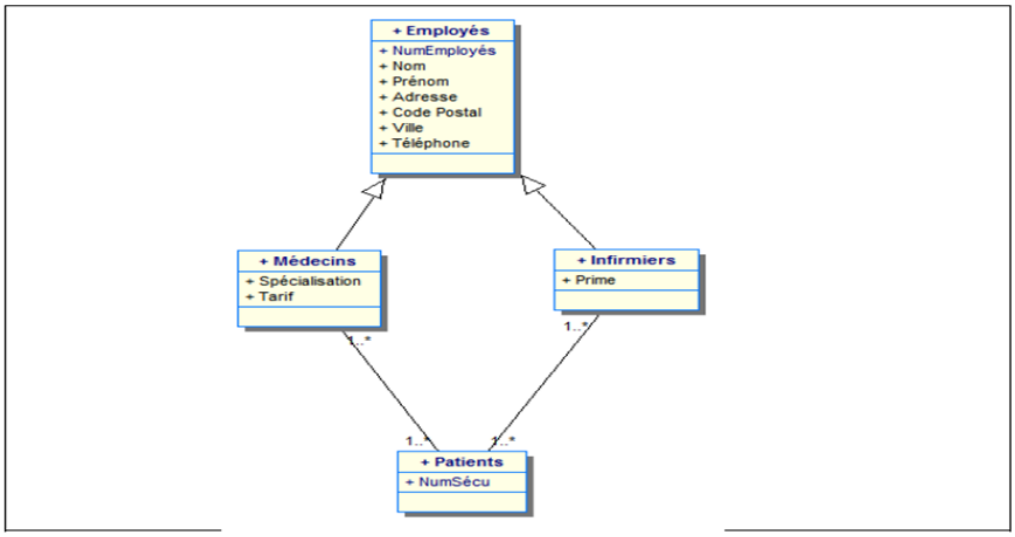


Représentation de la réflexivité

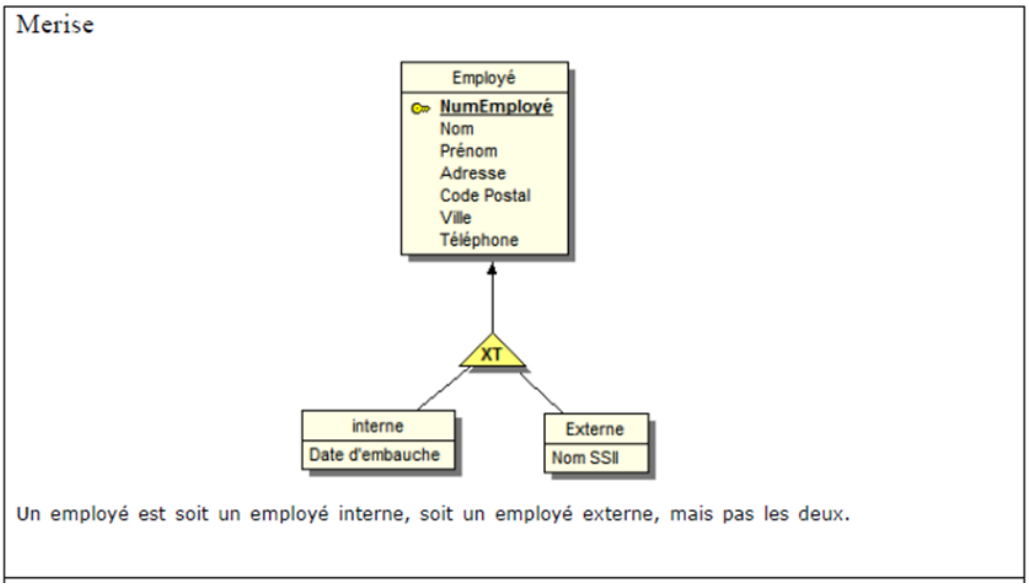
- **L'héritage**



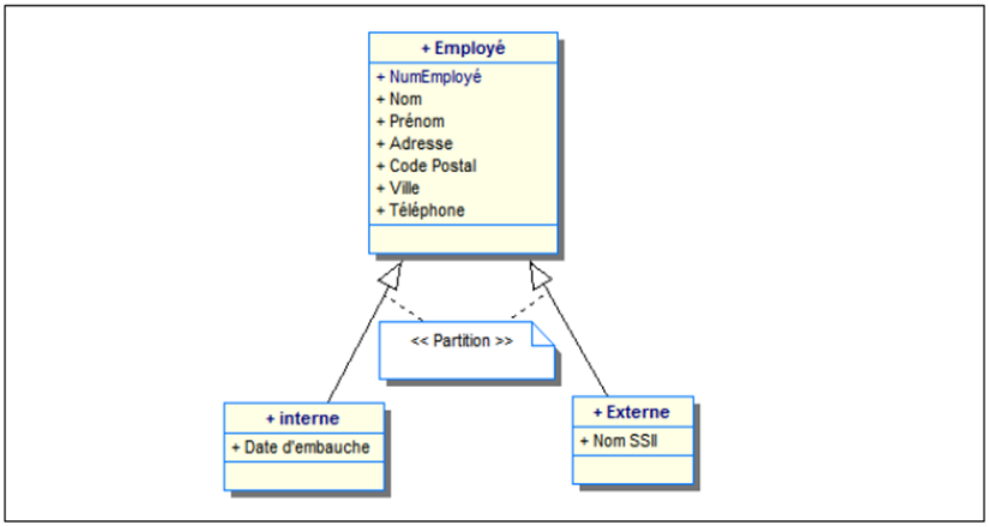
L'héritage MERISE



- **La contrainte de partition**

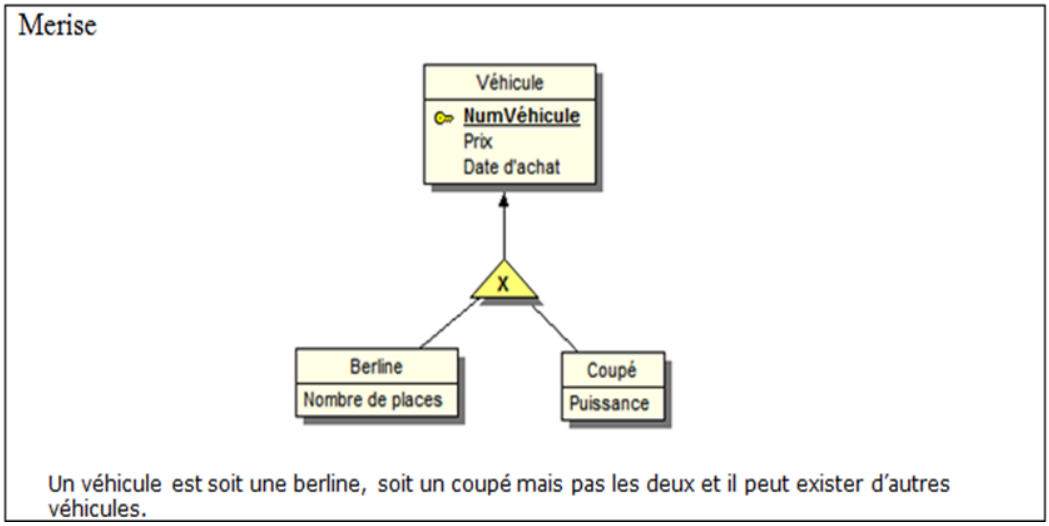


La contrainte de partition : MERISE

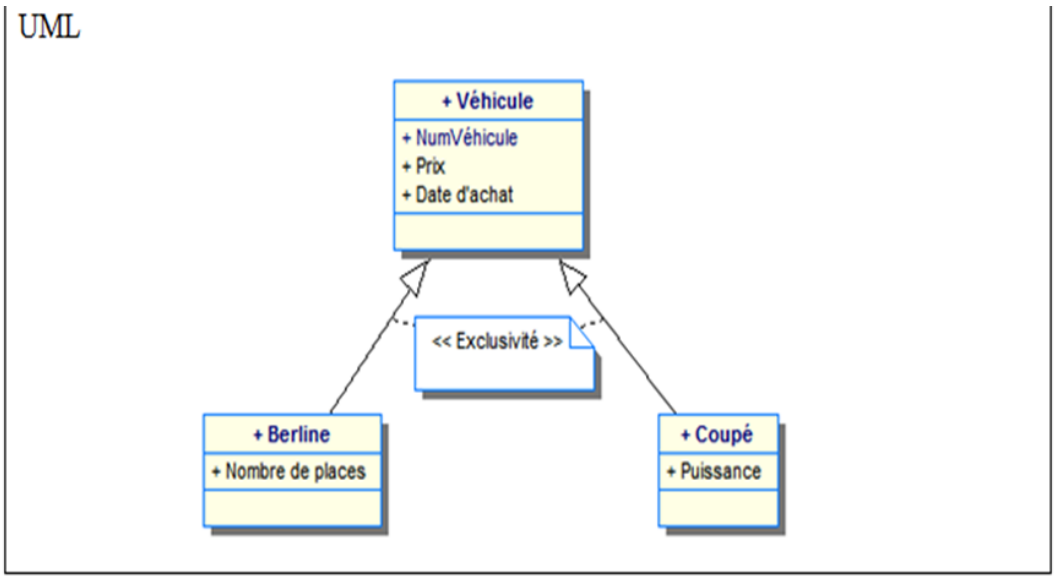


La contrainte de partition : UML

- **L'exclusion**



L'exclusion : MERISE



L'exclusion : UML

6.6. Exercices : Série de TD 06 et 07

Voici une série d'exercices comprenant des tâches de modélisation à la fois en utilisant MERISE et UML.

[cf. Série TD 06][cf. Série 07]

7. Conclusion

En conclusion de ce chapitre consacré à la présentation des méthodes de modélisation UML et MERISE pour les systèmes d'information, nous avons exploré deux approches fondamentales qui jouent un rôle crucial dans l'ingénierie logicielle. UML offre une méthode standardisée et polyvalente pour spécifier, concevoir et documenter les systèmes d'information à travers une variété de diagrammes et de perspectives. En revanche, MERISE se distingue par son approche structurée et rigoureuse, particulièrement adaptée à la gestion des données et aux processus métier complexes.

En comprenant les objectifs et les différences de ces deux méthodologies, vous êtes désormais mieux préparés à choisir et à appliquer la méthode la plus adaptée en fonction des exigences spécifiques de vos projets de développement de systèmes d'information. L'intégration efficace de ces approches peut non seulement améliorer la planification et la conception des systèmes, mais aussi renforcer leur robustesse et leur adaptabilité face aux défis technologiques et organisationnels.

En continuant à explorer et à approfondir ces concepts dans vos études et votre pratique professionnelle, vous serez en mesure de jouer un rôle essentiel dans la création et l'amélioration des systèmes d'information, contribuant ainsi de manière significative à l'innovation et à la performance des entreprises et des organisations dans un monde de plus en plus numérique et interconnecté.