III

CONCEPTION DU SYSTEME INFORMATISE D'INFORMATION

X. MODELISATION LOGIQUE DES TRAITEMENTS

10.1. Introduction

Après plusieurs années de pratique, on a été amené à considérer le comment du point de vue du gestionnaire (SIO et MOT) et du point de vue de l'informaticien (SII et le MLT). L'objectif tournera donc autour des moyens informatiques à réunir pour informatiser les activités présentes dans la modélisation organisationnelle des traitements (tâches, phases) compte tenu

- *Des ressources et contraintes logicielles et matérielles*
- *Des principes généraux d'ergonomie.*

Il s'agit de la mise en place d'un SGBD, de bases de données reparties, d'architectures client – serveur, avec la présentation des interfaces – homme – machine pour chaque poste de travail ainsi que pour chaque message ou état.

10.2. Formalisme de modélisation des traitements au niveau logique.

Bien que pas différent du formalisme des traitements organisationnels, le formalisme au niveau logique utilise les concepts suivants :

- *La machine logique*
- *L'événement / Résultat Message*
- *L'état*
- *L'unité logique de traitement ULT*
- *La procédure logique*

10.2.1. La machine logique

Une machine logique type est un ensemble de ressources informatique (matériel et logiciel) capable d'exécuter des traitements informatiques de façon autonome. Elle permet d'exprimer la répartition des traitements informatisés : Elle peut être :

- *Une machine physique*
  - *Micro-ordinateur autonome ou en réseau*
  - *Serveur*
  - *Mainframe ou mini avec terminaux fictifs*
- *Une partie de machine physique*
  - *Machine virtuelle sur un mainframe.*

Comme pour les postes du MOT, il faudra représenter chaque machine logique dans un tableau.

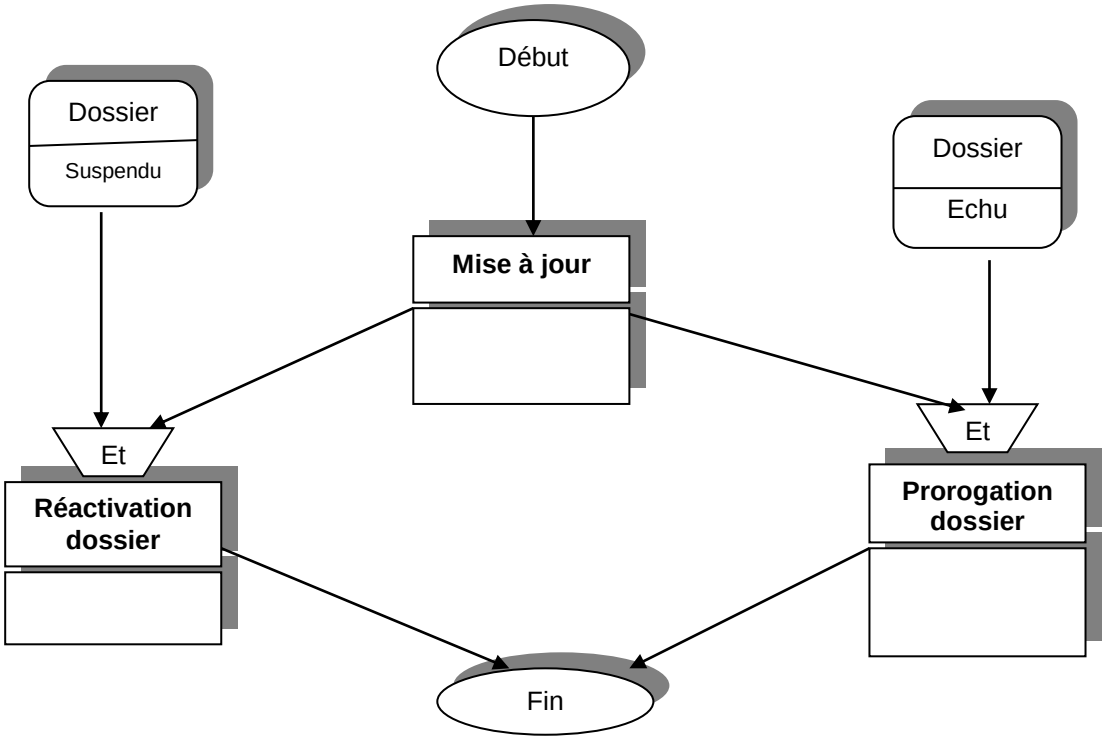
10.2.2. L'événement / Résultat – Message

A l'opposé du SIO où les événements / résultats – messages représentent les échanges du système avec son environnement ou entre les postes de travail, dans le MLT, ils désignent les échanges entre le SIO (les utilisateurs) et le SII.
Ils peuvent représenter :

- *Des échanges entre machines logiques ou unités de traitements logiques ULT*
- *Le début et la fin d'une procédure logique*

10.2.3. L'état

L'état (même modélisation que le SIO) exprime des conditions préalables ou des résultats conditionnels d'une ULT.



10.2.4. Unité logique de traitement ULT

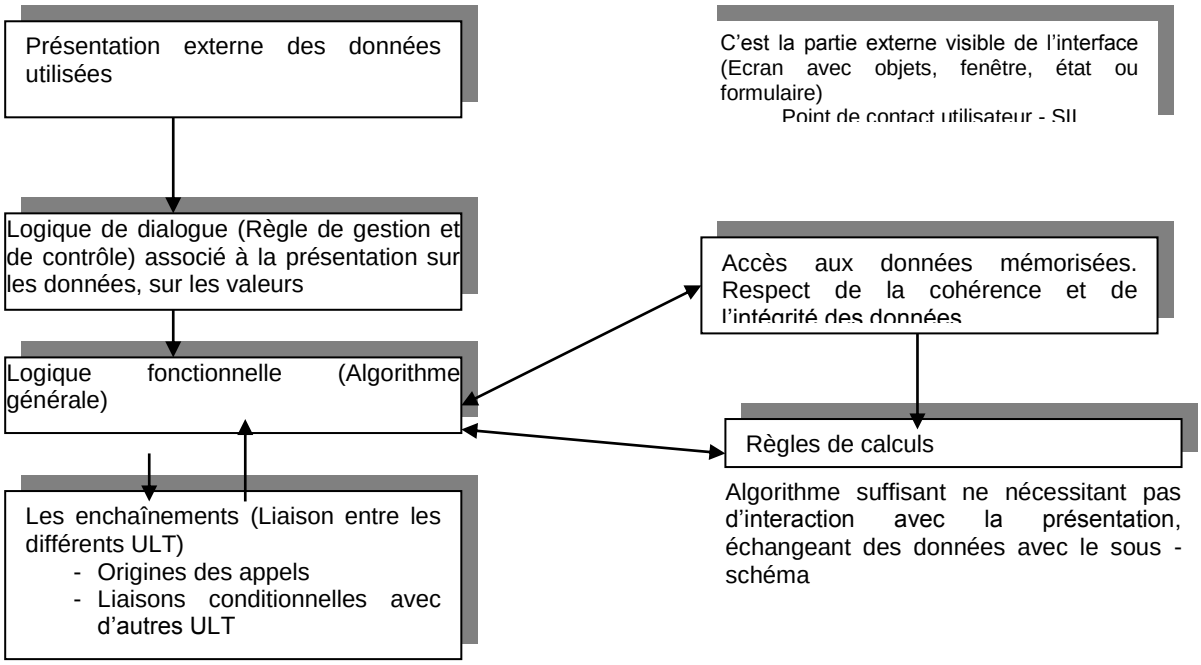
L'unité logique de traitement type est un ensemble de traitements informatiques perçus comme homogène en termes de finalités et elle se définit aussi par rapport à la cohérence des données du SII. Une ULT peut

- *Une transaction dans un système relationnel*
- *Une boîte de dialogue*
- *Une édition*
- *Un module dans une chaîne batch.*

Une ULT comprend selon leur nature :

- *Une interface*
- *Un traitement*
- *Un sous-schéma de données*

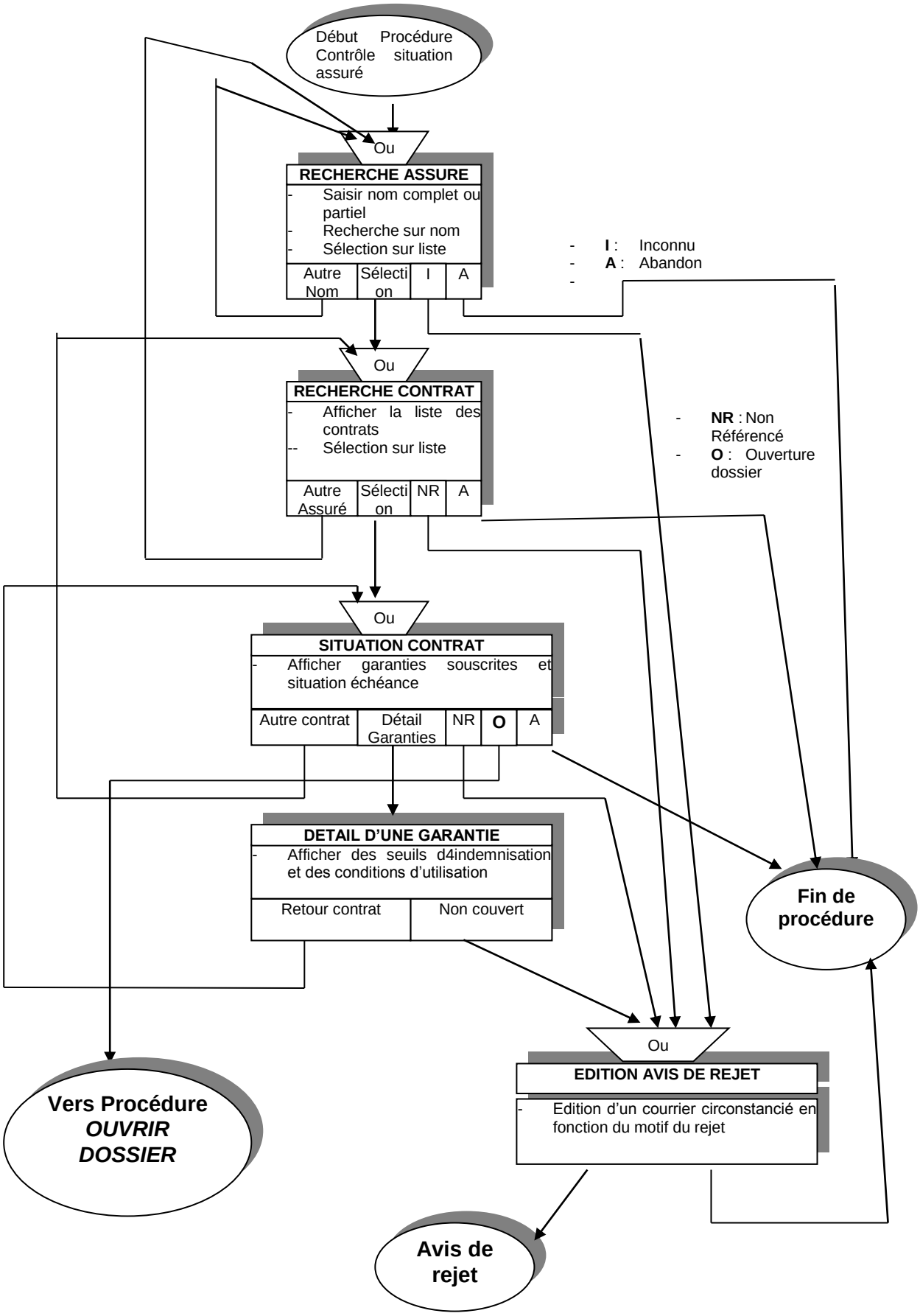
Ces éléments peuvent se décomposer en plusieurs fonctions représentées graphiquement de la façon suivante :



10.2.5. La procédure logique

C'est un enchaînement d'ULT réalisant l'informatisation d'une tâche ou d'une phase du MOT. Elle commence par un appel utilisateur du menu (ou du début spécifié) et se termine par le retour au menu d'appel. Au sein d'une procédure logique, les ULT et leur enchaînement correspondent à la résolution de l'activité organisationnelle associée.

Exemple : Procédure logique contrôle situation assuré



10.3. Conception des modèles logiques de traitements (MLT)

La construction d'un modèle logique de traitement exige une réflexion, une création, une invention et ne peut donc découler directement et automatiquement des modèles du SIO. Pour son élaboration plusieurs approches sont possibles.

10.3.1. Trois approches de conception des MLT

10.3.1.1. La décomposition des tâches organisationnelles

Les procédures logiques sont élaborées à partir du MOT. Les enchaînements et contenus des ULT sont construits pour chaque tâche du MOT. (Voir plus haut).

La procédure logique apparaît comme une décomposition de la tâche. Comme avantages, on peut dire que les ULT sont très proches des activités formulées dans le MOT. L'inconvénient est que pour des ULT identiques, on peut avoir à faire de longs développements.

10.3.1.2. Réutilisation des ULT

Il s'agit de rechercher dans le MOT, les tâches dont la description soit similaire ou proche et concevoir des ULT utilisable, en commun par différentes tâches. (Préconiser par le Génie logiciel). Dans leur utilisation les ULT communes seront amendées pour s'adapter éventuellement à chaque tâche.

L'avantage c'est qu'on écrira moins de code, moins de ULT. Comme inconvénients on peut avoir des ULT moins adaptées aux tâches, les appels ou enchaînements peuvent poser problèmes.

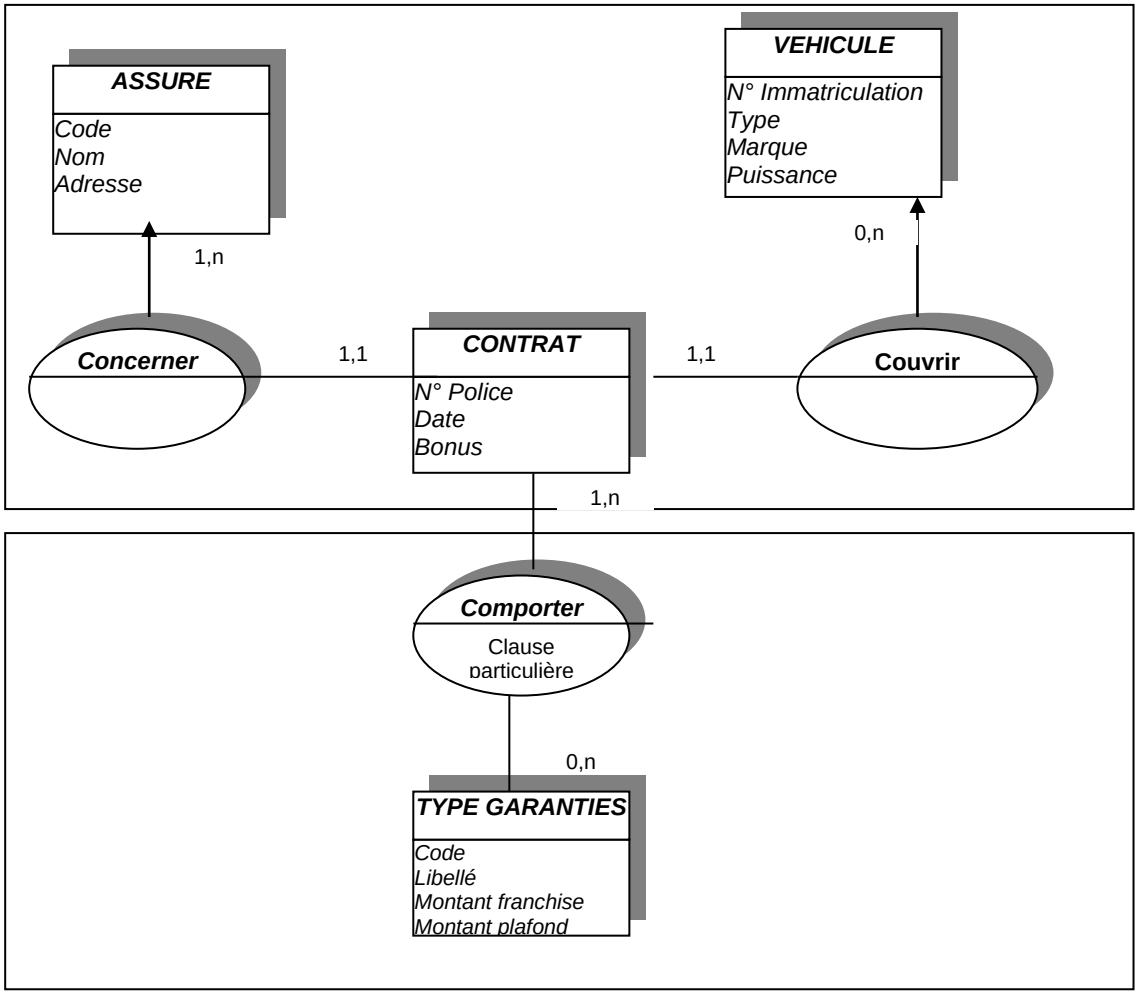
10.3.1.3. Conception des ULT autour des données.

Il s'agit ici de repérer dans le MOD, des ensembles de données perçus comme stables par les utilisateurs et se référant à des objets couramment utilisés dans les activités (Entités externes des vues externes). Si la vue externe comporte plusieurs entités reliées par une relation, s'appuyer sur la principales (pivot).

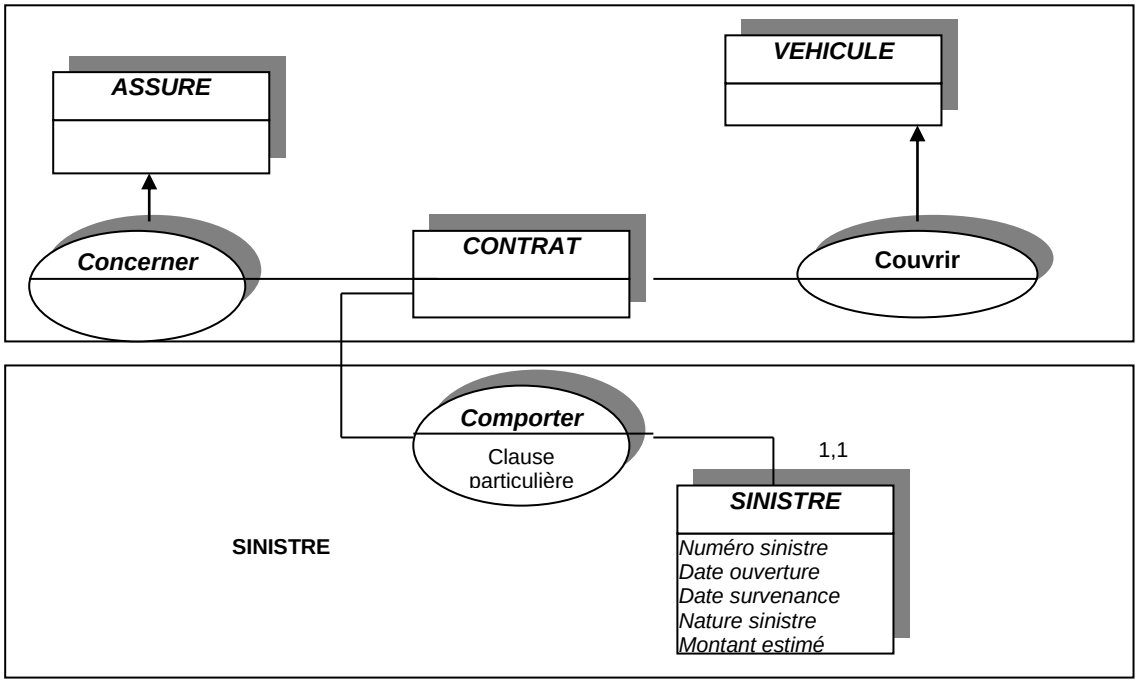
A ce sous-schéma, on associe une ULT qui permettra d'effectuer les opérations de base (création, modification, suppression, consultation)

Comme avantages, les ULT servent de base pour une approche par réutilisation et peuvent être directement implémentés dans des environnements graphiques en dehors d'approches procédurale. Comme inconvénient, on voit que la construction des procédures logiques à l'initiative des utilisateurs qui doivent maîtriser les enchaînements adéquats.

***Exemple** :*



Le contrat et ses garanties



10.3.2. Modularité des MLT

10.3.2.1 ULT et architecture logique d'application

Il s'agit de concevoir des applications respectant la séparation entre interfaces utilisatrices et le noyau de l'application afin de garantir une indépendance du dialogue interactif. On distingue En généra trois modules à savoir

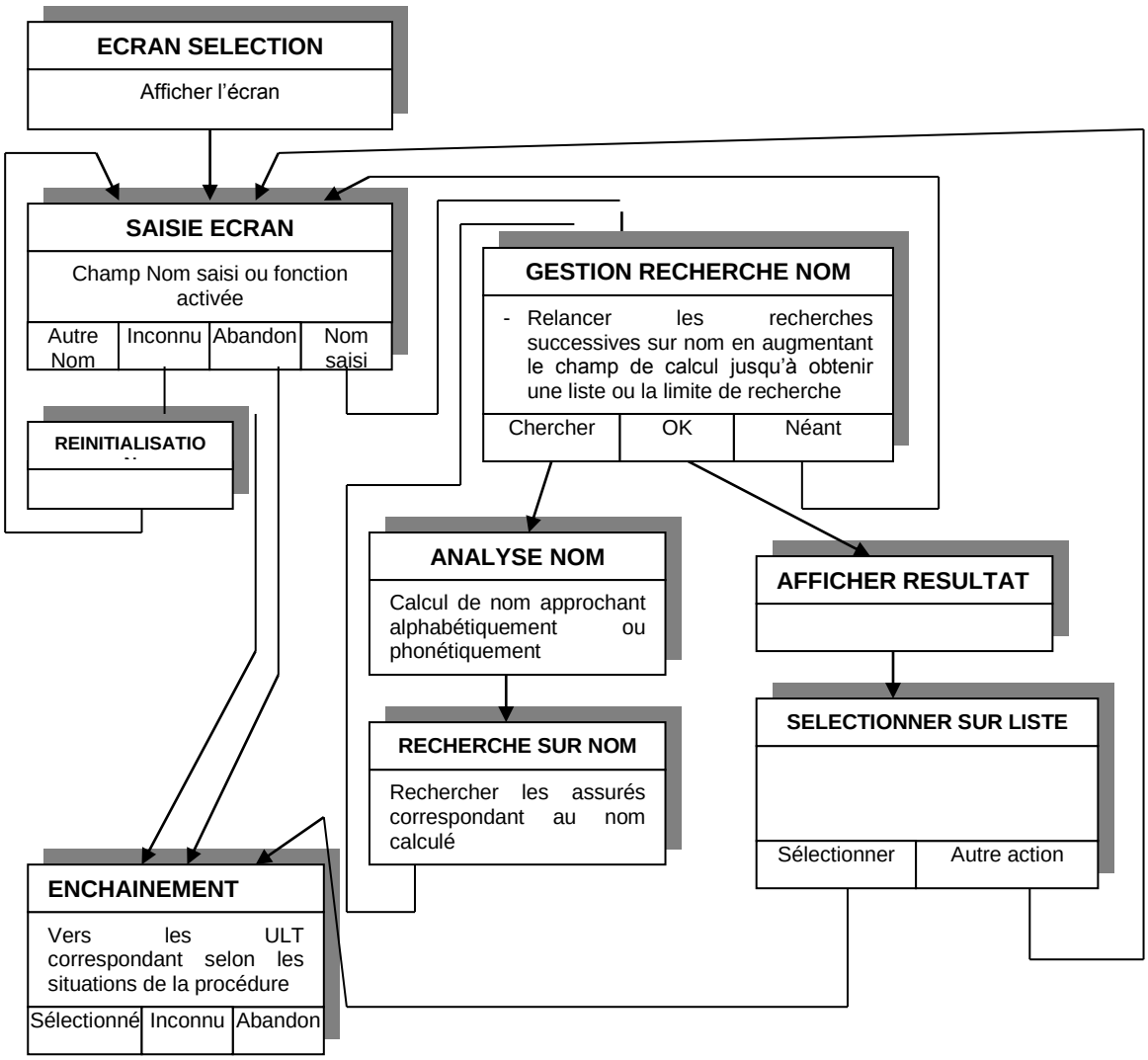
- *Les interfaces graphiques homme / machine*
- *Le noyau applicatif*
- *Le guidage fonctionnel*

10.3.2.2. Décomposition des ULT par nature

Dans cette décomposition, les ULT sont décomposés en ULT élémentaires respectant les règles suivantes :

- *De nature (Présentation, dialogue, logique fonctionnelle, accès aux données, règles, enchaînement)*
- *De machine logique en cas de répartition.*

Exemple : Décomposition de L'ULT Recherche assuré en ULT élémentaires.



10.4. MLT repartis

La répartition logique des traitements porte sur les unités logiques de traitement associées aux tâches du MOT. Grâce aux nouvelles technologies de l'information, il est aujourd'hui de plus en plus questions de traitements repartis.

10.4.1. Démarche de répartition

Pour construire un MLT reparti, il faut adopter la démarche suivante :

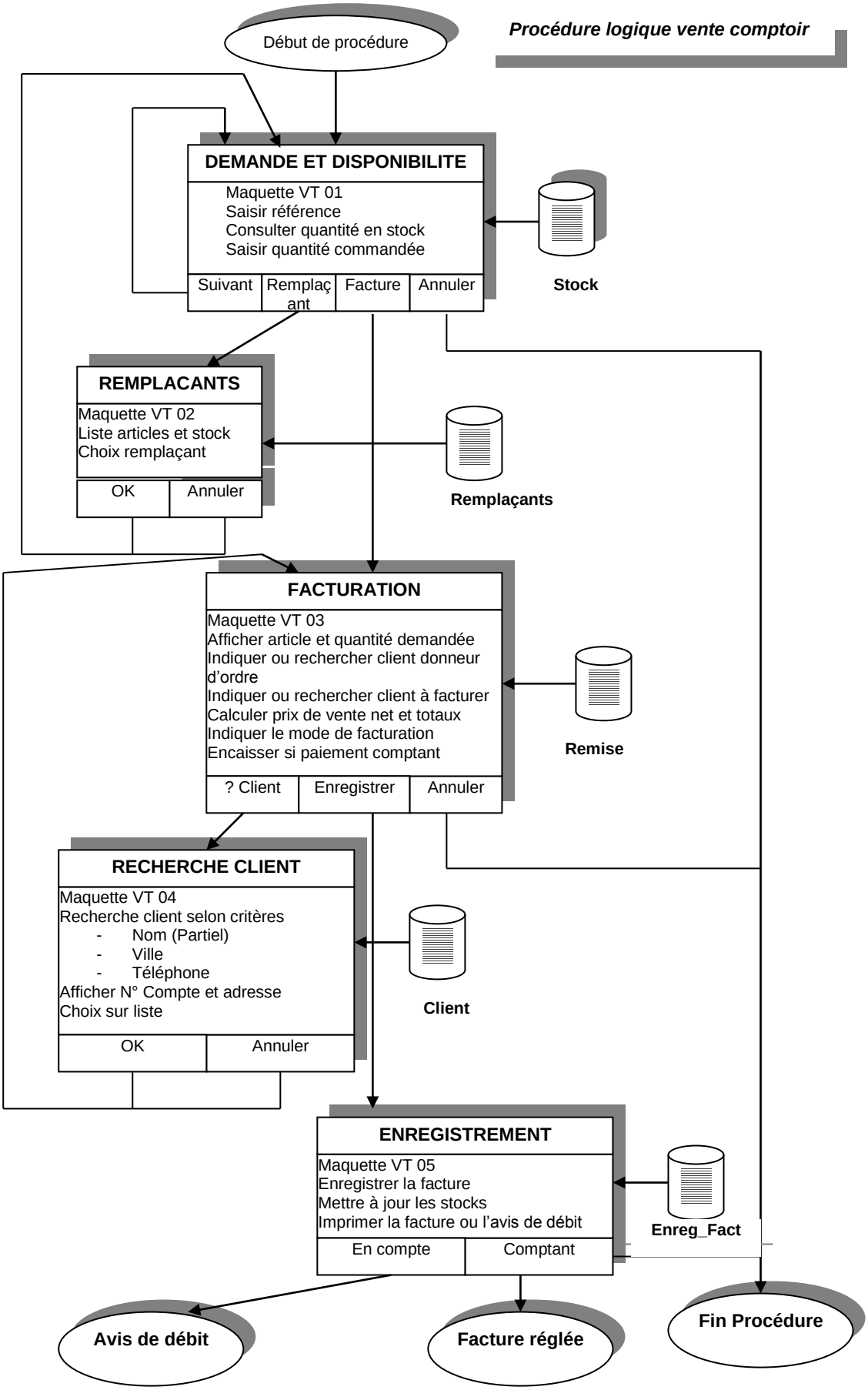
- *Elaborer un MLT non reparti sans tenir compte de la future répartition*
- *Définir une architecture matérielle, c'est à dire définir :*
  - *La machines logiques et leurs caractéristiques techniques*
  - *Les sites logiques*
  - *Les ressources d'environnement (Système d'exploitation, logiciel de développement, communication)*
- *Repartir les traitements en affectant les différents ULT aux machines logiques.*

10.4.2. Modalités de répartition

Repartir des données et les traitements, c'est les installer sur des machines logiques. On dispose de trois modalités de répartition essentielles :

- *Traitements coopératifs : Une ULT primaire est exécutée sur plusieurs machines logiques. (ULT non décomposé en ULT élémentaires)*
- *Données identiques sur des machines logiques différentes (Accès concourant)*
- *Client - Serveur*

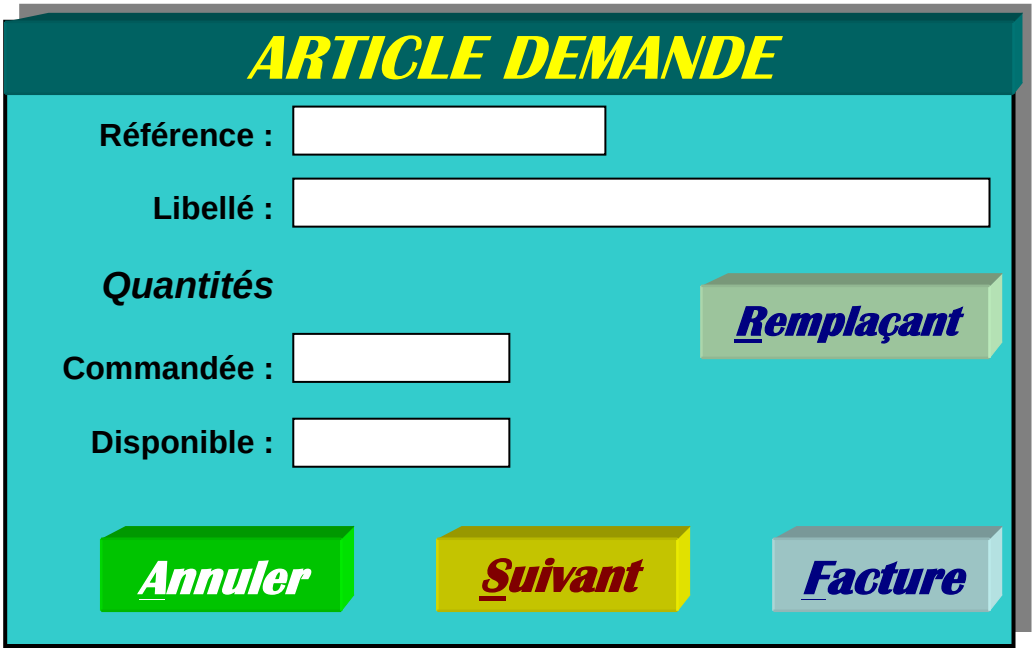
10.5. Exemple pratique d'un MLT futur



Représentation graphique des ULT

❶. *ULT demande et disponibilité*

Présentation de la maquette



Logique du dialogue

- *Saisir la référence. Afficher le libellé si référence existe et quantité en stock ou erreur si inconnue.*
- *Saisir quantité demandée*

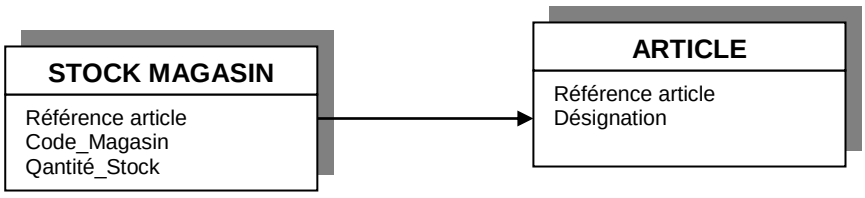
Logique fonctionnelle

Règle

- *si quantité demandée* **>** *quantité disponible*

Alors { *Erreur ou remplaçant*

Sous-schéma

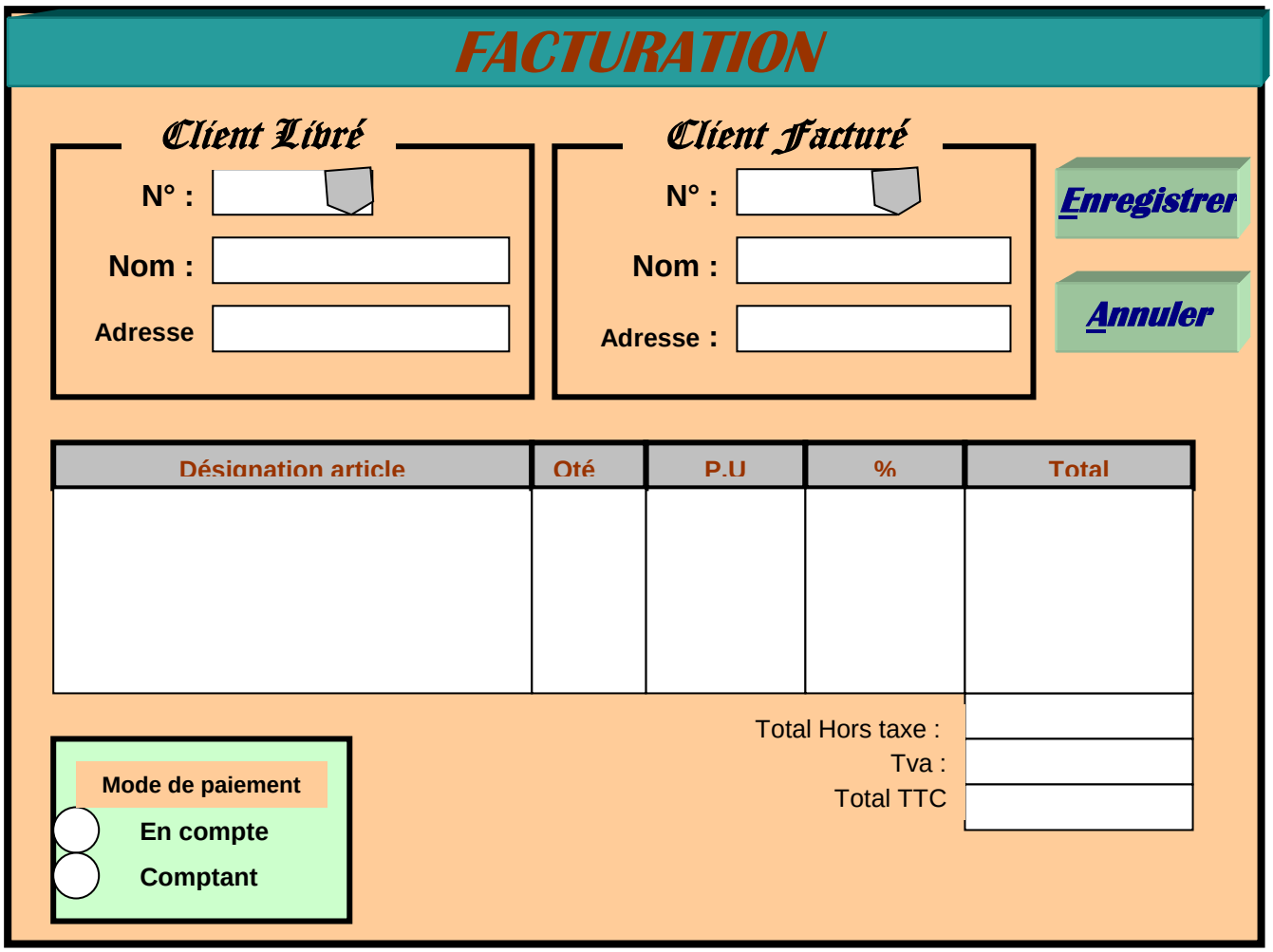


Enchaînement

Condition	Action	Résultats
Suivant	**Bouton**	*Conserver la quantité demandée* *Effacer les lignes pour ressaisie*
Remplaçant	**Bouton**	*Si l'article possède des remplaçants, appeler l'ULT avec passage de référence article* *Sinon afficher message Pas de remplaçant*
Facturer	**Bouton**	*Appeler l'ULT facturation avec passage des références et quantités des articles demandés*
Annuler	**Bouton**	*Fin de procédure*

❷. *ULT facture (appel par bouton)*

Présentation



Logique de dialogue

- *Saisir N° ou bouton ? pour le client livré afficher nom et adresse*
- *Saisir N° ou bouton ? pour le client facturé afficher Nom et adresse*

Si inconnu, afficher message client inconnu

- *Saisir mode de paiement*

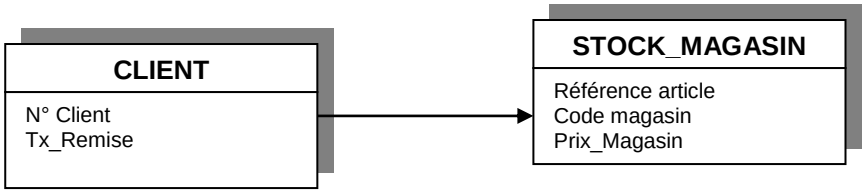
Logique fonctionnelle

Afficher les articles et les quantités provenant de l'ULT demande et disponibilité chiffrer la facture selon les règles.

Règles

- *Détermination de la remise. Le % de remise est le maximum des % remise client et remise article.*
- *% remise appliqué = Max (% remise client, % remise article)*
- *Calcul montant total ligne = PU* quantité* (1 -% Remise)*
- *Calcul montant total facture = Somme(Total lignes)*

Sous-schéma

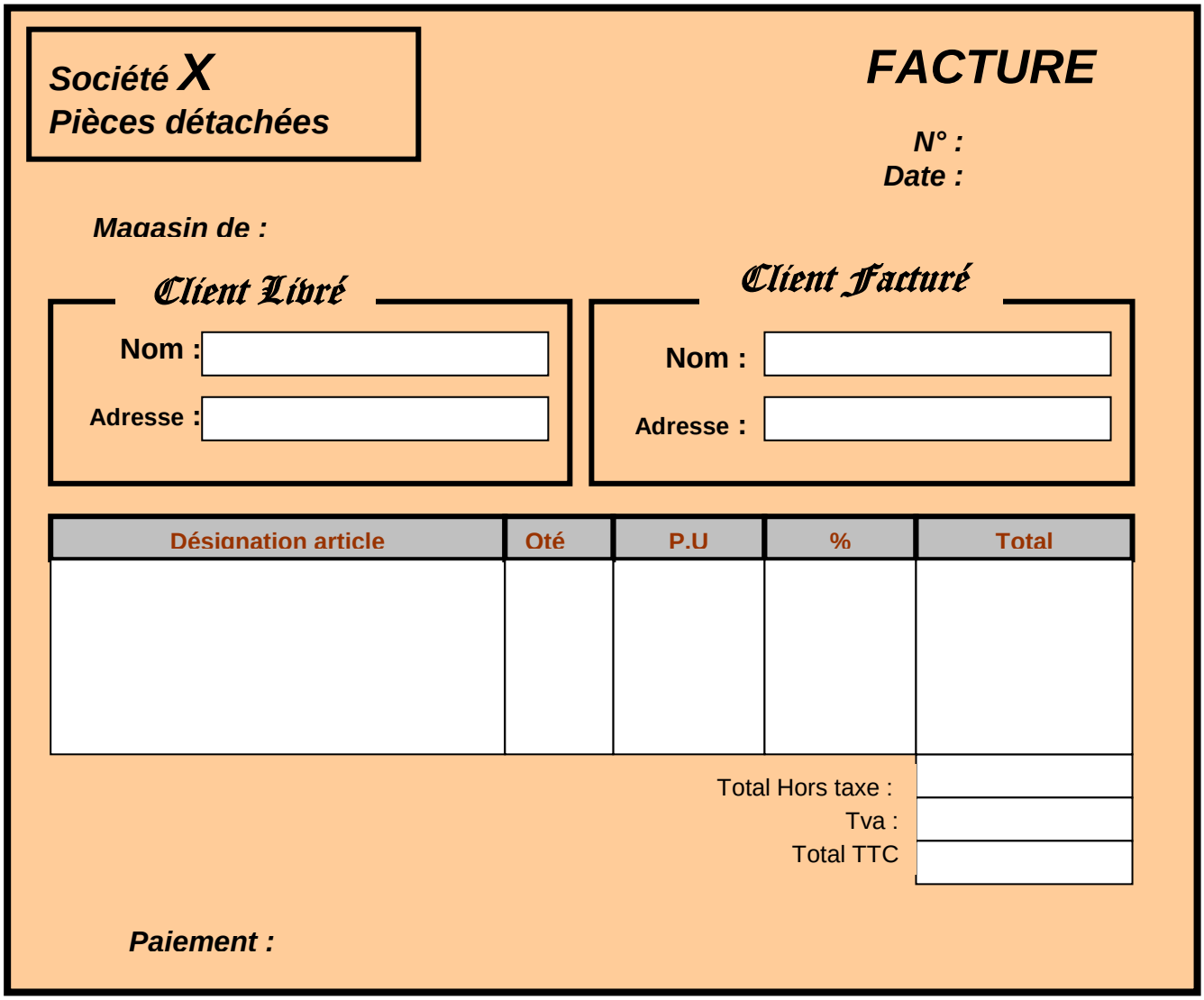


**** Enchaînement***

Condition	Action	Résultat
? client	***Bouton***	*Appeler l'ULT recherche client*
Enregistrement	***Bouton***	*Appeler l'ULT Enregistrement et édition avec passage de l'ensemble des informations présente*
Annules	***Bouton***	*Fin procédure*

❺. ULT Enregistrement et édition

Présentation



Société X
Pièces détachées

FACTURE

N° :
Date :

Magasin de :

Client Livré

Nom :
Adresse :

Client Facturé

Nom :
Adresse :

Désignation article	Oté	P.U	%	Total

Total Hors taxe :
Tva :
Total TTC

Paiement :

Logique de dialogue

sans objet

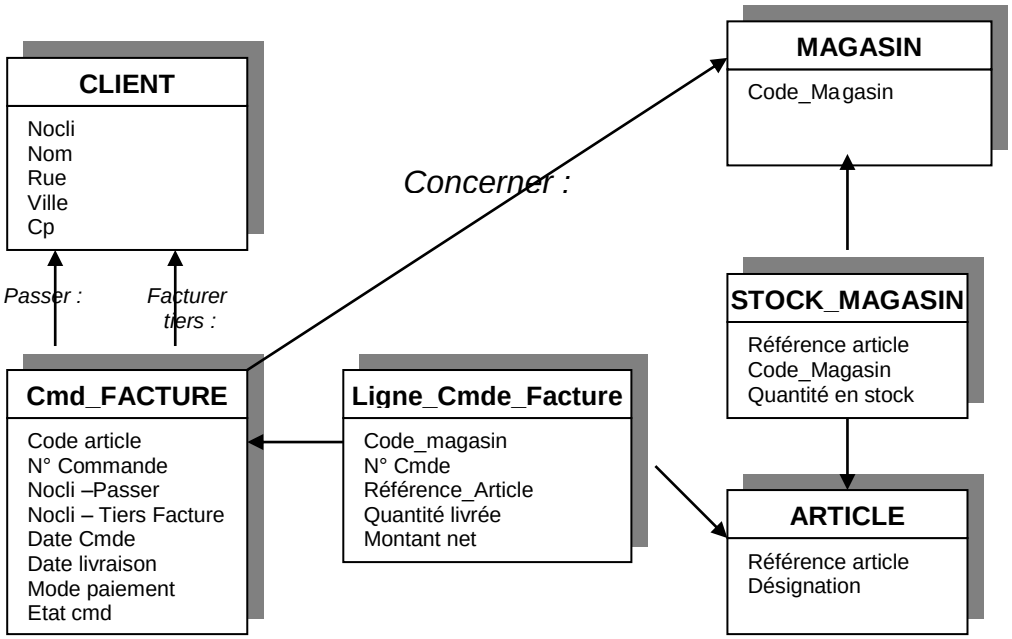
Logique fonctionnelle

- *Créer la facture*
- *Mettre à jour la quantité en stock*
- *Imprimer selon le formulaire correspondant*
- *Retour à l'ULT facturation*

Règle

- *Détermination N° commande – Génération par le système d'un numéro chronologique.*
- *Détermination du code du dépôt, de la date de commande, mise à jour stock*

Sous-schéma



Enchaînement

Condition	***Action***	***Résultat***
OK	***Bouton***	*Appeler ULT facturation, avec transmission du client sélectionné*
Annuler	***Bouton***	*Appeler l'ULT facturation*

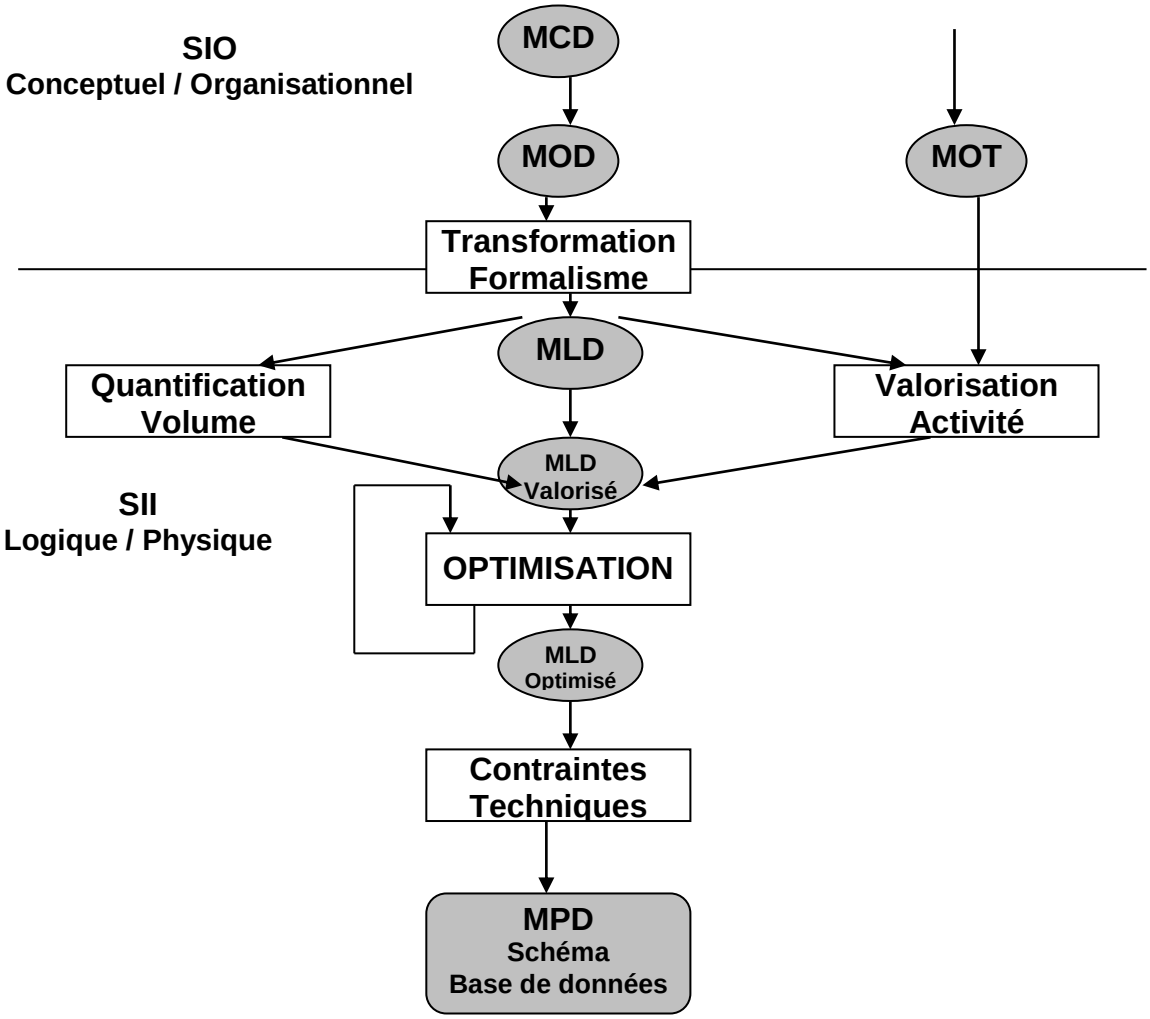
XI MODELISATION LOGIQUE DES DONNEES (MLD)

11.1 Introduction

La modélisation logique des données consiste à :

- *Transformer le MOD entité / relation en MLD exprimé dans un formalisme logique adapté au SGBD envisagé*
- *Quantifier en volume, le MLD*
- *Valoriser l'activité générée par les modèles externes associés aux traitements*
- *Une optimisation générale*

Deux formalismes théoriques de base de données existent pour la représentation du MLD : le modèle navigationnel ou CODASYL et le modèle relationnel ou SGBD relationnel. Ce dernier, parce qu'utilisé par presque tous les systèmes sur le marché sera utilisé.



11.2. Modèle logique de données relationnel

Le modèle relationnel présente deux aspects fondamentaux : La conception des tables relationnelles et la détermination de l'algèbre relationnel permettant la manipulation des tables.

11.2.1. Concepts de base d'un modèle relationnel

11.2.1.1. Concepts structuraux

Le concept relationnel s'appuie sur trois concepts structuraux qui sont la relation, l'attribut et le domaine.

La relation, peut être définie grossièrement comme un tableau de données. Au lieu de relation, on utilisera le mot table. Les colonnes de cette table sont appelées attributs. Le domaine d'une colonne correspondant aux types de valeurs (entier, réel, booléen, caractères ou type construits (couleurs, Nom, Date, etc.)).

- ***La cardinalité** d'une table définit le nombre de lignes de la table*
- ***Le degré** d'une table définit le nombre d'attributs ou de colonnes de la table.*
- ***Le schéma** d'une table est constitué du nom de la table suivi de la liste des attributs avec leurs domaines de valeurs.*
  ***Exemple** : **Pièce** (**N°** : entier **nom** : car (10); **couleur :** couleur; **Poids** : réel; **ville** : car (10))*
- ***L'extension** d'une table , c'est l'ensemble de lignes ou tuples (ou occurrences), définies par les valeurs prises par les attributs.*

Pièce

N°	Nom	Couleur	Poids	Ville
P_1	Ecrou	Rouge	14	Duékoué
P_2	Bouton	Vert	17	Oumé
P_3	Vis	Bleu	12	Guiglo
P_4	Vis	Rouge	14	Man

Cardinalité de la pièce = 4. degré = 5

Une base de données relationnelle est un ensemble de tables

11.2.1.2. Les contraintes d'intégrité

Une contrainte d'intégrité est une assertion qui doit être vérifiée par les valeurs d'attributs constituant une base de données. Les deux principales sont :

- *La contrainte d'unicité de valeurs*
- *La contrainte référentielle permettant de relier deux tables.*

**** Contrainte d'unicité de valeur, clé primaire d'une table.***

Les valeurs prises par un attribut ou une composition d'attributs d'une table peuvent être déclarées uniques pour toute extension de cette table. Ces attributs permettent d'identifier de façon unique, chaque tuple : on parle alors de clé primaire simple (un seul attribut) ou composé (plusieurs attributs).
*Pièce (**N° pièce**, Nom, dépôt) : clé primaire = N° pièce.)*

**** Contraintes référentielles***

C'est un lien sémantique entre deux tables réalisé par la duplication de la clé primaire d'une table dans une autre.
***FOURNIT** (**N° fournisseur, N° pièce**, délai)*

11.2.1.3. Conception du schéma relationnel

En principe, lorsque les tables sont drivées du MOD, elles sont déjà de bonnes tables. Lorsque le schéma déroulé d'une autre approche, il faudra corriger les éventuelles redondances et possibilités de valeurs nulles.

11.2.2. Eléments d'algèbre relationnelle

Les éléments d'algèbre sont un ensemble de sept (7) opérations qui permettent des manipulations ensemblistes sur les tables sans faire appel à un cheminement précis. Ce sont :

- ***La sélection** ou restriction, une opération unaire qui consiste à sélectionner un ensemble de triples selon un critère de sélection pouvant porter sur un ou plusieurs attributs. La table résultante à la même degré et les mêmes colonnes que la table initiale.*

- ***La projection**, une opération un aire qui consiste à :*
  - *Ne retenir que certains attributs de la table*
  - *Eliminer les occurrences identiques.*

  La table formée a sa structure

- ***La jointure**, opération qui permet d'obtenir une nouvelle table par la composition de deux ou plusieurs tables. Elle consiste à :*
  - *Faire le produit cartésien de deux tables, c'est à dire concaténer chacune des lignes de la 1ère table avec chacune des lignes de la deuxième table.*
  - *Effectuer une opération de sélection ou qualification. Généralement une égalité entre un attribut de la première table et un attribut de la seconde (appelé attribut de jointure.*

    ***Exemple** :*
    ***Client** (N° client, nom)*
    ***Commande** (N° commande, n°client , date)*
    ***Jointure** : N° client = Commande. N°client*

  - *Effectuer une opération de projection pour réduire le schéma de la table résultante.*

- ***La division**, une opération binaire dont le quotient d'une table R par une table S est constitué des éléments de R n'appartenant pas à S.*

- ***L'union, l'intersection et la différence**, sont des opérations binaires correspondant aux opérations de la théorie des ensembles. Elles s'appliquent sur des tables de même schéma.*

11.2.3. Processus de normalisation

*La normalisation est un processus de décomposition qui découle des dépendances fonctionnelles des attributs d'une table. Codd a proposé 3 formes **normales (1ère , 2ème et 3ème forme normale)** complétée par la 4ème, la 5ème et la forme de Boyce – Codd.*

11.2.3.1. Notion de dépendance fonctionnelle.

Un attribut B d'une table R est fonctionnellement dépendant d'un attribut de R si à tout instant, chaque valeur de A n'a qu'une valeur associée de B. On note A B
Les dépendances fonctionnelles sont :

- *Réflexive A → B*
- *Transitive A → B et B → C alors A → C*

*Une **clé primaire** d'une table est un attribut tel que tous les attributs (non clé) de la table sont en dépendance fonctionnelle avec la clé primaire. (clé unique ou composée)*

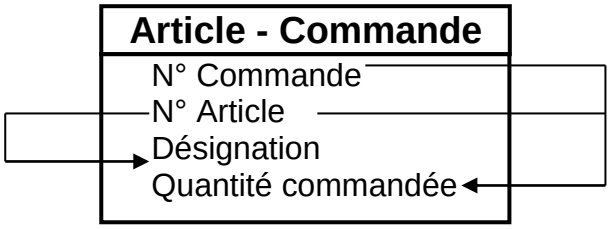
11.2.3.2. Première forme normale (1NF)

Une table est en première forme normale si elle possède une clé et si la table ainsi obtenue ne présente ni de tuples répétitifs, ni d'attributs nuls.

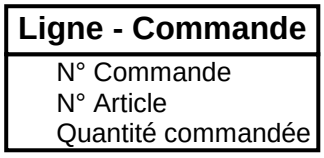
11.2.3.3. Deuxième forme normale (2NF)

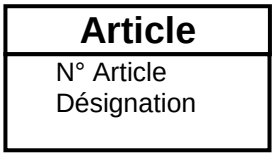
Une table en 2ème forme normale exige qu'elle soit d'abord en 1ère forme normale. Les attributs non clé (clé composée) primaire dépendent de la clé primaire.

***Exemple** : Table Article – Commande*



*Désignation ne dépend pas de la clé **N – Commande, N° Article**, mais de **N° Article**, une partie de la clé. D'où les deux tables.*

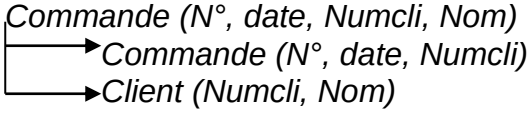




11.2.3.4. Troisième forme normale (3NF)

Une table en 3ème forme normale exige qu'elle soit en 2ème forme normale. La 3NF élimine les dépendances transitives. L'attribut de transitivité devient la clé d'une autre table.

Exemple

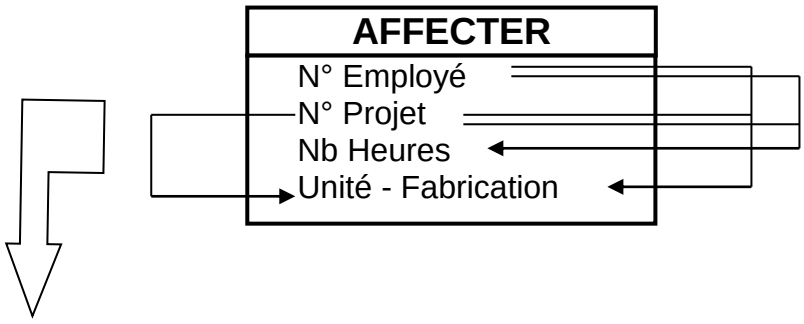


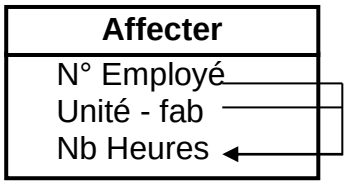
11.2.3.5. Forme normale de Boyce – Codd (BCNF)

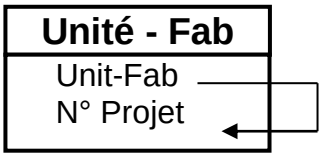
Une forme normale en BCNF permet d'éviter les redondances dues à l'existence de dépendances fonctionnelles autres que celles de la clé vers des attributs non-clé.

Exemple :

Reg1 : *Un employé exerce un certain nombre d'heures par projet*
Reg2 : *Une employé peut être affecté à un certains nombre de projets*
Reg3 : *Chaque projet est exécuté dans une unité de fabrication.*







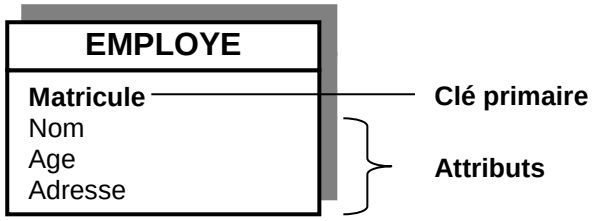
11.2.4. Notion de vues relationnelles

Une vue est une table virtuelle pouvant être composée d'une ou plusieurs tables obtenues par intégration (requêtes) mettant en œuvre les opérations d'algèbre relationnelle. Le langage normalisé SQL est utilisé.

11.2.5. Formalisme graphique du modèle relationnel.

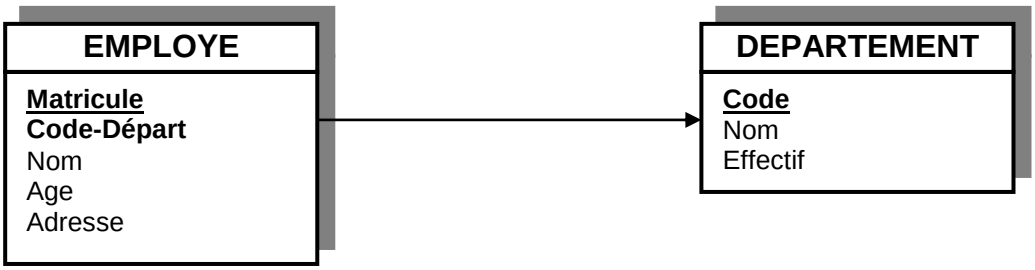
Plusieurs formalisations des modèles logiques existent.

11.2.5.1. Table, attributs et clé primaire



11.2.5.2. Contraintes d'intégrité

*Il y a une CIR (Contrainte d'intégrité Référentielle) entre deux tables **A** et **B** lorsqu'à une valeur de la clé primaire A ne correspond qu'une seule valeur de la clé primaire de B et qu'à la clé primaire de B corresponde plusieurs valeurs de la clé de A.*



Pour employé : *Matricule =* ***clé Primaire***
Code-départ ***= clé étrangère***

Pour une question de clarté, la clé primaire sera en gras souligné et la clé étrangère en gras.

Schéma relationnel

EMPLOYE *(**Matricule**,* ***Code-Départ****, Nom, Age, Adresse)*
Département *(**Code**, Nom effectif)*

11.2.53. Clé primaire composée référentielle

Exemple

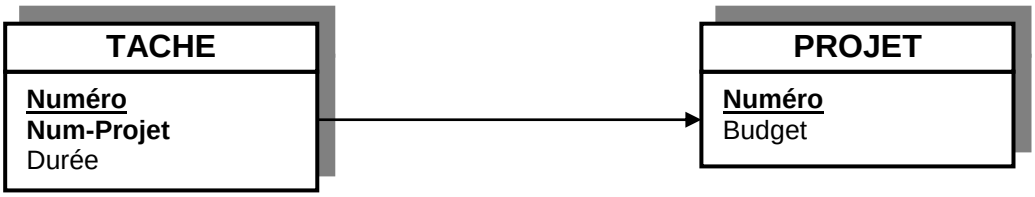


Schéma relationnel

Tâche *(**Numéro**,* ***Num-Projet****, Durée)*
Projet *(**Numéro** Budget)*

11.2.6. Règles de transformation du formalisme Entité-Relation en formalisme relationnel

Bien qu'il soit possible de construire un modèle relationnel à partir de plusieurs approches, l'approche ***MOD/MCD → MCD*** *est mieux adaptée. Le MLD qui en découle est déjà en 2ème forme normale, mais pas nécessairement en 3ème NF (On ne s'est pas trop préoccupé des redondances minimales)*