PROBLEMES D'ACQUISITION DANS LE SYSTEME PIVOINES

G. Anfré

Université de NANCY I

PROBLEMES D'ACQUISITION

DANS LE SYSTEME PIVOINES

Thèse soutenue le 5 Juillet 1977

par

Georges ANFRÉ

pour obtenir le grade de docteur de spécialité

en Informatique

devant la commission d'examen

Je veux tout d'abord remercier Monsieur Jean-Claude BOUSSARD qui m'a encouragé à préparer une thèse de troisième cycle et qui m'a facilité le contact avec l'Université de Nancy.

Je remercie également Monsieur Claude PAIR pour son accueil et pour l'aide précieuse de son expérience.

Je veux aussi exprimer ma reconnaissance à Madame Marion CREHANGE pour les conseils et les encouragements qu'elle n'a cessé de m'apporter, avec patience, durant la réalisation de ce travail.

Enfin je remercie Monsieur Jean-Claude DERNIAME qui a bien voulu accepter de faire partie du Jury.

INTRODUCTION

Une base de données est un "réservoir" d'informations mises en commun par plusieurs utilisateurs qui veulent pouvoir les traiter à tout moment et de façon variée.

Une qualité essentielle d'un système de gestion de base de données (S.G.B.D.) est la possibilité d'indépendance des données, en d'autres termes, la séparation entre la description des données et les programmes d'exploitation. Cette séparation permet à des utilisateurs et à des programmes multiples d'accéder aux mêmes informations sans difficultés excessives.

Nous allons nous intéresser plus particulièrement aux bases de données et aux traitements des données les constituant.

Dans tout notre travail nous appellerons aussi bien informations que données les objets sur lesquels opère un ordinateur.

La construction d'un système de gestion de base de données doit avoir pour objectif de permettre aux utilisateurs de fournir les renseignements nécessaires et suffisants sur les applications pour que chaque mise en oeuvre du système, c'est-à-dire chaque déroulement d'un traitement puisse se faire simplement. C'est l'un des buts atteints par le système PIVOINES (cf (2)). En effet, dans ce système, on définit une application aussi complexe soit-elle en apportant des renseignements sur le genre et la représentation des données que le système doit traiter.

Ces renseignements sont décrits une fois et, pour chaque mise en oeuvre du système, l'utilisateur n'a plus qu'à transmettre l'énoncé de la mise en oeuvre. Un programme général, partant de la description des données et de la description de la mise en oeuvre, construit alors un programme d'exploitation des données. Ainsi des

transformations au niveau du genre des données ou de leur représentation n'entraînent pas de bouleversement au niveau du traitement considéré.

Ce but est atteint et donc les traitements c'est-à-dire les acquisitions, interrogations, modifications ne sont pas à reformuler pour chaque genre et chaque représentation des données car les descriptions du genre et de la représentation sont des données explicites du système.

C'est dans cette optique que nous avons essayé de résoudre le problème de l'acquisition, c'est-à-dire de la transformation d' un objet appartenant à un support externe à la base de données en un objet appartenant à un support interne à celle-ci.

Remarquons, dès maintenant, que l'acquisition peut être considérée comme un cas particulier d'une modification: l'adjonctio (c'est alors l'adjonction à partir de rien, c'est-à-dire sans accès

Dans le premier chapître, nous précisons les objets sur lesquels nous allons travailler, le but de notre étude, les moyens pour y parvenir.

Dans les cinq chapîtres suivants, nous précisons en détail les outils construits.

Nous présentons enfin, sous forme modulaire, le programme d'acquisition d'une information.

CHAPITRE I

OBJETS - BUT - MOYENS

Dans ce chapître nous précisons d'abord les objets sur lesquels nous allons travailler. Ensuite nous précisons le but de notre étude et les moyens pour y parvenir.

1.1 Objets sur lesquels nous allons travailler

Nous reprenons ici des définitions déjà formulées et utilisées dans le système PIVOINES. Pour plus de détails on fera référence à la thèse de Marion CREHANGE (cf(2)).

1.1.1 Donnée

Tout objet devant subir un traitement informatique est appelé donnée ou information. (Rappelons que nous employons sans distinction ces deux mots).

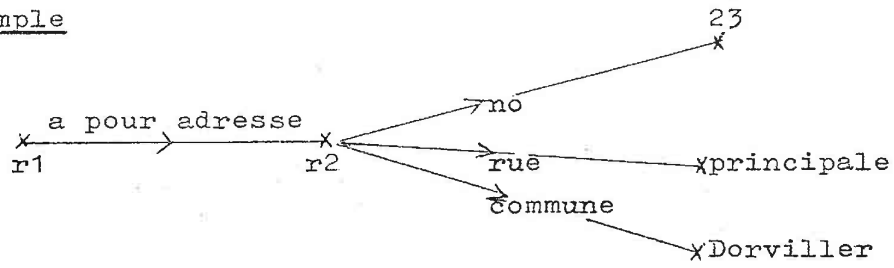
Une information peut être élémentaire, ou composée à partir d'informations élémentaires ayant des liens entre elles.

Il apparait très vite que la description de ces liens rend utile la notion de repère. Un repère peut être le but, l'origine, le "relais" dans un lien entre informations élémentaires. Son nom est arbitraire.

Cette notion de repère est à rapprocher de celle d'occurrence d'entité (cf (3)).

Donnons un exemple illustrant l'utilité des repères:

Exemple



"23", "principale", "Dorviller" sont associés au repère r2 respectivement par les relations no, rue, commune.

Une information c'est donc :

- un ensemble d'informations élémentaires.
- un ensemble fini de repères.
- un ensemble de liens élémentaires (que l'on pourra composer).

1.1.2 Structure de données

L'ensemble des caractéristiques qui font que, pour certains objectifs, les données les possédant se traitent pareillement sera appelé structure de données.

Les formalisations des structures de données sont nombreuses; elles ont la plupart du temps la même capacité d'expression.

Pour plus de détail sur ces différentes formalisations on fera référence au cours de C. DELOBEL (cf (3)).

Nous rappelons ici les principales caractéristiques du modèle proposé dans le projet PIVOINES, modèle qui d'ailleurs ne lui est pas propre:

- Le modèle est basé sur l'existence de relations d'accès logiques (indépendamment des accès physiques); on dispose d'accès logiques élémentaires pouvant se composer à l'aide d'une algèbre des accès.
- Les repères jouent un rôle important. Ils sont de différents types et sont utilisés dans le langage d'interrogation qui n'est pas un langage utilisant une écriture purement relationnelle.

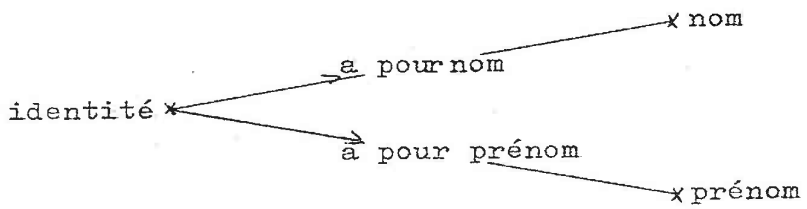
1.1.3 Structure logique et autres structures

La structure logique des données décrit la "façon de comprendre les données". Elle ne décrit ni la forme interne, ni la forme externe des données élémentaires. On peut bien sûr parler aussi d'une "structure interne" et d'une "structure externe" des données mais ces structures sont en général différentes de la structure logique des données. Nous reviendrons sur cette indépendance dans le paragraphe 1.4 mais dès maintenant nous l'illustrons par quelques exemples simples.

* Une même structure logique peut être associée à deux structures externes différentes.

Exemple :

Structure logique représentée par le schéma :



Structure externe n°1 : liste de noms suivie d'une liste de prénoms

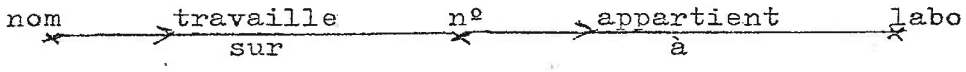
Structure externe n°2 : liste de couples (nom, prénom).

* A une même structure externe peuvent être associées deux structures logiques différentes.

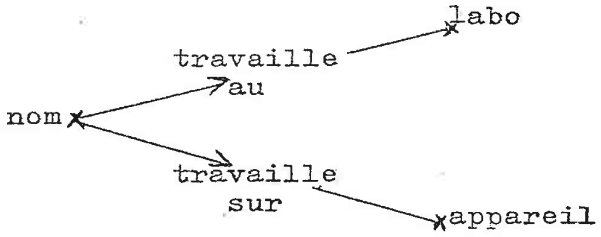
Exemple :

Structure externe : Identification d'un laboratoire suivie de noms de personnes y travaillant et de numéros d'appareils utilisés dans ce laboratoire.

Structure logique nº1 schématisée par :



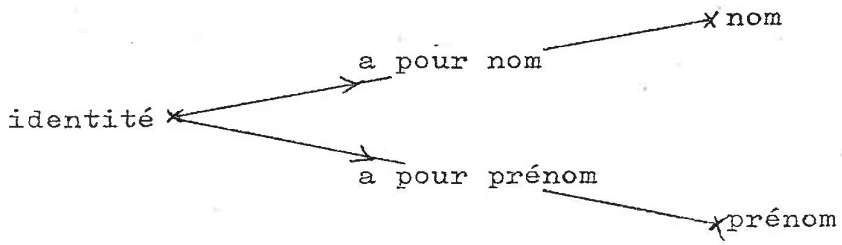
Structure logique nº2 schématisée par :



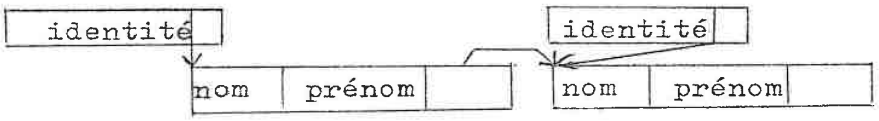
* A une même structure logique peuvent être associées deux représentations internes différentes.

Exemple :

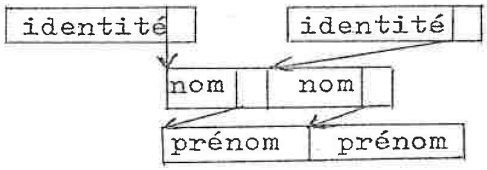
Structure logique schématisée par :



Représentation interne nº1 schématisée par :



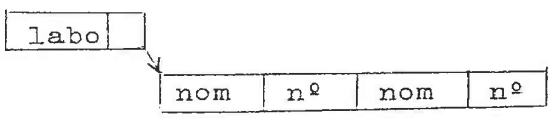
Représentation interne nº2 schématisée par :



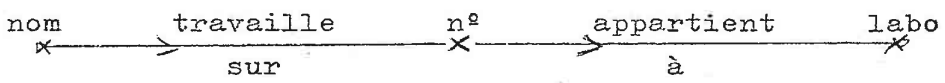
* A une même représentation interne peuvent être associées deux structures logiques différentes.

Exemple :

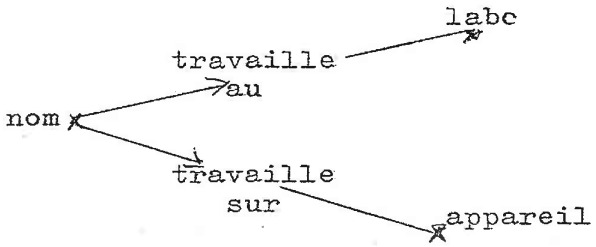
Représentation interne schématisée par



Structure logique nº1 schématisée par :



Structure logique nº2 schématisée par :



L'objet de travail de base est bien sûr l'information. Nous allons maintenant voir comment on peut la représenter en mémoire.

1.1.4 Représentation interne d'une information

Il nous faut ici faire référence à l'existence dans le projet PIVOINES (cf (ℓ)) d'une structure PHYLOG qui est la structure LOGique de l'organisation PHYsique interne de l'information. C'est une structure intermédiaire entre la structure logique et la représentation interne : elle "traduit" le fait que, parfois, la représentation "déforme la structure logique des données.

Pour représenter en mémoire l'information, il faut donc représenter en mémoire la structure PHYLOG. Il faut donc préciser comment représenter en mémoire les informations élémentaires, les repères, les relations élémentaires.

Une information élémentaire en mémoire est contenue dans une zone mémoire de taille mémoire plus ou moins grande. La taille de la zone mémoire est en général un sous-multiple ou un multiple d'un mot mémoire.

Accéder à des informations élémentaires pour éventuellement les traiter c'est donc accéder à des contenus de zone mémoire.

a) Les repères sont représentés en mémoire par des noms de zone mémoire.

b) Il faut bien distinguer les zones mémoire de leurs contenus. Citons deux exemples pour illustrer cette distinction:

* Une donnée élémentaire subit un traitement. Elle est transformée en une autre donnée élémentaire qui subit à nouveau un traitement (le même traitement que le précédent ou un traitement différent).

Il n'est pas nécessaire qu'à la fin des deux traitements la donnée élémentaire résultat du premier traitement soit conservée.

On peut donc définir une seule zone mémoire qui contiendra successivement la donnée élémentaire, la donnée élémentaire résultat du premier traitement, la donnée élémentaire résultat du deuxième traitement.

* Une même information élémentaire peut avoir été acquise à deux instants différents et devoir subir deux traitement différents, c'est-à-dire être associée à deux repères différents.

On doit alors définir deux zones mémoire distinctes bien que de contenus identiques.

Chaque traitement utilisera le contenu de l'une des zones mémoire et remplira cette zone mémoire par son résultat.

c) Nous avons vu (cf introduction) que les traitements que peut subir une information demandent d'y accéder.

Il faut donc aussi connaitre des accès dits élémentaires qui, éventuellement en les composant, permettront d'accéde à tous les éléments d'une information et donc de les traiter.Ces accès élémentaires permettront aussi de traduire tous les liens élémentaires et donc tous les

liens pouvant exister entre les données élémentaires.

Une information en mémoire peut donc être définie par:

- Un ensemble d'objets élémentaires: valeurs, noms de zones mémoire.
- Un ensemble d'accès élémentaires (ces accès pouvant être composés).

1.2 But de l'étude

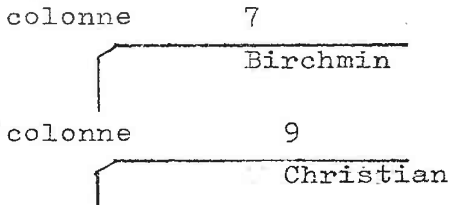
Acquérir l'information c'est transformer les objets la composant et se trouvant sur un support externe à la base en objets appartenant à un support interne à la base.

Remarquons que l'acquisition d'une information n'est qu'un cas particulier de l'adjonction à une information déjà acquise c'est-à-dire un cas particulier d'une modification.

Ici, avant d'aller plus loin dans la définition de l'acquisition il convient de bien préciser le langage que nous emploierons tout au long de notre travail.

Nous allons le faire à l'aide d'un exemple.

Soient deux cartes portant les informations suivantes :



* Connaître les données élémentaires c'est connaitre "Birchmin" et "Christian".
* Connaître la forme externe des données élémentaires c'est savoir :
  - sur une carte à partir de la colonne 7 il y a 8 lettres alphabétiques définissant une donnée.
  - sur la carte suivante à partir de la colonne 9 il y a 9 lettres alphabétiques définissant une donnée.
* Connaître la structure logique des données c'est savoir qu'une même personne est définie par un prénom et un nom.

* Connaître la donnée du point de vue logique c'est savoir qu'il est question d'une personne nommée Birchmin et prénommée Christian.
* Connaître la représentation interne des données élémentaires c'est savoir :
  - "Birchmin" est contenu dans un mot mémoire NOM (1) qui est le premier élément d'une liste de données élémentaires ayant la même représentation interne et définissant une liste de noms.
  - "Christian" est contenu dans un mot mémoire PRENOM (1) qui est le premier élément d'une liste de données élémentaires ayant la même représentation en mémoire et définissant une liste de prénoms.

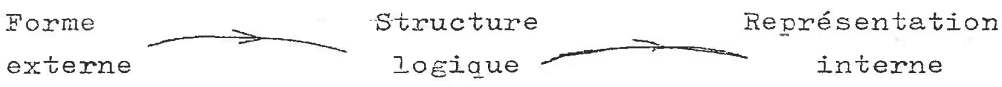
Remarque

Il ne faut donc pas confondre les données elles-mêmes et les renseignements que l'on peut avoir sur leurs formes externes, sur leurs structures logiques ou sur leurs représentations internes.

Acquérir l'information c'est introduire dans l'ordinateur l'information avec une certaine forme externe pour obtenir sa représentation interne.
Pour ce faire, nous avons choisi d'utiliser "la façon de comprendre les données" c'est-à-dire la structure logique des données.
Cet "écran" placé entre la forme externe et la forme interne des données nous apporte une souplesse d'utilisation dans la conception du programme d'acquisition.
En fait on part de la "façon de représenter les données sur des supports externes", on utilise la "façon de comprendre

les données" pour obtenir la "façon de représenter en mémoire les données".

Pour exprimer celà nous avons choisi de définir l'acquisition par une succession de correspondances qui peuvent se schématiser ainsi:



Ce schéma décrivant l'acquisition d'une information ne signifie nullement que pratiquement on aura deux étapes de transformation de l'information. Le but recherché est en fait un programme d'acquisition qui intégrerait les deux étapes, c'est-à-dire qui préciserait comment passer de la forme externe de l'information à sa représentation interne.

1.3 Les moyens

1.3.1 Les structures intermédiaires

Nous avons déjà parlé (paragraphe 1.1.4) de la structure PHYLOG, structure intermédiaire entre la structure logique et la représentation interne.

Si nous reprenons l'exemple étudié dans le paragraphe précédent (1.2) nous nous apercevons qu'il y a aussi une structure logique de la forme externe des données, différente de la structure logique des données, que l'on peut définir par les renseignements suivants:

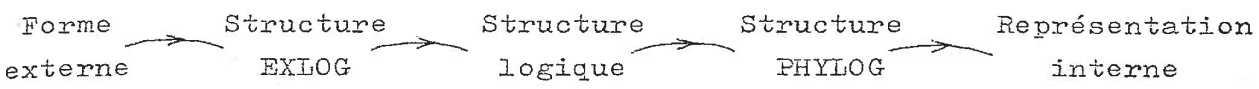
- Une donnée formée de caractères alphabétiques et se trouvant sur un support physique est un nom.
- Une donnée formée de caractères alphabétiques et se trouvant sur un autre support physique est un prénom. C'est le prénom associé au nom se trouvant sur le support physique précédent.

Nous allons donc définir une structure intermédiaire entre la forme externe de l'information et la structure logique.

Nous l'appellerons structure EXLOG c'est-à-dire la structure LOGique de la forme EXterne de l'information.

Ainsi apparaît une symétrie par rapport à la structure logique.

Le schéma représentant l'acquisition devient :



Il ne s'agit pas que l'acquisition se fasse avec quatre étapes de transformation de l'information.

Dans toute la suite de notre étude nous utiliserons une terminologie précise pour bien montrer que l'acquisition ne s'obtient pas en quatre étapes de transformation de l'information.

Des données se trouvent sur un support externe. Nous diron qu'une structure "affecte" des renseignements à ces donnée Nous dirons aussi que les données sont affectées par la structure considérée; par exemple quand nous parlerons des "données affectées par EXLOG" celà signifiera que l'on parle des données auxquelles on a ajouté tous les renseignements déduits de la description de la structure EXLOG sans que ces renseignements ne fassent subir une transformation aux données elles-mêmes.

Sur le schéma ci-dessus chaque flèche représente une phase

Le programme d'acquisition utilisera les descriptions des différentes phases qui sont des descriptions de correspondances entre structures.

Notre étude se place donc dans le cadre plus général des correspondances entre structures, c'est-à-dire de l'expression d'une structure en fonction d'une autre (mapping dans la documentation anglaise).

1.3.2 Le langage de description de la forme externe

Sur le support externe se trouvent des caractères. Ces caractères font partie d'un ensemble de caractères permis, qui peut lui-même être subdivisé en différents sous-ensembles.

Dans les langages de programmation (Fortran, Cobol) les différents renseignements sur les données se trouvant sur le support externe sont décrits dans les formats.

Une donnée élémentaire se trouvant sur le support externe peut être caractérisée par une ou plusieurs des caractéristiques suivantes:

- les caractères la formant
- le nombre de caractères la formant
- la position des caractères sur le support externe
- un ou plusieurs caractères qui précèdent ou suivent la donnée.

Certaines caractéristiques d'une donnée élémentaire peuvent exprimer un rôle particulier de cette donnée élémentaire par rapport aux autres données éléméntaires.

Par exemple une donnée élémentaire se trouvant seule sur un support externe définit la tête de liste d'une liste de données élémentaires se trouvant sur les supports externes suivants avec plusieurs données élémentaires sur un même support externe.

De même certaines caractéristiques de données élémentaires peuvent exprimer des relations entre ces données élémentaires Par exemple des données élémentaires qui se succèdent sur un même support sont les éléments d'une même liste.

Il faut donc pouvoir décrire beaucoup plus complètement que dans un format la forme externe de l'information. Nous avons pour cela un langage de description de la forme

externe qui permet de décrire les contenus des supports physiques et les relations pouvant exister entre ces contenus. Nous le présenterons en détail dans le chapitre 2.

Remarque : Description globale

Le langage de description de la forme externe va permettre de décrire globalement toute l'information en entrée.
Pour que cette description globale soit toujours possible nous allons émettre une hypothèse simplificatrice :
On suppose que l'on rentre à la fois toute l'information ou qu'une partie autonome de l'information peut être rentrée
On ne s'occupe pas de la décomposition de la chaîne de données en parties autonomes. On suppose que, si cette décomposition est possible, elle est faite au début avant toute autre procédure.

1.3.3 Déformation unique

Au départ, on connait donc la chaîne de données sur le support externe. On pourrait transformer plusieurs fois cette chaîne de données pour arriver à la forme interne de l'information.
En fait on garde la chaîne de données et en trois phases successives on affecte aux données les renseignements apportés par les structures.
Chaque phase peut être comparée à la fabrication d'un transparent que l'on collerait sur la chaîne de données en entrée
Un transparent est obtenu à l'issue d'une phase de traitemer portant sur la chaîne de données munie du transparent précédent.
Chaque phase a une ou plusieurs données et un ou plusieurs outils qui déterminent des actions dont le résultat est toujours d'affecter les données par la structure à laquelle

est liée la phase.

La déformation de la chaîne en entrée est unique. Elle s'effectue sur les données affectées par la structure PHYLO

1.3.4 Les trois phases et la déformation unique

Nous allons maintenant apporter des précisions sur les différentes phases et sur la déformation unique.

Pour chacune d'entre elles nous préciserons : ce que l'on a au départ, l'outil, "l'acteur" utilisant l'outil, le résultat de l'action de l'outil sur ce que l'on avait au départ.

Première phase

Ce que l'on a au départ	: Chaîne de données en entrée
Outil	: Langage de description de la forme externe
Acteur	: "Structureur"
Résultat	: "Données affectées par EXLOG'

Deuxième phase

Ce que l'on a au départ	: "Données affectées par EXLOG'
Outil	: Correspondane EXLOG → LOG
Acteur	: Programme de la 2º phase
Résultat	:"Données affectées par LOG"

Troisième phase

Ce que l'on a au départ	: "Données affectées par LOG"
Outil	: Correspondance LOG → PHYLOG
Acteur	: Programme de la 3º phase
Résultat	: "Données affectées par PHYLOG

Déformation unique

Ce que l'on a au départ : "Données affectées par PHYLOG

Outil : Procédures d'acquisition

Acteur : Programme de construction

Résultat : Données acquises

1.3.5 Les contrôles

Il y a tout d'abord les contrôles de conformité des données avec les descriptions écrites à l'aide du langage de description de la forme externe. Ces contrôles sont effectués dans la première phase avec le "structureur".

Il y a ensuite des contrôles de cohérence des données. Ces contrôles sont effectués dans chaque phase avec le programme correspondant.

Exemples : En entrée, on a une suite de 11 mesures. Dans la structure logique des données on sait que la 11ème mesure est la somme des 10 premières mesures.

a) Dans la description du langage de la forme externe on précise un encadrement pour les IO premières mesures.
Lorsque les données sont affectées par EXLOG, on peut faire des contrôles de conformité sur les IO premières mesures.
Lorsque les données sont affectées par LOG, on peut faire un contrôle de cohérence sur la 11ème mesure.

b) Dans la description du langage de la forme externe on précise un encadrement pour la 11ème mesure.
Lorsque les données sont affectées par EXLOG, on peut faire un contrôle de conformité sur la 11ème mesure.
Lorsque les données sont affectées par LOG, on peut faire des contrôles de cohérence sur les 10 premières mesures.

On voit, sur ces exemples, qu'à chaque phase il peut y avoir des contrôles de cohérence sur une ou plusieurs données en fonction des "affectations" dûes à la phase étudiée.

1.4 Avantages de la solution proposée

En terminant ce chapître insistons sur les deux principaux avantages que permet cette acquisition:

a) l'indépendance

La forme externe et la représentation interne des données sont indépendantes de la structure logique.

Ainsi, à une même information de forme externe définie, peuvent être associées deux structures logiques différentes. Dans ce cas, seule la première phase sera commune aux deux acquisitions.

De même la troisième phase peut être commune à deux acquisitions d'une information de même structure LOG et de même structure PHYLOG, mais de formes externes différentes.

b) la souplesse d'utilisation

Elle est essentiellement due à la déformation unique de la chaîne de données.

Prenons une comparaison pour justifier cette affirmation. Un tapissier-décorateur doit mettre à neuf une pièce. Plusieurs éléments entrent en ligne pour son travail:peinture, papier peint, rideaux, tissu d'ameublement. Il faut donc qu'il choisisse la peinture, le papier peint,les rideaux, le tissu d'ameublement en tenant compte de l'influence d'un élément sur les autres pour obtenir un ensemble harmonieux. Il accumule ainsi tous les matériaux nécessaires à la mise à neuf de la pièce. Ensuite il peut transformer la pièce.

Si, par contre, il commence par peindre puis qu'ensuite il place une tapisserie, s'il veut changer de tapisserie et que ce changement ne convient pas avec la peinture déjà faite, il lui faut repeindre...c'est-à-dire repartir de l'état initial de la pièce. Cette transformation de la pièce par étapes, risque donc de coûter cher.

De même, si l'on fait plusieurs déformations consécutives de la même chaîne de données, un changement dans l'une des structures oblige à revenir à des états précédents de la chaîne de données. Il faudrait alors conserver tous les états intermédiaires de la chaîne de données.
Cela coûterait cher en place mémoire.

Avec une déformation unique, un changement dans l'une des structures entraine seulement un changement dans les renseignements associés à la structure considérée. Les programmes ou les parties de programme utilisant ces renseignements seront aussi changés.

CHAPITRE 2

Le langage de description de la forme externe

Dans ce chapître nous allons préciser les caractères utilisables dans le langage puis la manière de décrire les données élémentaires et les relations.

2.1 Les caractères du langage

Le langage de description de la forme externe comprend les caractères suivants:

0 à 9	chiffres	caractères alphanumériques
A à Z	alphabet	
b	blanc	
+	signe d'addition	
-	signe de soustraction	
=	signe d'égalité	
*	signe de multiplication	
/	signe de division	
(	parenthèse gauche	
)	parenthèse droite	
[	crochet gauche	
]	crochet droit	
,	virgule	
;	point-virgule	
.	point	
:	deux points	
<	inférieur à	
>	supérieur à	
≤	inférieur ou égal à	
≥	supérieur ou égal à	

Certains caractères ont une signification particulière que nous verrons en détail au fur et à mesure de leur utilisation. La répétition de certains caractères peut aussi avoir une signification particulière.

2.2 Une description

Une description est une suite de descriptions de contenus de supports physiques et de descriptions de relations entre ces contenus.

Lorsque le changement de support physique est significatif, les descriptions des supports physiques sont séparées par le signe . (point).

A l'intérieur d'une description, il y a des descriptions élémentaires elles-mêmes séparées entre elles par le signe , (virgule).

Chaque description élémentaire décrit une ou plusieurs données élémentaires.

Quand toutes les descriptions des contenus de supports physiques sont faites, on décrit les relations entre ces contenus.

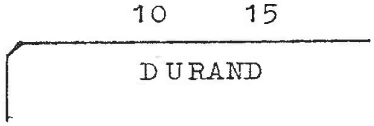
2.3 Les données élémentaires

Une donnée élémentaire est décrite par:

- son "type de valeur"
- son "type de repère"
- sa position sur le support physique

Donnons tout de suite un exemple pour préciser ces différentes caractéristiques d'une donnée élémentaire.

Soit le support physique



Le langage de description de la forme externe définira la donnée élémentaire se trouvant sur le support physique par le type de valeur: "des caractères alphabétiques" (en précisant éventuellement leur nombre: 6), et par le type de repère "nom".

2.3.1 Type de valeur

Définition

A chaque donnée de la chaîne de données est associé un type de valeur.

On peut définir un type de valeur par:

- une valeur
  ce peut être un ou plusieurs caractères
- un ensemble de valeurs
  il peut être défini comme:

1) Un ensemble de valeurs dit ensemble élémentaire de valeurs. Chaque ensemble élémentaire de valeurs est désigné par un nom. Nous avons défini les ensembles élémentaires de valeurs suivants:

   ALPH ensemble des suites de caractères alphabétiques

   SP ensemble des caractères spéciaux dont la liste est:

   {blanc, *, /, -, ≠, =, +, ≪, <, >, ≫, ,, ., ;, ., :, (,), ", !, ?, @, #, §}

   avec les sous-ensembles

   SPB = {blanc}

   SPOP = {*, /, -, +} signes d'opérations

   SPCO = {≠, =, <, ≪, >, ≫} signes de comparaison

   SPPO = {-, ;, ,, ., ., (,), ", !, :, ?} signes de ponctuation

   B ensemble des binaires

   CH ensemble des chiffres

   N ensemble des entiers naturels

   D ensemble des nombres décimaux

   R ensemble des réels

2) Un ensemble de valeurs construit à partir des ensembles élémentaires de valeurs et de valeurs, suivant la règle:

un nom d'ensemble de valeurs élémentaires, suivi d'un signe de comparaison, suivi d'une valeur.

Exemple : N $<$ 12

R $\geqslant$ 6

D $\neq$ 53

3) Un ensemble de valeurs construit à partir de types de valeurs (d'ensembles de valeurs et/ou de valeurs) à l'aide d'opérations.

Les opérations utilisables sont:

NON c'est un opérateur unaire; c'est l'opérateur de négation.

ET, OU, EX ce sont des opérateurs binaires: ET, OU, EX sont respectivement les opérateurs intersection réunion, différence symétrique.

REP c'est un opérateur unaire; c'est l'opérateur de répétition.

4) Si l'on fait précéder un type de valeur par un entier alors on aura autant de fois que cet entier des caractères de ce type de valeur.

Règles d'écriture

a) Tout type de valeur sera noté entre $\ll$ et $\gg$.

b) Toute valeur sera notée entre $\rightarrow$ et $\leftarrow$. Donc un type de valeur défini par la valeur * sera noté $\ll \rightarrow * \leftarrow \gg$.

c) Quand un type de valeur est défini avec des opérateurs on trouve des signes $\ll$ $\gg$ imbriqués.

Exemples

$\ll \rightarrow$ABC$\leftarrow \gg$: les caractères ABC

$\ll$NON$\ll \rightarrow$?$\leftarrow \gg \gg$: un caractère différent de ?

$\ll \ll$SPCO$\gg$ OU $\ll$SPOP$\gg \gg$: un caractère de comparaison ou d'opération

《《R《6》 ET 《R》3》》: un réel compris entre 3 et 6

《REP《SPPO》》 : des caractères de ponctuation

《ALPH》 : des caractères alphabétiques

《5 《ALPH》》 : 5 caractères alphabétiques

《←3 5→》 : les caractères 3 et 5

《N = 35》 : l'entier 35

2.3.2 Type de repère

Définition

A chaque donnée de la chaîne de données on doit associer un type de repére.

Un type de repére est un ensemble de repères auquel est associé un ensemble de données qui appartient à la chaîne de données.

Rang d'apparition

On peut déterminer une donnée par son rang d'apparition dans la chaîne de données.

- La valeur courante de la chaîne de données sera désignée par * .
- Certaines informations peuvent jouer un rôle de référence pour le rang d'apparition d'autres données.

  Exemple : Soient les trois données se trouvant séquentiellement sur un support: tulipe / orchidée

  tulipe et orchidée sont deux données de type de repère fleur; On peut définir :

  - tulipe comme étant la donnée de type de repère fleur se trouvant sur le support avant la donnée /.
  - orchidée comme étant la donnée de type de repère fleur se trouvant sur le support après la donnée /.

Par référence à la donnée / on peut donc définir le rang d'apparition d'autres données.
On dit que la donnée / joue ici un rôle de marque.

Toute donnée se trouvant sur le support externe peut définir une marque.

. Le rang d'apparition d'une donnée peut donc être défini par:

- la valeur courante * .
- des expressions élémentaires construites en utilisant la valeur courante et/ou des marques et/ou des compteurs associés à des répétitions.

Mal placé ici → "structureur"

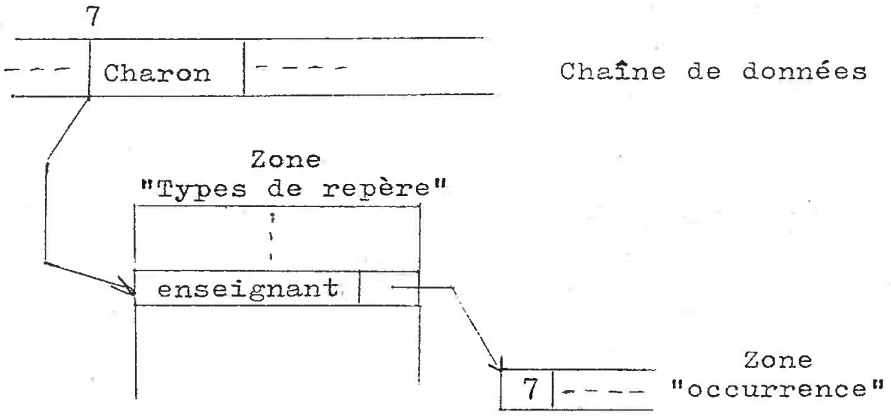
Zone "occurrence"

A chaque type de repère le "structureur" associe une zone mémoire appelée zone "occurrence". La zone "occurrence" est une liste d'éléments mémoire. Chaque élément mémoire contient l'entier ou les entiers qui permettent de définir un rang d'apparition. Ce dernier est bien sûr celui d'une donnée ayant le type de repère auquel la zone "occurrence" est associée.

Exemple : Soit le type de repère "enseignant".
Soit la donnée "Charon", élément de la chaîne de données en entrée, de type de repère enseignant.
Supposons que cette donnée ait comme "rang d'apparition" l'entier 7.
Alors on peut associer à un élément du type de repère enseignant son contenu: la donnée "Charon".
Si Charon est la première donnée de la liste de données de type de repère enseignant alors 7 sera le premier entier de la liste d'entiers

associée au type de repère enseignant, c'est-à-dire le contenu du premier élément de la zone "occurrence" associée au type de repère enseignant

On peut schématiser l'exemple par :



Au repère enseignant nº 1 du type de repère enseignant est associée la donnée nº 7 de la chaîne de données c'est-à-dire "Charon".

A partir des types de repère on peut construire des types de repère complexes; un type de repère complexe est formé de constituants.

Chaque constituant est défini par un type de repère ou un type de repère complexe (on impose un degré maximum d'imbrication: 10).

Les données associées aux constituants doivent se trouver séquentiellement sur un ou plusieurs supports physiques.

On définit deux types de repère complexes :

Groupe : Les constituants ont des types de repère (complexes ou non) différents.

Suite : Les constituants ont des types de repère (complexes ou non) identiques.

Règles d'écriture

1) Types de repère

- 1 à 30 caractères alphabétiques
- la chaine de caractères ne peut commencer par REL SU GR NON ET OU EX MA REP le caractère point .

2) Types de repère complexes

Groupe : 3 à 30 caractères alphanumériques commençant par GR

Suite : 3 à 30 caractères alphanumériques commençant par SU

Les descriptions des éléments d'un groupe ou d'une suite seront entourés par (et).

Exemple

(≪ALPH≫ nom, ≪SPB≫ sep , ≪ALPH≫ prénom) GRidentité

On définit un groupe identité formé d'une donnée formée de caractères alphabétiques et de type de repère nom et d'une donnée formée de caractères alphabétiques et de type de repère prénom qui sont séparés par un blanc de type de repère sep.

3) Marques

Pour différencier une marque d'une autre donnée on désignera son type de repère par une chaine de caractères alphanumériques (3-30) commençant par MA

Exemple

≪ALPH≫ expéditeur,≪→ /←≫ MA1,≪ALPH≫ destinataire

2.3.3 Position sur le support physique

Succession

Une donnée succède à une autre donnée si et seulement si leurs descriptions sont séparées par une , (virgule).

Exemple

≪ALPH≫ département, ≪N≫ code

Des caractères alphabétiques de type de repère département sont suivis par un entier de type de repère code.

Eventualité

L'existence d'une donnée ou de plusieurs données ainsi que le contenu d'un ou de plusieurs supports physiques peut être éventuelle.

Ce qui est éventuel est écrit entre [et].

Exemple

≪ALPH≫ nom, ≪N≫ codemarié [, ≪ALPH≫ nomépouse].

Le nom d'une épouse ne suit le nom de jeune fille d'une personne que si celle-ci est mariée.

dire, en rg, que l'on ne spécifie pas cette contrainte mais seul le caractère "éventuel"

Répétition

Une zone de répétition est décrite entre ((et)). Nous employons le mot zone car on peut répéter un type de repère simple et/ou un groupe et/ou une description de support physique et/ou une description.

Avant ((on peut préciser le nombre exact de répétitions.

Après ((on doit écrire un nom de compteur associé à la zone de répétition (chaîne de 15 caractères maximum ne commençant pas par REL SU GR NON OU ET EX MA REP .).

Un nom de compteur peut être associé à plusieurs zones de répétition. Ce peut être en particulier le nom d'un type de repère simple déjà défini. Par contre on ne permet pas de définir des compteurs par des expressions sur des types de repères ou des valeurs.

Exemple

a) répétition de groupe

2((compteur (≪ALPH≫ lieu, ≪N≫ quantité) grlivraison))

b) répétition de répétition et répétition de description de support physique

10((compteur1 ≪ALPH≫ rue.((compteur2 ≪ALPH≫ immeuble, ≪N≫ numéro.))))

c'est la description des supports physiques schématisés ainsi:

Support nº	Contenus	
1	rue_1	
2	$immeuble_{1,1}$	$numéro_{1,1}$
⋮	⋮	
x+1	$immeuble_{1,x}$	$numéro_{1,x}$
x+2	rue_2	
x+3	$immeuble_{2,1}$	$numéro_{2,1}$
⋮	⋮	
x+2+y	$immeuble_{2,y}$	$numéro_{2,y}$

Ici les nombres x, y sont définis par le compteur2

Changement de support

* Lorsqu'il est significatif, le changement de support est indiqué par un . (point).

* On peut avoir des données sur plusieurs supports physiques sans connaitre le nombre exact de données sur chaque support physique.

Le changement de support pourra alors être défini suivant des règles apportant des renseignements sur les données se trouvant au début et/ou à la fin des supports physiques Chacune de ces règles doit être associée à une zone répétition.

On considère les quatre règles suivantes:

Règle nº1 : on indique le nom du type de repère de la donnée devant se trouver au début de chaque support.

Règle n°2 : on indique le nom du type de repère de la donnée devant se trouver à la fin de chaque support.

Règle n°12 : on indique le nom du type de repère de la donnée devant se trouver au début de chaque support et le nom du type de repère de la donnée devant se trouver à la fin de chaque support.

Règle n°122: on indique le nom du type de repère de la donnée devant se trouver au début de chaque support et deux noms de type de repère qui forment un choix pour la donnée devant se trouver en fin de support.

Ecriture

La description de chaque règle se fait après la description de la zone répétition à laquelle elle est associée.
Chaque règle est décrite entre [n] et [n] où n désigne le numéro de la règle décrite (1 ou 2 ou 12 ou 122).
Entre [n] et [n] on écrit le nom (ou les noms) du (ou des type(s) de repère définissant la règle.
Pour indiquer le choix entre deux types de repère on entoure les deux noms de types de repère de (et).
Chaque nom de type de repère est séparé d'un autre nom de type de repère par une virgule.

Exemples

Soit la description d'une répétition de données:

((compteur1 ≪ALPH≫ nom, ≪ → * ←≫ sep, ≪ALPH≫prénom ≪→ * ←≫sep))

Si après cette description, on écrit:

a) [1] nom [1] alors on doit trouver au début de chaque support physique une donnée de type de repère nom.

b) [2] prénom [2] alors on doit trouver à la fin de chaque support physique une donnée de type de repère prénom.

c) [12] nom, prénom [12] alors on doit trouver au début de chaque support physique une donnée de type de repère nom et à la fin de chaque support physique une donnée de type de repère prénom.

d) [122] nom, (prénom, sep) [122] alors on doit trouver sur chaque support physique:
- au début une donnée de type de repère nom.
- à la fin soit une donnée de type de repère prénom, soit une donnée de type de repère sep.

Remarques

1) Il n'y a pas d'ambiguïté entre la description des règles de changement de support et la description d'une éventualité car l'un des entiers 1, 2, 12, 122 ne peut définir une donnée.

2) On n'a pas trouvé nécessaire de décrire dans les règles les séparateurs des données se trouvant sur les supports physiques. Celà aurait été redondant avec leurs descriptions dans les zones répétitions auxquelles sont attachées les règles.

3) Lorsqu'aucune règle n'est indiquée, les données se trouvent sur les supports physiques sans que l'on ait des renseignements précis sur les données pouvant se trouver au début et/ou à la fin des supports physiques. Des données peuvent donc notamment se trouver "à cheval" sur deux supports physiques (ce qui n'est pas le cas dans les différentes règles).

4) La règle 122 permet de supprimer un séparateur lorsqu'on arrive en fin de support; autrement dit, cette règle permet de considérer le changement de support comme un séparateur.
Ainsi dans l'exemple d) si à la fin d'un support on a un prénom alors le changement de support joue le rôle d'un séparateur et sur le support physique suivant on a un nom.

Remarques: ambiguïtés

1) Quand on a une répétition de données, il faut pouvoir différencier les données répétées des données suivantes dans le cas où l'on ne connait pas le nombre exact de répétitions.

Deux cas se présentent :

- les données de la zone de répétition et les données suivantes n'ont pas le même type de valeur alors on peut facilement les différencier.
- les données de la zone de répétition et les données suivantes ont le même type de valeur alors on ne peut les différencier; il faut que l'utilisateur sépare les données de la zone de répétition et les autres données.

Dans un premier temps, aux règles d'écriture d'une zone de répétition nous ajoutons la règle suivante : il faut définir une marque entre les données de la zone de répétition et les données suivantes dès que ces différentes données ont le même type de valeur. Cette marque ne devra pas être ambiguë avec la donnée du début de la zone de répétition.

2) Quand on a une éventualité, il faut pouvoir différencier les données de la zone éventualité des données suivantes. De même que précédemment, dans un premier temps, nous imposons donc la règle suivante : il faut définir une marque entre les données éventuelles et les données suivantes dès que ces différentes données ont le même type de valeur. Cette marque ne devra pas être ambiguë avec la donnée du début de la zone éventualité.

3) Malgré ces règles précises, il pourrait subsister une ambiguïté. Comment savoir si l'obtention d'un type de valeur différent de ceux rencontrés depuis le début de la répétition ou de l'éventualité signifie que la répétition est finie ou que l'éventualité est terminée, ou bien qu'il y a une erreur dans les données ? Pour enlever

cette ambiguïté nous étendons la règle d'utilisation des marques à toutes les répétitions dont on ne connait pas le nombre exact de répétitions et à toutes les éventualités et ceci sera la règle définitive.

Marque après une zone répétition ou une zone éventualité

Si nous voulons appliquer la règle de la remarque précédente pour toute base de données, il faudrait donc que figure dans la chaîne de données une donnée de type "marque Celà obligerait parfois à écrire à nouveau des données déjà existantes mais ne possédant pas de telles marques.
Pour éviter cet inconvénient et pour pouvoir prendre en compte des données déjà écrites sur des supports nous introduisons des marques qui correspondent à des parties de données existant sur le support physique.

Exemples

((compteur ≪ ALPH ≫ nom, ≪ N ≫ nb)), ≪ CH ≫ MA1, 1 ← ,
≪ N ≫ numéro.

≪ ALPH ≫ nom [, ≪ N ≫ nb] , ≪ 1 ≪ ALPH ≫ ≫ MA2, 1 ← ,
≪ ALPH ≫ nom.

Dans le premier exemple le premier chiffre de l'entier de type de repère numéro permet de différencier sans ambiguïté la donnée de type de repère numéro des données de la zone répétition.
Dans le deuxième exemple le premier caractère alphabétique de la donnée de type de repère nom permet de différencier cette donnée de la donnée de la zone éventualité.
Dans chaque exemple sur le support physique il n'y a, d'une part que la donnée de type de repère numéro, d'autre part que la donnée de type de repère nom : les données de types de repère MA1 et MA2 ne se trouvent pas sur le support physique.

Ecriture et Appellation

Lorsqu'on écrit n ⟵ où n est un entier naturel celà ne correspond pas à une donnée se trouvant sur le support. (Elle serait décrite entre ≪ et ≫). Celà correspond à une action à effectuer lors de la lecture de la chaîne de données : un "retour en arrière" d'un nombre de caractères égal à n.

On doit, après ce "retour en arrière", trouver dans la chaîne de données la donnée décrite après la virgule succédant à n ⟵ . Cette écriture sera appelée index de recul.

Dans le langage de description tel qu'il est conçu actuellement on ne peut décrire des formes externes conditionnelles. C'est un développement qu'il paraît intéressant d'envisager.

2.4 Les relations

2.4.1 Définition

On les définit par :

nom de relation : ensemble(s) de départ , ensemble d'arrivée , expression

- nom de relation : une chaîne de 15 caractères alphanumériques au maximum commençant par les lettres REL.
- ensemble(s) de départ } ce sont des types de repère
- ensemble d'arrivée } (complexes ou non) désignés par leurs noms.

Lorsqu'il y a n ensembles de départ, on les écrit sous forme de n-uple.

- expression : elle n'est pas obligatoire; elle permet de préciser éventuellement l'image de chaque élément de l'ensemble de départ.

2.4.2 Règles d'écriture des expressions

Les expressions qui apparaissent éventuellement dans la définition des relations sont construites à partir d'expressions élémentaires qui permettent de connaître le rang d'apparition des données.

Nous allons donc voir les règles d'écriture d'une expression élémentaire et celles d'une expression.

Pour définir une expression élémentaire et une expression, nous utiliserons les notations habituelles dans la définition de règles de grammaire et que nous rappelons ici brièvement :

Ecriture	Signification
ab	a suivi de b
[a]	éventuellement a
a|b	a ou b (ou exclusif)
a*	répétition de a (au moins une fois)
{a|b}c	pour noter ac | bc

Alors une expression élémentaire est définie par :

⟨expression élémentaire⟩ ::=⟨expression1⟩ | ⟨expression2⟩|const entière

⟨jalon⟩ ::= ⟨marque⟩ | * | ⟨compteur⟩

⟨expression1⟩::=(const.entière × ⟨compteur⟩)[⟨opération⟩ const.entière]

⟨expression2⟩::=⟨jalon⟩[⟨opération⟩ const.entière]

⟨marque⟩::= le nom du type de repère de la marque

⟨compteur⟩::= le nom d'un compteur associé à une répétition

⟨opération⟩::= +|−

Exemples

* + 2

MA1 − 5

(2xcompteur1) + 1

Une expression est alors définie par :

⟨expression⟩ ::= ⟨express⟩*

⟨express⟩ ::= {⟨expression élémentaire⟩ | ⟨suite⟩} , {⟨expression élémentaire⟩ | ⟨suite⟩} ;

⟨suite⟩ ::= (⟨expression élémentaire⟩ : ⟨expression élémentaire⟩ [: constante entière])

Exemples

1:10:2 signifie que l'on part du premier élément et l'on va jusqu'au dixième par pas de deux; c'est une ⟨suite⟩.

1:5:1 peut aussi se noter 1:5 le pas de 1 est facultatif; c'est aussi une ⟨suite⟩.

(1:10),(1:10); résume les expressions 1,1;2,2; ....;10,10;

SI l'on a une relation dont l'ensemble de départ est livre et l'ensemble d'arrivée est caractéristique, l'expression 1,(1:4);2,(5:8);3,(9:12) signifie:

Au premier livre sont associées les 4 premières caractéristiques

Au deuxième livre sont associées les 4 caractéristiques suivantes

Au troisième livre sont associées les 4 caractéristiques suivantes

Si l'on a une relation dont l'ensemble de départ est "distance" et dont l'ensemble d'arrivée est "prix", l'expression:

((ma1+1):(ma1+5)),((*-4):*); signifie que l'on a 5 couples (distance,prix) vérifiant la relation. Chacune des distances est définie par rapport à une marque. Chacun des prix est défini par rapport à la valeur courante.

Remarques

1) Chaque fois que nous spécifions un rang sur les données comme par exemple premier livre , cela signifie la donnée de type de repère livre ayant pour rang d'apparition dans la chaîne de données le contenu du premier

élément de la zone "occurence" associée au type de repère.

2) Dans les expressions, ce qu'il y a avant la virgule décrit toujours le "côté départ" d'une relation. Ce qu'il y a après la virgule décrit toujours le "côté arrivée".

2.5 Exemples d'utilisation du langage de description de la forme externe

1) Soit une section d'établissement scolaire formée de dix classes qui suivent le même type d'enseignement (même nombre de matières).

Pour chaque classe, on a les renseignements suivants:

- nombre d'élèves
- salles de classes occupées
- matières suivies

Pour chaque professeur enseignant dans cette section, on a les renseignements suivants:

- matière(s) enseignée(s)
- classe(s) enseignée(s)

On a ainsi une base de données à partir de laquelle on pourrait, en introduisant d'autres données (voeux des professeurs par exemple), étudier les problèmes de fabrication des emplois du temps d'un établissement scolaire.

Cette base de données peut être décrite sous différentes formes externes.

Supposons que nous ayons un paquet de cartes avec:

- sur la première carte: les caractéristiques d'une classe suivies du nombre d'élèves de cette classe.
- sur chacune des cartes suivantes: une matière enseignée dans cette classe, le numéro de la salle où elle est enseignée, le nom du professeur qui l'enseigne.

On peut schématiser cette forme externe ainsi:

Carte nº	Contenus
1	classe cl_1 / nb_1
2	$matière_{11}$ $salle_{11}$ $prof_{11}$
3	$matière_{12}$ $salle_{12}$ $prof_{12}$
⋮	⋮
k+1	$matière_{1k}$ $salle_{1k}$ $prof_{1k}$
k+2	*
k+3	classe cl_2 / nb_2
k+4	$matière_{21}$ $salle_{21}$ $prof_{21}$
⋮	⋮
2k+3	$matière_{2k}$ $salle_{2k}$ $prof_{2k}$
⋮	⋮

Ici dans chaque classe on a supposé qu'il y avait k matières enseignées.

Cette forme externe peut être décrite à l'aide du langage de description de la forme externe:

10((compteur1 ≪ALPH≫ classe,≪→/←≫ sep, ≪N≫ nombre.
((compteur2 ≪ALPH≫ matière, ≪SPB≫ se, ≪N≫ numéro, ≪SPB≫ se, ≪ALPH≫ prof.)) ≪→*←≫ ma.))

Relnombreélève classe,nombre (1:10),(1:10);

Relenseignant (classe,matière), prof

(1,(1:compteur2)),(1:compteur2);

(2,(compteur2+1:2xcompteur2)),(compteur2+1:2xcompteur2);

(10,((9xcompteur2)+1:10xcompteur2)),((9xcompteur2)+1
:10xcompteur2)

2) Nous prenons le même exemple mais cette fois-ci nous supposons que le nombre de matières enseignées n'est pas le même

pour toutes les classes. La forme externe peut alors être décrite par:

« ALPH » classe, « → /← » sep, « N » nombre.
((compteur1 « ALPH » matière, « SPB » se, « N » numéro, « SPB » se, « ALPH » prof.)) « →* ←» ma1.
« ALPH » classe, « → /← » sep, « N » nombre.
((compteur2 « ALPH » matière, « SPB » se, « N » numéro, « SPB » se, « ALPH » prof.)) « →*← » ma2.
⋮
« ALPH » classe, « → /← » sep, « N » nombre.
((compteur10 « ALPH » matière, « SPB » se, « N » numéro, « SPB » se, « ALPH » prof.)) « →*← » ma10.

Relnombreélève classe, nombre (1:10),(1:10);
Relenseignant (classe, matière), prof
(1,(1:compteur1)),(1:compteur1) ;
(2,(compteur1+1:compteur1+compteur2)),(compteur1+1:compteur1+compteur2) ;
⋮
$(10,(\sum_{i=1}^{9}$ compteur i+1: $\sum_{i=1}^{9}$ compteur i+compteur10)),($\sum_{i=1}^{9}$ compteur i+1: $\sum_{i=1}^{9}$ compteur i+compteur10);

Remarque

Dans les deux descriptions des deux formes externes, les descriptions des relations à l'aide d'expressions utilisant des compteurs paraissent bien lourdes.
On peut alléger l'écriture de ces expressions en utilisant les marques.

Les relations s'écrivent alors plus simplement :

```
Relenseignant (classe, matière), prof
(1,1:ma1),(1:ma1) ;
(2,ma1+1:ma2),(ma1+1:ma2) ;
    ⋮
(10,ma9+1:ma10),(ma9+1:ma10) ;
```

En conclusion rappelons que :

- L'utilisation des compteurs est obligatoire pour toute zone de répétition notamment pour permettre des contrôles.
- L'utilisation des marques est aussi obligatoire dans une répétition dès que le nombre de répétitions n'est pas connu. Il est recommandé d'utiliser les marques dans l'écriture des expressions des relations par souci de simplification.

CHAPITRE 3

Le "Structureur"

3.1 Définition

Le "structureur" est un outil qui, utilisant la description écrite à l'aide du langage de description de la forme externe et la chaîne de données en entrée, détermine les "données affectées par EXLOG".

3.2 Ce que fait le structureur

Dans ce paragraphe nous présentons les grandes lignes du "structureur". Nous l'étudierons plus en détail dans les paragraphes suivants.

3.2.1 Chronologie des actions

Aussi bien pour les descriptions de supports physiques que pour les descriptions de relations, nous avons le choix entre deux solutions pour la chronologie des actions.

1) interpréter un élément d'une description de support physique ou une description de relation, puis traiter immédiatement la donnée ou les données de la chaîne en entrée qui sont affectées par cette interprétation; répéter cette suite d'actions pour l'élément suivant de la description d'un support physique ou pour la description de la relation suivante.
2) interpréter toute la description, puis utiliser la structure EXLOG ainsi définie globalement pour traiter globalement toute la chaîne de données. C'est cette dernière hypothèse que nous retenons.

Nous rejetons la première hypothèse pour deux raisons essentielles:

- Si, pour une même description, on entre à nouveau une donnée il faudra tout recommencer si l'on suit la première chronologie d'actions.
- Une partie de la description n'est pas encore interprétée alors qu'on interpréte un élément d'une description. Celà peut entrainer des "retour en arrière" non seulement au niveau de l'interprétation (ce qui arrive même dans la deuxième hypothèse) mais aussi des "retour en arrière" au niveau du traitement des données.

3.2.2 L'interprétation de la description

On a une description de la forme externe des données, description écrite à l'aide du langage de description de la forme externe.

Le "structureur" interpréte cette description par une analyse et des actions associées aux résultats de cette analyse. L'analyse consiste, d'abord, à reconnaitre, à l'aide de la grammaire associée au langage, des symboles de ce langage.

Les résultats de l'analyse se traduisent par une série d'actions.

Ces actions peuvent être, par exemple: création d'une zone-mémoire, création d'un pointeur, incrémentation d'un compteur, remplissage d'une zone mémoire déjà existante..

Nous verrons plus tard en détail l'analyse puis les actions la traduisant.

Lorsque toute la description a été interprétée, on a en mémoire la représentation de la structure EXLOG. Il faut alors traiter les données pour les affecter par EXLOG.

3.2.3 Le traitement des données

Le traitement des données consiste à affecter les données par la structure EXLOG.

En mémoire, on a la représentation de la structure EXLOG ainsi que la chaîne de données; on veut attacher les renseignements de la structure EXLOG aux données.

Si nous reprenons la comparaison avec les transparents, on veut coller le transparent EXLOG sur la chaîne de données.

On parcourt la chaîne de données. On contrôle la cohérence de ce qui se trouve sur le support externe et des descriptions de données, descriptions définies dans la structure EXLOG. S'il y a cohérence, on attache les renseignements de la structure EXLOG aux données élémentaires considérées. L'affectation des données se fait au fur et à mesure du parcours de la chaîne de données. A la fin de la lecture de la chaîne de données on établit, d'après les relations d'EXLOG, les liens entre les données élémentaires affectées par EXLOG.

Nous verrons plus tard en détail le traitement des données. Avant de détailler, dans l'ordre chronologique, les actions du "structureur" nous allons voir ce que l'on obtient à la fin.

3.3 Ce que l'on obtient à la fin

Deux objectifs ont orienté la construction du résultat du "structureur", c'est-à-dire la représentation en mémoire des renseignements attachés aux données par une structure EXLOG. Il faut que, pour une donnée de la chaîne de données, on connaisse tous les renseignements qui lui sont attachés.

C'est le rôle premier du "structureur".

Il faut aussi qu'à partir d'un renseignement on puisse trouver la (ou les) donnée (s) de la chaîne de données ayant ce renseignement comme caractéristique.

Ainsi, par exemple, la connaissance d'un type de repère et de l'ensemble des données qui lui sont attachées sera utile notamment pour les correspondances entre structures.

Pour réaliser ces deux objectifs on crée des zones mémoire; chaque zone mémoire correspond à une caractéristique pouvant être attachée à une donnée.

Les zones mémoire créées dès le départ, quelle que soit la description à interpréter, sont :

1) zone mémoire "type de valeur"
2) zone mémoire "type de repère"
3) zone mémoire "occurrence"
4) zone mémoire "relations"
5) zones mémoire "départ" et "arrivée"
6) zones mémoire "traduction départ" et "traduction arrivée"

Chaque zone-mémoire est une liste d'éléments.

La taille-mémoire de chaque élément sera définie en unités mémoire. L'utilisateur pourra définir l'unité mémoire comme un 1/2 mot, un mot, un double-mot.

* Zone mémoire "type de valeur". élément / donnée

  Liste d'éléments contigus.

  Taille-mémoire de chaque élément : 3 unités mémoire.

  - la première unité contient des renseignements caractérisant le type de valeur.
  - la deuxième unité pointe vers la donnée de la chaîne de données ayant ce type de valeur.

- la troisième unité pointe vers le type de repère associé au type de valeur considéré c'est-à-dire le type de repère décrit après le type de valeur (on est bien sûr dans le cas où le type de repère n'est pas complexe).

* Zone mémoire "type de repère".

Liste d'éléments contigus.

Taille-mémoire de chaque élément : 2 unités mémoire + 1 bit

- la première unité contient des renseignements correspondant au nom du type de repère (complexe ou non).
- la deuxième unité pointe vers la liste des rangs d'apparitions, dans la chaîne de données, des données ayant ce type de repère (zone "occurrence").
- le bit supplémentaire est, au départ, mis à 1 pour tous les éléments de la zone "type de repère".

Remarque

Il n'y a pas redondance de pointeurs. En effet cette organisation des zones mémoire "type de valeur" et "type de repère" permet :
- d'associer à chaque donnée élémentaire son type de valeur et son type de repère.
- d'associer à un type de repère toutes les données élémentaires ayant ce type de repère.

* Zone mémoire "occurrence".

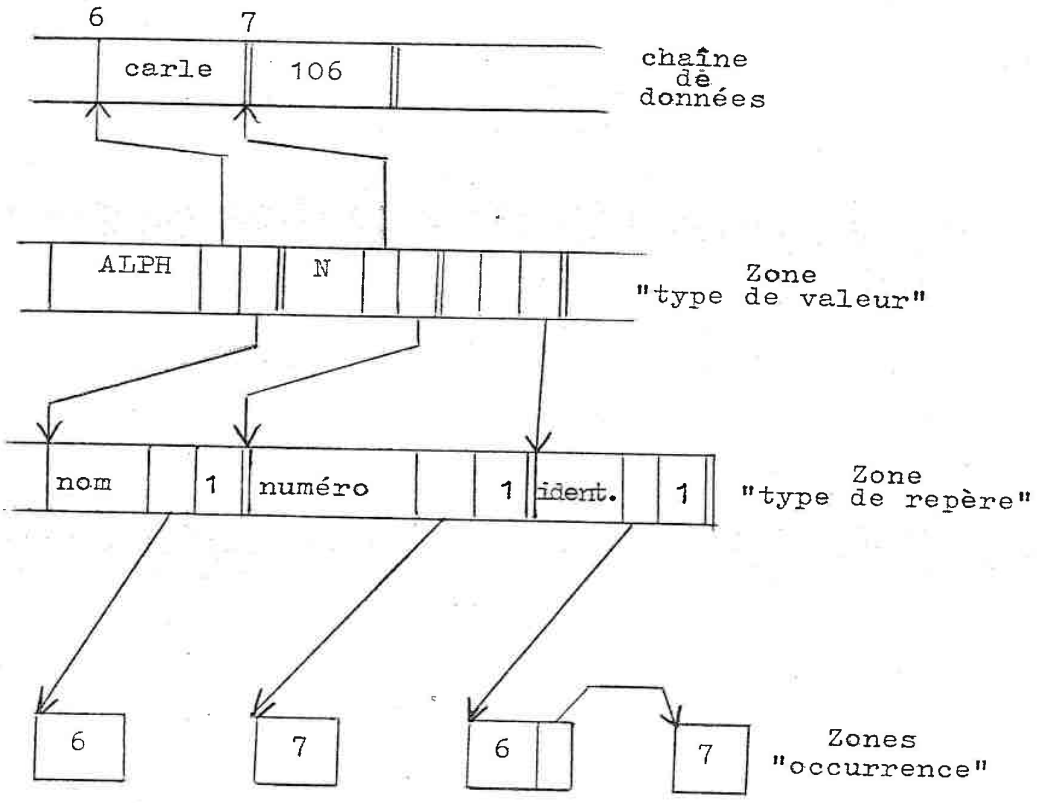
C'est une zone mémoire pointée par la 2ème unité d'un élément de la zone "type de repère".

Si la zone "occurrence" est associée à un type de repère, elle est formée d'une liste d'éléments contigus. Chaque élément est constitué d'une unité mémoire qui contient le rang d'apparition d'une donnée ayant le type de repère associé à la zone "occurrence".

Si la zone "occurrence" est associée à un type de repère complexe, elle est formée d'une liste d'éléments chaînés. Chaque élément a une taille mémoire fonction du nombre et de la nature des constituants du type de repère complexe.

Chaque unité mémoire contient le rang d'apparition d'une donnée ayant comme type de repère l'un des constituants du type de repère complexe.

Exemple



* Zone mémoire "relations".

Liste d'éléments contigus.

Taille-mémoire de chaque élément : variable, avec 3 unités-mémoire au minimum.

- la première unité contient des renseignements correspondant au nom de la relation.
- les unités suivantes sont des pointeurs vers des éléments de la zone départ ou de la zone arrivée.

* Zones mémoire "départ" et "arrivée".

  Listes d'éléments contigus.

  Taille-mémoire de chaque élément : 2 unités-mémoire.

  - la première unité contient des renseignements correspondant au nom du type de repère qui est un ensemble de départ (respectivement un ensemble d'arrivée).
  - la deuxième unité est un pointeur vers un élément de la zone "traduction départ" (respectivement "traduction arrivée").

* Zones mémoire "traduction départ" et "traduction arrivée".

  Listes d'éléments chaînés.

  Taille-mémoire de chaque élément : variable.

  Le contenu de chaque élément est déduit de ce qui, dans l'expression décrivant la relation, est attaché à l'ensemble de départ (respectivement l'ensemble d'arrivée).

  Si l'expression associée à une relation est constituée de valeurs entières, alors les zones mémoire "traduction départ" et "traduction arrivée" seront des listes de valeurs entières chaînées.

  Si l'expression associée à une relation est définie avec des marques, alors dans les zones mémoire "traduction départ" et "traduction arrivée" on trouvera les rangs d'apparition de ces marques.

  Si l'expression associée à une relation est définie avec des compteurs, alors dans les zones mémoire "traduction départ" et "traduction arrivée" on trouvera le nombre de répétitions auquel est associé ce compteur.

D'autres zones mémoire peuvent être créées. Nous les étudierons lorsque nous verrons en détail l'interprétation d'une description.

Pour mieux montrer ce que l'on obtient à la fin des actions du "structureur", c'est-à-dire après l'interprétation de la description et le traitement des données, voici un exemple:

Exemple

Soit la description :

≪ALPH≫ service.≪8 ≪ALPH≫≫ employé, ≪→/←≫ sep, ≪ N ≫ codequalif, ≪8 ≪ALPH ≫≫employé, ≪→/←≫ sep, ≪N ≫ codequalif. ≪8 ≪ALPH ≫≫employé, ≪→/←≫ sep, ≪N ≫ codequalif.

relemploi : (service, codequalif), employé

(1,(1:3)),(1:3);

Soit la chaîne de données formée des contenus des trois cartes suivantes :

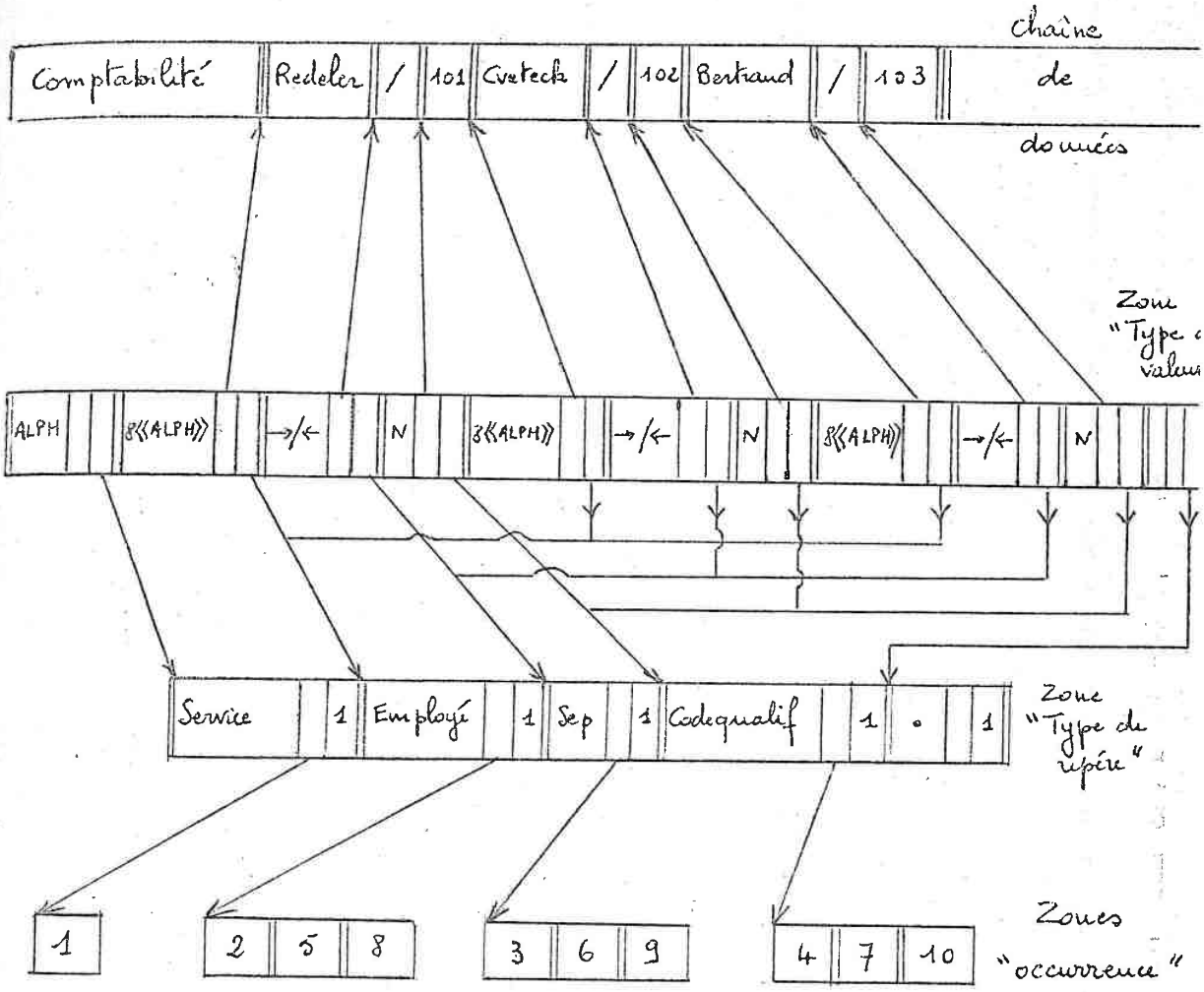
Carte nº 1 comptabilité

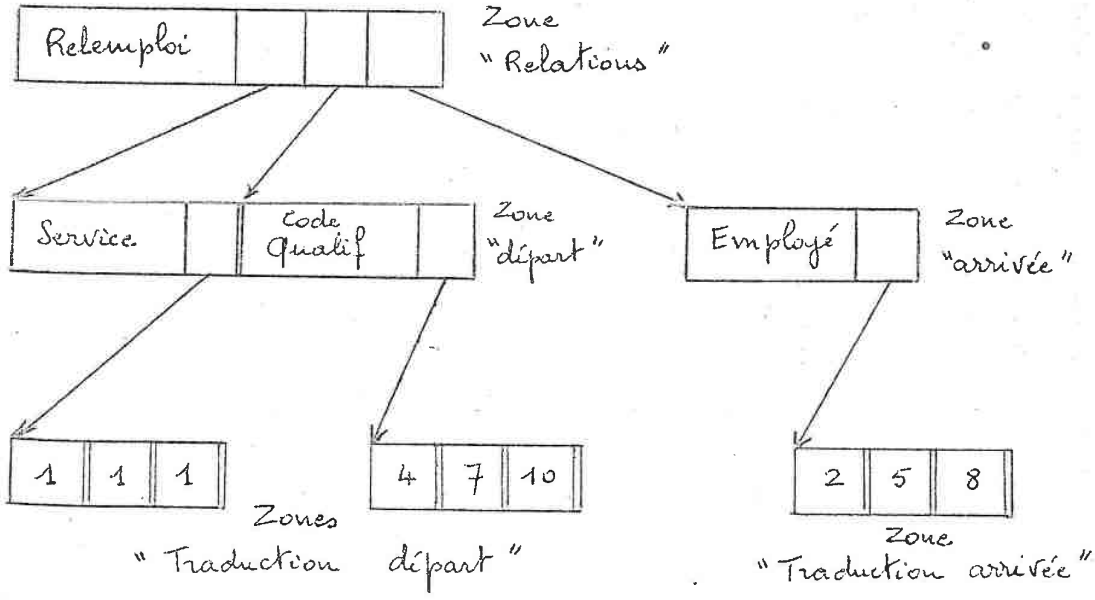
Carte nº 2 Redeler/101 Cveteck/102

Carte nº 3 Bertrand/I03

Pour présenter le résultat du "structureur" nous utiliserons les conventions suivantes :

- une suite ordonnée d'éléments sera représentée entre deux traits horizontaux.
- un trait vertical séparera deux parties d'un même élément.
- deux traits verticaux sépareront deux éléments.
- trois traits verticaux, dans la chaîne de données, symboliseront un changement de support.
- une flèche symbolisera un pointeur.
- les différents éléments de la chaîne de données seront numérotés dans l'ordre croissant.


Chaîne de données
Comptabilité
Redelec
/
101
Cvetech
/
102
Bertaud
/
103
Zone "Type
ALPH
8(ALPH)
→/←
N
Service
1
Employé
Sep
Codequalif
Zone "Type de rupte"
Zones "occurrence"


Relemploi
Zone "Relations"
Service
Code Qualif
Zone "départ"
Employé
Zone "arrivée"
Zones "Traduction départ"
Zone "Traduction arrivée"

3.4 Comment on l'obtient

Nous allons voir, en détail, dans l'ordre chronologique, les actions du "structureur".

3.4.1 L'analyse d'une description

Elle a un double rôle :

- vérifier la conformité des écritures des descriptions avec les règles d'écriture du langage de description de la forme externe (analyse lexicographique et syntaxique).
- reconnaître des symboles du langage de description de la forme externe et en déduire des caractéristiques des données (analyse sémantique).

L'analyse d'une description utilise les règles de la grammaire du langage de description ainsi que les automates permettant de reconnaître les symboles de ce langage.

A - Grammaire du Langage

Nos conventions d'écriture des règles de la grammaire du langage seront, pour la plupart, celles utilisées couramment. Rappelons leurs significations

a^*	répétition de a (au moins 1 fois)
a^n	répétition n fois de a
a_n	répétition au plus n fois de a
$[a]$	éventuellement a
$a \mid b$	a ou b (ou exclusif)
ab	a suivi de b
$ac \mid bc$	noté encore $\{a \mid b\}\, c$

On notera les caractères du vocabulaire terminal en caractères gras.

Avec ces conventions les règles de la grammaire s'écrivent :

⟨description générale⟩ ::= {⟨description support⟩|⟨entier⟩((⟨compteur⟩ ⟨description support⟩))|((⟨compteur⟩⟨description support⟩))⟨donnée marque⟩|[⟨entier⟩]((⟨compteur⟩⟨description règle⟩))⟨règle⟩|⟨description support⟩[⟨description support⟩]⟨donnée marque⟩}* ⟨description rela⟩*

⟨description support⟩ ::= ⟨description règle⟩ •

⟨description règle⟩ ::= ⟨description⟩|⟨entier⟩((⟨compteur⟩⟨description⟩))|((⟨compteur⟩⟨description⟩))⟨donnée marque⟩

⟨description⟩ ::= ⟨description donnée⟩[,⟨description donnée⟩]*|⟨description donnée⟩[,⟨description⟩]⟨donnée marque⟩|⟨description⟩[,⟨description⟩]*

⟨description donnée⟩ ::= ⟨type de valeur⟩⟨type de repère simple⟩|⟨donnée marque⟩((⟨type de valeur⟩⟨type de repère⟩,⟨type de valeur⟩⟨type de repère⟩[,⟨type de valeur⟩⟨type de repère⟩]*)⟨type de repère complexe⟩

⟨compteur⟩ ::= ⟨caractère⟩ 15

⟨donnée marque⟩ ::= ⟨type de valeur⟩ ⟨marque⟩

⟨marque⟩ ::= MA⟨caractère⟩ 28[, ⟨entier⟩ ←]

⟨règle⟩ ::= [1]⟨type de repère⟩[1]|[2]⟨type de repère⟩[2]|[12]⟨type de repère⟩,⟨type de repère⟩[12]|[122]⟨type de repère⟩(⟨type de repère⟩,⟨type de repère⟩)[122]

⟨type de valeur⟩ ::= ((→⟨valeur⟩←|⟨ensemble élémentaire⟩|⟨ensemble numérique⟩⟨opération⟩⟨valeur numérique⟩|NON⟨type de valeur⟩|REP⟨type de valeur⟩|⟨entier⟩⟨type de valeur⟩|⟨type de valeur⟩{ET|OU|EX}⟨type de valeur⟩

⟨valeur⟩ ::= ⟨caractère⟩*

⟨ensemble élémentaire⟩ ::= ALPH | SP | SPB | SPOP | SPPO | SPCO | CH | N | B | D | R

⟨ensemble numérique⟩ ::= CH | N | B | D | R

⟨opération⟩ ::= < | ≤ | = | ≥ | > | ≠

⟨valeur numérique⟩ ::= ⟨caractère numérique⟩*

⟨entier⟩ ::= ⟨chiffre non nul⟩⟨chiffre⟩*

⟨chiffre non nul⟩ ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

⟨chiffre⟩ ::= 0 | ⟨chiffre non nul⟩

⟨type de repère⟩ ::= ⟨type de repère simple⟩ | ⟨type de repère complexe⟩

⟨type de repère simple⟩ ::= 30 caractères au maximum ne commençant pas par REL SU GR NON ET OU EX MA REP .

⟨type de repère complexe⟩ ::= {SU | GR} ⟨caractère⟩$_{28}$

⟨description rela⟩ ::= ⟨nom relation⟩ { ⟨type de repère⟩ | (⟨type de repère⟩, ⟨type de repère⟩ [, ⟨type de repère⟩]*) }, ⟨type de repère⟩ [, ⟨expression⟩]

⟨nom relation⟩ ::= REL ⟨caractère⟩$_{12}$

⟨expression⟩ ::= ⟨express⟩*

⟨express⟩ ::= { ⟨expression élémentaire⟩ | ⟨suite⟩ }, { ⟨expression élémentaire⟩ | ⟨suite⟩ } ;

⟨suite⟩ ::= (⟨expression élémentaire⟩ : ⟨expression élémentaire⟩ [: ⟨constante entière⟩])

⟨expression élémentaire⟩ ::= ⟨expression1⟩ | ⟨expression2⟩ | ⟨entier⟩

⟨expression 1⟩ ::= (⟨entier⟩ × ⟨compteur⟩) [⟨opération⟩ ⟨entier⟩]

⟨expression 2⟩ ::= ⟨jalon⟩ [⟨opération⟩ ⟨entier⟩]

⟨opération⟩ ::= + | −

⟨jalon⟩ ::= ⟨marque⟩ | * | ⟨compteur⟩

⟨caractère⟩ est l'un des caractères utilisable dans le langage et dont la liste est donnée au paragraphe 2.1 du chapitre 2

⟨caractère numérique⟩ est l'un des caractères constituant les éléments des ensembles CH, N, B, D, R.

Lorsque dans la description d'une relation il y a n ensembles de départ alors il y a n + 1 parties dans l'expression qui peut être associée à cette relation.

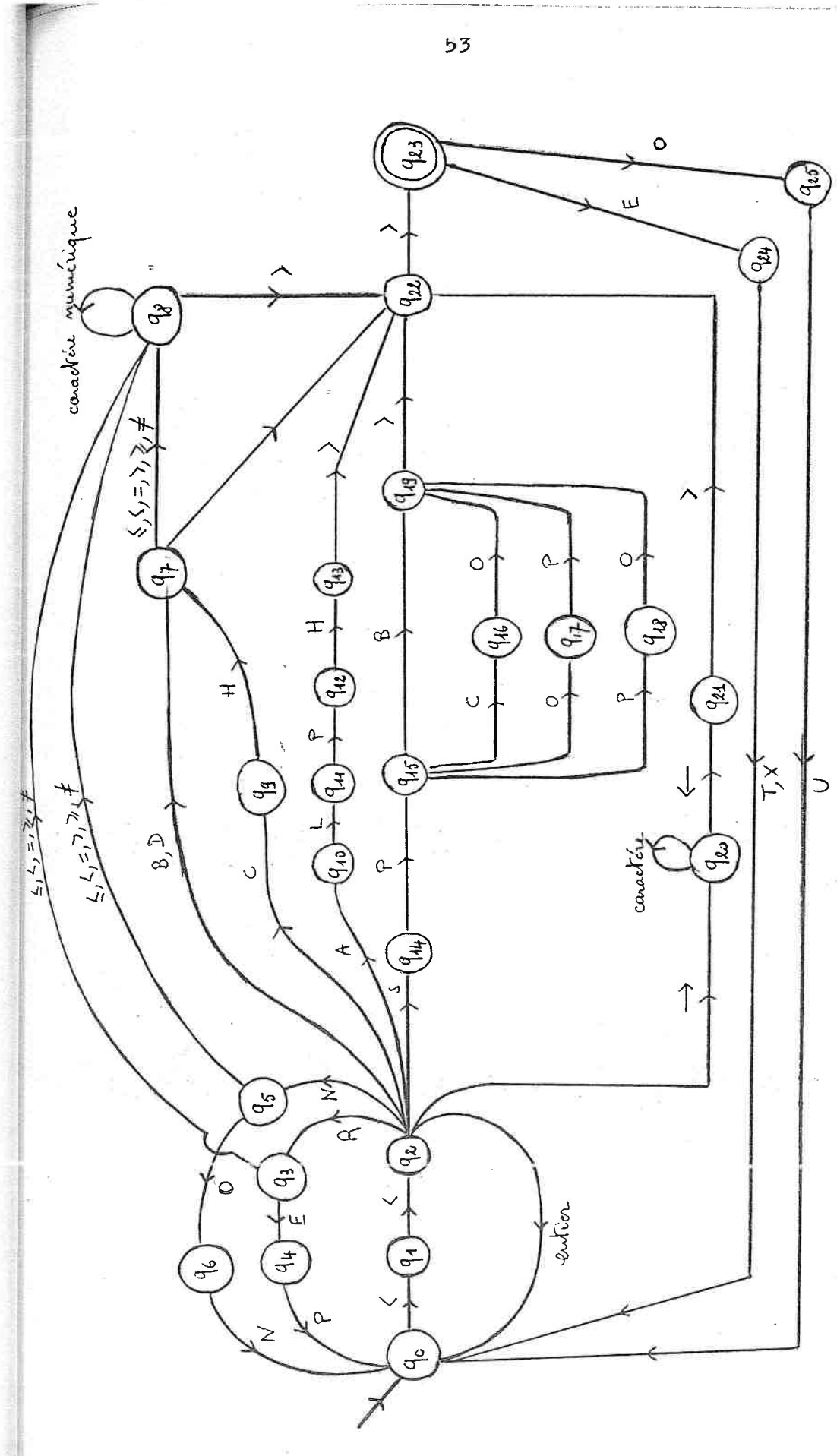
B - Automates reconnaissant les symboles du langage

Sans entrer dans le détail de tous les automates permettant la reconnaïssance de tous les symboles du langage nous donnons en exemple l'automate permettant de reconnaître un type de valeur.

Nous en faisons une description avec diagramme.

Rappelons les notations habituelles (cf (1)):

- un cercle définit un état
- une flèche vers un cercle définit l'état initial
- deux cercles concentriques définissent l'état terminal


caractère numérique
<, >, =, ≥, ≠
<, >, =, ≥, ≠
B, D
H
C
A
P
L
H
B
C
O
P
O
P
O
P
R
N
O
E
N
P
<
<
action
caractère
T, X
U
E
O

Ainsi ≪NON ≪ → /←≫≫ est lu par l'automate car il existe une suite d'états q_1, q_2, q_5, q_6, q_0, q_1, q_2, q_{20}, q_{20}, q_{21}, q_{22}, q_{23} avec

$q_1 = \delta(q_0, <)$
$q_2 = \delta(q_1, <)$
$q_5 = \delta(q_2, N)$
$q_6 = \delta(q_5, O)$
$q_0 = \delta(q_6, N)$
$q_1 = \delta(q_0, <)$
$q_2 = \delta(q_1, <)$
$q_{20} = \delta(q_2, \rightarrow)$
$q_{20} = \delta(q_{20}, /)$
$q_{21} = \delta(q_{20}, \leftarrow)$
$q_{22} = \delta(q_{21}, >)$
$q_{23} = \delta(q_{22}, >)$

q_0 est l'état initial
q_{23} est l'état final
δ désigne la fonction de transition associée à l'automate.

A l'aide des automates associés à la grammaire du langage, l'analyse de la description étant supposée faite, nous allons maintenant voir les actions associées à chaque reconnaissance d'un symbole du langage de description de la forme externe.

3.4.2 Les actions traduisant les résultats de l'analyse

- Si l'analyse reconnaît un type de valeur, alors :

on remplit la première unité de l'élément courant de la zone "type de valeur" par des renseignements représentant les caractères se trouvant entre ≪ et ≫ .

- Si l'analyse reconnaît un nom de type de repère associé à un type de valeur, (c'est-à-dire un nom de type de

repère non complexe) et ne définissant pas un constituant d'un type de repère complexe, alors :

on commence par chercher le nom du type de repère dans les premières unités des éléments déjà existants de la zone "type de repère".

a) Si le type de repère existe déjà.
La 3ème unité du type de valeur associé est pointée vers ce type de repère.

b) Si le type de repère n'existe pas encore.
On crée un nouvel élément de la zone "type de repère'
La première unité contient le nom du repère. Cet élément est pointé par la 3ème unité du type de valeur associé (pour être vraiment précis il faudrait écrire: "la 3ème unité de l'élément de la zone "type de valeur" dont la 1ère unité décrit le type de valeur associé, dans la description du type de repère considéré").

Remarque : Quand on parle du type de valeur associé au type de repère, celà signifie, bien sûr, que le type de valeur est celui décrit juste avant le nom du repère, c'est donc toujours le dernier élément connu de la zone "type de valeur".

. Si l'analyse reconnaît la parenthèse (qui définit le début de la description des constituants d'un type de repère complexe , alors :

on crée un élément de la zone "type de valeur".
la première unité est vide.
la deuxième unité pointe vers une zone "occurrence".
la troisième unité pointe vers une zone "constituant"

qui est formée d'éléments qui contiendront les noms des types de repère associés aux constituants du type de repère complexe.

- Si l'analyse reconnaît un nom de type de repère associé à un type de valeur, (c'est-à-dire un nom de type de repère non complexe) et définissant un constituant d'un type de repère complexe, alors :

  Les actions traduisant le résultat de cette analyse sont identiques à celles associées à un nom de type de repère, associé à un type de valeur, et ne définissant pas un constituant d'un type de repère complexe. De plus le type de repère est inscrit dans l'élément courant de la zone "constituant" associé à la parenthèse (définissant le début de la description d'un type de repère complexe .

- Si l'analyse reconnaît un type de repère complexe, alors

  On cherche le nom du type de repère complexe dans les premières unités des éléments déjà existants de la zone "type de repère".
  Si le type de repère n'existe pas encore :
  On crée un nouvel élément de la zone "type de repère". Cet élément est pointé par la troisième unité d'un élément de la zone "type de valeur". Ce dernier a sa première unité vide; sa deuxième unité ne pointe vers aucune donnée de la chaîne de données (par définition une seule donnée ne peut être associée à un type de repère complexe).

. Si l'analyse reconnaît une marque, alors :

La traduction est totalement identique à celle d'un type de repère.

. Si l'analyse reconnaît un index de recul, alors :

On remplit la deuxième unité de l'élément courant de la zone "type de valeur" par l'entier définissant l'index de recul.
Ni la première unité, ni la troisième unité de l'élément courant de la zone "type de valeur" ne sont utilisées. En effet à un index de recul ne sont associés ni un type de valeur, ni un type de repère.

. Si l'analyse reconnaît un changement de support, alors

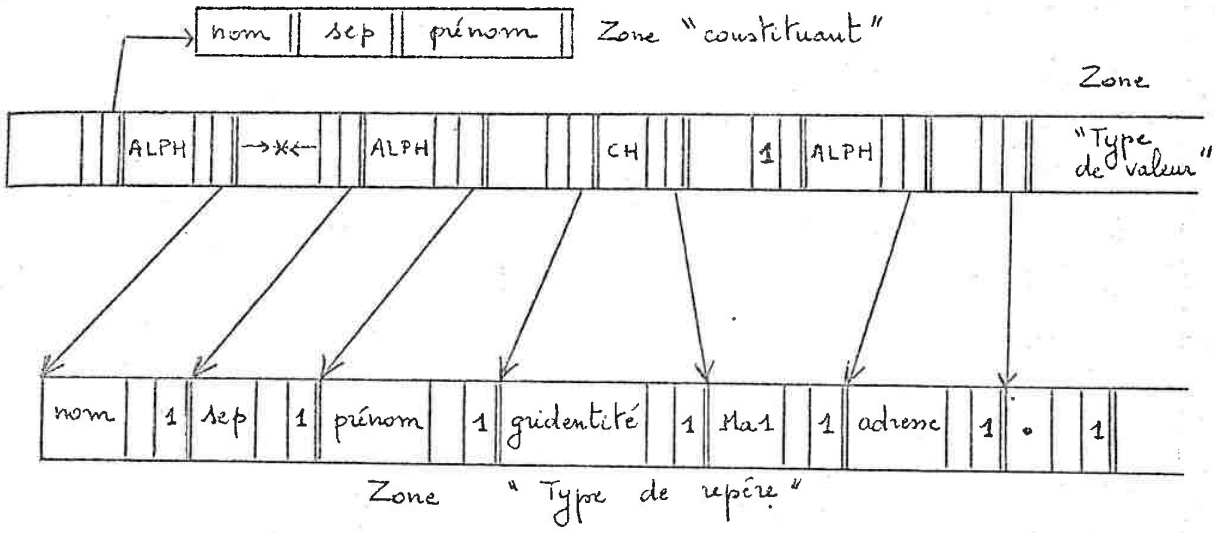
On cherche dans la zone "type de repère" un élément dont la première unité contient des renseignements représentant le caractère point.
Si cet élément n'existe pas encore :
On en crée un nouveau. Cet élément est pointé par la troisième unité d'un élément de la zone "type de valeur". Ce dernier a sa première unité vide; sa deuxième unité ne pointe vers aucune donnée de la chaîne de données.

Exemple

Soit la description :

(≪ALPH≫nom, ≪→ * ←≫sep, ≪ALPH≫ prénom)GRidentité, ≪CH≫Ma1, 1←,
≪ALPH≫ adresse.

L'analyse de cette description se traduit par:



. Si l'analyse reconnaît une zone de répétition, alors:

Un élément compteur est créé dans la zone "type de valeur".

Il est formé de 3 unités mémoire:

- la première unité contient le nom du compteur.
- la deuxième unité est un pointeur vers un buffer de répétition.
- la troisième unité est un pointeur vers une zone "vérification".

Le buffer de répétition est formé d'éléments identiques aux éléments de la zone "type de valeur". Ce sont les traductions des types de valeurs décrits entre ((et)). Les types de repère, auxquels ils sont associés sont traduits comme indiqué en 3.4.2 pages 54 - 55.

La zone "vérification" est formée de 3 unités mémoire. La première unité contient le nombre d'éléments déduit du nombre de types de valeur du buffer de répétition

et de leurs genres (dans le cas d'une répétition de répétition, dans le buffer de répétition on a un élément compteur auquel peuvent être associés plusieurs types de valeur décrits dans un autre buffer de répétition; dans le cas d'une répétition avec éventualité, dans le buffer de répétition on a un élément "éventualité" auquel peuvent être associés plusieurs types de valeur décrits dans une zone "éventualité").

Dans la deuxième unité se trouve, éventuellement, le nombre exact de répétitions tel qu'il peut être défini dans la description.

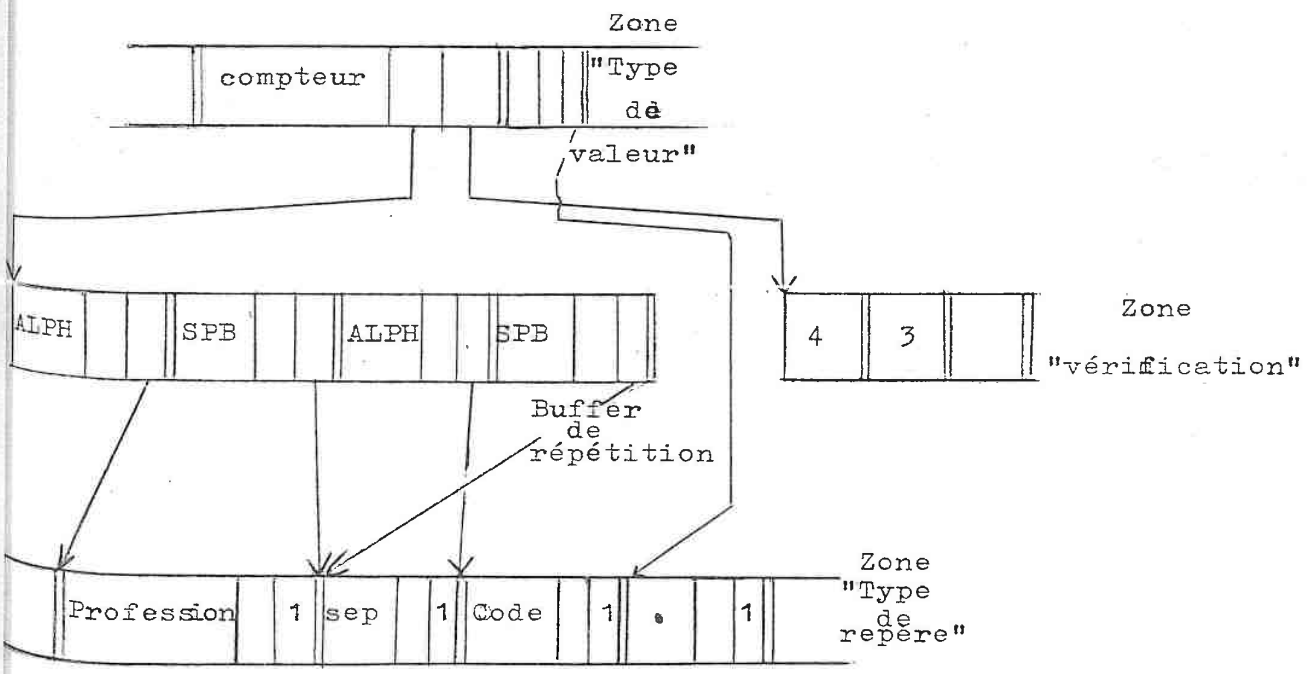
La troisième unité sera utilisée lors du traitement. Son contenu est initialisé à 1.

Ici aussi nous illustrons la traduction d'une zone de répétition par des exemples.

1) Soit la description :

3((compteur《ALPH》profession,《SPB》sep,《ALPH》code,《SPB》sep)).

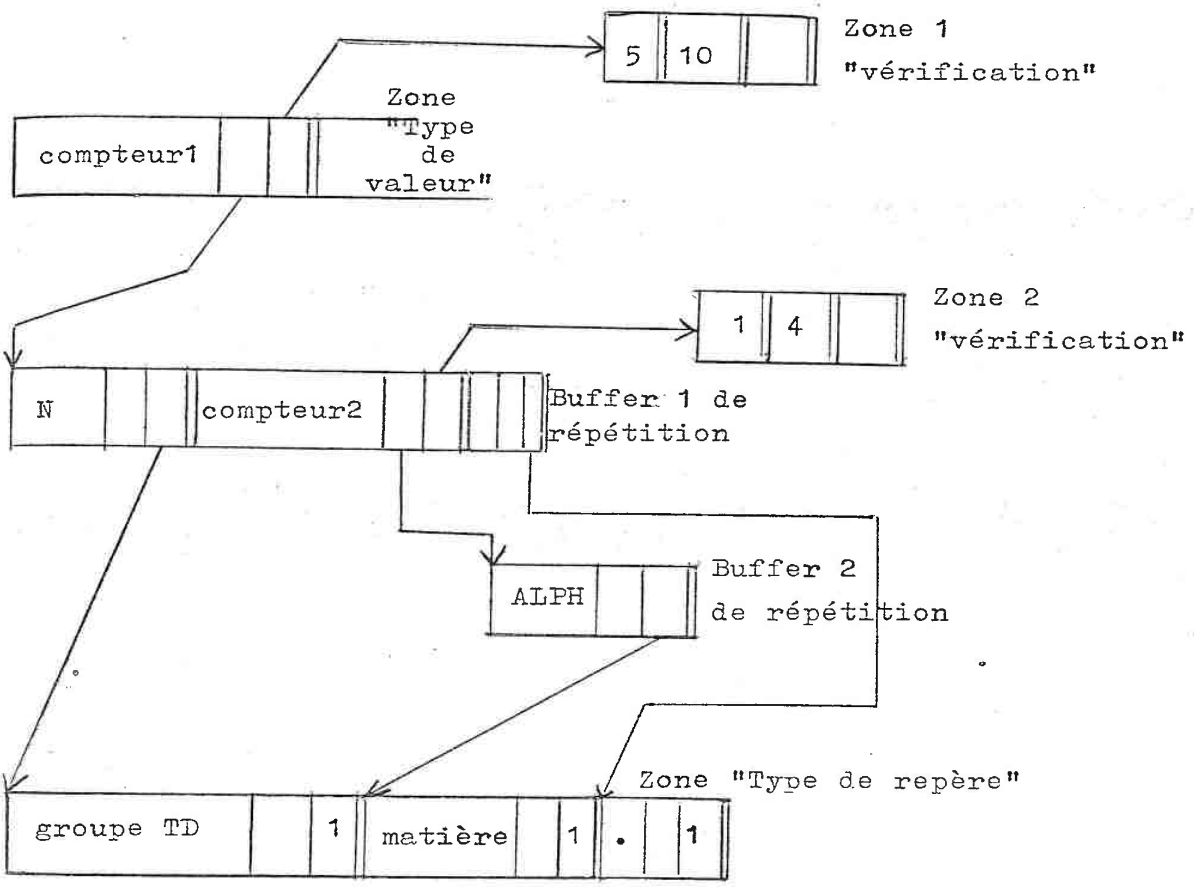
Après analyse, la traduction pourra être représentée en mémoire par :



2) Soit la description :

10((compteur1《N》groupe TD, 4((compteur2《ALPH》 matière)).))

Après analyse la traduction pourra être représentée en mémoire par :

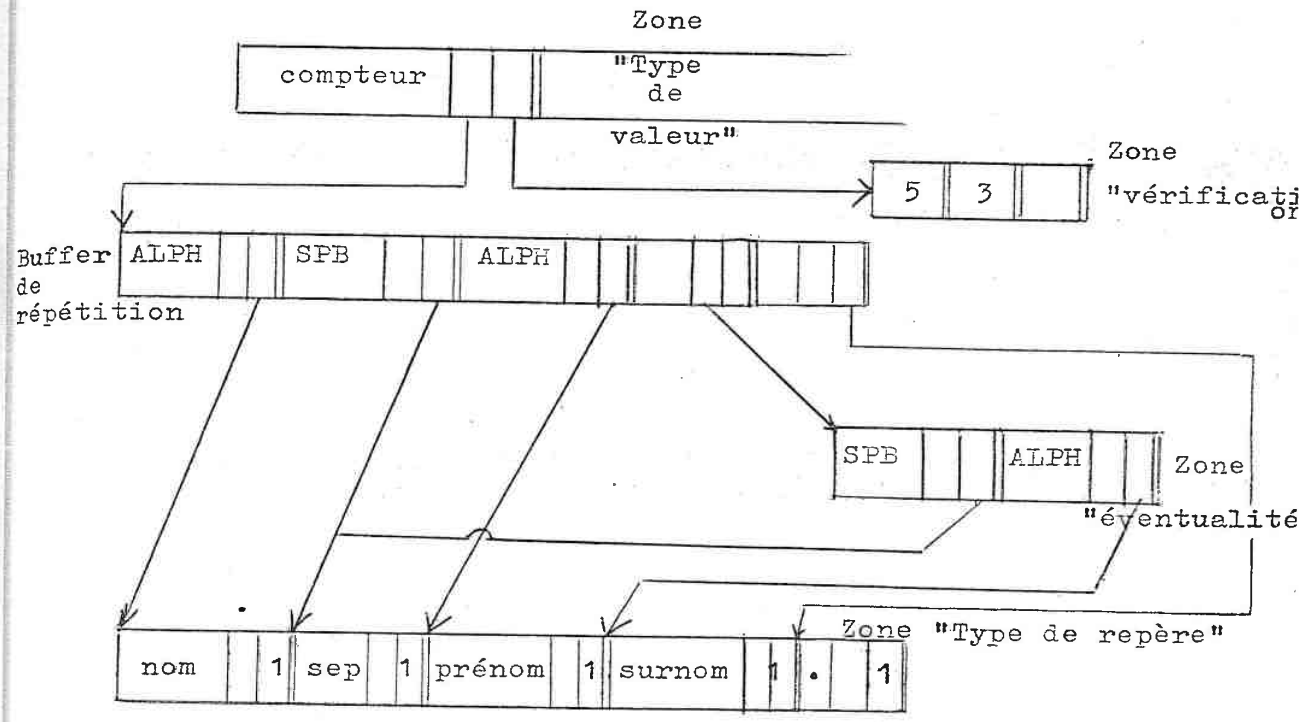


L'entier 5 de la première unité de la zoné 1 "vérification" est déduit du nombre de types de valeur du buffer de répétition 1 et du buffer de répétition 2.

3) Soit la description :

4((compteur《ALPH》nom,《SPB》sep,《ALPH》prénom[, 《SPB》 sep, 《ALPH》 surnom].))

Après analyse, la traduction pourra être représentée en mémoire par :



L'entier 5 de la première unité de la zone "vérification" est déduit du nombre de types de valeurs du buffer de répétitior et de la zone "éventualité".

- Si l'analyse reconnaît des changements de support physique suivant des règles associées à des répétitions, alors :

  1) Quelles que soient les règles, un élément compteur est créé dans la zone "type de valeur".
     Comme tous les éléments de cette zone, il est formé de 3 unités mémoire :

- la première unité contient le nom du compteur.
- la deuxième unité est un pointeur vers un buffer de répétition.
- la troisième unité est un pointeur vers une zone "règle".

Le buffer de répétition est formé d'éléments identiques aux éléments de la zone "type de valeur". Ces éléments sont la traduction des types de valeur rencontrés entre ((et)). Les types de repère auxquels ils sont associés sont traduits comme indiqué en 3.4.2 p. 54-5

2)Les zones "règles" ont toutes 3 unités mémoire :

- la première unité-mémoire contient le nombre d'éléments du buffer de répétition.
- la deuxième unité-mémoire contient éventuellement le nombre exact de répétitions.
- la troisième unité-mémoire sera utilisée lors du traitement.

De plus si la règle est la règle nº 1 ou la règle nº 2, la zone "règle" associée a une unité-mémoire supplémentaire contenant le nom du type de repère décrit dans la règle.

Si la règle est la règle Nº 12, la zone "règle" associée a deux unités-mémoire supplémentaires contenant les noms des types de repère décrits dans la règle, dans leur ordre de description.

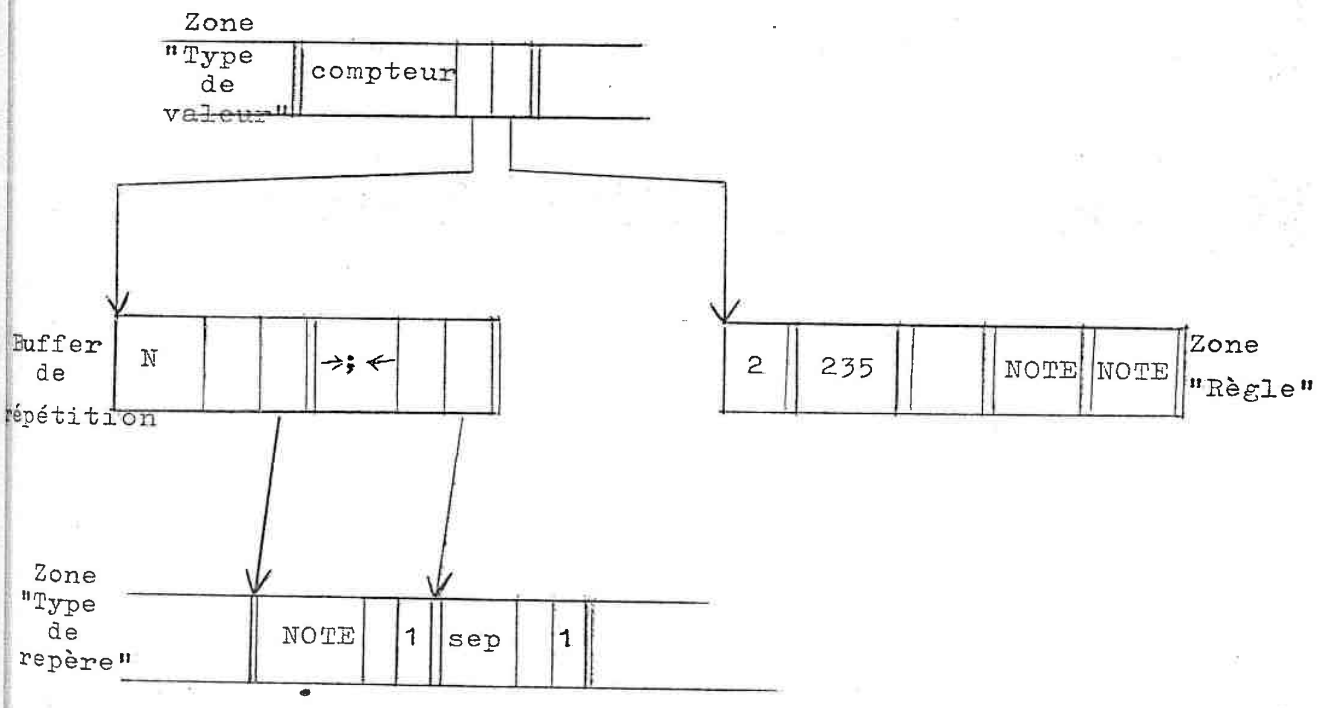
Si la règle est la règle nº 122, la zone "règle" associée a trois unités-mémoire supplémentaires contenant les noms des types de repères décrits dans la règle, dans leur ordre de description.

Nous illustrons ces traductions de changements de supports physiques par un exemple :

Soit la description :

235((compteur《N》note,《→;←》sep))[12]note,note[12]

Rappelons que cette description signifie qu'il y a 235 notes sur un certain nombre de supports physiques. Les notes sont séparées par un point virgule. Au début et à la fin de chaque support physique on a une note. Cette description, après analyse, est reconnue comme un changement de support suivant la règle nº 12. Elle est alors traduite par :



. Si l'analyse reconnaît une éventualité, alors :

Un élément éventualité est créé dans la zone "type de valeur". Il est formé de trois unités-mémoire.

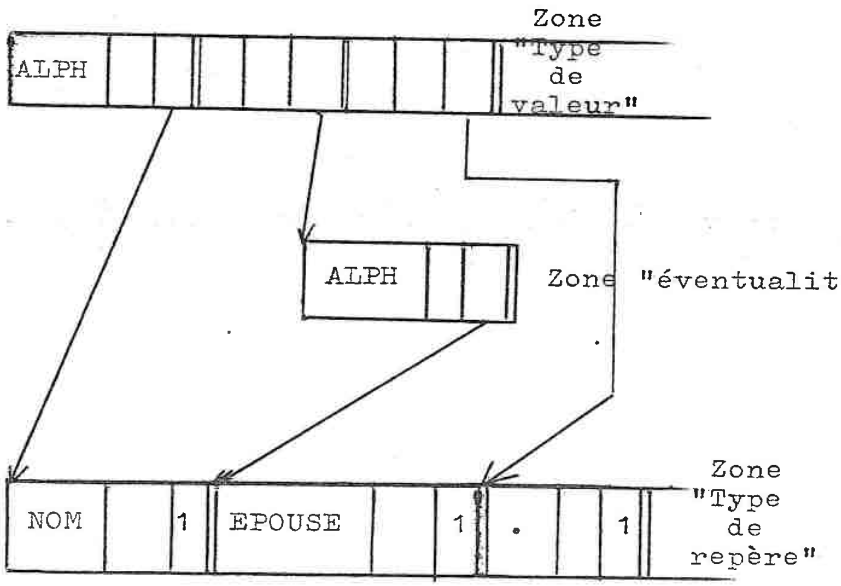
- la première unité n'est pas utilisée.
- la deuxième unité est un pointeur vers une zone "éventualité".
- la troisième unité sera utilisée lors du traitement.

La zone "éventualité" est formée d'éléments identiques à ceux de la zone "type de valeur". Ce sont les traductions des types de valeur décrits entre [et] . Les types de repères, auxquels ils sont associés, sont traduits comme indiqué en 3.4.2 pages 54 - 55

Si, par exemple, on a la description :

≪ALPH≫ nom [, ≪ALPH≫ épouse] .

Après analyse, la traduction peut être représentée en mémoire par :



- Si l'analyse reconnaît le nom d'une relation, alors :
  Un élément est créé dans la zone "relations"; on n'en connait pas encore la taille-mémoire.
  - la première unité contient le nom de la relation.
  - les unités suivantes sont des pointeurs vers des éléments de la zone "départ" ou de la zone "arrivée".
- Si l'analyse reconnaît le nom d'un ensemble de départ d'une relation, alors:

Un élément est créé dans la zone "départ".
Cet élément est pointé par l'unité courante de l'élémen de la zone "relations" auquel il est associé.

- la première unité contient le nom de l'ensemble.
- la deuxième unité est un pointeur vers une zone "traduction départ".

. Si l'analyse reconnaît le nom d'un ensemble d'arrivée d'une relation, alors :
Un élément est créé dans la zone "arrivée".
L'élément ainsi créé est pointé par l'unité courante de l'élément de la zone "relations" auquel il est associé.

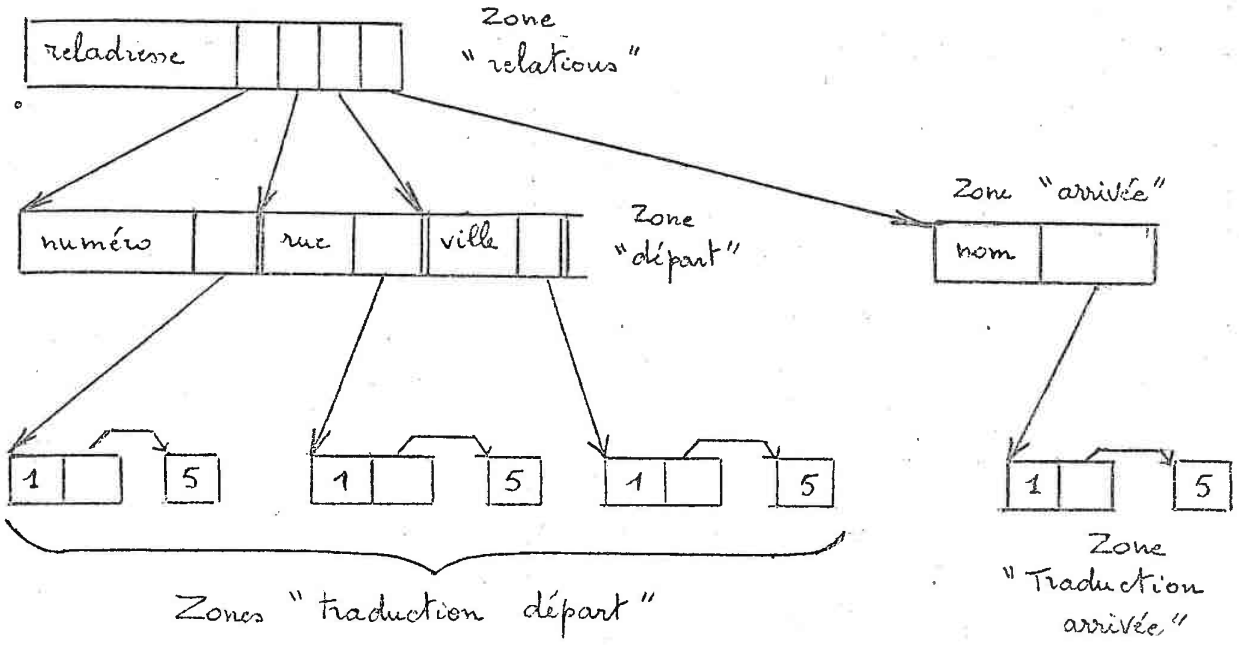
- la première unité contient le nom de l'ensemble.
- la deuxième unité est un pointeur vers une zone "traduction arrivée".

. Si l'analyse reconnait une expression d'une relation, alors :
On crée des éléments de la zone "traduction départ" (respectivement de la zone "traduction arrivée") en traduisant ce qui, dans l'expression, a trait à l'ensemble de départ ou aux ensembles de départ (respectivement à l'ensemble d'arrivée).

Exemples :

1) reladresse : (numéro, rue, ville), nom
((1:5),(1:5),(1:5)),(1:5);

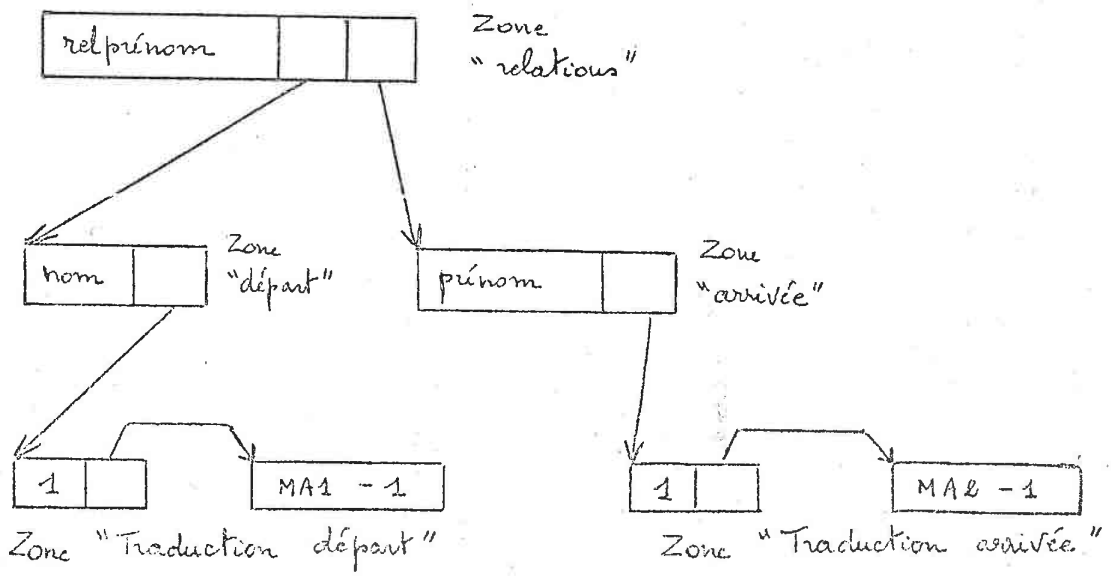
Après analyse la traduction de cette relation est représentée en mémoire par :



2) relprénom : nom,prénom,(1:(MA1-1)),(1:(MA2-1));

cette relation fait correspondre à un nom un prénom (à priori MA2 ≠ MA1)

Après analyse, la traduction de cette relation est représentée en mémoire par :



Remarque

Nous n'avons, par cette représentation en mémoire, qu'une traduction directe de la description de la relation. Nous verrons comment, à l'aide du traitement, on peut transformer cette représentation de la description de la relation en la représentation de la relation.

Nous venons de décrire les différentes représentations en mémoire des différents éléments d'une description écrite dans le langage de description de la forme externe. Nous allons illustrer la plupart de ces représentations à l'aide d'un exemple.

Exemple

Soit la description :

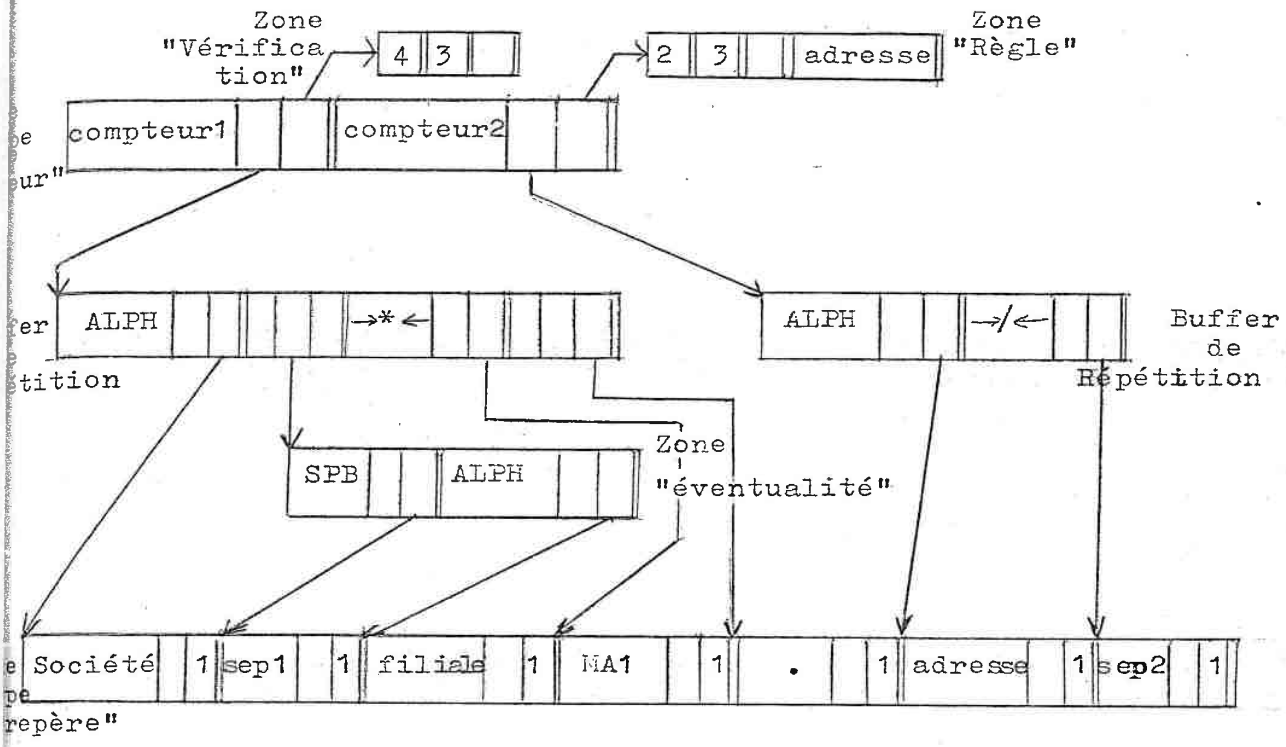
3((compteur1 ⟨⟨ALPH⟩⟩ société [, ⟨⟨SPB⟩⟩ sep1, ⟨⟨ ALPH ⟩⟩ filiale], ⟨⟨ → * ← ⟩⟩ MA1.))3((compteur2 ⟨⟨ALPH⟩⟩ adresse, ⟨⟨ →/← ⟩⟩ sep2)) [1] adresse [1]

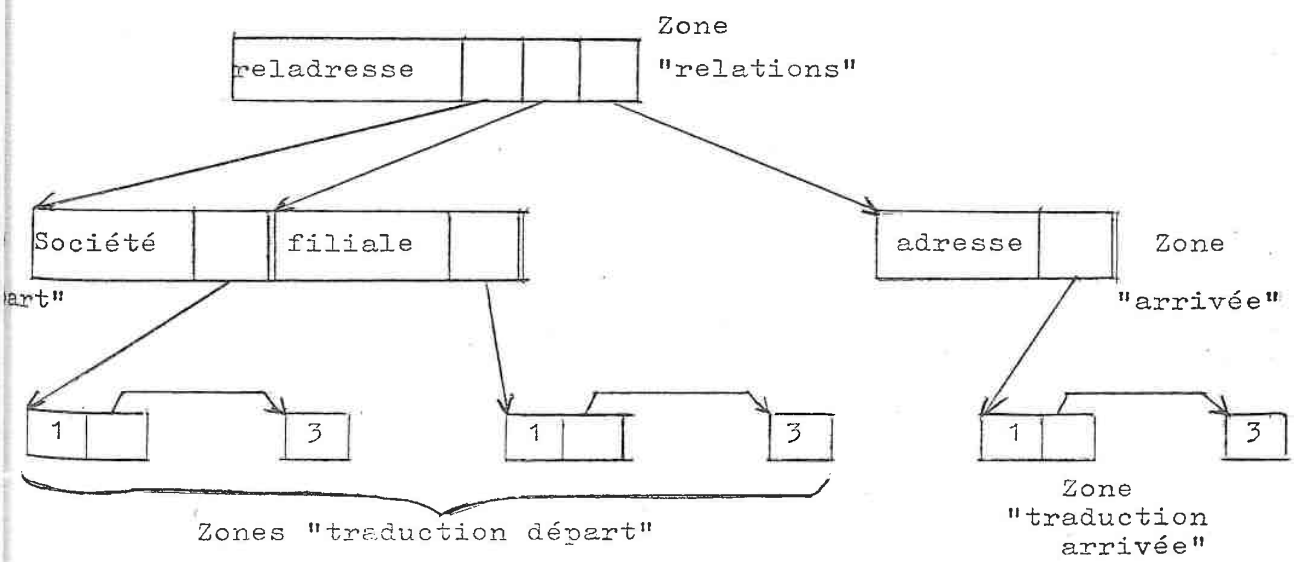
reladresse : (société,filiale),adresse,((1:3),(1:3)),(1:3);

Cette description signifie que :

- sur 3 cartes il y a 3 noms de sociétés, éventuellement suivis d'un nom de filiale (séparés par un blanc).
- il y a ensuite 3 adresses qui sont séparées par / et qui se trouvent sur un nombre indéterminé de supports. Chaque début de support commence par le début d'une adresse (règle nº 1).
- il y a une relation qui à un couple (société, filiale) fait correspondre une adresse.

Après interprétation, cette description est représentée par :





Si nous connaissons une description écrite dans le langage de description de la forme externe alors, après analyse, nous savons traduire cette description et la représenter en mémoire.

On a ainsi en mémoire la représentation de la structure EXLOG. Il reste à "affecter" les données par la structure EXLOG. C'est ce que nous allons faire en effectuant le traitement des données. Alors nous obtiendrons une représentation en mémoire des "données affectées par EXLOG", c'est-à-dire le résultat du "structureur".

3.4.3 Traitement des données

La chaîne de données est entrée en mémoire. Le changement de support physique y est noté.
La structure EXLOG est représentée en mémoire. C'est la traduction d'une description écrite dans le langage de la forme externe.
On part toujours de la zone "type de valeur". Dans la première unité de chaque élément, il y a la description d'un type de valeur.
On lit la chaîne de données caractère par caractère jusqu'à ce que le caractère lu ne corresponde plus au type de valeur considéré. La suite des caractères de la chaîne de données ainsi déterminée, constitue une donnée ou un élément de la chaîne de données.
Au fur et à mesure des traitements des éléments de la chaîne de données, on a une valeur courante de traitement de la chaîne de données. C'est le rang de l'élément de la chaîne tel que tous les éléments le précédant dans la chaîne soient déjà traités. On le fait varier chaque fois que l'on change d'élément dans la zone "type de valeur".

a) Traitement d'une donnée élémentaire

- On compare la première unité du premier élément de la zone "type de valeur" avec la chaîne de données.
- Si ce contrôle de cohérence convient, on pointe la deuxième unité du premier élément de la zone "type de valeur" vers la première donnée.
  Ainsi à la 1ère donnée sont attachés son type de valeur et son type de repère.
- On crée une zone "occurrence" associée au type de repère étudié et on remplit le premier élément de cette zone par le rang d'apparition, dans la chaîne de données, de la donnée ayant ce type de repère.
- On recommence le traitement en étudiant la première unité de l'élément suivant de la zone "type de valeur" et en le comparant à nouveau avec l'élément suivant de la chaîne de données.
  Les seuls changements sont au sujet de la zone "occurrence". Elle peut déjà exister et alors le rang d'apparition, dans la chaîne de données, de la donnée ayant ce type de repère, devra remplir le premier élément libre de cette zone "occurrence".

Remarques

1) Le rang d'une donnée, dans la chaîne de données, n'est pas automatiquement égal au rang du type de valeur associé à la donnée, dans la zone "type de valeur". En effet, le nombre d'éléments de la zone "type de valeur" n'est pas, en général, égal au nombre de données de la chaîne de données.

b) Traitement d'une marque

Que la marque corresponde à une donnée ou à une partie de donnée de la chaîne de données, le traitement est identique à celui d'une donnée élémentaire de la chaîne (donnée à laquelle sont associés un type de valeur et un type de repère).

c) Traitement d'un index de recul

L'index de recul ne correspond à aucune donnée de la chaîne de données. Dès l'interprétation de la description, nous avons mis en valeur cette particularité puisque, dans l'élément de la zone "type de valeur" associé à un index de recul, seule la deuxième unité contient l'entier se trouvant dans la description de l'index de recul.

Le traitement consiste à :

- "reculer" la lecture de la chaîne de données d'autant de caractères que le contenu de la deuxième unité de l'élément de la zone "type de valeur" associé à l'index de recul.
- diminuer de 1 la valeur courante de traitement de la chaîne de données.

d) Traitement d'un changement de support

Onsait que, dans la zone "type de valeur" on a un élément dont la première unité est vide et dont la deuxième unité pointe vers un élément de la zone "type de repère". Le premier élément de ce dernier contient le caractère point.

Le traitement consiste à vérifier d'abord qu'il y a changement de support dans la chaîne de données;

ensuite on remplit le premier élément libre de la zone "occurrence" par la valeur courante de traitement de la chaîne de données.

On n'incrémente pas la valeur courante de traitement de la chaîne de données.

e) Traitement d'un type de repère complexe et de ses constituants

Les types de repère associés aux constituants d'un type de repère complexe, sont traités comme tous les types de repère.

Le nom du type de repère de chaque constituant est cependant noté, dans l'ordre où on le trouve, dans les éléments de la zone "constituant" associée à la parenthèse (commençant la description du type de repère complexe.

Le rang d'apparition de la donnée ayant comme type de repère celui d'un constituant est noté dans l'un des éléments de la zone "occurrence" associée à la parenthèse (commençant la description du type de repère complexe. (l'ordre d'apparition de ces rangs d'apparition correspond à l'ordre des types de repère des constituants dans la zone "constituant").

Lorsqu'on traite le type de repère complexe on lui associe une zone "occurrence" formée de listes chaînées; chaque liste est constituée par les rangs d'apparition des données correspondant aux constituants du type de repère complexe.

f) TRAITEMENT d'une répétition

Dans la zone "type de valeur", on a un élément compteur auquel sont associés une zone "vérification" et un buffer de répétition.

Deux cas se présentent :

1) On connait le nombre exact de répétitions; il est indiqué dans la première unité de la zone "vérification".

   Si n est le nombre de répétitions on doit alors trouver dans la chaîne de données n fois les données décrites dans le buffer de répétition.

   Le traitement devra donc :

   a) contrôler la cohérence de chaque donnée en la comparant au contenu de la première unité de chaque élément du buffer de répétition.

   b) Si ce contrôle convient, remplir le premier élément libre de la zone "occurrence" associée au type de repère étudié par le rang d'apparition dans la chaîne de données, de la donnée ayant ce type de repère.

   c) incrémenter la valeur courante de traitement de la chaîne de données ainsi que le contenu de la troisième unité de la zone "vérification".

   d) vérifier, à la fin de la répétition, que le produit des contenus des deux premières unités de la zone "vérification" est égal au contenu de la troisième unité de cette zone (nombre de répétitions x nombre d'éléments du buffer = nombre de données).

2) On ne connait pas le nombre exact de répétitions. On sait qu'alors, après la description de la zone de répétition, il y a la description d'une marque pour différencier les données répétées et les données suivantes.

Le traitement devra donc contrôler l'existence de chaque donnée comme une donnée de la zone de répétition en la comparant au contenu de la première unité de cette marque.

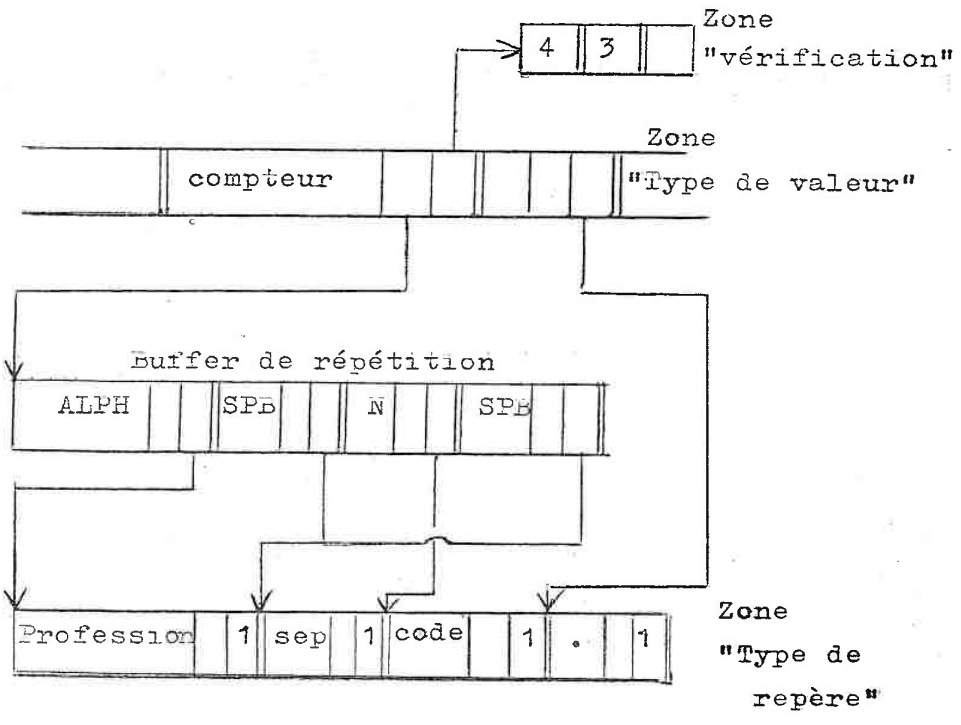
Si ce contrôle convient, on effectuera les points b et c du paragraphe précédent.

Exemple

Soit la description :

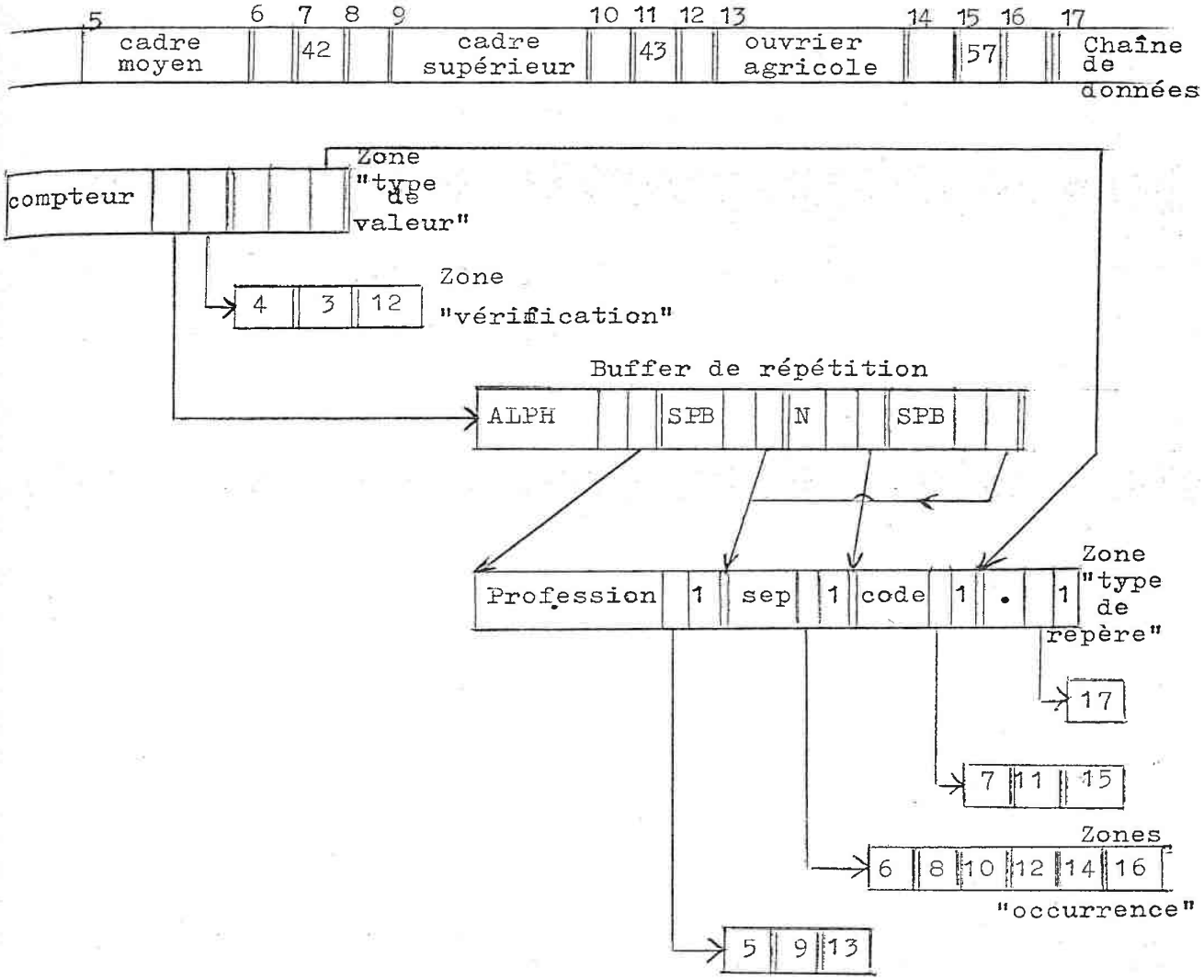
3((compteur ≪ALPH≫ profession, ≪SPB≫ sep, ≪N≫ code, ≪SPB≫ sep)).

L'interprétation de cette répétition donne :



On suppose que le premier élément de la chaîne de données qui est concerné par cette répétition a pour rang d'apparition : 5.

Après traitement cette répétition devient :



g) Traitement de changements de support avec règle associée à une répétition

Le traitement est identique à celui d'une répétition sans règle.

La zone "règle", associée au changement de support, joue un rôle identique à celui joué par la zone "vérification", associée à une répétition. (la troisième unité-mémoire s'incrémente donc en même temps que la valeur courante de

traitement de la chaîne de données).

Suivant le numéro de la règle il faut, en plus des autres contrôles, comparer les données rencontrées, en début et/ou en fin de support, avec les indications précisées dans la zone "règle".

h) Traitement d'une éventualité

Que l'éventualité se trouve ou non dans une zone de répétition, le traitement est le même ;

- On sait que la description de l'éventualité est suivie de la description d'une marque. Il faudra donc comparer chaque donnée avec le contenu de la première unité de l'élément de la zone "type de valeur" associé à cette marque, pour contrôler l'existence de chaque donnée comme une donnée de la zone "éventualité".
- La troisième unité de l'élément correspondant à la zone "éventualité" est un pointeur vers une zone "masque". Cette zone "masque" est une liste d'unités mémoire. Chaque unité-mémoire est associée à une donnée éventuelle. Elle est formée de bits déduits de la règle suivante :
  - quand une donnée est cohérente avec la première unité de la zone "éventualité" le bit 1 est inscrit.
  - quand il n'y a pas cohérence le bit 0 est inscrit.
- On a ainsi un masque qui permet de déterminer exactement l'existence de la (ou des) donnée (s) décrite (s) dans l'éventualité.
- Le traitement des données éventuelles est identique à celui des données élémentaires.
- La détermination des rangs des données éventuelles dans la chaîne de données ainsi que l'incrémentation de la

· valeur courante de traitement de la chaîne de données sont facilitées par l'existence de la zone "masque".

Exemple: Soit une répétition avec éventualité.

La zone "vérification" est | 2 | 3 | |. Cela signifie qu'il y a 3 fois une zone formée de deux éléments. Le deuxième élément est éventuel; il lui est attaché la zone "masque" | 0 | 1 | 0 |.

Cela signifie que: la 1ère fois il n'y a qu'une donnée.
la 2ème fois il y a 2 données.
la 3ème fois il n'y a qu'une donnée.

Il y a en tout 4 données.

A la fin du traitement de la répétition la zone "vérification" sera | 2 | 3 | 4 |

Et on pourra contrôler que le contenu de la troisième unité de la zone "vérification" est égale au produit des contenus des deux premières unités de cette zone auquel on enlève le nombre de zéros de la zone masque.

Dans l'exemple ci-dessus $4 = (2 \times 3) - 2$.

i) Traitement d'une relation

Les compteurs et les marques sont déjà traités donc tous les noms de compteurs et les noms de marques figurant dans les zones "traduction départ" et "traduction arrivée" peuvent être remplacés par des entiers.

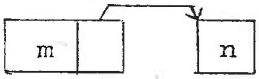
Pour les compteurs, le nom du compteur sera remplacé par le contenu de la troisième unité de la zone "vérification" associée au compteur.

Pour les marques, le nom de la marque sera remplacé par le rang d'apparition de la marque se trouvant dans la zone "occurrence" associée à la marque.(d'où la nécessité pour l'utilisateur, s'il veut décrire des expressions avec des marques, de définir des marques sans ambiguité, c'est-à-dire

sans occurrences multiples)

Le traitement des relations consiste:

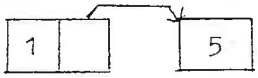
- Dans un premier temps à transformer les zones "traduction départ" et "traduction arrivée" en chaînes d'unités contenant des valeurs entières.
- Ensuite à transformer ces chaînes pour pouvoir connaitre tout élément du graphe de la relation considérée, c'est-à-dire connaitre:
  - la liste des occurrences de l'(ou des) ensemble(s) de départ.
  - la liste des occurrences de l'ensemble d'arrivée, occurrences associées aux précédentes.

Soit la chaîne 

($m < n$). Cette chaîne est une zone "traduction départ" associée à un élément de la zone "départ", c'est-à-dire à un type de repère.

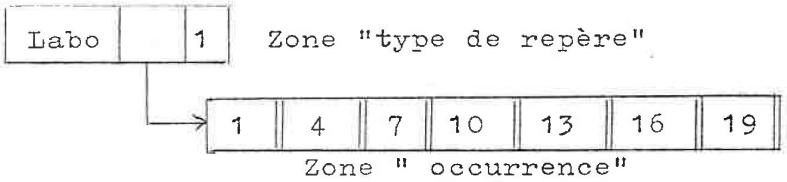
On va substituer à cette chaîne la liste des occurrences de ce type de repère à partir de la m-ième unité jusqu'à la n-ième unité.

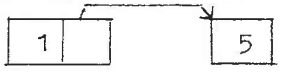
Exemple

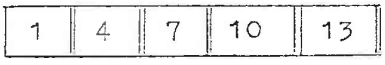
Soit un élément de la zone "départ" désigné, par exemple, par le repère Labo avec une zone "traduction départ" formée de la chaîne 

.

On va substituer à cette chaîne la liste des contenus des unités mémoire nº 1,2,3,4,5 de la zone "occurrence" associée au type de repère Labo.

Ainsi pour: 

On remplace la chaîne 

par 

Remarques

1) Il se peut que le nombre d'éléments de la zone "occurrence" associée à un type de repère soit moins grand que le nombre d'éléments nécessaires pour les substituer à une zone "traduction" (départ ou arrivée) associée à ce même type de repère (cela est le cas notamment pour un type de repère décrit dans une zone "éventualité").
Alors, bien sûr, seuls les éléments définis dans la zone "occurrence" apparaitront dans la substitution de la zone "traduction". Dans les autres éléments on laissera des blancs.

2) La représentation en mémoire des relations, après traitement, est coûteuse en place mémoire.
Il y avait deux solutions de représentation en mémoire d'une relation: en extension, en compréhension.
Au niveau de la description, les relations sont définies en compréhension à l'aide d'expressions permettant d'exprimer sous forme abrégée les occurrences des relations.
Ces expressions, après analyse et traduction, sont inutilisables si l'on veut connaître un élément précis du graphe d'une relation.
Les relations sont donc représentées en mémoire par extension avec:
- la liste des occurrences du (ou des) ensemble (s) de départ.
- la liste des occurrences de l'ensemble d'arrivée, occurrences associées aux premières par la relation.

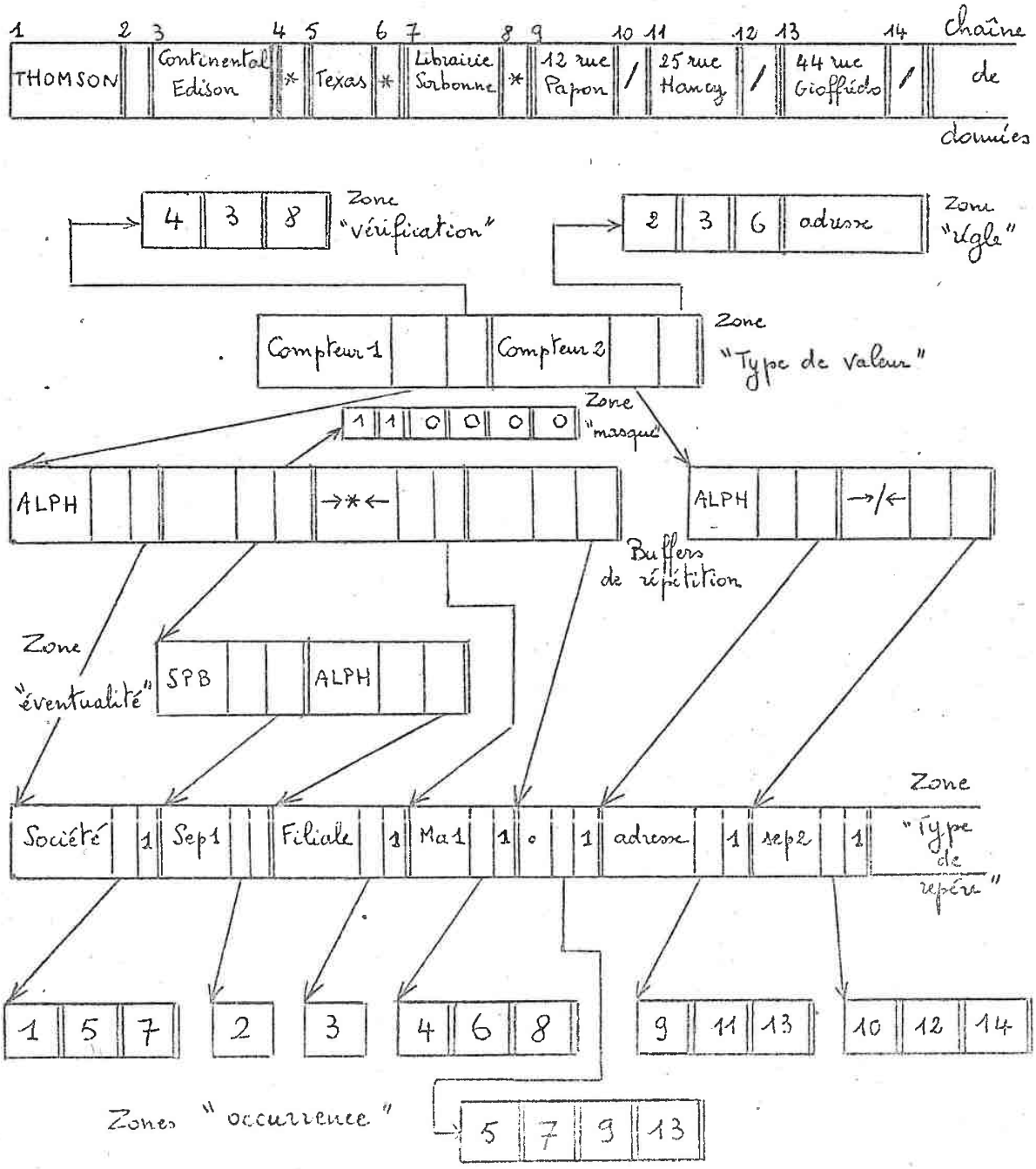
3.4.4 Exemple

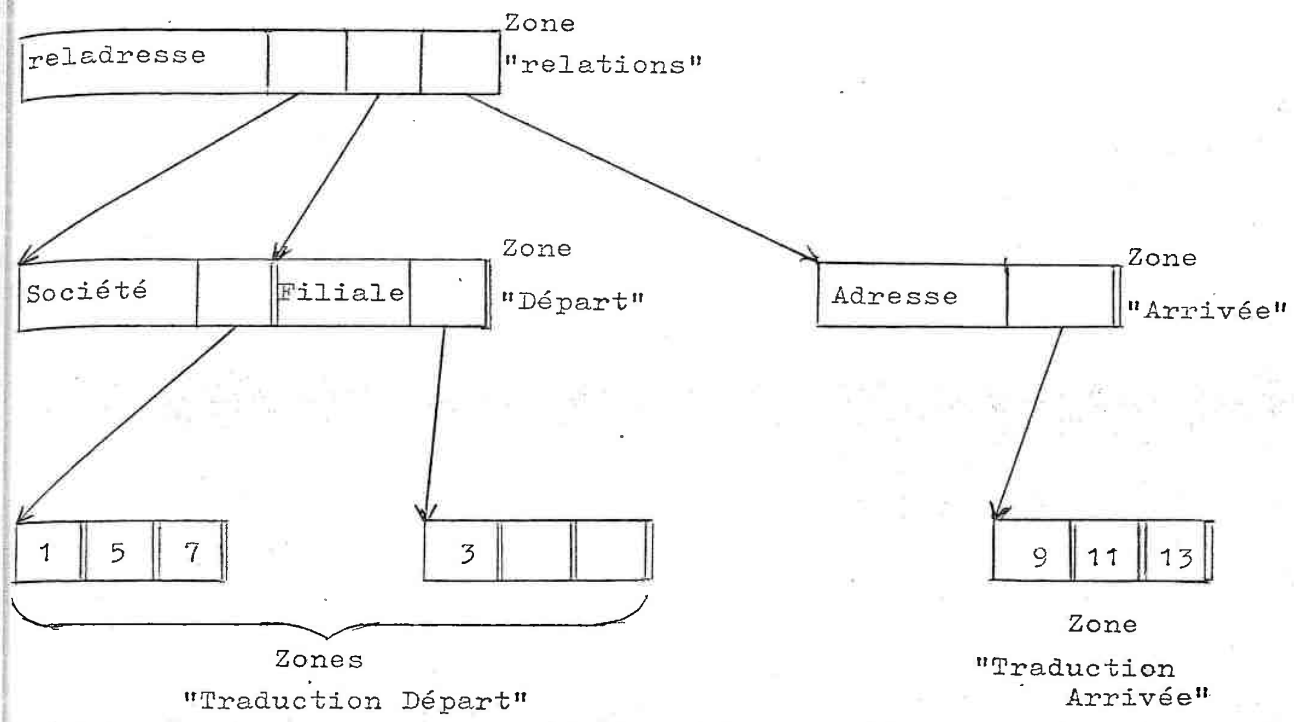
Pour terminer ce chapître, nous reprenons l'exemple de la fin du paragraphe 3.4.2. dont voici la description:

3((compteur1 ≪ALPH≫ société [, ≪SPB≫ sep1, ≪ALPH≫ filiale], ≪→*←≫ MA1.)) 3((compteur2 ≪ALPH≫ adresse, ≪→/←≫ sep2))

. [1] adresse [1]

Nous donnons le résultat du "structureur" pour la chaîne de données suivante:





Nous venons de voir l'action du "structureur sur une description écrite dans le langage de description de la forme externe. Le "structureur" affecte les données par EXLOG. Ce résultat est celui de la première phase.

Dans les chapîtres suivants nous allons décrire la deuxième phase: la correspondance EXLOG $\rightarrow$LOG, puis le programme utilisant cette correspondance. Cette deuxième phase transformera les "données affectées par EXLOG" en "données affectées par LOG".

CHAPITRE 4

La deuxième phase

Nous allons d'abord préciser la correspondance EXLOG ⟶ LOG, correspondance qui transforme une structure en une autre structure. Puis, dans les paragraphes suivants, nous verrons le programme qui, en utilisant cette correspondance, transformera les "données affectées par EXLOG" en "données affectées par LOG".

4.1 Définition de la correspondance EXLOG ⟶ LOG

La structure LOG est, comme nous l'avons indiqué dans le premier chapitre, la "façon de comprendre les données".
Nous n'allons pas définir la manière de décrire la structure LOG mais rappeler simplement sa nature.
Quelle que soit la façon de décrire la structure LOG, la nature de LOG sera toujours la même. On retrouvera :

- des types de repères associés aux données élémentaires.
- des relations définies à partir des types de repères.
  (avec éventuellement une expression permettant de connaitre tout n-uple des graphes des relations).

La correspondance EXLOG ⟶ LOG exprime tout élément de l'une des structures en fonction d'un ou de plusieurs éléments de l'autre structure. La correspondance utilise ainsi des éléments de chacune des structures. Elle peut aussi utiliser les données affectées par l'une ou l'autre des structures. (voir exemples paragraphe 4.2.2) Ce que nous appelons ici élément d'une structure est aussi bien un type de repère qu'une relation. Comme dans tous les problèmes de correspondance, il parait souhaitable

de partir d'un élément de la structure LOG, c'est-à-dire de la structure d'arrivée, et d'essayer de l'exprimer en fonction d'un ou plusieurs éléments de la structure EXLOG, c'est-à-dire de la structure de départ.

Etant donné un élément de la structure LOG, on pourrait essayer de déterminer cet élément en fonction d'éléments d'EXLOG en créant des opérations sur les éléments d'EXLOG (groupements de types de repère, composition de relations....). Cette correspondance formelle entrainerait la création d'un logiciel pour l'interpréter. Cette solution sera peut-être réalisée plus tard, mais, dans l'immédiat, pour garder une plus grande souplesse dans les possibilités de correspondance et pour ne pas présenter des développements trop longs à ce niveau, nous proposons une solution où la correspondance sera exprimée, en fait, d'une manière moins concise.

4.2 Résolution de la correspondance : les procédures

Nous proposons la solution suivante : le concepteur du système écrit pour chaque élément de la structure LOG une procédure qui lui permet de l'exprimer en fonction d'un ou plusieurs éléments de la structure EXLOG et/ou d'éléments de la structure LOG.

Pour chaque type de repère de LOG, pour chaque relation de LOG, l'utilisateur connait donc une procédure qui définit ce type de repère ou cette relation en fonction de types de repère et/ou de relations de la structure EXLOG ou de la structure LOG (pour ces derniers il faudra qu'ils aient été précédemment définis).

Donnons quelques exemples définissant un élément d'une structure LOG.

A un type de repère de LOG pourra être associée la procédure le définissant comme étant :

- un type de repère d'EXLOG.
- une "suite" de types de repère d'EXLOG.
- un "groupe" de types de repère d'EXLOG.
- le type de repère de l'ensemble d'arrivée d'une relation d'EXLOG.
- le type de repère de l'ensemble de départ d'une relation d'EXLOG.
- le "groupe" de types de repère des ensembles définissant une relation d'EXLOG.
- le type de repère de la restriction d'un ensemble de départ ou d'arrivée d'une relation.
- le type de repère de l'image d'une restriction d'un ensemble de départ par une relation.
- le type de repère d'une donnée obtenue en effectuant des opérations sur des contenus de repère d'EXLOG.

A une relation pourra être associée la procédure la définissant comme étant :

- la composée de relations d'EXLOG.
- l'inverse d'une relation d'EXLOG.
- l'itération d'une relation d'EXLOG.
- la composée de relations et/ou d'inverses de relations et/ou d'itérations de relations.

Remarques

1) Nous ne donnons ici que quelques possibilités; nous en illustrerons les plus caractéristiques dans des exemples au paragraphe 4.2.2

2) Parmi les autres possibilités signalons qu'à un élément de LOG peut être associée une procédure conditionnelle (définition de cet élément selon: des valeurs se trouvant dans la chaîne de données, des calculs sur ces valeurs....)

4.2.1 Les ensembles sur lesquels les procédures travaillent

Une procédure peut utiliser en particulier les ensembles suivants :

- Types de repères de LOG (respectivement EXLOG) désignés par TRLOG (resp. "type de repère")
- Noms de relations de LOG (respectivement EXLOG) désignés par RE LOG (resp. "relations")
- Ensembles de départ de relations de LOG (respectivement EXLOG) désignés par DRE LOG (resp. "départ")
- Ensembles d'arrivée de relations de LOG (respectivement EXLOG) désignés par ARE LOG (resp. "arrivée")

4.2.2 Ce que fait chaque procédure

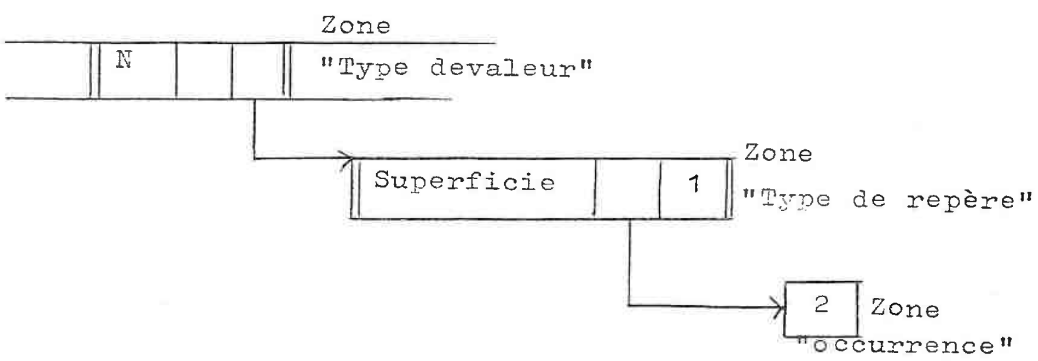
Chaque procédure est associée à un élément de la structure LOG (type de repère ou relation). Elle doit créer les occurrences de cet élément. Elle doit aussi établir les liens éventuels entre les occurrences créées et d'autres ou entre ces occurrences et les éléments de la chaîne de données.

Sans entrer dans le détail des procédures, nous donnons quelques exemples :

1) Un élément de TR LOG peut être identifié à un élément de la zone "type de repère" d'EXLOG.

   Exemple

   Soit l'élément superficie de la zone "type de repère" d'EXLOG représenté en mémoire par :



La procédure associée à l'élément superficie de TR LOG sera telle que :

- l'image de l'élément de la zone "type de valeur" pointera vers l'élément superficie de TR LOG
- l'image de la zone "occurrence" sera pointée par l'élément superficie de TR LOG.

2) Une donnée n'existant pas dans la chaîne de données peut être créée.

Exemple

Un laitier recueille chaque jour du lait provenant de différents producteurs.

A la fin de chaque semaine, il rentre sur ordinateur les noms des différents producteurs ainsi que les quantités de lait apportées chaque jour par chaque producteur, renseignements correspondant à ses bordereaux d'entrée.

Il ne veut garder en mémoire que la quantité de lait apportée par chaque producteur durant une semaine.

Dans EXLOG on a les types de repère :

- Laitier
- qLait / jour

Dans LOG on a les types de repère

- Laitier
- qLait / semaine

Au type de repère qLait / semaine est associée une procédure dont le résultat est la somme des 7 contenus des données de type de repère qLait / jour associées à un même laitier.

Par cette procédure, à une occurrence de type de repère

qLait / semaine, élément de la zone TR LOG est associée une suite de 7 occurrences du type de repère qLait/jour, élément de la zone "type de repère" d'EXLOG.
Insistons bien sur le fait qu'une nouvelle donnée est ici créée. Elle ne fait pas partie de la chaîne de données en entrée.

3) Un élément de TR LOG peut être construit en "groupant" des éléments de la zone "type de repère".

Exemple

Dans EXLOG on a les types de repère :

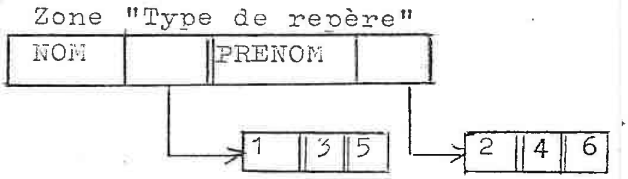
- nom
- prénom

Dans LOG on a le type de repère :

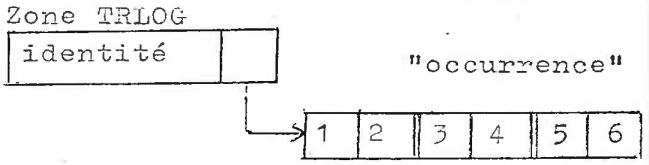
- identité

Au type de repère identité, élément de la zone "type de repère" de LOG, est associée une procédure qui définit ses occurrences à partir des occurrences de nom et prénom, éléments de la zone "type de repère" d'EXLOG.

Si l'on a dans EXLOG:



Alors dans LOG:



4) Une relation de la zone RELOG peut-être définie à partir de relations de la zone "relations" d'EXLOG.

Exemple

Dans EXLOG on a les relations :

r_1 : nom, numéro

r_2 : prénom, numéro

Dans LOG on a la relation :

r : (nom, prénom), numéro

Si nu_i désigne la i-ème occurrence du type de repère numéro, alors la procédure définissant la relation r pourra utiliser :

$$r\left(r_1^{-1}(nu_i),\ r_2^{-1}(nu_i)\right) = nu_i$$

Dans cet exemple, nous avons supposé qu'il y avait identification entre les types de repère d'EXLOG et ceux de LOG.

4.3 Construction des "données affectées par LOG"

On a en mémoire la représentation des "données affectées par EXLOG" et l'on veut construire les "données affectées par LOG". Si nous reprenons la comparaison des transparents, on veut fabriquer un nouveau transparent à partir du transparent précédent et des données.

Parallélement à la représentation en mémoire des "données affectées par EXLOG", on crée des zones mémoire : une zone mémoire notée TRLOG, une zone mémoire notée RELOG, des zones mémoire notées DOLOG.

- La zone mémoire TRLOG est du même genre que la zone mémoire "type de repère" décrite dans la représentation en mémoire d'EXLOG (cf p 44).

  Seul le bit de suppréssion n'est pas créé pour chaque élément.

. La zone mémoire RELOG est identique à la zone mémoire "relations" décrite dans la représentation en mémoire d'EXLOG (cf p 45).

. Chaque zone mémoire DOLOG est une liste d'éléments contigus, associée à un élément de TRLOG et pointée par la deuxième unité de cet élément.

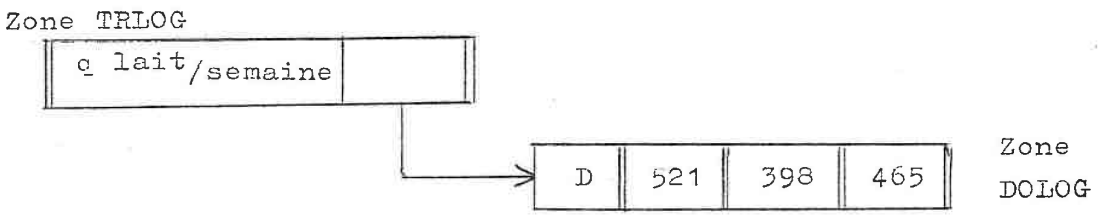
Le premier élément d'une zone mémoire DOLOG contient le caractère D (ce qui permet de différencier une zone DOLOG d'une zone "occurrence").

Les éléments suivants contiennent les données définies par la structure logique et n'appartenant pas à la chaîne de données.

Le contenu du n-ième élément de cette zone est la n-ième donnée associée au type de repère de TRLOG.

Exemple

Si l'on a la représentation en mémoire suivante:



465 est le contenu de l'élément de rang 4 de la zone associée à l'élément q $lait_{/semaine}$ de TRLOG. Ce contenu est en fait la troisième q $lait_{/semaine}$ car la zone pointée est une zone DOLOG (son premier élément contient le caractère D et non un entier comme ce serait le cas pour une zone "occurrence").

A chaque élément de LOG est associée une procédure.
Les appels des différentes procédures doivent se faire suivant un ordre précis. Nous donnons ici les règles qui régissent les appels des procédures.

1) Les éléments de TRLOG sont traités avant ceux de RELOG.

2) Tant que tous les éléments de TRLOG et de RELOG n'ont pas été traités on garde en mémoire la représentation des "données affectées par EXLOG".

3) Le traitement d'un élément de TRLOG consiste à mettre en oeuvre la procédure qui lui est associée; cette procédure a pour conséquences au moins l'une des suivantes:

   - le calcul des occurrences de l'élément considéré et la création d'une zone "occurrence" les contenant. La zone "occurrence" est, bien sûr, pointée par la deuxième unité de l'élément considéré.
   - le changement des bits de suppression d'éléments de la zone "type de repère".
   - le calcul d'une donnée et la création d'un élément de la zone mémoire DOLOG pour la contenir.

4) Le traitement d'un élément de RELOG consiste à mettre en oeuvre la procédure qui lui est associée; cette procédure a pour conséquences:

   - la définition des ensembles de départ et d'arrivée et la création d'éléments de DRELOG et ARELOG.
   - la définition de l'expression permettant de connaitre le graphe de la relation et la création de zones mémoire contenant la traduction de cette expression.

   Ces renseignements doivent être associés à l'élément de RELOG considéré.

5) Lorsque tous les éléments de RELOG ont été traités, alors la zone "relations" peut être supprimée ainsi que les zones "départ", "arrivée", "traduction départ", "traduction arrivée" qui lui sont associées.

6) Les éléments de la zone "type de repère", dont les bits de suppression sont à 0, sont supprimés.

7) Les éléments des zones mémoire "type de repère" et TRLOG sont réunis en une même liste d'éléments contigus: TR.

4.4 Exemple

Sur un exemple, nous allons voir la représentation en mémoire des "données affectées par LOG".

Une chaîne de garages reçoit de chacun de ses concessionnaires des factures où sont précisés:

- le prix de chacune des pièces placées sur les voitures.
- le prix de la pose.
- le prix de la T.V.A pour chacune des pièces et pour la pose.

Au service comptabilité on ne veut retenir pour chaque facture que le montant total des pièces posées, le montant total de la T.V.A, le montant des heures de pose.

Ainsi pour la facture suivante:

Nº : A 1052

Pièces	Prix pièce	T.V.A
1 silencieux S201	111,81	19,68
4 bougies 4xB12	26,80	4,72
pose	prix pose	
2h30'	50,90	8,96

Au niveau d' EXLOG, on a plusieurs types de repère: numérofacture, pièce, prixpièce, pose, prixpose, T.V.A et deux relations:

r_1 : relation avec ensemble de départ : prixpièce
ensemble d'arrivée : T.V.A

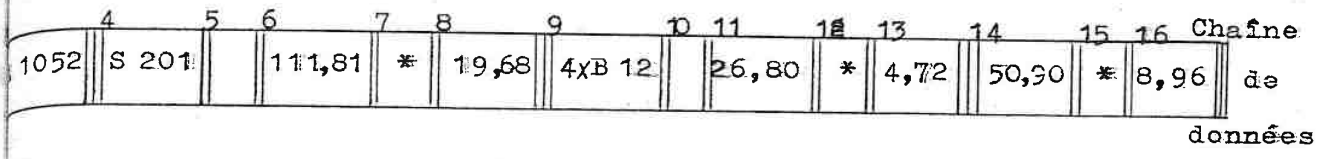
r_2 : relation avec ensemble de départ : pose
ensemble d'arrivée : T.V.A

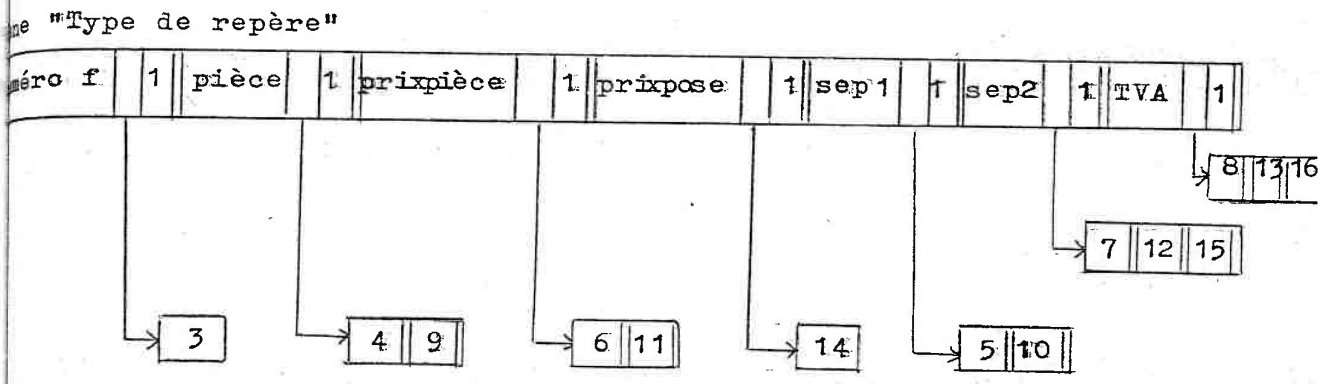
r_1 associe à une pièce la T.V.A de cette pièce.

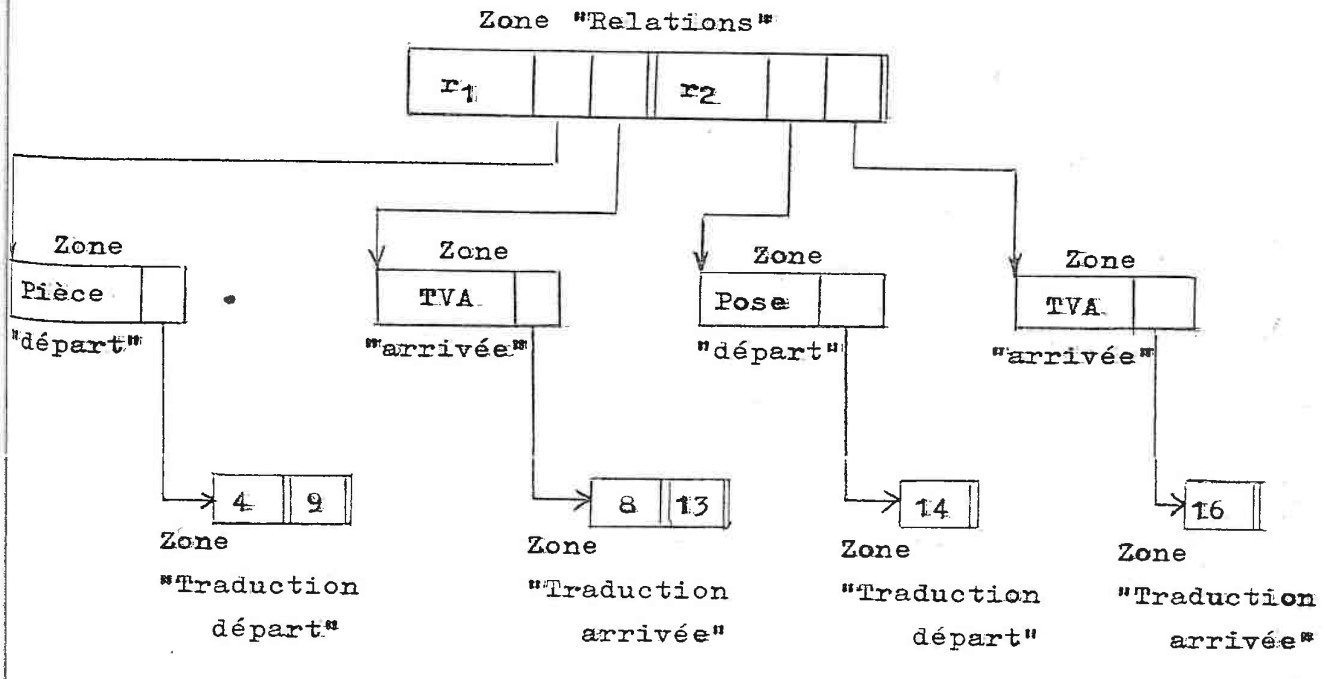
r_2 associe au montant de la pose la T V A sur ce montant.

Dans la représentation en mémoire des "données affectées par "EXLOG" que nous donnons ici, nous ne représentons que la chaîne de données, la zone "type de repère" et les zones "occurrences"

associées, la zone"relations" et les zones "départ", "arrivée", "traduction départ", "traduction arrivée" associées.







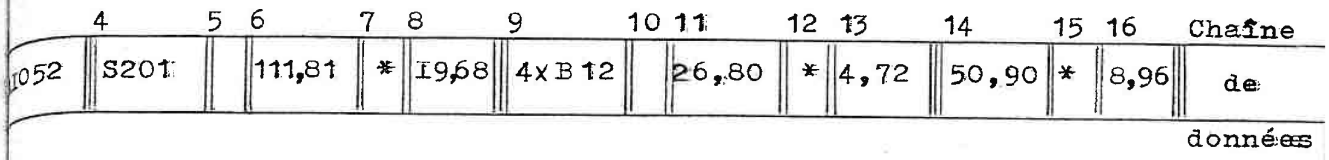
Au niveau de LOG, on a les types de repères suivants: numérofacture, totalpièce, pose, totalTVA et une relation:

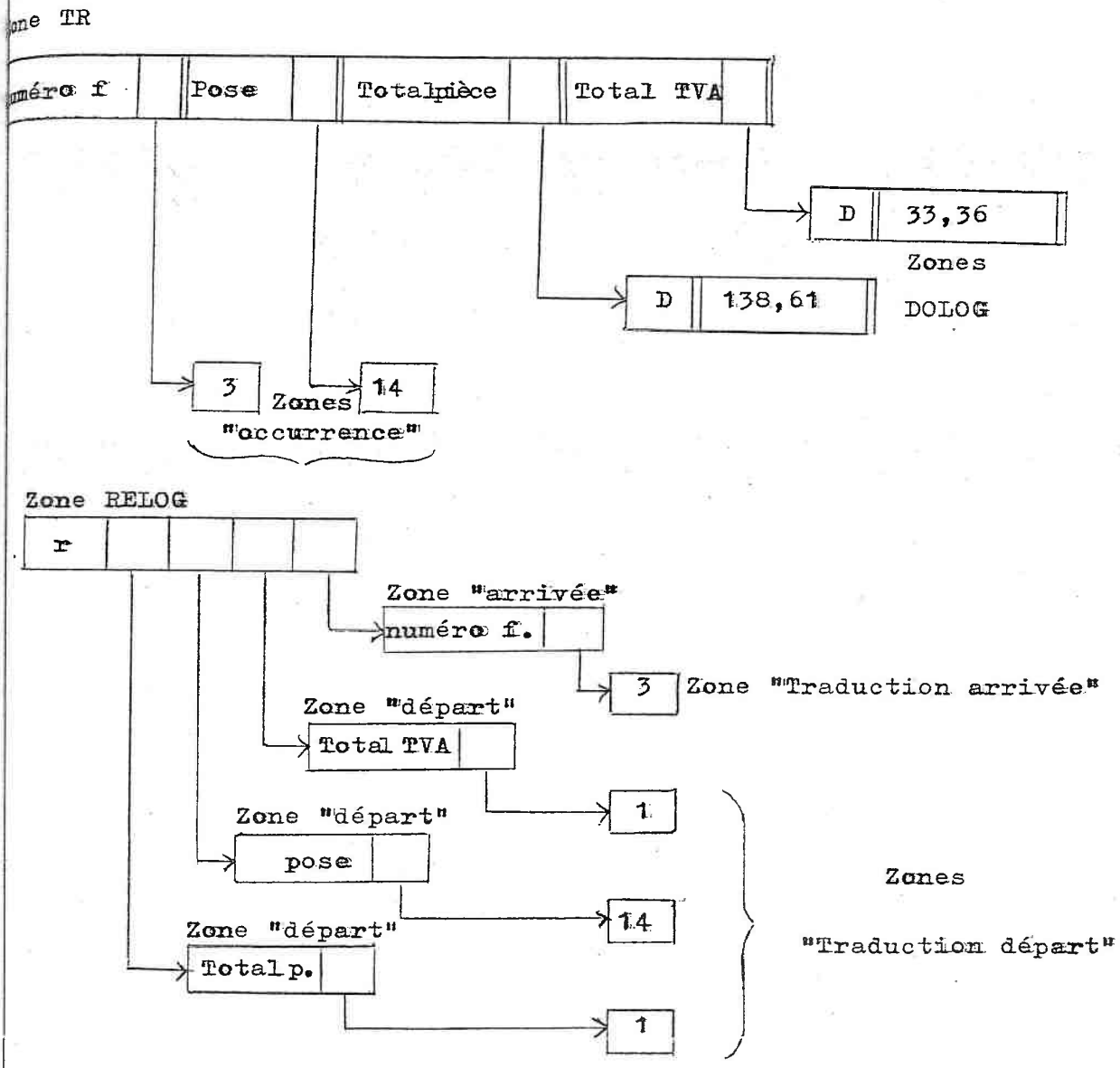
r: relation avec ensembles de départ: totalpièce, pose, totalTVA
ensemble d'arrivée : numérofacture.

r associe ce qui caractérise une facture pour le service comptabilité au numéro de cette facture.

Aux types de repère de LOG totalpièce et totalTVA sont associées deux procédures. Chacune de ces procédures calcule le total des montants des pièces d'une part, des montants de la T.V.A d'autre part, pour une même facture. Elle utilise donc les données associées aux types de repère d'EXLOG: pièce, T.V.A, numérofacture.

Dans la représentation en mémoire des "données affectées par LOG" que nous donnons ici, comme précédemment, nous ne représentons que la chaîne de données, la zone TR, les zones "occurrence" et DOLOG associées, la zone RELOG et les zones "départ", "arrivée", "traduction départ", "traduction arrivée" associées.





Nous savons donc représenter en mémoire les "données affectées par LOG". Nous allons maintenant étudier la troisième phase qui doit transformer les "données affectées par LOG" en "données affectées par PHYLOG".

CHAPITRE 5

La troisième phase

Comme nous l'avons fait au chapître précédent pour la deuxième phase, nous allons d'abord préciser la correspondance LOG → PHYLOG puis le programme qui, en utilisant cette correspondance, transforme les "données affectées par LOG" en "données affectées par PHYLOG".

5.1 Définitions

5.1.1 La structure PHYLOG

Par analogie avec la structure LOG que nous avions appelée "la façon de comprendre les données", nous pouvons définir la structure PHYLOG comme étant "la façon d'organiser les données en mémoire". C'est la structure logique de la représentation interne des données. Par sa définition même elle est indépendante des détails d'implémentation. Comme toute stucture, nous la définirons par:

- des types de repère (simples ou complexes)
- des relations (avec ensembles de départ, d'arrivée, expression)

Dès maintenant précisons que la structure PHYLOG est connue dans le système PIVOINES. Elle a déjà été introduite, pour l'interrogation, comme stucture intermédiaire entre la structure LOG et les procédures de recherche des données. Nous utiliserons d'ailleurs dans la suite la manière de décrire la correspondance LOG → PHYLOG qui avait été conçue pour l'interrogation (cf (2)).

5.1.2 La correspondance LOG ⟶ PHYLOG

C'est une correspondance entre structures. La correspondance doit utiliser des éléments de chacune des structures, ainsi que des données affectées par l'une ou l'autre des structures.

Il faut exprimer tout élément de la structure PHYLOG, structure d'arrivée, en fonction d'un ou plusieurs éléments de la structure LOG, structure de départ.

Nous nous plaçons, comme Marion CREHANGE dans sa thèse, dans l'hypothèse simplificatrice qui est que les repères de LOG et de PHYLOG sont les mêmes. Il suffit donc d'exprimer toute relation de PHYLOG en fonction d'un ou de plusieurs éléments de LOG.

5.2 Résolution de la correspondance : les tableaux

Nous proposons, au moins provisoirement, la solution suivante : le responsable du système écrit un tableau qui, à une relation de PHYLOG, fait correspondre un "groupement" ou une relation de LOG.

Ce que nous appelons ici "groupement" a le même sens que celui employé dans le système PIVOINES. Un groupement de LOG est la composition de relations de LOG sur lesquelles ont pu être effectuées des opérations (inverse, itération....).

Il existe des tables de traduction qui, pour l'interrogation, permettent de décrire la correspondance LOG ⟶ PHYLOG.

A une relation ou un "groupement" de relations de LOG, ces tables de traduction font correspondre une relation ou un "groupement" de relations de PHYLOG.

Leur lecture de droite à gauche ne permet donc pas de connaître chaque relation de PHYLOG en fonction de relations de LOG et

d'exprimer la correspondance LOG ⟶ PHYLOG comme nous le voulons.

Mais, pour autant, il serait dommage de ne pas utiliser ces tables de traductions pour écrire les tableaux.

Nous allons donner des règles simples d'utilisation "à la main" des tables de traductions qui permettent de trouver des lignes des tableaux.

5.2.1 Tables de traductions et Tableaux

Dans chaque ligne de l'une des tables de traductions il y a un groupement de LOG et une relation ou un groupement de PHYLOG qui "traduit" le groupement de LOG.

En fait, dans l'état actuel du système, on trouve toujours un nom de relation comme traduction en PHYLOG, mais cette relation est quelquefois égale à un groupement d'autres relations (c'est-à-dire que la procédure lui correspondant fait appel à d'autres procédures).

- Si l'on trouve une relation de PHYLOG, on peut directement utiliser cette ligne pour la construction du tableau.
- Si l'on trouve l'inverse d'une relation de PHYLOG on prend l'inverse du groupement de LOG associé pour connaitre la traduction dans LOG de la relation de PHYLOG. On utilisera alors les règles suivantes :

  - quelle que soit la relation r $(r^{-1})^{-1} = r$
  - quelles que soient les relations r et s $(rs)^{-1} = s^{-1} r^{-1}$

Exemple

De la ligne de la table de traduction :

LOG	PHYLOG
$a^{-1}bc$	r^{-1}

on déduit la ligne du tableau :

PHYLOG	LOG
r	$c^{-1}b^{-1}a$

- Si l'on trouve un groupement de PHYLOG, il faut composer ce groupement et d'autres groupements de PHYLOG se trouvant dans la table de traduction pour trouver une relation de PHYLOG.

Donnons un exemple très simple pour illustrer ces différents cas.

Soit la table de traduction suivante :

LOG	PHYLOG
R T	S^{-1}
$U R^{-1}$	$A B^{-1}$
V	B

De cette table de traduction on déduit le tableau suivant :

PHYLOG	LOG
S	$T^{-1}R^{-1}$
A	$U R^{-1}V$
B	$\overline{V}$

La première ligne du tableau a été déduite de la première ligne de la table de traduction puisque $S = (S^{-1})^{-1}$ et $T^{-1}R^{-1} = (RT)^{-1}$

La deuxième ligne du tableau a été déduite de la composition des deuxième et troisième lignes de la table de traduction. En effet $A = AB^{-1}B$ est associé à $U R^{-1}V$

La troisième ligne du tableau est déduite directement de la troisième ligne de la table de traduction.

Nous allons essayer de généraliser ces utilisations des tables de traduction et notamment l'obtention de nouvelles relations de PHYLOG pour créer les tableaux.
Nous allons d'abord donner quelques définitions ayant trait aux groupements.

5.2.2 Atomes – Groupements – Relations

a) Atome

Un atome est construit à partir d'une relation. C'est l'un des éléments suivants :

- une relation
- l'inverse d'une relation
- l'itération d'une relation
- l'itération de l'inverse d'une relation
- la négation d'une relation d'un des types précédents

Notations

- la relation est notée par son nom, exemple : r.
- l'inverse d'une relation r est notée classiquement : r^{-1} . (notation déjà utilisée au paragraphe précédent
- lorsqu'une relation r est composée n fois avec elle-même on la notera r^n, on dira puissance n de r.
- l'itération d'une relation r est notée : r^* .
- la puissance n de l'inverse d'une relation sera notée : $(r^{-1})^n$.
- l'inverse d'une puissance et donc d'une itération n'est pas autorisé.
- la négation d'une relation r est notée : $\neg r$.

b) Groupement

Un groupement est la relation obtenue en composant des atomes.

On appellera longueur d'un groupement le nombre d'atomes de ce groupement. C'est un entier naturel fini.

On appellera rang d'un atome dans un groupement le numéro d'ordre de cet atome dans ce groupement.

c) Ordre d'une relation définissant un atome

c'est 1 si le terme est la relation ou la négation de la relation.
- 1 si le terme est l'inverse de la relation.
n si le terme est l'itération n fois de la relation.
- n si le terme est l'itération n fois de l'inverse de la relation.

Nous allons maintenant décrire un progrmme d'évaluation de relations de PHYLOG à partir de groupements de PHYLOG.

5.2.3 Programme d'évaluation de relations de PHYLOG

Pour tout groupement de PHYLOG, on détermine d'abord les atomes le composant, le rang de chacun d'eux puis l'ordre de la relation définissant chaque atome.
On essaie d'abord de simplifier le groupement.
Dès qu'une même relation se trouve à des rangs successifs dans un même groupement alors on peut simplifier le groupement en remplaçant les atomes successifs par un seul atome. L'ordre de la relation constituant cet atome est la somme algébrique des ordres de la relation constituant les atomes initiaux.

Exemples

A B C D $D^{-1}C^{-1}B$ devient A B C $C^{-1}B$ puis A B B c'est-à-dire A B^2

A B^2 $B^{-1}C(C^{-1})^3$ est simplifié en A B $(C^{-1})^2$

On cherche ensuite les groupements composables.
Pour cela on cherche s'il existe deux groupements distincts, dans lesquels deux atomes sont construits à partir de la même relation.

On compare les rangs des deux atomes.
Si l'un des rangs est 1, et si l'autre est le nombre du groupement auquel l'atome appartient, alors on peut composer les deux groupements.

Exemple

Soit un groupement GR_1 : AB^{-1}

Soit un groupement GR_2 : B^2CD

Au groupement GR_1 sont associés

la longueur de groupement : 2

les atomes : A, B^{-1} de rangs respectifs : 1, 2

les relations : A, B d'ordres respectifs : 1, - 1

Au groupement GR_2 sont associés

la longueur de groupement : 3

les atomes : B^2, C, D de rangs respectifs : 1, 2, 3

les relations : B, C, D d'ordres respectifs : 2, 1, 1

Dans GR_1 et GR_2 deux atomes B^{-1} et B^2 sont construits à partir de la même relation B.

Dans GR_1 le rang de B^{-1} est 2 = la longueur du groupement GR_1

Dans GR_2 le rang de B^2 est 1

Les groupements GR_1 et GR_2 sont donc composables.

On compose GR_2 par GR_1 et on obtient : $A\ B^{-1}B^2\ C\ D$

qui peut se simplifier en : $A\ B\ C\ D$

On composera toujours le groupement dans lequel l'atome a pour rang 1 par le groupement dans lequel l'atome a pour rang le nombre du groupement auquel il appartient. (nous notons gf = gof pour la composition de f par g)

Par composition de deux groupements de PHYLOG on peut obtenir :

- soit un nouveau groupement de PHYLOG qui sera traité comme un groupement avec simplification puis recherche des groupements composables.
- soit une relation de PHYLOG. Dans ce cas-là on compose les groupements de LOG de la même manière que ceux de PHYLOG et l'on obtient le groupement (ou la relation) de LOG associé (e) à la relation de PHYLOG.

Exemple :

Soit la table de traduction

LOG	PHYLOG
$A\,B^{-1}$	$S^{-1}\,U^{-1}\,V$
$B\,C$	S
C	$T\,U\,S$
$E\,F\,G^{-1}$	T^{-1}
$G^{n}H$	$U\,T$

- La deuxième ligne de la table de traduction nous donne directement une relation de PHYLOG, S, à laquelle sera associé dans LOG le groupement B C.
- La quatrième ligne de la table de traduction nous permet de trouver la relation T de PHYLOG.

A T^{-1} de PHYLOG était associé le groupement EFG^{-1} de LOG

A $T = (T^{-1})^{-1}$ de PHYLOG sera associé le groupement

$$G\,F^{-1}E^{-1} = (EFG^{-1})^{-1} \text{ de LOG.}$$

- Par combinaison des 1er et 3ème groupements de PHYLOG on trouve $T\,V = T\,U\,S\,S^{-1}U^{-1}\,V = T\,U\,U^{-1}V$

par combinaison de ce nouveau groupement T V de PHYLOG et du groupement T^{-1} on trouve la relation V de PHYLOG à laquelle est associée le groupement $EFG^{-1}CAB^{-1}$ de LOG

- Par combinaison des groupements U T et T^{-1} de PHYLOG on obtient la relation U de PHYLOG à laquelle est associé

le groupement $G^{n}HEFG^{-1}$ de LOG

on obtient finalement le tableau suivant :

PHYLOG	LOG
S	BC
T	$GF^{-1}E^{-1}$
U	$G^{n}HEFG^{-1}$
V	$EFG^{-1}CAB^{-1}$

Remarques

1) Il se peut que deux groupements de LOG puissent être traités de deux manières différentes; on les traitera donc deux fois.

Exemple :

Soient les deux groupements ABC^{-1} et CDA^{-1}.

Par composition ils donnent :

$CDA^{-1}ABC^{-1}$ c'est-à-dire $CDBC^{-1}$

et $ABC^{-1}CDA^{-1}$ c'est-à-dire $ABDA^{-1}$

2) Pour que, lorsqu'on obtient une relation de PHYLOG, on puisse déduire le groupement de LOG qui lui est associé, il faut garder les différentes étapes de composition (dans PHYLOG) pour pouvoir les appliquer (dans LOG).

3) Le programme d'évaluation de relations de PHYLOG que nous proposons n'est qu'un programme d'aide à la construction des tableaux, c'est-à-dire à la détermination de la correspondance LOG ⟶ PHYLOG.

Il n'a pas la prétention d'être complet. Il ne permet pas de trouver toutes les relations de PHYLOG. Dans la solution, telle que nous la proposons actuellement, le responsable du système doit encore éventuellement exprimer des relations de PHYLOG en fonction de groupements de LOG.

4) En fait, pour être plus complet, le programme d'évaluation de relations de PHYLOG devrait prendre en compte des critères de choix pour la composition des groupements. Ainsi, par exemple, deux groupements sont composables de deux manières; on obtient deux groupements qui sont eux-mêmes composables de deux manières et ainsi de suite. A chaque étape les longueurs des groupements obtenus augmentent. Dans un tel cas il faudrait prévoir des critères de "convergence", qui permettraient de limiter le nombre de compositions.

Exemple :

Soient les deux groupements ABA^{-1} ET ACA^{-1} ; ils sont composables de deux manières. On obtient les groupements $ACBA^{-1}$ et $ABCA^{-1}$.

Ce sont deux groupements à nouveau composables de deux manières. On obtient les groupements $ACBBCA^{-1}$ et $ABCCBA^{-1}$.

Ce sont deux groupements à nouveau composables de deux manières et l'on s'aperçoit que les longueurs des groupements obtenus augmentent à chaque étape.

5.3 Construction des "données affectées par PHYLOG"

En mémoire, on a la représentation des "données affectées par LOG". Il faut d'abord représenter en mémoire la correspondance LOG $\longrightarrow$ PHYLOG. Celà signifie qu'il faut représenter en mémoire les tableaux (que nous supposons construits avec l'aide des tables de traduction).

Les tableaux sont des listes d'éléments contigus. Chaque élément a une taille-mémoire de 2 unités mémoire. Dans la première unité mémoire se trouve une relation de PHYLOG. Dans la deuxième

unité-mémoire se trouve le groupement de LOG qui est le traduction du contenu de la première unité.

D'après l'hypothèse que nous avons rappelée au paragraphe 5.1.2, les types de repère de la structure PHYLOG sont identiques à ceux de la structure LOG. Leurs représentations en mémoire sont donc inchangées.

Il faut maintenant représenter en mémoire les relations de PHYLOG et pour celà utiliser la représentation en mémoire du tableau exprimant la correspondance LOG ⟶ PHYLOG.

- On crée une zone relations de PHYLOG notée REPHYLOG qui est de la même configuration que les zones relations créées précédemment et à laquelle sont associées des zones "départ", "arrivée", "traduction départ", "traduction arrivée".
- A chaque élément de REPHYLOG est associée une ligne du tableau et donc un groupement de relations de LOG.
- On cherche dans chaque groupement la relation de rang 1. L'ensemble (ou les ensembles) de départ de cette relation de LOG sera l'ensemble (ou les ensembles) de départ de la relation de PHYLOG. On créera donc un élément (ou des éléments) de la zone "départ", en l'associant à l'élément de REPHYLOG considéré. L'élément ainsi créé n'est que la copie de l'élément de la zone "départ" associé à la relation de LOG de rang 1 dans le groupement.
- De même on cherche dans le groupement la relation de rang supérieur. L'ensemble d'arrivée de cette relation de LOG sera l'ensemble d'arrivée de la relation de PHYLOG. On créera donc un élément de la zone "arrivée" en l'associant à l'élément de REPHYLOG considéré. L'élément ainsi créé n'est que la copie de l'élément de la zone "arrivée" associé à la relation de LOG de rang supérieur dans le groupement.

- Les contenus des zones "traduction départ" et "traduction arrivée" associées aux éléments des zones "départ" et "arrivée" (elles-mêmes associées à la relation de PHYLOG) sont obtenus par une procédure utilisant les zones "traduction" des ensembles de départ et d'arrivée des relations de LOG qui sont les constituants du groupement équivalent à la relation de PHYLOG.

Exemple

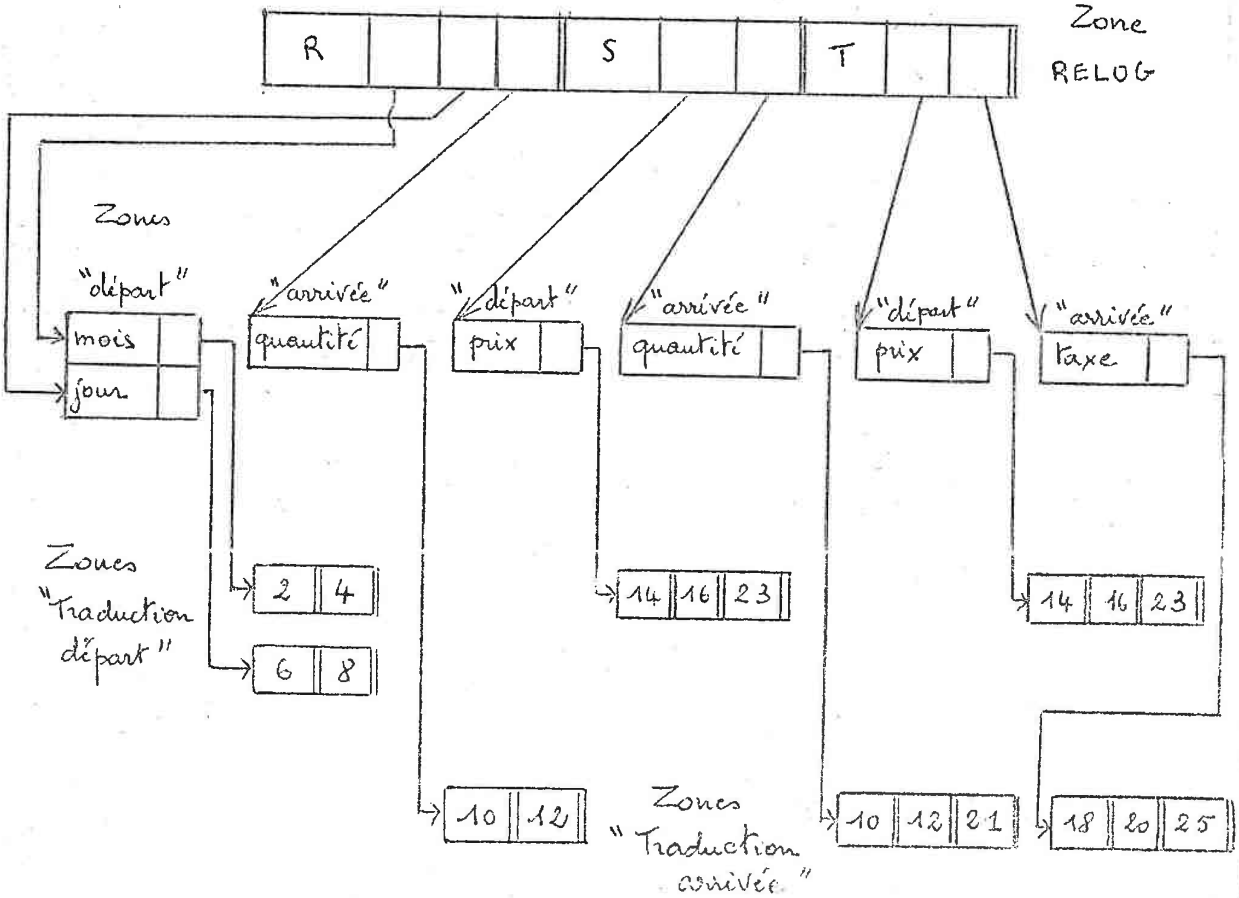
Soit l'extrait du tableau:

U	$RS^{-1}T$	

L'ensemble de départ de la relation U est celui de la relation R

L'ensemble d'arrivée de la relation U est celui de la relation T

Si, dans la représentation en mémoire des "données affectées par LOG", on a:



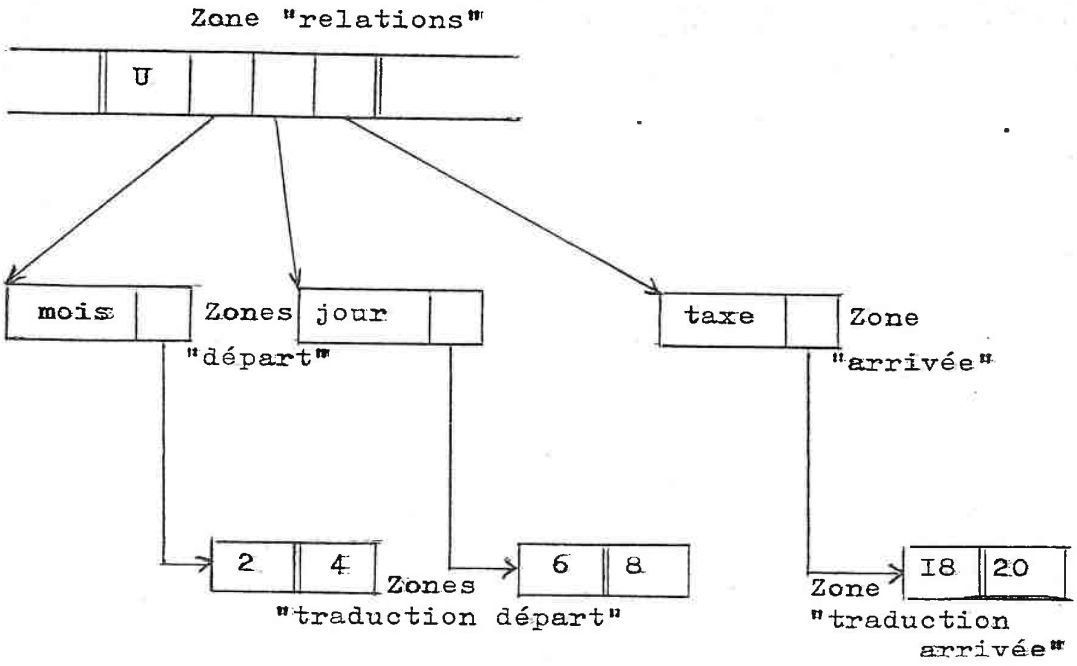
Si nous désignons par donnée(n) la donnée ayant le n-ième rang d'apparition dans la chaîne de données alors les différents constituants du groupe $RS^{-1}T$ trnsforment les données de l'ensemble de départ d'après le schéma suivant :

$$\Big(\text{donnée (2), donnée (6)}\Big) \xrightarrow{R} \text{donnée (10)} \xrightarrow{S^{-1}} \text{donnée (14)} \xrightarrow{T} \text{donnée (18)}$$

$$\Big(\text{donnée (4), donnée (8)}\Big) \xrightarrow{R} \text{donnée (12)} \xrightarrow{S^{-1}} \text{donnée (16)} \xrightarrow{T} \text{donnée (20)}$$

On obtient ainsi les zones "traduction départ" et la zone "traduction arrivée" associées aux ensembles de départ et à l'ensemble d'arrivée de U.

La zone "relations" associée aux "données affectées par PHYLOG" sera :



. Quand tous les éléments de REPHYLOG ont été ainsi traités on peut effacer les représentations des relations de LOG.

Nous avons donc vu successivement les trois phases.

La première phase, à l'aide du "structureur", définit la structure EXLOG associée à une description écrite dans le langage de description de la forme externe. Elle a pour résultat les "données affectées par EXLOG".

La deuxième phase transforme les "données affectées par EXLOG" en "données affectées par LOG". Elle utilise pour celà la correspondance EXLOG ⟶ LOG sous la forme d'un programme d'appels de procédures, procédures associées à chaque élément de LOG.

La troisième phase transforme les "données affectées par LOG" en "données affectées par PHYLOG". Elle utilise pour celà la correspondance LOG ⟶ PHYLOG sous la forme d'un tableau traduisant toute relation de PHYLOG en fonction de relations de LOG.

A l'issue de la troisième phase, nous avons donc les "données affectées par PHYLOG", c'est-à-dire la chaîne de données en entrée et la représentation en mémoire de la structure PHYLOG affectant ces données.

La structure PHYLOG décrit la "façon d'organiser les données en mémoire". Il reste maintenant à "utiliser" cette structure PHYLOG. "L'utilisation" de la structure PHYLOG affectant les données est ce que nous avons appelé la déformation unique de la chaîne de données.

Rappelons que l'appellation "déformation unique" est tout à fait justifiée puisque jusqu'à présent la chaîne de données est en mémoire telle qu'elle a été lue au départ. Aucune transformation n'y a été faite durant les trois phases précédentes.

Remarquons aussi que dans les programmes des trois premières phases aucune traduction de type de valeur n'a été faite. Il faudra

donc, lors de la déformation unique, tenir compte de tous les changements éventuels de codage des valeurs.

Dans le chapitre suivant nous allons donc étudier la déformation unique qui nous permettra alors de parler de données acquises.

CHAPITRE 6

La Déformation unique

6.1 Définitions

6.1.1 La déformation unique

La déformation unique transforme les "données affectées par PHYLOG" en données acquises.

La déformation unique n'est pas une correspondance entre structures. A une chaîne de données et à une structure PHYLOG les affectant, la déformation unique fait correspondre la représentation interne des données. Nous avons décidé de ne pas parler de structure dans le résultat de la déformation unique.

Remarque

La structure PHYLOG des données acquises n'est plus représentée en mémoire mais en fait elle est identique à celle associée aux "données affectées par PHYLOG".

6.1.2 Représentation interne

Les représentations internes des données sont, d'une manière plus ou moins complexe :

- les représentations internes des occurrences des types de repère.
- les représentations internes des graphes des relations.

Les façons "classiques" de représenter et d'organiser les données en mémoire sont nombreuses. Rappelons entre autres les listes (contigües ou chaînées), les piles, files, tables, tableaux.

Nous définierons les représentations internes des données sans utiliser ces "organisations classiques" mais en décrivant des parties de mémoire (unités mémoire, zones de travail....) et des applications entre ces parties et contenus (contenu, adressage....).

6.2 Résolution de la déformation unique : "les procédures d'acquisition

La déformation unique qui doit transformer les "données affectée par PHYLOG" en données acquises se fera par un programme activant des procédures de déformation que nous appellerons "procédures d'acquisition".

Nous allons voir successivement les ensembles sur lesquels les procédures travaillent, les différentes sortes de procédures et leurs buts, le programme les activant.

6.2.1 Les ensembles sur lesquels les procédures travaillent

Chaque procédure utilise au moins l'un des ensembles suivants :

- ensemble des types de repère noté TR
- ensemble des occurrences de types de repère noté OTR
- ensemble des unités mémoire noté M
- ensemble des contenus des unités mémoire noté C
- ensemble des adresses des unités memoire noté A (AcC)
- ensemble des applications suivantes :
  * contenu C : $M \longrightarrow C$

    qui, à une unité mémoire m, fait correspondre son contenu C (m).

* adressage m : $A \longrightarrow M$

  qui, à une adresse a, fait correspondre le mot m (a) d'adresse a (m est injective).

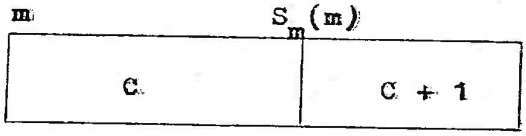
* succession de contenus S_c : $C \longrightarrow C$

  qui, à un contenu, c, associe un "successeur" S_c (C) noté aussi C + 1.

* succession d'unités mémoire S_m : $M \longrightarrow M$

  qui, à une unité mémoire, m, fait correspondre $S_m(m)$ qui est l'unité mémoire suivant m.

On a :



- ensemble des applications déduites des précédentes par combinaison :

Exemples :

mc : $M \longrightarrow M$, $x \longmapsto mc(x)$ qui fait passer d'une unité mémoire x vers l'unité mémoire dont l'adresse se trouve dans x.

cm : $A \longrightarrow C$, $y \longmapsto cm(y)$ qui fait passer d'une adresse y au contenu de l'unité mémoire d'adresse y.

cm_k : $A \longrightarrow C^k$, $z \longmapsto \langle cm(z), cm(z+1), \ldots, cm(z+k-1) \rangle$

qui fait passer d'une adresse z aux contenus des k unités mémoire d'adresses z, z+1, ...z+k-1.

Donnons tout de suite un exemple de procédure utilisant certains de ces ensembles.

On a 2 types de repères : nom, prénom ayant le même nombre d'occurrences. On veut représenter leurs occurrences par une liste d'éléments contigus avec un nom

suivi d'un prénom.

nom	prénom	nom	prénom	.....

Soit a l'adresse de l'unité mémoire contenant l'adresse de l'unité mémoire contenant la première occurrence du type de repère nom.

Alors les occurrences du type de repère nom sont les contenus des unités mémoire :

$$mcm\ (a + 2p)$$

Les occurrences du type de repère prénom sont les contenus des unités mémoires :

$$mcm\ (a + 2p + 1)$$

p prend les valeurs entières de 0 au nombre d'occurrences de chaque type de repère.

6.2.2 Les différentes procédures — Leurs buts

On distingue deux sortes de procédures :

a) les procédures attachées aux types de repère.

b) les procédures attachées aux relations.

Nous allons préciser leurs buts et nous les détaillerons sur un même exemple.

a) Les procédures attachées aux types de repère

Leur but est de définir les éléments de M (nombre, taille, succession ....) qui seront les "contenants" des occurrences des types de repère, et éventuellement de les "remplir".

Remarque

La définition de ces éléments s'effectue d'après un choix de représentation qui est supposé connu (et qui est fait d'après de multiples critères et notamment

les facilités de modification souhaitables). En effet la déformation unique n'a pas pour but de proposer un choix de représentations internes, ni d'aider à ce choix.

Exemple

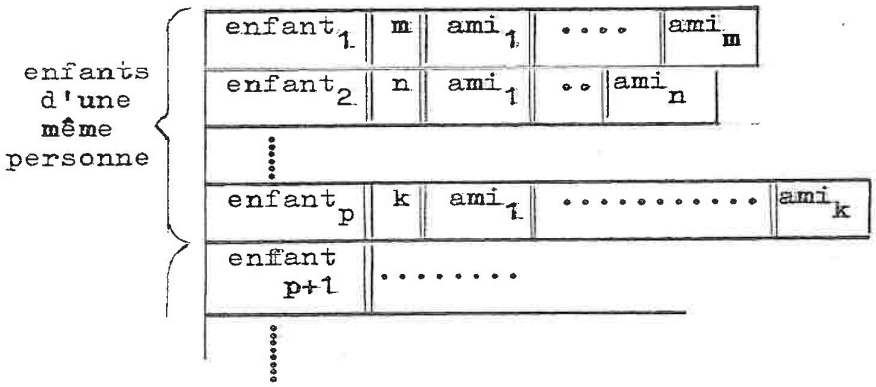
On suppose qu'au type de repère personne est associé le type de repère enfant par la relation parent qui, à une personne, fait correspondre éventuellement son (ou ses) enfant (s).

On suppose aussi qu'au type de repère enfant est associé le type de repère ami par la relation amitié qui à un enfant fait correspondre éventuellement son (ou ses) ami (s).

On veut représenter les occurrences du type de repère enfant par une liste d'éléments contigus. Chaque élément est constitué par la caractéristique de l'enfant, suivi des caractéristiques de ses amis (occurrences du type de repère ami associé à l'enfant par la relation amitié).

De plus on veut que les enfants de la même personne (associés à la même occurrence du type personne par la relation parent) soient représentés contiguëment.

Nous pouvons schématiser cette représentation par :



La procédure attachée au type de repère enfant précisera la forme de chaque élément de la liste d'enfants et la manière de les grouper. En fait cette procédure ne peut représenter en mémoire les occurrences du type de repère enfant. Celà ne se fera qu'après "l'activation" des procédures attachées aux relations parent et amitié.

b) Les procédures attachées aux relations

Leur but est de "remplir" les éléments de M définis par les procédures attachées aux types de repère et non encore remplis, et éventuellement de créer de nouveaux éléments de M et de les "remplir".

Exemple

Si nous reprenons l'exemple donné page 114, nous voyons que la procédure associée à la relation parent nous permet de remplir les premières unités mémoires des éléments de la liste des enfants (en les groupant par personne).

La procédure associée à la relation amitié permet de définir le nombre exact d'unités mémoire associées à chaque élément de la liste d'enfants puis de les remplir.

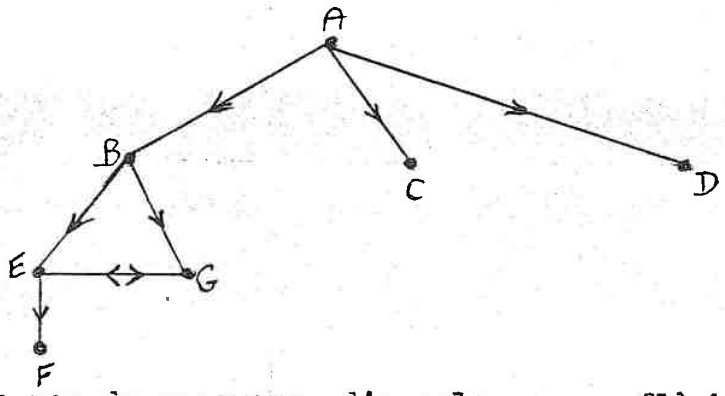
On s'aperçoit qu'en plus des différentes sortes de procédures la représentation interne des données demande un ordre d'appel de ces procédures. C'est le rôle du programme activant les procédures de définir cet ordre d'appel.

6.2.3 Le programme d'appel de procédures

On peut schématiser le programme d'appels de procédures par une arborescence d'appels.

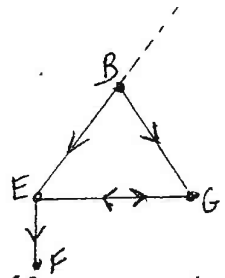
Nous allons donner les règles d'utilisation d'une arborescence en les illustrant sur un même exemple.

Exemple



- A est le nom du programme d'appel: aucune flèche n'arrive à A.
- Tout point auquel il arrive une (ou plusieurs) flèche (s) est une procédure.

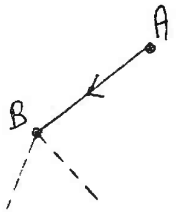
Exemple



B, E, F, G, sont des procédures.

- Pour des procédures se trouvant à des niveaux différents :
  - soit une flèche orientée de haut en bas les relie, alors la procédure du niveau le plus haut appelle la procédure du niveau le plus bas.

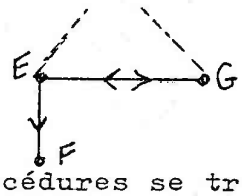
Exemple



A appelle B

- soit aucune flèche ne les relie, alors les deux procédures sont indépendantes.

Exemple

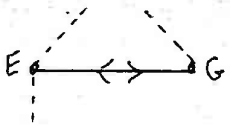


F et G sont indépendantes.

. Pour des procédures se trouvant au même niveau :

- soit une flèche double les relie alors les procédures sont imbriquées; chacune des deux appelle l'autre.

Exemple



E et G sont imbriquées.

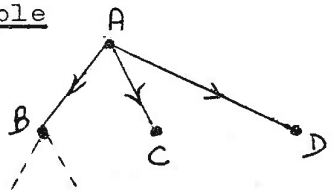
- soit aucune flèche ne les relie alors les procédures sont indépendantes.

Exemple

B et C sont indépendantes.

. Lorsqu'une même procédure appelle plusieurs procédures ne se trouvant pas au même niveau qu'elle-même, alors l'ordre des appels est celui de l'ordre d'écriture des procédures dans l'arborescence (de gauche à droite).

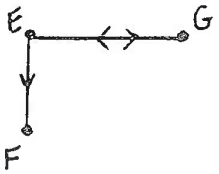
Exemple



A appelle B puis C puis D.

. Lorsqu'une même procédure appelle plusieurs procédures et que parmi elles il y a une procédure imbriquée cette dernière est appelée en dernier.

Exemple



E appelle F puis G.

Remarques

1) En plus des procédures attachées aux types de repère et aux relations il existe des procédures standards dont voici quelques exemples :
   - procédure de "compte" qui, associée à une procédure permet de connaître le nombre d'occurrences traitées par cette procédure.
   - procédure de "concaténation" qui transforme des chaînes d'éléments en éléments contigus.
   - procédure de "chaînage" qui transforme des éléments contigus en chaînes d'éléments.
   - procédure "d'ordonnancement" qui ordonne les occurrences d'un type de repère (en fonction des contenus d'autres occurrences par exemple)

2) Les représentations internes des données ne sont définitives que lorsque le programme d'appels de procédures est terminé; en effet une procédure appelée peut changer des représentations internes déjà existantes.

3) Lorsque l'on a des procédures imbriquées dans le programme d'appel, ce dernier utilise une zone de travail. Une zone de travail est constituée d'éléments chaînés qui seront les "contenants" des représentations internes des données associées aux procédures étudiées. Tant que les deux procédures ne sont pas terminées, il est fort utile d'avoir des éléments chaînés pour faciliter les modifications.

4) Sous-jacent aux solutions que nous ébauchons et aux remarques formulées se pose en fait le problème de

de la gestion de la mémoire qu'il serait intéressant d'approfondir.

Nous avons dit au début de ce paragraphe que le programme d'appels de procédures pouvait être schématisé par une arborescence et nous venons de donner les règles d'utilisation d'une telle arborescence.
Nous ne rentrerons pas plus dans l'étude générale de programmes d'appels.
Nous allons étudier maintenant en détail une déformation unique de données.

6.3 Exemple

Soit la structure PHYLOG définie par :

- les types de repère : pays, ville, population, groupevisiteur.
- les relations :

  appartient : ville, pays qui, à une ville, fait correspondre le pays auquel elle appartient.

  compte : ville, population qui, à une ville, fait correspondre le nombre de ses habitants.

  traverse : groupevisiteur, ville qui, à un groupevisiteur, fait correspondre la (ou les) ville (s) qu'il a traversée(s)

  visitépar : ville, groupevisiteur qui, à une ville, fait correspondre le (ou les) groupe (s) de visiteurs qui sont passés dans la ville.

Le choix de la représentation interne est le suivant :
liste de pays (formé d'éléments contigus), suivi de liste "d'éléments ville" (éléments contigus), suivi de liste "d'éléments groupevisiteur" (éléments contigus).

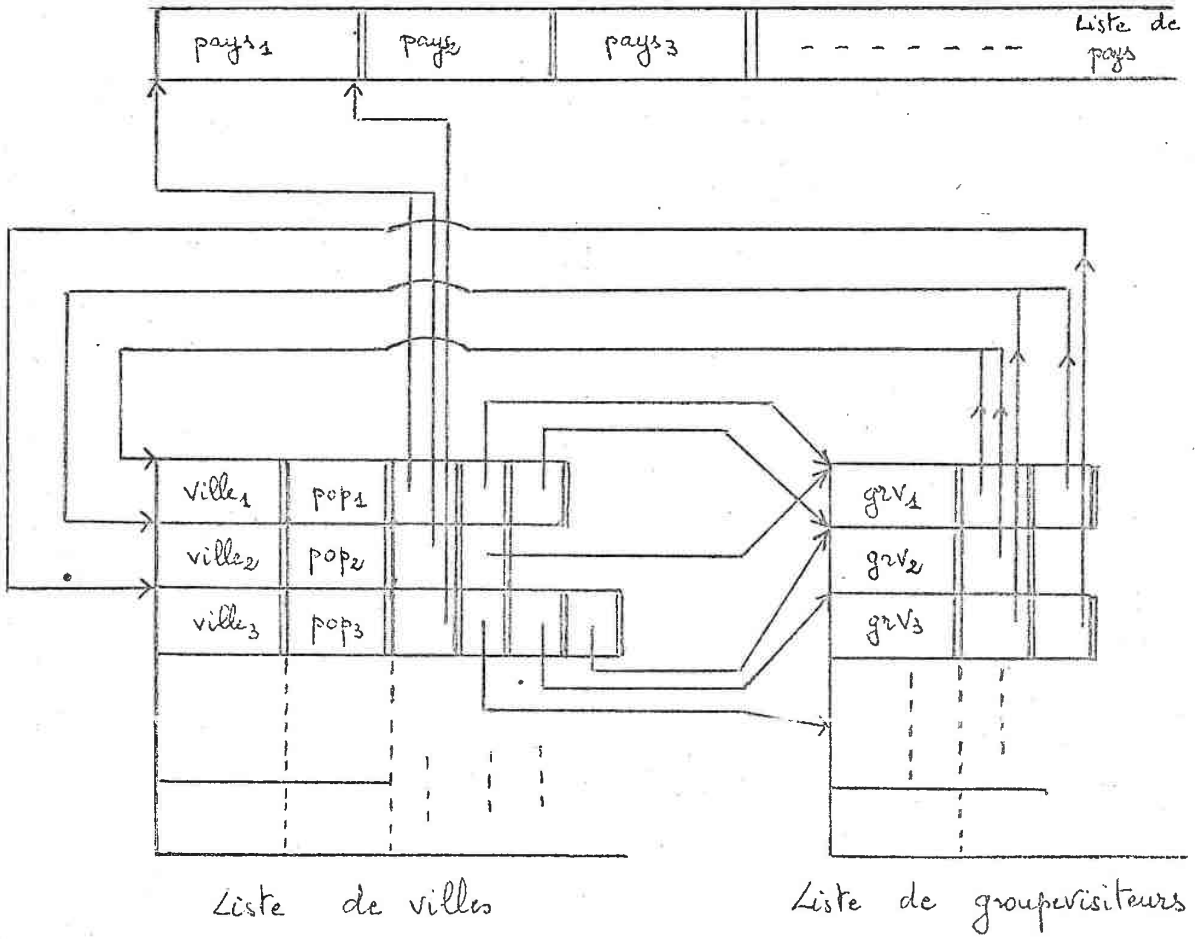
Chaque "élément ville" est constitué par :

- une unité mémoire contenant le nom de la ville.
- une unité mémoire contenant la population.
- un pointeur vers le pays auquel appartient la ville.
- un (ou plusieurs) pointeur (s) vers le (s) groupevisiteur (s) ayant visité la ville.

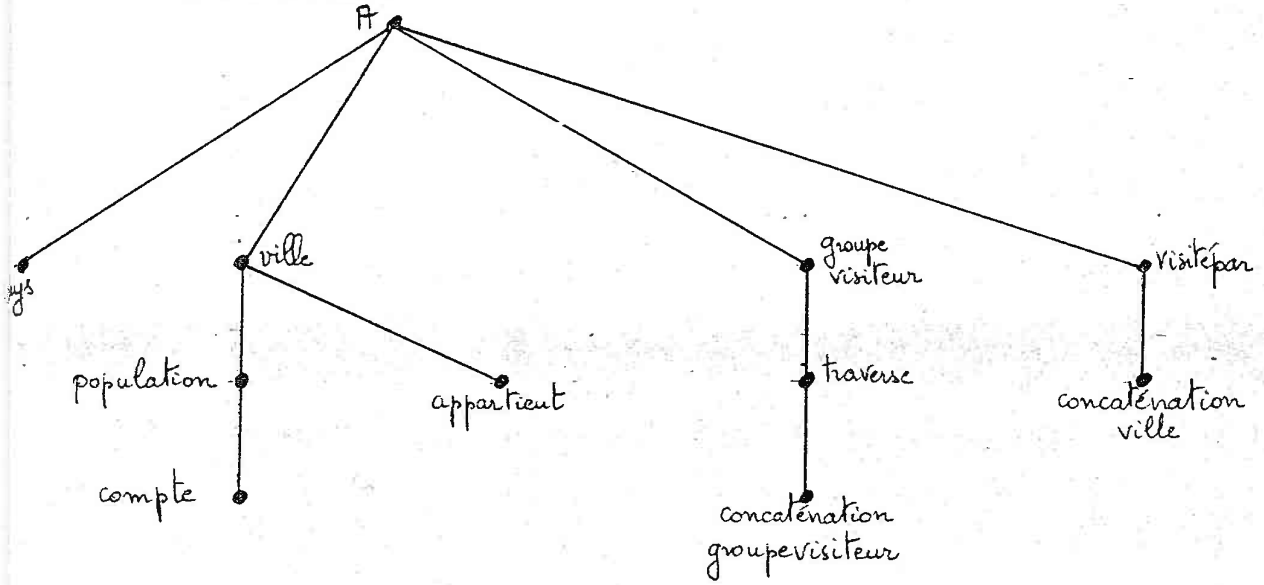
Chaque "élément groupevisiteur" est constitué par :

- une unité mémoire contenant la dénomination du groupe visiteur
- un (ou plusieurs) pointeur (s) vers la (ou les) ville (s) traversée (s) par le groupevisiteur.

Cette représentation en mémoire peut être schématisée ainsi :



Le programme d'appels de procédure est schématisé par l'arborescence :



Nous avons noté les procédures du nom de type de repère ou du nom de relation auxquelles elles sont associées.

On appelle, dans l'ordre les procédures :

pays, ville, population, compte, appartient, groupevisiteur, traverse, concaténation groupevisiteur, visitépar, concaténation ville;

Nous allons voir le but de chacune de ces procédures :

- la procédure pays définit les occurrences du type de repère pays comme les contenus d'une liste d'éléments mémoire.
- la procédure ville définit les "contenants" des occurrences du type de repère ville. Ce sont des éléments de taille-mémoire variable qui sont chaînés. Les premières unités de ces éléments sont "remplis" par les occurrences de type de repère ville.
- la procédure population définit les "contenants" des occurrences du type de repère population : ce sont les deuxièmes unités des éléments associés au type de repère ville.

- la procédure compte "remplit" les "contenants" créés par la procédure population. A chaque ville est associée sa population (déduite du graphe de la relation compte)
- la procédure appartient définit comme des pointeurs les troisièmes unités des éléments associés au type de repère ville. Elle les réalise en pointant le pays auquel appartient la ville (ce pays est déduit du graphe de la relation appartient)
- la procédure groupevisiteur définit les "contenants" des occurrences du type de repère groupevisiteur. Ce sont des éléments de taille mémoire variable qui sont chaînés. Les premières unités de ces éléments sont "remplis" par les occurrences du type de repère groupevisiteur.
- la procédure traverse définit comme des pointeurs les unités suivant les premières unités de chaque élément du type de repère groupevisiteur. Elle réalise un (ou des) pointeur (s) en pointant la (ou les) ville (s) visitée (s) par le groupevisiteur (ceci étant déduit du graphe de la relation traverse).
- la procédure concaténation groupevisiteur transforme la chaîne des éléments contenant les occurrences du type de repère groupevisiteur en liste d'éléments contigus.
- la procédure visitépar définit comme des pointeurs les unités suivant les trois premières unités de chaque élément du type de repère ville. Elle réalise un (ou des) pointeur (s) en pointant le (ou les) groupevisiteur (s) ayant traversé la ville (ceci étant déduit du graphe de la relation visitépar).
- la procédure concaténation ville trnasforme la chaîne des éléments contenant les occurrences du type de repère ville en liste d'éléments contigus.

Dans le dernier chapitre nous présentons une synthèse des solutions proposées pour résoudre le problème de l'acquisition.

CHAPITRE 7

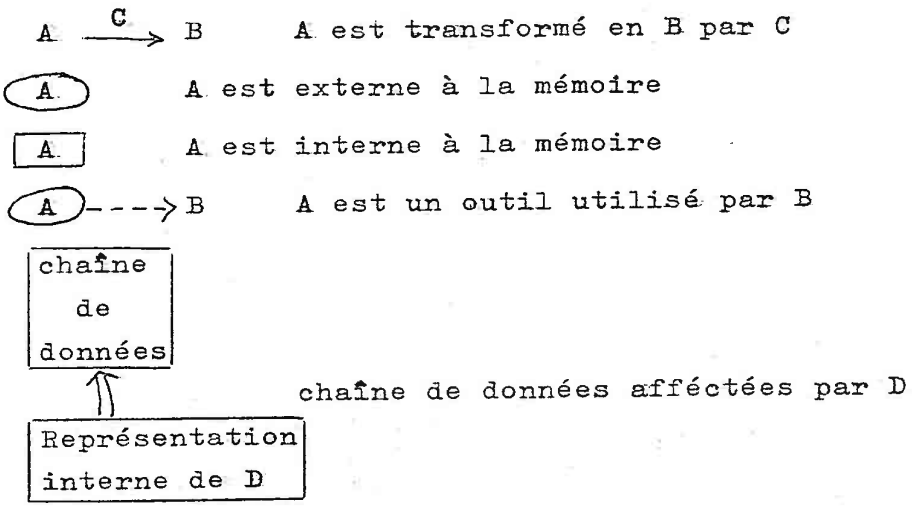
Synthèse

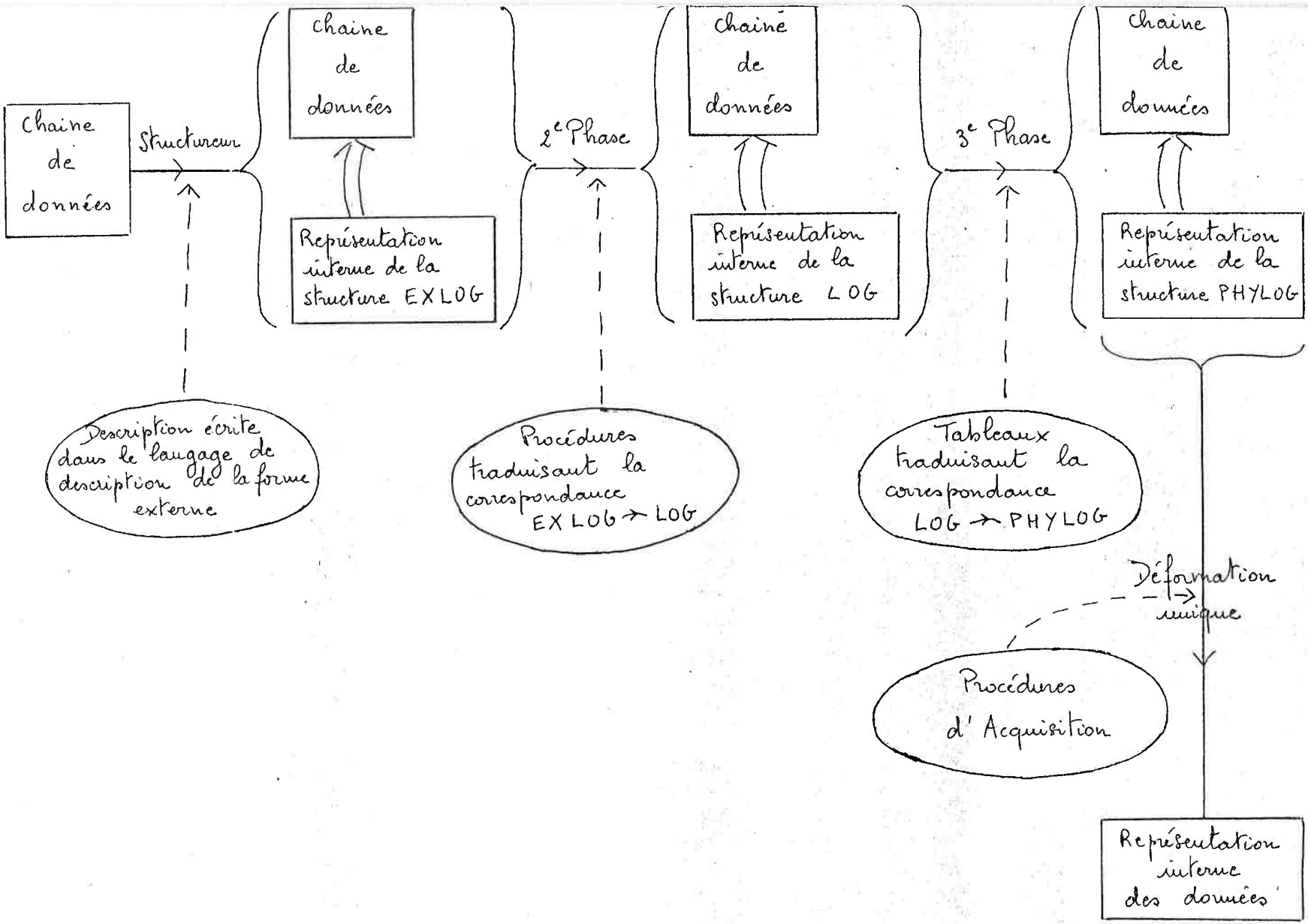
Dans le premier chapitre nous avions souligné que dans la solution proposée il y avait une symétrie par rapport à la structure LOG. Nous avions schématisé l'acquisition ainsi :

Forme Externe → Structure EXLOG → Structure LOG → Structure PHYLOG → Représentation interne

Nous allons schématiser le programme d'acquisition en faisant bien la distinction entre les informations se trouvant en mémoire et les informations externes à la mémoire.

Nous prendrons les conventions suivantes :




Chaine de données
Structureur
Chaine de données
Représentation interne de la structure EXLOG
2e Phase
Chaine de données
Représentation interne de la structure LOG
3e Phase
Chaine de données
Représentation interne de la structure PHYLOG
Description écrite dans le langage de description de la forme externe
Procédures traduisant la correspondance EXLOG → LOG
Tableaux traduisant la correspondance LOG → PHYLOG
Déformation unique
Procédures d'Acquisition
Représentation interne des données

Revenons sur la symétrie qui existe dans l'acquisition des donnée

D'une part : - La première phase part de la forme externe des données pour définir les "données affectées par EXLOG".

- De même symétriquement dans la déformation unique on part des "données affectées par PHYLOG" pour définir la forme interne des données.

D'autre part : - la deuxième phase transforme les "données affectées par EXLOG" en "données affectées par LOG" et pour celà elle utilise l'expression de chaque élément de LOG en fonction d'éléments d'EXLOG.

- la troisième phase transforme les "données affectées par LOG" en "données affectées par PHYLOG" et pour celà elle utilise l'expression de chaque élément de PHYLOG en fonction d'éléments de LOG.

Il y a donc symétrie de structures par rapport à la structure LOG mais, de plus, il y a similitude dans les solutions proposées comme outils permettant d'effectuer les phases.

CONCLUSION

Dans l'introduction, nous nous étions fixé comme but de cette étude d'essayer de résoudre le problème d'acquisition des données.

Le programme d'acquisition que nous proposons est encore une solution parfois lourde. Il est essentiellement un outil de travail

Son idée est la déformation unique obtenue grâce aux renseignemen "affectés" aux données.

Améliorer le langage de description de la forme externe mais surtout les solutions proposées pour décrire les correspondances entre structures et pour décrire la déformation unique sont autant d'objectifs qui permettront d'approfondir l'outil déjà construit.

A partir des solutions proposées, on peut essayer de décrire plus rigoureusement les correspondances et essayer de trouver une description standard.

De même on peut étudier le problème de la restructuration. Ne serait-il pas un nouveau problème d'acquisition avec une "forme externe" qui serait l'ancienne représentation interne ?

On pourrait aussi, bien sûr, étudier le problème général des modifications dont l'acquisition n'est qu'une partie.

D'une manière modeste, notre travail devrait apporter quelques nouveaux éléments de départ pour chacun de ces sujets de recherche.

BIBLIOGRAPHIE

(1) J. CEA "Cours de compilation"
Université de Nice .

(2) M. CREHANGE "Description formelle, Représentation, Interrogation des informations complexes. Systéme Pivoines"
Thése d'état (Nancy - 1975) .

(3) C. DELOBEL "Les systémes de base de données"
Ecole d'été de l'AFCET (Rabat - 1975)

(4) C. PAIR "Structure de données et algorithmes fondamentaux"
Université de Nancy .

NOM DE L'ETUDIANT : ANFRE Georges

NATURE DE LA THESE : DOCTORAT DE SPECIALITE EN MATHEMATIQUES APPLIQUEES

VU, APPROUVE

& PERMIS D'IMPRIMER

NANCY, le 17 juin 1977

LE PRESIDENT DE L'UNIVERSITE DE NANCY I

UNIVERSITÉ DE NANCY 1 - Le Président

M. BOULANGE