MASTER EN SCIENCES INFORMATIQUES

Outil interactif d'aide a l'évaluation et la classification de systèmes complexes

Fortemps, Joseph; Rome, Guy

Award date:
1978

Awarding institution:
Universite de Namur

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX

NAMUR

INSTITUT D'INFORMATIQUE

Année académique 1977-1978

OUTIL INTERACTIF D'AIDE A L'EVALUATION ET LA CLASSIFICATION DE SYSTEMES COMPLEXES

- JOSEPH fortemps

- GUY rome

- MÉMOIRE PRESENTÉ EN VUE D'OBTENIR LE GRADE DE LICENCIÉ ET MAÎTRE EN INFORMATIQUE

tome 1

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX

NAMUR

INSTITUT D'INFORMATIQUE

Année académique 1977-1978

OUTIL INTERACTIF D'AIDE A L'EVALUATION ET LA CLASSIFICATION DE SYSTEMES COMPLEXES

- JOSEPH fortemps

- GUY rome

- mémoire presenté en vue d'obtenir le grade de licencié et maître en informatique

tome 1

REMERCIEMENTS

Nous tenons à exprimer une profonde gratitude à toutes les personnes qui nous ont aidés à mener notre travail à bien, par leur direction, leurs conseils et leur aide:

Messieurs Ph. Van Bastelaer (Directeur du mémoire), J. Fichefet et J. Ramaekers (Professeurs de l'institut d'Informatique).

Qu'il nous soit également permis de remercier l'équipe "Grands Fichiers": J-L Hainaut, B. Le Charlier et W. Paulus, qui nous ont permis de clarifier les difficultés rencontrées lors de la réalisation de cette étude.

TABLE DES MATIERES.

TABLE DES FIGURES.

Page

REMARQUES.

Les renvois à la bibliographie sont précédés de la lettre "B".

Les renvois aux définitions des termes que nous employons dans le texte sont précédés de la lettre "D".

INTRODUCTION.

Plus que jamais, lorsque nous voulons acquérir un bien quelconque, se pose un problème de choix. Pour répondre aux exigences prévues, ce choix peut reposer sur toute une série de critères différents.

On remarque immédiatement que l'acquisition d'un objet complexe doit suivre un processus suffisamment planifié, afin d'assurer

- la garantie d'un choix aussi objectif que possible
- le moyen de contrôler et de maintenir l'étude préalable dans les limites de délai et de coût raisonnables.

En fonction de ces buts, "l'aide à la décision" vise à améliorer les processus de choix, de comparaison, d'évaluation et de classification. C'est pourquoi, en général, on (la direction) fixe des objectifs en fonction desquels une méthode d'étude et de choix seront définies. Parmi les phases intervenant dans un processus de sélection d'un objet, on retrouve notamment:

- l'élaboration d'un cahier de charges
- le lancement de l'appel d'offres
- le dépouillement des réponses
- la sélection.

Pour choisir dans un ensemble fini d'objets, en tenant compte d'un certain nombres de critères, non réductibles à un seul, il faut faire appel à des approches nouvelles pour remédier à des lacunes telles que:

- la connaissance imparfaite des préférences du décideur (D6)
- une information imprécise pour caractériser chaque objet selon chaque critère.

Parmi ces nouvelles approches, les méthodes Cat et Electre II, dont les développements théoriques et/ou pratiques ont déjà fait l'objet de mémoires aux Facultés N-D de la Paix (cfr B1 et B2), permettent de déterminer la classification et/ou les évaluations des objets.

Le but de ce mémoire consiste à créer un outil informatique permettant une introduction souple des données au niveau de ces deux méthodes de manière à les appliquer au problème de choix d'ordinateurs (D3). A cette fin, nous utilisons un langage interactif s'appuyant sur un système de gestion de base de données.

Pour des raisons méthodologiques, nous nous proposons de scinder ce travail en plusieurs parties:

1. - le dossier de conception qui permet de dégager la structure générale de la solution, et qui fait l'objet de ce volume,
2. - le dossier de développement qui définit de façon précise la solution informatique au plan logique (cfr Annexe),
   - le dossier de programmation qui définit les procédures techniques associées aux procédures logiques décrites dans le dossier de développement (cfr Annexe),
   - le dossier utilisateur qui contient les informations relatives à l'exploitation de la solution (cfr Annexe).

Au niveau de ce dossier de conception, nous nous intéressons spécialement à

- des généralités qui nous seront utiles pour la suite
- la situation du problème
- la détermination des solutions et le choix de l'une d'entre elles
- un planning de réalisation de cette solution
- l'avenir que l'on peut réserver à cette solution.

CHAPITRE I GENERALITES.

1.1 Problème de choix (D3).

En réalité, un système d'aide à la décision sous-entend souvent la présence

- d'un demandeur (D7): personne qui commande et juge l'étude
- d'un décideur (D6 et D2): personne pour laquelle et au nom de laquelle un homme d'étude travaille (le demandeur se confond avec le décideur, s'il ne doit satisfaire que ses propres désirs)
- d'un homme d'étude (D8) qui utilise ses connaissances scientifiques pour modeler le problème suivant les indications du demandeur (cfr B7).

Il importe que ces personnes restent en relation très étroites pour bien délimiter le contexte dans lequel le travail d'aide à la décision s'effectuera et pour mesurer les conséquences des décisions futures.

Dire qu'il y a aide signifie que ce travail a pour but d'éclairer la décision et non de la déterminer. C'est dans cette optique que l'homme d'étude devra affronter essentiellement trois sortes de travaux (cfr B7 et B14):

1. un travail d'analyse, visant à clarifier l'objet de la décision
2. un travail de modelage devant aboutir à un modèle des préférences du décideur et dont l'étude doit permettre d'éclairer la décision à prendre

3. un travail de résolution et d'interprétation comportant la collecte des données définitives, la mise au point des procédures de calcul et leur exécution, puis finalement, l'interprétation des résultats.

Il est donc nécessaire, pour une telle personne de comprendre la signification profonde du modèle, de même que les relations entre ce modèle et la réalité. Nous retenons à ce sujet, la définition d'un modèle apportée par B. Roy (cfr B7): "un modèle peut être défini comme un schéma qui est pris comme représentation abstraite d'une classe de phénomènes. Un objet abstrait est un modèle d'un objet concret lorsque la définition du premier est prise pour représentation du second".

Il existe actuellement une diversité de méthodes permettant la résolution partielle ou totale de l'évaluation et/ou la classification de systèmes complexes qui facilitent la tâche d'un homme d'étude; nous retenons:

- les méthodes qui déterminent, généralement par le bon sens, l'offre qui paraît la plus adéquate
- les méthodes à coéfficients
- la programmation linéaire
- la méthode cost-value
- les méthodes de classification
- les méthodes de simulation...

(cfr B1).

1.2 Processus de sélection.

Remarque préliminaire:

Il ne s'agit pas dans le cadre de ce paragraphe d'exposer une théorie sur un processus de sélection , mais plutôt d'attirer l'attention du lecteur sur certains aspects de ce problème; pour de plus amples renseignements, cfr B3, B4 et B5.

Un processus de sélection ne peut se concevoir sans l'application d'une méthodologie bien définie. Nous pouvons en effet distinguer un certain nombre d'étapes conceptuellement différentes, mais se succédant chronologiquement:

- spécification des besoins
- élaboration du cahier des charges
- examen des propositions
- méthode de sélection.

1.2.1 Spécification des besoins.

Il s'avère nécessaire de procéder à une analyse attentive des problèmes à résoudre, de manière à déboucher sur les exigences que le nouveau système devra respecter.

Lors de l'élaboration de cette analyse, l'homme d'étude mettra en évidence:

- les moyens existants
- les mesures de rentabilité
- les contraintes relatives aux moyens humains, techniques et financiers...

1.2.2 Elaboration du cahier des charges.

Lors de la rédaction du cahier des charges, l'homme d'étude doit être particulièrement attentif à certains points:

- les constructeurs doivent être clairement informés du contexte d'emploi du système: nous devons donc indiquer les performances minimales que le système devra respecter, sans multiplier exagérément les restrictions; en effet, le fait d'être intransigeant sur les performances souhaitées risque d'éliminer trop rapidement des constructeurs qui proposent malgré tout une configuration satisfaisante
- les constructeurs doivent être informés de manière précise et en particulier sur les caractéristiques devant servir ultérieurement de critères de choix; ils doivent également être avertis des modalités d'installation et d'acceptation du contrat
- les constructeurs acceptent généralement de tester une configuration par une série de programmes (appelés bench mark) spécialement conçus pour correspondre à un échantillonage représentatif de la charge du système futur
- il est à noter que dans bien des cas, toute cette analyse est réalisée soit par des sociétés spécialisées (offrant des services informatiques en tout genre) soit par les constructeurs eux-mêmes
- il est fortement conseillé d'adjoindre au cahier des charges un glossaire, car on rencontre de nombreuses divergences à propos des définitions, dans le jargon informatique des constructeurs.

1.2.3 Examen des propositions.

Afin d'éviter aux constructeurs de répondre à l'appel d'offres par une documentation volumineuse et difficilement maîtrisable, il n'est pas inutile de suivre l'exemple de la comission "Plan calcul" des Facultés N-D de la Paix: l'appel des propositions était accompagné d'un questionnaire très détaillé à remplir par chaque constructeur.

1.2.4 Méthode de sélection.

La sélection présente un problème très délicat, car si quelques critères de choix tel la taille mémoire, sont aisément quantifiables, il n'en est rien pour la plupart des autres: comment estimer une mémoire virtuelle par rapport à une mémoire réelle ?

1.3 Modèle de représentation d'un objet.

Un problème de sélection posé convenablement doit permettre la décomposition de l'objet sur lequel est porté le choix, en une hiérarchisation de critères (cfr fig. 1); aussi parle-t-on souvent d'approche multicritère. Le mot "multicritère" signifie qu'un problème est jugé selon plusieurs critères simultanément, c.à.d. que l'on cherche à réaliser plusieurs objectifs distincts en même temps. Ces critères risquent d'être incompatibles entre eux, du moins à partir d'un certain point. De plus si la décision relève d'un groupe de personnes, on peut supposer qu'elles ne percevront pas toutes le même problème de la même manière.

D'autre part, si la taille et la complexité d'un problème augmentent une hiérarchisation de critères s'impose. On obtient dès lors, des critères plus facilement quantifiables

- soit "cardinalement", c.à.d. par évaluation
- soit "ordonalement", c.à.d. par classification.

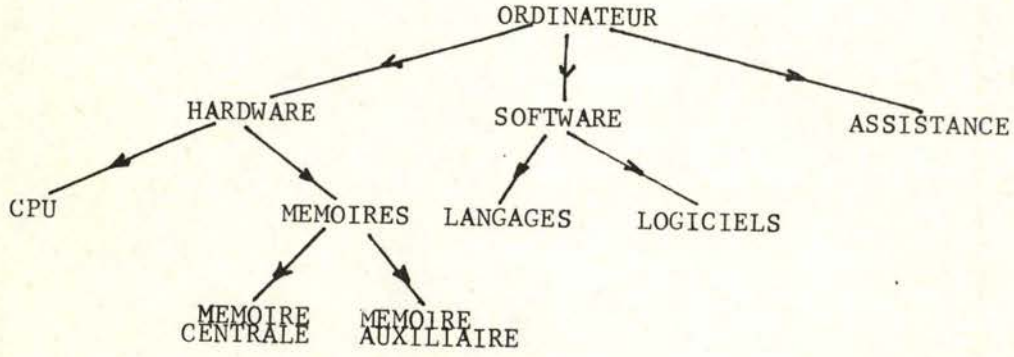


Fig. 1: décomposition "top-down" d'un ordinateur en critères

Une telle structure graphique est nécessaire pour l'exploitation de Cat, tandis qu'elle facilite grandement le dénombrement des critères utilisés dans Electre II.

Le graphe ainsi créé est constitué d'un ensemble de points appelés sommets, reliés entre eux par des branches orientées appelées arcs (cfr B6). On y distingue trois types de sommets:

- le sommet ascendant (D13) de tous les sommets du graphe (que nous appelons provisoirement racine)
- les sommets pendants: tout sommet ne possédant aucun suivant (D14) (que nous appelons provisoirement **feuille)**
- les sommets non pendants: tout sommet possédant au moins un précédent (D12) et un suivant; (nous l'appelons provisoirement: **noeud).**

On remarque qu'il n'existe aucun sommet isolé (D20) et tout sommet ne peut avoir au plus qu'un précédent (*). Ce graphe constitue donc une arborescence.

Suite à ces définitions, nous associons les notions de

- critère élémentaire à la notion de feuille
- critère non élémentaire à la notion de noeud
- critère global à la notion de racine.

On aboutit ainsi à une décomposition du graphe en un certain nombre de niveaux (cfr fig. 2): un niveau contient tous les sommets caractérisés par la même longueur de chemin entre eux et la racine.

(*) Il existe une exception qui est l'introduction de l'absorption partielle (cfr B1) dans le graphe associé à la méthode Cat; néanmoins, étant donné le caractère sporadique de ce cas, il sera traité au moyen d'une "acrobatie" (cfr dossier de développement).

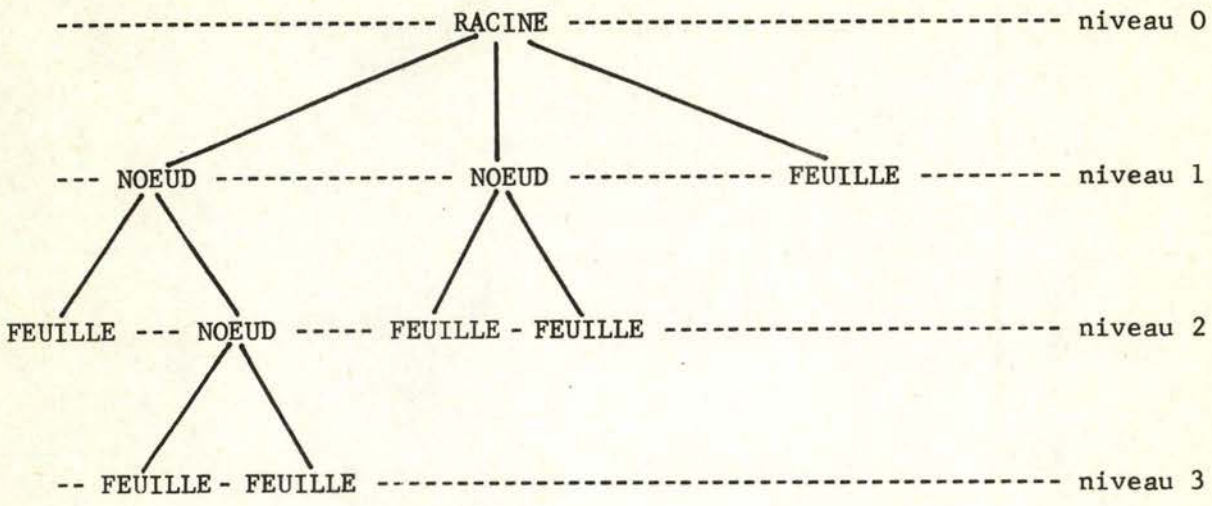


Fig. 2: décomposition d'une arborescence en niveaux.

1.4 Les méthodes Cat et Electre II.

Cat (*) et Electre II sont des méthodes de sélection qui permettent à des hommes d'étude d'évaluer et/ou de classifier des actions réalisables (D19), en tenant compte des critères retenus. Notre but n'est pas ici de les développer, mais de rappeler leurs grands principes.

Au départ, leur démarche est identique: on définit les critères par hiérarchisation et on passe ensuite au stade de pondération de ces critères (on leur donne un poids en fonction de l'importance qu'on leur accorde).

Elle est alors caractérisée par deux approches essentiellement différentes:

- la première (Cat) consiste, en partant des fonctions d'utilité (D15) associées aux critères élémentaires, à déterminer une évaluation du critère global, pour chacune des actions réalisables, par un processus d'aggrégation.
- la seconde (Electre II) essaye de déterminer un ordre (au sens mathématique du terme) entre les actions réalisables à partir des comparaisons de ces actions 2 à 2, pour chaque critère élémentaire.

(*) Nous reprenons par la suite l'abréviation Cat pour désigner la méthode "MAL" de J.J. Dujmovic pour l'évaluation et la comparaison de systèmes complexes.

1.4.1 Rappel de la méthode Cat.

La méthode Cat est caractérisée par plusieurs grandes fonctions (cfr B1):

- fonction d'évaluation
- fonction d'optimisation
- fonction d'analyse de sensibilité
- fonction d'actualisation.

Nous ne retenons à ce niveau que la première fonction citée, dont les différentes étapes sont:

- évaluation des critères élémentaires
- évaluation des critères non élémentaires
- comparaison et sélection des offres.

a. Evaluation des critères élémentaires.

Avant l'évaluation, il faut connaître les valeurs que l'offre du fournisseur apporte aux critères élémentaires. A partir de cette valeur d'entrée, ainsi que de la fonction d'efficacité (*) correspondant à chaque critère élémentaire, il est possible de calculer, une valeur d'efficacité associée à chacun des critères élémentaires (cfr fig. 3); c.à.d. le degré de satisfaction d'un besoin.

(*) Fonction d'efficacité: relation entre les valeurs d'entrée possibles (en abscisse) et leurs efficacités respectives (en ordonnée).

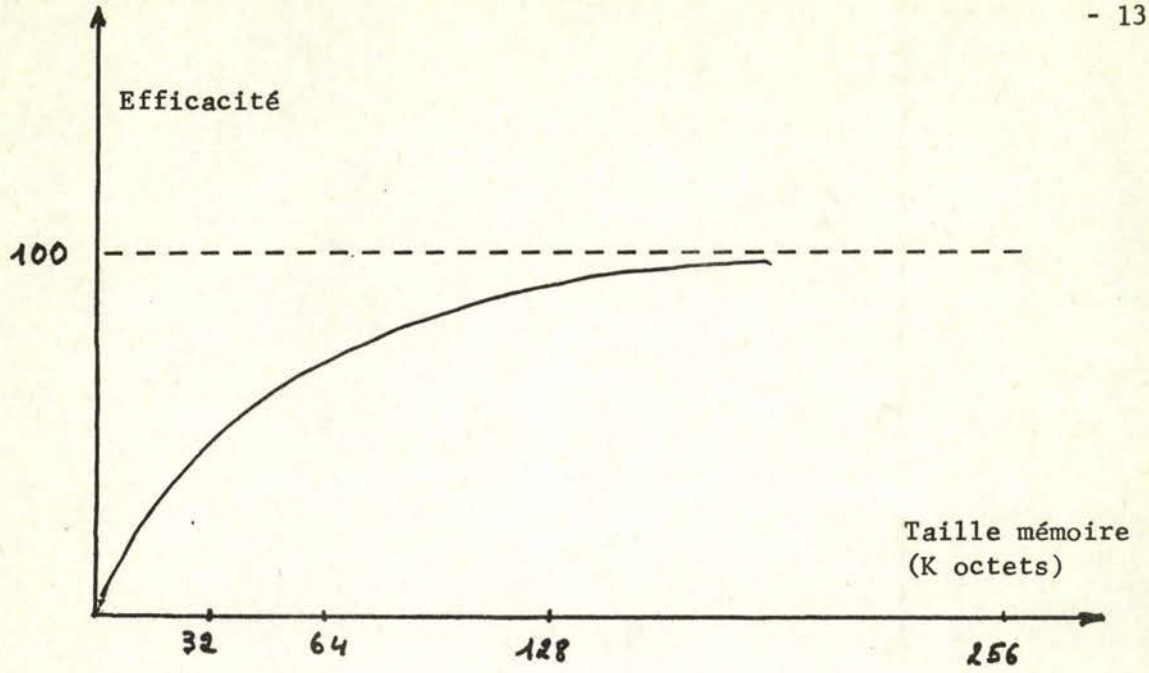


Fig. 3: exemple de fonction d'efficacité.

b. Evaluation des critères non élémentaires.

L'efficacité d'un critère non élémentaire sera calculée par une formule de moyenne généralisée où interviennent un opérateur d'aggrégation (*1), les efficacités des critères immédiatement subordonnés (*2) et leur poids. La formulation de cette moyenne est la suivante:

$$E = \left(\sum_{i=1}^{n} w_i \cdot E_i^{R} \right)^{1/R}$$

où E_i sont les efficacités des critères immédiatement subordonnés

w_i sont les poids des critères immédiatement subordonnés

avec $0 < w_i < 1$ et $\sum_{i=1}^{n} w_i = 1$

R est un coéfficient d'aggrégation: $R \in [-\infty, +\infty]$

On itérera cette procédure de manière à obtenir une valeur d'efficacité globale associée à la racine et donc, à une offre.

(*1) coéfficient d'aggrégation: ce coéfficient peut prendre certaines valeurs qui expriment les nuances dans la dépendance mutuelle des critères immédiatement subordonnés pour un critère non élémentaire; afin de faciliter la tâche de l'utilisateur, le choix d'une valeur du coéfficient d'aggrégation est remplacé par le choix d'un opérateur d'aggrégation. La liste de ces opérateurs ainsi que leurs explications ne sont pas reprises ici (cfr B1)

(*2) critères immédiatement subordonnés: ce sont les suivants d'un même critère.

Le poids d'un critère exprime son importance par rapport aux autres critères de même niveau avec lesquels il s'aggrège en un autre critère de niveau supérieur.

c._ _ Comparaison et sélection des offres.

Afin de comparer et de sélectionner des offres, on introduira certaines contraintes, par exemple:

- l'efficacité globale doit être supérieure à un certain seuil
- une contrainte de coût maximal.

Parmi les offres satisfaisant à ces contraintes, on sélectionnera par exemple celle qui maximise le rapport efficacité/coût, cfr fig. 4.

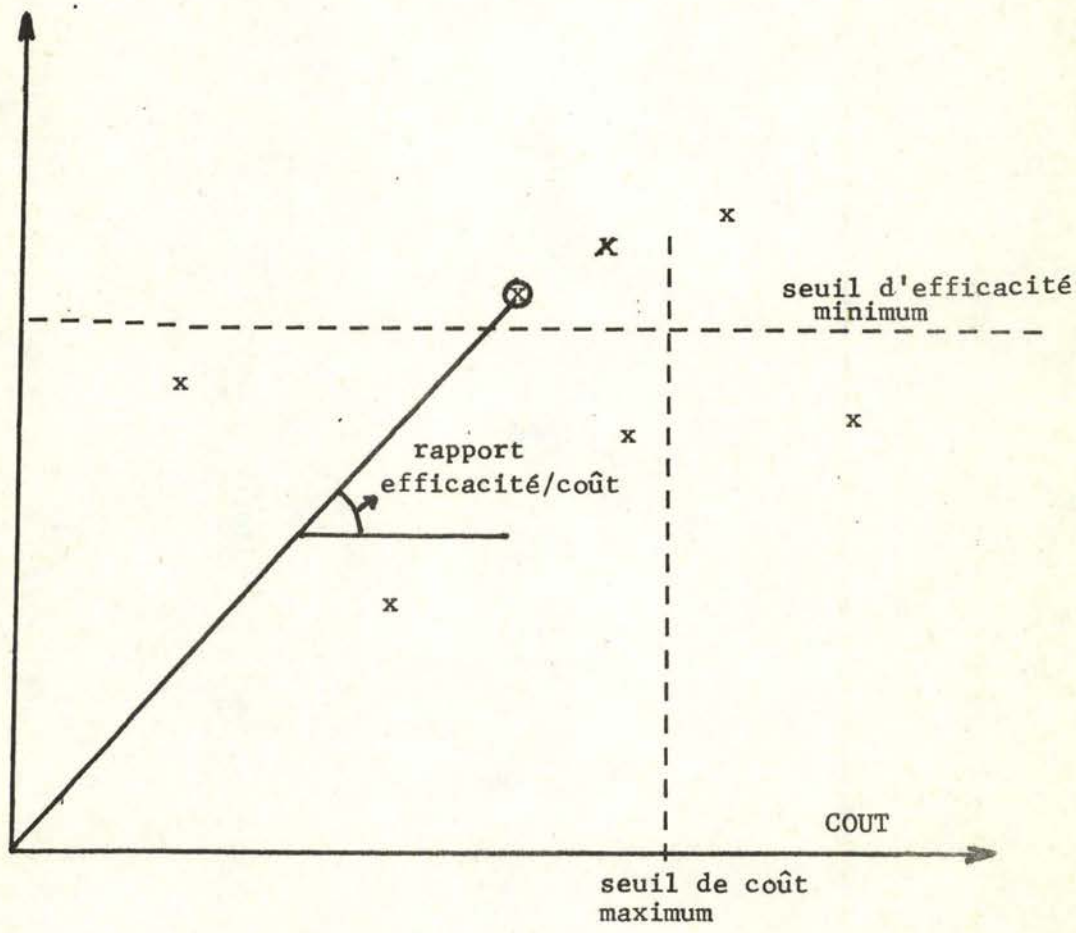


Fig. 4: exemple de diagramme coût/efficacité où chaque x représente une offre.

1.4.2 Rappel de la méthode Electre II.

On cherche à obtenir un ordre total (*) sur l'ensemble des actions réalisables (cfr B2, B8 et B18).

Le classement s'obtient en considérant uniquement les critères élémentaires pour lesquels on compare les évaluations des différentes actions.

Une première phase consiste alors à construire un graphe de surclassement (entre les différentes actions), à partir de la relation de surclassement.

Cette relation de surclassement peut se définir comme étant une relation binaire sur l'ensemble des actions réalisables telle que l'affirmation "x surclasse y", traduise une préférence de x relativement à y, suffisamment bien assise pour que l'on soit en droit d'accepter le risque de la considéré comme acquise. Elle peut prendre trois significations:

- surclassement fort
- surclassement faible
- incomparabilité.

(*) Une relation R définit un ordre total (ou complet) sur l'ensemble des actions réalisables (A), si

$\forall a \in A, \forall b \in B : \quad a R b \text{ ou } b R a$

Les notions de surclassement fort ou faible sont obtenues suite à des combinaisons de valeurs correspondant à des conditions de concordance et de discordance (cfr fig 4):

- la condition de concordance signifie que la somme des poids des critères où une première action est considérée comme meilleure ou égale à une seconde, est suffisamment élevée (c.à.d. $\geq$ seuil de concordance)
- la condition de discordance impose qu'il n'existe de critères où une première action est "trop nettement" inférieure à une seconde; (le terme "trop nettement" se traduit en pratique par les seuils de discordance); c'est pourquoi, on appelle souvent ceci la condition de non-discordance.

Si par contre, on ne peut pas aboutir à un tel graphe complet, on peut recourir à sa fermeture transitive (*); ceci risque de forcer quelque peu les préférences du décideur, en l'obligeant à rendre comparable ce qu'il ne veut pas, ne sait pas ou ne peut comparer.

Une deuxième phase consiste à trouver le classement multicritère à partir de ce graphe de surclassement:

a. classement direct où les sommets sont classés en fonction de la longueur des chemins incidents qui y aboutissent
b. classement inverse: un sommet est d'autant mieux classé que les chemins issus de ce sommet seront plus longs
c. classement médian: c'est la moyenne des rangs obtenus dans les classements direct et inverse

---

(*) Effectuer la fermeture transitive du graphe (formé de l'ensemble des projets et de la relation de surclassement) consiste à créer les arcs entre les sommets qui ne sont pas encore reliés par d'autres arcs.

Seuils de concordance:

$$1 > SC1 \geqslant SC2 \geqslant SC3 \geqslant 0$$

Seuils de discordance:

$$0 \leqslant SD1(K) \leqslant SD2(K)$$ K varie de 1 au nombre de critères

Comparaisons entre les projets I et J:

P^+ = somme des poids des critères pour lesquels $I > J$
$P^=$ = " " " " " " " $I = J$
P^- = " " " " " " " $I < J$

Soit $B = P^+ / P^-$

$C = P^+ + P^= / P^+ + P^= + P^-$

Table de décision:

B < 1		B ≥ 1			
	$\exists K$ tq $(VAL(J,K) - VAL(I,K) > SD2$	$\nexists K$.			
		C < SC3	SC3 ≤ C < SC2	SC2 ≤ C < SC1	SC1 ≤ C
				$\exists K$ tq $(VAL(J,K) - VAL(I,K)) > SD1$	$\nexists K$
Incomparabilité	Incomparabilité	Incomparabilité	SURCLASSE I FAIBLE	— FAIBLE	— FORT

Fig. 4': exemple d'établissement d'une relation de surclassement (cfr programme de choix rédigé aux Facultés).

1.4.3 Connexion Cat-Electre II.

Vu le caractère des méthodes que l'on vient de citer, nous remarquons que:

- pour la méthode Cat, il est nécessaire de décomposer l'objet sur lequel porte l'évaluation, de manière la plus fine possible, afin de déterminer des critères élémentaires aisément quantifiables (c.à.d., auxquels on peut attribuer facilement des fonctions d'efficacité)
- pour la méthode Electre, il semble peu réaliste de décomposer l'objet sur lequel porte la sélection, en un nombre trop considérable de critères.

Par conséquent, il pourrait être intéressant d'envisager une "connexion" entre ces deux méthodes: nous nous expliquons par un exemple (cfr fig 4").

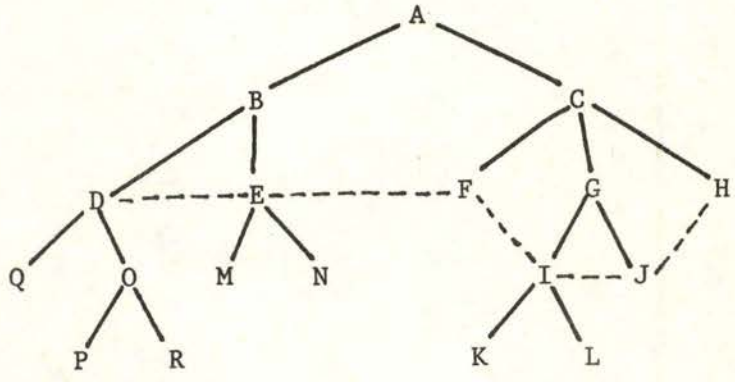


Fig. 4": Exemple de découpe arborescente, où Electre se limiterait aux critères D, E, F, I, J et H (considérés comme élémentaires) et Cat lui fournirait les évaluations des projets pour ces critères.

On pourrait penser que l'évaluation des projets pour chaque critère (c.à.d.: D, E, F, I, J et H) pris en compte par la méthode Electre serait déterminée par l'évaluation de ces critères au moyen de la méthode Cat (pour chaque projet).

CHAPITRE 2 SITUATION DU PROBLEME.

2.1 Description de l'existant (D5).

Nous tenons avant tout à rappeler que nous nous intéressons au problème de choix d'ordinateurs, à partir de l'utilisation des programmes de choix (D4) Cat et Electre II, qui sont disponibles aux Facultés N-D de la Paix.

Actuellement, l'exploitation de ces 2 programmes comprend:

- l'introduction des données
- le traitement proprement dit
- l'impression des résultats.

Ceci nous permet de préciser l'évolution du flux des informations dans un schéma élémentaire (cfr fig 5).

2.1.1 Introduction des données.

Les données d'entrée nécessaires au traitement de ces programmes de choix sont (cfr B1, B2 et B8)

pour Cat:

- les commandes et les instructions du langage SEL (*)
- la liste des critères élémentaires et la description des fonctions d'utilité associées à chacun de ses critères
- la liste des critères non élémentaires et leur coëfficient d'aggrégation, de même que la liste des critères immédiatement subordonnés ainsi que leur poids
- pour chacune des actions réalisables, la liste des valeurs d'entrée pour chaque fonction d'utilité

(*) Le langage SEL permet la programmation rapide de certaines fonctions de la méthode Cat.

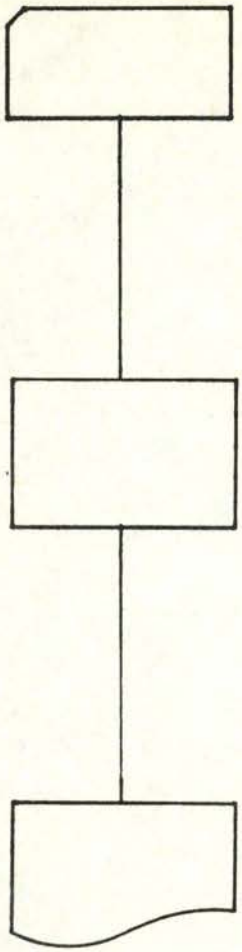
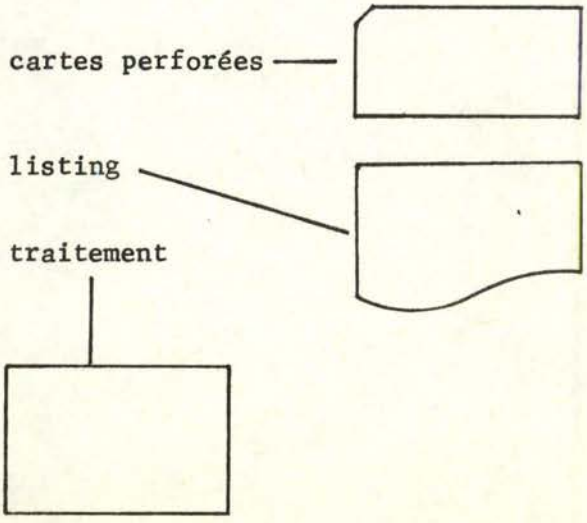


Fig. 5: Schéma du flux des informations dans le système existant.



pour Electre II:

- le nombre de projets, de même que leur libellé et leurs évaluations
- le nombre de critères et leur libellé
- le poids des critères
- les conditions de discordance et de concordance
- éventuellement, un facteur d'échelle des évaluations et une condition sine qua non.

Nous retrouvons donc des types de données communs aux deux méthodes:

- la structure arborescente des critères
- le poids des critères.

Nous parlons ici de types de données, car il n'est pas évident que la structure arborescente des critères, de même que les poids associés aux critères soient identiques, pour un même problème de sélection.

2.1.2 Traitement des données.

Comme nous l'avons déjà signalé dans l'introduction, nous nous intéressons ici à un outil informatique permettant de réaliser facilement l'introduction des données relatives aux programmes de choix, sans pour autant modifier ces programmes. C'est pourquoi, ce paragraphe ne sera pas développé davantage.

2.1.3 Impression des résultats.

Les résultats de ces traitements sont reproduits sur listings.

Nous avons ainsi pour Cat:

- les résultats de l'évaluation des critères élémentaires
- les résultats de l'évaluation des critères non élémentaires (et en particulier, du critère global)
- d'autres résultats sont également disponibles (cfr B1).

Dans le cas d'Electre II, nous retrouvons sur le listing de sortie:

- les résultats de comparaison des projets 2 à 2(éventuellement)
- le graphe de surclassement, présenté sous forme matricielle
- le résultat des classements direct et inverse, puis finalement, le classement médian.

2.1.4 Description des états d'entrée.

Etant donné que l'on s'intéresse essentiellement à l'introduction des données aux programmes de choix Cat et Electre, nous décrivons dans la suite, les états d'entrée relatifs à ces programmes.

Il existe un fichier d'entrée pour le programme de choix Cat, et trois fichiers d'entrée pour le programme de choix Electre: (cfr fig. 5' et 5").

Légende des figures 5' et 5":

- une ligne correspond à une carte (80 caractères)
- α : alphabétique
  αN : alphanumérique
  N : numérique
  P. : projet
  C.E.: critère élémentaire
  C.N.E.: critère non élémentaire
  C.I.S.: critère immédiatement subordonné

Fig. 5': description de l'état d'entrée relatif au programme de choix Cat.

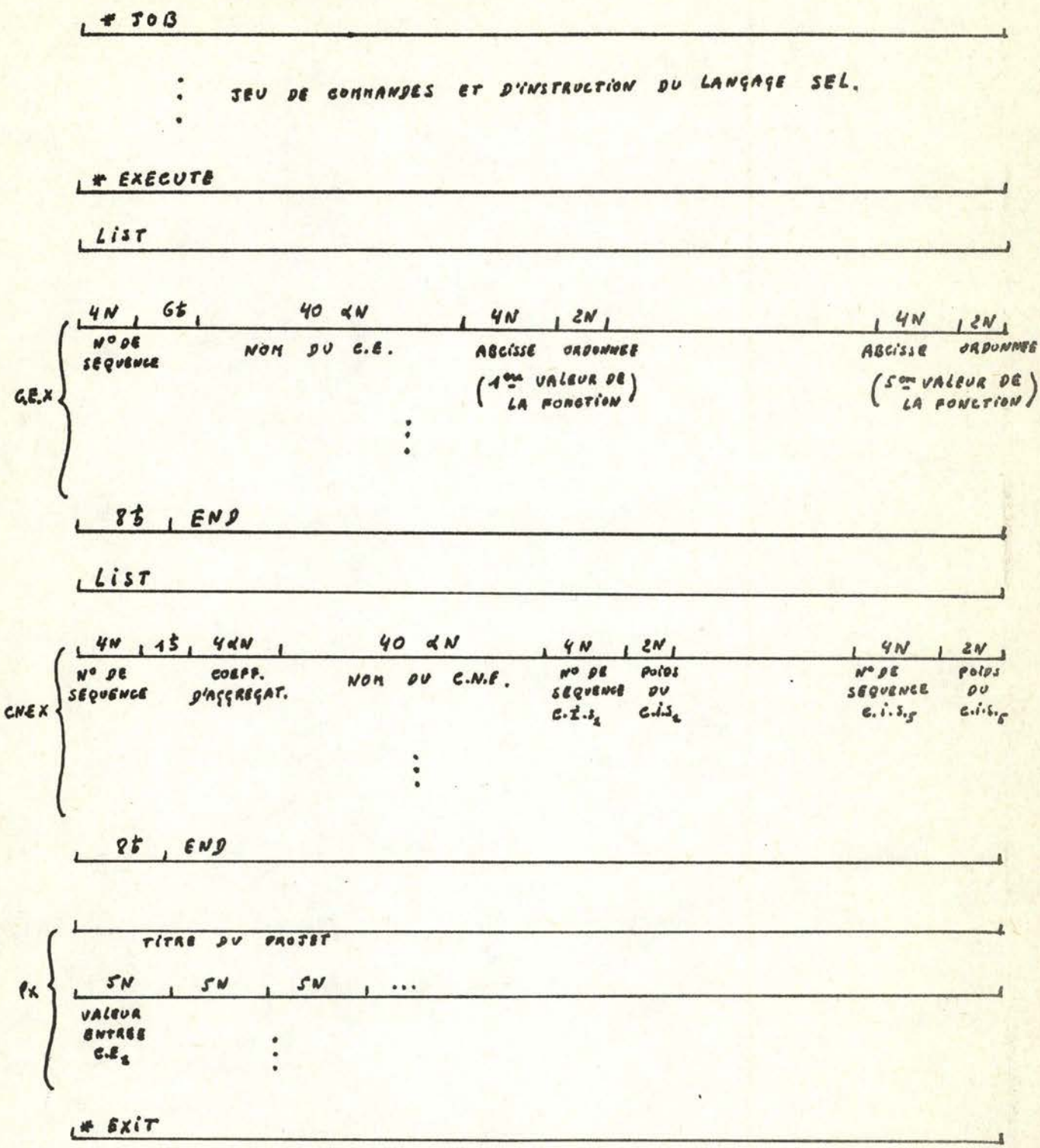
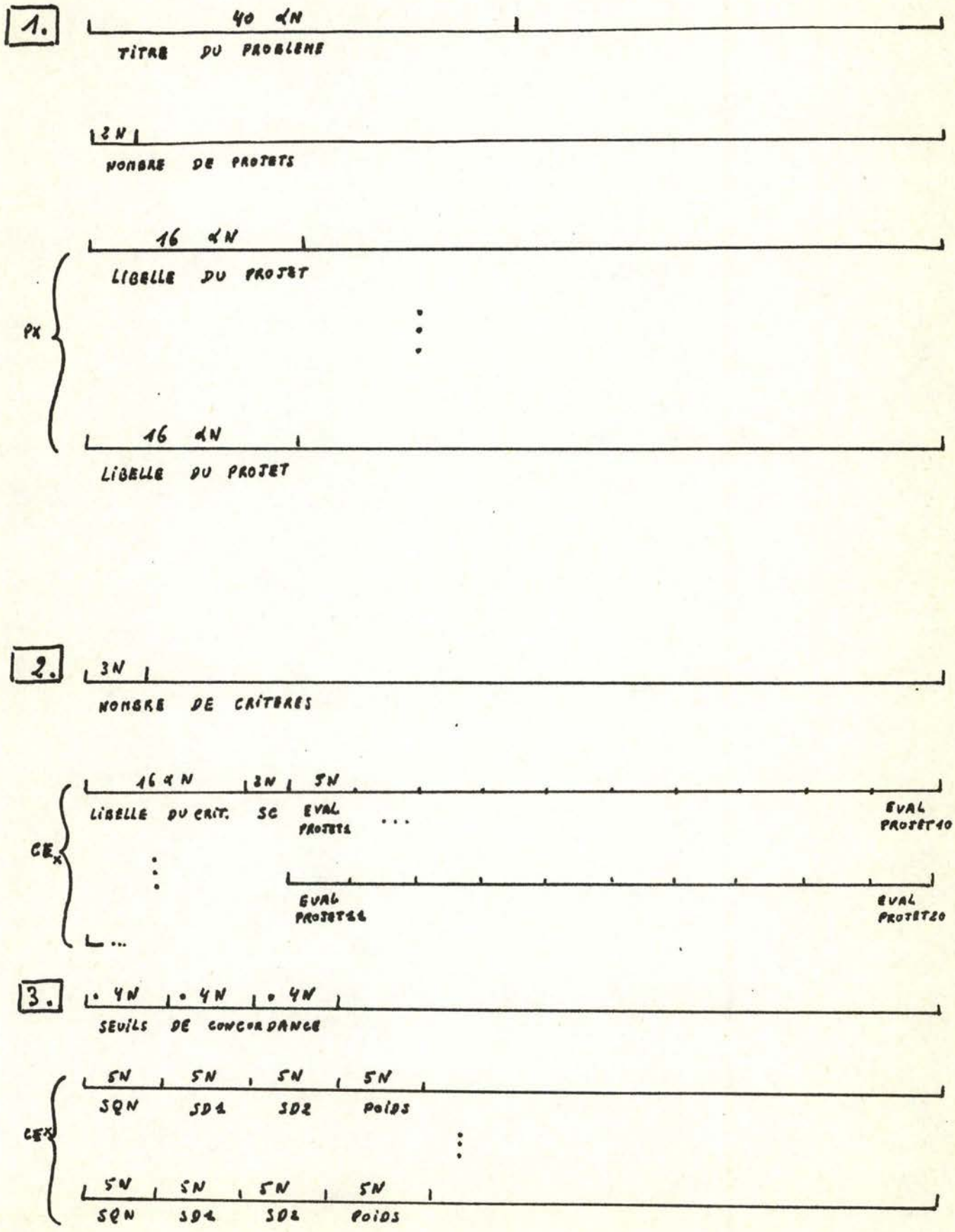


Fig. 5": description de l'état d'entrée relatif au programme Electre.



2.2 Critique de l'existant.

Suite à la description que nous venons de faire au sujet de l'existant, nous pouvons lui adresser certaines critiques.

Introduction redondante de données.

Si nous voulons tester les deux méthodes de sélection (Cat et Electre) avec en entrée des données "identiques", pour autant que ce soit réalisable, nous introduisons des informations de manière redondante. Ce sera en effet souvent le cas, ne fut-ce que si au départ, une sous-structure de hiérarchisation des critères est pareille pour les deux méthodes.

Contraintes de formatage des données., (cfr 2.1.4)

Le formatage à l'introduction des données est très rigoureux, ce qui la rend laborieuse, surtout lorsque les informations sont volumineuses. En effet, les valeurs des paramètres, par exemple, ne peuvent se situer qu'entre des colonnes bien déterminées. De plus, pour Cat, la liste des critères élémentaires doit s'introduire avant la liste des critères non élémentaires; de même, chacun des critères non élémentaires, ne doit apparaître dans cette liste que lorsque tous ses critères immédiatement subordonnés y sont déjà apparus.

Mauvaise connaissance de la structure des données.

Il est difficilement possible, à tout moment, de visualiser de manière claire et précise la structure des informations d'entrée: en effet, l'état d'entrée ne nous permet pas facilement de représenter la structure graphique de hiérarchisation des critères.

Temps de session trop long.

Le temps nécessaire au déroulement de toute une session de travail (perforation des cartes + introduction batch + traitement + réception des résultats) est assez long: de l'ordre de quelques heures.

Contrôle tardif et insuffisant des données d'entrée.

Une erreur, si elle est détectée, ne le sera que lors de sa prise en charge par le traitement. Dans le cas où elle n'est pas détectée, les résultats risquent d'être erronés ou le programme se termine prématurément.

2.3 Spécification du projet.

Il est intéressant au niveau pratique de connaître les limites des méthodes de sélection et de pouvoir les utiliser facilement. Ces méthodes sont en fait essentiellement employées par deux catégories d'utilisateurs:

- certains désirant en réaliser des études critiques (cfr B16)
- d'autres, dans un but de décision proprement dit.

On comprend aisément que l'élaboration du modèle de représentation d'un objet (cfr § 1.3) est primordiale pour l'exploitation des méthodes de sélection, dans le but de la décision (après dépouillement des offres), mais est aussi nécessaire au niveau de la formalisation du cahier des charges. En effet, la détermination de la structure arborescente des critères permet de faciliter la spécification des éléments intervenant dans le cahier des charges, ainsi que de leur importance.

Lors de l'emploi de ces méthodes, les utilisateurs espèrent trouver:

- une introduction et modification agréable des données (formatage)

- la possibilité de tester aussi rapidement que possible, la sensibilité de la méthode afin d'estimer le degré de fiabilité qu'il veut bien lui accorder dans son processus de sélection; ceci ne peut s'effectuer facilement que par un processus itératif: modification des données - traitement - analyse des résultats - décision de modification...

- une détection rapide des erreurs (notamment erreurs de formatage) lors de l'introduction des données

- la possibilité de visualiser les informations permettant ainsi à tout instant de connaître la situation existante; nous pensons spécialement à l'impression de la structure arborescente, de même que de certains paramètres.

En résumé, l'utilisateur souhaitera manipuler cette structure arborescente de même que tous les paramètres de la manière la plus aisée et la plus rapide possible.

Si nous avons actuellement deux programmes de choix (Cat et Electre) à notre disposition pour déterminer les choix d'ordinateurs, il n'est pas impossible que dans le futur, une ou plusieurs autres méthodes soient également prises en considération. Nous devons donc prévoir l'ajoute d'autres programmes de même type.

A partir du moment où l'on dispose de plusieurs méthodes de sélection, il est très utile de pouvoir comparer leurs différents résultats. Nous avons remarqué qu'il était intéressant de réaliser un "chapeau" commun à ces différentes méthodes, dans le but d'y rassembler les données et les paramètres nécessaires à l'exploitation d'une méthode de choix, quelqu'elle soit.

On remarque immédiatement que l'existence d'un tel "chapeau" permettrait la centralisation des données communes, ce qui est profitable surtout lorsqu'elles sont volumineuses.

Nous pouvons donc spécifier le schéma du flux des informations qui caractérise ce projet (cfr fig. 6).

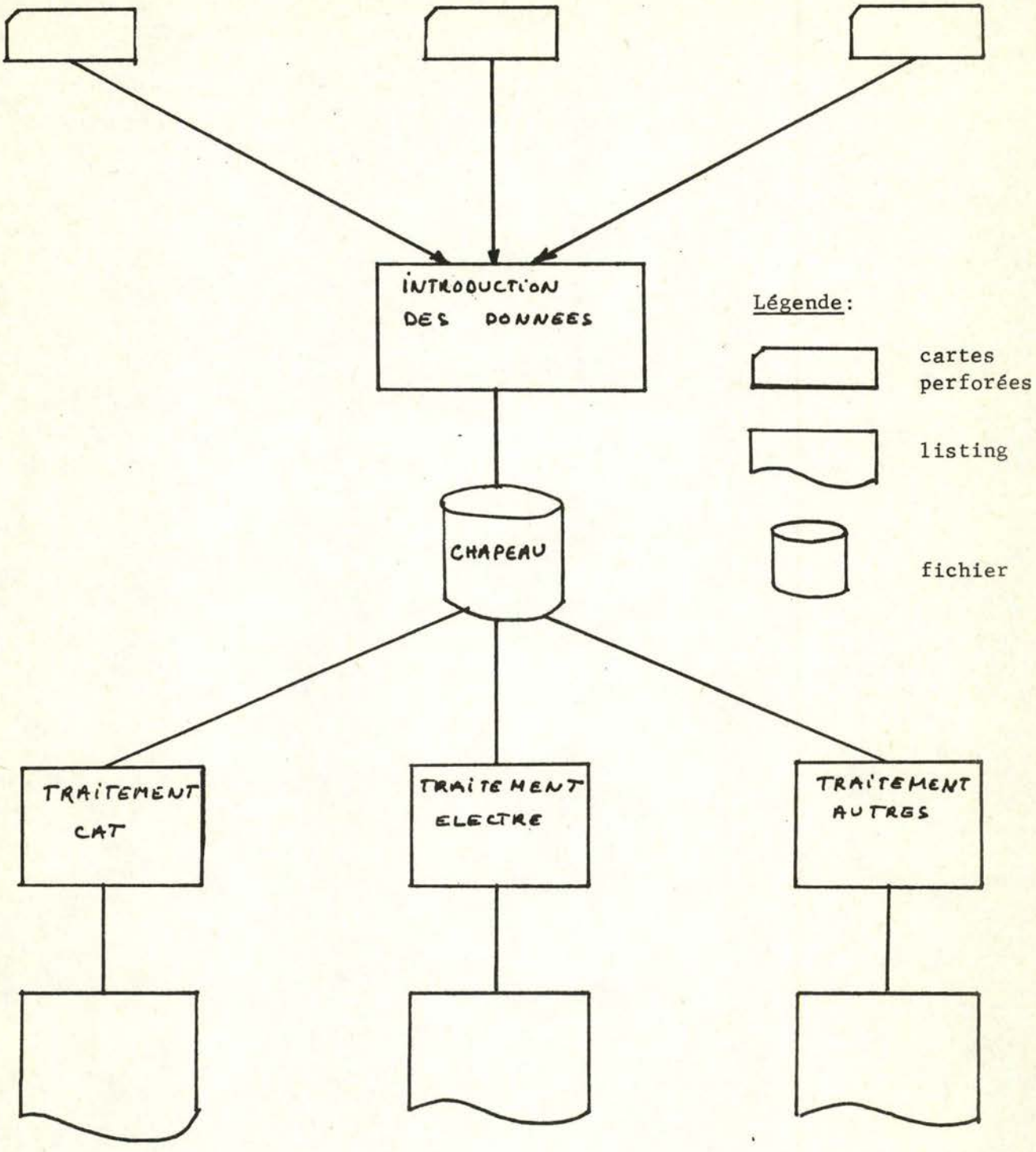


Fig. 6: schéma du flux des informations en introduisant le "chapeau" dans le système existant.

CHAPITRE III DETERMINATION DES SOLUTIONS ET CHOIX DE L'UNE D'ENTRE ELLES.

Comme nous l'avons signalé au § 2.3, l'utilisateur souhaite manipuler sa structure arborescente de manière la plus aisée et la plus rapide possible. C'est pourquoi nous nous sommes d'abord intéressé au problème de l'introduction des données (*).

3.1 Introduction des données par cartes (cfr chapitre II).

L'introduction des données par cartes est assez lourde. Cependant, il est possible de lui apporter un peu plus de souplesse, en rendant le formatage des données plus libre et donc plus agréable pour l'utilisateur; néanmoins cela complique considérablement le traitement de l'introduction des données. De plus, cela présentera toujours un inconvénient majeur pour la plupart des utilisateurs: le temps de session reste trop long.

3.2 Introduction des données au moyen de l'éditeur.

L'introduction des données au moyen de l'éditeur élimine l'inconvénient que nous venons de citer au § 3.1; il permet une introduction plus rapide des données en fonction des résultats obtenus.

Cependant, il est encore caractérisé par un formatage des données assez rigoureux. En effet, si la structure arborescente, ou de manière plus générale encore, l'ensemble des données doit faire l'objet de maintes manipulations (ajoutes, modifications, suppression...), cette méthode ne présente pas encore la solution idéale.

(*) Nous entendons par introduction des données, la création et la mise en forme de celles-ci de manière à ce qu'elles soient exploitables par les programmes de choix, sans modification de ceux-ci.

Nous nous rendons bien compte que toute liberté supplémentaire au niveau du formatage impliquerait des traitements nouveaux pour l'introduction des données.

De plus, il n'est pas facile de visualiser l'impact d'une modification (pris au sens large du terme) au niveau de la structure arborescente, ex: suppression d'une branche.

D'autre part, les données ne sont pas vérifiées avant l'exécution du traitement.

3.3 Introduction des données de manière interactive.

Un langage interactif permet évidemment de faciliter au maximum la tâche des utilisateurs en vue de l'exploitation des programmes de choix. Nous pensons spécialement au travail de mise en forme de la structure arborescente, notamment pour l'introduction des données, de même que toute manipulation au niveau de ces données.

Il devient dès lors très facile pour tout utilisateur de réaliser un certain nombre de fonctions qui sont par exemple: la création, l'impression et la modification des libellés des critères et de leurs paramètres:

a. création: on crée un critère ou une branche de critères (avec ou non leurs paramètres) de manière à obtenir l'arborescence désirée
b. impression: on visualise la structure arborescente, soit entièrement, soit partiellement suivant les désirs de l'utilisateur
c. modification: soit du libellé et/ou d'un ou de plusieurs paramètres de critères déjà créés, soit la suppression d'un critère, d'une branche ou de toute l'arborescence.

De plus, il s'avère dès lors permis d'effectuer un contrôle des données avant leur prise en charge par le traitement proprement dit et sans modification de ce traitement; on peut ainsi contrôler la validité des paramètres (syntaxe et sémantique).

Toutes ces fonctions sont réalisables, tout en gardant un temps de session très acceptable (de l'ordre du quart d'heure).

Il ne faut pas perdre de vue que la solution finale doit être retenue sur base de l'usage que l'on veut en réaliser. Or, comme nous ne nous trouvons pas dans le cas d'un utilisateur voulant employer ces méthodes de sélection de manière sporadique, mais bien, dans le cadre d'une université

- où certaines personnes sont soucieuses de tester la validité de ces méthodes de choix par
  - comparaison entre elles
  - confrontation des résultats avec la réalité
  - analyse de sensibilité...

  (cfr B16).
- où d'autres personnes sont plutôt intéressées par leur côté pratique (commission Plan Calcul),

nous avons choisi d'introduire les données de manière interactive.

N.B.: la liste des solutions proposées n'est certes pas exhaustive; cependant, nous pensons qu'elle est suffisamment complète pour justifier le choix de notre solution.

CHAPITRE IV REALISATION DE LA SOLUTION.

Nous trouvons intéressant le fait qu'un utilisateur quelconque puisse créer sa propre arborescence pour en retirer les éléments nécessaires à l'exploitation d'un programme de choix du type Cat ou Electre II.

De même, nous sommes persuadés qu'un langage interactif de manipulation offre en supplément des avantages déjà cités au § 3.3, d'autres avantages:

- la simplicité de ce langage de manipulation le rend accessible à des personnes ne possédant pas une connaissance approfondie de l'informatique
- ce langage de manipulation devient très utile si la structure arborescente varie dynamiquement en fonction du temps; nous entendons par là, le fait qu'un grand nombre de modifications devront être effectuées au niveau de la conception de l'arbre pour autoriser
  - des comparaisons des méthodes sur base de mêmes données initiales
  - des tests de validité des méthodes
  - des connexions éventuelles entre elles...

C'est la raison pour laquelle, nous avons créé un langage de manipulation permettant une efficacité et une souplesse souhaitées;(Le contenu de ce langage de manipulation fera l'objet de l'annexe); il permet donc l'introduction des données de manière interactive.

4.1 Premier degré de réalisation.

Une première étape de réalisation consiste à ce que chaque utilisateur

- décrive les données nécessaires à l'exploitation des programmes de choix
- crée les programmes de manipulation de ces données.

Ceci implique donc que l'utilisateur implémente lui-même la solution; cela nécessite un certain temps destiné à la réalisation et la mise au point des programmes, ce qui n'est pas toujours négligeable.

Cette réalisation devient inévitablement trop lourde pour un utilisateur moyen.

4.2 Deuxième degré de réalisation.

Une deuxième étape de réalisation consiste à effectuer l'implémentation à la place de l'utilisateur. Nous remarquons qu'il existe des données communes à chaque méthode de sélection (ex: le libellé des critères), par contre la plupart de ces méthodes sont caractérisées par des données spécifiques (ex: le coéfficient d'aggrégation est propre à Cat et n'intervient pas pour Electre).

Cette diversité des données n'est malheureusement pas propre à Cat où Electre, mais peut être généralisée à l'ensemble de ces méthodes. Etant donné qu'au départ, nous n'imposons aucune restriction quant à la nature ou au nombre des méthodes de sélection, il nous est impossible de prévoir les "inputs" à ces méthodes qui sont elles-mêmes du moins en partie, encore imprévues à ce jour.

Dans le but d'utiliser une terminologie, nous rappelons qu'un langage de base de données (*1) peut se décomposer en deux parties:

- le langage de définition des données (L.D.D.) définissant les types de données qui constitueront le schéma de la base de données
- le langage de manipulation des données (L.M.D.) formé d'un ensemble d'ordres (relatifs à des traitements) agissant sur les types de données.

Afin de ne pas confondre ce dernier langage de manipulation des données (L.M.D) avec le langage de manipulation constitué d'un ensemble de commandes permettant à un utilisateur de créer sa structure arborescente, nous appellerons le premier: L.M.D. et le second: langage de manipulation.

De même par la suite, nous parlerons de programmes L.L.D. et de programmes L.M.D., qu'il ne faut pas confondre non plus avec les L.L.D. et L.M.D.; en effet, les premiers (*2) sont écrits à partir des langages que sont les seconds.

(*1) A ce stade, nous employons le terme base de données pour un ensemble d'informations muni de certaines structures (et indépendant du support physique)

(*2) La définition des données ainsi que les manipulations de ces données peuvent intervenir dans un seul programme.

Notre optique de généralisation à un ensemble a priori illimité de méthodes de sélection implique lors de la création d'une base de données (plus exactement, au niveau du L.D.D) que l'on arrive à déterminer un schéma type de base de données qui soit suffisamment général pour englober tous les sous-schémas correspondant à chaque utilisateur particulier (cfr B11).

Il est possible de créer un schéma de base de données suffisamment général pour contenir les données de Cat et d'Electre, mais nous ne pouvons garantir qu'il puisse accepter des données spécifiques à une autre méthode.

Cependant, nous pouvons atteindre un certain niveau de généralité du point de vue

- du programme L.D.D: on peut par exemple définir des données pour satisfaire certaines méthodes de sélection; néanmoins, nous ne pouvons a priori prévoir la définition des données pour toutes les méthodes
- du programme L.M.D.: on peut déterminer quelques fonctions de manipulation, mais rien ne nous garantit qu'elles nous satisferont toujours.

A ce stade, nous pouvons concevoir une base de données sur laquelle agirait un langage de manipulation en "input" et à partir de laquelle on sortirait en "output" les données mises en formes pour chaque méthode de sélection, par un interface qui lui serait propre (cfr fig. 7).

Ceci permet donc à certains programmes de choix de profiter de l'existence d'une base de données pour y puiser les informations requises (si elles existent) et moyennant éventuellement un interface.

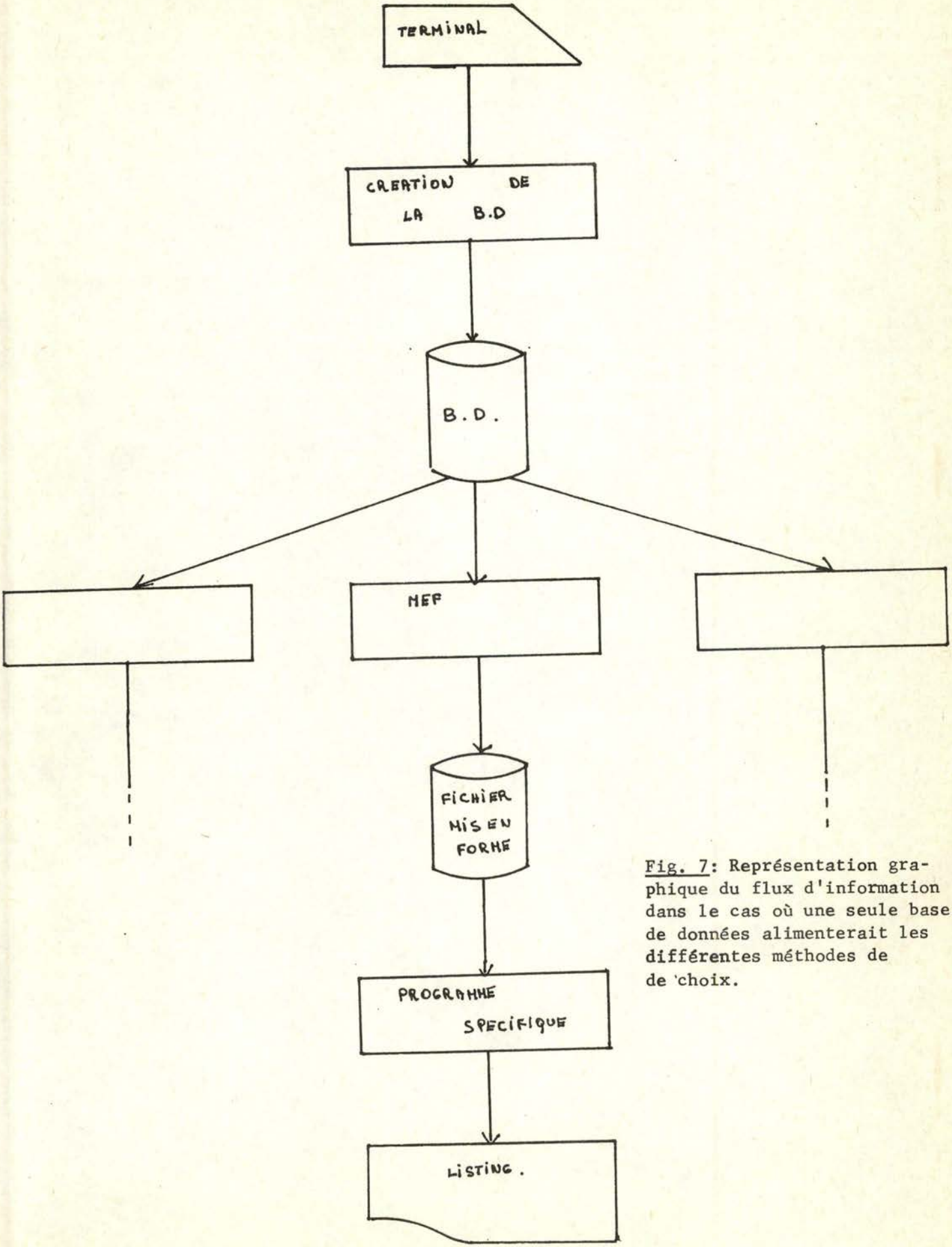


Fig. 7: Représentation graphique du flux d'information dans le cas où une seule base de données alimenterait les différentes méthodes de de choix.

Néanmoins, l'existence au préalable de ces informations dans la base de données n'est possible que par l'intermédiaire d'un programme L.D.D. et d'un programme L.M.D. bien définis, ce qui risque d'impliquer certaines restrictions parmi ces autres méthodes de sélection.

La non existence de ces informations dans la base de données nécessite une modification du programme L.D.D. et/ou du programme L.M.D. Nous remarquons donc qu'une telle situation, bien que non déplaisante au départ, nous impose des restrictions que nous ne pouvons surpasser que par une revision complète du problème.

4.3 Troisième degré de réalisation.

4.3.1 Description de la réalisation.

Nous avons envisagé la situation sous un autre angle, en remarquant que la plupart des méthodes de sélection se basent sur une hiérarchisation de critères (cfr § 1.3). Nous en concluons donc qu'elles possèdent au moins comme type de données communes, le libellé des critères de choix.

Cette analyse donne suite à l'idée suivante: pourquoi ne pas regrouper ces informations communes dans une base de données générale à laquelle viendrait se "greffer" toute une série de bases de données particulières à chaque méthode de sélection (cfr fig. 8), contenant chacune les informations qui lui sont propres.

Nous remarquons donc que la base de données générale peut être manipulée par n'importe quel utilisateur et qu'elle donne accès, au moyen d'un interface, à chacune des bases de données spécifiques aux méthodes de sélection. Ces dernières seront reliées aux programmes correspondants par l'intermédiaire d'autres interfaces permettant la mise en forme des données de manière à les introduire dans ces programmes.

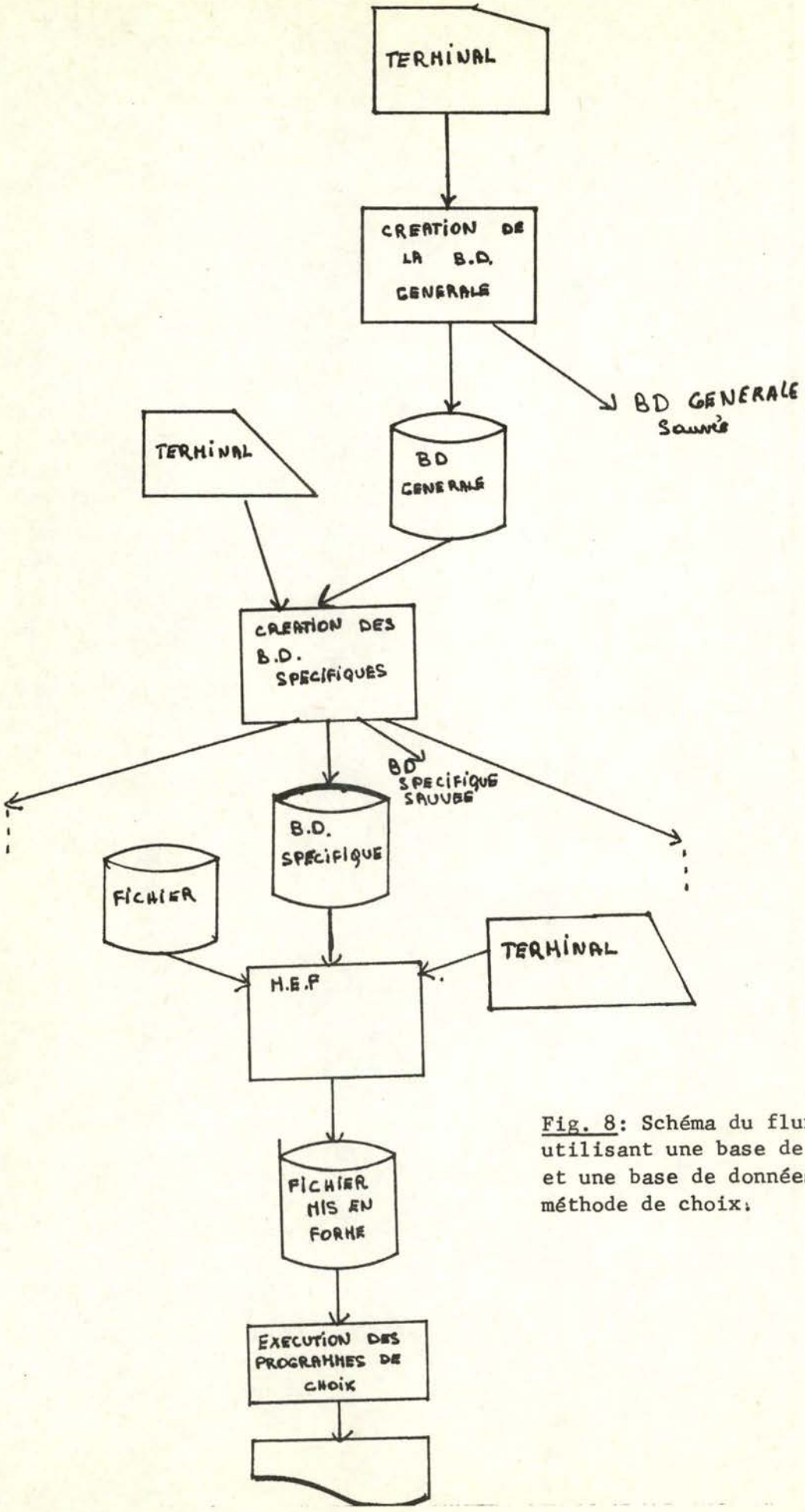


Fig. 8: Schéma du flux d'information en utilisant une base de données générale et une base de données propre à chaque méthode de choix.

Chaque base de données spécifique doit pouvoir être garnie indépendamment de toute autre, par un langage de manipulation écrit en fonction de la nature de chaque base de données spécifique.

Nous constatons que cette nouvelle vision du problème contrecarre les restrictions de l'ancienne (cfr §4.2).

Au niveau de l'introduction des données, la structure de hiérarchisation des critères peut être partiellement ou totalement identique. Les avantages que l'on retire de ce type d'introduction sont que l'on ne crée qu'une seule fois et de manière identique la structure hiérarchique commune.

Dans le but des comparaisons et/ou des interconnexions entre plusieurs méthodes, les bases de données spécifiques reprennent les informations communes dans la même base de données générale.

Lors de l'ajoute ou de suppression de méthodes de sélection, on ne doit pas modifier les programmes L.D.D et L.M.D existants; ceci permet d'éviter de connaître dans les détails les programmes existants, et d'introduire des erreurs dans ces programmes lors d'une nouvelle mise au point. De plus, les utilisateurs travaillant avec les méthodes de sélection dont les programmes sont déjà créés, ne sont pas directement concernés par ces modifications. D'autres avantages de la modularité peuvent être repris à ce niveau (cfr B17).

Nous retrouvons ainsi, cfr fig. 8, quatre grandes phases de traitement de l'information:

- création de la base de données générale
- création des bases de données spécifiques
- mise en forme des données (M.E.F.)
- exécution des programmes de choix.

Remarque: Après chacune des deux premières phases, les bases de données peuvent être sauvées, puis restituées, en vue de manipulations ultérieures.

4.3.2 Précision au niveau des langages de manipulation.

Nous entendons par langage de manipulation, un ensemble de commandes permettant la manipulation de différentes bases de données.

Notre solution au problème nous impose de considérer 2 types de langages de manipulation:

- le premier est destiné à la base de données générale
- le second regroupe les langages de manipulation propres aux bases de données spécifiques.

Ces langages sont interactifs. Nous les avons conçus en reprennant les éléments intervenant dans les bases de données, de manière à déterminer un ensemble type de commandes de manipulation destinées aux utilisateurs. Par la suite, nous nous sommes rendus compte des manipulations que ce dernier aurait souhaité pouvoir réaliser.

Nous avons donc veillé à ce que ces commandes satisfassent certains souhaits que nous avons jugés importants:

- du point de vue compréhension, elles sont suffisamment explicites au niveau de l'identification du libellé de la commande elle-même et de ses paramètres (éventuellement); elles sont donc aisément mémorisables pour l'utilisateur
- on en donne un jeu suffisamment large de manière à effectuer les fonctions nécessaires (cfr § 3.3); elles sont cependant limitées en nombre, afin d'éviter la complexité
- elles sont d'une grande souplesse d'écriture, de manière à en faciliter la manipulation; par exemple:
  - commande abrégée
  - plusieurs possibilités d'écrire la même commande (ex: l'ordre des paramètres n'est pas fixé a priori).

De plus, dans la rédaction de ce programme interactif, nous sommes restés attentifs:

- au temps de réponse (qui dépend évidemment du genre de traitement effectué)
- à l'envoi de messages explicites et variés à l'utilisateur, qu'il s'agisse de messages
  - de communication
  - d'erreurs
  - explicatifs du traitement réellement effectué.

Nous avons remarqué également qu'il aurait été très intéressant de prévoir des commandes de documentation pour aider l'utilisateur (*):

- soit au niveau de l'ensemble des commandes
- soit à des niveaux plus particuliers (p.ex. après une commande erronée, on lui indiquerait les possibilités prévues).

Cependant, le manque de temps ne nous a pas permis de réaliser cette dernière fonction, aussi supposons-nous que l'utilisateur ait une idée suffisamment précise de ce qui lui est permis ou pas.

(*) Exemple: cfr Siemens 4004-151: la commande *HELP lors de l'utilisation du programme BDIAG.

4.3.3 Interface base de données générale - base de données spécifique.

Un interface entre la base de données générale et chaque base de données spécifique (Cat et Electre II) a été créé de manière à charger, à partir de la base de données générale, dans ces bases de données spécifiques, la structure totale ou partielle de la hiérarchisation des critères de choix. En effet, on peut prévoir le cas où deux méthodes de sélection désirent se baser sur une hiérarchisation de critères différente et telle que l'une englobe l'autre (cfr fig. 9). Dans ce cas, on peut désirer envoyer le sous-graphe vers une base de données spécifique et l'entièreté du graphe vers une autre base de données spécifique.

Il s'avère donc intéressant de prévoir une méthode de discrimination des critères existant dans la base de données générale, de manière à ce que l'utilisateur puisse charger sa base de données spécifique à partir des critères qu'il juge utile de retenir dans la base de données générale.

D'autre part, il n'est pas dit que l'utisateur veut toujours se referer à la structure arborescente de la base de données générale pour construire sa structure arborescente dans sa base de données spécifique. Il est donc nécessaire de prévoir l'introduction des données dans cette base de données spécifique, d'une autre manière que par la base de données générale; c'est pourquoi, on doit pouvoir autoriser l'utilisateur de charger cette base de données spécifique au moyen d'un langage de manipulation approprié.

Nous remarquons donc à ce niveau que l'interface entre la base de données générale et une base de données spécifique ne peut être considéré comme manipulation d'une des deux bases de données, car elle dépend en fait de l'existence des deux bases de données.

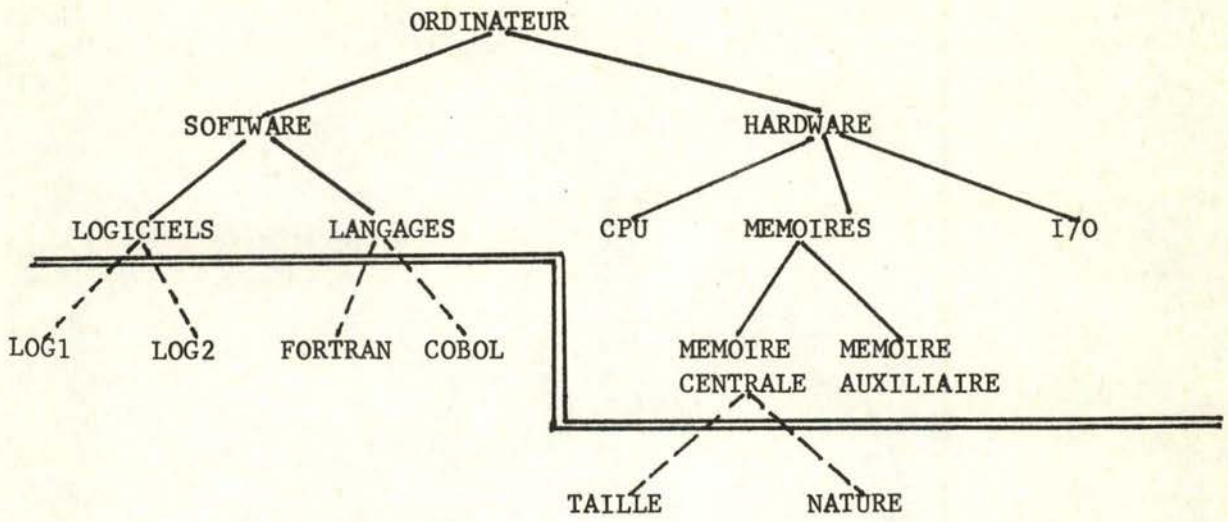


Fig 9: exemple de sous-graphe et de graphe englobant le premier.

4.3.3 Interface base de données spécifique -programme de choix.

L'interface entre la base de données spécifique et son programme de choix doit être réalisé de manière à mettre en forme toutes les informations nécessaires à l'exécution de ce programme.

Etant donné que cet interface est propre à chacune des bases de données (c.à.d. indépendante d'autres bases de données), nous pouvons le considérer comme manipulation au même titre qu'une manipulation de création...

Il existe des informations nécessaires à "l'input" de ces programmes de choix qui ne sont pas reprises au niveau de cette base de données. Il doit donc être possible d'entrer ces données, soit par le terminal (ex: seuils de concordance), soit au moyen d'un fichier auxiliaire (ex: commandes du langage Sel).

De plus, cet interface doit être réalisé de manière à n'effectuer aucune modification du programme de choix, de façon à encore autoriser l'introduction des données en mode batch (par cartes), comme c'était le cas auparavant.

CONCLUSIONS.

La première étape dans la réalisation de ce mémoire a consisté à quelques rappels sur le problème de choix, le processus de sélection ainsi que les méthodes de sélection Cat et Electre II.

Afin de résoudre les problèmes posés par l'existant, nous avons évalué différentes solutions pour enfin en retenir une. Cette solution permet l'introduction des données (création, modification...), grâce à un langage interactif, au niveau des bases de données dont les informations sont exploitables par les programmes de choix.

Actuellement le langage permet des manipulations que nous pouvons qualifier de primordiales. Il reste évidemment des manipulations possibles à créer pour faciliter la tâche des utilisateurs (cfr Conclusions de l'Annexe). La manière dont nous avons implémenté nos programmes doit permettre l'adaptation facile de tout nouveau programme désirant venir "se greffer" aux programmes existants.

La solution retenue n'est certes pas complète; en effet, nous avons créé un langage interactif facilitant l'introduction des données afin qu'elles soient exploitables par un programme de choix. Nous créons donc un état d'entrée équivalent à celui qui existait déjà avant notre application, mais de manière plus simple. Un complément à cette solution serait d'interagir avec le programme de choix: c.à.d. lui envoyer des données partielles et recevoir des résultats intermédiaires. Par exemple, pour le programme de choix Cat, il serait intéressant de connaître l'évaluation d'un (ou de plusieurs) critères, avant de connaître l'évaluation du critère global. Ceci nécessitrait bien sûr la modification des programmes de choix.

On peut trouver intéressant une connexion entre les méthodes Cat et Electre II (cfr 1.4). Nous y avons pensé tout au long de la réalisation de notre système, mais nous n'avons malheureusement pas eu le temps de l'implémenter.

Jusqu'à présent, les "outputs" des programmes de choix s'effectuent par l'imprimante. Nous concevons que si, l'introduction des données se réalise de manière interactive, l'impression des résultats devrait pouvoir se réaliser au choix de l'utilisateur: soit au terminal, soit à l'imprimante (ou les deux).

Il ne faut pas oublier, que lors de l'introduction des données, nous avons été particulièrement attentifs à la détection des erreurs syntaxiques et sémantiques. Il n'en reste pas moins vrai que certaines de ces erreurs sont détectées au niveau des programmes de choix; elles sont donc redondantes. Dans une étape ultérieure, il serait ainsi intéressant de les éliminer, si on n'envisage plus l'introduction qu'en mode interactif (avec vérification d'erreur).

Toutes les possibilités d'avenir que nous venons de citer n'ont pu être réalisées, faute de temps; cependant, dans le cas opposé, nous aurions accordé la priorité à l'application de cet outil informatique, afin de justifier son utilité (comparaison des méthodes de sélection...).

GLOSSAIRE.

Dans le cadre de ce mémoire, nous associons à certains termes, des définitions qui nous sont propres, ou qui nous sont proposées par différents auteurs.

D1: sélection: classification et/ou évaluation des offres des différents constructeurs.

D2: décision: action de retenir la configuration finale, parmi les offres sélectionnées.

D3: choix (problème de choix): englobe la sélection et la décision.

D4: programme de choix: ce sont les programmes correspondant aux méthodes de sélection.

D5: existant: système existant avant la réalisation de ce travail.

D6: décideur: personne pour laquelle et au nom de laquelle un homme d'étude travaille.

D7: demandeur: personne qui commande et juge l'étude.

D8; Homme d'étude: personne utilisant un bagage de connaissances scientifiques pour modeler le problème suivant les indications du demandeur.

D9: racine: sommet ascendant de tout sommet.

D10: feuille: sommet ne possédant aucun suivant.

D11: noeud: sommet possédant au moins un suivant et un précédent.

D12: précédent de x_j: tout sommet x_i duquel part un chemin de longueur 1 aboutissant à x_j.

D13: ascendant de x_j: tout sommet xi duquel part un chemin de longueur ≥ 1 : aboutissant à x_j.

D14: suivant de x_i: tout sommet x_j en lequel aboutit un chemin de longueur 1 partant de x_i.

D15: fonction d'utilité: relation entre les valeurs d'entrée possibles (valeurs des offres) et leurs utilités respectives.

D16: descendant de x_i: tout sommet x_j en lequel aboutit un chemin de longueur 1 partant de x_i.

D17: fonction d'efficacité: idem que fonction d'utilité (D15).

D19: actions réalisables: il s'agit des différents projets sur lesquels porte la sélection.

D20: sommet isolé: c'est un sommet n'ayant aucun précédent, ni aucun suivant.

D21: critères immédiatement subordonnés (à un critère): ce sont les suivants du critère donné.

BIBILIOGRAPHIE.

B1: J. Ph. Arnotte, Evaluation de systèmes complexes, choix d'un ordinateur par la méthode Cat (2 tomes), Mémoire de licence et maîtrise en Informatique, F.N.D.P., Namur, 1977.

B2: A. Nottebart, Analyse Multicritère, Mémoire de licence en sciences Mathématiques, F.N.D.P., Namur, 1977.

B3: F. Bodart, Méthodes d'Analyse fonctionnelle, cours, Institut d'informatique, Namur, 1978.

B4: J. Ramaekers, Efficacité des systèmes informatiques, cours, F.N.D.P., Namur, 1978.

B5: Ph. Van Bastelaer, Séminaire de Design de Configuration, cours, F.N.D.P., Namur, 1978.

B6: J. Fichefet, Théorie des graphes, cours, F.N.D.P., Namur, 1977.

B7: B. Roy, La méthode Electre II, une application au média planning, M. Ross, O.R. 1972, North-Holland publishing Company (1973).

B8: B. Roy, Vers une méthodologie générale d'aide à la décision, Revue Metra, vol XIV n° 3, 1975.

B9: Ph. Van Bastelaer, Essai de questionnaire pour un choix de configuration, FNDP, Namur, 1976.

B10: C. Deheneffe, J.L. Hainaut, H. Hennebert, B. Le Charlier, W. Paulus, Système de conception et d'exploitation d'une base de données, projet de recherche C.I.P.S. n° I.2/15, F.N.D.P., Namur, 1974.

B11: Colussi, Deheneffe, Guillebaud, Hainaut, Hennebert, Le Charlier, Paulus, Système de conception et d'exploitation de bases de données, projet de recherche C.I.P.S. n° I.2/15, F.N.D.P., Namur, 1978.

B13: B. Roy, Décision avec critères multiples: problèmes et méthodes, Revue Métra vol. XI, n° 1, 1972.

B14: B. Roy, La modélisation des préférences: un aspect crucial de l'aide à la décision, Revue Metra, vol XIII, n° 2, 1974.

B15: J-P. Laloux, Implémentation du modèle d'accès par lui-même, Mémoire de licence et maîtrise en Informatique, F.N.D.P., Namur, 1976.

B16: J. Fichefet, Critique de la méthode "MAL" de J.J. Dujmovic pour l'évaluation et la comparaison de systèmes complexes, FNDP, Namur,1978.

B17: A. Clarinval, Cours de méthodologie de l'Analyse et de la Programmation, F.N.D.P, Namur, 1977.

B18: P. Berthier, Analyse des données multidimensionnelles, Paris: P.U.F., 1975.

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX

NAMUR

INSTITUT D'INFORMATIQUE

Année académique 1977-1978

OUTIL INTERACTIF D'AIDE A L'EVALUATION ET LA CLASSIFICATION DE SYSTEMES COMPLEXES

- JOSEPH fortemps

- GUY rome

- MÉMOIRE PRESENTÉ EN VUE D'OBTENIR LE GRADE DE LICENCIE ET MAÎTRE EN INFORMATIQUE

tome 2 : annexe

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX

NAMUR

INSTITUT D'INFORMATIQUE

Année académique 1977-1978

OUTIL INTERACTIF D'AIDE A L'EVALUATION ET LA CLASSIFICATION DE SYSTEMES COMPLEXES

- JOSEPH fortemps

- GUY rome

- MÉMOIRE PRESENTÉ EN VUE D'OBTENIR LE GRADE DE LICENCIE ET MAÎTRE EN INFORMATIQUE

tome 2 : annexe

TABLES DES MATIERES.

TABLE DES FIGURES.

Page

Dossier de développement.

Dossier de programmation.

Dossier utilisateur.

INTRODUCTION.

Afin d'aider l'utilisateur dans sa tâche d'évaluation et/ou de comparaison de systèmes complexes, nous lui proposons une introduction interactive de données au niveau de bases de données dont les informations sont exploitables (moyennant un interface) par des programmes de choix.

Nous avons précisé dans le dossier de conception (cfr tome I) un planning de réalisation de cette proposition. Nous allons dans ce volume développer de façon plus précise cette proposition au niveau des dossiers de développement, de programmation et utilisateur.

Le dossier de développement contient

- l'analyse des unités d'information
- l'analyse logique des traitements.

Le dossier de programmation contient

- la justification du choix du système de gestion de bases de données (SGBD) Sphinx
- la méthode de programmation choisie
- les techniques de programmation employées
- les listings des procédures usuelles.

Le dossier utilisateur comprend les données relatives à l'exploitation: c.à.d.:

- les commandes de contrôle nécessaires à l'initialisation et à la terminaison des programmes d'application
- la description du langage de manipulation (général,Cat et Electre)
- un exemple de messages envoyés à l'utilisateur.

PREMIERE PARTIE: DOSSIER DE DEVELOPPEMENT.

CHAPITRE I ANALYSE DES UNITES D'INFORMATION.

1.1 Préliminaires.

L'ordre dans lequel nous décrivons les unités d'information est défini par le schéma d'enchaînement des flux des informations (cfr fig 1). A travers ces flux, nous distinguons quatre phases de traitement de l'information qui sont:

1. la création de la base de données générale
2. la création des bases de données spécifiques
3. la mise en forme des données en vue de les utiliser dans les programmes de choix
4. et finalement, l'exécution de ces programmes de choix (spécifiques à chaque méthode de sélection).

D'autre part, afin de mieux présenter la structure des informations, nous nous appuyons sur un modèle décrit par F. Bodart (B3) et dont nous rappelons brièvement les concepts: entités, association, propriété-valeur, cardinalité.

Entité:

> "une entité est ce qu'un individu ou un groupe voit comme un tout, ayant une existence propre; une entité est caractérisée par un ensemble de propriétés quantitatives et qualitatives et un comportement permanent; le nombre de propriétés et la permanence sont fonction du point de vue choisi".

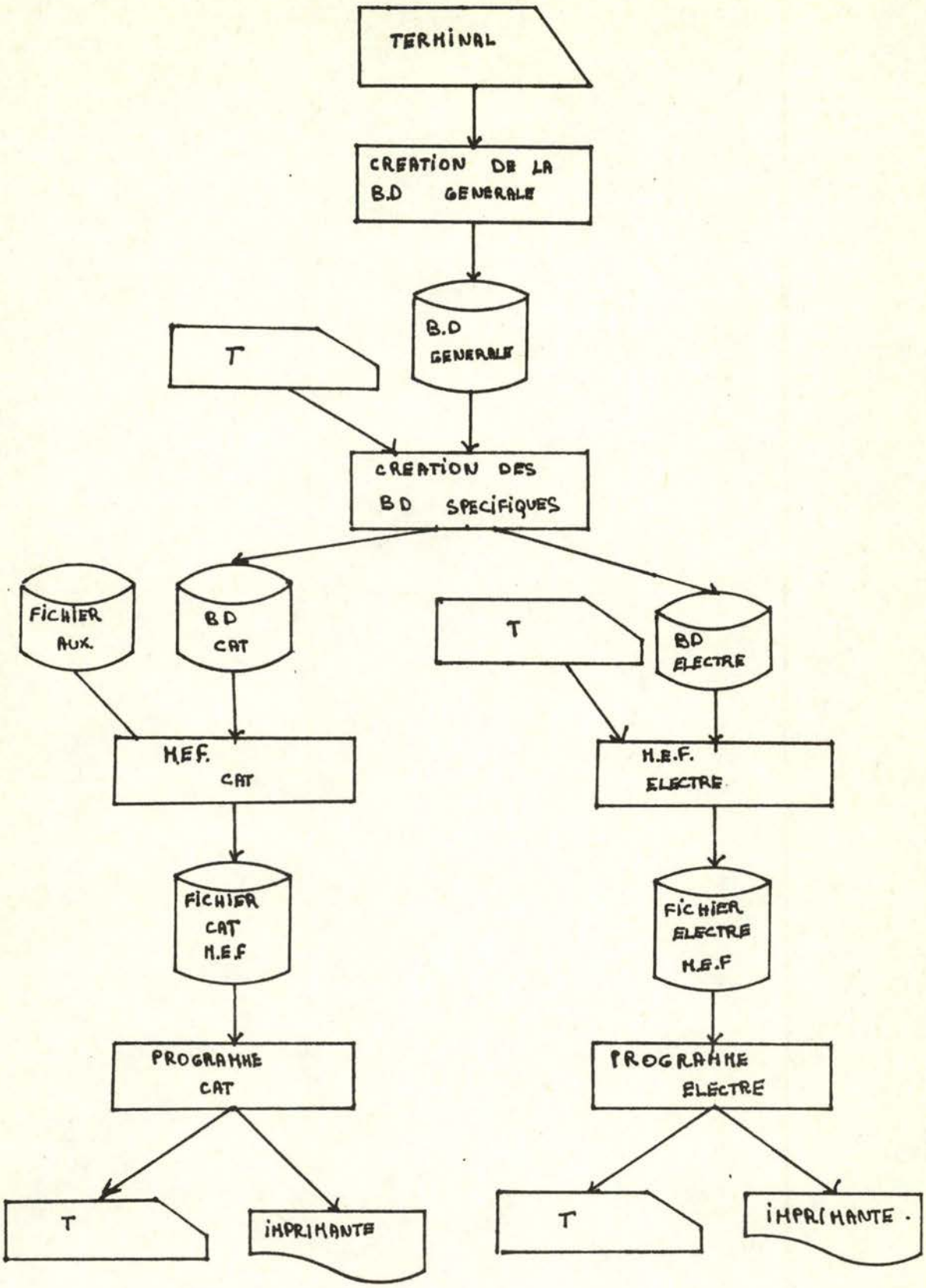


Fig. 1: enchaînement des phases au travers du flux des informations.

Association:

"une association est un ensemble de deux ou plusieurs entités où chacune assume un rôle donné. Une association peut posséder plusieurs propriétés. L'existence d'une association est contingente à l'existence des entités qu'elle relate".

Propriété-valeur:

"une propriété appartenant à une entité ou une association est une qualité que les individus attribuent à cette entité ou cette association. L'existence des propriétés attribuées est contingente à l'existence des entités ou de l'association concernée".

Cardinalité des liaisons binaires entre unités d'information.

"La cardinalité d'une liaison procure une information sur le nombre d'occurrences de l'entité terminal de la relation pour une réalisation de l'entité origine et réciproquement.
Une relation binaire sera dite one-to-one (1-1) si à une occurrence de l'entité origine de la relation correspond une seule occurrence de l'entité cible et réciproquement.
Une relation binaire sera dite one-to-many (1-n) si à une occurrence de l'entité origine de la relation peut correspondre plusieurs occurrences de l'entité cible; par contre, à une occurrence de l'entité cible correspond zéro ou une occurrence de l'entité origine.
Une relation binaire sera dite many-to-many (n-n) si à une occurrence de l'entité origine de la relation peut correspondre plusieurs occurrences de l'entité cible et réciproquement."

1.2 Description des unités d'information relatives à la phase de création de la base de données générale.

L'analyse des informations relatives à la phase(*1) de création de la base de données générale consiste à analyser et à décrire les commandes du langage de manipulation (*) ainsi que les unités d'information définies dans cette base de données. Le langage de manipulation est décrit et expliqué dans le dossier utilisateur; c'est pourquoi nous nous contentons dans ce paragraphe d'analyser les unités d'information de la base de données générale.

(*) Le langage de manipulation est constitué d'un ensemble de commandes qui permettent de définir des manipulations de données (introduction, modification, affichage des données ...). Ce langage de manipulation comprend:

- le langage de manipulation général
- le langage de manipulation Cat
- le langage de manipulation Electre.

Ces trois derniers langages de manipulation sont composés de commandes se rapportant respectivement aux bases de données générale, Cat et Electre II.

(*1) Nous entendons par unités d'information relatives à une phase, les "input" et les "output" de cette phase.

1.2.1 Remarque.

Avant de definir les unités d'information et leur structure logique, il nous a semblé utile de rappeler que la base de données générale contient une structure arborescente dont les sommets sont identifiés par les noms de critères et les relations entre ces sommets.

. On n'identifie pas ici la racine, les noeuds et les feuilles de l'arborescence par les informations associées aux sommets (c.à.d. l'identifiant), mais par le biais des relations:

- une racine n'a pas de précédent
- un noeud a au moins un précédent et un suivant
- les feuilles n'ont pas de suivants.

1.2.2 Description et quantification des unités d'information.

a. Description.

Les unités d'information associées à la base de données générale sont:

- NOM (identifiant): ce type d'information permet l'identification d'un sommet de l'arborescence ainsi que du critère associé à ce sommet
- VERSION: étant donné notre idée de "modularité" où les bases de données Cat, Electre et Autres puisent des informations dans la base de données générale, il nous semble indispensable de spécifier les critères à prendre en considération dans cette dernière (cfr dossier de conception § 3.3). Grâce à ce n° de version, chaque utilisateur peut caractériser les critères de la base de données générale qu'il souhaite créer dans sa base de données spécifique (cfr fig 2)

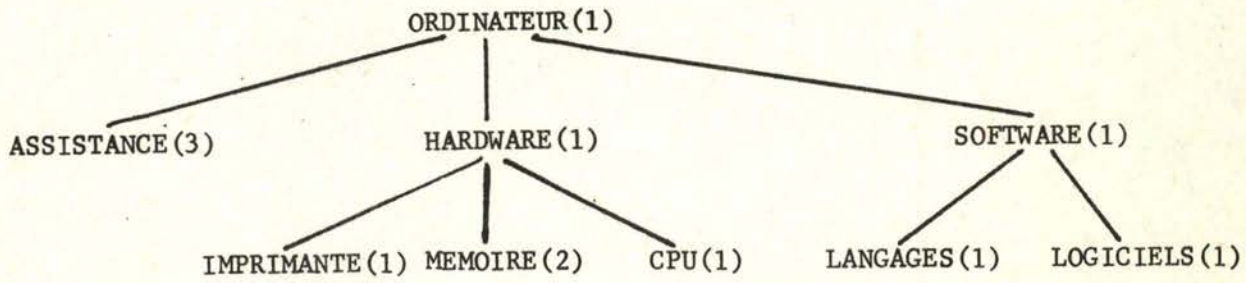


Fig 2: exemple d'arborescence où chacun des critères est caractérisé par un n° de version.

Reprenons l'exemple suivant pour bien comprendre la signification de cette information. Au départ, nous avons une structure arborescente (cfr fig. 2) chargée dans la base de données générale. Nous supposons ensuite que nous voulons charger dans la base de données correspondant aux deux méthodes de sélection:

Electre: l'entièreté du graphe, sauf le critère "MEMOIRE"

Cat: l'entièreté du graphe, sauf le critère "ASSISTANCE".

Il suffit dès lors de caractériser les critères communs à prendre dans les deux bases de données spécifiques par le même n° de version (p. ex.:1); nous associons ensuite aux critères

- "MEMOIRE" un n° de version différent du premier: (p.ex.2)
- "ASSISTANCE" un n° de version également différent: (p.ex.3)

Lors du passage de la base de données générale à la base de données spécifique, il suffit de préciser dans l'interface entre la première base citée et la base de données

- Cat, qu'on désire reprendre les critères correspondants aux n° de version 1 et 2
- Electre, qu'on reprend uniquement les critères de n° de version 1 et 3.

L'information version pourrait posséder d'autres significations; en effet, un n° de version serait en mesure de caractériser une ou plusieurs versions (ex: une date de création) de l'arborescence. Cette information permettrait entre autre de connaître l'évolution de création d'une arborescence. Dans l'exemple de la figure 2, le n° de version pourrait prendre les significations suivantes:

- n° 1: création le 10/03/78
- n° 2: création le 12/03/78
- n° 3: création le 15/03/78.

D'autres significations de cette information sont laissées à l'imagination de l'utilisateur.

b. Règles de vérification et quantification.

NOM: longueur ≤ 40 caractères alphanumériques.

La limite maximum spécifiée ici est une contrainte définie par l'existant; en effet, la longueur de l'identifiant d'un critère ne peut dépasser 40 pour le programme de choix Cat.

VERSION: 1 ≤ n° de version ≤ 9

cette dimension du n° de version est fixée arbitrairement.

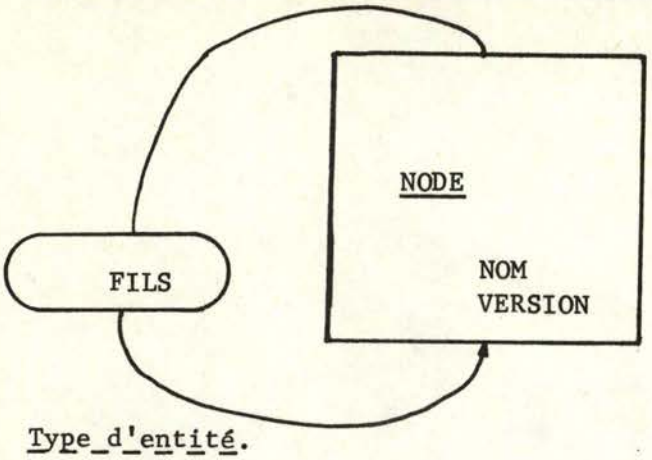
c. Remarque.

Nous imposons à ce niveau, pour garder toute généralité, qu'un sommet possède un et un seul précédent (sauf la racine) et qu'il peut avoir un nombre de suivants compris entre 0 et l'infini théorique.
Par conséquent, toutes les relations entre

- noeud (éventuellement racine) et feuille
- racine et noeud
- noeud et noeud

se caractérisent par un même type d'association d'une entité NODE (appelée précédent) à d'autres entités NODE (appelées suivants de la première entité);

1.2.3 Structure logique des informations.



Type_d'entité.

NODE

propriétés:

NOM

VERSION

Type_d'association.

FILS

Cardinalité.

FILS(NODE,NODE): 1-n

1.3 Description des unités d'information relatives à la phase de création des bases de données spécifiques.

1.3.1 Remarque concernant les bases de données spécifiques.

Les deux méthodes de sélection (Cat et Electre) réclament des unités d'information communes:

- les libellés des critères
- les poids associés aux critères.

De plus, elles sont caractérisées par des paramètres qui leur sont propres; ainsi retrouve-t-on dans la méthode:

- Cat: les coéfficients d'aggrégation
  les fonctions d'utilité
- Electre II: les seuils de concordance et de discordance
  les évaluations des projets
  une condition sine qua non, un facteur d'échelle
  (particuliers au programme Electre rédigé à l'Institut).

Nous avons créé une base de données générale contenant uniquement la structure arborescente des critères de choix. Des informations communes aux deux méthodes de sélection, nous n'avons retenu dans la base de données générale que les libellés des critères. En effet, le poids associé à ces critères n'y est pas repris pour la simple raison que d'autres méthodes de sélection ne possèdent pas nécessairement cette information. La prise en compte de ce paramètre enlèverait le caractère général que l'on veut donner à la base de données générale.

Nous avons ensuite décidé de créer des bases de données spécifiques contenant:

- la structure arborescente des critères (soit à partir de la base de données générale, soit directement à partir d'un langage de manipulation approprié)

- les paramètres propres aux méthodes de sélection correspondantes.

Nous utilisons trois types de définitions pour caractériser un critère ou un groupe de critères de l'arborescence:

1° type: racine
noeud
feuilles

2° type: critère global
critères non élémentaires
critères élémentaires

3° type: sommets non pendants
sommets pendants, (*)

La signification de
-racine correspond à celle de critère global
-noeud " " " " critère non élémentaire
-feuille " " " " critère élémentaire.

Nous n'associons pas *ici*
-critère élémentaire à sommet pendant
-critère non élémentaire et/ou racine à sommet non pendant.

En effet, lorsque nous créons par exemple une série de sommets pendants, nous ne créons pas nécessairement que des feuilles (Critère élémentaire); il est donc important de noter qu'à ce niveau, les notions de noeuds et de feuilles ne correspondent plus aux définitions données dans le dossier de conception.

(*) Un sommet est pendant s'il n'est adjacent qu'à un seul sommet.
Deux sommets sont adjacents s'ils sont distincts et s'il existe un arc entre ces deux sommets.

1.3.2 Description des unités d'information relatives à la phase de création de la base de données Cat.

La base de données Cat est créée, soit directement à partir d'un langage de manipulation, soit indirectement à partir de la base de données générale (au moyen d'un interface approprié). Le langage de manipulation "agissant" sur cette base de données Cat est défini dans le dossier Utilisateur, tandis que la description des unités d'informations relatives à la base de données générale, fait l'objet d'un paragraphe précédent (1.1). Il nous reste donc à ce stade à considérer les unités d'information relatives à la base de données Cat.

Comme nous venons de le dire au paragraphe précédent (2.2), nous devons ajouter de nouveaux paramètres pour décrire le schéma de cette base de données:

- le coéfficient d'aggrégation, qui intervient au niveau de chaque critère non élémentaire et du critère global
- les fonctions d'évaluation intervenant uniquement pour les critères élémentaires
- les poids qui sont propres aux critères non élémentaires et élémentaires.

1.3.2.1 Description et quantification des unités d'information.

A. Description.

Les types d'information associés à la base de données spécifique Cat sont:

- NOM: même signification que pour la base de données générale
- AGGREGAT: cet opérateur correspond à un coéfficient d'aggrégation; ce dernier intervient dans la formule d'aggrégation afin d'évaluer les critères non élémentaires (noeuds et racine)
- FONCTION: c'est une fonction qui permet de calculer, à partir des valeurs d'entrée (réponse des fournisseurs), l'efficacité d'un critère élémentaire
- POIDS: c'est un paramètre qui donne l'importance d'un critère par rapport aux autres critères de même précédent. Le poids intervient également dans la formule d'aggrégation, pour évaluer les critères non élémentaires.
- PROJET: cette unité se compose de deux éléments:
  - IDENT: libellé du projet
  - VALEUR: valeur d'entrée à la fonction d'utilité

B/ Règles de vérification et quantification des unités d'information.

- NOM: idem que pour la base de données générale
- AGGREGAT: l'opérateur d'aggrégation introduit au niveau de la base de données spécifique doit correspondre à un des opérateurs d'aggrégation définis par la méthode de sélection Cat; c.à.d. qu'il doit être un des opérateurs suivants:

  C, C++, C+, C+-, CA, C-+, C-, C--, A, D--, D-, DA, D+-, D+, D++, D,

- FONCTION: la description de la fonction d'efficacité doit suivre les règles suivantes:

  - pour les valeurs d'abscisse: $0 \leq$ abscisse ≤ 9999
  - pour les valeurs de l'ordonnée: $0 \leq$ ordonnée ≤ 99

La fonction est discrète et doit posséder au minimum 2 valeurs, et au maximum 5 valeurs.

- POIDS: la valeur du poids: $0 \leq$ poids ≤ 99
- PROJET: l'identification du projet:

  $0 <$ IDENT $\leq$ 10 caractères alphanumériques

  la valeur d'entrée:

  $0 \leq$ VALEUR ≤ 99999

Le nombre de projets varie entre 1 et 20.

C. Remarque.

Depuis le début du mémoire, on s'est basé sur une structure arborescente pour décrire les critères ainsi que les relations qui existaient entre ces critères. Or, la méthode de sélection Cat a besoin de certaines relations qui violent la structure de départ; en effet, la méthode admet qu'un sommet puisse avoir plusieurs précédents. On parle dans ce cas, non plus de structure arborescente, mais bien de structure hiérarchique. Par abus de langage, nous employons par la suite uniquement le terme "arborescence" aussi bien pour parler de structure arborescente que de structure hiérarchique.

Nous montrons (cfr fig. 3) un exemple d'une structure hiérarchique propre à la méthode de sélection Cat.

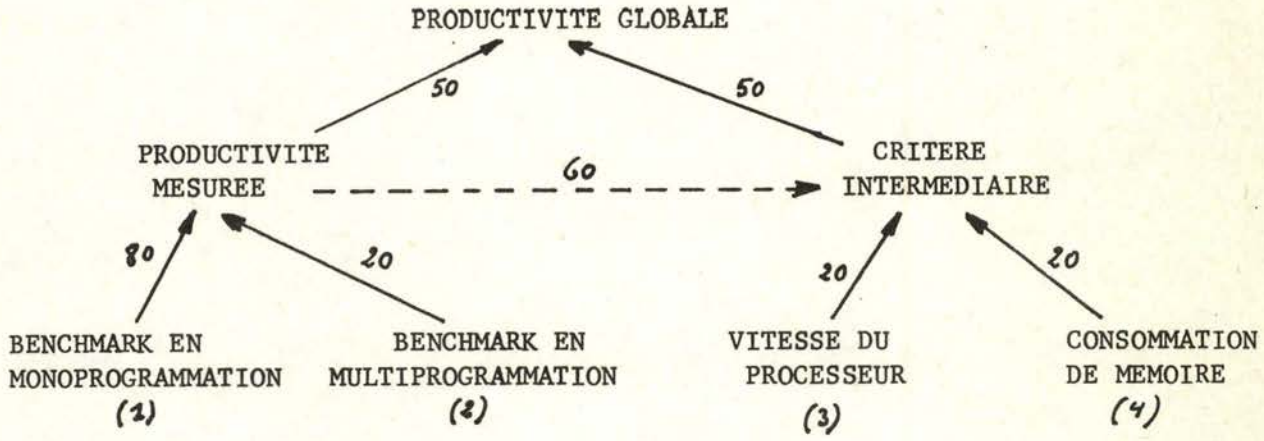


Fig. 3: exemple de structure hiérarchique spécifique à la méthode de sélection Cat. Les flèches indiquent le sens de l'évaluation (cfr formule d'aggrégation) des critères afin d'évaluer le critère global.

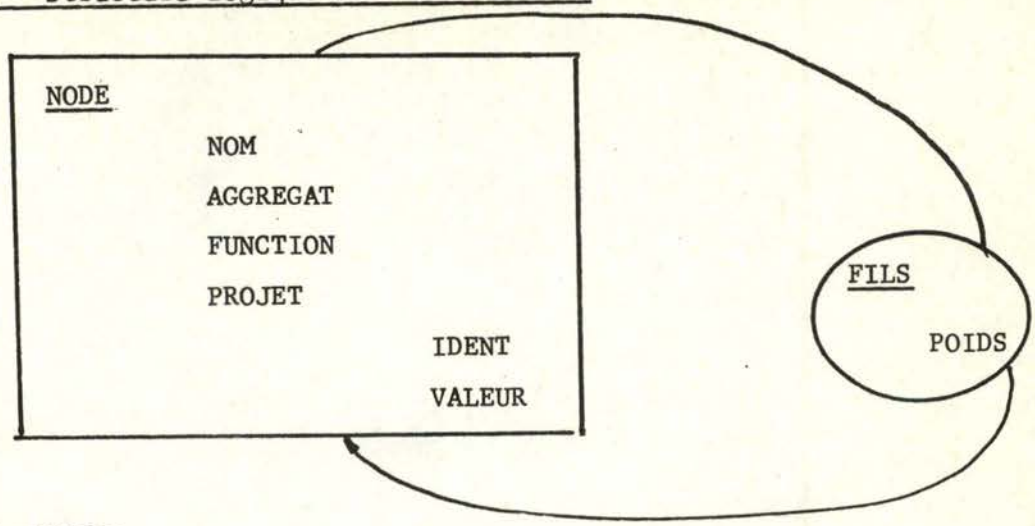
Pour évaluer la PRODUCTIVITE GLOBALE, on évalue d'abord la PRODUCTIVITE MESUREE en aggrégeant les critères élémentaires (1) et (2): cfr formule d'aggrégation. On estimera ensuite la PRODUCTIVITE GLOBALE par rapport à la PRODUCTIVITE MESUREE, mais tempérée par les critères élémentaires (3) et (4).

D'où l'introduction d'un CRITERE INTERMEDIAIRE et d'une relation entre PRODUCTIVITE MESUREE et CRITERE INTERMEDIAIRE. Dans ce cas précis, 2 poids sont associés à PRODUCTIVITE MESUREE:

- lors de l'évaluation de CRITERE INTERMEDIAIRE, elle est considérée comme un critère immédiatement subordonné au même titre que les critères élémentaires (3) et (4), et on lui associe le poids 60.
- lors de l'évaluation de la PRODUCTIVITE GLOBALE, elle est considérée comme un critère immédiatement subordonné au même titre que le CRITERE INTERMEDIAIRE, et on lui associe le poids 50.

Cet exemple représente en réalité le concept de l'absorption partielle (cfr (B1)). Par conséquent, le poids est un paramètre qui ne dépend plus d'un seul sommet, mais bien de deux sommets; il caractérise donc une relation d'évaluation.

1.3.2.2 Structure logique des informations.



Type_d'entités:

NODE

propriétés:

NOM

AGGREGAT

FUNCTION

PROJET

IDENT

VALEUR

Type_d'association:

FILS

propriété:

POIDS

Cardinalité:

FILS(NODE,NODE):n-n

1.3.3 Description des unités d'information relatives à la phase de création de la base de données Electre.

La création de la base de données Electre s'effectue soit indirectement à partir de la base de données générale (au moyen d'un interface approprié), soit directement, à partir d'un langage de manipulation. Le langage de manipulation agissant sur cette base de données Electre est défini dans le dossier Utilisateur, tandis que la description des unités d'information relatives à la base de données générale, fait l'objet d'un paragraphe précédent (1.1). Il nous reste donc à ce stade à considérer les unités d'information relatives à la base de données Electre.

Le schéma de la base de données Electre est très semblable au schéma de la base de données générale; nous devons simplement tenir compte de certains paramètres:

- le libellé des critères (cfr base de données générale)
- le poids attribué à chaque critère (sauf au critère global)
- les évaluations de tous les projets, pour chaque critère
- deux seuils de discordance (SD1 et SD2)
- trois seuils de concordance (SC1, SC2 et SC3)
- une condition sine qua non (SQN) imposant une restriction supplémentaire aux seuils de discordance et de concordance
- un facteur d'échelle indiquant le sens de la variation des évaluations des projets.

Il est à noter que ces deux derniers paramètres ne sont pas indispensables au bon déroulement de la méthode; ils ont été ajouté dans les programmes existants, pour bénéficier d'un peu plus de souplesse.

D'autre part, l'arborescence est caractérisée par une racine, des noeuds et des feuilles. Nous savons que la racine de ce graphe de choix sera caractérisée uniquement par son libellé. Par contre, les noeuds peuvent se voir accorder un poids: ceci s'explique surtout, lors de l'élaboration de la structure arborescente, pour mieux situer l'importance relative d'un critère par rapport à un autre.

Enfin, nous considérons que la méthode Electre II effectue ses comparaisons sur l'ensemble des critères correspondant aux feuilles du graphe. Ceci constitue une manière simple de sélectionner les critères devant intervenir finalement pour la sélection. Ces feuilles sont caractérisées par l'ensemble des données que nous avons spécifié plus haut.

1.3.3.1 Description et quantification des unités d'information.

a. Description.

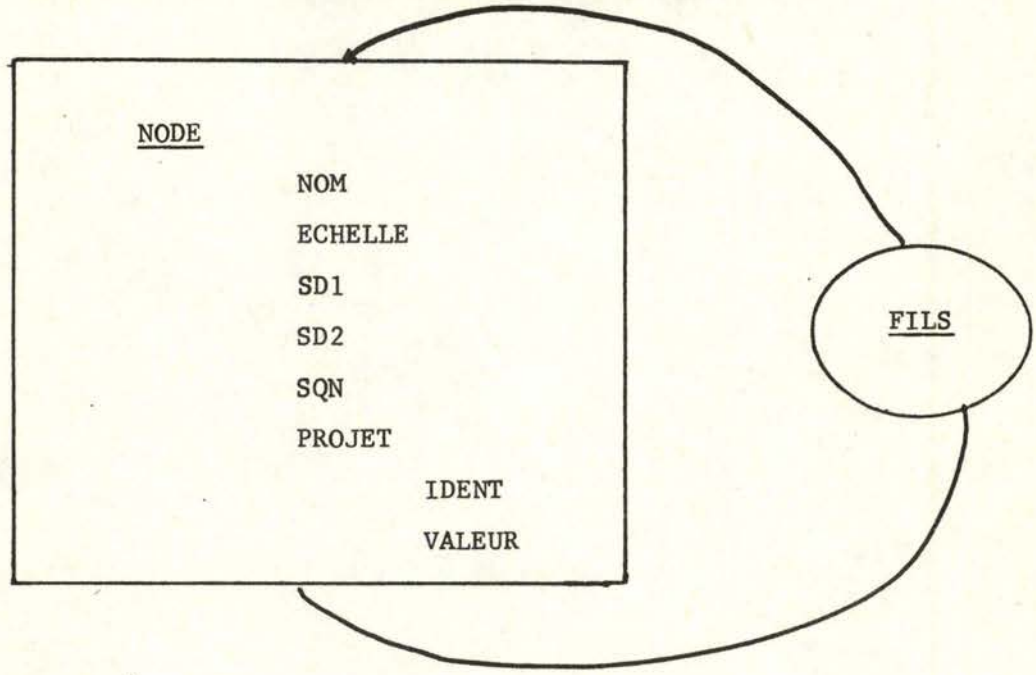
- NOM: (*)
- POIDS: (*)
- ECHELLE: permet de déterminer le sens de variation des évaluations des projets
- SD1 et SD2: sont les seuils de discordance qui contribuent à l'élaboration de la relation de surclassement
- SQN: condition sine qua non, qui implique une restriction supplémentaire possible: un projet dont l'évaluation (VALEUR) $\leq$ SQN ne pourra jamais surclasser un autre projet
- PROJET: (*)

(*) cfr Description des unités d'information:relatives à la base de données Cat.

b._ Règles_de vérification_et quantification_des_unités d'information.

- NOM : (*)
- SD1: la valeur doit être comprise entre 0 et 99999
- SD2: idem
- SQN: idem
- POIDS: idem
- ECHELLE: la valeur de l'échelle est normalement fixée à 1, mais elle peut être comprise entre 0 et 99.
- PROJET: (*)

1.3.3.2 Structure logique des unités d'information.



(*) cfr Description des unités d'information relatives à la base de données Cat (1.3.1); il faut cependant tenir compte que les valeurs d'entrée des fonctions d'utilité (de la méthode Cat) sont remplacées par les évaluations des projets (pour la méthode Electre).

Type_d'entité:

NODE

propriétés:

NOM

SD1

SD2

SQN

POIDS

ECHELLE

PROJET

IDENT

VALEUR

CArdinalité.

FILS (NODE,NODE): 1-n

1.4 Description des unités d'information relatives à la phase de mise en forme des données.

1.4.1 Description des unités d'information relatives à la phase de mise en forme des données pour le programme Cat.

La phase de mise en forme des données pour le programme Cat consiste à organiser les informations provenant

- de la base de données Cat
- et d'un fichier auxiliaire, de manière à exécuter ce programme (cfr fig. 1).

Au niveau de "l'input" de cette phase de mise en forme, il nous reste ici à développer la description des données relatives à ce fichier auxiliaire. Il contient les commandes et les instructions du langage SEL (cfr B1) qui, à partir de données provenant de la base de données spécifique Cat, permet de réaliser une ou plusieurs fonctions de la méthode Cat.

Dans notre cas, nous avons créé un jeu de commandes et d'instructions permettant de traiter ces données et d'en sortir les résultats qui nous semblent primordiaux. Il suffirait de définir un autre jeu d'instructions afin de réaliser des fonctions générant d'autres résultats.

"L'output" de la phase de mise en forme consiste en un fichier "mis en forme" (M.E.F.) contenant les données organisées de manière à ce qu'elles soient exploitables par le programme de choix Cat. L'organisation de ces données est décrite dans le dossier de conception (cfr chap. II).

1.4.2 Description des unités d'information relatives à la phase de mise en forme des données pour le programme Electre.

La phase de mise en forme des données pour le programme Electre consiste à organiser d'une part les informations provenant de la base de données Electre II, et d'autre part, des informations en provenance du terminal, cfr fig. 1.

Au niveau de "l'input", il nous reste donc ici à décrire les informations venant du terminal; il s'agit des seuils de concordance qui sont globaux à l'ensemble des critères et qui, par conséquent, ne doivent pas nécessairement être mémorisés dans la base de données Electre.

Ces seuils de concordance sont des nombres décimaux; ils doivent donc vérifier la condition suivante:

$$0 < \text{seuil de concordance} < 1$$

"L'output" de cette phase de mise en forme consiste en un "fichier mis en forme " (M.E.F.) contenant les données organisées de manière à ce qu'elles soient exploitables par le programme de choix Electre. L'organisation de ces données est décrite dans le dossier de conception (cfr chapitre II).

1.5 Description des unités d'information relatives à la phase d'exécution des programmes de choix.

La phase de création des programmes de choix consiste à évaluer et/ou classifier des objets complexes à partir des données décrites dans les fichiers "mis en forme" et à imprimer les résultats demandés. Les fichiers "mis en forme" sont décrits dans le dossier de conception (cfr chapitre II) tandis que les résultats sont expliqués dans les listings et en B1 (pour Cat).

CHAPITRE II ANALYSE DES TRAITEMENTS, PHASE PAR PHASE.

2.1 Remarque.

Nous rappelons que nous avons défini quatre phases de traitement de l'information (cfr fig. 1):

1. la création de la base de données générale
2. la création des bases de données spécifiques
3. la mise en forme des données en vue de les utiliser dans les programmes de choix
4. et finalement, l'exécution de ces programmes de choix (spécifiques à chaque méthode de sélection.

Chaque phase peut être décomposée en un certain nombre de types de fonctions ; on parle ainsi (cfr B3 et B17) de:

- fonctions génératives, qui portent essentiellement sur des transformations de fichiers
- fonctions constitutives: ce sont des fonctions d'enregistrement, de contrôle, d'édition...
- fonctions de manipulation de fichier: qui travaillent au niveau d'un groupe d'articles: on parle ainsi de fonctions de reproduction, d'éclatement, de tri...

2.2 Analyse des traitements de la phase 1.

2.2.1 Création de la base de données générale par l'intermédiaire du langage de manipulation.

La base de données générale contient une structure arborescente dont les caractéristiques utiles à l'utilisateur seront les n° de version et les libellés des critères. En général, un utilisateur désirant appliquer une méthode de choix, aura une idée des critères qui entreront en jeu. A cette fin, il utilisera un croquis représentant l'arbre qu'il voudrait mémoriser dans la base de données (cfr fig. 4). L'élaboration de la structure arborescente se fera souvent au prix de quelques modifications, ce qui souligne encore une fois l'importance d'un programme interactif, à ce niveau.

Nous tenons à rappeler ici, qu'une structure arborescente de ce type autorise une hiérarchisation en niveaux: nous dirons donc que, dans le cas de la figure 4:

- le niveau 00 sera caractérisé par ORDINATEUR
- le niveau 01 sera caractérisé par HARDWARE SOFTWARE RELATIONS-VENDEURS
- ...

2.2.1.1 Création.

Initialement, la base de données est vide. La première fonction à réaliser avant de l'utiliser, sera le chargement d'informations, dans celle-ci. Cette phase de création de la base de données ne peut s'exécuter au hasard: il s'avère indispensable que le futur contenu de la base de données reflète exactement les volontés de l'utilisateur.

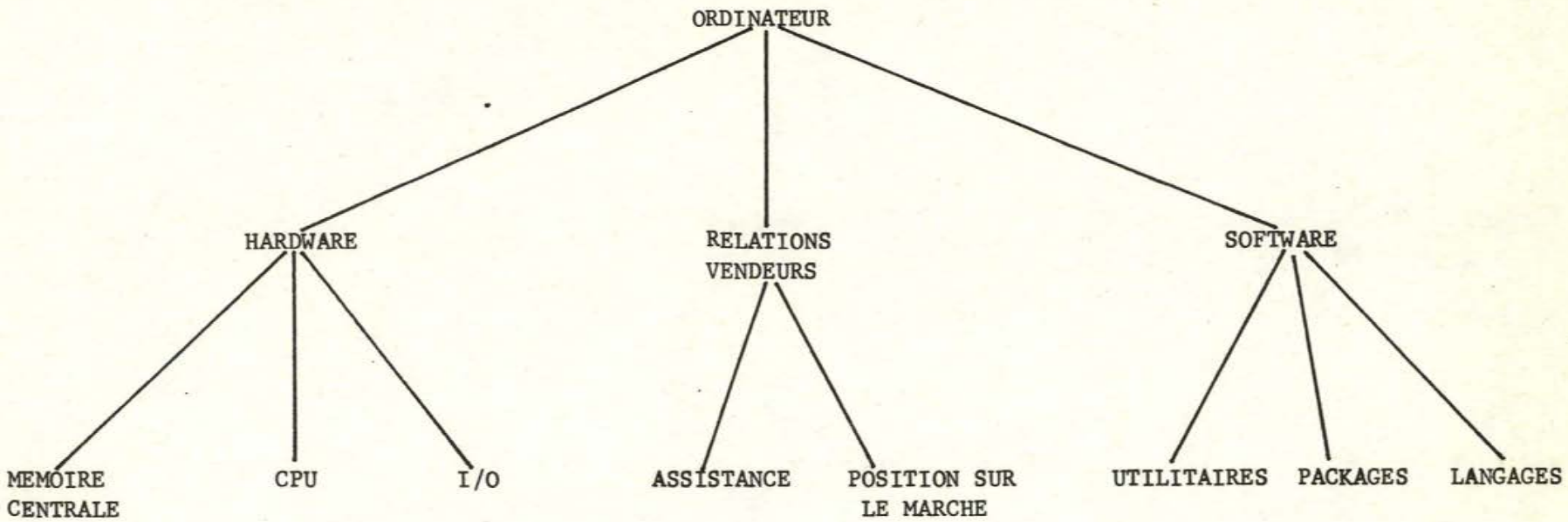


Fig. 4: exemple de critères agencés sous forme arborescente intervenant pour le choix d'ordinateur.

Si nous reprenons l'exemple de la figure 4, cela signifie que:

ORDINATEUR représente la racine du graphe de choix
MEMOIRE CENTRALE, CPU, I/O sont les suivants de HARDWARE...

Afin de réaliser la fonction de création, nous distinguons deux types de sommets:

- ceux qui n'ont pas de précédent
- ceux qui ont un précédent.

En premier lieu, l'utilisateur crée donc la racine et l'identifie par un libellé de son choix: ORDINATEUR, VOITURE ...

Lors de la création d'un autre sommet, l'utilisateur lui associe un libellé, mais en plus, il spécifie le sommet auquel il veut le rattacher. Nous imposons donc:

- que tout sommet (sauf la racine) soit caractérisé par deux identificateurs: un pour lui-même et l'autre pour son précédent dans le graphe de choix
- qu'un sommet ne puisse être créé si le sommet auquel nous voulons le rattacher n'existe pas dans la base de données.

Enfin, nous avons constaté que ce type de création, critère par critère, d'un graphe composé d'un grand nombre de sommets, devient une manipulation assez lourde. Nous avons par conséquent jugé intéressant la possibilité de créer toute une branche en une fois. A ce sujet, nous renvoyons le lecteur au dossier Utilisateur.

2.2.1.2 _ Impression.

L'impression reste le seul moyen de vérifier l'exactitude des informations créées; elle s'avère donc indispensable. Afin de réaliser cette impression, nous avons décidé d'utiliser la structure de représentation d'un article Cobol, bien connue des informaticiens.

De plus, nous avons considéré deux critères de sélection de l'impression:

- le nombre de niveaux maximum souhaité
- l'identification d'un sommet, à partir duquel on souhaite effectuer l'impression (de ce sommet et de ses descendants).

De même, la combinaison de ces deux critères de sélection de l'impression est également intéressante.

Ex: si nous voulons imprimer le graphe de la figure 4, nous obtenons:

```
00 ORDINATEUR
   01 HARDWARE
        02 MEMOIRE CENTRALE
        02 CPU
        02 I/O
   01 RELATIONS VENDEURS
        02 ASSISTANCE
        ...
```

2.2.1.3 _ Modification.

La consultation du contenu de la base de données rendue possible, l'utilisateur souhaite dans certains cas le modifier. Nous considérons ici le terme "modifier" dans toute sa généralité, c.à.d.:

- ajouter un ou plusieurs critères
- supprimer un ou plusieurs critères
- changer le libellé ou un paramètre d'un critère existant.

a. addition d'un élément:

L'addition d'un nouveau sommet dans la base de données doit pouvoir s'effectuer suivant deux points de vue différents:

- le premier correspond à la création d'un sommet pendant cfr 2.2.1.2
- le second correspond à la création d'un sommet non pendant; il faut dans ce cas spécifier son précédent et son (ou ses) suivant(s).

Nous nous expliquons par un exemple: nous voulons insérer à la figure 4: MEMOIRE et MEMOIRE AUXILIAIRE, de manière à obtenir le graphe de la figure 5.

Nous concevons aisément que la suite logique des opérations à effectuer sera:

- d'abord une déconnexion de MEMOIRE CENTRALE par rapport à HARDWARE
- ensuite la création de MEMOIRE et sa connexion à HARDWARE
- le rattachement de MEMOIRE CENTRALE à MEMOIRE
- et enfin, la création de MEMOIRE AUXILIAIRE et sa connexion à MEMOIRE.

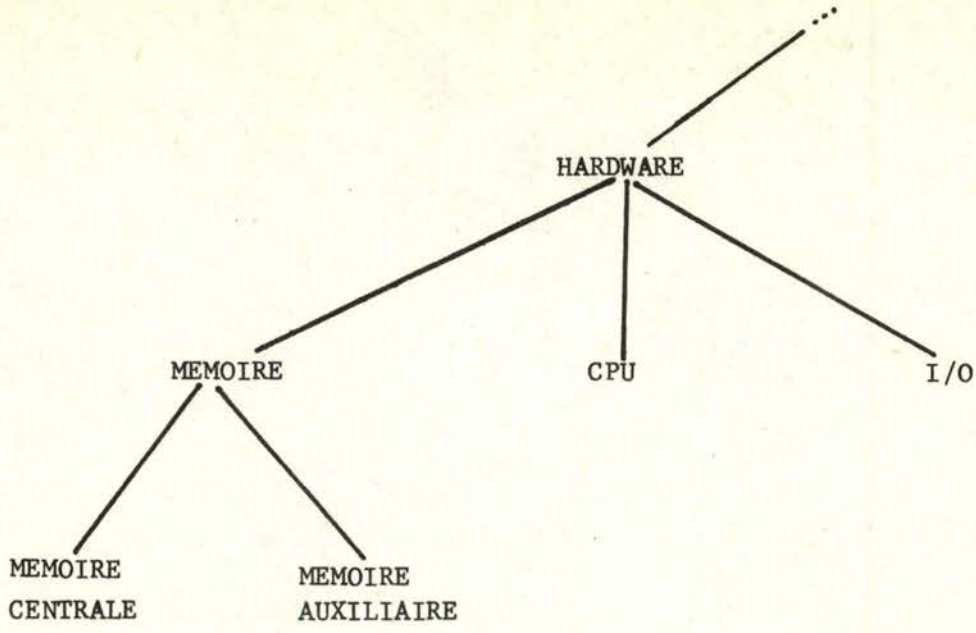


Fig. 5: Exemple d'addition de critères.

Les trois premières étapes correspondent à l'insertion d'un sommet non pendant, tandis que la quatrième étape réalise la création d'un sommet pendant.

Nous n'avons malheureusement pas eu le temps d'implémenter la fonction d'insertion, aussi avons-nous décidé d'utiliser une autre solution, malgré sa lourdeur. Cette solution consiste à supprimer tout ce qui est rattaché à HARDWARE par la branche MEMOIRE CENTRALE et à recréer par la suite ce que l'on désire.

b. Suppression d'un critère.

Conceptuellement, la suppression reflète la fonction inverse de l'addition (*): aussi contentons-nous de la survoler; nous considérons donc également deux stades différents:

- la suppression d'un sommet pendant ou d'une branche
- la suppression d'un sommet non pendant (fonction inverse de INSERTION = DELETE)

Dans le premier cas, nous imposons que le fait de supprimer un sommet du graphe existant implique la suppression de tous ses descendants (s'ils existent). Par conséquent, la suppression de la racine, en particulier provoque la suppression de l'entièreté de l'arborescence.

Reprenons la figure 5 pour illustrer le deuxième cas: si nous souhaitons supprimer le sommet MEMOIRE, sans éliminer ses descendants, nous devons déconnecter temporairement MEMOIRE CENTRALE et MEMOIRE AUXILIAIRE, pour effacer MEMOIRE; enfin, nous rattachons MEMOIRE CENTRALE et MEMOIRE AUXILIAIRE à HARDWARE.

Encore une fois, le temps ne nous a pas permis de réaliser complètement cette fonction de suppression d'un sommet non pendant.

(*) Une fonction inverse est considérée comme fonction "destructive" des informations créées par des fonctions "constructives".

c. Modification de la valeur d'un élément.

Au niveau de ce langage de manipulation général, la première valeur qu'un utilisateur désirera changer sera le libellé d'un critère. Ceci explique donc l'obligation de spécifier deux libellés:

- le premier indique le libellé du critère à remplacer
- le second fournira son nouveau libellé.

Si un utilisateur souhaite modifier la valeur du n° de version associé à un critère déterminé; il devra indiquer le libellé du critère en question, et son nouveau n° de version.

d. Modification globale du n° de version en vue de création ultérieure de critères.

Le n° de version permet de sélectionner les critères de la base de données générale que l'on veut charger dans les bases de données spécifiques. Dans certains cas, il s'avère en effet intéressant de créer dans la base de données générale:

- quelques critères caractérisés par un n° de version identique
- d'autres critères avec des n° de version différents.

Afin de ne pas devoir spécifier le n° de version lors de la création de chaque critère, on peut prévoir le fait

- de fournir au départ un n° de version par défaut
- de modifier ce n° de version en vue de créer d'autres critères correspondant à un même n° de version (différent du premier).

2.2.2 Classification des fonctions de la phase 1. (*)

a. Fonction de contrôle syntaxique.

La fonction de contrôle syntaxique permet de vérifier la validité

- du premier caractère de la commande: à
- de l'identification de la commande: verbe
- des paramètres standards
- des libellés de critères
- des paramètres spécifiques.

b. Fonction d'édition.

La fonction d'édition aide l'utilisateur en lui fournissant:

- des messages d'erreurs (syntaxiques et sémantiques) aussi précis que possible
- des messages lui permettant de se rendre compte du bon déroulement de ses commandes
- des messages de communication.

c. Fonction de contrôle sémantique.

L'existence de cette fonction nous permet de vérifier l'appartenance d'un critère déterminé dans la base de données.

(*) Afin de comprendre les termes utilisés dans ce chapitre, il faut se référer au dossier utilisateur (cfr chapitre III).

d. Fonction d'enregistrement.

C'est grâce à la fonction d'enregistrement que nous pouvons réaliser la création d'un critère et/ou de ses paramètres dans la base de données. De plus, cette fonction de création d'un critère peut facilement être généralisée en vue de la création d'une branche.

e. Fonction de transformation.

La fonction de transformation nous est nécessaire afin de réaliser:

- des modifications dans la base de données, qu'il s'agisse du libellé d'un critère ou de la valeur d'un de ses paramètres
- le changement du n° de version
- la suppression d'un critère, d'une branche de critères ou de l'entièreté de la base de données.

f. Fonction de manipulation.

A ce stade, la fonction de manipulation nous permet de réaliser l'impression de la structure arborescente, soit en entier, soit à partir d'un certain critère, soit jusqu'à un certain niveau...

2.3 Analyse des traitements de la phase 2.

2.3.1 Création de la base de données spécifique Cat par l'intermédiaire du langage de manipulation.

2.3.1.1. Création.

La création se réalise de manière similaire à la création définie dans le langage de manipulation général; il suffit de tenir compte des paramètres spécifiques à la méthode, qui peuvent se créer simultanément ou séparément avec le libellé du critère. Nous rappelons ici ces paramètres:

- pour la racine: coéfficient d'aggrégation
- pour un noeud: coéfficient d'aggrégation
  poids
- pour une feuille: poids
  fonction d'utilité.

Les paramètres spécifiques aux projets sont les libellés et les valeurs d'entrée pour chaque critère élémentaire. Nous pouvons considérer la création des informations relatives aux projets comme étant la phase finale d'introduction des données avant l'évaluation. En effet, pour Cat, les fonctions d'utilité doivent être définies avant d'y introduire les valeurs d'entrée. Nous avons donc pensé à une manière d'introduction globale de ces dernières, ceci nous permet en plus une vérification aisée de contraintes (*) à respecter avant le passage aux programmes de choix (cfr supra).

(*) Exemples de vérification de contraintes:
- tous les sommets pendants sont des feuilles ?
- la somme des poids des critères subordonnés (immédiatement) à un critère = valeur fixée ?

2.3.1.2 _ Impression.

L'impression s'effectue tout à fait de manière identique à ce que nous avons décrit pour le langage de manipulation général. Nous rappelons au lecteur que l'arborescence caractéristique pour Cat contient un grand nombre de paramètres. Si un utilisateur souhaite les connaître tous ou en partie, il doit le(s) spécifier dans la commande d'impression.

2.3.1.3 _ Modification.

La modification du contenu de la base de données Cat par le langage de manipulation qui lui est propre, s'effectue soit au niveau du libellé d'un critère, soit au niveau de la valeur d'un ou de plusieurs de ses paramètres.

Lors d'une suppression d'un critère, on supprime tous les critères correspondant aux suivants du critère supprimé, de même que leurs paramètres (*).

L'ajoute d'un nouveau critère ou de nouveaux paramètres propres à un critère déjà créé, s'effectue également de manière très simple.

(*) on ne reprend pas la suppression d'un sommet non pendant (cfr § 2.2.1.3. b).

2.3.1 bis Création de la base de données spécifique Cat par l'intermédiaire de l'interface.

La création de la base de données spécifique Cat par l'intermédiaire de l'interface n'est réalisée que partiellement. En effet, la base de données générale, dans laquelle nous allons puiser les informations ne possède que la structure arborescente des critères. Les valeurs des paramètres associés à chacun de ces critères devront donc être créées par l'intermédiaire du langage de manipulation.

2.3.2 Classification des fonctions (Cat).

La phase de création de la base de données spécifique Cat peut être découpée en fonctions, comme nous l'avons vu, et cette découpe est identique à celle que nous avons développée, pour la phase de création de la base de données générale.

2.3.3 Création de la base de données spécifique Electre par l'intermédiaire du langage de manipulation.

Le langage de manipulation Electre est en tout point semblable au langage de manipulation général, en tenant compte du fait que l'utilisateur peut préciser des paramètres.

2.3.3.1 _ Création.

La création de la racine nécessite la présence d'un seul libellé. La création d'un noeud implique la spécification de deux libellés, et éventuellement, la valeur d'un poids. La création d'une feuille demande également deux libellés et éventuellement la valeur d'un ou de plusieurs paramètres.

Les paramètres spécifiques aux projets sont les libellés et les valeurs des évaluations, pour chaque critère élémentaire. Nous pouvons considérer cette création des informations relatives aux projets, comme étant la phase finale d'introduction des données dans la base de données Electre.

2.3.1.2 _ Impression.

Il faut ajouter ici, aux caractéristiques de l'impression du langage de manipulation général, le fait qu'on imprime les paramètres spécifiques à la méthode que si on les précise.

2.3.1.4 _Modification._

L'addition d'un élément reflète exactement la création. La suppression est identique au cas général. Lors de la modification proprement dite, on peut soit

- -modifier le libellé d'un critère
- - modifier la valeur d'un de ses paramètres; dans ce cas, il faut le spécifier, de même que celle-ci.

2.3.3 bis Création de la base de données spécifique Electre par l'intermédiaire de l'interface.

La création de la base de données spécifique Electre par l'intermédiaire de l'interface n'est réalisée que partiellement. En effet, la base de données générale, dans laquelle nous puisons les informations ne possède que la structure arborescente des critères. Les valeurs des paramètres associés à chacun de ces critères devront donc être créées par l'intermédiaire du langage de manipulation.

2.3.4 Classification en fonctions (Electre).

Comme dans le cas des traitements propres à Cat, cette phase peut être découpée en fonctions, et cette découpe est aussi identique à celle que nous avons développée pour la phase de création de cette base de données générale.

2.4 Analyse des traitements de la phase 3 (Mise en forme des données).

Il est à remarquer que cette phase de mise en forme des données est très similaire pour les deux méthodes de sélection Cat et Electre: en effet, il s'agit du regroupement des informations provenant de la base de données spécifique et d'un fichier auxiliaire (pour Cat) ou du terminal (Electre), de manière à les introduire par après dans les programmes de choix. Ce traitement est exécuté à partir d'une commande introduite au terminal.

Nous y retrouvons certaines fonctions:

- contrôle syntaxique
- édition d'erreur — pour l'interprétation des commandes
- interprétation

- contrôle sémantique
- édition des erreurs — pour l'exécution des commandes.
- manipulation

Dans le cas d'Electre, nous devons également introduire les données correspondant aux seuils de concordance, par le terminal, tandis que pour Cat, le fichier auxiliaire contient les commandes du langage Sel qui lui sont nécessaires.

2.5 Analyse des traitements de la phase 4 (Exécution des programmes de choix).

Ces programmes existaient déjà avant notre application, et les traitements y sont très différents, suivant chaque méthode; c'est pourquoi nous ne les reprenons pas ici.

DEUXIEME PARTIE: DOSSIER DE PROGRAMMATION

CHAPITRE I CHOIX D'UN S.G.B.D.

Nous avons choisi le S.G.B.D. Sphinx plutôt qu'un système de gestion sur fichiers classiques pour plusieurs raisons.

1. Le caractère complexe des manipulations que l'on effectue au niveau de l'arborescence demandait une implémentation assez importante. En effet, si notre attention fut d'abord retenue par des structures de fichiers classiques nous remarquions que cette solution nous offrait des moyens simples à implémenter lorsque nous traitions chacune des fonctions séparément; mais lorsque nous regroupions les traitements de ces différentes fonctions, les accès à implémenter devenaient particulièrement complexes.
Nous allons montrer par un exemple les différents chemins d'accès que nous devrions "emprunter" pour réaliser certaines manipulations que l'on veut effectuer au niveau de l'arborescence; cfr fig. 10.

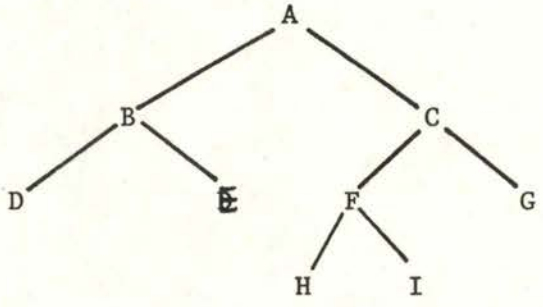


Fig. 10: exemple d'arborescence.

- les traitements de modification d'un libellé ou de paramètres spécifiques d'un critère demande un "accès direct" à ce critère
- le traitement de création d'un critère demande également un "accès direct" au critère précédent auquel on lie celui que l'on veut créer
- le traitement d'impression de l'arborescence complète demande un parcours dans cette arborescence suivant l'ordre de Tarry, c'est-à-dire dans notre exemple, l'accès dans l'ordre à A, B, D, E, C, F, G, H, I,
- le traitement de vérification de la somme des poids de critères immédiatement subordonnés à un critère demande un parcours de l'arborescence qui correspond à la phase: "pour chaque critère, visiter tous ses suivants"; un exemple de parcours est A, B, C, D, E, F, G, H, I
- l'évaluation du critère global demande un parcours de l'arborescence qui correspond à la phase: "visiter un critère après avoir visité tous ses suivants"; l'ordre est ici: D, E, B, H, I, F, G, C, A.

2. Le système Sphinx présente des avantages non négligeables, dont:
   - sa simplicité et sa généralité
   - son utilisation en batch aussi bien qu'en interactif
   - l'utilisation simultanément partagée de la base de données
   - son optimisation maximale des accès aux disques
   - son système de gestion dynamique et implicite de la mémoire centrale et des demandes d'accès
   - sa possibilité d'interroger ponctuellement ou/et en masse la base de données.

Le but poursuivi par l'équipe des "Grands Fichiers", lors de l'implémentation du modèle d'accès était de fournir aux utilisateurs, un système de structuration et d'exploitation de base de données qui soit aussi général et simple que possible.

L'utilisateur verra donc la structure de la base de données comme un graphe (que nous appellerons graphe base de données ou schéma de base de données, pour ne pas confondre avec la structure arborescente que nous appellerons plutôt graphe de choix) dont les sommets sont les objets et les arcs, des relations d'accès.

Lorsqu'on parle de langage de base de données, on s'intéresse plus spécialement aux types de données et aux actions primitives agissant sur ces données. Pour combiner ces primitives, on utilisera les mécanismes d'un langage hôte (Cobol, dans notre cas).

CHAPITRE II GENERALITES SUR LE S.G.B.D. SPHINX.

Un langage de base de données se compose de deux parties:

- le langage de définition des données (D.D.L.) définissant les types de données qui constituent le schéma de la base de données
- le langage de manipulation des données (D.M.L.) formé d'un ensemble de primitives agissant sur les types de données.

2.1 Types de données.

Il existe deux types de données: les objets et les relations.

Un objet correspond à une classe homogène d'informations; chaque élément de cette classe est une réalisation de l'objet. L'ensemble des objets de la base de données peut être partitionné en trois sous-classes:

- l'objet racine (*)
- les objets complexes: correspondant à des types d'entités logiques d'information et n'ayant aucune réalisation, a priori
- les objets élémentaires qui correspondent à des types de valeurs; l'ensemble des valeurs d'un objet élémentaire est défini a priori, et une fois pour toutes.

(*) L'objet racine permet
- de désigner une base de données ainsi que son schéma dans un ensemble de bases de données
- d'y accéder pour ouvrir la base de données
- à partir de cette racine, d'accéder aux réalisations d'objets complexes (accès séquentiel ou par clé)

Une relation est associée à un couple d'objets; tandis qu'une occurrence de relation est associée à un couple de réalisations de ces objets.

Toute relation est caractérisée par 4 paramètres: (i,j,k,l) assurant la cohérence de la base de données: si on a R tel que

A ——R—— B

R donne accès à partir de toute réalisation de A, à au moins i et au plus j réalisations de B
R permet d'accéder à toute réalisation de B à partir d'au moins k et au plus l réalisations de A.

Toute modification de l'ensemble des occurrences d'une relation s'effectue en accord avec les caractéristiques de la relation et l'existence d'une relation inverse (*). La gestion de cette dynamique n'est pas confiée aux utilisateurs.

2.2 Primitives.

Il existe plusieurs sortes de primitives:

- des primitives d'accès qui permettent la recherche de l'information, c.à.d. à partir d'une réalisation de l'origine d'une relation, fournissent toutes les réalisations cibles
- des primitives de
  - création/suppression d'une réalisation d'objet complexe
  - création/suppression d'une occurrence de relation
  - modification d'une occurence de relation.

(*) Relation inverse: deux relations R(A,B) et S(A,B) sont déclarées inverses l'une de l'autre; cela a pour conséquence:
- à toute occurrence (a,b) de R correspond l'occurrence (b,a) de S
- à toute occurrence (b,a) de S correspond l'occurrence (a,b) de R,

pour $a \in A$ et $b \in B$.

2.3 Langage de description des données (D.D.L.) de Sphinx.

Le but d'un programme D.D.L. (*) est de déclarer un schéma de base de données. La "compilation" de ce programme aura pour effet de vérifier et de générer un certain nombre de tables et de procédures permettant d'initialiser et d'exploiter une base de données. La base de données peut alors être exploitée par un ou des programmes D.M.L. (*)

Ce langage de description des données comporte notamment

- une clause d'identification autorisant l'accès à la base de données
- une clause permettant la définition d'un schéma de base de données
- les déclarations de la racine, qui associe son nom à la base de données, des objets complexes et objets élémentaires, et enfin des relations ainsi que leur domaine de définition.

Ce programme D.D.L. est compilé par un compilateur effectuant certaines fonctions telles que:

- analyse syntaxique
- vérification des contraintes d'intégrité
- génération des procédures Sesam de chargement et des différents schémas internes.

(*) Un programme D.M.L. est écrit au moyen du langage de manipulation des données (D.M.L.)
Un programme D.D.L. est écrit au moyen du langage de description des données (D.D.L.)

On peut donc à ce moment construire un schéma de base de données représentant les types de données et les liaisons entre elles. Il faut bien se rendre compte que lorsque ce schéma est défini, il ne permet que la création d'une base de données y correspondant; c.à.d. que l'on peut agir uniquement sur les occurrences des relations et les réalisations des objets qui y ont été définis.

2.4 Langage de manipulation des données (D.M.L.) de Sphinx.

Le langage de manipulation des données de Sphinx comprend un certain nombre d'ordres qui peuvent se regrouper en trois catégories:

- des ordres associés à la primitive d'accès:
  - REACH: accès à une réalisation
  - OPEN: ouverture de la base de données
- des ordres associés aux primitives de mise à jour:
  - PREPARE CREATE SUPPRESS: pour la création et la suppression des objets complexes
  - ATTACH DETACH TRANSFER: associés aux relations
  - MODIFY: modifications des valeurs des objets élémentaires associés à une réalisation d'un objet complexe
- des mécanismes de programmation:
  - CLOSE: fermeture de la base de données
  - ALLOCATE: modification de la gestion automatique de la mémoire
  - END: fin de contexte et itération
  - EXIT: fin d'itération prématurée
  - NEXT: itération prématurée
  - ...

CHAPITRE III DESCRIPTION DES INFORMATIONS RELATIVES AUX DIFFERENTES BASES DE DONNEES.

3.1 Remarque.

Pour des raisons de présentation et de compréhension plus aisée pour le lecteur, nous regroupons ici l'ensemble des informations que nous retrouvons dans les différentes bases de données.

Nous reprenons donc dans ce chapitre la description des unités d'information décrite dans le dossier de développement et nous y adjoignons la description des données de gestion nécessaires à l'exécution des manipulations sur l'arborescence ainsi que la justification du choix de ces données.

Nous suivons dans ce chapitre la chronologie des phases définie dans le dossier de développement: cfr fig. 1.

3.2 Description des informations relatives à la base de données générale.

3.2.1 Description des objets complexes et des objets élémentaires.

a. les objets complexes.

Au type d'entité NODE nous faisons correspondre l'objet complexe NODE.

Dans le cas précis où l'on veut accéder (avec un nombre d'accès minimum) à la racine de l'arborescence (= une réalisation de l'objet complexe NODE), sans devoir citer une caractéristique précise (ex: une clé) de cette racine, il s'avère intéressant de créer un autre objet complexe que nous appelerons SYSTEM, de manière à aboutir de manière la plus rapide possible à la réalisation de la racine du graphe de choix.
En effet, si on ne précise pas la réalisation de NODE à laquelle on veut accéder, le système pointe vers la réalisation de l'objet complexe NODE caractérisé par le plus petit aspect d'ordre (c.à.d. la valeur de la clé d'accès du système Sesam); cela signifie dans notre cas, que le système pourrait accéder

- soit à une réalisation correspondant à une feuille
- soit à une réalisation correspondant à un noeud.

Dans ces deux cas, cela nécessite l'utilisation d'un chemin d'accès, qui partirait de la réalisation atteinte à la réalisation correspondant à la racine du graphe de choix.

b. Les objets élémentaires associés à l'objet complexe NODE.

Nous avons quatre objets élémentaires associés à l'objet complexe NODE; à chaque réalisation de l'objet complexe correspond une et une seule réalisation de chacun des objets élémentaires. Nous les avons dénommés:

- NOM: (*)
- VERSION: (*)
- NIVEAU: le niveau est un nombre qui détermine la longueur du chemin entre la racine et un critère; c'est donc une caractéristique d'un sommet de l'arborescence qui permet une découpe horizontale de celle-ci; cette donnée est nécessaire afin de minimiser le nombre d'accès: en effet, lors d'une impression d'une branche de l'arborescence (affichage au terminal), nous imposons l'impression du niveau auquel appartient chacun des critères de cette branche; dans le cas de non existence de cette information, il nous faudrait calculer la longueur du chemin entre la racine du graphe et le premier critère de la branche (celui de niveau le plus bas); cette donnée est également nécessaire pour une gestion aisée d'autres manipulations,
- ORDRE: c'est un nombre caractérisant un sommet qui permet d'identifier ce sommet parmi tous les sommets immédiatement subordonnés à un même sommet; certaines manipulations sur l'arborescence sont récursives; or la structure de bloc définie par le S.G.B.D. Sphinx exclut l'utilisation de cette récursivité; afin déviter cette contrainte, nous utilisons un ensemble de piles de manière à connaître à tout moment quel est le n° de critère (fils) que l'on est en train de traiter par rapport au critère précédent (père); un exemple d'utilisation de ces piles est illustré dans les paragraphes suivants.

(*) cfr Dossier de développement

c. L'objet élémentaire associé à l'objet complexe SYSTEM.

- NOM: permet d'identifier l'objet complexe SYSTEM.

3.2.2 Résumé des objets complexes et objets élémentaires.

Nous allons reprendre la méthode (D.D.L. de Sphinx) de description des articles, de manière à préciser davantage les libellés, les formats et les niveaux de rubrique de nos informations.

a. Objet complexe SYSTEM.

```
01 NOM PIC X(6)
```

b. Objet complexe NODE.

```
01 NOM PIC X(27)
01 NIVEAU PIC 9(2)
01 ORDRE PIC 9(2)
01 VERSION PIC 9
```

3.2.3 Règles de vérification et quantification.

a. Vérifications.

- longueur $\leq$ 27 caractères alphanumériques; cette limitation est due essentiellement au fait que l'on identifie le nom du critère comme étant la clé d'accès et (cette clé d'accès) possède une longueur de 27 caractères
- 0 $\leq$ niveau $\leq$ 99 N(umérique); on conçoit difficilement une arborescence composée de plus d'une dizaine de niveaux; le niveau associé à un critère doit être égal à la longueur du chemin entre la racine de l'arborescence et ce critère
- 0 $<$ ORDRE $\leq$ 99 N

b. Longueurs totales maximum des valeurs d'objets élémentaires associés à une réalisation d'objet complexe.

Objet complexe SYSTEM: 6 caractères

Objet complexe NODE: 32 caractères

N.B. : nous ne reprenons pas dans ces longueurs, les pointeurs et compteurs associés à chaque réalisation d'objet complexe.

c. Dimension.

Il est bien évident que la dimension est proportionnelle au nombre de critères crés par l'utilisateur (N). Nous pouvons en déduire le volume brut total maximum pour un système:

(6 + 32 * N) caractères.

3.2.4 Relations.

Nous définissons deux relations d'accès inverses l'une de l'autre:

NODE —FILS— NODE

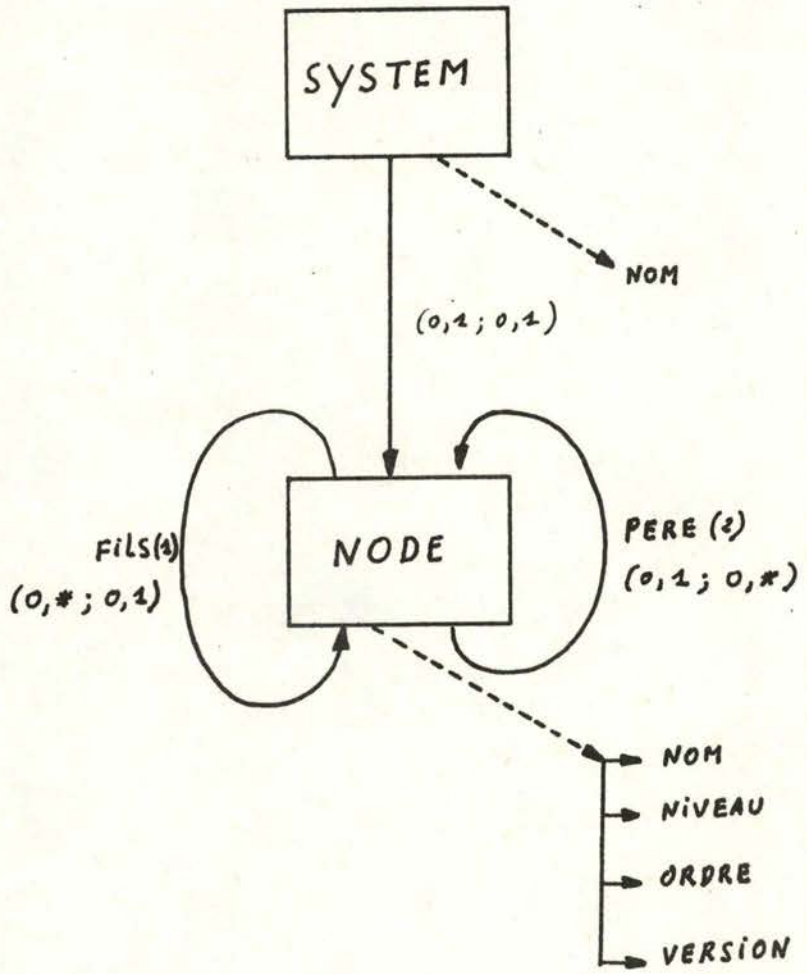
- cette relation nous permet de définir un chemin d'accès entre le père et le fils (précédent-suivant); cette relation est nécessaire, notamment pour le traitement de l'impression; ses caractéristiques sont: (0,*;0,1) (*)

NODE —PERE— NODE

- cette relation permet de définir un chemin d'accès entre le fils et le père; cette relation est nécessaire notamment pour le traitement de la suppression d'un critère (non pendant); ses caractéristiques sont (0,1;0,*)

(*) * est pris pour ∞ dans les caractéristiques d'une relation.

3.2.5 Le schéma.



Dans la description du D.D.L. (cfr listings)

(1): relation sans nom

(2): relation de nom FATHER

3.3 Description des informations relatives à la base de données Cat.

3.3.1 Description des objets complexes et des objets élémentaires.

a. Les objets complexes.

Au type d'entité NODE, nous faisons correspondre l'objet complexe NODE.

Nous définissons également l'objet complexe SYSTEM, pour les mêmes raisons que celles décrites dans le paragraphe 3.2.1.

Vu l'apparition très rare des relations particulières (violant les règles de l'arborescence) et afin d'éviter l'utilisation d'enregistrements de longueurs variables, ainsi que les complications apportées lors des modifications de ces enregistrements, il est de loin préférable d'introduire un objet complexe supplémentaire (FILIATION) qui caractérise donc une relation particulière entre deux réalisations de l'objet complexe NODE.

b. L'objet élémentaire associé à l'objet complexe FILIATION.

- POIDS: dans le cas particulier de la méthode de sélection Cat, le poids caractérise une relation d'évaluation; si nous considérons la figure 3 du dossier de développement, on considère que le poids 60 est associé à PRODUCTIVITE MESUREE pour l'évaluation du CRITERE INTERMEDIAIRE et que le poids 50 est associé à la PRODUCTIVITE MESUREE pour l'évaluation du critère PRODUCTIVITE GLOBALE; afin d'éviter des accès supplémentaires, nous introduisons l'information POIDS au niveau de l'objet complexe NODE et ceci uniquement pour les relations non particulières.

c. L'objet élémentaire associé à l'objet complexe SYSTEM.

- NOM: (*2)

d. Les objets élémentaires associés à l'objet complexe NODE.

- NOM : (*1)
- NIVEAU: (*2)
- ORDRE: (*2)
- POIDS: (*1)
- AGGREGAT: (*1)
- FONCTION: (*1)
- PROJET: (*1)
- TYPE-FONCT: est un numéro qui permet d'identifier le type d'un critère; vu qu'à un sommet pendant ne correspond pas nécessairement un critère élémentaire, il est dès lors impossible de retrouver ce type de critère par le biais des relations; c'est la raison pour laquelle nous identifions un critère élémentaire par une information; nous raisonnons de même pour les noeuds; les numéros attribués à chacun des types de critères sont:
  - le numéro 1 pour le critère global
  - le numéro 2 pour les critères non élémentaires
  - le numéro 3 pour les critères élémentaires
- TYPE-REL: est une information qui permet de caractériser le type de relation entre deux critères;
  cette information prend les valeurs:
  0: relation père-fils
  1: relation particulière (violant les règles de l'arborescence)

(*1): cfr description des unités d'information; dossier de développement, §1.3.2.1
(*2): cfr description des informations relatives à la base de données générale; dossier de programmation, §3.2.1

- NOSEQ: est un numéro qui permet d'attribuer une séquence entre les critères

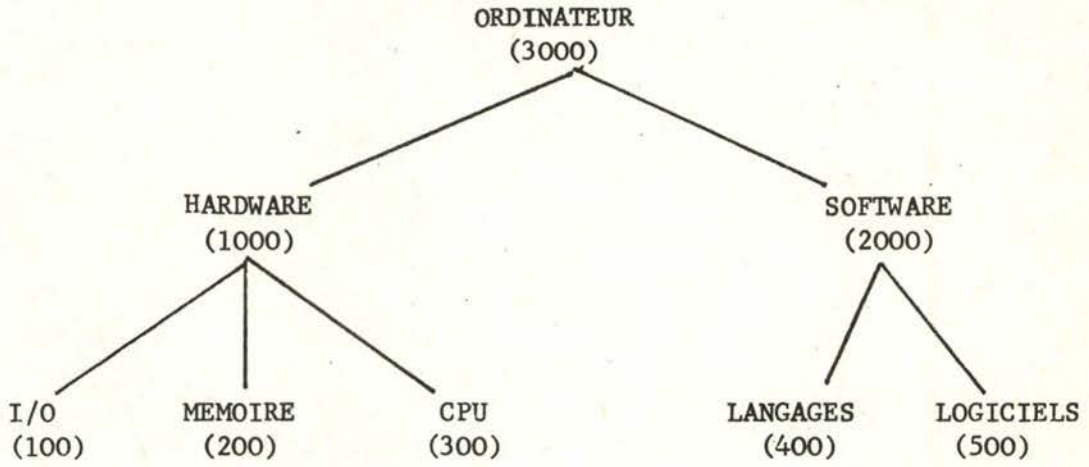


Fig. 12: exemple d'arborescence où nous avons repris les n° de séquence et où les critères élémentaires correspondent aux sommets pendants.

L'évaluation d'un critère non élémentaire nécessite la connaissance de l'évaluation de ses critères immédiatement subordonnés. La méthode de sélection Cat impose l'attribution de n° de séquence aux critères de façon à ce que tous les n° de séquence se présentent de manière croissante suivant le sens de l'évaluation.

Si nous reprenons l'exemple de la fig. 12; l'algorithme évalue d'abord tous les critères élémentaires dans l'ordre suivant: I/O, MEMOIRE, CPU, LANGAGES, LOGICIELS. Ensuite, il procède à l'évaluation du critère HARDWARE (en fonction des critères élémentaires I/O, CPU et MEMOIRE) et ensuite du critère SOFTWARE; il procède alors à l'évaluation du critère global: ORDINATEUR.

3.3.2 Résumé des objets complexes et objets élémentaires.

a. Objet complexe SYSTEM.

```
01 NOM PIC X(6)
```

b. Objet complexe NODE.

```
01 NOM PIC X(27)
01 NIVEAU PIC 9(2)
01 ORDRE PIC 9(2)
01 TYPE-FONCT PIC 9
01 TYPE-REL PIC 9
01 POIDS PIC 9(3)
01 AGGREGAT PIC X(3)
01 FONCTION
        02 EL-FONCTION OCCURS 5
                03 ABC-FONCT PIC 9(4)
                03 ORD-FONCT PIC 9(2)
01 PROJET OCCURS (0,20)
        02 IDENT PIC X(10)
        02 VALEUR PIC 9(5)
01 NOSEQ PIC 9(4)
```

c. Objet complexe FILIATION.

01 POIDS PIC 9(3)

3.3.3 Règles de vérification et quantification.

a. Vérifications.

-TYPE-FONCT: cette information doit prendre une des trois valeurs: 1, 2 ou 3

-TYPE-REL: cette information doit prendre une des deux valeurs: 1 OU 0

-NOSEQ: $0 \leq NOSEQ \leq 9999$

-POIDS(FILIATION): mêmes contraintes que le poids associé à l'objet complexe NODE

-Les autres règles de vérification associées aux informations citées sont décrites soit

- dans le paragraphe de description des informations de la base de données générale (3.2.3)
- soit dans le paragraphe des analyse des unités d'information (dossier de développement).

b. Longueurs totales maximum et minimum des valeurs d'objets élémentaires associés à une réalisation d'objet complexe.

Objet complexe SYSTEM: 6 caractères

Objet complexe NODE: 73 caractères minimum

973 caractères maximum

Objet complexe FILIATION: 3 caractères.

N.B.: Nous ne reprenons pas dans ces longueurs les pointeurs et les compteurs associés à chaque réalisation d'objet complexe.

c. Dimension.

La dimension est proportionnelle au nombre de critères (N) créés par l'utilisateur, de même qu'au nombre de relations particulières (F) et au nombre de paramètres spécifiques.

On peut en déduire le volume brut total maximum pour un système:

6 + (373 * N) + (3 * F) caractères au maximum

3.3.4 Les Relations.

SYSTEM----------NODE: (0,1;0,1)

NODE----père----NODE: (0,1;0,*)

NODE----fils----NODE: (0,*;0,1)

FILIATION----poids-père----NODE: (0,1;0,1)

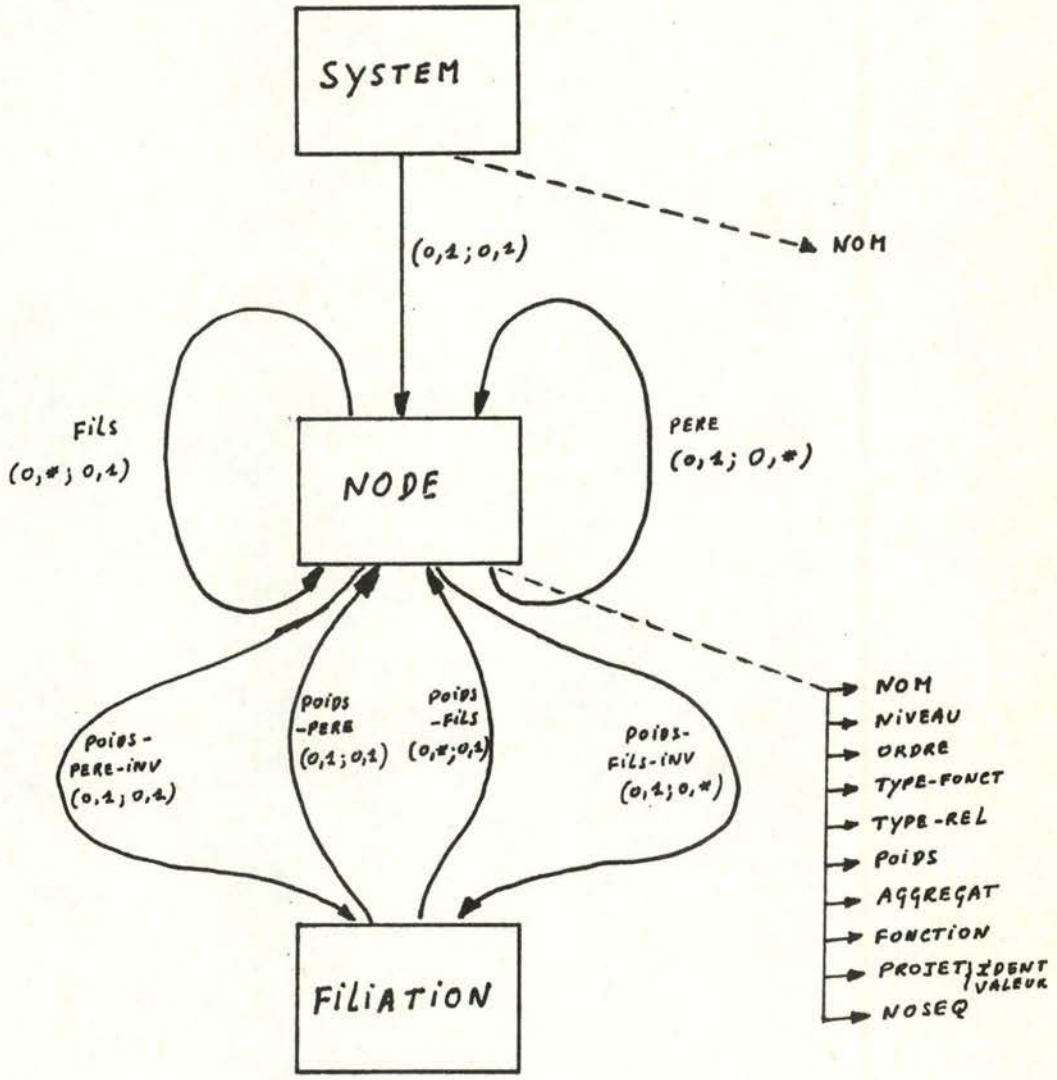
FILIATION----poids-fils----NODE (0,*;0,1)

NODE----poids-père-inv----FILIATION: relation inverse de poids-père

NODE----poids-fils-inv----FILIATION: relation inverse de poids-fils.

Certaines relations ne sont pas nécessaires pour les manipulations que l'on effectue sur l'arborescence, mais nous les avons introduites de manière à éviter une nouvelle compilation du schéma si on crée des manipulations utilisant ces relations.

3.3.5 Le schéma.



3.4 Description des informations relatives à la base de données Electre.

3.4.1 Description des objets complexes et des objets élémentaires.

a. les objets complexes.

NODE: cfr (*1)
SYSTEM: (*1)

b. les objets élémentaires associés à l'objet complexe NODE.

NOM: (*3)
ORDRE: (*2)
NIVEAU: (*2)
POIDS: (*3)
ECHELLE: (*3)
SD1: (*3)
SD2: (*3)
SQN: (*3)
TYPE-FONCT (*2)
PROJET (*3)

c. L'objet élémentaire associé à l'objet complexe SYSTEM.

NOM: (*1)

(*1): cfr description des objets complexes et élémentaires (§ 3.2.1)
(*2): cfr description des objets complexes et élémentaires (Cat: § 3.3.1)
(*3): cfr analyse des unités d'information (dossier de développement) : § 1.3.3

3.4.2 Résumé des objets complexes et objets élémentaires.

a. Objet complexe SYSTEM.

```
01 NOM PIC X(6)
```

b. Objet complexe NODE.

```
01 NOM PIC X(27)
01 NIVEAU PIC 9(2)
01 ORDRE PIC 9
01 POIDS PIC 9(5)
01 ECHELLE PIC 9(2)
01 SD1 PIC 9(5)
01 SD2 PIC 9(5)
01 TYPE-FONCT PIC 9
01 SQN PIC 9(5)
01 PROJET OCCURS (0,20)
        02 IDENT PIC X(27)
        02 VALEUR PIC 9(5)
```

3.4.3 Règles de vérification et quantification.

a. Vérifications.

Les règles de vérification associées aux informations citées sont décrites soit:

- dans le dossier de développement (cfr Analyse des unités d'information)
- dans le paragraphe: règles de vérification et quantification relatif à la base de données générale (§ 3.2.3)
- dans le paragraphe: règles de vérification et quantification relatif à la base de données Cat (§ 3.3.3)

b. Longueurs totales maximum et minimum des valeurs d'objets élémentaires associés à une réalisation d'objet complexe.

Objet complexe SYSTEM: 6 caractères

Objet complexe NODE: 31 caractères, au minimum

693 caractères, au maximum

N.B: Nous ne reprenons pas dans ces longueurs les pointeurs et les compteurs associés à chaque réalisation d'objet complexe.

c. Dimension.

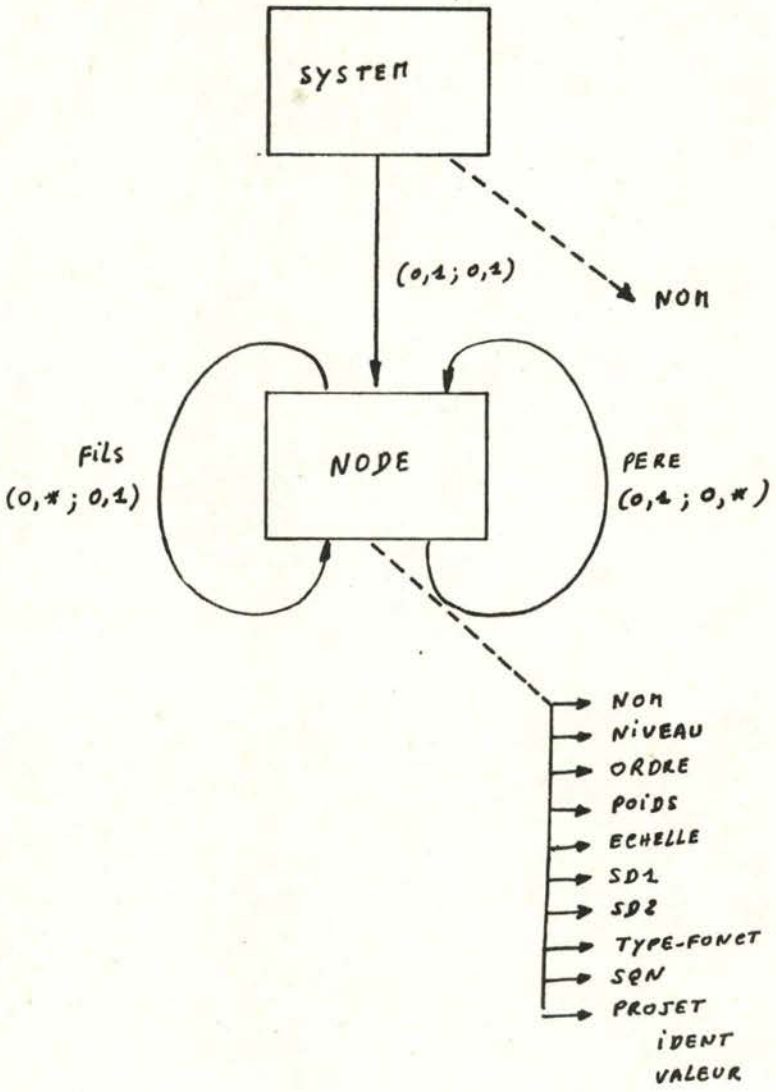
La dimension est proportionnelle au nombre de critères (N) créés par l'utilisateur, de même qu'au nombre de paramètres spécifiques. On peut en déduire le volume brut total maximum pour un système:

6 + (693 * N) caractères, au maximum.

3.4.4 Les Relations.

Les relations entre objets complexes sont identiques à celles définies dans le cas du schéma général (§ 3.2.4)

3.4.5 Le schéma.



CHAPITRE IV DESCRIPTION DES TRAITEMENTS, PHASE PAR PHASE.

4.1 Description des traitements relatifs à la phase 1.

La description de l'interprétation des commandes est effectuée principalement par l'intermédiaire de tables de décision, tandis que la description des traitements associés à chaque commande s'effectue par un pseudo-langage.

Par abus de langage, dans l'exposé des traitements, nous écrivons:

- "création d'un critère" pour création des objets élémentaires correspondant à une réalisation de l'objet complexe NODE
- "accès à un critère" pour accéder à l'ensemble des objets élémentaires correspondant à une réalisation de l'objet complexe NODE dont l'objet élémentaire NOM correspond au libellé de ce critère,
- "création de l'objet complexe SYSTEM" pour création de l'objet élémentaire associé à la réalisation de l'objet complexe
- "liaison de critères" pour liaison de réalisations.

4.1.1 Détection du type de commande.

PAS 1: pour chaque commande (*2)

PAS 1.1: contrôle (syntaxique) du premier caractère différent du caractère blanc

PAS 1.1.1: si ce 1er caractère = à (*), aller au PAS 1.2

PAS 1.1.2: sinon: - imprimer M1 (*3)
- aller au PAS 1

PAS 1.2: contrôle (syntaxique) du type de commande (D41) : (*1)

PAS 1.2.1: si la 1re lettre = C, aller à la Table C1

PAS 1.2.2: sinon, si 1re lettre = M, aller à la Table M1

PAS 1.2.3: sinon, si 1re lettre = P, aller à la Table P1

PAS 1.2.4: sinon, si 1re lettre = S, aller à la Table S1

PAS 1.2.5: sinon, si 1re lettre = A, aller à la Table A1

PAS 1.2.6: sinon, si 1re lettre = I, aller à la Table I1

PAS 1.2.7: sinon, si 1re lettre = D, aller à la Table D1

PAS 1.2.8: sinon, si 1re lettre = H, aller à la Table H1

PAS 1.2.9: sinon, - imprimer M2
- aller au PAS 1

(*) Le premier caractère (différent du caractère d'espacement) d'une commande doit être à, et ceci, afin de garder à l'esprit le fait que l'on se trouve dans un langage interactif.

(*1) Les définitions: D41 ... sont données à la page 76 bis.

(*2) Afin de comprendre les termes utilisés dans ce paragraphe, nous renvoyons le lecteur au dossier utilisateur (chapitre II).

(*3) Les messages sont repris à la page 84.

4.1.2 Interprétation du type de commande: à CREATE.

L'enchaînement des tables de décision décrites ici, permet l'interprétation des commandes:

- à CREATE FIRST
- à CREATE NODE
- à CREATE LEAF
- à CREATE BRANCH

Table C_1

caractère +1 $\stackrel{?}{=}$ R (D42)	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N
caractère +1 = E	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N	
caractère +1 = A	Y	Y	Y	Y	N	N	N	N	Y	Y	Y	Y	N	N	N	N	
caractère +1 = T	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N	
caractère +1 = E	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	
aller à la table C_2	X																X
imprimer H_3 aller au pas 1		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	

Table C_2

1er caractère ≠ b (D43)	F	N	B	
aller à la table C_3	X			
aller à la table C_4		X		
aller à la table C_5			X	
imprimer H_4 aller au pas 1				X

Table C_3

caractère +1 = I	Y	Y	Y	Y	Y	Y	Y	Y	N.
caractère +1 = R	Y	Y	Y	Y	N	N	N	N	
caractère +1 = S	Y	Y	N	N	Y	Y	N	N	
caractère +1 = T	Y	N	Y	N	Y	N	Y	N	
aller à la table C_6	X								X
imprimer H_5 aller au pas 1		X	X	X	X	X	X	X	

Table C_4

caractère +1 = 0	Y	Y	Y	Y	N
caractère +1 = D	Y	Y	N	N	
caractère +1 = E	Y	N	Y	N	
aller à la table C_7	X				X
imprimer M_3 aller au pas 1		X	X	X	

Table C_5

caractère +1 = R	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N
caractère +1 = A	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N	
caractère +1 = N	Y	Y	Y	Y	N	N	N	N	Y	Y	Y	Y	N	N	N	N	
caractère +1 = C	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N	
caractère +1 = #	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	
aller à la table C_8	X																X
imprimer M_5 aller au pas 1		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	

Table C_6

1er caractère ≠ b = "	Y	Y	N
nb de caract. avant le " suivant < 27	Y	N	
création de la Racine	X		
imprimer M_6 aller au pas 1			X
imprimer M_8 aller au pas 1		X	

Table C_7									
1er caract ≠ b = "	Y	Y	Y	Y	Y	Y	Y	Y	N
nb de caract. avant le " suivant ≤ 27	Y	Y	Y	Y	N	N	N	N	
1er caract ≠ b = "	Y	Y	N	N	Y	Y	N	N	
nb de caract. avant le " suivant ≤ 27.	Y	N	Y	N	Y	N	Y	N	
création d'un mot-clé	X								
imprimer H_6 aller au pas 1									X
imprimer H_7 aller au pas 1			X	X					
imprimer H_8. aller au pas 1		X			X	X	X	X	

Table C_8			
1er caract. ≠ b = "	Y	Y	N
nb de caract. avant le " suivant ≤ 27	Y	N	
initialiser NIVEAU=0 aller à la table C_9	X		
imprimer H_8 aller au pas 1		X	
imprimer H_6 aller au pas 1			X

Table C_9			
accepter une ligne du terminal.	X	X	X
1er caract ≠ b est numérique	Y	Y	N
longueur du niveau ≤ 2 caractères	Y	N	
aller à la table C_{11}	X		
imprimer H_{12} aller à la table C_9		X	
aller à la table C_{10}			X

Table C_{10}						
caractère = E	Y	Y	Y	Y	N	
caractère +1 = N	Y	Y	N	N		
caractère +1 = D	Y	N	Y	N		
aller au pas 1	X					
imprimer H_{11} aller à table C_9		X	X	X	X	

Table C_{11}				
0 < NIVEAU (N) = (D44)	NIVEAU (A) + 1	NIVEAU (A)	< NIVEAU (A)	
imprimer H_{13} aller table C_9.				X
CODE = 1 aller table C_{12}	X			
CODE = 2 aller table C_{12}		X		
CODE = 3 aller table C_{12}			X	

Table C_{12}												
CODE =	1				2				3			
1er caract. ≠ b = "	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N
nb de caract. avant le ' suivant ≤ 27	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N
CREATE FILS (D45)	X											
CREATE FRERE (D46)					X							
CREATE AUTRE (*)									X			
imprimer H_8 aller à table C_9		X				X				X		
imprimer H_6 aller à table C_9.			X	X			X	X			X	X

_Définitions:

D41: le type d'une commande est défini à partir de la première lettre du verbe qui la compose

D42: caractère + 1: on va lire le caractère suivant de la commande

D43: 1er caractère ≠ b; on va lire le caractère suivant, tant que celui-ci n'est pas un caractère d'espacement.

D44: niveau (A): c'est la valeur du niveau précisé lors de la dernière création d'un critère par la commande de création d'une branche; ce niveau est initialisé à 0.

D45: on appelle fils d'un critère, le suivant de ce critère

D46: on appelle frère d'un critère, un critère possédant le même précédent que ce dernier critère, et étant du même niveau.

4.1.3 Traitements relatifs au type de commande: à CREATE.

4.1.3.1 Traitement de la création de la racine.

PAS 1: création de l'objet complexe SYSTEM (transparent pour l'utilisateur)

PAS 2: création de la racine de l'arborescence

PAS 3: liaison de cette dernière avec l'objet complexe SYSTEM

PAS 4: - imprimer M50
- aller au PAS 1 de la phase 1

4.1.3.2 Traitement de la création d'un sommet pendant.

PAS 1: accès à son précédent dans la base de données

PAS 1.1: si son précédent existe, aller au PAS 2

PAS 1.2: sinon, - imprimer M10
- aller au PAS 1 de la phase 1

PAS 2: création du nouveau critère

PAS 3: liaison avec son précédent

PAS 4: - imprimer M50
- aller au PAS 1 de la phase 1.

4.1.3.3 Traitement de la création d'une branche de critères.

C'est une suite de traitements de création de sommets pendants (cfr § 4.1.3.2). L'avantage de création d'une branche de critères par rapport à la création consécutives de sommets pendants est bien sûr de ne pas devoir spécifier le précédent pour chacune des créations. Ceci permet donc d'éviter une perte de temps et un risque d'erreur supplémentaire.

Rem: on distingue ici le traitement de création de la racine du traitement de création d'un sommet pendant; de plus, on ne décrit pas le traitement de création d'un noeud correspondant à l'insertion, car nous ne l'avons pas retenu.

4.1.4 Interprétation des autres types de commandes.

Les tables M1, S1, P1, A1, I1, D1, H1, sont construites de manière similaire à celles que nous avons développées au § 4.1.2; c'est pourquoi nous ne les décrivons plus ici.

4.1.5 Traitements relatifs aux autres types de commandes.

4.1.5.1 Traitement de modification du libellé d'un critère.

PAS 1: accès au critère dont nous voulons changer le nom

PAS 1.1: si ce critère existe, aller au PAS 2

PAS 1.2: sinon: - imprimer M10
- aller au PAS 1 du traitement de détection du type de commande

PAS 2: modification du libellé du critère
imprimer M51
aller au PAS 1 de la phase 1

4.1.5.2:_ Traitement_de modification_du n°_de version.

PAS 1: accès au critère dont nous voulons changer le n° de version

PAS 1.1: si ce critère existe, aller au PAS 2

PAS 1.2: sinon, - imprimer M10
- aller au PAS 1 de la phase 1

PAS 2: vérification du n° de version

PAS 2.1: si l'ancien n° de version = le nouveau n° de version
- imprimer M51 et M9
- aller au PAS 1 de la phase 1

PAS 2.2: sinon:
- changer l'ancien n° de version (le remplacer par le nouveau)
- imprimer M51
- aller au PAS 1 de la phase 1

4.1.5.3 Traitement de l'impression.

PAS 1: accès au critère correspondant à la racine du graphe de choix

PAS 1.1: si ce critère existe:
- ~~imprimer M53~~
- aller au PAS 2

PAS 1.2: sinon:
- imprimer M53
- aller au PAS 1 de la phase 1

PAS 2: accès au critère correspondant au suivant du dernier critère accédé

PAS 2.1: si ce critère existe:
- imprimer son libellé
- aller au PAS 2

PAS 2.2: sinon, accès au critère correspondant au "frère" du dernier critère accédé

PAS 2.2.1: si ce critère existe:
- imprimer son libellé
- aller au PAS 2

PAS 2.2.2: sinon: retour au critère correspondant au critère précédent du dernier critère accédé

PAS 2.2.2.1: si ce critère existe, aller au PAS 2.2

PAS 2.2.2.2: sinon:
- imprimer M54
- aller au PAS 1 de la phase 1.

Remarque: l'ordre de visite aux critères correspond à l'ordre de Tarry.

4.1.5.4 _ Traitement de la suppression.

PAS 1: accès au critère correspondant à la racine du graphe de choix

PAS 1.1: si ce critère existe, aller au PAS 2

PAS 1.2: sinon:
- imprimer M53
- aller au PAS 1 de la phase 1

PAS 2: accès au critère correspondant au suivant du dernier critère accédé

PAS 2.1: si ce critère existe, aller au PAS 2

PAS 2.2: sinon:
- supprimer le dernier critère accédé
- accès au critère correspondant au frère du critère supprimé

PAS 2.2.1: si ce critère existe, aller au PAS 2

PAS 2.2.2: sinon, accès au critère correspondant au précédent du dernier critère accédé

PAS 2.2.2.1: si ce critère existe, aller au PAS 2

PAS 2.2.2.2: sinon:
- imprimer M55
- aller au PAS 1 de la phase 1.

4.1.5.5 Traitement du changement global du n° de version.

Il s'agit uniquement de la modification de la valeur d'un paramètre global au programme.

4.1.5.6 Traitement de fin de manipulation de la base de données.

PAS 1: fermeture de la base de données

PAS 2: impression de M57

Remarque: la base de données doit être ouverte lors du lancement de l'exécution du programme, pour des raisons que nous avons définies dans le dossier utilisateur

Messages de refus de commande.

M1: erreur sur le premier caractère de la commande (à).

M2: erreur sur le type de la commande.

M3: erreur sur le verbe de la commande.

M4: erreur sur le type du paramètre standard.

M6: erreur: pas d'identificateur de critère.

M5: erreur sur le paramètre standard.

M7: erreur: le nombre d'identificateurs de critères est incorrect.

M8: erreur sur la longueur de l'identificateur d'un critère.

M9: erreur: le n° de version est incorrect.

M10: le critère spécifié n'existe pas dans la base de données.

M11: erreur sur la valeur du niveau.

M12: erreur sur la longueur de la valeur d'un niveau.

M13: erreur de structure lors de la création d'une branche.

Messages d'acceptation de la commande:

M50: création terminée.

M51: modification terminée.

M53: la base de données est vide.

M54: impression terminée.

M55: suppression terminée.

M56: le nouveau n° de version est ...

M57: la base de données est fermée.

4.2 Description des traitements relatifs à la phase 2.

4.2.1 Interprétation des commandes.

L'interprétation des commandes se fait également de manière similaire à ce que nous avons décrit précédemment (cfr § 4.1.1 et § 4.1.2); c'est pourquoi il serait trop long de les reprendre ici.

4.2.2 Traitements relatifs aux commandes.

Nous nous intéressons seulement aux traitements qui n'ont aucune ressemblance aux traitements déjà évoqués dans le paragraphe 4.1.

4.2.2.1 _ Traitement du lien entre deux critères (LINK).

PAS 1: accès au premier critère (que l'on veut lier)

PAS 1.1: si ce critère existe, alors aller au PAS 2

PAS 1.2: sinon, imprimer M10
aller au PAS 1 de la phase 2

PAS 2: accès au second critère (que l'on veut lier)

PAS 2.1: si ce critère existe, aller au PAS 3

PAS 2.2: sinon: imprimer M10
aller au PAS 1 de la phase 2

PAS 3: création de la réalisation de l'objet complexe FILIATION (c.à.d. chargement du poids affecté à la relation, dans la base de données)

PAS 4: création des relations entre le premier critère et l'objet complexe FILIATION, de même qu'entre FILIATION et le second critère.

4.2.2.2 _ Traitement de suppression d'un lien, (UNLINK).

PAS 1: accès au premier critère cité

PAS 1.1: si ce critère existe, aller au PAS 2

PAS 1.2: sinon, imprimer M10
aller au PAS 1 de la phase 2

PAS 2: accès au second critère cité

PAS 2.1: si ce critère existe, aller au PAS 3

PAS 2.2: sinon, imprimer MIO
aller au PAS 1 de la phase 2

PAS 3: suppression de l'association entre ces 2 critères (objet complexe FILIATION)

PAS 4: suppression des relations entre le premier critère et l'objet complexe FILIATION, de même qu'entre le second critère et l'objet complexe FILIATION.

4.2.2.3 Traitement du passage complet (RESUME ALL) de la structure arborescente de la base de données générale à la base de données spécifique.(*)

PAS 1: accès à la racine du graphe de choix de la base de données spécifique (B.D.S.)

PAS 1.1: si elle n'existe pas: aller au PAS 2

PAS 1.2: sinon, imprimer M40

PAS 2: accès à la racine du graphe de choix de la base de données générale (B.D.G.)

PAS 2.1: si elle existe, aller au PAS 3

PAS 2.2: sinon, imprimer M41

PAS 3: créer la racine dans la B.D.S.

PAS 4: accès au critère suivant (par l'ordre de Tarry) dans la B.D.G.

PAS 4.1: s'il existe, aller au PAS 5

PAS 4.2: sinon, imprimer M60
retour au PAS 1 de la phase 2

PAS 5: création du même critère dans la B.D.S.
aller au PAS 4.

M40: B.D.S n'est pas vide

M41: B.D.G. est vide

M60: le passage de la structure arborescente est terminé.

(*) Cet ordre (RESUME) n'intervient pas au niveau du langage de manipulation, il représente l'interface entre la base de données générale et une base de données spécifique.

4.2.2.4 _ Traitement de création des "paramètres variables" (ENTRY).

Nous entendons par "paramètre variable", les libellés des projets et leurs évaluation ou fonction d'utilité (suivant la méthode de sélection Electre ou Cat) pour chaque critère élémentaire.

PAS 1: pour chaque projet, créer son libellé,

PAS 2: pour chaque critère élémentaire,

PAS 2.1: pour chaque projet,

PAS 2.1.1: introduction des valeurs d'entrée des fonctions d'utilité

PAS 2.1.1.1: si ces valeurs sont correctes, aller au PAS 2.1.2

PAS 2.1.1.2: sinon, imprimer M30 aller au PAS 2.1.1

PAS 2.1.2: création du libellé du projet et de la valeur d'entrée de sa fonction d'utilité

PAS 2.1.3: si le nombre de projets ≠ du nombre de projets introduits au PAS 1: imprimer M31

PAS 2.1.4: sinon, aller au PAS 2

PAS 3: imprimer M59

M30: la valeur des projets doit être numérique

M31: erreur: nombre de valeurs incorrect

M59: commande (ENTRY) terminée.

4.3 Description des traitements relatifs à la phase 3.

4.3.1 Interprétation des commandes

L'interprétation des commandes se fait également de manière similaire à ce que nous avons décrit précédemment (cfr 4.1.1 et § 4.1.2); c'est pourquoi nous ne les reprenons pas ici.

4.3.2 Traitements relatifs aux commandes

Ces traitements concernent la mise en forme des données en vue de l'exécution des programmes de choix. Nous allons donc envisager successivement les traitements relatifs à la mise en forme des données pour le programme Cat, et ensuite, pour le programme Electre.

4.3.2.1 Traitement de mise en forme pour le programme Cat.

PAS 1: créer le jeu de commandes et d'instructions nécessaires à l'exploitation du programme de choix Cat

PAS 2: pour chaque critère élémentaire,
créer les informations nécessaires (générées au travers de la base de données) définies dans l'état d'entrée correspondant à la description d'un critère non élémentaire

PAS 3: pour chaque critère non élémentaire,
créer les informations nécessaires définies dans l'état d'entrée et correspondant à la description d'un critère non élémentaire

PAS 4: pour chaque projet pris en compte,

PAS 4.1: introduire un titre correspondant au libellé du projet

PAS 4.2: pour chaque critère élémentaire, transférer la valeur d'entrée de la fonction d'utilité.

4.3.2.2 _ Traitement de mise en forme pour le programme Electre.

PAS 1: introduction des seuils de concordance au terminal

PAS 2: transfert du titre du problème (donné par défaut)
du nombre de projets
des libellés des projets

PAS 3: transfert des libellés des critères élémentaires, de même que
- le poids
- les seuils de discordance
- les seuils de concordance
- le facteur d'échelle, la condition sine qua non
- l'évaluation et les libellés des projets

PAS 4: créer les fichiers définis dans les états d'I/O du programme Electre.

CHAPITRE V METHODE DE PROGRAMMATION.

5.1 Modularité choisie.

Afin de réaliser la solution proposée dans le dossier de conception, nous devons suivre une découpe modulaire entre les différentes parties associées aux bases de données. Nous pouvons donc considérer qu'il existe trois grandes parties:

- une partie associée à la base de données générale
- une partie associée à la base de données Cat
- une partie associée à la base de données Electre.

Les parties associées à la création des bases de données spécifiques peuvent se décomposer en deux type de création:

- la création par l'intermédiaire d'un interface entre la base de données générale et la base de données spécifique
- la création par l'intermédiaire des commandes de manipulation.

Ces deux types de création ont été dissociés de manière à

- créer le deuxième type cité, indépendamment du premier type
- supprimer le premier type cité, indépendamment du second type.

Les phases 1 et 3 ainsi que le deuxième type de création de la phase 2 peuvent se décomposer en deux grandes fonctions:

- la première consiste en l'interprétation des commandes
- la seconde concerne l'exécution de ces commandes.

Le schéma de la figure 15 généralise la structure de la modularité des programmes travaillant sur les trois bases de données: Cat, Electre II et Générale (pour laquelle les modules 2 et 3 n'existent pas; cfr supra).

Nous reprenons ici une brève description de chacun de ces modules:

1. MAIN: contient les traitements de création, modification, impression, suppression des critères, de leurs paramètres et des relations entre ces critères
2. PRISE EN CHARGE DES PROJETS (phase 2): contient le traitement d'introduction des projets et des valeurs associées à ces projets.
3. MISE EN FORME DES DONNEES (phase 3): contient le traitement de mise en forme des données nécessaires à l'exécution des programmes de choix
4. ORIENTATION DE L'INTERPRETATION: contient le traitement d'interprétation du type de commande et d'orientation vers le module d'interprétation des commandes associées à ce type
5. INTERPRETATION: contient le traitement d'interprétation de l'entièreté (paramètres standards, verbes, identificateurs, paramètres spécifiques et leurs valeurs) des commandes de type détecté dans le module d'orientation.

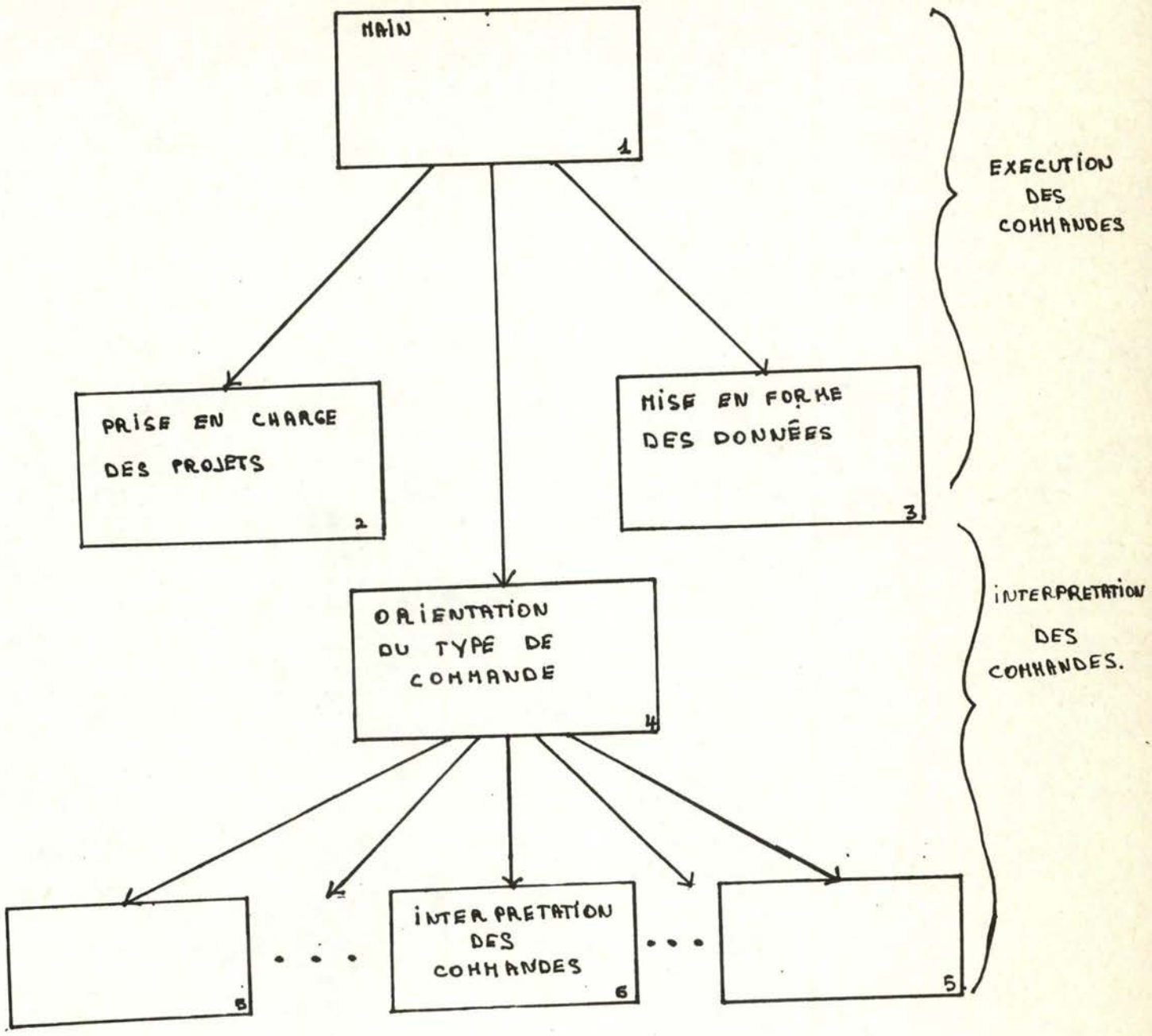


Fig. 15: schéma d'enchaînement des modules.

5.2 Justification de la modularité.

Le schéma idéal de la modularité pour notre application, aurait été le suivant (cfr fig. 16)

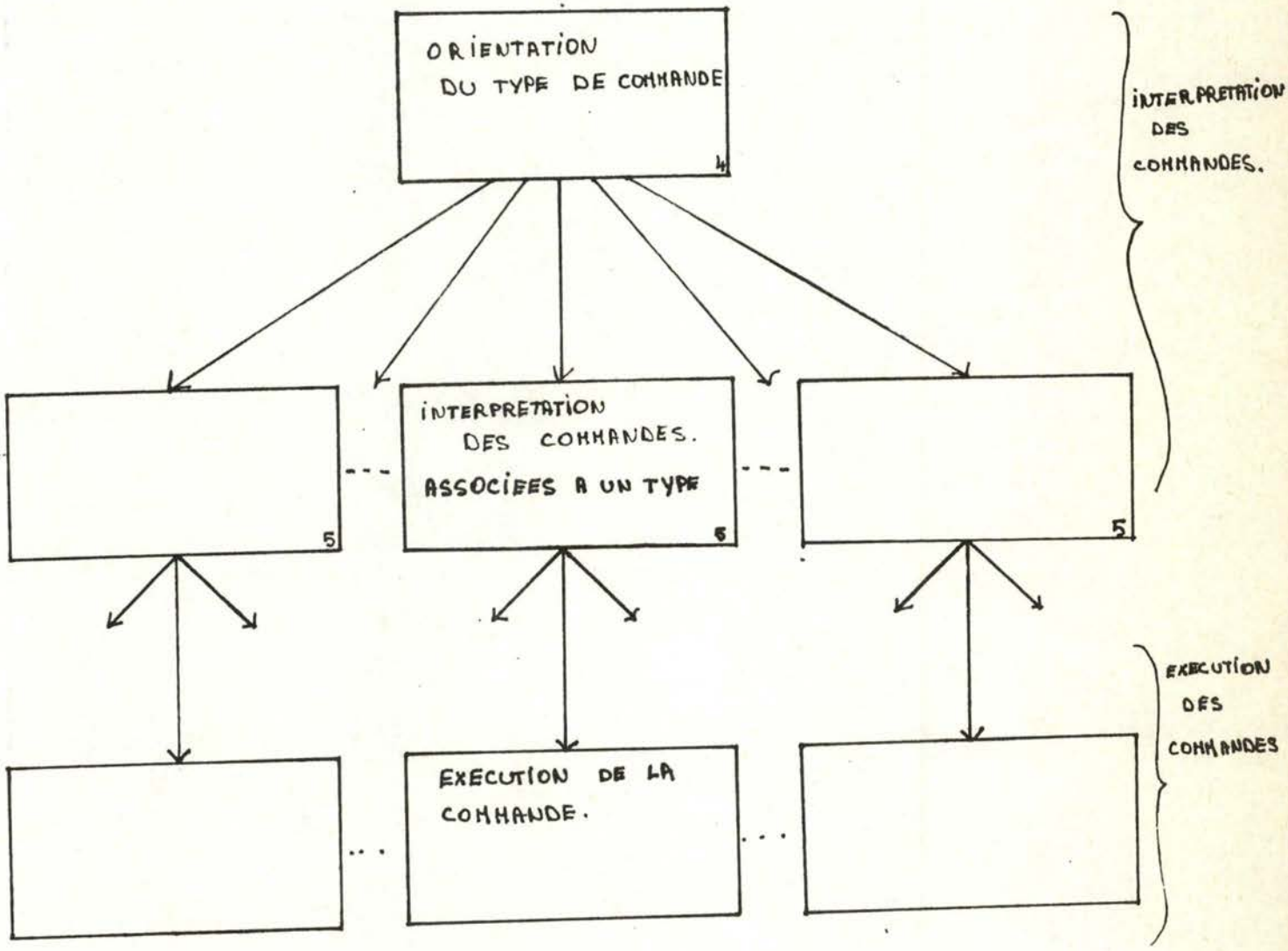


Fig. 16: schéma de représentation de la modularité idéale pour notre application.

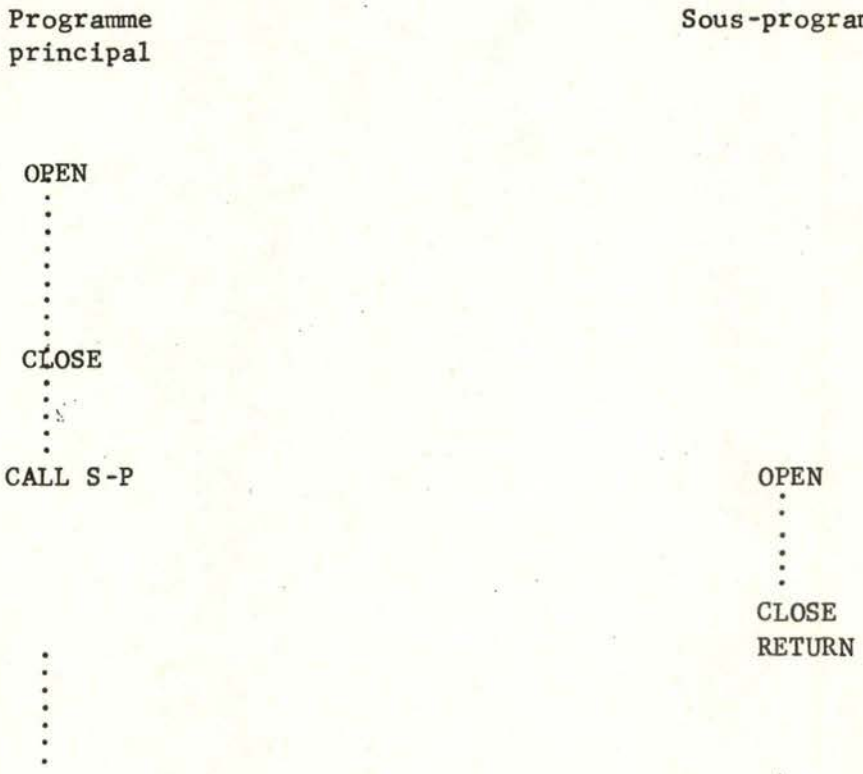
En effet, si on suit le schéma de la figure 16, on remarque que l'orientation du type de commande se fait à partir du premier module. Dès lors, il suffit de se brancher sur un module pour interpréter l'entièreté des commandes correspondant à ce type; et en fonction de cette interprétation complète, on débouche sur un module d'exécution d'une commande, ou d'un groupe de commandes, selon le degré de modularité souhaité.

Nous retrouvons ici de nombreux avantages de la modularité; il suffit de penser au fait que certains modules interviennent de manière identique, ou presque identique, dans les trois cas (Cat, Electre et général). De plus, le travail a été distribué entre 2 personnes; elles pouvaient donc implémenter chacune séparément des types de programmes bien déterminés. Nous n'allons pas énumérer ici tous les avantages de la programmation modulaire (cfr B17), remarquons simplement que ce schéma nous autorise une compatibilité et une autonomie maximum des modules.

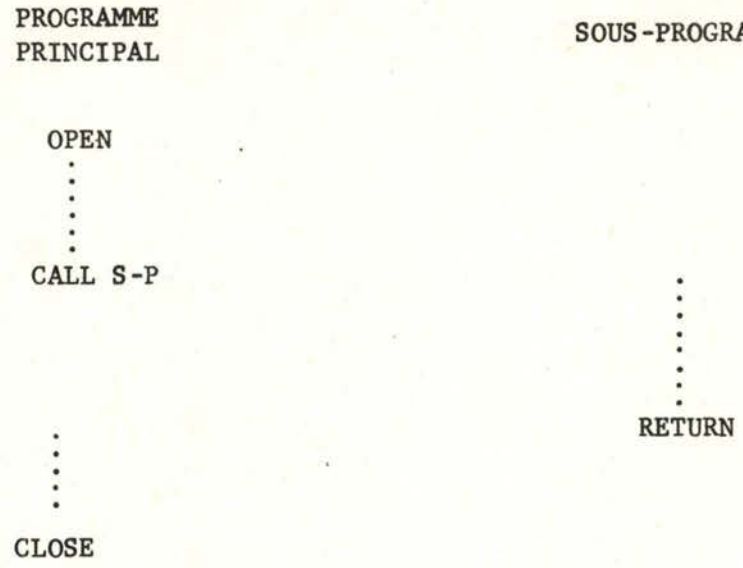
Cependant, tous les modules concernant l'exécution des commandes, sont des combinaisons de primitives de Sphinx et du langage Cobol; or, avant de pouvoir prendre en considération ces primitives, il est nécessaire d'ouvrir la base de données (éventuellement plusieurs bases de données) sur laquelle agissent ces primitives. De plus, Sphinx impose le fait que lors du passage d'un programme à un autre, toute base de données ouverte dans le premier doit être fermée dans celui-ci avant de passer à l'autre programme, où l'on peut évidemment réouvrir la base de données et la refermer.

De plus, l'ouverture et la fermeture de la base de données prend un temps considérable (pour notre problème). On comprend aisément, que pour un programme interactif, il est impensable de procéder à une ouverture, suivie d'une fermeture de la base de données pour l'exécution de chaque commande. Dès lors, une seule solution s'avérait intéressante: ouvrir la base de données dès le début de l'exécution du programme et la refermer à la fin d'exécution de celui-ci, tout en concentrant toutes les primitives de Sphinx dans le seul module principal.

Exemple correct de passage à un sous-programme contenant des primitives de Sphinx:



Exemple incorrect de passage à un sous-programme, contenant des primitives de Sphinx:



Il est bien évident, que dans ce dernier cas, si le sous-programme ne contient pas de primitives de Sphinx, le passage s'effectue correctement.

Ceci nous explique donc la raison pour laquelle, nous avons regroupé dans le module principal, les traitements d'exécution des commandes; ce programme principal appellant un sous-programme qui se charge de l'orientation du type de commande, passe la main à d'autre sous-programmes, pour l'interprétation complète des commandes associées à ce type; ceci étant terminé, ils repassent la main au programme principal, qui exécute le traitement détecté dans ces sous-programmes.

Il nous reste donc à expliquer l'éclatement de ce module principal en trois modules (1, 2 et 3). Ceci vient du fait, que chaque primitive de Sphinx s'exécute au moyen de sous-programmes, et le nombre de "CALL" dans un même programmme étant limité à 256, dans cette configuration, il nous était impossible de réaliser l'exécution de toutes ces commandes dans un seul module; ceci, principalement pour les cas d'Electre et de Cat. C'est pourquoi, nous avons décidé d'effectuer l'exécution de certaines commandes propres à ces deux méthodes dans un sous-programme. Etant donné le temps que nécessite l'ouverture et la fermeture d'une base de données, nous y avons donc inséré l'exécution des commandes qui interviennent le moins souvent ou qui forment un tout.

Nous nous expliquons: nous pouvons en effet considérer que l'insertion des libellés des projets, de même que leur valeur d'évaluation dans la base de données, se réalisent en une étape différente de la construction de l'arborescence; c'est pourquoi nous appelons souvent cette partie: partie "variable" du contenu de l'arborescence. Nous demandons par conséquent à l'utilisateur dans une première étape, de créer son arborescence avec ses paramètres "fixes", et dans une étape ultérieure, d'introduire les paramètres "variables".

D'autre part, nous pouvons également considérer le travail de mise en forme des données en vue de leur introduction dans les programmes de choix, comme une étape s'effectuant à un moment bien distinct des traitements précédents. C'est pourquoi ce traitement fait l'objet d'un module particulier.

En procédant de cette façon, nous déterminons trois modules (MAIN, MISE EN FORME et PRISE EN CHARGE DES PROJETS) caractérisés chacun par une ouverture et une fermeture de la base de données.

Exemple de technique de parcours de l'arborescence.

C'est par l'utilisation de piles que l'on a résolu le problème du parcours de l'arborescence pour son impression. Ce parcours s'effectue de la racine vers les feuilles et de la gauche vers la droite (cfr ordre de Tarry: B6)

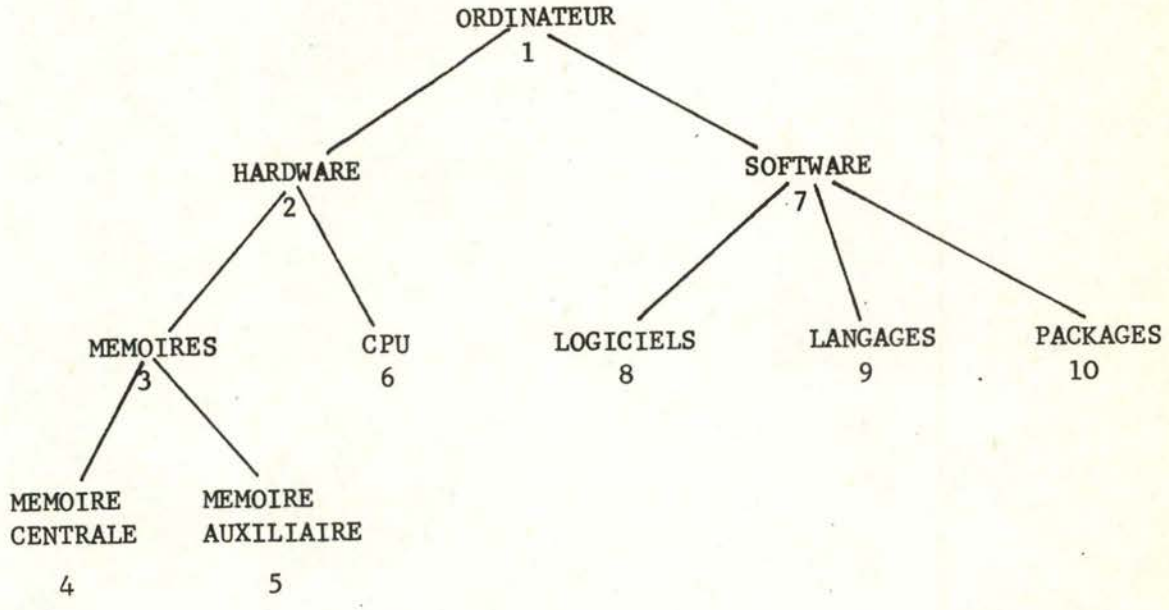
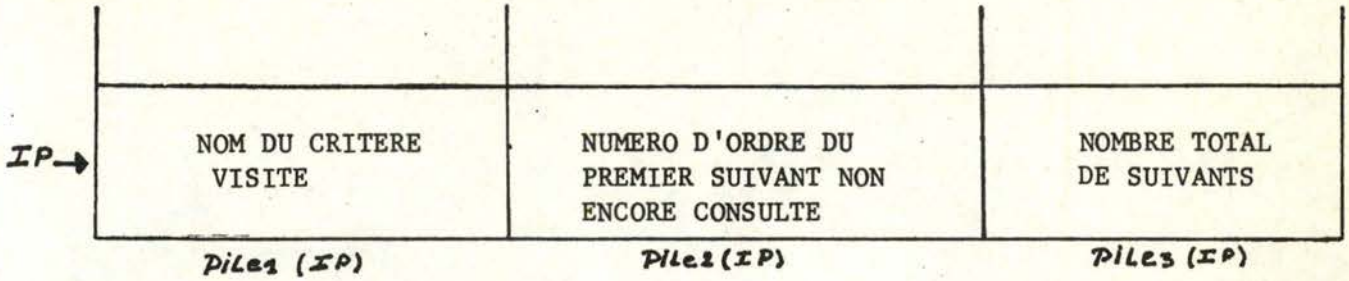


Fig. 17: l'arborescence est parcourue suivant les numéros associés aux critères.

Nous avons défini trois piles de manière suivante:



IP = indice permettant l'incrémentation de la pile.

Si nous reprenons l'exemple de la figure 17, nous obtenons l'évolution des piles décrites à la figure 18.

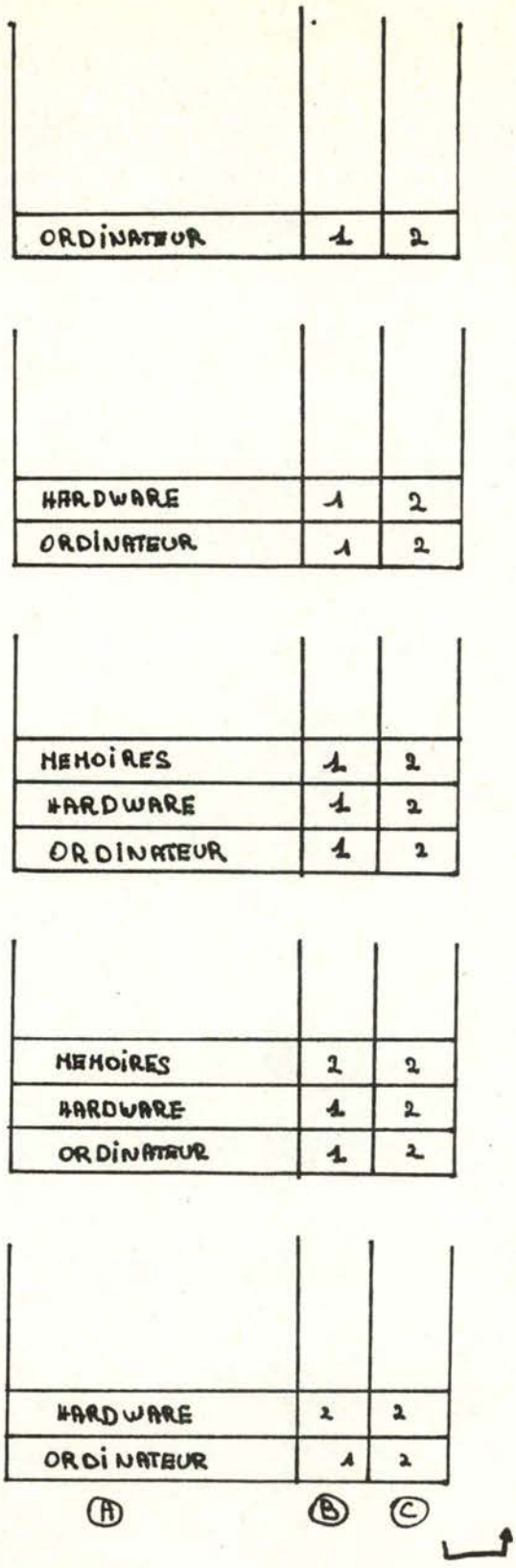
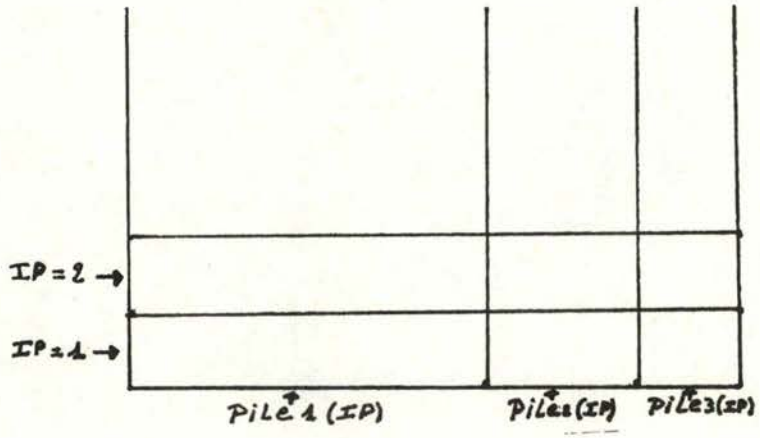


Fig. 18: évolution du contenu des piles.

Nous remarquons que pour l'impression, le traitement s'effectue lors de la première consultation du sommet.

A titre d'exemple, nous avons repris à la figure 19, l'organigramme correspondant à la commande d'impression de l'entièreté de l'arborescence, en employant les notations suivantes:



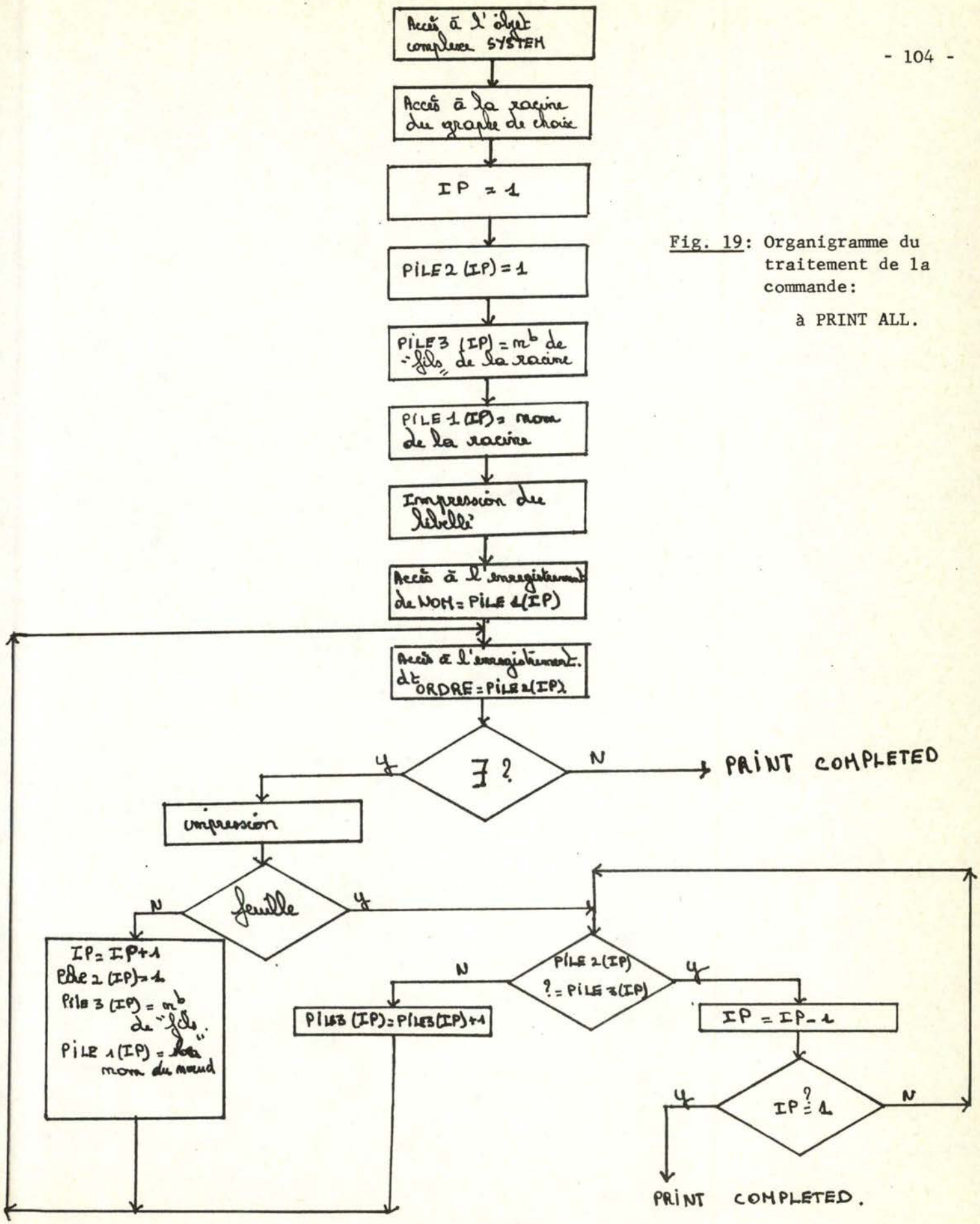


Fig. 19: Organigramme du traitement de la commande:

à PRINT ALL.

La résolution du problème de parcours de l'arborescence, par l'utilisation de piles, n'est pas la seule. Malgré la contrainte de non récursivité imposée par la structure de blocs au niveau de Sphinx, il était cependant possible de résoudre ce problème d'une autre manière. En effet, une solution aurait été d'imbriquer autant de blocs qu'il y avait de niveaux (après la racine) prévisibles dans l'arborescence (cfr fig. 20). Cette solution impliquait la contrainte du choix d'un nombre de niveaux maximum pour l'arborescence.

```
- ACCES A LA RACINE (A)
    - ACCES AU NIVEAU 1  (B, C, D)
        - ACCES AU NIVEAU 2  (E, F, G, H)
        - FIN ACCES
    - FIN ACCES
```

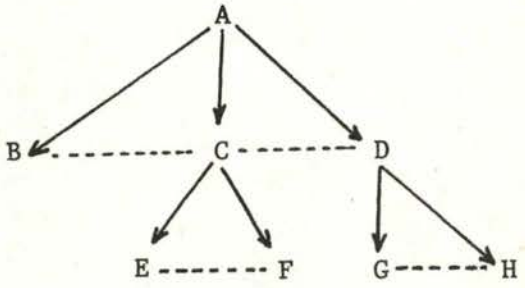


Fig. 20: Exemple de parcours de l'arborescence par une technique de blocs imbriqués.

Chaque bloc est une boucle implicite (cfr accès de masse) dont le nombre d'itérations est égal au nombre de critères du niveau. Le passage d'un bloc à un bloc plus imbriqué est caractérisé dans l'arborescence par le passage d'un niveau à un niveau inférieur (schématisé par les flèches dans la fig. 20). Le passage d'une itération d'un bloc à une autre itération de ce même bloc signifie le passage d'un critère à un autre critère de même niveau (schématisé par les pointillés). L'ordre de prise en charge (ce qui ne veut pas dire traitement) des critères est par exemple le suivant: A, B, C, E, F, D, G, H.

CHAPITRE VII LISTING DES PROCEDURES.

7.1 Utilisation de l'application.

Cfr Dossier Utilisateur.

7.2 Modification de l'application.

7.2.1 Compilation du schéma d'une base de données.

Les programmes sources correspondant au schéma des bases de données sont respectivement pour:

- la base de données générale, dans le fichier VBA.FOR.GEN.DDL (*)
- la base de données Cat, dans le fichier VBA.FOR.CAT.DDL
- la base de données Electre, dans le fichier VBA.FOR.ELE.DDL

La compilation d'un des schémas, après modification se réalise grâce à la procédure VBA.FOR.C.DDL

```
1:0000 /PROC C,(&N)
2:0000 /SYSFILE SYSDTA=&N
3.0000 /SYSFILE SYSLST=&N..DDL.LIST
4:0000 /FILE VBA.FOR.SEDI12,LINK=S12READ
5:0000 /FILE CDDL.SCHEMA,LINK=SCHWRIT
6:0000 /FILE CDDL.EXTERNAL.SCHEMA,LINK=SCHREAD
7:0000 /FILE CDDL.SOURCE,LINK=INPUT
8:0000 /FILE HIEFIL,LINK=HIEFIL
9:0000 /FILE CDDL.DATA,LINK=DTFORM
10:0000 /FILE CDDL.SEBE,LINK=SEBEOUT
11:0000 /EXEC VBA.FOR.DDLEXEC
12:0000 /SYSFILE SYSDTA=(SYSCMD)
13.0000 /SYSFILE SYSLST=(PRIMARY)
14:0000 /PRINT &N..DDL.LIST,SPACE=E,ERASE
15:0000 /ENDP
```

(*) Tous les noms de fichiers sont préfixés par VBA.FOR.
Les fichiers associés aux bases de données générale, Cat et Electre sont préfixés respectivement par VBA.FOR.GEN.
VBA.FOR.CAT.
VBA.FOR.ELE.

7.2.2 Chargement d'une base de données.

Ce chargement d'une base de données se réalise grâce à la procédure VBA.FOR.CHARGE

```
1.0000 /PROC  ,(&DBN)
2.0000 /SYSFILE SYSLST=&DBN..LIST
3.0000 /FILE &DBN..O
4.0000 /FILE &DBN..Z
5.0000 /SYSFILE SYSDTA=&DBN..SEDI12
6.0000 /EXEC VBA.FOR.PRO1
7.0000 /SYSFILE SYSDTA=&DBN..SEFO
8.0000 /EXEC VBA.FOR.PRO2
9.0000 /SYSFILE SYSDTA=&DBN..SEBE
10.0000 /EXEC VBA.FOR.PRO3
11.0000 /SYSFILE SYSLST=(PRIMARY)
12.0000 /SYSFILE SYSDTA=(SYSCMD)
13.0000 /PRINT &DBN..LIST,SPACE=E,ERASE
14.0000 /ENDP
```

7.2.3 Compilation d'un programme associé à l'interprétation des commandes.

Nous sous-entendons ici le fait que ce programme ne contient aucune primitive de Sphinx. Les fichiers associés à ces programmes sources se préfixent pour:

- la base de données générale par VBA.FOR.GEN.D.
- la base de données Cat par VBA.FOR.CAT.D.
- la base de données Electre par VBA.FOR.ELE.D.

La compilation de ces programmes, après modification, se réalise grâce à la procédure VBA.FOR.PROC.COMP

```
1.0000 /PROC C,&FILE
2.0000 /PARAM LIST=YES,ERRFIL=YES,DEBUG=YES
3.0000 /SYSFILE SYSDTA=VBA.FOR.&FILE
4.0000 /EXEC VBA.FOR.ANSICOB
5.0000 /SYSFILE SYSLST=(PRIMARY)
6.0000 /ENDP
```

Si la compilation est correcte, la procédure utilisée pour l'introduction du programme objet en bibliothèque est VBA.FOR.PROC.LMR

```
1.0000 /PROC C,&FILE,&FILE1
2.0000 /SYSFILE SYSDTA=(SYSCMD)
3.0000 /EXEC LMR
4.0000   CONTROL OUTFILE=(VBA.FOR.&FILE,OLDLIB),LISTING=(BOTH,SYSOUT)
5.0000   COPYALL SOURCE=*
6.0000   END
7.0000 /ER *
8.0000 /ER &FILE1.ERRFIL
9.0000 /ENDP
```

7.2.4 Compilation d'un programme associé à l'exécution des commandes.

Nous sous-entendons qu'un programme associé à l'exécution des commandes contienne des primitives de Sphinx. La compilation d'un tel programme se réalise grâce à la procédure VBA.FOR.C.DML

```
1.0000 /PROC N,(&SOURCE)
2.0000 /ERASE &SOURCE..
3.0000 /STEP
4.0000 /FILE &SOURCE..WORK.1,LINK=WORK1
5.0000 /FILE &SOURCE..WORK.2,LINK=WORK2
6.0000 /FILE &SOURCE..WORK.3,LINK=WORK3
7.0000 /FILE &SOURCE..GEN,LINK=GENFILE
8.0000 /FILE &SOURCE..DIAG,LINK=FDIAG
9.0000 /FILE CDDL.EXTERNAL.SCHEMA,LINK=SCHEMA-S
10.0000 /SYSFILE SYSLST=&SOURCE.-LIST
11.0000 /SYSFILE SYSIPT=&SOURCE
12.0000 /EXEC VBA.FOR.DML1
13.0000 /PARAM ERRFIL=YES,LIST=YES,SYMDIC=YES
14.0000 /PARAM DEBUG=YES
15.0000 /FILE &SOURCE..ERRFIL,LINK=ERRFIL
16.0000 /SYSFILE SYSDTA=&SOURCE..GEN
17.0000 /EXEC VBA.FOR.ANSICOB
18.0000 /STEP
19.0000 /SYSFILE SYSDTA=(SYSCMD)
20.0000 /FILE &SOURCE..ERRFIL,LINK=ERRFIL
21.0000 /FILE &SOURCE..GEN,LINK=ISOURCE
22.0000 /FILE &SOURCE..DIAG,LINK=FDIAG,OPEN=EXTEND
23.0000 /EXEC VBA.FOR.DML2
24.0000 /STEP
25.0000 /SYSFILE SYSLST=(PRIMARY)
26.0000 /PRINT &SOURCE.-LIST,SPACE=E,ERASE
27.0000 /EXEC LMR
28.0000 CONTROL OUTFILE=(TEST.LIB.OBJECT,NEWLIB),LISTING=(BOTH,SYSOUT)
29.0000 COPYALL SOURCE=*
30.0000 END
31.0000 /STEP
32.0000 /ERASE &SOURCE..WORK.
33.0000 /ERASE &SOURCE..GEN
34.0000 /ERASE *
35.0000 /ENDP
```

Les fichiers associés aux programmes source sont:

pour la base de données générale	VBA.FOR.GEN.DML	(LIBDML)
pour la base de données Cat	VBA.FOR.CAT.FICDML	(LIBDML)
	VBA.FOR.CAT.MEF	(LIBMEF)
	VBA.FOR.CAT.ENTRY	(LIBENTRY)
pour la base de données Electre	VBA.FOR.ELE.DML	(LIBDML)
	VBA.FOR.ELE.ENTRY	(LIBENTRY)
	VBA.FOR.ELE.MEF	(LIBMEF)
pour la base de données générale et la base de données Cat		
	VBA.FOR.CAT.INTER	(LIBINTER)
pour la base de données générale et la base de données Electre		
	VBA.FOR.ELE.INTER	(LIBENTER)

Si la compilation est correcte, on copie le fichier TEST.LIB.OBJECT dans la bibliothèque correspondante (dont le suffixe est cité entre parenthèses, en face des fichiers associés aux programmes source.

Exemple: COPY TEST.LIB.OBJECT,VBA.FOR.GEN.LIBDML

7.2.5 Création du module de chargement.

La création d'un "load module" se réalise grâce aux procédures respectivement pour la base de données générale, la base de données Cat, la base de données Electre:

VBA.FOR.GEN.LNK

```
1:0000 /PROC C
2:0000 /SYSFILE SYSDTA=(SYSCMD)
3:0000 /EXEC TSOSLNK
4:0000  PROGRAM VFOR,FILENAM=VBA.FOR.GEN.LOAD
5:0000  INCLUDE DMLCOM,VBA.FOR.GEN.LIBDML
6:0000  RESOLVE ,VBA.FOR.LIBO
7:0000  RESOLVE ,VBA.FOR.GEN.LIBGEN
8:0000  END
9:0000 /ENDP
```

VBA.FOR.CAT.LNK

```
1.0000 /PROC C
2.0000 /SYSFILE SYSDTA=(SYSCMD)
3.0000 /EXEC TSOSLNK
4.0000  PROGRAM VFOR,FILENAM=VBA.FOR.CAT.LOAD
5.0000  INCLUDE CATDML,FILENAM=VBA.FOR.CAT.LIBDML
6.0000  RESOLVE ,VBA.FOR.LIBO
7.0000  RESOLVE ,VBA.FOR.CAT.LIBGEN
8.0000  RESOLVE ,VBA.FOR.CAT.LIBENTRY
9.0000  RESOLVE ,VBA.FOR.CAT.LIBMEF
10.0000 END
11.0000 /ENDP
```

VBA.FOR.ELE.LNK

```
1.0000 /PROC C
2.0000 /SYSFILE SYSDTA=(SYSCMD)
3.0000 /EXEC TSOSLNK
4.0000  PROGRAM VFOR,FILENAM=VBA.FOR.ELE.LOAD
5.0000  INCLUDE ELEESS,VBA.FOR.ELE.LIBDML
6.0000  RESOLVE ,VBA.FOR.LIBO
7.0000  RESOLVE ,VBA.FOR.ELE.LIBGEN
8.0000  RESOLVE ,VBA.FOR.ELE.LIBENTRY
9.0000  RESOLVE ,VBA.FOR.ELE.LIBMEF
10.0000 END
11.0000 /ENDP
```

TROISIEME PARTIE DOSSIER UTILISATEUR

CHAPITRE I SPECIFICATIONS DU PROBLEME.

L'institut d'informatique (Facultés N-D de la Paix, Namur) dispose de deux programmes de choix correspondant à deux méthodes de sélection d'objets complexes: Cat et Electre II, qui se basent sur une arborescence (*1) des critères de choix.

L'outil élaboré tout au long de ce travail permet à ce niveau

- l'introduction (*2) interactive des données d'entrée relatives aux programmes de choix
- la vérification de ces données lors de leur introduction.

Cet outil engendre un certain nombre d'avantages que nous ne rappelons pas ici (cfr Dossier de Conception).

Rappels.

Suivant leur destination, les données peuvent être introduites dans trois bases de données différentes qui sont:

- une base de données générale qui contient la structure arborescente des critères
- une base de données Cat contenant la hiérarchisation des critères et les paramètres qui leur sont propres (fonction d'utilité, poids, opérateur d'aggrégation...)

(*1) Par abus de langage, nous employons le terme arborescence pour définir la structure hiérarchique des critères de choix intervenant pour la méthode Cat.

(*2) Introduire = créer et mettre en forme (au sens large)

- une base de données Electre contenant une structure arborescente et ses paramètres spécifiques (seuils de discordance, poids ...).

Chaque base de données peut être chargée par l'intermédiaire d'un langage de manipulation (*) qui lui est propre. De plus, les bases de données Cat et Electre II peuvent reprendre la structure arborescente des critères créée dans la base de données générale, par le biais d'un interface.

De plus, un fichier "mis en forme" peut être créé à partir des bases de données spécifiques par l'intermédiaire d'un interface, de manière à organiser les informations pour que celles-ci soient exploitables par les programmes de choix. Etant donné que cet interface manipule les bases de données spécifiques et est nécessaire (pour notre application) pour l'exécution des programmes de choix, on associe à cet interface, une commande de manipulation au même titre qu'une modification des bases de données.

(*) Le langage de manipulation est constitué d'un ensemble de commandes qui permettent de définir des manipulations de données (introduction, modification, affichage des données ...). Ce langage de manipulation comprend:

- le langage de manipulation général
- le langage de manipulation Cat
- le langage de manipulation Electre.

Ces trois derniers langages de manipulation sont composés de commandes se rapportant respectivement aux bases de données générale, Cat et Electre.

Des programmes de manipulation (*) sont créés en utilisant un système de gestion de bases de données (Sphinx) qui lui-même s'appuye sur le système de gestion des fichiers du Siemens 4004-151 (Sesam).

Il faut enfin préciser que, avant l'exécution de n'importe quel programme de manipulation, les systèmes Sesam et Sphinx doivent être chargés.

(*) Le programme de manipulation permet l'introduction, la modification ainsi que l'affichage des informations relatives aux différentes bases de données, à partir des commandes du langage de manipulation, et ceci, de manière interactive. Ce programme de manipulation se compose

- du programme de manipulation général
- du programme de manipulation Cat
- du programme de manipulation Electre.

Ces trois programmes de manipulation manipulent respectivement les bases de données générale, Cat et Electre.

CHAPITRE II MODE D'EMPLOI.

2.1 Chargement et déchargement des systèmes Sphinx et Sesam.

Les procédures de chargement et de déchargement de ces deux systèmes doivent s'effectuer dans un ordre bien précis:

a. chargement de Sesam
b. chargement de Sphinx
c. exécution d'un programme de manipulation

d. fin d'exécution du programme
e. déchargement de Sphinx
f. déchargement de Sesam.

Deux terminaux sont nécessaires:

- le premier pour charger Sesam
- le second pour charger Sphinx et utiliser nos programmes de manipulation.

Nous réservons un terminal uniquement pour Sesam, parce que celui-ci envoie des messages utiles à l'utilisateur (ouverture et fermeture des bases de données).

2.1.1 Chargement de Sesam.

Le chargement de Sesam (cfr fig. 31) se fait de manière interactive par la commande:

/ EXEC VBA.FOR.PROSAM

Le système Sesam demande ensuite à l'utilisateur de lui fournir des options; dans notre cas: 20b,20d, suffisent (b correspond aux nombre d'utilisateurs et d au nombre de bases de données).

2.1.2 Déchargement de Sesam.

Le déchargement du système Sesam s'effectue comme suit (cfr fig. 31):

- appui sur la touche "ESCAPE" du terminal par l'utilisateur
- la "main" lui est alors renvoyée (ce qui est symbolisé au terminal par le caractère "/")
- l'utilisateur précise ensuite:

  INTR
- le système lui demande:

  INT = ...
- l'utilisateur répond:

  STOP

```
%C E222 PLEASE LOGON.
Feeder#####Beschreibung#####################################
%C E223 LOGON ACCEPTED FROM LINE #028 AT 1317 ON 05/05/78, TSN 7074
%TIMESHARING SERVICE WILL BE AVAILABLE TILL  PM
%ON APRIL  2  AND 26 , MAY   3.
%CONTINUE?  (Y,N)  n
/exec vba.for.prosam
 %C P 00 * LOADING
   SESAM     BS2-VERS  1.0 TSN:7074
 1. OPTION #I,#Z,#P,#Q, B,#D,#S, T,RE,UZ,PC,PK,R,O,
*20b,20d,
 MASTER-DB*****************
 DB-NAME: GEN
 DB - # : 0B
 ** OPEN COMMUNICATION **  BS -VERS. 1.0 ;  07/01/76
 ** SESAM  ** NUR BEENDEN MIT / INTR 7074 ....STOP
 --OPEN--- 0001=TSN  NR: 1 BA  0B
 --OPEN--- 0001=TSN  NR: 1 BB  0B
 --OPEN--- 0001=TSN  NR: 1 BC  0B
  DB : GEN                  CLOSE NACH    41 DIREKTAENDERUNGEN
 --CLOSE-- 0001=TSN  NR: 1 BC  0B
 --CLOSE-- 0001=TSN  NR: 1 BB  0B
 --CLOSE-- 0001=TSN  NR: 1  *  0B

/intr
 INT = ....stop
 INT = STOP
 INT : STOP
 ** CLOSE COMMUNICATION - SESAM  **  0+ %BKPT  Z.ZT.1
%BKPT PCOUNT 003800
```

Fig. 31: exemple de chargement et de déchargement "régulier" de Sesam.

Si par hasard, l'utilisateur se trouvait confronté à une erreur dans un de nos programmes de manipulation, alors qu'une ou plusieurs bases de données sont encore "OPEN", il devra, afin de garder l'intégrité de sa (ou ses) base(s) de données, et avant de pratiquer au déchargement normal de Sesam, exécuter la fermeture de la dernière base de données ouverte, et ainsi de suite jusqu'à ce qu'elles soient toutes fermées.

Il suffira dans ce cas d'écrire (cfr fig. 32):

CLOSE n° de base de données spécifié à l'ouverture de celle-ci.

Dans la suite, nous appellerons le premier cas, déchargement "normal" de Sesam, et le second "déchargement irrégulier".

```
%C E[illegible] PLEASE LOGON.
[illegible]
%C E233 LOGON ACCEPTED FROM LINE #027 AT 1703 ON 03/23/78, TSN 1862
%TIMESHARING SERVICE WILL BE AVAILABLE TILL  PM
%ON MARCH  22 AND 29 , APRIL  .
%CONTINUE? (Y,N)
/exec vba.for.prosam
 %C P500 * LOADING

   SESAM     BS2-VERS  1.0 TSN:1862
 1. OPTION #I,#Z,#P,#Q,  B,#D,#S,  T,RE,UZ,PC,PK,R,O,
*20b,20d,
 MASTER-DB*****************
 DB-NAME: GEN
 DB - # : 0A
 ** OPEN COMMUNICATION **  BS2-VERS. 1.0 ;  07/01/76
 ** SESAM  ** NUR BEENDEN MIT /INTR 1862 ....STOP
 --OPEN--- 0001=TSN  NR: 1 AA  0A
 --OPEN--- 0001=TSN  NR: 1 AB  0A
 --OPEN--- 0001=TSN  NR: 1 AC  0A
/intr
 INT = ....close 0a
 INT = CLOSE 0A
 INT : REJ.
/intr
 INT = ....stop
 INT = STOP
 INT : STOP
  DB : GEN                    CLOSE NACH     8 DIREKTAENDERUNGEN
 MAX.RRA ORG =      151 , ZDR =         1
 DB-NAME: GEN                    : CLOSED
 ** CLOSE COMMUNICATION - SESAM  **  o  %BKPT  7.ZT.1
%BKPT PCOUNT 003000
/Logoff
%C E4 0 LOGOFF AT 1710 ON 03/23/78, FOR TSN 1862.
%C E4 1 CPU TIME USED:  000005.1210 SECONDS.
```

Fig. 32: exemple de fermeture "irrégulière" de Sesam.

2.1.3 Chargement du système Sphinx.

Le chargement du système Sphinx doit s'effectuer à partir d'un autre terminal que celui à partir duquel on charge le système Sesam; il peut se réaliser de deux manières différentes:

- soit en interactif: cela a pour conséquence de recourir à un troisième terminal pour exécuter un programme de manipulation
- soit un lancement batch à partir d'un deuxième terminal.

La première méthode se réalise à partir de la commande (cfr fig. 33):

```
/EXEC VBA.FOR.L.SPHINXV
```

La seconde méthode s'effectue par la commande (cfr fig. 34):

```
/ENTER VBA.FOR.ENTER.SPHINXV,PRIORITY=6
```

Nous lui avons associé une priorité maximum de manière à ce que ce "job" soit rapidement chargé. On aura cependant la précaution de vérifier qu'il est bien pris en charge par le système avant d'exécuter un programme: on utilise à cette fin la commande:

```
/STA n° de tâche
```

Ce n° de tâche est celui que le système lui a accordé et il nous est communiqué de suite après l'ordre ENTER.

```
/exec vba.for.l.sphinxv,time=50
%C P500 * LOADING

SYSTEM  SPHINX V  LOADED - 13:43:52

PLEASE FASTEN YOUR SEAT BELT AND RELAX ...


/logoff
```

Fig. 33: chargement interactif du système Sphinx et déchargement.

```
%C E222 PLEASE LOGON:
##########################################################################
%C E223 LOGON ACCEPTED FROM LINE #024 AT 1320 ON 05/05/78, TSN 7082
%TIMESHARING SERVICE WILL BE AVAILABLE TILL 6PM
%ON APRIL 12  AND 26 , MAY   3.
%CONTINUE?  (Y,N)  n
/enter vba.for.enter.sphinxv,priority=6,time=60
 %C E172 * TSN=7085
/sta 7085
%   TYPE  SPOOLIN LOGON   LOGOFF  CPU-TIME
%     2     1321   1321    0000  000000.1178
%   TYPE  PRI  CURR-CMD  PROG     SIZE  CLASS  TSN
%     2   06   EXEC                          7085
%   ITN VIR-ADDR PEND QUE#
%   24   22B698  03  004

/can 7085
```

Fig. 34: chargement batch à partir d'un terminal du système Sphinx et déchargement.

2.1.4 Déchargement du système Sphinx.

Si le chargement de Sphinx est réalisé au moyen de la commande /EXEC , il suffit d'appuyer sur la touche "ESCAPE" du terminal pour se retrouver en mode "/" (cfr fig. 33).

Par contre, si Sphinx a été lancé en batch, il faudra utiliser la commande (cfr fig. 34):

/CANCEL n° de tâche

2.2 Exécution des programmes de manipulation.

2.2.1 Programme de manipulation général.

L'exécution du programme de manipulation général s'effectue par la commande:

```
/EXEC VBA.FOR.GEN.LOAD
```

ceci a pour effet d'ouvrir la base de données générale et de renvoyer la main à l'utilisateur (symbolisé par le caractère "*").

2.2.2 Programme de manipulation Cat.

L'exécution du programme de manipulation Cat s'effectue par la commande:

```
/EXEC VBA.FOR.CAT.LOAD
```

lorsque l'on exécute la phase de mise en forme des données (nécessaire pour introduire les données dans le programme de choix), cette commande doit être précédée des commandes suivantes:

```
/FILE VBA.FOR.CAT.FICHMEF,LINK=FICHMEF
/FILE VBA.FOR.CAT.FICHPROG,LINK=FICHPROG
```

2.2.3 Programme de manipulation Electre.

L'exécution du programme de manipulation Electre II s'effectue par la commande:

```
/EXEC VBA.FOR.ELE.LOAD
```

Si l'on effectue par la suite, la phase de mise en forme des données (cfr commande à WRITE du langage de manipulation Electre), la commande (EXEC) doit être précédée des commandes suivantes:

```
/FILE VBA.FOR.ELE.FICHP1,LINK=FICHP1
/FILE VBA.FOR.ELE.FICHP2,LINK=FICHP2
/FILE VBA.FOR.ELE.FICHP3,LINK=FICHP3
```

2.3 Exécution de l'interface entre la base de données générale et les bases de données spécifiques.

Pour réaliser l'interface entre la base de données générale et la base de données Cat, il suffit d'utiliser la commande:

```
/EXEC VBA.FOR.CAT.LOAINTER
```

Pour réaliser l'interface entre la base de données générale et la base de données Electre, il suffit d'utiliser la commande:

```
/EXEC VBA.FOR.ELE.LOAINTER
```

2.4 Exécution des programmes de choix.

L'exécution du programme Cat se fait à partir de la commande suivante:

```
/DO VBA.ARN.C.SEL,(SYSDTA,SYSLST)
```

L'exécution du programme Electre II s'effectue à partir de la commande:

```
/DO VBA.FOR.ELE.EXEC
```

2.5 Sauvetage et récupération d'une base de données (générale, Cat et Electre).

A. Sauvetage.

```
/EXEC PRO.SEBE
*ZD10 NAME='nom de la base de données',SAVE
*end
```

B. Récupération.

```
/EXEC PRO.SEBE
*zd10 name='nom de la base de données',regen=9999
*end
```

où 9999 représente un nombre de quatres chiffres, renvoyé lors du sauvetage de la même base de données par le système.

CHAPITRE III REGLES GENERALES DU LANGAGE DE MANIPULATION.

De manière à exprimer simplement la syntaxe des commandes que nous avons créées, nous utilisons les conventions suivantes:

- nous notons en lettres majuscules les éléments faisant partie du langage
- les expressions encadrées de [et] sont facultatives
- une liste d'expressions encadrées de { et } indique que l'on doit choisir un et un seul élément de la liste
- tout symbole souligné est obligatoire
- la notation [b]$_i$ indique la possibilité d'écrire un nombre quelconque d'espaces (y compris 0)
- le symbole ... signifie que la ligne précédente peut être répétée un certain nombre de fois.

3.1 Elements du langage:

- Symboles: les lettres de notre alphabet (A à Z), les chiffres décimaux (0 à 9) et les symboles à , * "

-Mots:

- verbes (1): PRINT CREATE MODIFY SUPPRESS INSERT DELETE ALTER HALT ENTRY WRITE LINK UNLINK
- paramètres standards (2): FIRST NODE LEAF BY ALL END NAME PARAM TO FROM
- paramètres spécifiques (3): WEIGHT AGGREGAT FUNCTION SD1 SD2 SCALE SQN

- Identificateurs de critères (4)
- Valeurs associées aux paramètres spécifiques
- Valeurs associées au paramètre standard: niveau.

(1): le verbe représente le type de fonction de la commande (p.ex: création).

(2): les paramètres standards donnent une notion plus précise de la fonction de la commande (ex: création d'une feuille).

(3): les paramètres spécifiques permettent d'identifier les caractéristiques des critères suivant le type de base de données qu'on manipule (Cat, Electre II ou générale); (p.ex: création du poids associé à une feuille).

(4): par abus de langage, nous ne faisons aucune distinction entre les termes suivants: identificateur de critère, libellé de critère et :nom de critère.

3.2 Règles de séparation des éléments du langage.

- Tout identificateur de critère de choix sera précédé et suivi du symbole " .
- Deux valeurs numériques successives doivent être séparées par au moins un espace.
- Toute commande commence par le symbole "à" qui peut être précédé et suivi d'un nombre quelconque de blancs.
- Tous les mots, identificateurs, valeurs associées aux paramètres et symboles à , * peuvent être séparés d'un nombre quelconque de blancs.
- La longueur de chaque commande ne peut en aucun cas dépasser 70 caractères, sauf si on la poursuit à la ligne suivante (en utilisant le symbole *)
- Le symbole * permet la continuation d'une commande à la ligne suivante; cependant, ce symbole ne se place qu'avant un paramètre spécifique.
- Un identificateur de critère ne peut dépasser 27 caractères.

3.3 Classes de commandes.

On peut regrouper les commandes en différentes classes (*):

1. création:

à CREATE: création des critères et de ses paramètres

à ENTRY: création des projets et de leurs évaluations

à LINK: création d'un lien entre deux critères

2. modification:

à MODIFY: modification du libellé des critères ou de ses paramètres

à ALTER: changement du n° de version (dans la base de données générale) ou changement d'un noeud en feuille et vise-versa (dans les bases de données spécifiques)

à SUPPRESS: suppression de critère(s)

à UNLINK: suppression de relation entre deux critères

3. impression et mise en forme:

à PRINT: impression du contenu de la base de données

à WRITE: mise en forme des données en correspondance avec les inputs des programmes de choix

4. clôture:

à HALT: fermeture de la base de données et fin du programme.

(*) On ne reprend ici que la liste des verbes correspondant au type de commande. Chaque commande est détaillée par la suite.

CHAPITRE IV COMMANDES DU LANGAGE DE MANIPULATION GENERAL.

4.1 Type de commande: à CREATE.

A. Syntaxe.

```
[b]; à [b]; C[REATE] [b]; { F[IRST] [b]; "identificateur"
                          { N[ODE] [b]; "id1" [b]; "id2"
                          { B[RANCH] [b]; "id1"
                          {     [b]; niveau [b]; "id2"
                          {   [ ...                 ]
                          {     [b]; END
```

B. Sémantique, et fonctionnement.

Le but de ce type de commande est de créer la structure arborescente des critères (c.à.d. uniquement les libellés). Elle est nécessaire avant toute manipulation ultérieure. Le n° de version associé à un critère est créé implicitement lors de la création de ce critère (cfr type de commande: ALTER).

Nous devons distinguer la racine du graphe d'un autre sommet, étant donné qu'elle n'a aucun précédent. Nous associons à la création de cette racine ("identificateur") une commande caractérisée par le paramètre standard FIRST.

D'autre part, lors de la création d'un critère noeud ou feuille ("id1"), il est nécessaire de spécifier son précédent ("id2"). Nous associons à la création d'un noeud ou d'une feuille une seule commande caractérisée par le paramètre standard NODE.

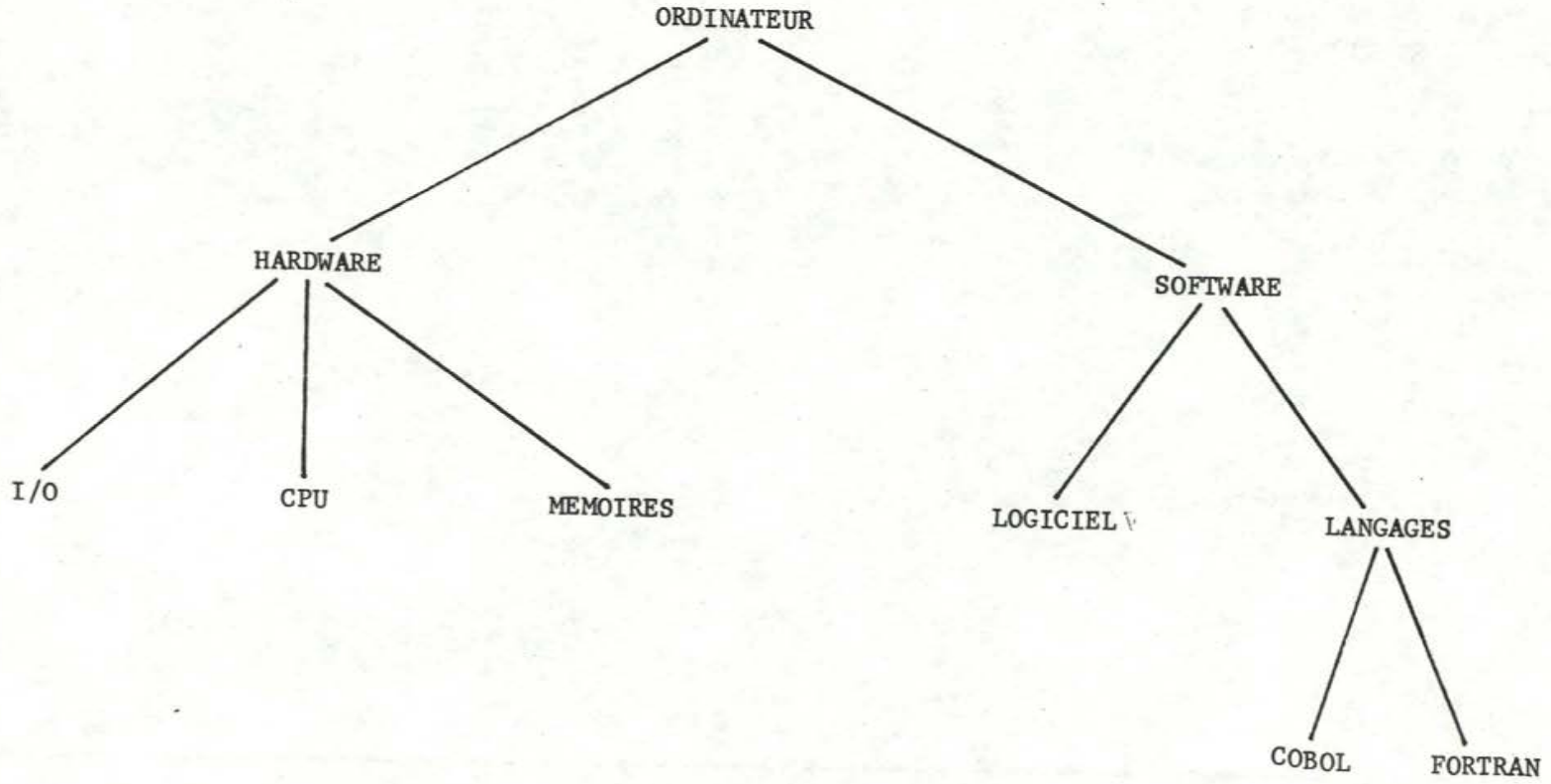


Fig. 35: Exemple de structure arborescente.

Nous avons également insisté sur l'importance de pouvoir créer une arborescence branche par branche; nous allons donc associer le paramètre BRANCH à une telle commande. On se rappellera aussi qu'il doit exister au moins un critère dans la base de données, comme lors de la création d'un noeud ou d'une feuille, avant de pouvoir lui rattacher des suivants.

Dans la commande CREATE BRANCH, l'utilisateur indique le critère auquel il veut rattacher une branche (et le considère comme faisant partie du niveau zéro). Il indique ensuite les éléments faisant partie des niveaux suivants dans sa branche selon la représentation de la structure d'un article Cobol; cette commande se termine enfin par l'ordre END.

La création de plusieurs critères au niveau 01 est en fait la création d'une forêt (c.à.d. plusieurs branches). Par abus de langage, nous parlerons de branche dans tous les cas.

C. Exemples.

Nous désirons créer la structure arborescente représentée à la figure 35. Nous exprimons une façon de procéder:

- la première opération consiste donc à créer la racine du graphe de choix: cfr C1; en effet, si on désire imprimer le contenu de la base de données avant la création de la racine, le système répondra qu'elle est "vide"
- la méthode la plus simple et la plus rapide consiste à créer l'arborescence au moyen de la commande à CREATE BRANCH (cfr C2)

- si ensuite, nous remarquons que nous avons oublié de créer un critère, nous pouvons utiliser la commande à CREATE NODE, cfr C3.

4.2 Type de commande: à PRINT.

A. Syntaxe.

[b]. à [b]. P[RINT] [b]. { A[LL] [b].[niveau] | "id" [b].[niveau] }

B. Sémantique et fonctionnement.

Le but de ce type de commande est d'obtenir un affichage complet ou partiel de l'arborescence au terminal, suivant que l'on désire des informations sur

- un critère
- une branche (tronquée ou non)
- l'arbre (tronqué ou non)

Nous entendons ici le terme "tronqué" dans le sens d'une coupure du graphe à un certain niveau.

Il est permis d'imprimer tous les critères (avec leur niveau et leur n° de version)

- soit à partir de la racine:

  à PRINT ALL

- soit à partir d'un critère désigné:

  à PRINT "identificateur"

De même il est possible d'imprimer l'arborescence ou une branche de cette arborescence jusqu'à un certain niveau (que l'on précise dans la commande):

- à PRINT ALL valeur numérique (1)
- à PRINT "identificateur" valeur numérique (2)

En employant (1), on imprime la racine du graphe (considérée comme de niveau 0) et ses descendants jusqu'aux critères dont la longueur du chemin entre eux et la racine est inférieure ou égale au n° de niveau défini dans la commande.

En utilisant (2), on imprime l'identificateur cité dans la commande et ses descendants jusqu'aux critères dont la longueur du chemin entre eux et l'identificateur cité, est inférieure ou égale au n° de niveau défini dans la commande.

C. Exemple.

Reprenons l'exemple de la figure 1:

- si nous désirons imprimer le contenu de la base de données avant d'y avoir créé quelque chose, le système nous répond qu'elle est vide: cfr P1
- si nous voulons imprimer la structure arborescente après l'ensemble des créations que nous avons effectuées précédemment, nous obtenons P2
- si par contre nous désirons connaître uniquement la branche correspondant au critère "mémoires", cfr P3.
- nous pouvons demander une impression tronquée de l'arbre (cfr P4) ou d'une branche (cfr P5)
- pour consulter la base de données, c.à.d. pour savoir si par exemple le critère "hardware" s'y trouve créé: P6.

4.3 Type de commande: à MODIFY.

A. Syntaxe.

$$[b]_{\epsilon}\ \underline{a}\ [b]_{+}\ \underline{M[ODIFY]}\ [b]_{+}\left\{\begin{array}{l}\underline{N[AME]}\ [b]_{+}\ \underline{"id1"}\ [b]_{+}\ \underline{BY}\ [b]_{+}\ \underline{"id2"}\\ \underline{V[ERSION]}\ [b]_{+}\ \underline{"id"}\ [b]_{+}\ \underline{n°vers}\ [b]_{+}\ \underline{BY}\ [b]_{+}\ \underline{n°vers}\end{array}\right\}$$

B. Sémantique et fonctionnement.

Le but de ce type de commande n'est pas l'ajoute de critères dans l'arborescence, mais bien la modification de l'identificateur d'un critère ou la modification de la valeur d'un ou de plusieurs paramètres spécifiques.

Dans les deux cas, nous devons spécifier l'identificateur du critère auquel nous voulons apporter la modification.

Au niveau de ce langage de manipulation général, deux types de modifications peuvent donc se présenter:

- la modification du libellé d'un critère (nous y associons une commande caractérisée par le paramètre standard: NAME):

  à MODIFY NAME "id1" BY "id2"

  où id2 sera le nouveau libellé du critère
- la modification du n° de version d'un critère (nous y associons une commande caractérisée par le paramètre standard: VERSION):

  à MODIFY VERSION "id" n°version1 BY n°version2

  où le nouveau n° de version de ce critère (id) sera n° de version2.

C. Exemple.

Dans le cas de notre figure 35, si on estime que le nom du critère software est trop long, on peut par exemple, le remplacer par OS, qui est bien plus court; la commande et l'impression consécutive (pour vérifier l'efficience de la commande) sont illustrées en M1.

Nous savons que chaque critère créé dans la base de données porte le n° de version 2. Si pour une raison quelconque, nous aimerions que "COBOL" et "FORTRAN" soient caractérisés par le n° de version 1, nous procéderions comme en M2.

4.4 Type de commande: à ALTER.

A. Remarque.

Lors du chargement du programme de manipulation général (/EXEC VBA.FOR.GEN.LOAD), un n° de version a été défini par défaut. Cela signifie que si aucune modification du n° de version n'est faite, les critères que l'on créera par la suite auront tous le même n° de version défini dès le départ, cfr A1.

B. Syntaxe.

[b]; à [b]; A[LTER] [b]; n°version

C. Sémantique et fonctionnement.

Ce type de commande est intéressant dans la mesure où on sait que les critères que nous voulons créer par la suite auront un autre n° de version que celui qui est pris en compte actuellement.

Il nous faut donc préciser quel n° de version nous avons choisi pour la suite. Ce n° de version doit être compris entre 1 et 9. Ce nouveau n° de version ne sera modifié que par une nouvelle commande à ALTER ou par un nouveau chargement du programme de manipulation général.

D. Exemple.

Supposons que nous désirons ajouter une autre branche, à la structure arborescente de la figure 35; cette dernière branche sera créée dans un but bien précis, et nous voulons donc lui affecter un n° de version particulier (3, dans notre cas). Cette nouvelle branche sera composée de "ASSISTANCE", qui sera le précédent de "TECHNIQUE" et "ASS. LOGICIEL". Elle est reliée directement à la racine, cfr A2.

4.5 Type de commande: à SUPPRESS.

A. Syntaxe.

[b]¿ à S[UPPRESS] [b]¿ { A[LL] / "identificateur" }

B. Sémantique et fonctionnement.

Le but de ce type de commande est de modifier la structure arborescente en supprimant soit

- un critère
- une branche
- l'arbre complet.

Si nous voulons supprimer tout le contenu de la base de données, nous utilisons la commande:

à SUPPRESS ALL

Par contre, si nous désirons supprimer une branche, c'est à dire un critère et tous ses descendants, nous employons la commande

à SUPPRESS "identificateur"

C. Exemple.

Dans le cas où nous désirons supprimer toute la branche correspondant à "MEMOIRES" du graphe obtenu, nous agirons comme en S1.

Si par contre, nous désirons supprimer tout l'arbre, nous ferons comme en S2. Si après cette dernière commande, nous désirons imprimer le contenu de la base de données, le système nous communiquera le message déjà obtenu en P1.

4.6 Type de commande: à DELETE.

A. Syntaxe.

[b]. à [b]. D[ELETE] [b]. "identificateur"

B. Sémantique et fonctionnement.

Le but de ce type de commande est de pouvoir supprimer un critère, qui ne correspond pas à un sommet pendant, sans pour autant supprimer ses suivants. Ces suivants sont alors rattachés au critère précédent du critère que nous voulons supprimer.

Il suffit d'indiquer dans la commande le libellé du critère que nous voulons supprimer.

C. Exemple.

Si nous reprenons l'exemple de la figure 35, en voulant supprimer le critère "MEMOIRES" et par conséquent, en rattachant les critères "MEMOIRE CENTRALE" et "MEMOIRE AUXILIAIRE" à "HARDWARE", nous écrirons la commande:

à DELETE "MEMOIRES"

4.7 Type de commande: à INSERT.

A. Syntaxe.

[b]. à [b]. I[NSERT] [b]. "id1" [b]. "id2"
[b]. "id3"
...
[b]. END

B. Sémantique et fonctionnement.

Le but de ce type de commande est de pouvoir insérer un critère correspondant à un sommet non pendant dans l'arborescence. Nous devons donc préciser

- le critère auquel nous voulons attacher ce nouveau critère
- le critère à créer
- les critères suivants du nouveau critère créé.

Nous indiquons sur la première ligne, après le verbe INSERT, le nom du critère à créer, suivi du nom du critère auquel nous voulons le connecter. Nous indiquons ensuite, ligne par ligne, le libellé des suivants que nous voulons rattacher au critère que nous venons de créer. La commande se termine par le paramètre END.

C. Exemple.

Si nous nous sommes trompés lors de la commande précédente en supprimant le critère "MEMOIRES", et que nous voulons le recréer, nous allons insérer "MEMOIRES" à "HARDWARE", tout en connectant "MEMOIRE CENTRALE" et "MEMOIRE AUXILIAIRE" à "MEMOIRES", nous agirons de la sorte:

```
à INSERT "MEMOIRES" "HARDWARE"
          "MEMOIRE CENTRALE"
          "MEMOIRE AUXILIAIRE"
          END
```

4.8 Type de commande: à HALT.

A. Syntaxe.

[b]; à [b]; H[ALT]

B. Sémantique et fonctionnement.

Le but de ce type de commande est, tout d'abord de fermer la base de données et ensuite de terminer le programme de manipulation général (cfr ¶1).

```
*à create first "ordinateur"
  CREATE COMPLETED
```

Fig. C1

```
*à create branch "ordinateur"
*Ø1 "hardware"
  CREATE COMPLETED
*Ø2 "memoires"
  CREATE COMPLETED
*Ø3 memoire centrale"
 ERROR: PARAMETER IDENTIFIER
*Ø3 "memoire centrale"
  CREATE COMPLETED
*Ø3 "memoire auxiliaire"
  CREATE COMPLETED
*Ø2 "cpu"
  CREATE COMPLETED
*Ø2 "i/o"
  CREATE COMPLETED
*Ø1 "software"
  CREATE COMPLETED
*Ø2 "langages"
  CREATE COMPLETED
*Ø3 "cobol"
  CREATE COMPLETED
*Ø3 "fortran"
  CREATE COMPLETED
*end
```

Fig. C2

```
*àcreate node "logiciels" "software"
  CREATE COMPLETED
```

Fig. C3

```
*à print all
 DATA BASE IS EMPTY
```

Fig. P1

```
*à print all
ØØ  ORDINATEUR                    2
   Ø1  HARDWARE                     2
      Ø2  MEMOIRES                     2
         Ø3  MEMOIRE CENTRALE             2
         Ø3  MEMOIRE AUXILIAIRE           2
      Ø2  CPU                          2
      Ø2  I/O                          2
   Ø1  SOFTWARE                     2
      Ø2  LANGAGES                     2
         Ø3  COBOL                        2
         Ø3  FORTRAN                      2
      Ø2  LOGICIELS                    2
PRINT COMPLETED
```

Fig. P2

```
*àprint "memoires"
Ø2  MEMOIRES                      2
   Ø3  MEMOIRE CENTRALE             2
   Ø3  MEMOIRE AUXILIAIRE           2
PRINT COMPLETED
```

Fig. P3

```
*àpa 2
ØØ  ORDINATEUR                    2
   Ø1  HARDWARE                     2
   Ø1  SOFTWARE                     2
PRINT COMPLETED
```

Fig. P4

```
*àprint "software" 2
Ø1  SOFTWARE                      2
   Ø2  LANGAGES                     2
   Ø2  LOGICIELS                    2
PRINT COMPLETED
```

Fig. P5

```
*à print "hardware"1
Ø1  HARDWARE                      2
PRINT COMPLETED
```

Fig. P6

```
*à modify name "software" by "os"
MODIFY COMPLETED

*àpa 2
ØØ  ORDINATEUR                    2
    Ø1  HARDWARE                      2

    Ø1  OS                            2

PRINT COMPLETED
```

Fig. M1

```
*àmodify version "fortran" 2 by 1
MODIFY COMPLETED

*àmodify v "cobol" 2 by 1
MODIFY COMPLETED

*à print "langages"
Ø2  LANGAGES                      2
    Ø3  COBOL                         1

    Ø3  FORTRAN                       1

PRINT COMPLETED
```

Fig. M2

```
BY DEFAULT YOUR NUMBER OF VERSION IS 1
DO YOU KEEP IT ?
*n
CHOICE A NUMBER OF VERSION BETWEEN 1 TO 9
*2
```

Fig. A1

```
*à alter 3
 YOUR NEW VERSION IS 3
*àcnode "assistance" "ordinateur"
  CREATE COMPLETED
*àcn "technique" "assistance"
  CREATE COMPLETED
*àc node "ass. logiciel" "assistance"
  CREATE COMPLETED
*àp "assistance"
 Ø1  ASSISTANCE                    3
    Ø2  TECHNIQUE                    3

    Ø2  ASS. LOGICIEL                3

 PRINT COMPLETED
```

Fig. A2

```
*à suppress "memoires"
SUPPRESS COMPLETED
*à print all
ØØ  ORDINATEUR                    2
    Ø1  HARDWARE                  2
        Ø2  I/O                   2
        Ø2  CPU                   2
    Ø1  OS                        2
        Ø2  LANGAGES              2
            Ø3  COBOL                 1
            Ø3  FORTRAN               1
        Ø2  LOGICIELS             2
    Ø1  ASSISTANCE                3
        Ø2  TECHNIQUE             3
        Ø2  ASS. LOGICIEL         3
PRINT COMPLETED.
```

Fig. S1

```
*àsa
SUPPRESS COMPLETED
```

Fig. S2

```
*à halt
THE DATA BASE IS CLOSED
EJCTERMX TERMINATED INTER PROCESS COMMUNICATION
```

Fig. H1

5.1 Type de commande: à CREATE.

A. Syntaxe.

```
[b]. a [b]. C[REATE] [b] { F[IRST] [b]. "id" [ [b]. A[GGREGAT] [b]. XXX ]
                         N[ODE] [b]. "id1" [ [b]. "id2" ] [ [b]. W[EIGHT] [b]. 999 ]
                                                          [ [b]. A[GGREGAT] [b]. XXX ]
                         L[EAF] [b]. "id1" [ [b]. "id2" ] [ [b]. W[EIGHT] [b] 999 ]
                                                          [ [b]. F[UNCT] 1b [b]. 9999b [b]. 99 ]
                                                          [ [b]. F[UNCT] 2b [b]. 9999b [b]. 99 ]
                                                          [ [b]. F[UNCT] 3b [b]. 9999b [b]. 99 ]
                                                          [ [b]. F[UNCT] 4b [b]. 9999b [b]. 99 ]
                                                          [ [b]. F[UNCT] 5b [b]. 9999b [b]. 99 ]
                         B[RANCH] [b] "id1"
                           { niveau [b]. N[ODE] [b]. "id2" [ [b]. W[EIGHT] [b]. 999 ]
                                                           [ [b]. A[GGREGAT] [b]. XXX ]
                             niveau [b]. L[EAF] [b]. "id2" [ [b]. F[UNCT] 1b [b]. 9999b [b]. 99 ]
                                                           [ [b]. F[UNCT] 2b [b]. 9999b [b]. 99 ]
                                                           [ [b] F[UNCT] 3b [b]. 9999b [b]. 99 ]
                                                           [ [b] F[UNCT] 4b [b]. 9999b [b]. 99 ]
                                                           [ [b] F[UNCT] 5b [b]. 9999b [b]. 99 ]
                             ...
                           END
                         }
                       }
```

B. Sémantique et fonctionnement.

Le but de ce type de commande est de créer l'arborescence des critères ainsi que les valeurs des paramètres spécifiques associés à chacun de ces critères.

La création de l'arborescence commence nécessairement par la création de la racine (FIRST) à laquelle est associé un seul paramètre spécifique qui est l'opérateur d'aggrégation.

Il est ensuite possible de créer

- soit un critère noeud (NODE) ou un critère feuille (LEAF)
- soit une branche de critères (BRANCH) noeud(s) et/ou feuille(s).

L'introduction des critères (c.à.d. les libellés uniquement) peut être indépendante de l'introduction des valeurs de ses paramètres spécifiques. En effet, il est permis de créer une arborescence complète ou partielle reprenant uniquement les identificateurs des critères, et lors d'une étape ultérieure de leurs associer les valeurs nécessaires.

Lors de la création d'un identificateur de critère (id1), il est nécessaire de spécifier son précédent (id2), sauf pour la racine.

Lors de l'association de valeurs de paramètres spécifiques à un critère existant, il est obligatoire de spécifier l'identificateur de ce critère (sauf la racine où il ne faut pas spécifier d'identificateur).

Les paramètres spécifiques associés à un noeud sont:

- le poids dont la valeur numérique entière doit être comprise entre 1 et 99
- l'opérateur d'aggrégation dont la gamme des valeurs permises est définie en (B1).

Les paramètres spécifiques associés à une feuille sont:

- le poids (même contrainte que pour les noeuds)
- la fonction d'utilité qui est caractérisée au minimum par deux valeurs et au maximum par cinq valeurs; chacune de ces valeurs est définie par une abscisse et une ordonnée
  - abscisse: valeur numérique entière comprise entre 0 et 9999
  - ordonnée: valeur numérique entière comprise entre 0 et 99.

Remarque:

Si les paramètres spécifiques ne sont pas cités lors de l'introduction d'un critère au niveau de l'arborescence, ils seront cependant initialisés:

- le poids sera initialisé à 0: cela signifie que ce critère ne possède pas de poids, car cette valeur nulle n'est pas considérée comme correcte; en effet, le poids doit être compris entre 1 et 99
- l'opérateur d'aggrégation est initialisé à "???"
- la fonction sera initialisée par des espaces vides.

En résumé, les initialisations de ces paramètres ne correspondent pas à des "valeurs" permises.

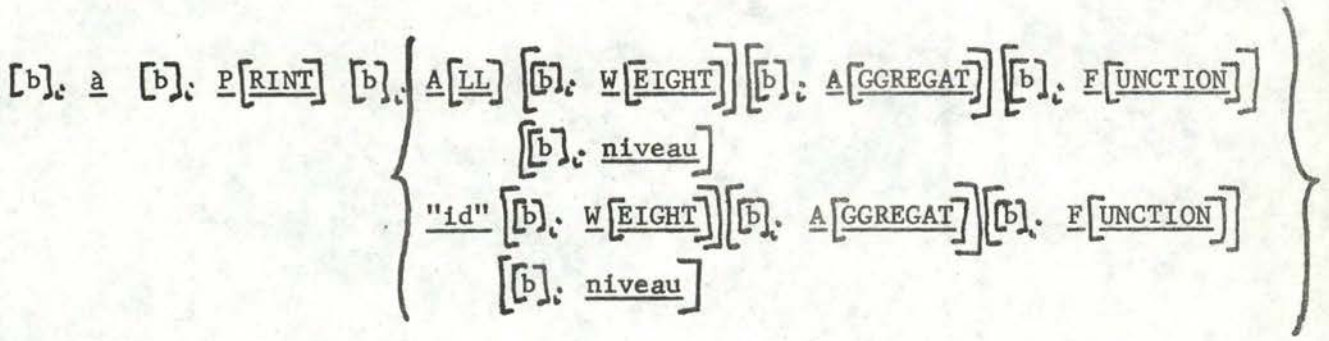
C. Exemple.

Nous désirons créer la structure arborescente représentée à la figure 35. Nous expliquons une façon de procéder:

- création de la racine de l'arborescence: cfr C10
- création d'une branche reliée à la racine "ORDINATEUR": cfr C11
- création d'un noeud "SOFTWARE" relié à la racine "ORDINATEUR": cfr C12
- création de la feuille "LOGICIEL" reliée au noeud "SOFTWARE": cfr C13
- création d'une branche reliée au noeud "SOFTWARE": cfr C14

5.2 Type de commande: à PRINT.

A. Syntaxe.



B. Sémantique et fonctionnement.

Le but de ce type de commande est d'effectuer l'affichage au terminal d'un état partiel ("id") ou complet (ALL) de l'arborescence de critères auquel on peut ajouter ou non les paramètres spécifiques de ces critères (suivant qu'ils sont ou non cités dans la commande).

Il est évident que si on spécifie le paramètre FUNCTION dans la commande, il intervient pour chaque feuille que l'on veut imprimer, mais il n'intervient pas pour les noeuds et la racine.

Afin de détecter plus facilement la racine, les noeuds et les feuilles sur l'état imprimé, on ajoute en face de chaque identificateur de critère, un message qui caractérise le type de critère.

Pour plus de détails, voir la commande à PRINT du langage de manipulation général.

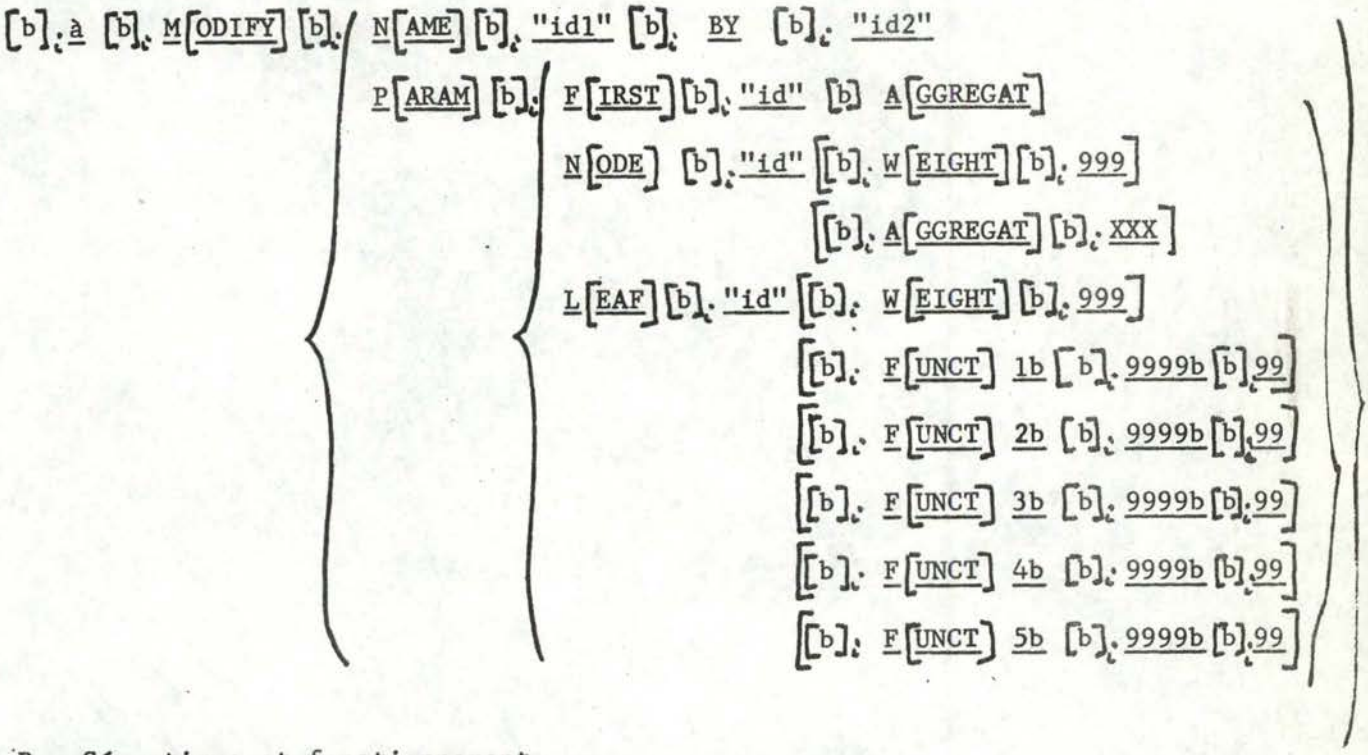
C. Exemple.

Nous désirons afficher un état partiel ou complet à partir des informations de la base de données qui ont été créées dans l'exemple de la création (§ 5.1):

- impression de tous les libellés de critères sans les valeurs de leurs paramètres spécifiques: cfr P10
- impression de l'arborescence complète, c.à.d. l'ensemble des critères ainsi que les valeurs des paramètres spécifiques associés: cfr P11
- impression de tous les libellés des critères ainsi que les valeurs du paramètre spécifique WEIGHT: cfr P12.

5.3 Type de commande: à MODIFY.

A. Syntaxe.



B. Sémantique et fonctionnement.

Le but de ce type de commande est

- soit de modifier l'identificateur d'un critère
- soit de modifier une ou plusieurs valeurs de paramètres associées à un critère.

Lors d'une modification du nom d'un critère, l'utilisateur doit spécifier le paramètre standard NAME, de même que l'ancien identificateur (id1) et le nouveau (id2).

Lors d'une modification d'un ou de plusieurs valeurs de paramètres spécifiques, l'utilisateur doit citer dans la commande le paramètre standard PARAM, de même que le ou les paramètres spécifiques qu'il veut changer, suivi de la nouvelle valeur qu'il veut lui affecter. Il faut évidemment spécifier le libellé du critère auquel on veut apporter cette modification (voir explication des paramètres spécifiques au type de commande: à CREATE).

La commande à MODIFY PARAM permet non seulement la modification de valeurs de paramètres déjà créés, mais aussi la création de celles-ci. En effet, les paramètres spécifiques définis lors de l'introduction de critères sont initialisés; on peut considérer qu'ils ont déjà reçu une valeur (cfr commande à CREATE ...) et il est par conséquent possible de modifier cette valeur.

C. Exemple.

Nous nous basons sur l'arborescence imprimée en P11:

- modification du libellé "LOGICIEL" par le libellé "OS": cfr M10
- modification de la deuxième valeur de la fonction d'utilité associée au critère élémentaire "FORTRAN": cfr M11
- modification du poids associé au critère élémentaire "FORTRAN": cfr M12.

Une impression est réalisée après chaque modification afin de visualiser ce changement.

5.4 Type de commande: à ALTER.

A. Syntaxe.

[b]. à [b]. A[LTER] [b]. "id" { N[ODE] [b]. BY [b]. L[EAF] | L[EAF] [b]. BY [b]. N[ODE] }

B. Sémantique et fonctionnement.

Le but de ce type de commande est de modifier le type de critère; deux cas sont prévus:

- soit qu'un noeud devient une feuille (la commande associée est caractérisée par les paramètres standards: NODE BY LEAF)
- soit qu'une feuille devient un noeud (la commande associée est caractérisée par les paramètres standards LEAF BY NODE).

Ce type de commande est nécessaire:

- dans le premier cas, lorsque l'on supprime une branche d'une arborescence, il reste un sommet pendant qui correspond à un noeud; vu le caractère des commandes, il est impossible d'introduire une fonction d'utilité comme paramètre spécifique de ce critère
- dans le second cas: lorsque l'on ajoute un critère (ou une branche) à la suite d'un sommet pendant qui est une feuille, il est nécessaire dans ce cas de modifier la feuille en noeud; en effet, il serait impossible d'introduire l'opérateur d'aggrégation comme paramètre spécifique de ce critère.

C. Exemple.

Les exemples se basent sur l'arborescence imprimée en P11:

- essai de modification du type de critère associé au critère "COBOL"; essai de changement d'une feuille en une feuille: cfr A10
- modification du type de critère associé au critère "COBOL"; changement d'une feuille en noeud: cfr A11
- modification du type de critère associé au critère "COBOL"; changement d'un noeud en feuille: cfr A12.

5.5 Type de commande: à LINK.

A. Syntaxe.

[b]¿ à L[INK] [b]¿ "id1" [b]¿ TO [b]¿ "id2" [b]¿ W[EIGHT] [b]¿ 999

B. Sémantique et fonctionnement.

Le but de ce type de commande est de lier deux critères existants déjà dans l'arborescence; c.à.d. déjà créé par le type de commande: à CREATE.

Elle permet de donner à un critère (id2) un nouveau suivant et à un autre critère (id1), un nouveau précédent.

Contrairement au type de commande à CREATE, qui crée des critères et établit des relations entre ces critères, la commande à LINK ne crée pas de critères, mais établit des liens entre eux. Ce type de lien ne peut pas être créé pour deux critères possédant déjà une arête les reliant.

La structure purement arborescente disparaît par l'introduction d'une telle relation.

C. Exemple.

L'exemple se base sur l'arborescence imprimée en P11; on crée un lien entre les critères "CPU" et "MEMOIRES". Le critère "CPU" est considéré comme critère immédiatement subordonné (suivant) du critère "HARDWARE" et du critère "MEMOIRES": cfr L10.

Cette création particulière est visualisée par l'introduction du message (RELATION) en face du critère "CPU" considéré comme critère immédiatement subordonné au critère "MEMOIRES".

5.6 Type de commande: à UNLINK.

A. Syntaxe.

[b]. à [b]. U[NLINK] [b]. "id1" [b]. FROM [b]. "id2"

B. Sémantique et fonctionnement.

Le but de ce type de commande est de supprimer un lien entre deux critères établis préalablement par un type de commande: à LINK.

Le précédent est représenté par "id1" et le suivant par "id2". Elle permet donc d'enlever la relation entre le critère "id2" et

- soit le suivant dans le cas où le critère "id1" est un sommet pendant
- soit les descendants dans le cas où le critère "id1" est un sommet non pendant.

C. Exemple.

Les exemples se basent sur l'arborescence imprimée en P11 et L10. L'exemple que nous avons repris (cfr U10) montre la façon de procéder pour enlever le lien entre les deux critères "CPU" et "MEMOIRES"; ce lien ayant été créé par la commande: à LINK.

5.7 Type de commande: à ENTRY.

A. Syntaxe.

[b]$_c$ à [b]$_c$ E[NTRY]

B. Sémantique et fonctionnement.

Le but de ce type de commande est de déterminer les projets et d'introduire les données variables pour chacun de ces projets; c.à.d. les valeurs d'entrée des fonctions d'utilité.

Pour chacun des critères élémentaires de l'arborescence, on spécifie les valeurs d'entrée dans l'ordre où les libellés des projets ont été déterminés.

Afin de faciliter la tâche de l'utilisateur, un message définissant l'ordre des projets apparaît au terminal ainsi que chacun des critères élémentaires pour lesquels, il faut introduire les valeurs.

Le libellé des projets ne peut dépasser 10 caractères alphanumériques et est entouré de double quotes. Le nombre de projets est limité à 20. Les valeurs d'entrée ne peuvent dépasser 5 caractères numériques et sont séparées par au moins un espace.

C. Exemple.

Les exemples se basent sur l'arborescence imprimée en P11.

- création d'une première série de projets auxquels on associe les valeurs d'entrée des fonctions d'utilité pour chacun des critères élémentaires: cfr E10
- création d'un nouveau projet ainsi que les valeurs d'entrée des fonctions d'utilité: cfr E11
- modification de la valeur d'entrée de la fonction d'utilité du critère élémentaire "COBOL" pour le projet "IBM"; plusieurs modifications peuvent être réalisées de manière consécutive; la fin d'une série est caractérisée par la spécification du paramètre standard: END; cfr E12.

5.8 Type de commande: à WRITE.

A. Syntaxe.

[b]₁ à [b]₂ W[RITE] [b]₂ "projet1" [[b]₂ "projetn"]₃

B. Sémantique et fonctionnement.

Le but de ce type de commande est de réaliser la mise en forme des données qui seront traitées par le programme de choix Cat.

On peut classer ces données en 2 grands types:

a. commandes et instructions du langage SEL qui déterminent un programme de traitement des données du type (b): cfr B1
b. les critères élémentaires et les valeurs de leurs paramètres spécifiques, les critères non élémentaires et les valeurs de leurs paramètres spécifiques, les valeurs d'entrée au fonction d'utilité pour chaque projet et chaque critère élémentaire.

Les commandes et instructions du langage SEL sont décrites dans le fichier VBA.FOR.CAT.FICHPROG. L'ensemble des données mises en forme après cette commande sont décrites dans le fichier VBA.FOR.CAT.FICHMEF.

C. Exemple.

L'exemple montre la mise en forme des données pour les projets "IBM" et "SIEMENS". La liste des critères élémentaires et celle des critères non élémentaires sont imprimées sur le terminal de manière à suivre l'évolution de la mise en forme: cfr W10. Le résultat obtenu après cette commande est imprimé en W11.

5.9 Type de commande: à SUPPRESS.

Voir type de commande à SUPPRESS du langage de manipulation général.

5.10 Type de commande: à HALT.

Voir type de commande à HALT du langage de manipulation général.

```
*àcf"ordinateur"a ca
 CREATE COMPLETED
*
```

Fig. C10

```
 àcb"ordinateur"
*Ø1 n "hardware" a c+- w 55
 CREATE COMPLETED
*Ø2l "i/o"w22 f1 1Ø 99 f2 2Ø 75 f3 3Ø 5Ø f4 4Ø 25 *
*f5 5Ø ØØ
 CREATE COMPLETED
*Ø2 l"cpu" w 33 f1 2 99 f2 5 75 f3 1Ø 2Ø f4 2Ø Ø
 CREATE COMPLETED
*Ø2 n "memoires" a d+- w 44
 CREATE COMPLETED
*Ø3 l"memoire centrale" f1 16 1Ø f2 32 2Ø f3 64 4Ø *
*f4 256 6Ø f5 1Ø24 99 w 66
 CREATE COMPLETED
*Ø3 l"memoires auxiliaires" f1 3 5Ø f2 4 99 w 25
 CREATE COMPLETED
*end
```

Fig. C11

```
 àcn"software""ordinateur"a c+- w 33
 CREATE COMPLETED
*
```

Fig.C12

```
 àcl"logiciel""software" f1 33 55 f2 55 77 w 22
 CREATE COMPLETED
*
```

Fig. C13

```
 àcb"software"
*Ø1n"langages" a c++ w 22
 CREATE COMPLETED
*Ø2l"cobol"f1 23 67 f2 45 89 w 11
 CREATE COMPLETED
*Ø2l"fortran"f1 32 45 f2 56 78
 CREATE COMPLETED
*end
*
```

Fig. C14

Fig. P10

```
ØØ ORDINATEUR                    (RACINE)
   Ø1 HARDWARE                     (NOEUD)
      Ø2 I/O                          (FEUILLE)
      Ø2 CPU                          (FEUILLE)
      Ø2 MEMOIRES                     (NOEUD)
         Ø3 MEMOIRE CENTRALE             (FEUILLE)
         Ø3 MEMOIRES AUXILIAIRES         (FEUILLE)
   Ø1 SOFTWARE                     (NOEUD)
      Ø2 LOGICIEL                     (FEUILLE)
      Ø2 LANGAGES                     (NOEUD)
         Ø3 COBOL                        (FEUILLE)
         Ø3 FORTRAN                      (FEUILLE)
PRINT COMPLETED
*
```

Fig. M10

```
àmn"logiciel"by"os"
MODIFY COMPLETED
*
```

```
àp"software"
Ø1 SOFTWARE                      (NOEUD)
   Ø2 OS                           (FEUILLE)
   Ø2 LANGAGES                     (NOEUD)
      Ø3 COBOL                        (FEUILLE)
      Ø3 FORTRAN                      (FEUILLE)
PRINT COMPLETED
*
```

Fig. M11

```
àmpl"fortran" f2 88 99
MODIFY COMPLETED
*
àp"fortran" f
Ø3 FORTRAN                       (FEUILLE)
                                 FONCTION = ØØ32 45
                                            ØØ88 99
PRINT COMPLETED
*
```

Fig. M12

```
àmpl"fortran" w 55
MODIFY COMPLETED
*
àp"fortran" w
Ø3 FORTRAN                       (FEUILLE)
                                 POIDS    = Ø55
PRINT COMPLETED
*
```

Fig. P11

apa wat
ØØ ORDINATEUR (RACINE)
AGGREGAT = CA
(NOEUD)

POIDS = Ø55

AGGREGAT = C+-

Ø1 HARDWARE

Ø2 I/O (FEUILLE)

POIDS = Ø22

FONCTION = ØØ1Ø 99
ØØ2Ø 75
ØØ3Ø 5Ø
ØØ4Ø 25
ØØ5Ø ØØ

Ø2 CPU (FEUILLE)

POIDS = Ø33

FONCTION = ØØØ2 99
ØØØ5 75
ØØ1Ø 2Ø
ØØ2Ø ØØ

Ø2 MEMOIRES (NOEUD)

POIDS = Ø44

AGGREGAT = D+-

Ø3 MEMOIRE CENTRALE (FEUILLE)

POIDS = Ø66

FONCTION = Ø016 1Ø
Ø032 2Ø
Ø064 4Ø
Ø256 6Ø
1Ø24 99

Ø3 MEMOIRES AUXILIAIRES (FEUILLE)

POIDS = Ø25

FONCTION = ØØØ3 5Ø
ØØØ4 99

```
Ø1 SOFTWARE                    (NOEUD)
                               POIDS    = Ø33
                               AGGREGAT = C+-

   Ø2 LOGICIEL                 (FEUILLE)
                               POIDS    = Ø22
                               FONCTION = ØØ33 55
                                          ØØ55 77

   Ø2 LANGAGES                 (NOEUD)
                               POIDS    = Ø22
                               AGGREGAT = C++

      Ø3 COBOL                 (FEUILLE)
                               POIDS    = Ø11
                               FONCTION = ØØ23 67
                                          ØØ45 89

      Ø3 FORTRAN               (FEUILLE)
                               POIDS    = ØØØ
                               FONCTION = ØØ32 45
                                          ØØ56 78

PRINT COMPLETED
*
```

```
àp a w
ØØ ORDINATEUR                    (RACINE)
   Ø1 HARDWARE                     (NOEUD)
                                   POIDS    = Ø55
      Ø2 I/O                         (FEUILLE)
                                     POIDS    = Ø22
      Ø2 CPU                         (FEUILLE)
                                     POIDS    = Ø33
      Ø2 MEMOIRES                    (NOEUD)
                                     POIDS    = Ø44
         Ø3 MEMOIRE CENTRALE           (FEUILLE)
                                       POIDS    = Ø66
         Ø3 MEMOIRES AUXILIAIRES       (FEUILLE)
                                       POIDS    = Ø25
   Ø1 SOFTWARE                   (NOEUD)
                                 POIDS    = Ø33
      Ø2 OS                        (FEUILLE)
                                   POIDS    = Ø22
      Ø2 LANGAGES                  (NOEUD)
                                   POIDS    = Ø22
         Ø3 COBOL                    (FEUILLE)
                                     POIDS    = Ø11
         Ø3 FORTRAN                  (FEUILLE)
                                     POIDS    = Ø55

PRINT COMPLETED
*
```

Fig. P12

```
àl"cpu"to"memoires"
LINK COMPLETED
*
```

Fig. L10

```
àp"hardware"
Ø1 HARDWARE                          (NOEUD)
   Ø2 I/O                               (FEUILLE)
   Ø2 CPU                               (FEUILLE)
   Ø2 MEMOIRES                          (NOEUD)
      Ø3 CPU                               (RELATION)
      Ø3 MEMOIRE CENTRALE                  (FEUILLE)
      Ø3 MEMOIRES AUXILIAIRES              (FEUILLE)
PRINT COMPLETED
*
```

```
àu"cpu"from"memoires"
UNLINK COMPLETED
*
```

Fig. U10

```
àp"hardware"
Ø1 HARDWARE                          (NOEUD)
   Ø2 I/O                               (FEUILLE)
   Ø2 CPU                               (FEUILLE)
   Ø2 MEMOIRES                          (NOEUD)
      Ø3 MEMOIRE CENTRALE                  (FEUILLE)
      Ø3 MEMOIRES AUXILIAIRES              (FEUILLE)
PRINT COMPLETED
*
```

```
àa"cobol" node by leaf
OBJECT IS ALREADY A LEAF
ALTER COMPLETED
*
```

Fig. A10

```
àa"cobol"lbyn
ALTER COMPLETED
*
```

Fig. A11

```
àa"cobol"nbyl
ALTER COMPLETED
*
```

Fig. A12

Fig. E10

```
 àe
 OPEN CAT
 CREATION (Y) OR MODIFY (N)
*y
 FIRST CREATION (Y OR N )
*y
 SPECIFY NAMES OF PROJECTS BETWEEN QUOTE
*"ibm""siemens""dec"
 PROJECT ORDER IS : IBM
                    SIEMENS
                    DEC
 I/O
*28 31 34
 IBM
 00028
 SIEMENS
 00031
 DEC
 00034
  CREATE OF VALUES COMPLETED
 CPU
*33 44 55
 IBM
 00033
 SIEMENS
 00044
 DEC
 00055
  CREATE OF VALUES COMPLETED
 MEMOIRE CENTRALE
*45 67 89
 IBM
 00045
 SIEMENS
 00067
 DEC
 00089
  CREATE OF VALUES COMPLETED
 MEMOIRES AUXILIAIRES
*53 21 23
 IBM
 00053
 SIEMENS
 00021
 DEC
 00023
  CREATE OF VALUES COMPLETED
 OS
*67 21 34
 IBM
 00067
 SIEMENS
 00021
 DEC
 00034
  CREATE OF VALUES COMPLETED
```

```
 COBOL
*78 54 32
 IBM
 ØØØ78
 SIEMENS
 ØØØ54
 DEC
 ØØØ32
  CREATE OF VALUES COMPLETED
 FORTRAN
*54 32 456
 IBM
 ØØ054
 SIEMENS
 ØØØ32
 DEC
 Ø0456
  CREATE OF VALUES COMPLETED

 ENTRY TERMINATED
```

Fig. E12

```
 àe
 OPEN CAT
 CREATION (Y) OR MODIFY (N)
*n
  YOU MUST SPECIFY CRITERIA PROJECT VALUE
*"cobol""ibm"9999
 COBOL
 IBM
 Ø9999
 MODIFY COMPLETED
*
 end
 ENTRY TERMINATED
*
```

Fig. E11

```
 OPEN CAT
 CREATION (Y) OR MODIFY (N)
*y
 FIRST CREATION (Y OR N )
*n
 SPECIFY NAMES OF PROJECTS BETWEEN QUOTE
*"cdc"
 PROJECT ORDER IS : CDC
 I/O
*543
 CDC
 00543
  CREATE OF VALUES COMPLETED
 CPU
*67
 CDC
 00067
  CREATE OF VALUES COMPLETED
 MEMOIRE CENTRALE
*65
 CDC
 00065
  CREATE OF VALUES COMPLETED
 MEMOIRES AUXILIAIRES
*78
 CDC
 00078
  CREATE OF VALUES COMPLETED
 OS
*123
 CDC
 00123
  CREATE OF VALUES COMPLETED
 COBOL
*56
 CDC
 00056
  CREATE OF VALUES COMPLETED
 FORTRAN
*54
 CDC
 00054
  CREATE OF VALUES COMPLETED

 ENTRY TERMINATED
```

Fig. W10

```
*àw"ibm""siemens"
 DATA BASE IS OPENED
I/O
CPU
MEMOIRE CENTRALE
MEMOIRES AUXILIAIRES
OS
COBOL
FORTRAN
END LIST
MEMOIRES
HARDWARE
LANGAGES
SOFTWARE
ORDINATEUR
END EXIT
DATA BASE IS CLOSED
```

Fig. W11

```
 2:0000 *SEL,LIS
 3:0000    INPUT ELC
 4:0000    INPUT CAS
 5:0000 1 TITLE 2,1
 6:0000    READ X(1,K99)
 7:0000    EFFECTIVENESS
 8:0000    OUTPUT (E) SORT,HIST
 9:0000    REPEAT 1,1
10:0000    STOP
11:0000    END
12:0000 *EXECUTE
13:0000 LIST
14:0000 1000        I/O                                  001099002075
003050004025005000
15:0000 1010        CPU                                  000299000575
001020002000
16:0000 1020        MEMOIRE CENTRALE                     001610003220
006440025660102499
17:0000 1030        MEMOIRES AUXILIAIRES                 000350000499
18:0000 1040        OS                                   003355005577
19:0000 1050        COBOL                                002367004589
20:0000 1060        FORTRAN                              003245008899
21:0000           END
22:0000 LIST
23:0000 5000 D+-  MEMOIRES                               102066103025
24:0000 5010 C+-  HARDWARE                               100022101033
500044
25:0000 5020 C++  LANGAGES                               105011106055
26:0000 5030 C+-  SOFTWARE                               104022502022
27:0000 5040 CA   ORDINATEUR                             501055503033
28:0000           END
29:0000 SYSTEM NUMBER 1
30:0000 000280003300045000530006700078000054
31:0000 SYSTEM NUMBER 2
32.0000 000310004400067000210002100054000032
33:0000 *EXIT
```

CHAPITRE VI COMMANDES DU LANGAGE DE MANIPULATION ELECTRE II.

6.1 Type de commande: à CREATE.

A. Syntaxe.

```
[b]. à [b]. C[REATE] [b]. / F[IRST] [b]. "identificateur"
                           N[ODE] [b]. "id1" [b]. "id2" [b]. [W[EIGHT] [b]. 99999]
                           L[EAF] [b]. "id1" [b]. "id2" [b]. [W[EIGHT] [b]. 99999]
                                                             [SD1 [b]. 99999]
                                                             [SD2 [b]. 99999]
                                                             [SQN [b] 99999]
                                                             [SC[ALE] [b]. 99999]
                           B[RANCH] [b]. "id1"
                             [b]. niveau [b]. N[ODE] [b]. "id2" [b]. [W[EIGHT] [b]. 99999]
                                         L[EAF] [b]. "id2" [b]. [W[EIGHT] [b]. 99999]
                                                               [SD1 [b]. 99999]
                                                               [SD2 [b]. 99999]
                                                               [SQN [b]. 99999]
                                                               [SC[ALE] [b]. 99999]
                           ...
                           END
```

B. Sémantique et fonctionnement.

Le but de ce type de commande est de créer l'arborescence des critères ainsi que les valeurs des paramètres spécifiques associés à chacun de ces critères.

La création de la racine doit s'effectuer en premier lieu, avant toute création de noeud ou de feuille. Nous lui associons le paramètre standard FIRST.

Il est ensuite possible de créer

- soit un critère noeud (NODE) ou un critère feuille (LEAF)
- soit une branche de critères (BRANCH) noeud(s) et/ou feuille(s).

L'introduction des libellés des critères peut être indépendante de l'introduction des valeurs des paramètres spécifiques qui leurs sont associés. Cependant, si un paramètre spécifique n'est pas créé lors de la création du libellé du critère, sa valeur est initialisée à zéro. La création de la valeur d'un tel paramètre spécifique (initialisé à 0) représente donc une modification de sa valeur; c'est pourquoi, nous n'avons pas autorisé la création d'un tel paramètre par le type de commande à CREATE, mais bien par, et uniquement par le type de commande à MODIFY (cfr supra).

Lors de la création du libellé d'un critère (id1), il est nécessaire de spécifier son précédent (id2), sauf pour la racine. On peut en plus créer ses paramètres spécifiques qui diffèrent suivant le type de sommet:

- pour la racine: aucun paramètre spécifique
- pour les noeuds: le poids dont la valeur numérique entière doit être comprise entre 0 et 99999

- pour les feuilles:
  - le poids: même contrainte que les noeuds
  - les seuils de discordance (SD1 et SD2) dont les valeurs numériques doivent être comprises entre 0 et 99999
  - la condition sine qua non (SQN) de valeur numérique comprise entre 0 et 99999
  - le facteur d'échelle (SCALE) de valeur numérique comprise entre 0 et 99 (*)

L'ordre d'écriture des paramètres spécifiques dans ce type de commande n'a pas d'importance. Nous rappelons donc que si un paramètre n'a pas été créé lors de la création du libellé du critère, il ne peut être créé que par l'ordre MODIFY ... (cfr supra).

(*) En théorie, le facteur d'échelle (SCALE) prend la valeur 1 ou -1 pour indiquer le sens de variation de l'évaluation des projets, cependant, nous n'avons pas eu le temps de l'implémenter; c'est pourquoi nous conseillons à l'utilisateur d'introduire la valeur 1 pour le facteur d'échelle.

C. Exemple.

Nous désirons créer la structure arborescente représentée à la figure 35. Nous expliquons une façon de procéder:

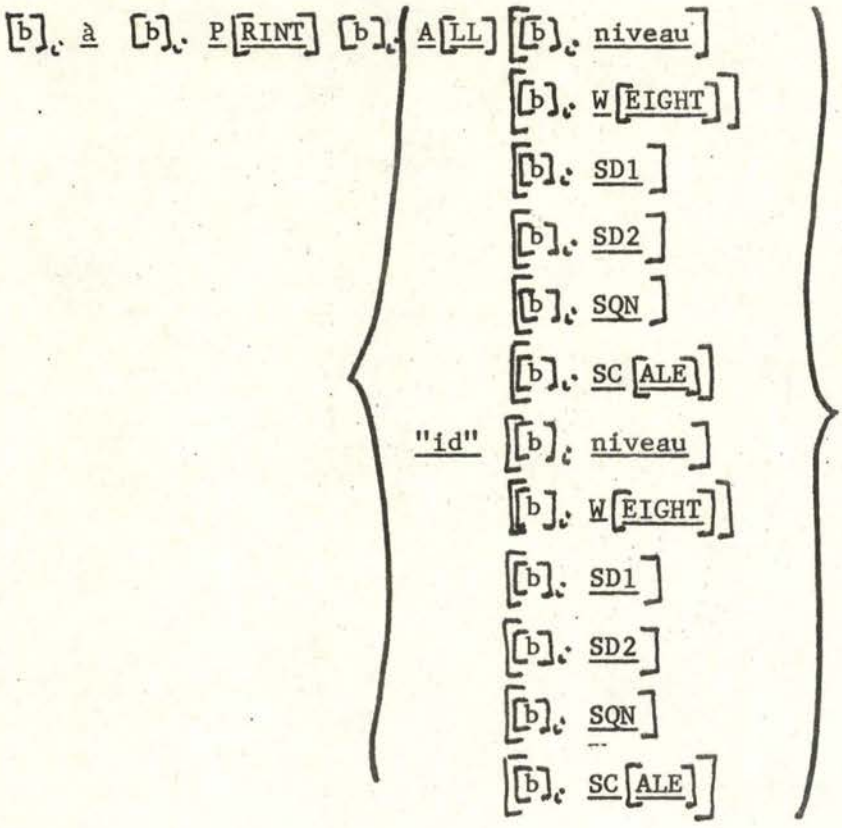
a. création de la racine de l'arborescence, création du critère "HARDWARE" et du critère "I/O" auquel nous avons associé comme paramètres spécifiques: WEIGHT (00005)
SD1 (00004)
SD2 (00005)
SQN (00000)
SCALE (01)
cfr C20,

b. création d'une branche reliée à "HARDWARE" cfr C21

c. création d'une branche reliée à "ORDINATEUR" cfr C22

6.2 Type de commande: à PRINT.

A. Syntaxe.



B. Sémantique et fonctionnement.

Le but de ce type de commande est d'effectuer l'affichage au terminal d'un état partiel ("id") ou complet (ALL) de l'arborescence de critères auquel on peut ajouter ou non les paramètres spécifiques de ces critères (suivant qu'ils sont ou non cités dans la commande).

Il est évident que si on spécifie le paramètre SD1 dans la commande, il intervient pour chaque feuille que l'on veut imprimer, mais il n'intervient pas pour les noeuds et la racine.

Afin de détecter plus facilement la racine, les noeuds et les feuilles sur l'état imprimé, on ajoute en face de chaque identificateur de critère, un chiffre qui caractérise le type de critère:

(1): pour la racine

(2): pour les noeuds

(3): pour les feuilles

Pour plus de détails, voir le type de commande à PRINT du langage de manipulation général.

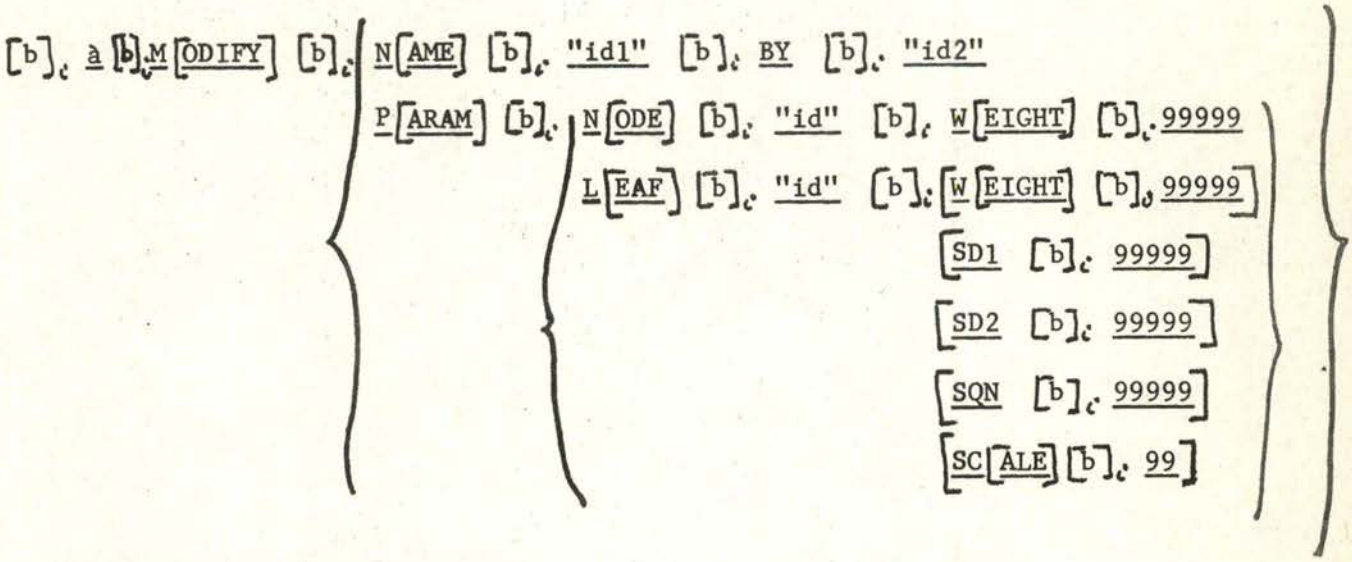
C. Exemple.

Nous désirons afficher l'état complet (avec tous les paramètres spécifiques) que nous venons de créer dans la base de données (cfr 6.1); nous obtenons la figure P20.

Nous pouvons également afficher un état partiel: cfr P21.

6.3 Type de commande: à MODIFY.

A. Syntaxe.



B. Sémantique et fonctionnement.

Le but de ce type de commande est

- soit de modifier l'identificateur d'un critère
- soit de modifier une ou plusieurs valeurs de paramètres associées à un critère.

Lors de la modification du libellé d'un critère, l'utilisateur doit spécifier le paramètre standard NAME, de même que l'ancien identificateur (id1) et le nouveau (id2).

Lors de la modification (ou de la création) de la valeur d'un ou de plusieurs paramètres spécifiques, l'utilisateur doit préciser le paramètre standard PARAM dans la commande, suivi de l'identificateur du critère concerné par cette modification; enfin, il indique le ou les paramètres spécifiques qu'il veut changer, suivi(s) de la nouvelle valeur qu'il désire affecter.

Ce type de commande permet également de créer les paramètres spécifiques non encore créés explicitement par l'utilisateur (en fait, le paramètre spécifique a été créé à son insu, puisqu'il a été initialisé à 0 lors de la création du libellé du critère correspondant: cfr type de commande: à CREATE).

C. Exemple.

1. modification du libellé "LOGICIEL" par le libellé "OS", sur base de l'arborescence imprimée en P20: cfr M20.
2. modification du paramètre spécifique WEIGHT du critère "OS", sur base de l'arborescence imprimée en P21: cfr M21.

6.4 Type de commande: à ALTER.

A. Syntaxe.

[b]; à [b]; A[LTER] [b]; "id" { N[ODE] [b]; BY [b]; L[EAF] | L[EAF] [b]; BY [b]; N[ODE] }

B. Sémantique et fonctionnement.

Cfr type de commande: à ALTER du langage de manipulation Cat (§ 5.4).

C. Exemple.

L'exemple se base sur l'arborescence imprimée en P20: modification du type de critère associé au critère "FORTRAN": changement de feuille en noeud: cfr A20.

6.5 Type de commande: à ENTRY.

A. Syntaxe.

[b]: à [b]: E[NTRY]

B. Sémantique et fonctionnement.

Cfr type de commande: à ENTRY du langage de manipulation Cat (§ 5.7), en tenant compte du fait que

- les libellés des projets ne peuvent dépasser 10 caractères
- les valeurs des évaluations des projets ne peuvent dépasser 5 caractères numériques.

C. Exemple.

L'exemple se base sur l'arborescence imprimée en P23:

- création d'une première série de projets, auxquels nous associons les valeurs des évaluations pour chaque critère élémentaire: cfr E20.
- pour d'autres exemples, cfr langage de manipulation Cat.

6.6 Type de commande: à WRITE.

A. Syntaxe.

[b]¿ à [b]¿ W[RITE] [b]¿ "projet1" [[b]¿ "projet2"]¿

B. Sémantique et fonctionnement.

Le but de ce type de commande est de réaliser la mise en forme des données qui seront traitées par le programme de choix Electre.

On peut classer ces données en 2 grands types:

a. les seuils de concordance (SC1, SC2 et SC3) qui sont introduits au terminal, en dernier lieu (juste avant l'exécution du programme de choix).
b. les critères élémentaires et les valeurs de leurs paramètres spécifiques.

Lors de l'exécution de cette commande, le système demande à l'utilisateur d'indiquer les seuils de concordance. L'utilisateur doit les préciser selon le format suivant:

.9999.9999.9999

où le symbole 9 représente n'importe quel nombre entier compris entre 0 et 9.

L'ensemble des données mises en forme après cette commande, sont décrites dans les fichiers

VBA.FOR.ELE.FICHP1
VBA.FOR.ELE.FICHP2
VBA.FOR.ELE.FICHP3.

C. Exemple.

L'exemple montre la mise en forme des données pour les projets IBM et Siemens. Les résultats (c.à.d. les fichiers

VBA.FOR.ELE.FICHP1

VBA.FOR.ELE.FICHP2

VBA.FOR.ELE.FICHP3)

sont imprimés en W20.

6.7 Type de commande: à SUPPRESS.

Voir type de commande à SUPPRESS du langage de manipulation général.

Dans notre exemple, on s'est volontairement trompé en raccordant "LANGAGES" à "ORDINATEUR". En fait, "LANGAGE" doit être subordonné à "OS"; nous supprimons donc "LANGAGES": cfr S20. Nous recréons ensuite "LANGAGE" par la commande CREATE BRANCH; cfr C23. Nous obtenons l'arborescence décrite P22.

6.8 Type de commande: à HALT.

Voir type de commande à HALT du langage de manipulation général.

```
*àcf"ordinateur"
 CREATE COMPLETED
*àcn"hardware""ordinateur"
 CREATE COMPLETED
*àcl"i/o""hardware" w5 sd14 sd2 5 sqn Ø sc1
 CREATE COMPLETED
```

Fig. C20

```
*àcb"hardware"
*Ø1 l"cpu"w5 sc1 sd1 4 sd2 5 sqn Ø
 CREATE COMPLETED
*Ø1n"memoires"w3Ø
 CREATE COMPLETED
*Ø2l"mem centrale"w4 sc 1 sqn Ø sd13 sd2 4
 CREATE COMPLETED
*Ø2l"mem auxilaire"w3 sd1 3 sd2 4 sqn Ø sc 1
 CREATE COMPLETED
*end
```

Fig. C21

```
*àcb"ordinateur"
*Ø1 "software" w 4Ø
 ERROR: PARAMETER IDENTIFIER AFTER CREATE IDENTIFIER
*Ø1 n "software" w 4Ø
 CREATE COMPLETED
*Ø2 l "logiciels" w 5 sd1 4 sd2 5 sqn Ø sc1
 CREATE COMPLETED
*Ø1 n "langages" w1Ø
 CREATE COMPLETED
*Ø2 l "cobol" w 2 sd1 3 sd2 4 sqn Ø sc 1
 CREATE COMPLETED
*Ø2 l "fortran" w 2 sd1 3 sd2 4 sqn Ø scale 1
 CREATE COMPLETED
*end
```

Fig. C22

```
àpa w sd1 sd2 sqn sc
ØØ ORDINATEUR                    1
   Ø1 HARDWARE                     2
                                     WEI ØØØØØ
      Ø2 I/O                         3
                                        WEI ØØØØ5
                                        SCA Ø1
                                        SQN ØØØØØ
                                        SD1 ØØØØ4
                                        SD2 ØØØØ5
      Ø2 CPU                         3
                                        WEI ØØØØ5
                                        SCA Ø1
                                        SQN ØØØØØ
                                        SD1 ØØØØ4
                                        SD2 ØØØØ5
      Ø2 MEMOIRES                    2
                                        WEI ØØØ3Ø
         Ø3 MEM CENTRALE              3
                                           WEI ØØØØ4
                                           SCA Ø1
                                           SQN ØØØØØ
                                           SD1 ØØØØ3
                                           SD2 ØØØØ4
```

Fig. P20

```
Ø3 MEM AUXILAIRE                3
                                  WEI ØØØØ3
                                  SCA Ø1
                                  SQN ØØØØØ
                                  SD1 ØØØØ3
                                  SD2 ØØØØ4
Ø1 SOFTWARE                     2
                                WEI ØØØ4Ø
   Ø2 LOGICIELS                   3
                                    WEI ØØØØ5
                                    SCA Ø1
                                    SQN ØØØØØ
                                    SD1 ØØØØ4
                                    SD2 ØØØØ5
Ø1 LANGAGES                     2
                                WEI ØØØ1Ø
   Ø2 COBOL                       3
                                    WEI ØØØØ2
                                    SCA Ø1
                                    SQN ØØØØØ
                                    SD1 ØØØØ3
                                    SD2 ØØØØ4
   Ø2 FORTRAN                     3
                                    WEI ØØØØ2
                                    SCA Ø1
                                    SQN ØØØØØ
                                    SD1 ØØØØ3
                                    SD2 ØØØØ4
PRINT COMPLETED
```

Fig. P20 (suite)

```
àmm "software" by "os"
MODIFY COMPLETED
```

Fig. M20

```
àa "fortran" leaf by node
ALTER COMPLETED
```

Fig. A20

```
*àp a
 ØØ ORDINATEUR                          1
    Ø1 HARDWARE                          2
       Ø2 I/O                             3
       Ø2 CPU                             3
       Ø2 MEMOIRES                        2
          Ø3 MEM CENTRALE                    3
          Ø3 MEM AUXILAIRE                   3
    Ø1 OS                                2
       Ø2 LOGICIELS                       3
    Ø1 LANGAGES                          2
       Ø2 COBOL                           3
       Ø2 FORTRAN                         2
 PRINT COMPLETED
```

```
*àp "os" w sd1 sd2
 Ø1 OS                                  2
                                          WEI ØØØØ5
    Ø2 LOGICIELS                          3
                                             WEI ØØØØ5
                                             SD1 ØØØØ4
                                             SD2 ØØØØ5
 PRINT COMPLETED
*
```

Fig. P21

```
*às "langages"
 SUPPRESS COMPLETED
```

Fig. S20

```
àmpn"os" w 5
MODIFY COMPLETED
```

Fig. M21

```
*àcb"os"
*Ø1 n "langages"
 CREATE COMPLETED
*Ø2 l "cobol" w 2 sqn Ø sc 1 sd1 3 sd2 4
 CREATE COMPLETED
*Ø2 l "fortran" w 2 sqn Ø sc 1 sd1 3 sd2 4
 CREATE COMPLETED
*end
```

Fig. C23

```
àpa w sd1 sd2 sc sqn
ØØ ORDINATEUR                    1
   Ø1 HARDWARE                     2
                                     WEI ØØØØØ
      Ø2 I/O                         3
                                        WEI ØØØØ5
                                        SCA Ø1
                                        SQN ØØØØØ
                                        SD1 ØØØØ4
                                        SD2 ØØØØ5
      Ø2 CPU                         3
                                        WEI ØØØØ5
                                        SCA Ø1
                                        SQN ØØØØØ
                                        SD1 ØØØØ4
                                        SD2 ØØØØ5
      Ø2 MEMOIRES                    2
                                        WEI ØØØ3Ø
         Ø3 MEM CENTRALE              3
                                           WEI ØØØØ4
                                           SCA Ø1
                                           SQN ØØØØØ
                                           SD1 ØØØØ3
                                           SD2 ØØØØ4
```

Fig. P22

```
Ø3 MEM AUXILAIRE                    3
                                      WEI ØØØØ3
                                      SCA Ø1
                                      SQN ØØØØØ
                                      SD1 ØØØØ3
                                      SD2 ØØØØ4
Ø1 OS                               2
                                    WEI ØØØØ5
  Ø2 LOGICIELS                        3
                                        WEI ØØØØ5
                                        SCA Ø1
                                        SQN ØØØØØ
                                        SD1 ØØØØ4
                                        SD2 ØØØØ5
  Ø2 LANGAGES                         2
                                        WEI ØØØØØ
    Ø3 COBOL                            3
                                          WEI ØØØØ2
                                          SCA Ø1
                                          SQN ØØØØØ
                                          SD1 ØØØØ3
                                          SD2 ØØØØ4
    Ø3 FORTRAN                          3
                                          WEI ØØØØ2
                                          SCA Ø1
                                          SQN ØØØØØ
                                          SD1 ØØØØ3
                                          SD2 ØØØØ4

PRINT COMPLETED
*
```

Fig. P22 (suite)

```
àe

 OPEN ELE
 CREATION (Y) OR MODIFY (N)
*y
 FIRST CREATION (Y OR N )
*y
 SPECIFY NAMES OF PROJECTS BETWEEN QUOTE
*"ibm""siemens""dec"
 PROJECT ORDER IS : IBM
                    SIEMENS
                    DEC


 I/O

*13 12 14
 IBM
 00013
 SIEMENS
 00012
 DEC
 00014

  CREATE OF VALUES COMPLETED


 CPU

*13 13 15
 IBM
 00013
 SIEMENS
 00013
 DEC
 00015

  CREATE OF VALUES COMPLETED


 MEM CENTRALE

*14 13 12
 IBM
 00014
 SIEMENS
 00013
 DEC
 00012

  CREATE OF VALUES COMPLETED
```

Fig. E20

```
MEM AUXILAIRE

*12 13 11
 IBM
 ØØØ12
 SIEMENS
 ØØØ13
 DEC
 ØØØ11

  CREATE OF VALUES COMPLETED


LOGICIELS

*15 12 14
 IBM
 ØØØ15
 SIEMENS
 ØØØ12
 DEC
 ØØØ14

  CREATE OF VALUES COMPLETED


COBOL

*1Ø 1Ø 15
 IBM
 ØØØ1Ø
 SIEMENS
 ØØØ1Ø
 DEC
 ØØØ15

  CREATE OF VALUES COMPLETED


FORTRAN

*12 13 14
 IBM
 ØØØ12
 SIEMENS
 ØØØ13
 DEC
 ØØØ14

  CREATE OF VALUES COMPLETED


ENTRY TERMINATED
```

Fig. E20 (suite)

```
*àw"ibm""siemens"

 DATA BASE IS OPEN

 CONCORDANCE
*.6543.5000.4321
 I/O
 CPU
 MEM CENTRALE
 MEM AUXILAIRE
 LOGICIELS
 COBOL
 FORTRAN

 FIN-WRITE

  1:0000 EXEMPLE UTILISATION ELECTRE
  2:0000 02
  3:0000 IBM
  4:0000 SIEMENS

  1:0000 007
  2:0000 I/O                  010001300012
  3:0000 CPU                  010001300013
  4:0000 MEM CENTRALE         010001400013
  5:0000 MEM AUXILAIRE        010001200013
  6:0000 LOGICIELS            010001500012
  7:0000 COBOL                010001000010
  8:0000 FORTRAN              010001200013
```

Fig. W20

```
1:0000 .6543.5000.4321
2:0000 0000000004000050000 5
3:0000 0000000004000050000 5
4:0000 0000000003000040000 4
5:0000 0000000003000040000 3
6:0000 0000000004000050000 5
7:0000 0000000003000040000 2
8:0000 0000000003000040000 2
```

Fig. W20 (suite)

CHAPITRE VII INTERFACE ENTRE LA BASE DE DONNEES GENERALE ET LES BASES DE DONNEES SPECIFIQUES.

A. Syntaxe de l'ordre: à RESUME.

[b]. à [b]. R[ESUME] [b]. { A[LL] / numéro(s) de version }

Les numéros de version doivent être séparés par au moins un espace.

B. Sémantique et fonctionnement.

Le transfert entre la base de données générale et les bases de données spécifiques se réalise grâce à l'utilisation d'un ordre: àRESUME.

Le paramètre standard ALL permet de transférer tous les libellés associés aux critères de la base de données initiale vers la base de données terminale. Dans ce cas, la base de données spécifique doit évidemment être vide.

Lorsqu'un numéro de version intervient comme paramètre spécifique de l'ordre RESUME, il permet le transfert des libellés dont le n° de version est le même que celui cité dans l'ordre.

C. Exemple.

Cet exemple n'a pu être réalisé, car certaines difficultés ont été rencontrées avec le système Sphinx lors de l'essai de cet ordre.

CHAPITRE VIII MESSAGES A L'UTILISATEUR.

M1: à PARAMETER ERROR

Le premier caractère (à) identifiant l'ensemble des commandes est incorrect.

M2: COMMAND ERROR: TYPE OF THE COMMAND

Le premier caractère identifiant le type de commande, c.à.d. la première lettre du verbe, est incorrect.

M3: ERROR: PARAMETER IDENTIFIER INCORRECT

Le type de commande est incorrect.

M4: ERROR: TYPE OF PARAMETER IDENTIFIER AFTER ..... IDENTIFIER

Le type du paramètre est incorrect.

M5: ERROR: PARAMETER IDENTIFIER INCORRECT.

Le paramètre standard est incorrect

M6: YOU MUST SPECIFIED AT LEAST 1 NAME OF CRITERIA

Il n'y a pas d'identificateur de critères dans la commande

M7: ERROR: I CAN'NT DISTINGUISH THE GOOD NAMES OF CRITERIA

Le nombre de critères précisé dans la commande n'est pas correct.

M8: ERROR: LENGHT IDENTIFIER

La longueur d'un critère ne peut pas dépasser 27 caractères;

M10: "nom-de-critère" NAME IS INCORRECT

La réalisation de ce critère n'existe pas dans la base de données.

M11: ERROR: NEVEL VALUE

Le niveau doit être en caractères numériques.

M12: ERROR: LENGHT OF NEVEL TOO LONG

La valeur du niveau doit être indiquée sur 2 caractères.

M13: ERROR IN THE STRUCTURE OF BRANCH COMMAND

— La structure de la branche (emboîtement des niveaux) dans la commande à Create Branch, est incorrecte.

M15: YOU ARE AT THE END OF THE AREA

Ceci signifie que la commande n'a pas été écrite entièrement dans les limites prévues (70 caractères, sauf si on utilise le symbole ±, pour continuation de la commande à la ligne suivante).

M16: 2 NAMES ARE NOT PERMITTED IN THIS COMMAND

Lors de la création de la racine du graphe, il faut spécifier un seul libellé de critère

M17: ERROR: WEIGHT VALUE

La valeur du poids n'est pas numérique

M18: WEIGHT VALUE TOO LONG

Le poids est exprimé sur une longueur dépassant 5 caractères

M19: ERROR PARAMETER VALUE

la valeur d'un paramètre n'est pas numérique

M20: ERROR: PARAM VALUE TOO LONG

— le paramètre est écrit sur une longueur supérieur au nombre de caractères autorisé.

M21: ERROR: YOU MUST SPECIFIED THE NAME OF PRECED. CRITERIA

Dans la commande: Create Branch, il faut préciser le critère auquel on veut rattacher une branche.

M22: ERROR: PARAMETER IS DUPLICATED

On ne peut pas créer deux fois le même paramètre spécifique dans la même commande.

M23: ERROR: YOU MUST WRITE THE IDENTIFIER BETWEEN 2 QUOTES

M24: YOU MUST SPECIFIED THE PARAMETER BY BETWEEN 2 IDENTIFIERS

M25: ERROR: YOU MUST SPECIFIED AT LEAST 1 PARAM

La commande exige la spécification d'un paramètre et d'une valeur à lui accorder.

M50: CREATE COMPLETED

M51: MODIFY COMPLETED

M53: DATA BASE IS EMPTY

M54: PRINT COMPLETED

M55: SUPPRESS COMPLETED

M56: YOUR NEW NUMBER OF VERSION IS ...

M57: DATA BASE IS CLOSED

...

Remarque.

Ceci n'est qu'un exemple des messages envoyés à l'utilisateur et n'est donc pas exhaustif. La liste complète de ces messages sera reprise avec un exemple complet, dans les listings joints au mémoire.

CONCLUSIONS.

Nous avons analysé dans ce deuxième tome, la solution proposée dans le dossier de conception. La plus grande partie de cette solution a été réalisée, malgré les ennuis rencontrés tout au long de son élaboration; nous citons entre autres:

- les problèmes rencontrés lors de la réalisation d'une première application concrète (méthodologie de l'Analyse ...)
- des problèmes d'ordre pratique, lors de l'utilisation du système Sphinx
- le manque de documentation (à jour) du SGBD Sphinx; (nous tenons cependant à signaler que cette dernière lacune a été entièrement comblée par l'aide fournie de la part de l'équipe "Grands Fichiers").
- ...

La réalisation de ce mémoire nous a permis d'approfondir nos connaissances dans des domaines bien divers:

- SGBD: Sphinx (compréhension et manipulation)
- technique de programmation particulière (piles,...)
- langage interactif (facilités accordées aux utilisateurs,...)
- problème de choix d'ordinateurs
- ...
- mais surtout, et enfin, la réalisation d'une application concrète.

Il reste bien sûr des améliorations à apporter à notre solution, nous pensons spécialement à

- l'accroissement des facilités apportées à l'utilisateur:
  - en proposant des manipulations supplémentaires (ex: INSERT, DELETE ...)

- en vérifiant les données de manière plus stricte dès leur introduction (ex: somme des poids des critères immédiatement subordonnés = 100; détection de libellés identiques dans l'arborescence...)
- en fournissant une documentation plus facilement manipulable par l'utilisateur (ex: commande *HELP)
- ...

-l'augmentation des performances (temps de réponse...)

- ...

Le lecteur trouvera la suite du dossier de programmation sous forme de listings comportant notamment

- les programmes
- la description de ces programmes
- des prolongements réalisables...

BIBILIOGRAPHIE.

B1: J. Ph. Arnotte, Evaluation de systèmes complexes, choix d'un ordinateur par la méthode Cat (2 tomes), Mémoire de licence et maîtrise en Informatique, F.N.D.P., Namur, 1977.

B2: A. Nottebart, Analyse Multicritère, Mémoire de licence en sciences Mathématiques, F.N.D.P., Namur, 1977.

B3: F. Bodart, Méthodes d'Analyse fonctionnelle, cours, Institut d'informatique, Namur, 1978.

B4: J. Ramaekers, Efficacité des systèmes informatiques, cours, F.N.D.P., Namur, 1978.

B5: Ph. Van Bastelaer, Séminaire de Design de Configuration, cours, F.N.D.P., Namur, 1978.

B6: J. Fichefet, Théorie des graphes, cours, F.N.D.P., Namur, 1977.

B7: B. Roy, La méthode Electre II, une application au média planning, M. Ross, O.R. 1972, North-Holland publishing Company (1973).

B8: B. Roy, Vers une méthodologie générale d'aide à la décision, Revue Metra, vol XIV n° 3, 1975.

B9: Ph. Van Bastelaer, Essai de questionnaire pour un choix de configuration, FNDP, Namur, 1976.

B10: C. Deheneffe, J.L. Hainaut, H. Hennebert, B. Le Charlier, W. Paulus, Système de conception et d'exploitation d'une base de données, projet de recherche C.I.P.S. n° I.2/15, F.N.D.P., Namur, 1974.

B11: Colussi, Deheneffe, Guillebaud, Hainaut, Hennebert, Le Charlier, Paulus, Système de conception et d'exploitation de bases de données, projet de recherche C.I.P.S. n° I.2/15, F.N.D.P., Namur, 1978.

B13: B. Roy, Décision avec critères multiples: problèmes et méthodes, Revue Métra vol. XI, n° 1, 1972.

B14: B. Roy, La modélisation des préférences: un aspect crucial de l'aide à la décision, Revue Metra, vol XIII, n° 2, 1974.

B15: J-P. Laloux, Implémentation du modèle d'accès par lui-même, Mémoire de licence et maîtrise en Informatique, F.N.D.P., Namur, 1976.

B16: J. Fichefet, Critique de la méthode "MAL" de J.J. Dujmovic pour l'évaluation et la comparaison de systèmes complexes, FNDP, Namur,1978.

B17: A. Clarinval, Cours de méthodologie de l'Analyse et de la Programmation, F.N.D.P, Namur, 1977.

B18: P. Berthier, Analyse des données multidimensionnelles, Paris: P.U.F., 1975.