Le systeme SICLA: Principes et architecture

Yves Lechevallier, Henri Ralambondrainy, Gérard Govaert

INRIA

CENTRE DE ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France

Tél. : (1) 39 63 55 11

Rapports de Recherche

N° 500

LE SYSTÈME SICLA : PRINCIPES ET ARCHITECTURE

Yves LECHEVALLIER
Henri RALAMBONDRAINY
Gérard GOVAERT

Mars 1986

LE SYSTEME SICLA : PRINCIPES ET ARCHITECTURE

Yves LECHEVALLIER - Henri RALAMBONDRAINY
INRIA
Domaine de Voluceau
BP 105 - Rocquencourt
78153 Le Chesnay Cedex - France

Gérard GOVAERT
Université de Metz
LRIM

Résumé

Le logiciel SICLA est un système interactif d'Analyse de Données développé dans le projet "Classification et Reconnaissance des Formes". Nous présentons les principes du système et la méthodologie de conception.

Abstract

The software SICLA is an interactif system for Data Analysis, developped in the project "Classification et Reconnaissance des Formes". The system design and the methodology are presented.

SOMMAIRE

I INTRODUCTION

L'Analyse des Données (A.D.) est fondée sur l'usage de l'ordinateur, ses développements n'ont été possibles que grâce aux progrès de l'informatique.

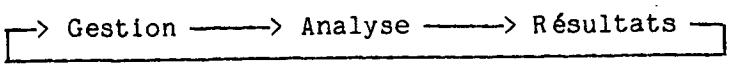
Le logiciel occupe donc une place centrale dans cette science, il n'est pas une simple transcription informatique d'un algorithme mathématique mais fait partie intégrante, selon nous, de la démarche même de l'A.D.. Proposer un programme ou un système A.D., ce n'est pas seulement offrir un outil informatique de calculs mais c'est proposer une méthodologie, une démarche pour traiter des données. Malheureusement, si l'on étudie les logiciels statistiques existants, que ce soient les "grands" systèmes connus comme BMDP, SPSS OU SAS ou ceux plus récents relatifs aux micro-ordinateurs, on s'aperçoit que les qualités mises en avant sont de type "informatique" : puissance du langage de commande, capacité de gestion, interactivité, etc, ...

La spécificité statistique du système venant simplement du nombre d'algorithmes statistiques fournis. Il apparaît alors que les concepteurs de logiciel ou les utilisateurs, considèrent un système statistique comme un produit informatique donnant l'accès à un ensemble de méthodes statistiques, la qualité du logiciel étant mesurée par ses capacités de gestion, la mise en forme préparatoire des données, ses possibilités graphiques d'édition de résultats et ses performances de calcul. Pour s'en convaincre, il suffit de considérer l'enquête faite par N. Lauro et I. Francis [FrL 82] concernant un ensemble de logiciels statistiques.

Des utilisateurs et des concepteurs sont amenés à juger un certain nombre de produits sur leurs fonctionnalités. L'examen des questions de l'enquête montre que la plupart concernent la capacité de gestion, d'édition, de mise en oeuvre, de portabilité, de maintenance, etc, ... du logiciel. Qualités générales que l'on attend de tout produit informatique, quelque soit le domaine considéré. L'aspect statistique n'apparaissant que sur le domaine d'application du logiciel : statistique en général ou orientation sur des problèmes spécifiques.

Selon nous, un système statistique est plus, qu'un moyen d'accéder de manière plus ou moins performante à des analyses statistiques. L'approche proposée aux utilisateurs, pour traiter leurs données, ne doit pas se résumer au schéma : Gestion ⟶ Analyse ⟶ Edition.

En effet, une des caractéristiques des méthodes d'A.D. est de fournir des structures-résumées (partition, coordonnées factorielles, ...) qui peuvent être réutilisées par d'autres méthodes. Un système d'A.D. doit donc permettre de "reboucler" c'est-à-dire que le schéma doit se présenter comme suit :



De manière générale, l'A.D. est, selon nous, une modélisation du "réel", ce modèle consistant en un ensemble d'objets mathématiques censés représenter la réalité et un ensemble d'algorithmes opérant sur ces objets, le tout étant coordonné par des stratégies dans la mise en application des algorithmes (cf. figure I.1).

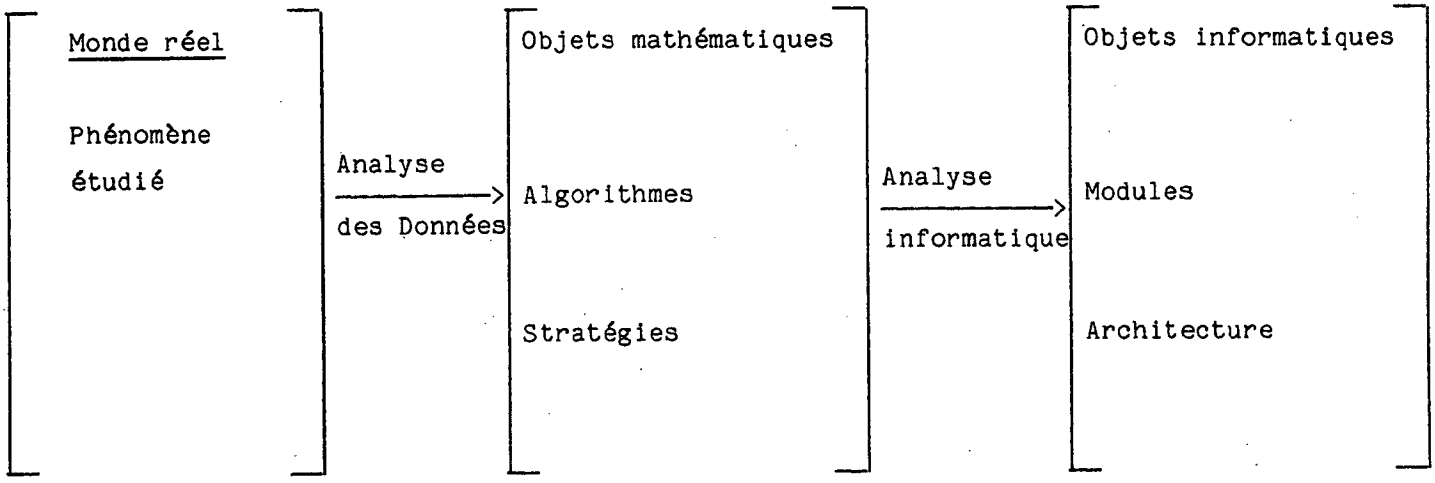


Figure I.1 : Schéma fonctionnel

Les travaux théoriques en A.D. consistent en l'étude des propriétés de ces objets (tableaux individus * variables, tableaux de distances, ...) et en la recherche de techniques de réduction de l'information relative à ces objets [Did 79] [Ben 73] et de méthodologies pour la mise en oeuvre de différents algorithmes d'étude de données.

Un système d'A.D. est alors formé d'un ensemble d'objets informatiques, représentant des objets mathématiques, de modules informatiques relatifs à la gestion et l'analyse de ces objets et d'un ensemble de règles (architecture) régissant la cohérence de l'ensemble. Sur ces aspects, on constate une faiblesse des systèmes existant. Considérons par exemple l'objet de base de l'Analyse des Données qui est le tableau individus * variables. Le type statistique d'une variable (type mesure, binaire, qualitatif ...) est une notion importante en A.D., car il n'existe pas de méthode universelle, mais des méthodes plus ou moins bien adaptées à des tableaux dont les variables sont d'un certain type. Les logiciels existant ne tiennent compte que d'un ou deux types de variables : (quantitatif et/ou qualitatif), aucun contrôle n'est en général fait sur l'adéquation entre type de données et les méthodes. De même, les structures résultats ne sont sauvegardées qu'accessoirement à des fins d'édition mais aucune réflexion n'est faite sur leur réutilisation par d'autres modules. Par exemple, dans le logiciel SPAD, les coordonnées factorielles sont sauvegardées mais la solution adaptée n'est pas satisfaisante car orientée vers une utilisation graphique, un module spécial est nécessaire si l'on veut effectuer une classification automatique sur ces coordonnées. La gestion de l'information dans les systèmes classiques est simple puisqu'il n'existe, d'une part que peu de structures de type différent et d'autre part, que la stratégie est toujours linéaire. Notre démarche conduit à un système plus complexe qui propose une représentation générale des structures statistiques existantes et, qui gère la cohérence de l'information entre les objets et modules, et les liens logiques pouvant exister entre ces objets. [cf. figure I.2].

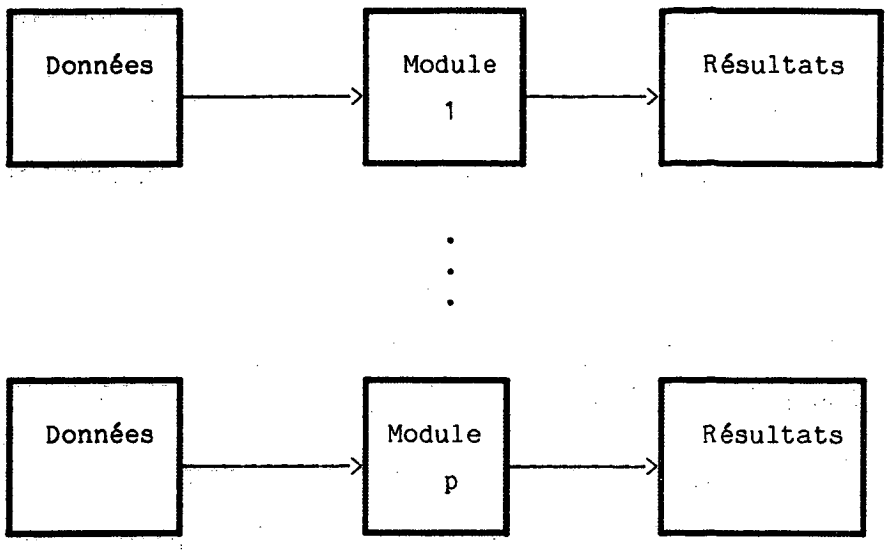
Dans cette partie, nous décrivons l'architecture et les principes du système SICLA. Elle est fondée sur l'analyse des structures et des traitements effectués en A.D.. Nous exposons les représentations informatiques des objets, les différents types de modules et les principes régissant l'architecture. Les objectifs que nous nous sommes fixés sont :

- réaliser un système "intelligent" c'est-à-dire qui garantit le bon emploi de l'information (protection contre l'utilisation abusive de méthodes, par exemple) et assurant le maintien de la cohérence et l'intégrité des données.

- fournir un produit informatique de bonne qualité : convivial , l'accent étant mis sur l'aspect interactivité, portable à des fins de diffusion, et performant pour le traitement des données

- proposer un ensemble de méthodes représentatives du domaine de la classification automatique.

Le système a été conçu de manière modulaire afin de faciliter les extensions et le langage utilisé est le FORTRAN 77.



Architecture d'un système classique

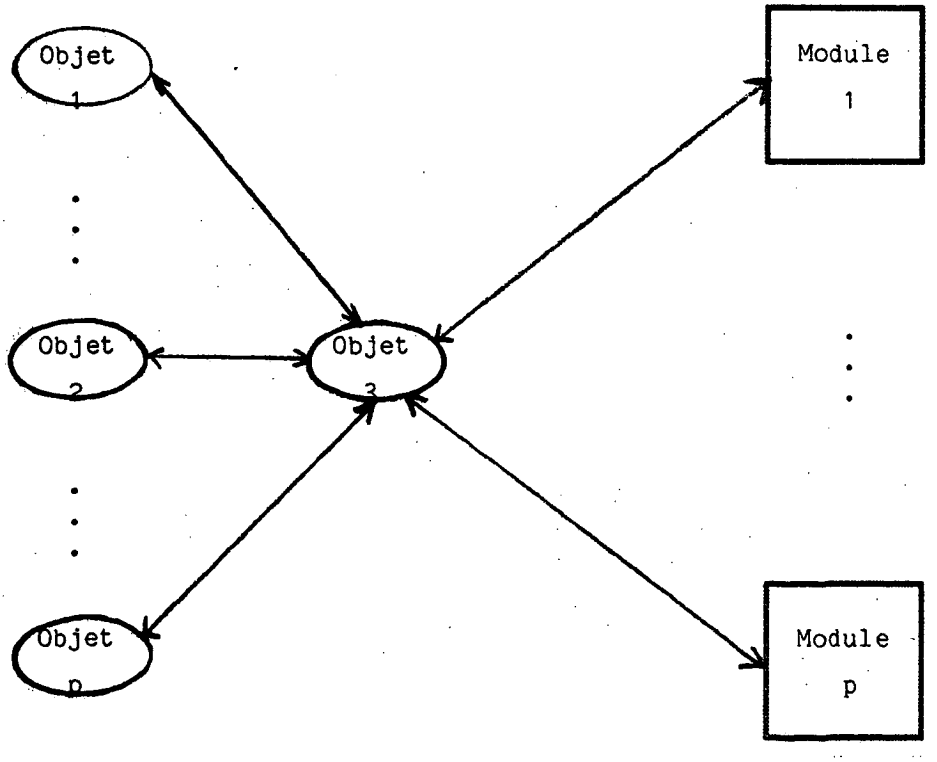


Figure I.2 : Architecture d'un système intégré

II PRESENTATION GENERALE DU SYSTEME

II.1 L'environnement

Le système SICLA présente plusieurs interfaces avec d'autres logiciels qui sont (cf figure II.1.1)

- une interface avec un système de base de données relationnel PEPIN [JKR 85] permettant une gestion complexe de l'information.

- une interface avec un système expert CLAVECIN [DQR 85] pour une assistance intelligente au système.

- une interface avec un logiciel graphique en Analyse des Données respectant les normes GKS (Rak 84).

- une interface avec le logiciel SPAD.

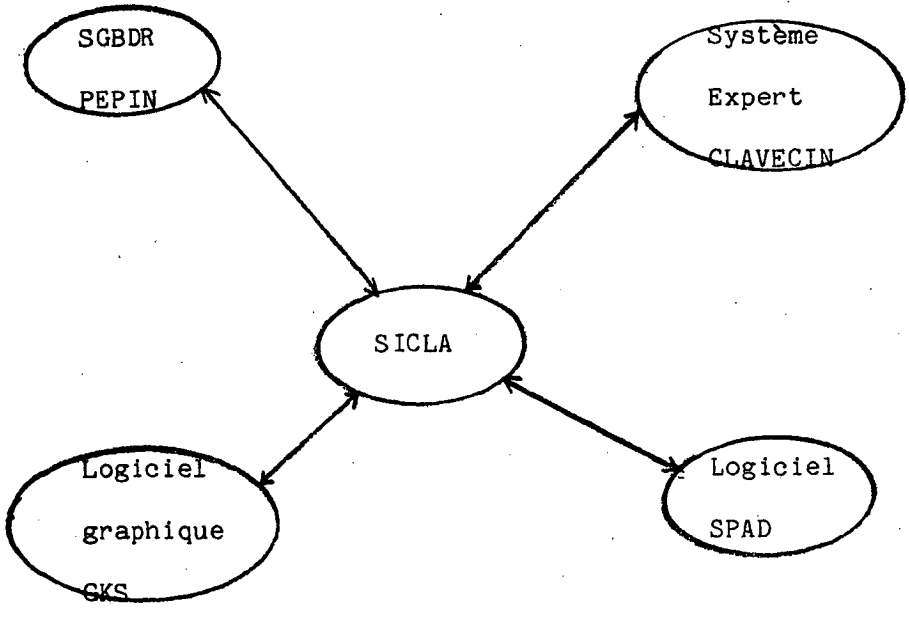


Figure II.1.1 : Environnement de SICLA

II.2 Les différents aspects du système

Dans notre présentation, nous considérons le système SICLA sous les deux points de vue suivants :

1) Point de vue : outil d'Analyse de Données :

C'est, en effet, la première fonction du logiciel. Le système assure le recueil de l'information relative aux données de l'utilisateur, son stockage et son traitement à des fins de gestion ou d'analyse statistique. Le système est alors étudié comme un système de gestion et de traitement de l'information.

Ainsi plusieurs modules permettent la réalisation des fonctions suivantes :

a) enregistrement et contrôle des données
b) description élémentaire (calcul d'histogrammes par exemple)
c) analyse des données par les méthodes multidimensionelles (Classifications, Analyse discriminante, Méthodes factorielles, etc,...)
d) gestion et édition des données et des structures résultats

2) Point de vue : environnement de programmation :

Un autre aspect important du système est de fournir un certain nombre d'outils et un cadre méthodologique pour la production de programmes d'A.D.. En se fondant sur l'analyse d'un programme type d'A.D., un schéma général est proposé et différents utilitaires de gestion de ressources communes (mémoire centrale, secondaire, ...) sont disponibles. La description de cette bibliothèque est fournie dans le manuel du programmeur. Nous décrivons, ici, simplement l'architecture interne du système et la manière dont il gère ces ressources.

III PRINCIPES ET CONCEPTS

III.1 Concepts de base

Nous appellerons "objets du système", la représentation informatique de structures d'A.D.. On appelle Structure de Données (S.D.), celles susceptibles d'être analysées par des méthodes statistiques (par exemple, le tableau individus * variables, les tableaux de distances). On appelle Structures Résultats (S.R.) celles issues de ces analyses. (Par exemple, la structure partition ou coordonnées factorielles). Un des objectifs principal d'une méthode d'A.D. est, la condensation des données brutes en des structures "résumées" mettant en évidence les relations entre objets ou variables.

Les logiciels statistiques existant s'intéressent en général peu aux résultats d'analyses statistiques en tant que structure. Si ces résultats sont sauvegardés, ils le sont essentiellement à des fins graphiques.

Une partition, par exemple, ne sera pas conservée en tant que S.D. mais sera directement intégré au tableau de données comme variable qualitative. On perd alors diverses informations comme le critère mesurant la qualité de la partition et il n'est pas possible de manipuler des ensembles de partitions. Dans le logiciel SPAD, les coordonnées factorielles sont sauvegardées en tant que S.R. à des fins graphiques. Il est cependant possible d'exécuter un module de classification sur cette structure. Cette confusion entre S.D. et S.R. conduit à avoir des modules spéciaux de classification automatique.

Nous considérerons conceptuellement comme fondamentalement différent une S.D. et une S.R.. Il sera toutefois possible de transformer une S.R. en une S.D. mais cela se fera par une opération explicite et dans des conditions précises.

1) Notion de modules : les différentes classes de modules

Un "module" sera une entité informatique se présentant sous forme de programme ou sous-programme exécutant une fonction précise sur une S.D. ou S.R.. Suivant ces fonctions, on distingue les classes de modules suivant (cf. figure III.1.1)

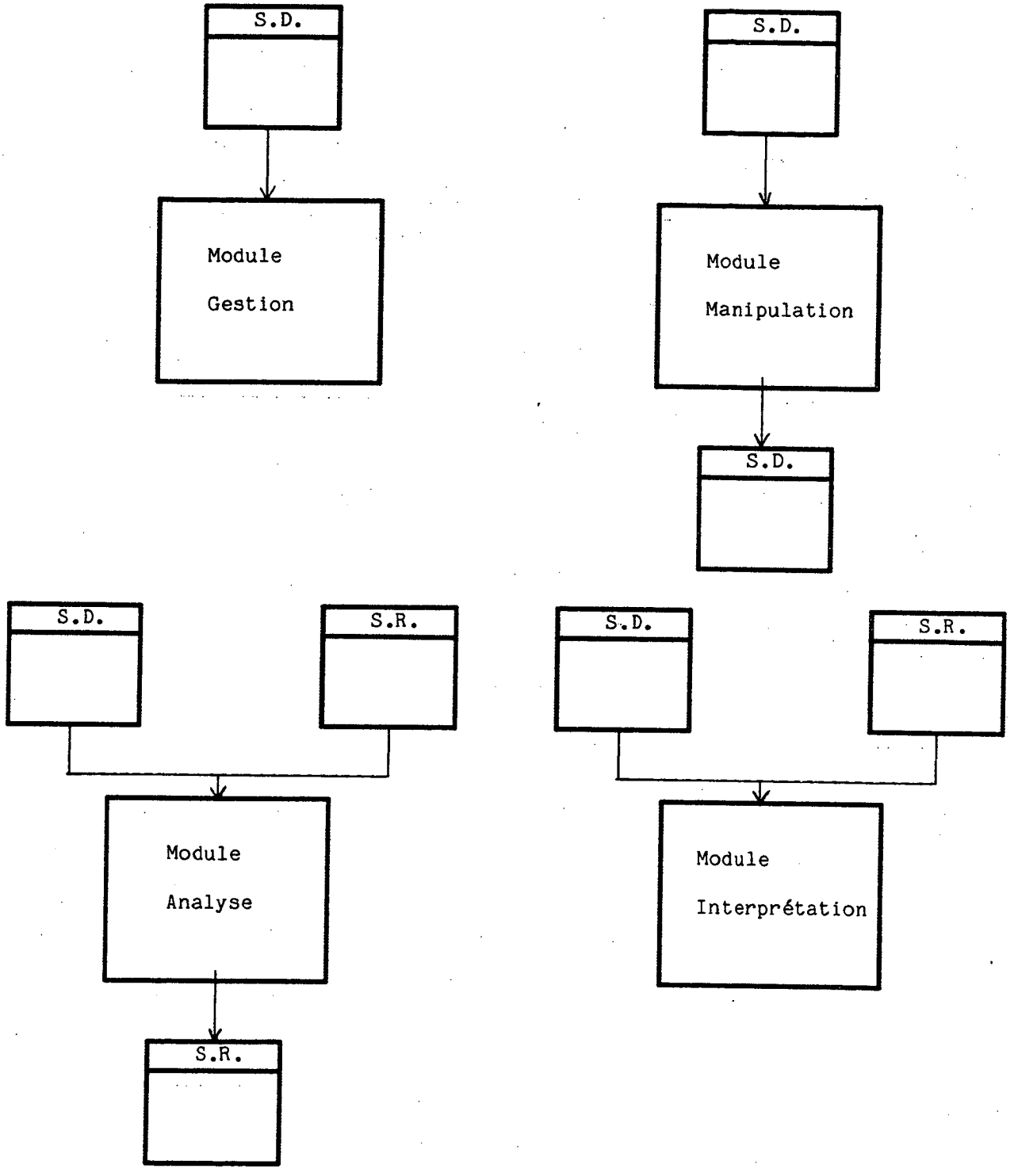


Figure III.1.1 : Les différentes classes de modules

classe I : modules de gestion :

ce sont les modules opérant sur une S.D. pour effectuer diverses opérations de modifications (mise à jour d'identificateurs de variables par exemple) de créations de variables, etc, ...

classe II : modules de manipulation :

ces modules comportent en entrée une ou plusieurs S.D. ou une S.R. et génèrent une S.D.. Les modules de sélection, par exemple, créent de nouvelles S.D. à partir de sous-tableaux de lignes ou de colonnes d'une S.D. d'origine. Ceux de concaténation génèrent une S.D. unique à partir de deux S.D. d'entrée.

classe III : modules d'analyses statistiques :

ce sont les programmes correspondant aux méthodes d'A.D.. Ils ont en entrée une S.D. et une S.R. éventuelle (configuration initiale constituée par une partition, par exemple) et génère une S.R. De plus ils ne peuvent modifier le S.D.

classe IV : modules d'interprétation et d'édition :

ce sont les modules qui à partir d'une S.R. et d'une S.D. éventuelle génèrent divers états de sorties et d'interprétation (par exemple, plans factoriels, description de partitions).Ils ne modifient pas le S.D. et S.R.

L'ensemble de ces modules que l'on appellera aussi "programmes d'application", s'appuient sur un ensemble de routines communes assurant la gestion des ressources physiques et logiques qui constituent le noyau. Le système SICLA peut donc être considéré comme l'ensemble : noyau + programmes d'application, agissant sur un ensemble d'objets selon des règles précises que nous allons exposer. (cf figure III.1.2)

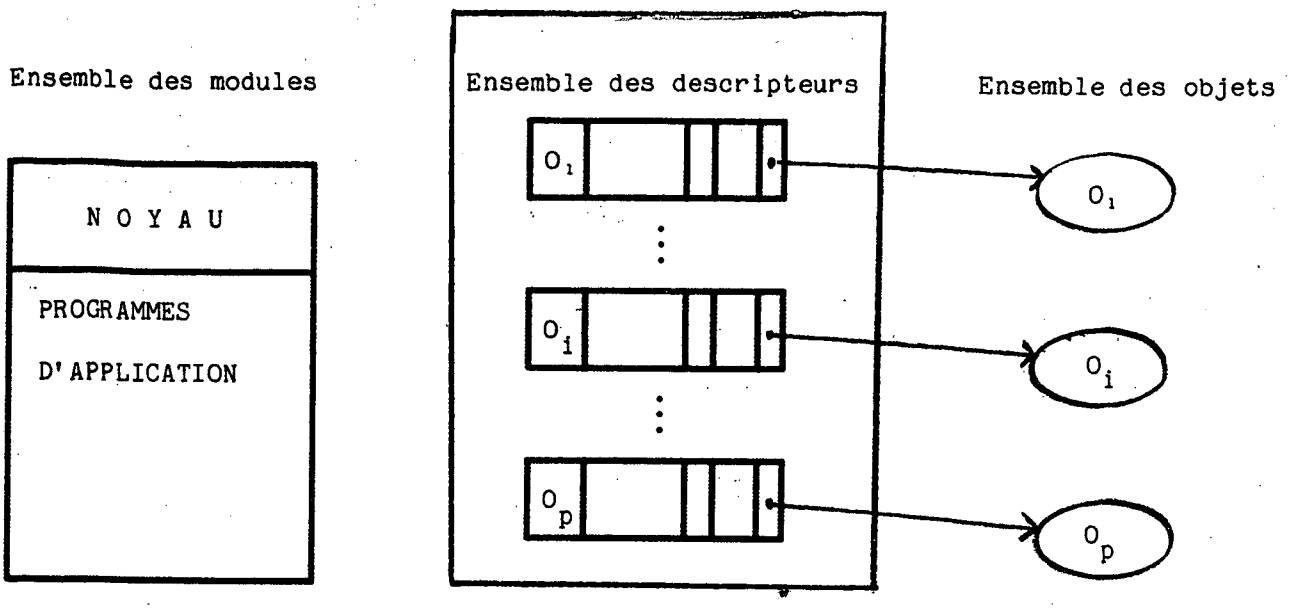


Figure III.1.2): Architecture du système

III.2 Orientations fondamentales

III.2.1 Les mécanismes de protection

a) Le problème:

Un des objectifs fondamentaux d'un système informatique est d'assurer le bon emploi de l'information. Le contrôle du bon emploi de l'information, et d'une manière générale, des ressources d'un système, est étudié sous le terme général de "protection" dans la théorie des systèmes d'exploitation [Cro 75]. La protection n'a pas comme rôle d'empêcher la production d'erreurs (elles sont inévitables) mais d'empêcher leur incidence sur les objets protégés. Ainsi, les mécanismes de protection assurent qu'un processus ne manipule que les objets qu'il a créés et qu'il n'accède pas à ceux d'un autre sans autorisation, ils gèrent la méfiance mutuelle dans le cas du partage d'un objet, etc, ...

- Dans notre contexte, les mécanismes de protection porteront essentiellement sur l'aspect logique du traitement de l'information. Ce seront des règles qui exprimeront l'adéquation modules-objets. Elles permettent de:

1) protéger un objet d'une modification intempestive par un module.
2) éviter l'application de modules sur des objets inadéquats.

b) Notion de descripteur : (cf figure III.2.1)

- Une approche possible pour la résolution du problème de la protection est basée sur la notion de descripteur [Fer 75].

Le descripteur est l'unique voie d'accès à tout objet du système, il contient les informations permettant de caractériser l'objet et les fonctions d'accès aux différents composants. Plus précisément, le descripteur d'un objet comprend :

- son type
- l'identificateur
- des informations générales
- les fonctions d'accès aux composantes de l'objet.

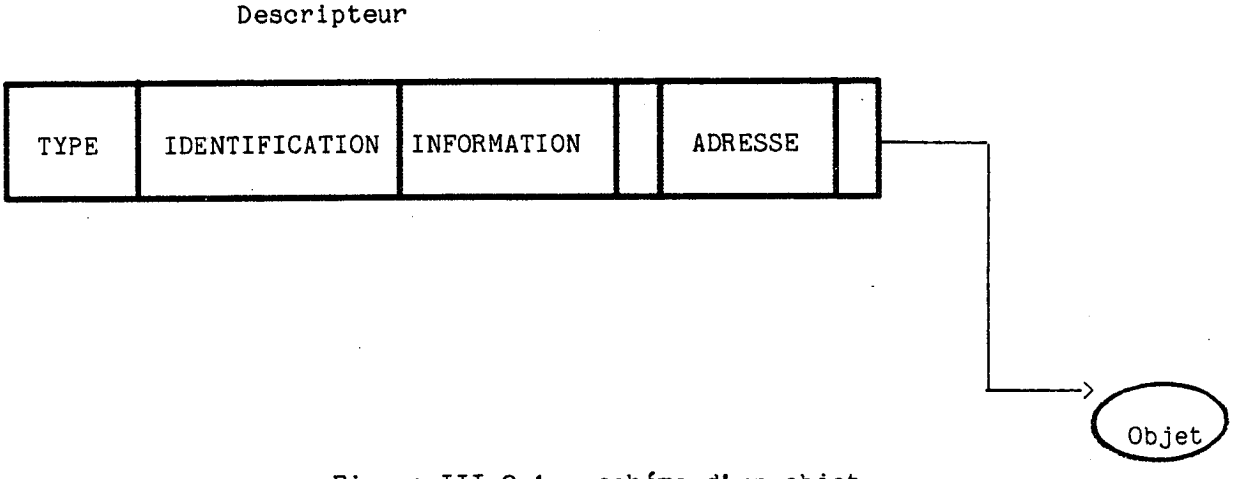


Figure III.2.1 : schéma d'un objet

1) Le type : le type permet de préciser la nature de l'objet et de connaître l'ensemble des modules pouvant opérer sur lui. Les contrôles d'adéquation objet-module seront donc basés sur le type. Ainsi, à partir des classes de modules, on peut construire le tableau (III.2.1) du mode d'accès d'un module et d'une S.D. ou S.R.. Mais, les règles de protection seront résumées plus précisément par la matrice M recensant les droits d'accès d'un module sur un objet (cf. tableau III.2.2). Dans ce tableau, la

case M_{ij} est égale à 0 si le module i peut opérer sur l'objet j et 1, dans le cas contraire. La ligne i décrit l'ensemble des objets sur lesquels peut opérer le module i et la colonne j, l'ensemble des modules pouvant traiter l'objet j.

2) L'identificateur : divers renseignements, titre, date, numéro de version, etc, ... sont utiles à conserver pour identifier l'objet.

3) Des informations générales : suivant le type de l'objet, seront stockées dans le descripteur, elles seront détaillées au paragraphe IV.3.

4) Les fonctions d'accès : un objet sera en général composé de plusieurs zones auxquelles on accèdera par divers pointeurs contenus dans le descripteur.

c) Exemples :

Grâce au descripteur, un module de génération de variables ne pourra pas modifier une S.D. type tableau de distances. Une S.R. partition ne sera accessible que par des modules de classification ou d'édition et d'interprétation. D'une manière générale, de tels mécanismes évitent l'utilisation abusive de méthodes sur des structures inadéquates.

d) La protection physique de l'information :

Chaque S.D. aura un support physique différent. La gestion de l'intégrité physique, la sécurité des données n'est point assurée par le système SICLA mais laissée au soin du système hôte. Il n'existe pas de mécanisme de validation de transaction, comme dans les bases de données, pour assurer la reprise en cas de pannes. Une telle solution qui a l'avantage de la simplicité de mise en oeuvre (ces mécanismes de protection sont coûteux et complexes en B.D.) est sans inconvénient majeur dans le domaine spécifique de l'A.D.. En effet, contrairement aux bases de données commerciales ou de gestion, les données sont stables en A.D., on ne fait pas d'opérations de mise à jour, si ce n'est que pour corriger d'éventuelles valeurs aberrantes.

La possibilité de sauvegarde et de restauration d'une S.D., à partir de fichiers textes, sont les solutions proposées et suffisent pour se protéger contre de tels incidents.

Modules \ Structure	S.D.	S.R.
Gestion	L/E	-
Manipulation	L/E	-
Analyse	L	L/E
Interprétation	L	L

L : accès en lecture

E : accès en écriture

Tableau III.2.1. : Modules et structures

	Objet 1	...	Objet j	...	Objet k
Module 1					
Module i			M_{ij}		
Module p					

M_{ij} = 1 si le module i peut accéder à l'objet j

0 sinon

Tableau III.2.2 : Matrice des droits d'accès objets-modules

Du point de vue du système d'exploitation, une S.D. est un fichier, le S.E. assure, par conséquent, la gestion des accès concurrentiels à cette ressource. De même, les aspects d'autorisation, de droits d'accès à la S.D. sont laissés au soin de l'utilisateur.

III.2.2 Gestion de la cohérence et de l'intégrité des données

a) Le problème :

Aux problèmes de compatibilité de modules et d'objets s'ajoutent ceux relatifs au maintien de la cohérence des informations de la S.D.. Ces questions sont désignées sous le terme de "intégrité sémantique" dans la théorie des bases de données [AdD82] [Gar83]. Dans le modèle relationnel, une S.D. correspond au concept de relation et la notion d'individu est appelée n-uplet et celle de variable est appelée attribut. La cohérence de la base s'exprime alors par un ensemble de contraintes d'intégrité portant sur les attributs et n-uplets. Les types de contraintes d'intégrité sont divers, par exemple, si l'on a une base de données relative à une enquête sur des familles, l'on doit avoir : AGE du PERE > AGE du FILS.

b) Les contraintes d'intégrité relatives à une S.D.

Les contraintes d'intégrité sont un ensemble d'assertions s'exprimant à l'aide de la logique des prédicats du premier ordre. Dans une base de données, elles sont spécifiées par l'utilisateur et dépendent de la nature du problème donné. Dans notre contexte, une S.D. est une représentation informatique d'une structure mathématique, les individus, variables sont considérés comme des objets appartenant à un espace mathématique. Les contraintes d'intégrité dépendant de la "signification" des variables ne sont donc pas prises en compte. Nous considérons uniquement les contraintes d'intégrité relatives aux propriétés mathématiques que doivent vérifier les éléments manipulés. Ce sont essentiellement :

1) l'unicité des identificateurs des objets d'une S.D. : individus, variables, et modalités éventuelles. Cette propriété évite d'avoir les graphiques illisibles issus de méthodes factorielles ou les ambiguités lors de la sélection d'éléments. En effet par exemple des modalités ayant des identificateurs identiques seraient indiscernables sur des plans factoriels relatifs à une Analyse des Correspondances multiples.

2) Les propriétés relatives aux types de S.D. considérés. Ainsi, pour une S.D. individus * variables, les variables auront un type statistique bien défini. Les contrôles porteront sur le domaine de variation de la variable. Une variable de type binaire ne peut prendre que les valeurs 0 ou 1, une variable qualitative doit avoir des modalités exclusives, etc, ... Il est certain que pour certaines structures, il ne sera pas possible de vérifier toutes les propriétés mathématiques. Pour une S.D. relative à un tableau de distances, par exemple, il serait trop coûteux de vérifier l'axiome relatif à l'inégalité triangulaire entre les éléments du tableau.

Ces contraintes d'intégrité sont en particulier vérifiées aux étapes suivantes de traitement des données :

- enregistrement des données dans le système :
  c'est à ce moment, en effet, que l'utilisateur "modélise" ses données c'est-à-dire qu'il spécifie la nature du tableau et les types statistiques des variables pour une S.D. individus * variables. Des vérifications doivent donc être faites sur la conformité des données pour la structure mathématique choisie

- manipulation de S.D. :
  lorsque, par des modules de gestion, l'on modifie ou l'on génère de nouvelles S.D., les contraintes d'unicité d'identificateurs, le respect des propriétés mathématiques des structures doivent être vérifiés. Cela n'est pas nécessaire, si l'on utilise des procédés entièrement automatiques pour générer de nouvelles S.D. (par exemple, module de création de tableaux de distances ou de contingence).

Toutefois, par exemple, pour créer des variables qualitatives, c'est l'utilisateur qui spécifie les conditions logiques relatives aux modalités, le système doit alors vérifier que les conditions sont exclusives et qu'une modalité est prise effectivement par un individu donné.

c) Les liaisons logiques entre les objets : S.D. et S.R.

Lors d'une session, l'utilisateur est amené à créer de nouvelles S.D. par des modules de gestion et des S.R. par des modules méthodes.

Les contraintes d'intégrité inter-relations, dont la mise en oeuvre en B.D. n'est pas toujours facile, permettent d'assurer la cohérence de l'information si l'on modifie, par exemple, une relation fils dérivée d'une relation père initiale. Nous avons déjà fait la remarque qu'en A.D., les S.D. sont stables et que l'on ne fait pas de mise à jour. Nous n'envisagerons donc pas de tels mécanismes de maintien de cohérence. Ce qui par contre, est courant en A.D., c'est la création de nouvelles variables calculées en fonction de celles existantes, la génération de nouvelles S.D. et S.R.. L'aspect historique d'un objet est donc important pour que l'utilisateur puisse se "retrouver" parmi toutes ses structures. Aussi, on conservera systématiquement les expressions logiques ou arithmétiques ayant servi à créer les variables ou les intervalles ayant servi au codage d'une variable. Pour une S.R., on gardera les informations permettant d'identifier la S.D. dont elle est issue et le type de la méthode ayant servi à la calculer. Cela permet de faire des contrôles de cohérence, lors de l'utilisation d'une S.R. par un module. Considérons le cas suivant :

A partir de la S.D. numéro 1, on a obtenu une S.R. partition P par un module M de classification automatique. On désire effectuer une classification sur une S.D. numéro 2 en prenant comme configuration initiale la partition P. Le module M doit alors vérifier que la partition P porte sur le même ensemble d'individus que S.D. numéro 2, cela ne sera possible que si l'on a conservé les identificateurs des individus de S.D. numéro 1 dans la S.R. partition P. De même, pour appliquer la technique des " éléments supplémentaires" où l'on confronte à des fins d'interprétation des S.R. de type partition ou factoriel à des S.D. relatives au même ensemble d'individus mais dont l'ensemble des variables diffère ; des contrôles de cohérence du type précédent sont à effectuer.

III.2.3 Protection des méthodes

Pour un système d'A.D., les contrôles généraux d'adéquation objets-modules portant uniquement sur le type de la S.D. ne suffisent pas. Il est admis actuellement qu'il n'existe pas de méthode universelle traitant n'importe quel tableau Individus * Variables, mais de famille de méthodes plus ou moins adaptées à des tableaux types : de mesures, binaires ou autres. Par exemple, une méthode factorielle comme l'Analyse en Composantes Principales (A.C.P.) est bien adaptée à des tableaux comportant un ensemble de variables quantitatives de type mesures. D'autres méthodes, comme l'Analyse Discriminante Linéaire, nécessitent deux groupes de

variables. Le premier groupe de variables étant la variable qualitative à expliquer et le deuxième groupe, celui relatif aux variables explicatives de type quantitatif. Pour éviter l'utilisation abusive de méthodes, des mécanismes de contrôle sont implantés, basés sur le tableau III.2.3.1 recensant pour chaque module, les types statistiques de variables autorisés. Un tel tableau est fourni par les auteurs de méthodes désirant intégrer leurs modules dans le système SICLA. La notion de groupe est essentielle pour une méthode. Chaque concepteur de programme a la possibilité de définir des ensembles de variables ou individus ayant un même statut (groupe des variables actives, passives, à expliquer, explicatives, individus actives, ...) et un module de mise en forme du système rangera les variables et individus selon les groupes ainsi définis. Une telle S.D. réordonnée sera alors facilement traitable par la méthode.

Au terme de ce paragraphe, nous avons donc introduit les concepts de base de module et objet du système SICLA et décrit les règles (mécanismes de protection, contrôles de cohérence et d'intégrité) régissant l'ensemble. Nous allons maintenant aborder plus en détail l'étude des différents objets : S.D. et S.R. constituant ce que l'on appelle "la structure d'information" du système SICLA.

Groupes de variables

	Groupe 1							Groupe j							Groupe 10						
	Types statistiques 1 2 ... 7							Types statistiques 1 2 ... 7							Types statistiques 1 2...7						
Méthode 1																					
. . .																					
Méthode i											$\delta^i_{j\ell}$										
. . .																					
Méthode p																					

$\delta^i_{j\ell} = \begin{cases} 0 \\ 1 \end{cases}$ suivant que le type statistique ℓ du groupe j est accepté par la méthode i

Tableau III.2.3.1 : Adéquation méthode et type de variables

IV STRUCTURE DE L'INFORMATION (S.I)

L'objectif du S.I. est de recenser les structures statistiques que l'on rencontre en A.D. et de définir leurs structures logiques et informatiques pour pouvoir les manipuler par des modules informatiques.

IV.1 Méthodologie

La conception d'un S.I. s'apparente à celle d'un système de gestion de base de données, nous emprunterons donc à ce domaine divers concepts et nous nous inspirerons pour notre démarche de [Gar83] [AdD82]. Nous rappellerons les principes fondamentaux à respecter pour la réalisation d'un bon S.I. et suivrons le rapport ANSI/X3/SPARC pour la présentation du système.

a) Principes :

Divers travaux en Bases de Données ont mis en évidence les points suivant à respecter pour un S.I.

1) Indépendance entre la structure de stockage et structure de données

2) Indépendance entre les programmes d'application et structure de données

Le respect de ces principes permet de faire des extensions faciles sans de grands bouleversements du logiciel.

b) Les différents niveaux :

Pour présenter le S.I., nous distinguerons les niveaux suivants correspondant aux différents stades de la construction d'un S.I..

1) Analyse du domaine :
Nous étudions les différentes structures statistiques que l'on rencontre en A.D. et leurs caractéristiques.

2) Conceptualisation :
Nous proposons une représentation abstraite des divers structures sans considération de contraintes d'implémentation informatique.

3) Le schéma logique :
Nous précisons l'organisation logique et le mode d'accès à l'information

4) L'implémentation physique :
A ce niveau, nous justifions les différents choix sur les techniques d'accès et l'organisation des fichiers.

IV.2 Le domaine de l'Analyse des Données

L'A.D. est une science datant déjà de plusieurs années. De nombreuses études ont été publiées dans différentes revues et livres de référence. Il est donc possible de recenser les différentes structures existantes. Remarquons, toutefois, que c'est une science en pleine expansion et que des chercheurs proposent constamment de nouveaux concepts ou êtres mathématiques pour appréhender davantage la réalité complexe des données. Toute liste est incomplète et ne reflètera que l'état du domaine à un instant donné.

Nous nous apercevons ici, de l'importance d'avoir une démarche rigoureuse et modulaire. En effet, il sera toujours facile d'ajouter de nouvelles structures statistiques sans grand investissement informatique si les principes d'indépendance entre les aspects logiques et physiques de l'information sont respectés.

Nous présenterons les structures de données sur lesquelles, il est possible d'effectuer des analyses, puis les structures résultats issues de ces dernières.

IV.2.1. Structure Individus * Variables :

L'objet de base de l'A.D. est représenté par une matrice X(n * p) à n lignes et p colonnes :

$$X(n * p) = (x_i^j \ / \ i \in I = [1, \ldots, n] \ ; \ j \in J = [1, \ldots, p])$$

X est considéré comme un ensemble de vecteurs individus :

$$X = (\underline{x}_i \ / \ i \in I) \subset R^J \text{ tel que } \underline{x}_i = (x_i^j \ / \ j \in J)$$

ou un ensemble de variables en considérant le transposé X' :

$$X' = (\underline{x}^j \ / \ j \in J) \subset R^I \text{ tel que } \underline{x}^j = (x_i^j \ / \ i \in I)$$

Au tableau X est associée une métrique M relative aux individus sur l'espace R^J et une métrique D_p relative aux variables sur l'espace R^I. La métrique D_p étant l'ensemble des pondérations relatives aux individus. On dira que l'on a un triplet (X, M, D_p).

Il existe plusieurs types de variables que l'on peut classer dans les catégories suivantes :

1) Variables de type quantitatif, on distingue :

- le type mesure : $\underline{x}^j = (x_i^j \ / \ i \in I \text{ et } x_i^j \in R)$
- le type binaire : $\underline{x}^j = (x_i^j \ / \ i \in I \text{ et } x_i^j \in \{0,1\})$
- le type rang ou note : $\underline{x}^j = (x_i^j \ / \ i \in I \text{ et } x_i^j \in A \subset N)$
- le type comptage : $\underline{x}^j = (x_i^j \ / \ i \in I \text{ et } x_i^j \in N)$

2) Variables de type qualitatif : c'est une application q de l'ensemble I dans un ensemble fini J_q que l'on peut considérer comme un ensemble d'indices $\{1, 2, \ldots, J_q\}$ en représentant comme cela se fait usuellement, l'ensemble et son cardinal par la même lettre.

$$q : I \longrightarrow J_q$$

$$i \longrightarrow q(i) \in \{1, 2, \ldots, J_q\}$$

En A.D., une variable qualitative sera en général traitée comme un tableau X_q de variables binaires, appelées indicatrices, tel que :

$$X_q = (\underline{x}^j \ / \ j \in J_q \text{ avec } x_i^j = \delta^j_{q(i)}, \ i \in I)$$

où δ est le symbole de Kroneker. Les conditions peuvent aussi s'exprimer de manière équivalente dans l'espace R^I :

$$\Sigma \ (\underline{x}^j \ / \ j \in J_q) = \mathbb{1}$$

où $\mathbb{1}$ est le vecteur de R^I dont toutes les coordonnées sont égales à 1.

On dira que la variable qualitative q est ordonnée s'il existe un ordre sur les modalités J_q.

Remarque : La structure variables * individus est la structure transposée de celle individus * variables. Elle sert surtout à optimiser l'éxécution de certaines méthodes.

IV.2.2 La structure Variables * Variables :

Elle est définie à partir d'une fonction f sur un couple de parties de J, J_1 et J_2 éléments de $\mathcal{P}(J)$ où $\mathcal{P}(J)$ est l'ensemble des parties de J. L'expression la plus générale d'une structure Variables * Variables :

$$Z = (Z_{j_1 j_2} = f(j_1, j_2) \;/\; j_1 \in J_1,\ j_2 \in J_2 \;;\; J_1, J_2 \in \mathcal{P}(J))$$

On distingue les structures particulières suivantes :

- X est un ensemble de variables quantitatives centrées :
On choisit $f = D_p$, on a donc les structures variables * variables suivantes relatives aux produits scalaires des variables :

$$Z = X' D_p X = (Z_{jj'} = \langle \underline{x}^j, \underline{x}^{j'} \rangle_{D_p} = \mathrm{cov}(\underline{x}^j, \underline{x}^{j'}) \;/\; j, j' \in J)$$

Z est la matrice de variances-covariances relative à X. Si X' est un ensemble de variables centrées-réduites alors :

$$Z = X' D_p X = (Z_{jj'} = \langle \underline{x}^j, \underline{x}^{j'} \rangle_{D_p} = \mathrm{cor}(\underline{x}^j, \underline{x}^{j'}) \;/\; j, j' \in J)$$

Z est la matrice de corrélation relative à X.

- X_{q_1} et X_{q_2} sont des tableaux de modalités relatives à deux variables qualitatives q_1 et q_2 les lignes correspondantes aux individus et les colonnes aux modalités.

$$Z = X'_{q_1} \mathrm{Id}\, X_{q_2} = (Z_{j_1 j_2} = \langle \underline{x}^{j_1}, \underline{x}^{j_2} \rangle \;/\; j_1 \in J_1,\ j_2 \in J)$$

où Id est la matrice identité alors Z est le tableau de contingence relatif aux variables q_1 et q_2.

- $\{X_{q_1} \;/\; q_1 \in Q_1\}$ et $\{X_{q_2} \;/\; q_2 \in Q_2\}$ sont des ensembles de tableaux de modalités alors :

$$Z = \pi \{X'_{q_1} \text{ Id } X_{q_2} / q_1 \in Q_1, q_2 \in Q_2\}$$

$$= \{Z_{j_1 j_2} = \langle \underline{x}^{j_1}, \underline{x}^{j_2} \rangle / j_1 \in J_{q_1}, j_2 \in J_{q_2}, q_1 \in Q_1, q_2 \in Q_2\}$$

alors Z est un tableau de contingence multiple, juxtaposant les tableaux de contingence relatifs aux variables qualitatives de l'ensemble Q_1 et Q_2.

Lorsque l'on a $Q = Q_1 = Q_2$, alors Z est le tableau de Burt

$$Z = \pi \{X'_q \text{ Id } X_{q'} / q, q' \in Q\} = \{Z_{jj'} = \langle \underline{x}^j, \underline{x}^{j'} \rangle / j, j' \in J\}$$

- Si X est un tableau quantitatif ou qualitatif, on peut définir des structures de tableaux de distances ou ressemblances ou dissemblances :

$$Z = \{Z_{jj'} = \delta(j, j') / j, j' \in J\}$$

avec $f = \delta$ qui peut être soit un indice de distance, de ressemblance ou de dissemblance défini sur $J \times J$.

Symétriquement, en considérant l'ensemble des individus, l'on a :

IV.2.3. La structure Individus * Individus :

Elle est définie à partir d'une fonction f sur l'ensemble produit, qui sera la plupart du temps $I \times I$, dans R^+. La fonction f sera essentiellement un indice de distance, de ressemblance ou de dissemblance. Ainsi, une structure Individus * Individus sera de la forme :

$$Z = \{ Z_{ii'} = \delta(i, i') / i, i' \in I\}$$

où δ est un indice de distance, de ressemblance ou de dissemblance.

Du point de vue structure de l'information, un tableau Variables * Variables et Individus * Individus sont semblables. Les différences ne portent que sur la nature de la fonction f et l'ensemble sur lequel elle est appliquée.

IV.2.4 Les structures résultats : S.R.

A partir des S.D., les méthodes d'A.D. génèrent des S.R. de nature diverses. Ces S.R. dépendent des méthodes et contrairement aux S.D., il n'est pas possible d'en faire une liste exhaustive. On distingue les principales structures suivantes correspondant aux méthodes de type factoriel, classification automatique et hiérarchique.

- La structure factorielle contenant la description des nuages d'individus et variables dans les plans factoriels.

- La structure partition : c'est la structure générée par les méthodes de classification automatique.

- La structure hiérarchique qui contient la description de la hiérarchie obtenue par les méthodes hiérarchiques.

D'autres structures existent : structures de métrique, centre de gravité, ... comme nous l'avons dit dans l'introduction, nous essaierons de donner une représentation informatique générale valable pour les structures existantes, mais pouvant inclure de nouvelles.

IV.3 Aspects conceptuels du système

Une fois recensés les objets du domaine, l'étape suivante est la définition d'un ensemble d'informations exhaustives et non redondantes pour chaque structure afin de pouvoir effectuer les opérations de gestion, calcul et d'édition sur ces objets.

Nous commencerons par les S.D.

a) Les Structures de Données : Individus * Variables

La figure [III.3.1] présente l'organisation et la hiérarchie des éléments d'une S.D.. Elle s'exprime comme suit :

STRUCTURE DE DONNEES (IDENTIFICATION, INDIVIDUS, VARIABLES, DONNEES)

Une S.D. comprend donc quatre constituants principaux divisés eux-même en divers champs. Les constituants principaux sont :

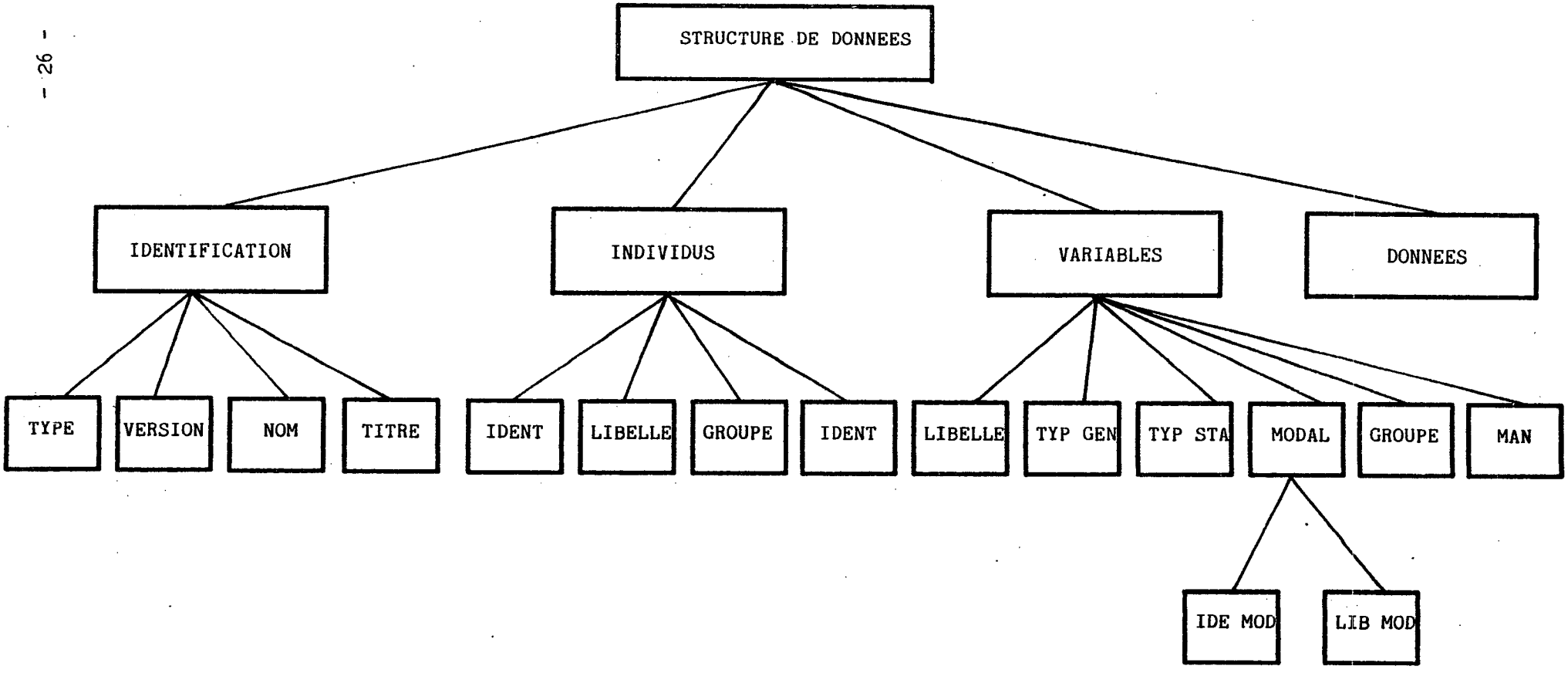


figure III.3.1 : Schéma conceptuel d'une Structure de Données

1) IDENTIFICATION (TYPE, VERSION, NOM, TITRE)
Les significations des divers champs étant :

TYPE : désigne le type de l'objet S.D.

VERSION : est un numéro permettant de distinguer les différentes versions

NOM : identificateur de la S.D.

TITRE : commentaire en clair relatif à la S.D.

2) INDIVIDUS (IDENTIND, LIBELIND, GROUPIND)

Ce constituant décrit l'ensemble des individus :

IDENTIND : identificateur abrégé des individus

LIBELIND : identificateur long ou libellé des individus

GROUPIND : numéro des groupes des individus en vue des méthodes

3) VARIABLES (IDENTVAR, LIBELVAR, TYPE GEN, TYPE STAT, NBRE MOD, IDEN MOD, LIBEL MOD, GROUP VAR, CODMAN)

IDENTVAR : identificateur abrégé des variables

LIBELVAR : identificateur long ou libellé des variables

TYPE GEN : type quantitatif ou qualitatif des variables

TYPE STAT : type statistique des variables (cf. III.3.2)

NBRE MOD : nombre de modalités des variables

IDENT MOD : identificateur abrégé des modalités (variables qualitatives)

LIBEL MOD : identificateur long ou libellé des modalités (variables qualitatives)

GROUP VAR : numéro des groupes des variables (cf. III.2.3)

COD MAN : code des données manquantes pour les variables quantitatives

4) DONNEES : c'est la matrice X (n * p) recensant les valeurs prises par les variables sur les individus.

b) Les Structures de Données : Variables * Variables et Individus * Individus

Nous avons vu que d'une manière générale, ces structures sont relatives aux valeurs d'une fonction f définie sur le produit de deux ensembles $E_1 \times E_2$. E_1 et E_2 pouvant être des ensembles ou sous-ensembles de variables ou d'individus et la fonction f un produit scalaire ou un indice de distances par exemple. La structure d'une S.D. est donc :

STRUCTURE DE DONNEES (IDENTIFICATION, ENSEMBLE 1, ENSEMBLE 2, DONNEES)

le constituant IDENTIFICATION est identique à celui de la structure Individus * Variables et le constituant ENSEMBLE 1 ou ENSEMBLE 2 à celui du constituant INDIVIDUS. Le constituant DONNEES recense les différentes valeurs de la fonction f relative aux deux ensembles.

c) Les Structures Résultats : S.R.

La forme générale d'un objet type S.R. est :

STRUCTURE RESULTAT (IDENTIFICATION, INFORMATIONS, COMPOSANTS)

Les constituants étant :

IDENTIFICATION (TYPE, DATE, NOM S.D., METHODE, NOM, TITRE)

où :

TYPE	: est le type de l'objet S.R.
DATE	: la date de création
NOM S.D.	: nom de la S.D. à partir de laquelle a été calculée la S.R.
METHODE	: numéro de la méthode ayant servi à la calculer
NOM	: identificateur abrégé de la S.R.
TITRE	: nom en clair de la S.R.

Le constituant INFORMATIONS contient des renseignements dépendant de la S.R., il en est de même de COMPOSANTS (COMP1, ..., COMPK) qui recense les différents composants de la S.R.

Exemple : la S.R. type partition

Elle contient les informations suivantes :

- nombre de classes de la partition
- nombre d'objets de la partition
- critère relatif à la partition

et les composants suivants :

- identificateurs des éléments de la partition
- les numéros des classes des individus.

IV.4 Le schéma logique

Tous les objets S.D. ou S.R. ont la même structure logique suivante:

1) un ensemble de tableaux contenant les informations relatives à l'objet

2) un descripteur permettant d'identifier (cf. III.2.2.1) l'objet et d'accéder aux différents tableaux. Il contient les différents dimensions et les pointeurs donnant l'adresse de début de ces tableaux (cf. figure IV.4.1)

Nous avons insisté au paragraphe (III.2.1.) sur la dissymétrie existant entre S.D. et S.R.. On peut ajouter qu'à une S.D. sera associée en général un ensemble de S.R. résultant de plusieurs analyses. Afin d'éviter une multiplicité de fichiers, on regroupera des S.R. dans des objets appelés archive. Une archive pourra, par exemple, contenir l'ensemble des S.R. relatives à l'étude d'une S.D. donnée. Un objet de type archive aura la même structure logique qu'une S.D. ou S.R., il sera formé d'un descripteur et d'un ensemble de zones dont une contiendra l'ensemble des descripteurs des S.R. composant l'archive (cf. figure IV.4.2)

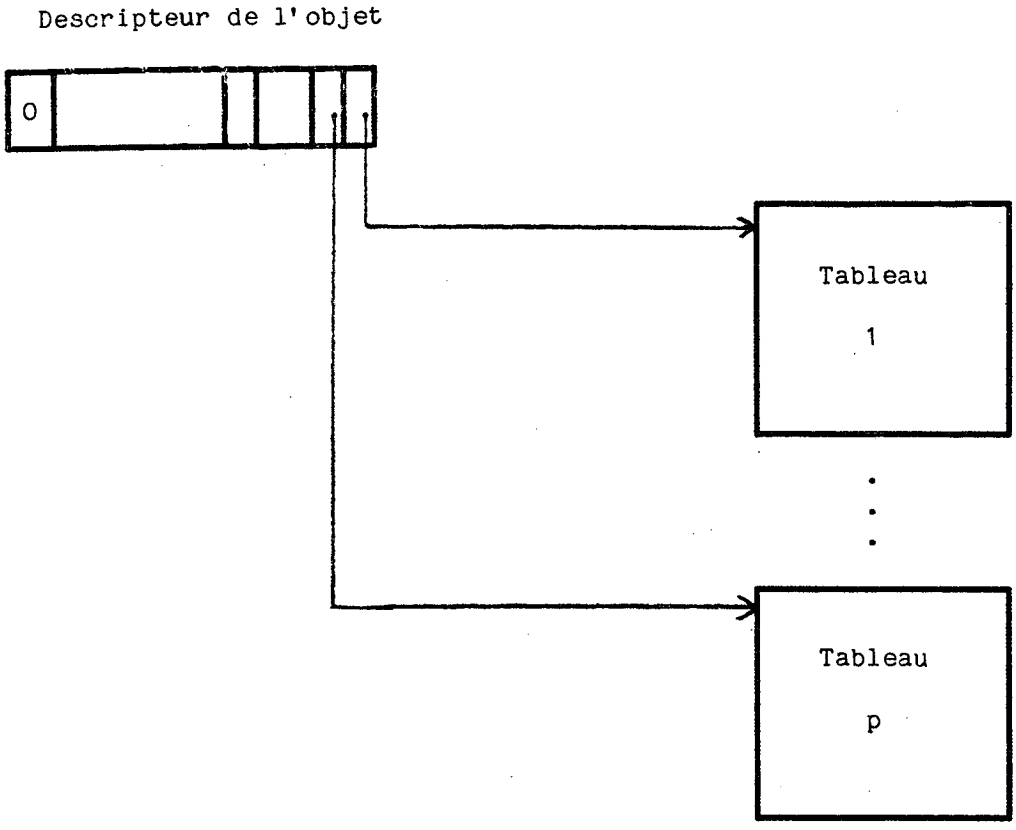


Figure IV.4.1 : Schéma logique d'un objet

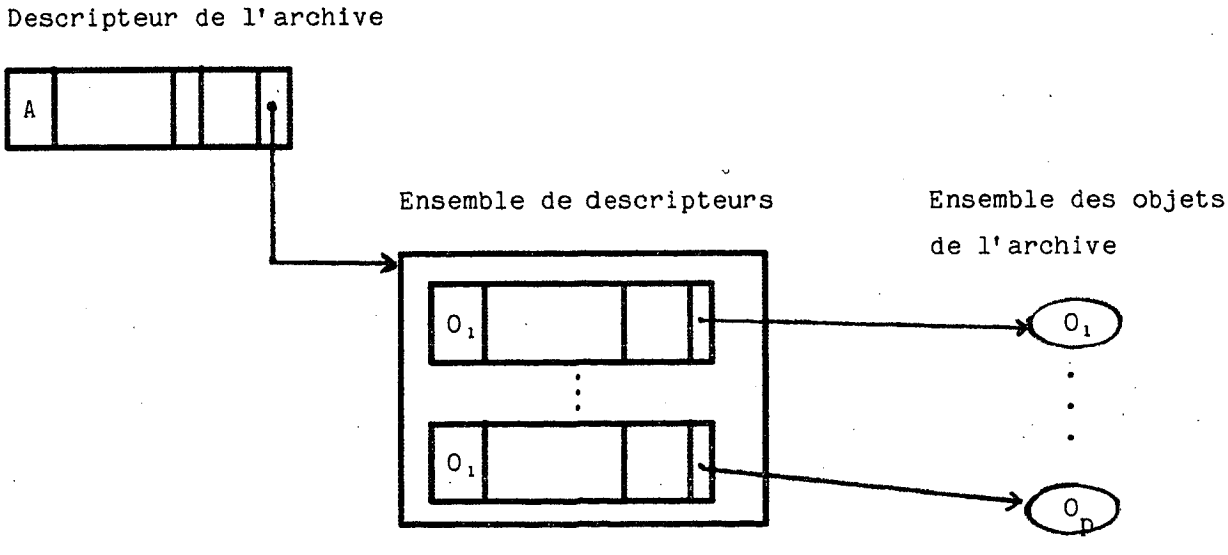


Figure IV.4.2 Schéma logique de l'archive

IV.5 L'implémentation physique

Les différents objets du système seront stockés sur des fichiers binaires organisés en accès direct. Le système comportera donc deux types de fichiers, ceux relatifs à des S.D. et ceux relatifs aux archives contenant des S.R.. On aurait pu imaginer de regrouper les S.D. et, d'une manière générale, les objets du système, dans un fichier unique, stratégie adoptée par des systèmes statistiques comme SAS, par exemple. Cette solution nécessite la ré-écriture des utilitaires (allocation, récupération d'espace disque, gestion de catologue, etc, ...) d'un système de gestion de fichier (SGF) classique en FORTRAN et cette ré-écriture est difficilement portable.

Des problèmes de performance se posent alors rapidement, puisque les S.D. peuvent être de grande taille et l'on est amené à fréquemment créer et supprimer des S.D.. Ce qui n'est pas le cas pour des S.R. qui sont des structures stables, de taille réduite et sur lesquelles, on n'effectue pas d'opérations de gestion.

D'autre part, lorsque tous les objets sont regroupés dans un même fichier, le problème de panne est plus crucial puisque, en cas d'incident, tous les objets du système risquent d'être inaccessibles. En outre, il ne sera pas possible d'exécuter des traitements en parallèle sur des S.D. différents d'un même fichier car la ressource n'est pas partageable.

Ces différentes considérations font qu'à chaque S.D., on associera un fichier grâce à l'allocation dynamique permise par le FORTRAN 77 et l'on utilisera les fonctionnalités du SGF local.

V ARCHITECTURE INTERNE

V.1 Les différentes couches du logiciel

Le système est structuré en trois niveaux (cf. figure V.1.1) qui sont :

Niveau 0 : la gestion des ressources physiques qui correspond à la couche la plus basse et nécessite, en général, une adaptation du système dans le transport sur un site donné.

Niveau 1 : la gestion logique de l'information, c'est l'ensemble des routines qui permettent d'accéder et de manipuler les différents constituants des objets du système.

Ces deux niveaux constituant les fonctions réalisées par le noyau.

Niveau 2 : les programmes d'applications regroupent les modules du niveau supérieur assurant les tâches de gestion et de traitement statistique de données (cf. paragraphe III.1).

La structure modulaire de l'ensemble, la séparation entre les niveaux garantissant une modification et extensibilité aisées du système.

V.2 Le niveau 0 : la gestion des ressources physiques

Les fonctions suivantes sont réalisées à ce niveau :

1) Gestion des fichiers : un ensemble de routines assurent l'interface avec le Système d'Exploitation pour la réservation de fichiers, la gestion d'unités logiques et l'assignation physique de ces fichiers. On distingue les types de fichiers suivants :

- les fichiers binaires en accés direct pour la gestion des objets du système
- les fichiers séquentiels avec format pour les sorties relatives à l'imprimante, les fichiers de sauvegarde, etc, ...
- les fichiers binaires de travail utilisables par les modules.

L'ensemble de ces routines gèrent les demandes d'allocation et de désallocation de ces ressources grâce à une table (cf. figure V.2.1) consignant la disponibilité des ressources.

Les routines du niveau 0 sont bien localisées car elle demandent en général une adaptation à la politique du site, en matière de noms de fichiers et numéros d'unités logiques. Aucune standardisation n'existe en effet sur ces points dans les systèmes d'exploitation.

2) La gestion virtuelle de l'espace disque :

Les supports relatifs aux objets du système seront gérés (cf. [Cro 75]) par un mécanisme de pagination. On considèrera l'espace disque comme une mémoire virtuelle découpée en zones de taille fixe appelées pages auxquelles corespondent n enregistrements physiques du support.

Ainsi, un mot à l'adresse virtuelle a du support s sera interprété comme un couple (p, d) où p est le numéro de la page virtuelle et d est le déplacement du mot recherché dans la page p (cf. figure V.2.2).

Cette adresse virtuelle sera traduite par une fonction topographique simple en une adresse réelle (r,d) où r est le numéro du premier enregistrement physique relatif à la page p.

Le schéma du calcul d'adresse se résume comme suit :

$$(s,a) \xrightarrow[\text{virtuelle}]{\text{adresse}} (s,p,d) \xrightarrow[\text{réelle}]{\text{adresse}} (s,r,d)$$

La mise en oeuvre de la gestion virtuelle des supports se fera par l'intermédiaire :

- d'une table contenant diverses informations (numéro d'unité logique, taille de la page, ...) nécessaires à la gestion virtuelle des espaces disques.

- d'une table contenant les pages actives c'est-à-dire actuellement en mémoire centrale.

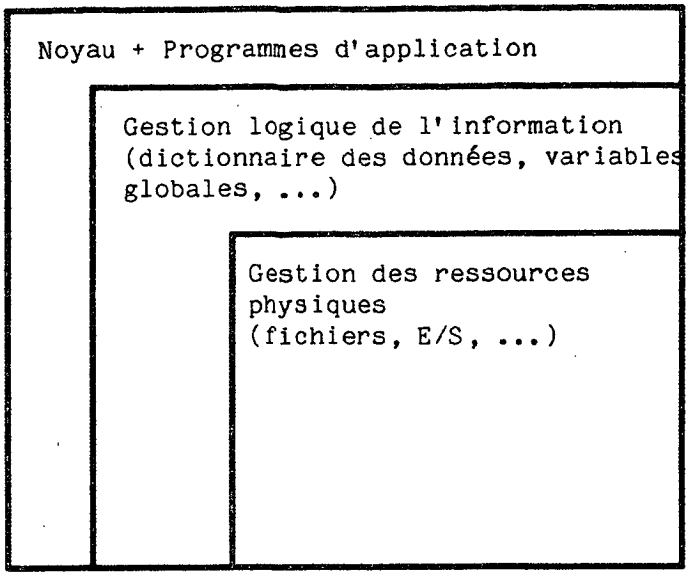


Figure V.1.1 Les différentes couches de SICLA

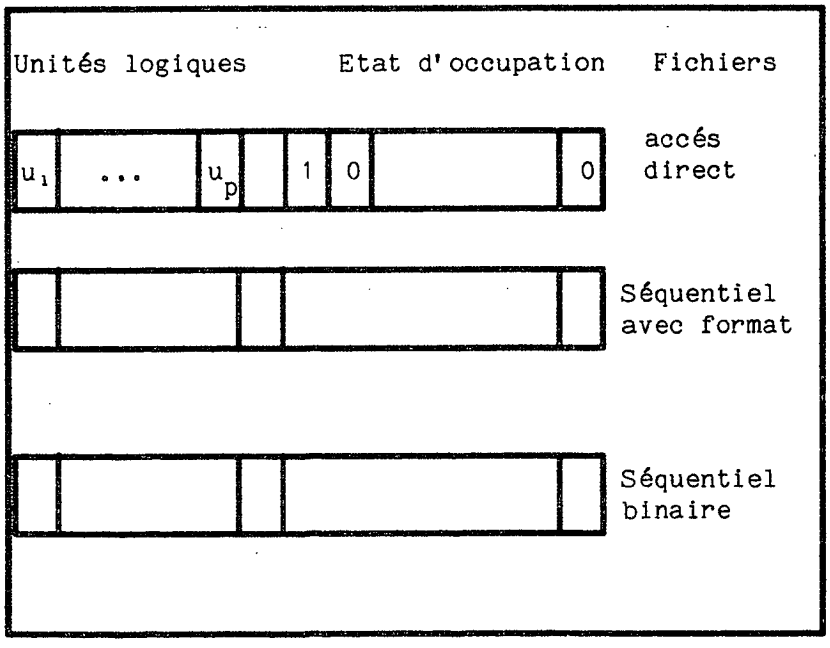
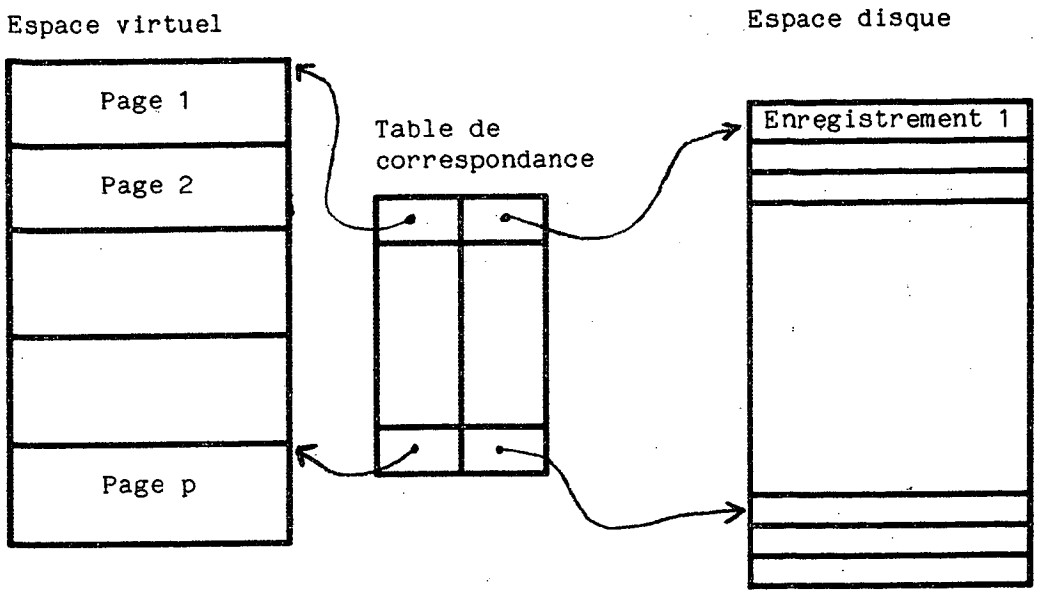


Figure V.2.1 Table de gestion des unités logiques



Correspondance : Espace virtuel et Espace disque

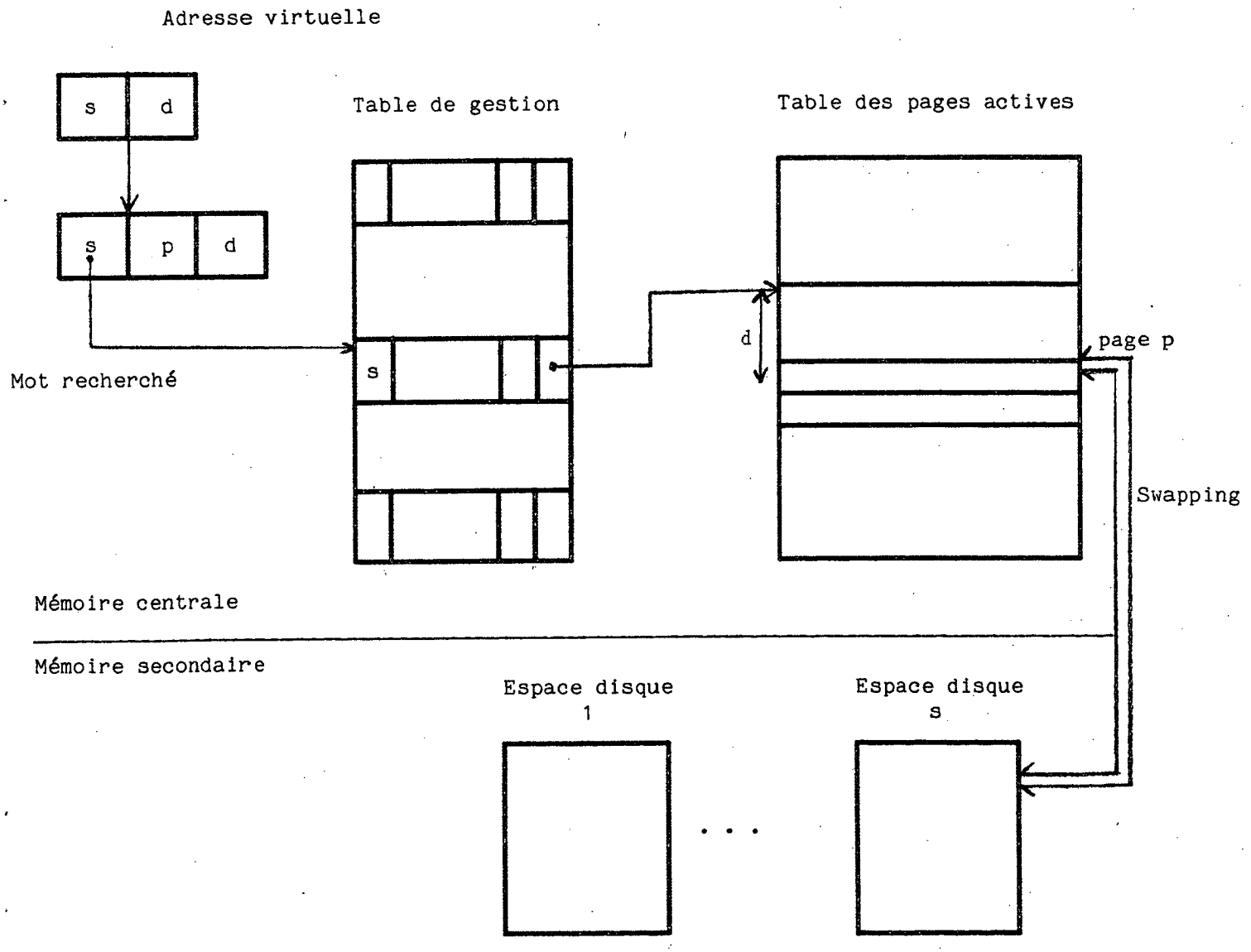


Figure V.2.2 Gestion virtuelle de l'espace disque

L'accés au mot d'adresse a du support s se fera alors de la manière suivante (cf. figure V.2.2) :

1) calcul de l'adresse virtuelle (s, p, d)
2) vérifier si la page demandée (s, p) est active si oui aller en 4, sinon
3) lire les enregistrements (s, r) relatifs à la page (s, p), après avoir éventuellement sauvegardé la page active si elle a été modifiée
4) accéder au mot recherché d'adresse d dans la page

3) Gestion de la mémoire centrale :

Un programme a besoin de zones pour stocker ses variables. Malheureusement, le FORTRAN ne permet pas d'allouer dynamiquement des tableaux, afin d'éviter des redimensionnements successifs et des compilations multiples, il est classique dans les programmes d'A.D. de réserver des super-tableaux par type et de définir les tableaux nécessaires à l'intérieur de ces super-tableaux.

Il nous semblait utile de fournir un certain nombre de modules pour assurer la gestion de ces super-tableaux. Les fonctions réalisées étant :

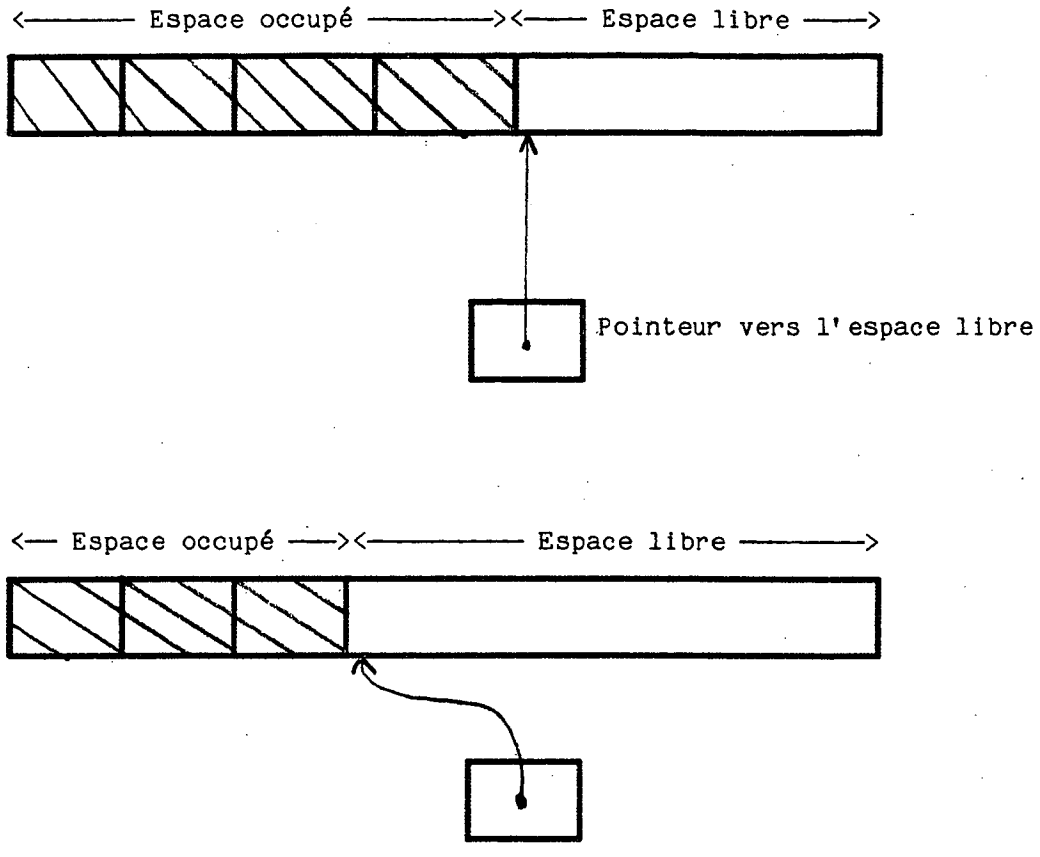
- réservation de zones
- libération de zones
- information sur l'état du super-tableau (place occupée, place disponible)

De telles facilités présentent les avantages suivants par rapport à la gestion manuelle classique :

- diminution des risques d'erreurs puisque c'est le système qui gère les adresses dans le super-tableau
- possibilité de réservation de zones à tout instant dans un sous-programme, ce qui n'est pas possible par une gestion manuelle.

Diverses techniques existent pour la gestion de mémoire par zones de taille variable [Pag 79]. La solution adoptée est une version simplifiée de ces techniques, en effet, la stratégie choisie est :

- choix de la première zone libre
- libération de la place à partir d'une adresse spécifiée (cf. figure V.2.3)



Etat de la mémoire après la libération

à partir de l'adresse a

Figure V.2.3 Gestion de la mémoire centrale

De tels choix suffisent pour les programmes d'A.D.. Rappelons, en effet, que les différentes stratégies de gestion de mémoire ont été étudiées pour la gestion d'une ressource commune mémoire centrale ou secondaire partagée par un ensemble de processus. Les caractéristiques de telles applications sont le caractère aléatoire et dynamique d'allocation et de libération des zones. Les algorithmes de choix de zones libres, de récupération d'espaces libérés sont donc très importantes pour éviter des blocages [Pau 77].

La réservation de place pour un programme d'A.D. est une situation complètement différente puisque le programmeur sait à l'avance les zones qu'il veut réserver. Il est même conseillé que le programme fasse une estimation le plutôt possible de l'espace nécessaire pour son exécution et qu'il compare à celui disponible pour éviter un blocage tardif et désagréable pour l'utilisateur. Les outils de gestion fournis nous semblent alors acceptables pour le domaine de l'A.D., la mise en oeuvre de techniques coûteuses de compactage pour récupérer la place d'une mémoire fragmentée ne nous apparaît pas comme indispensable.

4) Utilitaires de gestion :
Un ensemble de routines sont fournies assurant la gestion du dialogue à la console, le traitement des caractères et d'utilité diverse.

V.3 Le niveau 1 : la gestion logique de l'information

Ce sont l'ensemble des fonctions et sous-programmes permettant :

- d'accéder aux différentes informations relatives aux S.D. et S.R. : lecture du dictionnaire des données, du descripteur d'un constituant de l'archive, etc, ... tel que nous l'avons décrite au paragraphe IV relatif à la structure d'information de SICLA

- d'accéder aux variables globales du système : certaines informations communes à l'ensemble des modules sont accessibles par des fonctions. Ce sont, par exemple , le plus petit et le plus grand entier du système, le nombre maximum d'individus ou variables que SICLA peut traiter, etc, ...

L'ensemble de ces fonctions sont décrites dans le manuel du programmeur.

V.4 Les programmes d'applications

Nous avons indiqué dans l'introduction qu'un des objectifs de SICLA est de fournir un environnement de programmation pour la réalisation de programmes d'A.D.. Cet environnement étant constitué d'un ensemble d'utilitaires de gestion de ressources communes, précédemment décrits, des règles de bonne programmation concernant la lisibilité de programmes et le respect des normes du FORTRAN 77 et d'une méthodologie pour l'écriture d'un module d'A.D.. Elle est fondée sur l'analyse des différentes fonctions réalisées par un tel module.

1) Structure générale d'un programme d'A.D.

On distingue les phases suivantes (cf. figure V.4.1)

a) Initialisation, étape préparatoire où sont faites :

- la réservation de la place des super-tableaux
- le choix des données à analyser : assignation physique et contrôle des fichiers relatifs à la S.D. et à l'archive
- mise en forme de la S.D. : on effectue le contrôle d'adéquation S.D. module et pour une S.D. Individus * Variables, les sélections de groupes variables et individus éventuels. La compatibilité entre les types statistiques des variables et la méthode est vérifiée. A l'issue de cela, une S.D. intermédiaire est créée ne comportant que le sous-ensemble à traiter, ordonné suivant les groupes de variables et individus choisis.
- lecture des paramètres relatifs à la méthode. Cela peut inclure l'accés et le contrôle à des objets S.R. de l'archive pour initialiser l'algorithme par exemple.

b) algorithme :

c'est la phase de traitement proprement dit des données.

c) Terminaison :

à l'issue de l'algorithme, des résultats sont édités et des structures conservées dans l'archive.

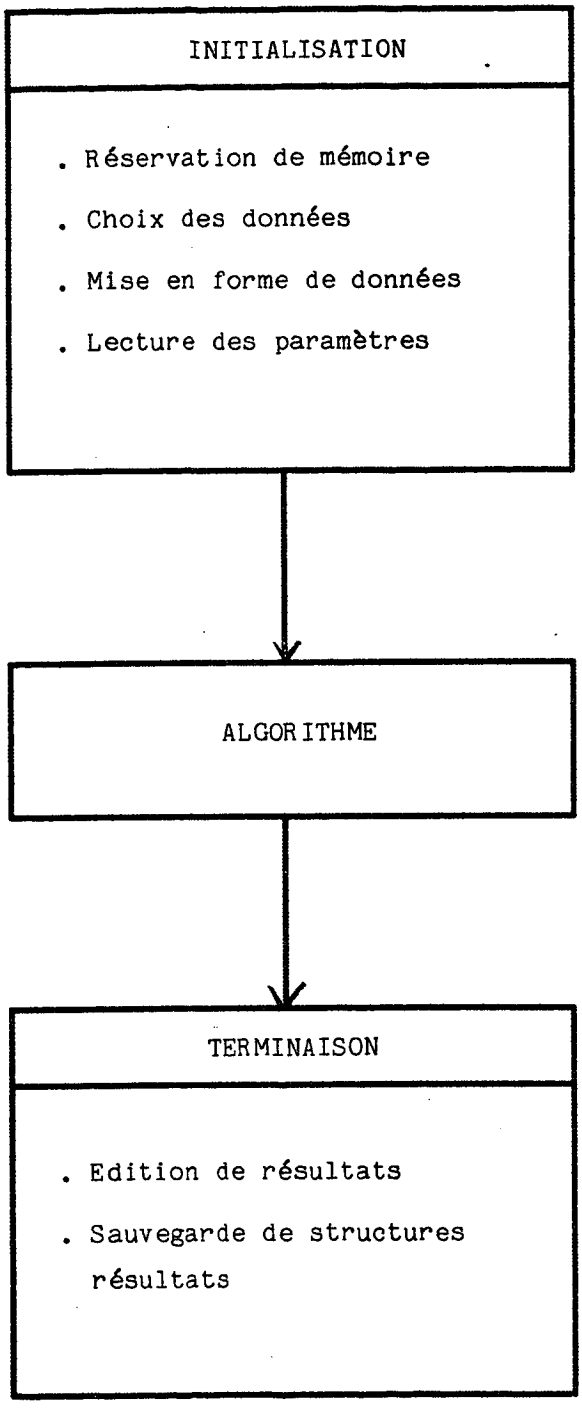


FIGURE V.4.1 Schéma d'un programme dans SICLA

2) Accès lignes ou colonnes

Les tableaux Individus * Variables traités en A.D. peuvent être de taille volumineuse et ne tiennent généralement pas en mémoire centrale.

Les modules opèrent donc séquentiellement, malheureusement, certains modules sont basés sur un accès ligne à ligne tandis que d'autres, colonne par colonne. Les deux possibilités doivent donc être offertes par le système. Le problème qui se pose est donc le choix du mode de stockage de la S.D. transposée ou non. Examinons les stratégies existantes :

1) Choisir un mode de stockage, individus en ligne par exemple et laisser aux méthodes, le soin de transposer les données si elles ont besoin d'un accès variable par variables.

2) Un mode de stockage est choisi pour la S.D. mais, selon les besoins des modules, une S.D. transposée est générée de manière temporaire, pendant la phase de mise en forme des données, par exemple.

3) Admettre dans le système, l'existence de deux S.D. stockées sous les deux formes : transposée (S.D. type Variables * Individus) et non transposée (S.D. type Individus * Variables)

4) Dans la S.D., le tableau de données est stocké sous la forme transposée et non transposée.

- Avant d'évaluer ces différentes possibilités, rappelons que l'espace disque est géré de manière virtuelle par un mécanisme de pagination (cf. paragraphe V.2). Si la taille d'une page est importante et selon la dimension des données, celles-ci peuvent résider entièrement en mémoire centrale. Dans ce cas, les modes d'accès ligne à ligne ou colonne par colonne sont équivalents et le mode de stockage de la S.D. importe peu. Toutefois, il faut envisager à priori que le tableau de données ne réside pas en mémoire centrale.

- La solution traditionnelle 1) est à rejeter car elle reporte sur les modules méthodes, une opération de gestion coûteuse qui est la transposition d'un grand tableau de données. La solution 2) évite cet inconvénient en chargeant le système d'effectuer cette opération. Toutefois, si l'on exécute plusieurs fois la méthode, la transposition des données est répétée à chaque fois, ce qui pénalise les modules dont

le mode d'accès aux données diffère de celui choisi par le système. La solution 3) remédie à cet inconvénient car la S.D. est stockée de manière permanente sous la forme voulue par la méthode mais pose des problèmes sur lesquels nous reviendrons. La solution 4) est la plus séduisante puisque les deux formes de stockage existent dans la S.D., mais elle pose des problèmes de performance, en effet le volume de la S.D. est doublé par rapport aux autres solutions et toutes les opérations de gestion (sélection de variables, génération de nouvelles variables, ...) et de manipulation sont faites en "double" c'est-à-dire sur le tableau transposé et non transposé, ce qui n'est pas souhaitable étant donné la fréquence de ces opérations. La solution adoptée est donc la 3) mais elle peut conduire à des incohérences dues à l'existence sous deux formes physiques distinctes d'un même tableau de données.

Les risques sont, toutefois, minimes car comme nous l'avons déjà fait remarqué, on n'effectue pas de mises à jour en A.D. et nous interdirons toute opération de gestion sur une S.D. Variables * Individus.

VI CONCLUSIONS

Au terme de cet article nous avons donc présenter une méthodologie pour la réalisation d'un système d'analyse des données et justifier les différents choix effectués. La démarche proposée semble être fructueuse d'après les quelques essais réussis de transportabilité du logiciel sur les systèmes VAX, IBM et UNIVAC. Il est à noter toutefois que la programmation de SICLA a été grandement facilitée par le bon environnement du système Multics (édition des biens dynamiques, gestion de bibliothèques, ...). Le travail qui reste à effectuer est d'intégrer de nouveaux programmes d'applications et de nouvelles structures statistiques.

BIBLIOGRAPHIE

(AdD 82) ADIBA A., DELOBEL C.
Base de données et systèmes relationnels.
Editeur Dunod 1982.

(Ben 73) BENZECRI J.P. et Collaborateurs
Analyse des données.
Editeur Dunod 1973.

(Cro 75) CROCUS
Système d'exploitation des ordinateurs.
Editeur Dunod 1975.

(DQR 85) DEMONCHAUX E., QUINQUETON J., RALAMBONDRAINY H.
CLAVECIN : un système expert en Analyse des Données.
Rapport de recherche n° 431, INRIA.

(Did 79) DIDAY E. et Collaborateurs
Optimisation en classification automatique.
Editeur INRIA 1979.

(Fer 75) FERRIE J.
Contrôle de l'accès aux objets dans les systèmes informatiques.
Thèse d'Etat Université Paris VI 1975.

(FrL 82) FRANCIS I., LAURO N.
An Analysis of Developers' and Users' Ratings of Statistical software Using Multiple Correspondence Analysis
5th Symposium Toulouse 1982.

(Gar 83) GARDARIN G.
Base de données, leurs systèmes et leurs langages.
Edition Eyrolles 1983.

(JkR 85) JOMIER G., KEZOUIT O., RALAMBONDRAINY H.
Contribution aux bases de données statistiques. Le système PEPIN-SICLA.
4ème journées Internationales d'Analyse de données, Versailles Octobre 1985.

(PaG 79) PAIR C., GAUDEL M.C.
Les structures de données et leurs représentations en mémoire.
Editeur INRIA, 1979.

(Pau 77 PAUNESCU F.
Analyse de la mémoire virtuelle et simulation du placement des segments d'une architecture à domaine.
Thèse de 3ème cycle, Université Paris VI, 1977.

(Rak 84) RAKOTOMALALA Y.
Graphique en Analyse des Données.
Mémoire d'Ingénieur IIE, 1984.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique