MASTER EN SCIENCES INFORMATIQUES

Contribution à l'évaluation des performances d'algorithmes d'accès à des bases de données

Kerstenne, Anne

Award date:
1983

Awarding institution:
Universite de Namur

FACULTES UNIVERSITAIRES
NOTRE-DAME DE LA PAIX — NAMUR
INSTITUT D'INFORMATIQUE

Contribution à l'évaluation des performances d'algorithmes d'accès à des bases de données

Mémoire présenté par

Anne KERSTENNE

en vue de l'obtention du diplôme de licenciée et maître en informatique

Année académique 1981-1982

C O N T R I B U T I O N A L ' E V A L U A T I O N D E S P E R F O R M A N C E S D ' A L G O R I T H M E S D ' A C C E S A U N E B A S E D E D O N N E E S.

Promoteur:

J.L. HAINAUT

Nous remercions vivement Monsieur HAINAUT pour l'aide précieuse qu'il nous a apportée au cours de ce travail.

TABLE DES MATIERES

Table des matières (suite)

1 - I N T R O D U C T I O N

1.1 - Objet.

Ce travail s'insère dans le cadre de l'élaboration d'un outil d'aide à la conception de systèmes d'information et plus particulièrement du projet I S D O S. Dans cette perspective il est apparu indispensable de chercher à évaluer les performances des différents algorithmes d'accès possibles à la base de données en vue de les optimaliser à un stade ultérieur.

Ce projet I S D O S s'appuie sur la réprésentation en trois niveaux de conception du cycle de vie d'un projet informatique:

- le niveau conceptuel qui fournit les spécifications des données et des traitements indépendamment des moyens de réalisation,
- le niveau logique qui prend en compte les caractéristiques logiques de ces moyens,

- le niveau physique qui tient compte des caractéristiques physiques de ces moyens.

Le présent travail s'inscrit par conséquent dans le niveau logique.

Son but est d'évaluer les performances des algorithmes d'accès: ces algorithmes peuvent être abstraits, en rapport avec le schéma des accès possibles, ou concrets, relatifs au schéma des accès logiques, ou , éventuellement , réels, se référant au schéma des accès logiques du S.G.B.D.

Ce mémoire se réalisera au départ de:

a - la description des algorithmes d'accès,

b - la description statistique de la base de données.

- Description des algorithmes d'accès

Il sera fait usage d'un langage particulier de description: A.D.L. (access algorithms description language) qui peut être décrit comme un langage de programmation au moyen duquel des collections d'occurences de types d'objets sont décrites et les opérations ordonnées.

- Description statistique de la base de données

La structure d'accès est considérée comme une structure binaire. Sa modélisation se base sur les notions d'objets et de relations.

Un objet représente un type d'unités d'information enregistrées: les objets élémentaires correspondent à des informations réprésentées par une valeur (nom, âge, ...), les objets complexes correspondent à des unités d'informations plus complexes décrites par leurs associations avec d'autres objets.

Une relation est associée au mécanisme permettant d'accéder, à partir d'une réalisation d'un objet, à des réalisations d'un objet identique ou différent.

A un objet complexe A , est associé le nombre N_A de ses réalisations.

A une relation R (A , B) sont associés quatre nombres entiers I_{RA}, J_{RA}, I_{RB}, et J_{RB}, tels que, à tout instant, le nombre des réalisations de B accessibles à partir d'une réalisation de A soit compris entre I_{RA} et J_{RA} :
pour tout $a \in A$, $I_{RA} \leq | R(a) | \leq J_{RA}$

et le nombre de réalisations de A accessibles à partir d'une réalisation de B soit compris entre I_{RB} et J_{RB} :
pour tout $b \in B$, $I_{RB} \leq | R(b) | \leq J_{RB}$

D'autre part, une réalisation est décrite par deux fonctions $F_{RA}(k)$ et $F_{RB}(k)$ telles que
pour tout k : $I_{RA} \leq k \leq J_{RA}$

F_{RA} (k) est égal à la proportion des réalisations de A participant à exactement k occurences de R,

et symétriquement pour tout k : $I_{RB} \leqslant k \leqslant J_{RB}$

F_{RB} (k) est égal à la proportion des réalisations de B participant à exactement k occurences de R.

Un objet élémentaire A est décrit par la relation R à laquelle il participe: à A est associé N_{AR} , le nombre de réalisations de A , et une fonction P_{AR} (a) telle que

pour tout $a \in A$, P_{AR} (a) est égal à la proportion des occurences de R dans lesquelles a apparait.

Cette description statistique ne dépend que des propriétés des données et peut, par conséquent, être évaluée avant toute réalisation.

1.2 - Limites.

La présente recherche constitue le prolongement du travail important qui élabore des formules de calcul et a été réalisé par M. J.L. Hainaut.

Il s'agit donc d'un package destiné à implémenter ces formules. Celà implique la création d'une base de données contenant les propriétés des objets et des relations utilisés

par les algorithmes à évaluer, ainsi que leur description statistique. Il existe, pour le projet ISDOS, une base de données décrivant ces objets et ces relations. A celà il faut ajouter une structure de données contenant les informations statistiques. (voir chapitre 5.)
Aucun module d'accès à cette base de données ne sera développé ici.
Les formules implémentées sont reprises à l'annexe 1 , elles se rapportent aux points suivants:

1.2.1 - Probabilité d'un critère de relation

L'expression d'un critère de relation à cardinal est donnée par

$$(r : < K > B(C_B))$$

où B est un objet,

r est le nom d'une relation d'un objet A vers un objet B , caractérisée par I_r , J_r

$< K >$ est un "cardinal" i-j où i et j sont deux entiers tels que $0 \leqslant i \leqslant j \leqslant J_r$

C_B est une condition portant sur les occurences de B et ayant une probabilité p .

La probabilité de cette expression est la probabilité que, parmi les occurences de B accédées par r à partir de A , entre i et j occurences vérifient la condition C_B.

< K > peut prendre des valeurs particulières:

i - : au moins i

j + : au plus j

TS : toutes les occurences.

L'expression d'un critère de relation à ordinal est donnée par

(r : < O > B (C_B))

où B , r , et C_B ont la même signification que ci-dessus,

< O > est un"ordinal" iE - jE où i et j sont deux entiers tels que $0 \leqslant i \leqslant j \leqslant J_r$.

La probabilité de cette expression est la probabilité que, parmi les occurences de B accédées par r à partir de A , celles de reng compris entre i et j vérifient toutes la condition C_B .

< O> peut prendre les valeurs particulières:

iE - DE : du $i^{ème}$ au dernier,

DE : le dernier.

Les probabilités de ces critères seront calculées pour tous les cas. Pour celà la probabilité p de la condition doit être connue.

1.2.2 - Probabilité d'un critère d'appartenance.

La forme générale d'un critère d'appartenance est op E où op est un opérateur égal à = (appartient) ou ≠ (n'appartient pas) et E une collection de réalisations d'objet.

Nous ne considérerons ici que le cas où la collection E est constituée d'une seule valeur.

Pour un objet élémentaire nous calculerons, en plus, la probabilité qu'une de ses occurences soit

$$\leqslant j, < j, > i, \geqslant i, \in [i , j] ,$$

i et j étant des valeurs du même type que l'objet.

1.2.3 - Calcul du coût d'une boucle d'accès.

Une boucle d'accès s'exprime par:

$$[r : <0 / TS> B (C_B) S]$$

où r est le nom de la relation de l'objet A vers l'objet B ,

B, l'objet cible ,

C_B, une condition portant sur B ,

S, une séquence d'action,

<0>, un ordinal : iE - jE ,

et <TS> , signifie "tous".

ce qui correspond à exécuter S , soit pour toutes les réalisations de B, soit pour les occurences de B de rang compris entre i et j accédées par r à partir de A et vérifiant la condition C_B .

Les formules ne concernent que le cas où l'objet A , origine de la relation, est un objet complexe.

Nous calculerons deux valeurs :

m'_r , nombre moyen de réalisations de B sélectionnées par la boucle et soumises à la séquence S ,

et

m''_r , nombre moyen d'accès à réaliser pour une origine de r.

Ces calculs seront faits dans deux cas:

- le cas d'une boucle ne comprenant pas d'ordre "next" et "exit" ,
- le cas d'une boucle contenant un ordre "next" et/ou "exit" relatif à cette boucle.

Nous considérons connues la probabilité de la condition et éventuellement la probabilité d'exécution d'un ordre "next". Nous devons envisager également l'existence, ou non, d'un compteur de cibles.

1.2.4 Nombre d'éléments distincts dans une collection.

Considérons une relation r(A,B) . Le problème est le suivant: étant donné un ensemble A* de réalisations de A , combien d'éléments distincts contiendra la collection de réalisations de B constituée par l'exécution d'une boucle d'accès opérant à partir de A* .

Nous calculerons la proportion des réalisations distinctes de B appartenant à cette collection.

Nous devons connaitre, en plus des valeurs relatives aux calculs associés à une boucle d'accès, la probabilité qu'une occurance de A appartienne à l'ensemble A* .

1.2.5 - Nombre de références distinctes dans une variable de travail.

Nous envisagerons le cas où, dans une séquence, se trouve un ordre "clear" suivi de une ou plusieurs affectations et/ou une ou plusieurs boucles contenant des ordres d'affectations mais pas d'ordre "clear" et éventuellement des ordres "clear" non soumis à un "if".

Dans ce cas nous calculerons la probabilité qu'une occurence de B soit contenue dans la variable, en supposant connue la probabilité d'appartenance à la variable après l'affectation précédente (cette probabilité est nulle avant la première affectation).

1.2.6 - Coût d'une condition.

Nous admettons l'hypothèse que les critères constitutifs de la condition sont indépendants l'un de l'autre et indépendants des accès.

- Critère d'appartenance:

Rappelons que nous ne considérons que l'appartenance à une collection désignée explicitement par une valeur. Dans ce cas, l'évaluation se fait par des comparaisons en mémoire centrale. Le coût peut par conséquent être considéré comme nul.

- Critère de relation:

Nous calculerons deux valeurs:

m_1 : le nombre moyen d'accès à effectuer pour vérifier le critère,

m_2 : le nombre moyen de cas où la condition sera évaluée.

Nous n'envisagerons que le cas où l'évaluation de la condition C_B est effectuée après l'accès à une réalisation et indépendamment de lui.

Il est nécessaire de prendre en compte l'existence, ou non, d'un compteur de cibles et de connaître la probabilité de la condition C_B .

2 - RECHERCHE DES TYPES DE FONCTIONS STATISTIQUES

Dans ce chapitre nous considérons une relation R (A , B) entre un objet A , objet origine et un objet B , objet cible . Nous avons défini prédemment la fonction F (k) associée à cette relation comme étant la fonction qui, à une valeur de k comprise entre I et J , associe la proportion des objets A reliés par R à k objets B, et qui est nulle par k inférieur à I ou supérieur à J .

Les formules de calculs sont basées sur la connaissance de cette fonction. Nous allons distinguer plusieurs cas.

2.1 - Fonction réelle.

La fonction F (k) est connue par ses valeurs pour tout k compris entre I et J . Si l'intervalle I , J est grand, les valeurs de F (k) peuvent être connues par classes de longueurs fixes ou variables.

Ce cas est le plus favorable et donne des résultats plus précis mais il peut s'avérer difficile de déterminer

avec exactitude les valeurs de la fonction.

2.2 - Fonctions classiques de probabilités.

Si la fonction F (k) ne peut être donnée avec précision la première possibilité est de se servir des fonctions élémentaires de probabilité . Celà suppose que l'utilisateur soit capable de distinguer l'allure générale de la fonction.

2.2.1 - Fonction uniforme. discrète

La fonction F (k) est donnée par

pour tout $k \in [I , J]$, $F(k) = p$.

où p est le paramètre de la fonction.

La formule

$$\sum_{k=I}^{J} F(k) = 1 .$$

peut permettre de déterminer p en fonction de I et de J

$$p = \frac{1}{J - I + 1}$$

Mais on peut déterminer à partir d'autres données:

J et F (0) si I = 0

ou la moyenne m de F si I = 1 ou I = 0 ?

développé

toujours en restriction

2.2.2 - Fonction de Poisson.

Nous avons alors :

$$\text{pour tout } k \geqslant 0 \quad , \quad F(k) = e^{-m} \frac{m^k}{k!}$$

où m est la moyenne de la fonction F .
Celà implique $I = 0$ et $J = \infty$. Seule la moyenne m doit être connue.

Nous obtenons ainsi une fonction définie dans $[0 , +\infty[$, et ayant un maximum pour k égal à la partie entière de m . ou $(m+1)$

2.2.3 - Fonction binomiale.

$I = 0$

La fonction est donnée par:

$$\text{pour tout } k \in [I , J] \quad , \quad F(k) = C_J^k \; p^k \; (1 - p)^{J-k}$$

où p est un paramètre tel que la moyenne $m = p\,J$

La moyenne étant en général une donnée plus facile à évaluer, nous la supposerons connue et nous en déduirons p par

$$p = \frac{m}{J}$$

Nous obtenons dans ce cas une fonction définie dans $[0 , J]$ et ayant une valeur maximale pour k égal à

partie entière de (m . (J + 1) / J) .

2.2.4 - Fonction géométrique.

Cette fonction peut être définie de deux façons différentes selon que I = 0 ou 1 .

- si I = 0

pour tout $k \geqslant 0$ $\quad F(k) = p\,(1-p)^k$

où le paramètre p est donné par la relation

$$m = \frac{1-p}{p}$$

c'est à dire

$$p = 1 / (1+m) \;;$$

- si I = 1

pour tout $k > 0$ $\quad F(k) = p\,(1-p)^{k-1}$

où le paramètre p vérifie

$$m = \frac{1}{p} \quad \text{d'où} \quad p = \frac{1}{m}$$

Nous obtenons ainsi une fonction définie dans

$[0\,,+\infty[$ ou $[1\,,+\infty[$

et uniformément décroissante.

2.3 - Insuffisance de ces fonctions.

Il est apparu que les fonctions précédentes (2.2) ne suffisent pas à approcher toutes les fonctions F .

En effet, considérons une base de données particulière dont la description statistique est connue: celle des FNDP .

Les résultats des calculs dont il est question ci-après figurent à l'annexe I .

Pour une relation k , la fonction F réelle étant connue, il est possible de calculer la valeur exacte de

- la probabilité p1 de (r : TS B (C_B)) ,
- la probabilité p2 de (r : 1- B (C_B)) ,
- la probabilité p3 de (r : iE B (C_B)) .

à partir des formules reprises en annexe II .

Si on essaye d'approcher cette fonction F par les fonctions de Poisson, géométrique, binomiale ou uniforme, il apparait que l'erreur relative due à l'approximation de p1 , p2 , p3 , par p'1 , p'2 , p'3 , peut être très importante.

D'autre part, on peut remarquer que la meilleure de ces approximations est obtenue par la fonction définie dans $[0 , J]$, uniforme dans $[1 , J]$, F (0) ayant sa valeur réelle. Or , pour $k > 0$, cette fonction a des valeurs très différentes de celles de la fonction réelle. Le simple fait de tenir compte de F (0) nous fournit une meilleure approximation. Il semble donc raisonnable d'essayer de trouver une fonction pour laquelle F (0) et la moyenne m de F sont connues et qui se rapproche de la fonction réelle. Ces deux valeurs sont en effet plus faciles à déterminer que la fonction réelle et peuvent être demandées à l'utilisateur .

Dans les exemples considérés la fonction réelle est décroissante ce qui amène à choisir une fonction approchée de type exponentiel.

2.4 - Fonctions de type exponentiel.

2.4.1 - Fonction décroissante.

La fonction qui approche le mieux les fonctions réelles choisies est du type :

pour tout $k \in [1 , J]$ $\quad F(k) = a\ e^{-bk}$

où a et b sont deux paramètres qui peuvent être déterminés par les égalités :

$$\sum_{k=1}^{J} F(k) = 1 - F(0) \qquad (1)$$

$$\sum_{k=1}^{J} k\ F(k) = m \qquad (2)$$

F (0) et m étant connus .

Les valeurs de a et b sont données par les équations

$$(m - f) (e^{b})^{J+1} - m (e^{b})^{J} - (m - (J + 1) f) e^{b}$$

$$+ m - J f = 0$$

et

$$a = \frac{f\ (\ e^{b} - 1\ )\ e^{bJ}}{e^{bJ} - 1}$$

où $f = 1 - F(0)$.

Cette fonction fournit la meilleure approximation p'1 , p'2 , p'3 de p1 , p2 , p3 . (voir annexe I). Elle fournit également une qpproximation satisfaisante pour les calculs de

- la probabilité de (r : iE - jE B (C_B)),
- m' et m'' pour la boucle [r : 1E - jE B S] (voir annexe I) .

2.4.2 - Fonctions ayant un maximum.

Les relations choisies nous ont fourni des fonctions F (k) décroissantes mais il peut exister des fonctions ayant un maximum pout une valeur connue de k : k_0 .
Nous distinguerons deux cas:

1) la valeur de F en k_0 est nettement supérieure aux valeurs de F en $k \neq k_0$. Dans ce cas nous choisirons la fonction F telle que

   pour tout $k \in [1 , J]$ $F(k) = a\ e^{-b\ |k - k_0|}$

   où a et b sont deux paramètres déterminés par les égalités (1) et (2) figurant en 2.4.1. Si on pose $f = 1 - F(0)$, les équations deviennent :

$$a = \frac{f\ e^{bJ}\ (\ e^{b} - 1\ )}{e^{bJ} - 1}$$

et

$$(m - k_0 f) (e^b)^{J+1} - (m - k_0 f) (e^b)^{J-1}$$
$$- m (e^b)^{J+1-k_0} + (m - f) (e^b)^{J-k_0}$$
$$- (m - (J + 1) f) (e^b)^{k_0} + (m - J f) (e^b)^{k_0-1}$$
$$= 0$$

2) Dans les autres cas on choisira la fonction F telle que

pour tout $k \in [1 , k_0]$ $F (k) = a (1 - e^{-bk})$

et

pour tout $k \in [k_0 , J]$

$$F (k) = a (1 - e^{bk_0}) e^{-b(k - k_0)}$$

où les paramètres a et b déterminés à partir des égalités (1) et (2) sont donnés par :

$$k_0 (2m - (k_0 + 1) f) (e^b)^{J+2}$$
$$- 2 k_0 (2 m - k_0 f) (e^b)^{J+1}$$
$$+ k_0 (2 m - (k_0 - 1) f) (e^b)^{J}$$
$$- 2 (m - (J + 1) f) (e^b)^{k_0+1}$$
$$+ 2 (m - J f) (e^b)^{k_0}$$
$$+ 2 (m - (J + 1) f) e^b$$
$$- 2 (m - J f) = 0$$

et

$$a = \frac{f e^{bJ} (e^b - 1)}{k_0 e^{bJ} (e^b - 1) + 1 - e^{bk_0}}$$

où $f = 1 - F (0)$

Dans le but de ne pas rendre le choix difficile à l'utilisateur nous éviterons de multiplier les formules et nous nous contenterons des fonctions ci - dessus .

2.5 - Cas où J = 1 et J = 2 .

Dans ces deux cas il peut y avoir une simplification.

1) J = 1

Les égalités (1) et (2) deviennent :

$$F(1) = 1 - F(0) \qquad (1')$$

$$F(1) = m \qquad (2')$$

ce qui nous permet de connaître la fonction F si on connait uniquement m ou F (0)

2) J = 2

Les égalités (1) et (2) donnent :

$$F(1) + F(2) = 1 - F(0)$$

$$F(1) + 2F(2) = m$$

d'où

$$F(1) = 2(1 - F(0)) - m$$

$$F(2) = m - 1 + F(0)$$

Si on connait m et F (0) la fonction est ainsi parfaitement définie.

Les valeurs F(1) et F (2) sont des valeurs de probabilité et doivent par conséquent être comprises entre 0 et 1 ce qui entraine la condition

$$1 - F(0) \leqslant m \leqslant 2(1 - F(0))$$

portant sur m et F (0)

3 - ALGORITHMES DE CALCUL

3.1 - Implémentation des formules comprenant F (k) .

La liste de ces formules est reprise en annexe II .

Il apparait à l'examen que certaines expressions se retrouvent dans des formules différentes, seuls les indices de sommation étant différents. Il est par conséquent utile de paramètrer ces expressions ce qui nous donne les sommes suivantes où nous poserons

$$a = \inf (k , y) \quad \text{et} \quad b = \inf (l , y) ,$$

$$somf1 (x , y) = \sum_{k=x}^{J} F (k) \sum_{l=x}^{a} \Pi (k , l)$$

$$somf2 (x , y) = \sum_{k=x}^{y} F (k)$$

$$somf3 (x , y) = \sum_{k=x}^{J} F (k) \; \Pi \; (a - x + 1 , a - x + 1)$$

$$somf4 (x , y) = \sum_{k=x}^{J} F (k) \sum_{l=x}^{k} (b - x + 1) \; \Pi (k , l)$$

$$somf5\ (x, y) = \sum_{k=x}^{J} F(k)\ (a - x + 1)$$

$$somf6\ (x, y) = \sum_{k=x}^{y} k\ F(k)$$

$$somf7\ (x, y) = \sum_{k=x+1}^{y} F(k)\ \left(k \sum_{l=0}^{x-1} \Pi(k, l) + x(k+1) \sum_{l=x}^{k} \frac{\Pi(k, l)}{l+1} \right)$$

$$somf8 = \sum_{k=2}^{J} F(k) \sum_{l=1}^{k} \Pi(k, l) \left(1 - \frac{l(k+1)}{l+1} \right)$$

$$somf9\ (y) = \frac{1}{(1 - p_N)} \sum_{k=0}^{J} F(k) \sum_{l=0}^{k} \Pi(k, l)\ p_N^b$$

$$somf10\ (x, y) = \frac{1}{(1 - p_N)} \sum_{k=x}^{J} F(k) \sum_{l=x}^{k} \Pi(k, l)\ (1 - p_N^{b-x+1})$$

$$somf11\ (x, y) = \frac{1}{(1 - p_N)} \sum_{k=x}^{J} F(k)\ (1 - p_N^{a-x+1})$$

$$somf12 = \frac{1}{(1 - p_N)} \left(1 - \sum_{k=0}^{J} F(k)\ p_N^k \right)$$

$$somf13\ (x, y) = \sum_{k=x}^{J} F(k)\ \left(k \sum_{l=0}^{x-1} \Pi\ (k, l) \right.$$

$$+ \sum_{l=x}^{a} \frac{\Pi\ (k, l)}{l+1}\ \left((k+1)\left(x - 1 + \frac{1}{1-p_N} \right) \right.$$

$$\left. - \left(1 - \frac{1 + kp_N}{1 - p_N} \right)\ p_N^{l-x+1} \right)$$

$$\left. + (k+1)\left(x - 1 + \frac{1 - p_N^{y-x+1}}{1-p_N} \right) \sum_{l=y+1}^{k} \frac{\Pi\ (k, l)}{l+1} \right)$$

$$somf14\ (y) = y + 1 - \sum_{k=0}^{y} (y + 1 - k)\ F(k)$$

$$somf15\ (x, y, z) = \sum_{k=x}^{J} F(k)\ \left(z \right.$$

$$+ \sum_{l=0}^{a-x} \Pi\ (a - x + 1, l)\ \frac{a - x + 2}{a - x + 2 - l}$$

$$\left. + (a - x + 1)\ \Pi\ (a - x + 1, a - x + 1) \right)$$

$$somf16\ (x, y) = \sum_{k=x}^{J} F(k)\left(x - 1 + \frac{1 - p_N^{a-x+1}}{1-p_N} \right)$$

Ces seize sommes font également apparaître l'utilité

de calculer séparément

$$\text{somp1}\ (x, y, k) = \sum_{l=x}^{y} \Pi\ (k, l)$$

$$\text{somp2}\ (x, k) = \sum_{l=x}^{k} \frac{\Pi\ (k, l)}{l + 1}$$

Les calculs des différentes probabilités et moyennes se feront par utilisation de ces expressions avec les valeurs adéquates pour x , y et éventuellement z . Ils seront indépendants du type de fonction choisi pour F qui n'interviendra que dans le calcul de F (k) .

3.2 - Emploi des expression paramètrées.

3.2.1 - Probabilité d'un critère de relation à cardinal .

a) p ≠ 1

i - j	pr = somf1 (i , j)
i -	pr = 1 - somf1 (0 , i - 1)
j +	pr = somf1 (0 , j)
TS	pr = somf3 (1 , J)

b) p = 1

i - j	pr = somf2 (i , j)
i -	pr = 1 - somf2 (0 , i - 1)
j +	pr = somf2 (0 , j)
TS	pr = 1 - F (0)

3.2.2 - Probabilité d'un critère de relation à ordinal.

i E - j E	pr = somf3 (i , j)
i E	pr = p (1 - somf2 (0 , i - 1))
1 E	pr = p (1 - F (0))
D E	pr = p (1 - F (0))
i E - D E	pr = somf3 (i , J)

3.2.3 - Boucle d'accès sans ordre "next" ni "exit".

a) TS et 1 E - D E m' = p m

m'' = m

b) i E - j E , dans ce cas, en l'absence de formule particulière " DE " doit être remplacé par J E .

p ≠ 1

m' = somf4 (i , j)

avec compteur :

m '' = somf6 (i , j) + somf7 (j , J)

sans compteur :

m'' = somf6 (1 , j) + somf7 (j , J)

p = 1

m' = somf5 (i , j)

avec compteur :

m'' = somf6 (i , j) + j somf2 (j + 1 , J)

sans compteur :

m'' = somf6 (1 , j) + j somf2 (j + 1 , J)

c) 1 E $\quad p \neq 1$

$m'' = m + somf8$

d) i E - D E $\quad$ sans compteur

$m'' = m$

3.2.4 - Boucle d'accès avec ordre "next" et "exit".

a) T S et 1 E - D E

$p \neq 1$

$$m' = \frac{1}{1 - p_N} - somf9\ (J)$$

$m'' = somf13\ (1, J)$

$p = 1$

$m' = m'' = somf12$

b) i E - j E $\quad$ (ici aussi DE = JE)

$p \neq 1$

$m' = somf10\ (i, j)$

avec compteur :

$m'' = somf13\ (i, j)$

sans compteur :

$m'' = somf13\ (i, j) + somf6\ (1, i - 1)$

$p = 1$

$m' = somf11\ (i, j)$

sans compteur :

$m'' = somf16\ (i, j) + somf6\ (1, i - 1)$

avec compteur :

$m'' = somf16\ (i, j)$.

3.2.5 - Coût d'évaluation d'un critère de relation à cardinal.

a) i - j

p ≠ 1

avec compteur:

$m_1 = m_2$ = somf7 (i - 1 , j) + somf7 (j , J)

sans compteur :

$m_1 = m_2$ = somf6 (1 , j) + somf7 (j , J)

p = 1

avec compteur :

$m_1 = m_2 = 0$

sans compteur :

$m_1 = m_2$ = somf14 (j)

b) i -

p ≠ 1

avec compteur :

$m_1 = m_2$ = somf7 (i - 1, J)

sans compteur :

$m_1 = m_2$ = somf6 (1 , i - 1) + somf7 (i - 1,J)

p = 1

sans compteur :

$m_1 = m_2$ = somf14 (i - 1)

c) j +

p ≠ 1

avec compteur :

$m_1 = m_2$ = somf7 (j , J)

d) T S

p ≠ 1

$m_1 = m_2 =$ somf15 (1 , J , 0)

p = 1

$m_1 = m_2 =$ somf14 (J)

e) i - i

p ≠ 1

avec compteur :

$m_1 = m_2 =$ i F (i) + somf7 (j , J)

3.2.6 - Coût d'évaluation d'un critère de relation à ordinal.

a) i E = j E (où DE = JE)

$m_2 =$ somf15 (i , j , 0)

avec compteur :

$m_1 =$ somf15 (i , j , i - 1)

sans compteur :

$m_1 =$ somf6 (1 , i - 1) + somf15 (i,j,i-1)

b) D E

$m_1 = m$

$m_2 = 1 - F(0)$

c) i E

$m_2 =$ somf2 (i , J)

avec compteur :

$m_1 = i\ m_2$

sans compteur :

$m_1 = i\ m_2 +$ somf6 (1 , i - 1)

3.2.7 - Nombre d'éléments dans une collection.

$$p\ (= B^{*}) = 1 - somf1\ (0\ ,\ 0)$$

3.3 - Critère d'appartenance.

3.3.1 - Objet élémentaire.

Dans le cas d'un objet élémentaire B dont les occurences sont du type réel, entier ou alphanumérique, nous considérons, pour la collection E , les cas
$= T$, $\neq T$, $< T$, $\leqslant T$, $> T$, $\geqslant T$, $\in [T\ ,\ T']$
où T et T' sont des valeurs de type réel,entier ou alphanumérique. Il faut par conséquent définir, pour une relation entre un objet origine et un objet cible élémentaire, la proportion Pr (b) des occurences de la relation dans lesquelles la valeur b de B apparait.
Nous distinguerons quatre types :

a) Les valeurs sont réelles et comprises entre I et J , nombres réels. Dans ce cas les proportions devront être définies par classes, le nombre de valeurs possibles étant infini.

b) Les valeurs sont entières et comprises entre I et J , entiers. Alors les proportions peuvent être données soit pour chaque valeur comprise entre I et J , soit pour chaque classe de valeurs si $[I\ ,\ J]$ est partagé en classes.

c) Les valeurs sont alphanumériques de longueur n et comprises entre I et J , chaine de caractères de longueur n . Dans ce cas les proportions sont données soit pour chaque valeur comprise entre I et J , soit pour chaque classe si l'ensemble des valeurs est partagé en classes.

d) Les valeurs sont de type alphanumérique ou entier mais ne prennent qu'un nombre limité de valeurs non consécutives. Citons comme exemples: un item sexe, un item couleur, un item pays ... . Pour chacune de ces valeurs on donnera la proportion.

A partir de là nous obtiendrons la probabilité cherchée comme suit:

1) $p (= \mathbb{T}) = Pr (\mathbb{T})$

2) $p (\neq \mathbb{T}) = 1 - Pr (T)$

3) $p (< \mathbb{T}) = \sum_{b \in [I , T[} Pr (b)$

4) $p (\leqslant T) = \sum_{b \in [I , T]} Pr (b)$

5) $p (> \mathbb{T}) = \sum_{b \in]T , J]} Pr (b)$

6) $p (\geqslant \mathbb{T}) = \sum_{b \in [T , J]} Pr (b)$

7) $p (\in [\mathbb{T} , \mathbb{T}']) = \sum_{b \in [T , T']} Pr (b)$

Remarquons que dans le cas d'un objet dont les occurences sont de type réel, la probabilité de l'égalité ne peut être

définie, il faut donc remplacer $p\ (= a)$
par $p\ (\in [a - \varepsilon\ ,\ a + \varepsilon])$

3.3.2 - Objet complexe.

Dans ce cas nous ne pouvons calculer que $p\ (= a)$, $p\ (\neq a)$ où a est une occurence de A, et on a

$$p\ (\neq a) = 1 - p\ (a)$$

Nous ne pouvons, ici, définir de proportion. Nous aurons alors :

$$p\ (a) = \frac{1}{\mathbb{N}_A}$$

4 - CATALOGUE ET MODES D'EMPLOI

4.1 - Catalogue des fonctions F.

4.1.1 - Fonction réelle.

Si on partage l'intervalle $[I , J]$ en n classes on a

pour tout i : $1 \leq i \leq n$

$$F(k_i) = Pr\left[k'_i \leq k \leq k_i \right]$$

pour tout i : $2 \leq i \leq n$

$$k'_i = k_{i-1} + 1$$

et $k'_1 = I$, $k_n = J$.

La variable aléatoire k est égale au nombre d'occurences de B reliées par R à une même occurence de A .

4.1.2 - Fonction binomiale.

On connait les valeurs de m et de J. Si on pose

$$p = m / J$$

alors

$$F(k) = C_j^k \; p^k \; (1 - p)^{J - k}$$

pour tout $k \in [0 , J]$

4.1.3 - Fonction géométrique 0 .

On connait la valeur de m . Si on pose :

$$p = 1 / (m + 1)$$

on a

$$F(k) = p (1 - p)^k$$

pour tout $k \geqslant 0$.

4.1.4 - Fonction géométrique 1 .

On connait la valeur de m qui est supérieure à 1 . Si on pose:

$$p = 1 / m$$

alors

$$F(k) = p (1 - p)^{k-1}$$

pour tout $k \geqslant 1$.

4.1.5 - Fonction de Poisson .

On connait la valeur de m et on a

$$F(k) = e^{-m} \frac{m^k}{k!}$$

pour tout $k \geqslant 0$.

4.1.6 - Fonction uniforme 1 .

On connait la valeur de I et de J . Si on pose

$$p = \frac{1}{J - I + 1}$$

alors

$$F(k) = p$$

pour tout $k \in [I , J]$

4.1.7 Fonction uniforme 2.

On connait les valeurs de J et F (0). Si on pose

$$p = (1 - F(0)) / J$$

alors

$$F(k) = p$$

pour tout $k \in [1 , J]$

4.1.8 - Fonction uniforme 3.

On connait la valeur de m et I = 0 . Si on pose

$$p = 1 / (2m + 1)$$

on a

$$F(k) = p$$

pour tout $k \in [0 , J]$ avec $J = 2m$

4.1.9 - Fonction uniforme 4.

On connait la valeur de m et I = 1 . Si on pose

$$p = 1 / (2m - 1)$$

alors

$$F(k) = p$$

pour tout $k \in [1 , J]$ avec $J = 2m - 1$

4.1.10 - Fonction décroissante.

On connait la valeur de m , J , et F (0). On a

$$F(k) = a\ e^{-bk}$$

pour tout $k \in [1 , J]$

Les paramètres a et b sont donnés par les équations se trouvant page 16 et page 17.

4.1.11 - Fonction à maximum 1.

On connait les valeurs de m , J , F (0) et k_0 valeur de k pour laquelle F est maximum. On a

$$F(k) = a\ e^{-b\,|k - k_0|}$$

pour tout $k \in [1 , J]$

Les paramètres a et b sont donnés par les équations se trouvant page 17 et page 18.

4.1.12 - Fonction à maximum 2.

On connait les valeurs de m , J , F (0) et k_0 .

On a

$$F(k) = a\ (1 - e^{-bk})$$

pour tout $k \in [1 , k_0]$

et

$$F(k) = a\ (1 - e^{-bk_0})\ e^{-b\,(k-k_0)}$$

pour tout $k \in [k_0 , J]$

Les paramètres a et b sont donnés par les équations se trouvant page 18 .

4.1.13 - Fonction pour laquelle J = 1 .

On connait

soit la valeur de m , d'où

$$F(0) = 1 - m$$

$$F(1) = m$$

soit la valeur de F (0) , d'où

$$F(1) = 1 - F(0)$$

4.1.14 - Fonction pour laquelle J = 2 .

On connait la valeur de m et de F (0) et on a

$$F(1) = 2(1 - F(0)) - m$$

$$F(2) = m - (1 - F(0)) .$$

4.2 - Mode d'emploi des fonctions F .

Pour choisir la fonction F , nous examinerons les cas suivants, dans l'ordre, tant que ne sera pas trouvée de fonction satisfaisante :

4.2.1 - 1er cas : J = 1 .

Dans ce cas il faut évidemment choisir la fonction figurant en 4.1.13 et donner soit la valeur de m , soit celle de F (0) . Si on donne les deux valeurs elles doivent vérifier l'équation :

$$F(0) = 1 - m$$

4.2.2 - 2ème cas : F est connue.

Nous choisirons donc la fonction réelle (4.1.1.). Il faut donner les valeurs de I, de J et de m . Pour chaque valeur de F nous donnerons la limite supérieure de la classe k_i .

Dans le cas où la valeur de la fonction est connue pour tout k compris entre I et J , on a $k_i = k$.

La limite supérieure k_n de la dernière classe doit être

égale à J . La fonction F doit évidemment vérifier

$$\sum_{k=I}^{J} F(k) = 1$$

$$\sum_{k=I}^{J} k\ F(k) = m$$

4.2.3 - 3ème cas : J = 2 .

Nous choisirons la fonction figurant en 4.1.14 . Les données seront alors F (0) et m .

Ces deux valeurs doivent vérifier les inégalités :

$$1 - F(0) \leqslant m \leqslant 2\,(1 - F(0)) .$$

4.2.4 - 4ème cas : F est une fonction uniforme.

Nous choisirons la fonction d'après les valeurs qui peuvent être connues: I et J ; F (0) et J ; m si I = 0 ; m si I = 1 .

a) On connait I et J : la fonction F est donnée en 4.1.6

b) Si la fonction est uniforme dans $[1 , J]$ et a une valeur connue F (0) en 0 , différente de sa valeur en $k \neq 0$, nous choisirons la fonction décrite en 4.1.7 . Les données sont par conséquent J et F (0) .

c) Si la moyenne m est connue, si I = 0 , la fonction F est décrite en 4.1.8, par contre, si I = 1 , la

fonction est décrite en 4.1.9 . Dans les deux cas, la seule donnée est la moyenne m .

4.2.5 - 5ème cas : F est une fonction classique.

Dans ce cas l'utilisateur est capable de déterminer que la fonction F est géométrique, binomiale ou de Poisson.

a) Si la fonction est géométrique deux possibilités se présentent :

si I = 0 la fonction est donnée en 4.1.3

si I = 1 elle est décrite en 4.1.4

Dans les deux cas la seule donnée est la moyenne m .

b) Si la fonction est binomiale (voir 4.1.2) les données sont m et J .

c) Si la fonction est de Poisson, la seule valeur à fournir est la valeur de la moyenne. La fonction est décrite en 4.1.5 .

4.2.6 - 6ème cas : F est une fonction décroissante.

Nous choisirons alors la fonction décrite en 4.1.10 . Il faudra fournit les valeurs de m , J et F (0) .

4.2.7 - 7ème cas : F a un maximum.

Les données seront alors m , J , F (0) et k_0 , valeur de k pour laquelle F (k) est maximum.

Si l'utilisateur estime que la valeur de la fonction F en k_0 est nettement supérieure à la valeur de F en k dif-

férent de k_0 il choisira la fonction décrite en 4.1.11 , sinon, celle donnée en 4.1.12 .

4.2.8 - Remarque.

Dans tous les cas l'utilisateur devra donner le type de la fonction choisie et le nombre N_B de cibles différentes existant dans le système.

4.3 - Mode d'emploi pour un objet élémentaire.

Nous avons distingué quatre types de proportions (voir 3.3.1) . Si les valeurs sont de type réel, la fonction de proportion est obligatoirement donnée par classe. Pour les valeurs de types entier et alphanumérique comprises entre deux valeurs, elle peut être donnée par classe ou pour chaque valeur possible. Dans le cas de valeurs alphanumériques ayant un nombre limité de valeurs consécutives, elle est donnée pour chaque valeur.

4.3.1 - Fonction de proportions donnée par classe.

Dans ce cas, pour chaque classe de valeurs, nous donnerons la limite maximale de la classe et la proportion pour cette classe. La première classe donnera la valeur minimale I et la proportion associée sera nulle.

4.3.2 - Fonction de proportion inconnue .

Nous traiterons ce cas de la même manière qu'en 4.3.1 en

considérant l'intervalle I , J formé d'une seule classe.
Nous définirons deux classes :

- l'une donnant I et de proportion nulle ,
- l'autre donnant J et ayant une proportion égale à 1 .

4.3.3 - Fonction de proportion donnée pour chaque valeur.

A chaque valeur a comprise entre I et J sera associée la proportion Pr (a)

5 - BASE DE DONNEES DES STATISTIQUES

5.1 - Introduction

Dans le cadre du projet I S D O S , a été développé un schéma conceptuel des données reprenant, entr'autres, les notion de type d'article, de type de chemin et d'item et servant à décrire la base de données de l'utilisateur.

Ces notions peuvent être mises en correspondance avec celles de relation et d'objet décrites précédemment : un type de chemin correspond à une relation, un item élémentaire à un objet élémentaire, un item décomposable et un type d'article à un objet complexe ; de plus , l'attribution d'un item à un type d'article correspond aussi à une relation.

A ce schéma existant il faut ajouter la description statistique de la base de données de l'utilisateur.

Pour celà nous allons ajouter :

1) trois types d'entité :
   - le type d'article "statistic description" ,
   - le type d'article "segment" ,
   - le type d'article "proportion" .

2) cinq types d'association :
   - le type d'association "item description" entre

"item" et "statistic description" ,

- le type d'association "path type description" entre "path type" et "statistic description" ,
- le type d'association "proportion function" entre "proportion" et "statistic description" ,
- le type d'association "direct function" entre "statistic description" et "segment" ,
- le type d'association "inverse function" entre "statistic description" et "segment" .

La représentation graphique se trouve en annexe III .

5.2 - Définitions.

5.2.1 - "Statistic description" .

Une entité de ce type décrit statistiquement une occurence de "item" , c'est à dire une relation R (A,B) où B est un objet élémentaire, ou une occurence de "path type" , c'est à dire d'une relation R (A,B) où A et B sont des objets complexes.

Elle se compose de trois groupes de propriétés :

1) "description de la fonction directe" décrivant statistiquement la relation R (A,B) :

Ce groupe contient

- type D : donnant le type de la fonction F (k) choisie.
- m D : donnant la moyenne de la fonction F .
- F (0) D : donnant la valeur de la fonction F en 0.
- k_0 D : donnant la valeur de k pour laquelle F (k) est maximum .

- par 1 D : donnant le 1er paramètre de la fonction F . Sa valeur est donnée dans le catalogue (p ou e^b ou F (1) si J = 1 ou 2).
- par 2 D : donnant le 2ème paramètre de la fonction F. Il n'est nécessaire que pour les fonctions de type exponentiel (paramètre a) ou si J = 2 , il est égal à F (2).
- I D : donnant la valeur de I pour la fonction F .
- J D : donnant la valeur de J pour la fonction F .
- NC D : donnant le nombre d'objets B .

2) "description de la fonction inverse" décrivant statistiquement la relation inverse de R (A,,B) : S (B , A). Ce groupe se compose de :
type I , m I , F (0) I, k_0 I , par 1 I , par 2 I , I I , J I , ces propriétés sont définies comme ci-dessus. et de NC I donnant le nombre d'objets A .

3) type P donnant le type de proportion.

Il est à remarquer que certaines propriétés peuvent avoir une valeur inconnue.

5.2.2 - "Segment".

Une entité de ce type décrit une classe pour la fonction F (k) associée à une relation R (A,B) ou à son inverse.
Elle contient trois propriétés:

- type S : détermine si la fonction F concernée se

rapporte à la relation R (A,B) ou à son inverse,
- maxclasse donne la limite supérieure de la classe,
- valeur S donne la proportion des réalisations de A (de B si inverse) participant à n occurences de R , n appartenant à cette classe.

5.2.3 - "Proportion".

Une entité de ce type décrit une classe pour la fonction de proportion Pr (b) associée à une relation R (A,B) où B est un objet élémentaire.

Elle contient deux propriétés;
- T , donnant la limite maximale de la classe,
- valeur P , donnant la proportion des réalisations de A auxquelles est associée une valeur de B appartenant à cette classe.

5.2.4 - "Item description".

Une association de ce type existe quand à une entité item représentant un objet élémentaire B lié à un objet complexe A , est associée une entité "statistic description" décrivant la relation R (A,B) . Sa connectivité est 0 - 1 , 0 - 1 .

5.2.5 - "Path type description".

Une association de ce type existe quand à une entité "path type" représentant une relation R (A,B) où A et

B sont des objets complexes, est associée une entité "statistic description" décrivant la relation. Sa connectivité est 0 - 1 , 0 - 1 .

5.2.6 - "Proportion function".

Une association de ce type existe quand à une entité "statistic description" décrivant statistiquement une relation R (A,B) où B est un objet élémentaire, est associée une entité "proportion" donnant la description d'une classe de la fonction de proportion Pr (b) , b $\in$ B. Sa connectivité est 1 - 1 pour "proportion" et 0 - * pour "statistic description".

5.2.7 - "Direct function".

Une association de ce type existe quand à une entité "statistic description" décrivant statistiquement une relation R (A,B) , est associée une entité "segment" décrivant une classe de la fonction associée à R . Sa connectivité est 0 - 1 pour "segment" et 0 - * pour"statistic description".

5.2.8 - "Inverse function".

Une association de ce type existe quand à une entité "statistic description" décrivant statistiquement une relation R (A,B) , est associée une entité "segment" décrivant une classe de la fonction F associée à l'inverse de R .

Sa connectivité est 0 - 1 pour "segment" et 0 - * pour "statistic description".

5.3 - Contraintes d'intégrité.

5.3.1 - "Statistic description" et ses propriétés.

a) Une entité "statistic description" intervient obligatoirement dans une association " item description " ou (exclusif) dans une association "path type description".

b) type D , NC D et NC I ne peuvent avoir de valeur inconnue.

c) Si ces valeurs sont connues on doit avoir:

$$m\,D \circ NC\,I = m\,I \circ NC\,D \circ$$

$$0 \leqslant I\,D \leqslant k_0 \leqslant J\,D \leqslant NC\,D$$

$$0 \leqslant I\,I \leqslant k_0 \leqslant J\,I \leqslant NC\,I$$

$$0 \leqslant F\,(0)\,D \leqslant 1$$

$$0 \leqslant F\,(0)\,I \leqslant 1$$

d) type D , type I ne peuvent prendre que les quatorze valeurs données dans le catalogue (voir 4.1).

e) type P ne peut prendre que les quatre valeurs données en 3.3.1 : réel , entier , alphanumérique continu , alphanumérique discret.

f) par 1 D , par 2 D , par 1 I , par 2 I , ont des valeurs attribuées par calcul et en fonction, respectivement de type D et de type I .

5.3.2 "Segment" et ses propriétés.

a) L'association "direct function" ne peut exister entre une entité "segment" et une entité "statistic description" que si type D de celle-ci est égal à "réel" et si type S de "segment" est égal à "direct" . De même pour l'association "inverse function" il faut que le type I égale "réel" et type S égale "inverse", (celà recouvre la contrainte : une entité "segment" ne peut intervenir à la fois dans uns association "direct function" et dans une association "inverse function").

b) type S ne peut prendre que deux valeurs : direct et inverse .

c) On a toujours:

$$0 \leqslant \text{valeur S} \leqslant 1$$

et la somme des "valeurs S" pour tous les "segment" associés à une même "statistic description" et de même type S est égale à $1 - F(0)$ D si type S = "direct" et à $1 - F(0)$ I si type S = "inverse."

d) La valeur de maxclasse d'un "segment" doit être comprise entre I D et J D si type S = "direct" , entre I I et J I si type S = "inverse". Pour tous les segments de même type S associés à une même "statistic description" on ne peut avoir deux fois la même valeur de maxclasse et il y a toujours une et une seule fois la valeur J D si type S = direct, J I si type S = inverse .

5.3.3 - "Proportion" et ses propriétés.

a) L'association "proportion function" entre "proportion" et "statistic description" ne peut exister que si type P de cette dernière est connu. Dans ce cas il y a au moins deux entités "proportion" associées à la même entité "statistic description".

b) Pour toutes les entités "proportion" associées à une même "statistic description" il ne peut y avoir deux fois la même valeur de T .

c) En fonction de la valeur de type P de la "statistic description" associée par "proportion function" à l'entité "proportion" , T doit avoir une valeur

- de type réel si type P est réel,
- de type entier si type P est entier,
- de type alphanumérique sinon.

6 - SPECIFICATION DE PACKAGE.

6.1 - Points d'entrée.

Nous allons définir sept points d'entrées dans ce package selon que l'on veut calculer:

- la probabilité d'un critère de relation,
- les valeurs de m' et m" associées à une boucle d'accès sans ordre "next" et "exit",
- ces mêmes valeurs s'il y a des ordres "next" et "exit",
- les valeurs m_1 et m_2 servant à déterminer le coût d'une condition,
- le nombre ν_B d'éléments dans une collection,
- la probabilité p_k d'appartenance à une variable de travail,
- la probabilité d'un critère d'appartenance.

6.2 - Spécification

6.2.1 - Arguments.

a) L'argument ident donne l'identification; c'est un tableau de dimension 3 :

- ident (1) donne la databasekey de la "statistic description" concernée,

- ident (2) indique que la"statistic descrption" est associée à un "item" alors : ident (2) = 1 , ou à un "path-type" et ident (2) = 0 ,
- ident (3) donne le sens de la relation, direct alors ident (3) = 1 ; ou inverse et indent (3) = -1.

b) L'argument p est égal à la probabilité de la condition C_B d'où $0 \leqslant p \leqslant 1$.

c) L'argument car détermine si le cas considéré est un cas cardinal (i - j) alors car a la valeur vrai, où si c'est un cas ordinal (iE - jE) et car a la valeur faux.

d) Les arguments i et j sont des entiers tels que :

si $< 0 >$ = iE - jE où $< K >$ = i - j

on a : $0 \leqslant i \leqslant j$

si car = vrai et

$< K >$ = TS alors i = -1 , j = -1

$< K >$ = i- alors $0 \leqslant i$, j = -1

$< K >$ = j+ alors i = -1 , $0 \leqslant j$

si car = faux et

$< 0 >$ = iE - DE alors $i \geqslant 0$, j = -1

$< 0 >$ = DE alors i = j = -1

(on donne donc à i et à j la valeur -1 lorsqu'il n'y a pas de valeur explicite de ce nombre).

e) L'argument compt indique l'existence ou non d'un compteur de cibles. Il a la valeur vrai en cas d'existence.

f) L'argument p_N donne, dans le cadre d'une boucle d'accès, la probabilité d'exécuter un ordre "next".

6.2.2 - Probabilité d'un critère de relation.

L'entrée se fait par un appel à

probcd (ident, p, car, i, j, res)

Dans ce cas on a :

$$res = pr (r : \langle 0 / K \rangle \; B (C_B))$$

et pr est donné par les formules se trouvant en annexe II .

6.2.3 - Boucle d'accès sans ordre next ni exit.

L'entrée se fait par un appel à

boucle (ident, p, car, i, j, compt, res)

Si la boucle s'écrit

$$\left[r : \langle K / 0 \rangle \; B (C_B) \; S \right]$$

alors

$$res (1) = m'$$

$$res (2) = m''$$

m' et m" étant donnés par les formules en annexe II.

6.2.4 - Boucle d'accès avec ordre next et exit.

L'entrée se fait par un appel à

nextex (ident, p, car, i, j, compt, p_N, res)

Dans ce cas on doit avoir

$$0 \leqslant p_N < 1$$

Si la boucle s'écrit

$$\left[r \ : \ < K / 0 > \ B \ (C_B) \ S \right]$$

alors

res (1) = m'

res (2) = m"

m' et m" étant donnés par les formules en annexe II.

6.2.5 - Coût d'évaluation d'une condition.

L'entrée se fait par un appel à

coutcd (ident, p, car, i, j, compt, res)

Si la condition s'écrit

$$(r \ : \ < K / 0 > \ B \ (C_B)) .$$

alors

res (1) = m_1

res (2) = m_2

où m_1 et m_2 sont donnés par les formules en annexe II.

6.2.6 - Nombre d'éléments distincts dans une collection.

L'entrée se fait par un appel à

elcoll (ident, p, car, i, j, compt, p_N, p_0, res)

Si on note R (A,B) la relation servant à constituer la collection, l'argument p_0 est la probabilité qu'une réalisation de A appartienne au sous-ensemble A* de A, A* servant d'ensemble de départ pour la boucle

$$\left[r \ : \ < 0 \ / \ K > \ B \ (C_B) \right]$$

servant à créer la collection.

On doit avoir :

$$0 \leqslant p_0 \leqslant 1$$

Les arguments p, car, i, j, compt, p_N, se rapportent à la boucle citée ci-dessus.

Alors

$$res = \frac{\nu'_B}{N_B}$$

où N_B est le nombre d'occurences de B et où $\frac{\nu'_B}{N_B}$ est donné par les formules en annexe II.

6.2.7 - Nombres de références distinctes dans une variable de travail.

L'entrée se fait par un appel à

vatrav (ident, p, car, i, j, compt, p_N, p_0, p_k, res)

Les arguments p, car, i, j, compt, p_N, p_0 ont la même signification que ci-dessus. L'argument p_k est la probabilité qu'une occurence de B appartienne déjà à la variable de travail. On doit donc avoir :

$$0 \leqslant p_k \leqslant 1$$

alors

$$res = p_{k+1} = p_k + \frac{\nu''_{k+1}}{N_B} \ (1 - p_k)$$

où $\frac{\nu'_{k+1}}{N_B}$ est la probabilité d'appartenance d'une occurence de B à la collection ajoutée à la variable de travail.

6.2.8 - Probabilité d'un critère d'appartenance.

L'entrée se fait par un appel à

probap (ident, op, T A , T B , N, res)

L'argument op donne l'opérateur logique :

$= a$, op = 1

$\neq a$, op = 2

$< a$, op = 3

$\leq a$, op = 4

$> a$, op = 5

$\geq a$, op = 6

$\in [a,b]$, op = 7

Les arguments TA et TB sont des tableaux d'entiers de dimension N 50.

Si ident (2) = 0 (cas de type de chemin), les arguments TA, TB, et N ne sont d'aucune utilité.

Si ident (1) = 1 (cas d'un item), l'argument TA doit toujours avoir une valeur correspondant au type de valeur de l'item, TB ne doit avoir une valeur que si op = 7 , elle correspond au type de valeur de l'item .

Si l'item est de type alphanumérique de longueur l , alors on a

N = l

et pour tout i : $1 \leq i \leq N$

TA (i) = $i^{ème}$ caractère de la chaine a

Dans le cas où l'item ne peut prendre qu'un nombre limité de valeurs, op ne peut prendre que les valeurs 1 et 2 .

Si l'item est de type entier, alors

N = 1 et

TA (1) est égal à a .

Si l'item est de type réel, alors

N = 2 et

TA (1) et TA (2) sont deux entiers

tels que

$$a = TA(1) * 10^{TA(2)}$$

Dans ce cas **op** ne peut prendre les valeurs 1 ou 2 .

L'argument TB est défini de la même manière.

Alors on a

res = pr (op E)

où E représente soit a , soit $[a,b]$.

7 - A R C H I T E C T U R E

7.1 - Hiérarchisation et modularisation.

Nous allons définir, pour le système à développer, une hiérarchie "utilise" à six niveaux :

- niveau 1 : module d'accès à la base de données,
- niveau 2 : module de traitement des valeurs associées à F ,
  module de traitement des classes de F ,
- niveau 3 : module de calcul de C_m^n ,
  module de traitement des valeurs d'item,
- niveau 4 : module de calcul de (k, l, N, p),
  module de traitement de la fonction E ,
- niveau 5 : module de calcul des sommes (somf *)
- niveau 6 : module de calcul de la probabilité d'un critère de relation,
  module de calcul de la probabilité d'un critère d'appartenance,
  module de calcul du coût d'évaluation d'un critère de relation,
  module des calculs concernant les boucles.

Un schéma figure en annexe IV .

7.2 - Spécification des modules.

7.2.1 - Modules d'entrée.

Ces modules ont été spécifiés au point 6.2 . Les deux points d'entrée relatifs aux boucles d'accès et ceux relatifs au nombre d'éléments distincts dans une collection ainsi qu'au nombre de références distinctes dans une variable de travail , ont été regroupés en un seul module.

7.2.2 - Module somme.

Ce module calcule des sommes du type :

$$\sum_{k=a}^{b} F(k) \quad \text{coeff}(k)$$

La liste de ces sommes est reprise en 3.1. Les résultats sont donnés par ces formules.

Toutes ces sommes ont un argument en commun: stat , qui contient les neuf valeurs réelles associées à la fonction statistique F .

L'argument p est une valeur de probabilité tel que

$$0 \leqslant p \leqslant 1$$

L'argument p_N est une valeur de probabilité tel que :

$$0 \leqslant p_N < 1$$

Les arguments x, y, z, sont des entiers positifs et

on doit avoir :

$$y \leqslant J$$

Les argument sont, pour les sommes

somf 1, somf 3, somf 4 et somf 7 : stat,x, y, p.

somf 5 : stat, x, y.

somf 10, somf 13 : stat, x, y, p, p_N.

somf 11, somf 16 : stat, x, y, p_N .

somf 15 : stat, x, y, z, p.

où

$$x \leqslant y, \text{ ou } x > J$$

Pour les sommes somf 2 et somf 6 les arguments sont : stat, x et y , où

$$x \leqslant y + 1 , \text{ ou } x > J$$

Les arguments sont, pour les sommes

somf 8 : stat et p.

somf 9 : stat, p et p_N.

somf 12 : stat et p_N.

somf 14 : stat et y.

7.2.3 - Module pi .

Ce module calcule la valeur de $\Pi(k, 1, p, N)$ et les sommes de $\Pi(k, 1, p, N)$. Les sommes ont été données en 3.1. avec leurs résultats.

L'argument p est une valeur telle que :

$$0 \leqslant p \leqslant 1$$

La somme somp 1 a pour arguments i, j, k, p, N où i, j, k, N sont des entiers positifs tels que :

$$i \leqslant j \leqslant k \leqslant N$$

La somme somp 2 a pour arguments i, j, p, N où i, j, N sont des entiers positifs tels que

$$i \leqslant j \leqslant N$$

Le calcul pi a pour arguments k, l, p, N où k, l, N sont des entiers positifs tels que

$$l \leqslant k \leqslant N$$

7.2.4 - Module fonction F.

Ce module traite tout ce qui concerne la fonction F. Il est connu par ses interfaces :

1) fonct (ident, stat) :
a pour argument ident donnant l'identification de la fonction F concernée par trois entiers.
Il a pour résultat stat donnant neuf valeurs réelles associées à la fonction F. Ces valeurs dépendent du type de la fonction F. Elles sont calculées et la base de données est mise à jour.

2) fk (stat, k, lim, valf) :
calcule la valeur de la fonction.
Ses arguments sont :
- stat donnant les neuf valeurs réelles associées à la fonction,
- une entier k donnant la valeur du début de la classe recherchée,
- un entier lim égal à la limite supérieure de la classe précédente ou à 0 si on calcule fk pour

la première classe. On doit avoir

$$I \leqslant k \leqslant J \quad \text{et} \quad 0 \leqslant \text{lim} < k$$

Ses résultats sont :

- l'entier lim égal à la limite supérieure de la classe pour laquelle on a calculé F (k). On doit avoir

$$\text{lim} \geqslant k$$

- valf donnant la probabilité pour la classe comprise entre k et la nouvelle valeur de lim .

7.2.5 - Module C_m^n .

Ce module calcule C_m^n . Il a comme arguments deux entiers m et n tels que

$$0 \leqslant n \leqslant m$$

Son résultat est la valeur de

$$C_m^n \left(\frac{m!}{n!\,(m-n)!}\right)$$

7.2.6 - Module valeur d'item.

Ce module traite des valeurs des items. Il est connu par ses interfaces.

Les arguments sont:

- type , donnant le type de l'item,
- N , entier , qui donne la longueur de l'item; les conditions sont :

si type = 1 , N = 2

si type = 2 , N = 1

si type = 3 ou 4 , $1 \leqslant N \leqslant 50$

- A , B des valeurs possibles de l'item.

Alors on a les interfaces :

- sup (A, B, N, type)

  à la valeur vrai si A > B

  condition type = 1, 2 ou 3.

- égal (A, B, N)

  à la valeur vrai si A = B

  condition type = 2, 3 ou 4.

- affect (A, B, N)

  donne à A la valeur de B .

- nbrval (A, B, N, type)

  donne le nombre de valeurs d'item comprises entre A et B .

  condition non sup (A, B, N, type) et type = 1, 2 ou 3.

7.2.8 - Module statistique.

Ce module donne les valeurs associées à la fonction F. Il est connu par ses interfaces.

L'argument stat contient les neuf valeurs réelles associées à F . L'argument p est un réel positif et l'argument k un entier positif .

Les interfaces sont:

- typ

  argument : stat

  résultat : entier donnant le type de la fonction F.

- moy

  argument : stat

  résultat : moyenne de F .

- F0

  argument : stat

  résultat : valeur de F en 0 .

- k0

  argument : stat

  résultat : entier égal à la valeur de k_0 pour F .

- par 1

  argument : stat

  résultat : premier paramètre de F .

- par 2

  argument : stat

  résultat : deuxième paramètre de F .

- IS

  argument : stat

  résultat : entier égal à la valeur de I pour F .

- JS

  argument : stat

  résultat : entier égal à la valeur de J pour F .

- NC

  argument : stat

  résultat : entier égal au nombre de cibles pour la relation à laquelle est associée F .

- ecmoy

  argument : stat, p

  résultat : la valeur de la moyenne de F dans stat est égale à p .

- ecfo

  argument : stat, p

  résultat : la valeur de F (0) dans stat est égale à p .

- ecpar 1

  argument : stat, p

  résultat : la valeur du premier paramètre de F dans stat est égale à p .

- ecpar 2

  argument : stat, p

  résultat : la valeur du deuxième paramètre de F dans stat est égale à p .

- ec IS

argument : stat, k

résultat : la valeur de ,I pour F dans stat est égale à k .

- ec JS

argument : stat, k

résultat : la valeur de J pour F dans stat est égale à k .

7.2.9 - Module classe.

Ce module donne les valeurs associées à une classe de F . Il est connu par ses interfaces.

L'argument seg contient les trois valeurs associées à une classe de F :

- maxcl

argument : seg

résultat : entier égal à la limite supérieure de la classe.

- val

argument : seg

résultat : la valeur de la fonction F pour la classe.

7.2.10 - Module d'accès.

Ce module traite de tous les accès à la base de données. Il est connu par ses interfaces.

- accfct (ident, stat)

argument : ident, qui donne l'identification de la fonction F par trois entiers.

résultat : stat contenant les neuf valeurs réelles associées à la fonction F. Les valeurs inconnues sont égales à -1 ; la valeur du type de la fonction F est :

0 pour le type réel,
1 " " binomial,
2 " " géométrique 0,
3 " " géométrique 1,
4 " " de Poisson,
5 " " uniforme (I,J),
6 " " uniforme (F (0) J),
7 " " uniforme (m, I=0),
8 " " uniforme (m, I=1),
9 " " décroissant,
10 " " à maximum 1 ,
11 " " à maximum 2 ,
12 " " J = 1
13 " " J = 2

- accprm (seg) : accès au premier segment,

et

- accprs (seg) : accès au segment suivant,

résultat : seg qui donne les valeurs associées à la classe

- écrit (ident, stat)

  argument :ident et stat

  résultat : les valeurs contenues dans stat sont mémorisées dans la base de données.

- acctyp (ident, typ)

  argument : ident

  résultat : typ donnant la valeur entière correspondant au type de proportion. Si le type n'est pas déterminé, typ a une valeur quelconque. Si l'item est de type réel, typ = 1 ; de type entier, typ = 2 ; de type alphanumérique et compris entre deux limites, typ = 3 ; de type alphanumérique à valeurs limitées, typ = 4 .

- accprp (T, N, pr) : accès à la première proportion, et

- accprs (T, N, pr) : accès à la proportion suivante

  argument : N , longueur de l'item considéré

  résultat : T , valeur d'item, limite supérieure de la classe, et pr proportion de cette classe.

- invfct (ident)

  argument : ident

  résultat : ident où le sens de la retation est inversé.

8 - A L G O R I T H M E S

8.1 - Modules d'entrée.

Ces algorithmes se basent sur les algorithmes de calcul développés au chapitre 3.

8.1.1 - probcd (ident, p, car, i, j, res):

```
fonct ( ident, stat )
si  j  >  JS ( stat )    j  =  JS ( stat )
si  car  =  vrai
   si  i  ≠  -1  ,  j  ≠  -1 ,     x  =  i ,  y  =  j
   si  i  ≠  -1  ,  j  =  -1 ,     x  =  0 ,  y  =  i - 1
   si  i  =  -1  ,  j  ≠  -1 ,     x  =  0 ,  y  =  j
   si  i  =  -1  ,  j  =  -1 ,
      x  =  1
      y  =  JS ( stat )
      si  p  =  1 ,  res  =  1 -  fo ( stat )
            sinon    res  =  somf3 ( stat, x, y, p )
   sinon
      si  p  =  1 ,  res  =  somf2 ( stat, x, y )
```

```
                sinon   res  =  somf1 ( stat, x, y, p )
        si  j  =  -1 , res  =  1  -  res
si  car  =  faux
    si  i  =  j
        si  i  =  1 , ou   i  =  -1
            res  =  p ( 1 - fo ( stat ) )
        sinon
            x  =  0
            y  =  i - 1
            res  =  p ( 1 - somf2 ( stat, x, y ) )
    sinon
        si  j  =  -1        y  =  JS ( stat )
                    sinon   y  =  j
        x  =  i
        res  =  somf3 ( stat, x, y, p )
```

8.1.2 - Boucle (ident, p, car, i, j, compt, res)

```
fonct  ( ident, stat )
si  j  >  JS ( stat ) ,  j  =  JS ( stat )
si  car  =  vrai  ou  ( car = faux,  i = 1  et  j = -1 )
    res (1)  =  p  moy ( stat )
    res (2)  =  moy ( stat )
sinon
    si  i  =  -1 ,      x  =  JS ( stat )
                sinon   x  =  i
```

```
si  j  =  -1 ,      y  =  JS ( stat )
          sinon     y  =  j
si  p  =  1 ,       res (1)  =  somf5 ( stat, x, y )
          sinon     res (1)  =  somf4 ( stat, x, y, p )
si  j  =  -1   et   compt  =  faux
   res (2)  =  moy ( stat )
sinon
   si  i  =  j  =  1   et   p  ≠  1
      res (2)  =  moy ( stat ) + somf8 ( stat, p )
   sinon
      si compt  =  faux ,     x  =  1
      res (2)  =  somf6 ( stat, x, y )
      si  p  =  1
         x  =  y  +  1
         y  =  JS ( stat )
         aux  =  ( x - 1 ) somf2 ( stat , x, y )
      sinon
         x  =  y
         y  =  JS ( stat )
         aux  =  somf7 ( stat, x, y, p )
      res (2)  =  res (2)  +  aux
```

8.1.3 - Nextex (ident, p, car, i, j, compt, pN, res)

```
fonct  ( ident , stat )
si  j  >  JS ( stat ),          j  =  JS ( stat )
si  car  =  vrai   ou   ( car  =  faux  et  i = 1  et j = -1 )
```

```
   si  p  =  1
      res (1)  =  somf12 ( stat, pN )
      res (2)  =  res (1)
   sinon
      x  =  1
      y  =  JS ( stat )
      res (1)  =  1 / ( 1 - pN ) - somf9 ( stat, y, p, pN )
      res (2)  =  somf13 ( stat, x, y, p, pN )
sinon
   si  i  =  -1 ,      x  =  JS ( stat )
               sinon  x  =  i
   si  j  =  -1 ,      y  =  JS ( stat )
               sinon  y  =  j
   si  p  =  1 ,
      res (1)  =  somf11 ( stat, x, y, pN )
      res (2)  =  somf16 ( stat, x, y, pN )
   sinon
      res (1)  =  somf10 ( stat, x, y, p, pN )
      res (2)  =  somf13 ( stat, x, y, p, pN )
   si compt  =  vrai
      y  =  x - 1
      x  =  1
      res (2)  =  res (2)  +  somf6 ( stat, x, y )
```

8.1.4 - elcoll (ident, p, car, i, j, compt, pN, po, res)

```
si  car  =  faux
```

```
   fonct ( ident, stat )
   mu  =  moy ( stat )
   si  pN  =  1
      boucle ( ident, p, car, i, j, compt, resaux )
   sinon
      nextex ( ident, p, car, i, j, compt, pN, resaux )
invfct ( ident )
fonct ( ident, stat )
res  =  1  -  somf1 ( stat, 0, 0, po )
si  car  =  vrai
   res  =  p  res
sinon
   res  =  resaux (1)  res  /  mu
```

8.1.5 - vatrav (ident, p, car, i, j, compt, pN, po, pk, res)

```
elcoll ( ident, p, car, i, j, compt, pN, po, res )
res  =  pk  +  res  ( 1 - pk )
```

8.1.6 - coutcd (ident, p, car, i, j, compt, res)

```
fonct ( ident, stat )
si  j  >  JS ( stat ),        j  =  JS ( stat)
si car  =  vrai
   si  p  =  1
      si compt  =  vrai
         res (1)  =  0
```

```
   sinon
      si  j  =  -1
         si  i  =  -1 ,     y  =  JS ( stat )
                    sinon   y  =  i - 1
      sinon
         y  =  j
      res (1)  =  somf14 ( stat, y )
sinon
   si  i  =  -1 et  j  =  -  1
      x  =  1
      y  =  JS ( stat )
      z  =  0
      res (1) =  somf15 ( stat, x, y, z, p )
   sinon
      y  =  JS ( stat )
      si  j  =  -1  ,      x  =  i - 1
                  sinon    x  =  j
      res (1)  = somf7 ( stat, x, y, p )
      si compt  =  faux
         y  =  x
         x  =  1
         aux  =  somf6 ( stat, x, y )
      sinon
         si  i  =  j
            aux  =  somf6 ( stat, i, i )
         sinon
```

```
                si  i ≠ -1  et  j ≠ -1
                   x = i - 1
                   y = j
                   aux = somf7 ( stat, x, y, p )
          res (1) = res (1) + aux
    res (2) = res (1)
si car = faux
    si  i = -1   et   j = -1
       res (1) = moy ( stat )
       res (2) = 1 - fo ( stat )
    sinon
       si  i = j
          x = i
          y = JS ( stat )
          res (2) = somf2 ( stat, x, y )
          res (1) = i  res (2)
       sinon
          si j = -1 ,       y = JS ( stat )
                      sinon  y = j
          x = i
          z = i - 1
          res (1) = somf15 ( stat, x, y, z, p )
          z = 0
          res (2) = somf15 ( stat, x, y, z, p )
       si compt = faux
          x = 1
          y = i - 1
          res (1) = res (1) + somf6 ( stat, x, y )
```

8.1.7 - probap (ident, op, TA, TB, N, res)

1) L'algorithme se base sur l'existence de deux fonctions :

- propeg (TA, N) donnant la probabilité d'égalité de l'item à la valeur donnée TA dans le cas où il n'y a pas de classe, c'est à dire si le type de l'item est égal à 4 .
- prop (TA, TB, N, type) calculant la probabilité qu'une valeur de l'item soit comprise entre TA et TB lorsque l'item peut prendre toutes les valeurs possibles entre deux valeurs extrêmes : cas pour lesquels type = 1, 2 ou 3 .

et sur les formules :

- pr (= TA) = pr ($\in$ [TA, TA])
- pr ($\neq$ TA) = 1 - pr (= TA)
- pr ($\leqslant$ TA) = pr ($\in$ [min, TA])
- pr ($<$ TA) = pr ($\leqslant$ TA) - pr (= TA)
- pr ($>$ TA) = 1 - pr ($\leqslant$ TA)
- pr ($\geqslant$ TA) = 1 - pr ($\leqslant$ TA) + pr (= TA)

c'est à dire qu'on se ramène toujours à la probabilité d'appartenance à un intervalle. Dans certains cas il est nécessaire d'accéder à la première classe pour trouver la valeur minimale.

D'où l'algorithme :

acctyp (ident, type)

si type = 1, 2 ou 3,

```
si  op  =  1 ,  res  =  prop ( TA, TA, N, type )
si  op  =  2 ,  res  =  1  -  prop ( TA, TA, N, type )
si  op  =  3 ,  accprp ( TP, N, pr )
                res  =  prop ( TP, TA, N, type )
                        - prop ( TA, TA, N, type )
si  op  =  4 ,  accprp ( TP, N, pr )
                res  =  prop ( TP, TA, N, type )
si  op  =  5 ,  accprp ( TP, N, pr )
                res  =  1 - prop ( TP, TA, N, type )
si  op  =  6 ,  accprp ( TP, N, pr )
                res  =  1 - prop ( TP, TA, N, type )
                        + prop (  TA, TA, N, type )
si  op  =  7 ,  res = prop ( TA, TB, N, type )
sinon
   si type  =  4
      si  op  =  1 ,  res  =  propeg ( TA, N )
      si  op  =  2 ,  res  =  1 - propeg ( TA, N )
   sinon
      accfct ( ident, stat )
      si  op  =  1 ,  res  =  1 / NC ( stat )
      si  op  =  2 ,  res  =  1 - 1 / NC ( stat )
```

2)propeg (TA, N)

Principe de l'algorithme :

Accéder aux différents articles proportion jusqu'à ce qu'on trouve la valeur d'item égale à TA . Le résultat

est alors donné par la valeur de cette proportion.
D'où :

```
accprp ( T, N, pr )          ( accès au premier )
tant que non égal  ( T, TA, N )          ( T ≠ TA )
   accprs ( T, N, pr )          ( accès au suivant )
propeg  =  pr
```

3) prop (TA, TB, N, type)

Principe de l'algorithme :

- accéder aux proportions jusqu'à trouver un T > TA on a alors une classe TA, T .
- pondérer la proportion trouvée par la taille réelle de la classe.
- accéder aux proportions suivantes jusqu'à trouver un T > TB :

  tant que T < TB , ajouter pr au résultat,
  quand T ⩾ TB , ajouter au résultat pr pondéré par la taille de la dernière classe.

d'où :

```
prop  =  0
accprp ( T, N, pr )          ( accès au premier )
si  non sup ( TA, T, N, type )     ( T ⩾ TA )
   affect ( TA, T, N )          ( TA  =  T )
si  non sup ( T, TB, N, type )          ( TB ⩾ T )
   affect ( Tpr, T, N )                 ( Tpr  =  T )
```

```
accprs ( T, N, pr )              ( accès au suivant )
tant que  sup ( TA, T, N, type )    ( TA > T )
   affect ( Tpr , T, N )
   accprs ( T, N, pr )
aux  =  aux  nbrval ( TA, T, N, type )
                / nbrval ( Tpr, T, N, type )
affect ( Tpr, TA, N )            ( Tpr  =  TA )
tant que  sup ( TB, T, N, type )    ( TB > T )
   prop  =  prop  +  aux
   affect ( Tpr, T, N )          ( Tpr  =  T )
   accprs ( T, N, pr )           ( accès au suivant )
   aux  =  pr
prop  =  prop + aux  nbrval ( Tpr, TB, N, type)
                    /nbrval ( Tpr, T, N, type ).
```

8.2 - Module somme.

Toutes les sommes sont basées sur le même principe :

- initialisation :
  - somf* = 0 (1 pour somf12, y+1 pour somf14)
  - lim = 0
  - initialisation de k
  - si k < I , k = I
- tant que k $\leqslant$ J (y pour somf2, somf6, somf14)
  - fk (stat, k, lim, valf)
  - pondération de valf si lim > y dans le cas k $\leqslant$ y
  - si valf $\neq$ 0

calcul du coéfficient de F pour k = (k + lim) / 2
(au milieu de la classe)

somf* = somf* + valf coefficient

k = lim + 1

d'où :

8.2.1 - somf1 (stat, x, y, p)

somf1 = 0 , lim = 0

k = x ; si k < IS (stat), k = IS (stat)

tant que k ⩽ JS (stat)

fk (stat, k, lim, valf)

si valf ≠ 0

si k ≠ lim , k = (k + lim) / 2

si k > y , l = y , sinon l = k

somf1 = somf1 + valf somp1 (x, l, k, p, NC (stat))

k = lim + 1

8.2.2 - somf2 (stat, x, y.)

somf2 = 0 , lim = 0

k = x , si k < IS (stat), k = IS (stat)

tant que k ⩽ y

fk (stat , k, lim, valf)

si lim > y , valf = valf (y-k+1) / (lim-k+1)

somf2 = somf2 + valf

k = lim + 1

8.2.3 - somf3 (stat, x, y, p)

```
somf3 = 0 ; lim = 0
k = x ;  si k < IS ( stat ), k = IS ( stat )
tant que k ≤ JS ( stat )
   fk ( stat, k, lim, valf )
   si valf ≠ 0
      si k ≠ lim , k = (k + lim ) / 2
      si k > y   , k = y
      k = k - x + 1
      somf3 = somf3 + valf pi ( k, k, p, NC (stat) )
   k = lim + 1
```

8.2.4 - somf4 (stat, x, y, p)

```
somf4 = 0 ; lim = 0
k = x ; si k < IS ( stat ), k = IS ( stat )
tant que k ≤ JS ( stat )
   fk ( stat, k, lim, valf )
   si valf ≠ o
      si k ≠ lim , k = ( k + lim ) / 2
      sompi = 0
      pour l = x à k
         si l < y , coeff = l , sinon coeff = y
         coeff = coeff - x + 1
         sompi = sompi + coeff pi ( k, l, p, NC ( stat ) )
      somf4 = somf4 + valf sompi
   k = lim + 1
```

8.2.5 - somf5 (stat, x, y)

```
somf5  =  0 ;  lim  =  0
k  =  x ;  si  k  <  IS ( stat ) ,  k  =  IS ( stat )
tant que  k  ≤  JS ( stat )
   fk ( stat, k, lim, valf )
   si valf  ≠  0
      si  k  ≠  lim ,  k  =  ( k + lim ) / 2
      si  k  >  y   ,  k  =  y
      somf5  =  somf5  +  valf  (k - x + 1)
   k  =  lim + 1
```

8.2.6 - somf6 (stat, x, y)

```
somf6  =  0 ;  lim  =  0
k  =  x ;  si  k  <  IS ( stat ) ,  k  =  IS ( stat )
tant que  k  ≤  y
   fk ( stat, k, lim, valf)
   si  valf  ≠  0
      si  lim  >  y , valf  =  valf ( y - k + 1 ) / (lim-k+1 )
      si  k  ≠  lim , k  =  ( k + lim ) / 2
      somf6  =  somf6  +  k valf
   k  =  lim  +  1
```

8.2.7 - somf7 (stat, x, y, p)

```
somf7  =  0 ;  lim  =  0
k  =  x + 1 ;  si  k  <  IS ( stat ),  k  =  IS ( stat )
```

```
tant que  k ⩽ JS ( stat )
   fk ( stat, k, lim, valf )
   si valf ≠ 0
      si  k ≠ lim ,  k  =  ( k + lim ) / 2
      sompi1  =  somp1  ( 0, x-1, k, p, NC ( stat ) )
      sompi2  =  somp2  ( x, k, p, NC ( stat ) )
      somf7   =  somf7 + valf ( k sompi1 + x ( k+1 ) sompi2 )
   k  =  lim  +  1
```

8.2.8 - somf8 (stat, p)

```
somf8  =  0 ;  lim  =  0
k  =  2 ;  si  k  <  IS (stat ),  k  =  IS ( stat )
tant que  k ⩽ JS ( stat )
   fk ( stat, k, lim, valf )
   si  valf ≠ 0
      si  k ≠ lim ,  k  =  ( k + lim ) / 2
      sompi  =  0
      pour  l  =  1  à  k
         sompi = pi ( k, l, p, NC(stat) )( 1- l( k+1)/( l+1 ) )
                 + sompi
      somf8  =  somf8  +  valf sompi
   k  =  lim  + 1
```

8.2.9 - somf9 (stat, y, p, pN)

```
somf9  =  0 ;  lim  =  0
k  =  0 ;  si  k  <  IS ( stat ),  k  =  IS ( stat )
```

```
tant que  k ≤ JS ( stat )
   fk ( stat, k, lim, valf )
   si  valf ≠ 0
      si  k ≠ lim ,  k  =  ( k + lim ) / 2
      sompi  =  0
      pour  l  =  0 à k
         si  l  >  y ,  a  =  y  sinon  a  =  l
         sompi  =  sompi + pi ( k, l, p, NC( stat )) pN^a
      somf9  =  somf9  +  valf  sompi
   k  =  lim  +  1
somf9  =  somf9  /  ( 1 - pN )
```

8.2.10 - somf10 (stat, x, y, p, pN)

```
somf10  =  0 ,  lim  =  0
k  = x ;  si  k  <  IS ( stat ),  k  =  IS ( stat )
tant que  k ≤ JS ( stat )
   fk ( stat, k, lim, valf )
   si valf ≠ 0
      si  k ≠ lim,  k  =  ( k + lim ) / 2
      sompi  =  0
      pour  l  =  x  à  k
         si  l  >  y ,  a  =  y,  sinon  a  =  l
         sompi =  sompi + pi( k, l, p, NC (stat) ) ( 1 - pN^a )
      somf10  =  somf10  +  sompi  valf
   k  =  lim  +  1
somf10  =  somf10 / ( 1 - pN )
```

8.2.11 - somf11 (stat, x, y, pN)

```
somf11 = 0 ; lim = 0
k = x ; si k < IS ( stat ) , k = IS ( stat )
tant que k ≤ JS ( stat)
   fk ( stat, k, lim, valf )
   si valf ≠ 0
      si k ≠ lim , k = ( k + lim ) / 2
      si k > y   , k = y
      somf 11 = somf11 + valf ( 1 - pN^(k-x+1) )
   k = lim + 1
somf11 = somf11 / ( 1 - pN )
```

8.2.12 - somf12 (stat, pN)

```
somf12 = 1 ; lim = 0
k = 0 ; si k < IS ( stat ) , k = IS ( stat )
tant que k ≤ JS ( stat )
   fk ( stat, k, lim, valf )
   si valf ≠ 0
      si k ≠ lim , k = ( k + lim ) / 2
      somf12 = somf12 - valf pN^k
   k = lim + 1
somf12 = somf12 / ( 1 - pN )
```

8.2.13 - somf13 (stat, x, y, p, pN)

```
somf13 = 0 ; lim = 0
k = x ; si k < IS ( stat ) , k = IS ( stat )
```

```
tant que  k  ≤ JS ( stat )
   fk ( stat, k, lim, valf )
   si  valf  ≠  0
      si  k  ≠  lim,  k  =  ( k + lim ) / 2
      sompi  =  k  sompi ( 0, x-1, k, p, NC ( stat) )
      si  k  >  y ,  a  =  y , sinon  a  =  k
      pour  l  =  x  à  a
         coeff =  ( k + 1 )( x - 1 + 1 / ( 1 - pN ))
                  - ( 1 - ( 1+k pN)/( 1 - pN ))pN^(l-x+1)
      sompi  =  sompi  +  coeff pi ( k, l, p, NC (stat))/(l+1)
   si  k  ≤  y + 1
      coeff  = ( k + 1 )( x - 1 + ( 1 - pN^(y-x+1))/( 1 - pN ) )
      sompi  =  sompi  +  coeff somp2 ( y+1, k, p, NC (stat))
   k  =  lim  +  1
```

8.2.14 - somf14 (stat, y)

```
somf14  =  y + 1  ;  lim  =  0
k  =  0 ;  si  k  <  IS ( stat ) ,  k  =  IS ( stat )
tant que  k  ≤  y
   fk ( stat, k, lim, valf )
   si valf ≠  0
      si  lim  >  y ,  valf  =  valf ( y-k+1 ) / ( lim-k+1 )
      si  k  ≠  lim ,  k  =  ( k  +  lim ) / 2
      somf14  =  somf14  -  valf ( y+1-k )
   k  =  lim  +  1
```

8.2.15 - somf15 (stat, x, y, z, p.)

```
somf15 = 0 ; lim = 0
k = x ; si k < IS ( stat ) , k = IS ( stat )
tant que k ⩽ JS ( stat )
   fk ( stat, k, lim, valf )
   si valf ≠ 0
      si k ≠ lim , k = ( k + lim ) / 2
      si k > y , k = y
      k = k - x + 1
      sompi = z
      pour l = 0 à k-1
         sompi = sompi + pi ( k,l,p,NC(stat))(k+1) / ( k+1-l )
      sompi = sompi + k pi ( k, k, p, NC (stat) )
      somf15 = somf15 + valf sompi
   k = lim + 1
```

8.2.16 - somf16 (stat, x, y, pN)

```
somf16 = 0 ; lim = 0
k = x ; si k < IS ( stat), k = IS ( stat )
tant que k ⩽ JS ( stat )
   fk ( stat, k, lim, valf )
   si valf ≠ 0
      si k ≠ lim , k = ( k + lim ) / 2
      si k > y , k = y
      k = k - x + 1
      somf16 = somf16 + valf ( x - 1 + ( 1 - pN^k) / ( 1 - pN ))
   k = lim + 1
```

8.3. - Module fonction F .

8.3.1 - fonct (ident, stat)

```
accfct ( ident, stat )          ( accès à la statistic descrip-
                                                    -tion)
si  par1 ( stat ) = - 1         ( paramètres non calculés )
   calpar ( stat )              ( calcul des paramètres )
   ecrit ( ident, stat )        ( réécriture dans la B.D. )
```

calpar (stat)

Les valeurs des paramètres à calculer sont données dans le catalogue (4.1). D'où :

1) si typ (stat) = 1 (binomiale)

$p = moy (stat) / JS (stat)$

$F = (1 - p)^{JS (stat)}$

ecIS (stat, 0)

ecpar1 (stat, p)

ecf0 (stat, F)

2) si typ (stat) = 2 (géométrique 0)

Dans ce cas la valeur de J sera déterminée pour que $F (J) \leqslant 10^{-6}$.

$p = 1 / (1 + moy (stat))$

$$J = int \left(\frac{\ln 10^{-6} - \ln p}{\ln (1-p)} \right) + 1$$

ecIS (stat, 0)

ecJS (stat, J)

ecpar1 (stat, p)

ecf0 (stat, p)

3) si typ (stat) = 3 (géométrique 1)

Dans ce cas J est solution de $F(J) \leqslant 10^{-6}$

p = 1 / moy (stat)

$$J = \text{int}\left(\frac{\ln 10^{-6} - \ln p}{\ln (1-p)}\right) + 2$$

ecIS (stat, 1)

ecJS (stat, J)

ecpar1 (stat, p)

ecf0 (stat, 0)

4) si typ (stat) = 4 (Poisson)

Dans ce cas la valeur de J sera calculée par la formule

$$J = m + \text{coeff } \sqrt{m}$$

Ce coefficient a été déterminé en fonction de la valeur de m de telle manière que

$$F (J) \leqslant 10^{-6}$$

On a alors

pour $m < 5$, coeff = 8

pour $m < 9$, coeff = 7

pour $m < 40$, coeff = 6

pour $m \geqslant 40$, coeff = 5

D'où

F = exp (- moy (stat))

si moy (stat) $<$ 5 , coeff = 8

sinon

si moy (stat) $<$ 9 , coeff = 7

```
        sinon
            si  moy ( stat ) < 40 ,  coeff  =  6
            sinon  coeff  =  5
        J  =  int ( moy ( stat ) + coeff √ moy ( stat) )
        ecIS ( stat, 0 )
        ecJS ( stat, J )
        ecpar1 ( stat, 0 )
        ecf0 ( stat, F )
5 ) si typ ( stat ) = 5          ( uniforme  1 )
        p = 1 / ( JS ( stat ) - IS ( stat ) + 1 )
        m = ( IS ( stat ) + JS ( stat ) ) / 2
        si  I = 0 ,  F = p , sinon  F  =  0
        ecpar1 ( stat, p )
        ecmoy ( stat, m )
        ecf0 ( stat, F )
6 ) si typ ( stat ) = 6          ( uniforme  2 )
        p = ( 1 - F0 ( stat )) / JS ( stat )
        m = ( 1 - F0 ( stat )) ( JS ( stat) + 1 ) / 2
        ecIS ( stat, 0 )
        ecpar1 ( stat, p )
        ecmoy ( stat, m )
7 ) si typ ( stat ) = 7          ( uniforme  3 )
        J  = int ( 2 moy ( stat ))
        p  = 1 / ( 2 moy ( stat ) + 1 )
        ecIS  ( stat, 0 )
        ecJS  ( stat, J )
        ecpar1 ( stat, p)
        ecf0 ( stat, p )
```

8) si typ (stat) = 8 (uniforme 4)

J = int (2 moy (stat)) - 1

p = 1 / (2 moy (stat) - 1)

ecIS (stat, 1)

ecJS (stat, J)

ecpar1 (stat, p)

ecf0 (stat, 0)

9) si typ (stat) = 9 (décroissante)

Dans ce cas, l'équation donnant la valeur de e^b en 2.4.1, sera résolue par approximations successives (Newton Rapson) : si l'équation est $f(x) = 0$ et si x_o est une solution approchée, on a

$$x = x_o - \frac{f(x_o)}{f'(x_o)}$$

D'où :

f = 1 - F0 (stat)

mu = moy (stat)

J = JS (stat)

x = mu / (mu - f) (lère approximation)

xo = 0

tant que $|x - xo| \geqslant 10^{-6}$

xo = x

$$x = \frac{J\,(mu-f)\,xo^{J+1} - (J-1)\,mu\,xo^{J} - mu + J\,f}{(J+1)(mu-f)\,xo^{J} - mu\,J\,xo^{J-1} - mu + f\,(J+1)}$$

p = f (x-1) x^J / (x^J-1)

ecIS (stat, 0)

ecpar1 (stat, x)

ecpar2 (stat, p)

10) si typ (stat) = 10 (maximum 1)

L'équation donnée en 2.4.2 sera résolue par Newton Rapson, mais il faudra deux approximations successives.

D'où :

f = 1 - FO (stat)

mu = moy (stat)

J = JS (stat)

k = kO (stat)

p = mu - k f

q = mu - J f

x = mu / (mu - f) (1ère approximation)

xo = 0

tant que $| x - xo | \geqslant 10^{-6}$

xo = x

$$x = \frac{k\ p\ xo^{k+1} - (k-2)\ p\ xo^{k-1} - mu + f}{(k+1)\ p\ xo^{k} - (k-1)\ p\ xo^{k-2} - mu}$$

tant que $| x - xo | \geqslant 10^{-6}$

xo = x

$$x = \frac{\begin{array}{l} J\ p\ xo^{J+1} - (J-2)\ p\ xo^{J-1} - (J-k)\ mu\ xo^{J-k+1} \\ + (J-k-1)(mu - f)\ xo^{J-k} - (k-1)(q-f)\ xo^{k} \\ + (k-2)\ q\ xo^{k-1} \end{array}}{\begin{array}{l} (J+1)\ p\ xo^{J} - (J-1)\ p\ xo^{J-2} - (J-k+1)\ mu\ xo^{J-k} \\ + (J-k)(mu-f)\ xo^{J-k-1} - k (q-f)\ xo^{k-1} \\ + (k-1)\ q\ xo^{k-2} \end{array}}$$

$$p = \frac{f\,(x-1)\,x^{J-1}}{x^{J-1}\,(x+1) - x^{J-k} - x^{k-1}}$$

ecIS (stat, 0)

ecpar1 (stat, x)

ecpar2 (stat, p)

11) si typ (stat) = 11 (maximum 2)

On a de même

f = 1 - f0 (stat)

mu = moy (stat)

J = JS (stat)

k = k0 (stat)

p = 2 mu - f k

q = mu - J f

x = (p+f) / (p-f)

xo = 0

tant que $|x - xo| \geqslant 10^{-6}$

xo = x

$$x = \frac{\begin{array}{l}(J+1)\,k\,(p-f)\,xo^{J+2} - 2\,J\,k\,p\,xo^{J+1} \\ +\,(J-1)\,k\,(p+f)\,xo^{J} - 2\,k\,(q-f)\,xo^{k+1} \\ +\,2\,(k-1)\,q\,xo^{k} - 2\,q\end{array}}{\begin{array}{l}(J+2)\,k\,(p-f)\,xo^{J+1} - 2\,(J+1)\,k\,p\,xo^{J} \\ +\,J\,k\,(p+f)\,xo^{J-1} - 2\,(k+1)\,(q-f)\,xo^{k} \\ +\,2\,k\,q\,xo^{k-1} + 2\,(q-f)\end{array}}$$

$$p = \frac{f\,(x-1)\,x^{J}}{k\,(x-1)\,x^{J} - x^{k} + 1}$$

```
      ecIS ( stat, 0 )
      ecpar1 ( stat, x )
      ecpar2 ( stat, p )

12) si  typ ( stat ) = 12          ( J = 1 )
      mu  =  moy ( stat )
      f  =  f0 ( stat )
      si  mu  =  -1 ,   mu  =  1 - f
      si  f  =  -1 ,   f  =  1 - mu
   ecIS ( stat, 0 )
   ecJS ( stat, 1 )
   ecmoy ( stat, mu )
   ecpar1 ( stat, mu )
   ecf0 ( stat, f )
13) si  typ ( stat ) =  13          ( J = 2 )
      p1 = 2 ( 1 - F0 ( stat ) ) - moy ( stat )
      p2 = moy ( stat ) -  ( 1 - F0 ( stat ) )
      ecIS ( stat, 0 )
      ecJS ( stat, 2 )
      ecpar1 ( stat, p1 )
      ecpar2 ( stat, p2 )
```

8.3.2 - fk (stat, k, lim, valf)

La valeur de valf sera déterminée en fonction du type de la fonction. Si cette fonction est réelle la valeur de lim sera égale à la limite supérieure de la classe contenant la valeur k , valf sera la probabilité de la classe

k , lim . Si la fonction est d'un autre type, alors nous limiterons le nombre de classes à 30 , par conséquent si $J \leqslant 30$, la valeur de valf est donnée par la formule (voir catalogue) et lim est égal à k ; par contre, si $J > 30$, nous déterminerons les classes de la manière suivante :

- fonction de type uniforme : nous prendrons comme largeur de classe lc = (J / 30) + 1 ; nous commencerons une classe en k d'où lim = k + lc - 1 ; la dernière classe sera limitée à J . Nous aurons valf = p (lim - k + 1).
- autres fonctions : nous choisirons des classes d'une seule valeur autour de la valeur max , valeur de k pour laquelle F (k) est maximum, c'est à dire pour les valeurs comprises entre max - 4 et max + 4 (neuf valeurs), sauf si $max - 4 \leqslant 0$, dans ce cas nous,considérerons les valeurs 1 à 9. Dans cet intervalle, lim = k et valf est donné par la formule de F (k). En dehors de cet intervalle nous prendrons des classes de largeur lc = (J-9) / 21 + 1 , d'où lim est égal à k + lc - 1 mais est limité à J. Nous calculerons valf en sommant les valeurs de F (k) données par la formule de k à lim , sauf pour la fonction de type binomiale pour laquelle nous prendrons la valeur de F au milieu de la classe, multipliée par la valeur de la classe.

Nous nous servirons d'une fonction calculant des sommes d' exponentielles

$$\sum_{i=m}^{n} val^{i}$$

```
somex ( val, m, n )
aux = val^m
somex = aux
pour  i  =  m+1  à  n
      aux  =  aux val
      somex  = somex + aux .

D'où :
1 ) si typ ( stat ) = 0
    si  lim  =  0
      si  k  =  0  et  f0 ( stat ) ≠ -1
         valf  =  f0 ( stat )
      sinon
         si k = 1  et  f0 ( stat ) ≠ -1
            maxpre  =  0
         sinon
            maxpre  =  IS ( stat ) -1
         accprm ( seg )
         tant que  k  >  maxcl ( seg )
            maxpre  =  maxcl ( seg )
            accsvt ( seg )
         lim  = maxcl ( seg )
         valf = val ( seg ) ( lim - k + 1 ) / ( lim - maxpre )
   sinon
      accsvt ( seg )
      lim  =  maxcl ( seg )
      valf =  val ( seg )
```

```
2 ) si typ ( stat ) ≠ 0
      si k = 0 , valf = f0 ( stat )
      sinon :
3 ) si typ ( stat ) = 12 ou 13
      lim = k
      si k = 1 , valf = par1 ( stat )
      si k = 2 , valf = par2 ( stat )
4 ) si typ ( stat ) = 5, 6, 7 ou 8
      si J ≤ 30
         lim = k
         valf = par1 ( stat )
      sinon :
         lc = int ( J/30 ) + 1
         lim = k + lc - 1
         si lim > JS ( stat ) , lim = JS ( stat )
         valf = par1 ( stat ) ( lim - k + 1 )
5) si typ ( stat ) = 1, 2, 3, 4, 9, 10 ou 11
      si J ≤ 30
         lim = k
      sinon :
         lc = int ( ( J-9) / 21 ) + 1
         si typ ( stat ) = 1
            max = int ( moy ( stat )( JS ( stat )+1) / JS(stat))
         si typ ( stat ) = 2, 3, ou 9
            max = 0
         si typ ( stat ) = 4
            max = int ( moy ( stat ) )
```

```
si  typ  ( stat ) = 10 ou 11
   max  =  k0 ( stat )
si  max  -  4  ≤ 0 ,  max  ≠  5
si  k  <  max - 4
   lim  =  k + lc - 1
   si  lim  >  max - 5 ,  lim  =  max - 5
sinon :
   si  k  ≤  max + 4
        lim  =  k
   sinon:
      lim  =  k + lc - 1
      si  lim  >  JS ( stat ),  lim  =  JS ( stat )
si  typ ( stat )  =  1
   valk  =  ( k + lim ) / 2
   valk  =  Cmn ( JS ( stat ) , valk )
            ( par1 ( stat))^valk
            ( 1 - par1 ( stat ))^JS ( stat ) - valk
si  typ ( stat )  =  2
   valf  =  ( 1 - par1 ( stat ))^k
            - ( 1 - par1 ( stat ) )^lim + 1
si  typ ( stat )  =  3
   valf  =  ( 1 - par1 ( stat ) )^k-1
            - ( 1 - par1 ( stat ))^lim
si  typ (stat )  =  4
   aux  =  f0 ( stat )
   pour  ind  =  1 à k
      aux  =  aux moy ( stat )  /  ind
```

```
      valf  =  aux
      pour  ind  =  k + 1  à  lim
         aux  =  aux moy ( stat ) / ind
         valf  =  valf  +  aux
   si  typ ( stat )  =  9
      aux  =  1 / par1 ( stat )
      valf  =  somex ( aux, k, lim )
   si  typ ( stat )  =  10
      si  k0 ( stat )  ⩾  k
         aux  =  p
         valf  =  somex ( aux, k, lim )
                  / ( par1 ( stat )) k0( stat )
      sinon
         aux  =  1 / par1 ( stat )
         valf  =  somex ( aux, k, lim )
                  ( par1 ( stat ))k0 ( stat )
      valf  =  valf par2 ( stat )
   si  typ  ( stat )  =  11
      aux  =  1 / par1 ( stat )
      si  k0 ( stat )  ⩾  k
         valf  =  lim - k + 1 - somex ( aux, k, lim )
      sinon
         valf  =  somex ( aux, k, lim )
                  / (( par1 ( stat ) )k0 (stat) - 1 )
      valf  =  valf par2 ( stat )
```

8.4 - Module pi.

8.4.1 - somp1 (i, j, k, p, N)

A partir de la formule (4.3.1) on a :

somp1 = 0

pour l = i à j

somp1 = somp1 + pi (k, l, p, N)

8.4.2 - somp2 (i, j, p, N)

somp2 = 0

pour l = i à j

somp2 = somp2 + pi (j, l, p, N) / (l + 1)

8.4.3 - pi (k, l, p, N)

A partir des calculs effectués par Monsieur Hainaut, on peut dire que la valeur de la fonction hypergéométrique π (k, l, p, N) peut être approchée par la fonction binomiale

$$C_k^l \; p^k \; (1-p)^{k-l}$$

quand

$$p > 0.1$$

et

$$N > 500 \quad \text{ou} \quad k < N / 5$$

D'où

si p = 1

si k = l , pi = 1

sinon pi = 0

sinon

si p $\leqslant$ 0.1 ou (N $\leqslant$ 500 et k $\geqslant$ N/5)

si N p $<$ 1 , pi = 0

sinon

si N - N p $<$ k - 1 , pi = 0

sinon

pi = Cmn (N p, 1) Cmn (N - N p, k - 1)
/ Cmn (N, k)

sinon

pi = Cmn (k, 1) p^1 (1-p)$^{k-1}$

8.5 - Module C_m^n .

En vue d'éviter des nombres trop grands, la valeur de C_m^n sera calculée par

si m $>$ 40

$$f(m,n) = \frac{12}{\sqrt{2\pi}} \sqrt{\frac{n(m-n)}{m}} \left(\frac{m}{m-n}\right)^m \left(\frac{m-n}{n}\right)^n \frac{1}{1 + \frac{144\ n\ (\ m-n\ )}{12\ m + 1}}$$

sinon, par la formule classique :

$$\frac{m\ !}{n\ !\ (\ m-n\ )\ !}$$

dans laquelle on aura préalablement simplifié par la factorielle la plus grande n ! ou (m-n) !

D'où :

Cmn (m, n)

si m $\leqslant$ 40

```
si  m - n  >  n  ,  k = n
   sinon  k  =  m - n
Cmn  =  1
pour  l  =  1  à  k
   Cmn  =  Cmn ( m - k + l ) / l
sinon
   Cmn  =  f ( m, n )        ( voir ci dessus )
```

8.6 - Module valeur d'item.

8.6.1 - sup (T1, T2, N, type)

1) si type = 1

$A = T1(1)\ 10^{T1(2)}$

$B = T2(1)\ 10^{T2(2)}$

sup = (A > B)

2) si type = 2

sup = (T1 (1) > T2 (1))

3) si type = 3

```
ind  =  1
tant que  T1 ( ind ) = T2 ( ind) et  ind  <  N
   ind  =  ind  +  1
sup  =  T1 ( ind )  >  T2 ( ind )
```

8.6.2 - égal (T1, T2, N)

```
ind  =  1
tant que  T1 ( ind ) = T2 ( ind ) et ind  <  N
   ind  =  ind + 1
```

égal = (T1 (ind) = T2 (ind))

8.6.3 - affect (T1, T2, N)

pour ind = 1 à N
   T1 (ind) = T2 (ind)

8.6.4 - nbrval (T1, T2, N, type)

1.) si type = 1

Les valeurs sont réelles: le nombre de valeurs dans la classe T1, T2 est la distance entre les deux nombres. D'où

nbrval = T2 (1) $10^{T2\ (2)}$ - T1 (1) $10^{T1\ (2)}$

2) si type = 2

Le nombre de valeurs dans ce cas est le nombre d'entiers entre T1 (1) et T2 (1), y compris les extrémités. D'où

nbrval = T2 (1) - T1 (1) + 1

3) si type = 3

Dans ce cas, nous considérons 27 valeurs possibles pour les N caractères de la chaine.
D'où

ind = 1
tant que T1 (ind) = T2 (ind) et ind < N
   ind = ind + 1
nbrval = (T2 (ind) - T1 (ind) -1) 27^{N-ind}
tant que ind < N

ind = ind + 1

nbrval = nbrval + (26- T1 (ind) +T2(ind)) 27^{N-ind}

nbrval = nbrval + 2

8.7 - Module statistique.

L'algorithme se base sur la représentation de stat par un tableau de 9 nombres réels :

stat (1) donne le type de F

stat (2) " la moyenne de F

stat (3) " la valeur de F en 0

stat (4) " la valeur de k_o pour F

stat (5) " le premier paramètre de F

stat (6) " le deuxième paramètre de F

stat (7) " la valeur de I pour F

stat (8) " la valeur de J pour F

stat (9) " le nombre de cibles.

A partir de là, chaque interface donne lieu à une égalité.

8.8 - Module classe.

L'algorithme se base sur la représentation de seg par un tableau de trois nombres réels:

seg (1) donne le sens de la relation

seg (2) " la limite supérieure de la classe

seg (3) " la valeur de la probabilité.

D'où les deux interfaces se réduisent à des égalités.

9

9 - C O N C L U S I O N

Les travaux présentés par ce mémoire mettent à la disposition de ceux qui seront chargés du développement ultérieur du projet, les calculs de probabilités et de coûts qui devront leur permettre une évaluation aussi approchée que possible des performances des algorithmes d'accès.

Les résultats obtenus sont des moyennes . Il n'a pas paru nécessaire de développer des formules et des calculs relatifs aux écarts. En effet, celà aurait demandé, dans la description statistique de la base de données, une précision que l'utilisateur ne pourra pas fournir la plupart du temps.

Nous expérons avoir contribué de la sorte, de manière modeste mais efficace, au succès final du projet ISDOS .

B I B L I O G R A P H I E

J.L. HAINAUT - Evaluation d'une base de données par modèle probabiliste.

- Some tools for data independence in multilevel data base systems.

- Calcul des probabilités et calcul des coûts.

KNUTH - The art of programming I : Fundemental algorithms.

ANNEXE I

RESULTATS DES CALCULS.

Les calculs ont été faits pour les relations suivantes:

1) adjoint (PERSONNE , COURS) caractérisée par

m = 0.47

F(0) = 0.825

I = 0

J = 10

N_A = 385

N_B = 872

2) CCD (COURS , COURS-DONNE)

m = 2

F(0) = 0.0045

I = 0

J = 11

N_A = 872

N_B = 1724

3) titulaire (PERSONNE , COURS)

m = 2.28

F(0) = 0.61

I = 0

J = 27

N_A = 385

N_B = 872

Les calculs ont été effectués pour :

1) la fonction réelle (R)
2) l'approximation par la fonction de poisson (P)
3) l'approximation par la fonction géométrique 0 (G)
4) l'approximation par la fonction binomiale (B)
5) l'approximation par la fonction uniforme , cas I = 0, la moyenne étant connue (UM)
6) l'approximation par la fonction uniforme , cas I = 0 , J étant connu (UJ)
7) l'approximation par la fonction uniforme , cas I = 0 , F(0) et J étant connus (UF)
8) l'approximation par la fonction de type exponentielle décroissante (ED) .

1. pr(r : TS B (C_B))

1.1 adjoint

p	R	P	G
0.001	0.00005504	0.00029382	0.00068049
0.01	0.00055443	0.00294442	0.00682454
0.1	0.00597334	0.03007636	0.07027407
0.5	0.04409473	0.16556858	0.40485830
0.8	0.10241120	0.28528049	0.73126055

p	B	UM	UJ
0.001	0.00030481	0.00051546	0.00009100
0.01	0.00305420	0.00515464	0.00091827
0.1	0.03115961	0.05154639	0.01010101
0.5	0.17044080	0.25773196	0.09082031
0.8	0.29196277	0.41237113	0.32459121

p	UF	ED
0.001	0.00004354	0.00006369
0.01	0.00043939	0.00063456
0.1	0.00483285	0.00673883
0.5	0.04078125	0.04650011
0.8	0.10272960	0.10392881

1.2 CCD

p	R	P	G
0.001	0.00045036	0.00027094	0.00033356
0.01	0.00453607	0.00273395	0.00335570
0.1	0.04867106	0.02996361	0.03571429
0.5	0.32791016	0.23254416	0.24999859
0.8	0.66376244	0.53498476	0.57086110

p	B	UM	UJ
0.001	0.00026916	0.00020020	0.00008342
0.01	0.00271869	0.00202020	0.00084175
0.1	0.03008176	0.02222000	0.00925926
0.5	0.24050520	0.18750000	0.08329264
0.8	0.55535336	0.47232000	0.30470022

p	UF	ED
0.001	0.00033217	0.00049459
0.01	0.00335185	0.00496841
0.1	0.03683350	0.05205589
0.5	0.29035417	0.33037177
0.8	0.64773867	0.66234679

1.3 titulaire

p	R	P	G
0.001	0.00009404	0.00023347	0.00030509
0.01	0.00094420	0.00235887	0.00307012
0.1	0.00986260	0.02619348	0.03276540
0.5	0.06691947	0.21753482	0.23364486
0.8	0.16813448	0.53152963	0.54945048

p	B	UM	UJ
0.001	0.00023028	0.00018004	0.00003575
0.01	0.00232784	0.00181673	0.00036075
0.1	0.02598253	0.01998381	0.00396825
0.5	0.21963492	0.17423561	0.03571429
0.8	0.53898955	0.48368345	0.14251174

p	UF	ED
0.001	0.00003549	0.00006497
0.01	0.00035813	0.00065463
0.1	0.00393939	0.00708293
0.5	0.03543723	0.05571353
0.8	0.12963609	0.15635391

2. pr (r : 1 - B (C_B))

2.1 adjoint

p	R	P	G
0.001	0.00048024	0.00046989	0.00047331
0.01	0.00473496	0.00468897	0.00468119
0.1	0.04044163	0.04591260	0.04489123
0.5	0.13090527	0.20942915	0.19028340
0.8	0.16195356	0.31339767	0.27325581

p	B	UM	UJ
0.001	0.00046990	0.	0.00498503
0.01	0.00469007	0.	0.04852958
0.1	0.04601831	0.02061856	0.37619145
0.5	0.21164382	0.22680412	0.81827060
0.8	0.31835831	0.38144330	0.88636364

p	UF	ED
0.001	0.00043457	0.00046928
0.01	0.00430672	0.00462787
0.1	0.03936315	0.04055958
0.5	0.13321875	0.12849989
0.8	0.16314240	0.16052577

2.2 CCD

p	R	P	G
0.001	0.00198352	0.00199800	0.00959617
0.01	0.01965889	0.01980133	0.02630556
0.1	0.18064045	0.18126925	0.16848065
0.5	0.66758984	0.63212056	0.50000094
0.8	0.89047819	0.79810348	0.61538462

p	B	UM	UJ
0.001	0.00199818	0.00199800	0.00548171
0.01	0.01981917	0.01980100	0.05320726
0.1	0.18277476	0.18098000	0.40202461
0.5	0.64950610	0.61250000	0.83337402
0.8	0.82254557	0.75008000	0.89583333

p	UF	ED
0.001	0.00198967	0.00199801
0.01	0.01977760	0.01980241
0.1	0.18615850	0.18180373
0.5	0.70514583	0.66512823
0.8	0.91320533	0.88555677

2.3 titulaire

p	R	P	G
0.001	0.00098568	0.00227740	0.00231150
0.01	0.01976664	0.02254204	0.02231965
0.1	0.14691203	0.20387574	0.18566936
0.5	0.32308053	0.68018098	0.53271028
0.8	0.36911774	0.83862106	0.64589235

p	B	UM	UJ
0.001	0.00227750	0.	0.01338373
0.01	0.02255146	0.	0.12399745
0.1	0.20464611	0.15726799	0.66154813
0.5	0.68804443	0.64590827	0.92857143
0.8	0.84870657	0.77519424	0.95535714

p	UF	ED
0.001	0.00233222	0.00226963
0.01	0.02263727	0.02180232
0.1	0.17104320	0.15514869
0.5	0.35456277	0.33428647
0.8	0.38113636	0.37441452

3.pr (r : 8 E B (C_B))

3.1 adjoint

p	R	P	G
0.001	0.00000500	0.	0.00000011
0.01	0.00005000	0.	0.00000109
0.1	0.00050000	0.	0.00001092
0.5	0.00250000	0.00000001	0.00005460
0.9	0.00450000	0.00000004	0.00009829

p	B	UF
0.001	0.	0.
0.01	0.	0.
0.1	0.	0.
0.5	0.	0.
0.9	0.	0.

3.2 CCD

p	R	P	G
0.001	0.00001250	0.00000110	0.00003902
0.01	0.00012500	0.00001097	0.00039018
0.1	0.00125000	0.00010967	0.00390184
0.5	0.00625000	0.00054836	0.01950922
0.9	0.01125000	0.00098705	0.03511660

p	B	UF
0.001	0.00000012	0.
0.01	0.00000116	0.
0.1	0.00001163	0.
0.5	0.00005814	0.
0.9	0.00010466	0.

3.3 titulaire

p	R	P	G
0.001	0.00009980	0.00000246	0.00005451
0.01	0.00099800	0.00002456	0.00054511
0.1	0.00998000	0.00024561	0.00545115
0.5	0.04990000	0.00122806	0.02725574
0.9	0.08982000	0.00221051	0.04906034

p	B	UF
0.001	0.00000132	0.00014182
0.01	0.00001323	0.00141818
0.1	0.00013233	0.01418182
0.5	0.00066164	0.07090909
0.9	0.00119096	0.12763636

4. pr (r : 5 E B (C_B))

4.1 adjoint

p	R	P	G
0.001	0.00002100	0.00000013	0.00000334
0.01	0.00021000	0.00000129	0.00003341
0.1	0.00210000	0.00001295	0.00033412
0.5	0.01050000	0.00006474	0.00167060
0.9	0.01890000	0.00011653	0.00300708

p	B	UF
0.001	0.00000005	0.
0.01	0.00000047	0.
0.1	0.00000474	0.
0.5	0.00002368	0.
0.9	0.00004262	0.

4.2 CCD

p	R	P	G
0.001	0.00006250	0.00005265	0.00013169
0.01	0.00062500	0.00052653	0.00131687
0.1	0.00625000	0.00526530	0.01316872
0.5	0.03125000	0.02632651	0.06584362
0.9	0.05625000	0.04738772	0.11851852

p	B	UF
0.001	0.00003474	0.
0.01	0.00034745	0.
0.1	0.00347448	0.
0.5	0.01737240	0.
0.9	0.03127033	0.

4.3 titulaire

p	R	P	G
0.001	0.00016580	0.00008143	0.00016229
0.01	0.00165800	0.00081430	0.00162295
0.1	0.01658000	0.00814300	0.01622950
0.5	0.08290000	0.04071499	0.08114748
0.9	0.14922000	0.07328699	0.14606546

p	B	UF
0.001	0.00007258	0.00024818
0.01	0.00072580	0.00248182
0.1	0.00725799	0.02481818
0.5	0.03628996	0.12409091
0.9	0.06532192	0.22336364

5. pr (r : 2 E B (C_B))

5.1 adjoint

p	R	P	G
0.001	0.00011900	0.00008125	0.00010223
0.01	0.00119000	0.00081247	0.00102226
0.1	0.01190000	0.00812467	0.01022259
0.5	0.05950000	0.04062333	0.05111296
0.9	0.10710000	0.07312200	0.09200333

p	B	UF
0.001	0.00007734	0.00013050
0.01	0.00077341	0.00130500
0.1	0.00773415	0.01305000
0.5	0.03867073	0.06525000
0.9	0.06960731	0.11745000

5.2 CCD

p	R	P	G
0.001	0.00054550	0.00059399	0.00044444
0.01	0.00545500	0.00593994	0.00444444
0.1	0.05455000	0.05939942	0.04444444
0.5	0.27275000	0.29699708	0.22222222
0.9	0.49095000	0.58805421	0.40000000

p	B	UF
0.001	0.00062115	0.00066367
0.01	0.00621150	0.00663667
0.1	0.06211500	0.06636667
0.5	0.31057502	0.33183333
0.9	0.55903503	0.59730000

5.3 titulaire

p	R	P	G
0.001	0.00029600	0.00066451	0.00048319
0.01	0.00296000	0.00664508	0.00483195
0.1	0.02960000	0.06645078	0.04831945
0.5	0.14800000	0.33225390	0.24159726
0.9	0.26640000	0.59805702	0.43487507

p	B	UF
0.001	0.00067763	0.00035455
0.01	0.00677634	0.00354545
0.1	0.06776344	0.03545455
0.5	0.33881722	0.17727273
0.9	0.60987100	0.31909091

6. pr (r : i E - j E B (C_B))

6.1 adjoint , p = 0.1

i	j	R	UF	ED
1	4	0.00597550	0.00486063	0.00674155
2	6	0.00473349	0.00485625	0.00434099
3	8	0.00333404	0.00481250	0.00279623
4	DE	0.00234030	0.00437500	0.00180122
7	DE	0.00030330	0.	0.00048144
1	2	0.00670000	0.00568750	0.00742427
1	1	0.01750000	0.01750000	0.01750000

6.2 adjoint , p = 0.5

i	j	R	UF	ED
1	4	0.04518750	0.04101563	0.04777070
2	6	0.03340625	0.03828125	0.03017755
3	8	0.02243750	0.03281250	0.01932160
4	DE	0.01375781	0.02187500	0.01242469
7	DE	0.00206250	0.	0.00328644
1	2	0.05750000	0.05468750	0.05951186
1	1	0.08750000	0.08750000	0.08750000

6.3 adjoint ; p = 0.9

i	j	R	UF	ED
1	4	0.13726350	0.13541063	0.13835220
2	6	0.09466461	0.10670625	0.08746590
3	8	0.06023812	0.07481250	0.05553134
4	DE	0.03541751	0.03937500	0.03529496
7	DE	0.00685530	0.	0.00855662
1	2	0.14670000	0.14568750	0.14742427
1	1	0.15750000	0.15750000	0.15750000

6.4 CCD , p = 0.1

i	j	R	UF	ED
1	4	0.04867705	0.03683350	0.05206191
2	6	0.03671070	0.03650167	0.02622015
3	8	0.00710553	0.03318333	0.01320674
4	DE	0.00605527	0.	0.00665209
7	DE	0.00027352	0.	0.00085005
1	2	0.05045500	0.03982000	0.05444550
1	1	0.09955000	0.09955000	0.09955000

6.5 CCD , p = 0.5

i	j	R	UF	ED
1	4	0.33065625	0.29035417	0.33301672
2	6	0.20617188	0.24887500	0.16672801
3	8	0.05169531	0.16591667	0.08385118
4	DE	0.03828125	0.	0.04221671
7	DE	0.00325000	0.	0.00538939
1	2	0.36137500	0.33183333	0.37245973
1	1	0.49775000	0.49775000	0.49775000

6.6 CCD , p = 0.9

i	j	R	UF	ED
1	4	0.82304505	0.80934150	0.82117964
2	6	0.45734666	0.56743500	0.41136373
3	8	0.14647349	0.29865000	0.20659239
4	DE	0.09722161	0.	0.10381203
7	DE	0.01120248	0.	0.01303517
1	2	0.84685500	0.83622000	0.85084549
1	1	0.89595000	0.89595000	0.89595000

6.7 titulaire , p = 0.1

i	j	R	UF	ED
1	4	0.00987900	0.00396382	0.00710137
2	6	0.00462719	0.00394113	0.00591424
3	8	0.00465983	0.00393950	0.00493635
4	DE	0.00499746	0.00393939	0.00412088
7	DE	0.00245649	0.00393935	0.00239755
1	2	0.01237200	0.00709091	0.00974247
1	1	0.03900000	0.03900000	0.03900000

6.8 titulaire , p = 0.5

i	j	R	UF	ED
1	4	0.07597500	0.04875000	0.06576476
2	6	0.04316875	0.03988636	0.04998477
3	8	0.03928438	0.03656250	0.04002441
4	DE	0.03455575	0.03531605	0.03241441
7	DE	0.01744600	0.03434659	0.01885886
1	2	0.12130000	0.10636364	0.00974247
1	1	0.19500000	0.19500000	0.19500000

6.9 titulaire , p = 0.9

i	j	R	UF	ED
1	4	0.28902780	0.27256745	0.28335058
2	6	0.20584791	0.23534869	0.22596308
3	8	0.17302031	0.20603891	0.18196506
4	DE	0.12958240	0.18173274	0.13655522
7	DE	0.07361783	0.13067092	0.07933543
1	2	0.32533200	0.31909091	0.32174247
1	1	0.35100000	0.35100000	0.35100000

7. r : i E - j E B S

7.1 adjoint

		m'		m"	
i	j	R	ED	R	ED
2	5	0.263	0.257	0.438	0.432
1	4	0.416	0.403	0.416	0.403
2	DE	0.306	0.295	0.481	0.470
1	DE	0.481	0.470	0.481	0.470
1	1	0.175	0.175	0.175	0.175
7	DE	0.024	0.021	0.481	0.470
2	3	0.196	0.183	0.371	0.358
3	7	0.171	0.173	0.466	0.460

7.2 CCD

		m'		m"	
i	j	R	ED	R	ED
2	5	0.914	0.944	1.910	1.939
1	4	1.847	1.876	1.847	1.876
2	DE	0.990	1.004	1.986	2.000
1	DE	1.986	2.000	1.986	2.000
1	1	0.996	0.995	0.996	0.995
7	DE	0.037	0.029	1.986	2.000
2	3	0.731	0.753	1.821	1.749
3	7	0.423	0.490	1.964	1.987

7.3 titulaire

		m'		m"	
i	j	R	ED	R	ED
2	5	0.934	1.010	1.325	1.400
1	4	1.158	1.212	1.158	1.212
2	DE	1.805	1.890	2.196	2.280
1	DE	2.196	2.280	2.196	2.280
1	1	0.391	0.390	0.391	0.390
7	DE	0.732	0.724	2.196	2.280
17	DE	0.088	0.081	2.196	2.280
2	3	0.553	0.596	0.944	0.986
3	7	0.900	0.971	1.588	1.686

ANNEXE 2 - FORMULES IMPLEMENTEES

1 - Probabilité d'un critère de relation

1.1 - pr (r : i - j B (C_B))

1.1.1 - cas général, p < 1.

$$pr = \sum_{k=i}^{J} F(k) \sum_{s=i}^{a} \Pi(k,s)$$

$$a = \inf(k,j)$$

1.1.2 - cas TS, p < 1.

$$pr = \sum_{k=1}^{J} F(k)\, \Pi(k,k)$$

1.1.3 - cas i- , p < 1.

$$pr = 1 - \sum_{k=o}^{J} F(k) \sum_{s=o}^{b} \Pi(k,s)$$

$$b = \inf(k, i-1)$$

1.1.4 - cas j+ , p < 1

$$pr = \sum_{k=o}^{J} F(k) \sum_{s=o}^{a} \Pi(k,s)$$

$$a = \inf(k,j)$$

1.1.5 - cas général, p = 1.

$$pr = \sum_{k=i}^{J} F(k)$$

1.1.6 - cas i-, p = 1.

$$pr = 1 - \sum_{k=o}^{i-1} F(k)$$

1.1.7 - cas j+, p = 1.

$$pr = \sum_{k=o}^{j} F(k)$$

1.2 - pr (r : i E - j E B (C_B))

1.2.1 - cas général.

$$pr = \sum_{k=i}^{J} F(k) \prod (b,b)$$

$$b = \inf(k,j) - i + 1$$

1.2.2 - cas i E , i $\geq$ 1.

$$pr = p\left(1 - \sum_{k=o}^{i-1} F(k)\right)$$

1.2.3 - cas 1 E .

$$pr = p\,(1 - F(o))$$

1.2.4 - cas D E .

$$pr = p\,(1 - F(o))$$

1.2.5 - cas i E - D E .

$$pr = \sum_{k=i}^{J} F(k)\ \Pi\,(k - i + 1\ ,\ k - i + 1)$$

2 - Boucle d'accès sans ordre next et exit.

2.1 - ⌈ r : TS B (C_B) S ⌉ .

$$m' = m\,p$$
$$m'' = m$$

2.2 - ⌈ r : i E - j E B (C_B) S ⌉ .

2.2.1 - cas général, p < 1.

$$m' = \sum_{k=i}^{J} F(k) \sum_{s=i}^{k} (s^* - i + 1)\ \Pi\,(k,s)$$

$$s^* = \inf\,(s,j)$$

S'il existe un compteur de cibles :

$$m'' = \sum_{k=i}^{j} k \; F(k)$$

$$+ \sum_{k=j+1}^{J} F(k) \left(k \sum_{s=o}^{j-1} \pi(k,s) + j\,(k+1) \sum_{s=j}^{k} \frac{\pi(k,s)}{s+1} \right)$$

Sinon :

$$m'' = \sum_{k=1}^{j} k \; F(k)$$

$$+ \sum_{k=j+1}^{J} F(k) \left(k \sum_{s=o}^{j-1} \pi(k,s) + j\,(k+1) \sum_{s=j}^{k} \frac{\pi(k,s)}{s+1} \right)$$

2.2.2 - cas p = 1.

$$m' = \sum_{k=i}^{J} F(k) \; (k^* - i + 1)$$

$$k^* = \inf (k,j)$$

S'il existe un compteur de cibles :

$$m'' = \sum_{k=i}^{j} k \; (F)(k) + j \sum_{k=j+1}^{J} F(k)$$

Sinon :

$$m'' = \sum_{k=1}^{j} k \; F(k) + j \sum_{k=j+1}^{J} F(k)$$

2.2.3 - cas 1 E - D E .

$m' = p\ m$

$m'' = m$

2.2.4 - cas i E - D E , sans compteur de cibles.

m' est donné par la formule générale où $j = J$,

$m'' = m$

2.2.5 - cas 1 E , $p \leq 1$.

m' est donné par la formule générale,

$$m'' = m + \sum_{k=2}^{J} F(k) \sum_{s=1}^{k} \Pi(k,s) \left(1 - \frac{s\,(k+1)}{s+1} \right)$$

3. - Boucle d'accès avec ordre next ou exit.

3.1 - [r : TS B (C_B) S]

3.1.1 cas général, $p \leq 1$.

$$m' = \frac{1}{1 - p_N} \left(1 - \sum_{k=o}^{J} F(k) \sum_{s=o}^{k} \Pi(k,s)\ p_N^s \right.$$

$$m'' = \sum_{k=1}^{J} F(k) \sum_{s=o}^{k} \Pi(k,s) \left(\frac{k+1}{s+1} \quad \frac{1-p_N^s}{1-p_N} + p_N^s \quad \frac{k-s}{s+1} \right) .$$

3.1.2 - cas $p = 1$.

$$m' = \frac{1}{1 - p_N} \left(1 - \sum_{k=o}^{J} F(k)\ p_N^k \right).$$

$$m'' = m'$$

3.2 - $\lceil r : i\,E - j\,E \;\; B(C_B)\; S \rceil$.

3.2.1 - cas général , $p < 1$.

$$m' = \frac{1}{1 - p_N} \sum_{k=i}^{J} F(k) \sum_{s=i}^{k} \Pi(k,s) \left(1 - p_N^{s^*-i+1} \right) .$$

$$s^* = \inf (s,j) .$$

Si on pose :

$$m''_o = \sum_{k=i}^{J} F(k) \left(\sum_{s=o}^{i-1} k \, \Pi(k,s) \right.$$

$$+ \sum_{s=i}^{k^*} \frac{\Pi(k,s)}{s+1} \left((k+1)\left(i-1 + \frac{1}{1-p_N} \right) - \left(s - \frac{1 + kp_N}{1 - p_N} \right) p_N^{s-i+1} \right)$$

$$\left. + (k+1) \left(i-1 + \frac{1 - p_N^{j-i+1}}{1 - p_N} \right) \sum_{s=j+1}^{k} \frac{\Pi(k,s)}{s+1} \right)$$

s'il existe un compteur de cibles :

$$m'' = m''_o$$

sinon :

$$m'' = \sum_{k=1}^{i-1} k\,F(k) + m''_o$$

3.2.2 - cas p = 1.

$$m' = \frac{1}{1 - p_N} \sum_{k=i}^{J} F(k) \left(1 - p_N^{k^*} - i + 1 \right)$$

$$k^* = \inf (k,j)$$

si on pose:

$$m''_o = \sum_{k=i}^{J} F(k) \left(i - 1 + \frac{1 - p_N^b}{1 - p_N} \right)$$

où $b = \inf (k,j) - i + 1$,

s'il existe un compteur de cibles:

$$m'' = m''_o$$

sinon :

$$m'' = \sum_{k-1}^{i-1} k \; F(k) + m''_o$$

3.2.3 - cas 1 E - D E .

m' et m" sont donnés par les formules du cas TS (voir A.II / 3.1)

4 - Coût d'évaluation d'un critère de relation.

4.1 - pr (r : i : j $B(C_B)$).

4.1.1 - cas général, p $<$ 1 , sans compteur de cibles.

$$m_1 = m_2 = \sum_{k=1}^{j} k \; F(k)$$

$$+ \sum_{k=j+1}^{J} F(k) \left(k \sum_{s=o}^{j} \pi(k,s) + (j+1)(k+1) \sum_{s=j+1}^{k} \frac{\pi(k,s)}{s+1} \right)$$

4.1.2 cas général, $p < 1$, avec compteur de cibles.

$$m_1 = m_2 = \sum_{k=i}^{j} F(k) \left(k \sum_{s=o}^{i-1} \pi(k,s) \right.$$

$$\left. + i\,(k+1) \sum_{s=i}^{k} \frac{\pi(k,s)}{s+1} \right)$$

$$+ \sum_{k=j+1}^{J} F(k) \left(k \sum_{s=o}^{j} \pi(k,s) \right.$$

$$\left. + (j+1)(k+1) \sum_{s=j+1}^{k} \frac{\pi(k,s)}{s+1} \right)$$

4.1.3 cas $p = 1$, sans compteur de cibles.

$$m_1 = m_2 = j + 1 - \sum_{k=o}^{j} (j + 1 - k) \; F(k)$$

4.1.4 cas p = 1 , avec compteur de cibles.

$m_1 = m_2 = o$

4.1.5 cas i = j , p < 1 , avec compteur de cibles.

Le premier terme de A II / 4.1.2 est remplacé par i F (i).

4.1.6 cas j+ , p < 1 ,avec compteur de cibles.

Le premier terme de A II / 4.1.2 est nul.

4.1.7 cas i- , p < 1 , avec compteur de cibles.

$$m_1 = m_2 = \sum_{k=i}^{J} F(k) \left(k \sum_{s=o}^{i-1} \Pi(k,s) + i\,(k+1) \sum_{s=i}^{k} \frac{\Pi(k,s)}{s+1} \right)$$

4.1.8 cas i- , p < 1 , sans compteur de cibles.

Il faut ajouter à la formule précédente (4.1.7), le terme

$$\sum_{k=1}^{i-1} k\,F(k)$$

4.1.9 cas i- , p = 1, sans compteur de cibles.

$$m_1 = m_2 = i - \sum_{k=o}^{i-1} (i - k) \; F(k)$$

4.1.10 cas TS , p < 1.

$$m_1 = m_2 = \sum_{k=1}^{J} F(k) \left(\sum_{s=o}^{k} \frac{\Pi (k,s)}{k - s + 1} (k+1) - \Pi(k,k) \right)$$

4.2 (r : i E - j E B (C_B)).

4.2.1 cas général, sans compteur de cibles.

$$m_1 = \sum_{k=1}^{i-1} k \; F(k) + \sum_{k=i}^{J} F(k) \; (\; i - 1 + \sum_{s=o}^{a-1} \Pi (a,s) \frac{a + 1}{a - s + 1} + a \Pi (a,a) \;).$$

$$a = \inf (k,j) - i + 1$$

$$m_2 = \sum_{k=i}^{J} F(k) \; (\sum_{s=o}^{a-1} \Pi (a,s) \frac{a + 1}{a - s + 1} + a \Pi (a,a) \;).$$

4.2.2 cas général, avec compteur de cibles.

m_2 est donné par sa valeur en 4.2.1 et

m_1 par le second terme de m_1 en 4.2.1 .

4.2.3 cas i = j , sans compteur de cibles.

$$m_1 = \sum_{k=1}^{i-1} k \; F(k) + i \sum_{k=i}^{J} F(k)$$

$$m2 = \sum_{k=i}^{J} F(k)$$

4.2.4 cas i = j , avec compteur de cibles.

$$m_1 = i \; m_2$$

$$m_2 = \sum_{k=i}^{J} F(k)$$

4.2.5 cas D E

$$m_1 = m$$

$$m_2 = 1 - F(o)$$

5 - Nombre d'éléments dans une collection.

Si on note $p\ (= B^*)$ la probabilité d'appartenance à B^* et $_B$ la nombre d'éléments de la collection, on a :

$$p\ (= B^*) = 1 - \sum_{k=o}^{J_s} F_s(k)\ \pi\ (k,o),$$

où F_s est la fonction statistique associée à la relation inverse de $R\ (A,B)$.

5.1 - Si on accède par la boucle [r : TS B (C_B) S]

On a :

$$\frac{\mathcal{V}'_B}{NB} = p_{C_B}\ p\ (= B^*)$$

où p_{C_B} est la probabilité de C_B.

5.2 - Si on accède par la boucle [r : iE - jE B (C_B) S]

On a

$$\frac{\mathcal{V}'_B}{NB} = \frac{m'}{m}\ p\ (= B^*)$$

A N N E X E III

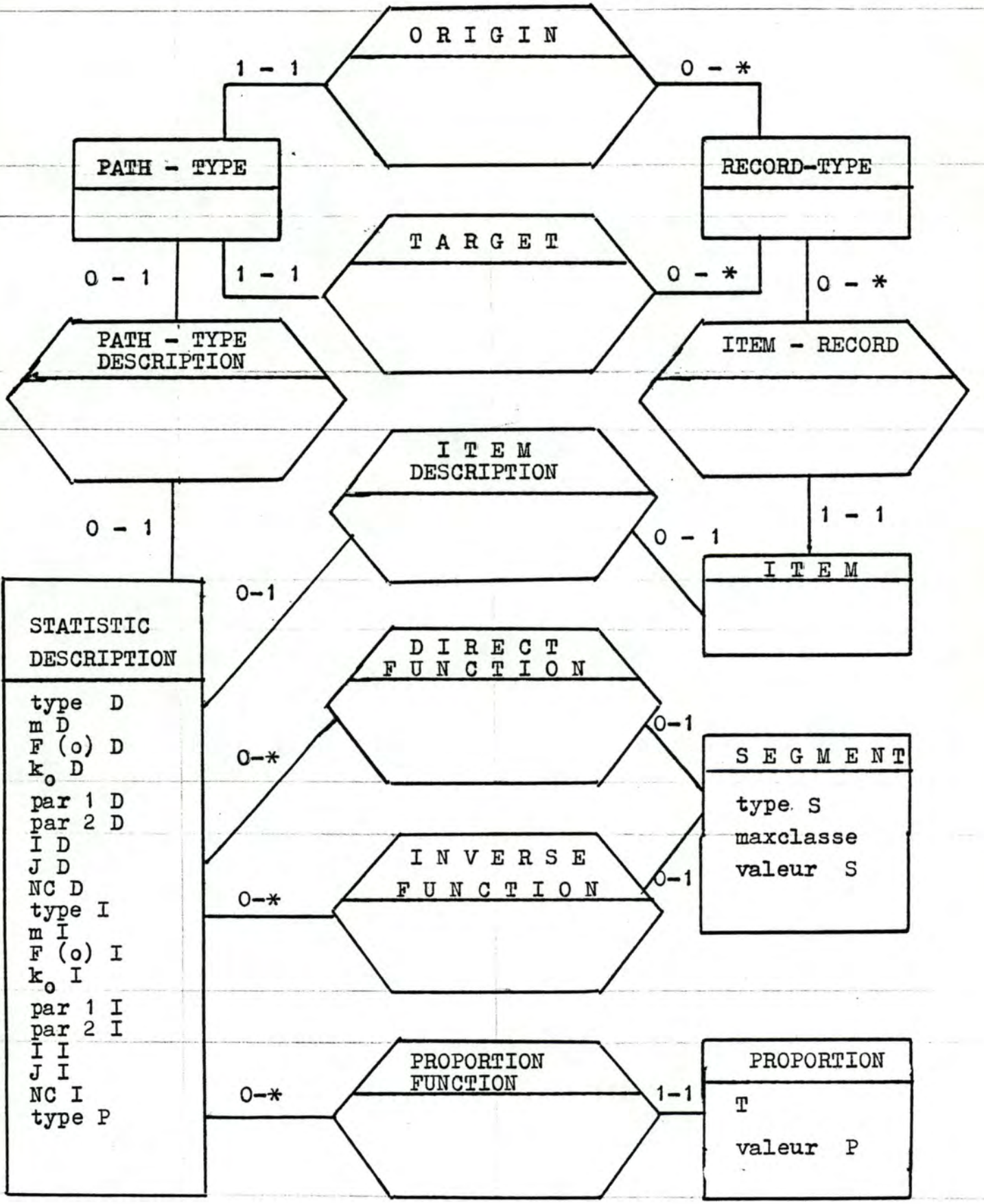
SCHEMA CONCEPTUEL

Réprésentation graphique

Le schéma développé pour le projet I S D O S n'a été repris que succinctement : les types d'entités, les types d'association et les propriétés dont il n'a pas été question dans ce travail ne sont pas repris.

Seuls figurent les types d'entité : path-type, record-type, item, et les associations origin, target et item-record.

voir page suivante :

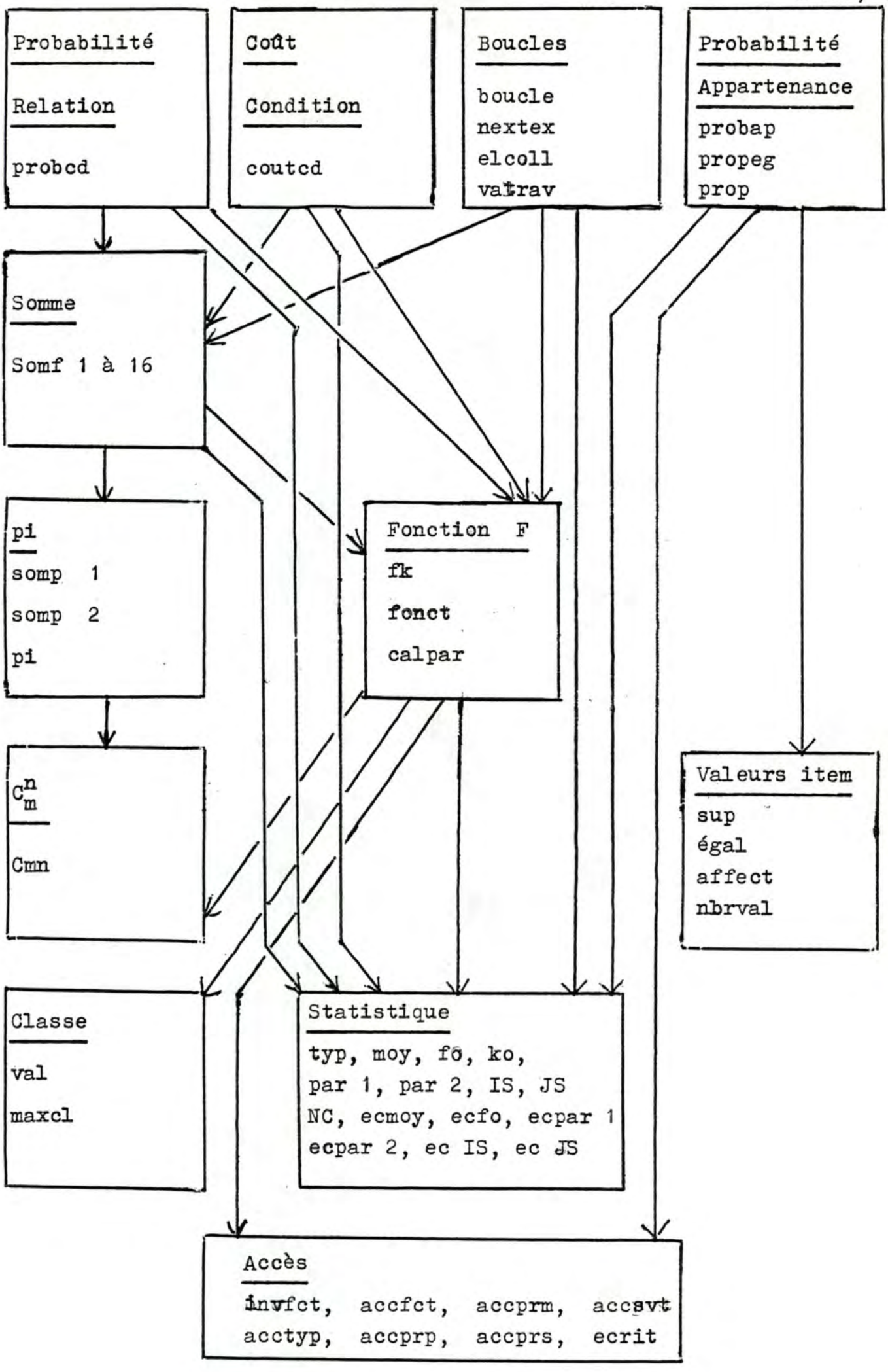

ORIGIN
1 - 1
0 - *
PATH - TYPE
RECORD-TYPE
TARGET
0 - 1
1 - 1
0 - *
0 - *
PATH - TYPE DESCRIPTION
ITEM - RECORD
ITEM DESCRIPTION
0 - 1
1 - 1
0 - 1
ITEM
0-1
STATISTIC DESCRIPTION
type D
m D
F (o) D
k o D
par 1 D
par 2 D
I D
J D
NC D
type I
m I
F (o) I
k o I
par 1 I
par 2 I
I I
J I
NC I
type P
DIRECT FUNCTION
0-1
0-*
SEGMENT
type S
maxclasse
valeur S
INVERSE FUNCTION
0-*
0-1
PROPORTION FUNCTION
0-*
1-1
PROPORTION
T
valeur P

A N N E X E IV

ARCHITECTURE

Représentation graphique

La hiérarchie est une hiérarchie "utilise" à 6 niveaux , les flèches représente cette relation ;


Probabilité
Relation
probcd
Coût
Condition
coutcd
Boucles
boucle
nextex
elcoll
vatrav
Probabilité
Appartenance
probap
propeg
prop
Somme
Somf 1 à 16
pi
somp 1
somp 2
pi
Fonction F
fk
fonct
calpar
C_m^n
Cmn
Valeurs item
sup
égal
affect
nbrval
Classe
val
maxcl
Statistique
typ, moy, fo, ko,
par 1, par 2, IS, JS
NC, ecmoy, ecfo, ecpar 1
ecpar 2, ec IS, ec JS
Accès
invfct, accfct, accprm, accsvt
acctyp, accprp, accprs, ecrit

A N N E X E V

P R O G R A M M E S

```
      SUBROUTINE PROBCD(IDENT,P,CAR,I,J,RES)

      DIMENSION STAT(9) , IDENT(3)
      REAL P , RES , STAT , SOMF1 , SOMF2 , SOMF3 , F0
      INTEGER IDENT , I , J , X , Y , JS
      LOGICAL CAR

      CALL FONCT(IDENT,STAT)
      IF (J.GT.JS(STAT)) J = JS(STAT)
      IF (.NOT.CAR) GO TO 100
      IF (I.NE.-1) GO TO 40
      IF (J.NE.-1) GO TO 30
      IF (P - 1) 20 , 10 , 200
 10   RES = 1 - F0(STAT)
      GO TO 200
 20   X = 1
      Y = JS(STAT)
      RES = SOMF3(STAT,X,Y,P)
      GO TO 200
 30   X = 0
      Y = J
      GO TO 60
 40   IF (J.NE.-1) GO TO 50
      X = 0
      Y = I - 1
      GO TO 60
 50   X = I
      Y = J
 60   IF (P - 1) 80 , 70 , 200
 70   RES = SOMF2(STAT,X,Y)
      GO TO 90
 80   RES = SOMF1(STAT,X,Y,P)
 90   IF (J.EQ.-1) RES = 1 - RES
      GO TO 200
100   IF (I.EQ.J) GO TO 110
      X = I
      Y = J
      IF (J.EQ.-1) Y = JS(STAT)
      RES = SOMF3(STAT,X,Y,P)
      GO TO 200
110   IF ((J.EQ.1).OR.(J.EQ.-1)) GO TO 120
      Y = I - 1
      X = 0
      RES = P * (1 - SOMF2(STAT,X,Y))
      GO TO 200
120   RES = P * (1 - F0(STAT))
      GO TO 200
200   RETURN
      END
```

```
      SUBROUTINE BOUCLE (IDENT,P,CAR,I,J,COMPT,RES)

      DIMENSION RES(2) , STAT(9) , IDENT(3)
      REAL P , RES , STAT , AUX , MOY
      REAL SOMF2 , SOMF4 , SOMF5 , SOMF6 , SOMF7 , SOMF8
      INTEGER IDENT , I , J , X , Y , JS
      LOGICAL CAR , COMPT

      CALL FORCE(IDENT,STAT)
      IF (J.GT.JS(STAT)) J = JS(STAT)
      IF (CAR.OR.(.NOT.CAR).AND.(I.EQ.1).AND.(J.EQ.-1)) GO TO 80
      X = I
      Y = J
      IF (I.EQ.-1) X = JS(STAT)
      IF (J.EQ.-1) Y = JS(STAT)
      IF (P - 1) 20 , 10 , 200
   10 RES(1) = SOMF5(STAT,X,Y)
      GO TO 30
   20 RES(1) = SOMF4(STAT,X,Y,P)
   30 IF ((J.EQ.-1).AND.(.NOT.COMPT)) GO TO 90
      IF ((I.EQ.1).AND.(J.EQ.1).AND.(P.NE.1)) GO TO 70
      IF (.NOT.COMPT) X = 1
      RES(2) = SOMF6(STAT,X,Y)
      IF (P - 1) 40 , 50 , 200
   40 X = Y
      Y = JS(STAT)
      AUX = SOMF7(STAT,X,Y,P)
      GO TO 60
   50 X = Y + 1
      Y = JS(STAT)
      AUX = (X - 1) * SOMF2(STAT,X,Y)
   60 RES(2) = RES(2) + AUX
      GO TO 200
   70 RES(2) = MOY(STAT) + SOMF8(STAT,P)
      GO TO 200
   80 RES(1) = P * MOY(STAT)
   90 RES(2) = MOY(STAT)
  200 RETURN
      END
```

```
      SUBROUTINE NEXTEX(IDENT,P,CAR,I,J,COMPT,PN,RES)

      DIMENSION STAT(9) , IDENT(3) , RES(2)
      REAL P , PN , RES , STAT
      REAL SOMF6 , SOMF9 , SOMF10 , SOMF11 , SOMF12 , SOMF13 , SOMF16
      INTEGER IDENT , I , J , X , Y , JS
      LOGICAL CAR , COMPT

      CALL FONCT(IDENT,STAT)
      IF (J.GT.JS(STAT)) J = JS(STAT)
      IF (CAR.OR.(.NOT.CAR).AND.(I.EQ.1).AND.(J.EQ.-1)) GO TO 40
      X = I
      Y = J
      IF (I.EQ.-1) X = JS(STAT)
      IF (J.EQ.-1) Y = JS(STAT)
      IF (P - 1) 10 , 20 , 200
 10   RES(1) = SOMF10(STAT,X,Y,P,PN)
      RES(2) = SOMF13(STAT,X,Y,P,PN)
      GO TO 30
 20   RES(1) = SOMF11(STAT,X,Y,PN)
      RES(2) = SOMF16(STAT,X,Y,PN)
 30   IF (COMPT) GO TO 200
      Y = X - 1
      X = 1
      RES(2) = RES(2) + SOMF6(STAT,X,Y)
      GO TO 200
 40   IF (P - 1) 50 , 60 , 200
 50   Y = JS(STAT)
      RES(1) = 1 / (1 - PN) - SOMF9(STAT,Y,P,PN)
      X = 1
      RES(2) = SOMF13(STAT,X,Y,P,PN)
      GO TO 200
 60   RES(1) = SOMF12(STAT,PN)
      RES(2) = RES(1)
      GO TO 200
200   RETURN
      END
```

```
      SUBROUTINE BLCOLL(IDENT,P,CAR,I,J,COMPT,PN,PO,RES)

      DIMENSION STAT(9) , IDENT(3) , RESAUX(2)
      REAL P , PN , PO , RES , MU , RESAUX , MOY , SOMF1
      INTEGER IDENT , I , J
      LOGICAL CAR , COMPT

      IF (CAR) GO TO 30
      CALL FONCT(IDENT,STAT)
      MU = MOY(STAT)
      IF (PN - 1) 10 , 20 , 100
 10   CALL NEXTEX(IDENT,P,CAR,I,J,COMPT,PN,RESAUX)
      GO TO 30
 20   CALL BOUCLE(IDENT,P,CAR,I,J,COMPT,RESAUX)
 30   CALL INVFCT(IDENT)
      CALL FONCT(IDENT,STAT)
      RES = 1 - SOMF1(STAT,0,0,PO)
      IF (CAR) RES = P * RES
      IF (.NOT.CAR) RES = RESAUX(1) * RES / MU
100   RETURN
      END
```

```
      SUBROUTINE VATRAV(IDENT,P,CAR,I,J,COMPT,PN,PO,PK,RES)

      DIMENSION IDENT(3)
      REAL P , PN , PO , PK , RES
      INTEGER IDENT , I , J
      LOGICAL CAR , COMPT

      CALL ENCOLL(IDENT,P,CAR,I,J,COMPT,PN,PO,RES)
      RES = PK + RES * (1 - PK)
      RETURN
      END
```

```
      SUBROUTINE COUTCD (IDENT,P,CAR,I,J,COMPT,RES)

      DIMENSION STAT(9) , RES(2) , IDENT(3)
      REAL P , RES , AUX , MOY , F0
      REAL SOMF2 , SOMF6 , SOMF7 , SOMF14 , SOMF15
      INTEGER IDENT , I , J , X , Y , Z , LIM , JS
      LOGICAL CAR , COMPT

      CALL FONCT(IDENT,STAT)
      IF (J.GT.JS(STAT)) J = JS(STAT)
      IF (.NOT.CAR) GO TO 90
      IF (P - 1) 10 , 60 , 200
   10 IF ((I.EQ.-1).AND.(J.EQ.-1)) GO TO 50
      X = J
      IF (J.EQ.-1) X = I - 1
      Y = JS(STAT)
      RES(1) = SOMF7(STAT,X,Y,P)
      IF (COMPT) GO TO 20
      Y = X
      X = 1
      AUX = SOMF6(STAT,X,Y,P)
      GO TO 40
   20 IF (I.NE.J) GO TO 30
      AUX = SOMF6(STAT,I,I)
      GO TO 40
   30 IF ((I.EQ.-1).OR.(J.EQ.-1)) GO TO 80
      X = I - 1
      Y = J
      AUX = SOMF7(STAT,X,Y,P)
      GO TO 40
   40 RES(1) = RES(1) + AUX
      GO TO 80
   50 X = 1
      Y = JS(STAT)
      Z = 0
      RES(1) = SOMF15(STAT,X,Y,Z,P)
      GO TO 80
   60 IF (COMPT) GO TO 70
      Y = J
      IF (J.EQ.-1) Y = I - 1
      RES(1) = SOMF14(STAT,Y)
      GO TO 80
   70 RES(1) = 0
   80 RES(2) = RES(1)
      GO TO 200
   90 IF ((I.NE.-1).OR.(J.NE.-1)) GO TO 100
      RES(1) = MOY(STAT)
      RES(2) = 1 - F0(STAT)
      GO TO 200
  100 IF (I.NE.J) GO TO 110
      X = I
      Y = JS(STAT)
      RES(2) = SOMF2(STAT,X,Y,P)
      RES(1) = I * RES(2)
      GO TO 120
  110 Y = J
      IF (J.EQ.-1) Y = JS(STAT)
      X = I
      Z = I - 1
      RES(1) = SOMF15(STAT,X,Y,Z,P)
```

```
      Z = 0
      RES(2) = SOMF15(STAT,X,Y,Z,P)
120   IF (COMPT) GO TO 200
      X = 1
      Y = I - 1
      RES(1) = RES(1) + SOMF6(STAT,X,Y)
      GO TO 200
200   RETURN
      END
```

```
      SUBROUTINE PROBAB(IDENT,OP,TA,TB,N,RES)

      DIMENSION IDENT(3) , TA(50) , TB(50) , TP(50) , STAT(9)
      REAL RES , PROPEG , PROP , PR , STAT
      INTEGER OP , TA , TB , IDENT , N , TYPE , NC , TP

      CALL ACCTYP(IDENT,TYPE)
      GO TO (60,60,60,30) TYPE
      CALL ACCFCT(IDENT,STAT)
      GO TO (10,20) OP
      GO TO 200
   10 RES = 1. / FLOAT(NC(STAT))
      GO TO 200
   20 RES = 1. - 1. / FLOAT(NC(STAT))
      GO TO 200
   30 GO TO (40,50) OP
      GO TO 200
   40 RES = PROPEG(TA,N)
      GO TO 200
   50 RES = PROPEG(TA,N)
      GO TO 200
   60 GO TO (70,80,90,100,110,120,130) OP
      GO TO 200
   70 RES = PROP(TA,TA,N,TYPE)
      GO TO 200
   80 RES = 1. - PROP(TA,TA,N,TYPE)
      GO TO 200
   90 CALL ACCPRP(TP,N,PR)
      RES = PROP(TP,TA,N,TYPE) - PROP(TA,TA,N,TYPE)
      GO TO 200
  100 CALL ACCPRP(TP,N,PR)
      RES = PROP(TP,TA,N,TYPE)
      GO TO 200
  110 CALL ACCPRP(TP,N,PR)
      RES = 1. - PROP(TP,TA,N,TYPE)
      GO TO 200
  120 CALL ACCPRP(TP,N,PR)
      RES = 1. - PROP(TP,TA,N,TYPE) + PROP(TA,TA,N,TYPE)
      GO TO 200
  130 RES = PROP(TA,TB,N,TYPE)
      GO TO 200
  200 CONTINUE
      RETURN
      END
```

```
      FUNCTION PROPEC(TA,N)

      DIMENSION TA(50) , I(50)
      REAL PROPEC , PR
      INTEGER TA , N , I
      LOGICAL EGAL

      CALL ACCPRP(I,N,PR)
10    IF (EGAL(I,TA,N)) GO TO 20
      CALL ACCPRS(I,N,PR)
      GO TO 10
20    CONTINUE
      PROPEC = PR
      RETURN
      END
```

```
      FUNCTION PROP(TA,TB,N,TYPE)

      DIMENSION TA(50) , TB(50) , T(50) , TPR(50)
      REAL PROP , PR , AUX , NBRVAL
      INTEGER TA , TB , N , TYPE , T , IND , TPR
      LOGICAL SUP

      PROP = 0
      CALL ACCPRP(T,N,PR)
      IF (SUP(TA,T,N,TYPE)) GO TO 10
      CALL AFFECT(TA,T,N)
   10 IF (SUP(T,TB,N,TYPE)) GO TO 200
   20 CALL AFFECT(TPR,T,N)
      CALL ACCPRS(T,N,PR)
      IF (SUP(TA,T,N,TYPE)) GO TO 20
      AUX = PR * NBRVAL(TA,T,N,TYPE) / NBRVAL(TPR,T,N,TYPE)
      CALL AFFECT(TPR,TA,N)
   30 IF (.NOT.SUP(TB,T,N,TYPE)) GO TO 40
      PROP = PROP + AUX
      CALL AFFECT(TPR,T,N)
      CALL ACCPRS(T,N,PR)
      AUX = PR
      GO TO 30
   40 CONTINUE
      AUX = AUX * NBRVAL(TPR,TB,N,TYPE) / NBRVAL(TPR,T,N,TYPE)
      PROP = PROP + AUX
  200 RETURN
      END
```

```
      FUNCTION SOMF1(STAT,X,Y,P)

      DIMENSION STAT(9)
      REAL P , STAT , VALF , SOMF1
      INTEGER X , Y , K , LIM , L , I , J , N , IS , JS , NC

      SOMF1 = 0
      K = X
      I = IS(STAT)
      J = JS(STAT)
      N = NC(STAT)
      IF (X.LT.I) K = I
      LIM = 0
10    IF (K.GT.J) GO TO 40
      CALL FK (STAT,K,LIM,VALF)
      IF (VALF) 30,30,20
20    IF (K.NE.LIM) K = INT((K + LIM)/2)
      IF (K.GT.Y) L = Y
      IF (K.LE.Y) L = K
      SOMF1 = SOMF1 + VALF * SOMF1(X,L,K,P,N)
30    K = LIM + 1
      GO TO 10
40    RETURN
      END
```

```
      FUNCTION SOMF2 (STAT,X,Y)

      DIMENSION STAT(9)
      REAL STAT , VALF
      INTEGER X , Y , LIM , K , I , IS

      SOMF2 = 0
      K = X
      I = IS(STAT)
      IF (K.LT.I) K = I
      LIM = 0
10    IF (K.GT.Y) GO TO 20
      CALL FK(STAT,K,LIM,VALF)
      IF (LIM.GT.Y) VALF = VALF * (Y - K + 1) / (LIM - K + 1)
      SOMF2 = SOMF2 + VALF
      K = LIM + 1
      GO TO 10
20    CONTINUE
      RETURN
      END
```

```
      FUNCTION SOMF3 (STAT,X,Y,P)

      DIMENSION STAT(9)
      REAL STAT , P , VALF , PT
      INTEGER X , Y , K , LIM , L , I , IS , J , JS , N , NC

      SOMF3 = 0
      K = X
      I = IS(STAT)
      J = JS(STAT)
      N = NC(STAT)
      IF (K.LT.I) K = I
      LIM = 0
10    IF (K.GT.J) GO TO 40
      CALL FK(STAT,K,LIM,VALF)
      IF (VALF) 30,30,20
20    IF (K.NE.LIM) K = INT((K + LIM) / 2)
      IF (K.GT.Y) K = Y
      K = K - X + 1
      L = K
      SOMF3 = SOMF3 + VALF * PT(K,L,P,N)
30    K = LIM + 1
      GO TO 10
40    CONTINUE
      RETURN
      END
```

```
      FUNCTION SOMF4 (STAT,X,Y,P)

      DIMENSION STAT(9)
      REAL STAT, P , VALF , PI , SOMPI
      INTEGER X , Y , K , L , LIM , COEFF , IND , I , J , N
      INTEGER IS , JS , NC

      SOMF4 = 0
      K = X
      I = IS(STAT)
      J = JS(STAT)
      N = NC(STAT)
      IF (X.LT.I) K = I
      LIM = 0
10    IF (K.GT.J) GO TO 50
      CALL FK(STAT,K,LIM,VALF)
      IF (VALF) 40,40,20
20    IF (K.NE.LIM) K = INT((K + LIM) / 2)
      SOMPI = 0
      DO 30, IND = X, K
      L = IND
      COEFF = Y
      IF (L.LT.Y) COEFF = L
      COEFF = COEFF - X + 1
      SOMPI = SOMPI + COEFF* PI(K,L,P,N)
30    CONTINUE
      SOMF4 = SOMF4 + VALF * SOMPI
40    K = LIM + 1
      GO TO 10
50    CONTINUE
      RETURN
      END
```

```
      FUNCTION SOMF5 (STAT,X,Y)

      DIMENSION STAT(9)
      REAL STAT , VALF
      INTEGER X , Y , K , LIM , I , IS , J , JS

      SOMF5 = 0
      K = X
      I = IS(STAT)
      J = JS(STAT)
      IF (X.LT.I) K = I
      LIM = 0
10    IF (K.GT.J) GO TO 40
      CALL FK(STAT,K,LIM,VALF)
      IF (VALF) 30 ,30, 20
20    IF (K.NE.LIM) K = INT((K + LIM) /2)
      IF (Y.LT.K) K = Y
      SOMF5 = SOMF5 + VALF * (K - X + 1)
30    K = LIM + 1
      GO TO 10
40    CONTINUE
      RETURN
      END
```

```
      FUNCTION SOMF6 (STAT,X,Y)

      DIMENSION STAT(9)
      REAL STAT , VALF
      INTEGER X , Y , K , LIM , I , IS

      SOMF6 = 0
      K = X
      I = IS(STAT)
      IF (K.LT.I) K = I
      LIM = 0
10    IF (K.GT.Y) GO TO 40
      CALL FK(STAT,K,LIM,VALF)
      IF (VALF) 30, 30 ,20
20    IF (LIM.GT.Y) VALF = VALF * (Y - K + 1)/(LIM - K + 1)
      IF (K.NE.LIM) K = INT((K + LIM)/2)
      SOMF6 = SOMF6 + VALF * K
30    K = LIM + 1
      GO TO 10
40    CONTINUE
      RETURN
      END
```

```
      FUNCTION SOMF7 (STAT,X,Y,P)

      DIMENSION STAT(9)
      REAL STAT , P , VALF , SOMP1 , SOMP2 , SOMPI1 , SOMPI2
      INTEGER X , Y , K , LIM , L , I , IS , N , NC

      SOMF7 = 0
      K = X + 1
      I = IS(STAT)
      N = NC(STAT)
      IF (K.LT.I) K = I
      LIM = 0
10    IF (K.GT.Y) GO TO 40
      CALL FK(STAT,K,LIM,VALF)
      IF (VALF) 30, 30 , 20
20    IF (LIM.GT.Y) VALF = VALF * (Y - K + 1)/(LIM - K + 1)
      IF (K.NE.LIM) K = INT((K + LIM)/2)
      L = X - 1
      SOMPI1 = SOMP1(0,L,K,P,N)
      SOMPI2 = SOMP2(X,K,P,N)
      SOMF7 = SOMF7 + VALF * (K * SOMPI1 + X * (K+1) * SOMPI2)
30    K = LIM + 1
      GO TO 10
40    CONTINUE
      RETURN
      END
```

```
      FUNCTION SOMF8 (STAT,P)

      DIMENSION STAT(9)
      REAL STAT , P , VALF , SOMPI , PI
      INTEGER K , LIM , L , IND , I , IS , J , JS , N , NC

      SOMF8 = 0
      K = 2
      I = IS(STAT)
      J = JS(STAT)
      N = NC(STAT)
      IF (K.LT.I) K = I
      LIM = 0
10    IF (K.GT.J) GO TO 50
      CALL FK(STAT,K,LIM,VALF)
      IF (VALF) 40 , 40 , 20
20    IF (K.NE.LIM) K = INT((K + LIM)/2)
      SOMPI = 0
      DO 30 , IND = 1 ,K
      L = IND
      SOMPI = SOMPI + PI(K,L,P,N) * (1 - L * (K + 1)/(L + 1))
30    CONTINUE
      SOMF8 = SOMF8 + VALF * SOMPI
40    K = LIM + 1
      GO TO 10
50    CONTINUE
      RETURN
      END
```

```
      FUNCTION SOMF9 (STAT,Y,P,PN)

      DIMENSION STAT(9)
      REAL STAT , P , PN , VALF, SOMPI , PI
      INTEGER K , LIM , L , A , I , IS , J , JS , N , NC , Y

      SOMF9 = 0
      K = 0
      I = IS(STAT)
      J = JS(STAT)
      N = NC(STAT)
      IF (K.LT.I) K = I
      LIM = 0
10    IF (K.GT.J) GO TO 50
      CALL FK(STAT,K,LIM,VALF)
      IF (VALF) 40 , 40 , 20
20    IF (K.NE.LIM) K = INT((K + LIM)/2)
      SOMPI = 0
      DO 30 , IND = 0, K
      L = IND
      A = L
      IF (L.GT.Y) A = Y
      SOMPI = SOMPI + PI(K,L,P,N) * PN ** A
30    CONTINUE
      SOMF9 = SOMF9 + VALF * SOMPI
40    K = LIM + 1
      GO TO 10
50    CONTINUE
      SOMF9 = SOMF9 / (1 - PN)
      RETURN
      END
```

```
      FUNCTION SOMF10 (STAT,X,Y,P,PN)

      DIMENSION STAT(9)
      REAL STAT , P , PN , VALF , SOMPI , PI
      INTEGER X , Y , K , LIM , L , A , IND , I , J , N
      INTEGER IS , JS , NC

      SOMF10 = 0
      K = X
      I = IS(STAT)
      J = JS(STAT)
      N = NC(STAT)
      IF (K.LT.I) K = I
      LIM = 0
10    IF (K.GT.J) GO TO 50
      CALL FK(STAT,K,LIM,VALF)
      IF (VALF) 40 , 40 ,20
20    IF (K.NE.LIM) K = INT((K + LIM)/2)
      SOMPI = 0
      DO 30, IND = X , K
      L = IND
      A = L
      IF (L.GT.Y) A = Y
      A = A - X + 1
      SOMPI = SOMPI + PI(N,L,P,K) * (1 - PN **A)
30    CONTINUE
      SOMF10 = SOMF10 + VALF * SOMPI
40    K = LIM + 1
      GO TO 10
50    CONTINUE
      SOMF10 = SOMF10 / (1 -PN)
      RETURN
      END
```

```
      FUNCTION SOMF11 (STAT,X,Y,PN)

      DIMENSION STAT(9)
      REAL STAT , PN , VALF
      INTEGER X , Y , K , LIM , I , IS , J , JS

      SOMF11 = 0
      K = X
      I = IS(STAT)
      J = JS(STAT)
      IF (K.LT.I) K = I
      LIM = 0
10    IF (K.GT.J) GO TO 40
      CALL FK(STAT,K,LIM,VALF)
      IF (VALF) 30 , 30, 20
20    IF (K.NE.LIM) K = INT((K + LIM)/2)
      IF (K.GT.Y) K = Y
      SOMF11 = SOMF11 + VALF * (1 - PN ** (K - X + 1) )
30    K = LIM + 1
      GO TO 10
40    CONTINUE
      SOMF11 = SOMF11 / (1 - PN)
      RETURN
      END
```

```
      FUNCTION SOMF12 (STAT,PN)

      DIMENSION STAT(9)
      REAL STAT , PN , VALF
      INTEGER K , LIM , I , IS , J , JS

      SOMF12 = 1
      K = 0
      I = IS(STAT)
      J = JS(STAT)
      IF (K.LT.I) K = I
      LIM = 0
10    IF (K.GT.J) GO TO 40
      CALL FK(STAT,K,LIM,VALF)
      IF (VALF) 30 , 30 , 20
20    IF (K.NE.LIM) K = INT((K + LIM)/2)
      SOMF12 = SOMF12 - VALF * PN ** K
30    K = LIM + 1
      GO TO 10
40    CONTINUE
      SOMF12 = SOMF12 / (1 - PN)
      RETURN
      END
```

```
      FUNCTION SOMF13 (STAT,X,Y,P,PN)

      DIMENSION STAT(9)
      REAL STAT, P , PN , VALF , SOMP1 , SOMP2 , COEFF , SOMPI , PI
      INTEGER X , Y , K , LIM , L , A , IND , I , J , N
      INTEGER IS , JS , NC

      SOMF13 = 0
      K = X
      I = IS(STAT)
      J = JS(STAT)
      N = NC(STAT)
      IF ( K.LT.I) K = I
      LIM = 0
10    IF (K.GT.J) GO TO 70
      CALL FK(STAT,K,LIM,VALF)
      IF (VALF) 60 , 60 , 20
20    IF (K.NE.LIM) K = INT((K + LIM)/2)
      L = X - 1
      SOMPI = K * SOMP1(0,L,K,P,N)
      A = K
      IF (K.GT.Y) A = Y
      DO 30 , IND = X , A
      L = IND
      COEFF = (K+1) * (X-1 + 1/(1-PN))
      COEFF = COEFF - (L - (1+K*PN)/(1-PN)) * PN ** (L-X+1)
      SOMPI = SOMPI + COEFF * PI(K,L,P,N) / (L+1)
30    CONTINUE
      IF (K - Y - 1) 50 , 40 , 40
40    L = Y + 1
      COEFF = (K+1) * (X-1 + (1 - PN ** (Y-X+1))/(1-PN))
      SOMPI = SOMPI + COEFF * SOMP2(L,K,P,N)
50    SOMF13 = SOMF13 + VALF * SOMPI
60    K = LIM + 1
      GO TO 10
70    CONTINUE
      RETURN
      END
```

```
      FUNCTION SOMF14 (STAT,Y)

      DIMENSION STAT(9)
      REAL STAT , VALF
      INTEGER Y , K , LIM , I , IS

      SOMF14 = Y + 1
      K = 0
      I = IS(STAT)
      IF (K.LT.I) K = I
      LIM = 0
10    IF (K.GT.Y) GO TO 40
      CALL FK(STAT,K,LIM,VALF)
      IF (VALF) 30 , 30 , 20
20    IF (LIM.GT.Y) VALF = VALF * (Y - K + 1) / (LIM - K + 1)
      IF (K.NE.LIM) K = INT((K + LIM) / 2)
      SOMF14 = SOMF14 - VALF * (Y + 1 - K)
30    K = LIM + 1
      GO TO 10
40    CONTINUE
      RETURN
      END
```

```
      FUNCTION SOMF15 (STAT,X,Y,Z,P)

      DIMENSION STAT(9)
      REAL STAT , P , VALF , SOMPI , PI
      INTEGER X , Y , Z , K , LIM , L , IND , I , J , N
      INTEGER IS , JS , NC

      SOMF15 = 0
      K = X
      I = IS(STAT)
      J = JS(STAT)
      N = NC(STAT)
      IF (K.LT.I) K = I
      LIM = 0
10    IF (K.GT.J) GO TO 50
      CALL FK(STAT,K,LIM,VALF)
      IF (VALF) 40 , 40 , 20
20    IF (K.NE.LIM) K = INT((K + LIM)/2)
      IF (K.GT.Y) K = Y
      K = K - X + 1
      SOMPI = Z
      DO 30, IND = 0, K-1
      L = IND
      SOMPI = SOMPI + PI(K,L,P,N) * (K+1) / (K-L+1)
30    CONTINUE
      L = K
      SOMPI = SOMPI + K * PI(K,L,P,N)
      SOMF15 = SOMF15 + VALF * SOMPI
40    K = LIM + 1
      GO TO 10
50    CONTINUE
      RETURN
      END
```

```
      FUNCTION SOMF16 (STAT,X,Y,PN)

      DIMENSION STAT(9)
      REAL STAT , PN , VALF
      INTEGER X , Y , K , LIM , I , IS , J , JS

      SOMF16 = 0
      K = X
      I = IS(STAT)
      J = JS(STAT)
      IF (K.LT.I) K = I
      LIM = 0
10    IF (K.GT.J) GO TO 40
      CALL FK(STAT,K,LIM,VALF)
      IF (VALF) 30 , 30 , 20
20    IF (K.NE.LIM) K = INT((K + LIM)/2)
      IF (K.GT.Y) K = Y
      K = K - X + 1
      SOMF16 = SOMF16 + VALF * (X - 1 + (1 - PN ** K)/(1 - PN))
30    K = LIM + 1
      GO TO 10
40    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE FONCT (IDENT,STAT)

      DIMENSION STAT(9) , IDENT(3)
      REAL STAT , PAR1
      INTEGER IDENT , K

      CALL ACCFCT(IDENT,STAT)
      IF (PAR1(STAT) + 1) 20 , 10 , 20
10    CALL CALPAR(STAT)
      CALL ECRIT(IDENT,STAT)
20    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CALPAR (STAT)

      DIMENSION STAT(9)
      REAL STAT , Z , F , X , XC , D , P , Q
      REAL MU , P1 , P2 , MOY , FO , PAR1
      INTEGER COEFF , I , J , K , T , TYP , JS , IS , FO

      Z = 1 / FLOAT(10 ** 6)
      T = TYP(STAT)
      GO TO (10,20,30,40,50,60,70,80,90,100,110,120,130) T
      GO TO 310
10    I = 0
      J = JS(STAT)
      P1 = MOY(STAT) / J
      F = (1 - P1) ** J
      GO TO 210
20    P = 1 / (1 + MOY(STAT))
      P1 = P
      F = P
      I = 0
      J = INT((ALOG(Z) - ALOG(P))/ ALOG(1-P) + 0.5) + 1
      GO TO 200
30    P = 1 / MOY(STAT)
      F = 0
      P1 = P
      I = 1
      J = INT((ALOG(Z) - ALOG(P))/ ALOG(1-P) + 0.5) + 2
      GO TO 200
40    MU = MOY(STAT)
      F = EXP(-MU)
      I = 0
      P1 = 0
      COEFF = 5
      IF (MU.LT.40) COEFF = 6
      IF (MU.LT.9) COEFF = 7
      IF (MU.LT.5) COEFF = 8
      J = INT(MU + COEFF * SQRT(MU) + 0.5)
      GO TO 200
50    I = IS(STAT)
      J = JS(STAT)
      P = 1 / FLOAT(J - I + 1)
      P1 = P
      MU = FLOAT(I + J) / 2
      F = 0
      IF (I.EQ.0) F = P
      GO TO 220
60    I = 0
      J = JS(STAT)
      F = FO(STAT)
      P1 = (1 - F) / J
      MU = (1 - F) * (J + 1) / 2
      GO TO 230
70    MU = MOY(STAT)
      I = 0
      J = INT(2 * MU + 0.5)
      P = 1 / (2 * MU + 1)
      P1 = P
      F = P
      GO TO 200
80    MU = MOY(STAT)
```

```
      I = 1
      J = INT(2 * MU + 0.5) -1
      P1 = 1 / (2 * MU - 1)
      F = 0
      GO TO 200
 90   I = 0
      F = 1 - F0(STAT)
      MU = MOY(STAT)
      J = JS(STAT)
      X = MU / (MU -F)
      X0 = 0
 92   IF (ABS(X - X0).LE.Z) GO TO 94
      X0 = X
      X = J * (MU - F) * X0 ** (J + 1)
      X = X - (J - 1) * MU * X0 ** J
      X = X - MU + J * F
      D = (J + 1) * (MU -F) * X0 ** J
      D = D - MU * J * X0 ** (J - 1)
      D = D - MU + F * (J + 1)
      X = X / D
      GO TO 92
 94   CONTINUE
      P1 = X
      P2 = F * (X-1) * X ** J / (X ** J - 1)
      GO TO 250
100   I = 0
      F = 1 - F0(STAT)
      MU = MOY(STAT)
      K = K0(STAT)
      J = JS(STAT)
      X = MU / (MU - F)
      X0 = 0
      P = MU - K * F
102   IF (ABS(X - X0).LE.Z) GO TO 104
      X0 = X
      X = K * P * X0 ** (K + 1)
      X = X - (K - 2) * P * X0 ** (K - 1)
      X = X - MU + F
      D = (K + 1) * P * X0 ** K
      D = D - (K - 1) * P * X0 ** (K - 2)
      D = D - MU
      X = X / D
      GO TO 102
104   CONTINUE
      X0 = 0
      Q = MU - J * F
106   IF (ABS(X - X0).LE.Z) GO TO 108
      X0 = X
      Y = J * P * X0 ** (J + 1)
      Y = Y - (J - 2) * P * X0 ** (J - 1)
      Y = Y - (J - K) * MU * X0 ** (J - K + 1)
      Y = Y + (J - K - 1) * (MU -F) * X0 ** (J - K)
      Y = Y - (K - 1) * (Q - F) * X0 ** K
      Y = Y + (K - 2) * Q * X0 ** (K - 1)
      D = (J + 1) * P * X0 ** J
      D = D - (J - 1) * P * X0 ** (J - 2)
      D = D - (J - K + 1) * MU * X0 ** (J - K)
      D = D + (J - K) * (MU - F) * X0 ** (J - K - 1)
      D = D - K * (Q - F) * X0 ** (K - 1)
      D = D + (K - 1) * Q * X0 ** (K - 2)
```

```
      X = X / D
      GO TO 106
108   CONTINUE
      P1 = X
      D = X ** (J - 1) * (X+1) - X ** (J - K) - X ** (K - 1)
      P2 = F * (X - 1) * X ** (J - 1) / D
      GO TO 250
110   I = 0
      F = 1 - F0(STAT)
      MU = MOY(STAT)
      J = JS(STAT)
      K = K0(STAT)
      P = 2 * MU - F * K
      Q = MU - J * F
      X = (P + F) / (P - F)
      X0 = 0
112   IF (ABS(X - X0).LE.Z) GO TO 114
      X0 = X
      X = (J + 1) * K * (P - F) * X0 ** (J + 2)
      X = X - 2 * J * K * P * X0 ** (J + 1)
      X = X + (J - 1) * K * (P + F) * X0 ** J
      X = X - 2 * K * (Q - F) * X0 ** (K + 1)
      X = X + 2 * (K - 1) * Q * X0 ** K
      X = X - 2 * Q
      D = (J + 2)  * K * (P - F) * X0 ** (J + 1)
      D = D - 2 * (J + 1) * K * P * X0 ** J
      D = D + J * K * (P + F) * X0 ** (J - 1)
      D = D - 2 * (K + 1) * (Q - F) * X0 ** K
      D = D + 2 * K * Q * X0 ** (K - 1)
      D = D + 2 * (Q - F)
      X = X / D
      GO TO 112
114   CONTINUE
      P1 = X
      D = K * (X - 1) * X ** J - X ** K + 1
      P2 = F * (X - 1) * X ** J / D
      GO TO 250
120   MU = MOY(STAT)
      F = F0(STAT)
      I = 0
      J = 1
      IF (MU.EQ.-1) MU = 1 - F
      IF (F.EQ.-1) F = 1 - MU
      P1 = MU
      GO TO 190
130   F = 1 - F0(STAT)
      MU = MOY(STAT)
      I = 0
      J = 2
      P1 = 2 * F - MU
      P2 = MU - F
      GO TO 240
190   CALL ECMOY(STAT,MU)
200   CALL ECJS(STAT,J)
210   CALL ECF0(STAT,F)
      CALL ECIS(STAT,I)
      GO TO 300
220   CALL ECF0(STAT,F)
      CALL ECMOY(STAT,MU)
      GO TO 300
```

```
230   CALL ECIS(STAT,I)
      CALL ECMOY(STAT,MU)
      GO TO 300
240   CALL ECOS(STAT,J)
250   CALL ECIS(STAT,I)
      CALL ECPAR2(STAT,P2)
300   CALL ECPAR1(STAT,P1)
310   RETURN
      END
```

```
      SUBROUTINE FK (STAT,K,LIM,VALF)

      DIMENSION STAT(9) , SEG(3)
      REAL STAT , SEG , VALF , AUX , AU , F , P
      REAL SOMEX , MOY , F0 , PAR1 , PAR2 , VAL , CHN
      INTEGER K , LIM , MAXPRE , LC , MAX , VALK , IND
      INTEGER T , MC , J , KM , TYP , K0 , JS , MAXCL

      T = TYP(STAT)
      IF (T) 500 , 10 , 120
 10   IF (LIM) 500 , 30 , 20
 20   CALL ACCSVT(SEG)
      LIM = MAXCL(SEG)
      VALF = VAL(SEG)
      GO TO 500
 30   F = F0(STAT)
      IF (K-1) 40 , 60 , 80
 40   IF (F) 80 , 50 , 50
 50   VALF = F
      GO TO 500
 60   IF (F) 80 , 70 , 70
 70   MAXPRE = 0
      GO TO 90
 80   MAXPRE = IS(STAT) - 1
 90   CALL ACCPRM(SEG)
      MC = MAXCL(SEG)
100   IF (K.LE.MC) GO TO 110
      MAXPRE = MC
      CALL ACCSVT(SEG)
      MC = MAXCL(SEG)
      GO TO 100
110   LIM = MC
      VALF = (VAL(SEG) * (LIM - K + 1)) / (LIM - MAXPRE)
      GO TO 500
120   IF (K) 500 , 130 , 140
130   LIM = 0
      VALF = F0(STAT)
      GO TO 500
140   P = PAR1(STAT)
      J = JS(STAT)
      GO TO (190,190,190,190,160,160,160,160,190,190,190,150,150) T
      GO TO 500
150   LIM = K
      IF (K.EQ.1) VALF = P
      IF (K.EQ.2) VALF = PAR2(STAT)
      GO TO 500
160   IF (J - 30) 170 , 170 , 180
170   LIM = K
      VALF = P
      GO TO 500
180   LC = INT(J/30) + 1
      LIM = K + LC - 1
      IF (LIM.GT.J) LIM = J
      VALF = P * (LIM - K + 1)
      GO TO 500
190   IF (J - 30) 200, 200 , 210
200   LIM = K
      GO TO 290
210   LC = INT((J - 9) / 21) + 1
      GO TO (220,230,230,240) T
```

```
      GO TO (230,250,250) (T - 8)
      GO TO 500
220   MAX = INT((MOY(STAT) * (J + 1)) / J)
      GO TO 260
230   MAX = 0
      GO TO 260
240   MAX = INT(MOY(STAT))
      GO TO 260
250   KM = KO(STAT)
      MAX = KM
260   IF (MAX.LE.4) MAX = 5
      IF (K.GE.(MAX-4)) GO TO 270
      LIM = K + LC - 1
      IF (LIM.GT.(MAX-5)) LIM = MAX - 5
      GO TO 290
270   IF (K.GT.(MAX+4)) GO TO 280
      LIM = K
      GO TO 290
280   LIM = K + LC - 1
      IF (LIM.GT.J) LIM = J
290   GO TO (300,310,320,330) T
      GO TO (360,370,410) (T - 8)
      GO TO 500
300   VALK = INT((K + LIM)/2)
      AUX = CNN(J,VALK)
      VALF = AUX * P ** VALK * (1 - P) ** (J - VALK)
      VALF = VALF * (LIM - K + 1)
      GO TO 500
310   VALF = (1 - P) ** K - (1 - P) ** (LIM + 1)
      GO TO 500
320   VALF = (1 - P) ** (K - 1) - (1 - P) ** LIM
      GO TO 500
330   AUX = FO(STAT)
      MU = MOY(STAT)
      DO 340 , IND = 1 ,K
      AUX = AUX * MU / IND
340   CONTINUE
      VALF = AUX
      DO 350 , IND = (K+1), LIM
      AUX = AUX * MU / IND
      VALF = VALF + AUX
350   CONTINUE
      GO TO 500
360   AUX = 1 / P
      VALF = SOMFX(AUX,K,LIM) * PAR2(STAT)
      GO TO 500
370   KM = KO(STAT)
      IF (KM - K) 380 , 390 , 390
380   AUX = 1 / P
      VALF = SOMFX(AUX,K,LIM) * P ** KM
      GO TO 400
390   AUX = P
      VALF = SOMFX (AUX,K,LIM) / P ** KM
400   VALF = VALF * PAR2(STAT)
      GO TO 500
410   KM = KO(STAT)
      AUX = 1 / P
      IF (KM - K) 430 , 420 , 420
420   VALF = LIM - K + 1 - SOMFX(AUX,K,LIM)
      GO TO 440
```

```
430     VALF = SOMFX(AUX,K,LIM) * (P ** KM - 1)
440     VALF = VALF * PAR2(STAT)
        GO TO 500
500     RETURN
        END
```

```
      FUNCTION SOMEX (VAL,M,N)

      REAL VAL , AUX
      INTEGER M , N , IND

      AUX = VAL ** M
      SOMEX = AUX
      DO 10 , IND = (M+1) , N
      AUX = AUX * VAL
      SOMEX = SOMEX + AUX
10    CONTINUE
      RETURN
      END
```

```
      FUNCTION SOMP1 (I,J,K,P,N)

      REAL P , PI
      INTEGER I , J , K , N , L , IND

      SOMP1 = 0
      DO 10, IND = I , J
      L = IND
      SOMP1 = SOMP1 + PI(K,L,P,N)
10    CONTINUE
      RETURN
      END
```

```
      FUNCTION SOMP2 (I,J,P,N)

      REAL P , PI
      INTEGER I , J , N , L , IND

      SOMP2 = 0
      DO 10, IND = I , J
      L = IND
      SOMP2 = SOMP2 + PI(J,L,P,N) / (L+1)
10    CONTINUE
      RETURN
      END
```

```
      FUNCTION PI (K,L,P,N)

      REAL P , CMN
      INTEGER K , L , N , M , I

      IF (P.EQ.1.) GO TO 20
      IF (((P.LE.0.1).OR.(N.LE.500)).AND.(K.GE.(N/5))) GO TO 10
      PI = CMN(K,L) * P ** L * (1.-P) ** (K-L)
      GO TO 20
   10 M = INT(N * P)
      IF (M.LT.L) GO TO 30
      PI = CMN(N,L) / CMN(N,K)
      M = N - M
      I = K - L
      IF (M.LT.I) GO TO 30
      PI = PI * CMN(M,I)
      GO TO 100
   20 IF (K.NE.L) GO TO 30
      PI = 1.
      GO TO 100
   30 PI = 0.
  100 CONTINUE
      RETURN
      END
```

```
      FUNCTION CMN (M,N)

      INTEGER  M , N , K , L
      REAL A , AUX

      IF (M - 40) 10 , 10 , 30
10    K = N
      IF (K.GT.(M-N)) K = M - N
      CMN = 1.
      DO 20 , L = 1 , K
      CMN = (CMN * (M - K + L)) / L
20    CONTINUE
      GO TO 40
30    A = 2. * 3.141592
      AUX = 12. * SQRT(FLOAT(N * (M-N)) / (M*A))
      AUX = AUX / (1. + FLOAT(144 * N * (M-N)) / FLOAT(12 * M + 1))
      AUX = AUX * (FLOAT(M-N) / FLOAT(N)) ** N
      AUX = AUX * (FLOAT(M) / FLOAT(M-N)) ** M
      CMN = AUX
40    CONTINUE
      RETURN
      END
```

```
      FUNCTION SUP(T1,T2,N,TYPE)

      DIMENSION T1(50) , T2(50)
      READ A , B
      INTEGER T1 , T2 , N , TYPE , IND
      LOGICAL SUP

      GO TO (10,20,30) TYPE
      GO TO 200
 10   A = T1(1) * 10. ** T1(2)
      B = T2(1) * 10. ** T2(2)
      SUP = A.GT.B
      GO TO 200
 20   SUP = T1(1).GT.T2(1)
      GO TO 200
 30   IND = 1
 40   IF ((T1(IND).NE.T2(IND)).OR.(IND.EQ.N)) GO TO 50
      IND = IND + 1
      GO TO 40
 50   CONTINUE
      SUP = T1(IND).GT.T2(IND)
      GO TO 200
200   RETURN
      END
```

```
      FUNCTION EGAL(T1,T2,N)

      DIMENSION T1(50) , T2(50)
      INTEGER T1 , T2 , N , IND
      LOGICAL EGAL

      IND = 1
10    IF ((T1(IND).NE.T2(IND)).OR.(IND.EQ.N)) GO TO 20
      IND = IND + 1
      GO TO 10
20    CONTINUE
      EGAL = T1(IND).EQ.T2(IND)
      RETURN
      END
```

```
      SUBROUTINE AFFECT(T1,T2,N)

      DIMENSION T1(50) , T2(50)
      INTEGER T1 , T2 , N , IND

      DO 10 , IND = 1 , N
      T1(IND) = T2(IND)
10    CONTINUE
      RETURN
      END
```

```
      FUNCTION NBRVAL(T1,T2,N,TYPE)

      DIMENSION T1(50) , T2(50)
      REAL NBRVAL
      INTEGER T1 , T2 , N , TYPE , IND , K , AUX

      GO TO (10,20,30) TYPE
      GO TO 200
 10   NBRVAL = T2(1) * 10. ** T2(2) - T1(1) * 10 ** T1(2)
      GO TO 200
 20   NBRVAL = T2(1) - T1(1) + 1
      GO TO 200
 30   K = 27
      IND = 1
 40   IF ((T1(IND).NE.T2(IND)).OR.(IND.EQ.N)) GO TO 50
      IND = IND + 1
      GO TO 40
 50   CONTINUE
      AUX = (T2(IND) - T1(IND) - 1)
      NBRVAL = 2. + FLOAT(AUX * K **(N - IND))
 60   IF (IND.EQ.N) GO TO 70
      IND = IND + 1
      AUX = K - 1 - T1(IND) + T2(IND)
      NBRVAL = NBRVAL + FLOAT(AUX * K ** (N - IND))
      GO TO 60
 70   CONTINUE
      GO TO 200
200   RETURN
      END
```

```
INTEGER FUNCTION TYP(STAT)

DIMENSION STAT(9)
REAL STAT

TYP = INT(STAT(1) + 0.5)
RETURN
END
```

```
REAL FUNCTION MOY(STAT)

DIMENSION STAT(9)
REAL STAT

MOY = STAT(2)
RETURN
END
```

```
REAL FUNCTION F0(STAT)

DIMENSION STAT(9)
REAL STAT

F0 = STAT(3)
RETURN
END
```

```
INTEGER FUNCTION KO(STAT)

DIMENSION STAT(9)
REAL STAT

KO = INT(STAT(4) + 0.5)
RETURN
END
```

```
REAL FUNCTION PAR1(STAT)

DIMENSION STAT(9)
REAL STAT

PAR1 = STAT(5)
RETURN
END
```

```
REAL FUNCTION PAR2(STAT)

DIMENSION STAT(9)
REAL STAT

PAR2 = STAT(6)
RETURN
END
```

```
INTEGER FUNCTION IS(STAT)

DIMENSION STAT(9)
REAL STAT

IS = INT(STAT(7) + 0.5)
RETURN
END
```

```
      INTEGER FUNCTION JS(STAT)

      DIMENSION STAT(9)
      REAL STAT

      JS = INT(STAT(8) + 0.5)
      RETURN
      END
```

```
      SUBROUTINE ECHOY(STAT,P)

      DIMENSION STAT(9)
      REAL STAT , P

      STAT(2) = P
      RETURN
      END
```

```
      SUBROUTINE ECHO(STAT,P)

      DIMENSION STAT(9)
      REAL STAT , P

      STAT(3) = P
      RETURN
      END
```

```
      SUBROUTINE ECPAR1(STAT,P)

      DIMENSION STAT(9)
      REAL STAT , P

      STAT(5) = P
      RETURN
      END
```

```
      SUBROUTINE ECPAR2(STAT,P)

      DIMENSION STAT(9)
      REAL STAT , P

      STAT(6) = P
      RETURN
      END
```

```
      SUBROUTINE ECIS(STAT,K)

      DIMENSION STAT(9)
      REAL STAT
      INTEGER K

      STAT(7) = FLOAT(K)
      RETURN
      END
```

```
      SUBROUTINE ECOS(STAT,K)

      DIMENSION STAT(9)
      REAL STAT
      INTEGER K

      STAT(8) = FLOAT(K)
      RETURN
      END
```

```
INTEGER FUNCTION MAXCL(SEG)

DIMENSION SEG(3)
REAL SEG

MAXCL = INT(SEG(2) + 0.5)
RETURN
END
```

```
REAL FUNCTION VAL(SEG)

DIMENSION SEG(3)
REAL SEG

VAL = SEG(3)
RETURN
END
```