THESIS / THÈSE

MASTER EN SCIENCES INFORMATIQUES

Etude et réalisation d'un interface pour l'implémentation d'un système de gestion de base de données relationnelle
un exemple d'utilisation dans le cadre de la protection

Materne, Cécile

Award date:
1977

Awarding institution:
Universite de Namur

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX - NAMUR

INSTITUT D'INFORMATIQUE

ANNEE ACADEMIQUE 1976-1977

Etude et Réalisation d'un Interface pour l'Implémentation d'un système de Gestion de Base de Données Relationnelle.

UN EXEMPLE D'UTILISATION DANS LE CADRE DE LA PROTECTION.

CÉCILE MATERNE

Mémoire présenté en vue de l'obtention du grade de Licencié et Maître en Informatique.

Nous remercions tout spécialement les chercheurs de l'équipe "Grands Fichiers" à l'Institut d'Informatique de Namur et particulièrement Monsieur Claude DEHENEFFE, pour l'aide précieuse et les conseils qu'ils ont bien voulu nous donner dans l'étude de l'implémentation du modèle relationnel de CODD.

Nous tenons également à exprimer notre reconnaissance envers Monsieur DELOBEL professeur à l'université scientifique et médicale de Grenoble ainsi qu'envers ses assistants qui nous ont accueillie dans leur équipe lors de notre stage.

TABLE DES MATIERES

0.- I N T R O D U C T I O N

0.1. L'INFORMATION AU SEIN D'UNE ENTREPRISE.

Au cours de ce travail, afin de rendre plus suggestives les notions développées, nous nous basons sur des exemples que nous supposerons extraits des données d'une entreprise industrielle.

Notre entreprise industrielle "type" :

- achète des matières premières.
- les transforme.
- les revend sur certains marchés afin de réaliser un profit.

Globalement, l'ensemble de ces activités peut se traduire en 4 flux d'informations interdépendants :
1) flux achats.
2) flux personnel.
3) flux fabrication.
4) flux ventes.

La survie, voire le développement de l'entreprise repose sur la nécessité de posséder des informations fiables afin de pouvoir décider et agir en temps voulu. Il est donc fondamental de maîtriser les flux d'informations circulant à travers elle, si l'on veut assurer un bon fonctionnement à tous ses organes vitaux.

Ces informations, il faut les stocker et les conserver ; cela se fait par l'intermédiaire des fichiers de données. Un fichier n'est pas un tout en soi ; c'est un ensemble homogène d'articles. L'information, objet de traitement, est un morcellement d'un réel pris dans un ensemble quelconque de données. Mais, à partir du moment où l'entreprise désire rendre compte exactement de la réalité, la complexité s'accroît et l'information

s'intègre dans un système d'informations, et devient un véritable outil de gestion.

Actuellement, le système d'aide à la gestion et à la direction d'une entreprise a évolué de façon à faciliter le contrôle des opérations et la conduite des affaires en général. Ce système agit en quelque sorte comme un grand mécanisme qui collecte, stocke, traite et produit de l'information.

0.2. FICHIERS, BASES DE DONNEES.

Dans tout système d'informations, tant informatique que manuel, les fichiers sont la base sur laquelle est construite la structure des applications. Il est dès lors normal d'insister sur leur importance et plus particulièrement sur leur organisation, leur maintenance et leur accessibilité.

Actuellement, une nouvelle aide à la gestion des fichiers tend de plus en plus à s'imposer : la notion de base de données.
Dès que les besoins de l'entreprise exigent des traitements multiples et différents, le seul concept de fichier apparaît inadéquat pour satisfaire les demandes d'informations souvent fortement corrélées.

Avec les anciennes méthodes, les divers départements ou groupes fonctionnels d'une entreprise maintenaient à jour leurs propres fichiers, les informations de ces derniers étant organisées de façon à satisfaire les besoins particuliers de chaque département. Ceci entraînait une prolifération de données souvent conservées en plusieurs exemplaires.

Au lieu de constituer un fichier client, un fichier produit,... pourquoi ne pas fondre tout en un seul réservoir commun, la base de données ?

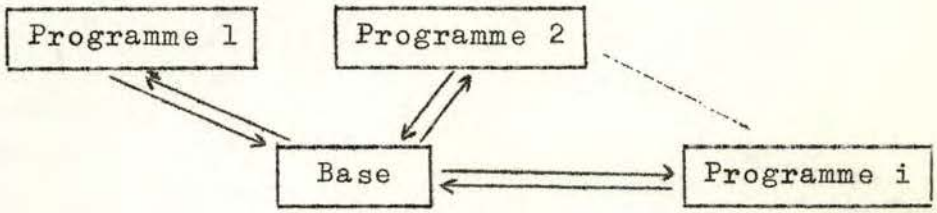
Il existe une différence fondamentale entre une base de données et un fichier, qui n'est une affaire ni de taille ni de complexité. Le terme "base de données" n'est pas non plus simplement un nom que l'on donnerait à un ensemble de fichiers, il a certaines implications particulières sur les caractéristiques de ces fichiers.

- La qualité la plus importante que nous pouvons attribuer à une base de données, contrairement à un fichier "ordinaire", est son caractère commun. Son organisation doit pouvoir satisfaire une grande variété d'applications différentes, rendre compatible l'accès ponctuel et "l'accès de masse", permettre des travaux par lots, en temps réel...

  De par son caractère commun, le système de base de données doit offrir la possibilité d'une gestion automatique des contraintes d'intégrité.

- Une autre qualité est son indépendance vis-à-vis des applications : les données et les programmes d'applications qui les utilisent sont indépendants si des modifications de l'organisation desdonnées n'impliquent pas des changements des programmes et inversément.
  Ainsi, le but que l'on désire atteindre est la compatibilité des programmes d'applications prévus ou non.

  Dans ce schéma, nous avons représenté les interactions de la base de données avec l'environnement que constitue l'entreprise.



Les différents programmes puisent simultanément leurs données dans le même ensemble, étant donné que la base de données possède une signification invariante par rapport à ses différents utilisateurs.

Nous pouvons donc définir une base de données comme une collection de données généralement fortement corrélées, stockées sur des supports avec une redondance contrôlée et qui satisfassent de manière optimale une grande variété d'applications. Les données sont stockées de manière indépendante des programmes qui les utilisent.

0.3. EXPLOITATION DE LA BASE DE DONNEES.

Nous savons maintenant qu'il est préférable, pour des raisons d'homogénéité, que l'ensemble des données d'une entreprise soit contenu dans un "pool" unique et accessible à tous les utilisateurs, la base de données.
Etant donné son unicité et sa non-redondance, cette dernière doit permettre d'obtenir une information à une échelle beaucoup plus étendue et plus globale que ne le permettent les applications isolées.

Toute base de données suppose donc un système de gestion de base données (S.G.B.D.). Par "exploiter", il est bon de souligner que nous entendons "questionner", "mettre à jour", "supprimer " et "protéger" les informations de la base de données.

1) Définition d'un S.G.B.D.

Un système de gestion de base de données (S.G.B.D.) est l'ensemble des programmes qui assurent :
- la structuration.
- le stockage.
- la mise à jour.

- la recherche.
- l'exploitation.

des informations dans une base de données, et qui au plus assume un certain nombre de fonctions particulières, de par l'intégration des données, pour maintenir la consistence, l'intégrité de la protection des données.

Les critères permettant d'étudier un S.G.B.D. sont les suivants :
- organisation des données.
- langages de manipulation de données (L.M.D.).
- interfaces externes.
- protections.

2) Organisation des données.

Le choix d'une structure permet d'exprimer les relations fondamentales existant entre les données, mais facilite aussi l'expression des manipulations que l'on souhaite effectuer sur ces données.

Pour assurer la compatibilité de toutes les applications, la conception de la base doit posséder les caractéristiques suivantes :

a)- La vue des programmes d'application sur les données doit être indépendante de la représentation physique des données, le système de gestion assurant l'interprétation d'un niveau vers l'autre (indépendance physique).

b)- Les besoins d'information peuvent évoluer avec le temps ; de nouvelles applications se créant cependant, ces modifications ne doivent pas entraîner la révision des programmes d'application existant (indépendance logique).

Conséquemment la description de la base de données exige trois niveaux distincts :

1°- Description logique des données pour les applications .

2°- Description logique globale des données (structure logique).

3°- Description physique des données (structure physique).

a) Description logique des données pour les applications.

Cette description est stockée dans un sous-schéma, elle concerne les informations telles qu'elles sont perçues par les programmes d'application, représentant les relations de dépendance qui peuvent exister entre objets et leurs attributs.

b) Description logique globale de la base de données.

Cette description, stockée dans le schéma, concerne l'ensemble intégré de toutes les informations.

c) Description physique de la base de données.

Elle concerne la représentation et l'organisation des données en mémoire physique, en termes d'index, de chaînages, de pointeurs.

Ces trois niveaux sont complémentaires et forment ce que l'on peut appeler le "modèle de la base de données". Celui-ci est donc le filtre vis-à-vis du réel, par lequel passent toutes les requêtes en informations.

La structure des données doit être mémorisée de façon à être accessible par le S.G.B.D. elle est généralement mémorisée dans un catalogue spécial que l'on appelle "Schéma". Un schéma est donc la description globale des informations contenues dans la base de données.

3) Langages de manipulation.

Le langage de manipulation permet de définir les opérations que l'on désir appliquer aux données. Il est évident que ce

langage tient compte de la structure des données préalablement définie.

Le langage d'interrogation assure :

- L'introduction, la suppression, la mise à jour de données.
- Le contrôle de validité de ces données.
- La définition d'un algorithme de traitement.
- L'extraction de données.
- Le formattage des résultats de sortie.

Suivant les systèmes, les langages de manipulation de données permettent d'effectuer tout ou partie des opérations énumérées ci-avant en temps réel. La mise à jour risque de soulever des problèmes délicats d'accès (deadlocks), car plusieurs utilisateurs peuvent souhaiter modifier les mêmes données dans le même laps de temps.

4) Les interfaces externes.

Un S.G.B.D. doit permettre l'utilisation multi-accès en conversationnel des bases de données qu'il gère. Cette utilisation se fait généralement à partir d'une ou de plusieurs consoles.

5) Les protections.

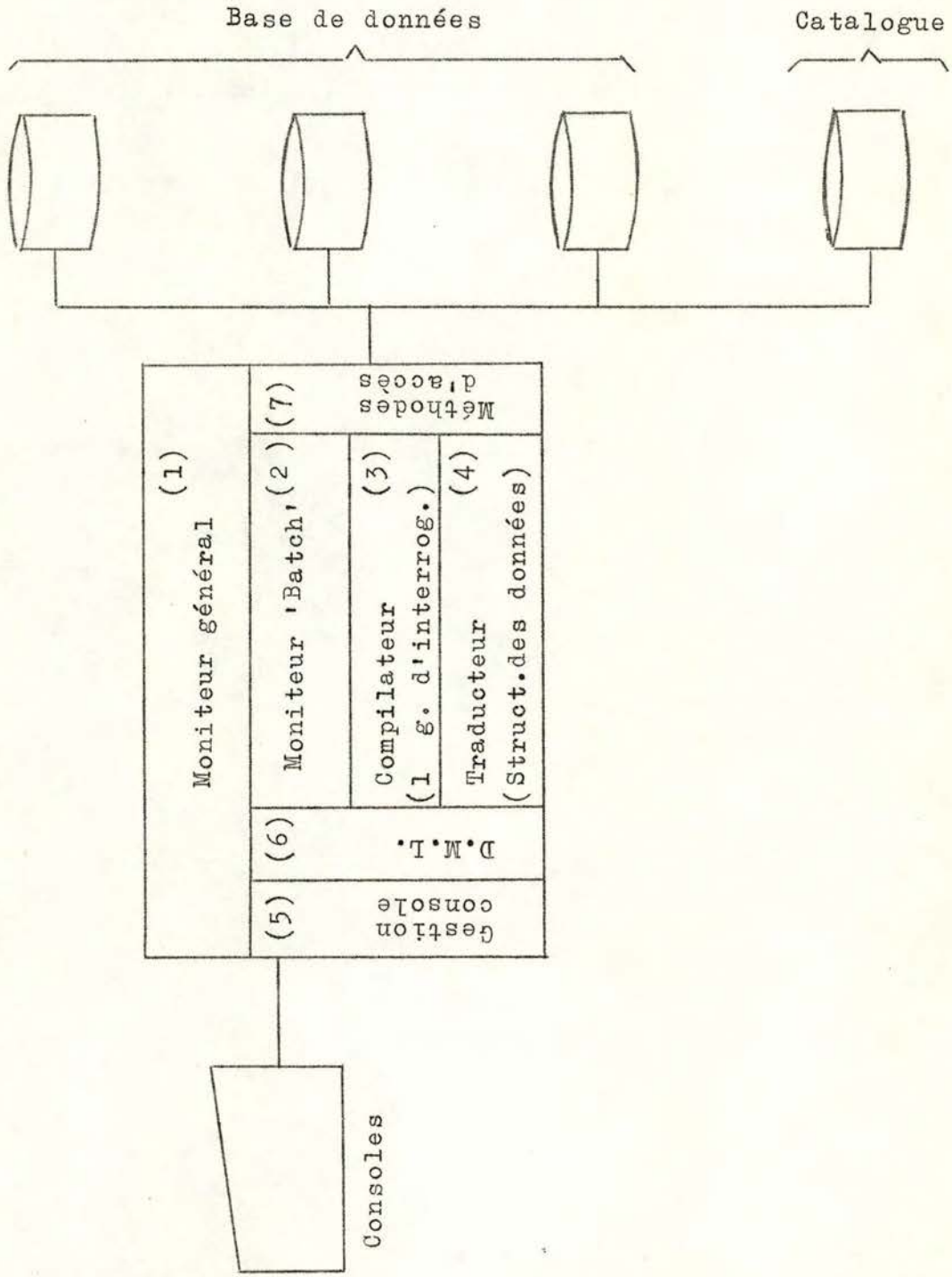
Un S.G.B.D. offre le moyen de se protéger par l'utilisation de mots de passe qui permettront :

- aux différentes banques de se protéger les unes par rapport aux autres.
- dans une même banque, de ne pas permettre à n'importe qui l'accès à n'importe quelle donnée.

Le S.G.B.D. a ainsi des dispositifs permettant d'assurer la sécurité et la cohérence des données.

En conclusion, le S.G.B.D. peut être résumé selon le schéma

suivant :


Base de données
Catalogue
Moniteur général
(1)
Moniteur 'Batch'
(2)
Méthodes d'accès
(7)
Compilateur
(1 g. d'interrog.)
(3)
Traducteur
(Struct.des données)
(4)
D.M.L.
(6)
Gestion console
(5)
Consoles

Dans la suite de ce mémoire, nous présentons quatre parties, chacune d'elles étant une étape dans l'étude de l'implémentation d'une base de données construite selon le modèle de CODD.

- Dans la première partie, nous donnons une définition du modèle relationnel de CODD.
- La deuxième partie décrit les caractéristiques d'un interface de base de données relationnelles appelé GAMMA-O.

Nous développons dans la troisième partie l'implémentation que nous avons adoptée comme base de notre programmation.

Enfin, la quatrième partie contient la description d'un mécanisme de protections des données de cette base.

CHAPITRE I.

LE MODELE RELATIONNEL DE CODD.

Alain Pirotte (I) souligne que le terme "modèle" est souvent employé de manière ambiguë. Très souvent, "modèle de données" simplifie le contenu informatique de la base de données tel qu'il est vu par les utilisateurs ; c'est donc le schéma.

Dans d'autres cas, il désigne un formalisme, ou une théorie pour décrire un schéma de base de données.

Le terme est ici utilisé dans son sens second. Le modèle relationnel est donc un formalisme, basé sur la théorie mathématique des relations pour décrire un schéma de base données.

Différents modèles de bases de données (hiérarchiques ou modèles de réseaux) peuvent être développés afin de traduire les relations sémantiques existant dans la structure des informations. Cependant, dans de nombreux cas, le modèle relationnel se révèle supérieur aux modèles hiérarchiques et aux modèles de réseaux. Il permet entre autres de décrire les informations en conservant leurs structures naturelles, ce qui signifie la compréhension de l'utilisateur vis-à-vis de la base de données. Le modèle relationnel constitue également la base d'un langage de haut niveau, garantissant la plus grande indépendance entre l'utilisateur et la machine.

Trois types de dépendance alourdissent encore les autres modèles :
- dépendance des données vis-à-vis de la clé de tri.
- vis-à-vis de l'indexation.
- vis-à-vis des chemins d'accès.

CODD montre sur les exemples que le modèle relationnel basé sur des relations normalisées est le seul qui puisse garantir une indépendance totale.

Une structure de base de données : le modèle de CODD.

Quelques notions préalables vont être introduites, afin de nous rendre compte de la façon dont est structurée une base de données relationnelles.

1.1. DEFINITIONS.

Domaine simple : Nous appelons "domaine simple", l'ensemble des valeurs homogènes d'une composante élémentaire d'un fait ou d'un objet. Chacune de ces valeurs est appelée une donnée. Pour exprimer que x est une donnée appartenant à d, nous écrivons : $x \varepsilon d$

Exemple

Nom	Ville	Age
Dupont	Namur	24
Durant	Andenne	28
Arnould		21
.....	....	....

Nom, ville et âge sont des domaines simples.

Domaine composé : C'est une liste finie et ordonnée de domaines simples, non nécessairement distincts. Si le domaine composé D comprend n domaines simples ($n > 1$), les éléments de D sont appelés des n-uplets. Les différentes composantes du n-uplet sont des occurrences des n domaines simples.

$D = (d\ 1, d\ 2, d\ 3)$ où $d\ 1$, $d\ 2$, $d\ 3$, sont des domaines simples.

$x \varepsilon D \quad x = (x\ 1, x\ 2, x\ 3)$ où $x\ 1\ \varepsilon\ d\ 1$

$x\ 2\ \varepsilon\ d\ 2$

$x\ 3\ \varepsilon\ d\ 3$

Exemple.

ADRESSE

NOM	NUMERO	RUE	VILLE
...	.....	...	.....

DATE

JOUR	MOIS	ANNEE
....	....	.....

ADRESSE et DATE sont des domaines composés.

Relation n-aire : d 1, d 2, d 3, ... d n, étant n domaines non nécessairement distincts, une relation n-aire est une partie du produit cartésien d 1 x d 2 x d 3 ... x d n (d 1, d 2, .. d n) est appelé l'espace des champs de la relation. Chaque élément de la relation aura donc la forme d'un n-uplet. Le nombre d'éléments contenus dans une relation est la cardinalité de la relation.

n = degré de la relation.

Exemple

ADRESSE

	NOM	NUMERO	RUE	VILLE
CLIENT	DUPONT	3	GRANDE	NAMUR
	DURAND	7	DUBOIS	JAMBES

degré (client) = 2

card (client) = 2

Exemple.

	DATE				
COMMANDE	NO	JJ	MM	AA	CLIENT
	1	03	02	75	CHARNIN
	2	26	04	75	DUPONT
	3	14	02	77	JACQUES

CLIENT est une relation binaire.

COMMANDE est une relation 3-aire.

degré (commande) = 3.

card (commande) = 3.

Une relation peut être vue comme une table à n colonnes (n = degré de la relation) et possédant un nombre arbitraire de lignes. Cette table possède également les propriétés suivantes :

1)- Elle est homogène dans ses colonnes. Les valeurs apparaissant dans une colonne sont toutes de la même nature.

2)- Les lignes sont distinctes entre elles.

3)- L'ordre des lignes à l'intérieur de la table est immatériel.

4)- On assigne des noms distincts aux colonnes de la table (ex : NUMERO, JOUR, NOM, ...) et l'ordre des colonnes est immatériel.

5)- Toutes les valeurs apparaissant dans une relation sont des nombres ou des chaînes de caractères (string).

Cette vision des relations sous forme de tables ne peut s'appliquer qu'à une certaine catégorie de relations, celles qui ne sont définies que sur des domaines simples.
Ces relations sont dites normales simples.

Une occurence d'une relation est constituée par une de ses li-

gnes ; nous appelons uplet cette occurrence.

1.2. CONCEPT DE CLE PRIMAIRE D'UNE RELATION.

Supposons que nous sommes arrivés au stade où la base de données a pu être structurée sous forme de relations.
L'utilisateur de la base de données sait quelles sont les relations existantes, c'est-à-dire qu'il connaît leurs noms et les noms de leurs différents domaines.
Une question se pose : comment pourra-t-il sélectionner un ou plusieurs uplets à l'intérieur d'une relation ?

L'étude de la dépendance fonctionnelle va nous permettre de déterminer quels sont les domaines de la relation qui pourraient faire partie de la clé d'identification des uplets.

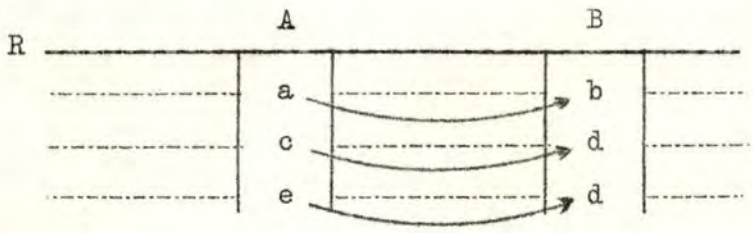
Dépendance fonctionnelle.

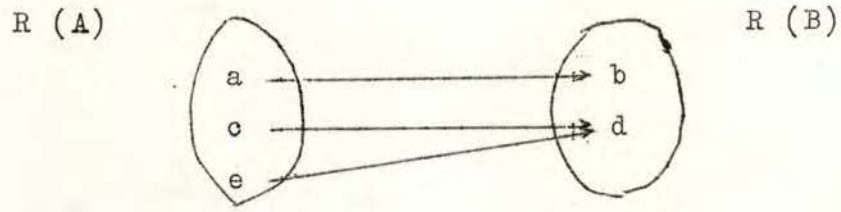
1) Le problème fondamental est de déterminer quels sont les domaines de la relation qui sont fonctionnellement dépendants d'autres domaines de la même relation.

Définition : soit R une relation.
soient A et B deux de ses domaines simples.
Un domaine B est fonctionnellement dépendant d'un domaine A, si à chaque instant R définit une fonction entre l'ensemble des valeurs de A et l'ensemble des valeurs de B.

La loi de correspondance est la suivante : une valeur de A correspond à une valeur de B si elles se trouvent toutes les deux dans le même uplet.

Exemple.





Il existe bien une fonction entre l'ensemble des éléments de A et l'ensemble des éléments de B.

En d'autres termes, la relation R restreinte à A et B, que nous notons R (A,B) (nous verrons plus loin qu'il s'agit d'une projection), est à chaque instant une fonction de R (A) dans R (B).

$$R\ (A,\ B)\ :\ R\ (A) \longrightarrow R\ (B).$$

<u>Notation</u> :- B est fonctionnellement dépendant de A dans la relation R ; se note $R.A \longrightarrow R.B$.

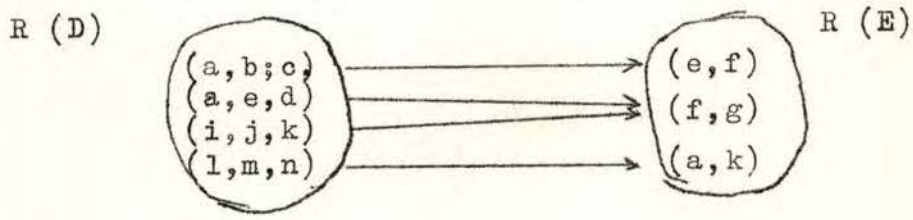
- B n'est pas fonctionnellement dépendant de A dans la relation R ; se note $R.A \not\longrightarrow R.B$.
- si $R.A \longrightarrow R.B$ et $R.B \longrightarrow R.A$, alors A et B sont dans une correspondance biunivoque. Cela se notera $R.A \longleftrightarrow R.B$.

2) La notion peut s'étendre facilement à un ensemble de domaines.

<u>N.B.</u> D et E étant des ensembles de domaines, un élément de D ou de E est un uplet.

<u>Exemple</u>.

D				E	
R 1	R 2	R 3	R 4	R 5	R 6
a	b	c	.....	e	f
a	e	d	.....	f	g
i	j	k	.....	f	g
l	m	n	.....	a	k



Il existe bien une fonction entre R (D) et R (E).

$$R.D. \longrightarrow R.E.$$

3) <u>Clés candidates</u> : Soit R une relation. Une clé candidate de la relation R est une combinaison de domaines de R possédant les propriétés suivantes :

- Une identification unique : Dans chaque uplet de R, la valeur de la clé candidate identifie uniquement cet uplet. Il existe une bijection entre l'ensemble des uplets d'une relation, et l'ensemble des valeurs de la clé candidate de cette relation.

<u>Exemple</u> : K = clé candidate de la relation R.

ω = ensemble de tous les domaines de R.

On peut dire que $R.K. \longrightarrow R.\omega$. Ce qui exprime bien qu'à une valeur de K, on ne peut associer qu'une et une seule valeur de ω .

- Une non redondance : si on enlève un des domaines de K, la propriété d'identification unique n'est plus valable.

<u>REMARQUES</u> :

a) Il existe toujours une clé candidate dans une relation. Elle peut valoir ω , ou la combinaison de tous les domaines.

b) La deuxième propriété fait apparaître le fait que la clé candidate est une clé minimum d'identification.

c) La clé candidate possède également deux autres propriétés :

- Chaque domaine de R est fonctionnellement dépendant de chaque clé candidate de R.
- L'ensemble de données de K est fonctionnellement indépendant de chaque sous-ensemble propre de domaines

de K et aucun autre domaine de R ne peut être ajouté sans détruire cette indépendance fonctionnelle.

4) Domaine primaire : Un domaine d'une relation est primaire lorsqu'il appartient à au moins une clé candidate de cette relation.

5) Clé primaire : Une clé primaire est une clé choisie arbitrairement parmi les clés candidates.

Contrairement aux clés candidates, les composantes de la clé primaire sont toujours définies.

3) pour plus de clarté, nous avons préféré nous limiter ici à une présentation sommaire. Nous décrivons le principe de la recherche filtrée et le mécanisme complet du scan dans le paragraphe consacré aux opérations dans GAMMA - O.

1.3. NORMALISATION DES RELATIONS.

INTRODUCTION.

Lors de la construction de la base de données relationnelle, une relation dont tous les domaines sont simples se représentera par un tableau à deux dimensions. Les colonnes représentant les domaines et les lignes, les différents uplets.

D1	D2	D3	D4	D5	.....	Dn

Une structure plus compliquée est nécessaire pour représenter des relations possédant un ou plusieurs domaines composés.

La normalisation consiste en fait en une standardisation qui remplace une collection de relations données par des collec-

tions successives où les relations ont une structure progressivement plus simple et plus régulière. Cette procédure s'effectue en trois étapes :

- 1. Elimination des domaines composés.
- 2. Recherche d'une dépendance fonctionnelle totale.
- 3. Recherche d'une dépendance transitivité.

Pour plus de clarté et de compréhension, nous basons sur un exemple notre étude de la normalisation.

<u>Exemple</u> : soit la relation P (produit) possédant les domaines suivants :

	Domaines	Types
NP	n° du produit	simple
DP	description du produit	simple
QS	quantité en stock	simple
UF	unité de fabrication de P	composé

UF constitue un domaine composé de P. Il possède les domaines suivants :

	Domaines	Types
NUF	n° de l'unité de fabrication	simple
NR	nom des responsables	simple
FC	fabrication en cours	simple
FM	capacité maximum	simple

Si nous prenons à un instant donné la relation P, nous avons par exemple les valeurs suivantes :

clé primaire

P(NP,	DP,	QS,	UF(NUF,	NR,	FC,	FM))
203	CAM	30	12	DUPONT	5	7
			73	DURANT	7	86

Exemple suite :

clé primaire

P((NP),	DP,	QS,	UF(NUF,	NR,	FC,	FM))
206	CST	155	12	DUPONT	7	33
			29	MAX	25	86
			36	ZENON	16	111

Conditions pour la normalisation.

1) La clé primaire peut être composée de plus d'un domaine.
2) Un domaine non primaire peut être non-simple. C'est-à-dire qu'il constitue en lui-même une relation.
3) Les domaines non primaires peuvent ne pas être fonctionnellement dépendants de l'entièreté de la clé primaire, mais seulement d'une partie de celle-ci.
4) Des domaines non primaires peuvent être fonctionnellement dépendants entre eux.

1.3.1. LA PREMIERE FORME NORMALE.

La première forme normale élimine les domaines non simples d'une relation. Ce processus de normalisation consiste à détacher les domaines non-simples et à propager la clé primaire de la relation, dans les nouvelles relations créées. Ce processus est répété jusqu'à ce que toutes les relations possèdent uniquement des domaines simples.

En reprenant notre exemple, nous remarquons que les domaines NP, DP et QS sont simples. Tandis que UF est un domaine composé et possède autant de uplets que d'unités de fabrication produisant le produit P identifié par la clé primaire NP. Dans cet état, la relation P est dite non normalisée.

Pour convertir P en la première forme normale, il faudra l'éclater en deux relations P 1 et PUF 1 respectivement de clés primaires NP et (NP, NUF).

Au même instant que précédemment, les deux relations se présentent comme suit :

P 1

(NP) clé primaire	DP	QS
203	CAM	30
206	CST	155

PUF 1

(NP NUF) clé primaire		NR	FC	FM
203	12	DUPONT	5	7
203	73	DURANT	7	86
206	12	DUPONT	7	33
206	29	MAX	25	89
206	33	ZENON	16	111

A ce stade, tous les domaines sont de type simple. On peut donc dire qu'une relation est dans la première forme normale si elle jouit de la propriété suivante : "Aucun des domaines de la relation n'a d'éléments qui eux-mêmes sont des relations".

La conversion s'effectue en deux étapes :

- migration de la clé primaire de l'ancienne relation vers les nouvelles.
- les nouvelles relations sont des restrictions de l'ancienne par rapport aux domaines appropriés.

1.3.2. LA SECONDE FORME NORMALE.

Dépendance fonctionnelle totale. Soit R une relation. Soient D et E des ensembles distincts de domaines de R où

$$R.D. \longrightarrow R.E.$$

Si E n'est pas fonctionnellement dépendant de n'importe quel sous-ensemble de D (distinct de D), alors on dit que E dépend totalement de D dans la relation R.

Une relation R est en seconde forme normale si :

- elle est en première forme normale.
- chaque attribut non primaire de R dépend totalement de chaque clé candidate de R.

La conversion de la première à la seconde forme normale s'accomplit en remplaçant la relation de départ par restrictions de cette relation, telles que leurs domaines non primaires dépendent totalement des clés candidates.

Si on reprend l'exemple, dans la relation PUF 1, on peut faire les constatations suivantes :

- FC et FM dépendent totalement de la clé primaire.
- NR ne dépend fonctionnellement que de NUF.

Ceci entraîne nécessairement les conséquences suivantes :

* si une unité de fabrication est ouverte, on ne pourra connaître les renseignements relatifs à cette unité de fabrication que lorsqu'elle deviendra opérationnelle, c'est-à-dire au moment où elle fournira effectivement un produit.

  anomalie d'insertion dans la base de données.

* si une unité de fabrication ne fabrique qu'un seul produit, et que les informations relatives à ce produit sont supprimées, les informations restantes concernant l'unité de fabrication seront également supprimées.
  Ceci se comprend par le fait que le n° de produit intervient dans la clé primaire.

  anomalie de suppression dans la base de données

* si une information relative à une unité de fabrication est modifiée , cela provoquera autant de mises à jour que de produits fabriqués par cette unité de fabrication, à l'instant de la mise à jour.

  anomalie de mise à jour dans la base de données.

Conversion.

Nous remplaçons (P 1 et PUF 1) par 3 relations P 2, PUF 2 et UF 2 possédant respectivement les clés primaires (NP),(NP, NUF) et(NUF).

P 2

(NP) clé primaire	DP	QS
203	CAM	30
206	CST	155

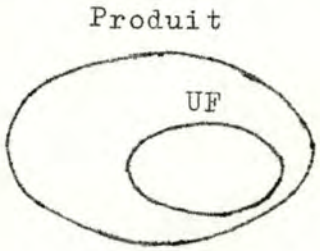
PUF 2

(PN	NUF) clé primaire	FC	FM
203	12	5	7
203	73	7	86
206	12	7	33
206	29	25	89
206	36	16	111

UF 2

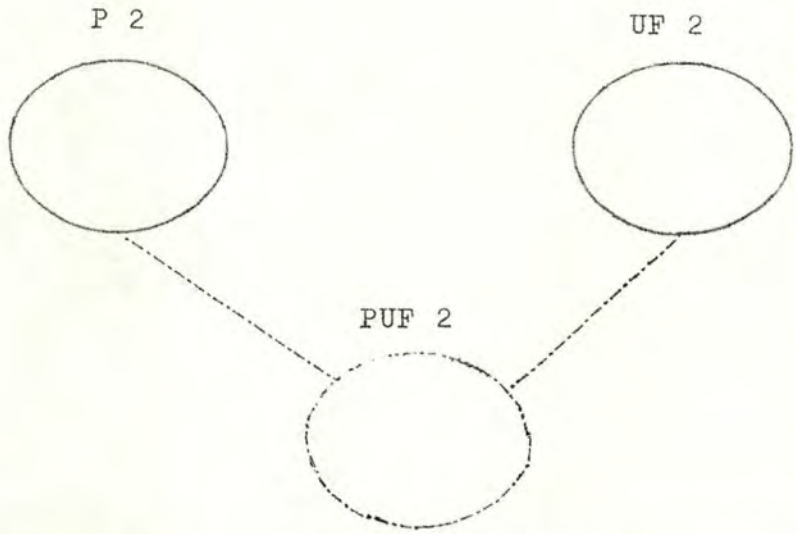
(NUF) clé primaire	NR
12	DUPONT
73	DURANT
29	MAX
36	ZENON

Il est très intéressant d'examiner l'évolution de la structure de la base de données. Au départ (voir notre exemple), la base de données était représentée sous la forme relationnelle, mais d'une manière brute, directement selon la saisie des données, et elle ne faisait pas ressortir les différentes unités. Lorsque la relation produit était sous forme non normalisée, nous étions en présence de cette situation :



L'unité de fabrication ne pouvait exister que par sa dépendance aux produits qu'elle fournissait. Après la normalisation en la seconde forme normale, la structure s'est modifiée de la

façon suivante :



Les produits et les unités de fabrications constituent à présent des entités indépendantes. La base de données est formée de deux types de relations, d'une part les relations qui représentent des entités effectives (clients, produits, unité de fabrication) et d'autre part, les relations qui représentent des liens entre plusieurs entités (encours de fabrication, commande clients ...). Généralement, ce second type est plus dynamique que le premier.

<u>REMARQUE</u> : Nous avons vu que lorsque nous normalisons, nous éclatons les relations en plusieurs autres. Cette opération augmente le nombre de relations de la base de données, et donc, nécessite la création de nouveaux noms de relations. L'utilisateur risque dès lors de faire des confusions vu le trop grand nombre de relations existantes. Pour cette raison, lorsque nous éclaterons les relations, nous le ferons en un nombre minimum de relations.

A partir d'une même base de données initiale, nous pouvons obtenir plusieurs bases de données C 1, C 2, ... C n, qui sont

des normalisations de la base de données de départ, mais qui contiennent des nombres différents de relations. A partir de ces n bases de données, nous pouvons cependant extraire les mêmes informations. La base de données qui contient le moins possible de relations est dite être "dans la seconde forme normale optimale".

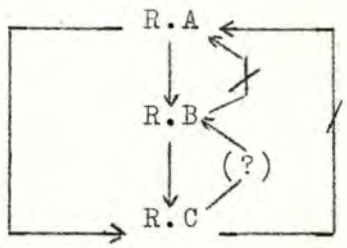
1.3.3. LA TROISIEME FORME NORMALE.

Le but de la conversion en la 3e forme normale est de supprimer les dépendances transitives à l'intérieur des relations.

Dépendance transitive : Soit R une relation. Soit A, B, C trois sous-ensembles distincts de domaines de R, tels que

$$\left.\begin{array}{ll} R.A \longrightarrow R.B & R.B \not\longrightarrow R.A \\ R.B \longrightarrow R.C & \end{array}\right\} \Rightarrow \begin{array}{l} R.A \longrightarrow R.C. \\ R.C \not\longrightarrow R.A. \end{array}$$

Schématiquement :

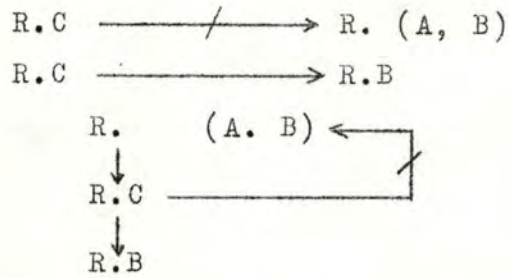


Nous disons que C est transitivement dépendant de A, dans la relation R.

La figure indique que nous ne possédons aucun renseignement sur la relation R.C. $\rightarrow$ R.B. Cependant, si R.C. $\rightarrow$ R.B ; alors B et C sont transitivement dépendants de A dans la relation R.

REMARQUE : Il ne sera pas toujours possible de refouler toutes les dépendances transitives sans casser une clé, ou perdre de l'information. Ceci est illustré par la relation R (A, B, C) de clé primaire (A, B) et dont les domaines vérifient

$$R.(A, B) \longrightarrow R.C$$



Donc, B est transitivement dépendant de la clé primaire (A. B). Nous ne pouvons éliminer la dépendance transitive de B sans casser la clé primaire.

<u>Définition</u> : Une relation est en troisième forme normale si :

- elle est en seconde forme normale.
- chaque domaine non primaire de R est non transitivement dépendant de chaque clé candidate de R.

C'est-à-dire que chaque domaine non primaire de R est totalement et non transitivement dépendant de chaque clé candidate de R.

Nous nous basons sur un second exemple, étant donné que l'exemple précédent se trouve déjà sous la troisième forme normale.

Nous considérons la relation E (employés) possédant les domaines simples suivants :

NE	numéro de l'employé.
CODE	code du travail.
ND	numéro de département.
NDR	nom du directeur du département.
TC	type du contrat.

Une des réalisations de E est par exemple

(voir feuille suivante)

Clé primaire

(NE)	CODE	ND	NDR	TC
1	a	1	DUPONT	g
2	c	1	DUPONT	g
3	a	2	DURANT	n
4	b	1	DUPONT	g
5	b	2	DURANT	n
6	c	2	DURANT	n
7	a	3	MAX	n
8	c	3	MAX	n

Nous pouvons mettre en évidence une dépendance transitive entre certains domaines de la relation E. En effet, TC ainsi que NDR dépendent de ND qui lui dépend de NE

$$E.NE \longrightarrow E.ND \longrightarrow E\ (NDR,\ TC)$$

<u>Conversion</u> : Nous réduisons en la troisième forme normale en explosant E en deux relations E 3 et D 3 de clé primaire NE et ND.

E 3 — Clé primaire

(NE)	CODE	ND
1	a	1
2	c	1
3	a	2
4	b	1
5	b	2
6	c	2
7	a	3
8	c	3

D3 — Clé primaire

(ND)	NDR	TC
1	DUPONT	g
2	DURANT	n
3	MAX	n

<u>REMARQUE</u> : Tout comme pour la seconde forme normale, la troisième forme normale possède une forme optimale : la base de

données optimale est celle qui possède le moins de relations.

CONSEQUENCES : Quelles peuvent être les conséquences de la normalisation des relations du modèle relationnel ?

1) N'importe quelle relation mise sous la première forme normale peut se représenter sous la forme d'une table lors du stockage des données.
2) Le nombre de suppressions, d'insertions et de mises à jour est réduit.
3) Le besoin de restructurer la collection de relations lorsque de nouveaux types de données sont introduits est moindre, ce qui entraîne une augmentation de la vie des programmes d'application.
4) Plus la normalisation est poussée et plus la base de données devient compréhensive et informative pour les utilisateurs occasionnels.
5) Le fait de normaliser les relations n'a aucune conséquence sur le potentiel d'informations de la base de données. En effet, il existe un procédé opposé à la normalisation, qui est la dénormalisation des relations. Il consiste à construire une structure de données non normalisée, à partir de sa représentation normalisée. Afin de procéder à la dénormalisation, certaines opérations de restructuration devront être effectuées. Toutefois, la réversibilité de ces deux procédés nous amène à conclure que la base de données ne subit aucune perte d'information.
6) Les diverses étapes de la normalisation nécessitent néanmoins la création de nouveaux noms de relations.

1.4. OPERATIONS SUR RELATIONS NORMALISEES.

La sélection de données hors de la base de données peut se

traduire par la combinaison de relations existantes, afin de former une nouvelle relation normalisée.

Nous allons énoncer quelques unes des diverses opérations qui peuvent être effectuées sur les relations normalisées afin d'obtenir les combinaisons nécessaires à la sélection des données.

1.4.1. LA PROJECTION.

Soit une relation R de degré m, et un uplet R de cette relation.

I 1)- Pour $j = 1,2,3, \ldots m$ $r(j)$ désigne le jième composant de r.

2)- Si $A = (J_1, j_2, \ldots j_k)$ est un sous-ensemble de $(1,2,\ldots m)$, $r(A) = (r(j_1), r(j_2) \ldots\ldots r(j_k))$.

3)- Si $A = \emptyset$, $r(A) = r$.

II 1)- $R(A) = \{r(A) \mid r \in R$

R (A) est appelé la projection de la relation R par rapport aux domaines $(j_1, j_2, \ldots j_k)$.

2)- Si A est une permutation de la liste $(1, 2, \ldots m)$, R (A) est une relation possédant les mêmes domaines que R mais dont l'ordre d'apparition a changé.

1.4.2. LA JONCTION.

Si θ est un des opérateurs : $=, \neq, <, \leqslant, >, \geqslant$ et si S et R sont deux relations, le θ - jonction de la relation R sur le domaine A, avec la relation S sur le domaine B est définie comme suit :

$$R\,[A\,\theta\,B]\,S = \{(r, s) \mid r \in R \text{ et } s \in S \text{ et } (r[A]\,\theta\,s[B])\}$$

N.B. : Un cas particulier de la θ jonction est l'équijonction. C'est le cas où θ vaut '='.

1.4.3. LA DIVISION.

1) Soit T une relation binaire. Nous définissons l'ensemble des images de x sous T comme étant

$$g_T \ (x) = \{y \mid (x, y) \ \varepsilon T\}$$

2) Soit R une relation de degré m et S une relation de degré n. Soit A un sous-ensemble de (1, 2, ... m) sans répétition et $\bar{A}$ son complémentaire, trié dans l'ordre croissant. Nous traitons la relation R comme une relation binaire définie sur les domaines ($\bar{A}$, A). Nous définissons comme suit la division de (R sur A) par(S sur B).

$$R\,[A \div B]\,S = \{r\,[\bar{A}] \mid r \varepsilon R \ \text{ et } s\,[B] \ \varepsilon g_R\,(r\,[\bar{A}])\}$$

3) Si $R = \emptyset$, $R \div S = \emptyset$

Si $R = \emptyset$ et $S = \emptyset$, $R \div S = \emptyset$

1.4.4. LA RESTRICTION.

Soit R une relation de degré m.

Soient A, B des sous-ensembles de (1, 2, ... m).

Soit θ, un des opérateurs : $=$, $\neq$, $<$, $\leqslant$, $>$, $\geqslant$.

Nous appelons θ - restriction de R sur A, B :

$$R\,[A\ \theta\ B] = \{r \mid r \varepsilon R \ \text{ et } \ r\,[A]\ \theta\ r\,[B]\}$$

REMARQUE.

Il est à remarquer que d'une part, la projection représente la contrepartie algébrique du quantificateur existentiel et que d'autre part, la division représente celle du quantificateur universel.

L'ensemble des opérations qui peuvent être définies sur les relations sera appelé un algèbre relationnel. Par la suite, un langage d'interrogation pourra directement être développé à partir de ces opérations (langage de niveau intermédiaire).

Afin de prouver que la force de sélection de données au moyen de l'algèbre relationnel est aussi complète que celle du langage ALPHA (langage de haut niveau), un algorithme de transformation d'une ALPHA-expression en une séquence d'opérations sémantiquement équivalente, a pu être développé.

CHAPITRE II.

GAMMA -O CONCEPTS.

2.1. INTRODUCTION.

Dans la première partie de ce travail, nous venons de rappeler les concepts du modèle relationnel de CODD. Nous allons nous attarder à l'étude de l'implémentation logique de ce modèle de données. Pour ce faire, nous présentons les spécifications d'un interface appelé GAMMA - O.

GAMMA - O fait partie de la série GAMMA des systèmes d'interfaces de bases de données relationnelles.

Cette série représente en fait un SGBD complet, conçu et structuré de façon modulaire.

Grâce à cette approche, l'implémentation d'une donnée se réalise par étapes successives, passant par des niveaux de plus en plus restrictifs quant aux conditions d'implémentation et de plus en plus précis quant à la nature du problème représenté.

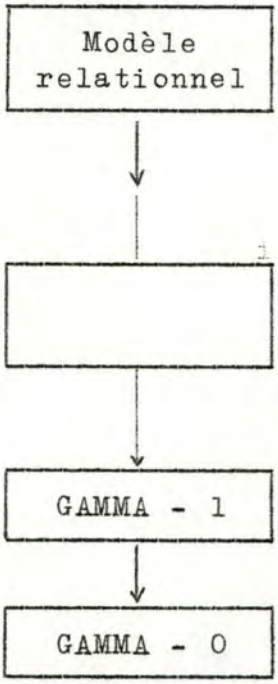
GAMMA - O est en fait l'interface constituant le plus bas niveau et implémentant directement la donnée. Il est un interface gérant les relations de la base au moyen de toute une série de primitives.

Etant donné le fait que GAMMA - O est du plus bas niveau, il se trouve limité au trafic d'un seul utilisateur. Nous pensons que par la suite, le SGBD GAMMA se complètera par la mise au point de GAMMA - 1 conçu notamment pour la gestion de protections des données contre des utilisateurs multiples.

Les niveaux supérieurs du système GAMMA pourraient par exemple offrir des possibilités telles que optimisation de la recherche, sécurité, évitement des interblocages...Les conceptions et la mise au point du SGBD GAMMA se simplifie, de par son caractère modulaire. De plus, étant donné un niveau d'implémentation, ce niveau utilise les objets et primitives définis aux niveaux inférieurs. Les fonctions exécutées par un des niveaux se traduisent en

terme de contraintes et se répercutent dans les niveaux inférieurs jusque GAMMA - O qui doit en tenir compte lors de son implémentation.

Schématiquement, l'enchaînement des divers systèmes peut se représenter sous la forme suivante :



Dans les paragraphes qui suivent, nous exposons le contenu de l'interface GAMMA - O.

2.2. GAMMA - O :

L'implémentation du modèle relationnel se construit sur 4 types de relations. Toutes ces relations possèdent la même structure physique mais diffèrent quant à leurs sémantiques et leurs fonctions au sein du système de gestion de la base de données.

2.2.1. LA RELATION MAITRE.

Cette relation est unique. Elle est de degré 7 et est utilisée pour stocker la description de toutes les relations définies dans la mémoire.
Elle comprend donc une auto-description. La relation MAITRE sert en fait de catalogue des relations de la base de données. La relation maître est créée à l'initialisation de GAMMA - O. Sa cardinalité évolue dynamiquement selon les créations et suppressions des autres relations de la base de données.

- Lors de la création d'une nouvelle relation, un uplet descriptif sera ajouté à la relation MAITRE.
- Lors de la suppression d'une relation, le uplet correspondant à cette dernière sera retranché de la relation MAITRE.

2.2.2. LES RELATIONS REGULIERES.

Le degré des relations régulières peut varier de 1 à 2^{32}-1. Leur cardinalité est quelconque, mais au moins égale à 1. Chaque uplet s'identifie d'une façon unique dans la relation grâce à la valeur de sa clé primaire. Chaque relation s'identifie de façon unique dans la base de données, grâce à un identificateur. Les relations régulières sont utilisées pour stocker d'une manière compacte toutes les informations de la base de données. Les domaines de telles relations sont tous de longueur fixe.

2.2.3. LES RELATIONS CLASSES.

Le degré de ces relations est toujours égal à 1. La différence essentielle avac les relations régulières réside dans le fait que les occurrences de cet unique domaine peuvent

avoir des longueurs variables. La clé primaire d'une relation classe est de ce fait toujours le premier domaine .

Exemple :

B E R L I N	longueur	= 6
M O N S		= 4
B R U X E L L E S		= 9

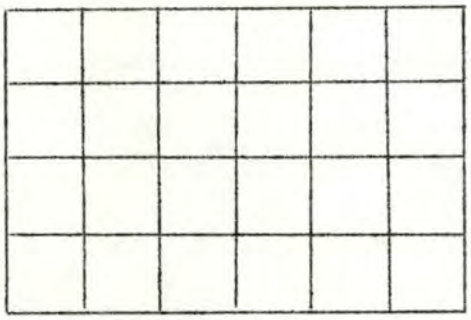
Chaque relation classe s'identifie de façon unique dans la base de données grâce à un identificateur.

2.2.4. LES RELATIONS INVERSIONS.

Le degré de ces relations est toujours égal à 2. La clé primaire d'une inversion sera toujours le deuxième domaine. Les différents domaines sont tous de longueur fixe.

RESUME.

1)- Jusqu'à présent, nous sommes restés au niveau du modèle relationnel de E.F. CODD.
En effet, en résumé, nous disposons de relations normalisées, puisqu'elles sont toutes représentables sous forme d'un tableau à deux dimensions.



N = degré

c = cardinalité

l = longueur du domaine.

Nous avons distingué quatre espèces de relations :

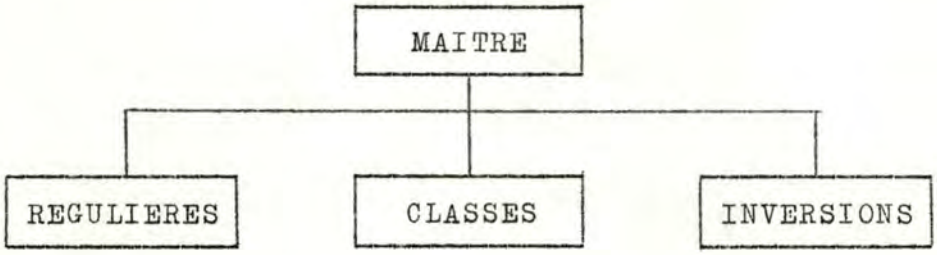
1°) N = 7 l fixe c $\geqslant$ 1

2°) $1 \leqslant N \leqslant 2^{32} - 1$ l fixe $c \geqslant 1$

3°) $N = 1$ l variable $c \geqslant 1$

4°) $N = 2$ l fixe $c \geqslant 1$

2)- De plus, nous pouvons également dire que la relation MAITRE est de nature sémantique différente du reste des relations. Elle peut être considérée comme la racine d'un arbre dont les feuilles sont les trois autres types de relations.



2.3. IDENTIFICATEURS.

Possédant un ensemble de relations décrites comme ci-avant, il est nécessaire, afin de gérer au mieux la base de données, de pouvoir les retrouver d'une façon unique, ainsi que leurs différents uplets.

2.3.1. IDENTIFICATEURS DE RELATIONS.

Lors de la création d'une relation, le système lui associe un identificateur. Nous distinguons quatre types d'identificateurs de relation :

- I D M R : Il est unique, c'est l'identificateur de la relation MAITRE.
- I D R R : Identificateurs de relations régulières.
- I D R I : Identificateurs d'inversions.
- I D R C : Identificateurs de classes.

2.3.2. IDENTIFICATEURS DE UPLETS.

Lorsqu'une relation a été sélectionnée, ses différents uplets doivent être identifiés de façon unique. A chaque création d'un uplet dans une relation, le système associe un I D U (identificateur de uplet) à ce uplet. Dans une même relation, chaque valeur de I D U est unique. En fait, les I D U sont synonymes de clés primaires, mais ils sont beaucoup plus faciles à manipuler par le système, puisqu'ils possèdent tous la même structure.

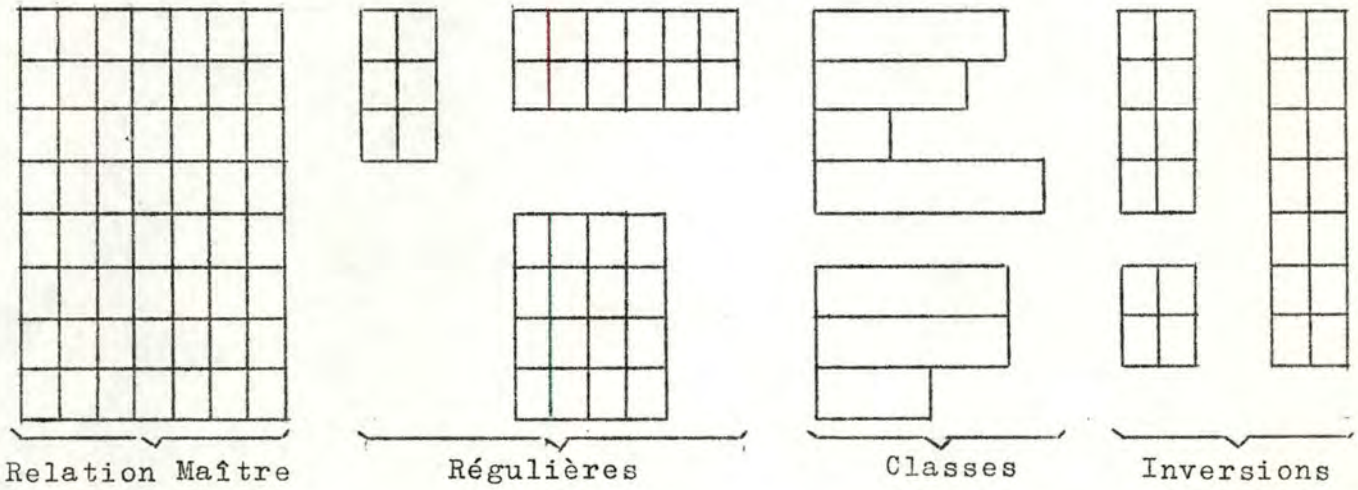
2.3.3. IDENTIFICATEURS DE DOMAINES.

Dans une relation, les domaines sont identifiés par un nombre entier, selon leur apparition dans la relation.

2.4. RELATIONS ET UPLETS.

Nous nous attardons maintenant à décrire le contenú sémantique et le rôle des différentes relations qui apparaissent dans la base de données.

Nous savons déjà qu'une relation de GAMMA - O peut se représenter sous la forme d'un tableau :



Lors de la création d'une relation, son garnissage se fait uplet par uplet. De plus, un uplet de contrôle lui est également associé lors de sa création. Cet uplet de contrôle comporte autant de domaines que la relation à laquelle il est associé. Son but est de gérer les valeurs sémantiques des domaines de la relation. En effet, les valeurs apparaissant dans un domaine,

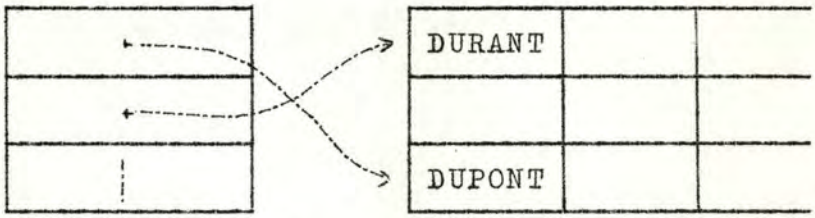
- soit se dénotent elles-mêmes.

  Exemple :

D U P O N T
D U R A N T
⋮

- soit constituent un pointeur vers un autre uplet.

  Exemple :



Cette différence dans la nature des domaines se traduit dans l'uplet de contrôle, de la façon suivante :

- Lorsque les valeurs d'un domaine i se dénotent elles-mêmes, l'uplet de contrôle (i) = 0.
- Lorsque les valeurs d'un domaine j représentent des pointeurs, l'uplet de contrôle (j) = identificateur d'une relation régulière ou d'une classe. Les pointeurs sont évidemment des identificateurs de uplets appartenant à la relation désignée par le uplet de contrôle.

Cas particuliers : le uplet de contrôle est sans utilité dans deux cas : - 1) La relation maître a le uplet de contrôle (0, 0, 0, 0, 0, 0,0).

- 2) Les classes ont le uplet de contrôle (0). Toutefois, ils sont créés, afin d'assurer une parfaite homogénéité dans la structure des relations de la base de données. Nous illustrons par un exemple l'emploi du uplet de contrôle.

Exemple :

Relation régulière n° 1

10	0	0	6	0	9
11	DURANT	3	61	5002	91
12	DUPONT	7	62	5000	92
13	VARON	26	63	7000	93

Classe n° 6

60	0
61	G R A N D E
62	D E B R U X E L L E S
63	D U B O I S

Classe n° 9

90	0
91	St S E R V A I S
92	N A M U R
93	M O N S

Nous avons convenu que le numéro de la relation est son identificateur et que les identificateurs de uplets sont dérivés de celui de la relation.

Nous remarquons dans cet exemple que l'utilisation du uplet de contrôle nous permet en quelque sorte d'avoir des domaines de longueur variable dans une relation régulière.

2.4.1. LA RELATION MAITRE.

Cette relation a déjà été définie comme étant la relation

utilisée pour stocker les caractéristiques logiques des relations existant dans le système.

Chacun de ses uplets à la structure suivante :

Type de la relation	Degré de la relation	Masque de la clé primaire	Identificateur du uplet de contrôle	Identificateur de la relation inversée	Identificateur du domaine inversé	Identificateur de la relation décrite
1	2	3	4	5	6	7

Description du contenu des domaines.

DOMAINE 1 : Il contient le type de la relation décrite. Nous savons que le type d'une relation peut être :

- MAITRE
- REGULIERE
- CLASSE
- INVERSION

Le système contrôle qu'il n'y a qu'une relation de type MAITRE.

DOMAINE 2 : Il spécifie le degré de la relation :

- si le type est MAITRE, degré = 7.
- si le type est REGULIER, $1 \leq$ degré $\leq 2^{32} - 1$.
- si le type est CLASSE, degré = 1.
- si le type est INVERSION, degré = 2.

DOMAINE 3 : Il contient le masque de la clé primaire. C'est un masque de 32 bits. La clé primaire de la relation est composée de la concaténation des domaines correspondant aux '1' du masque.

- type MAITRE : le masque possède un '1' en position 7, le reste étant nul.
- type REGULIER : n'importe quelle combinaison est permise, excepté le masque entièrement nul.

Une relation régulière ne peut donc posséder une clé primaire de plus de 32 domaines.

- type CLASSE : un '1' apparaît dans la première position, les autres bits restant nuls.
- type INVERSION : un '1' apparaît dans la seconde position et les autres bits restent nuls.

DOMAINE 4 : Ce domaine contient l'identificateur du uplet de contrôle de la relation décrite.

DOMAINE 5 : Dans le cas où le type est MAITRE, REGULIER ou CLASSE, ce domaine est toujours nul.
Dans le cas où le type est INVERSION, il est nécessaire de donner l'identificateur de la relation inversée par celle décrite dans le uplet de la relation maître (voir définition des inversions plus après).

DOMAINE 6 : Il contient l'identificateur du domaine qui a subi l'inversion. Comme dans le cas ci-avant, lorsque les relations sont du type MAITRE, REGULIER OU CLASSE, ce domaine est nul.
Dans le cas où le type est INVERSION, le domaine 6 contient l'identificateur du domaine qui a été inversé. Ce domaine appartient à la relation dont l'identificateur a été spécifié dans le domaine 5.

DOMAINE 7 : Il contient l'identificateur de la relation décrite.

Nous terminons l'exposé de ce paragraphe par un exemple :

(voir page suivante)

10	0	0	0	0	0	0	0
11	MASTER	7	000000100	10	0	0	1
12	REGULIER	3	0111000..	32	0	0	3
13	CLASSE	1	1000.....	61	0	0	6
14	CLASSE	1	1000.....	73	0	0	7

Auto-description

2.4.2. LES RELATIONS REGULIERES.

Il est bon de souligner que les relations régulières sont la plupart du temps utilisées pour stocker d'une manière compacte des informations qui caractérisent le même objet.

Nous prenons comme exemple une relation régulière regroupant des informations concernant des employés. Cette relation comporte 8 domaines :

- domaine 1 : n° de l'employé.
- domaine 2 : nom de l'employé.
- domaine 3 : prénom de l'employé.
- domaine 4 : ville de l'employé.
- domaine 5 : salaire.
- domaine 6 : code du travail.
- domaine 7 : code du département.
- domaine 8 : nom du directeur du département.

Le uplet de contrôle de cette relation est rempli de la façon suivante :

20	0	3	4	5	0	6	7	2

Les numéros 3, 4, 5, 6, 7 sont les identificateurs des classes contenant respectivement les noms, les prénoms, les

villes, les codes de travail et les codes de département des employés. Le numéro 2 est l'identificateur de la relation contenant les renseignements relatifs aux employés, c'est-à-dire la relation régulière elle-même. Cela signifie que le dernier domaine de la relation régulière pointe vers des uplets de la relation elle-même.

REMARQUE : Cette façon de représenter les informations nous suggère qu'à la limite, nous pourrions considérer les relations régulières comme des tableaux de pointeurs. Cette technique serait analogue à celle de la gestion des 'SETS' chez CODASYL, au moyen des "DYNAMIC POINTER ARRAY".

2.4.3. LES CLASSES.

Les classes ont été définies précédemment, mais il est nécessaire d'insister sur le fait que les valeurs contenues dans une classe se dénotent toujours elles-mêmes. En effet, l'utilité des classes réside dans le fait de pouvoir utiliser des domaines de longueurs variables. C'est pourquoi le uplet de contrôle est toujours '0'.

2.4.4. LES INVERSIONS.

Les différentes relations de la base des données, une fois créées, sont destinées à être exploitées (mises à jour, supprimées, lues). Des opérations de recherche seront notamment établies afin de retrouver les données nécessaires au sein des relations désirées. Une opération de recherche pourra tirer un énorme avantage du fait que les uplets de la relation auront été soumis à un tri. Afin de permettre d'une manière efficiente, la recherche par rapport à n'importe quel sous-ensemble de domaines, il est nécessaire de stocker des projections de la relation dans ce que nous appelons des inversions.

Une inversion est donc la projection d'une relation (MAITRE, régulière ou classe) par rapport à un et à un seul domaine, trié en majeur sur ce domaine et en mineur sur l'identification des uplets inversés.

Structure des uplets d'une inversion.

Comme nous l'avons précisé ci-avant, une inversion est toujours binaire. Le premier de ses domaines contient le domaine qu'elle inverse. Son second domaine donne les identificateurs des uplets correspondants.

Exemple :

Relation régulière n° 2

10	0	0	0	0	0
11	203	12	DUPONT	5	7
12	203	73	DURANT	7	86
13	206	12	DUPONT	7	33
14	206	29	MAX	25	89
15	206	36	ZENON	16	111

Inversion n° 3

30	0	1
31	DUPONT	11
32	DUPONT	13
33	DURANT	12
34	MAX	14
35	ZENON	15
	majeur	mineur

La relation inversion n° 3 est une inversion du domaine n° 3 de la relation régulière n° 1. La clé de tri est le nom du directeur de l'unité de production.

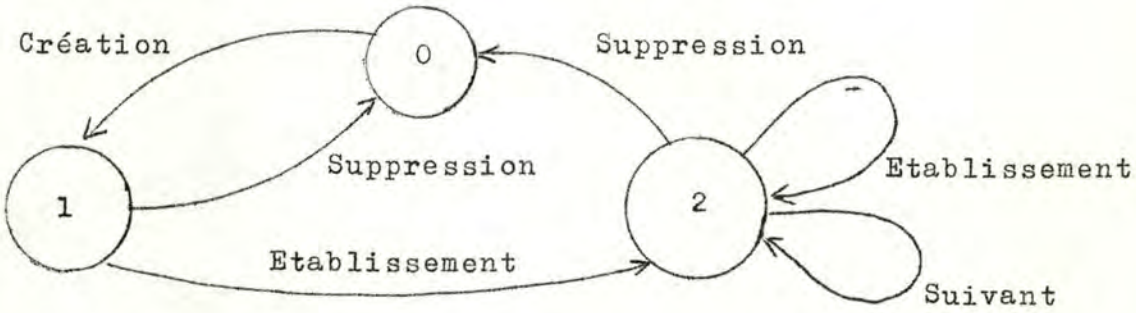
2.5. LES SCANS.

Un scan est une opération permettant l'extraction de données de la base. Il ne peut être appliqué qu'à une seule relation. Le scan soumis à certaines contraintes, soit en provenance du système, soit en provenance de l'utilisateur, balaie la relation concernée afin

de trouver le premier uplet vérifiant ces contraintes. Le uplet sélectionné est ensuite renvoyé à l'utilisateur.

2.5.1. SCANS INITIALISES PAR LE SYSTEME.

Le diagramme d'état de ce type de scan peut schématiquement être représenté par un automate fini à 3 états :



La commande CREATION a pour but de créer le scan. La commande ETABLISSEMENT a pour but d'établir les contraintes auxquelles le scan devra être soumis. La commande SUIVANT est analogue à un "GET". La commande SUPPRESSION supprime le scan.

Exemple :

CREATION

ETABLISSEMENT . . . . . .établissement des contraintes.

SUIVANT
SUIVANT . . . . . lecture de 3 uplets vérifiant
SUIVANT .les premières conditions.

ETABLISSEMENT . . . . . .établissement de nouvelles contraintes.

SUIVANT
SUIVANT . . . . . lecture de 2 uplets vérifiant .d'autres conditions.

SUPPRESSION

La commande SUIVANT est analogue à un "GET" c'est-à-dire qu'elle renvoie un uplet vérifiant les contraintes établies par une commande ETABLISSEMENT. Lorsque nous avons une séquence de commande SUIVANT, le système fournit une séquence de uplets issus de la relation.

Ce type de scan sera notamment utilisé :

- Si nous voulons connaître l'identification d'un uplet dans une relation connue.
- Si nous voulons lire tous les uplets d'une relation donnée.

REMARQUE : Etant donné qu'à chaque scan, nous pouvons associer un ensemble de contraintes, il existe une fonction entre l'ensemble des scans et l'ensemble des relations. Cette possibilité sera exploitée lors de l'implémentation physique des scans.

2.5.2. SCANS INITIALISES PAR L'UTILISATEUR.

La différence avec le cas précédent, est que le système ne renvoie plus un uplet et son identificateur, mais renvoie uniquement l'identificateur.
La séquence CREATION - ETABLISSEMENT - SUIVANT - SUPPRESSION doit être générée par l'utilisateur.

Ce scan sera utilisé lorsque l'utilisateur désire rechercher dans une relation, l'identificateur d'un uplet dont on connaît déjà un sous-uplet. L'uplet sélectionné par le scan sera le premier uplet de la relation, qui contient le sous-uplet spécifié. Cette démarche est celle de la recherche filtrée (voir plus après), le filtre étant la valeur du sous-uplet.

L'identificateur étant connu, l'utilisateur peut obtenir le uplet qui lui correspond. Ensuite, soit en gardant la même valeur de filtre, soit en la modifiant, l'utilisateur peut poursuivre sa recherche.

N.B.- Dans la suite de ce travail, "scan" signifiera "scan sous le contrôle du système".

2.5.3. RELATION D'ETAT DES SCANS D'UNE RELATION.

A un instant donné, à une relation de la base de données,

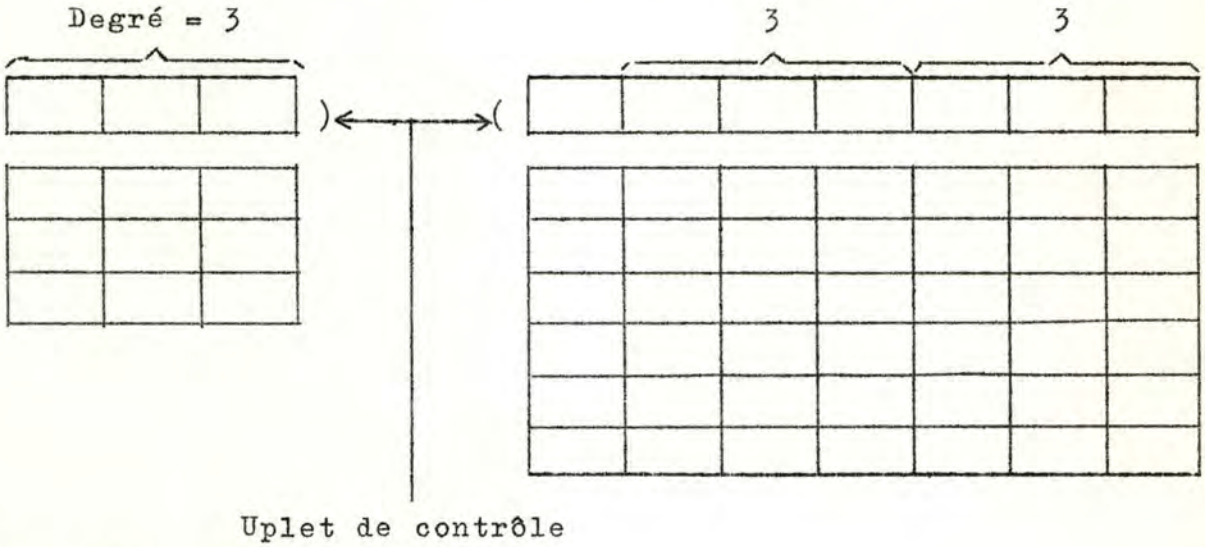
peuvent être associés n scans qui sont soit dans leur état 1, soit dans leur état 2 (voir diagramme d'états). A chaque scan de cette relation, le système associe un identificateur.

Lors de la création d'une relation, une relation de travail lui sera associée, "sa relation d'état des scans" (RES). La RES d'une relation mémorise l'état de tous les scans associés à cette relation.

Caractéristiques.

- Si la relation possède le degré N, sa RES est de degré (2 * N + 1).
- Chaque scan associé à la relation constitue un uplet de la RES de cette relation. Les identificateurs des différents scans sont les identificateurs des uplets de la RES.
- La RES possède comme toute autre relation, un uplet de contrôle.

Exemple : La relation est de degré 3 et sa RES est de degré 7. Il y a 6 scans associés à cette relation.



Signification des domaines.

Domaine 1 : contient l'identificateur du uplet qui fera

l'objet du scan.

Domaine (n + 2) jusqu'à domaine (2 * n + 1) : Ces n domaines forment un masque qui donne :

- les domaines devant participer à la recherche.
- les valeurs de filtre correspondantes.

Domaine 2 jusqu'à domaine (n + 1) : Masque des domaines à fournir à l'utilisateur.

Signification des domaines du uplet de contrôle.

Domaine 1 : contient l'identificateur de la relation "scannable".

Domaine 2 jusqu'à domaine (2 * n + 1) : seront mis à '0'.

Exemple =

50	23	0	0	0	0	0	0
51	234	1	2	*	60	60	60
52	231	1	2	3	50	*	*
53	237	*	2	*	10	*	*
54	235	*	*	3	*	15	12
55	234	*	2	3	*	40	50

Id. de scans — Id. de uplets — Domaines à retenir — Valeurs des filtres

Le scan recherche dans la relation "employés" le nom des personnes (domaine 2 de la relation 23) appartenant au département 10 (domaine 1).

REMARQUES :

1)- La sémantique du premier domaine de la RES est à rapprocher de la notion de CURRENT chez CODASYL. En effet, le CURRENT fournit l'identificateur du dernier

RECORD qui a été accédé. Tandis que le premier domaine de la RES fournit les identificateurs des uplets, à partir desquels vont débuter les différents scans.

2)- La RES est abusivement appelée relation. En effet, elle ne peut rentrer dans aucune des 4 catégories de relations définies dans GAMMA - O. Il n'y a en outre aucune interférence entre les différents domaines de la RES, il n'est donc pas possible de la normaliser. De plus, elle ne possède pas de clé primaire, et elle n'est pas décrite dans le catalogue des relations. Elle peut posséder plusieurs uplets identiques qui, cependant seront identifiés différemment.

3)- Pour plus de clarté, nous avons préféré nous limiter ici à une présentation sommaire. Nous décrivons le principe de la recherche filtrée et le mécanisme complet du scan dans le paragraphe consacré aux opérations dans GAMMA - O.

2.6. OPERATIONS DANS GAMMA - O.

2.6.1. INITIALISATION DE LA BASE DE DONNEES.

La mise en place d'une base de données relationnelle nécessite l'établissement d'un support ou noyau central. Dans GAMMA - O, cette opération consiste en la création de la relation maître. Il est évident que la cardinalité de la relation maître ne peut à ce moment qu'être égale à '1'. Les seuls uplets ne sont en effet que le uplet de contrôle (n'intervient pas dans la cardinalité) et le uplet d'auto-description. En effet, la relation-maître donne la description de relations n'existant pas encore dans la base de données.

La création de la relation maître est irréversible. Ses uplets de contrôle et d'auto-description ne peuvent être sup-

primés sans la destruction de la base de données.

A chaque création d'une nouvelle relation dans la base de données, un nouveau uplet est ajouté à la relation-maître. A chaque suppression, il est retranché.

Etat de la base de données : A l'initialisation, la base contient :

Relation n° 1 :

10	0	0	0	0	0	0	0
11	Maître	7	0000001	10	0	0	1

Séquence des opérations : voir figure n° 1.

2.6.2. COMMANDES DE GAMMA - O.

Les uplets des relations sont formattés par des opérations du langage hôte et insérés dans les relations grâce à l'utilisation des commandes de GAMMA - O. L'utilisateur n'applique jamais le langage hôte sur la base de données.

Si l'utilisateur veut par exemple insérer un uplet dans une relation de degré n, au moyen du langage hôte, il doit tout d'abord former un vecteur n-aire qui deviendra uplet lorsqu'il sera pris en charge par une des commandes de GAMMA-O.

1.- Types des commandes :

L'ensemble des commandes de GAMMA - O peut être divisé en 3 types :
- commandes qui créent et suppriment des relations.
- commandes de mise à jour, de lecture, d'insertion, ...
- commandes qui se rapportent aux inversions.

2.- Format des commandes.

Toutes les commandes sont mises sous la forme d'une pro-

cédure qui doit être appelée par un CALL.

Chaque appel de procédure :

- accepte une liste de paramètres d'entrée.
- effectue certaines opérations.
- renvoie à l'utilisateur soit une liste de valeurs de sortie, soit un message d'erreur.

3.- Format syntaxique général.

OP <paramètres d'entrée> <paramètres de sortie> message d'erreur.

4.- Liste des commandes.

	Nom	Paramètres d'entrées	Paramètres de sorties
Type 1.	1. create relation	<d-tuple, c-tuple>	<Rid, c tid>
	2. drop relation	<rid>	

	Nom	Entrées	sorties
Type 2.	3. Insert tuple	<rid, tid 1, tuple>	<tid 2>
	4. Delete tuple	<rid, tid>	-
	5. Move tuple	<rid, tid 1, tid 2>	-
	6. Update subtuple	<rid, tid, dlist,> subtuple	-
	7. Tuple id from next subtuple	<rid, tid,dlist,subtuple>	<tid 2>
	8. Subtuple from id	<rid, tid dlist>	<subtuple>
	9. Create scan	<rid, dlist 1, dlist 2>	<rid, ctid>
	10. Set scan	<rid, tid, subtuple>	-
	11. Next subtuple	<rid>	<tid,subtuple>
	12. Drop scan	<rid>	-

	Nom	Entrées	Sorties
Type	13. Inversion	<rid 1, did>	<rid 2, ctid>
	14. Drop inversion	<rid>	

Chaque fois qu'une commande est utilisée, le système effectue un contrôle syntaxique sur le format de cette commande.

2.6.2.1. CREATE RELATION.

Cette commande est utilisée pour créer soit une relation régulière, soit une relation classe.

CR < d - tuple, c - tuple > < rid, ctid > < cc >

L'utilisateur initialise la création d'une relation, en fournissant d - tuple et c - tuple comme paramètre d'entrée. Ce paramètre formera le uplet de description qui sera ajouté à la relation maître.
C - tuple est le uplet de contrôle de la relation que l'on désire créer.

Entrées : d - tuple.

type	degré	masque

c - tuple a le même degré que la relation. Une fois introduit dans la base de données, il ne pourra plus être mis à jour.

0	0	26 .....

Sorties :

rid
ctid) Une fois la création terminée, le système renvoie à l'utilisateur, les identificateurs qu'il vient d'attribuer à la relation (rid) et à son uplet de contrôle (ctid).
Le système laisse le soin aux utilisateurs de mémoriser leurs indentificateurs (au moyen du langage hôte) afin de pouvoir s'en servir par la suite.

Opérations effectuées : voir figure n° 2.

Le but de cette commande est donc d'établir une relation vide. Elle peut être garnie de uplets grâce à la commande IT (que nous détaillons plus loin). Elle ne peut être identifiée que par l'identificateur qui a été renvoyé à l'utilisateur.

2.6.2.2. DROP RELATION.

Cette commande est utilisée pour supprimer soit une relation régulière, soit une relation classe.

DR <rid> <message d'erreur>

rid : indentificateur de la relation à supprimer

Il est évident que la suppression d'une relation entraîne la suppression de toutes ses inversions. Si aucune relation n'est identifiée par rid, un message d'erreur est envoyé à l'utilisateur.

Opérations effectuées : voir figure n° 3.

2.6.2.3. INSERT TUPLE.

Elle est utilisée pour garnir de uplets une relation régulière ou une relation classe.

IT <rid, tid 1, tuple> <tid 2>

Entrée : rid : identificateur de la relation où doit être inséré le uplet.

tid 1 : identificateur du uplet après lequel il faut insérer.

tuple : vecteur représentant le uplet à insérer. Ce vecteur a le même degré que la relation.

Opérations effectuées : voir figure n° 4.

Sortie : rid 2 : identificateur donné par le système au uplet qui vient d'être inséré.

REMARQUE : Dans une même relation, il existe une bijection entre l'ensemble de ses uplets et l'ensemble des clés primaires de ses uplets. C'est pourquoi, dès que nous voulons insérer un uplet dont la valeur de la clé primaire existe déjà, ce uplet est redondant. Dans ce cas, le système renvoie à l'utilisateur l'identificateur du uplet qui se trouve déjà dans la base de données.

2.6.2.4. DELETE TUPLE.

DT <rid, tid> message d'erreur

Entrées : rid : identificateur de la relation où doit être supprimé un uplet.

Opérations effectuées : voir figure n° 5.

UPDATE SUBTUPLE.

Permet dans les relations régulières de mettre à jour des domaines non primaires d'un même uplet. La clé primaire ne peut être mise à jour, elle est un invariant dans le uplet.

US <rrid, tid, dlist, subtuple>

Entrée : rrid : Identification de la relation régulière où la mise à jour doit s'opérer.

tid : Identificateur du uplet qui doit être mis à jour.

dlist: masque indiquant au système quels sont les domaines intéressés dans la mise à jour. Le degré de dlist doit toujours être inférieur au degré du uplet à mettre à jour.

subtuple : mise à jour du sous-uplet. Même degré que dlist.

Opérations effectuées : voir figure n° 6.

2.6.2.5. CREATE SCAN. (sous le contrôle du système).

Nous savons que lorsqu'une relation est créée, le système lui associe une RES (relation d'état des scans) vide. Cette commande a pour but d'insérer des uplets dans la RES.

CS <Rid, dlist 1, dlist 2> <Sid, ctid>

Entrées : rid : identificateur de la relation à laquelle se rapporte le scan.

dlist 1 : indique au système les domaines qui intéressent l'utilisateur. dlist 1 possède le même degré que la relation.

dlist 2 : son but est de donner un masque des domaines sur lesquels sera appliquée une recherche filtrée. dlist 2 possède également le même degré que la relation.

Sorties : sid : identificateur du scan qui vient d'être créé.

ctid : identificateur du uplet de contrôle de la relation à laquelle se rapporte le scan.

Opérations effectuées : voir figure n° 7.

2.6.2.6. SET SCAN. Sc <sid, tid, subtuple>

Entrées : sid : Identificateur du scan qui doit être utilisé. Cet identificateur aura au préalable été fourni par une commande CS.

tid : Identificateur d'un uplet de la relation à scanner. En particulier, il peut être le uplet de contrôle. La recherche commencera à partir du uplet qui suit séquentiellement le uplet identifié par tid. Si tid = uplet de contrôle, la recherche

débute au premier uplet effectif de la relation.

subtuple : Le degré de ce sous-uplet doit être le même que celui de la dlist 2 de la commande CS associée. Ce sous-uplet est un vecteur contenant les valeurs effectives des filtres qui vont être la base de la recherche filtrée. - Le principe de la recherche filtrée est détaillé au paragraphe consacré à la recherche NS.

Opérations effectuées : voir figure n° 8.

2.6.2.7. NEXT SCAN.

Principe de la recherche filtrée : nous savons que les valeurs de filtres se trouvent dans les domaines (n + 2) à (2 n + 1) du scan. Nous savons aussi :

- que la liste des domaines que l'utilisateur désire recevoir est contenue dans les domaines 2 à n + 1 du scan.
- que l'identificateur du uplet à partir duquel va démarrer la recherche est celui qui suit séquentiellement l'uplet identifié par le premier domaine du scan. La recherche filtrée se déroule selon le schéma de la figure n° 9. Elle s'arrête dans deux cas :
  1) Elle est arrivée en fin de relation.
  2) Elle a trouvé un uplet qui satisfait aux filtres.

Format du next scan : NS <Sid tid, subtuple> <cc>
Avant de pouvoir effectuer cette commande, il faut au préalable avoir effectué une commande CS et une commande SC relatives au même scan.

Entrées : Sid : identificateur du scan à prendre en con-

sidération.

Sorties : Identificateur du uplet qui a satisfait la recherche.

Subtuple : Sous-uplet de tid. Ce sous-uplet n'est défini que sur les domaines apparaissant dans le masque dlist 1 de la commande CS.

Opérations effectuées : voir figure n° 10.

Afin de rendre plus explicite et plus compréhensive cette notion de scan ; nous l'illustrons par un exemple.

1°) Création d'un scan :

Avant :

30	0	0	0	0

31	3	7706	0	C
32	12	7703	1	C
33	13	7704	9	F
34	26	7705	3	B
35	32	7708	2	C

70	3	0	0	0	0	0	0	0	0

RES associée d'identificateur 7.

Relation régulière n° 3 relative par exemple à des commandes fournisseurs. Elle comporte les domaines suivants: n° du fournisseur, échéance de la facture, type des produits commandés, type de livraison.

Commandes : CS $<3, (1,3), 2>$

où Rid = 3 dlist 1 = (1,3) dlist 2 = 2

Après : La relation régulière reste inchangée. La RES est pourvue d'un nouveau uplet :

70	3	0	0	0	0	0	0	0	0
71	0	1	*	3	*	*	0	*	*

L'utilisateur reçoit l'identificateur du uplet de contrôle de la relation (30) et l'identificateur du scan qu'il vient de créer (71).

2°) Etablissement d'un scan.

Avant : Nous partons de l'état final de la commande de création du scan.

Commande : SS < 71, 31, 7703 >

où Sid = 71, tid = 34, subtuple = 7703

Après : La relation régulière reste inchangée. Le scan 71 de la RES est mis à jour.

70	3	0	0	0	0	0	0	0	0
71	31	1	*	3	*	*	7703	*	*

3°) Recherche.

Commande : NS <71> où Sid = 71.

Après :

70	3	0	0	0	0	0	0	0	0
71	32	1	*	3	*	*	7703	*	*

Tandis que la relation régulière reste inchangée.

L'utilisateur reçoit :

- identificateur du premier uplet satisfaisant le

filtre : 32.

- valeur des domaines demandés : n° fournisseur : 12 ; type des produits commandés : 1.

2.6.2.8. DROP SCAN.

Cette commande permet de mettre fin à un scan. Le "drop scan" se situe à la fin d'une structure de ce type :

DS <Sid> <cc>

Entrée : identificateur du scan à supprimer.

Opérations effectuées : voir figure n° 11.

2.6.2.9. TUPLE ID FROM SUBTUPLE. (scan sous le contrôle de l'utilisateur).

Etant donné un sous-uplet et une relation, le système, grâce à cette commande, peut fournir à l'utilisateur, l'identificateur d'un uplet de la relation dans lequel est contenu le sous-uplet.

T I F N S <Rid, tid 1, dlist, subtuple><tid 2><cc>

Entrées : Rid : identificateur de la relation sur laquelle va porter la recherche.

tid : identificateur du uplet après lequel commence la recherche.

dlist : Masque des domaines devant intervenir dans la recherche. dlist a le même degré que la relation.

subtuple : Masque des filtres à appliquer sur les domaines contenus dans dlist. Subtuple possède le même degré que la relation.

Opérations effectuées : voir figure n° 12.

2.6.3.0. GENERATE INVERSION.

Par cette commande, l'utilisateur demande au système de

générer et de maintenir une inversion sur un domaine donné d'une relation quelconque. Nous convenons qu'il ne peut exister d'inversions.

GI <rid, dit> <isid, ctid>

Entrées : Rid : identificateur de la relation concernée. Elle est appelée "relation parente".

did : identificateur du domaine sur lequel doit porter l'inversion.

Sorties : irid : identificateur de la relation inversion. Dans le cas où le domaine était déjà inversé, aucune nouvelle création n'est nécessaire et l'identificateur de l'inversion existante est renvoyé à l'utilisateur.

ctid : identificateur du uplet de contrôle de la relation inversion.

Opérations effectuées : voir figure n° 13.

REMARQUE : A l'inverse d'une relation régulière ou d'une classe, la création d'une inversion ne crée pas une relation vide, mais bien la relation complète. Cette inversion peut par la suite être affectée par toute insertion de uplets, suppression ou mise à jour de sa relation parente. Malgré les modifications pouvant intervenir dans une inversion, ses uplets sont toujours triés en majeur sur le premier domaine et en mineur sur le 2e domaine.

2.6.3.1. DROP INVERSION.

Permet à l'utilisateur de supprimer l'inversion d'un domaine d'une relation quelconque.

DI <irid> <cc>

Entrées : irid : identificateur de l'inversion à supprimer.

Opérations effectuées : voir figure n° 14.

2.6.3.2. MOVE TUPLE.

Lors de l'insertion d'un uplet dans une relation, celui-ci est toujours placé en queue de la relation. Avec la commande "move tuple", les uplets d'une relation peuvent être réarrangés indépendamment de leur valeur ou de leur identificateur.

MV <rid, tid 1, tid 2> <cc>

Entrées : rid : identificateur de la relation.
tid 1 : identificateur du uplet à changer.
tid 2 : identificateur du uplet après lequel il faut insérer le uplet tid 1.

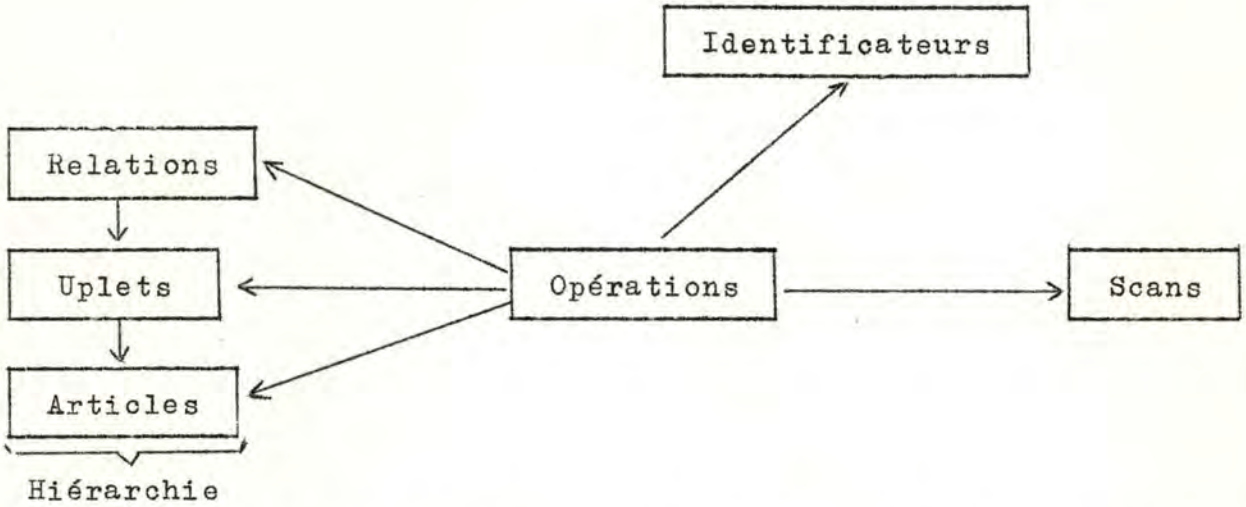
Cette commande, comme on le voit dans son format syntaxique, ne s'applique tout au plus qu'à un seul uplet. Pour effectuer un tri sur les uplets de la relation, l'utilisateur doit au préalable exécuter son tri au moyen du langage hôte. Une fois le tri effectué, la place que doit occuper chaque uplet est connue avec exactitude. Il suffit pour trier définitivement la relation, d'utiliser une cascade de commande MV.

Opérations effectuées : voir figure n° 15.

2.6.3. CONCLUSION ET COMMENTAIRES.

1)- D'une manière formelle, nous pouvons représenter GAMMA - O par un ensemble comprenant :

- des objets (relations, uplets, articles).
- des identificateurs.
- des scans.
- des opérations.



Cependant, une base de données est un outil permettant l'accès aisé des données à de multiples utilisateurs. Sa définition et son établissement se font avant tout dans la voie de cette finalité. Toutefois, lors de l'étude GAMMA - 0, seuls les aspects concernant un seul utilisateur ont été abordés et approfondis. L'interface que représente GAMMA -0 ne sera pas utilisé comme tel et servira de base à l'étude d'un second système d'interface GAMMA - 1, destiné à contrôler le trafic de plusieurs utilisateurs.

Les raisons de l'élaboration préalable du système GAMMA - 0 sont dues au souci de structurer le travail selon deux aspects :

1) D'une part, étude sémantique de l'interface, c'est-à-dire les notions de structures de données, d'identificateurs et de commandes.
2) D'autre part, extrapolation de GAMMA-0 en GAMMA-1 grâce à l'étude notamment du contrôle du partage des données entre les divers utilisateurs et des problèmes d'interblocage du système.

2)- <u>Intégration de GAMMA-0 dans un langage hôte</u>.

GAMMA-0, grâce à ses commandes, assure à lui seul l'implémentation

et l'exploitation d'une base de données construite selon le modèle relationnel de Codd. Il constitue donc l'interface entre l'utilisateur (milieu extérieur) et les données et réalise en quelque sorte les entrées - sorties de ces données vers la base.

Le langage hôte, quant à lui, ne peut manipuler directement les données de la base. Son but est de rendre plus accessible à l'utilisateur plus ou moins occasionnel les données délivrées par GAMMA-O.

Au cours de ce travail, nous avons volontairement identifié les relations de deux façons. En effet, dans le premier chapitre, lors de la définition des relations, nous les avons identifiées par des noms bien concrets et bien suggestifs tels que relation 'EMPLOYE', relation 'PRODUITS', ... Cependant, dans le second chapitre, lors de l'étude de GAMMA-O, nous avons été obligés d'identifier les relations par des identificateurs attribués par le système et que l'on peut considérer comme une numérotation.

En réalité, il est normal que l'utilisateur nomme lui-même ses relations et les identifie de façon concrète afin que le modèle de ses données soit plus proche encore du monde réel.

Le langage hôte se charge :

- d'une part des traitements à effectuer sur les données.
- d'autre part d'enregistrer les demandes de l'utilisateur et de lui délivrer les réponses de GAMMA-O, cela au terme de noms concrets affectés par l'utilisateur.

Il serait intéressant, afin de réaliser cette concrétisation, de concevoir un système qui maintiendrait des tables de correspondance entre les identificateurs orientés utilisateurs, et les identificateurs orientés GAMMA-O. De plus, ce système aurait également pour tâche le stockage et la restitution des

identificateurs de uplets, de scans et d'inversions relatifs à une même relation.

CHAPITRE III.

IMPLEMENTATION DE GAMMA - O.

Jusqu'à présent, nous avons défini le problème sur le plan purement logique : définition du modèle de Codd et description de l'interface Gamma-O.

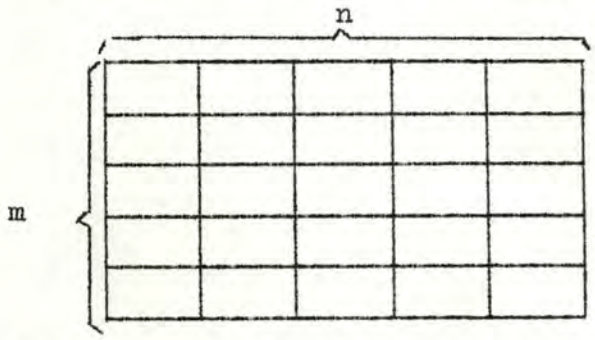
Afin que l'étude du modèle de notre base de données soit complète, il faut que nous abordions la structure de stockage ou implémentation physique.

Le premier point que nous abordons est l'organisation des fichiers.

3.1. ORGANISATION ET DEFINITION DES FICHIERS.

Il s'agit de stocker les éléments de Gamma-O dans des fichiers appropriés en conservant une assez bonne représentation du modèle relationnel et en exploitant au maximum les statistiques définies dans ce système.

Reprenons l'image que nous avions d'une relation normalisée : une table à n (degré) colonnes et m (cardinalité) lignes.



GAMMA-O impose en plus à cette table deux conditions :

1) La table comporte un uplet supplémentaire : le uplet de contrôle (qui n'intervient pas dans la cardinalité).
2) Les domaines pour les relations autres que les relations classes sont tous de longueur fixe.

3.1.1. ATTRIBUTION DES IDENTIFICATEURS :

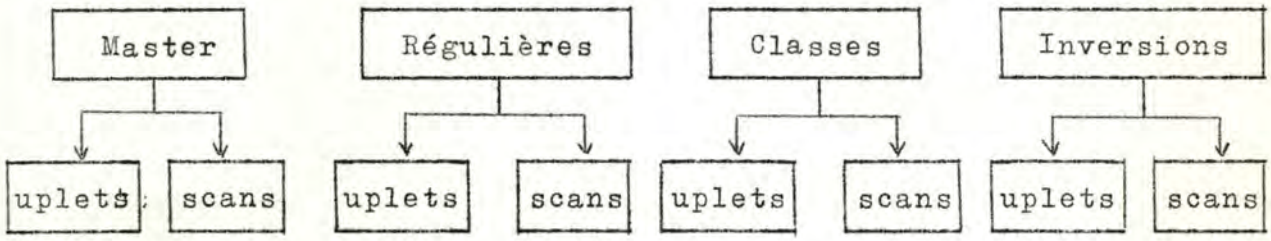
Dans la base de données, le système doit pouvoir identifier chaque relation d'une manière unique. Et pour une rela-

tion donnée, identifier d'une manière unique ses différents uplets. Afin d'exploiter une relation, chaque scan doit également pouvoir être retrouvé. Cette identification s'effectue grâce à un ensemble d'identificateurs assignés par le système à ses objets. En terme d'implémentation, on dira que les objets de GAMMA-O sont soumis à une clé unique.

Nous connaissons 4 types de relations :

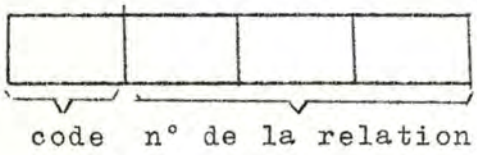
1) La relation maître (unique en son genre).
2) Les relations régulières.
3) Les relations classes.
4) Les relations inversions.

Cette partition dans l'ensemble des relations nous suggère une découpe analogue dans l'ensemble des identificateurs. De chaque identificateur de relation, dépendront hiérarchiquement les identificateurs de ses uplets, et des scans qui lui sont associés. Schématiquement, nous avons donc 4 sous-ensembles indépendants les uns des autres :



Relations.

Chaque identificateur de relation est représenté par un mot (4 bytes) dont la structure est la suivante :



Codes.

Nous choisissons un code pour chaque type de relation :

- relation maître : Code = 1.

- relation régulière : code = 2.
- relation classe : code = 3.
- relation inversion : code = 4.

L'identificateur de la relation maître est toujours 1 0 0 1 étant donné qu'elle est la seule dont le code peut être égal à 1. Grâce au code de la relation, lorsque dans une commande, un identificateur de la relation est spécifié, le système peut aisément vérifier de quel type de relation il s'agit.

Exemples :

1	0	0	1	relation maître.
2	0	3	6	36e relation régulière.
3	0	5	1	51e relation classe.
4	0	2	3	23e relation inversion.

Cette structure dans les identificateurs de relations nous impose cependant une limite dans le nombre des relations de type 2, 3 et 4. En effet, chaque type ne pourra comprendre plus de 999 relations. Nous estimons que pour une base de données expérimentale, cette limite est largement suffisante.

Uplets.

Lors de l'affectation des identificateurs de uplets d'une relation, le système ne distingue pas l'uplet de contrôle des autres uplets. En général, l'identificateur d'un uplet est représenté par la concaténation de 2 mots (ce qui exprime la hiérarchie) qui sont d'une part l'identificateur de la relation à laquelle il appartient, et d'autre part, le n° du uplet dans la relation.

Structure :

Identificateur de la relation. N° du uplet dans la relation.

Exemples :

Dans la relation maître :

1	0	0	1	0	0	0	1	uplet de contrôle
1	0	0	1	0	0	0	2	Ier uplet effectif

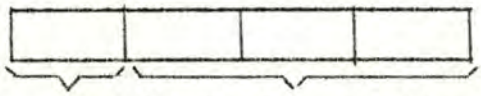
Dans les autres relations :

2	0	8	8	0	7	8	6	786e uplet de la 88e relation régulière.
3	0	2	5	0	0	3	8	38e uplet de la 25e relation classe.

Cette façon de faire nous permet d'insérer jusqu'à 9.998 uplets dans chaque relation.

Scans.

A n'importe quelle relation de la base de données, est toujours associée une pseudo-relation, la "relation d'état des scans". Cette pseudo-relation contient cependant un uplet de contrôle + un ensemble de uplets qui sont les scans définis sur la relation associée. Afin d'exprimer le lien entre une relation et sa relation d'état des scans, nous partitionnons l'ensemble des relations d'état de scans, de la même façon que l'ensemble des relations lui-même. Chaque identificateur de pseudo-relation est représenté par un mot et structuré de la façon suivante :



code N° de la relation

Code : RES associée à la relation maître : code = 5.

RES associée aux relations régulières : code = 6.

RES associée aux relations classes : code = 7.

RES associée aux relations inversions : code = 8.

Exemples :

5	0	0	1

Identificateur de la RES associée à la relation maître.

6	0	0	3

RES associée à la 3e relation régulière.

7	0	0	6

RES associée à la 6e relation classe.

⋮

Quant aux identificateurs de scans, ils sont formés de la même manière que les uplets dans une relation, c'est-à-dire par la concaténation de l'identificateur de la RES à laquelle ils appartiennent, et par leur n° de scan dans la RES.

Exemples :

5	0	0	1	0	0	3	2

Identificateur du 32e scan défini sur la relation maître.

7	0	0	8	0	0	2	6

26e scan défini sur la 8e relation classe.

8	0	3	9	1	5	0	0

1500e scan défini sur la 39e relation inversion.

Cette représentation des identificateurs nous permet de définir jusqu'à 9998 scans sur une même relation.

3.1.2. PERMANENCE DES IDENTIFICATEURS.

A chaque création d'un des éléments de GAMMA-O, ce dernier génère un identificateur qu'il associe à cet élément. On peut donc dire qu'il existe une relation biunivoque entre l'ensem-

ble des éléments identifiables et l'ensemble des identificateurs.

Les éléments de GAMMA-0, une fois créés, évoluent ensuite dynamiquement. Quels comportements possibles pouvons-nous adopter au sujet des identificateurs que le système a généré?

Lorsqu'un des éléments est supprimé, le système a le choix entre deux possibilités :

1)- Supprimer l'identificateur de l'élément en même temps que l'élément.

2)- Conserver l'identificateur.

1)- Cette méthode est l'affectation non permanente. Cette affectation comporte deux aspects :

a) L'élément identifiable ayant été supprimé, son identificateur ne peut plus être affecté à un élément nouvellement créé. Il n'existe donc plus pour le système.

b) L'élément identifiable supprimé, son identificateur est rendu disponible et fait partie de l'ensemble des identificateurs. Cette procédure doit nécessairement se passer au niveau des relations, des uplets et des scans.

Cette affectation nécessite une gestion des identificateurs, grâce à des vecteurs d'utilisation. Etant donné les 4 types de relations, un vecteur d'utilisation peut être établi par type de relation. Au type "maître", n'est pas affecté de vecteur d'utilisation, car la relation "maître" est unique, elle ne peut jamais être supprimée et son identificateur n'est jamais associé qu'à elle-même.

Nous sommes donc en présence de la situation suivante.

1	1	1	1	0	0	1	0	1	1	1	0

Relations régulières.

signifie que l'identificateur 2001 identifie une relation régulière

signifie que l'identificateur 2008 a été libéré, et est prêt à identifier une nouvelle relation régulière

Lors de la création d'une relation régulière, le système parcourt ce vecteur d'utilisation, en cherchant le Ier bit = 0. Ce bit trouvé, le système lui donne la valeur 1, puis il construit l'identificateur à affecter à la relation créée.

Dans ce cas, le nouvel identificateur est égal à 2 0 0 5.

Comme cette procédure doit se répéter au niveau des uplets et au niveau des scans, nous devons également tenir à jour pour une relation donnée :

- un vecteur d'utilisation des identificateurs de uplets.
- un vecteur d'utilisation des identificateurs de scans.

<u>Exemple</u> : relation n° 3024

1	0	1	1	1	1	1	0	----------

L'identificateur n° | 3 | 0 | 2 | 4 | 0 | 0 | 0 | 2 | est libre

Nous faisons remarquer qu'un vecteur d'utilisation des identificateurs de uplets et qu'un vecteur d'utilisation des identificateurs de scans, doivent être aussi associés à la relation maître.

<u>REMARQUES</u> :

1) Le premier aspect de l'affectation permanente est très simple. Il suffit de posséder un compteur par type de relation, pour les relations un compteur par relation pour les uplets

et un compteur par relation pour les scans.
Cependant, cet aspect n'est applicable que pour une base de données qui n'évolue pas, ou très peu (signalétiques).

2) Le second aspect contrairement au premier, s'avère beaucoup plus lourd. Mais il est applicable à une base de données qui évolue assez bien, c'est-à-dire qui comporte beaucoup de créations et de suppressions d'éléments. Cette technique est l'affectation permanente. Lorsqu'un élément de GAMMA-O est supprimé, son identificateur est réservé et est réaffecté à ce même élément lors de sa recréation éventuelle.

Il peut paraître paradoxal de supprimer puis de recréer un même élément. Cette façon de faire peut cependant se comprendre lors d'une modification du nombre des domaines d'une relation.

3.2. DEFINITION DES FICHIERS ET DES ARTICLES LOGIQUES.

Jusqu'à maintenant, nous avons élaboré toute notre théorie, indépendamment de la manière dont elle peut être prise en charge par l'ensemble des mémoires auxiliaires et principale. Afin de rendre complète cette analyse de base de données relationnelle, nous terminerons l'étude de l'implémentation par la définition des articles et des fichiers, et par le détail de la programmation des commandes de GAMMA-O. Comme les notions de structure des articles dans un fichier, et de mode d'accès à ce fichier sont totalement interdépendantes, nous les présenterons en corrélation.

3.2.1. CHOIX DES METHODES D'ACCES.

Afin de réaliser notre implémentation, nous avons opté pour les fichiers séquentiels indexés. Deux raisons furent à l'origine de ce choix :

a) la localisation d'un article se fait d'une façon directe

sans que l'utilisateur ni le programmeur n'aient à calculer son adresse. Ceci est très intéressant dans cette implémentation du modèle relationnel de B.F. CODD car la biunivocité entre un uplet et son identificateur doit toujours être vérifiée.

b) la structure indexée séquentielle s'adapte particulièrement bien à des fichiers dont la séquence d'indicatifs présente trop de discontinuité. Encore une fois, cette propriété s'adapte au problème qui nous intéresse. En effet, lors de la consultation et de l'exploitation de la base de données, l'ordre d'apparition des identificateurs des relations est tout à fait quelconque.

3.2.2. CHOIX DES FICHIERS.

Reprenons séparément nos quatre types de relations :

A. La relation maître.

Constituant en fait le catalogue des relations, elle possède une sémantique bien spécifique. Cependant, vis-à-vis de l'implémentation, nous la considérons comme une relation ordinaire. C'est-à-dire que nous la stockons dans un fichier indexé séquentiel dont les enregistrements sont de longueur fixe et dont la clé est formée par la concaténation de l'identificateur de la relation et l'identificateur du uplet.

B. Les relations régulières.

La plupart des données disponibles dans les fichiers ordinaires d'une entreprise peuvent se retrouver dans des relations régulières de la base de données.

Chacune de ces relations est représentée par un tableau à N colonnes (degré) et M lignes (cardinalité).

N est fixé par le créateur de la relation, tandis que M évolue au fur et à mesure des mises à jour.

Au point de vue implémentation, nous avons retenu la solu-

tion suivante :

Nous créons un fichier destiné à contenir l'ensemble des relations régulières. Ce fichier possède des enregistrements de longueurs variables étant donné que les relations sont de degrés différents.

Sa clé d'accès est formée de l'identificateur de la relation suivi de l'identificateur du uplet.

Au fond, ce fichier n'est autre qu'un ensemble de fichiers.

Cette façon de faire nous permet des suppressions et créations dynamiques de relations.

C. Les relations classes.

Comme pour les relations régulières, nous créons un fichier qui contient toutes les relations classes de la base de données. Ce fichier aura également des enregistrements de longueur variable.

Clé = identif. relation + id. uplet.

D. Les relations inversions.

On ouvre un fichier global qui contient toutes les relations inversions de la base de données. Contrairement aux fichiers des relations régulières et des relations classes ce dernier fichier est constitué d'enregistrements de longueur fixe. En effet, les inversions sont toutes de degré 2.

Clé = identif.relation + id. uplet.

E. Les relations d'état des scans.

Par type de relation, nous créons un fichier où sont stockées les relations d'état des scans. Nous avons donc 4 fichiers. A part celui associé à la relation maître, ils reflètent chacun l'état de l'utilisation de leurs fichiers correspondants.

Clé = identif.relation + identif scan.

3.3. PROGRAMMATION DES COMMANDES DE GAMMA-O.

Afin de compléter cette implémentation, il nous reste maintenant à présenter la programmation des commandes de GAMMA-O.

3.3.1. MODULES PRIMAIRES.

a) Création d'une relation : CR (voir figure n° 16).
b) Suppression d'une relation : DR (voir figure n° 17).
c) Insertion d'un uplet : IT (voir figure n° 18).
d) Suppression d'un uplet : DT (voir figure n° 19).
e) Mise à jour d'un sous-uplet : US (voir figure n° 20).
f) Création d'un scan : CS (voir figure n° 21).
g) Etablissement d'un scan : SS (voir figure n° 22).
h) Recherche filtrée (scan suivant) : NS (voir figure n° 23).
i) Suppression d'un scan : DS (voir figure n° 24).
j) Recherche de l'identificateur d'un uplet : TIFNS (voir figure n° 25).
k) Génération d'une inversion : GI (voir figure n° 26).
l) Suppression d'une inversion : DI (voir figure n° 27).
m) Déplacement d'un uplet : MT

Cette commande nous pose quelques problèmes. En effet, elle permet de modifier l'ordre d'apparition séquentielle des uplets d'une relation donnée. Cependant, notre méthode d'affectation des identificateurs de uplets représente une fonction croissante. C'est-à-dire que si dans une même relation, un uplet A est créé avant un uplet B, l'identificateur de A est inférieur à l'identificateur de B.

Comme nous avons choisi une structure d'accès indexée-séquentielle, cette commande n'est pas possible à réaliser. Une gestion de la relation au moyen de chaînages résoudrait ce problème, mais par la même occasion, rendrait impraticables certaines autres

commandes plus importantes quant à la maintenance de la base de données.

Si toutefois un certain ordre dev ait être respecté dans l'apparition des uplets, l'utilisateur devrait être capable de mettre un critère en évidence et de créer une inversion qui pourrait résoudre partiellement le rangement désiré.

3.3.2. MODULES SECONDAIRES.

Dans notre implémentation, il est à distinguer deux catégories de modules à appliquer aux données.

D'une part, les commandes directement représentatives de l'exploitation de la base de données (création, suppression).

D'autre part, les modules d'accès ou module de servitude dont le but est de rendre le plus linéaire possible les séquences d'opérations dont sont constitués les modules principaux.

Avant de terminer ce chapitre, nous nous proposons d'exposer brièvement quelques uns de ces modules secondaires.

a) Existence d'une relation :

Le fait de rechercher l'existence d'une relation dans la base de données consiste en le balayage de la colonne n°7 de la relation maître. Si l'identificateur de la relation cherchée figure effectivement dans la relation maître, la description de la relation concernée est bien cataloguée et cette dernière existe dans la base de données.

b) Recherche filtrée :

Le concept de recherche filtrée dans un fichier réside essentiellement en la lecture séquentielle de ce fichier et en la comparaison domaine par domaine avec les filtres spécifiés.

c) Existence d'un uplet ou d'un scan.

Lors de la recherche de l'existence d'un uplet ou d'un scan, il suffira d'accéder à ce uplet ou à ce scan.

D'une manière générale, nous utilisons un module pour chaque manipulation de fichier.

CHAPITRE IV.

LES PROTECTIONS

Un exemple d'utilisation.

4.1. POURQUOI PROTEGER LES DONNEES ?

Protéger les données en informatique signifie éviter les conséquences de manipulations intempestives dans le système d'informations.

Ces manipulations peuvent en fait être :

- soit des erreurs d'entrées, sorties de fichier,avec destruction des données.
- soit des indiscrétions de la part des utilisateurs.
- soit des erreurs de logiques dans des programmes.

4.1.1. DEFINITION PAR RAPPORT A UN CONTEXTE DE BASE DE DONNEES.

Dans une base de données, (de part son caractère commun) le problème de la protection des informations est essentiel. En général, un système de base de données offre le moyen de se protéger lui-même, contre le milieu extérieur. Cette protection s'effectue habituellement par l'utilisation de mots de passe et de clés ouvrant ou bloquant l'accès aux données. De plus ces mots de passe peuvent être organisés en hiérarchie et donc s'appliquer à des niveaux différents de données de la base. Dans ce contexte 'base de données', la protection peut donc se faire jusqu'à la donnée elle-même.

Pour en revenir à notre exemple de l'entreprise industrielle, cette particularité de la protection de la base de données pourrait assurer la structure des flux dans l'entreprise tout en ne permettant les manipulations des utilisateurs que dans les domaines dont la connaissance leur est nécessaire au sein de leurs activités.

Dans ce travail, deux aspects de la protection vont être envisagés :

1- d'une part : permettre ou interdire sous certaines conditions ou contraintes l'accès de la base de don-

nées à ses divers utilisateurs.

2- d'autre part, dans la même base de données, ne donner l'accès aux utilisateurs qu'à des ensembles restreints de données (hiérarchisation des utilisateurs et des ensembles de données).

3- une troisième forme de protection appelée 'consistance de la base de données' n'intervient que lorsque les données sont soumises à un certain ensemble de contraintes d'intégrité. Nous n'aborderons que très brièvement ce cas.

4.1.2. DEFINITION PAR RAPPORT A L'INTERFACE GAMMA-0 IMPLEMENTE.

Si un seul utilisateur peut exploiter la base de données, cet utilisateur en est le gestionnaire et il est pratiquement inutile de s'attarder aux deux premiers aspects de la protection.

Cependant, si plusieurs utilisateurs désirent se partager la base de données, un mécanisme contrôlant leurs accès doit être mis au point.

Dans l'étude de GAMMA-0, nous avons précisé qu'il s'agissait d'un système conçu uniquement pour les manipulations d'un seul utilisateur. C'est pourquoi, à partir du moment où nous désirons aborder les mécanismes de protections, nous nous imaginons en présence de GAMMA-1 qui est le niveau directement supérieur à GAMMA-0 et qui en restreint l'utilisation, grâce à sa gestion des protections de données.

Nous nous proposons donc dans ce chapitre de développer une partie de GAMMA-1 conçu notamment en fonction des mécanismes de protection.

Les primitives de GAMMA-1 régissant la protection s'expriment en terme de primitives et d'objets de GAMMA-0,

GAMMA-1 agit donc comme un mécanisme combinant les éléments de GAMMA-0 pour en dégager des contraintes qui se répercutent dans GAMMA-0 tout en limitant les possibilités de ce dernier en partageant la base de données.

Le système de protection de la base se construit sur deux catalogues analogues à la relation maître.

Ces catalogues ne sont en fait rien d'autre que des relations comme nous les avons définies dans GAMMA-0. Ces catalogues sont cependant uniques et utilisés au niveau de GAMMA-1.

A. Catalogue des utilisateurs :

Ce catalogue reprend en regard de chaque utilisateur la liste des opérations qu'il peut effectuer dans la base de données via GAMMA-0.

B. Catalogue des mots de passe :

Ce catalogue reprend en détail chaque relation et décrit pour chaque utilisateur la façon dont elle peut être exploitée.

4.1.2.1. Catalogue des utilisateurs :

Ce catalogue est unique et créé lors de l'initialisation de GAMMA-1. Il est de degré 1 et est utilisé pour stocker les informations relatives aux possibilités d'accès des utilisateurs aux données de la base. Sa cardinalité évolue dynamiquement selon les créations et suppressions de permissions d'accès.

Structure du catalogue :

Chacun de ses uplets à la structure suivante :

mot de passe	masque des primitives permises à l'utilisateur.

clé

L'identificateur de l'utilisateur ou mot de passe joue le rôle de clé et attribue une partie de la base à l'utilisateur.

Le uplet effectif contient le masque des primitives que l'utilisateur peut employer. Ce masque est de 12 positions, chaque position représentant une des primitives que nous avons décrites ci-avant dans GAMMA-O.

CR	DR	IT	DT	US	CS	SS	NS	DS	TIFNS	GI	DI

Lors de l'utilisation du système ce catalogue ne contient qu'un seul uplet : le uplet relatif au gestionnaire de la base de données.
Son masque d'opérations possibles est toujours

1	1	1	1	1	1	1	1	1	1	1	1	1	1

Avant qu'un utilisateur lance une commande afin d'exploiter la base de données, il doit obtenir l'approbation du gestionnaire.
Ce dernier rend possible l'opération soit par l'introduction dans le catalogue des utilisateurs d'un nouveau uplet, soit par la mise à jour d'un uplet déjà existant.

4.1.2.2. Catalogue des mots de passe :

Ce catalogue est également unique et créé lors de l'initialisation de GAMMA-1. Il a pour but de détailler par relation, les possibilités offertes aux utilisateurs.
Il est de degré 1 et sa cardinalité évolue également en fonction des créations et des suppressions de possibilités accordées par le gestionnaire.

Structure du catalogue :

- Le mot de passe de l'utilisateur suivi de l'identifica-

teur de la relation sert de clé d'accès à ce catalogue.

- Chaque uplet se compose d'une liste des 32 domaines possibles d'une relation. Chacune des positions de ce masque contient un code indiquant le nombre et le type des opérations que l'utilisateur peut appliquer aux domaines de la relation.

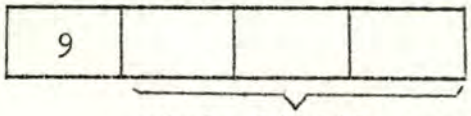
Dans l'ensemble des opérations permises à un utilisateur deux d'entre elles concernent directement les domaines d'une relation. Il s'agit de US (mise à jour d'un sous-uplet)
SS, NS (établissement de contraintes et / ou obtention de certains domaines).

Codes :
- La valeur '0' dans un domaine signifie que rien ne peut être permis sur ce domaine.
- 1 : US permis.
- 2 : SS1 (la première partie du SS, ou obtention d'une liste de domaines peut être assimilée au NS).
- 3 : SS2 (établissement de contraintes permise)
- 4 : US, SS1, NS.
- 5 : US, SS2.
- 6 : SS1, SS2, NS (ou SS complet + NS).
- 7 : US, SS, NS.

4.1.2.3. Mécanisme de la protection :

a) Attribution des mots de passe :

Afin d'identifier chaque utilisateur et de définir l'espace des données qui lui est attribué, le système reconnait tout un ensemble de mots de passe.
Chaque mot de passe d'utilisateur est représenté par un mot (4 positions) dont la structure est la suivante :



b) Création, mises à jour et suppressions de protections.

Par définition, le gestionnaire de la base de données est une personne physique ou une équipe, responsable de la (des) base(s) de données de l'entreprise. Ce gestionnaire assure donc la maintenance, l'enregistrement et la gestion des données vis-à-vis de leur partage entre leurs divers utilisateurs.

Il est dès lors bien évident que le gestionnaire puisse manipuler les outils de gestion de la base de données, c'est-à-dire, entre autres, accéder aux trois catalogues (relations, mots de passe, utilisateurs).

Nous savons déjà que le gestionnaire a le pouvoir d'utiliser toutes les opérations de GAMMA-O, ce qui revient à dire que son uplet dans le catalogue des utilisateurs possède des '1' dans chaque position de son masque.

Quant au catalogue des mots de passe : contrairement aux autres utilisateurs, il y est mentionné que le gestionnaire peut avoir accès aux deux catalogues relatifs à la protection des données.

Ces possibilités permettent au gestionnaire une protection dynamique de la base de données que nous détaillons ci-après :

1.- Création d'un utilisateur :

La création d'un utilisateur dans le catalogue des utilisateurs, se fait par une opération spéciale : CRU (create user) qui est implémentée par GAMMA-O. Cette opération à la forme suivante :

CRU <mot de passe> <paramètres>

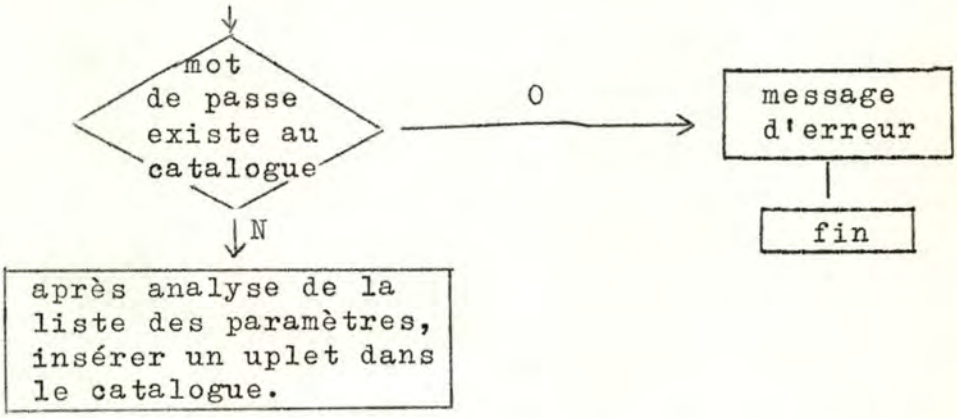
De manière plus précise, le catalogue des utilisateurs exige donc deux choses :

(1) la spécification d'un mot de passe.

(2) une liste de paramètres. Cette liste est rédigée sous forme de masque et n'est rien d'autre que l'image d'un uplet du catalogue des utilisateurs, c'est-à-dire la liste des opérations permises à l'utilisateur que l'on crée.

Cette opération CRU permet, si une opération est indiquée, de positionner l'indicateur correspondant dans le uplet à créer.

Nous pouvons représenter schématiquement l'implémentation de CRU de la façon suivante :



Nous pouvons aussi imaginer des primitives du même genre pour la mise à jour et la suppression des mots de passe, telles que :

UPU (update user), <mot de passe>, <paramètres>

DLU (delete user), <mot de passe>.

Leurs fonctionnement et implémentation sont analogues à ceux de CRU.

2.- Création de protections :

Il est bien évident qu'un utilisateur présent unique-

ment dans le catalogue des utilisateurs ne possède toujours aucun moyen d'action sur les domaines d'une relation de la base de données.

Il faut maintenant lui créer ce que nous pouvons appeler des possibilités, dans le catalogue des mots de passe.

L'opération qui va permettre cette création est une opération spéciale (CRP (create possibility)), dont la forme est la suivante :

CRP <mot de passe>, <ident. de la relation concernée>, < masque des domaines >.

La création d'un uplet dans le catalogue des mots de passe demande donc les spécifications suivantes :

- mot de passe de l'utilisateur.
- identificateur de la relation.
- masque des domaines : c'est l'image domaine par domaine des opérations possibles sur les domaines de la relation.

<u>Exemple pour une relation de degré 4</u> :

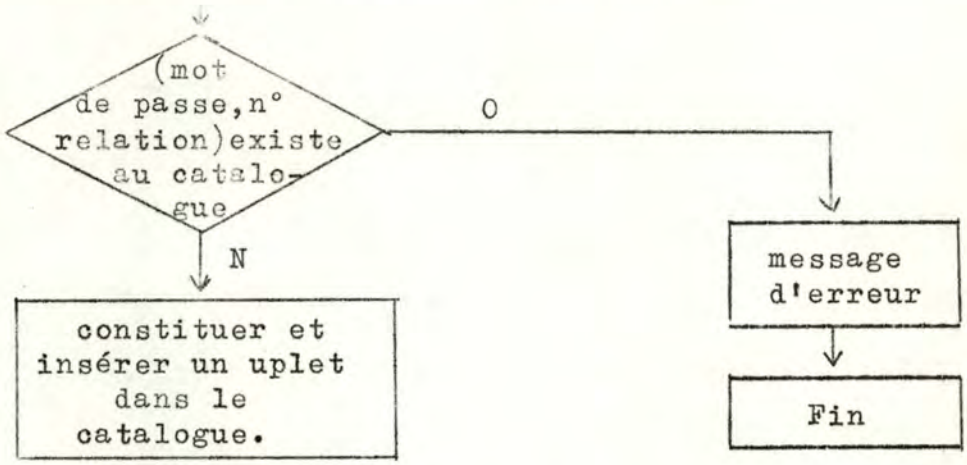
	US , SS	NS	NS , SS	SS1
domaine	1	2	3	4

ce qui signifie que l'utilisateur peut

- mettre à jour le premier domaine et établir un scan sur ce même domaine.
- lancer une opération de scan suivant pour le deuxième domaine.
- scan suivant et établissement d'un scan sont permis sur le troisième domaine.
- le quatrième domaine peut figurer dans le filtre d'établissement d'un scan.

Nous pouvons représenter schématiquement son implé-

mentation de la façon suivante :



Des primitives de mise à jour ou de suppressions dans ce même catalogue seraient les suivantes :

UPP (update possibility) <mot de passe>,
<identif. relation>, <masque>

DLP (delete possibility) <mot de passe>

c) Syntaxe des commandes :

Nous avons vu que la syntaxe des primitives du système GAMMA-0 a la forme suivante :

OP, <paramètres d'entrées> <paramètres de sorties>

Dans le système GAMMA-1 soumis au système de protection, nous pouvons donc concevoir que l'opération mentionne le mot de passe de l'utilisateur et prenne la forme :

OP, <mot de passe>, <paramètres d'entrée>,
<paramètres de sortie>.

d) Déroulement d'une opération :

1. Demande d'exécution d'une opération de la part d'un utilisateur.
2. Grâce au mot de passe de l'utilisateur accès au catalogue des utilisateurs.

3. La position du masque correspondant au type de l'opération est-elle positionnée ?
4. Si la position est à 0, l'opération est refusée.
5. Si la position est à 1 et si l'opération est du type US, NS, ou SS, en utilisant l'identificateur de la relation et le mot de passe de l'utilisateur, accès au catalogue des mots de passe.
6. Comparaison des domaines avec l'opération désirée.
   Si concordance, acceptation et exécution de la commande (à ce moment, nous rentrons dans le circuit normal GAMMA-0).
   Si non concordance, refus de la commande.

REMARQUE :

- Il est évident que si un utilisateur crée une relation, il est en soit le propriétaire. Toute création d'une relation entraîne donc automatiquement l'insertion d'un uplet dans le catalogue des mots de passe. Ce uplet prend la valeur '7' pour chaque position de son masque.
  La suppression d'une relation entraîne la suppression de tous ses uplets descriptifs dans ce même catalogue.
- La création d'une relation entraîne la création d'un uplet du catalogue des mots de passe, relatif au gestionnaire.

4.2.1. CONSISTANCE DANS LA BASE DE DONNEES.

Comme nous l'avons annoncé, nous nous bornons à présenter un aperçu très succin sur la consistance dans la base de données relationnelle.

En plus des protections associées à l'exploitation de la base de données, il existe tout un ensemble de règles de cohérence régissant les différentes associations de données.

Ces règles contrôlent les données au point de vue de leur contenu sémantique. Elles sont donc représentées par des contraintes d'intégrité sur les réalisations et les occurrences de ces données.

Nous pouvons donc considérer ces contraintes comme étant un ensemble de restrictions dues à la nature même de la base de données, permettant d'assurer au système toute sa cohérence.

Tout comme pour le problème des protections, il est possible d'imaginer dans GAMMA-1, un système maintenant la consistance de la base de données et s'exprimant en terme de relations.

Ce mécanisme s'appuierait tout comme le mécanisme de protection, sur un ensemble de catalogues possédant des fonctions bien particulières.

Nous pourrions distinguer par exemple :

- des catalogues de fourchettes de valeurs, interdisant à la donnée de se situer en dehors de ses limites.
- des catalogues sous forme de tables de décision exprimant des liaisons entre plusieurs domaines et interdisant à une donnée d'être prise en charge si elle ne vérifie aucune des conditions.
- des catalogues de pointeurs chaînant entre elles toutes

les occurences d'une même donnée, ce qui permettrait des mises à jour en cascades de cette donnée.

CONCLUSION.

Au terme de ce chapitre, il est bon de conclure en soulignant l'importance des protections et consistance dans une base de données.

En résumé - le système de protections partage la base de données en sous-ensembles répartis entre les utilisateurs.

- le système de consistance, une fois le sous-ensemble alloué, travaille au niveau de la donnée elle-même.

Protection et consistance sont les facteurs clés pour le développement d'un système sûr et assurent à tout instant une base de donnée cohérente et reflet de l'entreprise elle-même.

CONCLUSION ET PERSPECTIVES D'AVENIR.

Au cours de ce travail, nous avons parcouru différents aspects d'une base de données relationnelle.

Tout d'abord, nous l'avons située par rapport à une entreprise : Pourquoi est-il intéressant de posséder une base de données et quel y est son véritable rôle ?

Ensuite, nous avons défini les principes de la base de données relationnelle de CODD.

Enfin, nous avons étudié l'implémentation de cette base de données grâce au SGBD GAMMA et en particulier,grâce à son interface GAMMA-0.

Volontairement, nous n'avons pas insisté sur la facette programmation. Notre but n'était pas de développer chaque module en détail, mais bien de prouver qu'une application pratique relative à ce travail était réalisable. Il nous est apparu que les primitives gérant la base de données sont assez simples et assez souples à programmer dans un contexte de programmation modulaire.

Pour terminer, nous avons développé un exemple d'application relatif au problème des protections dans GAMMA-1. Nous pouvons en déduire que les primitives que nous avons définies peuvent finalement s'implémenter en termes de relations et de primitives de GAMMA-0. Par extension, il en serait de même pour tous les autres niveaux.

Une dernière partie aurait été très intéressante à développer sous forme de discussion dans ce travail : Les répercussions et modifications qu'apporte une telle base de données dans l'entreprise.
Cependant le manque de possibilités ne nous a pas permis de dégager ces conséquences pratiques. Nous pensons que l'expérience que nous acquerrons au cours de notre vie professionnelle, pourra nous guider dans cette voie et nous faire appliquer avec succès les théories que nous avons développées ci-avant.

F I G U R E S

Fig. 1

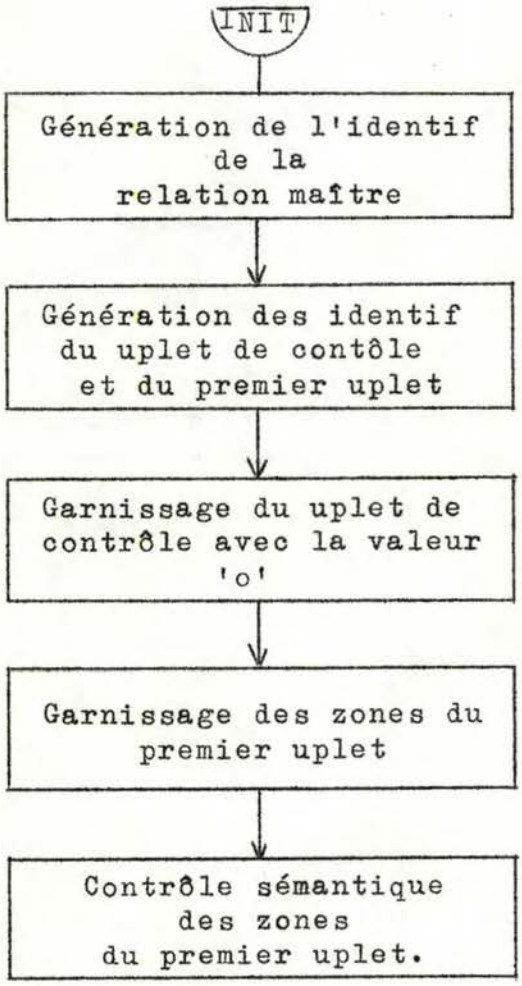

INIT
Génération de l'identif de la relation maître
Génération des identif du uplet de contôle et du premier uplet
Garnissage du uplet de contrôle avec la valeur 'o'
Garnissage des zones du premier uplet
Contrôle sémantique des zones du premier uplet.

Fig. 2

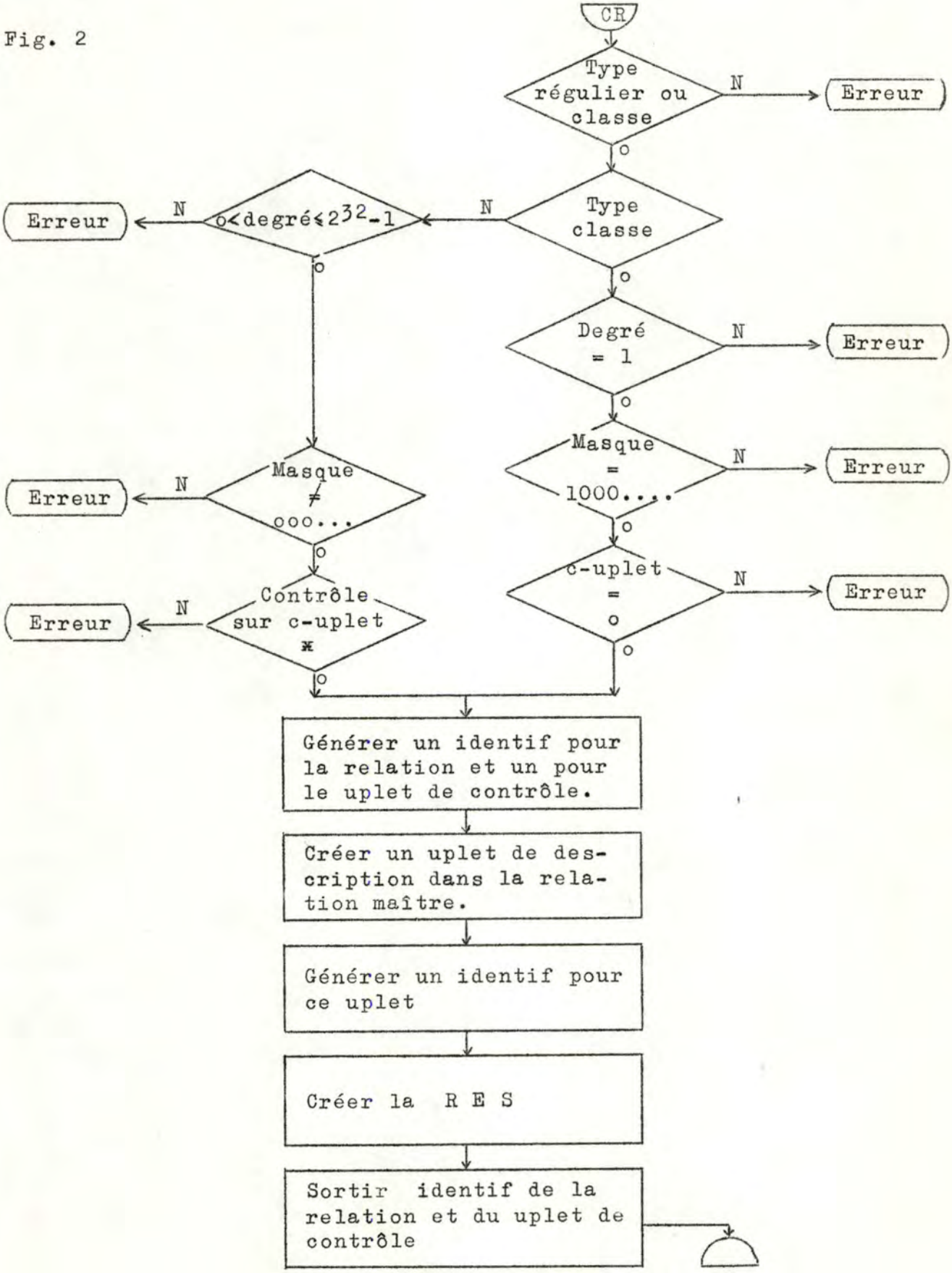

CR
Type régulier ou classe
N
Erreur
O
Type classe
N
0<degré≤2^32-1
N
Erreur
O
Degré = 1
N
Erreur
O
Masque = 1000....
N
Erreur
O
c-uplet = o
N
Erreur
O
Masque ≠ ooo...
N
Erreur
O
Contrôle sur c-uplet *
N
Erreur
O
Générer un identif pour la relation et un pour le uplet de contrôle.
Créer un uplet de description dans la relation maître.
Générer un identif pour ce uplet
Créer la R E S
Sortir identif de la relation et du uplet de contrôle

Figure 3

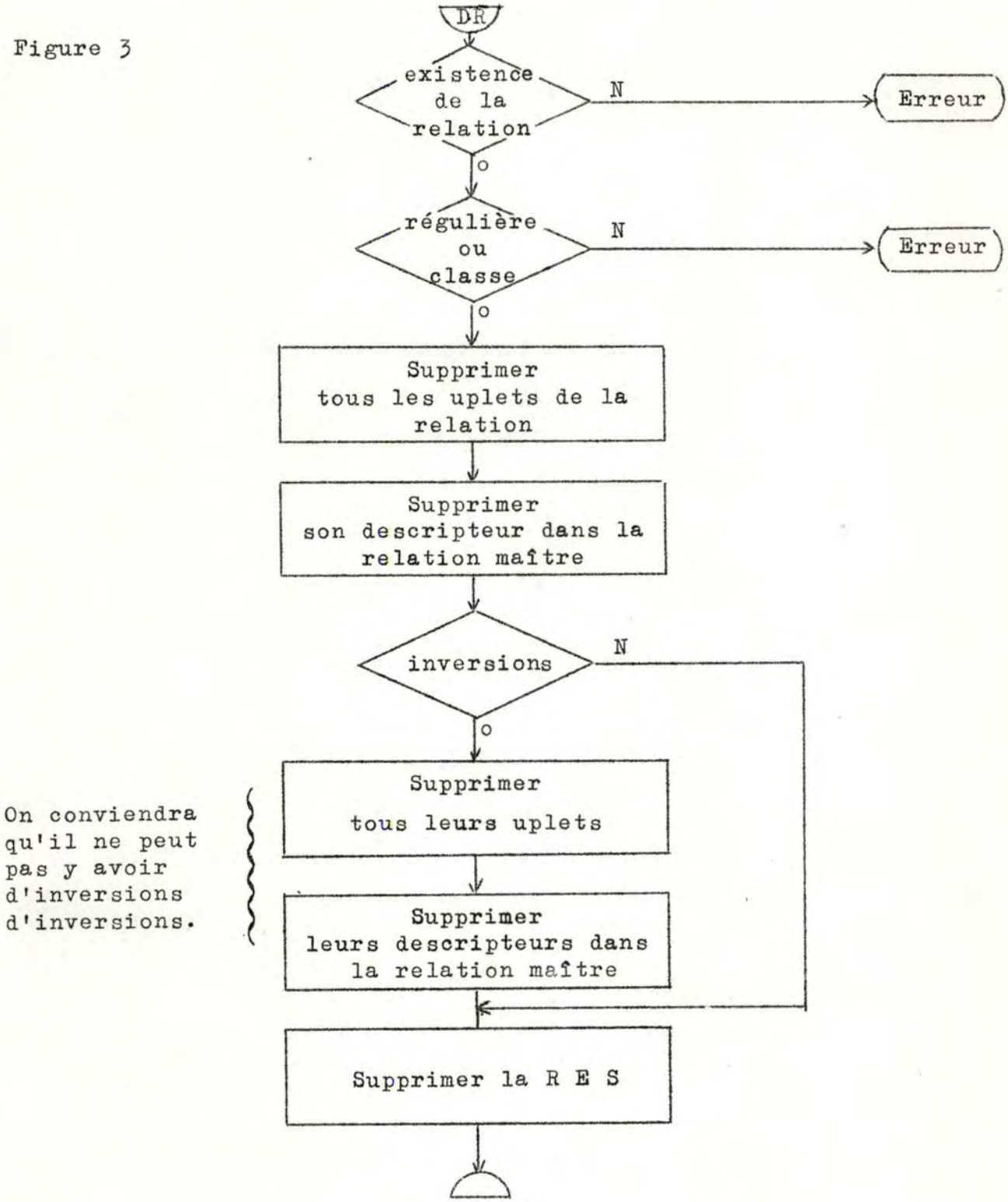

DR
existence de la relation
N
Erreur
o
régulière ou classe
N
Erreur
o
Supprimer tous les uplets de la relation
Supprimer son descripteur dans la relation maître
inversions
N
o
Supprimer tous leurs uplets
Supprimer leurs descripteurs dans la relation maître
Supprimer la R E S
On conviendra qu'il ne peut pas y avoir d'inversions d'inversions.

Figure 4.

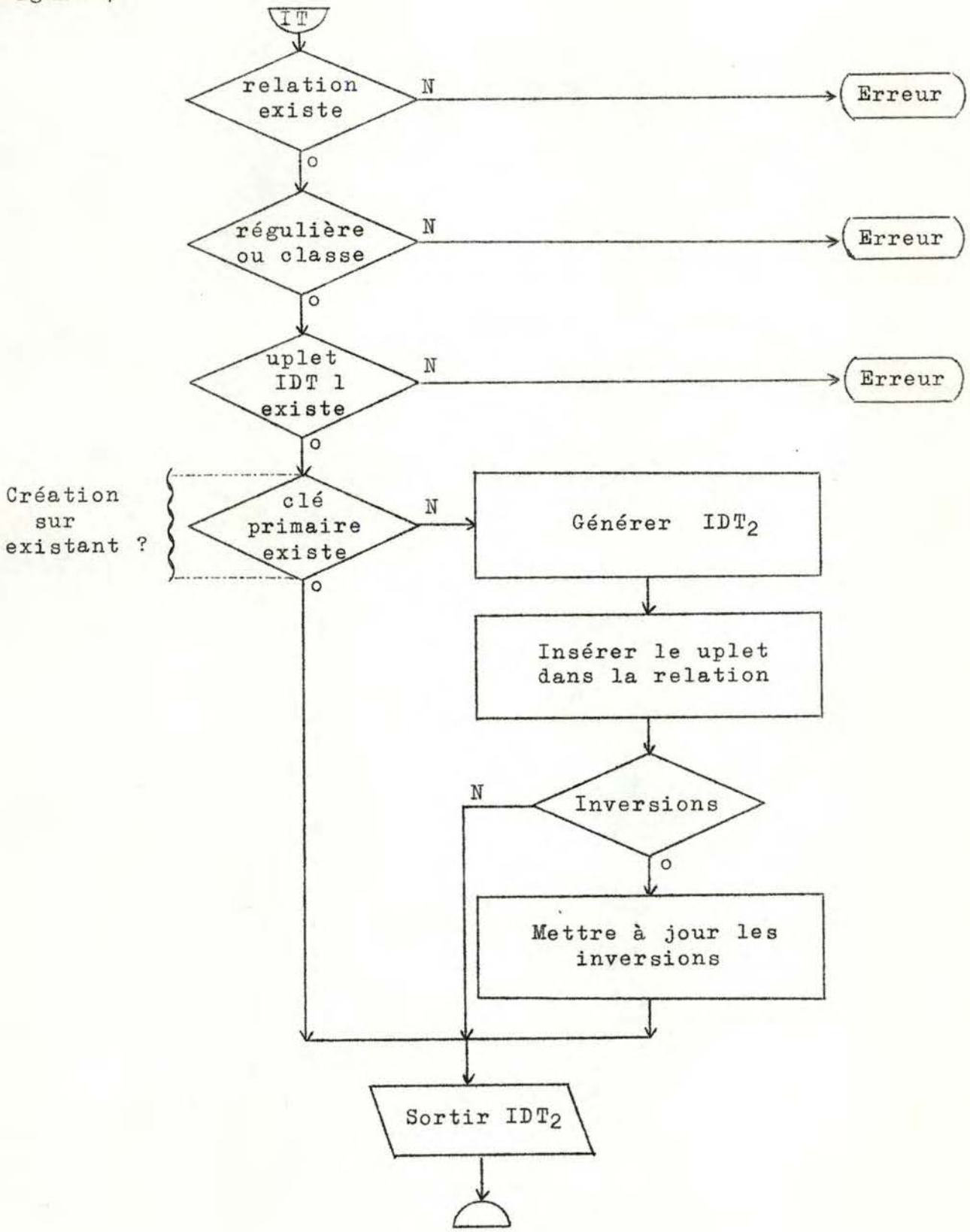

IT
relation existe
N
Erreur
O
régulière ou classe
N
Erreur
O
uplet IDT 1 existe
N
Erreur
O
Création sur existant ?
clé primaire existe
N
Générer IDT2
O
Insérer le uplet dans la relation
N
Inversions
O
Mettre à jour les inversions
Sortir IDT2

Figure 5.

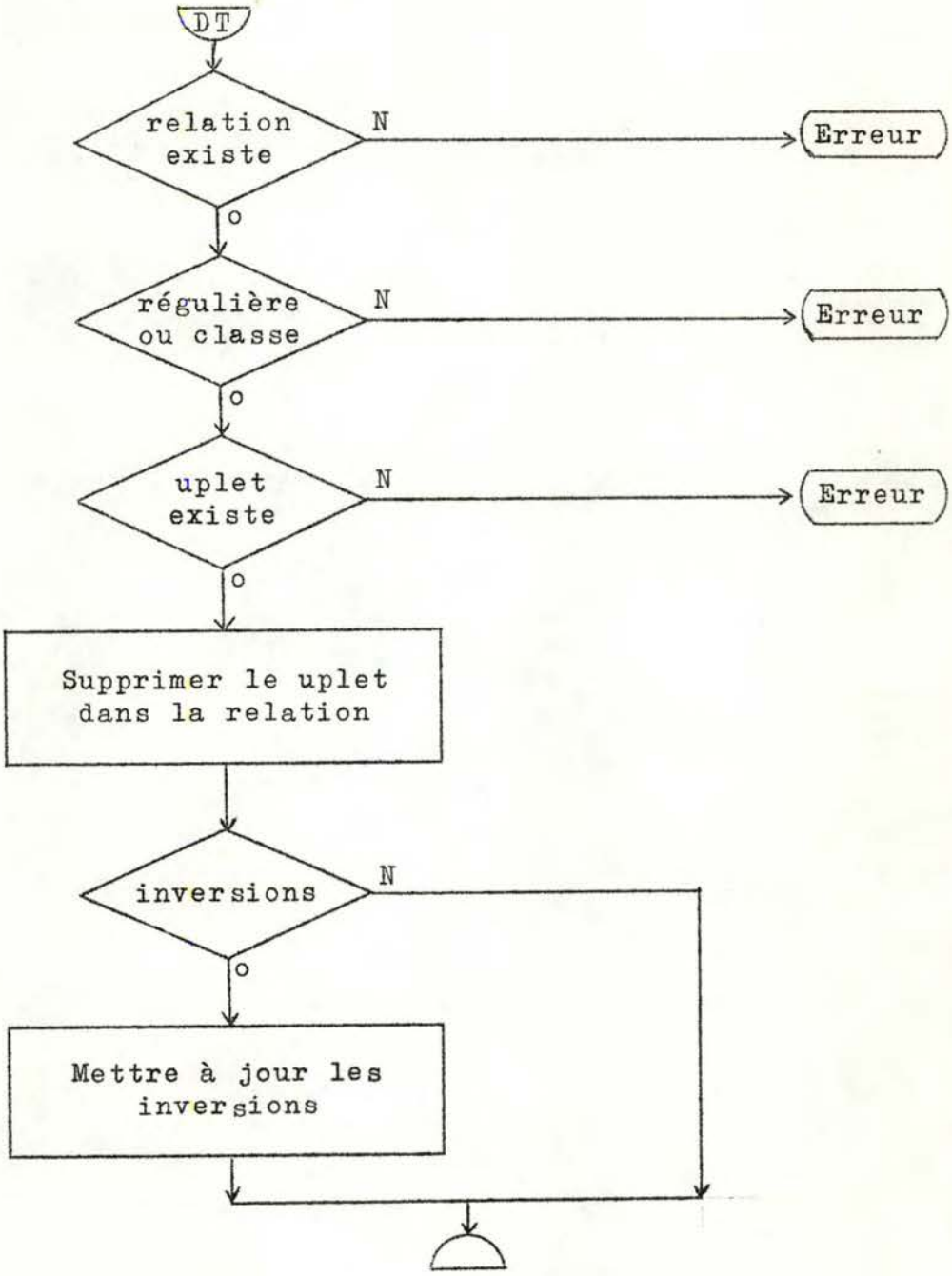

DT
relation existe
N
Erreur
O
régulière ou classe
N
Erreur
O
uplet existe
N
Erreur
O
Supprimer le uplet dans la relation
inversions
N
O
Mettre à jour les inversions

Figure 6

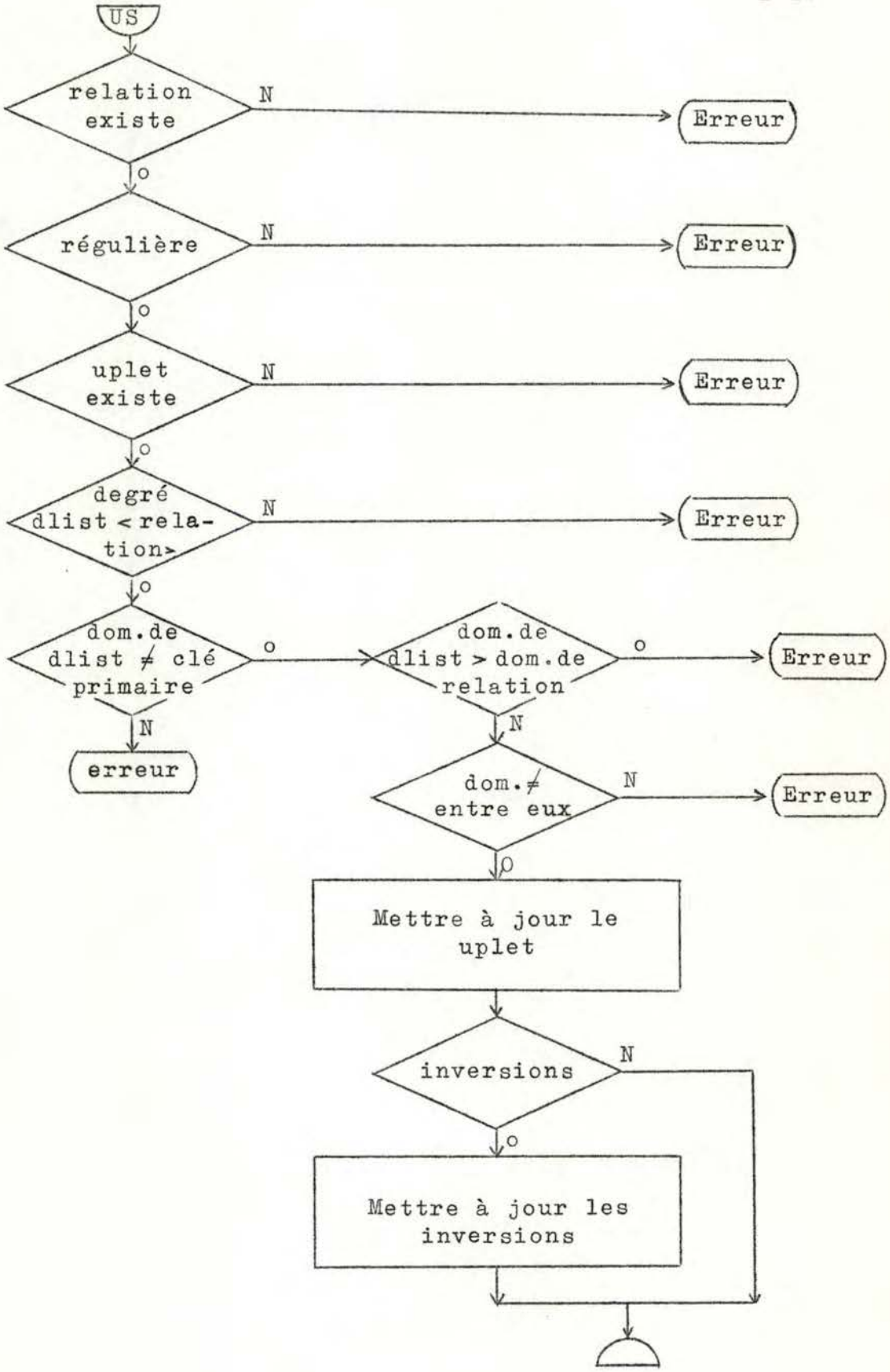

US
relation existe
N
Erreur
O
régulière
N
Erreur
O
uplet existe
N
Erreur
O
degré dlist < relation>
N
Erreur
O
dom.de dlist ≠ clé primaire
O
dom.de dlist > dom.de relation
O
Erreur
N
erreur
N
dom.≠ entre eux
N
Erreur
O
Mettre à jour le uplet
inversions
N
O
Mettre à jour les inversions

Figure 7

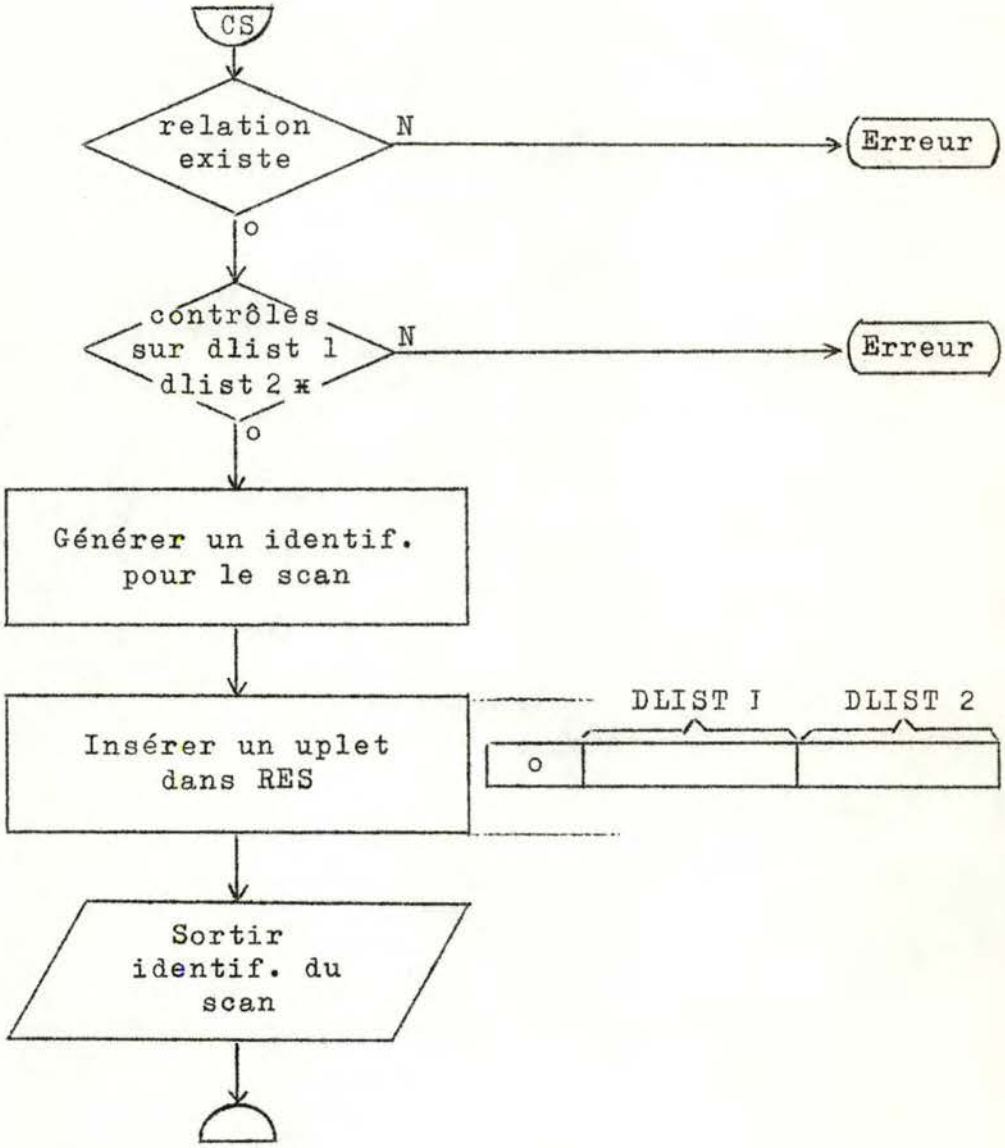

CS
relation existe
N
Erreur
O
contrôles sur dlist 1 dlist 2 *
N
Erreur
O
Générer un identif. pour le scan
Insérer un uplet dans RES
DLIST I
DLIST 2
O
Sortir identif. du scan

Figure 8

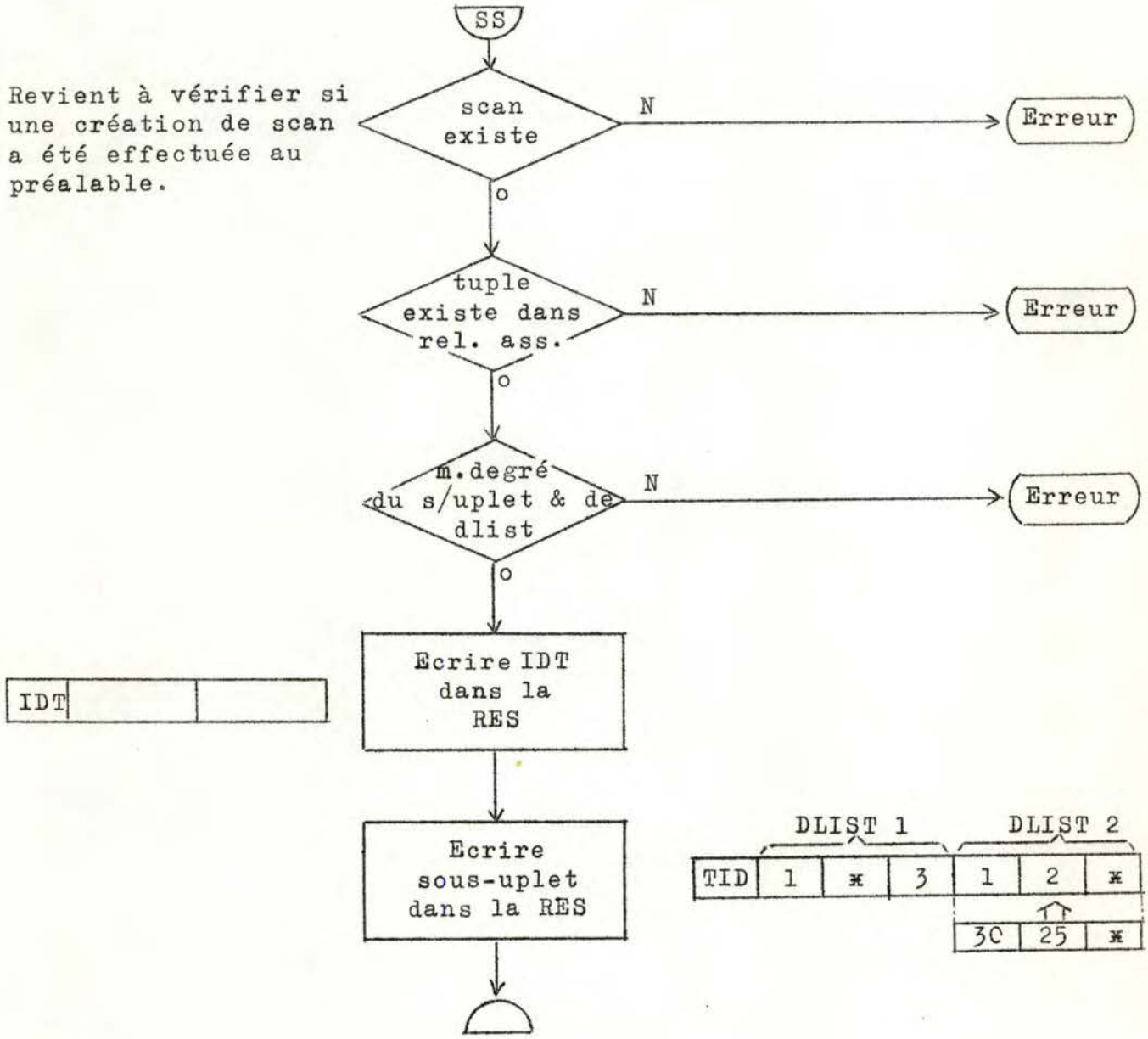

SS
Revient à vérifier si une création de scan a été effectuée au préalable.
scan existe
N
Erreur
o
tuple existe dans rel. ass.
N
Erreur
o
m.degré du s/uplet & de dlist
N
Erreur
o
Ecrire IDT dans la RES
IDT
Ecrire sous-uplet dans la RES
DLIST 1
DLIST 2
TID
1
*
3
1
2
*
30
25
*

Figure 9

DLIST 1 $\equiv \{D_1, D_2, \ldots D_n\}$

DLIST 2 $\equiv \{F_1, F_2, \ldots F_n\}$

UPLET $\equiv \{I_1, \ldots . I_n\}$

$\forall$ i, si $F_i = *$, le test n'est pas effectué, on passe directement à $F_i + 1$. $F_1 = *$ signifie que le domaine n'est pas soumis à un filtre.

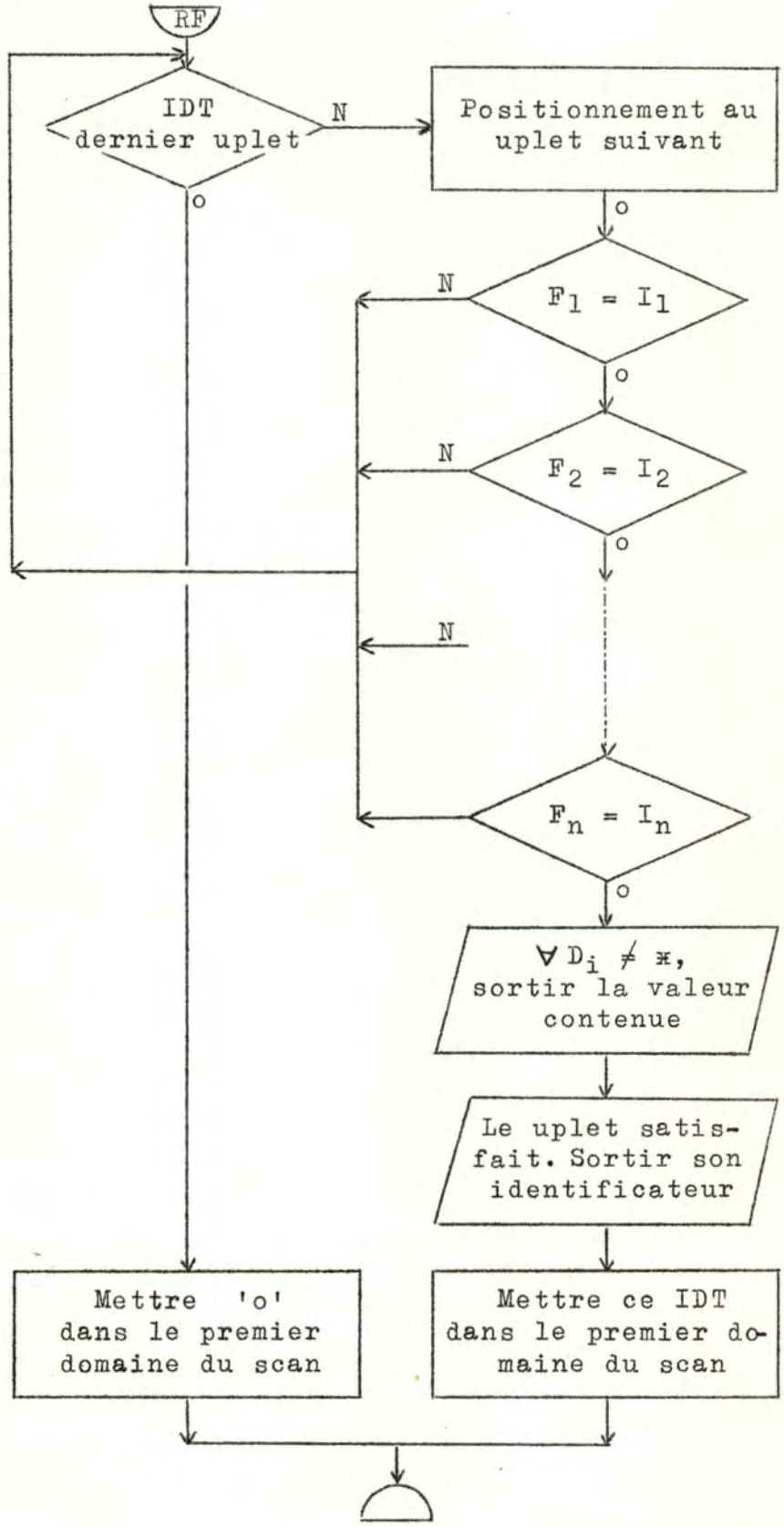
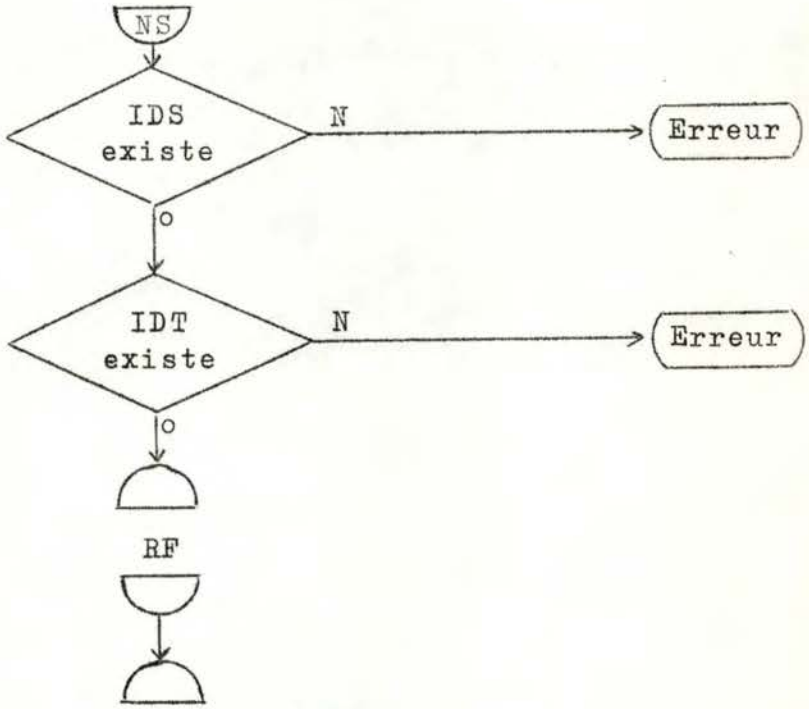


Figure 10



RF : recherche filtrée

Figure 11

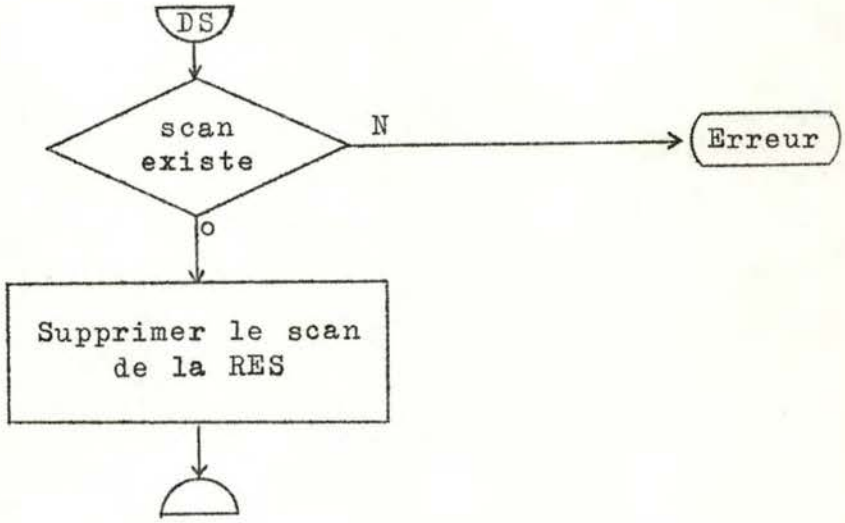


Figure 12

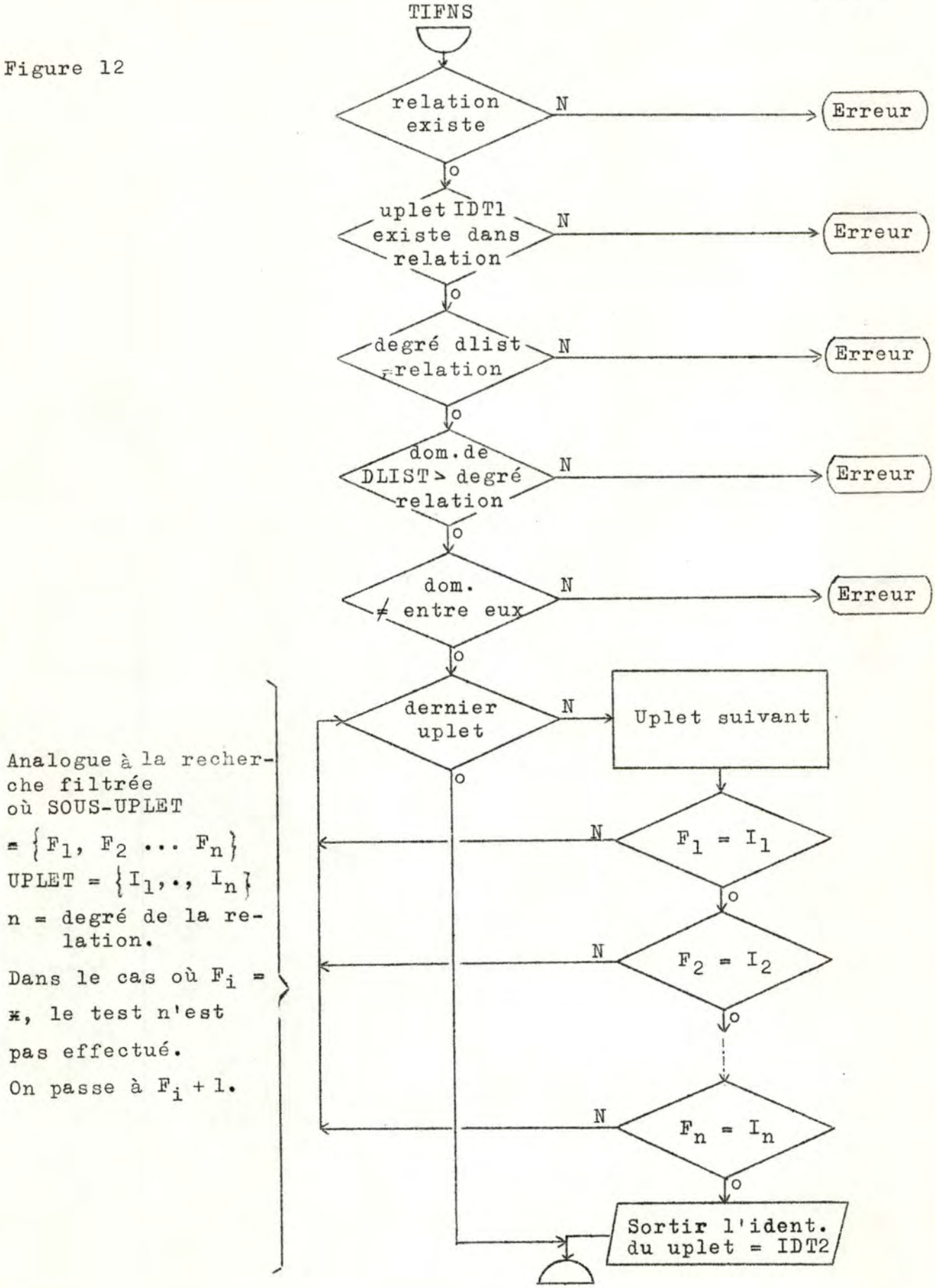
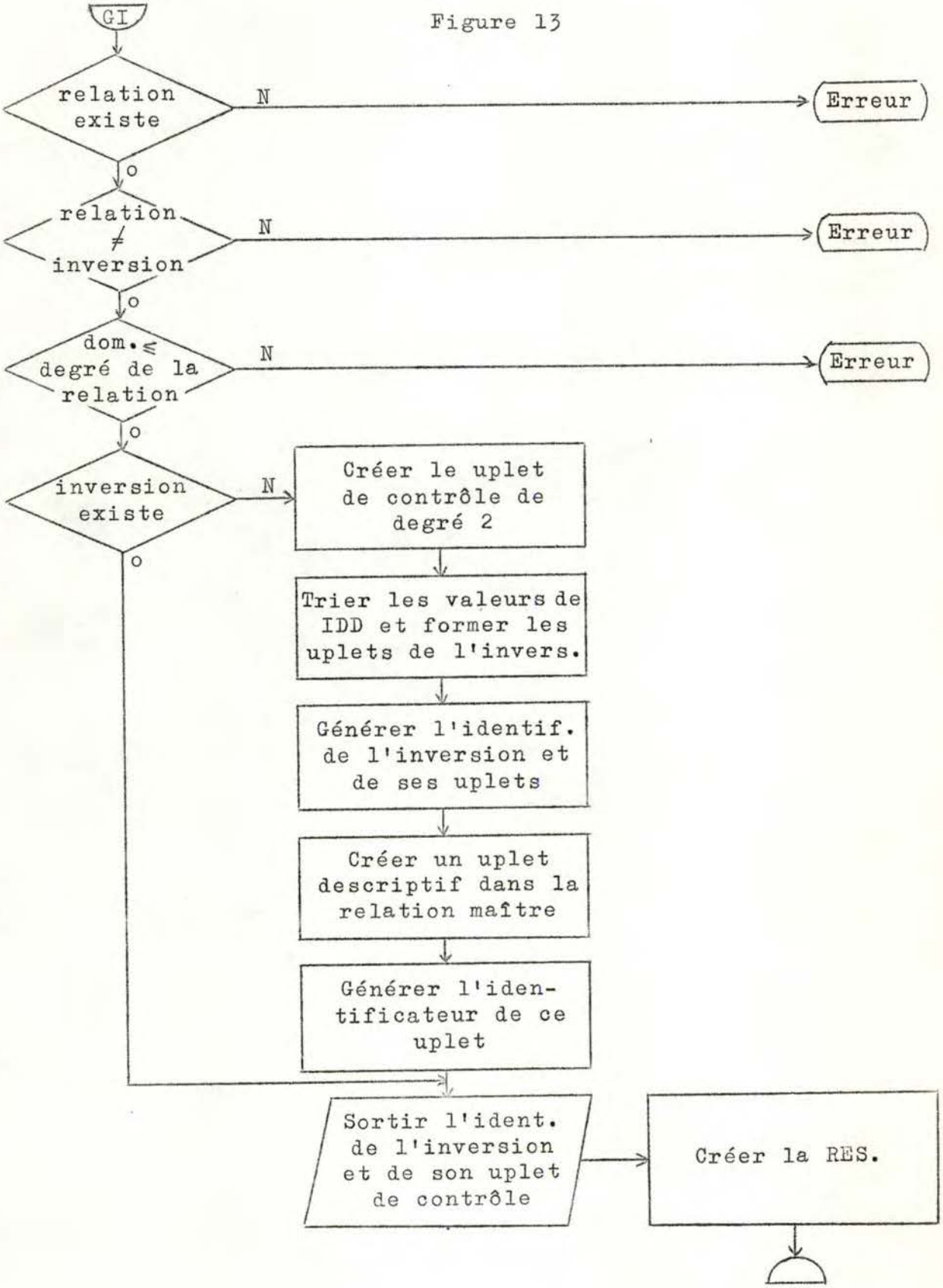

TIFNS
relation existe
N
Erreur
o
uplet IDT1 existe dans relation
N
Erreur
o
degré dlist = relation
N
Erreur
o
dom. de DLIST > degré relation
N
Erreur
o
dom. ≠ entre eux
N
Erreur
o
dernier uplet
N
Uplet suivant
o
N
$F_1 = I_1$
o
N
$F_2 = I_2$
o
N
$F_n = I_n$
o
Sortir l'ident. du uplet = IDT2
Analogue à la recherche filtrée où SOUS-UPLET = $\{F_1, F_2 \ldots F_n\}$
UPLET = $\{I_1, \ldots, I_n\}$
n = degré de la relation.
Dans le cas où F_i = *, le test n'est pas effectué.
On passe à $F_i + 1$.

Figure 13


GI
relation existe
N
Erreur
O
relation ≠ inversion
N
Erreur
O
dom. ≤ degré de la relation
N
Erreur
O
inversion existe
N
O
Créer le uplet de contrôle de degré 2
Trier les valeurs de IDD et former les uplets de l'invers.
Générer l'identif. de l'inversion et de ses uplets
Créer un uplet descriptif dans la relation maître
Générer l'identificateur de ce uplet
Sortir l'ident. de l'inversion et de son uplet de contrôle
Créer la RES.

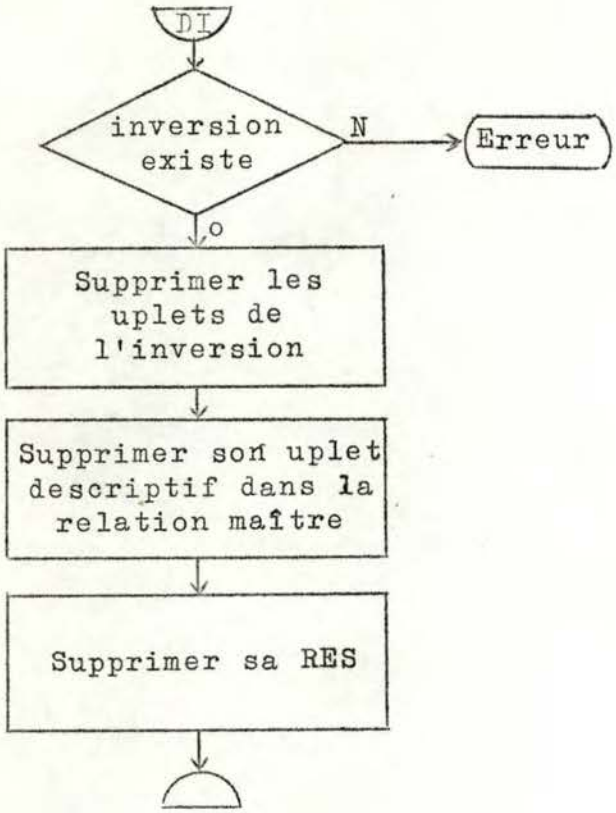
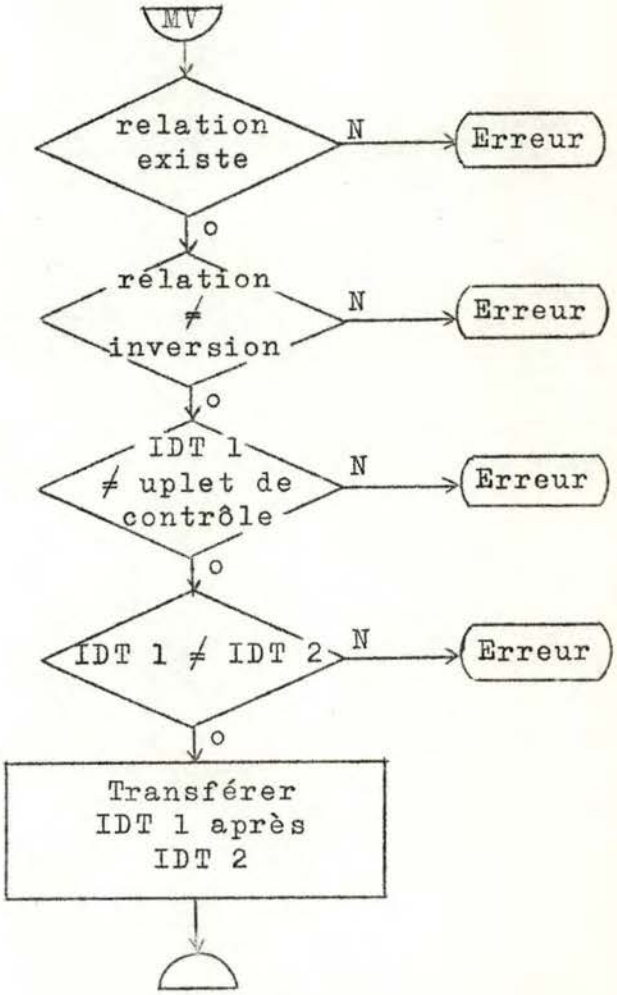


Figure 14

Figure 15



LISTE DES MODULES PROGRAMMES

MODULE MAITRE : GAMMA-O : Il détermine l'appel d'un des modules principaux, en fonction de la commande reçue.

MODULES PRINCIPAUX :

Nom du module	Fonction
CRREL	Création d'une relation régulière ou classe.
DREL	Suppression d'une relation régulière ou classe.
INSTU	Insertion d'un uplet dans une relation régulière ou classe.
DELTUP	Suppression d'un uplet dans une relation régulière ou classe.
MAJST	Mise à jour d'un sous-uplet dans une relation régulière ou classe.
CRSCA	Création d'un scan.
ETSCA	Etablissement d'un scan.
NESCA	Scan suivant.
SUPSCA	Suppression d'un scan.
GINV	Génération d'une inversion.
DINV	Suppression d'une inversion.

MODULES SECONDAIRES.

I.- MANIPULATION DE FICHIERS.

Nom du module	Fonction
MANMAS	Manipulation du master.
MANREG	Manipulation de relations régulières.
MANCL	Manipulation de relations classes.
MANINV	Manipulation d'inversions.
MANSSRM	Manipulation de la relation d'état de scan du master.
MANSSRR	Manipulation de relation d'état de scan de relations régulières.
MANSSRC	Manipulation de relation d'état de scan de relations classes.
MANSSRI	Manipulation de relation d'état de scan d'inversions.

II.- MODULES DE RECHERCHE.

Nom du module	Fonction
EXREL	Recherche de l'existence d'une relation.
EXINV	Recherche de l'existence d'inversions pour relations régulières
RF	Recherche filtrée pour relations régulières et classes.

FIGURE N° 16

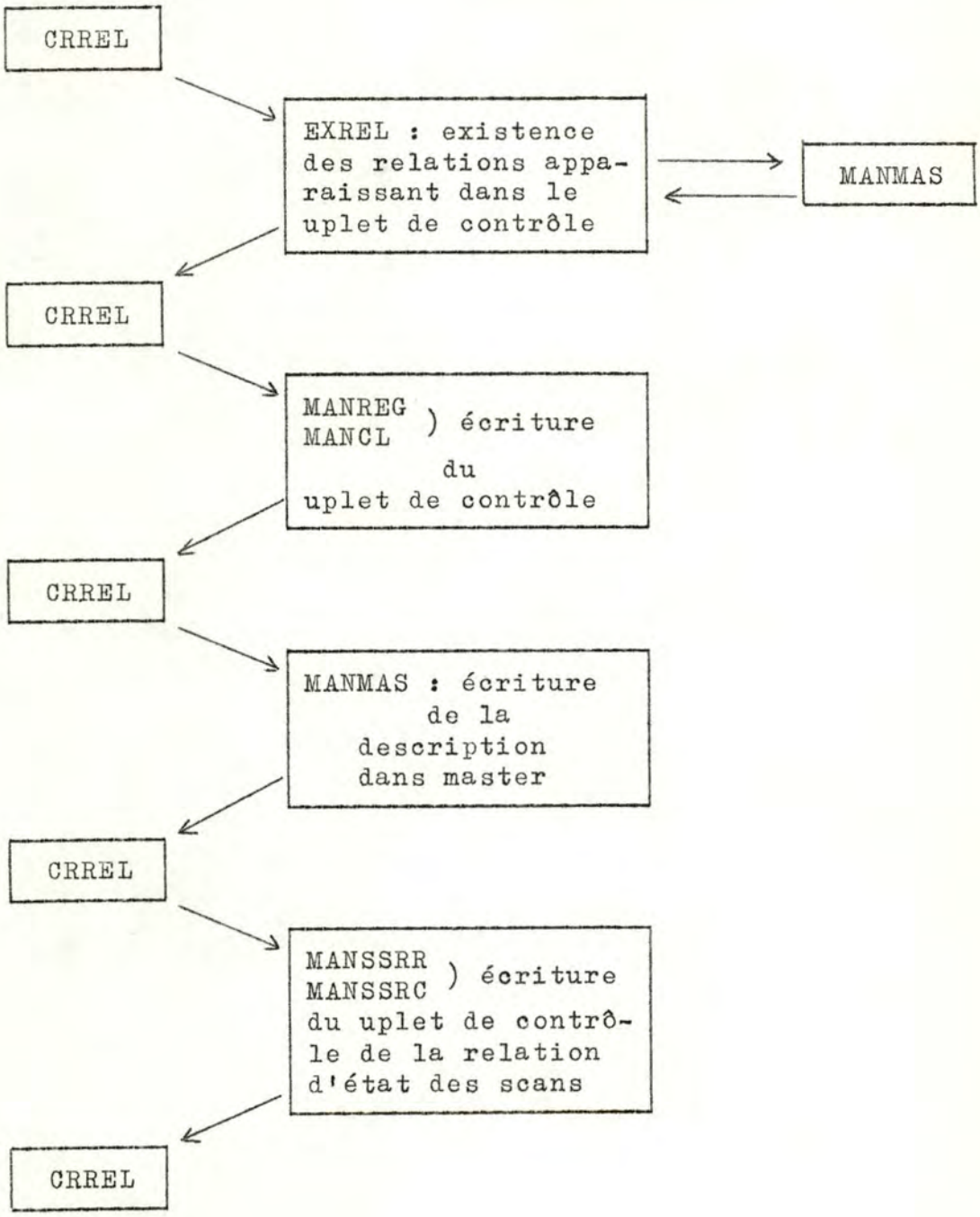

CRREL
EXREL : existence des relations apparaissant dans le uplet de contrôle
MANMAS
CRREL
MANREG
MANCL
) écriture du uplet de contrôle
CRREL
MANMAS : écriture de la description dans master
CRREL
MANSSRR
MANSSRC
) écriture du uplet de contrôle de la relation d'état des scans
CRREL

FIGURE N° 17.

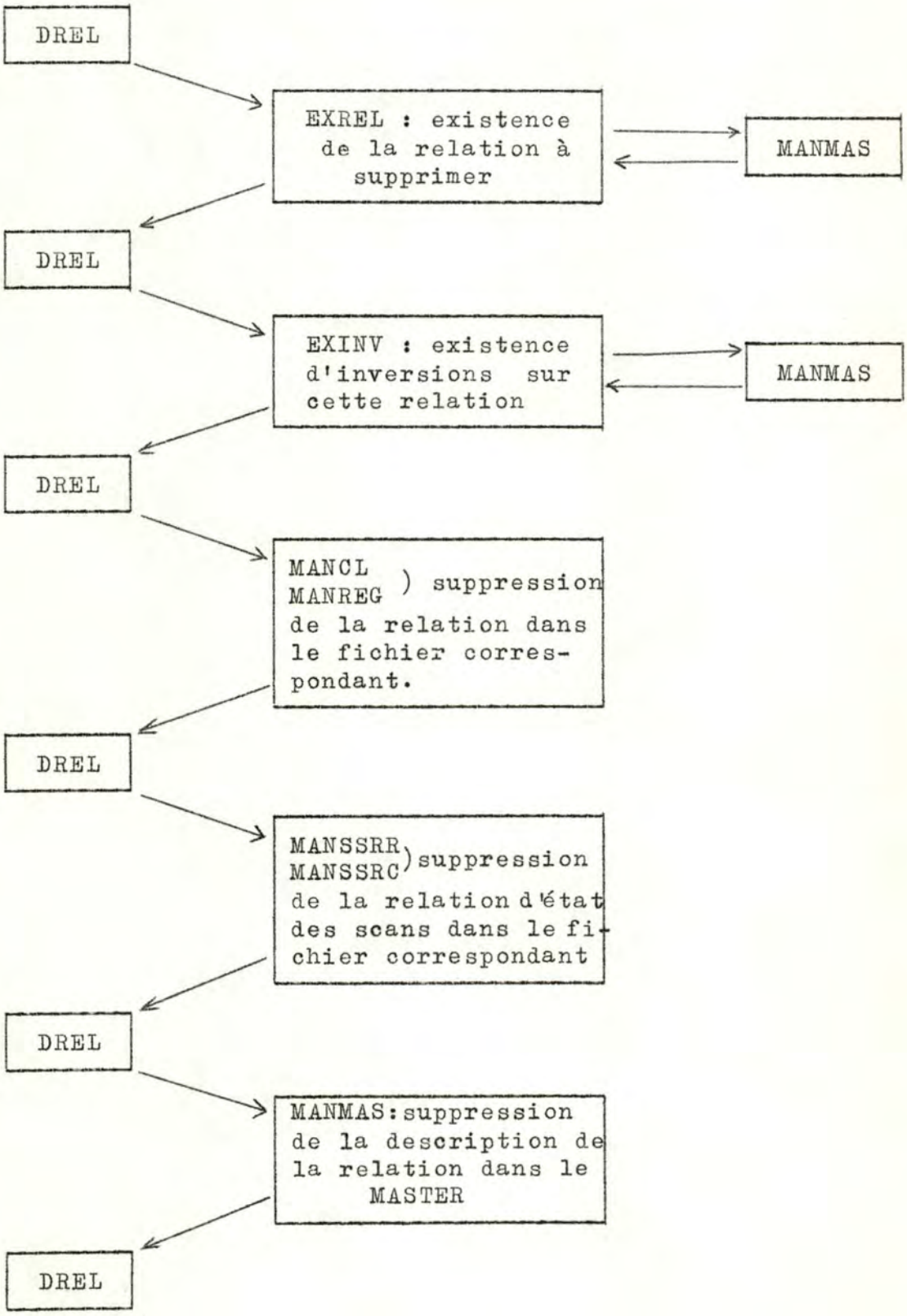

DREL
EXREL : existence de la relation à supprimer
MANMAS
DREL
EXINV : existence d'inversions sur cette relation
MANMAS
DREL
MANCL
MANREG) suppression de la relation dans le fichier correspondant.
DREL
MANSSRR
MANSSRC) suppression de la relation d'état des scans dans le fichier correspondant
DREL
MANMAS: suppression de la description de la relation dans le MASTER
DREL

FIGURE N° 18.

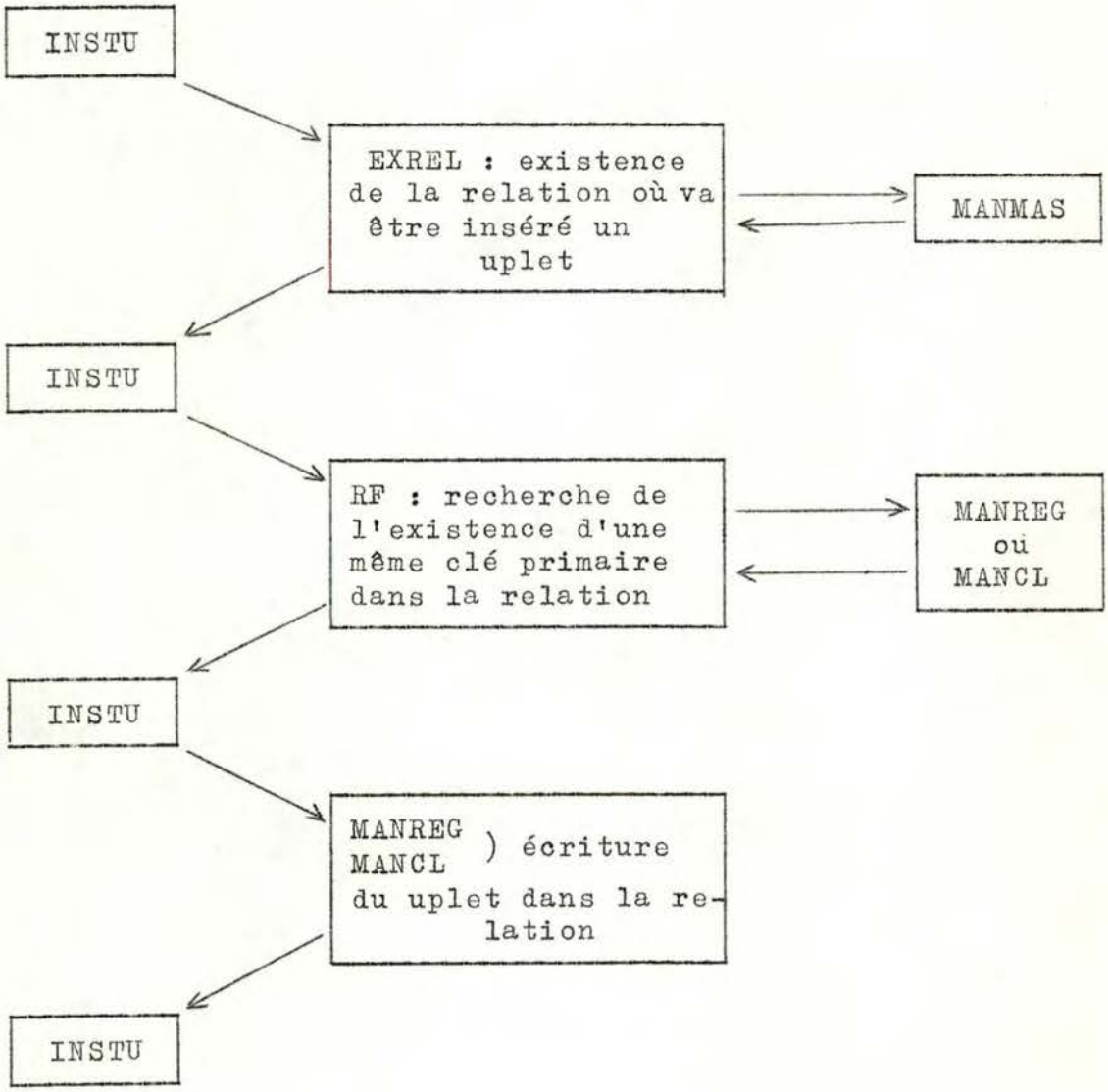

INSTU
EXREL : existence de la relation où va être inséré un uplet
MANMAS
INSTU
RF : recherche de l'existence d'une même clé primaire dans la relation
MANREG ou MANCL
INSTU
MANREG MANCL) écriture du uplet dans la relation
INSTU

FIGURE N° 19

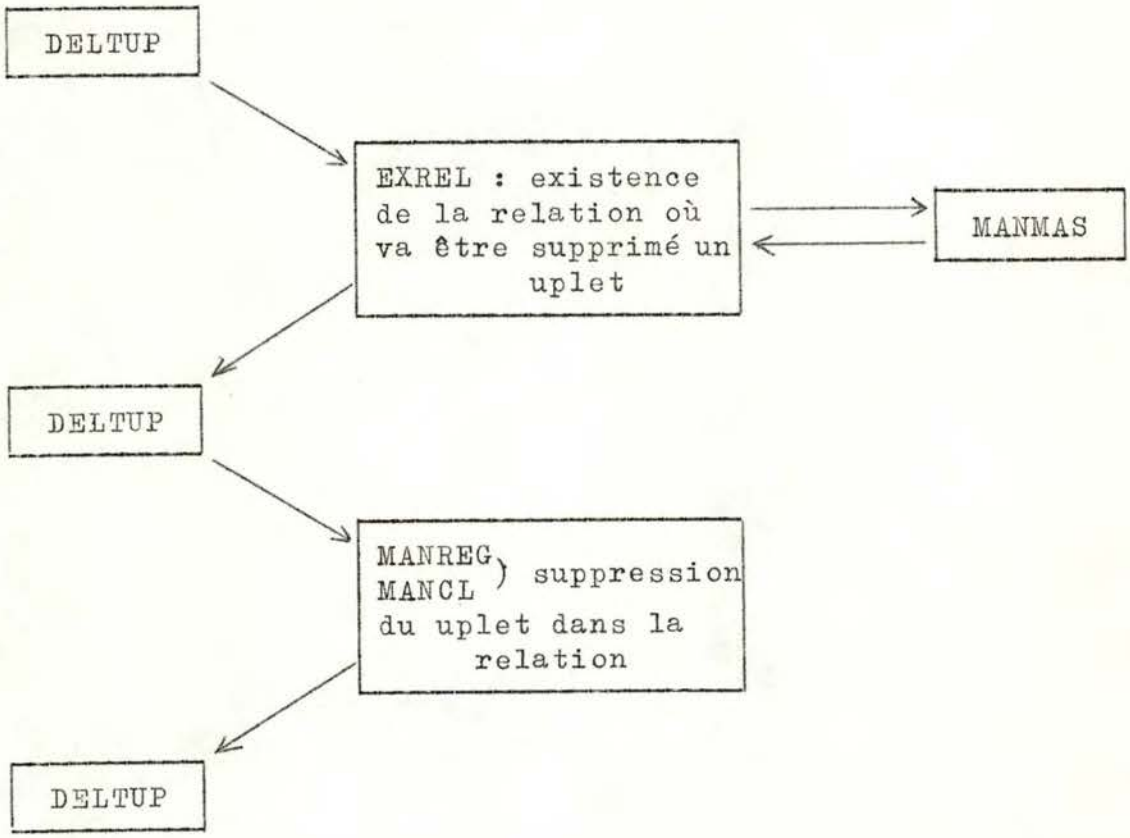

DELTUP
EXREL : existence de la relation où va être supprimé un uplet
MANMAS
DELTUP
MANREG
MANCL
suppression du uplet dans la relation
DELTUP

FIGURE N° 20

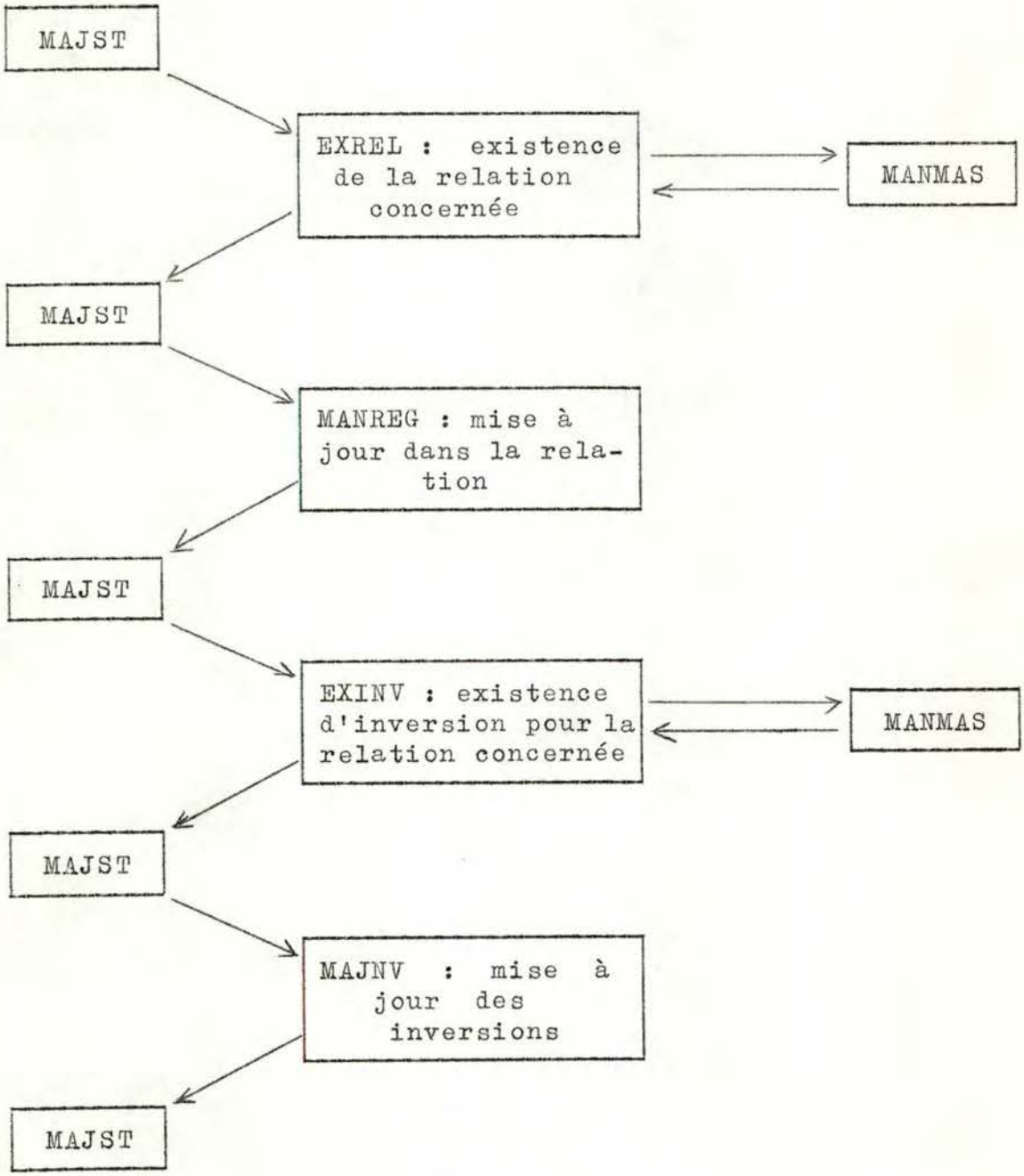

MAJST
EXREL : existence de la relation concernée
MANMAS
MAJST
MANREG : mise à jour dans la relation
MAJST
EXINV : existence d'inversion pour la relation concernée
MANMAS
MAJST
MAJNV : mise à jour des inversions
MAJST

FIGURE N° 21

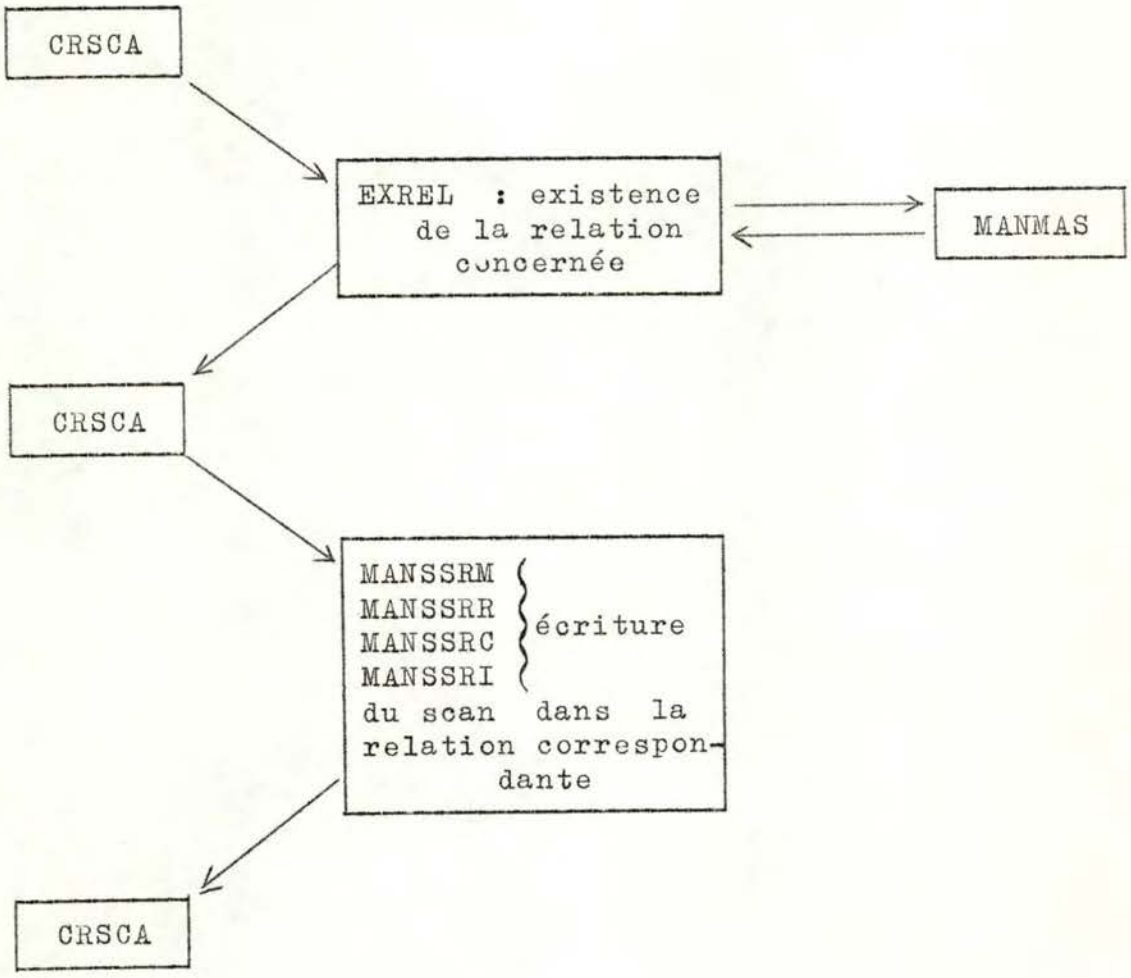

CRSCA
EXREL : existence de la relation concernée
MANMAS
CRSCA
MANSSRM
MANSSRR
MANSSRC
MANSSRI
écriture
du scan dans la relation correspondante
CRSCA

FIGURE N° 22

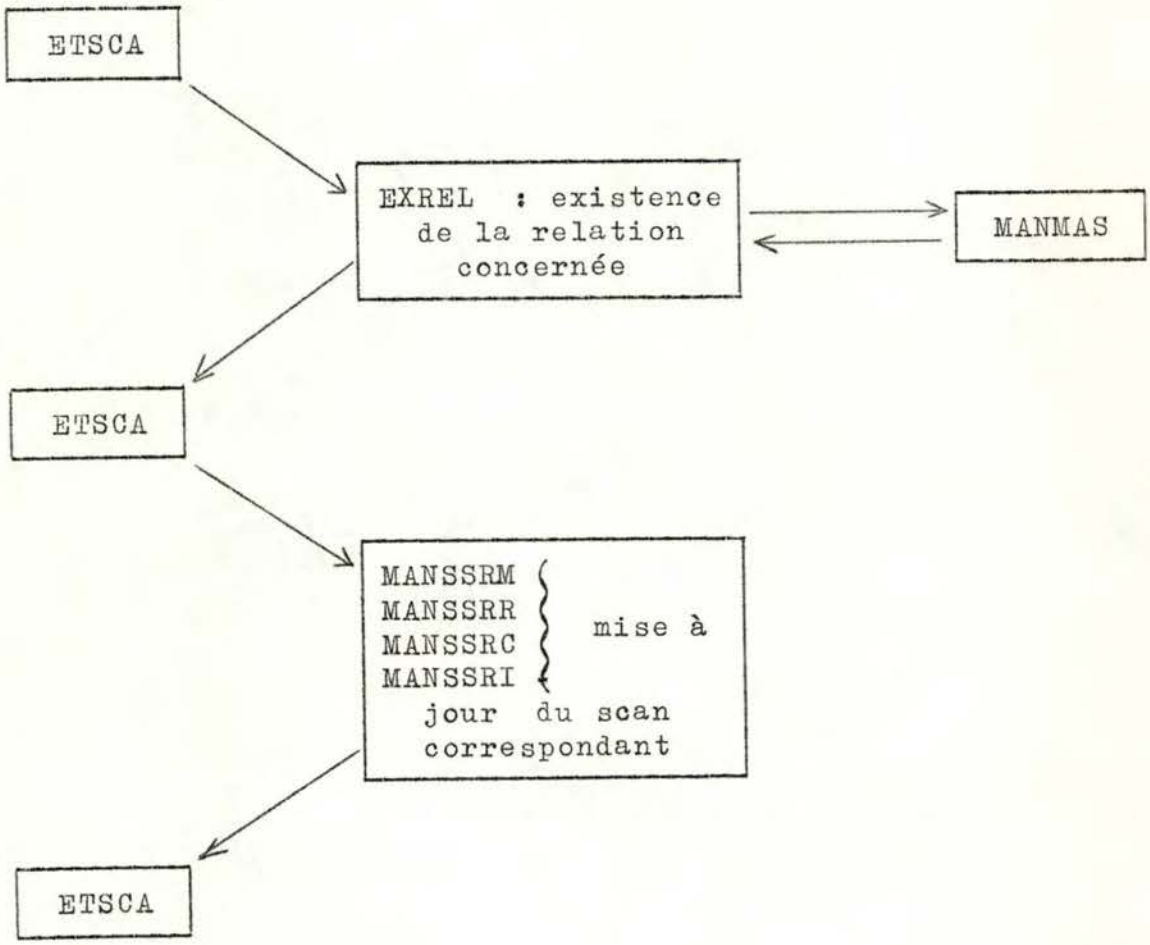

ETSCA
EXREL : existence de la relation concernée
MANMAS
ETSCA
MANSSRM
MANSSRR
MANSSRC
MANSSRI
mise à jour du scan correspondant
ETSCA

FIGURE N° 23

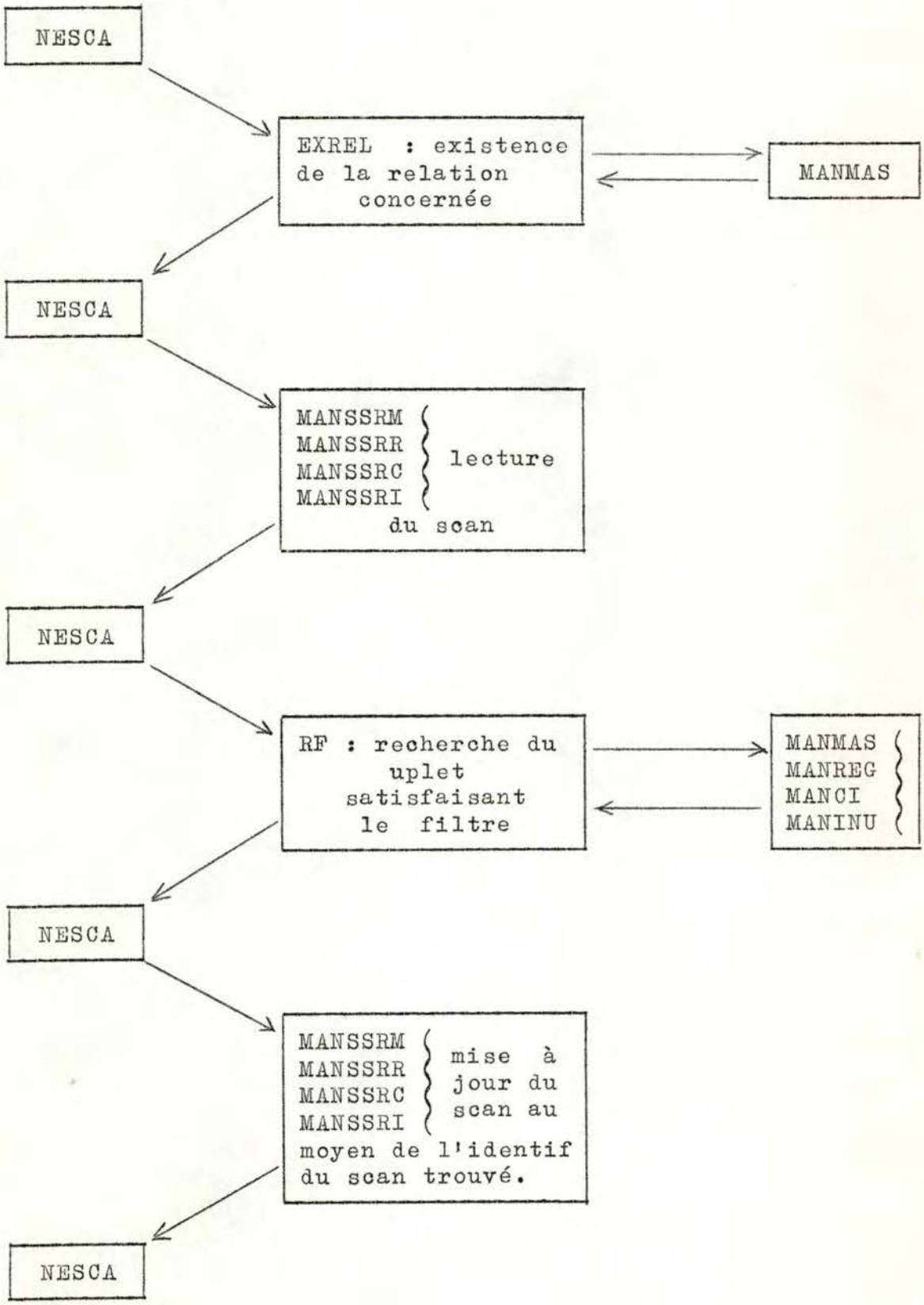

NESCA
EXREL : existence de la relation concernée
MANMAS
NESCA
MANSSRM
MANSSRR
MANSSRC
MANSSRI
lecture du scan
NESCA
RF : recherche du uplet satisfaisant le filtre
MANMAS
MANREG
MANCI
MANINU
NESCA
MANSSRM
MANSSRR
MANSSRC
MANSSRI
mise à jour du scan au moyen de l'identif du scan trouvé.
NESCA

FIGURE N° 24

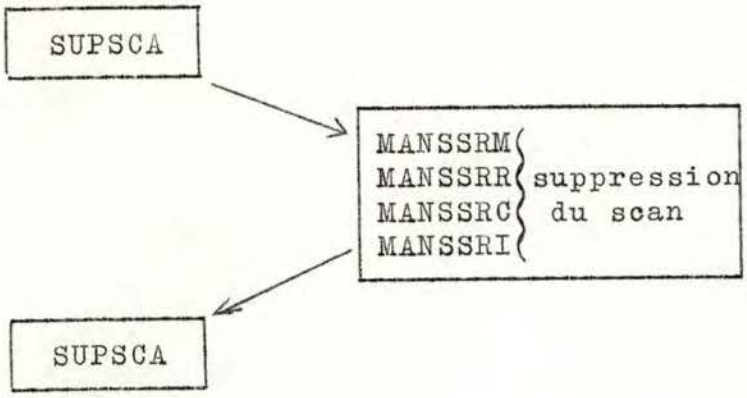


FIGURE N° 25

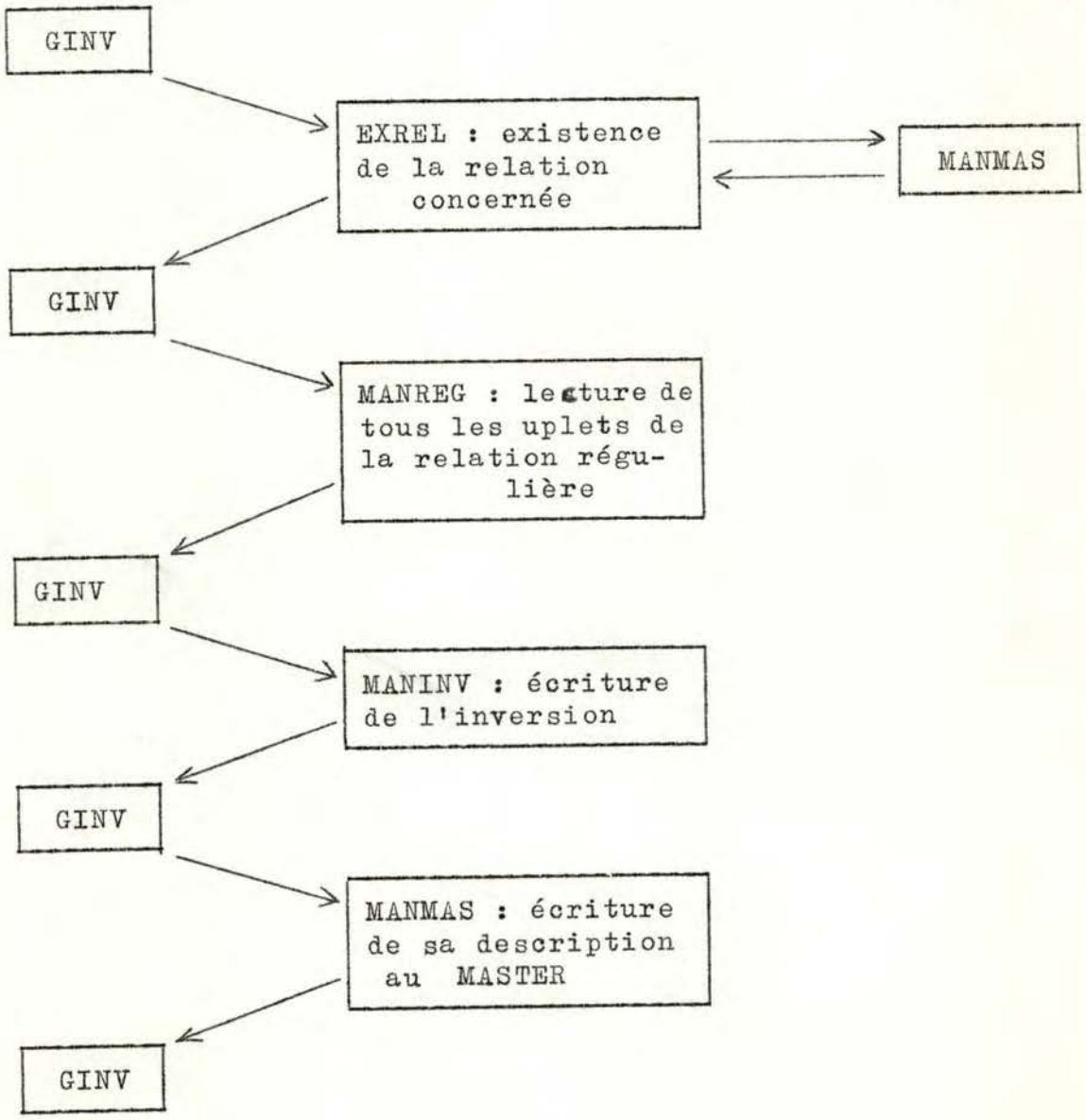
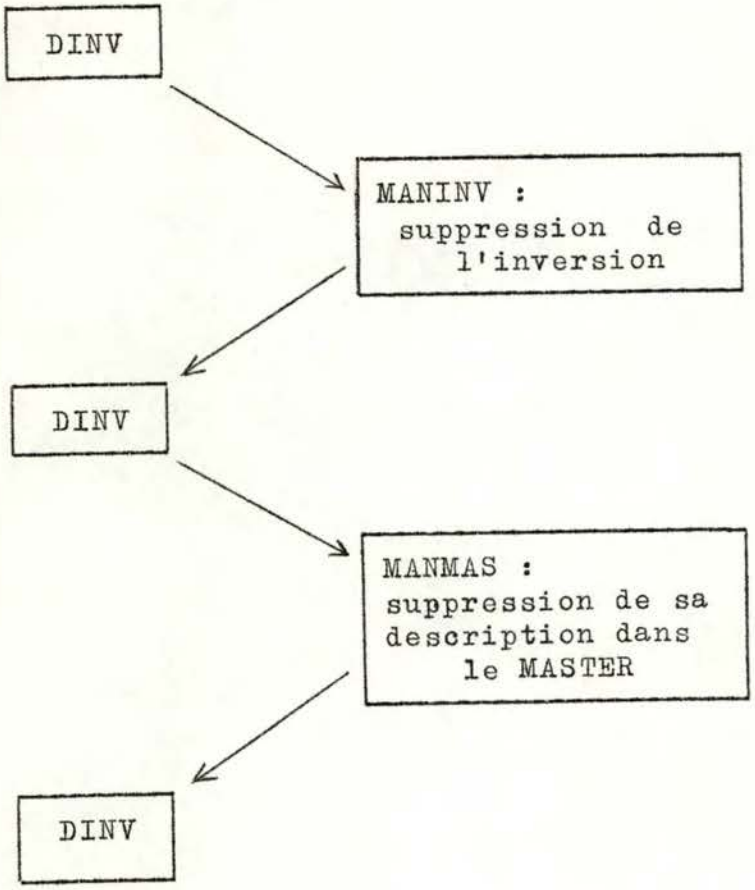


FIGURE N° 26


DINV
MANINV :
suppression de
l'inversion
DINV
MANMAS :
suppression de sa
description dans
le MASTER
DINV

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

BROWNE, P.S. : Data privacy and integrity : an overview.

BROWNE, P.S., STEINAUER, D.D.

: A model fer access control,
Headquarters strategic air command
Deputy Chief of Staff fer Operations
Offut AFB, Nebraska.

CHANBERLIN, D.D., GRAY, J.N., TRAIGER, I.L.

: Views, authorization and locking in a relational data base system.

IBM Research Laboratory.
San Jose, California, october 7, 1974.

CHAMBERLIN, D.D., BOYCE, R.F.

: Sequel : a structured englich query langage.

May 24, 1974 - IBM Research.

CODASYL : Data base task group report, ACM New-York (1971).

CODD, E.F. : A data base sublangage founded on the relational calculus.

IBM Research, July 26, 1971.

CODD, E.F. : A relational model for large shared data banks.

CACM vol. 13, n° 6 pp 377-387 (June 1970).

CODD, E.F. : Seven steps to rendez-vous with the casual user.

IBM Research Report, RJ 1333, January 1974.

CODD, E.F. : Further normalization of the data base relational model.

Courant computer science symposia, vol. 6, Data base systems, Prentice-Hall, New-York, May 1971.

CODD, E.F. : Relational completeness of data base sublangage.
Courant computer science symposia, vol. 6, Data base systems, Prentice-Hall, New-York, May 1971.

CODD, E.F. : A relational Algebra for data base system.
IBM Research, Technical report.

DELOBEL, C. : Contributions théoriques à la conception et l'évaluation d'un système d'information appliqué à la gestion.
Université Scientifique et Médicale de Grenoble, octobre 1973.

DELOBEL, C. : Systèmes de bases de données.
Université Scientifique et Médicale de Grenoble.
Cours MIAG 2e année, 1974-75.

ESWAREN, K.P., GRAY, J.N., LORIE, R.A. TRAIGER, I.L.
: On the notions on consistancy and predicate locks in a data base system.
IBM Research Laboratory, San José, California, 95193.

LORIE, R.A. : XRM - An extended (n - ary) relational memory.
IBM cambridge Scientific Center.
Technical report nr 320-2096 January 1974.

Rapport GAMMA - O (Notes de Monsieur Claude DEHENEFFE)

(I) Alain PIROTTE : Thèse de doctorat (notes personnelles de Monsieur Claude DEHENEFFE)