Projet SOCRATE (2-1) : Langage de requêtes

Georges Vigliano

N° D'ORDRE

THESE

présentée à

LA FACULTE DES SCIENCES DE L'UNIVERSITE DE GRENOBLE

Pour Obtenir

LE GRADE DE DOCTEUR DE TROISIEME CYCLE
"Mathématiques Appliquées"

par

Georges VIGLIANO

PROJET SOCRATE

(2-1) Langage de Requêtes

Thèse soutenue le 22 Décembre 1970 devant la commission d'examen

Monsieur	J. KUNTZMANN	Président
Monsieur	L. BOLLIET	Examinateurs
Monsieur	N. GASTINEL	
Monsieur	J.C. BOUSSARD	
Monsieur	F. SALLE	

L I S T E D E S P R O F E S S E U R S

Doyen honoraire : Monsieur M. MORET
Doyen : Monsieur E. BONNIER

PROFESSEURS TITULAIRES

MM.	NEEL Louis	Physique Expérimentale
	KRAVTCHENKO Julien	Mécanique Rationnelle
	CHABAUTY Claude	Calcul différentiel et intégral
	BENOIT Jean	Radioélectricité
	CHENE Marcel	Chimie Papetière
	FELICI Noël	Electrostatique
	KUNTZMANN Jean	Mathématiques Appliquées
	BARBIER Reynold	Géologie Appliquée
	SANTON Lucien	Mécanique des Fluides
	OZENDA Paul	Botanique
	FALLOT Maurice	Physique Industrielle
	KOSZUL Jean-Louis	Mathématiques
	GALVANI Octave	Mathématiques
	MOUSSA André	Chimie Nucléaire
	TRAYNARD Philippe	Chimie Générale
	SOUTIF Michel	Physique Générale
	CRAYA Antoine	Hydrodynamique
	REULOS René	Théorie des Champs
	BESSON Jean	Chimie Minérale
	AYANT Yves	Physique Approfondie
	GALLISSOT François	Mathématiques
Melle.	LUTZ Elisabeth	Mathématiques
MM.	BLAMBERT Maurice	Mathématiques
	BOUCHEZ Robert	Physique Nucléaire
	LLIBOUTRY Louis	Géophysique
	MICHEL Robert	Minéralogie et pétrographie
	BONNIER Etienne	Electrochimie et Electrométallurgie
	DESSAUX Georges	Physiologie animale
	PILLET Emile	Physique Industrielle-Electrotechnique
	YOCCOZ Jean	Physique Nucléaire théorique
	DEBELMAS Jacques	Géologie Générale
	GERBER Robert	Mathématiques
	PAUTHENET René	Electrotechnique
	MALGRANGE Bernard	Mathématiques Pures
	VAUQUOIS Bernard	Calcul Electronique
	BARJON Robert	Physique Nucléaire

MM.	BARBIER Jean-Claude	Physique
	SILBER Robert	Mécanique des Fluides
	BUYLE-BODIN Maurice	Electronique
	DREYFUS Bernard	Thermodynamique
	KLEIN Joseph	Mathématiques
	VAILLANT François	Zoologie et Hydrobiologie
	ARNAUD Paul	Chimie
	SENGEL Philippe	Zoologie
	BARNOUD Fernand	Biosynthèse de la Cellulose
	BRISSONNEAU Pierre	Physique
	GAGNAIRE Didier	Chimie Physique
Mme.	KOFLER Lucie	Botanique
MM.	DEGRANGE Charles	Zoologie
	PEBAY-PEROULA Jean-Claude	Physique
	RASSAT André	Chimie Systématique
	DUCROS Pierre	Cristallographie Physique
	DODU Jacques	Mécanique Appliquée I. U. T.
	ANGLES D'AURIAC Paul	Mécanique des Fluides
	LACAZE Albert	Thermodynamique
	GASTINEL Noël	Analyse numérique
	GIRAUD Pierre	Géologie
	PERRET René	Servo-mécanisme
	PAYAN Jean-Jacques	Mathématiques Pures

PROFESSEURS SANS CHAIRE

MM.	GIDON Paul	Géologie
Mme.	BARBIER Marie-Jeanne	Electrochimie
Mme.	SOUTIF Jeanne	Physique
	COHEN Joseph	Electrotechnique
	DEPASSEL R.	Mécanique des Fluides
	GLENAT René	Chimie
	BARRA Jean	Mathématiques Appliquées
	COUMES André	Electronique
	PERRIAUX Jacques	Géologie et Minéralogie
	ROBERT André	Chimie Papetière
	BIARREZ Jean	Mécanique Physique
	BONNET Georges	Electronique
	CAUQUIS Georges	Chimie Générale
	BONNETAIN Lucien	Chimie Minérale
	DEPOMIER Pierre	Physique Nucléaire-Génie Atomique
	HACQUES Gérard	Calcul numérique
	POLOUJADOFF Michel	Electrotechnique
Mme.	KAHANE Josette	Physique
Mme.	BONNIER Jane	Chimie
MM.	VALENTIN Jacques	Physique
	REBECQ Jacques	Biologie
	DEPORTES Charles	Chimie
	SARROT-REYNAULD Jean	Géologie
	BERTRANDIAS Jean-Paul	Mathématiques Appliquées
	AUBERT Guy	Physique

PROFESSEURS ASSOCIES

MM.	RODRIGUES Alexandre	Mathématiques Pures
	MORITA Susumu	Physique Nucléaire
	RADHAKRISHNA	Thermodynamique

MAITRES DE CONFERENCES

MM.	LANCIA Roland	Physique Atomique
Mme.	BOUCHE Liane	Mathématiques
MM.	KAHANE André	Physique Générale
	DOLIQUE Jean Michel	Electronique
	BRIERE Georges	Physique
	DESRE Georges	Chimie
	LAJZEHOWICZ Joseph	Physique
	LAURENT Pierre	Mathématiques Appliquées
Mme.	BERTRANDIAS Françoise	Mathématiques Pures
MM.	LONGEQUEUE Jean-Pierre	Physique
	SOHM Jean-Claude	Electrochimie
	ZADWORNY François	Electronique
	DURAND Francis	Chimie Physique
	CARLIER Georges	Biologie végétale
	PFISTER Jean-Claude	Physique
	CHIBON Pierre	Biologie animale
	IDELMAN Simon	Physiologie animale
	BLOCH Daniel	Electrotechnique I. P.
	MARTIN-BOUYER Michel	Chimie (C. S. U. Chambéry)
	SIBILLE Robert	Construction mécanique (I. U. T.)
	BRUGEL Lucien	Energétique I. U. T.
	BOUVARD Maurice	HYdrologie
	RICHARD Lucien	Botanique
	PELMONT Jean	Physiologie animale
	BOUSSARD Jean-Claude	Mathématiques Appliquées (I. P. G.)
	MOREAU René	Hydraulique I. P. G.
	ARMAND Yves	Chimie I. U. T.
	BOLLIET Louis	Informatique I. U. T.
	KUHN Gérard	Energétique I. U. T.
	PEFFEN René	Chimie I. U. T.
	GERMAIN Jean-Pierre	Mécanique
	JOLY Jean-René	Mathématiques Pures
Melle.	PIERY Yvette	Biologie animale
	BERNARD Alain	Mathématiques Pures
	MOHSEN Tahsin	Biologie (C. S. U. Chambéry)
	CONTE René	Mesures Physiques I. U. T.
	LE JUNTER Noël	Génie Electrique Electronique I. U. T.
	LE ROY Philippe	Génie Mécanique I. U. T.
	ROMIER Guy	Techniques Statistiques quantitatives I. U. T.
	VIALON Pierre	Géologie
	BENZAKEN Claude	Mathématiques Appliquées
	MAYNARD Roger	Physique

MM.	DUSSAUD René	Mathématiques (C. S. U. Chambéry)
	BELORIZKY Elie	Physique (C. S. U. Chambéry)
Mme.	LAJZEROWICZ Jeannine	Physique (C. S. U. Chambéry)
M.	JULLIEN Pierre	Mathématiques Pures
Mme.	RINAUDO Marguerite	Chimie
MM.	BLIMAN Samuel	E. I. E.
	BEGUIN Claude	Chimie Organique
	NEGRE Robert	I. U. T.

MAITRES DE CONFERENCES ASSOCIES

MM.	YAMADA Osamu	Physique du Solide
	NAGAO Makoto	Mathématiques Appliquées
	MAREZIO Massimo	Physique du Solide
	CHEECKE John	Thermodynamique
	BOUDOURIS Georges	Radioélectricité
	ROZMARIN Georges	Chimie Papetière

Je tiens à exprimer ma profonde gratitude envers :

Monsieur Jean KUNTZMANN, qui a bien voulu me faire l'honneur de présider le Jury de cette thèse,

Monsieur Noël GASTINEL, qui a accepté de faire partie de ce Jury,

Monsieur Jean-Claude BOUSSARD, dont j'ai toujours eu plaisir à suivre l'enseignement et les conseils,

Monsieur Louis BOLLIET, qui a toujours montré pour nos travaux un intérêt bienveillant,

Monsieur François SALLÉ, qui a bien voulu les juger.

Je suis surtout heureux de pouvoir remercier ici, Monsieur Jean-Raymond ABRIAL, qui, après avoir donné naissance au projet SOCRATE, a su communiquer à toute notre équipe l'esprit qui lui a permis de mener à bien ce projet.

Je veux associer dans les mêmes remerciements tous les membres de cette équipe Messieurs Jacques BAS, Georges BEAUME, Gérard HENNERON, Robert MORIN, pour ne citer que les plus anciens.

Tous ont contribué par leur gentillesse et leur compétence à créer l'ambiance amicale sans laquelle notre travail n'aurait pu s'accomplir. Encore une fois, je rends hommage à celui qui, par son enthousiasme communicatif, a su créer ce climat enrichissant de travail et d'amitié.

La réalisation du Système, que nous présentons ici, doit beaucoup aux remarques de ceux qui ont accepté d'en être les utilisateurs "cobayes". Qu'ils en soient tous sincèrement remerciés, en particulier le Docteur VALOIS pour la confiance qu'il nous a manifestée en toutes circonstances.

Je remercie aussi tous mes collègues du Laboratoire de Calcul qui nous ont apporté une aide patiente et amicale.

Enfin, je remercie tous ceux qui ont contribué à la réalisation technique de cet ouvrage. Que Madame TREVISAN accepte mes remerciements pour le soin qu'elle a apporté à rendre présentable cet ouvrage.

A MES PARENTS,

L'exposé qui suit est un complément aux spécifications générales du projet SOCRATE.

Les définitions données dans les spécifications générales sont supposées connues.

Les références à cet ouvrage sont notées [1].

Les autres références du projet SOCRATE sont notées :

[2.1.] *Langage de Requêtes*

[2.2.] *Gestion des mémoires*

[2.3.] *Interprétation et Compilation.*

L'ensemble de ces quatre ouvrages constitue une thèse de groupe.

o0o0o0o0o0o0o0o0o0o

S O M M A I R E

INTRODUCTION

Le système SOCRATE a été conçu initialement pour gérer une banque de données où l'utilisateur range ou puise des informations.

Si un utilisateur envisage d'intégrer le système SOCRATE à la vie même de son entreprise, quelle qu'elle soit, il est en droit d'en exiger une aide plus intelligente : les informations restituées par le Système devront l'être sous une forme plus élaborée qu'à leur entrée. Ainsi " SOCRATE " ne sera plus une simple banque de données mais plutôt une usine de transformation d'informations.

Prenons un exemple pour montrer cette évolution. Supposons qu'un utilisateur veuille confier la gestion d'un stock de produits à SOCRATE. Cet utilisateur se contentera d'indiquer chaque jour pour chaque produit la quantité écoulée et le cas échéant, la quantité réapprovisionnée. Il attendra du Système que celui-ci calcule, spontanément lors de chaque mise à jour d'un produit, la quantité restante de ce produit. Il peut même espèrer que le système lui signale spontanément encore, le fait que la quantité d'un produit atteint le seuil critique où il devient nécessaire de le réapprovisionner. Ce que nous appelons ici des actions spontanées du système sont des traitements particuliers qu'il devra effectuer sur un fichier en fonction de certaines conditions, ces conditions et ces traitements associés étant préalablement définis par l'utilisateur.

Enfin, le dernier besoin exigé pour un "Système Intégré" est la possibilité de modification de structure ([1] § 6.3.3 à 6.3.5.). En effet, au bout d'un certain temps d'apprentissage, l'utilisateur peut vouloir soit modifier les traitements particuliers déjà définis, soit même modifier profondément la structure du fichier sans perdre les informations recueillies jusqu'alors.

./.

Les trois points sur lesquels on doit porter des améliorations au système sont donc :

1) Calcul de bureau
2) Actions spontanées
3) Modification de structure.

Nous verrons les solutions qu'il est possible d'apporter à ces problèmes dans les chapitres suivants. Notre souci constant aura été de ne point bouleverser la conception même du système et de faire en sorte que les diverses améliorations s'intègrent de façon naturelle au système SOCRATE actuel.

Pour pouvoir opérer ces modifications, il convenait au préalable d'homogénéiser le langage à la disposition de l'utilisateur en intégrant les mots de commande au sein du langage de citation ([1] § 2.3.). Nous avons ainsi généralisé ce que nous avions commencé à faire au niveau de la boucle POUR.

C'est donc ce langage intégré, que nous appellerons langage de requêtes, que nous étudierons en préambule.

./.

C H A P I T R E 1

LANGAGE DE REQUETES

1.1. BUT ET DEFINITIONS

Jusqu'ici dans le système SOCRATE, des conditions ne pouvaient intervenir qu'au niveau de la définition d'une structure : il s'agissait de déclarations conditionnelles et ceci reste une originalité du système SOCRATE.

Dès l'instant où nous voulons opérer des traitements sur un fichier nous devons rejoindre les langages de programmation classique et permettre la conditionnalité de n'importe quelle action.

Les règles syntaxiques de base du langage de requêtes sont les suivantes :

<PR> → <LR> ?
<LR> → <R> <LR>
<R>
<R> → <RS>
<RC>
<BOUCLE POUR>
<RC> → si <COND> alors <LR> sinon <LR> fin
<BOUCLE POUR> → Pour <CIT D'ENSEMBLE> <LR> fin
<RS> → <MOT DE COMMANDE> <CORPS DE REQUETE>

./.

Une question d'un utilisateur est un véritable programme constitué d'une liste de requêtes et clos par un point d'interrogation. Une requête simple est une demande d'action de l'utilisateur vis à vis du système ; elle est constituée par un mot de commande précédant un corps de requête dont la syntaxe dépend de la nature de l'action commandée.

<RS> → <MOT DE COMMANDE> <CORPS DE REQUETE>

Les mots de commande autorisés dans les requêtes simples sont les suivants :

- { AS : Ajoût de structure

- { G : Génération d'entité
  { A : Ajoût d'entité
  { T : Suppression d'entité
- { M : Mise à jour

- { I : Interrogation
  { N : Dénombrement

Les commandes AS et G seront étudiées dans le cadre de la modification de structure ([2] § 4.1.).

Les commandes M et I ont eu leurs possibilités élargies par rapport à ce qu'elles étaient ; elles seront étudiées dans le chapitre consacré au "calcul de bureau".

Quant aux autres commandes, elles n'ont pas été modifiées ; leur corps de requête respectif, accolé maintenant au mot de commande, correspond à la citation qui, dans le mode classique, était écrite au terminal après intervention du système.

./.

1.2. MODES D'UTILISATION

Lorsqu'un utilisateur charge le système, il peut travailler selon l'un des trois modes suivants :

1) K : Création	Il sert à initialiser un nouveau fichier
2) S : Surveillance	Il permet de désigner une réalisation d'entité sur laquelle le travail s'effectuera.
3) PR : Programme	C'est ce mode qui devient le mode normal d'utilisation, remplaçant le mode classique exposé précédemment ([1] § 2.3.). L'utilisateur s'exprimera dans le langage de requêtes que nous présentons dans ce chapitre.

Les modes de Création et de Surveillance sont deux modes spéciaux que nous avons déjà présentés ([1] § 2.3.2.1., 2.3.2.7.) et auxquels il n'a rien été apporté de neuf. Cependant, remarquons que sous le mode Surveillance on utilisera à nouveau le langage de requêtes.

1.3. EXEMPLES

Nota : *La définition de structure à laquelle nous nous référerons dans tous les exemples suivants est donnée en annexe 1.*

Supposons que la première personne du fichier se nomme 'DUPONT' 'JEAN'. Un utilisateur veut après avoir demandé le nom et le prénom de cette personne changer son prénom de 'JEAN' en 'JOHN'. Nous allons montrer comment il doit le faire suivant le mode "classique", puis comment il peut le faire suivant le mode 'programme'.

./.

Mode "classique"

```
FONCTION (K,I,M,N,S,P,T)
QUELLE FONCTION VOULEZ-VOUS ?
- I
POSEZ VOTRE QUESTION.
- NOM DE UNE PERSONNE ?
NOM DUPONT
POSEZ VOTRE QUESTION.
- PRENOM DE UNE PERSONNE ?
PRENOM JEAN
POSEZ VOTRE QUESTION.
- M
QUE VOULEZ-VOUS METTRE A JOUR ?
- PRENOM DE UNE PERSONNE ?
PRENOM JEAN
PRENOM ?
- JOHN
QUE VOULEZ-VOUS METTRE A JOUR ?
```

Mode "Programme"

```
FONCTION (K,S,PR)
QUELLE FONCTION VOULEZ-VOUS ?
- PR
- I NOM DE UNE PERSONNE
- I PRENOM DE UNE PERSONNE
- M PRENOM DE UNE PERSONNE = 'JOHN' ?
NOM DUPONT
PRENOM JEAN
```

On peut améliorer ce programme, en faisant en sorte que l'objet physique correspondant à la première personne ne soit cherché qu'une seule fois par le système : ceci est réalisé en utilisant la boucle POUR et un démonstratif. En outre, on peut conditionner la mise à jour par le fait que le prénom est 'JEAN' et vérifier que la mise à jour a été effectuée. Ainsi :

```
POUR UNE PERSONNE X1
        I NOM DE X1
        I PRENOM DE X1
        SI PRENOM DE X1 = 'JEAN'
        ALORS M PRENOM DE X1 = 'JOHN'
              I PRENOM DE X1
        FIN
FIN ?
NOM DUPONT
PRENOM JEAN
PRENOM JOHN
```

Enfin, un traitement analogue peut être fait pour toutes les personnes du fichier. On a la possibilité de rendre le contrôle à l'utilisateur au moment d'une mise à jour grâce au symbole EXT en partie droite de mise à jour. Ainsi :

```
POUR TOUTE PERSONNE X1
        I NOM DE X1
        I PRENOM  DE X1
        M PRENOM DE X1 = EXT
FIN ?
        NOM DUPONT
        PRENOM JEAN
        PRENOM ?
      - JOHN
        NOM DURAND
        PRENOM CHARLES
        PRENOM ?
      - CHARLY
.........................
```

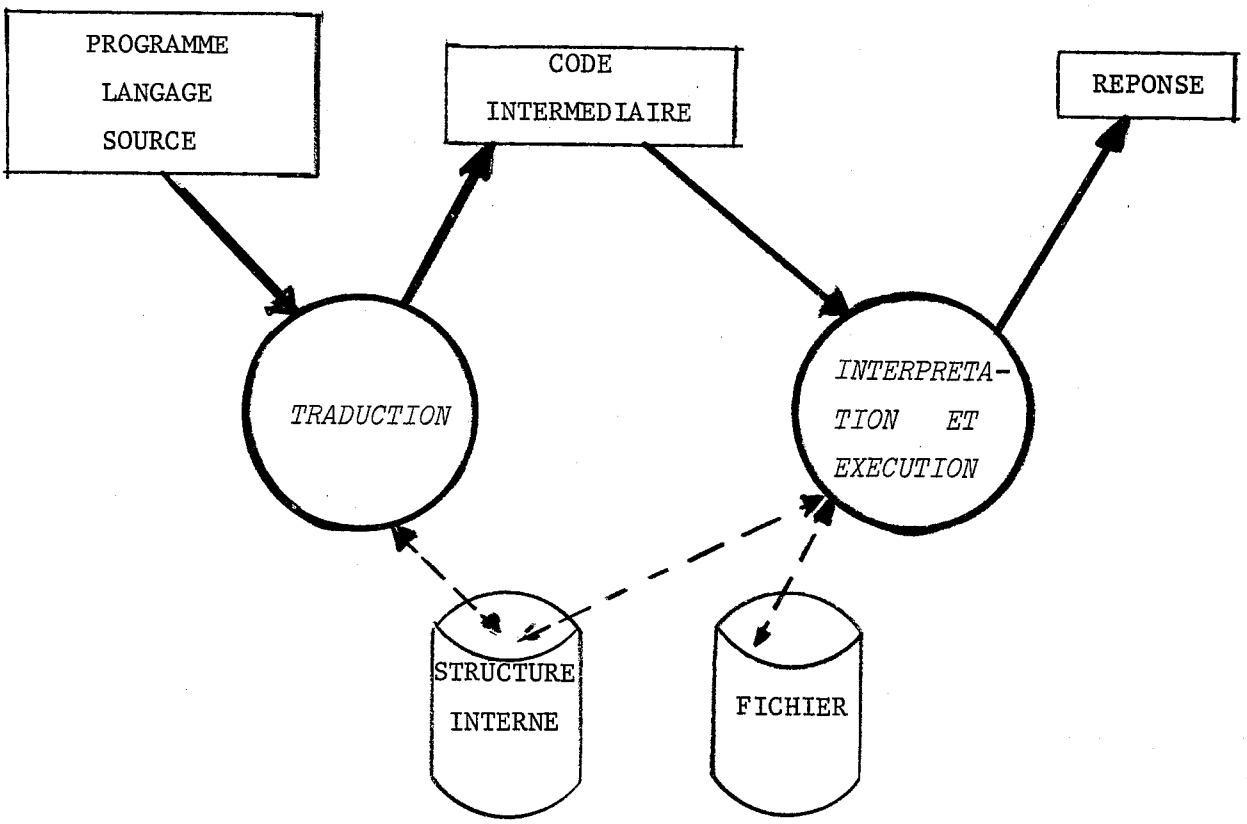
Nous présenterons des exemples plus complexes dans les chapitres suivants après avoir étudié en détail les modes d'interrogation et de mise à jour.

./.

1.4. TECHNIQUES UTILISEES

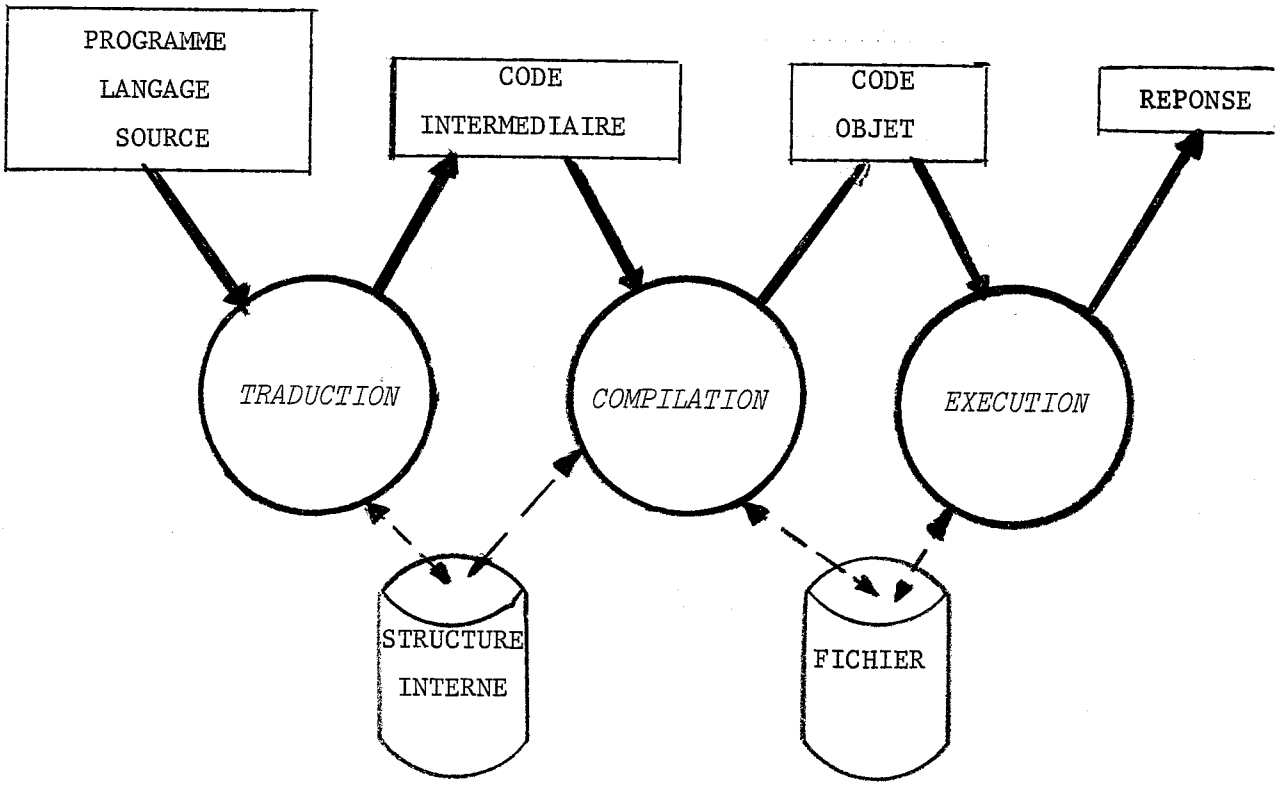
Rappelons que la compilation d'une question s'opère en deux phases, traduction d'un langage source en langage intermédiaire puis interprétation de ce langage intermédiaire ; la 1ère phase a pour données de travail la structure du fichier et la 2ème phase a en outre le fichier lui-même.

La technique utilisée jusqu'alors est représentée sur le schéma suivant :



Actuellement, la technique d'interprétation a été remplacée par une véritable compilation ([4]).

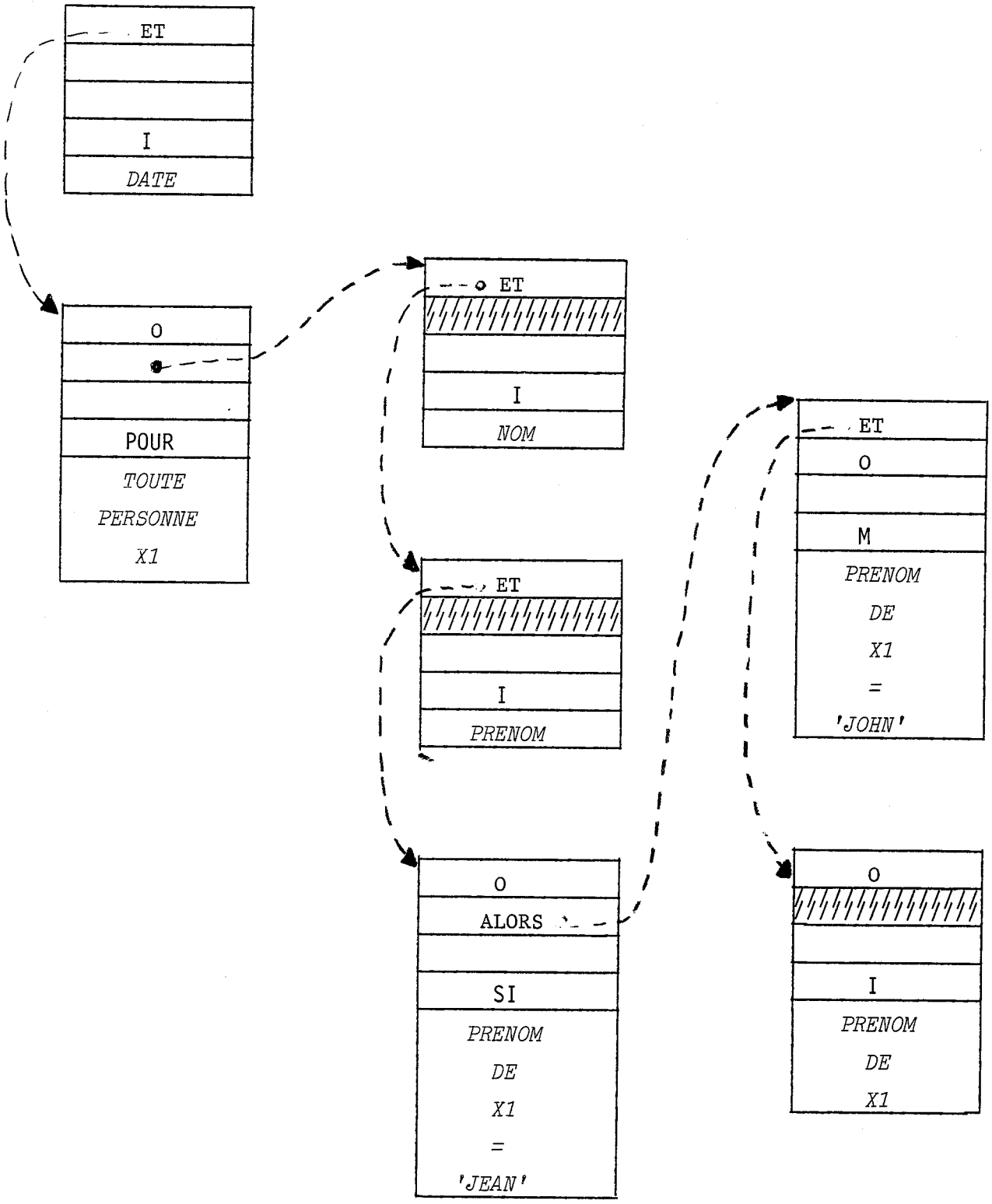
./.



La phase de traduction s'opère en deux passages sur le programme : le premier de vérification de conformité syntaxique, le second de vérification de conformité sémantique avec le contexte de la question et la structure de définition.

Lorsqu'un utilisateur écrit sur une console d'entrée directe un programme de requêtes, le premier passage de traduction est effectué ligne à ligne. Ceci permet, dès qu'une erreur syntaxique est détectée, de la signaler. L'utilisateur doit alors réécrire son programme. Dès l'instant où l'utilisateur va pouvoir écrire des programmes plus complexes donc plus longs il faudra peut être envisager d'effectuer les deux premiers passages de la compilation de façon incrémentale, ligne à ligne. Pour permettre un développement futur dans ce sens nous avons dès à présent utilisé dans le code intermédiaire un chaînage entre les requêtes. Ceci permet de modifier facilement une requête, d'en enlever ou d'en insérer une ou plusieurs au milieu d'un programme. ([1] § 4.3.2.)

./.


ET
I
DATE
O
POUR
TOUTE
PERSONNE
X1
ET
I
NOM
ET
I
PRENOM
O
ALORS
SI
PRENOM
DE
X1
=
'JEAN'
ET
O
M
PRENOM
DE
X1
=
'JOHN'
O
I
PRENOM
DE
X1

Nous avons représenté sur le schéma (p. 10) le code intermédiaire correspondant au programme suivant :

```
I DATE
POUR TOUTE PERSONNE X1
        I NOM
        I PRENOM
        SI PRENOM DE X1 = 'JEAN'
        ALORS M PRENOM DE X1 = 'JOHN'
              I PRENOM DE X1
        FIN
FIN
```

1.5. AVANTAGES ET INCONVENIENTS

En intégrant les mots de commande au langage et en regroupant les requêtes on diminue les interruptions "console", ce qui est particulièrement intéressant en multiprogrammation (cf. [3]).

En regroupant des requêtes d'interrogation on a la possibilité d'une édition souple, entièrement à la convenance de l'utilisateur. Peut-être doit-on envisager, pour améliorer cette édition, la possibilité de définir facteurs de cadrage et sauts de ligne.

Enfin, le principal intérêt de ce mode est de pouvoir en un seul passage sur le fichier effectuer des actions de toute nature et de tester le résultat de ces actions au fur et à mesure de leur déroulement. Le gain de temps est évidemment considérable par rapport à la méthode classique qui parcourait, pour chaque action, l'ensemble du fichier.

./.

C H A P I T R E 2

CALCUL DE BUREAU

2.1. DEFINITIONS

On désire pouvoir faire exécuter par le système des traitements numériques sur des valeurs prises dans un fichier. Pour cela, il faut mettre à la disposition des utilisateurs et du système des variables de travail dans lesquelles seront mémorisés les résultats de ces traitements.

Nous avons défini des variables de trois types :

1) Variables de Désignation Réelle, notées Xi - i pouvant varier de 1 à 10 - Ce sont les démonstratifs qui permettent de désigner une réalisation d'entité et qui ont donc pour valeur une adresse virtuelle ([1] § 2.2.2. § 2.2.

2) Variables numériques, notées Yi - i pouvant varier de 1 à 10
Elles ont pour valeur un nombre.

3) Variables alphanumériques, notées Zi - i pouvant varier de 1 à 10
Elles ont pour valeur une chaîne alphanumérique.
Elles permettent de mémoriser la valeur d'une caractéristique de type "MOT" ou "liste de valeurs" afin de la réutiliser dans une requête suivante d'un programme.

Les requêtes d'interrogation doivent permettre de connaître la valeur des variables numériques et alphanumériques. Le format de ces requêtes d'interrogation est donc de trois types :

./.

1) I → <CITATION ELEMENTAIRE>

2) I → Yi

3) I → Zi

La syntaxe des requêtes de mise à Jour est la suivante :

M <CITATION> = EXT

" = <VALEUR> → {nombre
{mot

" = <VARIABLE> → {Xi
{Yi
{Zi

" = <CITATION>

M <VARIABLE> = <VALEUR>

" = <VARIABLE> <OP> <VARIABLE>

" = <CITATION>

" = <FONCTION> <CITATION>

<OP> = {+
{-
{/
{*

<FONCTION> = n

La seule fonction numérique permise est le dénombrement. Mais il sera rendu possible pour un utilisateur d'ajouter des fonctions qu'il aura lui-même définies. Les points de modification du système ont été rendus extrêmement accessibles dans ce but.

2.2. EXEMPLES ET MACROS DE CALCUL

-- Calcul du Salaire annuel en fonction des salaires mensuels pour chaque membre du personnel puis calcul du salaire annuel moyen sur l'ensemble du personnel. Les exemples présentés ci-dessous correspondent à la structure donnée en l'annexe 1.

./.

```
y1 = 0
Pour toute personne X1
        y2 = 0
        Pour tout mois
                y3 = Salaire
                y2 = y2 + y3
        fin
        i  y2                          ← salaire annuel de X1
        y1 = y1 + y2
fin
        y3 = n toute personne
        y3 = y1/y3
        i  y3                          ← salaire annuel moyen
```

Si un tel calcul doit être employé souvent, l'utilisateur aura intérêt à cataloguer ce programme sous un nom de macro sans paramètres qu'il emploiera simplement sous forme de sous-programme (cf. [I] § 2.4.).

Mais il est possible que l'utilisateur veuille faire parfoir séparément des calculs de somme ou des calculs de moyenne. Il pourra définir ces macros de la façon suivante :

```
!Defmac Somme (!,!,!;!)
    !exp !4! = 0
          Pour tout !2!
               !3! = !1!
               !4! = !4! + !3!
          fin
    !fdef
```

L'appel Somme (Salaire, Mois, y3;y2) génèrera :

```
       y2 = 0
       pour tout Mois
              y3 = Salaire
              y2 = y2 + y3
       fin
```

./.

De même :

```
!Defmac Moyenne (!,!,!,!;!)
!Exp    !3! = 0
        Pour tout !2!
              !4! = !1!
              !3! = !3! + !4!
        fin
              !5! = n tout !2!
              !5! = !3! / !5!
!fedef
```

L'appel Moyenne (Salaire, Mois, y1, y3 ; y2) génèrera :

```
        y1 = 0
        Pour tout Mois
              y3 = Salaire
              y1 = y1 + y3
        fin
              y2 = n tout mois
              y2 = y1 / y2
```

Lorsqu'on veut appeler à l'intérieur d'une macro une autre macro, il faut prendre garde à ce qu'il n'y ait pas de double utilisation d'une même variable de travail.

Deux solutions sont possibles :

1) La première, qui est celle choisie jusqu'ici est de considérer les variables de travail comme des zones statiques ; elles correspondent aux Registres d'un langage d'assemblage. Dès lors c'est à l'utilisateur de gèrer ces registres Comme, d'autre part, le macrogénérateur n'accepte de paramètres que derrière le nom d'appel il faudra passer en paramètre, outre les noms des variables de travail utilisées localement, celui de la variable de retour. Ceci afin de pouvoir appeler les macros de calcul non comme des sous-programmes mais comme des fonctions.

Exemple :

```
!Defmac Somme (!,!,!;!)
   !exp
        0
        pour tout !2!
              !3! = !1!
              !4! = !4! + !3!
        fin
    !fdef


!defmac Moyenne (!,!,!,!;!)
!exp
     0
     !3! = 0
     Pour tout !2!
           !4! = !1!
           !3! = !3! + !4!
     fin
           !5! = n tout !2!
           !5! = !3! / !5!
!fdef
```

Ces 2 macros pourront dès lors être utilisées comme des fonctions et l'une pourra appeler l'autre à condition de prendre garde aux correspondances entre les variables de travail.

Exemple 1 : y5 = Somme (Salaire, Mois, y3, y5)

génère :

```
y5 = 0
     Pour tout mois
     y3 = Salaire
     y5 = y5 + y3
     fin
```

./.

Exemple 2 : y1 = Moyenne (Somme(Salaire, Mois, y3, y5), Personne, y4, y5, y

génèrera :

```
y1 = 0
     y4 = 0
     Pour tout Personne
          y5 = Somme (Salaire, Mois, y3, y5)
          y4 = y4 + y5
     fin
          y1 = n tout Personne
          y1 = y4 / y1
```

```
y5 = 0
     Pour tout moi
       y3 = Salair
       y5 = y5 + y
     fin
```

(2) La solution la plus souple serait évidemment de laisser le soin au Système de gèrer ces registres de travail. On aurait alors les macros suivantes :

```
!defmac Somme (!,!)
    !exp        yj = 0                    yj valeur de retour
                Pour tout !2!
                     yi = !1!
                     yj = yi + yj
                fin
    !fdef


!defmac Moyenne (!,!)
    !exp                ym valeur de retour
                yk = 0
                Pour tout !2!
                     y1 = !1!
                     yk = yk + yl
                fin
                     ym = n tout !2!
                     ym = yk / ym
    !fdef
```

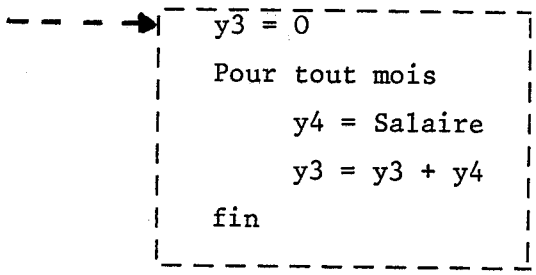
./.

La requête suivante :

```
y1 = Moyenne (Somme (Salaire, Mois), Personne)
```

génère :

```
y2 = 0
Pour toute Personne
        y3 = Somme (Salaire, Mois)
        y2 = y2 + y3
fin
        y1 = n toute Personne
        y1 = y2 / y1
```



Dans l'exemple ci-dessus le macrogénérateur partage la gestion des registres de travail avec le compilateur.

Une autre solution consiste à définir des variables de travail local à l'expansion d'une macro qui devient ainsi un bloc au point de vue de la réservation de mémoire.

L'inconvénient de telles solutions est de détruire le comportement simple et aveugle du macrogénérateur et d'aboutir en fait à la définition de procédures avec passage de paramètres, réservation de mémoire et passage de valeur retournée, analogues à celles des langages de programmation dits évolués.

2.3. CONDITIONS AU NIVEAU DE TOUTE REQUETE

Syntaxe :

- Si <COND> alors <LR> [sinon <LR>] fin

La syntaxe de <COND> est celle exposée dans le langage de citation étendu ([1] § 2.2.2. § 2.2.5.) avec, en outre, la possibilité de tester les variables de travail.

```
Pour toute personne X1
|       i nom
|   si état-civil de X1 ¬= 'marié'
|   alors | si existe une personne X2
|         | telque sexe de X2 ¬= sexe de X1
|         |             et état-civil de X2 ¬= 'marié' ;
|         | alors m y1 = age de X1
|         |       m y2 = age de X2
|         |       si y1 >= y2
|         |       alors y3 = y1-y2
|         |       sinon y3 = y2-y1
|         |       Fin
|         |       si y3 <= 10
|         |       alors m état-civil de X1 = 'marié'
|         |       |     m état-civil de X2 = 'marié'
|         |       |     m conjoint de X1 = X2
|         |       |     m conjoint de X2 = X1
|         |       fin
|         | fin
|   fin
fin
```

Ainsi en un seul passage sur le fichier on marie ensemble les couples de personnes de sexe différent et ayant une différence d'âge inférieure à 10 ans.

INTERET :

L'emploi des conditions à n'importe quel niveau d'une liste de requêtes assouplit la notion de filtre introduit par AYANT et TELQUE en la généralisant. Il permet en outre le regroupement de requêtes déterminé par un même filtre.

On remarque que l'on peut tester dans les filtres d'une condition la valeur d'une variable de travail. Donc en fonction d'un certain calcul effectué sur le fichier, on peut déterminer le choix des actions à exécuter.

./.

C H A P I T R E 3

ACTIONS SPONTANEES

3.1. DEFINITIONS ET BUT

Une action est dite spontanée lorsqu'elle ne répond pas à une requête directe d'un utilisateur mais qu'elle semble être exécutée spontanément par le système. Une telle action résulte en fait d'une requête à effet retardé, stockée dans le système par un utilisateur : cet utilisateur a conditionné son exécution à l'apparition d'un évènement.

Les évènements relatifs à un fichier sont liés à la vie même de ce fichier. Ces évènements dépendent des actions effectuées soit sur les réalisations de ce fichier, soit sur la structure même du fichier.

Nous donnons ci-dessous une classification non exhaustive de ces évènements :

Evènements :

1) Au niveau de la structure du fichier :

Une caractéristique est supprimée formellement.

2) Au niveau des réalisations du fichier :

Une certaine caractéristique prend une certaine valeur,
Une certaine caractéristique passe de telle à telle valeur.
Une entité donnée est créée, ajoutée ou supprimée,
Une fonction d'une certaine classe d'entités prend une certaine valeur,
Une fonction d'une certaine classe d'entités passe de telle à telle valeur.

./.

3.2. EXEMPLES

Reprenons l'exemple du chapitre précédent (2 § 2.3.) dans lequel nous modifions l'état-civil de certaines personnes. Il peut sembler souhaitable à l'utilisateur que le fait que l'état-civil d'une personne passe de 'célibatair à 'marié' soit un évènement qui conditionne la mise à jour du nom-de-jeune-fille de cette personne si elle est de sexe féminin.

Un tel évènement se décompose en trois points :

- il est lié à la caractéristique : ETAT-CIVIL DE PERSONNE
- il dépend de l'action directe : MISE A JOUR
- l'ancienne valeur de la caractéristique était : 'CELIBATAIRE'
  la nouvelle est : 'MARIE'

Remarquons que les actions entraînées "spontanément" par cet évènement peuvent être elles-mêmes conditionnées (ici, par le fait que le sexe de la personne est féminin).

3.3. LA NOTION DE TEMPS DANS LES EVENEMENTS

Nous avons dit précédemment qu'un évènement pouvait être aussi bien une prise de valeur qu'une transition de valeurs. Or dans le système SOCRATE il n'y a pas sédimentation des valeurs prises par une caractéristique en fonction du temps Si un utilisateur veut qu'un historique de ces valeurs soit conservé, il faut qu'il l'organise lui-même au niveau de la structure de définition. Par exemple :

```
entité ETAT-CIVIL
        début VALEUR (CELIBATAIRE, MARIE, VEUF, DIVORCE)
              DATE mot
        fin
```

Ainsi, les valeurs de l'état-civil d'une personne seront conservées avec leur date de mise à jour.

Une seconde remarque s'impose à propos de l'usage du temps dans les fichiers. Il est absurde de définir des caractéristiques simples dynamiques par rapport au temps, telles que l'âge d'une personne.

./.

La seule caractéristique temporellement dynamique doit être le temps lui-même mis sous forme par exemple, d'une caractéristique DATE qu'on modifiera chaque jour.

L'âge d'une personne est donné alors par une macro-fonction qui calcule la différence entre sa date de naissance (statique) et la date actuelle (dynamique

3.4. MODE D'UTILISATION

Les impératifs que nous nous sommes fixés sont les suivants :

1 L'utilisateur doit pouvoir attacher à une caractéristique formelle un ou plusieurs évènements.

2 Il doit pouvoir indiquer l'action qui, exercée sur cette caractéristique, est à l'origine de l'évènement.

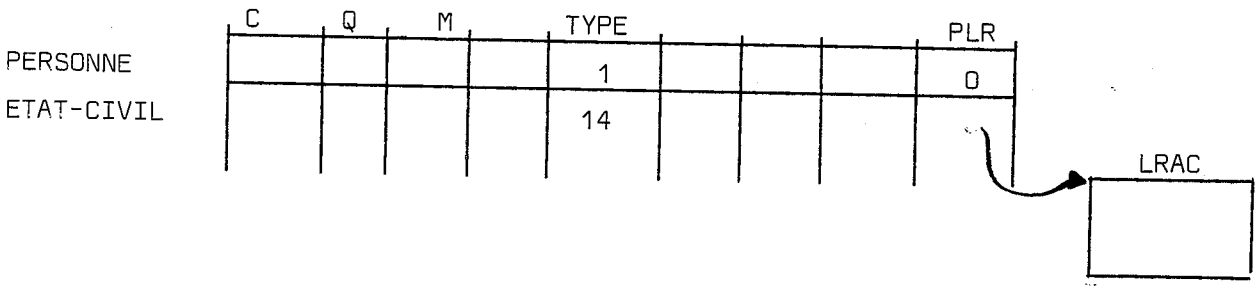
3 Il doit pouvoir décrire cet évènement et les actions qu'il entraîne. Ces actions doivent pouvoir s'exécuter aussi bien avant qu'après celle qui a provoqué l'évènement.

4 Enfin, l'utilisateur doit pouvoir mettre en oeuvre et modifier cet ensemble d'évènements et d'actions résultantes de façon très simple.

La description de l'évènement et des actions conséquentes est faite à l'aide du langage de requêtes décrit ci-dessus.

Leur insertion dans le système se fait au niveau de la structure formelle à l'aide d'une mise à jour simple de structure (COMMANDE : MS). Au niveau de chaque caractéristique formelle, un pointeur PLR désignera, s'il y en a une, la liste de Requêtes Associée à cette Caractéristique (LRAC).

Le schéma de la structure ([1] § 4.3.1.) est le suivant :



Pour ajouter ou mettre à jour une telle LRAC l'utilisateur emploiera la syntaxe suivante :

```
MS   POUR  <CITATION DE CARACTERISTIQUE FORMELLE>
           |AVANT <ACTION> <LR>|*
           |APRES <ACTION> <LR>|*
     FIN
```

Ce qui donne dans l'exemple précédent :

```
MS   POUR ETAT-CIVIL DE PERSONNE
       Avant Mise à Jour   Z1 = ETAT-CIVIL
                           Z2 = SEXE

       Après Mise à Jour   Z3 = ETAT-CIVIL
                           si Z1 = 'CELIBATAIRE'
                           et Z3 = 'MARIE'
                           et Z2 = 'FEMININ'
                           alors M NOM-DE-JEUNE-FILLE = EXT
                           fin
     fin
```

3.5. MISE EN OEUVRE TECHNIQUE

Nous avons souligné que certaines actions spontanées devaient être déclenchées avant même l'action provocatrice. Il faut donc tester l'existence de telles actions spontanées non pas à l'exécution d'une requête mais lors de sa compilation. En analysant un programme de requêtes, le système va rechercher pour toute caractéristique en tête de citation s'il existe une liste de requêtes qui lui est associée pour le code d'action de la citation où elle figure. Dans ce cas le système insèrera, avant et après la requête analysée, celles correspondant aux actions spontanées à effectuer.

Une manière simple de réaliser ceci est de le faire au premier stade de la compilation, au niveau du code intermédiaire en utilisant le chaînage entre les requêtes.

./.

Il convient cependant de faire en sorte que la désignation d'entité-mère de la caractéristique testée soit identique dans la requête analysée et dans celles ajoutées spontanément.

Or la citation dans laquelle intervient la caractéristique testée peut être de trois formes :

1 <CARACTERISTIQUE>
2 <CARACTERISTIQUE> de Xi
3 <CARACTERISTIQUE> de <CITATION COMPLEXE D'ENTITE>

Dans le premier cas, il n'y a rien à modifier à la forme des requêtes. Le deuxième cas est celui que nous avons rencontré dans notre exemple :

```
SI Y3 <= 10
ALORS M ETAT-CIVIL DE X1 = 'MARIE'
       ---------------------------
```

Il convient alors de complèter, dans les requêtes spontanées, toutes les caractéristiques filles de PERSONNE par "de X1". On obtient donc :

```
si Y3  = 10
alors Z1 = ETAT-CIVIL DE X1
      Z2 = SEXE DE X1

      M ETAT-CIVIL DE X1 = 'MARIE'
      Z3 = ETAT-CIVIL DE X1
      si Z1 = 'CELIBATAIRE'
      et Z3 = 'MARIE'
      et Z2 = 'FEMININ'
      alors m NOM-DE-JEUNE-FILLE DE X1 = EXT

      fin
       ------------------
       ------------------
```

Le 3ème cas se présenterait par exemple sous la forme suivante :

M ETAT-CIVIL DE UNE PERSONNE AYANT NOM = 'DUPONT' ; = 'MARIE'

./.

La solution consiste, au niveau du code, à transformer cette requête en :

```
POUR UNE PERSONNE AYANT NOM = 'DUPONT' ;
        M ETAT-CIVIL = 'MARIE'
FIN
```

Cette méthode de travail au niveau du code intermédiaire présente de gros inconvénients dès l'instant où les requêtes d'actions spontanées s'enchaînent entre elles, en cascade. Il se pose alors le problème des variables de travail locales à des LRAC. Il convient de définir un bloc limitant la portée de ces variables, ce bloc englobe les requêtes spontanées à ajouter avant et après une requête donnée. Cette allocation de variables se fait de façon beaucoup plus simple au niveau de la compilation proprement dite du code objet. Il semble qu'une solution doive être trouvée dans cette direction et dans ce cas les programmes d'actions spontanées seraient conservés sous forme de code objet ([4]).

3.6. AVANTAGES ET INCONVENIENTS

Le principal intérêt des listes de requêtes associées aux caractéristiques est de permettre à l'utilisateur de définir très simplement les traitements spontanés qu'il veut voir effectuer par le système en fonction d'évènements qu'il peut provoquer.

Ces traitements peuvent aller du calcul numérique, à la mise à jour de caractéristiques, voire même la modification de structure. L'immense avantage est de livrer à l'utilisateur un système initialement simple dont il peut à loisir augmenter le degré de complexité. Il existe aussi un danger : c'est de faire effectuer spontanément par le système des modifications profondes d'un fichier sans qu'il en soit fait mention extérieurement.

L'utilisateur devra donc se montrer extrêmement prudent et, au moins dans une phase initiale multiplier les traces au niveau de ces actions spontanées.

C H A P I T R E 4

AJOUT DE STRUCTURES

4.1. DEFINITION : MODES D'UTILISATION

Nous présentons ici un nouvel élément de solution au problème de la mise à jour de structure ([1] § 6.3.5.3.).

La commande AS (Ajout de Structure) permet l'ajout d'une ou plusieurs caractéristiques en filles directes benjamines d'une structure de fichier.

Si une caractéristique ainsi ajoutée est simple, on pourra la mettre à jour à l'aide d'une requête de Mise à Jour.

Si c'est une entité, on en créera des réalisations à l'aide de la commande G (Générer) ; pour chacune de ces réalisations ses caractéristiques prendront leur valeur grâce aux requêtes de mise à jour.

Les caractéristiques ainsi ajoutées auront leur place virtuelle réservée en mode "dispersé" à la fin du fichier.

Syntaxe : <AJOUT STRUCTURE> → AS <LK> FIN

<GENERATION> → G UN <NOM D'ENTITE> Xi

4.2. EXEMPLES

1) Au fichier dont la structure est donnée en annexe 2 , nous allons rajouter une date au niveau le plus externe et la mettre à jour.

QUELLE FONCTION VOULEZ-VOUS ?

- *PR*
- *AS DATE DE 1800 A 2000 FIN*
- *M DATE = 1970*

./.

2) Au même fichier, nous allons rajouter une entité RESUME que nous ferons correspondre à chaque malade du fichier et dans laquelle nous mettrons des renseignements pris dans chacune des réalisations des MALADES du fichier.

QUELLE FONCTION VOULEZ-VOUS ?

```
- PR
- AS ENTITE 100 000 RESUME
        DEBUT
            NOM MOT
            PRENOM MOT
            ANNEE DE 1960 A 2000
            RESULTAT-GLOBAL MOT
        FIN
FIN
   POUR TOUT MALADE X1
            G UN RESUME X2
            M NOM DE X2 = NOM DE X1
            M PRENOM DE X2 = PRENOM DE X1
            SI EXISTE UN RESULTAT X3 AYANT EXISTE GLOBAL;DE X1
            ALORS M RESULTAT-GLOBAL DE X2 = GLOBAL DE X3
   FIN
```

Ainsi, on peut recopier une partie ou la totalité d'un fichier dans un autre. Pour le fichier des malades d'un hôpital on peut générer résumé pour ceux qui n'ont pas séjourné dans l'hôpital depuis plus de deux ans, par exemple, puis faire une sortie OS, qu'on stockera sur bande magnétique, pour ces malades et enfin supprimer ces malades du fichier. De la sorte on conservera dans le fichier directement exploité une trace des dossiers des anciens malades avec une possibilité de les retrouver sur la bande magnétique de stockage.

3) Nous avons déjà souligné l'intérêt que peuvent avoir l'ensemble des utilisateurs d'un fichier à connaître la façon dont est employé ce fichier : savoir si une caractéristique est souvent interrogée, souvent mise à jour ou si elle sert souvent pour filtrer une entité.

Il suffit pour cela que l'utilisateur définisse un compteur pour chaque caractéristique formelle qui l'intéresse et pour chaque action qu'il veut étudier sur cette caractéristique.

La définition de ce fichier espion se fera à n'importe quel moment grâce à l'Ajout de Structure et la tenue de ces compteurs se fera à l'aide des requêtes de mises à jour spontanées. Lorsque l'utilisateur aura tiré les leçons de cet espionnage, il pourra supprimer fichier espion et traitements d'espionnage

4.3. AVANTAGES ET INCONVENIENTS

L'Ajout de Structures permet la création des fichiers de travail temporaires. C'est le cas de l'espionnage présenté dans l'exemple précédent. On peut aussi dans un fichier temporaire réaliser un réarrangement de certains objets du fichier général suivant certains critères. Ce nouveau fichier pourrait être exploité pour un besoin particulier de manière plus rapide.

Il permet aussi de créer ou de modifier le fichier permanent. Nous avons vu dans l'exemple 2 une façon agréable de résoudre le problème du vieillissement d'un fichier.

On peut vouloir modifier plus profondément la structure d'un fichier par exemple en introduisant une nouvelle caractéristique à l'intérieur d'une entité déjà existante. Dans ce cas il conviendra temporairement d'organiser "verticalement" les réalisations de cette caractéristique (cf. [I] § 6.3.3 et § 6.3.5.2.).

On peut citer enfin comme une application intéressante, la possibilité d'inverser la structure du fichier par rapport à une caractéristique si au bout d'un certain temps d'utilisation on s'aperçoit, grâce au fichier espion, que cette caractéristique sert souvent à filtrer une entité du fichier.

CONCLUSION

L'évolution de l'utilisation du système, que nous venons de présenter semble désormais irréversible.

Les possibilités considérables qu'apportent le langage de requêtes n'ont pas encore été toutes mises en évidence.

Deux écueils se présentent auxquels il conviendra de prendre garde dans l'évolution future du système.

Le premier est la complexité croissante de la programmation au niveau de l'utilisateur.

Le second est le risque aussi croissant pour un utilisateur jouant à "l'apprenti sorcier" de détériorer tout un fichier à l'aide d'un simple programme de requêtes.

Il convient donc de façon urgente de résoudre le problème de la sécurité des fichiers.

Une solution raisonnable serait de ne confier la mise au point des programmes de requêtes (actions spontanées, macros de calcul, programmes d'utilisation directe), qu'à des personnes extrêmement qualifiées. Dans ce cas la majorité des utilisateurs n'aurait à leur disposition que la méthode classique de travail moins souple mais facilement contrôlable.

Une fois encore, seuls les utilisateurs pourront nous donner une réponse à ces questions.

A N N E X E 1

Définition d'une structure de fichier simplifiée d'une entreprise.

```
DEBUT
    ENTITE PERSONNE
        DEBUT
            NOM MOT
            PRENOM MOT
            SEXE (MASCULIN FEMININ)
            ETAT-CIVIL (CELIBATAIRE MARIE VEUF DIVORCE)
            ENTITE MOIS
                DEBUT
                    SALAIRE DE 0 A 10 000
                FIN
            SI ETAT-CIVIL = 'MARIE'
            ALORS CONJOINT REFERENCE PERSONNE
                    SI SEXE = 'FEMININ'
                    ALORS NOM-DE-JEUNE-FILLE MOT
                    FIN
            FIN
        FIN
    DATE DE 1900 A 2000
FIN
```

A N N E X E 2

Définition d'une structure de Fichier Hospitalier simplifié :

```
DEBUT
    ENTITE MALADE
        DEBUT
            NOM MOT
            PRENOM MOT
            SEXE (MASCULIN FEMININ)
            ----------------------
            ENTITE SEJOUR
                DEBUT
                    DATE-ENTREE DEBUT
                                    JOUR DE 1 A 31
                                    MOIS DE 1 A 12
                                    ANNEE DE 1960 A 2000
                                FIN
                    DATE-SORTIE IDEM DATE-ENTREE

                    SERVICE MOT
                FIN
            ENTITE EXAMEN
                DEBUT
                    - - -
                    - - -
                    - - -
                FIN
            ENTITE RESULTAT
                DEBUT
                    GLOBAL MOT
                    COMMENTAIRE TEXTE
                    MEDECIN MOT
                FIN
        FIN
FIN
```

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

[1] JR. ABRIAL, J. BAS, G. BEAUME, G. HENNERON, R. MORIN, G. VIGLIANO

Projet SOCRATE
(I) Spécifications générales

communication I.M.A.G. Août 1970

[2] G. VIGLIANO

Projet SOCRATE
(2.1.) Langage de requêtes

Thèse présentée à l'Université de Grenoble, déc. 197

[3] G. BEAUME

Projet SOCRATE
(2.2.) Gestion des mémoires

Thèse présentée à l'Université de Grenoble, déc. 1970

[4] R. MORIN

Projet SOCRATE
(2.3.) Compilation et Interprétation

Thèse présentée à l'Université de Grenoble, déc. 1970

VU,

Grenoble, le

Le Président de la Thèse

Vu,
Grenoble, le

Le Doyen de la Faculté des Sciences

Vu, et permis d'imprimer

Le Recteur de l'Académie
de Grenoble