AN IMAGE MANAGEMENT KERNEL FOR THE DESIGN OF RELATIONAL AND PICTORIAL DATA BASES

UN NOYAU DE GESTION ICONOGRAPHIQUE POUR LA CONCEPTION DE BASES DE DONNEES IMAGE

Philippe CHASSIGNET

LACTAMME - Centre de Mathématiques Appliquées
Ecole Polytechnique, 91128 Palaiseau Cedex, FRANCE

RESUME

Dans cette note nous présentons un projet original pour la réalisation d'un Noyau de Gestion et de Description d'Images. Ce noyau peut être aisément implémenté sur tout système bâti autour d'une Machine Iconographique et d'un Système de Gestion de Bases de Données Relationnel.

Un des intérêts de ce système est de permettre la gestion conjointe de données iconographiques et sémantiques. Dans ce but, la puissance caractéristique d'un Système de Gestion de Base de Données Relationnel, nous permet d'associer finement des informations relatives à la forme et au fond des images ou tout autres.

Pour des raisons de performance, le SGBD ne manipule pas directement les images. Cette tâche est impartie à une Machine Iconographique spécialisée qui permet de modifier ou de combiner des images et donc d'en générer beaucoup d'autres à partir d'une bibliothèque de base. Le Noyau Iconographique est alors conçu pour faciliter la connexion de ces deux systèmes.

ABSTRACT

In this note we present an original scheme for the realization of an Image Management and Description Kernel. This kernel may be easy implemented on any system build around an Iconographic Machine and a Relationnal Data Base Management System.

Thus, one of the main interests of such a system is to permit the joint management of both iconographic and semantic data. To this end, the wealth of data handling, which characterizes a Relationnal Data Base Management System, allows to finely connect informations about the form and substance of pictures and about anything else.

Owing to performances, the DBMS does not directly handle images. This task is imparted to a specialized Iconographic Machine that allows to modify or combine pictures and thus permits to generate many more from a basic library.

Thus, the Iconographic Kernel is intended to connect easier these both systems.

KEYWORDS : Pictorial Data Base, Relationnal Data Base Management System.

1 - INTRODUCTION

L'image permet d'utiliser la richesse d'analyse de l'oeil pour appréhender efficacement des données complexes. Aussi, une part de plus en plus importante des données traitées et produites informatiquement est-elle traduite sous la forme d'images.

Il est alors intéressant de pouvoir traiter ce nouveau type de données de manière similaire à d'autres, qu'elles soient élémentaires comme des grandeurs numériques, des chaînes de caractères ou des codes spécifiques, ou alors plus complexes, comme par exemple des faits et des règles.

A cet effet les Systèmes de Gestion de Base de Données Relationnels, s'avèrent des outils très appropriés du fait de leurs qualités fondamentales de dynamisme et d'indépendance des données, de maintient aisé et sûr de la cohérence, de simplicité de présentation et de souplesse d'accès [Codd 81, Date 82, Zloo 75]. On privilégie alors la manipulation des relations existants entre les informations plutôt que celle des informations élémentaires. La plupart des informations s'expriment en effet naturellement selon la formulation relationnelle c'est à dire sous forme de tables dont chaque ligne représente une certaine relation entre les quelques données qui la composent.

De ce fait, les Systèmes Relationnels ont été largement utilisés dans des applications graphiques [ChFu 79-80, YaKu 82] et, en particulier dans le domaine des bases de données cartographiques [SmAl 79, VSHM 82].

Par réunion d'un certain nombre des moyens logiciels et/ou matériels, nous pouvons, par ailleurs, réaliser un système spécialisé dans la synthèse d'images [NoSp 79, FoVD 82, GuSt 82]. Ce système que nous appellerons Machine Iconographique assure les trois fonctions essentielles de stockage, de traitement et de diffusion de représentations que nous qualifierons de bas niveaux.

La Machine Iconographique se compose donc, en premier lieu, d'une Base d'Images Physiques qui permet de mémoriser un certain nombre de représentations. Celles-ci seront qualifiées, soit de "permanentes" (ou encore de base) si elles concernent des entités prédéfinies, soit de "temporaires" si elle sont allouées à des intermédiaires de calcul ou des résultats en cours de diffusion.

Un certain nombre de Processeurs Iconographiques permettent alors de réaliser sur ces représentations plusieurs types d'opérations. Par la suite, nous nous intéresserons uniquement aux processeurs permettant le calcul d'une représentation temporaire par transformations et combinaisons d'autres représentations.

La Machine Iconographique gère enfin des Moyens de Diffusion (c'est à dire, au moins un moniteur) adaptés au type des représentations manipulées.

Selon cette approche, le système SMC, développé au LACTAMME [Colo 84] comporte une grande variété de processeurs logiciels permettant une gamme étendue d'opérations iconographiques. Il s'agit d'un système souple et puissant mais présentant tout de même une lacune importante. Les accès aux programmes de synthèse se font en effet par éditeur, et ceux-ci se révèlent donc très hermétiques quant à leur contenu. Par exemple, les relations "génétiques" entre images ne sont pas exploitables simplement.

Le but de cette étude est de concevoir un système qui puisse combiner étroitement la synthèse d'images numériques et leur gestion en relation avec d'autres données. Certaines de ces données sont ainsi référencées dans des descriptions d'images et peuvent donc être exploitées lors de leur synthèse. Naturellement, pour les besoins spécifiques d'une application, des données conceptuellement indépendantes des représentations graphiques peuvent également être liées à ces descriptions.

2 - SCHEMA FONCTIONNEL DU SYSTEME

2.1 - Décomposition Fonctionnelle

Pour répondre aux besoins énoncés en introduction le système doit assurer deux fonctions essentielles.

La première consiste à mémoriser et manipuler des représentations de bas niveaux dont une des caractéristiques essentielles est d'être volumineuses. Par conséquent, ces entités nécéssitent des temps de traitements importants mais, pour la même raison, elles figurent en nombre restreint et leur gestion est très simple.

La deuxième fonction consiste à gérer les autres données et les relations établies entre elles. Dans ce cas, il s'agit de saisir, mémoriser, consulter et mettre à jour des entités dont le traitement unitaire est très rapide mais dont le grand nombre suppose des problèmes, entre autres, de sélection et d'accès.

Les données relatives à cette seconde fonction peuvent être réparties arbitrairement en deux ensembles. Le premier regroupe alors toutes les informations qui ne concernent pas explicitement la synthèse d'images et dont la gestion constitue l'un des buts essentiels du système. La nature de ces données, et des relations établies entre elles, dépend alors de l'application.

L'ensemble complémentaire se compose d'entités iconographiques, qui doivent être vues comme des programmes de synthèse d'images, mais que nous voulons gérer comme données descriptives de ces mêmes images. Ceci implique en particulier de les mettre en relation avec des informations classées dans le premier ensemble.

Pour cela, nous avons choisi d'intégrer les deux ensembles dans une même Base de Données.

Chacune des deux fonctions évoquées requiert des solutions très spécifiques et il est donc préférable de concevoir le système autour de la réunion de deux composantes indépendantes et spécialisées. Chacune pourra ainsi faire aisément appel aux solutions logicielles et matérielles, éprouvées et couramment utilisées dans son domaine d'application respectif.

Ainsi, la composante chargée de la première fonction sera une Machine Iconographique et nous avons choisi de confier la seconde fonction à un Système Relationnel.

2.2 - Noyau Iconographique

Le schéma d'organisation des entités iconographiques peut être considéré comme un squelette qui est défini indépendamment de l'application et autour duquel le concepteur de la base peut greffer les données qui composent le premier ensemble. Le schéma global qui en résulte est bien sûr fonction de l'application et "tient" sur le Squelette Iconographique par les relations établies entre entités des deux ensembles.

Du fait de sa structure "plate", le modèle relationnel s'avère alors trop limité pour manipuler les entités iconographiques lorsqu'elles sont considérées comme programmes de synthèse. Cette aspect impose, en particulier, l'existence de dépendances hiérarchiques établies entre des entités définies les unes par rapport aux autres.

Les accès se font donc par l'intermédiaire d'une Interface d'Extension qui, pour garantir une certaine portabilité, ne fait appel qu'à des opérations relationnelles standards. Les manipulations nécéssaires sont alors assurées par un ensemble d'outils qui accèdent aux entités via cette interface et via le Système Relationnel.

Les éléments que nous venons de mentionner constituent la troisième composante du système que nous appelons Noyau Iconographique. Le noyau établit la connexion entre les deux autres composantes et comprend, entre autres, deux outils essentiels. L'un est l'Editeur Iconographique qui permet de définir, sélectionner et modifier des images, des processeurs et des séquences d'animation. Le second est un Interpréteur de Pilotage chargé de contrôler la Machine Iconographique pour calculer et visualiser, les images et autres entités manipulées notamment par l'éditeur.

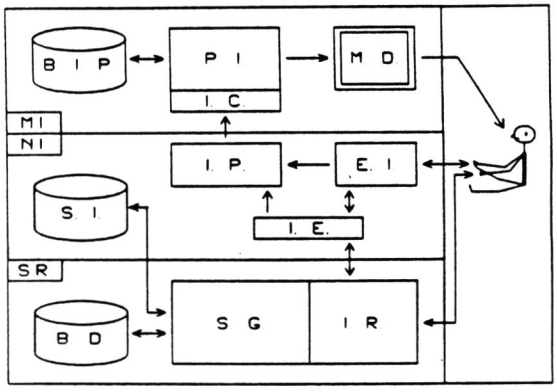

Figure 2.1 Schéma Fonctionnel

Légende :

MI : Machine Iconographique

BIP : Base des Images Physiques
PI : Processeurs Iconographiques
IC : Interface de Commande
MD : Moyens de Diffusion

NI : Noyau Iconographique

SI : Squelette Iconographique
IE : Interface d'Extension
EI : Editeur Iconographique
IP : Interface de Pilotage

SR : Système Relationnel

SG : Système de Gestion
IR : Interface Relationnelle
BD : Base de Données
(dépendantes de l'application)

2.3 - Aspects Pratiques

Libertés d'Implémentation

La décomposition fonctionnelle que nous venons d'effectuer laisse, sur le plan pratique, une grande liberté de mise en oeuvre. Elle peut donc, tout d'abord, se traduire par trois ensembles logiciels spécifiques et implémentés sur une machine universelle suffisamment puissante.

Mais elle peut aussi s'exprimer au niveau matériel et, dans ce cas, le système se compose de machines spécialisées, physiquement distinctes et éventuellement distantes. Selon les besoins particuliers de l'application, les divers éléments du noyau se répartissent alors entre les machines, voire entre les sites. C'est dans un tel contexte que se justifie la normalisation que nous avons prévue au niveau des interfaces entre composantes.

Parmi toutes les possibilités de configurations réparties alors envisageables, on peut remarquer deux exemples typiques.

Une Machine Iconographique "personnelle", faisant office de terminal évolué, peut permettre d'accéder à de larges Bases Relationnelles qui servent, entre autres données, des descriptions d'images. A l'inverse, à partir d'une Base de Données "personnelle", il est possible de soumettre des synthèses d'images de haute qualité à une Machine Iconographique suffisamment rapide et surtout environnée des moyens de production adaptés.

Notons que dans ces exemples les représentations de bas niveaux ne transitent pas entre les sites mais sont toujours calculées sur celui de destination.

Performances

Le problème du temps d'exécution des requêtes, qui est l'un des principaux inconvénients que l'on reconnaît aux systèmes relationnels, est encore amplifié par la complexité des requêtes que le Noyau soumet au SGBD.

Toutefois, ces requêtes sont en général associées à des calculs de représentations. L'intérêt pratique de cette approche repose alors fortement sur l'hypothèse que le temps de réponse global est essentiellement imputable à la Machine Iconographique. Autrement dit, la durée des accès aux descriptions logiques et des traitement afférents est au plus comparable à celle des manipulations effectives de représentations.

Ceci suppose, en particulier, que soit respecté un certain rapport de performance entre les composantes. Remarquons alors que, pour un SGBD donné, ce rapport peut toujours être ajusté dans le sens favorable en augmentant la

complexité, c'est à dire le réalisme, des représentations gérées par la Machine Iconographique.

3 - DESCRIPTION DES ENTITES ICONOGRAPHIQUES

3.1 - Principe et Terminologie

En règle générale, toute entité iconographique est définie de manière récursive par référence à d'autres entités. Les seules exceptions à cette règle seront des entités intrinsèques et, en fonction de leur sémantique, nous distinguerons plusieurs type d'entités non-intrinsèques, à savoir les listes, images, processeurs et séquences.

Type liste

Dans ce premier cas, une entité est simplement considérée comme une liste d'autres entités, et correspond donc à une S-expression du langage LISP [McCa 60].

Du point de vue relationnel, une entité-liste intervient alors comme un ensemble de nuplets traduisant la relation d'appartenance élément-liste. Par exemple, une entité-liste peut être le résultat d'une requête au SGBD et regroupe alors l'ensemble des entités satisfaisant un certain critère de sélection. Réciproquement une entité-liste peut être utilisée comme argument collectif dans la formulation de nouvelles requêtes.

Dans certains contextes, l'ordre des entités dans la liste sera également significatif.

Type image

L'image est, bien sûr, le type fondamental d'entité iconographique. A une telle entité est associé le résultat d'une opération iconographique virtuelle.

Cette opération peut être formulée par l'expression :

$$(\mathrm{Proc},\pi)(\mathrm{Arg}_1,\ldots,\mathrm{Arg}_k)$$

Sa forme est similaire à celle d'une instruction physique exécutable sur la Machine Iconographique.

Cependant, dans le cas présent, le processeur Proc et ses images arguments (Arg_i) sont d'autres entités faisant références à leurs descriptions respectives. Comme dans une instruction physique, une valuation de paramètres (π) est également nécéssaire pour définir précisément le résultat de cette opération. Ces paramètres peuvent être alors valués, soit explicitement, soit comme fonction d'autres paramètres.

Une entité-image est donc complètement décrite par un ensemble d'entités, un ensemble de références les liant entre elles et un ensemble de paramètres valués qui composent son programme de calcul.

En général, les valeurs des paramètres auront une influence relativement limitée sur l'apparence globale et la sémantique de résultat. Dans notre approche, les paramètres interviennent donc comme compléments plutôt que comme véritables informations et, par conséquent, leur gestion sera simplifiée par rapport à celle des autres éléments de l'expression.

A la formulation précédente, et en s'inspirant de la notation LISP, nous préférerons alors une expression :

$$(\mathrm{Proc},\mathrm{Arg}_1,\ldots,\mathrm{Arg}_k)(\pi)$$

qui sépare la liste des entités iconographiques référencées de celle des valeurs affectées aux paramètres.

Type processeur

Si l'on change l'un des éléments (référence ou paramètre) intervenant dans un programme de calcul, l'image résultat se trouve (en principe!) modifiée. Donc, si l'on désigne certains des éléments comme variables, le programme n'est plus descriptif d'une image particulière mais de toute une classe. Un représentant de cette classe n'est alors parfaitement défini que si l'on affecte convenablement les diverses variables.

L'entité générique qui est ainsi décrite est dite de type processeur. Un processeur est en fait une fonction dont le résultat est une entité, ayant comme arguments d'autres entités et des paramètres et dont l'usage est purement applicatif [Hend 80]. En particulier, il ne produit aucun effet de bords et en, termes plus concrets, une image ne sera jamais altérée quelque soit son usage.

Toute variable qui intervient dans la description d'une entité-processeur est alors définie comme étant une valeur particulière modifiable.

Cette valeur aura été déterminée pour donner par défaut un résultat "généralement satisfaisant". En outre, dans la mesure du possible, ce résultat doit alors constituer un bon exemple d'utilisation du processeur.

L'expression (Proc)( ) est donc licite et désigne l'image associée par défaut à un processeur Proc donné.

Pour définir d'autres résultats, cette valeur est alors modifiée en fonction des arguments fournis. Pour décrire une image donnée, on précise donc son processeur, puis les seuls arguments qui suffisent à la distinguer de l'image générique associée au processeur. Notons qu'il est également possible de "passer" un processeur en argument et qu'existe donc la notion de proceseeur par défaut.

On retrouve, dans ce sens, les concepts de classes, sous-classes et métaclasses tels qu'ils apparaissent dans les langages orientés objets du type SMALLTALK [GoRo 83].

Type séquence

Une entité-séquence est un cas particulier de processeur et définit des images fonctions uniquement d'un paramètre. Ainsi, à partir d'un processeur où n'interviennent que des variables paramètres, une image est définie par une valuation particulière et l'on peut donc définir une séquence par un chemin dans l'espace de paramètrisation. Il existe une multitude de chemins et donc de séquences possibles.

Lors d'une diffusion le paramètre du chemin sera alors le temps. Il faut donc noter que les images d'une séquence donnée sont virtuelles au sens où elles ne sont définies que lors de certaines générations de l'animation. Cette éventualité dépend alors de l'échantillonnage effectué, donc à la fois du rythme imposé et de la définition temporelle du support.

Remarques

On peut considérer les images comme des processeurs constants. Le type processeur est alors compris comme un sur-ensemble des types image et séquence, tous étant des cas particuliers de liste. On peut donc poser, par abus :

$$\{ \text{Images} \} \subset \{ \text{Séquences} \} \subset \{ \text{Processeurs} \} \subset \{ \text{Listes} \}$$

3.2 - Squelette Iconographique

Notations

Soient d un entier et Θ un ensemble de constantes.

Soit alors Γ l'ensemble des fonctions définies de Θ^d dans Θ.

Pour tout $f \in \Gamma$ nous définissons alors $\Delta(f)$ comme l'ensemble des entiers $i \in [1..d]$ tels qu'il existe :

$$(c_1,\ldots,c_{i-1},c_{i+1},\ldots,c_d) \in \Theta^{d-1}$$
$$\text{et} \quad c,c' \in \Theta$$
$$\text{tels que} \quad f(c_1,\ldots,c_{i-1},c,c_{i+1},\ldots,c_d) \neq f(c_1,\ldots,c_{i-1},c',c_{i+1},\ldots,c_d)$$

Soit enfin E, un ensemble d'entités iconographiques.

Références

Soit $R \subset (E \times N)^2$, l'ensemble des références établies entre les entités de E. Une référence iconographique est donc un quadruplet (α,i,β,j) où α,β sont deux entités iconographiques et i,j deux entiers positifs.

Nous dirons que l'entité α référence l'entité β et noterons :

$$\text{Réf}(\alpha) = \{ (\alpha,i,\beta,j) \in R \}$$
$$\text{et Arg}(\alpha) = \{ (\alpha,i,\beta,j) \in R \ \text{tq} \ i > 0 \}$$

Nous imposons ensuite que les références de R vérifient la propriété suivante :

Pour tous $r,r' \in R$
si $r = (\alpha,i,\beta,j)$, $r' = (\alpha',i',\beta',j')$
et si $\alpha = \alpha'$ et $i = i'$
alors $r = r'$

Dans ces conditions, nous dirons que l'entier i est le rang de la référence r. Celle-ci sera dite référence constante à l'entité β si $j = 0$. Si $j > 0$, nous dirons qu'il s'agit d'une référence variable, l'entité référencée est alors l'éventuel argument de rang j ou, à défaut, β.

Paramètres

Soit $P \subset E \times N \times \Theta \times \Gamma$, l'ensemble des paramètres associés aux entités de E. Un paramètre est donc un quadruplet (α,i,c,f) dont nous dirons qu'il est associé à l'entité α. Nous noterons alors :

$$\text{Par}(\alpha) = \{ (\alpha,i,c,f) \in P \}$$

Nous imposons que les paramètres de P vérifient la propriété similaire à celle établies pour les références. Soit :

Pour tous $p,p' \in P$
si $p = (\alpha,i,c,f)$, $p' = (\alpha',i',c',f)$
et si $\alpha = \alpha'$ et $i = i'$
alors $p = p'$

Et nous dirons que, de même, i est le rang du paramètre p, que la constante c est sa valeur par défaut, et que f est la fonction formelle de paramétrage.

Pour tout $c \in \Theta$, on notera par $(\alpha,i,c,\dagger)$ le paramètre constant de valeur c. On considère pour cela une fonction $\dagger$ qui est systématiquement non-évaluable, autrement dit, telle que $\Delta(\dagger) \not\subset [1..d]$.

Squelette

Soit maintenant $K = [E,R,P]$ le graphe orienté défini comme suit :
(1) l'ensemble de ses sommets est E,
(2) tout sommet α est étiqueté par l'ensemble $Par(\alpha)$,
(3) les arcs sont les couples $(\alpha,\beta) \in E^2$ tels que $(\alpha,i,\beta,j) \in R$,
(4) l'arc (α,β) est alors étiqueté par i,j.

Remarquons que le rang i suffit à identifier tout arc issu d'un sommet α donné.

Nous dirons alors que K constitue un Squelette Iconographique si :
(1) il est acyclique (DAG), cette propriété interdit donc, en particulier, les références croisées et autoréférences,
(2) l'ensemble de ses feuilles est inclus dans un ensemble donné d'entités dites intrinsèques.

Tout sous-graphe maximal de K dont la racine est une entité donnée sera appelé description ou programme de cette entité.

4 - INTERPRETATION DES ENTITES ICONOGRAPHIQUES

4.1 - Première Règle de Classification

Une entité α telle que $Réf(\alpha) = \phi$ sera dite du type intrinsèque.

Une entité α telle que $Réf(\alpha) = Arg(\alpha) \neq \phi$ sera dite du type liste (au sens strict du terme)

Dans les autres cas, il existe donc
$\beta \in E$ et $j \in N$
tels que $(\alpha,0,\beta,j) \in Réf(\alpha)$.
Nous dirons alors que :
(1) l'entité α est du type processeur (au sens large incluant les types image et séquence),
(2) β est l'opérateur de α.

4.2 - Fonction de Substitution

Introduisons la fonction de substitution S qui est définie par les règles suivantes :

Soit $K = [E,R,P]$ un Squelette Iconographique.

Soit α une entité de E et soit $\prod$ une partie de R.

Soit enfin $\sum$ une partie de R dite "compatible" avec S, c'est à dire vérifiant :
pour tout $(\gamma,j,\delta,k) \in \sum$ alors $j > 0$.

Soit alors $\alpha' = S(\alpha,\sum,\prod)$ et définissons les six ensembles suivants :

$\sum_1$ = l'ensemble des références $(\alpha',i,\delta,0)$

telles qu'il existe un entier $j > 0$ tel que :
(1) il existe $(\alpha,i,\beta,j) \in Réf(\alpha)$,
(2) il existe $(\gamma,j,\delta,k) \in \sum$.

$\sum_2$ = l'ensemble des références $(\alpha',i,\delta,0)$

telles qu'il existe un entier j tel que :
(1) il existe $(\alpha,i,\beta,j) \in Réf(\alpha)$,
(2) pour tout $(\gamma,j',\epsilon,k) \in \sum$ alors $j \neq j'$.

Si ces deux conditions sont vérifiées,

alors $\delta = S(\beta,\sum,\prod)$ si $i > 0$
et $\delta = \beta$ si $i = 0$.

$\sum_3$ = l'ensemble des références $(\alpha',i,\delta,0)$

telles que :
(1) il existe $(\gamma,i,\delta,k) \in \sum$,
(2) pour tout $(\alpha,i',\beta,j) \in Réf(\alpha)$
alors $i \neq i'$.

$\prod_1$ = l'ensemble des paramètres $(\alpha',i,k,\dagger)$

tels que :
(1) il existe $(\alpha,i,c,f) \in Par(\alpha)$,
(2) pour tout $j \in \Delta(f)$
il existe $(\beta,j,c_j,g) \in \prod$.

Si ces deux conditions sont vérifiées, alors $k = f(c_1,\ldots,c_d)$ après avoir complété (d'une manière quelconque) la suite des c_j.

$\prod_2$ = l'ensemble des paramètres $(\alpha',i,k,\dagger)$

tels que :
(1) il existe $(\alpha,i,k,f) \in Par(\alpha)$,
(2) il existe $j \in \Delta(f)$
tel que pour tout $(\beta,j',c,g) \in \prod$
alors $j \neq j'$.

$\prod_3$ = l'ensemble des paramètres $(\alpha',i,k,\dagger)$

tels que :
(1) il existe $(\beta,i,k,f) \in \prod$,
(2) pour tout $(\alpha,i',c,f) \in Par(\alpha)$
alors $i \neq i'$.

La fonction S est alors définie par la donnée des références et paramètres qui caractérisent l'entité α' et pour cela nous posons :

$$Réf(\alpha') = \sum_1 \cup \sum_2 \cup \sum_3$$
et
$$Par(\alpha') = \prod_1 \cup \prod_2 \cup \prod_3$$

4.3 - Interprétation

La fonction S définit une entité α' équivallente à l'interprétation de l'entité α dans le contexte donné par Σ et Π.

Elle permet de définir une fonction Rep de E dans I qui associe une représentation Rép(α) à toute entité non-intrinsèque α. Dans le cas général, cette fonction se définit alors par :

Soit α un processeur (au sens large) et soit alors β son opérateur (non-intrinsèque) alors :

$$Rep(\alpha) = Rep(S(\beta, Arg(\alpha), Par(\alpha)).$$

Toute référence $(\alpha', i, \delta, 0)$ de Σ_1 ou Σ_2 est ainsi directement dérivée d'un élément (α, i, β, j) de Réf(α).

Dans le cas de Σ_1, l'entité δ est substituée à l'entité par défaut β. Cette règle concerne éventuellement l'opérateur qui peut être aussi passé en argument. Cet ensemble résulte donc classiquement d'une substitution des arguments formels par les arguments d'appel.

Dans le cas de Σ_2, il n'y a pas de substitution, soit parce que la référence est constante, soit par défaut d'argument.

Selon la clause générale $i > 0$, la réécriture est alors répercutée sur l'entité β, autrement dit, les arguments sont propagés. Notons alors que cette définition récurrente de δ termine nécéssairement puisque le squelette est un DAG. Enfin, la clause particulière $i = 0$ permet, le cas échéant, de récupérer l'opérateur de l'entité α.

Tout paramètre constant (α', i, k, t) de Π_1 ou Π_2 est d'une manière comparable dérivé d'un élément (α, i, c, f) de Par(α).

Dans le cas de Π_1, la fonction f est évaluable sur Π et k est alors la valeur résultante.

Dans le cas de Π_2, la fonction f n'est pas évaluable (éventuellement constante) et k est alors simplement égal à la valeur par défaut c.

Enfin, Σ_3 et Π_3 regroupent, par simple recopie, les références de Σ et les paramètres de Π qui ne sont pas redéfinis dans la description de α et sont donc visibles de α'. Les critères de visibilités sont alors les mêmes que dans la plupart des langages procéduraux du type PASCAL.

4.4 - Seconde Règle de Classification

Soit α une entité-processeur (au sens large), nous dirons que :

(1) α est du type image si
pour tous $\Pi \subset P$
et $\Sigma \subset R$ compatible avec S
alors $S(\alpha, \Sigma, \Pi) = \alpha$,

(2) α est du type séquence si
pour tout $p = (\beta, 1, c, f) \in P$
alors $S(\alpha, \phi, \{p\})$ est une image.

5 - CONCLUSION

Ce papier présentait les aspects formels d'un système de Base de Données Images défini autour d'une Machine Iconographique et d'un SGBD Relationnel.

La répartition des tâches entre ces deux composantes lui confère une structure très simple et largement ouverte vers des évolutions spécifiques dans les deux domaines concernés.

Automatisation

Un premier moyen d'accroître les possibilités d'un tel système est d'attribuer à chaque image réalisée une note ou une appréciation sur divers aspects esthétiques ou techniques. Ce qui revient alors indirectement à noter l'usage des commandes et les choix des paramètres. Le système est alors à même de proposer une "base d'expériences" (aussi bien positives que négatives) pouvant donc mémoriser un certain savoir-faire en synthèse d'image numérique, aisément exploitable, s'enrichissant automatiquement au fil de l'utilisation et très utile à des fins pédagogiques.

Si de plus on a la possibilité d'effectuer certaines inférences basées par exemple sur la nature des commandes, il est alors possible d'effectuer quelques manipulations sémantiques autour du squelette que constitue le programme. Ainsi en création publicitaire si l'on dispose d'un ensemble d'images auxquelles sont associés des symboles, on peut retrouver des images répondant à certaines associations d'idées (relatives au produit à promouvoir) et ensuite les utiliser pour réaliser l'affiche ou le spot.

On peut même envisager qu'un système expert utilise cette base de connaissances, concernant aussi bien la synthèse des images que leur contenu, et permette alors une automatisation poussée de la tâche de description des images et scénarios.

Et, l'on pourrait, dans ce but, étendre la notion de valeur par défaut à celle de valeur la plus appropriée d'après le contexte et les expérience précédentes.

Parallélisme

Rappelons que les définitions d'images sont structurés en graphes de calcul dont les noeuds sont les opérations et dont les arcs figurent les transmissions d'images arguments. Il s'agit là d'une structure pouvant contrôler une synthèse avec synchronisation sur les arguments, en l'occurence les images intermédiaires; d'ou une notion d'Image Flow [SyCH77, McGr 80].

Le noyau pilote alors un ensemble de Machines Iconographiques (polyvalentes ou spécialisées) se partageant la même Base d'Images Physiques.

6 - REMERCIEMENTS

Je tiens à remercier le Pr. Erol GELENBE, qui me suggéra cette étude.

Je remercie également Jean Francois COLONNA pour ses commentaires fructueux.

7 - REFERENCES

[ChFu 79] N.S. CHANG, K.S. FU :
"A Relational Database System for Images".
Lecture Notes in Computer Science, vol. 80, Pictorial Information Systems, 1979.

[ChFu 80] N.S. CHANG, K.S. FU :
"Query By Pictorial Exemple".
IEEE Trans. on Soft. Eng., vol. SE-6, N$ 11, november 1980.

[Codd 81] E.F. CODD :
"The capabilities of Relational Database Management Systems".
RJ 3132 Research Report 5/11/81. IBM Res. Lab. San Jose (USA).

[Colo 84] J.F. COLONNA :
"De la Visualisation de Résultats de Calculs à la création artistique".
Premier Colloque Image. Biarritz, mai 1984.

[Date 82] C.J. DATE :
"An introduction to database systems".
Third edition, Addison Wesley 1982.

[FoDa 82] J.D. FOLEY, A. VAN DAM :
"Fundamentals of Interactive Computer Graphics",
Addison Wesley, 1982.

[GoRo 83] A. GOLDBERG, D. ROBSON :
"SMALLTALK-80 : The Language and its Implementation",
Addison Wesley, 1983.

[GuSt 82] L.J. GUIBAS, J. STOLFI :
"A Language for Bitmap Manipulation",
ACM Trans. on Graphics, vol. 1, No 3, july 1982, pp. 191-214.

[Hend 80] P. HENDERSON :
"Functional Programming, Application and Implementation",
Prentice-Hall, 1980.

[McCa 60] J. McCARTHY :
"Recursive Functions of Symbolic Expressions and their Computation by Machine"
ACM Comm. 3(4), 1960, pp. 184-195.

[McGr 80] J.R. McGRAW :
"Data flow computing software development".
IEEE Trans. on Computers, vol. C-29, N$ 12, december 1980.

[NeSp 79] W.M. NEWMANN, R.F. SPROULL :
"Principles of Interactive Computer Graphics",
McGraw-Hill, Second Edition, 1979.

[SmAl 79] B. SMEDLEY, B. ALDRED :
"Problems with geo-data".
Lecture Notes in Computer Science, vol. 81, Data Base Techniques for Pictorial Applications, 1979.

[SyCH 77] J.C. SYRE, D. COMTE, N. HIFDI :
"Pipelining, parallelism and asynchronism in the LAU system".
Int. Conf. on Parallel Processing, august 1977, pp. 87-92.

[VSHM 82] P.D. VAIDYA, L.G. SHAPIRO, R.M. HARALICK, G.J. MINDEN :
"Design and architectural implications of a Spatial Information System".
IEEE Trans. on Computers, vol. C-31, N$ 10, october 1982.

[YaKu 82] K. YAMAGUCHI, T.L. KUNII :
"PICCOLO logic for a picture database computer and its implémentation".
IEEE Trans. on Computers, vol. C-31, N$ 10, october 1982.

[Zloo 75] M.M. ZLOOF :
"Query By Example".
International Conference on Very Large Data Bases, ACM 1975.