Un modèle relationnel et une architecture pour les systèmes de bases de données réparties : application au projet Polypheme

Michel Adiba

THESE

présentée à

Université Scientifique et Médicale de Grenoble

Institut National Polytechnique de Grenoble

pour obtenir le grade de
DOCTEUR ES SCIENCES
« Mathématiques »

par

Michel ADIBA

UN MODELE RELATIONNEL ET UNE ARCHITECTURE POUR LES SYSTEMES DE BASES DE DONNEES REPARTIES
APPLICATION AU PROJET POLYPHEME

Thèse soutenue le 23 septembre 1978 devant la Commission d'Examen :

Président : L. BOLLIET

Examinateurs : J.C. BOUSSARD
C. DELOBEL
C. GIRAULT
S. KRAKOWIAK
J. LE BIHAN
F. PECCOUD

THESE

présentée à

Université Scientifique et Médicale de Grenoble

Institut National Polytechnique de Grenoble

pour obtenir le grade de
DOCTEUR ES SCIENCES
« Mathématiques »

par

Michel ADIBA

UN MODELE RELATIONNEL ET UNE ARCHITECTURE POUR LES SYSTEMES DE BASES DE DONNEES REPARTIES

APPLICATION AU PROJET POLYPHEME

Thèse soutenue le 23 septembre 1978 devant la Commission d'Examen :

Président : L. BOLLIET

Examinateurs : J.C. BOUSSARD
C. DELOBEL
C. GIRAULT
S. KRAKOWIAK
J. LE BIHAN
F. PECCOUD

UNIVERSITE SCIENTIFIQUE
ET MEDICALE DE GRENOBLE

Monsieur Gabriel CAU : Président
Monsieur Pierre JULLIEN : Vice Président

MEMBRES DU CORPS ENSEIGNANT DE L'U.S.M.G.

PROFESSEURS TITULAIRES

MM.	AMBLARD Pierre	Clinique de dermatologie
	ARNAUD Paul	Chimie
	ARVIEU Robert	I.S.N.
	AUBERT Guy	Physique
	AYANT Yves	Physique approfondie
Mme.	BARBIER Marie-Jeanne	Electrochimie
MM.	BARBIER Jean-Claude	Physique expérimentale
	BARBIER Reynold	Géologie appliquée
	BARJON Robert	Physique nucléaire
	BARNOU Fernand	Biosynthèse de la cellulose
	BARRA Jean-René	Statistiques
	BARRIE Joseph	Clinique chirurgicale
	BEAUDOING André	Clinique de pédiatrie et puériculture
	BELORIZKY Elie	Physique
	BERNARD Alain	Mathématiques pures
Mme.	BERTRANDIAS Françoise	Mathématiques pures
MM.	BERTRANDIAS Jean-Paul	Mathématiques pures
	BEZEZ Henri	Pathologie chirurgicale
	BLAMBERT Maurice	Mathématiques pures
	BOLLIET Louis	Informatique (IUT B)
	BONNET Jean-Louis	Clinique ophtalmologique
	BONNET-EYMARD Joseph	Clinique gastro-entérologique
Mme.	BONNIER Marie-Jeanne	Chimie générale
MM.	BOUCHERLE André	Chimie et toxicologie
	BOUCHEZ Robert	Physique nucléaire
	BOUSSARD Jean-Claude	Mathématiques appliquées
	BOUTET DE MONVEL Louis	Mathématiques pures
	BRAVARD Yves	Géographie
	CABANEL Guy	Clinique rhumatologique et hydrologique
	CALAS François	Anatomie
	CARLIER Georges	Biologie végétale
	CARRAZ Gilbert	Biologie animale et pharmocodynamie
	CAU Gabriel	Médecine légale et toxicologie
	CAUQUIS Georges	Chimie organique
	CHABAUTY Claude	Mathématiques pures
	CHARACHON Robert	Clinique oto-rhino-laryngologique
	CHATEAU Robert	Clinique de neurologie
	CHIBON Pierre	Biologie animale
	COEUR André	Pharmacie chimique et chimie analytique
	CONTAMIN Robert	Clinique gynécologique
	COUDERC Pierre	Anatomie pathologique

Mme.	DEBELMAS Anne-Marie	Matière médicale
MM.	DEBELMAS Jacques	Géologie générale
	DEGRANGE Charles	Zoologie
	DELORMAS Pierre	Pneumophtisiologie
	DEPORTES Charles	Chimie minérale
	DESRE Pierre	Métallurgie
	DESSAUX Georges	Physiologie animale
	DODU Jacques	Mécanique appliquée (IUT I)
	DOLIQUE Jean-Michel	Physique des plasmas
	DREYFUS Bernard	Thermodynamique
	DUCROS Pierre	Cristallographie
	GAGNAIRE Didier	Chimie physique
	GALVANI Octave	Mathématiques pures
	GASTINEL Noël	Analyse numérique
	GAVEND Michel	Pharmacologie
	GEINDRE Michel	Electroradiologie
	GERBER Robert	Mathématiques pures
	GERMAIN Jean-Pierre	Mécanique
	GIRAUD Pierre	Géologie
	JANIN Bernard	Géographie
	KAHANE André	Physique générale
	KOSZUL Jean-Louis	Mathématiques pures
	KLEIN Joseph	Mathématiques pures
	KRAVTCHENKO Julien	Mécanique
	KUNTZMANN Jean	Mathématiques appliquées
	LACAZE Albert	Thermodynamique
	LACHARME Jean	Biologie végétale
Mme.	LAJZEROWICZ Janine	Physique
MM.	LAJZEROWICZ Joseph	Physique
	LATREILLE René	Chirurgie générale
	LATURAZE Jean	Biochimie pharmaceutique
	LAURENT Pierre-Jean	Mathématiques Appliquées
	LEDRU Jean	Clinique médicale B
	LE ROY Philippe	Mécanique (IUT I)
	LLIBOUTRY Louis	Géophysique
	LOISEAUX Pierre	Sciences nucléaires
	LONGEQUEUE Jean-Pierre	Physique nucléaire
	LOUP Jean	Géographie
Melle	LUTZ Elisabeth	Mathématiques pures
MM.	MALINAS Yves	Clinique obstétricale
	MARTIN-NOEL Pierre	Clinique cardiologique
	MAZARE Yves	Clinique médicale A
	MICHEL Robert	Minéralogie et pétrographie
	MICOUD Max	Clinique maladies infectieuses
	MOURIQUAND Claude	Histologie
	MOUSSA André	Chimie nucléaire
	NOZIERES Philippe	Spectrométrie physique
	OZENDA Paul	Botanique
	PAYAN Jean-Jacques	Mathématiques pures
	PEBAY-PEYROULA Jean-Claude	Physique
	PERRET Jean	Semeiologie médicale (Neurologie)
	RASSAT André	Chimie systématique
	RENARD Michel	Thermodynamique
	REVOL Michel	Urologie
	RINALDI Renaud	Physique
	DE ROUGEMONT Jacques	Neuro-chirurgie
	SEIGNEURIN Raymond	Microbiologie et Hygiène
	SENGEL Philippe	Zoologie
	SIBILLE Robert	Construction mécanique (IUT I)

MM.	SOUTIF Michel	Physique générale
	TANCHE Maurice	Physiologie
	TRAYNARD Philippe	Chimie générale
	VAILLANT François	Zoologie
	VALENTIN Jacques	Physique nucléaire
	VAUQUOIS Bernard	Calcul électronique
Mme.	VERAIN Alice	Pharmacie galénique
MM.	VERAIN André	Physique
	VEYRET Paul	Géographie
	VIGNAIS Pierre	Biochimie médicale

PROFESSEURS ASSOCIES

MM.	CRABBE Pierre	CERMO
	DEMBICKI Eugéniuz	Mécanique
	JOHNSON Thomas	Mathématiques appliquées
	PENNEY Thomas	Physique

PROFESSEURS SANS CHAIRE

Melle	AGNIUS-DELORD Claudine	Physique pharmaceutique
	ALARY Josette	Chimie analytique
MM.	AMBROISE-THOMAS Pierre	Parasitologie
	ARMAND Gilbert	Géographie
	BENZAKEN Claude	Mathématiques appliquées
	BIAREZ Jean-Pierre	Mécanique
	BILLET Jean	Géographie
	BOUCHET Yves	Anatomie
	BRUGEL Lucien	Energétique (IUT I)
	BUISSON René	Physique (IUT I)
	BUTEL Jean	Orthopédie
	COHEN ADDAD Pierre	Spectrométrie physique
	COLOMB Maurice	Biochimie
	CONTE René	Physique (IUT I)
	DELOBEL Claude	M.I.A.G.
	DEPASSEL Roger	Mécanique des fluides
	FONTAINE Jean-Marc	Mathématiques pures
	GAUTRON René	Chimie
	GIDON Paul	Géologie et minéralogie
	GLENAT René	Chimie organique
	GROULADE Joseph	Biologie médicale
	HACQUES Gérard	Calcul numérique
	HOLLARD Daniel	Hématologie
	HUGONOT Robert	Hygiène et médecine préventive
	IDELMAN Simon	Physiologie animale
	JOLY Jean-René	Mathématiques pures
	JULLIEN Pierre	Mathématiques appliquées
Mme.	KAHANE Josette	Physique
MM.	KRAKOWIACK Sacha	Mathématiques appliquées
	KUHN Gérard	Physique (IUT I)
	LUU DUC Cuong	Chimie organique
	MAYNARD Roger	Physique du solide
Mme.	MINIER Colette	Physique (IUT I)
MM.	PELMONT Jean	Biochimie
	PERRIAUX Jean-Jacques	Géologie et minéralogie
	PFISTER Jean-Claude	Physique du solide
Melle	PIERY Yvette	Physiologie animale

MM.	RAYNAUD Hervé	M.I.A.G.
	REBECQ Jacques	Biologie (CUS)
	REYMOND Jean-Charles	Chirurgie générale
	RICHARD Lucien	Biologie végétale
Mme.	RINAUDO Marguerite	Chimie macromoléculaire
MM.	ROBERT André	Chimie papetière
	SARRAZIN Roger	Anatomie et chirurgie
	SARROT-REYNAULD Jean	Géologie
	SIROT Louis	Chirurgie générale
Mme.	SOUTIF Jeanne	Physique générale
MM.	STIEGLITZ Paul	Anesthésiologie
	VIALON Pierre	Géologie
	VAN CUTSEM Bernard	Mathématiques appliquées

MAITRES DE CONFERENCES ET MAITRES DE CONFERENCES AGREGES

MM.	ARMAND Yves	Chimie (IUT I)
	BACHELOT Yvan	Endocrinologie
	BARGE Michel	Neuro-chirurgie
	BEGUIN Claude	Chimie organique
Mme	BERIEL Hélène	Pharmacodynamie
MM.	BOST Michel	Pédiatrie
	BOUCHARLAT Jacques	Psychiatrie adultes
Mme.	BOUCHE Liane	Mathématiques (CUS)
MM.	BRODEAU François	Mathématiques (IUT B) (Personne étrangère habilitée à être directeur de thèse)
	CHAMBAZ Edmond	Biochimie médicale
	CHAMPETIER Jean	Anatomie et organogénèse
	CHARDON Michel	Géographie
	CHERADAME Hervé	Chimie papetière
	CHIAVERINA Jean	Biologie appliquée (EFP)
	CONTAMIN Charles	Chirurgie thoracique et cardio-vasculaire
	CORDONNIER Daniel	Néphrologie
	COULOMB Max	Radiologie
	CROUZET Guy	Radiologie
	CYROT Michel	Physique du solide
	DENIS Bernard	Cardiologie
	DOUCE Roland	Physiologie végétale
	DUSSAUD René	Mathématiques (CUS)
Mme.	ETERRADOSSI Jacqueline	Physiologie
MM.	FAURE Jacques	Médecine légale
	FAURE Gilbert	Urologie
	GAUTIER Robert	Chirurgie générale
	GIDON Maurice	Géologie
	GROS Yves	Physique (IUT I)
	GUIGNIER Michel	Thérapeutique
	GUITTON Jacques	Chimie
	HICTER Pierre	Chimie
	JALBERT Pierre	Histologie
	JULIEN-LAVILLAVROY Claude	O.R.L.
	KOLODIE Lucien	Hématologie
	LE NOC Pierre	Bactériologie-virologie
	MACHE Régis	Physiologie végétale
	MAGNIN Robert	Hygiène et médecine préventive
	MALLION Jean-Michel	Médecine du travail
	MARECHAL Jean	Mécanique (IUT I)
	MARTIN-BOUYER Michel	Chimie (CUS)
	MICHOULIER Jean	Physique (IUT I)

MAITRES DE CONFERENCES

MM.	ANCEAU François	Mathématiques appliquées
	CHARTIER Germain	Electronique
	GUYOT Pierre	Chimie minérale
	IVANES Marcel	Electrotechnique
	JOUBERT Jean-Claude	Physique du solide
	MORET Roger	Electrotechnique nucléaire
	PIERRARD Jean-Marie	Mécanique
	SABONNADIERE Jean-Claude	Informatique fondamentale et appliquée
Mme.	SAUCIER Gabrièle	Informatique fondamentale et appliquée

MAITRE DE CONFERENCES ASSOCIE

M.	LANDAU Ioan	Automatique

CHERCHEURS DU C.N.R.S. (Directeur et Maîtres de Recherche)

MM.	FRUCHART Robert	Directeur de Recherche
	ANSARA Ibrahim	Maître de Recherche
	CARRE René	Maître de Recherche
	DRIOLE Jean	Maître de Recherche
	MATHIEU Jean-Claude	Maître de Recherche
	MUNIER Jacques	Maître de Recherche

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Monsieur Philippe TRAYNARD : Président

Monsieur Pierre-Jean LAURENT : Vice Président

PROFESSEURS TITULAIRES

MM.	BENOIT Jean	Radioélectricité
	BESSON Jean	Electrochimie
	BLOCH Daniel	Physique du solide
	BONNETAIN Lucien	Chimie minérale
	BONNIER Etienne	Electrochimie et électrométallurgie
	BOUDOURIS Georges	Radioélectricité
	BRISSONNEAU Pierre	Physique du solide
	BUYLE-BODIN Maurice	Electronique
	COUMES André	Radioélectricité
	DURAND Francis	Métallurgie
	FELICI Noël	Electrostatique
	FOULARD Claude	Automatique
	LESPINARD Georges	Mécanique
	MOREAU René	Mécanique
	PARIAUD Jean-Charles	Chimie-Physique
	PAUTHENET René	Physique du solide
	PERRET René	Servomécanismes
	POLOUJADOFF Michel	Electrotechnique
	SILBER Robert	Mécanique des fluides

PROFESSEUR ASSOCIE

M.	ROUXEL Roland	Automatique

PROFESSEURS SANS CHAIRE

MM.	BLIMAN Samuel	Electronique
	BOUVARD Maurice	Génie mécanique
	COHEN Joseph	Electrotechnique
	LACOUME Jean-Louis	Géophysique
	LANCIA Roland	Electronique
	ROBERT François	Analyse Numérique
	VEILLON Gérard	Informatique fondamentale et appliquée
	ZADWORNY François	Electronique

MM.	NEGRE Robert	Mécanique (IUT I)
	NEMOZ Alain	Thermodynamique
	NOUGARET Marcel	Automatique (IUT I)
	PARAMELLE Bernard	Pneumologie
	PECCOUD François	Analyse (IUT B) (Personnalité étrangère habilité à être directeur de thèse)
	PEFFEN René	Métallurgie (IUT I)
	PERRIER Guy	Géophysique-Glaciologie
	PHELIP Xavier	Rhumatologie
	RACHAIL Michel	Médecine interne
	RACINET Claude	Gynécologie et obstétrique
	RAMBAUD André	Hygiène et hydrologie (Pharmacie)
	RAMBAUD Pierre	Pédiatrie
	RAPHAEL Bernard	Stomatologie
Mme.	RENAUDET Jacqueline	Bactériologie (Pharmacie)
MM.	ROBERT Jean-Bernard	Chimie physique
	Romier Guy	Mathématiques (IUT B) (Personnalité étrangère habilité à être directeur de thèse)
	SCHAERER René	Cancérologie
	SHOM Jean-Claude	Chimie générale
	STOEBNER Pierre	Anatomie pathologie
	VROUSOS Constantin	Radiologie

MAITRES DE CONFERENCES ASSOCIES

MM.	DEVINE Roderick	Spectro physique
	HODGES Christopher	Transition de phases

Fait à SAINT MARTIN D'HERES, NOVEMBRE 1976.

Je tiens particulièrement à remercier ici :

Monsieur Louis BOLLIET, Professeur à l'Université de Grenoble, qui m'a toujours accordé sa confiance et son soutien. Je lui suis très reconnaissant pour l'honneur qu'il me fait de présider ce jury ;

Monsieur Claude DELOBEL, Professeur à l'Université de Grenoble, qui est à l'origine de cette thèse. Je veux aussi lui exprimer ma plus sincère gratitude pour la confiance qu'il m'a toujours témoignée, pour sa totale disponibilité à mon égard, pour ses nombreux conseils et encouragements sans lesquels ce travail n'aurait pas été possible ;

Monsieur Claude GIRAULT, Professeur à l'Université de Paris VI, qui a accepté de juger ce travail ;

Monsieur Jean-Claude BOUSSARD, Professeur à l'Université de Nice, qui a guidé mes premiers pas d'enseignant d'Informatique et qui a bien voulu participer au jury ;

Messieurs Sacha KRAKOWIAK et François PECCOUD, Professeurs à l'Université de Grenoble, qui ont manifesté leur intérêt pour ce travail et ont accepté de le juger ;

Monsieur Jean LE BIHAN, Directeur du projet pilote IRIA "SIRIUS", pour l'intérêt qu'il a manifesté au projet POLYPHEME et pour les moyens matériels qu'il lui a fournis.

Je suis également reconnaissant :

à Monsieur Jean-Raymond ABRIAL, dont les travaux m'ont servi de point de départ et qui par ses critiques et suggestions m'a beaucoup apporté ;

à Michel LEONARD pour sa collaboration et pour sa lecture attentive du manuscrit. Les discussions que nous avons eues ont permis d'améliorer le présent document ;

à tous les membres de l'équipe POLYPHEME sans lesquels ce travail n'aurait pu être mené à bien : Edouard ANDRE, Juan-Manuel ANDRADE, Jean-Yves CALECA, Paul DECITRE, Christian EUZET, Andrée STIERS, PAIK-IN-SUP, ainsi qu'à tous ceux qui par leur compétence et leur collaboration m'ont beaucoup aidé, en particulier : Jean-Claude CHUPIN, Dominique PORTAL, Jean-Pierre GIRAUDIN ;

aux participants au projet SIRIUS en particulier : Robert DEMOLOMBE, Christian ESCULIER, Georges GARDARIN, Mireille JOUVE, Stefano SPACCAPIETRA, Witold LITWIN pour les fructueuses discussions que nous avons eues ;

à Madame Marie-José DOREL qui avec compétence, soin, rapidité et bonne humeur a assuré la dactylographie, aidée par Madame Clotilde CHALAND ;

au Service de Reprographie, en particulier à Messieurs IGLESIAS et ANGUILLE, pour la réalisation matérielle de cet ouvrage.

SOMMAIRE

N.B. Chaque Chapitre est précédé d'un sommaire plus détaillé.

Thèse Michel ADIBA

Errata

- Ajouter à la fin de la page 1.27 :

Ce qui donne :

R1

NUS	VILLE
u1	PARIS
u2	LYON
u3	PARIS

Site S1

R2

NP	PRIX
p1	25
p2	15
p3	52
p4	12

Site S2

R3

NUS	NP	QTE
u1	p1	3
u1	p2	6
u1	p3	2
u2	p3	14
u2	p4	25
u3	p2	17

Site S3

Entre S1 et S2, il y a partition du schéma mais également absence de lien entre les informations décrites, tandis qu'entre S1 et S3 ou entre S2 et S3 il existe un lien par le fait que NUS se retrouve dans S1 et dans S3 (NP dans S2 et S3).

Cas 2.2. Duplication

Considérons la répartition suivante :

Sites	Relations
S1	R1
S2	R2
S3	R1,R2,R3

- page 2.44 à la 21^{e} ligne, lire tableau 2.10 au lieu de tableau 2.11.

Avant-Propos

Notre travail aborde un domaine informatique auquel sont consacrés aujourd'hui de nombreux efforts de recherche et de développement : les bases de données réparties.

Notre contribution se situe sur le plan de la conception et de la réalisation de logiciels généraux pour construire et exploiter des bases de données réparties, par analogie avec les systèmes de gestion de bases de données actuellement disponibles.

Ce sujet constitue au sein de l'Université de Grenoble le projet POLYPHEME auquel collaborent l'équipe "bases de données" et l'équipe "réseaux d'ordinateurs" du Centre Scientifique CII-HB.

En ce qui nous concerne, nous avons essayé plus particulièrement d'aborder les problèmes liés à la conception de bases réparties dans des environnements logiciels et matériels hétérogènes, en définissant un ensemble d'outils conceptuels de modélisation : MOGADOR (MOdèle GénérAl de DOnnées Réparties).

Ces outils appliqués à la conception du système de bases de données réparties lui-même nous ont permis de définir une architecture fonctionnelle servant de cadre de référence à la réalisation effective d'une maquette.

POLYPHEME :

Etrangers, votre nom ? D'où nous arrivez-vous sur les routes des ondes ? Faites-vous le commerce ? N'êtes-vous que pirates qui follement courrez et croisez sur les flots et, risquant votre vie, vous en allez piller les côtes étrangères ?

Homère, ODYSSEE IX 219-258.

0. Introduction

0.1. L'INFORMATIQUE : DES OUTILS PUIS DES METHODES

Quand on regarde l'évolution de l'informatique, il est frappant de constater que jusqu'à la fin des années 60, de nombreux outils sont proposés, rarement des méthodes.

Cela est vrai aussi bien pour la multitude de langages de programmation qui voient le jour durant cette période, que pour tous les systèmes logiciels, systèmes d'exploitation en particulier. Cela est vrai également pour les systèmes de bases de données consacrés à la définition et à la manipulation de gros volumes d'informations.

On constate alors qu'un outil, aussi performant soit-il, demeure inutile tant que l'on en n'a pas acquis la maîtrise.

Sur le plan des langages, il faut attendre le début des années 70 pour qu'apparaissent les méthodes de programmation structurées [G11].

Dans le domaine des bases de données, on se préoccupe, seulement depuis ces dernières années, des problèmes logiques liés à la conception des bases, avec l'apparition de modèles dits "conceptuels", capables de décrire le plus fidèlement possible une partie du monde réel, pour arriver à des bases de données plus cohérentes [B29].

En fait, le fil directeur de ces travaux est d'arriver à une indépendance de plus en plus grande entre les données et les programmes qui les manipulent ou les manipuleront.

Avec l'apparition des réseaux d'ordinateurs et l'évolution du matériel dont le coût diminue tandis que les performances augmentent, de nouveaux problèmes sont abordés et résolus pour aboutir à des réseaux de calculateurs opérationnels et fiables [R49]. A partir de ces outils, peuvent se développer des "applications réparties" [R21]. Le point de convergence entre ces applicationset les bases de données constitue le domaine des bases de données réparties auquel notre travail se rattache.

0.2. LE DOMAINE DES BASES DE DONNEES REPARTIES

Ce domaine constitue, à l'heure actuelle, un vaste champ de recherches et d'applications qui préoccupe de nombreuses équipes (constructeurs, utilisateurs, chercheurs) tant à l'étranger qu'en France avec, en 1976, la création d'un projet pilote de l'Institut de Recherche en Informatique et Automatique (IRIA) : le projet SIRIUS - Bases de Données Réparties [R42]. Notre participation à ce projet nous a permis d'avoir des contacts intéressants avec d'autres équipes ainsi que de nombreuses et fructueuses discussions.

S'il existe à l'heure actuelle de nombreux logiciels de bases de données commercialisés, il n'existe pas encore de logiciels *généraux* de bases de données *réparties*. La principale raison en est que ce domaine est encore à ses débuts et que les concepts fondamentaux n'ont pas été tous complètement mis à jour ni bien définis. Notre travail est une tentative de clarification de ces concepts.

Se situant au point de convergence du domaine des bases de données et des réseaux d'ordinateurs, les bases de données réparties héritent des travaux déjà effectués dans ces deux domaines, mais également des problèmes qui n'y sont pas encore résolus. En outre, étant donné l'antagonisme entre "base de données" synonyme de centralisation (d'intégration) et "réseaux" synonyme de décentralisation (de répartition), les approches sont très différentes selon que des spécialistes réseaux s'intéressent aux bases de données ou au contraire que des spécialistes bases de données s'intéressent aux réseaux.

Le problème que nous avions à résoudre dans le cadre du projet grenoblois POLYPHEME, auquel collaboraient l'équipe réseaux d'ordinateurs du Centre Scientifique CII-HB, et l'équipe bases de données du Laboratoire d'Informatique de l'Université Scientifique et Médicale, concernait la conception et la réalisation d'un système permettant de faire coopérer des bases de données hétérogènes réparties sur les différents sites d'un réseau général d'ordinateurs.

Dans ce projet une approche volontairement ascendante est adoptée : on part a priori de bases de données existantes et en exploitation sur différents calculateurs d'un réseau et on cherche à les faire coopérer de manière à ce qu'elles forment, pour une nouvelle communauté d'utilisateurs

une base de données répartie.

Cette approche soulève les problèmes suivants [R1, R2] :

1) Homogénéiser les différentes bases locales qui, a priori, peuvent être implantées sous des systèmes de bases de données très disparates (SOCRATE pour l'une, I.M.S. pour l'autre par exemple) ;

2) Concevoir une vue d'ensemble (ou "vue globale") des informations ainsi réparties ;

3) Concevoir et réaliser un système logiciel pour permettre l'implantation de cette vue globale et l'exploitation de la base répartie ainsi créée. Ce système logiciel constitue un système de gestion de bases de données réparties ou SGBDR.

Dans ces problèmes, la dualité outil-méthode nous a conduit à trouver un point d'équilibre entre l'aspect méthodologique de la conception des bases de données réparties et l'aspect technique de spécification et de réalisation d'un système de coopération.

0.3. PROBLEMES TRAITES : MODELES ET ARCHITECTURE DE BASES REPARTIES

Avant d'aborder le domaine des bases de données réparties, nous avons travaillé au problème de la correspondance entre les modèles hiérarchiques ou réseaux et le modèle relationnel de Codd [B17, B3] ou modèle n-aire. Nous avons montré alors comment telle structure de base de données réseaux peut être transposée en termes relationnels [B4, B6].

Dans POLYPHEME, nous avons choisi d'utiliser le modèle relationnel comme modèle commun de données en ce sens que toutes les informations réparties dans les bases locales sont vues de manière externe comme des tables de valeurs (relations n-aires).

Cependant, pour définir le contenu de ces tables de façon à traduire plus précisément la sémantique des informations réparties, nous avons été amenés à définir un MOdèle GénérAl de DOnnées Réparties : MOGADOR.

MOGADOR se présente comme un ensemble d'outils conceptuels pour la description d'informations et pour l'expression des opérations qu'il est possible de leur appliquer. Il facilite le passage entre les deux niveaux suivants de modélisation :

- le niveau conceptuel où l'on s'intéresse à l'aspect sémantique du monde réel tel qu'il est perçu par notre esprit,
- le niveau externe où toutes les informations sont vues de manière homogène comme des tableaux de valeurs (relations n-aires) construits à partir de l'analyse fournie par le niveau conceptuel.

Pour chacun de ces niveaux, des opérations primitives sont définies et un langage de description et de manipulation de données relationnelles réparties est proposé.

S'il existe bon nombre de modèles pour les bases de données centralisées, il n'existe pas, à notre connaissance, de modèles pour les bases réparties. En outre, contrairement à de nombreux modèles qui n'offrent qu'un aspect statique de description, MOGADOR offre en plus un aspect dynamique grâce aux opérations primitives et à la notion de machine. MOGADOR repose sur la théorie des ensembles ainsi que sur les notions développées par J.R. ABRIAL dans Data Semantics [B1]. La description d'ensembles d'informations (ou catégories) associée à la description des opérations qu'il est possible de leur appliquer présente beaucoup d'analogie avec tous les travaux qui concernent les types abstraits de données [G15]. Un point de convergence entre ces travaux et ceux qui concernent les bases de données constitue un axe intéressant de recherche [B65].

Grâce à ses caractéristiques et à sa généralité, MOGADOR nous permet de traiter les problèmes suivants :

1) Le problème de l'homogénéisation non seulement des informations réparties, mais également des moyens de stockage, d'accès et de manipulation des bases locales. Chacune d'elles est vue comme un automate standard [R1, R3].

2) Le problème de la répartition des informations : nous montrons que cette notion recouvre les deux notions de partition (au sens mathématique) et de duplication permettant ainsi de caractériser différents cas de répartition.
MOGADOR permet également de définir la plus petite information qu'il est possible de répartir, sans risque d'incohérence pour la base globale : l'atome sémantique de répartition [R5]. En considérant l'aspect dynamique des informations réparties, nous abordons aussi le problème de la décomposition d'une transaction s'adressant à la vue globale en transactions s'adressant à chaque base locale concernée. Nous définissons alors des cas standard de répartition pour lesquels la répercussion des opérations du niveau global sur le niveau local peut être automatiquement effectuée [R6, R17].

3) Le problème de la conception de l'outil que constitue un SGBDR : l'architecture d'un tel système est définie comme un réseau logique de machines relationnelles abstraites [R4, R7] et constitue le cadre général dans lequel se développe l'implantation effective d'un prototype [P12].

0.4. ORGANISATION DU PRESENT DOCUMENT

Dans ce qui suit, nous consacrons le Chapitre 1 à la présentation des notions fondamentales concernant les bases de données et les systèmes répartis avant de définir plus précisément le domaine des bases réparties et les principaux problèmes qu'elles posent. Nous évoquons également les recherches en cours, tout en présentant le projet POLYPHEME, cadre de notre travail.

Le Chapitre 2 est consacré à la présentation des concepts de base du modèle MOGADOR : les éléments, les catégories d'éléments, les fonctions entre catégories et les opérations du niveau conceptuel. Le passage de ce niveau au niveau des relations n-aires est décrit en détail et des exemples sont donnés. C'est également au chapitre 2 qu'est présenté le formalisme pour décrire des vues relationnelles et exprimer des opérations sur ces vues.

Le Chapitre 3 traite de l'application de MOGADOR au niveau de bases de données hétérogènes : les bases locales ou bases coopérantes. Nous montrons comment les modèles hiérarchiques et réseaux peuvent être interprétés en termes relationnels, avant de définir la notion de machine locale qui permet de présenter des bases hétérogènes comme des automates standard.

Le Chapitre 4 s'attaque au problème de la conception d'une vue globale d'un ensemble de données réparties. Nous montrons alors comment le niveau conceptuel de MOGADOR permet de caractériser les atomes d'informations pouvant être répartis sans perte de cohérence pour la base globale (atomes sémantiques de répartition). Nous décrivons également les différents cas possibles de répartition et les informations qu'il est nécessaire de connaître au niveau de la vue globale, pour que la manipulation de la base répartie puisse se faire automatiquement : règles globales et processus de décomposition.

Le Chapitre 5 décrit l'architecture d'un système de gestion de base de données réparties comme un ensemble de machines MOGADOR de deux types : les machines locales construites autour d'une base coopérante et les machines globales permettant la définition et la manipulation d'une base répartie. Dans cette partie, l'accent est mis tout d'abord sur le point de vue utilisateur d'un tel système avant d'aborder l'architecture de manière plus détaillée.

Tout au long du chapitre, nous donnons les éléments qui constituent le système de Base Répartie en cours d'implantation sous forme d'une "Maquette POLYPHEME" dans l'environnement matériel et logiciel du Centre de Calcul de Grenoble et du réseau CYCLADES.

Chapitre 1

Bases de Données et Systèmes répartis

Si j'avais appris la technique, je serais technicien. Je fabriquerais des objets compliqués. Des objets très compliqués, de plus en plus compliqués, cela simplifierait l'existence.

E. IONESCO

Sommaire

1.3.4. Problèmes posés par la conception d'une BDR.

1.3.4.1. Création d'une BDR (Approche descendante).

1.3.4.2. Conversion d'une base centralisée en base répartie.

1.3.4.3. Transformation de bases existantes

. Intégration après modification

. Coopération.

1.3.5. Problèmes spécifiques aux systèmes de Bases de Données Réparties.

1.3.5.1. Description de la BDR.

1.3.5.2. Répartition des schémas d'une BDR.

1.3.5.3. Stratégies de traitement des transactions globales.

1.3.5.4. Problème des mises à jour concurrentes dans un SGBDR.

1.3.5.5. Cohérence des copies multiples.

1.3.5.6. Gestion des Pannes dans un SGBDR.

1.4. CONCLUSIONS : Propositions d'un cadre de travail.

1.1. L'APPROCHE BASE DE DONNEES

Dans le monde de l'informatique, les bases de données constituent depuis quelques années un important centre d'intérêt. En effet, en moins de dix ans, on a vu apparaître et se développer toute une série de méthodes et de techniques pour la conception, l'implantation et l'exploitation de gros volumes d'informations structurées.

Historiquement, chaque nouvelle application engendrait ses propres fichiers et ses propres programmes. Il en résultait une duplication non contrôlée des informations avec les dangers que cela impliquait :

- incohérence des informations du fait de la duplication
- déphasages des traitements les uns par rapport aux autres.

La création d'une base de données va à l'encontre de ce procédé : elle rend possible la centralisation, la coordination, l'intégration et la diffusion de l'information.

Une base de données est en effet un ensemble d'informations construit pour mettre en place toute une série d'applications informatiques destinées à une grande variété d'utilisateurs.

De ce fait, on désire obtenir :

- une meilleure cohérence des informations
- une réduction des redondances
- une réduction des efforts de saisie et de mise à jour.

Cependant, cette centralisation de l'information sous forme d'une base de données unique n'est pas sans soulever de nombreux problèmes :

- la base est vulnérable face aux risques de destruction : mauvais fonctionnement du matériel, erreurs humaines, action délibérée de destruction.
- le fait d'avoir une communauté d'utilisateurs exploitant simultanément la même base pose des problèmes liés au partage des données et à la confidentialité.

- la construction d'une base de données dans la mesure où elle permet la collecte de nombreuses informations sur des individus ou des organismes divers pose des problèmes d'ordre sociologique et juridique comme en témoignent les préoccupations des pouvoirs publics à l'étranger et en France [G12, G13].

Sur le plan informatique, on assiste depuis la fin des années 60 au développement de logiciels permettant la création, la mise à jour, l'exploitation des bases de données : les Systèmes de Gestion de Bases de Données ou SGBD [B10, G3, G10].

De plus, un grand effort de recherche est effectué dans ce domaine pour assurer davantage l'*intégrité* des bases et la *sécurité* des systèmes : Préserver l'intégrité d'une base de données c'est mettre en oeuvre les moyens techniques pour assurer la qualité de l'information enregistrée. Il faut s'assurer qu'au cours de son existence, l'information ne subit pas de déformation préjudiciable.

La distinction entre les notions de sécurité et d'intégrité peut être illustrée par le tableau suivant :

Préjudice	Accidentel	Délibéré
Destruction non autorisée	INTEGRITE	SECURITE
Modification non autorisée	INTEGRITE	SECURITE
Accès non autorisé	SECURITE	SECURITE

Dans la suite de ce chapitre, nous allons tout d'abord préciser un certain nombre de concepts fondamentaux concernant les bases de données et les SGBD. Ceci nous permettra en fait d'introduire la terminologie qui sera employée ici. Nous verrons ensuite comment l'apparition de la télé-informatique et de la micro-informatique, en combattant la centralisation, ont donné naissance à la notion de *bases de données réparties* qui constitue le thème essentiel de notre travail.

1.1.1. La conception des bases de données : processus de modélisation

Concevoir une base de données c'est tout d'abord modéliser une certaine partie du monde réel pour caractériser les objets, les événements qui présentent de l'intérêt vis-à-vis des applications informatiques que l'on veut mettre en place.

Pour nous, un modèle de données est un ensemble de concepts destinés à décrire la réalité de manière indépendante de tout traitement informatique. Il y a eu au cours de ces dernières années de nombreux travaux sur les modèles de données. C'est un sujet très vaste et dans la mesure où tous les êtres humains n'ont pas la même façon de penser, il s'agit là presque d'un problème philosophique. Ceci explique en particulier l'existence de nombreux modèles de données ayant chacun ses propres concepts et sa propre terminologie. Il est difficile d'établir une comparaison précise des modèles de données car la plupart du temps, ils utilisent des terminologies différentes pour des concepts identiques ou au contraire la même terminologie pour des concepts différents. D'autre part, ils ne fournissent pas tous le même niveau sémantique de description de la réalité [B35].

L'un des plus anciens modèles proposés a été le modèle hiérarchique où les informations sont organisées en arbres (TDMS [B64]).

Le groupe CODASYL, à qui l'on doit COBOL, a proposé en 1971 [B26] un modèle pour les bases de données où les informations sont organisées en *réseaux*. Ces deux types de modèles fournissent des structures qui peuvent se visualiser à l'aide de graphes où chaque noeud représente un type d'entités du monde réel et chaque arête une liaison entre ces entités.

Ces deux familles de modèles sont en fait trop liées à la technique informatique et en particulier aux organisations de fichiers (l'élément de base du modèle CODASYL n'est-il pas le "record" ou enregistrement). Une évolution s'est alors faite dans les recherches sur les modèles de données ayant pour objectifs :

- d'assurer l'indépendance des informations vis-à-vis des supports sur lesquels elles seront stockées et même vis-à-vis des traitements qui leur seront appliqués,
- de traduire au mieux la sémantique des informations afin d'assurer une meilleure cohérence des bases de données,
- de permettre à chaque base d'évoluer au fur et à mesure de l'évolution des besoins des utilisateurs : prendre en compte de nouveaux types d'informations, de nouvelles liaisons, de nouvelles règles sémantiques sans avoir à reconvertir toute la base.

C'est pour résoudre ces problèmes qu'est apparue une troisième catégorie de modèles : les modèles relationnels de données [B3] dont les plus représentatifs sont :

- le modèle relationnel de Codd ou modèle n-aire [B17, B20]
- le modèle entité-relation de Chen [B48]
- le modèle "Data Semantics" d'Abrial [B1].

Basés sur des fondements mathématiques (théorie des ensembles, des relations), ces modèles offrent un vaste domaine de recherches et de développement que l'on peut diviser en trois grandes parties :

- Développements théoriques des modèles relationnels : formes normales des relations, décomposition, relations fonctionnelles, ... [B29, B30, B33].
- Aide à la conception des systèmes d'informations [B39].
- Construction et Développement de Systèmes de Bases de Données fondés sur les modèles relationnels et en particulier sur celui de Codd (cf. § 1.1.3.2) [B9, B60, B63].

Selon le contexte réel dans lequel se situe le processus de modélisation, il faut tout d'abord caractériser les "objets" que l'on désire traiter et les classer par "catégories" : des employés, des étudiants, des comptes bancaires, des clients, des usines, des produits, des commandes, des machines, des vols, des appareils, des escales.

Pour chaque catégorie, il faut considérer les liaisons (les relations) qu'elle peut avoir avec une ou plusieurs autres : un employé travaille dans une usine, une commande concerne plusieurs produits, un vol correspond à un appareil mais à plusieurs escales, ...

Il faut aussi pouvoir décrire de manière plus fine les objets d'une catégorie en caractérisant :

- d'une part ce qui va permettre de les identifier les uns par rapport aux autres : le numéro INSEE pour des employés, un code pour des produits ...
- d'autre part, ce qui va donner des informations supplémentaires : le nom, le salaire, le sexe, l'état-civil, l'âge d'un employé, le titulaire d'un compte bancaire, la ville, le nombre d'employé d'une usine, le total à payer d'une commande. Dans la terminologie classique, on parle alors "d'attributs".

Enfin, il faut pouvoir indiquer un certain nombre de règles sémantiques ou de règles de cohérence que devront vérifier les objets, par exemple :

- un employé a toujours une date de naissance
- l'âge d'un employé doit être compris entre 18 et 65
- le total à payer d'une commande est la somme des prix des produits commandés, chaque prix est obtenu en multipliant la quantité commandée par le prix unitaire du produit
- à une usine donnée et à un produit donné, il correspond une capacité de production
- si on ajoute un nouvel employé, il faut augmenter de 1 le nombre d'employés de l'usine où il travaille.

  ....

Sur ces exemples volontairement simplistes, on s'aperçoit qu'un modèle doit non seulement assurer une description aussi fine que possible des informations, mais également des traitements qui leur seront appliqués. Nous reviendrons sur cet aspect avec la définition d'un modèle général de données réparties (Chapitre 2).

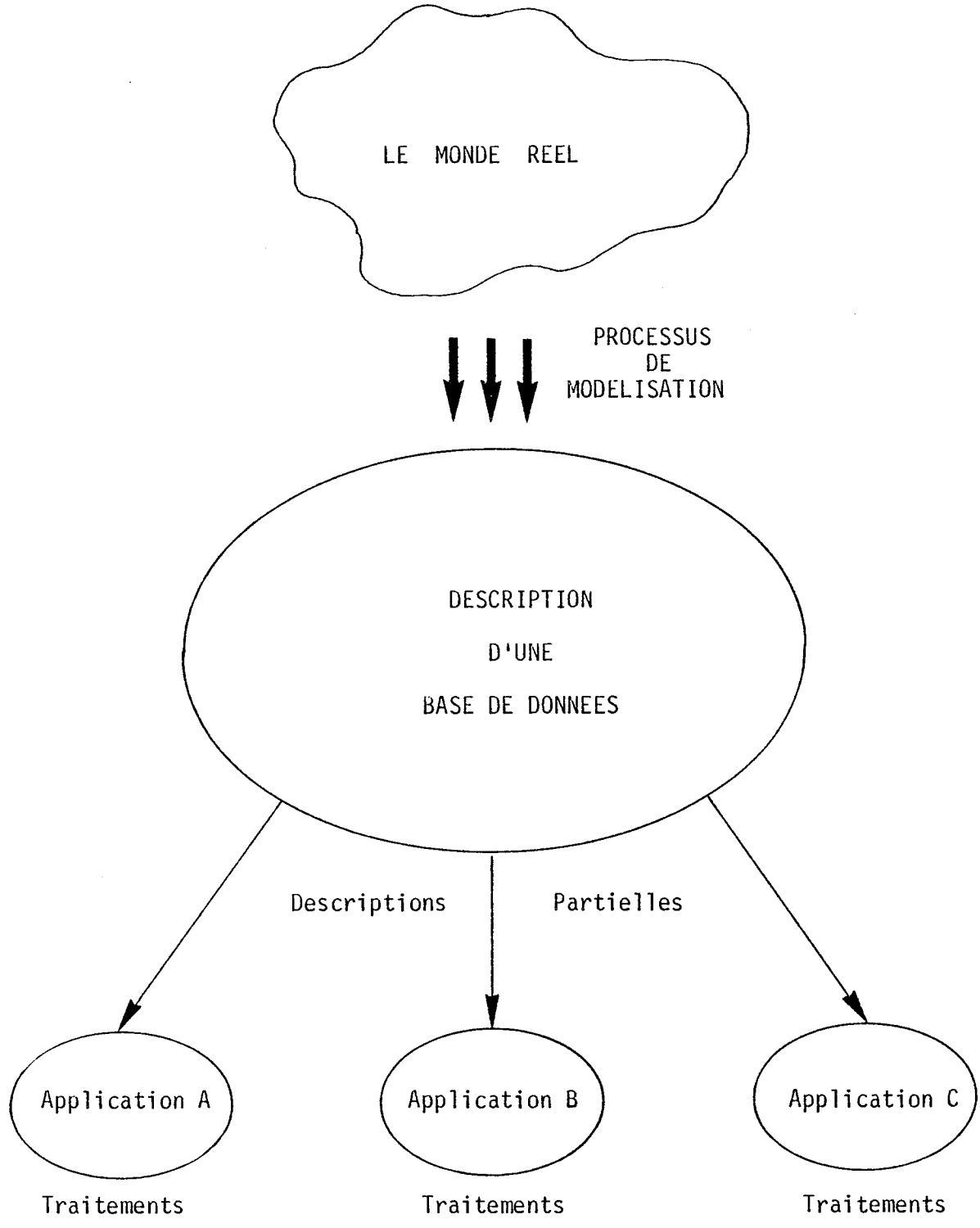


Figure 1.1 - Le Processus de Modélisation

1.1.2. La mise en place et l'exploitation d'une base de données

Pour décrire un ensemble d'informations à l'aide des concepts d'un modèle de données, il faut disposer de langages puis de systèmes informatiques qui permettent la mise en place et l'exploitation de cet ensemble d'informations : les *systèmes de gestion de bases de données* (ou *SGBD*).

Nous allons nous placer d'abord sous l'angle du concepteur de base de données pour caractériser les fonctions principales de ces systèmes avant de les regarder sous un angle plus technique au paragraphe 1.1.3.

1.1.2.1. Les niveaux de description d'une base de données

Dans un programme COBOL, la structure des informations manipulées sous forme de fichiers est décrite par la DATA DIVISION. Du fait de l'indépendance des données vis-à-vis des programmes qui les manipulent, on considère actuellement pour une base de données trois niveaux de description [B7], Figure 1.2 :

- un *niveau conceptuel* où l'on décrit l'ensemble des informations qui constituent la base de données sans tenir compte des applications qui seront faites ni de la représentation physique des données. Fournie par l'administrateur de la base, cette description est exprimée dans un langage de définition de données (LDD) et constitue le *schéma conceptuel* de la base. Traité par le SGBD, ce schéma est en général stocké sous un format interne pour constituer le *catalogue* de la base.

- le *niveau externe* concerne tout ou partie de la base de données vue par une application, c'est-à-dire un ensemble de programmes de traitement. Fournie par l'administrateur d'application, une telle description constitue un *schéma externe* (ou sous-schéma ou encore sous-structure) qui lui aussi sera stocké par le SGBD sous forme de catalogue. Les programmes d'applications se réfèrent alors à ce seul schéma externe.

- le *niveau interne* ou niveau physique concerne la structure des fichiers qui constituent la base. La structure de ces fichiers est importante sur le plan des performances de la base de données.
La présence d'un *schéma interne* décrivant ce niveau et la correspondance entre ce schéma et le schéma conceptuel permet d'assurer l'indépendance des données vis-à-vis des programmes d'accès.

Dans la plupart des SGBD actuels, ces trois niveaux sont en général très peu différenciés. Par exemple dans SOCRATE, les niveaux internes et conceptuels sont regroupés dans la structure tandis que le niveau externe correspond aux sous-structures.

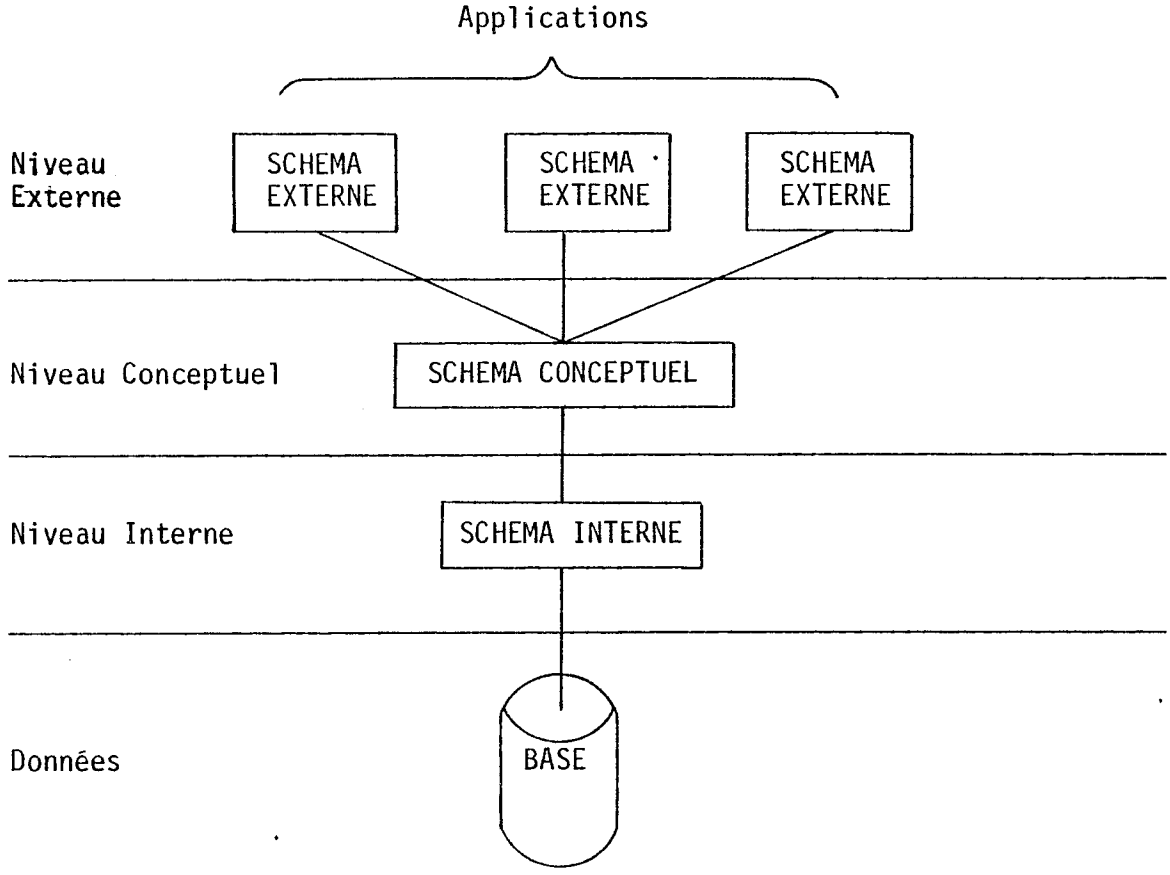


Figure 1.2 - Les Niveaux de Description

1.1.2.2. La manipulation des données

Outre un langage de description, un SGBD offre un langage de manipulation qui se compose de primitives qu'on peut appeler directement ou par l'intermédiaire d'un langage hôte (en général un langage évolué COBOL [B26], PL/1 [B45]).

Ces primitives concernent :

- la recherche dans la base d'un ensemble d'informations que l'on qualifie par certains critères
- l'accès aux informations, c'est-à-dire le transfert de la base vers le programme utilisateur

- la modification d'une information existante
- la création et la suppression d'information.

1.1.2.3. L'exploitation d'une base de données

En général, une base de données est destinée à être exploitée par de nombreux utilisateurs pour des applications diverses.

Ces utilisateurs n'ont pas tous la même vue des données de la base et pour des problèmes liés à la confidentialité des données, ils n'ont pas à voir certaines parties de la base. A un groupe d'utilisateurs qui ont tous la même application en commun, va correspondre un schéma externe et divers programmes d'application.

Nous considérons deux modes principaux d'exploitation :

- le *mode transactionnel* qui consiste pour l'utilisateur (en général non informaticien) à activer des programmes d'application tout faits : par exemple un programme de génération d'état de stock ou création d'une ou plusieurs commandes de produits, etc ...

- le *mode interactif* qui consiste à soumettre au SGBD des requêtes d'interrogation ou de mise à jour. Ces requêtes en général exprimées dans un langage plus proche de la langue naturelle que d'un langage de programmation sont interprétées par le SGBD en primitives de manipulation [B2]. Dans ce cas, ces requêtes correspondent à des actions ponctuelles sur la base.

Dans la pratique, c'est évidemment le premier mode d'exploitation qui est le plus courant.

1.1.3. L'architecture des Systèmes de Gestion de Bases de Données

Il existe actuellement de nombreux SGBD commercialisés [B10, B67]. On peut les classer en deux grandes familles, en fonction du modèle utilisé pour décrire la base :

- famille des SGBD *Hiérarchiques* dont le plus représentatif est I.M.S. (Information Management System) avec ses différentes versions. On y trouve également System 2000 bien qu'en fait ces deux systèmes offrent au niveau des schémas externes des sous-schémas réseaux.

- famille des SGBD *Réseaux* qui englobe tous les systèmes construits selon les spécifications CODASYL comme IDMS (Integrated Data Management System), IDS (Integrated Data Store), DMS 1100 (Data Management System), TOTAL [B67] .... Il faut citer également dans cette famille un système comme SOCRATE conçu à l'Université de Grenoble par l'équipe de J.R. ABRIAL [B2].

La plupart du temps, au niveau interne, ces systèmes gèrent (plus ou moins automatiquement) toute une série de fichiers inverses ou d'index pour permettre des accès plus rapides aux informations.

1.1.3.1. Niveaux et Composants

Un SGBD est un système d'exploitation spécialisé dans le traitement de gros volumes d'informations. C'est pourquoi, dans notre description, nous ignorons volontairement l'aspect système d'exploitation, à savoir : gestion des périphériques et des terminaux d'accès, multiprogrammation ou partage de temps [G2].

En ce qui concerne l'organisation des informations, nous considérons quatre niveaux : Figure 1.3.

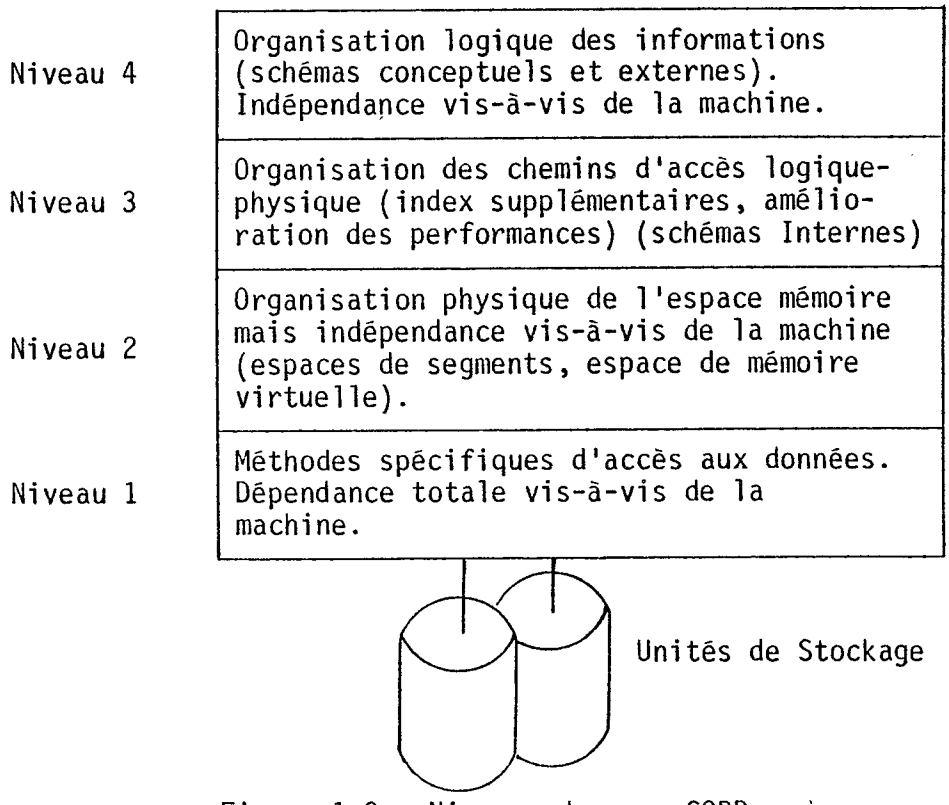


Figure 1.3 - Niveaux dans un SGBD

Ces niveaux se caractérisent par :

- la finesse de description liée aux schémas conceptuels, externes et internes
- les contraintes physiques liées à l'implantation des données dans le cadre d'une machine particulière.

D'autre part, pour réaliser les fonctions vues au § 1.1.2, on considère un ensemble de composants "système" (voir Figure 1.4).

La *base de données* est en fait un ensemble d'éléments qui vont des fichiers constituant la base jusqu'aux programmes d'exploitation, en passant par les différents schémas stockés sous forme interne.

1.1.3.2. L'évolution actuelle en matière de SGBD

Les recherches concernant les bases de données vont actuellement dans les deux grandes directions suivantes :

- la conception de systèmes d'informations qui regroupent tous les travaux sur les modèles conceptuels de données,
- la conception de systèmes de gestion de bases de données construits en particulier autour du modèle relationnel de Codd.

Nous ne nous intéressons pas ici au premier groupe de travaux qui est traité par de nombreux auteurs, par exemple [B36, B39, B47]. Nous allons essayer par contre de décrire brièvement les caractéristiques nouvelles des générations futures de SGBD.

Une première tendance se confirme : les futurs SGBD devront permettre à l'utilisateur de choisir le modèle qui lui convient le mieux : on arrive ainsi à des architectures basées sur la coexistence de modèles différents comme les hiérarchies, les réseaux ou les relations [B9, B46].

De plus, de gros efforts sont consacrés à l'implantation de prototypes de SGBD relationnels dont les plus importants sont :

- un système sous MULTICS [B74]
- le système PRTV (Peterlee Relational Test Vehicle) construit au Centre Scientifique IBM de Peterlee en Grande-Bretagne [B62, B63]
- le système INGRES (Interactive Graphics Relational system) construit à l'Université de Californie (Berkeley) [B60]

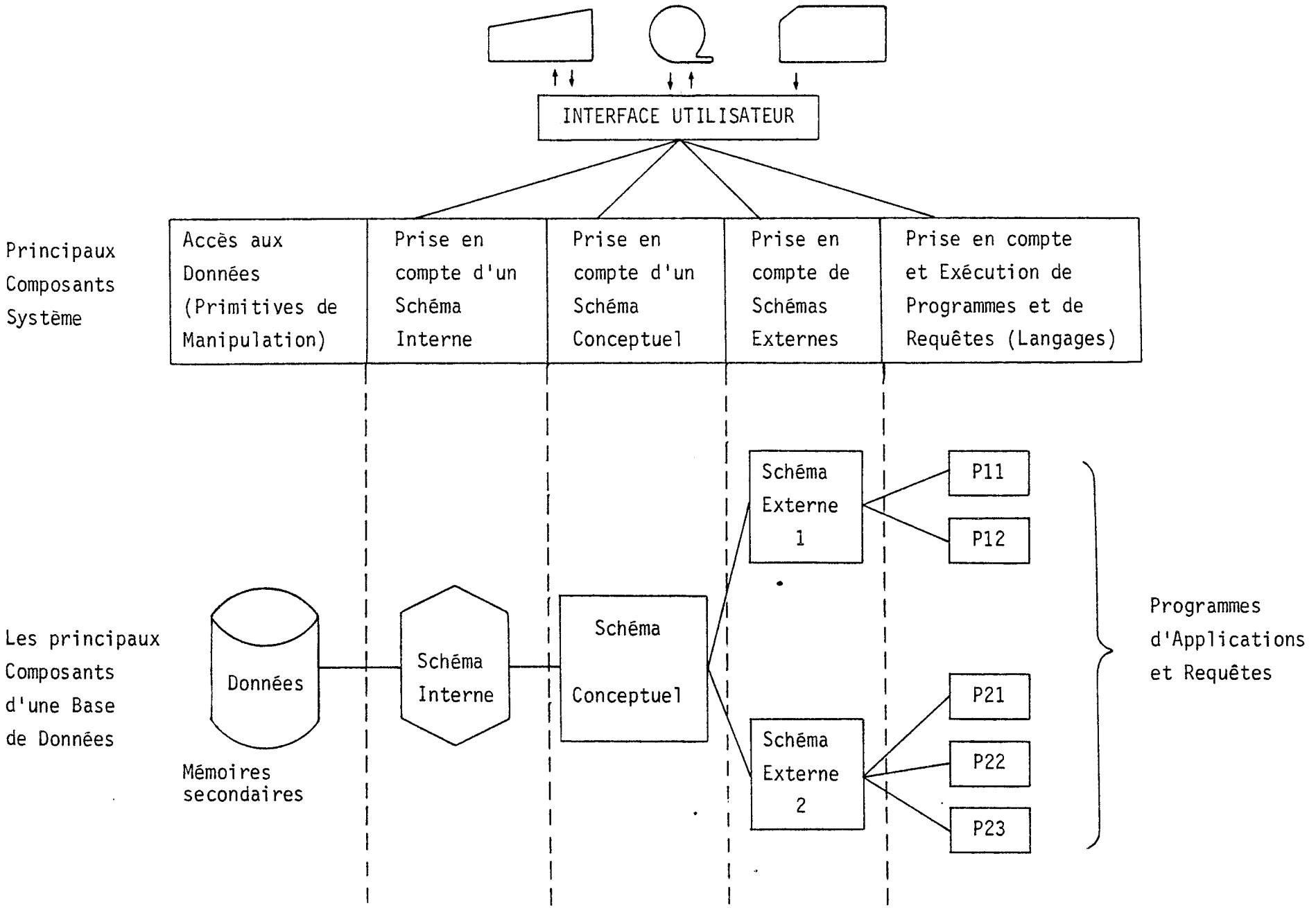


Figure 1.4 - L'architecture générale d'un SGBD

- le système R construit au Laboratoire de recherche IBM de San José (USA) [B9, B8].

1.1.3.2.1. Les prototypes relationnels

Les principales caractéristiques de ces systèmes sont les suivantes :

1) *Une architecture à plusieurs niveaux fonctionnels* définissant des sous-systèmes disponibles via des interfaces programmables. C'est le cas du système R avec :

- un *sous-système de stockage relationnel* gérant des unités de mémoires secondaires, l'allocation des zones, les actions concurrentes directes (blocage, déblocage), les mécanismes de recouvrement après une panne, les index secondaires et les relations de la base. Ce sous-système est accessible via un interface autorisant l'accès à des relations et à des n-uplets que nous avons décrits dans [B3].
- un *sous-système de données relationnelles* permettant la description et la manipulation des relations stockées par le sous-système précédent. A ce niveau, on dispose d'un langage relationnel complet, en l'occurrence SEQUEL [B23].

2) *Un langage relationnel complet* de haut niveau comme SEQUEL mais aussi QUEL pour INGRES. Ces langages sont utilisables directement ou grâce à un langage hôte (PL/1 pour SEQUEL). Ils regroupent de manière homogène les langages de définition et de manipulation des SGBD classiques (cf. § 1.1.2) permettant ainsi :

- la création, la destruction de relations, mais aussi leur modification structurelle : ajout d'un attribut par exemple. Ceci est un gros apport du modèle relationnel : la base peut être restructurée dynamiquement.
- l'insertion et la suppression de n-uplets.
- l'expression de contraintes d'intégrité que doit vérifier la base, par exemple :
  - . le salaire d'un employé ne peut jamais diminuer
  - . la moyenne des salaires d'un département ne peut dépasser une certaine somme.

- l'expression d'actions spontanées qui seront déclenchées automatiquement par le système si tel ou tel événement se produit : par exemple, spécifier que si l'on insère dans la base l'employé "DUPONT" travaillant au département "50", il faut automatiquement augmenter de 1 le nombre d'employés du département "50".
- l'expression de requêtes d'interrogation quelconques exprimées de manière non procédurale. Généralement le système décompose ces requêtes en opérations de l'algèbre relationnelle mais en tenant compte des chemins d'accès existant dans la base ce qui permet une optimisation de leur exécution [B9, B64]. Ces chemins d'accès sont matérialisés par toute une série d'index secondaires dont le contenu est géré automatiquement par le système, au fur et à mesure des opérations de mises à jour que subissent les relations à partir desquelles ces index sont construits.

3) *Des mécanismes complets pour contrôler le partage des données* et en particulier les mises à jour concurrentes. La notion de transaction qui correspond à un ensemble d'opérations cohérentes est alors définie.

On trouve également des mécanismes très sophistiqués concernant les droits qu'ont les utilisateurs pour accéder et modifier telles ou telles portions de la base. Celui qui crée dans la base une nouvelle relation en est le propriétaire, mais possède également le droit d'autoriser ou non d'autres utilisateurs à réaliser sur cette relation des opérations d'accès et de mise à jour. Ces utilisateurs peuvent également être autorisés à autoriser d'autres personnes à réaliser ces opérations, et ainsi de suite [B35].

4) La possibilité de définir des *vues* d'une base de données : ces vues peuvent être relationnelles et elles correspondent alors à un ensemble de relations qui peuvent s'obtenir par des opérations appliquées aux relations de base, *les seules à être effectivement stockées.* Cependant, ces vues peuvent aussi être hiérarchiques ou réseaux et il appartient au système de transformer les opérations s'adressant à ces vues en opérations réelles appliquées sur les informations stockées.

1.1.3.2.2. Les futurs SGBD

Comme on vient de le voir, les prototypes relationnels contiennent des caractéristiques intéressantes pour arriver à des SGBD plus simples à utiliser et où la cohérence des informations est renforcée.

Cependant, les autres modèles ne sont pas oubliés pour autant, étant donné le nombre de bases de données actuellement réalisées.

Le groupe ANSI/SPARC a proposé en 1975 des normes pour les futurs SGBD. Ces normes suscitent actuellement de nombreuses discussions et sont à l'origine de nombreux travaux [B11, B47].

Enfin, on peut signaler les travaux entrepris dans le domaine de la reconnaissance des langues naturelles pour l'interrogation des bases de données [B21] ainsi que ceux concernant les systèmes déductifs qui dépassent le cadre de notre étude.

1.2. TRAITEMENT DISTRIBUE ET SYSTEMES REPARTIS

1.2.1. Evolution des systèmes de traitement

La centralisation qui a eu son apogée dans les années 1965-1975 est actuellement concurrencée par l'informatique répartie qui deviendra vraisemblablement l'informatique des années 80. Il semblait en effet que regrouper le maximum d'informations et de traitements dans un gros ordinateur central permettait de résoudre le maximum de problèmes. Or, la diminution des coûts et l'augmentation de puissance des mini-ordinateurs ainsi que le développement des réseaux de communication amènent à penser que cette centralisation n'est pas la meilleure voie à suivre [R66, R18].

Alors que beaucoup d'utilisateurs en sont encore au stade de l'informatique centralisée, l'expérience de certains organismes (banques, compagnies d'assurances, administration) ainsi que les annonces des constructions en matière de systèmes répartis [R67] font que la tendance actuelle est plutôt à la décentralisation.

La première approche avait pour but d'amener l'utilisateur auprès de la machine, tandis que la seconde revient plutôt à amener la puissance de l'ordinateur auprès de l'utilisateur. Ainsi l'informatique répartie est-elle

mieux adaptée aux structures organisationnelles (administration, entreprise).

La répartition des utilisateurs est apparue dès la sortie des systèmes autorisant le télétraitement et le temps partagé : plusieurs terminaux répartis géographiquement permettent l'accès à distance aux services d'un ordinateur central assurant tous les traitements. C'est le fait de transformer ces terminaux en mini-ordinateurs (on dit aussi en terminaux "intelligents") qui permet la répartition des traitements et des données. Ainsi, un système réparti se caractérise-t-il par les deux critères suivants (voir Figure 1.5) :

-1- Il possède deux ou plusieurs ordinateurs répartis géographiquement sur plusieurs sites. Chaque machine locale gère un ensemble d'informations et exécute des programmes d'application, ou des parties de programmes lorsqu'il s'agit d'une application répartie, c'est-à-dire d'un traitement qui fait intervenir plusieurs sites.

-2- Les différents ordinateurs sont reliés entre eux au moyen de mécanismes de communication. La plupart du temps, il s'agit de télécommunications du fait de l'éloignement géographique des sites concernés. Pour correspondre entre eux, ces différents ordinateurs vont utiliser un ensemble de règles destinées à assurer les transferts d'informations diverses : données, messages, programmes. Ces règles constituent un *protocole* de communication [R49].

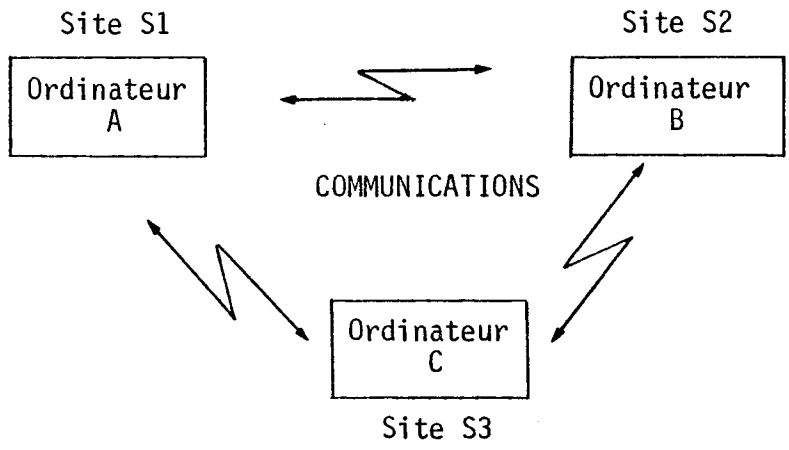


Figure 1.5 - Les principaux éléments d'un système réparti

Lorsqu'un système réparti se trouve dans les limites d'une entreprise ou d'une administration, il est constitué la plupart du temps d'ordinateurs homogènes (ou en tout cas compatibles) qui correspondent entre eux grâce à un réseau privé de communication. A l'inverse de nombreux travaux ont été consacrés à la réalisation et à la mise en place de réseaux généraux de communications permettant la connection d'ordinateurs hétérogènes (ARPA aux Etats-Unis, CYCLADES [R49] en France, TRANSPAC, EURONET, ...).

L'existence de tels réseaux qui fournissent un ensemble de services permet d'envisager la construction d'applications réparties non seulement dans les limites d'une administration ou d'une entreprise, mais encore entre des organismes divers désireux de partager des informations [R21].

Les bases de données réparties constituent un cas particulier de systèmes répartis et nous les étudierons au paragraphe 1.3. Avant cela, nous allons essayer de caractériser différents types de systèmes répartis :

1.2.2. Les différents types de répartition des systèmes

Par opposition à une structure centralisée, les systèmes répartis se divisent en deux grandes catégories :

- les systèmes à répartition hiérarchique ou verticale
- les systèmes à répartition horizontale ou en réseaux.

1.2.2.1. Systèmes répartis hiérarchiques

Les différentes machines composant le système se partagent les tâches de manière hiérarchique : au plus bas niveau des terminaux chargés d'assurer l'entrée des données, l'affichage ou l'impression. Ces terminaux sont gérés par des ordinateurs de petite taille assurant quelques traitements élémentaires et éventuellement le stockage d'informations peu volumineuses. Ces ordinateurs correspondent alors avec un ordinateur central chargé d'effectuer des tâches plus importantes (gros calculs, stockage des grosses masses d'information).

Cette structure, illustrée par la Figure 1.6, se rencontre fréquemment dans de grosses entreprises. Chaque noeud de la hiérarchie ne communique qu'avec celui qui se trouve au-dessus de lui. Deux noeuds de même niveau ne peuvent communiquer directement.

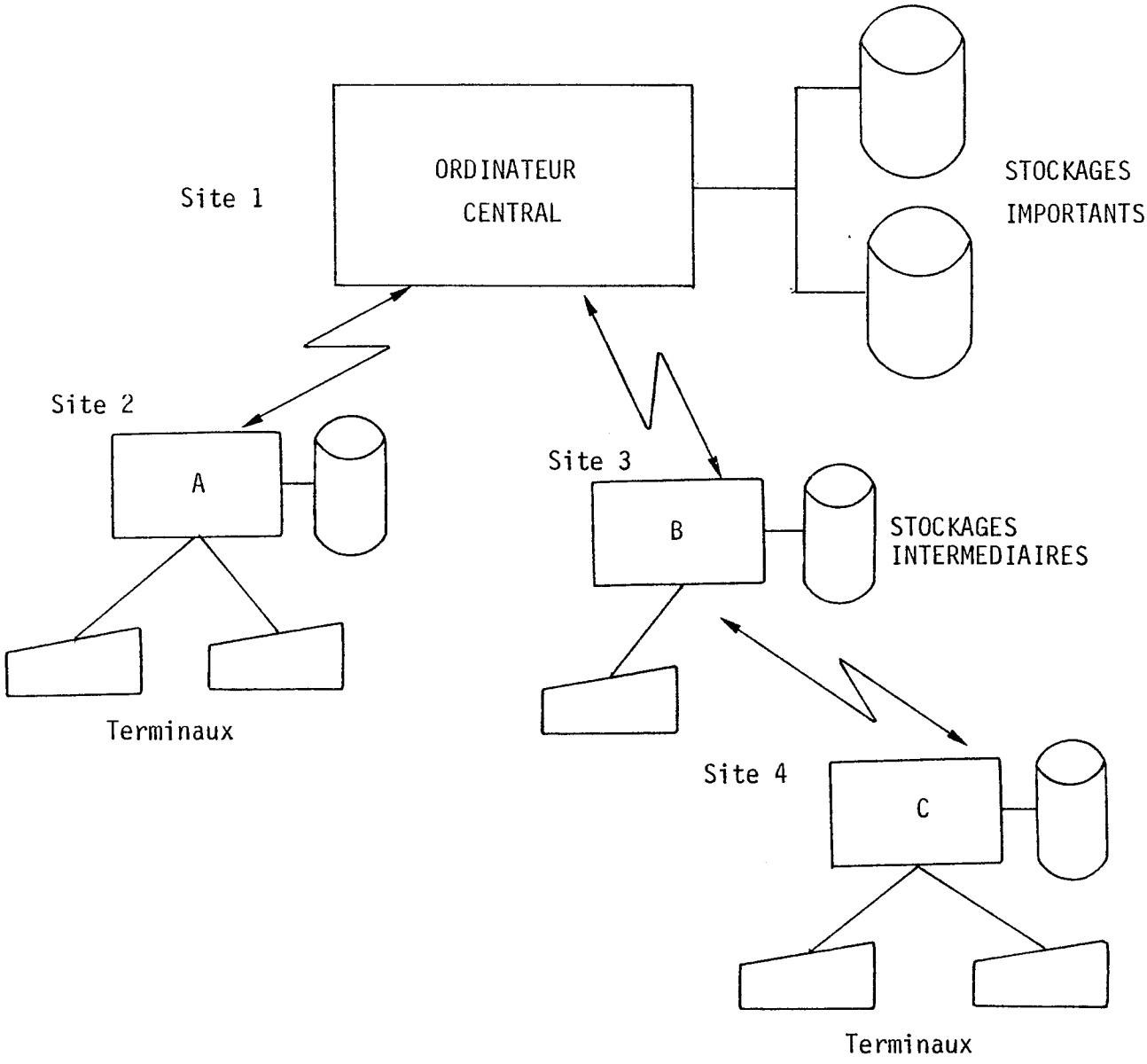


Figure 1.6 - Système Réparti hiérarchiquement

1.2.2.2. Systèmes répartis horizontalement

Les différentes machines qui composent un tel système coopèrent toutes au même niveau. Il en découle une plus grande autonomie de chaque machine, ce qui permet à ce type de systèmes d'évoluer par l'adjonction de nouveaux composants (Figure 1.7).

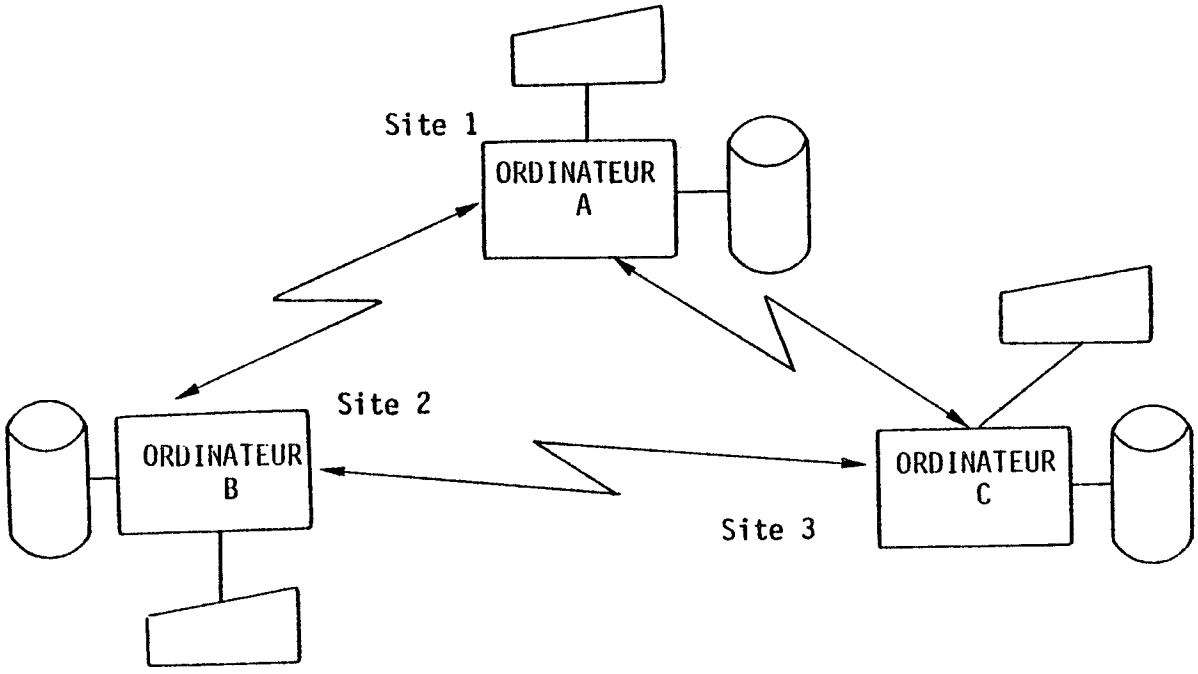


Figure 1.7 - Système Réparti horizontalement

Dans le réseau d'ordinateurs ainsi constitué, chaque noeud peut correspondre avec n'importe quel autre. Ce type de systèmes favorise l'hétérogénéité des machines composantes, et des réseaux comme CYCLADES paraissent tout à fait adaptés à leur mise en place.

En résumé, nous pouvons dresser le tableau suivant permettant de comparer l'approche centralisée avec les différents types de répartition :

TYPES DE SYSTEMES	CENTRALISATION	REPARTITION HIERARCHIQUE	REPARTITION HORIZONTALE
Principales Caractéristiques	- Tout le travail est fait dans un site central. - Les utilisateurs distants sont reliés par lignes téléphoniques	- Le travail est réparti de manière hiérarchique : il est contrôlé par le niveau supérieur.	- Tous les ordinateurs coopèrent au même niveau : il n'y a pas de contrôle central.
Avantages	- Economie d'échelle. - Maintenance du système plus facile.	- Système plus sûr. - Coût de communication réduit. - Permet le passage en souplesse d'un système centralisé à un système réparti.	- Chaque ordinateur a plus d'autonomie et peut avoir divers types d'activité. - Possibilité d'emploi de matériel divers. - Adaptation aux changements technologiques ainsi qu'à l'évolution des besoins plus aisés.
Inconvénients	- Vulnérabilité aux pannes. - Impossibilité de répondre facilement aux changements de technologie ou aux nouveaux besoins des usagers.	- Difficultés d'utiliser du matériel de constructeurs différents.	- Rares seront les constructeurs qui pourront fournir le système entier. - Difficultés de conversion à partir d'un système existant.

1.3. LES BASES DE DONNEES REPARTIES

L'approche "bases de données réparties" se situe au point de convergence des notions de systèmes réparties vues au § 1.2 et de l'approche bases de données évoquée au § 1.1.

Cependant, ce terme générique de bases de données réparties (ou BDR) recouvre un domaine informatique nouveau, en pleine expansion et dans lequel les concepts fondamentaux ne sont encore ni clarifiés, ni stabilisés.

Les travaux de recherche et de développement dans ce domaine sont d'une très grande diversité. Les auteurs de ces travaux présentent et utilisent des concepts a priori différents et nous allons essayer dans ce qui suit d'en faire l'analyse afin de dégager une sorte d'état de l'art dans ce domaine.

On trouvera dans les documents [R8, R10, R28, R51, R55, R59] d'autres synthèses sur les bases de données réparties.

1.3.1. Concepts fondamentaux et recherches en cours

1.3.1.1. Base répartie et Système de Base de Données Réparties

Une base de données est répartie lorsque :

1) Les informations qui la constituent se trouvent physiquement stockées dans des machines qui composent un système réparti (§ 1.2).
2) Les traitements initialisés sur un site font en général intervenir des informations stockées sur d'autres sites (Figure 1.8).

Un des principaux avantages d'une telle approche est que le système gérant la base répartie agit fonctionnellement comme un système centralisé, tandis que physiquement il s'adapte à la répartition des composants d'une entreprise ou d'une administration.

Considérons trois sites S1, S2, S3 (Figure 1.8) et une base de données B constituée de trois parties P1, P2, P3 représentant chacune un ensemble d'informations et telles que symboliquement on puisse écrire (cf. § 1.3.2) :

$$B = P1 \cup P2 \cup P3.$$

A partir d'un site quelconque Si, il est possible d'accéder à la totalité de la base B, c'est-à-dire non seulement à la portion Pi du site où se situe l'accès, mais également aux autres portions de la base. Ainsi, l'utilisateur a-t-il l'impression de s'adresser à un système centralisé tandis qu'en fait, on va chercher l'information là où elle se trouve.

Dans la réalité d'une exploitation, la portion Pi stockée sur le site Si correspond à des informations collectées et très souvent utilisées par ce site.

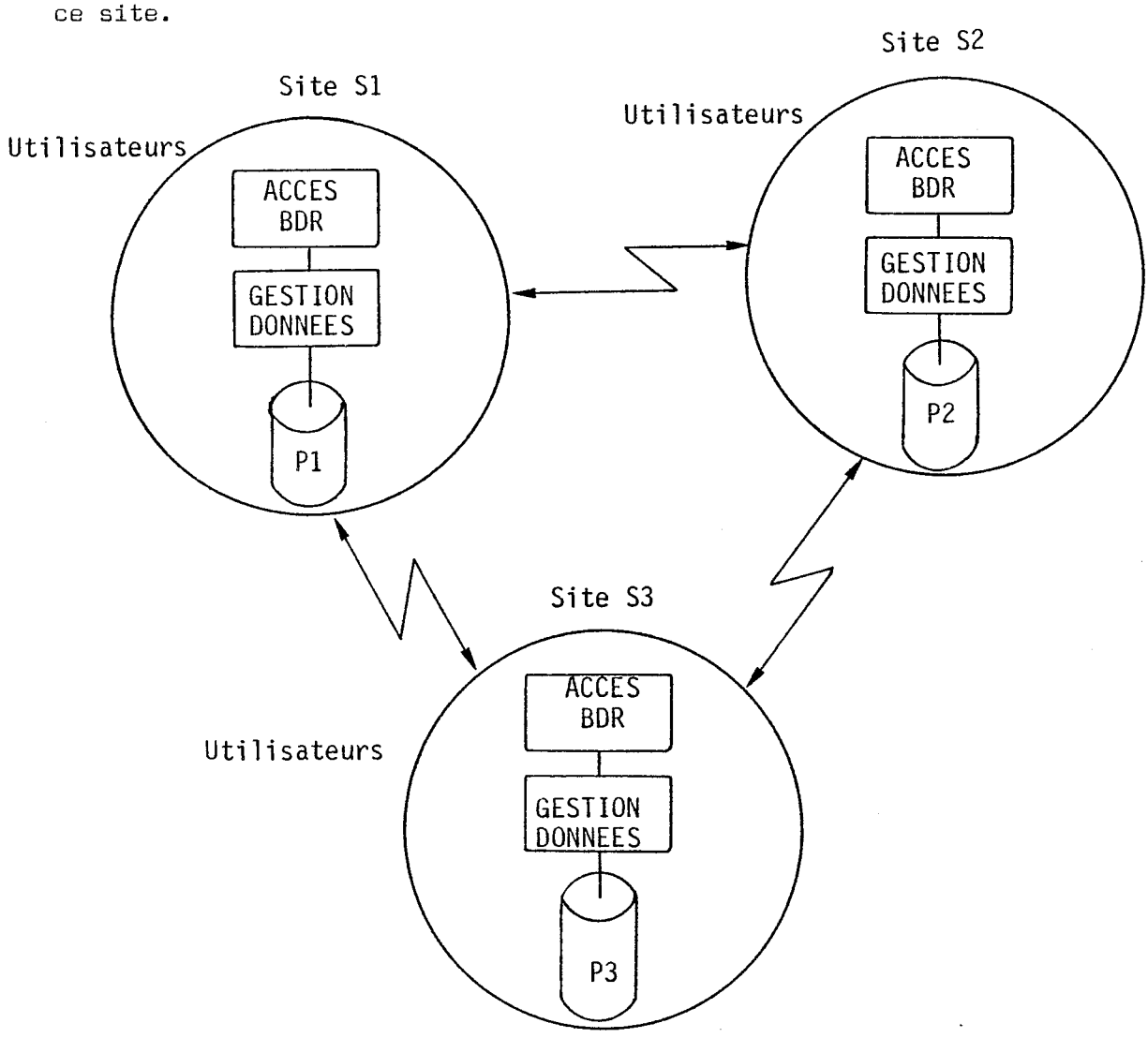


Figure 1.8 - Une base de données répartie sur S1, S2 et S3

Sur chaque site, le système réparti assure deux fonctions principales, à savoir :

1) Gérer un ensemble d'informations locales (ou base locale).

2) Permettre à un ensemble d'utilisateurs l'accès à cette base locale mais également à toutes les informations stockées sur les autres sites (la base répartie).

Le système informatique ainsi défini constitue un *Système de Gestion de Bases de Données Réparties* ou *SGBDR*. Par rapport aux systèmes de bases de données centralisées, il offre trois avantages principaux :

1. Une plus grande sûreté de fonctionnement car la panne d'un des composants n'arrête pas l'ensemble du système.

2. Les informations peuvent être stockées sur le site où se fait leur saisie et là où se situent la majorité des applications les concernant. On supprime ainsi les coûts de communication avec un site centralisateur tout en adaptant l'architecture du système informatique à la structure organisationnelle de l'entreprise ou de l'administration.

3. L'évolution de la base de données répartie se trouve facilitée : l'intégration d'une nouvelle base au système permet la construction et l'exploitation de très grosses bases de données (10^{18} bits) n'ayant pas une structure monolithique.

Il y a souvent confusion entre SGBDR et répartition de certains composants logiciels d'un SGBD classique qui ne font pas intervenir les données : par exemple, la gestion des terminaux utilisateurs (Figure 1.9) :

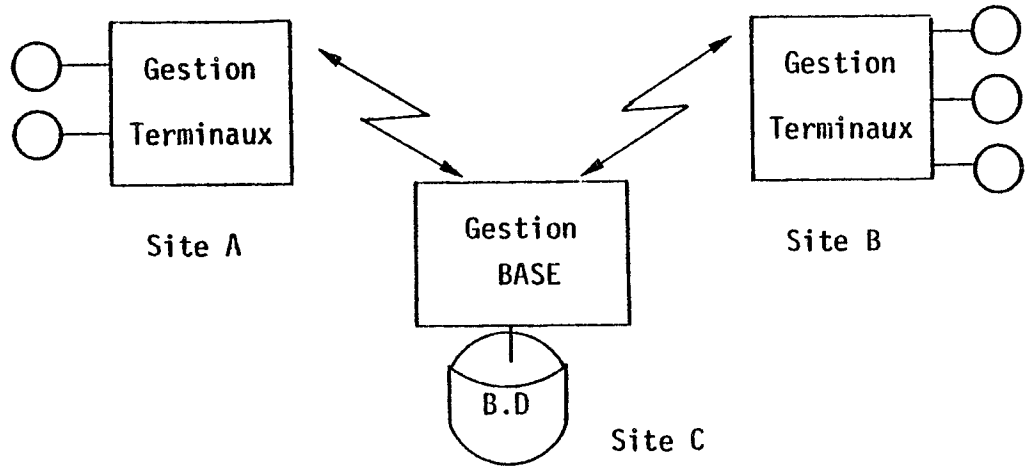


Figure 1.9

Il n'existe pas, à notre connaissance, de SGBDR généraux commercialisés comme il existe des SGBD. Plusieurs travaux de recherche et de développement sont en cours dans ce domaine.

1.3.1.2. Principaux travaux sur les bases de données réparties

Peu de travaux concernent la conception de SGBDR généraux. La plupart du temps, il s'agit de travaux ponctuels concernant un problème spécifique aux bases réparties : mises à jour concurrentes, allocation optimale de fichiers, gestion des pannes, outils de programmation d'applications réparties, ...

De nombreux travaux concernent des architectures de SGBDR construits autour de mini-ordinateurs et ceci aussi bien aux Etats-Unis [R53, R10, R55] qu'en Europe [R45, R46, R52, R22, R70].

Un projet américain important est mené actuellement par la "Computer Corporation of America" pour concevoir et réaliser un SGBDR complet : SDD/1 (System for Distributed Data) [R56, R57]. Ce projet qui utilise comme modèle commun de données le modèle relationnel est à rapprocher des travaux décrits dans [R44] qui concernent une version répartie du système relationnel INGRES, évoqué au § 1.1.3.2.

En France, l'IRIA a lancé, courant 1976, un projet pilote sur les bases de données réparties : le projet SIRIUS [R42].

Ce projet permet de fédérer les différents travaux effectués dans ce domaine, soit chez des utilisateurs, soit dans des universités :

- à *Rennes*, une réalisation concernant un système de fichiers répartis a été faite sur le réseau CYCLADES [R15]. Les travaux se poursuivent dans la direction des bases de données réparties.

- à *Toulouse*, en particulier au Centre d'Etudes et de Recherche, après l'expérience acquise par la conception et la réalisation d'un SGBD relationnel : SYNTEX [B28], les travaux se poursuivent vers des modèles de données réparties [R26, R27].

- à *Paris*, l'Institut de Programmation en collaboration avec la Régie Renault travaille sur la définition et la réalisation d'un SGBDR général [R33, R34, R35].

- à *Grenoble*, le projet POLYPHEME auquel notre travail est consacré a pour but la conception et la réalisation d'un système de coopération de bases de données hétérogènes réparties. Héritier, en quelque sorte, des travaux effectués d'une part par l'équipe réseaux d'ordinateurs qui a participé à CYCLADES [R21, R62] et d'autre part par l'équipe bases de données, POLYPHEME se place dans les hypothèses suivantes :
- existence de diverses bases de données hétérogènes en exploitation
- existence d'un réseau de communication connectant les différentes machines sur lesquelles sont implantées ces bases
- nécessité de mettre en place une application base de données réparties nécessitant une coopération de ces bases.

Nous ferons une large part dans les chapitres suivants au projet POLYPHEME, mais dans ce qui suit, nous allons caractériser d'abord les principaux problèmes concernant les BDR et les SGBDR.

1.3.2. Répartition des informations

La manière dont les informations sont réparties sur les différents sites repose sur deux notions principales :

- *partitionnement* ou découpage en sous-ensembles disjoints
- *duplication* ou création de plusieurs copies d'une même information.

Nous étudierons en détail ce problème au Chapitre 4, mais nous allons ici essayer de préciser ces notions en les appliquant aux objets d'une base de données répartie et à la description de sa structure (son schéma cf. § 1.1.2).

- Si une BDR B est *partitionnée*, cela signifie que les parties qui la composent forment des sous-ensembles disjoints (cf. § 1.3.1) :

$B = P1 \cup P2 \cup P3$ avec $P1 \cap P2 \cap P3 = \emptyset$.

- Si une BDR B est *totalement dupliquée*, cela signifie que les différentes bases locales sont des copies les unes des autres :

$B = P1 = P2 = P3$.

- Si une BDR est *partiellement dupliquée*, cela signifie qu'une même information peut se retrouver dans plus d'une base locale :

$\exists\ i,j$ tels que : $Pi \cap Pj \neq \emptyset$.

En considérant un exemple très simple, nous allons voir comment ces notions s'appliquent à un schéma relationnel de BDR.

Considérons une BDR B dont le schéma relationnel est le suivant :

R (NUS, NP, VILLE, PRIX, QTE)

et une réalisation de ce schéma que nous noterons par abus de langage de la même façon.

Cette réalisation est représentée conventionnellement par une table dont les lignes sont des n-uplets (Figure 1.10).

Un n-uplet $<u_i,p_i,v_i,f_i,q_i> \in R$ si et seulement si :
"l'usine de numéro u_i ($u_i \in$ NUS) localisée dans la ville v_i ($v_i \in$ VILLE) fabrique le produit p_i ($p_i \in$ NP) coûtant f_i francs ($f_i \in$ PRIX) en quantité q_i ($q_i \in$ QTE)".

1	u1	p1	PARIS	25	3
2	u1	p2	PARIS	15	6
3	u1	p3	PARIS	52	2
4	u2	p3	LYON	52	14
5	u2	p4	LYON	12	25
6	u3	p2	PARIS	15	17

Figure 1.10 - Une réalisation de R comportant 6 n-uplets

En supposant que B est à répartir sur trois sites S1, S2 et S3, examinons les différents cas de répartition :

Cas 1 : Non partitionnement de schéma :
Cela signifie que les portions P1, P2 et P3 ont toutes la même structure, à savoir :

R (NUS, NP, VILLE, PRIX, QTE).

On peut dire alors que le schéma est dupliqué.

Cas 1.1 : Partitionnement des objets :

Dans ce cas, il s'agit d'un découpage *horizontal* de la table R, par exemple :

u1	p1	PARIS	25	3
u1	p2	PARIS	15	6
u1	p3	PARIS	52	2

Site S1

u2	p3	LYON	52	14
u2	p4	LYON	12	25

Site S2

u3	p2	PARIS	15	17

Site S3

Cas 1.2 : Duplication des objets :

Dans ce cas, la notion d'objet correspond à celle de n-uplet et on peut avoir soit une duplication totale, soit une duplication partielle :

- duplication totale : table R sur S1, S2 et S3
- duplication partielle, par exemple :

u1	p1	PARIS	25	3
u1	p2	PARIS	15	6
u1	p3	PARIS	52	17
u3	p2	PARIS	15	17

S1

u2	p3	LYON	52	14
u2	p4	LYON	12	25

S2

u1	p1	PARIS	25	3
u1	p2	PARIS	15	6
		⋮		

S3
(Table R en entier)

Cas 2 : Partitionnement de Schéma :

Considérons le découpage de R en trois sous-relations R1, R2, R3 telles que R = R1 * R2 * R3 ("*" dénote la composition des relations, cf. § 2.5.3.2).

R1(NUS, VILLE) R2(NP, PRIX) R3(NUS, NP, QTE).

Cas 2.1 : Partitionnement sans duplication :

Considérons la répartition suivante :

Sites	Relations
S1	R1
S2	R2
S3	R3

Il y a duplication de R1 et R2 respectivement sur S1 et S3 et sur S2 et S3.
Il y a une incidence évidente de la répartition (duplication, partitionnement sur le traitement des requêtes qui s'adressent à la base répartie.

Nous reviendrons en détail sur ce problème au § 1.3.5.3, ainsi qu'au Chapitre 4. Dans la plupart des projets de SGBDR généraux, la redondance d'informations est systématiquement introduite (et donc contrôlée) de manière à améliorer les performances.

N.B. Nous ne considérons pas qu'un découpage vertical de la table R (par exemple les 2 premières colonnes sur S1, les trois suivantes sur S2) soit un cas de répartition de bases de données. C'est plutôt un cas de *répartition de fichiers*. En effet, si l'on a :

1	u1	p1
2	u1	p2
3	u1	p3
4	u2	p3
5	u2	p4
6	u3	p2

Site S1

1	PARIS	25	3
2	PARIS	15	6
3	PARIS	52	2
4	LYON	52	14
5	LYON	12	25
6	PARIS	15	17

Site S2

Il faut nécessairement qu'il existe des moyens techniques quelconques pour reconstituer les n-uplets : "pointeurs" inter-sites ou association par position, ... Il semble que cela soit envisagé dans le système SDD/1 [R57] avec la notion d'identificateur de n-uplet (tid : tuple identifier). Notons cependant que cela revient à disposer de la clé de la relation sur tous les sites où elle est répartie.

1.3.3. La typologie en matière de Systèmes de Gestion de Bases Réparties (SGBDR)

Deux critères permettent de caractériser les différents types de SGBDR :

- la nature horizontale ou hiérarchique du système réparti
- la nature homogène ou hétérogène des composants du SGBDR.

1.3.3.1. SGBDR Hiérarchiques ou SGBDR Horizontaux

Dans un SGBDR à répartition horizontale, les bases locales peuvent communiquer entre elles car elles se situent toutes au même niveau. On aura alors plutôt tendance à parler de coopération entre bases locales. Avec une telle architecture, la base répartie peut être partitionnée, dupliquée partiellement ou totalement.

Dans une répartition hiérarchique (cf. § 1.2.2.1), l'ordinateur central gère la plus grosse portion de la base répartie, tandis que les machines de niveau inférieur gèrent chacune des portions (en général disjointes) plus réduites. Les données locales sont saisies et mises à jour sans faire appel au site central. Cependant, à intervalles réguliers (chaque nuit, chaque semaine), la base centrale est mise à jour à partir des informations locales.

Avec les notations du § 1.3.1 et si S1 est le site central, on a :

$$P2 \subset P1 \qquad P3 \subset P1 \qquad P3 \cap P2 = \emptyset.$$

Ou bien, si P2 et P3 contiennent des informations qui leur sont propres :

$$P2 \cap P3 = \emptyset \qquad P1 \cap P2 \neq \emptyset \qquad P1 \cap P3 \neq \emptyset.$$

Ce type de SGBDR Hiérarchiques se rencontre fréquemment dans des applications où il existe un ou plusieurs sites centralisateurs du traitement informatique : Banques [R66], entreprises avec nombreuses succursales, grosses administrations (le CNRS par exemple [R47]).

1.3.3.2. SGBDR Standardisé (homogène) ou Intégré (hétérogène)

Dans un SGBDR standardisé, les composants logiciels implantés sur chaque site sont homogènes : par exemple, le même SGBDR, offrant ainsi un modèle unique de données et un même langage de description et de manipulation.

Cette approche s'accorde mieux avec une répartition hiérarchique dans la mesure où cette structure suppose un certain degré d'homogénéité dans le matériel et le logiciel [R22] (cf. § 1.2).

Dans un SGBDR intégré, les composants sont hétérogènes, ce qui entraîne la présence de différents modèles de données et de différents langages. La plupart du temps, cette approche coïncide avec une répartition horizontale. Elle tient compte du fait que dans la réalité, il existe de nombreux SGBD commercialisés et qu'au sein d'une même entreprise ou d'une même administration, l'informatisation a été faite par secteurs, créant ainsi

des bases de données, implantées sous des systèmes hétérogènes, dans différents centres de calcul. La présence de réseaux généraux d'ordinateurs, dans la mesure où ils assurent la connexion de matériels hétérogènes, fait qu'ils rendent possibles la conception et la réalisation de SGBDR intégrés. C'est ce problème qui est abordé dans le projet POLYPHEME (§ 1.3.1.2) auquel notre travail est consacré.

1.3.4. Problèmes posés par la conception d'une Base Répartie

Il importe avant tout de ne pas confondre la conception d'une BDR avec la conception d'un SGBDR qui constitue l'outil permettant de créer et d'exploiter une base répartie.

Dans la mesure où il n'existe pas encore de véritables SGBDR commercialisés comme il existe des SGBD, la conception de la BDR va souvent de pair avec celle de l'outil correspondant. Il en résulte des applications spécifiques enfermées dans le cadre d'une entreprise, d'une administration ou d'une organisation quelconque [R10, R66].

Selon les hypothèses de départ prises pour la conception d'une BDR, les approches sont très différentes.

1.3.4.1. Création d'une Base Répartie (Approche descendante)

Si l'on suppose qu'une BDR est créée de toutes pièces, sa conception et sa mise en place sont alors très voisines de celles d'une base classique (cf. § 1.1) avec bien entendu les problèmes liés à la répartition des données (cf. § 1.3.2).

Cependant, on est maître dans ce cas du découpage de la base et de la redondance que l'on y introduit (cf. Figure 1.11).

Il faut alors tenir compte, pour effectuer ce découpage, du type des transactions et du volume des informations qu'elles font intervenir. Il importe alors de trouver une répartition optimale qui réduise les coûts de traitement et de transfert [R55].

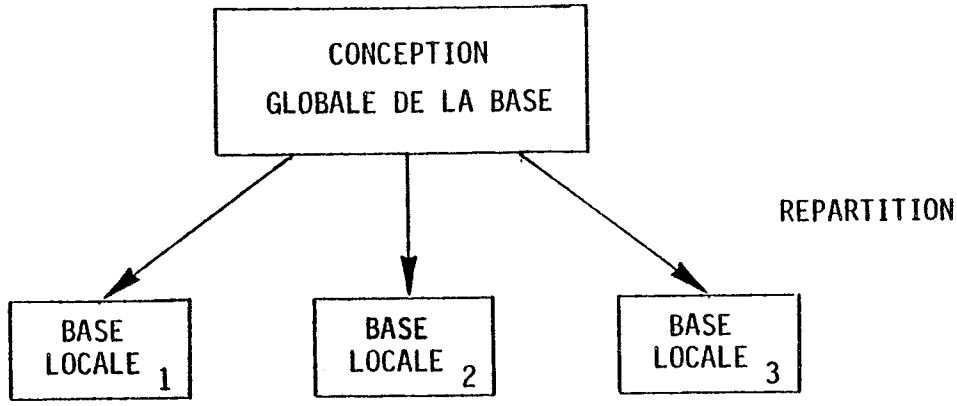


Figure 1.11 - L'approche descendante pour la conception d'une Base de Données Répartie

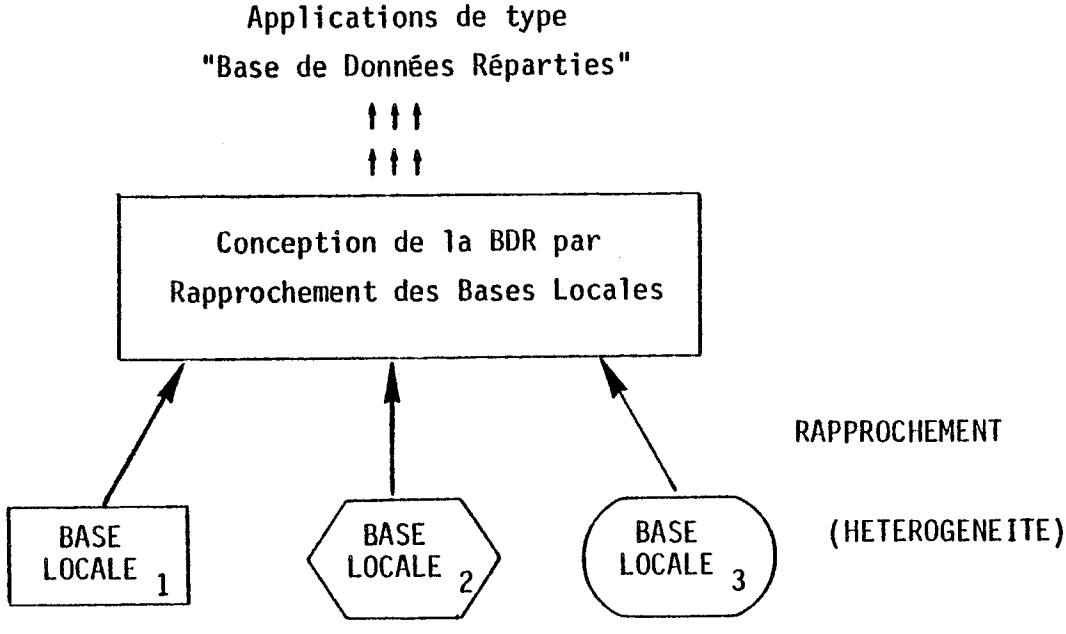


Figure 1.12 - L'approche ascendante

Ceci rejoint un problème général en matière de BDR, à savoir dans quelle mesure la répartition des informations doit-elle être visible au niveau de la conception et de l'utilisation de la BDR ? ou en d'autres termes, un utilisateur d'une BDR doit-il être concerné par la localisation des données qu'il manipule ?

Il n'existe pas à notre avis de réponse unique à ce genre de questions. Tout dépend de la nature de l'application envisagée et du niveau où se situe l'utilisateur. Il est sûr que pour certains types d'utilisateurs de BDR, la répartition des informations doit être totalement transparente. Cependant, dans d'autres cas, la localisation joue un rôle prépondérant dans l'application répartie.

Ce que l'on peut souligner à ce niveau, c'est que de toutes manières les objets constituant une BDR sont localisés et qu'il appartient aux couches logicielles qui permettent leur gestion, de cacher ou au contraire de faire ressortir cette localisation.

1.3.4.2. Conversion d'une base centralisée en une BDR

Il existe actuellement de nombreuses bases centralisées. Dans la mesure où la tendance décentralisatrice de l'informatique ira en s'accentuant, de nombreux organismes auront à faire face au problème de la transformation d'une base centralisée en une base répartie [R48].

La plupart du temps, l'évolution se fera des SGBD classiques vers des SGBDR hiérarchiques et homogènes dans lesquels les portions de la base centrale seront stockées et gérées de manière locale (cf. § 1.3.3.1). Là encore, il existe une certaine liberté dans le choix du découpage de la base et dans la localisation des différentes parties (partitionnement de la structure et des objets, contrôle de la redondance, contrôle de l'incohérence entre diverses copies par mise à jour à intervalles réguliers).

1.3.4.3. Transformation de bases existantes

Dans ce cas, on suppose a priori l'existence de plusieurs bases de données implantées et exploitées sous des SGBD classiques. Si l'on désire réutiliser ces bases pour mettre en place une BDR, deux approches sont possibles :

1°) Intégration des bases après modification :

Il s'agit d'abord de concevoir la BDR comme au § 1.3.4.1 (approche descendante) et au niveau de son implantation, de remodeler les bases existantes pour les adapter à la nouvelle application. Ce remodelage pouvant aller d'une restructuration (modification de la structure, déchargement et rechargement de la base) à une complète traduction si les bases existantes doivent être gérées par des SGBDs différents de ceux sous lesquels elles étaient implantées [B55]. Dans les deux cas, il semble difficile de conserver les programmes d'applications qui exploitaient les bases existantes, ce qui suppose une refonte totale des applications.

2°) Intégration des bases sans modification : coopération :

Il s'agit alors de réutiliser les bases sans modifier leur structure de manière à ne pas perturber les applications déjà existantes et ceci tout en mettant en place une structure de BDR pour permettre la prise en compte de nouvelles applications. On est alors en présence d'une démarche ascendante dans laquelle on construit la BDR en rapprochant des composants existants et possédant naturellement des liens logiques entre eux.

Un des plus gros problèmes à résoudre dans ce cas est celui de l'hétérogénéité (cf. § 1.3.3.2) : les SGBD utilisent des modèles et des langages différents : on est alors amené à définir un niveau d'homogénéisation pour la description des bases locales aussi bien en ce qui concerne les informations stockées que les fonctions qu'elles peuvent réaliser grâce à leur SGBD. C'est ce type d'approche qui est à la base du projet POLYPHEME et qui conduit à des SGBDR horizontaux intégrés que l'on désignera le plus souvent par systèmes de coopération de bases de données réparties [R50].

Notons enfin que ce type d'approche a les deux avantages suivants :

(1) une telle BDR peut ne pas être limitée à une seule entreprise ou une seule organisation puisqu'il est possible d'envisager un système de coopération sur un réseau général d'ordinateurs tel que CYCLADES ou TRANSPAC

(2) chaque base locale conserve son autonomie et ses utilisateurs locaux. Cependant, ceci conduit à de nouveaux types de problèmes concernant l'intégrité des données réparties et leur confidentialité [R39, R43, R58].

1.3.5. Problèmes spécifiques aux Systèmes de Bases de Données Réparties

1.3.5.1. Description de la base de données répartie (BDR)

Comme toute base de données, une BDR doit être décrite au moyen d'une structure ou d'un schéma. Si nous reprenons les trois niveaux du § 1.1.2.1, nous avons alors (Figure 1.13) :

- au *niveau local*, les schémas qui décrivent les données locales
- au *niveau global*, un schéma décrivant la totalité de la BDR et les correspondances avec les schémas locaux : c'est dans cette correspondance que s'exprime la localisation des informations
- enfin, toujours au niveau global, des schémas externes qui correspondent aux différents utilisateurs de la BDR.

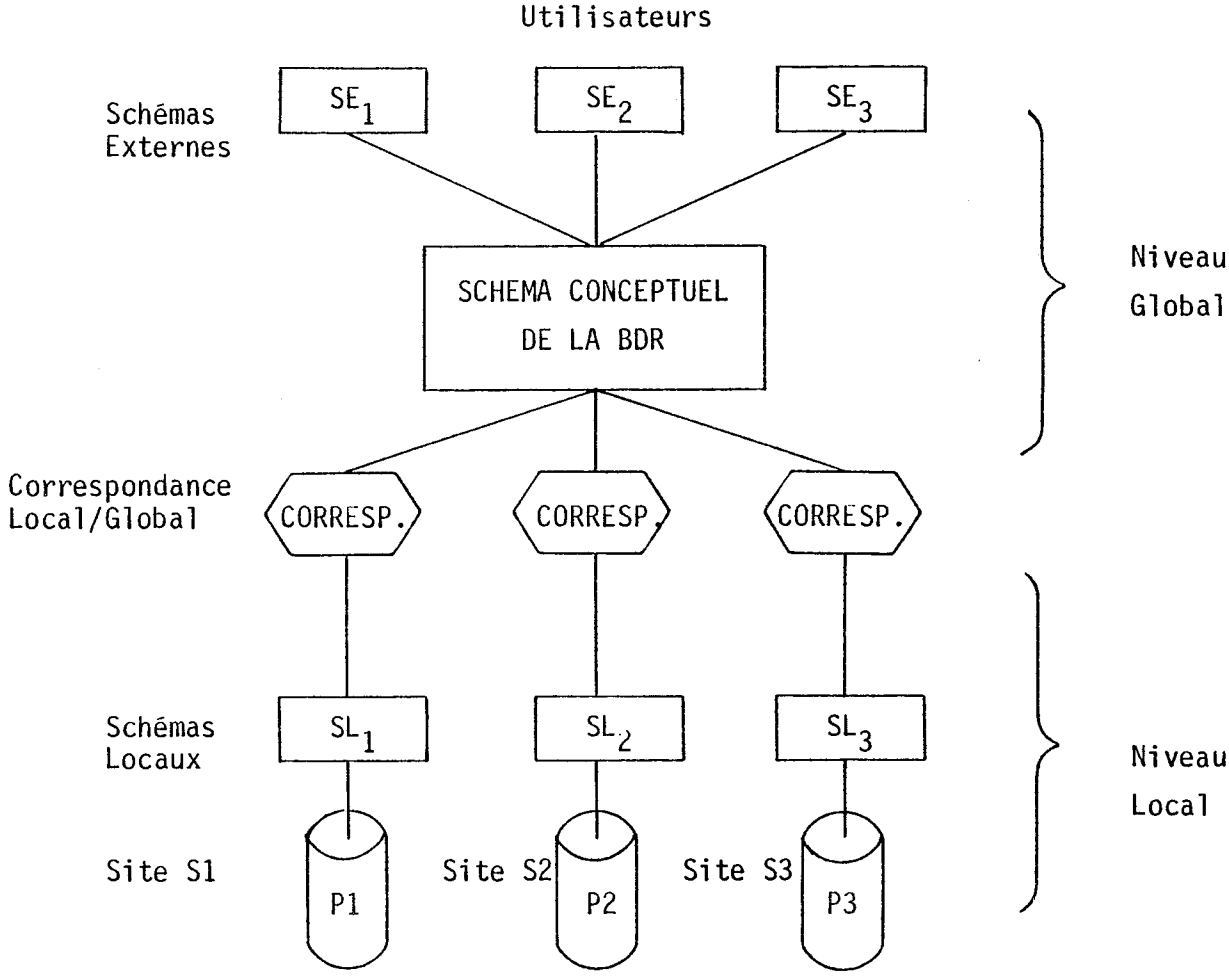


Figure 1.13 - Les niveaux de description d'une base répartie

Les problèmes qui se posent à ce niveau sont les suivants :

- avec quel modèle et quel langage décrire les différents schémas ?
- quelle est la nature des informations qu'ils contiennent ?
- tous ces schémas ont-ils une existence réelle ? En particulier, le schéma conceptuel existe-t-il en tant que schéma à part entière ou est-il en quelque sorte "dilué" dans les différents schémas externes ?
- comment exprimer la manière dont sont réparties les informations ?
- comment exprimer la manière dont peuvent être localisées les informations ainsi réparties ?

A priori le contenu des schémas externes d'une BDR correspond à celui d'un schéma classique si l'on considère qu'à ce niveau les usagers ne sont pas concernés par la répartition des informations.

De même, les schémas locaux décrivent, par des modèles classiques (hiérarchies, réseaux, relations) des bases de données considérées comme autonomes.

C'est par contre au niveau du schéma conceptuel de la BDR que se situent les problèmes de description de la base répartie avec prise en compte de contraintes d'intégrité propres à la répartition.

De plus, la correspondance avec les schémas locaux, qui sont en général exprimés avec des langages et des modèles hétérogènes, pose un problème de description et d'homogénéisation.

A notre connaissance, il n'existe pas encore de modèles et de langages complets, destinés aux bases réparties. La plupart du temps, il s'agit d'un modèle existant (CODASYL, par exemple) adapté au problème.

Une grande partie de notre travail a consisté à concevoir un *modèle général de données réparties* (MOGADOR) se situant au niveau conceptuel et permettant de décrire de manière relationnelle des informations réparties dans des bases locales hétérogènes. Ce modèle a servi également à définir l'architecture fonctionnelle d'un système de gestion de base de données réparties (cf. Chapitres suivants).

1.3.5.2. Répartition des schémas d'une BDR

Les informations contenues dans les schémas décrivant une BDR peuvent être elles-mêmes réparties (cf. § 1.3.2). Dans un article décrivant une version répartie du système INGRES développé à l'Université de Berkeley, Stonebraker et Neuhold [B44] comparent l'existence du schéma conceptuel de la BDR à celle de Dieu :

- Si un "Dieu" unique existe, il est stocké sur un site particulier et toutes les transactions concernant la base répartie passent par ce site.
- Si "Dieu" existe mais n'est pas unique, il en existe plusieurs "copies" sur des sites différents permettant l'accès à la BDR à partir de plusieurs sites.
- Si "Dieu" n'existe pas, cela signifie qu'il n'existe pas de description centralisée de la BDR mais plusieurs descriptions partielles et nécessairement réparties.

Nous avons là trois types de répartition possibles du schéma de la BDR. Elles ont été étudiées dans [R19] de manière à déterminer une approche optimale qui tienne compte des paramètres suivants :

- coût de communication inter-sites
- coût de stockage
- taille des schémas
- coût de conversation dans le cas de matériel hétérogène
- fréquence d'accès et de mise à jour des schémas.

On peut dresser le tableau récapitulatif suivant :

Types de Répartition	Schéma BDR centralisé ("Dieu" unique)	Schéma BDR réparti (copies)	Schémas locaux ("Dieu" n'existe pas)
Caractéristiques principales	- Un seul schéma stocké sur un site	- Chaque site a une copie du schéma complet	- Chaque site a le schéma des informations locales
Avantages	- Mise à jour facile - Coût de stockage faible	- Coût de communication réduit	- Mise à jour facile - Autonomie de chaque site
Inconvénients	- Coût élevé de communication	- Coût élevé de stockage si taille du schéma importante - Problème de mise à jour des différentes copies	- Coût de communication très élevé - Difficulté d'avoir une vue unique de la BDR

Dans les recherches en cours, la tendance actuelle est de considérer les schémas de la BDR comme des informations se situant au même niveau que celles figurant dans la base répartie SDD-1 [R57]. Cependant, cela ne fait que repousser le problème car il faut alors disposer d'un schéma des schémas indiquant leurs localisations et les éventuelles duplications !

Dans POLYPHEME, on considère avec la notion de Vue Globale l'existence d'un schéma BDR localisé au moins sur le site permettant l'accès à la base répartie (cf. Chapitre 5).

1.3.5.3. Stratégies de traitement des transactions globales

Par transactions globales, nous entendons celles qui s'adressent à tout ou partie de la BDR et dans lesquelles aucune référence explicite n'est faite quant à la localisation des informations.

En quoi le traitement d'une transaction globale dans un SGBDR diffère-t-il de celui à effectuer dans un SGBD ?

Il y a deux différences fondamentales :

1) Il faut tenir compte dans un SGBDR du temps nécessaire pour communiquer d'un site à l'autre : envoi et réception de messages, transferts de données en volumes plus ou moins importants.

2) Le fait qu'il existe plusieurs machines indépendantes implique une possibilité de traitement en parallèle.

Comme nous l'avions souligné au § 1.3.2, la nature de la répartition influence directement le traitement des transactions globales.
Prenons par exemple la requête suivante :
"Trouver les villes des usines fabriquant le produit p2"
qui s'applique sur la BDR B définie au § 1.3.2 par :

R (NUS, NP, VILLE, PRIX, QTE)

et découpée en R1(NUS,VILLE), R2(NP,PRIX), R3(NUS,NP,QTE).

La requête implique les opérations suivantes :

. connaître, grâce à R3, les usines qui fabriquent p2

. connaître, grâce à R1, les villes des dites usines.

Cas 1 :
Avec la répartition suivante (sans redondance) :

Sites	Relations
S1	R1
S2	R2
S3	R3

On a à composer deux relations se trouvant sur des sites différents (S1 et S3) donc nécessairement à effectuer des transferts entre ces deux sites.

Plusieurs stratégies peuvent alors être envisagées :

i) transférer R1 sur S3 ou R3 sur S1 et effectuer la requête. Le choix de la relation à transférer dépend des tailles respectives de R1 et R3 et du coût de transfert [R6, R38] (cf. § 5.5.3).

ii) effectuer sur chaque site une réduction des relations impliquées dans la requête. Dans notre cas, cela revient à effectuer sur S3 la restriction de R3 sur NP=p2. Ne transférer ensuite que les relations réduites avec les mêmes critères de choix que dans i).

iii) faire en sorte que chaque site ait une transaction locale à exécuter en mode coroutine avec les autres transactions locales.
Dans notre exemple, cela signifie que, dès qu'un numéro d'usine est produit par S3, il est transmis à S1 pour qu'il puisse y avoir recherche de la ville correspondante. Cette dernière stratégie est bien sûr inacceptable dans un environnement réparti si l'on doit effectuer un transfert réseau pour un seul n-uplet. Cependant, s'il existe des mécanismes permettant le regroupement des transferts et la synchronisation des transactions locales productrices et consommatrices, cette solution a le gros avantage d'assurer un maximum de parallélisme.
C'est cette stratégie qui a été développée dans POLYPHEME et sur laquelle nous reviendrons (§ 5.5.3).

Cas 2 :
Si nous considérons la répartition suivante (avec redondance) :

Sites	Relations
S1	R1
S2	R2
S3	R1,R2,R3

Le traitement de la transaction globale s'effectue uniquement sur S3 avec pour seul transfert celui des résultats vers le site interrogateur, s'il ne s'agit pas de S3.

A partir de cet exemple, nous constatons que le traitement d'une transaction globale se fait en deux étapes :

1) *Décomposition*, c'est-à-dire découpage de la transaction globale en transactions ne faisant intervenir que les informations d'un seul site (transactions locales). Ce problème a été étudié dans le cadre de POLYPHEME et fait l'objet de la thèse de J.Y. CALECA [R17]. Nous y revenons au chapitre 4.

2) *Exécution*. Comme nous l'avons évoqué dans notre exemple, cette étape fait intervenir des exécutions locales et des transferts.

Deux philosophies différentes sont alors envisagées dans le cadre des SGBDR :

1) *Transfert de données puis exécution*

Cela suppose que l'on duplique les données comme dans SDD-1 ou que l'on effectue un véritable transfert avec suppression du site initial. Dans ce cas, on dispose de systèmes où la localisation des données est complètement dynamique ; elles vont se stocker à l'endroit où on les réclame [R55].
Dans un environnement hétérogène, cette solution implique des mécanismes de traduction des données pour les adapter au site demandeur qui ne les voit pas selon le même modèle. De nombreuses études ont été consacrées à ce sujet, voir en particulier [B55].

2) *Transfert de transactions*

Au lieu d'envoyer les données, on peut envoyer les programmes qui doivent opérer sur ces données ou au moins envoyer des ordres d'exécution de programmes distants.

Ceci soulève les problèmes suivants :

- Est-il possible, à partir d'un site donné, d'envoyer un programme sur un autre site et de commander son exécution ?
Il faut pour cela un langage commun à tous les sites, ainsi que des mécanismes de synchronisation et de surveillance, répartis sur les sites du réseau. Ce problème est traité par le projet IGOR [R24, R25].
Dans le cadre de POLYPHEME, un mécanisme d'appels distants à des procédures PL/1 réparties a été étudié et implanté par E. André et P. Decitre [R11].
A partir de ce mécanisme, un interpréteur chargé de l'exécution et de la synchronisation des transactions locales a été étudié (cf. § 5.5.3).

- Quel niveau de langage faut-il choisir pour communiquer avec les bases locales ?

Ce problème se pose pour un SGBDR homogène, mais est plus complexe dans le cas d'un SGBDR intégré, étant donné l'hétérogénéité des langages. Des travaux ont été effectués concernant la portabilité des programmes utilisant un SGBD [B31] qui peuvent s'appliquer aux bases hétérogènes réparties. Une approche analogue dans POLYPHEME nous a conduit à choisir un langage interactif ou plutôt un langage hôte comme COBOL ou PL/1 pour communiquer avec les SGBDs (SOCRATE par exemple [B2]).

1.3.5.4. Problèmes des mises à jour concurrentes dans les SGBDR

Le problème de l'accès concurrent à un ensemble de données partageables n'est pas un problème spécifique aux SGBDR [B32]. Cependant, il n'est pas facilement soluble dans un environnement réparti.

En effet, une des solutions au problème des mises à jour concurrentes consiste à bloquer les portions de la base concernées par l'opération. Si ces portions sont réparties sur différents sites, on imagine le coût que cela entraîne au niveau des échanges de messages : si une transaction fait intervenir n sites, il faut 5n messages pour assurer le blocage : n messages de blocages, n messages d'acceptation, n transmissions de la mise à jour, n messages de fin de mise à jour et n messages de déblocage. Même en admettant que pendant tout le temps nécessaire à ces opérations tout ce passe bien, c'est-à-dire qu'aucune des bases locales concernées ne tombe en panne, de telles solutions sont inadéquates.

De plus, le blocage de certaines portions de la base répartie peut conduire à des situations de verrou mortel ("deadlock").

De nombreuses recherches sont actuellement faites dans ce domaine et [R8, R38] déjà des solutions ont été proposées pour la conception des SGBDR. La plupart de ces solutions consistent à réduire le coût des opérations de blocage en les groupant avec les opérations de demande de mise à jour. D'autres groupes de solutions consistent à désigner un site primaire qui regroupe toutes les mises à jour concernant un fichier de la base.

Dans SDD-1, une autre méthode est proposée pour éviter autant que possible les blocages de l'ensemble de la BDR [R13].

Cette méthode est basée sur une analyse des interférences que peuvent avoir entre elles les transactions, de manière à pouvoir éviter les interférences nocives. La synchronisation des mises à jour concurrentes est alors réalisée par différents types de protocoles qui diffèrent par le degré de blocage inter-site nécessaire : un seul site concerné, deux, plusieurs ou l'ensemble des sites.

Le choix d'un protocole pour une transaction doit être fait manuellement par l'administrateur de la base répartie qui définit, au moment de la mise en place de la base, des classes de transactions.

Au sein du projet SIRIUS, l'équipe de l'Institut de Programmation de Paris et en particulier G. Gardarin a proposé des mécanismes applicables au SGBDR [R35, R36].

1.3.5.5. Cohérence des copies multiples

Nous avons vu au § 1.3.2 que l'introduction de redondance dans les informations stockées facilitait l'exécution des transactions car elle permettait de minimiser les coûts de transfert. Qu'elle soit systématiquement introduite comme dans SDD-1 [R56] ou qu'elle ne soit qu'accidentelle, la présence de la redondance pose le problème de la cohérence des différentes copies d'un même objet. Ce problème a été traité dans de nombreuses études [R29, R30, R61]. Il rejoint d'ailleurs le problème des mises à jour concurrentes. Ce qu'il faut remarquer c'est qu'il est illusoire dans le cas général de dire qu'à un instant donné, toutes les copies d'un même objet seront dans le même état. C'est pourquoi, la plupart du temps, un certain degré d'incohérence est toléré.

1.3.5.6. Gestion des Pannes dans un SGBDR

Un des principaux avantages qu'offre un système réparti sur un système centralisé est d'être moins vulnérable aux pannes, en ce sens que la défaillance d'un composant ne doit pas arrêter tout le système. Cela signifie en particulier qu'un SGBDR doit pouvoir continuer à fonctionner même si une base locale n'est pas opérationnelle : le principal problème est alors d'assurer la cohérence de la BDR. En effet, considérons une transaction T1 qui met à jour des portions de bases stockées sur les sites S1 et S2. Si une panne se produit pendant l'exécution de T1 qui empêche la mise à jour d'être enregistrée sur S2, alors la base est incohérence : il faut que le système puisse éviter ces mises à jour partielles [R39, R41, R43].

Il est évident que ceci ne peut pas être évité en disant que chaque transaction qui détecte une panne sur un site doit attendre que ce dernier soit opérationnel pour continuer !

Ce problème de gestion des pannes dans un SGBDR peut en fait se découper en différents sous-problèmes :

- *S'assurer que plusieurs sites ont bien reçu un message :*

Lorsqu'un site S1 contrôle une transaction qui met à jour des portions stockées sur S2, S3 et S4, il doit leur envoyer un message déclanchant la mise à jour. Le SGBDR doit alors s'assurer que ce message est bien arrivé aux trois sites concernés.

- *Détection d'une panne :*

Pour s'apercevoir qu'un site est en panne, les techniques habituellement employées consistent à mesurer le temps que met ce site pour répondre à un message. Si ce temps est anormalement long, le site est considéré comme défaillant.

- *Opération avec des sites manquants :*

Les techniques habituelles dans ce genre de situations consistent à faire dans la mesure du possible comme si de rien n'était, mais en mémorisant les opérations que devront faire les sites en panne lorsqu'ils ne le seront plus.

- *Redémarrage d'un site :*

Quand un site redémarre, il faut qu'il soit réintégré dans le système. Pour cela, il faut d'abord qu'il rattrape son retard en effectuant tout ce qu'il aurait dû faire quand il était absent : cela suppose que l'on a quand même continé à lui adresser des opérations à effectuer et qu'elles ont été mémorisées dans le réseau.

- *Partitionnement du réseau :*

Un problème très important et qui n'a pas encore reçu de solution technique générale est celui où une panne isole deux parties du système qui continuent à travailler mais sans pouvoir communiquer entre elles.

1.4. CONCLUSIONS : Propositions d'un cadre de travail

Comme on vient de le voir au long de ce chapitre, le domaine des bases de données réparties bénéficie de la technologie des bases de données et des systèmes répartis.

De ce fait, les problèmes encore non résolus (partiellement ou totalement) dans l'un de ces deux domaines informatiques se retrouvent au niveau des bases réparties et prennent quelquefois d'avantage d'ampleur (par exemple, les mises à jour concurrentes).

En outre, des problèmes nouveaux surgissent (par exemple, le traitement de transactions globales) qui demandent de nombreux efforts et de nombreuses expérimentations.

Pour toutes ces raisons, il semble intéressant de vouloir clarifier ces problèmes en adoptant une approche ascendante. Tout en tenant compte de ce qui existe aujourd'hui dans le domaine des bases de données et dans le domaine des réseaux d'ordinateurs, comment concevoir un système de bases de données réparties, qui soit général ?

Cette conception doit aller de pair avec une expérimentation pour éviter les solutions irréalistes.

Cette approche est à la base du projet POLYPHEME auquel notre travail est consacré et sur lequel nous reviendrons en détails dans les chapitres suivants.

Chapitre 2

Un Modèle Général de Données Réparties : MOGADOR

Il nous faut peu de mots
pour exprimer l'essentiel ;
il nous faut tous les mots
pour le rendre réel.

P. ELUARD

Sommaire

2.4. MOGADOR COMME MODELE CONCEPTUEL DE BASES DE DONNEES

2.4.1. Catégories et Fonctions.

2.4.2. Passage à une vue relationnelle.

2.5. MACHINE RELATIONNELLE

2.5.1. Description de Vue Relationnelle.

2.5.2. Opérations sur une vue relationnelle.

2.5.3. L'exploitation d'une vue relationnelle.

2.5.3.1. Opérateurs ensemblistes.

2.5.3.2. Opérateurs relationnels.

2.5.3.3. Opérateurs de calculs.

2.5.3.4. Transactions et requêtes.

2.5.4. Correspondance noms-éléments : règles d'évolution.

2.1. POURQUOI UN MODELE GENERAL DE DONNEES REPARTIES ?

2.1.1. Problèmes de Conception

Le problème majeur auquel on est confronté dans la conception d'un Système de Gestion de Bases de Données Réparties (ou SGBDR) peut se résumer de la manière suivante :

> Etant donné :
>
> 1) des informations, des données,
> 2) des moyens de stockage et de traitement, éventuellement hétérogènes,
> 3) des moyens d'accès et de manipulation de données (eux aussi éventuellement hétérogènes),
> 4) des utilisateurs,
>
> le tout étant réparti sur les différents sites d'un réseau de communication, quel modèle de données, quels langages, quels outils faut-il fournir à ces utilisateurs pour qu'ils puissent :
>
> 1) exprimer la sémantique des informations réparties,
> 2) manipuler ces informations comme si elles constituaient une base de données unique, homogène.

Ces deux points semblent contradictoires, mais les deux possibilités offertes ne s'adressent pas aux mêmes types d'utilisateurs, à savoir ceux qui conçoivent la base de données répartie et ceux qui l'exploitent.

Afin d'éviter, dans la suite du texte, une telle ambiguïté, nous emploierons le terme générique d'*administrateur* pour désigner la (ou les) personne(s) chargée(s) de concevoir et de mettre en place la base répartie tandis que le terme d'*utilisateur* désignera plutôt ceux qui exploitent cette base grâce à un ensemble de programmes d'application.

Il convient alors de distinguer les deux points suivants :

- celui de la conception d'un SGBDR en tant qu'outil informatique (architecture logicielle),
- celui de l'utilisation d'un tel outil pour concevoir, mettre en place et exploiter des bases de données réparties (méthodologie de conception d'une base de données).

A notre sens, la conception d'un SGBDR doit passer par une phase de modélisation ou en s'affranchissant de toutes contraintes d'ordre technique on caractérise les éléments fondamentaux du système : les "objets" qu'il manipule, les fonctions qu'il offre.

Ces fonctions et ces objets dépendent en fait des hypothèses dans lesquelles on se place pour la conception du SGBDR ; aussi, allons-nous tout d'abord décrire le cadre de notre travail :

2.1.2. Cadre de travail : Projet POLYPHEME

Notre travail se place dans le cadre du projet POLYPHEME (cf. § 1.3.1.2) dont le but est la conception et la réalisation d'un système de coopération de bases de données hétérogènes, réparties sur les différents sites d'un réseau d'ordinateurs.

L'approche développée dans la conception d'une base répartie est de type ascendant, en ce sens qu'elle part de bases existantes que l'on désire faire coopérer. Cette approche répond à de nombreux cas réels rencontrés dans de grosses entreprises ou administrations [R47, R60, B31] qui ont développé et qui exploitent indépendamment plusieurs bases de données. Pour mettre en place de nouvelles applications, il faut alors envisager une coopération de ces bases sans pour autant revenir à un système centralisé qui irait à l'encontre de l'organisation décentralisatrice de l'entreprise.

Les hypothèses de travail du projet POLYPHEME sont les suivantes :

* H1 - Il existe a priori plusieurs bases de données dites *bases locales* qui diffèrent :
  - par le modèle utilisé pour leur description (hiérarchies, réseaux)
  - par le système (SGBD local) sous lequel elles sont implantées et exploitées (SOCRATE, IDSII, I.M.S.).

* H2 - Ces bases locales ont entre elles une sémantique commune permettant de rapprocher tout ou partie des ensembles d'informations qu'elles contiennent.

* H3 - Il existe un réseau général d'ordinateurs (type réseau CYCLADES) fournissant des moyens de communication entre les sites sur lesquels sont implantées ces bases.

Vouloir définir et mettre en place de nouvelles applications impliquant une vision globale de ces différentes bases locales suppose que soient réalisées :

- l'*homogénéisation* de chacune des bases
- leur *intégration* pour constituer une base de données répartie.

D'autre part on désire que le mécanisme de coopération ne remette pas en cause l'exploitation locale de chaque base par la communauté d'utilisateurs déjà existante.

Pouvoir prendre en compte différents types de SGBD suppose que soit défini un modèle commun que devra offrir le système de coopération.

De nombreux travaux ont montré que c'est le modèle relationnel de Codd qui répond le mieux aux objectifs d'un modèle commun de données [B4, B17].

- par sa *simplicité* dans la description logique des structures d'informations comme des tables de valeurs, il assure ainsi une grande indépendance vis-à-vis des représentations physiques des données qui peuvent être hétérogènes
- par sa *généralité*, il permet d'uniformiser la représentation des modèles hiérarchiques ou réseaux, rendant ainsi compatibles divers systèmes existants [B5, B6]
- par son *adaptabilité* à l'évolution des structures de l'information dans un environnement en perpétuel mutation.

Pour toutes ces raisons, le Système de Gestion de Bases de Données Réparties (SGBDR) que nous proposons se présente comme un réseau de machines relationnelles offrant aux administrateurs et aux utilisateurs un langage et des outils pour décrire et manipuler des vues relationnelles.

Nous pensons cependant que le modèle relationnel n'est pas suffisant pour traduire toute la sémantique d'une base de données [B1, B54], ce qui est un problème particulièrement important dans un environnement réparti.

Ceci nous a amené à définir un modèle de type conceptuel : *MOGADOR* ou *Modèle Général de Données Réparties*.

Notre but était de développer des outils conceptuels pour fournir un cadre méthodologique permettant une approche rationnelle de la conception des bases réparties.

MOGADOR permet de décrire de façon statique des informations, mais possède également des notions dynamiques pour décrire la manière dont doivent évoluer ces informations.

En effet, sur le plan méthodologique de l'utilisation de l'outil que constitue le SGBDR, le problème qui se pose à l'administrateur est de définir quelles sont les tables de valeurs (les relations) et de concevoir une base de données relationnelles qui soit la plus cohérente possible.

A lui seul ce problème constitue aujourd'hui un champ de recherche très important dans le domaine des bases de données [B29, B39, B33]. En définissant MOGADOR, nous n'avons pas cherché à résoudre ce problème dans le cas général. Nous nous sommes placés dans le cadre plus restreint de la conception de bases réparties et nous avons voulu apporter quelques éléments pour faciliter la définition de vues relationnelles (cf. § 2.1.3).

Disposant ainsi d'outils conceptuels, pourquoi ne pas nous en servir alors pour réaliser la phase de modélisation de l'architecture du SGBDR lui-même ? Ainsi, nous pouvons, grâce à MOGADOR, décrire les informations que manipule le système, mais assurer également l'homogénéisation des moyens de stockage, d'accès et de manipulation, en les considérant comme des automates ou des machines logiques. Nous ne voulions pas que MOGADOR fournisse un nouveau type de système de base de données avec un nouveau modèle, de nouveaux langages, car cela aurait eu pour effet de refermer le système sur lui-même sans aucune possibilité d'ouverture vers le monde extérieur.

2.1.3. MOGADOR : des outils conceptuels de modélisation

L'approche que nous avons eu dans la définition de MOGADOR présente certaines analogies avec les travaux concernant les modèles de données qui mettent l'accent sur les niveaux d'abstraction, de façon à distinguer entre elles :

- la sémantique des informations
- la structure logique des informations
- la description des chemins d'accès
- la structure physique de représentation.

Nous pouvons citer à ce sujet le modèle DIAM de M. Senko [B52], le "role model" de C. Bachman [B13], le modèle "entité-relation" de P. Chen [B50], celui de J.R. Abrial "Data Semantics" [B1] et plus récemment les travaux de G. Bracchi [B12] et ceux de J.M. Smith et D.C. Smith [B54].

En tant que modèle conceptuel, MOGADOR permet de s'intéresser à la portion du monde réel que l'on désire modéliser : on utilise alors les notions de *catégories* d'*éléments* et de *fonctions* pour décrire les objets du monde réel et les associations entre ces objets tels que nous les percevons.

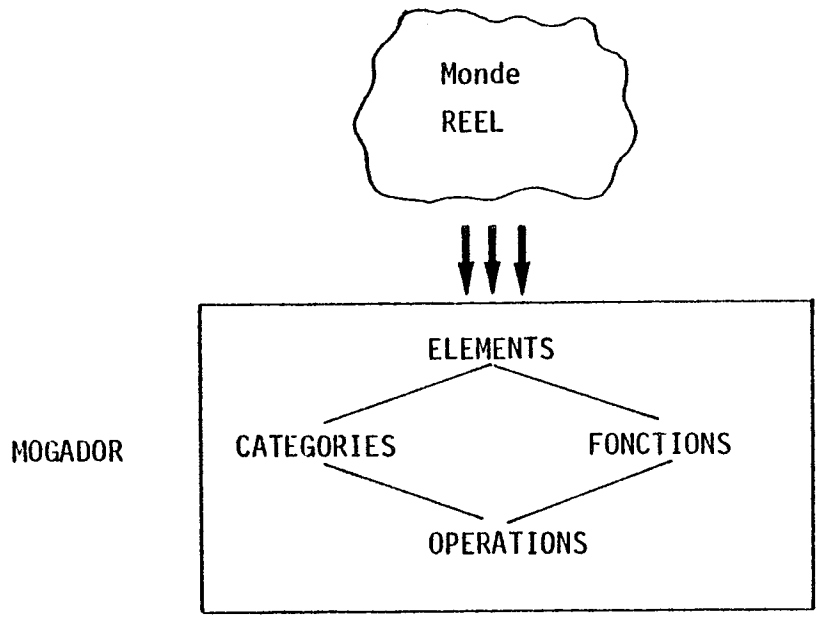


Figure 2.1 - Phase de modélisation

Cette portion du monde réel va nécessairement évoluer. A ce niveau, on utilise les notions d'opérations sur les catégories et les fonctions pour prendre en compte cet aspect dynamique (Figure 2.1).

Toutes ces notions sont décrites en détail dans ce qui suit.

MOGADOR repose sur la théorie des ensembles, ce qui le rend indépendant des systèmes de bases de données actuels. Nous avons utilisé comme base de départ les travaux sur le modèle de Codd [B17, B68], ceux de C. Delobel [B30] et surtout le modèle "Data Semantics" de J.R. Abrial [B1].

Notre première approche a été de partir des données réparties sous forme de bases locales pour dégager les concepts fondamentaux nécessaires à une coopération de ces bases. Cette approche a été décrite en détail dans un rapport de recherche [R3]. Le principal objectif était alors de pouvoir décrire une répartition des données aussi fine que possible, tout en conservant la sémantique des informations :

1) Les catégories et les fonctions fournissent les *atomes sémantiques de répartition* (ou niveau sémantique minimum évoqué dans [R21]).

2) L'approche relationnelle binaire à base de catégories et de liaisons entre catégories, exprimées à l'aide des fonctions, offre le maximum de souplesse vis-à-vis des changements intervenant dans la nature des catégories et/ou des liaisons.

3) La théorie des relations et l'algèbre relationnelle permettent une approche récurrente dans la définition d'une vue relationnelle à partir d'une autre et ainsi de suite. Ce point est particulièrement important dans une approche ascendante des bases réparties.

Du point de vue de l'administrateur d'une base de données répartie dans un environnement hétérogène, MOGADOR peut être utilisé à deux niveaux (Figure 2.2).

1) Au niveau de chaque base locale (cf. Chapitre 3), il s'agit alors d'interpréter le schéma local en termes relationnels grâce aux concepts de catégories et de fonctions (aspect statique). Il s'agit ensuite de définir les mécanismes de correspondance entre la vue relationnelle ainsi construite et les outils (description et manipulation) fournis par le système de Base de Données Local (SGBD local). On assimile les

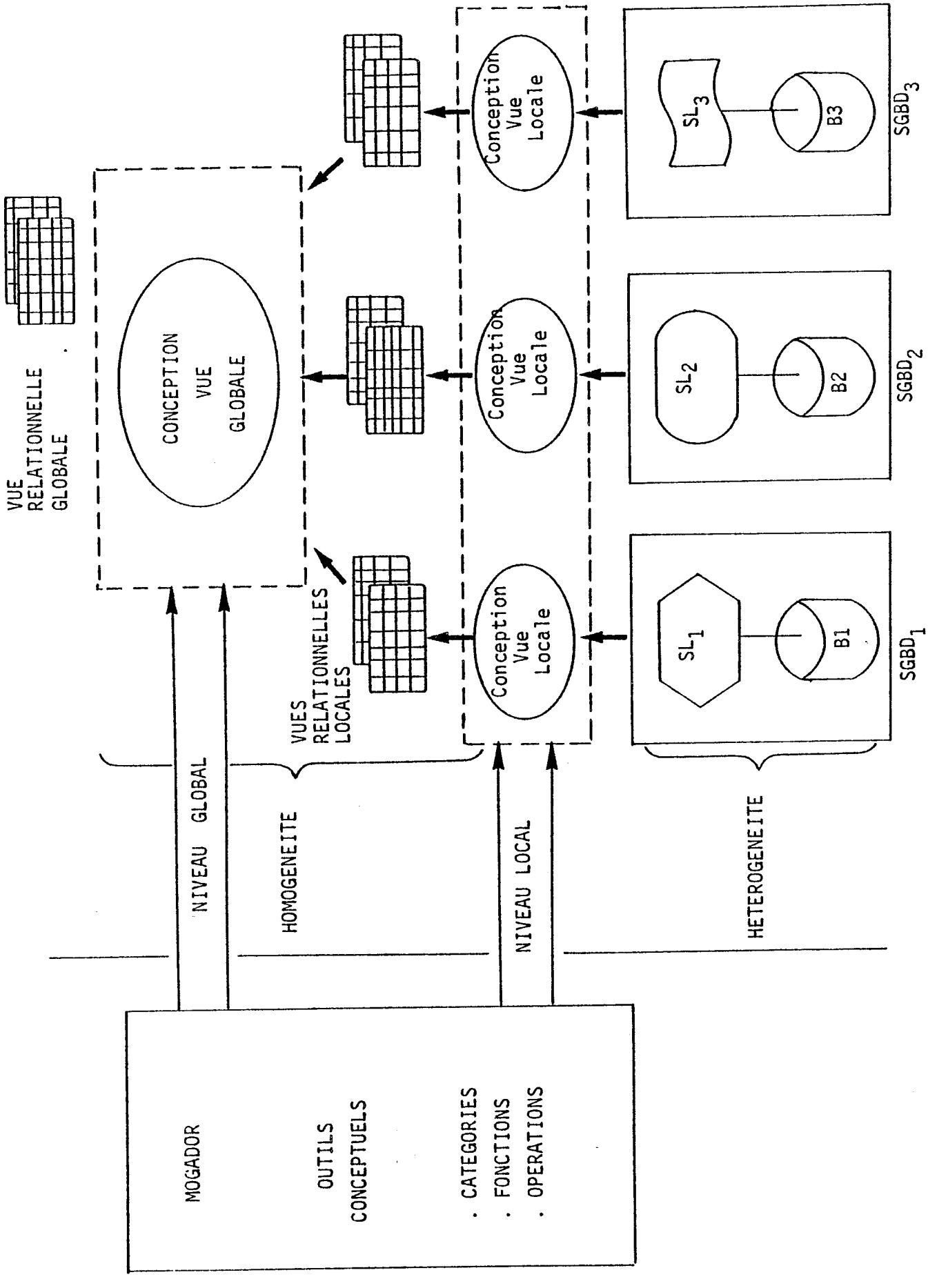


Figure 2.2 - MOGADOR et la coopération de bases de données hétérogènes

informations de la base à des tables de valeurs et le système à une machine logique capable d'effectuer des opérations standards d'accès et de manipulation (aspect dynamique).

2) Au niveau de la conception et de la mise en place de la base répartie (niveau global, cf. Chapitre 4), il s'agit de rapprocher les vues des différentes bases locales ainsi homogénéisées, de manière à en déduire une nouvelle vue relationnelle : la vue globale (aspect statique).
Il s'agit ensuite de définir les mécanismes de correspondance entre niveaux global et local pour faire en sorte que les opérations s'adressant au niveau global soient correctement répercutées sur les informations contenues dans les bases. On assimile alors l'ensemble des informations réparties, les machines locales et ces mécanismes de correspondance à une machine logique (aspect dynamique).

2.2. LES PRINCIPAUX CONCEPTS DE MOGADOR

Il y a *trois concepts fondamentaux* dans MOGADOR :

1 - Le concept d'*ELEMENT* : machine, usager, objet, nom, programme, processus.
2 - Le concept de *CATEGORIE* : ensemble d'éléments.
3 - Le concept de *FONCTION* : liaison entre une catégorie source et une catégorie cible.

Ces concepts sont utilisés à *trois niveaux* (Figure 2.3) :

1 - Le *méta-niveau* qui correspond à la définition de MOGADOR lui-même avec des méta-catégories et des méta-fonctions (cf. § 2.3).
2 - Le *niveau global* qui correspond à une première catégorie de machines logiques (les machines globales) permettant chacune la définition et la manipulation d'une base répartie.
3 - Le *niveau local* qui correspond à une seconde catégorie de machines logiques (les machines locales) construites autour d'une base coopérante et constituant les composantes d'une base répartie. Ces machines assurent le stockage et l'accès aux données locales.

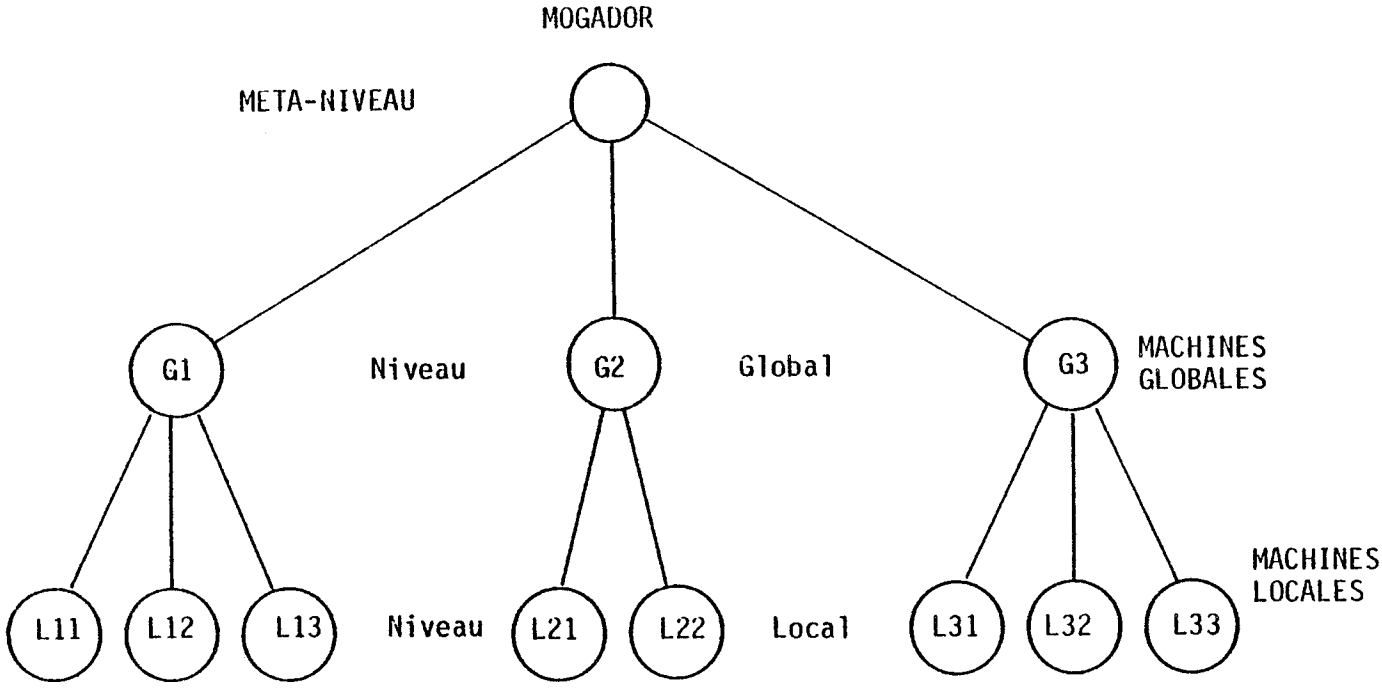


Figure 2.3 - Les Niveaux MOGADOR

2.2.1. Les éléments

2.2.1.1. La notion d'objet : objets imples, objets composés

- Objet simple :

C'est une valeur élémentaire appartenant à l'un des ensembles suivants (Méta-Catégories) :

- ENTIER : ensemble des entiers (Z)
- REEL : ensemble des réels (ℝ)
- LOGIQUE : ensemble = {vrai, faux}
- CHAINE : ensemble de chaînes de caractères.

- Objet composé :

C'est un élément d'un produit cartésien de méta-catégories définies ci-dessus (un n-uplet).

Par exemple <u57, PARIS, 525, 150000.00>

appartient à l'ensemble : CHAINE × CHAINE × ENTIER × REEL

où "×" dénote le produit cartésien.

2.2.1.2. La notion de nom

Dans MOGADOR, un nom est un élément de la catégorie CHAINE et il désigne soit une catégorie (on le notera alors avec des majuscules), soit une fonction (on le notera alors en minuscules).

Par exemple : - "AGE" est le nom de la catégorie AGE

- "longueur" est le nom de la fonction qui associe à une chaine c (c ∈ CHAINE) sa longueur ℓ (ℓ ∈ ENTIER)

ℓ = longueur (c).

2.2.1.3. Machine MOGADOR

Une machine MOGADOR est un élément composé de (Figure 2.4) :

- un ensemble d'éléments dit *espace des éléments* (figuré par un cercle)
- un ensemble de noms ou *espace de noms* (figuré par un rectangle)

qui correspond à la description de l'espace des éléments :

- noms des catégories
- noms des fonctions

- un *mécanisme de correspondance* entre espace des noms et espace des éléments (figuré par un hexagone).

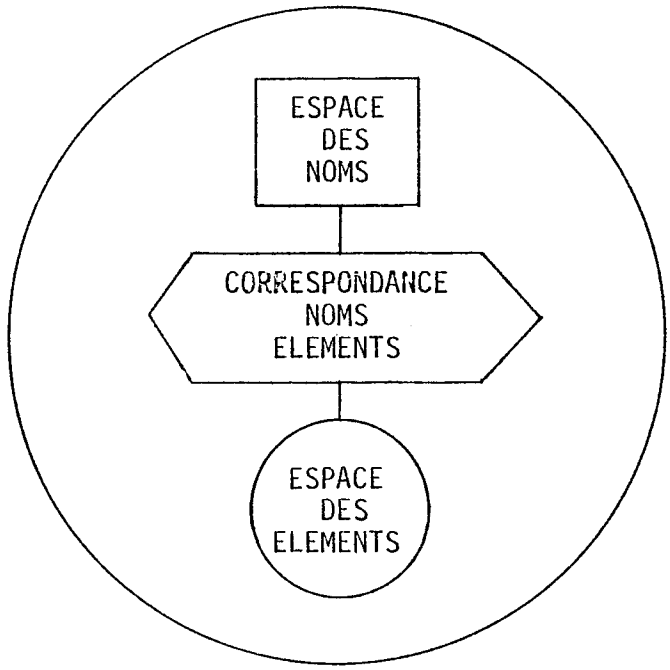


Figure 2.4 - Une machine MOGADOR

Fonctionnellement, une machine est capable d'effectuer sur les catégories et les fonctions les opérations définies aux § 2.2.2.2 et 2.2.3.2.

La méta-catégorie MACHINE regroupe l'ensemble des machines MOGADOR.

Une machine particulière constitue un élément de cette méta-catégorie et en tant que tel peut se retrouver dans l'espace des éléments d'une autre machine. C'est le cas par exemple des machines locales sur lesquelles est construite une machine globale (cf. § 4).

2.2.1.4. Notion d'usager

Un usager est celui qui s'adresse à une machine :

- soit pour définir l'espace des noms et la correspondance avec l'espace des éléments (rôle d'administration)
- soit pour manipuler les éléments grâce à toute une série d'opérations : création, suppression, modification d'un élément (rôle d'utilisation).

2.2.1.5. Notion de Programme et de Processus

- Un *programme* est un ensemble d'opérations exécutables par une machine (§ 2.2.2.2 et 2.2.3.2).

Un programme est un élément composé : il a un *nom*, des *entrées*, des *sorties* et les opérations qui le composent constituent le *corps* du programme.

- Un *processus* est l'exécution d'un programme donné (par son nom) par une machine donnée.

2.2.2. Les catégories

Conformément à la théorie des ensembles, une catégorie MOGADOR est un ensemble d'éléments (cf. § 2.2.1). Chaque catégorie est identifiée par un nom (cf. § 2.2.1.2).

Nous distinguerons deux sortes de catétories :

- les catégories abstraites (cf. § 2.2.2.3)
- les catégories concrètes (cf. § 2.2.2.4).

2.2.2.1. Définition de Catégories

Les méta-catégories (cf. § 2.3) servent à la définition de nouvelles catégories. En général, une catégorie peut être définie au moyen d'autres catégories déjà définies et ceci grâce aux opérateurs suivants :

a) *Affectation* (noté ":=")

B := A définit la catégorie B comme identique à A

B := A $\iff$ B = {a | a $\in$ A}.

b) *Produit cartésien* (noté "×")

A × B = {(a,b) | a $\in$ A $\wedge$ b $\in$ B}.

Dans ce cas, un élément du produit cartésien est un couple d'éléments. Il est possible, grâce à des fonctions de sélection (cf. § 2.2.3.3), d'isoler l'un des éléments du couple ou plus généralement d'un n-uplet.

c) *Union* (notée "∪"), *Intersection* (notée "∩"), *Différence* (notée "-")

A ∪ B = {x | x $\in$ A $\vee$ x $\in$ B}

A ∩ B = {x | x $\in$ A $\wedge$ x $\in$ B}

A - B = {x | x $\in$ A $\wedge$ x $\notin$ B}.

d) *Restriction* ("prédicat")

Cet opérateur s'utilise pour caractériser un sous-ensemble d'une catégorie donnée :

A (prédicat) = {a $\in$ A | prédicat (a)}.

Un prédicat peut être vu comme une méta-fonction MOGADOR (cf. § 2.3) qui a pour ensemble cible : LOGIQUE.
Quant à l'élément source de cette méta-fonction, il dépend de la catégorie sur laquelle on applique le prédicat.

Ainsi sur ENTIER et REEL, on pourra définir des sous-ensembles en donnant des intervalles de valeurs :

ENTIER (x..y) = {z $\in$ ENTIER | z $\geq$ x $\wedge$ z $\leq$ y}

REEL (x..y) = {z $\in$ REEL | z $\geq$ x $\wedge$ z $\leq$ y}.

Sur CHAINE, on considérera les deux prédicats suivants :

* égalité :

CHAINE (C1,C2,...,Cn) = {c $\in$ CHAINE | c = C1 $\vee$ c = C2 $\vee$ ... $\vee$ c = Cn}.

* contraintes de longueur :

A toute chaîne est associée sa longueur qui est le nombre (entier) de caractères qui la composent.

En fait, "longueur" est une méta-fonction qui va de CHAINE à ENTIER.

On notera :

CHAINE(x) = {c ∈ CHAINE | longueur (c) ≤ x}.

Exemples :

1. AGE := ENTIER (18..65) définit la catégorie AGE comme l'ensemble des entiers compris entre 18 et 65 (ensemble d'objets simples).

2. JOUR := ENTIER (1..31)
   MOIS := ENTIER (1..12)
   AN := ENTIER (0..99)
   DATE := JOUR × MOIS × AN.

DATE est un ensemble d'objets composés (triplets (j,m,a) avec j ∈ JOUR, m ∈ MOIS et a ∈ AN).

3. CCL := CHAINE (8) définit la catégorie CCL comme un ensemble de chaînes de longueur inférieure ou égale à 8.

4. COULEUR := CHAINE ('BLEUE', 'ROUGE', 'JAUNE', 'VERT') définit COULEUR comme l'ensemble :

{BLEUE, ROUGE, JAUNE, VERT}.

2.2.2.2. Opérations sur les Catégories

Quatre opérations primitives sont définies sur les catégories :

1. Créer un élément dans une catégorie

Si CAT est une catégorie quelconque et e un élément, on note l'opération de création de la manière suivante :

(1) créer (CAT, e)

ou bien (2) cat e.

La forme (1) est dite forme fonctionnelle (cf. § 2.3).

Le résultat d'une telle opération est un LOGIQUE : vrai si l'opération de création s'est faite normalement, faux sinon.

2. Supprimer un élément dans une catégorie

On note cette opération : supprimer (CAT, e) et le résultat est également un LOGIQUE : vrai si la suppression a pu se faire, faux sinon.

3. Tester si un élément donné appartient à une catégorie

On note cette opération :

(1) tester (CAT, e) ou bien (2) e ϵ CAT.

Le résultat est un logique.

4. Enumérer tous les éléments d'une catégorie

Cette opération permet d'obtenir tous les éléments d'une catégorie (bien entendu, celle-ci devra être finie).

On note cette opération :

(1) énumérer (CAT) ou bien (2) CAT tout court et dans ce cas, cela signifie que l'on assimile le nom d'une catégorie avec l'ensemble de ses éléments.

Exemple :

énumérer (COULEUR) (cf. § 2.2.2.1)

donne comme résultat :

{BLEU, ROUGE, JAUNE, VERT}.

2.2.2.3. Catégories Abstraites (CA)

Ce sont des ensembles d'*objets simples ou composés* sur lesquels on ne peut pas appliquer les opérations de création et de suppression.
Cela signifie que les objets abstraits existent a priori dans l'univers où l'on se place et qu'ils peuvent être ainsi utilisés directement.

Ainsi, les méta-catégories :

ENTIER, REEL, LOGIQUE, CHAINE sont des catégories abstraites. Toute catégorie construite sur ces méta-catégories seront également des catégories abstraites :

par exemple : MOIS := ENTIER (1..12).

On notera CA la méta-catégorie comprenant l'ensemble des noms de catégories abstraites MOGADOR.

2.2.2.4. Catégories Concrètes (CC)

Ce sont des *ensembles d'éléments* sur lesquels les opérations décrites au § 2.2.2.2 sont définies.

Une Catégorie Concrète MOGADOR est définie par :

1. Un Nom, par exemple PERSONNE, COMMANDE, MACHINE.
2. Un Nom d'Identificateur, c'est-à-dire le nom d'un ensemble d'objets simples ou composés servant à identifier les éléments de la catégorie concrète (ils sont donc tous distincts).

Considérons par exemple un ensemble de personnes. Pour identifier les différentes personnes, on utilise le numéro INSEE qui peut être vu lui-même comme un objet composé :

SEXE := ENTIER (1,2)
AN := ENTIER (0..99)
MOIS := ENTIER (1..12)
DPT := ENTIER (1..99)
NUMERO := ENTIER (0..999999)
INSEE := SEXE × AN × MOIS × DPT × NUMERO.

Toutes les catégories précédentes sont des catégories abstraites. On peut définir la catégorie concrète PERSONNE identifiée par INSEE.
Ainsi, créer un élément de la catégorie PERSONNE, c'est créer un numéro INSEE :
par exemple : créer (PERSONNE, 1 44 08 99 350 300).

Soit CC la méta-catégorie (concrète) constituée de toutes les catégories concrètes, et soit NI la méta-catégorie constituée de tous les produits cartésiens de catégories abstraites servant d'identificateur :

PERSONNE $\in$ CC
INSEE $\in$ NI.

La notion de Catégorie Concrète avec son identificateur correspond dans le modèle de Codd à la notion de relation et de clé. Cependant, cette notion recouvre deux formes diverses d'abstraction, à savoir [B54] :

- *généralisation* où des entités similaires sont considérées comme faisant partie d'une même catégorie. Par exemple, des hommes et des femmes rassemblés dans la catégorie PERSONNE ou EMPLOYE
- *agrégation* qui correspond à la matérialisation de liaisons entre entités. Par exemple, un client, une date, un numéro de vol constituant un élément de la catégorie RESERVATION.

Dans MOGADOR, ces deux formes d'abstraction peuvent être explicitées grâce à la notion de *fonction*.

Voyons d'abord comment s'appliquent les opérations définies plus haut.

Soit C une Catégorie Concrète quelconque et I son identificateur (dans le cas général I = I1 × I2 × ... × In, chaque Ij étant une catégorie abstraite).

- *Créer* un élément de C, c'est exprimer qu'un élément de I fait partie de l'ensemble C :
i ∈ I, créer (C, i) a une valeur vraie si et seulement si il n'existe pas dans C un autre élément identique à i.

- *Supprimer* un élément i de C, c'est exprimer le fait que i n'appartient plus à l'ensemble C.
Supprimer (C, i) a la valeur vraie sii l'élément i appartenait à C.

- *Tester* si i ∈ C, c'est regarder si parmi les éléments de C on trouve i et dans ce cas la valeur de tester (C, i) est vraie.

- Enfin *énumérer* tous les éléments d'une catégorie concrète revient à former un ensemble d'éléments : cet ensemble est en fait un ensemble d'objets composés :
énumérer (C) ⊆ {i ∈ I}.

2.2.3. Les Fonctions

La notion de fonction (ou d'application) est une notion mathématique bien connue. Cette notion présente un double aspect :

1) *Aspect Statique :* l'existence d'une liaison orientée entre deux ensembles : par exemple, une fonction f entre A et B. A est la *source*, B la *cible*.

2) *Aspect Dynamique :* étant donné un élément de l'ensemble source (a ∈ A) et une fonction f, comment obtenir les éléments f(a) correspondant ? Si la fonction f est complètement déterminée par son *graphe* (c'est-à-dire par l'ensemble des couples (a, f(a)), alors étant donné a on peut obtenir f(a) en consultant ce graphe. Une autre possibilité est d'appliquer à l'élément a une série d'opérations ; on dispose alors de l'équation de la fonction.

En appliquant cette notion aux modèles de données, on dispose d'un outil très puissant pour exprimer et exploiter les liaisons entre catégories d'éléments : Abrial [B1, G1].
Quels sont les éléments qui caractérisent une fonction MOGADOR ?

2.2.3.1. Définitions

Une fonction MOGADOR est définie par :

- son *nom* (écrit en minuscules)
- une *catégorie source*
- une *catégorie cible*
- un type, à savoir si elle est *monovaluée* ou *multivaluée* :

Si une fonction est monovaluée, cela signifie que pour un élément de la catégorie source on obtient un élément et un seul de la cible :

- soit f de A vers B si $a \in A$ $f(a) \in B$.

Si une fonction est multivaluée, cela signifie que pour un élément de la source on obtient un ensemble d'éléments de la cible :

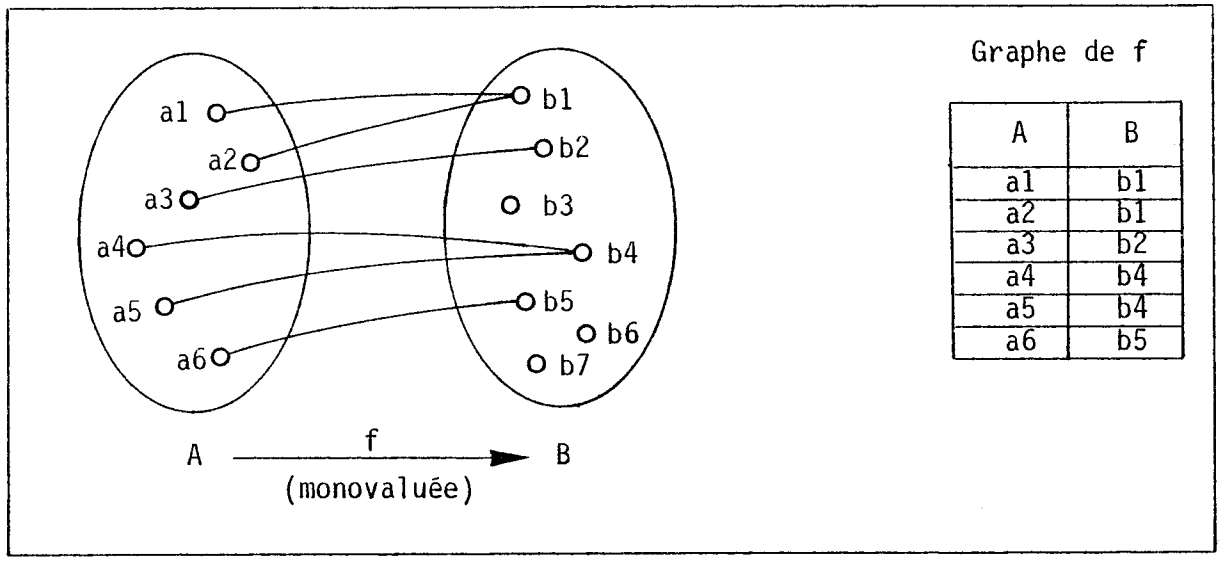
- soit g de A vers B si $a \in A$ $g(a) \subseteq B$.

A chaque fonction est associé un *graphe* qui est un ensemble d'objets composés. Ce graphe peut être donné de deux manières différentes :

- explicitement par la liste des couples (a, f(a)) :

(analogue à la définition d'un ensemble en extension) (Figure 2.5.a) ;

- implicitement par une expression donnant la série d'opérations à appliquer à un élément a pour déterminer f(a) (Figure 2.5.b). On emploiera alors le terme de fonction calculée.



Graphe de f

A	B
a1	b1
a2	b1
a3	b2
a4	b4
a5	b4
a6	b5

Figure 2.5.a - Une fonction f monovaluée avec graphe explicite

$$\text{ENTIER} \times \text{ENTIER} \xrightarrow[\text{(monovaluée)}]{\text{modulo}} \text{ENTIER}$$

pour x et y $\in$ ENTIER modulo (x,y) = x-y * (x/y)

Figure 2.5.b - Une fonction calculée

A chaque fonction est associée son *inverse* qui est ou non définie.
On note inv f l'inverse d'une fonction quelconque f :
si on a f avec source (f) = A et cible (f) = B
alors inv f est telle que source (inv f) = cible (f) = B
et cible (inv f) = source (f) = A.

N.B. "Source" et "cible" sont des méta-fonctions définies au § 2.3.

Désignons par FMONO et FMULTI respectivement l'ensemble des fonctions monovaluées et multivaluées MOGADOR.
Notons F = FMONO ∪ FMULTI.

Le tableau suivant donne les trois cas possibles de fonctions MOGADOR selon le type de l'inverse :

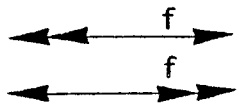
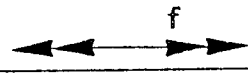
Cas	f	inv f	Notation graphique
1	$\in$ FMONO	$\in$ FMONO	
2	$\in$ FMONO $\in$ FMULTI	$\in$ FMULTI $\in$ FMONO	
3	$\in$ FMULTI	$\in$ FMULTI	

Table 2.1 - Les 3 cas de fonctions

On désignera par "id" la fonction *identité*.

$$id(a) = a \quad \forall a \in A.$$

2.2.3.2. Opérations concernant les fonctions

Il y a quatre opérations fondamentales concernant les fonctions :

1) *Accès*. Etant donné un élément source et une fonction, obtenir le ou les éléments cibles selon que la fonction est mono ou multivaluée.

- Si f est une fonction de A vers B.

On note cette opération :

(1) - soit accéder (f, a)

(2) - soit f(a).

Si $f \in$ FMONO et $a \in A$ alors $f(a) \in B$.

Si $f \in$ FMULTI et $a \in A$ alors $f(a) \subseteq B$.

- Par extension, f peut s'appliquer à un ensemble : elle s'applique successivement à chaque élément :

Si $X = \{x1,x2,\ldots,xn\} \subseteq A$ alors la notation f(X) est équivalente à :

$$(f(x1), f(x2), f(x3), \ldots, f(xn)).$$

2) *Relier deux éléments* entre eux par une fonction si elle est monovaluée, ou un élément à un ensemble dans le cas d'une multivaluée.

- Si f est une fonction de A vers B, on a

Si f ∈ FMONO on note lier(f,a,x) ou f(a) ← x.
Si f ∈ FMULTI on note lier(f,a,X) ou f(a) ← X.
Le résultat d'une telle opération est une valeur logique (vraie ou faux).
Cela signifie que l'expression lier(f,a,x) a pour valeur un logique.

3) *Délier* (inverse de l'opération précédente).
On la note délier (f,a,x) ou f(a) ≠ x si f ∈ FMONO
ou bien délier (f,a,X) ou f(a) ≠ X si f ∈ FMULTI.
Le résultat est une valeur logique.

N.B. Dans le cas d'une multivaluée, l'opérateur de différence (-) peut être utilisé pour enlever un ou plusieurs éléments de l'ensemble cible reliés à un élément source.

Exemple : $f(a) \leftarrow f(a) - \{x_k\}$.

4) *Obtenir le graphe d'une fonction*, c'est-à-dire former l'ensemble des couples {a, f(a)} pour tous les a pour lesquels f est définie, on note :
graphe (f) ou plus simplement f en assimilant le nom de la fonction à l'ensemble des objets composés du graphe.

Pour une fonction quelconque f de A vers B, le graphe de f (qui est aussi celui de son inverse) est un sous-ensemble du produit cartésien A × B.

Soit f telle que source (f) = A et cible (f) = B
graphe (f) $\subseteq$ A × B.

2.2.3.3. Sélecteurs

Lorsque l'on définit une catégorie C comme un produit cartésien d'autres catégories C1 × C2 × ... × Cn, on définit implicitement n fonctions monovaluées ayant pour source C et pour cible l'une des Ci. Ces fonctions seront appelées "*sélecteurs*".

Par convention, le nom du sélecteur sera celui de la catégorie cible notée en minuscule.

Exemple
Soit INSEE := SEXE × AN × MOIS × DPT × NUMERO.

5 fonctions monovaluées sont ainsi implicitement définies : sexe, an, mois, dpt, numéro

si 1, 44, 08, 99, 350, 300 ∈ INSEE

alors sexe (1,44,08,99, 350, 300) = 1.

2.3. LE META-NIVEAU MOGADOR

Ce méta-niveau peut être assimilé à une machine logique (abstraite) qui correspond fonctionnellement à un système de gestion de bases de données réparties, tel que nous l'avons défini au chapitre 1. Par analogie avec les machines MOGADOR, on considère alors un espace de noms dans lequel figurent des *méta-catégories* et des *méta-fonctions*.

L'espace des éléments est essentiellement constitué de machines globales, chacune d'elles englobant plusieurs autres machines (Figure 2.6).

Le tableau ci-dessous donne la composition des différents espaces pour chaque niveau de machines :

MACHINE	ESPACE DES NOMS	ESPACE DES ELEMENTS
"SYSTEME"	- META-CATEGORIES (§2.3.1) - META-FONCTIONS (§2.3.2)	MACHINES Globales
GLOBALE	CATEGORIES ET FONCTIONS GLOBALES	MACHINES Locales MACHINES Globales
LOCALE	CATEGORIES ET FONCTIONS LOCALES	- OBJETS LOCAUX (simples, composés) - PROGRAMMES LOCAUX

Chaque machine globale fait donc fonctionner n machines locales ou globales (cf. § 4.1).

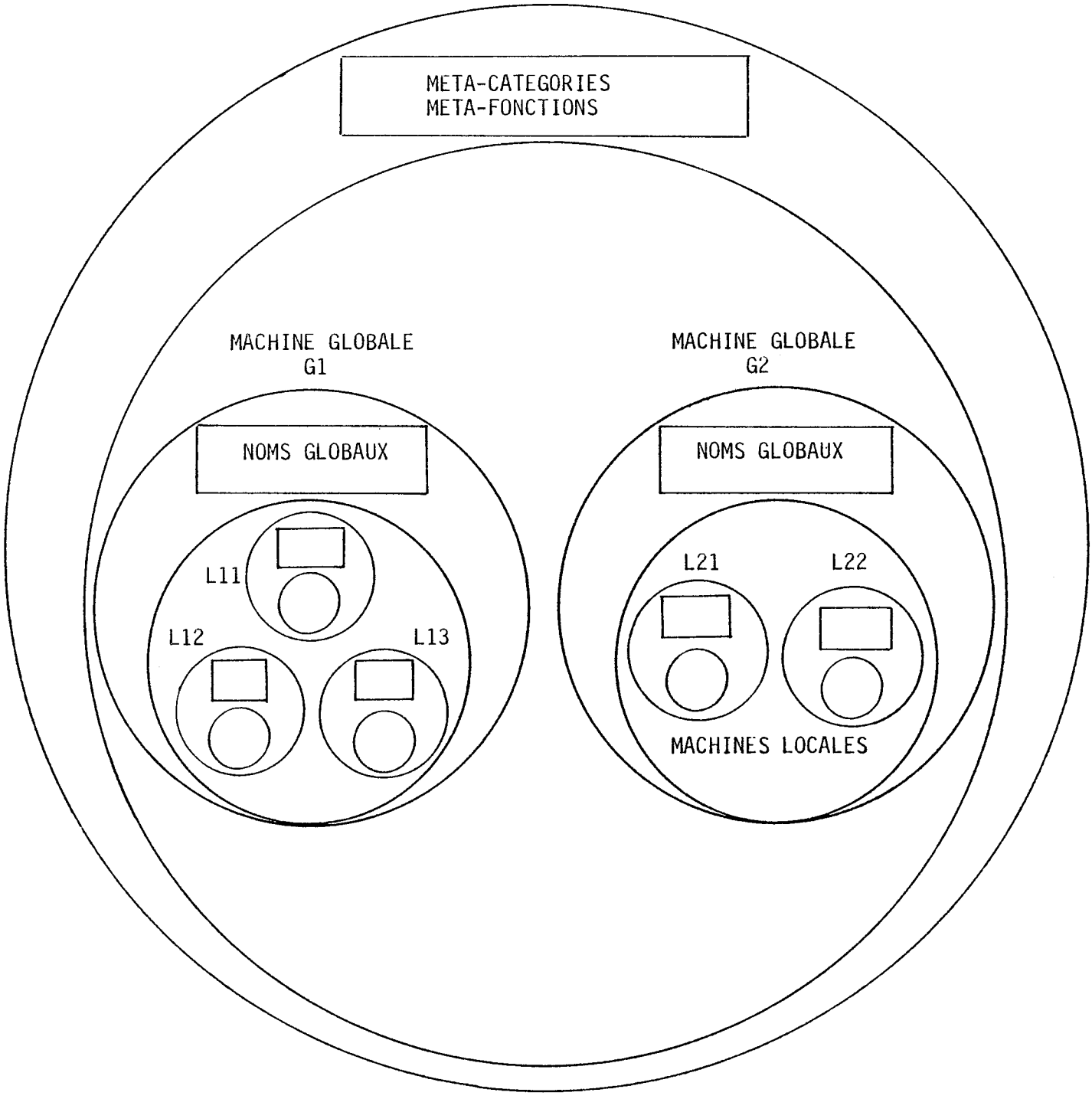


Figure 2.6 - La Machine "SYSTEME" MOGADOR

N.B. Par souci de simplification, les mécanismes de correspondance (§ 2.2.1.3) ne figurent pas dans la représentation des différentes machines.

2.3.1. Méta-Catégories

Il n'existe en fait dans MOGADOR que deux méta-catégories principales (*concrètes* toutes les deux) :

- CA ou ensemble des catégories abstraites
- CC ou ensemble des catégories concrètes.

Ces deux ensembles sont des ensembles de *noms*.

- L'ensemble CA comprend les noms : ENTIER, REEL, LOGIQUE, CHAINE, ainsi que tous les noms de catégories définies à partir de celles-ci (cf. § 2.2.2.1). On définit deux nouvelles catégories abstraites :
  - OS ensemble des objets simples :
    OS := ENTIER ∪ REEL ∪ LOGIQUE ∪ CHAINE
  - OC ensemble des objets composés (OS ⊂ OC)

(les identificateurs forment un sous-ensemble de OC : ID ⊂ OC).

- L'ensemble CC comprend les noms de toutes les catégories concrètes et en particulier de celles définies ci-après que l'on pourra assimiler à des META-CATEGORIES :

→ MACHINE : ensemble des machines MOGADOR, identifiées par exemple par 'S' pour la machine Système, G_i pour les machines Globales et L_{ij} pour les machines locales.

→ NI : ensemble des noms d'identificateurs de catégories concrètes (cf. § 2.2.2.4).

→ FMONO : ensemble des fonctions monovaluées.

→ FMULTI : ensemble des fonctions multivaluées.

→ PG : ensemble des programmes.

→ PS : ensemble des processus.

→ USAGER = ADM ∪ UTM : ensemble des usagers des machines, administrateurs (ADM) et utilisateurs (UTM).

2.3.2. Les Méta-Fonctions

Elles permettent de caractériser les liaisons entre méta-catégories. Nous allons les utiliser pour définir à la fois les espaces de noms, la correspondance avec les espaces d'éléments, et les opérations réalisables par une machine.

Les Méta-Fonctions sont des éléments des ensembles FMONO et FMULTI. On distingue deux types de méta-fonctions :

2.3.2.1. Méta-Fonctions de description

Ce sont des fonctions dont le graphe est stocké dans un espace de nom. Le tableau 2.2 donne la description des principales méta-fonctions de description.

2.3.2.2. Méta-Fonctions d'opérations ou opérateurs MOGADOR

Ce sont des fonctions calculées qui, pour un usager, assure la correspondance entre espace des noms et espace des éléments.

On distingue d'abord les *opérateurs primitifs* qui correspondent aux opérations sur les catégories et les fonctions décrites aux § 2.2.2.2 et 2.2.3.2 : voir table 2.3.

Les *opérateurs* proprement dits sont ceux qui s'appliquent à des ensembles d'objets simples ou composés. Ils correspondent à des opérations classiques de l'algèbre relationnelle [B3] et sont définis par la table 2.4. Nous y reviendrons au § 2.5.3.

Nature	Source	Méta-fonction	Type	Cible
Cat. Abstraites d'une machine	MACHINE	Camachine	multi	CA
Cat. Concrète d'une machine	MACHINE	Ccmachine	multi	CC
Fmono d'une machine	MACHINE	Fmonomachine	multi	FMONO
Fmulti		Fmultimachine	multi	FMULTI
Usagers d'une machine	MACHINE	usager	multi	USAGER
Identificateur d'une CC	CC	nomid	mono	NI
Source d'une fonction	FMONO ∪ FMULTI	source	mono	CA ∪ CC
Cible d'une fonction	FMONO ∪ FMULTI	cible	mono	CA ∪ CC
Inverse d'une fonction	FMONO ∪ FMULTI	inv	mono	FMONO ∪ FMULTI

Table 2.2 - META-FONCTIONS DE DESCRIPTION

Opération	Source	Méta-fonction	Type	Cible
Création	CC × ID	créer	mono	LOGIQUE
Suppression	CC × ID	supprimer	mono	LOGIQUE
Tester l'existence	CC × ID	test	mono	LOGIQUE
Enumération	CC	énumérer	multi	OC
Accéder	{ FMONO × OC FMULTI × OC	accéder mono accéder multi	mono multi }	OC
Lier Délier	{FMONO ∪ FMULTI} × OC × **P**(OC)	{ lier délier }	mono	LOGIQUE
Graphe d'une fonction	FMONO ∪ FMULTI	graphe	mono	**P**(OC)
N.B. **P**(E) dénote l'ensemble des parties de E.				

Table 2.3 - Opérateurs primitifs

Opération	Source	Méta-fonction	Type	Cible
arithmétiques	OBJETS SIMPLES {ENTIER×ENTIER} ∪ {REEL×REEL}	add + sous - mult * div /	mono	ENTIER ∪ REEL
somme produit moyenne	**P**(ENTIER) ∪ **P**(REEL)	somme prod moyenne	mono mono mono	ENTIER ∪ REEL
	OBJETS COMPOSES			
cardinalité	**P**(OC)	card	mono	ENTIER
éliminer redondance	**P**(OC) (pré-ensemble)	unique	mono	**P**(OC) ensemble
restriction	**P**(OC) × FILTRE	sélecter	mono	**P**(OC)
projection	**P**(OC) × **P**(FMONO)	projeter	mono	**P**(OC)
union intersection différence produit cartésien composition division	**P**(OC) × **P**(OC)	union (ou ∪) inter (ou ∩) diff (ou -) prodcart (ou ×) composer (ou *) diviser (ou /) (1)	mono	**P**(OC)
concaténation	**P**(OC) × **P**(OC) (pré-ensembles)	conc (ou +)	mono	**P**(OC) (pré-ensemble)

(1) - Il n'y a pas confusion avec les symboles de multiplication et de division, à cause du contexte.

Table 2.4 - Opérateurs

2.4. MOGADOR COMME MODELE CONCEPTUEL DE BASES DE DONNEES

2.4.1. Catégories et Fonctions

Pour modéliser un ensemble d'informations, les principaux outils qu'offre MOGADOR sont :

- les catégories abstraites (CA)
- les catégories concrètes (CC) avec leurs identificateurs (NI)
- les fonctions (FMONO, FMULTI).

Les catégories abstraites correspondent à des ensembles d'objets simples ou composés qui sont analogues aux ensembles de valeurs du modèle-entité-relation [B50] ou aux domaines du modèle de Codd. Ces objets jouent deux rôles principaux dans la modélisation :

- identificateurs de catégories concrètes (objets simples ou composés)
- éléments des graphes des fonctions (objets composés).

Les catégories concrètes et les fonctions constituent des ensembles d'éléments sur lesquels s'appliquent les opérations primitives ; elles permettent de caractériser les deux formes d'abstractions qui interviennent dans un processus de modélisation à savoir [B54] :

- le regroupement d'entités de même nature sous un nom générique : la généralisation
- le regroupement d'entités de natures différentes pour constituer de nouvelles entités : l'agrégation.

Ainsi, la généralisation de la notion d'individus peut donner naissance à la catégorie concrète PERSONNE comme ensemble d'individus.

Etant donné deux catégories concrètes, par exemple des PRODUITs et des USINEs, l'agrégation permet de caractériser une nouvelle entité constituée par l'association d'un produit et d'une usine. Ainsi, on a :

- une nouvelle catégorie concrète : FABRICATION (ensemble de couples (produit, usine))
- une fonction entre PRODUIT et USINE (mono ou multivaluée selon les cas)
- deux fonctions "triviales" respectivement entre FABRICATION et PRODUIT, et entre FABRICATION et USINE.

Nous allons illustrer un tel processus par un exemple avant de revenir au cas général.

Exemple :

Considérons les *catégories abstraites* suivantes :

- U# un ensemble de numéros d'usines
- P# un ensemble de numéros de référence de produits
- C# un ensemble de numéros de bons de commandes.

On définit les *catégories concrètes* suivantes :

- USINE identifiée par U# (généralisation : ensemble d'usines)
- PRODUIT identifiée par P# (généralisation : ensemble de produits)
- COMMANDE identifiée par U# × C# (généralisation : ensemble de commandes, mais en faisant l'hypothèse qu'une commande est gérée par une usine et que les usines peuvent utiliser la même numérotation pour les commandes)
- FABRICATION identifiée par U# × P# (agrégation : ensemble de produits fabriqués par une usine).

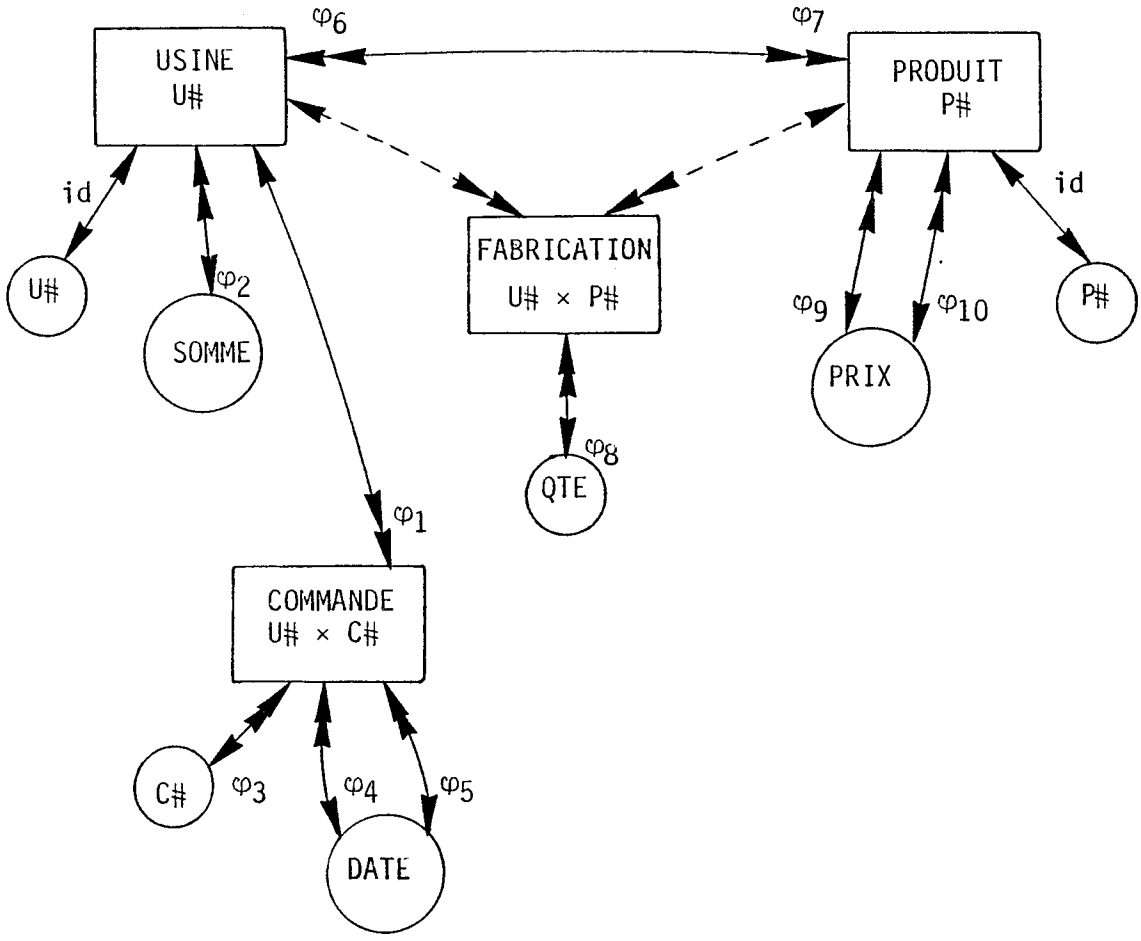
Considérons les autres catégories abstraites suivantes :

SOMME ensemble de valeurs réelles
QTE ensemble d'entiers
DATE ensemble de dates
PRIX ensemble de prix (valeurs réelles).

Nous allons définir des fonctions monovaluées à partir des faits élémentaires suivants :

- à une usine est allouée une certaine somme qui constitue son budget (fonction φ_2)
- une usine donnée fabrique par jour une certaine quantité d'un produit donné (φ_8)
- chaque produit a un prix de revient (φ_9) et un prix de vente (φ_{10})
- une commande a une date d'émission et une date de livraison (φ_4 et φ_5 respectivement).

Le processus de modélisation donne naissance à une description qui peut se visualiser graphiquement de la manière suivante (Figure 2.7) et se résumer à l'aide des tableaux 2.5, 2.6 et 2.7 :



Les symboles utilisés sont les suivants :

Catégorie Concrète et son identificateur

Catégorie Abstraite

φ

Fonction Monovaluée

Fonction Multivaluée

Fonction triviale

Figure 2.7 - Un exemple MOGADOR

Table 2.5 - Catégories Abstraites (CA)

Nom	Cat. Prédéfinie
U#	CHAINE
P#	CHAINE
C#	ENTIER
SOMME	REEL
QTE	ENTIER
DATE	ENTIER
PRIX	REEL

Table 2.6 - Catégories Concrètes (CC)

Nom	Identificateur
USINE	U#
PRODUIT	P#
FABRICATION	U# × P#
COMMANDE	U# × C#

Table 2.7 - Fonctions

Nom	Type	Source	Cible	Type inverse	Graphe
φ_1	multi	USINE	COMMANDE	mono	(U#, C#)
φ_2	mono	USINE	SOMME	multi	(U#, SOMME)
φ_3	mono	COMMANDE	C#	multi	(U#, C#)
φ_4	mono	COMMANDE	DATE	multi	(U#×C#, DATE)
φ_5	mono	COMMANDE	DATE	multi	(U#×C#, DATE)
φ_6	multi	PRODUIT	USINE	multi (φ_7)	(U#, P#)
φ_7	multi	USINE	PRODUIT	multi (φ_6)	(U#, P#)
φ_8	mono	FABRICATION	QTE	multi	(U#×P#, QTE)
φ_9	mono	PRODUIT	PRIX	multi	(P#, PRIX)
φ_{10}	mono	PRODUIT	PRIX	multi	(P#, PRIX)

Au niveau des graphes des fonctions, on peut remarquer que

1) graphe (φ_1) $\subseteq$ projection (graphe (φ_4), U#, C#)

2) graphe (φ_6) = graphe (φ_7) $\subseteq$ projection (graphe (φ_8), U#, P#).

Si nous nous plaçons dans le cas général d'une catégorie concrète C ayant pour identificateur I = I1 × I2 × ... × In, selon la composition de I par rapport à l'ensemble CC, dont fait partie C, nous avons les deux cas suivants :

1) *C représente une généralisation :*

- Il n'existe aucun sous-ensemble de I qui soit identificateur d'une autre catégorie concrète :

$$\nexists\ I' \subset I \text{ tel que } I' \in NI.$$

- Il existe un seul sous-ensemble I' de I qui est l'identificateur d'une autre catégorie concrète C' : il existe alors un lien hiérarchique entre C et C' (C est fils de C'). Dans ce cas, I contient le graphe de la fonction multivaluée qui va de C' vers C.

2) *C représente une agrégation :*

- Il existe alors dans I plusieurs sous-produits (au moins deux, soient I' et I") qui sont les identificateurs d'autres catégories concrètes (respectivement C' et C"). Dans ce cas, C' et C" sont reliées par une fonction dont le graphe est dans I.

2.4.2. Passage à une Vue Relationnelle

Les éléments des Catégories Concrètes et ceux des graphes des fonctions constituent des atomes sémantiques de répartition (cf. Chapitre 4), c'est-à-dire des unités d'informations indécomposables, pouvant être distribuées dans diverses bases de données.

Cependant, les catégories et les fonctions se placent au niveau conceptuel pour représenter la réalité perçue par notre esprit. Elles peuvent être utilisées pour concevoir une nouvelle base de données, mais aussi pour unifier différentes vues (hiérarchiques, réseaux, relationnelles), en supprimant certaines ambiguïtés sémantiques (cf. Chapitre 4).

A partir d'un ensemble de Catégories et de Fonctions MOGADOR, nous voulons construire un schéma relationnel, c'est-à-dire un ensemble de relations n-aires ne présentant pas de redondance ni d'inconvénients liés aux

opérations de mise à jour. En d'autres termes, cet ensemble ne doit contenir que des relations en troisième forme normale (3FN).

La construction à partir d'un schéma MOGADOR d'un schéma relationnel en 3FN constitue un problème non trivial qui dépasse le cadre des bases de données réparties. Cependant, en posant une série d'hypothèses simplificatrices, nous allons décrire un processus de transformations qui, partant d'un schéma MOGADOR quelconque, élimine les liaisons indésirables pour aboutir finalement à un ensemble de relations en 3FN (cf. Figures 2.2 et 2.8).

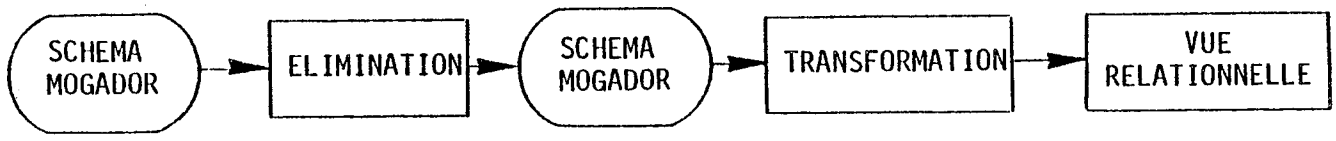


Figure 2.8 - Le passage d'un schéma MOGADOR à une Vue Relationnelle

Une vue relationnelle comprend la description :

- des domaines
- des relations avec pour chacune d'elles une clé primaire (ensemble d'attributs) et un ensemble d'attributs non clé.

Dire qu'il existe une relation fonctionnelle entre deux attributs A et B (éventuellement composés), c'est dire qu'à un élément a de A, il ne correspond qu'un et un seul élément b de B (on écrit A → B).

Si l'on appelle déterminant tout attribut (éventuellement composé) qui apparaît en partie gauche d'une relation fonctionnelle, la définition d'une relation en troisième forme normale est la suivante [G3] :
"Une relation R est en 3FN si chaque déterminant est une clé candidate".

L'idée de base concernant le passage de MOGADOR à une vue relationnelle est d'associer la notion de catégorie concrète à celle de relation et la notion d'identificateur à celle de clé primaire :

Considérons une catégorie concrète C, dont l'identificateur est $I = I1 \times I2 \times \ldots \times In$ avec les conditions suivantes :

$$\nexists\ J \subset I \text{ et } \nexists\ K \subset I \text{ tels que } J \text{ et } K \text{ soient dépendants}$$

c'est-à-dire : tous les composants de l'identificateur sont indépendants.

Dans ce cas, C en tant qu'ensemble d'objets composés (de n-uplets) peut être assimilée à une relation n-aire en 3FN :

$$R_C\ (\underline{I1,\ I2,\ \ldots,\ In})$$

Le problème ensuite est de considérer les fonctions et leurs graphes.

Pour une fonction f de C1 vers C2, le graphe étant un sous-ensemble du produit cartésien C1 × C2 peut être assimilé également à une relation n-aire, dont la forme dépend de deux paramètres :

- la nature de C1 et de C2 (concret, abstrait)
- la nature de f (mono ou multivaluée).

Le tableau suivant (2.8) donne les dix cas possibles de liaison entre catégories et fonctions MOGADOR.
Nous ne considérons que des catégories abstraites qui ne jouent pas le rôle d'identificateurs.

Table 2.8 - Fonctions entre catégories

Cas	Liaison	Graphe	Relation Fonctionnelle
1	C1 I — f → C2 J	I × J	I → J J → I
2	C1 I — f → A	I × A	I → A A → I
3	A — f → B	A × B	A → B B → A
4	C1 I — f → C2 J	I × J	I → J
5	C1 I — f → A	I × A	I → A
6	A — f → C1 I	A × I	A → I
7	A — f → B	A × B	A → B
8	C1 I — f → C2 J	I × J	/
9	C1 I — f → A	I × A	/
10	A — f → B	A × B	/

C
I

C représente un élément de CC, I de NI

A

Représente un élément de CA-NI.

- *Dans le cas 1*, la relation correspondante est soit $R_{C1}(\underline{I},J)$, soit $R_{C2}(\underline{J},I)$ à cause des relations fonctionnelles.
- *Dans le cas 2*, A constitue en fait une clé candidate pour la relation $R_{C1}(\underline{I},A)$.
- *Le cas 3* pose un problème car il correspond à des liaisons entre catégories abstraites, engendrant ainsi un circuit de relations fonctionnelles [B39]. Il faut donc l'éliminer en transformant par exemple A en catégorie concrète.
- *Dans le cas 4*, la relation $R_{C1}(\underline{I},J)$ est en troisième forme si $I \cap J = \emptyset$.
- *Le cas 5* correspond à une *fonction de type attribut* et ne pose pas de problème, la relation associée étant $R_{C1}(\underline{I},A)$.
- *Le cas 6* pose le problème d'une relation fonctionnelle ayant pour partie gauche une catégorie abstraite. On peut l'éliminer en transformant A en une catégorie concrète pour se ramener au cas 4.
- *Le cas 7* est analogue au cas 6 mais renvoie au cas 5.
- *Le cas 8* correspond à une fonction multivaluée entre deux catégories concrètes (généralisation). Si elle n'existe pas déjà, on peut générer une catégorie concrète C3 ayant pour identificateur $I \times J$ ou directement une relation $R_{C3}(\underline{I,J})$.
- *Les cas 9 et 10* peuvent également se résoudre en générant de nouvelles catégories concrètes (table 2.9).

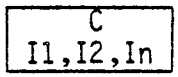
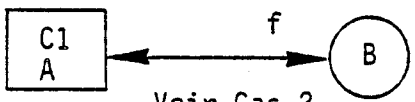
Cas	Représentation MOGADOR	Transformation	Conditions
0 Terminal		$R_C(\underline{I1,I2,...,In})$	$J \subset I$ et $\nexists K \subset I \nexists f$ tels que cible(f) = J, source (f) = K
1 Terminal		$R_{C1}(\underline{I},J)$ ou $R_{C2}(\underline{J},I)$	C1 et C2 comme au cas 0
2 Terminal		$R_{C1}(\underline{I},A)$ (A est clé candidate)	$(A \notin I) \wedge$ (C1 en cas 0) $\wedge$ $(\exists f' \in FMONO . f'=\underline{inv}\ f \wedge source(f') = A)$
3		 Voir Cas 2	Après transformation, $\forall f' \in F$, cible(f') = B faire cible(f') $\neq$ B et cible(f') $\leftarrow$ A
4 Terminal		$R_{C1}(\underline{I},J)$ $R_{C2}(J)$	C1 et C2 en cas 0 et $I \cap J = \emptyset$

Table 2.9 (début) - Transformations et passage à la vue relationnelle

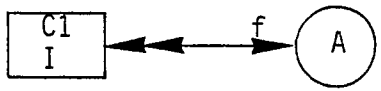
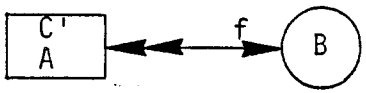
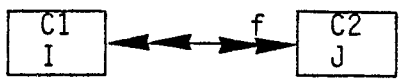
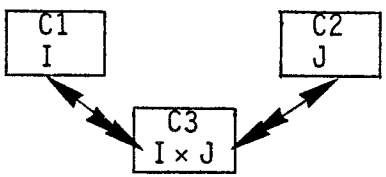
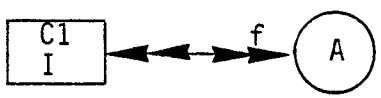
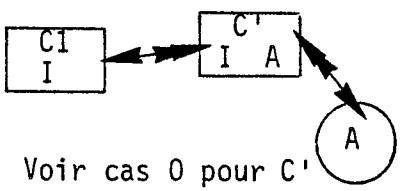
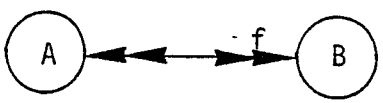
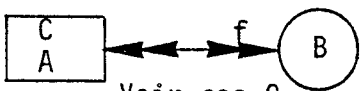
Cas	Représentation MOGADOR	Transformation	Conditions
5 Terminal		$R_{C1}(\underline{I},A)$ (attribut non clé)	C1 en cas 0
6		 Voir Cas 4	C1 en cas 0 et $\forall f' \in F$ tel que cible(f') = C1, faire cible(f') ← C'
7		 Voir Cas 5	Après transformation : $\forall f' \in F$ tel que cible(f') = B faire cible(f') ← B' et cible(f') ← C'
8		 Voir cas 0 pour C3	C1 et C2 en cas 0
9		 Voir cas 0 pour C'	C1 comme au cas 0
10		 Voir cas 9	/

Table 2.9 (fin) - Transformations et passage à la vue relationnelle

Ces transformations et leurs conditions sont résumées dans le tableau 2.9.

De plus, elles ne doivent pas être faites dans n'importe quel ordre, mais en suivant l'ordre donné par le diagramme de la Figure 2.9.

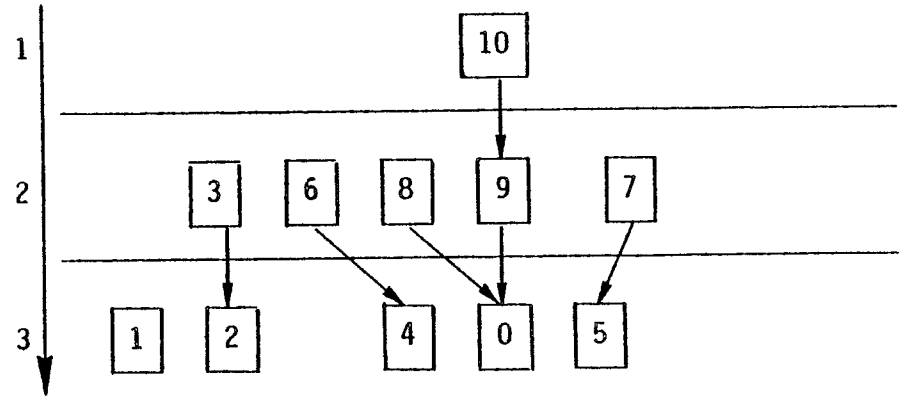


Figure 2.9 - Ordre dans lequel s'appliquent les transformations

On arrive ainsi à un ensemble $\mathcal{E}$ = (CC, CA, NI, FMONO, FMULTI) tel que :

$$\forall f \in FMULTI : (\underline{inv}\ f \in FMONO) \wedge (cible(f) \in CC)$$

Autrement dit :

Toute fonction multivaluée est telle que son inverse soit monovaluée et sa cible une catégorie concrète.

L'ensemble $\mathcal{E}$ ayant une telle propriété sera dit *normalisé* par analogie avec la normalisation des relations n-aires.

Le passage à un ensemble de relations n-aires en 3FN se fera alors de la manière suivante :

A chaque catégorie concrète, nous associons une relation n-aire :

- Soit C une catégorie concrète et I son identificateur. Notons i un élément quelconque de I.

Dans le cas général, I est un produit cartésien de catégories abstraites I1 × I2 × ... Ip.

- Soit F_C l'ensemble des fonctions monovaluées ayant pour source C :

$$F_C = \{f_j \mid f_j \in FMONO \wedge source(f_j) = C\}$$

- Soit A l'ensemble des catégories abstraites, cibles des fonctions de F_C :

$$A = \{A_j \mid (A_j \in CA) \wedge (\forall f_j \in F_C : A_j = cible(f_j))\}.$$

- On note C(I) la relation associée à C (cas 0 du tableau 2.9).
- On note $R_j(I,A_j)$ la relation associée à une $f_j \in F_C$.

La relation n-aire notée C^* qu'il est possible de construire à partir de C est alors :

$$C^*(\underline{I},A_1,A_2,\ldots,A_n) = C(I) * R_1(I,A_1) * \ldots * R_n(I,A_n)$$

C^* est donc l'ensemble des objets composés $(i, a_1, a_2,\ldots, a_n)$ comprenant chacun un identificateur i et une série d'objets a_j chacun lié fonctionnellement à i.

$$C^* = \{(i,f_1(i),f_2(i),\ldots,f_n(i)) \mid (i \in C) \wedge (f_j(i) \in A_j) \wedge (f_j \in FMONO)\}.$$

Avec cette approche, nous avons en fait la correspondance suivante entre les notions MOGADOR et le vocabulaire du modèle relationnel :

MOGADOR	Modèle Relationnel
Catégorie Abstraite (CA)	Domaine
Catégorie Concrète (CC)	Relation
Identificateur (NI)	Clé primaire
Fonction Monovaluée CC → CA	Attribut
Objet composé	n-uplet

Table 2.10 - Equivalence de Vocabulaire

N.B. Cette correspondance n'est valable qu'après obtention d'un ensemble $\mathcal{E}$ = {CC, CA, NI, FMONO, FMULTI} normalisé.

Avec l'exemple du § 2.4.1, illustré Figure 2.8, et en remplaçant φ_6 et φ_7 par les deux fonctions triviales, nous avons la vue relationnelle suivante (ensemble en 3FN) :

USINE* (U#, BUDGET)
PRODUIT* (P#, PRIVEN, PRIREV)
FABRICATION* (U#, P#, QTE)
COMMANDE* (U#, C#, DATEC, DATEL)

Nous avons fait volontairement apparaître les attributs des relations et non les domaines. Pour être complète, la description relationnelle doit contenir la liste des domaines et la correspondance attribut-domaine (cf. Table 2.11).

Fonction	Attribut	Domaine
φ_2	BUDGET	SOMME
φ_9	PRIVEN	PRIX
φ_{10}	PRIREV	PRIX
φ_8	QTE	QTE
φ_4	DATEC	DATE
φ_5	DATEL	DATE

Table 2.11 - Equivalences Attributs-Domaines

On trouvera au Chapitre 3 d'autres exemples de passage MOGADOR - Vue Relationnelle. Remarquons à ce niveau que la vue relationnelle obtenue à partir d'un schéma MOGADOR n'est pas unique.

2.5. MACHINE RELATIONNELLE

Nous venons de voir comment les outils conceptuels de MOGADOR permettaient, dans le cas général, de faciliter le passage à une vue relationnelle d'un ensemble d'informations. Nous appliquerons ces outils aussi bien au niveau local pour concevoir la vue locale qu'au niveau global pour la conception de la vue globale (Chapitres 3 et 4).

Dans ce qui suit, nous allons appliquer les outils MOGADOR pour définir de manière plus précise l'élément de base de l'architecture fonctionnelle du SGBDR que nous proposons, à savoir la machine relationnelle (Figure 2.10) [R7].

Une machine relationnelle comprend :

- un espace de noms, à savoir la vue relationnelle
- un espace d'éléments
- des mécanismes de correspondance entre noms et éléments.

Elle fournit à l'administrateur :

- un langage de description de vue relationnelle (fonction description).

Elle fournit à l'utilisateur :

- un langage de manipulation de vue relationnelle (fonction manipulation).

Nous regroupons ces deux langages sous le nom générique LA.DO.RE (LAngage pour DOnnées REparties ou RElationnelles). Sa syntaxe complète, telle qu'elle est fournie à l'utilisateur du SGBDR, est donnée en annexe.

Nous distinguons deux types de machines :

1) les *machines locales* (ML) construites autour d'un ensemble d'éléments stockés par un SGBD ou tout autre moyen équivalent, capable d'effectuer des opérations d'accès et de manipulation.
Nous dirons que ces machines fournissent des vues relationnelles d'ordre 1 ou vues locales (VL) (cf. Chapitre 3).

2) les *machines globales* (MG) construites autour de plusieurs autres machines (locales ou globales).

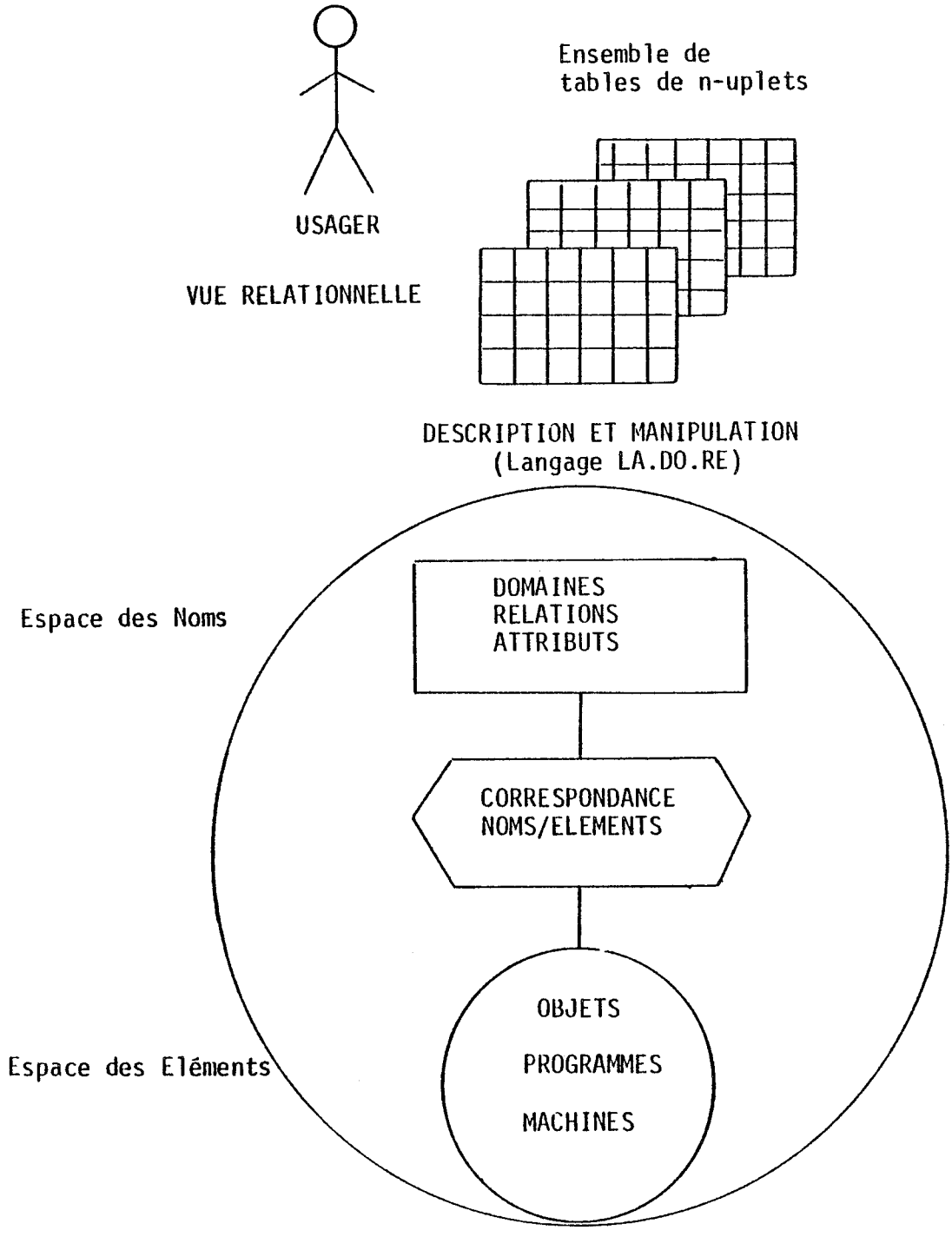


Figure 2.10 - Une machine logique relationnelle

Ces machines fournissent des vues relationnelles d'ordre n (n > 1). Une vue d'ordre n est construite à partir de k vues d'ordre (n-1).

Un nom global d'une vue d'ordre n a donc en général plusieurs synonymes d'ordres (n-1), tandis qu'un élément global d'ordre n est constuit à partir d'un ou plusieurs éléments d'ordre n-1.

Ce processus est décrit au Chapitre 4 en prenant n=2 pour définir une Vue Globale de plusieurs Vues Locales et la machine globale correspondante.

2.5.1. Description de Vue Relationnelle

La création d'une machine relationnelle passe conceptuellement par la création d'un élément de la méta-catégorie MACHINE.

La création de l'espace des noms correspond en fait à la création d'un ensemble de noms appartenant aux méta-catégories CA, CC, NI, FMONO, puis à relier entre eux ces noms par des méta-fonctions de description (cf. § 2.3).

Ainsi la fonction description se ramène-t-elle conceptuellement à des opérations de créations d'éléments et des opérations de liaisons (lier) pour constituer un ensemble d'informations. Il y a bien sûr à ce niveau une très forte analogie entre les éléments qui constituent le schéma de la base de données (des noms MOGADOR) et les éléments qui constituent la base de données elle-même (objets simples et composés MOGADOR).

Grâce au tableau 2.11, ces opérations conceptuelles se ramènent à la construction d'une vue relationnelle comprenant :

- une partie identification de la vue que l'on assimilera au nom de la machine
- une partie identification des utilisateurs et de leurs droits
- la description d'un ensemble de domaines

avec, pour chacun d'eux, un nom et la catégorie prédéfinie (ENTIER, REEL, LOGIQUE, CHAINE) correspondante, éventuellement restreinte à un sous-ensemble.

- la description des relations comprenant pour chaque relation :

- un nom
- la liste des attributs clés
- la liste des attributs non clés
- la correspondance entre un attribut et son domaine associé
- les groupes d'attributs qui sont en fait clés d'autres relations, de manière à expliciter les liaisons entre relations (Clés Etrangères).

A titre d'exemple, la relation COMMANDE*(U#, C#, DATEC, DATEL) sera décrite par :

```
relation COMMANDE
    | debclé U# : C#
    |        C# : C#
    | finclé
    | DATEC : DATE
    | DATEL : DATE
    | (U#) cléde USINE
```

Avec par exemple :

```
dom DATE := CHAINE (6).
```

Une autre partie de la fonction Description concerne la définition de la correspondance Noms-Eléments. Cependant, cette définition fait intervenir le langage de manipulation ; aussi y revenons-nous au § 2.5.4.

2.5.2. Opérations sur une Vue Relationnelle

Sur une relation quelconque R, on définit *quatre opérations principales*, à savoir :

1. *OBTENIR* : former l'ensemble des n-uplets de R : obtenir (R).
2. *INSERER* : un n-uplet dans R (ce n-uplet doit nécessairement contenir la valeur de la clé) : insérer (R,t).
3. *SUPPRIMER* : un n-uplet de R : supprimer (R,t).
4. *MODIFIER* : un n-uplet de R (modifier seulement les valeurs qui n'appartiennent pas à la clé) : modifier (R,t,t').

Puisque dans notre démarche une relation R est construite à partir d'une modélisation en termes de catégories et de fonctions, nous pouvons définir les quatre opérations ci-dessus à partir des opérations primitives sur les catégories et les fonctions sous-jacentes.

Pour cela, nous noterons :

1) COMP (X,X_k) $\forall k \in \{1,n\}$ l'opération de composition généralisée

$$\text{COMP}\ (X,X_k)_{\forall k\in\{1,n\}} = X * X_1 * X_2 * \ldots * X_n$$ (cf. § 2.5.3)

2) $OU(L_k)_{\forall k\in\{1,n\}}$ l'opération d'union généralisée de valeurs logiques

$$OU(L_k)_{\forall k\in\{1,n\}} = L_1 \vee L_2 \vee L_n.$$

3) $ET(L_k)_{\forall k\in\{1,n\}}$ l'opération d'intersection généralisée de valeurs logiques

$$ET_{\forall k\in\{1,n\}} = L_1 \wedge L_2 \wedge \ldots \wedge L_k.$$

Nous considérons une relation R construite sur les domaines $I,A_1,A_2,\ldots,A_n$ qui se décompose en $R(I,A_1,A_2,\ldots,A_n) = R(I) * R_1(I,A_1) * R_2(I,A_2) * \ldots * R_n(I,A_n)$ avec les équivalences suivantes :

- à R(I) correspond une catégorie concrète R identifiée par I
- à chaque $R_k(I,A_k)$ correspond une fonction monovaluée f_k telle que $source(f_k) = R$ et $cible(f_k) = A_k$.

La table 2.12 donne alors la définition des opérations n-aires, en fonction des primitives MOGADOR.

Opération n-aire	Notation	Primitives MOGADOR
Obtenir l'ensemble des n-uplets de R. Résultat : ensemble d'objets composés.	$R^{(1)}$ ou obtenir (R)	COMP $\forall k \in \{1,n\}$ (énumérer (R), graphe (f_k))
Insérer le n-uplet $<i,a_1,\ldots,a_n>$ dans R. Résultat : valeur LOGIQUE.	insérer (R,$<i,a_1,\ldots,a_n>$)	ET $\forall k \in \{1,n\}$ (créer (R,i), lier (f_k,i,a_k))
Supprimer le n-uplet $i,a_1,a_2,\ldots,a_n$ de R. Résultat : valeur LOGIQUE.	supprimer (R$<i,a_1,\ldots,a_n>$)	ET $\forall k \in \{1,n\}$ (supprimer (R,i), délier (f_k,i,a_k))
Modifier un n-uplet de R : i identifie le n-uplet $<a_1,a_2,\ldots,a_n>$ sont les nouvelles valeurs. Résultat : valeur LOGIQUE.	modifier (R,i,$<a_1,a_2,..a_n>$)	ET $\forall k \in \{1,n\}$ (délier (f_k,i,f_k(i)) ^ lier (f_k,i,a_k))

(1) - On assimile le nom de la relation à son ensemble de n-uplets.

Table 2.12 - Définition des opérations n-aires à partir des primitives MOGADOR

2.5.3. L'Exploitation d'une Vue Relationnelle

La manipulation d'un ensemble de relations se fait au moyen d'un langage relationnel. De très nombreux travaux ont été consacrés à l'étude et au développement de ce type de langage (cf. § 1.1.3.2.1) et notre but ici n'est pas de définir un nouveau langage.

Nous allons utiliser un langage fondé sur l'algèbre relationnelle qui, comme l'a montré Codd [B19], fournit un ensemble d'opérateurs s'appliquant sur des relations normalisées et assure la complétude de la puissance d'expression.

Cet ensemble d'opérateurs comporte deux parties :

- les *opérateurs ensemblistes* classiques (union, intersection, différence et produit cartésien)
- les *opérateurs relationnels* proprement dits (projection, sélection, composition et division).

Toutes ces opérations sont nécessaires pour exprimer des requêtes d'interrogation, tandis que seules l'union et la différence sont utiles pour les opérations de modification :

- l'*insertion* est vue comme l'*union* de la relation avec le nouvel n-uplet
- la *suppression* est vue comme la *différence* entre la relation et le n-uplet à enlever
- la *modification* de valeurs est vue comme une suppression suivie d'une insertion.

Nous allons définir précisément tous ces opérateurs en donnant pour chacun deux types de représentation, à savoir une représentation mathématique classique (R ∪ S par exemple) et une représentation graphique qui nous permettra de mieux visualiser les requêtes de manipulation [B3, B22] (cf. § 4.3).

En fait, aucune de ces deux formes ne sera adoptée pour la forme externe du langage de manipulation tel qu'il sera fourni aux utilisateurs du système de Base de Données Réparties. On préfèrera alors une notation

fonctionnelle comme celle utilisée dans [B44] (cf. Annexe 1).

Par exemple : union (R,S).

2.5.3.1. Opérateurs ensemblistes (table 2.13)

On considère deux relations R(X) et S(Y) où X et Y dénotent respectivement l'ensemble des attributs de R et S. En général, les n-uplets d'une relation seront notés avec des minuscules ($r \in R$ par exemple).

$r \cap s$ dénote la concaténation des n-uplets r et s [B19].

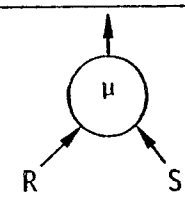
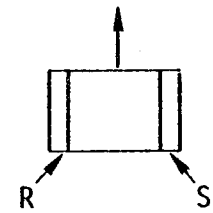
Opération	Notation algébrique	Définition du Résultat	Représentation graphique
union intersection différence	$R \mu S$ $\mu \in \{\cup, \cap, -\}$	$\{r \mid r \in R\} \mu \{s \mid s \in S\}$	
produit cartésien	$R \times S$	$\{r \cap s \mid r \in R \wedge s \in S\}$	

Table 2.13 - Opérateurs ensemblistes

2.5.3.2. Opérateurs relationnels (table 2.14)

- *Projection* : C'est une opération unaire notée $[\gamma]R$ où γ dénote un sous-ensemble des attributs de R ($\gamma \subseteq X$).
- *Restriction ou Sélection* : C'est une opération unaire notée $R : \varphi$ où φ dénote une expression conditionnelle définie à partir d'expressions atomiques [B3] :
  - $\langle X_i \ \theta \ X_j \rangle$ où X_i et X_j désignent des attributs de R et θ l'un des opérateurs $=, \neg =, <, \leq, >, \geq$
  - $\langle X_i \ \{\in, \notin\} \ \{(x_1, \ldots, x_n), [n_1, n_2[\} \rangle$

- $(x_1, x_2, \ldots, x_n)$ désigne un ensemble de valeurs
- $[n_1, n_2[$ désigne un invervalle dans $\mathbb{N}^+$ à savoir $n_1 \leq k < n_2$.

φ est définie de la façon suivante :

(1) toute expression atomique est une expression conditionnelle

(2) si φ_1 et φ_2 sont des expressions conditionnelles, alors $\neg\varphi_1$, $\varphi_1 \wedge \varphi_2$ et $\varphi_1 \vee \varphi_2$ le sont également.

- *Composition* : C'est une opération binaire définie par Codd [B17] (opération "join") et qui revêt plusieurs formes [B18, B38, B72, B73]. La composition de deux relations R(X) et S(Y) ne peut se faire que si elles possèdent chacune un attribut (respectivement $A \in X$ et $B \in Y$) construit sur un même domaine (de tels attributs seront dits compatibles). Nous ne nous intéressons ici qu'à l'égalité des valeurs entre les ensembles [A]R et [B]S. Cependant, le problème se pose s'il existe des valeurs appartenant à l'un de ces deux ensembles et pas à l'autre. Cela conduit alors à deux formes de composition :

- *Composition naturelle ou "intersection join"* notée $R[A \cap B]S$ où n'apparaissent dans le résultat que les éléments communs aux deux relations.

- *Composition naturelle généralisée ou "union join"* notée $R[A \cup B]S$ où apparaissent dans le résultat des n-uplets dans certains champs, la valeur définie que nous noterons "#" [B38][R17]. Cette valeur indéfinie sera interprétée comme "valeur non connue au moment présent" et considérée comme une valeur à part entière (# = # est vraie, # = x est faux).

On associe à cette opération deux opérations de composition dissymétriques notées respectivement $R[A \leftarrow B]S$ et $R[A \rightarrow B]S$ et telles que : $R[A \cup B]S = R[A \leftarrow B]S \cup R[A \rightarrow B]S$.

- *Division* : C'est une opération binaire qui ne peut se faire que si les deux relations ont un domaine en commun par l'intermédiaire de deux attributs.

Etant donné R(A,X) et S(B,Y) où A et B sont compatibles, la division de R par S sera notée R[A/B]S.
Le résultat est une relation définie sur X (pouvant être un groupe d'attributs). Sa définition comme celle des opérations ci-dessus est donnée par le tableau suivant (2.14).

2.5.3.3. Opérateurs de Calculs

Ils permettent d'effectuer des calculs sur des ensembles d'éléments. Nous retiendrons :

- *cardinalité* : pour compter le nombre d'éléments d'un ensemble
- *moyenne* : pour calculer la moyenne d'un ensemble d'entiers ou de réels
- *somme et produit* d'un ensemble de nombres
- *maximum et minimum*.

Le tableau 2.15 résume l'ensemble des opérateurs exécutables par une machine relationnelle, en donnant cette fois un formalisme voisin de celui adopté pour la forme externe du langage de manipulation LADORE (cf. Annexe).

Ce tableau est divisé en trois parties qui correspondent respectivement aux opérateurs ensemblistes (1), aux opérateurs de l'algèbre relationnelle (2) et aux opérateurs de calculs (3).

Pour chaque opération, la première colonne donne son nom et la notation habituellement utilisée. La seconde colonne indique le formalisme utilisé ici qui est de type fonctionnel en ce sens que les opérandes peuvent être soit le nom d'une relation de la vue, soit le résultat d'une autre opération.

A chaque opérande est associé un ensemble d'attributs (noté $A = \{A_1, A_2, \ldots, A_n\}$) et la colonne "résultat" montre comment se modifie cet ensemble pour former les attributs de la relation résultat.

La fonction "degré" appliquée à une relation fournit le nombre d'attributs de cette relation.

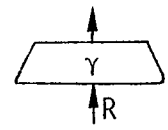
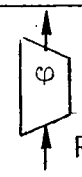
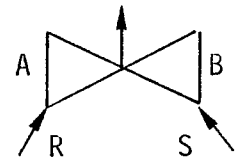
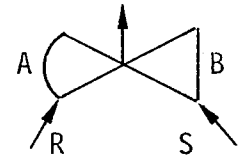
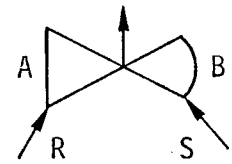
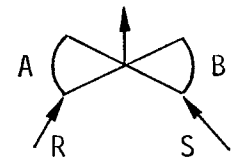
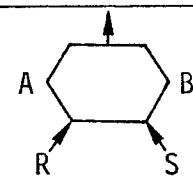
Opération	Notation	Définition	Représentation graphique
Projection	$[\gamma]R$	$\{[\gamma]r \mid r \in R\}$	
Sélection ou Restriction	$R : \varphi$	$\{r \mid r \in R \wedge \varphi_r\}$ φ_r : expression conditionnelle appliquée à r	
Composition naturelle ("intersection-join")	$R[A \cap B]S$ ou $R * S$	$\{[\bar{B}](r \cap s) \mid r \in R \wedge s \in S \wedge ([A]r = [B]s)\}$ où $\bar{B} = \{X,Y\}-\{B\}$	
Composition Naturelle Généralisée + ("union-join")	$R[A \leftarrow B]S$ ou $R \overset{\leftarrow}{*} S$	$(R[A \cap B]S) \cup ((R[A \cap A]([A]R-[B]S)) \times \{\#\})$	
	$R[A \rightarrow B]S$ ou $R \overset{\rightarrow}{*} S$	$(R[A \cap B]S \cup (\{\#\} \times (S[B \cap B]([B]S-[A]R)))$	
	$R[A \cup B]S$ ou $R \overset{+}{*} S$	$R[A \leftarrow B]S \cup R[A \rightarrow B]S$	
Division	$R[A/B]S$	$\{[X]r \mid (r \in R) \wedge [B]S \subseteq g_R([X]r)\}$ avec $g_R(x) = \{a \mid (a,x) \in R\}$	

Table 2.14 - Opérateurs relationnels

Tableau 2.10 - Opérateurs sur vue relationnelle (Langage LA.DO.RE)

Opération	Formalisme	Opérandes	Résultat	Remarques
1				
- union : R1 ∪ R2	union (R1, R2)	R1(A), R2(A)	R(A)	degré(R1)=degré(R2)=degré(R)
- intersection : R1 ∩ R2	inter (R1, R2)	R1(A), R2(A)	R(A)	"
- différence : R1 - R2	diff (R1, R2)	R1(A), R2(A)	R(A)	"
- produit cartésien : R1 × R2	prodcart (R1, R2)	R1(A), R2(B)	R(A,B)	degré(R) = degré(R1)+degré(R2)
2				
- projection [γ]R	projeter (R,<γ>)	R(A), γ ⊆ A	R(γ)	γ est la liste des attributs à projeter
- restriction R : φ	sélecter (R, φ)	R(A), φ	R(A)	φ est un filtre sur les attributs
- composition R1[A_i θ B_i]R2 ou R1 * R2	composer (R1,R2,A_i θ B_i)	R1(A), R2(B) $A_i \in A$ $B_i \in B$ θ = ∪ ou ∩	R(A,B-B_i)	il s'agit du "natural join"
- division R1[A/B_i]R2 ou R1/R2	diviser (R1,R2, <A_i,B_i>)	R1(A), R2(B)	R(A-A_i)	En général R1 est binaire et R2 unaire
3				
- somme	som (R)			R est une relation unaire composée d'objets simples ∈ ENTIER
- produit	prod (R)			
- maximum, minimum	max (R), min (R)	R(E)	x ∈ ENTIER	
- moyenne	moy (R)			
- cardinalité	card (R)			

2.5.3.4. Transactions et Requêtes

Un usager d'une machine dialogue avec cette machine à l'aide d'un organe d'entrée et d'un organe de sortie. L'unité d'interaction entre un usager et une machine est la *transaction* qui comprend plusieurs *requêtes* de deux types : interrogation ou modification (Figure 2.11).

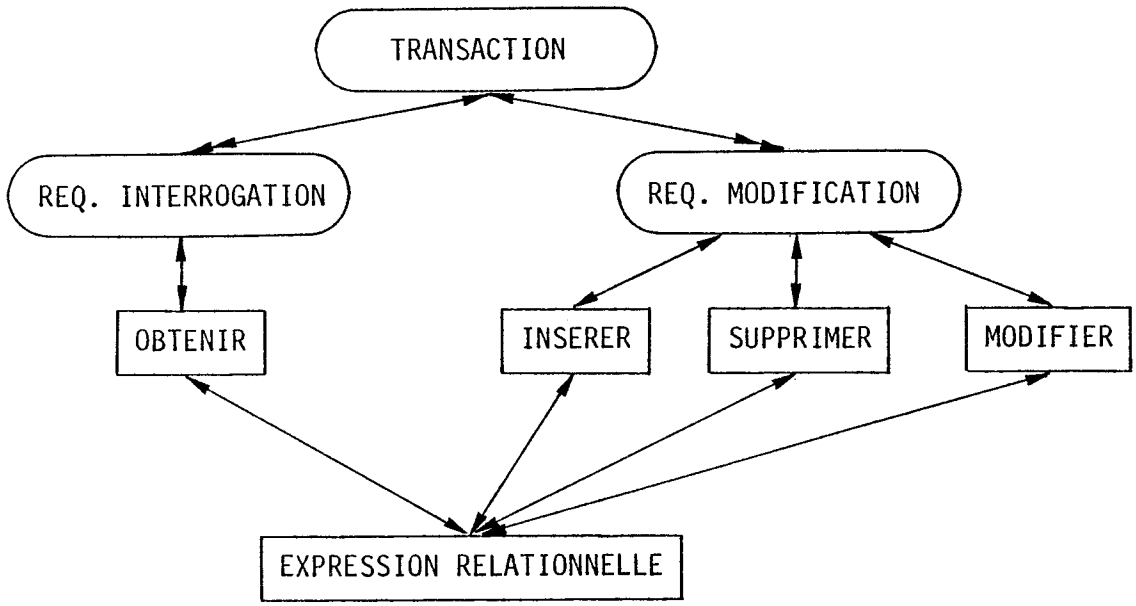


Figure 2.11

- Une *requête d'interrogation* correspond à une opération OBTENIR portant sur une expression relationnelle ε définie par les règles suivantes :
  - R1 : toute citation du nom d'une relation est une expression relationnelle.
  - R2 : Si ε_1 est une expression relationnelle, alors $[\gamma]\varepsilon_1$ et $(\varepsilon_1 : \varphi)$ le sont également.
  - R3 : Si ε_1 et ε_2 sont des expressions relationnelles, alors :
    - $\varepsilon_1 \ \theta \ \varepsilon_2$ (avec $\theta \in \{\cup, \cap, -, \times\}$)
    - ε_1 A μ B ε_2 (avec $\mu \in \{\cup, \cap, \leftarrow, \rightarrow, /\}$ et A et B deux attributs)

    le sont également.

- Une *requête de modification* correspond à l'une des opérations INSERER, SUPPRIMER, MODIFIER, portant elles aussi sur une expression relationnelle.

Le fait pour un utilisateur de créer une transaction sur l'organe d'entrée de la machine correspond en fait à la création d'un processus résultat de l'exécution d'un programme.

Ce programme correspond à la transformation des opérations demandées en opérations effectivement exécutables par la machine.

2.5.4. Correspondance Noms-Eléments : Règles d'évolution

Cette partie a pour but de permettre à l'administrateur de décrire la sémantique de manipulation associée à chaque relation de la vue, au moyen de *règles d'évolution*.

En effet, par le processus de modélisation MOGADOR, on décrit des catégories et des fonctions pour lesquelles il faut indiquer la manière dont se réalisent les opérations primitives. Ainsi, à une catégorie concrète sont associées quatre règles (création, suppression, test, énumération) et à une fonction quatre autres (accès, lier, délier, graphe).

Après passage à une vue relationnelle, on aura pour chaque relation quatre règles d'évolution (obtenir, insérer, supprimer, modifier) (cf. Table 2.12).

Chaque règle est exprimée au moyen d'un programme défini par les informations suivantes :

- un *nom*
- des *entrées* : noms des catégories auxquelles appartiennent les objets fournis en paramètres au programme
- des *sorties* : noms des catégories auxquelles appartiennent les objets fournis en sortie
- son *corps* : ensemble d'opérations :
  selon le type de machine où se situe ce programme, les opérations s'adressent :

- soit à un SGBD s'il s'agit d'une machine locale (cf. § 3.3)
- soit à un ensemble de machines s'il s'agit d'une machine globale (cf. § 4.3).

Nous verrons en fait que ces règles constituent une facilité offerte à l'administrateur local ou global, pour assurer une cohérence de sa base ; il peut en particulier interdire telle opération ou indiquer qu'elle est impossible.
Cependant, il ne faut pas que l'écriture "manuelle" de telles règles soit systématique car elle devient fastidieuse, c'est pourquoi nous aurons :

- au *niveau local*, l'écriture *explicite* des règles au moyen de *programmes locaux* (cf. § 3.3.2)
- au *niveau global*, la fourniture *implicite* par une machine de *règles standard* au moyen du langage de description (cf. cas standard de répartition § 4.2.6).

2.6. CONCLUSIONS

Nous venons de présenter en détails MOGADOR qui doit être vu comme un ensemble d'outils conceptuels utilisables :

1) pour concevoir une base de données relationnelle répartie
2) pour définir l'architecture fonctionnelle d'un SGBDR comme un réseau de machines logiques relationnelles.

Dans ce qui suit, nous montrons comment ces outils sont appliqués aux niveaux local (Chapitre 3) et global (Chapitre 4) en adoptant pour chacun d'eux la présentation suivante :

1. Description du problème à résoudre.
2. Apport de MOGADOR à la conception d'une vue relationnelle (locale ou globale) d'un ensemble d'informations.
3. Conception de la machine logique correspondante : description de ses composants et fonctionnement.

Chapitre 3

Vues Relationnelles de Bases Hétérogènes : MOGADOR au niveau local

La multitude qui ne se réduit pas à l'unité est confusion ; l'unité qui ne dépend pas de la multitude est tyrannie.

B. PASCAL

Sommaire

3.1. LE MODELE RELATIONNEL COMME MODELE COMMUN DE DESCRIPTION

Au Chapitre 2, ont été définis les principaux concepts de MOGADOR avec un bref exemple de modélisation. Le problème que nous abordons maintenant est le suivant : comment, en utilisant MOGADOR, reinterpréter le schéma d'une base de données hiérarchique ou réseau afin d'en déduire une vue relationnelle ?

D'autre part, comment réaliser cette vue relationnelle grâce à une machine locale ? (Figure 3.1).

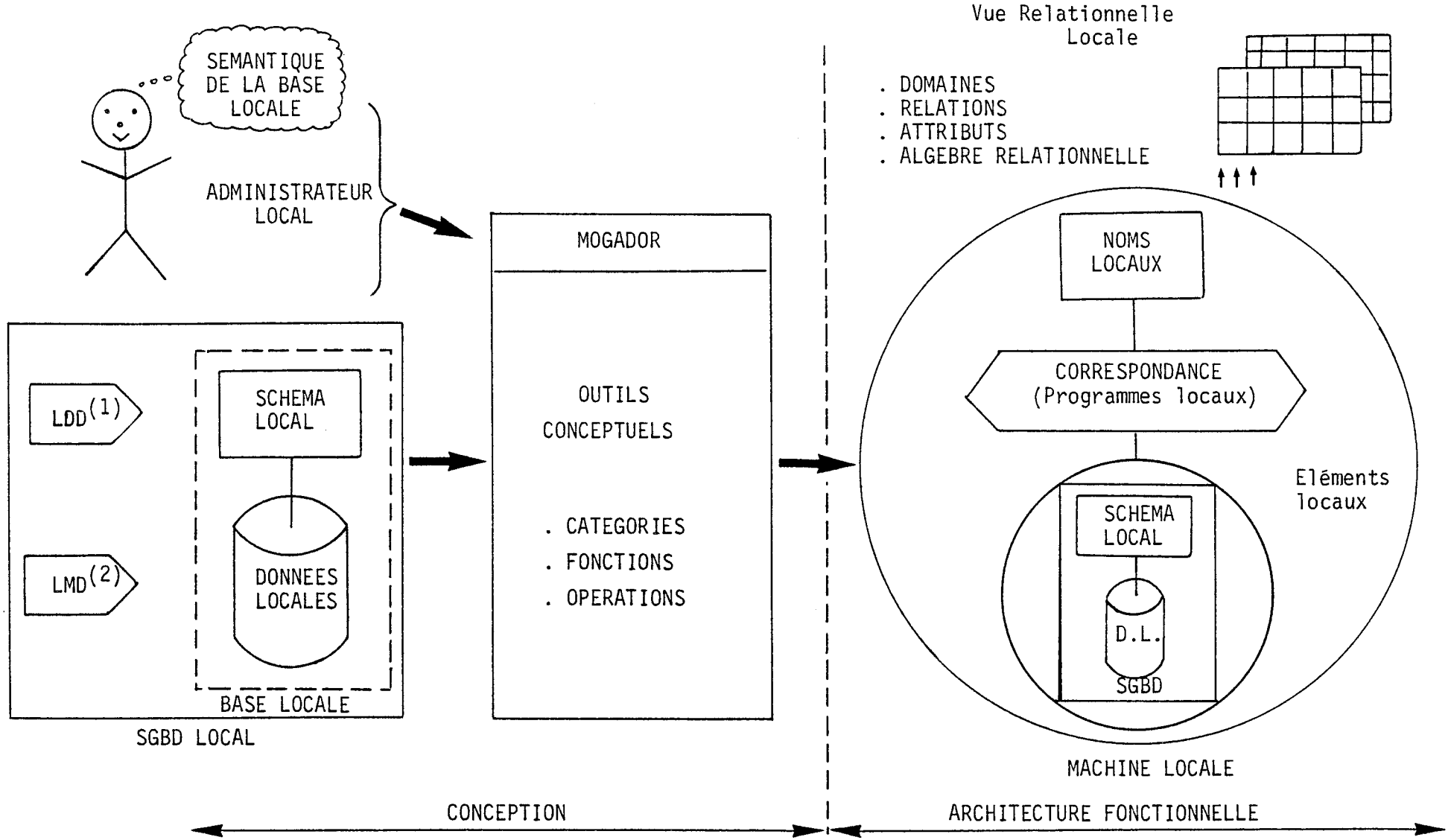
Avant d'aborder le domaine des bases réparties, nous avions travaillé à la comparaison des modèles hiérarchiques et réseaux avec le modèle relationnel. De plus, le système SOCRATE nous a fourni un champ d'expériences.

Dans les notes [B5, B6], nous avons montré avec M. Léonard comment les structures hiérarchiques et réseaux (SOCRATE, CODASYL) s'interprétaient en termes relationnels avant de réaliser un ensemble de programmes traduisant quasi-automatiquement des structures réseaux (SOCRATE) en ensemble de relations n-aires [B5].

L'automatisation d'une telle transformation ne peut en effet être complète car il est impossible d'exprimer dans un schéma toute la sémantique que contient la base de données et que seul son concepteur peut connaître. Un projet ENSIMAG a traité le cas général des structures réseaux (SOCRATE et CODASYL) [B16].

Dans un article de synthèse, nous proposions, avec C. Delobel et M. Léonard [B4] un processus de transformation des structures hiérarchiques ou réseaux en structures relationnelles avec, en tenant compte des relations fonctionnelles, l'élimination des imperfections (processus de normalisation, passage à la 3FN). Ce processus utilisait des outils mathématiques pour aider à la conception des bases relationnelles et a été développé et réalisé par M. Léonard [B39].

Nous nous plaçons ici dans une optique légèrement différente , à savoir appliquer les outils conceptuels fournis par MOGADOR pour aider l'administrateur à mieux concevoir une vue relationnelle d'un ensemble de données stockées et gérées par un SGBD (Figure 3.1).



(1) Langage de Définition de Données.

(2) Langage de Manipulation de Données.

Figure 3.1 - MOGADOR au niveau local

Chaque base locale et les moyens logiciels qui permettent son exploitation sont assimilés à une machine locale.

3.2. CONCEPTION D'UNE VUE RELATIONNELLE LOCALE

Les éléments constituant une base locale sont les suivants (Figure 3.1) :

- un SGBD hiérarchique ou réseau (s'il s'agit d'un SGBD relationnel, le problème est résolu) fournissant un langage de description de données (ou LDD) et un langage de manipulation (LMD)
- un schéma local et éventuellement plusieurs sous-schémas
- un ensemble d'objets simples ou composés : les données de la base
- un ensemble de programmes d'exploitation
- un administrateur local
- un ensemble d'utilisateurs.

Dans un environnement réseau, il convient d'ajouter :

- le site où se trouve la base
- le serveur réseau permettant l'accès à la base.

Le schéma local exprime, par des hiérarchies ou par un réseau, la structure de la base en indiquant en particulier les chemins d'accès et en faisant un mélange entre descriptions conceptuelle et logique.

Etablir une vue relationnelle, c'est arriver à interpréter la base de données en termes de catégories et de fonctions de telle sorte que les données soient vues comme des tables d'objets composés. Bien sûr, cette vue relationnelle n'est pas unique ; elle dépend de la manière dont on veut effectivement voir la base locale (certaines parties peuvent être volontairement ignorées) et des chemins d'accès que l'on exploite.

Un tel processus met en évidence :

- des catégories abstraites locales ou CAL ($CAL \subseteq CA$)
- des catégories concrètes locales ou CCL ($CCL \subseteq CC$) avec les identificateurs locaux (NIL)
- des fonctions locales (FL) monovaluées (FUL) ou multivaluées (FPL).

Le tableau 3.1 donne les parallèles que l'on peut faire entre la terminologie MOGADOR et celle employée dans le modèle hiérarchique représenté par I.M.S., le modèle CODASYL, le modèle réseau représenté par SOCRATE, le modèle relationnel et enfin le modèle Entité-Relation.

Nous allons voir successivement comment les notions de catégories et de fonctions peuvent servir à la reinterprétation d'un schéma local, puis comment les notions dynamiques d'opérations et de machine peuvent s'appliquer pour concevoir une machine relationnelle locale.

3.2.1. Catégories Abstraites Locales (CAL)

En considérant la table 3.1, on constate qu'elles n'ont pas d'équivalents dans les modèles hiérarchiques et réseaux. Il faut alors procéder par approximation : les assimiler aux items (CODASYL) ou aux caractéristiques (SOCRATE) avant de voir comment ces items peuvent être rapprochés parce qu'ils représentent le même ensemble de valeurs.

Considérons l'exemple suivant (SOCRATE) :

```
entité CLIENT                    entité COMMANDE
      NOM mot                          NCOM de 0 à 9999
      :                                NOMC mot
```

Avec une telle structure, seule la personne connaissant la sémantique de la base peut dire que NOM et NOMC correspondent au même ensemble de valeurs : dans l'entité CLIENT, il joue le rôle d'un nom de client, alors que dans l'entité COMMANDE, il joue le rôle du nom du client ayant passé cette commande. Dans MOGADOR, on ne considérera qu'une seule catégorie abstraite, NOM par exemple, qui peut se définir de la manière suivante :

```
cal NOM := CHAINE (16)
```

où cal représente l'opérateur conceptuel de création d'un élément de l'ensemble CAL (cf. Chapitre 2).

Le passage à la vue relationnelle se fera de façon simple en faisant correspondre les notions de catégorie abstraite et de domaine.

MOGADOR	Hiérarchies (I.M.S.)	CODASYL	SOCRATE	Modèle Relationnel	Modèle Entité-Relation
CATEGORIE ABSTRAITE (CA)	/	/	/	DOMAINE	ENSEMBLE DE VALEURS
CATEGORIE CONCRETE (CC)	SEGMENT	RECORD	ENTITE	RELATION	ENTITE - RELATION
IDENTIFICATEUR	CHAMP SEQUENCE	IDENTIFIER	CARACTERISTIQUE DISCRIMINANTE	CLE PRIMAIRE	CLE PRIMAIRE
FONCTION MONO de CC vers CA (attributs)	CHAMP	ITEM	CARACTERISTIQUE ou GROUPE	ATTRIBUT	ATTRIBUT
FONCTION MONO de CC1 vers CC2	HIERARCHIE particulière (1 - 1)	SET avec CC1 membre CC2 propriétaire	REFERENCE de CC1 vers CC2	CLE étrangère comme attribut de CC1	RELATION 1 - 1
FONCTION MULTI de CC1 vers CC2	HIERARCHIE	2 SETS avec un record de liaison	INVERSE ou ANNEAU ou ENTITE IMBRIQUEE	Clé composée de clés d'autres relations	RELATION 1 - N

Tableau 3.1 - Equivalences entre MOGADOR et les modèles de données

3.2.2. Catégories Concrètes Locales (CCL)

Là encore, on peut en première approximation assimiler noms d'entités SOCRATE, noms de "record" CODASYL ou noms de segment I.M.S. aux noms de Catégories Concrètes mais avec toutes les nuances sémantiques que cela entraîne.

En particulier, il faut pour définir une catégorie concrète indiquer quels sont les objets qui permettent d'identifier les éléments composant cette catégorie. Or, dans les SGBD classiques, cette notion d'identifiant ou de clé primaire n'est pas imposée aux administrateurs, qui sont libres de définir ou non un item, une caractéristique comme champ clé.

On distingue en fait dans les SGBD usuels deux types d'identificateurs :

- *identificateur interne* : il ne fait pas partie intégrante de l'objet concret mais est utilisé par le SGBD pour repérer une occurrence de manière interne : adresse virtuelle en SOCRATE, identificateur de n-uplet dans un système relationnel. Ce type d'identifiant ne présente aucun intérêt dans l'optique de la coopération de bases de données.

- *identificateur de type clé* : il fait partie intégrante de l'objet et a de ce fait une signification précise en dehors de la base locale : par exemple, le numéro INSEE d'une personne, le numéro d'immatriculation d'un véhicule.

Au niveau conceptuel, il importe alors de définir pour chaque catégorie concrète l'identifiant associé (ensemble NIL).

On peut ainsi définir des catégories concrètes locales, par exemple :

1) ccl PERSONNE — Création d'un élément de la catégorie CCL
2) nomid(PERSONNE) ← INSEE — Lier une CCL à son identifiant.

Un autre problème est d'analyser les liaisons entre catégories concrètes pour distinguer ce qui correspond à des chemins d'accès, de ce qui constitue réellement une liaison entre ensemble d'objets. Illustrons-le par l'exemple suivant (Figure 3.2.a) où on a représenté une structure hiérarchique :

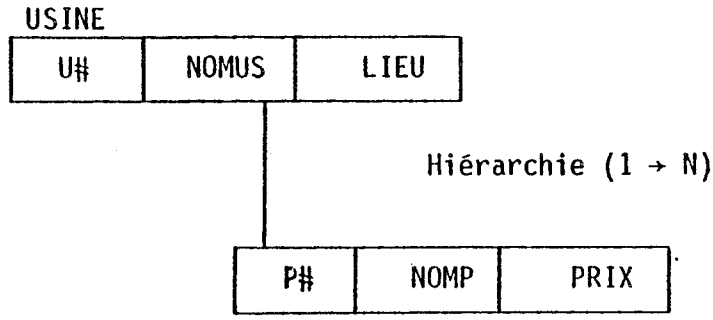


Figure 3.2.a - Exemple de structure hiérarchique

Cette structure représente les faits élémentaires suivants :

- l'existence d'usines décrites par un numéro (U#), un nom (NOMUS), et localisées (LIEU)
- le fait qu'une usine fabrique plusieurs produits décrits par un numéro (P#), un nom (NOMP), un prix.

En supposant U# et P# uniques respectivement pour les usines et les produits, on définit complètement deux catégories concrètes USINE et PRODUIT.

Il faut alors remarquer (relations fonctionnelles) :

1) qu'une usine n'a qu'un nom et qu'un lieu
2) qu'un produit n'a qu'un nom et qu'un prix.

On aboutit alors au schéma MOGADOR suivant (Figure 3.2.b) :

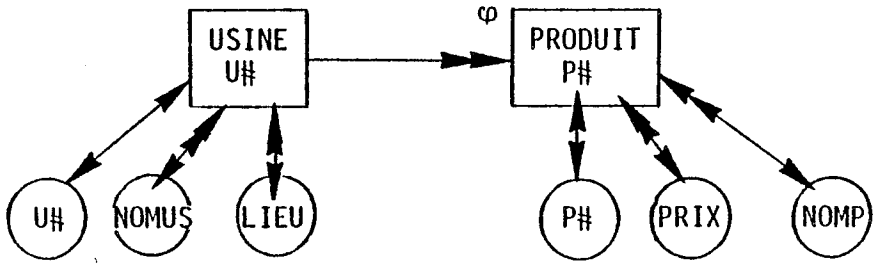
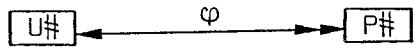


Figure 3.2.b

Il faut alors examiner la nature de l'inverse de la fonction φ :

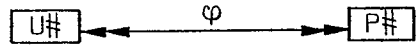
- si inv φ est monovaluée, cela signifie qu'un produit n'est fabriqué que dans une usine :



Et conformément aux transformations décrites au § 2.4.2, on aura la vue relationnelle suivante (relations en 3FN) :

{ USINE (U#, NOMUS, LIEU)
{ PRODUIT (P#, PRIX, NOMP, U#).

- si inv φ est multivaluée :



Les transformations donnent alors le schéma suivant (Figure 3.2.c) :

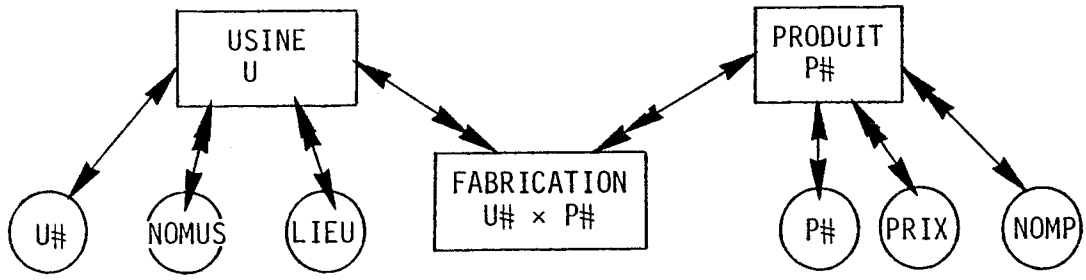


Figure 3.2.c

Et le passage à la vue relationnelle donne :

{ USINE (U#, NOMUS, LIEU)
{ PRODUIT (P#, PRIX, NOMP)
{ FABRICATION (U#, P#).

3.2.3. Fonctions locales (FUL, FPL)

Le schéma local de la base décrit des liaisons entre ensembles d'objets à partir desquelles il faut définir des fonctions MOGADOR mono ou multivaluées.

Nous allons étudier le problème sur une structure réseau exprimée à l'aide du langage SOCRATE : la structure S1 donnée ci-après.

En première approximation, on peut dresser un premier schéma MOGADOR en assimilant entité et catégorie concrète (Figure 3.3.a).

Base B1 : Structure SOCRATE S1 :

```
entité 20 USINE
    |   U# mot discr
    |   LIEU mot rapide
    |   BUDGET de 0 à 999999
    |   EMPLOYES anneau
    |   PRODUITS anneau
    | entité 5000 COMMANDE
    |     |  C# de 0 à 5000
    |     |  NOM-CLIENT mot
    |     |  ADR-CLIENT mot
    |     |  DATE-COM de 0 à 999999
    |     |  entité 500 PROD-COM
    |     |     |  P# mot
    |     |     |  QTEC de 0 à 999

entité 1000 PRODUIT
    |   P# mot discr
    |   NOMP mot rapide
    |   PROD-FAB anneau

entité 500 FABRICATION
    |   NUS réfère PRODUITS de une USINE
    |   NUP réfère PROD-FAB de un PRODUIT
    |   QTEF de 0 à 1000

entité 5000 EMPLOYE
    |   INSEE mot discr
    |   NOMEMP mot rapide
    |   AGEMP de 0 à 99
    |   SALAIRE de 0 à 99999
    |   USEMP réfère EMPLOYES de une USINE
    |   ADRESSE
    |         RUE mot
    |         VILLE mot
    |         CODE de 0 à 99999
```

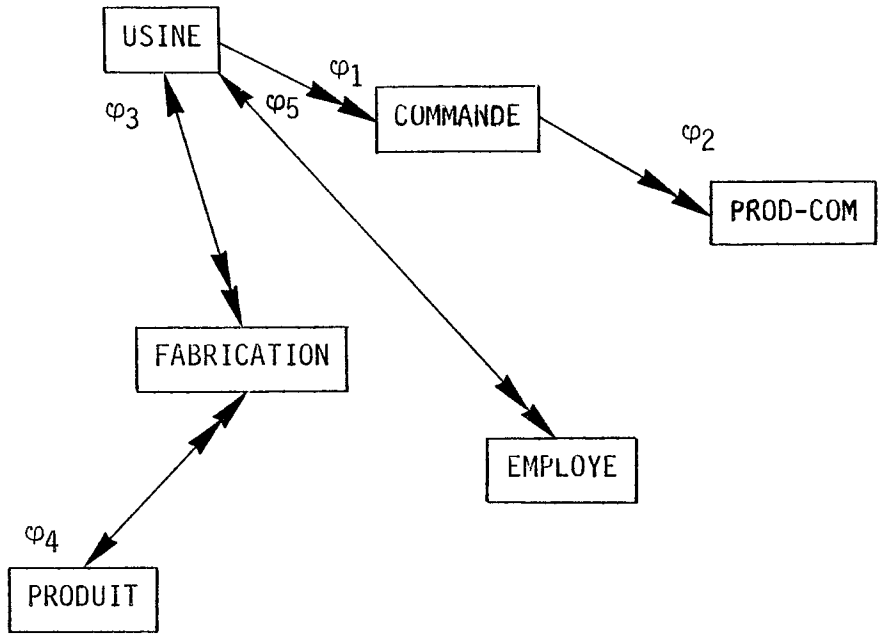


Figure 3.3.a

Dans ce schéma et conformément au tableau 3.1, on a représenté les liaisons au moyen de fonctions :

- φ_1 lien hiérarchique de USINE vers COMMANDE
- φ_2 lien hiérarchique de COMMANDE vers PROD-COM
- φ_3 référence de FABRICATION à USINE
- φ_4 référence de FABRICATION à PRODUIT
- φ_5 référence de EMPLOYE à USINE.

Les inverses de φ_1 et φ_2 ne sont pas mentionnées, essentiellement à cause des problèmes évoqués au paragraphe précédent. Par contre, les inverses des autres fonctions sont mentionnées, du fait de l'existence d'anneaux matérialisant le chemin inverse (liaisons de type 1→N).

Les fonctions MOGADOR représentant des liaisons orientées, il est possible de traduire l'existence d'un chemin d'accès et/ou de son inverse (cf. § 3.2.3.1 et 3.2.3.2).

Cependant, le premier problème à partir de ce schéma MOGADOR est de déterminer quels sont les identificateurs des catégories concrètes ainsi mises en évidence.

Lorsque l'identificateur est explicitement indiqué dans la structure, il n'y a pas de problème. C'est le cas pour celui de la CCL USINE :

U# est déclaré avec l'option "discr" (c'est-à-dire discriminant). Il en est de même avec PRODUIT (P#) et EMPLOYE (INSEE).

En ce qui concerne les entités imbriquées, on peut admettre par défaut que la première caractéristique sert d'identifiant par rapport à l'entité mère (analogue avec le champ séquence I.M.S), puis appliquer un processus de normalisation :

COMMANDE identifiée par U# × C#

PROD-COM identifié par U# × C# × P#.

En réalité, ces informations ne peuvent pas être déduites automatiquement sans apport sémantique de la part de l'administrateurs local.

Ainsi FABRICATION sera-t-elle identifiée par le couple U# × P#.

Notons qu'en CODASYL, il est par contre possible de spécifier dans le LDD qu'un ensemble de "data-items" forme un identificateur (cf. § 3.3.4).

Une fois chaque CCL définie complètement, on peut s'intéresser aux liaisons qui existent entre les autres éléments du schéma local, liaisons interprétées dans MOGADOR à l'aide de la notion de fonction.

Du fait de l'expression de la structure, deux types de fonctions sont (en général) mis en évidence :

- les fonctions entre catégories concrètes et abstraites
- les fonctions entre catégories concrètes.

Par contre, les fonctions entre catégories abstraites sont invisibles si on ne connaît pas la sémantique des données représentées. Nous en avons un exemple dans la structure S1 avec la liaison existante entre le nom d'un client et son adresse, sur laquelle nous revenons plus loin.

3.2.3.1. Fonctions entre Catégories Concrètes et Abstraites

Dans le sens concret vers abstrait, elles sont en général monovaluées et correspondent donc aux liaisons entité vers caractéristiques dans SOCRATE, "record" vers "data-item" dans CODASYL. Sur le plan statique de la description du schéma, ces fonctions décrivent ce type de chemin d'accès, tandis que sur le plan dynamique, elles correspondent aux possibilités d'accès aux objets liés à un objet donné : ce genre d'opération fait intervenir des ordres élémentaires du langage de manipulation propre au SGBD (LMD).

Par exemple, dans la structure S1, la liaison :

USINE ——► LIEU

s'interprète dans MOGADOR à l'aide d'une fonction monovaluée, appelons-la par exemple "lieu-usine".

L'opération élémentaire, qui à une usine u relie une valeur v qui représente la ville où est située l'usine u s'exprime dans MOGADOR :

lier (lieu-usine,u,v) ou bien lieu-usine(u) ← v.

Cette expression a un équivalent direct dans le langage de manipulation SOCRATE, à savoir :

"m X1 = une USINE avec U# = u ;
m LIEU de X1 = v".

L'USINE est identifiée par la catégorie abstraite U#.

De la même manière, l'opération élémentaire d'accès à partir d'une usine u à la valeur de sa caractéristique LIEU qui dans MOGADOR s'écrit :

accéder (lieu-usine, u)

correspond dans SOCRATE à

"LIEU de une USINE avec U# = u ;"

Sur le plan de l'existence de l'inverse d'une fonction, il faut regarder dans la structure SOCRATE ou dans un schéma local si le chemin d'accès inverse est défini.

Dans notre exemple, LIEU a été déclaré avec une option "rapide", ce qui signifie que cette caractéristique sert de clé d'accès (le SGBD construit des index secondaires pour permettre l'accès direct aux USINES dont on connaît le lieu).

Ainsi dans notre exemple, inv lieu-usine est définie (a priori elle est multivaluée) et l'expression MOGADOR inv lieu-usine (PARIS) a une équivalente directe en LMD SOCRATE, à savoir :

U# de toute USINE avec LIEU = "PARIS".

Par contre, si l'on considère la fonction (monovaluée) "budget" qui va de USINE à BUDGET,

USINE —budget►BUDGET

son inverse est a priori non défini, puisqu'il n'existe pas de chemin direct qui, à partir d'une valeur de budget, permette d'accéder aux usines correspondantes.

Ainsi l'opération :

(1) inv-budget (100)

qui se lit : "accéder aux usines ayant un budget égal à 100 kf" devra correspondre à un parcours séquentiel de toutes les usines, pour déterminer celles qui correspondent au critère.

L'opération (1) se transforme alors en (2) :

(2) projeter (sélecter (graphe (budget), BUDGET=100), U#)

qui signifie : sur le graphe de la fonction budget vu comme un tableau à deux colonnes : (BUDGET,U#), il faut sélectionner les couples où BUDGET = 100 et ne garder par projection que les numéros d'usines (U#).

L'ensemble des fonctions locales de ce type se définit par :

$F_{attribut} = \{f \in FMONO \mid source(f) \in CCL \wedge cible(f) \in CAL\text{-}NIL\}$.

3.2.3.2. Fonctions entre Catégories Concrètes

Elles peuvent être mono ou multi-valuées et correspondent aux liaisons existant entre catégories concrètes, ainsi qu'aux chemins d'accès correspondant.

On peut les définir de la manière suivante :

$F_{CCL} = \{f \in FMONO \cup FMULTI \mid source(f) \in CCL \wedge cible(f) \in CCL\}$.

Ces liaisons se dénotent dans un schéma soit par déclaration explicite de chemin (référence-anneau, entités imbriquées, inverses en SOCRATE, "set" en CODASYL) soit par partage implicite d'un domaine de valeurs entre deux entités (cf. § 3.2.1) (c'est le cas entre PROD-COM et PRODUIT).

En ce qui concerne la structure S1, les fonctions MOGADOR de ce type sont indiquées dans la Figure 3.3.a :

La fonction φ_1 qui va de USINE à COMMANDE est multivaluée :

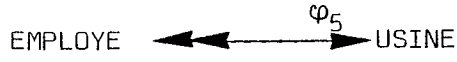
Ce chemin existe réellement et correspond dans le LMD SOCRATE à des ordres de la forme :

$\varphi_1(u)$ est équivalent à :

```
m X1 = une USINE avec  U# = u ;
pour toute COMMANDE X2 de X1
    |   i U# de X1
    |   i C# de X2
fin
```

Quant au chemin inverse, il peut ne pas exister, comme c'est le cas dans le SOCRATE prototype alors que dans les autres versions de SOCRATE, il est disponible grâce au mot clé "dont" [B56].

Si nous considérons un exemple de fonction correspondant à une liaison référence-anneau, nous avons par exemple pour φ_5 :



Ainsi φ_5(1440899350300) correspond à (utilisation de la référence) :
i U# de USEMP de un EMPLOYE avec INSEE = '1440899350300' tandis que
$\underline{\text{inv}}\ \varphi_5(u)$ s'écrit en SOCRATE (parcours de l'anneau)
i INSEE de tout EMPLOYES de une USINE avec U# = u ;

La figure 3.3.b donne les liaisons internes à l'entité USINE. Cependant, on peut remarquer une fonction monovaluée entre NOM-CLIENT et ADR-CLIENT qui a été détectée par l'administrateur local : le nom d'un client permet d'avoir son adresse. Cette situation est à éliminer, elle correspond au cas 7 table 2.9 (il s'agit ici d'une transitivité dans les relations fonctionnelles : C# → NOM-CLIENT, ADR-CLIENT or NOM-CLIENT → ADR-CLIENT). Après élimination, le schéma MOGADOR complet correspondant à S1 est donné par la figure 3.3.c. Les fonctions qui ne correspondent pas à un chemin explicite dans la structure sont signalées par une "*".

Ainsi que nous venons de le voir, la notion de fonctions MOGADOR s'applique pour interpréter la description des bases locales : elles explicitent les liaisons entre catégories et de plus leur graphe est matérialisé dans la base sóit par des pointeurs (références SOCRATE, "set" CODASYL) soit par d'autres méthodes (proximité physique par exemple) mais qui au niveau où nous nous plaçons ne nous intéressent pas.

Les seuls éléments importants à considérer sont :

- l'*existence* d'une liaison entre ensembles d'objets
- l'exploitation de cette liaison moyennant un certain coût.

- L'existence sera exprimée dans l'espace des noms de la machine locale grâce à la vue relationnelle.
- L'exploitation de la liaison sera réalisée au moyen de "programmes locaux" écrits dans le langage de manipulation du SGBD local (cf. § 3.3.2).
- Le coût d'accès dépend, d'une part de l'existence d'un chemin direct ou de la nécessité d'effectuer des parcours séquentiels, et d'autre part de la taille des ensembles sources et cibles.

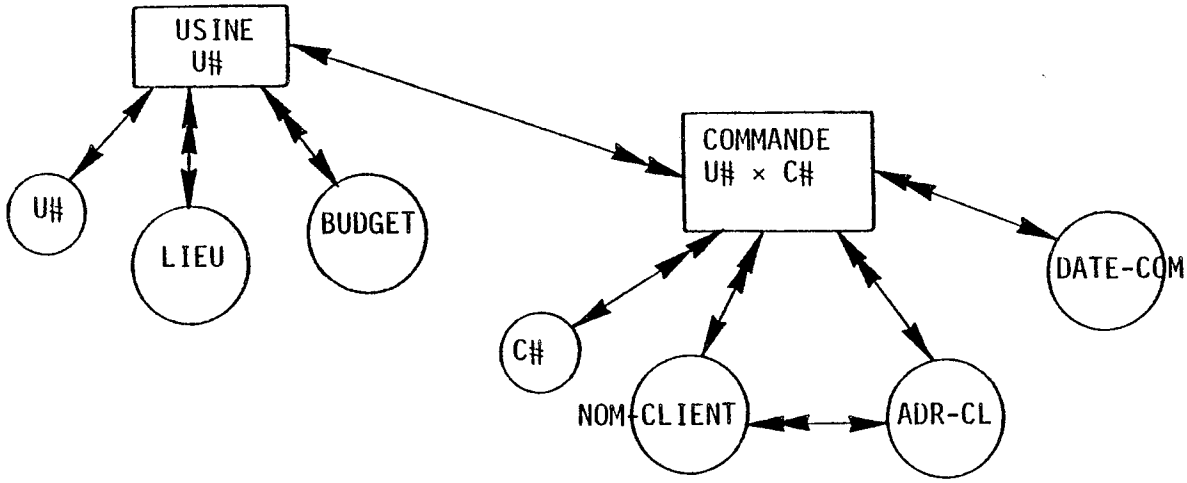


Figure 3.3.b

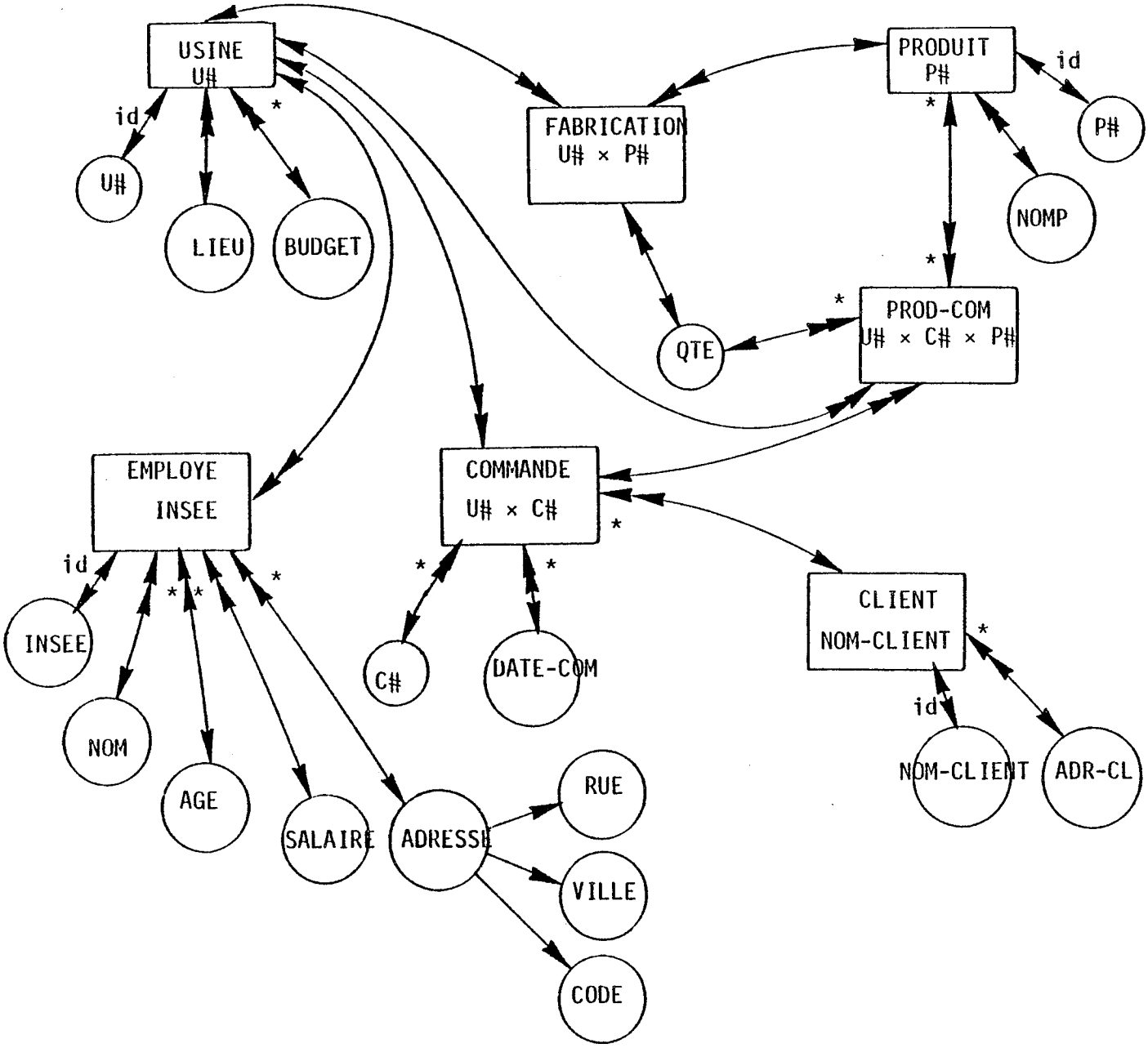


Figure 3.3.c

3.3. MACHINE LOCALE

La conception de la machine locale passe par l'interprétation grâce à MOGADOR des informations du schéma ainsi que nous venons de le voir aux paragraphes précédents. Cette interprétation revient en fait à décomposer la base locale en éléments ayant une sémantique propre (catégories concrètes, fonctions), avant d'effectuer un certain nombre de regroupements grâce au passage à une vue relationnelle n-aire.

Ces regroupements tiennent compte en effet de la réalité actuelle des SGBD pour qui l'unité minimale d'échange est l'occurrence de segment (I.M.S), l'occurrence de "record" (CODASYL), la réalisation d'entité (SOCRATE).

3.3.1. Vue Relationnelle Locale

Pour constituer l'espace des noms de la machine locale, il faut définir la vue locale en partant du schéma MOGADOR et en appliquant le processus décrit au § 2.4.

On arrive ainsi à la définition de :

- domaines locaux (correspondent aux catégories abstraites)
- relations locales avec leurs attributs clés et non clé.

Si nous appliquons ce processus au schéma donné par la figure 3.3.c, nous avons la vue relationnelle suivante :

- DOMAINES LOCAUX : U#, LIEU, BUDGET, INSEE, NOM, AGE, ADRESSE, SALAIRE, C#, DATE-COM, NOM-CLIENT, ADR-CLIENT, QTE, P#, NOMP.

- RELATIONS LOCALES :

  USINE (U#, LIEU, BUDGET)
  EMPLOYE (INSEE, U#, NOM, AGE, SALAIRE, ADRESSE)
  FABRICATION (U#, P#, QTEF)
  PRODUIT (P#, NOMP)
  COMMANDE (U#, C#, DATE-COM, NOM-CLIENT)
  CLIENT (NOM-CLIENT, ADR-CLIENT)
  PROD-COM (U#, C#, P#, QTEC)

D'autres paramètres descriptifs interviennent au niveau de la vue locale :

- l'*identification de la base* sous-jacente et de son environnement logiciel et matériel. Ce problème rejoint les problèmes d'implantation physique de la machine locale qui sont traités au § 5.4.

- *pour chaque relation* on peut indiquer (§ 5.5.3.4) :
  - la taille (en nombre de caractères) d'un n-uplet (en réalité ce paramètre peut être calculé, connaissant la représentation de chaque domaine et la correspondance domaine-attribut)
  - le nombre moyen de n-uplets composant la relation. Cette valeur, bornée supérieurement dans le cas de SOCRATE par le nombre maximum de réalisations déclaré pour chaque entité (cf. structure S1), n'est pas toujours facile à obtenir. En particulier, lorsque la relation locale ne provient pas directement d'une entité : quel est le nombre de clients par exemple ?

Les SGBD actuels donnent certains moyens de comptage pouvant être utilisés à la mise en place de la machine locale. Ce problème est crucial dans un environnement réparti du fait des coûts de communication entre machines situées sur des sites distants.

- l'*identification des usagers* habilités à se servir de la machine locale et leur mode d'utilisation. Dans une première approche nous avons choisi deux modes : un mode interrogatif (lecture seule) et un mode mise à jour (lecture, écriture) par utilisateur (cf. § 5.4). De plus, des mécanismes d'autorisations sont associés aux relations locales en ce qui concerne les opérations d'accès et de mise à jour. Ces autorisations se matérialisent par la présence ou l'absence de programmes locaux (cf. § 3.3.2).

Ce problème rejoint le problème général de la confidentialité, de l'intégrité des informations dans un environnement réparti, et constitue un sujet de travail au sein de l'équipe POLYPHEME.

3.3.2. Correspondance Noms-Eléments : Programmes locaux

Conceptuellement une machine locale est capable d'exécuter une transaction (locale) exprimée à l'aide d'un langage relationnel (LA.DO.RE) et portant sur les relations locales (cf. § 2.5).

Par ailleurs, la partie correspondance Noms-Eléments doit décrire de manière algorithmique les règles d'évolution pour chaque relation locale.

Dans une première approche [R3], nous avons proposé une solution à ce problème qui permet de construire des machines locales de manière relativement économique.

Cette solution consiste à disposer de programmes locaux, réalisant les opérations primitives (insérer, supprimer, obtenir, modifier) mais également les opérateurs de projection et de sélection relationnels, ceci dans le but de réduire les ensembles d'objets extraits d'une machine locale.

Dans la mesure où ces programmes locaux utilisent le langage de manipulation du SGBD local, cela revient à faire faire le maximum d'opérations à ce système ; l'investissement concernant la mise en place d'un interface avec le SGBD local se ramène ensuite à l'implantation d'activateurs de programmes locaux.

Ceci évite d'avoir sur chaque machine locale "physique" un interpréteur (relationnel) réalisant tous les opérateurs de l'algèbre relationnelle (cf. § 5.4).

Ainsi, les *quatre opérations de base* sont réalisées par les programmes locaux :

- *programme d'accès séquentiel (pas)* qui fournit l'ensemble des n-uplets d'une relation
- *programme de création (pcr)* pour l'insertion d'un nouvel n-uplet
- *programme de suppression (psp)*
- *programme de mise à jour (pmj)* pour la modification des valeurs d'un n-uplet.

- Les opérations de *sélection* sont réalisées par deux autres types de programmes locaux :

  - *programme d'accès par clé (pac)* qui fournit le n-uplet dont on donne la valeur de la clé
  - *programme d'accès associatif (paf)* qui pour une valeur de filtre donné, donne l'ensemble des n-uplets correspondants.

- Les opérations de *projection* sont réalisées en paramétrant les programmes de type pas, pac, paf pour qu'ils ne fournissent en sortie que des sous n-uplets.

A chaque relation locale sont donc associés six programmes locaux.

Pour une relation locale quelconque RL, il existe une zone de communication entre l'usager de la machine et les mécanismes assurant l'activation des programmes locaux : cette zone, en tous points analogue à la zone de travail utilisateur dans les systèmes CODASYL [B26] ou à la notion de "FORMAL" SOCRATE [B56], contient un emplacement pour chaque attribut, avec en plus un emplacement réservé à des paramètres de contrôle (code retour, filtre dans le cas d'un accès associatif). Selon le type du programme, cette zone est utilisée en entrée et en sortie de différentes façons :

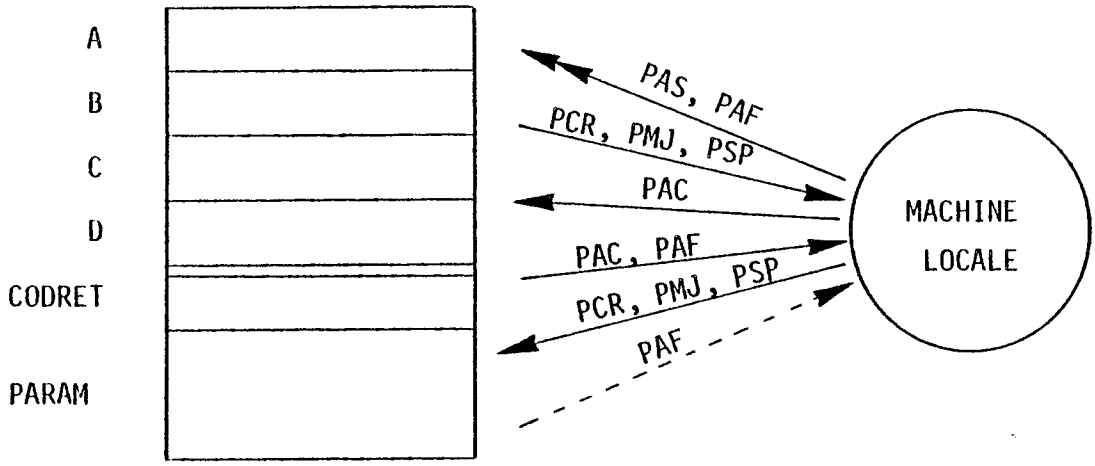


Figure 3.4 - Zone de communication associée à une relation locale : RL (A,B,C,D)

——► 1 seul n-uplet ——►► plusieurs n-uplets

- - -► paramètres supplémentaires (filtre pour PAF).

3.3.3. Description des Programmes Locaux

Les programmes locaux sont donc classés en six catégories (concrètes) PAC, PAF, PAS, PCR, PMJ, PSP et leur description consiste à générer les éléments qui les constituent (cf. § 2.2.1.5).

En général, de ces programmes ne sont connus au niveau de l'usager de la machine locale que leur nom, leur fonction (ou leur type), les entrées et les sorties. Le corps du programme écrit dans le langage de manipulation du SGBD local est en quelque sorte une "boîte noire" pour cet usager.

Au niveau de la description, nous avons :

- programmes de création (généré par pcr)
- programmes de suppression (généré par psp)
- programme de mise à jour (généré par pmj)
- programme d'accès : séquentiel (généré par pas), par clé (pac) et associatif (paf).

3.3.3.1. Programme de Création (pcr)

Il a pour but de créer un n-uplet dans la relation correspondante. Sa description comprend :

- le symbole pcr
- le nom du programme
- les paramètres d'entrée (précédés de entrée) : il s'agit des noms des domaines auxquels appartiennent les valeurs qui composent le n-uplet à créer (ces domaines peuvent être implicites par la définition de la relation dans l'espace des noms)
- le paramètre de sortie (précédé de sortie) : en général une variable CODRET, qui prend une valeur logique vraie ou faux (ou 0 et 1) selon que l'opération s'est effectuée correctement ou non.

Exemple :
Nous allons décrire le programme qui permet la création d'un n-uplet de la relation FABRICATION décrite ci-dessus.

```
pcr PCRFAB entrée U#, P#, QTEF
           sortie CODRET.
```

Le corps du programme, écrit en SOCRATE, est donné ci-après. On remarquera que si l'usine et le produit correspondant n'existent pas dans la base, la création sera refusée. Dans un souci de clarté, une certaine liberté est prise avec la syntaxe du langage SOCRATE, en particulier pour affecter une valeur à une variable sans passer par des variables systèmes (Yi, Zi, Mi

Nous noterons symboliquement les valeurs des paramètres d'entrée par le nom de la catégorie précédée de "&" :

```
corps
     si existe une USINE X1 avec  U# = &U  alors
     |  si existe un PRODUIT X2 avec  P# = &P  alors
     |  |     g une FABRICATION X3
     |  |     m NUS de X3 = X1
     |  |     m NUP de X3 = X2
     |  |     m QTEF de X3 = &QTEF
     |  |     m CODRET = 0 sortie
     |  fin
     fin
           m CODRET = 1 sortie
fin
```

3.3.3.2. Programme de suppression (psp)

Sa description comprend :

- le symbole psp
- le nom du programme
- les paramètres d'entrée : entrée suivi d'une liste des domaines composant la clé
- le paramètre de sortie (voir pcr).

3.3.3.3. Programme de mise à jour (pmj)

Sa description comprend :

- le symbole pmj
- le nom du programme
- les paramètres d'entrée : entrée suivi d'une liste de domaines qu'il est possible de mettre à jour. Notons qu'il est ainsi possible d'autoriser ou d'interdire certaines opérations de mise à jour sur une relation locale
- le paramètre de sortie (voir pcr).

3.3.3.4. Programmes d'accès (pas, pac, paf)

Nous considérons à ce niveau trois catégories de programmes d'accès : les accès *séquentiels* (correspondance nom → objets), les accès *associatifs* (correspondances nom-objets → objets) et les accès par *clé*.

Accès séquentiels

Un accès séquentiel correspond à l'énumération des objets composés de la relation locale concernée. Il peut y avoir plusieurs types d'ordre sur les éléments d'un ensemble d'objets. Dans une vue locale, il pourrait donc y avoir plusieurs programmes d'accès séquentiel pour une même relation.

Au niveau de la description, on trouve :

- le symbole pas
- le nom
- éventuellement le type d'ordre sur les objets énumérés
- pas de valeurs à fournir en entrée
- en sortie, la liste des domaines désirés pour réaliser éventuellement des projections.

Accès associatif

L'accès associatif aux objets d'une relation locale se fait en citant tout ou partie d'un objet dans un filtre. Un filtre est composé de plusieurs filtres élémentaires, chacun d'eux concernant un attribut de la relation.

Pour deux attributs A et B quelconques, un filtre élémentaire a la forme suivante : $<A\ \theta\ a>$ ou $<A\ \theta\ B>$.

- θ un des opérateurs $= \neq < \leq \geq <$
- a un objet appartenant au domaine associé à A.

Dans le cas général, un filtre quelconque peut être défini de la façon suivante :

. tout filtre élémentaire est une expression conditionnelle
. si γ est une expression conditionnelle, $\neg\gamma$ est une expression conditionnelle
. si $\gamma 1$ et $\gamma 2$ sont deux expressions conditionnelles alors $\gamma 1 \vee \gamma 2$ et $\gamma 1 \wedge \gamma 2$ sont deux expressions conditionnelles
. si γ est une expression conditionnelle alors (γ) est aussi une expression conditionnelle.

Par exemple, sur la relation R(A,B,C,D,E) on peut composer le filtre :
((<A = 10> ∧ <B ≤ 5>) ∨ <C = 3>) ∧ <D = E>.

Dans le cas des programmes locaux, nous nous limiterons à un seul filtre élémentaire par attribut et à des intersections de filtres élémentaires.

La description d'un programme d'accès associatif comprend :

- le symbole paf
- le nom du programme
- en entrée, les noms des domaines pouvant être filtrés. Ceci correspondant en fait aux fonctions multivaluées inverses des fonctions de type attribut et le filtrage ne peut se faire que si le chemin d'accès correspondant existe dans la base (cf. § 3.2.3)
- en sortie, les noms des domaines qu'il est possible d'obtenir, assurant ainsi une éventuelle projection.

Le résultat d'un tel programme est un ensemble de n-uplets.

Accès par clé

Un programme de ce type fournit le n-uplet ayant pour clé la valeur du n-uplet d'entrée.

Sa description comprend :

- le symbole pac
- le nom
- en entrée la valeur de la clé
- en sortie les noms des domaines désirés pour réaliser une éventuelle projection.

Exemples :

i) Considérons le programme d'accès séquentiel sur la relation EMPLOYE (INSEE, U#, NOM, AGE, SALAIRE, ADRESSE), projetée sur les attributs (INSEE, U#, NOM, SALAIRE) :

pas PASEMP sortie INSEE, U#, NOM, SALAIRE.

Au niveau de son écriture en SOCRATE, remarquons qu'il faut utiliser le chemin d'accès EMPLOYE vers USINE (référence USEMP) pour obtenir la valeur de U#.

```
corps
     pour tout EMPLOYE X1
        | m INSEE = INSEE de X1
        | m U# = U# de USEMP de X1
        | m NOM = NOM de X1
        | m SALAIRE = SALAIRE de X1
     fin
```

ii) Considérons un programme d'accès associatif sur la relation USINE. Comme seul LIEU est déclaré en rapide dans les attributs non clé, nous avons (on considère un filtre simple d'égalité avec une valeur donnée) :

```
paf PAFUSI entrée U#, LIEU
           sortie U#, LIEU, BUDGET
```

Selon la caractéristique filtrée, on a :

1) Pour U# :

```
m X1 = une USINE avec  U# = &U# ;
m LIEU = LIEU de X1
m BUDGET = BUDGET de X1
```

2) Pour LIEU :

```
pour toute USINE X1 avec LIEU = &LIEU ;
   | m U# = U# de X1
   | m BUDGET = BUDGET de X1
fin
```

iii) Programme d'accès séquentiel aux commandes

```
pas PASCOM sortie U#, C#, NOM-CLIENT, DATE-COM.
```

Il faut ici passer par l'entité mère :

```
pour toute USINE X1
   | m U# = U# de X1
   | pour toute COMMANDE X2 de X1
   |    | m C# = C# de X2
   |    | m NOM-CLIENT = NOM-CLIENT de X2
   |    | m DATE-COM = DATE-COM de X2
   | fin
fin
```

iv) Accès aux clients : remarquons qu'ici la génération d'une nouvelle relation CLIENT pose des problèmes au niveau de l'accès à la base. En effet, le parcours sequentiel de toutes les commandes donne la liste de tous les clients ayant passé des commandes, mais cette liste contient vraisemblablement des redondances. En d'autres termes, l'ensemble résultat est un pré-ensemble. C'est la raison pour laquelle nous avons introduit l'opérateur unique dont la fonction est d'éliminer les duplications (cf. § 2.4).

3.3.4. Exemples de machines locales

Avec l'exemple SOCRATE traité au cours de ce chapitre, nous avons défini la machine locale correspondante : soit M1 cette machine.

Afin de montrer que ce processus s'applique à des bases hétérogènes, nous allons considérer une autre base locale B2 décrite par un schéma CODASYL.

Cette base B2, nous supposons qu'elle a été réalisée dans la même entreprise que la base B1 en ce sens qu'elle gère également un ensemble d'unités de fabrication et d'employés.

Cependant, l'organisation de la structure diffère par les chemins d'accès et l'ordre dans lequel sont rangés les objets.

Schématiquement la structure CODASYL est donnée par la Figure 3.4.

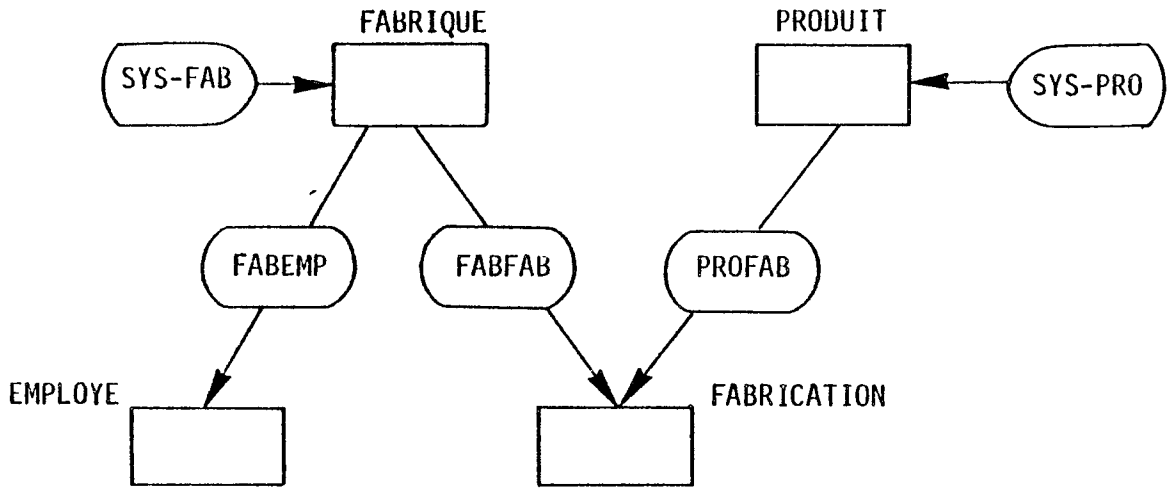


Figure 3.4 - Schéma CODASYL Base B2 : 4 "records" et 3 "sets"

"SYS-FAB" et "SYS-PRO" désignent des "singular sets" (sets dont le propriétaire est le système) qui vont permettre le parcours séquentiel des enregistrements FABRIQUE et PRODUIT.

Le schéma CODASYL de B2 exprimé dans le langage de description est :

```
SCHEMA NAME IS B2.
AREA NAME IS PERSONNEL.
AREA NAME IS FABRICATION.
RECORD NAME IS FABRIQUE ;
    LOCATION MODE IS CALC HASH-FAB USINE NF IN FABRIQUE ;
    WITHIN FABRICATION ;
    IDENTIFIER IS NF IN FABRIQUE.
    02 NF ; TYPE IS CHARACTER 6.
    02 NOMF ; TYPE IS CHARACTER 10.
    02 ADRF ; TYPE IS CHARACTER 15.
    02 BUDGET ; TYPE IS FIXED DECIMAL 6.
RECORD NAME IS EMPLOYE ;
    LOCATION MODE IS CALC H-EMP USING MATRICULE IN EMPLOYE ;
    WITHIN PERSONNEL ;
    IDENTIFIER IS MATRICULE IN EMPLOYE.
    02 MATRICULE ; TYPE IS CHARACTER 13
    02 NOM ; TYPE IS CHARACTER 10.
    02 AGE ; TYPE IS FIXED DECIMAL 2.
    02 NBH ; TYPE IS FIXED DECIMAL 3.
    02 TAUX ; TYPE IS FIXED DECIMAL 3.
RECORD NAME IS PRODUIT ;
    LOCATION MODE IS CALC H-PROD USING NP IN PRODUIT ;
    WITHIN FABRICATION ;
    IDENTIFIER IS NP IN PRODUIT.
    02 NP ; TYPE IS CHARACTER 6.
    02 NOMP ; TYPE IS CHARACTER 10.
RECORD NAME IS FABRICATION ;
    LOCATION MODE IS SYSTEM DEFAULT ;
    WITHIN FABRICATION ;
    IDENTIFIER IS NP IN FABRICATION, NF IN FABRICATION.
    02 NP ; TYPE IS CHARACTER 6.
    02 NF ; TYPE IS CHARACTER 6.
    02 QTE ; TYPE IS FIXED DECIMAL 4.
```

```
SET NAME IS FABEMP ;
    OWNER IS FABRIQUE ;
    ORDER IS PERMANENT SORTED BY DEFINED KEYS.
    MEMBER IS EMPLOYE ;
         INSERTION IS MANUAL
         RETENTION IS OPTIONAL ;
         KEY IS ASCENDING NOM IN EMPLOYE
               DUPLICATES ARE ALLOWED
               NULL IS NOT ALLOWED ;
               SET SELECTION IS THRU FABEMP OWNER
               IDENTIFIED BY IDENTIFIER NF IN FABRIQUE.
```

```
SET NAME IS PRODFAB ;
    OWNER IS PRODUIT ;
    ORDER IS PERMANENT SORTED BY DEFINED KEYS.
    MEMBER IS FABRICATION ;
         INSERTION IS AUTOMATIC
         RETENTION IS MANDATORY ;
         KEY IS DESCENDING NF IN FABRICATION
               DUPLICATES ARE NOT ALLOWED
               NULL IS NOT ALLOWED ;
    SET SELECTION IS THRU PROFAB OWNER
         IDENTIFIED BY IDENTIFIER NP IN PRODUIT.
```

```
SET NAME IS FABFAB ;
    OWNER IS FABRIQUE ;
    ORDER IS PERMANENT SORTED BY DEFINED KEYS.
    MEMBER IS FABRICATION ;
         INSERTION IS AUTOMATIC
         RETENTION IS MANDATORY ;
         KEY IS ASCENDING NP IN FABRICATION
               DUPLICATES ARE NOT ALLOWED
               NULL IS NOT ALLOWED ;
    SET SELECTION IS THRU FABFAB OWNER
         IDENTIFIED BY IDENTIFIER NF IN FABRIQUE.
```

Remarques sur le schéma de B2 :

- Les identificateurs de chaque "record" sont explicitement définis. Pour les records FABRIQUE, EMPLOYE et PRODUIT ces identificateurs servent de clé d'accès direct par transformation de clé en adresse.

- La liaison entre FABRIQUE et EMPLOYE, "set" FABEMP permet de classer les employés par FABRIQUE et à l'intérieur d'une FABRIQUE par ordre alphabétique sur les noms. Le lien entre une fabrique et un employé est "lache" (MANUAL/OPTIONAL) en ce sens qu'un employé peut exister dans la base sans être rattaché à une fabrique.

- Les liaisons FABRIQUE-FABRICATION et PRODUIT-FABRICATION sont par contre "fortes" (AUTOMATIC/MANDATORY).

- Pour tous les sets, l'accès à une occurrence se fait par l'accès à une occurrence du record propriétaire (SET SELECTION).

- Par souci de simplification, les singular sets "SYS-FAB" et SYS-PRO ne sont pas décrits.

En accord avec le tableau 3.1 et compte tenu d'une analyse sémantique du schéma CODASYL, on peut dresser la vue relationnelle suivante :

FABRIQUE (NF, NOMF, ADRF, BUDGEF)
EMPLOYE (MATRICULE, NOM, AGE, SALAIRE*, NF)
PRODUIT (NP, NOMP)
FABRICATION (NF, NP, QTE).

*SALAIRE est un attribut qui n'apparaît pas dans le record EMPLOYE. En fait on a dans ce record NBH et TAUX qui représentent respectivement le nombre d'heures et le taux horaire. Donc SALAIRE = NBH * TAUX et ce calcul sera fait dans les programmes locaux d'accès à la relation EMPLOYE. La contre-partie de cette possibilité offerte par la vue relationnelle est que toute mise à jour du salaire doit être interdite à moins que l'on sache à partir d'une valeur s ϵ SALAIRE retrouver le nombre d'heures et le taux.

D'autres différences apparaissent ici par rapport à une vue relationnelle construite sur une base SOCRATE.
Elles concernent en particulier l'ordre dans lequel sont rangés les objets locaux. Alors qu'avec SOCRATE c'est le système qui décide de l'ordre dans lequel il range les occurrences, ici l'administrateur indique explicitement

dans le schéma comment ces occurrences doivent être rangées (ORDER IS ..). Ceci est valable aussi bien pour des records pris individuellement que pour des occurrences de membres appartenant à un set.

Donnons à titre d'exemple le programme local qui correspond au parcours séquentiel de tous les EMPLOYEs. Ce parcours séquentiel correspond à une énumération des employés classés par fabrique grâce au set FABEMP.
La correspondance entre ce programme et la base de données CODASYL se fait par une zone de communication ZC tandis que la correspondance entre le programme local et celui qui l'a appelé se fait par une zone externe ZE (Figure 3.5).

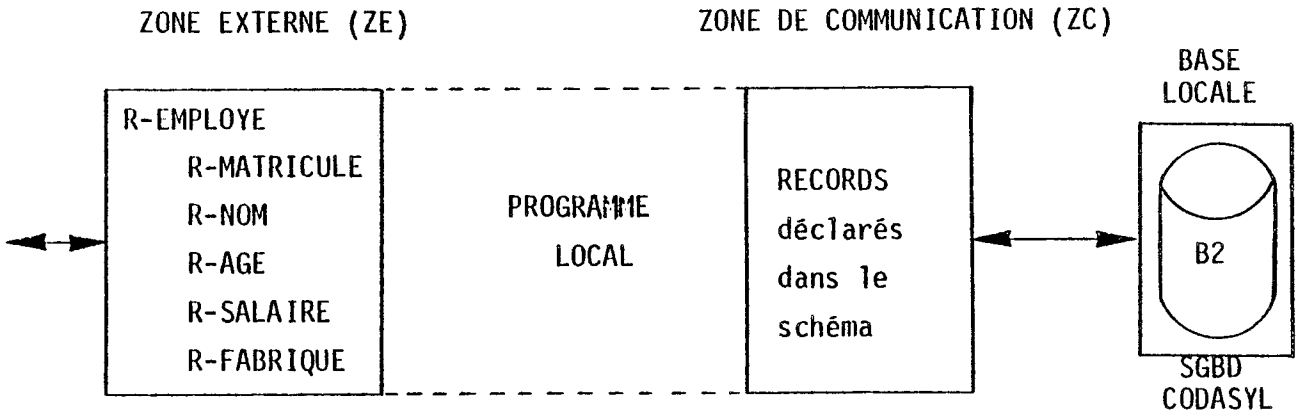


Figure 3.5 - Interface entre un SGBD Codasyl et un programme local

Le parcours séquentiel de tous les employés se fait donc de la manière suivante :

- parcours séquentiel de toutes les fabriques grâce au set singulier "SYS-FAB"
- pour chaque occurrence de fabrique, parcours séquentiel de tous les employés grâce au set FABEMP.

```
Programme PASFAB sortie MATRICULE × NOM × AGE × SALAIRE × NF

          FIND FIRST FABRIQUE IN SYS-FAB (1ère occurrence de fabrique)
BCLFAB.   IF ENDSET (SYS-FAB) GOTO FIN
          GET FABRIQUE
          MOVE NF TO R-FABRIQUE (Numéro dans zone externe)
          IF FABEMP EMPTY GOTO BCLFAB
SUIVEMP.  FIND NEXT EMPLOYE IN FABEMP.
          IF ENDSET (FABEMP) GOTO FABSUIV.
          GET EMPLOYE.
```

```
        MOVE MATRICULE TO R-MATRICULE.
        MOVE NOM TO R-NOM.
        MOVE AGE TO R-AGE.
        MULTIPLY NBH BY TAUX GIVING R-SALAIRE. (Calcul du Salaire)
        (envoyer un n-uplet R-EMPLOYE)
        GOTO SUIVEMP.
FABSUIV. FIND NEXT FABRIQUE IN SYS-FAB. (fabrique suivante)
        GOTO BCL FAB.
FIN.    (envoyer fin de relation).
```

N.B. ENDSET (nom de set) est une expression renvoyant une valeur logique si l'occurrence du set en question a été entièrement parcourue.

Les sorties du programme sont soit un n-uplet soit l'indication "fin de relation" (cf. § 5.5.4).

On considère enfin une base B3 implantée sous I.M.S et comprenant le catalogue des PRODUITs :

PRODUIT (NP, NOMP, DESCR, PRIVEN, PRIREV).

Nous faisons l'hypothèse que ces trois bases ont été construites au sein d'une même grosse entreprise gérant des usines, des employés et des produits. Les usines sont réparties dans plusieurs régions, mais le traitement informatique se fait dans trois centres de calcul :

- à *PARIS* est implantée sous SOCRATE la base B1. Cette base gère un ensemble d'usines localisées dans le nord de la France
- à *LYON*, il existe la base B2 implantée sous un système CODASYL et qui concerne les usines localisées au sud de la France
- à *TOULOUSE* se trouve la base B3 implantée sous I.M.S et qui gère en particulier la politique Produit de l'entreprise.

Après une période d'exploitation indépendante de ces trois bases, les dirigeants de l'entreprise décident de les faire coopérer pour mettre en place des applications qui permettent de mieux coordonner la politique Produit avec l'activité de fabrication.:

Il doit être possible d'effectuer les traitements suivants :

- connaître les usines qui fabriquent des produits dont le prix de revient est supérieur à une certaine valeur
- augmenter la capacité de production des usines dont le budget est supérieur à une certaine valeur et qui fabriquent des produits d'une gamme donnée

- répercuter la création dans le catalogue d'un nouveau produit au niveau de sa fabrication par plusieurs usines de B1 et de B2.
  etc ....

Il s'agit là d'un exemple de travail qui permet d'illustrer les principaux concepts des bases réparties. Cependant, il correspond à deux grands types de situations qui se rencontrent fréquemment dans la pratique :

- plusieurs bases de structures identiques mais au contenu différent (B1 et B2)
- des bases de structures différentes mais se complétant au niveau des informations décrites (B1 et B3, B2 et B3).

Nous reviendrons sur cet exemple au chapitre suivant.

CHAPITRE 4

VUE RELATIONNELLE D'UNE BASE RÉPARTIE : MOGADOR AU NIVEAU GLOBAL

Chacun est à chaque instant mené par ce qu'il voit le plus nettement, composé avec ce qu'il voit le moins clairement.

P. VALERY

Sommaire

4.3.2.1. Relations locales et globales.

4.3.2.2. Règles globales :

- règles non standard
- règles standard.

4.3.2.3. Exemple de Vue Globale.

4.3.3. Relations calculées.

4.3.4. Opérations sur les relations globales : problème de décomposition.

4.3.4.1. Duplication (en un ou plusieurs exemplaires).

4.3.4.2. Obtenir tout ou partie de relation globale

- projection et restriction
- composition.

4.3.4.3. Modification d'une relation globale

- insertion ($G \cup M$)
- suppression ($G - M$)
- mise à jour ($(G-M_1) \cup M_2$)
- résumé.

4.3.4.4. Modification au niveau local : cohérence local/global.

4.3.5. Expression de la correspondance Global/Local.

4.3.6. Processus de décomposition.

4.3.6.1. Optimisation de requêtes relationnelles.

4.3.6.2. Graphe Global.

4.3.6.3. Prise en compte de la localisation.

4.3.6.4. Stratégie de découpage.

4.1. NOTION DE VUE GLOBALE D'UN ENSEMBLE DE DONNEES REPARTIES

Nous prenons comme hypothèse de travail à ce niveau l'existence de n vues relationnelles V1,V2,...,Vn décrivant chacune des ensembles d'informations distincts mais possédant entre eux des liens sémantiques.

Le problème qui se pose alors est de construire une nouvelle vue relationnelle V qui regroupe l'ensemble des n vues. V constitue alors la vue globale de V1,V2,...,Vn (Figure 4.1).

Dans le cadre des bases réparties, V1,V2,...,Vn seront des vues locales mais le processus que nous allons décrire est récurrent, en ce sens qu'une vue globale V peut être utilisée à son tour, conjointement à d'autres vues pour la construction d'une nouvelle vue globale V' et ainsi de suite.

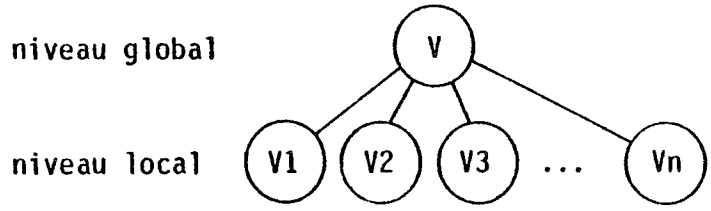


Figure 4.1 - V comme vue globale de n vues locales

Sur le plan de la mise en oeuvre, chaque vue locale correspond en fait à une machine locale MOGADOR, tandis qu'au niveau global, on est amené à définir une machine globale capable de gérer le fonctionnement des n machines locales.

Quels sont les problèmes posés par la construction d'une vue globale V de n vues locales V1,V2,...,Vn ?

- définir un espace de noms globaux en exprimant la correspondance entre ces noms et les noms locaux.
- définir la façon de traiter les objets locaux (simples ou composés) pour en faire des objets globaux, dans le cas d'opérations d'accès allant du local vers le global.
- définir la manière de répercuter des opérations de modification s'adressant au niveau global sur les ensembles d'objets locaux.

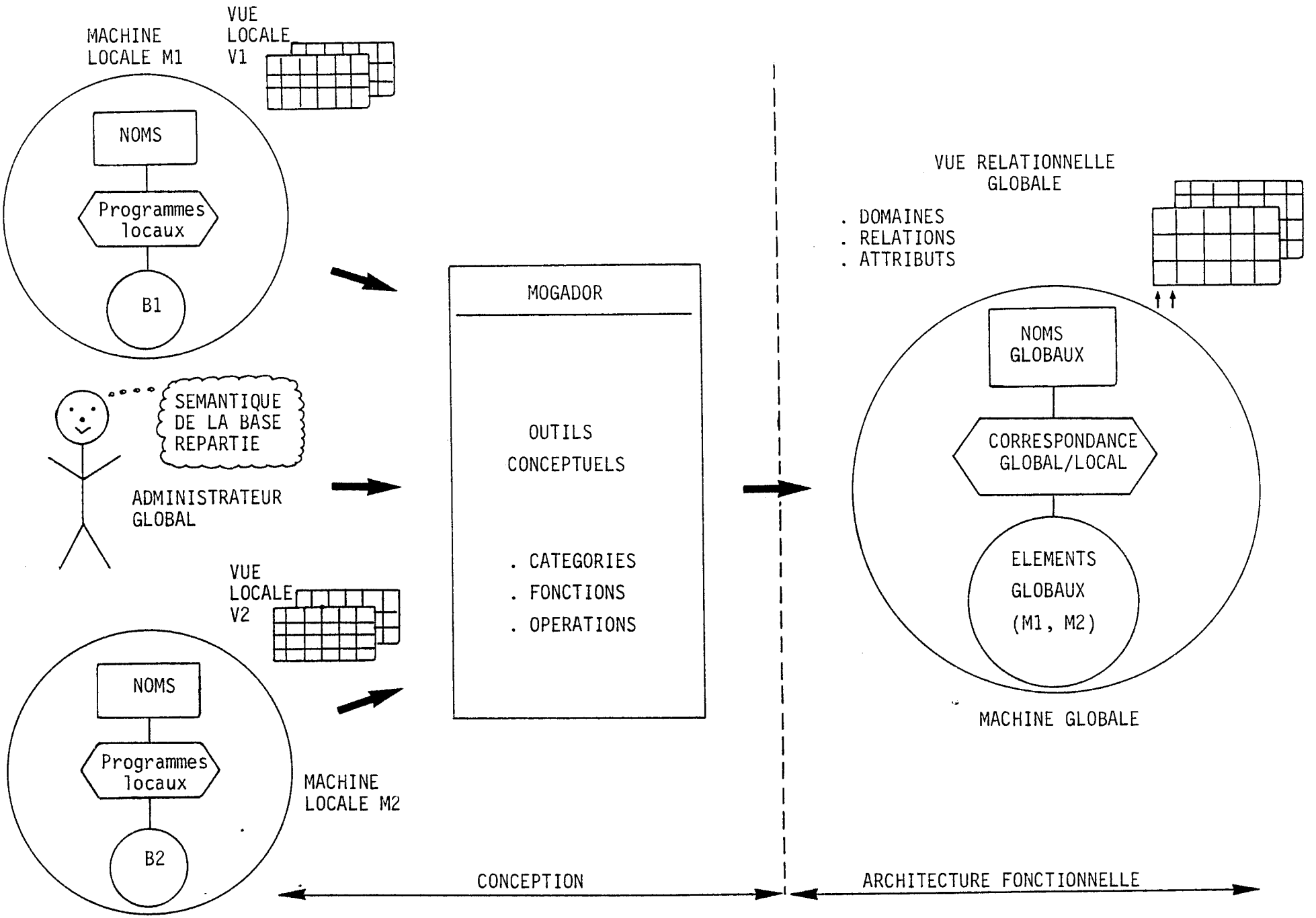


Figure 4.2 - MOGADOR au niveau global

Ces trois problèmes sont, bien entendu, liés à la sémantique des données réparties, dont a connaissance l'administrateur global.

Nous allons essayer de montrer comment les outils conceptuels fournis par MOGADOR permettent à cet administrateur de concevoir la vue globale et la machine correspondante (Figure 4.2).

Pour la construction de la base répartie, nous nous plaçons ici dans une démarche ascendante (cf. § 1.3.4) propre à l'optique de coopération que nous avons choisie.

Afin d'illustrer les contrastes avec l'approche descendante, considérons l'exemple suivant :

4.1.1. Approche descendante

Considérons la relation EMPLOYE (E#, NOM, AGE, SALAIRE, DPT, DIR) décrivant pour chaque employé identifié par un numéro (E#), son nom (NOM), son âge (AGE), son salaire (SALAIRE), le département où il travaille (DPT) et son directeur (DIR construit sur le même domaine que E#).

Dans l'approche descendante, cette relation est une relation globale et une possibilité de répartition est de la découper en trois fragments F1, F2 et F3 comme indiqué par la Figure 4.3.a. Ce découpage est en fait un partitionnement sans duplication (cf. § 1.3.2) qui permet un contrôle total de la répartition dans la mesure où chacun des fragments est ensuite stocké dans une base locale.

4.1.2. Approche ascendante

Le point de départ serait, par exemple, trois relations R1, R2 et R3 (Figure 4.3.b) :

R1 (E#, NOM, AGE)

R2 (E#, NOM, AGE)

R3 (E#, NBH, TAUX, DPT, DIR)

se trouvant chacune dans une base locale, mais concernant des employés appartenant à un même organisme.

L'approche ascendante, illustrée par la Figure 4.3.b consiste alors à définir, à partir de R1, R2, R3, une relation globale EMPLOYE :

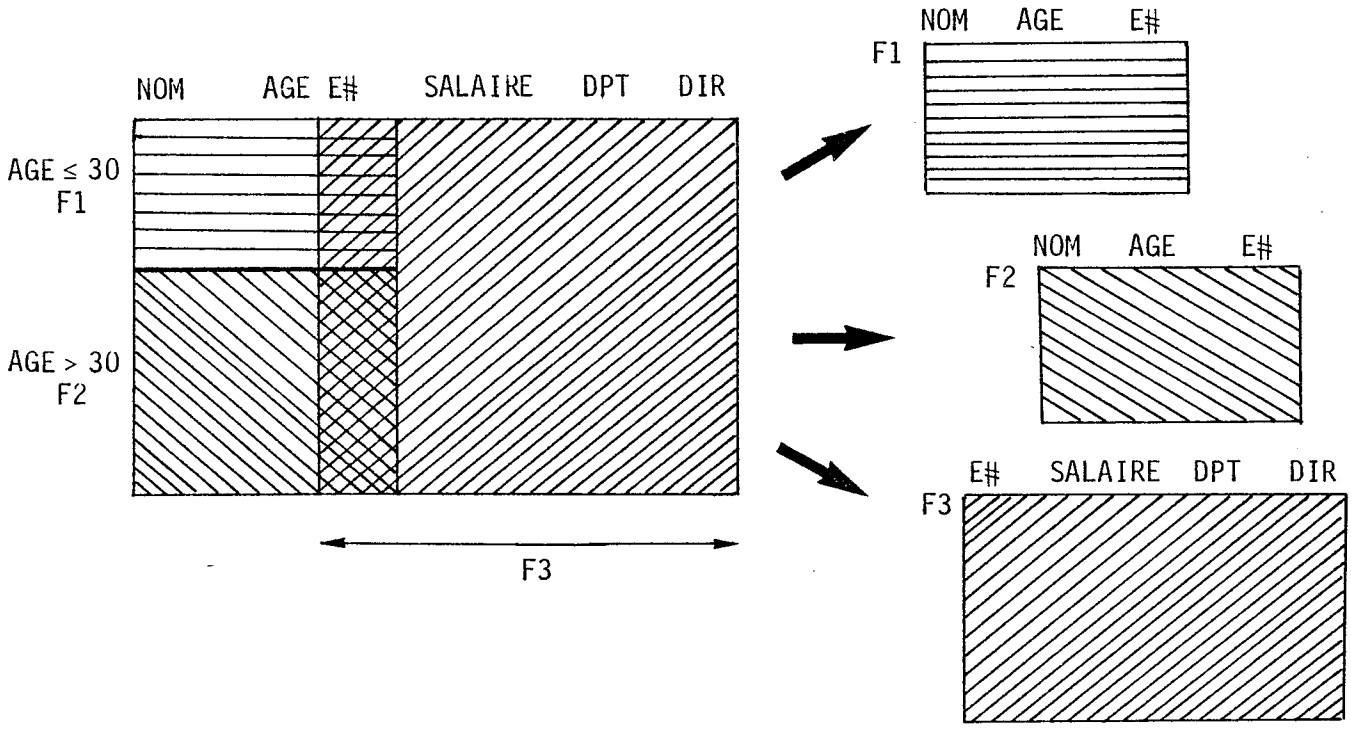


Figure 4.3.a - Approche descendante

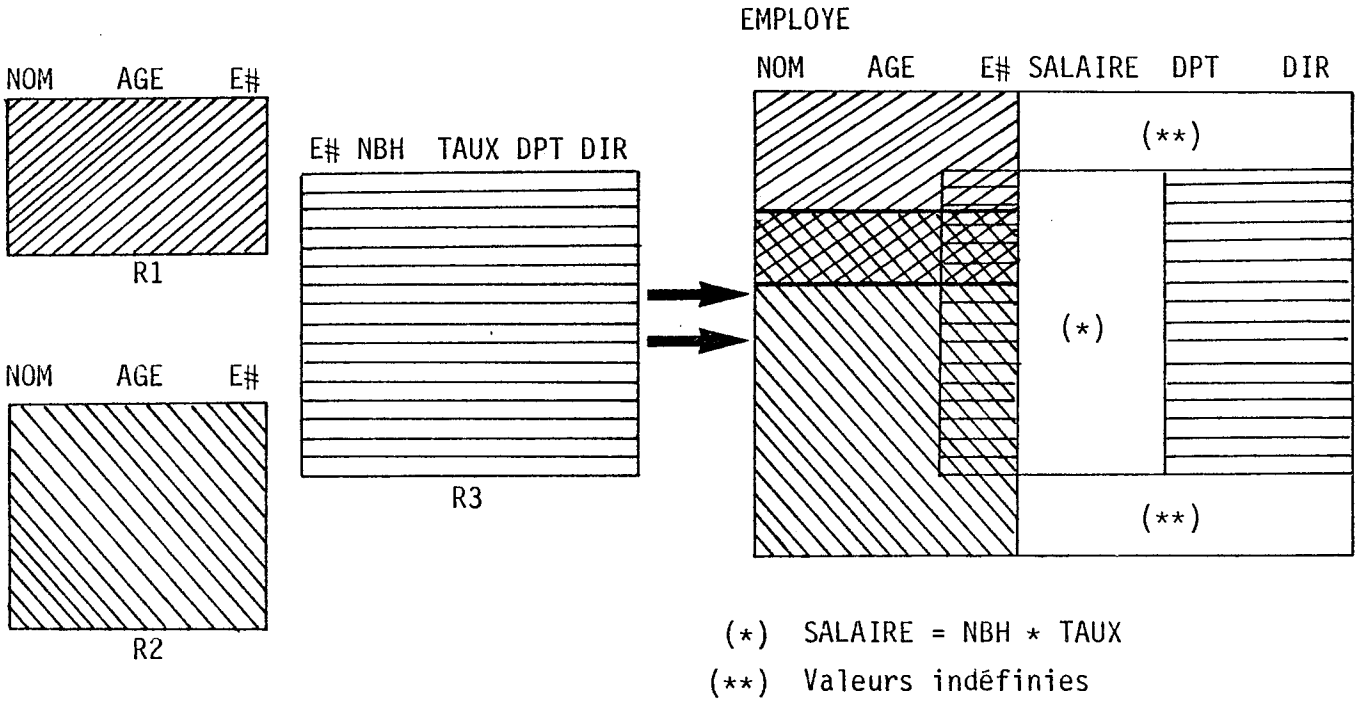


Figure 4.3.b - Approche ascendante

EMPLOYE (E#, NOM, AGE, SALAIRE, DPT, DIR).

Les attributs de EMPLOYE sont alors :

- soit des attributs existant déjà dans R1, R2, R3 : E#, NOM, AGE, DPT, DIR
- soit des attributs correspondant à des valeurs calculées à partir de celles existantes dans R1, R2, R3 : SALAIRE comme produit d'un nombre d'heures (NBH) par un taux horaire (TAUX).

Dans ce type d'approche, on rencontre également deux problèmes importants :

- les duplications éventuelles (par exemple, n-uplets communs à R1 et R2) avec les incohérences que cela peut entraîner, par exemple le même employé décrit deux fois avec des valeurs différentes pour un attribut (AGE)
- les valeurs indéfinies de certains attributs (c'est pour cette raison que la valeur "#" a été introduite au § 2.5).

4.2. CONCEPTION D'UNE VUE GLOBALE

Comme nous l'avions souligné en présentant MOGADOR, les éléments des catégories concrètes et ceux des graphes de fonctions constituent les *atomes sémantiques de répartition*. En effet, il s'agit d'éléments indécomposables, sans perte d'information.
De ce fait, les catégories concrètes et les graphes des fonctions sont les ensembles sur lesquels s'appliquent les deux notions liées à la répartition, à savoir partitionnement et duplication (cf. § 1.3.2).

L'approche ascendante, dans la conception d'une vue globale V de n vues locales V1,V2,...,Vn, nous oblige à déterminer les liens sémantiques existant entre les bases locales. Ces liens vont servir à construire des *catégories* et des *fonctions globales* tout en indiquant la correspondance avec catégories et fonctions locales sur les deux plans suivants :

- au niveau des noms : localisation sur les noms
- au niveau des éléments : localisation sur les éléments (correspondance Nom-Eléments et règles globales).

4.2.1. La répartition des informations

- Considérons l'ensemble E :

$$E = \{e1, e2, \ldots, ep\}$$

dont les éléments sont des objets composés (identificateurs de catégories concrètes ou éléments de graphe de fonctions).

- Considérons également un ensemble de machines M

$$M = \{m1, m2, \ldots, mq\}.$$

Chaque machine est capable de "stocker" dans son espace des éléments tout ou partie de E. La liaison ainsi caractérisée entre E et M s'exprime au moyen de la fonction "loc" :

$$E \xrightarrow{loc} M$$

Selon la nature de cette fonction, nous allons examiner les différents cas possibles de répartition.

4.2.1.1. Définition du degré de répartition

Etant donné un ensemble d'éléments E et un ensemble de machines M, nous appellerons degré de répartition de E (noté degrep(E)) le nombre de machines contenant des éléments de E.

Dans MOGADOR degrep est le nom d'une méta-fonction monovaluée :

$$CA \cup CC \cup FMONO \cup FMULTI \xrightarrow{degrep} ENTIER\ (\geq 0).$$

4.2.1.2. Différents cas de répartition

L'ensemble E que nous considérons est un ensemble global en ce sens que ses éléments se situent au niveau global.

4.2.1.2.1. Répartition Triviale (degrep (E) = 1)

L'ensemble E dans sa totalité est stockée dans une seule machine :
$(loc \in FMONO) \wedge (\exists\, m \in M\ loc(E) = m) \wedge (\forall m' \in M-\{m\},\ \underline{inv}\text{-}loc(m') \cap E = \emptyset)$.

Dans ce cas, l'ensemble global et l'ensemble local coïncidant.

4.2.1.2.2. Partitionnement (degrep (E) ≤ q)

L'ensemble E est découpé en sous-ensembles disjoints,chacun étant affecté à une et une seule machine :

$(loc \in FMONO) \wedge (\forall E_i \subseteq E, \exists m_i \in M \cdot loc(E_i) = m_i) \wedge$
$(\forall m_j, m_k \in M, m_j \neq m_k \Rightarrow \underline{inv}\ loc(m_j) \cap \underline{inv}\ loc(m_k) = \emptyset).$

L'ensemble global est l'union d'ensembles locaux disjoints.

4.2.1.2.3. Duplication totale (degrep (E) = q)

L'ensemble E est recopié dans son ensemble sur chacune des machines :

$(loc \in FMULTI) \wedge (loc(E) = M).$

L'ensemble global est l'un des ensembles locaux.

Les deux cas précédents sont en fait les deux extrêmes d'un spectre continu de cas pour lesquels partitionnement et duplication se mélangent.

4.2.1.2.4. Partitionnement avec Duplication (1 < degrep(E) ≤ q)

L'ensemble E est découpé et partiellement dupliqué dans plusieurs machines :

$(loc \in FMULTI) \wedge (\exists M' \subseteq M \text{ tel que } E = \bigcup_{m' \in M'} \underline{inv}\ loc(m'))$

Il existe un ensemble M' de machines telles que l'union des éléments qu'elles contiennent forme l'ensemble E.

Dans ce cas, l'ensemble global est l'union d'ensembles locaux non disjoints.

4.2.1.2.5. Cas particulier : ensemble calculé (degrep (E) = 0)

Si le degré de répartition est nul, cela signifie qu'il n'existe aucune machine contenant les éléments de E. De ce fait, ces éléments ne peuvent être obtenus que par calcul, à partir d'autres éléments qui sont eux-mêmes stockés dans une ou plusieurs machines.

4.2.2. Catégories Abstraites Globales (CAG)

Elles permettent en particulier de caractériser les liens sémantiques entre les bases locales. Si plusieurs catégories abstraites locales correspondent au même ensemble de valeurs, elles peuvent être regroupées pour former une même catégorie abstraite globale. Ce regroupement est en fait l'union des ensembles (abstraits) de valeurs.

Par exemple, soit dans M1 la CAL NOM1 et dans M2 la CAL NOM2, on définit la CAG NOM de la manière suivante :

cag NOM := NOM1 ∪ NOM2 avec degrep(NOM) = 2

ou en d'autres termes : M1.NOM = NOM1

M2.NOM = NOM2.

Si pour les besoins de la vue globale, on est amené à construire de nouvelles valeurs n'appartenant à aucune des CAL, on a alors des CAG de type calculé, dont le degré de répartition est nul.

La correspondance entre CAG et CAL est assurée par deux méta-fonctions :

1°) Méta-fonction multivaluée "Loccag" :

$$CAG \xrightarrow{loccag} MACHINE \times CAL$$

Ainsi, Loccag (NOM) = {(M1,NOM1), (M2,NOM2)}

et Loccag (A) = ∅ dans le cas d'une CAG calculée A.

2°) Méta-fonction monovaluée "nomloccag" :

$$MACHINE \times CAG \xrightarrow{nomloccag} CAL$$

nomloccag (M1,NOM) = NOM1.

4.2.3. Catégories Concrètes Globales (CCG)

Parmi les différentes bases locales, il existe des ensembles objets que l'on peut, selon les besoins de la nouvelle classe d'applications, regrouper en une seule et même catégorie concrète globale (CCG).

Par exemple, des employés dans une base B1 et des professeurs dans une base B2 peuvent être regroupés dans une CCG : PERSONNE. A chaque catégorie locale qui compose une catégorie globale, sont rattachées plusieurs fonctions (attributs, identificateurs). De la même façon, il faudra définir au niveau global des fonctions globales et les exprimer au moyen

des fonctions locales.

Au niveau global, définir une catégorie concrète c'est définir un nom appartenant à l'ensemble CCG en précisant son identificateur.

- Identificateur de catégories globales (NIG)

Un identificateur de catégorie concrète globale (nig) est un produit cartésien de catégories abstraites globales. Ce produit est tel qu'un élément lui appartenant permet de caractériser un et un seul élément de la catégorie concrète globale concernée.

Pour définir l'identifiant d'une catégorie globale, on dispose non seulement des catégories abstraites déclarées dans la vue globale, mais aussi de la catégorie prédéfinie MACHINE.

On peut ainsi définir l'identificateur global de manière à rendre différents tous les objets composés provenant des bases locales, ou au contraire de manière à détecter les éventuelles duplications.

Exemples :

- Dans M1, considérons une base de données comprenant des ENSEIGNANTS identifiés localement par un numéro d'INSEE
- Dans M2, une base comprenant des ETUDIANTS identifiés également par un numéro d'insee appelé MATRICULE.

(M1,INSEE) et (M2,MATRICULE) correspondent au même ensemble de valeurs, donc si INSEE est une CAG, on a

$$\text{Loccag (INSEE)} = \{(M1,\text{INSEE}), (M2,\text{MATRICULE})\}.$$

Soit PERSONNE la CCG regroupant les éléments de M1 (les ENSEIGNANTs) et ceux de M2 (les ETUDIANTs). Distinguons deux cas :

Cas 1 : Si PERSONNE est identifiée par :

$$\text{MACHINE} \times \text{INSEE}$$

alors nous pouvons dire que toutes les PERSONNES sont distinctes.

Cas 2 : Par contre, si PERSONNE est identifiée par INSEE, alors on aura un moyen pour déterminer les ENSEIGNANTs de M1 qui sont en même temps ETUDIANT dans M2.

Dans le cadre d'une implantation machine, ce dernier cas peut s'avérer extrêmement coûteux à mettre en place, voir dans certains cas impossible (catégories réparties sur de nombreuses bases et dont les objets sont très nombreux).

Remarquons cependant que les objets produits par les processus locaux voyagent vers la machine globale avec un certain nombre d'informations concernant leur catégorie et surtout leur provenance.

Ainsi, dans tous les cas, on se retrouvera dans une situation analogue au cas 1 de l'exemple ci-dessus. Le cas 2 correspond alors à la suppression de certaines informations de provenance, la difficulté restant alors de trier, parmi les ensembles d'objets qui proviennent de bases différentes, les objets identiques. Cette opération devra être demandée explicitement par l'utilisateur de la vue globale (analogie avec option "UNIQUE" d'un langage relationnel comme SEQUEL). Dans ce cas, l'utilisateur doit être informé du coût d'une telle opération.

Nous utilisons l'opérateur ccg pour générer des éléments de l'ensemble des noms de catégories concrètes globales CCG et l'opérateur nig pour générer l'identificateur global correspondant.

Exemple : ccg PERSONNE nig MACHINE × INSEE.

La correspondance entre noms locaux et noms globaux pour les catégories concrètes s'exprime au moyen des deux méta-fonctions suivantes :

1°) Méta-fonction multivaluée "Locccg"

$$CCG \xrightarrow{Locccg} MACHINE \times CCL$$

2°) Méta-fonction monovaluée "nomlocccg"

$$MACHINE \times CCG \xrightarrow{nomlocccg} CCL.$$

4.2.4. Fonctions Globales (FG = FUG ∪ FPG)

Les fonctions globales correspondent aux liaisons entre catégories globales. En accord avec les cas de répartition décrits ci-dessus, à une fonction globale il correspond zéro, une ou plusieurs fonctions locales.

La correspondance au niveau des noms s'établit grâce aux deux méta-fonctions suivantes :

1°) Méta-fonction multivaluée "Locfg"

$$FG \xrightarrow{Locfg} MACHINE \times FL$$

2°) Méta-fonction monovaluée "nomlocfg"

$$MACHINE \times FG \xrightarrow{nomlocfg} FL$$

Le tableau 4.1 résume les principales méta-catégories et méta-fonctions des niveaux local et global, en indiquant comment s'établissent conceptuellement les correspondances entre les niveaux local et global d'une part et entre noms et éléments d'autre part.

ESPACES / NIVEAU	NOMS	CORRESPONDANCE NOMS - ELEMENTS	ELEMENTS
LOCAL	CAL Cat. Abstraites CCL Cat. Concrètes NIL Noms Identificat. FUL Fonctions monoval. FL FPL Fonctions multival.	Opérateurs s'adressant à une machine locale	. OSL objets simples locaux . OCL objets composés locaux . IDL identificateurs locaux . PL programmes locaux
CORRESPONDANCE LOCAL / GLOBAL	Localisation sur les Noms : CAG → Loccag ↠ MACHINE×CAL CCG → Locccg ↠ MACHINE×CCL FG → Locfg ↠ MACHINE×FL	Processus de Décomposition	Localisation sur les éléments Règles Globales sur : - les catégories concrètes - les fonctions
GLOBAL	CAG Cat. Abstraites CCG Cat. Concrètes NIG Noms d'Identificat. FUG Fonctions Monoval. FL FPG Fonctions Multival.	Opérateurs s'adressant à une machine globale	. OSG objets simples globaux . OCG objets composés globaux . IDG identificateurs globaux . PG programmes globaux

Table 4.1 - Méta-Catégories et Méta-Fonctions du niveau local et du niveau global

4.2.5. Opérations sur Catégories et Fonctions Globales

En tenant compte des possibilités de répartition des catégories et des fonctions globales, il faut étudier comment conceptuellement les opérations primitives appliquées au niveau global se répercutent au niveau local.

Le tableau 4.2 donne pour chaque opération primitive globale le nombre d'opérations locales correspondantes. Si ce nombre est plus grand que un, l'opération globale correspond alors à un ensemble d'opérations locales fournissant chacune un résultat. Selon les cas, il faut appliquer à l'ensemble des résultats un nouvel opérateur pour avoir le résultat global. Cet opérateur est indiqué entre parenthèses après le nombre.

Par exemple, la création d'un élément global dans le cas d'une duplication totale a la valeur vraie si et seulement si toutes les créations locales ont été effectuées (d'où l'opérateur d'intersection).

De la même façon, l'énumération d'une catégorie globale partitionnée correspond à l'union des ensembles fournis par les q énumérations locales.

Dans le cas 4, le nombre k est compris entre 1 et q et correspond à la cardinalité de l'ensemble M' défini au paragraphe précédent. Un des principaux problèmes est alors de déterminer cet ensemble M' qui, la plupart du temps, ne peut être connu que par l'administrateur global.

Il faut remarquer la distinction faite à propos des fonctions mono et multivaluées concernant les opérations de liaisons et d'accès.

En effet, dans le cas d'une fonction multivaluée f partitionnée, les éléments de l'ensemble lié à un élément x (soit f(x)) peuvent se trouver à cheval sur deux machines :

Par exemple :

X	x	x	x	x	x	y	y
A	a1	a2	a3	a4	a5	a3	a4
		M1			M2		

Dans ce cas, l'opération accéder (f,x) doit être exécutée par M1 et M2 et les résultats fournis doivent être réunis en un seul et même ensemble.

Notation :
Etant donné : n machines M1,M2,...,Mn contenant respectivement des ensembles L1,L2,...,Ln qui représentent tout ou partie d'un ensemble global E, on notera Mi.E la portion de E se trouvant sur Mi (Mi.E = Li).

4.2.6. Passage à une Vue Relationnelle Globale

Concevoir une vue globale V par intégration de n-vues locales V1,V2,...,Vn (chacune associée à une machine locale M1,M2,...,Mn) c'est d'abord revenir à un niveau conceptuel pour caractériser les ensembles d'objets globaux que l'on veut considérer, et les liaisons entre ces ensembles.

C'est ensuite exprimer la correspondance entre ces ensembles d'objets globaux et les objets locaux déjà existants.
Or, les objets locaux sont finalement regroupés dans les vues locales comme des ensembles de n-uplets. De ce fait, la vue globale va également s'exprimer au moyen de relations n-aires construites sur les relations locales (Figure 4.2).

La vue globale V comprend donc un ensemble de domaines globaux et de relations globales :

- Soit RL^i l'ensemble des relations locales de la vue locale offerte par la machine Mi.
- Soit RG l'ensemble des relations globales.

Cet ensemble se compose de deux parties :

1) L'ensemble RG_R des relations globales, dont les éléments sont répartis dans les bases locales (degré de répartition ≥ 1). $R \in RG_R$ s'exprime en fonction de $\{L^i \in RL^i\}$.

2) L'ensemble RG_C, dont les éléments sont calculés à partir de valeurs locales (degré de répartition = 0). Nous considérons que cet ensemble est formé uniquement de relations binaires (cf. § 4.3.3).

	1 répartition triviale	2 partitionnement	3 duplication totale	4 partitionnement/duplication
- Sur Catégorie Concrète				
{ création suppression }	1	1	q (intersection)	k (intersection)
énumération	1	q (union)	1	k (union)
test	1	q (union)	1	k (union)
- Sur Fonctions				
graphe	1	q (union)	1	k (union)
{ lier mono délier mono }	1	1	q (intersection)	k (intersection)
{ lier multi délier multi }	1	ℓ (intersection)	q (intersection)	k (intersection)
accès mono	1	1	1	1
accès multi	1	ℓ (union)	1	k (union)

Table 4.2 - Correspondance entre opérations globales et locales

En nous intéressant tout d'abord aux relations réparties, nous allons reprendre les cas de répartition et étudier les critères de découpage d'un ensemble d'objets composés en plusieurs autres ensembles.

Ceci nous permettra de voir comment localiser un élément, c'est-à-dire trouver la machine qui le contient (ou qui le contiendra dans le cas d'une création).

4.2.7. Critères de découpage et localisation des éléments

On considère un ensemble E d'objets composés :

$$E = \{e = (e1,e2,\ldots,en)\}.$$

Un tel ensemble peut être soit un ensemble d'identificateurs d'une catégorie concrète, soit un ensemble d'éléments d'un graphe de fonction, soit, dans le cas général, un ensemble de n-uplets construits de la manière indiquée au § 2.4.2 (Conception d'une relation provenant d'une catégorie concrète avec un ensemble de fonctions monovaluées ayant pour source cette catégorie).

A chaque ensemble E, on associe une fonction de localisation notée Loc_E et élément de la méta-catégorie :

$$F_{LOC} = \{f \mid (f \in FMONO \cup FMULTI) \wedge (cible(f) = MACHINE)\}.$$

Selon les critères de découpage, cette fonction loc_E aura une catégorie source différente.

Remarquons cependant qu'un élément e de E est indécomposable en ce qui concerne la répartition : il est stocké en entier dans une machine.

4.2.7.1. Répartition triviale et critère de localisation directe

A la répartition triviale, correspond un critère de localisation qui porte sur le nom de l'ensemble : ce nom est en correspondance directe avec un nom de machine (et un seul) qui contient tous les éléments de E.

La fonction loc_E a alors pour source l'ensemble vide et est une fonction constante, par exemple :

$$\emptyset \xrightarrow{Loc_E} M3.$$

4.2.7.2. Partitionnement par restriction et localisation directe

Les sous-ensembles disjoints de E sont déterminés par restriction de cet ensemble. Cette restriction porte sur les valeurs composant les objets de E.

Exemple : Soit un ensemble de PERSONNE identifiées par un numéro INSEE et ayant trois attributs, le NOM, l'AGE, le SALAIRE.

Un critère de partitionnement dans deux machines M1 et M2 pourrait par exemple porter sur le SALAIRE :

sur M1 : PERSONNE (SALAIRE ≤ 10000)

sur M2 : PERSONNE (SALAIRE > 10000).

Chaque restriction est une expression prédicative et dans le cas d'un partitionnement, il faut que ces expressions soient disjonctives.

Ainsi la fonction de localisation a comme source un ensemble de valeurs des domaines sur lesquels sont formés les éléments de E.

Ainsi : $loc_{PERSONNE} \in FMONO \wedge source\ (loc_{PERSONNE}) = SALAIRE$.

Nous disons que ce type de localisation est directe car une ou plusieurs valeurs d'un objet permettent d'obtenir directement la machine où se trouve (où se trouvera) cet objet.

Soit $FLOC_D$ le sous-ensemble de FLOC qui correspond aux fonctions de ce type.

4.2.7.3. Partitionnement par voisinage et localisation transitive

Dans ce type de partitionnement, l'emplacement d'un élément dépend de celui d'un autre. Par exemple, répartir les employés dans les bases où figure la description de leur usine.

Cela signifie donc qu'il existe une liaison entre les deux éléments et donc une liaison entre leurs ensembles. De plus, puisque nous sommes dans un partitionnement, cette liaison est nécessairement monovaluée.

Au niveau d'un élément e = (e1,...,ep), il existe donc un sous-objet composé e' = $(e_k,\ldots,e_{k+\ell})$ de e tel que $E_k \times E_{k+1} \times \ldots \times E_{k+\ell}$, soit en fait l'identificateur d'une catégorie concrète E'.

Localiser e revient alors à localiser e', c'est-à-dire à utiliser la fonction de localisation associée à E'.

Nous nous limitons ici à une transitivité de niveau 1, c'est-à-dire que si $FLOC_T$ est le sous-ensemble de FLOC qui correspond aux fonctions de localisation transitives, à chaque élément de $FLOC_T$ il correspond un élément de $FLOC_D$.

Exemple :
On considère un ensemble d'EMPLOYE ayant comme attributs un numéro d'INSEE, un NOM, un SALAIRE et le numéro de l'usine où ils travaillent (NUS). Considérons alors un ensemble d'USINEs, identifiées par un numéro NUS et ayant comme attributs une ville (VILLE) et un BUDGET.

Soit : loc_{USINE} ayant pour source BUDGET et telle que le graphe soit donné par l'expression prédicative :

M1 : USINE (BUDGET ≥ 100 kf)

M2 : USINE (BUDGET < 100 kf)

$loc_{USINE} \in FLOC_D$: fonction de localisation directe.

Si l'on veut que les employés soient dans les machines qui décrivent les usines correspondantes, la localisation d'un employé : e = (i,n,s,u) revient à la localisation de l'usine u.

$loc_{EMPLOYE} \in FLOC_T$ et loc_{USINE} est la fonction correspondante.

N.B. Remarquons que dans ce cas, le problème de la détermination de la base qui contient u ∈ NUS n'est pas résolu car le critère de répartition ne porte pas sur la catégorie NUS mais sur BUDGET. Ainsi, il faudra regarder dans toutes les bases contenant des USINES pour trouver celle qui a pour identificateur u.

4.2.7.4. Partitionnement et duplication par restriction

Chaque restriction correspond à une expression prédicative et dans le cas d'un partitionnement, il faut que ces expressions soient disjonctives. Si ce n'est pas le cas, on est en présence d'un découpage qui entraînera des duplications mais on pourra malgré tout disposer d'une localisation directe : une valeur d'un objet permet d'obtenir directement

la ou les machines où se trouve (où se trouvera) cet objet.

Par exemple :

sur M1 PERSONNE (4000 < SALAIRE ≤ 10000)

sur M2 PERSONNE (SALAIRE > 8000).

Ainsi les personnes dont (8000 < SALAIRE ≤ 10000) sont stockées sur M1 et M2.

Dans ce cas, $loc_{PERONNE} \in FMULTI \wedge source\ (loc_{PERSONNE}) = SALAIRE$.

Ce cas donne naissance également à un autre critère de partitionnement : le partitionnement avec duplication par voisinage.

4.2.7.5. Duplication totale et localisation directe

On considère que E est dupliqué dans sa totalité sur un ensemble de machines M_k, M_{k+1}, ..., M_{k+p}. Ainsi la localisation de un élément de e est directe, en ce sens qu'il suffit de considérer l'une des machines.

On a :

$$loc_E \in FMULTI \wedge source\ (loc_E) = \emptyset$$

(loc_E est une fonction constante, multivaluée).

Exemple :

On considère un catalogue de PRODUIT dupliqué sur chaque machine M1, M2, M3.

Remarquons que lorsque E n'est "dupliqué" que sur une seule machine, on retombe sur le cas de la répartition triviale (§ 4.2.7.1).

4.2.7.6. Partitionnement arbitraire, localisation calculée

Dans ce cas, il n'existe aucun critère explicite de partitionnement qui puisse s'exprimer soit par une expression prédicative, soit par un voisinage. Il faut alors envisager des fonctions de localisation "calculées", qui dépendent chacune de critères propres. Ce calcul sera exprimé au moyen de règles globales, explicites (cf. § 4.3.3).

4.3. MACHINE GLOBALE

Nous nous intéressons ici à la définition et au fonctionnement de la machine associée à une vue globale.

Cette machine doit avoir connaissance :

1) des noms globaux définis par la vue globale en tant que vue relationnelle (domaines, relations, attributs).

2) des synonymes locaux de ces noms.

3) de la correspondance entre ensembles d'objets globaux et ensembles locaux.

Grâce à cette connaissance, la machine globale fonctionne de la manière suivante (Figure 4.4) :

- elle prend en compte une transaction globale (interrogation, modification)
- elle décompose cette transaction globale en transactions locales, chacune d'elles ne s'adressant qu'à une seule machine locale
- elle fait exécuter par chaque machine locale la transaction qui lui est destinée et contrôle la synchronisation de ces exécutions pour élaborer le résultat final.

4.3.1. Espace des Noms : expression de la Vue Globale

La vue globale proprement dite, comprenant la définition des noms de domaines, de relations et d'attributs, s'exprime dans le langage de définition (LA.DO.RE, cf. § 2.5), comme une vue relationnelle classique.

La seule différence provient ici de la partie identification qui doit comporter les noms des machines locales dont les vues composent la vue globale (cf. § 5).

Nous donnons un exemple au § 4.3.2.3.

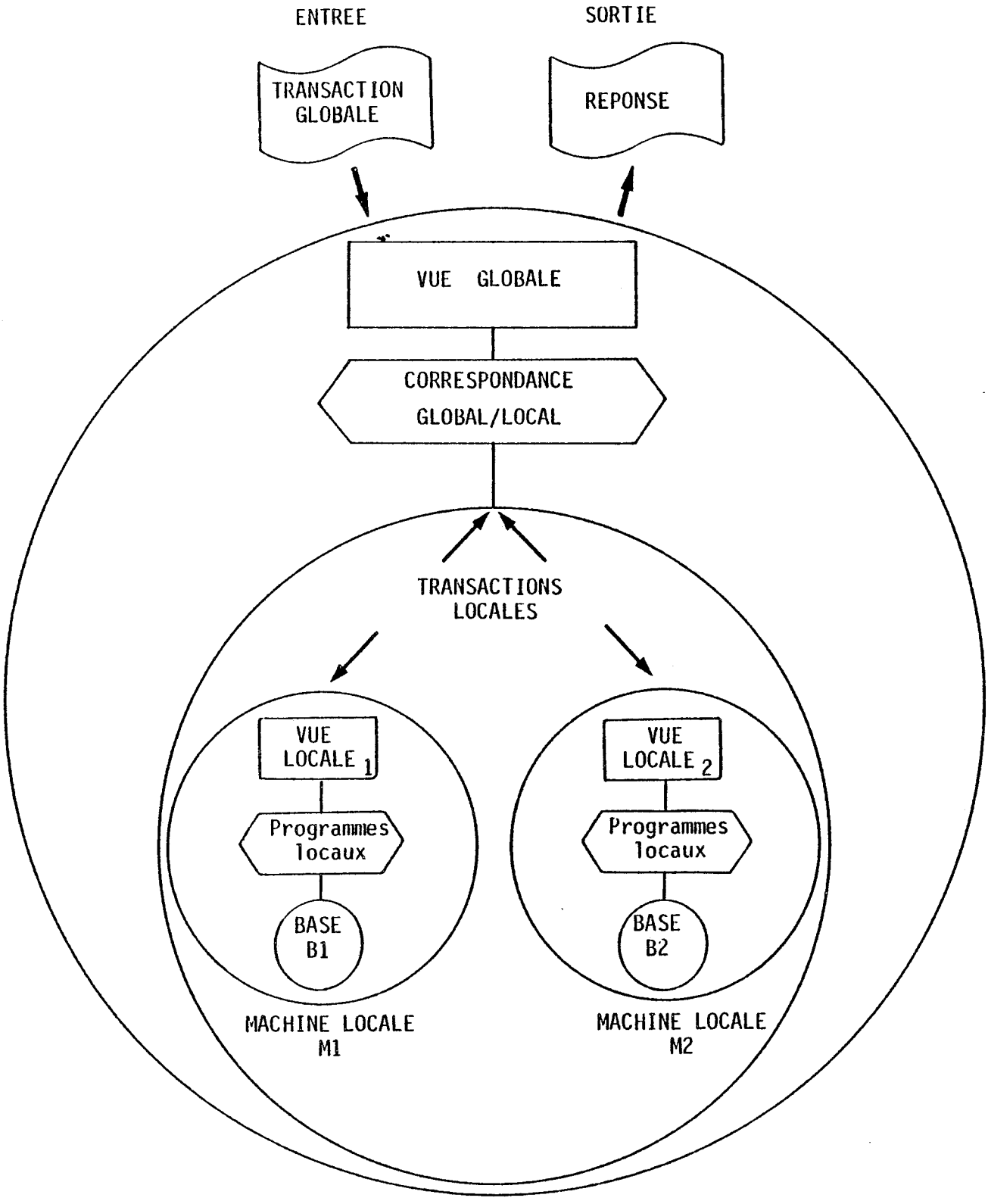


Figure 4.4 - Une Machine Globale autour de deux base locales B1 et B2

4.3.2. Correspondance global/local

C'est bien sûr la partie la plus importante de la machine globale.

Cette correspondance se subdivise logiquement en deux parties :

- la première concerne la correspondance entre noms globaux et locaux
- la seconde concerne la correspondance entre objets locaux et globaux.

A l'évidence, la première partie est une simple synonymie entre chaînes de caractères, alors que la seconde pose le problème de la sémantique des données réparties qui concerne leur comportement vis-à-vis des opérations d'accès et de modification (cf. § 4.2.2).

4.3.2.1. Relations locales et globales

Soit V la vue globale construite sur n vues locales V1,V2,...,Vn.
A chaque Vi est associée une machine Mi $\in$ MACHINE, i=1,2,...,n.

Soit L = $\{L^i(I^i,X^i)$, $\forall i \in LOC$, LOC $\subseteq$ MACHINE$\}$ un ensemble de relations locales : I^i désigne l'attribut clé (éventuellement composé) et X^i l'ensemble des attributs non clés.

La coopération de plusieurs relations locales L en une relation globale G(I,X) n'est possible que si elles possèdent entre elles une sémantique commune.

Nous définissons pour cela les trois conditions suivantes :

1) $\forall i \in LOC \quad I \equiv I^i$

Les relations locales L^i doivent avoir la même clé que la relation globale. I et les ensembles I^i se partagent le même ensemble de valeurs.

2) La répartition des éléments dans les L^i correspond à l'un des cas vus plus haut avec degré de répartition strictement plus grand que zéro :

$$I = \bigcup_{i \in LOC} I^i$$

De manière à pouvoir revenir au niveau de répartition le plus fin fourni par MOGADOR, nous exprimons la relation G par sa décomposition en relations binaires (cf. § 2.4.2).

(1) $G(I,X) \leftarrow G_1(I,X_1) \overset{\ddagger}{*} G_2(I,X_2) \overset{\ddagger}{*} \ldots G_p(I,X_p)$

avec :

- $X = \{X1,X2,\ldots,Xp\}$
- "$\leftarrow$" signifie "définie par"
- $\overset{\ddagger}{*}$ dénote l'opérateur de composition naturelle généralisée (table 2.14).

3) L'élément de graphe de fonction étant également un atome sémantique de répartition, donc indécomposable, chaque relation binaire G_k s'exprime comme une union de relations binaires locales :

(2) $k=1,2,\ldots,p \qquad G_k(I,X_k) = \underset{\forall j \in S_k \subseteq MACHINE}{\cup} [I^j, X_k^j] L^j$

où S_k est un sous-ensemble de machines et où l'on suppose que $X_k \equiv X_k^j$, c'est-à-dire qu'ils partagent le même ensemble de valeurs.

En reportant (2) dans (1), on a une définition "canonique" (produit d'unions) de G en fonction des relations locales.

Cependant, cette définition "canonique" est trop fine et ne tient pas compte des regroupements effectués dans les vues locales.

En effet, considérons $G_k(I,X_k)$ et $G_{k'}(I,X_{k'})$ telles que $S_k = S_{k'} = S_{kk'}$ (l'ensemble de localisation des deux sous-relations est identique).

On a alors :

(3) $G_k(I,X_k) = \underset{\forall j \in S_{kk'}}{\cup} [I^j, X_k^j] L^j$

(4) $G_{k'}(I,X_{k'}) = \underset{\forall j \in S_{kk'}}{\cup} [I^j, X_{k'}^j] L^j$

On notera que les L de G_k et $G_{k'}$ sont identiques.

Il vient :

$$G_k \overset{\ddagger}{*} G_{k'} = \underset{\forall j \in S_{kk'}}{\cup} [I^j, X_k^j, X_{k'}^j] L^j$$

d'où

(5) $G_k \overset{\ddagger}{*} G_{k'} = G_{kk'}(I,X_k,X_{k'})$.

La condition pour regrouper G_k et $G_{k'}$ est donc $S_k = S_{k'}$.

La décomposition finale de G s'écrit donc :

(6) $G(I,X) \leftarrow G_1(I,Y_1) \overset{+}{*} G_2(I,Y_2) \ldots \overset{+}{*} G_q(I,Y_q)$

où $q \leq p$

$Y_i \subseteq X = \{X1,X2,\ldots,Xp\}$

avec $\forall i \; \forall j \in \{1,2,\ldots,q\} \quad i \neq j \Rightarrow Y_i \cap Y_j = \emptyset$

Cette expression indique, pour chaque relation globale, comment les relations locales doivent être combinées pour former l'ensemble global. Ceci entraîne les conséquences suivantes :

1) L'obtention de G se ramène dans tous les cas à l'obtention des sous-relations qui sont combinées en une seule.
Autrement dit, une telle expression peut servir à réaliser l'opération obtenir(G), mais n'est pas toujours adéquate pour réaliser les autres opérations : insertion, suppression, mise à jour au niveau global.

2) Dans la mesure où G est définie par des composantes locales ayant même clé et mêmes domaines, mais avec des critères de partitionnement (restriction, voisinage), cette expression est incomplète (cf. § 4.3.4) et doit comporter ces critères.

4.3.2.2. Règles Globales

Nous avons vu en § 2.5.4 qu'à chaque relation étaient associées quatre règles d'évolution, une par opération de base : obtention, insertion, suppression, mise à jour. Une première façon de résoudre le problème de la correspondance global/local au niveau des objets serait d'écrire explicitement pour chaque relation globale les quatre règles indiquant ainsi comment les opérations globales doivent se répercuter au niveau local.

Cette solution n'est évidemment pas acceptable dans le cas général, c'est pourquoi nous allons parler de règles standard qui sont déduites automatiquement par la machine globale en fonction d'informations sur le type de répartition de la relation globale.

Nous dirons qu'une règle est *standard* pour une relation G si la définition du type de répartition et, éventuellement, du critère de localisation, est suffisante pour que la machine globale puisse sans ambiguïté, déterminer lors d'une opération d'accès ou de modification, les répercussions à mettre en oeuvre au niveau local.

Par exemple, le partitionnement sans critère de localisation permet de déterminer la règle d'obtention : obtenir G c'est obtenir les relations locales sous-jacentes et les composer de manière à construire le tableau de valeurs de G. Nous avons là un cas de règle standard.

En revanche, déterminer la règle d'insertion n'est pas possible car en l'absence d'un critère de localisation, on ne peut connaître la machine où stocker le nouvel élément. Il s'agit alors d'un cas non standard.

4.3.2.2.1. Règles non standard

Les règles non standard doivent être explicitées dans la description de la correspondance global/local.

De ce fait, il est possible de "micro-programmer" la vue globale, en associant à une relation des programmes globaux qui seront exécutés dès que l'opération correspondante (obtenir, insérer, ...) sera demandée par l'usager de la machine globale.

Cette approche permet d'abord de résoudre les cas de répartition arbitraire, car il faut indiquer comment les relations locales doivent être utilisées dans l'opération.

Un exemple est donné au § 4.3.5.
La règle globale est alors explicitée par un programme global qui contient des opérations s'adressant à plusieurs machines locales.

Si nous généralisons cette approche, nous avons là un champ très étendu de possibilités, car il est alors envisageable d'enregistrer à la création de la vue globale des règles d'intégrité à mettre en oeuvre automatiquement (on retrouve ici une notion analogue à celle des actions spontanées

évoquées dans [B1] et dans [B9]). Un travail sur ce sujet est en cours au sein du projet POLYPHEME.

Considérons par exemple deux relations globales :

EMPLOYE (NE, NOM, SALAIRE, NUS)
USINE (NUS, LIEU, NBEMP).

Chaque usine est décrite par un numéro, un lieu et le nombre de ses employés.

Insérer un nouvel employé doit provoquer automatiquement, pour maintenir la cohérence, la mise à jour du nombre d'employés de l'usine correspondante

La règle d'évolution concernant l'insertion dans EMPLOYE doit indiquer explicitement ces opérations :

ins (EMPLOYE,t) ← (insérer(EMPLOYE,t) ; modifier(USINE :
sélecter(USINE,NUS = [NUS]t), NBEMP := NBEMP+1)).

Nous avons là en fait une transaction globale à laquelle il faut appliquer le processus de décomposition décrit au paragraphe 4.3.6.

4.3.2.2.2. Règles standard

Le tableau 4.3 donne les cas où la règle globale peut être considérée comme standard ou non.

Pour tout ce qui est partitionnement, avec ou sans duplication, mais avec un critère de localisation, les règles sont standard (cf. § 4.2.7).

En ce qui concerne la répartition triviale et la duplication totale, nous sommes dans la même situation.

Au niveau du partitionnement arbitraire (avec ou sans duplication), il faut considérer chaque opération :

- *Obtention* : Elle consiste à rassembler tous les morceaux locaux et est donc standard. Remarquons cependant que dans le cas où il y a duplication, il est peut être inutile de rassembler *tous* les morceaux de *toutes* les machines où est réparti l'ensemble global s'il existe une (ou plusieurs) machine(s) contenant cet ensemble (exemple de PRODUIT au § 4.3.2.3).

- *Insertion* : Il est impossible de localiser directement le nouvel élémer

- *Suppression* : La règle est standard car même si on ne peut localiser l'élément à enlever, on peut demander à toutes les machines de le supprimer.

- *Modifier* : Il est impossible de localiser directement l'élément à modifier (il faut supprimer puis insérer).

N.B. Une règle standard peut naturellement être remplacée par une règle explicite.

4.3.2.3. Exemple de Vue Globale

Considérons les trois bases B1, B2 et B3 définies au chapitre 3.

B1 {
- USINE (U#, LIEU, BUDGET)
- EMPLOYE (INSEE, NOM, AGE, SALAIRE, ADRESSE, U#)
- PRODUIT (P#, NOMP)
- FABRICATION (U#, P#, QTE)

B2 {
- FABRIQUE (NF, ADRF, BUDGEF)
- EMPLOYE (MATRICULE, NOM, AGE, SALAIRE, NF)
- PRODUIT (NP, NOMP)
- FABRICATION (NF, NP, QTE)

B3 PRODUIT (NP, NOMP, DESCR, PRIVEN, PRIREV).

On suppose que U# et NF se partagent le même ensemble de valeurs et qu'il en est de même respectivement pour LIEU et ADRF, BUDGET et BUDGEF, INSEE et MATRICULE, NP et P#.

A chaque base locale Bi, est associée une machine Mi (machine locale) dont la construction a été détaillée au niveau local.

Au niveau global, on désire considérer :

- un ensemble d'USINEs composé des usines de B1 et des FABRIQUEs de B2, en considérant que ces deux ensembles sont disjoints.

  USINE (U#, LIEU, BUDGET)

- un ensemble d'EMPLOYEs composé de ceux de B1 et de B2

  EMPLOYE (INSEE, NOM, AGE, SALAIRE, U#).

OPERATION \ CAS	PARTITION AVEC CRITERE	PARTITION ARBITRAIRE	PART. & DUPLIC. AVEC CRITERE	PART. & DUPLIC. ARBITRAIRE	REPARTITION TRIVIALE	DUPLICATION TOTALE
OBTENIR	Standard	Standard	Standard	Standard	Standard	Standard
INSERER	Standard	Non standard	Standard	Non standard	Standard	Standard
SUPPRIMER	Standard	Standard	Standard	Standard	Standard	Standard
MODIFIER	Standard	Non standard	Standard	Non standard	Standard	Standard

Table 4.3 - Les Règles Globales

- un ensemble de PRODUITs composé de ceux de B3 en faisant l'hypothèse que ceux de B1 et B2 ne sont que des sous-ensembles de B3 :

PRODUIT (P#, NOMP, DESCR, PRIVEN, PRIREV)

- un ensemble de produits fabriqués FABRICATION composé de ceux de B1 et de B2

FABRICATION (P#, U#, QTE).

Les critères de répartition sont les suivants :

1) USINE est considérée comme partitionnée par restriction sur les valeurs de l'attribut LIEU :

M1 : USINE : (LIEU $\in$ ('Caen', 'Paris', 'Lille'))

M2 : FABRIQUE : (ADRF $\in$ ('Grenoble', 'Marseille', 'Nice')).

2) EMPLOYE et FABRICATION sont partitionnés par voisinage avec localisation transitive, tous deux sur leur attribut qui les relie à USINE.

Pour FABRICATION :

M1 : FABRICATION [U# $\cap$ U#]USINE

M2 : FABRICATION [NF $\cap$ NF]FABRIQUE.

Chaque morceau doit être composé avec son morceau voisin.

3) PRODUIT est un cas particulier dont le découpage est indiqué par la Figure 4.5 (partitionnement et duplication arbitraire).

PRODUIT

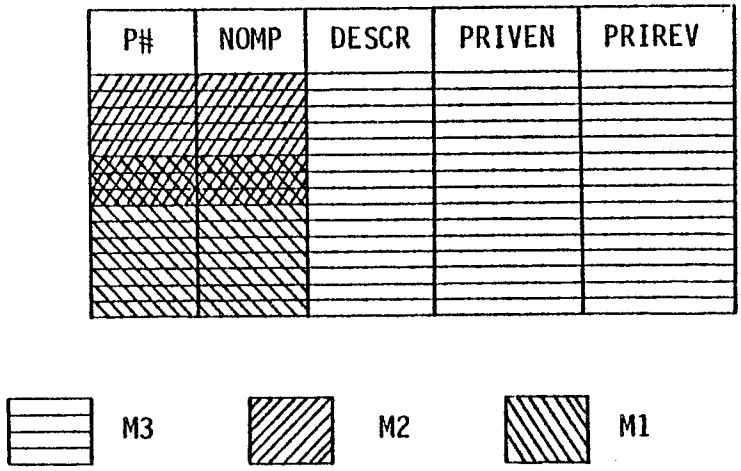


Figure 4.5 - Découpage de la relation PRODUIT

Pour exprimer PRODUIT en fonction des relations locales, il faut revenir à la décomposition binaire :

PRODUIT = P1(P#,NOMP) $\overset{+}{*}$ P2(P#,DESCR) $\overset{+}{*}$ P3(P#,PRIVEN) $\overset{+}{*}$ P4(P#,PRIREV)

avec :

P1(P#,NOMP) ← M1.PRODUIT ∪ M2.PRODUIT ∪ [P#,NOMP] M3.PRODUIT

P2(P#,DESCR) ← [P#,DESCR] M3.PRODUIT

P3(P#,PRIVEN) ← [P#,PRIVEN] M3.PRODUIT

P4(P#,PRIREV) ← [P#,PRIREV] M3.PRODUIT

Avec les critères donnés au § 4.3.2.1, on peut regrouper P2, P3 et P4 ; on obtient :

PRODUIT (P#, NOMP, DESCR, PRIREV, PRIVEN) ←

(M1.PRODUIT ∪ M2.PRODUIT ∪ [P#,NOMP] M3.PRODUIT) $\overset{+}{*}$

[P#, DESCR, PRIREV, PRIVEN] M3.PRODUIT.

Quant aux autres expressions, on a :

USINE (U#, LIEU, BUDGET) ← M1.USINE :(LIEU ∈ ('Caen','Paris','Lille')) ∪

M2.FABRIQUE : (ADRF ∈ ('Grenoble','Marseille','Nic

EMPLOYE (INSEE, NOM, AGE, SALAIRE, U#) ←

[INSEE, NOM, AGE, SALAIRE, U#] (M1.EMPLOYE [U# ∩ U#] M1.USINE) ∪

M2.EMPLOYE [NF ∩ NF] M2.FABRIQUE

FABRICATION (U#, P#, QTE) ←

M1.FABRICATION [U# ∩ U#] M1.USINE ∪

M2.FABRICATION [NF ∩ NF] M2.FABRIQUE.

N.B. En toute rigueur dans EMPLOYE et FABRICATION, il faudrait indiquer le critère de partitionnement de la relation voisine en l'occurrence USINE.

Au niveau des règles globales, nous aurons :

- pour USINE, FABRICATION et EMPLOYE des règles standard
- pour PRODUIT des règles explicites (en particulier, l'obtention ne se fera que sur M3).

On trouvera au § 4.3.5 l'expression de la Vue Globale correspondante.

4.3.3. Relations Calculées

Nous considérons ici l'ensemble RG_C des relations globales dont les éléments sont calculés, à partir de valeurs locales (degré de répartition = 0).

Chaque relation $R \in RG_C$ est une relation binaire qui correspond directement à une fonction MOGADOR de type calculée (§ 2.2.3).

Pour une relation $R(A,B) \in RG_C$, le problème est d'exprimer la manière dont on obtient les couples $(a,b) \in R$.

Cette manière consiste à donner, sous forme de programme global, "l'équation" de la fonction.

Supposons par exemple, dans la vue globale définie plus haut, que l'on veuille rajouter pour chaque produit sa capacité totale de production (attribut CAPROD).

On a la relation binaire PRODC (P#, CAPROD) pour laquelle les opérations d'insertion, de suppression et de mise à jour sont interdites alors que l'obtention d'un n-uplet est donnée par le programme suivant :

prog caprod entrée P#
sortie CAPROD.

Le corps du programme est l'expression relationnelle suivante (&P# désigne symboliquement la valeur du paramètre d'entrée) :

somme (projeter (sélecter (FABRICATION, P# = &P#), QTE)).

Notons ici que ce programme s'écrit en langage de manipulation de la vue globale, alors qu'une règle globale sera exprimée par une expression relationnelle localisée qui est en fait le résultat du processus de décomposition appliqué à une transaction globale (cf. § 4.3.6).

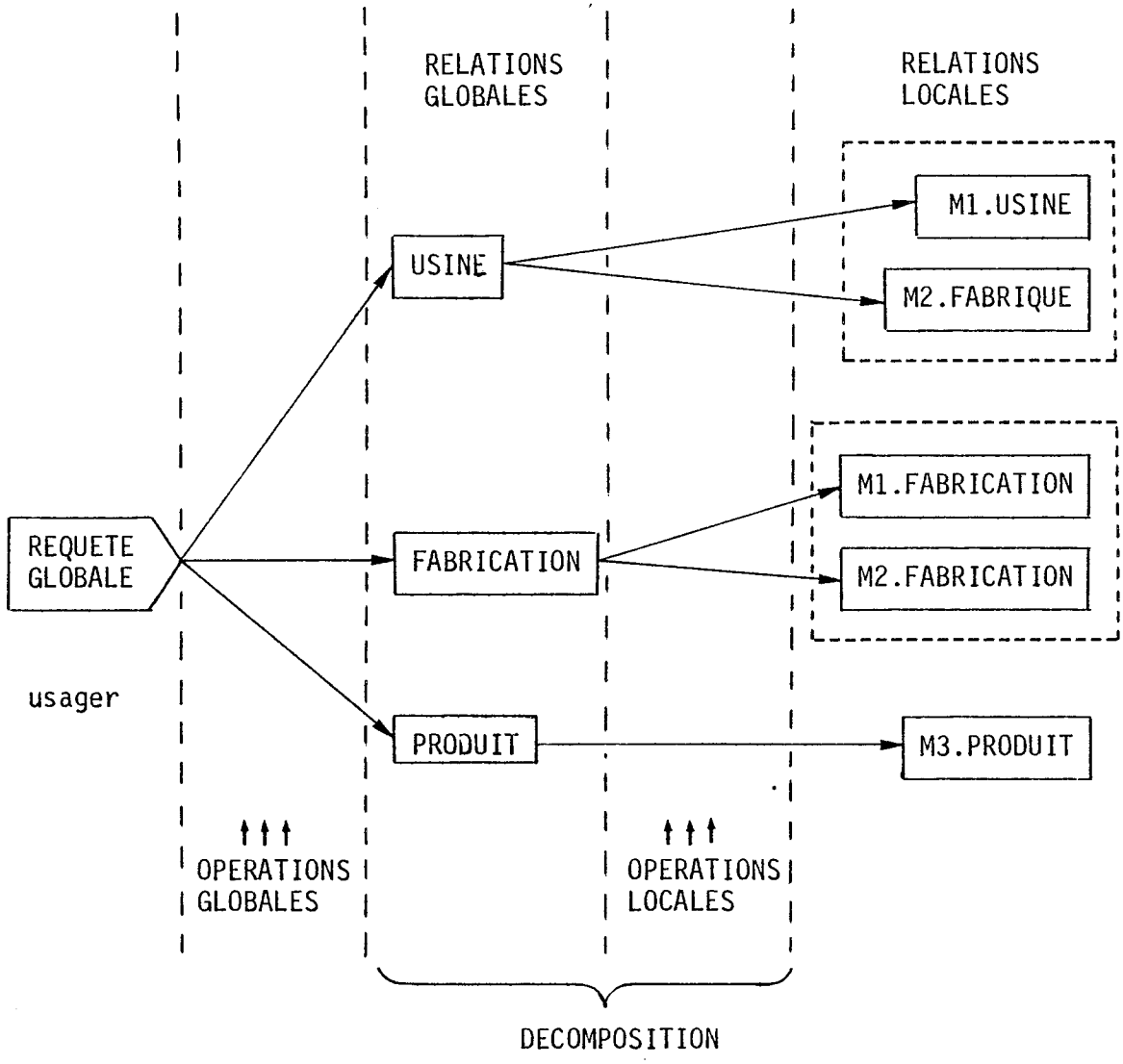


Figure 4.6 - Problème de décomposition : une requête globale portant sur 3 relations globales : USINE, FABRICATION, PRODUIT

4.3.4. Opérations sur les relations globales réparties : problème de décomposition

Toute relation globale G appartenant à l'ensemble RG_R (cf. § 4.2.6) s'exprime au moyen de relations locales. Le problème que l'on se pose alors est le suivant :

Sur les relations globales, il doit être possible d'appliquer tous les opérateurs de l'algèbre relationnelle (tables 2.13, 2.14). Comment ces opérations peuvent-elles être automatiquement répercutées sur les relations locales ?

Ce problème rejoint celui de la définition et de la manipulation des vues relationnelles comme dans le système R [B9], où l'usager peut, à partir d'un ensemble de relations de base (ici nos relations locales), définir d'autres relations en appliquant les opérateurs algébriques (relations globales).

Nous appellerons *décomposition* le processus qui consiste à transformer une série d'opérations manipulant les relations globales, en une série d'opérations manipulant les relations locales sous-jacentes (Figure 4.6).

Avant de voir comment ce processus peut être automatisé pour l'incorporer à la machine globale, nous allons étudier les problèmes qu'il pose en les séparant en deux classes :

- les problèmes liés à l'obtention de tout ou partie d'une relation globale
- les problèmes liés à la modification (insertion, suppression, mise à jour) d'une relation globale.

NOTATIONS :

Soit G(I,X) une relation globale quelconque avec

$$G(I,X) = G_1(I,X_1) \overset{+}{*} G_2(I,X_2) \overset{+}{*} \ldots \overset{+}{*} (G_n(I,X_n)$$

avec chaque $G_k(I,X_k)$ ayant la forme suivante (pour simplifier on n'utilise que des noms globaux) :

$$G_k(I,X_k) = \bigcup_{j \in S_k \subseteq LOC} [I,X_k]T^j$$

T^j pouvant prendre les trois formes suivantes :

1) - L^j une relation locale dans le cas général.
2) - $(L^j : \varphi_j)$ dans le cas d'un partitionnement par restriction.
3) - $L^j[X_p \cap K]$ $([K](V^j : \varphi_j))$ dans le cas d'un partitionnement par voisinage : V^j dénote le morceau voisin de L^j, X_p et K les attributs réciproques de L^j et V^j qui définissent la liaison.

N.B. Ces deux dernières expressions sont valables également dans le cas où il y a partitionnement avec duplication (cf. § 4.2.7.4).

Cas particuliers :

- Dans le cas d'une répartition triviale, la relation globale G correspond à une et une seule relation locale L.

- Dans le cas d'une duplication totale, la relation globale G se retrouve dans son entier sur N machines différentes (N > 1).

4.3.4.1. Duplication (en un ou plusieurs exemplaires)

Si l'on considère la répartition triviale comme cas particulier de duplication (N=1), ce cas est simple à traiter automatiquement :

- obtenir la relation globale, c'est obtenir l'une des relations locales
- modifier la relation globale, c'est modifier toutes les relations globales.

Notons cependant que pour réaliser effectivement toutes ces modifications, nous nous heurtons au problème de la cohérence de copies multiples évoquée au § 1.3.5.5.

4.3.4.2. Obtenir tout ou partie d'une relation globale

Théoriquement cette opération revient à obtenir chacun des morceaux locaux qui composent la relation globale.

Cependant, dans le cas où l'on ne s'intéresse qu'à certaines portions de la relation globale, le critère de partitionnement ou de voisinage ne permet-il pas d'éliminer a priori des opérations locales inutiles ?

Considérons deux exemples :

Ex.1 : Soit la requête globale (Q1) :

"Numéro et budget des usines localisées à Nice".

Q1 ← [U#, BUDGET] USINE : LIEU ∈ ('Nice').

La figure 4.7.a montre sur l'ensemble global, formé par les deux relations locales M1.USINE et M2.FABRIQUE, la sous-relation Q1, représentative du résultat de cette requête.

Le processus de décomposition doit détecter ces situations, quand c'est possible, de manière à ne produire que les opérations locales nécessaires à l'obtention du résultat. Ici, le critère de partitionnement sur l'attribut LIEU permet de réduire la requête globale (cf. § 4.3.4.2.1).

Ex.2 : Considérons la requête suivante (Q2) :

"Obtenir le numéro et le prix de vente des produits fabriqués par des usines localisées à Nice".

Pour simplifier, considérons que cette requête s'écrit en deux morceaux :

Q21 ← [P#](FABRICATION[U# ∩ U#](USINE : LIEU ∈ ('Nice')))

(ensemble des produits fabriqués à Nice).

Q22 ← [P#,PRIVEN] PRODUIT [P# ∩ P#] Q21

(prix de vente de ces produits).

La figure 4.7.b montre, sur un exemple, la sous-relation représentative de la sous-requête Q21 du fait de la dépendance qui lie FABRICATION et USINE (partitionnement par voisinage).

Là encore, le processus de décomposition doit détecter les compositions de relations globales liées par voisinage, de manière à réduire les opérations locales inutiles.

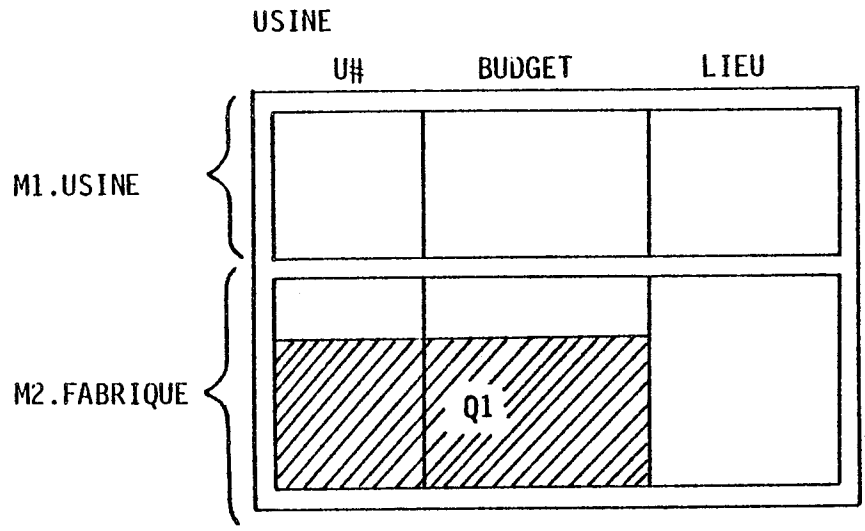
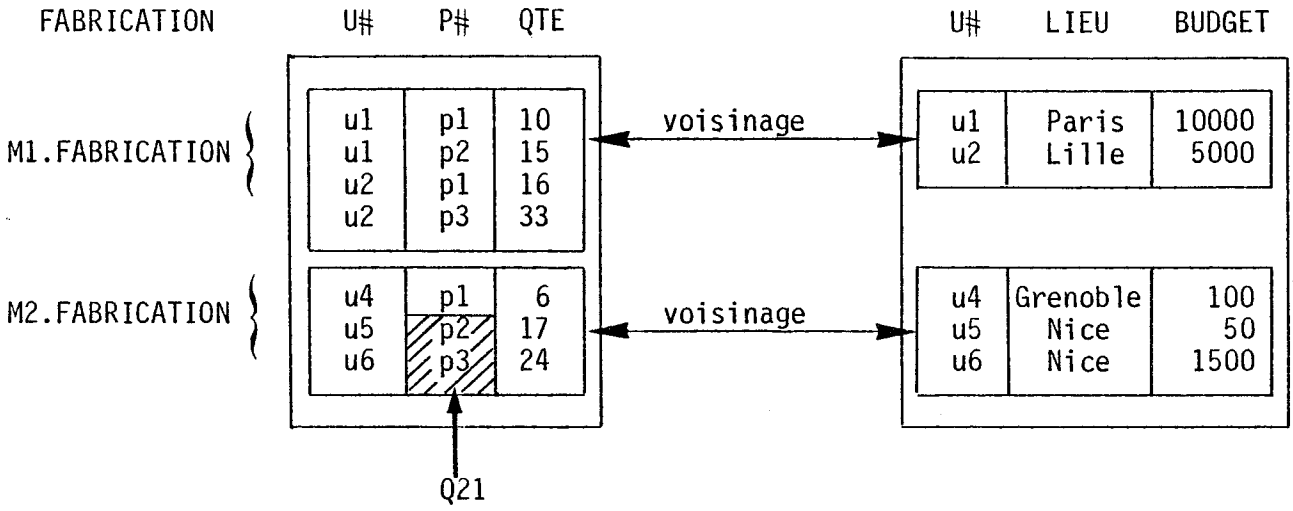


Figure 4.7.a - Projection et Restriction d'une relation globale partitionnée



où Q21 ← [P#] (FABRICATION [U# ∩ U#] (USINE : LIEU ∈ ('Nice')))

Figure 4.7.b - Composition de deux relations globales liées par voisinage

4.3.4.2.1. Projection et Restriction de relation globale

Pour voir comment se propage la projection et la restriction d'une relation globale G sur les relations locales correspondantes, considérons G définie par son expression de décomposition dans laquelle les regroupements ont été effectués (cf. § 4.3.2.1)

(1) $G(I,X) \leftarrow G_1(I,Y_1) \overset{+}{*} \ldots \overset{+}{*} G_p(I,Y_p)$.

Pour simplifier l'écriture, nous allons combiner les opérations de projection et de restriction en une seule : la ψ-projection [B3], où ψ désigne une expression conditionnelle (cf. § 2.5.3.2).

Une sous-relation globale s'exprime ainsi :

(2) $[\mu;\psi]G = \{[\mu]t : t \in G \wedge \psi_t\}$.

où ψ_t est l'expression conditionnelle appliquée au n-uplet t.

En développant (2) en fonction de (1), on obtient :

(3) $[\mu;\psi]G = [\mu;\psi]\ (G_1(I,Y_1) \overset{+}{*} \ldots \overset{+}{*} G_p(I,Y_p))$.

On peut considérer que ψ est de la forme $\psi_1 \wedge \psi_2 \ldots \wedge \psi_p$ où chaque ψ_i est une expression conditionnelle qui porte sur les attributs de G_i.

Ainsi, en appliquant une série de transformations algébriques décrites dans [R6] mais surtout dans [R17], (3) peut se réduire à :

(4) $[\mu](G_1(I,Y_1) : \psi_1 * G_2(I,Y_2) : \psi_2 \ldots * G_p(I,Y_p) : \psi_p)$

où le filtre restrictif est appliqué à chaque niveau de G.

Or, chaque $G_k(I,Y_k) = \underset{\forall j \in S_k}{\cup} [I,Y_k](L^j : \varphi_k^j)$ dans le cas d'un partitionnement par restriction.

$G_k(I,Y_k) : \psi_k$ se transforme en :

(5) $G'_k(I,Y_k) = \underset{j}{\cup} [I,Y_k] (L^j : \varphi_k^j \wedge \psi_k^j)$.

L'expression conditionnelle composée $\varphi_k^j \wedge \psi_k^j$ est alors évaluée pour être éventuellement réduite, éliminant ainsi automatiquement des morceaux locaux.

De la même manière, la projection sur μ peut se propager sur chaque G'_k. Il importe alors d'examiner la composition de μ par rapport à la clé d'une part, et à chaque ensemble d'attributs d'autre part.

3 cas sont à considérer ($I \cap \mu' = \emptyset \wedge \mu' \neq \emptyset$) :

C1 : $\mu = I \cup \mu'$ projection sur attributs comprenant la clé.
C2 : $\mu = \mu'$ projection sur attributs non clés.
C3 : $\mu = I$ projection sur attributs clés.

Les transformations permettent alors de réduire l'expression globale de la manière suivante :

- Cas C1 : G'_k se ramène à $\underset{j}{\cup} [I,\mu' \cap Y_k] (L^j : \varphi_k^j \wedge \psi_k^j)$.
- Cas C2 : (4) donne $[\mu'] ([\mu'] G'_1 * \ldots * [\mu']G'_p)$.
- Cas C3 : (4) donne : $\underset{k}{\cup} G'_k(I)$ (union des identificateurs).

Exemple :

Reprenons la requête globale Q1 vue plus haut :

"Donner le numéro et le budget des usines localisées à Nice" :

Q1 ← [U# ; BUDGET ; LIEU ∈ ('Nice')] USINE.

Or, USINE est définie par l'expression suivante (pour simplifier, nous n'avons utilisé que des noms globaux) :

(1) USINE ← [; LIEU ∈ ('Caen','Paris','Lille')] M1.USINE ∪
[; LIEU ∈ ('Grenoble','Marseille','Nice')] M2.USINE.

En reportant Q1 dans (1), on a :

[U#, BUDGET ; LIEU ∈ ('N')] ([; LIEU ∈ ('C','P','L')] M1.USINE
[; LIEU ∈ ('G','M','N')] M2.USINE).

La propagation de l'expression conditionnelle donne :

[U#, BUDGET] ([; LIEU ∈ ('C','P','L') ∧ LIEU ∈ ('N')] M1.USINE
[; LIEU ∈ ('G','M','N') ∧ LIEU ∈ ('N')] M2.USINE.

L'évaluation de LIEU ∈ ('C','P','L') ∧ LIEU ∈ ('N') donne faux tandis que LIEU ∈ ('G','M','N') ∧ LIEU ∈ ('N') se réduit à LIEU ∈ ('N') et il reste :

[U#, BUDGET].([; LIEU ∈ ('N')] M2.FABRIQUE)

ou bien en propageant le critère de projection :

[U#, BUDGET ; LIEU ∈ ('N')] M2.FABRIQUE.

Ce qui détermine la machine où il faut effectuer l'accès.

4.3.4.2.2. Composition de deux relations globales liées par voisinage

Le problème est ici de mettre en évidence dans une série d'opérations globales la composition de deux relations liées par voisinage. Cette composition caractérise une relation dont le découpage correspond à celui du partitionnement par restriction, ce qui permet de réduire l'expression globale (Figure 4.7.b).

Considérons deux relations globales R(I,X) et S(K,Z) liées par voisinage sur les attributs respectifs X_p et K.

Supposons que R soit composée de deux relations locales

$$M1.R = RL^1[X_p \cap K]\,([K]\,M1.S)$$
$$M2.R = RL^2[X_p \cap K]\,([K]\,M2.S)$$

avec $M1.S = (SL^1 : \varphi_1)$ et $M2.S = (SL^2 : \varphi_2)$.

Voyons comment peut se réduire l'expression globale W :

$$W = R[X_p \cap K]S.$$

L'ensemble de localisation de R et celui de S coïncident, W est donc composée de deux morceaux :

$$M1.W = M1.R[X_p \cap K]M1.S$$
$$M2.W = M2.R[X_p \cap K]M2.S.$$

En remplaçant Mi.R et Mi.S par leur expression termes locaux, on obtient pour chaque Mi.W :

$$Mi.W = (RL^i[X_p \cap K]([K]Mi.S))[X_p \cap K]Mi.S.$$

Ce qui en appliquant des transformations algébriques (cf. § 4.3.6.1) se réduit à :

$$Mi.W = RL^i[X_p \cap K](SL^i : \varphi_i)$$

(un exemple est donné au § 4.3.6).

4.3.4.3. Modification d'une relation globale

Du fait des dépendances entre relations locales et relations globales, il est important de se préserver dans la base répartie de toute incohérence, introduite par les opérations de modification.:

Ce problème revêt un double aspect [R6, B48] :

- quand on modifie une relation globale, quelles sont les répercussions sur les relations locales composantes ?
- quand on modifie directement une relation locale, quelles sont les répercussions au niveau global ? (nous abordons ce cas au § 4.3.4.5).

Les modifications sont de trois types :

- l'*insertion* qui se ramène à l'union d'une relation G avec une relation M (éventuellement réduite à un seul n-uplet)
- la *suppression* qui se ramène à la différence entre la relation G et la relation M (éventuellement réduite à un seul n-uplet)

- la *mise à jour* qui se ramène à une suppression suivie d'une insertion.

4.3.4.3.1. Insertion : G(I,X) ∪ M(I,X)

Conceptuellement, insérer un n-uplet correspond à la création d'un identificateur suivi d'une série de liaisons de type attribut (cf. § 4.2.2).

Considérons un n-uplet $t \in M$; créer un nouvel identificateur [I]t dans G s'écrit :

$$[I]G \cup [I]t$$

Or, [I]G est définie par : $\bigcup_{\forall i \in Loc} [I^i]L^i$ (cf. § 4.3.2.1).

Avant toute insertion globale d'un n-uplet $t \in M$, il faut donc s'assurer que [I]t n'existe dans aucune des machines où est répartie G.

Il faut aussi déterminer dans quelle relation locale, on va créer le nouvel identificateur.

L'insertion d'un nouvel n-uplet dans une relation définie comme l'union de plusieurs autres n'est pas valide car la répercussion de ce changement sur les relations sous-jacentes peut se faire de différentes façons, qui ne conduisent pas au même résultat. Il faut donc avoir des moyens d'enrichir la définition des relations globales. Ces moyens sont fournis soit par des règles explicites d'insertion (cf. § 4.3.2.2), soit par des règles standards qui se déduisent des types de répartition (cf. table 4.3).

Insertion dans une relation partitionnée :

Dans la mesure où les informations concernant le critère de partitionnement touchent toutes les relations locales, il est possible de garantir la validité des répercussions locales produites par une insertion :
Insérer un n-uplet dans une relation globale G, c'est l'insérer partout où la propriété qui le caractérise coïncide avec celle qui définit la relation locale.

L'insertion globale G ∪ M se décompose en autant d'insertions locales
$(L^i \cup M^i)$ $\forall i \in Loc$
où $M^i = [I,X^i]M$.

- Dans le cas du partitionnement par restriction, on a :

$$(L^i : \varphi_i) \cup M^i \supseteq L^i \cup (M^i : \varphi_i)$$

Les n-uplets de M se projettent sur les attributs locaux et sont soumis au filtre restrictif.

Dans le cas strict de partitionnement, les φ_i sont disjoints. De ce fait, plusieurs expressions $(M^i : \varphi_i)$ vont donner l'ensemble vide et l'insertion ne se fera que sur une seule machine.

Exemple :
"Insérer dans la relation USINE le n-uplet <u10, Grenoble, 200>".
Cette opération s'écrit : USINE ∪ <u10,Grenoble,200>
qui se décompose en :

i) M1.USINE ∪ ((<u10,Grenoble,200>) : LIEU ∈ ('Caen','Paris','Lille'))
ii) M2.USINE ∪ ((<u10,Grenoble,200>) : LIEU ∈ ('Grenoble','Marseille','Nice').

L'expression i) donne l'ensemble vide, car le n-uplet ne répond pas au critère, aussi l'insertion n'aura-t-elle lieu que dans M2.

- Dans le cas du partitionnement par voisinage, on a :

$$(L^i * (V^i : \varphi_i)) \cup M^i \supseteq L^i \cup M^i * (V^i : \varphi_i)$$

Soit G définie comme l'union de deux relations locales L1 et L2 :

$$G(I,X) = L1(I,X) \cup L2(I,X).$$

Et supposons l'existence d'une relation globale G', définie également comme l'union de deux relations locales L1' et L2'

$$G'(I',X') = L1'(I',X') \cup L2'(I',X').$$

Supposons que :

1) G' est partitionnée par restriction sur M1 et M2.
2) G est partitionnée par voisinage avec G'.

La liaison entre G et G' implique que $I' \subseteq I \cup X$, c'est-à-dire que les attributs de G contiennent la clé de G'.

Par exemple :

pour G FABRICATION (U#, P#, QTE)
pour G' USINE (U#, LIEU, BUDGET).

Au niveau local, chaque Li est composé avec le Li' :

L1(I,X) [I' ∩ I']([I' ; φ_1]L1') ou plus simplement :

L1 * ([I' ; φ_1]L1')

et L2 * ([I' ; φ_2]L2').

L'insertion de M se ramène à la suite d'opérations suivantes :

(1) (L1 * ([I' ; φ_1]L1')) ∪ M

(2) (L2 * ([I' ; φ_2]L2')) ∪ M

qui se remplacent respectivement par :

(3) L1 ∪ (M * [I' ; φ_1] L1')

(4) L2 ∪ (M * [I' ; φ_2] L2').

Si G est partitionnée, on a [I' ; φ_1]L1' ∩ [I' ; φ_2]L2' = ∅, ce qui fait que l'insertion n'aura lieu que dans une seule machine.

Cependant, selon les rapports existant entre la projection et le critère de restriction, la machine impliquée dans l'opération sera ou non déterminée explicitement à la décomposition :

Si φ porte sur l'attribut projeté, on pourra effectivement déterminer la machine, dans le cas contraire non.

Exemple :

Insérer dans FABRICATION le n-uplet <u10,p12,50>.

On aura :

i) M1.FABRICATION ∪ (<u10,p12,50> * [U#;LIEU∈('Paris','Lille','Caen')]
M1.USINE)

ii) M2.FABRICATION ∪ (<u10,p12,50> * [U#;LIEU∈('Grenoble','Marseille','Nice
M2.FABRIQUE).

Si u10 est dans M1; l'expression entre parenthèses dans ii) donnera l'ensemble vide, mais on ne pourra s'en apercevoir qu'à l'exécution de ces deux transactions respectivement par M1 et M2.

4.3.4.3.2. Suppression G(I,X) - M(I,X)

La suppression n'a de sens que si l'on supprime l'identificateur (la clé). Contrairement au cas de l'insertion, la suppression d'un n-uplet (ou d'un ensemble de n-uplets) d'une relation définie comme l'union de plusieurs autres, est valide.

$$[I]G - [I]t = (\bigcup_i [I^i]L^i) - [I]t$$

$$= \bigcup_i ([I^i]L^i - [I^i]t)$$

De sorte que la suppression globale se ramène à autant de suppressions locales : $L^i - M^i$.

La connaissance du critère de partitionnement permet d'affiner l'ensemble des suppressions locales.

- Dans le cas d'un *partitionnement par restriction*, on aura :

$$(L^i : \varphi_i) - M^i = L^i - M^i : \varphi_i.$$

Dans le cas où les φ_i sont disjoints, la suppression ne se fera que sur une seule machine.

Exemple : Supprimer de USINE le n-uplet <u20,PARIS,150>.

Cette opération se décompose en :

i) M1.USINE - <u20,Paris,150> : LIEU ∈ ('Paris','Lille','Caen')

ii) M2.USINE - <u20,Paris,150> : LIEU ∈ ('Grenoble','Marseille','Nice').

L'opération ne se fera ici que sur M1.

- Dans le cas d'un *partitionnement par voisinage*, on a :

$$(L^i * V^i : \varphi_i) - M^i = L^i - M^i * V^i : \varphi_i.$$

4.3.4.3.3. La mise à jour

Conceptuellement, modifier les valeurs d'un ou plusieurs n-uplets revient à le (les) supprimer puis à l'insérer (les insérer) :

$$(G(I,X) - M1(I,X)) \cup M2(I,X)$$

où M1 désigne l'ensemble des n-uplets à modifier et M2 les nouvelles valeurs.

Il suffit alors de se référer aux deux cas vus précédemment, avec la différence suivante :
avant l'insertion de [I]t (t ∈ M2), il est inutile de vérifier sa non-existence dans chaque L^i, car l'opération de suppression la garantit.

Si le n-uplet modifié est tel que l'on change la valeur de l'attribut sur lequel porte le partitionnement et si ce changement de valeur est tel que l'on passe d'une partie à une autre, la séquence d'opérations :

suppression-insertion garantit le déplacement du n-uplet dans la bonne base.

Exemple :

"Modifier dans USINE le n-uplet <u13,Paris,50> en <u13,Grenoble,150>.

USINE - <u13,Paris,50> puis USINE $\cup$ <u13,Grenoble,150>.

Soit :

i) M1.USINE - <u13,Paris,50> : LIEU $\in$ ('Paris','Lille','Cane')
M1.USINE $\cup$ <u13,Grenoble,150> : LIEU $\in$ ('Paris','Lille','Caen').

ii) M2.USINE - <u13,Paris,50> : LIEU $\in$ ('Grenoble','Marseille','Nice')
M2.USINE $\cup$ <u13,Grenoble,150> : LIEU $\in$ ('Grenoble','Marseille','Nice').

Il y aura transfert du n-uplet de M2 vers M1.

4.3.4.3.4. En résumé, le tableau 4.4 donne les répercussions d'une opération globale de modification sur les relations locales.

- G désigne une relation globale quelconque qui est répartie sur N machines ($N \geq 1$).(G(I,X)).
- On note $\{L^i\}$ l'ensemble des relations locales qui composent G. Selon les types de répartition et de localisation, la colonne "correspondance global/local" donne la composition exacte de cet ensemble L^i (φ désigne un critère de partitionnement et V^i désigne la relation voisine de L^i dans le cas d'un partitionnement par voisinage.

Pour chaque opération de modification (insertion, suppression), on note M la relation contenant le (ou les) n-uplet(s) à insérer ou à supprimer (M(I,X)).

L'opération globale G $\cup$ M ou G-M donne naissance, selon les cas, à un ensemble d'opérations locales sur les L^i.

La colonne (Min, Max) indique dans chaque cas le nombre minimum et maximum de machines où doit se faire l'opération pour que la cohérence globale soit respectée.

Dans le cas où M fait référence à des relations locales, il faut la remplacer par son expression en fonction de ces relations (cf. § 4.3.6.3).

Type de Répartition	Type de Localisation	Correspondance Global/Local (i=1,2,...N)	INSERTION		SUPPRESSION	
			G ∪ M	Min,Max	G - M	Min,Max
Partition	Directe	$G=\{L^i:\varphi_i\}$	$\{L^i \cup (M^i:\varphi_i)\}$	0, 1	$\{L^i - (M^i:\varphi_i)\}$	0, 1
Partition avec duplication				0, N		0, N
Partition	Transitive	$G=\{L^i*(V^i:\varphi_i)\}$	$\{L^i \cup (M^i*(V^i:\varphi_i))\}$	0, 1	$\{L^i-(M^i*(V^i:\varphi_i))\}$	0, 1
Partition et duplication				0, N		0, N
Répartition triviale (Duplication simple)	Directe	G = L	L ∪ M	1, 1	L - M	1, 1
Duplication totale	Directe	$G = \{L^i\}$	$\{L^i \cup M\}$	N, N	$\{L^i - M\}$	N, N

Table 4.4 - Modification au niveau global : décomposition en opérations locales

4.3.4.4. Modification au niveau local : cohérence local/global

La mise en place d'applications nécessitant la coopération de plusieurs bases de données rejaillit d'une certaine façon sur leur indépendance.

La définition d'une relation globale à partir de plusieurs relations locales crée, nécessairement, des dépendances entre ces relations qui auparavant étaient indépendantes. Or, ces relations ne sont connues qu'au niveau global ; au niveau local, toute modification effectuée sans contrôle global peut introduire des incohérences dans la base répartie.

Prenons, par exemple, la relation globale USINE construite sur les ensembles M1.USINE et M2.FABRIQUE suivants :

M1.USINE	M2.FABRIQUE
u1	u4
u2	u5
u3	u6

Si au niveau de M1 on crée le nouvel élément u4, on a deux ensembles cohérents de manière indépendante :

M1.USINE	M2.FABRIQUE
u1	u4
u2	u5
u3	u6
u4	

mais incohérents pris globalement, car il n'y a plus unicité de la clé.

Généralement, une vue locale correspond à un sous-schéma de la base locale ; aussi une solution, qui paraît naturelle et qui ne remet pas en cause complètement l'indépendance des bases locales est-elle d'interdire localement toutes les opérations de modifications sur le sous-schéma local qui correspond à une vue locale. Ces opérations seront mises en oeuvre et contrôlées uniquement par le niveau global (cf. Figure 4.8). Nous rejoignons ici le problème plus général de la cohérence d'une base répartie.

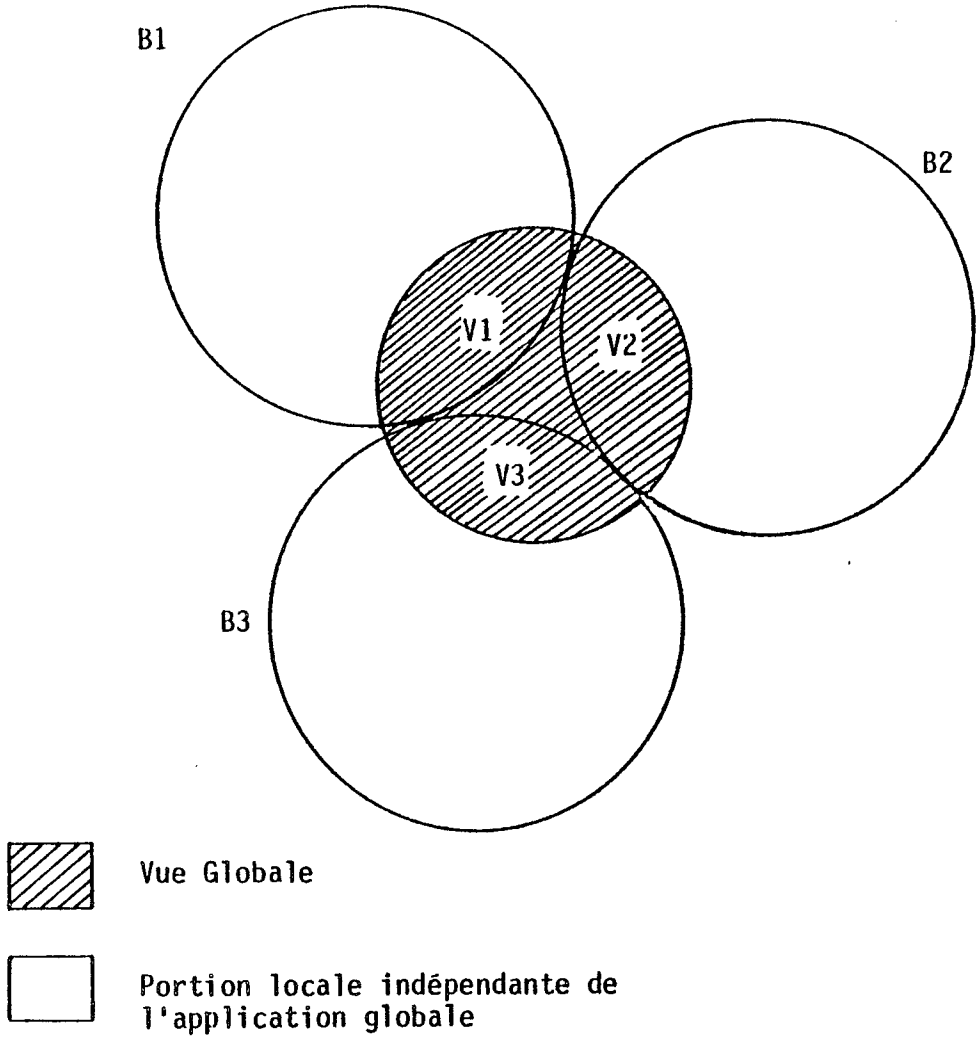


Figure 4.8 - Partage d'informations entre trois bases locales B1, B2, B3

4.3.5. Expression de la Correspondance Global/Local

Nous abordons ici l'aspect externe de la machine globale : comment l'administrateur global doit-il décrire la correspondance avec les bases locales pour que la machine puisse fonctionner automatiquement ?

La syntaxe complète du langage de description est donnée en annexe mais nous allons regarder comment est donnée à la machine globale la connaissance nécessaire, pour qu'elle puisse appliquer le processus de décomposition sur les transactions globales (cf. § 4.3.6).

Pour chaque relation globale G, il est nécessaire de fournir trois informations :

1) *Le type de répartition* de G et le critère éventuel de localisation.

G { partitionnée / partitionnée-dupliquée / dupliquée } [par { restriction / voisinage de <relation> }]

Les trois premiers mots-clés définissent le type de répartition. Le mot-clé "dupliquée" dénote une duplication simple ou multiple. La duplication simple correspond à la répartition triviale (cf. § 4.2.7.5) : ce sera le nombre de relations locales composantes qui fera la distinction.

Les deux premiers types de répartition peuvent être éventuellement complétée par l'information relative au critère de localisation.

Si une relation G est déclarée avec un critère de voisinage, il faut que G comporte dans ses attributs un ensemble qui sert de clé à une autre relation globale.

2) *L'ensemble des relations locales* qui composent G. Ceci permet d'exprimer par la même occasion la correspondance noms locaux - noms globaux.

Pour chaque portion locale L^i (machine M_i) de G, on aura une expression de la forme :

$$Mi.G\ (X1,X2,\ldots,Xq) \leftarrow \left\{ \begin{array}{l} L^i \\ L^i : \varphi \end{array} \right\}$$

où φ dénote la restriction de partitionnement.
Cette expression permet en effet d'indiquer la correspondance entre les noms :

- soit nomloc la fonction qui correspond soit à nomloccag, nomlocccg, nomlocfg (cf. § 4.3) et telle que :

$$MACHINE \times NOMGLOBAL \xrightarrow{nomloc} NOMLOCAL$$

- nomloc (Mi,G) = L^i
- nomloc (Mi,X_k) = nom du $k^{ième}$ attribut défini dans la déclaration de L^i soit X^i_k.

Quant à la correspondance entre les noms de domaines, elle est indirecte car on a la fonction monovaluée suivante :

$$\text{ATTRIBUT} \xrightarrow{\text{domatt}} \text{DOMAINE}$$

A un attribut, il ne correspond qu'un seul domaine :

$$\text{nomloc}(Mi, \text{domatt}(X_k)) = \text{domatt}(X_k^i).$$

3) La présence obligatoire de règles globales explicites pour tous les cas non standards (table 4.3).

Il y a quatre types de règles :

1. *Obtention* : obt(G) ← <expression relationnelle localisée>.

Il s'agit d'une expression ne manipulant que des relations locales.

2. *Insertion* : ins(G,t).
3. *Suppression* : sup(G,t). } ← <expr. relationnelle localisée>
4. *Modification* : mod(G,t,t')

Avec l'exemple du § 4.3.2.3, nous avons :

- Correspondance local/global :
  - USINE partitionnée par restriction
    M1.USINE (U#,LIEU,BUDGET) ← USINE : LIEU ∈ ('Caen','Paris','Lille')
    M2.USINE (U#,LIEU,BUDGET) ← FABRIQUE : LIEU ∈ ('G','M','N').
  - FABRICATION partitionnée par voisinage de USINE
    M1.FABRICATION (U#, P#, QTE) ← FABRICATION
    M2.FABRICATION (U#, P#, QTE) ← FABRICATION.

Pour PRODUIT, il faut expliciter les règles globales pour exprimer la sémantique suivante :

- la relation dans M3 correspond au catalogue des produits
- les relations dans M1 et M2 ne sont que des sous-ensembles du catalogue.

  - PRODUIT partitionnée-dupliquée
    M1.PRODUIT (P#, NOMP) ← PRODUIT
    M2.PRODUIT (P#, NOMP) ← PRODUIT
    M3.PRODUIT (P#, NOMP, DESCRP, PRIX-REVIENT, PRIX-VENTE)

← PRODUIT

obt (PRODUIT) ← M3.PRODUIT

ins (PRODUIT,t) ← (insérer(M1.PRODUIT,t), insérer(M2.PRODUIT,t), insérer(M3.PRODUIT,t))

et des règles analogues pour sup et mod.

4.3.6. Processus de décomposition

Après avoir pris en compte la vue globale et la correspondance avec les vues locales, l'autre fonction importante de la machine globale est d'effectuer automatiquement le processus de décomposition défini au § 4.3.4.

Ce processus consiste à transformer chaque requête globale d'interrogation ou de modification en requêtes locales mono-machines.

Il faut également produire un ordre partiel sur ces requêtes locales, de manière à décrire la synchronisation logique de leur activation.

Dans un environnement réparti, il est indispensable de tenir compte des deux considérations suivantes :

- les coûts de communication entre machines sont très élevés par rapport aux exécutions locales
- l'opportunité d'exécution parallèle.

La stratégie de décomposition doit alors mettre l'accent sur la minimisatio des coûts de transferts entre machines, tout en mettant en évidence les possibilités d'exécution en parallèle.

Ce problème fait actuellement l'objet de travaux dans les systèmes SDD-1 et INGRES [R32, R64, R65]. Dans le cadre du projet POLYPHEME, une première approche limitée à l'interrogation a été faite dans le projet de DEA de CALECA, FORESTIER [R16] et une étude plus complète du problème a été faite dans la thèse de J.Y. CALECA [R17].

4.3.6.1. Optimisation de requêtes relationnelles

Toute requête globale ou locale est en fait une expression (algébrique) relationnelle (cf. § 2.5.3.4) analogue à celle que reçoit un système centralisé. Or, de nombreux travaux [B22, B64, B9, B43] ont montré que ces expressions relationnelles devaient être optimisées pour réduire le temps de réponse.

Ces optimisations se classent en deux grandes familles :

- les *optimisations algébriques heuristiques* qui font usage des propriétés de l'algébre relationnelle pour rearranger les opérations de manière à diminuer les volumes d'éléments à manipuler [B22, B64, B43].

- les *optimisations non algébriques* englobées dans un modèle prévisionnel : elles nécessitent l'utilisation de procédés qui sortent du cadre de l'algèbre relationnelle et qui demandent la connaissance de l'organisation physique des données (index secondaires, tris intermédiaires) et de l'algorithme qui réalise chaque opérateur algébrique. Ainsi le mécanisme d'optimisation établit l'ensemble des solutions possibles avant d'estimer leur coût. La grande difficulté dans cette approche est d'obtenir des prédictions précises des coûts [B9, B49, B22].

Etant donné le contexte dans lequel nous nous plaçons - la machine globale est indépendante des considérations physiques d'implantation des bases locales - seules les premières optimisations sont retenues.

Dans [R17], on trouvera le détail de transformations algébriques de trois types :

1) - celles qui combinent un ensemble de restrictions en une restriction composée.

2) - celles qui réduisent des expressions relationnelles ou booléennes, contenant des expressions nulles qui proviennent de l'application de transformations de type 1).

3) - celles qui déplacent les opérateurs de projections et de restrictions. En effet, de manière à réduire les volumes de données, ces opérations doivent être exécutées le plus tôt possible.

Au cours des différentes étapes du processus de décomposition, ces transformations sont appliquées aussi bien au niveau de la requête globale, fournie par l'usager de la machine, qu'au niveau des requêtes locales, résultant de la prise en compte de la localisation (cf. § 4.3.6.3) (Figure 4.9) :

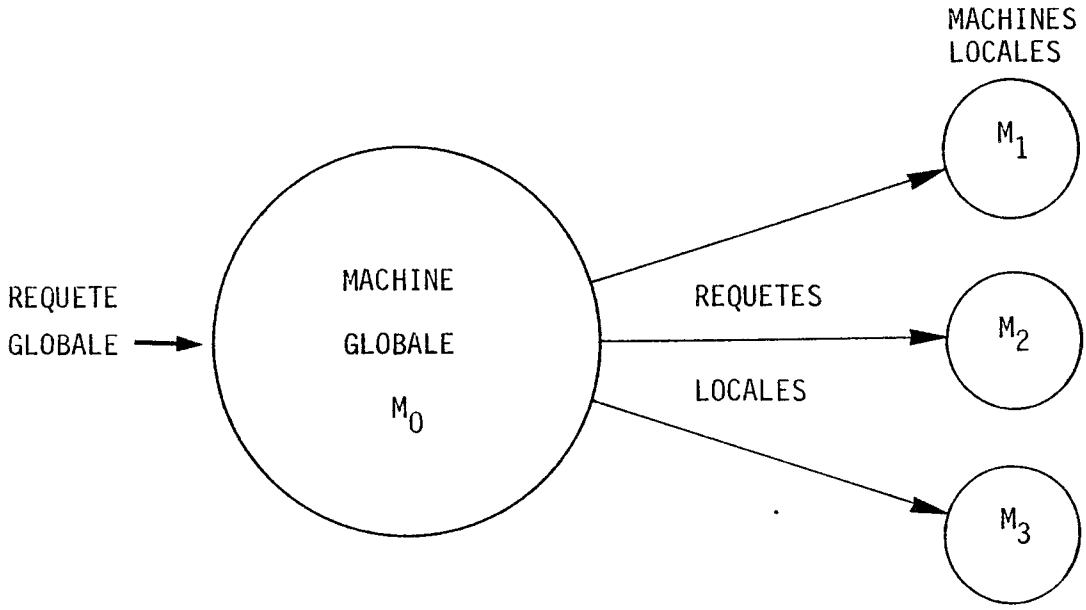


Figure 4.9

Le processus de décomposition se déroule en trois phases principales :

- . Construction d'un graphe global.
- . Prise en compte de la localisation des relations globales pour produire un graphe localisé.
- . Définition d'une stratégie de découpage du graphe localisé en sous-graphes représentatifs de requêtes mono-machines.

4.3.6.2. Graphe global

La requête source exprimée dans le langage de manipulation (LA.DO.RE, cf. Annexe) est transcodifiée un graphe global orienté G = [N,A] (sans circuits) :

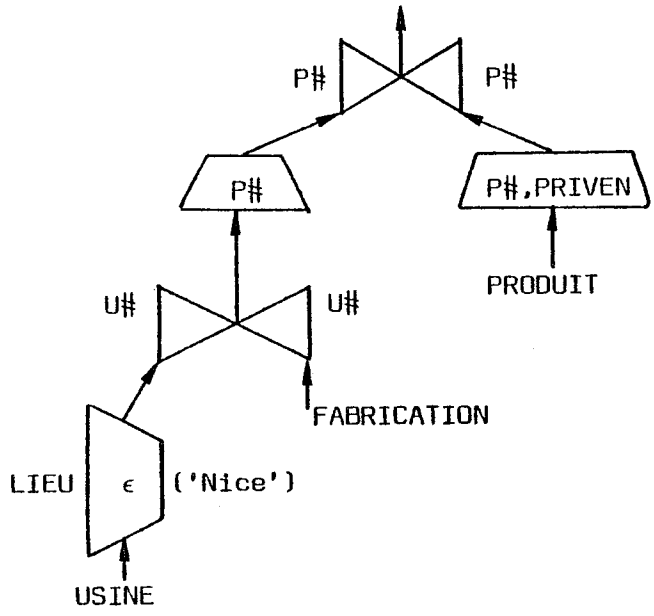
- N : ensemble fini des noeuds qui représentent les opérations mises en oeuvre par la requête (voir tables 2.13 et 2.14) pour le formalisme graphique de représentation des opérateurs.
- A : sous-ensembles de couples ordonnés (x,y) ; les arcs (x et y sont des éléments de N). Un arc matérialise le flot de données produit par l'opération x et consommé par l'opération y.

Exemple :

La requête : "obtenir le numéro et le prix de vente de tous les produits fabriqués par des usines localisées à Nice" évoquée au § 4.3.4.2 se traduit par l'expression relationnelle suivante :

(([P#,PRIVEN]PRODUIT)[P# ∩ P#]([P#](FABRICATION[U# ∩ U#](USINE : LIEU ∈ ('Nice')))))

qui se visualise mieux sur le graphe suivant (G4.1) :



Graphe G4.1

4.3.6.3. Prise en compte de la localisation

Une relation globale demeure sous sa forme implicite (en tant que définition à partir de relations locales), jusqu'à ce qu'elle soit utilisée dans une requête et rendue explicite comme nouvel élément de l'espace des éléments globaux.

Au cours du § 4.3.4, nous avons vu quels étaient les problèmes posés par la propagation des opérations d'accès ou de modifications globales sur les relations locales sous-jacentes.

Le processus de décomposition réalise cette propagation, aussi bien dans les cas standard de répartition que dans les cas non standard (cf. Table 4.3), pour produire finalement un graphe localisé et optimisé.

Tout en distinguant les requêtes d'interrogation et de modification, la prise en compte de la localisation se fait en plusieurs étapes que nous allons décrire ci-après.

Remarquons auparavant que toute citation de relation globale apparaît forcément à un noeud terminal du graphe. Cette citation correspond dans le cas général à une opération de lecture (Graphe G4.1).

* E1 : Prendre en compte les cas non-standard de répartition en substituant dans le graphe toute citation de relation globale par l'expression relationnelle localisée, apparaissant explicitement dans la règle globale (cf. § 4.3.5).

* E2 : Sur le nouveau graphe ainsi obtenu, mettre en oeuvre un ensemble de transformations destinées à :

- tenir compte des liens entre relations globales : mettre en évidence la composition de deux relations liées par voisinage (cf. § 4.3.4.2.2)
- réduire le volume des données transitant entre opérations (cf. § 4.3.6.1).

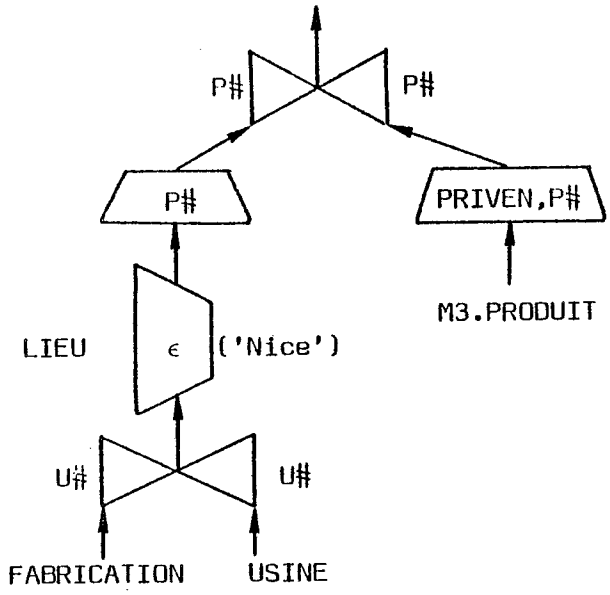
* E3 : Prendre en compte les cas standard de répartition : il faut combiner les informations relatives à la répartition (morceaux locaux) avec le contexte d'utilisation des relations locales, de manière à éliminer des sous-expressions locales inutiles (cf. § 4.3.4) ; on obtient ainsi un graphe optimisé.

Nous allons illustrer cette étape par deux exemples, l'un d'interrogation, l'autre de modification.

Exemple 1 :

Reprenons l'exemple du graphe G4.1.

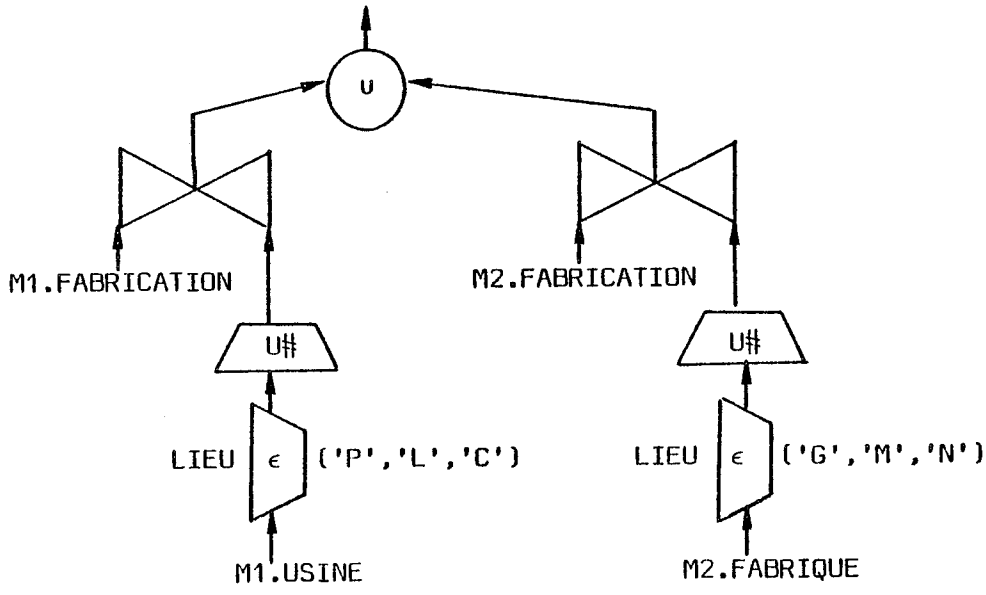
La prise en compte de la règle explicite d'obtention des produits et la mise en évidence de la composition de FABRIQUE et USINE liées par voisinage (étapes E1 et E2) donnent le graphe G4.2 équivalent à G4.1 :



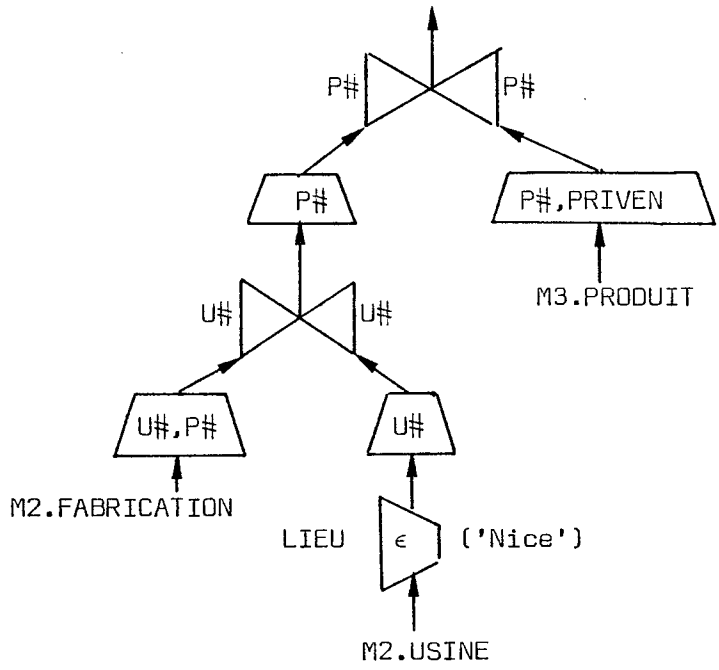
Graphe G4.2

L'étape E3 doit s'appliquer à FABRICATION et USINE. Si l'on ne combinait pas la prise en compte de la localisation et les réductions, FABRICATION devrait être remplacée dans G4.2 par le graphe G4.3.

L'application des réductions permet alors d'arriver au graphe localisé optimisé G4.4 (le critère de localisation a permis d'éliminer la machine M1).



Graphe G4.3 qui correspond à la relation globale FABRICATION



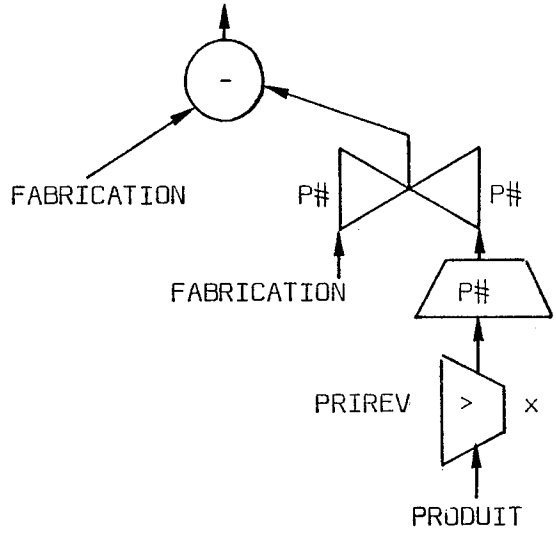
Graphe G4.4

Donnons également un exemple de modifications au niveau global. Le tableau 4.4 permet ici de prendre en compte la localisation.

Exemple 2 : Soit la requête suivante :

"Retirer de la fabrication tous les produits dont le prix de revient est > x".

Le graphe global est donné par la Figure G4.5 :



Graphe G4.5

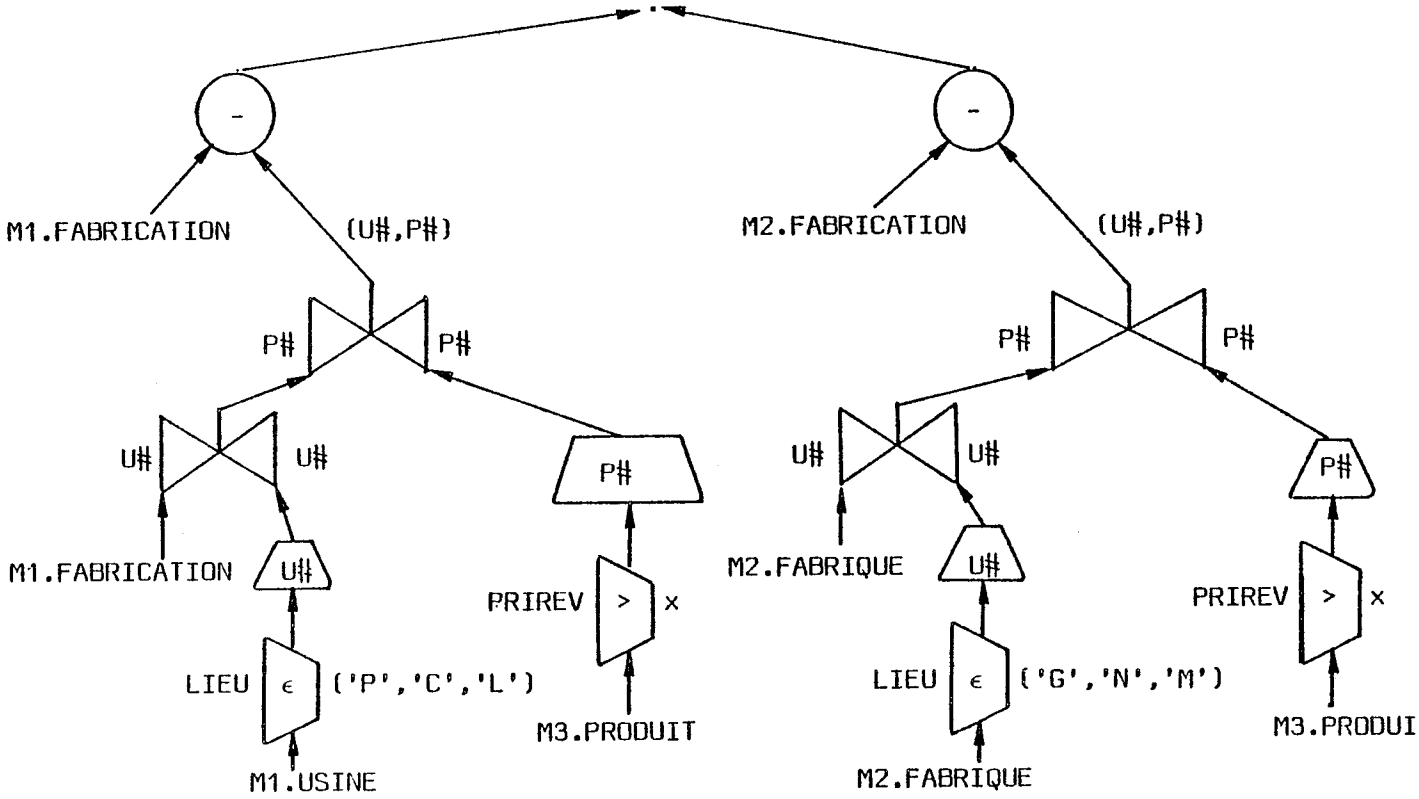
On a une opération du type G-M avec

G = FABRICATION

et M = FABRICATION[P# ∩ P#]([P#](PRODUIT : PRIREV > x)).

L'application des règles du tableau 4.4 et la prise en compte de la localisation de FABRICATION et de PRODUIT, donne le graphe G4.6.

Ce graphe donne lieu à deux transactions locales, effectuant chacune une suppression. Chaque transaction locale fait appel à des valeurs provenant de M3.



Graphe G4.6

4.3.6.4. Stratégie de découpage

La phase précédente a produit un graphe localisé où chaque opération d'obtention ou de modification de relation globale a été décomposée en opérations d'obtentions ou de modifications locales.

Déterminer un découpage du sous-graphe localisé, c'est mettre en évidence des sous-graphes mono-machines (représentatifs de requêtes mono-bases), de telle sorte que les coûts d'exécution et de communication soient minimisés.

Contrairement à l'étape précédente qui n'avait besoin que d'une vision logique de la répartition des données, élaborer une stratégie de découpage requiert une certaine connaissance de l'architecture physique du système réparti.

C'est la raison pour laquelle nous décrivons cette stratégie au chapitre suivant, qui concerne l'architecture du SGBDR (cf. § 5.5.3.3).

CHAPITRE 5

ARCHITECTURE D'UN SYSTÈME DE BASES DE DONNÉES RÉPARTIES

Je ne me défie pas de la machine
que je regarde avec curiosité
sur son socle ou sous sa verrière.
Je me défie de la machine qui est
en moi.

G. DUHAMEL

Sommaire

5.5.3. Manipulation de la base répartie.

5.5.3.1. Analyse et transcodification.

5.5.3.2. Transformations algébriques.

5.5.3.3. Prise en compte de la localisation.

5.5.3.4. Stratégie de découpage

- Découpage en sous-graphes mono-machines
- Préparation du plan d'exécution.

5.5.4. Exécution répartie.

5.5.4.1. Interpréteur global d'exécution.

5.5.4.2. Opérateurs globaux.

5.5.4.3. Services périphériques.

5.6. CONCLUSIONS SUR L'ARCHITECTURE DU SGBDR.

5.1. RAPPELS SUR LES HYPOTHESES DU PROJET POLYPHEME

Le projet POLYPHEME, développé depuis Octobre 1976, a pour but la conception et la réalisation d'un système de coopération pour des bases de données hétérogènes réparties dans un réseau général d'ordinateur comme ARPA, CYCLADES, TRANSPAC [R49, R68, R50].

Les hypothèses de travail sont les suivantes :

- les bases coopérantes sont implantées sous des systèmes existants comme IMS, IDS-II, SOCRATE et constituent les bases locales
- il existe un réseau d'ordinateurs qui interconnecte les sites sur lesquels sont implantées ces bases et qui fournit des mécanismes de base permettant l'échange d'informations entre sites.

Le sujet abordé par POLYPHEME se subdivise logiquement en deux grandes classes de problèmes :

- les problèmes liés à la sémantique des données réparties
- les problèmes liés à l'architecture logicielle d'un système permettant de faire coopérer des bases hétérogènes.

La première classe de problèmes a été abordée au cours des Chapitres 2, 3 et 4 avec la définition du modèle général de données réparties MOGADOR. Ce modèle résout en particulier :

- l'homogénéisation des modèles hiérarchiques ou réseaux utilisés pour la description de bases hétérogènes
- le problème de la granularité des données réparties, en définissant les atomes sémantiques de répartition comme des éléments de catégories et de graphes de fonctions
- l'homogénéisation des moyens de stockage et d'accès, en assimilant chaque base à une machine relationnelle, capable d'effectuer les opérations classiques d'accès et de manipulation. Cette approche induit alors l'architecture logique du système comme un réseau de machines logiques [R7].

Dans ce qui suit, nous abordons la seconde classe de problèmes : à partir des concepts de MOGADOR concernant les machines locales et globales, comment réaliser un SGBDR dans un environnement hétérogène ?

5.2. LE SYSTEME DE BASE DE DONNEES REPARTIES COMME UN RESEAU DE MACHINES RELATIONNELLES

Le SGBDR se présente comme un réseau de machines relationnelles de deux types (Figure 5.1) :

- les *Machines Globales (MG)*, chacune d'elles permettant la définition et la manipulation d'une base de données réparties (cf. § 4.3)
- les *Machines Locales (ML)* qui assurent l'homogénéisation des moyens logiciels et matériels de stockage et d'accès, en les présentant comme des automates standards (cf. § 3.3).

Nous ne faisons a priori aucune hypothèse sur l'emplacement physique d'une machine globale vis-à-vis des machines locales, en ce sens qu'une MG peut se trouver sur le même site que zéro, une ou plusieurs ML (cf. § 5.6).

Chaque machine, qu'elle soit locale ou globale, possède, sous forme de catalogues, une certaine connaissance de ce que l'on peut lui demander de faire et de la manière dont elle peut le faire elle-même ou le faire faire par d'autres machines.

Chaque machine accepte en entrée une transaction qu'elle interprète en opérations qui sont alors exécutées par elles-mêmes ou par d'autres machines qu'elle contrôle.

5.2.1. Point de vue Utilisateur

En tant que Système de base de données, le SGBDR se présente pour l'usager de la base de données répartie comme un système classique capable d'effectuer les fonctions suivantes (cf. § 1.1.2) :

- la description des données et la mise en place de la base
- la manipulation de la base en mode interactif ou en mode transactionnel.

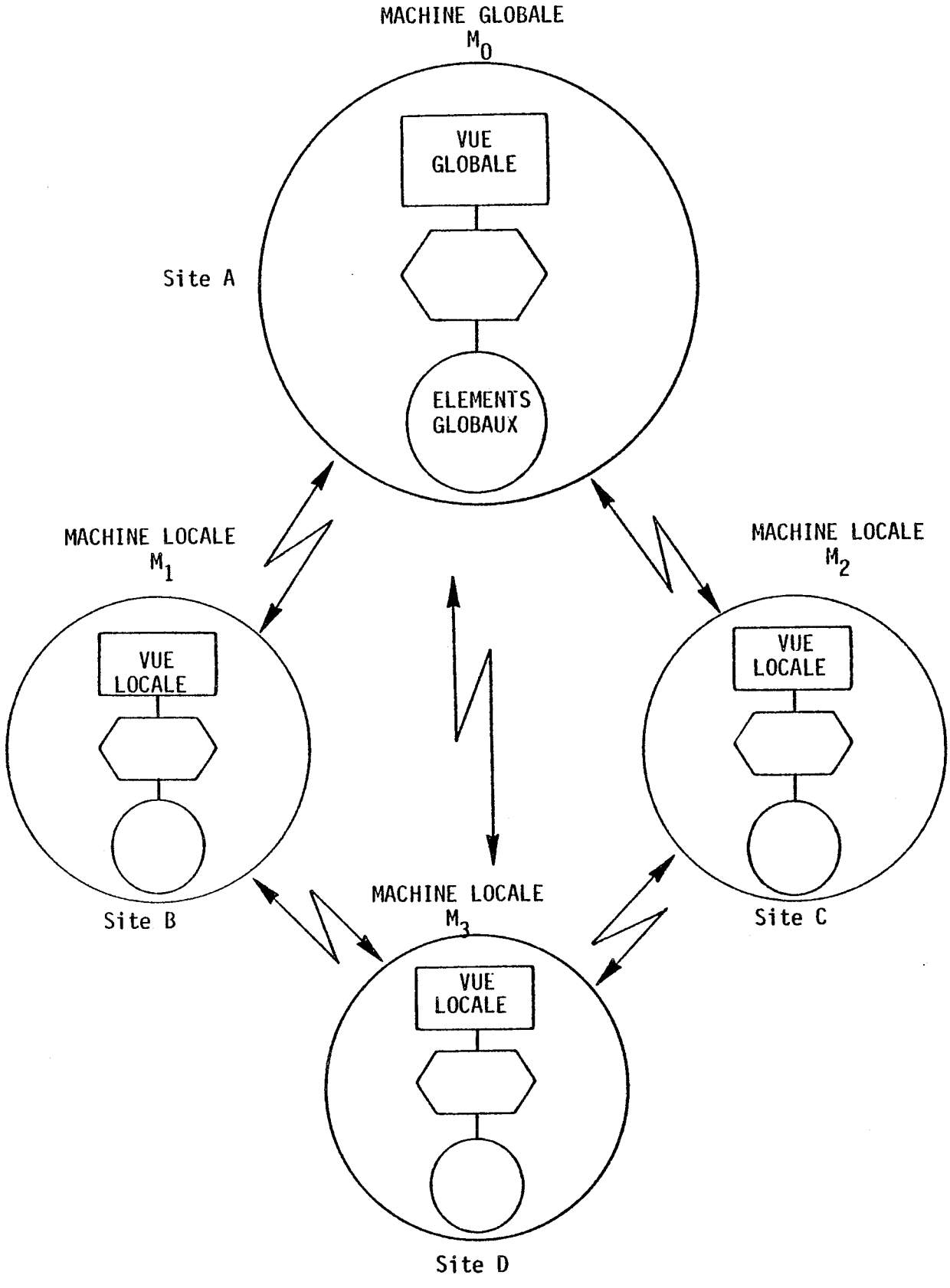


Figure 5.1 - Un réseau de machines logiques MOGADOR

Cependant, là encore, il convient de distinguer :

- d'une part les différents types d'usagers : l'administrateur davantage concerné par la fonction description, et les utilisateurs concernés uniquement par la fonction manipulation
- d'autre part les deux niveaux, à savoir local et global pour lesquels se pose le problème de la réalisation des deux fonctions.

Du fait de l'homogénéisation introduite par la nature relationnelle de MOGADOR, le langage utilisé pour décrire et manipuler des vues relationnelles, qu'elles soient locales ou globales, est le même. Ce langage que nous appelons LADORE (LAngage pour DOnnées REparties) est défini en annexe.

5.2.1.1. Fonction Description (Administration)

5.2.1.1.1. Niveau local : Administrateur local

Au niveau de la machine locale, le rôle de l'administrateur comprend :

- la *caractérisation du sous-schéma*, donc de la portion de la base locale qui va coopérer
- l'*interprétation de ce sous-schéma* en termes relationnels, grâce aux concepts de MOGADOR (cf. § 3.2) pour construire la vue locale. A ce niveau, des outils développés dans les travaux [B5, B39] peuvent être utilisés pour assister l'administrateur local dans ce processus.
- la *description des programmes locaux* qui revêt un aspect logique, avec la définition des droits que possèdera le niveau global sur le niveau local vis-à-vis des opérations d'accès et de mise à jour et un aspect technique avec l'écriture ou la génération de ces programmes.

Dans ce domaine les travaux décrits dans [B44] et développés dans le cadre du système URANUS, ainsi que ceux de [P1] dans le cadre de POLYPHEME constituent une approche qui concerne un système comme SOCRATE.

Cependant, dans la mesure où la plupart des SGBDs possèdent un interface avec le langage COBOL, il faut remarquer que l'écriture des programmes locaux peut facilement être automatisée. Citons dans ce domaine les travaux sur la portabilité des programmes utilisant un SGBD menés à l'EDF [B31].

Le résultat du travail de l'administrateur local est donc l'expression dans un langage relationnel (cf. § 2.5) de la *vue locale* avec les programmes locaux associés.

5.2.1.1.2. Niveau global : Administrateur Global

Le rôle de l'administrateur global est plus délicat dans la mesure où il lui faut effectuer le rapprochement de plusieurs vues locales fournies par les administrateurs locaux.

La conception de la Vue Globale a été étudiée au § 4.2.

Nous avons vu qu'il fallait revenir au niveau conceptuel de modélisation pour caractériser les ensembles d'éléments répartis et les différents cas de répartition.

A ce niveau, les problèmes les plus délicats proviennent de l'approche ascendante : Vues Locales vers Vue Globale et l'on peut remarquer que dans de nombreuses applications, une approche descendante s'avèrera plus facile puisque les vues locales seront déduites par partitionnement de la Vue Globale, ce partitionnement étant totalement explicite et donc contrôlé.

Le résultat du travail de l'administrateur global en collaboration avec les administrateurs locaux est donc l'expression d'une vue relationnelle globale exprimée avec le même langage que les vues locales.
En plus de cette vue globale, il faut donner (§ 4.3.1, 4.3.2) :

- la correspondance noms globaux, noms locaux
- les règles globales, soit sous forme d'expressions relationnelles, soit sous forme de programmes globaux.

5.2.1.2. Fonction Manipulation

5.2.1.2.1. "Utilisateur" local

Au niveau de la machine locale, dans le cadre du système réparti, l'usager est une machine globale. En effet, c'est à partir du processus de décomposition que l'on détermine les transactions locales qui sont transmises aux machines correspondantes (cf. § 4.3.6).

Chaque machine locale reçoit une requête exprimée en LA.DO.RE défini au Chapitre 2. Cette requête ne concerne que les relations locales sur

lesquelles des opérations de l'algèbre relationnelle sont demandées. Nous reviendrons sur ce point lorsque nous décrirons la réalisation d'une machine locale (§ 5.4).

5.2.1.2.2. Utilisateur Global

Il s'agit soit d'une personne qui au terminal émet une requête globale d'interrogation ou de manipulation, soit d'un programme si la machine globale possède un interface langage évolué.

Ce point a été traité dans les spécifications de la maquette POLYPHEME de la manière suivante :
l'interface avec l'usager global est un terminal ce qui permet dans un premier temps une utilisation interactive du système. Cependant, dans un cadre plus général de réalisation, la machine globale doit pouvoir être activée à partir d'un programme en langage évolué. Ce point mérite une étude particulière car il permet d'agrandir considérablement les possibilités du système et de lui offrir des débouchés dans le domaine industriel.

5.2.2. Aspects fonctionnels du SGBDR

Après avoir considéré le point de vue externe de l'utilisateur, nous abordons l'aspect interne du système, en considérant les fonctions qu'il a à réaliser à partir d'une demande externe.

La *fonction description* correspond à la construction de catalogues à partir d'une vue relationnelle donnée dans le langage de description. Cette fonction n'est pas spécifique aux SGBDR et se retrouve dans tous les SGBD, aussi la laissons-nous de côté pour le moment pour y revenir au § 5.5.

En supposant que ces catalogues aient été mis en place par l'administrateur global, voyons quelles opérations doit réaliser le SGBDR pour fournir la réponse à une transaction globale.

La Figure 5.2 montre les fonctions à réaliser aux différents niveaux du système :

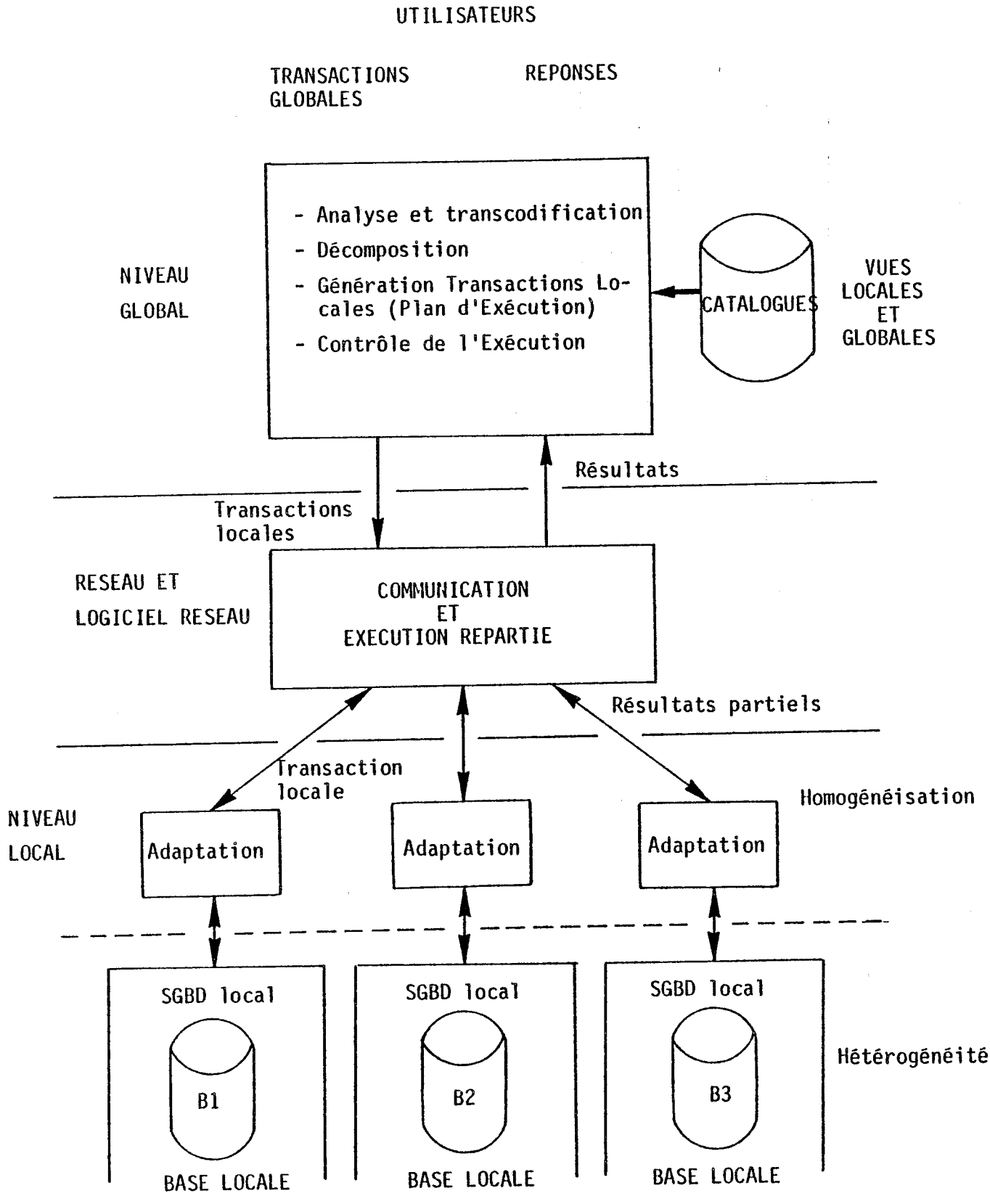


Figure 5.2 - Les principales fonctions du SGBDR

- Au *niveau global* (machine globale), la prise en compte de la transaction globale (TG) provoque son analyse syntaxique et sa transcodification sous forme interne (un graphe dans POLYPHEME) après contrôle de validité en fonction du catalogue de vue globale. A chaque TG, est alors appliqué un processus de décomposition, de manière à déterminer les transactions locales (TL1, TL2, ..., TLn) qu'il faut exécuter pour répondre à la demande de la TG (cf. § 4.3.6).

La partie génération a pour but, à partir du graphe localisé (cf. § 4.3.3), de générer les transactions locales sous forme compréhensible pour le niveau local. A ce sujet, deux solutions ont été étudiées pour POLYPHEME :

- *solution 1 :* générer dans un langage donné (PL/1 a été envisagé) chaque transaction locale. L'avantage est qu'une fois la transaction locale générée, elle peut être conservée et reexécutée autant de fois que possible. L'inconvénient est que la génération est alors une opération très complexe qui a été étudiée en particulier par C. EUZET [R31].
- *solution 2* ou solution interprétative, qui consiste à détacher du graphe localisé la portion qui correspond à la transaction locale et à transmettre le tout au niveau local qui est chargé de l'interpréter. Cette solution est celle choisie pour la maquette POLYPHEME et nous y revenons au § 5.5.4.

Dans tous les cas, nous *appellerons Plan d'Exécution (ou PEX)* l'ensemble des opérations qui correspond à la décomposition d'une transaction globale. Ces opérations sont soit des transactions locales, soit ce que nous appellerons des opérations charnières qui sont celles qui permettent l'enchaînement de deux transactions locales à des bases différentes (cf. § 5.5.4). Ces fonctions de la machine globale sont analogues à celles d'une compilation (Figure 5.4).

Un plan d'exécution est un programme dont certaines parties peuvent s'exécuter en parallèle puisqu'elles concernent des machines différentes. Le problème est alors d'une part d'exécuter à partir du niveau global des programmes situés sur d'autres sites, et d'autre part de contrôler et de synchroniser l'exécution de tous ces programmes.

Dans le cadre de POLYPHEME, E. André et P. Decitre ont conçu et réalisé un logiciel réseau (implanté sur CYCLADES) qui permet de résoudre ce problème [R11]. Ce logiciel permet à des procédures écrites en PL/1 et situées sur des sites différents de s'exécuter en parallèle, l'une sous contrôle de l'autre et de communiquer entre elles. Nous reviendrons sur cette fonction au § 5.3.

Grâce à la partie "communication et exécution répartie" (Figure 5.2), il est ainsi possible de contrôler l'exécution des transactions locales.:

- Au *niveau local*, chacune de ces TL exprimées dans le même langage (homogénéisation MOGADOR) doit être adaptée au SGBD local, avant que ce dernier puisse fournir les données locales demandées.

Une fois extraites de la base locale, ces données doivent également être adaptées à la vue que l'on en a du niveau global ; cette adaptation nécessite des conversions de format, de code de représentation et, même si elle ne pose que des problèmes techniques, peut s'avérer coûteuse et délicate à mettre en place sur des SGBD très hétérogènes.

Converties dans la forme demandée, les données locales voyagent au travers du réseau pour être traitées au niveau global qui :

- soit les utilise comme paramètres pour une autre transaction locale
- soit les adapte pour constituer la réponse finale.

Dans ce qui suit, nous allons développer ces aspects fonctionnels pour voir le détail de l'architecture logicielle du niveau global avec la réalisation d'une machine locale, et également du niveau fourni par les logiciels réseaux et qui se trouvent répartis dans les différentes machines.

5.3. COMMUNICATION ENTRE LES MACHINES

5.3.1. Réseau de Communication

Tout réseau d'ordinateurs fournit un certain nombre de mécanismes de communication de base sur lesquels peuvent se greffer les applications réparties.

Ces mécanismes classés en différentes couches ont d'ailleurs fait l'objet dernièrement de tentative de normalisation par le groupe ISO [R40, R8].

En ce qui nous concerne, nous retiendrons les couches suivantes que l'on trouve dans le réseau CYCLADES (Figure 5.3).

4	Protocole de Communication	Protocole App. Virtuel par exemple
3	Station de Transport	Station de Transport (ST)
2	Transport	- Commutation de paquets
1	Transmission	- Lignes - Contrôleurs de lignes

Figure 5.3 - Couches Réseaux dans CYCLADES

Au niveau de la station de transport, on peut échanger des informations moyennant un ensemble de règles précises concernant le format et la nature de ces informations. Cet ensemble de règles constitue un protocole de communication comme le protocole Appareil Virtuel destiné à l'utilisation de terminaux hétérogènes dans CYCLADES [R49].

Dans POLYPHEME, le problème posé par l'échange d'informations entre machines locales et globales d'une part et le problème de l'exécution répartie d'autre part ont donné lieu à la définition de deux couches logicielles, l'une se situant au niveau 4 avec un protocole de données réparties et l'autre au niveau 5 avec le moniteur d'exécution répartie.

Ces deux couches sont définies et réalisées par l'équipe réseau du Centre Scientifique CII-HB, E. André et P. Decitre [P7, P16].

5.3.2. Protocole de transport de données

Il est décrit en détail dans [P16] et est un protocole très général capable de transporter toutes sortes de données structurées. Il s'applique en effet aussi bien à des données relationnelles "plates" qu'à des données structurées en hiérarchies ou en réseaux (type CODASYL).

5.3.3. Moniteur d'exécution répartie

Il constitue à lui seul ce que l'on peut appeler la "machine d'exécution répartie". L'idée principale d'un tel mécanisme est de fournir à des programmeurs d'applications réparties un outil pour écrire et exécuter des programmes dont les morceaux peuvent se situer sur des sites différents. Cela suppose tout d'abord un langage dans lequel peuvent être écrits de tels programmes. Le choix fait ici s'est porté sur PL/1 qui, avec la notion de procédure, apporte l'unité programmable de répartition.

Des mécanismes spéciaux doivent être mis en place sur chacun des sites concernés par l'application. Ces mécanismes sont réalisés par une copie du moniteur d'exécution répartie (MER), placée sur chacun des sites (Figure 5.4).

Les différentes copies du moniteur dialoguent entre elles au travers du réseau et offrent localement les fonctions suivantes :

- possibilité pour une procédure utilisateur de demander le lancement d'une procédure située sur un site donné
- possibilité de passer dans les paramètres de lancement le nom d'une procédure, que la procédure lancée devra exécuter à distance lorsqu'elle aura terminé son exécution
- synchronisation des différents appels de manière à assurer une exécution cohérente du programme d'application
- activation locale d'un programme demandé par un autre moniteur.

Ces mécanismes sont utilisés par POLYPHEME pour l'exécution des PEX (Plan d'exécution) provenant de la transaction globale décomposée.

Afin d'illustrer le fonctionnement de ce moniteur, considérons un exemple simple.

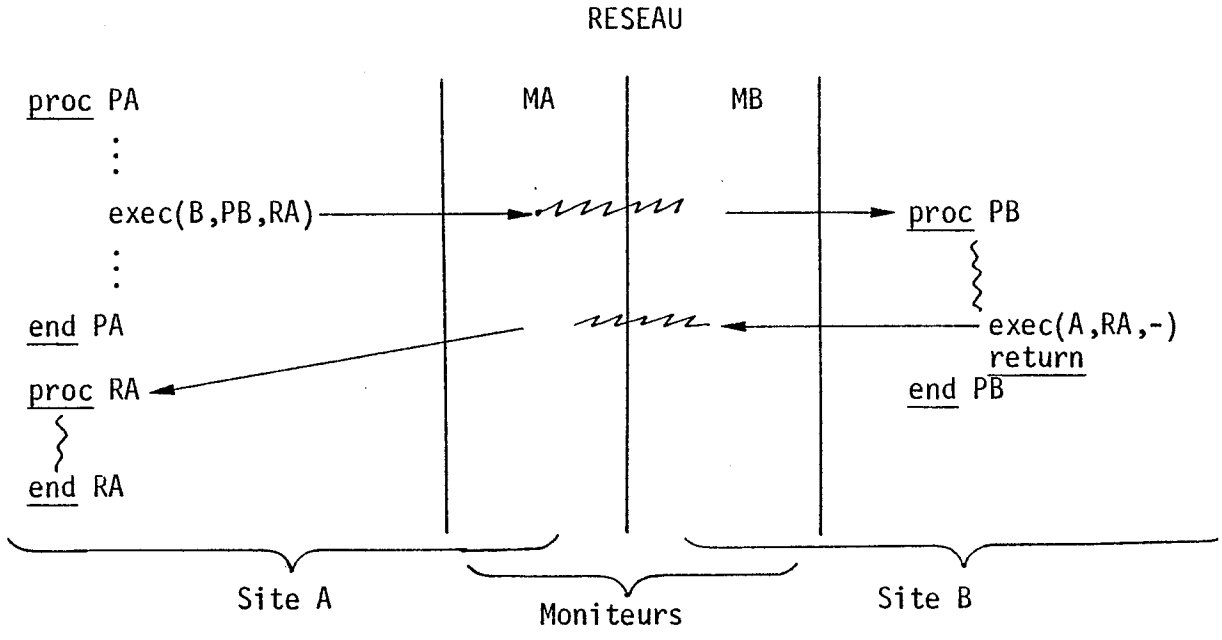
Exemple d'exécution répartie :

Considérons deux sites A et B avec :

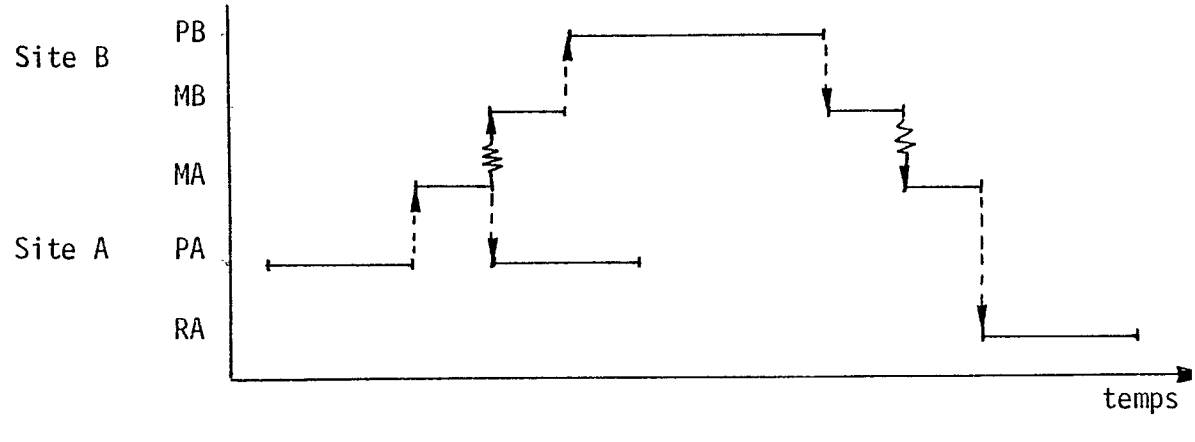
- sur A la procédure PA et la procédure RA. Notons MA le moniteur associé au site A
- sur B la procédure PB qui est appelée par PA avec en paramètre RA qui est la procédure à exécuter lorsque PB a fini.
  Soit MB le moniteur associé au site B.

Notons exec (Y,P,RP) l'ordre destiné au moniteur du site X qui signifie : "exécuter sur le site Y la procédure P qui, lorsqu'elle aura fini, activera RP".

Nous avons alors :



La figure suivante montre le parallélisme entre les sites :



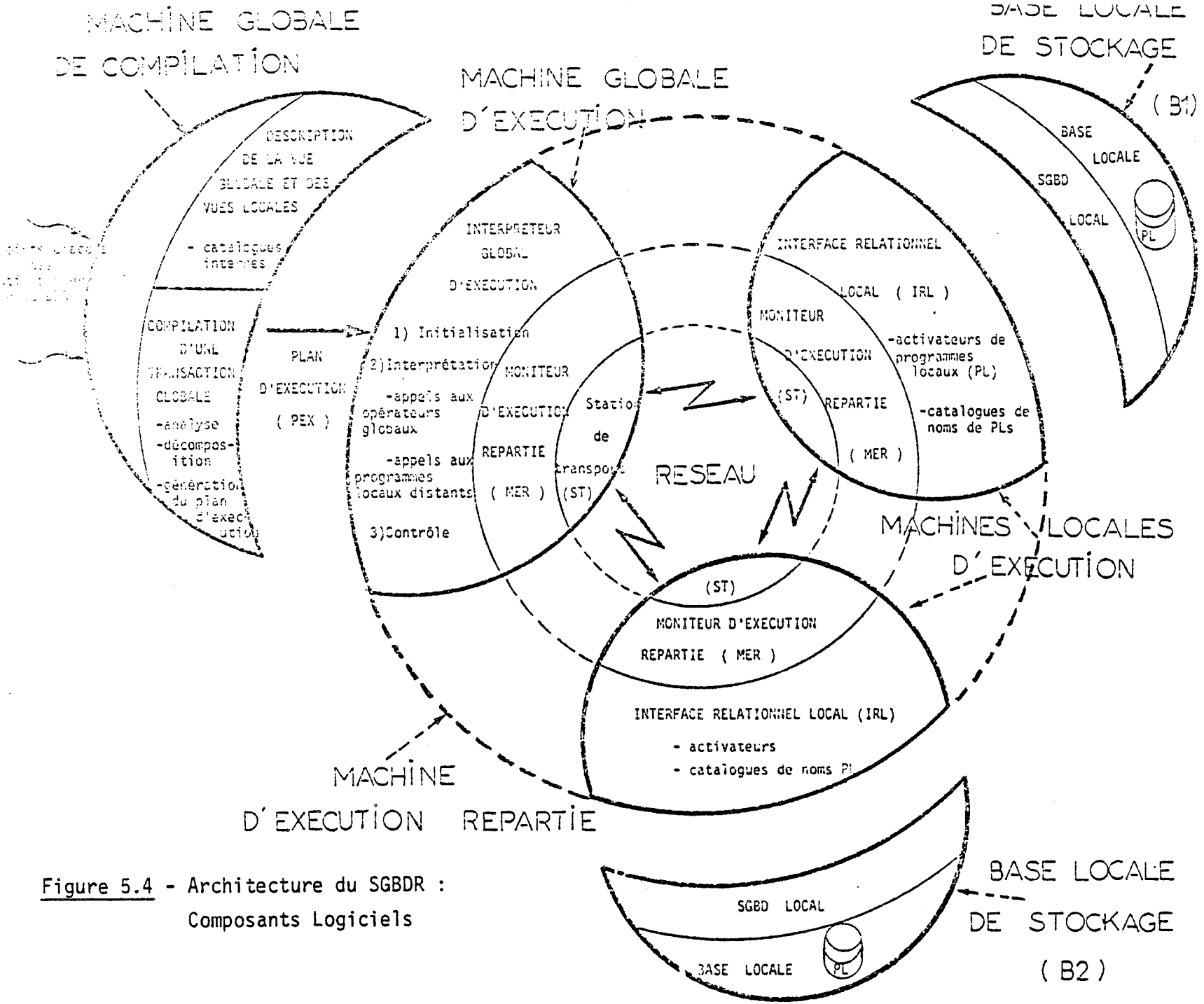


Figure 5.4 - Architecture du SGBDR :
Composants Logiciels

Comme on peut le constater sur cet exemple, le principal problème à résoudre ici est la gestion de l'asynchronisme qui trouve ici une dimension différente de celle que l'on rencontre dans les systèmes centralisés : une "entrée-sortie" dans un réseau est de l'ordre de 10 à 5000 ms contre 0.1 à 1 ms dans un environnement classique.

Le plan d'exécution visualisé sous forme de graphe localisé résultat de la décomposition (cf. § 4.3.6) n'est rien d'autre qu'un programme "parallèle" et son exécution rejoint tous les problèmes liés aux systèmes parallèles [G4, G6, G7, G8].

En particulier, pour une machine M donnée, nous avons les deux possibilités :

1. Lancement en parallèle de plusieurs autres machines (Figure 5.5) (notion "d'émetteur", cf. § 5.5.4, analogue à la notion de collatérale dans [G14]).

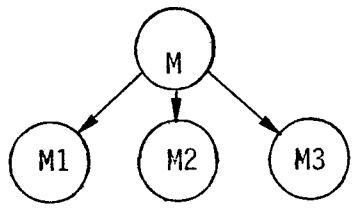


Figure 5.5

2. Regroupement d'évènements (arrivée de données, de messages) provenant de machines différentes (Figure 5.6) (notion de "collecteur", § 5.5.4).

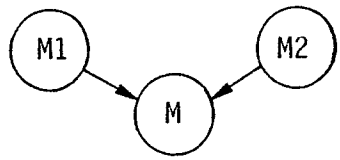


Figure 5.6

Dans un environnement réparti, les différentes machines communiquent entre elles mais n'ont pas de mémoire commune.

De ce fait, elles ne peuvent se partager des variables, ce qui permettrait l'utilisation de techniques devenues classiques en programmation système comme les sémaphores [G2].

L'approche qui est alors envisagée laisse au programmeur d'applications réparties le soin de gérer ses asynchronismes, en lui fournissant grâce à la primitive "exec P sur site X, réponse RP", la possibilité de lancer plusieurs exécutions parallèles (Figure 5.5), et grâce à la notion de procédure réponse, la possibilité de s'apercevoir de la fin de ces exécutions ; il suffit pour la machine M de n'avoir qu'une seule procédure réponse pour l'exécution sur M1 et sur M2 (Figure 5.6).

Une autre charge du moniteur d'exécution répartie est la gestion des pannes (cf. § 1.3.5.6). En s'appuyant sur des mécanismes fournis par les couches réseaux inférieures - en ce qui concerne en particulier la détection d'un site en panne - le moniteur fournit un service de surveillance qui active alors une procédure spéciale.

Comme on vient de le voir, ce moniteur d'exécution réparti fournit une couche de base pour la construction d'applications réparties et de ce fait est un composant fondamental du système POLYPHEME. Cependant, les mécanismes développés et le fait que les services qu'il offre sont disponibles dans un langage évolué, en font un outil très général [P11].

Dans le même domaine, le travail effectué par l'équipe de G. Sergeant et M. Dang avec le projet IGOR [R25] est d'un grand intérêt en ce qui concerne la mise en oeuvre des bases de données réparties et particulièrement des systèmes conçus comme un réseau de machines logiques homogènes.

La principale différence du système IGOR avec le moniteur d'exécution répartie (MER) est que le premier offre un transport d'algorithmes et de ce fait une exécution dynamique des programmes répartis.

Alors qu'avec le MER, il faut préparer sur chaque machine les procédures qui vont être activées, puis lancer et contrôler ces activations ; dans IGOR, le programmeur d'application répartie a la possibilité sur un site S d'écrire et de compiler un programme contenant les procédures qui seront activées sur S, mais également celles qu'il faudra transmettre et activer sur d'autres sites S1, S2, S3.

Il appartient alors au système IGOR de traduire le tout en morceaux locaux qui seront transmis puis interprétés sur place par le moniteur IGOR associé à chacun des sites du réseau.

De plus, le langage mis à la disposition du programmeur (englobé aussi dans PL/1) permet la déclaration de variables de types particuliers (rendez-vous, père fils, événement) grâce auxquelles il peut décentraliser davantage les traitements : ainsi deux machines peuvent-elles communiquer entre elles sans passer par le site centralisateur. Les coûts de transferts d'information s'en trouvent nécessairement réduits.

5.4. REALISATION D'UNE MACHINE LOCALE

Ce paragraphe est consacré à la réalisation effective d'une machine locale MOGADOR dans un environnement matériel et logiciel particulier.

La fonction principale d'une telle machine est d'homogénéiser une base de données hiérarchique ou réseau de sorte qu'elle se présente de manière relationnelle pour l'utilisateur de la machine locale (cf. Chapitre 3).

Au niveau de la réalisation d'une telle machine, nous distinguons deux parties (cf. Figures 5.4 et 5.7) :

- la première concerne l'interface avec le SGBD local de manière à pouvoir lui faire exécuter les opérations élémentaires d'accès et de mise à jour des données de la base locale (programmes locaux)
- la seconde concerne l'interface avec l'utilisateur de la machine locale qui est chargée de recevoir et d'exécuter une transaction exprimée de manière relationnelle.

Dans ce qui suit, nous allons décrire ces deux parties en nous plaçant sous deux angles différents :

1) l'angle de l'architecture du SGBDR telle qu'elle devrait être dans le cas général.
2) l'angle de l'architecture de la maquette POLYPHEME telle qu'elle est implantée. Dans ce cas, les solutions techniques choisies sont volontairement plus simples, pour répondre à un double objectif :
   - arriver à valider les principaux choix faits au niveau des hypothèses de travail (modèle, langage, exécution répartie) et

. se fixer des buts de réalisation à court terme.

5.4.1. Interface avec SGBD local et programmes locaux

Dans tous les cas, il faut accéder aux objets locaux via le SGBD local, pour respecter la cohérence de la base. Le problème se pose d'activer par programme un SGBD en lui demandant l'exécution de plusieurs opérations. Tous les systèmes de bases de données commercialisés sur le plan industriel offrent ce genre de possibilité avec, la plupart du temps, le langage COBOL comme langage de programmation.

Dans SOCRATE version CII [B56], on peut dans un programme COBOL faire un appel au système pour lui demander l'une des quatre fonctions suivantes (cette possibilité a été étendue au langage PL/1, voir [B45]) :

- F1 - ouverture de la base Bi
- F2 - fermeture de la base Bi
- F3 - exécution d'un ou plusieurs programmes précompilés
- F4 - sauvegarde de la base Bi.

La fonction F3 est la plus importante puisque c'est elle qui permet l'accès et la modification des données enregistrées par l'intermédiaire d'une zone de travail partagée par SOCRATE et le programme utilisateur. Elle implique que les programmes aient été au préalable précompilés et stockés dans le système. Ceci peut être fait par l'administrateur local au moment de la mise en place de la base répartie.

Dans le projet POLYPHEME, une première version d'un tel interface a été réalisée par J.Y. Caleca et A. Stiers [P1] tandis que dans [B44] c'est un moyen analogue qui permet de "coiffer" SOCRATE d'un chapeau relationnel.

Dans les systèmes de la famille CODASYL, le langage de programmation des applications sur une base de données est un COBOL dans lequel les primitives de manipulation CODASYL (FIND, GET, MODIFY, INSERT, REMOVE, DELETE, STORE) ont été incorporées en tant que verbes du langage.

De ce fait, les programmes locaux peuvent être directement écrits dans ce langage et être appelés de l'extérieur de la machine locale.

Il en résulte au niveau de l'architecture de la machine locale (Figure 5.7) une partie consacrée à l'activation des programmes locaux, tandis que doivent figurer dans une bibliothèque les informations concernant la vue locale et la liste des programmes locaux correspondants (cf. § 3.3).

5.4.2. Interface relationnel base locale

Le rôle de cette partie est d'accepter en entrée une requête élémentaire, exprimée sous forme relationnelle pour la traduire en activations de programmes locaux.

Ce problème n'est pas simple à résoudre dans le cas général de n'importe quelle base, aussi nous sommes-nous contentés dans le cadre de la maquette POLYPHEME d'avoir des programmes locaux réalisant les opérations de sélection et de projection (cf. § 3.3).

Dans le cas général, il faut disposer :

- d'un analyseur de requêtes relationnelles
- d'un traducteur de requêtes élémentaires en appels aux opérations primitives du langage de manipulation et en appels aux opérateurs de l'algèbre relationnelle
- d'un interpréteur pour réaliser l'interface avec le système local.

(Tous ces composants se retrouvent au niveau de la machine globale, cf. § 5.5).

5.4.3. Bases Locales

Dans l'approche conceptuelle que nous avons choisie, une base locale est assimilée à une machine logique. Sur le plan de l'architecture, la construction d'une machine locale est facilitée par le découpage en parties dépendantes ou indépendantes vis-à-vis du SGBD local et/ou de la base elle-même :

- l'interface relationnel est a priori indépendant et du SGBD et de la base
- l'activateur de programmes locaux est indépendant de la base, mais dépend d'un type de SGBD : il sera identique pour toutes les bases SOCRATE, par exemple
- les programmes locaux, eux, sont bien sûr dépendants de la base.

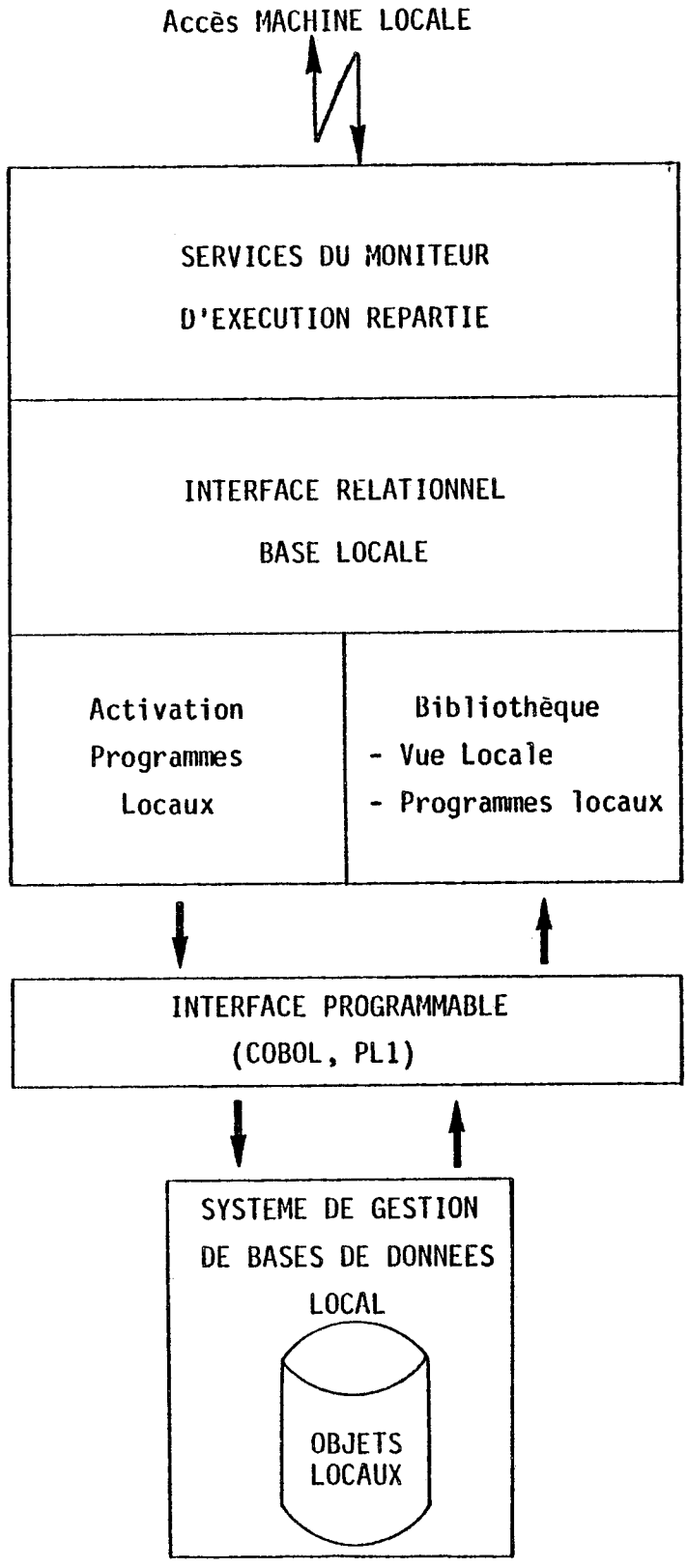


Figure 5.7 - La réalisation d'une machine locale

Dans le cadre du prototype POLYPHEME, nous mettons en oeuvre un exemple concret fourni par le CNET [P18] nécessitant la coopération de bases SOCRATEs :

- les unes faisant l'inventaire de matériel de centraux téléphoniques (deux bases aux structures très voisines)
- l'autre contenant le catalogue du matériel.

Cet exemple présente une certaine analogie avec celui développé au long des chapitres précédents.

Une autre application, envisagée sur le prototype et à laquelle tous les participants au projet SIRIUS travaillent, est l'application AGORA concernant des services de courrier électronique (échange de messages, de documents) [P19].

5.5. REALISATION D'UNE MACHINE GLOBALE

La machine globale joue un rôle prépondérant dans l'architecture du SGBDR puisqu'elle assure vis-à-vis de son utilisateur les mêmes fonctions que celles fournies par un système classique, tout en adaptant ces fonctions à la répartition des composants de la base.

Pour décrire les composants de la machine globale, nous allons partir des couches les plus basses assurant le stockage et l'accès des informations nécessaires au fonctionnement de ce niveau, pour aller jusqu'à la partie concernant l'exécution répartie, ceci en passant par l'aspect spécifique de la machine : décomposition et interprétation d'un plan d'exécution (Figure 5.8).

5.5.1. Stockage Relationnel ("LAMB")

Pour pouvoir fonctionner, la machine globale est en quelque sorte "paramétrée" par toute une série de catalogues (Vues Locales, Vue Globale). De manière homogène, ces catalogues ont eux-mêmes une structure relationnelle et nous avons défini une série de moyens de stockage et d'accès de données relationnelles.

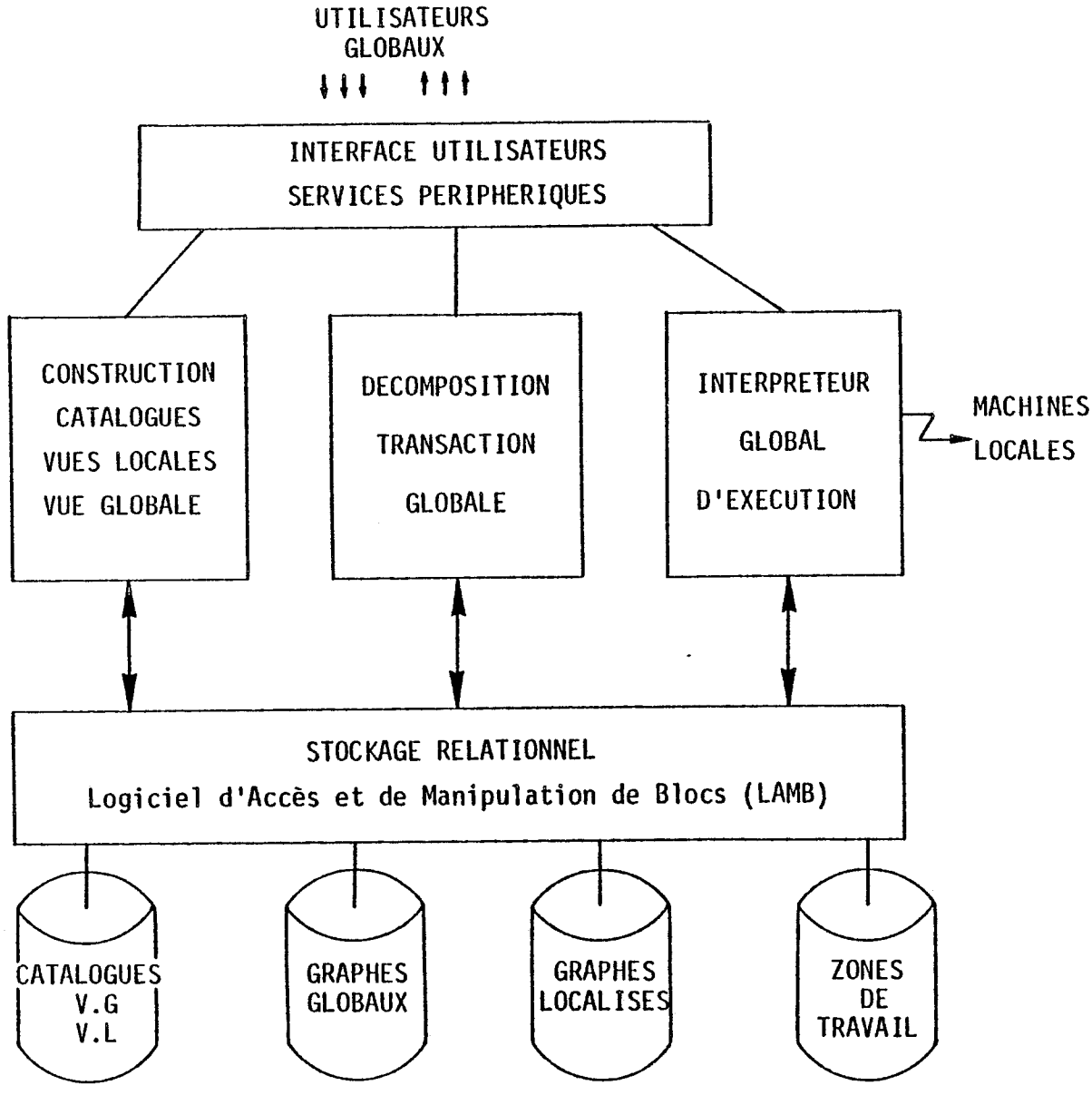


Figure 5.8 - Les Composants d'une machine globale

Ces moyens sont fournis par un Logiciel d'Accès et de Manipulation de Blocs (ou LAMB) qui, via un interface, assure des fonctions de base de stockage et d'accès.

Ce composant a été réalisé par J.M. Andrade et est décrit en détail dans [P9, P15].

Nous allons en donner les principales caractéristiques :

Via l'interface, l'utilisateur de LAMB manipule des *tableaux* d'informations. Chaque tableau est un ensemble de *n-uplets* et chaque n-uplet est composé de *champs* ayant une certaine longueur et contenant une valeur.

Chaque tableau a un *type* qui induit des caractéristiques structurelles : nombre de colonnes (degré), longueur de chaque champ, champs clés.

Les tableaux sont repérés par un identificateur de tableau (idt) tandis qu'un n-uplet est repéré par un identificateur de n-uplet (idu) ; ces deux identificateurs servent au niveau de l'interface.

Le tableau est l'unité logique que manipule l'utilisateur de LAMB, alors que sur le plan physique, le bloc constitue l'unité de travail : un tableau peut ainsi se trouver sur plusieurs blocs, mais cette notion est transparente pour l'utilisateur.

L'interface d'utilisation de LAMB est constitué par un ensemble de primitives, appelables dans un programme PL/1 ainsi que par un ensemble de règles concernant les passages de paramètres et les codes erreurs.
Les primitives sont soient fonctionnelles, si elles ne renvoient qu'une valeur, soit procédurales, si elles en retournent plusieurs.

On trouve dans LAMB, les opérations suivantes :

1. Sur les types :
   - définition d'un type de tableau (DTYP)
   - suppression de la définition (TTYP)

   (un type est une chaîne de 4 caractères).

2. Sur les tableaux :
   - définir un tableau d'un type donné (DTAB) et lui affecter un idt
   - supprimer une occurrence de tableau (TTAB)

- retrouver l'identificateur d'un tableau (LIDT)
- connaître les caractéristiques d'un tableau (LCRTL).

3. Sur les n-uplets
   - créer un n-uplet dans un tableau (CNPL)
   - supprimer un n-uplet (TNPL)
   - lire les valeurs des champs d'un n-uplet (LNPL)
   - obtenir le n-uplet suivant (SNPL)
   - rechercher un n-uplet par la valeur du champ clé (RCNPL)
   - rechercher un n-uplet par sa position (RDNPL)
   - modifier les valeurs d'un n-uplet (MNPL).

Grâce à ces primitives, les autres composants de la machine globale vont utiliser LAMB pour :

- le stockage et l'accès aux catalogues vues locales et vue globale
- le stockage et l'accès des graphes provenant des divers stades de traitement de la transaction globale (Graphes globaux, localisés, cf. § 5.5.3 et 5.5.4)
- le stockage et l'accès aux informations provenant des bases coopérantes (zones de travail) au cours de l'exécution d'une transaction globale.

Une première version de LAMB est d'ores et déjà opérationnelle. Certes, dans cette première implantation, les performances n'ont pas été le premier but visé. Cependant, elle constitue à notre sens un point de départ pour la réalisation d'un composant plus complet et plus performant, car il s'agit là d'un élément de base du SGBDR.

5.5.2. Description des Vues Relationnelles

Les vues locales et globales constituent un ensemble d'informations qui est vu également de manière relationnelle. Cela signifie qu'une vue est en fait un ensemble de tableaux gérés grâce au logiciel LAMB (cf. § 5.5.1).

Grâce à la partie description du langage LADORE (cf. Annexe), l'utilisateur de la machine globale donne l'expression des vues locales et de la vue globale.

Au niveau de la machine globale, cette description est transformée en catalogues internes dont la structure est la suivante (cf. [P14] et annexe) :

5.5.2.1. Catalogue Vue Globale

Le Catalogue de la Vue Globale comprend trois types de relations :

1. les *relations administratives* décrivant la vue globale, avec son nom, le nom de l'administrateur et la date à laquelle elle a été créée. On y trouve également les vues locales à partir desquelles est construite la vue globale, ainsi que la liste des utilisateurs globaux avec leurs droits.
Ces droits indiquent qu'un utilisateur ne peut faire que de l'interrogation ou qu'il peut lire et modifier les données réparties.

2. les *relations constituant le schéma relationnel :*

2.2. liste des domaines avec le type de valeurs associé (ENTIER, CHAINE, ...) et une éventuelle restriction.

2.2. liste des relations globales avec pour chacune d'elles :

- son nom
- la liste des attributs et le domaine associé
- la liste des attributs formant la clé
- la liste des attributs formant une clé pour une autre relation (liaisons entre relations).

3. les *relations décrivant la correspondance Global/Local :*
Nous avons fait à ce niveau un choix concernant les critères de partitionnement ; on prend en compte les six critères évoqués au § 4.2.7 :

- partitionnement par restriction (PAR)
- partitionnement par voisinage (PAV)
- duplication simple ou répartition triviale (DUS)
- duplication totale (DUT)
- partitionnement avec duplication par restriction (PDR)
- partitionnement avec duplication par voisinage (PDV).

Comme nous l'avons montré, ces critères induisent des règles standards concernant les opérations d'accès et de modification (§ 4.3.4).

Pour chaque relation globale, il faut donc indiquer :

- le type de répartition
- dans le cas d'un partitionnement par voisinage, la relation voisine
- les différents morceaux locaux (relations ou sous-relations locales) la composant. Ceci permet en particulier d'établir la correspondance noms globaux - noms locaux sur les relations

- les équivalences sur les noms de domaines et d'attributs entre le niveau global et local.

5.5.2.2. Catalogues Vues Locales

Les vues locales au niveau de la machine globale ne concernent que la description des relations locales et non les mécanismes de correspondance noms-éléments, qui eux sont réalisés par la machine locale.

On trouvera alors deux types de relations :

- les relations administratives pour décrire la vue et la base locale sur laquelle elle est construite (nom de base, nature du SGBD)
- les relations décrivant la vue relationnelle locale et qui sont analogues à celles constituant le schéma relationnel global.

5.5.3. Manipulation de la base répartie

A travers un interface utilisateur, la machine globale reçoit une transaction à exécuter. Cette transaction est exprimé dans le langage de manipulation (LADORE) donné en annexe.

La prise en compte d'une transaction globale est illustrée par la Figure 5.9. Elle se découpe logiquement en plusieurs phases :

5.5.3.1. Analyse et Transcodification

Cette phase a pour objet de réaliser l'analyse syntaxique de la transaction globale mais également sa mise sous forme interne qui correspond au codage d'un graphe global évoqué en 4.3.6. Le format précis de ce codage figure dans [P14] et nous y revenons au § 5.5.3.3.2.

Pour se réaliser, cette phase nécessite l'accès au catalogue Vue Globale stocké grâce au logiciel LAMB (cf. § 5.5.1).

Le résultat est donc un graphe global stocké également sous forme de tableaux LAMB (Figure 5.9).

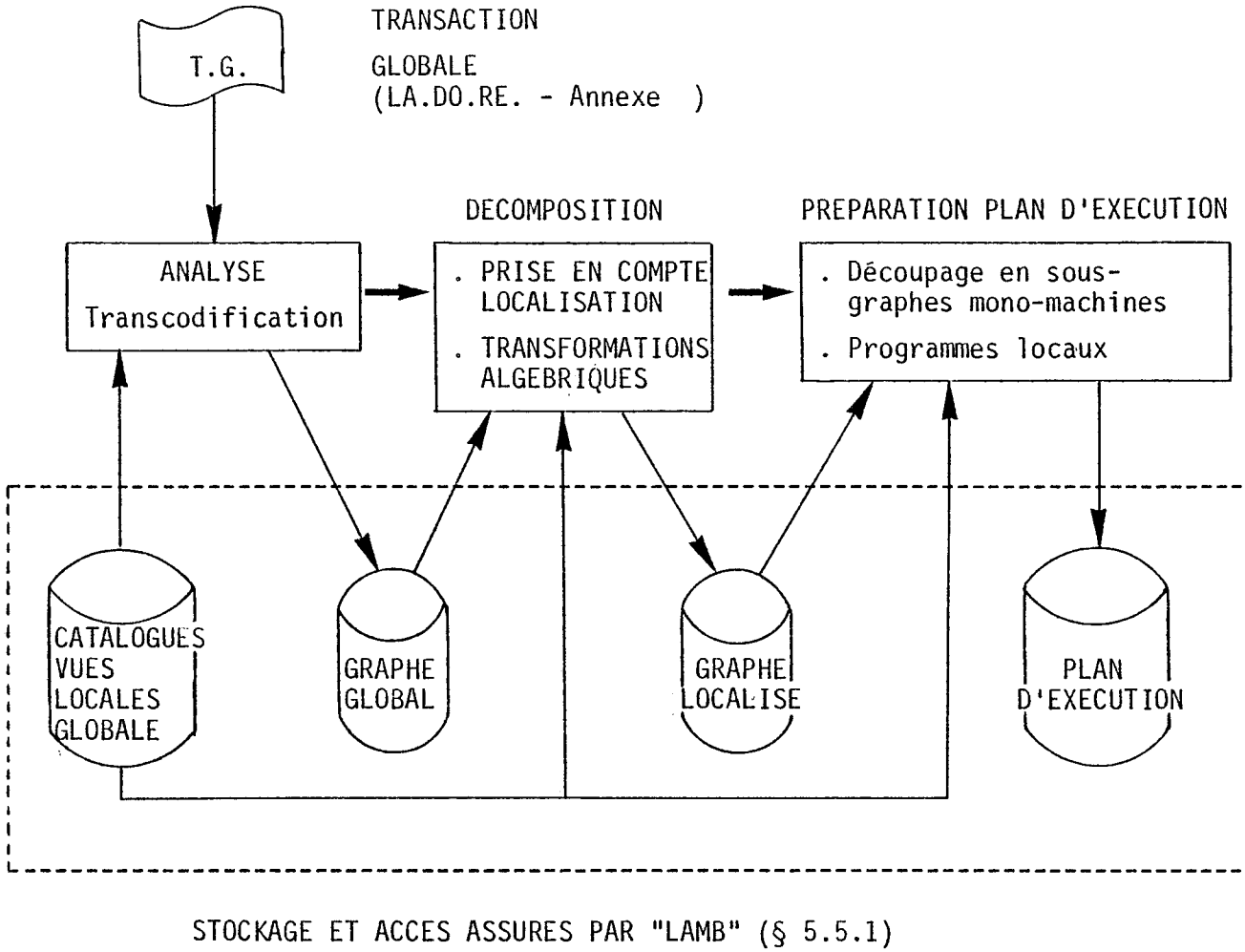


Figure 5.9 - La prise en compte d'une transaction globale

5.5.3.2. Transformations Algébriques

Comme nous l'avons vu au § 4.3.6.3, elles ont pour but de réaliser un certain nombre de transformations du graphe global pour en faire un graphe optimisé [R17].

5.5.3.3. Prise en compte de la localisation

Cette phase est décrite au § 4.3.6 et elle a pour but de produire un graphe localisé (et optimisé) représentant les transactions locales. Ces dernières sont exprimées à l'aide d'opérateurs relationnels, en faisant l'hypothèse que chaque machine locale est homogène quant à sa construction : elle dispose effectivement d'un jeu complet d'opérateurs

relationnels et d'un interpréteur associé.

Si les machines locales ne sont pas homogènes, la stratégie de découpage permettra de mettre en évidence des sous-graphes mono-machines, en fonction des opérateurs disponibles localement.

5.5.3.4. Stratégie de Découpage

Pour assurer finalement l'interprétation et l'exécution du graphe localisé, il est nécessaire de l'adapter à l'environnement réparti. En effet, le processus de décomposition procure un graphe décrivant l'enchaînement logique des opérations et les flots de données associés, mais n'exprime pas la synchronisation des différentes opérations.

Pour cela, il est nécessaire, à partir du graphe localisé, de produire un véritable plan d'exécution (cf. § 5.5.2) faisant apparaître cette synchronisation. Ce processus est réalisé en deux temps :

1. découpage en sous-graphes mono-machines
2. préparation du plan d'exécution.

5.5.3.4.1. Découpage en sous-graphes mono-machines

Rothnie et Goodman ont montré dans [B38], en considérant le réseau ARPANET, avec,sur une ligne de télécommunication,une capacité et un délai de transmission moyens (10 Kbits/s et 1 seconde) :

1) que les stratégies pour l'exécution d'une requête globale pouvaient engendrer de grandes variations dans les temps de réponse.
2) que ces variations dépendaient
   - a - du nombre d'interactions site à site
   - b - de la quantité totale de données transmises par interaction.
3) que les meilleures stratégies dans un réseau de ce type sont celles qui minimisent a et b et qui fournissent l'opportunité d'exécutions parallèles.

Dans le cadre de POLYPHEME qui est réalisé sur CYCLADES, ce réseau se caractérise par un faible débit et un délai de transmission élevé (commutation de paquet).

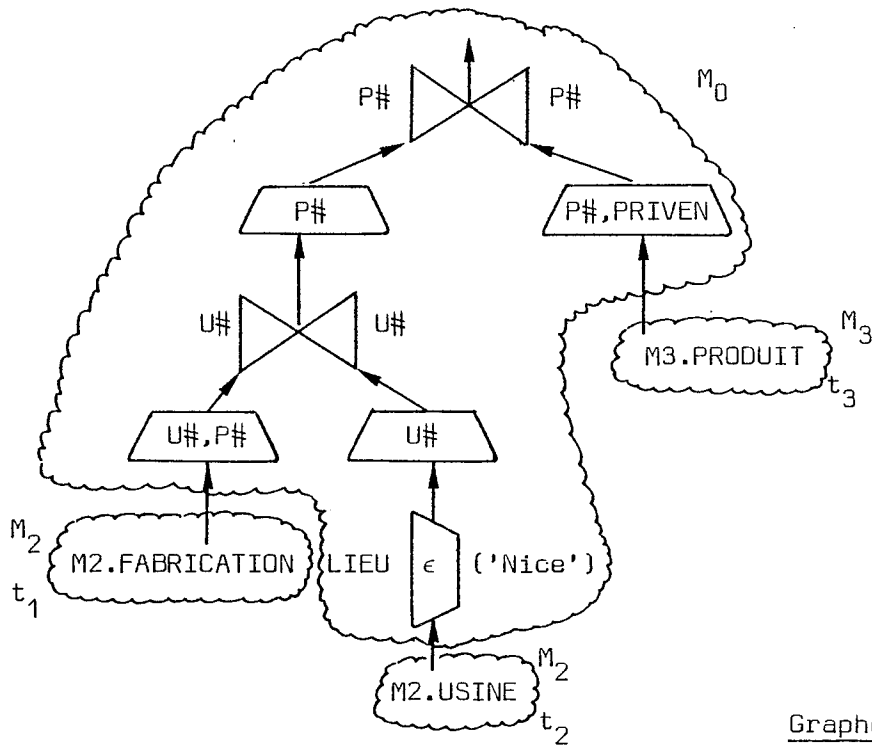
En conséquence, notre objectif est de minimiser les transferts inter-sites (en nombre et en volume) en mettant également en évidence la possibilité d'exécution en parallèle.

Le volume des données transférées n'est pas facile à connaître. A la mise en place des vues locales, l'administrateur peut indiquer la taille des relations locales, mais le volume des résultats intermédiaires dans une requête n'est pas facile à évaluer [B69].

Tout problème d'optimisation qui cherche à minimiser une fonction de coût (ou fonction économique) sur l'ensemble des solutions admissibles se retrouve dans la programmation mathématique. Malheureusement, dans le cadre où nous nous plaçons, un certain nombre de contraintes et de paramètres fondamentaux ne sont pas accessibles : c'est le cas pour tout ce qui concerne la communication et le routage des paquets d'information.

La solution choisie dans POLYPHEME n'est pas une solution optimale au sens mathématique mais plutôt une solution qui, en procédant dans le graphe localisé par regroupements successifs, assure toujours une réduction du coût de communication total [R17].

Nous allons l'illustrer sur l'exemple donné au paragraphe précédent qui donne le graphe localisé suivant (Graphe G5.1) :



Graphe G5.1

M_0 est la machine globale où la requête a été émise (notons que M_0 peut se trouver sur le même site que l'une des machines locales).

Ce graphe G5.1 correspond à une stratégie initiale de découpage S_0 caractérisée par un coût C_0 qui est égal à la somme des coûts de transfert entre machines (nous les assimilons uniquement aux volumes déplacés) :

C_0 = vol (M2.FABRICATION) + vol (M2.USINE) + vol (M3.PRODUIT)
où vol est une fonction monovaluée :

$$\text{RELATION} \xrightarrow{\text{vol}} \text{ENTIER}$$

vol(R) volume de R = Nb de n-uplet × taille d'un n-uplet.

En notant t_{ij} le transfert de données de la machine M_i vers la machine M_j, la stratégie de regroupement correspond aux quatre cas donnés par le tableau 5.1. On représente le coût associé à la stratégie S_k.

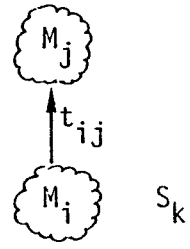
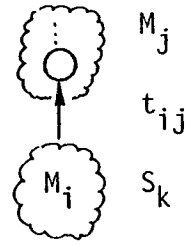
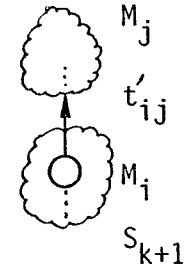
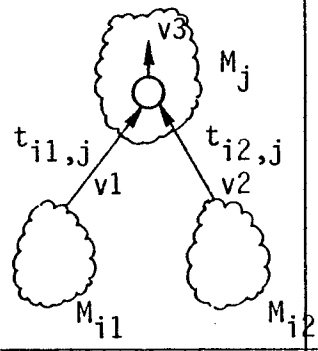
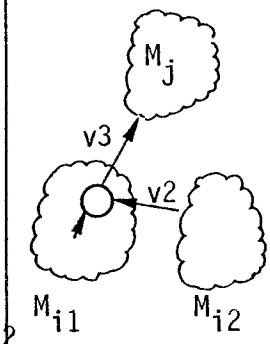
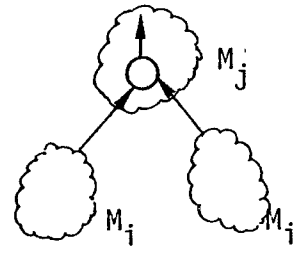
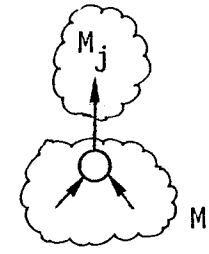
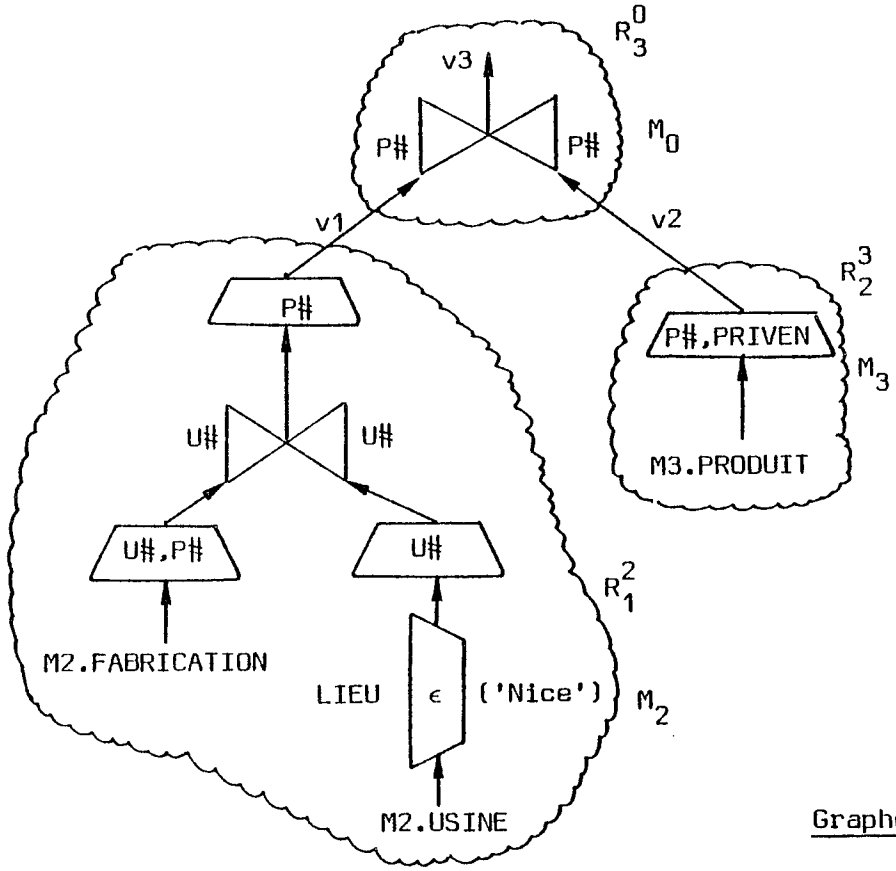
Cas	Condition Regroupement	Stratégie initiale S_k	Transformation S_{k+1}	Conséquence
Cas 1	i = j			$C_k > C_{k+1}$
Cas 2	t_{ij} consommé par une opération unaire (projection, restriction)		 On réduit la taille locale en premier	$C_k > C_{k+1}$
Cas 3	$t_{i1,j}$ et $t_{i2,j}$ sont consommés par une même opération binaire (v3 < v1) ou cas symétrique			$C_k > C_{k+1}$
Cas 4	idem cas 3 avec i1 = i2			$C_k > C_{k+1}$

Tableau 5.1 - Stratégies de Regroupements élémentaires

En appliquant ces regroupements au graphe G5.1, on obtient G5.2 :



Graphe G5.2

La Figure 5.10 montre le parallélisme des opérations et les transferts de données entre machines :

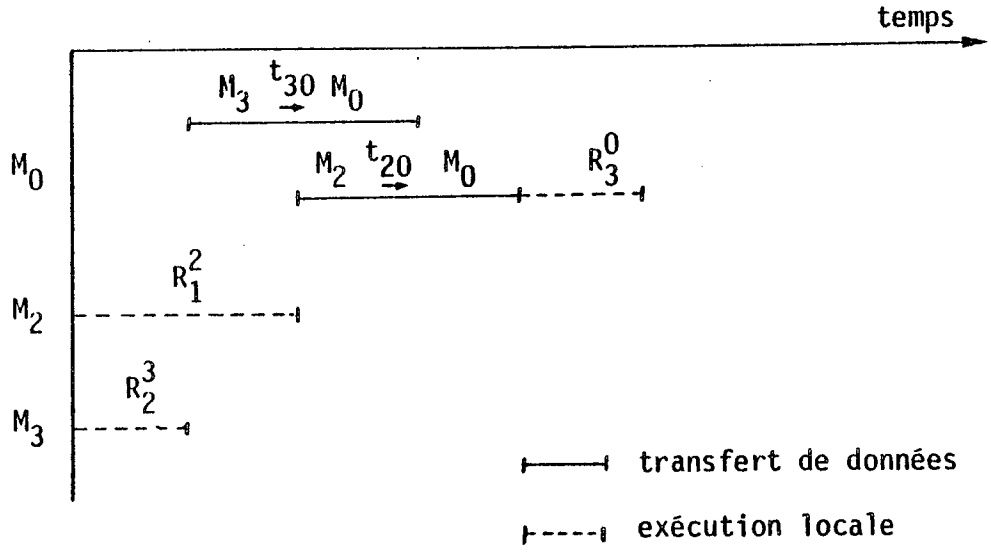


Figure 5.10 - Parallélisme entre machines

5.5.3.4.2. Préparation du plan d'exécution

Le processus de regroupement décrit ci-dessus suppose que toutes les machines soient homogènes ; or, dans l'approche que nous avons choisie concernant la réalisation des machines locales, celles-ci ne possèdent pas toutes les mêmes possibilités en ce qui concerne les opérateurs relationnels et ceci du fait de l'existence des programmes locaux.

Cependant, ces programmes locaux réalisent au moins la projection et la sélection (restriction), ce qui fait que les cas 1 et 2 restent valables. Dans le cadre du prototype POLYPHEME, les cas 3 et 4 ne sont pas applicables.

A partir du graphe ainsi regroupé, il faut mettre en évidence les appels aux programmes locaux et caractériser les informations sur la synchronisation des opérateurs, de manière à produire un plan d'exécution servant d'entrée à l'interpréteur global d'exécution (cf. § 5.5.4).

Un graphe représentatif d'un plan d'exécution se présente alors comme composé de [P13, P14] :

- noeuds qui représentent :
  - . soit des opérateurs
  - . soit des aiguillages (Figure 5.11)
    - * noeud émetteur représentant n-arcs sortants
    - * noeud collecteur représentant n-arcs rentrants
- d'arcs qui représentent le flot de données d'un noeud à un autre.

Figure 5.11 - Noeuds d'aiguillage

- A un émetteur est associé le lancement en parallèle de n opérateurs.
- A un collecteur est associé le rassemblement de m flots de données.

Ainsi en supposant l'existence :

- d'un programme d'accès séquentiel PASFAB sur M2 réalisant l'obtention et la projection de M2.FABRICATION sur (U#,P#)
- d'un programme d'accès associatif PAFUSI sur M2, réalisant l'obtention, la sélection sur LIEU, et la projection sur U# de M2.USINE
- d'un programme d'accès séquentiel PASPRO sur M3 réalisant l'obtention et la projection sur P#,PRIVEN de M3.PRODUIT, le graphe représentatif du plan d'exécution correspondant à la requête précédente est donné par G5.3.

En fait, ces noeuds de type émetteur et collecteur sont introduits dès la phase de codification évoquée au § 5.5.3.1, ce qui assure un codage uniforme de tous les graphes manipulés par la machine globale (graphe global, graphes localisés) [P13, P14].

C'est la raison pour laquelle nous employons un autre graphisme pour représenter les opérateurs : chacun est en quelque sorte mis dans une "boîte" représentative du module chargé de l'exécution (cf. § 5.5.4.2).

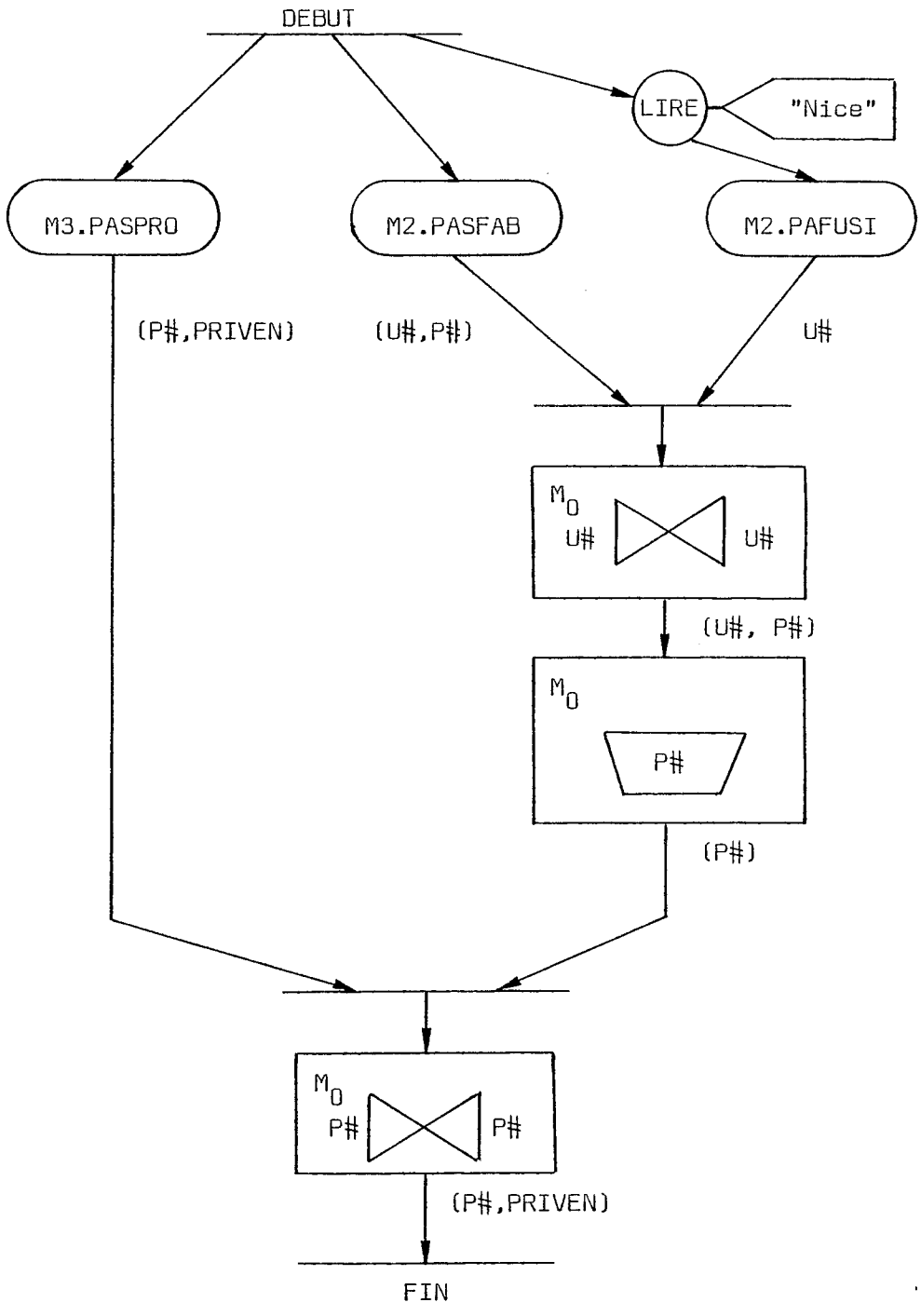
Chaque opérateur est exécuté par une machine (M_0 est la machine globale). L'opération "lire" est utilisée pour l'entrée du paramètre "Nice" (cf. § 5.5.4.3). Pour chaque arc, on indique la structure du tableau intermédiaire stocké dans les zones de travail.

5.5.4. Exécution Répartie

Cette partie de la machine globale est chargé - en s'appuyant sur les mécanismes fournis par le moniteur d'exécution répartie (§ 5.3) - d'interpréter le graphe localisé représentatif du plan d'exécution. Ceci dans le but d'exécuter et de synchroniser les opérations globales et locales pour élaborer finalement la réponse (cf. Figure 5.4).

Il s'agit là d'une partie très importante au niveau du SGBDR qui rejoint des problèmes de calcul parallèle et de synchronisation non triviaux [G4].

Dans le cadre de POLYPHEME, une première étude a été faite pour évaluer l'apport de travaux traitant de ces problèmes (en particulier les réseaux de Petri [G7, G8].



Graphe G5.3 - Un exemple de Plan d'exécution

A partir de là, la conception et la réalisation d'un interpréteur global d'exécution propre au projet POLYPHEME constitue la thèse développée par C. Euzet [R31].

Nous allons montrer comment cet interpréteur se place dans l'architecture du SGBDR en décrivant sa structure et les fonctions qu'il réalise.

5.5.4.1. Interpréteur Global d'Exécution

Son rôle est donc de parcourir le graphe localisé pour interpréter et contrôler les différentes opérations mises en oeuvre par chacun des noeuds.

A chaque opération est associé un nom de machine ; l'exécution d'une opération provoque sur cette machine la création d'un processus.

Les machines communiquent entre elles via le réseau, le protocole de communication et le moniteur d'exécution répartie.

Entre deux machines M_i et M_j qui échangent un flot de données t_{ij} (de i vers j), le moniteur d'exécution répartie crée, sur la demande de l'interpréteur, une "*interaction*" qui correspond à une association de type "producteur-consommateur" entre M_i et M_j [P11].

Une interaction se traduit, dans le système, par la création de deux files d'attente dans la copie du moniteur figurant sur chaque machine et par l'activation des processus de synchronisation associés (Figure 5.12).

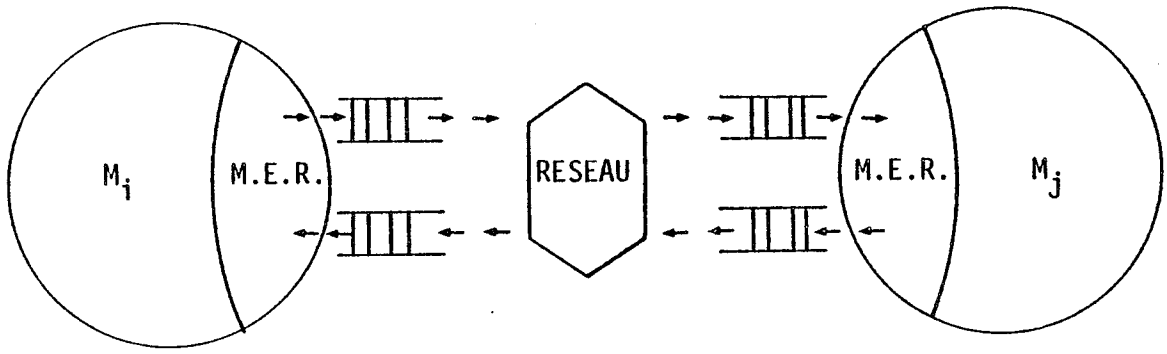


Figure 5.12 - Une Interaction entre deux machines

Pour un flot de données t_{ij} quelconque, on se trouve à un instant donné dans l'une des situations suivantes :

- M_i demande à M_j de lui envoyer quelque chose
- M_j envoie à M_i un ensemble d'informations (réponse) (sous-ensemble du flot t_{ij})
- M_i reçoit de M_j ce sous-ensemble
- M_j envoie à M_i un message signalant qu'il n'a rien à transmettre (fin)
- M_i reçoit de M_j ce message de fin.

Le schéma fonctionnel des modules de l'interpréteur global est donné par la Figure 5.13.
Il comprend trois parties :

- initialisations
- exécution
- contrôle.

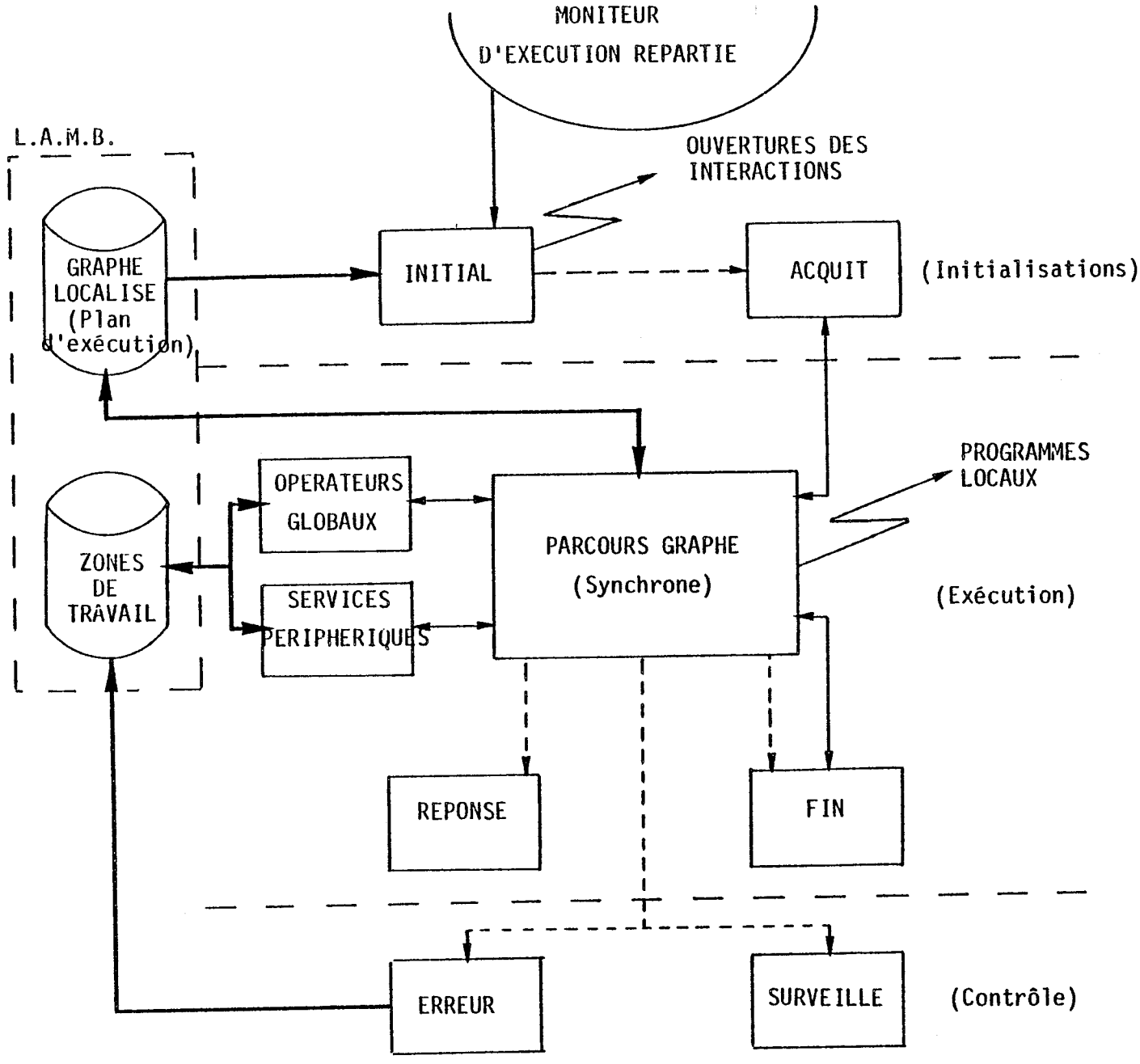
La partie initialisation a pour but de demander l'ouverture initiale des interactions telles qu'elles apparaissent dans le graphe localisé (transfert entre machines).

Cette ouverture fait appel à un service du moniteur d'exécution répartie qui provoque l'activation indirecte du module d'acquittement ACQUIT.

En effet, comme nous l'avons vu au § 5.3, la primitive essentielle d'appel aux services du moniteur a la forme générale suivante :

exécuter (S2, P2, REP, FIN, ERREUR, SURVEILLE)
└──────────────────┘
REPONSE

- où S2 est le site où lancer P2
- REP, FIN, ERREUR trois procédures réponses à activer par P2 respectivement dans le cas suivants :
  - REP dans le cas d'un envoi d'un sous-ensemble de données
  - FIN dans le cas où l'ensemble à transmettre l'a été intégralement
  - ERREUR dans le cas d'une erreur détectée par P2
- SURVEILLE est la procédure utilisateur à activer lorsqu'il se produit une panne réseau.



Légende :

——→ transfert de contrôle direct d'un module à l'autre

- - - → transfert de contrôle indirect via une procédure réponse

⇝ appel aux services du moniteur d'exécution répartie

——→ flot de données

Figure 5.13 - Schéma fonctionnel de l'Interpréteur Global d'Exécution

A chaque évènement est associé un modude de l'interpréteur :

- REPONSE reçoit une valeur qu'elle stocke dans une zone de travail où sont rangés tous les tableaux intermédiaires.
- FIN va lancer le module de parcours du graphe qui va interpréter les noeuds suivants et mettre à jour l'historique de l'interprétation. Il va également assurer la synchronisation en détectant les noeuds collecteurs.
- ERREUR est chargé de traiter les erreurs d'exécution

et - SURVEILLE les pannes réseaux.

5.5.4.2. Opérateurs Globaux

Cette partie correspond à toute une série de modules réalisant les opérateurs de l'algèbre relationnelle. Elle est réalisée dans le cadre d'un projet de DEA [P17] et utilise le stockage relationnel fourni par LAMB.

5.5.4.3. Services périphériques

Ce composant a pour fonction la gestion de périphériques d'entrée et de sortie associés à la machine globale [P12].

Une approche simple a été choisie : chaque machine a, par construction, un périphérique d'entrée et un de sortie qui sont des périphériques logiques pouvant être mis en correspondance avec des périphériques physiques (terminal, fichier, par exemple).

Sur le périphérique d'entrée est émise la transaction globale et sont disponibles les paramètres d'entrée de chaque transaction globale.

Sur le périphérique de sortie se trouvera la réponse à la transaction, les résultats intermédiaires étant stockés sous forme de tableaux LAMB.

5.6. CONCLUSIONS SUR L'ARCHITECTURE DU SGBDR

Nous venons de décrire l'architecture d'un SGBDR comme un réseau de machines relationnelles assurant deux fonctions principales :

- Gestion, Stockage d'un ensemble d'informations : les machines locales
- Accès, Manipulation d'un ensemble d'informations réparties : les machines globales.

Cette présentation a été faite sous l'angle de l'usager du SGBDR qui s'adresse à *une* machine globale, construite autour de plusieurs machines locales, chacune ayant la charge d'une portion de la base répartie.

Nous avions remarqué au début du chapitre qu'une machine globale pouvait se trouver sur le même site que zéro, une ou plusieurs machines locales. En généralisant cette architecture à un ensemble d'utilisateurs, exploitant plusieurs bases de données réparties différentes. On aboutit à l'architecture physique donnée par la Figure 5.14.

Sur chaque site du réseau dans lequel est implanté le SGBDR, cohabitent dans un seul calculateur :

- une machine globale
- une (ou plusieurs) machine(s) locale(s) gérant des données implantées dans ce calculateur
- les mécanismes propres à la communication et à l'exécution répartie.

Dans les choix de réalisation faits pour la maquette, le système ainsi construit est plutôt de type hiérarchique (cf. § 1.2), car les machines locales ne communiquent entre elles que par l'intermédiaire de la machine globale qui les "entoure".

Une étape future dans la réalisation du SGBDR serait d'envisager une structure de type horizontale, où les machines locales puissent communiquer directement entre elles. L'apport des travaux comme IGOR [R25] devrait faciliter une telle réalisation.

Au niveau de chaque composant du SGBDR, une série de mesures doit être faite à partir de la maquette. Le but est de déterminer les paramètres principaux affectant les performances. Une première étude partielle a été faite par simulation dans [P2]. Il faut généraliser ce genre d'études

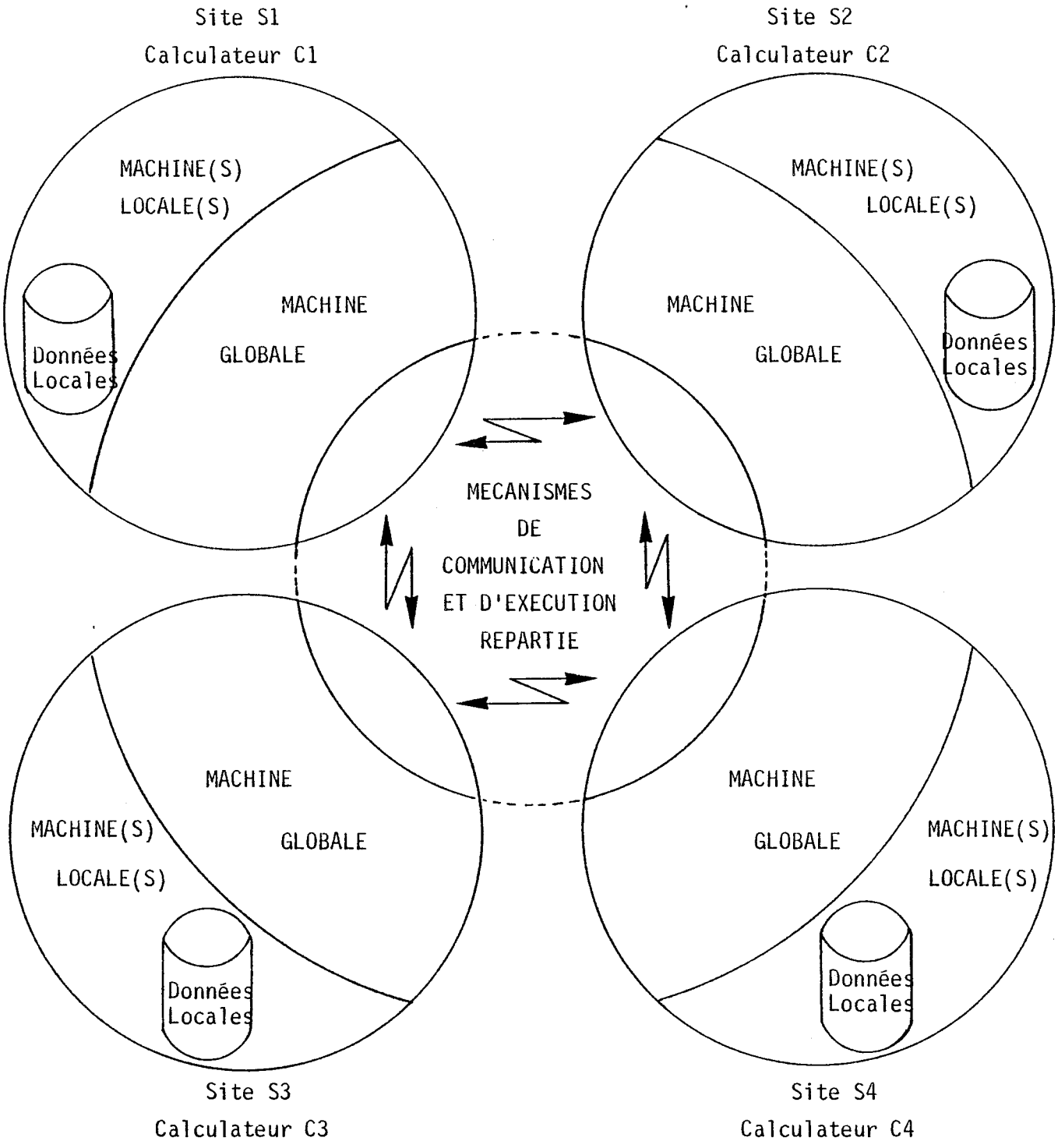


Figure 5.14 - Architecture Physique du SGBDR

de manière à aboutir aux spécifications d'un système véritablement opérationnel, la maquette ayant pour but principal de démontrer la validité de l'approche choisie. A ce propos, des extensions sont envisagées sur les deux plans suivants :

1) Sur le plan du logiciel d'accès et de manipulation de blocs (LAMB § 5.5.1) qui assure tout le stockage relationnel (catalogues, transactions sous forme interne, zones de travail). A lui seul, ce composant assure les fonctions d'un mini-système de base de données et ses performances doivent être à la hauteur d'une telle situation centrale.

2) Sur le plan des langages utilisés dans le système. Le langage LA.DO.RE, présenté en annexe , n'est qu'une première approche pour la description et la manipulation de données réparties. Au niveau de la description, il est spécifique de l'approche ascendante (coopération de bases existantes) et pourrait facilement être modifié pour l'adapter à une approche descendante. Il doit être d'autre part étendu pour exprimer des règles globales explicites, ce qui revêt les deux aspects suivants (cf. § 4.3.2.2) :

- expressions d'actions spontanées globales, associées aux opérations d'accès et de mises à jour, pour traduire les contraintes d'intégrité qui s'appliquent au niveau de la base répartie
- expressions de règles de décomposition explicites pour les cas non standard de répartition. Ces règles doivent alors exprimer le parallélisme des opérations locales, tel qu'il apparaît au niveau des graphes localisés.

Au niveau de la manipulation, il faut développer les possibilités d'entrée de données et de calcul. Pour cela, l'intégration de LA.DO.RE à un langage évolué de type PL/1 constitue un projet réalisable à court terme.

Au sein du projet POLYPHEME, nous n'avons pas abordé les problèmes de sécurité et de partage des données dans un environnement réparti, qui font l'objet, au sein du projet SIRIUS, de travaux spécifiques [R35, R39]. Nous avons concentré nos efforts sur les problèmes de description et de manipulation de données réparties dans un environnement hétérogène.

Nous avons ainsi caractérisé des cas standard et non standard de répartition. Dans le projet SDD-1 évoqué au § 1.3.1.2, seul le partitionnement par restriction est envisagé alors que le partitionnement par voisinage nous paraît très caractéristique des applications de type bases de données réparties.

Le problème de la décomposition des transactions globales en transactions locales a fait l'objet d'une attention particulière [R17], tandis que celui de l'exécution et de la synchronisation des transactions locales dans un environnement réparti fait l'objet d'une étude qui rejoint les problèmes du calcul parallèle [R31].

D'autres développements sont actuellement en cours pour assurer l'intégrité de la base répartie.

Enfin, le développement de logiciels réseaux permettant la mise en oeuvre d'applications réparties fournit le noyau de base sur lequel est construit le SGBDR [R21, R11].

6. Conclusions générales : Perspectives

On a le monde derrière soi
et devant soi. L'oeuvre
accomplie est oeuvre à faire,
car le temps de se retourner,
elle a changé.

G. DUHAMEL

Etant donné la nature du problème auquel nous nous sommes intéressé, et étant donné le domaine relativement nouveau qu'il aborde, nous avons essayé, au cours de notre travail, de concilier une approche pragmatique et une approche plus conceptuelle. Cette dernière a pour but de s'affranchir des contraintes matérielles, pour aller directement à l'essentiel, dégager puis clarifier les notions fondamentales propres aux bases de données hétérogènes réparties.

Un de nos soucis constant a été de nous appuyer sur des concepts très généraux comme ceux de la théorie des ensembles, sur laquelle sont bâtis de nombreux modèles de données, et de prendre comme point de départ des travaux ayant, dans le domaine des bases de données, une audience très large : en particulier les travaux de J.R. Abrial [B1,G1], ceux de C. Delobel [B29] et tous les travaux concernant le modèle relationnel de Codd [B17, B68].

Nous pensions en effet qu'avant de résoudre le problème de la coopération de bases hétérogènes réparties, il fallait évaluer l'apport de tels travaux pour décider de leur adéquation.

C'est ainsi que nous avons été amenés à choisir le modèle de Codd pour offrir aux usagers du SGBDR la possibilité de décrire et de manipuler des vues relationnelles mais également à définir un niveau conceptuel avec MOGADOR pour faciliter :

- d'une part la conception du SGBDR lui-même
- d'autre part la conception de vues relationnelles de données réparties dans un environnement hétérogène.

Certes, dans ce travail nous ne prétendons pas avoir résolu ni même abordé tous les problèmes spécifiques aux bases de données réparties, mais nous pensons avoir défini un cadre de travail dans lequel il est possible de mieux caractériser la nature des problèmes en vue d'arriver à des solutions réalisables à court ou moyen terme.

Nous pouvons d'ores et déjà dresser une liste non exhaustive des prolongements envisageables :

1 - Développement d'applications de type bases de données réparties

A partir des outils fournis par le SGBDR et par MOGADOR, il faut développer de nombreuses applications afin d'éprouver l'adéquation de ces outils à différentes situations concrètes. Deux applications sont du reste en cours au sein du développement de la maquette (§ 5.4.3) : l'une fournie par le CNET [P18] et l'autre constituant l'application commune aux participants du projet SIRIUS : AGORA [P19]. D'autres contacts sont pris avec des organismes français ou étrangers pour mettre en place une coopération dans ce domaine.

Un aspect fondamental de ce problème est la méthodologie à employer dans la conception de bases réparties, en partant d'un environnement logiciel et matériel existant. En effet, dans les années qui viennent l'informatique traditionnelle à base de systèmes centralisés va progressivement laisser la place à la "télématique" [G13]. De ce fait, de grosses entreprises ou des administrations auront à faire face au problème de la décentralisation des fonctions informatiques ainsi qu'au problème du partage et de la circulation des informations. Un gros travail reste à faire pour proposer une approche méthodologique du passage à l'informatique répartie. Les bases de données seraient alors construites par intégration successive de bases élémentaires (locales) dont le contenu et les fonctions auraient été bien définis au préalable. A partir de là, une coopération entre ces bases pourrait être mise en place, donnant ainsi naissance à une nouvelle base, qui, à son tour, pourrait être utilisée pour construire un niveau supérieur de coopération et ainsi de suite.

2 - Développement de l'architecture du SGBDR

A partir de l'architecture fonctionnelle sous formes de réseaux de machines relationnelles, il faut arriver à la réalisation d'un SGBDR homogène. Chaque machine posséderait des mécanismes de stockage et d'accès relationnels, dont l'élément de base serait inspiré du composant "LAMB" (Logiciel d'Accès et de Manipulation de Blocs - cf. § 5.5.1). D'autre part, le système serait plus à structure horizontale que hiérarchique : toutes les machines dialogueraient entre elles sans passer par un site centralisateur. L'évolution actuelle de la technologie n'interdit pas de penser que certains mécanismes de stockage et d'accès puissent être réalisés directement par du matériel spécialisé [B41,B70] améliorant ainsi considérablement les performances d'un tel système.

3 - Implantation d'outils d'aide à la conception

Une extension de l'approche que nous avons ici pour la réalisation du SGBDR est d'implanter effectivement des outils fournis par MOGADOR.

Cette implantation revient à construire un véritable système conversationnel destiné à l'Administrateur pour l'aider à concevoir une base de données répartie.

Il s'agit là d'un gros travail de réalisation puisqu'il faut prévoir dans le système deux niveaux :

- au niveau local, il faut fournir des mécanismes de transformations d'une structure de base de données (hiérarchique ou réseau) en une vue MOGADOR avec l'aspect dynamique que cela implique (cf. Chapitre 3)
- au niveau global, il faut aider au rapprochement de différentes vues pour les intégrer en une vue globale (cf. Chapitre 4).

A ces deux niveaux, l'intervention de l'administrateur est nécessaire pour expliciter les ambiguïtés sémantiques.

Sans remettre en cause le SGBDR tel qu'il a été défini (en particulier, en ce qui concerne le modèle de données et les langages de description et de manipulation), ce système d'aide à la conception apparaît comme complémentaire et fournit comme résultat des vues relationnelles.

Cependant, on pourrait envisager une version future du SGBDR qui fournirait un véritable niveau conceptuel au sens du rapport ANSI/SPARC [B7].
De ce fait, le système d'aide à la conception ferait partie intégrante du SGBDR.

Dans une telle optique, il faudrait développer des mécanismes de transformation dynamiques entre le schéma conceptuel et des schémas externes capables d'offrir aux usagers des vues de différentes natures (hiérarchiques, réseaux, relationnelles) et capables également d'adapter les transactions d'un niveau à l'autre (Figure 6.1).

Dans la mesure où, grâce à MOGADOR, le schéma conceptuel global serait de type relationnel binaire, il offirait ainsi une plus grande flexibilité pour la prise en compte des modifications dans la structure de la base répartie.

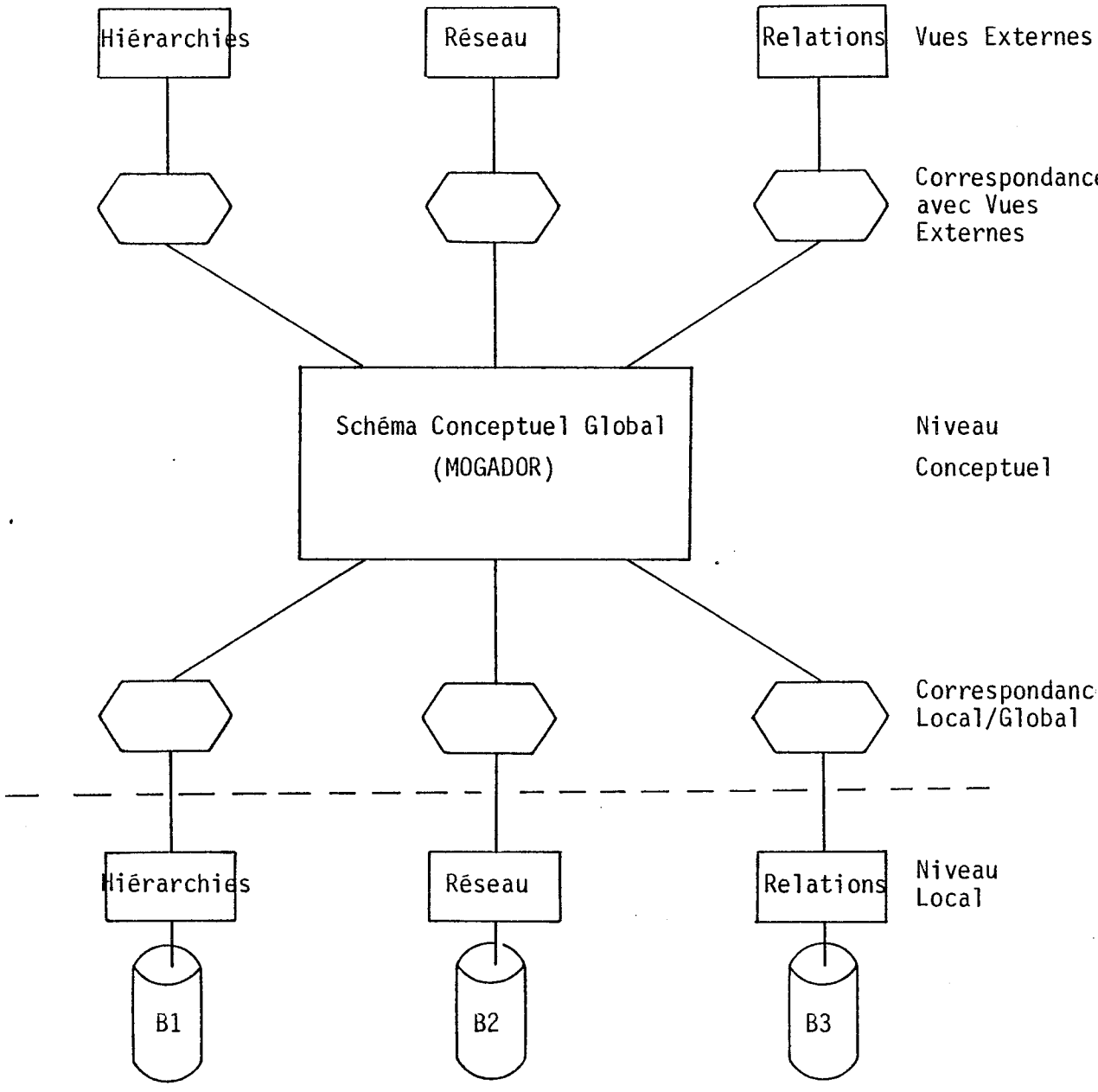


Figure 6.1 - Un SGBDR étendu

4 - Etude et développement de mécanismes d'autorisation

Dans le domaine des bases de données réparties, important est le problème de la confidentialité des données partagées [R58]. Ce problème a des prolongements sociologiques et politiques qui, sans être véritablement du ressort de l'informaticien, ne font pas moins partie de l'aspect déontologique de sa profession. Il importe de fournir des mécanismes

pour que l'outil que constitue le SGBDR ne se retourne pas contre l'individu.

5 - Modes d'utilisation d'un SGBDR

Il y a deux manières principales d'utiliser un système de base de données : le mode transactionnel, où l'usager ne demande au système que des fonctions précises pour lesquelles le système a été programmé (par exemple la réservation d'une place, la création d'une commande) et le mode interactif, où l'usager peut poser au système n'importe quel type de requête.

Le mode transactionnel correspond à une utilisation intensive du système, alors que le mode interactif correspond plutôt à une utilisation occasionnelle.

Dans le cadre des SGBDR, il faudrait cerner mieux ces deux modes d'utilisation et voir dans quel type d'applications ils peuvent être employés. Dans la mesure où des utilisateurs distants de partagent des informations, il est important qu'ils aient un langage commun. C'est pourquoi, dans une optique interactive, les travaux sur l'interrogation des bases de données en langue naturelle dont d'un grand intérêt et vont trouver une dimension nouvelle avec l'informatique répartie.

Il nous faut conclure en revenant au projet POLYPHEME. Ce projet est l'oeuvre de toute une équipe et les travaux présentés ici doivent beaucoup à chacun des membres de cette équipe. La réalisation effective de la maquette permet de matérialiser les efforts de chacun.

Certains informaticiens d'outre-atlantique ont distingué six phases dans l'évolution d'un projet :

1 - Enthousiasme délirant
2 - Désillusion
3 - Confusion totale
4 - Recherche du coupable
5 - Punition de l'innocent
6 - Promotion des non participants au projet.

En ce qui concerne POLYPHEME, nous sommes effectivement passés, non sans mal, par les trois premières phases. Cependant, la mise en chantier de la maquette et l'évolution encourageante de la réalisation nous ont permis de sortir de la confusion. Espérons que nous ne passerons pas par les trois dernières phases !

RÉFÉRENCES BIBLIOGRAPHIQUES

La connaissance conduit à l'unité
comme l'ignorance à la division.

RAMAKRISHNA

Les références sont classées en quatre groupes :

G. Documents généraux, ouvrages ;

B. Bases de données, modèles de données ;

R. Réseaux, systèmes répartis, bases de données réparties ;

P. Documents se rapportant plus directement au projet POLYPHEME (notes techniques en particulier).

G. OUVRAGES ET DOCUMENTS GÉNÉRAUX

[G1] J.R. ABRIAL
Manuel du langage Z (édition 1), Z/13 et Z/14
Paris, Mars 1977.

[G2] CROCUS
Systèmes d'exploitation des ordinateurs
Dunod, 1975.

[G3] C.J. DATE
An introduction to data base systems
(Second Edition), Addison Wesley Publishing Company, 1977.

[G4] Y. BEKKERS, J. BRIAT, J.P. VERJUS
Expression et décomposition du contrôle du parallélisme dans un système
Rapport de recherche n° 66, I.M.A.G., Janvier 1977.

[G5] S. KRAKOWIAK, M. LUCAS, J. MONTUELLE, J. MOSSIERE
A modular approach to the structured design of operating systems
Rapport de recherche n° 3, I.M.A.G., Mai 1975.

[G6] T. MUNTEAN
Formalisme pour la synchronisation des processus
Rapport de recherche n° 60, I.M.A.G., Septembre 1976.

[G7] J.L. PETERSON
Petri Nets
ACM Computing Surveys, vol 9, n° 3, Septembre 1977.

[G8] J. SIFAKIS
Structural properties of Petri nets
Rapport de recherche n° 102, I.M.A.G., Décembre 1977.

[G9] C.C. PINTER
Set theory
Addison Wesley Publishing Company, 1971.

[G10] D.C. TSICHRITZIS & F.H. LOCHOYSKY
Data base management systems
Academic Press, 1977.

[G11] N. WIRTH
Algorithms + data structures = programs
Prentice Hall, 1976.

[G12] **Informatique et Libertés**
(Rapport TRICOT), La Documentation Française, 1975.

[G13] S. NORA & A. MINC
L'informatisation de la société
La Documentation Française, 1978.

[G14] Groupe ALGOL de l'AFCET
Définition du langage algorithmique ALGOL 68
Hermann 1972.

[G15] P. DARONDEAU
Types et objet dans un système multi-langage
Thèse Doctorat d'Etat, USMG, Mars 1978.

B. BASES DE DONNÉES. MODÈLES DE DONNÉES

[B1] J.R. ABRIAL
Data Semantics
IFIP TC2 Working Conference, Cargèse, Avril 1974.

[B2] J.R. ABRIAL & al.,
Projet SOCRATE, Spécifications générales
Université de Grenoble, Septembre 1972.

[B3] M. ADIBA & C. DELOBEL
Les modèles relationnels de bases de données
Cours IRIA, Avril 1976.

[B4] M. ADIBA, C. DELOBEL, M. LEONARD
A unified approach for modelling data in logical data base design
IFIP TC2, Freudenstadt, Janvier 1976.

[B5] M. ADIBA
Transformation d'une structure SOCRATE en un ensemble de relations
Note technique I.M.A.G., Mars 1975.

[B6] M. ADIBA & M. LEONARD
TS1. Présentation d'une structure SOCRATE en termes de structure relationnelle
TS2. Présentation d'une structure CODASYL en termes relationnels
Notes Techniques I.M.A.G., Juin 1975.

[B7] ANSI (X3) SPARC
Study group on data base monoprocessor systems : interim report
Bulletin of ACM SIGMOD 7, n° 7, 1975.

[B8] M.M. ASTRAHAN & D.D. CHAMBERLIN
Implementation of a structured English query language
C. ACM, vol 18, n° 10, Octobre 1975.

[B9] M.M. ASTRAHAN & al.
System R. Relational approach to data base management
ACM TODS, vol 1, n° 2, Juin 1976.

[B10] P.G. BAZILLOU & G.E. BENCI
Présentation et analyse de SGBD commercialisés en France
IRIA S.T.I., Octobre 1975.

[B11] G. BRACCHI, A. FEDELINI, P. PAOLINI
A multilevel relational model for data base management systems.
Data Base Management Systems.
IFIP TC2, North Holland, 1974.

[B12] G. BRACCHI, G. PELAGATTI, P. PAOLINI
Models views and mappings in multilevel data base representation
Politechnico di Milano, 1976.

[B13] C.W. BACHMAN & M. DAYA
The role concept in data models
Advances in Data Base Technology, Infotech State of the Art Tutorial, Londres, Décembre 1977.

[B14] C.W. BACHMAN
Advances in data base technology
Londres 15-16 Décembre 1977, Infotech State of the Art Tutorial

[B15] R. BOYCE, D. CHAMBERLIN, F. KING, M. HAMMER
Specifying queries as relational expressions : the SQUARE data sublanguage
C. ACM, Novembre 1975.

[B16] B. BASTARAUD & M. DEVY
Transformations automatiques de structures
Projet de 3ème Année ENSIMAG, Juin 1976.

[B17] E. CODD
A relational model of data for large shared data banks
C.ACM, 13, 6, Juin 1970.

[B18] E.F. CODD
Understanding relations
FDT vol 7, n° 3-4, pp. 23-28, 1975.

[B19] E.F. CODD
Relational completeness of data base sublanguages
Courant Computer Science Symposia 6, "Data Base Systems", New York 71, Prentice Hall.

[B20] E.F. CODD
Further normalizations of the relational data base model
Courant Computer Science Symposia 6, New York 1971.

[B 21] E.F. CODD, R.S. ARNOLD, J.M. CADIOU, C.L. CHANG, N. ROSSOPOULOS
Rendez-Vous version 1 : an experimental English language query formulation system for casual users of relational data bases
RJ 2144, Janvier 1978, IBM Research Laboratory, San José, Cal.

[B22] P.Y. CHANG & J.M. SMITH
Optimizing the performance of a relational algebra data base interface
C. ACM, vol 18, n° 10, Octobre 1975.

[B23] D. CHAMBERLIN & al.
Sequel 2. A unified approach to data definition, manipulation and control
IBM Journal of Research and Development, vol 20, n° 6, Novembre 1976.

[B24] D. CHAMBERLIN & al.
Views authorization and locking in a relational data base system
Proceedings of the 1975 National Computer Conference, Anaheim Ca, Mai 1975.

[B25] D.D. CHAMBERLIN & R.F. BOYCE
Sequel : a structured English query language
Proceedings ACM-SIGMOD Workshop on Data Description, Access and Control, Ann Arbor, Mich., 1-3 Mai 1974.

[B26] CODASYL
Data Base Task Group Report, ACM New York, 1971.

[B27] **CODASYL, Introduction to feature analysis of generalized data base management systems**
C. ACM, vol 14, n° 5, Mai 1971.

[B 28] R. DEMOLOMBE
Syntex 2
Rappor final, CERT Toulouse, 1976.

[B29] C. DELOBEL
Contributions théoriques à la conception et l'évaluation d'un système d'informations appliqué à la gestion
Thèse de Doctorat d'Etat, Université Scientifique et Médicale de Grenoble, Octobre 1973.

[B30] C. DELOBEL
Sémantique des relations et processus de décomposition dans le modèle relationnel
Rapport de Recherche n° 45, Grenoble, Septembre 1976.

[B31] M. DEMUYNCK, P. MOULIN, S. VINSON
La portabilité des programmes utilisant un système de gestion de bases de données
Electricité De France, Service Informatique et Mathématiques Appliquées, Ref. HI.2431/04, Mai 1977.

[B32] K.P. ESMARAN, J.N. GRAY, R.A. LORIE, I.L. TRAIGER,
The notion of consistency and predicate locks in a data base system
C. ACM, vol 19/11, pp. 624-633, Novembre 1976.

[B33] R. FAGIN
Multivalued dependencies and a new normal form for relational data bases
ACM TODS, vol 2, n° 3, Septembre 1977.

[B34] J.P. FRY & E.H. SIBLEY
Evolution of data base management systems
ACM Computing Surveys, vol 8, n° 1, Mars 1976.

[B35] P.G. GRIFFITHS & B.W. WADE
An authorization mechanism for a relational data base system
ACM TODS, vol 1, n° 3, Septembre 1976.

[B36] J.P. GIRAUDIN
MACSI.P, méthode d'aide à la conception des systèmes d'information, orientée vers la fabrication de prototypes d'applications informatiques de gestion
Thèse de Doctorat de 3ème Cycle, Université de Grenoble, Juillet 1977.

[B37] L. KERSCHBERG, A. KLUG, D. TSICHRITZIS
A taxonomy of data models
Very Large Data Bases Conference, Bruxelles, Septembre 1976.

[B38] M. LACROIX & A. PIROTTE
Generalized joins
ACM SIGMOD Record, vol 8, n° 3, Septembre 1976.

[B39] M. LEONARD
Aides algorithmiques à la conception de bases de données
Thèse de Docteur Ingénieur, Université de Grenoble, Juin 1976.

[B40] R.A. LORRIE
XRM, an extended (n-ary) relational memory
IBM Scientific Centre report G320.2096, Cambridge, Mass., Janvier 1974.

[B41] C.S. LIN, D.L.P. SMITH, J.M. SMITH
The design of a rotating associative memory for relational data base applications
ACM Transactions on Database Systems, pp. 53-65, Mars 197

[B42] M. HAMMER & D.J. McLEOD
Semantic integrity in a relational data base system
First Very Large Data Bases Conference, 1975.

[B43] P.A.V. HALL
Optimization of simple expressions in a relational data base system
IBM Journal of Research and Development, Mai 1976.

[B44] NGUYEN GIA TOAN
URANUS : une approche relationnelle à la coopération de bases de données
Thèse de Doctorat de 3ème Cycle, USMG, Grenoble, Décembre 1977.

[B45] NGUYEN GIA TOAN
SOCRATE, interface PL/1
Note Technique CICG n° T37, Avril 1976.

[B46] G.M. NIJSSEN
A gross architecture for the next generation DBMS
Modelling in Database Management Systems, North Holland, 1976.

[B47] G.M. NIJSSEN
Current issues in conceptual schema concepts
Architecture and Models in Database Management Systems, IFIP TC2, Nice, Janvier 1977, North Holland.

[B48] I.M. OSMAN
The solution of some logical problems of defined relations
Working paper, Computer Centre, University of Khartoum, Soudan, 1977.

[B49] F.P. PALERMO
A data base search problem
Fourth International Symposium on Computer and Information Science, Miami Beach, Décembre 1972, Plenum Press (également IBM San Jose Research Report RJ1072).

[B50] P. PIN SHAN CHEN
The entity-relationship model. Toward a unified view of data
ACM TODS, Mars 1976.

[B51] A. PIROTTE
Comparaison de langages d'interrogation de bases de données relationnelles
Ecole d'Eté AFCET, Maroc, Juillet 1975.

[B52] M.E. SENKO
DIAM as a detailed exemple of the ANSI/SPARC architecture
IFIP TC2 Working Conference, Freudenstadt, Janvier 1976.

[B53] E.H. SIBLEY & R.W. TAYLOR
A data definition and mapping language
C.ACM, 16, 12, Décembre 1973.

[B54] J.M. SMITH & D.C.P. SMITH
Data base abstractions : aggregations and generalization
ACM TODS, vol 2, n° 2, Juin 1977.

[B55] N.C. SHU, B.C. HOUSEL, R.W. TAYLOR, S.P. GOSH, V.Y. LUM
EXPRESS : a data extraction, processing and restructuring system
ACM TODS, vol 2, n° 2, Juin 1977.

[B56] SOCRATE SIRIS 7/SIRIS 8
Manuel d'utilisation, Brochure CII, n° 4338 E/FR, Juillet 1973.

[B57] A. STIERS
Comparaison des systèmes SOCRATE disponibles au CICG
Note Technique T40, Octobre 1976.

[B58] M. STONEBRAKER & E. WONG
Access control in a data base management system by query modification
University of California, Berkeley, ERL M438, Mai 1974.

[B59] M. STONEBRAKER
Implementation of integrity constraints and views by query modification
Proceedings ACM SIGMOD Conference, San Jose, Mai 1975.

[B60] M. STONEBRAKER & al.
The design and implementation of INGRES
ACM TODS, vol 1, n° 3, Septembre 1976.

[B61] M. STONEBRAKER & G. HELD
Networks, hierarchies and relations in data base management systems
Memorandum ERL M504, Université de Californie, Berkeley, Mars 1975.

[B62] S. TODD
PRTV an efficient implementation of large relational data bases
Very Large Data Base Conference 1975.

[B63] S.J.P. TODD
Peterlee relational test vehicle, an overview
IBM Systems Journal, vol 15, n° 4, Novembre 1976.

[B64] S. TODD
Integrated architecture for transaction specification and optimization in relational data base systems
UKSC Scientific Centre Report n° 0085, Novembre 1976.

[B65] H. WEBER
The D-graph model of large shared data bases : a representation of integrity constraints and views on abstract data types
IBM Research Report, RJ 1875 (27024), 1976.

[B66] M. ZLOOF
Query by example
First Very Large Data Base Conference, 1975.

[B67] **Data base management systems**
ACM Computing Surveys : Special Issue, vol 8, n° 1, Mars 1976.

[B68] E.F. CODD
A list of references pertaining to relational data base management
IBM Research Laboratory, Août 1975.

[B69] R. DEMOLOMBE
A general semantic method for efficiently evaluating "AND" operators in relational DBMS
Internal report, LMB 77/5.

[B70] G. BERGER SABBATEL
Etude fonctionnelle d'un processeur de bases de données hiérarchi
Thèse de Doctorat de 3ème Cycle, Grenoble, Juin 1978.

[B71] REIN TURN & W.H. WARE
Privacy and security issues in information systems
IEEE Transactions on Computers, Novembre 1976.

[B72] T.H. MERRETT
The extended relational algebra, a basis for query languages
Technical report SOCS 78.2, Novembre 1977, McGill University, Montreal.

[B73] T.H. MERRETT
Aldat, augmenting the relational algebra for programmers
Technical report SOCS 78.1, Novembre 1977, McGill University, Montreal.

[B74] **MULTICS relational data store**
Reference manual. Juin 1976. Honeywell Information Systems.

R. RÉSEAUX. SYSTÈMES RÉPARTIS. BASES DE DONNÉES RÉPARTIES

[R1] M. ADIBA & C. DELOBEL
The cooperation problem between different data base management systems
IFIP TC2 Working Conference, Nice, Janvier 1977.

[R2] M. ADIBA
Présentation générale d'un système de coopération de bases de données réparties
Journées AFCET/IPP, "Bases de Données Réparties", Paris, 17-18 Mars 1977.

[R3] M. ADIBA
Projet POLYPHEME : MOGADOR un MOdèle Général de DOnnées Réparties
Rapport de Recherche n° 81, Grenoble, Juillet 1977.

[R4] M. ADIBA & D. PORTAL
A cooperation system for heterogeneous data base management systems
International Conference on Management of Data (ICMOD), Milan, 29-30 Juin 1978.

[R5] M. ADIBA
Modelling approach for distributed data bases
Congrès ECI 78, Venise, Octobre 1978.

[R6] M. ADIBA & J.Y. CALECA,
Modèle relationnel de données réparties. Problèmes de décomposition
Conférence Modèles Relationnels, Institut de Programmation, Paris, Mars 1978.

[R7] M. ADIBA, J.Y. CALECA, C. EUZET
A distributed data base system using logical relational machines
Very Large Data Base Conference, Berlin, Septembre 1978.

[R8] M. ADIBA, J.C. CHUPIN, R. DEMOLOMBE, G. GARDARIN, J. LE BIHAN
Issues in distributed data base management systems : a technical overview
"Papier invité", Very Large Data Base Conference, Berlin, Septembre 1978.

[R9] M. ADIBA, J.Y. CALECA, C. EUZET
Un SGBDR comme un réseau de machines relationnelles
A paraître, Congrès AFCET Informatique, Novembre 1978.

[R10] AUERBACH Publisher Inc.,
Operational and technological issues in distributed data bases
Data Base Management 1977.

[R11] E. ANDRE & P. DECITRE
On providing distributed applications programmers with control over synchronizations
Computer Network Protocols Symposium, Liège, Février 1978.

[R12] J.P. ANSART
Système interactif dans un environnement réseau. Connexion d'une machine à mémoire virtuelle IBM 360/67 au réseau CYCLADES
Thèse de Docteur Ingénieur, Université de Grenoble, Février 1976.

[R13] P.A. BERNSTEIN, N. GOODMAN, J.B. ROTHNIE, C.A. PAPADIMITRIOU
Analysis of serializability in SDD.1 : a system for distributed data bases
Computer Corporation of America, Technical Report, CCA.77.05.

[R14] L.O. BROOKS, A.F. CARDENAS, E. NAHOURAII
An approach to data communication between different GDBMS
Very Large Data Base Conference, Bruxelles, Septembre 1976.

[R15] P. BOSC, A. CHAUFFAUT, J. LE PALMEC, J.M. VILLARD
Projet Frères : interrogation de fichiers répartis sur un réseau de calculateurs hétérogènes
IRISA, Rennes, Revues Publication Interne n° 60, Janvier 1977.

[R16] J.Y. CALECA & J.M. FORESTIER
L'interrogation simultanée de plusieurs bases de données
Rapport de D.E.A., Université de Grenoble, Juin 1976.

[R17] J.Y. CALECA
Projet POLYPHEME : l'expression et la décomposition de transactions dans un système de bases de données réparties
Thèse de Doctorat de 3ème Cycle, USMG, Grenoble, Septembre 1978.

[R18] CANNING Publications
Distributed data systems
EDP Analyser, vol 14, n° 6, Juin 1976.

[R19] W.W. CHU & E. NAHOURAII
File directory design considerations for distributed data bases
Proceedings of the International Conference on Very Large Data Bases, Framington, Mass., Septembre 1975, pp. 543-545.

[R20] J.C. CHUPIN, H. RICHY, J. SEGUIN
Data sharing and cooperation between DBMS in heterogeneous computer networks
Proceedings AICA77, Pise, Octobre 1977.

[R21] J.C. CHUPIN
Répartition d'applications et de bases de données sur un réseau général d'ordinateurs
Thèse de Doctorat d'Etat, Université de Grenoble, Octobre 1977.

[R22] J. DOUCY & R. MORIN
SOCRATE/SOLAR réparti
Journées IRIA Bases de Données Réparties, Lans en Vercors, Mars 1976.

[R23] Ng.X. DANG & G. SERGEANT
Expression of parallelism and communication in distributed network processing
International Conference on Parallel Processing, Bellaire, Michigan, Août 1977.

[R24] Ng.X. DANG & G. SERGEANT
System and portable language intended for distributed and heterogeneous network applications
ENSIMAG, Grenoble, 1977.

[R25] Ng.X. DANG
Système et langage portable pour le traitement des applications réparties
Thèse de Doctorat de 3ème Cycle, USMG, Grenoble, Mars 1978.

[R26] R. DEMOLOMBE, H. GALLAIRE, M. LEMAITRE, J.M. NICOLAS
Système de gestion de bases de données réparties
Journées IRIA Bases de Données Réparties, Lans en Vercors, Mars 1976.

[R27] R. DEMOLOMBE & M. LEMAITRE
Rôles d'un modèle commun dans la conception d'un SGBD réparti : analyse des principaux modèles
CERT, Rapport de Recherche, Mars 1977.

[R28] M.E. DEPPE & J.P. FRY
Distributed data bases : a summary of research
Computer Network, 1, 1976.

[R29] C.A. ELLIS
Consistency and correctness of duplicate data base systems
ACM Symposium on Operating Systems, Novembre 1977.

[R30] C.A. ELLIS
A robust algorithm for updating duplicate data bases
1977 Berkeley Workshop on Distributed Data Management and Computer Networks, University of California, Berkeley, pp. 146-158, Mai 1977.

[R31] C. EUZET
Projet POLYPHEME : l'interprétation, l'exécution et la synchronisation des transactions dans un SGBDR
Thèse de Doctorat de 3ème Cycle, à paraître, Grenoble, Octobre 1978.

[R32] R. EPSTEIN, M. STONEBRAKER, E. WONG
Distributed query processing in a relational data base system
Memorandum n° UCB/ERL M78/18, Université de Berkeley, Avril 1978.

[R33] G. GARDARIN, R. GOMEZ, M. JOUVE, C. PARENT, S. SPACCAPIETRA
Architecture des systèmes de bases de données réparties
Journées IRIA Bases de Données Réparties, Lans en Vercors, Mars 1976.

[R34] G. GARDARIN, M. JOUVE, C. PARENT, S. SPACCAPIETRA
Designing a distributed data base management system
Proceedings AICA'77, Pise, Octobre 1977.

[R35] G. GARDARIN
Résolution des conflits d'accès simultanés à un ensemble d'informations. Applications aux bases de données réparties
Thèse de Doctorat d'Etat, Université Paris VI, Avril 1978.

[R36] G. GARDARIN & P. LEBEUX
Scheduling algorithms for avoiding inconsistency in large data bases
Proceedings of the Third International Conference on Very Large Data Bases, Tokyo, Octobre 1977, édité par IEEE, pp. 500-507.

[R37] J.N. GRAY & V. WATSON
A shared segment and interprocess communication facility
IBM Report RJ1579, Mai 1975.

[R38] N. GOODMAN & J.B. ROTHNIE
A survey of research and development in distributed data base management
International Conference on Very Large Data Base, Tokyo, pp. 48-6
Octobre 1977.

[R39] M. JOUVE
Reliability aspects in a distributed data base management system
Proceedings AICA'77, Pise, pp. 199-209, Octobre 1977.

[R40] ISO/T697/SC16
Provisional model of open-systems architecture
Working paper.

[R41] B. LAMPSON & H. STURGIS
Crash recovery in a distributed data storage system
Internal Report, Computer Science Laboratory, XEROX, Palo Alto Research Center, 1976.

[R42] J. LE BIHAN
La répartition des données dans les réseaux informatiques
Congrès AFCET, Novembre 1976.

[R43] S.M. MIRANDA
Data security in centralized and distributed data bases
Rapport IRIA 1978.

[R44] E. NEUHOLD & M. STONEBRAKER
A distributed data base version of INGRES
Memorandum ERL/M612, Université de Berkeley, Septembre 1976.

[R45] E.J. NEUHOLD & H. BILLER
Distributed data bases on a network of mini-computers
AFCET Workshop on Distributed Data Bases, Institut de Programmation, Paris, pp. 113-138, Mars 1977.

[R46] E.J. NEUHOLD & H. BILLER
POREL : a distributed data base on an inhomogeneous computer network
Conference on Very Large Data Bases, Tokyo, Octobre 1977.

[R47] D. MEYER & M. DESFEUILLET
Comment passer d'une base locale à une base répartie ? Exemple de la gestion comptable et financière du CNRS
AFCET, Journées BDR, Institut de Programmation, Paris, Mars 1977.

[R48] D. MEYER
Transformation d'une base de données centralisée en une base de données répartie sur un réseau d'ordinateurs
Thèse de Docteur Ingénieur, Université de Nancy, Novembre 1977.

[R49] L. POUZIN
Le réseau CYCLADES
Techniques Françaises, IEEE, Avril 1974.

[R50] POLYPHEME
Propositions pour un modèle de répartition et de coopération de bases de données dans un réseau d'ordinateurs
Laboratoires informatiques CII/ENSIMAG/USMG, Grenoble, Rapport de Recherche n° 29, Décembre 1975.

[R51] I. PALMER
Distributed data bases
Infotech State of the Art Report, Novembre 1976.

[R52] P. PAOLINI, G. PELAGATTI, F.A. SCHREIBER
An application oriented approach to distributed data bases
AFCET Workshop on Distributed Data Bases, Institut de Programmation, Paris, Mars 1977, pp. 139-151.

[R53] R. PEEBLES & E. MANNING
A computer architecture for large (distributed) data bases
Conference on Very Large Data Bases, 1976.

[R54] D. PORTAL & H. RICHY
Un concentrateur multi-connexions dit "concentrateur intelligent"
Journées IRIA Bases de Données Réparties, Lans en Vercors, Mars 1976.

[R55] G. PRAKASH HEBALKAR, CHIN TUNG
Design considerations for distributed data base systems
IBM Research Laboratory, San Jose, 1977.

[R56] J.B. ROTHNIE & N. GOODMAN
A study of updating in a redundant distributed data base environment
Technical Report CCA.77.01, Cambridge, Février 1977.

[R57] J.B. ROTHNIE & N. GOODMAN
An overview of the preliminary design of SDD.1, a system for distributed data bases
Proceedings of the Second Berkeley Workshop on Distributed Data Management and Computer Networks, Berkeley, Mai 1977.

[R58] H. RICHY
Confidentialité, bases de données et réseaux d'ordinateurs
Thèse de Docteur Ingénieur, INP Grenoble, Février 1978.

[R59] F.A. SCHREIBER
Problems and models in distributed data base systems
Rapport Interne n° 75.14, Istituto di Elettrotecnica ed Elettronica, Politecnico di Milano, 1975.

[R60] S. SAVOYSKI
Problèmes de répartition des données géo-techniques dans les laboratoires des Ponts et Chaussées
AFCET Journées BDR, Institut de Programmation, Paris, Mars 1977.

[R61] J. SEGUIN, G. SERGEANT, P. WILMS
Cohérence et gestion d'objets dupliqués dans les systèmes distribués
Rapport Interne, ENSIMAG, Grenoble, Mai 1977.

[R62] J. SEGUIN
Traitement distribué d'informations réparties dans les réseaux d'ordinateurs
Thèse de Doctorat d'Etat, USMG, Grenoble, Mars 1978.

[R63] G. SERGEANT
SOCYCRATE. Spécifications et manuel d'utilisation. Serveur SOCRATE sur CYCLADES
ENSIMAG, Grenoble, Mai 1975.

[R64] E. WONG & K. YOUSSEFI
Decomposition strategy for query processing
ACM Transactions on data base systems, vol 1, n° 3, pp. 223-241, Septembre 1976.

[R65] E. WONG
Retrieving dispersed data from SDD.1 : a system for distributed data bases
1977 Berkeley Workshop.

[R66] Rapport spécial sur les systèmes répartis
Informatique Nouvelle, Décembre 1976.

[R67] Distribuer l'informatique à travers l'architecture unifiée de réseau
IBM Magazine, n° 1, 1977.

[R68] S.R. KIMBLETON & G.M. SCHNEIDER
Computer communication networks : approaches, objectives and performance considerations
ACM Computing Surveys, vol 7, n° 3, Septembre 1975.

[R69] L. POUZIN
Les réseaux informatiques
Séminaire IMAG, Mars 1977.

[R70] H. BREITWIESER, U. KERSTEN, O. DROBNICK
DISCO : a distributed file management system for heterogeneous computer networks
International Conference on Data Base Management Systems, 29-30 Juin 1978, Milan.

P. RÉFÉRENCES "POLYPHEME"

[P1] J.Y. CALECA & A. STIERS
Interface relationnel POLYPHEME - SGBD - SOCRATE
NT1, POLYPHEME, Janvier 1977.

[P2] J.M. ANDRADE
Simulation du comportement de différentes bases de données réparties sur un réseau d'ordinateurs
Projet de DEA, USMG, Grenoble, Juin 1977.

[P3] E. ANDRE & P. DECITRE
Spécifications ZO du modèle POLYPHEME MOGADOR
NT8, POLYPHEME, Août 1977.

[P4] P. DECITRE
Accès depuis PL/1 à la station de transport ST2 sous SIRIS 8
NT9, POLYPHEME, Août 1977.

[P5] M. ADIBA & J.Y. CALECA
Décomposition fonctionnelle de la machine noyau
NT10, POLYPHEME, Octobre 1977.

[P6] D. D'AGARO & J. HAMEON
Bases de données documentaires réparties. Proposition pour un modèle d'accès aux données dans un réseau d'ordinateurs
Rapport de DEA, Grenoble, Septembre 1977.

[P7] E. ANDRE & P. DECITRE
Point de vue utilisateur sur la synchronisation dans les applications réparties
NT12, POLYPHEME, Octobre 1977.

[P8] A. STIERS
Programmation d'une application répartie en PL/1 asynchrone
NT13, POLYPHEME, Octobre 1977.

[P9] J.M. ANDRADE, NGUYEN GIA TOAN, A. STIERS
Spécifications d'un logiciel d'accès et de manipulation de blocs d'informations (LAMB), volume 1
NT15, POLYPHEME, Décembre 1977.

[P10] M. ADIBA & J.Y. CALECA
Un exemple complet de bases de données réparties en MOGADOR
NT16, POLYPHEME, Décembre 1977.

[P11] E. ANDRE, P. DECITRE, H. GALY
Spécifications Z d'un moniteur d'exécution répartie (MER), Version 1 et Version 2
NT17 et NT21, POLYPHEME, Janvier 1978.

[P12] E. PICHAT & A. STIERS
Spécifications générales de la maquette POLYPHEME
NT18, POLYPHEME, Séminaire Prelenfrey, Janvier 1978.

[P13] C. EUZET
Propositions de codification des graphes de décomposition
NT24, POLYPHEME, Avril 1978.

[P14] M. ADIBA & J.Y. CALECA
Langage de description et de manipulation de vues relationnelles. Structure interne de représentation des catalogues et requêtes
NT25, POLYPHEME, Juin 1978.

[P15] J.M. ANDRADE
LAMB : spécifications générales de réalisation
NT26, POLYPHEME, Juin 1978.

[P16] E. ANDRE
"PERE" : protocole d'exécution répartie
NT28, POLYPHEME, Juillet 1978.

[P17] PAIK IN SUP
Définition et réalisation d'un ensemble d'opérateurs relationnels pour une machine POLYPHEME
Rapport de DEA, Septembre 1978.

[P18] J. CLOAREC
Présentation de l'application "catalogue et inventaire des équipements utilisés dans le système E10
CNET, Fiche Technique FT/RCI/EGN/6, Lannion, Novembre 1977.

[P19] E. ANDRE G. BOBO, H. GALY
Spécifications de définition du système AGORA
NT29, POLYPHEME, Juillet 1978.

ANNEXE

LANGAGE POUR

DONNÉES

REPARTIES

Cette annexe comprend :

1. *La syntaxe du langage de description des vues relationnelles locales et globales.*

2. *La structure des catalogues constituant la description des vues locales et globales.*

3. *La syntaxe du langage de manipulation de la vue globale.*

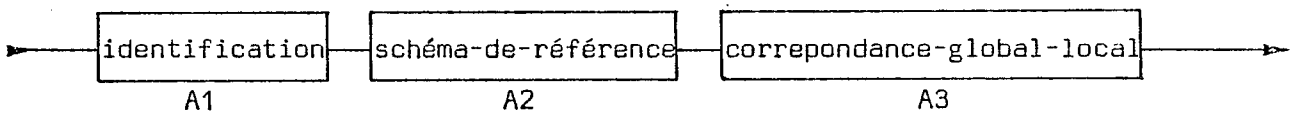
1. SYNTAXE DU LANGAGE DE DEFINITION

Le langage de définition permet de décrire des vues relationnelles qu'elles soient locales ou globales. Cette description comprend trois parties :

- Identification
- Schéma de référence de la vue
- Correspondance noms-objets.

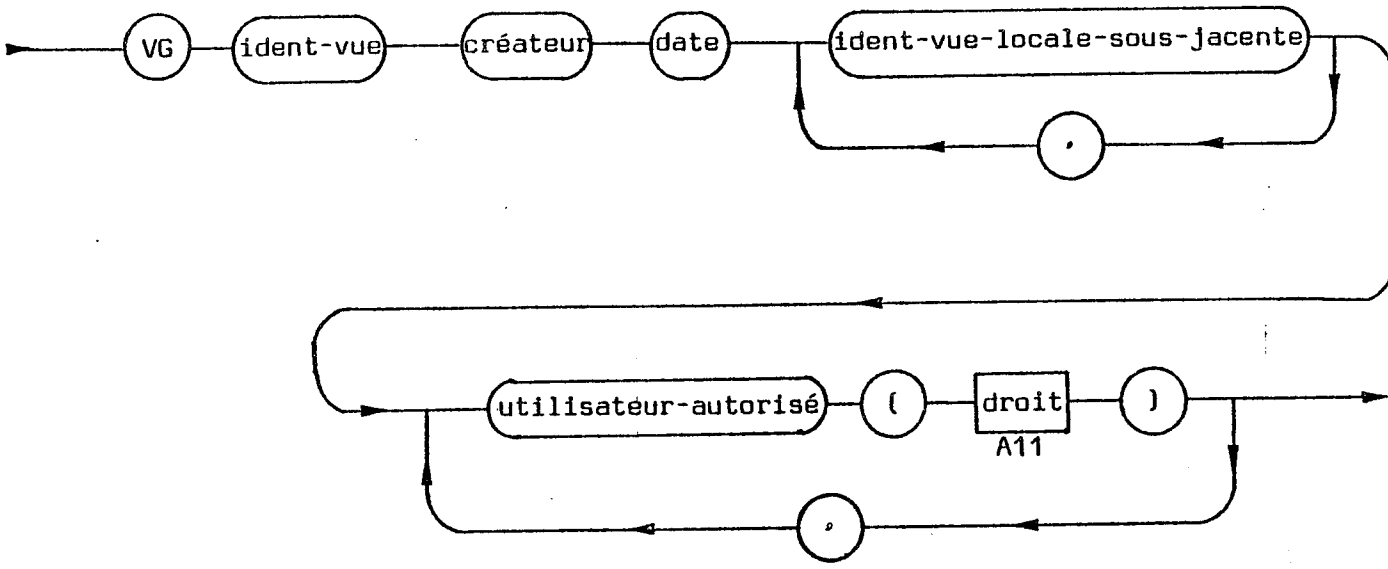
Nous donnons la carte syntaxique du langage, sous forme de diagrammes. Les constructions syntaxiques sont dénotées par des mots composés français, encadrés dans un rectangle. Ces mots décrivent la nature de la construction. Les symboles terminaux (mots-clés en majuscules ou construction terminale), sont entourés.

A : Description-d'une-vue-relationnelle-globale

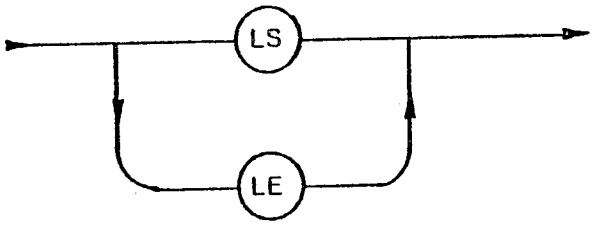


A1 : Identification

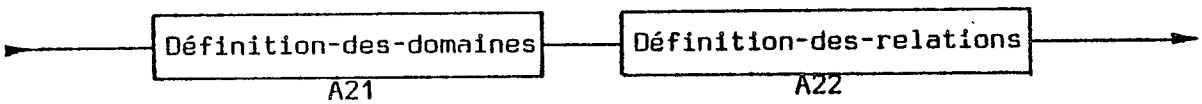
Elle décrit la vue globale,avec son nom,celui de l'administrateur et la date à laquelle elle a été créée.On y trouve également les vües locales partir desquelles est construite la vue globale, ainsi que la liste des utilisateurs globaux avec leurs droits(Lecture seule ou lecture/écriture).



A11 : Droit

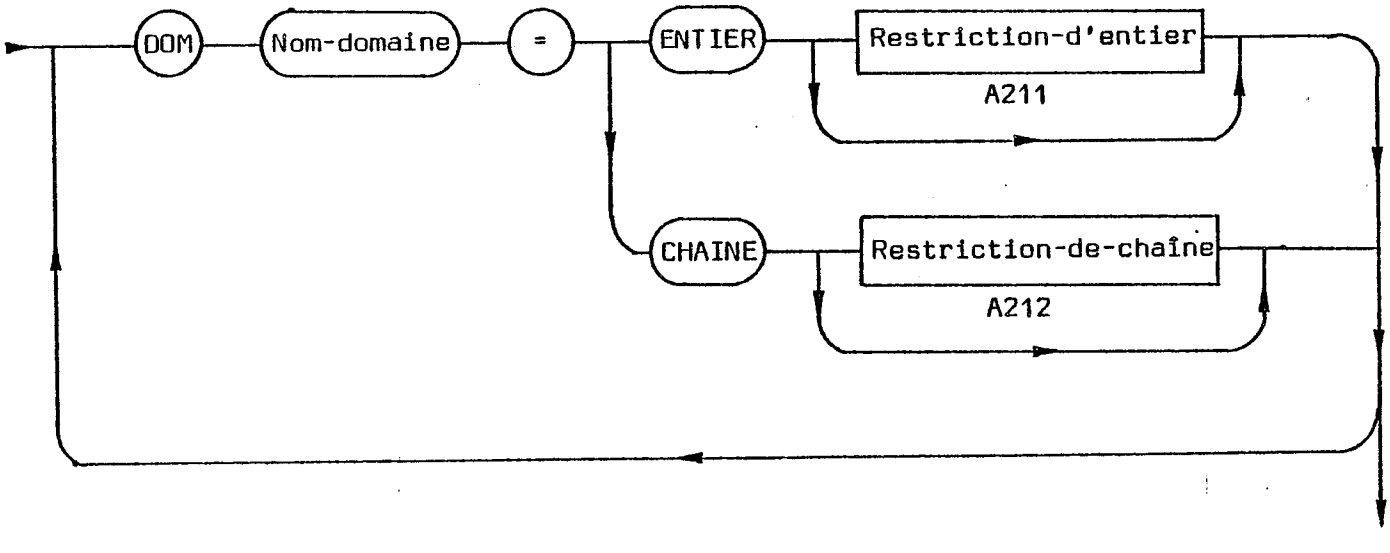


A2 : Schéma-de-référence



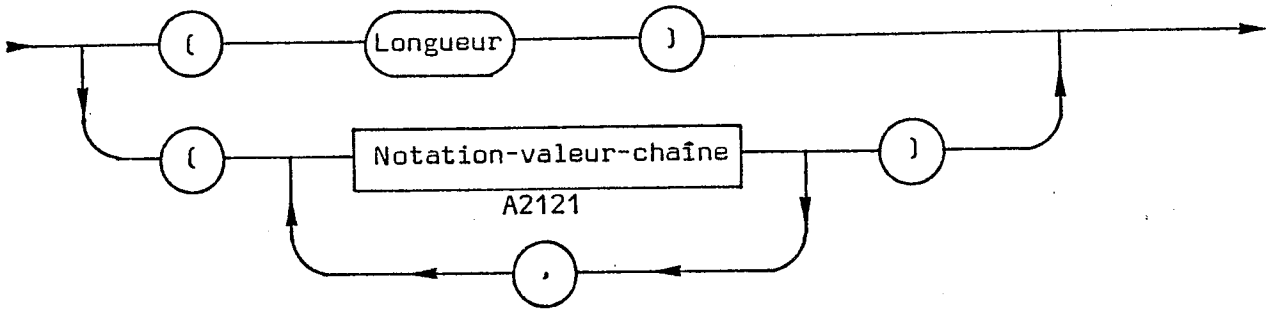
A21 : Définition-des-domaines

Elle liste les domaines avec le type de valeur associé(Entier,Chaîne) et une éventuelle restriction.

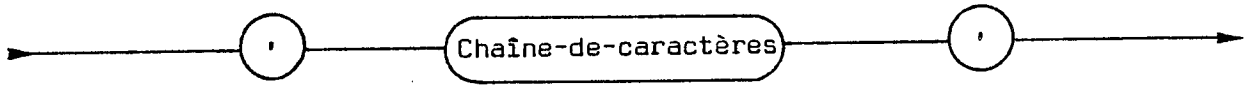


A211 : Restriction-d'entier

A212 : Restriction-de-chaîne



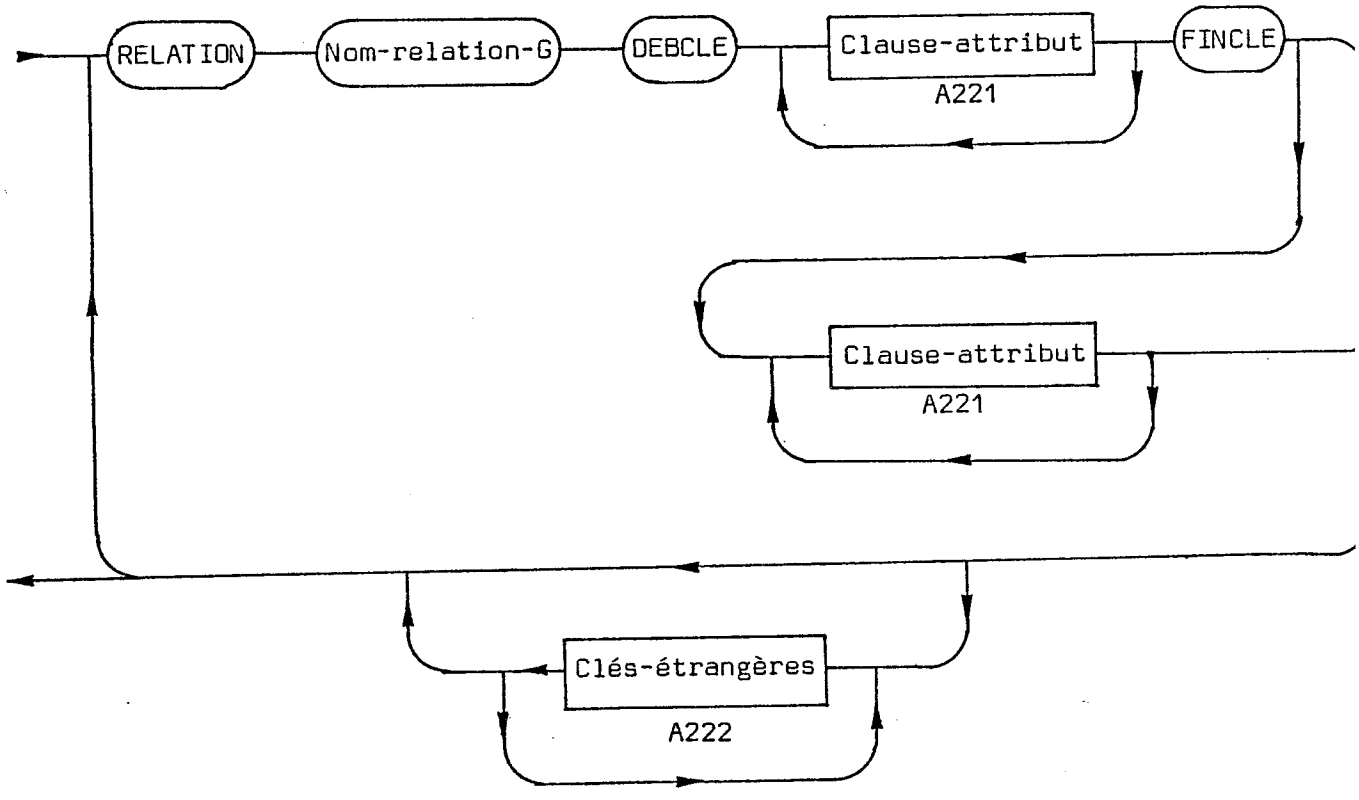
A2121 : Notation-valeur-chaîne



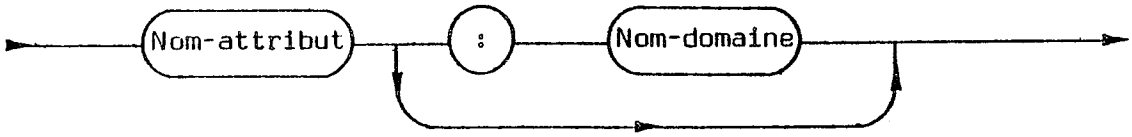
A22 : Définition-des-relations

Pour chaque relation globale, il faut indiquer :

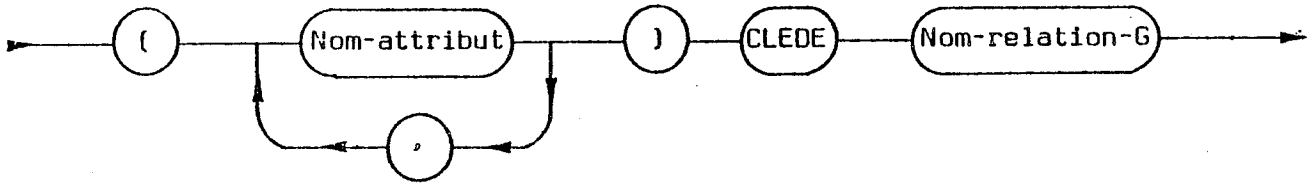
- son nom
- les attributs composant la clé de la relation
- les attributs non-clés
- les attributs formant une clé pour une autre relation(liaisons entre relations)



A221 : Clause-attribut



A222 : Clés-étrangères

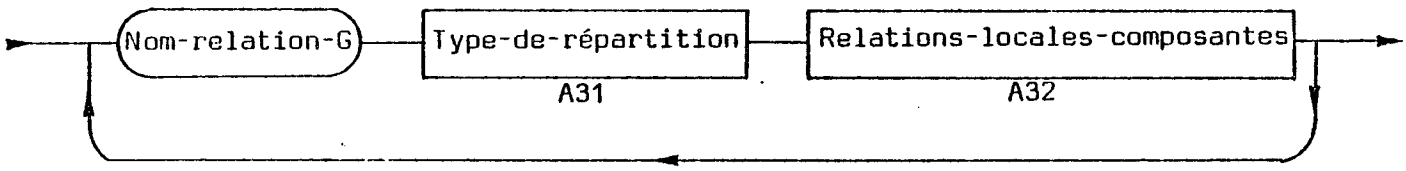


A3 : Correspondance-Global-Local

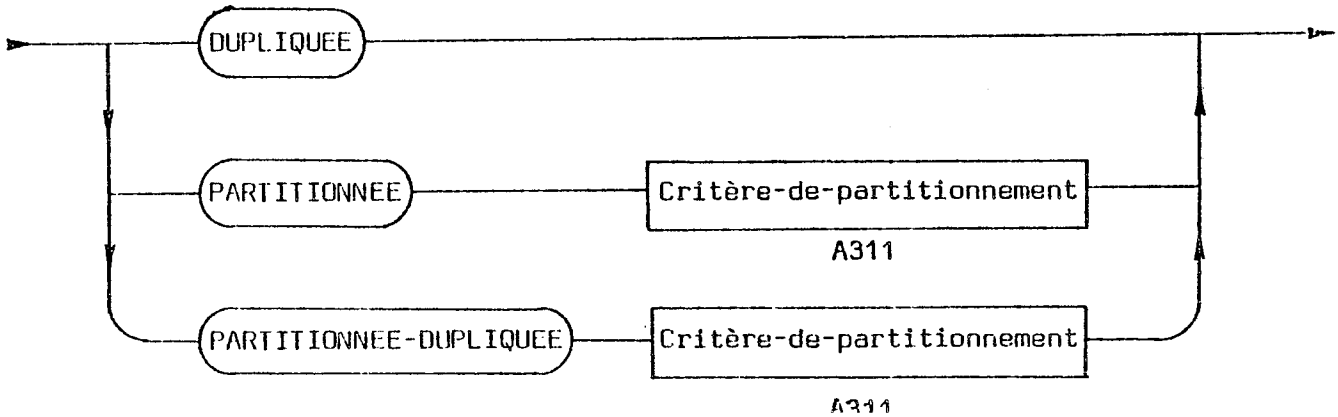
Nous avons fait à ce niveau un choix concernant les cas de répartition que nous voulions traiter.Nous ne considérons que les cas standard à savoir les six cas suivants :

- partitionnement par restriction(PAR) ou par voisinage(PAV)
- partitionnement avec duplication par restriction(PDR) ou par voisinage(PDV)
- Duplication simple(DUS) ou multiple(DUM)

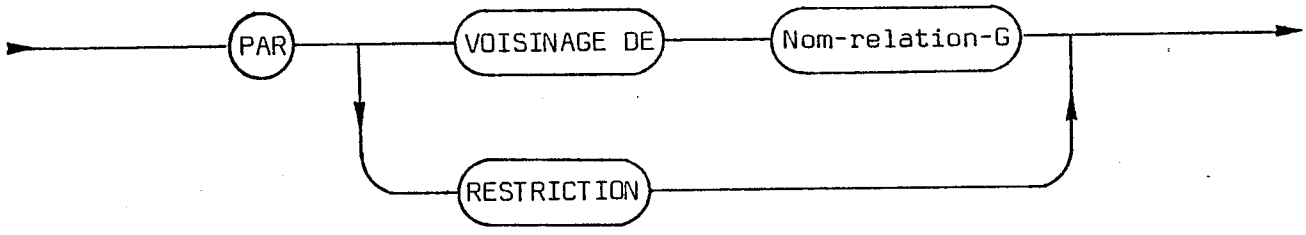
Comme nous l'avons montré au § 4.3.2. ces cas de répartition induisent des règles standard concernant les opérations d'accés et de modifications.



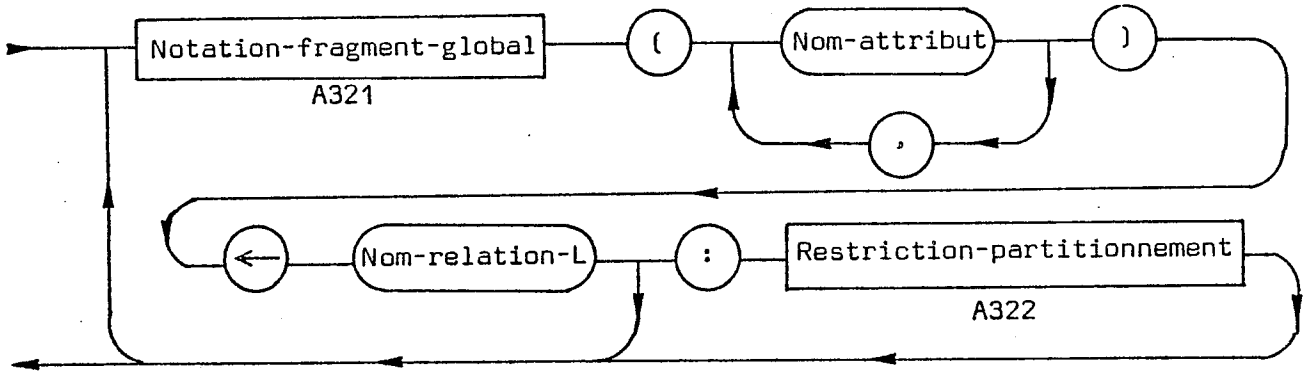
A31 : Type-de-répartition



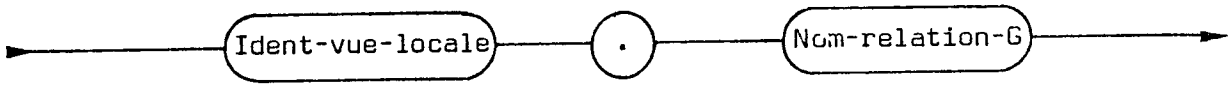
A311 : Critère-de-partitionnement



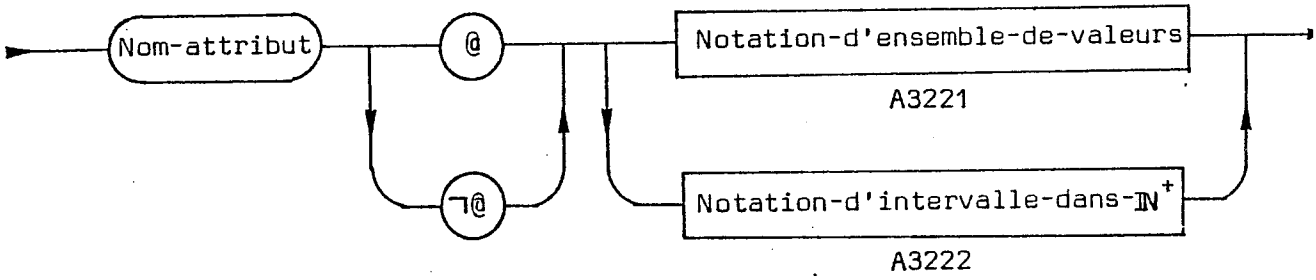
A32 : Relations-locales-composantes



A321 : Notation-fragment-global



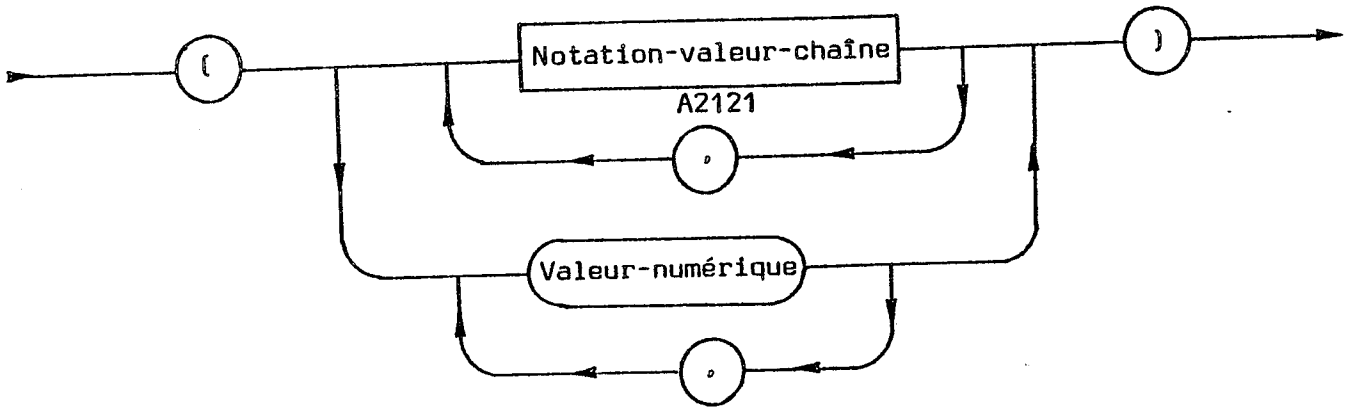
A322 : Restriction-partitionnement



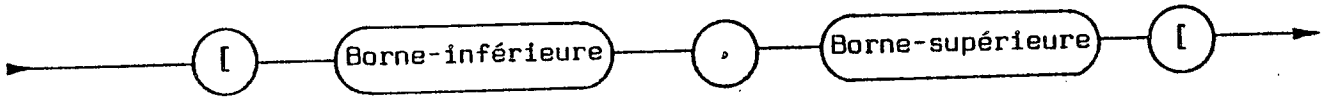
N.B. " @ " dénote l'opérateur ensembliste d'appartenance ("ϵ")

La restriction définissant le critère de partitionnement se limite à ces deux types de condition élémentaire :

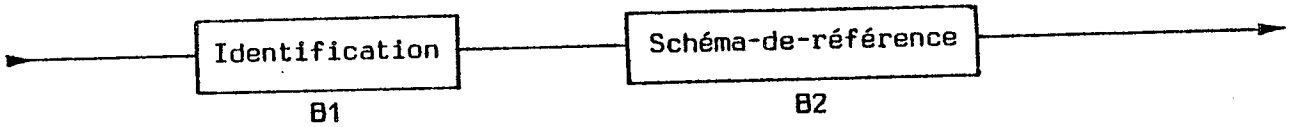
A3221 : Notation-d'ensemble-de-valeurs



A3222 : Notation-d'intervalle-dans-$\mathbb{N}^+$

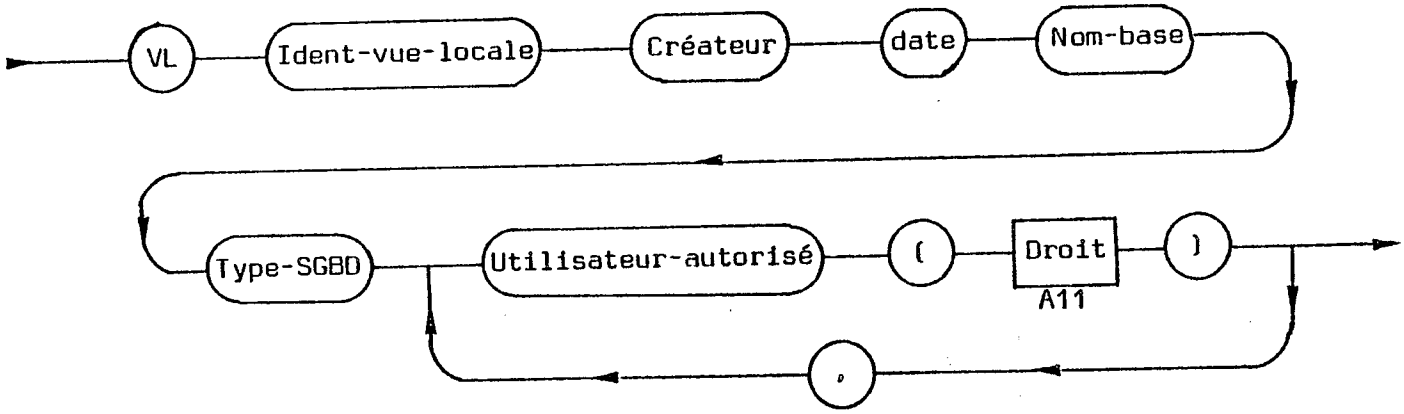


B : Description-d'une-vue-relationnelle-locale



N.B. La correspondance noms-objets se fait par l'intermédiaire des programmes locaux.Nous supposons qu'ils sont prédéfinis par le système (en tant que noms de programmes et fonctions qu'ils réalisent) et qu'ils ont été écrits et précompilés dans la base (§ 3.3).

B1 : Identification



B2 : Schéma-de-référence

Même construction syntaxique que A2. La seule différence est que les noms des domaines,des relations, des attributs sont locaux.

2. STRUCTURE INTERNE DES CATALOGUES

Les vues locales et globale constituent un ensemble d'informations qui est perçu également de manière relationnelle. Cela veut dire qu'une vue est en fait un ensemble de tableaux gérés grâce au logiciel LAMB (§ 5.5.1

Pour décrire la structure de ces tableaux , nous adoptons le formalisme relationnel.

DOMAINES :

```
DOM NOMP     = CHAINE(10)
DOM NOM      = CHAINE(16)
DOM IDVUE    = CHAINE(4)
DOM TYPE     = CHAINE('E','C')        (E=entier , C=chaîne )
DOM DROIT    = CHAINE('LS','LE')      (Lecture seule, Lecture/Ecriture
DOM DATE     = CHAINE(6)
DOM CODREST  = CHAINE('0','1','2')
DOM REST1    = ENTIER DE 0 A 999 999

DOM REST2    = ENTIER DE 0 A 999 999
DOM MASQUE   = CHAINE(16)
DOM TYPREP   = CHAINE('PAR','PAV','PDR','PDV','DUS','DUM')
```

2.1. Catalogues de vue globale

Nous adoptons une notation relationnelle succincte pour décrire les différents types de tableaux :

<type-tableau>(<nom-attribut> [:<nom-domaine>],.....)

Les attributs soulignés indiquent les attributs qui constituent la clé.

Les catalogues de vue globale peuvent être divisés en trois classes :

A - Les catalogues administratifs (partie identification du langage)

1°) VG (NVG:IDVUE, CREATEUR:NOMP, DATE, PUTVG, PDOMG, PRELG,PLOCDOM)

Il décrit le nom de la vue globale, celui de son administrateur, la date de création,puis les pointeurs qui repèrent respectivement l'ensemble des utilisateurs autorisés , l'ensemble des domaines et relations attachés à cette vue globale. PLOCDOM repère la table qui établit la correspondance entre les noms de domaines globaux et locaux.

2°) VLVG (NVG:IDVUE, NVL:IDVUE)

Il décrit les noms des vues locales sous-jacentes à la constitution de la vue globale.

3°) UTVG (UTG:NOMP, DROIT)

Il décrit les noms et droits (lecture seule,lecture/écriture) des utilisateurs d'une vue globale.

B - Les catalogues constituant le schema relationnel

1°) DOMG (NDOM:NOM, TYPE, CODREST, REST1, REST2)

Il décrit les noms des domaines attachés au schéma relationnel d'une vue globale avec leur type (chaîne , entier) et la restriction sur les valeurs du domaine.
CODREST indique de quel type de restriction il s'agit :

- '0' : restriction d'entier. REST1 contient la borne inférieure, REST2 la borne supérieure.
- '1' : restriction de chaîne(longueur).REST1 contient la longueur.
- '2' : restriction de chaîne(liste de valeurs),REST1 contient la référence (pointeur idt) vers la liste de valeurs.

2°) RELG (NRG:NOM, MASQCLEP:MASQUE, PATTG,PCLEX, TYPREP,
NRGVOIS:NOM, PCOMPREL, PLOCATT)

Il décrit en premier les informations propres à une relation globale :

- son nom, sa clé primaire(par un masque binaire construit sur le numéro d'ordre des attributs dans la relation) et deux pointeurs qui repèrent respectivement l'ensemble des attributs, l'ensemble des clés extérieures de la relation.

Il décrit également les informations liées à la répartition de la relatio

- son type de répartition, le nom de la relation voisine dans le cas de partitionnement par voisinage(PAV,PDV), et deux pointeur qui repèrent l'ensemble des relations locales composantes et la table qui assure la correspondance entre les noms d'attributs globaux et locaux pour cette relation.

3°) ATTG (NAT:NOM, NDOM:NOM)

Il décrit pour une relation globale le nom de ses attributs avec le domaine correspondant.

4°) CLEX (NRG2:NOM, MASQCLEX:MASQUE)

Il décrit pour une relation globale les attributs(par un masque)qui constituent la clé d'une autre relation NRG2(clé externe).

C - Catalogues décrivant la correspondance global/local

1°) COMPREL (NVL:IDVUE, NRL:NOM, MASQUATT:MASQUE, PREST)

Il décrit les relations locales composantes participant à la construction d'une relation globale donnée.
MASQUATT permet de définir les attributs de la relation globale,qui se retrouvent dans les relations locales composantes.
Dans le cas de partitionnement par restriction(PAR,PDR) , PREST repère la restriction de partitionnement.

2°) LOCDOM (NDOM:NOM, NVL:IDVUE, NDOML:NOM)

Il décrit l'équivalence entre les noms des domaines globaux et ceux des domaines locaux(synonymie).

3°) LOCATT (NAT:NOM, NVL:IDVUE, NATL:NOM)

Il décrit l'équivalence entre les noms des attributs globaux et ceux des attributs locaux.

2.2. Catalogues de vue locale

Ils sont divisés en deux classes (les mécanismes de correspondance noms-éléments sont prédéfinis par le système) :

A - Catalogues administratifs (identification)

1°) VL (NVL:IDVUE,CREATEUR:NOMP,DATE,NBASE:NOM,TYPSGBD;NOM,
PUTVL, PDOML, PRELL)

Il décrit chaque vue locale et base locale,avec l'ensemble des utilisateurs autorisés et l'ensemble des domaines et relations attachés à cette vue.

2°) UTVL (UTLOC:NOMP,DROIT)

Il décrit les noms et droits des utilisateurs d'une vue locale donnée.

B - Catalogues décrivant le schéma relationnel local

Ils sont en grande partie analogues à ceux décrivant le schéma relationnel d'une vue globale.

1°) DOML (NDOML,TYPE,CODREST,REST1,REST2)
2°) RELL (NRL,MASQCLEP,PATTL,PCLEXL)
3°) ATTL (NATL,NDOML)
4°) CLEXL(NRL2,MASQCLEX).

3. SYNTAXE DU LANGAGE DE MANIPULATION

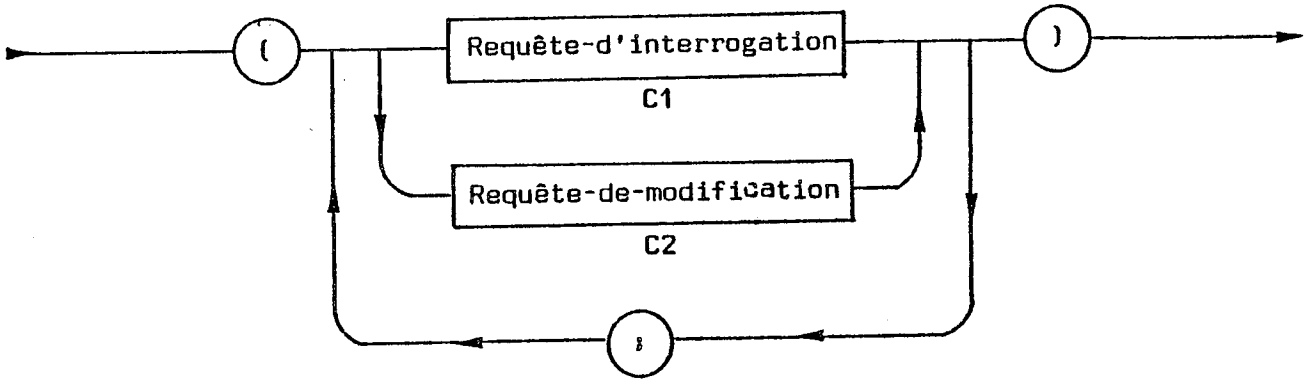
Le langage dont nous donnons la syntaxe, dans ce paragraphe,correspond au langage externe relationnel mis à la disposition d'utlisateurs globaux pour manipuler la base de données répartie.
Il doit être perçu comme un langage simple (de travail) permettant de formuler des requêtes qui correspondent aux opérations de base (accès, insérer,supprimer,modifier) plutôt que comme un nouveau langage relationnel.
Il repose sur l'algèbre relationnelle et il nous est apparu utile de procéder à quelques extensions telles que :

1) - fonctions de calcul : Une expression relationnelle a pour but de définir un ensemble de n-uplets.Il est possible d'appliquer sur cet ensemble des fonctions comme :
   - . CARD : donne la cardinalité de l'ensemble
   - . SOM : totalise les valeurs de l'ensemble (uniquement des valeurs numériques)
   - . MOY : donne la valeur moyenne des éléments de l'ensemble (uniquement valeurs numériques)
   - . MAX,MIN : donnent respectivement la valeur maximale et minimale de l'ensemble(uniquement numérique)
2) - la possibilité d'introduction de valeurs par lecture sur le périphérique d'entrée.Par exemple : la requête suivante
   " lister les numéros des usines implantées dans la ville = ? "
   On note une certaine analogie avec la fonction "EXT" dans Socrate.
3) - la possibilité d'exprimer des calculs élémentaires lors d'une opération de modification.Par exemple SALAIRE = SALAIRE * 1.1.

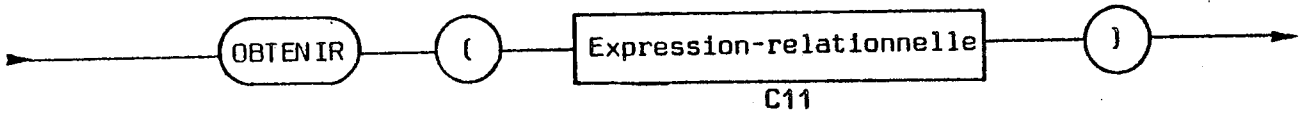
La syntaxe du langage est représentée sous la forme de diagrammes :

C : Transaction

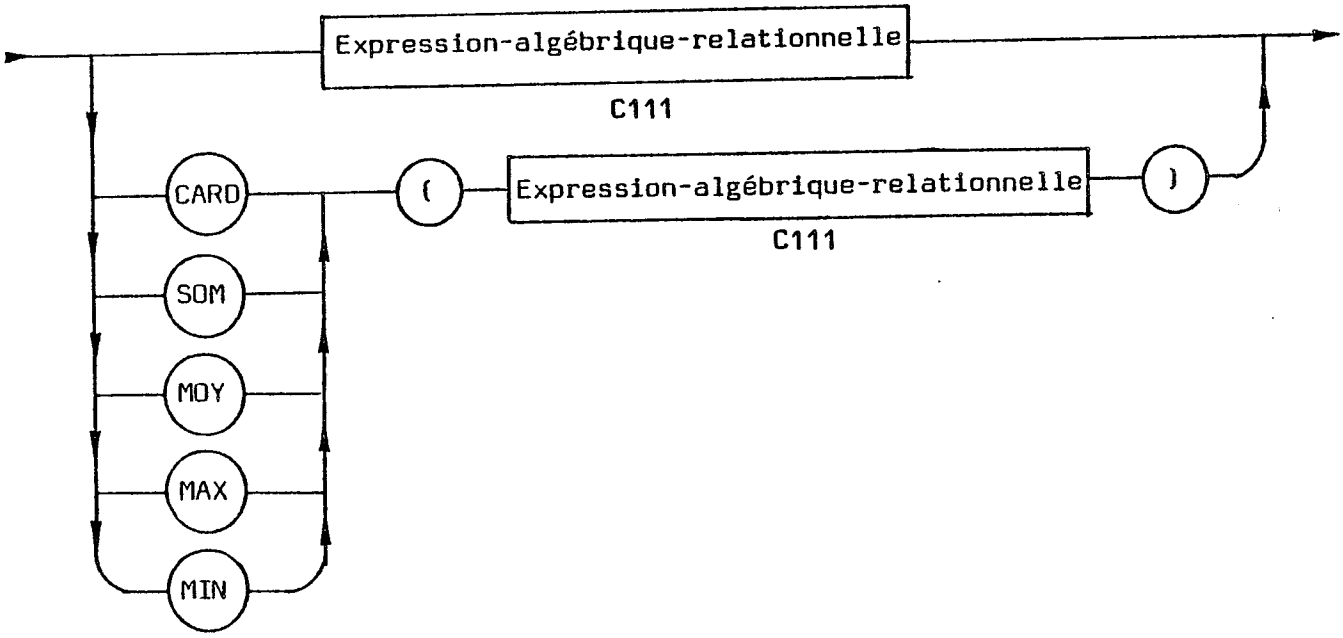
Une transaction globale est une séquence de requêtes d'interrogation ou de modification.



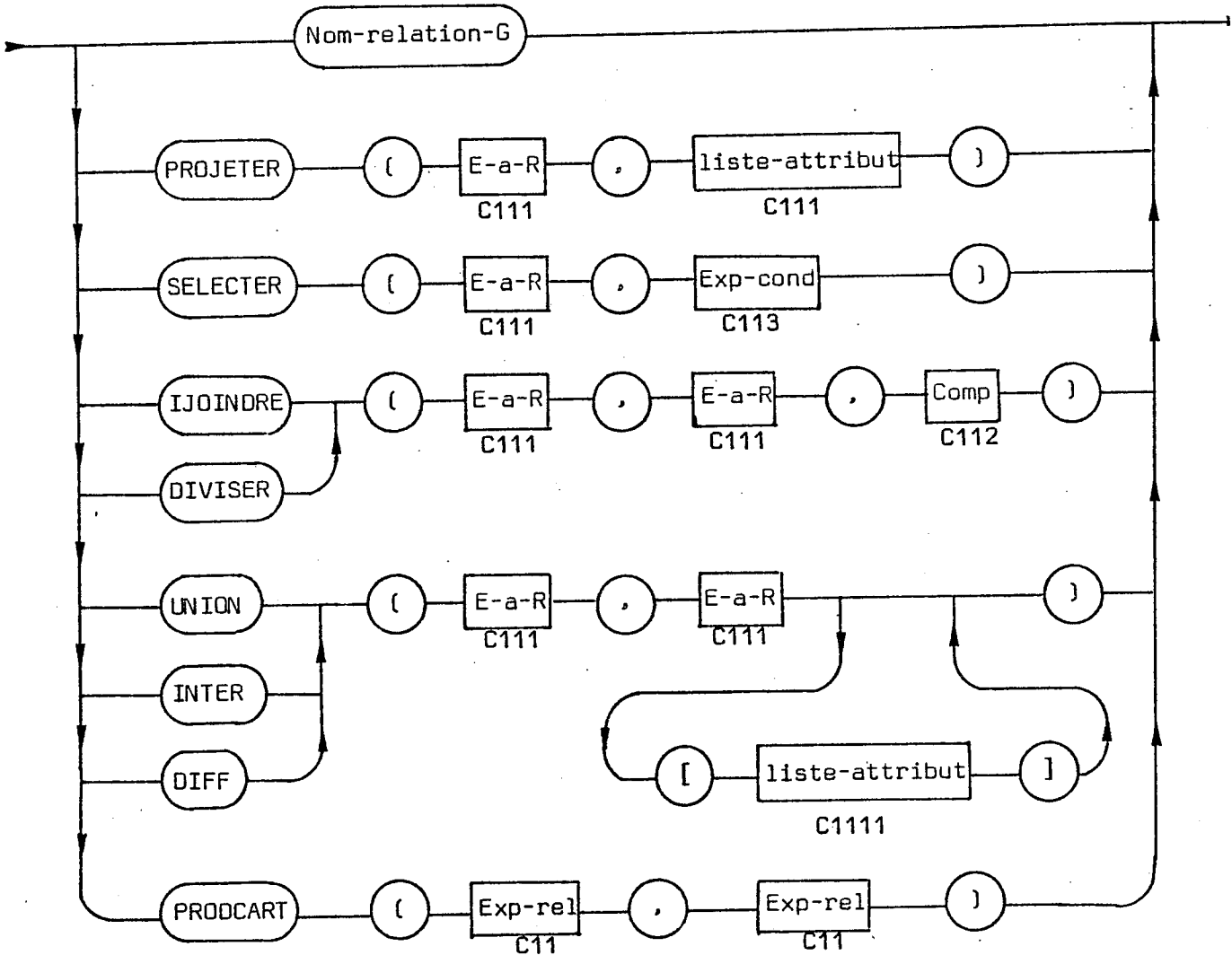
C1 : Requête-d'interrogation



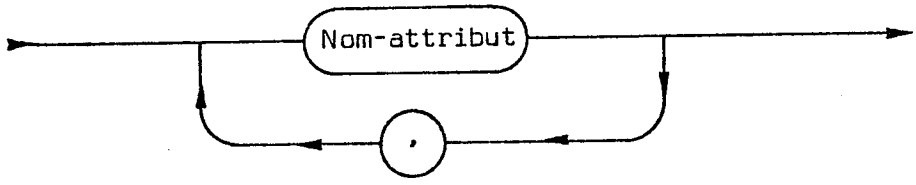
C11 : Expression-relationnelle ou Exp-rel



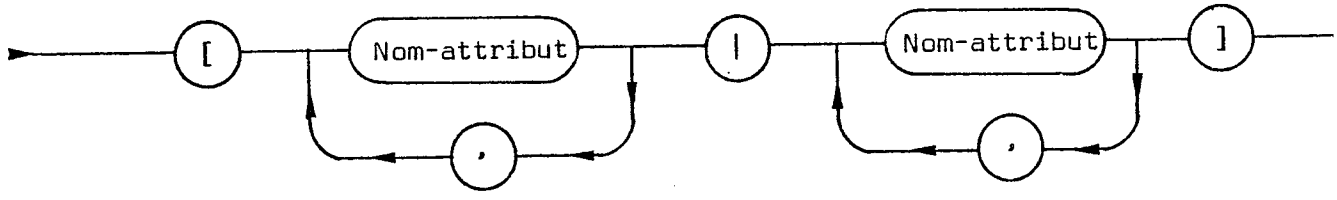
C111 : Expression-algébrique-relationnelle ou E-a-R



C1111 : Liste-attribut



C1112 : Attributs-compatibles ou comp



Par définition,une opération algébrique (projeter,sélecter,etc.,) a pour résultat une relation.Cette relation pouvant servir d'opérande à une autre opération , il importe à l'usager de connaître les noms des attributs qui composent la relation résultante.
Le problème se pose essentiellement, lorsque la relation résultante est construite à partir de deux relations (opérateurs binaires).

a) Opérateurs UNION,INTER,DIFF : Nous convenons d'adopter la règle suivante : les noms d'attributs de la relation résultat sont ceux de la première relation opérande et l'ordre des attributs de la deuxième relation opérande doit être le même que celui des attributs de la première relation pour des raisons de compatibilité.

Ex : soit R (A,B,C) et S (A,D,E) D et C, E et B compatibles

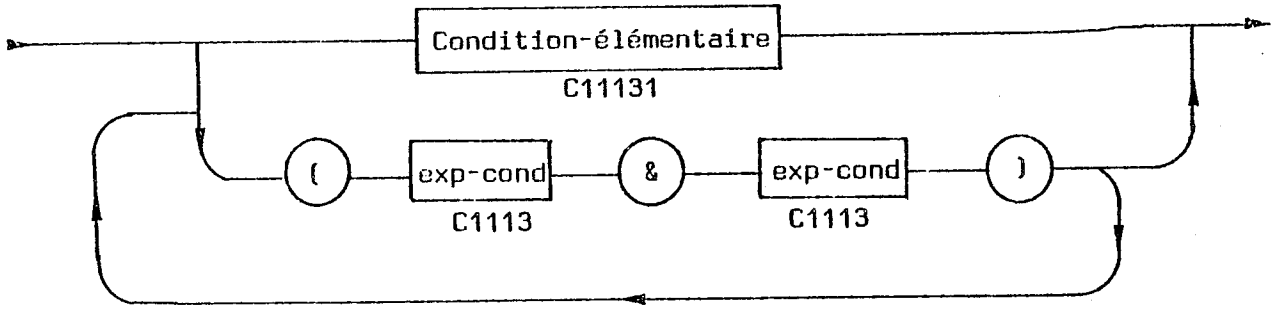
T⟵ UNION(R, S[A,E,D]) la relation T est définie sur les attributs (A,B,C)

b) Opérateurs IJOINDRE,PRODCART : les noms des attributs de la relation résultat seront dans l'ordre des opérandes. Pour éliminer certaines ambiguïtés, lorsque les attributs des deux relations opérandes sont homonymes, leur nom dans la relation résultat sera constitué par la concaténation <attribut>.<relation1> , <attribut>.<relation2>.

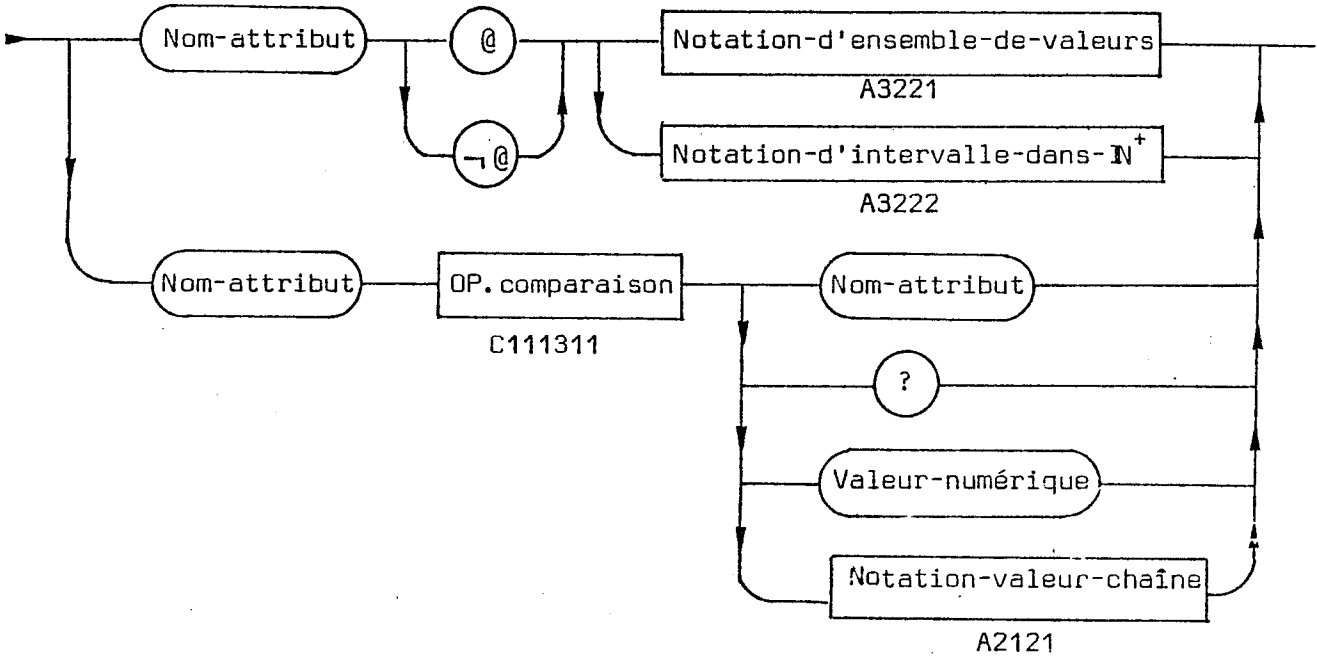
Ex: soit R (A,B,C) et U (A,X,Y) B et Y sont compatibles

T⟵ IJOINDRE(R,U,[B |Y]) T est définie sur les attributs (A.R,B,C,A.U,X)

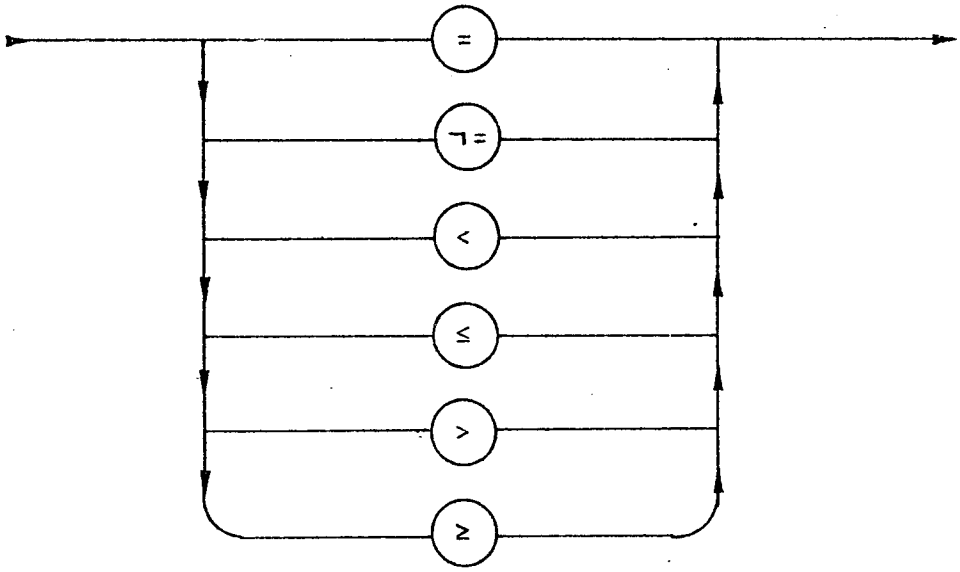
C1113 : Expression-conditionnelle ou exp-cond



C11131 : Condition-élémentaire



C111311 : Opérateur-de-comparaison



Donnons, à titre d'illustration, quelques requêtes d'interrogation, en utilisant l'exemple présenté au § 4.3.2.3 :

RI1 : donner les numéros des usines dont le budget est compris entre 10 000 kF et 50 000 kF

```
OBTENIR(PROJETER(SELECTER(USINE,BUDGET @[10000,50000[),U#))
```

RI2 : moyenne des budgets des usines implantées à Paris

```
OBTENIR(MOY(PROJETER(SELECTER(USINE,LIEU='PARIS'),BUDGET)))
```

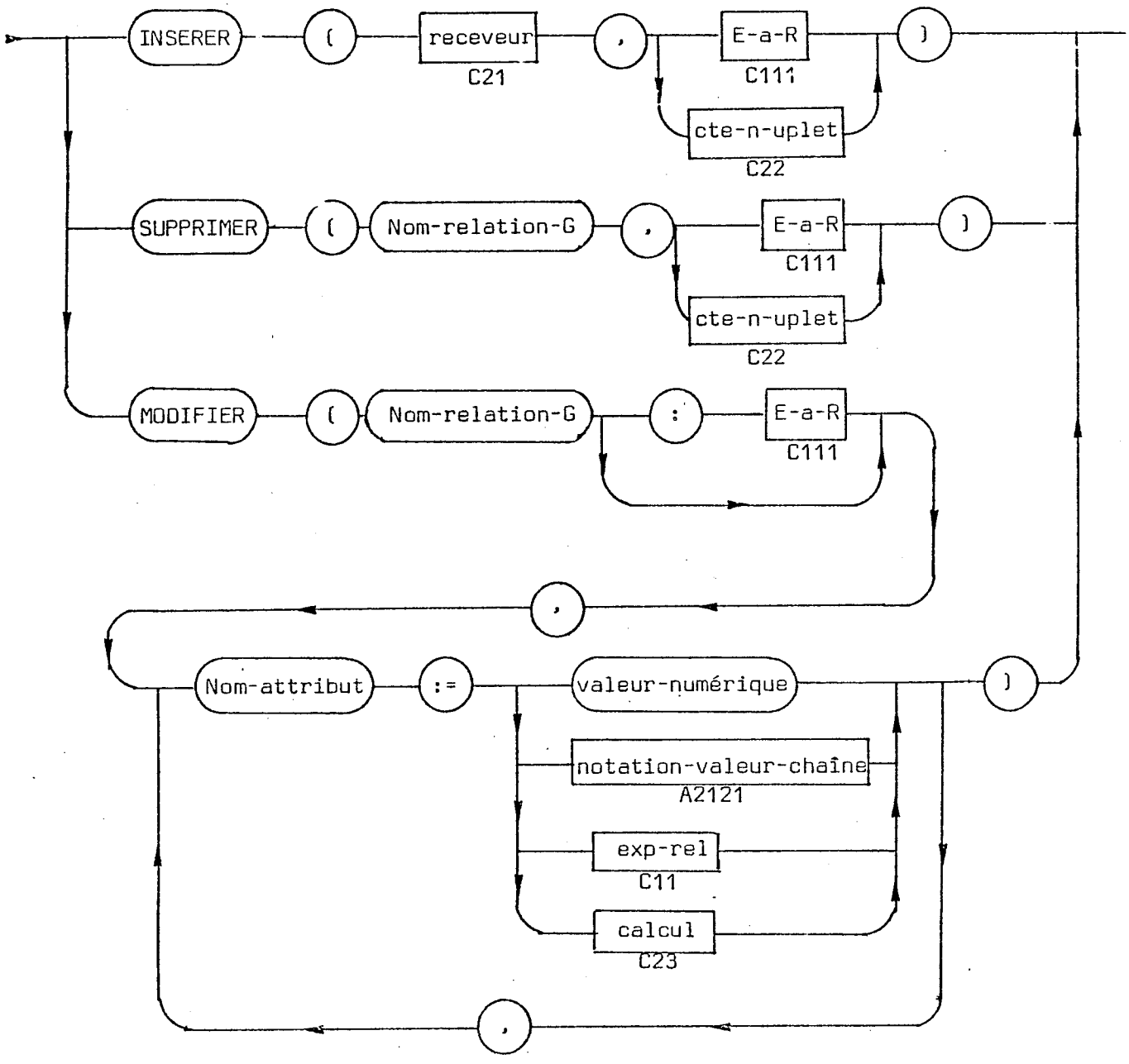
RI3 : donner la ville des usines qui fabriquent le produit numéro P20

```
OBTENIR(PROJETER(IJOINDRE(USINE,SELECTER(FABRICATION,P#='P20'),
                          [U#, U#]),LIEU))
```

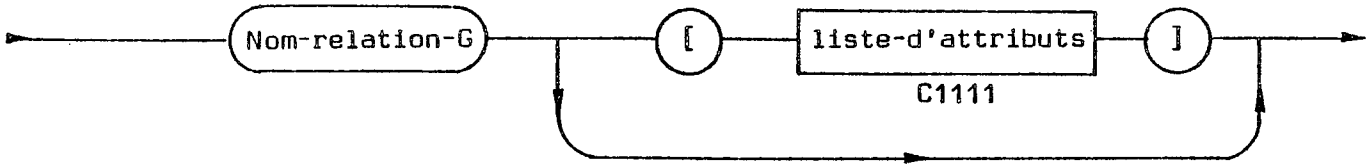
RI4 : donner le numéro des usines qui fabriquent au moins les produits fabriqués par l'usine de numéro ?

```
OBTENIR(DIVISER(PROJETER(FABRICATION,U#,P#),SELECTER(FABRICATION,U#=?),
                          [P#, P#]))
```

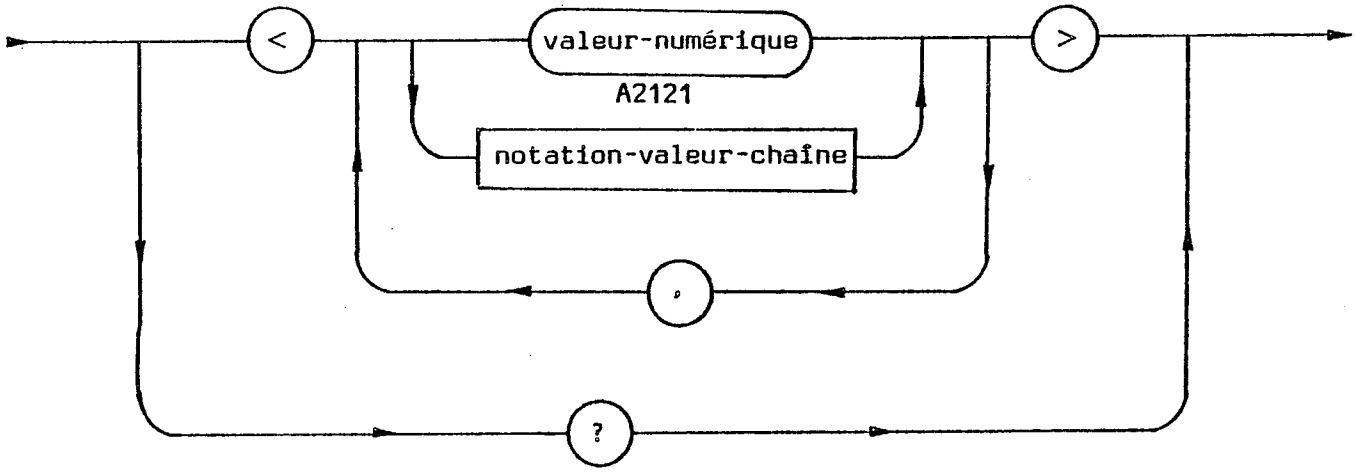
C2 : Requête-de-modification



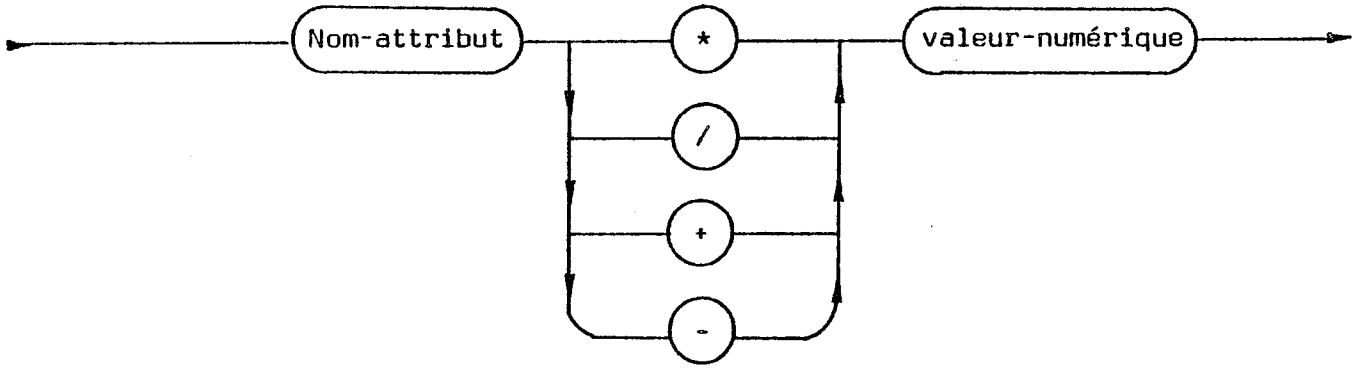
C21 : Receveur



C22 : Constante-de-n-uplet



C23 : Calcul



Exemple de requêtes de modification :

RM1 : insérer dans la relation USINE, un n-uplet donné en paramètre

INSERER(USINE,?) ;

RM2 : supprimer dans FABRICATION toutes les fabrications du produit P31

SUPPRIMER(FABRICATION,SELECTER(FABRICATION,P#='P31')) ;

RM3 : ajouter 50 à la quantité fabriquée pour le produit P39

MODIFIER(FABRICATION:SELECTER(FABRICATION,P#='P39'),QTE:=QTE+50) ;