L'optimisation des requêtes relationnelles : une application de l'intelligence artificielle

Henri Galy

THESE

présentée à

l' Université Scientifique et Médicale de Grenoble

et à

l' Institut National Polytechnique de Grenoble

pour obtenir le grade de

DOCTEUR DE 3ème CYCLE

Informatique

par

Henri GALY

L'OPTIMISATION DES REQUETES RELATIONNELLES : UNE APPLICATION DE L'INTELLIGENCE ARTIFICIELLE

Thèse soutenue le 19 mai 1983 devant la Commission d'Examen :

Monsieur	S. KRAKOWIAK	:	président
Messieurs	M. ADIBA		examinateurs
	C. BOITET		
	B. DAVID		
	C. DELOBEL		
	G. GARDARIN		

UNIVERSITE SCIENTIFIQUE ET MEDICALE DE GRENOBLE

année scolaire 1980-1981

Président de l'Université : M. J.J. PAYAN

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

PROFESSEURS DE 1ère CLASSE

Mlle	AGNIUS DELORD Claudine	Biophysique
	ALARY Josette	Chimie analytique
MM.	AMBLARD Pierre	Clinique dermatologie
	AMBROISE THOMAS Pierre	Parasitologie
	ARNAUD Paul	Chimie
	ARVIEU Robert	Physique nucléaire
	AUBERT Guy	Physique
	AYANT Yves	Physique approfondie
Mme	BARBIER Marie-Jeanne	Electrochimie
MM.	BARBIER Jean-Claude	Physique expérimentale
	BARBIER Reynold	Géologie
	BARJON Robert	Physique nucléaire
	BARNOUD Fernand	Biosynthèse de la cellulose
	BARRA Jean-René	Statistiques
	BARRIE Joseph	Clinique chirurgicale A
	BEAUDOING André	Clinique pédiatrie et puériculture
	BELORISKY Elie	Physique
	BENZAKEN Claude	Mathématiques appliquées
Mme	BERIEL Hélène	Pharmacodynamie
M.	BERNARD Alain	Mathématiques pures
Mme	BERTRANDIAS Françoise	Mathématiques pures
MM.	BERTRANDIAS Jean-Paul	Mathématiques pures
	BEZES Henri	Clinique chirurgicale & traumatologie
	BILLET Jean	Géographie
	BONNET Jean-Louis	Clinique ophtalmologique
	BONNET EYMARD Joseph	Clinique Hépato-gastro-entérologie
Mme	BONNIER Jane-Marie	Chimie générale
MM.	BOUCHERLE André	Chimie et toxicologie
	BOUCHET Yves	Anatomie
	BOUCHEZ Robert	Physique nucléaire
	BRAVARD Yves	Géographie

.../...

MM.	BUTEL Jean	Orthopédie
	CABANEL Guy	Clinique rhumatologie et hydrologie
	CARLIER Georges	Biologie végétale
	CAU Gabriel	Médecine légale et toxicologie
	CAUQUIS Georges	Chimie organique
	CHARACHON Robert	Clinique O.R.L.
	CHATEAU Robert	Clinique neurologique
	CHIBON Pierre	Biologie animale
	COEUR André	Chimie analytique et bromotologique
	COUDERC Pierre	Anatomie pathologique
	CRABBE Pierre	C.E.R.M.O.
	DAUMAS Max	Géographie
	DEBELMAS Jacques	Géologie générale
	DEGRANGE Charles	Zoologie
	DELOBEL Claude	M.I.A.G.
	DELORMAS Pierre	Pneumo-phtisiologique
	DENIS Bernard	Clinique cardiologique
	DEPORTES Charles	Chimie minérale
	DESRE Pierre	Electrochimie
	DODU Jacques	Mécanique appliquée IUT 1
	DOLIQUE Jean-Michel	Physique des plasmas
	DUCROS Pierre	Cristallographie
	FONTAINE Jean-Marc	Mathématiques pures
	GAGNAIRE Didier	Chimie physique
	GASTINEL Noël	Analyse numérique
	GAVEND Jean-Micnel	Pharmacologie
	GEINDRE Michel	Electro-radiologie
	GERBER Robert	Mathématiques pures
	GERMAIN Jean-Pierre	Mécanique
	GIRAUD Pierre	Géologie
	JANIN Bernard	Géographie
	JEANNIN Charles	Pharmacie galénique
	JOLY Jean-René	Mathématiques pures
	KAHANE André	Physique
	KAHANE Josette	Physique
	KLEIN Joseph	Mathématiques pures
	KOSZUL Jean-Louis	Mathématiques pures
	LACAZE Albert	Hermodynamique
	LACHARME Jean	Biologie cellulaire
	LAJZEROWICZ Joseph	Physique

.../...

Mme	LAJZEROWICZ Jeannine	Physique
MM.	LATREILLE René	Chirurgie thoracique
	LATURAZE Jean	Biochimie pharmaceutiques
	LAURENT Pierre	Mathématiques appliquées
	LE NOC Pierre	Bactériologie virologie
	LLIBOUTRY Louis	Géophysique
	LOISEAUX Jean-Marie	Sciences nucléaires
	LOUP Jean	Géographie
	LUU DUC Cuong	Chimie générale et minérale
	MALINAS Yves	Clinique obstétricale
Mlle	MARIOTTE Anne-Marie	Pharmacognostie
MM.	MAYNARD Roger	Physique du solide
	MAZARE Yves	Clinique médicale A
	MICHEL Robert	Minéralogie et pétrographie
	MICOUD Max	Clinique maladies infectieuses
	MOURIQUAND Claude	Histologie
	NEGRE Robert	Mécanique IUT 1
	MOZIERES Philippe	Spectrométrie physique
	OMONT Alain	Astrophysique
	OZENDA Paul	Botanique
	PAYAN Jean-Jacques	Mathématiques pures
	PEBAY PEYROULA Jean-Claude	Physique
	PERRET Jean	Sémeiologie médicale (neurologie)
	PERRIER Guy	Géophysique
	PIERRARD Jean-Marie	Mécanique
	RACHAIL Michel	Clinique médicale B
	RASSAT André	Chimie systématique
	RENARD Michel	Thermodynamique
Mme	RENAUDET Jacqueline	Bactériologie
M.	REVOL Michel	Urologie
Mme	RINAUDO Marguerite	Chimie CERMAV
MM.	DE ROUGEMONT Jacques	Neuro-chirurgie
	SARRAZIN Roger	Clinique chirurgicale B
Mme	SEIGLE MURANDI Françoise	Botanique et crytogamie
MM.	SENGEL Philippe	Biologie animale
	SIBILLE Robert	Construction mécanique IUT 1
	SOUTIF Michel	Physique
	TANCHE Maurice	Physiologie
	VAILLANT François	Zoologie
	VALENTIN Jacques	Physique nucléaire

.../...

MM.	VAN CUTSEM Bernard	Mathématiques appliquées
	VAUQUOIS Bernard	Mathématiques appliquées
	VERAIN Alice	Pharmacie galénique
	VERAIN André	Biophysique
	VIGNAIS Pierre	Biochimie médicale

PROFESSEURS DE 2ème CLASSE

MM.	ARNAUD Yves	Chimie IUT 1
	AURIAULT Jean-Louis	Mécanique IUT 1
	BEGUIN Claude	Chimie organique
	BOITET Christian	Mathématiques appliquées
	BOUTHINON Michel	E.E.A. IUT 1
	BRUGEL Lucien	Energétique IUT 1
	BUISSON Roger	Physique IUT 1
	CASTAING Bernard	Physique
	CHARDON Michel	Géographie
	CHEHIKIAN Alain	E.E.A. IUT 1
	COHEN Henri	Mathématiques pures
	COHENADDAD Jean-Pierre	Physique
	COLIN DE VERDIERE Yves	Mathématiques pures
	CONTE René	Physique IUT 1
	CYROT Michel	Physique du solide
	DEPASSEL Roger	Mécanique des fluides
	DOUCE Roland	Physiologie végétale
	DUFRESNOY Alain	Mathématiques pures
	GASPARD François	Physique
	GAUTRON René	Chimie
	GIDON Maurice	Géologie
	GIGNOUX Claude	Sciences nucléaires
	GLENAT René	Chimie organique
	GOSSE Jean-Pierre	E.E.A. IUT 1
	GROS Yves	Physique IUT 1
	GUITTON Jacques	Chimie
	HACQUES Gérard	Mathématiques appliquées
	HERBIN Jacky	Géographie
	HICTER Pierre	Chimie
	IDELMAN Simon	Physiologie animale
	JOSELEAU Jean-Paul	Biochimie
	JULLIEN Pierre	Mathématiques appliquées
	KERCKOVE Claude	Géologie

.../...

MM.	KRAKOWIACK Sacha	Mathématiques appliquées
	KUHN Gérard	Physique IUT 1
	KUPKA Yvon	Mathématiques pures
	LUNA Domingo	Mathématiques pures
	MACHE Régis	Physiologie végétale
	MARECHAL Jean	Mécanique
	MICHOULIER Jean	Physique IUT 1
Mme	MINIER Colette	Physique IUT 1
MM.	NEMOZ Alain	Thermodynamique
	NOUGARET Marcel	Automatique IUT 1
	OUDET Bruno	Mathématiques appliquées
	PEFFEN René	Métallurgie IUT 1
	PELMONT Jean	Biochimie
	PERRAUD Robert	Chimie IUT 1
	PERRIAUX Jean-Jacques	Géologie minéralogie
	PERRIN Claude	Sciences nucléaires
	PFISTER Jean-Claude	Physique du solide
	PIERRE Jean-Louis	Chimie organique
Mlle	PIERY Yvette	Physiologie animale
MM.	RAYNAUD Hervé	Mathématiques appliquées
	RICHARD Lucien	Biologie végétale
	ROBERT Gilles	Mathématiques pures
	ROBERT Jean-Bernard	Chimie physique
	ROSSI André	Physiologie végétale
	SAKAROVITCH Michel	Mathématiques appliquées
	SARROT REYNAUD Jean	Géologie
	SAXOD Raymond	Biologie animale
Mme	SOUTIF Jeanne	Physique
MM.	STUTZ Pierre	Mécanique
	VIALON Pierre	Géologie
	VIDAL Michel	Chimie organique
	VIVIAN Robert	Géographie

CHARGES D'ENSEIGNEMENT PHARMACIE

MM.	ROCHAS Jacques	Hygiène et hydrologie
	DEMENGE Pierre	Pharmacodynamie

PROFESSEURS SANS CHAIRE (médecine)

M.	BARGE Michel	Neuro-chirurgie

.../...

MM.	BOST Michel	Pédiatrie
	BOUCHARLAT Jacques	Psychiatrie
	CHAMBAZ Edmond	Biochimie (hormonologie)
	CHAMPETIER Jean	Anatomie
	COLOMB Maurice	Biochimie
	COULOMB Max	Radiologie
Mme	ETERRADOSSI Jacqueline	Physiologie
MM.	FAURE Jacques	Médecine légale
	GROULADE Joseph	Biochimie A
	HOLLARD Daniel	Hématologie
	HUGONOT Robert	Gérontologie
	JALBERT Pierre	Histologie
	MAGNIN Robert	Hygiène
	PHELIP Xavier	Rhumatologie
	REYMOND Jean-Charles	Chirurgie générale
	STIEGLITZ Paul	Anesthésiologie
	VROUSOS Constantin	Radiothérapie

MAITRES DE CONFERENCES AGREGES (médecine)

MM.	BACHELOT Yvan	Endocrinologie
	BENABID Alim Louis	Médecine et chirurgie
	BERNARD Pierre	Gynécologie obstétrique
	CONTAMIN Charles	Chirurgie thoracique
	CORDONNIER Daniel	Néphrologie
	CROUZET Guy	Radiologie
	DEBRU Jean-Luc	Médecine interne
	DYON Jean-François	Chirurgie infantile
	FAURE Claude	Anatomie et organogènèse
	FAURE Gilbert	Urologie
	FLOYRAC Roger	Biophysique
	FOURNET Jacques	Hépato-gastro-entérologie
	GAUTIER Robert	Chirurgie générale
	GIRARDET Pierre	Anesthésiologie
	GUIDICELLI Henri	Chirurgie générale
	GUIGNIER Michel	Thérapeutique (réanimation)
	JUNIEN-LAVILLAUROY Claude	Clinique O.R.L.
	KOLODIE Lucien	Hématologie biologique
	MALLION Jean-Michel	Médecine du travail
	MASSOT Christian	Médecine interne
	MOUILLON Michel	Ophtalmologie

.../...

MM.	PARAMELLE Bernard	Pneumologie
	RACINET Claude	Gynécologie-Obstétrique
	RAMBAUD Pierre	Pédiatrie
	RAPHAEL Bernard	Stomatologie
	SCHAEFER René	Cancérologie
	SEIGNEURIN Jean-Marie	Bactériologie-virologie
	SOTTO Jean-Jacques	Hématologie
	STOEBNER Pierre	Anatomie-pathologique

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Président : Daniel BLOCH

Vice-Présidents : René CARRE
Hervé CHERADAME
Marcel IVANES

Année universitaire 1982-1983

Professeurs des Universités

ANCEAU	François	E.N.S.I.M.A.G.	LACOUME	Jean Louis	E.N.S.I.E.G.
BARRAUD	Alain	E.N.S.I.E.G.	LATOMBE	Jean Claude	E.N.S.I.M.A.G.
BAUDELET	Bernard	E.N.S.I.E.G.	LESIEUR	Marcel	E.N.S.H.G.
BESSON	Jean	E.N.S.E.E.G.	LESPINARD	Georges	E.N.S.H.G.
BLIMAN	Samuel	E.N.S.E.R.G.	LONGEQUEUE	Jean Pierre	E.N.S.I.E.G.
BLOCH	Daniel	E.N.S.I.E.G.	MAZARE	Guy	E.N.S.I.M.A.G.
BOIS	Philippe	E.N.S.H.G.	MOREAU	René	E.N.S.H.G.
BONNETAIN	Lucien	E.N.S.E.E.G.	MORET	Roger	E.N.S.I.E.G.
BONNIER	Etienne	E.N.S.E.E.G.	MOSSIERE	Jacques	E.N.S.I.M.A.G.
BOUVARD	Maurice	E.N.S.H.G.	PARIAUD	Jean Charles	E.N.S.E.E.G.
BRISSONNEAU	Pierre	E.N.S.I.E.G.	PAUTHENET	René	E.N.S.I.E.G.
BUYLE BODIN	Maurice	E.N.S.E.R.G.	PERRET	René	E.N.S.I.E.G.
CAVAIGNAC	Jean François	E.N.S.I.E.G.	PERRET	Robert	E.N.S.I.E.G.
CHARTIER	Germain	E.N.S.I.E.G.	PIAU	Jean Michel	E.N.S.H.G.
CHENEVIER	Pierre	E.N.S.E.R.G.	POLOUJADOFF	Michel	E.N.S.I.E.G.
CHERADAME	Hervé	U.E.R.M.C.P.P.	POUPOT	Christian	E.N.S.E.R.G.
CHERUY	Arlette	E.N.S.I.E.G.	RAMEAU	Jean Jacques	E.N.S.E.E.G.
CHIAVERINA	Jean	U.E.R.M.C.P.P.	RENAUD	Maurice	U.E.R.M.C.P.P.
COHEN	Joseph	E.N.S.E.R.G.	ROBERT	André	U.E.R.M.C.P.P.
COUMES	André	E.N.S.E.R.G.	ROBERT	François	E.N.S.I.M.A.G.
DURAND	Francis	E.N.S.E.E.G.	SABONNADIERE	Jean Claude	E.N.S.I.E.G.
DURAND	Jean Louis	E.N.S.I.E.G.	SAUCIER	Gabrielle	E.N.S.I.M.A.G.
FELICI	Noël	E.N.S.I.E.G.	SCHLENKER	Claire	E.N.S.I.E.G.
FOULARD	Claude	E.N.S.I.E.G.	SCHLENKER	Michel	E.N.S.I.E.G.
GENTIL	Pierre	E.N.S.E.R.G.	SERMET	Pierre	E.N.S.E.R.G.
GUERIN	Bernard	E.N.S.E.R.G.	SILVY	Jacques	U.E.R.M.C.P.P.
GUYOT	Pierre	E.N.S.E.E.G.	SOHM	Jean Claude	E.N.S.E.E.G.
IVANES	Marcel	E.N.S.I.E.G.	SOUQUET	Jean Louis	E.N.S.E.E.G.
JAUSSAUD	Pierre	E.N.S.I.E.G.	VEILLON	Gérard	E.N.S.I.M.A.G.
JOUBERT	Jean Claude	E.N.S.I.E.G.	ZADWORNY	François	E.N.S.E.R.G.
JOURDAIN	Geneviève	E.N.S.I.E.G.			

Professeurs associés

BASTIN	Georges	E.N.S.H.G.	GANDINI	Alessandro	U.E.R.M.C.P.P.
BERRIL	John	E.N.S.H.G.	HAYASHI	Hirashi	E.N.S.I.E.G.
CARREAU	Pierre	E.N.S.H.G.			

Professeurs Université des Sciences Sociales (Grenoble II)

BOLLIET	Louis	CHATELIN	Françoise

Professeurs E.N.S. Mines de Saint Etienne

RIEU	Jean	SOUSTELLE	Michel

Chercheurs du C.N.R.S.

FRUCHART	Robert	Directeur de recherche	HOPFINGER	Emil	Maître de reche
VACHAUD	Georges	Directeur de Recherche	JOUD	Jean Charles	Maître de reche
ALLIBERT	Michel	Maître de recherche	KAMARINOS	Georges	Maître de reche
ANSARA	Ibrahim	Maître de Recherche	KLEITZ	Michel	Maître de reche
ARMAND	Michel	Maître de recherche	LANDAU	Ioan-Dore	Maître de reche
BINDER	Gilbert		LASJAUNIAS	J.C.	
CARRE	René	Maître de recherche	MERMET	Jean	Maître de reche
DAVID	René	Maître de recherche	MUNIER	Jacques	Maître de reche
DEPORTES	Jacques		PIAU	Monique	
DRIOLE	Jean	Maître de recherche	PORTESEIL	Jean Louis	
GIGNOUX	Damien		THOLENCE	Jean Louis	
GIVORD	Dominique		VERDILLON	André	
GUELIN	Pierre				

Chercheurs du Ministère de la Recherche et de la Technologie
(Directeurs et Maîtres de recherche - E.N.S. Mines de Saint Etienne)

LESBATS	Pierre	Directeur de recherche	LALAUZE	René	Maître de reche
BISCONDI	Michel	Maître de recherche	LANCELOT	Francis	Maître de reche
KOBYLANSKI	André	Maître de recherche	THEVENOT	François	Maître de reche
LE COZE	Jean	Maître de recherche	TRAN MINH	Canh	Maître de reche

Personnalités habilitées à diriger des travaux de recherche
(Décision du Conseil Scientifique)

E.N.S.E.E.G.

ALLIBERT	Colette
BERNARD	Claude
BONNET	Roland
CAILLET	Marcel
CHATILLON	Catherine
CHATILLON	Christian
COULON	Michel
DIARD	Jean Paul
EUSTATOPOULOS	Nicolas
FOSTER	Panayotis
GALERIE	Alain
HAMMOU	Abdelkader
MALMEJAC	Yves (CENG)
MARTIN GARIN	Régina
NGUYEN TRUONG	Bernadette
RAVAINE	Denis
SAINFORT	(CENG)
SARRAZIN	Pierre
SIMON	Jean Paul
TOUZAIN	Philippe
URBAIN	Georges (Laboratoire des ultra-réfractaires ODEILLO).

E.N.S.Mines Saint Etienne

GUILHOT	Bernard
THOMAS	Gérard
DRIVER	Julien

E.N.S.E.R.G.

BARIBAUD	Michel
BOREL	Joseph
CHOVET	Alain
CHEHIKIAN	Alain
DOLMAZON	Jean Marc
HERAULT	Jeanny
MONLLOR	Christian

E.N.S.I.E.G.

BORNARD	Guy
DESCHIZEAUX	Pierre
GLANGEAUD	François
KOFMAN	Walter
LEJEUNE	Gérard
MAZUER	Jean
PERARD	Jacques
REINISCH	Raymond

E.N.S.H.G.

ALEMANY	Antoine
BOIS	Daniel
DARVE	Félix
MICHEL	Jean Marie
OBLED	Charles
ROWE	Alain
VAUCLIN	Michel
WACK	Bernard

E.N.S.I.M.A.G.

BERT	Didier
CALMET	Jacques
COURTIN	Jacques
COURTOIS	Bernard
DELLA DORA	Jean
FONLUPT	Jean
SIFAKIS	Joseph

U.E.R.M.C.P.P.

CHARUEL	Robert

C.E.N.G.

ADET	Jean
BOEURE	Philippe (LETI)
DELHAYE	Jean Marc (STT)
DUPUY	Michel (LETI)
JOUVE	Hubert (LETI)
NICOLAU	Yvan (LETI)
NIFENECKER	Hervé
PERROUD	Paul-H
PEUZIN	Jean Claude (LETI)
TAIEB	Maurice
VINCENDON	Marc

Laboratoires extérieurs :

C.N.E.T.

DEMOULIN	Eric
DEVINE	R.A.B.
GERBER	Roland
MERCKEL	Gérard
PAULEAU	Yves

I.N.S.A. Lyon

GAUBERT C.

****

Je tiens à remercier

Monsieur S. KRAKOWIAK, Professeur à l'Université Scientifique et Médicale de Grenoble, qui m'a fait l'honneur de présider le jury de cette thèse

Monsieur le Professeur C. DELOBEL, Directeur du laboratoire IMAG, qui a dirigé et conseillé mes travaux

Messieurs G. GARDARIN, Professeur à Paris VI, M. ADIBA et C. BOITET, Professeurs à Grenoble, et B. DAVID, Chargé de recherche au CNRS, qui ont accepté d'examiner et de juger ce travail.

Je tiens en particulier à remercier mes collègues de travail, notamment ceux de l'équipe MICROBE — ainsi que le service de D. IGLESIAS qui a assuré le tirage de cette Thèse.

Grenoble, le 19 mai 1983

Gaby

TABLE DES MATIERES

INTRODUCTION

1 - LES SYSTEMES DE GESTION DE BASES DE DONNEES REPARTIES.
LE MODELE RELATIONNEL

2 - L'OPTIMISATION DES SGBD

3 - OBJECTIFS DE LA THESE

4 - EXAMEN PREALABLE

1 - LES SYSTÈMES DE GESTION DE BASES DE DONNÉES. LE MODÈLE RELATIONNEL

Les systèmes de Gestion de Bases de Données (SGBD) ont connu un développement considérable depuis les années 70. Initialement structurées de manière hiérarchique, liées à l'implémentation physique des données, les Bases de Données ont évolué vers des structures plus complexes mais plus souples (modèles réseaux), permettant une certaine indépendance entre le niveau physique et le niveau logique. Cette évolution était nécessaire pour répondre à la complexité croissante des SGBD et au besoin de faciliter l'évolution, l'adaptabilité à des machines très diverses, le développement d'applications nouvelles.

Poussant jusqu'au bout cette recherche d'indépendance logique/physique, CODD a défini en 1970 le modèle relationnel, modèle basé sur des notions mathématiques abstraites et non plus sur celles imposées par le stockage de l'information. [21]

Douze ans après, il faut reconnaître que le modèle relationnel reste un produit expérimental, même si quelques SGBD l'utilisant viennent d'être commercialisés : MRDS, QBE, SQL/DS, INGRES, ORACLE. Cela tient aux difficultés de mise en oeuvre, en particulier au manque de performances inhérent à la généralité et à l'abstraction de ce modèle.

Nous pensons que cette situation est provisoire, et ce pour deux raisons :

1) Le modèle relationnel s'est imposé en ce qui concerne l'informatique répartie, tous les prototypes de SGBD répartis l'utilisant. L'informatique répartie est à la veille d'un développement considérable.

2) Plus fondamentalement, la tendance à un niveau élevé d'abstraction est une tendance inéluctable, au même titre que l'évolution vers des langages de haut niveau.

L'explosion de la micro-informatique va précipiter cette évolution, de la même manière qu'elle a précipité l'évolution des langages (expension du PASCAL survenue brutalement, alors que ce langage existe depuis longtemps).

Un "langage machine" est plus performant qu'un "langage de haut niveau". Les modèles hiérarchiques seront toujours plus performants que les modèles relationnels. Mais ils limiteront, de par le fait qu'ils sont liés au niveau physique, les possibilités de concevoir et de développer des applications réellement nouvelles. En d'autres termes, ils manqueront toujours de puissance.

Les modèles de bases de données et les langages de manipulation de données qui s'imposeront dans l'avenir seront sans doute différents du modèle relationnel de CODD, mais ce qui est certain, c'est qu'ils permettront à un "utilisateur", qui sera souvent en même temps un "administrateur de Base de données", de formuler son problème sans trop se soucier de savoir ce qui se passe réellement dans sa machine.

Le problème sera donc pour nous de limiter le surcoût inhérent à cette indépendance physique / logique et à ce niveau d'abstraction plus élevé. Pour cela, nous allons partir du modèle relationnel qui est le modèle de haut niveau que nous connaissons actuellement.

2 - L'OPTIMISATION DES S.G.B.D. RÉPARTIS

Il n'existe pas de solution exacte, de méthode unique. En général, il est impossible de déduire à priori d'une requête répartie, la requête équivalente la moins coûteuse. C'est vrai aussi pour un système centralisé.

Dans le cas d'un modèle abstrait tel le modèle relationnel, les requêtes sont formulées sans considération d'efficacité. C'est le système qui doit assurer l'efficacité de la requête (un coût et un temps de réponse minimums) et ce de manière automatique. C'est là que réside l'intérêt de tels modèles. Il faut donc :

- En premier lieu éliminer à coup sûr les stratégies d'exécution inacceptables.
- En second lieu s'approcher de la meilleure stratégie possible, en veillant à ce que le surcoût de l'optimisation ne dépasse pas le gain obtenu : en d'autres termes, l'optimisation doit être optimale et adaptée à l'application.

Cette remarque préalable étant faite, les objectifs d'un optimiseur de SGBD Réparti sont :

1. minimiser le temps d'exécution
2. minimiser le temps de transmission
3. améliorer le parallélisme (exécution et transmission)
4. améliorer la distribution du trafic et de la charge.

Ces objectifs sont en partie contradictoires. Des choix et des compromis seront nécessaires. L'optimiseur d'un SGBDR doit donc être paramétrable, ses stratégies doivent pouvoir être modifiées.

Le programme optimiseur doit être invisible à l'utilisateur. Il peut toutefois jouer certains rôles "visibles", en particulier refuser certaines requêtes (ou mettre en garde l'utilisateur) pour éviter le blocage ou la saturation du Système.

Le choix de la localisation des données (partition/duplication) se situe à un autre niveau. C'est pourquoi nous n'en parlerons pas. Nous ne parlerons pas non plus des techniques à l'étude en ce qui concerne l'architecture du matériel, les processeurs spécialisés, etc.

Nous admettrons que les requêtes sont déjà décomposées en relations de base localisées, chaque relation ayant un, et un seul site. On peut toujours se ramener à ce cas-là.

a) Dans un premier temps, l'optimiseur à partir d'une requête donnée, fournit une requête optimisée dans l'absolu. C'est-à-dire où le coût d'exécution (correspondant au cas centralisé) est minimum. Dans le cas d'un arbre d'opérateurs relationnels, il s'agit de *minimiser les transferts inter-opérateurs*.

b) A partir du graphe obtenu, il faut faire une nouvelle *optimisation qui prenne en compte l'aspect réparti*, ainsi que l'aspect multi-utilisateurs et l'état du système (SGBDR, et aussi réseau).

L'hypothèse est que l'étape b) est impossible dans l'étape a). Concernant le surcoût entrainé par la répartition, il faut signaler que ce surcoût a longtemps été considéré comme déterminant par rapport au coût des exécutions locales (l'exécution proprement dite).

Michel ADIBA et Patricia GRIFFITHS-SELINGER ont montré que du point de vue temps, comme du point de vue coût, le facteur déterminant était l'exécution locale. [3]

Ce résultat ayant été obtenu avec un débit modeste (50 Kb/s), et étant donné le fait que pour l'avenir prévisible le débit des réseaux, même non locaux, va croitre beaucoup plus vite que les performances d'Entrée/Sortie - nous pouvons dire que les SGBD Répartis vont ressembler fortement de ce point de vue à des SGBD centralisés multiprocesseurs.

Le problème de la phase b) n'est donc pas de diminuer les transferts sur le réseau, mais d'optimiser la charge globale du système en équilibrant la charge des processeurs, et cela grâce à l'accroissement du parallélisme, en fonction de la puissance et de la disponibilité des sites.

Dans cette thèse nous étudions l'optimisation de la requête relationnelle elle-même. L'objet de l'étude est de minimiser les transferts inter-opérateurs.

3 - OBJECTIFS DE LA THÈSE

La plupart des méthodes que nous allons exposer ne sont pas nouvelles. Le lecteur qui voudra faire le point pourra se reporter à la thèse de I.S. PAIK [57] ainsi qu'à la bibliographie donnée en annexe. Nous supposerons acquise la terminologie de COOD [21,22] et celle de DELOBEL-ADIBA [5].

Toutefois, la multiplicité de ces méthodes, ainsi que, il faut bien le reconnaître, le peu de réalisation effective dans le cadre d'un système industrialisé, nécessitent une critique et un choix. C'est là l'objet de la première partie. Pour des raisons évidentes après avoir lu ce qui précède, nous n'avons pas pris en compte la répartition dans notre étude. Parmi les modèles relationnels, on pouvait choisir de formaliser les requêtes avec un langage basé sur le calcul relationnel, ou bien sur l'algèbre relationnelle. CODD a montré que tous deux étaient complets [22]. Le langage de type algèbre relationnelle a été choisi pour des raisons historiques propres à l'IMAG.[52][2][34]

Dans la deuxième partie, nous développerons un aspect original qui a fait l'objet d'une réalisation partielle : l'application des Systèmes Généraux de Transformation Paramétrées à l'optimisation des SGBD Relationnels. Nous prendrons en compte l'aspect multi-utilisateurs et multi-requêtes.

Avant d'aller plus loin, nous allons donner un exemple qui servira de référence pour la suite. [2][5]

4 - EXEMPLE PRÉALABLE

Soit les relations suivantes :

- WAGON (NW, NV, NTW, CHARGE) où :

NW est le numéro d'immatriculation du Wagon. Ce numéro est une clé pour la relation, c'est pourquoi il est souligné.

NV est un numéro de voie dans le cas où le wagon est détaché (sinon NV est indéfini).

NTW est le numéro du Type de Wagon.

CHARGE est une variable logique qui prend deux valeurs : "vrai" ou "faux".

La relation WAGON définit tous les wagons existants.

- TYPEW (NTW, PV, CM, CAT, DESCR)

PV est le Poids à Vide de ce type de wagon, CM la charge maximale.

CAT est la catégorie ("citerne", "frigo", "voyageurs",etc.).

DESCR est la description technique du type.

- VOIE (NV, GD, GA, TY)

NV est le numéro de voie.

GD la gare de départ, GA la gare d'arrivée.

TY le type de la voie.

TY = cc : voie de circulation. Il n'y a que des trains (wagons attachés)

ch : voie de chargement et de garage. Il n'y a que des wagons libres

tr : voie de triage. Il y a des trains, et des wagons détachés.

Lorsque TY = tr ou TY = ch on a bien sûr GA = GD. sinon, GA ≠ GD.

Les voies sont à sens unique.

- TRAIN (NT, NV)

  NT est le numéro du train.

- WAGON-TRAIN (NW, NT)

  Cette relation définit tous les trains en fonction de leurs wagons.

  WAGON-TRAIN [NW] ⊂ WAGON [NW]

Les horaires et les dates n'interviennent pas : le système doit fonctionner en temps réel.

Les clés principales sont des clés simples (formées d'un seul attribut). C'est une simplification.

Le cardinal des relations est supposé connu à chaque instant :

|WAGON| = 500 000

|WAGON-TRAIN| = 300 000

|TRAIN| = 10 000

|TYPEW| = 100

|VOIE| = 8 000

Les relations VOIE et TYPEW connaissent un taux de mise à jour très faible, au contraire de TRAIN et WAGON-TRAIN. WAGON est dans un situation intermédiaire.

Pour chaque domaine, on a un certain nombre de valeurs possibles :

WAGON

NW	NV	NTW	CHARGE
500 000	1 000	100	2

TYPEW

NTW	PV	CM	CAT	DESCR
100	10	10	10	100

VOIE

NV	GD	GA	TY
8 000	200	200	3

TRAIN

NT	NV
10 000	7 000

WAGON-TRAIN

NW	NT
300 000	10 00

Pour simplifier, on admet que les valeurs de chaque domaine ont une longueur égale à 1, sauf DESCR qui vaudra 10.

VOIE est partitionné en VOIE1 et VOIE2 :

VOIE1 comprend toutes les voies de circulation (TY = cc, GA ≠ GD) soient 2 000 unités

VOIE2 comprend 5 000 voies de triage et 1 000 voies de chargement.

PARTIE I

L'OPTIMISATION DES REQUETES RELATIONNELLES

CHAPITRE 1 : ASPECTS GENERAUX DU PROBLEME

CHAPITRE 2 : LES PRINCIPALES TRANSFORMATIONS ALGEBRIQUES

CHAPITRE 3 : LES AUTRES TRANSFORMATIONS ALGEBRIQUES

CHAPITRE 1 : ASPECTS GENERAUX DU PROBLEME

1.1 - DISCUSSION SUR LES STRATEGIES D'OPTIMISATION

1.1.1 - LA STRATEGIE STATIQUE

1.1.2 - LA STRATEGIE DYNAMIQUE

1.2 - LES OPERATEURS RELATIONNELS

1.3 - EXEMPLE

1 - 1. DISCUSSION SUR LES STRATÉGIES D'OPTIMISATION

Nous allons revenir sur un problème qui est l'objet d'une controverse. Il s'agit du choix de la stratégie de localisation des sites d'exécution, et au travers de ce choix de problèmes heuristiques, problèmes qui ont une incidence sur l'optimisation des requêtes non-réparties elles-mêmes.

PAIK définit quatre stratégies différentes en fonction des deux critères :

- études de plan d'exécution exhaustive/limitée
- localisation statique/dynamique [57].

EPSTEIN et STONEBRAKER ont montré que les études non exhaustives étaient assez médiocres et qu'il fallait les écarter dans la mesure du possible [31].
Voyons pour le deuxième critère.

Le seul moyen de connaître effectivement la meilleure répartition de la requête c'est d'essayer toute les répartitions possibles. Dans la pratique, ce n'est possible que pour des requêtes standard ; et encore, le fait que le contexte évolue (charge du réseau, présence d'applications non réparties, etc.) rend tout étalonnage hypothétique.

Il faut donc une heuristique. Cette heuristique peut être statique ou dynamique, ou bien combinée.

1.1.1. LA STRATEGIE STATIQUE

On examine toutes les solutions possibles et on essaie de les évaluer à priori à l'aide de certaines caractéristiques :

- cardinal des relations
- chemins d'accès
- facteurs de sélectivité.

Le calcul des facteurs de sélectivité (cf. ADIBA-DELOBEL [5]) nécessite un certain nombre de données statistiques en particulier sur les domaines des attributs des relations :

- y a-t-il redondance ? Si oui, quel est le taux ?
- Répartition des valeurs entre les bornes (uniformes, courbe de Gauss, etc.) ?

Généralement, on avance que la nécessité de données statistiques peut poser un grave problème de mise à jour et que c'est le principal inconvénient des méthodes statiques.

Nous pensons que c'est faux.

En effet, en vertu de la loi des grands nombres, *plus le domaine des valeurs est grand, plus le cardinal des relations est élevé et meilleures seront les statistiques; et elles auront d'autant moins besoin d'être mises à jour.*

Prenons un exemple extrême illustrant ce phénomène bien connu des statisticiens. Soit la relation RESERVATION (NØ, DATE, NOM, PRENOM, GD, GA, CAT-AGE) où CAT-AGE est une fonction de l'âge du voyageur qui permet de calculer la réduction à laquelle il a éventuellement droit :

CAT-AGE = 0 si le voyageur a moins de 10 ans
= 1 si le voyageur a entre 10 et 18 ans
= 2 si le voyageur a entre 18 et 65 ans
= 3 si le voyageur a plus de 65 ans.

Il est clair que le taux de mise à jour de la relation est extrêmement élevé ; les réservations étant limitées à 72 heures, ce taux est de 100 % pour trois jours.

Il est donc hors de question de gérer des statistiques instantanées, ou même réactualisées par périodes très rapprochées (d'une heure par exemple). La taille de la relation rendrait cela prohibitif.

Mais, il est parfaitement inutile de gérer des statistiques exactes. Ce qui nous intéresse, c'est de savoir (entre autre) le pourcentage p de voyageurs de plus de 65 ans (CAT-AGE = 3) parmi les réservations. Il est clair que ce pourcentage est bien connu de la SNCF, ainsi que sa variation saisonnière et/ou journalière. C'est d'ailleurs cette connaissance qui sert de base à la mise en place des réductions.

Idem pour n le nombre de voyageurs ayant réservé pour Grenoble-Paris. Si donc on cherche quel est le nombre n_1 de voyageurs de plus de 65 ans sur le train Grenoble-Paris du 1er octobre 1981, il suffit de faire le produit:

$$n \times p = n_1$$

Le résultat sera moins précis que les facteurs n et p, pris séparément. Mais, comme n_1 est nettement plus petit que n, l'erreur a moins de conséquences...

A la limite, si nous cherchons le nombre n_2 de voyageurs de plus de 65 ans, prenant le train Cahors-Capdenac du 1er octobre 1981 à 20 h, et ayant pour prénom Marcel, le résultat sera non significatif en ce sens qu'il est impossible de savoir à priori si $n_2 = 0,1,2$ ou 5 avec une quelconque garantie - mais ce qui est sûr, c'est que n_2 est très petit, et que le problème est résolu! La précision relative diminue, mais pas la précision absolue.

Les simulations amènent à une sous-estimation de la précision des évaluations statistiques parce qu'elles ne sont, précisément que des simulations. Une base de données réellement existante, a un contexte riche de renseignements, il ne s'agit pas de variables aléatoires.

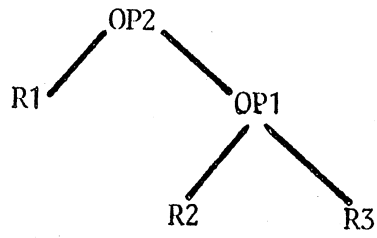
Une source d'erreur à considérer est le grand nombre d'opérateurs que peut éventuellement comporter la requête, les erreurs se cumulant. Mais il semble difficile de trouver des cas réels où le grand nombre d'opérateurs n'entraine pas une très forte diminution du volume des relations intermédiaires à mesure où on s'éloigne des feuilles.

1.1.2. LA STRATEGIE DYNAMIQUE

Cette stratégie n'est pas exclusive d'une étape initiale statique. Toutefois, dans l'esprit des partisans de la stratégie dynamique, il s'agit bien de supprimer l'évaluation statique des requêtes. [33][34][53][55]

L'idée de base est très simple : après avoir effectué toutes les optimisations statiques qui ne nécessitent pas d'évaluations et de statistiques (restructuration algébriques minimisant les transferts inter-opérateurs), on localise progressivement les opérateurs en fonction des résultats d'exécutions partielles de la requête.

Exemple



On choisira le site d'exécution de l'opérateur OP1 en fonction par exemple du volume V de R2 et R3.

Si V(R2) < V(R3), on effectuera OP1 sur le site de R2.

Le résultat intermédiaire sur OP1(R2,R3) permettra de localiser OP2.

Pour améliorer le mécanisme, on évite d'avoir à attendre la fin d'OP1 en utilisant une notion heuristique, le SEUIL : fonction définie par la comparaison entre le volume R1 et le volume (estimé) de OP1(R2,R3). Cette fonction permet d'introduire une grande souplesse dans le mécanisme car on peut y ajouter d'autre paramètres concernant la charge du système, la vitesse des processeurs etc. [33]

On pourrait penser que l'optimisation dynamique est plus précise que l'optimisation statique. Il n'en est rien.

Tous les algorithmes qui l'utilisent ont pour principe la localisation d'un noeud en fonction de ses descendants. L'hypothèse (non exprimée) qui est derrière est que le volume de la relation résultat d'un opérateur binaire est toujours inférieur à la somme des volumes des relations opérandes. Ce n'est pas toujours vrai, et point n'est besoin de recourir au produit cartésien pour trouver un contre exemple :

R1 — site1 — V = 4

A	B
a1	b1
a2	b1

R2 — site 2 — V = 6

B	C
b1	c1
b1	c2
b1	c3

R1 * R2 — V = 18

A	B	C
a1	b1	c1
a1	b1	c2
a1	b1	c3
a2	b1	c1
a2	b1	c2
a2	b1	c3

R3 — site 3 — V = 8

C	D
c1	d1
c2	d2
c3	d3
c3	d4

Si donc on effectue R1 * R2 * R3, qu'on veut ce résultat sur le site S1 en supposant que R1,R2,R3 sont localisées respectivement sur S1, S2, S3 (tous distincts), un algorithme dynamique [33] fournira les résultats suivants :

$V(R1) < V(R2) \Rightarrow Loc\ (x2) = site\ (R2) = S2$

d'où transfert de R1 sur S2 : T1=4.

$$\left.\begin{array}{l} V(R3) > V(R1) \\ V(R3) > V(R2) \end{array}\right| \Rightarrow Loc\ (x1) = Site\ (R3)=S3$$

d'où transfert de (R1 * R2) soit S2 : T2=18.

transfert du résultat final sur S1: T3=32.

[nous laissons le lecteur vérifier]

d'où au total : T = 4+18+32=54.

Eventuellement, une localisation dynamique donnera Loc (x1)=Site (R2)=S2

d'où T4 = V (R3)=8 et T=44.

La solution optimale consiste à transférer avant toute éxécution R2 et R3 sur S1 : T=6+8=14.

1.2. LES OPERATEURS RELATIONNELS [5]

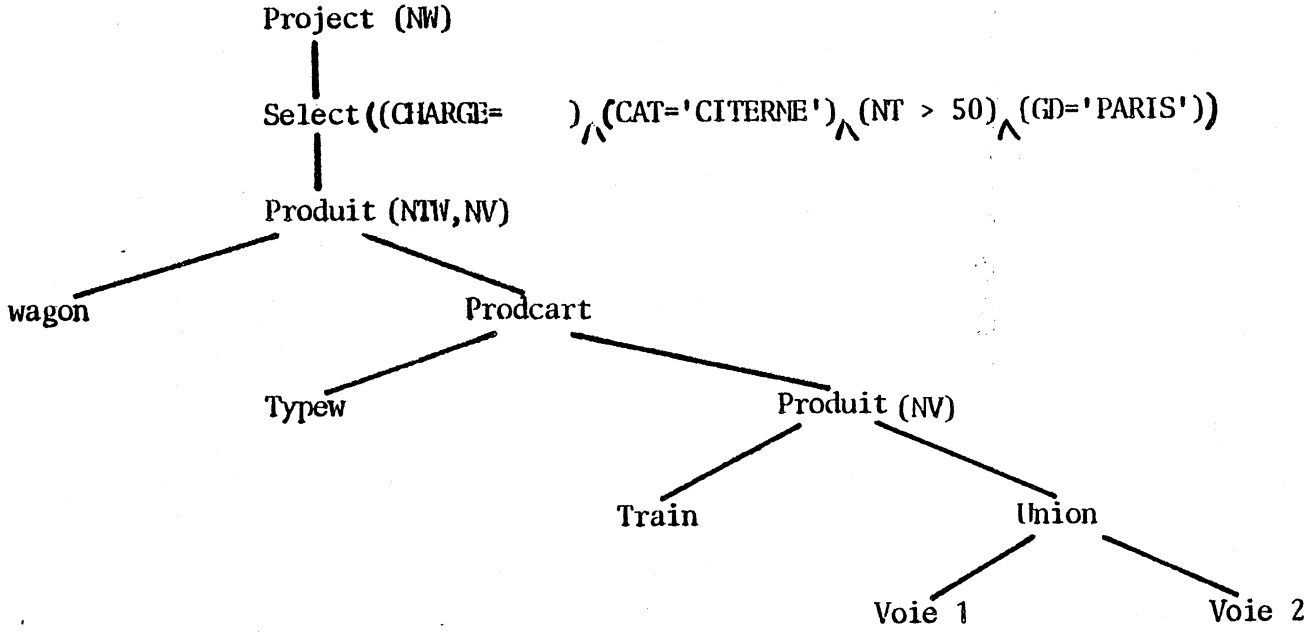
Il y a deux sortes d'opérateurs :

- Les opérateurs ensemblistes usuels : UNION, INTERSECTION, DIFFERENCE, PRODUIT CARTESIEN.
- Les opérateurs spécifiques à l'algèbre relationnelle : PROJECTION, SELECTION, PRODUIT, θ-PRODUIT, SOMME, DIVISION, ANTIPROJECTION, COMPLEMENT.

Cinq opérateurs suffisent à décrire tous les autres, ce sont l'UNION, la DIFFERENCE, le PRODUIT CARTESIEN, la PROJECTION et la SELECTION. Mais, les autres opérateurs facilitent l'expression des requêtes relationnelles.
Si nous reprenons l'exemple des chemins de fer, la requête

"Donner les numéros des wagons chargés, de catégorie citerne, appartenant aux trains de n° supérieurs à 50 et partant de Paris"

S'exprime sous la forme de la requête relationnelle suivante :



(Cet exemple est repris tout au long de la Thèse)

Cette requête peut-être obtenue de différentes manières :

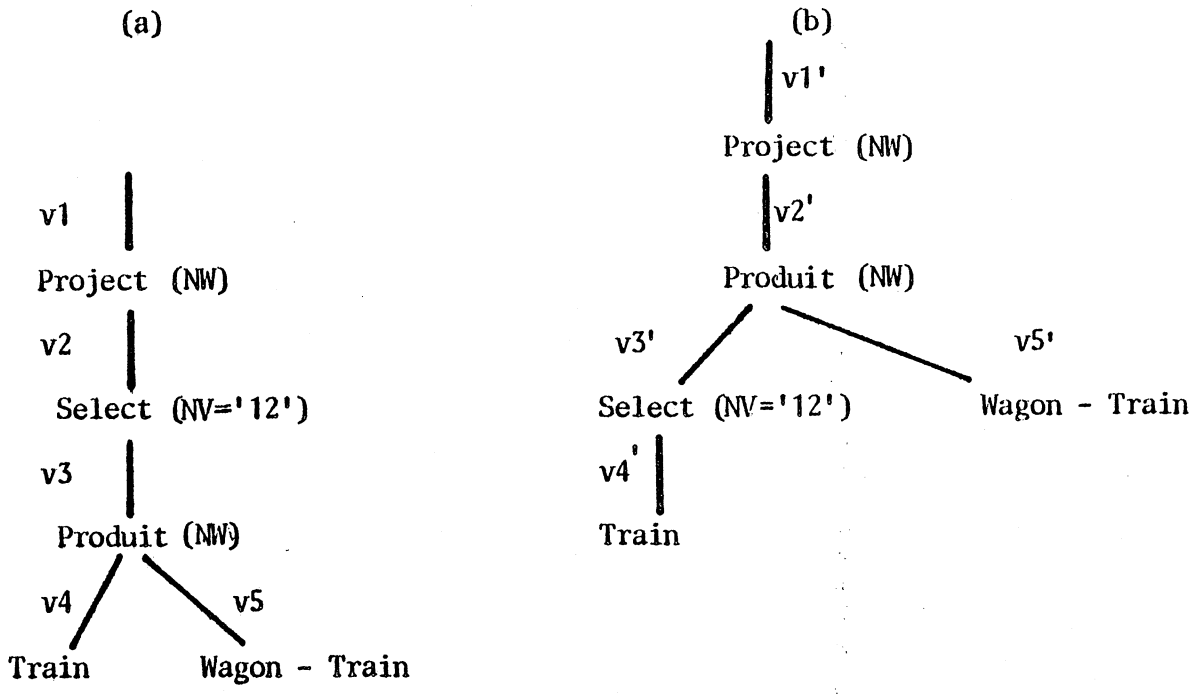
- par décomposition d'une requête globale exprimée au moyen de l'algèbre relationnelle - cf POLYPHEME [1][2][15]
- par traduction d'une requête exprimée dans un autre langage, notamment un langage naturel. Dans MICROBE, un traducteur transforme la requête initiale de type SEQUEL en un arbre d'opérateurs URANUS. [37][38]
- en étant précompilée.

Cette obtention peut-être automatique ou manuelle.

1.3. EXEMPLE :

La première partie de l'optimisation consiste en la minimisation absolue du volume des données manipulées par la requête, autrement dit la minimisation des transferts inter-opérateurs. Cette partie est d'une importance fondamentale ainsi qu'un exemple simple le montre.

Soit la requête : "Numéro des wagons circulant sur la voie n° 12", elle peut s'écrire sous forme de requêtes localisées :



Le lecteur vérifiera empiriquement que ces deux requêtes sont équivalentes.

Le problème essentiel est le Produit sur l'attribut NV des relations TRAIN et WAGON-TRAIN.

Supposons, pour simplifier, que les domaines des attributs des relations ont toujours la même longueur , soit 1 unité. Le volume V (R) de la relation R est alors égal au produit $|R|$. d (R), où d (R) est le degré de la relation R, autrement dit le nombre de ses attributs, et $|R|$ le cardinal de R.

V4 = V(TRAIN) = 10 000 x 2 = 20.10^3

V5 = V(WAGON-TRAIN) = 300 000 x 2 = 600.10^3

A chaque élément de la relation WAGON-TRAIN correspond un élément de TRAIN. Le produit TRAIN * WAGON-TRAIN est défini sur l'attribut NV, le résultat a pour attribut (NV,NT,NW).

D'où |TRAIN * WAGON-TRAIN |= 300 000 et d(TRAIN * WAGON-TRAIN)= 3

$V_3 = 300\ 000 \times 3 = 900.10^3$

Admettons qu'il y ait trois trains circulant sur la voie n° 12 et que chacun de ces trains ait trente wagons,ce qui correspond à des valeurs moyennes.

Il y a 90 wagons qui vérifient la condition (NV = '12').

$V_1 = V_2 \times 1/3 = 90$, ce qui est négligeable

$V_2 = 90 \times 3 = 270$, ce qui est négligeable,

$$V_a = \sum_{i=1}^{5} V_i \times 1{,}52.10^6$$

$V'_4=V_4$, $V'_5=V_5$, $V'_2=V_2$, $V'_1=V_1$

$$V_b = \sum_{i=1} V'_i \simeq V'4 + V'_5 = 620.10^3$$

Le gain représenté par la requête (b) est égal à $\frac{V_a - V_b}{V_a}$ soit près de 60 %

Cet écart est important, mais il peut être beaucoup plus important encore, en particulier si l'arbre est plus complexe.

Un calcul analogue pour l'exemple précédent (cf 1.2) donne un total des volumes de :

$VT_1 = 28.\ 10^6$

avec les hypothèses suivantes :

- 10% des wagons sont des citernes, 75% des wagons sont chargés
- 10% des voies et des trains partent de PARIS
- La quasi-totalité des trains ont un numéro supérieur à 50.

CHAPITRE 2 : LES PRINCIPALES TRANSFORMATIONS ALGEBRIQUES

2.1 - REDUCTIONS PREALABLES

2.1.1 - ARBRES R-HOMOGENES

2.1.2 - ELIMINATION DES EXPRESSIONS VIDES REDUCTIONS

2.1.3 - REGLES D'IDEMPOTENCE ET RECONNAISSANCE DE SOUS-ARBRES COMMUNS

2.2 - DISTRIBUTION DES OPERATEURS UNAIRES

2.2.1 - DISTRIBUTION DE LA SELECTION

2.2.2 - FACTORISATION DES EXPRESSIONS CONDITIONNELLES

2.2.3 - DISTRIBUTION DE LA PROJECTION

Une requête relationnelle s'exprime sous forme de requêtes relationnelles localisées.

A partir d'une requête relationnelle localisée, il s'agit d'obtenir une autre requête équivalente, dont la somme des transferts inter-opérateurs aura été minimisée, et ce, en plusieurs étapes. Chaque étape consiste à appliquer un ensemble de règles de transformations algébriques, ensemble que nous appellerons tables de transformations.

Il y a un grand nombre de règles. Les plus simples sont celles qui ne nécessitent pas de connaissances statistiques, ni de renseignements autres que la requête elle-même, exprimée sous forme d'un arbre d'opérateurs relationnels - et qui ne donnent pour résultat qu'un seul arbre (transformations de type [1 : 1]).

Nous supposons que la requête à optimiser est déjà localisée, et qu'elle est parfaitement correcte.

Toutes les définitions d'opérateurs qui suivent sont tirées du livre de DELOBEL et ADIBA auquel le lecteur se réfèrera pour plus de détails.[5]

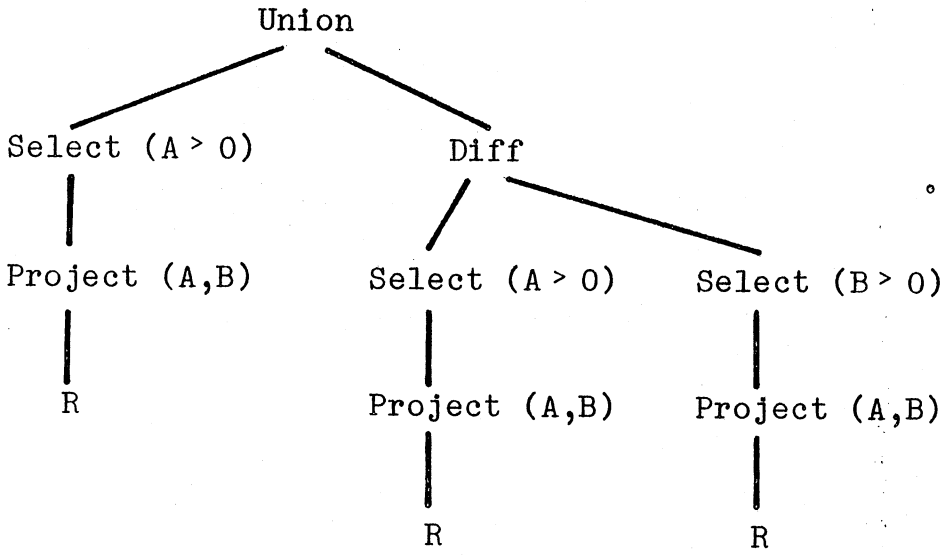
Le choix de ces 12 opérateurs (cf 1 . 2) se justifie pour les raisons suivantes : ils forment un ensemble cohérent du point de vue de leurs définitions, et ils couvrent un large éventail. La plupart des langages relationnels algébriques les utilisent, encore que certains opérateurs soient rarement implémentés : ANTIPROJECTION, COMPLEMENT, SOMME, DIVISION.

2.1 - REDUCTIONS PREALABLES

Les processus de décomposition et de traduction automatique font apparaitre de nombreuses redondances ou expressions inutiles.
Il convient de commencer par les éliminer.

2.1.1 - REDUCTION DES ARBRES R - HOMOGENES

Il peut y avoir des sous-arbres entièrement définis sur une seule relation de base, par exemple :



Dans cet exemple, toutes les feuilles de l'arbre sont identiques à une même relation R. Nous dirons qu'il est R-homogène.
Pour chaque sous-arbre R-homogène, il faut :

a) Déplacer vers le haut toutes les opérations de projection et les combiner en une seule en faisant l'union des listes d'attributs de chaque projection.

b) Combiner les autres opérateurs au moyen des règles suivantes :

i) $(R : E_1) \cup (R : E_2) = R : (E_1 \vee E_2)$

ii) $(R : E_1) \cap (R : E_2) = R : (E_1 \wedge E_2)$

iii) $(R : E_1) - (R : E_2) = R : (E_1 \wedge (\neg E_2))$

iv) $(R : E_1) : E_2 = R : (E_1 \wedge E_2)$

E_1 et E_2 étant des expressions conditionnelles.
Les conditions composées ainsi obtenues se simplifient en général. On trouvera dans la thèse de CALECA les tables logiques qui permettent ces simplifications [15].

Ici, on obtient :

Project (A,B)
|
Select ((A>0∧B ≤ 0) ∨ (A > 0))
|
R

Soit, après simplification :

Project (A, B)
|
Select (A >0)
|
R

2.1.2 - ELIMINATION DES EXPRESSIONS VIDES, REDUCTIONS

Certaines transformations peuvent entrainer des expressions vides, notamment la combinaison des conditions vues précédemment. Ainsi :

$T = (R : (A < 0)) \cap (R : (A = 2))$

$= R : ((A < 0) \wedge (A = 2))$

$= R :$ Faux

La relation résultante T est égale à la relation vide.

Soit R et S des relations, relations qui peuvent être des relations intermédiaires, autrement dit des sous-arbres d'opérateurs relationnels

Soit X_1, Y_1 des attributs; X, Y des ensembles d'attributs

Soit $\emptyset$ la relation vide et NIL un ensemble vide d'attributs

Soit Θ un opérateur appartenant à $\{=, \neq, <, \leq, >, \geq\}$

On a alors la table de réduction de la figure 1.

FIGURE 1 : TABLE DE REDUCTION (1ère partie)

NOM DE REGLE	OPERATEUR	CONDITION	SOUS-ARBRE A REDUIRE	SOUS-ARBRE REDUIT
R 1 a	Projection	$(R=\emptyset) \vee (X=NIL)$	R [X]	$\emptyset$
R 2 a	Sélection	$(R=\emptyset) \vee (E=faux)$	R : E	$\emptyset$
R 2 b	"	$(R\neq\emptyset) \wedge (E=Vrai)$	R : E	R
R 3 a	Antipro-jection	$(R=\emptyset) \vee (X=NIL)$	R]X[	$\emptyset$
R 4	Complément	$R = \emptyset$	$\neg$R	$\emptyset$
R 5 a	Union	$(R=\emptyset) \wedge (S=\emptyset)$	R $\cup$ S	$\emptyset$
R 5 b	"	$(R\neq\emptyset) \wedge (S=\emptyset)$	R $\cup$ S	R
R 5 c	"	$(R=\emptyset) \wedge (S\neq\emptyset)$	R $\cup$ S	S
R 6	Intersection	$(R=\emptyset) \vee (S=\emptyset)$	R $\cap$ S	$\emptyset$
R 7 a	Différence	$R = \emptyset$	R - S	$\emptyset$
R 7 b	"	$(R\neq\emptyset) \wedge (S=\emptyset)$	R - S	R
R 8 b	Produit Cartésien	$(R=\emptyset) \vee (S=\emptyset)$	R X S	$\emptyset$
R 9 a	Somme	$(R=\emptyset) \wedge (S=\emptyset)$	R + S	$\emptyset$
R 9 b	"	$(R\neq\emptyset) \wedge (S=\emptyset)$	R + S	R
R 9 c	"	$(R=\emptyset) \wedge (S\neq\emptyset)$	R + S	S
R10 a	Produit	$(R=\emptyset) \vee (S=\emptyset)$	R * S	$\emptyset$
R11	θ-Produit	$(R=\emptyset) \vee (S=\emptyset)$	$R*(X_1 \theta Y_1)S$	$\emptyset$
R12 a	Division	$R = \emptyset$	R (X,Y)÷S(Y)	$\emptyset$
R12 b	"	$(R\neq\emptyset) \wedge (S=\emptyset)$	R (X,Y)÷S(Y)	R(X)

La plupart de ces réductions sont triviales.

R 10 a et R 11 découlent de R 8.

Pour R 9b, il faut considérer la définition de la somme (cf [5])

$$||(R + S)\ (z)|| \triangleq ||R\ (X)|| \vee ||\ S\ (Y)|| \text{ avec } Z=X \cup Y$$

Si R (X) = Ø, alors le prédicat $||R(X)||$ est toujours faux, d'où :

$$||(R + S)\ (Z)|| \overset{\Delta}{=} ||S\ (Y)||$$

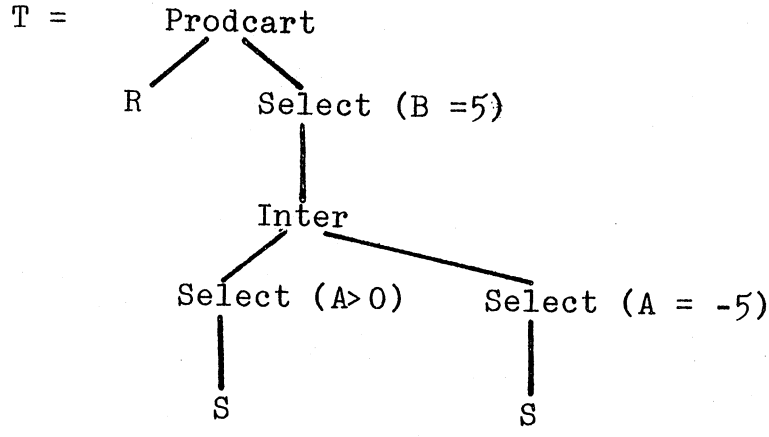
R 9 a et R 9 c sont obtenues de manière analogue.

Pour R 12 b, il faut considérer aussi la définition de la division :

$$||(R \div S)\ (X)|| \overset{\Delta}{=} \forall y \in S(Y)\ (||R\ (X,y)||)$$

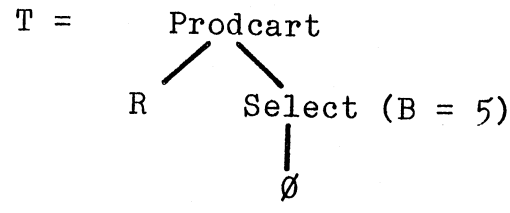
Si S (Y) = Ø, alors le prédicat sur R est toujours vérifié.

Exemple :



Il y a un regroupement R-homogène sur la relation S, qui donne après réduction la relation vide.

d'où :



On applique successivement R 2 a, puis R 8, d'où :

$T = \emptyset$

Si l'on considère les attributs des relations intermédiaires, on a d'autres réductions intéressantes :

FIGURE 2 : TABLE DE REDUCTION (2ème partie)

NOM DE REGLE	OPERATEUR	CONDITION	SOUS-ARBRE A REDUIRE	SOUS-ARBRE REDUIT
R1 b	Projection	$(R \neq \emptyset) \wedge (X \neq NIL) \wedge (LAT(R)=X)$	R [X]	R
R3 b	Anti-projection	$(R \neq \emptyset) \wedge (X \neq NIL) \wedge (LAT(R)=X)$	R]X[	R
R9 d	Somme	$LAT(R) = LAT(S)$	R+S	$R \cup S$
R9 e	"	$LAT(R) \cap LAT(S) = NIL$	R+S	R X S
R 10b	Produit	$LAT(R) = LAT(S)$	R*S	$R \cap S$
R 10c	"	$LAT(R) \cap LAT(S)=NIL$	R*S	R X S

On appelle LAT (R) la liste d'attributs sur laquelle R est définie.

Les règles R 1b et R 3 b correspondent au cas où l'opérateur est inutile.

Les règles R 9 d, R 9 e, R 10 b, R 10 c découlent des définitions de la SOMME et du PRODUIT. Elles permettent de remplacer un opérateur extrêmement coûteux par un autre relativement moins coûteux.

Deux remarques importantes :

- Il est nécessaire, dans une phase initiale, de calculer les attributs des relations intermédiaires. Cette phase initiale pourra être fusionnée avec une autre (par exemple vérification que la requête est correcte).

- La réduction se fait en un seul parcours de l'arbre, de bas en haut (des feuilles vers la racine). Le coût de l'algorithme de réduction est donc proportionnel au nombre de noeuds de l'arbre.

Les règles sont nombreuses, 25 au total, mais pour chaque opérateur, il n'y a que deux ou trois règles à tester en moyenne.

Notons que :

- Après avoir exécuté R 9 d, on teste : R 5 a, R 5 b, R 5 c
- Après avoir exécuté R 9 e ou R 10 c, on teste : R 8
- Après avoir exécuté R 10 b, on teste : R 6.

2.1.3 - REGLES D'IDEMPOTENCE ET RECONNAISSANCE DE SOUS-ARBRES COMMUNS.

Les opérateurs relationnels vérifient les règles d'idempotence suivantes (I) :

I 1 : $\neg(\neg R) = R$

I 2 : $R \cup R = R$

I 3 : $R \cap R = R$

I 4 : $R - R = \emptyset$

I 5 : $R * R = R$

I 6 : $R + R = R$

I 7 : $R \bullet R = \emptyset$

En ce qui concerne les expressions conditionnelles non atomiques, il faut chercher à mettre en évidence des expressions identiques à Vrai ou à Faux, de manière à obtenir des expressions vides ou inutiles.

On utilise les règles de l'Algèbre de Boole (B)

B 1 : $E \wedge Vrai = E$

B 2 : $E \wedge Faux = Faux$

B 3 : $E \vee Vrai = Vrai$

B 4 : $E \vee Faux = E$

B 5 : $E \wedge E = E$

B 6 : $E \vee E = E$

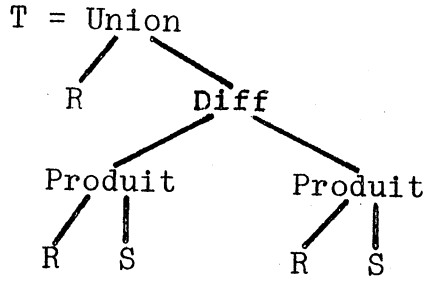
B 7 : $E \wedge (\neg E) = Faux$

B 8 : $E \vee (\neg E) = Vrai$

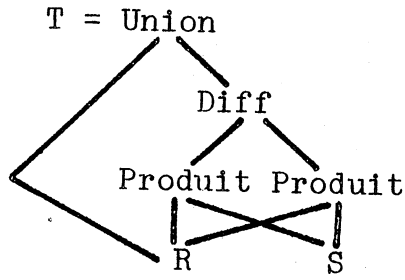
Il est clair que les règles (I) et (B) doivent être appliquées simultanément avec les réductions vues précédemment en 2.1.1 et 2.2.2. Pour appliquer (I), il faut reconnaitre des sous-arbres communs. HALL a exposé la méthode à utiliser [42] , nous ne don-

nerons qu'un exemple formel :

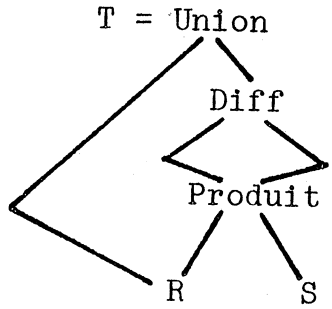
Exemple : Soit :



On supprime les feuilles inutiles, d'où :

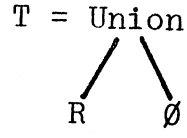


On voit qu'il y a un Produit inutile, d'où :



Il n'y a alors plus qu'à appliquer I4, puis R 5c.

On obtient successivement :



Puis finalement :

T = R

En conclusion, on voit que l'ensemble des réductions peut s'effectuer en parcourant l'arbre une fois, en remontant depuis les feuilles vers la racine et en suivant l'ordre :

- Reconnaissance des arbres R - homogènes
- Simplification des arbres R - homogènes et réduction des expressions inutiles (Tables : R, I, B)
- Reconnaissance des sous-arbres communs
- Simplification de ces sous-arbres (Tables : R, I, B).

Bien qu'effectué en un seul passage, l'ensemble des réductions peut être coûteux, surtout si l'on considère le problème des combinaisons d'expressions conditionnelles qui nécessitent de stocker des tables importantes, et de nombreux calculs.

Il convient toutefois de considérer que :

- ou bien il y a peu de réductions, et les calculs iront vite
- ou bien il y en aura beaucoup, et on peut s'attendre à un gain considérable en terme de transfert inter-opérateurs compensant largement le surcoût des calculs supplémentaires.

2.2 - DISTRIBUTION DES OPERATEURS UNAIRES

Les opérateurs PROJECTION et SELECTION se distribuent par rapport à la plupart des opérateurs : on peut les "descendre" dans l'arbre. Ces opérateurs ont une propriété intéressante : le volume de la relation résultante est toujours inférieur - et souvent très inférieur - au volume de la relation opérande.

Une importante et très efficace phase d'optimisation est donc la descente de ces deux opérateurs. Elle s'effectue en deux étapes : descente de l'opérateur SELECTION, puis descente de la PROJECTION. Il s'agit d'une transformation [1' : 1] qui ne nécessite pas d'évaluation des volumes des relations.

2.2.1 - DISTRIBUTION DE LA SELECTION

Soit E, E_1, E_2,...,E_n des expressions conditionnelles.

Rappelons qu'une <u>expression conditionnelle</u> est :

- soit une combinaison d'expressions conditionnelles, réalisée au moyen des opérateurs logiques NON ($\neg$), ET ($\wedge$), OU ($\vee$)

- soit une expression conditionnelle atomique.

Une <u>expression conditionnelle atomique</u> est une expression d'une des trois formes suivantes :

(1) $X_1 \ \theta \ a_1$

(2) $X_1 \in [a_1, b_1]$

(3) $X_1 \ \theta \ X_2$

où X_1, X_2 sont des attributs; a_1, a_2 des constantes et θ appartient à l'ensemble $\{=, \neq, <, \leq, >, \geq\}$

La table de distribution de l'opérateur SELECTION est donnée par la figure 3.

FIGURE 3 : TABLE DE DISTRIBUTION DE LA SELECTION (DS)

NOM DE LA REGLE	OPERATEUR	CONDITION	SOUS-ARBRE A TRANSFORMER	SOUS-ARBRE TRANSFORME
DS1	Sélection		$((R:E_1):\ldots):E_k$	$R:(E_1 \wedge \cdots \wedge E_k)$
DS2	Projection		$R[X]:E$	$(R:E)[X]$
DS3	Antiprojection	$LAT(E) \subset X$	$R]X[:E$	$(R:E)]X[$
DS4	Complément	Il existe $E', E_1, E_2, \ldots, E_n$ telles que: $E=(\bigwedge_{i=1}^{n} E_i)\wedge E'$ et : $\forall i \in \{1,2,..,n\}, E_i$ est soit vide (c.a.d. E_i = Vrai) soit telle que : $LAT(E_i)=X_i$, attribut de R	$\neg R :E$	$(\neg R: \bigwedge_{i=1}^{n} E_i):E'$
DS5	Union		$(R \cup S):E$	$(R:E)\cup(S:E)$
DS6	Intersection		$(R \cap S):E$	$(R:E)\cap(S:E)$
DS7	Différence		$(R-S):E$	$(R;E)-(S:E)$
DS8a	Produit Cartésien	Il existe E_R, E' telles que : $E = E_R \wedge E'$	$(RXS):E$	$((R:E_R)XS):E'$
DS8b	"	Condition symétrique	$(RXS):E$	$(RX(S:E_S)):E'$

NOM DE LA REGLE	OPERATEUR	CONDITION	SOUS-ARBRE A TRANSFORMER	SOUS-ARBRE TRANSFORME
DS9a	Somme	Il existe E_R, E' telles que : $E = E_R \wedge E'$ et: $E_R = \bigwedge_{i=1}^{n} E_i$) et : $\forall i \in \{1,n\}$, E_i est soit vide(c.a.d.E_i= Vrai) soit telle que : $LAT(E_i) = X_i$,attribut de R	(R+S):E	((R:E_R)+S):E'
DS9b	Somme	Condition symétrique	(R+S):E	(R+(S:E_S):E'
DS10a	Produit	Condition identique à celle de DS8a	(R*S):E	((R:E_R)*S):E'
DS10b	"	Condition symétrique	(R*S):E	(R*(S:E_S)):E'
DS11a	Θ-Produit	Condition identique à celle de DS8a	(R*($X_1 \Theta Y_1$) S):E	((R:E_R)*($X_1 \Theta Y_1$) S):E'
DS11b	"	Condition symétrique	(R*($X_1 \Theta Y_1$) S):E	(R*($X_1 \Theta Y_1$)(S:E_S)) :E'
DS12	Division	$LAT(\varepsilon) \subset X$	(R(X,Y)÷S (Y)):E	(R(X,Y):E)÷S(Y)

LAT (E) désigne l'ensemble des attributs intervenant dans l'expression conditionnelle E. Il est clair que dans la mesure où la requête est correcte, on a :

∀ R relation, ∀ E expression conditionnelle, telles que R : E, alors LAT (E) ⊂ LAT (R)

La notation E_R désigne une expression conditionnelle telle que LAT (E_R) ⊂ LAT (R)

$\bigwedge_{i=1}^{n} E_i$ est la notation abrégée de $E_1 \wedge E_2 \wedge \ldots \wedge E_n$

Les règles DS1, DS2, DS 5, DS6, DS7, D8 (a et b), DS 10 (a et b), DS 11 (a et b), DS 12 sont bien connues (voir notamment CALECA et DELOBEL-ADIBA [15][5]).

Pour distribuer la sélection par rapport au produit cartésien, on applique successivement DS 8 a et DS 8b. Dans le cas le plus favorable (E_R et E_S existent) on a :

(R X S) : E = ((R : E_R) X (S : E_S)) : E', avec E = $E_R \wedge E_S \wedge E'$

Eventuellement E' = Vrai, et on obtient finalement :

(R : E_R) X (S : E_S)

Ces remarques sont valables bien sûr pour D S 10 et D S 11.

Pour obtenir les autres règles, on utilise le fait que certains opérateurs peuvent s'exprimer en fonction des autres [5].

Ainsi l'antiprojection :

R (X,Y)]X[= R (X,Y) ÷ (R [Y])

D S 3 découle donc de D S 12.

DEMONSTRATION DE D S 4

Le complément se décompose ainsi :

$\neg R\ (X_1, X_2, \ldots, X_n) = (\ R[X_1] \quad X \quad R[X_2] \quad X \ldots X \quad R[X_n]\ ) - R$

d'où, d'après D S 7 :

$(\neg R) : E = (\ (\ R[X_1] \quad X \quad R[X_2] \quad X \ldots X \quad R[X_n]\ ) : E) - (R : E)$

Après D S 8 b, on obtient :

$$\Big(\ \big(\ R[X_1] \quad X \quad R[X_2] \quad X \ldots X \quad R[X_{n-1}] \quad X \quad (\ R[X_n] : E_n)\ \big) : E'_n\Big) - (R : E)$$

Sous réserve qu'il existe E_n, E'_n telles que $E = E_n \wedge E'_n$ et que $LAT(E_n) \subset LAT(R[X_n])$

$LAT(R[X_n]) = X_n$.

L'expression E_n est donc, soit vide (E_n = Vrai), soit définie sur le seul attribut X_n. (Par exemple : $X_n = 3$)

En réitérant le processus, on obtient :

$$\Big(\ \big(\ldots\big(\big(\ (\ R[X_1] : E_1)\ X\ (\ R[X_2] : E_2)\big) : E'_2\ X\ (\ R[X_3] : E_3)\big) : E'_3 \quad \text{---} \quad X\ (\ R[X_n] : E_n)\big) : E'_n\ \Big) - (R : E)$$

sous réserve que $\forall\, i \in [1, 2, \ldots, n]$, $LAT(E_i) = X_i$

En remontant $E'_2, E'_3, \ldots, E'_n$ dans un "reste commun" E' tel que $E' = E'_2 \wedge E'_3 \wedge \cdots \wedge E'_n$, on obtient :

$$\begin{array}{l} ((\quad (\quad R\,[X_1] \quad : E_1) \\ \quad X \quad (\quad R\,[X_2] \quad : E_2) \\ \qquad\qquad \text{----} \\ \quad X \quad (\quad R\,[X_n] \quad : E_n)) : E') \end{array}$$

- (R : E)

Après D S 2, on a :

$$\begin{array}{l} ((\quad (\quad (R : E_1)\,[X_1]) \\ \quad X \quad (\quad (R : E_2)\,[X_2]) \\ \qquad\qquad \text{----} \\ \quad X \quad (\quad (R : E_n)\,[X_n])) : E') \end{array}$$

- (R : E)

Ce qui est équivalent à :

$$\begin{array}{l} ((\quad (\quad (R : E_1) \; [X_1]) \\ \quad X \quad (\quad (R : E_2) \; [X_2]) \\ \qquad\qquad \text{----} \\ \quad X \quad (\quad (R : E_n) \; [X_n]))) \end{array}$$

$$- \; (R : (E_1 \wedge E_2 \wedge \cdots \wedge E_n) \quad) \quad : E'$$

D'où finalement :

$$((\neg R) : (E_1 \wedge E_2 \wedge \cdots \wedge E_n)) : E'$$

C.Q.F.D.

La somme s'exprime en fonction du complément et du produit :

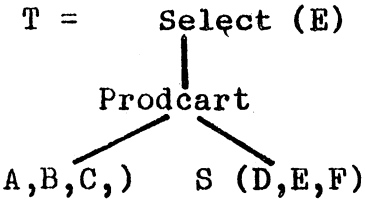
$$R + S = \neg((\neg R) * (\neg S))$$

En utilisant DS 4 et DS 10 a, on obtient D S 9a.

2.2.2 - FACTORISATION DES EXPRESSIONS CONDITIONNELLES ET EXEMPLE :

Pour appliquer les règles D S 4, D S 8, D S 9, D S 10, D S 11, il est nécessaire de factoriser des expressions conditionnelles. On trouvera dans CALECA, page 3-29 [15] un algorithme de factorisation qu'il n'est pas utile de rappeler ici.

Prenons un exemple, soit :

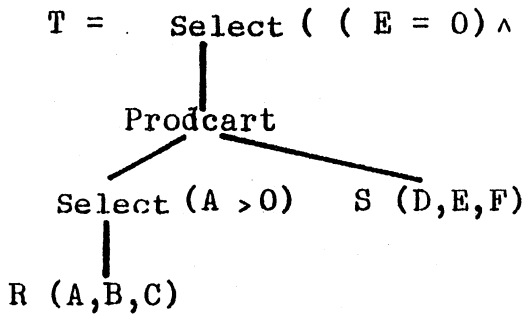


avec $E = ((A > 0 \wedge E = 0) \vee (A > 0 \wedge E = 0 \wedge B = 5)) \wedge C = D$

On effectue dans un premier temps la factorisation associée à DS 8 a, ce qui donne :

$$E = E_R \wedge E' \text{ avec } \begin{cases} E_R = (A > 0) \\ E' = (E = 0) \wedge (C = D) \end{cases}$$

On applique D S 8 a, d'où :

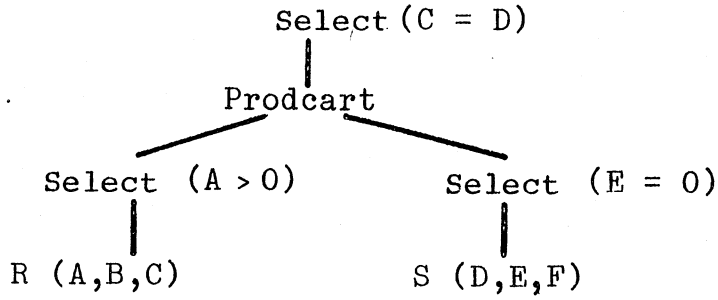


$E' = (E = 0) \wedge (C = D)$ est déjà factorisée :

$E' = E_S \wedge E''$ avec $E_S = (E = 0)$

$E'' = (C = D)$

D S 8 b est applicable, d'où finalement :



Mais avant d'effectuer la factorisation et les transformations il faut chercher à mettre en évidence des conditions implicites.

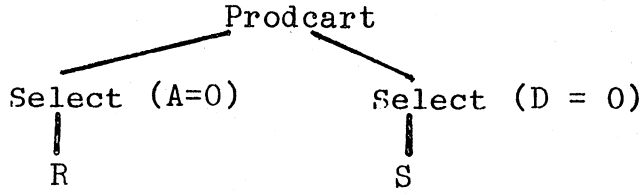
Si au lieu de l'expression E précédente, nous avions eu :

$E = (A = 0) \wedge (A = D)$

On voit qu'une telle expression est équivalente à :

$(A = 0) \wedge (D = 0)$, expression factorisée $E_R \wedge E_S$.

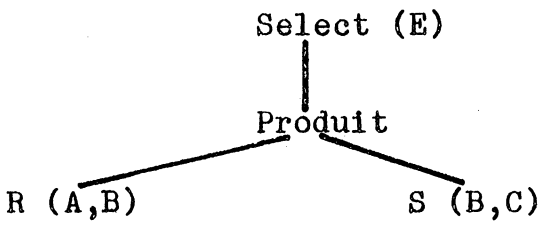
On peut alors appliquer les deux règles, la condition E se distribue totalement et on obtient :



Si au lieu d'un produit cartésien on a un produit, d'autres combinaisons sont possibles du fait que le produit admet par lui-même des conditions.

Il faut donc, préalablement à la factorisation d'une expression conditionnelle par rapport à un produit, mettre en évidence des conditions élémentaires de type $(X_1 = a_1)$ où X_1 est un attribut et a_1 une constante, ou de type $(X_1 = X_2)$ - implicites.

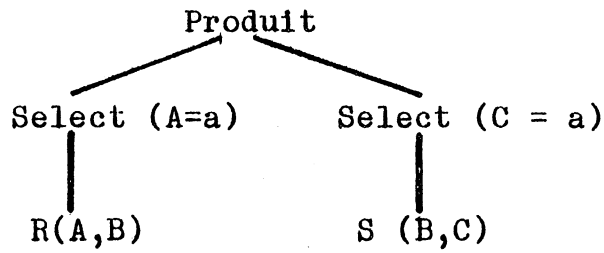
Exemple : Soit l'expression :



On voit que E est équivalent à E_1 $=(A = a) \wedge (A = C) \wedge (C = a)$

ou mieux : $E_2 = (A = a) \wedge (C = a)$

L'expression donnée est donc équivalente à :



De même, si au lieu de E, on avait eu :

$E' = (A = C) \wedge (B = C)$

Il aurait fallu remplacer E' par $E'_1 = (A = B) \wedge (B = C)$, expression équivalente qui permet de distribuer complètement la condition par rapport au produit.

Dans la pratique, il n'est pas évident de trouver l'expression équivalente "la plus intéressante ".

On se contentera d'appliquer les règles suivantes :

$$(A_1 = A_2) \wedge (A_1 = a) = (A_1 = a) \wedge (A_2 = a)$$

$$(A_1 = A_2) \wedge (A_2 = A_3) = (A_1 = A_2) \wedge (A_2 = A_3) \wedge (A_1 = A_3)$$

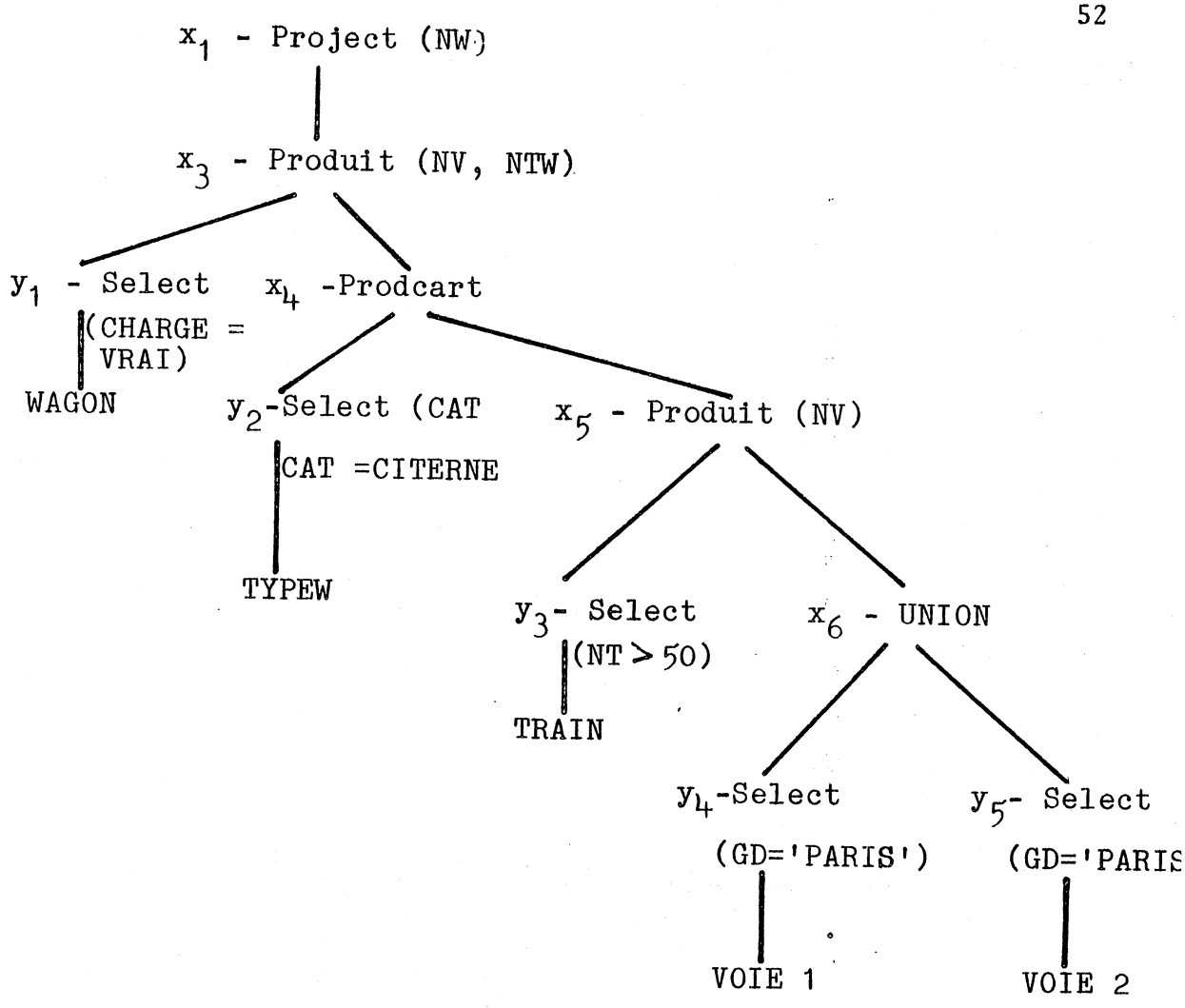


Figure : Requête obtenue après descente des sélections (DS)

Le total des volumes est de :

$VT_2 = 4,1. 10^6$

2.2.3 - DISTRIBUTION DE LA PROJECTION

Soit W_1, W_2,..., W_n des ensembles d'attributs.

La figure 4 donne la liste des distributions possibles de la projection par rapport aux autres opérateurs.

Pour DP1, la requête étant supposée correcte, on a :
$W_k \subset W_{k-1} \subset \ldots \subset W_1$

Pour D P 5,on s'est assuré au cours des phases précédentes que le produit R5 n'est ni une intersection, ni un produit cartésien.
Autrement dit :

$$(LAT(R) \cap LAT(S)) \neq \emptyset \text{ et } LAT(R) \neq LAT(S) .$$

Dans certains cas, on peut distribuer la projection par rapport aux opérateurs INTERSECTION et DIFFERENCE. [15] [38]
Dans la pratique, il n'est pas toujours possible de tester la condition sur les clés de D P 7 et D P 8, surtout si R et S ne sont pas des relations de bases.

Quant aux opérateurs ANTIPROJECTION, COMPLEMENT, SOMME, DIVISION, il n'existe pas de règles analogues.
En conclusion, la descente de la Projection est moins facile que celle de la Sélection, mais elle est moins coûteuse car il n'y a pas de problème de factorisation d'expressions.

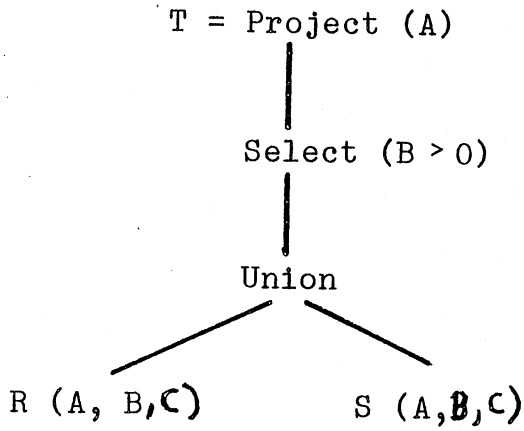
FIGURE 4 : TABLE DE DISTRIBUTION DE LA PROJECTION (DP)

NOM DE REGLE	OPERATEUR	CONDITION	SOUS-ARBRE A TRANSFORMER	SOUS-ARBRE TRANSFORME
DP1	Projection		$R[W1][W2] \ldots [Wn]$	$R[Wn]$
DP2a	Selection	$X \subset LAT(E)$	$(R:E)[X]$	$(R[X]):E$
DP2b	"	$X \not\subset LAT(E)$	$(R:E)[X]$	$((R[W_1]):E)[X]$ avec : $W_1 = X \cup LAT(E)$
DP3	Union		$(R \cup S)[X]$	$(R[X]) \cup (S[X])$
DP4	Produit Cartésien	$(LAT(R) \cap X \neq \emptyset) \wedge (LAT(S) \cap X \neq \emptyset)$	$(R \times S)[X]$	$(R[W_1]) \times (S[W_2])$ $W_1 = X \cap LAT(R)$ $W_2 = X \cap LAT(S)$
DP5a	Produit	$X = LAT(R) \cap LAT(S)$	$(R * S)[X]$	$(R[X]) \cap (S[X])$
DP5b	"	$(X \neq (LAT(R) \cap LAT(S))) \wedge (X \subset (LAT(R) \cap LAT(S)))$	$(R * S)[X]$	$(R[W_1]) * (S[W_2])$ avec : $W_1 = X \cap LAT(R)$ $W_2 = X \cap LAT(S)$
DP5c	"	$X \not\subset (LAT(R) \cap LAT(S))$	$(R * S)[X]$	$((R[W_1]) * (S[W_2]))[X]$
DP6a	θ-Produit	$X \subset (LAT(R) \cap LAT(S))$	$(R * (X_1 \theta Y_1) S)[X]$	$(R[W_1])(X_1 \theta Y_1)(S[W_2])$
DP6b	"	$X \not\subset (LAT(R)\ LAT(S))$	$(R * (X_1 \theta Y_1) S)[X]$	$((R[W_1]) * (X_1 \theta Y_1)(S[W_2]))[X]$

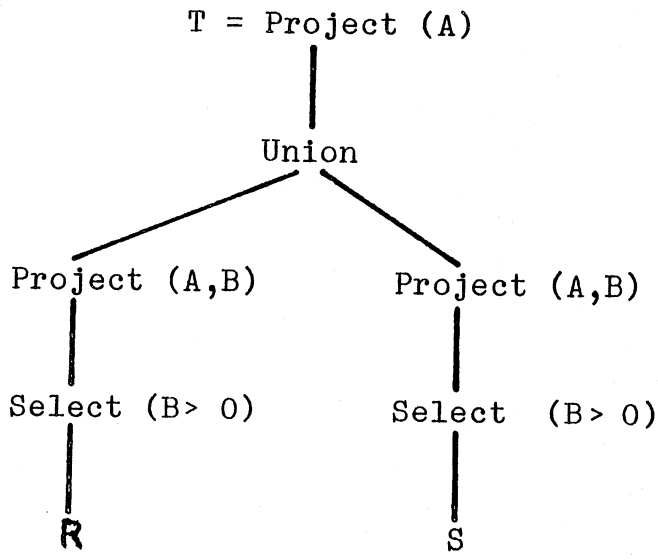
NOM DE REGLE	OPERATEUR	CONDITION	SOUS-ARBRE A TRANSFORMER	SOUS-ARBRE TRANSFORME
DP7a	Intersection	Il existe une clé W commune à R et S, telle que $W \subset X$	$(R\ S)[X]$	$(R[X]) \cap (S[X])$
DP7b	"	Même condition, mais $W \not\subset X$	$(R \cap S)[X]$	$((R[W_1]) \cap (S[W_1]))[X]$ $W_1 = X \cup W$
DP8a	Différence	Condition identique à celle de DP7a	$(R-S)[X]$	$(R[X])-(S[X])$
DP8b	"	Condition identique à celle de DP7b	$(R-S)[X]$	$((R[W_1])-(S[W_1]))[X]$ $W_1 = X \cup W$

La descente de la Projection doit s'effectuer après celle de la Sélection pour éviter qu'une Projection soit bloquée partiellement par une Sélection, ce qui aurait une incidence fâcheuse sur les branches inférieures de l'arbre.

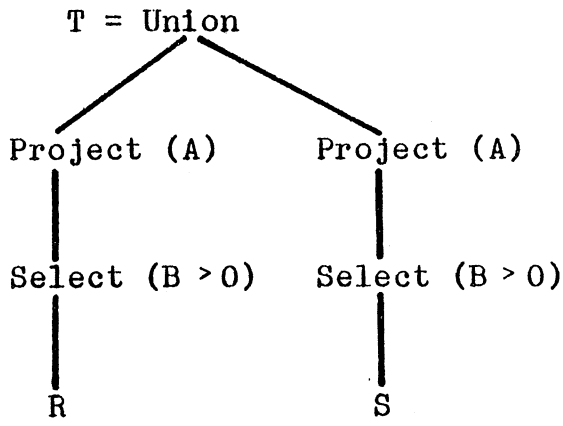
Exemple : Soit :



Si on effectue d'abord DP, puis ensuite DS, on obtient :

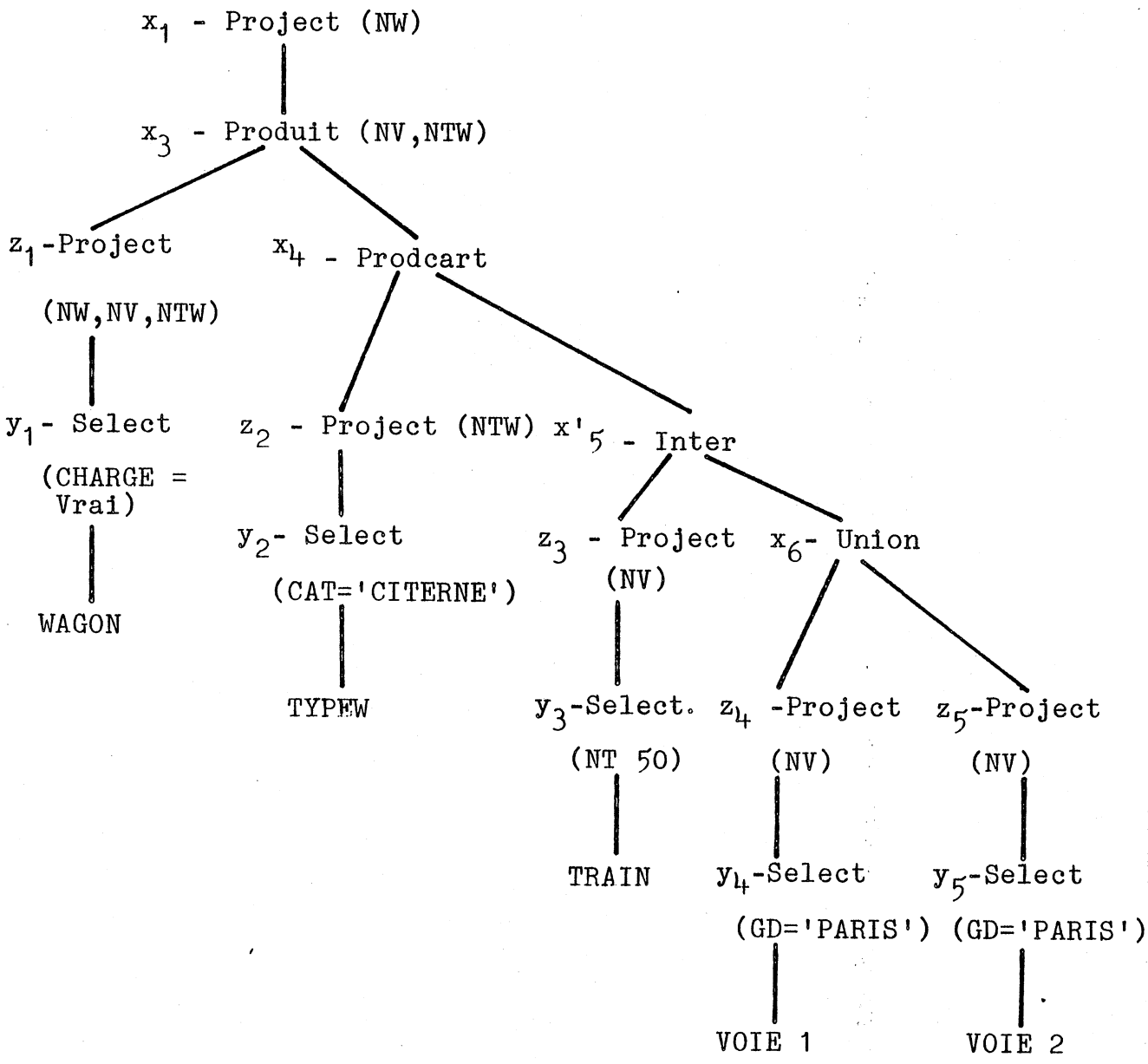


Si par contre, on applique D S puis D P, on obtient :



ce qui est nettement mieux.

FIGURE : Requête obtenue après descente des projections (DP)



Le total des volumes est de : $4,8.\ 10^6$

En réalité on peut considérer que la séquence sélection / projection constitue un seul opérateur, ce qui donne :

$VT_3 = 3,7.\ 10^6$

CHAPITRE 3 : LES AUTRES TRANSFORMATIONS ALGEBRIQUES

3.1 - PERMUTATION DES OPERATEURS BINAIRES

3.2 - LA DOUBLE DISTRIBUTIVITE

3.3 - CONCLUSION PROVISOIRE

3.3.1 - RECAPITULATIF

3.3.2 - LE PROBLEME DE LA REALISATION CONCRETE

3.1 - PERMUTATION DES OPERATEURS BINAIRES

Soit R, S, R_1, R_2, R_3, S_1, S_2,..., S_n, des relations (de bases ou non).

On a les règles suivantes pour les opérateurs binaires :

a) Commutativité

$R\ op_1\ S = S\ op_1 R$

b) Associativité

$(R_1\ op_1 R_2)\ op_1 R_3 = R_1\ op_1\ (R_2\ op_1\ R_3)$

et cela pour tout opérateur binaire op_1, à l'exception des opérateurs DIFFERENCE et DIVISION.

Si donc on a un sous-arbre composé de plusieurs occurences d'un même opérateur (UNION, INTERSECTION, PRODUIT CARTESIEN, PRODUIT, SOMME ou Θ- PRODUIT), on peut le réordonner comme on veut pour choisir l'ordre dans lequel on opère sur les relations de base.

Exemple 1 : Soit la "cascade" de produits :

$T_1 =$ Θ-Produit (B < C)

R_1(A,B) Θ-Produit (D ≠ E)

R_2 (C,D) R_3 (E,F)

On a :

$T_1 =$ Θ-Produit (E ≠ D)

R_3 Θ-Produit (C > B)

R_2 R_1

Dans la pratique, on rencontre rarement un sous-arbre composé avec un seul opérateur. On peut cependant remarquer que :

- Les différents produits (PRODUIT, Θ-PRODUIT, INTERSECTION, PRODUIT CARTESIEN) peuvent permuter les uns par rapport aux autres.

Ainsi :

$$R_1 * (R_2 \cap R_3) = (R_1 * R_2) * R_3$$

$$(R_1 * R_2) * (X_1 \Theta Y_1) \quad R_3 = R_1 * (X_1 \Theta Y_1) \quad (R_2 * R_3)$$

etc...

- Les différentes sommes (SOMME, UNION) peuvent également permuter les unes par rapport aux autres.

Finalement, il est souvent facile d'exhiber une cascade de produits (au sens large du terme).

Si la cascade de produits est une cascade d'intersections, une méthode simple et efficace est d'effectuer les produits par ordre croissant des Volumes des relations feuilles. [28][29]

Exemple 2 :

Soit : $T_2 = R_1 \cap (R_2 \cap (R_3 \cap R_4))$

Si $V(R_3) \subset V(R_2) \subset V(R_4) \subset V(R_1)$

alors, on choisira l'ordre suivant :

$T_2 = ((R_3 \cap R_2) \cap R_4) \cap R_1$

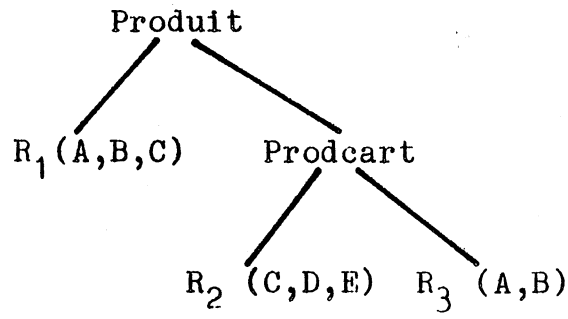
Il s'agit là d'un cas très particulier. En général, il faudra faire une évaluation des différentes permutations possibles, puis choisir la meilleure.

La permutation des opérateurs binaires est une transformation [1 : n] . Elle peut être très coûteuse : pour m produits, on a (m !) permutations possibles, et il faut ensuite évaluer les (m !) solutions candidates.

On élimine un certain nombre de possibilités en effectuant les produits de manière à éliminer les PRODUIT CARTESIEN, ou à les effectuer en dernier.

Exemple 3 :

Soit T_3 =



On a 6 possibilités :

$R_1 * (R_2 * R_3)$, $R_1 * (R_3 * R_2)$, $R_2 * (R_1 * R_3)$,

$R_2 * (R_3 * R_1)$, $R_3 * (R_1 * R_2)$ et $R_3 * (R_2 * R_1)$.

On peut déjà éliminer les deux premières : ($R_1 * (R_2 * R_3)$ et $R_1 * (R_3 * R_2)$) qui font effectuer un PRODUIT CARTESIEN d'entrée entre R_2 et R_3.

Entre les quatre solutions restantes, on peut remarquer qu'il est préférable de commencer par effectuer le produit $R_1 * R_3$ en premier, en effet, ce produit a pour domaine d'intersection deux attributs (A et B) au lieu d'un.

Les seules solutions à évaluer sont donc :

$R_2 * (R_1 * R_3)$ et $R_2 * (R_3 * R_1)$

Pour les SOMMES et les UNIONS, on peut appliquer des règles analogues.

En ce qui concerne les problèmes d'évaluation, nous renvoyons le lecteur à DELOBEL - ADIBA [5].

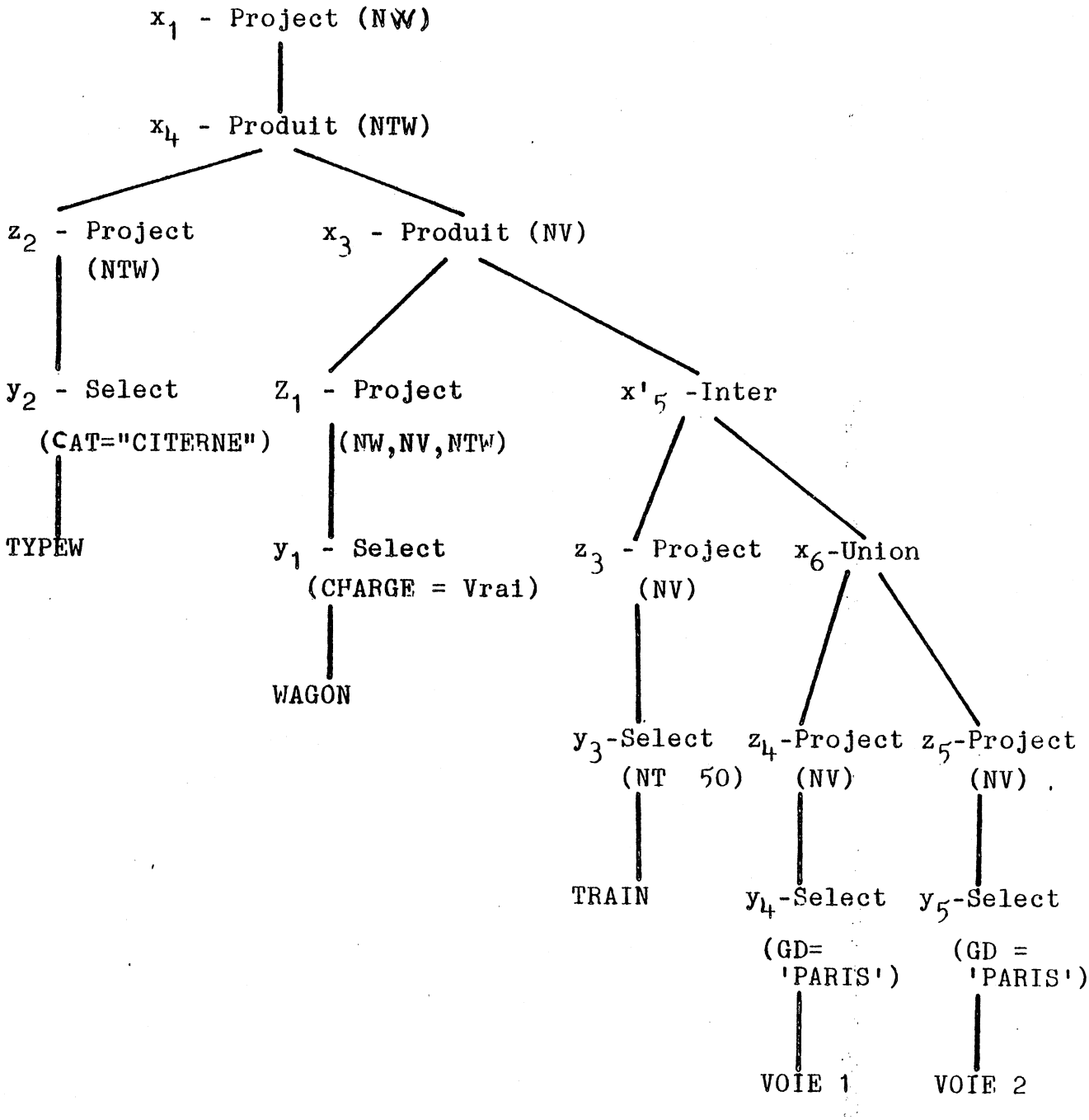


Figure : Requête définitive obtenue après permutation des opérateurs binaires.

Le total des volumes est de :

$$VT_4 = 3{,}6 \cdot 10^6$$

Gain total : $\frac{VT_1 - VT_4}{VT_1} = \frac{28 - 3{,}6}{28} \cdot 10^6 = 87\%$

3.2 - LA DOUBLE DISTRIBUTIVITE

L'UNION et l'INTERSECTION se distribuent l'une par rapport à l'autre.

Soit R, S_1, S_2,..., S_n des relations, on a :

$$R \cup (S_1 \cap S_2 \cap \ldots \cap S_n) = (R \cup S_1) \cap (R \cup S_2) \cap \ldots \cap (R \cup S_n)$$

$$R \cap (S_1 \cup S_2 \cup \ldots \cup S_n) = (R \cap S_1) \cup (R \cap S_2) \cup \ldots \cup (R \cap S_n)$$

Cette propriété s'étend à la SOMME et au produit :

$$R + (S_1 * S_2 * \ldots * S_n) = (R * S_1) + (R * S_2) + \ldots + (R * S_n)$$

$$R * (S_1 + S_2 + \ldots + S_n) = (R + S_1) * (R + S_2) * \ldots * (R + S_n)$$

La double distributivité est une transformation [1 : n] , elle est donc assez coûteuse. Elle ne permet pas, en général, de diminuer les transferts inter-opérateurs, bien au contraire. Elle sert à réduire le temps de réponse en améliorant le parallélisme d'éxécution - lorsqu'on dispose de plusieurs machines et que l'on souhaite distribuer les calculs.

Ce n'est donc pas une "optimisation" au sens où nous l'avons définie jusqu'à présent, mais elle entre dans le cadre des restructurations algébriques.

3.3 - CONCLUSION PROVISOIRE

3.3.1 - RECAPITULATIF

Les phases successives de l'optimisation d'une requête relationnelle localisée sont les suivantes :

1) L'Initialisation, c'est-à-dire le calcul des caractéristiques des relations intermédiaires(en particulier la liste LAT de leurs attributs), plus éventuellement un contrôle de la cohérence de la requête.

2) La réduction des expressions vides ou inutiles. Cette phase comprend trois transformations simultanées :
   - la réduction des arbres R - homogènes (cf chapitre 2.1.1)
   - l'élimination des expressions vides et la réduction - Table R (cf 2.1.2)
   - la recherche de sous-arbres communs et leur réduction par les règles d'idempotence (cf 2.1.3)

3) La distribution de l'opérateur SELECTION par la table DS (cf 2.2.1 et 2.2.2)

4) La distribution de l'opérateur PROJECTION par la table DP (cf 2.2.3)

5) Une pré-évaluation du coût de la requête, dont le but est d'estimer s'il est utile de continuer l'optimisation et d'appliquer les autres phases.

6) (Eventuellement) - Application des double-distributions pour accroitre le parallélisme d'éxécution (cf 3.2)

7) La permutation des opérateurs binaires (cf 3.1)

8) L'évaluation des solutions possibles parmi les graphes obtenus après les phases 6 et 7, et le choix de la solution optimale.

Commence ensuite la partie allocation de données et allocation de ressources, qui sort du cadre des restructurations algébriques
- Voir PAIK In-Sup [57]

Les phases 1 à 4 ne génèrent qu'un seul arbre. Elles s'effectuent chacune en un seul passage. Cet ensemble constitue le noyau minimum des restructurations algébriques nécessaires.
Les phases 6 et 7, optionnelles, génèrent plusieurs arbres.
Partant du fait que cinq opérateurs suffisent à décrire tous les autres, il a été proposé, pour les phases 3 et 4 en particulier, de décomposer les requêtes en fonction de ces opérateurs "basiques" (cf [5], chap. 8·3).

Nous avons écarté cette solution qui parait comporter plus d'inconvénients que d'avantages. Certes, on simplifie ainsi les table DS et DP, mais en retour, on rajoute deux phases supplémentaires la décomposition de l'arbre à l'aide des opérateurs basiques, la "recomposition" une fois effectuées les optimisations. En plus, on augmente considérablement la taille de l'arbre, et donc le nombre de noeuds à tester.

3.3.2 - LE PROBLEME DE LA REALISATION CONCRETE

Les différentes méthodes que nous avons vues ont été partiellement implémentées dans plusieurs systèmes. Toutes sont connues, et il y a une importante littérature sur la question. Cependant, aucun système basé sur l'Algèbre Relationnelle n'utilise l'ensemble de ces restructurations algébriques, qui sont pourtant toutes utiles à priori.
Cette contradiction illustre bien la complexité du problème et prouve que la question de l'optimisation des systèmes relationnels est loin d'être réglée.
S'il est certain que l'on peut placer des espoirs légitimes dans la mise en oeuvre de techniques "hard" nouvelles pour rendre efficace le Relationnel, ces techniques ne dispenseront jamais les concepteurs de la nécessité d'une optimisation "soft" extrêmement soignée.
Le découpage en différentes phases bien définies, l'utilisation de tables de transformation, la structure arborescente - tout cela impose le parallèle avec d'autres domaines de l'Informatique, en particulier l'Intelligence Artificielle.
Implémenter les huit phases que nous avons vues (sans parler du reste...), c'est un travail fastidieux étant donné le nombre considérable de règles à écrire , ce qui signifie autant de procédures, avec leurs fonctions auxilliaires...
Les Systèmes Généraux de Transformations Paramétrées(SGTP) fournissent, ainsi que leur nom l'indique, un outil bien adapté aux restructurations algébriques, sur le plan théorique. [48]
Restait à prouver leur efficacité pratique, c'est là l'objet de la deuxième partie de notre thèse.

PARTIE II

APPLICATION DES SYSTEMES DE TRANSFORMATION D'ARBORESCENCES

INTRODUCTION

CHAPITRE 1 : LES DEFINITIONS DE BASE ET LEUR REALISATION

CHAPITRE 2 : LES SYSTEMES GENERAUX DE TRANSFORMATIONS PARAMETREES

CONCLUSION GENERALE

INTRODUCTION A LA DEUXIEME PARTIE :

LE SYSTEME PIAF : PROGRAMMES INTERACTIFS POUR L'ANALYSE DU FRANCAIS

Ce système a été développé au sein de l'équipe Intelligence Artificielle de l'IMAG.

Il a pour but l'analyse du Français à tous les niveaux. Ses algorithmes sont paramétrés et fonctionnent en mode interactif.

Le système PIAF est formé d'un ensemble de modules indépendants :

1) Le module de traitement de chaîne qui segmente les chaînes (Données) et détermine un certain nombre de renseignements en fonction d'un dictionnaire, d'une liste de modèles, d'une grammaire à validation et saturation équivalente à une grammaire d'états finis.

2) L'analyseur de dépendance : le module de dépendance a pour but la construction de structures-types (structures de dépendance) de la phrase à l'aide de règles linguistiques. Ainsi, dans la phrase "LE PILOTE FERME LA PORTE", le groupe verbal "FERME" commande le groupe nominal "LE PILOTE" et le groupe complément "LA PORTE".

3) Le module de contrôle de variables : ce module vérifie que les chaînes d'entrée analysées par le 1er module sont conformes aux règles définies dans le 2e module.

4) Le module de reconnaissance et de transformation de ramifications. Etant donné une structure de dépendance d'une phrase (module n° 2), on veut reconnaitre une sous-ramification, et éventuellement la transformer.

5) Le module d'évaluation sémantique : il s'agit de reconnaitr dans la structure syntaxique d'une phrase donnée la fonction des éléments de la phrase par rapport à un contexte défini.

En ce qui nous concerne, c'est le 4ième module qui nous intéresse. Ce module a été développé par Julio LOPEZ-MEDINA([48]) sur un IBM 3(il est écrit en LISP. Nous l'avons testé en écrivant une table de distribution de la projection et de la sélection. Il est vite appari que le temps nécessaire à effectuer ces optimisations était élevé, trop élevé : sous CP/CMS interactif, plus de 1 minute pour restructurer une requête comportant six opérateurs relationnels. Ceci n'est pas surprenant : il s'agit d'un système très général, où on ne connait pas à priori les structures qu'on analyse.
Cependant, il a été montré que ce système permettait parfaitement d'effectuer les restructurations décrites en première partie.
Nous avons décidé alors de réécrire un système analogue, optimisé, et adapté à la manipulation des opérateurs relationnels, et de l'implémenter sur micro-ordinateur.
Nous allons d'abord développer, sans entrer dans le détail des justifications théoriques (cf V. JOLOBOFF [44] et J. LOPEZ [48]) les notions de base nécessaires, et leur traduction en terme d'algèbre relationnelle. Puis, nous donnerons les principaux algorithmes en justifiant les modifications et simplifications. Enfin, nous définirons les applications du système à l'optimisation des requêtes relationnelles, en distinguant ce qui a été fait et ce qui reste à faire.

CHAPITRE 1 : LES DEFINITIONS DE BASE ET LEUR REALISATION

1-1 - LES RAMIFICATIONS

1.1.1 - DEFINITIONS

Soit X un ensemble de sommets.

Pour distinguer ces sommets, on associe à chaque sommet de X une étiquette, c'est-à-dire un élément du vocabulaire V.

$$E : X \to V$$

V est un ensemble fini de chaînes de caractères alphanumériques.

Dans un premier temps, nous prendrons V = {noms d'opérateurs relationnels} ∪ {noms de relations}

U est une relation binaire sur X^2. On dira que xUy si le couple (x,y) appartient à U. x est le sommet antécédant, ou "père", et y le sommet successeur (ou "fils" ou "descendant immédiat"). On note U (x) l'ensemble des descendants immédiats de x.

(X,U) est un graphe.

Ce graphe est une arborescence s'il est connexe, sans cycle, et si :

i) Pour tout $x \in X$, il existe au plus un $y \in X$ tel que yUx et :

ii) Il existe un sommet $x_o \in X$, tel que pour tout $y \in X$, $y \not\mathcal{U} x_o$. Ce sommet x_o est appelé racine de l'arborescence.

(X,U,O) est une arborescence orientée si :

(X,U) est une arborescence

et (X,O) est un ordre partiel tel que :

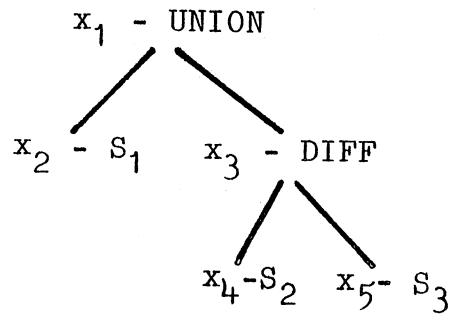
i) les restrictions de O à chaque sous-ensemble U(x), $x \in X$, sont des ordres totaux

ii) et pour tout $x,y,z \in X$,

xUy et $(x \not\mathcal{U} z) \Longrightarrow$ y et z non comparables.

Nous avons ainsi défini l'arborescence orientée étiquetée sur un vocabulaire V: (X,U,O,E)

Exemple :



$X = (x_1, x_2, x_3, x_4, x_5)$

$U = \{(x_1, x_2), (x_1, x_3), (x_3, x_4), (x_3, x_5)\}$

E : X $\longrightarrow$ V

$x_1 \dashrightarrow$ UNION

$x_2 \dashrightarrow S_1$

etc....

O est un ordre sur les sommets "frères".

Ainsi : $x_2 O x_3$, $x_4 O x_5$

Les relations opérandes gauche et droite sont ainsi distinguées.

Deux arborescences orientées étiquetées (X,U,O,E) et (X', U',O',E') définies sur un même vocabulaire V sont équivalentes s'il existe une bijection f entre X et X' telle que :

i) $x,y \in X$ $\quad xUy \Rightarrow f(x)\ Uf(y)$

ii) $x,y \in X$ $\quad xOy \Rightarrow f(x)\ Of(y)$

iii) $x \in X$ $\quad E(y) = E'(f(x))$

Exemple :

On définit ainsi une pseudo-arborescence sur V comme classe d'équivalence pour ~

Soit V! l'ensemble de toutes les classes sur V, c'est-à-dire toutes les pseudo-arborescences possibles.

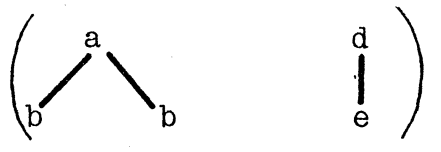
Une ramification V est une suite finie ordonnée de pseudo-arborescences sur V : $\hat{V} \subset V!$

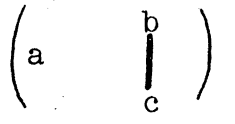
Pour nous, une pseudo-arborescence sera un arbre d'opérateurs relationnels et de relations (requête, ou partie de requête), et une ramification, un ensemble d'arbres.

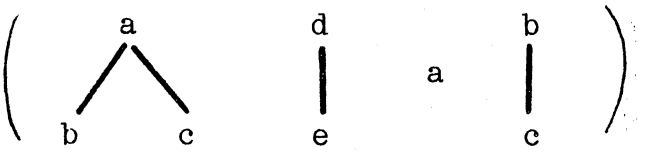
1.1.2 - OPERATIONS SUR LES RAMIFICATIONS

- La concaténation de ramifications : $R_1 + R_2$

Elément neutre : la ramification vide $\emptyset$

R_1 = 

R_2 = 

$R_1 + R_2$ = 

Pour simplifier, nous n'avons pas numéroté les sommets.

Remarque : une ramification est la concaténation de plusieurs pseudo-arborescences, lesquelles sont des cas particuliers de ramifications réduites à un seul élément (un seul arbre).

- L'enracinement d'un élément a (non vide) de V sur une ramification R : a X R

Cette opération permet de transformer une ramification en pseudo-arborescence en créant une racine commune à toutes les pseudo-arborescences de la ramification.

Exemple :

Soit

$$a = \text{UNION} \qquad R = \left(\begin{array}{c} x_1 - \text{UNION} \\ x_2 - S_1 \quad x_3 - S_2 \quad x_4 - S_3 \end{array} \right)$$

Alors

$$a \; X \; R = \left(\begin{array}{c} x_o - \text{UNION} \\ x_1 - \text{UNION} \quad x_4 - S_3 \\ x_2 - S_1 \quad x_3 - S_2 \end{array} \right)$$

On convient que : $a \; X \; \emptyset = a$

On peut montrer que, réciproquement :

$\forall R \in V$, $R \neq \emptyset$, il existe $a \in V$, R', $R'' \in V$ tels que

$R = (a \; X \; R') + R''$

R' et R" peuvent être vides.

Une sous-ramification $R' = (X', U', O', E')$ d'une ramification $R = (X, U, O, E)$ est une ramification telle que :

$X' \subset X$

E' est la restriction de E à X'

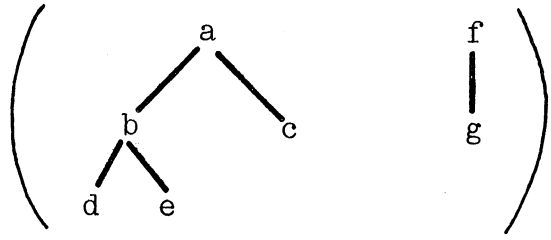
$\forall x, y \in X'$, $xU'y \iff xUy$ et $xO'_S y \iff xO_S y$

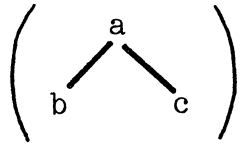
où O_S signifie "successeur immédiat" dans O.

R' est complète si $\forall x \in X'$, $x \; U \; y \Rightarrow y \in X'$

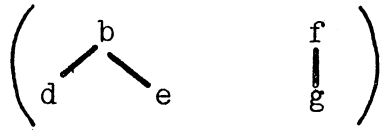
On note $R' \subset R$

Exemple :

R = 

R'_1 = 

est une sous-ramification non-complète de R.

R'_2 = 

est une sous-ramification complète de R

Nous allons maintenant définir un système de représentation plus général.

1.2 - LES RAMIFICATIONS PARAMETREES

1.2.1 - LES PARAMETRES

Chaque sommet d'une ramification paramétrée est un paramètre. Il peut représenter un ensemble d'étiquettes, ou un ensemble de ramifications.

Plus précisément, un paramètre $p = (p_1, p_2)$ est un couple (nom, pred) où nom est une chaîne de caractères alphanumériques (identificateur); et pred un prédicat logique ayant pour domaine un ensemble d'éléments d'étiquettes V_E.

L'évaluation de p_2 sur une étiquette a est notée p_2 (a).

Exemple :

On peut définir NOM-OPERATEUR, comme étant un prédicat ayant pour seule valeur possible θ - PRODUIT, ce qu'on écrit :

(NOM - OPERATEUR (θ - PRODUIT)) $\in V_E$

De même, on peut définir le prédicat TETAT en écrivant :

(TETA ('=', '≠', '<', '≤', '>', '≥') $\in V_E$

Nous pouvons alors donner un exemple de paramètre :

p = (x_1 , (NOM-OPERATEUR = (θ - PRODUIT)) et(TETA = ('< ')))

où : $p_1 = x_1$

P_2 = (NOM - OPERATEUR = (θ - PRODUIT)) et (TETA = ('< '))

L'évaluation de p_2 sur l'étiquette TETA, p_2 (TETA), est égale à : ('<').

Le prédicat pred est en fait une suite de prédicats élémentaires :

$$\text{pred} = \bigwedge_{i=1}^{n} pe_i$$

où chaque pe_i est de la forme :

<Identificateur> (< liste d'identificateurs >).

Dans l'exemple ci-dessus, on a deux prédicats élémentaires NOM-OPERATEUR et TETA (1).

(1) Il s'agit d'exemples formels. La définition précise des prédicats et la modélisation des opérateurs est donnée en 1.4.2 et 1.4.3

1.2.2 - NOM DE PARAMETRE

Un nom de paramètre est un élément d'un des ensembles disjoints suivants :

NR = {noms de ramifications}

NS = {noms de pseudo-arborescences}

NE = {noms d'étiquettes}

Un nom de paramètre est optionnel s'il peut éventuellement représenter l'élément vide. Il est obligatoire dans le cas contraire. Chaque ensemble (NR, NS, NE) se subdivise en ensemble de noms de paramètres obligatoires (NR_1, NS_1, NE_1) et ensemble de noms de paramètres optionnels (NR_2, NS_2, NE_2).

En conclusion, une ramification paramétrée R est un élément de $(P \tilde{X} \mathcal{P}_{V_E})$ où P est un ensemble de noms de paramètres et $\mathcal{P}_{V_E}$ un ensemble de prédicats sur V_E.

On impose que tout paramètre ramification ou pseudo-ramification soit une feuille de la ramification. C'est là que réside la principale limitation théorique du système.

1.2.3 - EQUIVALENCE ENTRE RAMIFICATIONS

Soit RP l'ensemble des ramifications paramétrées.
Grâce aux paramètres, toute ramification paramétrée X peut représenter une <u>famille de ramifications</u> F_X , définie par une <u>relation d'équivalence</u> entre ramifications sur V et ramifications paramétrées.

<u>Exemple</u>

Soit la ramification paramétrée définie sur le vocabulaire V

$$X = \left(\begin{array}{ccc} x_1 & - & \text{UNION} \\ & \diagup \quad \diagdown & \\ x_2 & & x_3 \end{array} \right) \quad \text{avec } x_1 \in NE_1 ; \; x_2, x_3 \in NS_1$$

Alors les deux ramifications suivantes sont équivalentes à X :

$$X_1 = \left(\begin{array}{ccc} & \text{UNION} & \\ & \diagup \quad \diagdown & \\ R_1 & & R_2 \end{array} \right) \qquad X_2 = \left(\begin{array}{ccc} & \text{UNION} & \\ & \diagup \quad \diagdown & \\ R_1 & & \text{INTER} \\ & & \diagup \quad \diagdown \\ & & R_2 \quad R_3 \end{array} \right)$$

A l'aide d'une ramification paramétrée X et d'un vocabulaire V, on peut représenter ainsi une infinité de ramifications équivalentes. *C'est là l'intérêt de la notion de paramètre.*
On vérifie l'équivalence en évaluant la valeur prise par chaque paramètre $p = (p_1, p_2)$ de X :

$$\underset{X = X_1}{VAL}\ (x_1) = \underset{X = X_2}{VAL}\ (x_1) = UNION$$

$$\underset{X = X_1}{VAL}\ (x_2) = \underset{X = X_2}{VAL}\ (x_2) = R_1$$

$$\underset{X = X_1}{VAL}\ (x_3) = R_2$$

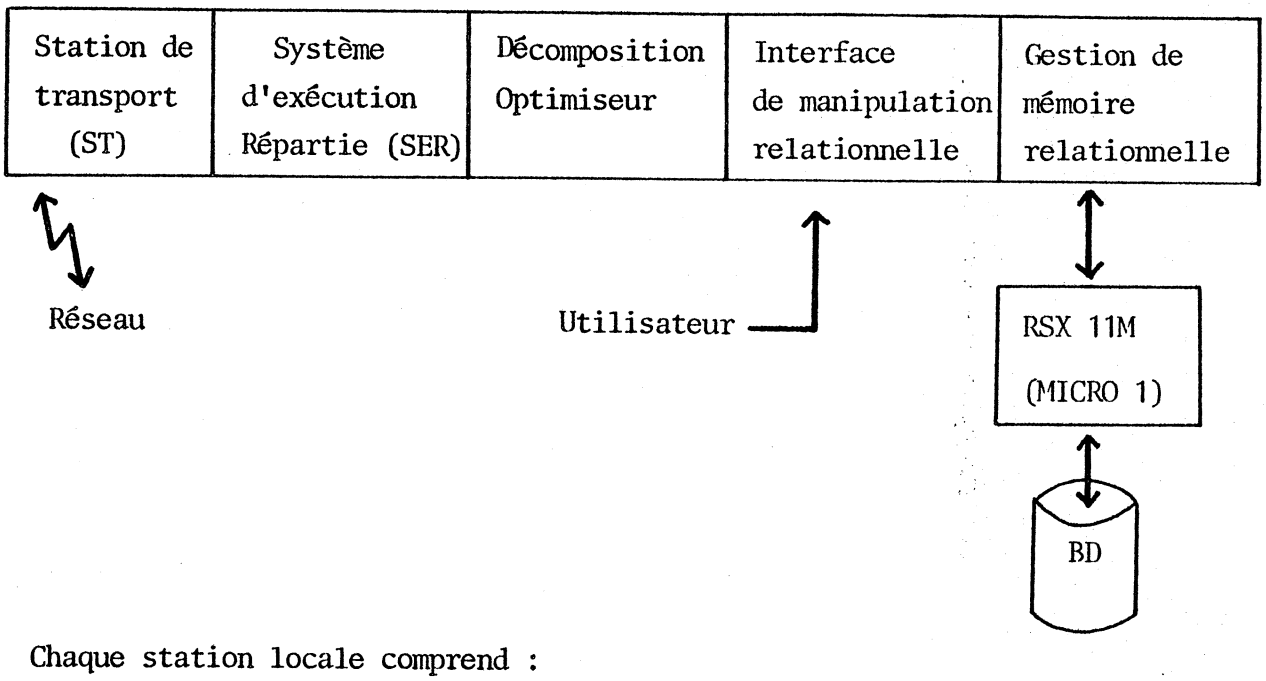
$$\underset{X = X_2}{VAL}\ (x_3) = \underset{R_2 \quad R_3}{INTER}$$

1 - 3. PRÉSENTATION DE MICROBE

Un ensemble de bâtiments du campus grenoblois est relié par un réseau local à diffusion. Il se compose de calculateurs PLESSEY MICRO1, MICRO2 - d'un mini-ordinateur SOLAR 16-65 et de l'ordinateur central HB 68 du Centre Scientifique de Calcul de Grenoble (CICG). D'autres connexions sont prévues.

Le projet MICROBE consiste en la conception et la réalisation d'un système de Gestion de Bases de Données Réparties sur les stations du réseau local. Il fait suite aux études menées à l'Institut de Mathématiques Appliquées de Grenoble dans le cadre du SGBD Réparti POLYPHEME [1][2][15][53][54][58].

La structure générale de MICROBE peut se représenter ainsi :

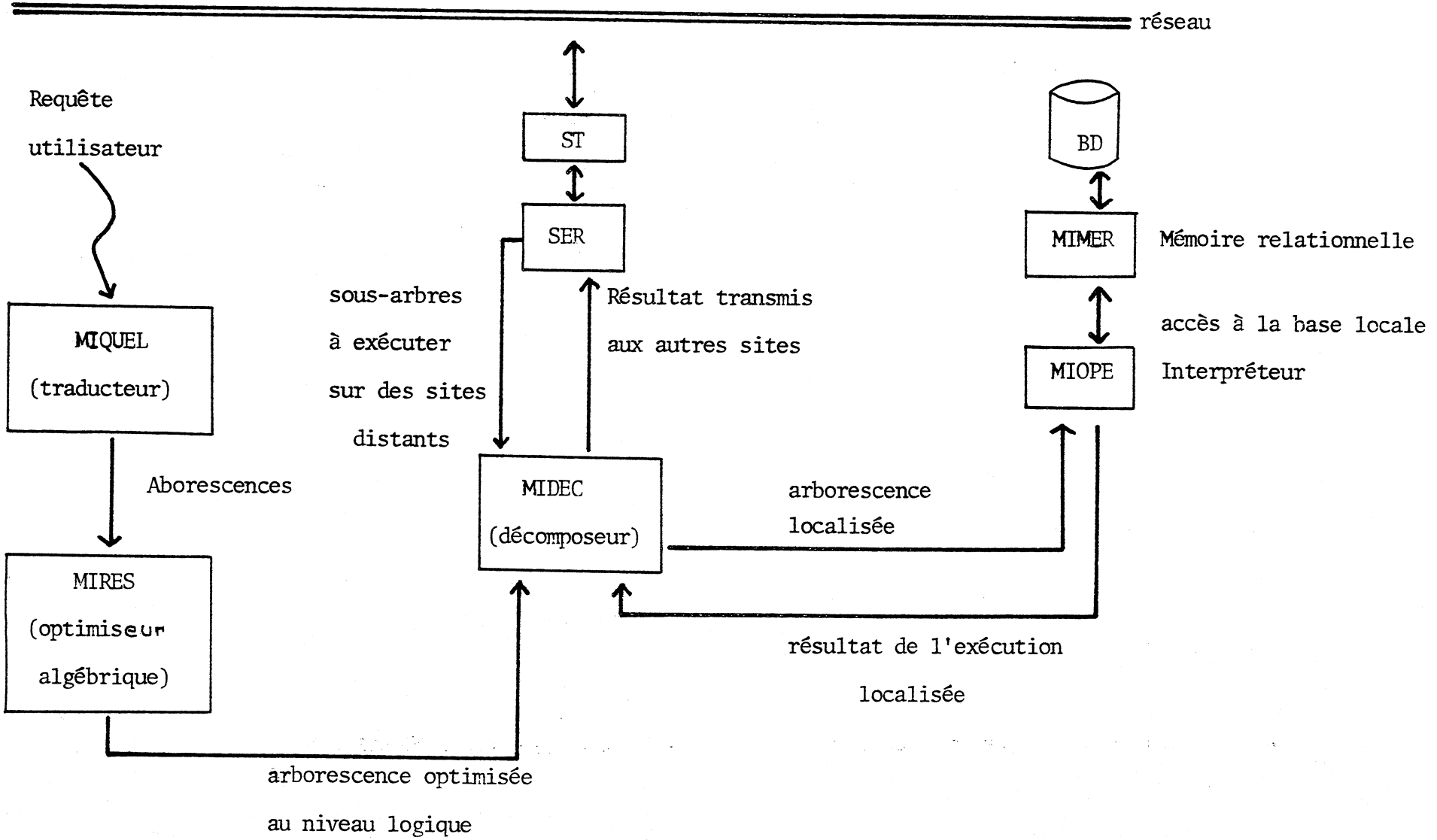


Chaque station locale comprend :

- Un interface de Manipulation relationnelle composé d'un traducteur de langage relationnel externe en des arborescences algébriques de type URANUS.

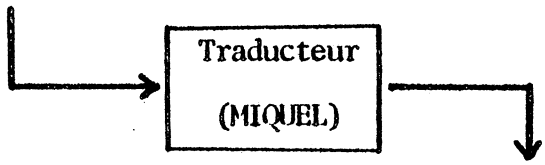
- Un décomposeur chargé de localiser les sous-arborescences, de façon dynamique
- Un optimiseur
- Un système d'exécution répartie accédant aux autres composants réseau du SGBDR via la ST
- Une mémoire relationnel (MIMER) assurant l'interface entre la vision logique relationnelle et le stockage physique des données.

Le SGBD MICROBE réparti a la structure suivante :


réseau
Requête
utilisateur
ST
BD
SER
MIMER
Mémoire relationnelle
sous-arbres
à exécuter
sur des sites
distants
Résultat transmis
aux autres sites
accès à la base locale
MIQUEL
(traducteur)
MIOPE
Interpréteur
Aborescences
MIDEC
(décomposeur)
arborescence
localisée
MIRES
(optimiseur
algébrique)
résultat de l'exécution
localisée
arborescence optimisée
au niveau logique

La requête utilisateur est exprimée dans un langage de type SEQUEL.

Requête utilisateur type SEQUEL



arborescence type URANUS

Il est certain que les langages basés sur l'algèbre des opérateurs relationnels sont inutilisables par des utilisateurs non spécialistes. La solution Langage(s) externe(s) de haut niveau/traducteur/langage de bas niveau est la solution qui semble devoir s'imposer. Il est en effet utile de pouvoir disposer de plusieurs langages externes de haut niveau au niveau utilisateur, de la même manière qu'il est utile qu'un système offre plusieurs langages de programmation : sans parler des recherches sur les "langages naturels", les traitement graphiques, etc... il est clair que la perspective du "langage Universel" qui permettrait de tout faire a vécu...

La couche URANUS restant inchangée, les optimisations effectuées à ce niveau resteront valables : en d'autre terme, il ne sera pas utile de réécrire l'optimiseur pour développer un nouveau langage, pourvu que l'optimiseur soit suffisamment complet au départ.

Actuellement, l'optimiseur MIRES écrit par Melle WINNINNGER effectue certaines restructurations, limitées, compte-tenu de l'état de réalisation et du contexte de travail de MICROBE. De nombreuses simplifications ont été faites, les Bases de Données prévues pour MICROBE étant petites.(Il n'y a pas de gestion de statistiques par exemple) [62].

Si on envisage des machines plus puissantes, des bases plus grosses, le problème doit être abordé de manière plus systématique. Par ailleurs, l'aspect multi-utilisateurs doit être pris en compte d'autant plus que la puissance des micros augmente très rapidement.

1.4 - IMPLEMENTATION

1.4.1 - LES ARBRES MICROBE

Pour comprendre l'implémentation des arborescences (paramétrées ou non) traduites en terme d'algèbre relationnelle, il faut décrire les structures manipulées.
Bien que le système de transformation ne soit pas couplé à MICROBE, il travaille sur des requêtes de type MICROBE, en particulier le tableau ARBRE, tel qu'il est rempli une fois exécutée la traduction MICROSEQUEL/URANUS. (cf manuel MICROBE [34]).
Le point d'entrée est ARBRE [1], c'est-à-dire la racine de l'arborescence.
Les structures PASCAL utilisées sont les suivantes :

Les Constantes :

LCODE = 100 : nombre d'éléments maximum du tableau ARBRE
NBCONST = 15 : nombre d'éléments maximum du tableau TCONST
NBREL = 10 : nombre d'éléments maximum du tableau TRELAT
NBDOM : 25 : nombre d'éléments maximum du tableau TDOM.

Les Types :

```
TEXTE       =   array [1..LTAMP] of char;
NOEUD       =   record
                  TTYPE  : char;
                  TNUM, TFILS, TFRERE, TIND : integrer;
                end;
RELENREG    = record
                  IDENTREL, ( * Nom de la relation * )
                  IDENTVAR : texte;
                end;
```

```
DOMENREG    = record
                  IDENTDOM, (* Nom du domaine ou attribut *)
                  IDENTREL, (* Relation à laquelle il est associé
                  IDENTVAR : TEXTE;
              end;
```

Les Tableaux

```
ARBRE      : array  [ 1...LCODE ]  of NOEUD;
TRELAT     : array  [ 1...NBREL ]  of RELENREG;
TDOM       : array  [ 1...NBDOM ]  of DOMENREG;
TCONST     : array  [ 1...NBCONST] of TEXTE;
```

La requête proprement dite est stockée dans le tableau ARBRE.

Le champ TTYPE, de 1 caractère, contient le type du noeud selon le code suivant :

'O' : Le noeud est un opérateur algébrique (PRODUIT, SELECT,... Le champ TNUM est l'indice de l'opérateur dans un tableau TABOP (voir plus loin).

'R' : Le noeud est une relation dont l'indice dans TRELAT est TNUM.

'K' : Constante, d'indice TNUM dans TCONST.

'D' : Domaine, d'indice TNUM dans TDOM.

'C' : Connecteur (=, <,and, or,...), d'indice TNUM dans TABOP.

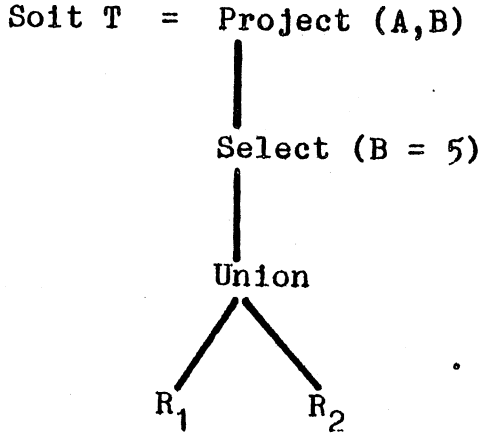
etc...

Le tableau TABOP, compatible avec URANUS [52] est un tableau de 60 variables TEXTE qui permet de coder les mots réservés du système.

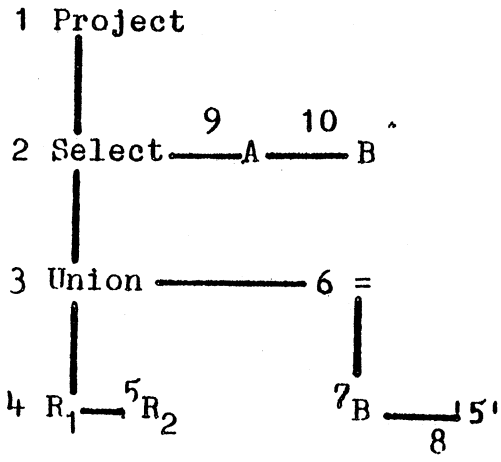
On y trouve les différents noms des opérateurs relationnels MICROBE; les opérateurs de comparaison (TTYPE = 'C'), les opérateurs logiques (TTYPE = 'C' également), les noms des commandes (CREER - BASE, TUER - RELATION , etc...).

Les champs TFILS et TFRERE chaînent les noeuds selon le modèle préfixé URANUS.

Exemple :



L'arbre URANUS est :



On obtiendra quelque chose du genre :

n° de noeud	TTYPE	TNUM	TFILS	TFRERE	SIGNIFICATION
1	0	28	2	0	PROJECT
2	0	27	3	9	SELECT
3	0	32	4	6	UNION
4	R	1	0	5	R1
5	R	2	0	0	R2
6	C	8	7	0	=
7	D	2	0	8	B
8	K	1	0	0	5
9	D	1	0	10	A
10	D	2	0	0	B

Une telle structure pose un réel problème en ce sens que des expressions de nature totalement différentes sont mises au même niveau. Un "noeud" peut donc être aussi bien une relation, un opérateur, une partie d'expression conditionnelle ou... n'importe quoi.

Pour l'optimisation, il est nécessaire de dissocier strictement

- les "noeuds" proprement dit : opérateurs, relations.
- les expressions autres, notamment les expressions conditionnelles et les listes d'attributs.

Comme il n'était pas question de toucher à la structure MICROBE puisque notre but était de prouver la faisabilité de notre système, nous avons dû multiplier les contrôles et les procédures spécialisées...

1.4.2 - LE PARAMETRAGE DES NOEUDS

Nous travaillons sur des requêtes relationnelles.
Pour nous, un paramètre représentera soit une relation, soit un opérateur relationnel.
Une relation n'a pas de descendant. Un opérateur a un nombre fixe de descendants directs (un ou bien deux, suivant son nom). Eventuellement, un des descendants directs de l'opérateur, (ou même les deux) est la relation vide.
En conséquence, nous pouvons poser pour principes :

- *Tous les paramètres sont obligatoires.*
- *Il n'y a pas de paramètres ramifications.*

Tout nom de paramètre est donc :

- Soit un élément de NE_1
- Soit un élément de NS_1 (cf 1.2.2)

Syntaxe :

< paramètre ::= <nom-paramètre> |
<nom-paramètre>, < prédicat>

< nom-paramètre > ::= < nom-étiquette > | < nom-pseudo-arborescence>

{ <nom-étiquette> est un identificateur alphanumérique préfixé par '&'. Exemple : '&x1', '&z2',etc...
{ <nom-pseudo-arborescence est un identificateur alphanumérique préfixé par "&&'. Exemple : '&&y2'

< prédicat > :: = /< prédicat-nom > / |
/< liste-prédicats- élémentaires> / |
/< prédicat-nom > , < liste-prédicats-élémentaires> /

<prédicat-nom> :: = <Identificateur> |
<Identificateur> ou< prédicat-nom >

<liste-prédicats-élémentaires> ::= < prédicat-élémentaire >|
< prédicat-élémentaire >,
<liste-prédicats-élémentaires>

< prédicat-élémentaire> ::= <Identificateur> (< liste-identificateurs >)

< liste-identificateurs> ::=<Identificateur >|
<Identificateur> , <liste-identificateurs>

4 exemples de paramètres :

&x1

&&y2/UNION/

&x3 /LAT (A, B, C)/

&&x1/PROJECT OU ANTIPROJECT, LAT (A,B),
SITEXECUTION (TARTEMPION)/

3 contre-exemples :

x1

/UNION/

&x1(UNION)

Une pseudo-arborescence sera codée de la manière suivante :

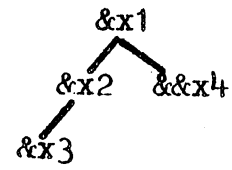
pseudo-arborescence ::= < paramètre >
< paramètre > (< pseudo-arborescence >)|
< paramètre > (< pseudo-arborescence >
<pseudo-arborescence>)

2 exemples :

&x1

&x1(&x2(&x3) &&x4)

Le dernier exemple représentant :



Bien entendu, si l'on autorise la configuration :

&x1 (&&x2)

On interdit formellement :

&&x2(&x1)

Puisque toute pseudo-ramification doit être une feuille.

(cf 1.2.2)

Pour coder les arborescences, on utilise les types PASCAL suivants :

```
PEP :  ELEMPRED;
ELEMPRED : record
             N: texte;
             FILS,
             FRERE : PEP
           end;
PTRANS :  TRANS;
TRANS  :   record
             P : TEXTE;
             M, E : integrer;
             ENSREG : set of integer;
             PEN, LPEI : PEP;
             SUIV : PTRANS;
           end;
```

```
ETAT :  record
          INVPT : set of integer;
          PT : PTRANS;
        end;

TETAT : array [1.. NBETATMAX] of ETAT;
```

TRANS

P	M	E	ENSREG	PEN	LPEI	SUIV

P est le nom du paramètre. Pour économiser de la place, on limite P à 4 caractères.

M est un indicateur de parenthèse. Il vaut :

+ 1 si le paramètre est suivi d'une parenthèse ouvrante (c'est-à-dire : le paramètre a un descendant)

- n si le paramètre est suivi de n parenthèses fermantes (c'est une feuille)

0 si le paramètre est suivi d'un autre paramètre sans être séparé par une parenthèse (ils sont "frères")

E est un indice dans le tableau TETAT. C'est l'état destinataire de la transition.

PEN est le pointeur vers la liste d'éléments du prédicat élémentaire nom

LPEI est le pointeur vers la liste des prédicats élémentaires

ENSREG, SUIV : leur rôle sera expliqué plus loin (chapitre 2.2)

ELEMPRED

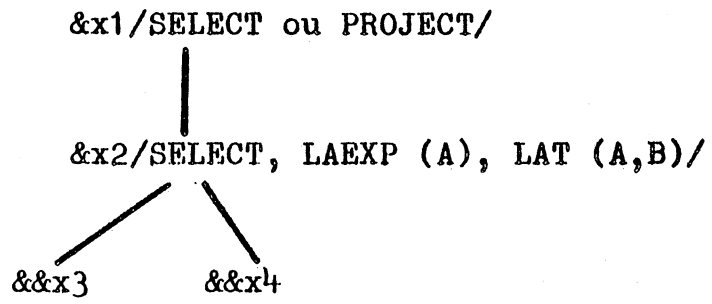
N	FILS	FRERE

N est un identificateur. En réalité, pour économiser de la place en mémoire, nous utilisons un codage analogue au codage MICROBE .

FILS, FRERE permettent de construire le prédicat élémentaire. Un exemple est nécessaire pour comprendre le chaînage et la signification de cette structure.

EXEMPLE COMPLET :

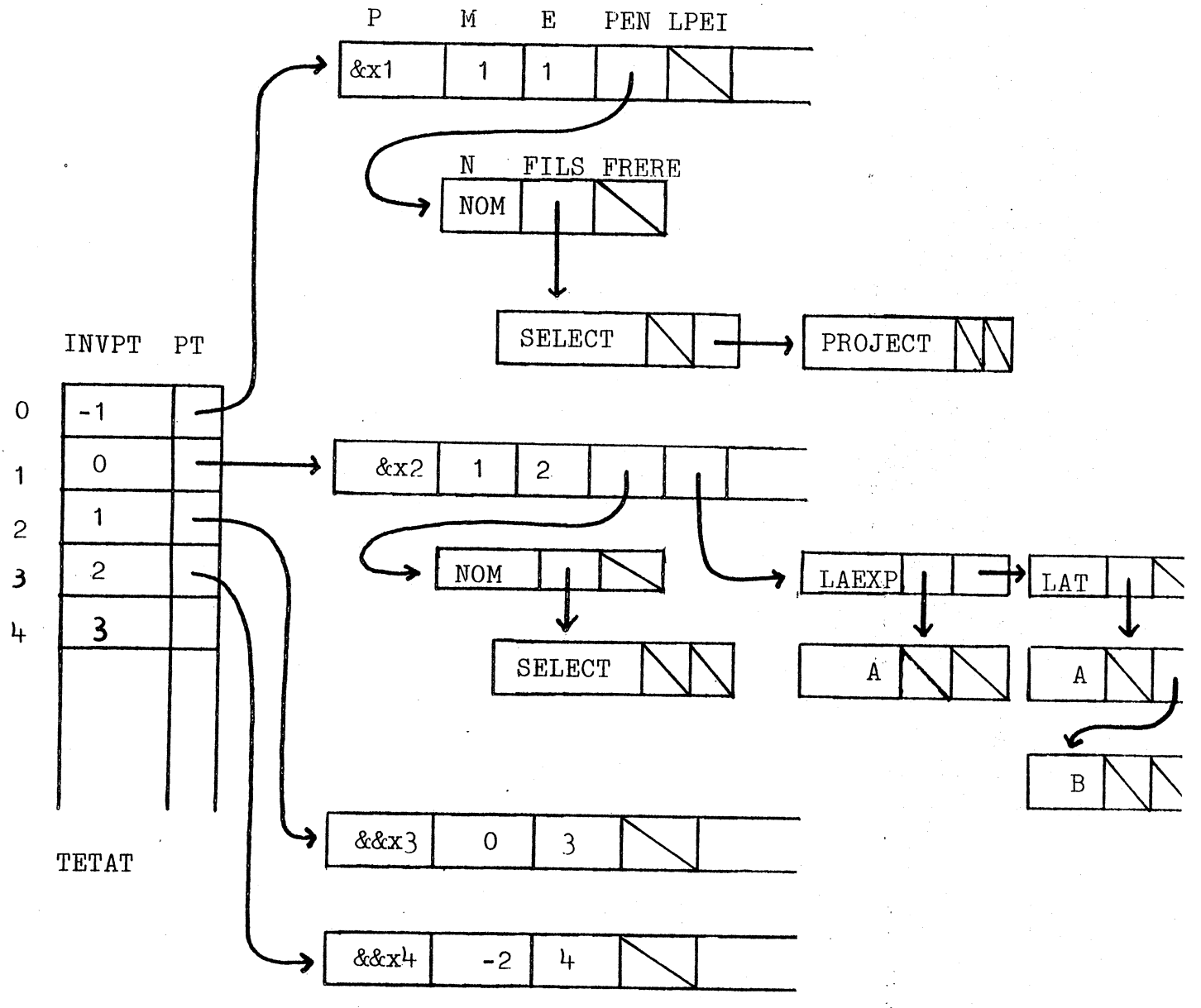
On veut coder l'arborescence :



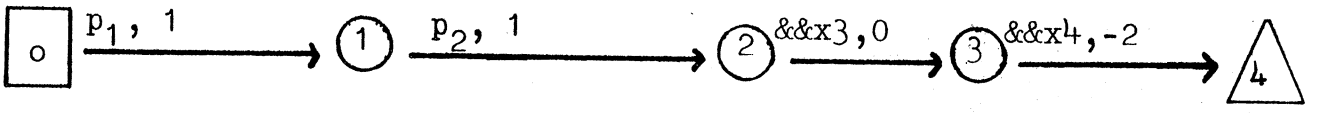
Elle doit être rentrée sous la forme suivante :

```
&x1/SELECT OU PROJECT/
    (&x2/SELECT, LAEXP (A), LAT (A,B)/
        (&&x3   &&x4))
```

Une fois codé, on aura le tableau de transitions d'états :



Nous avons en fait écrit un diagramme de transition, à savoir :



avec :

p_1 = &x1/SELECT OU PROJECT/

p_2 = &x2/SELECT, LAEXP (A), LAT (A B) /

Pour conclure, signalons qu'il faudra déclarer :

- Un ensemble d'états Q (ensemble d'entiers)

  ici Q = [0, 1, 2, 3, 4]

- Un état initial, ici c'est 0

- Un état final, ici c'est 4.

INVPT contient l'antécédant de l'état courant :

INVPT [3] =[2]

INVPT [1] =[0]

etc...

Par convention, INVPT [0] = [-1]

Nous verrons plus loin (chapitre 2) comment construire de tels diagrammes.

1.4.3 - MODELISATION DES OPERATEURS ET RELATIONS

Les paramètres étiquettes sont soit des relations de base, soit des opérateurs.

Explicitons les prédicats utilisés, et les contraintes.

Les opérateurs sont les opérateurs MICROBE. Ils diffèrent de ceux de la 1ère partie.

1) Relation de base :

&x1 - <identificateur> (TYPE = REL) (LAT) (SITE) (VOL)...

L'identificateur est le nom de la relation.

LAT, la liste des domaines de cette relation

SITE, le sit.

VOL le volume de la relation...

(liste non limitative)

Ces prédicats sont répercutés sur tous les noeuds opérateurs.

2) Projection, Antiprojection :

&x1 - PROJECT ou ANTIPROJECT (LA)

|

&&x2

LA désigne la liste d'attributs de la projection (ou de l'anti-projection.

Il faut vérifier que LA (&x1) $\subset$ LAT (&&x2)

On a : LAT (&x1) = LA (&x1)

3) Sélection :

&x1 - SELECT (LAEXP)

/

&&x2

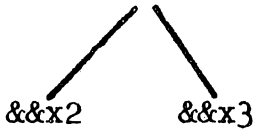
LAEXP désigne la liste d'attributs intervenant dans l'expression conditionnelle de la sélection.

On vérifie : LAEXP (&x1) $\subset$ LAT (&&x2)

On a : LAT (&x1) = LAT (&&x2)

4) Opérateurs ensemblistes usuels :

&x1 - UNION OU INTERSECTION OU DIFFERENCE

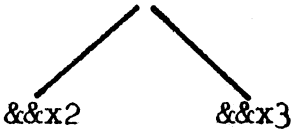


COMP (LAT (&&x2), LAT (&&x3)): vérifie si les domaines des attributs de &&x2 sont compatibles avec ceux de &&x3.

On a : LAT (&x1) = LAT (&&x2)

5) Join :

&x1 - JOIN (TYPE) (OP) (ATG) (ATD)



Il s'agit en fait du θ-PRODUIT.

ATG désigne l'attribut de la relation opérande de gauche intervenant dans le produit :

ATG (&x1) $\subset$ LAT (&&x2)

ATD idem, pour celle de droite :

ATD (&x1) $\subset$ LAT (&&x3)

OP est un opérateur de comparaison : $<$, $\leq$, $>$, $\geq$, $=$, $\neq$

TYPE est le type du Produit

. si OP = ' = ' alors TYPE = ' NATUREL'
(c'est le produit au sens de la partie 1)

. si OP $\neq$ ' = ' alors TYPE = ' GENERAL'

Dans le cas du Produit naturel, on a :

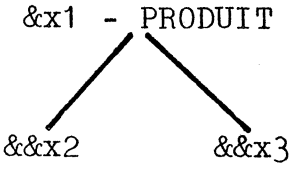
LAT (&x1) = LAT (&&x2) + (LAT (&&x3) - ATD (&x1))

Dans le cas général :

LAT (&x1) = LAT (&&x2) + LAT (&&x3)

Il faut aussi vérifier : COMP(ATG(&x1), ATD (&x1))

6) Produit



Il s'agit en réalité du Produit cartésien.

LAT (&x1) = LAT (&&x2) + LAT (&&x3)

CHAPITRE 2 : LES SYSTEMES GENERAUX DE TRANSFORMATIONS PARAMETREES

2.1 - LES SGTP

2.1.1 - DEFINITIONS

2.1.2 - LES NOTIONS DE FERMETURE

2.2 - AUTOMATES D'ETATS FINIS ADAPTES AUX RAMIFICATIONS PARAMETREES (A.R.P)

2.2.1 - LES A.R.P

2.2.2 - CONSTRUCTION

2.2.3 - REDUCTION

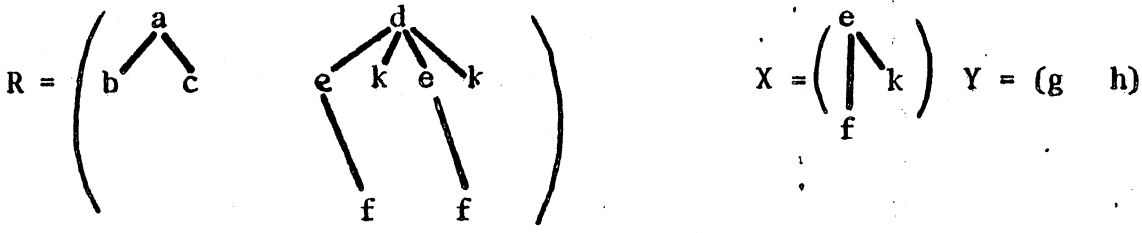
2.2.4 - LE SYSTEME DE TRANSFORMATION TRANSARBRE

2 - 1. LES SYSTÈMES GÉNÉRAUX DE TRANSFORMATIONS PARAMÉTRÉES (SGTP)

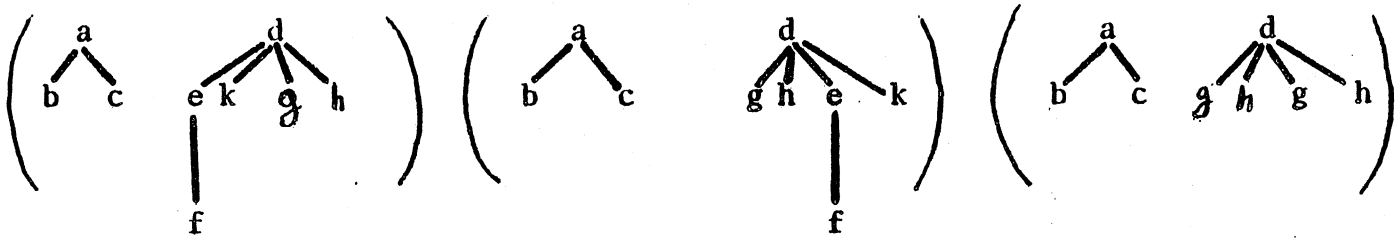
2.1.1. DEFINITIONS

On appelle REPL (R,X,Y) l'ensemble de ramifications produites par le remplacement d'une occurence et une seule d'une sous-ramification complète X d'une ramification R par une autre ramification Y. REPL représente l'ensemble de toutes les transformations possibles pour R,X et Y donnés.

Soit :



Alors REPL (R,X,Y) =



NB : REPL (ϕ,X,Y) = ϕ et REPL (R,X,Y) = R

un système de transformations paramétrées (STP) est un quintuplet G = (V,F,$\Rightarrow$,P,T):

V est un ensemble d'étiquettes ou vocabulaire

F est un ensemble de ramifications sur V; F est un sous-ensemble fini de $\hat{V}$, appelé ensemble de ramifications de base

T est l'ensemble de règles de transformations paramétrées

P est un ensemble de noms de paramètres

($\Rightarrow$) est la relation de transformation de ramifications.

T = $\{T_i\}$ i $\in$ {1,2,...,n}

Une règle de transformation paramétrée T_i est un quadruplet (X_i,Y_i,C_i,A_i) où :

X_i et Y_i sont des ramifications paramétrées appartenant à $\hat{V}$

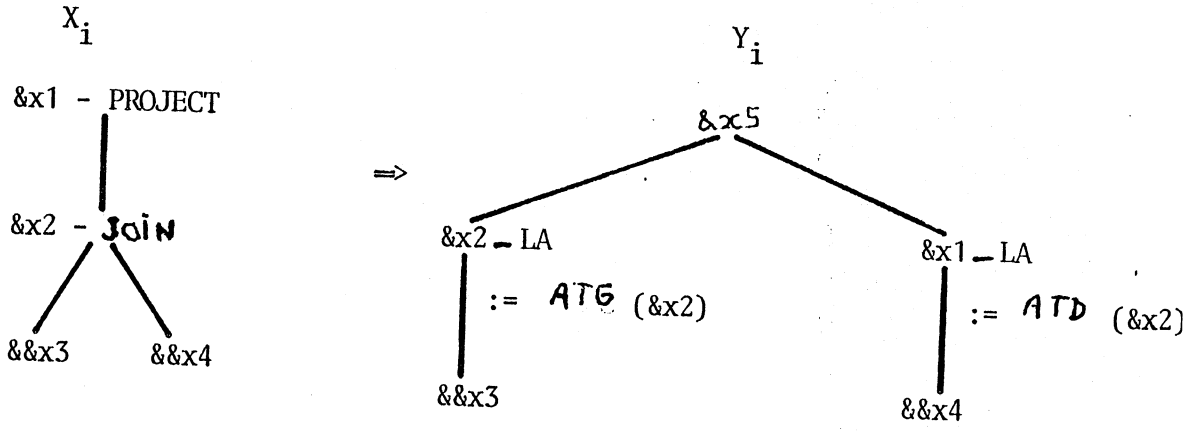
C_i est un prédicat qui a pour domaine l'ensemble des sommets de X_i

en fonction de conditions portant sur X_i (effets secondaires), et tel que :

i) les noms de paramètres sont uniques dans X_i .

ii) C_i ne porte sur aucun paramètre optionnel dans X_i.

Nous allons donner un exemple de règle de transformation paramétrée. Soit la règle de distribution de la projection par rapport au JOIN :



C_i : (LA (&x1) = ATG (&x2)) ou
(LA (&x1) = ATD (&x2))

A_i : si type (&x2) = 'NATUREL'
alors Nom (&x5) :=INTER
sinon début
NOM (&x5) := 'PRODUIT' ;
OP (&x5) := OP (&x2) ;
ATG (&x5) := ATG (&x2) ;
ATD (&x5) := ATD (&x2) ;
TYPE (&x5) := TYPE (&x2) ;
fin

Le Produit a deux types possibles : GENERAL ou NATUREL. Nous aurions pu choisir de considérer qu'il s'agissait de deux opérateurs différents et éclater la règle en deux.

Les prédicats (TYPE) et les effets de bord (A_i) permettent de réduire le nombre de règles, donc d'accroître l'efficacité.

Dans une affectation telle ATG (&x5) := ATG (&x2) ; le nom (&x5) à gauche se rapporte à Y_i, tandis que le nom à droite (&x2) se rapporte à X_i.

Nous conviendrons de remplacer la séquence encadrée par l'affectation globale &x5 := &x2 ; pour exprimer le fait que tous les prédicats sur &x5 (explicites ou non) prennent la même valeur que ceux de &x2.

Définition :

Une règle de transformation paramétrée $T_i = (X_i, Y_i, C_i, A_i)$ est applicable sur une ramification $R \in \hat{V}$ si, et seulement si :

1) dans R, il y a une sous-ramification complète X' telle que $X' \approx X_i$
2) le prédicat construit en remplaçant dans C_i tout paramètre $z = (z_1, z_2)$ de X_i par chaque étiquette correspondante de racine VAL [X' = X_i] (z_1) est vérifiée.

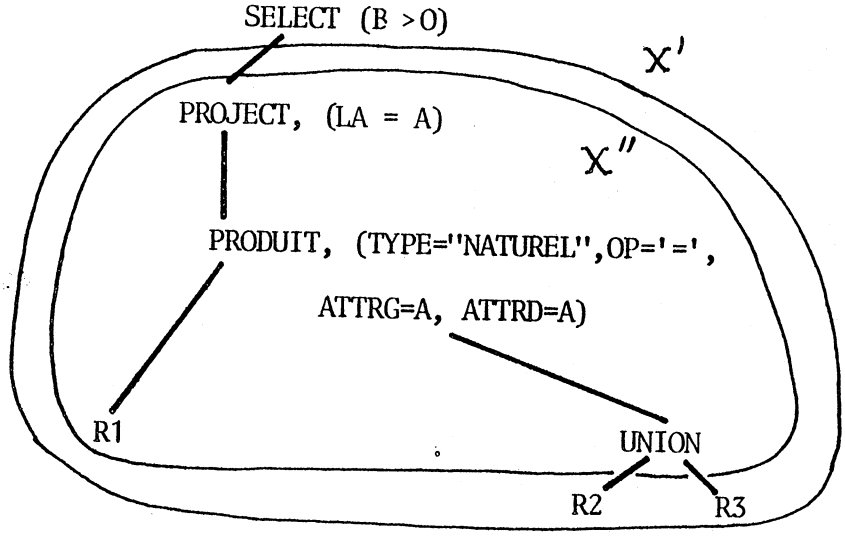
X' est le domaine d'application de T_i

La plus petite sous-ramification de X' équivalente à X_i est le domaine restreint d'application de T_i, soit X''.

La ramification construite en remplaçant dans Y_i tout paramètre z par VAL[X''=X_i] est construite par CONST. Il faut ensuite évaluer A_i par EVAL : $R \times \hat{V} \rightarrow \hat{V}$. Il y a donc deux étapes.

Y' = CONST (X'', X_i, Y_i) ; Y'' = EVAL (T_i, Y') ;

Exemple : Soit la ramification X :

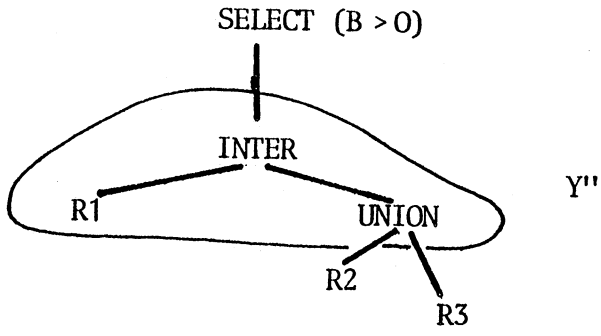


La sous-ramification X' est équivalente à Xi de la règle T_i précédente.

On vérifie que le prédicat construit en remplaçant dans Ci, chaque paramètre z=(z1,z2) de Xi par chaque étiquette de racine VAL [X=X_i] (z1) est vrai : la règle est applicable.

X' est le domaine d'application de Ti sur cet exemple . Le domaine restreint X'' est entouré.

Le lecteur vérifiera que le résultat est Y :



Y''

On écrira que :

$$X \overset{T_i}{\Longrightarrow} Y \qquad \text{et } (X,Y) \in (\overset{T_i}{\Longrightarrow})$$

Une même règle T1 peut être applicable sur plusieurs sous-arborescences distinctes de x. On peut donc avoir :

$X \overset{T1}{\Rightarrow} Y_1$, $X \overset{T_2}{\Rightarrow} Y_2$, et $X_1 \neq Y_2$.

2.1.2. LES NOTIONS DE FERMETURES

On appelle <u>dérivation</u> une séquence finie de ramifications telle que chaque ramification est le résultat de l'application d'une règle de transformation T_i sur la ramification précédente. Si la dérivation commence par R et finit par R', on dit que R' dérive de R.

$$R_1 \overset{T_i}{\Rightarrow} R_2 \ldots\ldots\ldots \overset{T_i}{\Rightarrow} R,$$

ou encore

$$R \overset{+}{\Rightarrow} R'$$

On écrira $R \overset{*}{\Rightarrow} R'$ si : $R = R'$ ou $R \overset{+}{\Rightarrow} R'$.

Une ramification est <u>invariable</u> relativement à un STP si aucune règle de transformation ne peut lui être appliquée dans ce STP.

Dans un STP, $G = (V,F,T,\Rightarrow,P)$, une ramification $S \in \hat{V}$ est une forme <u>normale</u> d'une ramification $R \in F$ si, et seulement si :

i) $R \overset{*}{\Rightarrow} S$

ii) S est invariable relativement à G.

Un STP, $G = (V,F,T,\Rightarrow,P)$ est <u>précis</u> si, et seulement si T est une fonction, c'est-à-dire s'il n'existe pas deux règles de transformation $T_i : X_i \Rightarrow Y_i$ et $T_j : X_j \Rightarrow Y_j$ telles que $X_i = X_j$ et $Y_i \neq Y_j$.

Un système de transformation $G = (V,F,T,\Rightarrow,P)$ possède la <u>propriété de Church-Rosser</u> si, et seulement si :

$\forall R \in F,\ S,S' \in \hat{V}$

$R \overset{*}{\Rightarrow} S$ et $R \overset{*}{\Rightarrow} S'$ impliquent

$\exists\ Q \in \hat{V} \mid S \overset{*}{\Rightarrow} Q,\ S' \overset{*}{\Rightarrow} Q$ et Q est invariable dans G.

De cette propriété découlent deux corollaires particulièrement importants :

(1) Toute ramification dans F a une forme normale unique,

(2) Le résultat de l'application des règles de transformation sur une ramification donnée ne dépend pas de l'ordre dans lequel les règles de transformation ont été appliquées.

Ces corollaires facilitent grandement le problème. Ils garantissent en particulier (1er corollaire) que l'algorithme induit par G s'arrête, que le résultat est unique; et enfin (2ème corollaire) que l'on n'est pas obligé d'explorer des configurations combinatoires. Notons qu'il faut prévoir un traitement informatique des boucles, mais dans le domaine qui nous intéresse, on arrive facilement à éviter les boucles et donc tout stokage de ramifications intermédiaires. Mais la propriété de Church-Rosser n'est pas facile à établir. C'est pourquoi, on est amené à utiliser une propriété plus forte : la T_ Fermeture.

Un SGTP G=(V,F,T,=>,P) est T - fermé si et seulement si la propriété suivante est vraie :

Si, sur une ramification $S \in \tilde{\tilde{V}}$, deux transformations T_i, $T_j \in T$ sont applicables, telles que :

$$S \xrightarrow{Ti} S_1$$

$$S \xrightarrow{Tj} S_1'$$

alors, il existe S_2 telle que :

$$S_1 \xrightarrow{Tj} S_2$$

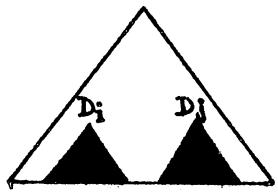
$$S'_1 \xrightarrow{Ti^*} S_2$$

Ceci veut dire que dans un système T-fermé toute transformation reconnue applicable, reste toujours applicable sans que son moment d'application ait une influence sur le résultat de la transformation.

On connaît une condition suffisante pour qu'un STP soit T-fermé. Cette condition peut s'exprimer ainsi :

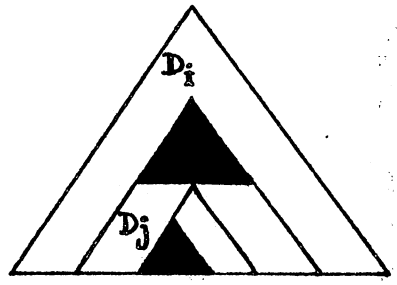
Soit un STP,G, auquel on peut appliquer plusieurs transformations (T_i) sur une même ramification; alors il faut que l'une de ces conditions soient remplies (Lemme) :

(1) Les domaines restreints Di et Dj d'application des règles de transformations Ti et Tj sont disjoints. (cas 1 ou 2)



Cas 1

ou bien



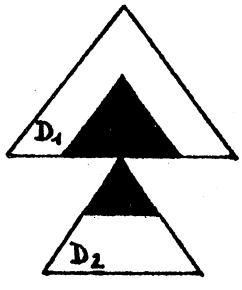
Cas 2

(2) D_i et D_j ont un sommet commun z, et vérifient certaines conditions à savoir :

- si la règle "plus petite" est exécutée d'abord, la nouvelle racine satisfait le prédicat d'application de la "règle grande". (la "règle grande" est applicable)
- si par contre la règle "la plus grande" est exécutée d'abord, il faut que les modifications faites aux étiquettes racine du domaine restreint de la "règle petite" ne changent pas son caractère de domaine d'application. [48]

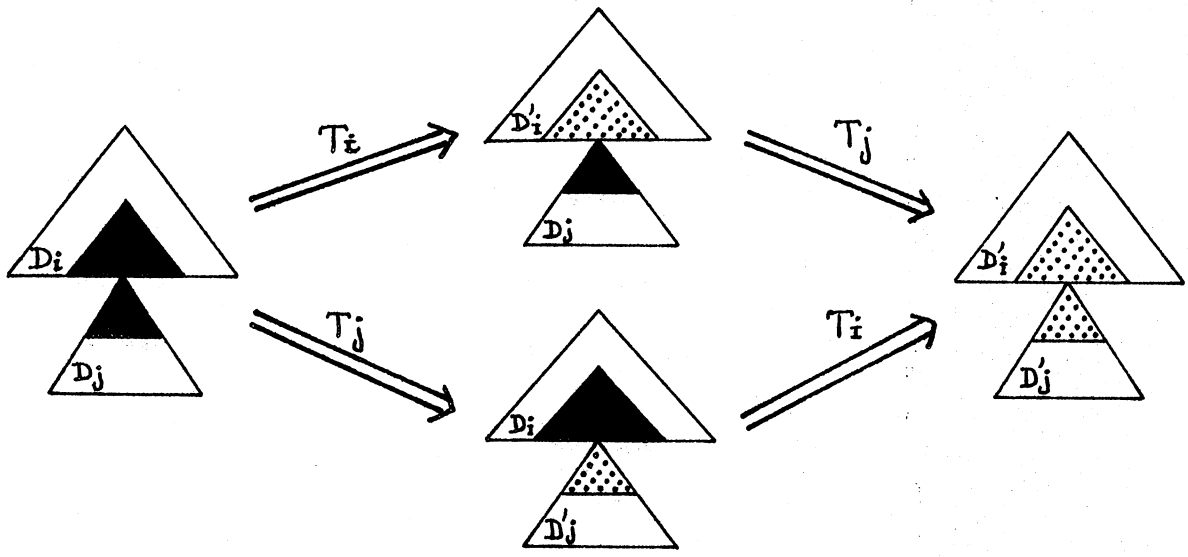
Précision : dans une arborescence, si deux domaines sont séquents, alors il y en a un qui est "plus grand" que l'autre, c'est-à-dire qu'un de ses sommets est racine de l'autre domaine qu'on qualifiera de "plus petit".

Exemple :



D_1 est plus grand que D_2

Les conditions du (2) ci-dessus peuvent s'exprimer ainsi graphiquement :



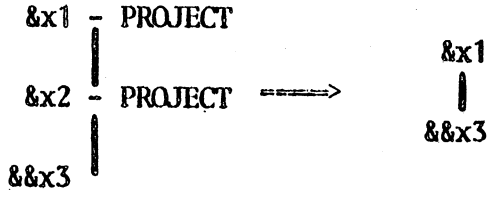
La propriété peut être étendue au cas où l'intersection des domaines restreints d'application des règles T_i et T_j est formé de plusieurs sommets ou même d'une sous-ramification.

On peut aussi définir des formes plus lâches de fermetures, mais elles sont moins faciles à vérifier que la T-fermeture. Les transformations T_i, T_j peuvent être deux applications différentes d'une même règle.

En conclusion, un bon STP sera pour nous un STP T-fermé ; pour l'assurer, nous chercherons à vérifier que si deux transformations sont susceptibles d'être appliquées sur une ramification produite à partir de F, elles satisfont le lemme donné.

Or,dans le cas des arborescences relationnelles, cette propriété est relativemer facile à vérifier, ou à forcer par des modifications sur les règles.
Voyons quelques exemples pour le montrer. Considérons la table DP de transformation qui réalise la distribution de la projection par rapport aux autres opérateurs (cf. Partie 1)

Soit la règle $DP_1 = (X_1, Y_1, C_1, A_1)$:

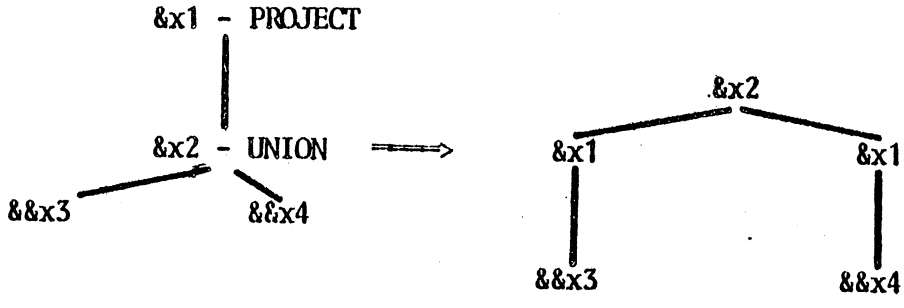


C_1 : ( )

A_1 : ( )

Un problème peut se poser si dans une requête on rencontre trois Project successifs : ($PROJECT_1$ ($PROJECT_2$ ($PROJECT_3$ (R1)))). DP_1 est applicable deux fois. On force alors l'ordre d'application de la règle DP_1 en la modifiant ainsi : on remplace (&&x3) par (&&x3 - ¬ PROJECT). La règle sera donc appliquée une première fois sur les PROJECT numéro 2 et 3, puis plus haut -soit successivement : ($PROJECT_1$ ($PROJECT_2$ (R1))) après la 1ère application ($PROJECT_1$ (R1)) après la 2ème application.

Il peut également y avoir conflit entre DP_1 et d'autres règles, par exemple DP_3:



C_1 = ( )

A_1 = ( )

Si on rencontre ($PROJECT_1$ ($PROJECT_2$ (UNION (R1 R2))))

Il y a deux transformations possibles, dont les résultats sont :

(1) (PROJECT$_1$ (UNION (R1 R2)))

(2) (PROJECT$_1$ (UNION (PROJECT$_2$ (R1)) (PROJECT$_2$ (R2)))

Dans les deux cas, on remarque que la deuxième règle est applicable, d'où :

(1') (UNION (PROJECT$_1$ (R1)) (PROJECT$_1$ (R2)))

(2') (UNION (PROJECT$_1$ (PROJECT$_2$ (R1))) (PROJECT$_1$ (PROJECT$_2$ (R2))))

Dans le deuxième cas, on remarque que la 1ère règle est applicable deux fois, d'où :

(2'') (UNION (PROJECT$_1$ (R1)) (PROJECT$_1$ (R2)))

On voit que (2'') est équivalent à (1'). Les deux cas conduisent au même résultat. Le résultat est indépendant de l'ordre d'application des règles, autrement dit nous avons vérifié que le Système, bien que n'étant pas à priori T-fermé, est Church-Rosser.

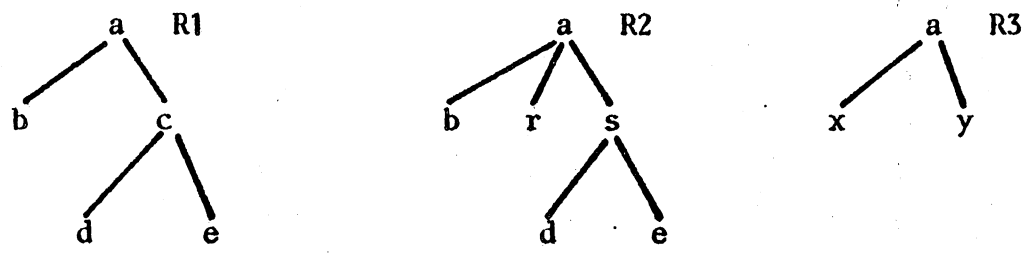
Il est clair cependant qu'il y a tout intérêt à appliquer d'abord la règle DP_1, c'est plus économique (deux transformation au lieu de quatre). Il faut donc rendre DP_1 prioritaire par rapport à DP_3 .

2 - 2. AUTOMATES D'ÉTATS FINIS ADAPTÉS AUX RAMIFICATIONS PARAMÉTRÉES (A.R.P.)

2.2.1. LES A.R.P.

Un réseau de transition est un ensemble de diagrammes d'états finis dans lesquels les noms des arcs peuvent être soit des symboles terminaux, soit des noms d'états initiaux d'un diagramme du réseau.

Soit à reconnaître les ramifications :



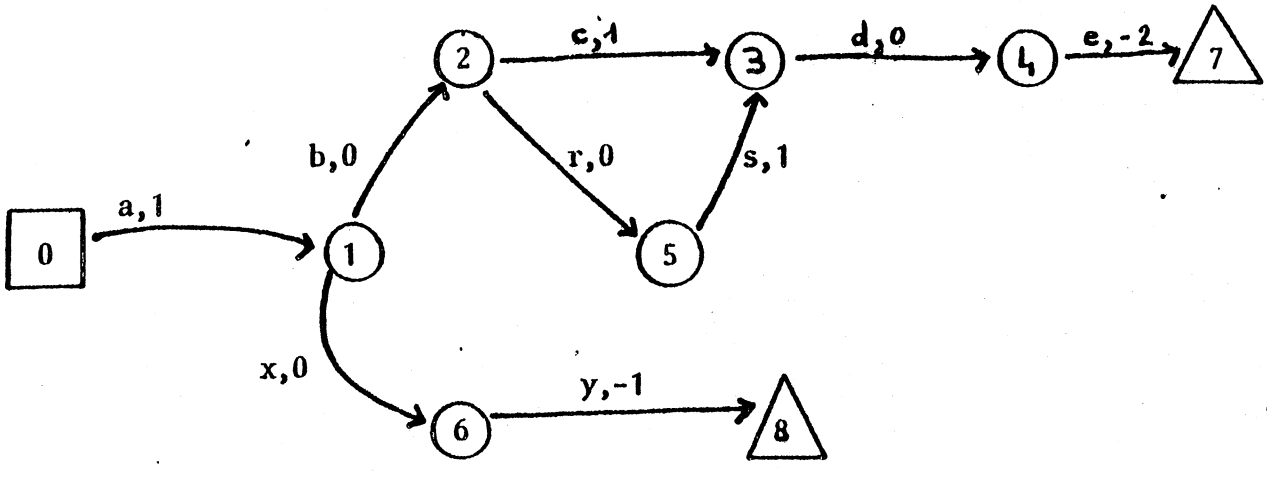
On peut associer un diagramme de transition au parcours en pré-ordre de chaque ramification . Ici, le parcours en pré-ordre donne avec les conventions usuelles :

PREORDRE (R1) = a(b,c (d,e))

PREORDRE (R2) = a(b,r,s (d,e))

PREORDRE (R3) = a(x,y)

Un seul réseau suffit à les reconnaître toutes :



A chaque arc, on a associé une variable entière (lg) qui remplace les parenthèses et permet donc de déterminer les niveaux des sous-arborescences.

L'état final 8 correspond à la ramification R3, l'état final 7 aux ramifications R1 et R2 (en mémorisant les transitions intermédiaires, on peut disitinguer)

Un tel réseau fournit un mode de stockage d'arborescences.

En introduisant la notion de paramètres comme précédemment, nous arrivons à un réseau paramétré. Nous parlerons d'Automate d'états finis adaptés aux Ramifications Paramétrées (ARP).

Un ARP est un 6-uplet M = (Q,V,P,qo,F,D) où :

Q est un ensemble fini d'états

qo est l'état final initial

$F \leq Q$ est l'ensemble des états finaux

V est le vocabulaire

P est l'ensemble des paramètres

D est la fonction de transition. C'est une application de QxP dans QxN où N est l'ensemble des entiers naturels.

De fait, construire l'ARP, c'est construire cette fonction de transition (et les états de Q).

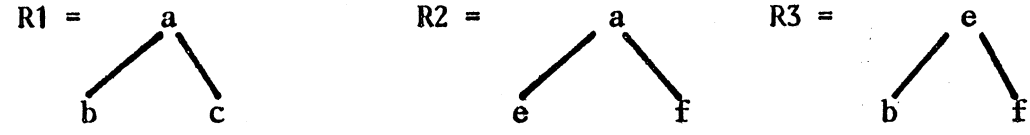
Cette construction se fait en deux étapes, soit deux procédures :

a) CONST-ARP permet la construction proprement dite à partir d'un état initial. Chaque état final correspond à une (sous)arborescence différente.

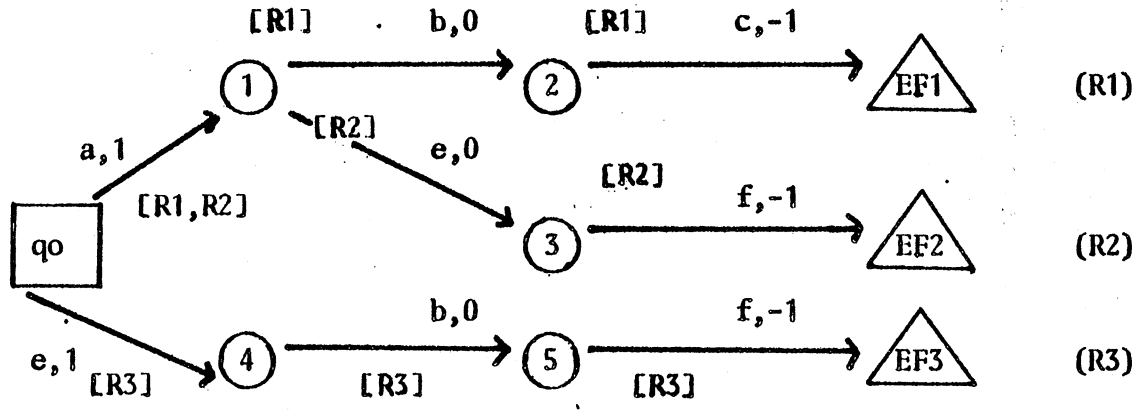
b) REDUCTION factorise à partir des états finaux afin de minimiser le nombre d'états et d'optimiser en conséquence la reconnaissance des arborescences.

Nous allons voir un exemple, soit R = (R1,R2,R3)

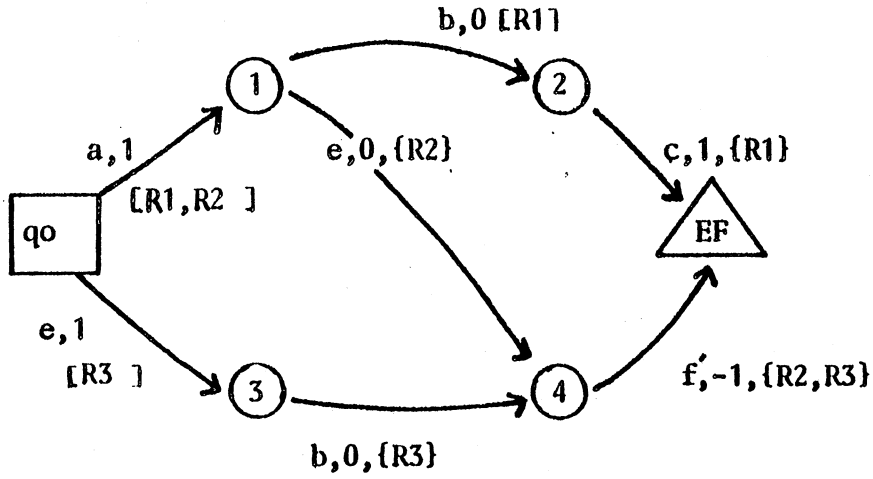
où R1 = a(b c), R2 = a(e f), R3 = e(b f)



L'ARP obtenu après CONST-ARP sur :



Puis, après REDUCTION :



On mémorise les étapes intermédiaires. Ainsi, le parcours qo-q1-q4-EF correspond à {R2} $\cap$ {R2,R3} = {R2} et qo-q3-q4-EF à {R3} $\cap$ {R2,R3} = {R3}

Le champ ENSERG des Transistions PTRANS permet de stocker les noms des arborescences (cf. 1.4.2.).

2.2.2 - CONSTRUCTION

Nous donnons en annexe la procédure PASCAL CONSTARP. Cette procédure lit une arborescence paramétrée au terminal ou dans un fichier REGLES.DAT, et l'ajoute dans un réseau de transition (cf 1.4.2), réseau représenté par un tableau de Type TETAT, à savoir :

XI pour les parties gauches des règles.

YI pour les parties droites.

Par rapport à l'algorithme de J. LOPEZ, la principale différence est qu'on a un seul état courant EC. Ceci est une conséquence directe du fait que l'arbre d'entrée correspond à la construction d'un automate déterministe, et du fait qu'il n'y a pas de paramètres optionnels.

Commentaires :

EQUIPRED compare deux paramètres, et délivre la valeur TRUE si ces deux paramètres (c'est-à-dire des transitions) ont le même prédicat.

TTRANS, Q, QF, EN, sont les variables globales correspondant à l'ARP en cours de construction, NREGLE le numéro de la règle courante.

PARAMETRE construit une Transition en lisant un paramètre sur le fichier donné (ou bien sur le terminal).

T est la variable texte courante sur le fichier d'entrée : c'est soit un identificateur, soit un symbole délimiteur. T est lu soit directement par LIRE, soit indirectement par PARAMETRE (qui utilise LIRE).

Schéma général de la procédure d'entrée des règles paramétrées :

```
Procédure LIREREGLE;
begin
  NREGLE : = o
  LIRE (T );
  While  T  < >'/' do
     begin
       NREGLE : = NREGLE + 1;
       CONSTARP (XI, QXI, QFXI, ENXI);
       CONSTARP (YI, QYI, QFYI, ENYI);
       LIRE_CONDITION (CI);
       LIRACTION (AI);
       LIRE (T );
     end;
  REDUCTION (XI, QXI, QFXI, EFXI, ENXI);
  REDUCTION (YI, QYI, QFYI, EFYI, ENYI);
end; (* LIREREGLE *)
```

2.2.3 - REDUCTION

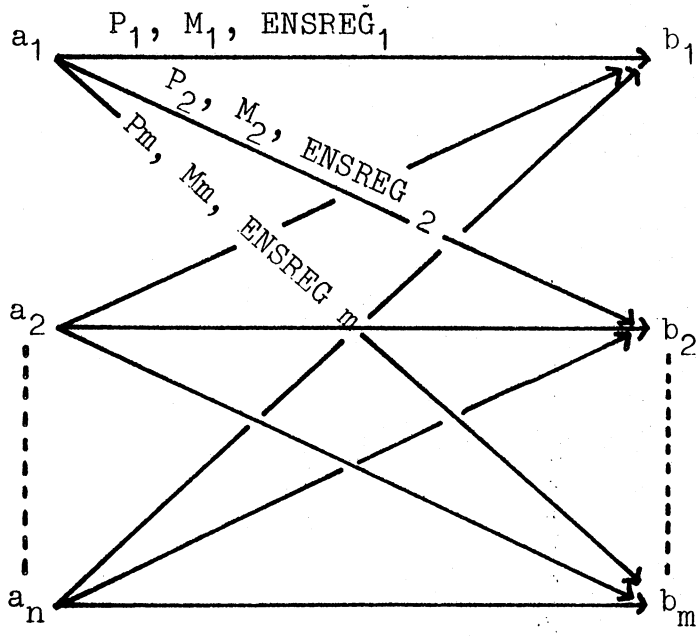
Nous donnons en annexe la procédure PASCAL REDUCTION.

Le principe de factorisation est le suivant :

1) On commence par remplacer tous les états finaux par un seul état final unique E F.

2) Ensuite, on factorise les états, en partant de l'état final, selon le principe suivant :

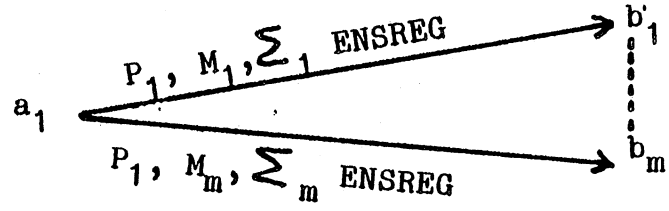
   - Soit a_1, a_2, ..., a_n des états transitant vers les mêmes états b_1, b_2,..., b_m



   - On fusionne a_1, a_2,...a_n si et seulement si les transitions de ces états sont identiques. Ici, on suppose qu'elles sont toutes identiques à l'ensemble

$\{(P_1, M_1, b_1), (P_2, M_2, b_2), \ldots, (P_m, M_m, b_m)\}$

On obtient alors :



Commentaires :

La procédure de REDUCTION utilise la fonction logique EQUIDELTA qui prend la valeur TRUE si la liste des transitions des états I2 et I3 qu'elle compare, est identique.

REDUCTION est une procédure optimisée :

Au lieu de créer un troisième état lorsqu'on rencontre deux états à fusionner (I2, I3) - on modifie les transitions, ce qui évite de créer un nouvel état et tout ce qui s'ensuit. En outre, lorsqu'il y a 'n' états à fusionner, on le fait en un seul coup, au contraire de l'algorithme LOPEZ (cf [48], page 88) qui procède par des fusions successives deux à deux, ce qui multiplie les comparaisons et les créations d'états.

Enfin, le traitement de l'ensemble des règles possibles (ENSREG chez nous, VALQ chez LOPEZ) a été radicalement simplifié.

Comme exemple, nous donnons en annexe la table de distribution de la Projection, après réduction.

2.2.4 - LE SYSTEME DE TRANSFORMATION TRANSARBRE

Nous donnons en annexe les procédures essentielles du système TRANSARBRE, écrites en PASCAL.

Ces procédures permettent :

- de lire une table de transformation (cf 2.2.2)
- d'exécuter ces transformations sur un arbre du type MICROBE

Nous supposons :

- que l'ensemble des règles paramétrées données et les arbres analysés constituent un SGTP T-fermé,

- qu'il n'y a pas de boucles dans les dérivations (on ne mémorise donc pas les ramifications intermédiaires),

- les dérivations sont de longueur finie.

Le système n'effectue aucun contrôle. Dans la pratique, il est apparu qu'il était facile d'écrire les règles de manière à assurer ces hypothèses, notamment en comparant les règles deux à deux pour assurer la condition nécessaire de T-fermeture vue précédemment (cf. 2.1.2)

Il n'y a pas de recopie des arbres transformés, mais seulement modification des noeuds du domaine d'application restreint - d'où un gain de temps considérable...au prix d'une grande complexité de l'ensemble des procédures qui réalisent l'exécution d'une règle!

CONCLUSION GENERALE

1 - REALISATION : LES RESTRUCTURATIONS ALGEBRIQUES

2 - AUTRES APPLICATIONS A L'OPTIMISATION

2.1 - PARTAGE DE REQUETES

2.2 - BIBLIOTHEQUE

3 - LES SGTP, UN OUTIL GENERAL D'AIDE A LA REALISATION DES SGBD RELATIONNELS

1 - REALISATION : LES RESTRUCTURATIONS ALGEBRIQUES, LE SYSTEME TRANSARBRE.

Les règles de transformations paramétrées (XI, YI, CI, AI) sont entrées au terminal, ou lues dans un fichier :

XI, YI sont des arborescences paramétrées, construites par CONSTARP et réduites par REDUCTION

CI, AI sont construites par la procédure PARAMETRE.

Un ensemble de règles constitue une table de transformations. Il y a les tables suivantes :

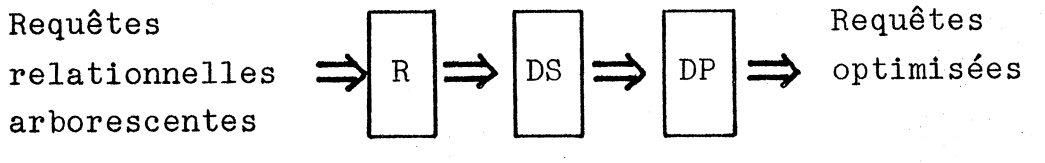
R : la table de réduction

DS : la table de distribution de la Sélection

DP : la table de distribution de la Projection.

On peut en concevoir d'autres, à volonté, sans avoir à modifier le système. Il suffit pour chaque nouvelle table, d'écrire un nouveau fichier qui sera lu à l'initialisation du Système de Gestion de la Base de Données.

Il en résulte une très grande souplesse du système de restructuration algébrique TRANSARBRE.
En appelant la procédure TRANSFORMER, on active le système de la manière suivante :



Actuellement, le système ne lit qu'une seule requête et n'exécut qu'une seule table à la fois.

Le paramétrage est encore rudimentaire, les conditions CI ne sont pas testées.

Nous arrivons cependant à effectuer la restructuration complète d'un arbre conséquent (6 opérateurs, 5 relations) en moins de 2 secondes sur un MICRO 1 de DEC, soit beaucoup plus vite que l'ancien système de transformation ne le faisait sur un IBM 360/67. (cf EXEMPLE COMPLET en ANNEXE)

Nous reconnaissons que le traitement complet des conditions et des paramètres (en cours de réalisation) conduira à une dégradation sensible des performances, en particulier à cause de l'overlay nécessaire du fait de la très faible taille de la mémoire disponible sur cette machine (124 K mots). Mais les nouveaux micro-ordinateurs offrent des possibilités nettement supérieures. Ceci compense cela, et nous sommes en droit d'affirmer avoir validé la méthode.

2 - AUTRES APPLICATIONS A L'OPTIMISATION

Tous les optimiseurs de SGBD relationnels s'attachent à optimise une requête donnée. Pour un SGBD d'une certaine importance, il y a utilisation partagée par beaucoup d'utilisateurs et donc pro babilité élevée d'avoir plusieurs requêtes simultanées ou quasi-simultanées. S'il s'agit de requêtes d'interrogation (lecture seule), et c'est l'immense majorité des cas, on a le droit de partager les données entre plusieurs utilisateurs - d'où l'idée de partager des requêtes ou des morceaux de requêtes.
Dans le traitement différé (batch processing), c'est relativement évident.
Dans le traitement instantané, ou quasi-instantané (real-time processing), on peut très bien se ramener au cas précédent. Il suffit de déterminer un délai d'attente, adapté aux besoins particuliers du SGBDR considéré. (Nous avançons l'hypothèse que ce délai d'attente est réaliste pour un grand nombre de cas).

Au bout de ce délai, le système examine les requêtes. Il les décompose, et les optimise à l'aide des procédés décrits plus haut (ceci permet de "normaliser" les requêtes). On a ainsi une liste de requêtes à exécuter.

Plusieurs cas se présentent .

Tout d'abord, s'il y a des requêtes de mise-à-jour, il faut bloquer les relations impliquées et exécuter ces m-a-j en priorité.

Après cela, il faut regrouper les requêtes d'interrogation.

2.1 - PARTAGE DE REQUETES

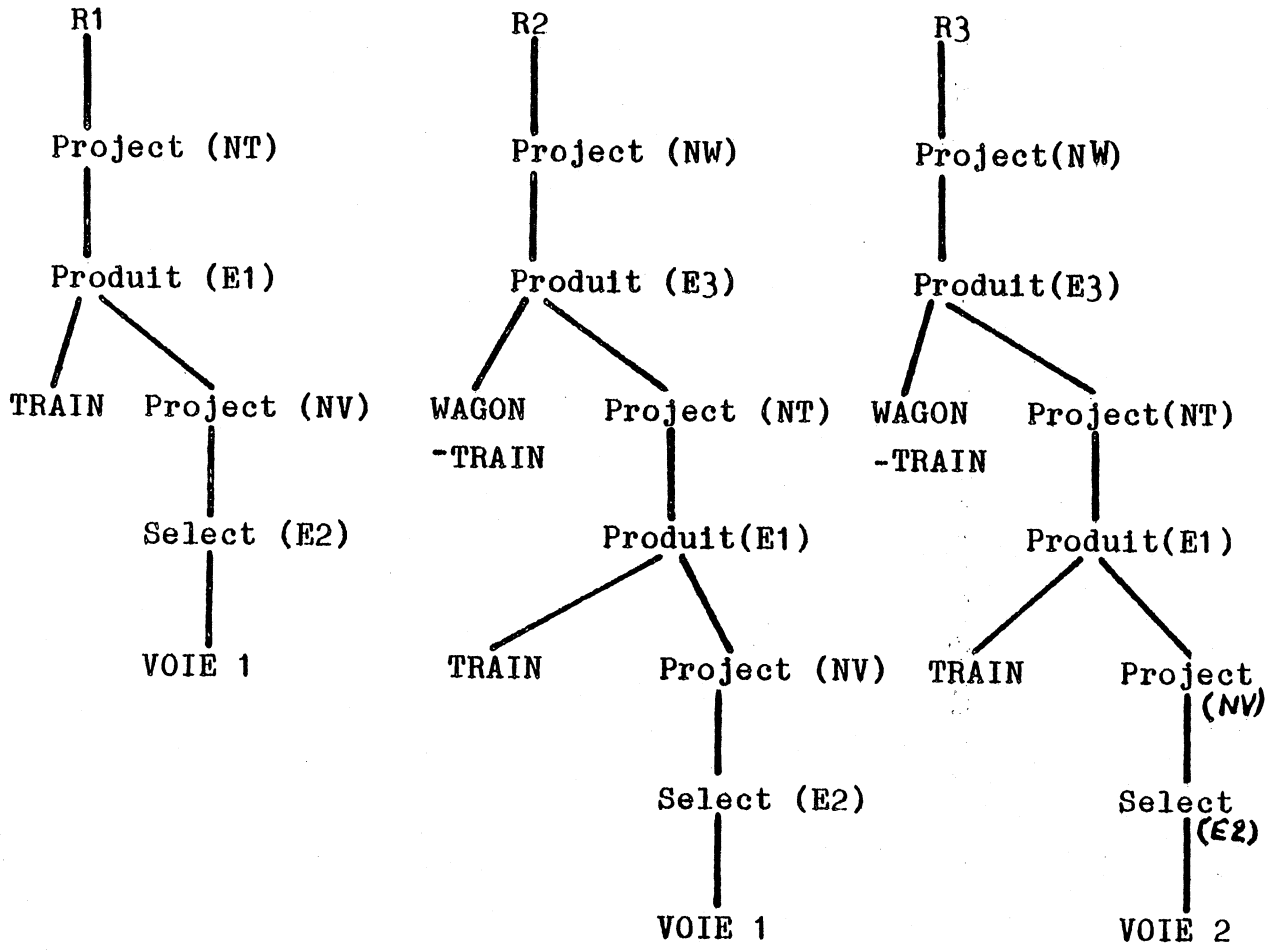
Soit les trois requêtes suivantes :

(R1) "Donner la liste des trains arrivant à Paris"

(R2) "Donner la liste des wagons arrivant à Paris"

(R3) "Donner la liste des wagons (attachés) en gare de triage à Paris". (cf Partie I)

Après décomposition et optimisation, ces trois requêtes s'écrivent ainsi :



où :

E1 = (NV=NV)

E2 = (GA='Paris')

E3 = (NT=NT)

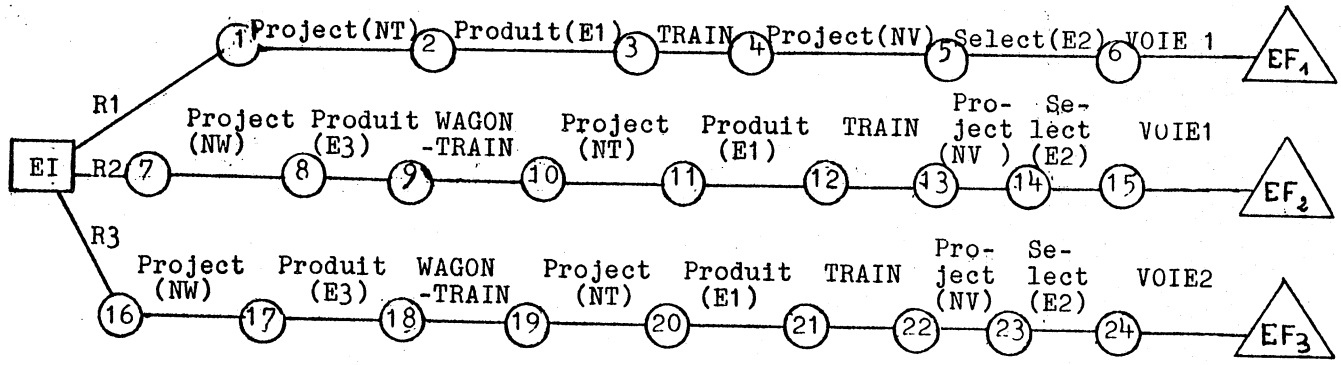
En notation préfixée :

R1 (Project (NT)(Produit(E1)(TRAIN, Project(NV)(Select(E2)(VOIE 1)))))

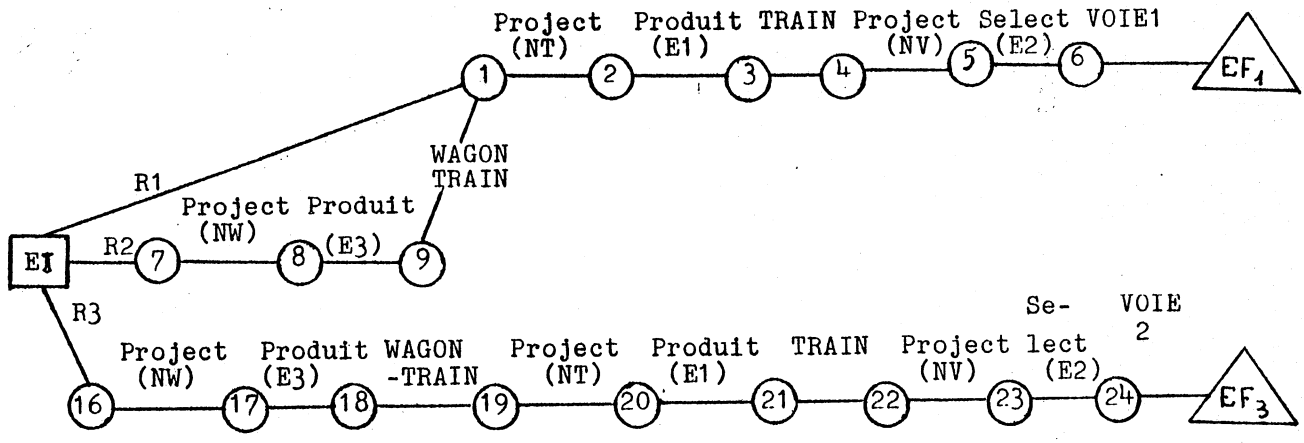
R2 (Project (NW)(Produit(E3)(WAGON-TRAIN, Project(NT)(Produit(E1)(TRAIN, Project(NV)(Select(E2)(VOIE1))))))

R3 (Project (NW)(Produit(E3)(WAGON-TRAIN, Project(NT)(Produit(E1)(TRAIN, Project(NV)(Select(E2)(VOIE2))))))

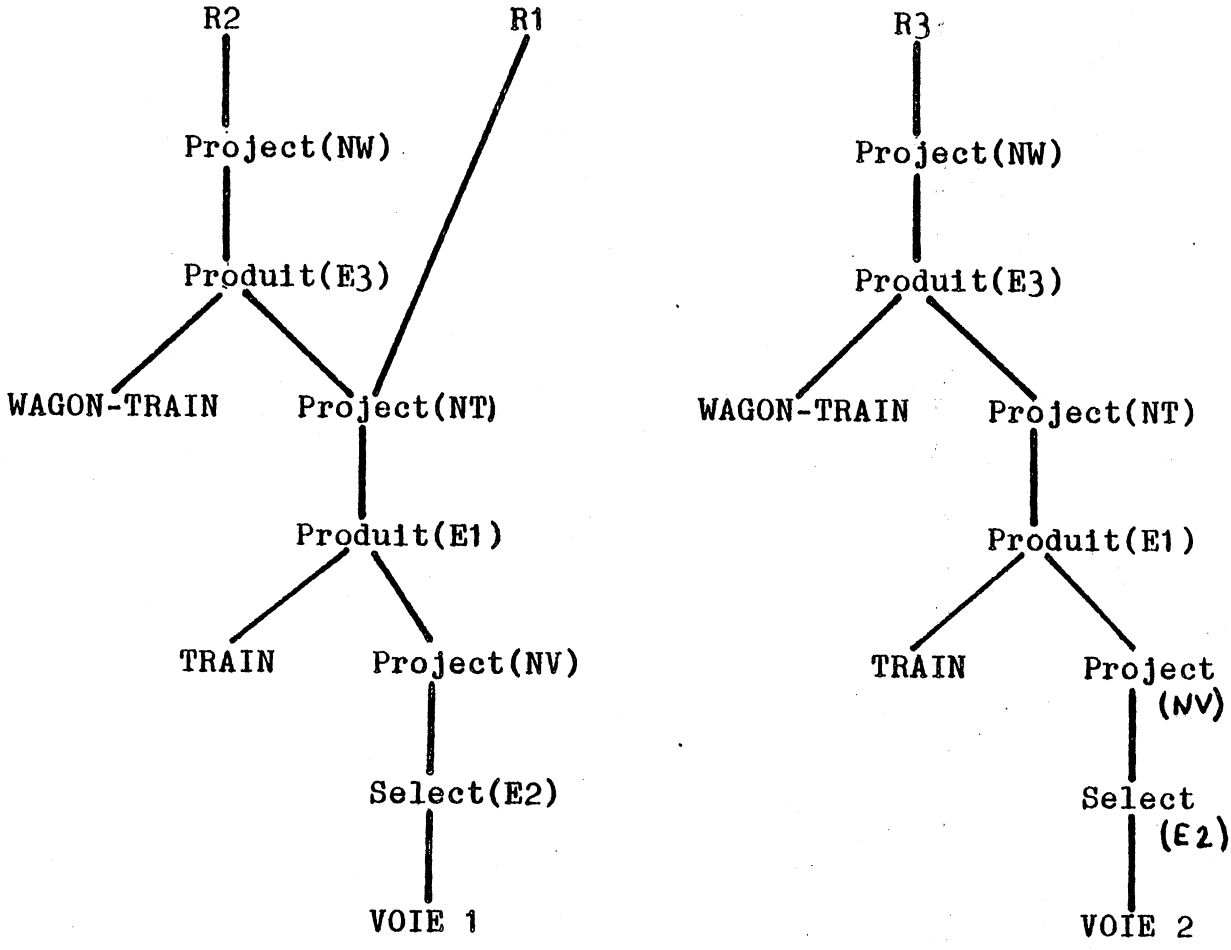
Soit, sous une forme analogue aux réseaux de transitions :



On factorise ensuite à partir des états finaux :



Soit finalement :



Ici le regroupement signifie en fait que R1 fournit un résultat intermédiaire pour R2. On économise le calcul d'une partie de R2.

2.2 - BIBLIOTHEQUE

Nécessairement, un certain nombre de requêtes reviennent.
Ainsi, pour notre exemple, on demandera souvent :
"Liste des trains, avec leurs wagons et la voie sur laquelle ils circulent dans la région parisienne"
soit :(WAGON-TRAIN * TRAIN * VOIE) : (GA ="Paris') (NV)
Il y a deux cas :

a) Si possible, stocker cette relation. Chaque fois qu'elle intervient dans une question, on a déjà le résultat calculé. On utilise ainsi la place éventuellement disponible en mémoire secondaire (disques) jusqu'à ce qu'il y ait mise-à-jour, ou encombrement.

b) La requête n'est pas stockée, faute de place (par exemple) mais on peut la compiler une fois pour toutes, après l'avoir optimisée; chaque fois que cette question est posée on peut alors utiliser directement la requête précompilée.

Ainsi, on définit un catalogue de requêtes, plus ou moins évolutif, qui permet de supprimer toute la première partie d'optimisation et, éventuellement de supprimer des exécutions.
L'expression relationnelle des requêtes précompilées est catalogguée sous forme d'A.R.P. au moyen de procédures analogues à CONSTARP et REDUCTION.
Lorsqu'une nouvelle requête est soumise au système, la reconnaissance d'une requête analogue précompilée s'effectue selon le même principe que la reconnaissance des sous-arbres dans le S.G.T.P.

3 - LES S.G.T.P., UN OUTIL GENERAL D'AIDE A LA REALISATION DES SGBD RELATIONNELS

Notre travail a essentiellement porté sur l'optimisation des SGBD Relationnels.
Nous avons vu que les SGTP étaient parfaitement applicables à cette optimisation, même pour des micro-ordinateurs.

Il ne faut pas oublier que TRANSARBRE est en fait un outil de restructuration d'arbres, et que nombre de parties d'un SGBD tel MICROBE utilisent des règles de transformations, écrites sous forme de procédures spécialisées.

Une autre application particulièrement intéressante du système TRANSARBRE nous parait être la traduction des requêtes formulées en langage de haut niveau (langage externe) en langage de bas niveau (langage interne), par exemple la traduction MICRO-SEQUEL ⟶ URANUS. [36]
Ceci pourrait faciliter la conception de langages externes plus orientés vers l'utilisateur, qu'il s'agisse de "langages naturels" ou non-langages dont le besoin devient de plus en plus aigü à l'ère de l'informatique individuelle où l'administrateur et l'utilisateur de la Base de Données sont souvent la seule et même personne.

C'est dans cette direction que porteront nos travaux futurs.

ANNEXE

1 - LES PRINCIPALES PROCEDURES

1.1 - CONSTRUCTION DE L' A.R.P

1.2 - REDUCTION

1.3 - TRANSFORMATION

2 - EXEMPLE COMPLET

1 - LES PRINCIPALES PROCEDURES

1.1 – CONSTRUCTION DE L'A.R.P

```
PROCEDURE CONSTARP(VAR TTRANS : TETAT; VAR Q, QF : ENS; VAR EN : INTEGER)

(*  TTRANS  : NOM DE L'A.R.P. EN COURS DE CONSTRUCTION                    *
(*       Q  : ENSEMBLE DES ETATS DE L'A.R.P.                              *
(*      QF  : ENSEMBLE DES ETATS FINALS                                   *
(*      EN  : ETAT NOUVEAU                                                *
(*    T, T1 : VARIABLES DE TYPE TEXTE, SERVANT A LIRE LES DONNEES         *

VAR

    EC,              (* ETAT COURANT DE L'ARP EN COURS DE CONSTRUCTION    *
    LG : INTEGER;    (* INDICATEUR DE PARENTHESES                         *
    NOMP : MOT4;     (* NOM DE PARAMETRE, LIMITE A 4 CARACTERES           *
    PAR,             (* PARAMETRE COURANT                                 *
    TR1 : PTRANS;
    TROUVE : BOOLEAN;


BEGIN (* CONSTARP *)

LG := 0; EC := 0;

T := T1;
IF (T^1$ <> ';') AND (T^1$ <> '/') THEN
  PAR := PARAMETRE(T,T1);

WHILE T^1$ <> ';' DO

IF T1^1$ = '(' THEN
  BEGIN
     LG := 1; LIRE(T1);
  END
    ELSE IF T1^1$ = ')' THEN
      BEGIN
         LG := LG - 1; LIRE(T1);
      END

      ELSE
      BEGIN
         (* ON REGARDE SI LA TRANSITION  ( EC, LG, PAR ) EXISTE DEJA *)
         TROUVE := FALSE;
         TR1 := TTRANS^EC$.PT;
         NOMP^1$ := T^1$; NOMP^2$ := T^2$;
         NOMP^3$ := T1^3$; NOMP^4$ := T1^4$;
         WHILE (NOT TROUVE) AND (TR1 <> NIL) DO WITH TR1^ DO
           IF (P = NOMP) AND (M = LG)
               AND (EQUIPRED(PAR,TR1))
           THEN TROUVE := TRUE
              ELSE TR1 := SUIV;
```

```
          IF TROUVE THEN
            BEGIN
               (* LA TRANSITION EXISTE : ON NE CREE PAS D'ETAT *)
               TR1^.ENSREG := °NREGLE$ + TR1^.ENSREG;
               EC := TR1^.E;
               DISPEP(PAR^.PRED); DISPOSE(PAR);
            END

              ELSE
              BEGIN
                 (* CREATION DU NOUVEL ETAT *)
                 IF EN = NBETAT THEN
                   WRITELN('ERREUR CONSTARP : TROP D''ETAT')
                     ELSE EN := EN + 1;
                 Q := Q + °EN$;
                 WITH PAR^ DO
                   BEGIN
                      M := LG; E := EN; ENSREG := °NREGLE$;
                      SUIV := TTRANS°EC$.PT;
                   END;
                 TTRANS°EC$.PT := PAR;
                 EC := EN;
              END; (* IF THEN ELSE TROUVE *)

          T := T1;
          IF (T°1$ <> ';') AND (T°1$ <> '/') THEN
             PAR := PARAMETRE(T,T1);

          LG := 0;
        END; (* IF THEN ELSE T2 *)

QF := °EC$ + QF;


END; (* CONSTARP *)
```

1.2 - REDUCTION

```
PROCEDURE REDUCTION(VAR TTRANS : TETAT; VAR Q, QF : ENS;
                    VAR EF, EN : INTEGER);
VAR
    CHANGE : BOOLEAN;
    I1, I2, I3, I4 : INTEGER;
    TR2, TR3, TR4 : PTRANS;
    QNF, Q1, INVI1 : SET OF INTEGER;

BEGIN

EF := 0;
IF Q <> °0§ THEN
BEGIN

(* CHAINAGE INVERSE *)
QNF := Q - QF;
FOR I1 := 0 TO NBETAT DO IF I1 IN QNF THEN
  BEGIN
    TR2 := TTRANS°I1§.PT;
    WHILE TR2 <> NIL DO WITH TR2^ DO
      BEGIN
        TTRANS°E§.INVPT := °I1§ + TTRANS°E§.INVPT;
        TR2 := SUIV;
      END;
  END;
(* FIN CHAINAGE INVERSE *)

(* CREATION DE L'ETAT FINAL UNIQUE EF *)

IF EN = NBETAT THEN
  WRITELN('ERREUR REDUCTION : TROP D''ETATS')
    ELSE EF := EN + 1;
EN := EF;
FOR I1 := 1 TO EF DO IF I1 IN QF THEN
  BEGIN
    FOR I2 := 0 TO EF DO IF I2 IN TTRANS°I1§.INVPT THEN
      BEGIN
        TR2 := TTRANS°I2§.PT;
        WHILE TR2 <> NIL DO WITH TR2^ DO
          BEGIN
            IF E = I1 THEN
              E := EF;
            TR2 := SUIV;
          END;
        TTRANS°EF§.INVPT := °I2§ + TTRANS°EF§.INVPT;
      END; (* FOR I2 *)
    TTRANS°I1§.INVPT := °§;
    Q := Q - °I1§;
    QF := QF - °I1§;
  END; (* FOR I1 *)
QF := °EF§; Q := °EF§ + Q;
(* FIN CREATION DE L'ETAT FINAL UNIQUE EF *)
```

```
CHANGE := TRUE;
Q1 := QF;
WHILE CHANGE DO
BEGIN
  CHANGE := FALSE;
  FOR I1 := 1 TO EF DO IF I1 IN Q1 THEN
  BEGIN
    INVI1 := TTRANS^I1$.INVPT;
    FOR I2 := 1 TO EF DO
     IF I2 IN INVI1 THEN
      FOR I3 := 1 TO EF DO
        IF (I3 <> I2) AND (I3 IN INVI1)
            AND (EQUIDELTA(I2,I3,TTRANS))
        THEN
        BEGIN
          CHANGE := TRUE;

          WITH TTRANS^I2$ DO
            BEGIN (* INTEGRATION DE L'ETAT I3 A L'ETAT I2 *)
              TR2 := PT;
              WHILE TR2 <> NIL DO
                BEGIN
                  TR3 := TTRANS^I3$.PT;
                  WHILE TR3 <> NIL DO WITH TR3^ DO
                    BEGIN
                      IF (P = TR2^.P) AND (M = TR2^.M)
                          AND (E = TR2^.E)
                          AND (EQUIPRED(TR3,TR2))
                          THEN  TR2^.ENSREG := TR2^.ENSREG + ENSREG;
                      TR3 := SUIV;
                    END; (* WITH TR3 *)
                  TR2 := TR2^.SUIV;
                END; (* WHILE TR2 *)
              INVPT := INVPT + TTRANS^I3$.INVPT;
            END; (* INTEGRATION DE L'ETAT I3 A L'ETAT I2 *)

          (* DEBUT SUPPRESSION I3 *)
          WITH TTRANS^I3$ DO
          FOR I4 := 0 TO EF DO IF I4 IN INVPT THEN
          BEGIN
            TR4 := TTRANS^I4$.PT;
            WHILE TR4 <> NIL DO WITH TR4^ DO
              BEGIN
                IF E = I3 THEN
                  E := I2;
                TR4 := SUIV;
              END;
            DISPTRANS(PT);
            INVPT := ^$; Q := Q - ^I3$; Q1 := Q1 - ^I3$;
            INVI1 := INVI1 - ^I3$;
            TTRANS^I1$.INVPT := INVI1;
          END; (* FIN SUPPRESSION I3 *)

        Q1 := ^I2$ + Q1;
      END; (* FOR I2, I3 *)
    END; (* FOR I1 *)
END; (* WHILE CHANGE *)

END; (* IF THEN Q *)

END; (* REDUCTION *)
```

1.3 - TRANSFORMATION

```
FUNCTION DELTA(VAR E1, N1 : INTEGER; VAR TTRANS : TETAT): PTRANS;


(* ETANT DONNE UN ETAT E1 DE L'A.R.P. DE CONTROLE ET UN  *)
(* NOEUD N1 D'UNE ARBORESCENCE A ANALYSER, DELTA FOURNIT *)
(* LA LISTE DES TRANSITIONS POSSIBLES A PARTIR DE E1     *)


VAR
    D, TR1, TR2 : PTRANS;
    T : TEXTE; NOMP, T4 : MOT4;


BEGIN

D := NIL;
WITH ARBRE^N1$ DO
  IF TTYPE = 'R'
    THEN T := TRELAT^TNUM$.IDENTREL
      ELSE
      IF TTYPE = 'O' THEN
        T := TABOP^TNUM$

TR1 := TTRANS^E1$.PT;
NOMP^1$ := T^1$; NOMP^2$ := T^2$; NOMP^3$ := T^3$; NOMP^4$ :=T^4$;

WHILE TR1 <> NIL DO
  BEGIN
  DECODE(TR1^.TTYPE,TR1^.TNUM,T);
  T4^1$ := T^1$; T4^2$ :=T^2$; T4^3$ := T^3$; T4^4$ := T^4$;
  IF (T4 = NOMP) OR (T4 = '    ')
    THEN
    BEGIN
    NEW(TR2);
    WITH TR2^ DO
      BEGIN
      P := NOMP; M := TR1^.M; E := TR1^.E; NOEUD := 0;
      ENSREG := TR1^.ENSREG;
      SUIV := D;
      END;
    D := TR2;
    END;
  TR1 := TR1^.SUIV;
  END; (* WHILE TR1 *)

DELTA := D;
END; (* DELTA *)
```

```
PROCEDURE TRANSITIONS(VAR LE : PTRANS; VAR N : INTEGER;
                      VAR TTRANS : TETAT; VAR  QF : ENS);


(* TRIE LA LISTE LE, TRANSITION PAR TRANSITION                          *)
(*     M < NIVEAU : ON GARDE LA TRANSITION POUR TRI ULTERIEUR           *)
(*     M = NIVEAU : EST-CE UNE TRANSITION POSSIBLE POUR LE NOEUD N      *)
(*                      NON : ON LA REJETTE                             *)
(*                      OUI : ON LA GARDE. SI ELLE EST TERMINALE,       *)
(*                                ALORS ON LA MET DANS LEF              *)

VAR
   LER, PLE1, TR1, TR2 : PTRANS;

BEGIN

LER := NIL;
PLE1 := LE;

WHILE PLE1 <> NIL DO
  BEGIN

  IF PLE1^.M < NIVEAU
  THEN
  BEGIN
  TR1 := PLE1; PLE1 := PLE1^.SUIV;
  TR1^.SUIV := LER; LER := TR1;
  END (* M < NIVEAU *)

    ELSE IF PLE1^.M = NIVEAU
    THEN
    BEGIN
    TR1 := DELTA(PLE1^.E,N,TTRANS);
    WHILE TR1 <> NIL DO WITH TR1^ DO
      BEGIN
      M := M + NIVEAU;
      ENSREG := ENSREG * PLE1^.ENSREG;
      TR2 := TR1;
      TR1 := SUIV;
      IF TR2^.E IN QF THEN
        BEGIN
        TR2^.SUIV := LEF; LEF := TR2;
        END
        ELSE BEGIN
             TR2^.SUIV := LER; LER := TR2;
             END;
      END;
    TR1 := PLE1;
    PLE1 := PLE1^.SUIV;
    DISPEP(TR1^.PRED); DISPOSE(TR1);
    END (* M = NIVEAU *)

      ELSE
      BEGIN
      TR1 := PLE1; PLE1 := PLE1^.SUIV;
      DISPEP(TR1^.PRED); DISPOSE(TR1);
      END; (* M > NIVEAU *)

  END; (* WHILE PLE1 *)

LE := LER;
END; (* TRANSITIONS *)
```

```
FUNCTION TRANSARB(VAR PAX : INTEGER; VAR TTRANS : TETAT; VAR QF : ENS)
         : BOOLEAN;
VAR
     SRAMTRANS : BOOLEAN;
     TAMP, TR1, TR2 : PTRANS;

BEGIN

SRAMTRANS := FALSE;
TAMP := NIL; NIVEAU := NIVEAU + 1;

NEW(TR1);
WITH TR1^ DO
  BEGIN
  P := '     ';
  M := NIVEAU; E := 0; NOEUD := 0;
  ENSREG := °1..NREGLE§;
  PRED := NIL;
  SUIV := LE;
  END;
LE := TR1;
TRANSITIONS(LE,PAX,TTRANS,QF);

WITH ARBRE°PAX§ DO
  IF (TTYPE = 'O') AND (TFILS > 0) THEN
    BEGIN
    SRAMTRANS := (SRAMTRANS) OR (TRANSARB(TFILS,TTRANS,QF));
    IF (ARBRE°TFILS§.TFRERE > 0) AND (TNUM IN °23,32,33,34,35§) THEN
      SRAMTRANS := (SRAMTRANS) OR (TRANSARB(ARBRE°TFILS§.TFRERE,TTRANS,
    END; (* IF *)                                                    QF));

TR1 := LEF;
WHILE TR1 <> NIL DO WITH TR1^ DO
  BEGIN
  TR2 := TR1; TR1 := SUIV;
  IF M = NIVEAU THEN
    BEGIN
    IF COND(PAX, ENSREG) THEN
      BEGIN
      EXECUTER(PAX,ENSREG);
      ACTION(PAX,ENSREG);
      SRAMTRANS := TRUE;
      END; (* EXECUTION *)
    DISPEP(TR2^.PRED); DISPOSE(TR2);
    END (* M = NIVEAU *)
      ELSE
      BEGIN
      TR2^.SUIV := TAMP; TAMP := TR2;
      END; (* M <> NIVEAU *)
  END; (* WHILE TR1 *)
LEF := TAMP;

NIVEAU := NIVEAU - 1;
TRANSARB := SRAMTRANS;

END; (* TRANSARB *)
```

```
PROCEDURE TRANSFORMER(VAR TTRANS : TETAT; VAR  QF : ENS);
VAR
    RAMTRANSF : BOOLEAN;
BEGIN
RAMTRANSF := TRUE;
NIVEAU := 0;
LE := NIL; LEF := NIL;
WHILE RAMTRANSF DO
      BEGIN
      RAMTRANSF := FALSE;
      RAMTRANSF := (RAMTRANSF) OR (TRANSARB(PTDA,TTRANS,QF));
      END;
EDITARBRE;
END; (* TRANS *)
```

2 - EXEMPLE COMPLET

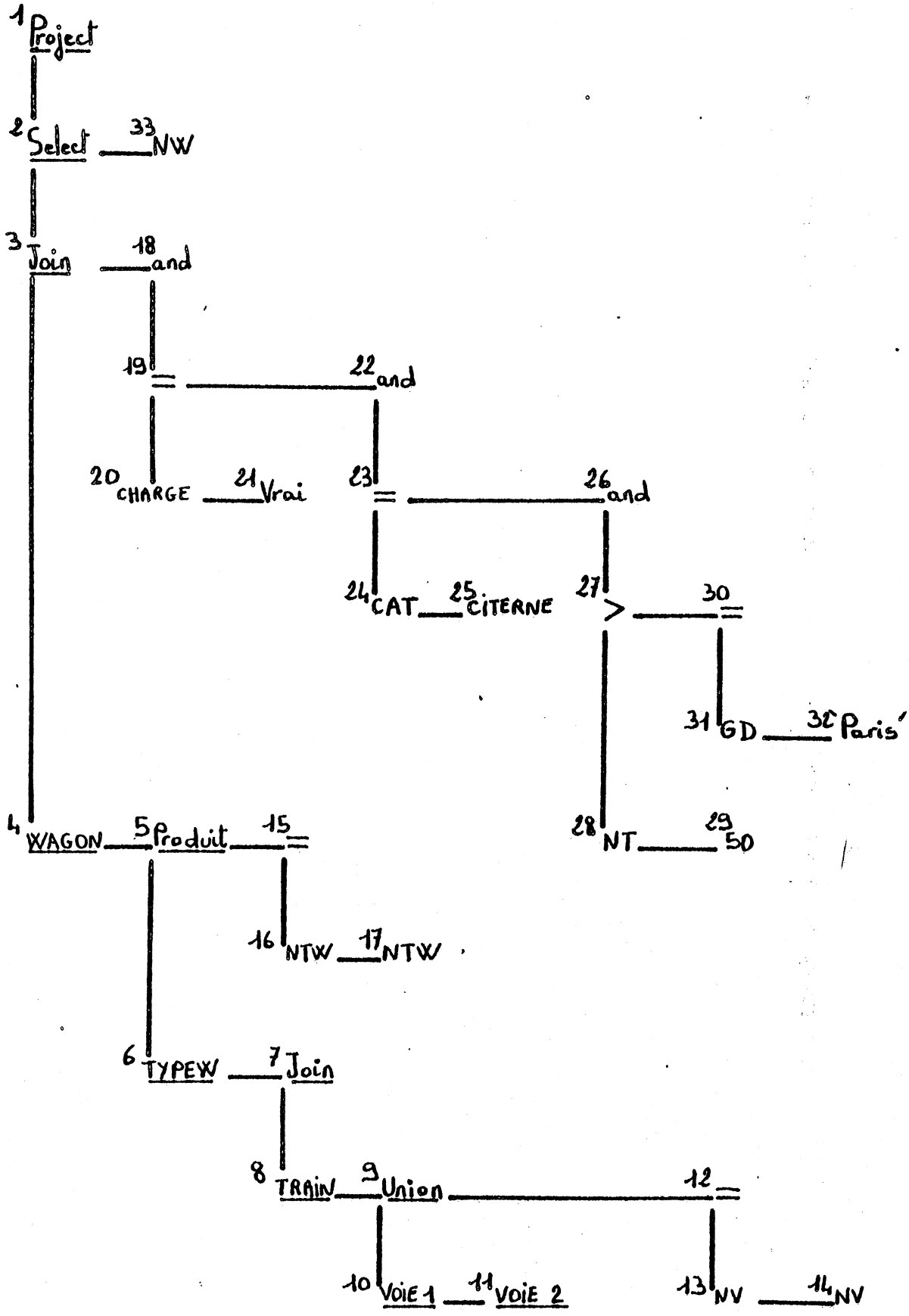
```
>PIP TI:=REGLES.DAT


&X1 /SELECT/ (&X2 /SELECT/ (&&X3)) ;
    &X1 / ET_EXPR (&X1 &X2) / (&&X3) ;
    ; ;
&X1 /SELECT/ (&X2 /PROJECT/ (&&X3)) ;
    &X2 (&X1 / LAT (&&X3) / (&&X3)) ;
    ; ;
&X1 /SELECT/ (&X2 /UNION/ (&&X3 &&X4)) ;
    &X2 (&X1 (&&X3) &X1 (&&X4));
    ; ;
&X1 /SELECT/ (&X2 /DIFFERENCE/ (&&X3 &&X4)) ;
    &X2 (&X1 (&&X3) &X1 (&&X4));
    ; ;
&X1 /SELECT/ (&X2 /INTERSECTION/ (&&X3 &&X4)) ;
    &X2 (&X1 (&&X3) &X1 (&&X4));
    ; ;
&X1 /SELECT/ (&X2 /JOIN/ (&&X3 &&X4));
    &X2 (&X1 / LAT (&&X3) EXPGAUCH (&X1 &&X3) / (&&X3)
          &X1 / LAT (&&X4) EXPDROIT (&X1 &&X4) / (&&X4)) ;
    / FACTEXPR (&X1 &&X3 &&X4) / ; ;
&X1 /SELECT/ (&X2 /PRODUIT/ (&&X3 &&X4));
    &X2 (&X1 / LAT (&&X3) EXPGAUCH (&X1 &&X3) / (&&X3)
          &X1 / LAT (&&X4) EXPDROIT (&X1 &&X4) / (&&X4)) ;
    / FACTEXPR (&X1 &&X3 &&X4) / ; ;
/
>


>
>RUN MAIN1
>

LE PROGRAMME DE TRANSFORMATION D'ARBRES  COMMENCE...

VOULEZ-VOUS ENTRER UN ARBRE A PARTIR D'UN FICHIER ?
              (OUI/NON)
>O
RELATIONS DE BASE : WAGON  TYPEW  TRAIN  VOIE1  VOIE
2
```

NUM	ATTRIBUTS	TYPES	RELATIONS
1	NW	INTEGER	WAGON
2	NV	INTEGER	WAGON
3	NTW	INTEGER	WAGON
4	CHARGE	BOOLEAN	WAGON
5	NTW	INTEGER	TYPEW
6	PV	INTEGER	TYPEW
7	CM	INTEGER	TYPEW
8	CAT	TEXTE	TYPEW
9	DESCR	TEXTE	TYPEW
10	NT	INTEGER	TRAIN
11	NV	INTEGER	TRAIN
12	NV	INTEGER	VOIE1
13	GD	TEXTE	VOIE1
14	GA	TEXTE	VOIE1
15	TY	TEXTE	VOIE1
16	NV	INTEGER	VOIE2
17	GD	TEXTE	VOIE2
18	GA	TEXTE	VOIE2
19	TY	TEXTE	VOIE2

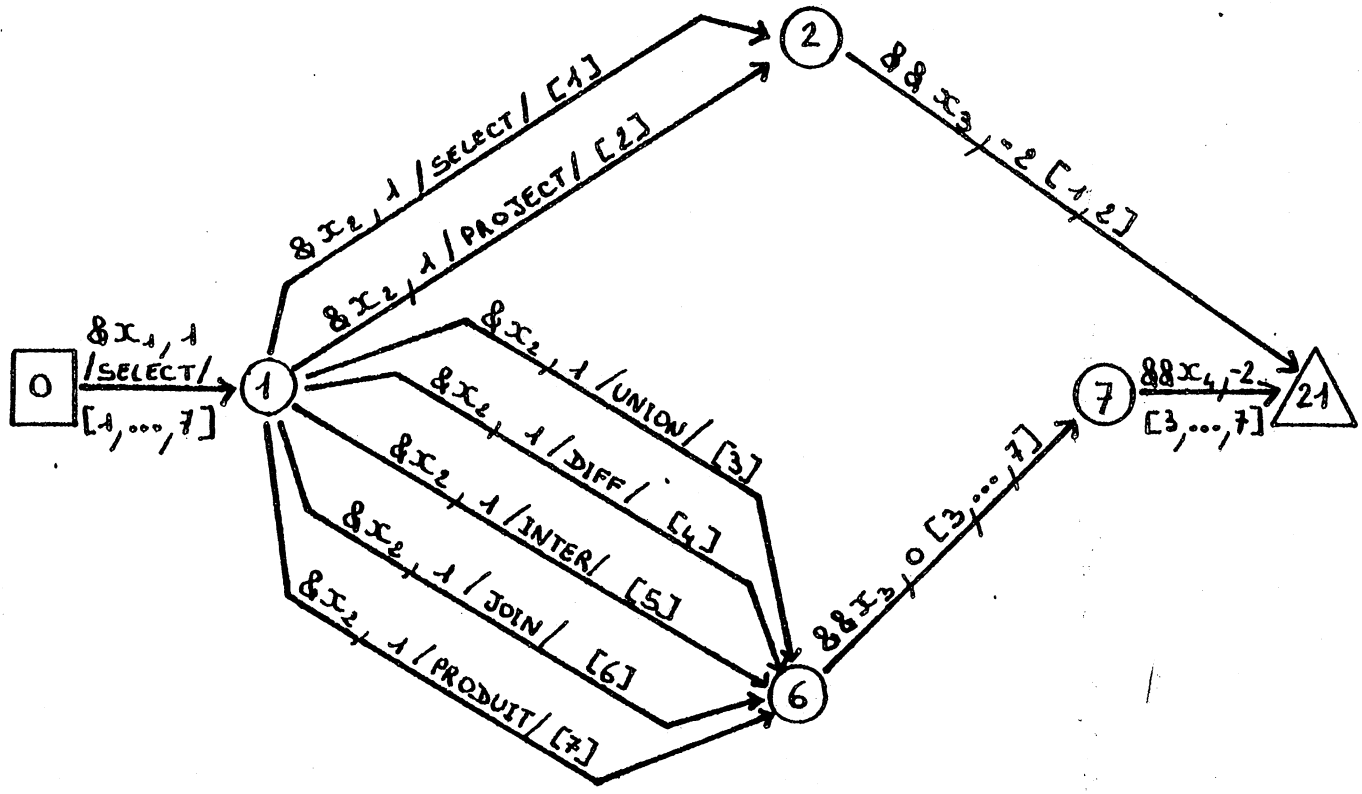
ARBRE, ENTREE NOEUD NO : 1

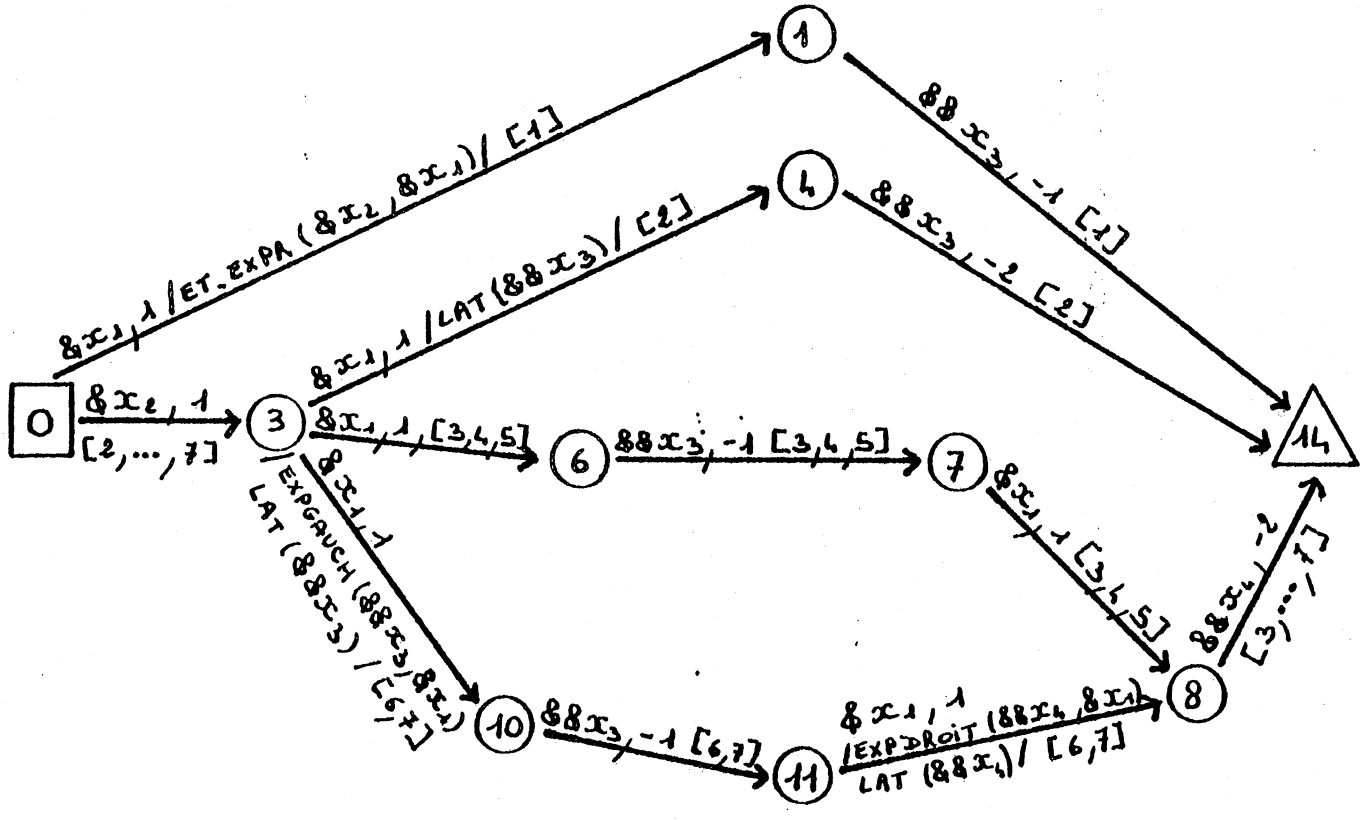
NO	NOEUD	FILS	FRERE	LAT
1	PROJECT	2	0	
2	SELECT	3	33	
3	JOIN	4	18	
4	WAGON	0	5	
5	PRODUIT	6	15	
6	TYPEW	0	7	
7	JOIN	8	0	
8	TRAIN	0	9	
9	UNION	10	12	
10	VOIE1	0	11	
11	VOIE2	0	0	
12	=	13	0	
13	NV	0	14	
14	NV	0	0	
15	=	16	0	
16	NTW	0	17	
17	NTW	0	0	
18	AND	19	0	
19	=	20	22	
20	CHARGE	0	21	
21	VRAI	0	0	
22	AND	23	0	
23	=	24	26	
24	CAT	0	25	
25	CITERNE	0	0	
26	AND	27	0	
27	>	28	30	
28	NT	0	29	
29	50	0	0	
30	=	31	0	
31	GD	0	32	
32	PARIS	0	0	
33	NW	0	0	

```
FAUT-IL INITIALISER L'OPTIMISATION ?
>0

                ARBRE, ENTREE NOEUD NO :     1

  NO   NOEUD     FILS FRERE    LAT
-------------------------------------------------------------------------------
   1   PROJECT     2      0 ( NW )
   2   SELECT      3     33 ( CHARGE NTW NV NW DESCR CAT CM PV GD GA TY NT NV
   3   JOIN        4     18 ( NV NT TY GA GD PV CM CAT DESCR NW NV NTW CHARGE
   4   WAGON       0      5 ( CHARGE NTW NV NW )
   5   PRODUIT     6     15 ( NV NT TY GA GD NTW PV CM CAT DESCR )
-------------------------------------------------------------------------------
   6   TYPEW       0      7 ( DESCR CAT CM PV NTW )
   7   JOIN        8      0 ( GD GA TY NT NV )
   8   TRAIN       0      9 ( NV NT )
   9   UNION      10     12 ( NV GD GA TY )
  10   VOIE1       0     11 ( TY GA GD NV )
-------------------------------------------------------------------------------
  11   VOIE2       0      0 ( TY GA GD NV )
  12   =          13      0 ( NV )
  13   NV          0     14
  14   NV          0      0
  15   =          16      0 ( NTW )
-------------------------------------------------------------------------------
  16   NTW         0     17
  17   NTW         0      0
  18   AND        19      0 ( CHARGE NT GD CAT )
  19   =          20     22
  20   CHARGE      0     21
-------------------------------------------------------------------------------
  21   VRAI        0      0
  22   AND        23      0
  23   =          24     26
  24   CAT         0     25
  25   CITERNE     0      0
-------------------------------------------------------------------------------
  26   AND        27      0
  27   >          28     30
  28   NT          0     29
  29   50          0      0
  30   =          31      0
-------------------------------------------------------------------------------
  31   GD          0     32
  32   PARIS       0      0
  33   NW          0      0
```

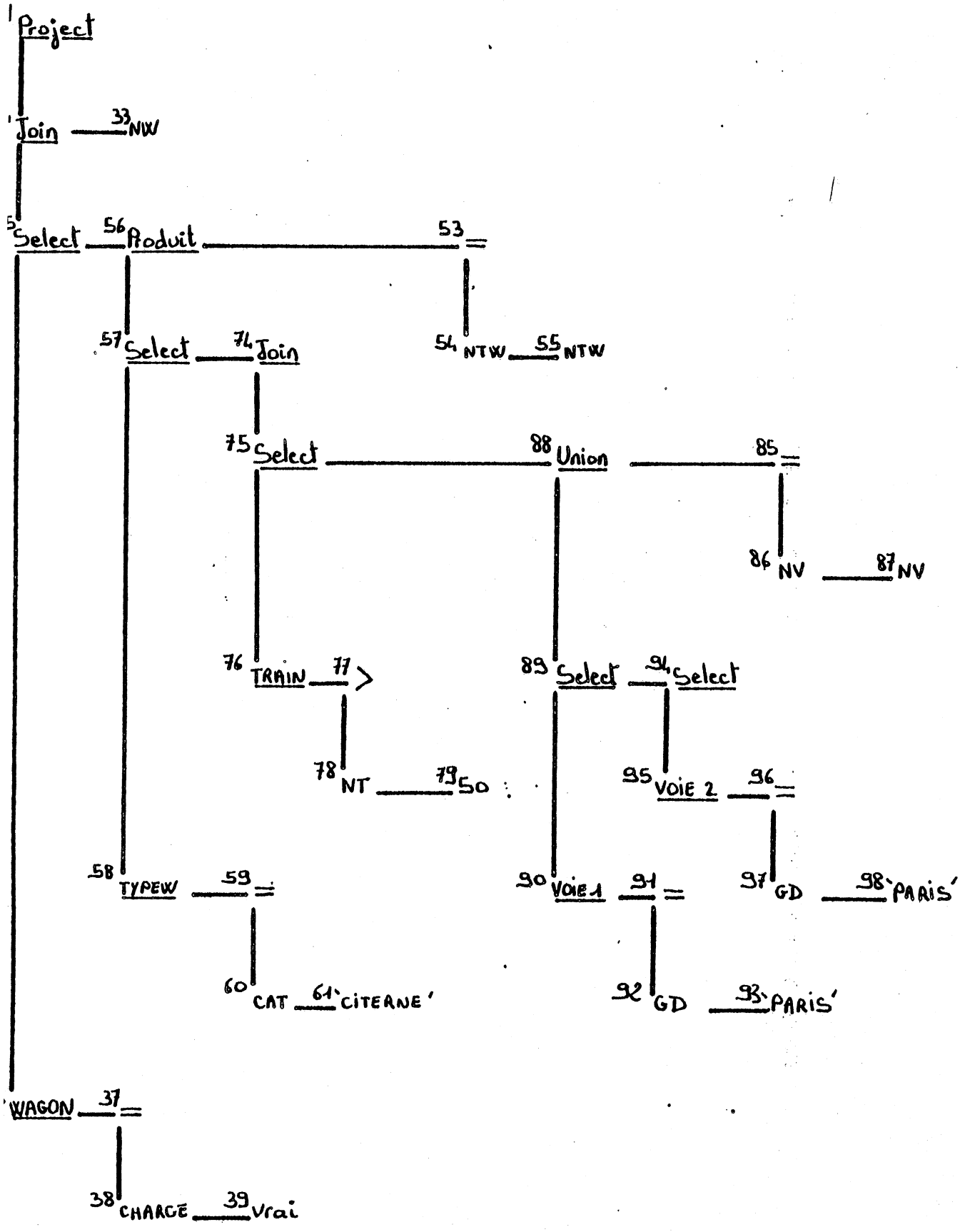

1 Project
2 Select
33 NW
3 Join
18 and
19 =
22 and
20 CHARGE
21 Vrai
23 =
26 and
24 CAT
25 CITERNE
27 >
30 =
31 GD
32 Paris
4 WAGON
5 Produit
15 =
28 NT
29 50
16 NTW
17 NTW
6 TYPEW
7 Join
8 TRAIN
9 Union
12 =
10 VOIE 1
11 VOIE 2
13 NV
14 NV

```
FAUT-IL ENTRER DES REGLES ?
>O
MODE DE LECTURE DES REGLES ?
          T °ERMINAL$  OU  F °ICHIER$
>F

VOULEZ-VOUS VERIFIER LA COMPILATION DES REGLES ?
>O
REGLES COMPILEES     <XI, YI, CI, AI> :

ARP REPRESENTANT LES PARTIES GAUCHE XI :

    Q =  (  0  1  2  6  7 21 )  ; QF =  ( 21 )
ETAT < NOMP  M  E  ENSREG      PREDICAT >
-------------------------------------------------------------------------
   0 < &X1   1  1  (  1  2  3  4  5  6  7 )   / SELECT / >
-------------------------------------------------------------------------
   1 < &X2   1  6  (  7 )   / PRODUIT / >
     < &X2   1  6  (  6 )   / JOIN / >
     < &X2   1  6  (  5 )   / INTERSEC / >
     < &X2   1  6  (  4 )   / DIFFEREN / >
     < &X2   1  6  (  3 )   / UNION / >
     < &X2   1  2  (  2 )   / PROJECT / >
     < &X2   1  2  (  1 )   / SELECT / >
-------------------------------------------------------------------------
   2 < &&X3 -2 21  (  1  2 )   /  / >
-------------------------------------------------------------------------
   6 < &&X3  0  7  (  3  4  5  6  7 )   /  / >
-------------------------------------------------------------------------
   7 < &&X4 -2 21  (  3  4  5  6  7 )   /  / >
-------------------------------------------------------------------------

ARP REPRESENTANT LES PARTIES DROITES YI :

    Q =  (  0  1  3  4  6  7  8 10 11 14 )  ; QF =  ( 14 )
ETAT < NOMP  M  E  ENSREG      PREDICAT >
-------------------------------------------------------------------------
   0 < &X2   1  3  (  2  3  4  5  6  7 )   /  / >
     < &X1   1  1  (  1 )   /  ET_EXPR( &X2 &X1 ) / >
-------------------------------------------------------------------------
   1 < &&X3 -1 14  (  1 )   /  / >
-------------------------------------------------------------------------
   3 < &X1   1 10  (  6  7 )   /  EXPGAUCH( &&X3 &X1 ) LAT( &&X3 ) / >
     < &X1   1  6  (  3  4  5 )   /  / >
     < &X1   1  4  (  2 )   /  LAT( &&X3 ) / >
-------------------------------------------------------------------------
   4 < &&X3 -2 14  (  2 )   /  / >
-------------------------------------------------------------------------
   6 < &&X3 -1  7  (  3  4  5 )   /  / >
-------------------------------------------------------------------------
   7 < &X1   1  8  (  3  4  5 )   /  / >
-------------------------------------------------------------------------
   8 < &&X4 -2 14  (  3  4  5  6  7 )   /  / >
-------------------------------------------------------------------------
  10 < &&X3 -1 11  (  6  7 )   /  / >
-------------------------------------------------------------------------
  11 < &X1   1  8  (  6  7 )   /  EXPDROIT( &&X4 &X1 ) LAT( &&X4 ) / >
-------------------------------------------------------------------------

CONDITIONS CI :

   6       /  FACTEXPR( &&X4 &&X3 &X1 ) /
   7       /  FACTEXPR( &&X4 &&X3 &X1 ) /

EFFETS DE BORD AI :
```


0
1
2
6
7
21
&x1, 1 /SELECT/ [1,...,7]
&x2, 1 /SELECT/ [1]
&x2, 1 /PROJECT/ [2]
&x2, 1 /UNION/ [3]
&x2, 1 /DIFF/ [4]
&x2, 1 /INTER/ [5]
&x2, 1 /JOIN/ [6]
&x2, 1 /PRODUIT/ [7]
&&x3, -2 [1,2]
&&x3, 0 [3,...,7]
&&x4, -2 [3,...,7]


0
1
3
4
6
7
8
10
11
14
&x1, 1 /ET_EXPA(&x2, &x1)/ [1]
&x1, 1 /LAT(&&x3)/ [2]
&x2, 1 [2,...,7]
&x1, 1 [3,4,5]
&&x3, -1 [3,4,5]
&x1, 1 [3,4,5]
&&x4, -2 [3,...,7]
&&x3, -1 [1]
&&x3, -2 [2]
&x1, 1 /EXPGAUCH(&&x3, &x1) LAT(&&x3)/ [6,7]
&&x3, -1 [6,7]
&x1, 1 /EXPDROIT(&&x4, &x1) LAT(&&x1)/ [6,7]

```
VOULEZ-VOUS LANCER L'ANALYSE DE L'ARBRE ?
>0

              ARBRE, ENTREE NOEUD NO :     1

  NO   NOEUD     FILS  FRERE    LAT
-------------------------------------------------------------------------------
   1   PROJECT    34      0  ( NW )
  33   NW          0      0
  34   JOIN       35     33  ( NT NV GD GA TY DESCR CAT CM PV CHARGE NTW NV NW
  35   SELECT     36     56  ( NW NV NTW CHARGE )
  36   WAGON       0     37  ( CHARGE NTW NV NW )
-------------------------------------------------------------------------------
  37   =          38      0  ( CHARGE )
  38   CHARGE      0     39
  39   VRAI        0      0
  53   =          54      0  ( NTW )
  54   NTW         0     55
-------------------------------------------------------------------------------
  55   NTW         0      0
  56   PRODUIT    57     53  ( NT NV GD GA TY DESCR CAT CM PV NTW )
  57   SELECT     58     74  ( NTW PV CM CAT DESCR )
  58   TYPEW       0     59  ( DESCR CAT CM PV NTW )
  59   =          60      0  ( CAT )
-------------------------------------------------------------------------------
  60   CAT         0     61
  61   CITERNE     0      0
  74   JOIN       75      0  ( TY GA GD NV NT )
  75   SELECT     76     88  ( NT NV )
  76   TRAIN       0     77  ( NV NT )
-------------------------------------------------------------------------------
  77   >          78      0  ( NT )
  78   NT          0     79
  79   50          0      0
  85   =          86      0  ( NV )
  86   NV          0     87
-------------------------------------------------------------------------------
  87   NV          0      0
  88   UNION      89     85  ( TY GA GD NV )
  89   SELECT     90     94  ( NV GD GA TY )
  90   VOIE1       0     91  ( TY GA GD NV )
  91   =          92      0  ( GD )
-------------------------------------------------------------------------------
  92   GD          0     93
  93   PARIS       0      0
  94   SELECT     95      0  ( NV GD GA TY )
  95   VOIE2       0     96  ( TY GA GD NV )
  96   =          97      0  ( GD )
-------------------------------------------------------------------------------
  97   GD          0     98
  98   PARIS       0      0


FIN DU PROGRAMME OPTIMISATION (MAIN1)
>
```


Project
Join 33 NW
5 Select 56 Produit 53 =
57 Select 74 Join
54 NTW 55 NTW
75 Select 88 Union 85 =
86 NV 87 NV
76 TRAIN 77 >
89 Select 94 Select
78 NT 79 50
95 VOIE 2 96 =
58 TYPEW 59 =
90 VOIE 1 91 =
97 GD 98 'PARIS'
60 CAT 61 'CITERNE'
92 GD 93 'PARIS'
WAGON 37 =
38 CHARGE 39 Vrai

BIBLIOGRAPHIE

BIBLIOGRAPHIE

(1) ADIBA (Michel) - un modèle relationnel et une architecture pour les systèmes de bases de données réparties. - Grenoble : IMAG. Thèse d'Etat : USMG : septembre 1978

(2) ADIBA (Michel), et autres. - Polyphème : un système de bases de données réparties. - Grenoble : IMAG/CII-HB. Rapport final du contrat IRIA-SIRIUS n° 77076, 1979.

(3) ADIBA (Michel), GRIFFITHS-SELINGER (Patricia). - "Access path selection in distributed database management systems". International Conference on database, Aberdeen, juillet 1980.

(4) ADIBA (Michel). - "Les bases de données relationnelles : centralisation ou répartition ?". - Grenoble : IMAG. Congrés AFCET, Nancy, novembre 1980.

(5) ADIBA (Michel), DELOBEL (Claude). - Les Bases de données relationnelles. - Grenoble : IMAG DUNOD Informatique : 1982 .

(6) ARMSTRONG (W.W.), DELOBEL (Claude). "Decompositions and Functional dependencies in relations". Transactions on Database systems ACM, Vol 5, n° 4

(7) ASTRAHAN (M.M.), KIM (W.), SCHKOLNICK (M.). - Evaluation of the system R access path selection mechanism. IBM Research Report RJ 2797, San Jose, Avril 1979 (Publié au Congrès IFIP 1980).

(8) AZROU (Fatma). - MIDEC : un algorithme de décomposition de requêtes dynamique, décentralisé. - Grenoble : IMAG. Rapport de DEA : INPG : juin 1981.

(9) BEERI (Catriel), RISSANEN (Jorma). - "Faithful representation of relational database schames". - San José : IBM research laratory.
Computer Science/Research report, 1 er octobre 1980.

(10) BENSAID (A). - Conception et réalisation d'un système d'éxécut réparti : SER.
Diplôme de fin d'études d'ingénieur, CERI, Alger, 1981.

(11) BING YAO (S), DE JONG (D). - "Evaluation of ratebase access pa ACM, SIGMOD, 1978.

(12) BING YAO (S), HEVNER (A.R). - "query processing in distributec database systems".
First Berkely Workshop on distributed data management, Août 19

(13) BING YAO (S). - "optimization of query avaluation algorithms". ACM, Transaction on database systems, Vol 4, n° 2, juin 1978.

(14) BOSC, CHAUFFAUT. - Evaluation de requêtes concernant des donne réparties dans SDD1.
SIRIUS, réf : LAN.i.002, 1978.

(15) CALECA (Jean-Yves). - Projet POLYPHEME. L'expression et la décomposition de transactions dans un système de bases de donnée réparties. - Grenoble : IMAG.
Thèse de 3° cycle : USMG/INPG : septembre 1978.

(16) CHAMBERLIN et autres. - "SEQUEL 2 - A unified approach date definition, manipulation and control".
IBM Journal of research and development, vol 20, n° 6, novembre 1976.

(17) CHAMPINE (G.A.). - "Six approaches to distributed databases". Datamation, vol 23, n° 5, mai 1977.

(18) CHANG (Philip yen-Tang), SMITH (John Miles). - "Optimizing the performance of a relational algebra database interface".
Communication de l'ACM, vol 18, n° 10, octobre 1975.

(19) CHU (W.W.), HURLEY (Paul). - "A model for optimal query processing for distributed databases."
COMPCON 79, San Francisco, février 1979.

(20) CHUPIN (Jean-Claude). - Répartition d'applications et de bases de données sur un réseau général d'ordinateurs. - Grenoble : IMAG.
Thèse d'Etat : USMG : octobre 1977.

(21) CODD (E.F.). - "A relational model of data for large shared data banks". - San José : IBM research laboratory.
Communication de l'ACM, vol 13, n° 6, juin 1970.

(22) CODD (E.F.). - "Relational completness of data bases sublanguages".
Data Base Systems, vol 6, Prentice Hall, 1971.

(23) COLLETTI (René), ESCULIER (Christian), GIBERT (Alain), Le BIHAN (Jean). - Les nouvelles bases de données.
Edité par la CENTI : Paris : 1981.

(24) COURTIN (Jacques), GRANDJEAN (Ernest), VEILLON (Gérard). - "PIAF : un système de manipulation de données en langues naturelles". - Grenoble : IMAG.
Colloque international de terminologie, juin 1976.

(25) COURTIN (Jacques). - Algorithmes pour le traitement interactif des langues naturelles. - Grenoble : IMAG.
Thèse d'Etat : USMG : octobre 1977.

(26) DELOBEL (Claude), PAIK (In Sup). - "A strategy for optimizing the distributed query processing." - Grenoble : IMAG.
First international conference on distributed computing systems, Haustville, Alabama, octobre 1979.

(27) DELOBEL (Claude), PAIK (In Sup). - L'optimisation des requêtes dans un environnement de bases de données réparties.
Grenoble : IMAG : juillet 1980.

(28) DEMOLOMBE (R). - A general semantic method for efficiently evaluating "AND" operators in relational DBMS.
Rapport interne ONERA-CERT : LBD 77-5 : mai 1977.

(29) DEMOLOMBE (R). - Principaux choix pour décomposer et évaluer l questions dans les SGBD relationnels.
Rapport interne ONERA-CERT : 1979.

(30) EPSTEIN (Robert), STONEBRAKER (Michaël), WONG (Eugene). - "Distributed query processing in a relational data basa system".
SIGMOD, mai 1978.

(31) EPSTEIN (Robert), STONEBRAKER (Michaël). - "Analysis of distributed data base processing strategies".
Very large data basas, Montréal, 1980.

(32) FAGIN (Ronald). - "Functional dependencies in a relational database and propositional logic". -
IBM Journal of research and development, novembre 1977.

(33) FERNANDEZ (Fernando). - Etudes d'algorithmes d'optimisations de requêtes dans les SGBD réparties. Mise en oeuvre de l'optimiseur DOPAGE. - Grenoble : IMAG
Rapport de DEA : USMG : septembre 1979.

(34) FERNANDEZ (Fernando), FERRAT (Lunès), N. GUYEN GIA TOAN, LEE (Y.J). - MICROBE : manuel de référence . - Grenoble : IMAG, Janvier 1982.

(35) FERNANDEZ (Fernando). - La micro-mémoire relationnelle (MIMER) un outil pour la construction de SGBD relationnel; projet MICR - Grenoble : IMAG.
Thèse de 3 è cycle : USMG : novembre 1981.

(36) FERRAT (Lunès). - Définition et traduction en arborescences algébriques d'un langage de bases de données réparties de haut niveau : MICROSEQUEL - Grenoble : IMAG.
Rapport de DEA : USMG/INPG : septembre 1980.

(37) FERRAT (Lunès). - Projet MICROBE : Guide utilisateur du langage MIQUEL. - Grenoble : IMAG.
Note interne : janvier 1981 (équipe MICROBE).

(38) FERRAT (Lunès), GALY (Henri), NGUYEN (Gia Toan). - "A high level user interface for a local network database system". - Grenoble : IMAG.
INFOCOM 82 Conference, Las Vegas, mars 1982.

(39) GOODMAN (Nathan), ROTHNIE (James). - "An overview of the preliminary design of SDD 1 : a system for distributed database".
2 nd Workshop distributed data bases, Berleclay, mai 1977.

(40) GOODMAN (Nathan), ROTHNIE (James). - "A survey of research and development in distributed data base management".
Very large data bases, Tokyo, 1977.

(41) GRIFFITHS-SELINGER (Patricia) et autres. - "Access paths selection in a relational data base management system". - San José : IBM research laboratory.
Research report for computer science : août 1979.

(42) HALL (P.A.V.). - "Optimization of simple expression in a relational data base system".
IBM Journal of research and development, vol 20, n° 3, mai 1977.

(43) HELD, KREPS, STONBRAKER (Michaël), WONG (Eugène). - "The design and implementation of INGRES."
ACM transactions on DBS, vol 1, n° 3, septembre 1976.

(44) JOLOBOFF (Vania). - Unification d'arborescences. Evaluation sémantique d'énoncés en langue naturelle. - Grenoble : IMAG.
Thèse de Docteur-Ingénieur : USMG/INPG : 8 septembre 1978.

(45) JOUVE (Mireille), PARENT (Christine), SPACCAPIETTRA (Stefano). "Principes des systèmes de gestion de bases de données réparties".
RAIRO-Informatique/Computer service, vol 14, n° 3, 1980.

(46) KIMBLETON (Stephen R). - "Computer communication network : approach, objectives, and performance considerations". Computer survey, vol 7, n° 3, septembre 1975.

(47) LEE YOON-JOON. - Conception et réalisation d'un ensemble d'opérateurs relationnels. Projet MICROBE. - Grenoble : IMAG. Rapport de DEA : INPG : septembre 1981.

(48) LOPEZ MEDINA (Julio). - Systèmes de transformation de ramifications paramètrées. Définitions et application. - Grenoble : IM Thèse de Docteur-Ingénieur : INPG : 25 juin 1978.

(49) LUX (Augustin). - Etudes d'un modèle abstrait pour une machine LISP et son implémentation. - Grenoble : IMAG. Thèse de 3 e cycle : USMG : mars 1977.

(50) MAHMOUD (S.A), RIOUDON (J.S), SHERIF (O). - "The ADD systems". 4 th international conference on Very large data bases, Berlin Septembre 1977.

(51) MAHMOUD (S.A), RIOUDON (J.S), TOTH (K.C.). - "Distributed data base partitionning and query processing". IFIP TC-2 Working conference on database architecture, Venise, 26-29 juin 1979.

(52) NGUYEN GIA TOAN. - URANUS : une approche relationnelle à la coopération de bases de données. - Grenoble : IMAG. Thèse de 3 e cycle : USMG : décembre 1977.

(53) NGUYEN GIA TOAN. - Optimisation de requêtes relationnelles sur des bases de données réparties. - Grenoble : IMAG. Document PP 13 : équipe POLYPHEME : USMG : 6 septembre 1978.

(54) NGUYEN GIA TOAN. - Représentation de l'arbre d'interprétation dans URANUS. - Grenoble : IMAG Projet POLYPHEME : note technique n° 35 : mars 1979.

(55) NGUYEN GIA TOAN - A unified method for query decomposition and shared information updating in distributed systems. - Grenoble : IMAG First International conference on distributed Computing systems, Hunstville, Alabama, Octobre 1979.

(56) PAIK (In Sup). - Définition et réalisation d'un ensemble d'opérateurs relationnels pour une machine POLYPHEME. - Grenoble : IMAG. Rapport de DEA : USMG/INPG : septembre 1978.

(57) PAIK (In Sup). - Une approche pour optimiser les traitements des requêtes dans un environnement de bases de données réparties. - Grenoble : IMAG
Thèse de 3 ° cycle : USMG/INPG : 12 février 1981.

(58) PARKER (Stott D.). - Some notes on distributed database optimization. - Grenoble : IMAG.
Projet POLYPHEME : Note technique n° 34, 12 décembre 1978.

(59) SEDILLOTS, SERGEANT (Guy). - "A protocol for distributed execution based on decentralized control technics". - Grenoble : IMAG.
COMPCON Fall 80, Washington, octobre 1980.

(60) SEGUIN (Jean). - Traitement distribué d'informations réparties dans les réseaux d'ordinateurs. - Grenoble : IMAG.
Thèse d'Etat : UMSG/INPG : 9 mars 1978.

(61) TODD (S). - Integrated architecture for transaction specification and optimization in relational database systems". -
UKSC Scientific Centre report n° 0085, novembre 1980.

(62) WINNINGER (Pascale). - Un optimiseur algébrique de requêtes dans MICROBE. - Grenoble : IMAG.
Rapport de DEA : INPG : septembre 1982.

(63) WONG (Eugene). - "Retriving dispersed data from SDD 1 : a system for distributed database".
2 nd Berkeley workshop on distributed data management and computer networks, mai 1977.

(64) WONG (Eugene), YOUSSEFI (K). - "Decomposition : a strategy for query processing".
ACM Transaction on DBS, vol 1, n° 3, septembre 1976.

AUTORISATION DE SOUTENANCE

VU les dispositions de l'article 3 de l'arrêté du 16 avril 1974,

VU le rapport de présentation de
Monsieur C. DELOBEL, Professeur

Monsieur Henri GALY

est autorisé à présenter une thèse en soutenance pour l'obtention du titre de DOCTEUR DE TROISIEME CYCLE, spécialité "Informatique".

Fait à Grenoble, le 05 mai 1983

Le Président de l'I.N.P.-G.

D. BLOCH
Président
de l'Institut National Polytechnique
de Grenoble

P.O. le Vice-Président,

E R R A T A :

p 23 (figure) : (CHARGE = vrai)

p 35 (FIGURE 1) : lire R 8 et non pas R 8 b

p 40 (exemple) : lire somme et non pas union tout au long de l'exemple
lire R 9 c et non pas R 5 c

67 : y_3 - Select (NT>50)

68 : dans la propriété de la somme et du produit, il y a eu inversion de lignes; il faut lire :

$$R + (S_1 * S_2 * \ldots * S_n) = (R + S_1) * (R + S_2) * \ldots * (R + S_n)$$

$$R * (S_1 + S_2 + \ldots + S_n) = (R * S_1) + (R * S_2) + \ldots + (R * S_n)$$

p 92 (ligne 9) : lire Institut d'Informatique et Mathématiques Appliquées de Grenoble

p 105 (EXEMPLE COMPLET): lire JOIN au lieu de SELECT dans le paramètre &x2

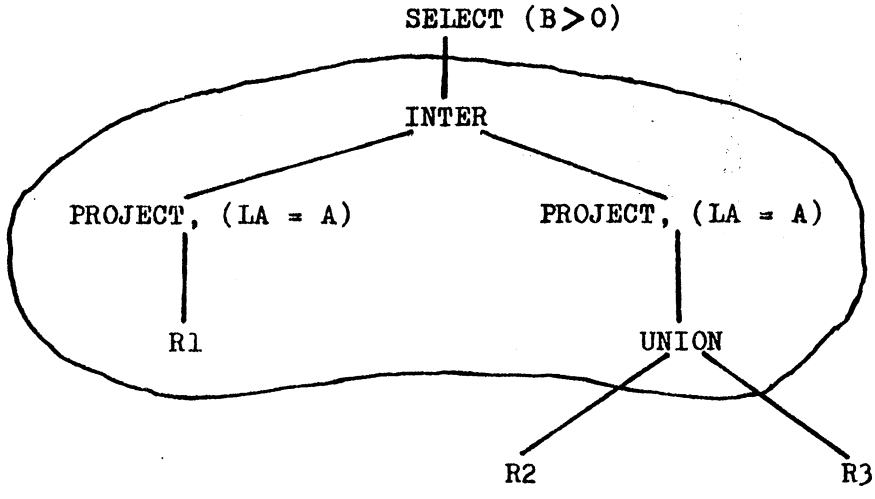
106 (figures) : Idem

115 (Nota Bene) : lire REPL (∅,X,Y) = ∅ et REPL (R,X,X) = R

p 116 (dans le sinon) : lire NOM (&x5) := 'JOIN' ;

p 118 (1ère figure) : lire JOIN et non pas PRODUIT
lire ATG et non pas ATTRG
lire ATD et non pas ATTRD

118 (2ème figure) : la figure complète est la suivante :



p 127 : lire L'ARP obtenu après CONST-ARP sur R est