Étude et réalisation d'une interface relationnelle pour un système de bases de données hiérarchiques multiples

Mohamed Nazir Hakim

THESE

Presentee a

L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Pour obtenir

LE TITRE DE DOCTEUR DE 3eme CYCLE

par

Mohamed Nazir HAKIM

ETUDE ET REALISATION D'UNE INTERFACE RELATIONNELLE POUR UN SYSTEME DE BASES DE DONNEES HIERARCHIQUES MULTIPLES

Soutenue le 5 Septembre1983 devant la commission d'Examen

JURY

Madame	G. SAUCIER	President
Messieurs	B. BAYLAC	
	P. LE DANOIS	Examinateurs
	G. MAZARE	
	G.T. NGUYEN	

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.C.	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 recherche
VACHAUD	Georges	Directeur de Recherche	JOUD	Jean Charles	Maître de recherche
ALLIBERT	Michel	Maître de recherche	KAMARINOS	Georges	Maître de recherche
ANSARA	Ibrahim	Maître de Recherche	KLEITZ	Michel	Maître de recherche
ARMAND	Michel	Maître de recherche	LANDAU	Ioan-Dore	Maître de recherche
BINDER	Gilbert		LASJAUNIAS	J.C.	
CARRE	René	Maître de recherche	MERMET	Jean	Maître de recherche
DAVID	René	Maître de recherche	MUNIER	Jacques	Maître de recherche
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 recherche
BISCONDI	Michel	Maître de recherche	LANCELOT	Francis	Maître de recherche
KOBYLANSKI	André	Maître de recherche	THEVENOT	François	Maître de recherche
LE COZE	Jean	Maître de recherche	TRAN MINH	Canh	Maître de recherche

.../...

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

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

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

E.N.S.Mines Saint Etienne

GUILHOT	Bernard	THOMAS	Gérard	DRIVER	Julien

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

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

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

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

E.N.S.H.G.

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

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

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

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

		CHARUEL	Robert		

C.E.N.G.

CADET	Jean	JOUVE	Hubert (LETI)	PERROUD	Paul
COEURE	Philippe (LETI)	NICOLAU	Yvan (LETI)	PEUZIN	Jean Claude (LETI
DELHAYE	Jean Marc (STT)	NIFENECKER	Hervé	TAIEB	Maurice
DUPUY	Michel (LETI)			VINCENDON	Marc

Laboratoires extérieurs :

C.N.E.T.

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

I.N.S.A. Lyon

		GAUBERT	C.		

****

A ma famille,

A mes enfants, qui, comme me disait jadis un ami,
sont bien ce que l'on fait de mieux
dans la vie.....

A Bouchra, pour tout, et le reste.

Cette étude a été menée au sein de la Société THOMSON-EFCIS (Société pour l'Etude et la Fabrication de Circuits Intégrés Spéciaux), dans le Service de Caractérisation, Modélisation et Simulation (C.M.S.) de la DIRECTION TECHNIQUE (D.T.).

Je remercie Monsieur J. BOREL, Directeur Technique de m'avoir permis d'effectuer ce travail.

Je suis reconnaissant à Madame G. SAUCIER, Professeur à l'INSTITUT NATIONAL POLYTECHNIQUE de GRENOBLE d'avoir accepté la direction de ma thèse, et la présidence du Jury. Ses discussions critiques et ses encouragements m'ont permis de progresser dans mon travail.

Je remercie Monsieur B. BAYLAC, Chef du Service de Caractérisation, Modélisation et Simulation (C.M.S.) de la Société (THOMSON-EFCIS), qui a bien voulu m'accueillir dans son service. Son étroite collaboration, ses nombreuses conversations et le vif intérêt qu'il m'a porté, ont été un facteur important, permettant la poursuite de mes travaux, et pour avoir accepté de faire partie du Jury.

Je suis très honoré que Monsieur G. MAZARE, Professeur à l'INSTITUT NATIONAL DE POLYTECHNIQUE de GRENOBLE, et Monsieur G.T. NGUYEN, Ingénieur de l'INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET AUTOMATIQUE (I.N.R.I.A.), aient accepté d'examiner ce travail et de faire partie du Jury.

Je suis particulièrement reconnaissant à Monsieur P. LE DANOIS, Ingénieur à THOMSON-EFCIS de m'avoir guidé au cours de cette recherche. Cette thèse doit beaucoup à ses relectures détaillées, et généralement à ses conseils.

Je le remercie également de m'avoir fait le plaisir de participer à ce Jury.

Mes remerciements à tous les membres du groupe C.M.S. : Messieurs N. BALLAY, P.-A. BRUNEL, B. CIALDELLA, J.-L. CZUBA, A. EL HENNAWY, G. MORIN, J.-P. MORIN et B. RAUBER, ainsi que Madame M. BONGIBAULT, Mademoiselle M.-L. SORDAGE et Monsieur J.-C. BERTHOLET qui m'ont tous aidé à une occasion ou à une autre, avec beaucoup de gentillesse et qui ont su créer une ambiance particulièrement sympathique.

Je remercie enfin Mesdames A. LOYAU, F. GUYON et S. ROCHE a qui je dois la réalisation de ce Mémoire. Je leur exprime encore mes remerciements pour le soin qu'elles ont apporté à la frappe de ce Mémoire.

TABLE DES MATIERES

CHAPITRE 2 : LANGAGE DE REQUETE

CHAPITRE 3 : OPTIMISATION DES REQUETES

CHAPITRE 4 : LE COMPILATEUR

CHAPITRE 0

INTRODUCTION

Les systèmes de gestion des bases de données (SGBD) sont considérés parmi les développements les plus importants en informatique, ces dernières années.

Les systèmes existants commercialisés sont basés sur un des trois types suivants : relationnel, réseau et hiérarchique.

Plusieurs organisations ont développé indépendamment leur propre base de données sur leurs propres ordinateurs, basée sur un de ces trois modèles, ayant des modèles de base, et des schémas d'accès adaptés à leurs besoins.

Chaque type de SGBD a son propre schéma, sa méthode d'accès, son degré d'efficacité, sa sécurité des informations et ses types d'opérateurs utilisés lors du traitement.

Souvent, différentes bases de données peuvent contenir les mêmes données, mais leurs structures et leurs schémas diffèrent.

Les problèmes qui se posent alors sont la coopération de plusieurs bases de données hétérogènes (ou homogènes) et l'interrogation des informations stockées dans ces bases.

Pour résoudre ces problèmes deux approches peuvent être utlisées :

- la première est de convertir et de transférer tous ces types de bases dans un même système (SHL 76, DAT 77) ;
- la deuxième est de garder les bases de données originales et d'implanter une interface permettant la manipulation des données entre les différents modèles sans aucun changement dans les schémas de base (ADT 78, CAR 80, DAT 77).

Notre approche peut être classée dans la deuxième catégorie.

La différence principale entre notre approche et celles déjà effectuées est que nous bénéficions des facilités proposées par les bases de données relationnelles.

Un modèle global permet de déterminer un schéma commun à chaque base, ces dernières peuvent garder leur propre modèle de données, mais participent au modèle global. Un langage de haut niveau non-procédural peut être utilisé comme langage d'interrogation.

D'autre part, plusieurs types de logiciels ont été proposés, il n'existe donc pas de logiciel universel pour tous les utilisateurs.

Nous proposons une interface permettant la manipulation en modèle "relationnel" d'une base de données gérée par le SGBD IMAGE.

L'interface est constituée de deux modules :

* Le premier module permet à l'utilisateur d'avoir une vue relationnelle de la base de données et ceci grâce à la traduction de la structure d'une base IMAGE.
  On définit un modèle théorique qui rassemble la plupart des concepts retrouvés dans les deux bases.

* Le deuxième module traduit les requêtes relationnelles de l'utilisateur (écrites en langage SQL) dans le langage SGBD IMAGE.
  C'est au moyen d'un programme qui permet de retrouver les chemins d'accès que doit suivre IMAGE pour retrouver les données concernées par la requête relationnelle, que se fait la constitution de la requête IMAGE (QUERY).

Cette étude définit une couche de logiciel traitant l'ensemble de bases locales dans le cadre de bases de données multiples.

Mais, pour que cette méthode passe du stade normatif au stade opératoire, il est indispensable qu'elle s'appuie :

- sur des concepts précis intégrés dans des modèles
- sur des langages qui permettent d'exprimer clairement et rigoureusement les résultats de la conception
- sur des outils qui facilitent la réalisaion d'interfaces qui manipulent les données.

Ces trois éléments sont intimement liés dans notre travail. En effet, le modèle d'un système de gestion de bases de données est un ensemble restreint de concepts applicables à une organisation déterminée. Cet ensemble limité et interdépendant de concepts constitue le concept de base de données. Pour le schéma conceptuel on détermine un langage de requêtes permettant d'interroger les informations stockées dans les bases. Enfin une méthode est d'autant plus efficace qu'elle comporte des outils appropriés pour résoudre complètement les problèmes qui sont formulés.

Pour chacune des premières étapes, nous présentons à la fois un schéma conceptuel qui définit les structures de stockage des informations, un langage de manipulation des données, et enfin, nous présentons un compilateur développé pour réaliser les deux dernières étapes.

Avant de créer et de faire fonctionner un logiciel dans un contexte de bases de données multiples, un important travail de conception doit être réalisé. Ce travail nous a conduit au découpage du travail en trois étapes successives.

Pour construire un système d'information qui interroge les données structurées sur plusieurs bases, on va définir trois étapes qui ne découlent pas d'un découpage arbitraire du processus de traitement. Au contraire chaque étape est structurée et définie d'une façon optimale (si on la traite avec les étapes suivantes).

1ère partie : on va organiser les trois premières étapes comme (figure 1) :

- La conception des données ;
- Les traitements de données ;
- L'optimisation des traitements.

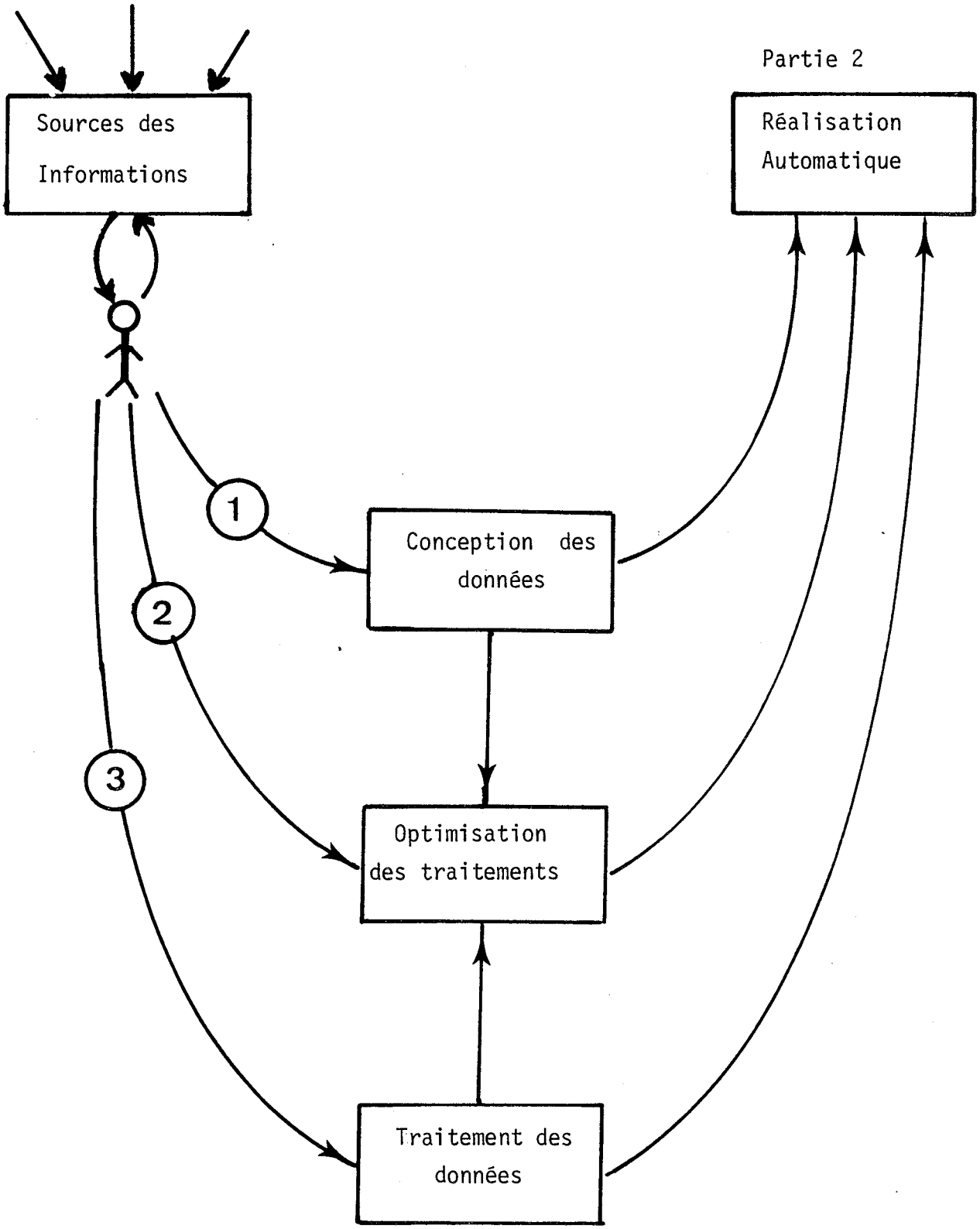


Fig. 1 - Etapes de la 1ère partie "Etude"

1°) Au cours de la première étape nous appelons la conception des données, nous abordons la consitution d'une interface entre le modèle relationnel et le SGBD IMAGE. Ceci pose les questions suivantes :

- Quelles sont les particularités de la base de données IMAGE ?
- Quelle sont les particularités d'une base de données relationnelle ?
- De quelle façon traduit l'interface, au mieux quelle est le schéma conceptuel de modèle IMREL qui définit cette interface ?

2°) Au cours de la deuxième étape (traitements des données), on traduit les requêtes relationnelles de l'utilisateur dans le langage SGBD IMAGE avec un temps de réponse acceptable, en tenant compte des contraintes suivantes :

- Les différents types de langages de données en particulier QUERY et SQL ;
- Les caractéristiques et les spécifications du langage de de requête utilisé.

3°) La troisième étape (optimisation des traitements), en étudiant une approche pour optimiser les traitements des requêtes, selon un compromis entre l'encombrement des attributs dans chaque base et les conditions de requête.

Dans cette étape on va définir les différentes techniques utilisées pour optimiser les traitements de requêtes, ensuite on va établir un modèle mathématique pour évaluer les coûts minimaux de l'opérateur Joindre ou P-Joindre.

Dans la 2ème partie nous parlons d'un domaine d'applications précis : la réalisation d'une interface relationnelle pour un système de bases de données hiérarchiques multiples. On va suivre une démarche semblable à celle de la réalisation d'un compilateur de compilateur (figure 2), et qui peut être réalisé en définissant l'aspect description et la réalisation d'un compilateur de requêtes. A cet effet, on va réaliser un analyseur sémantique qui traite les structures des bases et des requêtes pour avoir un traitement rapide.

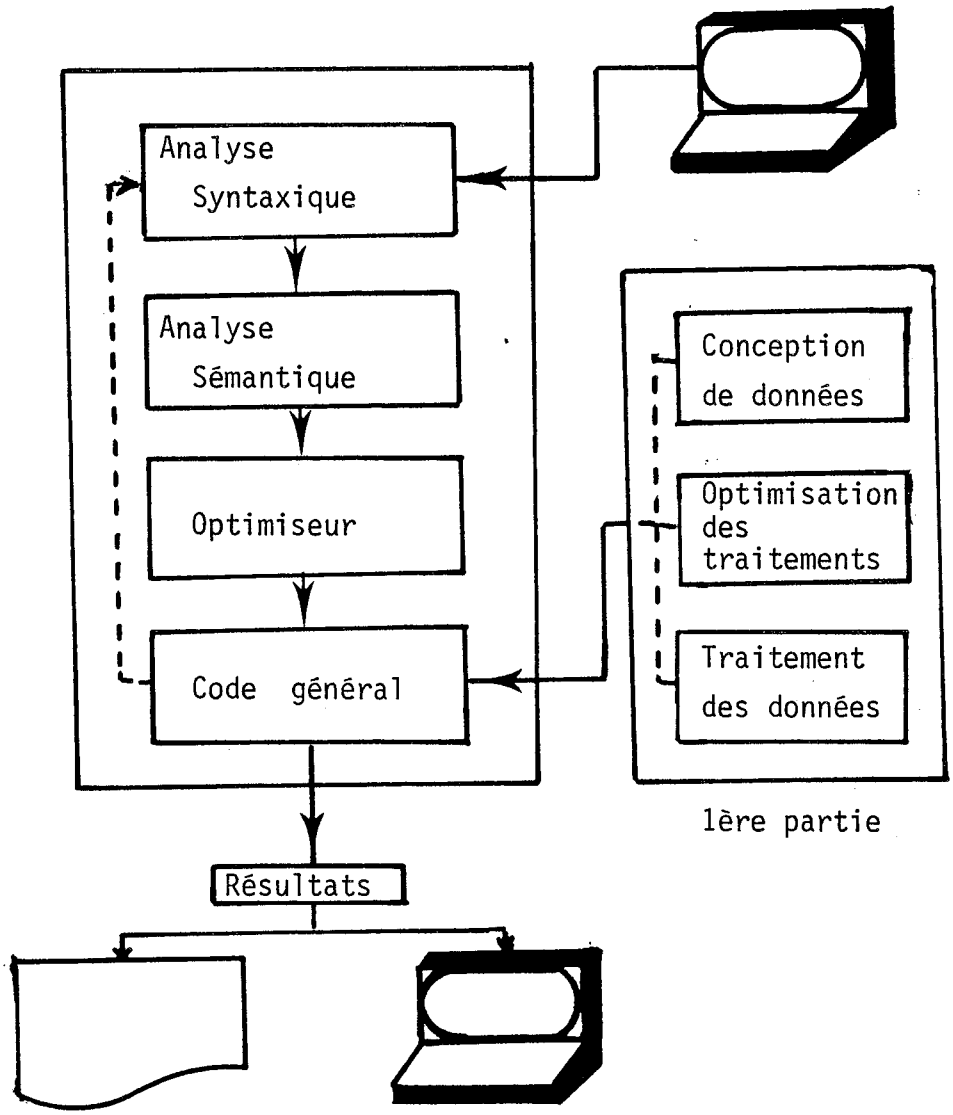


Fig. 2 Etape de la 2ème partie "Réalisation"

BIBLIOGRAPHIE

(ADI 78) M. ADIBA, D. PORTAL
"A cooperation system for heterogeneous data base management system"
Infor system 3-3, 1978, 205-219.

(CAR 80) A. CARDENAS, M. PIRAHESH
"Data base communication in a heterogeneous data base management system"
Inform system 5-1, 1980, 55-79.

(DAT 77) C.J. DATE
"An introduction to data base system"
2rd addison Wesley Reading, Mars 1977.

(SHL 76) N. SHU, V. LUM, B. HUUSEL
"An approach to data migration in computer networks"
IBM Repport 1703, 1976.

CHAPITRE I

MODELISATION DES BASES DE DONNEES

I - 1. OBJECTIF

Ce chapitre constitue un rappel des principales propriétés de la structuration des bases de données, qui consistent à obtenir une représentation claire, explicite, cohérante et condensée de la variété des types de données.

Cette étape a fait l'objet de nombreuses publications, en particulier (McG 81)(YAM 81)(COD 82).

D'après la présentation de la structure et du fonctionnement qualitatif des bases de données, nous allons exprimer la solution appliquée aux systèmes IMAGE et RELATIONNEL. Ce type de représentation est possible grâce à des considérations d'ordre technique, qui seront introduites dans les étapes suivantes de la conception. Enfin la solution obtenue, que l'on désigne par le terme type IMAGE-RELATIONNEL (IMREL) doit s'efforcer d'intégrer les différentes requêtes des utilisateurs qui désirent s'informer sur la réalité organisationnelle décrite.

La démarche de ce chapitre peut être atteint d'emblée. Au départ, on va être confronté à des faits qui se présentent dans les deux grandes classes de modèles des bases de données, leur diversité et leur apparente incohérence d'autre part. Ces faits nous ont conduit à concevoir une interface entre les deux modèles. En conclusion, on va représenter notre structure d'une façon hiérarchique, modélisée sous une forme mathématique et qui autorise l'application des opérations propres aux modèles relationnels.

I - 2. MODELES DE DONNEES

L'objectif d'un modèle de données est de permettre à un utilisateur de décrire, de désigner et de manipuler les données (COD 82).

La structure de données représente le résultat obtenu en employant un modèle de données et la concentration optimale de toutes les informations nécessaires au bon fonctionnement d'une organisation (TOM 80).

Nous pouvons classer les différents modèles existant en deux catégories : les modèles basés sur les types d'enregistrements, et les modèles conceptuels.

I - 2.1. Modèles basés sur les types d'enregistrement

- Modèles en Réseau

Cette approche est adaptée par les systèmes IDS, CODASYL, ADABAS, SOCRATE, IDMS, les noeuds du réseau sont des occurences individuelles d'enregistrements. L'occurece d'enregistrements est l'unité d'accès. Un réseau représente une structure plus générale qu'une hiérarchie car un noeud donné peut avoir n'importe quel nombre de supérieurs immédiats, ceci parmet de représenter une correspondance multiple en utilisant un lien direct (DEL 82).

- L'approche hiérarchique

Ce modèle qui permet à l'utilisateur de voir les structures des données sera comparée d'une ou plusieurs hiérarchies sous forme d'un graphe arborescent dont les noeuds correspondent aux classes d'objets et les arcs entre deux noeuds aux associations. Il est fondamental dans cette approche que toute occurence d'un segment donné n'a de sens que lorsqu'elle est vue dans la définition de schéma conceptuel des bases. En effet, aucune occurence de segment ne peut exister sans son "parent", excepté le segment "racine" qui par définition n'a pas de parents (McG 81).

Ainsi, pour retrouver l'occurence particulière d'un sujet, l'utilisateur doit fournir non seulement le sujet qui l'intéresse mais aussi il doit utiliser la notion de "chemin hiérarchique" qui va permettre de descendre de la racine au segment le plus bas dans la hiérarchie.

Cette approche est caractérisée par plusieurs systèmes parmi lesquels IMS, SYSTEM 2000, TOTAL, et IMAGE (HIP 81)(FU 76).

I - 3. PARTICULARITE DES BASES DE DONNEES "IMAGE"

En général on peut dire que IMAGE est un Logiciel complet qui assure le traitement de données avec une minimisation de la place nécessaire au stockage de ces données.

Pour bien comprendre le système IMAGE nous allons expliquer la philosophie

en exposant dans un premier temps, la présentation des données, à la suite de quoi on définira la base de données IMAGE.

Enfin, on parlera du schéma général du fonctionnement d'IMAGE.

I - 3.1. Présentation des données

Le processus développé dans le système de gestion de base de données IMAGE construit directement le chemin d'accès séquentiel vers les fichiers qui contiennent les données stockées.

Exemple

Nous supposons que les données dans IMAGE se rapportent à la qualité de tests, on doit conserver certaines données que l'on veut utiliser pendant un certain temps telles que

'Type de technologie, Provenance, Masque, Lots, Plaque, Date, Motif, ...'

Nous allons stocker les données dans quatre fichiers un de ces fichiers va être modélisé sous la forme suivante :

N TECH	I TECH	PROVN	MASQUE	LOT	TRANCHE	DATE	--- ---

IMAGE stocke les nouvelles données dans l'enregistrement qui est vide ou dans celui qui vient juste après le dernier enregistrement occupé.

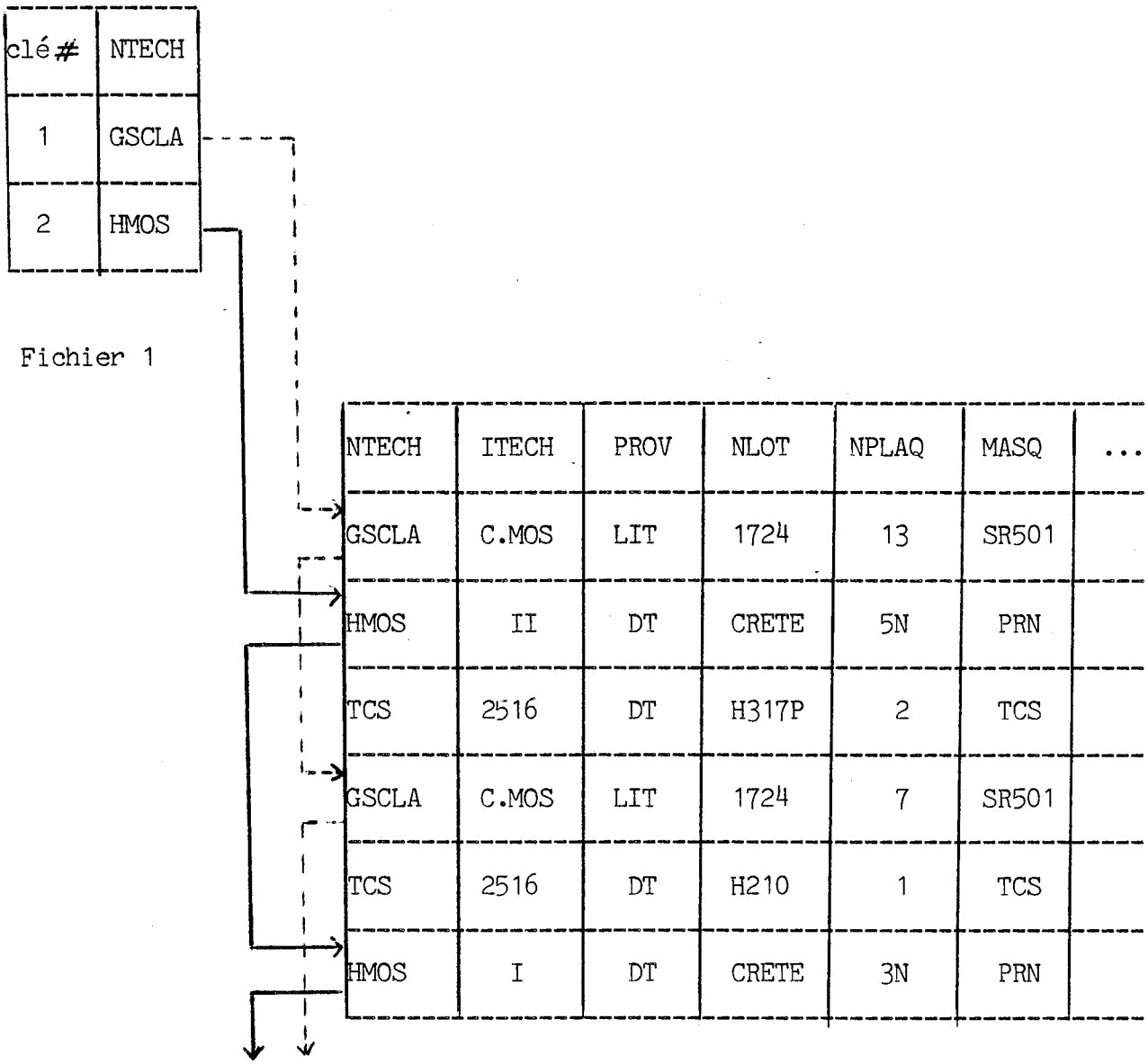
Pour chercher les données dont on a besoin dans certaines conditions, IMAGE parcourt séquentiellement des listes chaînées.

On peut chaîner l'ensemble des données de la manière suivante :

N TECH	INTECH	PROV	NLOT	TRANCHE	MASQ	DATE
GSCLA	C.MOS	LIT	1724	13	SR501	02/04
HMOS	II	DT	CRETE	5N	PRN	04/04
TCS	2516	DT	H317P	2	TCS	9/04
GSCLA	C.MOS	LIT	1724	7	SR501	15/04
TCS	2516	DT	H210	1	TCS	23/04
HMOS	I	DT	CRETE	3N	PRN	27/04
GSCLA	N.MOS	LIT	1067	1	SR603	30/04
HMOS	II	DT	CRETE	9N	PRN	01/05

La première difficulté est de trouver la première occurence pour avoir la possibilité d'entrer dans le fichier si les données stockées sont rangées dans un grand fichier, il nous faudra un temps relativement long pour trouver la première occurence (acces-séquentiel).

IMAGE propose dans ce cas de construire un autre fichier contenant les premières occurences de chaque type de données. On peut pointer vers la première occurence de chaque type de la façon suivante :



clé #	NTECH
1	GSCLA
2	HMOS

Fichier 1

NTECH	ITECH	PROV	NLOT	NPLAQ	MASQ	...
GSCLA	C.MOS	LIT	1724	13	SR501	
HMOS	II	DT	CRETE	5N	PRN	
TCS	2516	DT	H317P	2	TCS	
GSCLA	C.MOS	LIT	1724	7	SR501	
TCS	2516	DT	H210	1	TCS	
HMOS	I	DT	CRETE	3N	PRN	

Fichier 2

Avec la possibilité d'appliquer un algorithme de Hash-Code pour ranger l'entrée dans le fichier 1, cette méthode permet d'obtenir un accès aux données plus rapide. On trouve que la construction du fichier contenant les adresses des premières occurrences est efficace et nécessaire si les données stockées dans la base de données sont des nombres d'occurences limites, mais dans le cas contraire la méthode n'est plus efficace, dans ce cas on contruit un fichier maître pour faciliter l'accès aux données (voir le définite de la base IMAGE);

De cette manière de chaque fichier maître on peut accéder à plusieurs ensembles de données, et à chaque ensemble de donnes on peut associer plusieurs fichiers maîtres (figure 3).

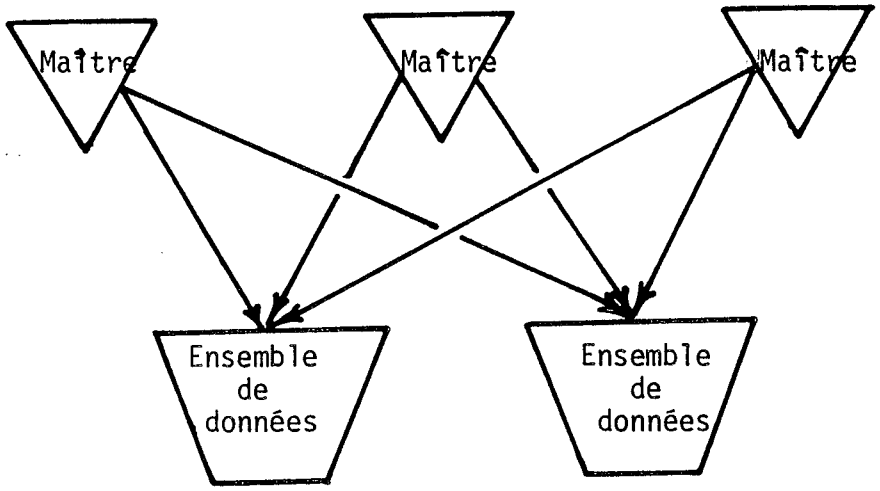


Fig. 3 Plusieurs fichiers maitres associer à plusieurs ensembles de données

I - 3.2. Définition de la base de données "IMAGE"

1.3.2.1. Introduction

la base de données IMAGE est l'ensemble des fichiers liés à un ou plusieurs fichiers maîtres : le fichier structure (racine), les fichiers données détails et les fichiers maîtres eux mêmes.

La (figure 4) présente le système de gestion de base de données "IMAGE/1000".

Ce système de données présente les avantages suivants :

- les programmes appliqués sur ces fichiers sont indépendants ;
- la rapidité d'accès aux données ;
- la sécurité des données à plusieurs niveaux ;
- possibilité de minimiser l'accès aux données (utilitaires) ;
- facilitée de manipulation des fichiers-maîtres et des données (utilitaires).

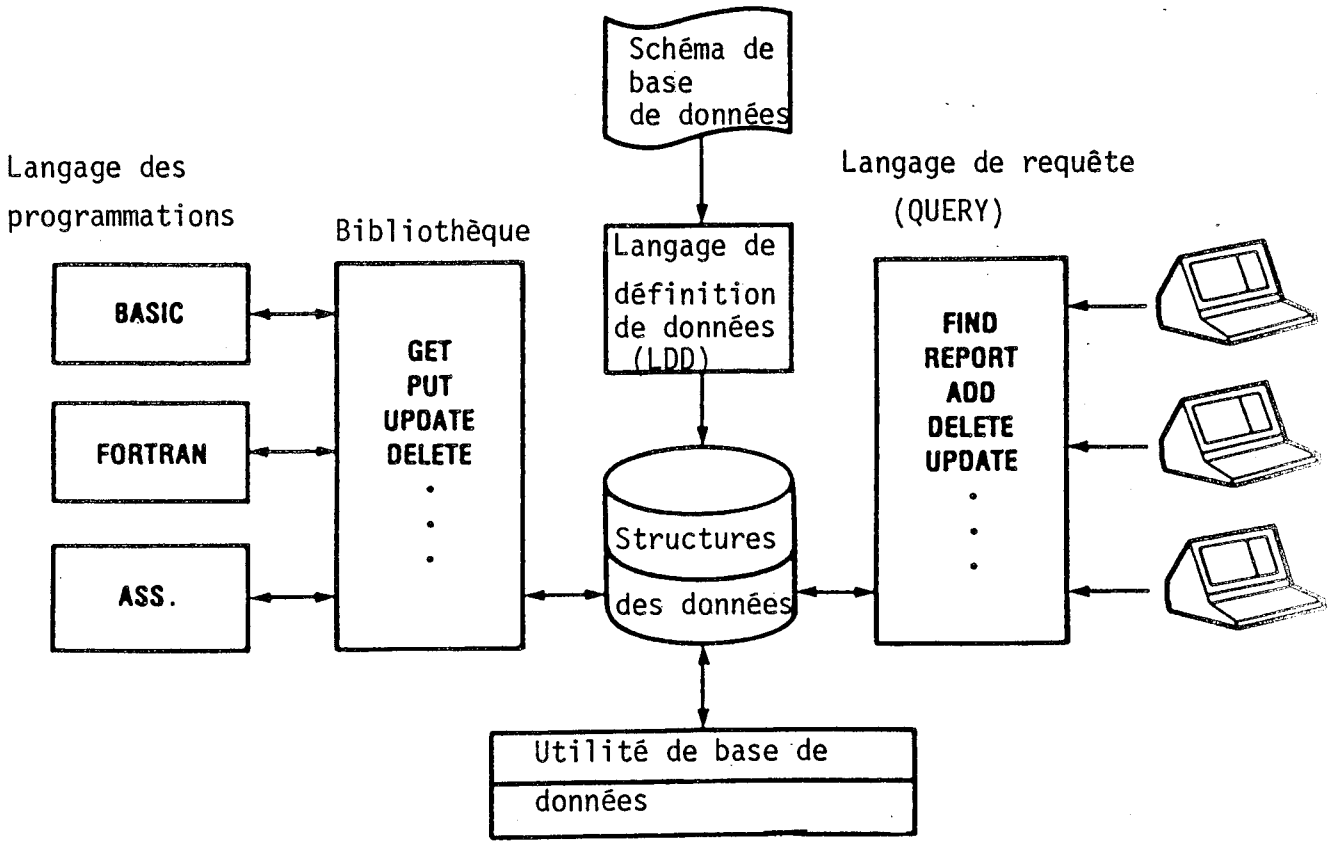


Fig. 4 Schéma général de SGBD "IMAGE"

1.3.2.2. Conception de la base de données IMAGE

La structure des données dans un fichier ne se trouve pas dans le fichier.

La caractérisation des éléments qui constituent la base de données IMAGE/1000 est faite dans plusieurs ensembles de données "Data set" (on peut avoir jusqu'à 50 ensembles).

La taille d'un seul fichier ne doit pas dépasser la capacité de son support. Pour accéder à l'ensemble des données on propose d'utiliser la clé d'item qui permet de trouver l'adresse d'entrée dans le fichier données. La valeur de ces cléfs constituent le fichier maître (figure 5)

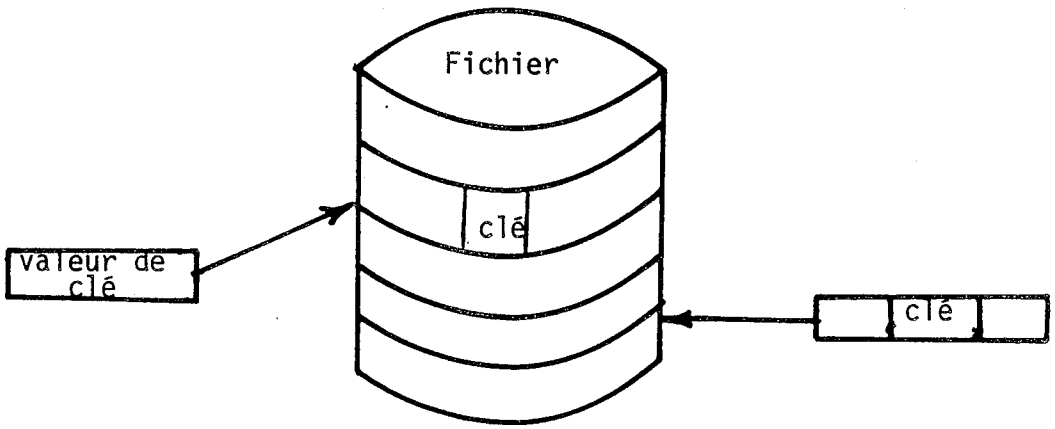


Fig. 5 Deux fichiers maitres sur un fichier "ensemble de données"

1.3.2.2.1. Fichier Maître

Le fichier maître est un tableau d'adresses et d'indices. Il contient l'adresse du premier item défini par la clé d'entrée dans l'ensemble de données (ED) et contient aussi les nombres d'occurrences de cet item dans l'ED.

Exemple

Soit la technologie comme clé d'entrée, on va montrer le tableau de fichier maître technologie.

Fichier maître technologie

item	adresse	indice
HMOS I	16	7
HMOS II	22	5

on va trouver la 1ère technologie de type HMOS à l'adresse 16, elle est répétée 7 fois dans le ED.

Avec ce type de fichier maître on peut chainer en avant et en arrière.

Il existe deux types de fichier maître :

* Un maître Manuel qui impose la gestion manuelle des chaînes d'accès
* Un maître automatique qui génère automatiquement toutes les chaînes d'accès.

L'indice qui fixe la répétition d'un item n'est pas calculé sur un seul ED, mais il tient compte de cette répétition sur tous les ED.

Les figures suivantes nous montrent quelques cas de types de fichier maître:

a) un fichier maître est construit sur deux ED exemple (figure 6)

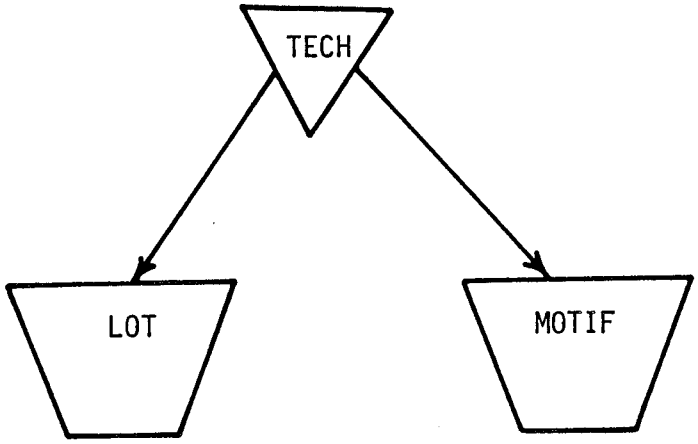


Fig. 6

b) deux fichiers maître sur un seul ED (figure 7)

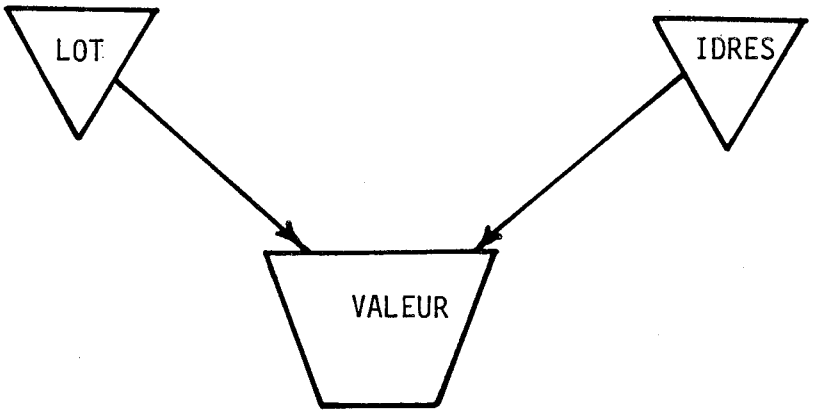


Fig. 7

c) deux fichiers maître sur deux ED (figure 8)

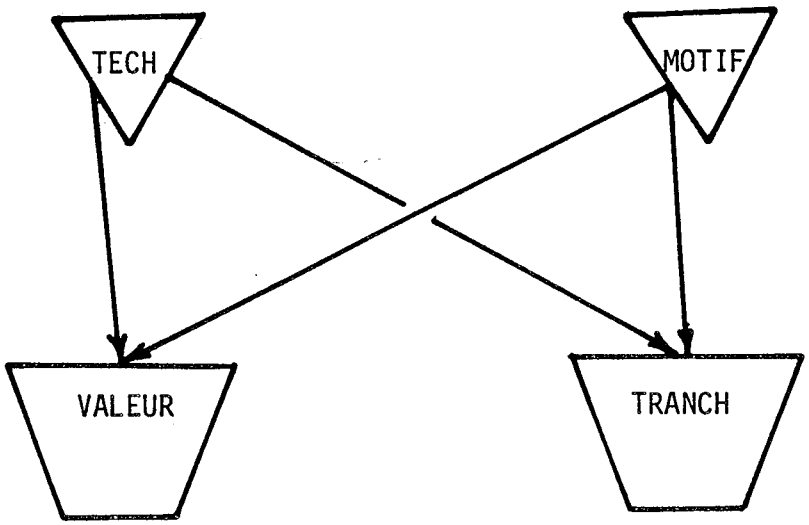


Fig. 8

1.3.2.3. Construction de la base de données

1. Définition

La longueur maximale d'un item dans le système est de 4096 octets, et il ne peut y avoir plus 127 items différents.

On peut trouver un item sur plusieurs ensembles de données, ce dernier nous permet de construire un système distribué. Cet item est l'item commun entre plusieurs ED.

2. Type de données

Le type dépend de la forme des items de données, mais en général il existe trois types de déclaration de données :

- réel R2 de deux mots
- entier (I1) d'un mot
- caractère (Xn) de n/2 mots

3. Spécification fonctionnelle

Dans le système de base de données on peut donner la spécificaion suivante:

- Dans la base de données on peut avoir jusqu'à 50 fichiers de données (ED)
- La taille d'un ED est 2^{31}-1 ou celle de son support
- Les intervalles de chacun des types d'items de données sont
  . entier (-32768, + 32767)
  . réel (∓ 147E-39, ∓ 170E38)
  . caractérisé (255 caractères maximum)
- le nombre maximum des items clés est égal à 16
- tous les noms des bases de données, des fichiers de base et les items dans la base de données sont de 1 à 6 caractères. Il existe différents niveaux de consultation dans une base de données IMAGE (1 à 8) qui permettent de faire des accès restreints aux informations.
- pour une base de données, il y a la possibilité d'avoir jusqu'à 255 noms d'items.

4. Synonym es

Il y a synonyme lorsque deux ou plusieurs valeurs de clefs donnent la même valeur de hash-code et par conséquent le même index d'accès dans le fichier maître.

Le premier synonyme occupe sa vraie place, c'est une première entrée mais tous les autres synonymes sont chainés à la première entrée. La(figure 9)montre le chemin de chainage.

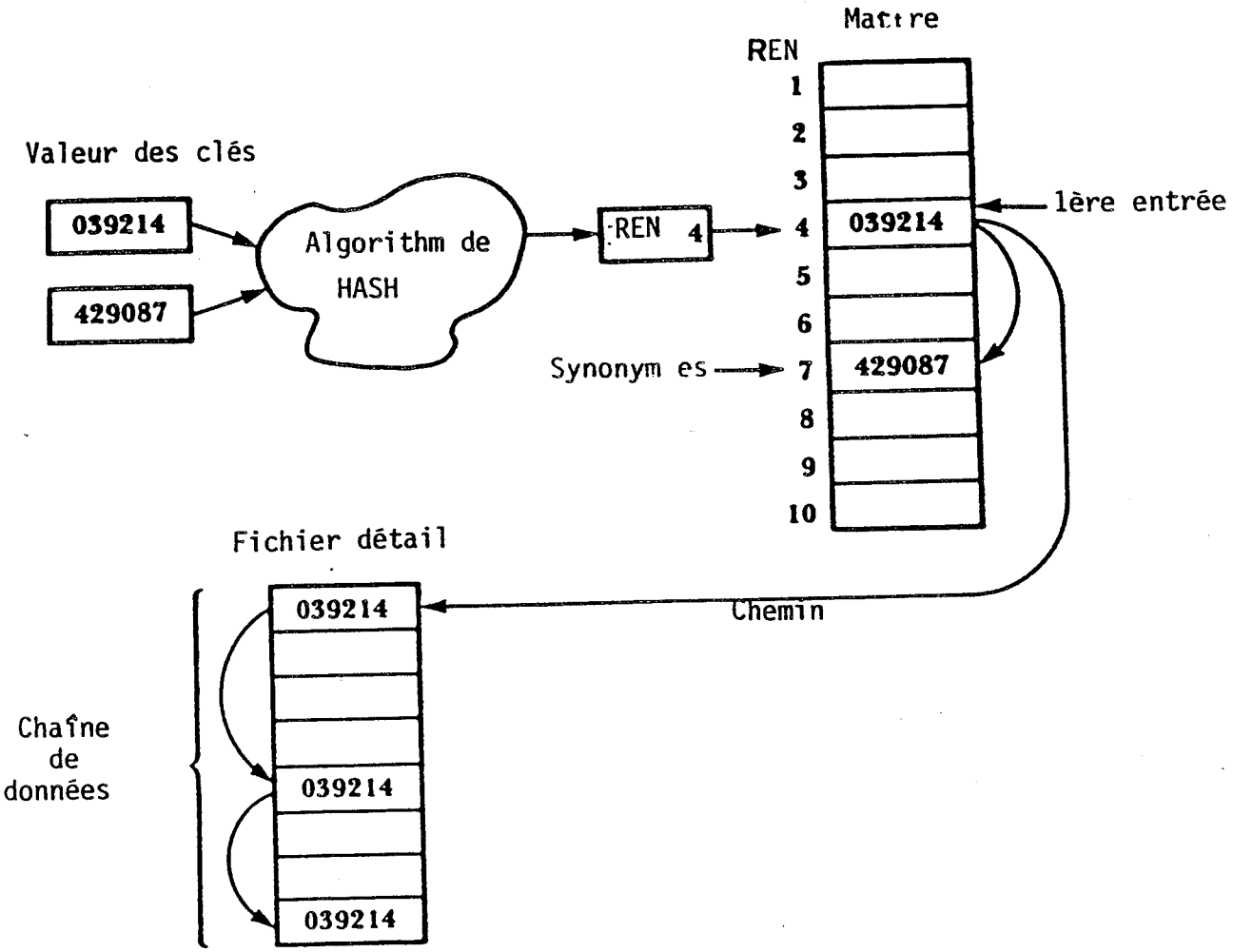


Fig. 9 Le chemin de chainage dans le système IMAGE

1.3.3. Conclusion

Nous pouvons dire qu'il n'existe pas de structure de stockage idéal, qui puisse être définie comme clé privilégiée d'entrée lors de la définition de la base.

Si les données n'ont pas justifié l'existence d'un fichier maître, on se trouve obligé d'examiner les structures séquentiellement.

I - 4. BASES DE DONNEES RELATIONNELLES

Ce modèle est fondé sur la notion mathématique de relation, où une relation est un ensemble de n-uplets avec n donné à l'avance. Pour comprendre le modèle relationnel nous ferons dans cette partie un rappel des concepts du modèle relationnel.

Concept de modèle relationnel

La principale force des modèles relationnels est de bâtir une frontière entre les aspects logiques et physiques dans la conception d'une base de données.

L'approche relationnelle est basée sur la transformation formelle des relations en utilisant les propriétés mathématiques et la normalisation des relations qui permettent d'obtenir une collection de relations dites normalisées. Les relations normalisées ne comportent aucune redondance.

Pour l'étude détaillée du modèle relationnel des données et de la dépendance fonctionnelle, on se reportera aux références (LOR 79)(COD 82) (HAL76)(BEG 80)(DEL 82).

Concepts de relation

I - 4.1. Définition

Etant donnés des ensembles (E1,E2,...,En) non nécessairement disjoints, on définit une relation R sur un ensemble comme une partie du produit cartésien de l'ensemble E1XE2X...XEn.

Si R est un ensemble ordonné de valeurs <e1,e2,...,en>, tels que e1$\in$E1,....,en$\in$En, où les ensembles E_i sont les domaines de valeurs associées à R, les différents attributs peuvent se partager un même domaine. Lorsqu'il n'y a pas de confusion possible, on ignore les attributs.

Soit un ensemble d'attributs A$\subseteq$E, une A-valeur est une application qui à chaque attribut E dans A associe une valeur dans le domaine E. Une relation R sur l'ensemble A notée (R(A)) est un ensemble de A-valeurs.

Graphiquement, une relation est un tableau à deux dimensions dans lequel les colonnes correspondant aux attributs, et les lignes aux n-uplets, la valeur "n" est appelée le degré de R.

On définit le tableau contenant les valeurs suivantes :

Relation ENTÊTE

N° ENR	TECHNO	PROVEN	LOT	N° MOTIF
II-52	HMOS II	DI	B25	CONTACT
A-30	HMOS I	DT-TI	SIM	CONTACT
I-52	HMOS II	DI	SIM	RCONTACT
A-38	HMOS I	DT-TI	C-30	CONTACT

A partir de ce tableau on construit le schéma de la relation associée au tableau précédent.

ENTETE (n° enregistrement, technologie, provenance, lot, n° motif) chaque relation peut comporter plusieurs identifiants (clé) et elle a toujours au moins un identifiant. On choisit un identifiant comme une clé principale de cette relation ; dans notre exemple c'est le n° d'enregistrement.

Relation ARCHIVE

N°BANDE	TECHNO	NOM-UTILISATEUR	DATE
BI 102	HMOS II	Martin	17 11 81
BI 107	HMOS II- LOCOS	Doupe	31 1 82
BO 13	HMOS I	Martin	07 02 82
BI 153	HMOS II	Martin	15 03 82
BD 212	HMOS	Doupe	15 03 82
BA 705	HMOS I	Feron	21 08 82
BF 73	HMOS II	Doupe	15 11 82

La clé primaire pour la relation ARCHIVE est le n° de Bande.

Entre les deux relations il y a l'association par les attributs TECH.

I - 4.2. Les opérateurs relationnels

- θ-sélection

C'est une opération qui sélectionne des tuples d'une relation R(A,B1,B2,C) tel que les valeurs de B1,B2 appartiennent au même domaine.

$\theta = \{ >, <, \geqslant, \leqslant, =, \neq \}$ si θ est l'égalité (ce qui est le plus courant), le θ-sélection est simplement appelé SELECTION.

$R[B1\ \theta\ B2]$ sous ensemble de n-uplets de R tel que chaque valeur de B1,B2 vérifie la relation $B1\ \theta\ B2$.

- Projection

C'est un item qui restreint une relation sur certains attributs soit R(X) un n-uplet sur X, si T$\subset$X, T sous-ensemble, si on appelle

U = X - T (complément d"T" dans U)

La projection de R sur T notée R(T) est l'ensemble

$R(T) = \{<t> | \exists\ u \in <t,u>\ R\}$

- θ-Join

C'est un item qui relie entre elles deux relations sur un attribut commun.

Soit $\theta = \{<, \leqslant, =, >, \geqslant, \neq\}$,{A,B1,...} = X, Y = {B2,C,....} deux sous ensembles d'attributs de R1,R2 tel que B1,B2 prennent leur valeur dans le même domaine.

R1[B1 θ B2]R2 relation obtenue en associant les n-uples de R1 et les n-uples de R2 dans les valeurs de B1,B2. Si θ est l'égalité, la θ-jointure est appelée Jointure-naturelle ou équijointure.

- Jointure-naturelle

La jointure-naturelle est un θ-join mais θ de valeur "=" dans lequel on supprime une des deux colonnes communes, il peut être étendu à plusieurs relations R1*R2*...*Rn lorsque les ensembles de définition des Ri sont disjoints, la jointure-naturelle se ramène à un produit cartésien.

- Produit cartésien

Soit R1(X) et R2(X) deux relations de degré (n1,n2) le produit cartésien R1xR2 est l'ensemble des (n1+n2)-uplets tel que X,Y de R1, R2 sont disjoints.

- Opérations ensemblistes

Ces opérations peuvent définir les opérations d'union, d'intersection, de différence (sur deux relations de même degré dont les domaines sont identiques) et de division (sur deux relations de degrés différents).

I - 4.3. Processus de normalisation d'une relation

Il permet une collection de relations dites normalisées. Les relations normalisées ne comportent aucune redondance dans la représentation du problème et permettent de vérifier que la représentation est cohérente.

On se limite dans les notions de processus de normalisation à la dépendance fonctionnelle et aux formes de normalisation (expliqués dans l'annexe II).

I - 4.4. Conclusion

Les objectifs du modèle relationnel consistent essentiellement à proposer des structures de représentation des faits élémentaires qui s'intéressent à la logique et à la formulation du problème sans se préoccuper de leur représentation physique et à fournir des langages de manipulation de haut niveau comme nous le verrons au chapitre II.

L'efficacité d'un SGBD relationnel "dépend principalement de deux facteurs: la représentation des relations à l'aide de structure de données permettant des accès variés et rapides aux éléments d'une relation par un coût comparable à ceux des SGBD réseau ou hiérarchique, et la mise en oeuvre de techniques d'optimisation dans l'interprétation du langage de manipulation" (DEL 82).

I - 5. PRESENTATION DU MODELE IMREL

Objectif

Les 2 approches que nous venons de voir : types d'enregistrement (hiérarchique et réseau) et modèles conceptuels (relationnels) différent dans la façon dont l'utilisateur voit et manipule les relations entre les données.

Dans le type d'enregistrement :

- hiérarchique : ces relations sont entièrement implicites
- réseau : ces relations sont représentées explicitement grâce à des pointeurs

Dans le modèle conceptuel :

- relationnel : les relations sont représentées explicitement, exactement de la même façon que les entités, c'est à dire grâce à des couples.

En conclusion de la même façon qu'il n'existe pas de structure de stockage idéale, il n'existe pas de modèle de données idéal. On s'oriente donc vers un modèle de données adapté, avec les avantages de chacune des classes de type IMREL (IMAGE _ RELATIONNEL).

Nous choisirons le modèle relationnel comme interface entre toutes les bases pour les raisons suivantes :

- Le modèle relationnel est capable de protéger les informations ayant une structure de stockage complexe.
- Il est soutenu par un langage de requête de haut niveau et non procédural.
- Les stockages et les structurations sont très simples toutes les informations sont représentées sous forme de registres.
- L'accès ne doit pas être nécessairement prédéfini. L'implantation des opérateurs de l'algèbre relationnelle font partie de primitives de plus haut niveau offerts par les SGBD relationnels.

- L'existence d'un langage de manipulation de haut niveau et non-procédural est nécessaire pour minimiser le coût de la communication entre les bases.
- Le modèle relationnel prévoit une interface simple avec les autres modèles.
- Le progrès dans les procédures de stockage offert par le modèle relationnel, qui est arrivé à un haut degré d'efficacité.
- le modèle relationnel a une réponse rapide en utilisant le chemin d'accès "ad-hoc" (en tenant compte des taux d'utilisation de requêtes).

Le principe fondamental dans IMREL consiste à définir l'information pouvant s'exprimer sur l'espace relationnel pour pouvoir appliquer des opérateurs d'algèbre relationnelle, la structure efficace du système IMAGE, va nous conduire aux propositions élémentaires suivantes :

- La possibilité d'échanger les informations entre diverses bases interrogées ;
- Chaque base est considérée comme étant capable de manipuler les données transférées dans d'autres bases ;
- Il existe un réseau de télécommunication qui lie les divers SGBD ;
- L'accès à un SGBD local n'est pas affecté par les informations qui doivent leur être transmises ;
- La communication entre les bases est le principal facteur du coût d'exécution local pratiquement nul.

Il est possible en utilisant l'approche d'IMAGE (hiérarchique) de générer les structures de données relationnelles.

Le schéma illustre le modèle de données. Les structures d'IMAGE n'ont qu'un seul niveau de dépendannce au niveau le plus bas. Les tableaux intermédiaires qui partent de l'occurence du supérieur, traversent toutes les occurrences des subordonnés, retournent finalement à l'occurrence du supérieur d'une façon directe.

La présentation d'IMREL va suivre le schéma suivant :

1. Introduction
2. Schéma de transformation
3. Définition
4. Schéma des relations
   - niveau interne (clé + index)
   - niveau externe (distance entre les niveaux)
5. Exemple

1.5.1.1. Introduction

Pour réaliser l'implantation d'une interface permettant la manipulation en modèle "relationnel" d'une base de données gérée par le SGBD IMAGE, nous définirons un module qui permet la transformation de la structure d'une base IMAGE afin que l'utilisateur puisse avoir une vue relationnelle de la base de données. Pour réaliser cette transformation, il a été nécessaire de définir un modèle de données rassemblant la plupart des concepts que l'on retrouve dans les deux systèmes, ce modèle permet d'établir les règles de correspondances qui sont à la base de la transformation (SEO 81)(DAT 77)(SIR 81)(GAR 81)(BRI 81) (SHO 81).

On appelle ce modèle par "IMREL".

1.5.1.2. Le schéma de transformation

le schéma est la description logique de la base de données.Un schéma de description dans IMREL contient quatre types de déclarations :

* la description du nom du schéma
* la déclaration de type d'enregistrement
* la déclaration de groupes (set)
* la déclaration de mémoire (capacité)

Nous avons deux formes d'enregistrement dans le modèle IMREL : un type d'enregistrement de description et un type d'enregistrement de connexion.

Le type d'enregistrement de description "ED" a un ou plusieurs attributs qui décrivent les propriétés de ces enregistrements. Il est tout à fait semblabe à une relation dans un modèle de données relationnel, et la clé d'enregistrement ED est la clé de la relation.

Le type d'enregistrement de connexion est utilisé quand nous voulons connecter n entités élémentaires (type d'enregistrement de description). Chaque entité est constituée d'un ensemble variable de types élémentaires (réel,entier, ...) mais l'enregistrement de connexion a un type unique pour la n-entité, et par suite pour chacune des entités qui les composent.

L'enregistrement de connexion de chaque entité est affecté de la même valeur caractéristique de la n-entité, et représente la connexion entre les "n" entités correspondantes. La (figure 10) montre les deux types d'enregistrements.

Pour ce nouveau type d'enregistrement (n-entité) nous pouvons définir une relation dont la clé est l'enregistrement de connexion, caratéristique de la n-entité .

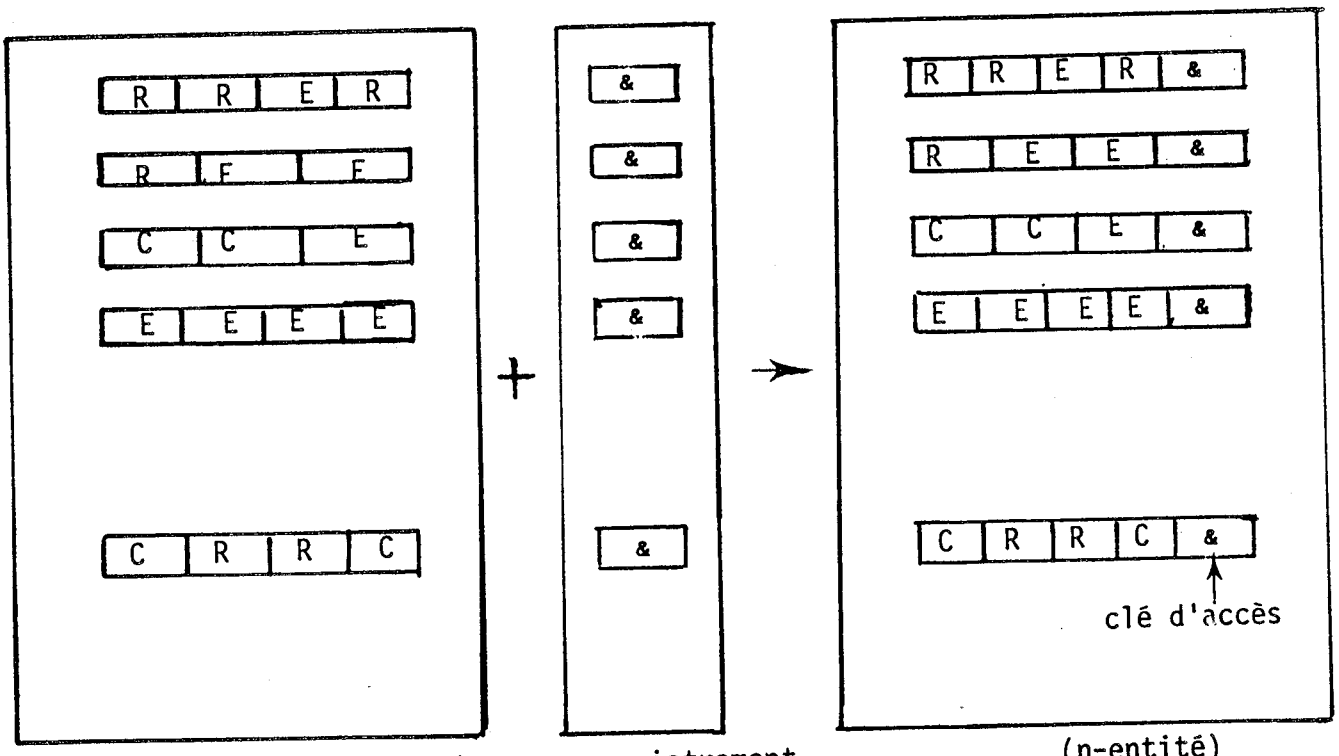


Fig.10 Le schéma de transformation

Un fichier est défini dans le schéma de la base afin d'avoir un certain type d'enregistrement comme clé et certain type d'enregistrement non-clé.

Pour chaque type de fichier ne contenant pas de clé, nous n'aurons pas de correspondance avec un schéma relationnel, mais on créera un tableau dans chaque relation pour définir tous les chemins d'accès et la hiérarchie d'une base.

Ce tableau contient trois noms d'attributs (clé, fichier père, fichier fils). On utilise dans le processus de requête multibase un opérateur relationnel avec les instructions du modèle IMREL pour identifier le chemin d'accès.

Le tableau peut être vu comme une nouvelle relation qui est stockée dans la base de données locale et qui est utilisée seulement pour la traduction de la requête.

Elle n'est pas reliée au schéma global qui va être présenté à l'utilisateur. Une zone est définie pour la sauvegarde du résultat d'un traitement de requête sur une base de données. Pour chaque type d'enregistrement le schéma de base spécifie la place dans laquelle les enregistrements doivent être placés. Ce type de place est organisé dans la répartition horizontale ou verticale d'une base de données relationnelle, on peut alors avoir besoin de créer un nouveau schéma relationnel.

Soient techno-motif-IDRES les fichiers d'une base qui utilisent le schéma IMAGE, on définit ce schéma de la manière suivante (figure 11) :

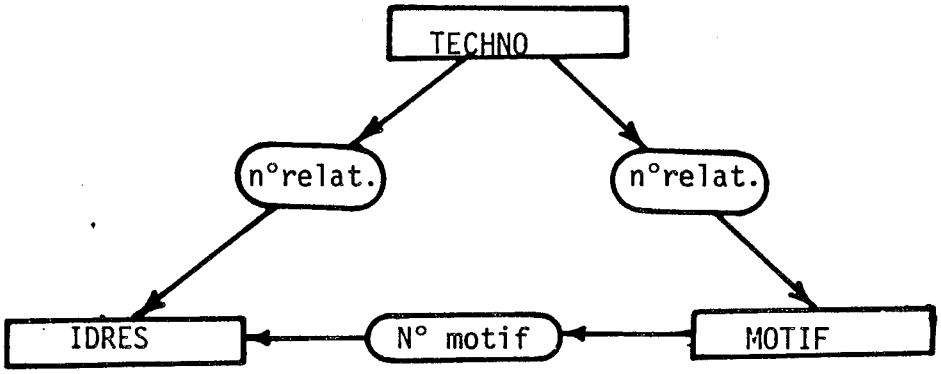


Fig.11 Schéma relationnel dans IMREL

On applique le schéma de transformation, on définit les relations suivantes:

Techno = (Technologie, Lot, Provenance, Tranche, n° relation)
MOTIF = (n° relation, n° motif, Nom motif, Méthode, Sous type)
IDRES = (n° relation, n° motif, Nom IDRES, MINI, MAXI, DATE)

Nous définissons l'accès ou relation dans le tableau :

clé	Fichier père	Fichier fils
N° relation	Techno	Motif
N° motif	Motif	IDRES
N° relation	Techno	IDRES
Système	Techno	Techno
Système	Motif	Motif
Système	IDRES	IDRES

1.5.1.3. Définition du modèle

Une relation n-aire est un ensemble de n-uplets.

Supposons que les éléments du fichier maître ne soient plus des indices mais des n-uplets. Lorsque l'on donne le schéma d'accès par un tableau, et qu'on lui demande de retourner des résultats, ce tableau est appele fichier maître.

Une base de données IMREL est un ensemble de relations; Chaque relation est composée d'un certain nombre d'attributs. Un attribut prend ses valeurs dans un

ensemble appelé item, et peut être déclaré comme clé primaire base sur la notion de fichier maître.

Un item peut être un entier, un réel, une chaine de caractères de longueur variable ou fixe.

On voit qu'à un index de fichier maître peuvent correspondre plusieurs résultats retournés, ceci est une différence entre le modèle relationnel et le système IMAGE, dans ce cas on définit la clé composée.

Les opérateurs relations SELECT et PROJECT,peuvent se mettre sous forme d'interrogations en utilisant le fichier maître, on définira les sous-programmes qui réaliseront ces deux opérations (voir II - 3.).

L'opérateur JOIN, est le plus coûteux dans un système distribué on va définir le sous programme exécuté, et la possibilité d'optimiser cet opérateur.

Au delà de ces définitions il faut remarquer que l'algorithme d'interprétation du système IMREL, est toujours valable dans les deux systèmes "Relationnel et hiérarchique".

On doit noter toutefois que si le nombre de fichiers maîtres dans une base de données IMAGE augmentent alors le nombre de schéma d'accès va augmenter aussi dans IMREL.

1.5.1.4. Modèle d'analyse de ces bases

Nous faisons l'hypothèse que l'on peut décrire une base par des classes de clés, des classes de sous-clé, et des classes d'attibuts.

Toutes ces classes montrent l'association entre les clés, les sous-clés, les attributs. Elles sont représentées par des types que nous appelons :

- type clé
- type sous-clé
- type association

noté en abrégé T-clé, T-s-clé, T-ass.

Le type est une abstraction qui permet de définir en compréhension l'ensemble des occurences d'une base. La description du type constitue le mode de description de chaque occurence de la classe.

La T-clé représente l'ensemble des attributs communs entre toutes les bases.

Exemple

La T-clé Technologie représente l'ensemble des technologies qui se trouvent dans toutes les bases (c'est un maître automatique).

La définition de la clé doit être faite au moment de la définition du schéma de base.

Une T-s-clé est un attribut attaché à une base comme clé d'entrée à la base.

Exemple

La T-s-clé Masque représente l'ensemble des types Masque qui permet de faire un accès à l'information stockée dans une base (BASE 1).

Enfin, puisque dans la réalité on peut donner un attribut comme point d'entrée, on va traduire le lien entre deux attributs pour constituer un point d'entrée direct. Ce type est une T-association.

Une association est une combinaison de deux attributs.

Exemple

A Val-Max on peut associer la sous-clé Résultats, on génére l'accès à val-max par l'accès au résultat si Val-Max est un élément de la liste d'objet, et on définit l'accès séquentiel si Val-max est une condition.

Une T-association définit une association entre les occurrences dans un même ED. Si dans une sous-requête on a une clé (ou sous-clé) avec un attribut, on accède à la clé (sous-clé), mais si on a un attribut avec une T-association (ex. val-max) l'accès est séquentiel. A l'aide de ces classes on définit le lien entre et dans les classes. Il y a cinq groupes de liens hiérarchiques.

Gi1 = (T-association (attribut⟶attribut))

Gi2 = (Gi1 $\cup$ (T-s-clé ⟶ T association))

Gi3 = (Gi2 $\cup$ (T-s-clé ⟶ T-s-clé))

Gi4 = (Gi3 $\cup$ (T-clé ⟶ T-s-clé))

Gi5 = (Gi4 $\cup$ (T-clé ⟶ T-clé))

Nous cherchons à modéliser une approche mathématique, qui nous permette l'utilisation de deux modèles sous la forme IMREL répartie. La (figure 12) représente graphiquement les hiérarchies maximales pour un modèle de base de données répartie à chaque niveau on traite une classe de groupes ce qui permet un traitement descendant, ascendant ou mixte. On définit le type de traitement après le calcul de la distance entre deux ED.

1.5.1.5. Approche mathématique

Soient les ensembles (B1,B2,...,Bn) $\in$ B, B bases de données répaties, on définit pour chaque base de données, les ensembles de relations R= (R1,R2,...,Rm) et ki la clé dans Ri $\forall i \in [1...m]$.

On dit que Ri et Rj sont hiérarchiquement conjointes s'il existe une dépendance fonctionnelle entre Ki, kj de ki⟶kj, le"⟶" signifie il existe une dépendance fonctionnelle avec la possibilité d'être triviale.

On considère une sous-base de Bi contenant les relations (Ri1,Ri2,...,Rij) et si on a ki1⟶ki2⟶....⟶kij on appelle cette sous-base une hiérarchie.

H = (Ri1,Ri2,....Rij)

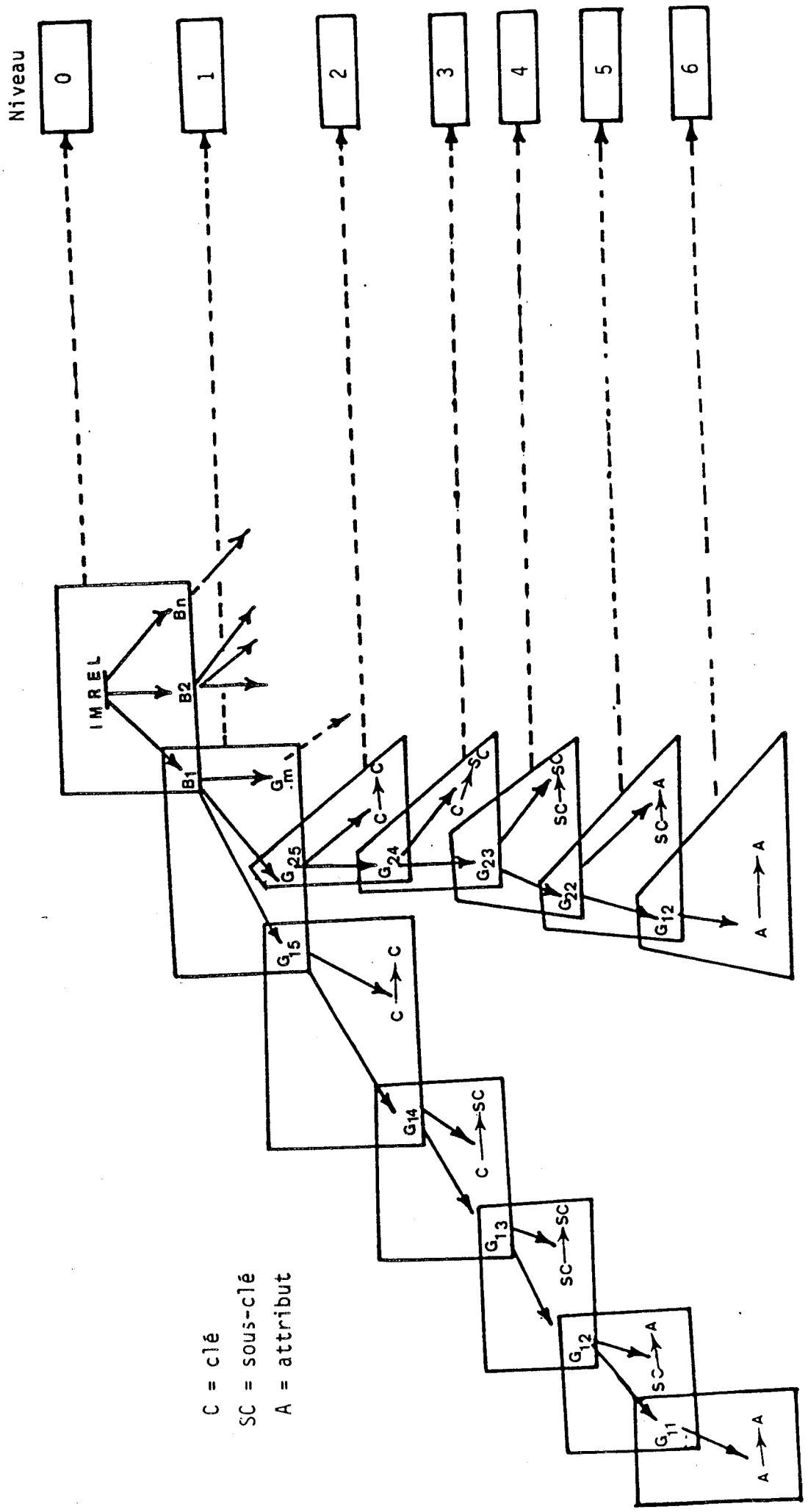


Fig.12 Représentation graphique IMREL

Exemple

Soient les trois relations :

- ENTETE (n° enrg,Tech, Lot, n° motif)
- Motif (n° motif, type motif, Sy1, Sy2, n° résultat)
- Résultat (n° résultat, valeur, CO.MES1,...)

On considèrera le schéma suivant (figure 13)

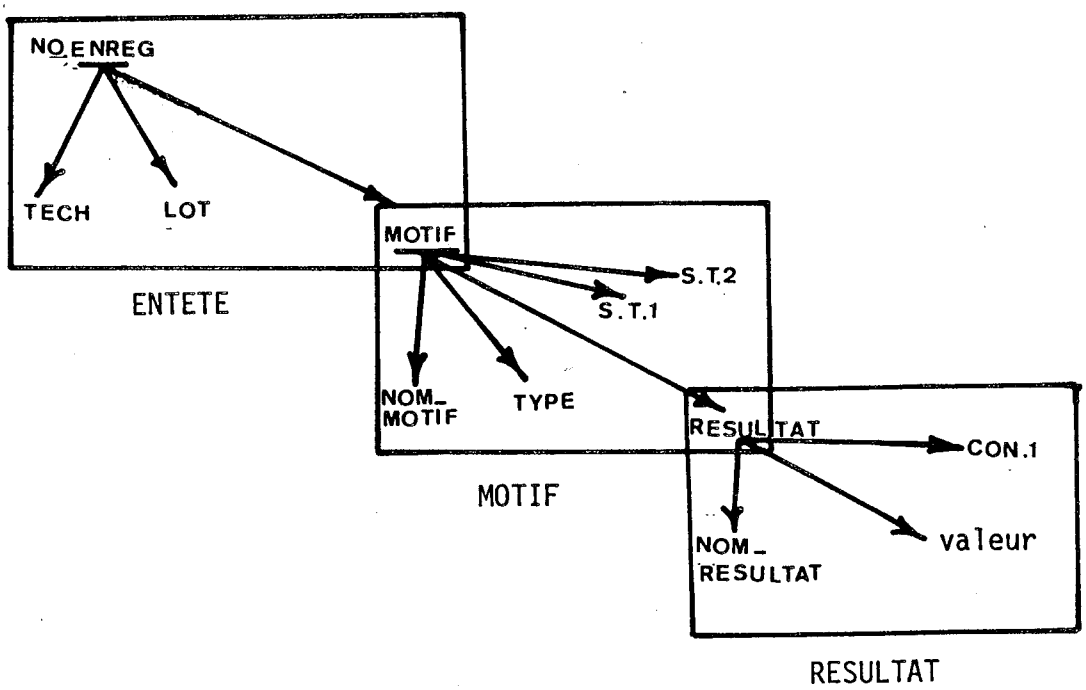


Fig.13 Schéma de base "exemple"

On a :

H = (ENTETE, MOTIF, RESULTAT)

Suivant les définitions nous savons que :

* chaque fichier dans IMAGE est une relation dans le modèle relationnel
* les dépendances fonctionnelles entre les fichiers ne sont appliquées que dans le cas où il n'existe pas de dépendance entre les attributs dans le fichier.
* pour déterminer les distances entre deux attributs, on calcule le schéma minimal de la façon suivante :

$d(A \rightarrow B) = d|(GA)| + d\,H|(RA \rightarrow RB)| + d|(GB)|$

$d|GA| = A \rightarrow ki$ si $A = ki = 0$

A = clé $\longrightarrow kA = 1$

A = s-clé $\longrightarrow kA = 2$

$A = A \longrightarrow kA = 3$

$d\,h|(RA \rightarrow RB)| = J - i$

ou H = (R1,R2.....RA.....RB.....Rn)

i

J

cordinalité = N Alors $d(H) \longrightarrow [1,N]$

$Ri \longrightarrow h(Ri)$

$h(R1) = N$

$h(R2) = N - 1$

$h(Ri) = N + 1 - i$

$h(Rj) = N + 1 - j$

$h(RN) = 1$

$d(RA \longrightarrow RB) = N + 1 - i - (N+1-j) = j - i$

Exemple

On considère le diagramme suivant (figure 14)

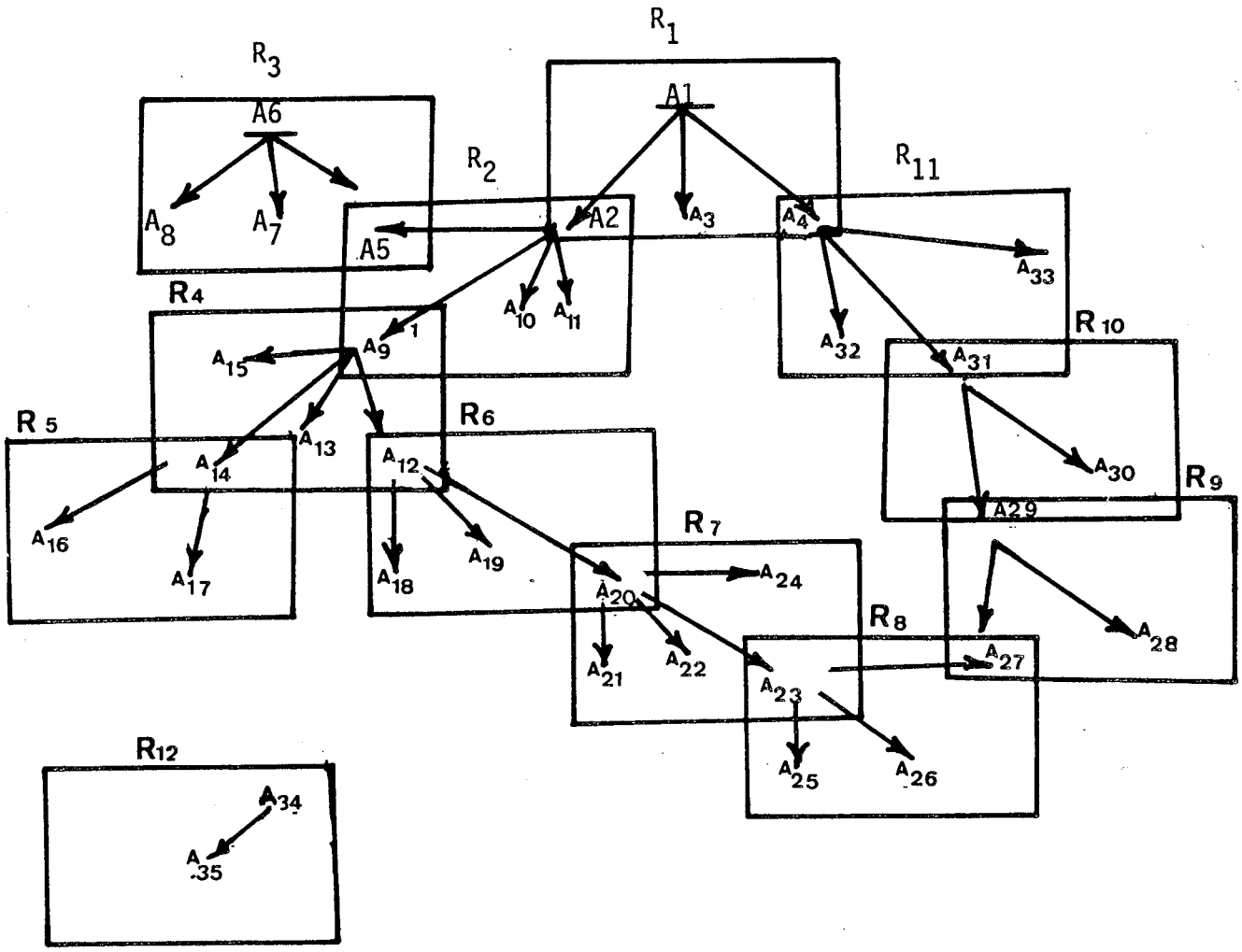


Fig.14 Diagramme d'une base

$d(A19 \longrightarrow A31) =$

$d(A19 \rightarrow A31) = R6 \rightarrow R11 = Min\left[(R6 \rightarrow R4 \rightarrow R1 \rightarrow R11), (R6 \rightarrow R8 \rightarrow R11)\right]$

$d(A19 \rightarrow A31) = Min\left[(d(GR6) + d(R6 \rightarrow R4) + d(R4 \rightarrow R1) + d(R1 \rightarrow R11) + d(GR11)),\right.$
$\left.(d(GR6) + d(R6 \rightarrow R8) + d(R8 \rightarrow R11) + d(GR11)\right]$

$= Min\left[2 + 1 + 2 + 1 + 1,\ 0 + 1 + 3 + 3\right]$

$= Min\left[7,\ 7\right] \rightarrow 7$

$d(A34 \longrightarrow A15) = \infty$

Exemple

Le concepteur dispose d'un ensemble d'énoncés sur la réalité organisationnelle que l'on veut traduire informationnellement on notera que ces énoncés, exprimés déjà d'une manière concise ont fait l'objet d'un premier effort de clarification. On sait par expérience que ce travail de clarification est souvent délicat à mener à bien (MOR 82)(G2P 82). Voir le tableau qui représente les syntaxes des bases.

La structure de la base de données "Technologie"

Les informations fournies par les résultats des tests, vont être stockées d'une façon simple, pour avoir le temps d'accès le plus faible possible et aussi parce qu'il est nécessaire de connaître la taille de chaque variable, pour limiter la place utilisée.

On adopte la méthode de stockage au mode de traitement en tenant compte de la longueur optimale des enregistrements.

Cette étude a permis de répartir l'information sur trois relations (fichier).

Chaque relation se caratérise par son propre domaine et sa composition.

Tableau Representant les Syntaxes des BASES

Inventaire des CLES << champ commun >>

TECHNOLOGIE

PROVENANCE

LOT

Inventaire des SOUS-CLES

Base TECHNOLOGIE	Base ARCHIVAGE	Base CIRCUITS
MASQUE	NO.BANDE MAGNETIQUE	MASQUE
TRANCHE	NOM FICHIER INPUT	TRANCHE
NOM DU FICHIER	NOM FICHIER TESTE	NOM DU FICHIER
MOTIF	INDEX ARCHIVAGE	MOTIF
IDRES	INDEX TYPE BANDE	IDRES
DATE	NOM PROGRAMME	DATE
VALEUR MOYENNE	NOM UTILISATEUR	VALEUR MOYENNE
	ANNEE UTILISATION	

Inventaire pour chaque FICHIER : cle,sous_cle et attributs .

*** BASE TECHNOLOGIE ***		
DB1031	DB1032	DB1033
TECHNOLOGIE	RELATION ENTETE	RELATION ENTETE
PROVENANCE	NOM-MOTIF	RELATION MOTIF
MASQUE	SOUS-TYPE 1	NOM-RESULTA
LOT	SOUS-TYPE 2	METHODE
PLAQUE/TRANCHE	SOUS-TYPE 3	CON.DE MESEUR 1 -NOM-
NOM DU FICHIER INPUT	DIM.GEMETRIQUE 1 -NOM-	CON.DE MESEUR 1 -VALEUR-
NOM DU FICHIER SIAM	DIM.GEMETRIQUE 1 -VALEUR-	CON.DE MESEUR 2 -NOM-
DATE DE TRANSFERT (AAMMJJ)	DIM.GEMETRIQUE 2 -NOM-	CON.DE MESEUR 2 -VALEUR-
RELATION ENTETE	DIM.GEMETRIQUE 2 -VALEUR-	CON.DE MESEUR 3 -NOM-
	DIM.GEMETRIQUE 3 -NOM-	CON.DE MESEUR 3 -VALEUR-
	DIM.GEMETRIQUE 3 -VALEUR-	CON.DE MESEUR 4 -NOM-
	RELATION MOTIF	CON.DE MESEUR 4 -VALEUR-
		CARTO OU AQF TYPE
		NBRE DE PUCES TESTEES
		NBRE DE TESTS BON
		NBRE DE TESTS RETENUS
		DATE DE TEST
		MINI
		MAXI
		MOYENNE
		SIGMA
		MEDIA
		MODE
		COMMENTAIRES

*** BASE ARCHIVAGE ***			
DF2ABM	DD2AFS	DD2ABC	DD2UPS
NO. DE B.M.	NO. DE B.M.	NO.DE B.M.	ITEM CLE NOM UTILISATION
TYPE D' UTILISATION.	NOM FICHIER -INPUT-	NOM FICHIER -INPUT-	ITEM CLE NOM PROGRAMME
TYPE D'ARCHIVAGE.	NOM FICHIER TEST -SIAM-	TECHNOLOGIE	ITEM CLE ANNEE UTILISATION
DERNIER DATE ARCHIVAGE	TECHNOLOGIE	PROVENANCE	DERNIERE DATE D'UTILISATION
DERNIER HEURE ARCHIVAGE	PROVENANCE	LOT	DERNIERE HUERE D'UTILISATION
DERNIERE PERSONNE.	LOT	NB ENREG. TOTAL	COMPTEUR ANNEE D'UTILISATIO
COMMENTAIRE	INDICATEUR D'ARCHIVAGE	NB ENREG. ENTETE	COMPTEUR MENSUEL
NB ENREG. DE B.M.	NB. ENREG. DU FICHIER	NB ENREG. MOTIF	
DATE DE LA RESTITUTION		NB ENREG. IDRES	
PERSONNE RESTITUTION			
INDICATEUR DE B.M.SUITE			
NOM B.M.SUITE			
NO.B.M.SUITE			
NOMBRE DE LAI B.M. SUITE			

*** BASE CIRCUITS ***		
DB3031	DB3032	DB3033
TECHNOLOGIE	RELATION ENTETE	RELATION ENTETE
PROVENANCE	NOM-MOTIF	RELATION MOTIF
MASQUE	SOUS-TYPE 1	NOM-RESULTA
LOT	SOUS-TYPE 2	METHODE
PLAQUE/TRANCHE	SOUS-TYPE 3	CON.DE MESEUR 1 -NOM-
NOM DU FICHIER INPUT	DIM.GEMETRIQUE 1 -NOM-	CON.DE MESEUR 1 -VALEUR-
NOM DU FICHIER SIAM	DIM.GEMETRIQUE 1 -VALEUR-	CON.DE MESEUR 2 -NOM-
DATE DE TRANSFERT (AAMMJJ)	DIM.GEMETRIQUE 2 -NOM-	CON.DE MESEUR 2 -VALEUR-
RELATION ENTETE	DIM.GEMETRIQUE 2 -VALEUR-	CON.DE MESEUR 3 -NOM-
	DIM.GEMETRIQUE 3 -NOM-	CON.DE MESEUR 3 -VALEUR-
	DIM.GEMETRIQUE 3 -VALEUR-	CON.DE MESEUR 4 -NOM-
	RELATION MOTIF	CON.DE MESEUR 4 -VALEUR-
		CARTO OU AQF T
		NBRE DE PUCES TESTEES
		NBRE DE TESTS
		NBRE DE TESTS RETENUS
		DATE DE TEST
		MINI
		MAXI
		MOYENNE
		SIGMA
		MEDIA
		MODE
		COMMENTAIRES

Cette composition doit s'adapter au mode d'utilisation le plus courant de la base de données, et favoriser l'opération la plus fréquente, et la condition la plus appliquée.

Pour que notre modèle de base ait la possibilité de travailler en mode relationnel, on va définir le descriptif de la base de la manière suivante : cette base de données se compose de trois relations pointées par plusieurs clés (fichier maître).

1. Relation ENTETE

Cette relation est composée par les items suivants :

* TECHNOLOGIE : on peut avoir 15 types différents par année (HMOS I ....), taille d'une technologie (20 car)
* PROVENANCE : 5 au total, taille d'une provenance (10 car)
* MASQUE : 10 au total, taille d'un masque (20 car)
* LOT : on peut avoir 30 lots différents pour une technologie, taille du type lot (16 car)
* TRANCHE : on peut avoir 25 tranches différentes pour un lot donné, taille du type tranche (6 car)
* FICHIER INPUT : clé spécifiant le nom du fichier de chargement permettant d'une part de ne pas charger plus d'une fois le même fichier dans la base (6 car)
** FICHIER TEST COMPLET : spécifie le nom de fichier SIAM à partir duquel les résultats ont été extraits (6 car)
* DATE DU CHARGEMENT : est la date à laquelle le chargement s'est effectué (6 car)
** RELATION ENTETE : clé permettant la liaison d'une Entete avec l'autre relation (2 réels)

Les clés qui ont été définies sur cette relation sont :

* TECHNO/PROVENANCE : clé groupée permettant de recherche tous les lots pour une technologie et une provenance spéficiques

* TECHNO : clé permettant de rechercher tous les lots (ainsi que leurs tranches) d'une technologie
* LOT-TRANCHE : clé groupée permettant de rechercher tous les Motifs et les IDRES s'y rapportant pour un lot et une Tranche spécifiques
* LOT : clé permettant de rechercher toutes les tranches pour un lot spécifique
* TECHNO-PROVEN-LOT : clé groupée permettant une recherche identique à la clé LOT
* TECHNO-PROVEN-LOT-TRANCHE : clé groupée permettant une recherche identique à la clé Lot-Tranche
* FICHIER ENTREE
* RELATION ENTETE

2. Relation MOTIF

Cette relation est composée par les items suivants :

* RELATION ENTETE : clé déjà décrite permettant la liaison d'une ENTETE à plusieurs motifs
* MOTIF COMPLET : clé groupée permettant de contrôler si un motif n'existe pas pour une ENTETE et un IDRES identiques. Il se décompose de la façon suivante :

Nom	(12 car)
Sous-Type-1	(12 car)
Sous-Type-2	(12 car)
Sous-Type-3	(12 car)
Nom-Dimension-Gémétrique-1	(12 car)
Valeur-Dimension-Gémétrique-1	(12 car)
Nom-Dimension-Gémétrique-2	(12 car)
Valeur-Dimension-Gémétrique-2	(12 car)
Relation Entete	(2 réel)

* RELATION MOTIF : clé groupée permettant de retrouver tous les IDRES d'un motif. Elle se compose de la relation Entete et d'un motif (ce dernier étant remis à zéro à chaque nouvelle relaiton) et permet de ne pas répéter le MOTIF avec chaque IDRES s'y rapportant (6 car)

On peut avoir 50 types différents pour une tranche.

3. Relation IDRES

Cette relation est composée par les items suivants :

* RELATION ENTETE : clé déjà décrite permettant la liaison d'une Entete à plusieurs identificateurs de résultat.
* IDRES COMPLET : clé groupée permettant de contrôler si un IDRES n'existe pas pour une ENTETE et un motif identiques. Il se décompose de la façon suivante :

NOM	(10 car)
Méthode	(10 car)
Nom Condition de Mesure 1	(10 car)
Valeur Condition de Mesure 1	(10 car)
Nom Condition de Mesure 2	(10 car)
Valeur Condition de Mesure 2	(10 car)
Nom Condition de Mesure 3	(10 car)
Valeur Condition de Mesure 3	(10 car)
Nom Condition de Mesure 4	(10 car)
Valeur Condition de Mesure 4	(10 car)
Relation Entete	(2 réel)

* RELATION MOTIF : clé déjà décrite permettant la liaison d'un MOTIF à plusieurs identificateurs de résultat
* VALEURS : groupe des items représentant certaines valeurs de tests d'un identificateur de résultat :

TYPE (CARTO ou AQF)	(2 car)
Nombre de puces testées	(1 I)
Nombre de tests bons	(1 I)
Nombre de tests retenus	(1 I)
Date de test	(6 car)
Minimum	(2 R)
Maximum	(2 R)
Moyenne	(2 R)

Sigma (Ecart type)	(2 R)
Mediane	(2 R)
Mode	(2 R)

* COMMENTAIRES : annotations spécifiques à un Identificateur de résultat (80 car)

On peut avoir 200 types différents pour une tranche.

On doit tenir compte des particularités suivantes :

* Une technologie est composée de 2 champs, le 2ème étant une variante du 1er

* Une technologie peut être de différentes provenances, mais un lot ne devrait être que d'une provenance

* 1 lot peut être examiné en employant plusieurs masques.

L'examen sur une année sera nécessaire afin d'obtenir une moyenne de ces informations, sachant qu'elles représentent les cas maximum et qu'elles sont extrêmement variables.

LEXIQUE

* TECHNOLOGIE : Technologie de fabrication, correspondant à une série précise d'opérations.

Exemple (HMOS II CMOS SOS GSCLA)

Une technologie peut avoir des sous-définitions telles que :

HMOS II LOGIQUE
HMOS II ANALOGIQUE
HMOS II NMOS

et également par exemple pour HMOS II : CAISSON N
CAISSON P
EPITAXIE

* LOT : Ensemble de plaques d'une même technologie ayant un grand nombre d'opérations identiques, réalisées dans un but particulier et qui ont été fabriquées ensembles.

* MOTIF : Le motif est l'élément physique sur lequel sont effectuées des mesures. Il peut exister plusieurs possibilités physiques pour un même motif d'où la nécessité d'avoir plusieurs sous-types.

Les dimensions gémétriques peuvent constituer des sous-types. Exemple des noms des motifs :

TMOS
RESISTANCE
CAPACITE
DIODE ....

Exemple de sous-types :

motif : TMOS
sous-type-1 : Naturel/Enrichi/Locos/...
sous-type-2 : P /N

* Identificateur de Résultat (IDRES). L'identificateur de résultat est l'élément électrique pour lequel on effectue des mesures sur un motif par exemple :

Nom Méthode.....etc
Dopage Habituelle

* VALEUR : La valeur est le résultat des tests effectués sur les testeurs après un premier traitement permettant d'obtenir des statistiques élémentaires.

La figure 15 montre le schéma global de la base.

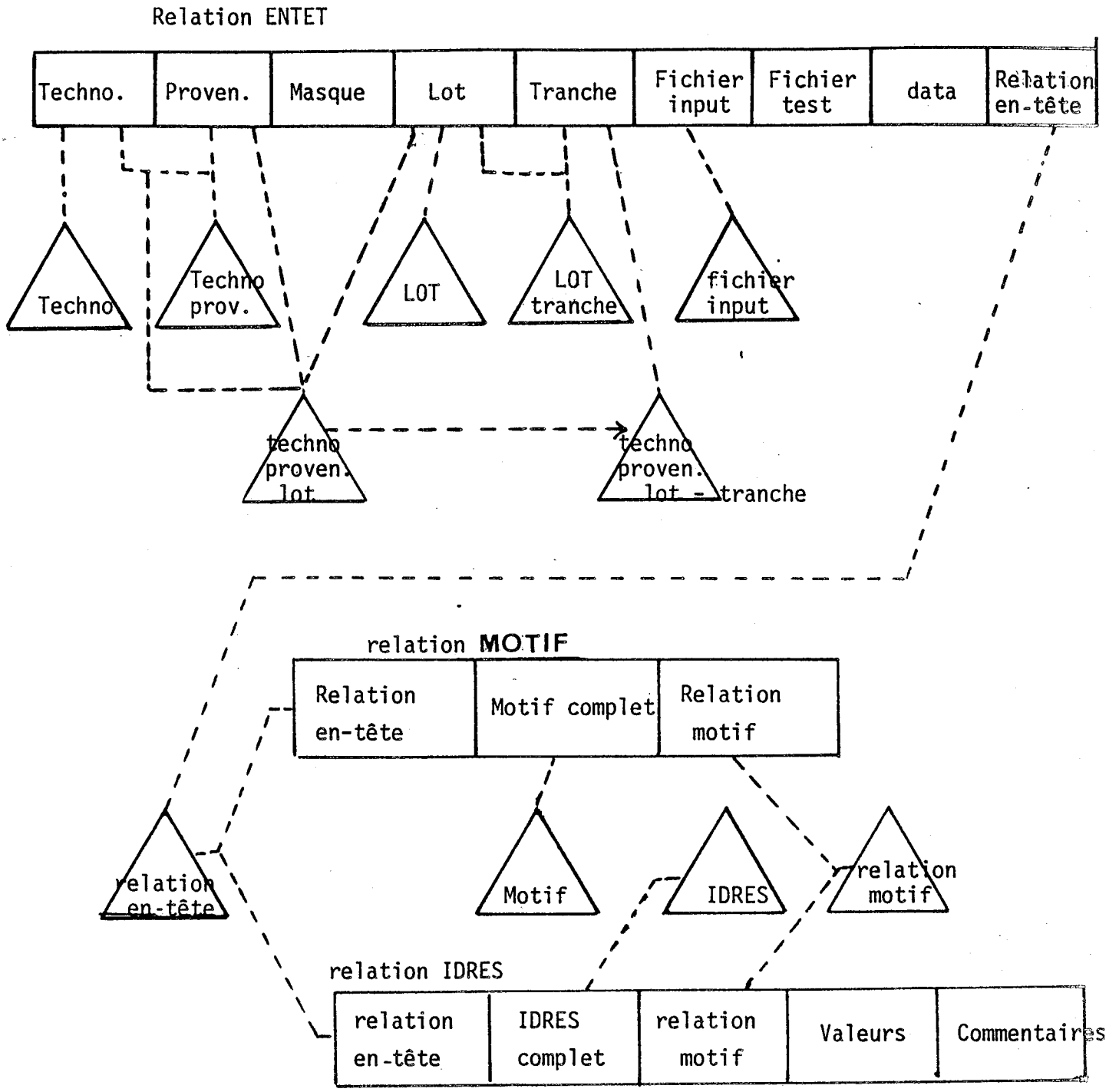


Fig.15 Schéma global de la base

I - 6. CONCLUSION

Ce chapitre a présenté l'implantation d'une interface permettant la manipulation en mode relationnel d'une base de données géréé par le SGBD IMAGE.

L'interface permet la transformation de la structure d'une base IMAGE afin que l'utilisateur puisse avoir une vue relationnelle de la base de données.

Pour réaliser cette transformation, nous avons défini un module de données (IMREL) rassemblant la plupart des concepts qui se retrouvent dans les deux modèles IMAGE et relationnel. Ce modèle permet d'établir les règles de correspondances qui sont à la base de la transformation.

Une approche mathématique est construite pour définir ce modèle qui a été développé pour interroger plusieurs bases de données, et pour déterminer les distances entre et dans deux relations pour avoir le schéma minimal.

BIBLIOGRAPHIE

(BEG 80) J. BEGUE
"Etude et réalisation d'une interface relationnelle sur un système REALITE 2000"
Rapport DEA, IIE, Paris, Février 1980

(BRI 81) F. BRIGES, K. HINANG
"PUMPS architecture for pattern analysis and IMAGE data base management"
IEEE 4 computer Patt. IMAGE, n°08, pp 178-187, 1981

(COD 82) E.F. CODD
"Relational data base : a practical foundation for productivity"
ACM Communication of the ACM, pp 109-117, Février 1982

(DAT 77) C.J. DATE
"An introduction to data base systems"
Addition Wesley Publish, 1977

(DEL 82) C. DELOBEL, M. ADIBA
"Bases de données et systèmes relationnels"
DUNOD informatique, Paris, 1982

(FU 76) K.S. FU, A. ROSENFELD
"Pattern recognition and IMAGE processing"
IEEE Transaction on Computers, Vol.25, n°12, pp 1339-1346? DECEMBRE 1976

(GAR 81) G. GARDARIN
"An introduction to SABRE : a Mult-Microprocessor data base Machine"
6th Workshop on Computer Architecture for non nomeric Processing, Hyeres, Juin 1981

(HAL 76) P.A.V. HALL
"Optimization of simple expression in a relational data base system"
IBM J. Res. Dev., pp 244-257, Mai 1976

(H/P 81) IMAGE/1000, H.P 92069 A, data base management system
Manual n°92069-90001, Update 2
HEWLETT/PACKARD, July 1981

(LOR 79) A. LORIE, J. NILSSON
"An access specification language for a relational data base system"
IBM J. Res. Dev., Vol.23, n°3, pp 286-298, Mai 1979

(McG 81) McGREE
"Data base technology"
IBM J. Res. Deve., n°5, pp 505-519, Septembre 1981

(SEO 81) K. SEO, A. MINEMATSU, H. AISO
"A look-Ahead data staging Architecture for relational data base machine"
IEEE Computer Architecture, pp 389-406, mai 1981

(SHO 81) Z. SHOUXUAN
"An approach to IMAGE data base organisation"
IEEE 4 Computer patt-IMAGE, pp 242-249, n°08, 1981

(SIR 81) SIRIUS
"Projet pilote sur les bases de données réparties"
Actes des Journées de présentation des résultats
Agence de l'Informatique, Paris, Novembre 1981

(TOM 80) F. TOMPA
"A pratical example of the specification of abstract data types"
Acta Informatica, Vol.13, pp 205-224, 1980

(YAM 81) R. YAMADA, K. TATEOKA, Y. NAKUE
"Dévelopment of a relational like data base management system for mini/micro computers"
IEEE 4 Computer sec., n°6, pp 403-410, 6-1981.

CHAPITRE II

LANGAGE DE REQUETE

I - INTRODUCTION

Les langages qui permettent d'interroger une base de données sont nombreux. Leur étude a montré qu'on ne peut approfondir l'étude des langages d'interrogation sans aborder le problème des modèles de données.

Comme nous l'avons précédemment mentionné, les trois modèles de structuration des données décrits par DATE (DAT 77) sont les plus utilisés qu'il s'agisse :

- du modèle relationnel où les données sont vues stockées sous forme de relations et d'attributs.
- du modèle hiérarchique où les données sont vues stockées sous forme de structure d'arbroscente.
- du modèle de réseau où les entitées sont vues stockées sous forme de structure de graphe.

Nous définissons un langage d'interrogation comme un langage spécialement conçu pour composer des requêtes ayant pour but de retrouver l'information, à partir d'une ou plusieurs bases de données stockées de type IMREL.

Avant d'aborder véritablement cette étude, nous présentons brièvement les langages d'interrogation les plus performants et en particulier QUERY et SQL.

Après nous cherchons à déterminer les conditions, les définitions, et les modalités d'un langage de requête permettant d'avoir un traitement optimal de donées vues sous modèle IMREL.

II - LANGAGE DE DONNEES DE HAUT NIVEAU

Ces dernières années les succès de l'informatique ont été marqué par un développement important des bases de données et donc les langages de manipulation de données, dû à leur impact dans les applications de gestion indutrielle, etc...

La technologie des bases de données est caractérisée par deux types de recherches. L'une ayant trait à l'amélioration de la définition des langages de données de haut niveau, l'autre aux langages de manipulation de données (LMD).

II - 1. Langage de Définition des Données (LDD)

Ce langage permet à l'utilisateur des systèmes de gestion de base de données (SGBD) de déclarer les attributs des structures de données à l'intérieur de la base utilisée. Ceci permet au système d'invoquer d'une façon implicite plusieurs opérations (par exemple : pour la technologie, la vérification du type de motif), ces dernières auraient du être autrement invoquées d'une façon explicite.

Normalement, un LDD permet la définition des entités, des attributs de la base de données, et des structures de données définissant les différents types de logiciels qui interrogent la base de données.

II - 2. Les Langages de Manipulation de Données (LMD)

Ils permettent à l'utilisateur d'envisager un schéma d'accès rapide vers les structures de données en utilisant des fichiers maîtres (clé) déjà établis au niveau de la définition de la base de données.

Les langages de manipulations de données sont rangés en deux classes :

a) les langages non procéduraux ou assertionnels, dits de "haut niveau"
b) les langages procéduraux dits de "bas niveau".

II - 2.1. Langages Procéduraux

Ils permettent la manipulation de la base de données par spécification d'une stratégie d'accès aux données suivie pas à pas. Ce langage peut employer un type de requête simple, mais dont l'utilisation requiert un formation préalable.

Le langage de manipulation QUERY est un exemple de ce type de langage.

II - 2.2. Langages non procéduraux "Assertionnels"

Ils permettent la manipulation des bases de données, par l'intermédiaire de programmes écrits dans des langages (FORTRAN, COBOL,...) et fournissent les instructions (statements), que l'utilisateur peut mettre dans un programme pour acceder a l'information dans les bases des données .

Quand une telle instrucion est rencontrée le contrôle est transféré au SGBD qui exécute l'opération par son chemin d'accès et renvoie les résultats au programme dans les unités de stockage prédéfinies.

Dans cette classe de langage, il existe quatre types d'approches au niveau de la manipulation :

a) Algébrique
b) Fondée sur le calcul relationnel
c) Mixte
d) Langage graphique

a) Langages Algérique
b

Dans un langage algébrique les requêtes consistent à définir une liste des opérateurs de l'algèbre relationnelle dont les opérandes sont des relations (DEL 82), tels que la jointure, la projection, la s elation, l'union et la différence.

Dans une expression relationnelle les seuls opérateurs sont les sélections, la projection, la jointure et l'union (SAG 80).

Un langage algérique manipule les relations comme des objets simples.

- ISBL (Information System Base Language) développé par le centre scientifique IBM (DEL 82), est utilisé dans le SGBD PRTV (Reterbre Relational Test Vehicule).

La requête ISBL est : soit la relation Techno (Technologie, Lot, Tranche, nom motif, n° relation) et la requête (trouver lot telque technologie = HMOSI et nom motif = résistance), en langage ISBL s'écrit :

```
LIST TECHNO : Technologie = "HMOS I" & nom motif = "Résistance" % LOT,
            : sélection % projection
```

- URANUS mis en oeuvre au Laboratoire IMAG de l'Université de Grenoble pour le système de base de données réparties POLYPHEME.

b) Langage basé sur le calcul relationnel

Il est constitué par le calcul des prédicats du premier ordre. On les appelle également langages prédicatifs.

Pour former les requêtes, on utilise la notion de quantifications (∀ et ∃) et le calcul des prédicats à variable domaine qui sont construcites selon les mêmes principes que les formules proposées dans (DEL 82).

Il est généralement admis que les langages prédicatifs sont moins procéduraux que les langages algébriques.

Parmi les langages de type calcul relationnel, on peut citer ALPHA, QUEL, SYNTEX.

- ALPHA : ce langage a été décrit par CODD (COD 71) c'est une adaptation du calcul relationnel aux langages conventionnels procéduraux (COD 81), la requête ISBL (déjà définie) s'écrit en langage ALPHA :

```
RANGE TECHNO X
GET (Y.LOT) : ∃ y((y. Technologie = "HMOS I")
              and (y. Nom motif = "Resistance"))
```

- SYNTEX : "SYstème d'iNTerrogation EXpérimental" est un langage d'interrogation de BD relationnelles dont le prototype a été réalisé au CERT à TOULOUSE(France) . Dans ce langage un type d'axiomes particuliers définit la structure des informations ; en effet, les requêtes posées à un système effectuent l'exécution de la partie logique quelque soit le type de prédicat.

c) Langages mixtes

Ces langages sont de plus en plus procéduraux, n'exigent pas de l'utilisateur de grandes connaissances en programmation.

Les langages de type calcul relationnel conviennent mieux aux logiciens.

Les deux générations de langages précédents avec leurs propriétés aboutissent après combinaison au troisième langage, ou le langage mixte. Ce dernier ayant la particularité de mieux convenir aux non informaticiens.

Ces langages allient les aspects algébriques et prédicatifs :

- GAMMA-0 : est un langage relationnel de bas niveau qui est destiné à exécuter les langages relationnels et les langages procéduraux.

- SQUARE : c'est un langage d'interrogation qui essaie par l'intermédiaire de conventions graphiques d'éviter certains aspects mathématiques du calcul relationnel. La requête précédente devrait être écrite en SQUARE comme suit :

nom de la relaion TECHNO qualification ("HMOS I", "Résistance")

attribut cherché LOT Technologie , nom motif

- SEQUEL : c'est un langage et une évolution du langage SQUARE initialement développé à IBM San José (DEL 82) dont l'idée de base était la construction d'une association entre consituants. L'écriture d'un bloc association se faisait sur 2 lignes ; l'une réservée aux noms des relations, l'autre aux noms des constituants.

La linéarisation de ce bloc d'associations a donné naissance au bloc de qualification du langage SEQUEL (DEL 82).

SEQUEL utilise les commandes SELECT, FROM et WHERE. Une caractéristique importante de SEQUEL, est de pouvoir développer la clause SELECT permettant des interrogations complexes sans perdre l'importante nature non-procédurale du langage.

Par exemple, l'interrogation pour la requête précédente s'écrit en SEQUEL :

```
SELECT    LOT
FROM      TECHNO
WHERE     TECHNOLOGIE  = HMOS I
      AND NOM-MOTIF    = Résistance
```

- SQL : c'est un langage développé pour le système R d'IBM et une version enrichie SEQUEL.

En plus des facilités d'interrogation de SEQUEL. SQL fournit :

- des facilités de manipulation de données qui permettent la sélection, la suppression et la modification les informations d'une base,
- facilités de définition de données pour exprimer les relations.
- facilités de contrôle des données pour définir le schéma d'accès et les transactions.

L'utilisation du langage SQL permet de construire un programme simplifié tout en permettant à des variables du langage d'apparaître dans les instructions SQL, et de faciliter la manipulation des tuples dans les bases de données.

Avec ces propriétés le langage SQL apparaît comme un langage d'interrogation de modèle IMREL.

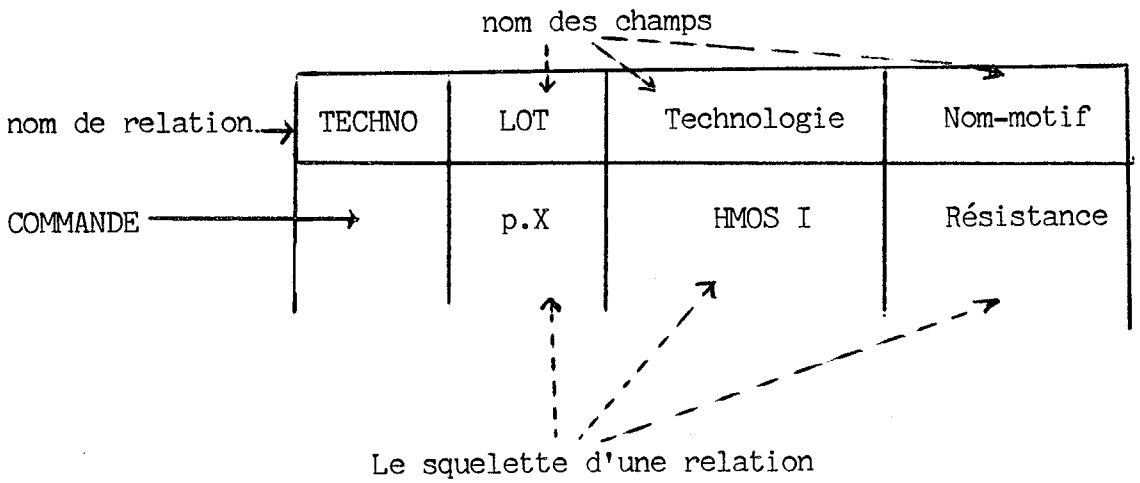
Ce langage va être défini au paragraphe II- 4 de même que l'on va définir toutes les propositions qui permettent d'utiliser ce type de langage.

d) Langages graphiques

Le plus connu est le langage QUERY BY EXAMPLE (QBE), qui a été développé par IBM (ZLO 77) si ce langage peut être considéré pour une large part, comme un langage du type prédicatif à variables domaines, il contient des caractéristiques spécifiques dans lesquelles l'originalité réside dans l'expression graphique des requêtes : l'utilisateur fait afficher à l'écran des tableaux représentant des squelettes de relation.

L'utilisateur dispose d'une commande lui permettant de faire apparaître le squelette, la structure de cette commande ensuite les colonnes dont il a besoin, il a aussi la possibilité de faire générer ces tableaux par le système afin d'éviter par exemple des erreurs type graphiques. L'utilisateur donne un exemple de la solution qu'il désire, une distinction est faite entre une constante, et une variable, (les variables sont soulignées alors que les constantes ne le sont pas).

La requête précédente devient :



Le squelette d'une relation

La lettre 'p' dans les colonnes indiquent les attributs dont les réalisations doivent être imprimés.

Autre langage graphique INGRID (une INterface GRaphique d'Interrogation base de Données), celle-ci transformant toute requête utilisateur en un ensemble d'appels à des primitives MIMER. Il répond aux trois caractéristiques suivantes : relationnel, graphique et Interactif (FER 82).

Tous les objectifs de la base de données sont représentés sur l'écran sous forme de tableau.

II - 3. PARTICULARITES DU LANGAGE DE REQUETE QUERY

Ce type de langage a été décrit par H.P. Celui ci consiste en une chaine de programmes qui permettent l'accès aux données stockées sous forme IMAGE. Ce langage IMAGE permet de décrire le modèle en structures simples dans une seule base de données (IMA 81).

Ce langage est décrit dans le paragraphe suivant.

II - 3.1. Types de requêtes

a) Requête FIND

Pour sélectionner quelques informations dans la base, l'utilisateur peut utilisé la commande FIND.

FIND est un type de commande qui contient les caractéristiques suivantes :

- pouvoir chercher dans (15) items de données différentes
- pouvoir stocker la requête comme une procédure sur disque afin de l'appeler au besoin
- on applique la commande FIND pour minimiser le temps de recherche par l'utilité de fichier maître
- pour la même commande FIND on peut chercher sur plusieurs items de données
- la commande FIND fournit un rapport sur des données déjà stockées.

Utilisation de la commande FIND

Nous avons vu que la commande FIND a fourni un rapport de données déjà entrées c'est ainsi que pour arriver à ces données on applique le type de requêtes suivantes :

FIND Set.Item Operator "VALUE" END ;

Dans la commande FIND, on peut trouver plusieurs procédures du type "RETROUVER".

Quand un item existe dans plusieurs fichiers, il sera nécessaire d'examiner les items clés pour trouver le fichier que nous cherchons.

En général si la requête FIND ne contient aucun item clé ou aucun opérateur relationnel la réalisation de la recherche se fait en série.

Les opérateurs relationnels sont IS, IE (égal à) , ISNOT,INE (non égal à), ILT(plus petit que), INLT (n'est pas plus petit que),IGT(est plus grand que), INGT(n'est pas plus grand que).

b) La requête MODIF

Si nous a besoin de modifier des entrées dans la base de données, on peut utiliser la commande UPDATE de plusieurs manières.

En utilisant la commande UPDATE ADD (ou UPDATE A) pour faire la mise à jour des données de la base en utilisant soit par un maître manuel soit par un fichier détail, la forme de la commande est :

UPDATE A(DD), (nom de fichier données), ou
UPDATE NAME = nom de procédure

avec la commande UPDATE delete (ou UPDATE D) nous peut supprimer une ou plusieurs entrées de l'ensemble des données en passant soit par un Maître manuel soit par un fichier détail. La forme de la commande est :

UPDATE D (ELETE); ou
UPDATE NAME = nom procédure

La commande UPDATE REPLACE(ou UPDATE R) sert à remplir une valeur spécifique dans un ou plusieurs item de données et qui est déjà rentrée avec une autre valeur spécifique. La forme de commande est :

UPDATE R(REPLA); ou
UPDATE NAME = nom procédure.

c) La requête REPORT

La requête REPORT permet d'imprimer les données déjà stockées dans le fichier déjà sélectionnées. Ce type de requête peut être utilisé à différents niveaux ; pour faire plusieurs types de travaux par exemple, faire un rapport des noms et des valeurs pour tous les items des données dans la base sous certaines conditions :

- Trouver des signes ou des mots qui sont stockés dans une base
- Faire des calculs des statistiques à la moyenne, le total pour des ensembles de données.

La requête REPORT remplit la même tâche que celle de FIND. La forme de la requête REPORT est :

```
REPORT ELL (,character) ; ou
REPORT NAME = nom de procédure,

   REPORT
ou CORPS
   END
```

Le résultat de la commande REPORT sera imprimé dans une liste divisée dont la forme de sortie peut être modélisée avec les expressions de contrôle appliquées sur la sortie des données.

II - 3.2. Utilisation de QUERY

A l'aide d'un exemple, nous verrons comment utiliser QUERY à l'aide d'un terminal.

Soit le schéma d'une base de données définie type IMAGE :

```
* base de données "Base 1" *

BEGIN  DATA BASE : base 1 : 100 : 58 ;
LEVELS :

    1 READ ;
    2 WRITE ;
    5 UPDATE ;

ITEMS          * les items suivants décrivent le fichier 1 *
    ENT 01          X 20(1,15) * technologie *
    ENT 02          X 20(1,05) * MASQUE *

                   * les items suivants décrivent le fichier 2 *

    ENT 04          X 30(1,15) * item clé ...*

SET
    NAME ; DB 1001 :: 58,A     * fichier automatique techno *
    ENTRY : ENT 01(01),
    CAPACITY
        :
        :
END.
```

Le dialogue entre l'utilisateur et le système pour utiliser QUERY peut se représenter par le tableau suivant :

	Système	Utilisateur (H.P/1000)
Appel de QUERY		QUERY
Ouverture de la base	NEXT ?	DATA-BASE = base 1 : 100 : 58
niveau d'accès	LEVEL ?	
	MODE	
fichier de travail	NEXT ?	SELECT-FILE = TOTO :: 53 :
trouver technologie=xxx		FIND DB 1001.ENT 01 IS "xxx" END ;
résultat		
sortie sur imprimante	NEXT ?	LIST = 6 ;
sortie des résultats	NEXT ?	REPORT ALL ;
sans aucune déclaration		
de format		
les résultats avec entete	NEXT ?	REPORT ,
H 1ère ligne, "déclaration",n°colonne finale		H1, "entete ", 50 ;
H 2ème ligne		H2, "xx xx xxx", 62 ;
fin condition de		
rapport		
		END :
résultat		
élimination	NEXT ?	UPDATE D :
	NEXT ?	FIND DB 1001.ENT 01 IS "xx1" END :
fin utilisation QUERY	NEXT ?	EXIST ?

II - 4. LANGAGE 'SQL'

II - 4.1. Présentation

SQL résulte d'une évolution du langage SQUARE, développé pour le SGBD Système R à IBM San José (AST 75)(CHA 76)(CHA 80)(CHA 81)(COD 82).

Le nom original de SQL est SEQUEL (Structure English QUEry Language).

SQL peut être utilisé comme une interface ad-hoc pour les non-spécialistes, ou comme un sous-langage utilisé dans un langage hôte (PL 1, FORTRAN, ..) servant aux programmes d'application.

La structure de base du langage est le bloc de qualification. Sa forme est du type suivant :

```
SELECT   A,A2,...,Aj
FROM     R
WHERE    F
```

ou A1,...,Aj attributs R une relation et F une expression conditionnelle, pouvant imbriquer d'autres blocs select. La clause SELECT définit la liste des attributs que nous voulons obtenir.

La clause FROM indique la relation concernée par la requête. La clause WHERE donne la condition F qui doit satisfaire un n-uplet de la relation R pour être sélectionné.

La requête :

```
SELECT   *
FROM     R
```

donne tous les n-uplets de la relation R

En général, le bloc de qualification retrouve un ensemble de réalisations des attributs dont les n-uplets vérifient la clause WHERE.

II - 4.2. Imbrication des blocs de qualification

La clause WHERE peut être utilisée pour tester l'existence d'une valeur dans un ensemble. Dans ce cas, on utilise l'opérateur d'appartenance ensembliste IN. Ceci permet d'imbriquer les blocs de qualification, pour finalement amener les résultats d'un bloc à l'autre.

Exemple : soit les deux relations

Techno (Technologie, Lot, tranche, n°relation)
Motif (n° relation, N-motif, méthode D data)

et la requête :

SELECT Lot TELQUE N-MOTIF "toto" ;

La requête s'écrit :

bloc externe	SELECT	LOT		
	FROM	Techno		
	WHERE	n° relation IN	SELECT	n° relation motif
		bloc interne	FROM	Motif
			WHERE	nom-motif "TOTO"

L'opérateur IN est parfois remplacé par l'opérateur de comparaison =,

II - 4.3. Opérateur d'ensemble :

- opérateur d'évaluation.

SQL offre des fonctions agissant sur un ensemble d'information :

COUNT : pour le dénombrement des éléments
SUM : additionne les éléments
AVG : calcule la moyenne
MAX : définit le maximum
MIN : définit le minimum

Ces fonctions peuvent apparaître soit dans la clause SELECT, soit dans la clause WHERE.

Opérateurs usuels

- les opérateurs UNION

INTERSECT, MINUS figurant dans SQL sont essentiellement destinés à combiner les résultats de deux blocs de qualification. Ils éliminent automatiquement les éléments dupliqués dans les opérations avant leur application.

- Opérateur de comparaison

Dans un sous-ensemble les relations ayant les mêmes valeurs sont regroupées par l'opérateur : GROUP BY;

Comme opérateurs de comparaison SQL dispose [:- :=, ≠, (IS)(NOT)IN, CONTAINS, DOES NOT CONTAINS].

Utilisation d'une variable n-uplet

Il est possible d'avoir un nom de variable n-uplet qui décrit une relation à l'aide de l'expression :

```
SELECT   A1 A2 ... AJ
FROM     R  P
WHERE    F
```

P est le nom de la variable n-uplet qui parcourt la relation R.

Exemple : soit les deux relations suivantes :

R1 = (Technologie, Lot, PROVENANCE, TRANCHE, n° relation)
R2 = (N° motif, nom motif, méthode, sous-types)

La requête : SELECT, Technologie, TELQUE méthode "classique"; la requête sous forme SQL s'écrit :

```
SELECT   Technologie
FROM     R1 P
WHERE    n°motif    SELECT    n° motif
                    FROM      R2
                    WHERE     méthode = classique
```

- Expression de jointure

Une requête peut définir des valeurs sélectionnées dans plusieurs relations. L'utilisateur peut avoir plusieurs relations dans la classe FROM. Il doit faire attention à la spécification des attributs d'intersection. Un attribut apparaissant dans une seule relation peut être non qualifié par le nom de la relation.

Supposons que d'après le schéma ci-dessus, l'utilisateur désire connaître la tranche et le sous-type.

La requête se forme de la façon suivante :

```
SELECT   Tranche, sous-type
FROM     R1,R2
WHERE    R1.N°relation = R2.N°motif
```

La condition R1.N°relaion = R2.N°motif traduit le JOINT-naturel R1*R2.

La même requête peut être désignée par :

```
SELECT   Tranche
FROM     R1
WHERE    sous-type   SELECT    sous-type
                     FROM      R2
                     WHERE     N°relation = N°motif
```

II - 4.4. Conclusion

L'utilisation des programmes dans SQL simplifie l'interrogation des données. Les avantages de cette utilisation sont :

- Facilité à manipuler les données en sélection, en modification et en suppression.
- Facilité de définition du schéma ou des relations.

Le langage SQL :

- a été construit autour de blocs de qualifications pouvant s'imbriquer,
- sa simplicité est une des caractéristiques fondamentales de ce langage.

II - 5. SPECIFICATION DU LANGAGE DE REQUETE

II - 5.1. Introduction

Basée sur l'architecture du modèle IMREL que nous avons proposé en (I - 3), nous adoptons pour notre requête le modèle relationnel comme le modèle de conception global.

Il est nécessaire de donner un schéma relationnel pour chaque base de données. Pour la base de données dont les modèles ne sont pas relationnels.

Nous avons besoin d'un schéma de traduction spécifique pour un modèle de données spécifiques. Ceci permettra la traduction d'un schéma vers un schéma relationnel.

Les vues sont composées d'informations concernant la requête d'interrogation. Ces vues peuvent être considérées comme une base de données réduite.

Il est nécessaire de donner les règles de traduction des opérateurs relationnels en des instructions des langages de manipulation de données stockées dans plusieurs bases.

Dans notre méthode chaque système de base de données est présenté à l'utilisateur avec un schéma relationnel global, la requête donne le schéma d'accès aux données ou faisant leur mise à jour.

Le langage de requête que nous proposons utilise les opérateurs de l'algèbre relationnelle sous une forme syntaxique. En pratique la syntaxe doit être compatible avec le langage de programmation utilisé quelque soit ce langage.

Nous avons utilisé la syntaxe de SQL expliquée en (II - 4). On reprend l'exemple du chapitre précédent, la liste des relations spécifiques par la clause FROM, la liste des objets spécifiés par la clause SELECT et la liste de qualifications définie par la clause WHERE.

L'optimisation de requête dans le système IMREL est basé sur l'optimisation des opérateurs algébriques. Pour exécuter ces opérateurs dans le SGBD local, nous devons faire la transformation des opérateurs algébriques en une chaine de programmes.

Nous supposons dans notre travail que l'exécution entre deux ensembles (relation, base, fichier) ne peut être prise qu'après l'exécution dans chaque base locale.

Les résultats seront stockés sous une forme relationnelle dans une zone de travail réservée par le programme pour chaque base. Notons qu'une équivalence (FORTRAN) existe entre ces zones de travail pour réduire la place occupée en mémoire. Ces propositions nous aident dans l'exécution des jointures entre les bases.

Dans la section suivante, nous allons définir la modélisation des requêtes et leur structure et leur portée locale ou globale.

II - 5.2. Modélisation de requête optimale

Il est nécessaire d'après le système IMREL, qui définit notre modèle de base, nous devons chercher une méthode qui permette d'optimiser les interrogations sur une ou plusieurs bases, de façon à minimiser le temps de réponse.

Notre système évite l'obligation d'avoir un autre système, pour traiter plusieurs bases de données. Avec le modèle IMAGE qui est développé dans le modèle IMREL et notre modèle de langage de requête nous pouvons interroger plusieurs bases en même temps.

Le temps de réponse est directement fonction du temps d'accès à l'information, lui même dépendant du nombre d'enregistrements examinés et de la nature du point d'entrée (clé, sous-clef, colonne) dans le cas de la jointure.

Notre propos est de modéliser la requête de façon très simple (langage naturel), par contre, la majorité des requêtes dans les langages relationnels sont formées de trois étapes ; la liste des relations (FROM), la liste des objets (SELECT) et la classe des conditions (WHERE).

Dans notre cas l'analyseur syntaxique définit le nom de la base et le nom des relations que contiennent les objets recherchées, sans fixer la liste (FROM). On se propose de modéliser les requêtes de la syntaxe suivante :

OPERATION xxx xxx .... xxx TELQUE xxx `bbb' xxx `bbb' .... xxx `bbb' ;

(xxx xxx .... xxx : objet recherché ; xxx `bbb' xxx `bbb' .... xxx `bbb' : conditions à réaliser (Références))

- opération : Après l'analyse syntaxique sur le premier mot de la requête, nous allons préciser si le type d'opération est une manipulation d'information (Sélectionner, Modifier, Supprimer,...) ou est une manipulation de structure de base (Créer, Détruire...) sur tous les niveaux : base, fichier (Relation), colonne ou présentation de structure et de fichier (Listes...)

- objet recherché : pour préciser le nom des colonnes qui contiennent les informations à rechercher.

- la condition : chaque condition est précisée par deux mots : le premier pour préciser le nom de la colonne, le deuxième pour préciser la valeur de la condition, on va séparer la condition et l'objet par le mot (TELQUE), et deux quelconques par au moins un espace blanc.

Pour chaque sous-requête, nous allons déterminer un organigramme de l'étape traitée par la requête.

1°) nous allons découper la requête en sous-requêtes concernant une seule base, soit Q la requête et q1,q2...qn les sous-requêtes "n" étant le numéro de la base ou de la relation.

2°) nous allons estimer la taille du résultat au niveau de chaque sous-requête, et on définit la nature du point d'entrée.

3°) nous allons déterminer la priorité de passage de chaque sous-requête en utilisant l'analyse sémantique.

4°) nous allons appliquer le traitement local et après global en utilisant les opérateurs relationnels.

Avant de passer au traitement, on va étudier les diffrentes commandes du langage.

II - 5.3. Les différentes commandes du langage de requête

Les commandes sont classées en trois types :

a) Commande de manipulation de données

* SELECT : interrogation des bases de données
* MODIFIER : mise à jour d'item dans un fichier
* SUPPRIMER-DE : suppression d'un item dans un fichier
* INSERER-DANS : chargée la base par des items faits par un programme spécial.

b) Commandes qui permettent d'avoir des informations sur les bases de données

* LISTER-FICHIER : edition des identifications de toutes les colonnes pour une base
* LISTER-COLON : edition des différentes valeurs d'un index donné

c) Commande de manipulation de la base de données

* OUVRIR-BASE : permet l'ouverture de la base de données pour une session
* FERMER-BASE : permet la fermeture de la base après une session
* OUVRIR-FICHIER : permet l'ouverture d'un fichier pour une session
* FERMER-FICHIER : permet la fermeture d'un fichier après une session

Dans notre cas, on va partager les types de requêtes proposées en deux ensembles.

1 - Requête composée

C'est l'occurence d'un événement attribut sur les valeurs de données, ces types de requête seront traitées en trois étapes.

Ces requêtes sont : select, modifier, supprimer, et insertion.

2 - Requête simple

C'est l'occurence d'un événement-attribut sur une autre action par exemple, à la fin du traitement on va fermer la base de données utilisée, ainsi que les fichiers de cette base.

Ces types de requêtes seront compilés directement dans la première étape. Ces types sont de deux sortes :

a) requête d'action : exécutée sur les données d'une ou plusieurs bases de données comme : lister-fichier et lister-colon.

b) Requête de manipulation de la base permet de disposer d'un accès à la base, comme : ouvrir-base, fermer-base, niveau utilisateurs, ouvrir-fichier et fermer-fichier niveau ci-dessous.

II - 5.4. Structure de l'automate généré pour une requête

Schématiquement l'automate est un couple (Q,L) : Q est l'ensemble des états de l'automate, L est une application de SxQ dans f(Q) où S représente l'ensemble des éléments du langage (identificateurs, mots-clés).

f est la fonction de transition, son rôle est d'indiquer, pour un état courant q de l'automate, quels sont les états suivants (q2...qn) possible lors de l'analyse d'un élément u, on note ainsi : f(q,u) = (q2,q3...qn).

L'automate est dit d'états finis si Q est fini et déterministe lorsque f(q,u) est un singleton quelque soit le couple (q,u).

Autrement dit un état et un élément étant donné l'état suivant est défini sans ambiguité.

Nous représentons généralement notre requête au moyen d'un graphe orienté (Figure 16). Les noeuds de ce graphe sont les différents états de l'automate.

II - 5.5. La méthode de traitement des requêtes

Une requête exprimée dans un langage de manipulation de donées IMREL comprend en général une ou plusieurs conditions de sélection exprimées à partir de conditions simples entre les attributs reliés par des opérateurs ensemblistes (ET,OU).

En dehors,des contrôles de validité d'ordre syntaxique et sémantique, le traitement de la requête doit franchir un certain nombre d'étapes :

- vérification de la connexité entre les attributs de la requête (BRA 80)
- restructuration éventuelle par application de l'algèbre relationnelle
- décomposition éventuelle de la requête en sous-requêtes plus simples au niveau de chaque base (BAN 80)(WON 76)(CER 82)

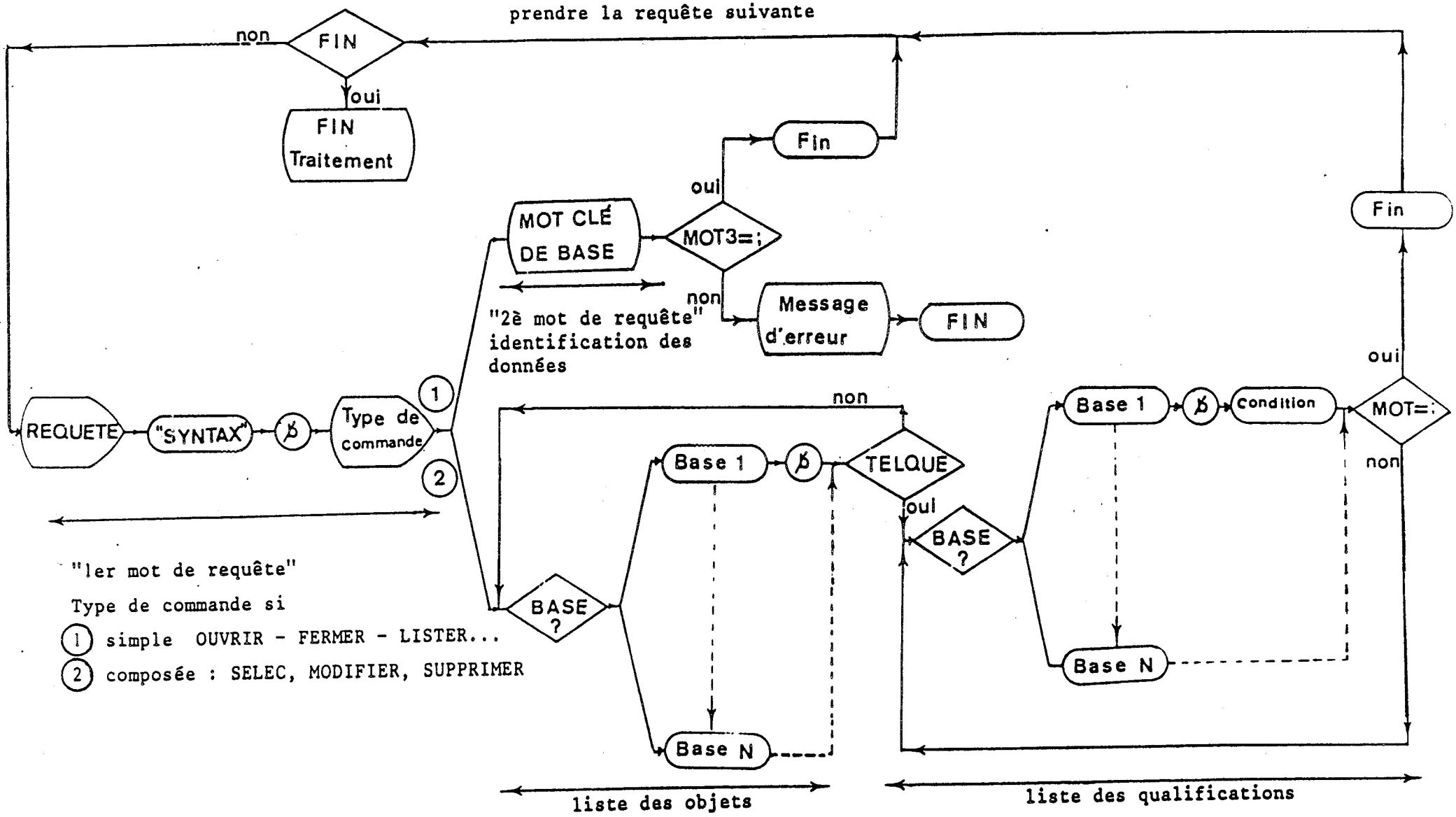


Fig 16 STRUCTURE DE L'AUTOMATE GENERE POUR UNE REQUETE

- optimisation du coût des réponses (YU 78)(BIN 79)(CHI 78)(CHU 82)(SIL 73)
- construction d'un plan d'exécution indiquant l'ordre dans lequel on va examiner ces sous-requêtes.

Nous allons détailler les étapes précédentes au moment de la costruction du compilateur. Tout d'abord nous définissons :

a) les définitions utilisées pour partager la requête en sous-requêtes, chaque sous-requête concerne une seule base, elle est modélisée de la même façon que la requête principale.

b) Choix d'un algorithme pour l'algébre relationnelle.

A) Décomposition d'une requête

Soit R une requête, on appelle sous-requête "s" sur R, toute application "s" de R sur B (B est une relation de l'ensemble des relations vers une ou plusieurs bases).

Soit s1,s2,...sn l'ensemble des sous-requêtes sur la requête R, qui vérifie:

$s1 \cup s2 \cup \ldots \cup sn = R$

$s1 \cap s2 \cap \ldots \cap sn = \{A \mid A$ est l'ensemble des colonnes qui réalisent les jointures entre les relations $\}$

où "A" est constitué d'au moins une colonne.

. Le couple (R,s) est appelé l'espace des requêtes
. soit R une requête et s1, s2 deux sous-requêtes sur R, on dit que s1 est plus fine que s2 si et seulement si :

$\forall A, A \subset B1, A \subset B2$, on a

$s1\ A \subset s2\ A$

Nous traitons toujours la sous-requête la plus fine.

Proposition 1

Soit 'R' une requête, l'ensemble "s" des sous-requêtes sur R, le produit "."

A tout couple (si, sj) de s on associe

si . sj = sij

VA, A⊂ B1, A ⊂B2 on définit

s12 A = s1(s2 A)

Proposition 2

Soient R une requête et s1,s2 deux sous-requêtes de (s)

s1 est plus fine que(s1.s2)

Démonstration

Soit A ⊂B alors A⊆ s2 A par suite S1 A⊆ s1(s2 A) est le résultat.

B) Choix d'algorithme d'algèbre relationnelle

Le langage algébrique définit des informations qui peuvent s'exprimer sous un aspect relationnel par des applications d'opérateurs algébriques parmi ceux qu'il est possible d'effectuer sur une ou plusieurs relations, mais nous nous limiterons à l'étude des opérateurs de base suivants :

* SELECT choix des tuples d'une relation
* PROJECTION choix d'une relation sur certains attributs
* JOINTURE lien entre deux relations sur un attribut commun.

Les raisons qui nous ont poussé à choisir ces opérateurs sont les suivants:

* Les opérateurs SELECT et PROJECTION permettentde réaliser les opérateurs courants de restriction de cardinalité et de largeur d'une relation.

* L'opérateur de JOINTURE a été préféré à l'opérateur de produit cartésien car il permet un parcours plus efficace des relations. Dans le cas du produit cartésien le nombre de tuples de la relation résultante peut être très important (m x n tuples où m taille R1, n taille R2).

* Les opérateurs SELECT et PROJECTION sont relativement simples à réaliser. De plus il est intéressant que ces opérateurs soient exécutés le plus tôt possible, car en réduisant respectivement la cardinalité et la largeur d'une relation, ils réduisent la taille des résultats intermédiaires.

Par contre, et notamment dans le cas de relations de taille importante, l'exécution d'une JOINTURE entre deux relations peut nécessiter des délais non négligeables. Le temps de réponse d'un système distribué peut s'en trouver sérieusement affecté.

Il est donc nécessaire de pouvoir disposer d'algorithmes performants pour réaliser cet opérateur (HAK 80).

Choix d'un algorithme

La détermination d'un algorithme revient au choix d'un parcours respectant certains critères, l'un de ces critères est l'optimisation d'un chemin minimal.

Notre algorithme détermine une série de tableaux intermédiaires dont on va parler dans le chapitre IV, l'exécution des opérateurs SELECT et PROJECTION au niveau local, l'exécution des opérateurs de JOINTURE.

La JOINTURE des relations s'applique de façon optimale si on remplace au cours de l'application la JOINTURE par une P-JOINTURE (voir chapitre III). Maintenant on définit l'algorithme de jointure.

La jointure entre les relations n'est pas une opération isolée, mais c'est une partie fondamentale d'une action afin d'arriver à la réponse de la requête posée.

Il existe de nombreuses méthodes d'étude de cet algorithme (VAL 82)(AHO 79) (LOR 79).

On constate que la forme mathématique de la JOINTURE entre les deux relations R, R' est

$$R[A = B]R'$$

Cela signifie que la JOINTURE sera appliquée sur deux colonnes A et B dans les relations R, R' successivement.

Type des relations

* deux relations non triées
* deux relations triées
* une des deux relations est triée.

Type de jointure

1° - si les données dans R,R' sont de taille suffisante, on pourra les trier en mémoire, on va suivre la méthode de résolution suivante :

On va considérer que R représente la colonne A et R' représente la colonne B et on va trier R et R' puis on applique l'algorithme suivant :

1 - I = 1 J = 1 m = 0

2 - K = 0

3 - si R < R' Alors I = I + 1 si (fin R) alors aller en 5
sinon aller en 2

4 - sinon : si R = R' alors K = K + 1
stocké ligne (I,J)
J = J + 1
si (fin R') Alors M = 1
I = I - 1
K = K - 1
aller en 5
sinon aller en 4

sinon si K = 0 alors J = J + 1
aller en 4
sinon aller en 5

5 - Fin

2° - si les données dans R' et R sont telles qu'il n'est pas possible de les traiter en mémoire, on applique la méthode suivante :

Au lieu de réaliser la JOINTURE extérieurement, on segmente R et R' intérieurement par l'utilisateur d'un fichier temporaire (M.S)

A la fin on trouve deux relations triées de la forme :

Y11, Y12, ..., Y1n
Y21, Y22, ..., Y2m de façon que:

$$R = \bigcup_{i=1}^{n} Y1i$$

$$R' = \bigcup_{j=1}^{m} Y2j$$

Yij sera mémorisée dans la M.S

On considère que les Yij sont triées par ordre croissant et on va mettre au moins un enregistrement de Yij dans la MC.

Après on fixe la valeur minimale pour les tuples de JOINTURE. De la forme :

* min 1 = min(Jointure valeur de Y1i)
  i = 1, ..., n

* min 2 = min(Jointure valeur de Y2j)
  j = 1, ..., m

Après observation des valeurs on prend le minimum des deux valeurs (min 1, min2). On regarde si la relation correspondant au minimum peut être stockée en mémoire, sinon on cherchera à ne stocker que la colonne correspondant à la jointure.

Si la colonne peut être stockée, on s'est ramené au cas précédent et on peut exécuter la jointure, sinon on est amené à découper la relation et à faire des jointures par partie.

II - 6. CONCLUSION

Le présent chapitre a montré la grande diversité des langages qui ont été développé autour des SGBD, ils s'inspirent largement des principaux modèles que nous avons vus. Avec le langage QUERY pour interroger les bases IMAGE et le langage SQL pour interroger les bases relationnelles.

Ils nous conduisent dans notre système IMREL à avoir un langage ayant pour rôle de reprendre les requêtes du langage relationnel (écrites en langage SQL développé) pour les traduire dans le langage du SGBD IMAGE.

La constitution de la requête IMREL se fait de façon à permettre de retrouver les chemins d'accès qu'on doit suivre pour accéder aux données.

Les modifications apportées dans SQL n'obligent pas l'utilisateur à spécifier les relations qui contiennent les attributs définis dans la requête, pour ce faire, un analyseur syntaxique est développé pour placer la clause WHERE dans la requête SQL. Ce type de requête est un outil efficace au service d'utilisateurs non-informaticiens.

BIBLIOGRAPHIE

(AHO 79) A. AHO, J. ULLMAN
"The theory of joins in relational data base"
ACM Transaction on data base systems, Vol 4, n°3, pp 297-314, Septembre 1979

(AST 80) M.M. ASTRAMAN, M. SCHKOLNICK, W. KIM
"Performance of the system R access path selection mechanism"
Congrès IFIP 1980, North Holland publ., pp 487-491, 1980

(BAN 80) J. BANERJEE, D. HSIAO, F. NG
"Data base transformation QUERY Translation and performance analysis of new data base computer in supporting hierarchical data base management"
IEEE Transaction on Software engineering, Vol SE 6, n°1, pp 91-109, January 1980

(BEL 81) Z. BELLAHSENE
"Méthodes d'évaluation de question dans les systèmes Monoprocesseurs"
Rapport : projet SABRE, INRIA, 1981

(BIN 79) S. BING YAO
"Optimization of QUERY evaluation algorithms"
ACM Transaction on DBS, Vol 4, n°2, PP 133-155, Juin 1979

(BRA 80) A. BRANSTAT, C. CHERNIAVSKY, A.N. RICHARDS
"Validation, Verification and testing for the individual programmer"
IEEE Computer, pp 24-30, Décembre 1980

(CER 82) S. CERI, G. PELAGATTI
"Allocation of operations in distributed data base access"
IEEE Transaction Computers, Vol C 31, n°2, Février 1982

(CHA 76) D.D. CHAMBERLIN et al.
"SEQUEL 2 : A unified approach to data definition manipulation and control"
IBM J. of Res and Develop., pp 560-575, Mars 1976

(CHA 80) D.D CHAMBERLIN
"A summary of user experience with the SQL data sublanguage"
Proceedings of the international conference on data base, University of ABerdenne, pp 181-203, Juin 1980

(CHA 81) D.D. CHAMBERLIN, A.M. GILBERT, R.A. YOST
"A history of system R and SQL/DATA system"
7th international conference on very large database, Cannes, France, pp 456-463, Septembre 1981

(CHI 78) F. CHIN
"Security in statistical database for QUERY with small counts"
ACM Transactions on DBS, Vol 3, n°1, pp 92-104, Mars 1978

(CHU 82) W. CHU, R. HURLEY
"Optimal QUERY processing for distributed database system"
IEEE Transaction on computers, Vol C 31, n°9, septembre 1982

(COD 72) E.F. CODD
"Relational completeness of data base sublanguages"
Data base systems, Prentice Hall, pp 65-98, 1972

(COD 82) E.F. CODD
"Relational data base : a practical fonndotion for productivity"
Comm. of ACM, Vol 25, n°2, pp 109-117, Février 1982

(DAT 77) C.J. DATE
"An introduction to data base systems"
Additional Wesly, Publishing Company, Second edition 1977

(DEL 82) C. DELOBEL, M. ADIBA
"Bases de données et systèmes relationnels"
Dunod Informatique, 1982

(FER 82) L. FERRAT, J.C. GUILLAUME, M. MOUTAMANI
"INGRID, MIQUEL et PASREL : les trois interfaces relationnelles de MICROBE"
IMAG Grenoble, Nov. 1982.

(HAK 80) M.N HAKIM
"Evaluation de requêtes dans les systèmes relaionnels"
Rapport DEA, Lyon 1980

(IMA 81) IMAGE/1000 HP 92069
"A data base management system"
Mannal n°92069-90001
Hewlelt/Packard, Juillet 1981

(LOR 79) A. LORIE, J. NELSON
"An access specification language for a relational database system"
IBM J. Res. et Develop., Vol 23, n°3, pp 286-298, Mai 1979

(SAG 80) V. SAGIV, M. YANNAKAKIS
"Equivalence among relational expression with the union and difference operators"
J. of ACM, Vol 27, n°4, pp 633-655, Octobre 1980

(SIL 73) H.F. SILVERMAN, P.C YUG
"Response time characterization of an information retrival system"
IBM J. Res. and Develop., pp 394-403, Septembre 1973

(VAL 82) P. VALDURIEZ, G. GARDARIN
"Multiprocessor JOIN algorithms of relations"
Rapport n° MBD-I-010, INRIA France, 1982

(WON 76) E. WONG, K. YOUSSEFI
"Decomposition strategy forQUERY processing"
ACM Transaction on DBS, Vol 1, n°3, pp 223-241, Septembre 1976

(YU 78) C.T. YU, W.S. LUK, M.K. SIU
"On the estimation of the nombre of desired Records with respet to a given query"
ACM Transaction on DBS, Vol 3, n°1, pp 41-56, Mars 1978

(ZLO 77) M.N. ZLOOF
"Queru-By-Example : a data base language"
IBM J. System, n°24, pp 324-343, 1977

CHAPITRE III

OPTIMISATION DE REQUETES

III - 1. INTRODUCTION

Avant d'aborder la réalisation effective du compilateur de requêtes (chapitre IV) nous allons nous intéresser à l'optimisation des requêtes. Après un rappel sur les méthodes utilisées dans différents systèmes, nous présenterons celle qui a été retenue et implantée dans notre compilateur.

Le mot "optimisation" convient mal en fait car la requête "optimisée" n'est pas toujours la meilleure possible. Les problèmes d'optimisation dans la base de données de type IMREL sont comme nous l'avons vu de manière implicite dans le chapitre précédent, étroitement liés à la façon dont sont implémentés les fichiers maîtres dans IMAGE et les opérateurs relationnels réalisant l'accès aux données. Nous l'utiliserons malgré tout pour faciliter le rapprochement de notre travail et des recherches effectuées dans le même domaine (YAo79) (HAL 76)(BEL 81)(CHU 82)

Les méthodes d'optimisation peuvent être traitées de manière spécifique, en étudiant les fonctions d'optimisation des requêtes, selon un compromis entre l'encombrement des attributs dans chaque base et les conditions de requêtes.

Il est possible de formaliser le processus d'optimisation d'une requête en le découpant en différentes étapes donnant lieu à des traitements bien définis.

Les outils informatiques basés sur le traitement de requêtes suscitent grand nombre de travaux de recherche pour améliorer le temps de réponse et la place nécessaire au traitement d'une requête.

On modélise les mécanismes pour analyser la structure de requête, et on définit pour chaque requête les sous-requêtes, toutefois, la partition de la requête terminée, nous avons modélisé le remplacement de ces sous-requêtes en utilisant le mécanisme de calcul des facteurs de priorité permettant d'avoir un schéma de traitement optimal.

Dans ce chapitre nous présentons les différentes techniques utilisées pour optimiser les traitements de requêtes, ensuite nous allons établir un modèle mathématique pour évaluer les coûts minimaux de l'opérateur jointure "P-jointure". Ce modèle peut être une solution optimale au niveau de l'exécution globale des requêtes.

III - 2. QUELQUES METHODES D'OPTIMISATION

III - 2.1. Optimisation par les tableaux

Il s'agit d'une technique d'optimisation des requêtes formulées dans un langage algébrique présenté dans (AHO 79). On l'appelle technique des tableaux.

Un tableau T est une matrice dans laquelle les colonnes sont les attributs d'un nombre I ; attributs déposés dans un ordre précis. La première ligne est appelée sommaire du tableau. Les autres lignes expriment les n-uplets de I.

L'idée générale est que le teableau est une illustration ensembliste d'une requête algébrique constituée par trois opérations suivantes : la sélection, la projection et le joint-naturel.

Une démarche est proposée pour trouver un tableau minimal (ayant moins de lignes) équivalent à un tableau donné. Le terme d'optimisation est de chercher seulement à exécuter les sélections et les projections le plus tôt possible, afin de diminuer le nombre de lignes des n-uplets dans les relations à joindre.

III - 2.2. Optimisation dans le système INGRES

L'hypothèse est qu'une opération de jointure entre trois ou plusieurs relations est décomposée en une série de jointures.

Le processus de décomposition partage la requête en une série de sous-requêtes monovariables (chaque sous-requête est définie sur une seule relation).

Un algorithme d'optimisation en quatre étapes est construit (WON 76).

a) Réduction

Son rôle est de scinder les requêtes en sous-requêtes irréductibles et de les disposer dans un ordre séquentiel. On dit qu'une sous-requête est irréductible si l'élimination d'une de ses variables la fait éclater en sous-requêtes disjointes (sans variable commune).

Chaque composant irréductible correspond à une ou plusieurs lignes de la matrice d'incidence. On appelle matrice d'incidence réduite, la matrice dont les colonnes sont les variables de la requête, et les lignes ses composants irréductibles.

b) Réorganisation des sous-requêtes

A partir de la matrice d'incidence réduite, nous générons une succession de sous-requêtes monovariables et nous trasmettons chacune à l'étape ci-dessous:

c) Substitution des n-uplets

Une requête à (n) variables ayant des composants irréductibles (a), et composée de sous-requêtes d'une variable au plus (b), est remplacée par une famille de requêtes à (n-1) variables par substitution des valeurs de la nème variable.

d) Sélection de la variable

Nous estimons le coût relatif à chaque substitution de variable, nous choisissons la variable de coût minimum. La taille de la requête est le principal paramètre du coût de l'évaluation. Il est donc essentiel de l'estimer.

III - 2.3. Estimation de la taille d'une requête

WONG et YOUSSEFFI (WON 76) proposent, une méthode pour estimer la taille et également d'autres estimations de résultats intermédiaires.

DEMOLOMBE (DEM 80) décrit un modèle probabiliste qui permet d'estimer la cardinalité des requêtes exprimées en langage prédicatif. Ce modèle est adaptable aux langages algébriques et de type calcul relationnel.

RICHARD (RIC 80) présente aussi un modèle probabiliste pour estimer la taille des relations.

III - 2.4. Optimisation "FEEDBACK"

Une autre méthode d'optimisation pour un langage de type calcul relationnel est implicite dans le system TGR (SVE 80). Elle est appelée optimisation "FEEDBACK".

Des filtres (conditions) supplémentaires sont créés dynamiquement pendant l'évaluation de la requête. Ils ont pour objectif de fournir des informations lors du parcours d'une relation, pour éviter des recherches sur d'autres relations. L'effet traduit l'influence de la restriction d'une relation sur le nombre de n-uplets à chercher dans une autre relation.

III - 2.5. Optimisation dans le système RDBM

Dans RDBM la première opération effectuée (AUE 81) est le tri des relations de base, c'est la méthode la plus efficace dans la plupart des jointures entre deux relations d'une requête.

III - 2.6. Optimisation dans le système R

C'est un modèle combinatoire retenu dans le sous-système relationnel, pour optimiser l'exécution de l'opérateur Jointure. Plusieurs méthodes de jointure ont été définies. Après avoir expérimenté ces méthodes, nous arrivons à trois grandes classes de solutions.

a) Méthode des boucles imbriquées

Le principe est le suivant : il faut parcourir la relation externe, et pour chaque n-uplet qualifié, rechercher dans la relation interne les n-uplets qualifiés vérifiant la condition de jointure (VAL 82).

b) Méthode de tri des relations

Elle est aussi connue sous le nom méthode de jointure par fusion. Le principe est de trier les deux relations sur les attributs de jointures et de la balayer pour réaliser la fusion (BLA 77).

c) Coût d'accès

Le caractère de la requête conduit à étudier deux cas dans l'optimisation :

1 - la requête est monovariable
2 - la requête est multivariable.

Dans tous les cas, la stratégie optimale est celle qui minimise l'équation (E).

coût Estimé = nombre de pages chargées en mémoire principale

+ W * IRS

où W est un facteur d'ajustement entre une E/S (Entrée-sortie) et le CPU (unité contrôle)

IRS est l'interface Relationnel de stockage. Il est supporté par le sous-système de stockage (SRS).Le SRS fournit les opérateurs classiques d'accès aux enregistrements.

Pour choisir le plan d'accès optimal, l'optimiseur génère l'arbre de toutes les combinaisons possibles des chemins d'accès, en estimant le coût de chaque chemin.

III - 2.7. Conclusion

L'efficacité des techniques d'optimisation dépend du choix des méthodes optimales qui traitent la jointure, ou des méthodes basées sur les estimations de coût, l'objectif est de réduire le plus possible le transfert des données.

III - 3. METHODE DEVELOPPEE POUR OPTIMISER LE TRAITEMENT DE REQUETES

III - 3.1. Introduction et définition

Nous avons établi un modèle mathématique évaluant les coûts minimaux de l'opérateur jointure "ou-P-jointure". Quelques définitions et conditions sont mentionnées. Nous définissons un processus pour chaque requête contenant l'opérateur Jointure et nous déterminons l'ensemble de modèles qui résoudra l'opérateur Jointure "P-jointure" dans la requête.

Nous avons établi un procédé pour estimer la taille des éléments dans les relations d'exécution. Cette taille est déterminée par la chaîne des éléments (attributs) jointure entre deux relations ; c'est l'élément le plus important dans le calcul du coût. la forme de calcul obtenue est de type linéaire, qui peut être optimale dans certain cas que nous préciserons. Les requêtes de la multibase de données permettant de translater la demande, qui a été écrite dans un type de langage non-procédural. Ce langage dépend essentiellement des opérateurs d'algèbre relationnelle, qui facilitent l'accès à l'information.

Etant donné le schéma d'une base de données :

$D = (R1,R2,...,Rn)$

La requête peut être écrite en un nombre d'expressions algébriques alternatives. En particulier chaque requête peut s'écrire sous la forme suivante :

$Q = L.q(R1 \times R2 \times ..... \times Rn)$

où "L" est la liste des objets recherchés dans une relation (ou fichier)

"q" est la liste des items d'une relation et chaque Ri est une relation (modèle relationnel) ou un fichier (modèle hiérarchique).

Supposons que toute la requête soit de la forme $Q = (L,q)$. Dans le cas d'une multibase de données, la requête va s'exécuter au niveau de chaque base (par sous requêtes).

Le coût d'exécution de chaque requête dépend de trois opérateurs (SELECT, PROJECT, JOINTURE). Les deux premiers opérateurs seront exécutés localement dans la même base, leur coût d'exécution est négligeable. Notre objectif est de minimiser le temps d'exécution de Jointure.

Pour Q = (L,q), soit (R1,R2,..,Rn) l'ensemble du schéma de base, symbolisé par "q" et soit "X" l'ensemble des attributs apparus dans "q" pendant l'exécution de ces requêtes distribuées. On peut exécuter la projection au niveau de chaque relation Ri, la projection est égale à $(X \cup L) \cap Ri$. Cette opération est exécutée localement ; soit Q = (L,q) la requête posée sur une base de données multiples. La qualification "q" est une conjonction entre les différentes classes de relations de la forme (Ri.X = Rj.Y), où X et Y sont respectivement deux sous-ensembles des attributs Ri et Rj.

L'exécution de la jointure entre deux relations basées se fait grâce au transfert de l'une vers l'autre. Supposons que la technique de P-Jointure définit essentiellement la localisation du traitement. Dans la 1ère étape on exécute la projection et la sélection au niveau de chaque relation ; ensuite on transfère le résultat de l'un à l'autre afin d'exécuter la jointure appelée P-jointure. Ce type de localisation permet de réduire le nombre des informations transférées.

Avant de parler de l'exécution des requêtes distribuées, il faut fixer la stratégie de la démarche à appliquer.

III - 3.2. Le processus de la requête

La requête Q spécifiée par la qualification "q" sur les relations R1,...Rn, et par'L',L la liste des objets recherchés pouvant être décomposées sur plusieurs ensembles (p1,p2,...pl) pour arriver à une réponse.

Quelque soit le chemin employé il est défini par une stratégie d'exécution. Supposons S(Q) l'ensemble des stratégies qui permettent d'arriver à la réponse de la requête.

Notre but est d'arrivèr à la réponse avec un coût d'exécution minimal.

La fonction objectif est :

$$\underset{P \in S(Q)}{MIN} \quad F(P,D(O)) = \sum_{i=1}^{L-1} F_i(P_i, D(i))$$

ou $P_i = P_1, P_2, \ldots, P_L$

$D(i+1) = P_i\ (D(i))$

D(0) est le schéma de la base de données initiale.

Définition

Nous supposons qu'on travaille sur une multi-base de données S1,S2,...Sn. On définit chaque base par un noeud dans un réseau, considéré comme une seule relation.

On va calculer le coût d'exécution entre deux bases i et j tout en prenant en compte le nombre d'élément tranféré de i à j.

Considérons que le coût soit égal à (C). La fonction du coût de tranfert (V) entre les deux bases sera une fonction linéaire

$$C_{ij}(V) = C_{ij} * V$$

Pour optimiser la fonction du coût d'exécution, il sera nécessaire de diminuer le nombre V entre i et j ; la façon la plus simple est de localiser le traitement.

On définit pour chaque requête les paramètres suivants :

n = le nombre d'ensembles (i.e. relation, base-fichier...) à interroger par une requête

$di = |Ri|$ le nombre des attributs existant dans l'ensemble (i)

$X_{ij} = R_i \cap R_j$ les ensembles des attributs qui réalisent la jointure entre deux ensembles Ri et Rj.

ri le nombre des tuples dans Ri

T(A) la taille des items de l'attribut A dans l'ensemble Ri

$Si = ri*T(Xij)$ la taille de l'ensemble Ri pour réaliser la jointure , ou

$$T(X_{ij}) = \sum_{A \in X_{ij}} T(A)$$

Ensuite on définit plusieurs terminologies utilisées soit (Ri,ri) et(Rj,rj) deux relations et $X \subseteq Ri \cap RJ$.

Définition

La jointure de Ri et Rj à X, notée par $Ri \bowtie Rj = \{ t \mid t$ est un tuple sur $Ri \cup Rj$ tel que $t(Ri) \in ri \cap t(Rj) \in rj \}$.

La P-jointure (ou semi-jointure) de Ri et Rj à X, notée par $Ri \overset{X}{\ltimes} Rj$ est égale à $Ri \bowtie Rj$ (X) ou $Ri(X) \bowtie Rj = Ri \ltimes Rj$ où Rj(X) est la projection de Rj sur X par équivalence est égal à : $\{ ti \mid ti \in ri \cap (\exists tj \in rj,\ ti(X) = tj(X)) \}$

La jointure-naturelle de Ri et Rj notée par $Ri \bowtie Rj$ est une jointure de Ri et Rj à $Ri \cap Rj$.

La P-jointure naturelle de Ri et Rj notée par $Ri \ltimes Rj$ est une P-jointure de Ri et Rj à $Ri \cap Rj$.

Définition

On dit que la qualification "q" est "sous-naturelle" si et seulement si pour tous les ensembles on a :

$Ri\ Aik = Rj.Ajl$ ou $Aik = Ajl$

On dit que la qualification "q" est "naturelle" si et seulement si pour tous les ensembles Ri et Rj, et pour tout $Ak \in Ri \cap Rj$ on définit :

Ri.Ak = Rj.Ak est un sous ensemble de "q"

Définition

Soit le schéma de la base de données

$D = \{ R1, R2, \ldots, Rn \}$

On appelle une requête de jointure naturelle (ou requête de jointure sous naturelle), s'il existe une qualification naturelle (ou qualification sous naturelle).

Définition

1 - $< \bowtie_{ij}$ (ou Ri $< \bowtie$ Rj) est un opérateur de jointure naturelle distribué qui envoie Rj à Ri pour réaliser la jointure naturelle de Ri et Rj et donne un nouvel ensemble Ri.

2 - $< \ltimes_{ij}$ (ou Ri $< \ltimes$ Rj) est un opérateur de P-jointure naturelle directe, qui projette x = Ri $\cap$ Rj sur Rj et envoie le résultat à Ri pour réaliser la jointure à Ri et à l'ensemble R'i ou R'i = Ri $\bowtie$ Rj

On note que :

$\bowtie >ij$ = Ri $\bowtie >$ Rj et
$\ltimes >ij$ = Ri $\ltimes >$ Rj

sont définis de la même façon, on peut utiliser l'opérateur P-jointure sans aucun changement dans le schéma de base.

Définition

Le programme de jointure (ou P-jointure) p = p1,p2....pl est une séquence de l'opérateur jointure naturelle distribuée (ou l'opérateur P-jointure naturelle distribuée).

La qualification "q" de jointure naturelle (ou noeud final R) peut envoyer tous les ensembles Ri,($i \neq 1$) à R1 et on exécute :

R1 $\bowtie$ R2 $\bowtie$ .....$\bowtie$ Rn ou noeud R1 donc R2 $\bowtie >$ R1, R3 $\bowtie >$ R1,..., Rn $\bowtie >$ R1

Dans ce cas leurs permutations sont des programmesjointure (ou P-jointure) pour cette qualification "q".

III - 3.3. Graphe de requête

On va définir le graphe de la qualification sur le schéma de base : $D = \left[Ri\right]_{i=1}^{n}$ de telle façon qu'il soit un graphe <Vq,Aq,Bq> où Vq est l'ensemble des noeuds égal à D, Aq est l'ensemble de la P-jointure qui réalise

$$\{ aij = (Ri;Rj) \in Aq \mid \text{si } Ri \cap Rj \neq \emptyset \text{ et } Ri \not\subseteq Rj \}$$

notée i-----→-----j "avec un seule flèche" Bq = Vq x Aq = {bij | ∀i ≠ j} est l'ensemble de la jointure.

Ce type est noté par i---→→------j "avec deux flèches", si $Ri \cap Rj \neq \emptyset$ il n'existe pas de jointure (ou P-jointure) entre Ri et Rj, par contre s'il n'y a pas de jointure (ou P-jointure) et $Ri \subseteq Rj \Longrightarrow Ri = Ri \cap Rj$, alors la P-jointure de $Ri \neq Rj$, $Ri \rtimes Rj$ est le même que la jointure de Ri à Rj.

Exemple

Soit : R1 = (A1,A2,A3,A4)
R2 = (A2,A3,A5,A6)
R3 = (A5,A7,A8)

Le graphe de qualification "q" de jointure naturelle est (figure 17) :

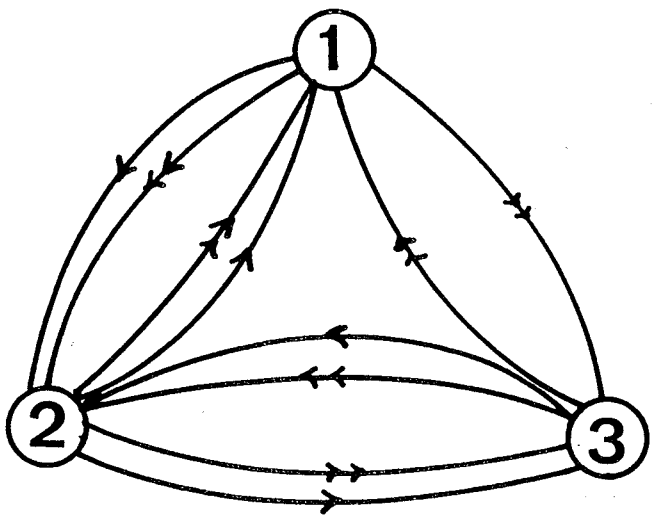


Fig.17 Graphe de qualification d'exemple

* On suppose toujours que 1 est le noeud final sans perte de généralité.

Définition

le programme P de jointure (P-jointure) est applicable pour la qualification "q" de jointure naturelle ; on montre d'après l'exécution de "p" que le noeud final va avoir une nouvelle relation "relation dérivé"

$R'1 = R1 \bowtie R2 \bowtie .... \bowtie Rn$

Lemme 1 :

Pour tout chemin d'accès (Bq) de R_{k_0} à R_{k_1} ; ou Bq = $bk_0,k_1.bk_1,k_2...bk_{L-1},k_L$; on peut construire le chemin :

Rko $\bowtie$ Rk1 $\bowtie$ ..... $\bowtie$ Rkl au noeud Rkl

Démonstration

Par récurrence : si I = 1 alors le chemin est $b_{k_0k_1}$. Après cette opération on va avoir :

$R'k_1$ <------- $Rk_0 \bowtie Rk_1$

Par induction pour l-1 on a la relation :

$R'k_{L-1}$ <------- $Rk_0 \bowtie Rk_1 \bowtie ..... \bowtie Rk_{L-1}$

où $R'k_{L-1} = R'k_{L-2} \bowtie Rk_{L-1}$

$R'k_{L-1} = Rk_0 \bowtie Rk_1 \bowtie .... \bowtie Rk_{L-1}$

Par induction, on montre que :

$R'k_L = R'k_{L-1} \bowtie Rk_L = Rk \bowtie Rk \bowtie ... \bowtie Rk_L$

Lemme 2 :

Pour chaque graphe de noeud final 1 de côté Bq, on va former
R1 ⋈ R2 ⋈ .... ⋈ Rn au noeud 1

Démonstration du lemme 1

Chaque chemin de Rk à R1 bva construire une nouvelle relation ou noeud 1, pour joindre toutes les relations dans ce chemin. En traitant par récurrence chemin après chemin vers le noeud 1, on va avoir à la fin la relation
R1 ⋈ R2 ⋈ ..... ⋈ Rn au noeud 1.

Théorème 1 :

Soit Q = (1,q) une requête de jointure naturelle et L = R1 U... URn
Soit P un programme de jointure (ou P-jointure) passant par le chemin de qualification "q".

On dit que "P" est correct si et seulement si il existe un sous-ensemble {bij} dans P, qui constitue un chemin inverse du noeud final.

Démonstration : condition nécessaire

La P-jointure diminue la taille des éléments dans la relation localisée, sans changement de chemin de relation. Après l'exécution de la Jointure par récurrence ver le noeud R1, on va retrouver la relation :

R1 ⋈ .... ⋈ Rn au noeud R1 et chaque

autre jointure ne change par la relation. Alors "P" est correct.

Condition suffisante

De la définition P-jointure "aij" a P, Ri $\cap$ Rj = $\emptyset$ et Ri $\subseteq$ Rj, qui transfère seulement une partie de la relation Ri à Rj. S'il n'existe pas de

sous-ensemble (bij) dans "P", on exécute la jointure qui transfère la totalité de Ri à Rj. Dans ce cas on perd la récurrence du chemin orienté vers R1, le résultat dans ce cas manque de lien entre les deux étapes et quelques informations peuvent s'égarer.

Du théorème 1 on définit les ensembles corrects de programmes "p" qui réalisent programmes de Jointure ou P-jointure.

Soit P l'ensemble des programmes et chaque exécution d'une requête distribuée va prendre p $\in$ P qui minimise le coût d'exécution.

Le changement d'ordre d'exécution des opérateurs dans le programme P va changer le coût total.

III - 3.4. Estimation de la taille des relations dérivées

Pour comparer le coût des différentes stratégies qui exécutent la requête, il sera nécessaire d'avoir une méthode pour estimer la taille de la relation après une seule opération. Aussi la technique d'estimation doit être solide car si deux suites des opérations sont identiques, les tailles estimées pour ces résultats selon les deux suites des opérations doivent être identiques.

Nous avons utilisé le facteur de réduction de P-jointure (ou jointure) pour les Ri et Rj ; noté αij, βij respectivement pour chaque couple des relations Ri et Rj.

αij ($0 \leqslant \alpha ij \leqslant 1$): est le pourcentage de la relation Rj qui va être éliminée après l'application de P-jointure (Ri $\rtimes$ Rj) à l'étape t, si les nombres des tuples dans Rj est rj(t-1) et si la valeur de P-jointure réduit de Ri à Rj est $\alpha ij(t-1)$, donc les nombres des tuples à Rj après l'exécution de P-jointure Ri $\rtimes$ > sera réduit à :

$$rj(t) = rj(t-1) * (1 - \alpha ij(t-1))$$

notons que la réduction de P-jointure de Ri à Rj n'est pas la même que Rj à Ri, $\alpha ii(t) = 0 \ \forall t$.

βij ($0 \leqslant \beta ij \leqslant 1$) : est le pourcentage de laréduction de la relation Rj après l'application de Jointure (Ri ⋈ Rj) à l'étape t, le nombre des tuples Rj après l'exécution de Jointure Ri ⋈ Rj sera égal à :

$$rj(t) = ri(t-1) * rj(t-1) * (1 - \alpha ij(t-1)) * (1- \alpha ij(t-1)) * (1-\beta ij(t-1))$$

l'effet de jointure Ri ⋈ > Rj est identique à l'exécution de P-jointure Ri ⋊ > Rj et Rj ⋊ > Ri.

La jointure de relation donne le même nombre de tuples pour la relation dérivée. Ce qui veut dire que la jointure est commutative, une propriété bien connue.

Ainsi on a : $\beta ij(t) = \beta ji(t)$

Aussi $\beta ii(t) = 0$, pour tout (t) dans notre démarche on peut définir αij, βij par une simple mesure statistique.

Si le nombre de tuple d'une relation est changée après une seule opération, le pourcentage de réduction de cette relation avec les autres relations va changer aussi.

Nous allons définir comment la réductibilité change après une seule opération.

Supposons que la base (Pt) avant l'opération est :

$$D = \{R1(t-1)....Rn(t-1)\}$$

donc le nombre des tuples pour une relation Ri(t-1) est ri(t-1), et $\beta ij(t-1)$, $\alpha ij(t-1)$ sont les nombres Jointure, P-jointure respectivement.

Si l'oparation Pt à l'étape t est αij(Ri ⋊ > Rj) seul le nombre des tuples dans la relation Rj(t) sera changée et sera égal à

$$rj(t-1) *(1- \alpha ij(t-1))$$

Et les autres relations vont garder les mêmes nombres des tuples.

Etant donné que l'opérateur P-jointure (αij) va réduire le nombre de tuples à Rj, l'opérateur va réduire toutes les autres relations Rk sur le chemin h -----><------ k, le $\alpha hk(t)$ de Rk(t) avec tous les autres Rk(t).Va prendre les valeurs suivantes :

$$\alpha_{hk}(t) = \begin{cases} 0 & \text{pour } h=i,\ k=i \\ \alpha hk(t-1) & h=i,\ k=j \\ \alpha hk(t-1) + \alpha ik(t-1) - \alpha hk(t-1)\,\alpha ik(t-1) & h=i,\ k\neq i,j \\ \alpha hk(t-1) + \alpha hj(t-1) - \alpha hk(t-1)\,\alpha hj(t-1) & h\neq ij,\ k=j \\ \alpha hk(t-1) & \text{autres cas} \end{cases}$$

L'opérateur P-jointure n'affecte pas la jointure de réduction. Ainsi toutes les jointures de réduction dans cette étape restant semblables à celles de la dernière étape.

$\beta ij(t) = \beta ij(t-1)$ pour toute valeur de k. L'opérateur Pt à l'étape t est $\beta ij(Ri \bowtie > Rj)$, donc la base au noeud Rj(t), se modifiera à :

Ri(t-1) U Rj(t-1) et
Ri(t-1) $\bowtie$ Rj(t-1)

est le nombre des tuples Rj(t), rj(t) est égal à :

$$rj(t-1) * ri(t-1) * (1-\alpha ij(t-1)) * (1-\alpha ji(t-1)) * (1-\beta ij(t-1))$$

et les autres relations restent sans changement, parce que ce sera une opération P-jointure qui affecte la jointure. En résumé, le P-jointure de réduction de Rh(t) à Rk(t) à les valeurs suivantes :

$$\alpha_{hk}(t) = \begin{cases} 0 & \text{si } h=i,\ k=i \\ \alpha hk(t-1) & h=i,\ k=j \\ \alpha hk(t-1) + \alpha ik(t-1) - (\alpha hk(t-1))*(\alpha ik(t-1)) & h=i,\ k\neq i,j \\ \alpha hk(t-1) + \alpha hj(t-1) - (\alpha hk(t-1))*(\alpha hj(t-1)) & h\neq i,j,\ k=j \\ \alpha hk(t-1) & \text{autres cas} \end{cases}$$

La jointure de réduction de Rh(t) à Rk(t) est :

$\beta\, hk(t) = \beta\, hk(t-1) + \beta ik(t-1) - (\beta hk(t-1)) * (\beta ik(t-1)) \quad \forall\, k \neq i,j$

La compatibilité de système

Le système est dit compatible, s'il donne les mêmes estimations quand les deux programmes donnent les mêmes résultats.

Nous définissons alors le systèmes compatible que nous allons utiliser pour estimer les tailles des relations dévelopées, dont les paramètres sont ri, αij, βij.

Théorème

le système des paramètres $(ri(t), \alpha ij(t), \beta ij(t))$ que nous avons défini ci-dessus est compatible.

Démonstration

L'exécution de l'opérateur jointure (ouP-jointure) sera modifiée seulement par la taille des relations dérivées et pas par l'ordre de la dérivation.

Exemple

Supposons que nous avons trois relations R1,R2,R3 ou $R_i \cap R_j = \emptyset$ et $Ri \subseteq R_j$ $\forall ij$ et supposons que αij (0) βij sont données (ou calcul).

Alors le graphe de processus entre les relations va être (figure 18)

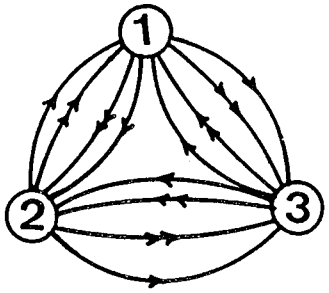


Fig.18 Graphe de processus entre les relations

étant donné P1 = P31,P21 et P2 = α_{12} α_{13} β_{23} β_{31}

Les deux programmes vont produire les mêmes résultats
R1(0) ⋈ R2(0) ⋈ R3(0) dans le noeud 1.

Par les règles d'estimation de la taille de la relation dérivée, les tailles estimées de R1(0) ⋈ R2(0) ⋈ R3(0) dérivées par ces deux programmes va être les mêmes. C'est à dire

$$\prod_{i=1}^{3} ri(0) * \prod_{\substack{i,j=1 \\ i<j}}^{3} (1-\alpha ij(0)) * \prod_{\substack{i,j=1 \\ i<j}}^{3} (1-\beta ij(0))$$

Formulation du modèle

Pour dériver la formulation mathématique du modèle d'optimisation de requête distribuée.

Nous avons besoin de connaître la fonction de coût de chaque opérateur, nous avons supposé que la fonction de coût est linéaire, donc nous pouvons écrire la fonctions de coût de ce type d'opération à l'étape "t".

La projection de Ri sur Ri ∩ Rj peut donner un nombre de tuples plus petit que celui des tuples dans Ri. Nous n'allons pas tenir compte du coût de cette opération.

Nous supposons que le nombre des tuples après la projection en Ri sur Ri ∩ Rj est égal à :

$$W_{Ri} = \text{Min} \{ri(t), \prod_{A \in Xi} |dom(A)|\}$$

Le coût αij va être

$$\text{coût } (\alpha ij) = Cij * (W_{Ri} * \sum_{A \in Xij} W(A))$$

Le coût de βij va être

$$\text{coût } (\beta ij) = Cij * (ri(t) * \sum_{A \in Ri} W(A))$$

Nous développons la formulation du modèle de requête distribué comme suit :

1- Un schéma des bases de données distribuées
2- La largeur W(A) de chaque attribut A à U (D)
3- le nombre des tuples ri(0) de chaque relation Ri
4- La réductibilité de P-jointure $\alpha ij(0)$ de chaque paire de relations Ri, Rj avec $\alpha ii(0) = 0$ et $\alpha ij(0) \neq \alpha ji(0)$
5- La réductibilité de Jointure $\beta ij(0)$ de chaque paire de relations RiRj avec $\beta ii(0) = 0$ et $\beta ij(0) = \beta ji(0)$.

III - 4. CONCLUSION

La taille des résultats doit être limitée (c'est très important pour réaliser l'opérateur Jointure).

Notre objectif est de proposer une démarche pour effectuer l'optimisation des requêtes dans un système de bases de données multiples.

Dans une première partie on a énuméré l'ensemble des tous les critères d'optimisation. dans une deuxième partie nous avons établi une méthode mathématique pour optimiser la requête distribuée sur plusieurs bases. Cette méthode évaluant les coûts minimaux pour exécuter l'opérateur Jointure (ou P-jointure). Quelques définitions et conditions sont mentionnées, une procédure pour estimer la taille des éléments dans les relations dérivées et pour chaque requête nous déterminons le processus et le graphe de la requête.

BIBLIOGRAPHIE

(AHO 79) A.V. AHO, Y. SAGIV, J.D. ULLMAN
"Efficient optimization of a class of relational expressions"
ACM Transaction on DBS, Vol.4, n°4, pp 435-454, Dec. 1979.

(AST 80) N.M. ASTRAHAM, M. SEMKOLNICK, W. KIM
"Performance of the system R acces path selection mechanism"
Congrès IFIP 1980, North Holland publi., pp 487-491, 1980

(AVE 81) H. AVER et al
"RDBM : a relational data base machine"
INFO system, Vol.6, n°2, pp 91-100, 1981.

(BEL 81) Z. BELLHASENE
"Méthodes d'évaluation de question dans les systèmes Monoprocesseurs"
Rapport Pojet SABRE, INRIA France, 1981

(BLA 77) M.W. BLASGEN, K.P. ESWARAN
"Storage and access in relational data base"
IMB Sytem Journal , n°4, pp 363-377, 1977

(CHU 82) W. CHU, R. HURLEY
"Optimal processing ofr distributed data base systems"
IEEE Transactions on Computers, vol. C31, n°9, Sept. 1982

(DEM 80) R. DEMOLOMBE
"Estimation of the number of tuples satisfying a query expressed in predicate calculus language"
Very Large Data base Conference, Montréal, Canada, pp 55-63, 1980

(HAL 76) P.A.V. HALL
"Optimization of simple expression in a relational data base systems"
IBM J. of Res. Develop., vol.20, n°3, pp 244-257, Mai 1976

(RIC 80) P. RICHARD
"On the evolution of the size of the answer of a relational query"
Rapport de recherche, INRIA France, Déc. 1980

(SEV 80) SVEND, ERIK, CLAUSEN
"Optimizing the evaluation of calcul expression in a relational data base system"
INFO. Systems, vol.5, pp 41-54, 1980

(VAL 82) P. VALDURIEZ, G. GARDARIN
"Multiprocessor JOIN algorithms of relations"
Rapport n° MBO-I-010, INRIA, 1982

(WON 76) E. WONE, K. YOUSSEFI
"Decomposition a strategy for query processing"
ACM Transation on DBS, vol.1, n°3, pp 223-242, Sept. 1976

(YA 79) S.B. YAO
"Optimization of query evaluation algorithms"
ACM Transaction on DBS, vol.4, n°2, pp 133-155, Juin 1979

CHAPITRE IV

LE COMPILATEUR

IV - 1. INTRODUCTION

Les techniques de compilation sont actuellement bien connues, et de nombreuses méthodes efficaces permettent de faire face à la totalité des langages de programmation existants.

La compilation sur les bases de données ne relève pas d'une théorie particulière, mais les techniques mises en oeuvre doivent être assez spécifiques et sélectionnées surtout sur des critères d'économie de réalisation.

De nombreuses applications sur les bases de données nécessitent la définition d'un langage d'interrogation et la possibilité de reconnaître les phrases de ce langage (analyse syntaxique).

Le travail qui a été réalisé a pour but de faciliter la tâche aux informaticiens qui désirent développer de telles applications. Le produit qui a été écrit est du type "Compilateur de compilateur". Son rôle consiste essentiellement à libérer l'utilisateur de l'écriture de l'analyseur syntaxique : en lui fournissant une description syntaxique (grammaire hors-contexte) de son langage.L'utilisateur obtient alors le "squelette" d'un compilateur dirigé par la syntaxe ; des points de génération (i.e. nom de procédure d'analyse sémantique) placés dans la grammaire lui permettent d'indiquer à quels endroits, au cours de l'analyse syntaxique, doivent être activées les procédures d'analyse sémantique.

Ces procédures, écrites par l'utilisateur, effectuent la vérification des règles syntaxiques non spécifiées dans la grammaire (par exemple : double déclaration, cohérence des types dans les expressions....) et produisent le code objet (i.e. traduction du texte source).

On peut décrire l'aspect visualisation et description d'un compilateur par le schéma suivant (figure 19)

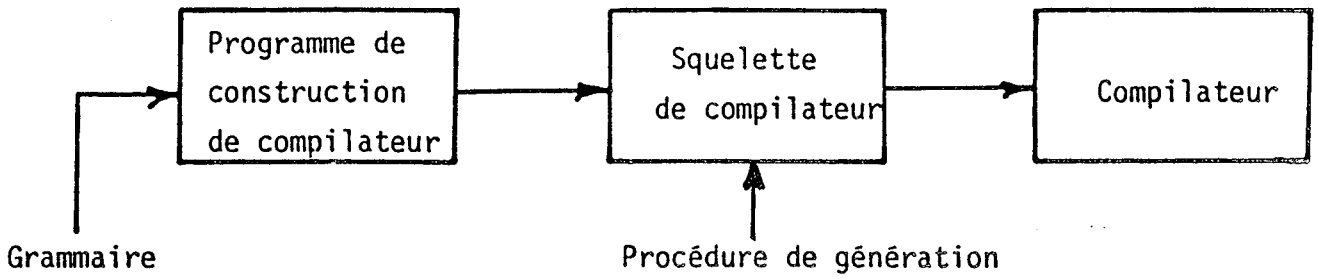


Fig.19 L'aspect description d'un compilateur

Les décompositions les plus courantes font intervenir :

1- l'analyse lexicographiques

2- le traitement des mécanismes déclaratifs : construction d'une table de symboles complétée par extraction des déclarations de la requête d'entrée. Tous les attributs des identificateurs sont disponibles à la fin de cette passe sauf la valeur des passages.

3- l'analyse syntaxique faite en général par un processus LR(1). On obtient comme résultat une notation post-fixée de la forme d'entrée, d'où sont enlevées les sous expressions constantes ; on construit en même temps des relations locales entre les différents objets qui donneront en fin de passe les conditions d'application des mécanismes de minimisation globale.

4- les minimisations globales : ce sont toutes les minimisations que l'on ne peut effectuer que par une connaissance du contexte où se situe la séquence à minimiser : on évite l'évaluation multiple de sous expressions communes, l'évaluation répétée de sous expressions localement constantes dans un contexte donné (qui regroupe la méthode dite de clacul du coût, et les schémas de processus de la requête telle que celle au (III)) et dans (GAV 80)(PAR 79)(DES 80)(LOV 81). Ces contextes sont définis uniquement par

des relations sémantiques entre objets et ne sont pas nécessairement reflétés par la syntaxe du langage à traduire.

5- La génération du code objet : on est ramené à cette étape à faire le travail d'un compilateur de requête, réalisé selon le cas en une ou deux passes.

Par contre, l'écriture des autres composantes du compilateur est actuellement à l'état de recherche, le plus souvent portant sur la réalisation d'un outil d'interrogation des données stockées dans une base. Il s'agit d'un système de fonctionnement d'un compilateur de compilateur, qui utilise l'analyse syntaxique pour diriger l'exécution d'un programme ; cette analyse n'est pas nouvelle, on la trouve dans (AHO 79)(CUN 80) (HOP 70)(WEL 82)(GAL 77).

Ces produits sont essentiellement des langages permettant d'insérer des appels de sous-programmes d'un langage existant (FORTRAN) dans une description grammatical (sous forme BNF en général) de données analysées. A partir de cette description ainsi enrichie d'actions sémantiques, le compilateur de compilateur construit un analyseur syntaxique qui appellera les sous programmes indiqués lorsque la règle qui les précède dans la description aura été reconnue.

le but de ce chapitre n'est pas d'énumérer les avantages et les inconvénients éventuels de ces types de compilateur ; mais en gardant les définitions du compilateur d'étudier l'aspect visualisation et description d'un compilateur :

- en définissant des grammaires et des symboles
- en analysant des données selon ces modèles
- en effectuant des actions sémantiques lors de cette analyse.

IV - 2. L'ASPECT VISUALISATION ET DESCRIPTION D'UN COMPILATEUR

Le compilateur est un ensemble de programmes dont le but est de traduire un programme source en un programme équivalent écrit en code intermédiaire. Ce code intermédiaire contient deux types d'informations :

- les ordres d'exécution
- les indications pour l'éditeur de liens entre les différents types de programmes d'exécution, en utilisant le schéma défini par l'arborescence.

Le compilateur possède deux fonctions principales.

IV - 2.1. Partie lexicale

L'analyseur syntaxique est un interpréteur des automates produits. Il fait appel aux procédures "sémantiques" au cours de son exécution. On remarque que les transitions se font en général sur la reconnaissance d'un mot de requête. L'analyseur syntaxique ne connaîtra en fait les terminaux que sous la forme d'un code. Un analyseur lexical (figure 20) sera chargé d'isoler les terminaux dans la requête passée et de délivrer leur code à la partie syntaxique.

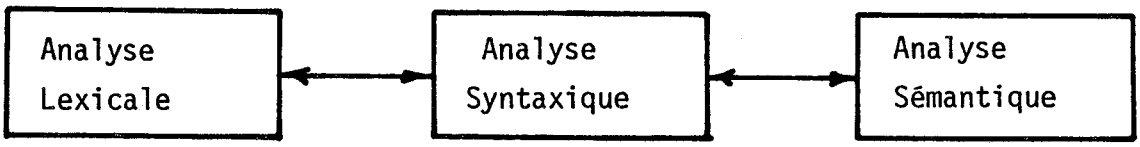


Fig. 20 Partie de l'analyseur lexical

IV - 2.1.1. Analyse lexicographique

Elle a pour but d'isoler et d'identifier les éléments (mots) qui composent la requête. Dans notre cas on va construire notre analyseur pour connaître les blancs séparateurs dans une requête, et transférer la requête dans une table réservée pour chaque mot. Il est possible d'avoir 16 mots au maximum dans une requête (mais ceci peut être modifié).

IV - 2.1.2. Analyse syntaxique

L'utilisation de descriptions syntaxiques pour les langages de requête est un facteur important dans la construction de compilateur de compilateur. Ceci ne veut pas dire que le réalisateur d'un compilateur ait à effectuer de gros efforts sur la partie syntaxique, mais plutôt que la structure de contrôle de compilateur est imposée par la structure syntaxique choisie.

Le côté automatique de la mise en oeuvre des mécanismes syntaxiques est important car il permet l'utilisation de grammaires, non seulement pour l'écriture de compilateurs, mais aussi pour la description de données dans des applications très variées (CUN 80).

En fait, cette technique est très utile dès que les données ont des structures complexes. Ceci est notre cas car nous exploitons diverses bases de données.

Pour décrire un analyseur syntaxique, il est important de choisir une grammaire qui convienne au traitement que la requête va subir. On considère une classe de grammaires hors contexte (les grammaires LL(1)) qui associent une analyse descendante déterministe à un automate d'état fini.

Le mode d'écriture des la grammaire est défini en (CUN 80)(KOH 70) (COV 71) à l'aide de la grammaire BNF.

Syntaxe de langage de requête

<Requête> ::= <Opération><Fin Requête>

<Opération> :: = <opération 1> < suite type 1>|
<opération 2> <suite type 2> |
<opération 3>

<opération 1> ::= SELECT | MODIFIER | SUPPRIMER

<opération 2> ::= LISTE-CLE | LISTER-FICHIER | OUVRIR-BASE | FERMER-BASE |
OUVRIR-FICHIER | FERMER_FICHIER

<opération 3> ::= HELP

<suite type 1> ::= {<blanc>}* <condition 1><suite de condition>

<condition 1> ::= <nom colon> [|{<nom colon> {<blanc>}*}*]

<nom colon> ::= <identif>

<identif> ::= {<carc>}* [|<car spec> {<carac> }*]

<carc> ::= A|B|C|.....|Z

<car spec> ::= -|+|(|)|*|/|$|:|%|&|<|>|=|?

<fin requêt> ::= {<blanc>}* <car fin>

<car fin> ::= ;

<blanc> ::= ∅

<suite type 2> ::= {<blanc> }* <nom colo> <fin requête>

<suite de condition> ::= { <blanc> }*[TEL QUE]{<blanc>}* <condition>
<fin requête>

<condition> ::= <nom colon> {<blanc>}* <valeur> [|<condition>]

<valeur> ::= <predicat> <fonction 1> <predicat> {<blanc>}*

<fonction 1> ::= <fonction 2> <car> |
<op-arthm> <car> |
<CAR> <CARC>

<fonction 2> ::= MIN|MAX|MOY|SOM

<op-arthm> ::= LT|GT|EQ|NE|LE|GE

<CAR> ::= <nombre>|<nombre><carac>|
<carc><nombre>

<nombre> ::= 0|1|2|.....|9

<prédicat> ::= "

Symbole du langage

1 - les opérations :

```
MOS (1) = 'SELECT          '
MOS (2) = 'SUPPRIMER       '
MOS (3) = 'MODIFIER        '
MOS (4) = 'LISTER-FICHIER  '
MOS (5) = 'LISTER-COLON    '
MOS (6) = 'OUVRIR BASE     '
MOS (7) = 'FERMER BASE     '
MOS (8) = 'OUVRIR FICHIER  '
MOS (9) = 'FERMER FICHIER  '
MOS (10)= 'HELP            '
MOS (11)= 'TELQUE          '
MOS (12)= 'FIN             '
MOS (13)= ' "              '
MOS (14)= '  ;             '
```

Condition d'écriture

On remarque qu'aucun symbole ne contient de blanc. Dans le cas de plusieurs blancs entre deux symboles, seul le 1er d'entre eux sera pris en considération comme séparateur.

{ }* pour signifié un nombre non nul qui peut être répété plusieurs fois

[ ] pour une factorisation explicite
A⟶BC|BD
A⟶B [C|D]

< > entourent chaque notion non terminale

::= est

| ou bien

$\not{b}$ symbole blanc

Liste des erreurs

L'imprimante nous donne différents messages d'erreur :

ERREUR (1) Déclaration de mot trop long (16 car)
ERREUR (2) requête très longue, la taille maxi = 16 mots
ERREUR (3) la base n'existe pas
ERREUR (4) le fichier n'existe pas
ERREUR (5) type de requête traitant trois objets
ERREUR (6) type de requête traitant un seul objet
ERREUR (7) 1er mot dans requête faux ou opération n'existe pas
ERREUR (8) mot non trouvé dans la base := "n° base"
ERREUR (9) requête sans condition
ERREUR (10) requête sans points d'entrée
ERREUR (11) dans la routine <off set> pour le fichier
ERREUR (12) traitement base de données : n° base

IV - 2.1.3. Analyse sémantique

Comme on l'a vu aux paragraphes précédents, le travail d'un compilateur se décompose en trois grandes parties :

a) Reconnaissance des éléments (mot) du langage de requête qui sont représentés sous la forme d'une chaîne de caractères. Le compilateur distingue les éléments (mots) du langage ; c'et le travail de l'analyseur lexicographique.

b) Reconnaissance des éléments de la grammaire, lorsqu'un élément du langage est distingué,il faut décider si celui-ci est utilisé correctement par le progammeur du programme source ; cette partie est réalisée par l'analyseur syntaxique.

c) Une fois les deux phases précédentes accomplies, un certain nombre d'actions doivent être entreprises, le déroulement de cette partie requiert un certain nombre de fonctions spécialisées désignées généralement sous les noms :

1 - Filtrage
2 - Procédures sémantiques

IV - 2.1.3.1 Filtrage

Dans notre travail nous avons déjà parlé de la possibilité d'interroger les données en utilisant les mots clés comme un point d'entrée dans la base.Pour augmenter l'efficacité de l'interrogation des données nous pouvons aussi améliorer les structures des requêtes et des algorithmes permettant de les utiliser, et le traitement de données résidant sur des supports dont l'accès est séquentiel.

On peut résumer les efforts pour améliorer ces deux points en utilisant des mécanismes de filtrage qui seront expliqués dans les trois paragraphes suivants.

Filtrage au niveau lexical

Nous avons rappelé que la recherche de mots clés dans une requête est réalisable grâce à la construction de l'automate de reconnaissance de mots clés cette réalisation matérielle d'opérateur capables d'exécuter un automate est très simple. Cette solution est effectuée en une seule passe par le balayage séquentiel des alternatives dans la table qui contient les mots clés, en commençant par le 1er mot des mots clés.

Si la requête ne contient pas de mots clés on applique ici la recherche avec le filtrage.

Dans l'exemple suivant on explique la possibilité d'une réalisation de ce type.

Soient les deux relations :

R1 (technologie, provenance, lot, tranch, n° relation)
R2 (motif, nom sous type 1, valeur sous type 1, n° relation)

on veut la liste de nom sous type 1 dont la tranche est 1 ou 2.

Dans un premier temps la requête va s'écrire sous forme SQL

```
SELECT      nom sous type 1
FROM        R2
WHERE       n° relation      SELECT      n° relation
                             FROM        R1
                             WHERE       tranche="1"
                                      OR tranche="2"
```

Dans un premier temps, l'automate qui reconnaîtra deux tranches est construit, et on explore R1 avec un filtre contenant cet automate. Chaque fois qu'il trouve un enregistrment avec l'une de ces tranches, le filtre envoie le champ n° relation vers le compilateur d'automates, qui travaille donc en pipeline avec le filtre (Figure 21). A la fin de l'exploration de R1, il suffit de balayer R2 avec le filtre utilisant l'automate que l'on vient de construire.

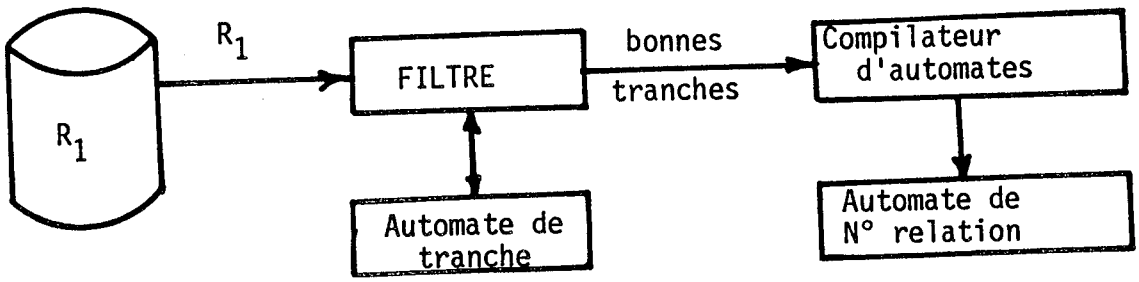


Fig.21 Filtrage à pipeline local

Au cours de ce second balayage, le filtre sélectionne les enregistrement (R2) dont le n° relation est reconnu par le 1er automate, il envoie le champ "nom sous type 1" de ces enregistrement vers la construction d'automate.

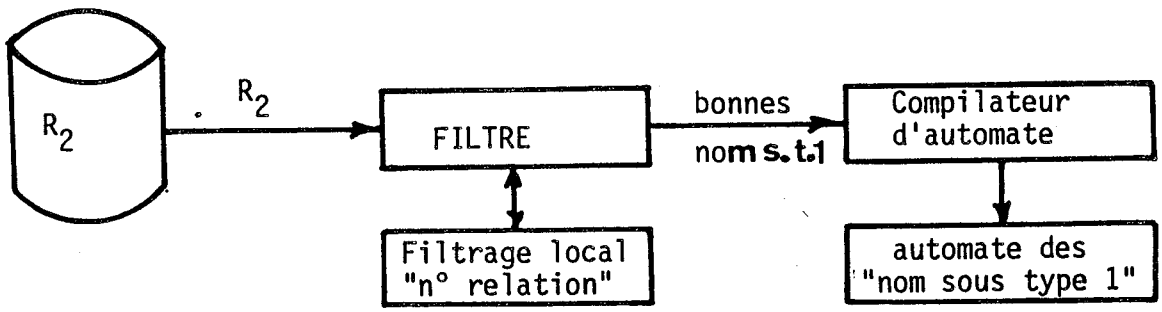


Fig.22 Filtrage à pipeline réparti

On dispose finalement de l'ensemble des noms de sous type 1 recherchés sous la forme de l'automate qui le reconnait.

En conclusion la requête a été exécutée dans un temps rapide égal au temps nécessaire pour lire les opérandes de cette relation (R2).

La base de cette technique est en effet de transformer une relation en exécutant l'opération algébrique P-JOINTURE (qui était définie au chapitre III).

Filtrage au niveau syntaxique

Cette partie concerne la prise en compte de la structure des données à traiter. Nous avons toujours considéré que la structure des données est composée de différentes parties que l'on peut désigner au moyen d'un langage de requête.

Par exemple, une structure d'une base de données IMREL sera constituée d'un ensemble de champs appelés domaines, chacun de ces champs ayant un nom et un type :

(IDRES : texte, méthode; texte, DATE : entir, moy: réel)

Il est possible de trouver un facteur minimal en respectant l'ordre hiérarchique dans la structure. On appelle facteur le nombre de types différents constituant un champ. Le facteur minimal est celui qui contiendra le moins de types différents parmi tous les facteurs considérés.

Exemple

On a deux champs, le premier contient (5) types différents et le deuxième (8), on choisira donc le premier.

Considérons par exemple la structure de base ayant la forme hiérarchique :

1 TECHNOLOGIE
  2 PROVENANCE
    3 LOT
      4 TRANCHE
    3 MOTIF
    3 IDRES
      4 DATE
      4 MOY

Si on cherche les tranches de Motif xx et provenance yy ont les moyennes > R.

Il va falloir :

- savoir si on est ou non dans une sous structure provenance de nom yy "filtre 1" ;

- mémoriser tout nom motif jusqu'à la fin de leur sous-structure ;

- savoir si on est ou non dans une sous structure motif de non xx "filtre 2" ;
- mémoriser toute valeur moyenne et tranche jusqu'à la fin de la liste motifs ;
- comparer la valeur moyenne avec R "filtre 3"

- garder finalement le nom tranche si la condition sur la valeur moyenne est satisfaite.

Nb. Si on a deux sous structures de même niveau on prendra la sous structure de facteur minimal. Le schéma de réalisation nous donne la (figure 23).

En effet, les besoins de l'analyse des structures de données confortent notre analogie avec les mécanismes de compilation d'une requête.

L'analyse des structures est simple à réaliser matériellement, elle joint de façon assez rapide deux bases :

- la structure des données analysées est totalement schématisée au moment de construire les bases, et programmée pour s'exécuter au moment de la compilation de la requête.

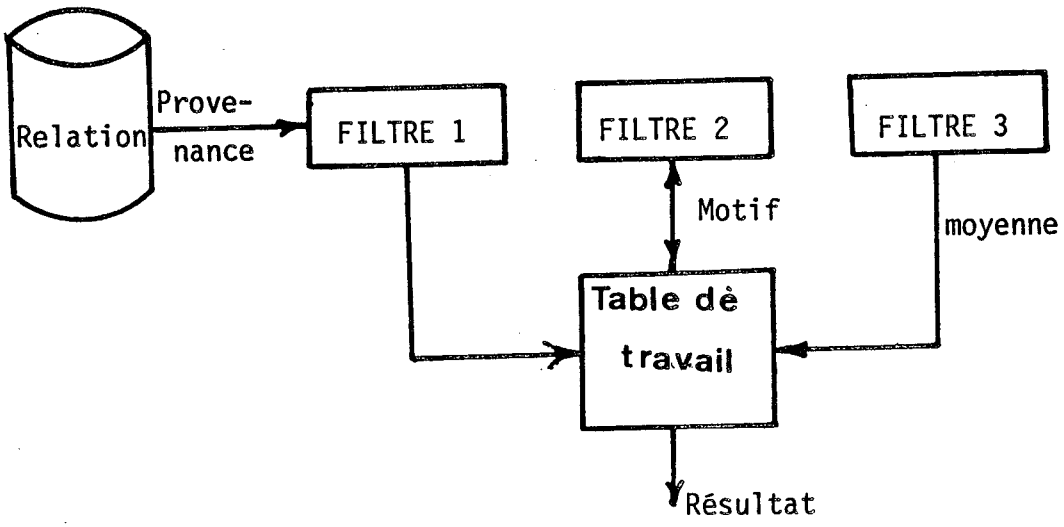


Fig.23 Filtrage au niveau syntaxique

Filtrage au niveau sémantique

Le but initial d'un filtre est de filtrer, c'est à dire de laisser passer ou non un enregistrement. Cela revient à considérer le filtrage comme l'application à chaque enregistrement E d'une fonction booléenne F. L'enregistrement est conservé si et seulement si F(E) vaut 1.

Ceci correspond à la notion habituelle d'opération de sélection sur une

base de données.... trouver toutes les tranches dont la moyenne est > 1.959 et date < 100283.

Une manière générale de considérer le filtrage est de dire que le résultat de F(E) est lui-même un enregistrement quelconque.

Ceci recouvre l'opération de projection de l'algèbre relationnelle, qui revient à supprimer certains champs de E, et qui peut être généralisée à la projection conditionnelle, comme dans l'exemple suivant :

F(E) ——————→ si champ Moyenne < 1.959 alors champ Min
F(E) ——————→ si champ Moyenne ⩾ 1.959 alors champ Max

On peut considérer les traitements à effectuer pendant le défilement des requêtes dans le filtre comme des actions sémantiques. Dans ces conditions une alternative très simple consiste à reporter l'essentiel du travail au niveau de la compilation de requête du critère de filtrage.

Soit "p" le nombre des mots clés dans chaque base, on construit un champ de vérité, on range la valeur de présence (1 ou 0) dans un bit de l'adresse. On attribue à chaque mot clé un numéro de variable de 0 à p-1.

La variable i vaut 1 si et seulement si le mot clé i est présent.

Quand on remplit dans le champ de vérité les valeurs des mots d'une sous-requête, le filtre envoie vers l'unité d'exécution la valeur de cette adresse pour définir un sous-programme qui traite la sous requête.

IV - 2.1.3.2. Procédures sémantiques

C'est un certain nombre d'actions qui doivent être effectuées au niveau de la traduction du langage de requête en code intermédiaire.

Le déroulement de cette partie requiert un certain nombre de fonction spécialisée, on peut regrouper les procédures sémantiques en trois groupes:

- Logique : ce type prend la décomposition logique de la requête (ou sous requête), soit la sous requête

SELECT Z TELQUE X = "a" ou Y = "b",

cette sous requête peut s'écrire :

s1 : select Z tel que x = a
s2 : select Z tel que y = b

si on réalise un des deux cas le résultat est vrai.

En respectant les définitions logiques nous arrivons à une amélioration au niveau de l'exécution d'une requête.

- Qualitative : Ce type prend en compte le calcul de la distance entre deux attributs, défini en (I-5.1.5). Ce calcul est utilisé pour obtenir le schéma de manipulation minimal.

- Quantitative : En respectant les résultats de (CH. III). On exécutera en priorité la sous requête qui a le coût minimal.

Dans notre travail nous avons réalisé un sous programme qui exécute les différents types de filtrage et les procédures sémantiques et dont le résultat est la sous requête qui va être exécutée en premier.

IV - 2.2. Génération de code

C'est la partie qui fournit le code translatable. Elle comprend les procédures de génération du code qui sont générées par les analyses syntaxiques et sémantiques.

D'autre part, le compilateur de requête utilise un certain nombre de variables (figure 24) que l'on appellera par la suite des variables de travail du compilateur.

Ces variables de travail sont de deux types :

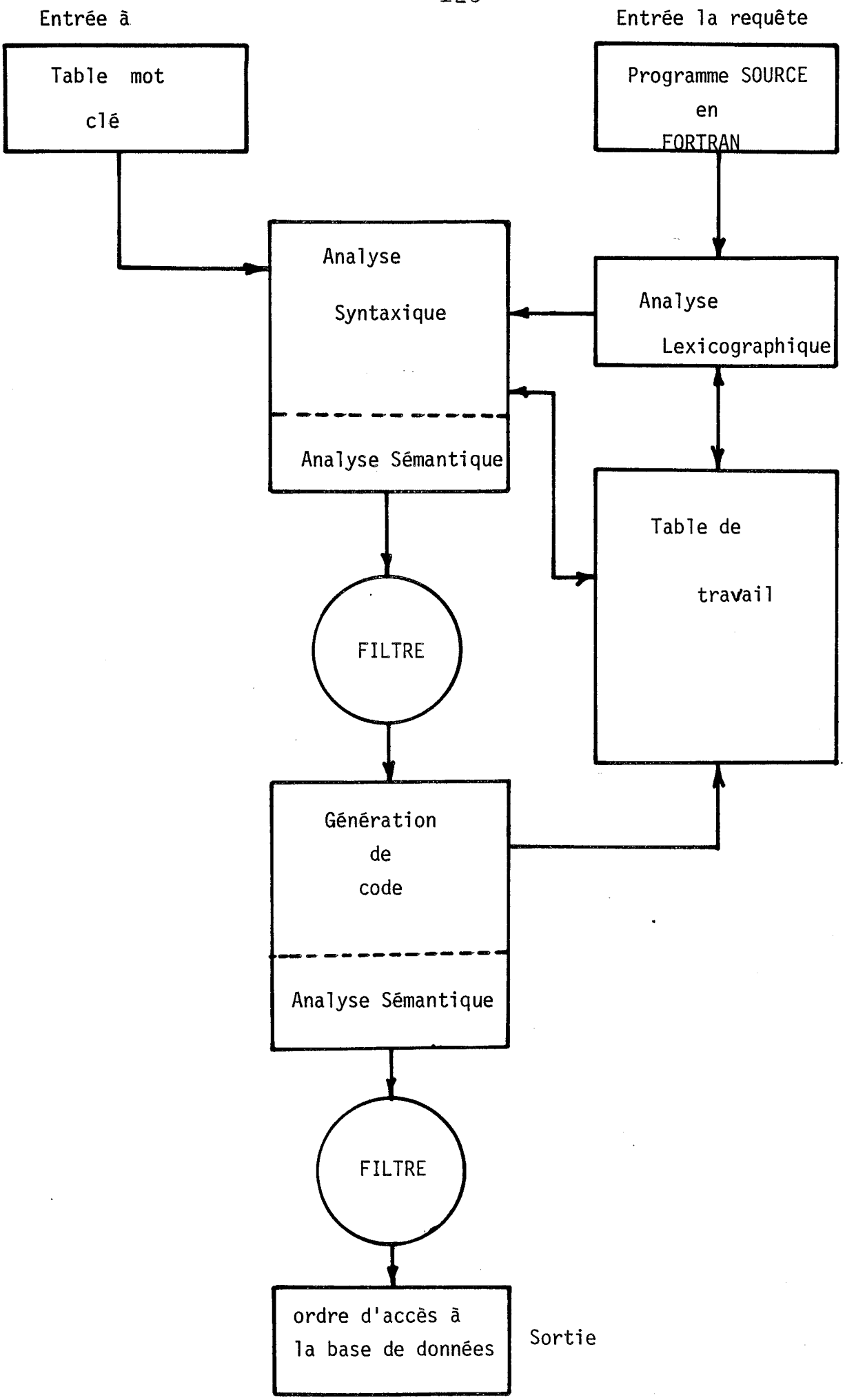


Fig.**24** Les différentes parties du compilateur

Les tables construites pendant la génération du langage de requête que le compilateur consulte comme une table intermédiaire (SAK 80)(AMI 74)

Les variables propres au compilateur, comme par exemple l'analyseur de filtre que l'on détaillera dans le paragraphe d'analyse sémantique.

De plus, le compilateur utilise des zones tampons, en sortie pour stocker l'ordre généré d'accès à la base de données (l'arborescence algébrique).

Dans le schéma suivant figurent :

- les principales parties du compilateur
- les liaisions entre les constituants.

Comme on vient de le voir, le compilateur utilise un certain nombre de variables de travail. Ces variables peuvent être regroupées en deux parties:

* celles fournies à l'entrée d'une requête (table de symbole, grammaire de syntaxe, règle de sémantique)
* les variables de travail du compilateur

    a) l'arborescence Algébrique
    b) la table de mots clés et des identificateurs

Arborescence de chaque opérateur de base on donne ci-dessous les arborescences d'interprétation des opérateurs de base défini dans le système MICROBE (FER 80).

a) Type des requêtes

SELECT, MODIFIER, SUPPRIMER, LSTER-C, OUVRIR FICHIER, LISTER-F, OUVRIR BASE, FERMER BASE, FERMER FICHIER.

b) Opérateurs algébriques

JOIN, SELECT, PROJECT, UNION, INTERSECTION, DIFFERENCE, INSERT, MODIFIE, ANTI-PROJECT,

c) Fonctions

MAX, MIN, SOM, MOY.

A - Types de requêtes

1 - Requêtes composées

Soit la requête :

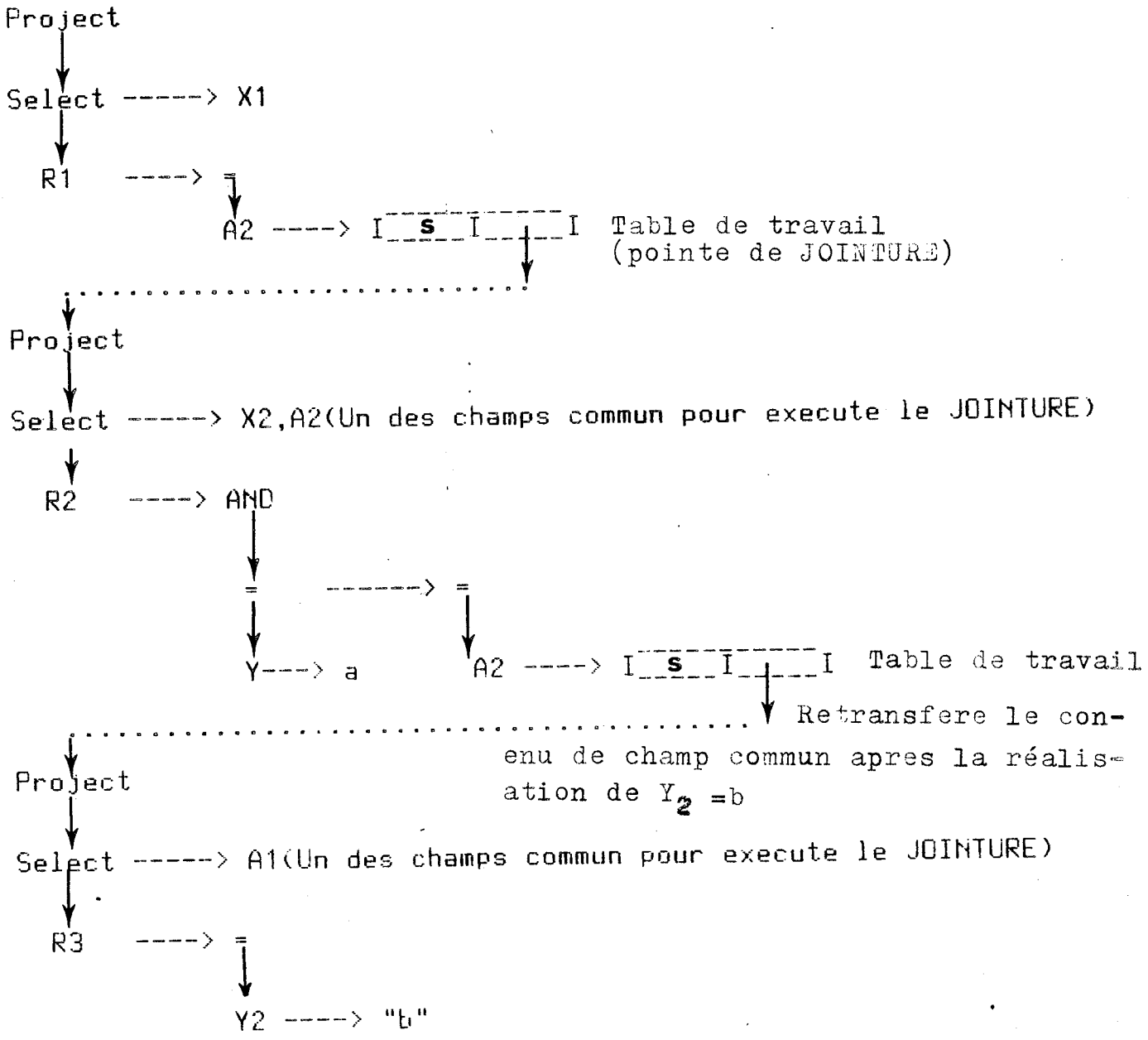
SELECT X1 X2 TELQUE Y1 = "a" et Y2 = "b" ;

d'après l'analyse syntaxique et sémantique la requête sera de type SQL, comme

```
SELECT   X1 X2
FROM     R1 R2
WHERE    R2.Y1 = a
     et  R3.Y2 = b
```

l'arbre général est défini sur trois relations (ou trois bases).

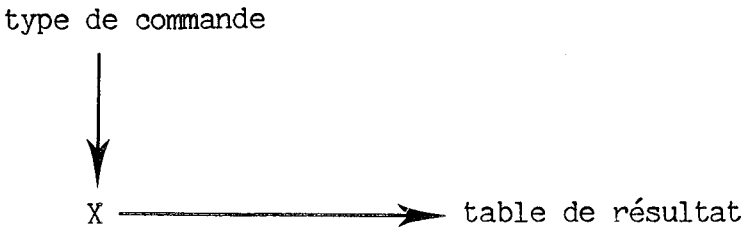
Pour réaliser la JOINTURE entre ces trois bases (ou relations) on génère pour chaque résultat le champ commun aux trois base. Dans la table de travail, ce sera le champ de JOINTURE, et l'arbre de la forme :


Project
Select -----> X1
R1 ----> =
A2 ----> I S I I Table de travail
(pointe de JOINTURE)
Project
Select -----> X2,A2(Un des champs commun pour execute le JOINTURE)
R2 ----> AND
= ------> =
Y---> a
A2 ----> I S I I Table de travail
Retransfere le con-
enu de champ commun apres la réalis-
ation de Y2 =b
Project
Select -----> A1(Un des champs commun pour execute le JOINTURE)
R3 ----> =
Y2 ----> "b"

2 - Requête simple

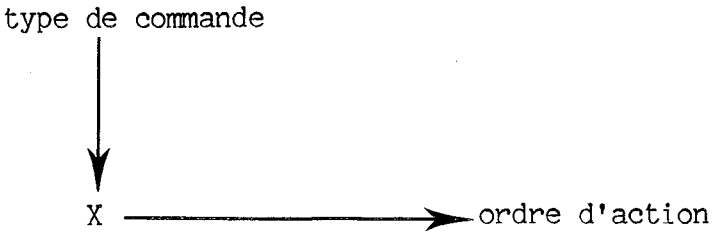
D'après la fixation de la relation à la base l'arbre prend le schéma simple.

Requête d'action



où X : identificateur d'attribut (ou de fichier).

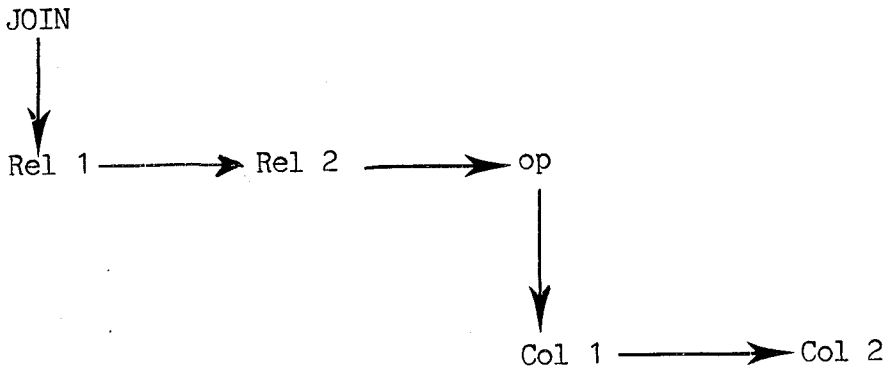
Requête d'utilisation



B - Opérateurs algébriques

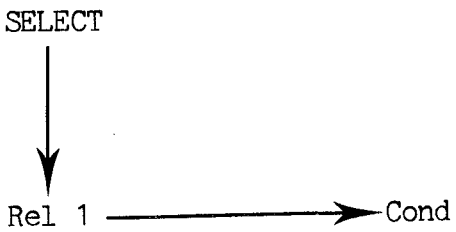
1 - JOIN : entre deux relations (ou bases) :

Le résultat de JOIN est une nouvelle relation de la forme :



où "op" est l'un des opérateurs arithmétiques suivants : { =,<>,<=,>=,>,<,≠ }
Col 1 (Col 2) appartient à l'ensemble des colonnes de Relation 1 (Relation 2).

2 - SELECT



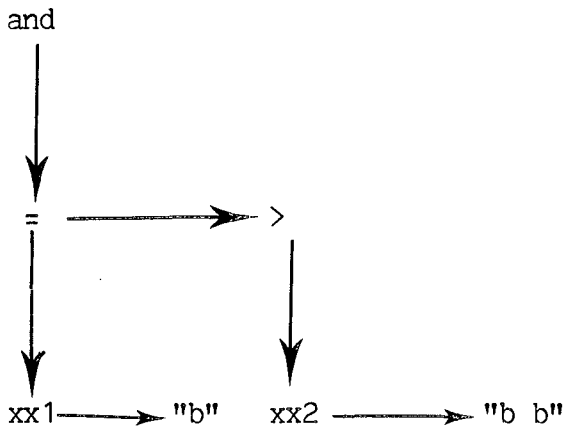
Le résultat SELECT est une relation dont les colonnes sont exactement celles de REL 1.

La condition notée COND est un sous-arbre dont les noeuds sont les opérateurs arithmétiques ou logiques et les feuilles de l'arbre des items ou des contantes.

Exemple

((xx1 = "b") and (xx2 > "b b"))

l'arborescence correspondante est :



3 - PROJECT

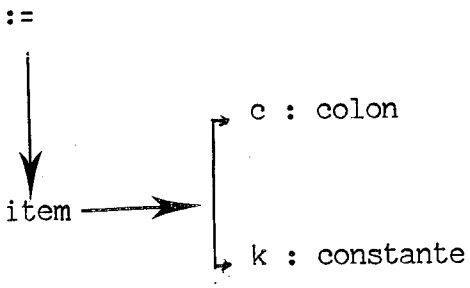
Le résultat de PROJECT est une relation.

COL 1, COL 2 .... appartient à l'ensemble des colonnes de REL 1

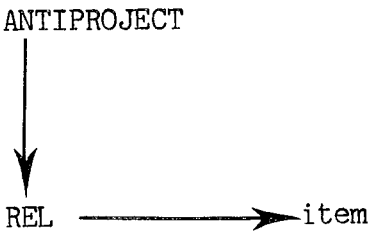
4 - MODIF

. Le résultat de cet opérateur est une relation

. Affectation est le sous-arbre suivant :

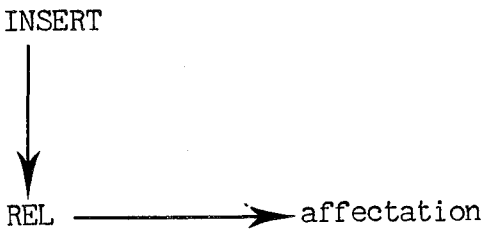


5 - ANTIPROJECT



La relation obtenue ici est formée d'une seule colonne (attribut).

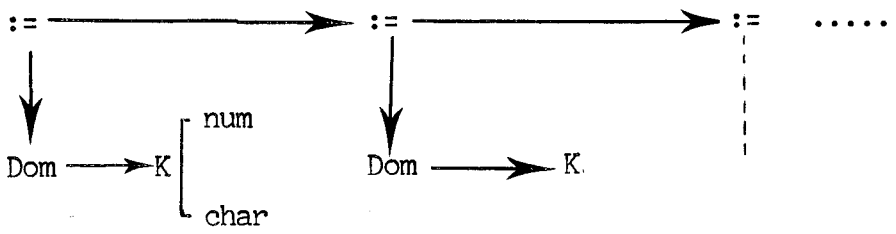
6 - INSERT



Cet opérateur permet d'insérer des items dans les relations (REL) précisées dans "affectations"

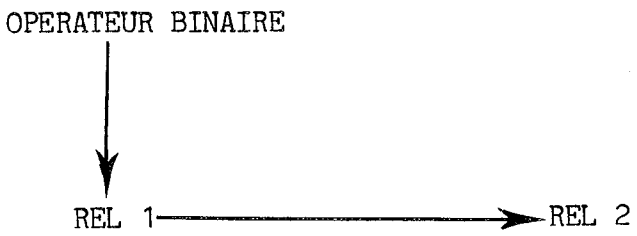
. Affectation est une suite d'affectation élémentaires

Exemple d'affectation



7 - PRODUIT, UNION, INTERSECTION, DIFFERENCE

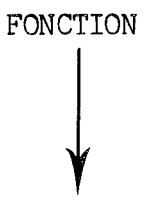
Ces 4 opérateurs sont binaires ; leur arborescence d'interprétation est la suivante :



Pour les opérateurs UNION, INTERSECTION, et DIFFERENCE les 2 relations opérandes REL 1 et REL 2 doivent avoir le même nombre de domaines, un à un compatible.

C - Fonction

MAX, MIN, MOY, SOM



. Le résultat de la fonction est réel ou entier.

. Les attributs de relation doivent être de type réel ou entier.

3) Fonctionnement du compilateur

On va dégager dans ce paragraphe les grandes lignes du fonctionnement d'un compilateur de compilateur pendant le passage de la requête écrite en langage source, à la forme de l'arborescence finale qui permet d'accéder aux informations.

Ce compilateur de compilateur fonctionne en trois étapes, en utilisant: les structures de système IMREL, la définition du langage de requête et les structures de compilateur. On va présenter l'algorithme général du fonctionnement du compilateur.

Algorithme du fonctionnement du compilateur

I - Initialisation et analyse lexicale

II - Passage à l'analyse syntaxique et partage de la requête en sous requête

III - Passage à l'analyse sémantique

si - requête simple :
- génération du code
- aller en IV

si - requête composée :
- passage sur les trois types de filtres et procédure sémantique pour fixer l'ordre du traitement des sous requêtes.
- génération du code
- exécution de la sous requête
- aller en IV

IV - Si fin d'exécution aller en V
Sinon prendre la sous requête suivante
Aller en III

V - FIN

- Support matériel

La configuration matérielle qui est un support pour la réalisation de ce travail se compose de :

* un mini-ordinateur HP, modèle 1000, la taille de mémoire 1,5 mego octet
* plusieurs terminaux à écran
* deux unités de disque de 120 Mo
* une imprimante de 400 lignes/minute
* le logiciel RTE6/VM constitue le système d'exploitation, d'autres logiciels sont disponibles :

  - Macro Assembleur
  - FORTRAN 77
  - Basic
  - Pascal

Tableaux de travail dans un compilateur

* 16 tableaux, pour les 16 mots de requête
  (MREij) si i = "Q" Alors j∈{1,2,.....,9}
  i = "1" Alors j∈{1,2,.....,6}

* Tableau de condition pour chaque points d'entrée à la base (MiTEC, MiPRO, MiLOT, MiTRA, MiMOT, MiRES, MiDAT, MiMOY) ou i∈ 1,2,3}

* Tableau de génération en utilisant l'équivalence entre eux : (TARBE, TCOND).

Le chemin d'exécution

Comme nous l'avons vu dans l'algorithme de fonctionnement de compilateur, l'exécution passe par trois étapes :

Première étape : Dans cette étape, on effectue l'analyse syntaxique de la requête, par le passage de sous-programme (PRANC), en décomposé pour chaque requête les sous-requêtes. Nous définissons leurs identificateurs de fichier et leurs bases par le passage du sous-programme DBASi

(DBASi) Vi ∈ {1,2,.....,n } "n" nombre des bases), et avec l'analyse syntaxique des différentes bases, on assure la division de la requête en sous requêtes.

Après le message qui dit, l'analyse syntaxique est faite, on applique le sous programme appelé (SELEC) qui réalise les actions sémantiques. Il fixe l'ordre de passage de traitement pour chaque sous-requête.

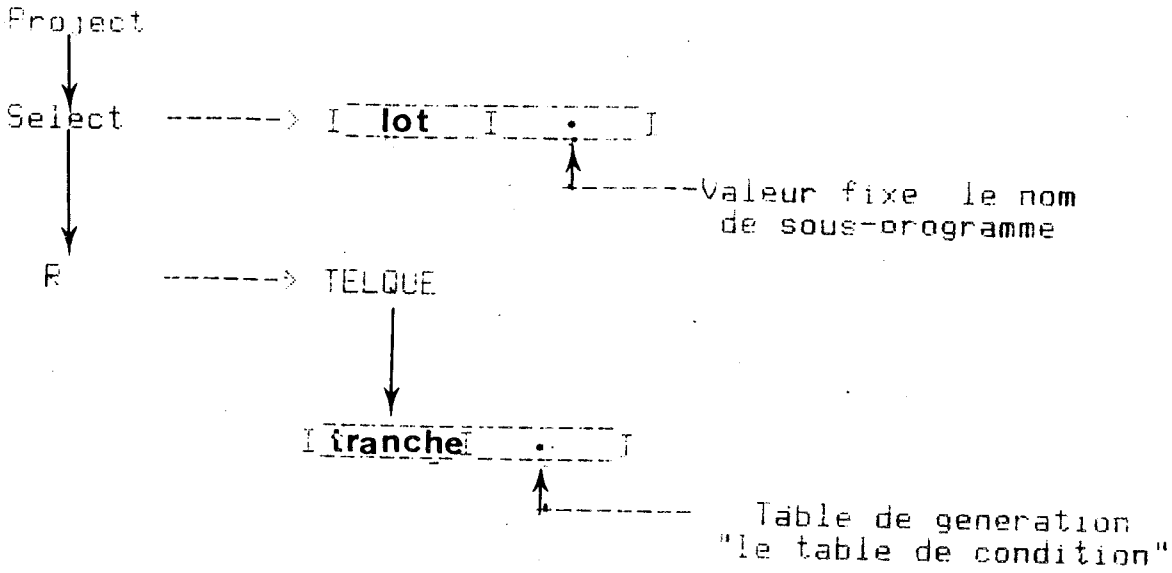
On génére pour chaque sous-requête le sous programme qui réalise leur exécution.

Exemple

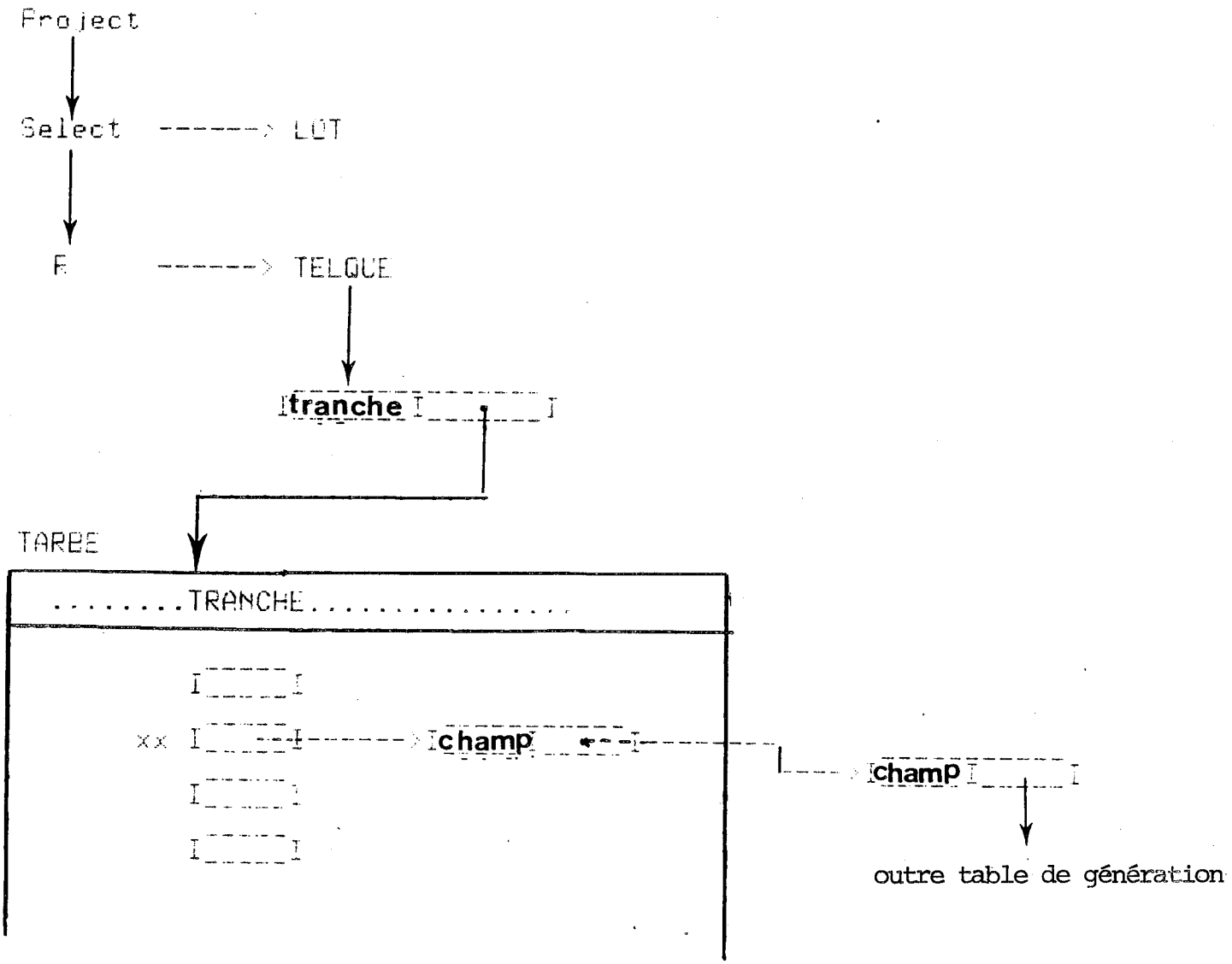
Soit la sous-requête suivante :

SELECT LOT TELQUE TRANCHE "xx" MOTIF "yy" ;

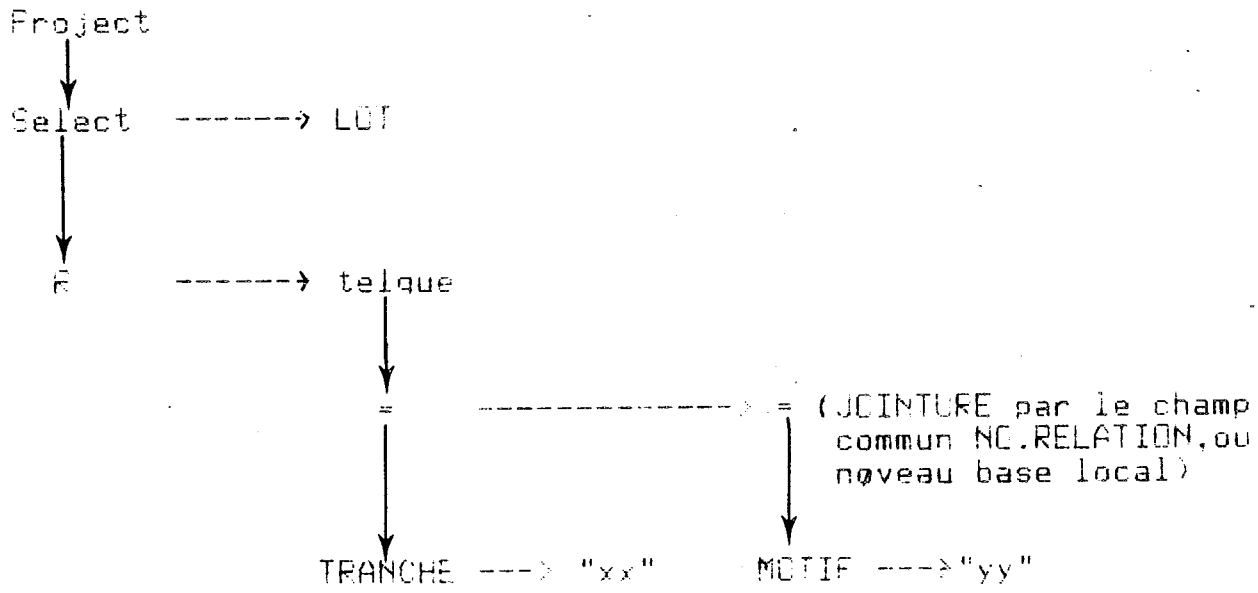
L'arborescence partielle après la transformation en sous requête, de type SQL est :



Deuxième étape : Dans cette étape, on traduit la 1ère condition existant dans la table de génération, d'après le sous programme fixé au niveau select dans l'arborescence. On va simuler dans le tableau "TARBE" le contenu du fichier R et on va stocker le champ commun avec l'autre relation, et l'arborscence à ce niveau est :



d'après la table de génération, on va toujours monter dans l'arborescence conditionnelle jusqu'à la fin des conditions. On effectue la transformation à partir des valeurs finales dans la table de génération puis on s'arrête, et l'arborescence finale est



NB : dans la requête on peut accéder à tous les champs existant dans la base.

Troisième étape : C'est l'étape la plus importante. En effet dans cette étape on parcourt plusieurs bases, on cherche le résultat de la deuxième étape qui est stocké dans une table de travail et ce résultat avec l'autre condition contenue dans la sous requête va interroger le niveau Base en utilisant l'opérateur Algébrique (JOINTURE).

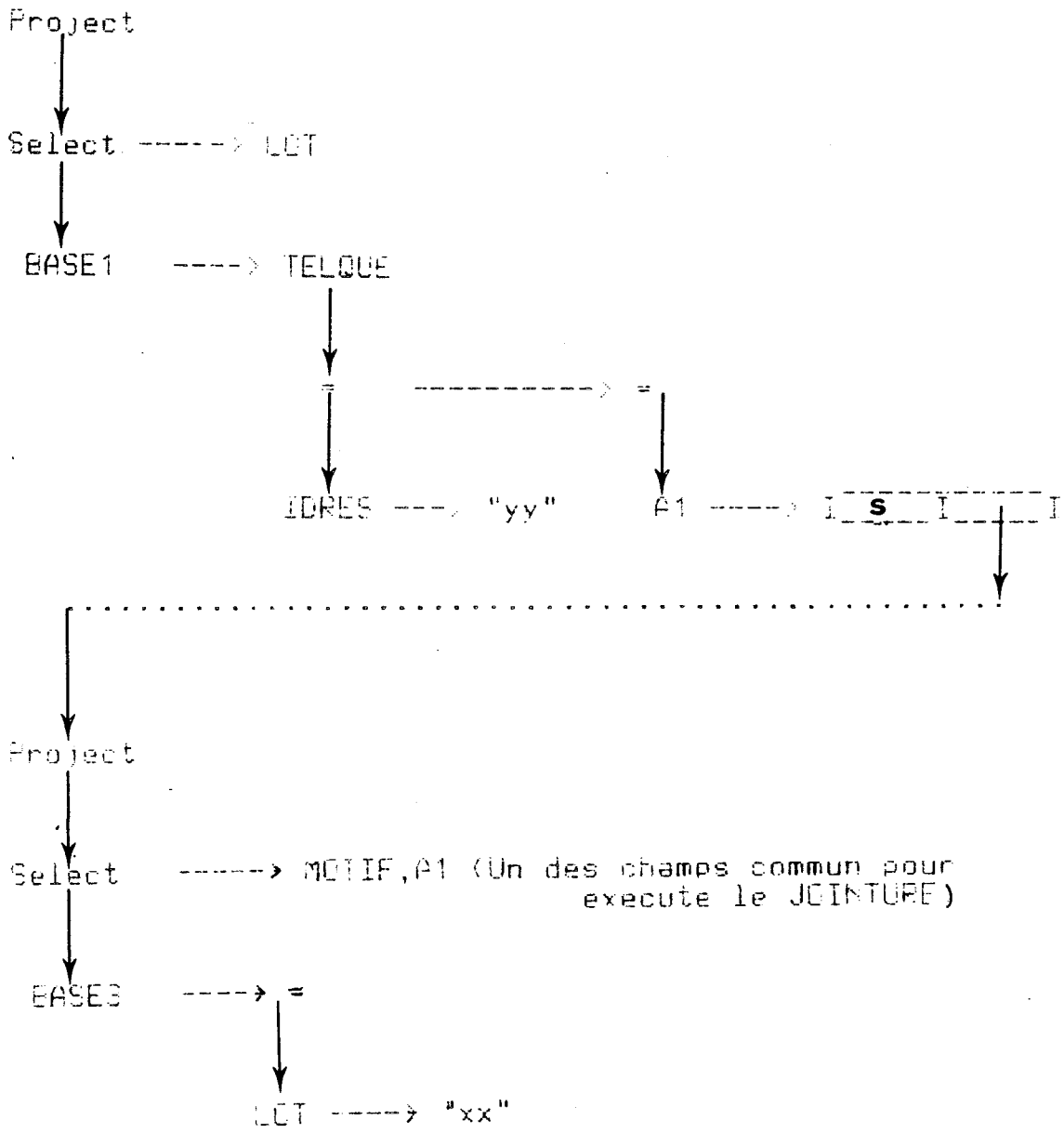
Exemple

Soit la requête :

SELECT LOT + 1 MOTIF + 3 TELQUE LOT + 3 "xx" IDRES + 1 "yy" ;

ou LOT + 1, IDRES + 1 de base 1, et LOT + 3, MOTIF + 3 de base 3

D'après les trois étapes l'arborescence prend la forme suivante :



IV - 3.4. Vue d'ensemble

La configuration générale de notre travail peut être représentée de la façon suivante (figure 25)

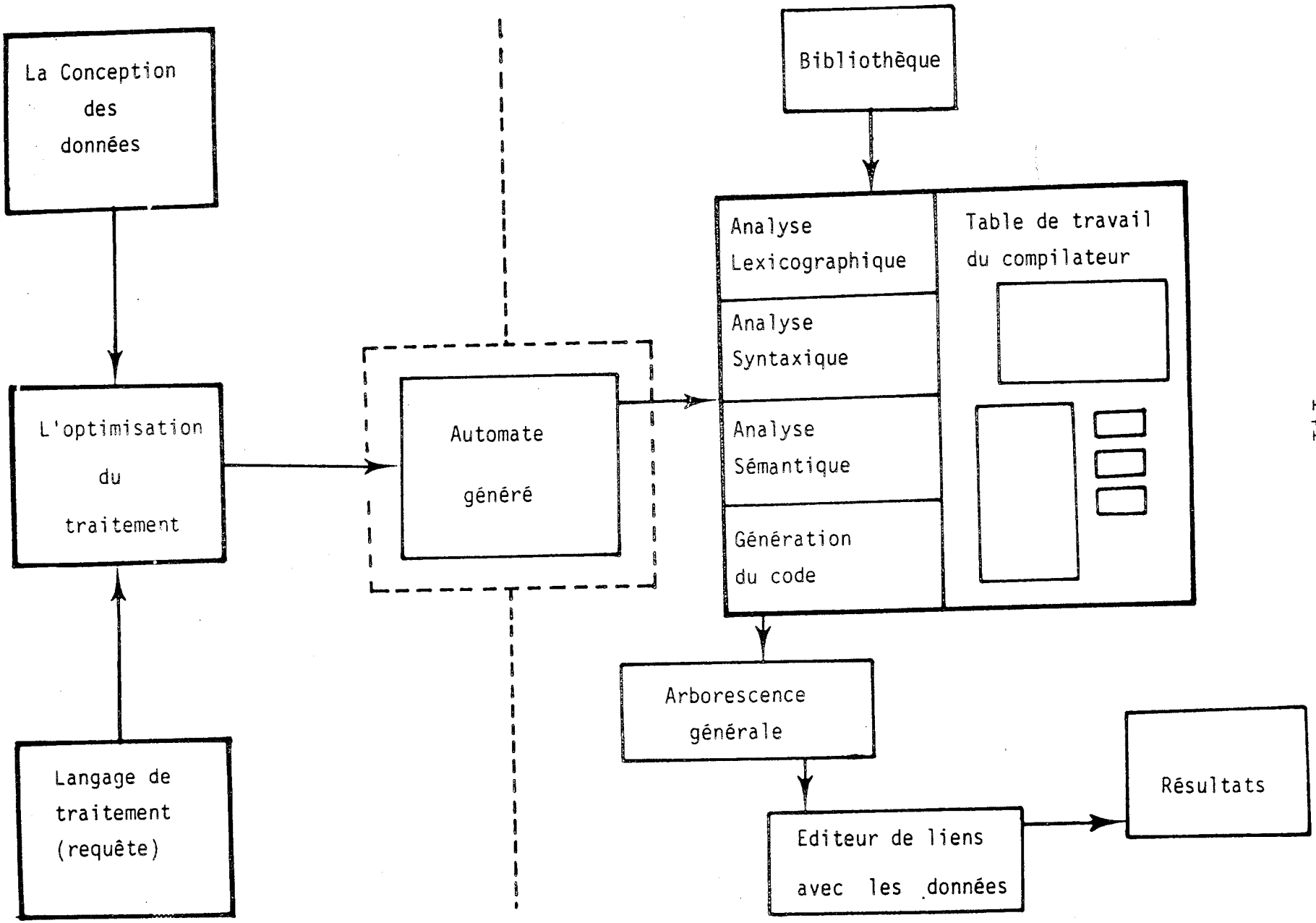


Fig.25 Vue d'ensemble

IV - 4. CONCLUSION

Conscient de l'utilité de notre modèle de Base de données et du langage de requête et de son optimisation, nous avons réalisé l'écriture du compilateur de requête.

Nous avons essayé d'en rendre l'emploi le plus souple possible : facilité d'écriture de la requête, définition de tous les points dans le compilateur, optimisation du chemin de traitement d'une requête en utilisant la règle sémantique qui a été défini dans le compilateur, interrogation de plusieurs bases de données, à partir de tous les champs définis dans la base.

BIBLIOGRAPHIE

(AHO 79) A. AHO, J. ULLMAN
"Principles of compiler design"
Addition WESLEY publishing Company, April 1979.

(AMI 74) M. AMIRCHAHY, M. MAZAUD, D. NOEL
"Projet DELTA : optimisation de ocde généré par attributs"
Rapport de recherche, n°59, IRIA, Février 1974.

(BAR 79) J. BARRE
"Système d'apprentissage des techniques élémentaires de compilation"
Thèse 3ème Cycle, Rennes, Juin 1979.

(COU 71) A. COUVER, J.P. ROUTEAU
"Description général du programme de construction du compilateur P1/002 et P1/002"
Rapport IRISA, Rennes, 1971.

(CUN 80) P.Y. CUNIN, M. GRIFFITHS, J. VOIRON
"Comprendre la compilation"
Springer Verlhag, 1980.

(DES 80) P. DESCHAMP
"Production de compilateur à partir d'une description sémantique des langages de programmation, le système PERLUETTE"
Thèse Docteur-Ingénieur, INPL (Lorraine), Octobre 1980.

(FER 80) L. FERRAT
"Définition et traduction en arborescence algébrique d'un langage de base de données relationnel de haut niveau MICROSEQUEL"
Rapport DEA, IMAG (Grenoble), Septembre 1980.

(GAL 77) H. GALLAIRE
"Techniques de compilation - Méthodes d'analyses syntaxiques"
Cepadues Edition, Toulouse, 2ème trimestre 1977.

(GAUD 80) M.C. GAUDEL
"Génération et preuve de compilateurs basées sur une sémantique formelle des langages de programmation"
Thèse Docteur d'Etat, INPL (Lorraine), 1980.

(GAUT 80) Th. GAUTIER
"Conception et écriture d'un producteur d'analyseurs descendants récursifs"
Rapport INSA, Rennes, Juin 1980.

(HOP 70) F.R.A HOPGOOD
"Techniques de compilation"
Dunod, Paris, 1970.

(KOH 70) Z. KOHAVI
"Switching and finite automata theory"
McGraw Hill Book Company, 1970.

(LOU 81) A.M. LOUNT, W.F. SMYTH
"Stand alone data manipulation programs"
Computers en industry, n°2, 1981.

(ROH 80) J. ROHMER
"Machines et langages pour traiter les ensembles de données (textes, tableaux, fichiers)"
Thèse d'Etat, INPG (Grenoble), 1980.

(SAK 80) U. SAKELLARIDIS
"Generation automatique de compilation PL. CROISES"
Rapport DEA, St Etienne, Novembre 1980.

(WEL 82) M.C. WELL, H.A. SHULL
"An expression model for extraction and evolution of parallelism in control structures"
IEEE Transaction on computer, Septembre 1982.

CHAPITRE V

CONCLUSION

La démarche adoptée dans cette étude consiste dans un premier temps, à montrer qu'il est possible de construire des mécanismes pour définir les informations stockées dans un système Hiérarchique (IMAGE), et s'exprimer en langage relationnel. Nous avons défini la notion d'enregistrement, de description et connexions permettant d'établir un schéma d'accès à chaque relation, en utilisant les mécanismes de fichier maître. Nous avons également présenté les différents tableaux de travail que nous considérons comme un passage intermédiaire entre les relations ne contenant pas de clés communes et un calcul de la distance entre deux relations qui nous donne la possibilité de réaliser un filtrage de traitement au niveau des structures de données.

Et dans un deuxième temps, l'idée du système IMREL nous a amené à définir un langage de requête pour interroger une ou plusieurs bases. Nous avons modélisé la requête, (langage naturel) traité et traduit sous la forme SQL par l'analyseur syntaxique, chaque requête a été découpée sous forme de plusieurs sous requêtes (ch. I). On a optimisé le traitement au niveau des structures de requête en utilisant le calcul de la priorité de passage par l'analyseur sémantique (ch. I, III), dans toutes les étapes nous avons utilisé les opérateurs relationnels au niveau du traitement global, et au niveau local nous avons utilisé des procédures optimales définies pour le modèle hiérarchique, et les opérateurs relationnels pour le modèle relationnel.

Dans un troisième temps, nous avons formalisé le processus d'optimisation (ch. III) d'une requête en utilisant le mécanisme de calcul des coûts à l'aide d'un modèle mathématique pour évaluer les exécutions de l'opérateur P-Jointure (semi-Jointure), le calcul des coûts minimaux est fait après l'estimation de la taille de chaque relation à la fin de chaque étape de traitement.

Enfin nous sommes passés du stade théorique de l'étude du phénomène de modélisation, conception, optimisation et traitement au stade de réalisation par le compilateur qui était mis en service sur l'application. La taille des programmes source est = 27500 lignes non commentées.

Applications typiques et recherches futures

L'idée au départ était de réaliser un compilateur de requêtes pour les bases de données technologie-modèle.

En effet, la construction d'une base de données est maintenant une techniques bien au point et largement utilisée, mais les conceptions et la réalisation d'une interface entre les différents types est un travail de recherche, de plus l'impossibilité actuelle du modèle hiérarchique (IMAGE) pour interroger plusieurs bases, nous a conduit à définir sur le SGBD implanté, une interface permettant la manipulation du modèle relationnel d'une base de données gérée par un SGBD hiérarchique.

La constitution de l'interface a été faite en quatre étapes :

- le développement d'un système spécifique entre les deux modèles avec pour interface le système IMREL.

- La nécessité d'avoir un langage pour traduire la requête relationnelle en une requête hiérarchique.

- Optimiser le langage d'interrogation pour arriver à un traitement optimal.

- La réalisation pratique découlant des trois étapes précédentes.

Certains points présentés dans cette étude et réalisation mériteraient des prolongements. Il s'agit de parvenir à une nouvelle description concernant le traitement des requêtes données, et répercuter les modifications au niveau du traitement des données. Le concepteur doit s'intéresser en premier lieu à définir comment implanter les données dans l'espace de mémorisation, afin de répondre complètement et correctement aux demandes des différents utilisateurs de la future base.

Il est conduit à exprimer la solution logique, qui montre les regroupements de données et les cheminements logiques entre les structures. Le schéma logique est une structure d'accès aux données qui représente une réalité organisationnelle.

Le schéma logique est le résultat d'un travail de modèlisation qui tient compte de la modèlisation des bases et des besoins de renseignements permanents ou aléatoires exprimées sous forme de fonctions d'accès.

La première difficulté de cette étape réside dans l'inventaire des besoins à prendre en compte. Cet inventaire inclut les données qui ont été exprimées lors de l'analyse par les utilisateurs déjà répertoriés dans la future base, mais doit également s'étendre aux nouvelles demandes qui pourraient être présentées soit par les utilisateurs déjà répertoriés, soit par de nouveaux utilisateurs. Il n'est pas inutile dans cet inventaire que le concepteur simule le comportement de certains utilisatuers pour compléter l'inventaire.

La deuxième difficulté est liée à l'abondance des solutions possibles. La richesse des besoins peut conduire le concepteur à multiplier les fonctions d'accès pour se mettre plus aisèment dans l'espace de mémorisation. La multiplication des fonctions d'accès peut amener le concepteur à introduire des redondances systématiquement écartées à la conception. A l'opposé, il a intérêt à conserver un maximum de simplicité dans le schéma logique. Celui-ci assure un compromis délicat à déterminer, entre la redondance, la simplicité de la structure et la charge supplémentaire de programmation.

Cela ne signifie pas que le concepteur remet en cause la solution obtenue lors de l'étape de modélisation des bases. Il est simplement conduit, dans la poursuite de ce travail, à compléter une solution qui au niveau conceptuel, atteint un certain contenu et une certaine forme.

Ces deux points font l'objet d'une recherche, et d'uatre part, cette étude permet dans le cadre du développement des bases de données multiples et réparties hétérogènes, de définir une couche de logiciel universel homogénéisant les ensembles de bases de données. On prend en compte l'existence des différents travaux qui a été fait dans ce domaine (URANUS, POLYPHEME, ...).

ANNEXE I

Cette annexe contient les resultats d'exécution de plusieurs exemples de requétes,par le compilateur qu'on a mis au point (programme source ≃ 27500 lignes hors commentairès) en suivant le schéma de réalisation élaboré dans la partie théorique de cette thése .

REMARQUE 1:

On a pu enlever l'ambiguite dans les attributs des bases,gráce au suffixe [+i](où i desigine le numéro de la base corspondante) qu'on attache au mot qui définit l'attribut.

EX : "TECHNOLOGIE" est prévue dans trois bases
alors : TECHNOLOGIE+1 correspond a la BASE1
TECHNOLOGIE+2 correspond a la BASE2
TECHNOLOGIE+3 correspond a la BASE3

REMARQUE 2:

L'expression entre " " représente la valeur de l'attribut qui le précéde, le blanc etant pris en compte

EX : TECHNOLOGIE "HMOS I" signifie qu'on donne à TECHNOLOGIE la valeur "HMOS I" ;

REMARQUE 3:

La valeur "EN830101830131" représente le délai temporaire compris entre le 01/01/83 et le 31/01/83.

```
-----------------------------------------------
I                                             I
I     COmpilateur de Requetes RELationnel     I
I              ***  CORREL  ***               I
I            a votre se rvice ...             I
I                                             I
-----------------------------------------------
```

```
 ENTREZ VOTRE REQUETE S.V.P...

-->SELECT TRANCHE+1 LOT+3 SIG+3 TELQUE TECHNOLOGIE+1 "HMOS II   LOCOS"
>>==>PROVENANCE+1 "DT TI" NOM-MOTIF+1 "TMOS" NOM-RESULTATS+3 "DELTAL"
>>==>PROVENANCE+3 "DT TI"  ;
 Requete valide au niveau de l analyse SYNTAXIQUE
  ANALYSE SEMANTIQUE TERMINEE...


 Ouverture de la BASE1...

 Recherche en BASE1    :  TRANCHE
          Telque       :  TECHNOLOGIE      : HMOS II   LOCOS
          Telque       :  PROVENANCE       : DT TI
          Telque       :  NOM-MOTIF        : TMOS
Fermeture de la BASE1 .


**************************************************
* INTERROGATION DE LA BASE1-CARACTERISATION.R: 7*
**************************************************

 Technologie              Provenance          Lots              Tranche
 HMOS II   LOCOS            DT TI           SIM                  S4 T
MOTIF : nom                 sous-types1     sous-types2    sous-types3
         TMOS              ENRICHI          N
        dim.geo.1:nom      valeur        -   dim.geo.2:nom      Valeur
         LARGEUR           100              LONGUEUR        100
        dim.geo.3:nom       valeur


IDRES : Nom                  Methode
         VS                CLASSIQUE
        Mesure 1:Nom       Valeur     Mesure 2:Nom     Valeur
         VSUBSTRAT     0               TEMPERATUR    22
        Mesure 3:Nom       Valeur     Mesure 4:Nom     Valeur


 Technologie              Provenance          Lots              Tranche
 HMOS II   LOCOS            DT TI           SIM                  S4 T
MOTIF : nom                 sous-types1     sous-types2    sous-types3
         TMOS              ENRICHI          N
        dim.geo.1:nom      valeur            dim.geo.2:nom      Valeur
         LARGEUR           50               LONGUEUR        2.8
        dim.geo.3:nom       valeur


IDRES : Nom                  Methode
         VS                CLASSIQUE
        Mesure 1:Nom       Valeur     Mesure 2:Nom     Valeur
```

```
        VSUBSTRAT     0              TEMPERATUR    22
       Mesure 3:Nom    Valeur     Mesure 4:Nom     Valeur

 Technologie              Provenance          Lots              Tranche
 HMOS II   LOCOS             DT TI          SIM                   S4 T
MOTIF : nom                  sous-types1    sous-types2    sous-types3
        TMOS                ENRICHI         N
       dim.geo.1:nom        valeur           dim.geo.2:nom      Valeur
        LARGEUR             50              LONGUEUR        2.4
       dim.geo.3:nom         valeur

IDRES : Nom                   Methode
        VS                  CLASSIQUE
       Mesure 1:Nom         Valeur     Mesure 2:Nom     Valeur
        VSUBSTRAT     0               TEMPERATUR    22
       Mesure 3:Nom         Valeur     Mesure 4:Nom     Valeur

 Technologie              Provenance          Lots              Tranche
 HMOS II   LOCOS             DT TI          SIM                   S4 T
MOTIF : nom                  sous-types1    sous-types2    sous-types3
        TMOS                ENRICHI         N
       dim.geo.1:nom        valeur           dim.geo.2:nom      Valeur
        LARGEUR             6               LONGUEUR        20
       dim.geo.3:nom         valeur

IDRES : Nom                   Methode
        VS                  CLASSIQUE
       Mesure 1:Nom         Valeur     Mesure 2:Nom     Valeur
        VSUBSTRAT     0               TEMPERATUR    22
       Mesure 3:Nom         Valeur     Mesure 4:Nom     Valeur

 Technologie              Provenance          Lots              Tranche
 HMOS II   LOCOS             DT TI          SIM                   S4 T
MOTIF : nom                  sous-types1    sous-types2    sous-types3
        TMOS                ENRICHI         N
       dim.geo.1:nom        valeur           dim.geo.2:nom      Valeur
        LARGEUR             4               LONGUEUR        20
       dim.geo.3:nom         valeur

IDRES : Nom                   Methode
        VS                  CLASSIQUE
       Mesure 1:Nom         Valeur     Mesure 2:Nom     Valeur
        VSUBSTRAT     0.              TEMPERATUR    22
       Mesure 3:Nom         Valeur     Mesure 4:Nom     Valeur

IDRES : Nom                   Methode
        MOBILITE            SUR 1 TMOS
       Mesure 1:Nom         Valeur     Mesure 2:Nom     Valeur

       Mesure 3:Nom         Valeur     Mesure 4:Nom     Valeur

 Technologie              Provenance          Lots              Tranche
```

```
 HMOS II   LOCOS          DT TI          SIM                    S4 T
MOTIF : nom               sous-types1    sous-types2   sous-types3
         TMOS            ENRICHI        N
        dim.geo.1:nom    valeur          dim.geo.2:nom      Valeur

        dim.geo.3:nom     valeur


IDRES : Nom               Methode
         DELTAL         DL ET DZ#0
        Mesure 1:Nom     Valeur     Mesure 2:Nom     Valeur

        Mesure 3:Nom     Valeur     Mesure 4:Nom     Valeur


IDRES : Nom               Methode
         DELTAZ         DL ET DZ#0
        Mesure 1:Nom     Valeur     Mesure 2:Nom     Valeur

        Mesure 3:Nom     Valeur     Mesure 4:Nom     Valeur


IDRES : Nom               Methode
         MOBILITE       DL ET DZ#0
        Mesure 1:Nom     Valeur     Mesure 2:Nom     Valeur

        Mesure 3:Nom     Valeur     Mesure 4:Nom     Valeur


 Recherche en BASE3    :  LOT
 Recherche en BASE3    :  SIG
          Telque       :  TECHNOLOGIE       : HMOS II   LOCOS
          Telque       :  NOM-RESULTATS     : DELTAL
          Telque       :  PROVENANCE        : DT TI
 Fermeture de la BASE3.
 Generation de la condition de sous requete dans la BASE3 . TYPE R.= 11


**************************************************
* INTERROGATION DE LA BASE3-CARACTERISATION.R:11*
**************************************************

 Technologie              Provenance   Lots                   Tranche
 HMOS II   LOCOS          DT TI        SIM                    S4 T


MOTIF :   Nom             Sous-types1     Sous-types2   Sous-types3
          TMOS            ENRICHI         N
          dim.geo.1:Nom Valeur            dim.geo.2:Nom Valeur

          dim.geo.3:Nom  Valeur


IDRES :   Nom             Methode
          DELTAL          DL ET DZ#0
          Mesure 1:Nom Valeur          Mesure 2:Nom Valeur

          Mesure 3:Nom Valeur          Mesure 4:Nom Valeur
```

```
VALEUR: Type      du            N.P.T.     N.T.B.   N.T.R.
        CA     25/01/83           52          46        37
             MIN                 MAX                  MOY
             3.410E-01           4.847E-01          4.262E-01
             SIG                 SKE                  KUP
             1.133E-03          -3.358E-01         -1.586E-01
        Commentaires



FIN traitement requete courante. Prendre la SUIVANTE

 ENTREZ VOTRE REQUETE S.V.P...

-->SELECT MIN+3 NOM-MOTIF+1 TELQUE TECHNOLOGIE+1 "DMOS HT"  LOT+1 "4-12"
>>==>TRANCHE+1 "1"  NOM-RESULTATS+3 "VS"  ;
 Requete valide au niveau de l analyse SYNTAXIQUE
  ANALYSE SEMANTIQUE TERMINEE...


 Ouverture de la BASE1...

 Recherche en BASE1    :  NOM-MOTIF
          Telque       :  TECHNOLOGIE       : DMOS HT
          Telque       :  LOT               : 4-12
          Telque       :  TRANCHE           : 1
Fermeture de la BASE1 .


**************************************************
* INTERROGATION DE LA BASE1-CARACTERISATION.R: 5*
**************************************************



Technologie              Lot                        Tranche
 DMOS HT                   4-12                       1

MOTIF : nom               sous-types1   sous-types2   sous-types3
         RESISTANCE       C. ENT. N+     2 POINTES
        dim.geo.1:nom     valeur        dim.geo.2:nom    Valeur
         LONGUEUR         187            LARGEUR         75
        dim.geo.3:nom      valeur

IDRES : Nom                Methode
         RESISTANCE     2 POINTES
        Mesure 1 :Nom    Valeur     Mesure 2 :Nom    Valeur

        Mesure 3 :Nom    Valeur     Mesure 4 :Nom    Valeur



Technologie              Lot                        Tranche
```

```
 DMOS HT                    4-12                          1

MOTIF : nom                 sous-types1   sous-types2   sous-types3
         RESISTANCE         PUIT COLLEC.   2 POINTES
        dim.geo.1:nom       valeur        dim.geo.2:nom     Valeur
         LONGUEUR           375            LARGEUR          75
        dim.geo.3:nom        valeur

IDRES : Nom                  Methode
         RESISTANCE    2 POINTES
        Mesure 1 :Nom   Valeur     Mesure 2 :Nom   Valeur

        Mesure 3 :Nom   Valeur     Mesure 4 :Nom   Valeur


Il n y a plus d autres elements pour cette Technologie-Lot-Tranche. _
 Recherche en BASE3    :  MIN
          Telque       :  TECHNOLOGIE       : DMOS HT
          Telque       :  NOM-RESULTATS     : VS
 Fermeture de la BASE3.
 Generation de la condition de sous requete dans la BASE3 . TYPE R.= 10


**************************************************
* INTERROGATION DE LA BASE3-CARACTERISATION.R:10*
**************************************************

 Technologie                Provenance   Lots                  Tranche
 DMOS HT                    DCS          4-12                  2

MOTIF :  Nom                sous-types1    sous-types2   sous-types3
         TMOS               ENRICHI        P
         dim.geo.1:Nom Valeur              dim.geo.2:nom Valeur
         LARGEUR            50             LONGUEUR       50
         dim.geo.3:Nom  Valeur

IDRES :  Nom                Methode
         VS                 CLASSIQUE
         Mesure 1:Nom Valeur          Mesure 2:Nom Valeur
         VSUBSTRAT    0               TEMP.        22
         Mesure 3:Nom Valeur          Mesure 4:Nom Valeur

VALEUR:  Type      du          N.P.T.     N.T.B.   N.T.R.
         CA     21/03/83         5          5        5
              MIN                MAX                MOY
           -1.638E+00         -1.560E+00         -1.578E+00
              SIG                SKE                KUP
              3.337E-02          -1.056E+00         -9.407E-01
          Commentaires

MOTIF :  Nom                sous-types1    sous-types2   sous-types3
         TMOS               ENRICHI        P
         dim.geo.1:Nom Valeur              dim.geo.2:nom Valeur
         LARGEUR            100            LONGUEUR       11
```

dim.geo.3:Nom Valeur

IDRES : Nom Methode
VS CLASSIQUE
Mesure 1:Nom Valeur Mesure 2:Nom Valeur
VSUBSTRAT 0 TEMP. 22
Mesure 3:Nom Valeur Mesure 4:Nom Valeur

VALEUR: Type du N.P.T. N.T.B. N.T.R.
CA 21/03/83 5 3 3
MIN MAX MOY
-1.455E+00 -1.447E+00 -1.451E+00
SIG SKE KUP
4.046E-03 6.245E-02 -2.333E+00
Commentaires

Technologie Provenance Lots Tranche
DMOS HT DCS 4-12 1

MOTIF : Nom sous-types1 sous-types2 sous-types3
DMOS VERTICAL N
dim.geo.1:Nom Valeur dim.geo.2:nom Valeur
LARGEUR 220 LONGUEUR 18
dim.geo.3:Nom Valeur

IDRES : Nom Methode
VS CLASSIQUE
Mesure 1:Nom Valeur Mesure 2:Nom Valeur
VSUBSTRAT 0 TEMP. 22
Mesure 3:Nom Valeur Mesure 4:Nom Valeur

VALEUR: Type du N.P.T. N.T.B. N.T.R.
CA 23/03/83 5 4 4
MIN MAX MOY
1.651E+00 1.680E+00 1.666E+00
SIG SKE KUP
1.453E-02 -3.633E-02 -2.360E+00
Commentaires

MOTIF : Nom sous-types1 sous-types2 sous-types3
DMOS VERTICAL N
dim.geo.1:Nom Valeur dim.geo.2:nom Valeur
LARGEUR 60 LONGUEUR 18
dim.geo.3:Nom Valeur

IDRES : Nom Methode
VS CLASSIQUE
Mesure 1:Nom Valeur Mesure 2:Nom Valeur
VSUBSTRAT 0 TEMP. 22
Mesure 3:Nom Valeur Mesure 4:Nom Valeur

```
VALEUR:  Type      du          N.P.T.     N.T.B.   N.T.R.
         CA     23/03/83         5          5         5
             MIN                MAX                MOY
             1.589E+00        1.656E+00          1.626E+00
             SIG                SKE                KUP
             2.738E-02       -1.958E-01         -1.929E+00
              Commentaires
```

FIN traitement requete courante. Prendre la SUIVANTE

ENTREZ VOTRE REQUETE S.V.P...

```
-->SELECT NOM-MOTIF SOUS-TYPE:1 TRANCHE MOY TELQUE DATE "EN8301018301311"
 Requete valide au niveau de l analyse SYNTAXIQUE
 La requete concerne une seule base : BASE3 .
  ANALYSE SEMANTIQUE TERMINEE...
 Recherche en BASE3    :  NOM-MOTIF
 Recherche en BASE3    :  SOUS-TYPE:1
 Recherche en BASE3    :  TRANCHE
 Recherche en BASE3    :  MOY
          Telque       :  DATE                : EN8301018301311
 Fermeture de la BASE3.
 Generation de la condition de sous requete dans la BASE3 . TYPE R.= 20
```

```
*****************************************************
* INTERROGATION DE LA BASE3-CARACTERISATION.        *
* Question : DATE      Reponses : TOUTES            *
*****************************************************
```

```
 Technologie                Provenance   Lots                    Tranche
 HMOS II   LOCOS            DT TI        SIM                     I S5

MOTIF :  Nom              sous-types1     sous-types2    sous-types3
         RESISTANCE       N+              4 POINTES
         dim.geo.1:Nom Valeur             dim.geo.2:nom Valeur
         LONGUEUR         200             LARGEUR         4
         dim.geo.3:Nom  Valeur

IDRES :  Nom              Methode
         RCARRE           4 POINTES
         Mesure 1:Nom Valeur          Mesure 2:Nom Valeur
                                      VSUBSTRAT    -2.5
         Mesure 3:Nom Valeur          Mesure 4:Nom Valeur

VALEUR:  Type      du          N.P.T.     N.T.B.   N.T.R.
         CA     20/01/83         68         63        53
             MIN                MAX                MOY
             4.096E+01        4.539E+01          4.324E+01
             SIG                SKE                KUP
             1.063E+00        5.893E-02         -6.903E-01
              Commentaires
```

MOTIF :	Nom	sous-types1	sous-types2	sous-types3
	RESISTANCE	N+	4 POINTES	
	dim.geo.1:Nom	Valeur	dim.geo.2:nom	Valeur
	LONGUEUR	200	LARGEUR	4
	dim.geo.3:Nom	Valeur		

IDRES :	Nom	Methode		
	RCARRE	VDP		
	Mesure 1:Nom	Valeur	Mesure 2:Nom	Valeur
			VSUBSTRAT	-2.5
	Mesure 3:Nom	Valeur	Mesure 4:Nom	Valeur

VALEUR:	Type	du	N.P.T.	N.T.B. N.T.R.
	CA	20/01/83	68	64 63
	MIN		MAX	MOY
	3.057E+01		3.413E+01	3.256E+01
	SIG		SKE	KUP
	9.742E-01		-9.270E-02	-1.351E+00
	Commentaires			

MOTIF :	Nom	sous-types1	sous-types2	sous-types3
	RESISTANCE	N+	4 POINTES	
	dim.geo.1:Nom	Valeur	dim.geo.2:nom	Valeur
	LONGUEUR	200	LARGEUR	4
	dim.geo.3:Nom	Valeur		

IDRES :	Nom	Methode		
	SURGRAVURE	VDP - R4P		
	Mesure 1:Nom	Valeur	Mesure 2:Nom	Valeur
	Mesure 3:Nom	Valeur	Mesure 4:Nom	Valeur

VALEUR:	Type	du	N.P.T.	N.T.B. N.T.R.
	CA	20/01/83	68	63 49
	MIN		MAX	MOY
	4.380E-01		5.382E-01	4.784E-01
	SIG		SKE	KUP
	6.560E-04		7.317E-01	-2.581E-01
	Commentaires			

MOTIF :	Nom	sous-types1	sous-types2	sous-types3
	RESISTANCE	N+	4 POINTES	
	dim.geo.1:Nom	Valeur	dim.geo.2:nom	Valeur
	LONGUEUR	200	LARGEUR	4
	dim.geo.3:Nom	Valeur		

IDRES :	Nom	Methode		
	RCARRE	4 POINTES		
	Mesure 1:Nom	Valeur	Mesure 2:Nom	Valeur

```
          Mesure 3:Nom Valeur          Mesure 4:Nom Valeur

VALEUR:   Type      du            N.P.T.     N.T.B.   N.T.R.
          CA     20/01/83          68          66         47
              MIN                 MAX                  MOY
              2.809E+00         4.738E+00          3.604E+00
              SIG                 SKE                  KUP
              4.887E-02         1.937E+00          1.541E+01
           Commentaires

MOTIF  :  Nom                sous-types1     sous-types2    sous-types3
          RESISTANCE         N+              4 POINTES
          dim.geo.1:Nom Valeur               dim.geo.2:nom Valeur
          LONGUEUR           200             LARGEUR          4
          dim.geo.3:Nom  Valeur

IDRES  :  Nom              Methode
          RCARRE           VDP
          Mesure 1:Nom Valeur          Mesure 2:Nom Valeur

          Mesure 3:Nom Valeur          Mesure 4:Nom Valeur

VALEUR:   Type      du            N.P.T.     N.T.B.   N.T.R.
          CA     20/01/83          68          67         50
              MIN                 MAX                  MOY
              3.145E+00         2.714E+01          4.981E+00
              SIG                 SKE                  KUP
              2.146E+01         3.059E+00          9.494E+00
           Commentaires

MOTIF  :  Nom                sous-types1     sous-types2    sous-types3
          RESISTANCE         N+              4 POINTES
          dim.geo.1:Nom Valeur               dim.geo.2:nom Valeur
          LONGUEUR           200             LARGEUR          4
          dim.geo.3:Nom  Valeur

IDRES  :  Nom              Methode
          SURGRAVURE       VDP - R4P
          Mesure 1:Nom Valeur          Mesure 2:Nom Valeur

          Mesure 3:Nom Valeur          Mesure 4:Nom Valeur

VALEUR:   Type      du            N.P.T.     N.T.B.   N.T.R.
          CA     20/01/83          68          65         50
              MIN                 MAX                  MOY
             -9.158E+02         1.982E+00         -6.028E+01
              SIG                 SKE                  KUP
              3.212E+04        -3.461E+00          1.174E+01
           Commentaires
```

```
MOTIF :  Nom              sous-types1     sous-types2    sous-types3
         RESISTANCE       N+              4 POINTES
         dim.geo.1:Nom Valeur             dim.geo.2:nom Valeur
         LONGUEUR         200             LARGEUR         4
         dim.geo.3:Nom  Valeur

IDRES :  Nom              Methode
         RCARRE           4 POINTES
         Mesure 1:Nom Valeur          Mesure 2:Nom Valeur

         Mesure 3:Nom Valeur          Mesure 4:Nom Valeur

VALEUR:  Type      du            N.P.T.     N.T.B.   N.T.R.
         CA     20/01/83          68          54         50
             MIN                 MAX                  MOY
             2.723E-02         3.503E-02        3.150E-02
             SIG                 SKE                  KUP
             3.742E-06         1.217E-01       -1.001E+00
          Commentaires

MOTIF :  Nom              sous-types1     sous-types2    sous-types3
         RESISTANCE       N+              4 POINTES
         dim.geo.1:Nom Valeur             dim.geo.2:nom Valeur
         LONGUEUR         200             LARGEUR         4
         dim.geo.3:Nom  Valeur

IDRES :  Nom              Methode
         RCARRE           VDP
         Mesure 1:Nom Valeur          Mesure 2:Nom Valeur

         Mesure 3:Nom Valeur          Mesure 4:Nom Valeur

VALEUR:  Type      du            N.P.T.     N.T.B.   N.T.R.
         CA     20/01/83          68          67         61
             MIN                 MAX                  MOY
             2.468E-02         3.122E-02        2.800E-02
             SIG                 SKE                  KUP
             3.037E-06        -1.804E-01       -9.493E-01
          Commentaires

MOTIF :  Nom              sous-types1     sous-types2    sous-types3
         RESISTANCE       N+              4 POINTES
         dim.geo.1:Nom Valeur             dim.geo.2:nom Valeur
         LONGUEUR         200             LARGEUR         4
         dim.geo.3:Nom  Valeur

IDRES :  Nom              Methode
         SURGRAVURE       VDP - R4P
         Mesure 1:Nom Valeur          Mesure 2:Nom Valeur
```

Mesure 3:Nom Valeur Mesure 4:Nom Valeur

VALEUR:	Type	du	N.P.T.	N.T.B.	N.T.R.
	CA	20/01/83	68	54	41
	MIN		MAX		MOY
	9.758E-02		3.898E-01		2.374E-01
	SIG		SKE		KUP
	5.311E-03		1.160E-01		-8.379E-01
	Commentaires				

Technologie	Provenance	Lots	Tranche
HMOS II LOCOS	DT TI	SIM	S4 T

MOTIF :	Nom	sous-types1	sous-types2	sous-types3
	TMOS	ENRICHI	N	
	dim.geo.1:Nom	Valeur	dim.geo.2:nom	Valeur
	LARGEUR	100	LONGUEUR	100
	dim.geo.3:Nom	Valeur		

IDRES :	Nom	Methode		
	VS	CLASSIQUE		
	Mesure 1:Nom	Valeur	Mesure 2:Nom	Valeur
	VSUBSTRAT	0	TEMPERATUR	22
	Mesure 3:Nom	Valeur	Mesure 4:Nom	Valeur

VALEUR:	Type	du	N.P.T.	N.T.B.	N.T.R.
	CA	25/01/83	52	46	37
	MIN		MAX		MOY
	8.183E-01		8.569E-01		8.328E-01
	SIG		SKE		KUP
	1.146E-04		7.419E-01		-4.422E-01
	Commentaires				

MOTIF :	Nom	sous-types1	sous-types2	sous-types3
	TMOS	ENRICHI	N	
	dim.geo.1:Nom	Valeur	dim.geo.2:nom	Valeur
	LARGEUR	100	LONGUEUR	100
	dim.geo.3:Nom	Valeur		

IDRES :	Nom	Methode		
	VS	CLASSIQUE		
	Mesure 1:Nom	Valeur	Mesure 2:Nom	Valeur
	VSUBSTRAT	0	TEMPERATUR	22
	Mesure 3:Nom	Valeur	Mesure 4:Nom	Valeur

VALEUR:	Type	du	N.P.T.	N.T.B.	N.T.R.
	CA	25/01/83	52	48	37
	MIN		MAX		MOY
	7.985E-01		8.624E-01		8.250E-01
	SIG		SKE		KUP
	2.814E-04		4.843E-01		-6.520E-01

Commentaires

MOTIF :	Nom	sous-types1	sous-types2	sous-types3
	TMOS	ENRICHI	N	
	dim.geo.1:Nom	Valeur	dim.geo.2:nom	Valeur
	LARGEUR	100	LONGUEUR	100
	dim.geo.3:Nom	Valeur		

IDRES :	Nom	Methode		
	VS	CLASSIQUE		
	Mesure 1:Nom	Valeur	Mesure 2:Nom	Valeur
	VSUBSTRAT	0	TEMPERATUR	22
	Mesure 3:Nom	Valeur	Mesure 4:Nom	Valeur

VALEUR:	Type	du	N.P.T.	N.T.B. N.T.R.
	CA	25/01/83	52	48 37
	MIN		MAX	MOY
	7.797E-01		8.450E-01	8.122E-01
	SIG		SKE	KUP
	2.281E-04		1.228E-01	-3.914E-01

Commentaires

MOTIF :	Nom	sous-types1	sous-types2	sous-types3
	TMOS	ENRICHI	N	
	dim.geo.1:Nom	Valeur	dim.geo.2:nom	Valeur
	LARGEUR	100	LONGUEUR	100
	dim.geo.3:Nom	Valeur		

IDRES :	Nom	Methode		
	VS	CLASSIQUE		
	Mesure 1:Nom	Valeur	Mesure 2:Nom	Valeur
	VSUBSTRAT	0	TEMPERATUR	22
	Mesure 3:Nom	Valeur	Mesure 4:Nom	Valeur

VALEUR:	Type	du	N.P.T.	N.T.B. N.T.R.
	CA	25/01/83	52	48 38
	MIN		MAX	MOY
	8.811E-01		9.204E-01	8.980E-01
	SIG		SKE	KUP
	9.395E-05		2.726E-01	-7.706E-01

Commentaires

MOTIF :	Nom	sous-types1	sous-types2	sous-types3
	TMOS	ENRICHI	N	
	dim.geo.1:Nom	Valeur	dim.geo.2:nom	Valeur
	LARGEUR	100	LONGUEUR	100
	dim.geo.3:Nom	Valeur		

IDRES : Nom Methode

```
        VS              CLASSIQUE
        Mesure 1:Nom Valeur          Mesure 2:Nom Valeur
        VSUBSTRAT    0               TEMPERATUR   22
        Mesure 3:Nom Valeur          Mesure 4:Nom Valeur

VALEUR: Type      du            N.P.T.     N.T.B.   N.T.R.
        CA     25/01/83          52          48         39
            MIN                 MAX                 MOY
            8.948E-01          9.375E-01          9.145E-01
            SIG                 SKE                 KUP
            1.063E-04          1.417E-01         -7.895E-01
         Commentaires

MOTIF :  Nom              sous-types1     sous-types2    sous-types3
         TMOS             ENRICHI         N
         dim.geo.1:Nom Valeur             dim.geo.2:nom Valeur
         LARGEUR          100             LONGUEUR       100
         dim.geo.3:Nom  Valeur

IDRES :  Nom           Methode
         MOBILITE      SUR 1 TMOS
         Mesure 1:Nom Valeur          Mesure 2:Nom Valeur

         Mesure 3:Nom Valeur          Mesure 4:Nom Valeur

VALEUR: Type      du            N.P.T.     N.T.B.   N.T.R.
        CA     25/01/83          52          46         39
            MIN                 MAX                 MOY
            5.648E+02          5.874E+02          5.769E+02
            SIG                 SKE                 KUP
            4.075E+01         -3.520E-01         -6.407E-01
         Commentaires

MOTIF :  Nom              sous-types1     sous-types2    sous-types3
         TMOS             ENRICHI         N
         dim.geo.1:Nom Valeur             dim.geo.2:nom Valeur
         LARGEUR          100             LONGUEUR       100
         dim.geo.3:Nom  Valeur

IDRES :  Nom           Methode
         DELTAL        DL ET DZ#0
         Mesure 1:Nom Valeur          Mesure 2:Nom Valeur

         Mesure 3:Nom Valeur          Mesure 4:Nom Valeur

VALEUR: Type      du            N.P.T.     N.T.B.   N.T.R.
        CA     25/01/83          52          46         37
            MIN                 MAX                 MOY
            3.410E-01          4.847E-01          4.262E-01
            SIG                 SKE                 KUP
            1.133E-03         -3.358E-01         -1.586E-01
```

Commentaires

```
MOTIF :  Nom              sous-types1     sous-types2    sous-types3
         TMOS             ENRICHI         N
         dim.geo.1:Nom Valeur             dim.geo.2:nom Valeur
         LARGEUR          100             LONGUEUR       100
         dim.geo.3:Nom  Valeur

IDRES :  Nom           Methode
         DELTAZ        DL ET DZ#0
         Mesure 1:Nom Valeur        Mesure 2:Nom Valeur

         Mesure 3:Nom Valeur        Mesure 4:Nom Valeur

VALEUR:  Type      du          N.P.T.     N.T.B.   N.T.R.
         CA     25/01/83        52         46        33
             MIN                MAX              MOY
             1.476E+00        1.610E+00        1.537E+00
             SIG                SKE              KUP
             9.404E-04        1.891E-01       -3.244E-01
          Commentaires


MOTIF :  Nom              sous-types1     sous-types2    sous-types3
         TMOS             ENRICHI         N
         dim.geo.1:Nom Valeur             dim.geo.2:nom Valeur
         LARGEUR          100             LONGUEUR       100
         dim.geo.3:Nom  Valeur

IDRES :  Nom           Methode
         MOBILITE      DL ET DZ#0
         Mesure 1:Nom Valeur        Mesure 2:Nom Valeur

         Mesure 3:Nom Valeur        Mesure 4:Nom Valeur

VALEUR:  Type      du          N.P.T.     N.T.B.   N.T.R.
         CA     25/01/83        52         46        39
             MIN                MAX              MOY
             5.659E+02        5.871E+02        5.786E+02
             SIG                SKE              KUP
             4.102E+01       -5.729E-01       -6.773E-01
          Commentaires


FIN traitement requete courante. Prendre la SUIVANTE

 ENTREZ VOTRE REQUETE S.V.P...

-->SELECT  TRANCHE LOT TELQUE TECHNOLOGIE "HMOS I" PROVENANCE "DI"
>>==> NOM-MOTIF "TMOS" NOM-RESULTATS "VS" ;
 Requete valide au niveau de l analyse SYNTAXIQUE
```

```
 La requete concerne une seule base : BASE1 .
  ANALYSE SEMANTIQUE TERMINEE...


 Ouverture de la BASE1...

 Recherche en BASE1    :  TRANCHE
 Recherche en BASE1    :  LOT
          Telque       :  TECHNOLOGIE       : HMOS I
          Telque       :  PROVENANCE        : DI
          Telque       :  NOM-MOTIF         : TMOS
          Telque       :  NOM-RESULTATS     : VS
Fermeture de la BASE1 .


**************************************************
* INTERROGATION DE LA BASE1-CARACTERISATION.R: 8*
**************************************************

 Technologie             Provenance
 HMOS I                    DI

MOTIF : nom                sous-types1   sous-types2   sous-types3
         TMOS             ENRICHI       N
        dim.geo.1:nom     valeur         dim.geo.2:nom     Valeur

        dim.geo.3:nom      valeur


IDRES : Nom                Methode
         VS               CLASSIQUE
        Mesure 1:Nom       Valeur     Mesure 2:Nom     Valeur
         VSUBSTRAT        0
        Mesure 3:Nom       Valeur     Mesure 4:Nom     Valeur


 H736E              18

FIN traitement requete courante. Prendre la SUIVANTE

 ENTREZ VOTRE REQUETE S.V.P...

-->FIN
FIN TRAITEMENT
```

ANNEXE II

Dépendance fonctionnelle

Un constituant B est dit fonctionnellement dépendant d'un constituant A dans un schéma de relation R(A,B,C,D), si et seulement si, à un instant donné à toute valeur $a \in A$ n'est associée qu'une et une seule valeur $b \in B$, nous noterons $A \rightarrow B$ la dépendance fonctionnelle de B par rapport à A.

Exemple

Soit le schéma de relation :

ENTETE (n° enregistrement, techno, Prov, lot, n° méth) il existe entre les constituants n°enregist, techno, prov, lot, n°meth des dépendances fonctionnelle, par exemple :

techno $\longrightarrow$ lot
(une technologie n'ayant qu'un seul lot à un instant donné)

Chaque dépendance fonctionnelle définit un certain nombre de propriétés réalisant certains types de dépendances

* Réflexivité : si $B \subseteq A$ alors $A \rightarrow B$
* Augmentation : si $C \subseteq D$ etsi $A \rightarrow B$ alors $AD \rightarrow BC$
* Transitivité : si $A \rightarrow B$ et $B \rightarrow C$ alors $A \longrightarrow C$
* Pseudo-transivité : si $A \rightarrow B$ et $BC \rightarrow B$ $AC \rightarrow D$
  AC ensemble des couples des valeurs
* Union : si $A \rightarrow B$ et $A \rightarrow C$ $\Rightarrow A \rightarrow BC$
* Décomposition : si $A \rightarrow BC$ alors $A \rightarrow B$ et $A \rightarrow C$

D'après ARMSTRONG (ARM 74) les trois dernières propriétés sont déduites des trois premières.

Soit $A \rightarrow B$ deux dépendances fonctionnelles, on dit que ces dépendances sont élémentaires dans T(attributs de relations), si A et $B \subset T$, $A \cap B = \emptyset$ si $\nexists C \subset A$ tel que $C \rightarrow B$ alors $A \rightarrow B$ est élémentaire.

Soit $A \rightarrow B$ dépendances fonctionnelles, on dit que ces dépendances sont directes dans R(A,B,C) et A,B et $C \subset T$ si $\nexists C \subset T$ tel que $A \rightarrow C$ et $C \rightarrow B$ alors $A \rightarrow B$ est directe.

Soit $A \rightarrow B$ deux dépendances fonctionnelles, on dit que ces dépendances sont cononiques si B est compose d'un attribut simple.

Formes normales des relations

* Première forme normale(1FN) : une relation R est en première forme normale si et seulement si, chacun des constituants qui ne font pas partie de l'identifiant est dépendant fonctionnellement de l'identifiant (ni élémentaire, ni directe).

* Deuxième forme normale (2FN) : une relation R est en deuxième forme normale si et seulement si, tous les constituants qui ne font pas partie de l'identifiant sont en dépendnance fonctionnelle élémentaire avec l'identifiant.

* Troisième forme normale (3FN) : une relation R est en troisième forme normale si et seulement si les constituants qui ne font pas partie de l'identifiant sont en dépendance fonctionnelle élémentaire et directe avec l'identifiant.

Remarque : d'une façon générale pour assurer qu'une relation R(A,B,C,D) est en 2FN ou est en 3FN, il suffit de s'assurer que B, C et D sont indépendanntes, c'est à dire qu'il n'existe pas entre elles de dépendance fonctionnelle.

Supposons en effet, qu'il existe $A \rightarrow B$ et $B \rightarrow C$ alors la dépendance fonctionnelle $A \rightarrow C$ n'est pas directe puisqu'elle est obtenue par transitivité $A \rightarrow B \rightarrow C$.

Des relations en 3FN sont aussi nécessairement en 1FN et 2FN.

ANNEXE III

BIBLIOGRAPHIE CHOISIE ET COMMENTEE

Parmi les quelques mille titres traitant de près ou de loin des bases de données, nous avons retenu quelques titres : c'est à dire que toutes les références listées ci-après présentent à notre avis un intérêt.

Pour guider l'utilisateur dans cette liste nous avons joint à chaque titre une notation (BH,BR,LR,OP,MO ou *) :

BH : Références concernant les bases de données hiérarchiques
BR : Références présentant les bases de données relationnelles
LR : Références qui définissent les langages de requêtes soit hiérachiques soit relationnels
OP : Références présentant une originalité ou montrant une façon d'optimiser les requêtes
MO : Références qui donnent une idée sur les méthodes de stockage et les types d'accès
* : Références des articles et des ouvrages fondamentaux pour la connaissance de bases de données.

BH ABRIAL J.R. et al "L'architecture du système SOCRATE"
Revue de l'AFET, Juin 1972

Une étude très technique de SOCRATE par son concepteur et l'équipe qui l'a développée. Claire et utile au spécialiste elle dégage bien l'intérêt de la structure "virtuelle" de SOCRATE.

* ADIBA M. "Les systèmes de bases de données relationnelles"
Informatique et gestion, n°125, pp 56-64, Juin 1981

Cette étude s'attache à trois problèmes fondamentaux dans les bases relationnelles : celui du langage et de l'architecture des systèmes, celui du stockage des données et leurs accès, enfin, celui de leur intégrité.

LR AHO AV. et al "The theory of Joins in relational data base"
ACM Trans. on DBS, Vol.4, n°3, Sept. 1979, pp 297-314

Une étude simple des différents cas de jointures entre les relations, et il définit la dépendance fonctionnelle et multivaluée, ainsi que la décomposition des relations pour avoir une exécution de la jointure.

LR ANTONACCI F. et al "AQL : a problem solving Query language for relational data bases"
IBM J. Res. dev., Vol.22, n°5, pp541-559, sept 1978.

Il définit un langage de requête pour une base de données relationnelle du type "AQL" et il montre les différents types de requêtes.

* ASTRAHAN H. et al "System R : Relational Approach to data base Management"
ACM Trans. on DBS, Vol 1, n°2, pp 97-137, juin 1976.

C'est un modèle combinatoire qui a été retenu dans le sous système relationnel de données (SRO) du système R. Le constat raisonnable de ses concepteurs est que la principale difficulté lors du traitement d'une requête est l'exécution de l'opérateur de Jointure.

BR AUER H. et al "RDBM, a relational data base machine"
LR Info Systems, Vol.6, n°2, pp 91-100, 1981

Présente une technique de construction d'une machine de base de données relationnelle, avec la méthode d'optimisation du tri, une démarche mise au point permet l'exécution de l'opérateur JOIN.

BR BANERJEE J. et al "Concepts and capabilities of a data base computer"
LR ACM Trans. on DBS, Vol.3, n°4, pp 347-384, Déc. 1981.

Une étude qui décrit la majorité des problèmes qui existent dans la conception d'une base de données et propose un modèle de conception de base de données.

BR BANERJEE J. et al "Data base transformation, Querry translation, and
BH performance analysis of a new data base computer in supporting hierarchical data base management"
Transaction on software engineering, Vol.SE 6, n°1, pp 91-119, 1980.

Cette étude définit la possibilité de transformation d'une base de données du type CODASYL au type relationnel et inversement. Elle définit aussi la requête adaptée à ce type de transformation ainsi que l'algorithme qui les réalise.

BR BENCY C., "Bases de données une méthode de conception
ROLLAND C. Techn. In centeur, Vol.38, n°20, 12-1981.

Il définit une méthode de conception de base de données en deux étapes conceptuelles (définit le chemin descriptif et la conception du modèle relationnel) et logique (que définit la fonction du schéma d'accès à l'information)

OP GING YAO J. "Optimization of query evaluation algorithms"
ACM Trans. on DBS, Vol.4, n°2, pp 133-15, Juin 1979.

Méthode d'optimisation de requête basée sur l'optimisation d'arborescence qui était construit par la déinition de l'algèbre relationnelle. Elle définit aussi le calcul du coût du schéma d'accès.

LR BLASGEN H.
MO ESWARAN K.

"Storage and access in relational data base"
IBM Syst. Jour., n°4, 1977.

Une méthode d'accès à l'information qui permet de construire plusieurs types d'algorithmes qui facilitent l'accès aux données.

LR BOYER R., MOORE S.

"A fast string searching algorithm"
Communications of the ACM, Vol.20, n°10, pp 762-771, Oct. 1981.

Il met au point un algorithme de construction d'une bibliothèque de recherche contenant des mots et des syntaxes puis il montre que cet algorithme est optimal.

BR BRIGGS F.A. et al
LR

"PUMPS architecture for pattern analysis and image data base management"
IEEE, Vol.18, n°8, pp 178-187, 1981.

Il définit l'architecture d'une machine de base de données aux plans matériels et logiciels, la machine est définie pour le traitement de l'image.

LR BUNEMAN P. et al

"An implementation technique for database Querry languages"
ACM Trans. on DBS, Vol.7, n°2, pp 164-186, Juin 1982.

Une définition d'un langage de requête présentée comme une adaptation de plusieurs langages de requêtes. Tous ces langages interrogent le modèle de base CODASYL.

BR CAMPBILL L. et al "Un sous système base de données pour le système SOL
LR atome"
Bulletin de liaison, INRIA, n°77, pp 24-28, 1982.

Un langage de définition permet la description de la structure physique et logique des données. Un langage de manipulation permet la gestion de ces données. Ce sont les programmes écrits en langage de requêtes qui sont pré-compilés pour les données (langage de programmation Pascal).

BH CANNING R. et al Rapport spécial sur les tendances en gestion de bases de données
L'informatique n°43, p4, Sept. 1973.

Une synthèse claire et complète des éléments d'une BD et une discussion des propositions du CODASYL.

MO CERI S., "Allocation of operation in distributed data base
PELAGATTI G. access"
IEEE Tran. on Computers, Vol.C32, n°2, pp 112-128, 1982.

Une méthode d'accès vers un système de gestion de base de données réparti avec une définiton du schéma d'accès par des termes mathématiques.

OP CHAN A., "On estimating the cost of accessing records in
NIAMIR B. blocked data base organization"
the computer journal, Vol.25, n°3, pp 368-374, 1982.

Une méthode d'estimation du coût du schéma d'accès d'exécution d'une requête en utilisant le calcul combinatoire.

LR CHANG N.S., FU K.S.

"Picture query languages for pictorial data base systems"
IEEE Computer, vol. 14, n°11, pp 23-33, 1981.

Une méthode de langage de requête de type "QBE" utilisée dans la manipulation de données stockées sur le modèle IMAIO (système de conception de base de données relationnelle utilisé dans le traitement d'image).

OP CHIN F.Y.

"Security in staical data base for queries with small counts"
ACM Transaction on data base systems, Vol. 3, n° 1, pp 92-104, 1978.

Un modèle de conception et de définition d'une structure de sécurité pour une base de données, méthode d'optimisation par la minimisation du nombre de comparaisons entre les données.

MO CHARLES W.

"Some systems shouldn't use chained file technique"
Data management, Vol. 11, n°9, Sept 1973.

Discussion des coûts en mémoire et en temps optimaux par l'emploi de chaînage, étude des techniques complexes d'adressage, comparées au stockage séquentiel sur barde, et recommandations sur le type de systèmes auquel elles sont surtout destinées.

BR CODD E.F.

"Normalized DB structures a brief tutorial"
ACM SIGFIDET Workshop, nov. 1971.

Article fondamental sur les bases de données relationnelles par un de leurs meilleurs spécialistes conception de base et comparaison avec les modèles hiérarchiques ou en réseau ; explication du concept de normalisation et introduction aux travaux les plus récents en la matière.

BR CODD E.F.

"Relational completeners of data base sublanguages"
Extrait de data base systems, courant computer serence symposia series, vol. 6, Prentice hall, p 65, 1972.

Compte rendu d'un exposé fait lors d'un séminaire sur les BD introduit le calcul et l'algèbre relationnels ainsi qu'un algorihtme de recherche dans les bases relationnelles.

* CODD E.F.

"Relational data base : A practical foundation for productivity"
Communications of the ACM, Vol.25, n°2, pp 109-117, 1982.

Une étude comparative sur le modèle relationnel et ses paramètres qui justifient l'utilisation de ce modèle, il donne à la fin les possibilités de recherche sur le domaine relationnel.

LR CORSON Y.
OP

"Aspects psychologiques liés à l'interrogation d'une base de données"
Rapport de recherche 126, INRIA, 1982.

Etude en deux parties : l'apport de la méthodologie expérimentale propre aux recherches psychologiques permet de quantifier la facilité d'utilisation des divers langages existants, la syntaxe servant de variable dépendante. D'autre part, des recherches plus ambitieuses conduisent à des études spécifiques des fonctions syntaxiques ou à des tentatives de modélisation des comportements. Les deux parties nécessitent de définir les processus cognitifs mis en jeu dans l'établissement des requêtes. Une seconde partie montre l'importance de la vision conceptuelle que peuvent avoir les opérateurs de l'organisation de la base de données.

BH CURTICE R. "Data independance in data base system"
Datamation, vol.21, n°4, pp 65, Avril 1975.

Une comparaison entre le modèle CODASYL, system 2000, IMS, TOAL et ADABAS sous l'angle de l'indépendance de données (le meilleur est le modèle CODASYL).

* DATE C.J. "An introduction to data base systems"
Addison Wesley Reading, Mass. USA, 1975.

Un ouvrage en 5 parties : architecture de SGBD, l'approche relationnelle, l'approche hiérarchique, l'approche en réseau, la sécurité et l'intégrité, doté d'exercices avec solutions et d'une abondante bibliographie.

OP DEMOLOMBE R. "Estimation of the number of tuples satisfying a query expressend in predicate calculus language"
ONERA CERT, Toulouse, France.

Une méthode d'estimation du nombre de tuples pour une requête du type "calcul de prédicat" utilisant la théorie des ensembles.

BR DIECKMANN M. "Three relational DBMS"
Datamation, vol.27, n° 9, pp 137-148, sept 1981.

Dans un aperçu historique sur les bases de données relationnelle ainsi que sur le matériel et le logiciel développés pour utiliser ces bases parmi les trois grands types : INGRES, ORACLE et SQLIOS.

* FERNANDEZ F. et al "MICROBE : un système de données relationnel réparties sur un réseau local de micro-ordinateurs". Rapport IMAG, Grenoble, 1980.

Présente une conception de la base de données relationnelle répartie (MICROBE) ; ainsi que ses caractéristiques les plus originales : contrôle de l'exécution algorithme de distribution de la requête basé sur la manipulation d'arborescences et la gestion de la mémoire relationnelle.

MO FERNANDEZ F. "Les B*-arbres dans MIMER"
SARJLIC M. Pojet MICROBE, IMAG Grenoble, 1981.

Etude de la méthode d'accès dite B*-arbres qui permet d'accéder soit à un tuple d'une relation dont on connaît la clé, soit à une relation toute entière selon un ordre différent de celui dans lequel la relation est physiquement stockée. En outre il donne le calcul du coût d'accès à un tuple d'une relation.

MO FERNANDEZ F. "MIcro MEmoire Relationnelle (MIMER)"
Rapport de recherche 233, IMAG Grenoble, 1981.

MIMER est une interface entre la vision relationnelle logique et le stockage physique d'une base de données relationnelle. L'accès aux relations est fait grâce à un langage de primitives utilisables depuis un langage de haut niveau, MIMER peut être utilisée dans des systèmes répartis et locaux.

BR FISHER P. "Data base design technique"
BH Computer communications , vol.4, n°6, pp 237-280, 1981.

Une étude théorique d'analyse, de modélisation et de définition d'une base de données.

BR GARDARIN G.
MELKANOFF M.

"Concurrency control principles in distributed and centralized data bases"
Rapport de recherche 133, INRIA FRANCE, 1982.

Discussion sur les problèmes du contrôle de concurrence. Une vue unifiée des diverses techniques de contrôle est présentée. Introduction des principes de base et dérivées de ces principes des preuves de corrections des algorithmes présentés. Ces algorithmes sont classés en trois groupes généraux basés sur le verrouillage de granules, l'ordonnancement à priori des accès des transactions.

OP GLORIEUX A.M.

"Optimisation dans un SGBD réparti suite à l'expérience SIRIUS DELTA"
Rapport interne DEX I 007, 08/1981.

Une reflexion sur l'optimisation du temps de traitement d'une requête dans un SGBD réparti, l'optimisation d'une requête dépend du choix des sous requêtes locales, et dépend aussi de ce que l'on veut optimiser (transfert, coût, temps de réponse..).

LR HAEDER T.

"Implementing a generalized access path structure for a relational DBS"
ACM Trans. on DBS, vol.3, n°3, pp 285-298, Sept 1978.

Un schéma d'accès généralisé qui permet d'accéder rapidement à l'information en utilisant un index de lien au début de chaque tuple.

OP HALL P.A.

"Optimization of single expressions in a relational data base system"
IBM J. Res. Deve., pp 244-257, May 1976.

Une méthode d'optimisation de requête de type algèbre relationnelle par l'utilisation d'une transformation alternative après la définition de l'expression logique de la structure de requête.

LR HARDGRAVE W.T. "Ambiguity in processing boolean queries on TDMS tres structures. A study of four different philosophies" Transaction on software engineering, vol.SE6, n°4, pp 357-372, 1980.

Cette étude définit la philosophie du processus de requête, elle traite cette requête après la construction de son arborescence par la structure sémantique qui la définit et après l'avoir décomposée.

BR HURSON A. "An associative backend machine for data base management"
IEEE, 1981, Vol.18, n°8, 225/256.

Un modèle de base de données technologique basée sur l'opération des tests sur circuits intégrés.

* IRIA ou INRIA Bulletin de liaison
Institut de recherche d'informatique et d'Automatique, 1970-1983.

Différents numéros sur des sujets très divers liés aux bases de données. Une mine pour le spécialiste, d'excellents rapports sur les systèmes existants, des études théoriques, des comptes rendus de débats. une source précieuse sur les grands projets de l'administration française.

BH JONES G.L. "CODASYL FORTRAN data base Facility"
Inform. System, Vol.4, n°3, 161-199, 1981.

Il parle dans le début, de la conception de base de données et il définit le principal chemin d'accès. Après il définit le langage de manipulation et les différents types de commandes. Tout cela pour arriver à définir le modèle CODASYL FORTRAN.

* JOUFFROY G.
LETANG Ch.

"Les fichiers et choix de l'organisation des données informatiques"
Dunod techniques, Paris, 1974.

Petit ouvrage sur les structures de fichiers, un chapitre est consacré aux BD. A lire comme introduction aux structures physiques (il y a quelques algorithmes traditionnels).

BR. KIYOKI Y. et al
LR

"Design and evaluation of a relational data base machine employing advanced data structures and algorithms"
IEEE Computer Architecture, vol.18, n)4, pp 407-423, 05/1981.

Etude présentant l'architecture d'une machine de base de données relationnelle du point de vue de la modélisation de la relation dérivée (relation après l'exécution).

* KNUTH D.

"The art of computer programming"
Volume 1 Fundamental algorithms 1968
Volume 3 Surting and searching 1973
Addison Wesley Reading, Mass. USA.

Le volume 1 décrit les structures de données de manière exhaustive, le volume 3 est consacré aux tris et aux recherches.

MO LITWIN W.

"Hachage virtuel une nouvelle technique d'adressage de mémoires"
Thèse docteur d'Etat, Paris, 6 mars 1979.

Méthode qui propose une nouvelle technique d'adressage qui range et trouve les enregistrements d'après leurs clés primaires. Le principe est de trouver les insertions et les suppression qui peuvent modifier la fonction de hachage.

* LITWIN W. et al "SIRIUS systems for distributed data management" Rapport GAL-I-052, INRIA France, Sept. 1982.

Une vue générale du projet SIRIUS DELTA pour une multibase de données, en passant par les définitions de l'action locale, la synchronisation et la modélisation d'une interface relationnelle et propositions de quelques définitions d'optimisation.

LR LORIE R., NILSSON J. "An access specification language for a relational data base system" IBM J. Res. Devel., Vol; 23, n°3, pp 286,298, Mai 1979.

Il construit une méthode efficace pour interroger la base de données par un accès rapide vers les données stockées sous modèle R, toutes les définitions sont contenues dans le système R.

LR LOZINKI E. "Construction of relations in relational data base"
OP ACM Trans. on DBS, vol.5, n°2, pp 208-224, 1980.

Cette étude décrit plusieurs algorithmes qui permettent d'exécuter les opérateurs algébriques (en particulier la jointure) d'une façon optimale.

MO MAIER D., SALVETER C. "Hysterical B-Trees" Inf. Processing letters, vol.12, n°4, pp 199-202, August 1981.

B-arbre une méthode d'accès avec un algorithme orienté vers l'indexation et la structuration des données.

* MARTIN J. "Computer data base organisation"
Prentice Hall, Englewood Cliffs, New Jersey, 1975.

Il nous donne une synthèse très fouillée des structures physiques de stockage avec une profusion d'illustrations fort bien conçues. Trois parties, prologue, organisation logique, (modèle IMS, modèle CODASYL, et bases relationnelles) et organisation physique.

* McGEE M. "Data base technology"
IBM J.Res. etDeve., vol.25, n°5, pp 505-519, 1981.

Etude qui présente l'aspect théorique des différents types de langages de manipulation.

MO MERRETT T., OTOO E. "Distribution models of relation"
IEEE Communication 1406, pp 418-425, 8/1979.

Etude d'une méthode permettant la distribution des tuples dans une relation pour arriver à avoir le stockage dans différentes zones de mémoire et application des opérateurs relationnels à ce type de mémorisation.

LR MICHARD A. "A new data base query language for non-profession users : design principles and ergonomie evaluation"
Rapport de recherche 127, INRIA FRANCE, 1982.

Un nouveau langage d'interrogation de base de données conçu pour améliorer la facilité d'apprentissage et d'utilisation par des utilisateurs occasionnels. Son principal intérêt est d'éviter l'usage explicite des opérateurs booléens lors des manipulations ensemblistes.

LR NGUYEN GIA TOAN "Distributed execution of queries on a local network
OP data base system"
Projet SIRIUS, Rapport IMAG GRENOBLE, oct. 1980.

Une vue générale du système MICROBE, il définit la requête d'exécution construite sur l'algèbre relationnelle, il définit aussi quelques stratégies d'optimisation statique (estimation la taille par la cardinalité) et dynamique (par la localisation).

LR NIJSSEN G.M. "Common data base languages"
ACM Data base, vol.4, n°4, 1972.

Courte synthèse sur les modèles DDL et DML et leurs relations avec les langages de programmation classiques.

LR PATRICIA P. et al "An authorization mechanism for a relational data
OP base system"
ACM Trans. on DBS, vol.1, n°3, Sept. 1976.

Vers une stratégie qui permet l'exécution de requêtes de type SEQUEL. En utilisant quelques définitions sémantiques qui donnent une exécution optimale;

OP RICHARD Ph. "On the evaluation of the size of the answer of a relational query"
Rapport de recherche 51, INRIA FRANCE, dec. 1980.

Présente un modèle probabiliste pour estimer la taille des relations construites par une requête en langage algébrique.

OP ROSENKRANTE D. "Processing conjunctive predicates and queries"
HUNT H.B. IEEE, ch1534, pp 64-73, 7/1980.

Il définit un algorithme qui minimise le nombre de conjonctions dans une requête posée et donné les définitions qu'on utilise pour réaliser cet algorithme.

* SABRE

Publication SABRE projet SIRIUS
INRIA BTR-I-005 1980.
Projet SABRE (Système d'Accès à des Bases de données RElationnelles).

L'objectif est la conception et la réalisation d'une machine bases de données multiprocesseurs utilisant le modèle relationnel et permettant de réduire les temps de réponse d'un facteur supérieur à 10 par rapport aux systèmes relationnels existants, il résume tous les articles faits sur ce projet pendant l'année 80, nouvelle classe de filtres, mémoires et distribution d'algorithmes.

* SABRE

Publication SABRE projet SIRIUS
INRIA 1981
Projet SABRE

Suite aux publications SABRE de 1980 regroupe les articles écrits par l'équipe SABRE.

MO SEO K. et al

"A look-ahead data staging architecture for relational data base machines"
IEEE Computer architecture, vol.18, n°4, pp 389-406, 05/1981.

Etude qui présente l'architecture d'une machine de base de données relationnelle du point de vue de la mémoire hiérarchique.

MO SHUN LIN C. et al

"The design of a rotating associative memory for relational data base applications"
ACM Trans. on DBS, vol. 1, n°1, pp 53-65, Mars 1976.

Une méthode qui définit une façon de mémoriser les données et les différentes stratégies qui permettent la réalisation de cette mémorisation.

OP SILVERMAN H.
YUE P.C.

"Response time characterization of an information retrieval system"
IBM J. Res. et dev., pp 394-403, Sept 1973.

Une méthode pour évaluer le temps de réponse dans le système distribué qui n'est pas nécessairement une base de données. Il définit plusieurs méthodes d'analyse de données, de file d'attente et de dérivations de données.

* SIRIUS

Atelier SIRIUS
Congrès AFCET 1981, Rapport CAL-I-049, INRIA, 1981

Objectif du projet SIRIUS, état et orientation des recherches, présentation des équipes et bibliographie du projet pilote SIRIUS.

* SIRIUS

Liste des publications de la bibliothèque SIRIUS
INRIA, FRANCE, 1981

Une liste des publications internes et externes de la bibliothèque SIRIUS les plus récemment parues.

LR SMITH J. et al
OP

"Optimizing the performance of a relational algebra data base interface"
Communications of the ACM, vol.18, n°10, pp 563-579, oct. 1975.

Une méthode qui propose d'écrire un programme permettant de réaliser le traitement de l'opérateur relationnel, pour minimiser le temps de réponse et la taille des résultats intermédiaires.

BR STONEBRAKER M. "Retropestion on a data base system"
ACM Trans. on DBS, vol.5, n°2, juin 1980, pp 225,240.

Il décrit un SGBD qui offre la possibilité de définir l'implantation d'une base de données relationnelle. Il montre comment est choisi le meilleur schéma d'accès pour arriver à optimiser l'exécution d'une requête en évaluant le coût à chaque pas.

BR. SUDHIR K. et al "WCRC : An maxi spare machine architecture for data base management"
IEEE Computer architecture, vol.18, n°4, pp373-388, 05/1981.

Etude qui présente l'architecture pour arriver à la meilleure conception d'une machine relationnelle. En plus il calcule le temps nécessaire pour exécuter une requête.

BR TAYLOR O. et al "Redundancy in data structures : improving software
MO fault tolerance"
IEEE Trans. on software engineering, vol.SE6, n°6, nov. 1980.

Cette étude sert à définir la structure de données en utilisant une méthode d'accès basée sur la méthode B-arbre après la définition d'un enregistrement comme un point d'entrée.

* TOMAS J.L. "Bases de données conception réalisation et implantation sur mini-ordinateurs"
MASSON 1981.

Le but de cette étude est de concevoir et de réaliser sur un mini-ordinateur, un système automatique construit sur une base de données pour un service d'assitance technique. Ce système doit d'abord permettre de faire la suite très détaillée des appels refus, puis deviendra au cours de son utilisation un outil efficace pour résoudre les problèmes soumis.

OP TO-YAT CHEUNG "Estimating block accesses and nomber of records in file management"
Communications of the ACM, vol.25, n°7, pp 484-487, 1982.

Une méthode d'estimation de la taille nécessaire pour stocker l'information intermédiaire (information utilisée pour trouver la réponse d'une requête)

LR TREILLE L., SERGEANT G. "SER système d'exécution répartie"
Rappport INRIA SCH-I-067? Pojet SIRIUS, Aout 1979.

Il décrit les mécanismes proposés pour la mise en oeuvre d'un plan d'exécution réparti fourni par une application de niveau supérieur, il définit une existence d'un médium de communication entre les différentes machines intéressées.

LR WRONG E., OP YOUSSEF K. "Décomposition. A strategy for query processing"
ACM Trans. on DBS, vol.1, n°3,pp 233-241, sept 1976.

Cette étude définit une stratégie de composition de requêtes pour le système INGRES. Cette stratégie est basée sur la décomposition de la requête en requêtes monovariables, elle définit l'algorithme de décomposition et l'estimation du coût de traitement.

MO XAMAGUCHI, KUNT "Logical fram ework of a picture data base computer"
IEEE 1981, vol.18, n°8, pp 284-322.

Un type de logiciel qui contient l'aspect mémorisation avec différentes stratégies pour avoir un stockage d'information.

LR YAO S.B. "An attribut model for data base access cost analysis"
ACM Trans. on DBS, vol.2, n°1, pp 45-67, mars 1977.

Une méthode basée sur la construction d'un schéma d'accès vers les éléments de base utilisant l'indice séquentiel et le type B-arbre.

OP YU C.T. et al "On the estimation of the number of desired records with respect to a given query"
ACM Trans. on DBS, vol.3, n°1, pp 41-56, Mars 1978.

Une méthode d'optimisation du traitement de requête d'après l'estimation du nombre d'attributs à chaque niveau de schéma d'accès.

LR ZLOOF M.M. "QUERY-BY-EXAMPLE : a data base language"
IBM system Journal, n°4, pp 324-343, 1977.

QBE est un langage graphique. Son originalité réside dans l'expression graphique des requêtes : l'utilisateur emploie un terminal à écran sur lequel il fait afficher des tableaux représentant des entités dans la relation. Dans ces tableaux on définit la requête.

Note bibliographique - compilateur

Les premiers compilateurs pour langages évolués ont été conçus dans la période allant de 1956-1958. Parmi ceux-ci le plus important a été le compilateur FORTRAN pour l'ordinateur IBM 704. Un grand nombre de publications anciennes sur les techniques de compilation ont été rassemblées par Rosen (1967) (Progreamming systems and languages Rosen S. Mc Graw Hill 1967), sur les techniques pour les réaliser (IRO 61, EVA 62), les outils d'aide à leur production (FEL 66, FG 68) (appelés selon les auteurs, "systèmes d'écriture de traduction", "générateurs de compilateurs", "compilateurs de compilateurs"), ou sur les méthodes pour les prouver (MAC 62, MAC 67, LON 71).

Très tôt dans ce domaine, il est apparu qu'il était essentiel de se donner des outils de description formelle des langages de programmation (MAC 63, NIV 66, KNU 68, BUR 69, SCO 71). Ces nombreux travaux ont en commun une description bien formalisée de la syntaxe (MAC 63, LAU 68). Parmi les méthodes de description des contraintes contextuelles, les plus connues sont probablement les doubles grammaires (VAN 69), les grammaires affixes (KOS 71) et les attributs de Knuth (KNU 68, LOR 74) appelés généralement attributs sémantiques. On trouvera une bonne bibliographie dans (AHO 79, CUN 80).

Un grand nombre d'algorithmes différents destinés à l'analyse syntaxiques et sémantiques des expressions arithmétiques ont été produits en France notamment GAUDEL "Génération de compilateurs basée sur une sémantique formelle" (GAU 80), BARRE "système d'apprentissage des techniques élémentaires de compilations" (BAR 79), (APP 79), MAUGER "Techniques de compilation sur mini-ordinateurs" (MAU 76), DESCHAMP "Production de compilateurs à partir d'une description sémantiques" (DES 80).

(AHO 79) A.V. AHO, J.D. ULLMANN
"Principles of compiler design"
Addison Wesley, 1979.

(BAR 79) J. BARRE
"Système d'apprentissage des techniques élémentaires de compilation"
Thèse 3ème cycle, Rennes, France, 1979.

(BUR 69) R.M. BUR STALL, P.J. LANDIN
"Programs and their proofs an algebric approach"
Edimburgh University press, 1969.

(CUN 80) P.Y. CUNIN, M. GRIFFITHS, J. VOIRON
"Comprendre la compilation"
Springer verlage, 1980.

(DES 80) Ph. DESCHAMP
"Production de compilation à partir d'une description sémantique des langages de programmation, le système PERLUETTE"
Thèse Docteur Ingénieur, INP Lorraine, 1980.

(EVA 62) A. EVANS
"An algol 60 compiler"
ACM National conference, Denver 1962

(GAL 77) H. GALLAIRE
"Techniques de compilation Méthode d'analyses syntaxiques"
Cepadues edition,Toulouse, 1977.

(GAU 80) M.C. GAUDEL
"Généralisation et preuve de compilateurs basées sur une sémantique formelle des langages de programmation).
Thèse d'Etat INPL Lorraine, 1980.

(IRO 61) E. IRONS
"A syntax directed compiler for algol 60"
CACM 4, n°1, 1961.

(KOS 71) C.H.A KOSTER
"Affix grammars algol 68 implementation"
North Holland, 1971

(KNU 68) D. KNUTH
"Semantics of context free languages"
Maths. system theory, 2, n°2, 1968.

(LAU 68) P. LAUER, P. LUCAS, H. STIGLEITNER
"Method and notation for formal definition of programming languages"
Technical report (TR 25.087) IBM Laboratory Vienna, 1968.

(LON 71) R.L. LONDON
"Correctness of two compilers of LISP subsets"
A.I. memo 151, Standford University, 1971.

(LOR 74) B. LORHO
"De la définition à la traduction des langages de programmation"
Thèse d'Etat, Toulouse, 1974.

(Mac 62) J. Mac CARTHY
"Towards a mathematical science of computation"
Proceeedings IFIP congress 1962

(Mac 63) J. Mac CARTHY
"Computer programming and formal systems"
Amsterdam, 1963.

(Mac 67) J. Mac CARTHY, J.A. PAINTER
"Correctness of a compiler for arithmetic expressions"
Proceedings of a symposium in applied Mathematics, Mathematical aspects of science, n°19, 1967.

(MAU 76) C. MAUGER
"Techniques de compilation sur mini-ordinateur"
Thèse Docteur Ingenieur, Lyon 1976.

(NIV 66) M. NIVAT, N. NOLIN
"Contribution to definition of Algol semantics"
Baden , 1966

(SCO 71) D. SCOTT, C. STRACHEY
"Towards a mathematical semantics for computer languages"
Porgamming reseach group monograph, Oxford, 1971.

(VAN 69) A. VAN WIJNGARDEN, B.J. MAILLOUX, J.E. PECK
"Report on the algorithmic language algol 68"
Mathematische Centrum Amsterdam, 1969.

AUTORISATION de SOUTENANCE

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

VU le rapport de présentation de

. Madame G. SAUCIER, Professeur

Monsieur HAKIM Mohamed Nazir

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

Fait à Grenoble, le 17 août 1983

Le Président de l'I.N.P.-G.,

RESUME

Ce travail a contribué à étudier et réaliser une interface relationnelle pour un système de bases de données hiérarchiques multiples.

Nous présentons une telle interface (IMREL) permettant la transformation des structures de base de données hiérarchique (IMAGE) afin de la consulter de manière relationnelle. Ce modèle inclus la possibilité de définir un langage de manipulation de données.

Nous proposons une stratégie d'optimisation de requête par la recherche du chemin de coût minimal.

Enfin, nous terminons par la réalisation d'un compilateur de requête (CORREL) permettant de tester la faisabilité et la validité de nos propositions.

MOTS CLES

- base de données hiérarchique (IMAGE) ;
- base de données relationnelle,
- interface hiérarchique relationelle ;
- langage de requête ;
- optimisation de requête ;
- compilateur de requête.