URANUS : une approche relationnelle à la coopération de bases de données

Gia Toan Nguyen

THESE

présentée à

Université Scientifique et Médicale de Grenoble

pour obtenir le grade de
DOCTEUR DE 3ème CYCLE

par

NGUYEN GIA TOAN

URANUS

UNE APPROCHE RELATIONNELLE A LA COOPERATION DE BASES DE DONNEES.

Thèse soutenue le 15 décembre 1977 devant la Commission d'Examen :

Président : L. BOLLIET

Examinateurs : C. DELOBEL
H. GALLAIRE
S. KRAKOWIAK

UNIVERSITE SCIENTIFIQUE
ET MEDICALE DE GRENOBLE

Monsieur Gabriel CAU : Président
Monsieur Pierre JULLIEN : Vice Président

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

PROFESSEURS TITULAIRES

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

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

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

PROFESSEURS ASSOCIES

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

PROFESSEURS SANS CHAIRE

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

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

MAITRES DE CONFERENCES ET MAITRES DE CONFERENCES AGREGES

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

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

MAITRES DE CONFERENCES ASSOCIES

MM.	DEVINE Roderick	Spectro physique
	HODGES Christopher	Transition de phases

Fait à SAINT MARTIN D'HERES, NOVEMBRE 1976.

Je tiens à remercier Monsieur Louis BOLLIET, Directeur du Département Informatique de l'Institut Universitaire de Technologie B de Grenoble, de m'avoir fait l'honneur de présider le jury de cette thèse, après avoir dirigé mon Diplôme d'Etudes Approfondies. Ceci m'a permis d'être initié à des domaines de pointe de la recherche industrielle et universitaire.

Je dois exprimer ma très grande reconnaissance à Monsieur Claude DELOBEL qui, grâce à ses encouragements, ses conseils et la confiance qu'il m'a accordée, a permis en maintes occasions à ce travail de se poursuivre, puis d'aboutir et de se prolonger. Je lui en suis entièrement redevable. Qu'il trouve ici en des mots très ternes le témoignage de ma profonde gratitude.

Je remercie Monsieur H. GALLAIRE, Directeur du Département Informatique du Centre d'Etudes et de Recherches de Toulouse,
et Monsieur S. KRAKOWIAK, Directeur du Laboratoire d'Informatique de l'Université Scientifique et Médicale de Grenoble pour leurs critiques et conseils concernant le manuscrit de ce travail et leur participation au jury.

Les réalisations décrites doivent beaucoup à Dominique PORTAL, Chef du Service Informatique de Gestion de l'U.S.M.G., dont la collaboration dans la phase initiale du projet fut très fructueuse, ainsi qu'aux membres de l'équipe POLYPHEME, en particulier Andrée STIERS, Michel ADIBA et Jean-Yves CALECA.

Marie-José DOREL a assuré avec beaucoup de soin, de patience et de gentillesse la dactylographie d'un manuscrit souvent illisible.

Enfin, l'équipe de reprographie de Monsieur IGLESIAS en a assuré le tirage avec compétence.

Ce travail a été réalisé dans le cadre du contrat IRIA-Sesori 76027 :
"Adaptabilité d'un Système de Gestion de Base de Données à l'Evolution des Applications".

SOMMAIRE

CHAPITRE 1

INTRODUCTION

1.1. OBJECTIFS

Les systèmes d'informations sont communément conçus comme des outils de modélisation de situations réelles, perçues et manipulées par des utilisateurs administratifs ou scientifiques.

La tâche des analystes informaticiens appelés à collaborer avec ces derniers est souvent ardue, car la perception conceptuelle de situations complexes - la gestion du personnel dans une entreprise par exemple - nécessite une connaissance exhaustive du cheminement de l'information et des règles qui régissent son existence.

La définition d'un modèle et sa mise en oeuvre concrète constituent deux étapes fondamentales du processus d'automatisation et la démarche effectuée alors s'appuie généralement sur ces deux principes :

1 - les modèles, conçus dans des buts précis, doivent répondre aux besoins des usagers en toute circonstance qu'ils auront auparavant définie

2 - l'ensemble des informations régissant une situation réelle, objet du modèle, doit être défini complètement, de façon à éliminer toute introduction d'éléments ultérieurs imprévus.

Cette axiomatique de base est conforme avec une conception "utilitaire" des moyens informatiques.

Ils sont dans ce cas des outils puissants par la quantité d'information qu'ils permettent de traiter et par l'aide qu'ils peuvent apporter aux services qui les utilisent.

Ils doivent alors être parfaitement conformes aux règles qui y sont en vigueur : les modèles sont de ce fait figés dans un ensemble de contraintes d'utilisation répondant à des besoins entièrement spécifiés, dont la modification nécessite souvent une adaptation d'importance variable mais rarement indolore pour les usagers.

Nous pensons qu'à l'heure actuelle, ceci peut être partiellement évité et qu'il est possible d'étendre les services rendus par les systèmes d'information - dans une optique d'aide à la décision par exemple [31] - grâce à une conception plus dynamique de ceux-ci , c'est-à-dire à la possibilité de les adapter à des conditions réelles variant avec le temps.

Les principes de base précédents doivent pour ce faire être complétés par celui-ci :

- le modèle doit répondre à *l'évolution* des besoins définis par les usagers, et doit être capable de prendre en compte certaines interactions extérieures, non déterminées auparavant, mais susceptibles d'advenir et de modifier le comportement qu'on en attend.

L'application de ce principe est malaisée dans le cadre des systèmes actuels car ils sont bâtis comme des outils de *traitement* et il est souvent impossible de les transformer, ou de prendre en considération de nouvelles classes d'informations ou de contraintes d'utilisation à un coût "raisonnable".

En effet, les besoins constants d'une modélisation toujours plus fidèle des faits réels ont jusqu'à présent conduit à une croissance des possibilités de *représentation* de l'information dans les systèmes informatiques. Chronologiquement, ce furent les apparitions de fichiers séquentiels puis à accès direct, par clé d'enregistrement, par indexation à niveaux multiples, et plus récemment celle des sytèmes de gestion de bases de données où l'on a introduit la notion de *sous-ensembles logiques* - set CODASYL, inverses et anneaux SOCRATE par exemple [3].

La contrepartie de cette sophistication croissante des structures de données fut, et reste encore dans une large mesure aujourd'hui, une complexité grandissante de définition, de représentation et de manipulation de l'information, avec pour corrolaire une "étanchéité" ou incompatibilité accrue des systèmes entre eux.

Des palliatifs ont été mis au point pour faciliter le travail quotidien d'exploitation, par exemple par le développement des systèmes "presse-bouton" ou à "questions précompilées".

A une rigidité croissante des moyens informatiques, due à leurs difficultés de mise en oeuvre, on a donc répondu par un appauvrissement de leurs possibilités d'utilisation courante.

Cette solution contre-nature, coupant de plus l'utilisateur final des techniciens assurant l'état de marche des bases de données, dénote probablement dans son paradoxe un point extrême dans l'évolution du traitement de l'information.

La nécessité apparemment contradictoire de simplifier et de banaliser l'utilisation des systèmes d'informations représentant des situations de plus en plus complexes a suscité la recherche de solutions nouvelles.

Celles-ci reprennent en général une évolution déjà sensible dans les développements les plus récents, en systématisant aux niveaux logiques de modélisation et de conceptualisation des faits la dichotomie entre définition et représentation des informations. Cela présente l'avantage décisif d'autoriser l'introduction dans les modèles de la *sémantique* qui leur est attachée. Ceci était limité auparavant à l'interprétation de la définition structurelle des données et à la logique des programmes de traitement, toutes choses impossibles ou fastidieuses à modifier.

Cette distinction entre niveaux conceptuel, de modélisation et de représentation [9] a permis le développement des *modèles relationnels* de bases de données [4][32]. Ils se distinguent par le compromis réalisé entre les exigences contradictoires citées plus haut et le maniement facilité de la sémantique de l'information.

De plus, leur généralité, qui est due pour l'essentiel aux fondements mathématiques sur lesquels ils s'appuient - le calcul des prédicats du premier ordre et le calcul des relations [1] - les rend aptes à une approche systématique des problèmes de coopération de données hétérogènes. Dans le domaine des moyens de traitement, ils ont permis de concrétiser une évolution très sensible vers des langages simples mais très puissants, répondant ainsi à une simplification frustrante des possibilités d'utilisation instituée par les systèmes à "question-réponse".

Ainsi, parallèlement aux notions de relations et de prédicats sont apparus des langages fondés sur l'algèbre relationnelle - QUEL - et le calcul des prédicats - ALPHA [14], SYNTEX [41] -. Leur caracté-

ristique fondamentale est de n'être pas algorithmiques, c'est-à-dire *non-procéduraux* - SQUARE, SEQUEL [15]. Ils sont de ce fait beaucoup plus aisés à apprendre et à manier, tout en conservant en général une puissance équivalente aux langages d'interrogation algorithmiques. Ils permettent de rendre ainsi aux systèmes d'informations toute leur versatilité, en évitant une dégénérescence dangereuse parce que simpliste des possibilités d'exploitation de systèmes très évolués de gestion de données.

Prenant pour point de départ ces considérations, nous entendons permettre aux utilisateurs de bases de données actuelles de les adapter, ainsi que leurs applications, de façon à prendre en compte :

1 - de nouvelles règles d'utilisation des informations,
2 - de nouvelles données venant perturber les règles précédemment admises,
3 - et ceci en conservant au maximum les outils déjà mis en oeuvre.

Cette démarche sera illustrée par la réalisation d'un système relationnel de coopération de bases de données Socrate : URANUS.

Au niveau méthodologique, elle sera scindée en deux volets complémentaires :

1 - *intégration de nouveaux éléments*, extérieurs aux systèmes existants, ou adaptabilité structurelle ;
2 - *évolution des contraintes de traitement des informations*, ou adaptabilité logique (Fig. 1.1).

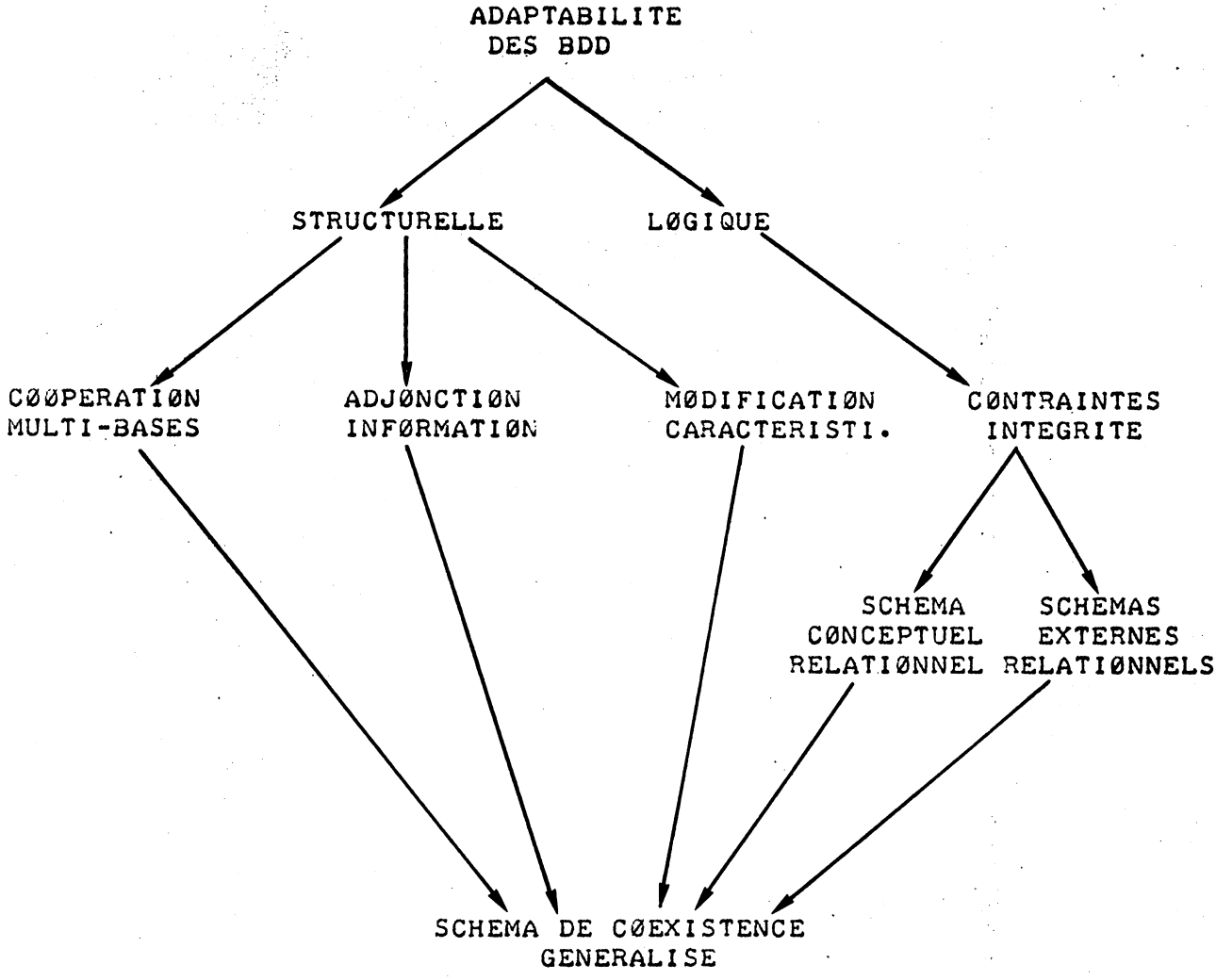

ADAPTABILITE
DES BDD
STRUCTURELLE
LØGIQUE
CØØPERATIØN
MULTI-BASES
ADJØNCTIØN
INFØRMATIØN
MØDIFICATIØN
CARACTERISTI.
CØNTRAINTES
INTEGRITE
SCHEMA
CØNCEPTUEL
RELATIØNNEL
SCHEMAS
EXTERNES
RELATIØNNELS
SCHEMA DE CØEXISTENCE
GENERALISE

Fig. 1.1

1.2. METHODE

L'*adaptation structurelle* nous permettra de définir les conditions et les moyens d'introduction de nouvelles informations dans un ensemble donné et par là de nous délivrer des aléas d'une définition temporairement exhaustive de celles-ci.

L'*adaptation logique* permettra de définir les règles d'utilisation résultant du changement de nomenclature précédent, et aussi de mettre en oeuvre des contraintes d'intégrité nouvelles sur des modèles existants, sans en perturber les caractéristiques de base.

La conjonction de ces deux aspects nous permettra enfin de réaliser des systèmes d'informations *évolutifs*, c'est-à-dire capables d'être à tout instant le reflet exact de situations changeantes, sur lesquels on pourra définir de nouvelles normes d'exploitation :

1 - informations et liens logiques non pris en compte précédemment,
2 - changement des règles de cohérence définies sur les collections de données.

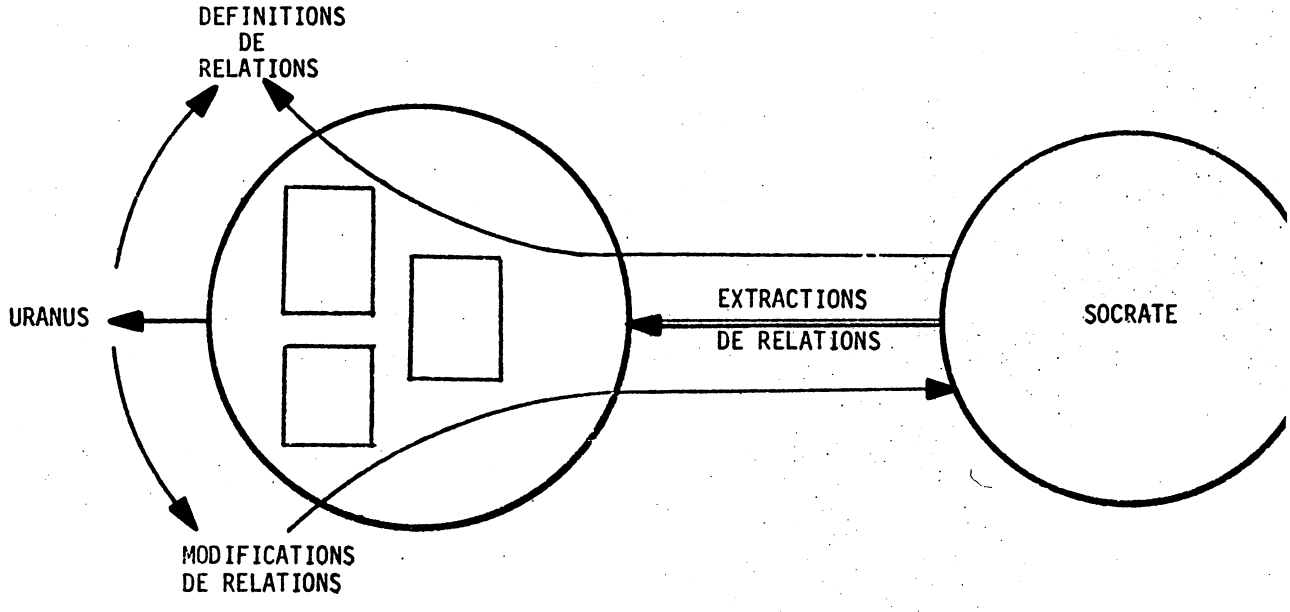


Fig. 1.2

L'adaptation structurelle sera scindée en 3 étapes complémentaires :

1 - définition de relations n-aires sur une base de données, ici Socrate, et manipulation relationnelle des données ainsi décrites : extraction, modifications, report dans les bases (Fig. 1.2),

2 - extension du domaine relationnel à des relations spécifiques, c'est-à-dire totalement indépendantes du système Socrate (Fig. 1.3),

3 - extension multi-socrate (Fig. 1.4).

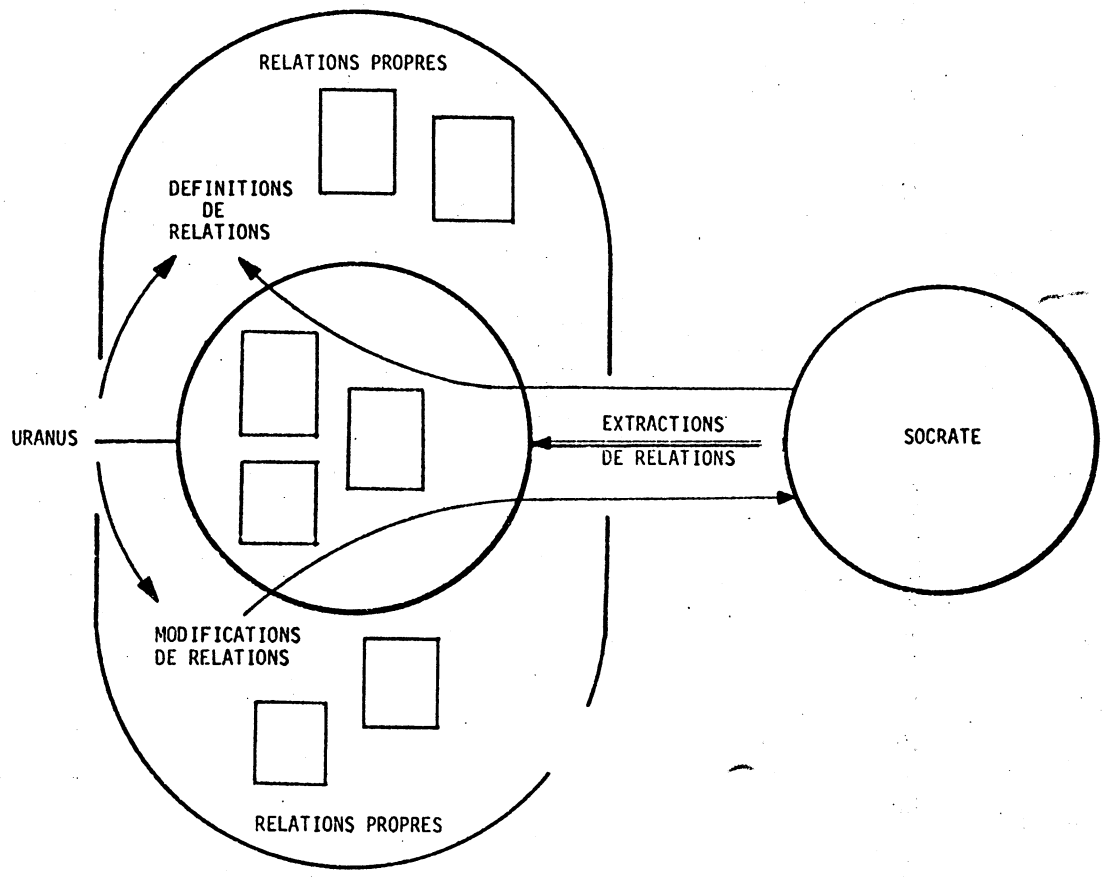


Fig. 1.3

L'adaptation logique consistera en la conception de *prédicats* ou *contraintes logiques* associés aux informations.

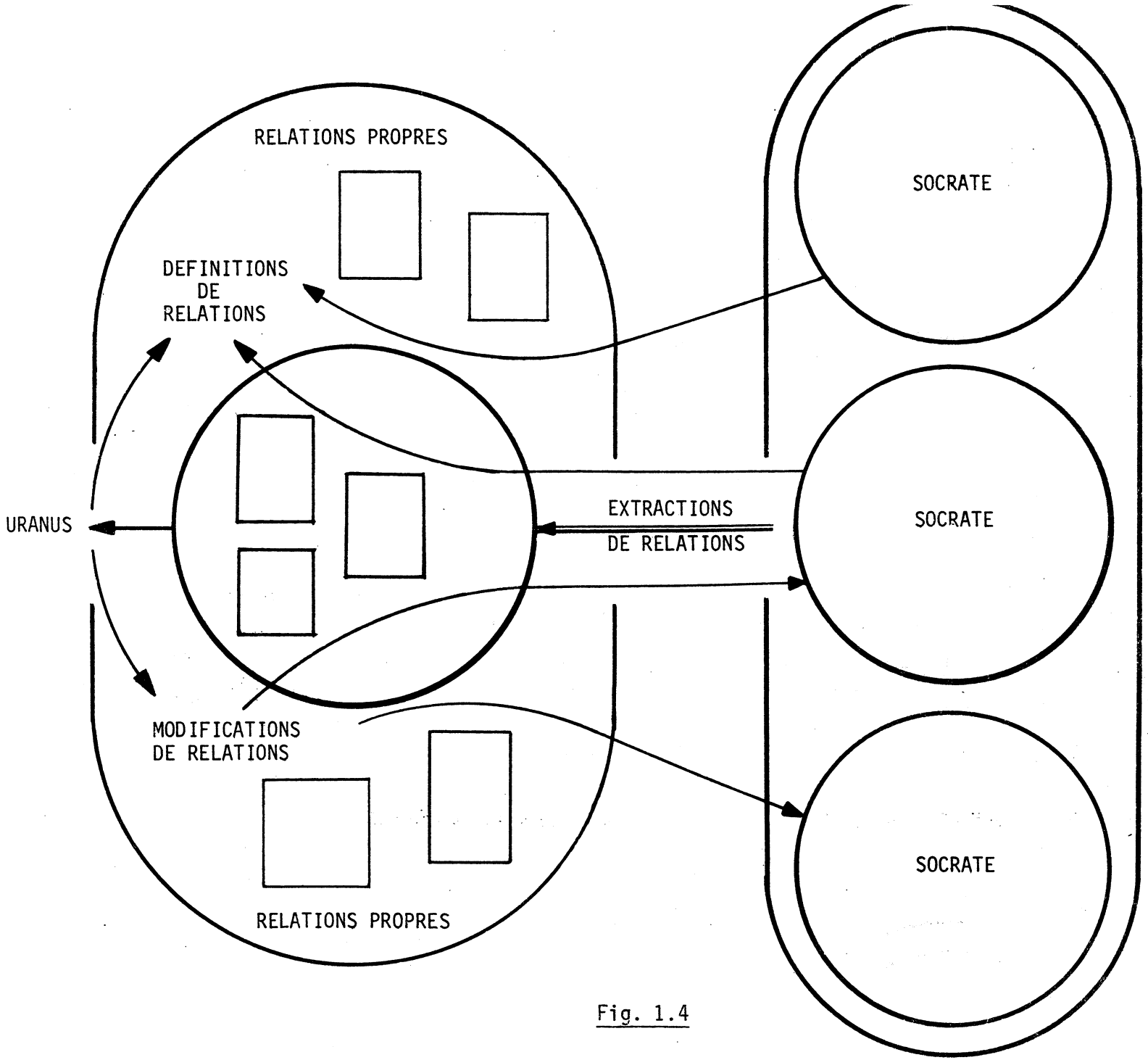

RELATIONS PROPRES
DEFINITIONS
DE
RELATIONS
URANUS
EXTRACTIONS
DE RELATIONS
MODIFICATIONS
DE RELATIONS
RELATIONS PROPRES
SOCRATE
SOCRATE
SOCRATE

Fig. 1.4

1.3. MOYENS

Grâce à la grande liberté de définition de l'information qu'ils procurent et à la finesse de leurs critères de manipulation, les modèles relationnels offrent des possibilités inégalées de compatibilité avec les systèmes existants.

De par leur conception même, ils permettent une évolution des structures de l'information répondant à l'adaptabilité structurelle (Chap. 2).

La définition, dans leurs langages de définition et de manipulation de relations, de prédicats associés aux données, procure les outils d'expression de règles d'utilisation contribuant à l'adaptabilité logique (Chap. 5).

Nous avons réalisé un prototype d'un tel système - URANUS - qui permet d'une part de concevoir des applications purement relationnelles, d'autre part d'utiliser des systèmes d'informations généraux du type Socrate dans un contexte général de coopération relationnelle multi-bases (Chap. 3 et 4) (Fig. 1.5).

Ceci représente une application des concepts relationnels développés par E.F. CODD [1]. Nous ne cherchons pas à garantir une transparence complète des moyens réalisés, mais plutôt à fournir un outil exploitable, dans un cadre suffisamment large pour y permettre l'intégration aisée de tout nouvel ensemble de gestion de bases de données.

Le projet a été conçu dans la perspective d'applications concrètes et réalisables à court terme, pour des utilisateurs de bases de données de type hiérarchique ou en réseau [2], afin de rendre leurs modèles évolutifs, et ceci sans rupture brutale du processus d'exploitation [16].

Nous verrons par ailleurs que ses perspectives d'application sont loin d'être épuisées (Chap. 6, § 2.2 à 2.4).

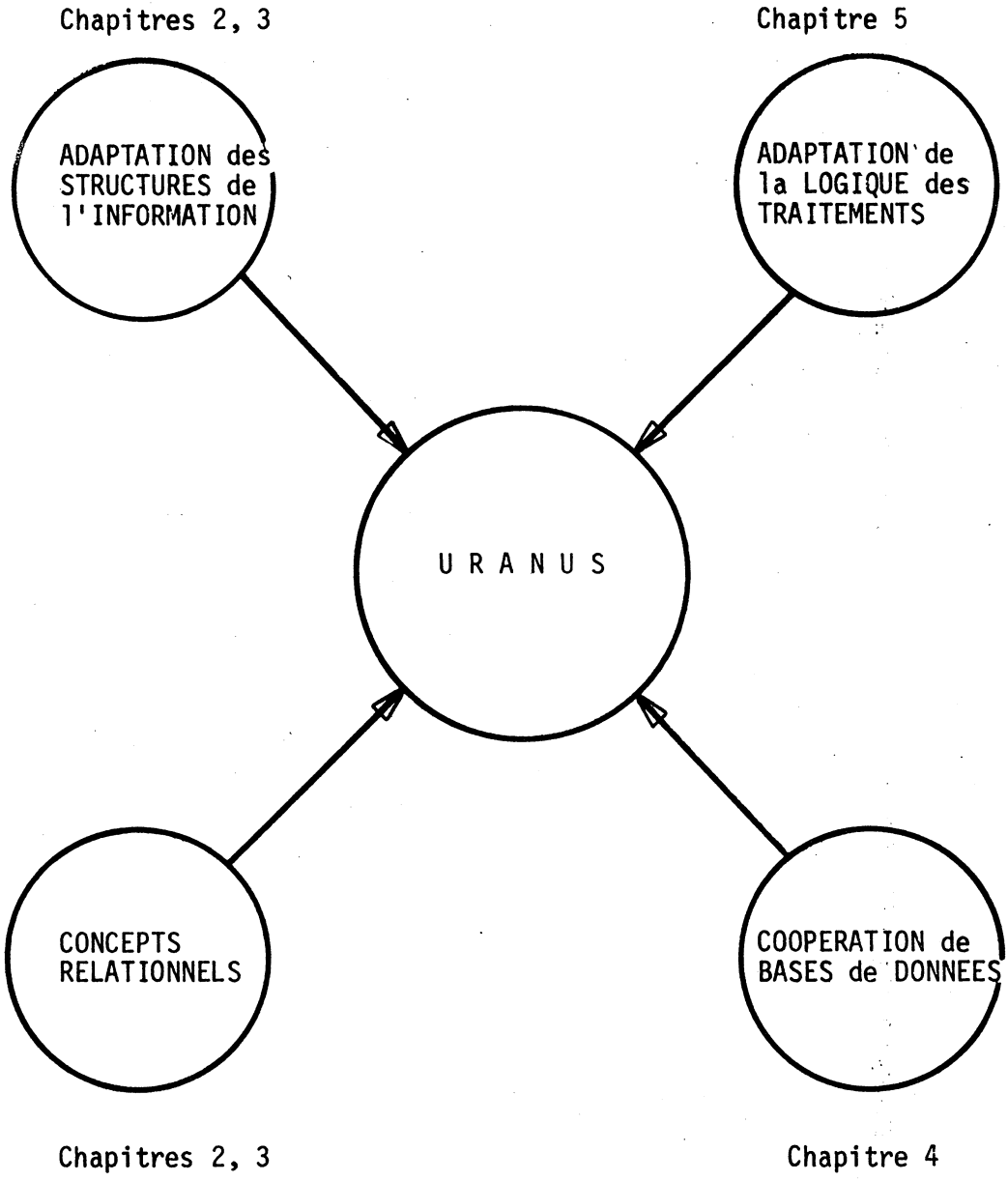

Chapitres 2, 3
Chapitre 5
ADAPTATION des STRUCTURES de l'INFORMATION
ADAPTATION de la LOGIQUE des TRAITEMENTS
U R A N U S
CONCEPTS RELATIONNELS
COOPERATION de BASES de DONNEES
Chapitres 2, 3
Chapitre 4

Fig. 1.5

1.4. TERMINOLOGIE

Afin de préciser les notions dont il sera question par la suite, nous rappelons brièvement les éléments de base du vocabulaire relationnel. On trouvera en [4] et [32] des commentaires complets sur ces définitions. Nous nous bornons ici à reprendre pour l'essentiel celles qui sont fournies en [2].

En termes mathématiques, une *relation* R est un ensemble de *n-uplets* $X_i = (x_1, x_2, \ldots x_n)$, où chaque élément x_i est pris dans un *domaine* D_i de valeurs.

Le nombre d'éléments x_i constituant un n-uplet est le *degré* de la relation.

Le nombre de n-uplets X_i est le *cardinal* de cette relation.

Une relation R peut être représentée sous forme de tableau. Chaque X_i correspond dans ce cas à une ligne du tableau. Les colonnes peuvent être nommées et l'ensemble de leurs éléments respectifs est alors un *constituant* de R.

De la définition, il découle que :

1 - pour tout $i \neq j$, $X_i \neq X_j$,
2 - l'ordonnancement $X_1, X_2, \ldots X_n$ est indifférent,
3 - l'ordonnancement des colonnes est indifférent.

Lorsque les valeurs d'un constituant ou d'un ensemble de constituants K détermine de façon univoque chaque n-uplet X_i, on appellera cet ensemble de constituants la *clé* de la relation.

Il peut exister plusieurs clés K_i pour une même relation R.

Une relation *normalisée*, plus précisément en troisième forme normale [33], est telle que :

1 - tout constituant est élémentaire, c'est-à-dire n'est pas lui-même une relation (première forme normale),
2 - tout sous-ensemble de constituants E déterminant de façon unique un constituant C_j ne lui appartenant pas, implique que *tous* les constituants $C_1, C_2, \ldots C_n$ soient déterminés de façon unique par E.

Cette dernière propriété fait appel à la notion de *dépendance fonctionnelle* [4].

CHAPITRE 2

DEFINITION DE RELATIONS N-AIRES SUR UNE BASE DE DONNEES

2.1. OBJECTIFS : ADAPTABILITE STRUCTURELLE

Sous le terme d'adaptabilité structurelle, nous regrouperons l'ensemble des méthodes et des moyens mis en oeuvre pour obtenir l'extensibilité des systèmes d'information, c'est-à-dire leur prise en charge de phénomènes externes non initialement répertoriés.

Ceux-ci se présentent entre autre sous la forme de données élémentaires, dont l'immixion dans une application conduit à en modifier la logique, conformément à l'évolution des situations modélisées.

A cet apport d'informations doit être associé, sans perturbation des applications non impliquées, un ensemble de liens sémantiques avec les données préexistantes.

Ceci sera réalisé, au niveau des structures de données, par la définition de relations n-aires sur les bases de données, et au niveau des traitements, par leur prise en charge simultanée avec les éléments ajoutés postérieurement de façon relationnelle (Fig. 1.3).

La définition de relations n-aires permet d'*uniformiser la vision des informations* et de simplifier l'utilisation de structures parfois complexes de celles-ci, tels que chemins d'accès et autres caractéristiques non élémentaires associées - inverses et références Socrate par exemple [3].

Ce cadre relationnel unique s'avère nécessaire au maintien intégral de tous les moyens en cours d'exploitation. Les données ajoutées ne sont donc pas intégrées aux systèmes existants ; elles sont créées et modifiées de façon physiquement indépendante des bases initiales. Celles-ci fournissent ensuite au système relationnel qui les englobe les informations déjà mémorisées (Fig. 2.1), dont le maniement conjoint avec les éléments ajoutés assure l'objectif visé grâce à un interface compatible.

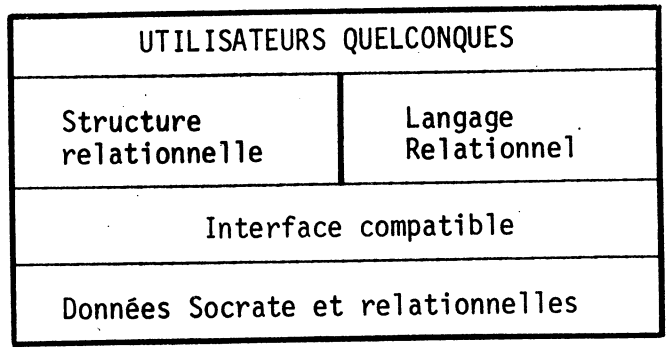


Fig. 2.1 - Objectifs : Schéma d'utilisation unique

Le problème que nous devons aborder en adoptant cette démarche est de déterminer, en partant de la multiplication des intermédiaires actuels entre un utilisateur administratif et l'information qui l'intéresse (Fig. 2.2), quelles structures et quels outils de manipulation lui fournir, donc quels moyens sous-jacents réaliser, afin de lui permettre de déterminer lui-même les meilleures formes de données susceptibles de lui convenir dans une application, ainsi que les traitements les mieux adaptés à celle-ci.

Indirectement, ceci doit contribuer à supprimer les obstacles techniques qui font qu'à l'heure actuelle, un utilisateur n'a pas généralement directement accès aux données dont il a besoin et ne peut se passer d'interlocuteurs informaticiens.

Pour cela, nous développerons autour des bases de données Socrate les outils permettant d'une part l'expression des relations n-aires qui y sont contenues, et d'autre part leur manipulation (Fig. 1.2) que l'on pourra coupler avec celle d'autres données, propres au système relationnel (Fig. 1.3). La mise en oeuvre de nouvelles applications sera faite par :

1 - l'interprétation des nouvelles associations qu'elles expriment,
2 - l'ajout et la modification des informations nécessaires.

Ce schéma se prêtant bien à une séparation nette des fonctions d'exécution, on distinguera donc les trois aspects complémentaires :

1 - utilisation de données des bases par des moyens relationnels
2 - apport d'information extérieure
3 - manipulation relationnelle, quelle que soit l'origine des informations.

Ce dernier point permettra, grâce aux relations précédentes, la manipulation de données plus évoluées à travers un interface simplifié.

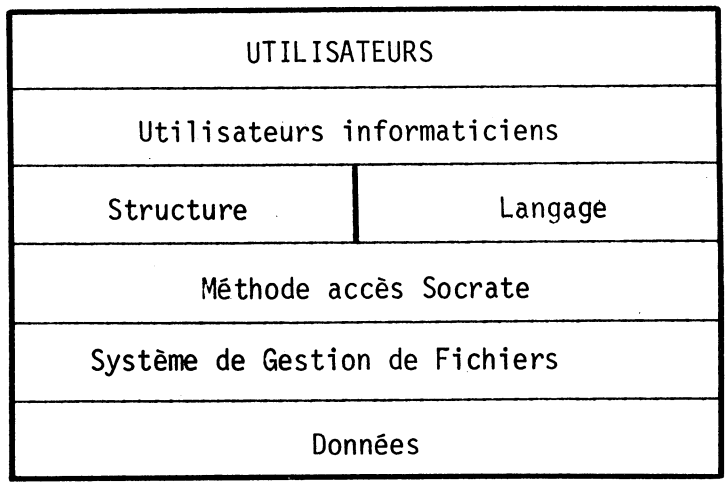


Fig. 2.2 - Interfaces Usagers-Socrate

Pour le premier aspect, on se limitera aux associations les plus simples, c'est-à-dire qu'aux entités Socrate on fera correspondre des relations, et aux caractéristiques Socrate des constituants de relations.
Ceci n'est pas conforme au principe des relations normalisées [4], [33], mais à ce stade, nous ne cherchons pas à optimiser la représentation des structures de l'information. On verra par la suite qu'il est possible d'étendre ce principe aux données non élémentaires et de prendre en compte indifféremment tout élément de structure Socrate (cf. § 3.2)

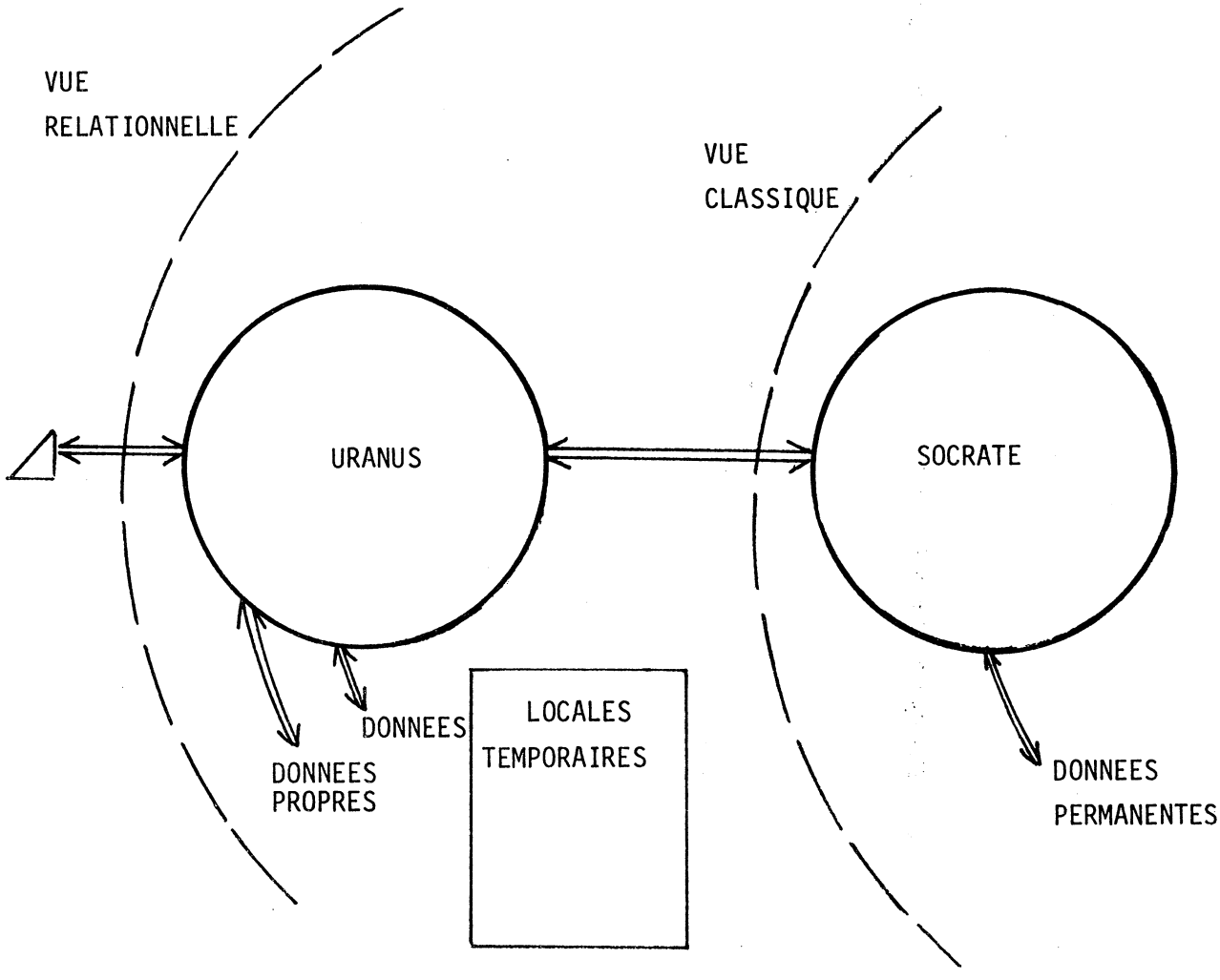


Fig. 2.3

On imposera de définir les associations dès la constitution des relations, en autorisant à la fois des données autonomes, c'est-à-dire spécifiques au système relationnel, et des relations mixtes, c'est-à-dire comportant, en partie ou en totalité, des domaines dont les valeurs sont *extraites* des bases de données. Les constituants correspondant seront qualifiés de "*corrélés*". La combinaison de ces deux ensembles permettra par la suite l'expression relationnelle de nouveaux traitements.

En ce qui concerne la répercussion des modifications relationnelles sur les données extraites, un souci d'intégrité maximum voudrait qu'elles soient effectuées immédiatement. Si cette solution est la plus satisfaisante théoriquement, on notera que c'est celle qui se traduit par les traitements les plus lourds, car elle implique la conception d'un "prisme" relationnel au travers duquel sont manipulées les entités. Notre optique consistera plutôt à extraire celles-ci momentanément et à les reporter une fois modifiées. Ceci autorisant la création de copies partielles multiples de données, et d'effectuer sur des informations-témoins des simulations prospectives à partir de situations existantes.

2.2. NIVEAU DE REALISATION

Dans l'optique de coopération de données d'origines diverses, un certain nombre d'ambiguïtés liées aux systèmes de gestion de bases de données classiques nous impose des choix de modélisation particuliers.

Il est connu qu'actuellement, les moyens informatiques disponibles n'offrent pas la possibilité de distinguer clairement [5] :

1 - l'objet que l'on désire modéliser,
2 - sa représentation concrète,
3 - son identification à l'intérieur du système d'information,
4 - ses propriétés intrinsèques,
5 - les valeurs associées à ces dernières,

c'est-à-dire que l'on ne sait pas décrire, une perturbation atmosphérique (1) par exemple, autrement qu'en fusionnant dans un "enregistrement", "entité" ou encore "set" (2), les caractéristiques (5) dans un ensemble (coordonnées, rayon, vitesse, direction, gradient, ...) fixé a priori (4) dont on aura arbitrairement fixé la dénomination (3).

L'absence de l'un des 3 derniers éléments conduit à une modélisation incomplète et rend l'exploitation de l'information aléatoire ou impossible.

En tentant d'associer des bases de données dans un ensemble plus général et en visant leur coopération, nous sommes confrontés à la nécessité d'une meilleure précision.

C'est en fait poser le choix d'une stratégie de réalisation.

Pour des raisons essentiellement liées aux objectifs initiaux du projet, c'est-à-dire la réalisation d'applications concrètes sur des collections d'informations existantes, nous avons choisi un niveau compatible avec toute structure de données existante (hiérarchique, réseau, etc ...) [9], donc "*structure-independant*", et autorisant tout développement ultérieur vers une modélisation plus sophistiquée, faisant appel aux notions d' "entité abstraite" ou "objet conceptuel" [6].

Le bien-fondé de cette approche est conforté par des travaux récents (Chen [7], Todd [8]) et montrant soit que :

1 - l'approche relationnelle est déductible de considérations théoriques plus générales ("entity-relationship model" de Chen), au même titre que d'autres philosophies (modèle hiérarchique, "entity-set model") et peut donc être inséré dans un ensemble plus large,

soit que :

2 - elle peut entrer dans la réalisation de schémas conceptuels faisant appel conjointement aux notions d' "entité" et de "lien sémantique" (Todd).

Dans les deux cas, la compatibilité entre (Fig. 2.4) :

1 - le niveau de réalisation choisi pour URANUS, structure-indépendant [36],
2 - le niveau inférieur de structure classique de données, faisant explicitement appel aux chemins d'accès à l'information,
3 - le niveau supérieur de description dans les systèmes d'information des objets modélisés et de leurs liens sémantiques,

assure que nous répondrons aux deux objectifs suivants :

1 - évolution des applications liées aux systèmes opérationnels,
2 - extension et coopération de ceux-ci par une intégration possible dans un modèle conceptuel englobant.

NIVEAU CONCEPTUEL	Objets, liens sémantiques
NIVEAU MODELISATION	Attributs, domaines, rôles
NIVEAU "STRUCTURE-INDEPENDT"	Relations
NIVEAU "STRUCTURE-DEPENDANT"	Entités, caractéristiques enregistrement, champs

Fig. 2.4

La réalisation du premier objectif sera décrite dans les chapitres 3 et 4.

En ce qui concerne le second qui n'a pas été abordé, nous mettons en avant le fait que tous les moyens développés dans URANUS sont exploitables pour sa réalisation.

Celle-ci pose essentiellement deux contraintes :

1 - description des objets conceptuels,
2 - description des liens sémantiques entre ceux-ci.

2.3. ATTRIBUTS, DOMAINES, LIENS SEMANTIQUES

Il a été montré par Todd que l'apport simultané de la notion de liens et d'objets conceptuels ("entities") permet de décrire complètement [8] un modèle de façon relationnelle en associant :

1 - à une classe d'éléments l'ensemble de ses propriétés,
2 - à un lien sémantique ses propriétés et les objets qui le composent.

En intégrant dans une même relation un domaine particulier ("entity-set") représentatif de la classe d'objets et un domaine correspondant à une propriété de celle-ci, "value-set", on réalise une description qui permet de plus de distinguer l'objet modélisé de son identification dans le système.

Exemple : description d'un élément de la classe "véhicule" défini par ses propriétés de "marque", "couleur" et "numéro", puisant leurs valeu dans les domaines "FABRICANT", "COULEUR" et "IMMATRICULATION".
- MARQUE-VOITURE (véhicule : VOITURE, marque : FABRICANT) ;
- COULEUR-VOITURE (véhicule : VOITURE, couleur : COULEUR) ;
- NUMERO-VOITURE (véhicule : VOITURE, numéro : IMMATRICULAT).

Le domaine VOITURE est représentatif de la classe d'objets représentés, et recouvre une identification discriminante liée au système et non accessible à l'usager qui ne peut référencer un élément de la classe "véhicule" que par une sélection sur ses propriétés "marque", "couleur" ou "numéro". C'est dans cet ensemble uniquement qu'il aura la possibilité de définir des clés d'accès, donc distinctes de celle utilisée de façon interne.

Cette philosophie a pour autre avantage d'autoriser l'existence de relations puisant simultanément leurs constituants dans un même ensemble de valeurs,

ce qui n'est pas possible dans les systèmes structure-dépendants.

Chen [7] a développé cette approche en généralisant les notions de lien ("relationship") et d'objet ("entity").

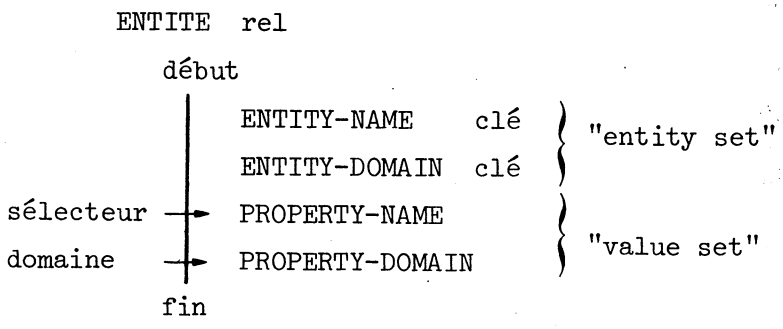
Il précise la fonction de chaque propriété par un "rôle" sémantique [10] alloué à un "attribut". L'intégration de ces notions dans un ensemble plus général d'où sont déductibles les modèles relationnels, hiérarchiques et réseau ouvre des perspectives très importantes aux applications de URANUS.

Il est en effet possible à partir de la représentation des "entités" sous forme de relations de modéliser les liens sémantiques selon des relations en 3ème forme normale de Codd, équivalentes au niveau "structure-indépendant" de Chen (Fig. 2.5), ou selon des relations binaires.

Ceci assure que tout modèle conceptuel de système d'information peut être décrit à l'aide des éléments de base présentés plus loin (Chap. 3 et 4).

2.4. APPLICATION

En anticipant sur les éléments syntaxiques de définition de relation dans URANUS, la description d'un objet selon les normes de Todd peut être alors :



PROPERTY-NAME est un "sélecteur" ou constituant puisant ses valeurs dans PROPERTY-DOMAIN. ENTITY-DOMAIN identifie la clé interne ("surrogate") et ENTITY-NAME la désignation de l'objet par l'utilisateur. Sa déclaration comme "cle" dans URANUS crée, au moment de l'utilisation, un chaînage interne de toutes les propriétés de même ENTITY-NAME, donc appartenant au même objet.

Chen schématise son approche par un parallèle entre les modèles existants et celui qu'il propose, en les structurant en niveaux selon le degré de conceptualisation cherché et de réalisation visé :

	entity-relationship	relationnel	réseau
NIVEAU 1 conceptuel	objets conceptuels liens sémantiques	relations binaires	
NIVEAU 2 modélisation	relations représentant : - objets conceptuels - liens sémantiques	relations en 3ème forme normale	entités abstraites relations sémantiques
NIVEAU 3 structure-independant	attributs, domaines, rôles	n-uplets	
NIVEAU 4 structure-dependant			aggrégats, chemins d'accès

Fig. 2.5

Exemple de description d'objet selon Chen (au niveau 2) :

- L'objet est décrit par une relation (Fig. 2.6).
- Chaque élément est un attribut de l'objet (NOM, INSEE, ...).
- Il puise ses valeurs dans un certain ensemble de domaines (NOM et PRENOM, NO, NO-RUE et VILLE ...).

ATTRIBUTS	NOM		INSEE	ADRESSE		
DOMAINES	NOM	PRENOM	N°	N°	RUE	VILLE
n-uplets	n1	p1	n°1	n°1	r1	v1
	n2	p2	n°2	n°2	r2	v2
	....	.....	.....	.....	.....	.....
	nm	pm	n°n	n°m	rm	vm

} schéma relationnel classique

Fig. 2.6 - Objet "PERSONNE"

- Les notions relationnelles sont incluses dans cette vision, par fusion des noms d'attributs et de domaines.

Exemple de description de lien sémantique selon Chen :

ENTITE	PERSONNE	VOITURE	attributs de la relation	
ROLE	PROPRIETAIRE	VEHICULE		
ATTRIBUT	NOM	NUMERO	ACCIDENT	GARAGE
DOMAINE	NOMS	N°	LIEU	ADRESSE
n-uplets				

schéma relationnel classique

Fig. 2.7 - Relation "CONDUCTEUR"

Le schéma relationnel est incorporable au modèle de Chen par association d'un ensemble d'attributs aux constituants d'un objet ou d'un lien sémantique, et d'un ensemble de rôles associés aux objets de chaque lien. Le "rôle" est ici considéré comme élément caractéristique de chaque objet associé au "lien sémantique". Il joue un rôle analogue au "selector" de Todd, mais de façon beaucoup plus explicite et formalisée (Fig. 2.7).

Selon Todd, cela donnerait :

```
RELATION rel
  début
    |  NOM
    |  ENTITY1-NAME
    |  ENTITY2-NAME
    |  ENTITY1-DOMAIN
    |  ENTITY2-DOMAIN
  fin
```

De façon plus complète, nous donnons ci-dessous une structure possible d'objet selon Chen dans URANUS.

La valeur de l' "attribut" d'un objet est un pointeur (ATTRIBUTE-PTR) vers un n-uplet de la relation "ENTITY-ATTRIBUTE", associant à un nom d'entité et un nom d'attribut une certaine valeur (l'attribut est une fonction de ENTITE dans un produit cartésien de valeurs, celles-ci pouvant elles-mêmes être définies comme ensembles prédéclarés ("value-sets").

Selon Chen, il peut y avoir un nombre quelconque de domaines par attributs et un nombre quelconque d'attributs par objet.

On aboutit alors à la méta-structure suivante ("entity-relation") :

```
DOMAIN rel
  début
    |  NOM
    |  VALUE-PTR
       ATTRIBUTE-PTR
  fin

ENTITY-ATTRIBUTE rel
  début
    |  NOM
    |  ENTITY-PTR
    |  DOMAIN-PTR
  fin

OBJET rel
  début
    |  NOM
    |  ATTRIBUTE-PTR
  fin
```

```
VALUE-SET rel
  début
    |  NOM
    |  VAL
  fin
```

De la même façon, la description d'une relation ou lien sémantique ("relationship") entre divers objets peut être faite complètement à l'aide de deux éléments complétant les précédents ("relationship relation") :

```
RELATION rel
  début
    |  NOM
    |  ENTITY-PTR
    |  ROLE
    |  ENTITY-ATTRIBUTE-PTR
    |  DOMAIN-PTR
    |  RELATION-ATTRIBUTE-PTR
  fin

RELATION-ATTRIBUTE rel
  début
    |  RELATION-PTR
    |  NOM
    |  DOMAIN-PTR
    |  ATTRIBUTE-PTR
  fin
```

Ce qui a été décrit dans les paragraphes précédents permet de situer URANUS dans un contexte de développement de schémas conceptuels généralisés, et de montrer comment celui-ci peut constituer un outil pour une telle recherche.

Nous donnons maintenant les détails de ce qui a été réalisé concrètement afin d'amorcer ce processus, en rendant possible, à un niveau plus élémentaire, indépendant des structures de l'information, la coopération de bases de données diverses. Ceci constitue une première étape dans la conception et la mise en oeuvre d'un schéma de coexistence [6] s'inscrivant dans un cadre théorique plus général.

2.5. DEFINITION DES RELATIONS

Afin d'uniformiser l'ensemble des traitements liés au système et à son exploitation, toutes les informations qu'elles soient internes ou propres à une application donnée sont traitées sous forme relationnelle.

On a donc deux types de relations :

- relations système : "relation-maître" (cf. annexe 7.3), "corrélations" (cf. § 3.2), etc ...
- relations utilisateur :
  - . relations liées à des données Socrate, dites *corrélées*,
  - . relations indépendantes de Socrate, dites *spécifiques*.

Une relation est déterminée par son nom, ses constituants, son degré, c'est-à-dire le nombre de constituants qui la compose. Ceux-ci sont caractérisés par leur nom et le type de leurs valeurs prises dans un certain domaine. On en admet trois, compatibles avec ceux du langage de définition de structure Socrate :

- mot : chaîne alphanumérique quelconque,
- numérique : valeur entière quelconque,
- liste de valeur : ensemble de valeurs prises dans un domaine prédéfini.

Exemple :

```
ETUDIANT rel 1000
  début
    |   NOM mot 30
    |   AGE de 0 à 100
  fin
```

Pour des raisons d'implémentation, on doit préciser le cardinal de la relation ou nombre maximum de n-uplets qu'elle peut comporter. De même, on devra fournir la longueur maximum des chaînes alphanumériques et les bornes des valeurs numériques, des valeurs par défaut étant prises dans le cas contraire.

Les listes de valeurs étant des relations unaires, on les définira comme telles, exprimant qu'un constituant peut y puiser ses valeurs par une référence explicite :

```
ETAT-CIVIL rel 10 5 (CELIB VEUF MARIE DIV)
ETUDIANT rel 500
  début
  |   NOM mot 30
  |   INSEE de 0 à 999
  |   ETAT dans ETAT-CIVIL
  fin
```

Ceci exprimant que le constituant ETAT de la relation ETUDIANT prend ses valeurs dans la liste ETAT-CIVIL, relation unaire comportant au plus 10 éléments de longueur maximum 5 caractères. On constate que les relati de ce type peuvent s'accroître par adjonction ultérieure d'éléments et inversement.

Tout constituant appartenant à la clé de la relation est dénoté par adjonction de ce qualificatif, en imposant que si elle en comporte plusieurs, ils soient définis de façon contiguë. Exemple :

```
ETUDIANT rel 1000
  début
  |   NUMERO de 0 à 1000 CLE
  fin

ENSEIGNEMENT rel 500
  début
  |   CODE de 0 à 500 CLE
  |   LIBELLE mot
  fin

CLASSEMENT rel 10000
  début
  |   NUMERO de 0 à 1000 CLE
  |   CODE de 0 à 500 CLE
  |   ORDRE de 0 à 1000
  fin
```

L'ensemble des définitions de ce type fournies au terminal par l'utilisateur ou en données d'entrée d'un programme batch permet la mise à jour de l'espace de travail de celui-ci (Fig. 2.8).

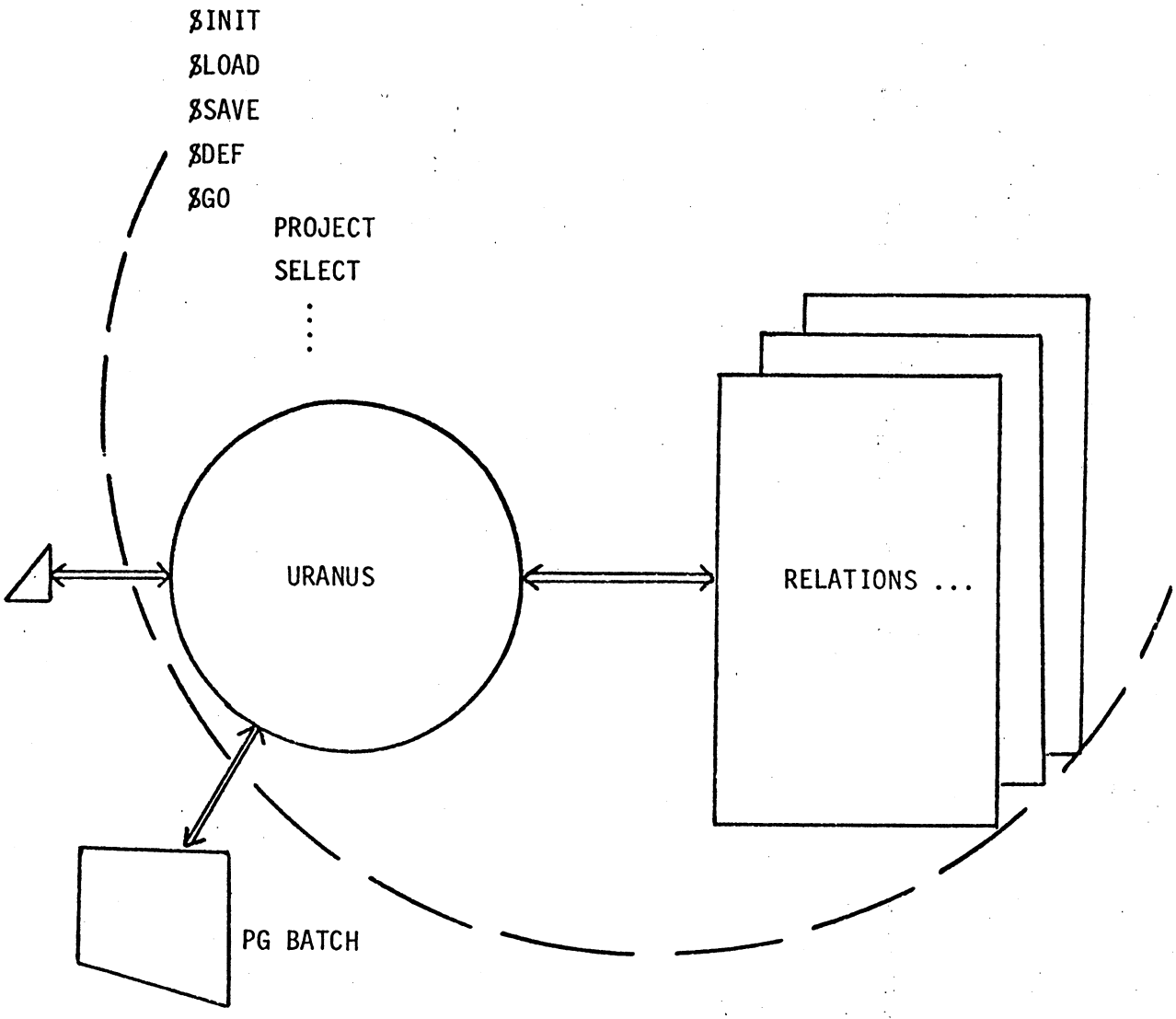


Fig. 2.8

Il est caractérisé par son nom, fourni à l'initialisation dans la commande $INIT <nom> (Annexe 7.2), et constamment décrit par la "relation-maître".

Le rôle de cette dernière est à la fois de mémoriser l'ensemble des paramètres de l'usager en question - nom, taille de son espace de travail, adresse interne des relations définies, etc - et de décrire l'ensemble de ses données relationnelles - noms de relations, de constituants, degré, clés, etc ...

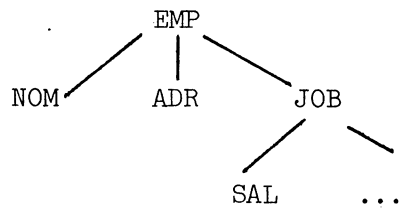
Dans la définition de relations corrélées, les constituants liés à des données Socrate doivent être complétés par l'indication des caractéristiq associées, selon leur filiation hiérarchique dans la structure correspondante. Ceci doit être mentionné par l'utilisateur à la suite du mot-clé "IDEM" (Fig. 2.9).

EXEMPLES DE DEFINITION

La partie soulignée juste à titre de présentation correspond à la définition d'un constituant d'une relation à partir d'une structure SOCRATE.

A)
```
EMPLOYE REL 500 IDEM EMP
    début
    |   NOM MOT IDEM NOM
    |   ADRESSE MOT IDEM ADR
    |   SALAIRE DE 0 A 1000 IDEM SAL DE JOB
    |   AGE DE 0 A 100
    fin
```

QUALIFICATION SOCRATE CORRESPONDANTE



B)
```
VOITURE REL
    |   NUMERO            IDEM NUMERO de VOITURE
    |   PROPRIETAIRE
    |   DATE-ACCIDENT    IDEM DATE de ACCIDENT
    |                           de VOITURE
    |   LIEU-ACCIDENT    IDEM LIEU de ACCIDENT
    |                           de VOITURE
    fin
```

```
entité VOITURE
    |   NUMERO
    |   MARQUE
    |   entité ACCIDENT
    |       |   DATE
    |       |   LIEU
    |   fin
fin
```

Fig. 2.9

Les relations doivent être définies conformément à la grammaire suivante.

```
<RELATIØN> := <RELID> 'REL' <CARDINAL> <QUALO> <BLØC>
              ! <RELID> 'RELVAL' <CARDINAL> <LØNG> <LISTVAL>
<QUALO>    := 'IDEM'  <ENTITE> <QUAL1> ! &
<QUAL1>    := 'DE'    <ENTITE> <QUAL1> ! 'DANS' <BASE> ! &
<BLØC>     := 'DEBUT' <CITATIØN> 'FIN'
<CITATIØN> := <DØMAINE> <TYPE> <QUALIFIC> <CITATIØN> ! &
<QUALIFIC> := 'IDEM'  <ENTITE> <QUALIF> ! &
<QUALIF>   := 'DE'    <ENTITE> <QUALIF> ! 'DANS' <BASE> ! &
<TYPE>     := 'DE' <BINF> 'A' <BSUP> ! 'MØT' <LØNG>
<LISTVAL>  := '('  <LISTVAL> ')' ! <VAL> <LISTVAL>
<VAL>      := NB ! IDENT ! &
<RELID>    := IDENT
<ENTITE>   := IDENT
<DØMAINE>  := IDENT
<BASE>     := IDENT
<CARDINAL> := NB
<BINF>     := NB
<LØNG>     := NB ! &
```

GRAMMAIRE DE DEFINITIØN DES RELATIØNS.
****

2.6. LIMITES

Des contraintes particulières nous sont imposées par la nature des opérations que nous entendons effectuer au niveau de la coopération URANUS-Socrate. D'une part, l'ensemble des modifications effectuées dans le cadre relationnel posera des problèmes d'intégrité en cas de traitements concurrents sur la base de données. D'autre part, et de façon plus immédiate, on verra qu'il n'est pas toujours aisé d'exprimer en termes relationnels l'ensemble des éléments d'une base de données Socrate. Ceci est dû essentiellement au fait qu'elle peut contenir des informations qui ne sont pas élémentaires, mais des chemins d'accès [3]. On a vu que cette façon de procéder n'a plus de raison d'être au niveau relationnel, et nous aurons donc à adapter ce type d'éléments aux possibilités d'expression de celui-ci. De plus, la nature hiérarchique des structures Socrate imposera aussi ses propres règles du fait que nous envisageons la *génération automatique* des relations à partir de leurs définitions par l'utilisateur. Ceci

1 - pour la suppression des éléments liés aux inverses et références imbriquées
2 - pour la création des éléments appartenant à des entités de même niveau.

2.6.1. Entités de même niveau

On s'est cantonné jusqu'ici aux relations ne comportant que des domaines simples, c'est-à-dire normalisées. Pour ne pas nous limiter à des structures Socrate élémentaires, on admettra cependant la création de relations à partir d'entités imbriquées sur plusieurs niveaux.

On conçoit sur l'exemple suivant le problème posé par la création automatique de telles relations, comportant des constituants appartenant à deux entités parallèles englobées.

La relation EMPLOYE devant contenir des caractéristiques NUMERO et PRENC dont le lien dans la base est seulement d'être associées au même père hiérarchique SALARIE, il est peu probable qu'elles aient un lien sémantique clair au niveau relationnel. Même si cela était le cas, il subsisterait un problème quant au choix de constitution de la relation : produit cartésien des deux ensembles NUMERO et PRENOM, ou autre ?

Le résultat serait une relation formée par la composition de deux relations plus élémentaires : R1 (NOM, VOITURE) et R2 (NOM, ENFANT).

Exemple :

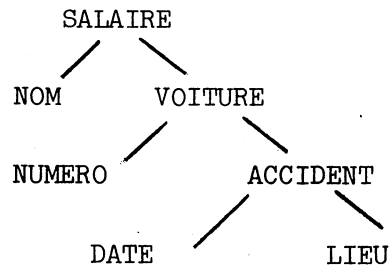
RELATION

```
EMPLOYE REL IDEM SALARIE
  | NOM mot IDEM NOM
  | NUMERO de 0 à 9 idem NUMERO DE VOITURE
  | PRENOM mot idem PRENOM de ENFANT
fin
```

SOCRATE

```
Entité SALARIE
  |  NOM
  |  entité VOITURE
  |    | NUMERO
  |  fin
  |  entité ENFANT
  |    | PRENOM
  |  fin
fin
```

Elle devrait être par conséquent décomposable, ce qui n'est pas assurément conforme aux données initiales Socrate [35].
On interdira donc de créer des relations de ce type, en se bornant à le faire pour seulement des entités incluses en une hiérarchie dégénérée du type :



Il paraît en effet plus naturel de créer deux relations SALARIE-NOM et SALARIE-ENFANT, combinables par la suite en fonction des besoins de l'utilisateur, mais il est difficile de les devancer a priori.

2.6.2. Inverses et références incluses

On a, dans ce cas, une relation de type (1,n) et l'exemple suivant :

```
entité PERSONNE                          entité VOITURE
  |   NOM mot                             |   NUMERO de 0 à 9
  |   AUTO inverse toute VOITURE          |   MARQUE mot
fin                                      fin
```

peut se traduire en termes relationnels en :

```
PERSONNE rel idem PERSONNE
  |   NOM mot idem NOM
  |   NUMERO de 0 à 9 idem NUMERO de AUTO
fin
```

Si la création et la modification des données relatives aux deux entités peut se faire sans inconvénients moyennant l'existence de clés compatibles dans les deux systèmes, par exemple NOM et NUMERO, il n'en va pas de même de la suppression par des moyens automatiques. Deux types d'opérations peuvent être effectués dans la base de données : suppressior de l'inverse seul ou de l'entité. On laissera le soin à l'utilisateur de préciser lequel car la seule alternative est de choisir un type d'opération, ce qui est a priori restrictif.

Ces remarques posent clairement le choix qui nous est offert. A savoir assurer une cohérence plus grande entre les traitements effectués au niveau relationnel et les données Socrate, en interdisant certains types de définitions de relations ou certaines opérations sur celles-ci. Soit admettre un certain laxisme, laissant à l'utilisateur le soin de contrôler lui-même la bonne exécution de ce qu'il a effectué.

Ces limitations sont liées au principe d'expression en termes relationnel de structures hiérarchiques et en réseau. Il semble que l'on soit aussi obligé de tenir compte des éventuelles transformations qu'il est possible de faire subir à l'ensemble des relations de départ, où seront prises les données corrélées. Il sera en effet possible de construire de nouvelles relations à partir de celles-là, de les combiner, d'en optimiser les représentations. Toute manipulation de données corrélées devra alors être liée à cet ensemble si l'on ne veut pas courir le risque d'avoir autant d'opérateurs que de relations.

On devra donc soit associer toute opération aux données initiales pour tenir compte immédiatement des transformations subies en cours de traitement, soit s'astreindre à reconstituer l'ensemble modifié in fine, ce qui peut avoir un avantage d'efficacité. Cette dernière optique nous semble plus simple et plus réaliste, dans la mesure où elle permet une

moindre charge d'exécution malgré une contrainte plus sévère de travail : la mise à jour des données initiales. Il peut exister par ailleurs des possibilités d'incohérence incontrôlables liées à l'existence de multiples exemplaires d'un même constituant dans diverses relations, susceptibles de subir des traitements indépendants.

On imposera donc que toute donnée corrélée soit modifiée dans la relation où est définie son association avec des éléments Socrate pour que le report des mises à jour puisse avoir lieu (§ 3.8).

Enfin, le parti que nous avons suivi de n'utiliser qu'un langage de manipulation réduit et très orienté vers une utilisation relationnelle (cf. § 3.7) permettant une grande fonctionnalité et une bonne adaptation aux objectifs retenus, pourra s'avérer restrictif. On en détaillera plus loin les éléments et on montrera sur un exemple les cas de difficultés qui peuvent surgir dans la résolution de certains problèmes. Disons qu'ils ne semble pas liés à la nature du système relationnel, mais plutôt au type de possibilités choisies dans les éléments du langage et qu'il existe une façon simple de les éviter - l'inclusion dans un langage hôte.

2.7. REALISATION : URANUS

Avant d'étudier de façon détaillée les différents composants du système et le type d'application qui peut être mise en place grâce à l'existence du système relationnel associé à l'interface base de données - ensemble que nous appellons "URANUS" pour Utilisation Relationnelle de bases Socrate - nous spécifierons ici les fonctions de manipulation de base sur les relations.

1 - Les objets de manipulation sont les relations et dans celles-ci les n-uplets, les constituants et les valeurs immédiates.

2 - Les critères de travail sont les filtres de sélection sur ceux-ci, ou le contexte, c'est-à-dire le séquencement.

3 - Le résultat d'une opération est toujours une relation.

4 - Les opérations sont :

a) création de relations à partir de données Socrate ;

b) création de n-uplets par critère de sélection, par séquencement (insertion) ;

c) modification de relations, de constituants, de la même façon ;
d) création de relation sur critère ou par combinaisons d'autres relations ;
e) suppression de n-uplet, id.

Ces principes constituent la base de l'aspect externe d'utilisation du système. On les a choisis aussi simples que possible, afin de pouvoir créer un système très ouvert, conversationnel et éventuellement incorporable à des chaînes de traitement plus élaborées et plus classiques. Sa réalisation devra faire appel à des traitements d'interprétation efficaces et concis (Annexe 7.6).

L'aspect pratique nous a conduit de même à concevoir un ensemble mono-utilisateur, d'une part pour des raisons de simplicité, mais aussi parce que l'aspect concurrentiel des traitements est déjà partiellement résolu par la conception de l'interface relationnel, dont les opérateurs sont strictement liés à une vue d'un utilisateur, et donc totalement indépendants entre eux.

On trouvera plus loin quelques exemples des possibilités du langage retenu et d'expression des corrélations Socrate (Annexes 7.4, 7.7).

L'ensemble des définitions de relations et des paramètres de leur exploitation sera mémorisé dans une relation particulière, dite relation-maître, décrivant à un instant donné l'état du système, les relations connues, leurs constituants, les clés, les corrélations définies et toute donnée nécessaire à leur manipulation. Sa constitution initiale sera liée à l'interprétation des définitions et des corrélations fournies par l'usager. Elle sera constamment mise à jour pendant l'activité de celui-ci. Les corrélations donnant lieu à la définition de relations internes (cf. § 3.2) seront exploitées par un ensemble de fonctions propres au système, générant automatiquement les opérateurs constituant les diverses fonctions d'accès et de modification d'une application donnée sur les éléments des bases Socrate (Annexe 7.5).

On tirera parti de l'existence d'un interface standard de la méthode d'accès dite batch, ce qui permettra une génération directe en langage source de manipulation Socrate, une compilation autonome et parallèle, suivie d'un cataloguage décrit plus loin (§ 3.2).

CHAPITRE 3

COOPERATION D'UNE BASE SOCRATE & RELATIONNELLE

3.1. OBJECTIFS

Les propriétés des systèmes relationnels doivent pouvoir, selon des procédés à définir, être adaptés et appliqués à *l'extension du champ d'application* de base de données non relationnelles. Si l'on note que leur mise en pratique a permis de réaliser des systèmes généralisés de conversion de données [11] ou d'optimisation de structures en réseau [12], à notre connaissance la mise en oeuvre parallèle et concertée de ces deux ensembles n'a jamais été faite de façon opérationnelle. Dans le premier cas, il s'agit d'établir les modalités de transition d'une phase de définition de données en une autre, par un intermédiaire rigoureux et bien formalisé, se prêtant donc à des opérations complexes mais ne relevant que de constatations statiques de définition des données. Dans le second, il s'agit de même d'initialiser le processus de mise en oeuvre d'une application par une définition optimisée des structures de l'information, sans envisager une extension de celles-ci puisque leur domaine reste équivalent à lui-même.

On est donc confronté au double problème de *coopération* entre deux systèmes, et d'*extension* de l'un d'entre eux grâce à celle-ci. On définit ici les modalités et les bases d'une réalisation possible (Fig. 3.1).

Il est clair ~~que les~~ possibilités d'expression de structures réseau en termes relationnels nous sont très utiles. Notre démarche consiste à définir des relations n-aires sur des bases de données Socrate et à déterminer les conditions précises de leur manipulation, le niveau auquel cela doit avoir lieu, enfin les contraintes et possibilités d'adaptation que cela apporte quant à leur utilisation. Précisons cependant que ce principe de coopération n'implique pas une gestion directe de bas niveau des données des bases initiales. L'ensemble des opérateurs qui seront appliqués à celles-ci sera conforme aux impératifs de mise en oeuvre du système choisi, en l'occurrence Socrate, c'est-à-dire que nous respecterons les règles d'accès définies pour une utilisation "normale" de celui-ci, à l'exclusion de toute modification qui aurait pu nous paraître souhaitable par ailleurs [13].

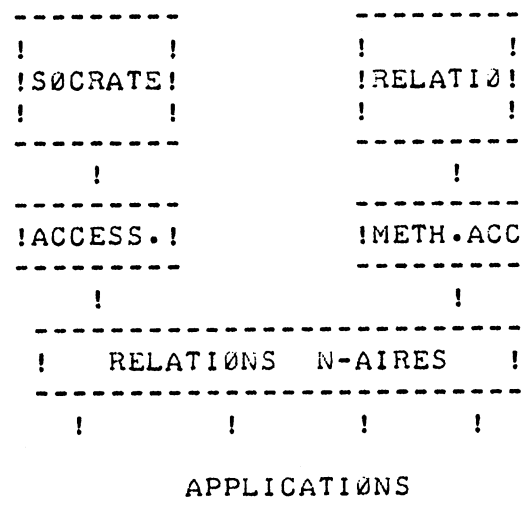


Fig. 3.1 - . CØØPERATIØN D' UNE BASE SØCRATE ET RELATIØNNELLE.

Le système conçu devra donc présenter un double aspect :

- gestion relationnelle de données autonomes,
- coopération de données relationnelles et Socrate,

tout en garantissant l'intégralité des possibilités antérieures.

Les relations définies devront d'une part traduire l'ensemble des informations contenues dans la base de données afin d'en permettre le traitement, d'autre part assurer la compatibilité avec les informations ajoutées.

Dans ce cadre, nous nous attacherons de plus à étudier les possibilités de coopération de base de données multiples :

1 - définition de relations sur chaque base,
2 - manipulation parallèle de ces relations,
3 - modification des informations contenues dans chaque base,

afin d'étendre les applications potentielles découlant d'un environnement réel où l'information se trouve disséminée dans une multitude de moyens de stockage.

3.2. CORRELATIONS SOCRATE

Une relation pouvant être associée globalement ou partiellement à des éléments d'une structure Socrate (les éléments relationnels correspondants étant dits "corrélés"), on permettra l'expression de liens à tous les niveaux d'imbrication hiérarchique de celle-ci.

En ce qui concerne les éléments qui ne sont pas élémentaires - références, inverses, anneaux [3] - il n'y a pas de contradiction à les exprimer en tant que constituants corrélés, puisque c'est leur interprétation qui devra fournir les données adéquates correspondantes. Ceci est possible parce que les opérateurs sur les bases de données seront générés directement en langage source Socrate, et que l'on traitera donc correctement les éléments de citation correspondants. Au niveau des définitions, on permettra donc *tout élément* de structure Socrate dans l'expression des corrélations.

Lorsque plusieurs niveaux de qualification Socrate seront impliqués - pour les éléments correspondant à des entités imbriquées - il suffira de fournir correctement leur filiation hiérarchique.

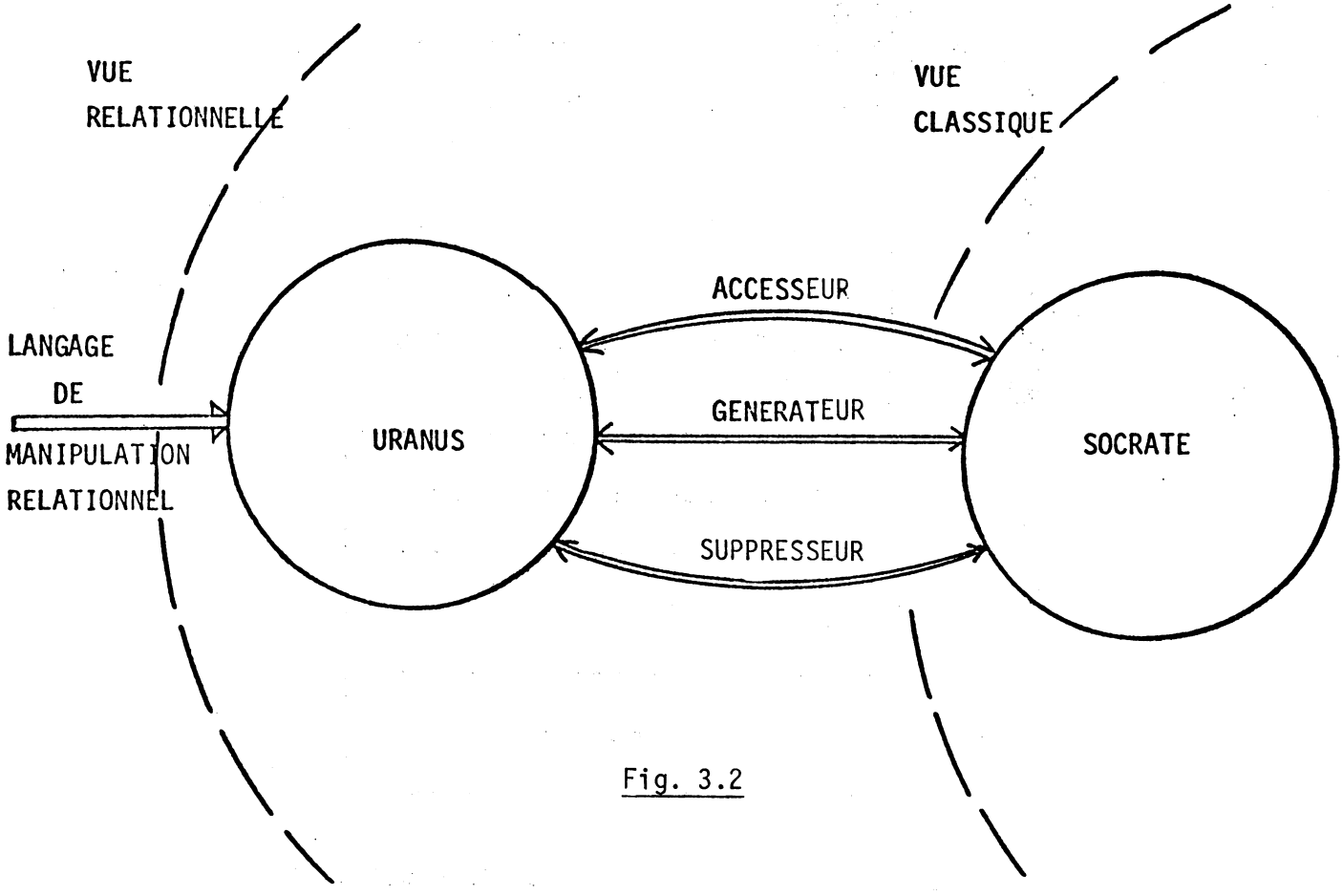


Fig. 3.2

Exemple :

```
EMPLOYE rel 500 idem SALARIE                    entité 1000 SALARIE
   début                                           début
      |   NOM mot CLE idem NOM                       |   NOM mot 30 avec CLE
      |   AGE de 0 à 100 idem AGE                    |   AGE de 0 à 100
      |   SALAIRE de 0 à 1000 idem SALAIRE           |   SALAIRE de 0 à 1000
      |   ADRESSE mot 100                           fin
   fin
```

La relation EMPLOYE, dont tous les éléments corrélés par "idem" sont associés à l'entité SALARIE, citée au niveau "rel", a trois constituants AGE, NOM et SALARIE liés par "idem" à trois caractéristiques de l'entité. Seul, ADRESSE est un constituant propre.

Les corrélations Socrate seront mémorisées de façon particulière afin de permettre la *génération automatique des opérateurs correspondants* sur la base de données : accesseur, suppresseur, générateur (Fig. 3.2). Elles assureront le lien entre une définition relationnelle de l'information et son équivalent dans une structure Socrate (Fig. 3.3).

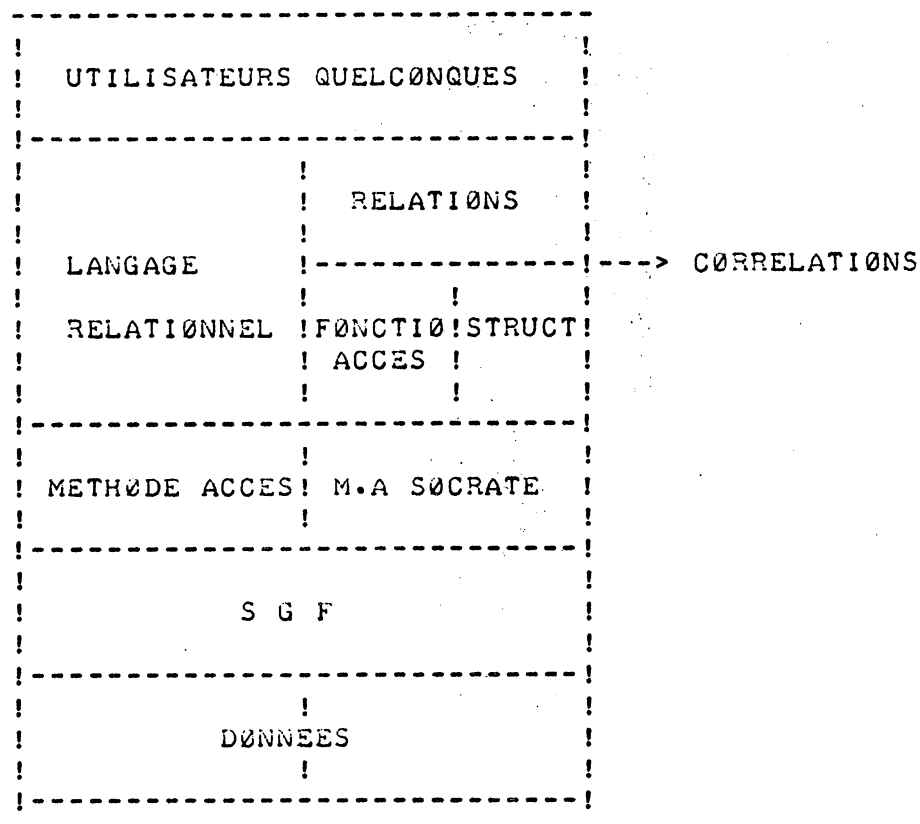


Fig. 3.3 - CØØPERATIØN RELATIØNNELLE - SØCRATE
****

Il sera en effet nécessaire de pouvoir interpréter et traduire les qualifications hiérarchiques fournies par les définitions des relations URANUS en termes de chemins d'accès Socrate. *On supposera évidemment que la structure Socrate est connue de l'utilisateur, le service fourni étant ici de lui éviter une programmation des fonctions d'accès lui délivrant les informations, et d'initialiser automatiquement les relations correspondantes.* Les fonctions de suppression et de modification lui seront accessibles de la même façon.

On crée donc pour toute relation comportant des éléments corrélés une relation "#Soc" accessible du seul système et comportant une image des qualifications décrites dans la relation (Fig. 3.4) :

- nom de la relation,
- nom des entités Socrate associées,
- nom des caractéristiques Socrate associées,
- qualification de ces caractéristiques dans la structure (niveau hiérarchique),
- informations de travail nécessaires aux opérateurs (X_i affectés aux entités, numéros de réalisation courants, etc ...).

Cette relation est interprétée par un module de l'analyseur du langage de définition, afin de générer un fichier contenant l'ensemble des fonctions d'accès Socrate, disponibles en langage Source. Ce fichier est à son tour soumis au système d'exploitation de façon à être compilé et catalogué par Socrate dans la base de données (Fig. 3.5).

La génération des opérateurs en langage source étant uniquement fonction des qualifications exprimées dans les définitions de relation toute erreur détectée par le compilateur Socrate provient d'une mauvaise connaissance de la structure de la base. Dans ces conditions, le palliatif sera soit de vérifier directement si les définitions de corrélations sont conformes au "fichier structure" Socrate, soit de rééditer les définitions après modifications.

Exemple : Table de qualification Socrate

```
EMPLOYE rel 500 idem PERSONNE de SOCIETE
   début
      NOM mot idem NAME
      SALAIRE de 0 à 1000 idem SALARY de JOB de PERIODE
      ETATCIV (MARIE CELIB DIV)
   fin
```

CITATION	NIVEAU	PERE	Xi	NUMBE
EMPLOYE	-1	2	0	6
PERSONNE	2	3	1	0
SOCIETE	1	0	2	0
NAME	2	2	0	0
SALARY	4	6	0	0
JOB	4	7	3	0
PERIODE	3	2	4	0

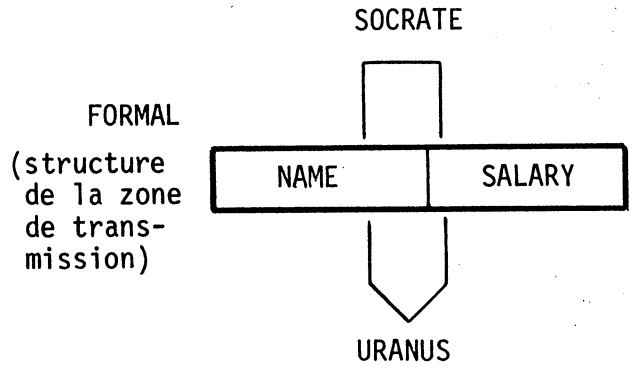
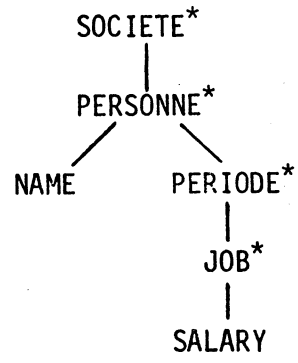


Fig. 3.4

Structure Socrate correspondante (les entités sont marquées d'un astérisque : *) :



La mise en oeuvre des opérateurs d'accès Socrate sera explicitement faite par la création des relations par l'usager, par la demande de sauvegarde d'un contexte de corrélations, précédée de la mise à jour des relations appropriées (§ 3.4).

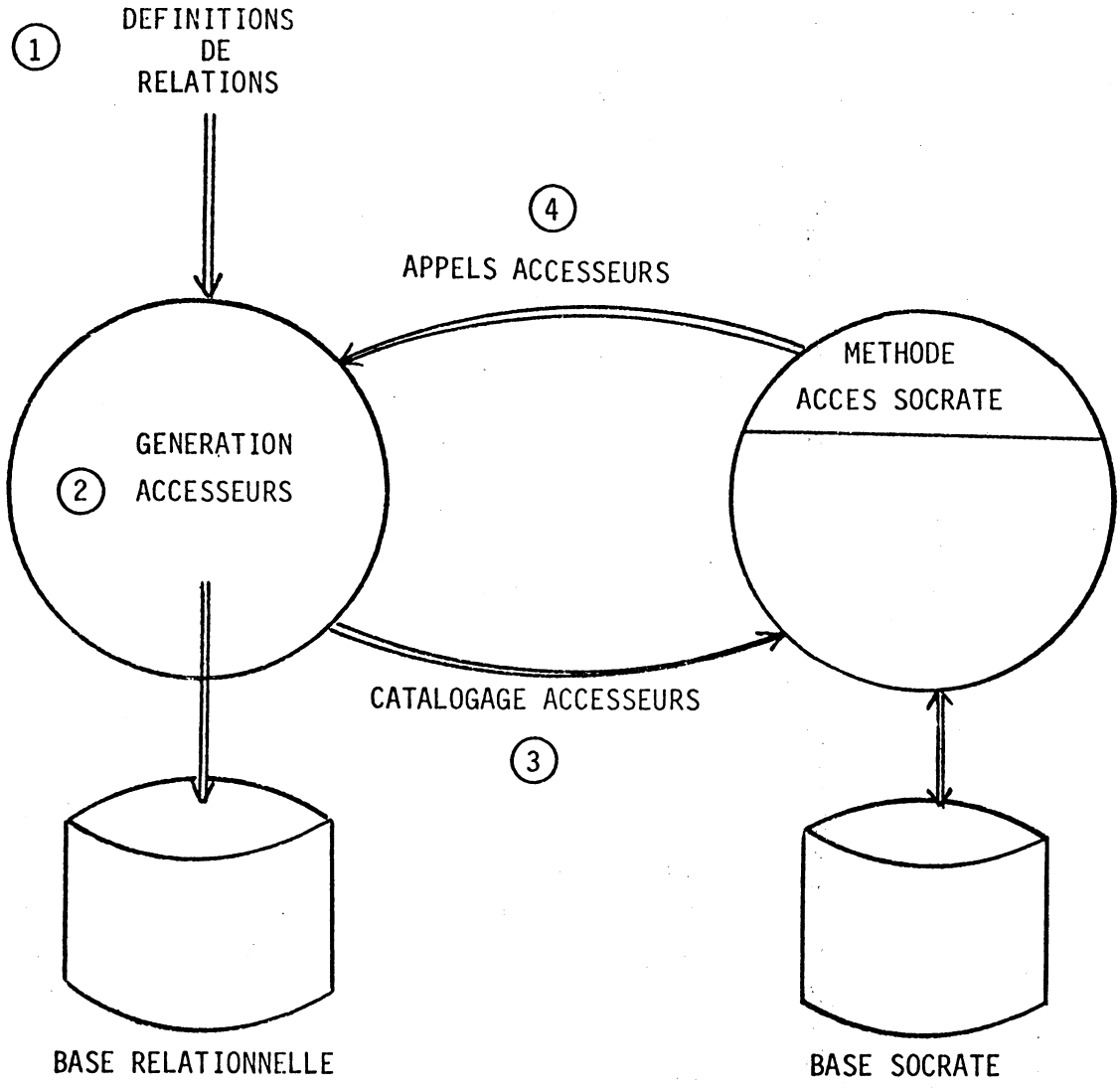


Fig. 3.5

De manière plus précise, le format de la table RSOC construite à l'analyse des définitions de relation pour mémoriser les corrélations est :

```
DCL 1 RSOC (NBR SOC) EXT CTL,
        2 XI FIXED BIN (15),        NO DE XI SOCRATE ALLOUE
        2 CITATION CHAR (16),       NOM D'ENTITE OU DE CARACTERISTIQUE
        2 NIV FIXED BIN (15),       NIVEAU DANS STRUCTURE
        2 PERE FIXED BIN (15),      PERE "    "
        2 NUMDE FIXED BIN (15) ;    NUMERO DE REALISATION COURANT
```

Pour chaque relation corrélée, un n-uplet de RSOC est généré avec :

- XI = -1
- CITATION = NOM DE RELATION
- NIV = 0
- PERE = INDICE DU 1er ELEMENT POUR CETTE RELATION dans RMAITRE
- NUMDE = NOMBRE D'ELEMENTS DE RSOC POUR CETTE RELATION.

De plus, une caractéristique a un XI nul, un NIV égal à celui de son père dans RSOC et un NUMDE nul. Une entité a un XI non nul, égal au XI Socrate qui le référencera dans le sous-programme Batch Socrate.

Exemple : Sur la relation suivante

```
PROFESSEUR REL 100 IDEM ENSEIGNANT
  début
  |   NOM MOT CLE IDEM NOM
  |   SALAIRE de 0 à 9999 IDEM INDICE
  |   COURS de 0 à 9999   IDEM CODE DE ENSEIGNEMENT
  |   TYPE DANS TYPE IDEM NATURE DE FRACTION DE ENSEIGNEMENT
  fin
```

correspondant à la structure Socrate suivante

```
entité 1000 ENSEIGNANT
  début
  |   NOM MOT
  |   INDICE de 0 à 9999
  |   entité de ENSEIGNEMENT
  |     début
  |     |   CODE de 0 à 999
  |     |   entité 10 FRACTION
  |     |     |     NATURE (TD, TP, Cours)
  |     |   fin
  |     fin
  fin
```

La table RSOC aura la configuration suivante :

XI	CITATION	NIV	PERE	NUMDE
-1	PROFESSEUR	RID	2	7
1	ENSEIGNANT	1	0	0
0	NOM	1	2	0
0	INDICE	1	2	0
0	CODE	2	6	0
2	ENSEIGNEMENT	2	2	0
0	NATURE	3	8	0
3	FRACTION	3	6	0

Cette table étant générée séquentiellement dans RSOC, les pères hiérarchiques d'un élément Socrate s'y trouvent dans l'ordre inverse à celui nécessaire à la génération des qualifications dans les accesseurs. Une procédure particulière est donc chargée de fournir ces ascendants en suivant la chaîne des pères.

La relation '#SOC' est construite à l'image de la table précédente, afin de pouvoir recréer les accesseurs si besoin en était. Lorsque ceci est fait, la table est détruite en fin de session.

Les corrélations permettent d'expliciter les liens structurels qui unissent des constituants de relations et des caractéristiques Socrate.

L'utilisation relationnelle de ces dernières dans URANUS nécessite cependant une vision uniforme compatible avec les exigences d'une manipulation ne faisant plus appel qu'à la notion de n-uplet.

Ceci est réalisé au moyen d'un intermédiaire formel : les "fenêtres relationnelles", définies sur une base Socrate.

3.3. FENETRES SUR UNE BASE SOCRATE

Les impératifs de la coopération de données d'origines diverses et de structurations différentes impliquent la définition d'un niveau de vision et de manipulation commun. L'ensemble minimum compatible entre tous étant trop restrictif, la confrontation des structures relationnelles avec celles des bases hiérarchiques et en réseau couramment utilisées autorise la définition d'un interface de haut niveau entre le système relationnel et les autres.

Il est, en effet, possible de concevoir l'objectif poursuivi comme une intégration de données "classiques" dans un ensemble plus général mettant en application les concepts relationnels : la coopération des bases découle de ce principe (chap. 4).

L'uniformisation de représentation des différentes informations extraites de bases quelconques facilite cette approche. On a ainsi été conduit à définir sur les bases Socrate utilisées des "vues" ou "fenêtres relationnelles".

Leur principe consiste à former une suite de n-uplets à partir des caractéristiques des entités sur lesquelles ont été définies les corrélations. Une fenêtre est ainsi la définition relationnelle d'un certain nombre de n-uplets, formés à partir d'un certain rang de réalisation d'entité Socrate (Fig. 3.6).

Exemple :

Fenêtres	Compositions
(1,100)	100 n-uplets formés à partir de l'entité de rang 1
(50,50)	50 n-uplets formés à partir de l'entité de rang 50
(30,60)	60 n-uplets formés à partir de l'entité de rang 30

Les entités imbriquées nous imposent une entorse au principe des relation normalisées. Ainsi sur l'exemple suivant :

```
MACHINE rel 500 idem MACHINE
  début
    |  PIECE de 0 à 99 idem PART
    |  EXECUTANT mot idem WORKER de CRASH
    |  DATE de 0 à 99 idem DATE de CRASH
  fin
```

```
entité 1000 MACHINE
  début
    |  PART de 0 à 999
    |  entité 1000 CRASH
    |    début
    |      |  WORKER
    |      |  DATE
    |    fin
  fin
```

le développement des entités MACHINE et CRASH conduit à des n-uplets où l'information PIECE correspondant à la caractéristique PART est répété autant de fois que l'entité CRASH est présente dans chaque entité englobante MACHINE. Ceci n'est gênant que par la redondance d'information que cela impose. En fait, nous sommes ici limités par les contraintes struc-

DEFINITION DES FENETRES SUR SOCRATE

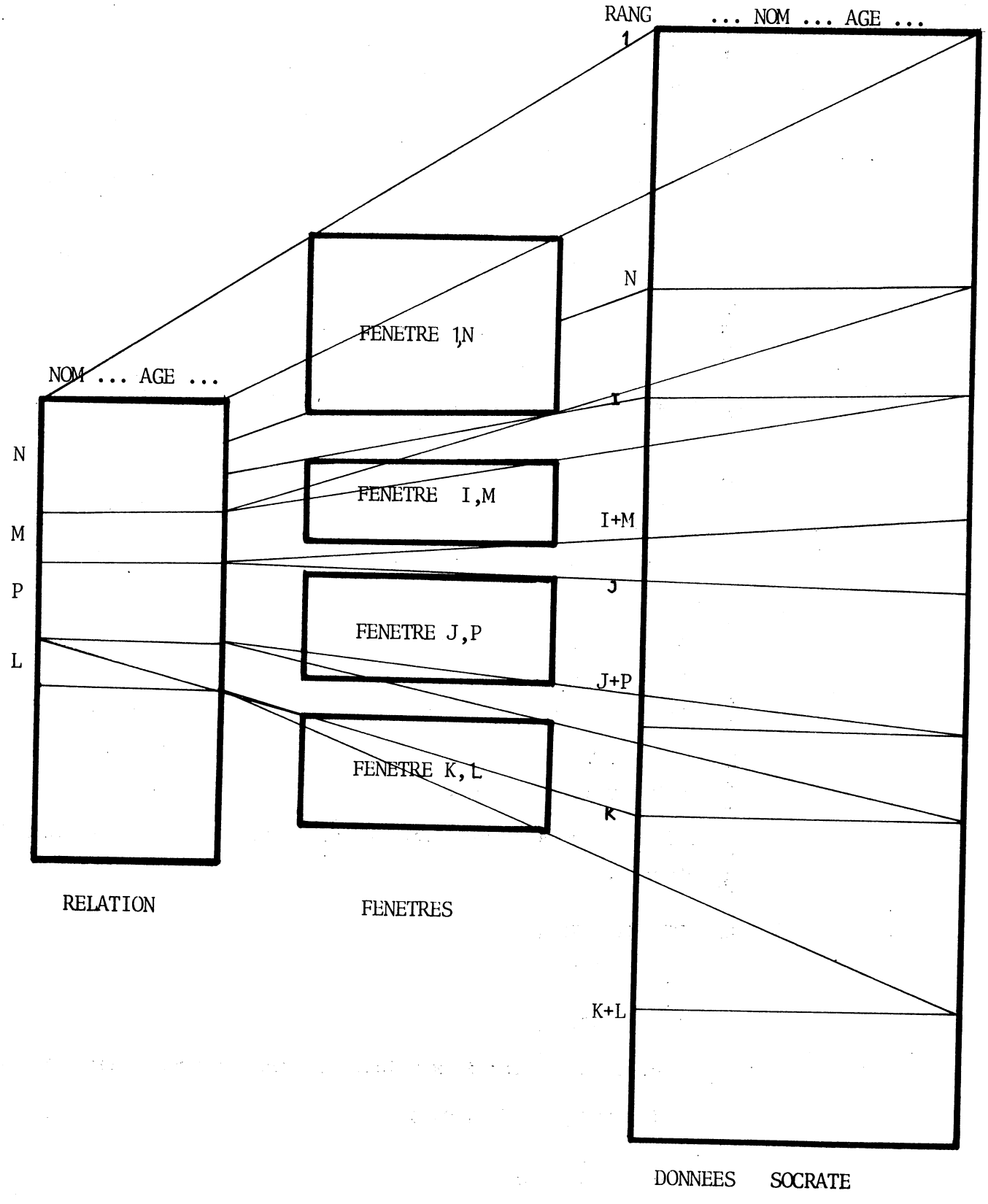


Fig. 3.6

turelles Socrate qu'on ne peut pas éviter à ce stade de manipulation.

Cette vision relationnelle de données Socrate est directement utilisable pour la définition (cf. § 2.5) et l'initialisation de relations, ainsi que pour la coopération de plusieurs bases (cf. Chap. 4). Elle constitue un moyen particulièrement simple de définition d'un interface entre des données de structures trop diverses pour pouvoir être simplement juxtaposées.

Les besoins de manipulation étant a priori aléatoires, les fenêtres doive être adaptées à toutes les circonstances de mise en oeuvre des applicatio Il est donc possible d'en faire varier la taille et l'origine. Une même application relationnelle peut donc mettre en oeuvre sur plusieurs classe d'entités un nombre quelconque d'entre elles, de longueurs et de rangs initiaux variables au cours d'une même session.

Exemple : Sur deux entités MACHINE et OUTILLAGE, il est possible de défin 3 fenêtres : (1,20), (101,10) et (114,5) sur la première, 2 sur la deuxiè (1,30) et (50,40) pour initialiser deux relations : MACHINE et OUTIL de la façon suivante :

```
GET   MACHINE, 1, 20 ;
GET   MACHINE, 101, 10 ;
GET   MACHINE, 114, 5 ;

GET   OUTIL, 1, 30 ;
GET   OUTIL, 50, 40 ;
```

Un traitement relationnel étant terminé, on répercute les modifications effectuées :

```
PUT MACHINE ;
 .       .
 .       .
 .       .
PUT   OUTIL ;
```

et on réinitialise les deux relations avec d'autres données Socrate :

```
GET   MACHINE, 201, 80 ;
 .
 .
 .
GET   OUTIL, 80, 10 ;
```

permettant la poursuite du traitement sur d'autres valeurs de la base.

3.4. ACCESSEURS SUR UNE BASE SOCRATE

Les vues relationnelles utilisées sur des données Socrate constituent un concept qui doit être exploité par des moyens appropriés. Etant destinés à uniformiser la vision d'éléments extérieurs à un système relationnel afin d'y être intégrés, leur mise en oeuvre fait intervenir deux notions :

1 - une manipulation "externe" et relationnelle par un utilisateur,
2 - un accès "interne" Socrate, impliquant l'extraction ou la reécriture des n-uplets dans une base.

Les accesseurs sont destinés à assurer la liaison entre ces deux aspects, relationnel d'une part, structure-dépendant de l'autre. Le premier concerne l'usage qui est fait de l'information. On fournit dans ce cadre deux possibilités :

1 - accès séquentiel de données Socrate dans une fenêtre donnée,
2 - accès sélectif sur critère donné, à l'intérieur d'une fenêtre.

L'accès interne est automatiquement mis en oeuvre par celui, séquentiel ou filtré, effectué au niveau relationnel. Il y a exécution des fonctions préalablement cataloguées (cf. § 3.1).

Il existe donc deux types d'opérateurs :

1 - séquentiels :
  - GET <relation>, <fenêtre>
  - PUT <relation>

2 - sélectifs :
  - READ <relation>, <filtre>, <fenêtre>
  - WRITE <relation>.

Dans tous les cas, <fenêtre> peut être omis ; il y a alors initialisation ou report depuis la première entité jusqu'à saturation de la relation ou épuisement des n-uplets.

L'appel à ces deux types d'opérateurs provoque de la part de l'interface URANUS-SOCRATE l'exécution des fonctions d'accès correspondantes dans Socrate :

1 - GET<relation>
  PUT<relation>
  DEL<relation>

2 - READ<relation>
WRITE<relation>
SUP<relation>

Ces six accesseurs sont générés automatiquement et de façon transparente dès la définition de relations corrélées avec des données Socrate. On trouvera en annexe 7.5 un exemple complet d'une telle génération.

Les fenêtres sont interprétées en transmettant sous forme de paramètres leur origine et leur hauteur aux accesseurs. Ceux-ci fournissent les données Socrate sous la forme de n-uplets conformes en structuration, en nombre ainsi qu'en origine aux définitions de relations et aux requêtes exprimées :

Exemple :

```
EMPLOYE rel 1000 idem PERSONNE dans SOC3
  début
  |   NOM mot cle idem NOM
  |   PRENOM mot idem PRENOM
  |   ETAT-CIVIL mot idem ETAT-CIVIL cle
  fin
READ EMPLOYE, ETAT-CIVIL = CELIBATAIRE, 20, 50 ;
```

Initialisation de la relation EMPLOYE avec 50 n-uplets répondant au filtre ETAT-CIVIL = 'CELIBATAIRE', formés à partir de l'entité PERSONNE de rang 20 dans la base SOC3.

A aucun moment l'utilisateur n'a à connaître l'existence des fonctions d'accès sous-jacentes. Elles sont générées de façon totalement transparente et mises en oeuvre par la méthode d'accès Socrate sur appel du système relationnel. *Il n'a plus à se préoccuper de leur écriture et donc de la connaissance d'un langage d'interrogation associé aux bases Socrate.*

Le seul interlocuteur est le système relationnel URANUS, délivrant des n-uplets provenant d'un nombre quelconque de bases, dans lesquelles on peut puiser, modifier ou supprimer des données.

Il suffit pour cela de manipuler des relations où ont été correctement définies des qualifications hiérarchiques correspondant aux imbrications d'entités Socrate.

On aboutit ce faisant à une simplification globale de mise en oeuvre de

bases de données, dont l'accès est uniformisé et les applications multipliées par la coopération qu'autorise les moyens relationnels.

3.5. ADJONCTION D'INFORMATIONS RELATIONNELLES

La définition des fenêtres sur les données appartenant à des bases non relationnelles et leur mise en oeuvre par les fonctions d'accès permettent de réaliser l'adaptabilité structurelle selon les deux axes définis précédemment :

1 - adjonction d'informations,
2 - modification des caractéristiques.

Moyennant un traitement relationnel de toutes les applications faisant intervenir de nouvelles données, celles-ci étant liées aux bases existantes par leur définition avec des données corrélées au sein des mêmes relations, l'extension de la nomenclature par adjonction est immédiate.

Il suffit pour cela de définir des constituants propres au système relationnel, conjointement à des constituants corrélés. Leur lien sémantique étant matérialisé par leur présence dans une même définition de relation, l'initialisation et la manipulation de ces éléments permet, à travers une vision commune des informations, d'étendre la structure des bases existantes et ainsi de rendre possibles de nouvelles applications.

Exemple :

Définition d'une relation PERSONNE dont le nom et l'adresse sont extraits d'une base Socrate SOC1, et dont le numéro insee est propre au système relationnel.

```
PERSONNE rel 100 idem PERSONNE dans SOC1
  début
    |   NOM mot idem NOM
    |   ADRESSE mot 80 idem ADRESSE
    |   INSEE de 100000 à 2999999
  fin
```

Il est clair que ceci trouvera un champ d'application très étendu avec la mise en commun d'informations provenant de plusieurs bases (Chap. 4). On peut en effet définir sous la forme de données ajoutées des informations

assurant le lien entre diverses classes d'informations éparpillées et mettre en oeuvre des traitements relationnels bâtis sur la connaissance de cette association logique, donc offrir de larges possibilités d' extension.

3.6. MODIFICATION DES CARACTERISTIQUES DE L'INFORMATION

Le problème de la modification des caractéristiques de l'information s'apparente de très près aux besoins généralement répertoriés sous le terme de "conversion de données". Certains systèmes récents ont abondamment utilisés les moyens relationnels pour y parvenir (Navathe) [38], [11]. Notre objectif a été différent dans la mesure où nous tendons à uniformiser les structures et les moyens de manipulation de données d'origines diverses. La nécessité de conversion de données dans URANUS s'est trouvée justifiée par cette constatation relativement simple : la définition des caractéristiques des données élémentaires dans Socrate ne souffrait aucune modification une fois la base de données structurée et initialisée. Il est apparu que ceci était imposé aux utilisateurs pour des raisons qui n'avaient rien à voir avec leur vision de l'information : la notion d'espace virtuel et l'organisation physique des fichiers de données [20].

Cette remarque apparemment anodine prend des proportions dramatiques si l'on songe aux coûts de refonte d'une base importante pour des pourcentages minimes d'information pourtant essentielle. A titre d'exemple, le fichier des étudiants inscrits aux trois universités grenobloises comporte 45 000 éléments de 1K octets environ. La modification du numéro insee à 16 chiffres implique seulement 1,5 % de l'ensemble, et est pourtant indispensable à la validité des informations.

Outre que l'on hésite toujours, lors de la définition d'une structure de données, à prévoir des caractéristiques surdimensionnées, à la fois pour des impératifs d'opportunité et d'encombrement des espaces physiques de mémorisation, toute solution prospective dans ce domaine ne peut être qu'aléatoire.

Nous introduisons donc la possibilité de modifier le type et les caractéristiques élémentaires (longueur des chaînes, bornes numériques) de façon dynamique et en fonction de chaque application, et non plus rendre a priori des dernières tributaires de ces éléments.

Ceci a été réalisé à la fois par des facilités au niveau du langage de manipulation (cf. § 3.7.1.7. et § 3.7.1.8.) et par la possibilité pour l'utilisateur de définir le format de transmission des données entre le système relationnel et les bases externes, à l'aide des "formal" [3] dans le cas de Socrate, et ceci pour chaque groupe d'accesseurs, donc pour chaque relation ayant des éléments corrélés.

Exemple :

Modification de toutes les caractéristiques élémentaires d'une entité VOITURE associée à la relation VEHICULE : le format est changé à l'extraction de la base (Formal U-VEHICULE) et à l'écriture dans celle-ci des données relationnelles modifiées.

```
entité 1000 VOITURE
  début
    | NUMERO mot 9
    | ANNEE de 0000 à 2999
    | PROPRIETAIRE mot 30
    | ADRESSE mot 80
    | PUISSANCE de 0 à 999
  fin

Formal U-VEHICULE
  début
    | NUMERO dilate 10
    | ANNEE mot 4
    | PROPRIETAIRE mot 40
    | ADRESSE mot 120
    | PUISSANCE mot 2
  fin

VEHICULE rel 500 idem VOITURE dans SOC1
  début
    | NUMERO mot 10 idem NUMERO
    | ANNEE mot 2 idem ANNEE
    | PUISSANCE mot 2 idem PUISSANCE
    | PROPRIO mot 160 idem PROPRIETAIRE
  fin
```

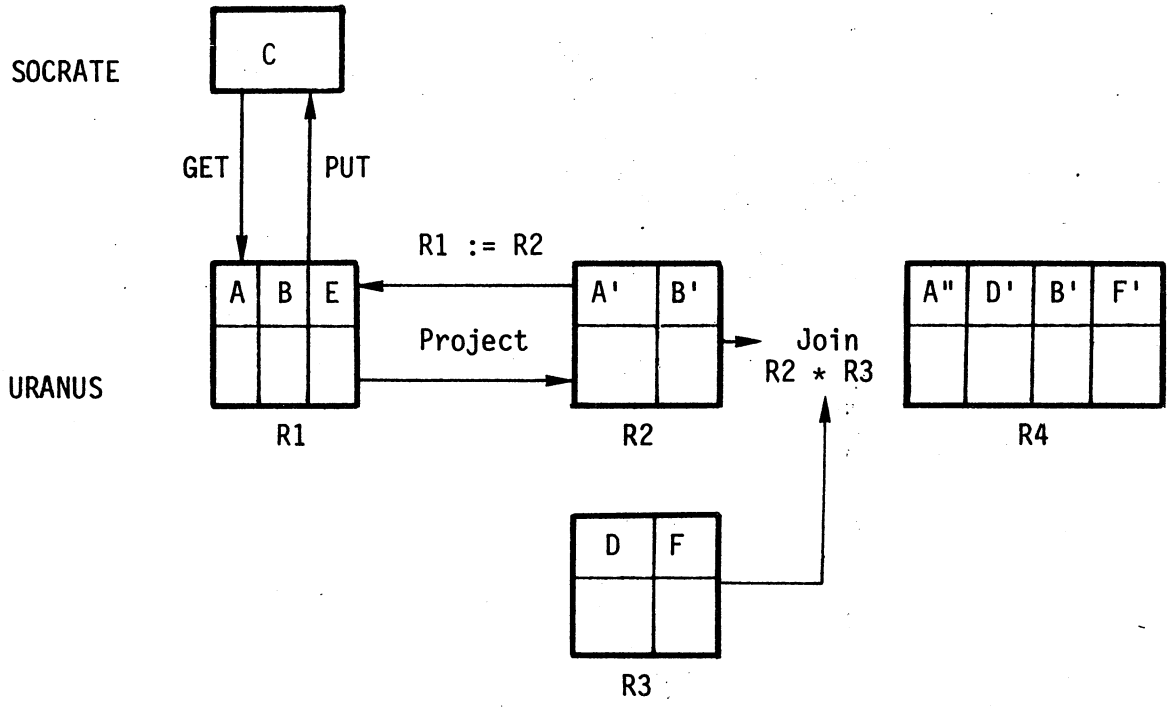


Fig. 3.8

3.7. MANIPULATION DE RELATIONS

La conception d'un interface relationnel nous permet d'envisager, par la mise en oeuvre d'opérateurs standards, d'uniformiser la manipulation des données, qu'elles soient propres au système relationnel ou associées aux bases Socrate. Cette simplification vise à atteindre la facilité de représentation de l'information et à alléger considérablement la charge de programmation liée aux traitements. Notre expérience des services administratifs nous pousse à concevoir pour la diffusion des informations et leur impact opérationnel un ensemble simple de manipulation, dégagé de toute considération technique de représentation de l'information, de règles d'écriture complexes, et qui, en fin de compte, puisse être décentralisé aisément.

On a donc réalisé un langage d'expressions relationnelles réduit [14], [15] procurant des facilités d'expression de problèmes complexes, par une grande possibilité de combinaison des opérateurs entre eux. Les objets manipulés seront des relations, leur traitement faisant appel

à des critères de sélection sur les constituants si nécessaire.

L'aspect uniformisé des traitements devra cependant tenir compte des associations faites avec des éléments provenant ou à répercuter dans les bases Socrate. Ceci nécessitera donc un certain nombre de fonctions annexes, intégrées au système de manipulation, permettant le va-et-vient des informations (cf. § 3.4). C'est ainsi que sera fait appel aux divers opérateurs d'accès sur les bases de données dont il a été question, et que seront mises en oeuvre les fonctions de l'interface.

Au niveau relationnel, on fournira les éléments nécessaires à la définition des relations (noms, constituants, types, domaines, clés) (cf. § 2.5), à l'expression des corrélations Socrate (entités et caractéristiques associées) (cf. § 3.2) et à la manipulation des relations (création, suppression, modification, insertion, combinaison). Associées aux opérations relationnelles et aux corrélations sont fournies des procédures de génération automatique d'opérateurs Socrate de (cf. § 3.4) :

- création de relations et d'initialisation de constituants à partir de données Socrate,
- modifications de données Socrate à partir des relations.

3.7.1. Eléments de base du langage

Les opérations de base sur les relations sont : création, suppression, mise à jour, composition. Sauf la dernière, elles peuvent comporter un aspect lié aux données provenant d'une base Socrate. Elles devront toutes pouvoir opérer des sélections d'information avant traitement proprement dit. La composition ou "equi-join" aura un aspect algorithmique plus important dû aux critères déterminant l'opération. Elles auront toutes au moins un argument objet : la relation sur laquelle elles doivent porter.

Les critères de sélection pour les mises à jour et les compositions devront permettre de déterminer les n-uplets répondant à certains filtres. On admettra donc les comparaisons alphanumériques sur les valeurs des constituants, (<, >, =, #, etc ...), les constantes immédiates alphanumériques, les conditions composées par & / ou. Ces critères de détermination constitueront le deuxième argument des fonctions. Afin d'étendre les possibilités de calcul numérique sur les relations, on admettra comme dans [14] et [15] les fonctions incorporées standards du type SOMME, MAXIMUM, MINIMUM et MOYENNE, portant sur l'ensemble des occurrences

d'un constituant, ainsi que les filtres UN et TOUS associés aux comparaisons numériques.

3.7.1.1. Création

INSERT (<relation>, <liste d'affectation>) ;

Ajout d'un n-uplet à une relation en fournissant les valeurs des divers constituants, mise à indéfini de ceux non cités.

Exemple : Création de n-uplets dans les relations définies de la façon suivante :

```
PERSONNE rel 500                VOITURE rel 1000
  début                           début
    |  NOM mot clé                 |  NUMERO de 0 à 9...9
    |  PRENOM mot                  |  MARQUE mot
    |  ADRESSE mot 80               |  PUISSANCE de 0 à 999
    |  INSEE de 0 à 299...9         |  PROPRIETAIRE mot
  fin                             fin
```

```
INSERT (VOITURE, NUMERO := 210, MARQUE := "BMW", PUISSANCE := 95,
        PROPRIET. := "DULAC") ;
INSERT (PERSONNE, NOM = "DULAC", PRENOM := "ALAIN") ;
```

ou plus simplement :

```
INSERT (PERSONNE, ?) ;
```

Les constituants non initialisés sont mis à la valeur "indéfini".

3.7.1.2. Mise à jour

MODIFY (<relation>, <condition>, <liste d'affectation>) ;

On change la valeur de certains constituants d'une relation répondant à certains critères.

Exemple : Sur les relations PERSONNE et VOITURE précédentes :

```
MODIFY (VOITURE, PROPRIET = "DULAC", NUMERO := 302, MARQUE :=
        "CITROEN", PUISSANCE := 80) ;
MODIFY (PERSONNE, NOM = "DULAC", ADRESSE := "10 R.LATOUR PARIS 12") ;
```

Les constituants non modifiés sont laissés à leur ancienne valeur.

3.7.1.3. Sélection

SELECT (<relation>, <condition>) ;

On crée une nouvelle relation avec les n-uplets d'une autre, répondant à certains critères.

Exemple : SELECT (VOITURE, PUISSANCE > 7) ;
SELECT (PERSONNE, PRENOM = "ALAIN") ;

3.7.1.4. Composition

JOIN (<relation1>,<relation2>,<domaine><op><domaine>) ;

On crée une relation avec la composition de deux autres, sur deux domaines compatibles répondant à certains critères. Les noms de constituants de la relation résultat seront, dans l'ordre des opérandes :
<domaine>.<relation1>,<domaine2>.<relation1>,...<domaine1>.<relation2>, etc.

Exemple : R1 := JOIN (VOITURE, PERSONNE, PROPRIET = NOM) ;
R2 := JOIN (SELECT (VOITURE, PUISSANCE > 7), PERSONNE, PROPRIET=NOM);

Création d'une relation comportant tous les constituants des relations VOITURE et PERSONNE dans cet ordre :
NUMERO.VOITURE, MARQUE.VOITURE, PUISSANCE.VOITURE, PROPRIET.VOITURE, NOM.PERSONNE, PRENOM.PERSONNE, etc ...
Pour éliminer certaines ambiguïtés ultérieures de traitement, il est possible de renommer des domaines issus d'un JOIN de la façon suivante :
R3 := JOIN (VOITURE, PERSONNE, PROPRIET = NOM) <V1,V2,V3,V4,P1,P2...> ;
où les V_i et P_i sont des noms quelconques associés un par un avec les constituants de R3.

3.7.1.5. Projection

PROJECT (<relation>, <liste domaine>) ;

Exemple : 1) Pour toute VOITURE de PUISSANCE comprise entre 5 et 9 CV, nom, adresse du propriétaire et numéro de la voiture :

```
PROJECT (JOIN (SELECT (VOITURE, ε(PUISSANCE >= 5, PUISSANCE < 9)),
                      PERSONNE,
                      PROPRIETAIRE = NOM),
                                   NUMERO.VOITURE, NOM.PERSONNE,
                                   ADRESSE.PERSONNE) ;
```

2) nom de tous les propriétaires ayant une voiture CITROEN :
PROJECT (SELECT (VOITURE, MARQUE = "CITROEN"), PROPRIET) ;

3) numéro de toutes les voitures de puissance ≥ 8 cv :
PROJECT (SELECT (VOITURE, PUISSANCE >= 8), NUMERO) ;

4) numéro et marque de toutes les voitures :
PROJECT (VOITURE, NUMERO, MARQUE) ;

3.7.1.6. Création de relation

<identificateur> := <relation> | <opération>

On crée une relation à partir d'une autre, par simple transfert d'information ou par l'une des cinq opérations précédentes (Exemple § 3.7.1.4.).

Dans chaque cas, le terme <relation> désigne soit un identificateur préalablement initialisé par une affectation, soit une combinaison quelconque des quatre premières opérations. On notera que c'est grâce à cette récursivité des définitions qu'il est possible avec seulement quatre opérations de base d'écrire des requêtes de type relationnel complexes. Il est vrai que si cela permet une interprétation simple, la résolution de certaines questions peut s'avérer difficile à mettre en forme, au moins dans son aspect externe, car elle répond à une logique très différente dans son essence de la programmation classique procédurale.

Certains problèmes pratiques doivent par ailleurs être résolus, tels que la redéfinition de n-uplets répondant aux critères de sélection de filtres inclus (d'où la notion de variable libre introduite pour la circonstance dans SEQUEL et reprise ici [15]).

3.7.1.7. Affectation par défaut

Lorsqu'une relation R1 n'a pas fait l'objet d'une définition explicite, une opération du type :

R1 := R2 ;

crée la relation R1 dont toutes les caractéristiques (cardinal, degré, nom et type des constituants) sont celles de R2. Si R2 est elle-même une combinaison d'opérations élémentaires, R1 aura les caractéristiques de la relation résultat (Ex. 3.7.1.4. et suivants).

3.7.1.8. Ajout de constituants à une relation

Cette opération s'effectue par affection à une nouvelle relation, définie explicitement, des constituants d'une relation existante. Les correspondances sont faites par noms de constituant identiques. Il est donc possible de redéfinir une relation avec un nombre quelconque de domaines supplémentaires ou en moins. Les caractéristiques (bornes, longueur) et type de constituants (numérique ou chaîne de caractères) peuvent être modifiés.

Exemple :

```
EMPLOYE rel 1000 idem ...
début
    NOM mot 20 idem ...
    ADRESSE mot 60
    AGE de 0 à 100
fin
SALARIE := EMPLOYE ;
```

```
SALARIE rel 10.000
début
    NOM mot 30
    ADRESSE mot 80
    AGE mot 3
    PROFESSION mot
    ETAT-CIVIL mot
fin
```

Les constituants PROFESSION et ETAT-CIVIL sont ajoutés à la relation EMPLOYE dont on a modifié les caractéristiques initiales. Elle n'est pas détruite lors de cette opération.
Le degré de la nouvelle relation SALARIE peut être différent de celui de l'ancienne. Les corrélations Socrate ne sont pas répercutées d'une relation à l'autre.

Notons que la nature même des systèmes relationnels, et d'URANUS en particulier, nous permet de nous affranchir des contraintes structurelles qui constituaient jusqu'à présent une source de blocage à l'évolution des systèmes d'information.

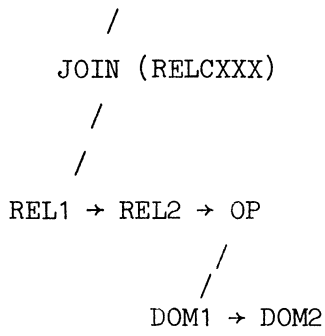
En effet, s'il est parfois possible de rajouter ultérieurement des entités à une structure Socrate figée afin de pallier l'insuffisance antérieure de caractéristiques, les temps de traitement peuvent en être sérieusement perturbés.

Par contre, la création d'une nouvelle relation ou le rajout d'un nouveau constituant à une relation sont des opérations ponctuelles permettant d'intégrer rapidement de nouveaux éléments, et d'ingérer en douceur l'évolution des données avec le temps.

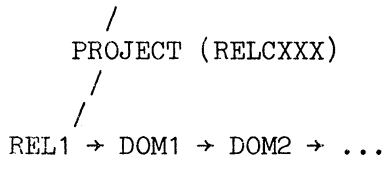
3.7.2. Interprétation du langage

Toutes les requêtes sont interprétées de façon arborescente. On donne ci-dessous la forme du sous-arbre correspondant à chaque opération élémentaire. Dans les 3 premiers cas, REL, REL1 et REL2 désignent soit un identificateur de relation, soit une combinaison des opérations, donc un sous-arbre composé d'un nombre quelconque des éléments décrits.

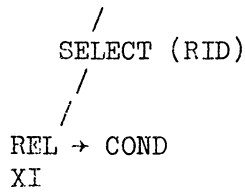
A) JOIN



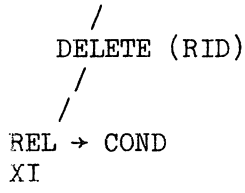
B) PROJECT



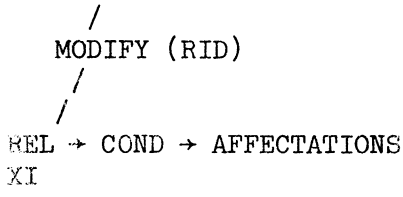
C) SELECT



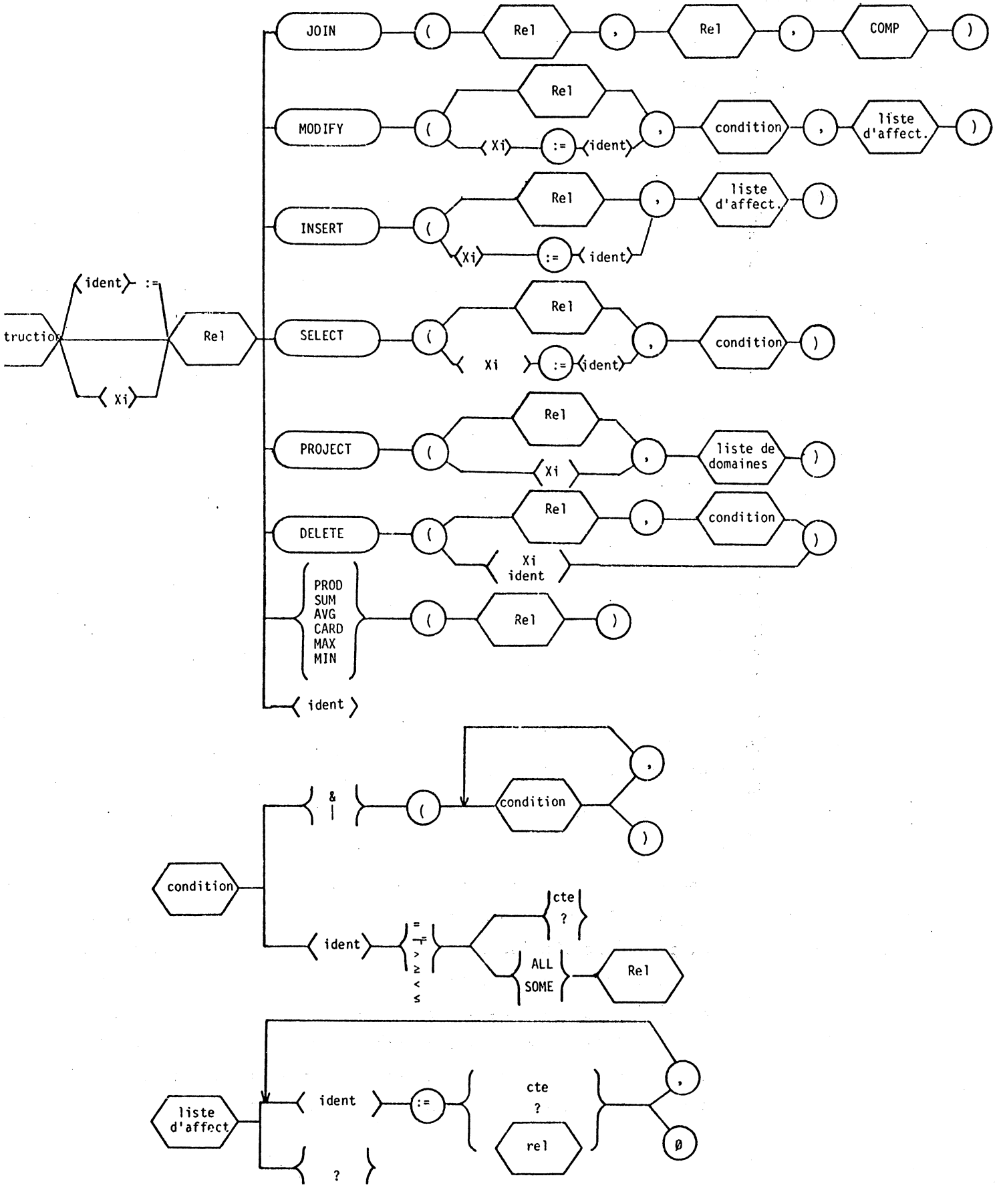
D) DELETE



E) MODIFY



Un diagramme schématique de la grammaire du langage de manipulation est donné ci-dessous :



3.8. MISE A JOUR DES DONNEES SOCRATE

La modification des caractéristiques Socrate à partir des mises à jour effectuées sur les relations URANUS est possible parce que l'on mémorise les opérations altérant les constituants corrélés.

Sur un plan théorique, ceci est très restrictif dans la mesure où l'on ne sait pas exprimer l'appartenance de constituants pris dans plusieurs relations au même domaine de valeur.

En effet, l'extraction de données Socrate ne pose aucun problème particulier. Il est évident cependant que la combinaison des relations entre elles et les opérations élémentaires du type : "création d'une relation par projection selon un ou plusieurs constituants d'une autre", impliquent un cheminement de l'information que dans l'état actuel des réalisations il serait très difficile de suivre, car il y a recopie systématique des informations pour chaque relation.

Nous avons donc imposé de ne répercuter les modifications de données dans les bases Socrate qu'à partir des relations où sont définies les corrélations.

Ainsi, un constituant A d'une relation R1 peut-il être modifié à loisir. La caractéristique Socrate C qui lui est associée peut l'être à son tour (Fig. 3.8 page 51).

Si d'aventure, ce constituant A est utilisé pour former une nouvelle relation R2, et former ainsi le constituant A', aucune mise à jour directe de C ne pourra être faite à partir de A'.

Ceci parce que la création de R2 grâce à R1, donc de A' grâce à A, ne suppose pas la mémorisation de la corrélation initiale A-C dans R2.

Il en sera de même pour toute utilisation ultérieure de A', par exemple création de : R4 := JOIN (R2, R3, A' = D).

La seule possibilité que l'on autorise est de reporter les valeurs modifiée de R2 et R4 dans R1, puis de mettre à jour C avec les nouvelles valeurs de A. Ceci peut être réalisé par simple affectation R1 := R2, si l'on a pris la précaution de renommer A'. Mais on a jugé plus simple à réaliser un système où les contraintes de traitement imposent la description explicite des modifications des informations extraites des bases Socrate, ceci permettant par ailleurs d'effectuer au niveau relationnel des simulations dans la perspective des systèmes d'aide à la décision.

CHAPITRE 4

COOPERATION MULTI-BASES

4.1. RAPPROCHEMENT D'INFORMATIONS

Les besoins des services de scolarité de l'Université pour une gestion intégrée de traitements associés aux diverses tâches administratives furent à l'origine du présent travail [16].

L'existence de plusieurs types de fichiers, de traitements spécifiques à chaque catégorie d'information (fichier d'étudiants séquentiel traité en Cobol et PL/1, fichier des enseignements sous forme de base de données Socrate [17]) nécessitait un outil de juxtaposition et de confrontation suffisamment général pour conserver les moyens opérationnels et permettre des développements futurs plus élaborés (création de mandatement des bourses automatisé, inscription directe des étudiants sur terminal, etc ...) (Fig. 4.1).

La coopération multi-bases sous son aspect relationnel, rendue opérationnelle avec URANUS, répond à ce souci de rapprochement d'informations extraites de fichiers mis en oeuvre à des époques différentes, selon des techniques diverses et appelées à être utilisées conjointement.

Nous pensons que leur intégration dans un cadre plus général, ne remettant pas en cause des chaînes de traitement opérationnelles, bien rôdées et utilisées sur une grande échelle, permet une évolution compatible avec les impératifs d'une gestion quotidienne.

Il est en effet probable que les principaux obstacles à l'introduction de techniques évoluées dans les services administratifs tiennent non pas à des problèmes de coût, mais à des contraintes de formation du personnel et de résistance psychologique. Le souci de maintenir en l'état les applications existantes, tout en permettant leur évolution, selon les besoins exprimés par les utilisateurs eux-mêmes, semble rendre ces inconvénients plus acceptables que par le passé.

La coopération des divers moyens cités précédemment étant rendue parfois difficile en raison des incompatibilités des structures d'information mises en jeu, l'unité de vue apportée par les relations créées à l'image des données existantes, et uniquement en fonction des nécessités d'une

application, évite la prolifération d'interfaces particuliers correspondant à chaque type de programme (création d'enseignements, de cursus d'étudiants, de résultats d'examens, etc ...).

En ce sens, la suppression d'un nombre considérable de fichiers intermédiaires et l'uniformisation de représentation relationnelle de toutes les données apporte un progrès considérable dans l'allègement des traitements.

Dans une hypothèse plus vaste de travail, nous avons étudié et réalisé les moyens d'une coopération de bases de données multiples.

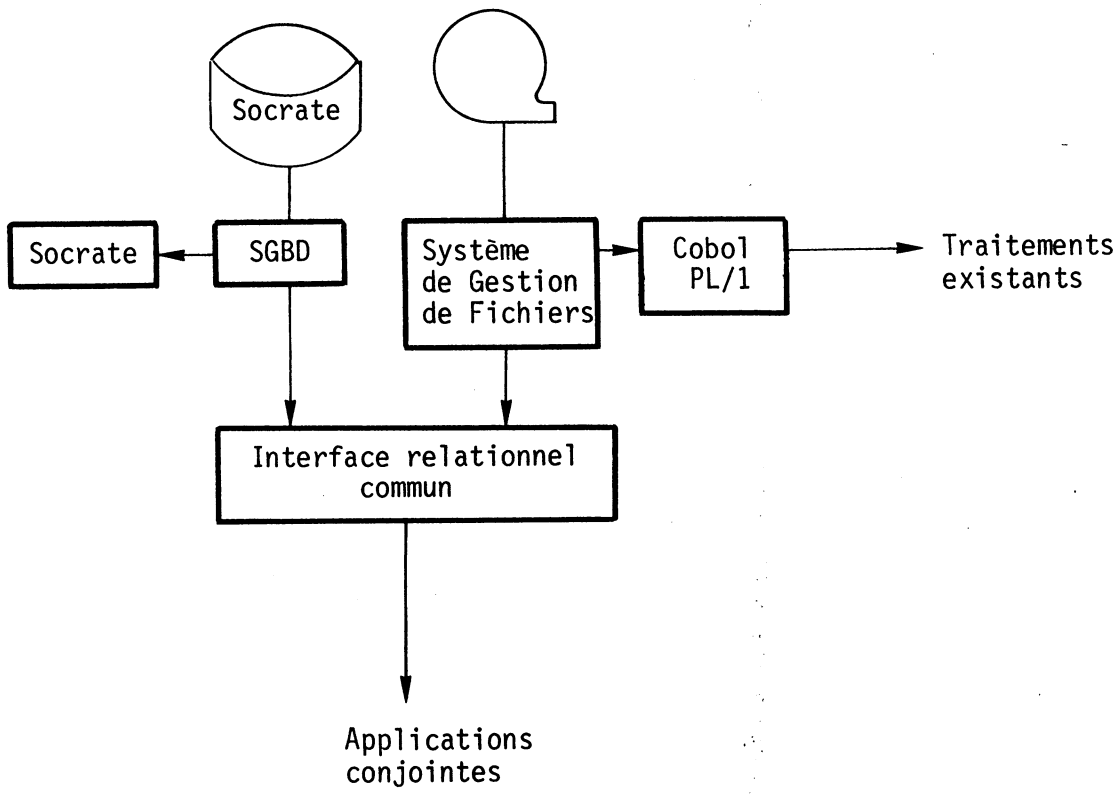


Fig. 4.1. - Mise en oeuvre d'applications relationnelles sur des données d'origines diverses.

4.2. MULTI-SOCRATE

Les principes mono-base exposés au chapitre précédent sont directement adaptables à un environnement multi-Socrate.

Comme précédemment, la connaissance des structures de données est un préalable indispensable à la bonne constitution des relations et à l'expression correcte des corrélations. De la même façon, une relation ne peut être associée qu'à une seule base de données, toutes les structures étant alors considérées comme parallèles entre elles (Fig. 4.2).

Cette hypothèse est la seule qui soit nécessaire à leur utilisation. Les paramètres d'accès à l'exécution sont simplement les noms des divers fichiers Socrate : source, structure et espace réel [20].

On établit ainsi les fondements d'une coopération relationnelle de bases de données physiquement indépendantes.

Le rapprochement d'informations sémantiquement liées donne alors la possibilité de créer de nouvelles applications jusqu'alors impossibles à mettre en oeuvre aisément (cf. Annexe 7.7).

On notera que la juxtaposition des entités ainsi réalisée à travers les relations permet, grâce au langage de manipulation relationnel, d'associer des données non seulement par classes ou catégories entières - étudiants, enseignements, ... -, mais d'effectuer toutes opérations de sélection sur des critères appropriés par l'intermédiaire des opérateurs Socrate (§ 3.4).

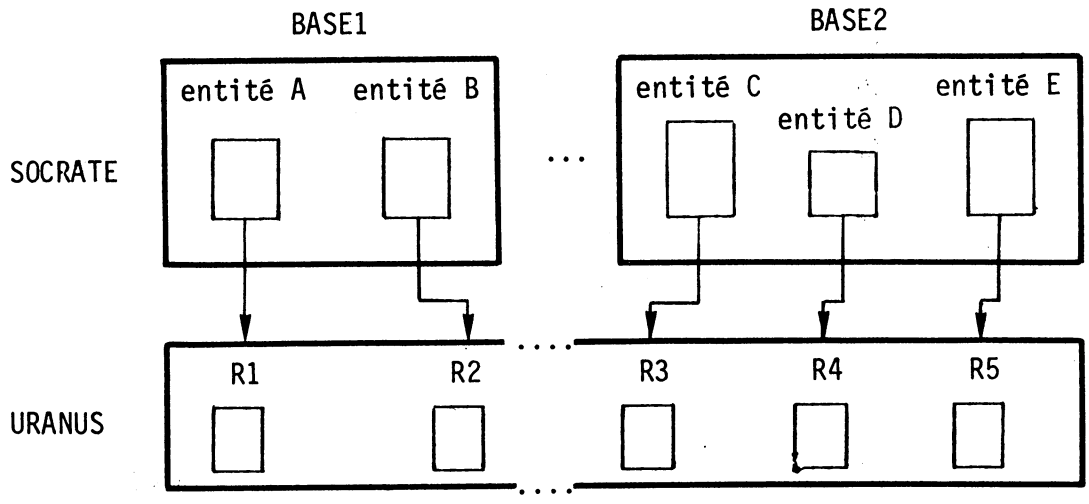


Fig. 4.2

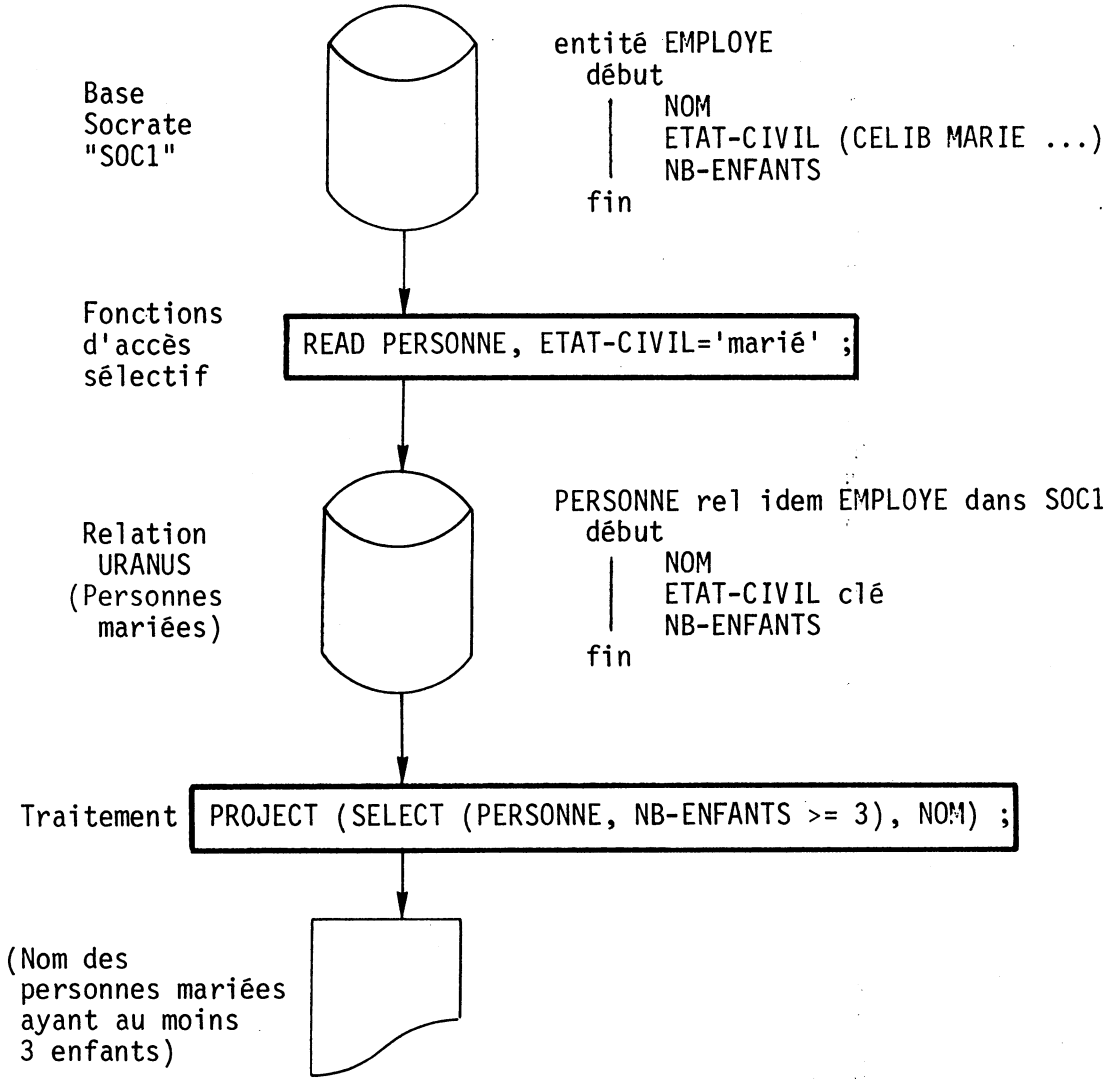


Fig. 4.3 - Juxtaposition des moyens de sélection des informations

On réalise ainsi un moyen de filtrage, permettant l'association de valeurs et non plus seulement d'objets, représentés par les caractéristiques Socrate. Ceci, grâce à la présence des fonctions d'accès sélectif sur les bases, semble un complément utile à la réalisation d'outils d'adaptabilité (Fig. 4.3).

En ce qui concerne URANUS, on ne réalisera pas de décomposition de requêtes mettant en jeu des données d'origines multiples, ainsi que cela est fait par ailleurs [18], [19]. Ceci pour trois raisons :

- la décomposition nécessite des moyens de synchronisation des réponses et de sélection des supports d'informations lourds à mettre en oeuvre,
- la nécessité de moyens simples et performants nous paraît primordiale dansla perspective d'applications concrètes,
- les concepts d'association des relations aux entités - pas plus d'une base et d'une hiérarchie d'entités imbriquées par relation - nous en préservent.

Les problèmes soulevés par la coopération multi-bases seraient simplifiés si Socrate était lui-même multi-bases de façon transparente. Ceci ne résoudrait cependant pas le problème dans le cas d'autres types de SGBD, ou dans un environnement réparti. Jusqu'ici nous avons montré sur un cas particulier qu'il est possible de construire un interface de haut niveau qui soit adapté à nos exigences.

Il nous semble utile de préciser que l'interface multi-bases s'adapte aussi bien aux données Socrate qu'aux données relationnelles. En ce sens qu'il serait possible d'envisager une coopération entre plusieurs bases Socrate et plusieurs bases de type URANUS, l'extension à d'autres types de systèmes restant soumise à la réalisation des moyens appropriés.
La discussion selon laquelle les applications ne doivent pas être perturbées par les changements d'organisation physique des informations est ici évitée par une vision uniforme d'espaces physiques disparates [9].

4.3. EXPRESSION DES LIENS SEMANTIQUES

La coopération multi-bases trouve son champ d'application dans la définition de nouveaux traitements mettant en jeu des informations associables d'un point de vue logique.

On doit remarquer que la définition des constituants et des corrélations Socrate telles qu'elles existent dans URANUS ne fournissent pas la possibilité d'exprimer les associations de ceux-ci à travers diverses relations. Ceci est sans doute la restriction majeure du système tel qu'il se présente à l'heure actuelle, car chaque constituant est représenté par son domaine de valeurs (Fig. 4.4).

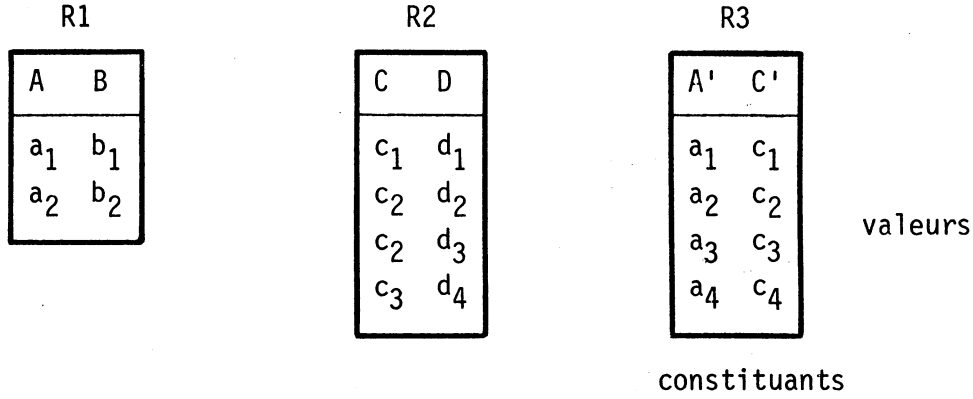


Fig. 4.4 - Relations où les constituants sont représentés par leurs domaines de valeurs

La confrontation de données provenant de diverses bases et de domaines relationnels étant le fondement de la coopération développée ici (cf. § 1.2), il serait sans doute souhaitable de pouvoir intégrer dans la définition même des informations leurs rapports sémantiques réciproques.

Ceci ne peut être réalisé de façon systématique qu'au moyen de schémas conceptuels et de l'implémentation de constituants puisant leurs valeurs dans des domaines définis indépendamment (Fig. 4.5).

On en a exclu ici l'usage pour des raisons évoquées précédemment [16].

Le système n'aura donc un comportement cohérent - selon la logique d'une application - que dans la mesure où l'usager associera correctement dans ses requêtes des informations appartenant à diverses relations et tenant compte de contraintes qu'il aura su expliciter dans celles-ci.

On touche ici un point qui sera détaillé dans les extensions présentées au chapitre 6, et qui pourrait être un axe de développement possible vers une adaptabilité logique plus évoluée que celle réalisée jusqu'à présent.

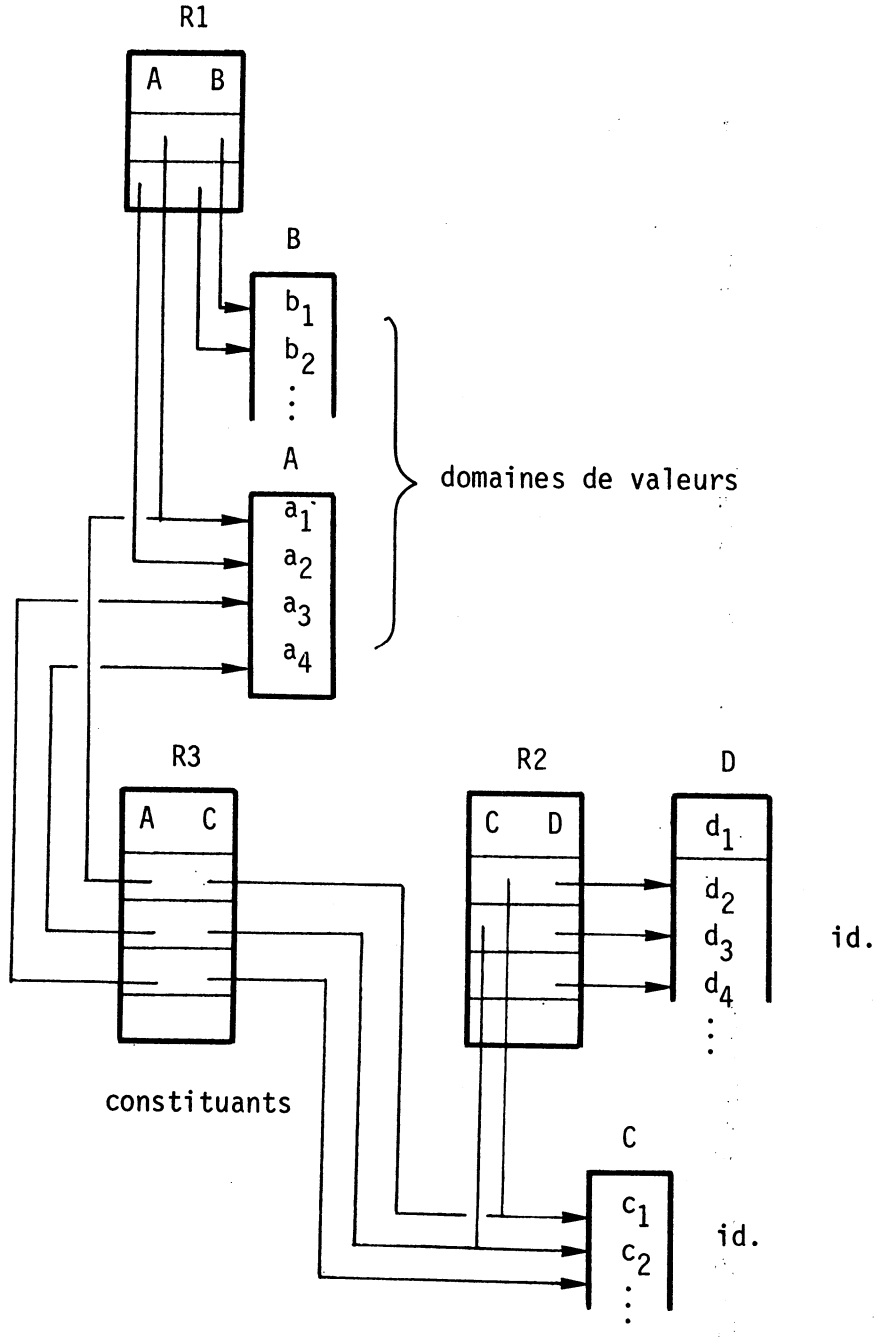


Fig. 4.5 - Relations où les constituants prennent leurs valeurs dans des domaines spécifiques.

4.4. MISE EN OEUVRE

L'environnement multi-bases peut être réalisé moyennant une description des bases Socrate utilisées. Elle se compose essentiellement des étiquett logiques des fichiers correspondants, ainsi que de leurs noms.

On associe aux diverses bases utilisées des noms symboliques, fournis par l'utilisateur. Ils permettent de les repérer de façon unique et de déterminer les fichiers qui les constituent.

Dans notre cas, on impose les étiquettes logiques suivantes [20] :

- STRX : fichier structure Socrate,
- SOUX : fichier source,
- ESPX : fichier espace réel.

Le caractère "X" étant remplacé par le numéro de n-uplet descripteur de l base dans la relation '#BASE', elles serviront à la génération des access

Ainsi, une définition de la forme :

ENSEIGNANT rel 5000 IDEM TEACHER dans SOC2

signifie que les données Socrate doivent être prises dans la base de nom symbolique "SOC2". Un utilisateur doit donc toujours prendre le même nom pour une même base Socrate. Ceci permet de retrouver son descripteur dans une relation système particulière "#BASE" dont les constituants sont

```
BASE REL 10
  début
    | #STRUCT  MOT 8
    | #SOURCE  MOT 8
    | #ESPR    MOT 8
    | #SUSP    MOT 8
    | #SSTRUCT MOT 12
    | #USER    MOT 8
    | #ACCNO   MOT 4
    | #PASSW   MOT 8
    | #NOM     MOT 4
  fin
```

Les quatre premiers désignent les noms des fichiers Socrate cités précédemment, les quatre suivants les paramètres nécessaires au "login" Socrate pour le cataloguage des accesseurs et la table d'interface Batch utilisée dans la méthode d'accès Uranus-Socrate. Le dernier, le nom symbolique de la base.

Les paramètres de celle-ci sont donnés par l'usager. Ils sont demandés de façon conversationnelle immédiatement derrière la définition de relation correspondante, de la façon suivante :

*** Paramètres de SOCX : (STRUCT, SOURCE, ESPR, SUSP, SSTRUCT, USER, ACCNO) ?

et l'utilisateur doit y répondre en donnant les éléments demandés dans cet ordre, séparés par une virgule, par exemple :

→ STR1, SOURCE1, ESPR1, SUSP1, : STRUCT, XYZ, 1234, WWW ;

On autorise actuellement la coopération de 10 bases différentes entre elles.

Afin de répondre au besoin de la méthode d'accès URANUS-SOCRATE, un certain nombre de pointeurs sont mis à jour entre :

- la relation en cours de définition,
- la relation "#BASE",
- la relation "#SOC".

Ainsi la relation en cours de définition comporte dans le champ "RINV" le numéro de base Socrate associé, et dans le champ "RDINV" le début de la description des corrélation dans "#SOC".

D'autre part, le champ "NIV" de la relation "#SOC" comporte dans son en-tête le RELID - ou numéro d'ordre dans "#MAITRE" - de la relation.

Exemple : L'utilisateur ayant défini une relation 'VOITURE' de RELID 23, dont certains éléments sont corrélés avec des données Socrate appartenant à la base numéro 2 (de paramètres STR2, ...).

La génération des informations correspondantes dans "#SOC" début au n-uplet 8 de celle-ci. Le champ "NIV" contient le RELID 23 de 'VOITURE', soit (Fig. 4.6) :

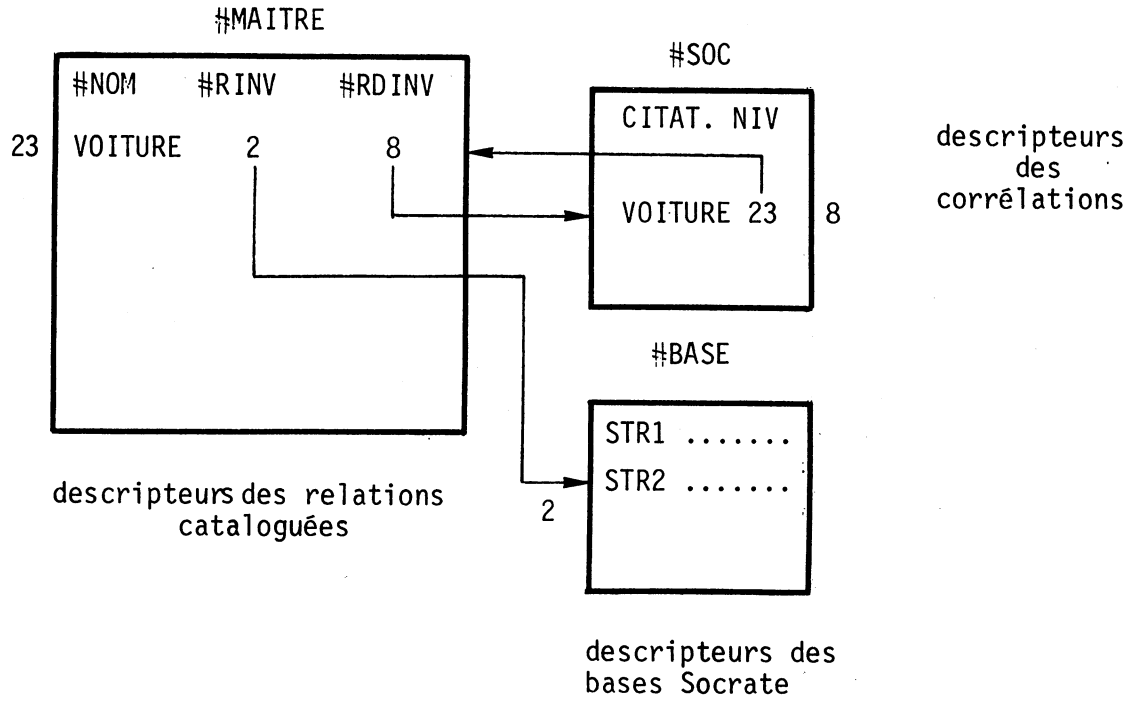


Fig. 4.6 - Liens entre les informations décrivant une relation corrélée

CHAPITRE 5

REGLES D'EVOLUTION

5.1. ADAPTABILITE LOGIQUE

La coopération multi-bases permet le rapprochement d'entités Socrate et de relations en nombre quasiment illimité.

Ceci représente un aspect de l'adaptabilité des bases de données, lié à l'évolution des structures de l'information. Il doit être complété [36] par un volet *sémantique* (§ 1.2), dans la mesure où une entité et une relation n'ont de signification qu'implicite. Les associations mises en évidence par les corrélations ne donnent aucune indication quant aux règles de cohérence qui s'appliquent aux données correspondantes : on perd en effet au niveau relationnel la logique des programmes d'application existant par ailleurs.

Dans URANUS, le langage de manipulation ne permet son expression qu'en surchargeant chaque requête à l'aide des filtres de sélection (cf. § 4.2).

On rappelle à ce sujet que nous ne faisons aucune hypothèse sur les relations constituées à partir des données Socrate, et qu'en particulier les associations qu'elles matérialisent peuvent n'avoir aucun sens connu - si ce n'est de la personne les ayant constituées. Ce point est en retrait par rapport aux modèles basés sur les schémas conceptuels [6], [7], [8]. Il permet cependant de ne définir de relations qu'en fonction des besoins du moment et de ne pas imposer la création et la sélection de chemins d'accès a priori (cf. les "STRING PATHS" de DIAM II par exemple [21], les "fonctions d'accès" de MOGADOR [27]).

On est donc amené à compléter cet aspect de "classification" structurelle des données par un ensemble de moyens logiques de "sélection", en introduisant des règles d'utilisation des informations (Fig. 5.1), restreignant les possibilités de modification sur celles-ci.

Les structures Socrate ne donnent en effet aucun moyen de contrôle des opérations effectuées par un utilisateur, autre que le partage de sous-structures et l'interdiction de certaines manipulations sur *tous* les éléments de celles-ci [3].

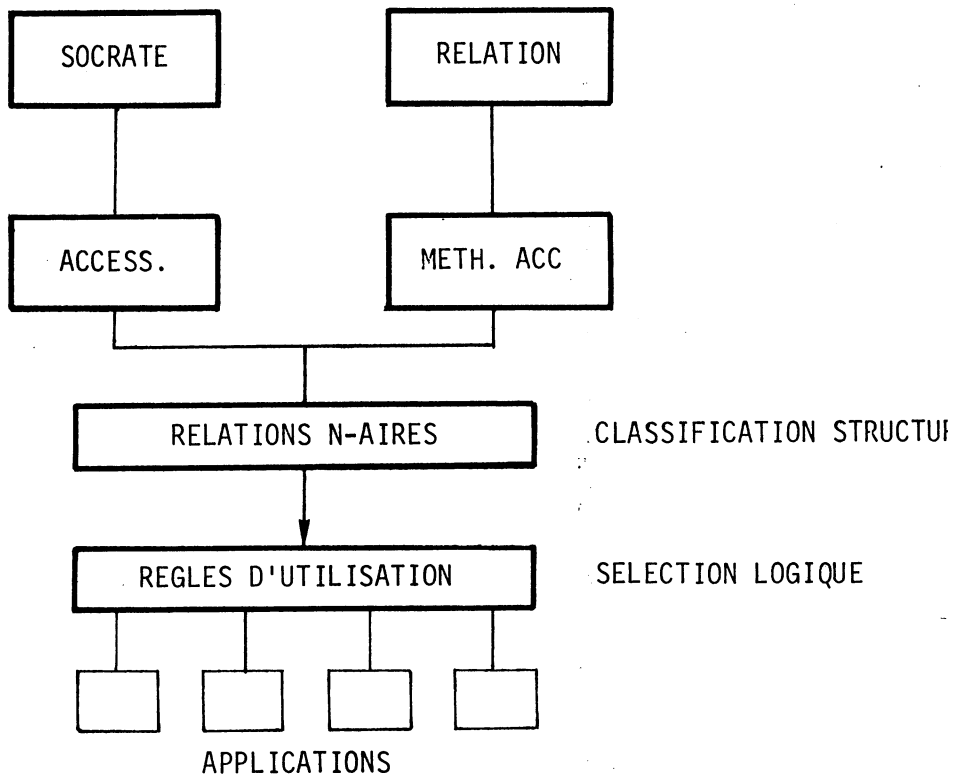


Fig. 5.1 - Coopération relationnelle et règles d'évolution

La nécessité d'effectuer ces contrôles sur les éléments accédés à l'inté d'une même classe d'objets ou d'entités en fonction de leurs valeurs est maintenant largement reconnue [6], [21], [22], et revêt une utilité prat non négligeable.

Ceci peut être généralisé (Fig. 5.2) par l'édiction de "règles d'évoluti [5] dépassant le cadre des opérations élémentaires et intégrant de plus des règles de cohérence, permettant d'assurer un contrôle sémantique des actions entreprises par les usagers. Leur définition indépendante des applications et de la représentation des données leur assure une grande souplesse d'utilisation, impossible à réaliser jusqu'ici en raison de la dichotomie imposée entre l'expression des structures de l'information et les traitements associés.

La possibilité de créer, modifier, supprimer ou masquer temporairement de telles contraintes fournit un outil d'adaptabilité très versatile, conforme à l'évolution des applications et à la prise en compte de facte nouveaux modifiant les normes d'utilisation des informations.

Ceci est actuellement en cours de réalisation sur URANUS.

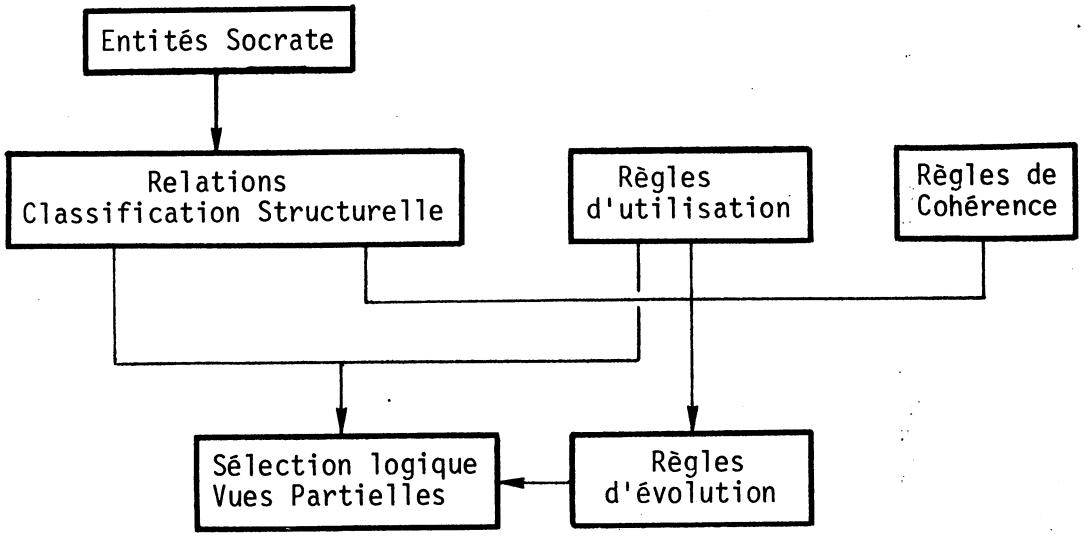


Fig. 5.2 - Classification structurelle et adaptabilité logique

On verra au Chapitre 6 que l'introduction conjointe de relations et de règles de cohérence permet la construction d'un "schéma de coexistence" [6], élargi à la coopération des bases de données.

Sur le plan théorique, ces relations, si elles sont normalisées et complétées par les contraintes logiques, suffisent à définir des schémas conceptuels [9]. URANUS peut dans ce cas être un bon outil pour l'élaboration d'un modèle s'appuyant sur cette notion (cf. § 2.4).

5.2. ASPECT TEMPOREL

Les règles d'utilisation et d'évolution des informations ont toutes une même finalité, qui est l'expression d'un ensemble de contraintes liées aux modifications subies par les éléments des bases les contenant.

Cette terminologie recouvre en fait deux aspects différents du but poursuivi.

D'une part, ce qui concerne les actions extérieures élémentaires qui modifient l'état des informations, c'est-à-dire les altérations dues aux procédures d'utilisation. Elles peuvent être associées à l'expression explicite de règles émises par les usagers (cf. § 5.3) ou l'administration.

D'autre part, l'interaction entre l'état d'une base de données et le facteur "temps". Au niveau le plus élémentaire, ceci peut être illustré par la définition d'une durée de validité de l'information, sous forme

de date de péremption.

A un stade plus évolué, par l'imposition d'une chronologie d'opérations sémantiquement liées. Par exemple : calcul d'un total facturé *après* enregistrement de toutes les lignes d'une commande, enregistrement d'une commande *après* vérification du compte-client, expédition de pièces *avant* mise à jour des stocks, etc ...

Ceci a été réalisé jusqu'à présent par l'intervention "dirigiste" et explicite des programmes de traitement associés aux applications. Il nous semble possible de tirer parti de la définition indépendante des règles d'intégrité pour assurer un usage plus dynamique des contraintes temporelles, exprimées de façon totalement dissociée des programmes utilisateurs (Fig. 5.3).

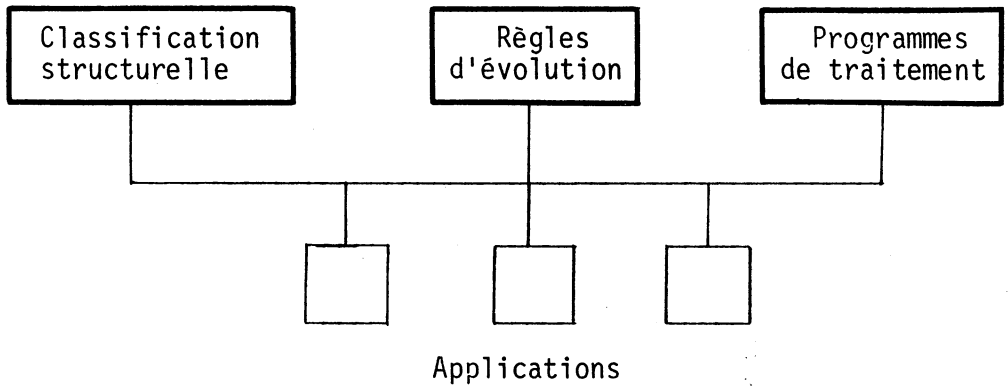


Fig. 5.3 - Applications et ensembles fonctionnels sous-jacents

On peut penser que cette prise en compte du facteur temps, sous la forme de contraintes d'intégrité est insuffisante. Ce point est essentiel, car d'une certaine manière, c'est toujours la confrontation des structures de données ou de leurs valeurs avec la notion abstraite de temps qui est la source des difficultés liées à l'adaptabilité des systèmes d'informations. Tous les conflits qui peuvent surgir ont en général pour origine une mauvaise adéquation de l'interprétation de cet élément [24].

Il est difficile dans l'état actuel des choses de représenter l'information en question autrement que :

- par l'introduction de domaines comportant expressément sa représentation,
- par la création d'actions spontanées, analogues aux "triggers" de "system R" [22].

Dans le premier cas, c'est l'interprétation des valeurs des nouveaux domaines qui doit se substituer à une logique contenue auparavant dans les programmes d'application.

Dans le deuxième cas, le système d'information réagit de lui-même à des évènements qui ont été portés à sa connaissance préalablement.

Les prédicats qui sont développés dans URANUS intègrent cette dernière notion indirectement.

Ils sont actuellement limités aux opérations (SELECT, INSERT, etc...) effectuées sur les relations.

5.3. MISE EN OEUVRE

Les prédicats permettent de définir un ensemble de conditions devant être vérifiées par les valeurs des constituants manipulés.

Ils sont définis par l'administrateur d'une base pour des raisons de sécurité, ou par un utilisateur pour définir un ensemble de règles de cohérence propres à une application.

Ils permettent d'introduire un ensemble de filtres de sélection prédéfinis, dont l'activation est totalement transparente en cours de traitement. Ceci donne la possibilité de créer des interfaces logiques (Fig. 5.4) analogues à des "vues" partielles [22] sur des informations existantes, dont la manipulation doit être protégée.

Ils peuvent être définis dynamiquement en cours de session sous l'environnement '$PRED' (cf. Annexe 7.2). Leur syntaxe est de la forme :

```
<NOM> PRED <REL>
        début
          | <OP> : <CONDITION> ;
          | <OP> : IF <CONDITION> THEN <PRED> ;
          | <OP> : IF <CONDITION> THEN <PRED> ELSE <PRED> ;
        fin
```

Les <CONDITION> sont du type préfixe, le champ <OP> indique l'opérateur auquel doit s'appliquer la condition élémentaire qui suit. Il peut prendre la forme : 'S', 'P', 'I', 'D', 'J', ou 'M', pour SELECT, PROJECT, INSERT, DELETE ou MODIFY respectivement. L'absence de <OP> dans un CONDITION indique que celle-ci doit s'appliquer à toutes les opérations indistinctement.

Les champs <PRED> des tests conditionnels sont des noms de prédicats déjà définis.

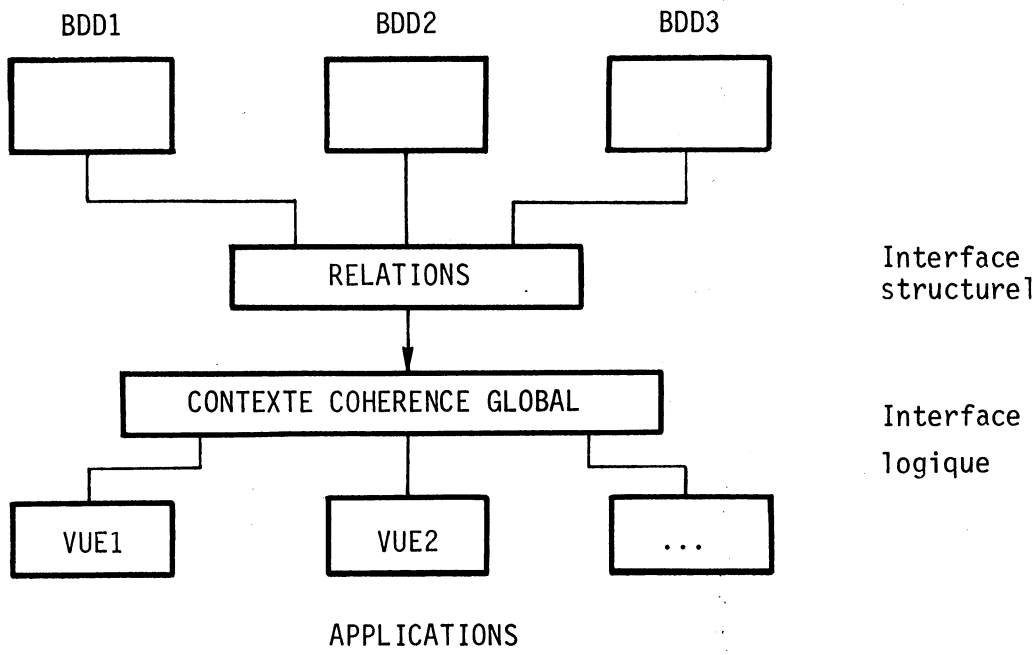


Fig. 5.4 - Superposition des niveaux structurels et logiques

Exemples :

Sur la relation :

```
SALARIE rel 1000
  début
    |  NOM mot 30
    |  INSEE mot 16
    |  NB-ENFANTS de 0 à 10
    |  SALAIRE de 0 à 10000
    |  SERVICE mot 20
  fin
```

on désire ne pas autoriser la mise à jour des personnes gagnant plus de 5000 F. On définit pour cela le prédicat :

```
P1 pred SALARIE
  début
  |  M : SALAIRE <= 5000 ;
  fin
```

dont l'activation est implicite dès l'analyse.
De même, on peut interdire la suppression d'employés ayant des enfants, soit :

```
P2 pred SALARIE
  début
  |    D : NB-ENFANTS = 0 ;
  fin
```

Si l'on désire interdire tout accès aux employés du service comptabilité :

```
P3 pred SALARIE
  début
  |    SERVICE ¬= 'COMPTABILITE'
  fin
```

ou seulement à ceux dont le salaire est inférieur à 2000 F et ayant plus de 2 enfants :

```
P4 pred SALARIE
  début
  |    if SERVICE = 'COMPTABILITE' then P5 ;
  fin

P5 pred SALARIE
  début
  |    ε(SALAIRE <= 2000, NB-ENFANTS >= 2) ;
  fin
```

L'ensemble de ces éléments est actuellement en cours d'intégration dans le langage de manipulation d'URANUS et devrait contribuer à la mise en oeuvre d'une plus grande adaptabilité sémantique.

Leur analyse est basée sur le même principe que les définitions de relations, c'est-à-dire génération directe en une passe. Les conditions sont analysées par une procédure identique à celle utilisée dans le langage de manipulation.

L'analyse correcte d'une définition donne lieu à la génération de deux types d'éléments :

- un descripteur de prédicat dans la relation-maître,
- une expansion du code interprétable des conditions sous forme arborescente dans la base relationnelle.

Exemple :
Le prédicat suivant : "P1 PRED REL1 DEBUT <COND1> FIN",
donne lieu à la génération du n-uplet suivant dans RMAITRE :

RNOM	RTYPE	RLG	RDEG	RNBMAX	RNB	RCLE	RINV	RDINV	RADRF	RADRL
P1	PRED	17	0	N	N	0	0	0	X	X+17*N

La valeur de RLG, c'est-à-dire la longueur d'un n-uplet de prédicat, est fixée à la génération du système, ceci pour être compatible avec la structure de l'arbre d'interprétation du langage de manipulation, selon le schéma suivant :

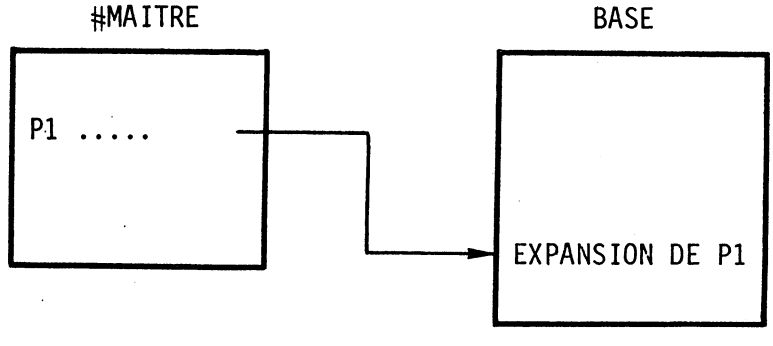
ATYPE	AREL	ACODE	AFRERE	AFILS	
1	4	4	4	4	OCTETS

Le champ RNB contient le nombre total de n-uplets générés pour ce prédicat dans la base.

Les adresses DEBUT et FIN de l'expansion du prédicat dans la base sont contenues dans RADRF et RADRL.

Les prédicats sont donc traités comme des relations particulières. Leur expansion est considérée comme un ensemble de n-uplets. Le descripteur de cette relation dans '#MAITRE' ne comporte qu'un seul élément, contrairement à une relation normale.

La génération du prédicat précédent est alors :



Lorsque la condition citée dans le prédicat est simple, l'expansion est celle fournie par l'analyse des requêtes élémentaires, précédée d'un n-uplet d'en-tête de la forme :

ATYPE	AREL	ACODE	AFRERE	AFILS
'R'	RID	0	0	2

le RID est l'indice de la relation sur laquelle s'applique le prédicat dans '#MAITRE'.

CHAPITRE 6

PERSPECTIVES

Bien que le système présenté ici ait été conçu pour la réalisation d'applications opérationnelles, il peut à bien des égards être considéré comme une première étape vers des développements ultérieurs plus importants.

Ceci tient aux fondements théoriques sur lesquels il s'appuie et aux techniques mises en oeuvre :

- utilisation des concepts relationnels,
- coopération de bases de données évoluées,
- conception ouverte et segmentée du système.

Les propositions suivantes nous semblent réalisables en des temps relativement courts, de l'ordre de quelques semaines (§ 6.2), de quelques mois (§ 6.3, 6.4), ou nécessiter des recherches plus fondamentales (§ 6.1 et 6.5).

Sans chercher à fournir des recettes, nous indiquons dans chaque cas les voies d'approche d'une possible mise en oeuvre.

URANUS constitue ainsi un outil de recherche immédiatement exploitable dont les prolongements relèvent tout à la fois du développement de systèmes de gestion d'information et de l'expérimentation de traitements évolués dans le domaine administratif.

6.1. EXTENSION MULTI-USAGERS ET MULTI-FICHIERS

La conception du système (cf. Annexe 7.1) rend possible une extension multi-accès à URANUS par l'adjonction d'un module de distribution des diverses fonctions du noyau, dans lequel seraient centralisées toutes les demandes de ressources conflictuelles :

- analyseur de relation,
- analyseur de requête,
- interface Socrate,
- méthode d'accès relationnelle.

Les interfaces entre les modules actuels ont été systématiquement définis de façon relationnelle (cf. Annexe 7.1), à l'exception de deux d'entre eux, la pile d'exécution des requêtes utilisateur ("A"), la table de mémorisation des modifications de données Socrate ("MODSOC"), qui ne sont pas rémanentes. On peut donc penser que :

- la coopération de bases relationnelles et Socrate en nombre quelconque,
- leur utilisation concurrente par plusieurs usagers

serait rendue possible par l'écriture d'un superviseur distribuant les requêtes selon une multiprogrammation sans préemption [29]. Pour cela, les requêtes provenant des consoles utilisateur doivent être traitées séquentiellement.

Une structure de donnée analogue à celle qui décrit actuellement la base relationnelle active [25] peut être développée pour mémoriser les caractéristiques des utilisateurs en cours de traitement.

Les accès physiques aux bases de données doivent être centralisés au niveau du superviseur de façon à éliminer toute interférence entre utilisateurs.

Sur le plan logique, les conflits d'accès aux relations peuvent être partiellement résolus à l'aide des règles de cohérences (Chap. 5), auxquelles on devra associer, en plus des noms de relations, un nom d'utilisateur et la possibilité d'exprimer des contraintes du type "usage exclusif", "usage partagé", etc ... [30].

Une étude approfondie de ces problèmes sort du cadre de ce travail et mériterait à elle seule un investissement matériel et humain au moins équivalent.

Il est d'autre part prévu d'étendre les possibilités du système réalisé en lui adjoignant des interfaces de manipulation de fichiers classiques et de nouveaux systèmes de gestion de Base de Données [23]. Ceci permettra d'assurer à un niveau concret une grande généralité au projet.

6.2. CONTEXTES DE COHERENCE

L'interface avec un grand nombre d'usagers amène à considérer les problèmes de conflits d'accès aux données sous la forme de droits d'utilisation concurrente des informations.

Un développement qui est d'ailleurs applicable au système mono-console tel qu'il existe, et permettant de masquer totalement ou partiellement des constituants de relations à certains utilisateurs semble d'une grande utilité à la fois pour des raisons de discrétion, de sécurité et de simplicité de manipulation de données.

Ici encore, la définition des vues partielles de relations nécessite un descripteur d'utilisateur, doublé d'un descripteur des sous-ensembles de relations accessibles.

Une application directe de cette extension sur le plan de la logique des traitements concerne la définition de contextes de cohérence superposés (Fig. 6.1) :

- un contexte global défini par un administrateur gestionnaire des bases permet d'édicter des règles générales de mise en oeuvre des relations (définition de données privilégiées, d'autorisation d'accès, etc ...),
- des contextes locaux, propres à un type d'utilisateur, lui garantissant des droits, autorisant les partages d'information,
- des contextes d'utilisation, associés à chaque application et définis par l'usager lui-même. Ceci peut être d'ores et déjà effectué au moyen des prédicats associés aux relations (§ 5.3).

Cette démarche est d'ailleurs conforme au "modèle de coexistence" de Nijssen [6], dont le schéma conceptuel comporte (Fig. 6.2), outre :

- un module structurel : "structure division", et
- un module de cohérence attaché aux données : "constraints division",

deux sous-ensembles relatifs aux contrôles des actions utilisateur, à savoir :

- un module de sécurité, "security-division", définissant pour chacun l'ensemble des accès autorisés,
- un module de partage, "concurrency division", assurant une distribution correcte des ressources en fonction du précédent.

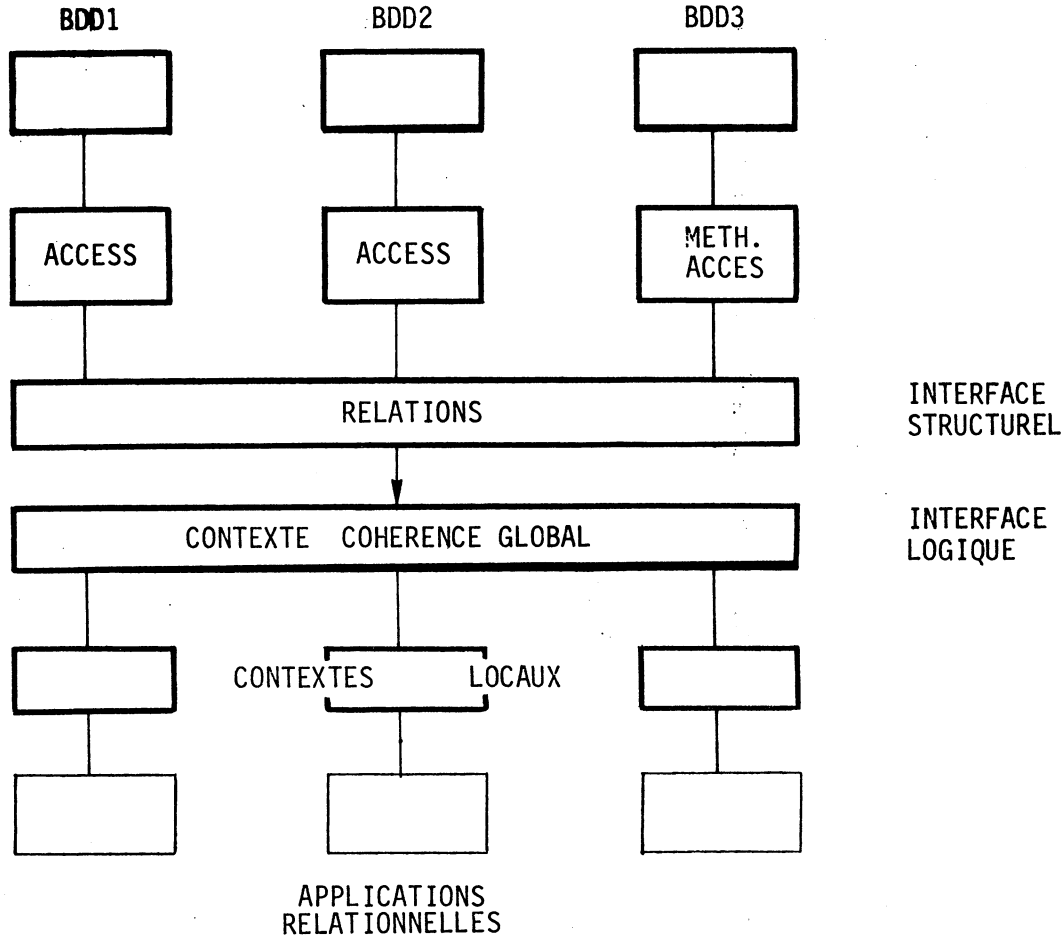


Fig. 6.1 - Un schéma de coexistence élargi à la coopération.

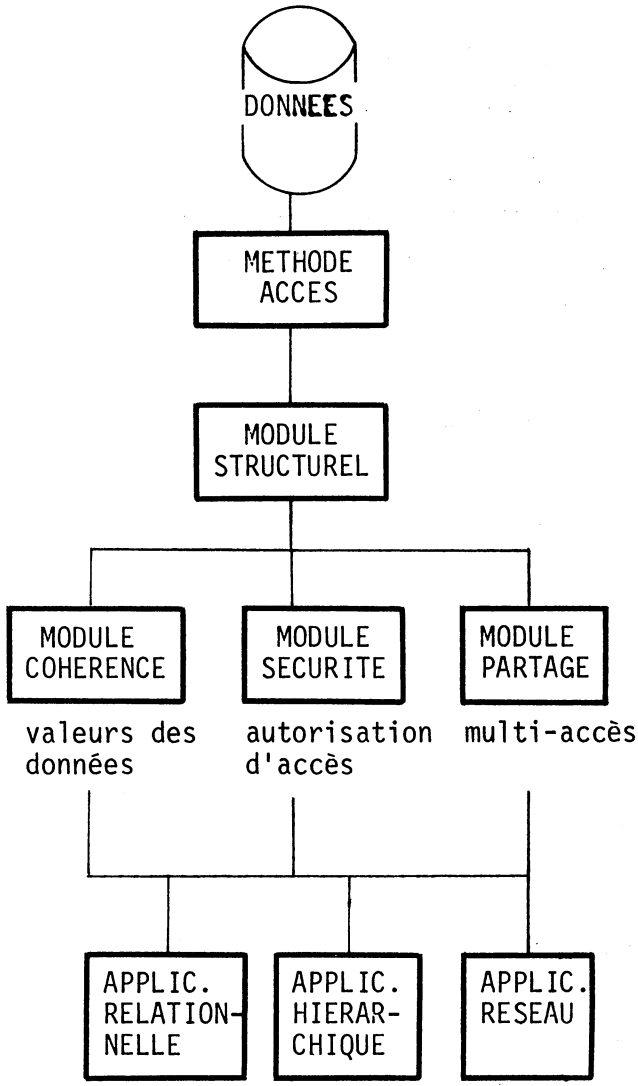


Fig. 6.2 - Modèle coexistence de Nijssen

6.3. UN SCHEMA DE COEXISTENCE GENERALISE

La terminologie du modèle de coexistence recouvre en général une utilisation de bases de données selon des techniques de représentation et de manipulation des informations différentes, voire incompatibles (Fig. 6.3).

La généralisation apportée par cette notion implique l'interaction entre un système de gestion et des traitements quelconques : relationnels, hiérarchiques, etc ...

Récemment est apparu l'intérêt d'une coopération entre des systèmes d'informations diversifiés [31] et des techniques de traitement multiples (Fig. 6.4) [6].

Notre travail s'est délibérément inscrit dans cette ligne, dans la mesure où seules des représentations particulières et des contraintes d'utilisation adaptées à chaque application peuvent satisfaire les besoins des non-informaticiens.

L'extension immédiate qui en découle, et qui permettra de réaliser un modèle de coexistence, généralisé à la coopération de données est la réalisation d'interfaces entre URANUS et d'autres SGBD (IMS, IDS, Mistral, etc ...), à l'instar de ce qui a été fait avec le multi-Socrate.

Les concepts relationnels garantissent toute intégration de techniques existantes et permettent donc d'envisager des développements importants dans le domaine de la coopération de bases de données (§ 2.3).

La définition de relations n-aires et l'introduction de contraintes logiques sur celles-ci permettent en effet de définir un schéma conceptuel décrivant, selon une logique indépendante de leur utilisation, les données nécessaires à une application.

Plus généralement, la coopération multi-bases introduit une plus grande généralité dans l'élaboration d'un modèle ou la "data-independence" [2] ne fait pas d'hypothèse sur l'unicité des moyens de stockage de l'information. On donne avec URANUS un exemple de non-unicité, dans le cas où les "schémas externes" [9] sont relationnels, les "schémas internes" relationnels et en réseau. On n'y construit cependant pas de schéma conceptuel complet puisque les relations sont calquées sur les structures de données préexistantes.

L'interface "interne" entre ce schéma et les données est constitué sur le plan relationnel par une méthode d'accès à un espace virtuel adapté [25], et au niveau Socrate par un ensemble d'opérateurs fonctionnels de haut niveau - les accesseurs - propres à chaque application.

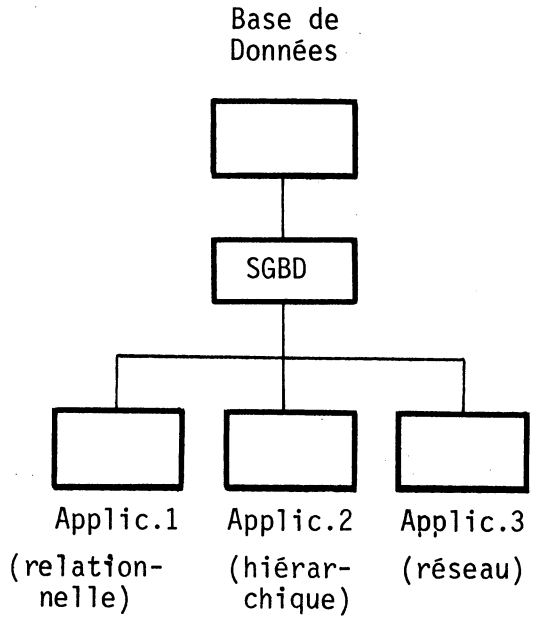


Fig. 6.3 - Schéma de coexistence primitif

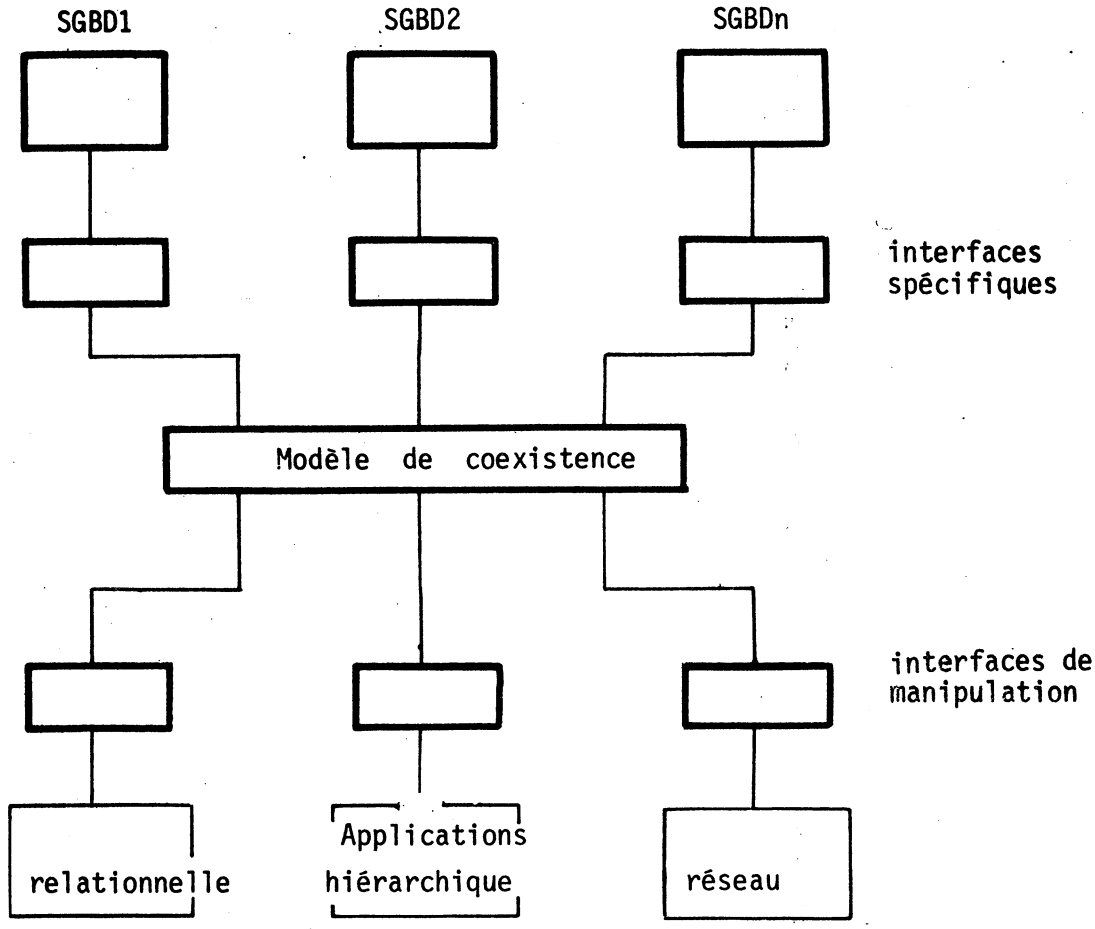


Fig. 6.4 - Schéma de coexistence généralisé

6.4. ENVIRONNEMENT REPARTI

La diversité créée par la réalisation du modèle de coexistence grâce à URANUS, l'origine et donc l'existence sur des systèmes d'exploitation incompatibles et disséminés de bases de données quelconques rendent nécessaire la réalisation d'un interface de télécommunication, permettant, pour bénéficier de l'intégralité de ses possibilités, de dialoguer au delà d'un site d'implantation.

Pour les accès distants, une partie des fonctions d'accès aux bases de données externes pourra être elle-même implantée sur les autres sites, en particulier en ce qui concerne :

- l'initialisation des relations à partir de données non relationnelles,
- la sauvegarde de ces mêmes relations dans les bases distantes,

la constitution locale d'ensembles complets de données permettant d'optimiser, en les minimisant, les transferts physiques de n-uplets à travers le réseau.

Les problèmes de répartition du modèle et la résolution sur un réseau de ses fonctions internes sortent du cadre de ce travail, et sont actuellement prévus par ailleurs [23] (Fig. 6.5).

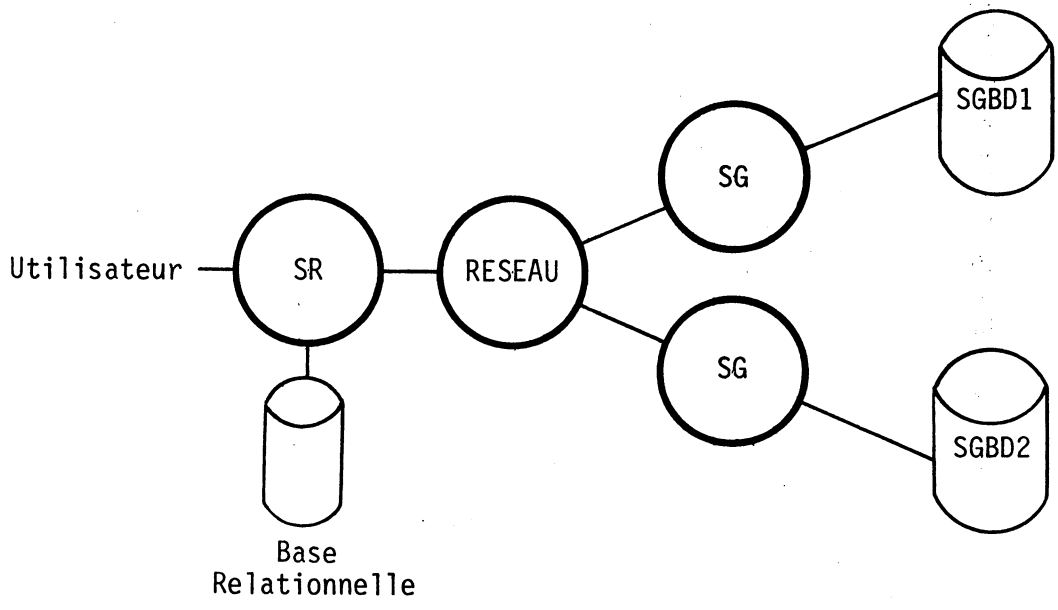


Fig. 6.5 - Répartition d'URANUS en sous-systèmes

URANUS doit en effet être scindé en deux composants distincts, à savoir :

- un sous-système relationnel de bases de données (SR),
- un sous-système de gestion de données (SG) disséminées dans plusieurs bases hétérogènes.

On aboutit ainsi à l'architecture définie par la figure 6.5, où les liaisons entre les différents composants peuvent être locales ou distantes selon le contexte de l'application à mettre en place.

Dans cet ensemble, l'utilisateur s'adresse au système relationnel (SR) pour définir et exploiter un ensemble de relations n-aires.

Lorsque dans une transaction, des données distantes, sont en jeu, par exemple initialisation d'une relation à partir des données d'une base résidant sur un autre site, le SR doit faire appel au SG par l'intermédiaire du réseau, pour demander l'extraction et le transfert des informations.

6.5. UN SYSTEME RELATIONNEL EXTENSIBLE

La conception d'un modèle de coexistence généralisé allié à la notion de schémas conceptuels peut être faite à l'aide des outils présentement disponibles dans URANUS (§ 2.4).

Une technique utilisée dans la réalisation des langages extensibles [26] peut ici être développée de façon analogue, avec l'utilisation des éléments de base du système dans un processus de développement récurrent.

Au stade actuel, ceci nécessite la mise en oeuvre de facilités de définition d'opérations complexes du type :

- définition d'objets,
- définition de liens sémantiques,
- de schémas conceptuels,
- de schémas internes et externes,
- de contraintes logiques sur ces éléments.

Pour cela, il peut être fait appel conjointement au langage de manipulation et de définition étudiés précédemment et qui peuvent être appliqués aux exemples donnés en 2.3 sur les modèles de Chen et de Todd [7], [8].

La définition de macro-requêtes paramétrables, exprimées en langage de manipulation de relations, peut alors être un bon moyen de mise en pratique

de ces développements : création de paramètres de définition des relations et des domaines (noms, degré ou bornes, types, ...) et exécution de requêtes préalablement cataloguées (Fig. 6.6).

La manipulation de ces éléments nécessite sans doute le développement d'interface avec des routines externes écrites de façon algorithmique, car la complexité de sélection des informations répertoriées de façon relationnelle dans ce modèle extensible peut en rendre l'expression malaisée. Sur le plan théorique, toutes les informations étant mémorisées de façon uniforme selon les normes du système relationnel de base URANUS, rien ne s'oppose à ce que l'ensemble des objets et liens soit manipulé à l'aide de composition d'opérations élémentaires déjà réalisées : SELECT, PROJECT, INSERT, JOIN, etc ...

Cette généralisation fait l'objet de travaux menés par ailleurs [28] [39], visant à la conception de langages de haut niveau comme PASCAL dotés de "types" et "modes" relationnels, et grâce auxquels peuvent être appliquées des méthodes évoluées d'écriture de systèmes de gestion de bases de données [40].

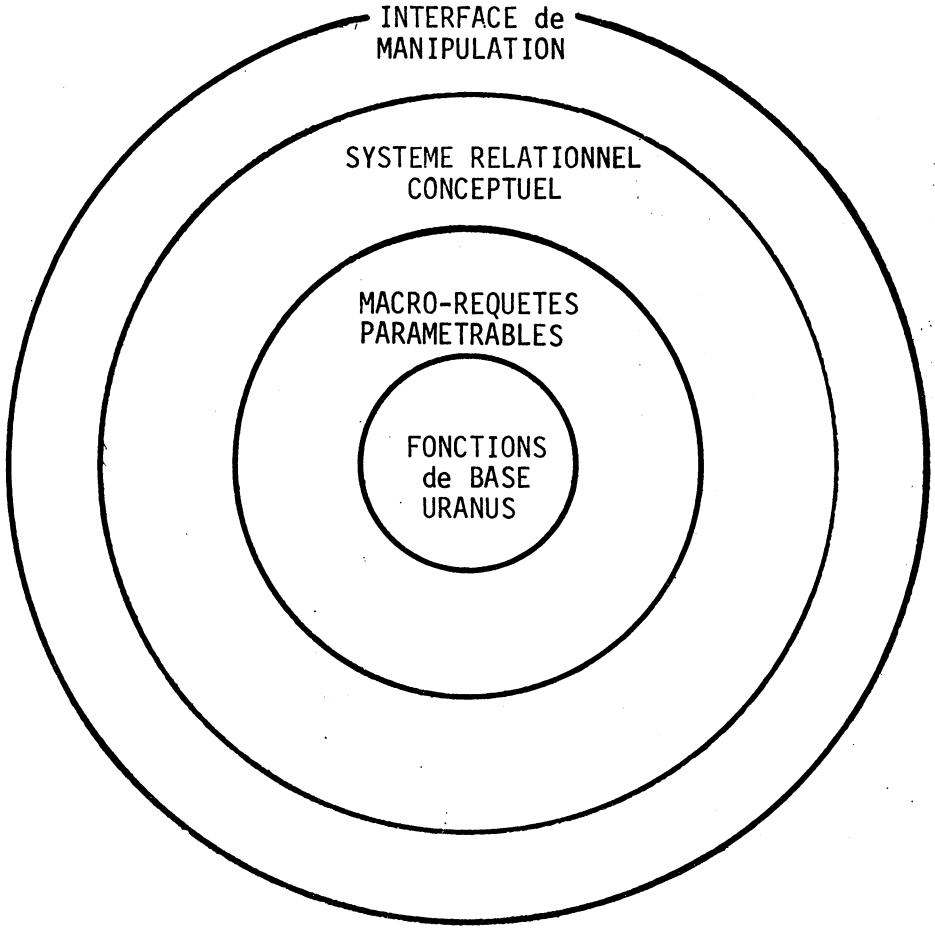


Fig. 6.6 - Un système relationnel extensible

CONCLUSIONS

Les travaux réalisés dans le cadre du projet URANUS ont abouti à la réalisation d'un prototype opérationnel, dont la mise au point a été effectuée à l'aide de fichiers et de bases de données réelles :

- cursus universitaire des étudiants inscrits à l'USMG [17],
- services effectués par les personnels enseignants [13], etc ...

Il est clair que les concepts relationnels qui ont guidé les choix fondamentaux de sa définition ont des retombées importantes à un niveau pratique, essentiellement en ce qui concerne :

- la *simplicité de représentation* de l'information, excluant toute notion de chemin d'accès ou de lien explicite entre les données,
- la *symétrie de manipulation* qu'elle permet grâce à
- un *langage simple de haut niveau*, procurant par ailleurs
- la possibilité d'exprimer les *contraintes sémantiques* attachées aux données indépendamment de leur définition structurelle et des programmes de traitement.

Ceci constitue une première étape vers une meilleure appréhension de l'information par des utilisateurs non spécialisés, objectif initial du projet.

L'extension proposée avec la coopération multi-bases s'appuie sur des fondements théoriques lui ouvrant de larges perspectives de développement, en particulier dans le domaine des *modèles conceptuels* de systèmes d'information et des *applications réparties*.

Dans cette optique, URANUS présente deux caractéristiques complémentaires. Il peut contribuer à

- la réalisation d'applications informatiques nouvelles jusqu'alors impraticables,
- la construction et la mise au point de systèmes de gestion de bases de données plus évolués.

Nous avons montré sur un exemple concret, à l'aide de bases de données Socrate, les possibilités offertes dans le premier cas. Elles peuvent être utilement étendues par l'incorporation d'autres logiciels et de fichiers standards.

Le deuxième point a été illustré par l'application des théories récentes de conception de modèles informatiques, et mérite une étude plus approfondie. Il fait actuellement l'objet de nouvelles propositions de recherches dont l'aboutissement devrait achever de démontrer la disponibilité et la généralité d'un système, conçu initialement dans une perspective plus restreinte.

Enfin, selon certains auteurs [36], "la tendance actuelle est au développement de structures de mémorisation opposées à la représentation tabulaire simplifiée de l'information, c'est-à-dire plus complexes et plus générales ...". Ainsi "la valeur des systèmes relationnels sera comparable à celle des systèmes de bases de données hiérarchiques ou en réseau lorsqu'ils permettront la description et la manipulation de ces derniers". Il nous est permis de penser que par ses possibilités et dans la mesure de sa réalisation, URANUS aura contribué à la démontrer.

CHAPITRE 7

ANNEXES

7.1. URANUS : Structure générale du système

L'environnement "initialisation-chargement" permet à un nouvel utilisateur de se faire connaître du système, de se faire allouer un espace de travail dans l'ensemble des données relationnelles, ou de réactiver un espace précédemment initialisé.

L'environnement "définition" préalable à toute application permet la définition des relations et vues d'un usager sur une base Socrate, ainsi que des éléments propres au contexte relationnel. A ce stade, toutes les données devront être connues afin de poursuivre les traitements. Leur manipulation est alors autorisée dans le contexte d' "exécution", permettant les traitements relationnels nouveaux tenant compte des prédicats définis sous l'environnement "contraintes logiques".

L'environnement Socrate est, quant à lui, réservé aux manipulations des informations des bases de données. Il nous a semblé, en effet, nécessaire de le séparer des contextes d'initialisation et d'exécution pour des raisons de sécurité : l'ensemble des données relationnelles pouvant ainsi être traitées de façon totalement indépendante, les risques éventuels de perte d'information Socrate dûs à des incidents peuvent être dans ce cas minimisés. Il permet par ailleurs une plus grande modularité du système (Fig. 7.1).

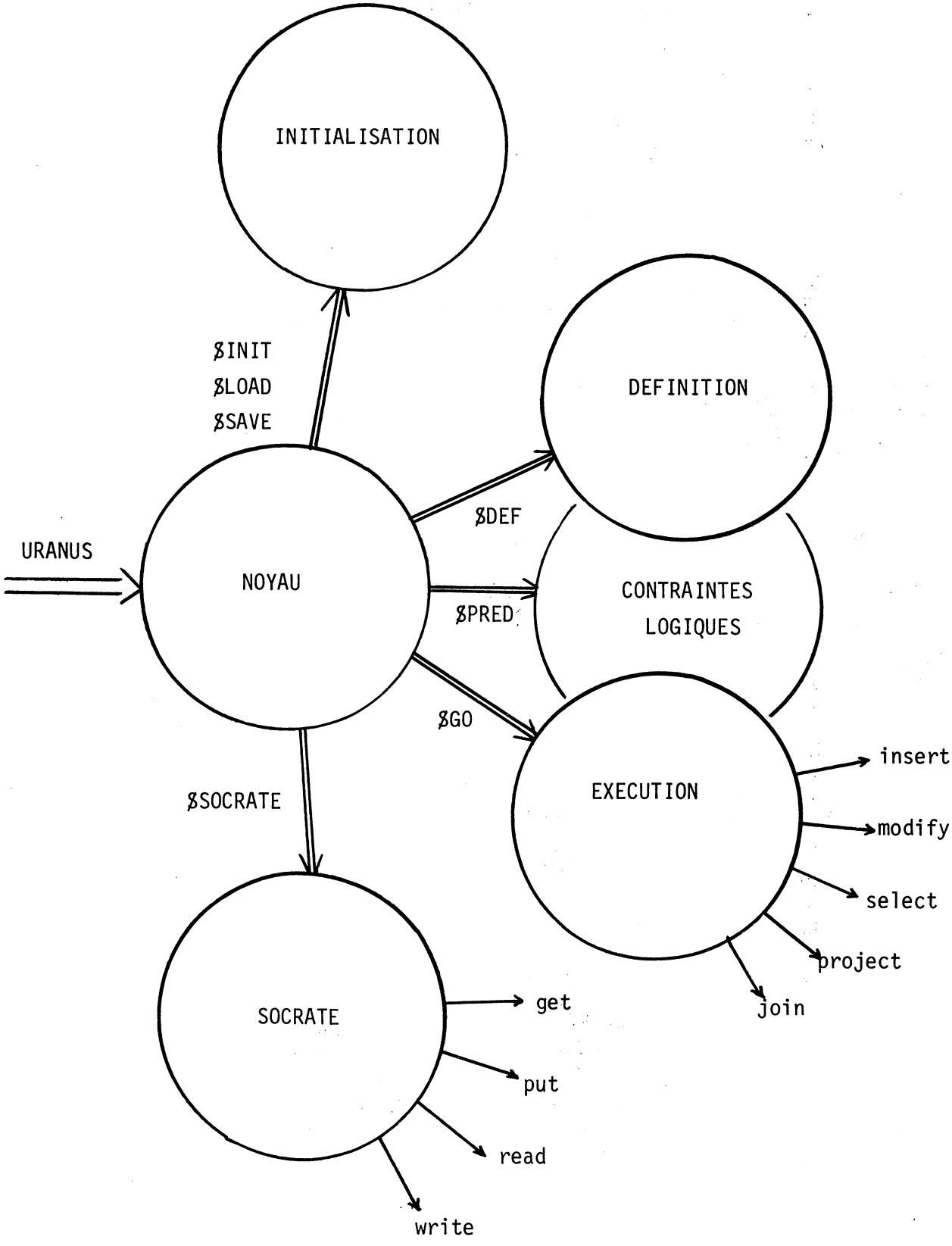
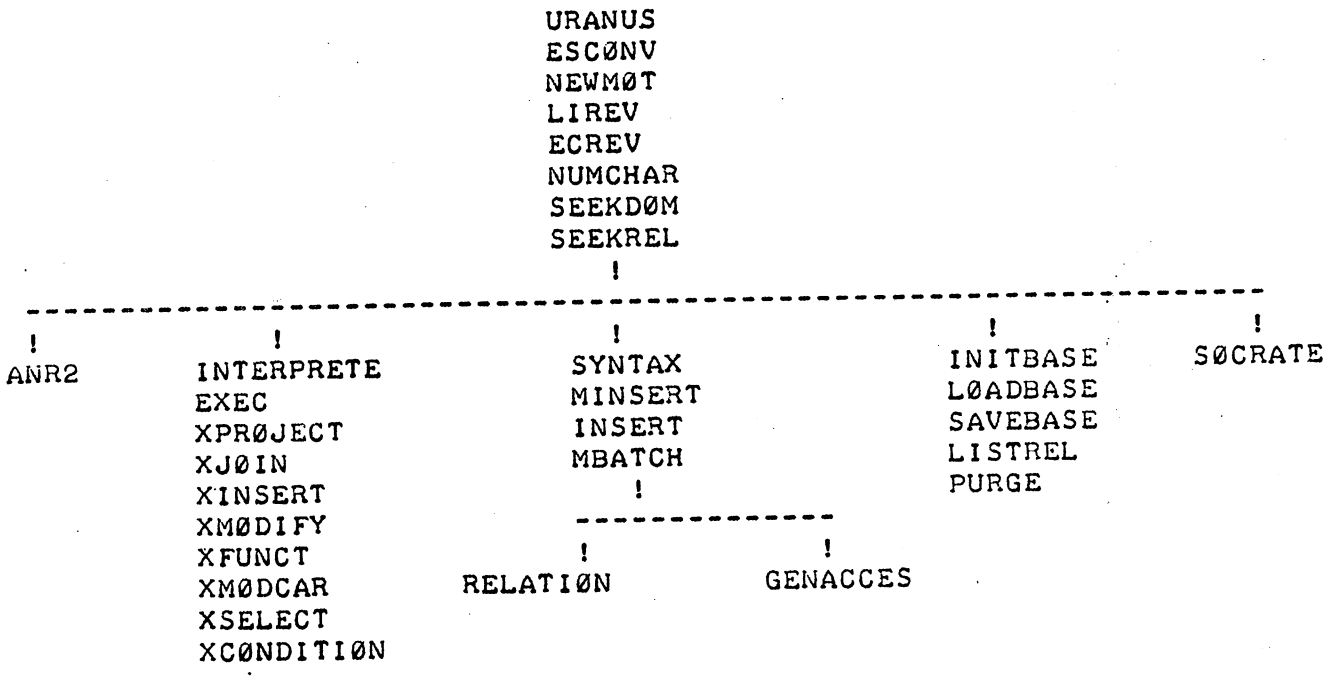
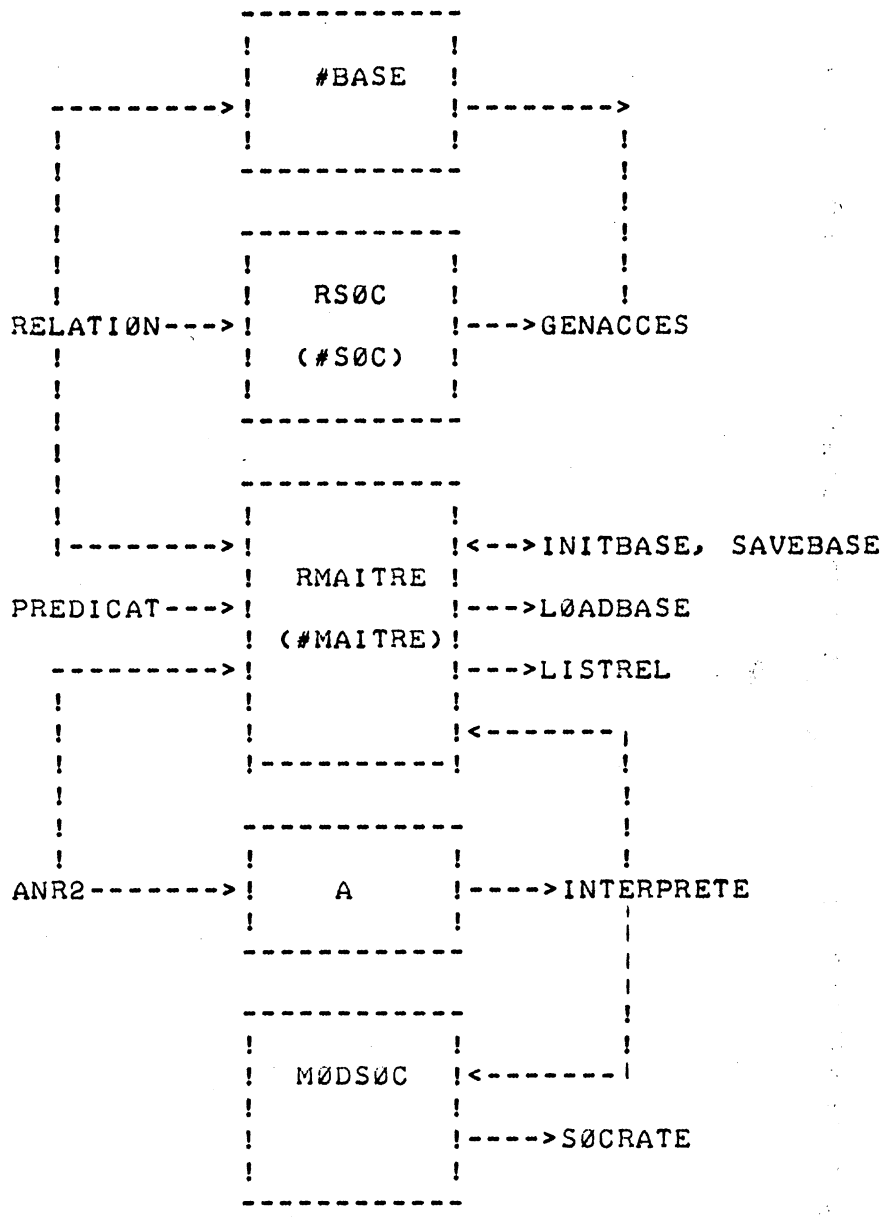


Fig. 7.1 - COMMANDES URANUS ET CONTEXTES ASSOCIES

Arbre de segmentation du module exécutable



URANUS : STRUCTURE DU SYSTEME.
****



INTERFACES ENTRE LES DIVERS MØDULES .

Les relations : "#BASE", "#SOC" et "#MAITRE" sont des relations-système particulières, inaccessibles aux usagers et nécessaires au fonctionnement interne d'URANUS.

Les tables "A" et "MODSOC" n'existent qu'en cours de session et servent à l'interprétation des requêtes et à la mémorisation des modifications de données corrélées respectivement.

ANNEXE 7.2. URANUS : Commandes utilisateur

7.2.1 INITIALISATIØN D' UNE BASE : $INIT <BASE> .

LE PARAMETRE FØURNIT <BASE> PERMET DE DESIGNER L'ØPL DU FICHIER DES DØNNEES, ØBLIGATØIREMENT FØURNIT AU LANCEMENT DU LØAD MØDULE URANUS DANS UN CARTE ASSIGN, AVEC STATUS NEW .

CETTE CØMMANDE A PØUR BUT DE FØRMATER L' ESPACE REEL DØNNE, ET D' INITIALISER LA RELATIØN MAITRE .

7.2.2 CHARGEMENT D' UNE BASE : $LØAD <BASE> .

CETTE CØMMANDE PERMET D' INITIALISER UNE SESSIØN URANUS AVEC UNE BASE EXISTANTE . LE FICHIER ASSØCIE A L' ØPL <BASE> DØIT EXISTER ET AVØIR LE STATUS MØD DANS LA CARTE ASSIGN CØRRESPØNDANTE .

LA FRAPPE DE LA CØMMANDE $LØAD <BASE> EN CØURS DE SESSIØN A LE MEME EFFET QUE LA FRAPPE DE $SAVE .

7.2.3 SAUVEGARDE D' UNE SESSIØN : $SAVE .

CETTE CØMMANDE PRØVØQUE L'ECRITURE EN MEMØIRE SECØNDAIRE DE TØUTES LES INFØRMATIØNS ACTIVES PRESENTES EN MEMØIRE CENTRALE . LA SESSIØN SE PØURSUIT NØRMALEMENT .

7.2.4 LISTE DES RELATIØNS CATALØGUEES : $LISTREL .

CETTE CØMMANDE IMPRIME SUR L' ØRGANE DE SØRTIE LA LISTE DE TØUTES LES RELATIØNS EXISTANT DANS LE SYSTEME, SØUS LA FØRME : <NØM DE REL> (DØMAINE1, DØMAINE2, ...)

7.2.5 PARAMETRES D' EXECUTIØN DE SYSTEME : $BASE .

CETTE CØMMANDE FØURNIT LES PARAMETRES SYSTEME SØUS LA FØRME :

PARAM1 = VAL1, PARAM2 = VAL2, ...

LES PARAMETRES SØNT DYNAMIQUES ET PEUVENT ETRE CHANGES PAR L' UTILISATEUR A TØUT MØMENT AU MØYEN DE LA CØMMANDE $GØ .
LEUR LISTE EST :

- TIME = ØBTENTIØN DES TEMPS D' EXECUTIØN DE TØUTE CØMMANDE ELEMENTAIRE ØU SYSTEME FRAPPEE .
- TRACE = ØBTENTIØN DES TEMPS D' EXECUTIØN ET INDICATEURS : NØM DES PRØCEDURES EXECUTEES,
- TRLEXICØ = IDEM PØUR L'ANALISEUR DU LANGAGE DE MANIPULATIØN SEUL .
- TRANAL = IDEM PØUR L'INTERPRETEUR .
- TREXEC = IDEM PØUR LES RØUTINES D' EXECUTIØN .
- TRDEF = IDEM PØUR L'ANALYSEUR DU LANGAGE DE DEFINITIØN .
- SYSTEM = MØDE SYSTEME PRIVILEGIE .

7.2.6 DEFINITIØN DE RELATIØNS : $DEF .

ACTIVATIØN DE L' ENVIRØNNEMENT DE DEFINITIØN DES RELATIØNS DE L' UTILISATEUR .

TØUTE DETECTIØN D' ERREUR PRØVØQUE UNE REMISE A ZERØ DES VARIABLES DE L' ANALYSEUR : L' ANALYSE DE LA RELATIØN CØNTINUE JUSQU' A LA FIN , MAIS NE GENERE PAS LA RELATIØN .

7.2.7 DEFINITIØNS DE PREDICATS : $PRED .

UN PREDICAT EST CØNSIDERE CØMME UNE CØNTRAINTE LØGIQUE S' APPLIQUANT A UN TRIPLET : RELATIØN-USAGER-ØPERATIØN . ILS PEUVENT ETRE CREES EN CØURS DE SESSIØN, ET DØIVENT PØRTER SUR UNE RELATIØN DEJA DEFINIE DANS LE SYSTEME .

IL EST PØSSIBLE DE SPECIFIER PLUSIEURS CØNTRAINTES A L' INTERIEUR D' UN MEME PREDICAT, CELLES-CI DEVANT S' APPLIQUER A UNE ØPERATIØN BIEN DETERMINEE (SELECT,PRØJECT,JØIN, ...) .

LES CØNTRAINTES DØIVENT ETRE EXPRIMEES SØUS FØRME DE CØNDITIØNS PREFIXEES PØRTANT SUR LES DØMAINES DES RELATIØNS CØRRESPØNDANTES.

IL EST PØSSIBLE D' IMBRIQUER PLUSIEURS PREDICATS PAR L' INTERMEDIAIRE DE TESTS LØGIQUES SUR LES CØNDITIØNS PRECEDENTES .

7.2.8 MANIPULATIØN DE RELATIØNS : $GØ .

PERMET L' ACTIVATIØN DE L' ANALYSEUR DU LANGAGE D' EXECUTIØN ET/ØU LA MØDIFICATIØN DES PARAMETRES D' EXECUTIØN DU SYSTEME, SØUS LA FØRME :

PARAM1 PARAM2 .....

ØU PARAM EST L' UN DES PARAMETRES DEFINIS CI-DESSUS (7.2.5).

7.2.9 FIN DE SESSIØN : $ØFF .

CETTE CØMMANDE SUVEGARDE EN MEMØIRE SECØNDAIRE LES DØNNEES RELATIØNNELLES (RØLE IDENTIQUE A '$SAVE') ET TERMINE L' EXECUTIØN DU MØDULE SYSTEME . ØN ENVØIE SUR L' ØRGANE DE SØRTIE UN ETAT DES ENTREES/SØRTIES EFFECTUEES DEPUIS LE DEBUT DE LA SESSIØN .

7.2.10 REMISE A INDEFINI D' UNE RELATIØN : $PURGE .

L' ENSEMBLE DES N-UPLETS DE LA RELATIØN FØURNIE EN PARAMETRE EST RENDUE INNACCESSIBLE. LA RELATIØN RESTE DEFINIE DANS LE SYSTEME DE FACØN A PØUVØIR ETRE REINITIALISEE .

7.2.11 ACCES A DES BASES SØCRATE : $SØCRATE .

CETTE CØMMANDE PERMET D' ATTEINDRE L' ENVIRØNNEMENT SØCRATE ET D' UTILISER LES FØNCTIØNS D' ACCES SUR LES BASES DEFINIES PAR UN UTILISATEUR (CF. LANGAGE DE DEFINITIØN DE RELATIØNS) :

```
GET    <RELATIØN>,<FENETRE>;
PUT    <RELATIØN>;

READ   <RELATIØN>,<FILTRE>,<FENETRE>;
WRITE  <RELATIØN>;
```

ANNEXE 7.3. URANUS : Relation-maître

```
#MAITRE REL 100
   |    #NOM MOT                          nom de relation ou de constituant
   |    #TYPE (REL MOT NUM LVAL ...)       type de constituant
   |    #LG DE 0 A 999                     Lg d'1 n-uplet ou du constituant
   |    #DEGRE DE 0 A 999                  degré de relation ou pos. du constituant
   |    #CLE DE 0 A 999                    nombre de champ clé
   |    #NBMAX DE 0 A 999                  cardinal de relation
   |    #NB DE 0 A 999                     nb de n-uplets courants
   |    #RINV DE 0 A 100
   |    #DINV DE 0 A 999
   |    #ADRF DE 0 A 999                   adresse début relation ou borne inférieure
   |    #ADRL DE 0 A 999                   adresse fin relation ou borne supérieure
FIN
```

La relation-maître est une relation prédéfinie du système, non accessible aux utilisateurs. Elle contient à chaque instant l'ensemble des informations nécessaires au bon fonctionnement d'une session : initialisation, sauvegarde, descripteurs de toutes les relations utilisateur, paramètres d'exécution (taille de la mémoire relationnelle, etc ...).

	RELATIØN	PREDICAT	LISTE VAL
#NØM		IDENTIFICATEUR	
#TYPE	REL	PRED	REL
#LG		LØNGUEUR EN ØCTETS D' UN N-UPLET	
#DEG	DEGRE DE REL		1
#NBMAX	CARDINAL	NB ELEMENTS	CARDINAL
#NB	NB ELEMENTS CØUR.	NB ELEMENTS	NB ELEMENTS CØUR.
#CLE	NB CHAMPS CLE		
#RINV	NØ BASE SØCRATE		
#RDINV	PØINTEUR DANS #SØC		
#RADRF	ADRESSE DEBUT	ADRESSE DEBUT	ADRESSE DEBUT
#RADRL	ADRESSE FIN	ADRESSE FIN	ADRESSE FIN

RØLE DES CØNSTITUANTS DE RMAITRE .
****

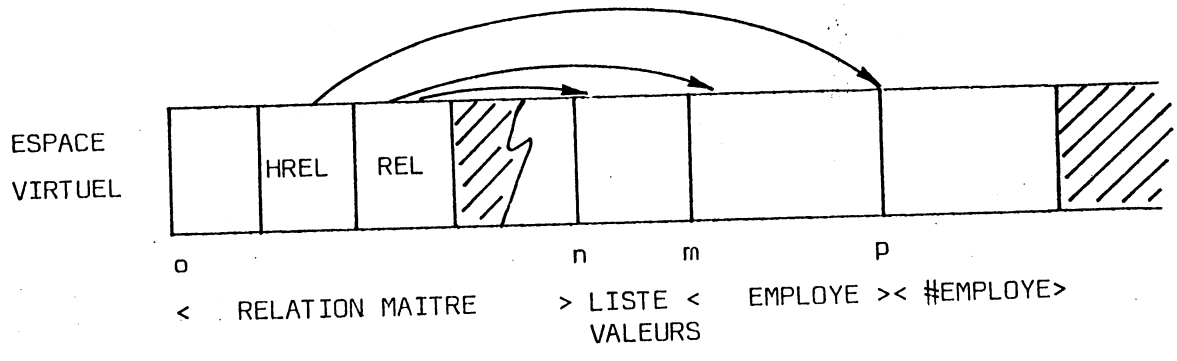
	VAL. NUMERIQUE	CHAINE DE CAR	
#NØM	IDENTIFICATEUR		
#TYPE	NUM	MØT	
#LG	LØNGUEUR EN ØCTETS		
#DEG	PØSITIØN DANS LE N-UPLET		
#NBMAX			
#NB			
#CLE			
#RINV			
#RDINV			
#RADRF	BØRNE INF		
#RADRL	BØRNE SUP		

MISE EN OEUVRE : DESCRIPTEUR DE RELATION

INTERPRETATION DU LANGAGE DE DEFINITION

```
EMPLOYE REL 100
    | NOM mot clé
    | AGE de 0 à 100
    | ETATCIV (CELIB VEUF DIV)
FIN
```

NOM	TYPE	LG	DEG	NBMAX	NB	CLE	ADRF	ADRL
EMPLOYE	REL	24	3	100	0	1	m	m = n+3*5
NOM	MOT	16	0			1		
AGE	NUM	3	16				o	100
ETATCIV	LVAL	5	19		3		n	n+15
#EMPLOYE	HREL	20	3	100	0		p	m+100*24
NOM	MOT	16	0					
#VAL	NUM	3	16				o	100
#LINK	NUM	3	19				o	100



ANNEXE 7.4. Corrélations URANUS-SOCRATE : exemple

7.4.1. Structure Socrate .

```
ENTITE  15000  ETUDIANT

   DEBUT
   |      NØ DE 0 A 9999999 AVEC CLE (40 $NØ) IN
   |      NØM MØT 23        AVEC CLE (27 $LT) IN
   |      PRENØM MØT 10
   |      INSEE  MØT 16
   |      ADRESSE MØT 80
   |
   |      ENTITE  30  FDC
   |         DEBUT
   |         |      AN     DE 0 A 99
   |         |      ETAPE  DE 0 A 20000
   |         |      RESUL  DE 0 A 9
   |         |
   |         |      ENTITE  15  ENS
   |         |         DEBUT
   |         |         |        CØDE-ENS  DE 0 A 20000
   |         |         |        JUIN      DE 0 A 9
   |         |         |        SEPT      DE 0 A 9
   |         |         FIN
   |         FIN
   FIN
```

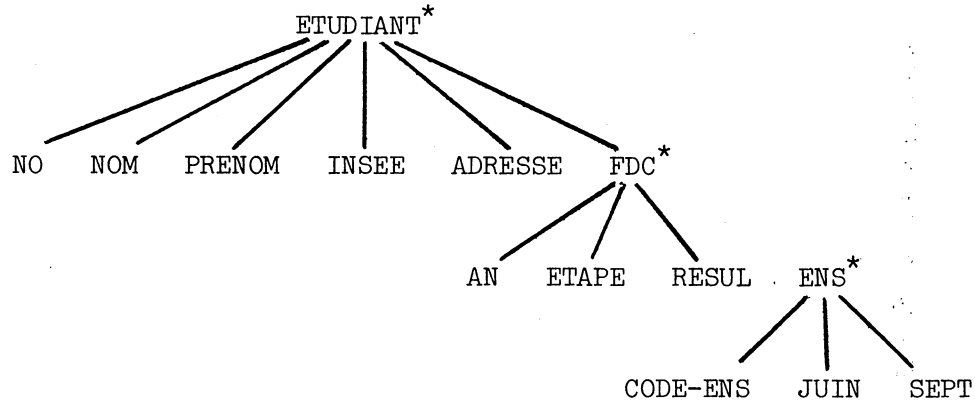
7.4.2. Relation correspondante .

```
ETUDIANT  REL  10  IDEM  ETUDIANT DANS SØC2
->
    DEBUT
->
          NUMERØ   DE 0 A 9999999   IDEM NØ
->
          NØM      MØT 23           IDEM NØM
->
          INSEE    MØT 16           IDEM INSEE
->
          ANNEE    DE 0 A 99        IDEM AN DE FDC
->
          ETAPE    DE 0 A 20000     IDEM ETAPE DE FDC
->
          ENS      DE 0 A 20000     IDEM CØDE-ENS CLE DE
                                           ENS DE FDC
->
    FIN
```

7.4.3. Table de corrélations : n-uplets de #SOC .

-1	ETUDIANT	40	2	0	0
1	ETUDIANT	1	0	0	1
0	NØ	1	2	0	0
0	NØM	1	2	0	1
0	INSEE	1	2	0	1
0	AN	2	7	0	0
2	FDC	2	2	0	0
0	ETAPE	2	7	0	0
0	CØDE-ENS	3	10	0	0
3	ENS	3	7	0	0

7.4.4. Qualification hiérarchique .



ANNEXE 7.5. Accesseurs Socrate : exemple de génération

LES ACCESSEURS SØCRATE CØNSTITUENT L' INTERFACE DE HAUT NIVEAU URANUS - SØCRATE .

C' EST UN ENSEMBLE DE SØUS-PRØGRAMMES BATCH SØCRATE GENERES AUTØMATIQUEMENT ET DE FACØN TRANSPARENTE . LE MØDULE URANUS EFFECTUANT CE TRAVAIL EST GENACCES . IL ANALYSE LA TABLE RSØC GENEREE PAR L'INTERPRETEUR DU LANGAGE DE DEFINITIØN, ET CØNSTITUE UN FICHIER IMAGE DE SESSIØN SØCRATE FILI-FILØ SØUMIS A LA FILE D' ATTENTE DES TRAVAUX SIRIS 8 .

CE FICHIER CØNTIENT LE CØDE SØURCE DES ACCESSEURS GENERES PAR LE MØDULE GENACCES . L'EN-TETE ET LA FIN DE CE FICHIER D' ØPL : ISØC , SØNT GENERES A PARTIR D' UN MØDELE CØNTENU DANS UN FICHIER D' ØPL : FSØC ET DE NØM : MØDELE . CE FICHIER EST PARAMETRE A LA DEMANDE DANS UN ENVIRØNNEMENT MULTI-BASE .

L' ENSEMBLE DES RØUTINES DE GENACCES EST STRUCTURE DE LA MANIERE SUIVANTE :

```
GENACCES    -> INITSØCREL -> GEN-DEFMAC -> GEN-DEBUT -> GEN-FØRMAL
            -> GEN-FIN -> MBATCH -> INITSØCREL

GEN-DEBUT   -> GEN-PARCØURS -> GEN-EXIST
                            -> GEN-XI

GEN -FIN    -> GEN-SVT

INITSØCREL -> CRE-SØCREL -> INSERT

CRE-SØCREL -> INITMAITRE -> INITADREL -> CATALREL
```

LE RØLE DES DIFERENTES PRØCEDURES EST DØNNE CI-DESSØUS :

- GEN-DEFMAC : GENERATIØN DE L' EN-TETE DE SØUS-PRØGRAMME BATCH SØCRATE ACCESSEUR .
- GEN-DEBUT : GENERATIØN DE L' INITIALISATIØN DU FØRMAL ET DES ZI : SØCRATE UTILISES CØMME INDICATEURS INTERNES DE : FIN D' ENTITE .
- GEN-FØRMAL : GENERATIØN DE L' INITIALISATIØN DU FØRMAL AVEC : LES CARACTERISTIQUES DE L' ENTITE TRAITEE .
- GEN-FIN : GENERATIØN DES FINS DE BØUCLES 'FAIRE-REFAIRE-FIN' : SØCRATE, AVEC TEST DES INDICATEURS ZI .
- GEN-PARCØURS : PARCØURS DE LA TABLE RSØC DE FACØN A FØURNIR LES : QUALIFICATIØNS D' UNE ENTITE DANS L' ØRDRE HIERARCHIQUE .
- GEN-EXIST : GENERATIØN DU TEST D' EXISTENCE D' UNE ENTITE .
- GEN-XI : GENERATIØN DE L' AFFECTATIØN DES XI SØCRATE AUX : ENTITES .
- GEN-SVT : GENERATIØN DE L' APPEL A LA MACRØ SVT <ENTITE> .
- CRE-SØCREL : CREATIØN DE LA RELATIØN #SØC .
- INITSØCREL : INITIALISATIØN DE #SØC A PARTIR DU TABLEAU RSØC .

AFIN DE SE PØSITIØNNER CØRRECTEMENT SUR UNE ENTITE APRES CHAQUE APPEL A UN ACCESSEUR SEQUENTIEL , ØN UTILISE LA MACRØ PREDEFINIE 'SVT <ENTITE> <XI> <YI>' , DØNNANT PØUR <ENTITE> UNE INITIALISATIØN DE <XI> A L' ENTITE SUIVANTE DE NUMERØ DE REALISATIØN <YI> .

SUR LA RELATIØN :

```
ENTITE  15000  ETUDIANT
     DEBUT
       |   NØM    MØT 30
       |   NØ     DE 0 A 1000
       |   INSEE MØT 16
       |
       |   ENTITE  15  FDC
       |      DEBUT
       |        |       AN     DE 0 A 99
       |        |       ETAPE DE 0 A 9999
       |        |
       |        |       ENTITE  10  ENS
       |        |          DEBUT
       |        |            |        CØDE-ENS DE 0 A 9999
       |        |          FIN
       |        |
       |      FIN
     FIN
```

L' ENSEMBLE DU FICHIER GENERE PØUR LE CATALØGUAGE DE L' ACCESSEUR SEQUENTIEL EST :

```
/* EN-TETE DE TRAVAIL */

!JØB,B 0248:ACC,0286,5052:248
!LIMIT (CØRE,99),(SPDI,90),(TIME,2)
!ASSIGN ESP1,FIL,(NAM,H:ESPR2),(STS,MØD)
!ASSIGN SØU1,FIL,(NAM,H:SØURCE),(STS,MØD)
!ASSIGN STR1,FIL,(NAM,H:STRUCT),(STS,MØD)
!ASSIGN SSPD,FIL,(NAM,H:SUSP),(STS,MØD)
!ASSIGN SI,FIL,(NAM,PARBATCH),(STS,ØLD)
!ASSIGN FILI,DEV,IN
!ASSIGN FILØ,DEV,ØUT
!SØCRATE,C
L 0,N,STR1,:STRUCT
PASSW

GØ EDIT
F/MACRØ/
OI
```

```
/* MACRØ-REQUETE DE PØSITIØNNEMENT SUR LES ENTITES */

:DEFMAC SVT : : : :
:EXP
M Z5 = 'ØK' M Y10 = D TØUT :1: DE :4:
 FAIRE SI :2: < Y10 ALØRS
M :2: = :2: + 1
SI PAS UN :1: :2: DE :4: ALØRS REFAIRE FIN
SINØN M :2: = 1 M Z5 = 'EØE' SØRTIE FIN FIN
:FDEF
ETX
F/BATCH/
OI

/* ACCESSEUR SEQUENTIEL EN LECTURE */

:DEFMAC   GETETUDIANT          :EXP
   M X10 = FØRMAL UETUDIANT          DANS BUFFER 1
   M Z4 = 'FAUX'  M Z5 = 'EØE'
 FAIRE SI EXISTE UN ETUDIANT          X1 Y1 DE X0 ALØRS
 FAIRE SI EXISTE UN FDC               X2 Y2 DE X1 ALØRS
 FAIRE SI EXISTE UN ENS               X3 Y3 DE X2 ALØRS
     M NØ                DE X10 = NØ                DE X1
     M NØM               DE X10 = NØM               DE X1
     M INSEE             DE X10 = INSEE             DE X1
     M AN                DE X10 = AN                DE X2
     M ETAPE             DE X10 = ETAPE             DE X2
     M CØDE-ENS          DE X10 = CØDE-ENS          DE X3
    SVT ENS               Y3 X3 X2
   M Z4 = 'VRAI' M Y9 = NUMDE X1  SØRTIE FIN
    SVT ENS               Y3 X3 X2
   SI Z5 = 'EØE' ALØRS SØRTIE FIN
   REFAIRE FIN FIN


   SI Z5 = 'EØE' ALØRS
    SVT FDC               Y2 X2 X1
   FIN
   SI Z4 = 'VRAI' ØU Z5 = 'EØE' ALØRS SØRTIE FIN
   REFAIRE FIN FIN
   SI Z5 = 'EØE' ALØRS
    SVT ETUDIANT          Y1 X1 X0
   FIN
   SI Z4 = 'VRAI' ØU Z5 = 'EØE' ALØRS SØRTIE FIN
   REFAIRE FIN
SI Z5 = 'EØE' ØU Z4 = 'FAUX' ALØRS M Z5 = 'EØR'                FIN
:FDEF

/* ACCESSEUR SEQUENTIEL EN ECRITURE */

:DEFMAC   PUTETUDIANT          :EXP
   M X10 = FØRMAL UETUDIANT          DANS BUFFER 1
   M Z4 = 'FAUX'  M Z5 = 'EØE'
 FAIRE SI EXISTE UN ETUDIANT          X1 Y1 DE X0 ALØRS
 FAIRE SI EXISTE UN FDC               X2 Y2 DE X1 ALØRS
 FAIRE SI EXISTE UN ENS               X3 Y3 DE X2 ALØRS
  SI Y9 = 0 ALØRS
 M NØ                DE X1  = NØ                DE X10
 M NØM               DE X1  = NØM               DE X10
 M INSEE             DE X1  = INSEE             DE X10
```

```
 M AN                DE X2   = AN                DE X10
 M ETAPE             DE X2   = ETAPE            DE X10
 M CØDE-ENS          DE X3   = CØDE-ENS         DE X10
    SVT ENS                Y3 X3 X2
   M Z4 = 'VRAI' SØRTIE SINØN M Y9 = Y9          - 1 FIN FIN
    SVT ENS                Y3 X3 X2
   SI Z5 = 'EØE' ALØRS SØRTIE FIN
   REFAIRE FIN FIN
   SI Z5 = 'EØE' ALØRS
    SVT FDC                Y2 X2 X1
   FIN
   SI Z4 = 'VRAI' ØU Z5 = 'EØE' ALØRS SØRTIE FIN
   REFAIRE FIN FIN
   SI Z5 = 'EØE' ALØRS
    SVT ETUDIANT           Y1 X1 XO
   FIN
   SI Z4 = 'VRAI' ØU Z5 = 'EØE' ALØRS SØRTIE FIN
   REFAIRE FIN
SI Z5 = 'EØE' ØU Z4 = 'FAUX' ALØRS M Z5 = 'EØR'              FIN
:FDEF

/* ACCESSEUR SELECTIF EN LECTURE */

:DEFMAC  READETUDIANT          :EXP
   M X10 = FØRMAL UETUDIANT          DANS BUFFER 1
   M Z4 = 'FAUX'  M Z5 = 'EØE'
 FAIRE SI EXISTE UN ETUDIANT          X1 Y1 DE XO ALØRS
 FAIRE SI EXISTE UN FDC               X2 Y2 DE X1 ALØRS
 FAIRE DEPUIS Y3
       SI EXISTE UN ENS  X3 AYANT CØDE-ENS = Z1 ; DE X2 ALØRS
     M NØ                DE X10 = NØ               DE X1
     M NØM               DE X10 = NØM              DE X1


     M INSEE             DE X10 = INSEE            DE X1
     M AN                DE X10 = AN               DE X2
     M ETAPE             DE X10 = ETAPE            DE X2
     M CØDE-ENS          DE X10 = CØDE-ENS         DE X3
    SVT ENS               Y3 X3 X2
   M Z4 = 'VRAI' M Y9 = NUMDE X1  SØRTIE FIN
    SVT ENS               Y3 X3 X2
   SI Z5 = 'EØE' ALØRS SØRTIE FIN
   REFAIRE FIN FIN
   SI Z5 = 'EØE' ALØRS
    SVT FDC               Y2 X2 X1
   FIN
   SI Z4 = 'VRAI' ØU Z5 = 'EØE' ALØRS SØRTIE FIN
   REFAIRE FIN FIN
   SI Z5 = 'EØE' ALØRS
    SVT ETUDIANT          Y1 X1 XO
   FIN
   SI Z4 = 'VRAI' ØU Z5 = 'EØE' ALØRS SØRTIE FIN
   REFAIRE FIN
SI Z5 = 'EØE' ØU Z4 = 'FAUX' ALØRS M Z5 = 'EØR'              FIN
:FDEF
```

```
/* ACCESSEUR SELECTIF EN ECRITURE */

:DEFMAC WRITEETUDIANT          :EXP
   M X10 = FØRMAL UETUDIANT          DANS BUFFER 1
   M Z4 = 'FAUX'  M Z5 = 'EØE'
 FAIRE SI EXISTE UN ETUDIANT          X1 Y1 DE X0 ALØRS
 FAIRE SI EXISTE UN FDC               X2 Y2 DE X1 ALØRS
 FAIRE DEPUIS Y3
       SI EXISTE UN ENS  X3 AYANT CØDE-ENS = Z1 ; DE X2 ALØRS
  SI Y9 = 0 ALØRS
 M NØ                DE X1  = NØ                DE X10
 M NØM               DE X1  = NØM               DE X10
 M INSEE             DE X1  = INSEE             DE X10
 M AN                DE X2  = AN                DE X10
 M ETAPE             DE X2  = ETAPE             DE X10
 M CØDE-ENS          DE X3  = CØDE-ENS          DE X10
    SVT ENS               Y3 X3 X2
   M Z4 = 'VRAI' SØRTIE SINØN M Y9 = Y9        - 1 FIN FIN
    SVT ENS               Y3 X3 X2
   SI Z5 = 'EØE' ALØRS SØRTIE FIN
   REFAIRE FIN FIN
   SI Z5 = 'EØE' ALØRS
    SVT FDC               Y2 X2 X1
   FIN
   SI Z4 = 'VRAI' ØU Z5 = 'EØE' ALØRS SØRTIE FIN
   REFAIRE FIN FIN
   SI Z5 = 'EØE' ALØRS
    SVT ETUDIANT          Y1 X1 X0
   FIN
   SI Z4 = 'VRAI' ØU Z5 = 'EØE' ALØRS SØRTIE FIN
   REFAIRE FIN
SI Z5 = 'EØE' ØU Z4 = 'FAUX' ALØRS M Z5 = 'EØR'          FIN
:FDEF

/* FIN DE FICHIER SØURCE ACCESSEURS */



CIN
GØ MACRØ
S SVT
P SVT
CIN
GØ BATCH
S   GETETUDIANT
P   GETETUDIANT
S   PUTETUDIANT
P   PUTETUDIANT
S  READETUDIANT
P  READETUDIANT
S WRITEETUDIANT
P WRITEETUDIANT
L
END
CIN
LØGØUT
!FIN
```

IL CØNVIENT DE NØTER QUE DANS LA VERSIØN ACTUELLE DE SØCRATE IL EST IMPØSSIBLE DE DEFINIR LES FØRMAL ASSØCIES A UN ACCESSEUR DE FACØN AUTØMATIQUE . ILS DØIVENT ETRE DEFINIS ET CØMPILES A LA MAIN DANS LA STRUCTURE DE LA BASE . CETTE RESTRICTIØN EST ELIMINEE DANS LA VERSIØN 2 DU PRØDUIT, QUI SERA DISPØNIBLE CØURANT 1977 .

ANNEXE 7.6. URANUS : Interprétation du langage relationnel

LE MØDULE D' INTERPRETATIØN DU LANGAGE DE DEFINITIØN FØNCTIØNNE EN DEUX PASSES :

- UNE PREMIERE PASSE D' ANALYSE SYNTAXIQUE DES REQUETES ET DE GENERATIØN DE L' ARBRE D' INTERPRETATIØN (MØDULE 'ANR2'),

- UNE DEUXIEME PASSE D' INTERPRETATIØN DE L' ARBRE GENERE CI-DESSUS (MØDULE 'INTERPRETE').

L' ANALYSEUR GENERE PØUR CHAQUE ELEMENT DE REQUETE UN ELEMENT DE L' ARBRE. LE FØRMAT DE CELUI-CI EST :

```
DCL 1 A (50) EXT ,
    2 AFILS  FIXED BIN (15) ,
    2 AFRERE FIXED BIN (15) ,
    2 AREL   FIXED BIN (15) ,
    2 ACØDE  FIXED BIN (15) ,
    2 ATYPE  CHAR (1) ;
```

LES ELEMENTS DE A SØNT CARACTERISES PAR LEUR TYPE :

- '0' : ØPERATEUR (SELECT, JØIN, MØDIFY, ...)
- 'R' : RELATIØN (RACINE D' UN SØUS-ARBRE)
- 'X' : VARIABLE LIBRE XI (PØUR DELETE ET PRØJECT)
- 'Q' : " " (AUTRES ØPERATEURS)
- 'C' : CØNNECTEUR (= ,>=, ¬=, ...)
- 'F' : FØNCTIØN (MAX, MIN, SUM, AVG, ...)
- 'K' : CØNSTANTE CHAINE DE CARACTERES
- 'N' : CØNSTANTE NUMERIQUE
- '?' : CØNSTANTE EXTERNE (EN CØNVERSATIØNNEL)
- 'D' : DØMAINE DE RELATIØN
- 'P' : PREDICAT

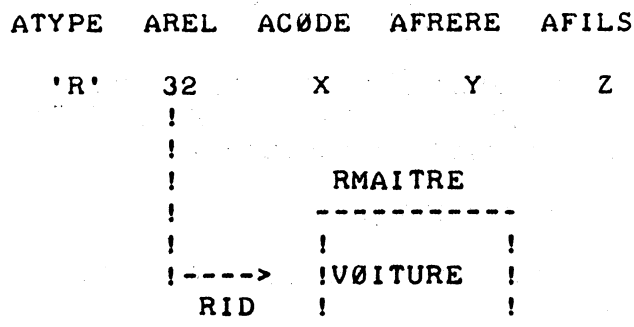
LE MØDULE 'ANR2' CØMPØRTE 5200 INSTRUCTIØNS ET ØCCUPE GLØBALEMENT (DØNNEES EXTERNES EXCLUSES) 6500 MØTS MACHINE.

7.6.1 ELEMENTS DE TYPE 'R'.

ILS DENØTENT L' IDENTIFICATIØN DE LA RELATIØN A LAQUELLE S' APPLIQUE UNE ØPERATIØN ELEMENTAIRE (SELECT,PRØJECT, ...) LØRSQUE ELLE EST CITEE EXPLICITEMENT EN ØPERANDE . EXEMPLE :

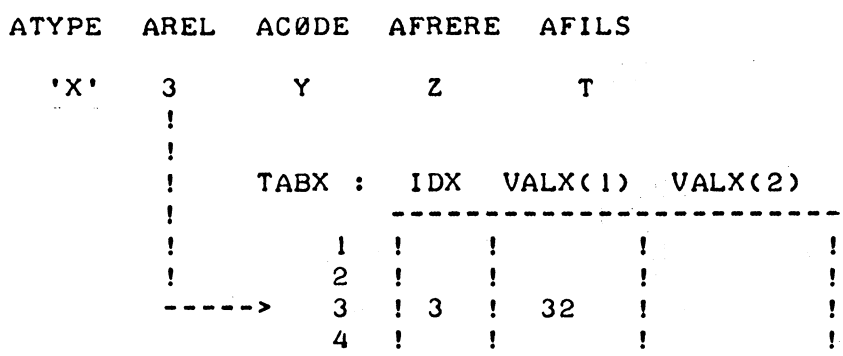
```
SELECT (VØITURE,NUMERØ = 12) ;
```

SI LE RELID DE LA RELATIØN VØITURE (CAD SØN INDICE DE DESCRIPTEUR DANS RMAITRE) EST 32, LA CØNFIGURATIØN SERA :



7.6.2 ELEMENT DE TYPE 'X'.

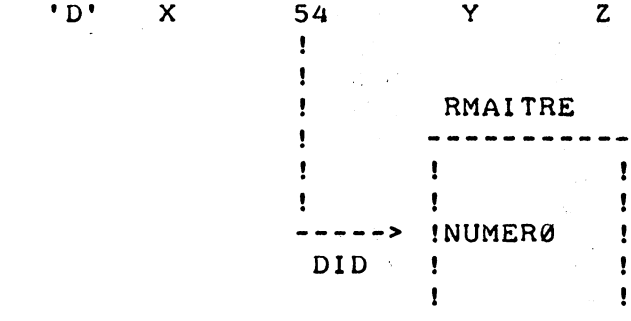
SI LA RELATIØN EST SPECIFIEE PAR UN XI, LE CHAMPS AREL CØNTIENT L' INDICE I DE L' ELEMENT DE TABX DEFINISSANT CE XI. PAR EXEMPLE SI X3 EST UNE REFERENCE A LA RELATIØN VØITURE DE RELID 32, LA STRUCTURE DE LA PILE ET DE TABX SERA :



7.6.3 ELEMENT DE TYPE 'D'.

ILS INDIQUENT LE DID D' UN DØMAINE, CAD L' INDICE DE SØN DESCRIPTEUR DANS RMAITRE.

SUR L' ØPERATIØN PRECEDENTE, SI LE DID DE NUMERØ EST 54 DANS RMAITRE, LA CØNFIGURATIØN SERA :



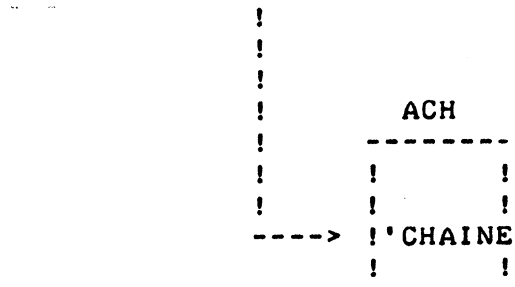
7.6.4 ELEMENT DE TYPE 'K'.

ILS INDIQUENT L'APPARITIØN D' UNE CØNSTANTE DE TYPE CAINE ALPHANUMERIQUE DANS LA REQUETE ANALYSEE . CES DERNIERES SØNT TØUTES MEMØRISEES SEQUENTIELLEMENT DANS UN TABLEAU :

```
DCL  ACH (20) CHAR (16) VAR EXT ;
```

LE CHAMPS ACØDE DE L' ARBRE D' INTERPRETATIØN CØNTIENT ALØRS L' INDICE DANS ACH DE LA CHAINE :

ATYPE	AREL	ACØDE	AFRERE	AFILS
'K'	X	4	Y	Z



7.6.5 ELEMENT DE TYPE 'N'.

ILS INDIQUENT LES CØNSTANTES NUMERIQUES . LE CHAMPS ACØDE DE L' ARBRE CØNTIENT LEUR VALEUR BINAIRE :

ATYPE	AREL	ACØDE	AFRERE	AFILS
'N'	X	VAL	Y	Z

7.6.6 ELEMENT DE TYPE 'Ø'.

ILS CØNSTITUENT TØUJØURS LA RACINE D' UN SØUS-ARBRE. LE CHAMPS ACØDE CØNTIENT LE CØDE DE L' ØPERATEUR CØNSIDERE, SØIT :

1	';'	16	SØME
2	','	17	MAX
3	'('	18	MIN
4	')'	19	SUM
5	'?'	20	AVG
6	'&'	21	CARD
7	'!'	22	PRØD
8	'='	23	JØIN
9	'>='	24	INSERT
10	'<='	25	MØDIFY
11	'>'	26	DELETE
12	'<'	27	SELECT
13	'¬='	27	SELECT
14	':='	28	PRØJECT
15	ALL	29	THEN
		30	ELSE

CES VALEURS SØNT MEMØRISEES DANS LE TABLEAU :

```
DCL 1 MØTCLE (30) EXT ,
    2 LGCLE  FIXED BIN (15) ,
    2 VALCLE CHAR (7) VAR ;
```

IL EST UTILISE PØUR L' ANALYSE LEXICØGRAPHIQUE DES REQUETES (RØUTINE 'LEXICØ' DU MØDULE ANR2).

ANNEXE 7.7. Exemple de session conversationnelle

Exemple réalisé sur des données réelles .

```
URANUS STARTING...
->
   $INIT BASE
LGCADRE = 1024, MAXCADRE = 2, TBASE = 64 K
BASE INITIALISED
->
   $GO  TIME
->
   $BASE
BASE NOTRACE TIME
->
   $DEF
->

  **** DEFINITION D' UNE RELATION SUR UNE PREMIERE BASE SOCRATE ****
->
      PROFESSEUR REL 10 IDEM ENSEIGNANT DANS SOC1
->
          DEBUT
->
           NOM MOT 30 CLE IDEM NOM
->
           SALAIRE DE 0 A 9999 IDEM INDICE
->
           COURS DE 0 A 99999 IDEM SYGESCO CLE DE ENSEIGNEMENT
->
           ETABLISSEMENT MOT 10
->
          FIN
 .... FIN DE REL

 .... RELATION CATALOGUEE : PROFESSEUR

 .... RELATION CATALOGUEE : #BASE

 .... PARAMETRES DE SOC1(STRUCT,SOURCE,ESPR,SUSP,SSTRUCT,USER,ACCNO,PW
       P:STRUCT,U:SOURCE,P:ESPR,P:SUSP,:STRUCT,NGT,2236,AS ;
->


  **** DEFINITION SUR UNE 2 EME BASE SOCRATE ****
->
     ETUDIANT REL 10 IDEM ETUDIANT DANS SOC2
->
        DEBUT
->
           NUMERO DE 0 A 9999999 IDEM NO
->
           NOM MOT 23 IDEM NOM
->
```

```
          INSEE  MOT 16 IDEM INSEE
->
          ANNEE DE 0 A 99 IDEM AN DE FDC
->
          ETAPE  DE 0 A 20000 IDEM ETAPE DE FDC
->
          ENS    DE 0 A 20000 IDEM CODE-ENS CLE DE ENS DE FDC
->
        FIN
 .... FIN DE REL

 .... RELATION CATALOGUEE : ETUDIANT

 .... PARAMETRES DE SOC2(STRUCT,SOURCE,ESPR,SUSP,SSTRUCT,USER,ACCNO,PW ) ?
          H:STRUCT,H:SOURCE,H:ESPR2,H:SUSP,:STRUCT,N,0,Q ;
->


       ENS REL 50 IDEM SGS DANS SOC1
->
          DEBUT
->
             CODE DE 0 A 99999 IDEM CODE
->
             TITRE MOT 30
->
          FIN
 .... FIN DE REL

 .... RELATION CATALOGUEE : ENS
->


       COMITE REL 10 IDEM COMITE DANS SOC1
->
          DEBUT
->
             LIBELLE MOT 30 IDEM LIBELLE
->
          FIN
 .... FIN DE REL

 .... RELATION CATALOGUEE : COMITE
->


       GRADE REL 20 IDEM GRADE DANS SOC1
->
          DEBUT
->
             CODE DE 0 A 9999 IDEM CODE
->
             LIBELLE MOT 30 IDEM LIBELLE
->
          FIN
 .... FIN DE REL

 .... RELATION CATALOGUEE : GRADE
```

```
->
    $GO

.... RELATION CATALOGUEE : #SOC

 .... ACCESSEUR(S) GENERE(S) :

  -          GETPROFESSEUR      .
  -          PUTPROFESSEUR      .
  -         READPROFESSEUR      .
  -        WRITEPROFESSEUR      .

  -          GETETUDIANT        .
  -          PUTETUDIANT        .
  -         READETUDIANT        .
  -        WRITEETUDIANT        .

  -          GETENS             .
  -          PUTENS             .

  -          GETCOMITE          .
  -          PUTCOMITE          .

  -          GETGRADE           .
  -          PUTGRADE           .

 .... FIN DE GENERATION .

00 MIN 10 SEC 640 MILI
->

  **** EDITION DE LA RELATION DESCRIPTEUR DES BASES SOCRATE UTILISEES ****
->
          #BASE;
#STRUCT   #SOURCE   #ESPR     #SUSP      #SSTRUCT      #USER      #AC
 P:STRUCT   U:SOURCE   P:ESPR     P:SUSP     :STRUCT      NGT        2
 H:STRUCT   H:SOURCE   H:ESPR2    H:SUSP     :STRUCT      N          0
00 MIN 00 SEC 450 MILI
->

    $SOC

->
  **** INITIALISATION SEQUENTIELLE DE LA RELATION GRADE ****
->
          GET GRADE;

BASE OUVERTE : SOC1

       20 N-UPLETS TRANSFERES, BASE FERMEE .
```

```
->
  **** INITIALISATION SEQUENTIELLE DE LA RELATION COMITE ****
->
          GET COMITE;
BASE OUVERTE : SOC1
       10 N-UPLETS TRANSFERES, BASE FERMEE .
->

  **** INITIALISATION DES 6 PREMIERS N-UPLETS DE LA RELATION ****
->
  **** PROFESSEUR A PARTIR DE LA 10 EME ENTITE SOCRATE        ****
->
          GET PROFESSEUR,10,6;
BASE OUVERTE : SOC1
        6 N-UPLETS TRANSFERES, BASE FERMEE .
->

  **** INITIALISATION DES 5 PREMIERS N-UPLETS DE LA RELATION ****
->
  **** ETUDIANT A PARTIR DE LA 20 EME ENTITE SOCRATE          ****
->
          GET ETUDIANT,20,5;
BASE OUVERTE : SOC2
        5 N-UPLETS TRANSFERES, BASE FERMEE .
->

  **** INITIALISATION DES 2 N-UPLETS SUIVANTS A PARTIR DES  ****
->
  **** ENTITES SOCRATE REPONDANT AU FILTRE                  ****
->
          READ ETUDIANT,ENS=1960,25,2;
BASE OUVERTE : SOC2
        2 N-UPLETS TRANSFERES, BASE FERMEE .
->

                                                            ****
  **** IDEM SUR LA RELATION PROFESSEUR
->
          READ PROFESSEUR,COURS=1960,110,3;
BASE OUVERTE : SOC1
        3 N-UPLETS TRANSFERES, BASE FERMEE .
->
```

```
     $GO
00 MIN 92 SEC 260 MILI
->

  **** EDITION DE LA RELATION ETUDIANT ****
->
           ETUDIANT ;
NUMERO     NOM                    INSEE                ANNE  ETAPE    ENS
                                                       E

 7525416   THIEBAUD               157012505619 9       77    01246    01963
 7525416   THIEBAUD               157012505619 9       80    02056    02984
 7600127   ABBE                   158117401 0033       80    01246    01963
 7525363   ABDELKADER             155127027 0033       72    01246    01963
 7451072   ABDERHALDEN            255077403 1071       76    00257    02165
 7564533   ABUET                  156117511 4523       78    00239    01963
 7200673   ABBERISSET             156026622 5567       77    00257    01963
 7603356   ABELARD                253276459 345        77    01246    01963
 7603356   ABELARD                253276459345         77    01246    01963
 7603356   ABELARD                253276459345         77    01646    03624
00 MIN 00 SEC 770 MILI
->


  **** EDITION DE LA RELATION PROFESSEUR ****
->
           PROFESSEUR ;
NOM                               SALAIR  COURS    ETABLISSEMENT
                                  E

 DECAUWERT J.MARC                 0503    00213    ..........
 TURCOT ALINE                     0593    00333    ..........
 TURCOT ALINE                     0593    00331    ..........
 TURCOT ALINE                     0596    00581    ..........
 TURCOT ALINE                     0593    00309    ..........
 GERBER ROBERT                    1173    00349    ..........
 GERBER ROBERT                    1173    01413    ..........
 JOL RENE                         0369    01963    ..........
 JOL RENE                         0369    00328    ..........
 DECAUS ALAIN                     0793    00237    ..........
00 MIN 00 SEC 590 MILI
->


  **** CALCULER LA RELATION J DONNANT TOUS LES ETUDIANTS ET TOUS LES ****
->
  **** PROFESSEUR AYANT MEME CODE ENSEIGNEMENT                      ****
->
           J:=JOIN(ETUDIANT,PROFESSEUR,ENS=COURS); J;
00 MIN 47 SEC 130 MILI
NUMERO.ET NOM.ETUDIANT               INSEE.ETUDIANT       ANNE  ETAPE.E  ENS.ET
UDIANT                                                    E.ET  TUDIANT  DIANT
 7525416   THIEBAUD               157012505619 9       77    01246    01963
 7600127   ABBE                   158117401 0033       80    01246    01963
 7525363   ABDELKADER             155127027 0033       72    01246    01963
 7564533   ABUET                  156117511 4523       78    00239    01963
 7200673   ABBERISSET             156026622 5567       77    00257    01963
 7603356   ABELARD                253276459345         77    01246    01963
 7603356   ABELARD                253276459345         77    01246    01963
00 MIN 00 SEC 320 MILI
->
```

```
  **** AJOUT D' UN CONSTITUANT ET SUPPRESSION DE CONSTITUANT ****
->
  **** SUR LA RELATION ETUDIANT                                  ****
->
     $DEF
->
           EFFECTIF REL 20
->
              DEBUT
->
                 NUMERO DE 0 A 9999999
->
                 NOM    MOT  16
->
                 ENS    DE 0 A 99999
->
                 GROUPE-TP DE 0 A 99
->
              FIN
 .... FIN DE REL

 .... RELATION CATALOGUEE : EFFECTIF
->


     $GO
00 MIN 01 SEC 260 MILI
->
  **** INITIALISATION DE CETTE RELATION A PARTIR DE LA RELATION ****
->
  **** ETUDIANT
->
          EFFECTIF := ETUDIANT ;
00 MIN 04 SEC 210 MILI
->


  **** EDITION DE EFFECTIF ****
->
          EFFECTIF ;
NUMERO     NOM                    ENS      GROU
                                           PE-T
 7525416   THIERAUD                01962    ..
 7525416   THIERAUD                02934    ..
 7600127   ABBE                    01963    ..
 7525366   ABDELKADER              01960    ..
 7451072   ABDERHALDEN             02165    ..
 7564533   ABUET                   01962    ..
 7200673   ABBERISSET              01962    ..
 7603356   ABELARD                 01960    ..
 7603356   ABELARD                 01960    ..
 7603356   ABELARD                 03624    ..
00 MIN 00 SEC 710 MILI
->
```

```
     $SOC
->
  **** REPORT DES MODIFICATIONS EFFECTUEES SUR ETUDIANT ****
->
          WRITE ETUDIANT;

BASE OUVERTE : SOC2

        0 N-UPLETS TRANSFERES, BASE FERMEE .
->

  **** VIDER LA RELATION ETUDIANT ****
->
     $P ETUDIANT
00 MIN 02 SEC 930 MILI
ETUDIANT PURGED.
00 MIN 00 SEC 395 MILI
->


  **** LISTE DE TOUTES LES RELATIONS PRESENTES DANS LE SYSTEME ****
->
     $LR
#MAITRE          (#NOM #TYPE #LONGUEUR #DEGRE #NBMAX #NB #CLE #RELINVERSE
TYPE             (#TYPE )
PROFESSEUR       (NOM SALAIRE COURS ETABLISSEMENT )
#PROFESSEUR      (NOM #VAL #LINK )
#BASE            (#STRUCT #SOURCE #ESPR #SUSR #SSTRUCT #USER #ACCNO #PASSW
ETUDIANT         (NUMERO NOM INSEE ANNEE ETAPE ENS )
ENS              (CODE TITRE )
COMITE           (LIBELLE )
GRADE            (CODE LIBELLE )
#SCC             (CITATION XI NIV PERE CLE NUMDE )
J                (NUMERO.ETUDIANT NOM.ETUDIANT INSEE.ETUDIANT ANNEE.ETUDIA
                 SALAIRE.PROFESSEUR COURS.PROFESSEUR ETABLISSEMENT.PR)
E                (NUMERO NOM INSEE ANNEE ETAPE ENS )
P                (NOM SALAIRE COURS ETABLISSEMENT )
EFFECTIF         (NUMERO NOM ENS GROUPE-TP )
00 MIN 00 SEC 390 MILI
->


  **** FIN DE LA SESSION ****
->
     $OFF
BASE SAVED
00 MIN 04 SEC 160 MILI


    E/S VIRT. :          4418 LECT.,          347 ECRIT.
    E/S REEL. :           237 LECT.,          136 ECRIT.


*** URANUS TERMINATED ***
```

ANNEXE 7.8. Interface PL/1-Socrate sous SIRIS 8

Centre Interuniversitaire de Calcul de Grenoble

note technique

Interface PL/1 - Socrate
sous SIRIS 8

NGUYEN GIA TOAN

N°T37 Avril 1976

Ce travail a été financé par l'Administration de l'Université Scientifique et Médicale de Grenoble dans le cadre des Services de Gestion de Scolarité.
Sa compréhension nécessite la connaissance de la terminologie Socrate citée en bibliographie.

I - PRESENTATION

Il est possible d'accéder à une base de données Socrate via un programme écrit en langage PL/1 et d'utiliser toutes les facilités de traitement de celui-ci, en particulier pour l'édition et le traitement de données alphanumériques. Ceci nécessite certaines conventions de liaison avec la méthode d'accès Socrate, analogues à l'interface Cobol-Socrate ([1]), qui sont décrites ci-après.

L'interface se compose de quatre primitives permettant de dialoguer avec ce système :

1- Ouverture d'une base, primitive SSOPEN
2- Appel d'un sous-programme batch Socrate, SSGBD,
3- Création d'un point de reprise, PSAVE,
4- Fermeture d'une base, SSCLOSE.

a/ Ouverture d'une base

Elle permet de définir un certain nombre de paramètres de traitement :

- type d'opérations (interrogation, création, mise à jour de la base),
- paramètres d'exécution des sous-programmes Socrate appelés :
  . nombre de buffer programme,
  . nombre de buffer données,
  . nombre de repères utilisés (variables X_i),

EX : CALL SSØPEN (INTERF , 5 , 10 , 10 , 8);

b/ Appel de sous-programmes Socrate

Avec la version 1 de Socrate, tout traitement à partir de PL/1 doit d'abord demander l'exécution d'un sous-programme précompilé dans Socrate. L'exécution de celui-ci est lancée par l'appel à la primitive SSGBD, le résultat en étant la transmission à PL/1 des données obtenues dans des buffers utilisateurs.

Il faut donc fournir à Socrate les paramètres d'exécution de ces sous-programmes :

- leur nom,
- les noms des zones où sont récupérées les résultats,
- les variables X_i, Y_i et Z_i Socrate que l'on désire aussi récupérer dans PL/1.

EX : CALL SSGBD (INTERF, BUFF, LGBUFF, 1, 'INTERROG', 0, 0);

c/ Prise d'un point de cohérence

Elle permet de sauvegarder sur des fichiers "journaux" l'état courant de la base de données pour des raisons de sécurité.

EX : CALL PSAVE (INTERF);

d/ Fermeture de la base

Elle permet de terminer le traitement d'une base et de restituer les fichiers dans un état inactif cohérent.

EX : CALL SSCLOSE (INTERF);

II - MISE EN OEUVRE

1/ Le dialogue PL/1 - Socrate nécessite la définition d'une table d'interface transmise à Socrate, comportant sous forme de structure PL/1 [3], la définition :

- de l'espace réel de la base de données,
- du nom de l'utilisateur déclaré dans la base,
- de son numéro de compte,
- de la sous-structure utilisée pour travailler.

EX :
```
DCL   1  INTERF
      2  ØPL CHAR (4) INIT ('STR'),
      2  KEY CHAR (8) INIT (':STRUCT'),
      2  USER CHAR (8) INIT ('NGT'),
      2  ACCNO CHAR (4) INIT ('0286'),
      2  PASSW CHAR (8) INIT ('7XY'),
      2  CODANO FIXED BIN (31),
      2  SSPG CHAR (30),
      2  FILL CHAR (58);
```

Le format des éléments est impératif, leur nom est quelconque; CODANO et SSPG sont utilisés par Socrate en cas d'anomalie de traitement pour renvoyer le nom du sous-programme ayant provoqué ou détecté une erreur, et le code erreur [1] et [2]; Fill est une zone réservée système.

- ØPL est l'opl de la carte assign définissant le fichier base Socrate.

EX:
```
!ASSIGN    STR, FIL, (NAM, BASESDC) ,(STS , MOD)
```

- KEY : nom de la structure Socrate utilisée ou ':STRUCT' si toute la structure est désirée.

- USER : nom de l'utilisateur déclaré dans l'en-tête de définition de la structure Socrate.

- ACCNO : id

- PASSW : id

EX : base comportant l'en-tête

```
ß PRØG    0287  (NGT(7XY))
ß STRU
        ENTETE  1000  PERSONNE
          DEBUT
            .   NOM   MOT
            :
           etc ...
```

Le nom cité en USER doit de plus être obligatoirement rappelé dans le JOB-id de la carte job si l'on travaille en batch.

EX : !JOB NGT , 0286 , 6414 : 286

2/ Ouverture de la base

Il faut déclarer le point d'entrée externe SSOPEN

EX : DCL SSOPEN ENTRY ;

Appel du module :

EX : CALL SSOPEN (<table>, <typ>, m, o, p, q);

où :

<table> = nom de la table d'interface (ici : INTERF),

<typ> = 'I' , 'Ø', 'u' selon le traitement effectué (interrogation, création, mise à jour).

m = entier ≤ 10 ou variable de type FIXED BIN (31)
nombre de repères ou points courants Socrates utilisés.

o = entier ou variable FIXED BIN (31)
nombre de cadres programmes de 2 Koctets nécessaires à l'exécution des sous-programmes Socrate.

p = entier ou variable FIXED BIN (31)
nombre de cadres donnés de 2K octets qui réceptionnent à l'exécution les pages de données.

q = entier ou variable FIXED BIN (31)
nombre de cadre structure Socrate, contenant à l'exécution les éléments de structure interne Socrate nécessaire à l'accès aux données.

Par défaut, les valeurs prises sont m = 10 ; O = ; p = 10;
q = 1 ; o,p et q influent considérablement sur les temps de traitement, dûs à la pagination mise en oeuvre par Socrate. On a intérêt à les prendre assez grands (10 cadres pour chaque paramètre par ex.).

3/ Appel de sous-programmes Socrate

Après déclaration du point d'entrée externe SSGBD, il faut préciser toutes les variables Socrate que l'on désire conserver d'un appel sur l'autre. Ceci est utile par exemple pour effectuer des traitements séquentiels sur toutes les réalisations d'une même entité.

On peut ainsi mémoriser les variables Socrate X_i, Y_i et Z_i après exécution d'un sous-programme batch initialisant un buffer avec des caractéristiques simples d'une entité;.

EX : CALL SSGBD (INTERF, BUFF, LGBUFF, up, prog1,...
progn, nx, X_1,...X_n, n_y, Y_1,...
Y_n, n_z, Z_1,...,
Z_n);

Cet appel provoque :

1) L'initialisation des variables Socrate X_1,...X_n, Y_1,...Y_n, Z_1, Z_n avec les valeurs déclarées dans le programme PL/1,

2) L'exécution séquentielle des np programmes batch Prog1,... progn,

3) Le retour à PL/1 avec mise-à-jour des variables X_i, Y_i, Z_i avec leur valeur courante dans la base de données;
np : entier ou variable FIXED BIN (31), nombre de $prog_i$ qui suivent ,

nx, ny : entiers ≤ 10 ou nuls ou variables FIXED BIN (31), nombre de X_i et Y_i qui suivent respectivement,

nz : entier ≤ 5 ou nul ou variable FIXED BIN (31), nombre de Z_i qui suivent.

<$prog_i$>= chaîne de caractères ≤ 30 ou variable CHAR (30)
nom de sous-programme Socrate à exécuter.

<X_i> = entier ≤ 10 ou variable FIXED BIN (31)
indice de point

<Y_i> = Variable FIXED DEC (15) dans lesquelles sont mises les valeurs des variables Y_i Socrate après chaque exécution de sous-programme.

<Z_i> = Chaîne CHAR (30) dans lesquelles seront mises les variables Z_i Socrate.

<buff> = variable CHAR (n) n quelconque mise à jour par les sous-programmes Socrate à l'aide de FORMAC.

= Longueur utile dans <buff> disponible au sous-programme.

INTERF = Nom de la table d'interface PL/1 - Socrate.

EX : CALL SSGBD (INTERF, BUFF, 140, 1, 'INTERROG',0, 0,0);

4/ Fermeture de la base, prise d'un point de cohérence

```
DCL   (SSCLOSE, PSAVE) ENTRY;
CALL  SSØPEN (INTERF);
CALL  PSAVE (INTERF);
```

5/ A l'édition de liens

Ajouter les cartes assign correspondant :

1/ au module d'interface PL/1-Socrate :

```
!ASSIGN   BIB1, FIL, (NAM, PL1:SOC),(UNT,AC,0286),(STS,OLD)
```

2/ Eventuellement, a la bibliothèque PL1 :

```
!ASSIGN   BIB2, FIL, (NAM, PL1LIB),(UNT,AC,:SYS),(STS,OLD)
```

3/ A la méthode d'accès batch Socrate :

```
!ASSIGN  GØ , FIL , (NAM, BO-SOCRATE),(UNT,AC,0001)(STS,OLD)
```

4/ A l'arbre de segmentation Socrate :

```
!ASSIGN  TREE, FIL, (NAM,TREE-COBSOC),(UNT,AC,0001);(STS,OLD)
```

Ajouter au link l'option :

```
!ØPTION          (UNSAT,BIB1,BIB2)
!TREE,FIL
```

6/ A l'exécution

Il faut mettre les cartes assign correspondant à tous les fichiers nécessaires à Socrate [2] , c'est-à-dire :

- fichier base (avec le même opl que celui déclaré dans la structure, ici 'STRU')
- les fichiers de structure interne Socrate (d'opl imposé par la table d'interface, ici 'STR').

III - EXEMPLE

A- Programme PL/1 d'appel

La table d'interface est "INTERF". A l'exécution, il faudra mettre une carte !ASSIGN STR, FIL, (NAM,<base>) pour atteindre le fichier structure Socrate. On utilise toute la structure (KEY = ":STRUCT"). Celle-ci doit comporter comme numéro de compte "0286", comme nom d'utilisateur "NGT". Comme mot de passe "7".

On appelle le sous-programme batch Socrate "CREBASE" (paramètre PROGRAM de CALL SSGBD).

La base est utilisée en interrogation (TYP = "I").

```
STMT LEVEL  NEST                                                    PL
   1                PLISOC : PROC OPTIONS (MAIN) ;
   2     1          DCL 1 INTERF ,
         1            2 OPL CHAR (4) INIT ('STR ') ,
         1            2 STRUCT CHAR (12) INIT (':STRUCT      ') ,
         1            2 USER CHAR (8) INIT ('NGT') ,
         1            2 ACCNO CHAR (4) INIT ('0286') ,
         1            2 PASS CHAR (8) INIT ('7') ,
         1            2 CODE FIXED BIN (31) ,
         1            2 SSPG CHAR (30) ,
         1            2 FILL CHAR (58) ;
   3     1          /* */
         1          DCL TYP CHAR (1) INIT ('I') ;
   4     1          DCL NPC FIXED BIN (31) INIT (10) ;
   5     1          DCL NCP FIXED BIN (31) INIT (10) ;
   6     1          DCL NCD FIXED BIN (31) INIT (10) ;
   7     1          DCL NSI FIXED BIN (31) INIT (8) ;
   8     1          /* */
         1          DCL BUFF CHAR (140) INIT (' ') ;
   9     1          DCL LGBUF FIXED BIN (31) INIT (140) ;
  10     1          DCL PROGNAM CHAR (30) INIT ('CREBASE') ;
  11     1          DCL (PCOUR,NP,NX,NY,NZ) FIXED BIN (31) ;
  12     1          DCL (SSOPEN,SSCLOSE,SSGBD) ENTRY ;
  13     1          /* */
         1           NP = 1 ;
  14     1           CALL SSOPEN (INTERF,TYP,NPC,NCP,NCD,NSI) ;
  15     1           CALL SSGBD (INTERF,BUFF,LGBUF,NP,PROGNAM,0,0,0) ;
  16     1           CALL SSCLOSE (INTERF) ;
  17     1          /* */
         1          END PLISOC ;
```

B - Edition de liens

- BO-PLISOC est le fichier résultat de la compilation du programme PL/1.
- BO-SOCRATE-GO5 permet d'atteindre les sous-programmes de services Socrate.
- PL1 : SOC est le fichier contenant les routines d'interface SSGBD, SSCLOSE, etc...
- TREE-COBSOC est l'arbre de segmentation standard batch Socrate.
- Il faut préciser les options suivantes :

```
:OPTION          (UNSSAT, BIB1, BIB3)
:TREE,FIL
```

- Le fichier PLI-SOC est le résultat de l'édition de liens.

```
!ASSIGN BO,MTN,FIL,(NAM,BO-PLISOC),(STS,OLD)
!ASSIGN GO,FIL,(NAM,BO-SOCRATE-GO5),(STS,OLD),(UNT,AC,0001)
!ASSIGN LM,MTN,FIL,(NAM,PLI-SOC),(SIZ,10,1)
!ASSIGN BIB1,FIL,(NAM,PL1LIB),(STS,OLD),(UNT,AC,:SYS)
!ASSIGN BIB2,FIL,(NAM,COBLIB),(STS,OLD),(UNT,AC,:SYS)
!ASSIGN BIB3,FIL,(NAM,PL1:SOC),(STS,OLD)
!ASSIGN TREE,FIL,(NAM,TREE-COBSOC),(STS,OLD),(UNT,AC,0001)
!LINK
```

C - Exécution

- Le Job-id, ici "NGT" est obligatoirement celui de l'utilisateur cité dans INTERF.USER.
- Le fichier PLI-SOC est le résultat du LINK
- Le fichier X:STRUCT est la structure interne Socrate
- Le fichier X:SOURCE est le fichier dans source des programmes Socrate ('PROP' est l'étiquette logique déclarée dans la structure Socrate).
- Le fichier P:ESPR est l'espace réel de la base de données ("STRS" : même remarque que pour "PROP").

```
!JOB NGT,0286,6414:286
!LIMIT (CORE,90),(SPOI,80)
!ASSIGN LM,MTN,FIL,(NAM,PLI-SOC),(STS,OLD)
!ASSIGN STR,FIL,(NAM,X:STRUCT),(STS,MOD)
!ASSIGN PROP,FIL,(NAM,X:SOURCE),(STS,MOD)
!ASSIGN STR9,FIL,(NAM,P:ESPR),(STS,MOD)
!ASSIGN SSPD,FIL,(NAM,P:SUSP),(STS,MOD)
!DEBUG (LMN,PLI-SOC)
```

IV - REMARQUES

1/ Il n'est possible avec l'interface actuel de travailler qu'avec une seule base Socrate.

2/ Les quatre routines de l'interface sont écrites en assembleur (118 instructions au total) et font elles-mêmes appel à la méthode d'accès batch Socrate. Les noms SSOPEN, SSGBD, PSAVE et SSCASE sont donc réservés à cet effet s'ils sont déclarés en ENTRY dans le programme PL/1, avec assignation du fichier PL1:SOC.

3/ Les modules SSCLOSE et PSAVE font chacun 20 instructions machine, SSOPEN 36 instructions, et SSGBD 42 instructions.
La surcharge dûe à l'interface est donc extrêmement faible en regard du service offert. L'espace de travail utilisé par l'interface est de 89 mots.

4/ Lors des appels à SSGBD la détermination des paramètres est faite à l'aide des <up>,<nx>,<uy>. Mais si ceux-ci sont nuls, il est possible d'abréger l'écriture des instructions d'appel.
Ainsi, les appels suivants sont équivalents :

```
et  { CALL   SSGBD (INTERF,BUFF,LGBUFF,1,'INTERROG',0,0,0);
    { CALL   SSGBD (INTERF,BUFF,LGBUFF,1,'INTERROG');

et  { CALL   SSGBD (INTERF,BUFF,LGBUFF,1'PROG',0,1,Y1,0);
    { CALL   SSGBD (INTERF,BUFF,LGBUFF,1,'PROG',0,1,Y1);

et  { CALL   SSGBD (INTERF,BUFF,LG,1,'PROG',2,X1,X2,0,0);
    { CALL   SSGBD (INTERF,BUFF,LG,1,'PROG',2,X1,X2);
```

Toutefois, il faut spécifier l'absence d'une série de paramètres par 0 s'il existe d'autres paramètres derrière :

```
CALL   SSGBD (INTRF,BUFF,LG,1,'PROG',0,0,1,Z1);
```

5/ La version utilisée de Socrate lors de la mise au point : 1.5 .
La version utilisée de PL/1 lors de la mise au point : 4.1.

BIBLIOGRAPHIE

Brochures CII.

[1] 4338 E1 : Socrate manuel d'utilisation (version 1)

[2] 4553 E : Socrate manuel d'opérations, (version 1)

[3] : PL/1 manuel d'utilisation, (version 1)

[5] 4505 E : " " d'opérations.

BIBLIOGRAPHIE

[1] CODD E.F. : "A relational model for large shared data-banks". CACM. Juin 1970.

[2] CHAMBERLIN D.D. : "Relational Data-Base Management Systems". ACM Computing Surveys. Mars 1976.

[3] CII-HB : "Socrate sous Siris 7/8 : Manuel d'Utilisation".

[4] ADIBA M., DELOBEL C. : "Les modèles relationnels de bases de données". IRIA-Sefi. Avril 1976.

[5] BENCI, BODART et al. : " Relational Model for a Conceptual Schema". IFIP 76. Freudenstadt. Janvier 1976.

[6] NIJSSEN : "A gross Architecture for the next generation DBMS". Ibid.

[7] CHEN P.P.S. : "The Entity-Relationship Model : Toward a Unified view of Data". ACM TODS. Mars 1976.

[8] TODD S., HALL P. : "Relation and Entities". IFIP 76. Freudenstadt. Janvier 1976.

[9] ANSI/X3/SPARC : DBMS Study Group : "Interim Report". Décembre 1975.

[10] SENKO M.E. et al. : "Data Structuring and Accessing in Data-Base Systems". IBM System Journal. Janvier 1973.

[11] NAVATHE, MERTEN : "Investigation into the Application of the Relational Model to Data Translation". ACM Sigmod Workshop. San José. Mai 1975.

[12] LEONARD M. : "Aide Algorithmique à la Conception de Bases de Données". Thèse Docteur-Ingénieur. Grenoble. Juin 1976.

[13] NGUYEN G.T. : "Socrate V1.5 sous Siris 8 : Bilan d'une Expérience". Administration USMG. Février 1976.

[14] PIROTTE A. : "Comparaison de langages d'interrogation de bases de données relationnelles".
IRIA. Club Banques de Données. Juin 1976.

[15] CHAMBERLIN D.D. et al. : "SEQUEL2 : A unified Approach to data Definition, Manipulation and Control".
IBM Journal of Research and Development. Novembre 197

[16] U.S.M.G. : "Projet de contrat de Recherche en Informatique : Etude de l'Adaptabilité d'un système d'information à travers un exemple concret".
Grenoble. Mars 1975.

[17] PORTAL D. : "Conception d'un Système Automatisé de Gestion de la Scolarité de l'Enseignement Supérieur".
Thèse Docteur-Ingénieur. Grenoble. Mars 1975.

[18] CALECA J.Y., FORESTIER J.M. : "Interrogation simultanée de plusieu bases de données".
Rapport DEA. Grenoble. Juin 1976.

[19] ADIBA, ANDRE et al. : "POLYPHEME : Propositions pour un Modèle de Répartition et de Coopération de Bases de Données dans un Réseau d'Ordinateurs".
Rapport de Recherche 29, Grenoble, Décembre 1975.

[20] CII-HB : "Socrate sous Siris 7/8 : Manuel d'Opérations".

[21] BROOKS, CARDENAS et al. : "An approach to Data Communication between different Generalised DBMS".
Proc. Symp. "Very Large Data Bases". Bruxelles. Septembre 1976.

[22] ASTRAHAN, CHAMBERLIN et al. : "System R : a Relational Approach to Database Management".
ACM TODS. Juin 1976.

[23] U.S.M.G. : "Projet de contrat de recherche en Informatique : Intégration URANUS-POLYPHEME - Développement d'Applications Relationnelles".
Grenoble, Novembre 1977.

[24] BUBENKO : "The temporal Dimension in Information Modelling". IFIP TC2. Nice. Janvier 1977.

[25] NGUYEN G.T. : "URANUS : Spécifications de Réalisation". Contrat IRIA-Sesori 76027. Grenoble. Mars 1977.

[26] JORRAND P. : "Contribution au développement des langages extensibles". Thèse d'Etat. Grenoble. Janvier 1975.

[27] ADIBA M. : "MOGADOR : un Modèle Général de Données Réparties". Rapport de Recherche 81. Grenoble. Juillet 1977.

[28] SMITH M.S., SMITH D.C.P. : "Data base Abstractions : Aggregation and Generalization". ACM TODS. Juin 1977.

[29] CROCUS : "Systèmes d'exploitation des ordinateurs". DUNOD Ed. Paris. 1975.

[30] GRIFFITHS P.P., BRADFORD W.W. : "An Authorization Mechanism for a Relational Data-Base System". RJ 1721. IBM Research Lab. San José. Février 1976.

[31] DONOVAN J.J. : "Data Base Approach to Management Decision Support". ACM TODS. Décembre 1976.

[32] DATE C.J. : "An introduction to data-base systems". Addison-Wesley. 1975.

[33] CODD E.F. : "Further normalization of the data-base relational model". CCSS. New-York. Mai 1971.

[34] DELOBEL C. : "Contributions théoriques à la conception et l'évaluation d'un système d'informations appliqué à la gestion". Thèse d'Etat. Grenoble. Octobre 1973.

[35] DELOBEL C. : "Sémantique des relations et processus de décomposition dans le modèle relationnel".
Rapport de Recherche 45. Grenoble. Septembre 1976.

[36] SENKO M.E. : "Data structures and data accessing in data base systems : past, present, future".
IBM System Journal, Septembre 1977.

[37] ABRIAL J.R. : "Data Semantics".
IFIP TCL. Cargese. Avril 1974.

[38] SHU N.C., HOUSEL B.C. et al. : "EXPRESS : A Data Extraction, Processing and Restructuring System".
ACM TODS. Juin 1977.

[39] SCHMIDT J.W. : "Some High Level Language Constructs for Data of Type Relation".
ACM TODS. Septembre 1977.

[40] MOSSIERE J. : "Méthode pour l'écriture des systèmes d'exploitatio
Thèse d'Etat. Grenoble. Septembre 1977.

[41] DEMOLOMBE R. et al. : "SYNTEX 2". Rapport final. CERT. 1976.