Aides algorithmiques à la conception de bases de données

Michel Leonard

THESE

présentée à

UNIVERSITE SCIENTIFIQUE ET MEDICALE DE GRENOBLE

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

POUR OBTENIR LE GRADE DE

DOCTEUR-INGENIEUR

Michel LEONARD

AIDES ALGORITHMIQUES A LA CONCEPTION DE BASES DE DONNÉES.

Thèse soutenue le 28 Juin 1976 devant la Commission d'Examen :

Président : L. BOLLIET

Examinateurs : C. DELOBEL
F. PECCOUD
E. PICHAT

UNIVERSITE SCIENTIFIQUE ET MEDICALE DE GRENOBLE

Monsieur Michel SOUTIF : Président

Monsieur Gabriel CAU : Vice-Président

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

PROFESSEURS TITULAIRES

MM.	ANGLES D'AURIAC Paul	Mécanique des Fluides
	ARNAUD Paul	Chimie
	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
	BARNOUD Fernand	Biosynthèse de la Cellulose
	BARRA Jean-René	Statistiques
	BARRIE Joseph	Clinique Chirurgicale
	BEAUDOING André	Clinique de Pédiatrie et Puériculture
	BERNARD Alain	Mathématiques Pures
Mme	BERTRANDIAS Françoise	Mathématiques Pures
MM.	BERTRANDIAS Jean-Paul	Mathématiques Pures
	BEZES Henri	Pathologie Chirurgicale
	BLAMBERT Maurice	Mathématiques Pures
	BOLLIET Louis	Informatique (I.U.T. B)
	BONNET Georges	Electrotechnique
	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
	BRAVARD Yves	Géographie
	CABANEL Guy	Clinique Rhumatologique et Hydrologique
	CALAS François	Anatomie
	CARLIER Georges	Biologie Végétale
	CARRAZ Gilbert	Biologie Animale et Pharmacodynamie
	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
	CRAYA Antoine	Mécanique
Mme	DEBELMAS Anne-Marie	Matière Médicale
MM.	DEBELMAS Jacques	Géologie Générale
	DEGRANGE Charles	Zoologie
	DELORMAS Pierre	Pneumo-Phtisiologie
	DEPORTES Charles	Chimie Minérale
	DESRE Pierre	Métallurgie

MM.	DESSAUX Georges	Physiologie Animale
	DODU Jacques	Mécanique Appliquée (I.U.T. A)
	DOLIQUE Jean-Michel	Physique des Plasmas
	DREYFUS Bernard	Thermodynamique
	DUCROS Pierre	Cristallographie
	DUGOIS Pierre	Clinique de Dermatologie et Syphiligraphie
	GAGNAIRE Didier	Chimie Physique
	GALLISSOT François	Mathématiques Pures
	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
	KLEIN Joseph	Mathématiques Pures
	KOSZUL Jean-Louis	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
	LLIBOUTRY Louis	Géophysique
	LOISEAUX Pierre	Sciences Nucléaires
	LONGEQUEUE Jean-Pierre	Physique Nucléaires
	LOUP Jean	Géographie
Melle	LUTZ Elisabeth	Mathématiques Pures
	MALGRANGE Bernard	Mathématiques Pures
	BOUTET DE MONVEL Louis	Mathématiques Pures
	MALINAS Yves	Clinique Obstétricale
	MARTIN-NOEL Pierre	Seméiologie médicale
	MAZARE Yves	Clinique Médicale A
	MICHEL Robert	Minéralogie et Pétrographie
	MICOUD Max	Clinique Maladies Infectieuses
	MOURIQUAND Claude	Histologie
	MOUSSA André	Chimie Nucléaire
	MULLER Jean-Michel	Thérapeutique (Néphrologie)
	NEEL Louis	Physique du solide
	OZENDA Paul	Botanique
	PAYAN Jean-Jacques	Mathématiques Pures
	PEBAY-PEYROULA Jean-Claude	Physique
	RASSAT André	Chimie Systématique
	RENARD Michel	Thermodynamique
	RINALDI Renaud	Physique
	DE ROUGEMONT Jacques	Neuro-Chirurgie
	SEIGNEURIN Raymond	Microbiologie et Hygiène
	SENGEL Philippe	Zoologie
	SIBILLE Robert	Construction Mécanique (I.U.T. A)
	SOUTIF Michel	Physique Générale
	TANCHE Maurice	Physiologie

MM.	TRAYNARD Philippe	Chimie Générale
	VAILLANT François	Zoologie
	VALENTIN Jacques	Physique Nucléaire
	VAUQUOIS Bernard	Calcul Electronique
Mme	VERAIN Alice	Pharmacie Galénique
MM.	VERAIN André	Physique
	VEYRET Paul	Géographie
	VIGNAIS Pierre	Biochimie médicale
	YOCCOZ Jean	Physique Nucléaire Théorique

PROFESSEURS ASSOCIES

MM.	CLARK Gilbert	Spectrométrie Physique
	CRABBE Pierre	CERMO
	ENGLMAN Robert	Spectrométrie Physique
	HOLTZBERG Frédéric	Basses Températures
	ROST Ernest	Sciences Nucléaires

PROFESSEURS SANS CHAIRE

Melle	AGNIUS-DELORD Claudine	Physique Pharmaceutique
	ALARY Josette	Chimie Analytique
MM.	AMBROISE-THOMAS Pierre	Parasitologie
	BELORIZKY Elie	Physique
	BENZAKEN Claude	Mathématiques Appliquées
	BIAREZ Jean-Pierre	Mécanique
	BILLET Jean	Géographie
	BOUCHET Yves	Anatomie
	BRUGEL Lucien	Energétique (I.U.T. A)
	BUISSON René	Physique (I.U.T. A)
	CONTE René	Physique (I.U.T. A)
	DEPASSEL Roger	Mécanique des Fluides
	GAUTHIER Yves	Sciences Biologiques
	GAUTRON René	Chimie
	GIDON Paul	Géologie et Minéralogie
	GLENAT René	Chimie organique
	GROULADE Joseph	Biochimie 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.	KUHN Gérard	Physique (I.U.T. A)
	LE ROY Philippe	Mécanique (I.U.T. A)
	LUU DUC Cuong	Chimie Organique
	MAYNARD Roger	Physique du Solide
	PELMONT Jean	Biochimie
	PERRIAUX Jean-Jacques	Géologie et Minéralogie
	PSISTER Jean-Claude	Physique du Solide
Melle	PIERY Yvette	Physiologie Animale
MM.	RAYNAUD Hervé	M.I.A.G.
	REBECQ Jacques	Biologie (CUS)
	REVOL Michel	Urologie
	REYMOND Jean-Charles	Chirurgie Générale
	RICHARD Lucien	Biologie Végétale
Mme	RINAUDO Marguerite	Chimie Macromoléculaire
MM.	ROBERT André	Chimie Papetière

MM.	SARRAZIN Roger	Anatomie et Chirurgie
	SARROT-REYNAUD Jean	Géologie
	SIROT Louis	Chirurgie Générale
Mme	SOUTIF Jeanne	Physique Générale
MM.	STREGLITZ Paul	Anesthésiologie
	VIALON Pierre	Géologie
	VAN CUTSEM Bernard	Mathématiques Appliquées

MAITRES DE CONFERENCES ET MAITRES DE CONFERENCES AGREGES

MM.	AMBLARD Pierre	Dermatologie
	ARMAND Gilbert	Géographie
	ARMAND Yves	Chimie (I.U.T. A)
	BACHELOT Yvan	Endocrinologie
	BARGE Michel	Neuro chirurgie
	BARJOLLE Michel	MIAG
	BEGUIN Claude	Chimie organique
Mme	BERIEL Hélène	Pharmacodynamie
MM.	BOST Michel	Pédiatrie
	BOUCHARLAT Jacques	Psychiatrie adultes
Mme	BOUCHE Liane	Mathématiques (C.U.S.)
MM.	BRODEAU François	Mathématiques (I.U.T. B)
	BUTEL Jean	Orthopédie
	CHAMBAZ Edmond	Biochimie médicale
	CHAMPETIER Jean	Anatomie et Organogénèse
	CHARDON Michel	Géographie
	CHERADAME Hervé	Chimie Papetière
	CHIAVERINA Jean	Biologie Appliquée (EFP)
	COHEN-ADDAD Jean-Pierre	Spectrométrie Physique
	COLOMB Maurice	Biochimie Médicale
	CONTAMIN Charles	Chirurgie thoracique et cardio-vasculaire
	CORDONNIER Daniel	Néphrologie
	COULOMB Max	Radiologie
	CROUZET Guy	Radiologie
	CYROT Michel	Physique du solide
	DELOBEL Claude	M.I.A.G.
	DENIS Bernard	Cardiologie
	DOUCE Roland	Physiologie Végétale
	DUSSAUD René	Mathématiques (C.U.S.)
Mme	ETERRADOSSI Jacqueline	Physiologie
MM.	FAURE Jacques	Médecine Légale
	FAURE Gilbert	Urologie
	FONTAINE Jean-Marc	Mathématiques Pures
	GAUTIER Robert	Chirurgie Générale
	GENSAC Pierre	Botanique
	GIDON Maurice	Géologie
	GROS Yves	Physique (I.U.T. A)
	GUITTON Jacques	Chimie
	HICTER Pierre	Chimie
	IVANES Marcel	Electricité
	JALBERT Pierre	Histologie
	KOLODIE Lucien	Hématologie
	KRAKOWIAK Sacha	Mathématiques Appliquées
	LE NOC Pierre	Bactériologie-virologie
	LEROY Philippe	I.U.T. A
	MACHE Régis	Physiologie Végétale
	MAGNIN Robert	Hygiène et Médecine Préventive
	MALLION Jean-Michel	Médecine du Travail
	MARECHAL Jean	Mécanique (I.U.T. A)
	MARTIN-BOUYER Michel	Chimie (C.U.S.)

M.	MICHOULIER Jean	Physique (I.U.T. A)
Mme	MINIER Colette	Physique (I.U.T. A)
MM.	NEGRE Robert	Mécanique (I.U.T. A)
	NEMOZ Alain	Thermodynamique
	NOUGARET Marcel	Automatique (I.U.T.A)
	PARAMELLE Bernard	Pneumologie
	PECCOUD François	Analyse (I.U.T. B)
	PEFFEN René	Métallurgie (I.U.T. A)
	PERRET Jean	Neurologie
	PERRIER Guy	Géophysique - Glaciologie
	PHELIP Xavier	Rhumatologie
	RACHAIL Michel	Médecine Interne
	RACINET Claude	Gynécologie et Obstétrique
	RAMBAUD André	Hygiène et Hydrologie
	RAMBAUD Pierre	Pédiatrie
Mme	RENAUDET Jacqueline	Bactériologie
MM.	ROBERT Jean-Bernard	Chimie-Physique
	ROMIER Guy	Mathématiques (I.U.T. B)
	SHOM Jean-Claude	Chimie générale
	STOEBNER Pierre	Anatomie pathologique
	VROUSOS Constantin	Radiologie

MAITRE DE CONFERENCES ASSOCIES

M.	COLE Antony	Sciences Nucléaires

CHARGE DE FONCTIONS DE MAITRE DE CONFERENCES

M.	JUNIEN-LAVILLAVROY Paul	O.R.L.

Fait à SAINT MARTIN D'HERES,
DECEMBRE 1975.

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Président : M. NEEL Louis

Vice-Présidents : M. BENOIT Jean

M. BONNETAIN Lucien

PROFESSEURS TITULAIRES

MM.	BENOIT Jean	Radioélectricité
	BESSON Jean	Electrochimie
	BLOCH Daniel	Physique du solide
	BONNETAIN Lucien	Chimie Minérale
	BONNIER Etienne	Electrochimie et Electrométallurgie
	BRISSONNEAU Pierre	Physique du solide
	BUYLE-BODIN Maurice	Electronique
	COUMES André	Radioélectricité
	FELICI Noël	Electrostatique
	LESPINARD Georges	Mécanique
	MOREAU René	Mécanique
	PARIAUD Jean-Charles	Chimie-Physique
	PAUTHENET René	Physique du solide
	PERRET René	Servomécanismes
	POLOUJADOFF Michel	Electrotechnique
	SILBERT Robert	Mécanique des Fluides

PROFESSEURS ASSOCIES

MM.	RUPPERSBERG Albert, Henner	Chimie
	ROUXEL Roland	Automatique

PROFESSEURS SANS CHAIRE

MM.	BLIMAN Samuel	Electronique
	BOUVARD Maurice	Génie Mécanique
	COHEN Joseph	Electrotechnique
	DURAND Francis	Métallurgie
	FOULARD Claude	Automatique
	LACOUME Jean-Louis	Géophysique
	LANCIA Roland	Electronique
	VEILLON Gérard	Informatique Fondamentale & Appliquée
	ZADWORNY François	Electronique

MAITRES DE CONFERENCES

MM.	ANCEAU François	Mathématiques Appliquées
	BOUDOURIS Georges	Radioélectricité
	CHARTIER Germain	Electronique
	GUYOT Pierre	Chimie Minérale
	IVANES Marcel	Electrotechnique
	JOUBERT Jean-Claude	Physique du solide
	MORET Roger	Electrotechnique Nucléaire
	PIERRARD Jean-Marie	Hydraulique
	ROBERT François	Analyse Numérique
	SABONNADIERE Jean-Claude	Informatique Fondamentale & Appliquée
Mme	SAUCIER Gabrièle	Informatique Fondamentale & Appliquée

MAITRE DE CONFERENCES ASSOCIE

M.	LANDAU Ioan	Automatique

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

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

Je tiens à remercier :

Monsieur le Professeur BOLLIET : il m'a fait l'honneur de présider le Jury de cette thèse,

Monsieur DELOBEL, Maître de Conférences : il m'a initié à la recherche et ne m'a ménagé ni son temps ni ses conseils,

Monsieur PECCOUD, Maître de Conférences, et Monsieur PICHAT, Maître de Conférences : ils m'ont apporté les critiques et les conseils indispensables pour réaliser cette thèse,

Monsieur ADIBA : il m'a particulièrement aidé par ses encouragements et ses compétences,

Monsieur DE DONMARTIN : il m'a montré l'application des résultats de recherche dans le domaine industriel et l'intérêt d'une démarche rigoureuse dans la prise de décisions,

Madame REYNAUD-GAROCHE et Madame STIERS : elles ont toujours répondu avec toute leur compétence à mes nombreuses questions,

Monsieur PORTAL : il m'a fait bénéficier de son expérience dans le domaine des bases de données,

Madame DOREL : elle a su rendre cette thèse lisible de par son application et son goût du travail bien fait,

Monsieur IGLESIAS et les membres de son service : ils ont réalisé matériellement cet ouvrage.

Michel LEONARD

Le Laboratoire de Mathématiques Appliquées de Grenoble a offert le cadre intellectuel et matériel de cette thèse ; le C.N.R.S. en a fourni le support financier par l'intermédiaire de l'ATP n° 650.764.73.01.

Plan de thèse

avec mes parents, mon frère, mes amis

INTRODUCTION

Les responsables de la gestion des informations dans les entreprises et les administrations vont être confrontés plus directement grâce à l'introduction des systèmes de gestion de bases de données, aux aspects de structuration des informations contenues dans la base alors qu'ils seront déchargés des préoccupations de gestion et de recherche de données dans l'espace physique. Ainsi pourront-ils prendre en compte une plus grande hétérogénéité des informations et aménager un accès plus facile à des demandes diverses.

Face à une telle évolution, ils vont se trouver confrontés à plusieurs problèmes :

1) la conception de la structure logique de la base de données, qui définit les chemins d'accès logiques aux données, prend une importance accrue aussi bien devant la variété des informations à prendre en compte que devant le nombre des utilisateurs aux préoccupations différentes. Aussi faut-il construire cette structure de manière rigoureuse et précise ;

2) l'évolution de la base est nécessaire afin d'en améliorer sa qualité et de tenir compte des modifications de son environnement ; elle pose de délicats problèmes de réorganisation. Il s'agit de déterminer les opérations à entreprendre face à de tels changements ;

3) la mise en commun d'informations contenues dans des bases différentes peut être réclamée par des utilisateurs ; il s'agit pour satisfaire une telle demande de pouvoir fusionner ces bases.

Ce sont ces trois points que nous allons traiter dans notre travail, sans toutefois les présenter dans cet ordre.

CONCEPTION DE LA STRUCTURE D'UNE BASE DE DONNEES

L'analyse du champ d'utilisation d'une future base de données doit contenir dans ses résultats les informations nécessaires à sa conception. Plusieurs méthodes d'analyse ont été développées jusqu'à maintenant, mais peu d'entre elles permettent de construire la base à l'aide d'un système de gestion de base de données (SGBD).

Pourtant l'utilisation des SGBD semble devoir se répandre dans l'avenir : en effet, les informaticiens sont alors déchargés des préoccupations de gestion et de recherche de données dans l'espace physique et peuvent se consacrer plus pleinement aux problèmes de structuration de la base ; or, ce sont ces problèmes qui vont devenir de plus en plus importants de par la grande diversité des informations à prendre en compte et l'hétérogénéité des demandes à satisfaire. Aussi est-il nécessaire d'adapter les méthodes d'analyse à l'utilisation future des SGBD.

Dans ce but, Wang et Wedekind [26] proposent que les résultats de l'analyse contiennent les relations existant entre les constituants d'une base.

L'intérêt d'une telle démarche est d'aborder tout de suite la partie la plus fondamentale pour construire convenablement une structure de base de données : la sémantique des liaisons entre les différents constituants de la base. En effet, ces liaisons existent, il s'agit de leur donner une réalité en les explicitant rigoureusement.

Ainsi elles serviront de trame à la construction d'une structure de la base. A partir d'elles, il est important de construire un ensemble minimal de liaisons qui recouvrent toutes les autres, comme le font Wang et Wedekind pour celles qui s'expriment sous forme de relations fonctionnelles : un tel ensemble contient toutes les relations que la structure doit prendre en compte. Ce n'est qu'ensuite dans une autre étape qu'il s'agira de considérer d'autres résultats de l'analyse relatifs aux chemins d'accès, aux mécanismes de mises à jour, de destruction de données.

Une telle démarche pose les questions suivantes :

A : Comment trouver, à partir d'un ensemble de relations, un ensemble non redondant les recouvrant toutes ? (Chapitre 3).

B : Comment, à partir d'un ensemble de relations relatif à une application, construire la structure de données à l'aide du langage de définition de données d'un SGBD donné ? (Chapitre 5).

C : Quels sont les paramètres à définir pour créer la structure opérationnelle correspondant à un ensemble de relations et aux résultats d'analyse concernant les chemins d'accès, les règles des opérations de mise à jour et de destruction d'informations ? (Chapitre 5).

EVOLUTION D'UNE BASE DE DONNEES

L'évolution d'une base de données peut être rendue nécessaire pour plusieurs raisons :

1) manque de pertinence des informations fournies qui, comme l'a montré CODD [9], peut résulter de la structure elle-même : celle-ci de par sa définition, contient des sources d'incohérences logiques lors des mises à jour des informations (cf. Chapitre 3) ;

2) manque d'efficacité des traitements qui peut être causé par l'exécution des procédures liées à la cohérence d'informations ou de relations redondantes. DELOBEL et CASEY [13] ont montré le lien étroit qui existe entre ce point et le précédent;

3) modification des règles de fonctionnement de l'organisation pour laquelle la base a été créée ; comme à ces règles correspondent généralement des règles d'intégrité de la base, c'est une restructuration qu'il est nécessaire d'envisager ;

4) évolution des besoins en informations et manque d'efficacité des traitements qui réclament la création de nouveaux chemins d'accès logiques aux données et ainsi une restructuration

5) évolution des logiciels et des configurations de matériel ; "pour permettre aux utilisateurs de bénéficier des avantages et des possibilités de nouveaux logiciels et (ou) de configurations de matériel plus économiques, il devient de plus en plus important de développer des techniques qui permettent la conversion entre le nouveau et l'ancien système" (Fry et Jeris [18]).

Toutes ces raisons obligent à reconsidérer l'organisation d'une base de données. Fry et Jeris [18] séparent les problèmes posés en deux :

- restructuration
et - réorganisation de l'espace physique,

tout en faisant remarquer qu'une réorganisation de base de données dissocie rarement dans la pratique ces deux aspects. Ils définissent la restructuration plus comme un changement du schéma de structure des données que de celui de sa représentation physique.

"Ce changement est susceptible de modifier les relations et la composition de blocs. Certes, la représentation physique change également, mais sa modification est commandée par le changement du schéma" [18].

Comme Fry et Jeris [18] et Navathe et Fry [22], nous nous sommes intéressés seulement à la restructuration et nous avons cherché à répondre aux questions suivantes :

D : Comment peut-on transformer une structure pour éliminer les incohérences d'informations qu'elle cause ? (Chapitre 3).

E : Comment diminuer la redondance de relations et d'informations que contient une base de données ? (Chapitre 3).

La réponse à ces questions permettra d'améliorer la qualité des services rendus par une base de données.

F : Comment peut-on transformer une structure pour tenir compte de l'évolution des règles d'intégrité ? (Chapitre 4).

Il s'agit de déterminer une nouvelle structure cohérente avec les nouvelles règles et les opérations à mettre en oeuvre pour atteindre ce but.

G : Quelles sont toutes les relations qui sont contenues implicitement dans une base de données ? (Chapitre 2).

H : Quand peut-on affirmer que deux structures sont équivalentes ? (Chapitre 4).

Mac Gee [21] a montré comment ce problème est lié à celui de la transposition d'une base de données d'un logiciel vers un autre logiciel quand on cherche à préserver l'information ; il a cherché à donner des éléments intuitifs de solution.

Fry et Jeris [18] pensent que "la définition de la conservation de l'information dans une réorganisation revient à celle de la préservation des relations (logiques) d'ordre structurel". "Généralement ceci implique que toutes les relations de données sont conservées au niveau du schéma de structure".

Trouver une structure équivalente d'une autre passe par la détermination des relations qui lui sont associées et conduit à poser les questions suivantes :

I : Comment obtenir à partir d'une structure de donnés un ensemble de relations ? (Chapitre 2).

J : Réciproquement, commentobtenir une structure de données à partir d'un ensemble de relations ? (Chapitre 5).

K : Quand peut-on affirmer qu'un ensemble de relations et une structure de données sont compatibles ? (Chapitre 3).

FUSION DE PLUSIEURS BASES DE DONNEES

La mise en commun d'informations stockées dans plusieurs bases peut prendre plusieurs solutions techniques :

- regroupement de toutes ces informations dans une même base,

- maintien de ces informations dans leurs bases respectives et création d'un système de communication interbase utilisant un réseau d'ordinateurs,

- restructuration de chacune de ces bases afin qu'elles puissent communiquer entre elles.

Un tel projet est très vaste ; il nous semble important du point de vue des structures logiques de données de s'intéresser en premier lieu à définir la structure de la base résultant de la fusion : déjà ce cadre restreint est suffisamment vaste.

L : Comment peut-on, à partir des structures de départ, construire la structure de la base résultant de la fusion des bases, que cette base soit concentrée ou non sur un seul site ? (Chapitre 2).

M : Comment peut-on déterminer, à partir des règles d'intégrité des bases initiales, celles de la base finale ? Existe-t-il un ensemble non redondant de ces règles ? (Chapitre 3).

Pour répondre à de telles questions, nous avons choisi de transformer chaque structure en un ensemble de relations et ensuite de poser le problème de la fusion des bases, en termes de fusion d'ensembles de relations. Une telle démarche pose de nouveau le problème de la transformation d'une structure en un ensemble de relations (N, Ch. 2), de la détermination d'un ensemble de relations non redondant (O, Ch. 3), du passage d'un ensemble de relations à une structure de données (P, Ch. 5).

Nous venons de dresser un inventaire des problèmes posés par l'utilisation des SGBD. Ces problèmes sont essentiellement d'ordre méthodologique et ne relèvent pas toujours d'une approche formelle. Néanmoins l'objectif de notre travail est d'essayer d'y apporter des éléments de réponse et la thèse* que nous vous présentons est que l'étude de ces problèmes peut se faire en utilisant une approche relationnelle des modèles de bases de données. Nous montrerons comment les concepts développés à propos des modèles relationnels peuvent être utiles dans les problèmes évoqués ci-dessus.

Le plan de notre travail ne suivra pas l'ordre des questions posées. En effet certaines questions peuvent être regroupées et conduit au plan suivant :

Introduction

Chapitre 1 : Modèle relationnel de représentation des données : nous rappelons brièvement les principaux concepts du modèle relationnel qui nous seront utiles.

Chapitre 2 : Transformation d'une structure de données sous forme réseau en un ensemble de relations (questions G, I, L, N) : nous montrons comment il est possible de passer d'une structure arborescente ou sous forme de réseau à un ensemble de relations ; nous considérons en particulier les structures qu'il est possible de définir en CODASYL ou en SOCRATE.

Chapitre 3 : Elimination de redondance et d'incohérences d'informations (questions A, D, E, K, M, O) : nous rappelons comment la définition d'une structure peut conduire elle-même à des incohérences d'informations et nous donnons ensuite un moyen de les éliminer, en montrant qu'il suffit d'éliminer la redondance de relations sous certaines conditions.

---

*Nous prenons ici thèse dans son sens original, c'est-à-dire qui défend une idée.

Chapitre 4 : Réorganisation d'une structure ; structures équivalentes (questions F, H) ; nous montrons comment l'étude des propriétés sémantiques des informations permet d'aborder la restructuration d'une base de données et de définir la notion de structures équivalentes.

Chapitre 5 : Transformation d'un ensemble de relations dans une structure de données sous forme réseau (questions B, C, J, P) ; nous mettons en évidence différentes étapes dans une telle transformation ; à chacune d'elles correspondent des choix que les résultats de l'analyse du champ d'application doivent permettre d'éclairer.

Chapitre 6 : Réalisations :
elles concernent les chapitres 2 et 3.

Conclusions

Annexe A : Algorithme de fermeture et de couverture minimale d'une forme relationnelle ;
cet algorithme permet l'élimination de redondance de relations (Chapitre 3) ; nous donnons les démonstrations qui nous ont permis de le construire.

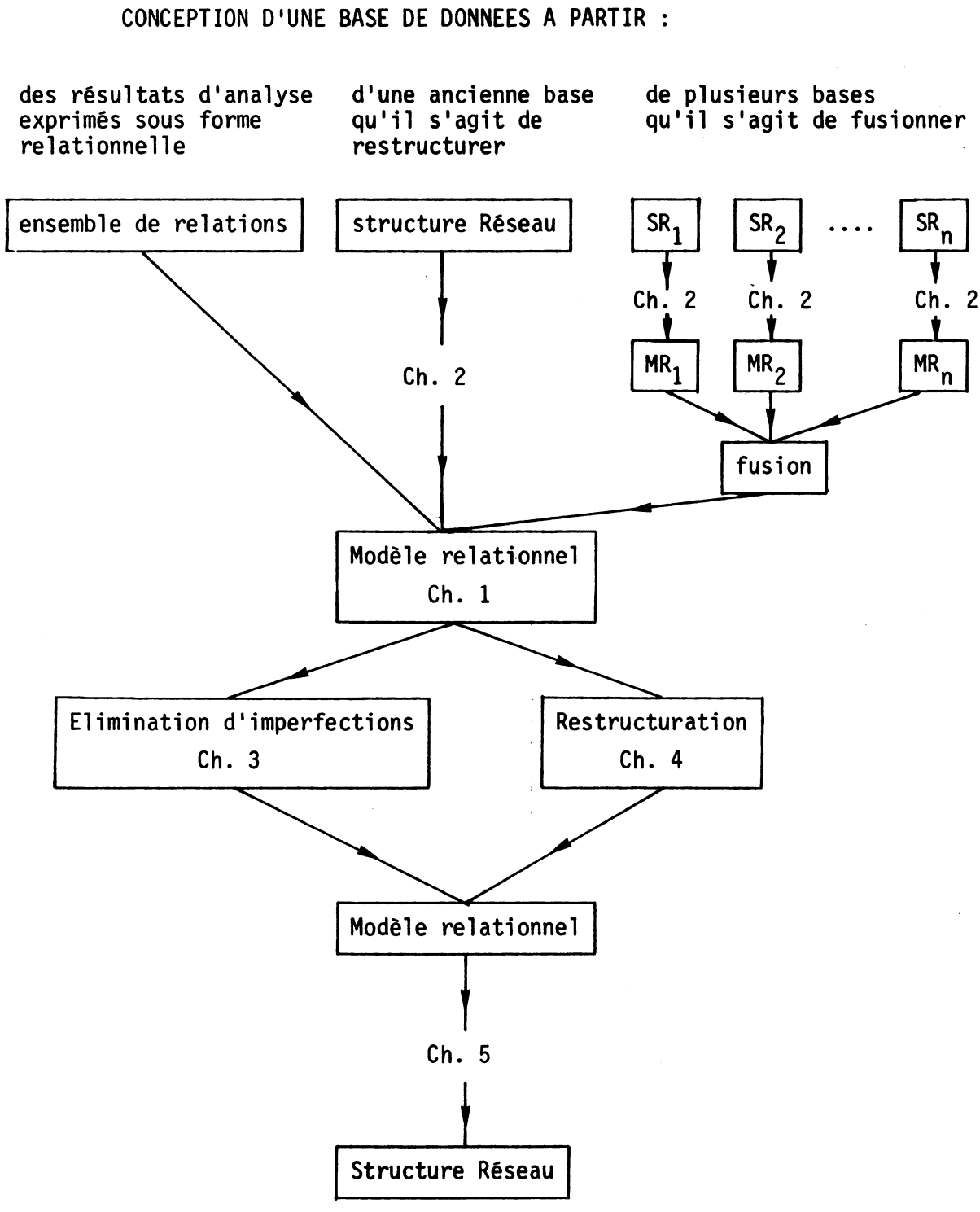
Annexe B : Propriétés des décompositions arborescentes et de leurs combinaisons avec des relations fonctionnelles ; nous donnons les différentes démonstrations qui nous ont permis de donner des types de restructuration (Chapitre 4).

Tableau de correspondance des questions posées et des chapitres

	CONCEPTION DE LA STRUCTURE D'UNE BASE DE DONNEES	
A	Comment trouver, à partir d'un ensemble de relations, un ensemble non redondant les recouvrant toutes ?	Chap. 3
B	Comment, à partir d'un ensemble de relations relatif à une application, construire la structure de données à l'aide du langage de définition de données d'un SGBD donné ?	Chap. 5
C	Quels sont les paramètres à définir pour créer la structure opérationnelle correspondant à un ensemble de relations et aux résultats d'analyse concernant les chemins d'accès, les règles des opérations de mise à jour et de destruction d'informations ?	Chap. 5
	EVOLUTION D'UNE BASE DE DONNEES	
D	Comment peut-on transformer une structure pour tenir compte de l'évolution des règles d'intégrité ?	Chap. 3
E	Comment déminuer la redondance de relations et d'informations que contient une base de données ?	Chap. 3
F	Comment peut-on transformer une structure pour tenir compte de l'évolution des règles d'intégrité ?	Chap. 4
G	Quelles sont toutes les relations qui sont contenues implicitement dans une base de données ?	Chap. 2
H	Quand peut-on affirmer que deux structures sont équivalentes ?	Chap. 4
I	Comment obtenir, à partir d'une structure de données, un ensemble de relations ?	Chap. 5
J	Réciproquement, comment obtenir une structure de données à partir d'un ensemble de relations ?	Chap. 5
K	Quand peut-on affirmer qu'un ensemble de relations et une structure de données sont compatibles ?	Chap. 3
	FUSION DE PLUSIEURS BASES DE DONNEES	
L	Comment peut-on, à partir des structures de départ, construire la structure de la base résultant de la fusion des bases, que cette base soit concentrée ou non sur un seul site ?	Chap. 2

M	Comment peut-on déterminer, à partir des règles d'intégrité des bases initiales, celles de la base finale ? Existe-t-il un ensemble non redondant de ces règles ?	Chap. 3
N	Comment transformer une structure en un ensemble de relations ?	Chap. 2
O	Comment déterminer un ensemble de relations non redondant ?	Chap. 3
P	Comment transformer un ensemble de relations dans une structure de SGBD ?	Chap. 5

Plan de notre démarche



C H A P I T R E 1

MODELE RELATIONNEL DE REPRESENTATION D'UNE BASE DE DONNEES

RAPPELS

S O M M A I R E

Dans ce chapitre, nous introduisons les principaux concepts du modèle relationnel qui nous seront utiles dans notre démarche ; nous reprenons les définitions introduites dans [14] et la présentation de [17].

1.1. RELATIONS

Champs :

Nous nommerons champ un ensemble U non vide. Pour exprimer que l'élément a appartient à U, nous écrirons :

$$a \in U$$

Constituants :

Nous considérerons une fois pour toutes une classe dont les éléments seront des *constituants*. Nous les désignerons par des lettres majuscules : X, Y, Z.

Affectation d'une donnée - Champ d'un constituant :

Un constituant X peut dynamiquement prendre diverses valeurs qui sont des êtres mathématiques précis. Si a est donnée, nous écrirons :

$$X := a$$

pour dire que le constituant X prend momentanément la valeur a. On dit alors que a est affecté à X.

Attribuer un champ U à un constituant X, c'est décider de choisir les affectations de X parmi les valeurs de U. Une telle attribution se notera :

$$X :\in U$$

P-uple de constituants :

Un ensemble de constituants $\{X_1, X_2, \ldots, X_p\}$ se nommera p-uple de constituants et pourra être désigné par une seule lettre telle que X. Dans la suite, lorsque nécessaire, on précisera si X désigne un seul constituant ou un p-uple de constituants ; dans ce dernier cas, on dira que X est un *constituant composé*.

Si $a_1,a_2,\ldots,a_p$ désignent p valeurs (non forcément distinctes), les p affectations :

$$X_i := a_i \qquad i = 1,2,\ldots,p$$

seront notées symboliquement :

$$\langle X:a\rangle = \{\langle X_1:a_1\rangle, \langle X_2:a_2\rangle, \ldots, \langle X_p:a_p\rangle\}$$

Il pourra être commode dans certains cas de découper un p-uple constituants en parties disjointes, dont certaines peuvent être vides : soient par exemple Y,Z,T ces parties, on écrira alors :

$$X = (Y,Z,T)$$

Si, par contre, ces parties ne sont pas disjointes, on écrira :

$$X = (Y \vee Z \vee T)$$

Dans d'autres cas, il sera commode de considérer le constituant composé X comme l'intersection des constituants composés Y et Z ; on écrira :

$$X = (Y \wedge Z)$$

Relations n-aires, entités :

On définit une relation n-aire par trois éléments :

- la donnée d'un n-uplet de constituants $X = \{X_1,X_2,\ldots,X_n\}$,
- l'attribution d'un champ U_i à chaque constituant X_i,
- un prédicat, c'est-à-dire une règle précise qui, à toute affectation $\langle X:a\rangle$ conforme aux champs, donne une réponse exclusivement *vraie* ou *fausse*.

Nous noterons une relation par :

$$R\,[X_1,X_2,\ldots,X_n]\ ;\ X_i :\in U_i \qquad i = 1,2,\ldots,n$$

où de façon plus condensée :

$$R(X)\ ;\ X :\in U$$

On appellera *entité* une affectation <X:a> pour laquelle l'évaluation du prédicat est vraie, soit :

$$\| R[<X:a>] \| = \underline{vrai}$$

ou bien encore :

$$\| R[<X_1:a_1>, <X_2:a_2>, \ldots, <X_n:a_n>] \| = vrai$$

Dans une telle définition, on remarquera qu'une entité est un ensemble d'affectations. Lorsqu'on ne veut pas préciser l'affectation, on notera :

$\| R[X_1,X_2,\ldots,X_n] \|$ ou plus simplement $\| R[X] \|$ l'évaluation potentielle du prédicat.

Pour simplifier l'écriture et en se fixant un *ordre de référence* pour énumérer les constituants, on écrira :

$$\| R[a_1,a_2,\ldots,a_n] \| = \underline{vrai}$$

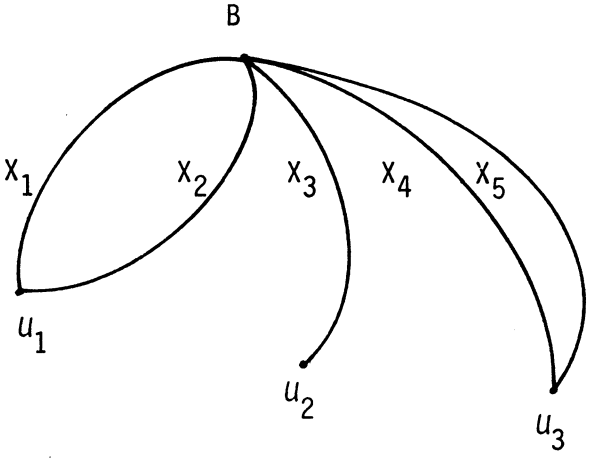
Dans une telle écriture, une entité a sera une suite d'affectations, on écrira :

$$a = (a_1,a_2,\ldots,a_n)$$

et ceci par abus de langage.

Représentation d'une relation n-aire :

Il sera commode d'introduire le schéma suivant pour représenter une relation n-aire



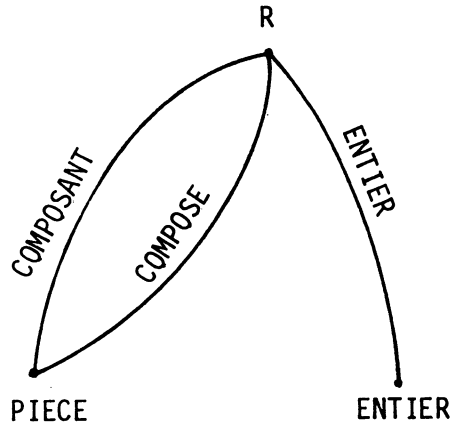
Un exemple de relation ternaire nous est donné par la composition d'un produit.

Etant donné deux champs PIECE et ENTIER et trois constituants COMPOSANT, COMPOSE et QUANTITE

COMPOSANT :ϵ PIECE
COMPOSE :ϵ PIECE
QUANTITE :ϵ ENTIER

et le prédicat

$\| R[a_1,a_2,a_3] \|$ = <u>vrai</u> si la pièce a_1 comprend pour sa fabrication a_3 fois la pièce a_2.



Une réalisation de R sera un ensemble d'entités que l'on peut représenter sous forme d'un tableau de valeurs :

COMPOSANT	COMPOSE	QUANTITE
vélo	roue	2
vélo	cadre	1
vélo	guidon	1
roue	rayon	56
roue	pneu	1

Dans un tel modèle, on peut interpréter la notion de constituant comme une fonction d'accès qui, à une entité de la relation R, fait correspondre une valeur d'un champ, le constituant étant une fonction d'accès unidirectionnelle d'un élément de R vers le champ.

Notation :

Dans le cas où aucune ambiguïté n'est à craindre, la notation [X,Y,Z] peut suffire pour désigner une relation ; nous l'utiliserons souvent dans nos développements futurs.

1.2. PRINCIPALES OPERATIONS DEFINIES SUR LES RELATIONS

Projection :

Soit R[X,Y] une relation où l'on a distingué le constituant X de champ U du constituant Y. La projection suivant X de cette relation est la relation au constituant Y notée [Y] R[X,Y] définie par son prédicat :

$$\| [Y]\ R[X,Y] \| \triangleq \exists\, a \ \| R[a,y] \| , \quad a \in U$$

Soit R[X,Y,Z] une relation où l'on a distingué le constituant X de champ U, le constituant Y de champ V et le constituant Z ; si <Y:b> est une affectation, on définira la projection suivant X à partir de b comme la relation au constituant Z notée :

$$[Z, \langle Y:b \rangle]\ R[X,Y,Z]$$

et définie par son prédicat :

$a \in U,\ b \in V, \| [Z, \langle Y:b \rangle]\ R[X,Y,Z] \| \triangleq \exists a \ \| R[a,b,Z] \|$

Propriété de la projection :

La projection d'une relation ne dépend pas du chemin utilisé :

$$\| [Z]\ R[X,Y,Z] \| = \| [Z]\ [X,Z]\ R[X,Y,Z] \| = \| [Z]\ [Y,Z]\ R[X,Y,Z] \|$$

Sous-relation :

Toute relation obtenue par projection de la relation R est appelée sous-relation de R.

Somme et Produit de deux relations :

Soient R[X] et S[Y] deux relations (les constituants communs éventuels étant munis de mêmes champs). On pose $Z = X \vee Y$.

Nous définirons la somme et le produit de ces deux relations par leur prédicat associé :

$$\|(R+S)\ [Z]\| \triangleq (\| R[X]\| \text{ ou } \| S[Y]\|)$$

$$\|(R*S)\ [Z]\| \triangleq (\| R[X]\| \text{ et } \| S[Y]\|)$$

Exemple 1.1

R(A,B) et S(A,C) sont définies par un tableau de valeurs :

R[A,B]	
a_1	b_1
a_1	b_2
a_1	b_3
a_2	b_1
a_2	b_4

S[A,C]	
a_1	c_1
a_1	c_2
a_2	c_1
a_3	c_3

(R*S) [A,B,C]		
a_1	b_1	c_1
a_1	b_1	c_2
a_1	b_2	c_1
a_1	b_2	c_2
a_1	b_3	c_1
a_1	b_3	c_2
a_2	b_1	c_1
a_2	b_4	c_1

Quand $X \wedge Y \neq \emptyset$, le produit est également appelé *composition normale*.

Les sommes et les produits ont les propriétés habituelles de l'algèbre de Boole :

- idempotence
- associativité
- commutativité
- double distributivité.

1.3. RELATION FONCTIONNELLE

1.3.1. Relation fonctionnelle et Index

Soit la relation R(X,Y,Z) (Z ensemble de constituants éventuellement vide).

On dit que cette relation admet une *sous-relation fonctionnelle* de X vers Y notée

$$X \xrightarrow{R} Y$$

lorsque :

a,b,c étant des affectations à X,Y,Z

$\forall a\ \forall b\ \forall b'\ \forall c\ \forall c'\ \|R[a,b,c]\|$ et $\|R[a,b',c']\| \Rightarrow (b=b')$

En d'autres termes, la connaissance de X détermine au plus un seul Y sans avoir besoin de préciser Z.

$X \xrightarrow{R} X$ est de manière évidence une relation fonctionnelle : elle est dite *triviale.*

Une relation fonctionnelle $X \xrightarrow{R} Y$ sera *élémentaire* si pour toute partie $X' \subset X$, $X' \xrightarrow{R} Y$ n'est pas une relation fonctionnelle et si elle n'est pas triviale.

Lorsqu'il n'y a pas d'ambiguïté sur la relation R, on notera plus simplement $X \rightarrow Y$ au lieu de $X \xrightarrow{R} Y$.

X sera appelé *partie gauche* de la relation fonctionnelle et Y *partie droite.*

Soit la relation $R[A_1, A_2, \ldots, A_n]$ formée sur l'ensemble des constituants $A = \{A_1, A_2 \ldots A_n\}$. Nous définissons *l'index* de cette relation, comme le constituant éventuellement composé I de A tel que :

$\forall A_j \in A$, il existe la sous-relation fonctionnelle $I \rightarrow A_j$

et $\nexists I' \subset I$, tel que $\forall A_j \in A$, il existe la sous-relation fonctionnelle $I' \rightarrow A_j$.

Exemple 1.2

R [PROFESSEUR, HEURE, SALLE]

Cette relation admet une sous-relation fonctionnelle car un professeur donné à une heure précise ne peut être que dans une *seule* salle :

PROFESSEUR, HEURE → SALLE.

1.3.2. Propriétés des relations fonctionnelles ([14] p.77)

Dans tout ce paragraphe, nous supposons une relation R formée sur les constituants E, F, G, H ;

E → F désigne simplement la sous-relation fonctionnelle de R :

[E,F] R.

Les relations fonctionnelles ont les propriétés suivantes :

- transitivité

  ∀E,F,G, si E → F et F → G, alors E → G

- réflexivité

  ∀E E → E

- projection

  si E → F,G alors E → F et E → G

- augmentation

  si E → F, alors ∀G E,G → F

- additivité

  si E → F et E → G, alors E → F,G

- pseudo-transitivité

  si E → F

  F,G → H, alors E,G → H

1.3.3. Forme relationnelle

Il est montré dans [14] qu'à cause de ces propriétés, il existe un *isomorphisme* entre l'*ensemble des relations fonctionnelles canoniques** d'une relation et une fonction booléenne de monômes à une seule variable primée appelée *forme relationnelle*, l'opération de pseudo-transitivité de relations fonctionnelles correspondant à celle de consensus des monômes ([20]) ; les relations fonctionnelles élémentaires correspondent aux monômes premiers de la forme booléenne et réciproquement.

Nous présentons, à l'aide de l'exemple suivant, la correspondance existant entre un ensemble de relations fonctionnelles et un graphe orienté de noeuds et d'étoiles**: les constituants sont aux noeuds, les relations fonctionnelles aux étoiles.

Un *ensemble de relations fonctionnelles* est dit *sans circuit* s'il n'existe aucun circuit de noeuds et d'étoiles (ou plus simplement de sommets) dans le graphe précédent.

1.3.4. Exemple 1.3

Soit R_{11} [A,B,C,D,E] et l'ensemble E des relations fonctionnelles :

$$E = \begin{cases} A \rightarrow B & (1) \\ B \rightarrow C & (2) \\ A,B \rightarrow C & (3) \\ C,D \rightarrow E & (4) \end{cases}$$

A l'ensemble E, on peut associer la forme relationnelle f

$$f = ab' + bc' + abc' + cde'$$

Le graphe de noeuds et d'étoiles correspondant est alors :

---

* *Une relation fonctionnelle est canonique ([6]) si sa partie droite est formée d'un simple constituant élémentaire. Ceci est toujours possible grâce à la propriété de projection.*

** *KUNTZMANN Théorie des réseaux graphes - DUNOD*

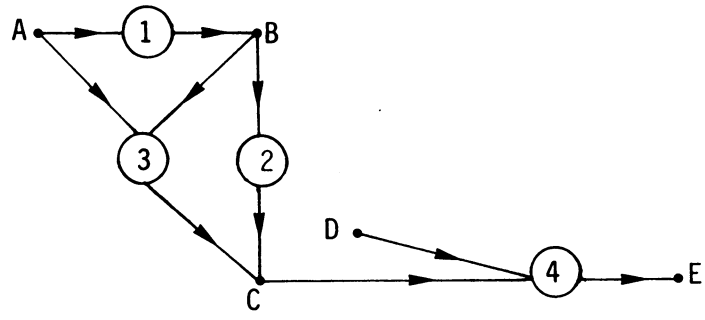


Fig. 1.1

1.3.5. Opération de pseudo-transitivité +>

Soient deux relations fonctionnelles : f_1 et f_2 qui peuvent se combiner par transitivité ou pseudo-transitivité pour former une autre relation fonctionnelle f. Nous noterons alors :

$$f = f_1 +> f_2$$

où +> désigne cette opération de composition interne sur l'ensemble des relations fonctionnelles.

Dans l'annexe A, nous donnons la condition que doivent remplir f_1 et f_2 pour donner f et nous montrons que cette opération +> est non commutative et non associative.

Ainsi, si $f_1 = (E \rightarrow F)$ et $f_2 = (F,G \rightarrow H)$, on peut obtenir par pseudo-transitivité $f = (E,G \rightarrow H)$; on notera

$$f = f_1 +> f_2$$

ou $$(E,G \rightarrow H) = (E \rightarrow F) +> (F,G \rightarrow H).$$

1.4. DECOMPOSITION D'UNE RELATION

Le problème de la décomposition est ainsi posé dans ([14] p. 3.14) : étant donné une relation R et les opérateurs de projection et de décomposition normale, est-il possible de trouver des relations R_1 et R_2 telles que R peut s'exprimer à partir de R_1 et de R_2 en utilisant les opérateurs de composition normale et de projection ?

Dans l'affirmative, R_1 et R_2 forment une *décomposition* de R.

Nous montrerons combien ce problème est fondamental pour la conception de structures de bases de données. Nous allons maintenant définir les *décompositions arborescentes* qui sont des décompositions particulières mais importantes ; elles sont de deux types :

- décomposition arborescente sans racine secondaire,
- décomposition arborescente avec racines secondaires.

Dans ce paragraphe, nous considérerons une relation R définie sur les constituants $A = \{A_1, A_2, \ldots, A_n\}$ et nous désignerons par X,Y,Z,U des parties de A.

1.4.1. Définition d'une décomposition arborescente sans racine secondaire

Etant donné une partie (X,Y,Z,U) où X,Y,Z et U sont disjoints deux à deux, nous dirons que cette partie constitue une décomposition arborescente par rapport à la partie X si :

$$[X,Y]R * [X,Z]R * [X,U]R = [X,Y,Z,U]R$$

et on notera

$$X : Y \mid Z \mid U$$

Nous dirons que X constitue la racine de la décomposition arborescente et Y,Z,U des feuilles.

Remarque : Une décomposition arborescente sans racine secondaire sera souvent appelée décomposition arborescente.

Nous rappelons deux résultats fondamentaux démontrés dans [14].

Proposition 1 : Etant donné une partie (X,Y,Z) où X,Y,Z sont disjoints deux à deux ; elle constitue une décomposition arborescente :

$$[X,Y,Z]R = [X,Y]R * [X,Z]R$$

si et seulement si

$$\forall \langle X:x\rangle \in [X]R,\ \forall \langle Y:y\rangle \in [\langle X:x\rangle,Y]R,$$
$$[\langle X:x\rangle, \langle Y:y\rangle,Z]R = [\langle X:x\rangle,Z]R$$

Proposition 2 : Soit une partie (X,Y,Z) où X,Y,Z sont disjoints deux à deux et telle qu'il existe la relation fonctionnelle $X \rightarrow Y$,
alors [X,Y,Z]R peut se décomposer en :

$$[X,Y,Z]R = [X,Y]R * [X,Z]R.$$

Une telle décomposition est appelée *fonctionnelle*.

Exemple 1.4 de décomposition arborescente

Soit la relation

R [ETUDIANT, PROFESSEUR, EXAMEN]

X : EXAMEN
Y : ETUDIANT
Z : PROFESSEUR.

Un étudiant peut passer plusieurs examens et a plusieurs professeurs ; tous les professeurs concernés par un examen s'intéressent à tous les étudiants qui le passent, et donc :

$\forall \langle X:x\rangle \in [X]R,\ \forall \langle Y:y\rangle \in [\langle X:x\rangle,Y]R,\ [\langle X:x\rangle,\langle Y:y\rangle,Z] = [\langle X:x\rangle,Z]$
$\Longrightarrow [X,Y,Z]R = [X,Y]R * [X,Z]R.$

Exemple 1.5 de décomposition fonctionnelle

FILM [NOM, METTEUR EN SCENE, ACTEUR]

R : FILM
X : NOM
Y : METTEUR EN SCENE
Z : ACTEUR.

A chaque NOM ne correspond qu'un seul METTEUR EN SCENE ; il existe donc une relation fonctionnelle X → Y et R peut se décomposer en :

$$[X,Y,Z]R = [X,Y]R * [X,Z]R.$$

Ces deux exemples font ressortir le fait que la décomposition d'une relation est étroitement liée à la connaissance des propriétés intrinsèques reliant les informations entre elles. De plus, ils montrent l'intérêt pratique des décompositions qui permet de découper des relations en sous-relations sans perte d'informations puisqu'à partir d'elles, on peut toujours reformer la relation initiale.

A chaque décomposition d'une relation, correspond donc un choix d'organisation des données.

1.4.2. Définition d'une décomposition arborescente avec racines secondaires, également appelée décomposition arborescente généralisée

Dans la notion de décomposition arborescente sans racine secondaire notée symboliquement X : Y|Z|U, et équivalente à la décomposition

$$[X,Y,Z,U]R = [X,Y]R * [X,Z]R * [X,U]R$$

l'opération de composition exprimée par l'opérateur "*" s'effectue sur le même constituant charnière X qui est la racine de la décomposition arborescente.

Par contre, l'expression suivante :

$$[X,Y,Z,U,V]R = [X,Y,Z]R * [X,Y,U]R * [X,V]R$$

ne peut se traduire par une décomposition arborescente sans racine secondaire car les feuilles ne seraient pas disjointes.

Pour pallier à cet inconvénient, nous allons introduire la notion de *décomposition arborescente avec racines secondaires* ou *décomposition arborescente généralisée.*

Définition : Etant donné une arborescence construite sur les parties de A que nous noterons $(P(A),H)$ où $P(A)$ désigne l'ensemble des parties de A et H une relation d'ordre. Cette arborescence peut être représentée par la figure 1.2, où X est la racine et $X_1,X_2,\ldots,X_p$ le sommet de sous-arborescences dont les éléments sont $B_1,B_2,\ldots,B_p$, éléments de P(A).

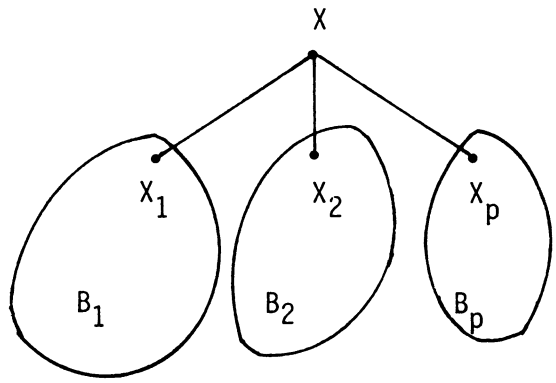


Fig. 1.2

De la même façon, les sous-arborescences de sommet X_i, i=1,2,...,p peuvent être vues de la même façon, les X_i seront appelées des racines secondaires.

A cette arborescence, nous associerons de façon unique un ensemble de décompositions arborescentes sans racine secondaire en procédant de la façon suivante :

- à la racine X on associera la décomposition

$$X : B_1|B_2 \ldots|B_p$$

- à la sous-arborescence de sommet X_1 la décomposition

$$XX_1 : B_{11}|B_{12} \ldots|B_{1p_1}$$

si X_1 n'est pas une feuille de l'arborescence

- on répètera ce processus de proche en proche.

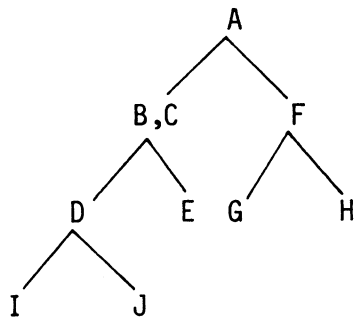
Inversement à un ensemble quelconque de décompositions arborescentes sans racine secondaire peut ne correspondre aucune arborescence. Toutefois, si à un ensemble de décompositions arborescentes sans racine secondaire on peut faire correspondre une arborescence, alors cette arborescence représentera une décomposition arborescente généralisée d'une relation construite sur un ensemble A de constituants.

Exemple 1.6 de décomposition arborescente généralisée

Considérons une relation construite sur les constituants A = {A,B,C,D,E,F,G,H,I,J} telle que cette relation possède au moins comme invariant les décompositions arborescentes :

A : B,C,D,E,I,J|F,G,H
A,B,C : D,I,J|E
A,B,C,D : I|J
A,F : G|H

A cet ensemble, on peut faire correspondre l'arborescence



qui correspond à une décomposition arborescente généralisée.

Exemple 1.7 - autre exemple de décomposition arborescente généralisée

Considérons la relation R construite sur les constituants MAITRISE, PROFESSEUR-RESPONSABLE, CERTIFICAT, ETUDIANTS, PROFESSEURS ; une maîtrise est une unité d'enseignement dont est responsable un professeur et qui comporte 4 certificats. Chaque certificat de chaque maîtrise est assuré par des cours de professeurs et suivi par des étudiants ; tout étudiant d'un certificat suit tous les cours de chaque professeur du certificat.

Cette relation contient la relation fonctionnelle :

MAITRISE → PROFESSEUR-RESPONSABLE

qui conduit à la décomposition de R suivante d'après la proposition 2 du § 1.4.1.

R = [MAITRISE, PROFESSEUR-RESPONSABLE]R*[MAITRISE, CERTIFICAT, ETUDIANTS, PROFESSEURS]R.

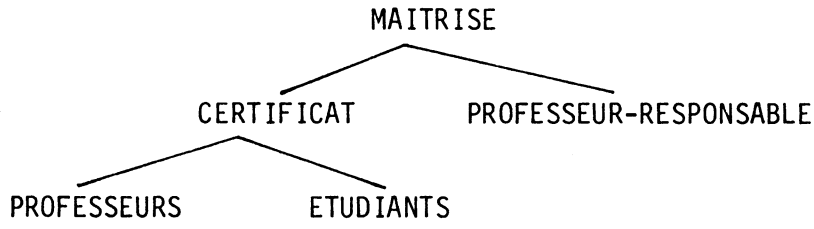
La sous-relation [MAITRISE, CERTIFICAT, ETUDIANTS, PROFESSEURS]R peut se décomposer en :

[MAITRISE, CERTIFICAT, ETUDIANTS]R * [MAITRISE, CERTIFICAT, PROFESSEUR]R.

Ces deux propriétés permettent donc de décomposer R suivant une décomposition arborescente généralisée :

MAITRISE : PROFESSEUR-RESPONSABLE|CERTIFICAT, ETUDIANTS, PROFESSEURS
MAITRISE, CERTIFICATS : ETUDIANTS|PROFESSEURS.

Cette décomposition correspond à l'arborescence suivante :



C H A P I T R E 2

TRANSFORMATION D'UNE STRUCTURE RESEAU DANS UN ENSEMBLE DE RELATIONS

S O M M A I R E

Nous avons indiqué, dans l'introduction, que toute notre démarche se sert du modèle relationnel pour l'étude de la conception d'une base, de sa restructuration et de la fusion de plusieurs bases. Aussi, avant de présenter de telles études, est-il nécessaire, dans une première approche, de montrer comment, d'une structure réseau d'une base de données, on peut dégager les relations qu'elle contient implicitement.

Tel est le but de ce chapitre qui cherche ainsi à répondre aux questions suivantes posées dans l'introduction au sujet de :

- la restructuration d'une base de données : quelles sont toutes les relations qui sont contenues implicitement dans une structure de bases de données ? (question G)
- la recherche d'une structure équivalente à une structure donnée (question I)
- la fusion de plusieurs bases de données :
  comment peut-on, à partir des structures de départ, construire la structure de la base résultant de la fusion des bases, que cette base soit ou non sur un seul site ? (question L)
  Comment obtenir d'une structure un ensemble de relations ? (question N qui découle de la précédente).

Pour répondre à de telles questions, nous allons présenter en premier lieu les principaux aspects d'une structure arborescente, celle-ci étant un cas particulier d'une structure réseau, l'étude de sa transformation en un ensemble de relations est incluse dans celle d'une structure réseau.

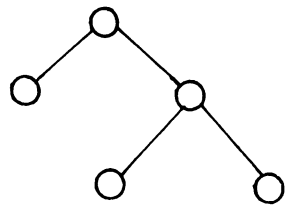
Nous définirons ensuite une structure RESEAU qui pourra aussi bien décrire une structure CODASYL qu'une structure SOCRATE. Ce type de structure nous permettra dans la dernière partie d'obtenir l'ensemble de relations qui correspond à la structure réseau initiale.

En dernier lieu, nous illustrerons une telle transformation à l'aide d'un exemple.

2.1. STRUCTURE ARBORESCENTE ET RELATIONS

Une structure arborescente est formée de groupes de constituants reliés entre eux par des liaisons orientées de telle sorte que le graphe, dont les noeuds sont des groupes et les arêtes les liaisons, est un arbre.

Un groupe A est supérieur à un autre B s'il se trouve dans le graphe sur le chemin conduisant de la racine à B ; B est dit inférieur à A.



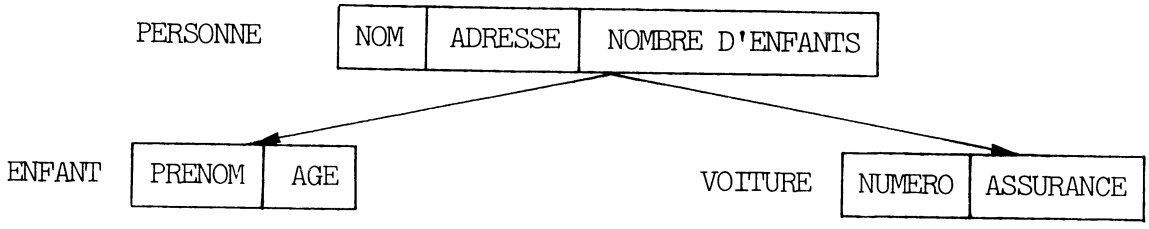
Propriétés fondamentales d'une structure arborescente ([12]) :

- une réalisation d'un groupe inférieur ne peut être créée que si les réalisations des groupes supérieurs correspondants existent
- la destruction d'une réalisation d'un groupe détruit celles des groupes inférieurs, qui lui sont associées.

Nous montrons dans le paragraphe suivant qu'une structure arborescente est un cas particulier d'une structure réseau. Nous ne décrivons pas ici tout le processus de transformation d'une structure arborescente en un ensemble de relations, mais nous présentons les éléments essentiels :

- chaque groupe de constituants crée une relation n-aire
- son index contient tous les index des groupes supérieurs ; ceci est la conséquence des deux propriétés énoncées précédemment.

Exemple 2.1



Les relations contenues implicitement dans cette structure sont :

PERSONNE [NOM, ADRESSE, NOMBRE D'ENFANTS]
ENFANT [NOM, PRENOM, AGE]
VOITURE [NOM, NUMERO, ASSURANCE]

où les constituants soulignés d'une relation représentent son index.

2.2. ASPECTS RELATIONNELS D'UNE STRUCTURE RESEAU : CAS PARTICULIER DES STRUCTURES SOCRATE ET CODASYL

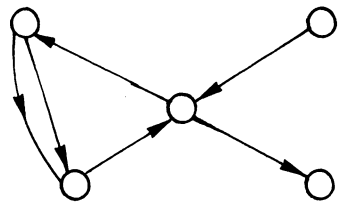
2.2.1. Caractéristiques d'une structure réseau

Une structure réseau est composée :

- de groupes de constituants :
  à chaque groupe A est associé un index k(A) ; celui-ci est tel qu'à une de ses réalisations correspond une et une seule réalisation du groupe ;
- de liaisons directionnelles entre deux groupes de constituants,

et a les propriétés suivantes :

- chaque groupe peut atteindre plusieurs groupes et être atteint par plusieurs groupes ;
- il peut exister plusieurs liaisons entre deux mêmes groupes.



Une structure arborescente est un cas particulier d'une structure réseau : elle admet en effet la propriété plus restrictive qu'un groupe ne peut atteindre qu'un autre groupe et ceci par une seule liaison.

Nous allons dégager les *aspects relationnels* d'une structure réseau en considérant les structures qu'il est possible de construire à partir de systèmes comme SOCRATE [1] ou CODASYL [8] ; la transposition à d'autres systèmes serait analogue.

Etant donné une structure SOCRATE (puis CODASYL), nous allons d'abord considérer les liaisons entre groupes et leur associer des relations correspondantes ; ensuite, nous associerons aux groupes des relations n-aires.

2.2.2. Aspects relationnels d'une structure SOCRATE

Une structure SOCRATE est formée d'ENTITES reliées entre elles par différents types de liaisons.

Dans ce qui suit, une réalisation d'une ENTITE est notée : entité.

2.2.2.a Les différents types de liaisons entre ENTITES sont :

- Référence :

Si une ENTITE A contient une référence vers une ENTITE B, ceci signifie qu'à une entité de A ne correspond qu'une entité de B ; il existe donc la relation fonctionnelle entre index :

$$k(A) \rightarrow k(B)$$

Exemple 2.2*

ENTITE LIVRE (A)
- CODLIV
- TITRE
- AUTEUR référence un ECRIVAIN
- DATE

ENTITE ECRIVAIN (B)
- NOM
- AGE

Si CODLIV et NOM sont respectivement les index de LIVRE et ECRIVAIN, alors

$$CODLIV \rightarrow NOM$$

---

* *Dans tous les exemples SOCRATE et CODASYL, nous n'indiquons dans les structures que les éléments nécessaires à la compréhension.*

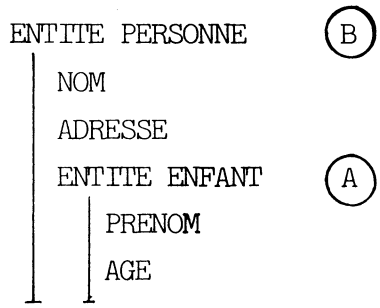
- Entité englobée :

Si une ENTITE A est englobée dans une ENTITE B, ceci signifie qu'il existe une arborescence (§ 2.1) entre les deux ENTITES dont B est l'ENTITE supérieure, puisqu'elle vérifie les deux propriétés :

- toute entité A ne peut exister que si une entité B correspondante existe ;
- la destruction d'une entité B entraîne la perte de toutes les entités A correspondantes.

Ainsi, si l'on considère la relation réellement implantée associée à l'ENTITE A, il existe un de ses index k(A) qui contient un index k(B) de B.

Exemple 2.3

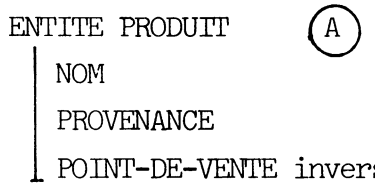


L'ENTITE ENFANT est englobée dans l'ENTITE PERSONNE ; l'index de ENFANT contient donc l'index de PERSONNE ; ainsi si NOM est index de PERSONNE, NOM est contenu dans celui de ENFANT, à savoir : (NOM, PRENOM).

- Inverse (ou anneau*) :

Si une ENTITE A contient un INVERSE vers une ENTITE B, ceci signifie qu'à une entité A peut correspondre une ou plusieurs entités B ; à une liaison *inverse*, correspond une relation binaire formée sur [k(A), k(B)].

Exemple 2.4



anneau existe dans une version de SOCRATE : il désigne la liaison inverse

ENTITE MAGASIN (B)
- NUMERO INSEE
- ADRESSE

- Entité-référence :

Si une ENTITE A contient une ENTITE englobée ne contenant qu'un seul élément, à savoir une référence vers une ENTITE B, nous considérerons qu'il existe une liaison allant de A vers B, de type ENTITE-REFERENCE ; ce type est analogue à un INVERSE du point de vue relationnel ; la différence réside dans un choix d'implantation physique.

Exemple 2.5

ENTITE EMPLOYE (A)
- NOM
- ADRESSE
- ENTITE VEHICULE
  - VOITURE référence une VOITURE

ENTITE VOITURE (B)
- NUMERO
- ASSURANCE

Deux seuls groupes de constituants sont à considérer : EMPLOYE et VOITURE ; ils sont reliés par une liaison de type ENTITE-REFERENCE.

En conclusion : A et B désignant deux ENTITES

STRUCTURE SOCRATE		RELATIONS
REFERENCE de A vers B	→	$k(A) \rightarrow k(B)$
INVERSE ou ANNEAU de A vers B	↗	$[k(A), k(B)]$
ENTITE-REFERENCE de A vers B	↗	
ENTITE ENGLOBEE A dans B	→	$k(B) \subset k(A)$

Tableau 2.1 - Aspects relationnels d'une structure SOCRATE

2.2.2.b Obtention de relations n-aires correspondant aux ENTITES

La relation n-aire associée à l'ENTITE A est formée sur l'ensemble des constituants de A et l'ensemble des index associés à A ;
si tous les constituants des index ne se trouvant pas dans A, ils se trouvent dans les index

- soit d'une ENTITE englobant A (cf. exemple 2.2)
- soit d'une ENTITE B vers laquelle l'ENTITE A fait référence :

Exemple 2.6

ENTITE ENFANT (A)
- PARENT référence une PERSONNE
- PRENOM
- AGE

ENTITE PERSONNE (B)
- NOM
- ADRESSE

La relation associée à A est formée sur [NOM, PRENOM, AGE], PARENT n'étant le nom que d'un pointeur, et NOM étant l'index de la relation PERSONNE.

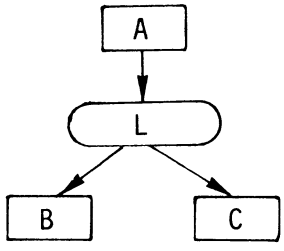
2.2.3. Aspects relationnels d'une structure CODASYL

Une structure CODASYL est formée d'ENREGISTREMENTS reliés entre eux par différents types de liaisons. Une réalisation d'un ENREGISTREMENT est notée enregistrement.

2.2.3.a Les différents types de liaisons entre ENREGISTREMENTS sont :

- LIEN :

Un LIEN est un ensemble d'au moins deux ENREGISTREMENTS dont l'un (*et un seul*) est le PARENT du LIEN et les autres sont les ENFANTS



L = (A, B, C)

Le LIEN L a comme PARENT : A
et comme ENFANTS : B et C.

Un LIEN, pour exister, doit avoir un PARENT et au moins un ENFANT.
Un ENREGISTREMENT ne peut pas être PARENT et ENFANT d'un même lien.
Un ENREGISTREMENT peut être plusieurs fois PARENTS de LIENS différents et plusieurs fois ENFANTS de LIENS différents.

De telles spécifications montrent qu'une *structure CODASYL* est une *structure réseau*.

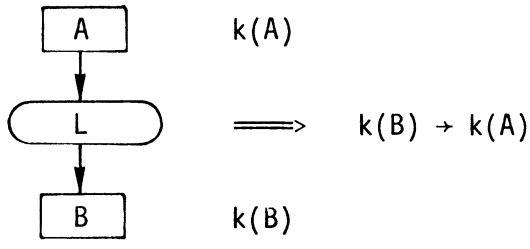
Une réalisation d'un LIEN est notée lien :
un lien ℓ de L est un ensemble de réalisations du PARENT A de L et des différents ENFANTS B et C de L telles que les propositions suivantes soient vérifiées ([8], p. 44-45) :

P1 : un enregistrement ne peut apparaître dans plus d'un lien.

P2 : chaque lien contient *une* réalisation de son PARENT ; en fait, celle-ci est une condition nécessaire de l'existence du lien et permet de le distinguer des autres liens.

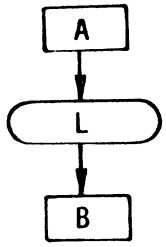
D'après ces propositions, une réalisation b d'un ENFANT B ne peut appartenir qu'à un lien ℓ (Proposition P1) ; comme un lien contient *une* réalisation a de son PARENT, à une réalisation b ne peut correspondre qu'une réalisation a :

ainsi, si k(B) et k(A) désignent respectivement des index des relations associées à A et B, il existe la *relation fonctionnelle* $k(B) \rightarrow k(A)$.



Dans CODASYL, toute liaison entre ENREGISTREMENTS est faite par un LIEN ; à celui-ci peuvent être associées des règles particulières d'élimination ou de création des réalisations d'un ENREGISTREMENT ENFANT du LIEN.
Seul, le cas du LIEN HIERARCHIQUE est intéressant du point de vue relationnel.

. LIEN HIERARCHIQUE*

Si 

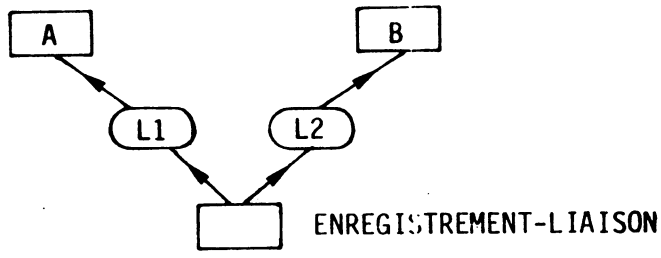
A est PARENT, B ENFANT ;

L est LIEN HIERARCHIQUE,

ce qui conditionne l'existence de toute réalisation de B.
(cf. propriété des structures arborescentes, § 2.1).

La relation réellement implantée et associée à B a un index k(B) qui contient donc celui de A : k(A) (cf. structure arborescente) ; ainsi, la relation fonctionnelle k(B) → k(A) est triviale.

. ENREGISTREMENT-LIAISON est un autre type particulier de liaison entre ENREGISTREMENTS ;
un ENREGISTREMENT-LIAISON est déclaré dans la structure CODASYL comme un ENREGISTREMENT ;
mais : - il n'est PARENT d'aucun LIEN
- il est ENFANT de deux liens dont les PARENTS sont respectivement A et B
- il ne contient comme constituants que k(A) et k(B) de manière virtuelle.



Cet ENREGISTREMENT-LIAISON définit en fait la relation binaire définie sur k(A), k(B).

---

*Dans la terminologie CODASYL, il correspond à un LIEN dont B serait un ENFANT "A VIE" et "A DECLARER" ([4]).

A un ENREGISTREMENT-LIAISON est associée une liaison de type ENREGISTREMENT-LIAISON, reliant les deux ENREGISTREMENTS A et B ; les LIENS L1 et L2 sont ignorés.

. GROUPE REPETITIF correspond à un sous-ensemble de constituants d'un même ENREGISTREMENT, dont une réalisation peut contenir plusieurs réalisations de ce sous-ensemble. C'est une autre forme d'une structure hiérarchique.

A l'ENREGISTREMENT A et au groupe répétitif, sont associés deux groupes de constituants reliés entre eux par une liaison de type LIEN HIERARCHIQUE. En effet, du point de vue logique, ce cas est identique à celui du LIEN HIERARCHIQUE.

En conclusion, A et B désignent deux ENREGISTREMENTS

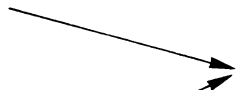
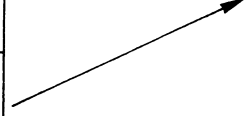
STRUCTURE CODASYL		RELATIONS
LIEN B : PARENT A : ENFANT		k(A) → k(B)
LIEN HIERARCHIQUE B : PARENT A : ENFANT		k(B) ⊂ k(A)
A : GROUPE REPETITIF de B		
ENREGISTREMENT-LIAISON ENTRE A et B		[k(A), k(B)]

Tableau 2.2 - Aspects relationnels d'une structure CODASYL

2.2.3.b Obtention des relations n-aires correspondant aux ENREGISTREMENTS

La relation n-aire associée à l'ENREGISTREMENT A est formée sur l'ensemble des constituants de A et l'ensemble des index associés à A.

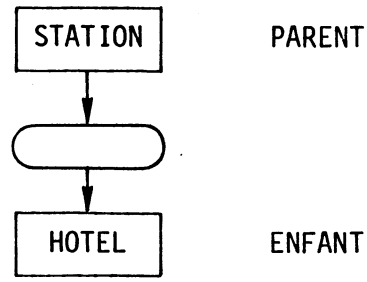
Si tous les constituants des index ne se trouvent pas dans A, ils se trouvent dans les index d'ENREGISTREMENTS PARENTS de LIENS (ou de LIENS HIERARCHIQUES) dont A est un ENFANT.

Exemple 2.7

ENREGISTREMENT HOTEL (A)
- STATION VIRTUEL
- NOM-HOTEL
- NOMBRE-DE-CHAMBRES

ENREGISTREMENT STATION
- NOM-STATION
- ALTITUDE

LIEN HOTEL-STATION
- PARENT : STATION
- ENFANT : HOTEL



La relation associée à A est formée de [NOM-STATION, NOM-HOTEL, NOMBRE-DE-CHAMBRES]. STATION dans l'ENREGISTREMENT HOTEL n'est qu'un pointeur, et NOM-STATION est l'index de STATION.

2.3. DEFINITION D'UNE STRUCTURE RESEAU ET TRANSFORMATION DES STRUCTURES CODASYL ET SOCRATE EN UNE STRUCTURE RESEAU

Après avoir présenté les aspects relationnels des différents composants d'une structure réseau, nous allons étudier la transformation d'une telle structure en un ensemble de relations.

Dans un premier temps, nous allons définir une structure réseau indépendante des systèmes utilisés SOCRATE, CODASYL ..., que nous appellerons structure RESEAU.

Nous allons alors définir la transformation d'une structure SOCRATE, puis CODASYL en une structure RESEAU. Ensuite, nous étudierons la transformation d'une structure RESEAU en un ensemble de relations.

2.3.1. Définition d'une structure RESEAU

Une structure RESEAU est formée :

- de *groupes* de constituants avec leur index : si X désigne un groupe de constituants, k(X) désigne son index
- des *liaisons entre deux groupes* telles que :
  X et Y désignant deux groupes

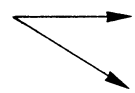
liaison fonctionnelle (R-F) allant de X vers Y		$k(X) \rightarrow k(Y)$ non triviale
liaison hiérarchique (H) allant de X vers Y		$k(Y) \subset k(X)$
liaison binaire (B) allant de X vers Y ou de Y vers X		[k(X), k(Y)] n'est ni fonctionnelle ni hiérarchique

Tableau 2.3 - Aspects relationnels d'une structure RESEAU

Dans les exemples, nous représenterons simplement une structure RESEAU par un graphe orienté de noeuds (les groupes) et d'arêtes (les liaisons).

Suivant le même principe que pour les structures SOCRATE ou CODASYL, l'index d'un groupe X peut contenir celui d'un autre groupe Y qui est relié à X par une liaison fonctionnelle ou hiérarchique allant de X vers Y.

2.3.2. Transformation d'une structure SOCRATE ou CODASYL en une structure RESEAU

Il s'agit, à partir d'une structure SOCRATE ou CODASYL, de construire la structure RESEAU associée :

- Construction des groupes

pour SOCRATE : toutes les ENTITES donnent naissance à des GROUPES, excepté les ENTITES ENGLOBEES qui ne contiennent qu'une référence.

pour CODASYL : tous les ENREGISTREMENTS et GROUPES REPETITIFS fournissent des GROUPES, excepté les ENREGISTREMENTS-LIAISON.

- Construction des liaisons

Les paragraphes 2.2.2 et 2.2.3 permettent de classifier les différentes liaisons SOCRATE ou CODASYL en liaisons RESEAU. Le tableau suivant récapitule ces résultats.

X et Y désignant deux groupes :

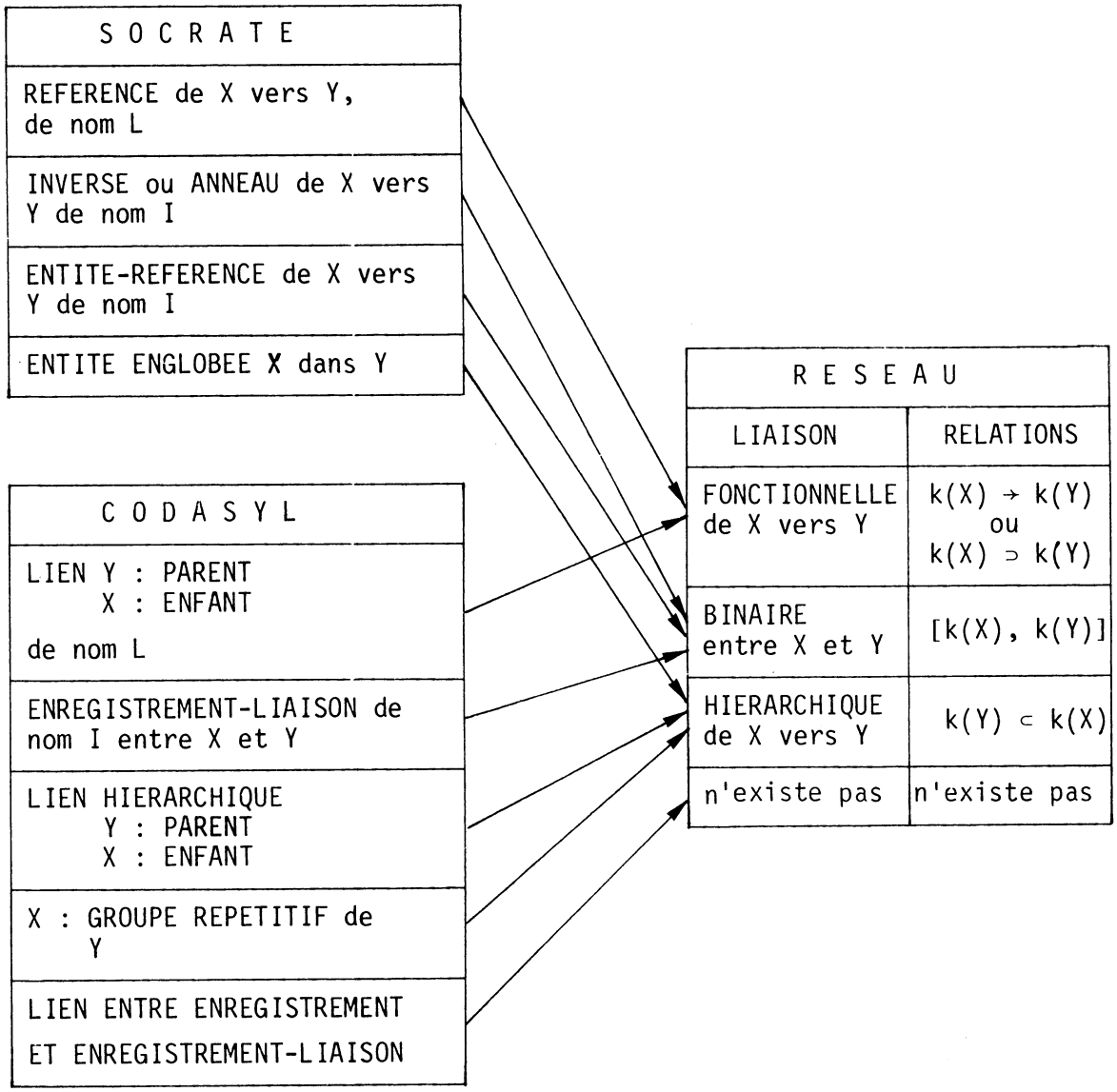


Tableau 2.4 - Obtention des liaisons d'une structure RESEAU à partir d'une structure SOCRATE ou CODASYL

2.4. TRANSFORMATION D'UNE STRUCTURE RESEAU EN UN ENSEMBLE DE RELATIONS

Donnons maintenant les éléments algorithmiques de la transformation d'une structure RESEAU en un ensemble de relations ; cette transformation constitue la seconde partie de la transformation d'une structure SOCRATE ou CODASYL en un ensemble de relations.

A partir d'une structure RESEAU donnée, les relations proviennent :

1) - des groupes de constituants d'une structure RESEAU.
Les relations obtenues sont formées sur l'ensemble des constituants du groupe et de ses index (cf. § 2.2.2.b, 2.2.3.b, 2.3.1).
Comme ces index ne sont pas décelables automatiquement, seul un apport d'informations peut les fournir.

2) - des liaisons entre groupes : les relations sont obtenues à l'aide du tableau 2.3 :

. à chaque liaison fonctionnelle allant de X vers Y correspond une relation fonctionnelle :

$$k(X) \rightarrow k(Y)$$

celle-ci peut être triviale si un des index de X contient k(Y).

. à chaque liaison binaire entre deux groupes X et Y correspond une relation définie sur [k(X), k(Y)] avec comme index [k(X), k(Y)].

. à chaque liaison hiérarchique allant de X vers Y correspond une relation fonctionnelle : $k(X) \rightarrow k(Y)$; mais celle-ci est triviale car $k(Y) \subset k(X)$.

Ainsi, pour obtenir les relations d'une structure RESEAU, le responsable de la base de données doit *apporter des informations* :

- il doit préciser les index des différents groupes
- il doit indiquer les constituants synonymes appartenant à des groupes différents ; ces constituants doivent en effet être considérés comme formant un seul constituant au niveau des relations.

Remarque 1 :

On obtient ainsi un ensemble de relations à partir d'une structure réseau d'une base de données ; mais il peut en exister d'autres relatives à cette structure :

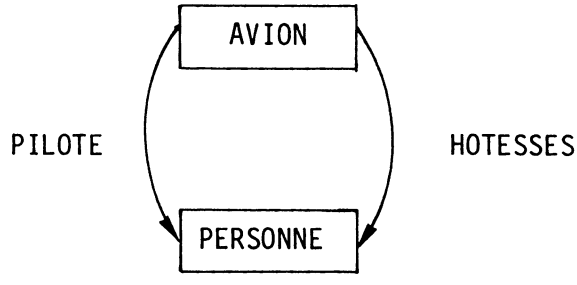
- toutes celles que l'on peut déduire de ces premières relations par composition, projection ou combinaison de ces deux opérations ;
- toutes celles non contenues dans la structure mais que le responsable de la base de données peut fournir.

Dans la suite de notre démarche, nous appellerons *relations traduites d'une structure* S toutes les relations que l'on peut obtenir à partir de S, à l'aide de l'algorithme de transformation prédécent.
Nous appellerons *relations déduites* d'une structure S les relations qui s'obtiennent des relations traduites par des opérations de composition et de projection.
L'ensemble des relations traduites et déduites forme l'ensemble des *relations représentées* dans la structure.

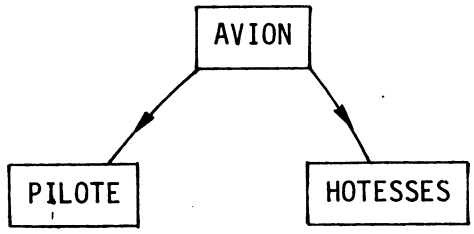
Remarque 2 :

Avant d'exécuter le processus de transformation d'une structure RESEAU en un ensemble de relations, il faut parfois modifier la structure RESEAU considérée, pour une *meilleure identification des constituants*.

Ainsi, par exemple, soient les deux GROUPES AVION et PERSONNE reliés entre eux de la manière suivante :



Il faut alors transformer cette structure en dédoublant le GROUPE PERSONNE :



2.5. EXEMPLE DE TRANSFORMATION D'UNE STRUCTURE SOCRATE OU CODASYL EN UN ENSEMBLE DE RELATIONS

Pour illustrer ce processus de transformation, nous considérons un système de réservation de chambres d'*hôtels* et de *stations de ski*.

Un *client* potentiel interroge le système de réservation et indique son intention de retenir une chambre. Cette *demande* doit contenir les informations suivantes :

- le nom, l'adresse, le numéro de téléphone du client,
- la période désirée,
- la classe de l'hôtel et le nombre de chambres désirés.

Le service de réservation cherche ensuite à satisfaire cette demande et donc à la transformer en une réservation effective pour l'envoyer au client.

Pour faire une telle opération, il doit posséder différents renseignements sur les stations et les hôtels, ainsi que sur les tarifs des hôtels en fonction de la période de l'année considérée.

Nous allons indiquer la modélisation de cette application en structure réseau, en utilisant les termes SOCRATE, puis CODASYL.

Les structures que nous donnons ne contiennent que les seules informations nécessaires à notre démarche. Pour écrire la structure CODASYL, nous nous sommes servis de [8] et des recommandations de [23].

Ensuite, nous transformerons ces structures en une structure RESEAU, puis celle-ci en un ensemble de relations.

Remarque : Nous reprendrons cet exemple à la fin des chapitres 3 et 5 pour illustrer notre démarche.

2.5.1. Structure SOCRATE

entité STATION (S)
début
NOM (s)
ALTITUDE } (s_1)
DEPARTEMENT }
fin

entité HOTEL (H)
début
NOM (h)
NOM-STATION (s)
CLASSE } (h_1)
NOMBRE-DE-CHAMBRES }
entité CONDITIONS (P)
début
PERIODE (p)
NOMBRE DE CHAMBRES LIBRES } (p_1)
PRIX }
fin
fin

entité RESERVATION (R)
début
HOTEL référence un HOTEL
CLIENT référence un CLIENT
CONDITIONS référence une CONDITIONS d'un HOTEL
NOMBRE DE CHAMBRES
PRIX TOTAL } (r_1)
NOMBRE DE PERSONNES
fin

entité CLIENT (C)
début
NOM (c)
ADRESSE } (c_1)
TELEPHONE }
fin

entité DEMANDE (D)
début
CLIENT référence un CLIENT
PERIODE (p)
RESERVATION référence une RESERVATION
STATION référence une STATION
CLASSE D'HOTEL } (d_1)
NOMBRE DE CHAMBRES }
fin

2.5.2. Structure CODASYL

enregistrement STATION (S)
NOM
ALTITUDE
DEPARTEMENT

index est NOM

enregistrement HOTEL (H)
NOM
NOM-STATION
CLASSE
NOMBRE DE CHAMBRES

index est NOM, NOM-STATION

enregistrement CONDITIONS (P)
NOM HOTEL virtuel
NOM STATION virtuel
PERIODE
NOMBRE DE CHAMBRES LIBRES
PRIX

index est NOM-HOTEL, NOM-STATION, PERIODE

enregistrement RESERVATION (R)
NOM-HOTEL virtuel
NOM-STATION virtuel
NOM-CLIENT virtuel
NOMBRE DE CHAMBRES
PRIX TOTAL
NOMBRE DE PERSONNES

index est NOM-HOTEL, NOM-STATION, NOM-CLIENT, PERIODE

enregistrement CLIENT (C)
NOM
ADRESSE
TELEPHONE

index est NOM

enregistrement DEMANDE (D)
NOM-CLIENT virtuel
PERIODE
CLASSE D'HOTEL
NOMBRE DE CHAMBRES
NOM STATION virtuel
NOM HOTEL virtuel

index est NOM-CLIENT, PERIODE

lien HOTEL-CONDITIONS
parent est HOTEL
enfant est CONDITIONS
lien est hiérarchique

NOM, NOM STATION dans parent correspondent à NOM HOTEL,
NOM STATION dans enfant

lien STATION-DEMANDE
parent est STATION
enfant est DEMANDE

NOM dans parent correspond à NOM STATION dans enfant

lien HOTEL-RESERVATION
parent est HOTEL
enfant est RESERVATION

NOM, NOM STATION dans parent correspondent à NOM HOTEL, NOM STATION dans enfant

lien CONDITIONS-RESERVATION
parent est CONDITIONS
enfant est RESERVATION

NOM HOTEL, NOM STATION, PERIODE dans parent correspondent à NOM HOTEL, NOM STATION, PERIODE dans enfant

lien CLIENT-RESERVATION
parent est CLIENT
enfant est RESERVATION

NOM dans parent correspond à NOM CLIENT dans RESERVATION

lien CLIENT-DEMANDE
parent est CLIENT
enfant est DEMANDE

NOM dans parent correspond à NOM CLIENT dans DEMANDE

lien DEMANDE-RESERVATION
parent est RESERVATION
enfant est DEMANDE

NOM CLIENT, PERIODE dans parent correspondent à NOM CLIENT, PERIODE dans enfant.

2.5.3. Structure RESEAU obtenue à partir de la structure SOCRATE ou CODASYL précédente

Voici la structure RESEAU obtenue sous forme de graphe. A un noeud, nous n'avons mentionné que le nom du groupe, à une arête, le nom et le type de la liaison concernée.

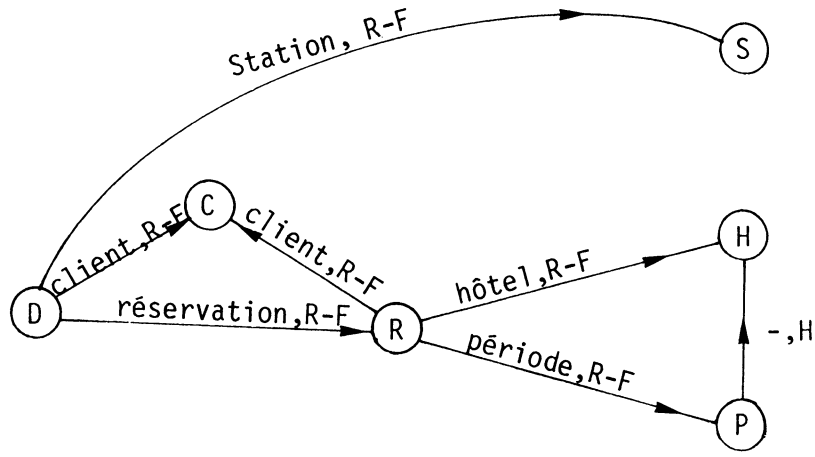


Fig. 2.1 - Structure RESEAU de l'exemple

2.5.4. Apport d'informations

. Index des groupes

Groupes	Index
S	s
H	(h,s)
P	(h,s,p)
R	(c,h,s,p)
D	(c,p)
C	c

. Constituants synonymes

NOM-STATION dans le groupe HOTEL est synonyme de NOM dans le groupe STATION.

2.5.5. Relations traduites de la structure RESEAU précédente

Relations obtenues à partir des groupes

groupe S ⟹ relation $S[s,s_1]$

groupe H ⟹ relation $H[h,s,h_1]$

groupe P ⟹ relation $P[h,s,p,p_1]$

groupe R ⟹ relation $R[c,h,s,p,r_1]$

groupe D ⟹ relation $D[c,p,d_1]$

groupe C ⟹ relation $D[c,c_1]$

Relations obtenues à partir des liaisons

relation DC[C,P]	absorbée par la relation D
relation DR[c,p,h,s]	absorbée par la relation R
relation DS[c,p,s]	absorbée par la relation R
relation RC[c,h,p,s]	absorbée par la relation R
relation RP[c,h,p,s]	absorbée par la relation R
relation RH[c,h,p,s]	absorbée par la relation R
relation HS[h,s]	absorbée par la relation H

Relations fonctionnelles

associées aux groupes

S $f_1 = (s \rightarrow s_1)$

H $f_2 = (h,s \rightarrow h_1)$

P $f_3 = (h,s,p \rightarrow p_1)$

R $f_4 = (c,h,p,s \rightarrow r_1)$

D $f_5 = (c,p \rightarrow d_1)$

C $f_6 = (c \rightarrow c_1)$

associées aux liaisons entre groupes

DR $f_7 = (c,p \rightarrow h)$

DS $f_8 = (c,p \rightarrow s)$

2.6. CONCLUSIONS

Notre objectif est de déceler les relations qui sont contenues implicitement dans une structure d'une base de données construite pour des systèmes comme SOCRATE ou CODASYL ...

Pour l'atteindre, nous donnons les spécifications de la transformation d'une telle structure en un ensemble de relations. En cherchant à rendre la plus grande partie de cette transformation, indépendante du système utilisé, nous définissons une structure réseau générale (structure RESEAU) et nous construisons ce processus en deux étapes comme le montre le schéma suivant :

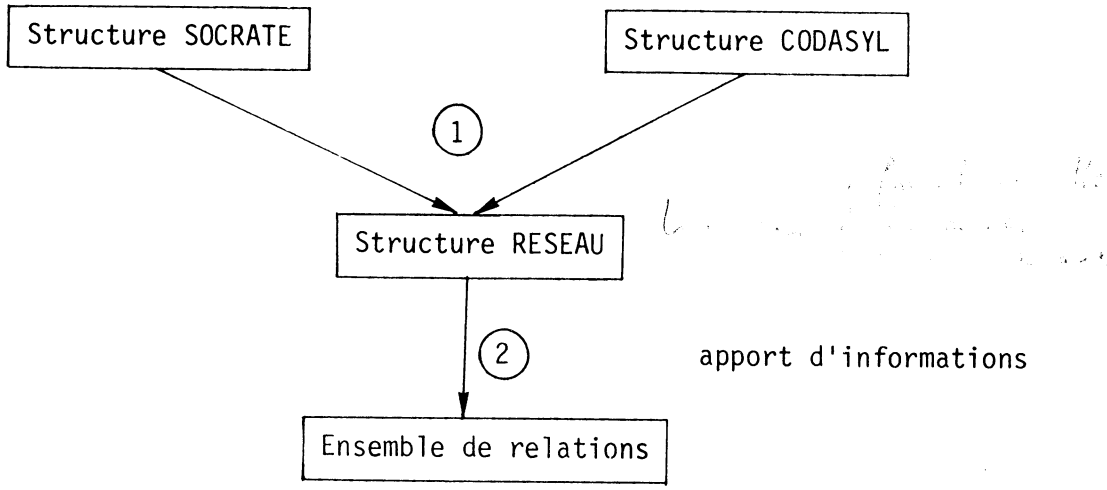


Tableau 2.5 - Transformation d'une structure SOCRATE ou CODASYL en un ensemble de relations

La première partie (1) est entièrement automatique.
La seconde (2) demande un apport d'informations de la part du responsable de la base de données. Cependant, elle est entièrement indépendante du système utilisé pour gérer cette base.

C'est par cette transformation que l'on peut expliciter les relations que contient implicitement une structure. Le fait qu'une telle opération nécessite un apport d'informations montre bien que la structure réseau d'une base de données contient moins d'informations

sémantiques que l'ensemble de relations associées à cette base.

Cette remarque montre l'importance du modèle relationnel dans la représentation du champ d'application d'une base de données. C'est la raison pour laquelle nous n'étudierons la restructuration d'une base ou la fusion de plusieurs bases, qu'après avoir extrait de leurs structures l'ensemble de relations correspondant.

C H A P I T R E 3

ELIMINATION DE REDONDANCE ET D'INCOHERENCES D'INFORMATIONS

S O M M A I R E

INTRODUCTION

Nous allons montrer dans la première partie de ce chapitre (§ 3.1) comment la structure d'une base de données peut provoquer des imperfections au niveau de l'exploitation :

- redondances d'informations,
- incohérences,
- règles d'intégrité non assurées.

Face à une structure contenant de telles imperfections, nous allons chercher à la réorganiser. Pour cela, nous considérons l'ensemble des relations que l'on peut traduire (§ 2.4) de cette structure (Chapitre 2) et nous étudions comment le transformer dans un autre ensemble de relations de telle sorte que ce dernier conduit à une structure dépourvue d'imperfections, par un processus inverse à celui du Chapitre 2.

Le problème à résoudre est alors de savoir :
comment choisir, à partir d'un ensemble de relations, celle qui doivent être traduites dans la nouvelle structure, de telle sorte que les mises à jour et les règles d'intégrité concernant aussi bien ces relations que celles que l'on peut déduire d'elles (§ 2.4) par projection ou composition, ne causent aucune imperfection.

Avant de donner des éléments de réponse à cette question, nous rappelons les notions de fermeture et de couverture minimale d'un ensemble de relations fonctionnelles, ainsi que les propriétés de la couverture minimale.

De ce fait, nous pensons traiter les sujets des questions A, D, E, M, O de l'Introduction ayant trait à :

- l'élimination de la redondance de relations,
- l'élimination d'incohérences d'informations,
- la détermination d'un ensemble non redondant de règles d'intégrité.

Nous montrerons que ce dernier point est lié aux précédents de manière théorique.

Toute cette étude nous conduit vers la construction d'un algorithme : son mécanisme et sa justification se trouvent dans l'annexe A.

3.1. IMPERFECTIONS D'UNE BASE ET REGLES D'INTEGRITE

Nous allons déclarer les différents types d'imperfections et de règles d'intégrité que nous avons pris en compte.

3.1.a. Imperfections d'une base de données dues à sa structure

3.1.a.1. Une des premières causes provient de la *redondance d'informations* qui conduit à des *contrôles de cohérence* longs et coûteux.

Nous allons montrer leur origine sur deux exemples :

Exemple 3.1

Soit la relation R_1 qui associe à un client et à un produit l'adresse du client, le nombre d'articles du produit achetés en une année, le nombre de commandes passées en une année :

R_1[P,C,AC,NP,M] formée sur les constituants

P : Produit
C : Client
AC : adresse de client
NP : nombre d'articles du produit
M : nombre de commandes

avec les relations fonctionnelles suivantes :

P,C → NP,M
C → AC

et dont une réalisation se compose de :

P	C	AC	NP	M
p_1	c_1	ac_1	np_1	m_1
p_2	c_1	ac_1	np_2	m_2
p_3	c_1	ac_1	np_3	m_3
p_1	c_2	ac_2	np_4	m_4
p_3	c_3	ac_3	np_5	m_5

Exemple 3.2

Soit la relation R_2 qui associe à un projet et à un produit le nom du fournisseur avec son adresse, et le nombre d'articles du produit reçus pour le projet : R_2[PR,P,F,AF,N] formée sur

PR : Projet
P : Produit
F : Fournisseur
AF : Adresse de Fournisseur
N : Nombre d'articles du produit

avec les relations fonctionnelles

$$\begin{cases} PR,P \rightarrow F,N \\ F \rightarrow AF \end{cases}$$

et dont une réalisation se compose de :

PR	P	F	AF	N
pr_1	p_1	f_1	af_1	n_1
pr_1	p_2	f_1	af_1	n_2
pr_2	p_3	f_2	af_2	n_3
pr_3	p_4	f_3	af_3	n_4
pr_3	p_1	f_2	af_2	n_5

Si les deux relations R_1 et R_2 des deux exemples précédents sont traduites dans une structure réseau, alors pour toute mise à jour de l'adresse du client c_1 dans R_1 ou de l'adresse du fournisseur f_1 dans R_2, il faut, pour assurer l'intégrité des relations fonctionnelles C → AC pour R_1 et F → AF pour R_2, rechercher toutes les entités relatives au client c_1 et au fournisseur f_1 et donc balayer tout le fichier correspondant. Par contre, si les sous-relations [C,AC]R_1 et [F,AF]R_2 étaient traduites, de telles mises à jour ne causeraient la modification que des seules entités relatives au client c_1 pour R_1, au fournisseur f_1 pour R_2 ; l'intégrité des relations fonctionnelles serait alors assurée. C'est dans le premier cas que nous parlons de lourdeur de contrôles de cohérence.

3.1.a.2. La structure peut être à l'origine de *pertes d'informations* lors d'élimination d'entités.

Ainsi, dans les exemples précédents, et dans le cas où les relations R_1 et R_2 sont traduites dans la structure, l'élimination du produit p_3 dans R_1 ou du projet pr_3 dans R_2 a pour conséquences respectivement la perte du client c_3 pour R_1 ou celle du fournisseur f_3 pour R_2.

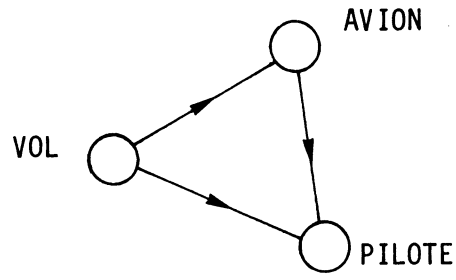
3.1.a.3. La structure peut provoquer des *incohérences d'informations*.

Nous allons présenter ce point sur deux exemples :

Exemple 3.3

Soit la relation R_3 qui associe à un vol l'avion qui l'effectue et le pilote qui l'assure :
R_3[VOL, AVION, PILOTE] avec les relations fonctionnelles :

VOL → AVION
VOL → PILOTE
AVION → PILOTE



dont une réalisation se compose de :

VOL	AVION	PILOTE
v_1	a_1	p_1
v_2	a_1	p_1
v_4	a_3	p_2
v_3	a_2	p_2

Comme il existe la relation fonctionnelle VOL → AVION, R_3 peut se décomposer en :

$$R_3 = [\text{VOL, AVION}]R_3 * [\text{VOL, PILOTE}]R_3 = R_{31} * R_{32} \quad \text{(cf. § 1.4)}$$

Nous supposons justement que R_3 est représentée dans la structure à l'aide des relations R_{31} et R_{32} et ainsi que l'on a deux sous-relations :

R_{31}

VOL	AVION
v_1	a_1
v_2	a_1
v_3	a_2
v_4	a_3

R_{32}

VOL	PILOTE
v_1	p_1
v_2	p_1
v_3	p_2
v_4	p_2

Supposons que l'entité (<VOL : v_4>, <AVION : a_3>) de R_{31} soit modifiée en (<VOL : v_4>, <AVION : a_1>). Cette mise à jour est cohérente vis à vis de R_{31} et de R_{32} mais non de R_3.

En effet, si l'on compose les nouvelles relations R_{31} et R_{32}, on obtient :

VOL	AVION	PILOTE
v_1	a_1	p_1
v_2	a_1	p_1
v_3	a_2	p_2
v_4	a_1	p_2

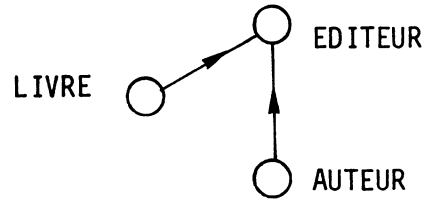
La relation fonctionnelle AVION → PILOTE est détruite.

Exemple 3.4

Soit la relation R_4 qui associe à un livre son ou ses auteurs et son éditeur :
R_4[LIVRE, AUTEUR, ADITEUR].

Nous supposons, de plus, les relations fonctionnelles suivantes :

LIVRE → EDITEUR
AUTEUR → EDITEUR.



Une réalisation de R_4 est donnée par :

LIVRE	AUTEUR	EDITEUR
ℓ_1	a_1	e_1
ℓ_1	a_2	e_1
ℓ_2	a_1	e_1
ℓ_3	a_3	e_2

qui est décomposée en R_{41}, R_{42} :

R_{41} = [LIVRE, AUTEUR] R_4

R_{42} = [LIVRE, EDITEUR] R_4

et $R_4 = R_{41} * R_{42}$ à cause de la relation fonctionnelle

LIVRE → EDITEUR

Supposons que R_4 soit représentée dans la structure par R_{41} et R_{42}.

Les réalisations de R_{41} et R_{42} sont obtenues par projection :

R_{41}

LIVRE	AUTEUR
ℓ_1	a_1
ℓ_1	a_2
ℓ_2	a_1
ℓ_3	a_3

R_{42}

LIVRE	EDITEUR
ℓ_1	e_1
ℓ_2	e_1
ℓ_3	e_2

Supposons que l'entité (<LIVRE : ℓ_2>, <EDITEUR : e_1>) de R_{42} soit transformée en (<LIVRE : ℓ_2>, <EDITEUR : e_2>) de R_{42} ; cette mise à jour cohérente vis à vis de R_{41} et de R_{42} ne l'est pas pour R_4 : en effet, si l'on compose les nouvelles collections R_{41} et R_{42}, on obtient :

LIVRE	AUTEUR	EDITEUR
ℓ_1	a_1	e_1
ℓ_1	a_2	e_1
ℓ_2	a_1	e_2
ℓ_3	a_3	e_2

La relation fonctionnelle AUTEUR → EDITEUR est détruite.

3.1.a.4. Pour une plus grande clarté, nous allons classer ces imperfections en deux catégories indépendantes :

Imperfections intra-relations :

Elles sont liées à une seule relation traduite dans une structure ;
elles provoquent :

- des contrôles de cohérence trop longs
- des pertes d'informations indésirables.

Imperfections inter-relations :

Elles sont liées à une relation obtenue par composition de plusieurs relations traduites dans une structure ;
elles causent des incohérences d'informations.

Ce sont justement ces deux sortes d'imperfections que nous tenterons d'éviter dans la conception de structures de bases de données en choisissant convenablement l'ensemble des relations à traduire.

3.1.b. Règles d'intégrité d'un système d'informations

Les règles d'intégrité expriment des propriétés intrinsèques des informations et sont fournies par quelqu'un de familier avec le système d'informations considéré.

Certaines d'entre elles peuvent s'exprimer de manière précise et concise par des relations fonctionnelles comme l'illustre l'exemple suivant :

Exemple 3.5

Considérons l'organisation de la force de vente d'une entreprise et donc la relation R_5 : FORCE-DE-VENTE ; elle est formée sur les constituants :

Représentants	: R
Types de Points de Vente	: TV
Département géographique	: D
Gammes de Produits	: G
Produits	: P
Points de Vente	: PV

R_5[R, TV, D, G, P, PV]

- Certaines règles d'intégrité de l'organisation de la force de vente sont générales :

  * un point de vente appartient à un seul type de point de vente et à un seul département ;

  * un produit ne fait partie que d'une seule gamme de produits.

Elles s'expriment très simplement au moyen de relations fonctionnelles :

(1) PV → TV
(2) PV → D
(3) P → G

- D'autres sont plus caractéristiques de l'entreprise étudiée

  * si le représentant a la charge d'un seul département ; alors un point de vente est visité par un seul représentant.

    En termes de relations fonctionnelles :

    (4) R <--> D

    et (5) PV → R qui se déduit de (2) et (4) par transitivité.

  * si au contraire le représentant vend une seule gamme de produits à cause de la complexité des différentes gammes, alors un point de vente peut être visité par plusieurs représentants de la même entreprise ; la seule relation fonctionnelle est donc :

    (6) R → G.

  * si le représentant vend à un seul type de point de vente à cause de la complexité des règlements commerciaux et de plus si à un type de point de vente et à un département ne correspond qu'un représentant :

    (7) R → TV

    (8) TV,D → R,

    on en déduit alors que chaque point de vente n'est visité que par un représentant :

    (9) PV → R qui se déduit par pseudo-transitivité de (1), (2), (8).

Ainsi, les relations fonctionnelles permettent d'exprimer de manière concise et claire certaines propriétés sémantiques d'un système d'informations. Si elles sont très nombreuses, il est primordial d'en réduire le nombre. Une telle opération peut être réalisée en déduisant de l'ensemble de départ, un sous-ensemble minimal les recouvrant toutes.

Nous allons chercher à construire un tel ensemble à partir d'un ensemble de relations fonctionnelles de départ.

3.2. FORMES DE RELATION

Dans ce chapitre, nous allons montrer comment les formes normales de relations, introduites par Codd [9], permettent de concevoir des structures dépourvues d'imperfections intra-relations mises en évidence précédemment.

Au préalable, nous allons rappeler les définitions d'un index d'une relation et des différentes formes.

3.2.1. Index d'une relation

Nous avons donné au § 1.3.1., (p. I.8) la définition d'un index d'une relation.

Remarque : Une relation formée sur A peut avoir plusieurs index comme le montre l'exemple suivant.

Exemple 3.6 relatif à un emploi du temps d'un lycée

R_6[HEURE, PROFESSEUR, CLASSE, COURS]

avec les relations fonctionnelles :

PROFESSEUR → CLASSE
HEURE, CLASSE → PROFESSEUR, COURS
HEURE, PROFESSEUR → COURS

(HEURE, PROFESSEUR) et (HEURE, CLASSE) sont deux index de la relation R_6.

Propriété

Si l'ensemble E des relations fonctionnelles de R est sans circuit, alors R possède un seul index.

En effet, si R possédait deux index I et J, il existerait alors les sous-relations fonctionnelles : I → J et J → I par définition même de l'index : E contiendrait alors un circuit, ce qui est contraire à l'hypothèse.

3.2.2. Relation en troisième forme

Soit une relation R formée sur $\mathcal{a}$.

Une relation fonctionnelle élémentaire (§ 1.3.1) A → B sera dite *directe* s'il n'existe aucun constituant composé C partie de $\mathcal{a}$ non index, tel que :

$$A \rightarrow C \rightarrow B$$

où les relations fonctionnelles A → C et C → B sont élémentaires.

Dans le cas où C serait un index de R, on se trouverait dans le cas suivant : A <—> C → B. Par extension, on considérera que la relation fonctionnelle élémentaire A → B est directe.

Une relation R est en troisième forme si pour tout index I et pour tout constituant A n'appartenant pas à I,

I → A est *élémentaire directe.*

Exemple 3.7 (cf. exemple 3.6)

R_6[HEURE, PROFESSEUR, CLASSE, COURS] est en troisième forme.

Remarque : Cette définition est celle de Delobel [14] et correspond à celle appelée,dans les publications anglo-saxonnes, forme "Boyce-Codd" (Codd [11] d'après Bernstein [6]).

La première définition de la troisième forme, donnée par Codd [9] n'était pas suffisamment précise dans le cas d'une relation avec plusieurs index comme le rappelle Bernstein [6].

3.2.3. Relation en deuxième forme et relation en première forme

- Une relation R est en seconde forme si pour tout index I et pour tout constituant A n'appartenant pas à I, I → A est *élémentaire*.

Exemple 3.8 (cf. exemple 3.3)

R_3[VOL, AVION, PILOTE] a un seul index VOL

VOL → AVION et VOL → PILOTE sont élémentaires, mais VOL → PILOTE n'est pas directe car :

VOL → AVION → PILOTE

R_3 est donc en deuxième forme ; il en est de même de R_2 dans l'exemple 3.2.

- Toute relation R qui n'est ni en troisième forme ni en deuxième forme est en *première forme*. Ceci signifie qu'il existe un index I de R et un constituant B de R n'appartenant pas à I, tel que I → B n'est pas élémentaire.

Exemple 3.9

R_1[P, C, AC, NP, M] définie dans l'exemple 3.1 est en première forme car l'index P,C est tel que la relation fonctionnelle P,C → AC n'est pas élémentaire. Il en est de même de la relation R_4 dans l'exemple 3.4.

Exemple 3.10

Soit R_{10}[A, B, C, D]

et les relations fonctionnelles : A,B → C,D

et C → B.

Les index de R_{10} sont (A,B) et (A,C) ;

A,C → B n'étant pas élémentaire, R_{10} est en première forme.

Remarque : Une relation en troisième forme satisfait également les conditions pour être en deuxième forme ; une relation en troisième ou en seconde forme peut également être considérée comme en première forme.

3.2.4. Intérêt des relations en troisième forme

Les relations en troisième forme ont la propriété de ne contenir comme relations fonctionnelles que celles dont la partie gauche est un index.

Nous allons montrer en quoi cette propriété est intéressante dans le cadre de l'élimination des imperfections intra-relations.

. *Dans le cas général*, nous allons examiner comment des imperfections intra-relations peuvent être causées par une relation R traduite dans une structure.

Soit la relation $R[M_1,M_2,M_3,M_4]$, d'index (M_1,M_2) et une réalisation C de R.

Comme (M_1,M_2) est un index de R, à chaque entité de C correspond une et une seule réalisation de (M_1,M_2) ; ainsi les seules relations réellement implantées à travers C sont R bien sûr et toutes les sous-relations de R qui contiennent l'index, par exemple $[M_1,M_2,M_3]R$, $[M_1,M_2,M_4]R$.

Pour ces sous-relations, aucune redondance d'entités n'est à redouter à cause de la propriété même de l'index et ainsi leur cohérence est assurée : ce ne sont pas elles qui provoquent des imperfections intra-relations.

Par contre, toutes les autres relations R' déduites de R par projection (ex : $[M_2,M_3]R$ ou $[M_3,M_4]R$) ne sont prises en compte que de manière indirecte : ce qui signifie qu'une réalisation r' de R' n'est accessible qu'à partir d'une réalisation i de l'index (M_1,M_2) de R.

Or, comme un index de R' n'est pas un index de R, par construction même de R', à une réalisation de R' correspondent en général plusieurs réalisations de l'index de R et donc plusieurs entités de R.

Ceci a deux conséquences importantes :

- la redondance des réalisations de R'
- la lourdeur des opérations de mises à jour, de création ou d'élimination d'une réalisation r' de R'. En effet, pour que ces opérations soient cohérentes, il faut qu'elles soient effectuées sur toutes les entités contenant r'.

Ce sont donc ces relations R' qui risquent de provoquer des imperfections intra-relations si R est traduite dans la structure.

. *Cas particulier* : une des sous-relations R' ne contenant pas l'index est fonctionnelle. Par exemple :

$M_2 \rightarrow M_3$ alors R est en première forme

ou $M_3 \rightarrow M_4$ alors R est en deuxième forme ;

ce cas est important car :

- d'une part, à une relation fonctionnelle, est associée une règle d'intégrité dont il faut assurer la cohérence ;
- d'autre part, bien souvent une telle relation intéresse l'utilisateur. Il faut donc éviter les pertes d'informations, assurer la cohérence, et fournir un accès facile à cette relation.

Or, dans une telle situation, R est en première forme ou en deuxième forme, et le respect de $(M_2 \rightarrow M_3)$ ou de $(M_3 \rightarrow M_4)$ ne peut être préservé qu'en balayant C (cf. § 3.1.a.1).

Ainsi, les relations en première et deuxième forme conduisent à des imperfections intra-relations si elles sont traduites dans une structure.

. Par contre, une relation en troisième forme peut être traduite dans la structure puisque toutes les sous-relations fonctionnelles qu'elle contient ont comme partie gauche un index.

Aussi, allons-nous chercher à construire des structures de bases de données à partir d'ensembles de relations en troisième forme pour éviter les imperfections intra-relations.

3.3. DECOMPOSITION D'UNE RELATION EN SOUS-RELATIONS DIRECTES

Nous avons mis en évidence dans le § 3.1 les imperfections intra-relations et inter-relations.
Notre premier résultat (§ 3.2.4) a été de montrer que si l'ensemble de relations à traduire en une structure ne comporte que des relations en troisième forme, alors aucune imperfection intra-relations n'est à craindre. Par contre, nous avons montré comment les relations en première et en deuxième forme conduisent également à des imperfections inter-relations.

Ainsi, représenter une relation R en première ou en deuxième forme dans une structure revient, si l'on veut éliminer les risques d'imperfections, à établir les décompositions de R en sous-relations de troisième forme et à en choisir une qui ne cause aucune imperfection inter-relation. Tel est le but de ce paragraphe : trouver et construire une telle décomposition de R.

Dans tout le § 3.3, nous supposons qu'à la relation R est associé l'ensemble E des sous-relations fonctionnelles canoniques, supposé sans circuits (§ 1.3.3).

3.3.1. Sous-relation directe d'une relation

Une sous-relation R' de la relation R est dite directe si toutes les sous-relations fonctionnelles de R' qui relient l'index* de R' aux autres constituants de R' sont *directes* (cf. § 3.2.2) par rapport à l'ensemble E des relations fonctionnelles de R.

Exemple 3.11

Soit R[A,B,C] et E = {A → B ; B → C ; A → C}.

Soit R_1 = [A,B]R ; l'index de R_1 est A et la relation fonctionnelle A → B est directe par rapport à E.

La sous-relation R_1 est donc directe.

---

* *l'index est bien unique comme le montre la propriété du § 3.2.1. p III.11*

Soit R_2 = [A,C]R ; l'index de R_2 est A, mais la relation fonctionnelle A → C n'est pas directe par rapport à E :

$$A \rightarrow B \rightarrow C ;$$

aussi, R_2 n'est pas directe.

Propriété :

Toute sous-relation R' de R qui est directe est en troisième forme par construction même.

La réciproque est fausse, l'exemple précédent le montrant bien :

R_2 est en troisième forme et pourtant R_2 n'est pas une sous-relation directe.

3.3.2. Fermeture et couverture minimale d'un ensemble de relations fonctionnelles

Nous avons rappelé dans le § 1.3 l'isomorphisme existant entre un ensemble de relations fonctionnelles canonique E et une fonction booléenne f appelée forme relationnelle dont les monômes ont tous une partie non complémentée et une seule variable complémentée, l'opération de pseudo-transitivité définie sur les relations fonctionnelles correspondant à celles de consensus des monômes.

Ainsi aux définitions de base complète de f et de base irredondante correspondent respectivement celles de fermeture et de couverture minimale de E :

- La *fermeture* de E, notée FM, est l'ensemble des relations fonctionnelles *élémentaires* (§ 1.3.1) qu'il est possible d'obtenir à partir de E en appliquant les règles de pseudo-transitivité et d'absorption.

Comme la forme relationnelle ne possède qu'une base complète, E admet une seule fermeture.

- La *couverture minimale* de E notée CM est un ensemble de relations fonctionnelles *élémentaires* tel que la fermeture de cet ensemble est égale à celle de l'ensemble de départ, cette propriété n'étant plus vraie si l'on supprime une relation fonctionnelle de la couverture minimale.

Exemple 3.12

Soit la relation R définie sur (A,B,C,D,E) et l'ensemble E de ses relations fonctionnelles :

$$E = \begin{cases} A \rightarrow B \\ B \rightarrow C \\ A,B \rightarrow C \\ C,D \rightarrow E \end{cases}$$

FM est alors :

$$\begin{cases} A \rightarrow B \\ B \rightarrow C \\ A \rightarrow C \\ C,D \rightarrow E \\ A,D \rightarrow E \\ B,D \rightarrow E \end{cases}$$

CM :

$$\begin{cases} A \rightarrow B \\ B \rightarrow C \\ C,D \rightarrow E \end{cases}$$

Dans [16] il est montré que si E ne contient *aucun circuit* (§ 1.3.3), E a une seule *couverture minimale*.

Un premier intérêt de la notion de couverture minimale vient du fait que les relations fonctionnelles peuvent représenter des règles d'intégrité. Trouver la couverture minimale revient alors à éliminer la redondance dans l'ensemble des règles d'intégrité et ainsi à trouver l'ensemble minimal de règles dont il suffit d'assurer la cohérence pour obtenir la cohérence globale.

Nous allons maintenant montrer un deuxième intérêt de la couverture minimale qui est lié à un autre aspect des relations fonctionnelles : celui d'être des sous-relations particulières de R et de pouvoir donc appartenir à une décomposition de R.

3.3.3. Couverture minimale et ensemble de sous-relations directes

Par définition une relation fonctionnelle f de la fermeture FM (§ 3.2.2) est directe si et seulement si il n'existe pas deux relations f_1 et f_2 de FM telles que :

$f = f_1 +> f_2$ où +> désigne l'opérateur de pseudo-transitivité (§ 1.3.5)

Exemple 3.13

Si FM = {A → B ; B → C ; A → C},
(A → C) n'est pas directe car (A → C) = (A → B) +> (B → C) ;
par contre (A → B) l'est.

Propriété : D'après la définition même de la couverture minimale, toute relation fonctionnelle *directe* de FM *appartient* à la *couverture minimale* CM.

Avant de démontrer la réciproque, nous allons d'abord démontrer le lemme suivant :

- soit E un ensemble de relations fonctionnelles
- soit FM sa fermeture et G le graphe associé (cf. § 1.3)
- soit $f(V_1,V_2...V_n)$ une f-expression où $V_1,V_2...V_n$ sont des relations de FM et qui est définie par les règles syntaxiques suivantes :

<f-expression> ::= <chaîne> | (<f-expression>) +> (<f-expression>)
<chaîne> ::= <RFE> | <RFE> +> (<chaîne>)

où RFE désigne une relation fonctionnelle élémentaire de FM ; à une f-expression correspond une relation fonctionnelle.

Lemme : Soit $S = f(V_1, V_2 \ldots V_n)$, alors pour tout constituant A de f, il existe dans G un chemin allant de A à d(S) où d(S) désigne la partie droite de S (§ 1.3.1).

La démonstration de ce lemme se fait par induction à partir de la définition d'une f-expression :

(a) si f est une chaîne, la proposition est vraie, elle est évidente.

(b) si f_1 et f_2 sont deux f-expressions où la proposition est vérifiée, alors la proposition l'est également sur $f = f_1 \mathbin{+>} f_2$ si cette opération est possible. En effet :

(1) si le constituant A appartient à f_2, il existe alors un chemin de A vers $d(f_2)$. Comme $f = f_1 \mathbin{+>} f_2$, $d(f) = d(f_2)$; il existe donc un chemin de A vers d(f).

(2) si le constituant A appartient à f_1, il existe alors un chemin de A à $d(f_1)$. Comme l'opération $f_1 \mathbin{+>} f_2$ est possible, $d(f_1)$ est un constituant de f_2 et donc, par transitivité, il existe un chemin de A à $d(f_2)$, c'est-à-dire de A à d(f).

Théorème* [13]

- Si E est un ensemble de relations fonctionnelles *sans circuit*, associé à la relation R,
- alors toutes les *relations fonctionnelles de la couverture minimale* CM sont *directes*.

En d'autres termes, nous voulons démontrer que si $Ra = (A_1, A_2 \rightarrow C)$ appartient à la couverture minimale CM, alors il n'existe pas dans la fermeture CM de relations fonctionnelles Rb et Rc telles que :

$Rb = (A_1 \rightarrow X)$
$Rc = (A_2, X \rightarrow C)$ avec $X \wedge A_2 = \emptyset$

---

*Ce résultat a été obtenu avec l'aide de D. Casey (IBM Research, San José), P. Bernstein (University of Toronto) et C. Delobel.

Pour démontrer ce théorème, nous allons supposer que la proposition contraire est vérifiée et nous allons montrer qu'elle conduit soit à l'existence de circuits, soit à la non-appartenance de Ra à la couverture minimale.

Nous avons à examiner quatre cas selon que :

(1) - Rb $\in$ CM et Rc $\in$ CM
(2) - Rb $\notin$ CM et Rc $\in$ CM
(3) - Rb $\in$ CM et Rc $\notin$ CM
(4) - Rb $\notin$ CM et Rc $\notin$ CM

Cas (1) :

Comme Ra = Rb +> Rc, les trois relations fonctionnelles Ra, Rb, Rc ne peuvent appartenir à la même couverture minimale ; il faut donc rejeter ce cas.

Cas (2) :

Si Rb $\notin$ CM, il existe donc une f-expression telle que :

- Rb = $f(R_1 .Ri. R_n)$
- $\forall i$ $i \in [1,n]$ $\quad$ Ri $\in$ CM

puisque par définition on peut générer la fermeture FM à partir de CM.

De deux choses l'une :

- ou bien $\nexists i \in [1,n]$ tel que Ra = Ri,
  et alors Ra = $f(R_1,R_2...R_n)$ +> Rc, Rc $\in$ CM.
  Comme dans le cas (1), Ra, R_1, R_2 ... R_n, R_c ne peuvent pas appartenir à la même couverture minimale ; ce cas est donc à rejeter.
- ou bien $\exists i \in [1,n]$ tel que Ra = Ri.
  Ra = $f(R_1,R_2,. Ra.,R_n)$ +> Rc ;
  soit G le graphe associé à FM ; d'après le lemme précédent, il existe alors un chemin de G conduisant de C, partie droite de Ra à X, partie droite de Rb. Comme Rc = (A_2, X → C), il existe également un chemin conduisant de X vers C.

Il existe donc un circuit dans G, ce qui est contraire à l'hypothèse.

Les cas 3 et 4 conduisent aux mêmes conclusions par les mêmes raisonnements.

Ainsi, pour une relation R dont l'ensemble E de relations fonctionnelles est sans circuit, nous venons de montrer l'égalité de la couverture minimale CM et de l'ensemble des relations fonctionnelles directes.

La conséquence d'une telle proposition est que toutes les *sous-relations de R* formées à partir d'une relation fonctionnelle de *CM sont directes.*

3.3.4. Décomposition particulière d'une relation

. D'après [14] nous savons que si I désigne l'index de la relation R et que si $(R_1,R_2,\ldots,R_n)$ représente l'ensemble complet des sous-relations directes de R, alors R peut se décomposer de la manière suivante :

(D) - $R = R_1 * R_2 \ldots * R_n$ s'il existe Ri telle que sa partie gauche contient I

- ou sinon $R = I * R_1 * R_2 \ldots * R_n$.

. Du fait de l'unicité de l'ensemble complet des sous-relations directes de R, et de l'unicité de l'index, une telle décomposition D est *unique* et est appelée *la* décomposition de R.

. L'ensemble des relations composant cette décomposition est appelé *ensemble minimal irredondant.*

Remarque : Dans le cas où R est en deuxième forme ou en troisième forme, l'index I est obligatoirement contenu dans une des sous-relations directes Ri.

Exemple 3.14

Soit la relation R_{12} dont la couverture minimale est donnée dans l'exemple 3.12, page III.18). Son index (A,D) n'est contenu dans aucun monôme de la couverture minimale.

R_{12} se décompose en : $R_{12} = [A,D] * [A,B] * [B,C] * [C,D,E]$.

Exemple 3.15

Soit la relation R_{16} avec l'ensemble des relations fonctionnelles

$$\begin{cases} A,B \rightarrow C \\ C \rightarrow D \\ A,B \rightarrow E \end{cases}$$

qui forment la couverture minimale ;
l'index de R_{16} est (A,B) ;
R_{16} se décompose en R_{16} = [A,B,C] * [C,D] * [A,B,E].

3.3.5. Intérêt d'une décomposition d'une relation en sous-relations directes

Tous ces résultats théoriques nous ont permis de mettre en évidence la décomposition D d'une relation R en sous-relations directes.

Nous allons montrer comment cette décomposition permet d'éviter les imperfections inter-relations qui risqueraient de provenir de la représentation de R par composition.

- Soit, dans le cas général, la relation R représentée par composition à l'aide de : $R = S_1 * S_2 \ldots * S_p$ où tous les S_p sont des sous-relations en troisième forme.

Les risques d'imperfections inter-relations que contient une telle représentation proviennent d'après le § 3.1.a, des mises à jour d'une sous-relation S_i de R qui provoquent la non vérification de toutes les règles d'intégrité de R, bien qu'elles vérifient les règles des S_i.

En effet, dans le cas général, il ne suffit pas d'assurer la cohérence de toutes les règles d'intégrité liées aux S_i pour assurer la cohérence globale de toutes les règles d'intégrité de R ; les exemples 3.3 et 3.4 du § 3.1 illustrent cette proposition.

- Par contre, dans le cas particulier où R est représentée par composition à l'aide de la décomposition D (§ 3.3.4) :

$$R = R_1 * R_2 \ldots * R_n \ (*I),$$

à chaque sous-relation Ri est associée une règle d'intégrité, et réciproquement (Théorème du § 3.3.3).

Donc si la mise à jour de l'une ou de plusieurs sous-relations Ri est cohérente par rapport aux seules règles d'intégrité des sous-relations concernées, elle est cohérente par rapport à l'ensemble des règles de R d'après la définition même de la couverture minimale (§ 3.3.2).

Les risques d'imperfection inter-relations qui proviennent de la représentation de R dans la structure sont donc éliminés si R est représenté par composition à l'aide de D.

Exemple 3.16

C'est ainsi que dans l'exemple 3.3, la décomposition de R_3 n'étant pas formée seulement de sous-relations directes, est source d'incohérence, ce qui n'est pas le cas de la décomposition de R_3 en sous-relations directes :

$$R_3 = [VOL, AVION]R_3 * [AVION, PILOTE]R_3 ;$$

en reprenant la même réalisation, on la décompose suivant :

R_{31}

VOL	AVION
v_1	a_1
v_2	a_1
v_3	a_2
v_4	a_3

R_{33}

AVION	PILOTE
a_1	p_1
a_2	p_2
a_3	p_2

Le remplacement de l'entité (<VOL : v_4>, <AVION : a_3>) en (<VOL : v_4>, <AVION : a_1>) ne provoque aucune incohérence, même au niveau de R_3.

De même dans l'exemple 3.4, la relation fonctionnelle AUTEUR → EDITEUR appartient à la couverture minimale mais n'apparaît pas dans la décomposition proposée ; aussi est-elle à l'origine des incohérences signalées. Voici la décomposition en sous-relations directes de R_4 :

$$R_4 = [\text{LIVRE}, \text{AUTEUR}]R_4 * [\text{LIVRE}, \text{EDITEUR}]R_4 * [\text{AUTEUR}, \text{EDITEUR}]$$

- Une telle représentation élimine également les imperfections intra relations, puisque les sous-relations sont en troisième forme.
Nous avons ainsi montré comment représenter une relation en première ou deuxième forme.

3.4. CONSTRUCTION D'UN ALGORITHME DE DECOMPOSITION D'UNE RELATION EN SOUS-RELATIONS DIRECTES

Etant donné une relation R et un ensemble E de sous-relations fonctionnelles de R *sans circuit*, un tel algorithme consiste, d'après les résultats du § 3.3, à trouver la couverture minimale CM de E et l'index de R.

Trouver l'index est trivial. Aussi ne nous intéressons-nous dans ce paragraphe qu'à la construction d'un algorithme de couverture minimale.

. Plusieurs travaux ont déjà été réalisés dans ce domaine. Delobel [14] a donné un premier algorithme théorique d'obtention de la couverture minimale, repris par Wang et Wedekind [26].

Portal [24] a construit plusieurs algorithmes de fermeture fondés soit sur les principes de double dualisation, soit sur ceux du tour de consensus. Delobel et Portal [16] ont donné des comparaisons d'efficacité de ces programmes. Bernstein [6] a fourni un algorithme de couverture minimale construit sur des bases opérationnelles : en résumé, il teste, les unes après les autres, toutes les relations fonctionnelles de l'ensemble de départ pour savoir si elles peuvent être générées par les autres ; si oui, elles n'appartiennent pas à la couverture minimale, si non, elles y appartiennent.

. L'algorithme que nous avons mis au point est un algorithme qui construit en même temps la fermeture et la couverture minimale d'un ensemble de relations fonctionnelles dépourvu de circuit ; la fermeture est en effet très utile dans le domaine de l'interrogation d'une base de données.

Ainsi l'algorithme que nous proposons est composé de deux processus qui agissent simultanément :

- le processus de fermeture qui s'inspire de l'algorithme de fermeture à tour de consensus et à relations fonctionnelles préalablement triées ([16])
- le processus de couverture minimale qui, pour chaque nouvelle relation fonctionnelle générée par le processus de fermeture, décide de son appartenance à la couverture minimale. Ce processus

suppose initialement que toutes les relations fonctionnelles de l'ensemble de départ sont susceptibles d'appartenir à la couverture minimale. Son principe de fonctionnement repose sur le théorème du § 3.3.3.

Nous donnons dans l'annexe A tous les résultats qui nous ont permis de construire cet algorithme et la description détaillée de son fonctionnement.

Exemple 3.17

Nous reprenons l'exemple de la base de données de réservation de chambres d'hôtel dans des stations de ski. Nous avons mis en évidence dans le § 2.5 (page II.18) les relations fonctionnelles suivantes :

$f_1 = (s \rightarrow s_1)$

$f_2 = (h,s \rightarrow h_1)$

$f_3 = (h,s,p \rightarrow p_1)$

$f_4 = (c,h,s,p \rightarrow r_1)$

$f_5 = (c,p \rightarrow d_1)$

$f_6 = (c \rightarrow c_1)$

$f_7 = (c,p \rightarrow h)$

$f_8 = (c,p \rightarrow s)$

Dans cet ensemble

$f_4 = (c,h,p,s \rightarrow r_1)$ est absorbée par $f'_4 = (c,p \rightarrow r_1)$ obtenue par $(c,p \rightarrow s) \text{ +> } ((c,p \rightarrow h) \text{ +> } (c,h,p,s \rightarrow r_1))$

$= (c,p \rightarrow s) \text{ +> } (c,p,s \rightarrow r_1)$

$= (c,p \rightarrow r_1)$.

La couverture minimale est alors :

$s \rightarrow s_1$

$h,s \rightarrow h_1$

$h,s,p \rightarrow p_1$

$c,p \rightarrow s$

$c,p \rightarrow h$
$c,p \rightarrow d_1$
$c,p \rightarrow r_1$
$c \rightarrow c_1$

3.5. CONCLUSIONS

Toute notre étude a eu pour but d'éliminer les imperfections d'une structure d'une base de données, et de concevoir des structures dépourvues de telles imperfections. Considérant ce problème à partir d'un ensemble de relations associé à la base de données, nous avons alors montré le rôle déterminant du mode de représentation des relations dans la nouvelle structure et nous avons mis au point un algorithme qui permet de trouver pour toute relation sa représentation cohérente dans la structure (l'ensemble des relations fonctionnelles associé étant sans circuit).
C'est ainsi que nous pensons avoir répondu en partie à la question : comment concevoir une structure ne contenant aucune imperfection de fonctionnement ?

De plus, à cause de la double signification des relations fonctionnelles qui peuvent être considérées à la fois comme des relations et des règles d'intégrité, le même algorithme peut servir également à trouver, à partir d'un ensemble de règles d'intégrité, l'ensemble minimal dont il faut et il suffit que la cohérence soit vérifiée pour que celle de l'ensemble le soit. Nous pouvons ainsi éliminer les règles d'intégrité redondantes.

Enfin, les résultats de ce chapitre permettent de fournir les questions que doit remplir un ensemble de relations pour être compatible avec une structure (question K de l'Introduction). Il suffit, en effet, que la fermeture de cet ensemble et celle des relations traduites de la structure soient identiques.

C H A P I T R E 4

RESTRUCTURATION D'UNE BASE DE DONNEES

S O M M A I R E

INTRODUCTION

. Comme nous l'avons rappelé dans l'introduction, l'importance de l'étude de la restructuration vient de l'évolution constante et des besoins des utilisateurs et des règles du champ d'application de la base. Ces évolutions interdisent donc à une base d'avoir une structure figée.

Savoir comment effectuer une restructuration fait partie également de l'étude de la conception d'une base. Tel est le cadre de ce chapitre.

. Nous avons déjà donné des éléments de réponse à la première question dans le chapitre 3, où en effet nous montrons comment il est possible de restructurer une base pour faire de telles éliminations. Aussi les conclusions du chapitre 3 sont-elles également des éléments de restructuration.

. Fry et Jeris [18], puis Navathe et Fry [22] ont abordé le problème de la restructuration dans le cadre du développement d'une méthodologie et d'un système de transposition effective des données d'une base, à partir d'une organisation initiale vers une organisation finale.

Ils ont ainsi mis en évidence différents "niveaux abstraits" de restructuration" [22] et différentes opérations élémentaires de restructuration.

Nous allons tout d'abord reprendre une partie de leur approche.

. Nous reposons ensuite le problème de la restructuration dans le contexte du champ d'application de la base.

Les principaux motifs qui peuvent conduire à une restructuration peuvent se résumer en trois points (Introduction) :

- introduction de nouveaux constituants
- modification des règles d'intégrité
- amélioration de l'efficacité de l'utilisation.

Face à de telles situations, quelles sont les opérations de restructuration que l'on peut entreprendre sur la base ?
C'est leur recherche qui constitue notre objectif et notre apport à l'étude de Navathe et Fry.

Comme dans leur étude, nous nous sommes restreints au cas de structures arborescentes.

. Dans le § 4.2, nous proposons en premier lieu une formalisation d'un certain type de restructuration ; celle-ci repose sur la modélisation mathématique des structures arborescentes, introduite dans [14] par la théorie des catalogues [25]. Nous rappelons cette modélisation en reprenant la présentation de [14].

Ensuite (§ 4.3), à l'aide des propriétés mathématiques que nous avons développées à partir de ce modèle (Annexe B), nous présentons différentes possibilités d'opération de restructuration pour une base à structure arborescente et compte tenu de son environnement.

. Nous pensons par cette approche répondre aux questions suivantes liées à la restructuration (cf. Introduction) :

- en vue de rendre l'utilisation de la base plus efficace, comment la restructurer :
  - . pour éliminer les incohérences dues à la structure elle-même (D) ?
  - . pour éliminer les relations redondantes qu'elle contient et qui demandent des mécanismes trop lourds pour le maintien de leur cohérence (E) ?
- quand peut-on dire que deux structures sont équivalentes (H) ?
- comment peut-on restructurer une base de données pour tenir compte de l'évolution des règles d'intégrité (F) ?

Dans ce chapitre, nous allons utiliser le mot **branche** *pour désigner un chemin dans un arbre, dont une extrémité est la* **racine**, *chemin étant pris dans le sens défini par BERGE dans Graphes et Hypergraphes (p. 8) de chez DUNOD.*

4.1. "TROIS NIVEAUX D'ABSTRACTION DE LA RESTRUCTURATION D'UNE BASE DE DONNEES" [22]

4.1.1. Notion de restructuration

. Navathe et Fry [22] définissent ainsi la restructuration des données d'une base.
Etant donné :

(a) une structure initiale SS et une de ses réalisations I_s

(b) une structure finale TS et une de ses réalisations I_t

(c) un ensemble de spécifications de transformations de SS en TS,

restructurer consiste à générer un ensemble de réalisations I_t compatibles avec la structure finale TS suivant les conditions :

(a) I_t doit découler de I_s

(b) toutes les réalisations de I_s doivent être utilisées dans le processus de transformation

(c) les spécifications de transformation doivent être respectées.

. Trois niveaux de restructuration sont ensuite présentées :

- *niveau du schéma* : c'est-à-dire de tout ce qui concerne l'organisation des données. La notion de schéma renferme aussi bien les notions de structure que celles de constituants, groupes*, liaisons entre groupes, noms de constituants, de groupes de liaison.

- *niveau des réalisations* : celui-ci est directement lié aux données qu'il faut réarranger compte tenu des modifications du schéma. Lors d'une restructuration, les opérations à entreprendre à ce niveau découlent de celles requises au niveau précédent.

- *niveau des valeurs des constituants* : les opérations à ce niveau sont du type duplication, création d'une valeur indéterminée, création de valeurs pour discriminer plusieurs groupes, etc.

---

*dans le sens du chapitre 2.

.2. Opérations au niveau du schéma

Navathe et Fry distinguent trois types fondamentaux de modification du schéma :

- renomination* d'éléments du schéma initial.
- combinaison d'éléments du schéma initial en éléments du schéma final.
  Ces combinaisons sont essentiellement la compression et l'expansion.
- introduction de nouvelles relations dans le schéma final.

Ces trois types comprennent les opérations suivantes sur les schémas :

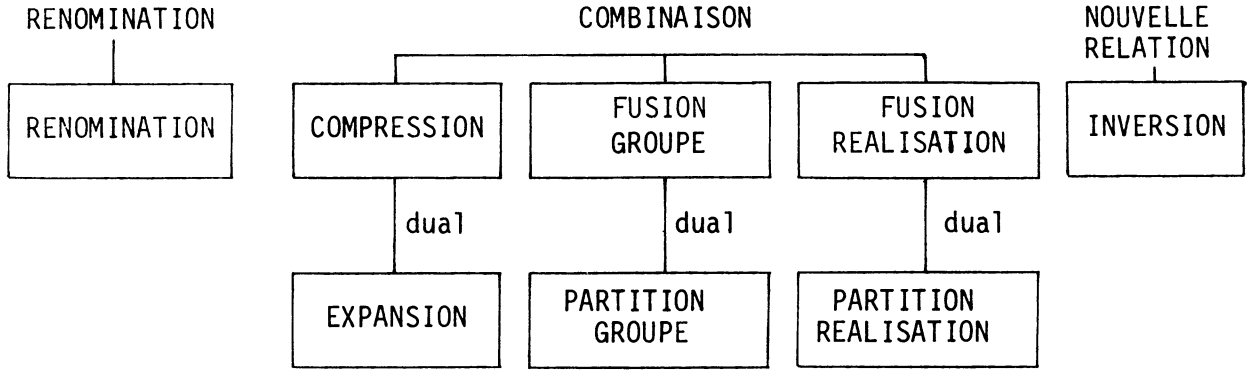


Fig. 4.1

Chacune des opérations sont définies comme ci-dessous. Leurs effets au niveau des réalisations sont expliqués par des exemples au paragraphe 4.1.3.

Renomination : il s'agit de renommer soit des groupes, soit des liaisons de la structure.

---

*_Renomination, combinaison, compression, expansion, fusion, partition, inversion sont respectivement nos traductions de : renaming, combining, compression, expansion, merging, partitionning, inversion._

Compression : de deux ou plusieurs groupes $G_1, G_2 \ldots G_i$. Cette opération n'est définie que pour des groupes formant une partie d'une branche de la structure arborescente initiale. Elle consiste à fusionner tous ces groupes en un seul (cf. sur la fig. 4.2, A-B, B-C, B-D, A-B-C, A-B-D ..., mais non A-C, A-B-F, C-D, D-G ...).

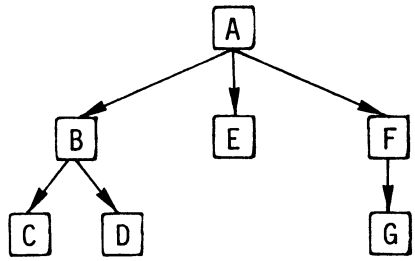


Fig. 4.2

Expansion : elle est définie à partir d'un seul groupe de la structure arborescente initiale. Elle consiste à créer un ou plusieurs groupes à partir d'un groupe donné. Les nouveaux groupes créés sont hiérarchiquement dépendants chacun l'un de l'autre et appartiennent à une même branche de la structure arborescente finale. C'est l'opération duale de l'opération de compression.

Fusion de groupes : elle est définie pour deux ou plusieurs groupes qui ont le même groupe supérieur (ainsi C-D, B-E-F, B-E ...). Elle consiste à fusionner ces groupes en un seul groupe G de la structure finale. Cette fusion s'opère par une condition de réunion des réalisations et par l'introduction d'un constituant dans G indiquant les provenances des réalisations.

Partition de groupes : un seul groupe de la structure initiale participe à cette opération. Elle consiste, suivant un ensemble de critères, à partitionner l'ensemble des réalisations de ce groupe, et à affecter à chaque partition un groupe de la structure finale. C'est l'opération duale de la précédente.

Inversion : elle est définie pour deux groupes appartenant à la même branche (cf. A-B, A-C, A-D, B-D, B-C, A-E, A-F, A-G, F-G), et dont les positions sont échangées dans l'arborescence. Cette opération n'est définie que dans ce cas car sinon elle pourrait conduire à des structures non arborescentes.

4.1.3. Opérations au niveau des réalisations : exemple 4.1

Nous allons rapidement illustrer les opérations de restructuration à l'aide d'un exemple [18]. Ces opérations sont des conséquences de celles effectuées au niveau du schéma.

Soit une base de données d'usine de fabrication divisée en DEPARTEMENTS ; les relations entre DEPARTEMENTS, EMPLOYE, PRODUIT, TRAITEMENT, PIECES UTILISEES sont modélisées hiérarchiquement de la manière suivante :

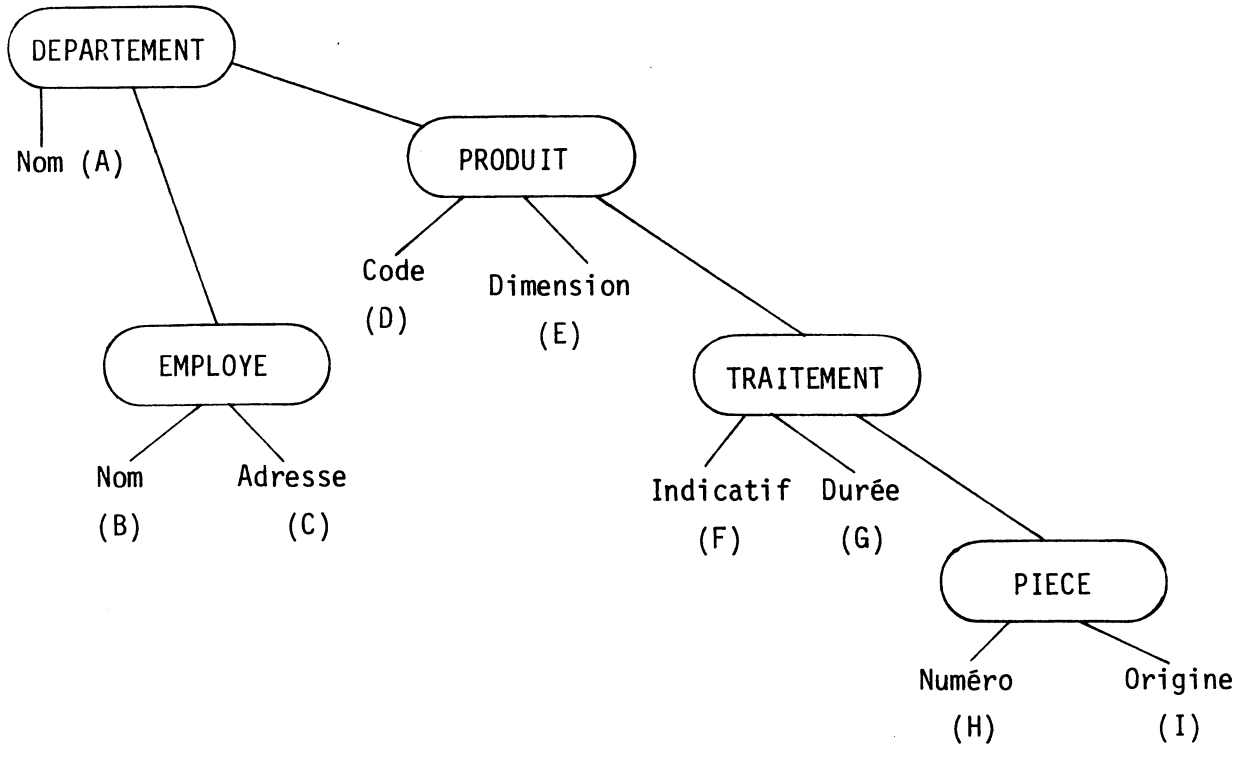


Fig. 4.3 - Structure 1

Chaque DEPARTEMENT est déterminé par son Nom, chaque EMPLOYE par son Nom, chaque PRODUIT par son Code, chaque TRAITEMENT par son Indicatif, chaque PIECE-UTILISEE par son Numéro.

Chaque DEPARTEMENT emploie un certain nombre d'EMPLOYES et développe un certain nombre de PRODUITS. Ceux-ci sont fabriqués par des TRAITEMENTS qui se servent de PIECES UTILISEES. De plus chaque EMPLOYE d'un DEPARTEMENT travaille sur tous les PRODUITS développés dans celui-ci. La modélisation représentée par la figure 4.3 correspond à une certaine vue de la base de données qui privilégie certains chemins d'accès : ainsi on n'atteint les TRAITEMENTS effectués dans un DEPARTEMENT que par l'intermédiaire des PRODUITS.

- Compression

La transformation de la structure 1 dans la structure 2 (fig. 4.4) s'effectue par des opérations de compression sur les réalisations de PRODUIT, TRAITEMENT, PIECE. Ceci est possible puisque ces groupes forment bien une partie d'une branche.

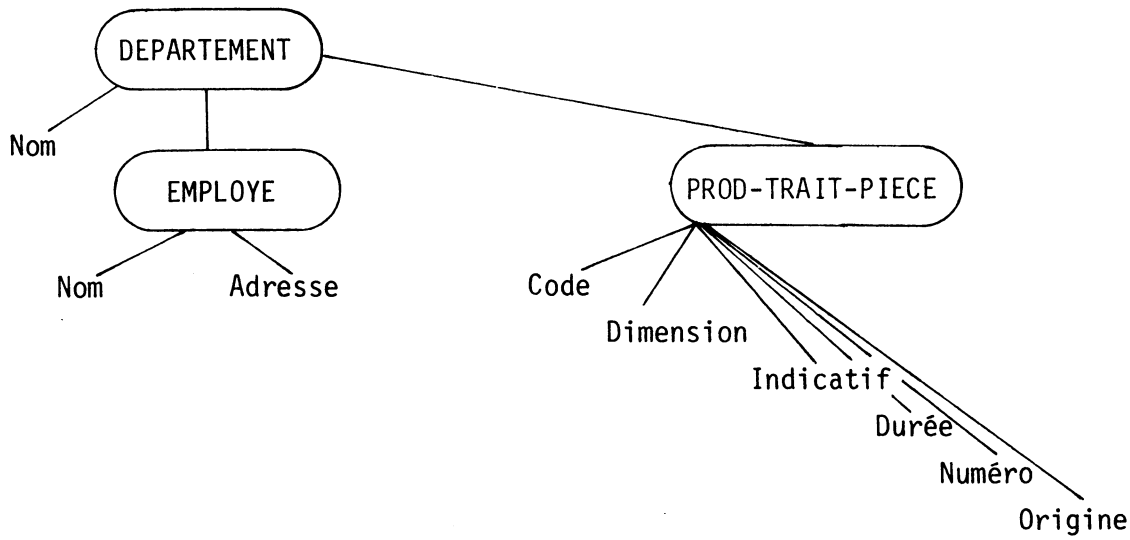


Fig. 4.4 - Structure 2

La transformation inverse de la structure 2 dans la structure 1 fait appel à des opérations d'expansion.

- Fusion de groupes

Une telle transformation peut être illustrée par l'exemple suivant :

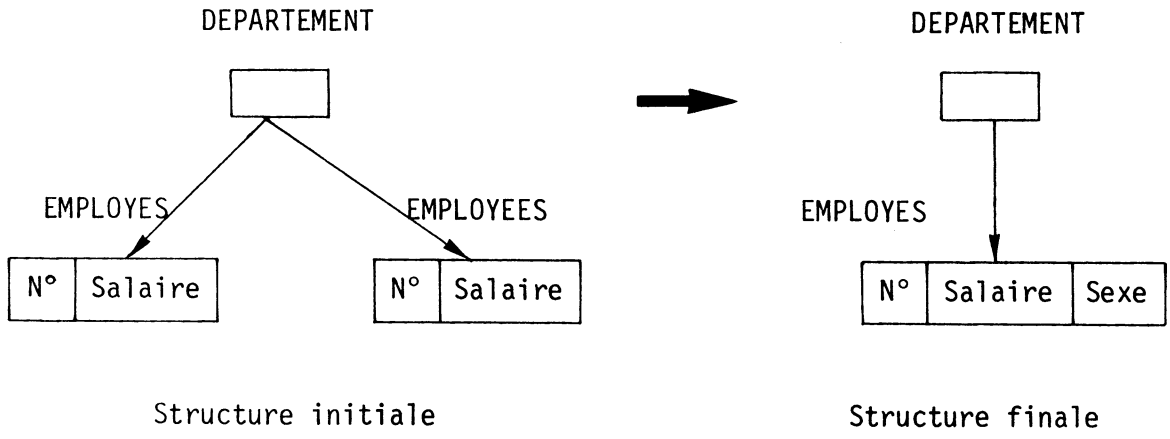


Fig. 4.5

Les deux groupes EMPLOYES et EMPLOYEES de la structure initiale sont fusionnés en un seul groupe ; cette opération est possible car les deux groupes initiaux ont le même groupe supérieur.

La transformation inverse correspond à des opérations sur les réalisations de type partition de groupes.

- Fusion de réalisations

L'exemple illustrant cette transformation ne se trouve pas dans [18] : c'est nous qui l'avons construit.
La transformation de la structure 1 dans la structure 3 s'opère par des opérations de fusion de réalisations de EMPLOYE et de PRODUIT, à l'aide de la composition normale suivante :

$$[G1, G2, G3] = [G1, G2] * [G1, G3]$$

avec G1, ensemble des constituants du groupe DEPARTEMENT
G2, ensemble des constituants du groupe EMPLOYE
G3, ensemble des constituants du groupe PRODUIT.

Cette transformation conduit à la structure suivante :

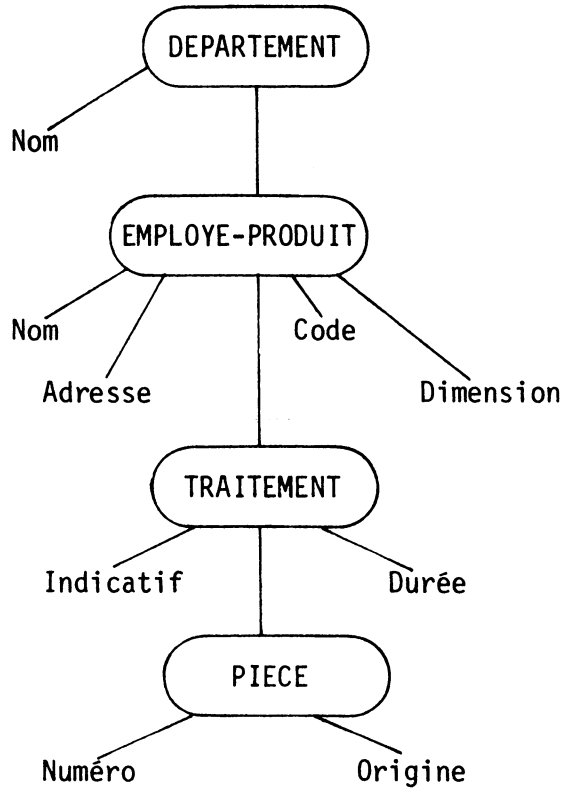


Fig. 4.6 - Structure 3

La transformation inverse de la structure 3 dans la structure 1 fait appel à des opérations de partition de réalisations.

- Inversion

La transformation de la structure 1 dans la structure 4 s'opère par des opérations d'inversion sur les réalisations de PIECE et de PRODUIT.
Une telle transformation est possible car PIECE et PRODUIT appartiennent à la même branche.

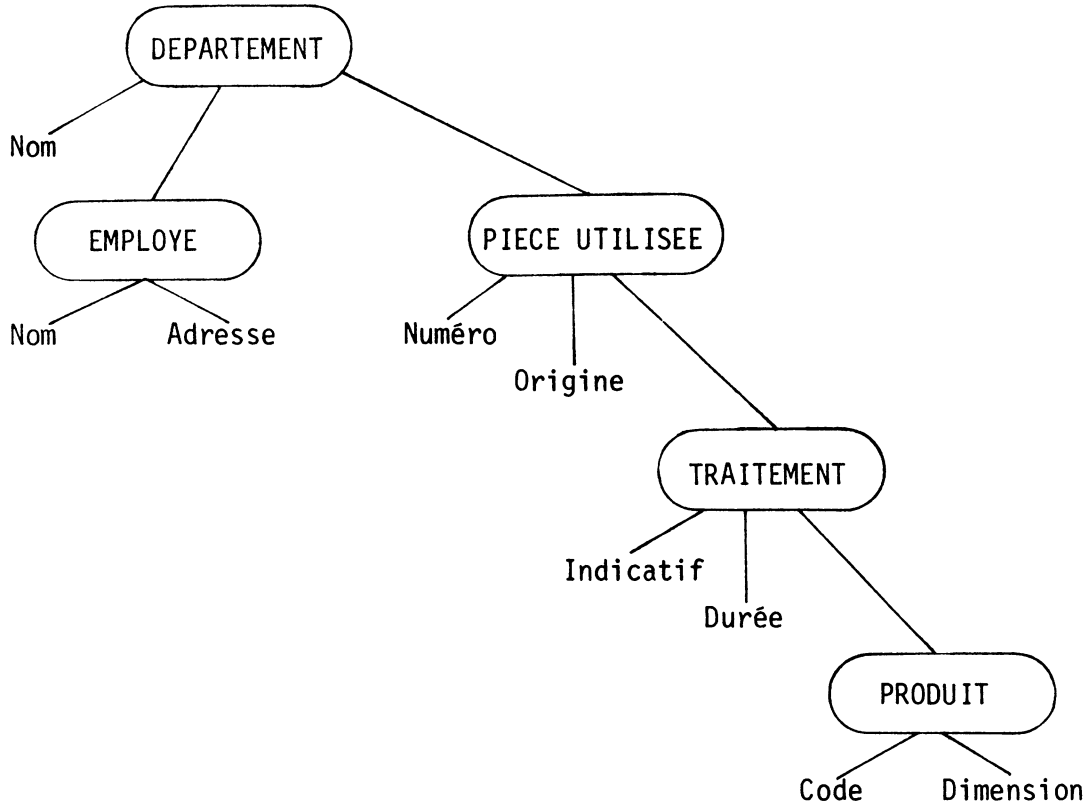


Fig. 4.7 - Structure 4

4.1.4. Conclusions

Navathe et Fry [22] ont classé les opérations déclenchées par une restructuration suivant les niveaux de la base de données qu'elles atteignent :

- niveau du schéma
- niveau des réalisations, c'est-à-dire de l'organisation des données
- niveau des valeurs que peuvent prendre ces données.

Ils ont ensuite donné une liste d'opérations possibles à chaque niveau, en remarquant que celles du second niveau sont étroitement liées à celles du premier, tant par leur définition que par l'exécution d'une restructuration.

Comme seuls ces deux niveaux nous intéressent, nous rappelons brièvement les opérations mises en évidence :

- compression et expansion
- fusion de groupes et partition de groupes
- fusion de réalisations et partition de réalisations
- inversion.

Nous allons maintenant nous attacher à replacer ces opérations dans le contexte du champ d'application pour tâcher d'en justifier l'emploi.

4.2. FORMALISATION DU PROCESSUS DE RESTRUCTURATION

Nous voulons maintenant étudier les conditions dans lesquelles de telles opérations de restructuration sont justifiées. Aussi devons-nous tenir compte des règles d'intégrité qui représentent les règles du champ d'application de la base (cf. § 3.1.b).

Nous allons dans ce but formaliser le processus de restructuration après avoir rappelé la correspondance entre une structure arborescente S et une décomposition arborescente D d'une relation R.

Note : Nous restreignons notre étude aux seules opérations de restructuration du premier et du second niveau [22].

4.2.1. Correspondance entre une structure arborescente et une décomposition arborescente

Nous rappelons dans ce paragraphe les résultats acquis dans [14].

4.2.1.1. Structure arborescente et catalogue

Les structures arborescentes sont par nature identiques à la notion de catalogue dont la définition est donnée dans [25] et que nous rappelons brièvement ici :

- Une *arborescence** est un ensemble ordonné fini tels que deux éléments quelconques ont un minorant commun mais pas de majorant commun.

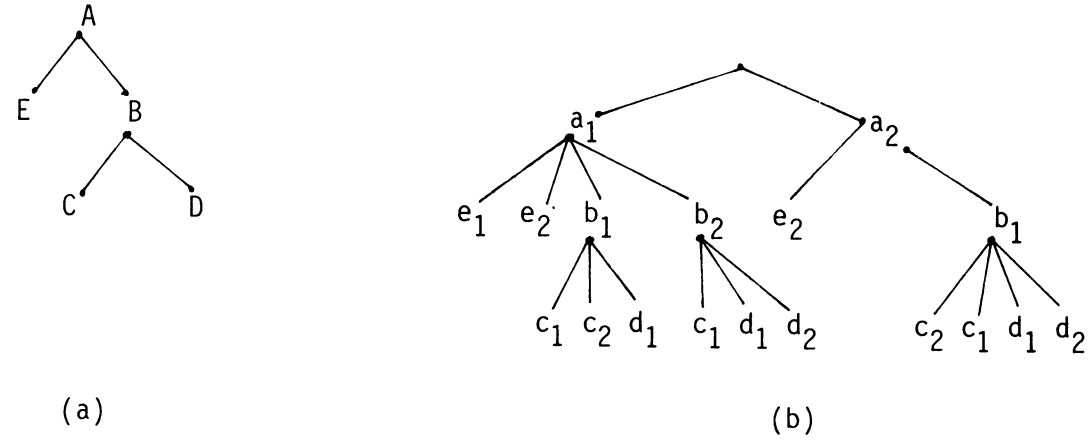
On distinguera l'ensemble X des éléments d'une arborescence Y de l'arborescence elle-même, qui est la donnée de X et d'une relation d'ordre H qui en fait une arborescence. On notera Y = (X,H).

- Une *forêt* est une union d'arborescences disjointes.

Le terme "d'arborescence" qui va être utilisé dans ce paragraphe ne doit pas être confondu avec celui de "décomposition arborescente". Toutefois les deux notions sont très proches l'une de l'autre.

Exemple 4.2

La figure 4.8 (a) représente la carte arborescente du catalogue 4.8 (b).



(a) (b)

Fig. 4.8 - Catalogue associé à une carte arborescente

- Une *pseudo-arborescence* est une arborescence valuée, c'est-à-dire qu'à chaque élément de l'arborescence on associe une valeur. Dans les pseudo-arborescences dont il sera question, les éléments distincts comparables porteront des valeurs distinctes. En conséquence, la notion de *chaîne* d'un ensemble ordonné peut s'appliquer à des pseudo-arborescences.

- Un *homomorphisme d'ordre strict de première catégorie* par rapport à la première variable est une application ψ d'un ensemble ordonné E dans un ensemble ordonné F telle que :

(1) $a < b \Longrightarrow \psi(a) < \psi(b) \quad a,b \in E$

(2) $\psi(a) < \psi(b) \Longrightarrow \forall a_1 \in \psi^{-1}\psi(a) \ \exists b_1 \in \psi^{-1}(b)$ tel que $a_1 < b_1$

pour un homomorphisme par rapport à la deuxième variable (2) est à remplacer par (3)

(3) $\psi(a) < \psi(b) \Longrightarrow \forall b_1 \in \psi^{-1}\psi(b) \ \exists a_1 \in \psi^{-1}(a)$ tel que $a_1 < b_1$

On appellera carte *arborescente du catalogue* la donnée de l'ensemble $\mathcal{A}$ des constituants ordonnés par la relation H, que nous noterons $(\mathcal{A},H)$.

Une forêt F est dite un *précatalogue* sur la carte arborescente $(\mathcal{A},H)$ s'il existe une application surjective de F sur $(\mathcal{A},H)$ qui soit un homomorphisme d'ordre strict de première catégorie successivement par rapport aux deux variables.

Valuons les éléments du précatalogue de sorte que

- si $A \in \mathcal{A}$ les éléments de $\psi^{-1}(A)$ sont valués avec des éléments appartenant à l'ensemble A
- deux éléments n'ont même valuation que s'ils appartiennent au même $\psi^{-1}(A)$ et s'ils ne couvrent pas un même élément.

La pseudo-arborescence obtenue en ajoutant alors une racine est un *catalogue* K.

4.2.1.2. Catalogue et décomposition

A tout catalogue on peut associer une collection de données, en considérant que chaque chaîne du catalogue est représentative d'une entité d'une relation.

C'est ainsi qu'à partir du catalogue de l'exemple 4.2 on peut construire les 3 relations correspondant aux branches de la carte arborescente :

A	E
a_1	e_1
a_1	e_2
a_2	e_2

A	B	C
a_1	b_1	c_1
a_1	b_1	c_2
a_1	b_2	c_1
a_2	b_1	c_2
a_2	b_1	c_1

A	B	D
a_1	b_1	d_1
a_1	b_2	d_1
a_1	b_2	d_2
a_2	b_1	d_1
a_2	b_1	d_2

Ces trois relations par composition donnent la collection [A,B,C,D,E] :
[A,B,C,D,E] = [A,E] * ([A,B,C] * [A,B,D]).

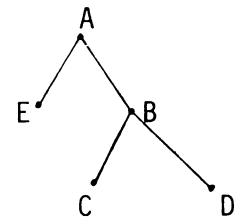
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_1	b_1	c_2	d_1	e_1
a_1	b_2	c_1	d_1	e_1
a_1	b_2	c_1	d_2	e_1
a_2	b_1	c_2	d_1	e_2
a_2	b_1	c_2	d_2	e_2
a_2	b_1	c_1	d_1	e_2
a_2	b_1	c_1	d_2	e_2
a_1	b_1	c_1	d_1	e_2
a_1	b_1	c_2	d_1	e_2
a_1	b_2	c_1	d_1	e_2
a_1	b_2	c_1	d_2	e_2

Inversement si la relation construire sur [A, B, C, D, E] vérifie la décomposition arborescente

[A,B,C,D,E] = [A,E] * [A,B,C,D]

et

[A,B,C,D] = [A,B,C] * [A,B,D]



alors elle peut être représentée par un catalogue dont la carte arborescente est celle de la figure 4.8 (a).

4.2.1.3. Définition d'un schéma d'une base de données à structure arborescente

Ainsi par l'intermédiaire de leur catalogue K, on associe à une structure arborescente S une décomposition arborescente D d'une relation R.

Cette propriété nous permet de définir le schéma d'une base de données B à *structure arborescente* comme un *triplet* (R,I,D) formé de :

- R : une relation définie sur les constituants qui composent la structure
- I : un ensemble de règles d'intégrité liées à la structure. Elles expriment des propriétés sémantiques soit sous forme de relations fonctionnelles, soit sous forme de décompositions de sous-relations de R
- D : la décomposition arborescente de R qui a même catalogue que S.

Nous noterons B(R,I,D).

Remarque : D est un invariant de B et est donc contenue dans I ; mais nous tenons à la mentionner explicitement car à D correspond la structure de la base de données : c'est le seul invariant de I qui a cette propriété. D contient donc les relations traduites dans la structure (cf. § 2.4).

Exemple 4.3

Reprenons l'exemple 4.1 (page IV.6) ; on peut définir le schéma de la base de données B par :

- la relation R[A,B,C,E,F,G,H,I]
- l'ensemble des règles d'intégrité I :
  I = {B → C ; E → F ; G → H ; J → K ;
  et la décomposition arborescente avec racines secondaires D de R

$$D \begin{cases} [A,B,C,E,F,G,H,I] = [A,B,C] * [A,E,F,G,H,I] \\ [A,E,F,G,H,I] = [A,E,F] * [A,E,G,H,I] \\ [A,E,G,H,I] = [A,E,G,H] * [A,E,G,I] \} \end{cases}$$

- la décomposition D de R qui a même catalogue que la structure S de B.

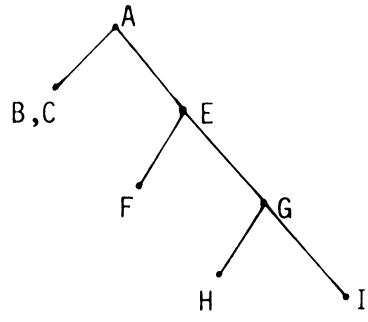


Fig. 4.9

4.2.1.4. Rapprochement des notions précédentes de celles de Navathe et Fry [22]

. Schéma d'une base de données de structure S : (R,I,D)

Cette notion recouvre dans [22] les groupes, les liaisons entre groupes d'une base de données, les constituants des groupes. Dans le chapitre 2, nous avons montré comment il est possible de transformer une structure arborescente S en un arbre. Celui-ci est construit à partir des groupes, des liaisons entre ces groupes, de leurs constituants et correspond à la carte arborescente de la décomposition arborescente D.

Ainsi notre définition de schéma contient en plus de celle de [22] les règles d'intégrité associées à la structure. Cette différence provient de notre approche de la restructuration qui s'intéresse à la cohérence des opérations de restructuration par rapport aux règles d'intégrité de la base.

. A la structure S correspondent une carte arborescente et un catalogue. Les opérations de restructuration au niveau du schéma dans [22] correspondent à des opérations sur D et donc sur la carte arborescente, alors que celles au niveau des réalisations s'effectuent sur le catalogue.

A cause de l'homomorphisme d'ordre strict de première catégorie par rapport à deux variables, du catalogue sur sa carte arborescente, toutes les opérations de restructuration au niveau de la carte arborescente (= niveau du schéma) se répercutent en opérations au niveau du catalogue (= niveau des réalisations). Cette conclusion rejoint la constatation de Navathe et Fry [22] : "les modifications au niveau du schéma ont des répercussions directes au niveau des réalisations".

Nous en déduisons que la seule étude au niveau de la carte arborescente nous permettra de définir des opérations de restructuration des deux niveaux, à savoir carte arborescente (schéma) et catalogue (réalisation).

4.2.2. Formalisation du processus de restructuration

Nous définissons la *transformation d'un schéma* d'une base de données $B_0(R_0,I_0,D_0)$ dans un autre schéma $B_1(R_1,I_1,D_1)$ comme étant successivement :

- la transformation de R_0 en R_1 par la modification éventuelle de l'espace de constituants de B_0
- la transformation de I_0 en I_1 par la modification des règles d'intégrité de B_0
- l'obtention d'une décomposition arborescente D_1 de R_1, compte tenu des invariants I_1.

Cette *définition* contient celle de la *restructuration* qui se limite en effet à la seule transformation de D_0 en D_1.

Elle tient compte des trois motifs principaux de restructuration et ainsi doit permettre de l'étudier compte tenu des règles d'intégrité.

Elle conduit aux définitions de restructuration cohérente, de structures équivalentes, de restructuration réversible.

Pour ces trois définitions, nous supposons que :

$B_0(R_0,I_0,D_0)$ désigne le schéma initial de la base correspondant à la structure S_0,

$B_1(R_1,I_1,D_1)$, le schéma final correspondant à la structure S_1,

T la transformation de B_0 en B_1 :

$$B_0(R_0,I_0,D_0) \underset{T^{-1}}{\overset{T}{\rightleftarrows}} B_1(R_1,I_1,D_1).$$

Restructuration cohérente

La restructuration de la base de S_0 en S_1 est *cohérente* si la transformation T de B_0 en B_1 l'est également ; c'est-à-dire si compte tenu des transformations de I_0 en I_1, *D_1 est bien une décomposition de R_1 avec les règles d'intégrité I_1.*

Ainsi, démontrer la cohérence d'une restructuration revient à un problème de décomposition de relations.

Restructurations réversibles

La restructuration de S_0 en S_1 est réversible :

- si elle est cohérente
- si la transformation T de B_0 en B_1 vérifie :
  - $R_0 \subset R_1$: ainsi B_1 contient toute l'information de B_0
  - D_0 décomposition de R_0 sous I_1 ; ainsi, par une transformation cohérente, peut-on déduire de B_1 le schéma $B_0'(R_0, I_1, D_0)$ qui a la même structure que B_0.

Structures équivalentes

S_0 et S_1 sont équivalentes si la transformation T de B_0 en B_1 vérifie les propriétés suivantes :

- $R_0 = R_1$: T conserve l'information
- D_0 est une décomposition de R_0 sous I_1 : T est ainsi réversible
- D_1 est une décomposition de R_1 sous I_0 : T^{-1} est réversible.

Donc démontrer l'équivalence de deux structures revient à un problème de décompositions de relations.

Par cette définition, nous pensons rejoindre la notion de "conservation de l'information" qui a permis une première définition de l'équivalence de structure [21] et qui, ensuite, a été définie dans [18] ainsi :

"La conservation de l'information dans la réorganisation est en gros définie par la préservation des relations structurelles (logiques)". A ces définitions, nous avons rajouté la conservation des règles d'intégrité.

Il existe des exemples de *restructurations réversibles* de S_0 en S_1 où S_0 et S_1 *ne sont pas équivalentes* : celles par exemple qui sont obtenues par un apport d'informations.

Navathe et Fry le font d'ailleurs remarquer en disant [22] : "une restructuration réversible conserve l'information mais n'interdit pas d'éventuels rajouts d'informations". Par rapport à ces propriétés, nous rajoutons qu'elle doit conserver toutes les règles d'intégrité nécessaires pour décomposer R_0 en D_0, qui est une autre formulation du fait que D_0 appartient à I_1.

Ainsi, *restructurer une base de données* consiste :

- d'abord à étudier les transformations des règles d'intégrité et de l'espace des constituants de la structure initiale
- ensuite, à décomposer la nouvelle relation compte tenu du nouvel ensemble d'invariants.

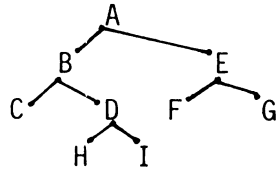
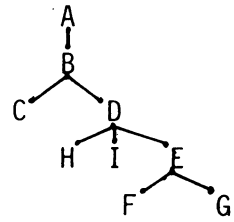
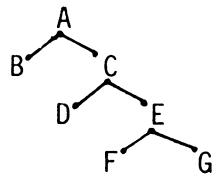
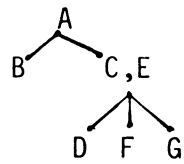
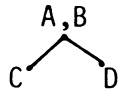
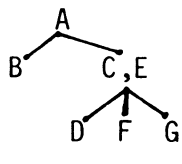
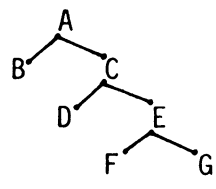
C'est donc *l'étude des décompositions* d'une relation, qui doit nous permettre de déterminer dans quelles circonstances telle opération de restructuration est cohérente.

4.3. RESTRUCTURATIONS POSSIBLES D'UNE BASE DE DONNEES

Nous venons de mettre en évidence la place des décompositions arborescentes dans le processus de restructuration. Nous allons maintenant supposer connues les propriétés de ces décompositions que nous démontrons dans l'annexe B. Elles nous permettront d'une part de mettre en évidence des opérations élémentaires de restructuration et les conditions dans lesquelles ces opérations sont cohérentes ; d'autre part elles nous permettront la détermination de structures équivalentes et de restructurations réversibles.

4.3.1. Opérations élémentaires de restructuration

Soit le schéma $B_0(R_0,I_0,D_0)$ d'une base de structure S_0 qui est transformé dans le schéma $B_1(R_1,I_1,D_1)$ de structure S_1 :

Nom de l'opération	Structure initiale S_0	Structure finale S_1	Conditions
regroupement 1			-
alourdissement 2			-
la projection 3			A : B|C,D
4			A,C : D|E

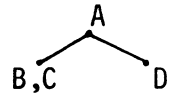
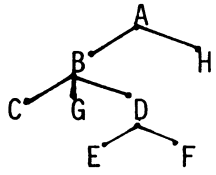
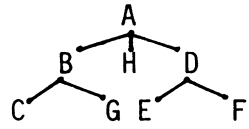
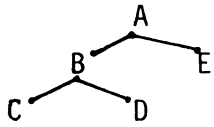
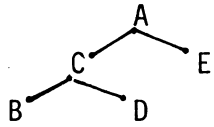
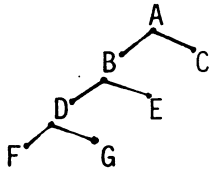
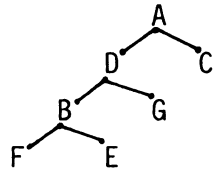
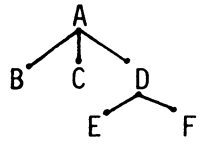
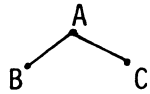
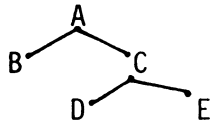
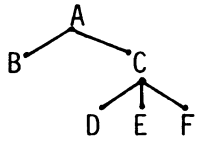
Nom de l'opération	Structure initiale S_0	Structure finale S_1	Conditions
dédoublement 5			A : B|C
			A : B|D,E,F
inversion 6			A,C → B
7			D → G
élimination de constituants 8			-
rajout de constituants 9			A,C → F

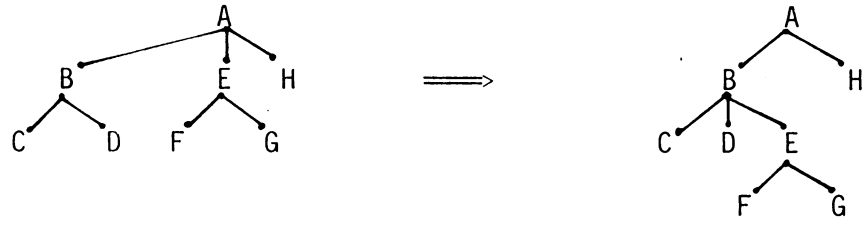
Fig. 4.10 - Tableau des opérations élémentaires de restructuration

Les conditions indiquées permettent de telles opérations ; il peut en exister d'autres, par exemple la condition A → B autorise l'opération de dédoublement. En effet, on peut en déduire la condition A : B|C.

4.3.2. Transformations élémentaires réversibles et à structures équivalentes

- **Regroupement** : On peut le schématiser par :

$$B_0(R_0,I_0,D_0) \xrightarrow{T_1} B_1(R_1,I_1,D_1)$$



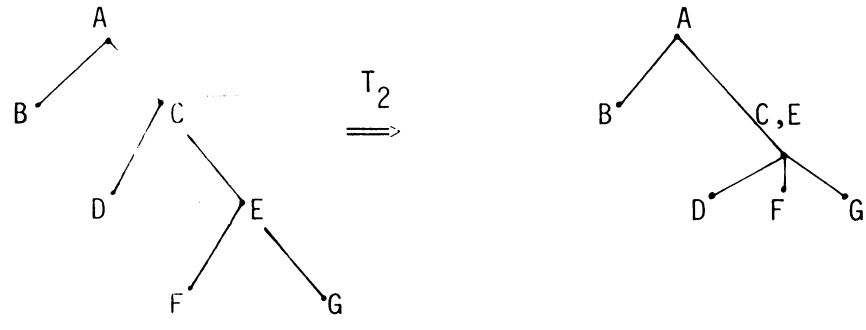
Une telle transformation T_1 est cohérente car elle ne nécessite aucune condition. La transformation inverse, le dédoublement, est cohérente si I_1 contient la règle d'intégrité :

A : B|E,F,G ;

ainsi T_1 est réversible si A : B|E,F,G appartient à I_1.

- **Alourdissement** : On peut le schématiser par :

$$B_0(R_0,I_0,D_0) \xrightarrow{T_2} B_1(R_1,I_1,D_1)$$



T_2 est cohérente car elle ne nécessite aucune condition, la transformation inverse, projection est cohérente si I_1 contient la règle d'intégrité :
AC : D|E.
Ainsi T_2 est réversible si justement AC : D|E appartient à I_1

La projection et le dédoublement conduisent à des structures équivalentes si les conditions qui permettent ces opérations.
respectivement dans les cas précédents : AC : D|E et A : B|C ne sont pas éliminées au cours de la transformation.

- Transformations réversibles

- toutes les transformations à structures équivalentes sont des transformations réversibles
- la projection, le dédoublement sont des transformations réversibles car il est alors toujours possible de passer de S_1 à S_0.
- le rajout de constituants l'est également pour la même raison.

4.3.3. Conclusions

- Restructuration

Nous avons mis en évidence des opérations élémentaires de restructuration qui peuvent elles-aussi se combiner pour former une restructuration.

Une combinaison d'opérations à structures équivalentes forme une restructuration à structures équivalentes.

Une combinaison d'opérations réversibles forme une restructuration réversible.

Exemple 4.4 de restructuration

Soit S_0 :

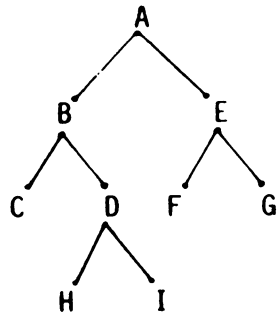


Fig. 4.11

et l'apparition de la règle d'intégrité : E → D ; alors S_0 peut être transformée en S_1 :

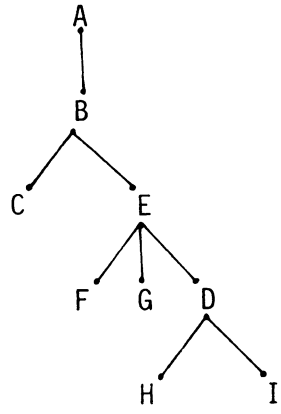


Fig. 4.12

En effet, il suffit d'appliquer à S_0 la transformation élémentaire de regroupement. On obtient S_2 :

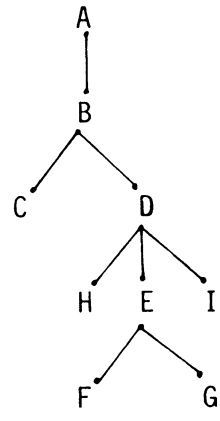


Fig. 4.13

On obtient ensuite S_1 de S_2 par application de la transformation élémentaire d'inversion.

- Comparaison des opérations élémentaires mises en évidence par [22] et celles que nous venons de présenter

- La compression (resp. l'expansion) est équivalente à l'alourdissement (resp. projection).

- La fusion de réalisations (resp. partition de réalisations) correspond au regroupement (resp. dédoublement), la notion de regroupement étant plus générale.

- L'inversion correspond à la même opération dans les deux cas.

- La modification de l'espace de constituants n'est pas étudiée dans [22].

- La fusion et la partition de groupes ne relèvent pas du cadre de ce chapitre car ils ne posent aucun problème lié aux règles d'intégrité et ne relèvent que de la transformation d'une structure théorique en une structure opérationnelle (cf. chapitre 5).

Nous avons ainsi étudié la cohérence des opérations élémentaires mises en évidence par [22] en déterminant des conditions dans lesquelles elles peuvent être appliquées.

4.4. CONCLUSIONS

. Notre objectif était de déterminer les opérations de restructuration susceptibles d'être effectuées sur une base de données, face aux situations suivantes :

- introduction de nouveauxconstituants
- modification des règles d'intégrité
- amélioration de l'efficacité de l'utilisation.

. Pour l'atteindre, nous nous sommes servis aussi bien de l'équivalence entre une structure arborescente S et une décomposition arborescente D que de l'expression des règles d'intégrité en termes de relations fonctionnelles et de décompositions. Nous avons ramené le *problème de la restructuration* à un *problème de décomposition* d'une relation compte tenu des règles d'intégrité.

Ainsi nous avons pu, pour la classe des structures arborescentes :

- définir les notions de restructuration cohérente, d'équivalence de structures, de restructuration réversible
- déterminer des opérations élémentaires de restructuration : regroupement, projection, alourdissement, dédoublement, inversion, modification de l'espace des constituants
- déterminer les circonstances dans lesquelles ces opérations peuvent être appliquées.

. Pour déterminer ces opérations de restructuration cohérente, nous nous sommes servis des propriétés des décompositions arborescentes et de leurs interactions avec un ensemble de règles d'intégrité (cf. Annexe B).

Ainsi, pensons-nous avoir précisé les opérations de restructuration et leur domaine d'application,et avoir montré l'apport du modèle relationnel dans l'explication du processus de restructuration d'une base de données.

C H A P I T R E 5

TRANSFORMATION D'UN ENSEMBLE DE RELATIONS DANS UNE STRUCTURE DE DONNEES DE TYPE RESEAU

S O M M A I R E

Cette transformation est une partie importante de notre démarche. Elle correspond d'une part à la situation dans laquelle se trouve le concepteur d'une base de données pour l'analyse du champ d'application de cette base : il ne peut, en effet, exprimer toutes les propriétés sémantiques directement à l'aide d'une structure réseau, mais (cf. Chapitre 2) doit les exprimer sous forme de relations. Il cherchera ensuite à traduire ces relations en une structure réseau. D'autre part, cette transformation constitue l'aboutissement de nos démarches précédentes : étude de la conception d'une base, restructuration, fusion de plusieurs bases. En effet, pour de telles études, nous avons commencé par exprimer les relations représentées dans une structure réseau (chapitre 2), puis selon les cas, nous avons transformé l'ensemble de relations obtenu en un autre ensemble. Maintenant il s'agit justement de le traduire en une structure réseau.

Ainsi, l'étude de la transformation d'un ensemble de relations dans une structure réseau constitue la dernière partie de notre démarche : elle doit nous permettre de passer des aspects purement logiques de conception, exprimés en termes relationnels, à la construction d'une structure effective en termes des systèmes de gestion de base de données actuels.

C'est dans cet esprit que nous avons posé dans l'introduction les questions suivantes :

- à propos de la conception d'une base de données : comment, à partir d'un ensemble de relations relatif à une application, construire la structure de données correspondante à l'aide d'un SGBD donné ? (B)
- la restructuration d'une base de données : comment obtenir une structure de données à partir d'un ensemble de relations ? (J)
- la fusion de plusieurs bases de données : comment effectuer le passage d'un ensemble de relations irredondant à une structure de données ? (P).

Une telle transformation : ensemble de relations → structure réseau constitue l'opération inverse du chapitre 2. Aussi, retrouverons-nous dans ce chapitre des notions déjà présentées dans le chapitre 2.

Nous nous proposons de construire cette transformation en plusieurs étapes. La première ordonne l'ensemble de relations considéré en une structure RELATIONNELLE. A la fin de cette étape, toutes les relations à *traduire* (cf. § 2.4) dans la structure finale sont contenues dans la structure RELATIONNELLE. La deuxième étape consiste à passer de la structure RELATIONNELLE à une structure RESEAU (cf. chapitre 2). Dans la dernière étape enfin, on obtient à partir de cette structure RESEAU, la structure réseau en termes du SGBD considéré.

A chacune de ces étapes, nous définirons des paramètres dont la détermination des valeurs conduit à la structure opérationnelle. Nous donnerons des valeurs canoniques à ces paramètres, pour établir sans apport d'informations, une structure de données : elle aura la particularité de n'accepter que le minimum de redondance et de ne privilégier aucun chemin d'accès.

Pour illustrer toute cette démarche, nous reprenons dans le dernier paragraphe l'exemple de la réservation des chambres d'hôtel dans une station.

5.1. TRANSFORMATION D'UN ENSEMBLE DE RELATIONS EN UNE STRUCTURE RELATIONNELLE

C'est la première partie de la transformation d'un ensemble de relations E en une structure réseau. Elle a pour but d'ordonner les relations dans une structure appelée structure RELATIONNELLE afin de faciliter ensuite le passage à une structure réseau.

5.1.1. Définition d'une structure RELATIONNELLE

Une structure RELATIONNELLE est formée :

- des relations R d'index k(R), de l'ensemble considéré E,
- de liaisons entre deux relations R et S de cet ensemble ; elles sont de deux types :

  . liaison fonctionnelle (R-F).
  Dans ce cas, k(S) appartient à R sans être contenue dans k(R) ; la relation fonctionnelle $k(R) \rightarrow k(S)$ existe donc sans être triviale.

  . liaison hiérarchique (H).
  Dans ce cas, k(S) appartient à R et à son index k(R) : la relation fonctionnelle $k(R) \rightarrow k(S)$ existe et est triviale.

Par définition même de la structure RELATIONNELLE, son obtention à partir d'un ensemble de relations s'appuie sur le tableau suivant :

A et B désignant deux relations

ENSEMBLE DE RELATIONS ⟶	STRUCTURE RELATIONNELLE
$k(B) \subset k(A)$	liaison hiérarchique allant de A vers B
$k(A) \rightarrow k(B)$	liaison fonctionnelle allant de A vers B

Tableau 5.1 - Transformation d'un ensemble de relations en une structure RELATIONNELLE

Dans les exemples, nous représenterons souvent une structure RELATIONNELLE par un graphe orienté dont les arêtes représentent les liaisons et les noeuds, les relations non transformées en liaison ; les arêtes sont orientées dans le même sens que les liaisons, et sont marquées du type de la liaison.

Exemple 5.1

Soit l'ensemble irredondant et optimisé de relations RD :

R_1	[A,B]	avec	A → B	index :	A
R_2	[A,C,D]	avec	A,C → D		A,C
R_3	[E,A,C]	avec	E → A,C		E
R_4	[E,F]				E,F
R_5	[F,G]	avec	F → G		F

Liaison fonctionnelle (R-F)

celle de R_3 vers R_2.

Liaison hiérarchique (H)

celle de R_2 vers R_1
celle de R_4 vers R_3
celle de R_4 vers R_5.

On peut représenter simplement la structure RELATIONNELLE par le graphe suivant :

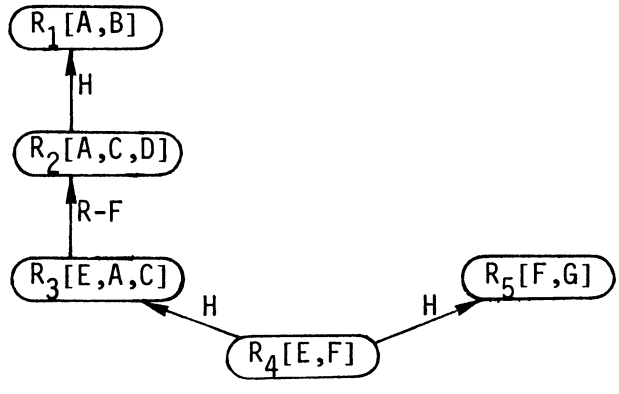


Fig. 5.2

5.1.2. Paramètres de la transformation

Par définition même d'une structure RELATIONNELLE, à un ensemble de relations il ne correspond qu'une structure et réciproquement. De plus, dans la suite du processus de transformation d'un ensemble de relations en une structure réseau, il ne sera plus possible de rajouter ou d'enlever des relations. Il s'agit donc, au cours de cette étape, de *choisir l'ensemble de relations que l'on veut traduire* (au sens du chapitre 2) dans la structure réseau finale : c'est le seul paramètre de cette étape.

Quelles sont les contraintes de ce choix ?

Soit E l'ensemble de relations de départ, E_T l'ensemble des relations à traduire et E_M l'ensemble minimal de relations mis en évidence dans le § 3.3.4.

Pour que E_T puisse être choisi, il faut qu'il conserve toute l'information contenue dans E, c'est-à-dire dans E_M puisque E_M contient toutes les relations irredondantes de E et puisque l'on peut obtenir E à partir de E_M par des opérations de composition et de projection (propriété de E_M : cf. § 3.3). Aussi pour que E_T puisse être choisi, il faut et il suffit que *l'on puisse déduire de E_T toutes les relations contenues dans E_M*. Les relations de E_M qui n'appartiennent pas à E_T seront alors des relations *déduites* de la structure réseau finale.

C'est la seule contrainte liée à ce choix.

Pourquoi ne pas prendre E_M pour E_T ?

C'est une solution cohérente du point de vue de la conservation de l'information puisque toutes les relations contenues dans E peuvent être déduites de celles de E_M (cf. § 3.3). De plus, nous avons montré l'intérêt qu'il y aurait à prendre E_M comme ensemble de relations à traduire : la structure réseau finale ne contiendrait alors aucune imperfection intra ou inter-relation (§ 3.1).

Tous ces avantages nous font retenir cette solution comme solution canonique. Mais généralement une telle solution ne peut être retenue si l'on cherche à construire une structure opérationnelle : il s'agit bien souvent d'introduire d'autres chemins d'accès pour améliorer les performances de la base (exemple : temps de réponse). Ces contraintes opérationnelles obligent généralement le concepteur de la base à introduire de la redondance et à modifier E_M. Comme E_M est unique (sous certaines conditions), il est sûr que l'ensemble E_T considéré contiendra alors des relations en première ou deuxième forme et conduira donc à des imperfections de fonctionnement, notamment des contrôles d'intégrité plus longs.

C'est au concepteur de savoir si E_T présente un intérêt opérationnel suffisant par rapport aux imperfections de structure qu'il entraîne.

Par quelles transformations élémentaires peut-on transformer E_M en E_T tout en conservant l'information ?

- en rajoutant des relations redondantes qui se déduisent de celles de E_M par composition ou composition-projection. Ce rajout peut éliminer de E_T des relations de E_M qui se déduisent alors de celles de E_T (cf. exemple 5.2) ; une telle transformation est causée par la fréquence importante de l'utilisation de ces relations ;

- en rajoutant à l'index d'une relation R l'index d'une relation S afin de conditionner l'existence des réalisations de la relation R à celle des réalisations correspondantes de S. Une telle transformation correspond à la transformation d'une relation de E_m en relation non élémentaire (cf. exemple 5.3).

- en regroupant deux relations R et S de E_M en une seule dans le cas où il existe une liaison de R vers S. Une telle transformation est intéressante si la relation obtenue est souvent demandée, ou bien si la relation S est rarement mise à jour : ainsi les imperfections liées aux relations en première et deuxième forme sont peu à craindre (cf. exemple 5.4).

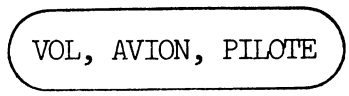
Exemple 5.2 - Rajout de relations redondantes

Soit la relation R_3[VOL, AVION, PILOTE] (cf. exemple 3.3) munie des relations fonctionnelles non redondantes :

$$E_M \begin{cases} \text{VOL} \rightarrow \text{AVION} \\ \text{AVION} \rightarrow \text{PILOTE} \end{cases}$$

Si l'on s'aperçoit qu'en fait c'est la relation VOL, PILOTE qui est plus utilisée opérationnellement, on peut la rajouter dans l'ensemble des relations à traduire E_T, tout en sachant qu'il faudra en assurer la cohérence par programme. Mais alors, point n'est besoin de traduire [AVION, PILOTE] : elle peut se déduire en effet de R_3 par projection. E_T se résume à la seule relation R_3.

La structure RELATIONNELLE correspondante est alors :

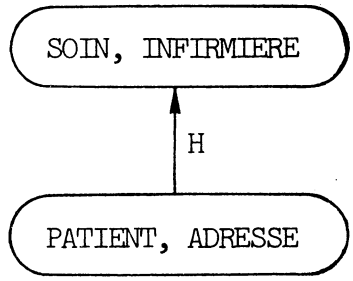


Exemple 5.3 - Création de liaisons hiérarchiques

Soit la base de données d'un hôpital dont l'ensemble non redondant de relations est :

$$E_M \begin{cases} \text{PATIENT} \rightarrow \text{ADRESSE} \\ \text{SOIN} \rightarrow \text{INFIRMIERE} \\ [\text{PATIENT, SOINS}] \end{cases}$$

Si l'on veut que dans les traitements de la base, l'élimination d'un type de soin correspond automatiquement à l'élimination de tous les patients traités par ces soins, il suffit alors d'introduire la liaison hiérarchique suivante :

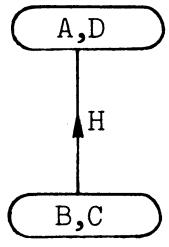


qui correspond en fait à l'ensemble E_T de relations à traduire suivant :

$$\begin{cases} \text{SOIN} \rightarrow \text{INFIRMIERE} \\ \text{SOIN, PATIENT} \rightarrow \text{ADRESSE} \end{cases}$$

E_T se déduit trivialement de E_M puisque SOIN, PATIENT → ADRESSE n'est pas élémentaire.

Remarque : Dans le cas général où $E_M = \{A \rightarrow D, B \rightarrow C \text{ et } B \rightarrow A\}$, et que l'on désire une structure arborescente, la structure représentée simplement par le graphe



ne contient aucune redondance.

Exemple 5.4 - Regroupement de relations

Soit l'ensemble non redondant E_M

$$\begin{cases} \text{VILLE} \rightarrow \text{DEPARTEMENT} \\ \text{DEPARTEMENT} \rightarrow \text{PREFECTURE} \end{cases}$$

il est possible qu'on préfère traduire dans la structure la relation [VILLE, DEPARTEMENT, PREFECTURE] :

- d'une part elle permet de retrouver par projection les relations irredondantes
- d'autre part bien qu'elle soit en deuxième forme, elle ne risque pas de causer trop d'incohérences étant donné le nombre de fois qu'un DEPARTEMENT change de PREFECTURE.

5.1.3. Exemple 5.5

Reprenons l'exemple de la réservation des chambres d'hôtel dans une station (§ 3.4 - page III.27). L'ensemble minimal E_M est :

$s \rightarrow s_1$	qui devient par regroupe-	$s \rightarrow s_1$
$h,s \rightarrow h_1$	ment des relations fonc-	$h,s \rightarrow h_1$
$h,s,p \rightarrow p_1$	tionnelles de même partie	$h,s,p \rightarrow p_1$
$c,p \rightarrow s$	gauche :	$c,p \rightarrow s,h,d_1,r_1$
$c,p \rightarrow h$		$c \rightarrow c_1$
$c,p \rightarrow d_1$		
$c,p \rightarrow r_1$		
$c \rightarrow c_1$		

La structure RELATIONNELLE canonique est alors représentée simplement par ce graphe où les arêtes indiquent les liaisons :

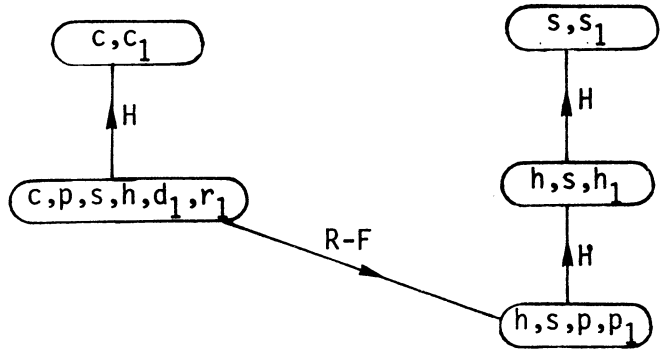


Fig. 5.3

5.2. TRANSFORMATION D'UNE STRUCTURE RELATIONNELLE EN UNE STRUCTURE RESEAU

Dans le paragraphe précédent, nous avons construit la première partie du processus de transformation d'un ensemble de relations en une structure réseau. Maintenant nous allons étudier le passage d'une structure RELATIONNELLE à une structure RESEAU.

Nous commençons par dégager les aspects relationnels d'une structure RESEAU (cf. Chapitre 2) qui nous permettront ensuite de donner les règles de transformation d'une structure RELATIONNELLE en une structure RESEAU. Nous en déterminons les différents paramètres dont il est nécessaire de préciser les valeurs pour construire une structure opérationnelle. En dernier lieu, nous reprenons l'exemple du § 5.1.3.

5.2.1. Aspects relationnels d'une structure RESEAU

Nous avons défini la notion de structure RESEAU dans le § 2.3.
Dans le § 2.4, nous avons donné les éléments qui permettent de passer d'une structure RESEAU à un ensemble de relations.
C'est à la démarche inverse à laquelle nous nous intéressons maintenant.

Soit R un ensemble de relations mises sous forme canonique* ; à chaque relation R_i de R est associé son espace de constituants A_i et son index $k(R_i)$.

Il y a 4 types de relations qui vont se transformer en liaison ou en groupe dans une structure RESEAU :

- Relation fonctionnelle

$\exists\ R_i, R_j, R_\ell \in \mathbf{R}$ telles que

- $k(R_i) = k(R_\ell)$
- $A_\ell = k(R_i) \vee k(R_j)$
- $k(R_i) \rightarrow k(R_j)$ non triviale

C'est-à-dire si les relations $(A \rightarrow B)$ et $(A \rightarrow C)$ existent, elles n'ont pas été regroupées en $(A \rightarrow B,C)$ (cf. [6]).

R_ℓ ne peut être représentée dans une structure RESEAU que par une liaison fonctionnelle allant de R_i vers R_j.

- Relation de type binaire

$\exists\ R_i, R_j, R_\ell \in R$ telles que

- $k(R_\ell) = A_\ell$
- $k(R_\ell) = k(R_i) \vee k(R_j)$

R est alors une relation de type binaire ; elle est représentée dans une structure RESEAU par deux liaisons binaires, l'une reliant R_i à R_j, l'autre R_j à R_i.

- Relation hiérarchique

$\exists\ R_i, R_j \in R$ telles que

- R_i n'est pas une relation de type binaire
- $k(R_j) \subset k(R_i)$

Une telle situation est représentée dans une structure RESEAU par :
soit une liaison hiérarchique allant de R_i vers R_j
soit une liaison fonctionnelle allant de R_i vers R_j.

- Relation noeud

$\exists\ R_i \in R$ telle que $\nexists j$, $R_j \in R$ et $k(R_j) \subset A_i$.

Une telle relation est représentée par un GROUPE dans une structure RESEAU.

Le tableau suivant récapitule tous ces résultats :

Soient X et Y deux relations inter-noeuds :

ENSEMBLE DE RELATIONS		STRUCTURE RESEAU
relation noeud	X	GROUPE X
relation fonctionnelle	K(X) → K(Y)	LIAISON FONCTIONNELLE de X vers Y
relation hiérarchique	K(Y) ⊂ K(X)	LIAISON HIERARCHIQUE de X vers Y
relation de type binaire	[K(X), K(Y)]	LIAISON BINAIRE entre X et Y

Tableau 5.4 - Aspects relationnels d'une structure RESEAU

Le rapprochement des tableaux 2.3 et 5.4 montre qu'il existe une correspondance biunivoque entre un ensemble de relations et une classe de structures RESEAU. Du fait qu'un ensemble de relations se transforme en une seule structure RELATIONNELLE et réciproquement, il existe une *correspondance biunivoque entre une structure RELATIONNELLE et une classe de structures RESEAU* : deux structures RESEAU appartiennent à la même classe si et seulement si elles ne diffèrent que par la représentation de relations hiérarchiques.

5.2.2. Obtention d'une structure RESEAU à partir d'une structure RELATIONNELLE

Le paragraphe précédent a associé à une structure RELATIONNELLE : RL, une classe de structures RESEAU : RS. Il s'agit maintenant de choisir une structure parmi cette dernière classe.

Pour cela, il faut :

- choisir la représentation de toutes les liaisons hiérarchiques de RL
- créer dans RS les liaisons binaires qui n'existent pas explicitement dans RL ; de plus il s'agit d'éliminer dans RS la redondance de constituants puisque, contrairement aux relations de RL,

un GROUPE de RS peut ne pas contenir tous les constituants qui lui sont associés (cf. § 2.2) et qui appartiennent à un de ses index.

5.2.2.1. Détermination d'une structure RESEAU

- Représentation des liaisons hiérarchiques de RL dans RS

Deux solutions sont possibles :

soit - choisir l'une d'entre elles et la représenter dans RS par une liaison hiérarchique, les autres étant représentées par des liaisons fonctionnelles,

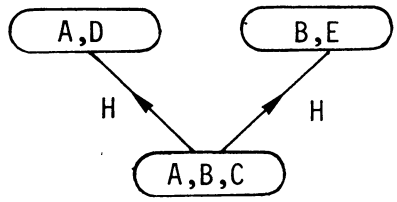
soit - les transformer toutes en liaisons fonctionnelles de RS.

Exemple 5.6

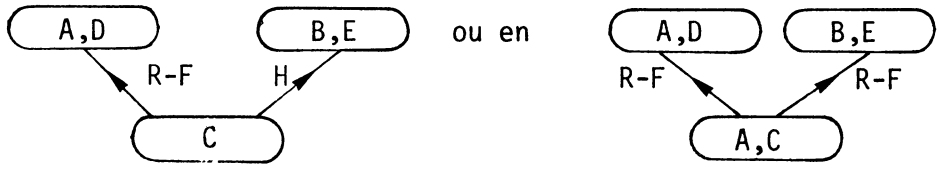
Soit E_T l'ensemble des relations à traduire

$$E_T \begin{cases} A,B \rightarrow C \\ A \rightarrow D \\ B \rightarrow E \end{cases}$$

la structure RELATIONNELLE correspondante peut être représentée par ce graphe :



elle peut se transformer dans la structure RESEAU soit en



ou en

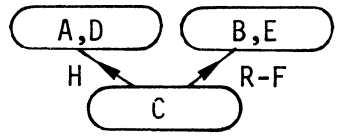


Fig. 5.5

- Création de liaisons binaires dans RS

Si d'une relation X de RL partent deux liaisons hiérarchiques et deux seulement, vers les noeuds Y et Z et si la relation X ne contient que les index de Y et de Z, alors la relation de X est une *relation de type binaire.*

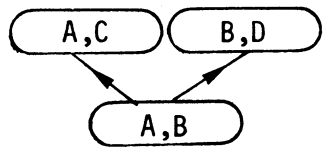
Elle est transformée dans RS en deux liaisons binaires, l'une allant de Y vers Z, l'autre de Z vers Y.

Exemple 5.7

Soit E_T l'ensemble des relations à traduire en la structure RESEAU.

$$E_T \begin{cases} [A,B] \\ A \rightarrow C \\ B \rightarrow D \end{cases}$$

Voici la structure RELATIONNELLE correspondante :



Elle se transforme dans la structure RESEAU en :

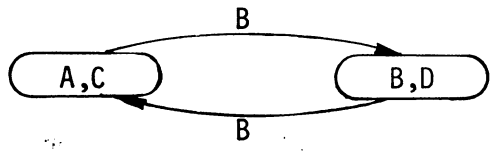


Fig. 5.6

5.2.2.2. **Elimination de la redondance de constituants**

- Elimination de la redondance de constituants d'index

Une relation de RL est formée de tous ses constituants ; par contre, un GROUPE G (cf. § 2.3.1) de constituants de RS peut ne pas contenir tous les constituants de la relation associée (cf. § 2.4.1). Ils se trouvent alors dans les index de GROUPES G_i auxquels G est relié par une liaison hiérarchique ou fonctionnelle.

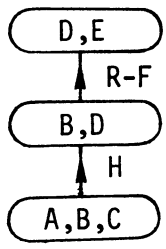
Ainsi, si une relation X de RL est reliée à une relation Y par une liaison allant de X vers Y et si l'index k(Y) fait partie de l'espace de constituants de X, alors X est transformée lors du passage de RL en RS en un GROUPE GX dont l'ensemble de constituants ne comprend pas k(Y).

Exemple 5.8

Soient les relations

$$E_T \begin{cases} A,B \rightarrow C \\ B \rightarrow D \\ D \rightarrow E \end{cases}$$

Voici la structure RELATIONNELLE :



qui se transforme dans la structure RESEAU :

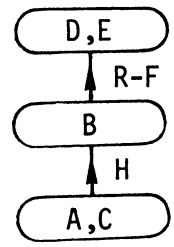


Fig. 5.7

- Elimination de la redondance de constituants simples

Le même constituant peut appartenir à plusieurs relations sous d'autres conditions que les précédentes ; c'est un constituant *simple* ; plutôt que d'être répété dans RS, il peut être mis en commun par la création dans RS d'un GROUPE spécial ne contenant que lui : tous les GROUPES des relations le contenant sont reliés à celui-ci par une liaison fonctionnelle.

Exemple 5.9

$$E_T \begin{cases} [A,B,C] & \text{avec} \quad A,B \rightarrow C \\ [D,B] & \text{avec} \quad D \rightarrow B \end{cases}$$

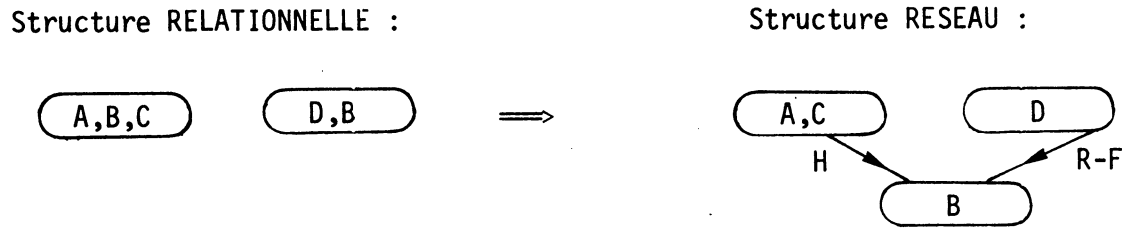


Fig. 5.8

5.2.2.3. Tableau récapitulatif

STRUCTURE RELATIONNELLE		STRUCTURE RESEAU
relation simple	→	GROUPE avec élimination des constituants multiples de la relation
relation type binaire	→	-
constituant multiple simple	→	GROUPE
liaison fonctionnelle	→	liaison fonctionnelle
liaison hiérarchique	⟩	liaison hiérarchique
liaison entre relation de type binaire et relations simples N1 et N2	→	liaison binaire entre N1 et N2

Tableau 5.9 - Transformation d'une structure RELATIONNELLE en une structure RESEAU

5.2.3. Paramètres de cette transformation

Ils sont de deux types différents :

- comme à une structure RELATIONNELLE correspond une classe de structures RESEAU, il s'agit de préciser (§ 5.2.2.1) la transformation de toutes les liaisons hiérarchiques de la structure RELATIONNELLE pour n'obtenir qu'une structure RESEAU.
Si d'une relation partent plusieurs liaisons hiérarchiques, il s'agit en particulier d'en privilégier une, celle qui sera matérialisée par une liaison hiérarchique dans la structure RESEAU.

- le § 5.2.2.2 cherche à éliminer toute redondance de constituants. L'objectif poursuivi est d'éviter des mises à jour compliquées. Par contre, une certaine redondance peut être souhaitée dès qu'il s'agit d'améliorer les chemins d'accès.

Structure canonique

- si d'une relation partent plusieurs liaison hiérarchiques dans RL, celles-ci sont toutes transformées en liaisons fonctionnelles dans la structure RESEAU

- si d'une relation ne part qu'une liaison hiérarchique dans RL, celle-ci est transformée en une liaison hiérarchique dans la structure RESEAU

- aucune redondance de constituants n'est autorisée.

5.2.4. Exemple 5.10

Reprenons la structure RELATIONNELLE de l'exemple de la réservation des chambres d'hôtel (§ 5.1.3 - page V.9).

Sa transformation dans une structure RESEAU ne pose que des problèmes d'élimination de la redondance de constituants ; elle donne :

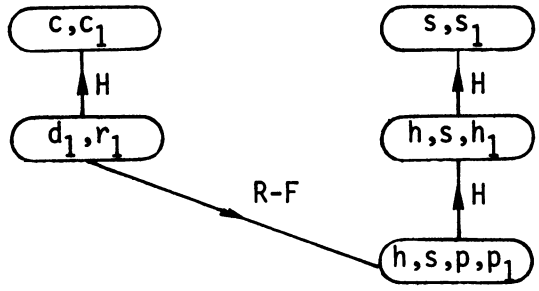


Fig. 5.10

5.3. TRANSFORMATION D'UNE STRUCTURE RESEAU EN UNE STRUCTURE SOCRATE OU CODASYL

C'est la dernière étape de l'obtention d'une structure SOCRATE ou CODASYL à partir d'un ensemble de relations. Elle consiste à traduire la structure RESEAU obtenue précédemment en termes du langage de définition de structure du SGBD utilisé, c'est-à-dire pour notre étude : SOCRATE ou CODASYL. Elle est la transformation inverse de celle construite dans le § 2.3.

5.3.1. Obtention à partir d'une structure RESEAU d'une structure SOCRATE ou CODASYL

La définition d'une structure RESEAU fige tous les choix entre liaisons fonctionnelles et liaisons hiérarchiques ; de plus, elle contient toutes les redondances de relations et de constituants désirées par l'utilisateur. Aussi à une structure RESEAU ne peut correspondre par construction qu'une classe de structures SOCRATE ou CODASYL.

Les différences entre deux structures d'une même classe ne sont dues qu'aux différences de représentation possibles d'un même type de liaison :

pour SOCRATE :

Il y a le choix de l'orientation de la liaison. En effet, une liaison SOCRATE entre deux entités A, B allant de A vers B correspond à un chemin d'accès non symétrique qui permet d'atteindre B à partir de A

et non l'inverse. Aussi, face à une liaison RESEAU, faut-il choisir le sens de la liaison SOCRATE correspondante. Il en est particulièrement ainsi pour les liaisons RESEAU binaires.

De plus, il faut choisir le moyen de représenter une liaison binaire entre : INVERSE, ENTITE-REFERENCE.

pour CODASYL :

Il faut choisir la représentation des liaisons hiérarchiques entre : LIEN HIERARCHIQUE ou GROUPE REPETITIF.

Le tableau suivant fournit les informations nécessaires pour obtenir, à partir d'une structure RESEAU, une structure SOCRATE ou CODASYL.

A et B désignant deux GROUPES :

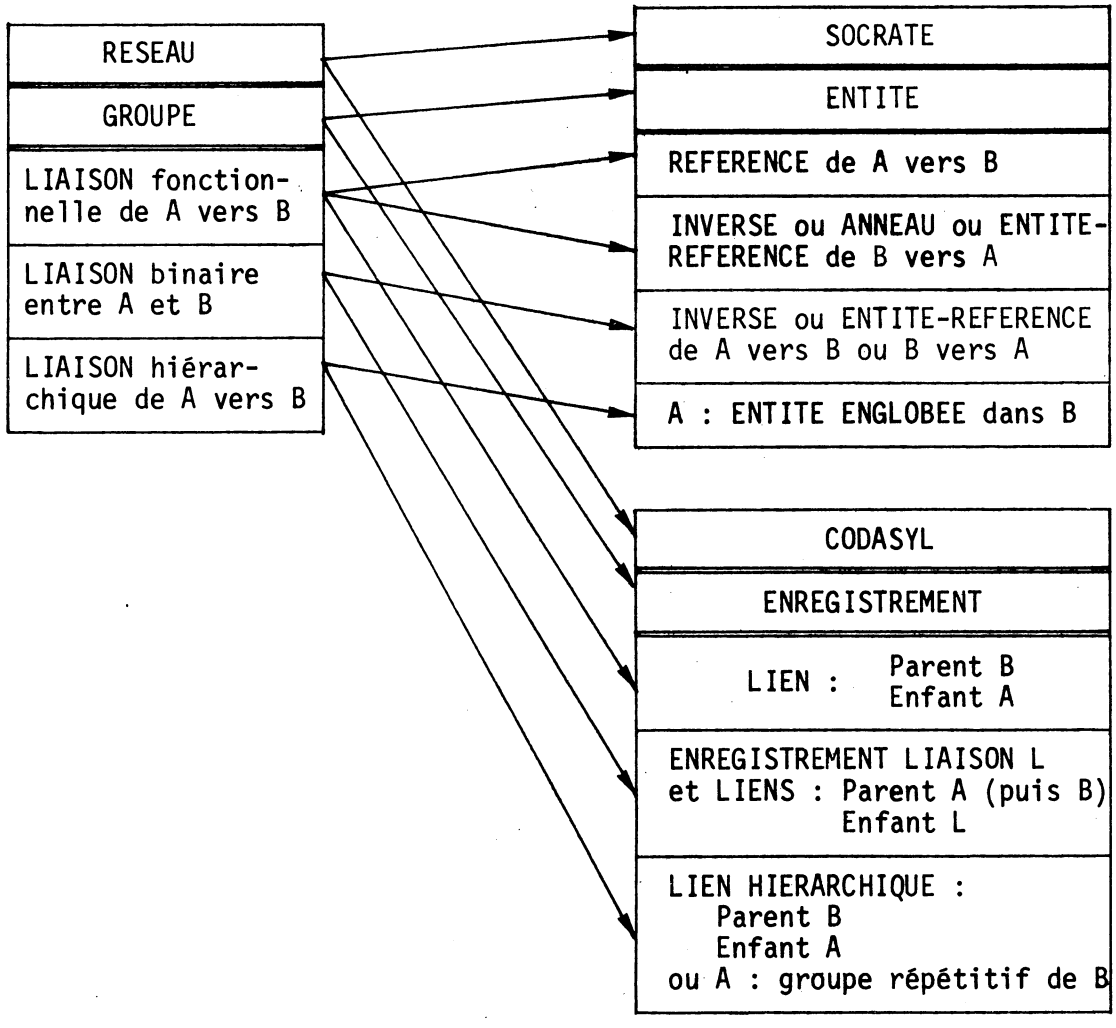


Tableau 5.11 - Transformation d'une structure RESEAU en une structure SOCRATE ou CODASYL

5.3.2. Paramètres de cette transformation

A une structure SOCRATE ou CODASYL, il ne correspond qu'une seule structure RESEAU (§ 2.3), il existe donc une correspondance biunivoque entre une classe de structures SOCRATE ou CODASYL et une structure RESEAU. Pour déterminer la structure SOCRATE ou CODASYL recherchée, il faut préciser certains paramètres : pour SOCRATE, le sens du chemin d'accès à faire correspondre à une liaison RESEAU et ensuite le type de la liaison SOCRATE ; pour CODASYL, le choix d'implantation des LIENS (liste unidirectionnelle, bidirectionnelle, en anneau...) et le type de liaison à faire correspondre à une liaison hiérarchique RESEAU.

Les choix des types de liaison font intervenir essentiellement des critères de rapports de masses d'informations.

5.3.3. Exemple 5.11

A partir de la struceure RESEAU de l'exemple de réservation des chambres (§ 5.2.4), nous donnons la structure SOCRATE correspondante puis la structure CODASYL.

Structure SOCRATE

```
entité STATION
    début
    |   NOM
    |   ALTITUDE
    |   DEPARTEMENT
    |   entité HOTEL
    |       début
    |       |   NOM
    |       |   CLASSE
    |       |   NOMBRE-DE-CHAMBRES
    |       |   entité CONDITIONS
    |       |       début
    |       |       |   PERIODE
    |       |       |   NOMBRE DE CHAMBRES LIBRES
    |       |       |   PRIX
    |       |       fin
    |       fin
    fin
```

entité CLIENT
début
NOM
ADRESSE
TELEPHONE
entité RESERVATION-DEMANDE
début
CONDITIONS référence une CONDITION d'un HOTEL d'une STATION
CLASSE D'HOTEL
NOMBRE-DE-CHAMBRES
NOMBRE-DE-PERSONNES
PRIX TOTAL
fin
fin

Structure CODASYL

enregistrement STATION
NOM
ALTITUDE
DEPARTEMENT
index est NOM

enregistrement HOTEL
NOM
CLASSE
NOMBRE-DE-CHAMBRES
NOM-STATION virtuel
index est NOM, NOM-STATION

enregistrement CONDITIONS
NOM-STATION virtuel
NOM-HOTEL virtuel
PERIODE
NOMBRE-DE-CHAMBRES LIBRES
PRIX
index est NOM-HOTEL, NOM-STATION, PERIODE

enregistrement CLIENT
    NOM
    ADRESSE
    TELEPHONE
index est NOM

enregistrement RESERVATION-DEMANDE
    NOM-STATION virtuel
    NOM-HOTEL virtuel
    NOM-CLIENT virtuel
    PERIODE virtuel
    CLASSE-HOTEL
    NOMBRE-DE-CHAMBRES
    NOMBRE-DE-PERSONNES
    PRIX TOTAL
index est NOM-CLIENT, PERIODE

lien STATION, HOTEL
    parent est STATION
    enfant est HOTEL
    lien est hiérarchique
    NOM dans parent correspond à NOM-STATION dans enfant

lien HOTEL, CONDITIONS
    parent est HÔTEL
    enfant est CONDITIONS
    lien est hiérarchique
    NOM, NOM-STATION dans parent correspondent à NOM-HOTEL, NOM-STATION
    dans enfant

lien CLIENT, RESERVATION-DEMANDE
    parent est CLIENT
    enfant est RESERVATION-DEMANDE
    lien est hiérarchique
    NOM dans parent correspond à NOM-CLIENT dans enfant

lien CONDITIONS, RESERVATION-DEMANDE
    parent est CONDITIONS
    enfant est RESERVATION-DEMANDE
    NOM-STATION, NOM-HOTEL, PERIODE dans parent correspondent à
    NOM-STATION, NOM-HOTEL, PERIODE dans enfant.

5.4. CONCLUSIONS

Notre objectif était de transformer un ensemble de relations en une structure réseau d'un SGBD donné. Cette transformation correspond à la dernière partie de notre démarche qui étudie la conception d'une base de données à partir des informations sémantiques exprimées à l'aide du modèle relationnel.

Comme elle est l'inverse de celle étudiée précédemment, nous nous sommes servis des notions introduites au chapitre 2.

Nous l'avons décomposé en trois parties dans le but de la rendre la plus indépendante possible du SGBD utilisé :

- la première consiste à ordonner l'ensemble de relations en une structure réseau appelée structure RELATIONNELLE. Une fois l'ensemble de relations déterminé, il existe une correspondance biunivoque entre cet ensemble et la structure RELATIONNELLE correspondante ;
- la deuxième consiste à transformer cette structure RELATIONNELLE en une structure RESEAU. Nous avons montré qu'il existe une correspondance biunivoque entre une structure RELATIONNELLE et une classe de structures RESEAU ;
- la troisième consiste à transformer cette structure RESEAU en une structure réseau du SGBD concerné. Seule cette dernière partie dépend d'un SGBD particulier.

En conséquence de cette démarche, on peut en déduire la correspondance biunivoque entre un ensemble de relations minimal non redondant E_M tel qu'il est défini dans le § 3, et une classe de structures réseau du SGBD concerné. Pour choisir la structure opérationnelle la mieux appropriée, parmi les structures possibles de la même classe, le concepteur de la base de données doit préciser les valeurs des paramètres que nous avons mis en évidence.

Ici, se trouve un intérêt supplémentaire de l'approche relationnelle : celui de rendre explicites les choix à faire.

Il existe plusieurs types de paramètres :

. Le premier type est lié au choix de l'ensemble de relations à traduire dans la structure. Les critères de ce choix reposent essentiellement sur la fréquence plus ou moins grande de l'utilisation future de certaines relations par rapport à d'autres. Il est sûr que si l'ensemble de relations à traduire n'est pas minimal irredondant, la structure réseau finale contiendra des imperfections de fonctionnement comme l'ont montré les résultats du chapitre 3. Aussi le concepteur de la base de données doit-il trouver un compromis entre ces imperfections et les avantages d'utilisation d'un tel choix.
Les paramètres dont il est question concernent seulement la première partie de la transformation.

. Le second type de paramètres est relatif à la deuxième partie de la transformation structure RELATIONNELLE → structure RESEAU. Il concerne les relations dont le concepteur veut rendre l'existence des réalisations correspondantes, dépendante de celles d'autres relations.

. Le troisième type de paramètres est également relatif à la deuxième partie. Il concerne la redondance des constituants.

. Le quatrième type est relatif à la dernière partie de la transformation : structure RESEAU → structure réseau d'un SGBD donné. Il concerne des choix de modes de liaison propres au SGBD et également pour SOCRATE le sens des chemins d'accès.

Nous résumons notre démarche à l'aide du tableau suivant :

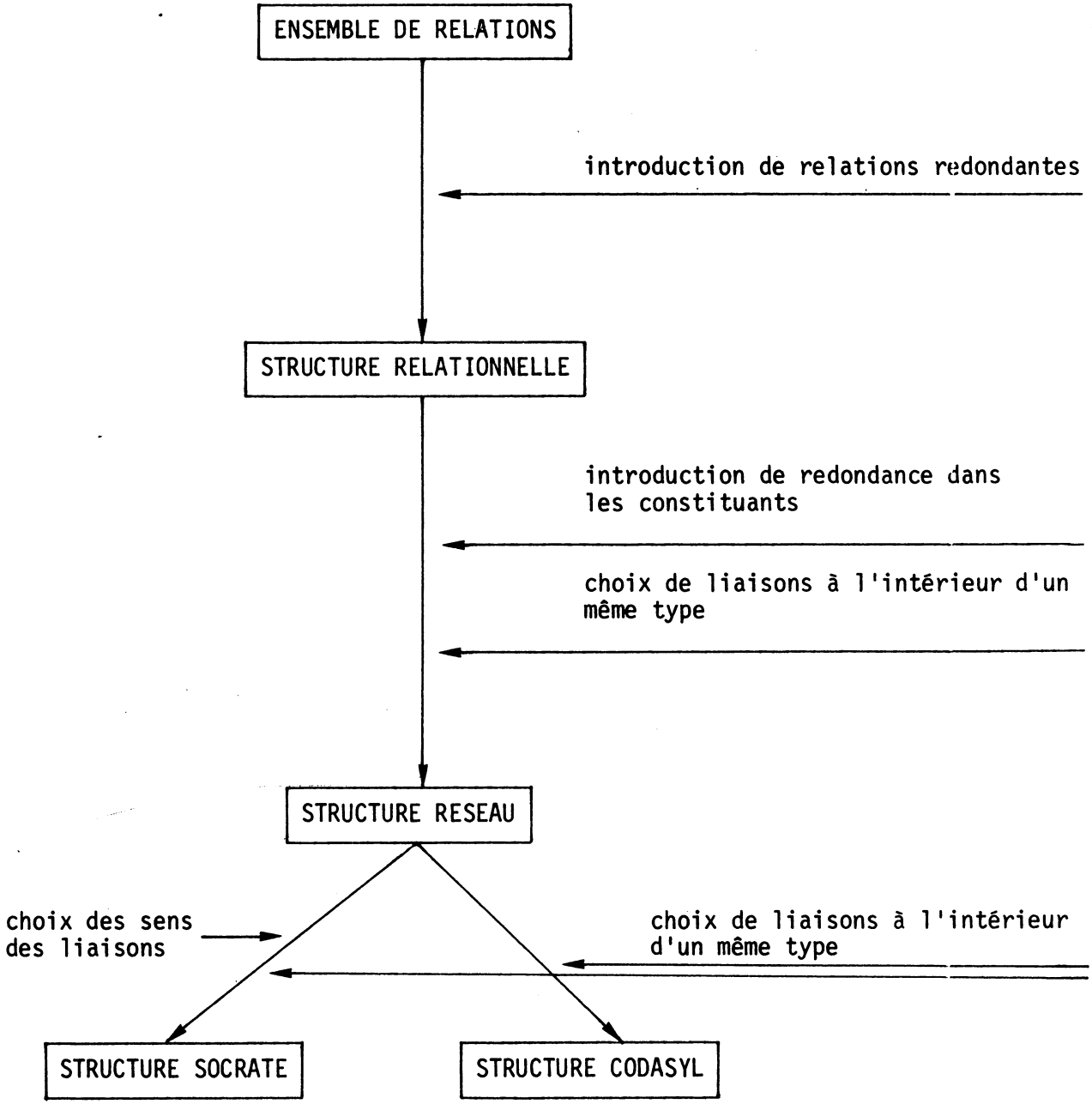


Tableau 5.12 - Transformation d'un ensemble de relations dans une structure SOCRATE ou CODASYL

C H A P I T R E 6

REALISATIONS TECHNIQUES

S O M M A I R E

6. REALISATIONS

Nous présentons dans ce chapitre la première version d'un système informatique qui permet :

soit d'extraire l'ensemble de relations "traduites d'une structure SOCRATE" (§ 2.4) pour ensuite en trouver l'ensemble minimal irredondant (chapitre 2 et 3). Cette partie correspond à notre démarche de conception d'une base de données à partir d'une base déjà existante ;

soit de rentrer directement un ensemble de relations et d'en trouver l'ensemble minimal non redondant (Chapitre 3) ; cette partie correspond à notre démarche de conception d'une base de données à partir des résultats d'analyse exprimés sous forme de relations.

Du point de vue technique, nous traitons ces deux points en trois parties :

- la transformation d'une structure SOCRATE dans un ensemble de relations,
- l'analyse d'un ensemble de relations,
- la saisie d'un ensemble de relations.

Tous les programmes ont été écrits en PL1 sous la système CP/CMS. Dans ce qui suit, nous reprenons la présentation des programmes écrits en collaboration avec M. ADIBA, telle qu'elle figure dans [4].

Voici l'organigramme du système informatique considéré.

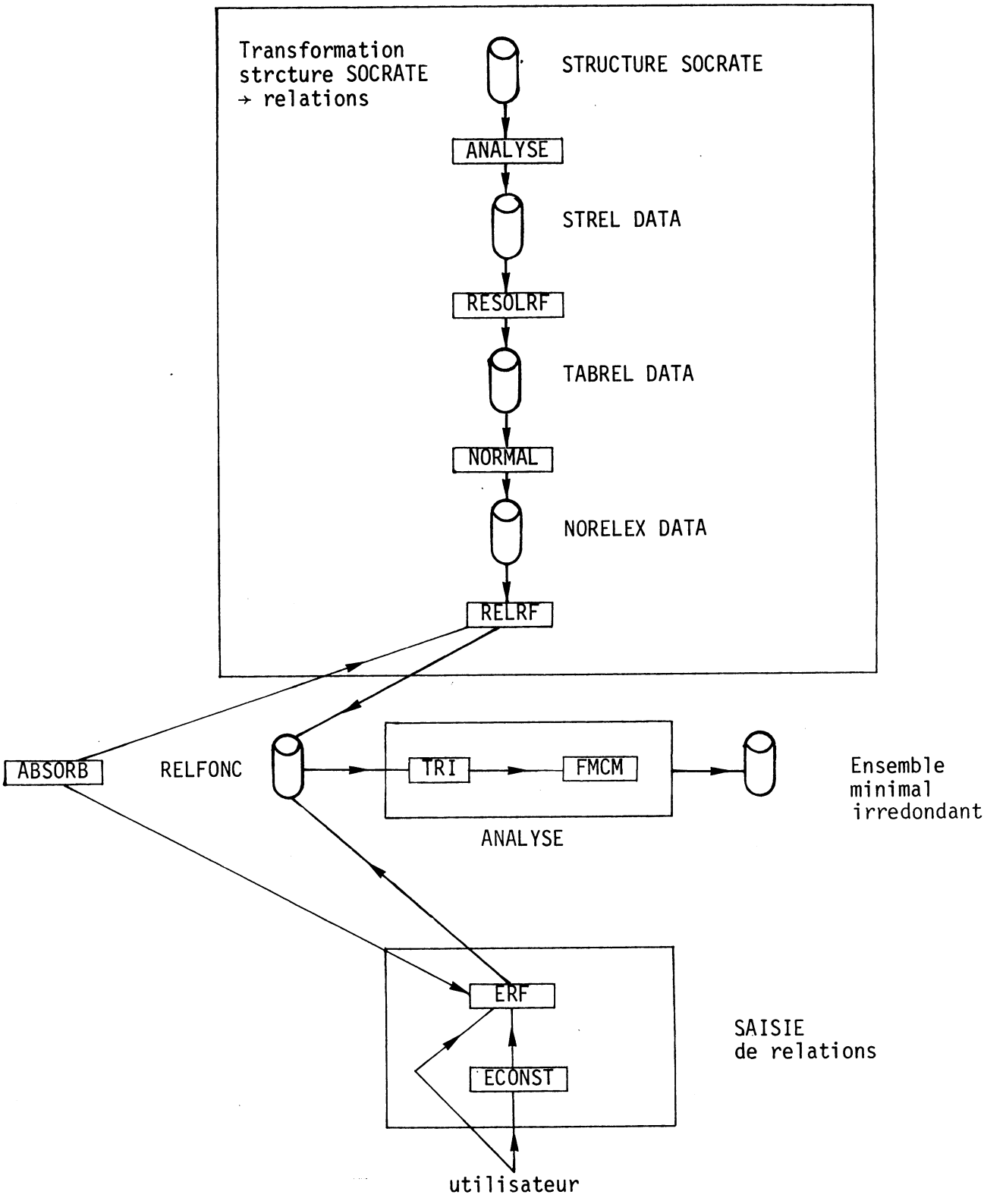


Fig. 6.1 - Organigramme de notre réalisation

6.1. TRANSFORMATION D'UNE STRUCTURE SOCRATE EN UN ENSEMBLE DE RELATIONS

Rappel : Les notions relatives à SOCRATE sont présentées au § 2.2.2.

6.1.1. Hypothèses préalables

Comme nous n'avons pas cherché à construire dans un premier temps le système informatique complet correspondant à une telle transformation, nous nous sommes placés dans des conditions restrictives quant aux structures SOCRATE à analyser :

- une entité n'a pas plusieurs index
- les constituants de même signification, répartis dans des entités différentes, ne sont pas considérés comme équivalents
- aucune entité ne possède de constituants index dans une autre entité, excepté les entités englobées.

Ces limitations ne sont pas gênantes dans la mesure où d'un point de vue pratique, elles correspondent à 90 % des cas réels.

La première restriction évite la prise en compte de circuits dans l'ensemble des relations traduites. En effet, ce point ne nous intéresse pas encore puisque l'algorithme de fermeture-couverture minimale n'est opérationnel que dans le cas d'absence de circuits.

Les deux autres restrictions évitent les interactions avec l'utilisateur qui autrement seraient nécessaires. Ce point ne nous intéresse pas à ce stade car il ne masque que les problèmes suivants :

. la prise en compte par le système de l'équivalence de constituants

. le tri des entités dans le cadre de la détermination des constituants index d'une entité n'y appartenant pas. Ceux-ci ne peuvent être déterminés dans le cas de liaisons de type REFERENCE que par l'utilisateur : il s'agit alors de présenter les entités concernées dans un ordre tel qu'aucun retour en arrière ne soit nécessaire. Ainsi si l'entité E_1 fait référence à l'entité E_2, l'utilisateur doit déterminer l'index de E_2 avant celui de E_1

dans le cas où E_1 contiendrait des constituants index qui appartiennent à E_2 (§ 2.2.2.b).

Le premier point revient à construire un mécanisme d'équivalence de constituants ; le second point revient à utiliser l'algorithme de tri que nous présentons au § 6.3.

6.1.2. Etape n° 1 : Analyse de la structure SOCRATE

Procédure principale ANALYSE.

Fichier d'entrée

La structure source doit se trouver dans un fichier CMS de type DATA dont le nom est précisé au moment de l'exécution ; à chaque enregistrement de ce fichier correspond une déclaration de la structure SOCRATE.

La structure source ne diffère d'une structure compilable par le système SOCRATE par le seul fait que l'utilisateur *doit* préciser l'index de chaque ENTITE au moyen de l'option DISCR prévue dans SOCRATE.
Ainsi tous les traitements ultérieurs se feront sans l'intervention de l'utilisateur.

Traitement d'analyse

. *L'analyse d'une ligne source* SOCRATE consiste à extraire les informations suivantes :

- le nom symbolique choisi par l'utilisateur SOCRATE pour identifier un élément de sa structure (entité, constituant, référence, ...)
- le type de cet élément qui est indiqué par un mot-clé SOCRATE
- les noms symboliques des entités référencées ou inversées.

On distingue six types différents, définis par le tableau ci-dessous :

N°	Type	Symbole Mnémonique	Mot-clé SOCRATE	Exemple
1	Entité	E	ENTITE	ENTITE 100 PERSONNE
2	Groupe	G		ADRESSE DEBUT
3	Constituant simple	C	MOT, TEXTE Liste de valeurs	NOM MOT AGE DE 0 à 99 SEXE (F M)
4	Référence	R	REFERENCE	AUTEUR REFERENCE UN ECRIVAIN
5	Inverse	I	INVERSE	LIVRES INVERSE TOUT LIVRE
6	Constituant discriminant	K	MOT DISCR	NO-INSEE MOT DISCR

Figure 6.2

. Outre les mots-clés définis ci-dessus, la procédure d'analyse recherche 'DEBUT' et 'FIN' pour caractériser les *niveaux des entités*. Ces niveaux sont numérotés de la manière suivante :

- au début, l'entité la plus externe est au niveau zéro. Ses constituants sont au niveau 1 et ainsi de suite, jusqu'à la rencontre d'un FIN qui fait retrancher 1 au niveau ;
- à chaque fois que l'on descend d'un cran, les informations relatives au niveau précédent sont empilées.

. *Les constructions syntaxiques* reconnues *valides* d'une structure SOCRATE sont les suivantes :

- ENTITE Nombre Nom
- Nom Symbolique { MOT / TEXTE / DE n A p / ( ) }
- Nom Symbolique MOT DISCR

- Nom { REFERENCE / INVERSE } { UN / TOUT } NOM DE NOM DE ..... (jusqu'à 4 noms)

- Nom Symbolique DEBUT (Groupe non répétitif)

Dans le cas d'une erreur, il y a impression au terminal de la ligne "erronnée" qui est ignorée.

. Chaque élément de la structure reconnu valide donne lieu à *l'élaboration d'un code spécial* sur 12 caractères appelé par la suite code interne.

Notons xxxxNyytzzzz ce code à 12 caractères :

xxxx : numéro de 4 chiffres qui indique le numéro d'ordre de l'élément dans le fichier résultat.
Ce numéro est utilisable pour caractériser de façon non ambiguë l'élément correspondant. D'autre part, il va servir à accéder directement aux autres informations de l'élément par accès direct au fichier résultat.

N : lettre 'N' qui indique que les 2 chiffres yy qui suivent constituent le numéro de niveau.

yy : 2 chiffres désignant le numéro de niveau. Rappelons que les entités les plus externes ont le niveau 0 et qu'à chaque nouvelle entité ou nouveau groupe ce numéro est augmenté de 1. Il est diminué de 1 à chaque FIN rencontré.

t : 1 caractère qui indique le type de l'élément. C'est l'un des caractères qui figurent dans la colonne notée : symbole mnémonique du tableau du § II.2.

zzzz : nombre de 4 chiffres qui constitue un numéro de séquence à l'intérieur d'un niveau, dans un groupe ou dans une entité. Cependant, ce numéro va également servir à conserver le numéro de l'entité qui correspond à une référence ou à un inverse.

Exemples

- Entité 100 personne au niveau 0 donne :

0001 N00 E 0001

- Une référence, la 3ème, dans une entité au niveau 2 :

0025 N02 R 0003

Fichier de sortie

Le résultat de l'analyse est un ensemble d'enregistrements rangés dans le fichier 'STREL DATA'.

Dans chaque enregistrement figurent deux parties :

1) le code interne sur 12 caractères (colonne 1 à 12)
2) les noms symboliques (tronqués à 8 caractères) et le nombre de noms symboliques.

Exemple

entité 100 personne
0001N00E0001
poste référence un postrav de division de département
0024N02R003 POSTE POSTRAV DIVISION DEPARTEM

6.1.3. Etape n° 2 : Résolution des références et des inverses

Procédure principale : RESOLRF

- Le *but de cette étape* est de préciser pour chaque élément du fichier STREL DATA de type 'R' ou 'I', le code interne de l'entité référencée ou inversée.

- Le *fichier d'entrée* de cette étape est le fichier précédent : STREL DATA où seuls les codes internes des éléments sont en fait utiles.

- Le *traitement de la procédure RESOLRF* se décrit de la manière suivante : on dispose en entrée d'un enregistrement de STREL DATA comprenant :

- le code interne xNyRz par exemple
- le nom par exemple de la référence, disons nom_1
- le nom de l'entité référencée ou inversée, nom_2
- les noms qui permettent de caractériser l'accès à cette entité : nom_3, nom_4, etc ...

Exemple 6.3

Poste référence un postrav de division de département a donné :

0025N02R003	poste	postrav	division	département
code	nom1	nom2	nom3	nom4

On recherche dans le fichier STREL l'entité de nom = nom_4, puis celle de nom_3, etc ..., jusqu'à la rencontre de celle de nom_2. Cette dernière a un code dans lequel figure, dans les quatre premiers caractères, un numéro de 4 chiffres, disons p. Ce numéro vient modifier le code interne xNyRz de la manière suivante : xNyRp.
Ce code est alors rangé dans le fichier de sortie.

Le fichier de sortie TABREL DATA contient les codes internes des éléments de STREL DATA qui ne sont pas de type 'R' ou 'I' et ceux qui viennent d'être modifiés par RESOLRF.

6.1.4. Etape n° 3 : Extraction des relations

Procédure principale : NORMAL

- Le *but de cette étape* est d'extraire du fichier TABREL DATA les relations traduites dans la structure initiale.
Ces relations proviennent soit des entités de niveau 0,
soit des entités englobées,
soit des inverses et des références.

- Le *traitement de la procédure* NORMAL comprend deux étapes :

. Pour chaque entité de niveau 0, on initialise une relation avec le code de son nom, le ou les constituants discriminants et éventuellement d'autres constituants.

1) pour chaque entité englobée dans la précédente, on initialise une nouvelle relation avec :

1.a) le code du nom de cette entité qui est modifié afin d'y inclure le numéro de la relation mère correspondante

1.b) le (ou les) constituant(s) discriminant(s) de la (ou les) relation(s) de niveau supérieur (procédure INSCLE) qui sont marqués comme constituant(s) discriminant(s) de la relation en cours.

2) tant que le niveau des constituants simples est égal à celui de l'entité, augmenté de 1, on crée de nouveaux constituants dans la même relation.

3) si l'on rencontre un constituant de niveau égal ou inférieur, il faut revenir à l'entité englobante concernée et à la relation associée.

Exemple 6.4 de la structure suivante

```
entité département
    début
    |   nomdep mot discr
    |   fonction texte 2
    |   directeur-dpt référence un employé
    |
    |       entité division
    |           début
    |           |   nomdiv mot discr
    |           |   date mot
    |           |
    |           |       entité postrav
    |           |           début
    |           |           |   nompost mot discr
    |           |           |   lieu mot
    |           |           fin
    |           fin
    |
    fin
```

Cette première partie du traitement extrait les relations :

département [nomdep, fonction, directeur-département]
division [nomdep, nomdiv, date]
postrav [nomdep, nomdiv, nompost, lieu]

les constituants soulignés représentant les index.

. Reste maintenant à extraire les relations provenant des références et des inverses :

on reparcourt les relations précédemment mises en évidence et à chaque élément de type 'R' ou 'I', on génère une nouvelle relation ayant :

- comme nom, le nom de la référence ou de l'inverse,
- comme constituants, les constituants index des deux relations concernées,
- comme index, l'ensemble de ces constituants s'il s'agit d'un inverse, sinon les constituants index de l'entité qui fait référence (procédure INSCLE).

Exemple 6.5

```
entité livre
    début
    |   codliv mot discr
    |   titre texte
    |   auteur référence un écrivain
    fin
```

```
entité écrivain
    début
    |   nom discr
    |   sexe mot
    |   âge mot
    fin
```

Cette structure correspond aux relations :

livre [codliv, titre]
écrivain [nom, sexe, âge]
auteur [codliv, nom]

- *De tels traitements ont nécessité* l'utilisation du mécanisme *de pile* pour gérer les constituants déterminants des entités englobées (cas 1.b) et les constituants des entités englobantes (cas 3) ; de plus, ils ont nécessité de *représenter les relations sous forme de liste unidirectionnelle*

afin de pouvoir compléter les relations associées à des entités englobantes dans le cas 5 :

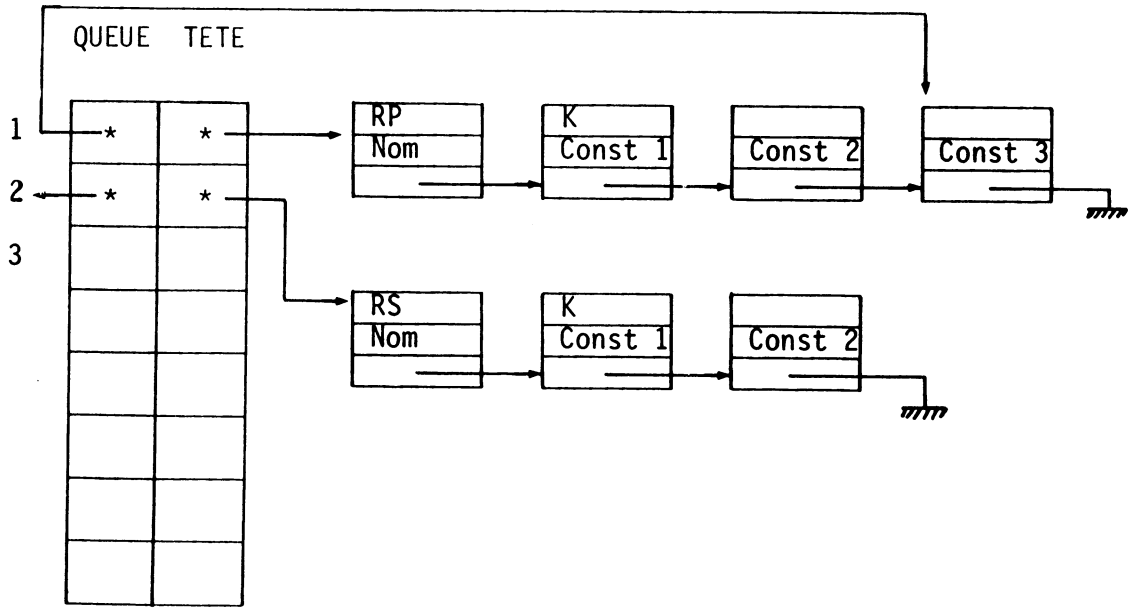


Fig. 6.2 - Représentation des relations sous forme de listes

- *Le fichier sortie* NORELEX DATA contient pour chaque relation :
  - . le(s) code(s) interne(s) et le(s) nom(s) des entités concernées
  - . les codes internes et les noms des constituants de la relation
  - . les constituants-index de la relation.

Chaque relation est enregistrée sous forme d'un tableau.

6.1.5. Exemple 6.6

Structure source

```
printf str2 data

ENTITE 50 DEPARTEMENT
 DEBUT
    NOMDEP MOT DISCR
   DIRECTEUR MOT
   ENTITE 50 DIVISION
    DEBUT
     NOMDIV MOT DISCR
     DIRECDIV MOT
      ENTITE 50 POSTRAV
       DEBUT
        NOMPOSTE MOT DISCR
        LIEU MOT
       FIN
    FIN
 FIN
ENTITE 10000 EMPLOYE
 DEBUT
   MATRICULE MOT DISCR
   NOM MOT
   PRENOM MOT
   ADRESSE TEXTE 3
   DATENAISS
    DEBUT
     JOUR DE 1 A 31
     MOIS DE 1 A 12
     AN DE 0 A 99
    FIN
    ENTITE 100 POSTOCCUP
    DEBUT
     DATE  MOT DISCR
     LIEU MOT
     POSTE REFERENCE UN POSTRAV DE DIVISION DE DEPARTEMENT
      DEPARTEMENT REFERENCE UN DEPARTEMENT
    FIN
FIN
```

Fichier de sortie après l'exécution de RESOLRF

```
printf tabrel data

  1   1N 0E   1   2N 1K   1   3N 1C   1   4N 1E   1   5N 2K   1   6N 2C
  1   7N 2E   1   8N 3K   1   9N 3C   1***********
  2  10N 0E   2  11N 1K   1  12N 1C   1  13N 1C   2  14N 1C   3  15N 1G
  2  16N 2C   1  17N 2C   2  18N 2C   3  19N 1E   1  20N 2K   1  21N 2C
  2  22N 2R0007  23N 2R0001
```

Fichier de sortie après l'exécution de NORMAL

printf norelex data

```
****** RELATION : ( RP )   1N 0E   1  DEPARTEM
( K  )   2N 1K0001  NOMDEP       DEPARTEM
(    )   3N 1C   1  DIRECTEU
****** RELATION : ( RS )   4N 1E0001  DIVISION   DEPARTEM
( K  )   2N 1K0001  NOMDEP       DEPARTEM
( K  )   5N 2K0004  NOMDIV       DIVISION
(    )   6N 2C   1  DIRECDIV
****** RELATION : ( RS )   7N 2E0004  POSTRAV    DIVISION
( K  )   2N 1K0001  NOMDEP       DEPARTEM
( K  )   5N 2K0004  NOMDIV       DIVISION
( K  )   8N 3K0007  NOMPOSTE     POSTRAV
(    )   9N 3C   1  LIEU
****** RELATION : ( RP )  10N 0E   2  EMPLOYE
( K  )  11N 1K0010  MATRICUL     EMPLOYE
(    )  12N 1C   1  NOM
(    )  13N 1C   2  PRENOM
(    )  14N 1C   3  ADRESSE
(    )  15N 1G   1  DATENAIS
(    )  16N 2C   1  JOUR
(    )  17N 2C   2  MOIS
(    )  18N 2C   3  AN
****** RELATION : ( RS )  19N 1E0010  POSTOCCU   EMPLOYE
( K  )  11N 1K0010  MATRICUL     EMPLOYE
( K  )  20N 2K0019  DATE         POSTOCCU
(    )  21N 2C   1  LIEU
****** RELATION : ( RS )  22N 2R0019  POSTE      POSTOCCU
( K  )  11N 1K0010  MATRICUL     EMPLOYE
( K  )  20N 2K0019  DATE         POSTOCCU
(    )   2N 1K0001  NOMDEP       DEPARTEM
(    )   5N 2K0004  NOMDIV       DIVISION
(    )   8N 3K0007  NOMPOSTE     POSTRAV
****** RELATION : ( RS )  23N 2R0019  DEPARTEM   POSTOCCU
( K  )  11N 1K0010  MATRICUL     EMPLOYE
( K  )  20N 2K0019  DATE         POSTOCCU
(    )   2N 1K0001  NOMDEP       DEPARTEM
```

6.2. ANALYSE D'UN ENSEMBLE DE RELATIONS

Procédures principales : FMCM
TRI

6.2.1. Procédure de fermeture et de couverture minimale

- *Le but de cette procédure* est de fournir la fermeture et la couverture minimale d'un ensemble de relations fonctionnelles. Il est construit suivant les principes décrits dans l'annexe A (§ A.3.3).

De plus, cette procédure permet de déterminer l'index et la forme d'une relation dont on connaît les relations fonctionnelles (procédure INDFORM).

- *Hypothèses sur l'ensemble des relations fonctionnelles de départ*

  . Cet ensemble ne doit pas contenir de circuits. Si c'était le cas, l'utilisateur casserait le circuit par l'introduction d'un nom supplémentaire en un point qu'il choisirait.

  . Il ne doit pas contenir de relations fonctionnelles qui peuvent être absorbées (§ A.1.1) par d'autres du même ensemble.

  . Il doit être trié suivant le rang des parties droites des relations fonctionnelles (§ A.1.1 et § A.2.3.1).

- *Paramètres de la procédure*

. L'ensemble trié des relations fonctionnelles, celles dont les parties droites sont de rang 1, précèdent les autres (§ A.2.3.1). Il est représenté à l'aide de deux tableaux de chaînes de bits RFG et RFD : si le constituant de code interne i est concerné, le $i^{ème}$ bit vaut 1. RFG contient les parties gauches, RFD les parties droites.

. Après le traitement, cet ensemble contient la *fermeture*.

. La *couverture minimale* est décelable après l'exécution grâce au tableau CM de bits : si le $j^{ème}$ bit vaut 1, cela signifie que la $j^{ème}$ relation fonctionnelle appartient à la couverture minimale.

. Le nombre de relations fonctionnelles de l'ensemble de départ NBRF qui devient en sortie le nombre de relations de la fermeture.

. La liste des constituants de rang 1 sous forme d'une chaîne de bits.

. Le nombre de constituants NBCONST.

- *Exemples de fonctionnement*

Ce sont ceux que nous avons donnés dans le § A.3.3.2 de l'annexe A.

6.2.2. Algorithme de tri topologique

- *Le but de cet algorithme* est de fournir le rang de chaque constituant à partir de la donnée d'un ensemble de relations fonctionnelles.

La base de cet algorithme est celui donné dans [19] et en plus il s'applique également dans le cas où l'ensemble des relations fonctionnelles contient des circuits. Il détecte les circuits et donne le même rang aux constituants appartenant au même circuit.

- *Paramètres d'entrée*

. Pour chaque constituant, il faut indiquer les constituants CLES et les constituants DEDUITS qui sont définis ainsi :
si dans l'ensemble de départ, il existe la relation fonctionnelle A,B → C, alors : - A et B sont des constituants CLES de C
- C est DEDUIT de A et de B.

Les constituants CLES et les constituants DEDUITS sont rangés dans des tableaux de chaînes de bits. Si le $i^{ème}$ bit est à 1, alors le constituant de code interne i est du type correspondant.

. Le nombre de constituants NBCONST.

- *Paramètres de sortie*

. Pour chaque constituant, on fournit son rang dans le tableau NIV.

. On chaîne les constituants de même niveau au moyen d'une liste unidirectionnelle :
TCONST pour chaque rang fournit le premier constituant.
LIENNIV pour chaque constituant fournit le successeur dans la liste des constituants de même rang.

. Pour chaque circuit, NIVCIRC en donne le rang et CIRCUIT la composition sous forme d'une chaîne de bits.

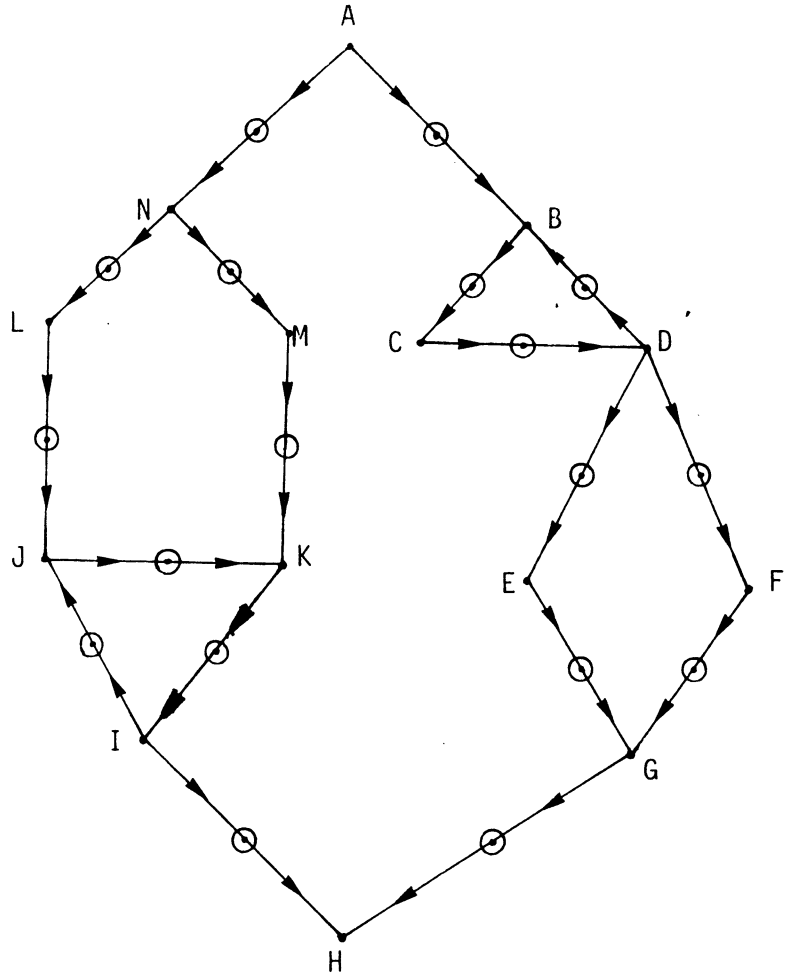
. Pour chaque constituant, CIRC donne le numéro du circuit auquel il appartient (0 s'il n'appartient à aucun d'entre eux).

Exemple 6.7 pris dans le jeu d'essais qui nous a permis de tester notre programme

Soit l'ensemble des relations fonctionnelles :

A → B	A → N
B → C	N → M
C → D	N → L
D → B	M → K
D → E	K → I
D → F	I → J
E → G	J → K
F → G	L → J
G → H	I → H

qui correspond au graphe suivant :



Les circuits sont alors formés des constituants :

I,J,K pour le premier
B,C,D pour le second.

Les rangs des constituants sont alors :

A	5
B	4
C	4
D	4
E	3
F	3
G	2
H	1
I	2
J	2
K	2
L	3
M	3
N	4

6.3. SAISIE D'UN ENSEMBLE DE RELATIONS

2 parties :

- obtention d'un ensemble de relations à partir de l'analyse d'une structure SOCRATE
- obtention d'un ensemble de relations fournies par l'utilisateur.

6.3.1. Obtention d'un ensemble de relations fonctionnelles à partir de l'analyse d'une structure SOCRATE

Procédure : RELRF

- *Le but de cette procédure* est de transformer l'ensemble de relations obtenu après l'analyse d'une structure SOCRATE et contenue dans le fichier NORELEX DATA, dans un ensemble de relations fonctionnelles qui, ensuite, est destiné à être traité par la procédure FMCM.

- *Fonctionnement*

La procédure RELRF transforme les relations du fichier NORELEX en relations fonctionnelles, et ceci de la manière suivante :

. les constituants index sont mis dans une chaîne de bits qui forme la partie gauche RFG de la relation fonctionnelle

. les constituants non index sont regroupés en un constituant global qui forme la partie droite RFD de la relation fonctionnelle.

Ainsi obtient-on deux tableaux RFG et RFD qui, pour chaque relation fonctionnelle, donnent respectivement la partie gauche et la partie droite.

Pour des raisons de commodité d'utilisation, les relations fonctionnelles de même partie droite sont chaînées entre elles. Pour les relations de NORELEX qui proviennent des liaisons inverses de la structure SOCRATE (cf. § 6.1.4), il n'existe pas de constituants non-index : on introduit alors le constituant de champ vide comme partie droite de la relation fonctionnelle associée (cf. [6] § 2.2.3).

Exemple 6.8

Soit la relation [A,B] ne contenant aucune relation fonctionnelle ; elle est transformée en $A,B \rightarrow \lambda$.

De plus, cette procédure élimine les relations fonctionnelles qui peuvent être absorbées par d'autres (procédure ABSORB). Ainsi, l'ensemble des relations fonctionnelles obtenu satisfait-il une des conditions d'application de la procédure FMCM (§ 6.2.1).

Enfin, elle détermine pour chaque constituant les CLES et DEDUITS (cf. § 6.2.2).

- Ainsi, RELREF *permet ensuite l'analyse de cet ensemble de relations* par les procédures TRI et FMCM.

6.3.2. Prise en compte d'un ensemble de relations fournies par l'utilisateur

Procédures : ECONST
ERF

- *Le but de ces procédures* est de permettre l'entrée d'un ensemble de constituants et de relations fonctionnelles définies sur ces constituants.

- *ECONST* permet d'entrer les noms des constituants et leur affecte un code interne.

- *ERF* permet d'entrer des relations fonctionnelles.
Elle les transforme en relations fonctionnelles canoniques (§ 1.3.2) (c'est-à-dire à un seul constituant en partie droite, cf. [6]) et les représente en machine de la façon suivante :

. la partie gauche par une chaîne de bits : si le $i^{ème}$ bit vaut 1, cette partie gauche contient le constituant de code interne i

. la partie droite par le code interne du constituant qui la forme

. la procédure ERF élimine les relations qui peuvent être absorbées par d'autres relations fonctionnelles (sous-procédure ABSORB)

. elle construit pour chaque constituant les ensembles CLES et DEDUITS (cf. § 6.2.2) en les représentant en machine par des chaînes de bits.

- Ainsi obtient-on un ensemble qui peut être ensuite analysé par les procédures TRI et FMCM.

6.3.3. **Exemple**

Obtention et analyse des relations fonctionnelles de l'exemple 6.6

```
lexrel

EXECUTION BEGINS...
NBCONST       16
NBRE DE RF DANS FERMETURE          17
****** FORME DE LA RELATION :            1
************ COUVERTURE MINIMALE
MATRICUL-EMPLOYE DATE-POSTOCC ==> LIEU
MATRICUL-EMPLOYE ==> AN
MATRICUL-EMPLOYE ==> MOIS
MATRICUL-EMPLOYE ==> JOUR
MATRICUL-EMPLOYE ==> DATENAIS
MATRICUL-EMPLOYE ==> ADRESSE
MATRICUL-EMPLOYE ==> PRENOM
MATRICUL-EMPLOYE ==> NOM
NOMDEP-DEPARTE NOMDIV-DIVISIO NOMPOSTE-POSTRAV ==> LIEU
NOMDEP-DEPARTE NOMDIV-DIVISIO ==> DIRECDIV
NOMDEP-DEPARTE ==> DIRECTEU
MATRICUL-EMPLOYE DATE-POSTOCC ==> NOMPOSTE-POSTRAV
MATRICUL-EMPLOYE DATE-POSTOCC ==> NOMDIV-DIVISIO
MATRICUL-EMPLOYE DATE-POSTOCC ==> NOMDEP-DEPARTE
*************     INDEX :
MATRICUL-EMPLOYE DATE-POSTOCC
```

Exemple 6.9 de saisie et d'analyse de relations fonctionnelles

Cet exemple provient de l'exemple 3.17 (p. III.27).

Saisie des constituants

```
 leconst
 EXECUTION BEGINS...
RENTREE DES CONSTITUANTS
NOM DE L APPLICATION
_exemple
 CODE DE L APPLICATION
_a1
 DATE
 _01/01/76
NBRE DES CONSTITUANTS EST IL INFERIEUR A 65?
_oui
 DONNER CONSTITUANTS (MAX:8 CARACTERES) SEPARES PAR BLANCS
_s s1

 _c c1

 _h h1

 _p p1

 _r1 d1

 _
```

Saisie des relations fonctionnelles

```
 lerf
EXECUTION BEGINS...
CODE APPLICATION ?
_a1
 RFG ET RFD : 2 ENSEMBLES SEPARES PAR --> ET
COMPOSES DE CODUTIL SEPARES PAR DES BLANCS
....SI PLUSIEURS CODUTIL POUR  RFD  LES METTRE
 OBLIGATOIREMENT SUR 2NDE LIGNE,PRECEDES PAR -->
_s ---@> s1

 _h s --> h1

 _h s p --> p1

 _c h s p --> r1

 _c p  --> d1

 _c --> c1


 AVEZ VOUS TERMINE ? SI NON CONTINUER
_c p --> i s
RETAPER LA R..F.. S.V.P.
_c p --> h

 _c p --> h s


 AVEZ VOUS TERMINE ? SI NON CONTINUER
_
```

Tri des relations fonctionnelles

```
 ltri
 EXECUTION BEGINS...
DONNER LE CODE DE L'APPLICATION
_a1
 VOULEZ VOUS DES RESULTATS SUR CONSOLE ?
_non
```

Fermeture et couverture minimale

```
 lfmcm
 EXECUTION BEGINS...
CODE APPLICATION ?
_a1
 TRACE? 1 DANS FMCM ET EFMCM  2DANS EFMCM
_0
 !NBRE DE RF DANS FERMETURE          11
 11  REL.FONCT.
VOULEZ VOUS LA FMCM SUR CONSOLE?
_oui

 OUI P C ==> D1
 OUI C ==> C1
OUI S H P ==> P1
OUI S H ==> H1
OUI S ==> S1
OUI P C ==> H
OUI P C ==> S
NON P C ==> S1
OUI P C ==> R1
NON P C ==> P1
NON P C ==> H1
VOULER VOUS CONSERVER L APPLICATION ?
_non
```

Les relations fonctionnelles notées OUI sont celles qui appartiennent à la couverture minimale, les autres appartenant à la fermeture.

C H A P I T R E 7

CONCLUSIONS

Notre objectif était de construire des outils algorithmiques d'aide à la conception d'une base de données. Nous nous sommes placés dans trois hypothèses opérationnelles :

1) concevoir une base de données à partir des résultats de l'analyse du champ d'application.
2) concevoir une nouvelle base de données à partir d'une ancienne déjà existante, mais devenue inadaptée à cause de l'évolution de l'environnement.
3) concevoir une nouvelle base de données, fusion de plusieurs bases de données existantes.

Pour atteindre un tel objectif, nous avons commencé par prendre en compte les propriétés sémantiques qui relient les informations de la base entre elles. Notre première préoccupation a donc été de choisir une méthode pour modéliser le champ d'application de la base traduisant ces propriétés sémantiques. Celle proposée par les systèmes de gestion de bases de données existant s'appuie sur une modélisation des informations sous forme de réseau ou sous forme arborescente.

Nous avons montré dans le chapitre 2 qu'une telle modélisation n'est pas suffisamment précise pour décrire les propriétés sémantiques. Par contre, le modèle relationnel proposé dans [9] et [14] permet justement de les exprimer : aussi l'avons-nous choisi pour modéliser le champ d'application d'une base.

La première partie de notre approche a donc été d'assurer la correspondance entre ces deux types de modélisation (relationnelle et sous forme réseau ou arborescente (Chapitres 2 et 5)). Dans le Chapitre 6, nous avons donné une première réalisation de la transformation d'un modèle réseau dans un modèle relationnel.

Ensuite nous avons pu

1) tenir compte des règles d'intégrité : les chapitres 3 et 4 ont montré le rôle fondamental qu'elles tiennent dans la conception et la restructuration d'une base. Il est important de remarquer à ce propos que les résultats obtenus sont dus au fait qu'aussi bien les règles d'intégrité que les éléments du modèle relationnel s'expriment par des notions de même nature mathématique : ainsi leurs interactions ont pu être facilement étudiées.

2) mettre en évidence des risques d'imperfections de fonctionnement dues à la structure elle-même de la base (Chapitre 3) ; nous avons montré comment ces imperfections peuvent être éliminées grâce à la modélisation relationnelle. La réalisation technique correspondante se trouve dans le Chapitre 6 et est justifiée dans l'Annexe A.

3) déterminer des règles de restructuration cohérente d'une base en tenant compte de l'évolution éventuelle des règles d'intégrité (Chapitre 4). Les résultats obtenus ne sont que théoriques et applicables seulement à des structures arborescentes ; leur justification se trouve dans l'Annexe B.

4) déterminer les différentes décisions à prendre pour construire une structure opérationnelle d'une base de données (Chapitre 5). Les premières à prendre en compte proviennent du fait qu'au modèle relationnel associé à la base correspond une classe de structures réseau qui s'expriment en termes du SGBD utilisé. Aussi faut-il en choisir une, et ce choix ne peut être fait que d'après des critères opérationnels en vue d'une utilisation efficace de la base de données.

La toute première décision qui est la plus importante concerne le choix de l'ensemble des relations à "traduire dans la structure" (Chapitre 5), c'est-à-dire celles qui seront directement prises en compte. Nous avons proposé pour guider ce choix de considérer en premier lieu l'ensemble de relations qui ne conduit à aucune imperfection de fonctionnement dans la structure (Chapitre 3). Il est certain qu'il ne peut convenir pour construire une structure opérationnel puisqu'il ne contient aucune redondance. Il a simplement pour but de fournir une première trame à partir de laquelle le concepteur de la base de données va pouvoir introduire de la redondance tout en se rendant compte des imperfections qu'il risque de causer et des

mécanismes de contrôle qu'il doit veiller à implanter. La difficulté de la tâche du concepteur réside justement en ce point, à savoir de trouver un compromis entre satisfaire les besoins des futurs utilisateurs par la création de chemins d'accès redondants et ne pas introduire trop de mécanismes de contrôle qui risquent d'alourdir le fonctionnement de la base.

Tous les résultats montrent bien *l'importance de l'étude des liens sémantiques* dans la conception d'une base de données. Ils nous conduisent à proposer une démarche en trois étapes pour la conception d'une base de données :

1 - détermination du modèle relationnel du champ d'application de la base, soit à partir des résultats d'analyse exprimés en termes relationnels, soit à partir du ou des modèles, exprimés sous forme de réseau ou sous forme arborescente.

2 - modification du modèle relationnel pour tenir compte de l'évolution des besoins des futurs utilisateurs ou des règles d'intégrité de l'environnement.

3 - traduction des relations retenues en une structure exprimée dans le langage de définition de données du système de gestion de base de données utilisé.

Au-delà des résultats présentés, nous devons bien constater que notre démarche se situe exclusivement au niveau d'une analyse de caractéristiques statiques d'une base de données et que, par ailleurs, face à l'ensemble des problèmes posés par la mise en oeuvre d'une base de données ou l'amélioration des performances des services rendus d'une base existante, nous n'avons pris en compte que les caractéristiques de la base elle-même, laissant complètement de côté les caractéristiques des entrées-sorties sur la base.

Le prolongement de notre travail nous semblerait devoir suivre les deux objectifs suivants :

a) prise en compte de certaines difficultés liées aux caractéristiques dynamiques dans l'utilisation d'une base : nous pensons par exemple à :

(1) la reconfiguration nécessaire d'une base, compte

tenu de l'obsolescence inévitable de certaines données et du rajout d'autres.

(2) un outil d'analyse automatique pour mesurer l'évolution des accès dans la base en fonction de l'évolution de son exploitation, et pour déterminer les dates auxquelles il est opportun d'envisager une restructuration de la base.

(3) des critères nécessaires pour établir certaines redondances obligatoires dans une base quand plusieurs programmes parallèles veulent se partager les mêmes données.

Les trois problèmes cités ci-dessus ne sont pas la liste exhaustive de ceux qui nous amènent à souhaiter la prise en compte du paramètre temps dans des modèles plus évolués que le modèle relationnel pour représenter une base de données. Ils sont cependant suffisamment critiques pour justifier l'urgence d'un tel domaine de recherche.

b) dans toute notre démarche, nous avons parlé des besoins des utilisateurs sans chercher à les préciser. Il serait maintenant important de les prendre en compte et dans ce but d'étudier certaines extensions cohérentes des modèles de structures de bases de données à la représentation de ces besoins, c'est-à-dire des requêtes entrée-sortie de la base. Le but d'un tel prolongement serait alors de concilier les démarches de conception d'une base de données à partir de l'étude des liens sémantiques, de celles qui partent de l'étude des requêtes (cf. GERRITSEN)*.

Face à l'importance des travaux qui restent à faire dans le domaine de la représentation des systèmes de bases de données au seul niveau de la formalisation des données et des programmes, nous hésitons à faire part au lecteur d'autres difficultés que nous avons pu constater expérimentalement et qui nous semblent tout aussi importantes que celles mentionnées précédemment. Elles relèvent du choix fondamental qui consiste à décider si tel ensemble de données doit être projeté sur une base unique ou plusieurs bases indépendantes de fonctionnement

---

* *GERRITSEN, Rob, "Understanding Data Structures", PhD Thesis, Carnegie Mellon University, Pittsburg, Pennsylvania, 1975.*

mais cohérentes dans leur conception. Pour permettre qu'un choix réel existe entre ces deux possibilités, il faut résoudre non seulement des problèmes sémantiques et d'aspect système, mais également des problèmes de coordination de données, de cohérences d'actions entreprises sans oublier les problèmes essentiels d'organisation administrative et de gestion.

Qu'il me soit permis de souhaiter que de tels projets puissent être entrepris, et si le lecteur est intéressé par de tels domaines et se sent un certain goût du risque, nous l'encourageons bien volontiers.

A N N E X E A

ALGORITHME DE FERMETURE-COUVERTURE MINIMALE

S O M M A I R E

A.1. NOTATIONS, DEFINITIONS ET RESULTATS DE BASE

A.1.1. Notations et définitions

Soit E un ensemble de relations fonctionnelles définies sur l'espace de constituants Z.

. Notations

Soit R_a une RF de E.
Nous notons d(a) le constituant de sa partie droite et g(a) l'ensemble des constituants de sa partie gauche.

$\overline{d(a)}$ désigne Z-{d(a)}.

. A E nous pouvons associer un *graphe de noeuds et d'étoiles* :
les étoiles représentant les constituants, les noeuds les RF.

Exemple A.1

$$Z = \{A_1, A_2, B, C, X\}$$

$E = \{A_1, A_2 \rightarrow C \ ; \ A_1, C \rightarrow B \ ; \ A_2, B \rightarrow X \ ; \ X \rightarrow C\}$;

le graphe de noeuds et d'étoiles associé à E est le suivant :

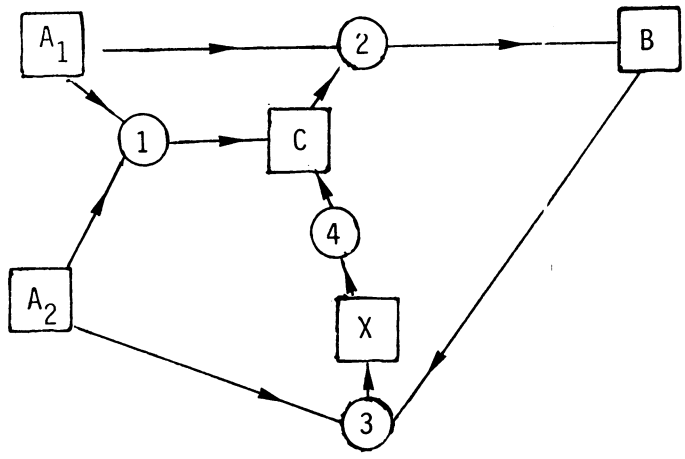


Fig. A.1

. E est dit sans-circuit si le graphe de noeuds et d'étoiles associé est sans-circuit.

Dans l'exemple précédent, il y a un circuit dans le réseau :
C, 2, B, 3, X, 4, C qui correspond au circuit dans E formé par :
A_1,C → B ; A_2,B → X ; X → C.

Dans toute cette annexe, nous supposons que *E est sans-circuit.*

. Absorption d'une RF par une autre RF

R_a absorbe R_b si et seulement si :

$$\begin{cases} d(a) = d(b) \\ g(a) \subset g(b) \end{cases}$$ inclusion ensembliste ;

nous notons alors $R_a \supset R_b$ (en accord avec les notations d'algèbre de Boole).

. Dans toute l'annexe A, nous supposerons E *dépourvu de RF pouvant être absorbées* par d'autres RF de E.

. Les différentes propriétés des RF ont été rappelées dans le chapitre 1.

. Rang d'un constituant

Définition

Comme le graphe de noeuds et d'étoiles représentant l'ensemble E de relations fonctionnelles est sans circuit, il est possible de définir une fonction de rang sur l'ensemble des constituants :

* un constituant A est de rang j (j > 1) si et seulement si :
  - . il existe obligatoirement un constituant B de rang (j-1) tel que A,X → B est une relation de E, X étant un constituant éventuellement vide.
  - . il n'existe aucun constituant C de rang k, k > j, tel que A,X → C appartiendrait à E.

* un constituant A est de rang 1 s'il n'existe aucun constituant C tel que A,X → C appartiendrait à E.

Exemple A.2

Soit Z = {A,B,C,H,I,K,L,M}
Soit E = {A,B → C ; M → L ; K,L → B ; H,I → A ; B → H ; L → C ; K → H}

et le graphe de noeuds et d'étoiles correspondant :

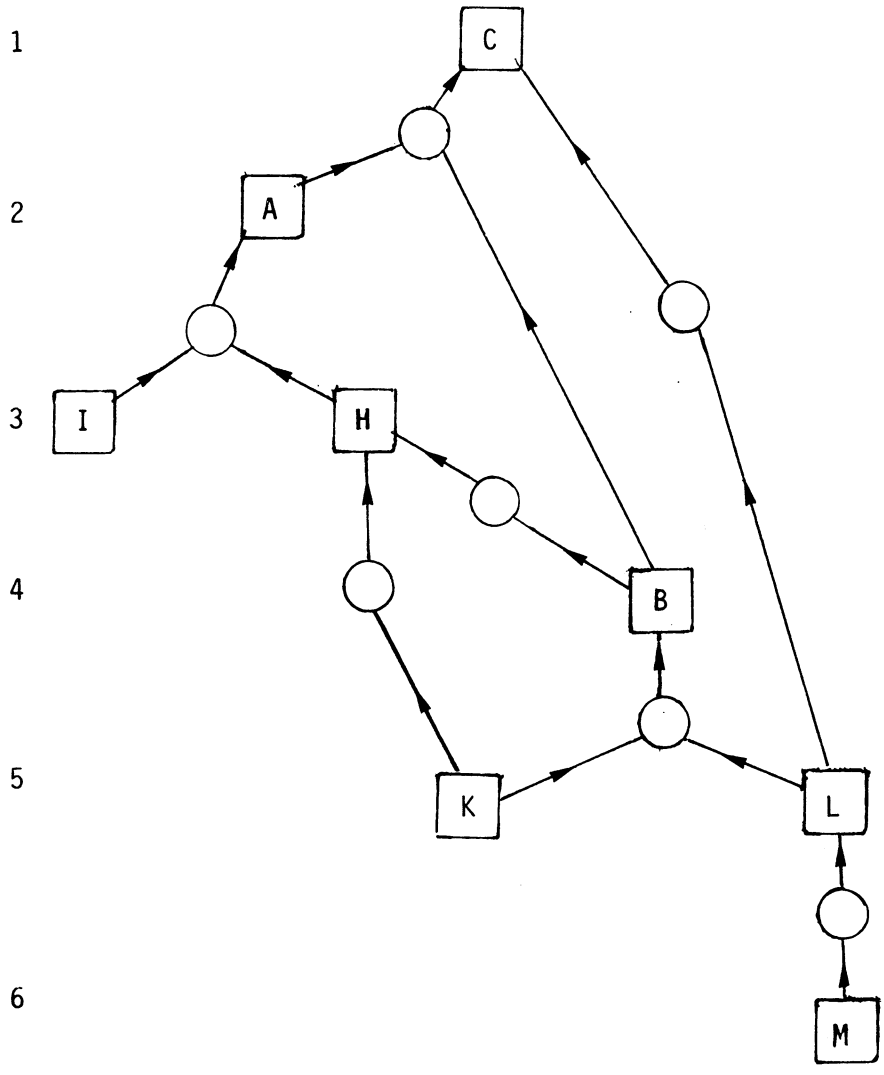


Fig. A.2

Les constituants ont les rangs indiqués sur la figure A.2.

A.1.2. Opérateur +> de pseudo-transitivité

A.1.2.1. Définition (§ 1.3.5)

Cet opérateur +> définit une opération interne sur l'ensemble des RF, en faisant correspondre à deux RF satisfaisant aux conditions de *pseudo-transitivité* la RF obtenue à partir d'elles par pseudo-transitivité.

Soient Ri et Rj telles que $d(i) \wedge g(j) \neq \emptyset$
alors Rk = Ri +> Rj signifie que :

- $g(k) = g(i) \vee (g(j) \wedge \overline{d(i)})$
- $d(k) = d(j)$.

Exemple A.3

Ri = (A,B → C)
Rj = (C,D → E) → Rk = (A,B,D → E)

On note : Rk = Ri +> Rj.

A.1.2.2. Propriétés de l'opération +>

A.1.2.2.1. Elle est non commutative

En effet, les indices i et j jouent des rôles non symétriques dans la définition de l'opération +>.

A.1.2.2.2. Elle est non associative

Soit Rm = Ri +> (Rk +> Rℓ)
avec Rj = Rk +> Rℓ

a) Rm = Ri +> Rj

(1) $d(i) \wedge g(j) \neq \emptyset$
(2) $g(m) = g(i) \vee (g(j) \wedge \overline{d(i)})$
(3) $d(m) = d(j)$

b) $Rj = Rk \mathbin{+\!\!>} R\ell$

(4) $d(k) \wedge g(\ell) \neq \emptyset$

(5) $g(j) = g(k) \vee (g(\ell) \wedge \overline{d(k)})$

(6) $d(j) = d(\ell)$

1) Premier cas : $d(i) \wedge g(k) \neq \emptyset$

On peut former $Ri \mathbin{+\!\!>} Rk$

Soit $Rh = Ri \mathbin{+\!\!>} Rk$

(7) $g(h) = g(i) \vee (g(k) \wedge \overline{d(i)})$

(8) $d(h) = d(k)$

(9) $d(i) \wedge g(k) \neq \emptyset$

Comme $d(k) \wedge g(\ell) \neq \emptyset$ (4), on peut former :

$Rf = Rh \mathbin{+\!\!>} R\ell$

(10) $d(h) \wedge g(\ell) \neq \emptyset$

(11) $g(f) = g(h) \vee (g(\ell) \wedge \overline{d(h)})$

(12) $d(f) = d(\ell)$

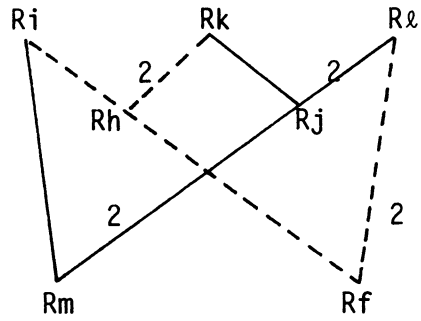


Fig. A.3

Si l'opération est associative, on doit avoir :

$$\underline{Rm = Rf}$$

(11) peut s'écrire à l'aide de (7) et (8)

$$\begin{aligned} g(f) &= g(h) \vee (g(\ell) \wedge \overline{d(h)}) \\ &= g(i) \vee (g(k) \wedge \overline{d(i)}) \vee (g(\ell) \wedge \overline{d(h)}) \\ &= g(i) \vee (g(k) \wedge \overline{d(i)}) \vee (g(\ell) \wedge \overline{d(k)}) \end{aligned}$$

(2) peut s'écrire à l'aide de (5) :

$$\begin{aligned} g(m) &= g(i) \vee (g(j) \wedge \overline{d(i)}) \\ &= g(i) \vee ((g(k) \vee (g(\ell) \wedge \overline{d(k)})) \wedge \overline{d(i)}) \\ &= g(i) \vee (g(k) \wedge \overline{d(i)}) \vee (g(\ell) \wedge \overline{d(k)} \wedge \overline{d(i)}) \end{aligned}$$

- Si $d(i) \wedge g(\ell) = \emptyset$

$$g(\ell) \wedge \overline{d(k)} \wedge \overline{d(i)} = g(\ell) \wedge \overline{d(k)}$$

et alors $g(m) = g(f)$

et $\underline{Ri \mathbin{+>} (Rk \mathbin{+>} R\ell) = (Ri \mathbin{+>} Rk) \mathbin{+>} R\ell}$

et $\underline{Rm = Rf}$

Exemple A.4

$Ri = (A,B \rightarrow C)$

$Rk = (C,D \rightarrow E)$

$R\ell = (E,F \rightarrow G)$

$\Longrightarrow \quad d(i) \wedge g(\ell) = \emptyset$

$Rh = Ri \mathbin{+>} Rk = (A,B,D \rightarrow E)$

$Rj = Rk \mathbin{+>} R\ell = (C,D,F \rightarrow G)$

$Rm = Ri \mathbin{+>} Rj = (A,B,D,F \rightarrow G)$

$Rf = Rh \mathbin{+>} R\ell = (A,B,D,F \rightarrow G)$

- Si $d(i) \wedge g(\ell) \neq \emptyset$

$(g(\ell) \wedge \overline{d(k)} \wedge \overline{d(i)}) \subset g(\ell) \wedge \overline{d(k)}$

$g(m) \subset g(f)$

et ainsi Rm absorbe R .

Exemple A.5

Ri = (A,B → C)
Rk = (C,D → E)
Rℓ = (E,C → G)

D'une part :

Rh = Ri +> Rk = (A,B,D → E)
Rf = Rh +> Rℓ = (A,B,D,C → G)

D'autre part :

Rj = Rk +> Rℓ = (C,D → G)
Rm = Ri +> Rj = (A,B,D → G).

Rm absorbe Rf car $d(i) \wedge g(\ell) = (C) \wedge (E,C) = (C) \neq \emptyset$.

2) Deuxième cas : $d(i) \wedge g(k) = \emptyset$

On ne peut former Ri +> Rk ; alors Rm existe mais Rf n'existe pas.

3) Rm existe si Rf existe

Preuve : Rf existe (10) $d(h) \wedge g(\ell) \neq \emptyset$
(9) $d(i) \wedge g(k) \neq \emptyset$

Comme d'après (8) d(h) = d(k), (10) devient $d(k) \wedge g(\ell) \neq \emptyset$;
on peut donc former Rk +> Rℓ = Rj avec $g(j) = g(k) \vee (g(\ell) \wedge \overline{d(k)})$;
comme d'après (9) $d(i) \wedge g(k) \neq \emptyset$ et que $g(j) \supseteq g(k)$, $d(i) \wedge g(j) \neq \emptyset$
et on peut former

Rm = Ri +> Rj.

A.1.2.2.3. L'opération +> est donc non associative

Nous pouvons récapituler les résultats précédents dans le tableau suivant, avec les mêmes notations :

Conditions	$d(i) \wedge g(k) = \emptyset$	$d(i) \wedge g(k) \neq \emptyset$
$d(i) \wedge g(\ell) = \emptyset$	Rm et Rf n'existent pas	Rm = Rf
$d(i) \wedge g(\ell) \neq \emptyset$	Rf n'existe pas	Rm absorbe Rf

Nous appellerons cette propriété : *pseudo-associativité* des RF.

A.2. ALGORITHME DE FERMETURE

La fermeture FM de l'ensemble E de RF est par définition (§ 3.3.1.2) l'ensemble des relations fonctionnelles élémentaires qu'il est possible d'obtenir à partir de E en appliquant les règles de pseudo-transitivité et d'absorption.

. Nous avons adopté les conventions suivantes pour la description des algorithmes :

E désigne l'ensemble des RF de départ
G désigne l'ensemble des RF construites
G1, G2, D désignent des ensembles de RF
Ri désigne une RF.

. De plus, nous définissons l'opération +> sur deux ensembles de RF comme suit :

(E +> G) désigne l'ensemble des RF Ra que l'on peut construire de la manière suivante :

Ra = Rb +> Rc avec $\underline{Rb \in E}$ et $\underline{Rc \in C}$:
si $\emptyset$ désigne l'ensemble vide de RF, on a alors :
$\emptyset = \emptyset$ +> E et $\emptyset$ = E +> $\emptyset$.

. L'expression "nettoyer G et E" signifie l'élimination de toutes les RF de G et de E, qui peuvent être absorbées par des RF de G ou de E ; les duplications éventuelles sont éliminées, étant entendu qu'une RF de E ne peut être éliminée que si elle est absorbée au sens strict.

A.2.1. Algorithmes de fermeture de base

Nous avons montré dans le § 1.3 comment l'obtention de la fermeture d'un ensemble E de RF revient en fait à la recherche de la base complète de la forme relationnelle associée à E.

Aussi nous avons pris comme point de départ l'algorithme donné dans ([20] p. 102) qui fournit l'ensemble des monômes premiers d'une fonction booléenne quelconque.

Avec nos notations, l'algorithme de [20] s'écrit :

Algorithme 1

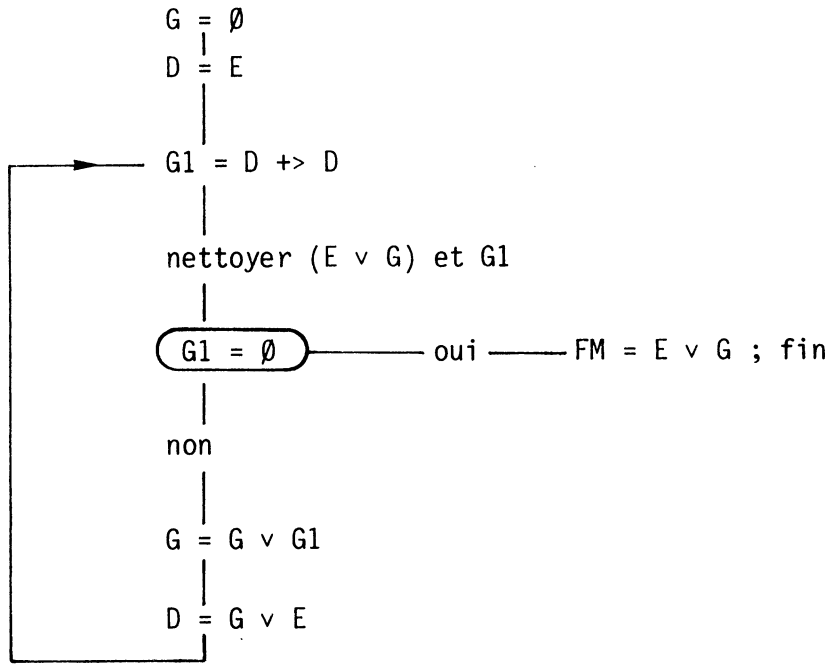


Fig. A.4 - Algorithme 1 de fermeture

A chaque boucle, l'algorithme 1 refait des combinaisons déjà effectuées lors de boucles précédentes, comme par exemple E +> E ; l'algorithme 2 les élimine :

Algorithme 2

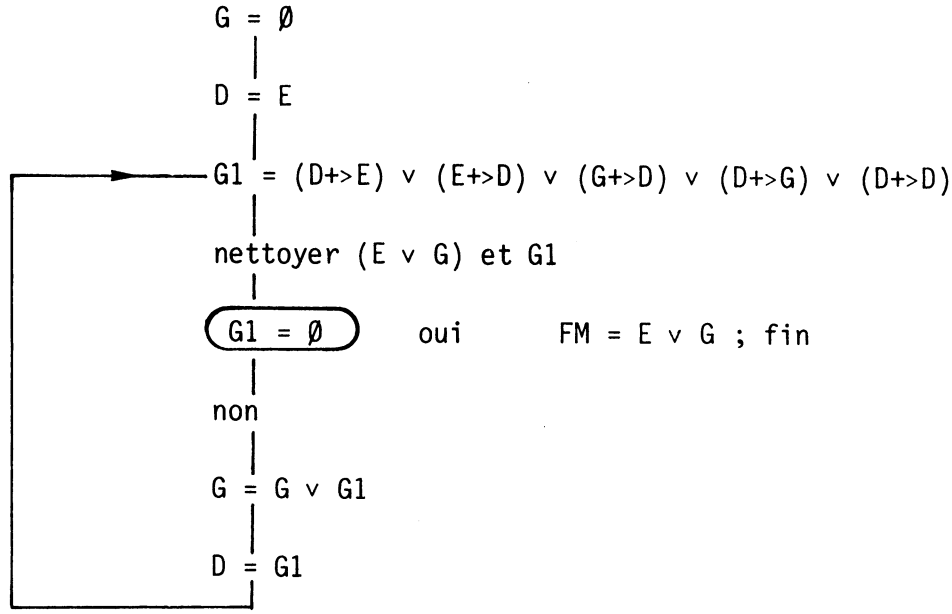


Fig. A.5 - Algorithme 2 de fermeture

Cet algorithme illustre bien les deux mécanismes propres d'un algorithme de fermeture :

- Construction de nouvelles RF grâce à la propriété de pseudo-transitivité (opération +>)
- Elimination de RF grâce à l'opération d'absorption (au sens large qui comprend l'égalité) (opération : "nettoyer").

A.2.2. Propriétés des relations fonctionnelles

A.2.2.1. Proposition 1

Toute RF Ra construite par l'exécution de l'algorithme 2 de fermeture à partir de E peut l'être en particulier par :

(1) Ra = Rb +> Rc avec Rb ϵ E.

Exemple A.6

Soit E : $\begin{cases} Rd = (A,B \rightarrow C) \\ Rb = (A \rightarrow B) \\ Re = (C \rightarrow D) \end{cases}$

La RF Ra = (A → D) peut être construite à partir de :

Rf = Rb +> Rd = (A → C)

et Ra = Rf +> Re, mais avec $Rf \in G$;

mais Ra peut être également construite par :

Rc = Rd +> Re = (A,B → D)

et Ra = Rb +> Rc = (A → D) avec $Rb \in E$.

Démonstration

Supposons que Ra soit construite lors de la $n^{ème}$ boucle par :

(2) $Ra = R_1 +> R_2$

Si $R_1 \in E$, la démonstration est terminée ;
si $R_1 \notin E$, c'est-à-dire $R_1 \in G$:

R_1 a été construite lors d'une boucle précédente par :

(3) $R_1 = R_3 +> R_4$;

en combinant (2) et (3), on trouve :

(4) $Ra = (R_3 +> R_4) +> R_2$.

D'après la pseudo-transitivité de l'opérateur +>, et comme Ra appartient effectivement à la fermeture, on peut écrire :

(5) $Ra = R_3 +> (R_4 +> R_2)$.

De deux choses l'une :

ou bien $R_3 \in E$:

dans ce cas en prenant $Rb = R_3$ et $Rc = R_4 +> R_2$, on trouve bien : Ra = Rb +> Rc avec $Rb \in E$; la démonstration est terminée

ou bien $R_3 \notin E$, c'est-à-dire $R_3 \in G$;
alors on recommence le raisonnement précédent mais en remarquant que comme R_3 sert à la construction de R_1 et donc de Ra, R_3 a été construite lors de la $m^{ème}$ boucle avec $m < n$; cette remarque assure que cette démarche a une fin et qu'elle se termine par l'obtention de Rb et de Rc.

A.2.2.2. Proposition 2

Soient deux RF de E : Ra et Rb telles que :

(1) $d(a) \wedge g(b) = \emptyset$;

alors si Rabc = Ra +> (Rb +> Rc) existe, elle est soit égale ou absorbée par Rac = Ra +> Rc, soit égale à Rbac = Rb +> (Ra +> Rc).

Exemple A.7

* Soit E : $\begin{cases} Ra = (B,D \rightarrow A) \\ Rb = (C \rightarrow B) \\ Rc = (A,B \rightarrow E) \end{cases}$

$d(a) \wedge g(b) = \emptyset$

Rabc = Ra +> (Rb +> Rc) = (B,D → A) +> (A,C → E)
= (B,C,D → E)

Or, Rac = Ra +> Rc = (B,D → A) +> (A,B → E)
= (B,D → E)

Rac absorbe Rabc.

* Soit E : $\begin{cases} Ra = (D \rightarrow A) \\ Rb = (C \rightarrow B) \\ Rc = (A,B \rightarrow E) \end{cases}$

$d(a) \wedge g(b) = \emptyset$

Rabc = (D → A) +> ((C → B) +> (A,B → E)) = (D → A) +> (A,C → E) = (C,D → E)

Rbac = (C → B) +> ((D → A) +> (A,B → E)) = (C → B) +> (B,D → E)
= (C,D → E) = Rabc.

Démonstration

. Etablissons en premier lieu les conditions liées à l'existence de Rbc et Rabc

(2) $d(b) \wedge g(c) \neq \emptyset$

(3) $d(a) \wedge g(bc) \neq \emptyset$

Comme $g(bc) = [g(b) \vee g(c)] \wedge \overline{d(b)}$, (3) devient :

(4) $(d(a) \wedge g(b) \wedge \overline{d(b)}) \vee (d(a) \wedge g(c) \wedge \overline{d(b)}) \neq \emptyset$

Pour que (4) soit vérifiée, il faut que $d(a) \wedge \overline{d(b)} \neq \emptyset$, c'est-à-dire que (5) $d(a) \neq d(b)$.

En supposant cette condition vérifiée, (4) devient :

$(d(a) \wedge g(b)) \vee (d(a) \wedge g(c)) \neq \emptyset$

qui devient d'après (1) :

(6) : $d(a) \wedge g(c) \neq \emptyset$.

Les conditions que sont supposées vérifier Ra, Rb, Rc sont donc :

(1) $d(a) \wedge g(b) = \emptyset$

(2) $d(b) \wedge g(c) \neq \emptyset$

(5) $d(a) \neq d(b)$

(6) $d(a) \wedge g(c) \neq 0$.

. Montrons maintenant la proposition 2

Les parties gauches et droites des relations Rabc, Rac, Rbac s'expriment respectivement :

(7) $g(abc) = (g(a) \vee g(bc)) \wedge \overline{d(a)}$
$= (g(a) \vee g(b) \vee g(c)) \wedge (g(a) \vee \overline{d(b)}) \wedge \overline{d(a)}$

(8) $d(abc) = d(bc) = d(c)$

Comme $d(a) \wedge g(c) \neq \emptyset$, on peut former Rac = Ra +> Rc

avec (9) $d(ac) = d(c)$

(10) $g(ac) = (g(a) \vee g(c)) \wedge \overline{d(a)}$.

Pour que Rbac existe, il faut que

$$d(b) \wedge [(g(a) \vee g(c)) \wedge \overline{d(a)}] \neq \emptyset \ ;$$

ceci est vérifié car $d(b) \wedge \overline{d(a)} = d(b)$ d'après (5)
et $d(b) \wedge g(c) \neq \emptyset$ d'après (2)

(11) $g(bac) = (g(b) \vee g(a) \vee g(c)) \wedge (g(b) \vee \overline{d(a)}) \wedge \overline{d(b)}$

(12) $d(bac) = d(ac) = d(c)$

De deux choses l'une :

- 1er cas : $d(b) \wedge g(a) \neq \emptyset$

alors $\overline{d(b)} \vee g(a) = A$, l'ensemble de tous les constituants et (10) devient $g(abc) = (g(a) \vee g(b) \vee g(c)) \wedge \overline{d(a)}$:

$$g(abc) \subseteq g(ac) \ ;$$

comme $d(abc) = d(ac) = d(c)$, nous en concluons que dans ce cas Rabc est égale ou absorbée par Rac ;

le premier exemple de l'exemple A.7, page A.12, illustre ce cas.

- 2ème cas : $d(b) \wedge g(a) = \emptyset$

Ainsi $g(a) \wedge \overline{d(b)} = \overline{d(b)}$ et (7) devient :

(13) $g(abc) = (g(a) \vee g(b) \vee g(c)) \wedge \overline{d(b)} \wedge \overline{d(a)}$;

(11) devient d'après (1) :

(14) $g(abc) = (g(a) \vee g(b) \vee g(c)) \wedge \overline{d(a)} \wedge \overline{d(b)}$

et donc $g(bac) = g(abc)$.

Ainsi Rabc et Rbac sont identiques.

Le deuxième exemple de l'exemple A.7 (page A.12) illustre ce cas.

. La proposition 2 est ainsi démontrée.

A.2.2.3. **Proposition 3**

Si Ra est absorbée par Rb, alors toutes RF construites à partir de Ra suivant Ra +> Rc ou Re +> Ra sont

soit absorbées par Rb ou les RF construites symétriquement à partir de Rb (Rb +> Rc ou Re +> Rb),

soit égales à celles-ci.

Exemple A.8

* Soit E_1 : $\begin{cases} Ra = (A,B \rightarrow C) \\ Rc = (C \rightarrow D) \\ Rf = (A \rightarrow B) \end{cases}$

Rac = Ra +> Rc = (A,B → D) ;
Rb = Rf +> Ra = (A → C) et absorbe Ra ;
et Rbc = Rb +> Rc = (A → D) absorbe Rac.

* Soit E_2 : $\begin{cases} Ra = (A,B \rightarrow C) \\ Rc = (B,C \rightarrow D) \\ Rf = (A \rightarrow B) \end{cases}$

Rac = Ra +> Rc = (A,B → D)
Rb = Rf +> Ra = (A → C) et absorbe Ra ;
Rbc = Rb +> Rc = (A → C) +> (B,C → D) = (A,B → D).
Dans ce cas, Rac = Rbc.

* Soit E_3 : $\begin{cases} Ra = (A,B \rightarrow C) \\ Rb = (B \rightarrow C) \\ Re = (E \rightarrow B) \end{cases}$

Rea = (E → B) +> (A,B → C)
= (A,E → C)
Reb = (E → B) +> (B → C)
= (E → C)
Reb absorbe Rea.

* Soit E_4 : $\begin{cases} Ra = (A,B \rightarrow C) \\ Rb = (B \rightarrow C) \\ Re = (A,E \rightarrow B) \end{cases}$

$Rea = (A,E \rightarrow B) +> (A,B \rightarrow C)$
$= (A,E \rightarrow C)$
$Reb = (A,E \rightarrow B) +> (B \rightarrow C)$
$= (A,E \rightarrow C)$

Reb et Rea sont identiques.

Démonstration

Rb absorbe Ra $\Longleftrightarrow$ (1) $g(b) \subset g(a)$
$d(b) = d(a)$

- Soit Rac = Ra +> Rc

Si Rac existe, cela signifie que (3) $d(a) \wedge g(c) \neq \emptyset$;
de plus (4) $g(ac) = [g(a) \vee g(c)] \wedge \overline{d(a)} = g(a) \vee [g(c) \wedge \overline{d(a)}]$;
d'après (2), $d(b) \wedge g(c) \neq \emptyset$ et donc

Rbc = Rb +> Rc existe ;

(5) $g(bc) = [g(b) \vee g(c)] \wedge \overline{d(b)}$
$= g(b) \vee [g(c) \wedge \overline{d(b)}]$

Ainsi $g(bc) \subseteq g(ac)$ d'après (1) et (2) ;

comme $d(bc) = d(ac) = d(c)$, Rbc absorbe Rac ou lui est égale.

Dans l'exemple A.8, les cas de E_1 et de E_2 montrent ces deux possibilités.

- Soit Rea = Re +> Ra

. Si Rea existe, cela signifie que : (6) $d(e) \wedge g(a) \neq \emptyset$
de plus : (7) $g(k) = (g(e) \vee g(a)) \wedge \overline{d(e)}$

. De deux choses l'une : ou bien (8) $d(e) \wedge g(b) \neq \emptyset$
ou bien (9) $d(e) \wedge g(b) = \emptyset$;
supposons que (8) soit vraie ;
alors Reb = Re +> Rb existe ;

(10) $g(eb) = [g(e) \vee g(b)] \wedge \overline{d(e)}$
$= g(e) \vee [g(b) \wedge \overline{d(e)}]$;

à cause de (1) $g(eb) \subseteq g(k)$ et comme $d(b) = d(eb) = d(ea)$
ceci signifie que Re absorbe Rk ou lui est égale.

Dans l'exemple A.8 (page A.15), les cas de E_3 et E_4 montrent ces deux possibilités.

. Supposons que (9) soit vraie :
alors (1) devient $g(b) \subset g(a) \wedge \overline{d(e)}$
et donc (7) devient (11) $g(b) \subset g(k)$;
ainsi Rb absorbe Rk.

Nous en concluons que dans tous les cas les RF construites à partir de Ra sont soit absorbées par Rb ou par celles construites à partir de Rb de manière symétrique, soit égales à celles-ci.

A.2.3. Conclusions algorithmiques

Nous allons montrer les conséquences des propositions précédentes d'abord sur les mécanismes de construction, et ensuite sur ceux d'élimination d'un algorithme de fermeture.

A.2.3.1. Processus de construction

. Conséquence de la proposition 1

Une interprétation possible de cette proposition est de dire que l'algorithme 2 construit des RF R de trop : toutes celles dont la formation est du type R1 +> R2 avec $R1 \in G$;
en effet, si R appartient effectivement à la fermeture, elle peut être formée à partir de R = Ra +> Rb avec $Ra \in E$ d'après la proposition 1 ;
de plus, si R n'appartient pas à la fermeture, toutes les RF qui pourraient être construites à partir de R seront absorbées ou dupliquées d'après la proposition 3 ;

aussi peut-on limiter le processus de construction de nouvelles RF au seul type Ra +> Rb avec Ra ϵ E sans perte de relations.

. Conséquences de la proposition 2

Une interprétation de la proposition 2 est de dire que Ra et Rb étant deux RF de E telles que

(1) $d(a) \wedge g(b) = \emptyset$

la construction des RF : Rac = Ra +> Rc ne nécessite la connaissance d'aucune RF construite à partir de Rb dans un algorithme de fermeture : il suffit donc de construire toutes les RF à partir de Ra et ensuite toutes celles à partir de Rb.

Ainsi on établit un classement des RF de E selon la propriété (1) en plaçant Ra avant Rb.

Un classement particulier, plus faible que le précédent, consiste à ordonner les RF de E suivant le rang de leur partie droite : en effet, si Ra et Rb sont telles que

rang (d(a)) $\leq$ rang (d(b)) alors elles vérifient la propriété (1) par définition même de la fonction rang (§ A.1).

Les RF de E sont donc classées :

- suivant le rang de leur partie droite, les premières étant celles dont la partie droite n'est dans aucune partie gauche (rang 1)
- par paquet de RF ayant même partie droite, deux paquets de parties droites de même rang étant placés indifféremment l'un devant l'autre
- à l'intérieur de chacun de ces paquets, de manière indifférente.

Exemple A.9 de RF triées :

En reprenant l'exemple A.2 (page A.3) et les rangs donnés, on peut trier E ainsi :

A,B → C
L → C
H,I → A
B → H
K → H
K,L → B
M → L

. Processus de construction de nouvelles RF

En appliquant les conclusions algorithmiques précédentes, nous obtenons le processus de construction PG suivant :

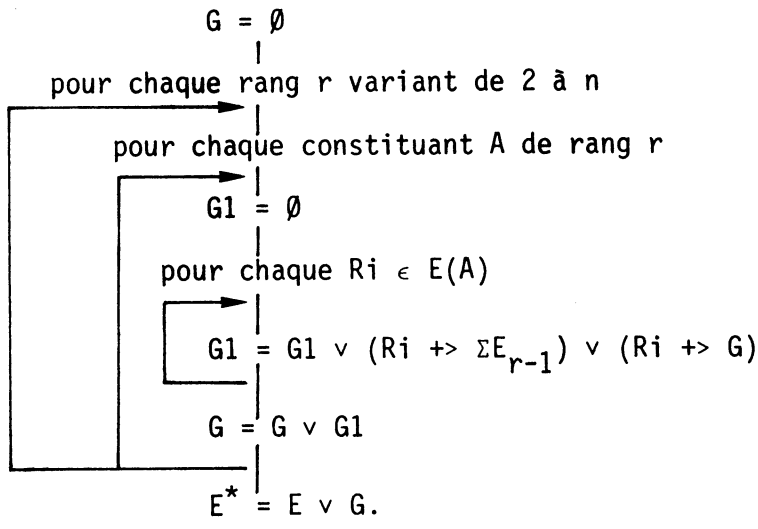


Fig. A.6 - Processus de construction (PG)

où E(A) désigne l'ensemble des RF Ri de E telles que

$$d(i) = A$$

et ΣE_{r-1} désigne l'ensemble des RF Rj de E telles que

$$\text{rang } (d(j)) \leq r-1$$

Propriété 1 de PC

Quand PC s'intéresse à une RF Ri de E, alors il a construit auparavant toutes les RF Rc de la fermeture, telles que l'opération Ri +> Rc est exécutable.

Cette propriété découle de la proposition 1.

A.2.3.2. Processus d'élimination

. Si PC est appliqué à un ensemble de RF E, il fournit comme résultat un ensemble de RF E^* qui contient la fermeture de E : FM. Aussi pour trouver FM, faut-il éliminer de E^* les RF non élémentaires par comparaison deux à deux.

. Pour améliorer l'efficacité de l'algorithme de fermeture, nous allons démontrer la règle suivante :

- dans le contexte de PC, une RF Ra peut être éliminée dès que son absorption par une autre RF Rb est mise en évidence.

Démonstration

Nous allons d'abord montrer que si ces deux opérations sont possibles :

Ra +> Rc
et Re +> Ra,

alors PC associé à la règle précédente construit quand même

Rb +> Rc
et Re +> Rb si $d(e) \wedge g(b) \neq \emptyset$.

(a) Si Ra +> Rc peut être formée, peut-on construire Rb +> Rc avec PC ?

Si $Ra \in G$, PC assure qu'il n'y a pas lieu de former Ra +> Rc pour trouver la fermeture ; ce cas est écarté ;

$Ra \in E$, alors $Rb \in G$ puisque E est supposée ne contenir aucune RF susceptible d'être absorbée par une autre RF de E (cf. A.1.1.) :

Rb = Ri +> Rj avec $Ri \in E$.

PC ne construit pas Rb +> Rc puisque $Rb \in G$; mais d'après la propriété 1 de PC, on sait que (Rj +> Rc) est formée avant que PC ne s'intéresse à Ri : il forme donc Ri +> (Rj +> Rc) qui absorbe (Rb +> Rc) ou lui est égale.

(b) Si Re +> Ra peut être formée et si $d(e) \wedge g(b) \neq \emptyset$, peut-on construire Re +> Rb avec PC ?

- Si $Re \in G$, PC assure qu'il n'y a pas lieu de considérer Re +> Ra pour construire la fermeture.
- Si $Re \in E$, d'après la propriété 1 de PC, l'on sait que Rb est construite avant que PC ne s'intéresse à Re et ainsi que PC construit bien Re +> Rb.

(c) Conclusions

La proposition 3 montre que si Ra est absorbée par Rb, alors les RF construites à partir de Ra (Ra +> Rc et Re +> Ra) sont absorbées par celles construites à partir de Rb (Rb +> Rc et Rb ou Re +> Rb) ou égales à celles-ci.
Ainsi, si PC construisait les premières, celles-ci seraient inutiles dans le processus de fermeture.

A.2.3.3. Algorithme FM de fermeture

Il se déduit des conclusions algorithmiques précédentes et il nécessite pour être appliqué que E, ensemble de départ, soit trié comme nous l'avons indiqué précédemment.

En reprenant les mêmes conventions que dans la Fig. A.6, nous décrivons ainsi l'algorithme FM :

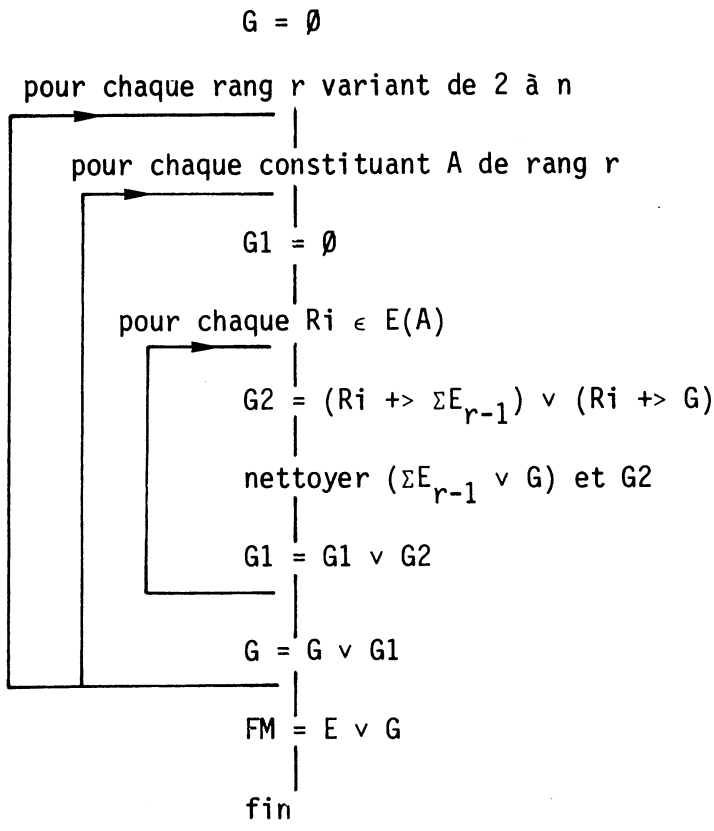


Fig. A.7 - Algorithme FM de fermeture

A.3. ALGORITHME FM-CM DE FERMETURE ET DE COUVERTURE MINIMALE

La couverture minimale CM d'un ensemble E de RF est par définition (§ 3.3.1.2) un ensemble de RF élémentaires tel que la fermeture de cet ensemble est égale à la fermeture de l'ensemble de départ, cette propriété n'étant plus vraie si l'on supprime une RF de la couverture minimale.

Nous avons donné dans le chapitre 3 les raisons qui nous poussent à construire un algorithme qui construit à la fois la fermeture et la couverture minimale d'un même ensemble de départ.

A.3.1. Principe de l'algorithme FM-CM

. Nous avons montré (§ 3.3.2) que toute RF de la couverture minimale CM d'un ensemble E est directe et réciproquement si elle est directe, elle appartient à la couverture minimale.

$$\forall Ra \in CM \iff \nexists\, Rb \\ \nexists\, Rc \in FM \text{ telles que } Ra = Rb \mathbin{+\!\!>} Rc$$

Ra est dite *irredondante*.

. Une interprétation de ce théorème est de dire que Ra $\in$ FM n'appartient à la couverture minimale que si et seulement si il est impossible de trouver Rb, Rc de FM telles que Ra = Rb +> Rc.

. Conclusion algorithmique :
Quand un processus de fermeture construit Ra par

$$Ra = Rb \mathbin{+\!\!>} Rc,$$

Ra ne peut appartenir à la couverture minimale que si Ra absorbe Rc.

. Règles de construction d'un algorithme FM-CM :
Si le processus de construction de RF d'un algorithme de fermeture fait toutes les opérations possibles de pseudo-transitivité entre RF (cf. algorithme 2 du § A.2), alors pour construire un algorithme FM-CM

il suffit de considérer que :

- CM est initialisée par E, ensemble de départ
- chaque nouvelle RF R construite par R = Ra +> Rb doit être examinée par :

  . le processus de fermeture qui :

    1) l'élimine si elle peut être absorbée par une RF déjà connue
    2) l'élimine si elle est égale à une RF déjà connue
    3) la conserve sinon.

  . le processus de couverture minimale qui :

    4) dans le cas 2) élimine de CM la RF égalée par R
    5) la range dans CM si elle absorbe Rb
       et si elle absorbe une RF qui était considérée comme ϵ CM (qui peut être Rb elle-même).

Nous allons maintenant montrer comment l'algorithme FM de fermeture proposé au § A.2 peut avec des aménagements servir de base à la construction d'un algorithme FM-CM.

A.3.2. Construction de l'algorithme FM-CM à partir de l'algorithme FM

L'algorithme FM ne fait pas toutes les combinaisons de RF puisque justement il en élimine certaines pour des raisons d'efficacité, en s'appuyant :

- sur l'ordonnancement préalable des RF de E
- sur les propriétés de la pseudo-transitivité de l'opération +>.

Dans ces deux cas, nous allons vérifier s'il est possible de décider de l'appartenance d'une RF à CM, bien qu'elle ne soit pas construite de toutes les manières possibles ; si cette décision s'avère impossible, il faudra aménager l'algorithme FM.

A.3.2.1. Suppression de construction de RF à cause de l'ordonnancement des RF de E

Soient Ra et Rb $\in$ E,
et soit Rd qui a été construite par : Rd = Ra +> (Rb +> Rc)
mais qui aurait pu l'être également par :

$$\begin{cases} \text{Re} = \text{Rb} +> (\text{Ra} +> \text{Rc}) \\ \text{Re} = \text{Rd} \end{cases}$$

Soit Rf = Rb +> Rc
et Rh = Ra +> Rc.

Re n'est pas construite par l'algorithme FM, à cause de l'ordonnancement des RF, Rb étant supposée placée avant Rb dans E trié.

. Proposition :

La connaissance de Re est inutile pour savoir si oui ou non Rd peut appartenir à la couverture minimale.

. Preuve :

Pour que Rd $\in$ CM, il faut que Rd absorbe Rf.

$d(a) \wedge [g(b) \vee g(c)] \neq \emptyset$ car Rd existe

- si : $d(a) \wedge g(b) = \emptyset$
  $d(a) \wedge g(c) \neq \emptyset$
  $g(d) = g(a) \vee g(b) \vee [g(c) \wedge \overline{d(a)} \wedge \overline{d(b)}]$
  $g(f) = g(b) \vee [g(c) \wedge \overline{d(b)}]$

Il faut que $g(d) \subset g(f)$ pour que Rd $\in$ CM. Or, $d(a) \notin g(d)$ et $d(a) \in g(f)$; comme $d(a) \neq d(b)$ (car sinon Rd ne peut être construite), Rd $\notin$ CM dans cette hypothèse ;

- donc : $d(a) \wedge g(b) \neq \emptyset$
  alors $g(d) = g(a) \vee [g(b) \wedge \overline{d(a)}] \vee [g(c) \wedge \overline{d(b)} \wedge \overline{d(a)}]$
  et $g(e) = g(b) \vee [g(a) \wedge \overline{d(b)}] \vee [g(c) \wedge \overline{d(b)} \wedge \overline{d(a)}]$

Comme il n'y a pas de circuits, $g(a) \wedge \overline{d(b)} = g(a)$
et $d(a) \in g(e)$ car $d(a) \in g(b)$ sans que $d(a) \in g(d)$:
Rd absorbe Re.

- Ainsi si Rd absorbe Rf, Rd ne peut être formée d'une autre façon avec les mêmes relations fonctionnelles ; la connaissance de Re n'est donc pas indispensable pour déterminer si oui ou non Rd peut appartenir à CM.

A.3.2.2. Suppression de construction de RF à cause de la pseudo-associativité de l'opération +>

. Soient Ra,Rb ϵ E et Rc ϵ E $\vee$ G, telles qu'elles permettent de construire :

Rd = Ra +> (Rb +> Rc)
et Re = (Ra +> Rb) +> Rc avec Re = Rd

Face à une telle situation, l'algorithme FM ne génère que Rd pour des raisons d'efficacité et par application de la propriété de pseudo-associativité de l'opération +> ; ainsi Re est masquée.

Nous allons montrer sur un exemple pourquoi ce mécanisme est inacceptable dans le cadre d'un algorithme de couverture minimale.

Exemple A.10

Soit E :

1 Rz = (A,C,X → E)
2 Rc = (A,D → E)
3 Rb = (C,B → D)
4 Ra = (A,C → B)

Le processus de fermeture FM construit à partir de E :

5 Rf = (A,B,C → E) (3 +> 2)
6 Ri = (A,C → D) (4 +> 3)
7 Rd = (A,C → E) (4 +> 5)

Il construit ainsi Rd = (Ra +> Rb) +> Rc ;
si nous appliquons sans ménagement les règles du § A.3.1., nous allons décider que Rd appartient à la couverture minimale CM puisque :

- Rd absorbe Rf ;
- Rd absorbe Rz qui, comme RF de E, était considérée comme susceptible d'appartenir à CM : ainsi Rd devient susceptible d'appartenir à CM ;
- le processus s'arrêtant après la construction de Rd, Rd appartiendrait donc à la couverture minimale.
  Or, en fait, il n'en est rien : en effet, le processus de fermeture a masqué le fait que : Rd = Re = (Ra +> Rb) +> Rc et a interdit de constater que Rd n'absorbe pas Rc et ne peut donc pas appartenir à la couverture minimale.

. Cas général

Soient Ra, Rb, Rc telles que les opérations suivantes sont possibles :

Rh = Ra +> Rb
Rf = Rb +> Rc
Rd = Ra +> (Rb +> Rc) = Ra +> Rf
Re = (Ra +> Rb) +> Rc = Rh +> Rc.

Dans quelle mesure la connaissance de Re est-elle indispensable dans l'exécution du processus de couverture minimale ?

Re n'est intéressant que par rapport à Rd ; mais :

(1) si Rd n'absorbe pas Rf
(2) si Re est absorbée par Rd
(3) si Re absorbe Rc,

point n'est besoin de connaître Re pour décider si Rd est susceptible d'appartenir à CM ou non ;
en effet, pour (1), Rd ne peut appartenir à CM ;
pour (2), Re n'est même pas équivalent à Rd ;
pour (3), Re est susceptible d'appartenir à CM ;
la seule connaissance de Rd suffit alors pour décider de son appartenance possible à CM.

Ces conditions sont équivalentes à :

(1) $\Leftrightarrow$ $g(d) \neq g(b) \vee g(c) \Leftrightarrow g(a) \neq g(b) \vee g(c)$

(2) $\Leftrightarrow$ $d(a) \wedge g(c) \neq \emptyset$ d'après la pseudo-associativité de +> (§ A.1.2.)

(3) $\Leftrightarrow$ $g(e) \subset g(c) \Leftrightarrow g(a) \vee [g(b) \wedge \overline{d(a)}] \subset g(c)$.

A.3.2.3. Conséquences algorithmiques

Soient Ra, Rb, Rc, Rd, Re, Rf, Rn les relations définies précédemment.

. Création de l'ensemble de RF ENPLUS

ENPLUS est l'ensemble de RF Re construites telles que :

$Re = Rh +> Rc$ avec $Rh \in G$

et que les conditions précédentes sont vérifiées.

ENPLUS est mis à jour dès que Rh est construite.

. Synchronisation des mises à jour de ENPLUS et de G

Quand Rd est construite par l'algorithme, est-ce que Re est construite avant Rd ? S'il en était autrement, il serait impossible de décider de l'appartenance ou non de Rd à la CM.

$Rd = Ra +> (Rb +> Rc) = Ra +> Rf$
$Re = (Ra +> Rb) +> Rc = Rh +> Rc$

Comme toutes les opérations sont possibles, on peut en conclure que : rang (d(a)) > rang (d(b)) > rang (d(c)) ; les RF sont construites dans l'ordre suivant, compte tenu de l'ordonnancement des RF :

$$\begin{cases} Rf = Rb +> Rc \\ Rh = Ra +> Rb \\ Rd = Ra +> Rf \end{cases}$$

Ainsi Rh est bien connue avant Rd.

Reste la question : est-ce que Rc est connue avant Rh ?

Si $Rc \in E$: c'est évident
si $Rc \in G$: $Rc = Rx +> Ry$ avec $Rx \in E$;

(4) $d(h) \wedge g(x) \neq 0$ car s'il en était autrement, $d(h) \wedge g(y) \neq \emptyset$ et Re et Rd seraient absorbées par $Rh +> Ry$

Ainsi comme d(c) = d(h), rang (d(c)) > rang (d(x)) et donc rang (d(a)) > rang (d(x)).

Ainsi Rc = Rx +> Ry est construite avant Rh = Ra +> Rb.

A.3.3. Algorithme de fermeture et de couverture minimale

A.3.3.1. Algorithme FM-CM

L'algorithme FM-CM reprend l'algorithme FM décrit dans la Fig. A.7 (page A.22). Pour chaque RF Rh construite, il examine non seulement si elle peut être éliminée mais aussi il détermine

. si elle est susceptible d'appartenir à CM en vérifiant si
  Rh absorbe Rb
  Rh n'est égale à aucune RF de ENPLUS
  Rh absorbe une RF qui est susceptible d'appartenir à CM (et qui peut être Rb elle-même)

. si elle peut construire des RF de ENPLUS :
  c'est vrai s'il existe une RF Rc connue telle que :
  $d(h) \wedge g(c) \neq \emptyset$
  et (1) $g(a) \subset g(b) \vee g(c)$
  (2) $d(a) \wedge g(c) = \emptyset$
  (3) $g(h) \not\subset g(c)$,
  si ces conditions sont vérifiées, on range Rh +> Rc dans ENPLUS.

A.3.3.2. Exemple de fonctionnement

Exemple A.11

Ensemble de RF de départ :

1 Y,Z → W
2 X,Y → Z
3 X,V → Z
4 U,X → Y
5 X → U

Cet ehsemble correspond à l'ordonnancement des constituants suivants :

rang 1 : W
rang 2 : Z
rang 3 : Y, V
rang 4 : U
rang 5 : X

Le processus de fermeture crée dans l'ordre :

6 = (2 +> 1)	X,Y → W	
(3 +> 1)	X,V,Y → W	qui est éliminé par 6
7 = (4 +> 1)	U,X,Z → W	
8 = (4 +> 2)	U,X → Z	
9 = (4 +> 6)	U,X → W	qui élimine 7
10 = (5 +> 4)	X → Y	qui élimine 4
11 = (5 +> 8)	X → Z	qui élimine 2, 3 et 8
12 = (5 +> 9)	X → W	qui élimine 9, 11 et 6

Le processus de couverture considère a priori que toutes les RF de départ font partie de la couverture minimale, puis il raisonne étape par étape.

ENPLUS

6, 7 : RAS

8 : $\notin$ CM ; (8 +> 1) $\notin$ ENPLUS
(8 +> 7) $\notin$ ENPLUS

9 : RAS

10 : 10 = 5 +> 4 et 10 absorbe 4 ;
or, 4 $\in$ CM et 10 $\notin$ ENPLUS
10 $\in$ CM
(10 +> 1) $\in$ ENPLUS — (X,Y → W)
(10 +> 2) = (X → Y) +> (X,Y → Z)
= (X → Z)
n'appartient pas à ENPLUS car
(10 +> 2) absorbe 2 ;
donc si 5 +> (4 +> 2) absorbe
(4 +> 2), alors il est possible
qu'elle appartienne à la CM ; la
connaissance de (10 +> 2) ne sert
à rien
(10 +> 6) $\notin$ ENPLUS.

11 : 11 = 5 +> 8 : élimine non seulement 8, mais aussi 2 et 3 qui appartiennent à CM et ne se trouve pas dans ENPLUS $\Longrightarrow$ 11 $\in$ CM — 11 +> 1 = (X,Z → W)

12 : 12 = 5 +> 9 : élimine bien 9 mais n'élimine aucune RF de ENPLUS

Couverture minimale :

Y,Z → W
X → Z
X → Y
X → U

Exemple A.12

Ensemble de RF de départ et construites par le processus de fermeture.

			éliminées par :
1	A,C,X → E		11
2	F → E		
3	A,D → F		
4	B,C → D		
5	A,C → B		
6	A,D → E	6 = 3 +> 2	
7	A,B,C → F	7 = 4 +> 3	10
8	A,B,C → E	8 = 4 +> 6	11
9	A,C → D	9 = 5 +> 4	
10	A,C → F	10 = 5 +> 7	
11	A,C → E	11 = 5 +> 8	

La relation 11 (11 = 5 +> 8) absorbe non seulement 8 mais aussi 1 que le processus de couverture minimale suppose a priori appartenant à CM : pourtant 11 n'appartient pas à CM ;
en effet, à la relation 9, le processus de couverture minimale essaie de faire 9 +> 6, c'est-à-dire :

$$(A,C \rightarrow D) +> (A,D \rightarrow E) = (A,C \rightarrow E) \qquad (11)$$

La relation obtenue n'absorbe pas (A,D → E) ; elle ne peut donc pas appartenir à la CM ; elle est inscrite dans ENPLUS ; ainsi quand le processus de fermeture construira 11, elle se trouvera déjà dans ENPLUS ; le processus de couverture minimale rejettera alors 11 de CM.

A N N E X E B

PROPRIETES DES DECOMPOSITIONS ARBORESCENTES ET DE LEURS COMBINAISONS AVEC DES RELATIONS FONCTIONNELLES

SOMMAIRE

Cette annexe a pour but de démontrer les propositions de restructuration du § 4.3 ; dans ce but, elle contient l'étude des propriétés des décompositions arborescentes et notamment des interactions existant entre une décomposition arborescente et une relation fonctionnelle ou bien entre deux décompositions arborescentes.

Nous donnons dans ce tableau la correspondance entre le type de restructuration du § 4.3 et les numéros des propriétés des décompositions arborescentes dans cette annexe :

TYPE DE RESTRUCTURATION	PROPRIETES DES DECOMPOSITIONS ARBORESCENTES
regroupement	P-4, P-20
alourdissement	P-7, P-19
projection	P-8, P-9, P-23, P-27
dédoublement	P-9, P-28
inversion	P-16, P-17, P-24, P-26
élimination de constituants	P-5, P-21
rajout de constituants	P-15, P-25

Tableau B.1

Remarque : Dans cette annexe, nous allons simplifier nos notations afin de rendre moins lourde l'écriture des démonstrations :

- les constituants A, B, C, ..., X, Y, Z appartiennent tous à une même relation S ;

- [A,B] est la notation simplifiée de [A,B]S ;
- a $\in$ [A] désigne une valeur de A : <A:a>.

B.1. DECOMPOSITIONS ARBORESCENTES SIMPLES

B.1.1. Conditions de décomposition d'une relation

Le problème de la décomposition d'une relation se pose ainsi : étant donné une relation S d'espace de constituants $\mathcal{A}$ = {A,B,C}. Est-il possible de trouver deux relations P et Q telles que

$$T = P * Q$$
$$\text{avec} \quad P = [A,B]T$$
$$\text{et} \quad Q = [A,C]T$$

Si ce problème admet une solution, on peut remplacer la relation T par ces deux projections et on a la garantie de pouvoir retrouver la relation T à partir d'elles.

(P-1) : La condition nécessaire et suffisante pour que T admette une décomposition est que pour :

$\forall a \in [A]T$ on ait $[a,B,C]T = [a,B]T * [a,C]T$.

(P-2) : Une condition nécessaire et suffisante pour que la collection T soit décomposable est que pour :

$\forall a \in [A]T$ et $\forall b \in [a,B]T$ on ait $[a,b,C]T = [a,C]T$.

(P-3) : Si une relation T possède comme invariant la relation fonctionnelle :

$$A \overset{T}{\rightarrow} B$$

alors T est décomposable en :

$T = [A,B]T * [A,C]T$.

B.1.2. Propriétés élémentaires d'une décomposition

B.1.2.1. Groupement d'éléments

(P-4) : Si X : Y|Z|U est une décomposition arborescente et Y et Z deux parties choisies parmi Y, Z et U, alors X : Y,Z|U est une décomposition arborescente.

Cette propriété peut être écrite également sous la forme de :
$[X,Y,Z,U] = [X,Y] * [X,Z] * [X,U] \Rightarrow [X,Y,Z,U] = [X,Y,Z] * [X,U]$;
elle découle directement d'une propriété fondamentale de la décomposition normale qui est que :

$$[X,Y] * [X,Z] \geq [X,Y,Z] ;$$

ce qui implique que :

$$[X,Y] * [X,Z] * [X,U] \geq [X,Y,Z] * [X,U]$$

et également :

$$[X,Y,Z] * [X,U] \geq [X,Y,Z,U]$$

or, comme

$$[X,Y,Z,U] = [X,Y] * [X,Z] * [X,U]$$
$$[X,Y,Z,U] = [X,Y,Z] * [X,U].$$

B.1.2.2. Suppression d'éléments

(P-5) : Si X : Y|Z|U est une décomposition arborescente et Y et Z deux parties choisies, alors X : Y|Z est une décomposition arborescente.

En effet, d'après (P-1), X : Y|Z|U signifie que

$$\forall x \in X \quad [x,Y,Z,U] = [x,Y] * [x,Z] * [x,U]$$

donc en particulier

$$\forall x \in X \quad [x,Y,Z] = [x,Y] * [x,Z]$$
$$\Leftrightarrow \quad X : Y|Z$$

(P-6) : Si X : Y|Z est une décomposition arborescente et T une partie telle que T ⊂ Y alors X : T|Z est aussi une décomposition arborescente.

$$X : Y|Z \iff [X,Y] * [X,Z] = [X,Y,Z]$$

mais d'après la (P-2), on peut écrire :

$$[x,y,Z] = [x,Z]$$

mais puisque T ⊂ Y, $[x,y,Z] = [x,t,Z]$ et donc :

$$[x,Z] = [x,t,Z]$$

qui est équivalent à X : T|Z.

B.1.2.3. Alourdissement de la racine

(P-7) : Si X : Y,Z|U,V est une décomposition arborescente et si Y et U sont deux parties choisies parmi Y,Z,U,V alors :

$$X,Y,U : Z|V$$

est une décomposition arborescente.

Comme le constituant composé [X,Y,U] est contenu dans [X,Y,Z,U,V], on peut écrire :

$$[X,Y,U] * [X,Y,Z,U,V] = [X,Y,Z,U,V] \quad ;$$

en utilisant la décomposition de (X,Y,Z,U,V) on en déduit :

$$[X,Y,Z,U,V] = [X,Y,U] * [X,Y,Z] * [X,U,V]$$

ou bien encore

$$[X,Y,Z,U,V] = [X,Y,U] * [X,Y,Z] * [X,Y,U] * [X,U,V]$$

et par application de P-4 :

$$[X,Y,Z,U,V] = [X,Y,Z,U] * [X,Y,U,V].$$

B.1.2.4. Projection de la racine

(P-8) : Si X,Y,Z : U|V est une décomposition arborescente et si de plus X,Y : Z|U,V est aussi une décomposition arborescente, alors :

X,Y : Z|U|V

est une décomposition arborescente.

En effet, il suffit de traduire les décompositions arborescentes en terme de composition et on obtient :

$$[X,Y,Z,U,V] = [X,Y,Z,U] * [X,Y,Z,V] \qquad (1)$$

D'autre part, par application de P-6 à la décomposition X,Y : Z|U,V, on peut générer les deux décompositions :

$$X,Y : Z|U \iff [X,Y,Z,U] = [X,Y,Z] * [X,Y,U] \qquad (2)$$

$$X,Y : Z|V \iff [X,Y,Z,V] = [X,Y,Z] * [X,Y,V] \qquad (3)$$

En substituant (2) et (3) dans (1), il vient :

$$[X,Y,Z,U,V] = [X,Y,Z] * [X,Y,U] * [X,Y,Z] * [Z,Y,V] \qquad (4)$$

ce qui peut s'écrire d'après la loi d'absorption :

$$[X,Y,Z,U,V] = [X,Y,Z] * [X,Y,U] * [X,Y,V] \qquad (5)$$

L'équation (5) est bien la décomposition arborescente :

X,Y : Z|U|V.

B.1.2.5. Décomposition d'une branche

(P-9) : Etant donné deux décompositions arborescentes

X : Y,Z|V (1)

X : Y|Z (2)

alors

X : Y|Z|V (3)

est une décomposition arborescente.

D'après (P-1), on peut écrire pour (1)

$\forall x \in [X], \forall v \in [x,V], [x,v,Y,Z] = [x,Y,Z]$

pour (2)

$\forall x \in [X], \forall y \in [x,Y], [x,y,Z] = [x,Z]$

$\Rightarrow \forall x \in [X], \forall v \in [x,V], [x,v,Z] = [x,Z]$; ainsi : $X : V|Y$ et de même $X : V|Z \Rightarrow X : V|Y|Z$.

B.1.2.6. Autres interactions entre décompositions

(P-10) :

Si $A : C|F$ (1)

et $A,C : D|F$ (2)

sont deux décompositions arborescentes,

alors $A : C,D|F$ (3)

en est une également.

Démonstration :

(1) $\Leftrightarrow$ $\forall$ <A:a> $\in$ A, $\forall$ <C:c> $\in$ [<A:a>,C],
[<A:a>, <C:c>, F] = [<A:a>,F]

(2) $\Leftrightarrow$ $\forall$ [<A:a>, <C:c>] $\in$ [A,C], $\forall$ <D:d> $\in$ [<A:a>, <C:c>, D],
[<A:a>, <C:c>, <D:d>, F] = [<A:a>, <C:c>, F]

En rapprochant les deux résultats et en constatant que l'expression
$\forall$ [<A:a>, <C:c>] $\in$ [A,C] est équivalente à
($\forall$ <A:a> $\in$ A, $\forall$ <C:c> $\in$ [<A:a>,C])
on obtient
$\forall$ <A:a> $\in$ A, $\forall$ (<C:c>, <D:d>) $\in$ [<A:a>,C,D]

[<A:a>, <C:c>, <D:d>, F] = [<A:a>, F]

$\Leftrightarrow$ (3).

(P-11) :

(D1) $X_1 : Y_1|Z_1|T_1|U_1$

(D2) $X_2 : Y_2|Z_2|T_2|U_2$

vérifiant soit la condition (a) ou la condition (b) :

(a) si $X_1 \wedge X_2 = \emptyset$ $\quad Y_1,Z_1,T_1,U_1 \supset X_2$

$\qquad\qquad\qquad\qquad\qquad\quad Y_2,Z_2,T_2,U_2 \supset X_1$

(b) si $X_1 \wedge X_2 \neq \emptyset$ $\quad Y_1,Z_1,T_1,U_1 \supset X_2 \wedge X_1'$

$\qquad\qquad\qquad\qquad\qquad\quad Y_2,Z_2,T_2,U_2 \supset X_1 \wedge X_2'$

où X_1' et X_2' désignent le complément de X_1 et X_2 par rapport à l'ensemble $\mathcal{A}$ de tous les constituants.

Alors il est démontré dans [14] que Y_1 et Z_1 étant deux feuilles choisies parmi toutes les feuilles de la décomposition :

$$X_1, (Y_1,Z_1) \wedge X_2 : Y_1 \wedge Y_2 \mid Y_1 \wedge Z_2 \mid Y_1 \wedge T_2 \mid Y_1 \wedge U_2 \mid$$
$$Z_1 \wedge Y_2 \mid Z_1 \wedge Z_2 \mid Z_1 \wedge T_2 \mid Z_1 \wedge U_2 \mid T_1 \mid U_1$$

est également une décomposition arborescente (D3).

B.1.3. Interactions entre relations fonctionnelles et décompositions arborescentes

Désignons symboliquement par r l'ensemble des constituants qui constituent la racine de la décomposition arborescente, par f l'ensemble des constituants relatif aux feuilles et par e le complément de F et de r par rapport à $\mathcal{A}$. Nous dirons qu'une relation fonctionnelle est de la classe :

$$e,f,r \rightarrow r$$

si dans sa partie gauche il y a des constituants qui appartiennent à e, f et r, tandis que la partie droite (qui contient un constituant unique) appartient à r. Il y a au total 21 classes possibles, correspondant aux différentes combinaisons. Le problème que l'on se pose est de savoir si, étant donné une décomposition particulière et une relation fonctionnelle, cette conjonction permet de générer de nouvelles relations fonctionnelles et/ou de nouvelles décompositions arborescentes. L'étude a montré que la classe de la relation fonctionnelle par rapport à la décomposition arborescente était un facteur primordial. Il n'est pas possible de reproduire ici la totalité des résultats obtenus ; nous illustrerons ce problème à partir des points essentiels.

B.1.3.1. Génération de nouvelles relations fonctionnelles

Elle résulte de l'interaction d'une décomposition arborescente et d'une relation fonctionnelle de type r,f → f ou f → f.

(P-12) : Si X : Y|Z est une décomposition arborescente
et Y → Z est une relation fonctionnelle,
alors X → Z est une relation fonctionnelle.

D'après (P-2), X : Y|Z est équivalente à :

$\forall x \in [X], \forall y \in [x,Y], [x,y,Z] = [x,Z]$.

A cause de la relation fonctionnelle Y → Z, le cardinal de l'ensemble [x,y,Z] est égal à 1 et par conséquent celui de [x,Z], ce qui signifie qu'à un x ne correspond qu'un z, quel que soit x. Ainsi il existe la relation fonctionnelle X → Z.

(P-13) : Si $A_1,A_2 : A_3,A_4|A_5,A_6|A_7$ est une décomposition arborescente et si $A_2,A_3,A_5 \rightarrow A_6$ est une relation fonctionnelle, alors $A_1,A_2,A_5 \rightarrow A_6$ est une relation fonctionnelle.

Par alourdissement de la racine (P-5), on peut obtenir :

$$A_1,A_2,A_5 : A_3,A_4|A_6|A_7$$

et par allègement des feuilles (6)

$$A_1,A_2,A_5 : A_3|A_6$$

D'après la (P-2), on en déduit :

$$[a_1,a_2,a_5,a_3,A_6] = [a_1,a_2,a_5,A_6]$$

pour a_1,a_2,a_5,a_3 donnés ; mais puisque $A_2,A_3,A_5 \rightarrow A_6$

$$\text{card } [a_1,a_2,a_5,a_3,A_6] = \text{card } [a_1,a_2,a_5,A_6] = 1.$$

Ce qui fait que $A_1,A_2,A_5 \rightarrow A_6$ est une relation fonctionnelle.

(P-13 bis) : De même si $A_1,A_2 : A_3,A_4|A_5,A_6|A_7$ est une décomposition arborescente et si $A_2,A_3 \rightarrow A_6$ est une relation fonctionnelle, alors $A_1,A_2 \rightarrow A_6$ est une relation fonctionnelle.

(P-14) : Si X : Y|Z est une décomposition arborescente et si X,Y → U et X,Z → U sont deux relations fonctionnelles, alors X → U est une relation fonctionnelle.

Démonstration :

Par application de (P-3) on peut écrire puisque X,Y → U est une relation fonctionnelle :

$$[X,Y,Z,U] = [X,Y,U] * [X,Y,Z]$$
$$= [X,Y,U] * [X,Y] * [X,Z] \quad \text{(par application de (P-9))}$$
$$= [X,Y,U] * [X,Z] \quad \text{(par application de (P-4))}$$

Comme X,Z → U est également une relation fonctionnelle, on déduit par application de (P-12) que X → U est une relation fonctionnelle.

B.1.3.2. Introduction de nouveaux constituants dans une décomposition

Elle résulte de la combinaison d'une décomposition et d'une relation fonctionnelle de la classe f,r → e.

(P.15) : Si X : Y|Z est une décomposition arborescente et X,Y → A une relation fonctionnelle, alors X : Y,A|Z est une décomposition arborescente. Par application de (P-2), on peut écrire :

$$[x,y,Z] = [x,Z]$$

mais puisque à un x et y ne correspond qu'une seule valeur a, on aura

$$[x,y,Z] = [x,y,a,Z] = [x,Z]$$

qui représente la décomposition arborescente :

$$X : Y,A|Z$$

Une fois de plus, suivant la classe de la relation fonctionnelle, on obtiendra des résultats différents que ceux présentés précédemment.

Autre cas : Etant donné la décomposition :

$$A_1,A_2 : A_3,A_4|A_5,A_6|A_7 \qquad (1)$$

et une relation fonctionnelle $A_5 \rightarrow A_8$.

Alors $A_1,A_2,A_5 \rightarrow A_8$ est également une relation fonctionnelle et d'après le résultat précédent $A_1,A_2 : A_3,A_4|A_5,A_6,A_8|A_7$ représente une décomposition arborescente simple.

B.1.3.3. Modification de l'arborescence de la décomposition

Elle résulte de l'interaction d'une décomposition arborescente et d'une relation fonctionnelle de la classe $f,r \rightarrow r$.

(P-16) : Si $X,Y,Z : U|V$ est une décomposition arborescente et si $X,Y \rightarrow Z$ une relation fonctionnelle, alors $X,Y : Z|U|V$ est une décomposition arborescente.

Démonstration :

Comme $X,Y \rightarrow Z$, on peut former $X,Y : Z|U,V$ par application de (P-3) ; cette nouvelle décomposition combinée avec la première permet de créer $X,Y : Z|U|V$ par application de (P-8).

(P-17) : Si $X,Y,Z : U|V|W$ est une décomposition arborescente, et si $X,U \rightarrow Y$ est une relation fonctionnelle, alors $X,U,Z : Y|V|W$ est une décomposition arborescente.

La démonstration de (P-17) se fait par alourdissement de la racine (P-7) et ensuite par application de (P-16).

B.1.4. Tableau récapitulatif des interactions entre une relation fonctionnelle et une décomposition arborescente

f,r,e désignent les types de constituants d'une relation fonctionnelle par rapport à une décomposition.

TYPE		Exemples		ACTIONS A ENTREPRENDRE
f,r,e → f		A,B : C,D/F,G/H	A,C,L → D	rien à faire
f,r,e → r		A,B : C,D/F,G/H	A,C,L → B	rien à faire
f,r,e → e		A,B : C,D/F,G/H	A,C,L → M	rien à faire
f,r → f	12	A,B : C,D/F,G,H/I	B,C → F	A,B → F A,B : C,D/F/G,H/I
			B,C,G → F	A,B,G → F A,B,G : C,D/F/H/I
f,r → r	17	A,B : C,D/F,G/I	B,C → A	B,C : D/F,G/I/A
f,r → e	14	A,B : C,D/F,G/I	B,C → L	A,B : C,D,L/F,G/I ou A,B,C : D/L/F,G/I
			B,C,F → L	A,B,C,F : D/L/G/I ou A,B,C : D/F,G,L/I ou A,B,F : C,D,L/G/I ou A,B : C,D,F,G,L/I
f,e →		A,B : C,D/F,G/I	C,L →	rien à faire
r,e →		A,B : C,D/F,G/I	A,L →	rien à faire
f → f	12	A,B : C,D/F,G,L/I	C → F	A,B → F A,B : C,D/F/G,L/I
			C,G → F	A,B,G → F A,B,G : C,D/F/L/I
f → r	17	A,B : C,D/F,G/I	F,C → A	F,C,B : D/G/I/A
f → e	15	A,B : C,D/F,G/I	C → L	A,B,C : L/D/F,G/I A,B : C,L,D/F,G/I
			C,G → L	A,B,C,G : D/F/L/I ou A,B,C : D/G,F,L/I ou A,B,G : C,D,L/F/I ou A,B,G,C : D/G/L/I

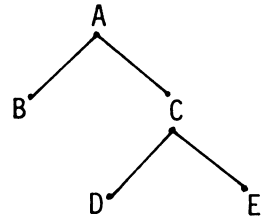
r → f	9	A,B : C,D/F,G	A → C	A,B : C/D/F,G
r → r	16	A,B : C,D/F,G	A → B	A : C,D/F,G/B
r → e	15	A,B : C,D/F,G	A → L	A,B : C,D/F,G/L
e →		A,B : C,D/F,G	L →	rien à faire

Tableau B.2

B.2. DECOMPOSITIONS ARBORESCENTES GENERALISEES

Nous avons donné la définition d'une décomposition arborescente généralisée (DAG) dans le chapitre 1.

Soit [A,B,C,D,E] = [A,B] * [A,C,D] * [A,C,E], une DAG ;
elle est représentée graphiquement ainsi :



Les noeuds non terminaux sont appelés racines secondaires, excepté pour la racine de l'arborescence appelée racine principale.

A chaque racine d'une DAG, on peut associer une décomposition arborescente simple ;
dans l'exemple pour la racine A : [A,B,C,D,E] = [A,B] * [A,C,D,E]
pour la racine C : [A,C,D,E] = [A,C,D] * [A,C,E]
rappelons que toute DAG est équivalente à un ensemble de décompositions arborescentes simples, justement celles obtenues aux racines de la DAG.

Par suite de cette équivalence, il existe une très grande corrélation entre les propriétés d'une DAG et d'une décomposition arborescente simple.

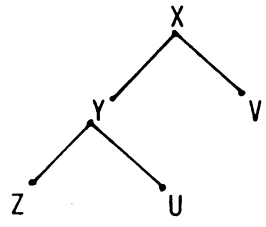
Dans les démonstrations suivantes, nous considérerons toujours une relation R et ses décompositions possibles ; quand nous utiliserons l'expression "D_1 est une décomposition", il sera sous-entendu :
"il existe une relation R_1 projection de R, dont D_1 est une décomposition" ;
nous utiliserons le mot *schéma* pour désigner un ensemble de constituants, ordonné suivant une arborescence, dont on ne sait si elle correspond à une décomposition d'une relation.

Si N est un noeud d'un schéma, N est une racine secondaire d'une DAG : il désignera également l'ensemble des constituants associés à ce noeud ;
S(N) désignera l'ensemble des racines secondaires qui séparent N de la racine principale, celle-ci y appartenant également ;
P(N) désignera l'ensemble des constituants séparés de la racine principale par N ;
P(C) où C désigne un ensemble de constituants : $C = \{C_1, C_2 \ldots C_n\}$,

$$P(C) = \bigcup_{i=1}^{n} P(C_i) - C \text{ par définition ;}$$

si C désigne un ensemble de constituants tel que $P(C) = \emptyset$, alors C sera appelé un *sous-arbre*.

Exemple :

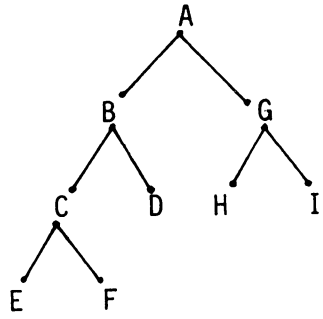


C = {Y,Z,U} est un sous-arbre
car $P(C) = \emptyset$.

B.2.1. Propriétés élémentaires d'une DAG

B.2.1.1. Lemme (P-18)

Soit la DAG suivante :



alors A,B,C : E|F|G,H,I est une décomposition arborescente.

Cas général : Soit D une DAG dont N est une de ses racines secondaires et C un ensemble de constituants tels que :

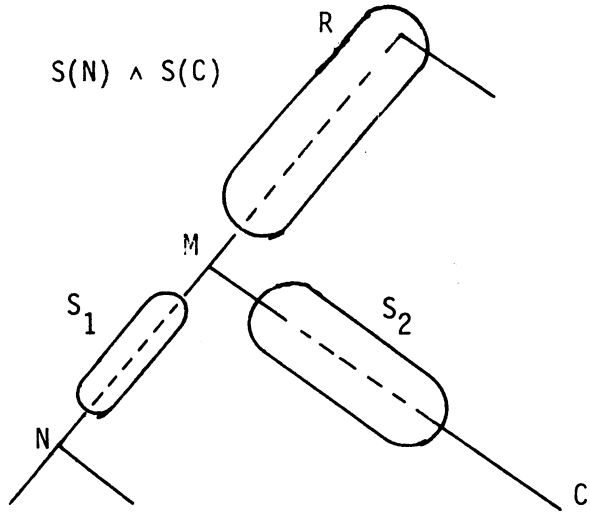
$$C \wedge (S(N) \vee P(N) \vee N) = \emptyset$$
$$P(C) = \emptyset$$

D est équivalent à un ensemble de décompositions arborescentes simples relatives à chaque racine de D ; soit D_n celle de N : elle est de la forme :

$$S(N),\ N \quad F_1|F_2 \ldots|F_n \text{ avec } P(N) = \bigcup_{k=1}^{n} F_k \ ;$$

nous voulons montrer que (D'_n) : $S(N),N : F_1|F_2\ldots|F_n|C$ est également une décomposition arborescente.

Dans ce but, nous considérons la racine M qui est la racine la plus proche de N de l'ensemble $(S(N) \wedge S(C))$; une telle racine existe toujours car au pire c'est la racine principale qui appartient à $(S(N) \wedge S(C))$.



Décomposition D

A M correspond la décomposition D_m :

D_m S(M), M : S_1, N, P(N) | S_2, C, X|Y ...

avec S_1 = P(N)-P(M)-M et S_2 = P(C)-P(M)-M ;

D_m devient après allègement des feuilles (P-6) :

D'_m S(M), M : S_1, N, P(N) | S_2, C

Comme S(N) = S_1 ∨ M ∨ S(M), D_n peut s'écrire également :

D'_n S_1 ∨ M ∨ S(M) ∨ N : $F_1|...|F_n$

D'_m et D'_n vérifient les conditions de (P-11) :

la racine de D'_m est bien incluse dans celle de D'_n et N,S_1 appartient bien à l'ensemble des feuilles de D'_m.

On peut former D''_n à partir de D'_m et D'_n :

D''_n S_1, M, S(M), N : $F_1|...|F_n|S_2$,C

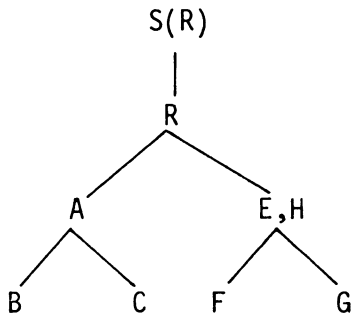
qui, par allègement d'une feuille (P-6), devient finalement la décomposition cherchée D'_n :

S(N), N : $F_1|...|F_n|$C.

B.2.1.2. Alourdissement d'une racine par un ensemble de constituants prédécesseurs de cette racine (P-19)

- Alourdissement d'une racine par un constituant d'une racine juste prédécesseur de celle-ci.

Soit D une DAG :



$\Longleftrightarrow$ $\begin{cases} D(R) & S(R),R : A,B,C|E,F,G,H \\ D(A) & S(R),A,R : B|C \\ D(E,H) & S(R),R,E,H : F|G \end{cases}$

On peut transformer D(R) en D'(R) par alourdissement de la racine (P-7)

D'(R) S(R),R,E : A,B,C|F,G,H

d'après le lemme précédent, on peut transformer D(A) en une autre décomposition :

D_1(A) S(R),A,R : B|C|E,F,G,H

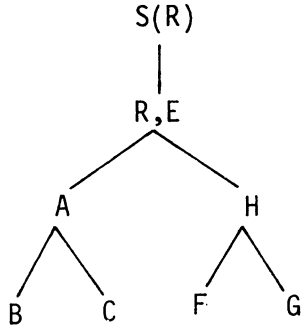
qui, par suppression d'éléments (P-6), devient :

D_2(A) S(R),A,R : B|C|E

et enfin par alourdissement de la racine

D'(A) S(R),A,R,E : B|C

L'ensemble des trois décompositions D'(R), D'(A) et D(E,H) satisfont les conditions pour former une DAG puisqu'elles sont les décompositions relatives aux racines de ce schéma :

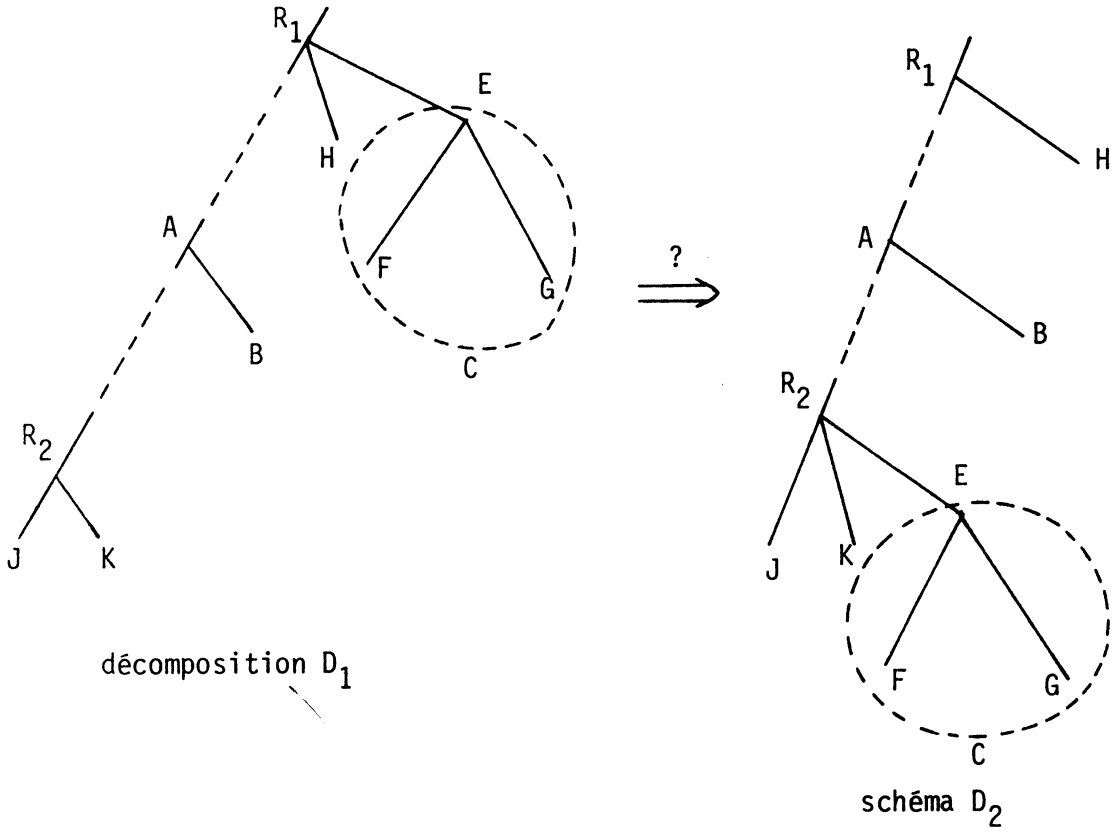


$\Longleftrightarrow$ D'(R) S(R),R,E : A,B,C|F,G,H
D'(A) S(R),R,E,A : B|C
D'(H) S(R),R,E,H : F|G.

- Ainsi l'alourdissement d'une racine par un constituant d'une racine juste prédécesseur de celle-ci transforme donc une DAG dans une autre DAG ;
la généralisation de cette propriété à plusieurs constituants appartenant aux prédécesseurs de la racine considérée est immédiate ; il suffit de les amener un par un, en les rapprochant à chaque fois d'un cran.

B.2.1.3. Sous-arbre d'une racine secondaire R_1 raccrochée à une racine secondaire R_2 appartenant aux prédécesseurs de R_1 (P-20)

Nous illustrons cette transformation de décomposition arborescente par :



décomposition D_1

schéma D_2

Correspond-il au schéma de droite une décomposition arborescente généralisée ?

Pour le démontrer, il suffit de vérifier qu'à chaque noeud de ce schéma correspond une décomposition arborescente simple cohérente.

- Les décompositions relatives aux racines successeurs de R_1 sont les mêmes dans D_1 que dans D_2.

- La décomposition de D_1 relative à la racine R_1 est formée par :

$$S(R_1),R_1 : H|C|X,R_2,P(R_2) \qquad (1)$$

où $X = P(R_1)-C-H-R_2-P(R_2)$

Elle se transforme par groupement d'éléments (P-4) en :

$$S(R_1),R_1 : H|X,R_2,P(R_2),C \qquad (2)$$

qui est celle qui correspondrait au noeud R_1 dans le schéma D_2.

- Celle de D_1 relative à une racine comprise entre R_1 et R_2, A, par exemple, s'écrit :

$$S(A),A : B|Y,R_2,P(R_2) \qquad (3)$$

avec $Y = P(A)-R_2-P(R_2)$;

d'après le lemme précédent, elle peut se transformer en

$$S(A),A : B|Y,R_2,P(R_2)|C \qquad (4)$$

qui, par regroupement d'éléments, donne :

$$S(A),A : B|Y,R_2,P(R_2),C \qquad (5)$$

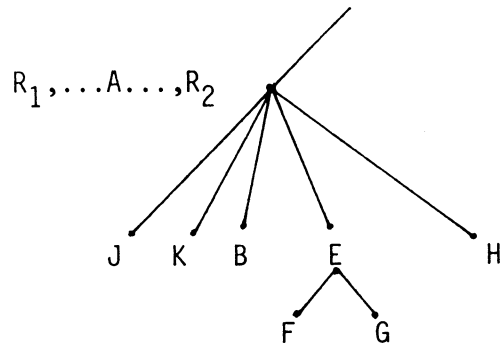
c'est-à-dire la décomposition qui correspondrait au noeud A dans le schéma D_2.

- De même celle de R_2 peut être transformée en :

$$S(R_2),R_2 : J|K|C \qquad (6)$$

qui correspond bien à celle relative à R_2 du schéma D_2.

- Restent à examiner les racines secondaires de C :
Nous savons, d'après le résultat précédent, qu'on peut alourdir R_1 de toutes les racines secondaires comprises entre R_1 et R_2, R_2 compris et obtenir ainsi une nouvelle décomposition arborescente D_3 ;



Décomposition D_3

La décomposition associée au noeud E s'écrit :

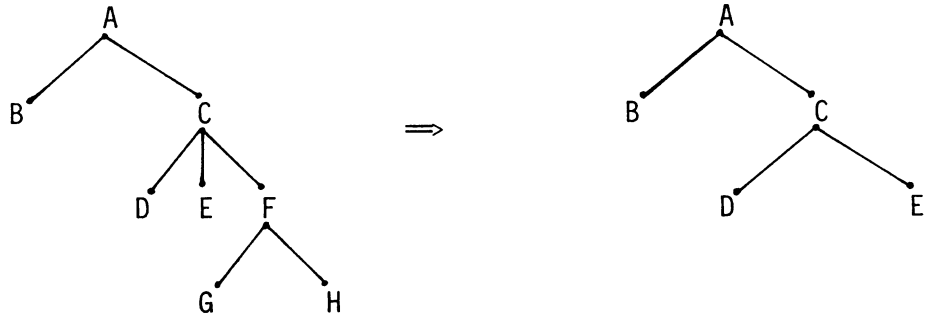
$$S(R_2),R_2,E : F|G \qquad (7)$$

Cette décomposition est exactement celle qui correspondrait au noeud E du schéma D_2 si celui-ci représentait une DAG.

- Comme à tous les noeuds du schéma D_2 correspondent les décompositions arborescentes simples compatibles à la formation d'une DAG, le schéma D_2 représente une DAG.

B.2.1.4. Suppression d'éléments X tels que P(X) = Ø (P-21)

Exemple :



Décomposition D_1 Schéma D_2

Pour démontrer que la suppression d'éléments X tels que P(X) = Ø transforme une DAG D_1 dans une autre DAG D_2, il suffit de considérer l'ensemble équivalent de décompositions arborescentes simples de D_1 et leur supprimer des éléments (P-5) ; on obtient ainsi l'ensemble des décompositions qui forment D_2.

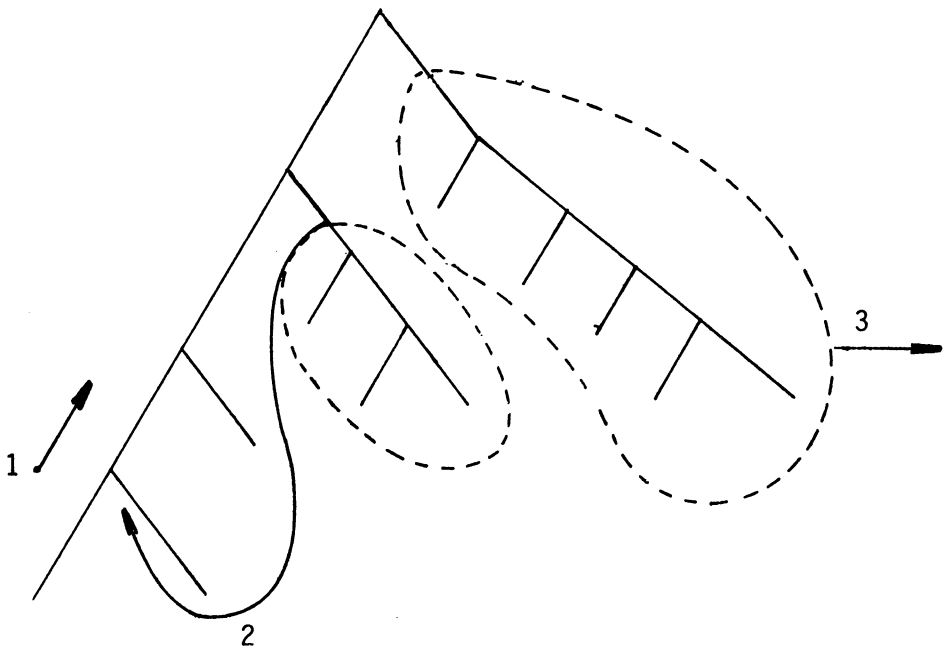
Dans l'exemple :

$$D_1 \iff \left\{ \begin{array}{l} A : B|C,D,E|F,G,H \\ A,C : D|E|F,G,H \\ A,C,F : G|H \end{array} \right. \implies \begin{array}{l} A : B|C,D,E \\ A,C : D|E \end{array} \iff D_2$$

B.2.1.5. Visualisation des propriétés élémentaires des DAG

Les propriétés élémentaires des DAG permettent donc de :

1. Alourdir une racine par un ensemble de constituants prédécesseurs de celle-ci.
2. Raccrocher un sous-arbre d'une racine secondaire R_1 à une racine secondaire R_2 prédécesseur de R_1.
3. Supprimer des éléments C tels que P(C) = Ø.



B.2.2. Interactions entre une relation fonctionnelle et une DAG

B.2.2.1. Classification des relations fonctionnelles par rapport à une DAG

- Si une relation fonctionnelle a sa partie droite n'appartenant à l'espace de définition $\mathcal{A}$ de la DAG, elle est dite de type [→ e].

- Sinon, toute relation fonctionnelle est de la forme :

$$S_1(M),\ P_1(M),\ F_1(M),\ C_1(M) \rightarrow M$$

où $S_1(M)$ désigne un sous-ensemble des successeurs de M

$P_1(M)$ désigne un sous-ensemble des prédécesseurs de M

$F_1(M)$ désigne un sous-ensemble des constituants appartenant à la même racine que M

$C_1(M)$ désigne un sous-ensemble de $\mathcal{A}$-(S(M), P(M), F(M)).

Suivant que la partie gauche d'une relation fonctionnelle admet (= 1) ou n'admet pas (= 0) de constituants d'un de ses sous-ensembles, elle est d'un type ou d'un autre :

$S_1(M)$	$P_1(M)$	$C_1(M)$	$F_1(M)$	Type
0	0	0	1	r → r
0	0	1	-	f → f
0	1	0	-	f → r
0	1	1	-	f → f
1	0	0	-	r → r
1	0	1	-	f → f
1	1	0	-	r,f→r
1	1	1	-	f → f

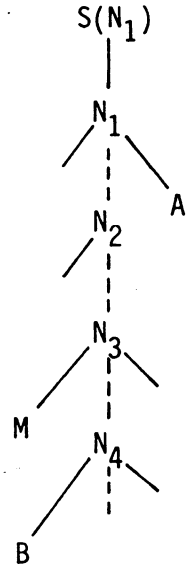
B.2.2.2. Génération de nouvelles relations fonctionnelles (P-22)

La relation fonctionnelle considérée est de type $\boxed{f \rightarrow f}$, c'est-à-dire $C_1(M) \neq \emptyset$.

- Dans un premier temps nous supposons également que :

$$S_1(M) = P_1(M) = F_1(M) = \emptyset.$$

Soit une DAG représentée par l'arborescente suivante :



et la relation fonctionnelle A,B → M

De cette DAG, on peut extraire les décompositions arborescentes simples suivantes à partir de l'ensemble équivalent de la DAG :

$D(N_1)$: $S(N_1)$, N_1 : $A|N_2$, N_3, N_4, M, B

$D(N_3)$: $S(N_3)$, N_3 : $M|N_4$, B

En appliquant (P-12) du § 4.2, on peut générer à partir de $D(N_1)$, $D(N_3)$ et de A,B → M de nouvelles relations fonctionnelles :

$$\begin{cases} S(N_1),\ N_1,\ B \rightarrow M \\ S(N_3),\ N_3 \rightarrow M \end{cases}$$

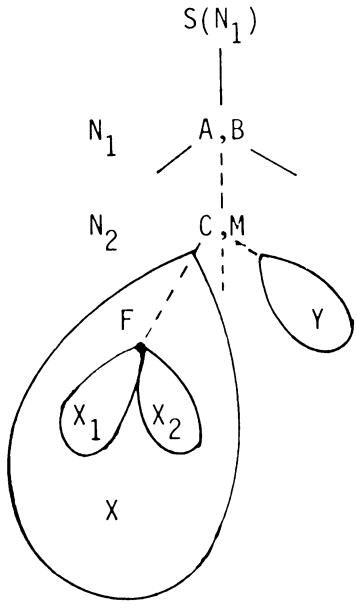
- Ce résultat est facilement généralisable dans le cas où $S_1(M)$ ou $P_1(M)$ ou $F_1(M)$ sont différents de l'ensemble vide ;
toutes les nouvelles relations fonctionnelles générées contiennent ces trois ensembles dans leur partie gauche ; certaines d'entre elles sont alors de type $\boxed{r \to f}$ ou $\boxed{f \to r}$ ou $\boxed{r,f \to r}$ et peuvent provoquer ensuite des modifications dans l'arborescence de la DAG.

Pour trouver toutes les nouvelles relations fonctionnelles, il suffit de combiner la relation fonctionnelle considérée avec toutes les décompositions arborescentes simples relatives aux racines N_i qui ont les propriétés suivantes :

- $N_i \in S(M)$
- il existe un constituant A_j de la partie gauche de la relation fonctionnelle, telle que $A_j \in C(M)$ et $N_i \in S(A_j)$
- N_i est la racine la plus éloignée de la racine principale de $(S(M) \wedge S(A_j))$.

B.2.2.3. Projection d'une racine (P-23)

La relation fonctionnelle associée est de type $\boxed{r \to r}$ et est de la forme $S_1(M), F_1(M) \to M$.



Soit D_1 la DAG dont l'arborescence est représentée ci-contre et soit la relation fonctionnelle A,C → M qui est bien du type $\boxed{r \to r}$ car $A \in S(M)$ et $C \in F(M)$.

Soit E_1 l'ensemble des décompositions arborescentes simples équivalentes à D_1 ;

celle relative au noeud N_2 s'écrit :

$D(N_2)$ $S(N_2)$, C, M : X|Y ;

celle du noeud F :

D(F) S(F), F : X|Y.

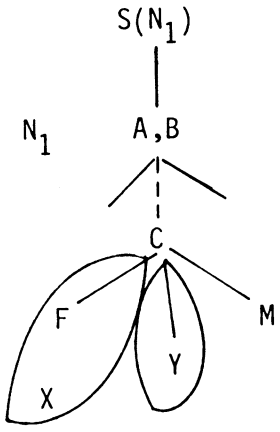
Les combinaisons entre A,C → M et $D(N_2)$ d'une part et D(F) d'autre part fournissent, d'après (P-18) :

$D'(N_2)$ $S(N_2)$, C : M|X|Y puisque A ∈ $S(N_2)$

D'(F) (S(F)-M), F : M|X|Y puisque C et M ∈ S(F).

Par allègement, D'(F) devient (S(F)-M), F : X|Y.

En remplaçant dans E_1 $D(N_2)$ par $D'(N_2)$ et D(F) par D'(F), on obtient E_2' qui correspond à l'arborescence suivante :



et donc à la DAG : D_2'

La combinaison d'une relation fonctionnelle de type $\boxed{r \to r}$ et d'une DAG conduit donc à une autre DAG par la création d'une nouvelle feuille.

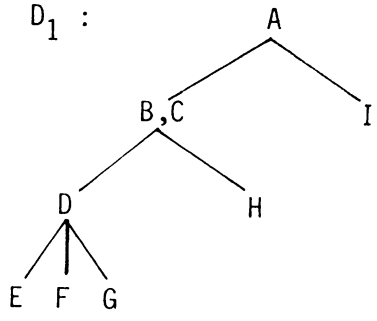
B.2.2.4. Inversion dans l'arborescence (P-24)

On cherche à combiner une DAG D_1 avec une relation fonctionnelle de type $\boxed{f \rightarrow r}$ et donc de former $P_1(M)$, $F_1(M) \rightarrow M$.

Le processus de transformation de D_1 consiste alors à alourdir de $P_1(M)$ la racine qui contient M, et ensuite à projeter la racine :

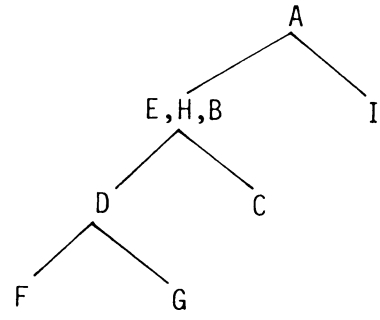
Exemple

D_1 :



et B,E,H → C qui est bien de type $\boxed{f \rightarrow r}$

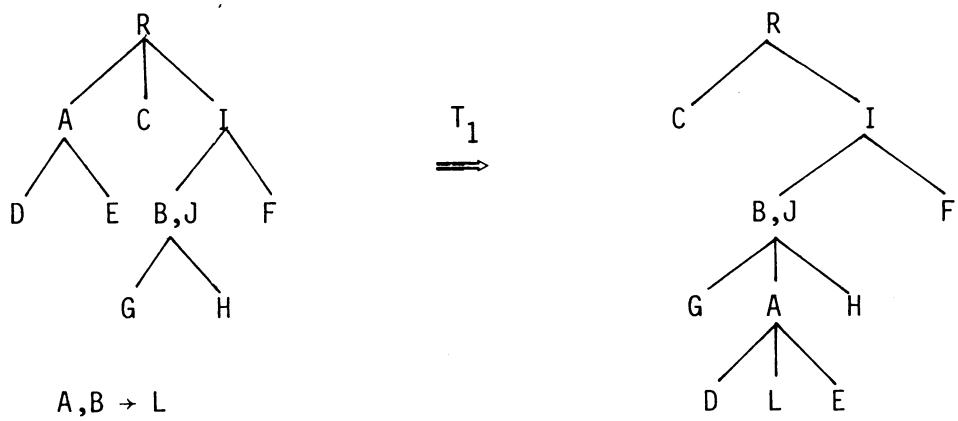
D_2 :



B.2.2.5. Introduction d'un nouveau constituant (P-25)

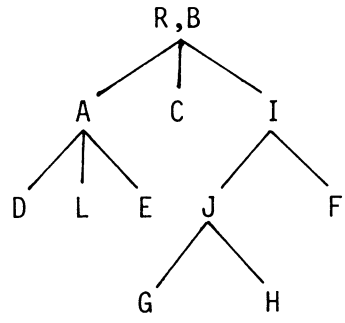
Soit D_1 une DAG et une relation fonctionnelle (A,B → L) où L désigne un constituant n'appartenant pas à l'espace de définition de D_1 ; il y a plusieurs transformations possibles de D_1 qui trouvent facilement leurs justifications aux propriétés et aux méthodes de raisonnement précédentes :

Aussi ne donnons-nous que des exemples de ces transformations :

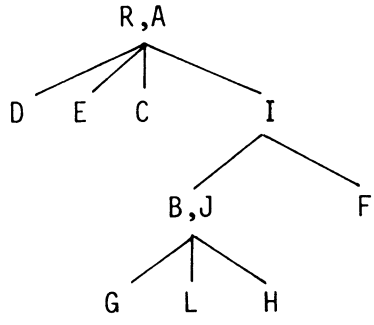


A,B → L

ou bien $\xrightarrow{T_2}$

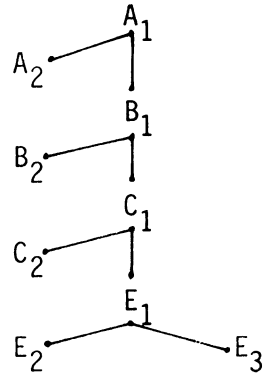


ou bien $\xrightarrow{T_3}$



B.2.2.6. Cas particulier d'interactions de relation fonctionnelle et de DAG (P-26)

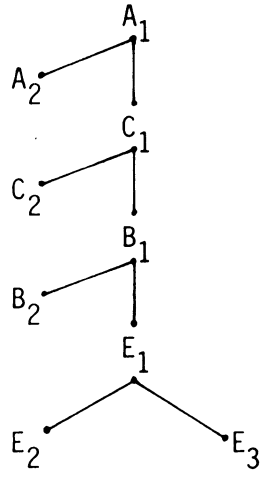
Soit la décomposition arborescente D_1 :



avec relation fonctionnelle suivante :

$$C_1 \rightarrow C_2 \qquad (1)$$

Alors D_2 est également une décomposition arborescente :



Démonstration

A D_1 on peut associer les décompositions arborescentes :

$$A_1 : A_2 | B_1, B_2, C_1, C_2, E_1, E_2, E_3 \qquad (2)$$

$$A_1, B_1 : B_2 | C_1, C_2, E_1, E_2, E_3 \qquad (3)$$

$$A_1, B_1, C_1 : C_2 | E_1, E_2, E_3 \qquad (4)$$

$$A_1, B_1, C_1, E_1 : E_2 | E_3 \qquad (5)$$

Par alourdissement de la racine et élimination de constituants, on obtient à partir de (3) : $A_1,B_1,C_1 : B_2|E_1,E_2,E_3$ (6)

Par (1), on peut en déduire :

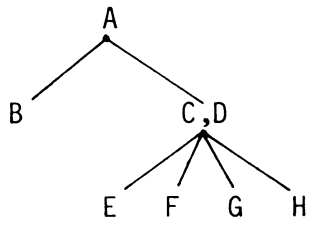
$$C_1 : C_2|A_1,B_1,B_2,E_1,E_2,E_3$$
$$\Rightarrow \quad A_1,C_1 : C_2|B_1,B_2,E_1,E_2,E_3 \qquad (7)$$

(2), (7), (6), (5) forment la décomposition arborescente D_2.

B.2.3. Interactions d'une DAG et d'une autre décomposition arborescente

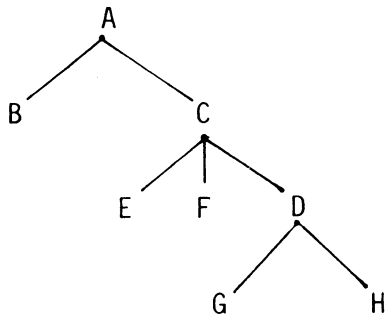
B.2.3.1. Projection d'une racine secondaire (P-27)

Soit D_1 la DAG suivante :



et la décomposition arborescente A,C : E,F|D (1)

Alors D_1 peut se transformer en D_2 :



Démonstration

$$(D_1) \iff \begin{cases} A : B|C,D,E,F,G,H & (2) \\ A,C,D : E|F|G|H & (3) \end{cases}$$

En combinant (1) et (3), on obtient par (P-8) :

$$A,C : D|E|F \qquad (4)$$

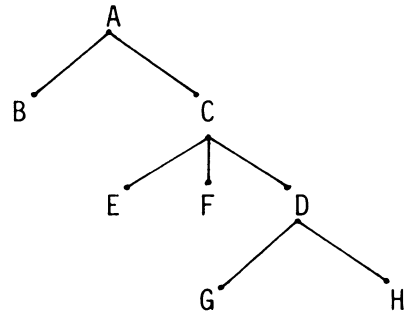
En combinant ensuite (4) et (3), on obtient par (P-10) :

$$A,C : D,G,H|E|F \qquad (5)$$

Par allègement de constituants, (3) devient :

$$A,C,D : G|H \qquad (6)$$

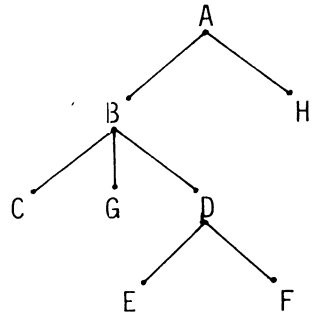
(2), (5), (6) sont les décompositions correspondant à la décomposition arborescente :



(D_2)

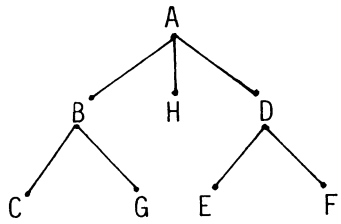
B.2.3.2. Dédoublement d'une racine secondaire (P-28)

Soit D_1 la DAG suivante :



et la décomposition arborescente (0) : A : B|D,E,F ;

alors D_1 peut se transformer en D_2 :



A D_1 sont associées les décompositions arborescentes suivantes :

(1) A : H|B,C,D,E,F,G

(2) A,B : C|D,E,F|G

(3) A,B,D : E|F

(1) et (0) vérifient les conditions de P-11 et donnent

(d) A : H|B|D,E,F ;

(2) et (d) par application de P-10 donne

(a) A : H|B,C,G|D,E,F ;

(2) par allègement de feuilles donne

(b) A,B : C|G

(0) et (3) vérifient les conditions de P-11 et donnent

A,D : B|E|F

et ainsi par allègement de feuilles :

(c) A,D : E|F

(a), (b), (c) sont bien les décompositions arborescentes relatives au noeud du schéma de D_2 ; D_2 est donc bien une DAG.

Liste des principales définitions

Bibliographie

[1] J.R. ABRIAL

"Projet SOCRATE"
Cours avancé sur l'Architecture des Systèmes.
Alpe d'Huez. Décembre 1972.

[2] J.R. ABRIAL

"Data Semantics"
IFIP TC-2 Working Conférence on Data Base Management Systems,
Cargèse Avril 1974.

[3] M. ADIBA, C. DELOBEL, M. LEONARD

"An Unified Approach for Modelling Data in Logical Data base Design".
IFIP Workshop TC2, Freudenstadt, Janvier 1976.

[4] M. ADIBA, M. LEONARD

"Approche algorithmique de la conception d'un système de bases de données".
Rapports TS0, TS1, TS2.
Séminaire de Programmation de Grenoble, Novembre 1975.

[5] P.A. BERNSTEIN, J.R. SWENSON, D.C. TSICHRITZIS

"A unified Approach to Functional Dependancies and Relations".
Proc. 1975 ACM-SIGMOD Workshop on Management of Data, San Jose,
Mai 1975.

[6] **P.A. BERNSTEIN**

Normalization and Functional Dependancies in the Relational Data Base Model.
Thèse, Université de Toronto, Octobre 1975.

[7] **J. BOITTIEAUX - ZIDANI**

"Etude mathématique des relations d'un ensemble de notions"
Contrat DGRST 65 FR 201 (juillet 1967).

[8] **CODASYL**

Data Base tack group report,
ACM, New York, 1971.

[9] **E. CODD**

"Further normalization of the relational data base model"
IBM Research report, RJ 909 (august 1971).

[10] **E. CODD**

"Normalized data base structure : a brief tutorial"
ACM-SIGFIDET Workshop on data description access and control,
San Diego, novembre 1971.

[11] **E. CODD**

"Recent Investigations in Relational Data Base Systems",
Proc. of IFIP 1974, North-Holland.

[12] **C.J. DATE**

"An Introduction to Database Systems".
ADDISON-WESLEY PUBLISHING COMPANY, 1975.

[13] **C. DELOBEL, R.G. CASEY**

"Decomposition of a data base and the theory of boolean switching functions"
IBM RJ. 1016 (avril 1972).

[14] C. DELOBEL
"Contributions théoriques à la conception et l'évaluation d'un système d'informations appliqué à la gestion".
Thèse, Université de Grenoble, Octobre 1973.

[15] C. DELOBEL, M. LEONARD
"The Decomposition Process in a relational model".
IRIA Workshop on Data Structures, Namur, Mai 1974.

[16] C. DELOBEL, D. PORTAL
"Aide algorithmique à l'organisation logique d'un fichier".
Colloque INFORSID-IRIA, Aix-en-Provence, Mai 1975.

[17] C. DELOBEL
"Les Systèmes de Bases de Données.
Ecole d'Eté de l'AFCET 1975.

[18] J.P. FRY, D.W. JERIS
"Towards a Formulation and Definition of data reorganization".
Data Translation Working Paper 801.
Université du Michigan, Octobre 1973.

[19] D. KNUTH
"The Art of Computer Programming".
Vol. 3. Addison Wesley Publishing Company (1973).

[20] J. KUNTZMANN
"Algèbre de Boole".
Dunod, 1965.

[21] W.C. MCGEE
"A Contribution to the study of Data Equivalence".
IFIP TC-2 Working Conference on Data Base Management Systems,
Cargèse, Corse, Avril 1974.

[22] S.B. NAVATHE, J.P. FRY
"Restructuring for large data bases : Three levels of abstraction".
Data Translation Project : Technical Report 8.1, Septembre 1975.

[23] G.M. NIJSSEN
"Set and Codasyl set or coset".
Ecole d'Eté de Freudenstadt, Août 1975.

[24] D. PORTAL
"Conception d'un système automatisé de gestion de la scolarité de l'enseignement supérieur".
Thèse, Université de Grenoble, Mars 1975.

[25] M. TREHEL
"Catalogues, treillis de catalogues, génération et décomposition".
Mathématiques et Sciences Humaines n° 40, 1972.

[26] C.P. WANG, H.H. WEDEKIND
"An Approach for Segment Synthesis in Logical Data Base Design".
IBM Journal of Research and Development, Vol. 19, n° 1, Janvier 1975.