MASTER EN SCIENCES INFORMATIQUES

Élaboration critique d'un schéma conceptuel d'une base de données

Frennet, Marc

Award date:
1977

Awarding institution:
Universite de Namur

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX
NAMUR

INSTITUT D'INFORMATIQUE

Année Académique 1976-1977

ELABORATION CRITIQUE D'UN SCHEMA CONCEPTUEL D'UNE BASE DE DONNEES

MARC FRENNET

Mémoire
présenté en vue de l'obtention
du grade de
Licencié et Maître en Informatique

Je tiens à exprimer ma profonde gratitude à Monsieur F. BODART pour l'intérêt porté à ce mémoire et pour ses conseils très précieux.

J'adresse ma reconnaissance toute particulière à l'équipe du service informatique de la banque PARIBAS de Bruxelles qui m'a aidé à recueillir les données de ce travail.

Qu'il me soit permis de remercier Monsieur A. CABANES, chargé de cours au Conservatoire National des Arts et Métiers de Paris, pour son accueil et sa collaboration durant mon stage.

Je suis également reconnaissant envers Monsieur H. BOGAERT pour ses critiques et conseils.

Je remercie enfin tous ceux qui, par leurs réponses ou par leur aide, ont collaboré à ce travail.

TABLE DES MATIERES

TABLE DES MATIERES

Pages

CHAPITRE VI. INSERTION DU SCHEMA CONCEPTUEL DANS UNE METHODE D'ANALYSE

TROISIEME PARTIE. EVALUATION CRITIQUE: INDEPENDANCE LOGIQUE

CHAPITRE VII. INDEPENDANCE LOGIQUE

INTRODUCTION

Ce mémoire comprend trois parties.

Dans la première, nous présenterons plus particulièrement la notion de schéma conceptuel, ce qui nous permettra de définir la portée exacte de ce travail.

Les autres parties développeront séparément les deux points sur lesquels nous avons décidé de porter notre attention: l'aspect documentaire et l'indépendance logique comme rôles du schéma conceptuel.

PREMIERE PARTIE

Le but de cette première partie est de présenter la notion de schéma conceptuel.

Après avoir constaté l'expansion que prend l'utilisation des bases de données, nous soulèverons le problème de la nécessité d'une méthode d'analyse et de conception des bases de données.

La décomposition en niveaux, partant de l'organisation des données pour aboutir à leur implémentation physique, nous permettra de situer la structure conceptuelle dans tout le processus d'analyse. Nous exposerons alors les concepts proposés par le modèle de BENCI, BODART, BOGAERT et CABANES pour la description formelle de l'organisation conceptuelle. Cette approche sera comparée à d'autres modèles conceptuels. Après avoir relevé les rôles fondamentaux associés à ce schéma conceptuel, nous définirons exactement l'objet de ce travail, à savoir une application critique de ce schéma du point de vue de son aspect documentaire et de son apport vis-à-vis de l'indépendance des données.

Pour effectuer cette critique, nous nous baserons sur l'exemple d'un organisme bancaire qui sera décrit dans le troisième chapitre.

CHAPITRE I. INTRODUCTION

Dans cette introduction, nous mentionnerons les avantages fournis par l'utilisation des bases de données, justifiant ainsi leur expansion. Après avoir rappelé les grandes fonctions des "Systèmes de Gestion de Bases de Données" (SGBD), nous poserons le problème de la nécessité d'une méthode d'analyse et de conception des bases de données. Nous présenterons ensuite, de façon succinte, l'objet de ce travail.

I.1. L'expansion des bases de données.

Dans les systèmes traditionnels, de nouvelles applications engendrent bien souvent leurs propres fichiers, leurs propres programmes. L'introduction d'une base de données dans un organisme va à l'encontre de cette façon de faire en rendant possible la centralisation, la coordination, la diffusion de l'information. Elle permet également l'amélioration de la cohérence des informations, la réduction des redondances, la facilité de saisie et de mise à jour des données.

L'expansion des bases de données et l'engouement vis-à-vis de ce nouvel outil s'expliquent quand on examine la situation devant laquelle se trouve un gestionnaire d'une organisation dont la taille est sans cesse croissante et dont les intéractions avec un environnement évolutif se font de plus en plus nombreuses. Il doit faire face, parfois dans des délais très courts, à des situations multiples et diverses, aidé en cela par différents moyens d'action parmi lesquels il doit effectuer certains choix.(1)

---

(1) cfr. ouvrage repris dans la bibliographie sous le numéro 4.

La représentation de son entreprise qu'il perçoit sous forme de sous-structures, peut faliciter la résolution de ces problèmes, si elle correspond à la réalité qu'il tente de modéliser. Elle doit en outre faire apparaître les relations existant entre les informations. La prise en compte de ces relations est un des avantages que peuvent fournir les bases de données.

I.2. Les Systèmes de Gestion de Bases de Données (SGBD).

Les bases de données se substituent aux fichiers traditionnels, les SGBD se substituent aux systèmes de gestion de fichiers. Ils ont pour but de fournir aux différents utilisateurs les outils leur permettant d'exploiter des structures d'informations complexes, et cela, dans un contexte d'indépendance des données vis-à-vis des programmes d'application.(1)

Les grandes fonctions d'un SGBD sont les suivantes (2):

-la description des données (forme logique des données);

-la gestion physique des données;

-la gestion des échanges des données avec l'utilisateur.

I.3. Nécessité d'une méthode d'analyse et de conception.

Le problème nous semble assez bien résumé dans l'introduction de l'article "Sémantique et structures de données" de Flory A. et Kouloumdjian J. (3). Si les concepteurs ont développé des SGBD très complets et fourni aux utilisateurs toutes les caractéristiques de leurs créations en vue de son exploitation, ils ne se sont guère penchés sur le problème de les munir de méthodes leur permettant d'assurer une utilisation adéquate de l'outil proposé. Celle-ci ne peut être atteinte

---

(1) cfr. l'ouvrage repris dans la bibliographie sous le numéro 8.
(2) cfr. le cours de Monsieur Cabanes, repris dans la bibliographie sous le numéro 2.
(3) cfr. l'article de Flory A. et de Kouloumdjian J., repris dans la bibliographie sous le numéro 5.

que si l'utilisateur dispose des moyens nécessaires pour définir correctement son système d'informations, conduisant à une implantation optimale par rapport aux traitements effectués sur les données.

Le principal reproche que l'on puisse faire à l'égard des méthodes d'analyse traditionnelles, c'est qu'elles ne tiennent pas entièrement compte des relations existant entre les informations, se limitant dans de nombreux cas aux règles de calcul entre données. Il est apparu nécessaire de développer une méthodologie d'analyse et de conception des bases de données.

I.4. Objet du travail: application critique d'un schéma conceptuel.

Sans entrer dans des détails pour l'instant, disons que le schéma conceptuel comprend la première modélisation du système d'informations ainsi que l'expression des traitements à faire subir aux données. Le but de ce travail sera de critiquer ce schéma du point de vue de son aspect documentaire et de son apport vis-à-vis de l'indépendance logique.

Le développement de la notion de schéma conceptuel dans le second chapitre nous permettra de fournir quelques précisions quant à la portée de ce travail.

CHAPITRE II. SCHEMA CONCEPTUEL

Une des démarches actuelles dans la conception des systèmes d'informations prône un découpage de l'analyse en niveaux: la présentation de celle-ci nous permettra de situer le niveau conceptuel dans l'ensemble du processus d'analyse. Nous donnerons alors une définition précise du schéma conceptuel proposé par Benci E., Bodart F., Bogaert H., Cabanes A. (B.B.B.C.) dans le cadre de l'article "Concepts for the design of a conceptuel schema". Cette approche sera ensuite comparée à d'autres modèles conceptuels. La mise en évidence des différents rôles de ce schéma nous amènera à une définition plus explicite de la portée de ce travail.

II.1. Démarche actuelle: décomposition en niveaux.

Les définitions de chacun de ces niveaux sont tirées des ouvrages repris dans la bibliographie sous les numéros 1., 2. et 3.

II.1.1. Le réel perçu.

Le réel perçu correspond à cette partie du réel, c'est-à-dire de l'organisation et de son environnement, qui est perçu par cette même organisation. Son support linguistique est le langage naturel.

Le réel perçu est l'ensemble des modèles ou représentations qui aident à la conception, la structuration et le contrôle des objets du monde réel. L'information de chacun de ces modèles peut être décrite au moyen de trois concepts de base: les objets (ce qu'un individu ou un groupe voit comme un tout), les associations (ensemble de deux ou plusieurs

objets où chacun assume un rôle donné) et les propriétés de ces objets et associations.(1)

II.1.2. Structure conceptuelle.

Nous appellerons organisation conceptuelle le résultat d'une phase d'analyse et de modélisation du système. C'est une image aussi complète que possible des phénomènes d'organisation qu'il est intéressant de représenter par un système d'informations...

L'organisation conceptuelle comprend d'une part, la structure conceptuelle dont le but est de décrire l'ensemble des données et leurs propriétés(contraintes d'intégrité), et d'autre part, les règles d'évolution correspondant à l'ensemble des opérations qui modifient l'état de la base de données(elles décrivent l'interaction du système d'informations avec l'environnement).

Le niveau structurel constitue une étape d'analyse du réel perçu qui est indépendante du SGBD choisi pour l'implémentation. Il marque le point d'accord entre les analystes et les programmeurs quant au contenu de la représentation. Il joue le rôle de cahier des charges pour l'implémentation et constitue un point de référence dans l'organisation : toute modification de la base de données doit être répercutée sur la structure conceptuelle.

II.1.3. Structure d'accès.

Une structure d'accès est une représentation de la structure conceptuelle, manipulable par un programme. On s'intéresse, à ce niveau, non seulement à la déclaration d'existence d'informations, mais aussi à la façon dont on peut accéder à celles-ci. Les chemins d'accès décrits sont les seuls auxquels le programmeur peut faire appel. On y décrit également la manière dont certaines propriétés qualitatives vont être vérifiées(ex: un client possède toujours au moins une adresse). C'est à ce niveau que s'effectue le choix des informations contenues dans la base de données.

---

(1) Nous reprendrons ces définitions au paragraphe II.2.1.1., point a).

II.1.4. Structure physique.

La structure physique est l'implémentation, sur le matériel dont on dispose, de la structure décrite au niveau précédent. A ce stade de réalisation on choisit:-les méthodes de codage de l'information;

-la façon dont est gérée la mémoire secondaire;

-la façon dont doivent être réglés les problèmes de concurrence entre accès à une même donnée;

-la façon dont les accès sont réalisés (pointeurs, hash-code,...).

II.2. Définition du schéma conceptuel. (1)

II.2.1. Le modèle B.B.B.C.

La présentation de l'organisation conceptuelle comme le résultat d'un processus d'analyse et de modélisation du système nous conduit à démarrer la structuration des informations en partant de la perception que nous avons de celles-ci au niveau du monde réel. Par après, la structure d'informations obtenue sera intégrée et représentée par un schéma conceptuel, c'est-à-dire la description de la structure conceptuelle à l'aide d'un langage donné.

Le modèle conceptuel qui nous est proposé se place donc à deux niveaux :-il suggère un ensemble de concepts qui sont à la base de la structuration du monde réel perçu.

-il nous livre un ensemble de concepts (formalisme) utilisés pour la représentation de la structure conceptuelle.

Le mapping entre ces deux ensembles de concepts est destiné à montrer l'interaction existant entre la structure du réel perçu(sémantique) et sa représentation.

Après avoir défini ces concepts relatifs à la structuration des données (statique), nous nous pencherons sur ceux qui vont nous

(1) Ce chapitre est un résumé des concepts du modèle conceptuel proposé dans l'article repris dans la bibliographie sous le numéro 1.

permettre de décrire l'aspect dynamique du système (différentes opérations sur la base).

II.2.1.1. Concepts relatifs à la structuration des données.

a) Perception du réel.

Les concepts pour décrire le monde réel perçu sont les suivants:

-Les objets:

"Un objet est ce qu'un individu ou un groupe voit comme un tout, ayant une existence propre. Un objet est caractérisé par un ensemble de propriétés quantitatives et qualitatives et un comportement permanent; le nombre de propriétés et la permanence sont fonction du point de vue choisi".

Exemple: un client, une agence, un produit
propriété quantitative: il y a environ 4000 clients;
propriété qualitative: un client est toujours attaché à au moins une agence.

-Les associations:

"Une association est un ensemble de deux ou plusieurs objets où chacun assume un rôle donné. Elle peut posséder plusieurs propriétés. Son existence est contingente à l'existence des objets qu'elle relate".

Exemple: un compte est une association entre un client, une agence et un produit.

-Les propriétés:

"Une propriété appartenant à un objet ou une association est une qualité que les individus attribuent à cet objet ou association. L'existence de ces propriétés est contingente à l'exis-

tence de l'objet ou de l'association concernée".

Exemple: numéro de matricule du client, date de naissance, numéro de l'agence,...

-La notion de type:

Les objets, les propriétés et les associations constituent les concepts de base qui sont à l'origine d'une notion plus large : la notion de type. Nous la définirons simplement comme suit:

-type d'objets: c'est un ensemble d'objets.

Exemple: Le type PERSONNE est l'ensemble des individus que l'on peut désigner par leur nom: Dupont, Durant,...

-type de propriétés: c'est un ensemble de propriétés.

Exemple: couleur={jaune, vert, bleu,...}

L'espace de définition de cet ensemble est toujours défini par l'ensemble d'objets et/ou d'associations auquel l'ensemble de propriétés est attaché.

-type d'associations: c'est un ensemble d'associations. L'espace de définition de cet ensemble est toujours constitué par les ensembles d'objets définis par l'ensemble des associations.

b) Représentation formelle.

-Formalisme relationnel.

Plusieurs formalismes peuvent être utilisés pour décrire l'organisation conceptuelle. Bon nombre de travaux récents montrent que le formalisme relationnel, par sa simplicité et sa généralité, convient remarquablementpour la description d'une base de données conceptuelle.

Rappels:Soient les ensembles A1,A2,...,An; une relation R(A1,A2,...,An) définie sur ces ensembles est une partie du produit cartésien de ces ensembles. Un n-uple (a1,a2,...,an) de cette relation est tel que a1$\in$A1,a2$\in$A2,...,an$\in$An.

-Représentation des objets, des associations, des propriétés.

L'existence d'un objet ou d'une association est représentée dans la structure conceptuelle par un n-uple d'une relation. Cependant, un problème d'identification se pose lorsque plusieurs n-uples peuvent contenir les mêmes données dans une relation. La relation PERSONNE (nom, prénom, ville de résidence) est ambiguë s'il existe deux Dupont Joseph habitant Namur. Un mécanisme d'identification est nécessaire pour représenter de façon non ambiguë les objets du monde réel. Dans le cas où les valeurs des propriétés repèrent de façon non équivoque les objets, ceux-ci seront identifiés au niveau de la structure conceptuelle par les données représentant les valeurs des propriétés. Dans le cas contraire, on créera le concept d'entité propre à la structure conceptuelle. Une entité est une composante d'une relation. La valeur de cette composante n'est pas connue dans l'organisation, mais elle est donnée par un mécanisme décrit dans la base de données conceptuelle.

Les concepts de base pour la description de la structure conceptuelle sont donc au nombre de 3:

- -les types de données: NOMATR "numéro de matricule"
  DATNAIS "date de naissance";
- -les types d'entités;
- -les relations.

Nous pouvons effectuer la correspondance entre ces concepts et ceux utilisés au niveau du monde réel perçu. (1)

(1) Pour plus de détails se référer à l'exemple traité au chapitre IV.1.

monde réel perçu		structure conceptuelle
-un type de propriétés (numéro de matricule)	est représenté par	-un ou plusieurs types de données (NOMATR)
-un type d'associations (association compte)	est représenté par	-une ou plusieurs relations (relation COMPTE(...))
-un type d'objets (objet client)	est représenté par - est identifié par -	-une ou plusieurs relations (relation CLIENT(...)) -un type d'entités qui l'identifie uniquement au niveau conceptuel
-une valeur de propriété	est représentée par	-une donnée

-Les contraintes d'intégrité.

Dans le cadre du schéma conceptuel, les contraintes d'intégrité sont des prédicats relatifs aux données de la base , définis au niveau d'éléments de la structure fonctionnelle en vue de fournir à l'utilisateur des données correctes comme résultat de ses programmes d'application. Par données correctes, nous entendons qu'elles doivent être conformes aux propriétés du monde réel qu'elles représentent, propriétés prévues par l'utilisateur en fonction du modèle qu'il s'est donné de celui-ci. Le problème de l'identification des contraintes relève de l'analyse du monde réel; la problématique des contraintes d'intégrité, posée dans le contexte d'un schéma conceptuel, se limite essentiellement à un problème d'outil (de langage), d'expression de celles-ci. Nous examinerons ci-après, les caractéristiques des contraintes d'intégrité les plus fréquemment rencontrées.

-contraintes portant sur les valeurs possibles d'un type de données, lorsque ces valeurs ne sont pas fonction d'autres types de données de la base. (énumération, définition des bornes de l'ensemble , définition de la fonction de calcul de l'ensemble des valeurs)

-spécification du (des) format(s) d'enregistrement d'un type de données. (codage(s) utilisé(s) dans l'organisation)

-cardinalité: caractérisation du nombre des valeurs possibles

(donné, minimum, maximum, moyen, ...) d'un type de données dans la base fonctionnelle.

-composantes obligatoires d'une relation: les valeurs de ces composantes ne peuvent être inconnues.

-spécification des valeurs d'un domaine d'une relation, conditionnellement aux valeurs d'un ou de plusieurs autres domaines.

-contraintes de cardinalité spécifiant le nombre de n-uples d'une relation, ou le nombre de valeurs possibles d'un domaine de la relation conditionnellement aux valeurs d'un ou plusieurs domaines de la relation.

-relation fonctionnelle entre domaines d'une relation.

-clé, index d'une relation .

-contrainte spécifiant que l'ensemble des valeurs d'un ou de plusieurs domaines est inclus ou est égal à l'ensemble des valeurs d'un ou de plusieurs domaines d'une autre relation.

II.2.1.2. Concepts relatifs à la dynamique du système.(1)

Les règles d'évolution coïncident avec les opérations élémentaires sur la base: création, modification, suppression, adjonction. Ceconcept est nécessaire pour décrire la dynamique du système d'informations : la description d'une application sera constitué par un ensemble, une combinaison de règles d'évolution. Examinons les éléments qui permettent de définir ces règles d'évolution.

a) Contexte de cohérence d'une application.

Soit un ensemble de prédicats s'appliquant à tout ou partie de la base, ou à des éléments de l'organisation environnante. Une application s'exécutant sur la base nécessite qu'une partie seulement des prédicats soit vraie : cet ensemble sera appelé 'contexte de cohérence de l'application'.

On peut classer les contraintes d'intégrité en trois groupes: les contraintes d'intégrité externes, internes, mixtes.

-Externes: elles ne s'appliquent qu' aux paramètres

(1) Nous exposons brièvement les concepts relatifs aux règles d'évolution; leur description détaillée sera donnée au chapitre V.

d'entrée de l'application, c'est-à-dire aux données ne figurant pas dans la base et qui sont nécessaires à l'application.

Exemple: enregistrement d'un mouvement(montant des opérations) sur un compte.
La contrainte relative au format du numéro de compte(paramètre d'entrée de l'application) est une contrainte externe.

-Internes: les contraintes d'intégrité internes ne s'appliquent qu'aux données déjà enregistrées dans la base; elles doivent être vérifiées au moment du déclenchement de l'application.

Exemple: les contraintes relatives à la définition des clés des relations, à la détermination des relations fonctionnelles entre domaines de relations,... sont des contraintes internes.

-Mixtes: les contraintes d'intégrité mixtes s'appliquent à la fois aux paramètres de l'application et aux données de la base.

Exemple: vérifier que le numéro de compte entré comme paramètre existe déjà dans la base.

b) Notion d'événement.

Les contraintes d'intégrité s'inscrivent activement dans les séquences "décision - exécution d'actions sur la base", soit qu'une décision provoque l'activation d'une contrainte, soit qu'une contrainte enclenche une décision(déclenchement d'une procédure consécutive au fait que la contrainte n'est pas vérifiée). Nous appellerons événement toute décision qui produit l'exécution d'opérations sur la base. Un événement est soit au repos, soit activé. L'activation d'un événement peut être due soit à l'organisation environnante(événément d'entrée: décision de calcul des intérêts d'un compte à vue à une date déterminée), soit à un état particulier de la base. Dans ce dernier cas, on distinguera les événements qui déclenchent des opérations internes à la base (événement interne: procédure d'erreur), de ceux qui sont connus de l'organisation(événement de sortie: édition de messages, fin de traitement).

Dans la plupart des cas, des paramètres sont associés à un événement. Ainsi par exemple, l'événement "décision de calcul des intérêts d'un compte à vue à une date donnée" est accompagné du numéro de compte et de la date en question.

Une application ou procédure pourra être déclenchée par des états d'activation de plusieurs événements simultanément, et éventuellement par plusieurs groupes d'événements. Notons également que la fin d'une procédure peut être considérée comme un événement.

II.2.2. Situation du modèle B.B.B.C. par rapport à d'autres modèles conceptuels.

II.2.2.1. Le modèle individuel. (1)

Le rapport de synthèse de l'étude menée par un groupe de chercheurs français sur "une méthodologie d'analyse et de conception d'une base de données" propose les fondements d'un schéma conceptuel de référence appelé modèle individuel. Il se base sur quatre espaces fondamentaux:

1) L'espace I des individus: un individu étant une abstraction représentant une collection d'êtres.
   Exemple: I={client, produit, agence, voiture,...}
2) L'espace P des propriétés: ou type de données.
   Exemple: P={numéro de matricule, nom, couleur,...}
3) L'espace V des valeurs: ou données.
   Exemple: V={55, Durant, rouge,...}
4) L'espace R des relations: une relation unit 2, 3,...,n individus.
   Exemple: RELATION(CLIENT, AGENCE, PRODUIT)= COMPTE.

On définit quatre applications sur l'espace des individus et sur l'espace des relations.

1) ENTITE: l'application ENTITE associe à un individu ou à une relation un ensemble de propriétés, partie de l'espace des propriétés.
   Exemple: ENTITE(CLIENT)= numéro de matricule, nom, date de naissance,...
   ENTITE(COMPTE)= numéro de compte, solde,...

(1) cfr. ouvrage repris dans la bibliographie sous le numéro 8.

2) PROJECTION: la projection permet d'associer à un individu ou à une relation la valeur que prend une caractéristique pour une occurrence donnée.
Exemple: PROJECTION $_{nom}$(CLIENT 25)= DURAND

3) REALISATION: l'application REALISATION associe à une occurrence d'un individu ou d'une relation les valeurs de leurs différentes caractéristiques.
Exemple: REALISATION(CLIENT $_{25}$)= 010239, DURAND, 01/08/1952

4) IDENTIFIANT: l'identifiant d'un individu ou d'une relation est l'équivalent de la notion de clé d'une relation dans le modèle B.B.B.C.
Exemple: IDENTIFIANT(CLIENT)= numéro de matricule

Nous pouvons donner un exemple de structuration des données:

CLIENT: INDIVIDU
IDENTIFIANT(CLIENT)= NOMATRICULE
ENTITE(CLIENT)= NOMATRICULE, NOM, ADRESSE
AGENCE: INDIVIDU
IDENTIFIANT(AGENCE)= NOAGENCE
ENTITE(AGENCE)= NOAGENCE, TYPEAGENCE, ADRESAGENCE
PRODUIT: INDIVIDU
IDENTIFIANT(PRODUIT)= TYPEPRODUIT
ENTITE(PRODUIT)= TYPEPRODUIT, PRIXREVIENT
COMPTE: RELATION(CLIENT, AGENCE, PRODUIT)
IDENTIFIANT(COMPTE)= NOMATRICULE,NOAGENCE, TYPEPRODUIT
ENTITE(COMPTE)= NOMATRICULE, NOAGENCE, TYPEPRODUIT,SOLDE

Comme pour le modèle B.B.B.C., on a la possibilité d'exprimer les contraintes d'intégrité qui s'appliquent sur les espaces I, R, P, V. Elles portent sur les valeurs d'une propriété d'un individu, sur les valeurs de plusieurs propriétés d'un individu, sur les relations entre individus.

Les seuls objets indépendants, manipulables, ayant une existence propre sont les individus. La relation, pour être manipulée, doit être reliée aux individus qui la composent; elle n'a d'existence que si les individus qu'elle relie existent.

Nous remarquons aisément qu'il existe certaines analogies entre le modèle individuel et le modèle B.B.B.C. Dans ces deux modèles nous voyons apparaître les notions "d'objet" ou "d'individu" et les notions "d'association" ou "de relation entre individus". La différence primordiale réside dans le fait que le modèle individuel les utilise pour la représentation formelle du système d'informations, le modèle B.B.B.C. pour la perception du réel. Pour ce dernier, seul le concept de relation (au sens de Codd) suffit à la description formelle de la structure conceptuelle. La conclusion de cette étude sur une méthodologie d'analyse et de conception d'une base de données porte d'ailleurs sur cette perception du réel: "...Il nous semble utile de signaler un nouveau besoin en matière de méthodologie concernant le passage du réel au réel perçu; en effet, il serait paradoxal d'appliquer une méthode qui se veut rigoureuse(représentation sous forme d'individus et de relations entre individus) à des bases totalement empiriques...". Le modèle B.B.B.C. remédie à cet inconvénient en proposant une structuration du réel sous forme d'objets et d'associations, facilitant ainsi la construction du schéma conceptuel. A noter également que le modèle individuel ne parle à aucun moment de la dynamique du système d'informations.

II.2.2.2. Le modèle "entité - association". (1)

Le modèle de Peter Pin-Shan Chen est très proche du modèle B.B.B.C. Pour ce qui est de la perception du monde réel, les concepts proposés dans les deux modèles sont identiques(objet - entité, et association entre objets - entité). Le modèle "entité-association" spécifie explicitement le rôle joué par chaque entité entrant dans la constitution des associations. Les entités et les associations de P. Pin-Shan Chen possèdent des attributs(propriétés du modèle B.B.B.C.) qui peuvent prendre différentes valeurs. Le passage à la description formelle du système d'informations s'effectue en maintenant(comme pour le modèle individuel) la distinction entre entité et association; les éléments de base de cette représentation formelle sont les relations "entité" et les relations "association". Pour chacune d'entre elles, on spécifie également la ou les clés(nécessité de

(1) Ce qui suit est tiré de l'ouvrage repris dans la bibliographie sous le numéro 14.

déclarer une clé principale, appelée clé primaire). On inclut également dans la description des données la notion de dépendance: l'existence d'une entité dépend de l'existence d'autres entités.

Exemple: L'existence d'un compte dépend de l'existence du client qui le détient, de l'agence qui en assure la tenue.

En ce qui concerne la dynamique du système, il analyse le passage des opérations élémentaires (1) sur les données du monde réel aux opérations élémentaires sur la représentation de ces données dans la structure conceptuelle.

II.3. Rôles fondamentaux du schéma conceptuel.(2)

II.3.1. Outil de documentation.

II.3.1.1. Point d'accord entre analystes et utilisateurs.

L'organisation conceptuelle a pour but de représenter de façon complète, exhaustive les catégories de phénomènes que les différents membres de l'organisation désirent voir figurer dans la base de données. La démarche d'analyse se fait par discussions et échanges entre les analystes et les utilisateurs de manière à intégrer, dans un schéma conceptuel unique, les vues que ces derniers ont des problèmes qui sont de leur ressort. Une structure conceptuelle ne pourra prétendre représenter un certain monde réel (celui des utilisateurs) que lorsque l'accord complet sera réalisé sur son contenu.

De plus, l'organisation conceptuelle met en évidence les événements du monde réel qui déclenchent les mises à jour des données mémorisées; elle décrit les séquences d'opérations déclenchées par les événements externes et qui conduisent à la modification des données.

---

(1) Création, suppression, modification des entités et associations.

(2) Tirés des ouvrages repris dans la bibliographie sous les numéros 1.et2.

II.3.1.2. Cahier des charges pour l'implémentation.

Lorsque l'implémenteur prend en charge la réalisation de la base dans un certain software, il doit connaître tous les paramètres qui motiveront ses choix sans faire de retour aux utilisateurs, sans repasser par une phase d'analyse et de modélisation du système. Dans le schéma conceptuel, nous devrons donc retrouver la description de toutes les propriétés des informations qui sont susceptibles d'influencer les prises de décision de l'implémenteur.

II.3.2. Outil d'indépendance logique.

"L'adaptabilité est la qualité du système qui lui permet de s'adapter aux modifications de la structure même de la base de données. Ces modifications peuvent être la conséquence d'une évolution des structures et des besoins de l'organisation de l'entreprise, ou bien la conséquence d'un changement de technologie. Le concept d'indépendance logique est la caractéristique d'un système de base de données qui permet d'assurer un certain degré d'adaptabilité..."(1).

Pour atteindre cet objectif, une des solutions est de séparer le niveau logique et le niveau physique. On imagine sans peine le coût de réécriture des programmes écrits sur une structure d'accès dont les caractéristiques ne satisfont plus l'ensemble des utilisateurs. Le schéma conceptuel est un outil d'indépendance logique en ce sens qu'il permet aux différents utilisateurs, d'exprimer leurs besoins sur une structure relativement stable, indépendante de toute notion d'accès.

La notion de mapping d'une structure dans une autre est associée à celle d'indépendance logique; il y aura transformation de la structure conceptuelle(logique) en une structure d'accès par application de règles d'équivalence entre les différents éléments qui les composent. De même, les traitements décrits sur le schéma conceptuel seront transformés en programmes exécutables sur la structure d'accès. La modification de cette dernière n'entraînera plus nécessairement la modification des programmes: ce sont les changements opérés au niveau du mapping qui sauvegarderont l'intégrité des procédures décrites.

(1) Tiré de l'ouvrage repris dans la bibliographie sous le numéro 15.

II.4. Redéfinition de la portée du travail.

L'application critique que nous nous proposons d'effectuer comprendra deux parties qui porteront sur les rôles fondamentaux assignés au schéma conceptuel: outil de documentation et outil d'indépendance logique.

Le modèle B.B.B.C. propose uniquement un ensemble de concepts pour décrire un schéma conceptuel. Pour ce qui est de l'aspect "documentation", nous procèderons dans un premier temps à une critique des concepts qui permettent la structuration des informations(point de vue statique) et la description des règles d'évolution(point de vue dynamique). Dans un second temps, nous proposerons certains outils qui pourront être d'une aide précieuse pour atteindre le but fixé. Pour ce qui est du caractère d'indépendance logique, nous montrerons à partir d'exemples, comment il est possible de passer de la description d'une structure conceptuelle à une structure d'accès. Après s'être donné des langages de manipulation de données sur ces deux structures, nous indiquerons quelques principes qui régissent la transformation de traitements exprimés sur la première en procédures exécutables sur la seconde.

CHAPITRE III. CADRE DE L'APPLICATION

A titre d'exemple, un organisme bancaire nous servira de support pour la réalisation de l'application critique du schéma conceptuel. Après avoir exposé brièvement l'historique et la tendance actuelle de l'informatique dans le milieu bancaire, nous soulignerons l'intérêt de l'introduction des bases de données dans ce domaine. Nous inclurons également dans ce chapitre les limites de notre description ainsi que quelques remarques concernant la terminologie employée.

III.1. Le cadre du stage.

En vue d'illustrer les concepts proposés pour la description du schéma conceptuel, nous nous baserons sur le stage de deux mois effectué dans un organisme bancaire. La banque PARIBAS (Belgique) a mis sur pied une base de données implémentée en IDS qui est en place depuis trois ans environ.

Le but du stage était de prendre connaissance de leur réalisation, d'appliquer les principes du modèle B.B.B.C. à une démarche personnelle et finalement d'évaluer celle-ci par rapport à la leur par comparaison des résultats obtenus.

III.2. L'informatique et la banque: historique et tendance actuelle.

Les banques sont les entreprises qui, dans les premières, ont senti le besoin de s'informatiser: étant donné le développement du mouvement bancaire, la multiplicité des clients, la diversité des "produits" vendus, le nombre toujours croissant des transactions, le service administratif s'est rapidement vu submerger. Pour remédier à une telle situation, on procéda en premier lieu à l'automatisation du niveau

opérationnel: gestion des comptes des clients(arrêté du compte, position, envoi des extraits de compte,...), gestion des comptes de la banque, opérations concernant les moyens de paiement et de recouvrement(chèques, virements, effets,...).

Jusqu'il y a peu, les responsables de la production disposaient d'un temps suffisant pour faire parvenir au service de gestion les données qui leur étaient nécessaires. Mais les volumes continuant à croître, les services rendus se compliquant de plus en plus, l'intensification de la concurrence , la fréquence des changements dans l'économie et dans les marchés font que la situation continue à se dégrader. Au plan opérationnel, on ne parvient plus à fournir aux responsables de la gestion des données rigoureusement exactes, en quantité adéquate, et cela dans des délais suffisamment courts.

III.3. Les bases de données: une nécessité.

L'emploi de personnel plus nombreux pour pallier cet état de choses n'est qu'une solution temporaire et, de plus, elle risque de créer d'autres problèmes au niveau du contrôle de comportement d'un ensemble trop complexe. Si des modèles de gestion automatisés(modèles de trésorerie, d'études de marchés,...) ont été conçus pour faciliter les prises de décision des gestionnaires, ils présentent le désavantage que nous avons déjà mentionné, à savoir la non prise en compte des relations existant entre les informations. Les bases de données réalisent l'intermédiaire indispensable entre le niveau opérationnel et le niveau stratégique en enregistrant les interdépendances au niveau informatique.

Nous pouvons illustrer ces propos par quelques problèmes ou questions qu'un banquier peut se poser à un moment donné.

-Quels sont les marchés rentables?
-Quels sont les clients rentables?
-Quels sont les volumes des dépôts et des crédits?
-Comment orienter la politique des crédits en fonction de la politique de récolte des capitaux et vice versa?
-Une telle agence est-elle encore rentable vu le type de clients qu'elle possède et les services que ceux-ci lui demandent?

-Quel est prix de revient de chaque produit?
-Est-il avantageux de promouvoir de nouveaux produits étant donné le marché actuel?
-Sommes-nous à un moment propice pour effectuer une pénétration géographique, pour implanter de nouvelles agences?
-Dans telle région, les clients déposent leurs fonds chez nous mais demandent le crédit ailleurs: quelles sont les causes d'une telle situation et comment y faire face?
-Quel est l'état de l'octroi des crédits pour toutes les agences de Wallonie?
-Quel est l'état de tous les comptes de tel client?
-Quels types de client ont l'habitude de s'ouvrir un tel genre de compte?

Nous reviendrons sur ces problèmes dans un chapitre ultérieur et nous essaierons de montrer comment l'approche "base de données" facilite leur résolution.(1)

III.4. Les limites de la description.

Les documents mis à notre disposition furent les suivants:

-Le schéma IDS de la base de données.
-Le contenu du fichier central à implanter: description des articles, désignation symbolique des données, format et clé des articles, type des articles.
-Les fiches "data base" : dossier récapitulatif concernant le client à destination du secrétariat administratif.
-Les documents relatifs à la création, modification, suppression et consultation de certains articles.
-La liste des contrôles à effectuer sur les éléments constitutifs des articles.
-Possibilité de consulter toutes les applications pour lesquelles il existe des documents écrits.

C'est à l'aide de tous ces documents et des entretiens avec l'équipe du service informatique que je me suis représenté ce que pouvait être le fonctionnement d'une banque.

Il paraît assez évident de dégager trois sous-structures qui sont étroitement liées: les CLIENTS, les AGENCES et les PRODUITS. Nous limiterons la description de la banque à ces trois éléments. Pour les agences, nous en donnerons les caractéristiques propres, leur(s) adresse(s), les services qui les composent. Pour un client, nous aurons ses éléments signalétiques ainsi que son (ses) adresse(s). En ce qui

---

(1) cfr. chapitre VIII.

concerne les produits, nous avons sélectionné les comptes dépôt, les crédits, les devises, la location des coffres, les cartes de banque, les chèques. Les valeurs, les titres et toute la gestion des portefeuilles n'ont pas été abordés uniquement dans le but de ne pas allonger inutilement la description.(1)

La partie "règles d'évolution" décrite dans l'exemple s'insère dans l'application relative aux comptes à vue.(1) Selon la terminologie utilisée dans le cours de MIS de Monsieur Bodart (2), nous appellerons ces traitements "phase d'enregistrement des mouvements" et "phase de calcul des intérêts sur les comptes à vue".

III.5. Remarques sur la terminologie employée.

Si on examine le flux réel de la banque, certaines ambiguïtés apparaissent. On y distingue les flux approvisionnement, production et vente: le premier consiste en l'apport de capitaux venant de l'extérieur, le troisième comprend l'octroi des crédits et les retraits de fonds mais il est courant de voir les mêmes postes de travail effectuer ces deux opérations. La distinction se fera dans l'analyse simplement à un niveau conceptuel.

Dans le même ordre d'idées, une personne se présentant à un guichet de la banque peut être tantôt un client(demande de crédit, location de coffre, gestion de portefeuille), tantôt un fournisseur(dépôt de fonds). Nous voyons donc qu'une même personne change de qualificatif suivant la nature de sa démarche . Pour notre part, nous ne ferons aucune distinction et nous parlerons uniquement en terme de client.

Il serait peut-être plus logique de décomposer la banque en fonction de ses activités plutôt qu'en fonction de l'approvisionnement et de la vente. On ne parlera plus alors de produit mais bien de services rendus par la banque. Le client exprime des demandes de services, on les exécute moyennant un certain intérêt (vente). Nous avons pris l'optique de parler en termes de "clients" et de "produits": c'est dans leurs associations et surtout dans les caractéristiques de ces associations que nous retrouverons la notion de "services".

(1) Pour plus de détails, cfr. annexes I,II et III.
(2) cfr. ouvrage repris dans la bibliographie sous le numéro 4.

DEUXIEME PARTIE

EVALUATION CRITIQUE : ASPECT DOCUMENTAIRE DU MODELE CONCEPTUEL

Dans cette seconde partie, nous nous intéresserons à l'aspect documentaire du modèle conceptuel. Ce dernier comprend la structure conceptuelle statique(description du système d'informations) et la structure conceptuelle dynamique(description des traitements). Chacune d'elles sera traitée respectivement dans les quatrième et cinquième chapitres. Nous commencerons chaque fois cette étude par la présentation d'un exemple. Nous critiquerons ensuite les concepts qui nous sont proposés dans B.B.B.C. pour décrire ces deux structures. Dans les deux cas, nous terminerons par quelques suggestions dont le but est d'améliorer l'aspect documentaire du modèle conceptuel.

Dans le sixième chapitre, nous verrons dans quelle mesure il est possible d'insérer le schéma conceptuel au sein d'une méthode d'analyse.

CHAPITRE IV. STRUCTURE CONCEPTUELLE

Rappelons tout d'abord que la structure conceptuelle comprend deux parties: la description de l'ensemble des données(base de données conceptuelle) et celle de leurs propriétés(contraintes d'intégrité). Le processus qui nous conduit à cette structure conceptuelle comporte deux phases: la perception des éléments de base au niveau du monde réel(objets, associations, propriétés) et la représentation de ceux-ci dans un schéma conceptuel(données, entités, relations, contraintes d'intégrité).

Nous illustrerons ce processus en nous basant sur un exemple restreint de la banque PARIBAS qui nous a accueillis lors de notre stage. Nous procéderons ensuite à une critique des concepts développés dans B.B.B.C. tant au niveau de la perception qu'au niveau de la représentation. Nous terminerons ce chapitre en émettant quelques propositions dont le but est d'améliorer l'aspect documentaire du schéma conceptuel.

IV.1. Présentation de la structure conceptuelle.

Notre intention n'était pas de décrire de façon complète les informations relatives à un organisme bancaire. Nous avons relevé les éléments que nous avons jugés essentiels. De plus, nous y avons fait figurer les données nécessaires à la description des traitements relatifs au calcul des intérêts sur les comptes à vue. Les détails concernant ces illustrations sont donnés en annexes I,II,III.

IV.1.1. Perception du réel.

Nous faisons ci-après le relevé des types d'objets et d'associations en mentionnant pour chacun leurs caractéristiques essentielles.

IV.1.1.1. Les objets.

a) Agence.

Il est important pour le siège central de PARIBAS à Bruxelles d'avoir des renseignements sur ses diverses agences réparties dans le pays. Chaque agence se voit attribuer un numéro qui lui est propre. Elle possède une ou deux adresses(suivant la langue de rédaction).

b) Service.

La description des services d'une agence peut être intéressante dans la perspective d'applications de gestion futures(études de rentabilité de chacun des services d'une agence).

c) Client.

La notion de client est une des sous-structures de base de la banque. Chaque client est identifié par un numéro de matricule qui lui est propre. Dans la description de ses caractéristiques, nous trouvons entre autres: la nationalité, la profession, son état civil, sa date de naissance et divers renseignements utiles pour la banque concernant par exemple l'honorabilité du client.

d) Adresse du client.

Pourquoi avoir distingué l'objet "adresse-client" de l'objet "client"? Cette distinction provient du fait que la correspondance joue un rôle important dans les rapports entre le client et le banquier. Le client peut posséder plusieurs adresses(en Belgique et éventuellement à l'étranger).

e) Produit.

Nous avons considéré la banque comme une entreprise ordinaire qui "fabrique" et "vend" des produits. Nous distinguons les produits suivant leur type. Chaque produit d'un type donné peut être identifié par un numéro et un libellé. Le "prix de vente" est également une propriété attachée à chacun de ceux-ci.

f) Devise.

Chaque devise est identifiée par un code numérique. Pour chacune d'entre elles, nous aurons les caractéristiques concernant leurs cours(vendeur,acheteur). Nous ferons également mention des conditions standard pour le calcul des intérêts sur les comptes tenus en monnaies étrangères.

g) Code série de compte.

Ce code est en fait une caractéristique du client : un code coïncide à un type de client bien particulier(employé, agriculteur,...). A chacun de ces types est associé un intervalle qui correspond aux numéros de matricule attribuables à cette classe de clients. A un code série de compte sont associées les conditions standard pour le calcul des intérêts sur les comptes tenus en francs belges.

h) Taux de base.

Le taux d'intérêt est composé d'un taux de base augmenté d'une marge,d'un taux spécial ou d'une commission suivant le type de compte, le type de client, la devise du compte,...

IV.1.1.2. Les associations.

a) Compte(client-produit-agence).

Un compte est relatif à une certaine catégorie de produits, est détenu par un client au sein d'une agence qui en assure la tenue. Un compte peut jouir de conditions particulières

pour le calcul des intérêts(créditeurs ou débiteurs).

b) Crédit(client-produit-agence).
Chaque crédit accordé à un client dans une agence se voit attribuer un numéro. On leur associe les garanties émises par le client et, selon le cas, celles émises par la banque pour un crédit de garantie accordé à un client.

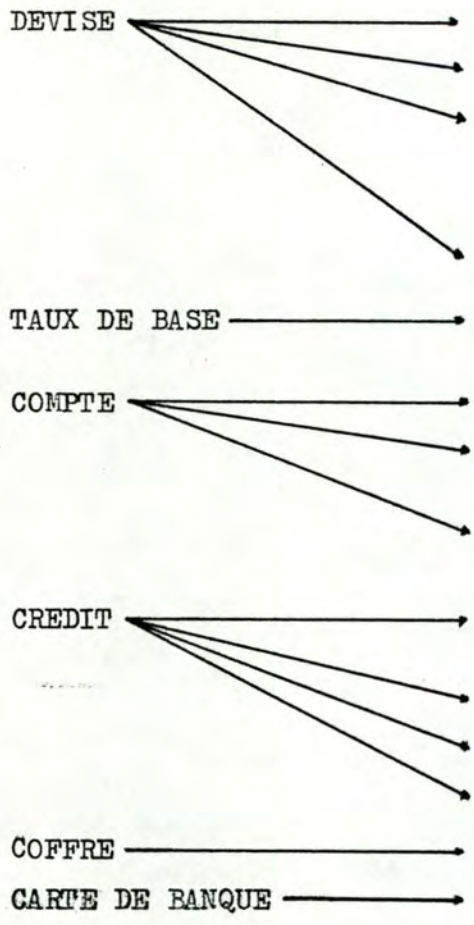
c) Coffre(client-produit-agence).
Un coffre est loué par un client dans une agence; on indiquera pour ce coffre la durée de location.

d) Carte de banque(client-produit-agence).
Une agence délivre à ses clients différents types de cartes de banque.

IV.1.2. Représentation formelle.

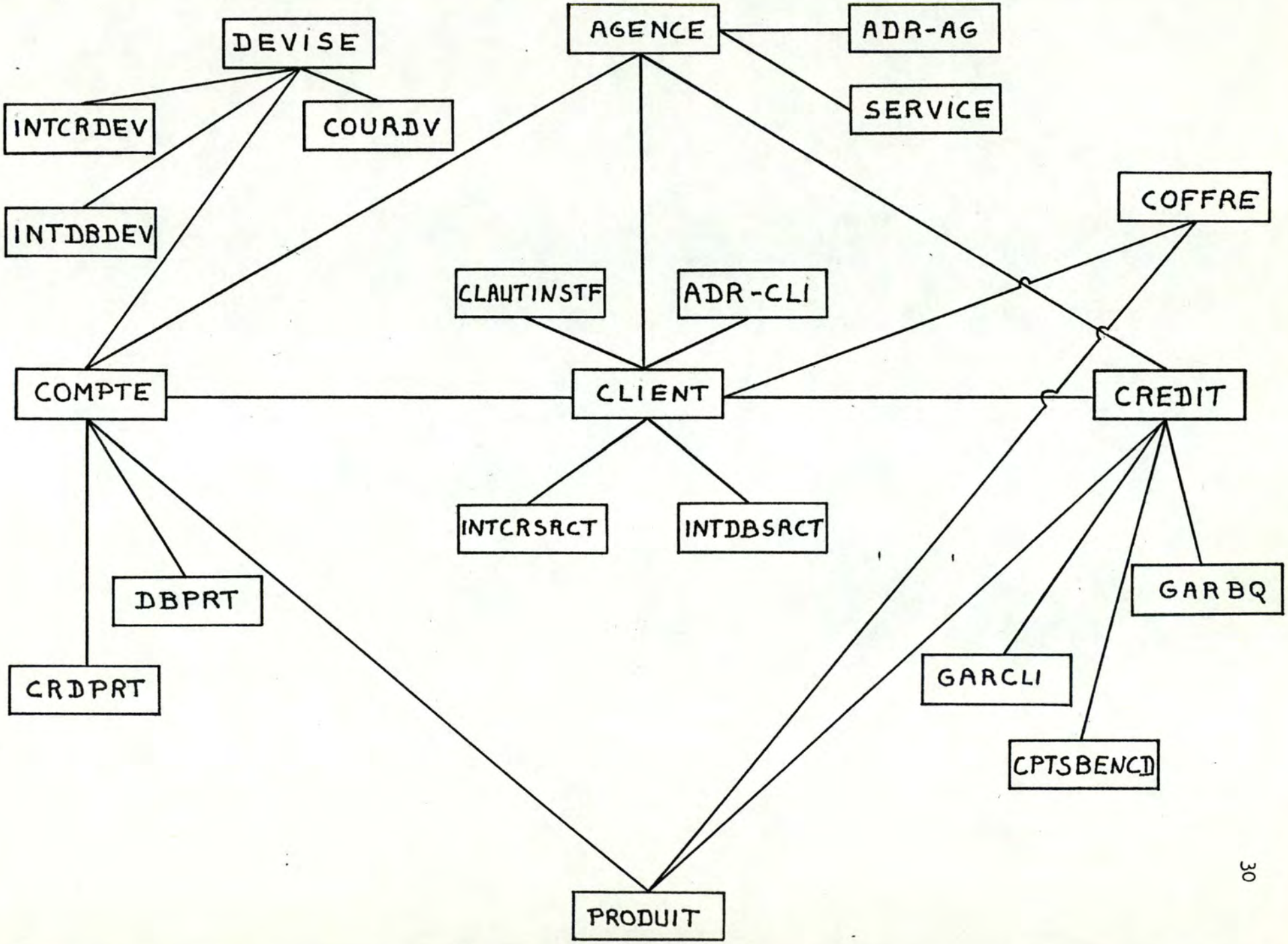
Nous analyserons dans ce paragraphe le passage de la représentation sous forme d'objets et d'associations à la représentation sous forme de relations. Nous nous contenterons de déterminer les différents types de relations: le détail de leurs propriétés se trouve en annexe II.

OBJETS - ASSOCIATIONS.		RELATIONS.	
AGENCE	→	AGENCE	: caractéristiques de l'agence.
	→	ADR-AG	: adresse(s) de l'agence.
SERVICE	→	SERVICE	: services de l'agence.
CLIENT	→	CLIENT	: caractéristiques générales du client.
	→	CLAUTINSTF	: numéro(s) de compte du client dans d'aures institutions financières.
ADRESSE-CLIENT	→	ADR-CLI	: adresse(s) du client.
PRODUIT	→	PRODUIT	: ensemble des produits.
CODE SERIE DE COMPTE	→	INTCRSRCT	: par série de compte on aura les conditions standards pour le calcul des intérêts créditeurs.
	→	INTDBSRCT	: idem pour intérêts débiteurs.



DEVISE : description des devises.
COURDV : cours des devises.
INTCRDEV: conditions standards pour le calcul des intérêts créditeurs sur les comptes tenus en monnaies étrangères.
INTDBDEV: idem pour les intérêts débiteurs.
TAUXBASE: taux de base avec date de validité.
COMPTE : description générale du compte.
CRDPRT : conditions particulières pour le calcul des intérêts créditeurs.
DBPRT : conditions particulières pour le calcul des intérêts débiteurs.
CREDIT : caractéristiques du crédit accordé à un client.
GARCLI : garanties données par le client.
GARBQ : garanties émises par la banque.
CPTSBENCD:comptes bénéficiant du crédit.
COFFRE : location des coffres.
CRTBQ : carte de banque.

Nous dressons à la page 30 le graphe des types de relations que nous venons de décrire succintement.


DEVISE
AGENCE
ADR-AG
SERVICE
INTCRDEV
COURDV
INTDBDEV
COFFRE
CLAUTINSTF
ADR-CLI
COMPTE
CLIENT
CREDIT
INTCRSRCT
INTDBSRCT
DBPRT
GARBQ
CRDPRT
GARCLI
CPTSBENCD
PRODUIT

IV.2. Critique des concepts.

IV.2.1. Perception du réel.

Il n'est pas dans notre intention de critiquer les définitions qui nous sont données des objets et des associations. Nous sommes convaincus de la nécessité de tels concepts pour une première approche de structuration du système d'informations. La distinction entre objet et association nous paraît naturelle et permet, au niveau du monde réel, la détermination des relations existant entre différents ensembles d'informations. Néanmoins certaines ambiguïtés peuvent apparaître dans la dénomination de ces ensembles: objet ou association? Prenons un exemple:

> Le gestionnaire d'une banque distinguera les objets CLIENT, AGENCE, PRODUIT et considèrera l'association COMPTE entre ces trois objets. L'utilisateur s'occupant du calcul des intérêts sur les comptes à vue considèrera sans doute l'objet COMPTE.

Il est évident que la perception du monde réel ne peut s'opérer en une seule étape: premièrement, elle résulte du recensement des problèmes qui sont du ressort des différents utilisateurs et deuxièmement, de l'intégration de ces points de vue en une description unique.

IV.2.2. Représentation formelle(relations).

Si un objet entre dans la composition d'une association, c'est pour y jouer un rôle bien précis: cela résulte de l'existence même d'une association qui est contingente à celle des objets qui la composent. Dans B.B.B.C., dès que nous avons déterminé les objets et les associations dans le réel perçu et que nous passons à la description de ceux-ci dans l'organisation conceptuelle, nous ne parlons plus qu'en termes de relations. Que penser de cette approche par rapport à celles qui nous sont proposées par le modèle individuel (1) et le modèle entité association? (2) Rappelons que ces modèles reprennent la distinction entre objet et association au niveau de la représentation formelle des informations. Cette divergence de vue provient sans doute du rôle que l'on assigne au schéma conceptuel. Bien sûr, il est une représentation conforme de l'organisation

(1) cfr. chapitre II.2.2.1.
(2) cfr. chapitre II.2.2.2.

décrite et par conséquent, il doit contenir les éléments permettant de retrouver les caractéristiques décelées au niveau de la perception du monde réel.

Exemple: les liens existant entre les relations représentatives des objets et des associations seront décrits au moyen de projections sur les domaines de ces relations.

Un des buts du schéma conceptuel est de mettre à la disposition des utilisateurs des ensembles d'informations manipulables, indépendamment de leurs origines: la relation CLIENT est représentée de la même manière que la relation COMPTE, en faisant abstraction du fait que l'objet CLIENT entre dans la composition de l'association COMPTE. Dans cet ordre d'idées, il nous semble que le concept de relation s'adapte bien à la manipulation des données: facilité de représentation, facilité de création de nouvelles relations à partir de celles existantes...

IV.3. L'aspect documentaire du schéma conceptuel.

IV.3.1. Synthèse des documents.

Dans B.B.B.C., un canevas général pour décrire l'ensemble des informations et leurs propriétés nous est proposé. Par rapport aux documents mis à notre disposition à PARIBAS, cela présente un avantage certain. En effet, pour repérer les informations, le plus facile est de partir du schéma IDS qui permet une localisation plus aisée.(schémaIDS= représentation de la structure d'accès). Les données contenues dans l'article choisi se trouvent dans le document descriptif des articles (format des données). Le dossier des désignations symboliques reprend ces mêmes caractéristiques. Pour les contrôles à effectuer sur les composantes d'un article, nous pouvons consulter le dossier des contrôles qui contient à nouveau des éléments des dossiers précités.

Il n'est nullement dans notre intention de critiquer la façon dont sont présentés les documents à la banque PARIBAS. Tout en reconnaissant la lisibilité et la bonne présentation des dossiers, nous pouvons néanmoins regretter une certaine dispersion des renseignements relatifs à un même élément.

A notre avis, le modèle B.B.B.C. remédie à cet inconvénient et, de plus, évite la redondance des descriptions.

La suite de ce chapitre sera consacrée à l'exposé de quelques propositions dont le but est d'améliorer l'aspect documentaire du schéma conceptuel.

IV.3.2. Graphe des objets et des associations.

Le modèle B.B.B.C. ne propose pas de schéma pour représenter le contenu de l'organisation conceptuelle. Nous avons mentionné ci-dessus l'intérêt du schéma IDS à PARIBAS pour la recherche des informations(1) Nous retrouvons le même problème au niveau conceptuel: il nous semble important de visualiser les objets et les associations que l'on distingue. Monsieur Bodart, dans son cours de MIS (2), propose un symbolisme permettant de faire la distinction entre les objets et les associations(symboles différents pour les représenter). Il est possible également de procéder à une quantification des liens entre objets et associations et de les indiquer sur le graphe: relation 1-n, n-n,n-1.
Nous aurons par exemple:

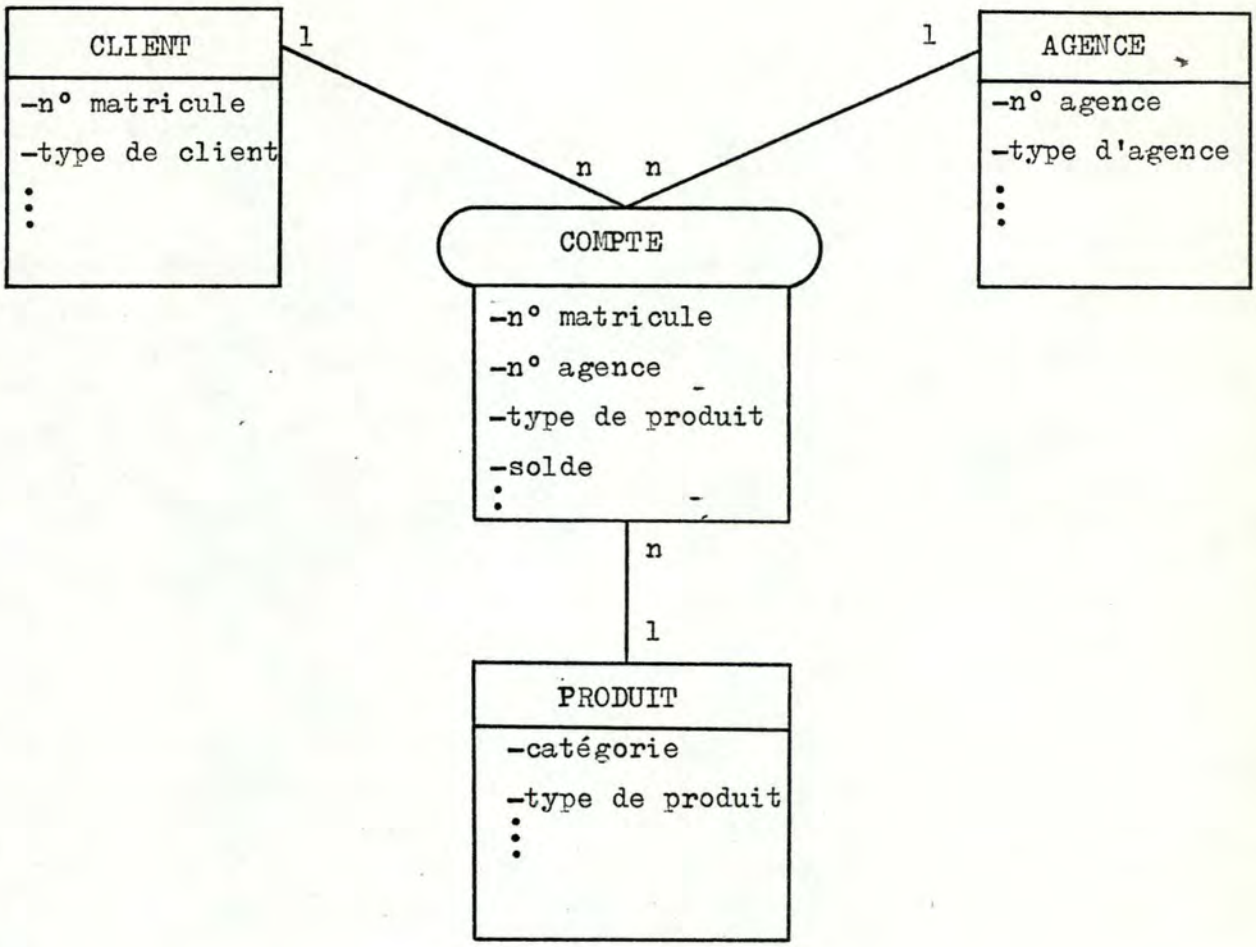
a) Objets: CLIENT, AGENCE, PRODUIT.
   Association: COMPTE.

b) Quantification des liens:
   -un client peut posséder plusieurs comptes;
   -un compte appartient à un seul client;
   -une agence détient plusieurs comptes;
   -un compte est tenu dans une seule agence;
   -un compte se réfère à un type de produit;
   -plusieurs comptes correspondent à un type de produit.

---

(1) cfr. chapitre IV.3.1.

(2) cfr. ouvrage repris dans la bibliographie sous le numéro 4.



IV.3.3. Relation sous 3ème forme normale.(1)

La représentation d'un ensemble de relations sous 3ème forme normale est particulièrement commode lorsqu'on considère un modèle relationnel comme modèle conceptuel. Les relations sous 3ème forme normale décrivent indépendamment les unes des autres les relations de base d'un ensemble d'informations. Nous allons définir ce concept et en montrer les avantages en l'appliquant à un exemple.

Soient les domaines NOMATR: numéro de matricule
NOAGENCE: numéro d'agence
TYPCPTE: type de compte
SIECPTE: code série de compte
SOLDE: solde du compte
CDNUDEV: code numérique de la devise
CDALDEV: code alphabétique de la devise

et la relation COMPTE(NOMATR, NOAGENCE, TYPCPTE, SIECPTE, SOLDE, CDNUDEV, CDALDEV)

(1) repris de l'ouvrage 2 de la bibliographie.

La clé de cette relation est (NOMATR, NOAGENCE, TYPCPTE) qui correspond au numéro de compte. Nous avons également les relations fonctionnelles(1) suivantes:

NOMATR ⟶ SIECPTE
CDNUDEV ⟶ CDALDEV

Soit un exemple de réalisation de COMPTE:

$$\text{COMPTE} = \left\{ \begin{array}{l} \text{nm1, nag1, typ1, sc1, sld1, cnud1, cald1} \\ \text{nm1, nag1, typ2, sc1, sld2, cnud2, cald2} \\ \text{nm2, nag3, typ3, sc3, sld3, cnud1, cald1} \\ \text{nm2, nag2, typ2, sc3, sld4, cnud2, cald3} \\ \text{nm3, nag1, typ4, sc2, sld5, cnud3, cald2} \end{array} \right\}$$

Supposons que nous changions de code série de compte du client de matricule nm1. Nous constatons alors qu'il est nécessaire de modifier 2n-uples de COMPTE. Il faut donc donner une représentation de COMPTE qui élimine ce problème.

> Définition: 2ème forme normale de Codd.
>
> Soient R(A, B, C,...)
> et $I_1, I_2, \ldots, I_j, \ldots, I_n$ les clés de R.
>
> Alors, si pour tout domaine J n'appartenant à aucune clé et pour tout j, j=1,...,n il n'existe pas $I'_j \subset I_j$ tel que J soit fonctionnellement dépendant de I'_j, alors la relation R est dite sous 2ème forme normale de Codd.

La relation COMPTE n'est pas sous 2ème forme normale vu que NOMATR⟶SIECPTE; COMPTE est dite sous 1ère forme normale.

Décomposons la relation COMPTE en 2 relations

$$\text{COMPTE-1(NOMATR, SIECPTE)} = \left\{ \begin{array}{l} \text{nm1, sc1} \\ \text{nm2, sc3} \\ \text{nm3, sc2} \end{array} \right\}$$

COMPTE-2(NOMATR, NOAGENCE, TYPCPTE, SOLDE, CDNUDEV, CDALDEV)

$$= \left\{ \begin{array}{l} \text{nm1, nag1, typ1, sld1, cnud1, cald1} \\ \text{nm1, nag1, typ2, sld2, cnud2, cald3} \\ \text{nm2, nag3, typ3, sld3, cnud1, cald1} \\ \text{nm2, nag2, typ3, sld4, cnud2, cald3} \\ \text{nm3, nag1, typ4, sld5, cnud3, cald2} \end{array} \right\}$$

(1) cfr. chapitre VII.3.2.1.

La relation COMPTE est la composition de ces deux relations sur le domaine NOMATR. La modification du code série de compte du matricule nml se résume à la modification d'un seul n-uple de la relation COMPTE-1.

Supposons que l'on change le code alphabétique de la devise caldl. Cela implique la modification de 2 n-uples de la relation COMPTE-2.

> Définition : Soient R(A, B, C, D)
> et $B \rightarrow C$ $(C \not\rightarrow B)$
> $C \rightarrow D$ $(D \not\rightarrow C)$
> On dit que D est transitivement dépendant de B.
>
> Définition: $3^{ème}$ forme normale de Codd.
> Soient $S(A_1,\ldots,A_n)$ sous $2^{ème}$ forme normale.
> Si aucun domaine n'est transitivement dépendant d'autres domaines, alors S est sous $3^{ème}$ forme normale.

Etant donné un compte, il n'est exprimé que dans une seule devise et, à un code numérique d'une devise ne correspond qu'un seul code alphabétique. Décomposons la relation COMPTE-2 en deux relations COMPTE-3 et COMPTE-4:

COMPTE-3(NOMATR, NOAGENCE, TYPCPTE, SOLDE, CDNUDEV)

$$= \begin{Bmatrix} nml, & nag1, & typ1, & sc1, & sld1, & cnud1 \\ nml, & nag1, & typ2, & sc1, & sld2, & cnud2 \\ nm2, & nag3, & typ3, & sc3, & sld3, & cnud1 \\ nm2, & nag2, & typ2, & sc3, & sld4, & cnud2 \\ nm3, & nag1, & typ4, & sc2, & sld5, & cnud3 \end{Bmatrix}$$

$$\text{COMPTE-4(CDNUDEV, CDALDEV)} = \begin{Bmatrix} cnud1, & cald1 \\ cnud2, & cald3 \\ cnud3, & cald2 \end{Bmatrix}$$

Nous aurons donc une seule modification de la relation COMPTE-4. La relation COMPTE peut être représentée par trois relations: COMPTE-1, COMPTE-3, COMPTE-4. Comme nous le montre l'exemple, le concept de $3^{ème}$ forme normale évite de multiples mises à jour, et est d'un intérêt certain pour la manipulation des relations (éviter d'avoir des relations trop longues).

IV.3.4. Qualifier plus les données.

Sans pour autant alourdir la description des données et y faire mention de caractéristiques superflues, il nous semble néanmoins utile d'apporter quelques ajouts. Le schéma conceptuel doit constituer le cahier des charges pour l'implémentation: c'est au niveau de la structure d'accès que nous effectuerons le choix des informations effectivement contenues dans la base de données.Une classe de données qu'il est intéressant de distinguer est constituée des "données calculées": leurs valeurs peuvent être obtenues en effectuant certaines opérations sur d'autres données. L'implémenteur aura le choix entre leur mémorisation ou leur calcul à chaque utilisation. Monsieur Dambrine dans "L'organisation des données" (1), apporte cette distinction en rangeant les informations en données "primaires" et "secondaires". Dans son article "Sémantique et structure des données" (2), Flory mentionne une autre caractéristique, la stabilité : une donnée est susceptible de mise à jour ou non. Ces quelques considérations nous conduisent à dresser le tableau suivant:

Donnée calculée.	Stable.	Montant des intérêts calculés à une date précise.
	Non stable.	Solde du compte.
Donnée non calculée.	Stable.	Date de naissance.
	Non stable.	Nombre d'enfants, adresse.

Il est à noter que nous retrouvons indirectement la notion de "données calculées" dans les concepts proposés dans B.B.B.C.: les règles de calcul nous sont fournies sous la forme de contraintes d'intégrité insérées dans la description des relations. Une règle de calcul concernant une donnée d'une relation fait appel à des éléments d'une ou de plusieurs relations. Cette règle est une propriété de toutes les occurrences de la relation, mais il nous semble que c'est avant tout une propriété de la donnée. Pour jouer son rôle de documentation, l'organisation concep-

(1) cfr. ouvrage repris dans la bibliographie sous le numéro 7.
(2) cfr. ouvrage repris dans la bibliographie sous le numéro 5.

tuelle a intérêt à faire figurer explicitement ces caractéristiques: elles ont pour but d'introduire, là où il le faut les règles de calcul qui détermineront correctement les mises à jour.

IV.3.5. Lisibilité.

Si au début de ce chapitre(1), nous avions émis certaines critiques au sujet de la disponibilité des informations à partir des dossiers présentés à PARIBAS, il serait faux de prétendre que ce qui nous est proposé dans B.B.B.C. constitue ce que l'on pourrait faire de mieux comme outil de documentation. Ilest vrai que l'exemple développé dans cet article est purement indicatif; l'objet des paragraphes qui suivent est d'émettre quelques suggestions en vue d'augmenter la lisibilité de la description.

IV.3.5.1. Lisibilité des données.

Un des rôles assigné à une base de données est de résoudre le problème de la centralisation de la gestion des données. Devant le nombre croissant des applications, des utilisateurs, ilest apparu nécessaire de constituer une équipe responsable de l'administration de l'ensemble des données.(1)

"L'administrateur de la base est la personne ou le groupe assigné à la responsabilité, la gestion, la définition, l'organisation, la génération, la protection et à l'efficacité de la base de données".(2)

" En utilisant un langage de description des données, il définit les attributs logiques et physiques des données, leurs associations, les différents mappings entre la D.B. logique et la D.B. physique, les méthodes d'accès, les contraintes d'intégrité.....".(3)

Ilest évident que cet administrateur de la base de données doit disposer d'un instrument lui permettant de synthétiser les caractéristiques recueillies au sujet des données. Il constituera un dictionnaire

(1) cfr. chapitre IV.3.1.
(2) cfr. ouvrage repris dans la bibliographie sous le numéro 2.
(3) cfr. ouvrage repris dans la bibliographie sous le numéro 9.

où il pourra reprendre toutes les informations qu'il juge utiles en vue de la gestion des données: -nom

-désignations
-codage externe
-contraintes d'intégrité
-endroits où elle est utilisée
-qui peut la manipuler
-peut-on la mettre à jour
-la fréquence d'utilisation
⋮

Si la description des données ne doit pas nécessairement contenir à ce stade de l'analyse autant de renseignements qu'un dictionnaire, elle offrirait néanmoins des avantages sérieux si elle pouvait en avoir les propriétés. Parmi celles-ci, il en est trois qui nous paraissent utiles: 1) Le dictionnaire est un inventaire des seules données auxquelles on peut faire appel.

2) Il permet d'éviter la redondance, les synonymes.

3) C'est un outil de documentation rapide et facilement manipulable.

Au sein de cette description, une règle comme le rangement par ordre alpbabétique serait une solution simple à réaliser et très bénéfique (encore faudrait-il que la première lettre, ou peut-être même les deux premières, de la désignation symbolique soit identique à celle du nom usuel de la donnée). Cela permettrait de repérer directement la donnée qui nous intéresse, d'éviter l'attribution de deux désignations identiques à des données sémantiquement différentes.

L'introduction dans la description d'une donnée de la liste des relations auxquelles elle fait partie est un élément qui favoriserait la recherche de renseignements. Par exemple, si nous désirons trouver la règle de calcul correspondant à une donnée calculée, il suffit de la rechercher parmi les contraintes d'intégrité relatives à la(aux) relation(s) contenant cette donnée secondaire.

IV.3.5.2. Lisibilité des relations.

Comme nous venons de le voir, il est possible que nous soyons amenés dans certains cas à faire une recherche parmi les relations. Ceci implique, comme au chapitre précédent, qu'une classification suivant un critère bien précis soit proposée en vue d'éliminer une exploration trop fastidieuse si le nombre de relations est élevé. La désignation symbolique de la relation ne devrait pas être trop différente du nom usuel de l'objet ou de l'association qu'elle représente. Un classement par ordre alphabétique éviterait une perte de temps, permettrait une consultation rapide et une vérification aisée à propos de l'unicité de l'emploi.

IV.3.5.3. Lisibilité des contraintes d'intégrité.

La logique des prédicats présente l'avantage de décrire de manière non ambiguë tous les types de contraintes d'intégrité que l'on puisse imaginer. Cependant, nous pouvons lui faire le reproche de faire preuve parfois d'une certaine lourdeur, d'une certaine longueur. Il est à noter que cet aspect négatif se remarque plus lors de la lecture que lors de l'élaboration de ces contraintes.

L'emploi "d'opérateurs" comme CARD-PROJEC,INCLUDED,EQUAL(1), permet de remédier à cet inconvénient.

Dans l'exemple décrit en annexe, nous employons une notation qui nous a permis de clarifier, d'alléger l'expression de certaines propriétés (Ex. la détermination de la(des) valeur(s) d'un domaine A d'une relation R, conditionnellement à la(les) valeur(s) prise(s) par un domaine B de cette même relation R ou d'une autre relation R). Décrivons un exemple:

> Soit la relation COURDV donnant les cours des devises.
> Soient 2 domaines de COURDV -CDNUDEV: code numérique de la devise.
> -MARCHE: code marché.
> Si le code de la devise est 000 ou 150 alors le code marché est F,C,B,S,E,P ou D; s'il est différent de 000 et 150 alors le code marché est L,R ou P.

---

(1) voir annexe II, description des relations.

Normalement, nous devrions écrire: (l=occurrence de COURDV)

$\forall l(((l \in COURDV) \wedge ((CDNUDEV.l='000') \vee (CDNUDEV.l='150'))) \Rightarrow$
$((MARCHE.l='F') \vee (MARCHE.l='C') \vee \ldots \vee (MARCHE.l='\not{b}')))$

Nous aurions une expression semblable pour la 2ème propriété. Pour alléger cette expression, nous déclarons l'existence d'ensembles de valeurs que nous utilisons à l'intérieur des prédicats.

Soient DVNUM={000,150}
MRCH1={F,C,B,S,E,P,$\not{b}$}
MRCH2={L,R,P}

Et nous aurons:

$\forall l(((l \in COURDV) \wedge (CDNUDEV.l \in DVNUM)) \Rightarrow (MARCHE.l \in MRCH1))$
$\forall l(((l \in COURDV) \wedge (CDNUDEV.l \notin DVNUM)) \Rightarrow (MARCHE.l \in MRCH2))$

CHAPITRE V. LES REGLES D'EVOLUTION.

Les règles d'évolution ont pour but de décrire la dynamique du système d'informations. Théoriquement, elles coïncident aux opérations élémentaires sur la base: création, suppression, modification. Comme nous le verrons plus tard (1), elles sont la source des programmes d'application s'exécutant sur la base de données.

Nous décrirons ci-après un exemple de règle d'évolution. Nous critiquerons les concepts qui nous sont proposés pour cette description et ensuite, nous soumettrons quelques propositions visant à la compléter et à faciliter son exploitation.

V.1. Exemple de règles d'évolution.

Nous avons choisi de décrire un exemple très simple de règle d'évolution, à savoir la création d'une nouvelle agence. Elle se résume en l'adjonction d'un nouvel n-uple à la relation AGENCE.

AGENCE(NOAGENCE, TPAGENCE, LGEDTC, NOTLPHN, NOTELEX, MBCOMP, NOCOMP, SUCGEST, SUCEXPL)

où NOAGENCE: numéro de l'agence
TPAGENCE: type de l'agence
LGEDTC: langue pour l'édition des tableaux comptables
NOTLPHN: numéro de téléphone
NOTELEX: numéro de télex
MBCOMP: membre ou pas de la chambre de compensation
NOCOMP: numéro de la chambre de compensation
SUCGEST: succursale dont dépend l'agence décrite
SUCEXPL: succursale exécutant certains travaux pour l'agence décrite.

Les propriétés de cette relation et de ses composantes sont données en détails en annexe II. Elles nous serviront lors de la détermination du contexte de cohérence de l'application.

(1) cfr. chapitre VII.2.2.

Ci-dessous, vous trouvez la description de la règle d'évolution (1): elle est déclenchée par un événement EVENT-1 caractérisé par une liste de paramètres PAR-1 qui nous renseignent sur les valeurs des données à ajouter. Nous expliquerons ensuite le sens des différentes expressions rencontrées.

```
PROC   CR-AG   'création agence'
 TRIGGER   EVENT-1
 PARAMETERS
  EVENT-1  PAR-1  AG(NOAGENCE, TPAGENCE, LGEDTC, NOTLPHN, NOTELEX,
                     MBCOMP, NOCOMP, SUCGEST, SUCEXPL)
 EXT-INT-CONST-CONTEXT
  EIC-1:∀ag((ag∈AG)⇒((NOAGENCE.ag≠ '5XX')∧(SUCGEST.ag≠'5XX')
        ∧(SUCEXPL.ag≠'5XX')))
  EIC-2: ∀ag((ag∈AG)⇒((NOAGENCE.ag≠'bbb')∧(TPAGENCE.ag≠'b')
        ∧(NOCOMP.ag≠'bb')))
  FORM-1  FORM-2 FORM-8 FORM-9 FORM-10 FORM-11 FORM-12 FORM-13 FORM-14
  EVENT-3: EIC-1  PAR-1
  EVENT-4: EIC-2  PAR-1
  EVENT-5: (FORM-1 OR FORM-2 OR...OR FORM-14) PAR-1
 INT-INT-CONST-CONTEXT
  IC-3
  EVENT-12: IC-3  PAR-1
 MIC-INT-CONST-CONTEXT
  MIC-1: ∀1((1∈AGENCE)⇒∃ag((ag∈AG)∧(NOAGENCE.1=NOAGENCE.ag)))
  MIC-2: ∀ag(((ag∈AG) ∧ (SUCEXPL.ag≠'bbb'))⇒∃1((1∈AGENCE)
        ∧(NOAGENCE.1=SUCEXPL.ag)))
  MIC-3: ∀ag(((ag∈AG)∧(SUCGEST.ag≠'bbb'))⇒∃1((1∈AGENCE)
        ∧(NOAGENCE.1=SUCGEST.ag)))
  EVENT-13: MIC-1  PAR-1
  EVENT-14: (MIC-2 OR MIC-3)  PAR-1
 PROC-DESCRIPTION
  BEGIN
    AJOUTER AG A AGENCE.
  END
END  CR-AG.
```

(1) Cette règle d'évolution est décrite indépendamment de celles exposées en annexe III.

Commentaires sur la description.(contexte de cobérence)

1) Contraintes d'intégrité externes.

Elles portent uniquement sur les paramètres d'entrée, c'est-à-dire les données qui ne sont pas encore enregistrées dans la base.

EIC-1: NOAGENCE,SUCEXPL et SUCGEST ne peuvent débuter par le chiffre 5.

EIC-2: NOAGENCE,TPAGENCE et NOCOMP sont des composantes obligatoires: leurs valeurs doivent être différentes de la valeur inconnue.

FORM-1,...,FORM-14: contraintes relatives aux formats des paramètres.

EVENT-3: événement déclenchant une procédure d'erreur suite au fait que la contrainte EIC-1 n'est pas vérifiée: création refusée pour donnée(s) incorrecte(s).

EVENT-4: événement déclenchant une procédure d'erreur suite au fait que la contrainte EIC-2 n'est pas vérifiée: création refusée pour donnée(s) insuffisante(s).

EVENT-5: les formats des paramètres ne sont pas tous conformes: dans certains cas, on forcera la donnée à une valeur déterminée et on procédera à la création; dans d'autres cas, on refusera la création pour format(s) incorrect(s).

2) Contraintes d'intégrité internes.

IC-3: NOAGENCE est la clé de la relation AGENCE.

EVENT-12: événement déclenchant une procédure d'erreur signalant le fait que IC-3 n'est pas vérifiée.

3) Contraintes d'intégrité mixtes.

MIC-1: Le NOAGENCE donné comme paramètre ne peut déjà figurer dans une occurrence de la relation AGENCE.

MIC-2 et MIC-3: si SUCEXPL et SUCGEST sont significatifs dans la liste des paramètres, il existe des occurrences de la relation AGENCE avec ces numéros comme NOAGENCE.

EVENT-13: événement déclenchant une procédure d'erreur: création refusée, NOAGENCE existe déjà.

EVENT-14: événement déclenchant une procédure d'erreur: création refusée, agence absente.

V.2. Critique des concepts proposés.

V.2.1. Règles d'évolution.

Une base de données est en perpétuelle évolution. Elle est conçue pour être mise à la disposition de nombreux utilisateurs qui la consultent, en modifient les données, en créent de nouvelles. Les règles d'évolution ont pour but de décrire les différentes opérations élémentaires auxquelles est sujette la base: créations, modifications, suppressions. La description du schéma conceptuel eut été incomplète si l'on ne s'était pas penché sur la façon dont on pouvait exploiter les données sélectionnées.

L'évolution d'une base de données étant liée aux opérations qu'on lui fait subir, il semble normal de décrire les traitements en faisant apparaître la manière dont s'exécutentles opérations élémentaires(dans l'exemple traité au paragraphe précédent: création d'un nouvel n-uple de la relation AGENCE).

Cet objectif n'a pas été complètement atteint dans la présentation des procédures de l'annexe III: cela est dû en partie au langage de description utilisé et à la démarche algorithmique suivie pour décrire les règles d'évolution. Résumons brièvement l'objet de ces traitements:

> Il s'agit de l'arrêté d'un compte à une date déterminée. Cet arrêté s'accompagne d'un calcul et d'une comptabilisation d'intérêts créditeurs ou débiteurs. Il faut calculer les soldes correspondant aux dates des valeurs(dates à partir desquelles on comptabilise les intérêts) classées par ordre croissant et effectuer les calculs d'intérêts. Après avoir repéré les mouvements sur les comptes durant la période considérée, il suffit de classer les soldes par dates de valeur, de calculer les intérêts et de sortir les résultats.

Examinons les règles d'évolution décrites en annexe et montrons comment elles contiennent la notion d'opération élémentaire sur la base.

a) Enregistrement des mouvements: adjonction de nouveaux n-uples à une relationMOUVEMENTS enregistrant les différents mouvements sur les comptes.

b) Calculs des soldes successifs du compte: adjonction de n-uples respectivement aux relations SLDINTDB ou SLDINTCR suivant que les soldes correpondant aux dates de valeur sont négatifs ou positifs.

c) Calculs des intérêts pour chaque solde: mise à jour des n-uples des relations SLDINTDB et SLDINTCR par garnissage de la zone contenant le montant des intérêts pour les différents soldes.

d) Résultats: mise à jour de l'historique du compte du point de vue du calcul des intérêts(mise à jour de la relation INTERET = faire mention de la dernière date de calcul des intérêts, du solde et des intérêts cumulés à cette date).

V.2.2. Evénements déclencheurs.

Selon la terminologie employée dans B.B.B.C., un événement est une décision qui déclenche une opération sur la base. Cet événement est caractérisé ou non par une liste de paramètres. Cette façon d'opérer correspond aux situations rencontrées dans une organisation. Cette forme d'expression traduit naturellement les besoins, les demandes des utilisateurs(le non-programmeur): ils s'occupent de la représentation des différentes situations qu'ils rencontrent, sans se soucier des opérations qu'elles entraînent. Donnons quelques exemples:

-création d'une nouvelle agence

EVENT-1 PAR-1 AG(NOAGENCE,...)

-enregistrement des mouvements sur un compte

EVENT-2 PAR-1(NOMATR,TYPCPTE,NOAGENCE,VALEUR,MONTANT)
(NOAGENCE,TYPCPTE : numéro de compte ; VALEUR : date de l'opération)

V.2.3. Contexte de cohérence.

Le problème de l'intégrité et de la sécurité des données est d'une importance capitale dans la mise en place d'une base de données. La validité intrinsèque des données accessibles par les utilisateurs se réalise par l'expression de contraintes d'intégrité (logiques) sur les éléments de la structure conceptuelle. Dans ce paragraphe, nous nous

préoccuperons de la présentation du contexte de cohérence d'une règle d'évolution, c'est-à-dire l'ensemble des contraintes d'intégrité qui doivent être vérifiées pour en assurer une exécution correcte. Nous tenterons de dégager les avantages d'une telle présentation. Nous analyserons deux points de vue: celui de l'utilisateur qui décrit la procédure au niveau conceptuel et celui du programmeur qui reprendra cette description pour la transformer en une application exécutable sur la base de données.

V.2.3.1. Point de vue de l'utilisateur.

Ayant décrit quelques traitements sur la structure conceptuelle, nous nous permettons de faire ces quelques remarques concernant les avantages de cette présentation:

a) Situer le problème:
Lorsque nous décrivons un traitement, nous déterminons les informations dont nous avons besoin. Le fait d'obliger l'utilisateur à exprimer les contraintes d'intégrité sur les paramètres d'entrée et sur les données de la base permet à celui-ci de se rendre compte du cadre dans lequel il va travailler. Le report aux relations utilisées met en évidence les possibilités offertes à l'utilisateur ainsi que les contraintes auxquelles il doit se plier.

b) Etre complet:
La validité étant une des principales préoccupations de l'utilisateur, il est normal de s'y attarder. Le canevas qui lui est proposé permet de mettre de l'ordre dans l'expression des contraintes. L'analyse complète des contraintes d'intégrité définies sur les relations utilisées évite les oublis, soulève des problèmes que l'on n'avait pas soupçonnés.

c) Modularité:
Cette présentation offre tous les avantages de la modularité; nous mentionnerons ci-dessous ceux qui nous sont

apparus les plus marquants durant l'élaboration des exemples.

1) Simplification dans la conception.

La description des cas d'erreur prend une place très importante; l'expression du contexte de cohérence permet de différer ces traitements particuliers par rapport au traitement proprement dit. On a la possibilité de s'intéresser à la logique du problème en premier lieu. L'utilisateur jouit également de l'avantage de pouvoir ordonnancer son travail.

2) Liaisons entre modules.

Les liaisons entre les différentes règles d'évolution sont facilement réalisables par le biais des événements déclencheurs de ces procédures. Pour chaque traitement, il est possible de dresser un schéma qui reflète l'enchaînement des différentes règles d'évolution identifiées au moyen de leurs événements déclencheurs.

3) Standardisation.

Un bon nombre de procédures d'erreur se présentent plusieurs fois.(Exemple: contrôle des formats, des clés de relations; existence d'un compte pour tel client;...). Il serait intéressant de "standardiser" certaines fonctions: dans l'exemple décrit en annexe, la vérification du format du numéro de matricule d'un client(EVENT-3) peut n'être décrite qu'une seule fois, évitant ainsi certaines redondances dans l'écriture de cette procédure.

4) Facilité de mise à jour.

La modularité facilite les opérations de mise à jour des traitements: nous ne sommes pas obligés de modifier toute la procédure, mais uniquement la règle d'évolution sujette à ces modifications.

Exemple: si la façon de calculer les intérêts débiteurs d'un compte à vue change, le traitement décrit subira uniquement des modifications localisées au niveau de

la règle d'évolution INTDEB déclenchée par l'événement EVENT-10 (voir annexe III.). Le reste de l'application demeure identique.

V.2.3.2. Point de vue du programmeur.

Les règles d'évolution sont transformées en programmes d'application exécutables sur la structure d'accès. Dans ce paragraphe, nous dégagerons les avantages que le programmeur peut retirer de la présentation de ces règles d'évolution pour réaliser ce passage.

a) Modularité:

Si pour l'utilisateur, le non-programmeur, la modularité offre des avantages quant à la conception des traitements, elle est également d'une aide très précieuse pour le programmeur. Sans affirmer que l'enchaînement des règles d'évolution coïncidera avec celui des modules programmés, il préfigurera dans une large mesure leur contenu et leur agencement.

Examinons les traitements que nous avons décrits. La création de nouvelles agences peut faire l'objet d'un module bien particulier, de même que l'enregistrement des mouvements sur un compte. Pour le calcul des intérêts, nous avons distingué quatre règles d'évolution(CALSLD, INTDEB, INTCRD, MAJINT). Comme ces quatre opérations forment quand même un tout pour le calcul des intérêts, elles seront peut-être regroupées au sein d'un seul module. Nous pouvons néanmoins remarquer que ces quatre règles d'évolution partitionneront le programme et fourniront au programmeur une sorte d'organigramme.

A la vue de ces règles d'évolution, il aura le choix de les regrouper, de les fusionner en modules suivant leur contenu, leur fréquence d'appel. En effet, une règle d'évolution se présentant de nombreuses fois (contrôle du numéro de matricule) sera sujette à une standardisation pour en éviter la réécriture. Les événements induits auxquels on assigne une liste de paramètres

peuvent être interprétés et traduits comme des appels de modules.

b) Contrôles à effectuer:

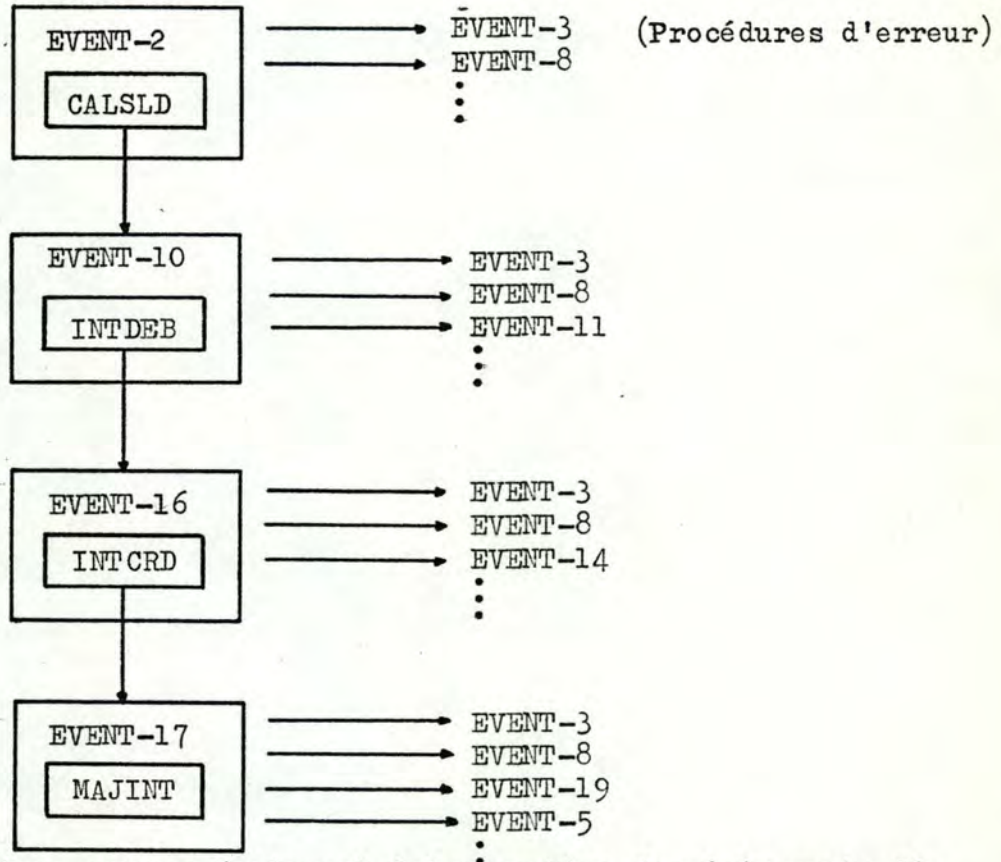
Il est indéniable que les contrôles prennent une grande place dans les programmes. Le fait d'attacher à chaque opération l'ensemble des contraintes devant être réalisées pour former un contexte aussi exact que possible pour sa bonne exécution, facilite le travail du programmeur: il dispose de la liste des contrôles à effectuer, des données sur lesquelles ils portent. Les contraintes d'intégrité externes, internes et mixtes ont une signification particulière pour le programmeur. Lorsqu'il s'intéresse aux premières, il sait qu'il a affaire à des données, des paramètres qui lui viennent de l'extérieur. Il obtiendra ceux-ci par simple lecture. Par contre, les contraintes d'intégrité internes relatives à des données déjà enregistrées dans la base l'obligeront à faire des accès à la base des données en utilisant un langage de navigation. Les contraintes d'intégrité mixtes sont un mélange des deux types. Le programmeur qui veut procéder à des évaluations de coûts trouve là des indications pertinentes et précieuses.

V.3. Suggestions pour faciliter l'exploitation des règles d'évolution.

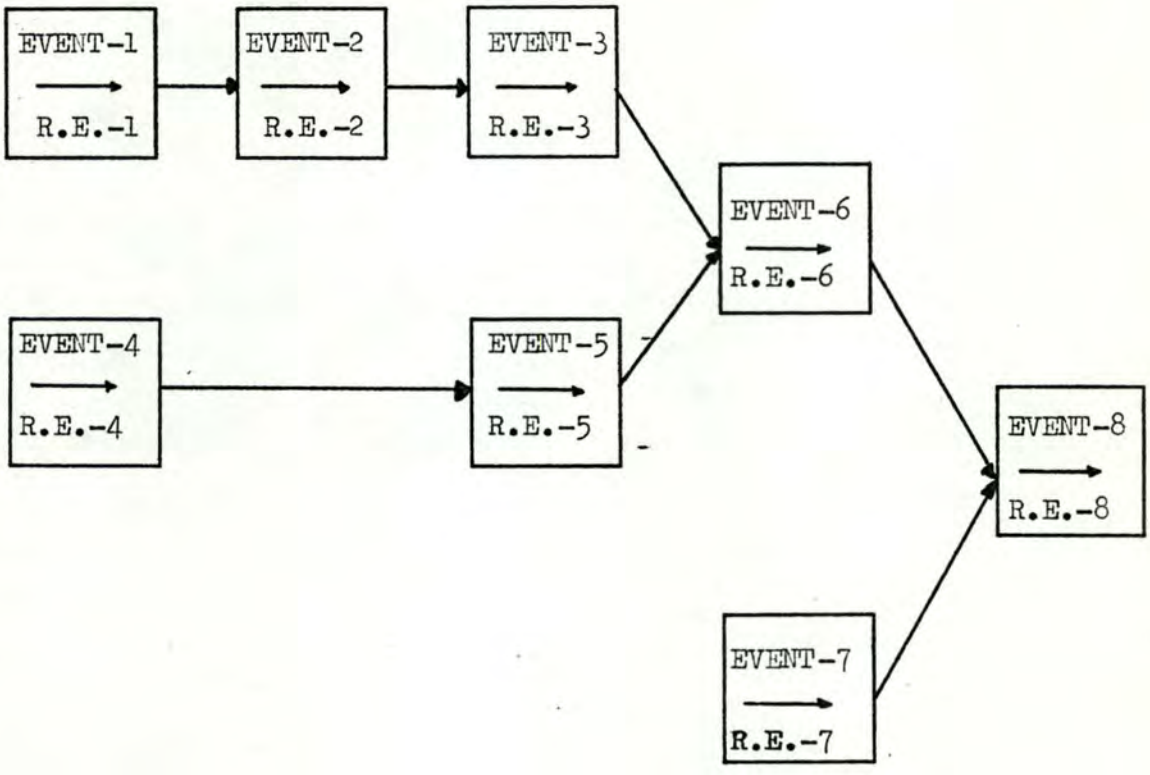
La description des règles d'évolution prendra une place importante dans l'organisation conceptuelle. Les règles d'évolution ont pour but de décrire les différents traitements que les utilisateurs désirent exécuter sur la base. Le schéma conceptuel étant présenté comme un outil de documentation, cet ensemble de descriptions doit être facilement consultable. Nous faisons ci-après quelques suggestions visant à améliorer la consultation des règles d'évolution.

V.3.1. Graphe d'enchaînement des règles d'évolution.

Dans l'exemple décrit en annexe III., le calcul des intérêts se décompose en quatre règles d'évolution: CALSLD,INTDEB,INTCRD et MAJINT. Si nous convenons de les identifier par l'événement qui les déclenche, nous pouvons dresser le graphe suivant(conformément aux notations utilisées en annexe):



Nous pouvons également faire mention des événements déclenchant les procédures d'erreur attachées à chaque règle d'évolution principale. Les événements sont accompagnés éventuellement d'une(de) liste(s) de paramètres. Cet exemple est très simple, il se résume à une séquence de règles d'évolution: l'événement induit par une règle d'évolution déclenche la suivante. Nous pouvons avoir des traitements plus compliqués, justifiant de façon plus évidente la présentation d'un graphe d'enchaînement des règles d'évolution. Ceci nous permettra entre autres, de visualiser le fait qu'une règle d'évolution peut être déclenchée par l'activation de plusieurs événements. Illustrons cela par le schéma suivant:



V.3.2. Les relations utilisées.

Un renseignement utile à introduire dans la description des traitements est la liste des relations auxquelles ils font appel. Elles nous donneront une première indication quant à la localisation spatiale de ces traitements. Lors d'une lecture ou d'une vérification, elles permettent un retour rapide aux informations manipulées, ainsi qu'aux propriétés qui sont prises en compte dans la règle d'évolution.

V.3.3. Inventaire.

On peut également procéder à la tenue d'un inventaire des règles d'évolution. Comme nous l'avons indiqué au point V.2.1., chaque règle serait identifiée par l'événement qui la déclenche et par la liste des paramètres. Cela ne présente peut-être pas beaucoup d'intérêt pour ce qui est des événements déclencheurs des règles d'évolution proprement dites vu que ceux-ci sont habituellement spécifiques à l'application qu'ils activent. Par contre, la situation est différente pour les événements déclencheurs des procédures d'erreur consécutives à la non-vali-

dation de contraintes d'intégrité(contraintes internes, externes et mixtes). Ce sont des contrôles qui s'opèrent dans de multiples opérations, d'où l'intérêt de les cataloguer pour éviter la redondance dans les descriptions.

V.3.4. Facilité de lecture.

La lecture de la description de certains traitements peut parfois être pénible. L'emploi de fonctions comme COMPTER(compter le nombre d'occurrences d'une relation), SOMME(sommer les valeurs de certaines données des différentes occurrences d'une relation), MAX, MIN, MOYENNE,... facilite l'écriture de certaines opérations, la raccourcit et offre une présentation plus agréable. De même, certaines rédactions seraient allégées si on introduisait un ordre sur les occurrences dans la description de la relation.

V.3.5. Renseignements divers.

Lorsqu'on se penche sur les traitements et leur représentation sous forme de règles d'évolution, certains éléments concernant le contexte de l'application nous sont mentionnés et peuvent également figurer dans la description. Nous pensons principalement à la fréquence d'utilisation de la procédure(jour, mois,...), à la personne ou service qui en a la responsabilité, à la personne ou service auquel sont destinés les documents de sortie de l'application,...

CHAPITRE VI. INSERTION DU SCHEMA CONCEPTUEL DANS UNE METHODE D'ANALYSE

Si dans les deux chapitres précédents, nous nous sommes attardés sur les concepts du modèle conceptuel et sur la manière dont ils pouvaient être optimisés en vue de remplir au mieux les objectifs assignés à la structure conceptuelle, nous ne nous sommes pas penchés sur le niveau précédent, à savoir le réel perçu qui constitue le point de départ de tout le processus d'analyse.

Le B.B.B.C. ne propose pas de méthodes d'analyse: les objets, associations et propriétés permettent simplement de décrire, de représenter le réel. Nous nous trouvons donc devant la situation suivante: comment aborder l'analyse d'un système pour en dégager ces trois types d'éléments et faciliter le passage à une représentation sous forme de relations au sein du schéma conceptuel.

Dans ce chapitre, nous rappellerons les deux tendances principales dans les méthodes d'analyse, à savoir les méthodes "top-down" et "bottom-up". Nous les illustrerons respectivement par la méthode de J.D. Warnier et par celle préconisée par Flory; cela nous permettra d'en retirer les avantages et les inconvénients. Après avoir exposé notre démarche lors de l'étude du système bancaire auquel nous nous sommes intéressés, nous tacherons de voir s'il n'existe pas une méthode propre à l'élaboration des bases de données.

VI.1. Méthodes d'analyse.

VI.1.1. Méthodes "bottom-up".

VI.1.1.1. Définition.

"Les méthodes du type "bottom-up" ont pour point de départ l'analyse de l'existant. Après avoir procédé à une description détaillée de ce dernier, on en entreprend une critique en fonction des objectifs mentionnés par la direction. Sur la base de ces critiques, on propose différentes solutions d'automatisation et de réorganisation." (1)

VI.1.1.2. Critiques.

Trois critiques principales sont adressées à ces méthodes "bottom-up" :(1)

- -le risque de donner trop d'importance à l'existant dans la définition des solutions nouvelles;
- -l'analyse de l'existant risque de s'étendre sur une période telle qu'au terme de celle-ci, on devrait déjà procéder à certaines mises à jour;
- -le risque de prendre en charge trop de détails lors de l'analyse de l'existant, ceux-ci pouvant masquer les options fondamentales.

VI.1.1.3. A. Flory:"Algorithmes d'aide à la conception de système d'informations." (2)

Dans ce paragraphe, nous nous pencherons sur une méthode de type "bottom-up" proposée par Flory: il nous présente un processus de détermination des ensembles de base(objets) et de leurs associations. Nous en ferons une critique quant à son apport à la conception et à l'élaboration des bases de données.

---

(1) cfr. ouvrage repris dans la bibliographie sous le numéro 4.
(2) cfr. ouvrage repris dans la bibliographie sous le numéro 6.

a) Exposé de la méthode.

Elle est basée sur la notion de rubrique qui représente le plus petit lot d'informations qui puisse être utilisé de façon autonome. Elle peut être assimilée à la notion de type de propriété dans B.B.B.C.. Après avoir recensé toutes les rubriques auprès de l'utilisateur, on effectue une sélection parmi celles-ci: on conserve uniquement celles qui sont non calculées, c'est-à-dire que leurs valeurs ne dépendent pas des valeurs d'autres rubriques. On demande alors à l'utilisateur d'exprimer toutes les relations fonctionnelles existant entre celles qui restent.
On dresse une matrice MAT(i,j) i=1,n; j=1,n dont les lignes et les colonnes reprennent les n rubriques sélectionnées. Chaque élément MAT(i,j) peut prendre 2 valeurs: 1 si la rubrique de la colonne j est en relation fonctionnelle avec la rubrique de la ligne i, 0 dans le cas contraire. Toute colonne MAT(i,j_1) i=1,n (j_1 fixé) telle qu'il existe plus de un élément ayant la valeur 1, détermine un ensemble de base ou groupe primaire d'ordre 1. Il est composé des rubriques correspondant aux indices i pour lesquels les éléments valent 1 et il est indexé par la rubrique correspondant à la colonne j_1.
Dans un deuxième temps, l'utilisateur exprimera de nouvelles relations fonctionnelles entre les index de ces ensembles et les rubriques qui n'ont pas été reprises lors de la première étape. Cette opération traduira les associations entre les ensembles de base ou groupes primaires d'ordre n ($n \geq 2$) (index formé de n rubriques).
Illustrons cette méthode par un exemple.
Soient les rubriques NOMATR=numéro de matricule
NOMCLT=nom du client
ETCV=état civil
NOAGENCE=numéro d'agence
TPAGENCE=type d'agence
NOSERVI=numéro de service
NMSERV=nom du service
TYPCPTE=type de compte
SOLDE=solde du compte
DEVISE=devise du compte

1ère étape: construire la matrice (expression des relations fonctionnelles entre rubriques)

	NOMATR	NOMCLI	ETCV	NOAGENCE	TPAGENCE	NOSERVI	NMSERV	TYPCPTE	SOLDE	DEVISE
NOMATR	1	0	0	0	0	0	0	0	0	0
NOMCLI	1	1	0	0	0	0	0	0	0	0
ETCV	1	0	1	0	0	0	0	0	0	0
NOAGENCE	0	0	0	1	0	0	0	0	0	0
TPAGENCE	0	0	0	1	1	0	0	0	0	0
NOSERVI	0	0	0	0	0	1	0	0	0	0
NMSERV	0	0	0	0	0	0	1	0	0	0
TYPCPTE	0	0	0	0	0	0	0	1	0	0
SOLDE	0	0	0	0	0	0	0	0	1	0
DEVISE	0	0	0	0	0	0	0	0	0	1

Cette première étape nous fournit 2 ensembles de base, 2 groupes primaires d'ordre 1:

- (NOMATR, NOMCLT, ETCV) indexé par NOMATR
- (NOAGENCE, TPAGENCE) indexé par NOAGENCE

2ème étape: détermination des associations entre ensembles de base.

Les index et les rubriques non reprises à l'étape précédente sont NOMATR, NOAGENCE, NOSERVI, NMSERV, TYPCPTE, SOLDE et DEVISE. On exprime les relations fonctionnelles entre une

ou plusieurs rubriques.

Exemple: A un numéro de matricule, un numéro d'agence et un type de compte ne correspond qu'un solde.

On marque par le signe "S" la(les) rubrique(s) qui est (sont) "source(s) d'association" et par "B" celle qui est "but d'association".

NOMATR	S	-	
NOAGENCE	S	S	
NOSERVI		S	
NMSERV		B	
TYPCPTE	S		
SOLDE	B		
DEVISE	B		

Ce tableau nous fournit un groupe primaire d'ordre 3 et un autre d'ordre 2, à savoir respectivement:

(NOMATR, NOAGENCE, TYPCPTE, SOLDE, DEVISE)

et (NOAGENCE, NOSERVI, NMSERV)

b) Critique de la méthode.

-Avantages.

Les algorithmes qui nous fournissent les groupes primaires d'ordre 1 et d'ordre $n(n \geq 2)$ sont faciles à réaliser. Ils sont surtout intéressants par le fait qu'ils permettent de synthétiser les éléments de l'analyse: répertoire des rubriques et des relations fonctionnelles.

Le parallèle entre les concepts de B.B.B.C. et ceux exposés par Flory est évident: les objets correspondent aux groupes primaires d'ordre 1 et les associations aux groupes primaires d'ordre n. Cependant les ensembles construits par la méthode de Flory présentent un avantage certain quant au passage du réel perçu au schéma conceptuel: en effet, ils sont aisément transformables en relations représentant les ensembles de base et les associations entre ceux-ci. Ces relations seront directement sous 3ème forme normale au sens défini par Codd (1): cela provient du fait que la méthode est basée sur la détermination de toutes les relations fonctionnelles entre les rubriques.

-Inconvénients.

Les algorithmes nous sont présentés comme une facilité pour déterminer les ensembles de base, déchargeant l'utilisateur de cette tâche. Il est vrai que les problèmes disparaissent lorsqu'on arrive au stade de son application. Mais il nous semble que cette méthode s'attarde trop sur la nécessité d'un tel outil et néglige la phase antérieure, à savoir la collecte de données. En effet, avant de commencer, nous sommes obligés de réunir toutes les rubriques,

(1) cfr. chapitre IV.3.3.

toutes les relations fonctionnelles. Est-ce possible, est-ce réalisable? Ce recensement risque d'être plus ou moins long; au cours de cette étape, des divergences peuvent apparaître entre utilisateurs au sujet du rôle de chacune de ces rubriques. Il existe des données dont la sémantique est la même et qui sont manipulées par des utilisateurs différents: chacun d'eux peut être influencé par le milieu dans lequel il travaille pour donner les relations fonctionnelles entre les rubriques.
Si après la détermination des groupes primaires d'ordre 1 et d'ordre n, un utilisateur ajoute une nouvelle rubrique ainsi que de nouvelles relations fonctionnelles concernant cette rubrique, qu'advient-il de ces ensembles de base? Il sera alors nécessaire de réitérer l'algorithme avec les données mises à jour. Il nous semble que cette démarche manque de souplesse. Nous devrions pouvoir, lors de l'adjonction de nouvelles rubriques, prévoir les modifications ou du moins, les localiser à l'intérieur du graphe représentant les ensembles de base et leurs associations.

VI.1.2. Méthodes "top-down".

VI.1.2.1. Définition.

"Les méthodes "top-down" procèdent essentiellement par spécification progressive des solutions d'automatisation et d'organisation à partir des objectifs mentionnés par la direction. Chaque étape amène la prise en considération d'éléments qui développent ceux exprimés à l'étape précédente." (1)

---

(1) cfr. ouvrage repris dans la bibliographie sous le numéro 4.

VI.1.2.2. Critiques .

"Si ces méthodes sont susceptibles de conduire à des solutions plus cohérentes et mieux intégrées, elles exigent une collaboration active des utilisateurs à tous les niveaux de responsabilité concernés, sinon elles risquent de conduire à des solutions qui ne sont pas adaptées à leurs problèmes." (1)

VI.1.2.3. J.D. Warnier: "L'organisation des données d'un système."

Comme nous l'avons fait pour les méthodes de type "bottom-up", nous prenons une méthode de type "top-down" pour illustrer cette démarche. Elle est issue de l'article de M. Dambrine qui s'inspire des idées de J.D. Warnier. (2)

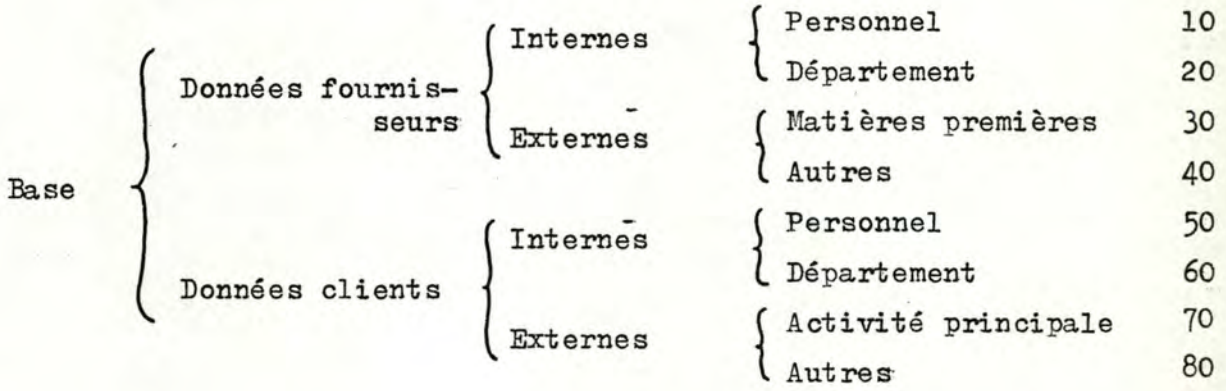
a) Exposé de la méthode.

Cette méthode a comme objectif, d'une part de recenser les rubriques nécessaires pour représenter le système et, d'autre part, de fournir une structure de rangement pour ces rubriques. Elle est basée sur un découpage de l'organisation en fonction de sa structure, de ses interactions avec l'extérieur, de ses activités.
Dans un premier temps, cette décomposition comporte trois niveaux. Le premier consiste en la distinction entre les clients et les fournisseurs. Le second éclate les précédents en données externes et internes (internes=rapport entre les différents services, départements de l'entreprise).

---

(1) cfr. ouvrage repris dans la bibliographie sous le numéro 4.
(2) cfr. ouvrage repris dans la bibliographie sous le numéro 7.

Le troisième tient compte des tiers en relation avec l'entreprise et des objets qu'ils échangent avec celle-ci; ce qui nous fournit le schéma suivant:



Chaque extrémité forme ce qu'on appelle une "base". Chacune de celles-ci est de nouveau subdivisée en "fichiers logiques de base" en considérant que les données se rapportent :

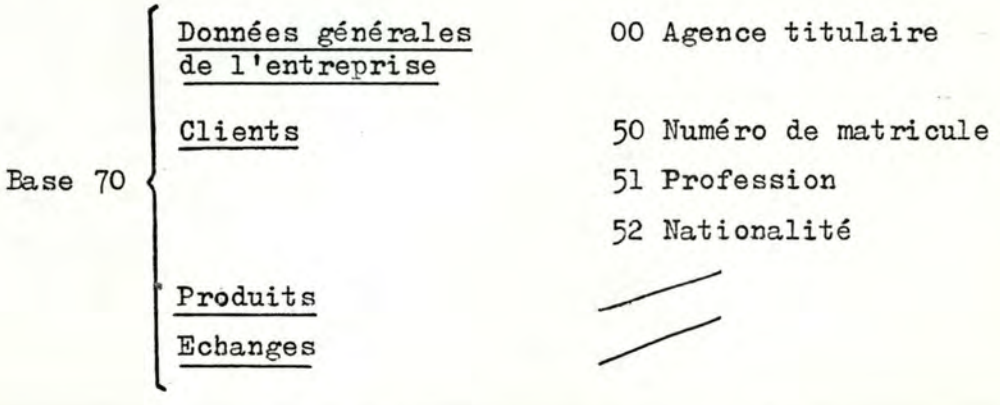
-à l'entreprise
-aux tiers
-aux produits de l'échange
-aux échanges proprement dits

Certains de ces ensembles peuvent être vides.

Bien que le schéma ci-dessus ne s'adapte pas tout à fait au problème de la banque(clients et fournisseurs externes se confondent; dans notre description, nous ne nous sommes pas attardés sur les échanges entre les agences, les services), nous allons néanmoins illustrer cette méthode par trois exemples.

1) Création d'un nouveau client, collecte des renseignements généraux le concernant.

Nous reprenons la base 70 que nous éclatons comme suit:



2) Création d'un compte.

Base 70	Données générales de l'entreprise	00 Agence titulaire du compte
	Clients	50 Numéro de matricule
	Produits	60 Type de produit
		61 Type de compte
		62 Devise
		⋮
	Echanges	

3) Mouvements sur un compte.

Base 70	Données générales de l'entreprise	00 Agence
	Clients	50 Numéro de matricule
	Produits	61 Type de compte
	Echanges	80 Date du mouvement
		81 Montant du mouvement
		⋮

Chaque fois que nous décrirons un événement, une situation, nous nous réfèrerons toujours au schéma. Nous procéderons, suivant les besoins exprimés par les utilisateurs, à des fusions, des regroupements de fichiers logiques de base. Il sera également indispensable de définir les identifiants, les index de chacun de ces fichiers.

b) Critique de la méthode.

-Avantages.

Nous soulignerons d'abord la nécessité de trouver un schéma de référence qui soit conforme à la structure de l'entreprise, aux directives énoncées par le gestionnaire. Contrainte ou non, il nous semble que cela présente l'avantage de fournir un outil permettant de mieux situer les situations rencontrées durant le processus d'analyse. Et ceci correspond à l'objectif premier de cette méthode: tenter de rendre plus souple la façon de concevoir, de diriger l'analyse du système à informatiser, ce qui est nécessaire surtout du point de

vue de la maîtrise de la complexité et du gigantisme.

Si dans la méthode "bottom-up" proposée par Flory, l'utilisateur ne participe pas directement à la conception du système, il n'en est pas de même pour la méthode de Warnier: "La partie fonctionnelle est de leur ressort, l'informaticien aura surtout un rôle de conseiller en logique, leur signalant les conséquences des options qu'ils choisissent."(1)

Si la méthode "bottom-up" de Flory nécessite le recensement préliminaire de l'ensemble des rubriques pour la représentation de l'organisation, la méthode "top-down" de Warnier n'est pas sujette à cette contrainte; bien plus, il nous semble qu'elle permet l'approfondissement de certaines parties du système sans se soucier des interactions avec les autres. Vue sous cet angle, elle nous paraît moins rigide.

-Inconvénients.

Le principal inconvénient est constitué par le fait qu'une telle méthode nécessite une cohésion et une participation totale de l'ensemble du personnel à l' élaboration des bases et des fichiers logiques de base. Tous doivent se référer au même schéma afin d'éviter, de la part des analystes, des prises de décision qui ne sont pas de leur ressort. Une des conditions primordiales requises des membres de l'équipe pour l'application des méthodes de type "top-down" est la discipline, la rigueur.

Dans un tout autre domaine, nous pouvons regretter l'absence d'un algorithme du type de celui proposé par Flory: ce dernier, en plus de nous fournir les groupes primaires d'ordre n (n=1,...), nous donne également les index de ces ensembles et les liaisons entre ces différents lots d'informations. Il n'en est pas de même pour la méthode de Warnier: celle-ci nous délivre aussi les ensembles d'informations, mais ne nous donne aucun renseignement sur les index et sur les liens entre ces fichiers logiques de base.

---

(1) cfr. ouvrage repris dans la bibliographie sous le numéro 7.

VI.2. Démarche personnelle.

La démarche suivie pour la structuration des données du système bancaire diffère de celle suivie pour la description des traitements.

VI.2.1. Structuration générale du système.

L'article "Concepts for the design of a conceptual schema"(1) ne propose pas de méthode d'analyse: comme son titre l'indique, il nous livre uniquement des concepts permettant de décrire un schéma conceptuel.

Nous avons vu précédemment que la détermination des objets et des associations peut s'effectuer aussi bien par une approche "top-down"(Warnier) que par une approche "bottom-up"(Flory); notre démarche est apparentée à ces deux types de méthodes: elle est du type "top-down" en ce qui concerne la détermination des objets et des associations, et du type "bottom-up" pour ce qui est du passage à une représentation sous forme de relations.

VI.2.1.1. Détermination des objets et des associations.

Nous avons abordé ce sujet au chapitre IV (2) et il est repris en détails en annexe I. Nous rappellons le raisonnement qui nous a aidé dans la constitution du répertoire des objets:

> La banque PARIBAS est composée de plusieurs agences identifiées par un numéro qui leur est propre. Chacune d'elles se décompose en services; elles ont des clients(ou fournisseurs suivant le point de vue où l'on se place) identifiés par un numéro de matricule. Pour chacun de ceux-ci, on donnera ses caractéristiques et sa (ses) adresse(s) indipensable(s) pour la correpondance. La banque vend des produits(ou rend des services suivant le rôle que l'on assigne à un organisme bancaire).

Lorsque nous avons déterminé un objet, nous nous intéressons à ses propriétés: celles qui permettent d'identifier l'objet, de le localiser dans le temps et dans l'espace.

---

(1) Cfr. ouvrage 1. dans la bibliographie.
(2) Cfr. IV.1.1.1. et IV.1.1.2.

Disposant de ces objets, nous nous penchons sur le problème de leurs associations éventuelles qui traduisent les liens entre les différents objets élémentaires. Pour ceux-ci également, nous recensons l'ensemble de leurs propriétés:

> Un compte se réfère à un type de produit et n'existe qu'en relation avec le client qui le détient et l'agence qui en assure la tenue. Il en est de même pour les crédits, la location des coffres,...

Notre démarche pour la détermination des objets et des associations est étrangement liée à celle préconisée par Warnier: les tiers (clients) sont en relation avec l'entreprise(agence) avec laquelle ils effectuent des échanges(ventes des produits). Le schéma CLIENT - AGENCE - PRODUIT reflète bien la base représentative d'un organisme bancaire.

VI.2.1.2. Représentation formelle: les relations.

Sur l'exemple choisi au chapitre IV (1), nous avions opéré le passage de l'ensemble des objets, des associations et de leurs propriétés respectives à une représentation sous forme de relations. Sur quels principes s'est-il effectué? Analysons les différentes étapes de cette démarche:

1. Toutes les propriétés des objets et des associations reçoivent un nom symbolique. On en précise le format, on détermine l'ensemble des valeurs que peut prendre cet élément ainsi que toutes les caractéristiques possibles (valeur minimale, valeur maximale, valeur(s) impossible(s), cardinalité,...). Cette première étape nous fournit la rubrique DATA dont il est question dans B.B.B.C. .

   Exemple: NOMATR 'numéro de matricule'

FORMAT	FORM-n	DECIMAL 8 MIN=55000000 MAX=55999999
	IC-m	CARDINAL 30000, 50000

2. La seconde étape consiste en la mise sous forme de relations des objets et des associations. Nous avons considéré qu'un objet (ou une association) et les propriétés qui lui sont attachées constituent un ensemble homogène. Au sein de celui-ci, nous avons appliqué un algorithme du type de Flory: en déterminant toutes les relations fonctionnelles entre les

(1) cfr. Chapitre IV.1.2.

éléments, nous avons constitué différents sous-ensembles que nous avons transformés en relations. Celles-ci, comme nous l'avons précisé lors de la discussion sur la méthode utilisée par Flory, sont sous 3ème forme normale, but que nous nous étions fixé. Il ne nous restait plus qu'à exprimer les différentes propriétés (contraintes) de ces relations (clé, composantes obligatoires, nombre d'occurrences, contraintes sur la relation toute entière, sur certains domaines en particulier...). Pour plus de détails, nous vous renvoyons en annexe II où se trouve la description complète de toutes les relations que nous avons utilisées.

VI.2.1.3. Difficultés rencontrées.

On peut ne pas être d'accord avec le point de vue qui a prévalu dans la décomposition du système: c'est-à-dire la notion de "produit". La structuration aurait été aussi équivalente si l'on avait considéré les comptes, les crédits, la location des coffres, les titres,... comme des objets distincts. Cette notion de "produit" nous permet de faire le rapprochement entre une banque et une entreprise quelconque.

La principale difficulté rencontrée est la détermination des objets et des associations: elle dépend de la perception que chaque utilisateur a du réel.

Exemples:

-Outre les caractéristiques générales du client, celui-ci possède une ou plusieurs adresses. Distingue-t-on un objet CLIENT et un objet ADRESSE?

-De même pour un compte: outre les caractéristiques générales du compte, celui-ci peut posséder des conditions particulières pour le calcul des intérêts. Forment-elles un objet particulier?

Ce problème est assez ambigu et se confond souvent, non plus à la détermination des objets et des associations, mais à leur représentation sous forme de relations mises sous 3ème forme normale.

VI.2.2. Règles d'évolution.

Pour la description des règles d'évolution (les traitements), la démarche fut du genre "bottom-up".
Le schéma RESULTATS→TRAITEMENTS→DONNEES reflète bien le raisonnement que nous avons suivi.

Considérons le calcul des intérêts sur un compte à vue (1). Le résultat que nous voulons obtenir est composé du solde et des intérêts cumulés d'un compte donné à une date bien précise. Pour effectuer cette opération, il nous faut tous les mouvements sur ce compte depuis le dernier calcul des intérêts jusqu'à la date considérée. A l'aide de ces éléments, nous calculons les soldes successifs du compte que nous multiplions par certains taux (taux standards ou taux particuliers si le compte jouit de conditions particulières). Ces traitements nous permettront de mettre à jour le solde et les intérêts cumulés du compte.

Cette séquence d'opérations nous a amenés à la constitution d'un ensemble de relations utiles pour effectuer les différents calculs. Nous les reprenons ci-après:

-INTERET: sorte d'historique des comptes quant au déroulement des calculs successifs des intérêts. Chaque occurrence contient un numéro de compte, la date du calcul des intérêts, le solde et les intérêts cumulés à cette date. Pour l'opération qui nous intéresse, il suffit de sélectionner l'occurrence correspondant au dernier calcul des intérêts et d'en créer une nouvelle avec les différents domaines mis à jour.

-MOUVEMENTS: chaque occurrence contient un numéro de compte, une date et le montant du mouvement à cette date.

-COMPTE: les renseignements utiles de cette relation sont les codes devise et marché.

-CLIENT: nous fournit le code série de compte du client.

-Les données des relations COMPTE et CLIENT nous donnent le moyen d'accéder aux informations des relations contenant les différents taux pour le calcul des intérêts.

---

(1) Pour plus de détails, voir annexe III.

Pour la description des traitements, il est fréquent de faire référence aux informations de la structure générale (ou à une partie) et de créer des relations groupant les seuls éléments que l'on juge nécessaires pour le développement de l'application. Le processus de description des procédures fait très souvent appel à la désignation de ces seules données utiles. Les objets manipulés par les utilisateurs sont ceux qu'ils créent en fonction de leurs besoins(relations intermédiaires pour les calculs = zones de travail) et des résultats qu'ils veulent fournir.

VI.3. Méthode d'analyse: "approche base de données".

De tout ce qui vient d'être dit, nous allons dégager ce qui nous paraît être le plus intéressant pour l'analyse d'un système en vue de la mise en place d'une base de données. Plus exactement, notre attention se portera sur la méthode à suivre pour parvenir à la construction de la structure conceptuelle. Comme nous l'avons dit dans le premier chapitre, la tendance actuelle est de distinguer plusieurs niveaux dans la conception et l'élaboration d'une base de données. Notre réflexion portera sur le passage du monde réel au réel perçu, et de ce dernier au niveau conceptuel.

VI.3.1. Du monde réel au réel perçu.

Il nous semble important de s'attarder dès le début de l'analyse au passage du réel au réel perçu. Et cela pour une raison que l'on peut qualifier d'efficacité. Le niveau conceptuel s'appuie pour sa description sur des concepts rigoureux: relation, clé, relation fonctionnelle, contrainte d'intégrité,... Il a comme propriétés, entre autres, d'être le plus universel possible, un point d'accord entre tous les utilisateurs quant à son contenu. Peut-on arriver à une représentation de ce genre en partant d'une conception de l'organisation basée sur une perception totalement empirique? (1)

Nous avons mentionné, lors de notre réflexion sur les méthodes "top-down", la nécessité de trouver une image de la réalité à laquelle

(1) Tiré de l'ouvrage repris dans la bibliographie sous le numéro 8.

on puisse faire référence. Cette façon d'opérer fait apparaître les principaux éléments du système et leurs liaisons. La structuration sous forme d'objets et d'associations permet de dénombrer des ensembles d'informations stables. Cette perception de la réalité sera conforme à la structure de l'entreprise, aux options désirées par la direction, aux besoins exprimés par les utilisateurs.

La mise en place d'une base de données remplaçant un système "décentralisé" existant bouleverse bon nombre d'habitudes : chacun travaille dans son coin, développe ses propres applications, crée ses propres fichiers, sans se soucier du travail accompli par le voisin. La non-redondance dans la prise d'informations est assurée par la détermination des types d'objets et des types d'associations et est le point de départ d'une réorganisation effective du système.

VI.3.2. Du réel perçu au modèle conceptuel.

Le but essentiel de ce passage est de représenter les objets et les associations sous forme de relations (c'est le formalisme qui a été retenu dans B.B.B.C.). Lors de l'analyse de la méthode préconisée par Flory, nous avons vu que l'algorithme proposé nous fournissait les ensembles de base et les associations entre ceux-ci et qu'il était aisé de les transformer en relations mises sous $3^{ème}$ forme normale. Nous lui avions fait le reproche de ne pas être assez souple lors de l'adjonction de nouvelles données.

Si nous appliquons le processus de Flory, non plus à l'entièreté des rubriques représentatives de la structure générale du système, mais seulement dans le cadre restreint des propriétés relatives à un objet (ou une association) déterminé(e), il nous semble que cela remédie au problème que nous soulevons ci-dessus. Lors de la modification ou de l'adjonction de propriétés à un objet, l'algorithme ne travaillera que sur les données relatives à celui-ci.

Cette démarche assure une cohésion, une stabilité dans le processus de décomposition des objets et des associations (réel perçu) en relations mises sous $3^{ème}$ forme normale (représentation formelle).

TROISIEME PARTIE -

EVALUATION CRITIQUE: INDEPENDANCE LOGIQUE

Dans cette troisième partie, notre attention se portera donc sur la structure conceptuelle du point de vue de son apport vis-à-vis de l'indépendance logique.

Au début du septième chapitre, nous situerons le problème, ce qui nous amènera à définir les notions de sous-schéma et de "mapping" d'une structure dans une autre. Nous appliquerons cette dernière notion au passage de la représentation conceptuelle à une structure d'accès qualifiée de théorique.

Le passage de cette dernière à une structure d'accès implémentée en IDS sera exposé brièvement dans le huitième chapitre.

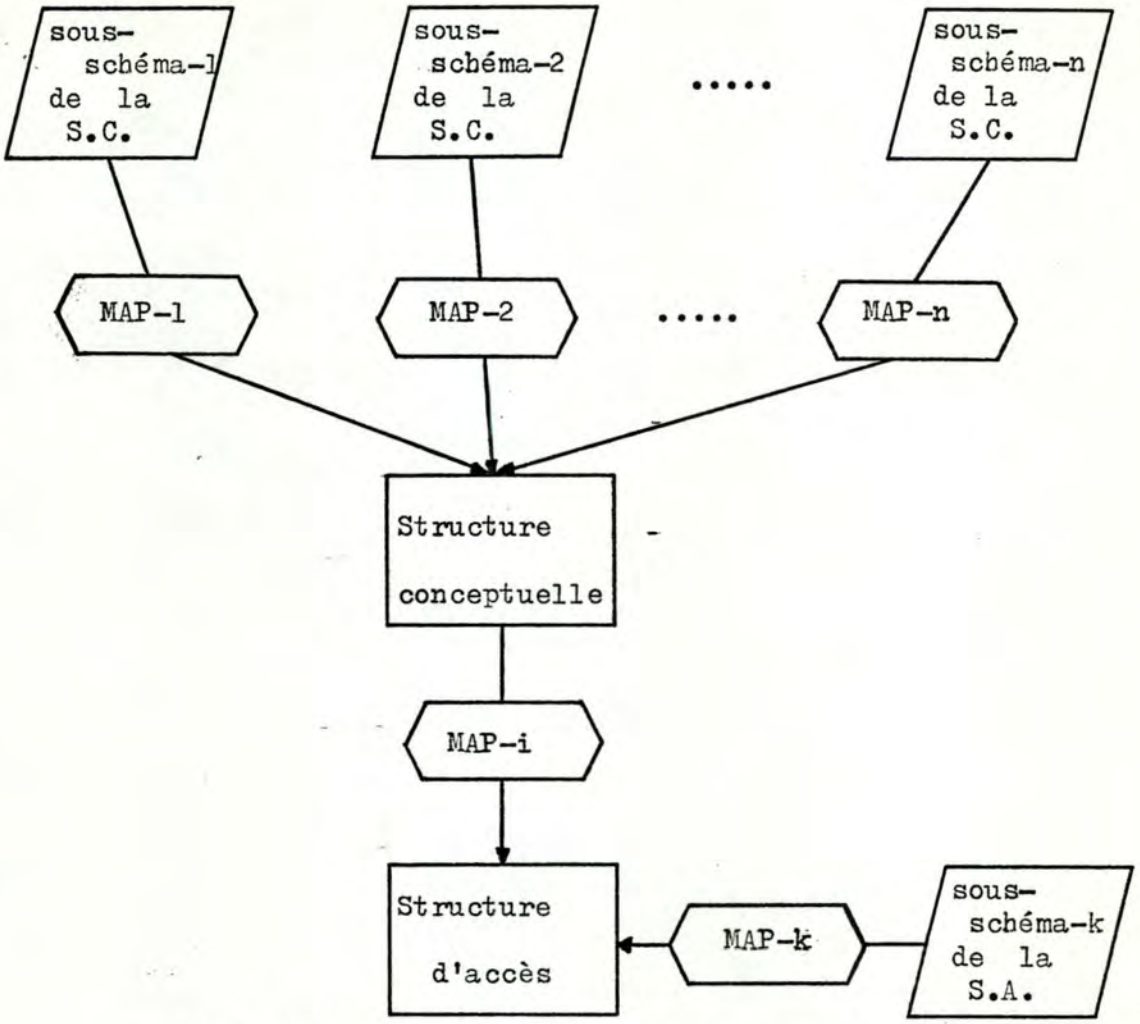
CHAPITRE VII. INDEPENDANCE LOGIQUE

Le schéma conceptuel a été présenté comme un outil d'indépendance logique: il offre la possibilité aux utilisateurs d'exprimer leurs besoins sur une structure relativement stable et indépendante de toutes considérations physiques, technologiques. Elle leur permet d'ignorer les modifications survenant à ce niveau.

Au début de ce chapitre, nous ferons un bref rappel au sujet de la structure générale de décomposition du système en niveaux, afin de situer à quels stades se présente cette notion d'indépendance (ceci nous permettra d'introduire la notion de sous-schéma); nous définirons ensuite ce que l'on entend par "mapping" d'une structure dans une autre; nous indiquerons les besoins et les avantages que présente cet outil de transformation. Nous tenterons d'appliquer, à partir d'exemples, les règles qui permettent de passer d'une structure à une autre: cette étude se fera tant au niveau de l'organisation générale des données (structure conceptuelle statique) qu'au niveau de la manipulation de celles-ci(aspect dynamique des règles d'évolution).

VII.1. Structure générale.

Au début de ce travail, nous avons rappelé en quoi consistait la décomposition en niveaux sur laquelle se base une des démarches actuelles dans l'analyse et la conception des bases de données. Nous nous intéresserons plus particulièrement dans ce chapitre aux structures conceptuelle et d'accès, ou plus exactement au passage de l'une à l'autre. Illustrons grossièrement l'architecture du système par la figure suivante:



VII.1.1. Structure conceptuelle et structure d'accès.

Nous ne reviendrons pas sur les définitions de ces deux structures. Rappelons simplement que la première constitue la représentation formelle de la structure d'informations décrivant l'entreprise. Elle a pour but de faire apparaître les objets manipulables dont la perception est commune aux concepteurs et aux utilisateurs de la base de données. La seconde est la structure qui sera physiquement implémentée et qui décrit les seuls chemins d'accès aux données auxquelles les programmeurs pourront faire appel.

La sémantique exprimée à chaque niveau est la même, chacun d'eux présente et organise les données de façon différente.

VII.1.2. Sous-schémas.

La notion de sous-schéma correspond à la description des données qui intéressent certains utilisateurs dans le cadre bien précis d'une ou de plusieurs applications qui sont de leur ressort.

VII.1.2.1. Sous-schémas conceptuels.

Il est évident que le modèle conceptuel d'une base de données peut être élaboré en une seule étape. Chaque utilisateur possède un point de vue qui lui est propre au sujet de la structure des informations qu'il manipule. Le schéma conceptuel a pour but de synthétiser ces différentes approches pour constituer un modèle de référence pour l'organisation.

> "Le résultat de l'intégration des structures logiques particulières doit être un modèle général des vues des utilisateurs, fournissant une représentation non redondante et exhaustive des informations élémentaires". (1)

Néanmoins, il est possible que ce schéma conceptuel unique ne contente pas tout à fait certaines catégories d'utilisateurs. Cela peut être dû, par exemple, à la détermination des types de relations: une autre décomposition de l'ensemble des données serait sans doute aussi adaptée au problème particulier qu'il traduit. Cette possibilité leur est offerte dans la construction de sous-schémas conceptuels.

D'autres raisons conduisent à une telle nécessité. Le formalisme employé pour le schéma conceptuel ne convenant pas à certains utilisateurs, on leur permet de décrire leurs informations à l'aide de leur propre langage. Par exemple, leurs données seront présentées suivant le canevas de la DATA DIVISION du COBOL.

Le rôle de l'administrateur de la base des données est de veiller à ce que les sous-schémas soient bien dérivables du schéma conceptuel. En particulier, chacun d'eux doit vérifier les contraintes d'intégrité exprimées sur le schéma conceptuel pour que la structure toute entière reste cohérente.

---

(1) tiré du cours MIS de Monsieur Bodart, repris dans la bibliographie sous le numéro 4.

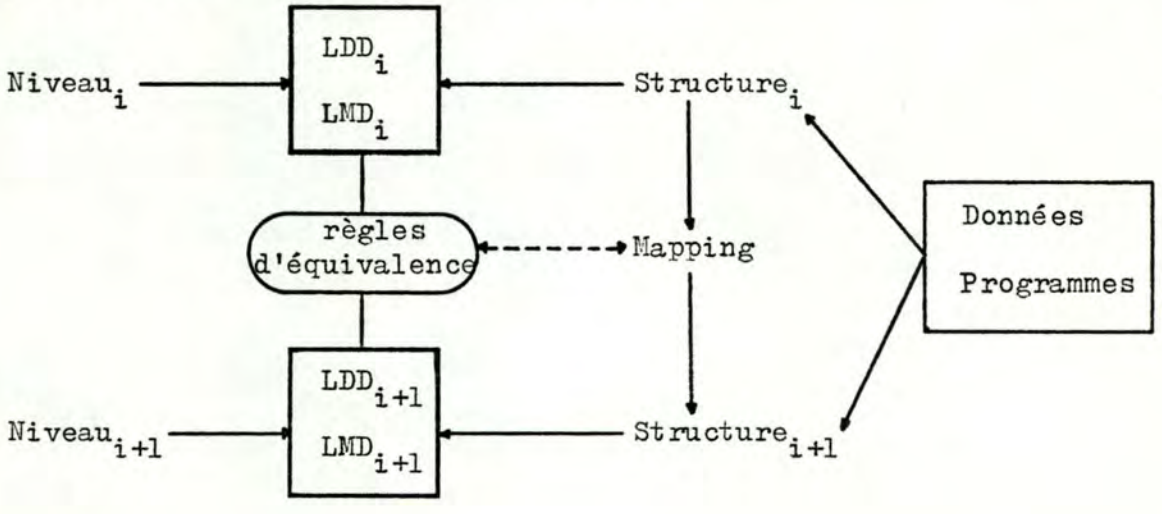
VII.1.2.2. Sous-schémas de la structure d'accès.

De manière analogue, on donne la possibilité aux utilisateurs de créer leurs propres sous-schémas au niveau de la structure d'accès . Ce besoin peut résulter d'une organisation différente des données dans la définition des articles ou d'une détermination de chemins d'accès mieux adaptés aux problèmes qu'ils ont à résoudre.

VII.2. Mapping d'une structure dans une autre.

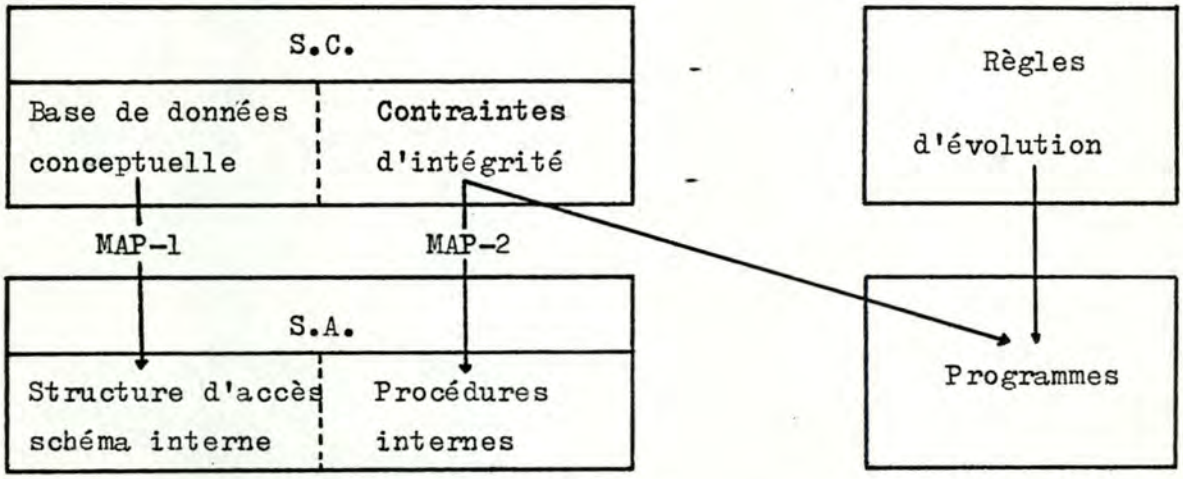
VII.2.1. Description.

Chaque niveau utilise un formalisme qui lui est propre(tant au niveau de la structure conceptuelle et de la structure d'accès qu'au niveau des sous-schémas). Ce formalisme s'appelle un langage de description des données(L.D.D.). Nous n'avons guère parlé jusqu'à présent de l'aspect dynamique du système; la manipulation des données mémorisées joue un rôle très important dans l'expression des traitements à réaliser sur les différentes structures. Chaque structure est dotée d'un langage de manipulation des données qui lui est propre(L.M.D.). Des règles d'équivalence établiront la correspondance entre les "verbes" du L.D.D. et du L.M.D. d'une structure et ceux du niveau qui le suit immédiatement.
Le mapping est la fonction qui permet d'exprimer la $structure_{i+1}$ en fonction de la $structure_i$ en respectant ces règles d'équivalence. Ceci peut se résumer dans le schéma suivant:



VII.2.2. Mapping de la sructure conceptuelle dans la structure d'accès.

Analysons comment s'effectue le passage de la structure conceptuelle à la structure d'accès.(1)



La description des données sans aucun souci d'implémentation, que nous appelons "la base de données conceptuelle", sert d'entrée au mapping qui la transforme en une représentation conforme à la structure d'accès retenue (MAP-1). Les données seront organisées suivant le SGBD choisi. On appellera "schéma interne" la description d'une structure d'accès à l'aide d'un langage.

Les propriétés de ces données, les contraintes d'intégrité exprimées sur la base de données conceptuelle sont transformées d'une part, en procédures internes au SGBD (MAP-2) et, d'autre part, en une partie des programmes d'application qu'il sera nécessaire d'écrire. Il n'y a pas une seule structure d'accès possible mais plusieurs: suivant la structure retenue, certaines contraintes pourront être ou non prises en compte. Les performances d'exécution des programmes seront différentes suivant la forme que l'on donnera aux données dans la structure d'accès. Il y aura donc un compromis à faire entre les performances attendues et les prises en compte automatiques de certaines propriétés.

Les règles d'évolution seront transformées en programmes d'application.

(1) cfr. ouvrage repris dans la bibliographie sous le numéro 1.

VII.2.3. Besoins de mapping.

VII.2.3.1. Augmentation du nombre des utilisateurs.

Les bases de données ont pour but, entre autres, de centraliser les données en vue d'éviter la redondance des informations, problème crucial quant à la validité des données à un instant déterminé. Une base de données possède également la propriété d'être accessible, partagée par de nombreux utilisateurs. L'expression de leurs besoins, de leurs demandes ne peut se faire que sur la structure d'accès: c'est elle qui permet la programmation, qui définit les seuls chemins d'accès qu'ils peuvent emprunter. Cela implique qu'ils doivent se plier aux contraintes de la représentation de la structure d'accès, du système de gestion de la base de données qui la sous-tend.

La possibilité offerte aux utilisateurs d'exprimer leurs besoins sur la structure conceptuelle, leur permet d'éviter les inconvénients que nous avons mentionnés précédemment: il n'y a pas de notion d'accès et la logique des prédicats possède des qualités de simplicité et de facilité d'expression. Ces avantages et la construction de sous-schémas mieux adaptés à leurs problèmes, sont deux raisons qui provoquent l'augmentation du nombre des utilisateurs.

VII.2.3.2. Point de vue du programmeur.

Qui dit augmentation du nombre des utilisateurs dit augmentation du nombre des programmes. Qu'en est-il de la programmation sur la structure d'accès d'une base de données? On distingue deux langages: le "langage de navigation" et le "langage hôte". Le premier, comme son nom l'indique, permet de parcourir la structure d'accès en suivant les chemins prédéfinis, de rechercher les données(IDS). A l'aide du second, on exprime les traitements que l'on fait subir à ces données(COBOL).

Les opérations élémentaires sur une base de données sont la création, la suppression et la modification. Au niveau conceptuel aussi bien qu'au niveau des accès, nous disposons des outils permettant

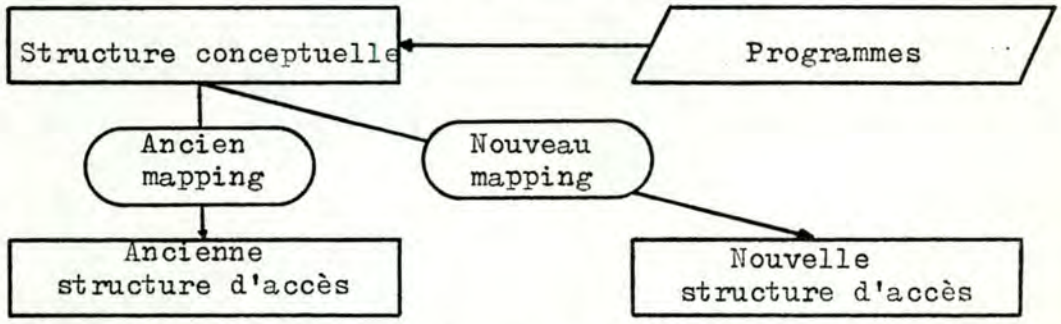
d'effectuer ces opérations. Pour l'utilisateur exprimant ses besoins sur la structure conceptuelle, tout se passe comme si son programme s'exécutait sur celle-ci, alors qu'en réalité, grâce au mapping, sa requête est transformée en un ou plusieurs accès à la base de données. Le mapping définit une fois pour toutes les fonctions réalisant la transformation de ces opérations élémentaires.

Ce qui vient d'être dit est aussi valable pour la déclaration des données aux différents niveaux(structure conceptuelle, structure d'accès et sous-schéma de ces deux structures). La rencontre d'une déclaration de données à un niveau déclenchera aussitôt une procédure de localisation sur le niveau suivant, effectuant ainsi la correspondance entre les deux représentations.

VII.2.3.3. Indépendance.

Une base de données est en perpétuelle évolution: la structure conceptuelle joue le rôle de référence dans l'organisation: toute modification des données doit être répercutée sur cette structure. Le but est de conserver une structure relativement stable de sorte que l'utilisateur puisse toujours y faire référence.

De modifications en modifications, nous pouvons en arriver à une situation telle que nous soyons obligés de décider, de repenser la structure d'accès et d'en construire une autre. Si tel est le cas, qu'advient-il des programmes des utilisateurs exprimés sur cette structure? Il est évident qu'il s'agit de trouver une solution minimisant le nombre de programmes à modifier. Analysons le schéma suivant:

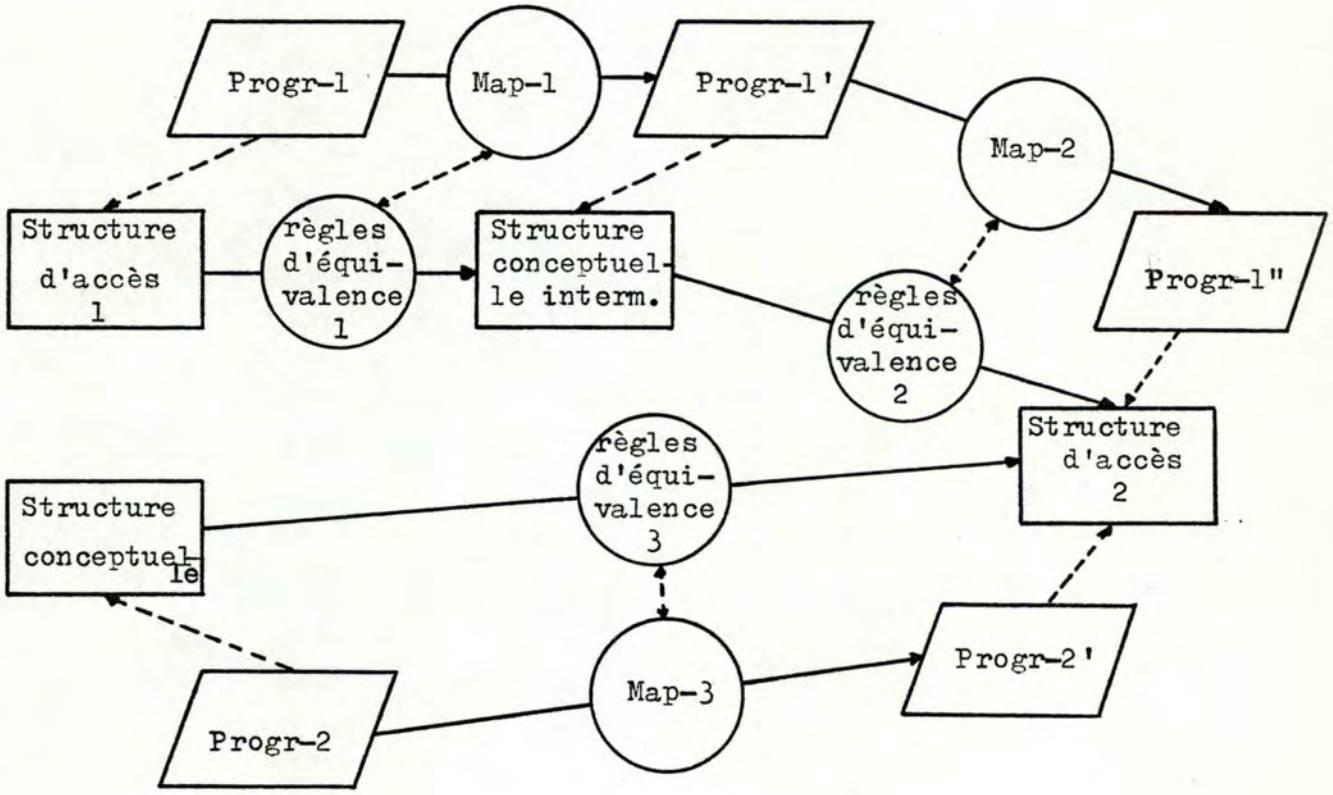


Les programmes exprimés sur la structure conceptuelle ne subiront aucune modification lors du passage de l'ancienne structure

d'accès à la nouvelle. Pour qu'ils puissent s'exécuter sur cette dernière, il suffit de modifier le mapping entre ces types de structure.

VII.2.3.4. Les cas à analyser.

Si nous nous reportons au schéma général du système(1), nous voyons qu'il existe deux types de mapping: mapping d'une structure conceptuelle dans une structure d'accès et mapping d'une structure d'accès dans une autre structure d'accès. Pour le mapping entre deux structures d'accès, il semble que la solution la plus facile soit de passer par une structure "conceptuelle" intermédiaire. Cela est dû à la difficulté d'exprimer des accès sur une structure en fonction des accès sur l'autre. Il est plus simple de supprimer cette notion d'accès sur la première en la transformant en une structure "conceptuelle"(sans accès), et ensuite, de les reconstruire sur la deuxième. Nous pouvons donc résumer ces considérations dans le schéma suivant:



(1) cfr. chapitre VII.1.

Les deux types de mapping nous font donc aborder les problèmes suivants: passage d'une structure conceptuelle à une structure d'accès et inversément.

VII.3. Méthodologie.

VII.3.1. Avertissements.

Le but de ce chapitre est de donner quelques grands principes qui régissent la transformation d'une structure conceptuelle en une structure d'accès. Il n'est pas dans notre intention de développer des algorithmes rigoureux et complets: nous nous contenterons d'illustrer tout ce qui sera dit au moyen de quelques exemples.

Après avoir rappelé et illustré les principaux concepts utilisés pour décrire les deux structures, nous nous donnerons des langages de manipulation de données sur chacune d'elles. Nous montrerons alors comment il est possible de passer de la description d'une structure conceptuelle à la description d'une structure d'accès. Nous effectuerons la même opération pour la dynamique du système: transformation de requêtes sur la structure conceptuelle en A-requêtes(ou requêtes sur la structure d'accès), passage des opérations élémentaires définies sur la première aux opérations élémentaires s'exécutant sur la seconde.

VII.3.2. Rappels.

VII.3.2.1. Principaux concepts pour décrire une structure conceptuelle.

a) Exemple.

Nous illustrons ces concepts par un exemple restreint;nous donnons ci-après les relations et les domaines qui les composent:

Domaines: NOAGENCE: numéro de l'agence
NOMAG: nom de l'agence
LOCALITE: localité
TPAGENCE: type de l'agence
NOMATR: numéro de matricule du client
NOM: nom du client
SIECPTE: code série de compte
NTMATR: nature du matricule
NOADR: numéro de l'adresse
RUE: rue
NO: numéro
CDPOSTAL: code postal
TYPCPTE: type de compte
SOLDE: solde du compte
AGTITUL: agence titulaire du matricule

Relations: AGENCE(NOAGENCE,NOMAG,LOCALITE,TPAGENCE) 'agence'
CLIENT(NOMATR,NOM,SIECPTE,NTMATR,AGTITUL) 'client'
ADRCLI(NOMATR,NOADR,RUE,NO,CDPOSTAL,LOCALITE)'adresse du client'
COMPTE(NOMATR,NOAGENCE,TYPCPTE,SOLDE) 'compte du client'

b) Données.

Les objets et les associations définis au niveau de la perception du réel possèdent différentes propriétés. Ces propriétés sont représentées par des données. Une donnée peut prendre plusieurs valeurs.

Exemple: Données: NOAGENCE, NOM,...
Valeurs: 552, DUPONT,...

c) Entité.

L'entité est une notion qui n'existe que dans la structure conceptuelle: elle permet de distinguer deux occurrences d'une même relation possédant les mêmes propriétés.

Exemple: Supposons que l'on envoie des extraits de compte aux clients avec les indications suivantes: numéro de compte, date et montant de l'opération. Pour identifier univoquement les occurrences de la relation EXTRCPTE reprenant ces informations, il est nécessaire de créer une entité EXCPT: il se peut qu'un client ait fait deux mouvements sur un même compte, à la même date et du même montant.

d) Relations.

Les relations représentent les objets et les associations;

c'est une partie du produit cartésien des domaines représentant les propriétés de ces objets et associations: c'est un ensemble de n-uples.
Exemple: relations AGENCE, CLIENT,...

e) Clé d'une relation.

Etant donné une valeur de la clé d'une relation, elle identifie une et une seule occurrence de cette relation.

Exemple: (NOAGENCE) : clé de la relation AGENCE.
(NOMATR) : clé de la relation CLIENT.
(NOMATR, NOAGENCE, TYPCPTE) : clé de la relation COMPTE.
(NOMATR, NOADR) : clé de la relation ADRCLI.

f) Projections.

- Projection d'une relation sur des domaines:

  Soit R(A,B,C,D):

  $$\text{Project}(R|(C,D)) = \{(c,d) \text{ tq } \exists a,b \text{ tq } (a,b,c,d) \in R\}$$

  Exemple: Nous désirons l'ensemble des numéros des agences.
  = (NOAGENCE)AGENCE.

- Projection d'une relation sur des valeurs et sur des domaines:

  Soit R(A,B,C,D):

  $$\text{Project}(R|(a,b,D)) = \{(a',b',d) \text{ tq } (a',b',d) \in R \wedge (a'=a) \wedge (b'=b)\}$$

  Nous noterons (a,b,D)R.

  Exemple: Nous désirons l'ensemble des soldes de tous les comptes des clients dont le matricule est 55678312
  =(55678312,SOLDE)COMPTE

g) Egalité et inclusion de relations.

Ce type de contraintes spécifie que les valeurs d'un ou de plusieurs domaines d'une relation sont incluses dans l'ensemble des valeurs d'un ou de plusieurs domaines d'une autre relation.

Exemple: l'ensemble des numéros de matricule de la relation ADRCLI est égal à l'ensemble des numéros de matricule de la relation CLIENT.
EQUAL (NOMATR)CLIENT (NOMATR)ADRCLI.(Tout client a au moins une adresse).

b) Relation fonctionnelle.

Si L1 et L2 sont deux sous-listes quelconques de domaines d'une relation R, on dit que(L1)R est en relation fonctionnelle avec (L2)R s'il existe une fonction f telle que (L2)R=fo(L1)R.

Exemple: à un numéro de matricule correspond un seul nom;
à une localité correspond un seul code postal.

i) Cardinalité.

Soit une relation R(L1,L2); une propriété de cardinalité s'exprime par

$\forall 1 (1 \in (L1)R) \Rightarrow (i \leq CARD((L1,L2)R)_{\leq} j)$ avec $i \geq 1$, $j \geq i$

Exemple: un client a au moins une adresse, au plus 10.

CARD-PROJEC

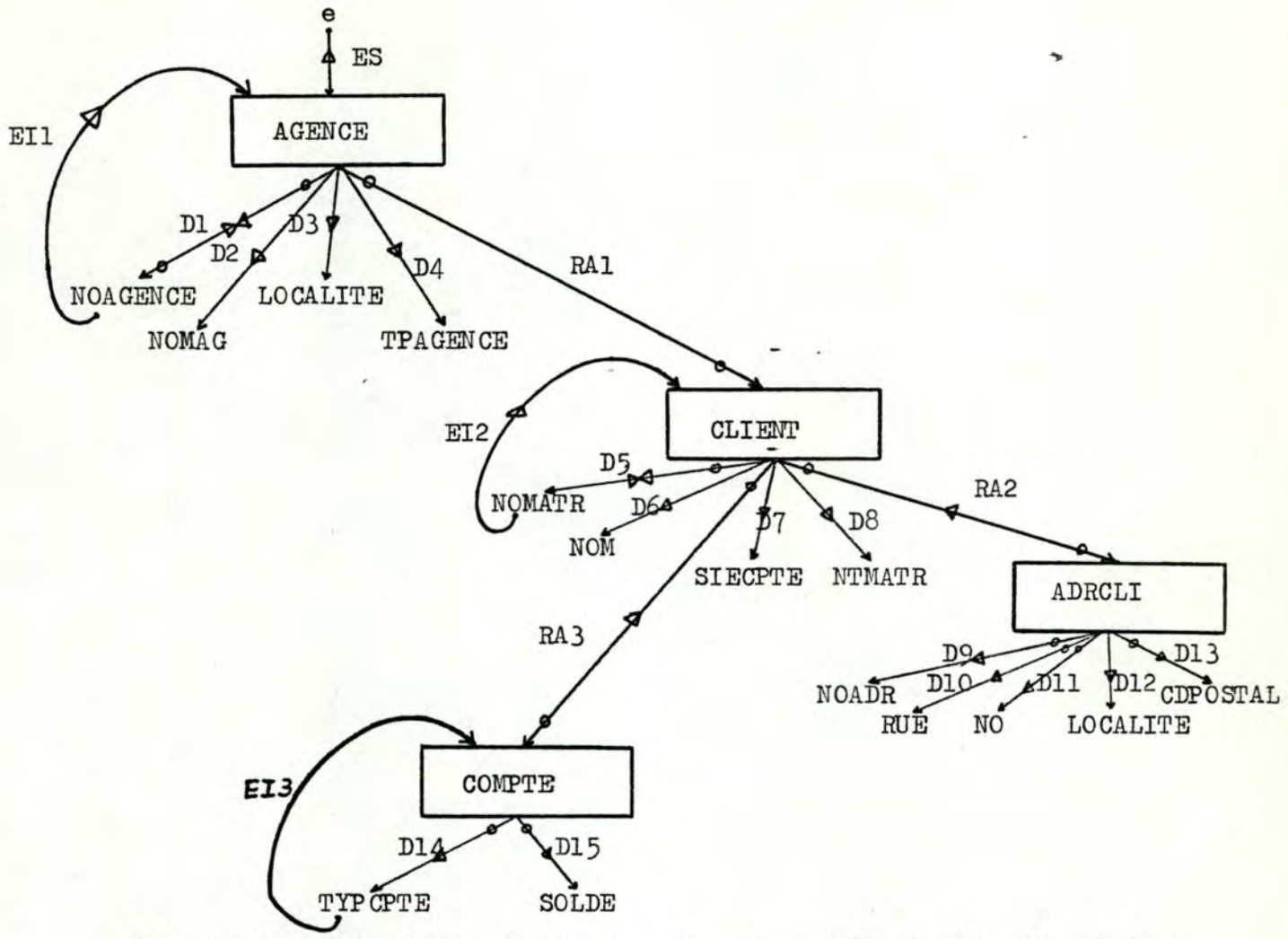
$1 \leq \underline{CARD}((NOMATR,NOADR)ADRCLI \leq 10.$

VII.3.2.2. Principaux concepts pour décrire une structure d'accès.

Remarque: Nous allons décrire les concepts d'une structure d'accès que nous pouvons qualifier de théorique (1). Nous verrons au chapitre VIII. comment il est possible de passer de cette structure à une structure IDS.

a) Exemple: Reprenons l'exemple du paragraphe VII.3.2.1. et construisons le graphe d'une structure d'accès avec ces éléments.

(1)cfr. ouvrage repris dans la bibliographie sous le numéro 2.



Une structure d'accès est d'une part, une représentation de relations n-aires à l'aide de relations binaires et, d'autre part, une suite de moyens de désignation des données, c'est-à-dire un ensemble de chemins d'accès à ces données. On trouvera en annexe un bref rappel au sujet des relations binaires et des symboles utilisés dans l'exemple ci-dessus.

b) Type d'article:

Soit la déclaration du type d'article **CLIENT**:

ARTICLE CLIENT

NOMATR	ENTIER(8)
NOM	CHARAC(10)
SIECPTE	ENTIER(2)
NTMATR	ENTIER(1)

Elle a pour but de déclarer l'existence d'un ensemble d'informations concernant le client; ces informations sont appelées types de données(ils déclarent l'existence d'un ensemble de données: NOMATR,NOM,...).

Deux articles distincts peuvent être constitués des mêmes données: pour les distinguer, on associe à chaque type d'article un type de donnée spécial, appelé type objet; le nom du type objet est en fait celui du type d'article auquel il est associé. Une déclaration d'un type d'article sera donc composée d'un ensemble d'objets, des ensembles des données et des fonctions (relations binaires) associant à un objet les différentes données(nous les noterons "D" pour "données").

D1: AGENCE ⟷ NOAGENCE

fonction biunivoque: à l'objet AGENCE correspond un numéro d'agence et inversément(D_1^{-1} est aussi une fonction).

D6: CLIENT ⟶ NOM

D6 est une fonction

$D6^{-1}$ est une application multivoque: il peut exister deux clients dont le nom soit Dupont.

c) Relations binaires entre types d'article.

Soient deux types d'article CLIENT et AGENCE; la relation RA1(AGENCE,CLIENT) traduit le fait que les clients entretiennent des rapports avec cette agence. A une agence seront donc reliés tous ses clients. Ces relations entre articles définissent les chemins d'accès: elles se traduisent par des relations binaires entre les types d'objet associés aux types d'article.

Exemple: -RA1(AGENCE,CLIENT): une agence a plusieurs clients et un client peut se rendre dans plusieurs agences ⇒ RA1 est une application multivoque.

-RA2(CLIENT,ADRCLI): un client a une ou plusieurs adresses et une adresse appartient à un seul client ⇒ RA2 est une application multivoque et $RA2^{-1}$ est une fonction.

d) Points d'entrée.

Ils déterminent les débuts des chemins d'accès auxquels peut faire appel l'utilisateur.

-Entrées par le système: c'est un élément e unique connu de l'utilisateur; on peut déclarer des relations entre e et n'importe quel type d'article. Les relations inverses

sont toujours des fonctions vu que e est unique. Nous les noterons ES pour "Entrées par le Système".

Exemple: ES(e,AGENCE)

-Entrées inversées: elles rendent possible l'accès à un type d'article à partir d'un type de donnée(ou de relations entre types de donnée). Nous les noterons EI pour "Entrées Inversées".

Exemple: accès aux clients à partir de leur numéro de matricule.

EI2(NOMATR,CLIENT)

VII.3.2.3. Définition du langage de manipulation des données sur une structure conceptuelle (1).

Nous distinguerons deux langages: le langage de désignation et le langage de commande. Nous n'en donnerons qu'une description très simplifiée.

a) Langage de désignation:

Lorsque l'utilisateur interroge la base de données, il désigne l'ensemble des informations dont il a besoin à l'aide d'un langage approprié, puis laisse au système le soin d'effectuer l'extraction. On désignera les informations sur la structure conceptuelle en créant de nouvelles relations au moyen de ce langage de désignation. Ce type de manipulation ne modifie pas le contenu de la base de données.

DESIGNER R=$\{$<expression de la logique des prédicats>$\}$
(R est le nom de la relation créée)

Exemple: Soit la requête suivante:"Nous désirons le solde du compte de type 'tc' du client dont le numéro de matricule est 'nm' ".

DESIGNER R=$\{(SOLDE.1)\,|\,(1 \in COMPTE) \wedge (NOMATR.1='nm') \wedge (TYPCPTE.1='tc')\}$
(1 étant une occurrence de la relation COMPTE)

b) Langage de commande:

Il a pour but de gérer les échanges entre la base de données et une zone de travail qui est propre à chaque utilisateur, et d'indiquer

(1) Pour plus de détails, cfr. ouvrage repris dans la bibliographie sous le numéro 3.

au système les modifications à apporter au contenu de la base de données. Lorsque l'utilisateur veut effectuer des traitements sur des données de la base de données, celles-ci doivent auparavant être transférées dans cette zone de travail(ZT).

1) Transferts dans la zone de travail.

On désigne d'abord les informations, ensuite on les transfère.

DESIGNER R={.....}

OBTENIR R (R créée dans ZT)

2) Obtention de n-uplets dans ZT.

DESIGNER R={.....} les n-uplets sont ordonnés suivant un ordre à priori non significatif

OBTENIR N DE R transfert du premier n-uplet de R dans la zone N de ZT

⋮

OBTENIR N DE R transferts des n-uplets suivants

3) Mise à jour des relations.

Le processus de mise à jour est le suivant:

1. Désignation d'une partie d'une relation de la base de données ou utilisation d'une désignation implicite.
2. Transfert du n-uplet à modifier dans la ZT.
3. Modification du n-uplet à l'aide du langage hôte.
4. Ordre de modification sur le n-uplet dans la relation physique(dans la base de données ou dans ZT).
   MODIFIER N DE nom-relation

4) Adjonction de n-uplets dans une relation.

Supposons que l'on ait une relation physique à n composantes; le problème est de transférer le contenu de n variables ou constantes de la ZT dans cette relation.

AJOUTER $N_1, N_2, \ldots, N_n$ A R

5) Supprimer des n-uplets d'une relation(physique).

Le processus est le suivant:

1. Désigner une partie d'une relation(ou utiliser une désignation implicite d'une relation).
2. Donner la commande de suppression.
   SUPPRIMER R
   ou SUPPRIMER $N_1, N_2, \ldots, N_n$ DE R

6) Suppression d'une désignation ou d'une relation.

La commande LIBERER nom-relation supprime la désignation de cette relation. Si "nom-relation" est une relation de la base de données, elle est supprimée.

VII.3.2.4. Définition des langages de manipulation des données sur une structure d'accès.

Comme pour le LMD sur une structure conceptuelle, nous distinguerons deux langages: le langage de désignation et le langage de commande. Pour ce qui est du premier, nous en donnerons deux versions différentes: un langage de désignation de haut niveau(type SOCRATE) et un langage de navigation(type IDS). A ce dernier, nous associerons un langage de commande qui sera décrit brièvement.

a) Langage de désignation de haut niveau.

Ce langage permet de désigner un ensemble d'articles d'un type donné répondant à des propriétés complexes portant sur les liens entre différents types d'article ainsi que sur les données qui leur sont attachées. Il n'est pas dans notre intention de décrire complètement ce langage dans ce paragraphe: une description approfondie en est donnée dans l'ouvrage repris sous le numéro 3 dans la bibliographie. Nous croyons que nous pouvons le résumer en ces termes: il traduit, en fonction

des accès à réaliser sur la structure d'accès, ce qui est exprimé dans la logique des prédicats dans une désignation sur la structure relationnelle.

Exemple:- Soit la requête suivante exprimée sur l'exemple du paragraphe VII.3.2.1.
"Nous voulons les types de compte de tous les comptes des clients dont le code série de compte est 'csc' et qui habitent à la localité 'local'".

-Elle peut s'écrire:

DESIGNER R={(TYPCPTE.1)|((1∈COMPTE)∧∃v((v∈CLIENT)∧(NOMATR.1=NOMATR.v)∧(SIECPTE.v='csc')∧∃m((m∈ADRCLI)∧(NOMATR.m=NOMATR.v)∧(LOCALITE.m='local'))))}

Exprimons cette requête sur la structure d'accès de l'exemple du paragraphe VII.3.2.2. à l'aide du langage de désignation de haut niveau:

```
POUR TOUT AGENCE DE e VIA ES
        POUR TOUT CLIENT DE AGENCE VIA RA1
                TQ( ∃ SIECPTE DE CLIENT VIA D7 TQ SIECPTE='csc')
                  ∧(∃ ADRCLI DE CLIENT VIA RA2 TQ LOCALITE='local')
             POUR TOUT COMPTE DE CLIENT VIA RA3
                  Début
                    SELECTIONNER (TYPCPTE DE COMPTE VIA D14)
                  Fin
```

Nous verrons ultérieurement comment il est possible de passer de la première expression à la seconde(1).

b) Langage de navigation.

Il ne permet de désigner un article qu'en utilisant un seul accès élémentaire. A chaque type d'article est associé une zone standard unique pouvant contenir l'objet associé à ce type d'article; on l'appelle le "courant" du type d'article; il contient toujours l'objet associé au dernier article désigné de ce type. Comme dans le langage de haut niveau, il est possible d'utiliser des désignateurs qui permettent de donner des noms aux différents articles désignés. Ce langage désigne les articles mais, à l'encontre du précédent, ne les sélectionne pas en fonction des différentes propriétés énoncées. Celles-ci devront être réalisées à l'aide du langage hôte.

(1) cfr. chapitre VII.3.4.1.

La syntaxe de cette désignation peut s'écrire:

DESIGNER[dés ign-1=]article-désigné[SUIVANT[dés ign-2[=article-suivi]]]

DE{dés ign-3[=article-source] / article-source / dom-entrée e / donnée≡'constante'} [VIA nom-relation][PAR nom-ordre]

Exemple: DESIGNER cli1=CLIENT SUIVANT cli2=CLIENT
DE ag1=AGENCE VIA RA1

c) Langage de commande. -

Chaque programme utilisateur dispose d'une zone de travail qui contient les désignateurs des enregistrements logiques correspondant à chaque type d'article de la base de données et portant les mêmes noms. Le SGBD tiendra également à jour les différentes zones "courant".

1) Transfert d'un article dans ZT.

OBTENIR {article / désign [=article]}

Si le'courant' de l'article contient un objet, alors la commande transfère cet article en ZT.(idem pour le désignateur)

Exemple: OBTENIR cli1=CLIENT

2) Supprimer un article.

SUPPRIMER {article / désign [= article]}

Cette commande provoque la suppression de l'article désigné par le 'courant' de celui-ci ou par le désignateur.

3) Modifier un article.

MODIFIER {article / désign [=article]}

L'article désigné dans la commande est remplacé par le contenu de la zone de travail correspondante.

4) Adjonction d'un article.

AJOUTER {article / désign-1 [=article]} || LIE A {article / désign-2[=article]} [VIA nom-relation] ||

Le principe est le suivant: construire dans la zone de travail l'article à ajouter et donner l'ordre d'adjonction.

Exemple:AJOUTER cli-1=CLIENT LIE A ag-1=AGENCE VIA RA1.

VII.3.3. Passage de la description d'une structure conceptuelle à la description d'une structure d'accès.

Pour chaque concept utilisé dans la structure conceptuelle, nous donnerons son équivalent dans la structure d'accès. Pour effectuer ce parallèle, nous nous baserons sur la structure conceptuelle dont nous avons le graphe au chapitre IV.1.2.(elle est décrite plus en détails en annexe II.). Nous nous sommes choisi une structure d'accès qui couvre une partie de la structure conceptuelle: nous nous sommes contentés d'une description graphique.(Voir page92)

Remarques: Nous adopterons les conventions d'écriture suivantes: -en majuscules, nous aurons les relations de la structure conceptuelle, les types de données et les types d'articles.

-en minuscules, nous aurons les occurrences des relations (ou variables relationnelles), les occurrences des articles (c'est-à-dire les objets qui leur sont associés), les valeurs.

-NOMATR.l où NOMATR = numéro de matricule du client
l=variable relationnelle

Cette expression signifie que l'on prend le numéro de matricule de l'occurrence 1 de la relation CLIENT.

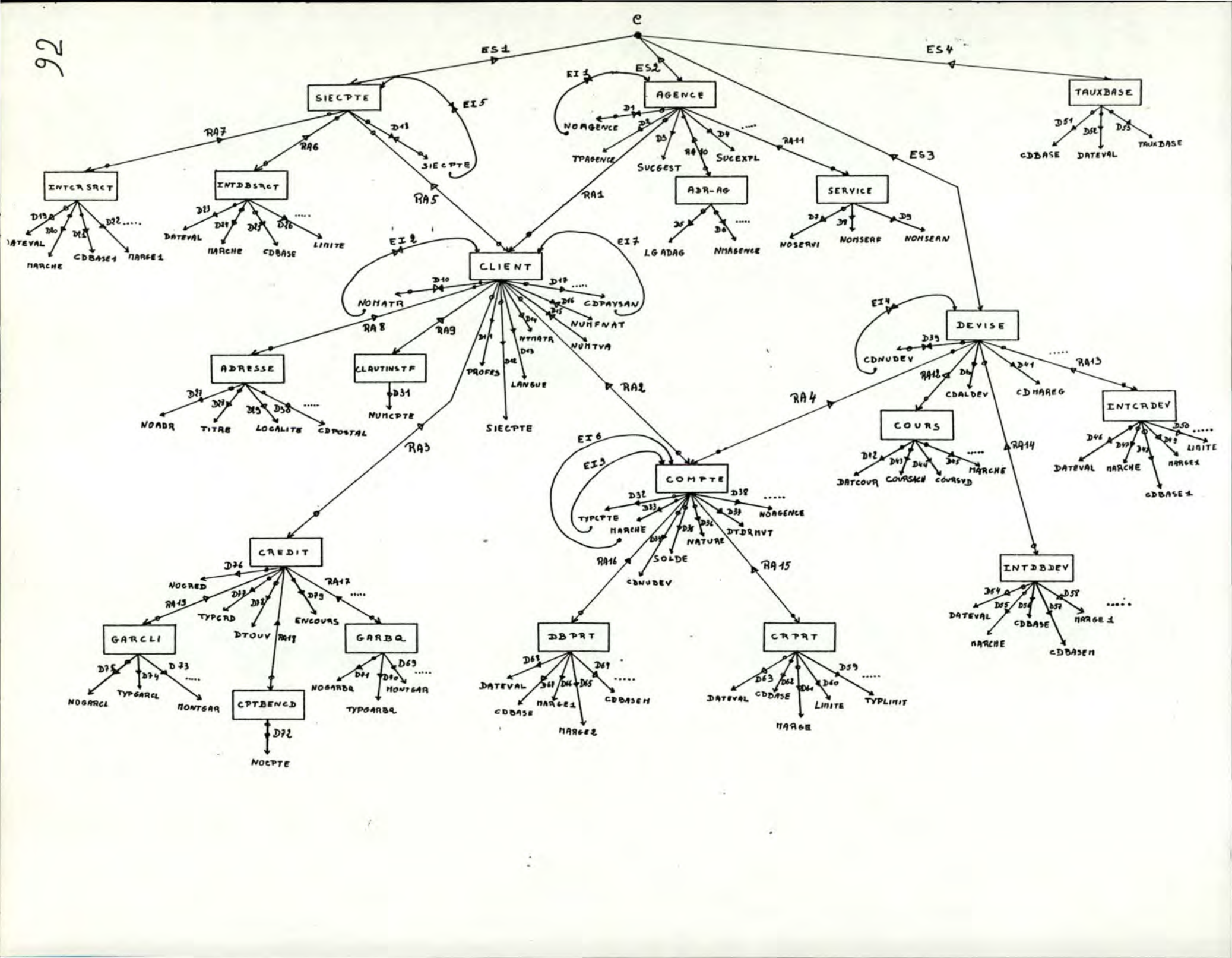
VII.3.3.1. Composantes d'une relation.

Elles coïncident avec les types de données de la structure d'accès(cfr. chapitre VII.3.2.2. b)

Exemple: NOMATR, NOAGENCE,...

VII.3.3.2. Relations.

Une relation est un ensemble de n-uples. En développant quelques exemples, nous montrons ci-dessous comment nous pouvons trouver l'équivalent d'un n-uple dans la structure d'accès. Plusieurs combinaisons de chemins d'accès sont possibles pour effectuer la correspondance entre les données organisées de manière différente dans les deux structures. Il est important de les signaler toutes: nous ferons appel à l'une ou


C
ES1
ES2
ES3
ES4
SIECPTE
AGENCE
TAUXBASE
CLIENT
DEVISE
COMPTE
CREDIT
INTCRSRCT
INTDBSRCT
ADR-AG
SERVICE
ADRESSE
CLAUTINSTF
COURS
INTCRDEV
INTDBDEV
GARCLI
GARBQ
CPTBENCD
DBPRT
CRPRT
RA1
RA2
RA3
RA4
RA5
RA6
RA7
RA8
RA9
RA10
RA11
RA12
RA13
RA14
RA15
RA16
RA17
EI1
EI2
EI3
EI4
EI5
EI6
EI7
NOAGENCE
TPAGENCE
SUCGEST
SUCEXPL
LGADAG
NMAGENCE
NOSERVI
NOMSERF
NOMSERN
NOMATR
CDPAYSAN
NUMFNAT
NUMTVA
NTMATR
PROFES
LANGUE
SIECPTE
NOADR
TITRE
LOCALITE
CDPOSTAL
NUMCPTE
CDNUDEV
CDALDEV
CDMAREG
DATCOUR
COURSACH
COURSVD
MARCHE
LIMITE
DATEVAL
CDBASE1
MARGE1
TYPCPTE
NATURE
DTDRNVT
SOLDE
NOCRED
TYPCRD
DTOUV
ENCOURS
NOGARCL
TYPGARCL
MONTGAR
NOGARBQ
TYPGARBQ
NOCPTE
CDBASE
MARGE2
CDBASEM
TYPLIMIT
CDBASE2

l'autre suivant son aptitude à résoudre le problème qui se posera.

Exemples:

Soient les types de données: NOAGENCE:numéro de l'agence
TPAGENCE:type de l'agence
SUCEXPL:succursale d'exploitation
NOMATR:numéro de matricule
SIECPTE:code série de compte
TYPCPTE:type de compte
SOLDE:solde du compte

Soient les relations: AGENCE(NOAGENCE,TPAGENCE,...,SUCEXPL)
CLIENT(NOMATR,SIECPTE,...)
COMPTE(NOMATR,NOAGENCE,TYPCPTE,SOLDE,...)

Soient l une occurrence de la relation AGENCE
m une occurrence de la relation CLIENT
n une occurrence de la relation COMPTE
ag une occurrence du type d'article AGENCE
cli une occurrence du type d'article CLIENT
scpte une occurrence du type d'article SIECPTE
cpte une occurrence du type d'article COMPTE ;

nous pouvons écrire:

a) $(l\ \text{AGENCE}) \Leftrightarrow \exists ag((e,ag)\in ES2 \wedge (ag,NOAGENCE.1)\in D1 \wedge (ag,TPAGENCE.1)\in D2 \wedge (ag,SUCEXPL.1)\in D4)$.

Cette expression se lit: il existe ag de l'article AGENCE telle que:-elle soit liée au point d'entée du système par la relation ES2,
-le numéro de l'agence de la relation AGENCE lui soit lié par la relation D1,
⋮

b) $(l\in \text{CLIENT}) \Leftrightarrow \exists ag((e,ag)\in ES2 \wedge \exists cli((ag,cli)\in RA1 \wedge (cli,NOMATR.1)\in D10 \wedge (cli,SIECPTE.1)\in D12 \wedge .....)$

$(l\in \text{CLIENT}) \Leftrightarrow \exists cli((NOMATR.1,cli)\in EI2 \wedge .....)$

$(l\in \text{CLIENT}) \Leftrightarrow \exists scpte((SIECPTE.1,scpte)\in EI5 \wedge \exists cli((scpte,cli)\in RA5 \wedge .....)$
⋮

c) $(l\in \text{COMPTE}) \Leftrightarrow \exists ag((e,ag)\in ES2 \wedge (ag,NOAGENCE.1\in D1 \wedge \exists cli((ag,cli)\in RA1 \wedge (cli,NOMATR.1)\in D10 \wedge \exists cpte((cli,cpte)\in RA2 \wedge (cpte,TYPCPTE.1)\in D32 \wedge (cpte,SOLDE.1)\in D35 \wedge .....)$

VII.3.3.3. Clé ou index d'une relation.

Développons quelques exemples:

a) Dans la structure conceptuelle, NOAGENCE est la clé de la relation AGENCE = étant donné une valeur de NOAGENCE, elle permet d'identifier le n-uple qui la contient.

Dans la structure d'accès, D1 est une fonction biunivoque de l'article AGENCE sur la donnée NOAGENCE(=à une occurrence ag de AGENCE correspond un seul numéro d'agence et inversément) et toutes les autres relations de description liant AGENCE à ses caractéristiques sont des fonctions.

b) NOMATR est la clé de la relation CLIENT.

D10 est une fonction biunivoque et toutes les autres relations(D11,D12,..) sont des fonctions.

c) (NOAGENCE,NOSERVI) clé de la relation SERVICE, n'est pas représentée dans la structure d'accès.

d) (NOMATR,NOAGENCE,TYPCPTE) clé de la relation COMPTE, n'est pas représentée dans la structure d'accès.

VII.3.3.4. Relations fonctionnelles.

a) A un client est associé un et un seul code série de compte.

Dans la structure d'accès: $RA5^{-1}$(CLIENT,SIECPTE) est une fonction et, D10 et D18 sont des fonctions biunivoques.

b) Un compte n'est exprimé que dans une et une seule devise.

$RA4^{-1}$(COMPTE,DEVISE) est une fonction et D39(DEVISE,CDNUDEV) est une fonction biunivoque.

c) A une localité ne correspond qu'un seul code postal: cette propriété n'est pas représentée dans la structure d'accès; une procédure devra tester s'il n'existe pas une localité possédant deux numéros postaux différents.

VII.3.3.5. Projections.

Les relations de type entrée par les domaines de définition permettent de définir les représentations des projections des relations n-aires sur des composantes et (ou) sur des valeurs.

a) EI1(NOAGENCE,AGENCE): étant donné un numéro d'agence, on retrouve l'enregistrement contenant les données relatives à cette agence.

b) EI2(NOMATR,CLIENT): idem pour un client.

c) EI3(TYPCPTE,COMPTE): nous pouvons accéder à tous les comptes d'un même type; EI3 est une représentation de (TYPCPTE)COMPTE de la structure conceptuelle.

VII.3.3.6. Egalité.

a) (NOAGENCE)AGENCE = (NOAGENCE)ADR-AG: toute agence a au moins une adresse. Cette propriété est représentée par le fait que la relation RA10(AGENCE,ADR-AG) est totale.

b) (NOMATR)ADRCLI = (NOMATR)CLIENT: RA8(CLIENT,ADRESSE) est totale.

c) (NOMATR)COMPTE = (NOMATR)CLIENT: RA2(CLIENT,COMPTE) est totale.

VII.3.3.7. Inclusion.

a) (NOAGENCE)AGENCE $\subseteq$ (NOAGENCE)SERVICE: toutes les agences ne sont pas subdivisées en services.

RA11(AGENCE,SERVICE) est une relation partielle.

b) (NOMATR,NOAGENCE,TYPCPTE)DBPRT $\subseteq$ (NOMATR,NOAGENCE,TYPCPTE)COMPTE: tous les comptes ne jouissent pas de conditions particulières pour le calcul des intérêts.

RA16(COMPTE,DBPRT) est une relation partielle.

Il n'en est pas de même pour les conditions pour le calcul des intérêts correspondant au code série de compte et à la devise; dans ces deux cas, elles doivent exister pour tout code série de compte et pour toute devise.

VII.3.3.8. Cardinalité.

A toute relation de la structure d'accès, nous associons quatre nombres: R(A,B) i,j-k,l tq $i \geq 0$, $k \geq 0$, $j \geq i$, $l \geq k$

$$\Leftrightarrow \forall\, a \in A : i \leq CARD(R(a)) \leq k$$
$$\forall\, b \in B : j \leq CARD(R^{-1}(b)) \leq l.$$

Examinons deux cas pour nous familiariser avec cette expression: a) une agence a au plus 50 services.

Dans la structure conceptuelle, nous avions l'expression CARD-PROJEC (NOAGENCE,NOSERVI) 0,50: étant donné un NOAGENCE, on aura au plus 50 numéros de service(NOSERVI).

Dans la structure d'accès, nous aurons RA11(AGENCE,SERVICE) 0,50-1,1: pour un article AGENCE, il y aura au plus 50 articles SERVICE et un service décrit n'appartient qu'à une seule agence.

b) une agence a au moins une adresse, au plus deux.

Nous aurons: CARD-PROJEC(NOAGENCE,LGADAG) 1,2
RA10(AGENCE,ADR-AG) 1,2-1,1.

VII.3.3.9. Autres propriétés: les prédicats.

Il existe des limites au mapping automatique. Certaines contraintes d'intégrité exprimées sur les relations et leurs domaines ne sont pas transformées immédiatement en contrôles sur les articles et leurs données.

Exemple: Contrôle sur la valeur prise par une composante d'une relation: appartient-elle à l'ensemble des valeurs possibles(clause VALUE)? Dans la structure d'accès, il sera nécessaire d'écrire une procédure vérifiant l'appartenance de cette valeur à l'ensemble des valeurs possibles.

VII.3.4. Passage du LMD sur une structure conceptuelle au LMD sur une structure d'accès.(1)

Dans ce paragraphe, nous tenterons d'appliquer les règles, les principes énoncés dans le cours de Monsieur Cabanes.(2)

VII.3.4.1. Passage d'une désignation sur la structure conceptuelle à une désignation sur la structure d'accès.

a) Définition de l'A-requête(requête sur une structure d'accès)

Dans une structure d'accès, nous désignerons une information par son appartenance à l'image d'une valeur donnée par une application spécifiée et par les propriétés qu'elle possède. Celles-ci s'expriment par des tests et par les chemins suivis pour accéder à la donnée. Le chemin d'accès à une information est caractérisé par la suite des données nécessaires à sa définition. Soit l'expression qui nous donne tous les numéros des agences:

DESIGNER $R=\{(noagence) \mid \exists agence((e,agence)\in ES2 \wedge (agence,noagence)\in D1)\}$.

Cette désignation s'effectue en donnant le nom de la donnée et le chemin pour y accéder.

---

(1) LMD = langage de manipulation des données.

(2) cfr. ouvrage repris dans la bibliographie sous le numéro 3.

Soit cette nouvelle expression:

POUR TOUT AGENCE DE e VIA ES2

Début

SELECTIONNER (NOAGENCE DE AGENCE VIA D1)

Fin.

Nous l'appellerons A-requête, requête sur la structure d'accès. Nous faisons l'bypothèse que les deux expressions ci-dessus sont équivalentes.

b) Première approche: langage de désignation de haut niveau.

Dans ce paragraphe, nous tâcherons de déterminer certaines règles qui permettent une transformation progressive d'une requête sur une structure relationnelle en une A-requête. Au paragraphe suivant, nous essayerons d'optimiser cette transformation en fonction du nombre d'accès à réaliser sur la structure d'accès.

Pour les deux exemples qui vont suivre, nous aurons besoin:

-de la relation TAUXBASE dont les domaines sont TAUXBASE = taux de base
CDBASE = code base
DATEVAL = date de validité;

-de la relation INTDBSRCT: conditions standards par série de compte pour le calcul des intérêts. Les domaines sont CDBASE, DATEVAL, MARCHE,...

-d'une occurrence l de la relation TAUXBASE;

-d'une occurrence s de la relation INTDBSRCT;

-d'une occurrence txbs de l'article TAUXBASE;

-d'une occurrence siecpte de l'article SIECPTE;

-d'une occurrence intdbct de l'article INTDBSRCT;

-de tauxbase = TAUXBASE.l
de cdbase = CDBASE.l
de dateval = DATEVAL.l
de cdbs = CDBASE.s
de dtval = DATEVAL.s

Soit l'énoncé de la requête suivante: "Nous désirons le taux de base(TAUXBASE) correspondant au code base 'cdbs'(CDBASE) et dont la date de validité est 'dtval'(DATEVAL)". Elle peut s'écrire:

DESIGNER $R=\{(TAUXBASE.l) \mid ((l \in TAUXBASE) \wedge (CDBASE.l='cdbs') \wedge (DATEVAL.l='dtval'))\}$

La notion de variable relationnelle (l) n'existe pas dans la structure d'accès. Dans un premier temps, nous la remplacerons par l'occurrence toute entière, par le n-uple.

Règle 1: Supprimer les variables relationnelles et les remplacer par des n-uples; pour cela, introduire à l'aide de quantificateurs existentiels les données appartenant aux domaines(des relations) n'apparaissant pas dans l'expression.

Ce qui nous donne:

DESIGNER $R=\{(\text{tauxbase}) \mid \exists\, \text{cdbase}(\exists\, \text{dateval}((\text{cdbase}, \text{dateval}, \text{tauxbase}) \in \text{TAUXBASE} \wedge (\text{cdbase}=\text{'cdbs'}) \wedge (\text{dateval}=\text{'dtval'})))\}$.

Règle 2: A l'aide des règles d'équivalence définies au chapitre VII.3.3.2., remplacer les n-uples par leurs équivalents dans la structure d'accès.

Nous aurons alors:

DESIGNER $R=\{(\text{tauxbase}) \mid \exists\, \text{cdbase}(\exists\, \text{dateval}(\exists\, \text{txbs}((e,\text{txbs}) \in \text{ES4} \wedge (\text{txbs},\text{cdbase}) \in \text{D51} \wedge (\text{txbs},\text{dateval}) \in \text{D52} \wedge (\text{txbs},\text{tauxbase}) \in \text{D53}) \wedge (\text{cdbase}=\text{'cdbs'}) \wedge (\text{dateval}=\text{'dtval'})))\}$.

Règle 3: Permuter l'ordre des quantificateurs existentiels de manière à ce que leur portée se limite uniquement à la variable qui leur est associée.

Nous aboutissons à:

DESIGNER $R=\{(\text{tauxbase}) \mid \exists\, \text{txbs}((e,\text{txbs}) \in \text{ES4} \wedge (\text{txbs},\text{tauxbase}) \in \text{D53} \wedge \exists\, \text{cdbase}((\text{txbs},\text{cdbase}) \in \text{D51} \wedge (\text{cdbase}=\text{'cdbs'}) \wedge \exists\, \text{dateval}((\text{txbs},\text{dateval}) \in \text{D52} \wedge (\text{dateval}=\text{'dtval'})))\}$.

Règle 4: Passage à l'écriture de l'A-requête grâce à l'bypothèse émise au chapitre VII.3.4.1 a).

Nous obtenons:

```
POUR TOUT TXBS DE e VIA ES4
        TQ (∃ CDBASE DE TXBS VIA D51  TQ CDBASE = 'cdbs')
           ∧(∃ DATEVAL DE TXBS VIA D52  TQ DATEVAL = 'dtval')
    Début
      SELECTIONNER (TAUXBASE DE TXBS VIA D53)
    Fin.
```

Remarque: Les relations D51 et D52 étant totales, il est inutile de spécifier de nouveau l'existence de CDBASE et de DATEVAL.

Nous aurons donc finalement:

```
POUR TOUT TXBS DE e VIA ES4  TQ ((CDBASE = 'cdbs')∧(DATEVAL = 'dtval'))
     Début
       SELECTIONNER (TAUXBASE DE TXBS VIA D53)
     Fin.
```

Prenons un second exemple. Soit la requête suivante:"Rechercher tous les taux de base correspondant aux conditions standards, par série de compte, pour le calcul des intérêts débiteurs sur les comptes en francs belges".

Remarque: Nous supposons que toutes les relations de description utilisées dans cet exemple sont totales.

La requête s'écrit:

DESIGNER $R=\{(TAUXBASE.1) \mid ((1 \in TAUXBASE) \wedge \exists s((s \in INTDBSRCT) \wedge (CDBASE.1=CDBASE.s) \wedge (DATEVAL.1=DATEVAL.s)))\}$.

Appliquons la règle 1:

DESIGNER $R=\{(tauxbase) \mid \exists cdbase(\exists dateval((cdbase,dateval,tauxbase) \in TAUXBASE \wedge \exists marché(\exists codbs(\exists dtval(...((dtval,codbs,marché,...) \in INTDBSRCT \wedge (cdbase=codbs) \wedge (dateval=dtval))...)\}$.

Appliquons la règle 2:

DESIGNER $R=\{(tauxbase) \mid \exists cdbase(\exists dateval(\exists txbs((e,txbs) \in ES4 \wedge (txbs,cdbase) \in D51 \wedge ... \wedge (txbs,tauxbase) \in D53 \wedge \exists marché(\exists ...(\exists siecpte((e,siecpte) \in ES1 \wedge \exists intdbct((siecpte,intdbct) \in RA6 \wedge ... \wedge (cdbase=codbs) \wedge (dateval=dtval))...)\}$.

Appliquons la règle 3:

DESIGNER $R=\{(tauxbase) \mid \exists siecpte((e,siecpte) \in ES1 \wedge \exists intdbct((siecpte,intdbct) \in RA6 \wedge \exists txbs((e,txbs) \in ES4 \wedge (txbs,tauxbase) \in D53 \wedge \exists cdbase((txbs,cdbase) \in D51 \wedge \exists dateval((txbs,dateval) \in D52 \wedge \exists codbs((intdbct,codbs) \in D25 \wedge \exists dtval((intdbct,dtval) \in D23 \wedge (cdbase=codbs) \wedge (dateval=dtval))))..)\}$

Appliquons la règle 4:

```
POUR TOUT SIECPTE DE e VIA ES1
     Début POUR TOUT INTDBSRCT DE SIECPTE VIA RA6
          Début POUR TOUT TXBS DE e VIA ES4
                     TQ ((CDBASE DE TXBS = CDBASE DE INTDBSRCT)
                         (DATEVAL DE TXBS = DATEVAL DE INTDBSRCT))
               Début
                 SELECTIONNER (TAUXBASE DE TXBS VIA D53)
               Fin
          Fin
     Fin.
```

c) Seconde approche: tentative d'optimisation (langage de navigation).

Soit la requête suivante: "Rechercher les soldes des comptes de type 'tc' détenus par les clients dont le code série de compte est 'sc'".

Nous pouvons écrire:

DESIGNER R={(SOLDE.l)|((l∈COMPTE)∧(TYPCPTE.l='tc')∧∃s((s∈CLIENT)
∧(NOMATR.l=NOMATR.s)∧(SIECPTE.s='sc')))}.

OBTENIR SLD-1 DE R.
OBTENIR SLD-2 DE R. (SLD-1,SLD-2,... appartiennent à la zone de travail)
.
.
.

Nous avons déjà vu que la requête peut être transformée directement en une A-requête:

```
POUR TOUT AGENCE DE e VIA ES2
    Début POUR TOUT CLIENT DE AGENCE VIA RA1
              TQ SIECPTE DE CLIENT VIA D12 ='sc'
          Début POUR TOUT COMPTE DE CLIENT VIA RA2
                    TQ TYCPTE DE COMPTE VIA D32 ='tc'
                Début
                  SELECTIONNER (SOLDE DE COMPTE VIA D35)
                Fin
          Fin
    Fin.
```

Cette forme théorique que nous obtenons par transformation directe doit être réécrite dans un langage de navigation. Nous devrons tenir compte d'un élément très important: éviter les balayages inutiles. La requête désigne les soldes des comptes de type 'tc' détenus par des clients de code série de compte 'sc'. Que faisons-nous pour obtenir le premier solde? Nous prenons tout d'abord la première agence, ensuite le premier client de cette agence qui vérifie la condition, et enfin le premier compte de type 'tc' de ce client. Lorsque nous voudrons le second solde, nous ne recommencerons pas le processus depuis le début, mais nous repartirons là où nous nous étions arrêtés: nous prendrons le deuxième compte de même type du client sélectionné; s'il n'en a pas, nous prendrons le client suivant,... et puis l'agence suivante.

Remarques: 1)Nous allons distinguer le premier passage dans la structure d'accès des suivants au moyen d'une variable PREM: PREM=V(vrai) signifie que nous sommes dans le premier passage.

2) Les deux tests à effectuer feront partie des procédures CONDITION-1 et CONDITION-2 qui ne seront pas décrites ici:

CONDITION-1: Le code série de compte du client est 'sc'
CONDITION-2: Le type du compte est 'tc'.

Nous pouvons écrire le programme suivant:

```
        SI PREM=V ALORS PREM=F
                        DESIGNER X1=AGENCE DE e
                        DESIGNER X2=CLIENT DE X1
                        PERFORM CONDITION-1
                        SI C1=V ALORS ALLER A PRCPTE
                                SINON ALLER A CLISV
                    SINON ALLER A CPTESV
PRCPTE: DESIGNER X3=COMPTE DE X2
 COND2: PERFORM CONDITION-2
        OBTENIR X3
        SI C2=V ALORS ALLER A SORTIE
CPTESV: DESIGNER X3 SUIVANT X3 DE X2
        SI X3=0 ALORS ALLER A COND2
 CLISV: DESIGNER X2 SUIVANT X2 DE X1
        SI X2=0 ALORS PERFORM CONDITION-1
                      SI C1=V ALORS ALLER A PRCPTE
                              SINON ALLER A CLISV
  AGSV: DESIGNER X1 SUIVANT X1 DE e
        SI X1=0 ALORS DESIGNER X2 DE X1
                      PERFORM CONDITION-1
                      SI C1=V ALORS ALLER A PRCPTE
                              SINON ALLER A CLISV
        .
SORTIE: SOLDE := SOLDE.X3
        .
   FIN:
```

De ce premier programme, nous voyons apparaître que:

1) Chaque étape de l'A-requête(caractérisée par un POUR TOUT) se voit attribuer un élément désignateur qui lui est propre(Xi).
2) Une variable par niveau est nécessaire pour tester la condition qui peut éventuellement apparaître(Ci).
3) Il est utile de placer les OBTENIR uniquement lorsque les éléments concernés apparaissent dans le SELECTIONNER ou quand ils sont utiles pour effectuer certains tests.

De cette brève analyse nous pouvons déduire un canevas général d'une A-requête en un programme s'exécutant sur la structure d'accès.

Canevas général:

```
        SI PREM=V ALORS PREM=F
                        ALLER A PRAG
                  SINON ALLER A CPTESV
 PRAG:  DESIGNER X1=AGENCE DE e
COND1:  PERFORM CONDITION-1
        OBTENIR X1
        SI C1=F ALORS ALLER A AGSV
PRCLI:  DESIGNER X2=CLIENT DE X1
COND2:  PERFORM CONDITION-2
        OBTENIR X2
        SI C2=F ALORS ALLER A CLISV
PRCPTE: DESIGNER X3=COMPTE DE X1
COND3:  PERFORM CONDITION-3
        OBTENIR X3
        SI C3=F ALORS ALLER A CPTESV
                SINON ALLER A SORTIE
CPTESV: DESIGNER X3 SUIVANT X3 DE X2
        SI X3=Ø ALORS ALLER A COND3
 CLISV: DESIGNER X2 SUIVANT X2 DE X1
        SI X2=Ø ALORS ALLER A COND2
  AGSV: DESIGNER X1 SUIVANT X1 DE e
        SI X1=Ø ALORS ALLER A COND1
   FIN: RETURN
SORTIE: SOLDE := SOLDE.X3
        RETURN
        END
```

Commentaires:

1) Si c'est le premier passage, nous débutons le processus en recherchant la première agence. Dans le cas contraire, nous continuons le processus en cours en recherchant le compte suivant d'un client.
2) Il suffit de regarder le canevas général pour se persuader qu'il est possible de générer ce programme en parcourant l'A-requête. Il est composé de trois parties: -un test pour savoir si c'est le premier passage;
   -si oui, alors descendre dans la structure d'accès;
   -si non, prendre l'élément suivant et s'il n'est pas valable, remonter dans la SA.

3) Nous pouvons nous poser la question de savoir s'il est possible de générer ce programme en une seule lecture de l'A-requête. En effet, il faut: -repérer tous les niveaux (Xi);
-repérer les niveaux où interviennent des conditions à tester (Ci);
-générer les OBTENIR là où ils sont indispensables.

4) Ce canevas général est bien entendu sujet à des optimisations qui portent sur le fait que: -certains domaines sont reliés entre eux par des fonctions.

ex: si un client ne peut détenir qu'un seul compte d'un type donné, alorstout ce qui concerne "le compte suivant(CPTESV)" n'existe pas. Une fois que l'on a obtenu le compte d'un client, on passe au suivant.

-à certains niveaux, il n'existe pas de conditions à tester. (ex: AGENCE)

d) Remarque.

Nous n'avons pas présenté d'exemples dans lesquels interviennent des quantificateurs universels($\forall$). Pour pouvoir utiliser les équivalences entre les descriptions, le plus simple est de prendre l'opposé de la requête sur la structure relationnelle. En effet, l'opposé de $\forall X\ (A \Rightarrow B)$ est $\exists X\ (A \wedge \neg B)$. Après avoir transformé cette dernière, il suffit de repasser à l'inverse.

VII.3.4.2. Passage de l'adjonction de n-uples à des relations, à l'adjonction de nouveaux articles.

Soit une expression du genre:

AJOUTER N A R où N=n-uple
R=relation.

Suivant les règles d'équivalence définies au chapitre VII.3.3.2 nous pouvons faire la correspondance entre d'une part, la relation et ses composantes et, d'autre part, les articles, les données et les relations d'accès. De ce passage, nous pouvons constater qu'il existe trois cas à analyser: 1) A un n-uple de la relation correspond un article auquel sont liés tous les domaines de définition de la relation.

L'adjonction d'un n-uple à la relation engendrera un et un seul ordre d'adjonction d'un nouvel article.

2) A un n-uple de la relation correspond une partie des données liées à un article. L'ordre d'adjonction du n-uple à la relation engendrera un ordre d'adjonction d'un nouvel article, les données concernées auront la valeur inconnue.

3) A un n-uple de la relation correspondent plusieurs articles de la structure d'accès. L'ordre d'adjonction du n-uple engendrera plusieurs ordres d'adjonction de plusieurs articles.

Il est important également de mémoriser les liens entre les différents articles cités lors du passage d'une structure à l'autre, en vue de les réaliser lors de l'adjonction des articles(LIE A...).

Nous allons analyser un cas du type 1); soit le problème suivant: il s'agit d'ajouter un n-uple à la relation CLIENT, le client étant lié à l'agence portant le numéro 'agc'.

Nous pouvons écrire le programme suivant sur la structure conceptuelle:

DESIGNER R=$\{(1)\mid((1\in$AGENCE$)\wedge$(NOAGENCE.1='agc')) .

OBTENIR R

NOMATR.CLI='nm'
PROFES.CLI='pf'
SIECPTE.CLI='sc'
LANGUE.CLI='lg'
:
:

(garnissage de la zone CLI de la zone de travail de la structure conceptuelle)

AJOUTER CLI A CLIENT

A chaque type d'article correspond dans la zone de travail de la structure d'accès une zone destinée à recevoir le contenu des articles. Nous aurons donc les zones AGC(agence) et CLT(client). Grâce à l'équivalence ci-dessous, nous pouvons garnir ces zones:

(1 CLIENT)$\Leftrightarrow\exists$ag((e,ag)$\in$ES2$\wedge\exists$cli((ag,cli)$\in$RA1$\wedge$(cli,NOMATR.1)$\in$D10$\wedge$...))

Nous aurons donc:

AGC	agc	tpag			
	NOAGENCE	TPAGENCE			
CLT	nm	pf	sc		
	NOMATR	PROFES	SIECPTE		

L'équivalence entre les deux structures nous montre que l'adjonction d'un nouvel article CLIENT entraîne sa liaison avec un article AGENCE. Nous devons donc rechercher cette agence et lui lier l'article CLIENT. Cela nous donnera le programme suivant:

DESIGNER AGC=AGENCE DE NOAGENCE='ag' VIA EI1
OBTENIR AGC
NOMATR.CLT='nm'
PROFES.CLT='pf'
SIECPTE.CLT='sc'
LANGUE.CLT='lg'
:
:
AJOUTER CLT=CLIENT LIE A AGC=AGENCE VIA RA1

Remarque: Nous avons signalé au paragraphe VII.3.3.2. qu'il était important de mentionner toutes les équivalences entre les n-uples d'une relation et les articles de la structure d'accès. Pour les clients, nous remarquons dans le schéma de la page 92 que les articles CLIENT sont obligatoirement liés à un article SIECPTE($RA5^{-1}$ est totale). Donc, pour être plus complets, nous devrions ajouter ce qui suit à l'ordre d'adjonction:

LIE A SC=SIECPTE VIA RA5.

L'article SC en question correspond au code série de compte 'sc' de la zone CLT.

VII.3.4.3. Passage de la modification de n-uples d'une relation à la modification des articles.

Ce problème peut se traiter de la même manière que l'adjonction: nous rencontrerons également les trois cas mentionnés plus haut. Dressons un parallèle entre les deux processus:

Structure conceptuelle.	Structure d'accès.
-désignation des informations à modifier: R=	-désignation du(des) article(s) concerné(s).
-obtention des n-uples dans la zone de travail.	-garnissage des zones du(des) article(s) concerné(s) dans la ZT.
-instructions du langage hôte modifiant les éléments du n-uple.	-modifications à l'aide du langage hôte des éléments des zones concernées
-ordre de modification: MODIFIER	-ordre(s) de modification: MODIFIER.

VII.3.4.4. Passage de la suppression de n-uples de relations à la suppression d'articles.

Nous pouvons, comme pour la modification dresser un parallèle entre les opérations dans les deux structures:

Structure conceptuelle.	Structure d'accès.
-désignation des informations à supprimer.	-désignation du(des) article(s) à supprimer.
-ordre de suppression.	-suppression du(des) article(s).

Remarque: Si à une relation correspond plusieurs articles, alors ceux-ci sont reliés par des relations totales entre les objets associés. En effet, supposons qu'une relation soit représentée par deux articles: si à un objet d'un type d'article n'est associé aucun objet de l'autre, cela veut dire que l'occurrence de la relation concernée est amputée d'une partie de ses informations. La suppression de n-uples de cette relation entraînera automatiquement la suppression de deux articles.

CHAPITRE VIII. SCHEMA IDS

Dans ce chapitre, nous allons d'abord voir comment peut s'effectuer le passage de la structure d'accès décrite précédemment au schéma IDS. Pour ce faire, nous procéderons à un rappel des différents éléments de l'IDS et nous verrons, à partir d'un exemple, la façon dont se transforme une structure dans une autre.

Dans un second temps, nous analyserons l'aboutissement de notre démarche(élaboration d'une structure conceptuelle en vue de la construction d'une structure d'accès optimale) par rapport au schéma IDS de la banque PARIBAS.

VIII.1. Rappels: les principaux éléments de l'IDS.(1)

Un fichier IDS est un fichier dont l'organisation est basée sur une méthode de rangement intégré des données permettant de définir une organisation logique des données et la non-duplication de celles-ci.

Les unités élémentaires d'informations de l'IDS sont les articles. Les relations logiques entre les articles sont définies par des chaînes.

VIII.1.1. Les articles.

Les informations contenues dans un fichier IDS sont groupées par article. Les articles sont répertoriés en type ou modèle dont le nombre peut varier entre 1 et 999. Chaque type d'article peut comprendre un nombre quelconque d'articles de cette espèce. La structure d'un arti-

(1) tiré de l'ouvrage repris dans la bibliographie sous le numéro 19.

cle IDS est la suivante:-zone d'identification: type et longueur

-zone(s) de données: zone de longueur quelconque identique pour chaque type d'article

-zone(s) de chaînage: elle contient un ensemble de pointeurs, chaque pointeur permet de retrouver une information enregistrée dans la zone de données d'un autre article.

VIII.1.2. Les chaînages.

Un chaînage définit les relations logiques entre les unités élémentaires d'informations. Parmi ces relations on mentionnera:

-celles qui relient une information d'un type donné et un nombre plus ou moins élevé d'informations d'un autre type;

-celles qui relient des informations d'un même type.

Un chaînage s'exprime sous la forme d'une suite de pointeurs; un pointeur désigne l'adresse de l'article suivant dans la chaîne.

VIII.1.3. Le graphe logique IDS.

Soient X l'ensemble des types ou modèles d'article,
Γ l'ensemble des relations entre les modèles d'article;
le couple $G=(X,\Gamma)$ représente le graphe IDS:

-les sommets du graphe sont les modèles d'article;

-les arcs du graphe sont les modèles des chaînes;

-ce graphe est orienté et sans circuits;

-pour chaque chaîne: -le sommet origine correspond au modèle d'article, maître de la chaîne;
-le sommet terminal de l'arc correspond au modèle d'article, détail de la chaîne.

Chaque modèle d'article peut être maître ou détail de plusieurs chaînes. Chaque modèle de chaîne admet un seul modèle d'article maître; par contre, des articles de types différents peuvent être détails d'une même chaîne.

VIII.1.4. L'accès aux informations.

IDS comprend trois classes d'articles: les articles calculés, primaires et secondaires.

a) Les articles calculés.(maître ou détail)

L'adresse IDS d'un article calculé est obtenue par un algorithme de calcul(randomisation) effectué à partir d'un indicatif appartenant à la zone de données de l'article. Les articles calculés sont liés entre eux par une chaîne calculée qui n'est pas décrite par l'utilisateur. Pour les identifier, il mentionne l'identifiant des articles -clé de randomisation-sur lequel IDS applique sa formule.

b) Les articles primaires.(maître ou détail)

L'adresse IDS des articles primaires est définie en partie par l'utilisateur, en partie par IDS lui-même. L'utilisateur garnit la zone d'identification avec un nombre quelconque et IDS calcule à partir de celui-ci une adresse à laquelle il tente de stocker cet article. Ils ont pour but de créer une certaine géographie de la base de données et une répartition en sous-fichiers fonctionnels.

c)Les articles secondaires.(détail)

L'article étant détail doit être nécessairement rattaché à un maître par l'intermédiaire d'une chaîne. Ce maître peut être un article primaire, secondaire ou calculé. Le rattachement à un maître pose trois problèmes: -le choix de la chaîne;

-la détermination du point d'insertion logique dans la chaîne;

-le stockage physique.

VIII.1.5.Le langage IDS.

Le langage IDS n'est pas un langage en soi. Il utilise le COBOL en lui adjoignant quelques instructions spécifiques au traitement d'articles sur disque. Parmi celles-ci, citons:

-OPEN et CLOSE: ouverture et fermeture du fichier IDS;

-STORE: stocker sur disque un article qui vient d'être composé dans le buffer;

-RETRIEVE: retrouver un article sur disque et l'amener dans le buffer;

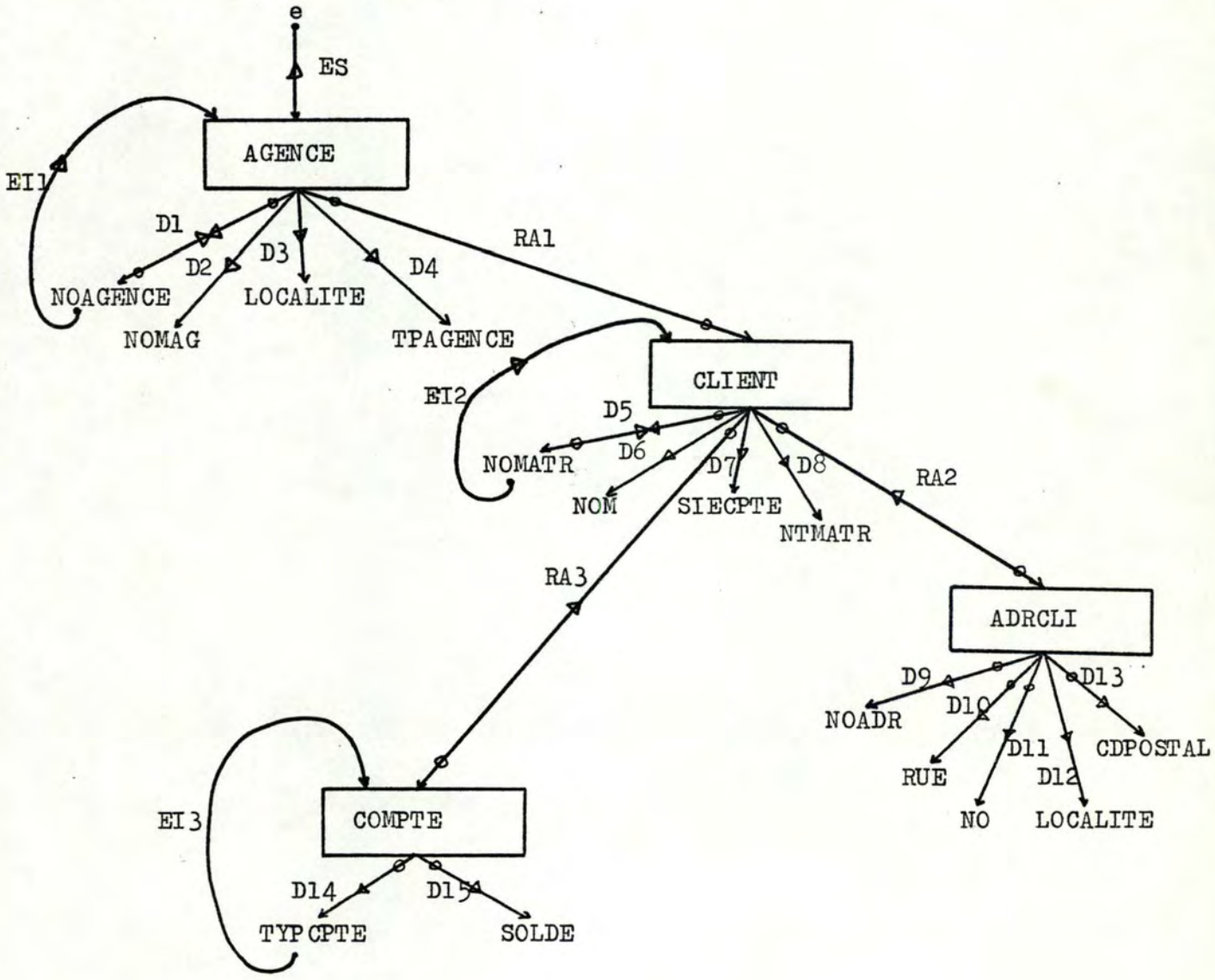
-DELETE : effacer un article sur disque;

-MODIFY: modifier une zone d'un article(précédé d'un RETRIEVE).

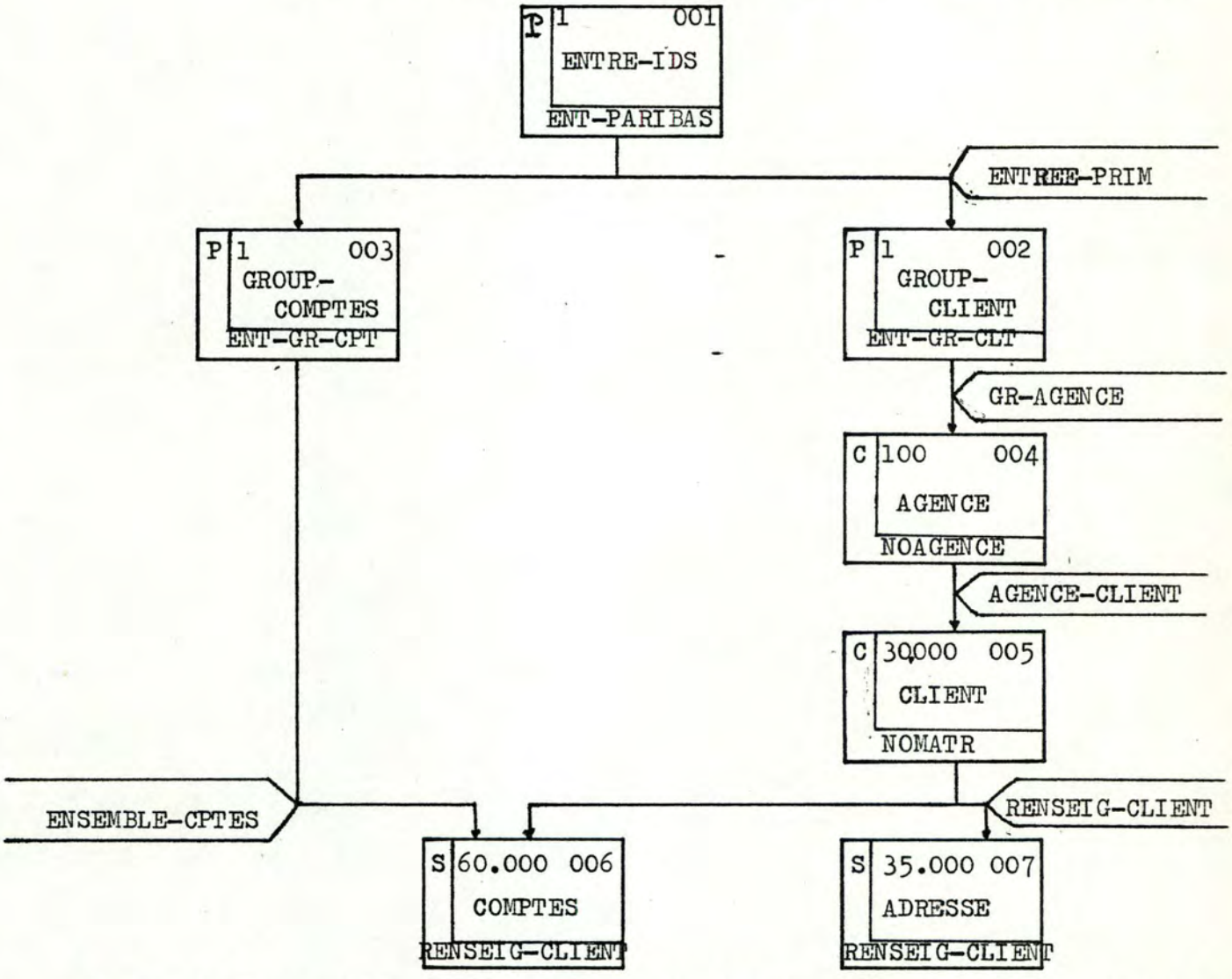
VIII.2. Passage au schéma IDS.

VIII.2.1. Exemple.

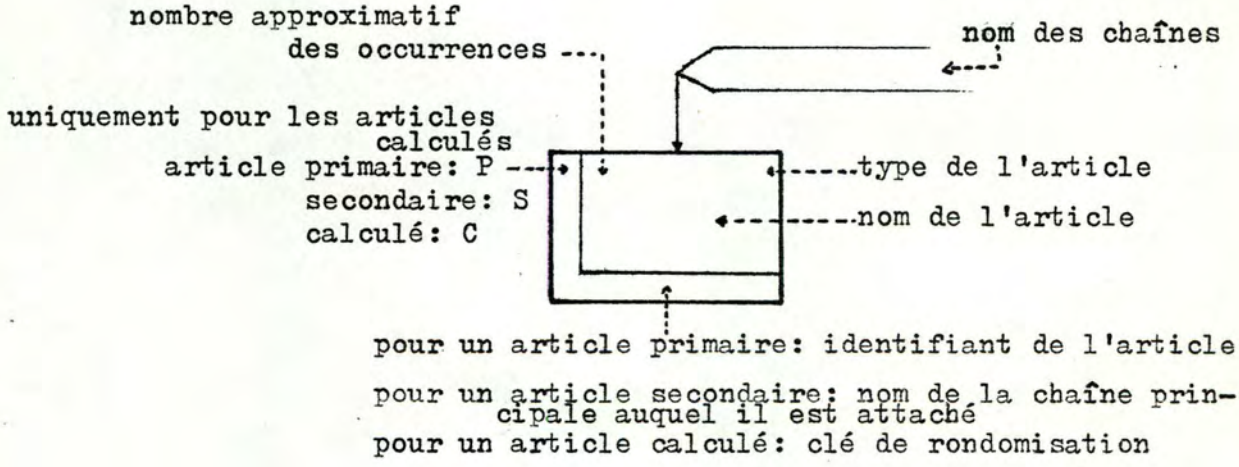
Reprenons le schéma de la structure d'accès que nous avons construite au paragraphe VII.3.2.2.



Construisons un graphe logique IDS reprenant ces éléments:



Remarque: Le symbolisme utilisé pour décrire ce graphe est celui qui est adopté à la banque PARIBAS.



VIII.2.2. Passage de la structure d'accès "théorique" à la structure IDS.

En IDS, chaque article utilisé dans un programme doit faire l'objet d'une description de niveau 01 de la section IDS. Nous reprenons ici le schéma de cette description: nous verrons ensuite comment il est possible d'y retrouver tous les éléments de la structure d'accès du chapitre précédent.

VIII.2.2.1. Description simplifiée d'un type d'article IDS.

```
01 | nom-article
   | TYPE IS i1
   | RETRIEVAL VIA { nom-zone-1 FIELD
   |                nom-chaîne-1 CHAIN
   |                CALC CHAIN }
02 | nom-zone-1 PIC ...
01 | ARTICLE
02 | ZONE-1
02 | ZONE-2
 :
98 | nom-chaîne-2 CHAIN MASTER
98 | nom-chaîne-3 CHAIN DETAIL
98 | CALC CHAIN DETAIL {RANDOMIZE ON nom-zone-2}
```

Commentaires:

-Chaque article est identifié par le symbole correspondant à "nom-article".

-i_1 donne la valeur numérique du type article: $1 \leq i_1 \leq 999$.

-La procédure pour stocker et (ou) retrouver l'article est indiquée par:

- "nom-zone-1 FIELD" pour la procédure primaire(article primaire); "nom-zone-1", indiquant la zone d'identification, sera défini immédiatement après le 01: 02 nom-zone-1...
- "nom-chaîne-1 CHAIN" pour la procédure secondaire(article secondaire);
- "CALC CHAIN" pour la procédure calculée(article calculé).

-La structure de l'article dans le fichier est décrite par l'ensemble des niveaux 02.

-Un niveau 98 doit être défini pour chacune des chaînes où l'article est inclus: -CHAIN MASTER: l'article est maître dans la chaîne citée par nom-chaîne-2

-CHAIN DETAIL: l'article est détail dans la chaîne citée par nom-chaîne-3

-Les articles calculés étant détails de la CALC CHAIN, ils doivent être suivis de la description de cette chaîne pour fournir la clé de randomisation(RANDOMIZE ON nom-zone-2).

Exemple:

```
01 | CLIENT
   | TYPE IS 005
   | RETRIEVAL VIA CALC CHAIN
01 | ARTCLT
02 | NOMATR...
02 | NOM ...
 :
 :
98 | RENSEIG-CLIENT CHAIN MASTER
98 | AGENCE-CLIENT CHAIN DETAIL
98 | CALC CHAIN DETAIL RANDOMIZE ON NOMATR
```

VIII.2.2.2. Recherche des éléments descriptifs de la structure d'accès du chapitre précédent.

a) Nous avions décrit un type d'article par un ensemble d'objets(un objet est associé à un type d'article), un ensemble de données et les relations binaires associant à un objet les différentes données.

Exemple: Type d'article CLIENT

Données: NOMATR,NOM,...

Relations: D5(CLIENT,NOMATR),D6(CLIENT,NOM)...

La notion d'objet associé à un type d'article disparaît: deux articles d'un même type se distinguent par leur adresse dans le fichier. La notion de type d'article est commune. Les relations binaires n'existent pas mais sont remplacées par la description des différentes zones de l'article. Remarquons également que l'on indique la classe(primaire, secondaire, calculée) à laquelle appartient l'article et, par le fait même,la façon dont on peut y accéder.

b) Les relations binaires entre les types d'article trouvent leur équivalent dans la détermination des différentes chaînes(descriptions des niveaux 98).

Exemples: La relation binaire RA1(AGENCE,CLIENT) est équivalente à la chaîne AGENCE-CLIENT. Les relations RA2(CLIENT,ADRCLI) et RA3(CLIENT,COMPTE) sont regroupées dans la chaîne RENSEIG-CLIENT.

Remarques: La chaîne RENSEIG-CLIENT a comme maître unique l'article CLIENT. Les types d'article COMPTES et ADRESSE sont détails de la chaîne RENSEIG-CLIENT. Le type d'article COMPTES est également détail de la chaîne ENSEMBLE-CPTES.

c) Les points d'entrée déterminent les débuts des chemins d'accès auxquels l'utilisateur peut faire appel.

Le point d'entrée par le système(élément e unique) est représenté par l'article primaire ENTRE-IDS.

Les articles calculés traduisent les "entrées inversées" qui permettent l'accès à un article d'un type donné à partir de différentes données.

Exemples: Les relations EI1(NOAGENCE,AGENCE) et EI2(NOMATR,CLIENT) sont représentées par le fait que les articles AGENCE et CLIENT du schéma IDS sont des articles calculés.

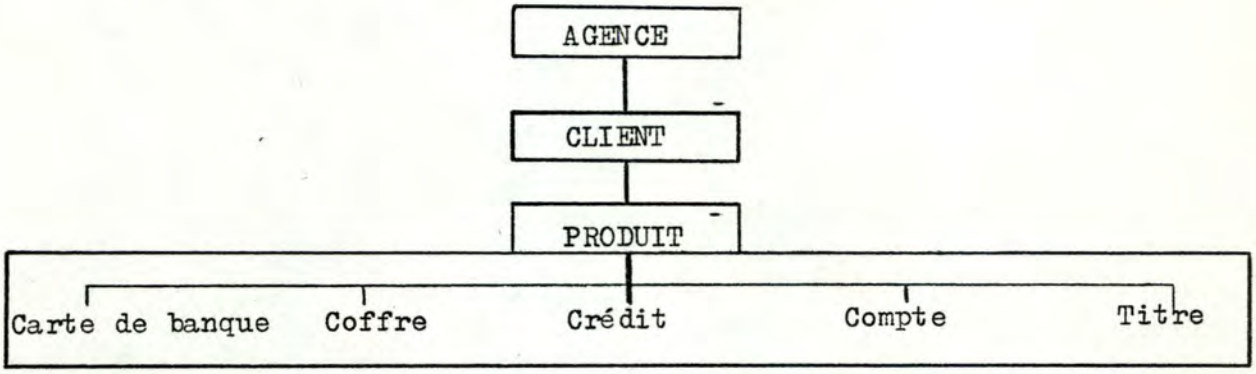
VIII.3. Description du schéma IDS de PARIBAS.

Dans ce paragraphe, nous allons voir comment s'effectue le parallèle entre les éléments que nous avons dégagés de la structure conceptuelle et ceux du schéma IDS retenu par la banque PARIBAS. Nous montrerons de quelle manière cette structure est d'une aide appréciable pour le gestionnaire dans la recherche de solutions aux problèmes qu'il rencontre.(1)

(1) cfr.chapitre III.3.

VIII.3.1. La structure CLIENT-AGENCE-PRODUIT.

Très schématiquement, les liaisons entre ces trois structures sont réalisées de la façon suivante:



Il est important pour la banque PARIBAS dont le siège de Bruxelles est le siège central de posséder les renseignements concernant toutes ses agences. A une agence sont rattachés tous ses clients, et à chaque client sont attachés tous ses produits: coffres, crédits, comptes,... Reprenons chacune de ces trois sous-structures et analysons-les séparément.

VIII.3.1.1. AGENCE.

Le choix de l'agence comme sommet de la biérarchie de ces trois sous-structures nous paraît justifié dans le cadre d'une banque multi-sièges. Outre le fait de représenter la structure de l'organisme financier, cela permet également d'effectuer des études de rentabilité agence par agence. Cette rentabilité dépendra, entre autres, du nombre de clients et des produits qui leur sont vendus: ceci peut justifier la dépendance des deux autres sous-structures par rapport à l'agence. Elle est aussi fonction de la rentabilité de chacun de ses services. L'objet SERVICE peut être associé à la notion de centre de frais et, vu sous cet angle, répond aux besoins des gestionnaires(contrôle budgétaire: dépenses mensuelles, budget annuel,...).

VIII.3.1.2. CLIENT.

L'objet CLIENT contient les renseignements généraux concernant tous les clients; nous avons distingué aussi un objet ADRESSE-CLIENT. Ces objets sont à l'origine de deux articles. Cette façon de voir vise deux objectifs: l'économie de place mémoire et une meilleure adaptation de la structure pour les besoins de l'utilisateur. En effet, un client possède un nombre variable d'adresses(une ou plusieurs en Belgique, parfois même à l'étranger): tous les articles d'un type donné ayant des longueurs identiques, il serait très difficile de trouver un compromis qui minimise la perte de place. D'autre part, la correspondance joue un rôle très important pour les contacts entre les clients et la banque. Pour les services intéressés, il est plus facile de manipuler l'entité ADRESSE dépouillée de toutes les autres données qui ne les intéressent pas.

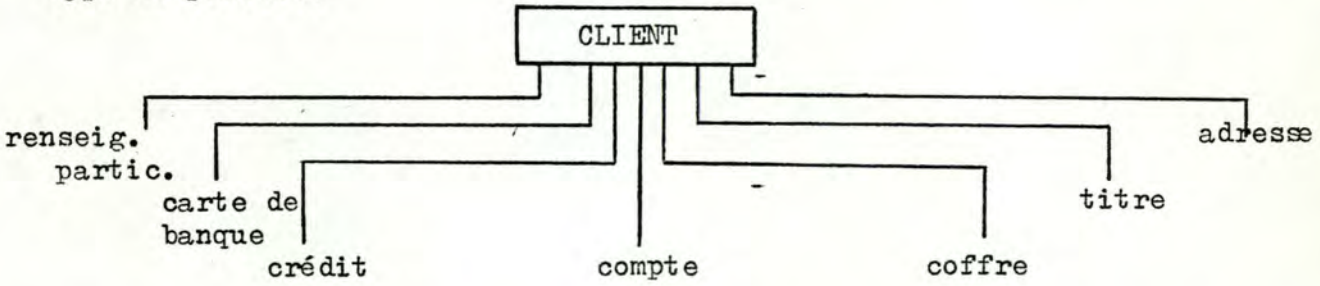
Un autre problème lié à la représentation de l'objet CLIENT est constitué par le fait que certains renseignements ne sont pas significatifs pour tous les clients(numéro de TVA, numéro de code du "Central des Risques",...). Chez PARIBAS, toujours pour économiser de la place, ils ont fait figurer ces renseignements dans des articles liés à l'article maître CLIENT. Ils appartiennent, ainsi que les articles ADRESSE, à la même chaîne reliant les produits au client.

L'article CLIENT étant très important dans la structure générale, un accès direct est autorisé à l'aide de la clé que constitue le numéro de matricule(NOMATR). Ceci a pour effet d'accélérer le processus de recherche des données.

Une autre caractéristique du client à laquelle on attache beaucoup d'intérêt est le code série de compte(classification des clients par type: particuliers, sociétés, notaires, banquiers,...). A chaque type de client est associé un intervalle qui correspond aux numéros de matricule attribuables à cette classe de clients. Un article SIECPTE (série de compte) est créé et chaîne tous les clients d'un même type. On dispose également d'un accès direct sur cet article dont la clé est le code série de compte.

VIII.3.1.3. CLIENT-PRODUIT.

Tous les produits sont reliés au client qui les détient. Ceci constitue une facilité pour l'étude de l'avoir d'un client. On aurait pu croire que la solution idéale consistait à créer une chaîne par type de produit.



Une étude quantitative nous montre le contraire: on dénombre environ 40.000 clients, 60.000 comptes, 10.000 titres, 6.000 crédits, 43.000 adresses, 600 coffres, 7.000 cartes de banque et 7.000 "renseignements particuliers". En moyenne un client possède donc un compte, une adresse et les autres informations sont absentes. Les chaînages qu'il faudrait établir seraient d'un coût trop élevé.

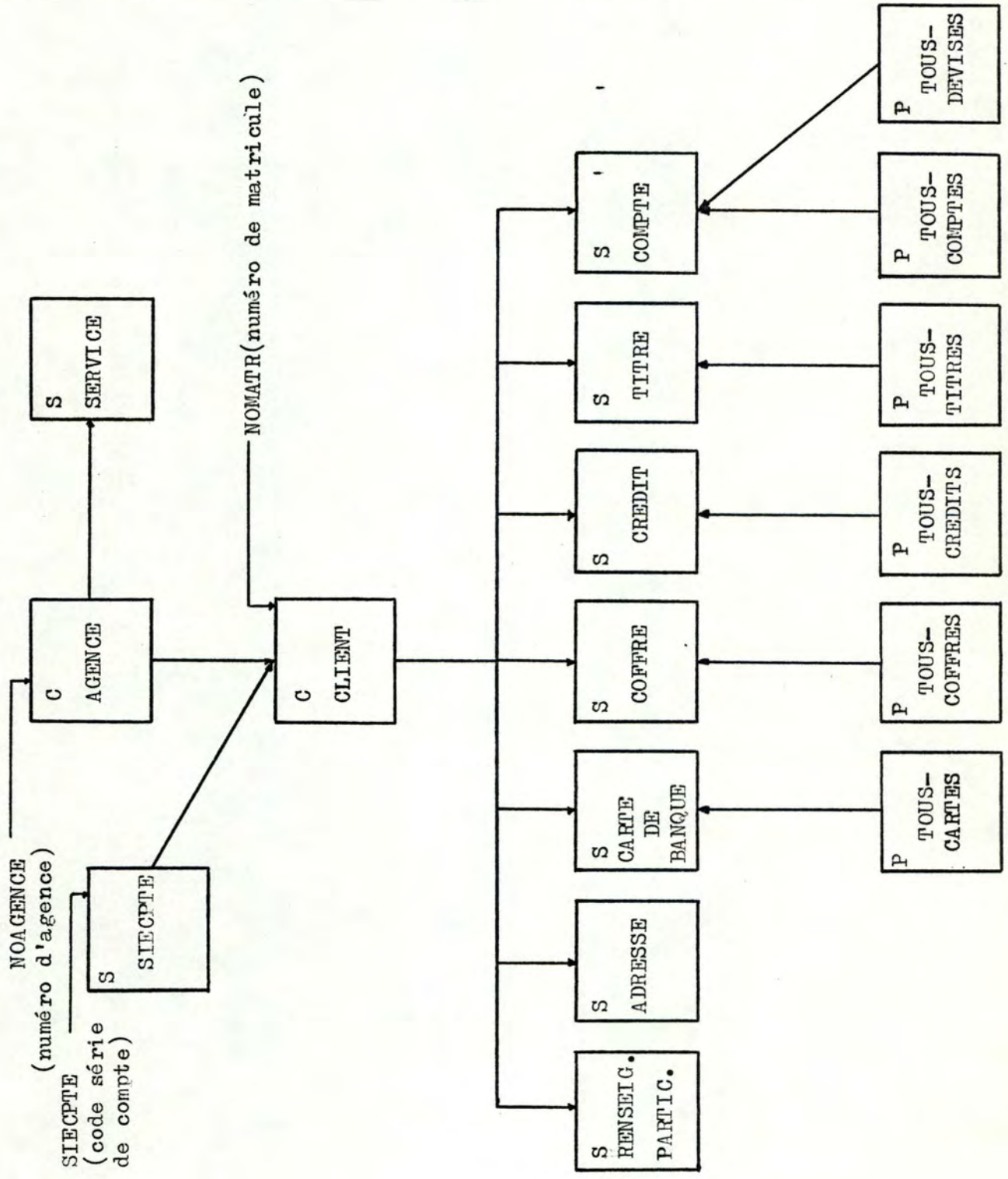
Le fait de relier tous les produits au client rend facile la suppression des données le concernant lors de sa disparition: il suffit de supprimer l'article CLIENT maître de la chaîne.

VIII.3.1.4. PRODUIT.

La notion de produit telle que nous l'avons présentée dans la structure conceptuelle n'est pas implémentée. Néanmoins pour faciliter les applications de gestion(études de rentabilité des différents types de produit), les produits d'un même type de tous les clients sont chaînés et reliés à un article maître(il s'agit d'un article primaire). De cette façon, il est possible d'accéder à tous les comptes d'un certain type, à tous les crédits,... Les comptes tenus dans une même devise sont également reliés dans une même chaîne.

VIII.3.2. Schéma IDS de PARIBAS.

Nous reprenons dans le schéma ci-dessous les principaux éléments dont nous venons de parler. Il coïncide avec le schéma IDS retenu par la banque PARIBAS.



CONCLUSIONS

CONCLUSIONS

La banque PARIBAS utilise une base de données implémentée en IDS. L'idée première de ce mémoire était la suivante: en se donnant une démarche méthodologique personnelle, aboutirons-nous à un modèle différent? Cette méthodologie est du type "top-down", c'est-à-dire que nous procédons par spécifications progressives des différentes étapes: construction du modèle conceptuel à partir d'un recensement des données(sans aucun souci d'implémentation), élaboration de la structure d'accès en fonction de l'ensemble des applications, passage au modèle IDS. Cette conception d'une base de données nous aurait peut-être permis d'aboutir à une structuration différente, à une mise en évidence de certains sous-schémas(cette notion n'existe pas en IDS).

Cette étude fut en partie infaisable et cela, principalement pour deux raisons. D'une part, ma connaissance du système bancaire étant assez réduite, la structure conceptuelle s'inspire en partie de leur réalisation. Et, d'autre part, le temps qu'il m'était imparti s'est avéré insuffisant pour dégager de l'étude de l'ensemble des applications les paramètres indispensables pour la construction d'une structure d'accès optimale(critères de performance, minimisation du nombre des accès, redondance dans les données,...). Nous avons donc choisi un exemple restreint et nous nous sommes attardés sur l'aspect documentaire du schéma conceptuel et sur son apport vis-à-vis de l'indépendance logique.

Ce travail de fin d'études est assez limité; il ne doit être considéré que comme une étape d'analyse préliminaire qui pourra éventuellement servir de base à l'étude d'un cas similaire à celui qui nous était initialement suggéré. Nous pouvons néanmoins en tirer quelques conclusions ainsi que les prolongements qui en découlent:

- La description de la structure conceptuelle et des règles d'évolution a permis de développer un exemple complet à l'aide des concepts qui nous étaient proposés. Il nous semble que ceux-ci facilitent la construction du modèle conceptuel. Les critiques émises ne sont pas d'une importance primordiale

mais peuvent être reprises pour une étude plus approfondie.

-Nous nous sommes contentés d'une description graphique pour la structure d'accès: celle-ci mériterait qu'on s'y attarde plus pour la clarifier. Le modèle conceptuel est séparé de la structure d'accès: dans celui-ci nous décrivons les données et leurs propriétés tandis que dans la structure d'accès nous lions les lots d'informations les uns aux autres en vue d'optimiser l'exécution des programmes. Il serait éventuellement intéressant d'étudier la manière dont les éléments permettant cette optimisation devraient être collectés et exprimés en fonction des données du niveau précédent.

-A la notion d'indépendance logique nous avons associé celle de mapping. Les exemples traités sont assez simples: une description plus fouillée devrait être réalisée dans ce domaine.

-Le passage au modèle IDS fut trop rapide: certains éléments n'ont pas été abordés(l'ordre des articles dans les chaînes, par exemple). La question est de savoir si la structure d'accès est assez générale pour parvenir à en dégager complètement le schéma IDS.

ANNEXES

ANNEXE I

Description du monde réel perçu.

I. Définitions des objets et de leurs propriétés.

a) AGENCE :

-numéro de l'agence;
-type de l'agence (mécanisée ou non, subdivisée en services ou non);
-dénomination de l'agence;
-rue;
-numéro;
-code postal;
-localité;
-numéro de téléphone;
-numéro de télex;
-langue pour l'édition des tableaux comptables;
-chambre de compensation: -membre ou pas membre
-numéro de chambre de compensation;
-numéro de l'agence dont dépend l'agence en question;
-numéro de l'agence effectuant certains travaux pour l'agence en question.

b) SERVICE :

-numéro du service;
-numéro de l'agence auquel il appartient;
-dénomination du service.

c) CLIENT :

-numéro de matricule;
-pays;

-nationalité;
-état civil;
-profession;
-langue;
-date de naissance;
-numéro de fichier national;
-numéro de TVA éventuel;
-code INS du secteur économique dont fait partie le client;
-agence titulaire du client;
-date de création;
-code série de compte (=subdivision en série des numéros de compte sur base du numéro de matricule);
-nature du matricule;
-régime matrimonial;
-autorisation maritale ou parentale;
-mauvais renseignements;
-date de mise sur la liste noire;
-taux précompte mobilier;
-autorisation de détenir un compte en marché réglementé;
-compte financier;
-code statistiques;
-code échelle;
-numéro de CCP et (ou) numéro(s) du (des) compte(s) de ce client dans une autre banque;
-numéro de code du central des risques;
- numéro du secteur économique;
-code envoi jeux émissions;
-code utilisation client (jamais débiteur,pas de carnet de chèque,..)
-code carte de banque (peut ou non avoir des cartes de banque);

d) ADRES-CLIENT :

-numéro de matricule;
-titre (nom,raison sociale);
-rue;
-numéro;
-code postal;
-localité;

-blocage;

-langue pour la correspondance;

-code frais de port;

-pays;

-code groupe expédition;

-périodicité d'envoi d'ES (à chaque mouvement,semaine,mois,...);

-code destination;

-code groupe envoi.

e) PRODUIT :

-type de produit = catégorie -compte
-coffre
-carte
-crédit
-chèque;

-numéro -type de compte
-numéro de carte
-numéro de crédit
-numéro de chèque;

-libellé;

-taux (intérêt,tarif,commission);

-terme (durée de validité pour le taux).

f) DEVISE :

-identification de la devise;

-code marché réglementé (devise peut-elle être employée en marché réglementé);

-dénomination de la devise en néerlandais;

-dénomination de la devise en français;

-cours de la devise -date, terme du cours
-cours vendeur
-cours acheteur;

-conditions standards, par devise, pour le calcul des intérêts sur les comptes en monnaie étrangère:-taux créditeur ou débiteur
-limite
-code limite
-marge
-taux débiteur minimum
-commission débiteur

-type de commission;
-code base;
-date de début d'application des conditions reprises ici.

g) SER-CPTES : par série de compte, on aura les conditions standards pour le calcul des intérêts sur les comptes en francs belges.

-code série de compte;
-type de taux; (créditeur ou débiteur)
-marge;
-taux débiteur minimum;
-commission débiteur;
-limite;
-code limite;
-date de validité pour ces conditions;
-code base.

b) TAUXBASE :

-code base;
-taux de base;
-date de validité.

Remarque: le taux d'intérêt est composé de
-un taux de base
-plus une marge ou un taux spécial ou une commission dépendant de l'état du compte par rapport à certaines limites.

II. Définitions des associations et de leurs propriétés.

a) COMPTE (= PRODUIT - CLIENT - AGENCE) :

-numéro de matricule;
-type de compte;
-numéro de l'agence;
-devise;

-marché;

-nature du compte (libre, bloqué,...);

-solde comptable;

-date du dernier mouvement sur ce compte;

-date à laquelle le solde a changé de sens;

-numéro d'ES;

-nombre d'ES à envoyer
d'avis ES à envoyer;

-rubrique littérale = dénomination donnée par le client pour ses comptes à rubriques;

-mode de calcul des intérêts (30 jours par mois, jours exacts);

-périodicité de clôture des intérêts (mois,trimestre,semestre,an);

-compensation de solde;

-code intérêt (calcul ou pas);

-conditions non standards pour le calcul des intérêts débiteurs:
- -code base
- -marge
- -taux minimum débiteur
- -limite
- -code limite
- -taux commission
- -type de commission
- -date de validité de ces conditions
- -limite de crédit sur laquelle il y a lieu de prendre la commission;

-conditions non standards pour le calcul des intérêts créditeurs:
- -code base
- -marge
- -limite
- -code limite
- -date de validité des conditions reprises ici.

b) COFFRE (= CLIENT - PRODUIT - AGENCE) :

-numéro de matricule;

-numéro du coffre;

-numéro de l'agence;

-type de coffre;

-période de location;

-date d'échéance;

-date de préavis;

-code blocage pour succession;

-type de compte pour comptabiliser les frais de location.

c) CARTE-BANQUE (= CLIENT - PRODUIT - AGENCE) :

-numéro de matricule;
-numéro de la carte de banque;
-numéro de l'agence qui l'a délivrée;
-type de carte;
-date opposition;
-nom de la personne;
-type de compte sur lequel la carte est identifiée.

d) CREDIT (= CLIENT - PRODUIT - AGENCE):

-numéro de matricule;
-numéro du crédit;
-numéro de l'agence;
-type de crédit;
-date d'ouverture du crédit;
-délai de préavis;
-date de préavis donné;
-montant de la limite intérieure;
-montant de la limite extérieure;
-devise;
-marché;
-rubrique littérale à mettre par le secrétariat des crédits;
-date d'expiration du crédit accordé;
-garanties données par le client:-type de garantie
-montant engagé
-valeur réelle
-devise dans laquelle sont exprimés ces montants
-numéro de la garantie
-date de validité
-origine de la garantie;
-garanties émises par la banque:-type de garantie
-montant de la garantie
-devise dans laquelle ce montant est exprimé
-date d'échéance
-numéro de la garantie émise
-rubrique littérale;

-compte(s) bénéficiaire(s) du crédit;

-encours;

-limite à partir de laquelle on applique un taux d'intérêt;

-type de taux à appliquer;

-taux d'intérêt.

ANNEXE II

Représentation formelle.

```
CONCEPTUAL-ORGANIZATION
 CONCEPTUAL-STRUCTURE
  DATA;
   NOAGENCE=NUMAG                'numéro de l'agence'
                 FORMAT          FORM-1 : DECIMAL 3
                 IC-1 : CARDINAL 50,100 ;
   TPAGENCE                      'type de l'agence'
                 FORMAT          FORM-2 : VALUE(0,1,2,3,4,5,6,7,8) ;
   NMAGENCE                      'nom de l'agence'
                 FORMAT          FORM-3 : CHARACTER 16 ;
   RUE                           'rue'
                 FORMAT          FORM-4 : CHARACTER 20 ;
   NO                            'numéro'
                 FORMAT          FORM-5 : CHARACTER 4 ;
   CDPOSTAL                      'code postal'
                 FORMAT          FORM-6 : DECIMAL 4 ;
   LOCALITE                      'localité'
                 FORMAT          FORM-7 : CHARACTER 18 ;
   LGEDTC                        'langue pour l'édition des tableaux comptables
                 FORMAT          FORM-8 : VALUE(0,1,2,3)  UNKN=0 ;
   NOTLPHN                       'numéro de téléphone'
                 FORMAT          FORM-9 : DECIMAL 12  UNKN=0(12) ;
   NOTELEX                       'numéro de télex'
                 FORMAT          FORM-10 : DECIMAL 5  UNKN=0(5) ;
   MBCOMP                        'membre de la chambre de compensation'
                 FORMAT          FORM-11 : VALUE(0,1,2) ;
   NOCOMP                        'numéro chambre de compensation'
                 FORMAT          FORM-12 : CHARACTER 2 ;
```

```
SUCGEST                 'succursale gestion'
        FORMAT          FORM-13 : DECIMAL 3  UNKN=b(3) ;
SUCEXPL                 'succursale exploitation'
        FORMAT          FORM-14 : DECIMAL 3  UNKN=b(3) ;
NOSERVI                 'numéro de service'
        FORMAT          FORM-15 : DECIMAL 2 ;
NOMSERF                 'nom du service en français'
        FORMAT          FORM-16 : CHARACTER 12 ;
NOMSERN                 'nom du service en néerlandais'
        FORMAT          FORM-17 : CHARACTER 12 ;
LGADAG                  'langue pour la rédaction de l'adresse
                        d'une agence'
        FORMAT          FORM-18 : VALUE(1,2) ;
NOMATR                  'numéro de matricule du client'
        FORMAT          FORM-19 : DECIMAL 8  MIN=55000000
                                             MAX=55999999
        IC-2 : CARDINAL  30000,50000 ;
TITRE                   'titre,nom,raison sociale'
        FORMAT          FORM-20 : CHARACTER 56 ;
CDPAYSAN                'code pays analytique'
        FORMAT          FORM-21 : CHARACTER 3 ;
CDGNPAYS                'code général pays'
        FORMAT          FORM-22 : VALUE(B,L,R,C,E) ;
NATION                  'nationalité'
        FORMAT          FORM-23 : CHARACTER 3 ;
ETCIV                   'état civil'
        FORMAT          FORM-24 : CHARACTER 4 ;
PROFES                  'profession'
        FORMAT          FORM-25 : CHARACTER 2 ;
CDREDPR                 'code réduit profession'
        FORMAT          FORM-26 : VALUE(0,1,2,3,4,5,6,7,A,B,C,D,E,Z) ;
DATNAIS                 'date de naissance'
        FORMAT          FORM-27 : DECIMAL 6 ;
NUMFNAT                 'numéro fichier national'
        FORMAT          FORM-28 : DECIMAL 11 ;
```

```
CDINS                   'code INS'
        FORMAT          FORM-29 : DECIMAL 5  UNKN=0(5) ;
NUMTVA                  'numéro de TVA'
        FORMAT          FORM-30 : CHARACTER 10  UNKN=b(10) ;
LANGUE                  'langue'
        FORMAT          FORM-31 : VALUE(1,2,3,4,5,6) ;
NTMATR                  'nature du matricule'
        FORMAT          FORM-32 : VALUE(0,1,2,3,4) ;
DTCREAT                 'date de création'
        FORMAT          FORM-33 : DECIMAL 6 ;
SIECPTE                 'code série de compte'
        FORMAT          FORM-34 : DECIMAL 2   MIN=01
                                              MAX=79 ;
REGMAT                  'code régime matrimonial'
        FORMAT          FORM-35 : VALUE(0,1,2,3) ;
AUTPRMR                 'autorisation maritale ou parentale'
        FORMAT          FORM-36 : VALUE(0,1,2) ;
TAUPREC                 'taux précompte mobilier'
        FORMAT          FORM-37 : REAL 2,2  UNKN=9(4) ;
CDFRPORT                'code frais de port'
        FORMAT          FORM-38 : VALUE(0,1,2,3,4) ;
MVRENS                  'mauvais renseignements reçus'
        FORMAT          FORM-39 : DECIMAL 2  MIN=00
                                             MAX=63 ;
DTLSTNR                 'date de mise sur la liste noire'
        FORMAT          FORM-40 : CHARACTER 2
                        FORM-41 : VALUE(01,02,...,00,0N,0D,11,12,...,
                                        10,1N,1D,...,91,92,...,99,90,
                                        9N,9D) ;
AUTCPTR                 'autorisation compte réglementé'
        FORMAT          FORM-42 : VALUE(N,0,D) ;
UTILCLI                 'code utilisation client'
        FORMAT          FORM-43 : VALUE(0,1,2,3) ;
CPTEFIN                 'code compte financier'
        FORMAT          FORM-44 : VALUE(0,1,2,3) ;
STATS                   'code statistiques'
        FORMAT          FORM-45 : VALUE(0,1) ;
```

```
ECHELLE                    'code échelle'
          FORMAT           FORM-46 : VALUE(0,1) ;
NOBNRISQ                   'numéro banque nationale risque'
          FORMAT           FORM-47 : CHARACTER 7  UNKN=b(7) ;
SECTECO                    'secteur économique, banque nationale'
          FORMAT           FORM-48 : CHARACTER 4  UNKN=b(4) ;
CARTBQ                     'code carte de-banque'
          FORMAT           FORM-49 : VALUE(0,1,2,3) ;
NADREMI                    'numéro d'adresse pour le courrier autre
                           que ES'
          FORMAT           FORM-50 : CHARACTER 1 ;
ENVEMIS                    'code envois jeux émissions'
          FORMAT           FORM-51 : DECIMAL 1
                           FORM-52 : VALUE(0,1,2,3,4,5,6,7,8,9) ;
BLOCAGE                    'code blocage'
          FORMAT           FORM-53 : VALUE(0,1) ;
GRPEXP                     'code groupe expédition'
          FORMAT           FORM-54 : VALUE(0,1,2,3,4) ;
DESTIN                     'code destination'
          FORMAT           FORM-55 : CHARACTER 3 ;
PERES                      'périodicité ES'
          FORMAT           FORM-56 : VALUE(0,1,2,3) ;
GRENVOI                    'code groupe envoi'
          FORMAT           FORM-57 : VALUE(0,1,2) ;
NUMCPTE                    'numéro de compte dans d'autres institu-
                           tions financières'
          FORMAT           FORM-58 : DECIMAL 12 ;
TYPCPTE                    'type de compte'
          FORMAT           FORM-59 : CHARACTER 4 ;
CDNUDEV                    'codification numérique de la devise'
          FORMAT           FORM-60 : CHARACTER 3 ;
CDALDEV                    'codification alphabétique de la devise'
          FORMAT           FORM-61 : CHARACTER 3 ;
MARCHE                     'marché'
          FORMAT           FORM-62 : VALUE(b,F,C,B,S,L,R,E)  UNKN=b ;
```

```
NATURE                          'nature du compte'
                FORMAT          FORM-63 : VALUE(0,1,2,3,4,8,9) ;
SOLDE                           'solde du compte'
                FORMAT          FORM-64 : REAL  1,2,4 ;
RUBLIT                          'rubrique littérale'
                FORMAT          FORM-65 : CHARACTER 24 ;
DTDRMVT                         'date du dernier mouvement'
                FORMAT          FORM-66 : DECIMAL 6 ;
DTDEBIT                         'date à laquelle le solde a changé de sens'
                FORMAT          FORM-67 : DECIMAL 6 ;
NBEXES                          'nombre d'exemplaires d'ES'
                FORMAT          FORM-68 : DECIMAL 1 ;
AVIS                            'code avis'
                FORMAT          FORM-69 : DECIMAL 1 ;
NBEXAVES                        'nombre d'exemplaires d'avis ES'
                FORMAT          FORM-70 : CHARACTER 1 ;
MODCALC                         'mode de calcul des intérêts'
                FORMAT          FORM-71 : VALUE(0,1,2) ;
PERCLOT                         'périodicité pour la clôture des intérêts'
                FORMAT          FORM-72 : VALUE(0,1,2,3) ;
CDFRCLOT                        'code frais de clôture'
                FORMAT          FORM-73 : VALUE(0,1) ;
CCSOLDE                         'compensation de solde'
                FORMAT          FORM-74 : CHARACTER 2 ;
INTERET                         'code intérêt'
                FORMAT          FORM-75 : VALUE(0,1,2,3) ;
CDBASE=CDBASEM                  'code base'
                FORMAT          FORM-76 : DECIMAL 2  MIN=00
                                                     MAX=96 ;
TYPLIMIT                        'type de limite'
                FORMAT          FORM-77 : CHARACTER 1 ;
MARGE=MARGE1=MARGE2             'marge'
                FORMAT          FORM-78 : REAL  2,5 ;
LIMITE                          'limite'
                FORMAT          FORM-79 : DECIMAL 11 ;
```

```
TAUXDEB                    'taux minimum débiteur'
          FORMAT           FORM-80 : REAL  2,5 ;
TAUCOM                     'taux commission'
          FORMAT           FORM-81 : REAL  2,5 ;
TYPCOM                     'type de commission'
          FORMAT           FORM-82 : VALUE(0,1,2,3,4,5) ;
LIMCRED                    'limite de crédit'
          FORMAT           FORM-83 : DECIMAL 11 ;
DATEVAL                    'date de validité'
          FORMAT           FORM-84 : DECIMAL 6 ;
TYPCRD                     'type de crédit'
          FORMAT           FORM-85 : CHARACTER 3  MIN=001
                                                  MAX=992 ;
DATOUV                     'date d'ouverture du crédit'
          FORMAT           FORM-86 : DECIMAL 6 ;
DELPRV                     'délai de préavis'
          FORMAT           FORM-87 : CHARACTER 3  UNKN=000 ;
DTPREAV                    'date de préavis donné'
          FORMAT           FORM-88 : DECIMAL 6  UNKN=0(6) ;
LIMEXT                     'limite extérieure'
          FORMAT           FORM-89 : DECIMAL 11  UNKN=0(11) ;
LIMINT                     'limite intérieure'
          FORMAT           FORM-90 : DECIMAL 11  UNKN=0(11) ;
RUBLCR                     'rubrique littérale'
          FORMAT           FORM-91 : CHARACTER 30 ;
DATEXP                     'date d'expiration'
          FORMAT           FORM-92 : DECIMAL 6  UNKN=0(6) ;
TYPGARCL                   'type de garantie client'
          FORMAT           FORM-93 : VALUE(01,02,...,21,99) ;
MONTGAR                    'montant de la garantie'
          FORMAT           FORM-94 : DECIMAL 11 ;
MONTENG                    'valeur réelle engagée'
          FORMAT           FORM-95 : DECIMAL 11  UNKN=0(11) ;
DATECHRV                   'date d'échéance ou de renouvellement'
          FORMAT           FORM-96 : DECIMAL 6 ;
```

```
ORIGINE                    'origine de la garantie'
          FORMAT           FORM-97 : VALUE(0,1,2) ;
TYPGARBQ                   'type de garantie banque'
          FORMAT           FORM-98 : VALUE(01,02,...,14,99) ;
RUBLITCD                   'rubrique littérale'
          FORMAT           FORM-99 : CHARACTER 30 ;
DENDEVN                    'dénomination de la devise en néerlandais'
          FORMAT           FORM-100 : CHARACTER 18 ;
DENDEVF                    'dénomination de la devise en français'
          FORMAT           FORM-101 : CHARACTER 18 ;
CDMAREG                    'code marché réglementé'
          FORMAT           FORM-102 : VALUE(1,2) ;
COURSACH                   'cours acheteur'
          FORMAT           FORM-103 : REAL  7,7 ;
DATCOUR                    'date du cours'
          FORMAT           FORM-104 : DECIMAL 6 ;
TRMCOUR                    'terme du cours'
          FORMAT           FORM-105 : DECIMAL 6 ;
DENMARN                    'dénomination du marché en néerlandais'
          FORMAT           FORM-106 : CHARACTER 18 ;
DENMARF                    'dénomination du marché en français'
          FORMAT           FORM-107 : CHARACTER 18 ;
MARCHDEV                   'relation marché devise'
          FORMAT           FORM-108 : VALUE(1,2,3) ;
TAUXBASE                   'taux de base'
          FORMAT           FORM-109 : REAL  2,5 ;
NOCFFRE                    'numéro de coffre'
          FORMAT           FORM-110 : CHARACTER 4 ;
TYPCFF                     'type de coffre'
          FORMAT           FORM-111 : VALUE(A,B,C,D,E,F,G,H,I,J,T,M) ;
PERIOD                     'périodicité, durée de location'
          FORMAT           FORM-112 : VALUE(3,6,1) ;
DATECH                     'date d'échéance'
          FORMAT           FORM-113 : DECIMAL 6 ;
```

```
DAPREAV                    'date de préavis'
          FORMAT           FORM-114 : DECIMAL 6  UNKN=0(6) ;
BLOCSUC                    'blocage succession'
          FORMAT           FORM-115 : VALUE(0,1) ;
DOMICIL                    'domiciliation'
          FORMAT           FORM-116 : VALUE(0,N) ;
NOCART                     'numéro de carte'
          FORMAT           FORM-117 : CHARACTER 9 ;
DTOPPOS                    'date d'opposition'
          FORMAT           FORM-118 : VALUE(01,02,...,00,0N,0D,11,...,
                                             1D,...,91,92,...,90,9N,9D)
                                        UNKN=bb ;
NOCRTBQ                    'numéro carte de banque'
          FORMAT           FORM-119 : CHARACTER 8 ;
TYPCART                    'type de carte'
          FORMAT           FORM-120 : VALUE(1,2,3,4) ;
NOM                        'nom de la personne'
          FORMAT           FORM-121 : CHARACTER 26 ;
ENCOURS                    'encours crédit'
          FORMAT           FORM-122 : REAL  1,2,4 ;
TYPTAUX                    'type de taux'
          FORMAT           FORM-123 : CHARACTER 1 ;
DATOPER                    'date de l'opération'
          FORMAT           FORM-124 : DECIMAL 6 ;
NATOPRT                    'nature de l'opération'
          FORMAT           FORM-125 : CHARACTER 30 ;
MONTANT                    'montant de l'opération'
          FORMAT           FORM-126 : REAL  1,2,4 ;
NVSLD                      'nouveau solde'
          FORMAT           FORM-127 : REAL  1,2,4 ;
PERSDST                    'nom de la personne destinataire'
          FORMAT           FORM-128 : CHARACTER 25 ;
NOPRCHQ                    'numéro du premier chèque du carnet délivré'
          FORMAT           FORM-129 : DECIMAL 6 ;
NBCHUTI                    'nombre de chèques non encore utilisés'
          FORMAT           FORM-130 : CHARACTER 4 ;
```

```
DTDRCPT                    'dernière date de comptabilisation d'un
                           chèque de ce carnet'
          FORMAT           FORM-131 : CHARACTER 2 ;
TYPOPPOS                   'type d'opposition'
          FORMAT           FORM-132 : VALUE(1,2,3,4) ;
NOCPTE                     'numéro de compte'
          FORMAT           FORM-133 : DECIMAL 12 ;
TERME                      'terme'
          FORMAT           FORM-134 : DECIMAL 6 ;
COURSVD                    'cours vendeur'
          FORMAT           FORM-135 : REAL  7,7 ;
DBCR                       'débit ou crédit'
          FORMAT           FORM-136 : VALUE(D,C) ;
VALEUR                     'valeur'
          FORMAT           FORM-137 : DECIMAL 6 ;
DTDRINT=DTINT=DT           'date du (dernier) calcul des intérêts'
          FORMAT           FORM-138 : DECIMAL 6 ;
SLD=SLDJ                   'solde à la date DTDRINT'
          FORMAT           FORM-139 : REAL  1,2,4 ;
INTCUM=INT                 'intérêts cumulés à la date DTDRINT'
          FORMAT           FORM-140 : REAL  6,2 ;
NOCHQ                      'numéro du chèque'
          FORMAT           FORM-141 : DECIMAL 6 ;
```

ENTITIES;

NOADR		'numéro de l'adresse'
	GENERATOR	GENE-1 ;
NOES		'numéro d'ES'
	GENERATOR	GENE-2 ;
NOCRED		'numéro de crédit'
	GENERATOR	GENE-3 ;
NOGARCL		'numéro de garantie client'
	GENERATOR	GENE-4 ;
NOGARBQ		'numéro de garantie banque'
	GENERATOR	GENE-5 ;
CCPBQ		'numéro CCP-numéro compte banque'
	GENERATOR	GENE-6 ;
MVT		'numéro de mouvement'
	GENERATOR	GENE-7 ;
CP		'créditeur particulier'
	GENERATOR	GENE-8 ;
DP		'débiteur particulier'
	GENERATOR	GENE-9 ;

RELATIONS;

AGENCE (NOAGENCE,TPAGENCE,LGEDTC,NOTLPHN,NOTELEX,MBCOMP,NOCOMP,
SUCGEST,SUCEXPL) 'agence'

IC-3 : KEY (NOAGENCE)

IC-4 : OBLIGATORY-COMP (NOAGENCE,TPAGENCE,NOCOMP)

IC-5 : INT-CONST/OCCUR
$\forall l((l \in AGENCE) \Rightarrow ((NOAGENCE.l \neq '5XX') \wedge (SUCGEST.l \neq '5XX') \wedge (SUCEXPL.l \neq '5XX')))$

IC-6 : INT-CONST/REL
$\forall l((l \in AGENCE) \Rightarrow \exists v((v \in AGENCE) \wedge (l \neq v) \wedge (NOAGENCE.l = NOAGENCE.v)))$

IC-7 : INT-CONST/REL
$\forall l(((l \in AGENCE) \wedge (SUCEXPL \neq 'bbb')) \Rightarrow \exists v((v \in AGENCE) \wedge (NOAGENCE.v = SUCEXPL.l)))$

IC-8 : INT-CONST/REL
$\forall l(((l \in AGENCE) \wedge (SUCGEST \neq 'bbb')) \Rightarrow \exists v((v \in AGENCE) \wedge (NOAGENCE.v = SUCGEST.l)))$;

Commentaires:

IC-5 : le premier caractère de NOAGENCE,SUCEXPL,SUCGEST est différent de 5.

IC-6 : pas de création en double.

IC-7 et IC-8 : si SUCEXPL (SUCGEST) est différent de bbb, alors il existe une occurrence de la relation AGENCE avec ce numéro comme NOAGENCE.

ADR-AG (NOAGENCE,LGADAG,NMAGENCE,RUE,NO,CDPOSTAL,LOCALITE) 'adresse de l'agence'

IC-9 : KEY (NOAGENCE,LGADAG)

IC-10 : OBLIGATORY-COMP (NOAGENCE,LGADAG,RUE,LOCALITE)

IC-11 : FUNCT-REL (LOCALITE) (CDPOSTAL)

IC-12 : INT-CONST/REL
CARD-PROJEC (NOAGENCE,LGADAG) 1,2

IC-13 : INT-CONST/REL
$\forall l((l \in ADR\text{-}AG) \Rightarrow \exists v((v \in ADR\text{-}AG) \wedge (NOAGENCE.l = NOAGENCE.v) \wedge (LGADAG.l = LGADAG.v)))$;

Commentaires:

IC-12 : une agence a au moins une adresse et au plus 2(suivant la langue de rédaction).

IC-13 : pas de création en double.

SERVICE (NOAGENCE,NOSERVI,NOMSERF,NOMSERN) - 'service'

IC-14 : KEY (NOAGENCE,NOSERVI)

IC-15 : INT-CONST/OCCUR

$\forall 1((1 \in \text{SERVICE}) \Rightarrow (\text{NOSERVI}.1 \neq \text{'00'}))$

IC-16 : INT-CONST/REL

CARD-PROJEC (NOAGENCE,NOSERVI) 0,50 ;

Commentaires :

IC-16 : une agence a au plus 50 services.

CLIENT (NOMATR,CDPAYSAN,CDGNPAYS,NATION,ETCIV,PROFES,CDREDPR,DATNAIS, NUMFNAT,CDINS,NUMTVA,LANGUE,NTMATR,DTCREAT,SIECPTE,AGTITUL, REGMAT,AUTPRMR,TAUPREC,MVRENS,DTLSTNR,AUTCPTR,UTILCLI,CPTEFIN, STATS,ECHELLE,NOBNRISQ,SECTECO,CARTBQ,NADREMI,ENVEMIS) 'client'

IC-17 : KEY (NOMATR)

IC-18 : OBLIGATORY-COMP (NOMATR,CDPAYSAN,SIECPTE,DATNAIS,CDPROF,NTMATR, MVRENS,UTILCLI)

IC-19 : FUNCT-REL (SIECPTE,CDPAYSAN) (CDGNPAYS)

IC-20 : INT-CONST/REL

CARDINAL :50000

IC-21 : INT-CONST/REL

$\forall 1((1 \in \text{CLIENT}) \Rightarrow \nexists v((v \in \text{CLIENT}) \wedge (\text{NOMATR}.1 = \text{NOMATR}.v)))$

IC-22 : INT-CONST/OCCUR

$\forall 1((1 \in \text{CLIENT}) \Rightarrow (\text{NOMATR}.1 = \text{'55xxxxxx'}))$

TIME DTCREAT DATE NOMATR 'date de création du n° matricule'

IC-23 : INT-CONST/OCCUR

$\forall 1(((1 \in \text{CLIENT}) \wedge (\text{CDPAYSAN}.1 = \text{'999'})) \Rightarrow (\text{CDGNPAYS}.1 = \text{'R'}))$

IC-24 : INT-CONST/OCCUR

$\forall l(((l \in CLIENT) \wedge (CDPAYSAN.l = '150')) \Rightarrow (CDGNPAYS.l = 'L'))$

IC-25 : INT-CONST/OCCUR

$PAYS1 = \{999, 150\}$ $PAYS2 = \{380, 450, 370\}$ $PAYS3 = \{000\}$

$SCPTE1 = \{51, 63, 59\}$ $SCPTE2 = \{56, 60, 72\}$

$\forall l(((l \in CLIENT) \wedge (CDPAYSAN.l \notin (PAYS1 \cup PAYS3)) \wedge (SIECPTE.l \notin (SCPTE1 \cup SCPTE2))) \Rightarrow (CDGNPAYS.l = 'E'))$

IC-26 : INT-CONST/OCCUR

$\forall l(((l \in CLIENT) \wedge (CDPAYSAN.l \in PAYS3) \wedge (SIECPTE.l \notin (SCPTE1 \cup SCPTE2))) \Rightarrow (CDGNPAYS.l = 'B'))$

IC-27 : INT-CONST/OCCUR

$\forall l(((l \in CLIENT) \wedge (CDPAYSAN.l \notin PAYS1) \wedge (SIECPTE.l \in SCPTE2)) \Rightarrow (CDGNPAYS.l = 'R'))$

IC-28 : INT-CONST/OCCUR

$\forall l(((l \in CLIENT) \wedge (CDPAYSAN.l \notin (PAYS1 \cup PAYS2)) \wedge (SIECPTE.l \in SCPTE1)) \Rightarrow (CDGNPAYS.l = 'E'))$

IC-29 : INT-CONST/OCCUR

$\forall l(((l \in CLIENT) \wedge (CDPAYSAN.l \in PAYS2) \wedge (SIECPTE.l \in SCPTE1)) \Rightarrow (CDGNPAYS.l = 'R'))$

IC-30 : INT-CONST/OCCUR

$\forall l(((l \in CLIENT) \wedge (ETCIV.l = '2')) \Rightarrow ((REGMAT.l = '0') \wedge (AUTPRMR.l \neq '1')))$

IC-31 : INT-CONST/OCCUR

$ETCV = \{1, 3, 4\}$

$\forall l(((l \in CLIENT) \wedge (ETCIV.l \in ETCV)) \Rightarrow (AUTPRMR.l = '2'))$

IC-32 : INT-CONST/OCCUR

$PROF = \{S1, S2, S3, S4, S5, S6, S7\}$

$SCPTE3 = \{02, 05, 15, 16, 17, 18, 20, 30, 31, 32, 33, 35, 36, 37, 52, 53, 54, 55, 64, 65, 66, 68, 75, 76, 77, 78, 79\}$

$\forall l(((l \in CLIENT) \wedge (PROFES.l \in PROF)) \Rightarrow (SIECPTE.l \in SCPTE3))$

IC-33 : INT-CONST/OCCUR

$UTCL = \{2, 3\}$

$CRTBQ = \{1, 0\}$

$\forall l(((l \in CLIENT) \wedge (UTILCLI.l \in UTCL)) \Rightarrow (CARTBQ.l \in CRTBQ))$;

Commentaires :

IC-23 à IC-29 : cet ensemble de contraintes décrit le calcul du CDGNPAYS à partir de CDPAYSAN et SIECPTE ; il explicite ce qui avait été déclaré dans IC-19.

IC-21 : pas de création en double.

IC-30, IC-31 : décrivent les interdépendances entre l'état civil, le régime matrimonial et les autorisations maritales.

CLAUTINSTF (NOMATR,CCPBQ,NUMCPTE) 'client dans d'autres institutions financières'

IC-34 : KEY (NOMATR,CCPBQ)

IC-35 : INT-CONST/OCCUR

$\forall v((v \in \text{CLAUTINSTF}) \Rightarrow ((\text{NUMCPTE}.v \neq \text{'55xxxxxxxxxx'}) \wedge (\text{NUMCPTE}.v \neq \text{'56xxxxxxxxxx'})))$;

Commentaires :

Cette relation nous donne, pour un client, tous ses numéros de compte dans les autres institutions financières.

IC-35 : on emploie les numéros de compte normalisés. Chaque institution se voit attribuer un ou plusieurs numéros qui figurera (ront) en tête des numéros de compte des clients. Paribas a reçu les n° 55 et 56 : les numéros de compte dans les autres institutions ne peuvent donc commencer par ces deux nombres.

ADRCLI (NOMATR,NOADR,TITRE,RUE,NO,CDPOSTAL,LOCALITE,BLOCAGE,GRPEXP, DESTIN,PERES,GRENVOI,LANGUE,CDPAYSAN,CDFRPORT) 'adresses du client'

IC-36 : KEY (NOMATR,NOADR)

IC-37 : OBLIGATORY-COMP (NOMATR,NOADR,TITRE,LOCALITE,CDPAYSAN,BLOCAGE)

IC-38 : FUNCT-REL (LOCALITE) (CDPOSTAL)

IC-39 : INT-CONST/OCCUR

$\forall l(((l \in \text{ADRCLI}) \wedge (\text{CDPOSTAL}.l \neq \text{'bbbb'})) \Rightarrow (\text{LOCALITE}.l \neq \text{'b(18)'}))$

IC-40 : INT-CONST/OCCUR

$\forall l(((l \in \text{ADRCLI}) \wedge (\text{BLOCAGE}.l = \text{'1'})) \Rightarrow (\text{DESTIN}.l = \text{'**A'}))$

IC-41 : INT-CONST/OCCUR
$\forall 1(((1 \in ADRCLI) \wedge (RUE.1='\flat(20)') \wedge (NO.1='\flat(4)') \wedge (LOCALITE.1='\flat(18)')) \Rightarrow (BLOCAGE.1='1'))$

IC-42 : INT-CONST/REL
$\forall 1(((1 \in ADRCLI) \wedge (NOADR.1 \neq '1')) \Rightarrow \exists v((v \in ADRCLI) \wedge (NOMATR.1=NOMATR.v) \wedge (NOADR.v='1')))$

IC-43 : INT-CONST/OCCUR
$\forall 1(((1 \in ADRCLI) \wedge (CDFRPORT.1='2')) \Rightarrow (DESTIN.1='RRR'))$

IC-44 : INT-CONST/OCCUR
$\forall 1(((1 \in ADRCLI) \wedge (CDFRPORT.1='3')) \Rightarrow (DESTIN.1='EEE'))$

IC-45 : INT-CONST/OCCUR
$\forall 1(((1 \in ADRCLI) \wedge (CDFRPORT.1='4')) \Rightarrow (DESTIN.1='REE'))$

IC-46 : INT-CONST/OCCUR
$\forall 1(((1 \in ADRCLI) \wedge ((GRPEXP.1='3') \wedge (GRPEXP.1='4'))) \Rightarrow (CDFRPORT.1='0'))$

IC-47 : INT-CONST/OCCUR
$\forall 1(((1 \in ADRCLI) \wedge (PERES.1 \neq '0')) \Rightarrow (GRPEXP.1='0'))$

IC-48 : INT-CONST/OCCUR
$\forall 1((1 \in ADRCLI) \wedge (GRPEXP \neq '0')) \Rightarrow (PERES.1='0'))$

IC-49 : INT-CONST/OCCUR
$CDST=\{MES, IIS, ANO, CAI, VEN, PER, xxA, xxC, SPF, SCA, SCC, RRR, EEE, REE, ESF, MAN, SV5\}$
$\forall 1(((1 \in ADRCLI) \wedge (DESTIN.1 \in CDST)) \Rightarrow (GRPEXP.1='0'))$;

Commentaires :

IC-39 : si le code postal est mentionné, la localité doit l'être aussi.

IC-41 : si le n°, la rue, la localité sont absents, il s'agit d'une adresse bloquée.

IC-42 : pour un matricule, il y a nécessairement une adresse qui porte le numéro 1.

COMPTE (NOMATR, NOAGENCE, TYPCPTE, CDNUDEV, CDALDEV, MARCHE, NATURE, SOLDE, RUBLIT, DTDRMVT, DTDEBIT, NBEXES, AVIS, NBEXAVES, MODCALC, PERCLOT, CCSOLDE, INTERET, NOADR) 'compte du client'

IC-50 : KEY (NOMATR, NOAGENCE, TYPCPTE)

IC-51 : OBLIGATORY-COMP (NOMATR,NOAGENCE,TYPCPTE,CDNUDEV,MARCHE,SOLDE)
IC-52 : FUNCT-REL (CDALDEV) (CDNUDEV)
IC-53 : INT-CONST/OCCUR
$\forall 1(((1\in COMPTE)\wedge((TYPCPTE.1='06xx')\wedge(TYPCPTE.1='76xx')))$
$\Rightarrow(NATURE.1\neq'0'))$
IC-54 : INT-CONST/OCCUR
$\forall 1(((1\in COMPTE)\wedge(TYPCPTE.1='xx41'))\Rightarrow((MARCHE.1='L')\wedge$
$(CDNUDEV.1\neq'000')\wedge(CDNUDEV.1\neq'150')))$
IC-55 : INT-CONST/OCCUR
$\forall 1(((1\in COMPTE)\wedge(NBEXES.1='0'))\Rightarrow((NBEXAVES.1='0')\wedge(NOADR.1='1')))$;

CRDPRT (CP,NOMATR,NOAGENCE,TYPCPTE,CDBASE,MARGE,LIMITE,TYPLIMIT,
DATEVAL) 'conditions particulières pour le calcul des intérêts créditeurs d'un compte'

IC-56 : KEY (CP)
IC-179 : INT-CONST/REL
CARD-PROJEC ((NOMATR,NOAGENCE,TYPCPTE,DATEVAL),(CDBASE,
MARGE,LIMITE,TYPLIMIT)) 3,3 ;

DBPRT (DP,NOMATR,NOAGENCE,TYPCPTE,CDBASE,CDBASEM,MARGE1,MARGE2,TAUXDEB,
TAUCOM,TYPCOM,LIMITE,TYPLIMIT,LIMCRED,DATEVAL) 'conditions particulières pour le calcul des intérêts débiteurs d'un compte'

IC-57 : KEY (DP)
IC-180 : INT-CONST/REL
CARD-PROJEC ((NOMATR,NOAGENCE,TYPCPTE,DATEVAL),(CDBASE,
CDBASEM,MARGE1,MARGE2,TAUXDEB,TAUCOM,TYPCOM,
LIMITE,TYPLIMIT,LIMCRED)) 3,3 ;

CREDIT (NOMATR,NOAGENCE,NOCRED,TYPCRD,DTOUV,DELPRV,DTPREAV,LIMEXT,
LIMINT,CDALDEV,MARCHE,RUBLCR,DATEXP,ENCOURS,LIMCRED,TYPTAUX,
MARGE1,MARGE2) 'crédit'

IC-58 : KEY (NOMATR,NOAGENCE,NOCRED)
IC-59 : OBLIGATORY-COMP (NOMATR,NOCRED,TYPCRD,ENCOURS,DTOUV,DATEXP) ;

GARCLI (NOMATR,NOAGENCE,NOCRED,NOGARCL,TYPGARCL,MONTGAR,CDALDEV,
MONTENG,DATECHRV,ORIGINE) 'garanties données par le client'

IC-60 : KEY (NOMATR,NOAGENCE,NOCRED,NOGARCL)
IC-61 : OBLIGATORY-COMP (NOMATR,NOAGENCE,NOCRED,NOGARCL,TYPGARCL,
MONTENG)
IC-62 : INT-CONST/OCCUR
$\forall 1(((1 \in GARCLI) \wedge (TYPGARCL.1='01')) \Rightarrow ((MONTGAR.1='0(11)') \wedge$
$(MONTENG.1='0(11)'))) ;$

GARBQ (NOMATR,NOAGENCE,NOCRED,NOGARBQ,TYPGARBQ,MONTGAR,CDALDEV,
DATECHRV,RUBLITCD) 'garanties émises par la banque'

IC-63 : KEY (NOMATR,NOAGENCE,NOCRED,NOGARBQ)
IC-64 : OBLIGATORY-COMP (NOMATR,NOAGENCE,NOCRED,NOGARBQ) ;

CPTSBENCD (NOMATR,NOAGENCE,NOCRED,CPTCRD,NOCPTE) 'comptes bénéficiant du crédit'

IC-65 : KEY (NOMATR,NOAGENCE,NOCRED,CPTCRD) ;

DEVISE (CDNUDEV,CDALDEV,DENDEVN,DENDEVF,CDMAREG) 'devise'

IC-68 : KEY (CDNUDEV)
IC-69 : OBLIGATORY-COMP (CDNUDEV,CDMAREG)
IC-70 : INT-CONST/REL
$\forall 1((1 \in DEVISE) \Rightarrow \nexists v((v \in DEVISE) \wedge (v \neq 1) \wedge (CDNUDEV.1=CDNUDEV.v)))$
IC-71 : FUNCT-REL (DENDEVN) (DENDEVF) ;

COURDV (CDNUDEV,DATCOUR,TRMCOUR,COURSACH,COURSVD,MARCHE) 'cours de la devise'

IC-71 : KEY (CDNUDEV,DATCOUR)
IC-72 : INT-CONST/OCCUR
DVNUM={000,150}
MRCH1={F,C,B,S,E,b,P}
MRCH2={L,R,P}
$\forall 1(((1 \in COURDV) \wedge (CDNUDEV.1 \in DVNUM)) \Rightarrow (MARCHE.1 \in MRCH1))$

IC-73 : INT-CONST/OCCUR
$\forall 1(((1 \in COURDV) \wedge (CDNUDEV.1 \notin DVNUM)) \Rightarrow (MARCHE.1 \in MRCH2))$;

INTCRDEV (CDNUDEV,MARCHE,DATEVAL,CDBASE1,MARGE1,LIMITE,TYPLIMIT,
CDBASE2,MARGE2) 'conditions pour le calcul des intérêts créditeurs sur les comptes en ME'

IC-74 : KEY (CDNUDEV,MARCHE,DATEVAL)
IC-181 : INT-CONST/REL
CARD-PROJEC ((CDNUDEV,MARCHE,DATEVAL),(CDBASE,MARGE,LIMITE,
TYPLIMIT)) 1,1 ;

INTDBDEV (CDNUDEV,MARCHE,DATEVAL,CDBASE,CDBASEM,MARGE1,MARGE2,TAUXDEB,
TAUCOM,TYPCOM,LIMITE,TYPLIMIT,LIMCRED) 'conditions pour le calcul des intérêts débiteurs sur les comptes en ME'

IC-75 : KEY (CDNUDEV,MARCHE,DATEVAL)
IC-182 : INT-CONST/REL
CARD-PROJEC ((CDNUDEV,MARCHE,DATEVAL),(CDBASE,CDBASEM,
MARGE1,MARGE2,TAUXDEB,TAUCOM,TYPCOM,LIMITE,
TYPLIMIT,LIMCRED)) 1,1 ;

INTCRSRCT (SIECPTE,MARCHE,DATEVAL,CDBASE1,MARGE1,LIMITE,TYPLIMIT,
CDBASE2,MARGE2) 'par série de comptes, on aura les conditions standards pour le calcul des intérêts créditeurs sur les comptes en FB'

IC-79 : KEY (SIECPTE,MARCHE,DATEVAL)
IC-80 : INT-CONST/REL
CARD-PROJEC ((SIECPTE,MARCHE,DATEVAL),(CDBASE,MARGE,LIMITE,
TYPLIMIT)) 1,1 ;

INTDBSRCT (SIECPTE,MARCHE,DATEVAL,CDBASE,CDBASEM,MARGE1,MARGE2,TAUXDEB,
TAUCOM,TYPCOM,LIMITE,TYPLIMIT,LIMCRED) 'par série de comptes, on aura les conditions standards pour le calcul des intérêts débiteurs sur les comptes en FB'

IC-81 : KEY (SIECPTE,MARCHE,DATEVAL)

IC-82 : INT-CONST/REL
CARD-PROJEC ((SIECPTE,MARCHE,DATEVAL),(CDBASE,CDBASEM,MARGE1,
MARGE2,TAUXDEB,TAUCOM,TYPCOM,LIMITE,TYPLIMIT,
LIMCRED)) 1,1 ;

TAUXBASE (CDBASE,DATEVAL,TAUXBASE) 'taux de base'

IC-83 : KEY (CDBASE,DATEVAL)
IC-84 : OBLIGATORY-COMP (TAUXBASE) ;

COFFRE (NOMATR,NOAGENCE,NOCFFRE,TYPCFF,PERIOD,DATECH,DAPREAV,BLOCSUC,
DOMICIL,TYPCPTE) 'location de coffre'

IC-85 : KEY (NOMATR,NOAGENCE,NOCFFRE)
IC-86 : OBLIGATORY-COMP (TYPCFF,DATECH,PERIOD,DOMICIL,BLOCSUC)
IC-87 : INT-CONST/OCCUR
$\forall 1((1 \in COFFRE) \Rightarrow (DAPREAV.1 \leq DATECH.1))$
IC-88 : INT-CONST/OCCUR
$\forall 1(((1 \in COFFRE) \wedge (DOMICIL.1='O')) \Rightarrow (TYPCPTE.1='b(4)'))$;

Commentaires :
IC-87 : la date de préavis est postérieure à la date d'échéance.
IC-88 : en cas de domiciliation du contrat de location, il doit exister un compte où enregistrer cette location.

CRTBQ (NOMATR,NOAGENCE,NOCRTBQ,TYPCART,DTOPPOS,NOM,TYPCPTE) 'carte de banque'

IC-92 : KEY (NOMATR,NOAGENCE,NOCRTBQ)
IC-93 : OBLIGATORY-COMP (TYPCART,TYPCPTE) ;

INTER-RELATIONS ;

IC-I02: EQUAL (NOAGENCE)AGENCE (NOAGENCE)ADR-AG

IC-I03: INCLUDED (NOAGENCE)SERVICE (NOAGENCE)AGENCE

IC-I04: INCLUDED (NOMATR)CLAUTINSTF (NOMATR)CLIENT

IC-I05: EQUAL (NOMATR)ADRCLI (NOMATR)CLIENT

IC-I06: EQUAL (NOMATR)COMPTE (NOMATR)CLIENT

IC-I07: INCLUDED (NOAGENCE)COMPTE (NOAGENCE)AGENCE

IC-I08: INCLUDED (NADREMI)CLIENT (NOADR)ADRCLI

IC-I09: INCLUDED (NOADR)COMPTE (NOADR)ADRCLI

IC-I10: INCLUDED (NOMATR,NOAGENCE,TYPCPTE)CRDPRT
(NOMATR,NOAGENCE,TYPCPTE)COMPTE

IC-I11: EQUAL (NOAGENCE)AGENCE (AGTITUL)CLIENT

IC-I12: INCLUDED (NOMATR,NOAGENCE,TYPCPTE)DBPRT
(NOMATR,NOAGENCE,TYPCPTE)COMPTE

IC-I13: INCLUDED (NOMATR)CREDIT (NOMATR)CLIENT

IC-I14: INCLUDED (NOAGENCE)CREDIT (NOAGENCE)AGENCE

IC-I15: INCLUDED (NOMATR,NOAGENCE,NOCRED)GARCLI
(NOMATR,NOAGENCE,NOCRED)CREDIT

IC-I16: INCLUDED (NOMATR,NOAGENCE,NOCRED)GARBQ (NOMATR,NOAGENCE,NOCRED)CREDIT

IC-I17: INCLUDED (NOMATR,NOAGENCE,NOCRED)CPTSBENCD
(NOMATR,NOAGENCE,NOCRED)CREDIT

IC-I18: EQUAL (TYPCPTE)COMPTE (TYPCPTE)INTYPCPTE

IC-I19: EQUAL (CDNUDEV,CDALDEV)COMPTE (CDNUDEV,CDALDEV)DEVISE

IC-I20: EQUAL (CDNUDEV)COURDV (CDNUDEV)DEVISE

IC-I21: EQUAL (CDNUDEV)DEVISE (CDNUDEV)INTCRDEV

IC-I22: EQUAL (CDNUDEV)DEVISE (CDNUDEV)INTDBDEV

IC-I23: EQUAL (SIECPTE)CLIENT (SIECPTE)INTCRSRCT

IC-I24: EQUAL (SIECPTE)CLIENT (SIECPTE)INTDBSRCT

IC-I25: INCLUDED (CDBASE,DATEVAL)CRDPRT (CDBASE,DATEVAL)TAUXBASE

IC-I26: INCLUDED (CDBASE,DATEVAL)DBPRT (CDBASE,DATEVAL)TAUXBASE

IC-I27: INCLUDED (CDBASEM,DATEVAL)DBPRT (CDBASE,DATEVAL)TAUXBASE

IC-I28: INCLUDED (CDBASE,DATEVAL)INTCRDEV (CDBASE,DATEVAL)TAUXBASE

IC-I29: INCLUDED (CDBASE,DATEVAL)INTDBDEV (CDBASE,DATEVAL)TAUXBASE

IC-I30: INCLUDED (CDBASEM,DATEVAL)INTDBDEV (CDBASE,DATEVAL)TAUXBASE

IC-I31: INCLUDED (CDBASE,DATEVAL)INTCRSRCT (CDBASE,DATEVAL)TAUXBASE

IC-I32: INCLUDED (CDBASE,DATEVAL)INTDBSRCT (CDBASE,DATEVAL)TAUXBASE

IC-I33: INCLUDED (CDBASEM,DATEVAL)INTDBSRCT (CDBASE,DATEVAL)TAUXBASE

IC-I34: INCLUDED (NOMATR)COFFRE (NOMATR)CLIENT

IC-I35: INCLUDED (NOAGENCE)COFFRE (NOAGENCE)AGENCE

IC-I36: INCLUDED (NOAGENCE,NOMATR,TYPCPTE)CRTBQ
(NOAGENCE,NOMATR,TYPCPTE)COMPTE

IC-I37: EVEMI={2,5,8}

$\forall 1(((1\in CLIENT)\wedge(ENVEMIS.1\ EVEMI))\Rightarrow\exists v((v\in ADRCLI)\wedge(NOMATR.v=NOMATR.1)$
$\wedge(NADREMI.1=NOADR.v)\wedge(CDPOSTAL.v='1xxx')\wedge(GRPEXP.v\neq'4')))$

IC-I38: EVEM2={3,6,9}

$\forall 1(((1\in CLIENT)\wedge(ENVEMIS.1\ EVEM2))\Rightarrow\exists v((v\in ADRCLI)\wedge(NOMATR.v=NOMATR.1)$
$\wedge(NADREMI.1=NOADR.v)\wedge(CDPOSTAL.v\neq'1xxx')\wedge(GRPEXP.v\neq'4')))$

IC-I39: EVEM3={1,4,7}

$\forall 1(((1\in CLIENT)\wedge(ENVEMIS.1\ EVEM3))\Rightarrow\exists v((v\in ADRCLI)\wedge(NOMATR.v=NOMATR.1)$
$\wedge(NADREMI.1\neq NOADR.v)\wedge\exists m((m\in AGENCE)\wedge(NOAGENCE.1=DESTIN.v))))$

IC-I40: $\forall 1(((1\in CLIENT)\wedge(NTMATR.1='3'))\Rightarrow\exists v((v\in COMPTE)\wedge(NOMATR.1=NOMATR.v)$
$\wedge(NATURE.v\neq'0')))$

IC-I41: NTMTR={1,2,4}

$\forall 1(((1\in CLIENT)\wedge(NTMATR.1\in NTMTR)\wedge\forall v((v\in COMPTE)\wedge(NOMATR.1=NOMATR.v)))\Rightarrow$
$\exists m((m\in ADRCLI)\wedge(NOMATR.1=NOMATR.m)\wedge(NOADR.m=NOADR.v)$
$\wedge(DESTIN.m='xx6')\wedge(GRPEXP.m='0')))$

IC-I42: $\forall 1(((1\in CLIENT)\wedge(CPTEFIN.1='1'))\Rightarrow\exists v((v\in COMPTE)\wedge(NOMATR.1=NOMATR.v)$
$\wedge(MARCHE.v='C')))$

IC-I43: $\forall 1((1\in COFFRE)\Rightarrow\exists v((v\in COMPTE)\wedge(NOMATR.1=NOMATR.v)\wedge(TYPCPTE.1=TYPCPTE.v)$
$\wedge(CDNUDEV.v='000')))$

IC-I44: GREP={3,4}

$\forall 1(((1\in ADRCLI)\wedge(GRPEXP.1\in GREP))\Rightarrow((CDFRPORT.1='0')\wedge((DESTIN.1='WTC')$
$\vee\exists v((v\in AGENCE)\wedge(DESTIN.1=NOAGENCE.v)))))$

IC-I45: TPCT1={06xx,76xx}

$\forall 1(((1\in COMPTE)\wedge(TYCPTE.1\ TPCT1))\Rightarrow\exists v((v\in CLIENT)\wedge(NOMATR.1=NOMATR.v)$
$\wedge(NTMATR.v\neq'0')\wedge(NATURE.1\neq'0')))$

IC-I46: $\forall 1(((1\in COMPTE)\wedge(TYPCPTE.1\neq'42')\wedge(MARCHE.1='R'))\Rightarrow$
$\exists v((v\in CLIENT)\wedge(NOMATR.1=NOMATR.v)\wedge((AUTCPTR.1='0')\vee(AUTCPTR.1='D'))))$

IC-I47: PAYS={380,450,370,SPZ}

$\forall 1((1\in COMPTE)\Rightarrow((CDNUDEV.1='000')\wedge\exists v((v\in CLIENT)\wedge(NOMATR.1=NOMATR.v)$
$\wedge(CDPAYSAN.v\in PAYS))))$

IC-I48: TPCT2={xx03,xx04,xx05}

$\forall 1(((1\in COMPTE)\wedge\exists v((v\in CLIENT)\wedge(NOMATR.1=NOMATR.v)\wedge(SIECPTE.v='61')))$
$\Rightarrow(TYPCPTE.1\in TPCPT2))$

IC-I49: SCPTE3={20,37,64,65,77,79,68,78}
DEVNUM={000,150}

$\forall 1(((1\in CLIENT)\wedge(SIECPTE.1\in SCPTE3)\wedge\forall v((v\in COMPTE)\wedge(NOMATR.1=NOMATR.v)$
$\wedge(CDNUDEV.v\ DEVNUM)))\Rightarrow(MARCHE.v='\text{ß}'))$

IC-I50: $\forall 1(((1\in COMPTE)\wedge(MARCHE.1='F'))\Rightarrow\exists v((v\in CLIENT)\wedge(NOMATR.1=NOMATR.v)$
$\wedge(CPTEFIN.v='0')))$

IC-I51: SCPTE4={56,58,60,72}
MRCH1={ß,B,C}

$\forall 1(((1\in CLIENT)\wedge(CDPAYSAN.1\neq'999')\wedge(CDGNPAYS.1='R')\wedge(SIECPTE.1\notin SCPTE4)$
$\wedge\exists v((v\in COMPTE)\wedge(NOMATR.1=NOMATR.v)\wedge(CDNUDEV.v\in DEVNUM)))\Rightarrow(MARCHE.v\in MRCH)$

IC-I52: $\forall l((l \in CLIENT) \wedge (CDPAYSAN.l \neq '999') \wedge (CDGNPAYS.l = 'R') \wedge (SIECPTE.l \notin SCPTE4) \wedge \exists v((v \in COMPTE) \wedge (NOMATR.l = NOMATR.v) \wedge (CDNUDEV.v \notin DEVNUM))) \Rightarrow (MARCHE.v \in MRCH))$

IC-I53: $\forall l((l \in CLIENT) \wedge (CDGNPAYS.l = 'E') \wedge \exists v((v \in COMPTE) \wedge (NOMATR.l = NOMATR.v) \wedge (CDNUDEV.v \notin DEVNUM))) \Rightarrow (MARCHE.v = 'L'))$

IC-I54: $MRCH2 = \{\delta, E, S\}$

$\forall l(((l \in CLIENT) \wedge (CDGNPAYS.l = 'E') \wedge (SIECPTE.l \notin SCPTE4) \wedge \exists v((v \in COMPTE) \wedge (NOMATR.l = NOMATR.v) \wedge (CDNUDEV.v \in DEVNUM))) \Rightarrow (MARCHE.v \notin MRCH2))$

IC-I55: $PSGN = \{B, L\}$

$\forall l(((l \in CLIENT) \wedge (CDGNPAYS.l \in PSGN) \wedge (SIECPTE.l \notin SCPTE4) \wedge \exists v((v \in COMPTE) \wedge (NOMATR.l = NOMATR.v) = (CDNUDEV.v\ DEVNUM))) \Rightarrow (MARCHE.v = '\delta'))$

IC-I56: $NAT = \{1, 2, 3, 4\}$

$\forall l(((l \in COMPTE) \wedge (NATURE.l \in NAT)) \Rightarrow ((TYPCPTE.l \in (TPCT1 \cup \{98\})) \wedge \exists v((v \in CLIENT) \wedge (NOMATR.l = NOMATR.v) \wedge (NTMATR.v \neq '0'))))$

IC-I57: $\forall l(((l \in CLIENT) \wedge (CDGNPAYS.l \in PSGN) \wedge \exists v((v \in COMPTE) \wedge (NOMATR.l = NOMATR.v) \wedge (CDNUDEV.v\ DEVNUM))) \Rightarrow (MARCHE.v \neq '\delta'))$

ANNEXE III

Les règles d'évolution.

I. Présentation des deux traitements décrits.

I.1. Quels sont les traitements décrits?

Pour illustrer par un exemple la notion de "règle d'évolution", nous avons choisi des traitements relatifs à un type de compte particulier: les comptes à vue. Pour être plus précis et pour se conformer à la terminologie employée dans le cours de M.I.S. (1), nous parlerons en termes d'application et de phase.

Nous aurons donc l'application "compte à vue" qui peut se décomposer en plusieurs phases telles que:

1. enregistrements des mouvements;
2. édition récapitulative des mouvements;
3. création d'un compte;
4. suppression d'un compte;
5. calcul des intérêts;
6. édition de la liste des clients débiteurs;
7. comptabilisation des intérêts;
8. éditions des échelles;
:

Les deux phases reprises ci-après en détails sont les phases 1. et 5.

I.2. Quels sont les événements déclencheurs de ces deux phases?

I.2.1. Enregistrement des mouvements.

En entrée, nous aurons un événement EVENT-1 qui nous renseignera sur le numéro de compte, sur la date du mouvement et sur le montant de celui-ci. Nous supposerons que tous ces montants sont signés. La procédure se résumera simplement à l'enregistrement de ce mouvement

(1) cfr. ouvrage 4. dans la bibliographie.

dans une relation que nous noterons 'MOUVEMENTS'.

I.2.2.Calcul des intérêts.

L'événement EVENT-2 qui déclenche cette seconde phase nous renseigne sur le numéro de compte pour lequel on décide de calculer les intérêts et la date à laquelle ce calcul est effectué.

I.3. Relations nécessaires pour les deux procédures.

I.3.1. Enregistrement des mouvements.

a)EVENT-1(NOMATR,NOAGENCE,TYPCPTE,VALEUR,MONTANT)

VALEUR = date à partir de laquelle on comptabilise les intérêts pour le montant en question.

MONTANT = montant du mouvement.

b)MOUVEMENTS(NOMATR,NOAGENCE,TYPCPTE,MVT,VALEUR,MONTANT)

Relation contenant les mouvements sur un compte à différentes dates. MVT est une entité générée dans la structure coceptuelle qui permet de distinguer deux mouvements de même montant sur un même compte et à une même date.

IC-175: KEY (NOMATR,NOAGENCE,TYPCPTE,MVT)

IC-178: OBLIGATORY-COMP (VALEUR,MONTANT)

I.3.2. Calcul des intérêts.

a)EVENT-2(NOMATR,NOAGENCE,TYPCPTE,DTINT)

DTINT est la date du calcul des intérêts pour le compte en question. Elle doit être supérieure à la dernière date du calcul des intérêts pour ce même compte.

b)Nous aurons également besoin de la relation 'COMPTE' qui nous fournira la devise et le marché du compte.

c)La relation 'CLIENT' nous renseignera sur le code série de compte du client.

d)INTERET(NOMATR,NOAGENCE,TYPCPTE,DTDRINT,SLD,INTCUM)

IC-176:KEY (NOMATR,NOAGENCE,TYPCPTE,DTDRINT)

IC-177:OBLIGATORY-COMP (DTDRINT,SLD,INTCUM)

Cette relation nous renseigne sur la date du dernier calcul des intérêts(DTDRINT), ainsi que sur le solde et les intérêts cumulés pour un compte à cette date. Elle nous permet d'avoir un historique du calcul des intérêts pour un compte.

e)Grâce aux informations mentionnées en b) et c), nous aurons accès aux éléments de 'CRDPRT,DBPRT,INTCRDEV,INTDBDEV, INTCRSRCT,INTDBSRCT' qui seront à la base du calcul des intérêts. Chacune de ces relations fait également appel à la relation 'TAUXBASE'.

I.4. Description du déroulement des procédures.

I.4.1. Enregistrement des mouvements.

PAS 1. Pour tout événement EVENT-1, pour tout mouvement
  PAS 1.1. Contrôle sur le compte indiqué :
    si le client existe
    si le numéro de compte est correct
    Aller à PAS 1.3.
  PAS 1.2. Sinon, déclencher la procédure 'référence erronnée'.
    Aller à PAS 1.4.
  PAS 1.3. Enregistrement du mouvement.
  PAS 1.4. Fin du traitement.

I.4.2. Calcul des intérêts.

PAS 1. Pour chaque événement EVENT-2, pour chaque décision de calcul des intérêts,
  PAS 1.1. Contrôle sur le compte indiqué :
    si le client est correct, connu de l'organisation, si le numéro de compte est correct,
    Aller à PAS 1.3.
  PAS 1.2. Sinon déclencher la procédure 'référence erronnée'.
    Aller à PAS 1.6.

PAS 1.3. Pour ce compte, recherche de la date du dernier calcul des intérêts.

PAS 1.4. Sélectionner, rechercher tous les mouvements sur le compte depuis le dernier calcul des intérêts jusqu'à la date du nouveau calcul.

PAS 1.5. Pour chaque jour de la période considérée.

PAS 1.5.1. Calcul du solde du compte avec les indications : solde créditeur ou débiteur .

PAS 1.5.2. Fin du calcul des soldes successifs du compte durant la période considérée.

Aller à PAS 2.

PAS 1.6. Fin du traitement.

PAS 2. Pour chaque solde créditeur.

PAS 2.1. Voir si le compte jouit de conditions particulières.

Si oui, aller à PAS 2.7.

PAS 2.2. Sinon, regarder si le compte est tenu en francs belges ou en devises étrangères.

Si c'est en devises étrangères, aller à PAS 2.5.

PAS 2.3. Si c'est en francs belges, à l'aide du code série de compte du client, rechercher les conditions standards pour le calcul des intérêts

PAS 2.4. Calcul des intérêts.

Aller à PAS 2.9.

PAS 2.5. Rechercher les conditions standards pour le calcul des intérêts sur le compte tenu en monnaies étrangères.

PAS 2.6. Calcul des intérêts.

Aller à PAS 2.9.

PAS 2.7. Rechercher les conditions particulières adéquates (date de validité).

PAS 2.8. Calcul des intérêts.

PAS 2.9. Fin du calcul des intérêts créditeurs pour chaque solde successif.

PAS 3. Pour chaque solde débiteur.

⋮ (traitement semblable au PAS 2.)

PAS 4. Mise à jour du solde et des intérêts cumulés à la date du dernier calcul des intérêts.

II. Description des deux procédures.

a) Enregistrement des mouvements.

```
PROC  ENREG-MVTS  'enregistrement des mouvements'
 TRIGGER  (EVENT-1)
 PARAMETERS
  EVENT-1  PAR-1  (NOMATR,NOAGENCE,TYPCPTE,VALEUR,MONTANT)
 EXT-INT-CONST-CONTEXT
  FORM-1  FORM-19  FORM-59  FORM-137  FORM-126
  EVENT-3 : (FORM-1 AND FORM-19 AND FORM-59)  PAR-1
  EVENT-4 : (FORM-137 AND FORM-126)  PAR-1
 INT-INT-CONST-CONTEXT
  IC-50  IC-51  IC-175  IC-178
  EVENT-5 : (IC-50 AND IC-51)  PAR-1
  EVENT-6 : (IC-175 AND IC-178)  PAR-1
 MIC-INT-CONST-CONTEXT
  MIC-1 : ∃1((1∈COMPTE)∧(NOMATR.1=NOMATR.PAR-1)
          ∧(NOAGENCE.1=NOAGENCE.PAR-1)∧(TYPCPTE.1=TYPCPTE.PAR-1))
  EVENT-7 : (MIC-1)  PAR-1
 PROC-DESCRIPTION
  BEGIN
   NM := NOMATR.PAR-1
   NAG := NOAGENCE.PAR-1
   TC := TYPCPTE.PAR-1
   VL := VALEUR.PAR-1
   MT := MONTANT.PAR-1
   AJOUTER (NM,NAG,TC,VL,MT,GENE-7) A MOUVEMENTS
  END
END  ENREG-MVTS
```

b) Calcul des intérêts.

```
PROC   CALSLD   'calcul des soldes'
 TRIGGER  (EVENT-2)
 PARAMETERS
  EVENT-2  PAR-1  (NOMATR,NOAGENCE,TYPCPTE,DTINT)
 EXT-INT-CONST-CONTEXT
  FORM-1 FORM-19 FORM-59 FORM-138
  EVENT-3 : (FORM-1 AND FORM-19 AND FORM-59)  PAR-1
  EVENT-8 : (FORM-138)  PAR-1
 INT-INT-CONST-CONTEXT
  IC-50  IC-51  IC-175  IC-176  IC-177  IC-178
  EVENT-5 : (IC-50 AND IC-51)  PAR-1
  EVENT-9 : (IC-176 AND IC-177)  PAR-1
  EVENT-6 : (IC-175 AND IC-178)  PAR-1
 MIC-INT-CONST-CONTEXT
  MIC-1 : ∃1((1∈COMPTE)∧(NOMATR.1=NOMATR.PAR-1)∧(NOAGENCE.1=NOAGENCE.PAR-1)
          ∧(TYPCPTE.1=TYPCPTE.PAR-1))
  EVENT-7 : (MIC-1)  PAR-1
 PROC-DESCRIPTION
  BEGIN
   DESIGN R1={(1)|((1∈INTERET)∧∀ v(((v∈INTERET)∧(v≠1)∧(NOMATR.1=NOMATR.v=
             NOMATR.PAR-1)∧(NOAGENCE.1=NOAGENCE.v=NOAGENCE.PAR-1)∧(TYPCPTE.1=
             TYPCPTE.v=TYPCPTE.PAR-1))⇒(DTDRINT.1>DTDRINT.v)))}
   OBTENIR R1
   DESIGN R2={(VALEUR.m,MONTANT.m)|((m∈MOUVEMENTS)∧(NOMATR.m=NOMATR.PAR-1)
             ∧(NOAGENCE.m=NOAGENCE.PAR-1)∧(TYPCPTE.m=TYPCPTE.PAR-1)
             ∧(DTDRINT.R1≤VALEUR.m<DTINT.PAR-1))}
   OBTENIR R2
   DATE := DTDRINT.R1
   NM := NOMATR.PAR-1
   NAG := NOAGENCE.PAR-1
   TC := TYPCPTE.PAR-1
   ITR := 0
   Suite 1
   SD := SLD.R1+SOM(((VALEUR,MONTANT)R2,MONTANT)|(VALEUR.R2≤DATE))
   SI (SD≥0) ALORS ALLER A Suite 2
   AJOUTER (NM,NAG,TC,DATE,SD,ITR) A SLDINTDB
   ALLER A Suite 3
```

```
   Suite 2
   AJOUTER (NM,NAG,TC,DATE,SD,ITR) A SLDINTCR
   Suite 3
   DATE := DATE+1
   SI (DATE≤(DTINT.PAR-1 - 1)) ALORS ALLER A Suite 1
  END
 INDUCED-EVENT  EVENT-10  PAR-1  SLDINTDB(NOMATR,NOAGENCE,TYPCPTE,DT,SLDJ,INT)
END


PROC   INTDEB   'calcul des intérêts débiteurs'
 TRIGGER  (EVENT-10)
 PARAMETERS
  EVENT-10  PAR-1  SLDINTDB(NOMATR,NOAGENCE,TYPCPTE,DT,SLDJ,INT)
 EXT-INT-CONST-CONTEXT
  EIC-1 : ∀1((1∈SLDINTDB)⇒∄m((m∈SLDINTDB)∧(m≠1)∧(DT.m=DT.1)))
  FORM-1  FORM-19  FORM-59  FORM-137  FORM-139  FORM-140
  EVENT-3 : (FORM-1 AND FORM-19 AND FORM-59)  PAR-1
  EVENT-8 : (FORM-137 AND FORM-139 AND FORM-140)  PAR-1
  EVENT-20 : (EIC-1)  PAR-1
 INT-INT-CONST-CONTEXT
  IC-17  IC-18  IC-50  IC-51  IC-57  IC-75  IC-81
  IC-83  IC-111  IC-121  IC-125  IC-127  IC-129  IC-131
  IC-132  IC-134  IC-135  IC-137  IC-138  IC-180  IC-182  IC-82
  EVENT-11 : (IC-17 AND IC-18)  PAR-1
  EVENT-5 : (IC-50 AND IC-51)  PAR-1
  EVENT-12 : (IC-57 AND IC-75 AND IC-81 AND IC-83)  PAR-1
  EVENT-13 : (IC-111 AND IC-121 AND IC-125 AND IC-127 AND IC-129)  PAR-1
  EVENT-14 : (IC-131 AND IC-132 AND IC-134 AND IC-135 AND IC-137 AND IC-138)
                                                                    PAR-1
  EVENT-15 : (IC-180 AND IC-182 AND IC-82)  PAR-1
 MIC-INT-CONST-CONTEXT
  MIC-1 : ∃1((1∈COMPTE)∧(NOMATR.1=NOMATR.PAR-1)∧(NOAGENCE.1=NOAGENCE.PAR-1)
          ∧(TYPCPTE.1=TYPCPTE.PAR-1))
  EVENT-7 : (MIC-1)  PAR-1
```

```
PROC-DESCRIPTION
 BEGIN
  Débiteur
  OBTENIR N DE SLDINTDB
  SI (N=∅) ALORS ALLER A Fin
  DESIGN R3={(1)|((1∈DBPRT)∧∀v(((v∈DBPRT)∧(DB.1≠DP.v)∧(NOMATR.1=NOMATR.v=
            NOMATR.N)∧(NOAGENCE.1=NOAGENCE.v=NOAGENCE.N)∧(TYPCPTE.1=
            TYPCPTE.v=TYPCPTE.N))⇒((DATEVAL.v<DATEVAL.1≤DT.N)∨(DATEVAL.1≤
            DT.N<DATEVAL.v))))}
  SI (R3=∅) ALORS ALLER A Stand
  OBTENIR DP1 DE R3
  DESIGN R4={(1)|((1∈TAUXBASE)∧(CDBASE.1=CDBASE.DP1)∧(DATEVAL.DP1=DATEVAL.1))}
  OBTENIR TX DE R4
  TX1-1 := MARGE1.DP1 + TAUXBASE.TX
  LIBERER R4
  DESIGN R4={(1)|((1∈TAUXBASE)∧(CDBASE.1 CDBASEM.DP1)∧(DATEVAL.DP1=DATEVAL.1))
  OBTENIR TX DE R4
  TX1-2 := MARGE2.DP1 + TAUXBASE.TX
  LIBERER R4
  OBTENIR DP2 DE R3
  DESIGN R4={(1)|((1∈TAUXBASE)∧(CDBASE.1=CDBASE.DP2)∧(DATEVAL.DP2=DATEVAL.1))}
  OBTENIR TX DE R4
  TX2-1 := MARGE1.DP2 + TAUXBASE.TX
  LIBERER R4
  DESIGN R4={(1)|((1∈TAUXBASE)∧(CDBASE.1=CDBASEM.DP2)∧(DATEVAL.DP2=DATEVAL.1))
  OBTENIR TX DE R4
  TX2-2 := MARGE2.DP2 + TAUXBASE.TX
  LIBERER R4
  OBTENIR DP3 DE R3
  DESIGN R4={(1)|((1 TAUXBASE)∈(CDBASE.1=CDBASE.DP3)∈(DATEVAL.DP3=DATEVAL.1))}
  OBTENIR TX DE R4
  TX3-1 := MARGE1.DP3 + TAUXBASE.TX
  LIBERER R4
  DESIGN R4={(1)|((1∈TAUXBASE)∧(CDBASE.1=CDBASEM.DP3)∧(DATEVAL.DP3=DATEVAL.1))
  OBTENIR TX DE R4
  TX3-2 := MARGE2.DP3 + TAUXBASE.TX
```

```
SI (TX1-1<TX1-2) ALORS (TX1-1=TX1-2)
SI (TX1-1≤TAUXDEB.DP1) ALORS (TX1-1=TAUXDEB.DP1)
SI (TX2-1<TX2-2) ALORS (TX2-1=TX2-2)
SI (TX2-1≤TAUXDEB.DP2) ALORS (TX2-1=TAUXDEB.DP2)
SI (TX3-1<TX3-2) ALORS (TX3-1=TX3-2)
SI (TX3-1≤TAUXDEB.DP3) ALORS (TX3-1=TAUXDEB.DP3)
SI (SLDJ.N≤LIMITE.DP1)
ALORS (INT.N=SLDJ.N*TX1-1)
SINON SI ((SLDJ.N>LIMITE.DP1)∧(TYPLIMIT.DP2=1)∧(SLDJ.N≤LIMITE.DP2))
      ALORS (INT.N=SLDJ.N*TX2-1)
      SINON SI ((SLDJ.N>LIMITE.DP1)∧(TYPLIMIT.DP2=1)∧(TYPLIMIT.DP3=1)
            ∧(SLDJ.N>LIMITE.DP2))
            ALORS (INT.N=SLDJ.N*TX3-1)
            SINON SI ((SLDJ.N>LIMITE.DP1)∧(TYPLIMIT.DP2=1)∧(TYPLIMIT.DP3≠1)
                  ∧(SLDJ.N>LIMITE.DP2))
                  ALORS (INT.N=(SLDJ.N+LIMITE.DP2)*TX3-1 -LIMITE.DP2*TX2-1)
                  SINON SI ((SLDJ.N>LIMITE.DP1)∧(TYPLIMIT.DP2≠1)
                        ∧(SLDJ.N≤LIMITE.DP2))
                        ALORS (INT.N=(SLDJ.N+LIMITE.DP1)*TX2-1
                              -LIMITE.DP1*TX1-1)
                        SINON SI ((SLDJ.N>LIMITE.DP1)∧(TYPLIMIT.DP2≠1)
                              (SLDJ.N>LIMITE.DP2)∧(TYPLIMIT.DP3=1))
                              ALORS (INT.N=SLDJ.N*TX3-1)
                              SINON SI ((SLDJ.N>LIMITE.DP1)∧(TYPLIMIT.DP2≠1)
                                    (SLDJ.N>LIMITE.DP2)∧(TYPLIMIT.DP3≠1)
                                    ALORS (INT.N=(SLDJ.N+LIMITE.DP2)*TX3-1
                                          +(LIMITE.DP1-LIMITE.DP2)*TX2-1
                                          -(LIMITE.DP1*TX1-1))

ALLER A Int
Stand
DESIGN R15={(1)|((1∈COMPTE)∧(NOMATR.1=NOMATR.N)∧(NOAGENCE.1=NOAGENCE.N)
          ∧(TYPCPTE.1=TYPCPTE.N))}
OBTENIR R15
SI (CDNUDEV.R15≠000) ALORS ALLER A Devise
DESIGN R16={(SIECPTE.1)|((1∈CLIENT)∧(NOMATR.1=NOMATR.N))}
OBTENIR R16
DESIGN R17={(m)|((m∈INTDBSRCT)∧∀1(((1∈INTDBSRCT)∧(m≠1)∧(SIECPTE.R16=
          SIECPTE.1=SIECPTE.m)∧(MARCHE.R15=MARCHE.1=MARCHE.m))
          ⇒((DATEVAL.1<DATEVAL.m≤DT.N)∨(DATEVAL.m≤DT.N<DATEVAL.1))))}
OBTENIR R17
ALLER A Base
Devise
DESIGN R17={(m)|((m∈INTDBDEV)∧∀1(((1∈INTDBDEV)∧(m≠1)∧(CDNUDEV.R15=CDNUDEV.1=
          CDNUDEV.m)∧(MARCHE.R15=MARCHE.1=MARCHE.m))
          ⇒((DATEVAL.1<DATEVAL.m≤DT.N)∨(DATEVAL.m≤DT.N<DATEVAL.1))))}
OBTENIR R17
```

```
   Base
   DESIGN R10={(1)|((1∈TAUXBASE)∧(CDBASE.1=CDBASE.R17)∧(DATEVAL.1=DATEVAL.R17))}
   OBTENIR R10
   TX1 := MARGE1.R17 + TAUXBASE.R10
   LIBERER R10
   DESIGN R10={(1)|((1∈TAUXBASE)∧(CDBASE.1=CDBASE.R17)∧(DATEVAL.1=DATEVAL.R17))}
   OBTENIR R11
   TX2 := MARGE2.R17 + TAUXBASE.R10
   SI (TX1≤TAUXDEB.R17) ALORS (TX1=TAUXDEB.R17) -
   SI (TX2≤TAUXDEB.R17) ALORS (TX2=TAUXDEB.R17)
   SI (SLDJ.N≤LIMITE.R17)
   ALORS (INT.N=SLDJ.N*TX1)
   SINON SI ((SLDJ.N>LIMITE.R17)∧(TYPLIMIT.R17=1))
         ALORS (INT.N=SLDJ.N*TX2)
         SINON SI ((SLDJ.N>LIMITE.R17)∧(TYPLIMIT.R17≠1))
               ALORS (INT.N=(SLDJ.N+LIMITE.R17)*TX2 -LIMITE.R17*TX1)
   Int
   MODIFIER N
   ALLER A Débiteur
   Fin
  END
 INDUCED-EVENT EVENT-16 PAR-1 SLDINTCR(NOMATR,NOAGENCE,TYPCPTE,DT,SLDJ,INT)
END   INTDEB


PROC   INTCRD   'calcul des intérêts créditeurs'
 TRIGGER  (EVENT-16)
 PARAMETERS
  EVENT-16  PAR-1  SLDINTCR(NOMATR,NOAGENCE,TYPCPTE,DT,SLDJ,INT)
 EXT-INT-CONST-CONTEXT
  EIC-2 : ∀1((1∈SLDINTCR)⇒∄m((m∈SLDINTCR)∧(m≠1)∧(DT.1=DT.m)))
  FORM-1  FORM-19  FORM-59  FORM-137  FORM-139  FORM-140
  EVENT-3 : (FORM-1 AND FORM-19 AND FORM-59)  PAR-1
  EVENT-8 : (FORM-137 AND FORM-139 AND FORM-140)  PAR-1
  EVENT-21 : (EIC-2)  PAR-1
 INT-INT-CONST-CONTEXT
  IC-17  IC-18  IC-50  IC-51  IC-56  IC-74  IC-79  IC-83  IC-110  IC-120
  IC-124  IC-126  IC-128  IC-130  IC-133  IC-136  IC-179  IC-181  IC-80
```

EVENT-11 : (IC-17 AND IC-18) PAR-1

EVENT-5 : (IC-50 AND IC-51) PAR-1

EVENT-12 : (IC-56 AND IC-74 AND IC-79 AND IC-83) PAR-1

EVENT-13 : (IC-110 AND IC-120 AND IC-124 AND IC-126 AND IC-128) PAR-1

EVENT-14 : (IC-130 AND IC-133 AND IC-136) PAR-1

EVENT-15 : (IC-179 AND IC-181 AND IC-80) PAR-1

MIC-INT-CONST-CONTEXT

MIC-1 : $\exists 1((1 \in \text{COMPTE}) \wedge (\text{NOMATR.1}=\text{NOMATR.PAR-1}) \wedge (\text{NOAGENCE.1}=\text{NOAGENCE.PAR-1}) \wedge (\text{TYPCPTE.1}=\text{TYPCPTE.PAR-1}))$

EVENT-7 : (MIC-1) PAR-1

PROC-DESCRIPTION

BEGIN

Créditeur

OBTENIR N DE SLDINTCR

SI (N=$\emptyset$) ALORS ALLER A Fin

DESIGN R3=$\{(1) \mid ((1 \in \text{CRDPRT}) \wedge \forall v(((v \in \text{CRDPRT}) \wedge (\text{CP.1} \neq \text{CP.v}) \wedge (\text{NOMATR.N}=\text{NOMATR.1}=\text{NOMATR.v}) \wedge (\text{NOAGENCE.N}=\text{NOAGENCE.1}=\text{NOAGENCE.v}) \wedge (\text{TYPCPTE.N}=\text{TYPCPTE.1}=\text{TYPCPTE.v})) \Rightarrow ((\text{DATEVAL.v} < \text{DATEVAL.1} \leq \text{DT.N}) \vee (\text{DATEVAL.1} \leq \text{DT.N} < \text{DATEVAL.v}))))\}$

SI (R3=$\emptyset$) ALORS ALLER A Stand

OBTENIR CR1 DE R3

DESIGN R4=$\{(1) \mid ((1 \in \text{TAUXBASE}) \wedge (\text{CDBASE.1}=\text{CDBASE.CR1}) \wedge (\text{DATEVAL.1}=\text{DATEVAL.CR1}))\}$

OBTENIR TX DE R4

TX1 := MARGE.CR1 + TAUXBASE.TX

LIBERER R4

OBTENIR CR2 DE R3

DESIGN R4=$\{(1) \mid ((1 \in \text{TAUXBASE}) \wedge (\text{CDBASE.1}=\text{CDBASE.CR2}) \wedge (\text{DATEVAL.1}=\text{DATEVAL.CR2}))\}$

OBTENIR TX DE R4

TX2 := MARGE.CR2 + TAUXBASE.TX

LIBERER R4

OBTENIR CR3 DE R3

DESIGN R4=$\{(1) \mid ((1 \in \text{TAUXBASE}) \wedge (\text{CDBASE.1}=\text{CDBASE.CR3}) \wedge (\text{DATEVAL.1}=\text{DATEVAL.CR3}))\}$

OBTENIR TX DE R4

TX3 := MARGE.CR3 + TAUXBASE.TX

LIBERER R3

LIBERER R4

```
SI (SLDJ.N≤LIMITE.CR1)
ALORS (INT.N=SLDJ.N*TX1)
SINON SI ((SLDJ.N>LIMITE.CR1)∧(TYPLIMIT.CR2=1)∧(SLDJ.N≤LIMITE.CR2))
      ALORS (INT.N=SLDJ.N*TX2)
      SINON SI ((SLDJ.N>LIMITE.CR1)∧(TYPLIMIT.CR2=1)∧(SLDJ.N>LIMITE.CR2)
                ∧(TYPLIMIT.CR3=1))
            ALORS (INT.N=SLDJ.N*TX3)
            SINON SI ((SLDJ.N>LIMITE.CR1)∧(TYPLIMIT.CR2=1)
                      ∧(SLDJ.N>LIMITE.CR2)∧(TYPLIMIT.CR3≠1))
                  ALORS (INT.N=(SLDJ.N -LIMITE.CR2)*TX3+LIMITE.CR2*TX2)
                  SINON SI ((SLDJ.N>LIMITE.CR1)∧(TYPLIMIT.CR2≠1)
                            ∧(SLDJ.N≤LIMITE.CR2))
                        ALORS (INT.N=(SLDJ.N -LIMITE.CR1)* TX2
                              +LIMITE.CR2*TX1)
                        SINON SI ((SLDJ.N>LIMITE.CR1)∧(TYPLIMIT.CR2≠1)
                                  ∧(SLDJ.N>LIMITE.CR2)∧(TYPLIMIT.CR3=1))
                              ALORS (INT.N=SLDJ.N*TX3)
                              SINON SI ((SLDJ.N>LIMITE.CR1)∧(TYPLIMIT.CR2≠1)
                                        ∧(SLDJ.N>LIMITE.CR2)∧(TYPLIMIT.CR3≠1))
                                    ALORS (INT.N=(SLDJ.N -LIMITE.CR2)*TX3
                                          +(LIMITE.CR2 -LIMITE.CR1)* TX2
                                          +LIMITE.CR1*TX1)

ALLER A Int

Stand

DESIGN R5={(1)|((1∈COMPTE)∧(NOMATR.1=NOMATR.N)∧(NOAGENCE.1=NOAGENCE.N)
          ∧(TYPCPTE.1=TYPCPTE.N))}

OBTENIR R5

SI (CDNUDEV.R5≠'000') ALORS ALLER A Devise

DESIGN R6={(SIECPTE.1)|((1∈CLIENT)∧(NOMATR.1=NOMATR.N))}

OBTENIR R6

DESIGN R7={(m)|((m∈INTCRSRCT)∧∀1(((1∈INTCRSRCT)∧(m≠1)∧(SIECPTE.R6=
          SIECPTE.m=SIECPTE.1)∧(MARCHE.R5=MARCHE.m=MARCHE.1))⇒((DATEVAL.1<
          DATEVAL.m≤DT.N)∨(DATEVAL.m≤DT.N<DATEVAL.1))))}

OBTENIR R7

ALLER A Base

Devise

DESIGN R7={(m)|((m∈INTCRDEV)∧∀1(((1∈INTCRDEV)∧(m≠1)∧(CDNUDEV.R5=CDNUDEV.m=
          CDNUDEV.1)∧(MARCHE.R5=MARCHE.m=MARCHE.1))⇒((DATEVAL.1≤DATEVAL.m≤
          DT.N)∨(DATEVAL.m≤DT.N<DATEVAL.1))))}

OBTENIR R7

Base

DESIGN R4={(1)|((1∈TAUXBASE)∧(CDBASE.1=CDBASE1.R7)∧(DATEVAL.1=DATEVAL.R7))}

OBTENIR R4

TX1 := MARGE1.R7 + TAUXBASE.R4
```

```
   LIBERER R4
   DESIGN R4={(1)|((1∈TAUXBASE)∧(CDBASE.1=CDBASE2.R7)∧(DATEVAL.1=DATEVAL.R7))}
   OBTENIR R4
   TX2 := MARGE2.R7 + TAUXBASE.R4
   LIBERER R4
   SI (SLDJ.N≤LIMITE.R7)
   ALORS (INT.N=SLDJ.N*TX1)
   SINON SI ((SLDJ.N>LIMITE.R7)∧(TYPLIMIT.R7=1))
         ALORS (INT.N=SLDJ.N*TX2)
         SINON SI ((SLDJ.N>LIMITE.R7)∧(TYPLIMIT.R7≠1))
               ALORS (INT.N=(SJDJ.N -LIMITE.R7)*TX2+LIMITE.R7*TX1)
   LIBERER R7
   Int
   MODIFIER N
   ALLER A Créditeur
   Fin
  END
 INDUCED-EVENT  EVENT-17 PAR-1 SLDINTCR(NOMATR,NOAGENCE,TYPCPTE,DT,SLDJ,INT)
                         PAR-2 SLDINTDB(NOMATR,NOAGENCE,TYPCPTE,DT,SLDJ,INT)
                EVENT-2  PAR-3 (NOMATR,NOAGENCE,TYPCPTE,DTINT)
END      INTCRD


PROC   MAJINT   'mise à jour de la relation 'INTERET' '
 TRIGGER  (EVENT-17)
 PARAMETERS
  EVENT-17 PAR-1 SLDINTCR(NOMATR,NOAGENCE,TYPCPTE,DT,SLDJ,INT)
           PAR-2 SLDINTDB(NOMATR,NOAGENCE,TYPCPTE,DT,SLDJ,INT)
  EVENT-2  PAR-3 (NOMATR,NOAGENCE,TYPCPTE,DTINT)
 EXT-INT-CONST-CONTEXT
  EIC-3 : ∀1((1∈SLDINTCR)⇒((NOMATR.1=NOMATR.PAR-3)∧(NOAGENCE.1=NOAGENCE.PAR-3)
          ∧(TYPCPTE.1=TYPCPTE.PAR-3)))
  EIC-4 : ∀1((1∈SLDINTDB)⇒((NOMATR.1=NOMATR.PAR-3)∧(NOAGENCE.1=NOAGENCE.PAR-3)
          ∧(TYPCPTE.1=TYPCPTE.PAR-3)))
  FORM-1  FORM-19  FORM-59  FORM-137  FORM-139  FORM-140
  EVENT-3 : (FORM-1 AND FORM-19 AND FORM-59)  PAR-1, PAR-2, PAR-3
  EVENT-8 : (FORM-137 AND FORM-139 AND FORM-140)  PAR-1, PAR-2, PAR-3
  EVENT-19 : (EIC-3 OR EIC-4)  PAR-1, PAR-2, PAR-3
```

INT-INT-CONST-CONTEXT

IC-176 IC-177 IC-50 IC-51

EVENT-5 : (IC-50 AND IC-51) PAR-1, PAR-2, PAR-3

EVENT-18 : (IC-176 AND IC-177) PAR-3

MIC-INT-CONST-CONTEXT

MIC-1 : $\exists 1((1 \in \text{COMPTE}) \wedge (\text{NOMATR.1}=\text{NOMATR.PAR-3}) \wedge (\text{NOAGENCE.1}=\text{NOAGENCE.PAR-3}) \wedge (\text{TYPCPTE.1}=\text{TYPCPTE.PAR-3}))$

EVENT-7 : (MIC-1) PAR-3

PROC-DESCRIPTION

BEGIN

DESIGN R10=$\{(1) \mid ((1 \in \text{INTERET}) \wedge \forall v(((v \in \text{INTERET}) \wedge (v \neq 1) \wedge (\text{NOMATR.1}=\text{NOMATR.v}=\text{NOMATR.PAR-3}) \wedge (\text{NOAGENCE.1}=\text{NOAGENCE.v}=\text{NOAGENCE.PAR-3}) \wedge (\text{TYPCPTE.1}=\text{TYPCPTE.v}=\text{TYPCPTE.PAR-3})) \Rightarrow (\text{DTDRINT.1}>\text{DTDRINT.v})))\}$

OBTENIR R10

INTC:= INTCUM.R10 + SOM(((DT,INT)SLDINTCR,INT)|(DTDRINT.R10$\leq$DT$\leq$(DTINT.PAR-1 -1))) + SOM(((DT,INT)SLDINTDB,INT)|(DTDRINT.R10$\leq$DT$\leq$(DTINT.PAR-1 -1)))

DESIGN R11=$\{(1) \mid ((1 \in \text{SLDINTDB}) \wedge (\text{DT.1}=(\text{DTINT.PAR-3 -1}))\}$

OBTENIR R11

SI (R11=$\emptyset$) ALORS DESIGN R12=$\{(1) \mid ((1 \in \text{SLDINTCR}) \wedge (\text{DT.1}=(\text{DTINT.PAR-3 -1}))\}$

OBTENIR R12

SD := SLDJ.R12

SINON SD := SLDJ.R11

AJOUTER (PAR-3,SD,INTC) A INTERET

LIBERER R10,R11,R12, SLDINTCR,SLDINTDB

END

END MAJINT

ANNEXE IV

Rappels sur les relations binaires.

I. Relation binaire.

Soient deux ensembles A et B : une relation binaire R sur A et B, notée R(A,B), est une partie du produit cartésien AxB.

II. Application et fonction.

-A une relation binaire R(A,B), on peut associer une application Γ de A dans B qui à $a \in A$ fait correspondre un, plusieurs ou aucun $b \in B$.

Notation: $A \xrightarrow{R} B$: $b = \Gamma(a) \Leftrightarrow (a,b) \in R$

Par convention, on note R pour Γ.

-Le domaine de R: $DOM(R) = \{a \in A \mid \exists b \in B \text{ tq } (a,b) \in R\}$

-L'image de R: $IM(R) = \{b \in B \mid \exists a \in A \text{ tq } (a,b) \in R\}$

-Fonction de A dans B: $\forall a \in DOM(R) : \exists! b \in B \text{ tq } (a,b) \in R$

= pour tout a appartenant au domaine de R, il existe un et un seul b appartenant à B tel que le couple (a,b) appartienne à R.

Une application qui n'est pas une fonction sera dite application multivoque.

Notation pour une fonction: $A \xrightarrow{R} B$ et $R^{-1}(B,A)$ est multivoque.

III. Application totale ou partielle.

-Totale : le domaine de R est égal à A $\equiv DOM(R) = A$

notation : $A \xrightarrow{R} B$

-Partielle : le domaine de R est inclus dans A

-Si l'inverse $R^{-1}(B,A)$ d'une application R(A,B) est totale, alors R est dite application de A sur B.

Notation: $A \xrightarrow{R} B$

IV. Injection, surjection, bijection.

-Injection: $\forall a, a' \in A$ et $a \neq a' : R(a) \neq R(a')$

-Surjection: $\forall b \in B : \exists a \in A \text{ tq } b = R(a)$.

-Bijection: R est en même temps injective et surjective.

BIBLIOGRAPHIE

BIBLIOGRAPHIE

1. BENCI E., BODART F., BOGAERT H. et CABANES A., Concepts for the design of a conceptual schema, 1975
2. CABANES A., Base de données, Cours FNDP, Institut d'informatique, Namur, 1974-1975
3. CABANES A., Base de données, matières approfondies, Cours FNDP, Institut d'informatique, Namur, 1975-1976
4. BODART F., Introduction à l'analyse fonctionnelle des systèmes informatiques de gestion, Cours FNDP, Institut d'informatique, Namur, 1975-1976
5. FLORY A. et KOULOUMDJIAN J., Sémantique et structure de données, Université de Lyon I, Laboratoire informatique, 1975
6. FLORY A., Algorithmes d'aide à la conception de système d'information, Université Claude Bernard de Lyon, 1976
7. DAMBRINE M., L'organisation des données d'un système, INFORSID- Groupe II, Séminaire de Pont-à-Mousson, 17 et 18 mai 1976
8. TARDIEU H., HECKENROTH H., NANCI D., GAMBACH M., LEMOIGNE J-L., Une méthodologie d'analyse et de conception d'une Base de Données, 1975
9. FERNANDEZ E. et SUMMERS R., Integrity aspects of a shared Data Base, IBM Los Angeles, Scientific Center
10. WEBER H., Semantic model to specify contraints on a relationnel Data Base, Massachusetts Institute of Technology, Project MAC, July 1975
11. WANG C-P., Parametrization of information system applications, IBM Research Labarotary San Jose California, April 1973
12. CHAMBERLIN D. and BOYCE R., Sequel: a structured english query language, IBM Research Laboratory San Jose California, 1974
13. PATIGNY R., "M.I.S." bancaire et Banques de Données: certains aspects, mémoire FNDP Namur, Institut d'informatique, 1973-1974
14. PIN-SHAN CHEN P., The entity-relationship model- toward a unified view of data, Massachusetts Institute of technology, 1975

15. DELOBEL C., Les systèmes de Base de données, Cours pour l'Ecole d'Eté de l'AFCET, 1975

16. DE LAMAZIERE A., Conception de systèmes: comment construit-on pratiquement une Base de Données?, Zéro-un-informatique, n°97, Mars 1976

17. RUFF A. et BELAN P., L'ordinateur de la banque, Editions d'informatique, 1974

18. DECOSTER E., Initiation bancaire: étude analytique de la banque et notions générales de finances

19. BODART F. et BRENY A., Description de l'I.D.S., Institut d'informatique, 1972.