UNIVERSITÉ DE NEUCHATEL
FACULTÉ DE DROIT ET DES SCIENCES ÉCONOMIQUES

Méthode d'approche pour la conception d'une banque de données

THÈSE

PRÉSENTÉE A LA FACULTÉ DE DROIT ET DES SCIENCES ÉCONOMIQUES
POUR OBTENIR LE GRADE DE DOCTEUR ÈS SCIENCES ÉCONOMIQUES

PAR

ERNEST BULA

IMPRIMERIE WILLY BLASER SA, TRIMBACH
1978

Monsieur Ernest BULA est autorisé à imprimer sa thèse de doctorat en sciences économiques intitulée "Méthode d'approche pour la conception d'une banque de données".

Il assume seul la responsabilité des opinions énoncées.

Neuchâtel, 22 février 1978

Le doyen
de la Faculté de droit
et des sciences économiques

Michel Rousson

avec Doris

SOMMAIRE

4. METHODE D'APPROCHE PROPOSEE

5. MODELISATION DU SYSTEME DE GESTION

6. ANALYSE SEMANTIQUE DU MONDE REEL

7. CONCEPTION DE LA STRUCTURE LOGIQUE

INTRODUCTION

Lors de la conception d'un système informatique, le concepteur se trouve entre autre confronté aux problèmes de la gestion des données. L'utilisation d'un SGBD est-elle la solution ? Comment répondre à cette question ? Une réponse affirmative ouvre un nouvel éventail de problèmes:

- quel SGBD utiliser ?
- conception de la banque de données
- quels critères considérer ?
- influences sur les méthodes d'approche utilisées lors de la conception des applications
- une flexibilité suffisante est-elle garantie ?
- contraintes économiques.

Face à cette situation incertaine, le concepteur cherchera d'abord à s'informer dans la littérature spécialisée et/ou chez le constructeur de SGBD. En général, et l'expérience pratique le prouve, il ne trouvera que des réponses partielles aux différents problèmes soulevés.

La difficulté principale à laquelle le concepteur se trouvera confronté est d'ordre méthodologique. Quels sont les critères à considérer et selon quel schéma doit s'effectuer la conception, afin de respecter les objectifs et les contraintes découlant des caractéristiques des applications concernées.

Le but de notre étude consiste à développer une méthode d'approche pour la conception d'une banque de données dans le domaine de l'informatique de gestion, soit:

- définir les critères à prendre en considération
- déterminer les activités à effectuer et proposer des méthodes adéquates
- développer un schéma du déroulement des opérations
- proposer des solutions en fonction des SGBD auxquels le concepteur peut se référer
- donner une réponse aux problèmes que posent

l'utilisation d'un SGBD
- intégrer le concept banque de données dans un ensemble plus vaste, le système d'information.

Le nombre, la diversité et la complexité des problèmes posés sont énormes, néanmoins nous tenteront de proposer des solutions en tant que ligne de conduite à suivre lors de la conception d'une banque de données.

Le plan de notre travail ne suivra pas l'ordre des questions et ceci principalement pour des raisons de clareté et de présentation. Nous distinguerons d'une part les critères à prendre en considération et d'autre part l'ordre dans lequel ils seront pris en compte.

Dans un premier temps, nous donnerons un bref aperçu des notions utilisées en rapport avec le concept banque de données (chapitre 1).

L'objectif du chapitre 2 consistera à décrire les critères influençant la conception. Nous les analyserons au cours des différentes étapes de notre méthode d'approche.

Quant au chapitre 3, il sera voué aux problèmes fondamentaux posés par la réalisation d'un système informatique. Nous insisterons d'abord sur la nécessité d'établir un modèle de l'entreprise pour passer ensuite à l'élaboration des objectifs auxquels une méthode d'approche devrait satisfaire. Un rapide survol des méthodes existantes servira d'introduction au chapitre suivant.

La présentation de la méthode d'approche que nous proposons fera l'objet du chapitre 4, au cours duquel nous définirons les concepts de base et les éléments

de notre méthode. Nous proposerons une chronologie de réalisation en insistant sur les interférences entre les activités des différents éléments de notre méthode d'approche.

L'objectif du chapitre 5 consistera à présenter une possibilité de modélisation du système entreprise et de préparer ainsi le terrain à la conception de la banque de données.

La première phase de la méthode d'approche fera l'objet du chapitre 6. Après avoir abordé les problèmes relatifs à la saisie des informations en tant qu'interprétation du monde réel considéré, nous passerons à une activité très importante: l'analyse sémantique. Une étude critique des modèles de données permettra de déterminer une méthode de représentation du monde réel conforme entre autre aux contraintes de flexibilité et d'indépendance par rapport à un SGBD existant.

Après avoir décrit les problèmes relatifs à l'intégrité des données, nous étudierons les résultats de l'analyse sémantique, afin de déterminer le type de relation associant les éléments du modèle de données. Nous démontrerons également l'indépendance de la structure logique ainsi élaborée (chapitre7).

La phase suivante occupe une place importante dans notre méthode d'approche: elle concerne les chemins d'accès logiques. L'objectif principal du chapitre 8 consistera à décrire les caractéristiques des différents besoins en information en termes d'accès aux éléments de la structure logique.

Le passage à la réalisation physique implique en premier lieu un choix quant au SGBD à utiliser.

Une fois ce choix effectué, une analyse détaillée des caractéristiques de ce SGBD permettra la transposition de la structure logique élaborée précédemment en une structure conforme au langage de description du SGBD. Le modèle qui en résulte sera confronté aux contraintes découlant des chemins d'accès logiques et aux caractéristiques de la structure physique du SGBD, afin d'obtenir une banque de données optimale quant à son exploitation. Nous aborderons brièvement les problèmes relatifs à la réalisation des programmes d'application (chapitre 9).

L'implantation et l'exploitation d'une banque de données soulèvent des problèmes spécifiques que nous analyserons au cours du chapitre 10.

Chapitre 1

Le concept banque de données - Définitions

1.1. TERMINOLOGIE DE BASE

L'objectif de ce chapitre consiste à définir d'une part la terminologie utilisée en rapport avec le concept banque de données et ensuite de clarifier les différents éléments d'un système de gestion d'une banque de données (SGBD).

La première constatation que l'on peut faire en étudiant la littérature spécialisée dans ce domaine est la multitude des termes utilisés. Des tentatives de normalisation ont été effectuées, mais aucune ne fit l'unanimité. La plus connue, la proposition du Data Base Task Group du Codasyl [1.1], n'a pas réussi à s'imposer, certaines critiques ayant conduit à d'autres normes [1.2].

1.1.1. Le concept banque de données

a) Banque de données

Il existe plusieurs définitions de ce terme. Celle proposée par Gibert en [1.3] stipule qu'une banque de données représente un ensemble d'information archivé sur des mémoires accessibles à l'ordinateur pour une ou plusieurs applications déterminées. Cette définition est à notre avis trop générale, car elle n'implique pas les notions de:

- ensemble structuré
- communauté à plusieurs utilisateurs
- accès simultanés par différents utilisateurs

et ne fait pas de distinctions nettes avec les méthodes conventionnelles d'organisation des données dont les caractéristiques principales se résument ainsi:

- création de fichiers spécifiques à des programmes déterminés, leur organisation étant fixée en fonction des caractéristiques de traitement des programmes concernés

- le même fichier peut être utilisé par plusieurs programmes, mais un tri s'avère en général nécessaire puisqu'un tel fichier est organisé selon un seul critère
- chaque programme spécifie la définition et la structure des données du fichier utilisé ainsi que la méthode d'accès.

Dans le cadre de notre étude [1.4], nous entendons par banque de données

> un ensemble structuré de données enregistré sur des supports accessibles par l'ordinateur pour satisfaire simultanément à plusieurs utilisateurs, de manière sélective et en temps opportun.

b) Système de gestion d'une banque de données

Cette notion diffère du concept banque de données défini ci-dessus [1.5] en ce sens qu'un SGBD forme un ensemble de procédures

- de description
- d'accès
- de mise à jour
- de réalisation d'association entre données
- de maintien de l'intégrité
- de sécurité d'exploitation

d'une banque de données.

Un SGBD peut être considéré comme un système d'exploitation évolué traduisant les requêtes des utilisateurs en primitives d'accès du système d'exploitation. Un système informatique n'implique pas forcément l'existence d'un SGBD et d'une banque de données, il peut se baser uniquement sur des fichiers à organisation conventionnelle (fig. 1.1).

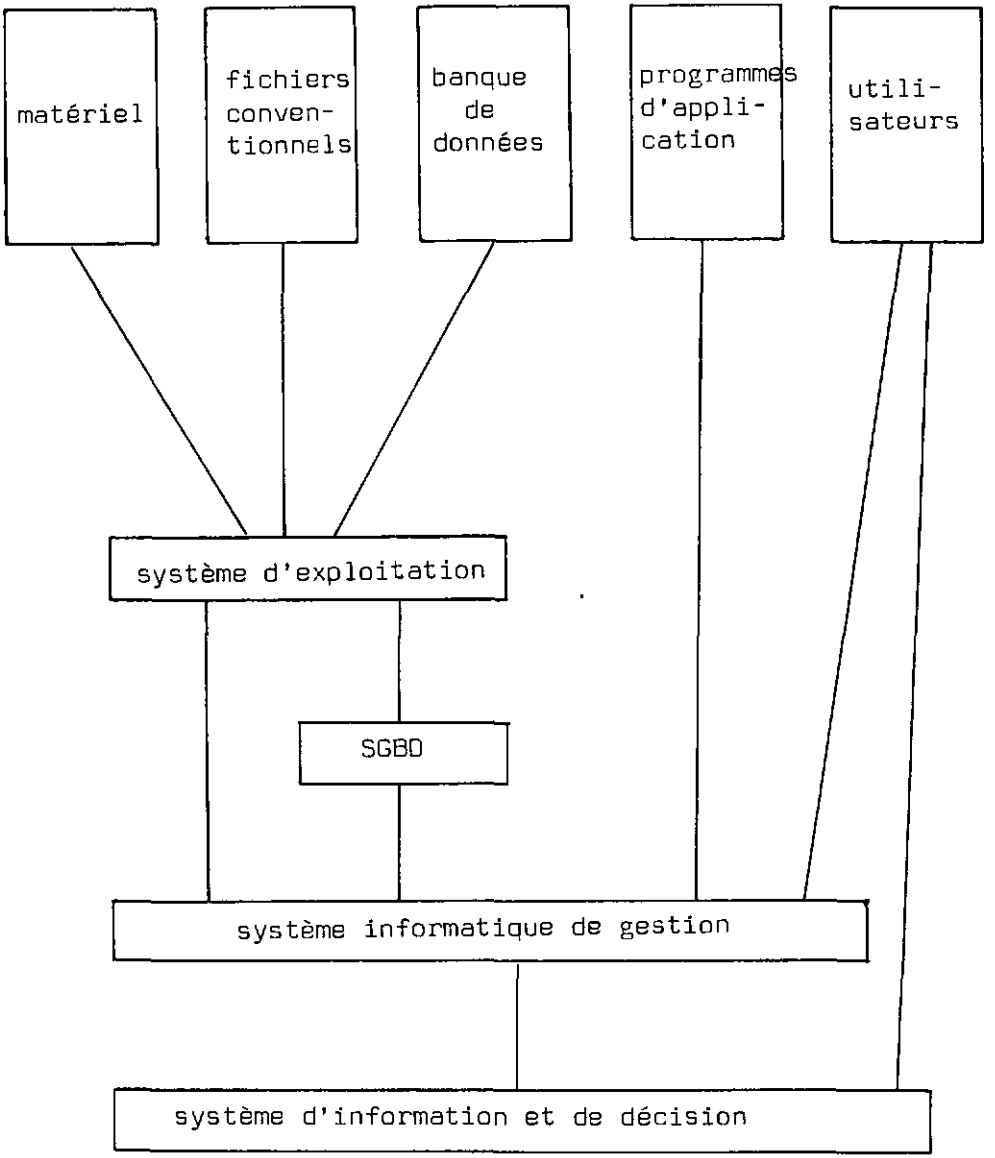


Fig. 1.1 - Hiérarchie des systèmes

c) La structure des données

Nous proposons une distinction des termes utilisés au niveau logique de ceux employés au niveau physique. Nous nous limiterons à une définition très générale, les différentes interprétations seront présentées aux chapitres 2.2 et 2.3 (fig. 1.2).

c1 Niveau logique

Une donnée est un ensemble de signes qui peuvent être des chiffres ou tout autre caractère.

Quant au terme information, il englobe la notion de signification allouée à ces données. En d'autre terme il représente leurs utilisations dans un but déterminé. Les règles qui relient entre elles les données se dénomme la syntaxe, alors que la signification de cette liaison s'appelle la sémantique. La plus petite unité sémantique est le constituant qui se compose du couple donnée et de l'interprétation de cette donnée [1.6], les termes item et attribut étant des dénominations synonymes.

L'objectif primaire de la création d'une banque de données est la représentation d'une interprétation du monde réel de l'entreprise, dont le but consiste à définir les objets ou événements, les entités, manipulés. En fonction des problèmes que doit résoudre la banque de données, par exemple la gestion du personnel ou de la production, les entités concernées forment un ensemble de constituants ou d'attributs.

Cette collection d'information structurée et formalisée se dénomme le modèle conceptuel, la structure logique ou fonctionnelle [1.7]. Elle implique une définition non ambigue et sémantiquement non redondante des entités, des constituants et des contraintes d'intégrité.

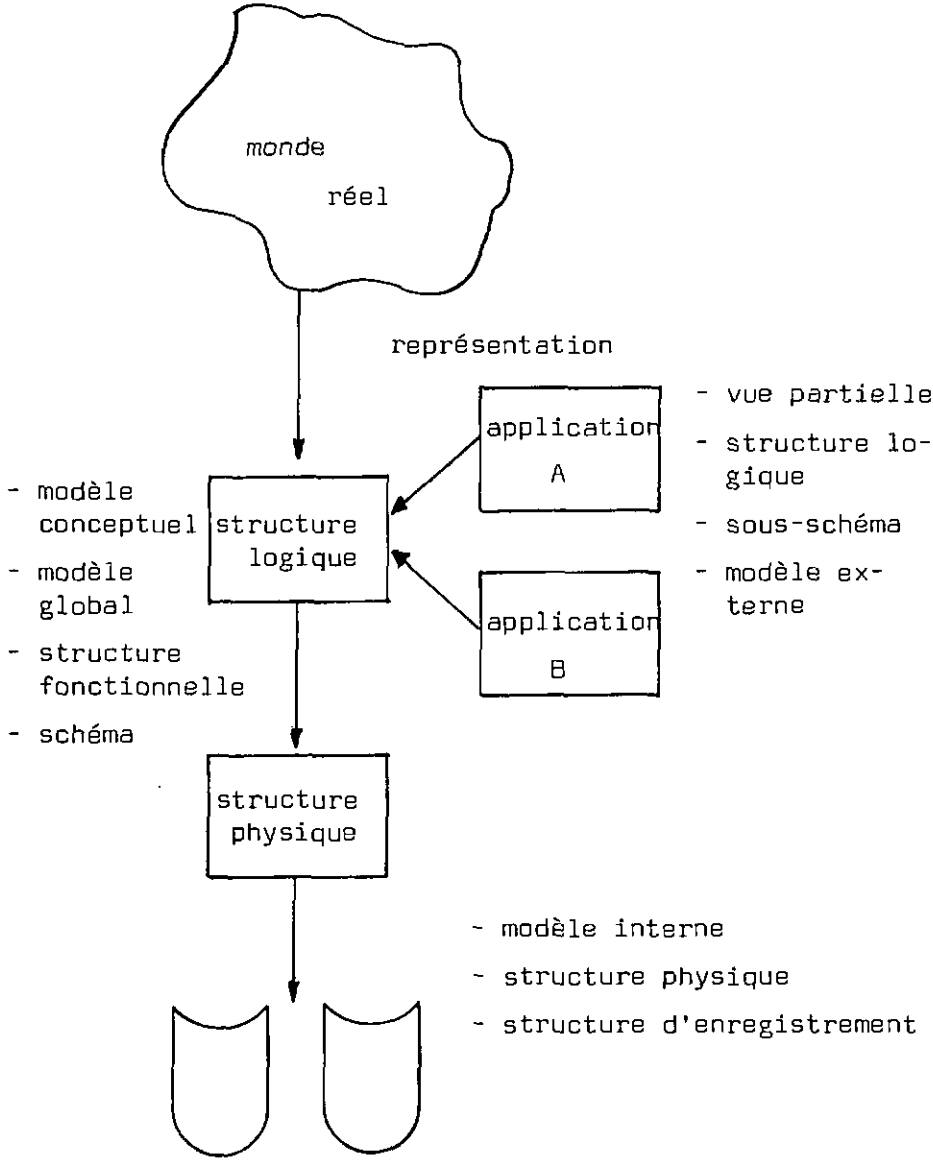


Fig. 1.2 - Concept banque de données

Comme le précise notre définition, la banque de données est commune à plusieurs utilisateurs, mais chacun d'eux n'a en fait pas besoin de toutes les informations enregistrées. Le sous-ensemble des entités concerné par une application donnée, la vue partielle, se dénomme également modèle externe ou structure logique [1.7].

c2 Niveau physique

L'enregistrement des données sur des supports physiques à grande capacité s'effectue selon les paramètres définis par le modèle interne, ce terme étant synonyme de structure physique.

Les entités décrites par la structure logique seront enregistrées sous forme d'article, ensemble structuré de constituants, alloué à un espace physique appelé le bloc ou enregistrement physique par opposition à l'enregistrement logique.

Une représentation de la structure des informations en trois niveaux est généralement admise [1.8], à savoir:

1. le modèle conceptuel
2. le modèle externe
3. le modèle interne.

1.2. LES OBJECTIFS D'UN SGBD

Nous nous limiterons à ne citer que les objectifs principaux du concept SGBD, la question étant étudiée de long en large par des organismes de normalisation [1.9]. Une analyse moins détaillée se trouve dans les ouvrages cités en [1.10].

a) Gestion centralisée des données

Dans un environnement banque de données, toutes les fonctions (programmes d'application, langage d'interrogation,

utilitaires) transmettent au SGBD leurs demandes d'accès en vue de manipuler les données.

b) Minimiser les redondances

Une gestion centralisée permet de réduire les redondances au niveau de l'enregistrement des données, ce qui facilite les opérations de mise à jour. Du point de vue utilisation l'extraction des données s'effectue plus simplement. Dans le cas des organisations conventionnelles, ce genre d'opération se complique singulièrement, car les données concernées sont réparties sur des fichiers indépendants. La redondance pose des problèmes quant à l'actualisation des données.

c) Partage des données

Les données enregistrées doivent être accessibles à plusieurs utilisateurs et selon des critères différents. Un inconvénient inhérent au partage des données, la place importante occupée par

- le maintien de l'intégrité
- les droits d'accès
- les phénomènes d'interblocage.

d) Faciliter l'utilisation

Des langages de description et de manipulation adaptés aux différents types d'utilisateur ne limitent pas l'utilisation de la banque de données à un petit groupe de spécialistes mais facilitent l'accès à d'autres utilisateurs.

e) Flexibilité et adaptabilité

La structure logique et physique doit pouvoir évoluer en fonction des nouveaux besoins et des modifications dans la représentation du monde réel considéré, sans impliquer une remise en cause du modèle et sans exiger

une adaptation des programmes d'application existants.

f) L'intégrité

La vérification du respect des contraintes d'intégrité des données à enregistrer doit s'effectuer sous le contrôle du SGBD afin de

- garantir la qualité des données
- éviter la dégradation des données
- éviter les pertes de données
- protéger les données contre des manipulations non autorisées
- maintenir l'existence de la sémantique.

Dans les systèmes conventionnels, cette fonction est du ressort de chaque programme d'application. Le maintien de l'intégrité ne s'obtient que très difficilement.

g) Augmenter le degré d'actualité

La gestion centralisée garantit un degré d'actualité supérieur, puisque la redondance au niveau de l'enregistrement peut se réduire à un strict minimum, la mise à jour d'une entité s'effectuant par un seul programme.

h) Différencier entre manipulation et traitement des données

Les opérations d'accès et de mise à jour s'exécutent à l'aide d'un langage spécifique, le langage de manipulation de données (LMD ou DML en anglais) qui peut s'intégrer à un langage hôte du type Cobol ou exister en tant que langage indépendant. Le traitement des données s'effectue par un langage de programmation classique.

i) L'indépendance des données

On entend par là indépendance entre les programmes et les données. Ce concept implique que toute modification apportée à l'un de ces deux éléments n'ait pas d'influence sur l'autre. Dans les systèmes conventionnels, la structure des données est spécifique à chaque programme d'application, par conséquent, la dépendance entre données et programme devient complète.

Un des principaux objectifs de tout SGBD est de promouvoir un degré d'indépendance élevé. Il existe différent degré d'indépendance [1.11]:

- l'indépendance physique a comme conséquence qu'une modification de la structure d'enregistrement n'affecte les programmes d'application
- l'indépendance logique signifie qu'un changement au niveau de la structure physique n'implique pas une adaptation de la structure logique
- l'indépendance d'accès n'exige pas la connaissance des chemins d'accès physiques aux données en vue d'extraire ou de mettre à jour les entités
- l'indépendance de structure signifie que l'utilisateur ne doit pas nécessairement connaître la structure logique, le système lui indiquant les éléments du modèle en mode interactif. Dans ce cas l'on parle du principe des menus.

Les objectifs ci-dessus ne doivent pas être confondus avec les notions telles que:

- autorisation d'accès (privacy)
- phénomène d'interblocage
- sécurité (journalisation, reprise)
- partage des ressources
- mesures en vue de garantir la stabilité.

1.3. ARCHITECTURE D'UN SGBD

Le schéma de la figure 1.3 représente les principaux éléments d'un SGBD, alors que la figure 1.4 a comme objectif la spécification des flux d'information entre ces éléments.

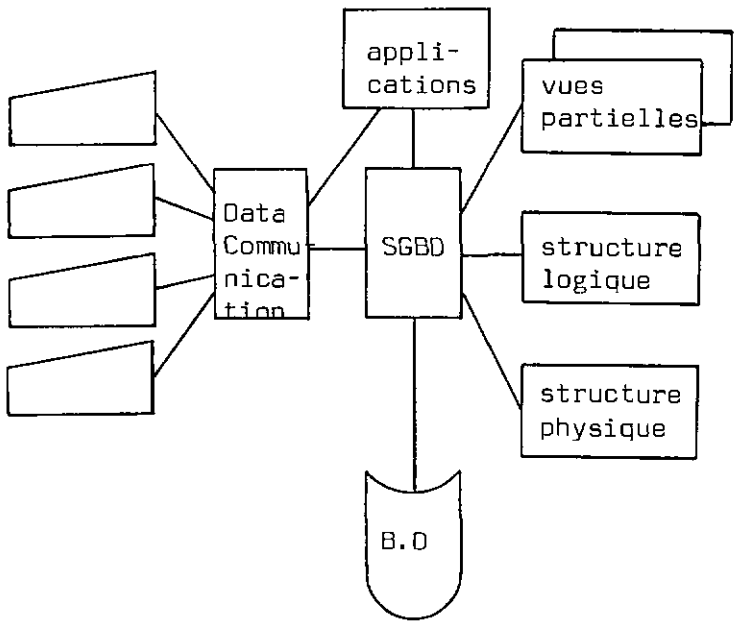


Fig. 1.3 - Principaux éléments d'un SGBD

La séquence des événements se résume de la manière suivante:

1. Le programme d'application transmet une requête à l'aide du langage de manipulation.
2. Le système vérifie la requête.
3. Il détermine les éléments de la structure logique concernée, vérifie les droits d'accès et le respect des contraintes d'intégrité.

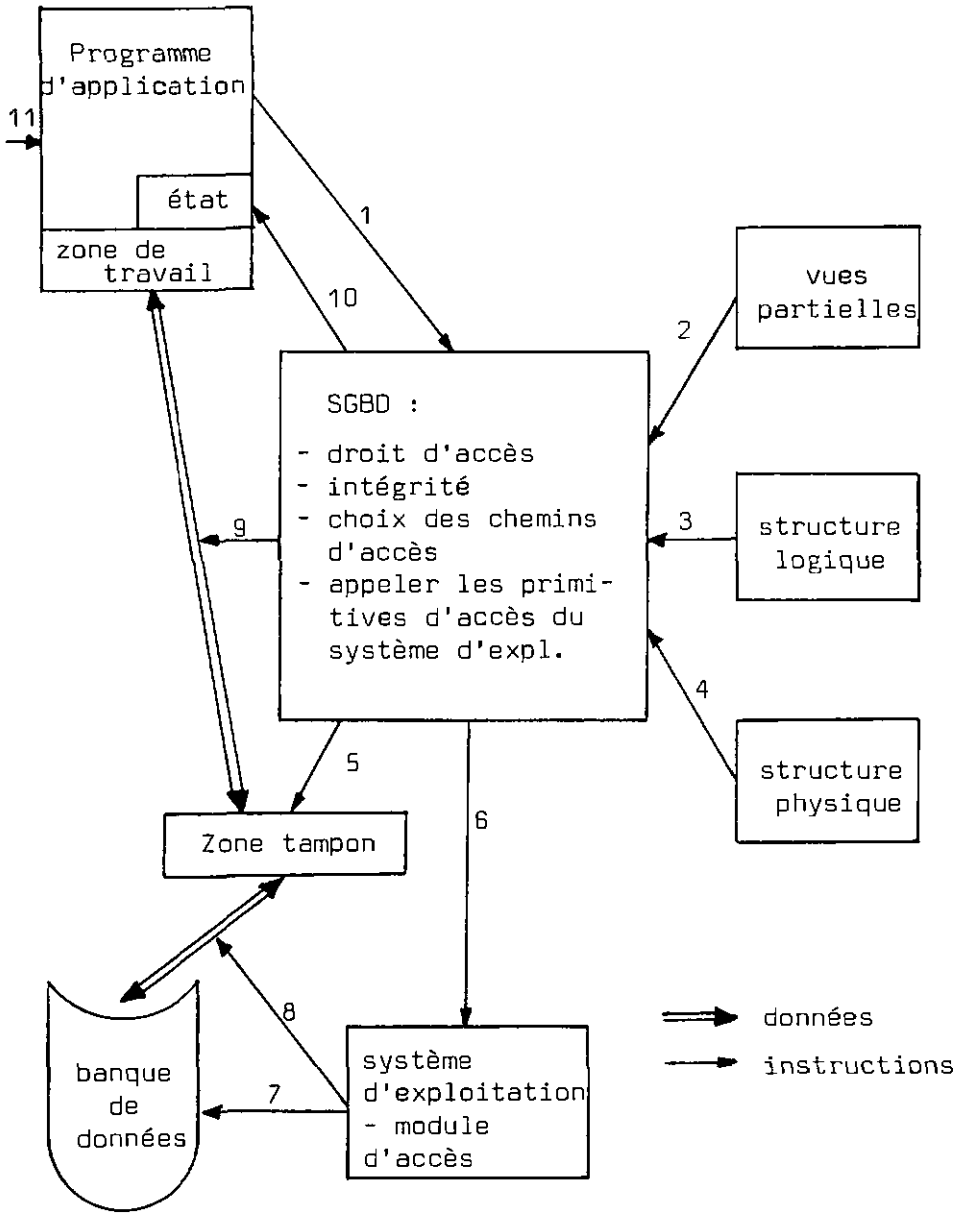


Fig. 1.4 - Principes de fonctionnement [1.12]

4. Le système analyse les caractéristiques de la structure physique et choisit le chemin d'accès optimal.
5. Il vérifie si les données recherchées ne figurent pas dans la zone buffer, si nécessaire il exécute les opérations 6, 7 et 8.
6. Le système d'exploitation est chargé de transférer les enregistrements physiques dont les coordonnées logiques lui sont transmises par le SGBD.
7. Le système d'exploitation détermine l'adresse physique des enregistrements concernés.
8. Il transfère ces enregistrements dans la zone buffer du SGBD.
9. En fonction de la vue partielle décrite par le requérant, le SGBD transfère les données dans la zone de travail du programme d'application.
10. Le SGBD transfère au programme des indications quant à l'exécution de la requête.
11. Le programme peut disposer des données et continuer son travail.

REFERENCES

[1.1] Codasyl, "Data Base Task Group Report".

[1.2] Guide/Share, "Data Base Management System ..."
Le modèle relationnel de CODD.
Ansi/Sparc, "Report".

[1.3] GIBERT A., "Les Banques de Données", Tome 1, p. 12.
Une définition du même type est proposée par DATE C.J., dans "Introduction ..." p. 1 ss.

[1.4] Définition proche de celles proposées par
MERTEN H., "Datenbankorganisation", p. 16.
YVON P.J., SEMIN C., "Comment concevoir un système ...", p. 87.
MARTIN J., "Computer Data Base Organization", pp. 13 et 19.

[1.5] Cette distinction est également proposée par
GIBERT A., cité en [1.3] p. 1.2.
MERTEN H., cité en [1.4] p. 16 ss.

[1.6] KIRSCH W., "Entscheidungsprozesse", p. 78 ss.

[1.7] Terme utilisé par le Club Banque de Données dans "Rapport du Groupe ...", p. I.5 ss.

[1.8] Ansi/Sparc, cité en [1.2].
ADIBA M., DELOBEL C., "Les modèles relationnels ...", p. 2.11 ss.

[1.9] Codasyl cité en [1.1].
Guide/Share cité en [1.2].

[1.10] DATE C.J., cité en [1.3] p. 4 ss.
MARTIN J., cité en [1.4] p. 31 ss.

[1.11] WEDEKIND H., "Datenbanksysteme I" p. 9 ss., ainsi que les précisions données au chapitre 9.5 de notre étude.

[1.12] Ce schéma est repris de MARTIN J., cité en [1.4] p. 67 et a été quelque peu modifié.

CHAPITRE 2

LES FACTEURS INFLUENÇANT LA CONCEPTION D'UNE BANQUE DE DONNÉES

L'objectif de ce chapitre consiste à définir les facteurs à prendre en compte lors de la conception de la banque de données.

2.1. MATERIEL ET LOGICIEL

a) Matériel

L'existence ou le choix en faveur d'un matériel de type donné conditionnent fortement la conception, la réalisation et l'exploitation du système, entre autre par:

- la capacité mémoire interne détermine le type et l'efficacité du SGBD qui peut entrer en ligne de compte. En général, les SGBD des constructeurs sont limités à leurs matériels, alors que les SGBD de sociétés indépendantes sont d'utilisation moins restrictive. La limite inférieure de capacité mémoire indispensable varie également d'un SGBD à l'autre et ceci dépend du mode d'exploitation (exclusivement par lots ou mixte)
- la capacité et le type de mémoire périphérique influencent la conception en ce sens qu'ils déterminent le volume maximal de la banque de données et les performances
- la configuration du matériel joue également un rôle en ce qui concerne l'efficacité des traitements:
  - . nombre de canaux d'entrée-sortie
  - . temps de transfert des données
  - . nombre d'unités périphériques par canal
  - . nombre de supports par unité de contrôle
  - . caractéristiques des supports.

b) Le système d'exploitation

Les systèmes d'exploitation peuvent varier selon le type de machine et en fonction de la capacité mémoire (IBM-360,

DOS pour les machines du bas de la gamme, OS-MFT ou MVT pour celles du haut de la gamme, alors que DOS-VS ou OS-VS sont prévus pour les systèmes IBM-370). Le système d'exploitation à disposition limite les possiblités de choix entre SGBD, certains ne fonctionnant que sous le contrôle d'un type donné de système d'exploitation (IMS-VS sous OS-VS, alors que IDMS ou Adabas fonctionne tant sous DOS ou OS que sous DOS-VS ou OS-VS).

Il existe un lien étroit entre la version du système d'exploitation et celle du SGBD, la modification de l'une pouvant forcer à installer l'autre. Par exemple, une nouvelle version du IMS peut imposer l'installation d'une nouvelle version du système d'exploitation.

c) Occupation mémoire centrale

Selon les cas, le SGBD peut occuper un secteur (partition) propre de la mémoire centrale, ce qui implique des transferts interpartition (Adabas). Il peut également opérer dans la même partition que le moniteur de télétraitement et les programmes transactionnels, ce qui limite les communications interpartition aux programmes traités par lot.

2.2. LES MODELES DE DONNEES

La description de la structure des données peut se faire selon différents modèles, dont les caractéristiques principales seront décrites au cours de ce chapitre. En ce qui concerne le choix d'un modèle satisfaisant en vue de la description du monde réel considéré, une analyse critique de modèles représentatifs fera l'objet d'un chapitre spécifique (6.2.2).

Lors de la définition du concept banque de données, nous avons distingué trois différents niveaux de description

de la structure des données:

- modèle conceptuel (structure logique)
- modèle externe (vue partielle)
- modèle interne (structure physique).

Les modèles décrits ci-dessous seront également analysés en fonction de ces niveaux.

Conformément à la définition proposée au chapitre 1, un SGBD devrait également assurer le maintien de l'intégrité et protéger les données contre des utilisations abusives. Ces problèmes spécifiques seront analysés au cours du chapitre 7.1.

Le langage utilisé pour la description de la structure logique se dénomme langage de description des données (LDD ou DDL = Data Description Language). Les éléments de la structure ainsi décrite seront enregistrés dans une bibliothèque spécifique au SGBD appelée le dictionnaire des données (data dictionary).

2.2.1. Modèle hiérarchique

Ce modèle fait en quelque sorte suite aux techniques conventionnelles d'organisation des données de type séquentiel. Cette structure correspond à un arbre dont les noeuds ne peuvent être reliés qu'à un seul précédent, excepté le premier qui constitue la racine de l'arbre. La descente de la racine vers les feuilles s'effectue de manière séquentielle.

Le principal représentant de ce type de structure est le modèle du système IMS [2.1], les noeuds de l'arbre se dénommant un segment (figure 2.1). Une réalisation d'un segment n'a de signification que par rapport à sa position dans l'arbre et ne peut exister sans son précédent.

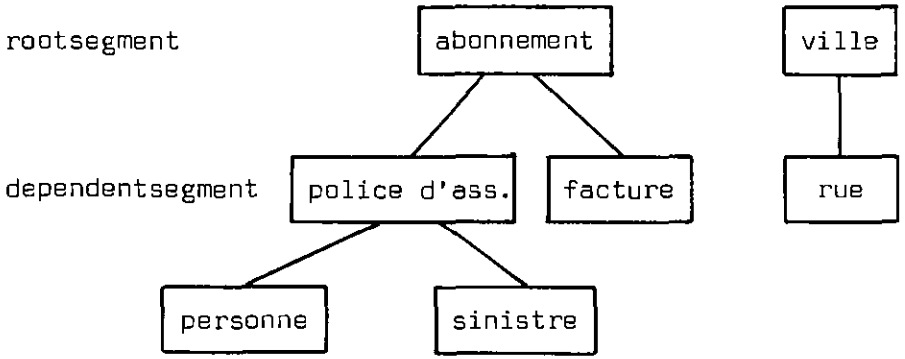


Fig. 2.1 - Exemple de structure hiérarchique

Toutes les occurences de segments dépendants appartenant à une réalisation du segment racine lui sont rattachées dans une séquence déterminée. Cette séquence englobe les noeuds de l'arbre dans le sens, de haut en bas et de gauche vers la droite, et forme un ensemble dénommé le "data base record".

Les limites de ce modèle dans la description des faits réels exigèrent une adaptation , en ce sens que deux arbres parallèles reliés entre eux peuvent former un réseau. Au niveau conceptuel, il est donc possible de décrire des structures de type réseau. Dans la terminologie IMS, un tel modèle se définit à l'aide d'un ensemble de "physical data base" (PDB). Des PDB reliées entre elles forment une "logical data base".

La liaison entre deux PDB peut s'effectuer dans un seul sens, comme c'est le cas dans l'exemple de la figure 2.2. Dans la figure 2.3, l'association entre EMP et VILLE est de type 1:n, mais en réalité la relation est du type n:m, ce qui nécessite la définition d'un lien

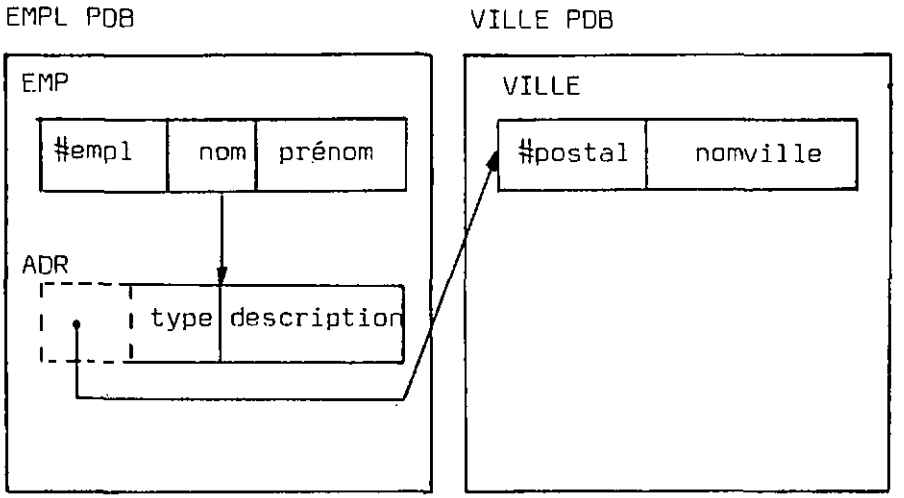


Fig. 2.2 - Exemple de liaisons entre deux PDB's

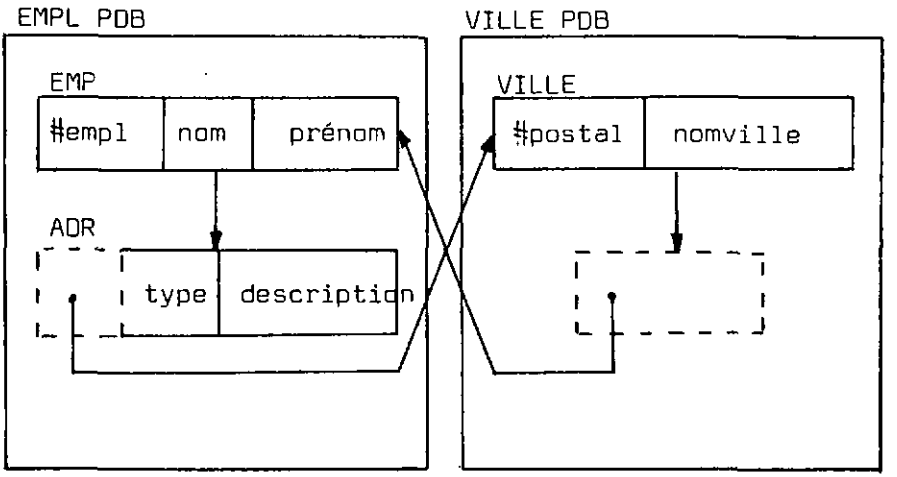


Fig. 2.3 - Exemple de liaison bidirectionnelle

bidirectionnel logique entre les deux PDB. En terminologie IMS, on parle dans ce cas de "pairing".

La vue partielle des utilisateurs ne peut se définir que sous forme hiérarchique et ne peut se référer qu'à un nombre restreint de constituants. Les segments peuvent appartenir à plus d'un arbre hiérarchique, à condition que ceux-ci soient liés logiquement entre eux. La vue partielle se dénomme en IMS "logical data base" (LDB), notion qui n'a pas la même signification que ci-dessus.

Pour chaque utilisateur, l'administrateur de la banque de données génère un interface (programm specification bloc PSB) qui définit entre autre la vue partielle et la correspondance entre celle-ci et la description de la ou des PDB concernées. La figure 2.4 représente le lien entre le programme d'application, la vue partielle et le modèle conceptuel. La différence entre la vue partielle, le programme d'application et la banque de données logique réside dans le fait que, dans le PSB, seule une partie des constituants est déclarée utilisable par le programme concerné. En terminologie IMS, on parle dans ce cas de "sensitivity" (voir exemple à la figure 2.5).

Quant au modèle interne, celui-ci est en partie intégré au modèle conceptuel dont la description nécessite la définition, entre autre :

- de la méthode d'organisation
- du type de pointeurs en vue de la réalisation des liaisons entre les segments
- de sous-arbres (data set group) afin de faciliter les accès aux segments d'un tel groupe
- du nom de la zone de débordement
- de la nature du support.

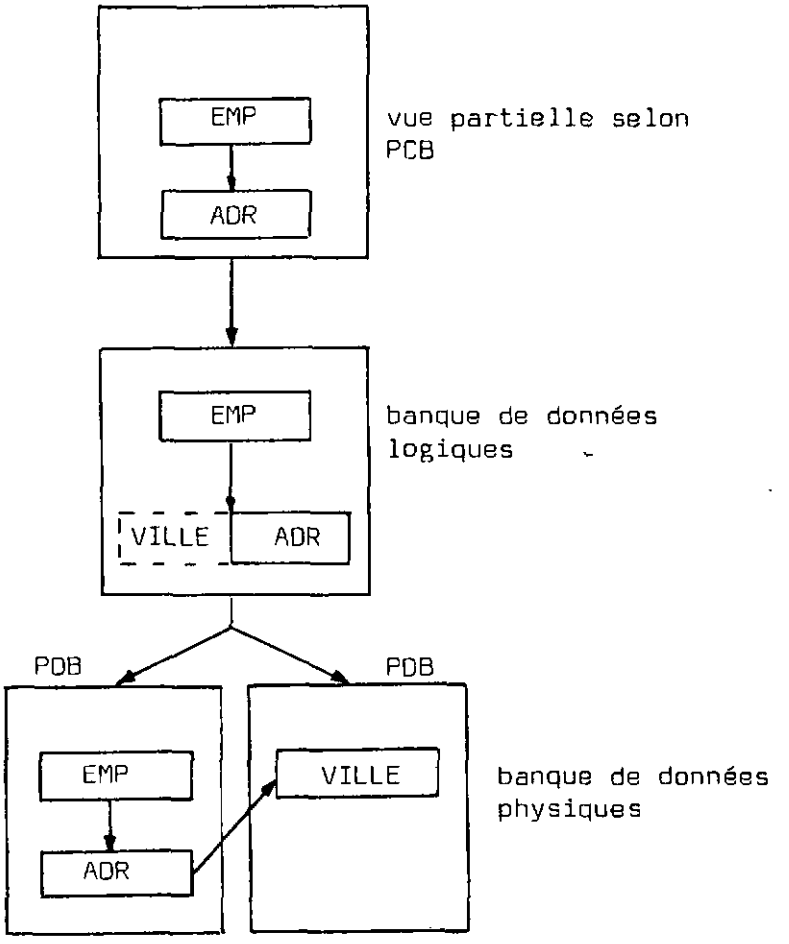
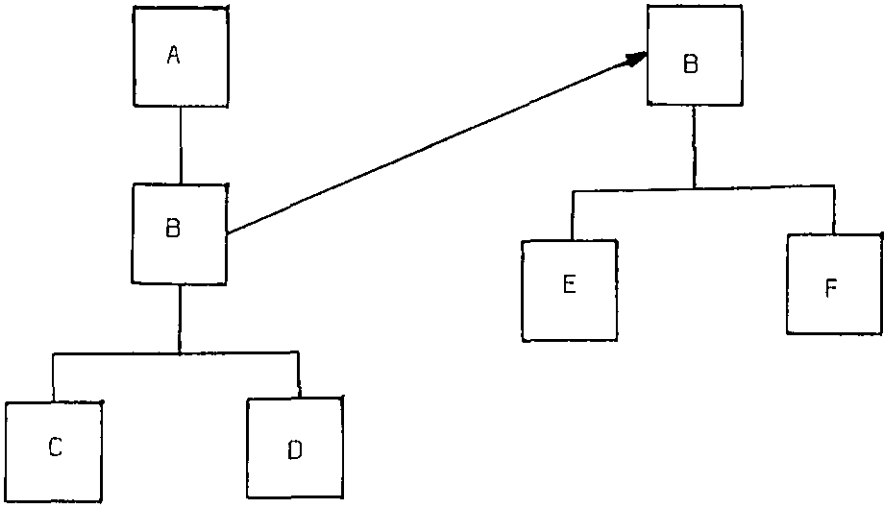


Fig. 2.4 - Relation vue partielle PDB



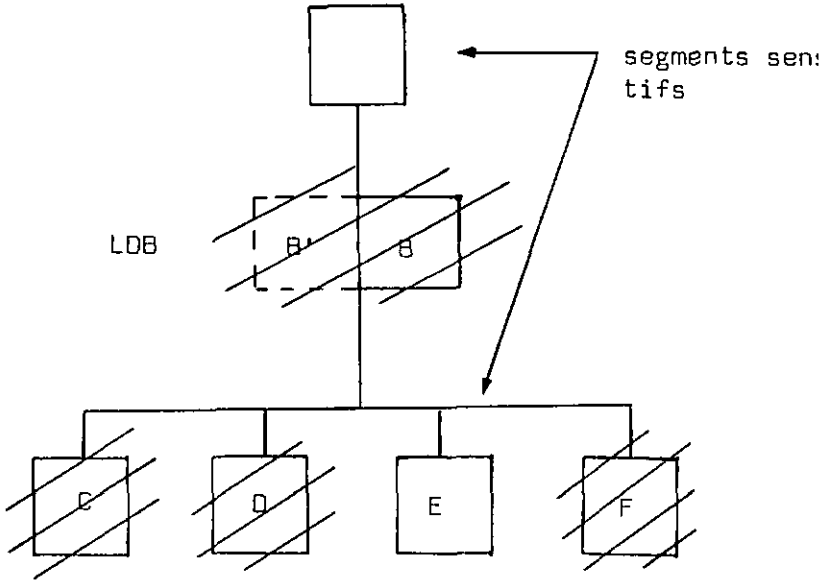


Fig. 2.5 - La notion de "sensitivity"

2.2.2. Modèle réseau

Ce modèle est la forme la plus générale de représentation de structure de données. Décrit sous forme de graphe, les noeuds peuvent être reliés à plus d'un précédent, ce qui autorise la définition de structures plus complexes et plus variées (figure 2.6).

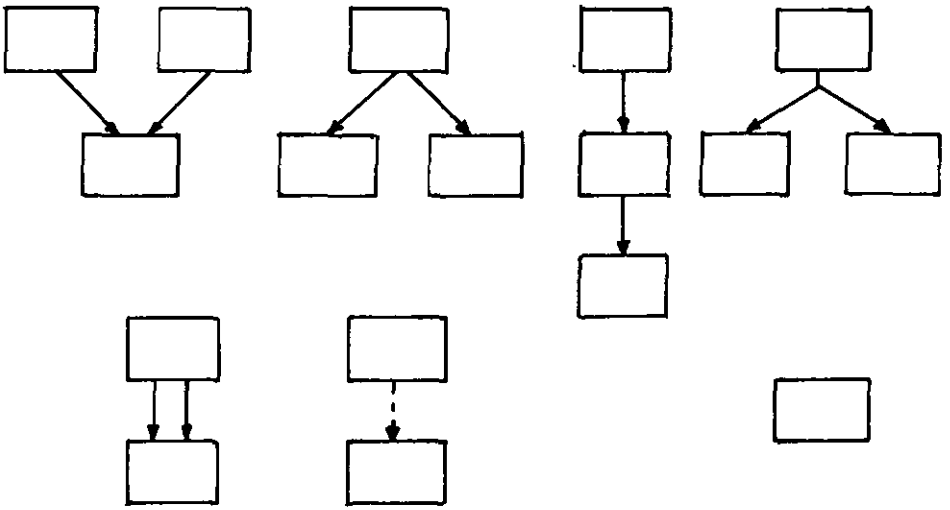


Fig. 2.6 - Structures de type réseau

Remarque : L'arbre est une application particulière du réseau. IMS est tout de même considéré comme un système hiérarchique, malgré le fait que la réalisation de réseau soit possible, car l'arbre est la structure de base et la séquence hiérarchique reste dans tous les cas valable. D'autre part; l'utilisateur n'aura toujours qu'une vue hiérarchique des faits élémentaires représentés par le modèle.

Ce modèle est à la base des propositions du "Data Base Task Group" (DBTG) de Codasyl [2.2]. Parmi les principaux systèmes ayant adopté en grande partie les propositions de Codasyl, citons entre autre:

- IDS (Honeywell)
- DMS 1100 (Univac)
- UDS (Siemens)
- IDMS (Cullinane Corp.).

Le noeud d'un graphe est dénommé le "record" et est composé d'un ensemble de champs (item) pouvant être membre d'un groupe.

La liaison entre deux "records" est dénommée le "set", elle est de type 1:n et relie tous les enregistrements fils (member) d'un enregistrement père (owner). Ce lien est représenté par une flèche, le graphe étant donc orienté.

La description des faits élémentaires, les "records" et les "sets", se nomme le "schema" ou schéma. Un cas particulier est la possibilité de définir différents degrés d'appartenance de l'élément fils dans un "set" :

- "mandatory", une occurrence d'un élément fils ne peut exister sans être relié à tous les "sets" dont il est membre et ne peut être transféré d'un "set" vers un autre ;
- "optional", il peut exister dans un "set" mais pas dans l'autre ou dans aucun d'entre eux, ou à l'extrême ne pas être rattaché à un père ;
- "automatic", l'insertion d'un élément fils dans l'occurrence du set concerné s'exécute automatiquement ;
- "manual", l'insertion ne se fait que si celle-ci est spécifiée explicitement lors de la mise à jour.

La vue de l'utilisateur intitulée "sub-schema", un sous-ensemble du schéma, définit le type de "record", les constituants et les liens entre les "record" concernés par l'application. Dans un sous-schéma, il est impossible de remettre en cause les éléments décrits dans le schéma.

Comme dans le cas de IMS, la distinction entre physique et logique n'est que partielle. La description du schéma implique en effet la définition

- du type de chaînage afin de réaliser le set
- de la méthode d'organisation (location mode)
- de l'appartenance d'un "record" à un secteur donné (within area)
- de la notion de position courante dans un "set" (set occurrence selection)
- du critère de tri d'une chaîne et le type de tri.

D'autres critères physiques qui ne sont pas définis dans le schéma se décrivent à l'aide d'un langage spécifique, le "device media control language" (DMCL).

Une autre possibilité de représenter un modèle réseau que celle proposée par Codasyl, qui est basée sur le concept du "set", utilise le principe hiérarchique et le complète par la notion de référence. Au lieu de représenter la liaison entre deux éléments par une séquence hiérarchique, on utilise le concept référence.

Le SGBD Socrate est le principal représentant de cette catégorie de modèle réseau [2.3]. Illustrons les possibilités offertes par Socrate au niveau de la description de la structure logique à l'aide de l'exemple présenté à la figure 2.7, selon le LDD nous obtenons:

```
entité cours
       nocours
       descr
       entité coursreq
              coursreq référence un cours
       entité offre
              date
              lieu
              prof inverse tout enseign
              entité étudiant
                     étu référence un étud

entité enseign
       nomatr
       nom

entité étud
       noenreg
       adresse
       diplôme
```

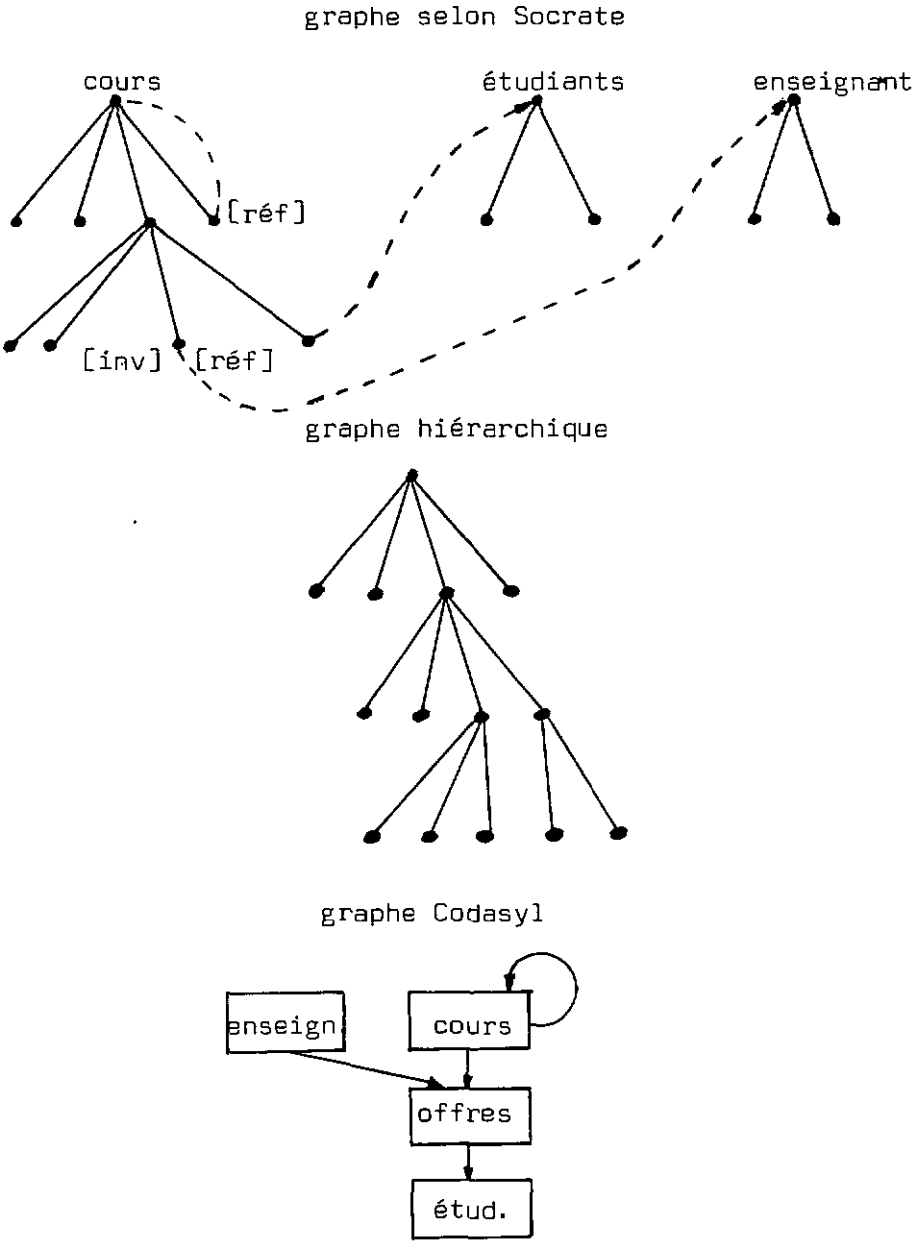


Fig. 2.7 - Structure logique selon Socrate

Résumons brièvement les caractéristiques du LDD de Socrate:

- la notion entité correspond au segment de IMS et au "record" du Codasyl
- la relation hiérarchique est représentée par l'imbrication des entités fils dans l'entité père
- pour éliminer la redondance, une liaison hiérarchique peut être représentée par une référence à une entité non imbriquée
- un groupe répétitif peut s'intégrer à l'entité père et être représenté sous forme d'inverse, d'où nouvelle possibilité de réduire la redondance
- on peut définir des boucles, ce qui n'est pas possible avec tous les SGBD de type Codasyl, entre autre IDMS
- la notion de "set" n'existe pas
- aucun critère d'implantation physique (type de pointeurs, méthodes d'organisation) ne doit être définit à ce niveau de la description logique
- il est possible de définir des plages de valeur par constituant
- la description d'une structure est simple en comparaison avec les LDD de IMS ou IDMS.

2.2.3 Modèle relationnel

Basé sur la théorie mathématique des relations, ce modèle a comme origine les propositions émises par Codd [2.4]. Une relation est généralement représentée sous forme de tableau (figure 2.8) dont les colonnes contiennent les domaines et les lignes les réalisations (n-uplets) de l'entité décrite par la relation [2.5].

#article	nom	couleur	qté en stock
11121	A	R	10
11517	A	V	19
14314	C	V	24
12510	D	D	91

Fig. 2.8 - Relations sous forme de tableau

De ce tableau on peut déduire les caractéristiques suivantes:

- le nombre et l'ordre des colonnes sont indifférents
- l'ordre des colonnes n'a pas de signification particulière
- chaque colonne a un nom unique
- l'ordre des réalisations des relations, n-uplets, ne joue aucun rôle
- il n'existe qu'un seul n-uplet composé de valeurs identiques
- pour une valeur d'un constituant donné, la clé primaire, il n'existe pas plus d'un n-uplet identique. Un groupe de constituants peut former cette clé, alors que le reste des constituants se dénomme des clés secondaires (candidate key).

Une représentation plus pratique que le tableau et qui s'utilise dans la description du modèle est la forme

$$R : (D_1, D_2, D_3, \ldots, D_n)$$

où R est le nom de la relation et D le nom des constituants. Le tableau de la figure 2.8 s'écrit donc comme suit:

Article :(numéro, nom, couleur, qté en stock)

la clé de la relation étant soulignée. Le nombre de constituants d'une relation indique le degré de la relation, 2 = binaire, 3 = ternaire, n = n-aire.

Le type de relation entre les constituants est une notion centrale du modèle relationnel. On parle ici de la fonctionnalité d'une relation. Une relation fonctionnelle est élémentaire si pour une valeur de la clé, il n'existe qu'une seule réalisation parmi l'ensemble des n-uplets d'une relation donnée. Cette relation abrégée RFE peut être directe ou indirecte et fera l'objet d'une présentation plus détaillée au chapitre 6.2.2.

Cette notion permet de caractériser les relations et d'en déterminer leur forme. Une relation est dite normalisée si tout domaine représente un ensemble de valeurs non décomposables. Le passage d'une forme à une autre se dénomme processus de normalisation. Ce processus sera également précisé au cours du chapitre 6.2.2.

La description des faits élémentaires du monde réel considéré s'effectue en définnissant d'une part les domaines et d'autre part les relations et leur clé primaire respective (figure 2.9).

La différence fondamentale de cette forme de modèle par rapport aux deux précédents (chapitres 2.2.1 et 2.2.2) réside dans le fait que les asscciations entre relations ne sont pas décrites dans le modèle. Des critères d'enregistrement physique n'apparaissent pas à ce niveau. Utilisé au niveau conceptuel, ce modèle peut être qualifié de neutre face aux problèmes d'enregistrement.

```
Domain : #empl
         nomempl
         prénom
         type
         description
         #postal
         nomville

Relation : EMP : (#empl, nomempl, prénom)
           key : #empl
           ADR : (#empl, type, description, #postal)
           VILLE : (#postal, nomville)
           key : #postal
```

Fig. 2.9 - Exemple de modèle conceptuel selon le modèle relationnel, excepté les contraintes d'intégrité

La vue des utilisateurs est décrite explicitement à l'aide d'un langage de manipulation, par exemple Sequel (chapitre 2.3) dans le cas du système R, qui permet de

- définir les sous-ensembles concernés en désignant les relations, les constituants et même certaines conditions
- déterminer les associations entre relations
- définir de nouvelles relations.

Les vues partielles des programmes d'application seront générées par le système, les programmes y faisant référence et le SGBD peut ainsi, en passant par le modèle conceptuel et la description de la structure physique, accéder aux n-uplets concernés et les transmettre dans la zone de travail du programe.

La vue partielle peut également se définir lors de la formulation de la requête, puisqu'à cette occasion l'utilisateur détermine le sous-ensemble auquel il désire accéder. Nous reviendrons sur ce problème au cours du chapitre 8.

Quant aux critères spécifiques à la structure physique, ils sont décrits séparément et n'interviennent pas aux deux autres niveaux de description de la structure.

Il n'existe encore aucun SGBD commercialisé de ce type, mais des prototypes sont en cours de réalisation, entre autre [2.6]:

- System R
- Ingrès

Un système comme Adabas peut être considéré comme une application restreinte du modèle relationnel, puisqu'il permet

- d'effectuer des opérations de type relationnel sur un nombre limité de constituants à l'aide d'un langage non procédural (voir chapitre 2.3.1). La démonstration fait l'objet d'une étude citée en [2.7]
- d'ajouter de nouveaux constituants à une relation sans modification des applications et sans réorganisation: il suffit de modifier le dictionnaire à l'aide d'un utilitaire
- d'ajouter ou de supprimer des possibilités de sélection sur des constituants (descripteur), ce qui n'implique ni modification, ni réorganisation. Cette opération s'effectue également à l'aide d'utilitaires
- de créer et de supprimer des couplages entre relations, afin de permettre l'exécution de requêtes se référant à plus d'une seule relation

- d'ajouter ou de supprimer des relations sans modifier la structure, ni exiger des modifications de programmes d'application et sans provoquer de réorganisations.

2.3. LES LANGAGES DE MANIPULATION

Un LMD a comme objectif l'accès, la mise à jour et la création de données. Il peut s'intégrer à un langage de programmation, par exemple Cobol ou PL1, qui joue dans ce cas le rôle de langage hôte (host language). Tous les SGBD cités au chapitre précédent offrent cette possibilité, leur utilisation principale ayant pour cadre des applications de gestion.

Dans un langage hôte, les primitives du langage de manipulation sont appelées à l'aide d'une instruction spécifique. Prenons à titre d'exemple le langage de manipulation DL/1 de IMS intégré au Cobol, la définition des primitives se présente sous la forme

```
enter linkage
call 'CBLTDL1' using [paramcount,] fonction,
nom PCB, zone entrée-sortie, SSA1, ... SSAn
enter cobol
```

Le paramètre [paramcount] est facultatif, il sert à la spécification du nombre de paramètres de l'instruction "call". Quant au paramètre fonction, il indique l'opération de manipulation prévue, alors que SSA signifie "segment search argument" et a comme fonction principale la description des données à manipuler:

- nom du segment
- code de commande qui permet de compléter le paramètre fonction
- partie qualitative.

Les données sont transférées dans la zone de travail du programme alors que les informations relatives aux communications entre le langage DL/1 et le programme (code de réponse, adresse de l'occurrence du segment transféré, etc) sont transférées dans un interface spécifique.

D'autre part, il existe des langages non intégrés, indépendants (self contained ou stand alone language) que l'on dénomme souvent langage d'interrogation (query language), entre autre:

- alpha [2.8]
- square [2.9]
- sequel [2.10]
- adascript [2.11]
- IQF de IMS [2.12].

Une autre distinction parmi les langages de manipulation se réfère au type de modèle sur lequel ils se basent. Les langages tel alpha, square ou sequel s'appliquent à des modèles de type relationnel, IQF à un modèle de type hiérarchique.

Les langages de type relationnel, également utilisables avec un langage hôte, font appel à des notions mathématiques et peuvent être classés en deux catégories:

- langage algébrique qui utilise pour interroger la banque de données des opérations algébriques complétées par des opérations ensemblistes telles que projection, anti-projection, restriction [2.13]
- langage des prédicats qui repose sur la logique mathématique. Le principe de base, expliqué de façon très simpliste, signifie qu'une proposition ne peut être que vraie ou fausse et ce indépendamment des valeurs attribuées [2.14].

A titre d'exemple, nous formulons différentes manipulations basées sur le modèle conceptuel de la figure 2.2 à l'aide du langage Sequel, celui-ci étant le plus simple à manier parmi les langages relationnels et utilisables tant intégrés à un langage hôte qu'indépendamment.

a) *Quel est le nom de l'employé de # = 113 ?*

```
Select nom, prénom
    from emp
    where #empl = '113'.
```

b) *Quels sont les numéros des employés habitant Zurich ?*

```
Select #empl
    from adr
    where #postal =
          select #postal
          from ville
          where nomville = 'Zurich'.
```

c) *Insérer un nouvel employé 'Dupont', 'Marcel', 129*

```
Insert into emp (#empl, nom, prénom)
          <129, Dupont, Marcel>.
```

d) *Supprimer l'employé de numéro 102*

```
Delete emp
    where #empl = '102'.
```

e) *Modifier la description de l'adresse privée (type = 1) de l'employé numéro 110*

```
Update adr (description)
     <villa bel-air>
     where #empl = '110' and
           type = '1'.
```

2.3.1. Langage procédural et non-procédural

Cette distinction est primordiale, car elle influence

- la structure des programmes
- les possibilités d'accès
- les efforts à effectuer au niveau de la programmation.

Un langage est dit procédural si l'accès aux données se fait en transférant enregistrement par enregistrement dans la zone de travail. Avec l'opération "get next" (DL/1), le programmeur demande de transférer chaque segment en vue d'une manipulation quelconque dans sa zone de traitement. L'aspect recherche domine, il est nécessaire de spécifier les procédures d'accès en fonction des caractéristiques d'enregistrement. Le terme naviguer (2.15) est souvent employé pour définir cette méthode d'utilisation, puisque le programmeur suit un chemin précis qu'il formule à l'aide des primitives d'accès du langage concerné. Dans le cas de IMS, le programmeur se déplace le long de la séquence hiérarchique, alors que dans le cas de Codasyl, il suit les occurences d'un "set" donné.

Le programmeur doit donc connaître le profil du chemin d'accès. La position courante est une notion importante qu'il doit mémoriser. Dans le cas de IMS, elle indique la position dans la séquence hiérarchique avant l'exécution d'un "get". Dans le cas de Codasyl, elle peut, en fonction de la spécification de la structure logique, représenter un autre enregistrement du "set".

Le fait que le segment recherché doive se trouver à droite de la position courante est une autre caractéristique importante relative au langage DL/1. Le déplacement sur une séquence hiérarchique s'effectue donc toujours de gauche à droite, un retour en arrière n'étant possible qu'en relation avec une opération "get next within

parent" en le spécifiant avec un code de commande figurant dans l'argument de recherche.

A l'opposé, un langage non-procédural n'exige pas la définition du chemin d'accès à suivre, il suffit de formuler les caractéristiques des données auxquelles on désire accéder. Il faut entre autre spécifier:

- le nom de l'entité
- les noms des constituants concernés
- les conditions relatives à cet ensemble,

le système traduisant ces requêtes en procédures d'accès.

L'objectif d'un tel type de langage consiste à définir un ensemble de données, le SGBD se chargeant du transfert dans la zone de travail. Dans le cas de Adabas, les adresses virtuelles (ISN) sont transférées dans la zone de travail, le programmeur peut ensuite appeler une occurence après l'autre dans sa propre zone de travail en vue du traitement. Une requête de ce type ne s'applique, dans le cas de Adabas, exclusivement sur un nombre restreint de constituants définis dans le modèle conceptuel. Par contre, un langage de type purement relationnel ne prescrit aucune restriction de ce genre.

Cette notion de procéduralité n'est pas absolue, car lors de la spécification d'une requête, la séquence des opérations est plus ou moins importante selon les systèmes. Les langages algébriques sont plus procéduraux que ceux basés sur le langage des prédicats, car dans le premier cas la séquence indique comment les résultats sont obtenus. Mais le critère de distinction décisif est la non-référence à un chemin d'accès.

Les langages non-procéduraux sont caractérisés par :

- leur relative simplicité
- leur indépendance face aux critères d'enregistrements
- le fait qu'ils n'impliquent pas de connaissances en informatique.

Jusqu'ici nous avons sous-entendu que l'utilisateur doive connaître les relations et les constituants définis dans le modèle conceptuel. L'initiative de l'interaction avec le SGBD incombe à l'utilisateur qui doit initialiser l'exécution de la requête. Les langages cités ci-dessus (alpha, square, squel, adascript) sont de ce type.

Une autre optique est celle fondée sur le principe de la délégation de l'initiative du dialogue au système, en ce sens que l'utilisateur, une fois son désir d'accès transmis, communique en mode interactif. A titre de réponse, il reçoit les noms des relations et de leurs constituants respectifs décrits par la structure logique. Il doit ensuite indiquer les éléments qui l'intéressent. A l'étape suivante toutes les caractéristiques de ces éléments lui sont transmises, ce qui lui permet de formuler sa requête. Une telle approche est un dialogue dirigé, puisque le système offre à l'utilisateur différentes possibilités à choix. Les langages "rendez-vous" [2.16] ou "query by example" [2.17] se basent sur ce principe.

2.3.2. Le type d'utilisateur

Une banque de données est prévue d'une part pour des utilisations selon différents modes (transactionnel, conversationnel ou par lot) et d'autre part pour différents types d'utilisateurs. L'accent est moins porté sur les connaissances en informatique que sur les raisons d'accéder aux informations. En effet, il est fort possible qu'un informaticien (programmeur-système ou d'application) utilise dans certains cas un langage non-procédural.

Le rapport Codasyl [2.18] distingue quatre catégories d'utilisateurs qui peuvent être complétées, en relation avec les développements du modèle relationnel et des langages qui s'y réfèrent, par un cinquième groupe.

a) *Personnel d'administration et d'exploitation*

Ce groupe englobe principalement la fonction de l'administrateur de la banque de données dont les fonctions et les activités seront décrites au cours du chapitre 10.2.2. Il s'agit dans ce cas d'un groupe d'utilisateurs hautement qualifiés.

La simplicité des différents langages (LDD, LMD) et les possibilités de structurer les données auront une influence capitale sur les efforts que le groupe doit fournir en cours de conception comme pendant l'exploitation et déterminent la grandeur de ce groupe.

b) *Les programmeurs d'application*

Ce groupe englobe également des personnes jouissant de connaissances informatiques suffisantes pour créer des programmes de type procédural.

Le problème qui se pose dans le cadre de ce groupe est l'effort exigé pour l'apprentissage du langage de manipulation, effort qui peut varier fortement d'un système à l'autre. Les langages de type relationnel sont, de ce point de vue, idéals.

c) *Utilisateurs non-informaticiens aux exigences élevées*

Toutes les personnes capables de formuler des requêtes complexes, par exemple des ingénieurs, économistes ou cadres commerciaux, sont classés dans ce groupe. Ces personnes s'occupent en général plutôt d'activités liées au système dispositif qu'opérationnel. Elles doivent utiliser le dictionnaire des données afin d'y trouver les noms et les caractéristiques des entités qui les intéressent. Elles doivent donc posséder des aptitudes quant à la formulation et à la logique mathématique.

Les langages de type relationnel sont prévus pour ce groupe d'utilisateurs. Mais des différences relativement importantes existent selon les connaissances mathématiques plus ou moins approfondies qu'implique l'utilisation d'un de ces langages. Un langage comme alpha exige des notions mathématiques approfondies, ce qui n'est pas le cas du langage Sequel. Si aucune référence aux mathématiques ne peut être exigé, l'utilisation d'un langage conversationnel en mode interactif s'impose.

d) Utilisateurs paramétriques

Aucune formalisation de requêtes selon la syntaxe d'un langage de manipulation n'est exigée de ce type d'utilisateur. Les personnes de ce groupe communiquent avec le SGBD selon un schéma prédéfini à l'avance sous forme d'un programme. Elles doivent introduire les données en fonction de règles déterminées qu'elles ont apprises en quelques jours ou semaines. Le mode d'utilisation est appelé dans ce cas transactionnel, par opposition au mode conversationnel du groupe précédent.

En informatique de gestion, ce groupe est le plus important et recouvre tout le personnel chargé de l'exécution des activités du niveau opérationnel, l'accent reposant principalement sur

- l'efficacité
- la simplicité
- la disposition des paramètres
- la clareté des opérations à effectuer
- le temps de traitement

e) Utilisateurs occasionnels

Le niveau technologique actuel ne permet pas à ce groupe d'utilisateur d'accéder à une banque de données. Ces personnes ne sont en général pas intéressées à une interaction avec le système en fonction d'une activité précise.

2.4. LES METHODES D'ORGANISATION DES DONNEES

L'objectif de ce chapitre ne consiste pas à décrire les caractéristiques techniques et les principes de fonctionnement des différentes méthodes d'organisation des données. A cet effet, il existe une littérature spécialisée, entre autre les ouvrages cités en [2.19]. Notre but consiste à les analyser en fonction de la conception d'une banque de données. Nous aborderons chaque méthode en relation aux problèmes suivants:

- capacité mémoire
- mise à jour
- mode d'accès, ceux-ci feront l'objet du chapitre 2.5.

Le choix des méthodes d'organisation dépend en premier lieu des modes d'accès et des contraintes de temps de réponse auxquels le système doit pouvoir satisfaire.

Une méthode d'organisation détermine une relation entre une structure logique et la structure physique, le bloc, l'enregistrement des données s'effectuant en fonction de cette relation.

Tout SGBD se base sur les méthodes proposées par les systèmes d'exploitation et les adapte selon des techniques spécifiques en vue de pouvoir réaliser des procédures d'accès aux données décrites par la structure logique.

Comme nous l'avons fait en ce qui concerne les modèles de données, nous analyserons les méthodes selon la même classification:

- hiérarchique
- réseau
- relationnel.

Les SGBD analysés peuvent également être classés selon le critère de réalisation des relations : physiquement dans la zone donnée (alinéas a et b) ou non (alinéa c). Ce critère est très important puisqu'il influence de manière décisive les possibilités de conception.

a) Représentation physique de structure hiérarchique

Nous nous limiterons à analyser les possibilités proposées par le système IMS qui est le représentant type de système hiérarchique [2.20].

Chaque arborescence (Physical Data Base) est enregistrée selon une des quatre méthodes:

- HSAM	hierarchical	sequential access method
- HISAM		indexed sequential access method
- HIDAM		indexed direct access method
- HDAM		direct access method

Les liaisons hiérarchiques sont réalisées selon deux méthodes différentes : la juxtaposition physique (séquentielle) et le chaînage à l'aide de pointeurs (adresse physique relative).

a1 Liaison séquentielle

La méthode HSAM enregistre les occurrences de segment *selon la séquence hiérarchique, de haut en bas et de gauche* à droite, dans des blocs de longueur fixe. Un segment n'est jamais scindé en deux en fin de bloc, mais est mémorisé dans le bloc suivant. Comme dans le cas de l'organisation séquentielle simple, l'actualisation s'exécute en réécrivant le fichier. Son utilisation est donc très limitée.

Quant à la méthode HISAM, la séquence hiérarchique est enregistrée dans deux secteurs différents : ISAM (zone primaire) et OSAM (zone de débordement) qui sont subdivi-

sés en blocs de longueur fixe. La zone ISAM est organisée selon la méthode séquentielle indexée, la clé primaire étant celle du segment racine. Une occurence de ce segment racine (root) est enregistrée avec autant de dépendants hiérarchiques que le bloc peut contenir, le reste étant mémorisé dans un ou plusieurs blocs OSAM. Les blocs (ISAM et OSAM) contenant toutes les occurences dépendant d'une seule occurence du segment racine sont reliés entre eux par une chaîne simple.

Les opérations d'insertion détruisent la séquence physique en ce sens qu'un nouvel arrivant doit être enregistré dans la zone OSAM si la séquence logique ne peut être réalisée physiquement. Cet événement se mémorise à l'aide d'un pointeur en début de bloc (root overflow pointer) qui initialise une chaîne des réalisations d'arbre qui se placent dans la séquence logique entre les valeurs de ce bloc et du précédent (figure 2.10).

L'insertion de segments dépendants nécessite des déplacements de segments dans la zone OSAM si le bloc qui doit les accueillir n'offre pas assez de place et des opérations de mise à jour de la chaîne afin de maintenir la séquence logique (figure 2.10, bloc A10). Cette opération exige le balayage de l'arborescence afin de positionner l'emplacement du nouvel arrivant.

L'occupation mémoire est détériorée par la longueur des différents types de segment, la longueur d'une occurence d'une arborescence et les opérations d'insertion de segments dépendants.

L'accès au segment racine est direct. Un balayage séquentiel des blocs formant l'arborescence, entrecoupé d'accès direct au bloc OSAM, s'impose pour accéder aux autres segments de l'arborescence. Les temps d'accès se détériorent rapidement si le taux d'insertion est relativement élevé.

Les suppressions s'effectuent en modifiant la valeur d'un "flag" dans le préfixe du segment.

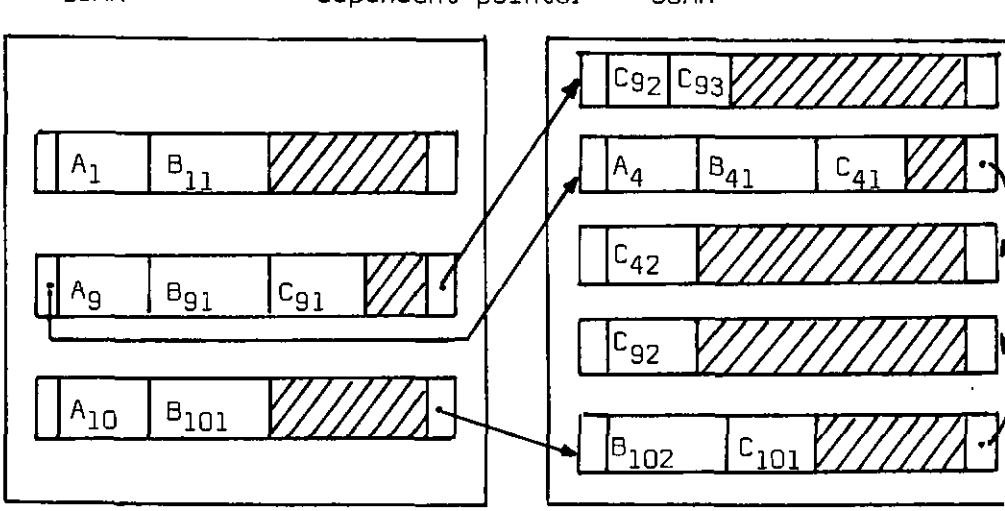


Fig. 2.1.0 - Exemple HISAM

a2) Liaison à l'aide de pointeurs

Les deux méthodes directes réalisant la séquence hiérarchique d'une arborescence (PDBR) en reliant les occurrences des segments à l'aide de pointeurs de type différent :

- pointeur hiérarchique selon la séquence hiérarchique (hierarchical pointer)
- pointeurs fils reliant un segment père à la première occurrence du segment fils concerné (child pointer)
- pointeurs frères reliant toutes les occurrences d'un segment de type donné (twin pointer).

Les deux derniers types de pointeur permettent d'accéder directement à partir du segment père à toutes les occurrences d'un segment de type donné.

Dans tous les cas cités ci-dessus, le chaînage peut également se faire dans le sens inverse en reliant chaque segment avec son précédent.

Au niveau du segment père, il est possible de définir un pointeur reliant la première ou la dernière des occurrences d'un ou de tous les segments fils.

Au niveau du segment racine, il est possible de relier toutes les occurrences à l'aide d'un pointeur frère dans un ou les deux sens.

Le choix du type de pointeur hiérarchique combiné dans certain cas avec des pointeurs fils et frères est fonction des fréquences d'accès et des contraintes de temps réponse.

- *HDAM*

Les segments d'une arborescence sont enregistrés dans une zone OSAM divisée en deux parties, l'une adressable (root segment adressable area) et l'autre étant une zone de débordement. Ces deux zones sont composées de blocs de longueur fixe.

L'enregistrement dans la zone adressable s'effectue selon un algorithme de calcul d'adresse (hash-code). Si le bloc alloué ne peut contenir le segment-racine, celui-ci sera enregistré dans le bloc le plus proche ou à l'extrême, dans la zone de débordement. Les collisions sont chaînées entre elles en reliant entre eux les blocs concernés par les collisions (anchor et twin pointer). Les segments dépendants quant à eux sont mémorisés dans le même bloc que le "root" ou dans un des blocs suivants. Si le volume occupé par une réalisation de l'arborescence dépasse le

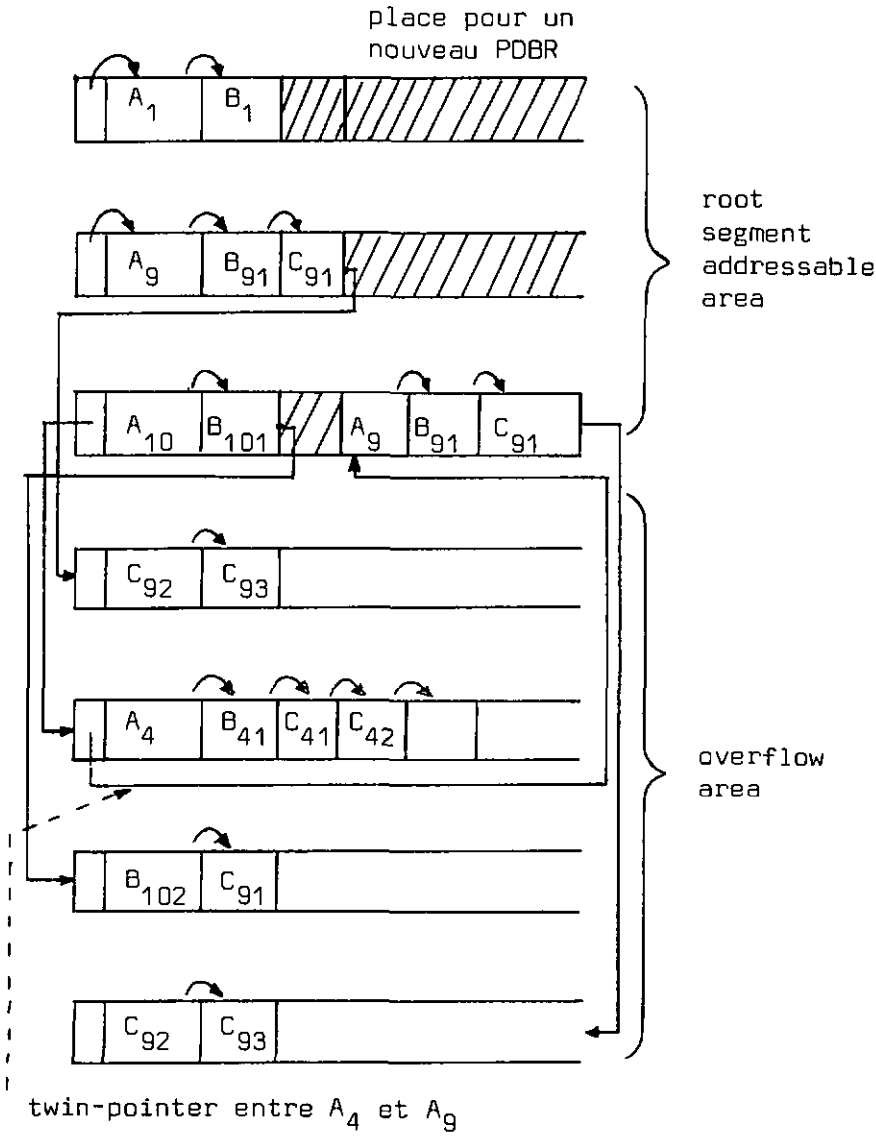


Fig. 2.11 - Exemple HDAM de la figure 2.10 avec pointeurs hiérarchiques uniquement

volume moyen alloué à la partie de l'arbre mémorisable dans la partie adressable, les dépendants seront enregistrés dans la zone de débordement (figure 2.11).

Les opérations d'insertion nécessitent la maintenance d'un nombre plus ou moins élevé de pointeurs, mais ceux-ci permettent l'accès direct aux éléments dépendants sans suivre la séquence hiérarchique. Le pointeur arrière facilite les insertions lorsque les chaînes concernées sont relativement longues.

L'occupation mémoire est mauvaise, car les collisions seraient trop nombreuses et les temps d'accès et de maintenance du fichier seraient par trop inefficaces si le taux d'occupation était trop élevé.

Les temps d'accès dépendent donc du taux de collision, de la longueur des chaînes de collision, du type de chaînage entre les éléments et la longueur de ces chaînes.

- *HIDAM*

Cette méthode implique la création d'une banque de données (PDB) index de type HISAM, formée d'une zone ISAM et OSAM, et une banque de données sous forme OSAM simple (figure 2.12).

La valeur de la clé primaire du segment racine est utilisée comme élément d'une table d'adresse gérée selon ISAM ou OSAM. Toutes les remarques faites à ce sujet sont donc valables ici.

L'enregistrement des segments dans la zone donnée, OSAM simple, s'effectue en mémorisant tous les segments d'un PDBR dans le même bloc ou le plus proche, tout en maintenant les pointeurs définis au niveau de la description de la structure logique (PDB).

Il est possible de relier entre elles toutes les occurrences du segment racine par une chaîne twin, ce qui

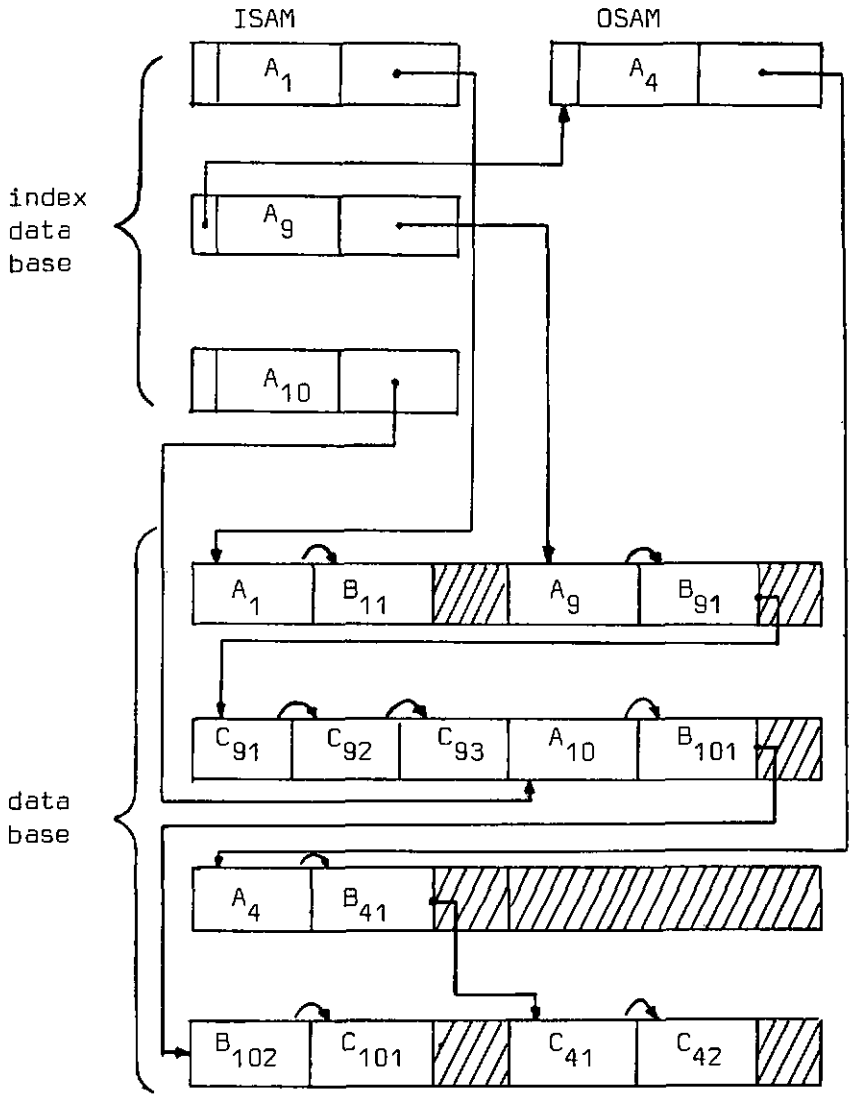


Fig. 2.12 - Exemple HIDAM de la figure 2.11

facilite l'accès séquentiel logique puisque l'index n'est utilisé que pour le positionnement.

Par rapport à l'organisation HDAM, les accès séquentiels logiques sont possibles tout en gardant la possibilité d'accès direct, par l'intermédiaire de l'index, au segment racine et aux descendants en suivant les pointeurs.

L'emplacement mémoire est mis à contribution par la base index, mais par contre une meilleure utilisation est possible dans la zone donnée non adressable.

La zone non adressable de ces méthodes (OSAM overflow en HDAM et OSAM simple en HIDAM) exige une gestion des emplacements libres qui est réalisée par une combinaison de chaîne (anchor point) et de liste de bit (IMS/VS). La zone adressable en HDAM peut également être complétée par une gestion des places libres.

- Secondary set groups

Les méthodes d'organisation HISAM, HDAM et HIDAM permettent la création de sous-groupes (2 à 10 au maximum). La racine d'un groupe secondaire ne peut être qu'un segment du deuxième niveau hiérarchique en HISAM.

L'avantage est de favoriser l'accès direct à la racine du groupe et d'allouer les groupes sur différents supports disques.

Cette technique n'est utilisable avec HISAM si VSAM est utilisé en tant que méthode d'organisation du système d'exploitation ou si l'indexation secondaire est définie sur cette banque de données.

- Indexation secondaire

La méthode d'indexation secondaire simple est appliquée. L'adresse peut être soit celle du segment source ou d'un autre segment, un supérieur hiérarchique.

La définition du segment cible dépend des fréquences d'accès au segment source et du taux de mise à jour du constituant indexé de ce segment.

Au maximum, cinq clés secondaires par segment source peuvent exister.

- Liaisons logiques entre arborescences

Il existe trois associations possible:

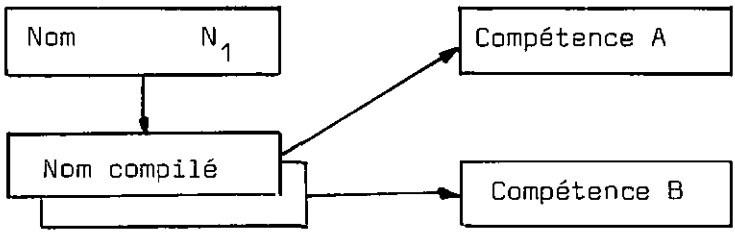
- unidirectionnelle
- bidirectionnelle physique
- bidirectionnelle virtuelle.

La profusion de termes complique l'explication de l'organisation. En effet, un segment est désigné différemment, selon la place qu'il occupe dans la séquence physique et logique. Donner des précisions supplémentaires dépasserait le contexte de notre travail. Par conséquent, nous nous limitons à représenter ces techniques à l'aide des figures 2.13 a, b et c.

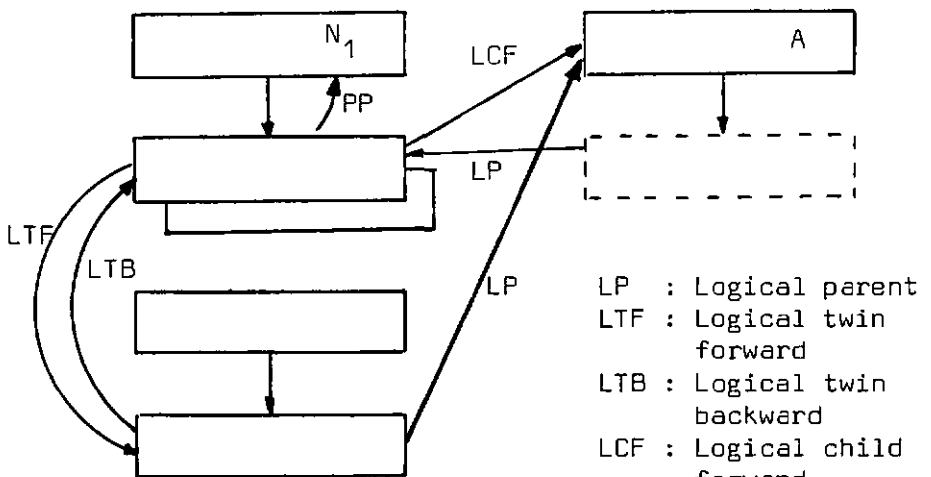
L'emplacement mémoire exigé par la profusion de pointeurs supplémentaires et la redondance prend des dimensions relativement importantes.

Les opérations d'actualisation sont très coûteuses en temps d'exécution et très compliquées. Des règles spécifiques à chaque type de manipulation se définissent lors de la description de la PDB et dépendent des accès que l'on désire maintenir après les manipulations.

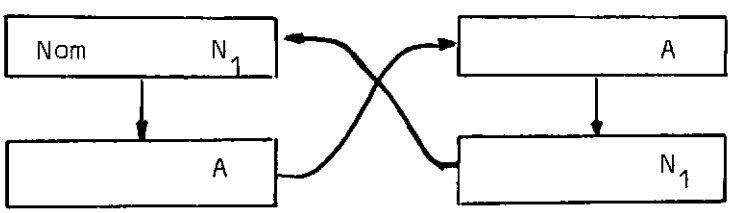
Remarque: L'énumération sommaire présentée ci-dessus donne un aperçu de la complexité du SGBD IMS. L'assimilation à ce système exige des efforts de formation non négligeables.



a) unidirectionnel



b) bidirectionnel virtuel



c) bidirectionnel physique

Fig. 2.13 - Liaisons entre arborescences

b) La structure physique d'un réseau

b1 Codasyl

La représentation de structures de type réseau peut également se réaliser avec le système IMS, mais la liaison s'effectue en reliant des hiérarchies entres elles.

Les systèmes de type Codasyl, nous nous référons ici à IDMS [2.21] enregistrent les "records" dans des blocs de longueur fixe dénommé "page" selon la méthode d'organisation avec adressage indirect. Si le schéma précise comme mode d'allocation "calc", le système transforme la clé primaire en une adresse correspondant à une page. Complétée par la position relative de l'enregistrement dans la page, cette adresse forme le "data base key". Les collisions se gèrent à l'aide d'une chaîne simple, les enregistrements nouveaux concernés sont transférés dans la page la plus proche de la page origine.

Un autre mode d'allocation existe, il s'agit du mode "direct". Dans ce cas, leSGBD enregistre l'occurence en fonction des pages libres qu'il gère lui-même. L'accès à une telle occurence ne peut s'effectuer que si l'on connaît le "data base key".

Quant à l'allocation "via set", elle a pour objectif la mémorisation des dépendants dans la page de l'enregistrement père ou le plus près possible de cette dernière.

Chaque type de "record" est alloué à un secteur (area) lors de la définition du schéma et chacun d'eux se compose d'un nombre défini de pages.

Quant au "set", il se réalise en utilisant le "data base key" en tant que pointeur. Trois types de pointeurs en permettent la réalisation et se définissent au niveau du

schéma.

En plus du mode d'allocation, d'autres critères sont à considérer:

- un "set" peut s'enregistrer en fonction d'un ordre spécifique défini dans le schéma
- la classe d'appartenance dans le "set", le "member ship class" (voir chapitre 2.2)
- la définition de la position courante "set occurence selection" au niveau du schéma.

Par rapport à IMS, il est possible d'accéder directement à n'importe quel noeud du graphe (allocation "calc", "direct" ou indexation secondaire) et la répartition des différents types de "record" sur la mémoire à disposition s'avère plus simple. D'autre part, la structure physique selon Codasyl introduit moins de notions techniques à assimiler.

b2 Socrate

Par rapport aux systèmes IDMS et IMS, le concepteur n'a aucune possibilité de choix pour organiser la structure physique des données. Dans le cas de Socrate [2.22], le passage se fait directement, l'algorithme d'adressage déterminant l'adresse réelle.

L'espace virtuel est un espace structuré à partir du nombre total de réalisations indiqué pour chaque entité dans la description du modèle conceptuel. Pour chaque entité, cet espace comprend :

- une chaîne d'existence dont la longueur est égale au nombre total de réalisations en bits répartis sur un nombre entier de mots (32 bits)
- un mot système qui sert à chaîner toutes les entités qui référencent la même réalisation d'une entité donnée

- place réservée aux réalisations d'entités.

L'adresse virtuelle d'une donnée élémentaire se calcule à partir de la description du modèle, il suffit d'en connaître la longueur qui dépend de la nature de la donnée (mot, inverse, référence, texte).

Le plus petit élément manipulable est donc le constituant, alors que dans IMS et IDMS, il s'agit du segment, respectivement du "record".

Quant à l'espace réel, il correspond à l'espace disponible sur le support. Le problème consiste donc à une projection de l'espace virtuel sur l'espace réel qui est découpé en pages de 256 mots, elles-mêmes divisées en sous-pages dont le volume est à déterminer par le concepteur. Chaque sous-page contient un mot d'en-tête comprenant :

- une partie de l'adresse virtuelle, le reste étant calculable
- un bit indiquant si la sous-page contient des données transférées à la suite de collisions
- chaînage de sous-pages soit pour effectuer la chaîne de collision, soit pour chaîner les sous-pages vides d'une page
- le nombre de mots occupés dans la sous-page.

La transformation de l'adresse virtuelle en adresse réelle pose, en outre, le problème du traitement des phénomènes de collisions. Socrate résoud la difficulté en cherchant la première sous-page entièrement vide dans la même page ou dans une des pages voisines. Dans le dernier cas, les sous-pages sont chaînées entre elles. Si la transformation indique un espace déjà occupé par une donnée qui s'y trouve après une collision, c'est cette dernière qui devra être à nouveau déplacée.

Le constituant qui référence une autre entité contient l'adresse de l'entité qu'il référence. Dans le cas de l'inverse, l'adresse est remplacée par une chaîne de bits d'existence, ce qui permet de réaliser des gains en emplacement mémoire si le nombre des réalisations de l'entité référencée n'est pas trop élevé.

c) Représentation physique du modèle relationnel

Un des objectifs principal d'un SGBD de type relationnel est, entre autre (voir chapitre 1.2), l'obtention d'un degré d'indépendance maximum. La réalisation d'un tel objectif implique au niveau de la représentation physique une organisation qui permette d'effectuer le choix du chemin d'accès optimal en fonction des requêtes formulées, celles-ci n'y faisant pas référence.

Actuellement, aucun SGBD commercialisé est de type relationnel, mais un certain nombre de prototypes existent [2.23]. Pour cette raison, nous analyserons les caractéristiques d'un système quasi relationnel, le SGBD Adabas (voir chapitre 2.2.3).

Supposons au départ que les relations décrites au niveau du modèle conceptuel sont en 3FN.

Les occurences d'une relation de type donné sont enregistrées dans un secteur qui peut se répartir sur différents supports.

Un tel secteur se dénomme un fichier (file) et est divisé en blocs de longueur fixe organisé selon la méthode d'adressage indirecte (sans calcul d'adresse).

Les occurences d'une relation sont compressées selon un module spécifique, les zéros à gauche dans un champ numérique, les espaces vides dans un champ alpha-numérique et

les champs vides ne sont pas mémorisés physiquement, alors que les champs numériques sont automatiquement compactés (packed).

A chaque occurrence de relation est alloué un numéro interne logique unique (ISN) pour chaque type de relation. Ce numéro interne peut être alloué

- par le SGBD de façon continue
- par le SGBD en réutilisant les numéros libérés par les suppressions
- par l'utilisateur, le SGBD vérifiant la règle d'unicité

celui-ci étant enregistré avec les données.

L'enregistrement dans un bloc s'effectue en fonction d'une table gérant les blocs vides. La correspondance ISN adresse du bloc est réalisée à l'aide d'une table, l'ISN étant la position relative dans la table.

Un exemple de réalisations d'une relation est présenté à la figure 9.6.

En fonction de la définition dans le modèle conceptuel, une liste inversée est créée pour chaque constituant descripteur, au maximum 200 par relation.

La liste inversée est organisée en 4 niveaux et selon les mêmes techniques que pour la mémorisation des données primaires. La liste contient pour chaque élément un compteur (nombre de réalisations) et 1 à n numéros internes. L'option "NU" a pour conséquence la non représentation de la liste des ISN référençant des relations dont le champ descripteur est vide.

Les opérations de mise à jour impliquent la définition d'une réserve dans chaque bloc, afin de pouvoir absorber les modifications de longueur après une mise à jour. Si le bloc ne peut accueillir la modification, l'enregistrement est complètement transféré dans un bloc vide. Seule,

la table des correspondances doit être modifiée, les listes inversées ne sont pas concernées.

Afin de pouvoir formuler une requête qui référence des constituants inversés appartenant à deux relations différentes, il faut coupler les relations entre elles par l'intermédiaire d'un constituant inversé figurant dans les deux relations concernées. Au niveau physique, le couplage est réalisé par une liste inversée.

Une relation peut contenir des groupes périodiques. Il existe deux types différents de groupe:

- champ multiple, composé d'un seul constituant
- groupe périodique, composé de plus d'un constituant et peut contenir des champs multiples.

Avec chaque groupe, le système enregistre un octet contenant le nombre d'occurences mémorisé dans la banque de données. Au niveau du programme d'application, il est possible de consulter ce compteur.

Au niveau des données primaires, l'occupation mémoire s'effectue de façon optimale, car le système comprime les données. Mais l'enregistrement des listes inversées rogne une partie de ce gain. La méthode d'adressage utilisée n'implique aucune zone de débordement ni des emplacements supplémentaires. Il suffit de prévoir la place suffisante pour accueillir les nouveaux arrivants et pour absorber le rallongement des enregistrements existants, ceux-ci étant de longueur variable. Le volume des fichiers données est plus limité que dans le cas de IDMS ou IMS, car il n'y a pas de place occupée par des pointeurs, les données sont comprimées et la méthode d'organisation n'implique aucun emplacement supplémentaire. Il en résulte des gains de temps lors de l'exécution de balayages séquentiels.

D'autre part, la correspondance entre la réalisation d'une relation et son adresse physique n'est pas directe, Adabas alloue à chacune d'elles un numéro interne (ISN).

2.5. LES METHODES D'ACCES

L'objectif principal de l'implantation d'une banque de données est de faciliter la recherche des informations contenues dans celle-ci, conformément au modèle conceptuel. Les demandes d'accès impliquent donc l'existence de procédures de recherche en vue de déterminer les données correspondant à une telle demande et de les transférer dans la zone de travail allouée à l'utilisateur, afin que ce dernier puisse effectuer les manipulations voulues.

Formuler une requête (chemin d'accès logique) nécessite une procédure de choix de méthodes d'accès afin d'obtenir les résultats en un temps acceptable pour l'utilisateur. Les méthodes d'implantation des chemins d'accès au niveau physique conditionnent directement

- le type de requête qui peut être traité
- les procédures d'accès
- le type d'utilisateur.

Une première distinction importante quant aux caractéristiques des chemins d'accès physiques est la réalisation des associations entre les catégories (segment, "record" ou relation) qui peut s'effectuer par des méthodes d'organisation

- séparant les chemins d'accès des données primaires
- intégrant les chemins d'accès aux données.

Dans le premier cas, le système peut choisir le chemin d'accès le plus court, en déterminant le sous-ensemble des données satisfaisant à la requête à l'aide des données secondaires avant d'accéder effectivement aux données primaires en vue de les mettre à disposition de l'uti-

lisateur. L'intégration partielle ou complète des chemins d'accès limite les possibilités de choix du chemin d'accès, car la structure physique dicte dans ce cas l'accès aux données (voir chapitre 2.4). Cette distinction aura des incidences aux diverses étapes de la méthode d'approche, entre autre (chapitre 4 et suivants):

- la réalisation de l'indépendance entre les phases de la conception
- l'indépendance physique
- les possibilités offertes au niveau de la conception.

Les méthodes d'accès peuvent également être classées en fonction de l'ensemble des données auxquelles l'on peut accéder. Une recherche selon la clé primaire n'aboutit qu'à un seul enregistrement, alors que dans le cas d'une clé secondaire, plusieurs enregistrements sont accessibles. Les procédures d'accès dépendent en premier lieu de la méthode d'organisation des données.

Au cours de ce chapitre, nous nous bornerons à étudier les méthodes d'accès en fonction:

- des méthodes d'organisation
- des possibilités de formuler des requêtes
- des aspects liés au critère de performance
- du mode de traitement.

a) Les méthodes d'accès mono-critère

Il existe plusieurs méthodes pour accéder par l'intermédiaire d'une clé primaire à l'enregistrement logique correspondant. Chacune d'elles implique une méthode d'organisation adéquate [2.24].

a1 Accès par comparaison de clés

Le balayage séquentiel physique peut s'effectuer avec toutes les méthodes d'organisation, mais dans le cas de l'organisation séquentielle sans critère de classement et celle à adressage indirect, il n'existe aucune relation entre la procédure d'accès et la structure d'enregistrement. Un balayage complet du fichier s'impose. Cette méthode n'a d'intérêt que si le sous-ensemble cible est relativement important par rapport à l'ensemble à balayer, car elle permet d'optimiser les mouvements des têtes de lectures-écritures et le volume des données à transférer en mémoire centrale se maintient dans des limites acceptables. Ce type d'accès est utilisable dans le cas du mode de traitement par lot, le traitement en temps réel n'étant possible que si le volume à balayer est très réduit (table de conversion).

Le balayage séquentiel logique implique soit une organisation séquentielle dont les données sont enregistrées selon l'ordre de la clé primaire, soit une organisation index-séquentielle (ISAM, VSAM, HSAM, HISAM, HIDAM). Dans ce cas, des accès supplémentaires suivant les pointeurs permettant de maintenir la séquence après des opérations d'insertion sont nécessaires. Le nombre des mouvements des têtes de lectures-écritures et des blocs à transférer augmentent. D'autre part, les temps d'accès se dégradent rapidement en fonction du taux d'insertion et de la longueur des chaînes. Dans le cas de Adabas, un tel accès peut s'effectuer en utilisant la table d'adresse de la clé primaire (= champ inversé). Le temps de traitement est plus réduit et varie fortement selon la distribution des enregistrements dans la zone des données primaires. Dans tous les cas, le chargement initial s'exécutera selon le critère d'ordre de la clé primaire. Des réorganisations périodiques permettrons de maintenir les temps d'accès dans des limites acceptables. Un tel type de balayage autorise des accès à des enregis-

trements selon un certain ordre, ce qui évite l'exécution de procédures de tri coûteuses en temps de traitement.

Deux autres techniques d'accès permettant d'accélérer la recherche d'un enregistrement à l'aide de la clé primaire, à condition que l'organisation séquentielle simple s'effectue selon l'ordre de cette clé, existent:

- la recherche binaire (dichotomique) où l'accès se fait en divisant le fichier par deux et en comparant les clés afin de déterminer si la procédure doit ensuite s'appliquer au secteur de gauche ou de droite et ainsi de suite
- la recherche par secteur, la comparaison s'effectue dans ce cas en analysant la clé primaire du dernier élément de secteurs de longueur fixe.

Les méthodes d'organisation index-séquentielle permettent un accès relativement rapide à un enregistrement donné. Les temps d'accès dépendent:

- du nombre de niveaux d'index
- de la position des indexes sur les supports
- de l'organisation des zones de débordement
- du taux d'insertion
- de la distribution des nouvelles valeurs de la clé primaire
- des méthodes d'accès aux indexes.

L'accès à ces indexes peut prendre différentes formes:

- recherche binaire
- recherche par secteur
- recherche arborescente
  - . binaire, le nombre de noeuds dépendants est limité à deux
  - . arbre balancé, le nombre de noeuds dépendants n'est pas limité, mais le nombre d'éléments par noeud est fixe

. arbre non balancé, se distingue du précédent par le fait que le nombre d'éléments par noeud peut être variable.

Dans les deux derniers cas, la recherche dans un noeud peut s'exécuter par une recherche séquentielle, binaire ou par secteur.

L'existence d'indexes permet, en plus d'un balayage séquentiel, d'effectuer un positionnement sur un enregistrement selon une valeur donnée de la clé primaire et d'accéder séquentiellement aux enregistrements suivants.

a2 Accès par adressage

Cette méthode permet l'accès le plus rapide à un enregistrement donné.

Dans le cas de l'adressage calculé, les collisions provoquent une dégradation des temps d'accès.

Des accès logiques séquentiels et le positionnement suivi d'une recherche séquentielle ne peuvent s'exécuter. Seul un enregistrement donné peut être recherché et non un sous-ensemble des réalisations en fonction de valeurs données de la clé primaire.

Les méthodes d'adressage mono-critère citées ci-dessus ne permettent pas de formuler des requêtes très complexes. Mais l'accès est, dans certains cas, très performant lorsqu'il s'agit d'effectuer des opérations de mises à jour, d'insertion et de suppression, car elles permettent une identification non ambigue de l'enregistrement concerné.

L'utilisation de pointeurs en tant que possibilité de réalisation de la séquence logique a entre autre comme objectif principal l'accélération des accès aux données

tout en facilitant les opérations d'insertion et de suppression (IDMS, IMS).

b) Les méthodes d'accès multi-critères

Les chemins d'accès se réalisent avantageusement à l'aide d'une méthode d'indexation secondaire ou selon une technique d'organisation liste inversée [2.25]. D'une part, les chemins d'accès ne sont pas intégrés aux données primaires ce qui permet de déterminer le sous-ensemble de données concerné par une requête avant d'accéder aux données primaires. D'autre part, le fait de ne pas utiliser d'adresses physiques, en tant que référence dans les tables, procure un degré d'indépendance physique maximal.

La méthode liste inversée offre un avantage supplémentaire en ce sens qu'il devient possible d'optimiser les chemins d'accès en comparant:

- le nombre d'occurences concernées par une requête en fonction du nombre total de réalisations, ce qui permet de déterminer le mode d'accès, séquentiel ou direct
- les listes inversées concernées les plus courtes avec les plus longues afin de déterminer les références des enregistrements cibles.

Cette stratégie d'accès implique pour chaque valeur figurant dans la table d'adresse l'enregistrement, en plus des adresses elles-mêmes, de la longueur de la liste.

Des requêtes peuvent donc se référer tant à la clé primaire qu'à des clés secondaires et même à des combinaisons faisant appel aux deux types de clés.

Les temps d'exécution sont influencés directement par la distribution des valeurs des clés et par la longueur des listes.

Le SGBD Adabas, quant à lui, sépare complètement les données primaires des données secondaires. Il permet donc de formuler des requêtes complexes faisant intervenir les champs inversés d'une ou, grâce à la possibilité du couplage, de plusieurs relations. L'accès aux données primaires ne s'effectue qu'une fois la sélection des enregistrements concernés réalisée. Cette sélection s'effectue au niveau des données secondaires.

2.6. CARACTERISTIQUES PROPRES AUX APPLICATIONS

La création d'une banque de données est conditionnée par les applications à réaliser et inversément leur création dépend directement des facteurs cités au chapitre précédent. Considérons les principales caractéristiques à prendre en considération.

a) Les données

Les données sont formatisées puisque nous nous limitons aux applications de gestion.

Le critère principal à considérer consiste à déterminer la signification des données, leur définition non ambigue et les associations qui les relient. La structure des données du monde réel examiné doit être connue, afin de pouvoir la décrire en fonction des caractéristiques du LDD du SGBD à disposition. Cette activité d'analyse des données occupe une place importante dans le processus de conception car, si elle n'est pas effectuée de manière efficace, les conséquences qui en résultent peuvent provoquer des situations très graves:

- impossibilité de satisfaire certaines exigences
- coûts de réalisation et d'exploitation supplémentaires
- obstacles à l'évolution des applications.

Les données doivent d'autre part être soumises à une analyse en fonction de leur variabilité. Certaines se caractérisent par leur constance dans le temps et impliquent rarement des mises à jour, alors que d'autres montrent une constance relative dans le temps.

La relation des données en fonction de l'aspect temporel permet de définir leur existence par rapport à un instant précis dans le temps.

b) Les besoins en information

L'objectif principal de la réalisation d'une banque de données consiste à mettre à disposition des différents utilisateurs les informations enregistrées en fonction de leurs besoins. Par conséquent, la banque de données doit pouvoir répondre aux différentes demandes.

L'extraction des données peut se faire en accédant directement à la banque de données ou par l'intermédiaire de procédures de traitement.

D'autre part, l'analyse des besoins en fonction de l'utilisateur qui les exprime s'impose, ce qui permet d'évaluer leur importance [2.26].

c) Stratégie d'accès

L'accès aux données peut se déterminer à l'avance et se dérouler toujours selon le même schéma. Concrètement, une telle stratégie se réalise à l'aide de programmes transactionnels ou par lots.

Si par contre l'accès ne peut se définir à l'avance, l'utilisateur doit pouvoir formuler sa requête en fonction de son niveau de connaissances en informatique et en logique mathématique. De telles demandes ne peuvent donc être posées que si un langage de requête adéquat existe.

Les stratégie d'accès font appel à un certain nombre de critères de recherche dépendant du besoin en information. Par conséquent, il est nécessaire de les recenser afin de pouvoir étudier les possibilités de réalisation. Comme nous l'avons vu au chapitre précédent, la liberté d'expression des requêtes varie fortement d'un SGBD à l'autre.

d) Objectifs des manipulations de données

L'accès aux données résultent d'objectifs divers:

- satisfaire les besoins en information des utilisateurs
- exécuter les opérations de mises à jour
- exécuter des procédures en fonction de règles de gestion précises (facturation, bons de commandes, etc.).

Pour chaque opération, le volume des données à traiter, le volume de la banque de données, le taux d'activité et la périodicité déterminent, en plus des contraintes de temps, les caractéristiques des structures physiques et des procédures chargées d'effectuer ces manipulations.

e) Modes de traitement des données

Les applications peuvent, suivant les exigences de temps réponse, se réaliser à l'aide de procédures de traitement par lot ou en temps réel. Le volume des données à traiter et le volume du sous-ensemble cible concerné influencent le choix du mode de traitement.

Le mode de traitement détermine les caractéristiques de la structure physique et élimine les méthodes d'accès non adéquates.

f) Temps de réponse

Ce genre de contraintes détermine directement les possibilités de réalisation de la structure des données au niveau physique. Cette contrainte est d'autant plus critique que l'utilisateur communique directement avec le système et que toutes dégradations ou interruptions bloquent le déroulement des activités. Dans le cas du mode de traitement en temps réel, des mesures adéquates sont nécessaires non seulement au niveau de la structure physique, mais éagalement lors de la conception des applications.

En général, dans le cas de traitements par lots, les conséquences ne sont pas aussi critiques et il est possible de combiner des séquences de balayage séquentiel avec des accès directs et d'y intercaler des procédures de tri tout en conservant des temps de réponse satisfaisants.

g) Sécurité

De par la centralisation des données se pose de nouveaux problèmes liés d'une part au maintien de la cohérence des faits enregistrés et d'autre part à la confidentialité. L'intégrité ne dépend pas d'un type d'application donné, mais une violation de cette contrainte aura des incidences variables selon les applications.

h) La redondance

Un des objectifs de la réalisation d'une banque de données consiste à minimiser la redondance au niveau physique. Mais pour des raisons de sécurité et de performance, une certaine redondance peut s'avérer nécessaire.

i) La flexibilité

La plupart des applications se caractérisent par leur

dynamique. Les adaptations nécessaires au cours de la vie du système informatique devront se réaliser avec un minimum d'efforts et surtout ne pas remettre en cause la structure logique. Le SGBD choisi détermine en premier lieu les possibilités d'adaptation, celles-ci variant fortement d'un SGBD à l'autre.

j) Migration vers le nouveau système

Toute nouvelle application s'insère dans un système existant, informatisé ou non. Le passage vers le nouveau système doit garantir un déroulement continu des opérations et ne pas provoquer d'interruptions. Il est nécessaire de considérer ce critère au cours de la phase de conception.

k) Les coûts

La réalisation d'une banque de données n'est justifiable que si des avantages économiques sont garantis. Donner une réponse à cette importante question pose le problème de la mesure des paramètres à considérer qui sont principalement de nature qualitative et donc difficilement mesurables. L'étude de rentabilité devra entre autre faire ressortir les coûts suivants:

- coûts d'achat ou de réalisation du SGBD
- coûts de réalisation des applications informatiques, de la banque de données et de tous les efforts à produire en vue et au cours de l'exploitation
- coûts exigés par des adaptations futures
- coûts provoqués par l'adaptation du SGBD à son environnement.

On confrontera les avantages, chiffrés si possible, avec les coûts probables afin de déterminer si le système de gestion fera appel à une banque de données ou plutôt à des fichiers conventionnels.

REFERENCES

[2.1] IBM, "IMS/VS ...".
DATE C.J., "An Introduction ..." p. 137 ss.

[2.2] CODASYL, "DBTG Report".
CODASYL, "DDL Journal of ...".
CODASYL, "Feature Analysis".
DATE C.J., cité en [2.1] p. 225 ss.

[2.3] ABRIAL J.R. et a., "Projet Socrate".
ADIBA M., DELOBEL C., "Les modèles relationnels ..." p. 2.21 ss.
DELOBEL C., "Les systèmes ..." p. 109 ss.

[2.4] CODD E.F., "A relational model of data ...".
CODD E.F., "Further normalization ...".

[2.5] ADIBA M., DELOBEL C., cité en [2.3].
DATE C.J., cité en [2.1] p. 61 ss.
DELOBEL C., cité en [2.3] p. 21 ss.
MARTIN J., "Computer based ..." p. 149 ss.
WEDEKIND H., "Datenbanksysteme I" p. 41 ss.

[2.6] ASTRAHAN M.M. et a., "System R ...".
STONEBAKER M. et a., "The design and implementation ...".

[2.7] RENAUD O., "Le système de base de données ...".

[2.8] ADIBA M., DELOBEL C., cité en [2.3] p. 7.1 ss.
CODD E.F., "A data base sublanguage ...".
WEDEKIND H., cité en [2.5] p. 130 ss.

[2.9] BOYCE R.F. et a., "Specifying Queries ...".
WEDEKIND H., cité en [2.5] p. 163 ss.

[2.10] BOYCE R.F. et a., "Using a structured ...".
CHAMBERLIN D.D. et a., "Sequel ...".
WEDEKIND H., cité en [2.5] p. 163 ss.

[2.11] ADABAS, Adascript user manual.

[2.12] IBM, "IQF ...".

[2.13] ADIBA M., DELOBEL C., cité en [2.3] p. 7.15 ss.
WEDEKIND H., cité en [2.5] p. 115 ss.

[2.14] ADIBA M., DELOBEL C., cité en [2.3] p. 7.2 ss.
WEDEKIND H., cité en [2.5] p. 130 ss.

[2.15] BACHMANN W., "The programmer ...".

[2.16] CODD E.F., "Seven steps to rendez-vous".

[2.17] ZLOOF M.M., "Query by Example".

[2.18] CODASYL, "Feature Analysis" p. 21 ss.

[2.19] GIBERT A., "Les banques de données" tome II,
p. 4.1 ss.
JOUFFROY C., LETANG C., "Les fichiers ...".
MARTIN J., cité en [2.5] p. 197 ss.
WEDEKIND H., "Systemanalyse".
WEDEKIND H., "Datenorganisation".

[2.20] DATE C.J., cité en [2.1] p. 171 ss.
IBM, cité en [2.1].

[2.21] IDMS, user manual.

[2.22] DELOBEL C., cité en [2.3] p. 109 ss.

[2.23] Voir références citées en [2.5].

[2.24] MARTIN J., cité en [2.5] p. 404 ss.
WEDEKIND H., "Datenbanksysteme II" p. 122 ss.

[2.25] HAERDER T., "Auswahl von Zugriffspfadstrukturen".
SCHROEDER K., "Vergleich von Verweistechniken ..".

[2.26] Pour plus de détail, consultez entre autre:
GROCHLA E. et a., "Gestaltungskriterien ..."
p. 68 ss.

CHAPITRE 3

PROBLEMES LIES A L'APPROCHE POUR LA REALISATION D'UN SYSTEME INFORMATIQUE DE GESTION

3.1. NECESSITE D'ETABLIR UN MODELE DE L'ENTREPRISE CONSIDEREE

3.1.1. Définition

L'entreprise peut se définir comme étant un système socio-technique très complexe dont l'objectif principal consiste à produire des biens matériels ou immatériels [3.1]. La réalisation de ces objectifs impliquent l'exécution d'une foule d'activités que l'on peut regrouper en trois catégories [3.2]:

1. activités opérationnelles
2. activités décisionnelles ou dispositives
3. activités stratégiques.

Ces trois groupes d'activités peuvent être considérés comme des systèmes partiels reliés entre eux par des flux d'information. Chacun de ces systèmes reçoit et envoie des informations à l'environnement (fig. 3.1).

Le système de gestion a pour objectif principal la détection des changements dans l'environnement de l'entreprise et de fixer les mesures adéquates afin de garantir le bon fonctionnement et l'existence de l'entreprise. Le système de gestion peut se subdiviser en deux sous-systèmes [3.3]:

- le système d'information
- le système de décision

Quant au système informatique, il englobe tous les processus de traitement de l'information et les prises de décisions automatisables avec l'aide de l'ordinateur. Une partie plus ou moins importante des activités de ces deux sous-systèmes forme le système informatique (figure 3.2), le degré d'automatisation dépendant entre autre:

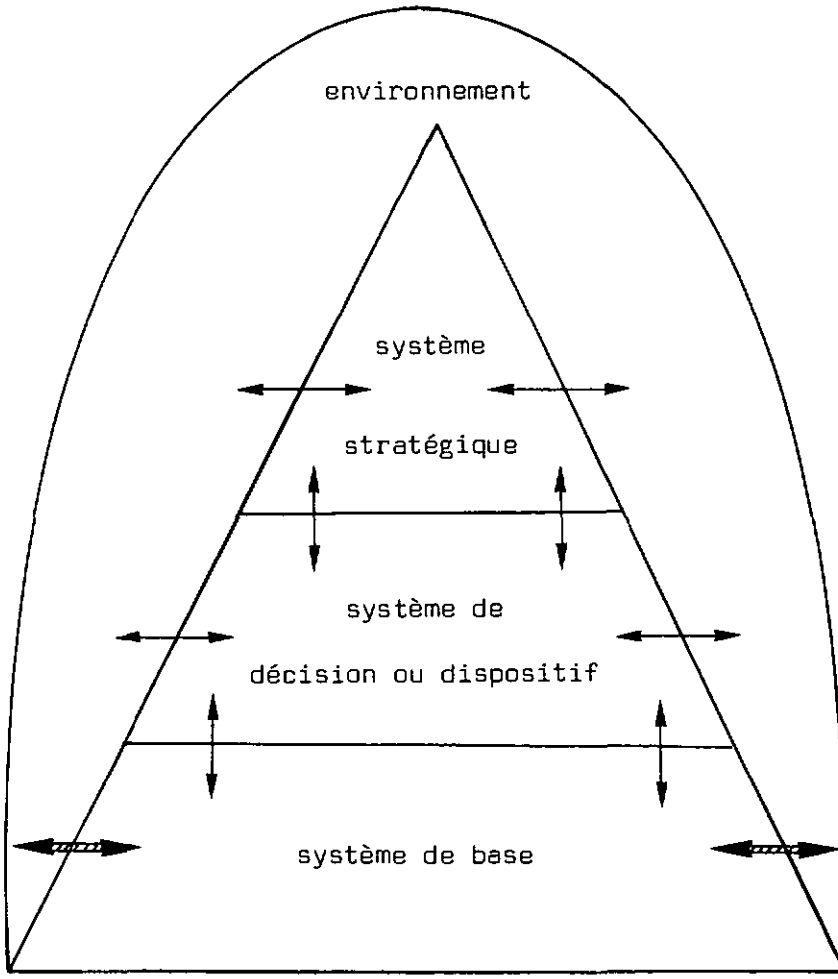


Fig. 3.1 - Hiérarchie des systèmes

Flux de matières et d'informations

Flux d'informations

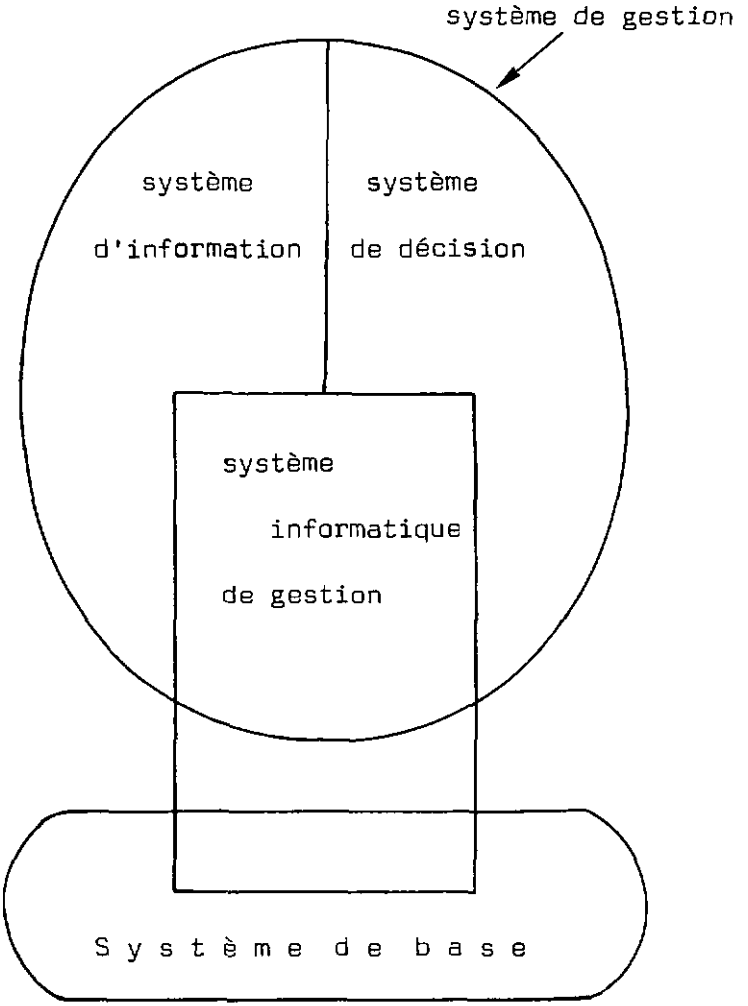


Fig. 3.2 - Interférences des systèmes

- des objectifs fixés quant à la réalisation de ces deux sous-systèmes
- des possibilités de formalisation
- du niveau technologique
- du niveau des connaissances.

3.1.2. Objectifs d'un modèle

La complexité et les exigences auxquelles doit satisfaire le système de gestion augmentent constamment. D'autre part, le rythme des modifications de l'environnement s'accentue. Afin de pouvoir maîtriser cette situation, le gestionnaire exige une masse d'information supplémentaire.

La modélisation du système de gestion s'impose, car elle empêche le développement prolifique d'îlots de mécanisation. L'une des causes de cette multiplication de systèmes isolés réside dans l'incapacité à maîtriser les interrelations entre les applications. D'autre part, l'absence de méthodes cohérentes permettant une intégration progressive favorisa cette évolution. Un autre phénomène aggrava la situation: l'inexistence d'objectifs en vue de délimiter les champs d'application et les priorités dans la réalisation des systèmes.

Afin de contrer efficacement ces faiblesses, le processus de modélisation doit satisfaire aux conditions suivantes:

- éviter que des parties de systèmes utilisables dans différents secteurs de l'organisation ne soient développées plusieurs fois
- délimiter le domaine d'application du système
- fixer les objectifs afin de pouvoir évaluer les solutions proposées

- favoriser une intégration progressive
- contribuer à un enchaînement logique des réalisations
- garantir un degré d'adaptation compte tenu de l'évolution future
- faciliter les liaisons à tous les niveaux de l'organisation
- maîtriser les coûts de réalisation.

La réalisation d'un modèle a pour objectif de décrire les caractéristiques principales du système considéré, étant donné qu'il est pratiquement impossible de prendre en compte la totalité des aspects. Il faut donc faire abstraction de certains éléments de la réalité et ne reproduire que les critères influençant directement la conception du système [3.4].

A l'aide d'un modèle, il est relativement facile de simuler les répercussions provoquées par la modification de l'un ou l'autre des critères considérés. De même, certaines éventualités futures doivent être intégrées et analysées au niveau du modèle. Des prévisions quant aux changements possibles dans le futur s'imposent donc.

De plus, le modèle joue un rôle d'unificateur en ce sens qu'il offre une vision unique du monde réel. Ne sous-estimons pas cette caractéristique, car les personnes concernées par l'automatisation d'un système ont tendance à appréhender la réalité de manière différente. Le risque de provoquer des interprétations divergentes des faits réels représentés peut donc se ramener à des limites acceptables.

3.2. LES METHODES D'APPROCHE

De manière générale, une telle méthode doit servir de guide en indiquant les étapes à suivre, les activités à effectuer et proposer des techniques de travail. Elle doit donc faciliter la tâche du concepteur.

Comme nous l'avons signalé, toute réalisation doit se baser sur un modèle. Par conséquent, l'approche s'effectuera dans une perspective système en vue de la création d'un modèle évolutif représentant les éléments du système et leurs interrelations. Seul un processus de conception itératif permet l'obtention de résultats cohérents et conformes aux objectifs. La méthode implique les étapes suivantes:

- fixation des objectifs
- analyse
  - . des fonctions principales des centres d'activité
  - . des besoins en information
  - . des flux d'information
  - . de la structure des données
  - . des processus de traitement
  - . des contraintes technologiques, quantitatives et temporelles
- synthèse
- confrontation des résultats et des objectifs
- si nécessaire, reprendre au début.

Aujourd'hui, le processus de conception d'un système informatique est tout autre que systématisé. Les méthodes proposées traitent le problème de façon ponctuelle, alors qu'un cadre de référence plus vaste s'impose.

3.2.1. Essai de classification des méthodes d'approche existantes

En considérant les perspectives sur lesquelles ces méthodes se basent, nous proposons la classification suivante:

a) Approche partielle

Selon ce principe, l'analyse détaillée et exhaustive de la situation actuelle se limite à un champ d'application restreint que l'on désire automatiser indépendamment des autres applications. Les avantages (+) et inconvénients (-) peuvent se résumer de la manière suivante:

- néglige les développements futurs
- ne favorise pas l'analyse de structure
- tend à aborder des problèmes n'ayant aucune incidence sur la conception
+ le concepteur connaît la situation actuelle en détail et peut donc mieux défendre son point de vue
+ limite le risque de réaliser des systèmes inadéquats.

b) Aproche globale ou "total system approach"

Diamétralement opposée à la précédente, cette méthode préconise de concevoir et de réaliser le système de gestion en tant que système complet. Les faiblesses de cette façon de procéder se résument ainsi:

- la durée du projet peut être telle que le concept soit dépassé au moment de sa mise en exploitation
- impossibilités d'accumuler des expériences partielles
- l'implantation se faisant attendre, l'intérêt et la motivation des futurs utilisateurs

diminuent rapidement.

c) Approche système

Méthode médiane [3.6] dont le principe se base sur l'élaboration d'un concept global du système de gestion suivi d'un découpage en sous-système. Cette solution favorise la réalisation par étape.

3.2.1. Techniques d'aide à la conception

Indépendamment des méthodes d'approche, différentes techniques d'aide à la conception, dont l'objectif primaire consiste à faciliter le travail du concepteur, ont vu le jour.

a) Techniques non automatisées

Le but de telles techniques réside dans la description et la documentation des caractéristiques du système considéré. Parmi les plus connues citons:

- l'organigramme
- le graphe
- l'ordinogramme (flow-chart)

La complexité, le nombre des paramètres à prendre en considération et le manque de rigueur dans l'utilisation de ces techniques poussèrent à la création d'outils plus perfectionnés.

b) Automation partielle

Automatiser le passage de l'analyse vers la programmation constitue l'un des principaux objectifs de ces techniques. D'autre part, des solutions partielles utilisables dans un cas particulier, par exemple au niveau de la documentation des programmes, sont

proposées. Une caractéristique spécifique à ces techniques réside dans leur orientation vers la programmation.

Les méthodes d'analyse sont les représentants type de cette catégorie, entre autre les "packages" Ariane et Protée [3.8]. D'autre part, des processeurs de tables de décision [3.9], de documentation de programmes [3.10] ou des algorithmes de traitement de matrice pour l'analyse des flux d'information [3.11] font également partie de ce groupe. Des systèmes ont été développés dans le but d'intégrer plusieurs phases de la conception. Citons entre autre Dataflow, Autosate et Cadis [3.12].

En ce qui concerne l'aide à la conception d'une banque de données, différentes réalisations existent:

- analyse de cohérence du modèle des données au niveau conceptuel [3.13]
- passage du modèle logique au modèle physique [3.14]
- les algorithmes proposés dans le cadre du projet Ansi/Sparc [3.15], des modules d'analyse et de transposition de modèles

L'inconvénient majeur de ces techniques provient du fait qu'elles proposent des solutions ponctuelles et ne se réfèrent à un système global.

c) Automation complète

L'objectif de ces méthodes consiste à automatiser entièrement toutes les phases de la conception et de la réalisation. Les énoncés des problèmes à résoudre se transmettent en entrée à la machine qui se charge de proposer en sortie des solutions. Le principal représentant de cette tendance est le système Isdos [3.16].

L'utilisation de telles méthodes est problématique:

- il est difficile de formaliser toutes les phases du processus de conception et de les automatiser
- des difficultés apparaissent lorsqu'il s'agit de concevoir des activités qui impliquent une suite de prises de décision
- le problème de la quantification se pose avec certains critères influençant la conception
- il n'est pas toujours judicieux d'utiliser des algorithmes, étant donné l'inexactitude des paramètres à prendre en considération et la durée des traitements pouvant dans certains cas largement dépasser les limites tolérables
- l'analyse de la sémantique des données fait défaut et ne peut être automatisée
- la conception est en grande partie un processus créatif et innovatif. L'automatisation ne semble pas encore possible et les méthodes d'optimisation existantes sont insuffisantes, étant par trop déterministes
- l'automate doit pouvoir résoudre tous les problèmes posés par la conception et la réalisation.

REFERENCES

[3.1] HABERFELLNER R., dans "Die Unternehmung als dynamisches System" p. 16 ss., définit la notion de système et indique des critères de classification afin de mieux pouvoir cerner la notion de système qui est sujette à différentes interprétations.

MELESE J., dans "L'analyse modulaire des systèmes de gestion" p. 41 ss., distingue le système technologique, de pilotage, d'information et de mesure. Il cite également d'autres définitions avec les références correspondantes.

[3.2] GROCHLA E., MELLER F., dans "Datenverarbeitung in der Unternehmung" p. 21 ss., ne distinguent que deux catégories.

[3.3] Consultez LE MOIGNE J.L., "Les systèmes d'information dans les organisations".

[3.4] GROCHLA E., dans "Das Kölner Integrations-Modell (KIM)" p. 22 ss., énumère différents types de modèles.

[3.5] DANIELS A., YEATES D., "Systemanalyse"
KLEIN, "Betriebliche Organisation - vom Ist zum Soll".
SYSTOR, "So verwirklichen wir EDV-Projekte".

[3.6] MELESE J., "Analyse modulaire des systèmes".
MZG, "Systementwicklung".

[3.7] BLUMENTHAL S.C., "Système de gestion..." p. 20 ss.

[3.8] DOSSIER, "Méthodes d'analyse" p. 35 ss. et les rérérences citées.

[3.9] ROMERA S., "Table de décision..." p. 66 ss.
THURNER R., "Entscheidungstabellen".

[3.10] ZEDA, "Quick-Draw".

[3.11] GAGSCH S., "Eine Methode zur computer-gestützten Vorbereitung..." p. 91 ss.

[3.12] NICK F. et a., "Systeme der computergestützten Systemgestaltung" p. 17 ss.

[3.13] LEONARD M., "Aides algorithmiques à la conception de bases...".

[3.14] TEOREY T.J., DAS K.S., "Application of an Analitical Model..." p. 9 ss.

[3.15] ANSI/X3/SPARC, "Interim Report" p. 8 ss.

[3.16] LESUISSE R., "Exposé introductif au projet Isdos" p. 163 ss.
TEICHROEW D., SIBLEY E.,"Isdos Phase I".
TEICHROEW D., SAYANI H., "Automation of System Building" p. 379 ss.

CHAPITRE 4

METHODE D'APPROCHE PROPOSEE

4.1. CARACTERISTIQUES DE LA METHODE D'APPROCHE

En premier lieu, la méthode doit satisfaire aux objectifs définis au chapitre 3.2. Le problème de la conception d'une banque de données ne peut en aucun cas être considéré individuellement. Il est à intégrer au processus de conception d'un système informatique de gestion, dont il est un des éléments.

Il est nécessaire d'insister sur ce point car cet axiome est très souvent ignoré ou en tout cas on ne lui attache que trop peu d'importance. Les conséquences qui en découlent peuvent prendre de l'importance:

- les exigences de l'utilisateur ne peuvent être satisfaites, à l'extrême le système ne peut être utilisé
- le volume des traitements ne peut être maîtrisé conformément au délai fixé
- le système peut ne pas être assez performant
- incompatibilité entre sous-systèmes.

Conformément à la classification des différentes méthodes d'approche décrites au chapitre 3.2.1 et en tenant compte des avantages et inconvénients respectifs, la méthode devra être du type "analyse système". L'idée de base consiste à étudier en premier lieu les objectifs et le fonctionnement du système de gestion, de réaliser un modèle global et de le subdiviser en sous-systèmes qui seront analysés individuellement par étape successive à degré de précision croissant. La réalisation de ces sous-systèmes ne doit pas se faire obligatoirement en parallèle. Il est possible et même souhaitable de la répartir dans le temps. Cette mise en oeuvre progressive du système de gestion permet de raccourcir le processus de réalisation, ce qui minimise le risque de mettre en exploitation un système inadapté aux besoins du moment.

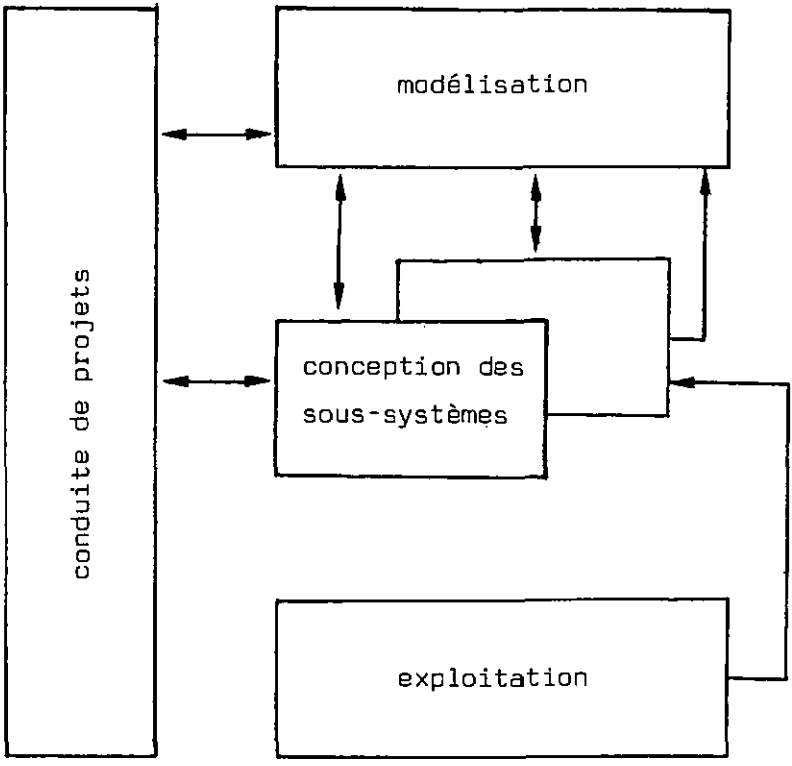


Fig. 4.1 - Eléments de la méthode

Cette approche hiérarchique est caractérisée par deux critères :

- il est nécessaire de distinguer entre la conception d'un modèle du système de gestion (4.1) et la conception des sous-systèmes
- la conception de la banque de données s'effectue parallèlement à la conception du ou des sous-systèmes, selon le degré d'intégration.

De cette approche en parallèle, il en résultera un système informatique de gestion formé de 2 éléments: la banque des données et la banque des programmes.

Les activités prévues dans le cadre de la méthode d'approche sont subdivisées en phases, chacune d'elle nécessitant une planification. Celle-ci consiste à définir les activités à effectuer et leur déroulement chronologique, ainsi qu'à fixer les méthodes à utiliser et les moyens à disposition (homme, matériel et finance). Un contrôle continu des plans élaborés est une condition impérative à la réalisation des objectifs.

De la chronologie des phases, il ne faut pas en déduire que celles-ci seront strictement réalisées les unes après les autres. L'approche est un processus itératif dans les deux sens. En parcourant une phase, il est judicieux de jeter un coup d'oeil sur les phases suivantes, afin de pouvoir tenir compte de certaines contraintes. Il est par exemple important de tenir compte assez tôt de contraintes découlant du passage d'un système existant au nouveau système. De même, il est nécessaire de faire la critique de la cohérence des résultats obtenus à la phase précédente par rapport aux problèmes à résoudre dans la phase en cours. Des adaptations constantes de résultats déjà acquis doivent être évitées.

Dans cette étude, nous nous bornerons à développer les activités ayant une influence directe sur la conception de la banque de données, les autres ne seront que citées, afin d'avoir une vue d'ensemble complète et de pouvoir situer la conception d'une banque de données dans le contexte conception d'un système informatique de gestion.

4.2. CONCEPTS DE BASE DE LA METHODE PROPOSEE

La représentation de la structure logique de la banque de données a été très discutée ces dernières années et d'importants travaux de recherche ont apporté différentes solutions que l'on peut classer en trois catégories: le modèle réseau (4.2), les hiérarchies étant considérées comme cas particulier du réseau (4.3), le modèle relationnel (4.4), le modèle des relations entre entités ou relations binaires (4.5). Certains auteurs, comme Date (4.6), ne font aucune distinction entre les deux derniers modèles et ne considère que les modèles réseau, hiérarchique et relationnel. Par contre, Delobel (4.7) classe les modèles en quatre catégories et différencie les deux types de relation n-aires et binaires.

Le but de cette étude n'est pas de développer un nouveau type de modèle, mais plutôt d'analyser les avantages et inconvénients des modèles existants et de les faire intervenir dans le processus de conception en fonction des objectifs fixés, dont le principal est l'indépendance envers toutes contraintes d'implantation, ce qui implique:

a) une analyse sémantique des faits réels indépendamment des modes d'utilisation. Ce qui importe, ce n'est pas le besoin en information, par exemple une liste des clients ayant une facture de plus de 1.000 F, mais l'existence d'éléments d'information tels que "client", "facture" et "montant", leurs significations et les relations qui les associent. Cette approche, indépendante de toutes contraintes de réalisation et de performance, facilitera l'étude de la dynamique du modèle:

. une modification des requêtes d'une application, par exemple à la liste des clients citée s'ajou-

te une liste des clients d'un secteur donné et avec un chiffre d'affaires de plus de 1.000 F, ne doit impliquer une modification du modèle
- évaluer l'influence d'utilisations potentielles, déduisibles du modèle, sur la structure logique à élaborer au niveau suivant
- l'apparition de nouvelles catégories d'objets et de relations ne doit remettre en question le modèle et ni en exiger sa reconstruction complète.

b) un haut degré d'indépendance entre les différentes phases de l'approche afin qu'une modification à un niveau donné ne nécessite la reconception du niveau précédent (4.8). Une modification de la réalisation physique d'un chemin d'accès existant ne doit influencer le modèle des chemins d'accès logiques.

c) la description des besoins en information se fera en fonction du modèle ainsi élaboré, au lieu de déduire la structure directement à partir des besoins en information (4.9). Le risque d'omettre un fait élémentaire du monde réel est minimisé, puisque l'analyse ne se fait pas en fonction des utilisations qui varient dans le temps et qu'il est difficile de fixer à l'avance, mais selon la signification des objets et les relations entre ceux-ci.

d) l'utilisateur doit pouvoir formuler ses besoins en information de manière systématique à l'aide d'une technique qui n'exige pas de connaissances approfondies et spécialisées, ce qui facilitera sa participation à la conception. La collaboration de l'utilisateur est indispensable, puisqu'il connaît les faits du monde réel considéré et peut donc contribuer activement à l'analyse de la cohérence du modèle.

e) le modèle doit être neutre, c'est-à-dire ne pas dépendre d'un SGBD donné afin que la description

du modèle à l'aide du SGBD choisi puisse s'effectuer sans difficultés. Cette condition ne doit pas être négligée car, dans la pratique, la conception d'une banque de données aboutit toujours à une utilisation dans le cadre d'un système informatique de gestion et les contraintes de performance doivent être satisfaites. Seuls des SGBD ayant fait leurs preuves, les entreprises qui font dans ce domaine oeuvre de pionniers sont en minorité, seront pris en considération, par exemple pour le matériel IBM: Adabas, IDMS, IMS, Socrate, etc.

f) les critères d'implantation physiques, tels que méthode d'organisation, méthode d'accès, doivent être étudiés indépendamment et sans influencer les aspects logiques. Cette distinction entre les aspects physiques et logiques apparaît déjà dans les premiers travaux sur les SGBD (4.10).

g) le modèle doit être sémantiquement non-redondant, cohérent et ne pas contenir d'ambiguïtés afin que des opérations de manipulation au niveau des données ne provoquent un état inconsistant de la banque de données.

4.3. LES PHASES DU CYCLE DE DEVELOPPEMENT D'UN SYSTEME INFORMATIQUE DE GESTION

De la diversité des techniques et méthodes proposées dans la littérature spécialisée (voir chapitre 3.2), l'on peut en déduire une seule caractéristique commune: toutes proposent au concepteur de l'aider à résoudre les difficultés qu'il rencontrera au cours du cycle de développement d'un projet, mais la plupart diffèrent par les solutions proposées pour remédier à ces difficultés. Entre autre, en ce qui concerne les

Couger [4.12] ISDOS [4.13]	Documentation du système existant	Design logique	Design physique	Programmation	Implantation	Opération	Maintenance
Matthews [4.14]	Synthèse	Analyse	Design et documentation	Implantation		Maintenance	
Wedekind [4.15]	Analyse de situation	Etude d'opportunité	Conception	Implantation		Production	
Chénique [4.16] Castellani [4.17]	Etude d'opportunité	Analyse fonctionnelle	Analyse organique	Programmation		Exploitation	Maintenance

Fig. 4.2 - Comparaison des phases de développement

phases de réalisation d'un projet, le choix est grand, et en plus, la terminologie utilisée diverge fortement. Pour en faciliter la comparaison, nous avons essayé de classer les dénominations des différentes phases par auteur dans un tableau (figure 4.2). Pour plus de détails, nous proposons de consulter les références citées dans ce tableau.

Toutes les méthodes citées ont l'inconvénient majeur d'ignorer le problème de la conception d'une banque de données. Elles sont axées sur la réalisation de systèmes informatiques de gestion utilisant des fichiers conventionnels (4.11). D'autre part, elles ne facilitent pas la participation active des utilisateurs au processus de la conception logique du système, car trop de connaissances techniques en informatique sont exigées.

Remarque : Dans le cadre d'un projet informatique, nous avons participé à la mise en place de la méthode d'analyse standardisée ORGWARE II (4.18). Au cours de son utilisation, les lacunes énumérées ci-dessus ont fait leur apparition. Des adaptations de l'approche proposée étaient inévitables. Récemment, les constructeurs ont fait de nouvelles propositions (4.19), mais celles-ci ne sont pas conformes aux objectifs fixés au chapitre 3.2 et aux concepts de base formulés au cours de ce chapitre 4.

La conception d'un sous-système (voir figure 4.3) se déroulera selon deux méthodes d'approche qui s'effectueront en parallèle et qui ont pour nom:

- conception de la banque de données
- conception de l'application

Celles-ci sont subdivisées en trois modules :

- conception
- réalisation
- implantation.

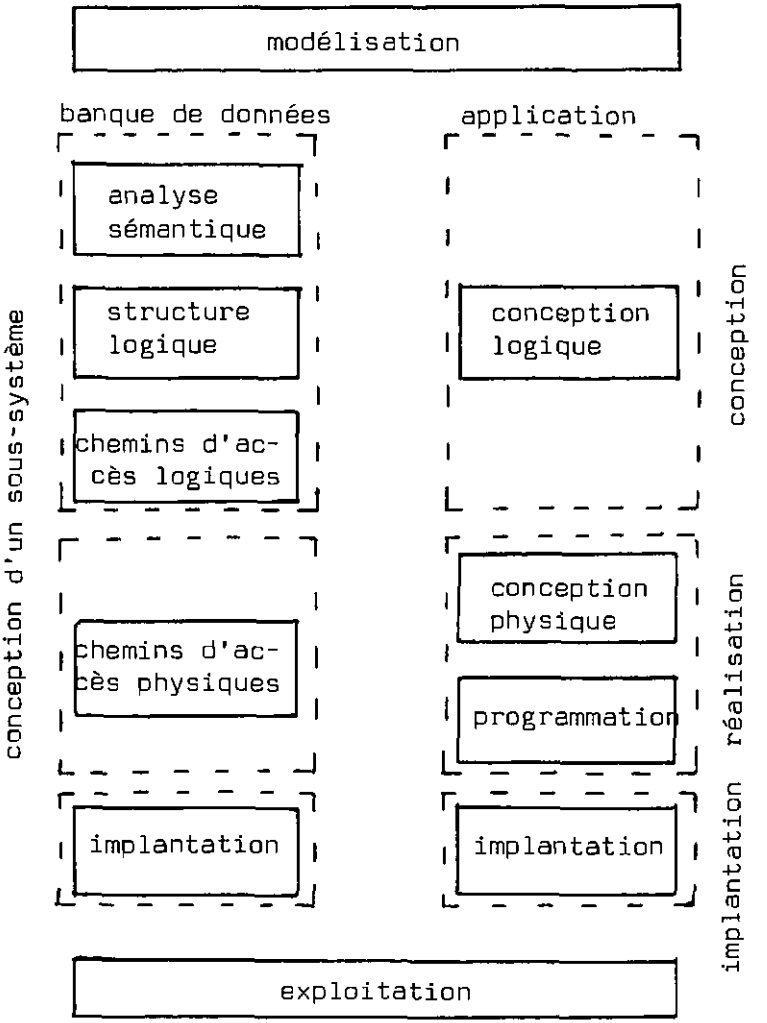


Fig. 4.3 - Modules de l'approche

Les activités des deux méthodes s'effectueront imbriquées les unes dans les autres et il sera relativement difficile de dire si une activité donnée appartient à l'une ou l'autre des méthodes. Par exemple, l'analyse des besoins en information des utilisateurs concernés est une activité nécessaire à la conception de la banque de données ainsi qu'à la réalisation des supports de données en sortie. Il est évident qu'une telle activité ne sera exécutée qu'une seule fois.

Les raisons qui nous ont poussés à proposer deux méthodes d'approche pour la conception d'un système informatique de gestion sont les suivantes:

- la solution adoptée est plus générale et offre un cadre de référence plus large
- il est possible d'utiliser des méthodes et techniques de conception partielles à des niveaux bien précis de notre approche
- l'optique 'procédures de traitement' et l'optique 'données' sont étudiées séparément
- l'approche proposée peut servir de guide, même lorsque la création d'une banque de données n'est pas prévue
- l'approche est également valable dans le cadre de concepts tels ceux développés par le groupe ANSI/SPARC (4.20)
- la méthode n'impose pas à priori la réalisation d'une seule banque de données pour l'ensemble du système d'information. Au contraire, elle oblige le concepteur à étudier ce problème de manière objective.

4.4. LES ACTIVITES PREVUES A CHAQUE PHASE DU CYCLE DE DEVELOPPEMENT D'UN SYSTEME INFORMATIQUE DE GESTION

Au cours du chapitre 4.1, nous avons insisté sur l'importance de l'intégration du processus de conception d'une banque de données dans une approche en vue de la réalisation d'un système informatique et sur les avantages d'une approche hiérarchique.

Par conséquent, nous proposons de fixer en premier lieu les objectifs auxquels doit satisfaire le système de gestion afin de passer ensuite à la modélisation de ce système. Une liste des activités à effectuer est présentée en annexe 1.1 en vue de faciliter la compréhension et l'utilisation pratique de la méthode d'approche que nous proposons.

En ce qui concerne les activités propres à la conception des applications, nous nous sommes bornés à les énumérer en annexe 1.2, conformément à un des buts de notre étude, c'est-à-dire proposer une méthode en tant que cadre de référence général.

Le grand nombre de critères à considérer lors de la conception d'une banque de données, de même que l'impossibilité de les analyser simultanément et le respect de l'indépendance des données face à des critères d'implantation, justifie une approche par étape. Nous proposons de subdiviser cette méthode d'approche en cinq phases:

1. Analyse sémantique
2. Structure logique
3. Chemins d'accès logiques

4. Chemins d'accès physiques
5. Implantation.

Cette délimitation est conforme aux concepts de base formulés au chapitre 4.2. Elle s'inspire en partie du modèle DIAM proposé par Senko (4.21), mais en diffère par le fait que Senko, dans son approche, ne prévoit pas d'analyse sémantique. Il ne considère que les vues partielles des utilisateurs. Il est par conséquent sans autre possible que certains aspects du monde réel soient ignorés, que le modèle contienne des redondances sémantiques et même une certaine ambiguïté. L'introduction d'une phase supplémentaire, dont l'objectif est de remédier à ces inconvénients, s'impose.

La proposition du groupe ANSI/SPARC (4.20) de réaliser une approche en trois niveaux

1. conceptuel
2. interne
3. externe

ne permet pas de distinguer explicitement la structure logique, les chemins d'accès logiques et physiques. Cette séparation s'impose, car elle facilite le passage du niveau logique à la description de la structure physique. D'autre part, une telle distinction simplifie le travail du concepteur, les critères à respecter sont regroupés, la démarche est plus structurée et le risque de sous-estimer certains aspects importants est limité.

L'interprétation proposée en référence (4.22) est déjà plus proche du concept présenté par notre étude. Quant à l'approche décrite en (4.23), elle est basée sur le modèle 'Codasyl', mais implique une analyse des données et des caractéristiques des applications avant de passer à l'élaboration du 'schema' et des 'subschemas'.

Les activités relatives à ces phases sont énumérées en

annexe 1.3. Au cours des chapitres 6 et suivants, nous passerons à leur description détaillée.

L'interdépendance entre les phases des deux méthodes d'approche (banque de données et applications) peut s'illustrer à l'aide du graphe de la figure 4.4. Les problèmes qui en résultent et les répercussions sur le déroulement des activités concernées seront traités dans les chapitres relatifs aux phases correspondantes.

Pour compléter l'énumération des activités, nous citons à titre d'exemple en annexe 1.4, les principales activités propres à la phase d'exploitation. Les problèmes spécifiques à une banque de données seront traités au cours du chapitre 10.

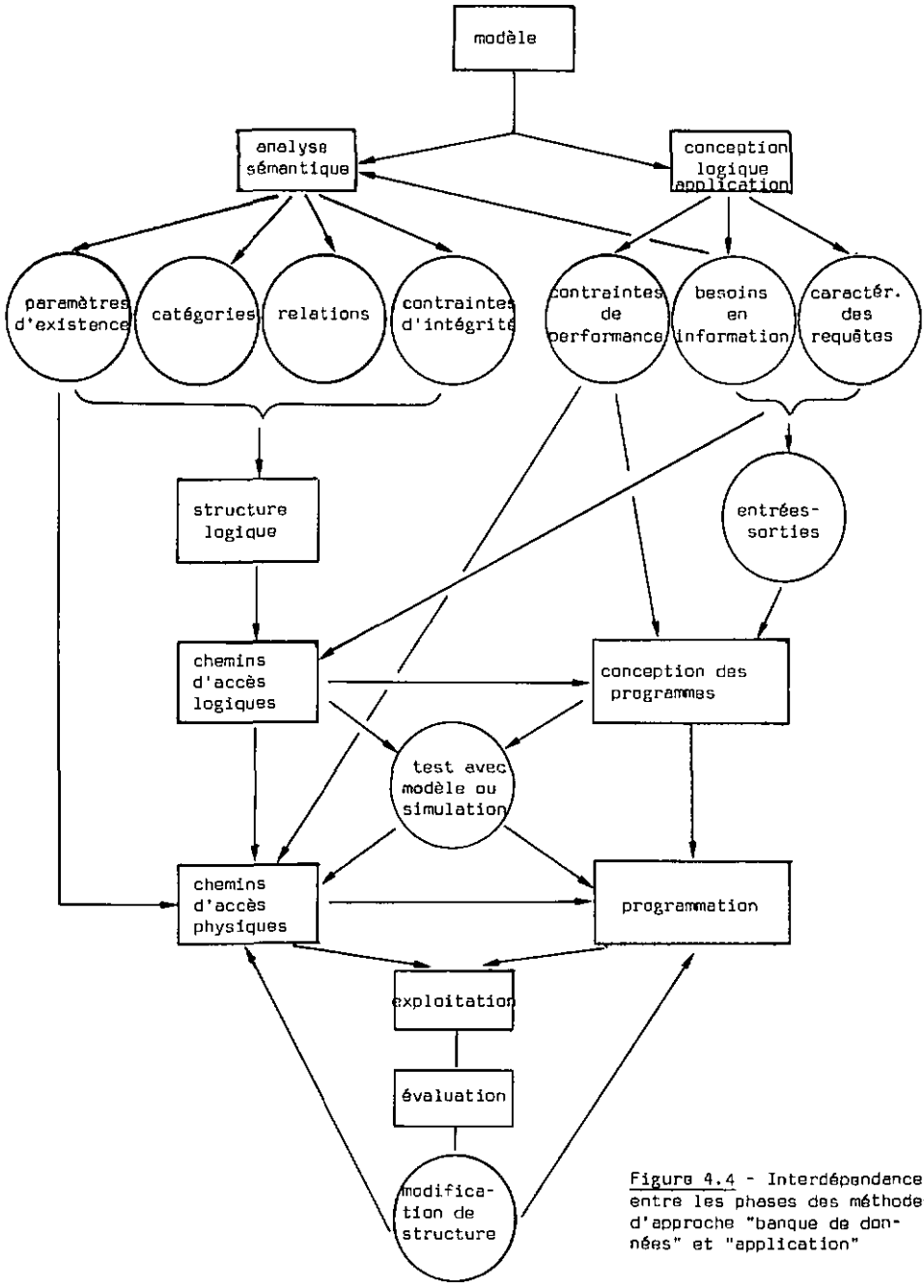


Figure 4.4 - Interdépendances entre les phases des méthodes d'approche "banque de données" et "application"

REFERENCES

[4.1] Définition au chapitre 3.1

[4.2] CODASYL "DBTG Report".

[4.3] WEDEKIND H., "Datenbanksysteme I", p. 31.

[4.4] CODD E.F., "Relational Model for Large ...".
"Further Normalization ...".
DELOBEL C., "Les systèmes de banques de ...".
"Contributions théoriques ...".
DATE J.C., "An Introduction ...", p. 61 ss.

[4.5] ABRIAL J.R., "Data Semantics".
SENKO M. et a., "Data Structures and Accessing ...".

[4.6] DATE C.J., cité en [4.4], p. 41 ss.

[4.7] DELOBEL C., cité en [4.4], p. 8 ss.

[4.8] SENKO M. et a., cité en [4.5], p. 64 ss., fut un des premiers à formuler cette notion.
WEDEKIND H., "Datenbanksysteme II", p. 117 reprend cette idée.

[4.9] GILLNER R., dans "Die Strukturierung ...", p. 104 ss., déduit la structure logique directement des besoins en information.

[4.10] CODASYL cité en [4.2].
EDP ANALYSER "Organizing the Corporate ...".
LYON J.K., "An Introduction to Data Base ...".
WILL H.J., "Datenbank-Systeme", p. 817 ss.

[4.11] YVON P.J. et SEMIN C., dans "Comment concevoir un système ...", p. 192 ss., proposent d'étudier la conception de la banque de données séparément de la conception des applications.

[4.12] COUGER J.D. et KNAPP R.W., "System Analysis Techniques".

[4.13] LESUISSE R., "Introduction au projet ISOOS".

[4.14] MATTHEWS D., "The Design of the management ...".

[4.15] WEDEKIND H., "Systemanalyse".

[4.16] CHENIQUE F., "Analyse fonctionnelle et organique".

[4.17] CASTELLANI X., "Méthode générale d'analyse ...".

[4.18] ADV-ORGA, "Orgware II".

[4.19] ADV-ORGA, "Orgware IV " et "Orgware V".

[4.2o] ANSI/X3/SPARC, "Interim Report", p. 8 ss.

[4.21] SENKO M.E. et a., cité en [4.5], p. 64 ss. "DIAM".

[4.22] BENCI G. et a., "Introductory Report", p. 18 ss.
CLUB BANQUE DE DONNEES, "Rapport du groupe ...", p. I-1 ss.

[4.23] TOZER E.E., "Oatabase Systems Analysis ...".

CHAPITRE 5

MODELISATION DU SYSTEME DE GESTION

5.1. FIXER LES OBJECTIFS A LONG TERME

Une condition préalable à toute conception consiste à fixer, au niveau de l'entreprise, pour les différents secteurs d'activité, la nature et les moyens de traitement de l'information, les exigences auxquelles un nouveau système doit satisfaire, ainsi que les faiblesses du système existant. Dans la pratique, les objectifs à long terme sont rarement formulés et la première activité consiste à définir ceux-ci.

Ensuite, il faudra élaborer une liste de critères qui servira de base à la prise de décision quant aux priorités de réalisation des objectifs:

- les possibilités de rationalisation
- les avantages qualitatifs que le système peut offrir
- les critères de rentabilité
- une décision politique
- la disponibilité technologique.

5.2. MODELISATION DU SYSTEME

Conformément aux explications du chapitre 3.2.2, éviter la réalisation "d'îlot de mécanisation" est un des buts de toute méthode d'approche. D'autre part, les inconvénients du "total system approach" imposent, poussent le concepteur à délimiter la portée du système à réaliser, que celui-ci utilise ou non un SGBD. La conception d'un modèle comporte deux phases:

a) élaboration d'un modèle global du système

b) étude des sous-systèmes.

5.2.1. Modèle global

La complexité du système de gestion implique une analyse globale des principaux éléments du système informatique de gestion que l'on projette de réaliser. Sont à prendre en considération:

- les centres d'activités
- les activités de traitement de l'information propres à chaque centre (flux internes)
- les échanges d'informations entre les centres, problème central à ce niveau de l'analyse (flux externes).

Une analyse de synthèse intégrant les objectifs fixés doit aboutir à l'établissement d'un modèle global représentant les centres d'activités et les flux externes. Les solutions globales proposées seront ensuite étudiées en détail lors de la conception des différents sous-systèmes. Il est absolument nécessaire, à ce niveau de l'analyse, de ne pas étudier trop minutieusement chaque élément et interrelation du système de gestion, mais de concevoir un modèle en fonction des objectifs fixés pour ce système. Le but de cette opération ne consiste pas à automatiser les activités et les règles de fonctionnement actuellement en vigueur. En général, celles-ci ne permettent pas d'atteindre les objectifs fixés d'une façon satisfaisante et peuvent même représenter dans certains cas un obstacle pour la réalisation de ceux-ci.

Un bon remède contre l'attachement au système existant, contre certaines idées préconçues quant aux solutions réalisables consiste, une fois les objectifs et les principes de fonctionnement du système de gestion fixés, à étudier les systèmes existant dans des entreprises du même secteur d'activité. Il est alors possi-

ble d'en déduire un modèle théorique que l'on confrontera ensuite aux objectifs et aux caractéristiques de l'entreprise considérée, le modèle étant adapté en conséquence. Cette approche est réaliste, car l'utilisation de nouvelles techniques et l'adoption de nouvelles règles de gestion nécessitent très souvent une refonte de la structure des activités.

Une approche standardisée n'aboutirait certainement pas à de bons résultats, car à ce niveau, l'analyse est une activité purement créative. Nous proposons donc de suivre une démarche cohérente (voir annexe 1.1) et d'utiliser des outils de travail facilitant la tâche de conception. En premier lieu, il convient d'élaborer un schéma représentant les centres d'activités et les interrelations entre ceux-ci (figure 5.1). Il servira comme base de départ à la phase de conception des sous-systèmes. Plus le nombre des centres d'activités et des flux circulant entre ces centres est important, plus il devient difficile d'élaborer un schéma clair et lisible. La représentation graphique peut se faire selon la méthode "trial-and-error" ou à l'aide de la méthode proposée par Gagsch (5.1), qui consiste à représenter les flux à l'aide d'une matrice. Un algorithme optimalise l'alignement des centres d'activités en minimisant le nombre des entrecroisements de flux, l'ordre dépendant uniquement des entrées et sorties par centre.

5.2.2. Sous-système

La conception globale des sous-systèmes implique une analyse plus détaillée des centres d'activités représentés au niveau du modèle principal. Notre attention se portera en premier lieu vers les phénomènes suivants:

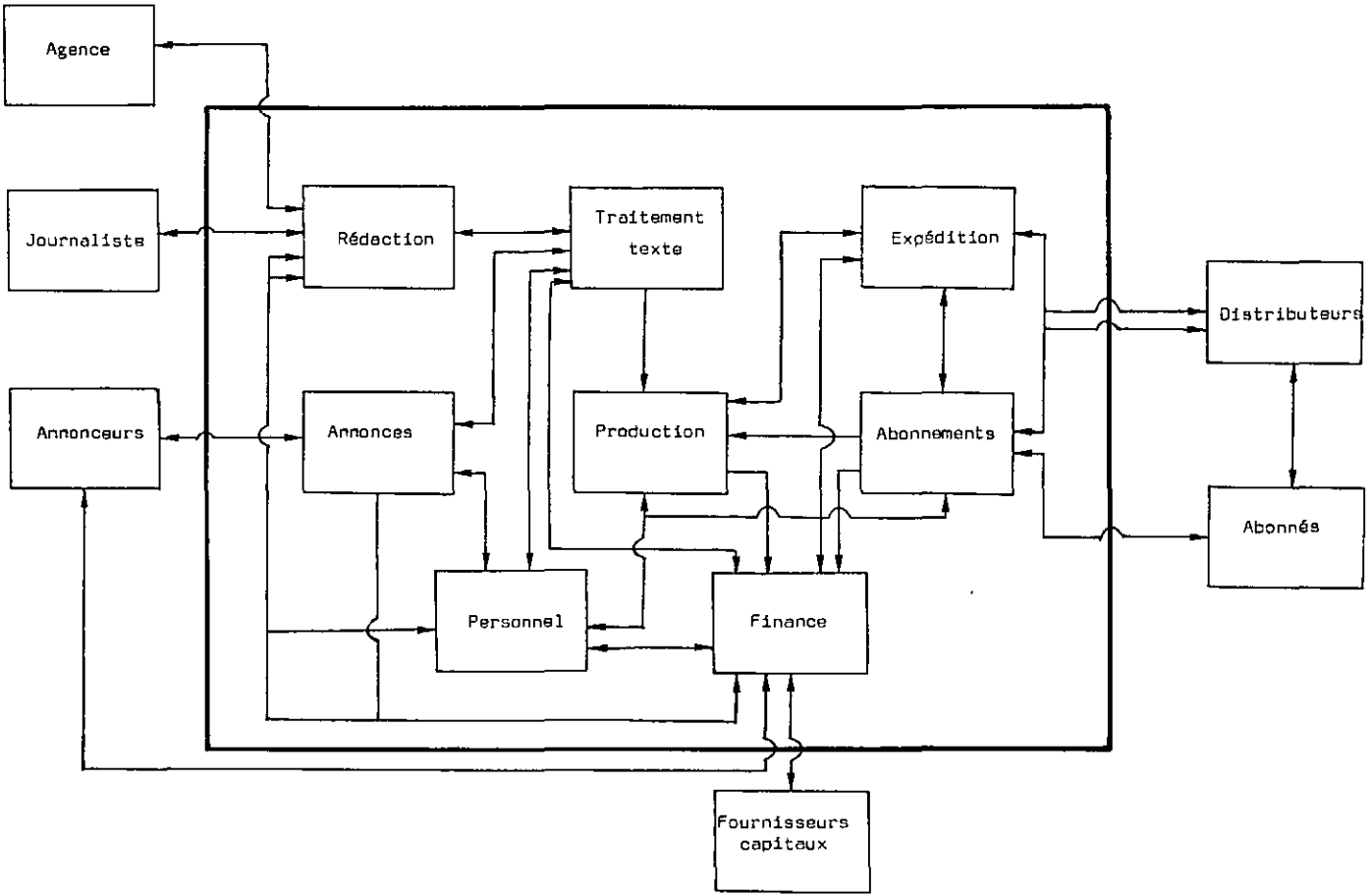


Fig. 5.1 - Exemple d'un modèle global

- nature des activités
- méthodes de traitement
- flux des informations
- besoins en information
- utilisation de ces informations
- faiblesses et avantages du système existant
- capacité de traitement
- volumes et fréquences
- contraintes de délai
- processus de décision.

L'étude détaillée de tous les phénomènes est superflue, il importe d'analyser ceux qui influencent la conception et ceux liés aux objectifs fixés.

La subdivision du modèle global en sous-système est un processus itératif de type évolutif. A chaque itération, de nouvelles connaissances sont acquises, le concept prend forme et dans la plupart des cas une adaptation du modèle global n'est pas à exclure.

Cette stratégie est fondamentalement différente de celle proposée par les méthodes d'approche présentée en général par la littérature spécialisée dans ce domaine particulier, entre autre par les ouvrages cités en référence (5.2).

Les avantages de la méthode d'approche itérative et hiérarchisée que nous développons dans notre étude ne deviennent effectifs que si l'utilisateur, en étroite collaboration avec le concepteur, peut se décider à éliminer à chaque étape les solutions les moins satisfaisantes. La période qui s'écoule entre l'élaboration de propositions et la prise de décision doit être le plus court possible, afin d'éviter le risque d'une baisse d'intérêt pour le projet (5.3).

La délimitation en sous-systèmes implique la prise en considération de plusieurs critères. La préférence accordée à l'un ou à l'autre modifie directement les résultats

- les flux entre les sous-systèmes doivent être peu nombreux et de faible intensité
- les activités qui nécessitent d'intenses et nombreux échanges d'informations sont à regrouper. Cette concentration diminue les interactions entre sous-systèmes et facilite le contrôle des échanges d'information
- niveau technologique
- la nature et le type d'activité
- restrictions de nature socio-psychologique, tels que les pouvoirs acquis par certaines fonctions, les traditions ou le retrait de responsabilités d'un secteur
- nature des traitements et des moyens à disposition
- niveau de rationalisation du système existant
- les objectifs spécifiques
- les contraintes imposées par le modèle global.

Cette énumération n'a pas la prétention d'être exhaustive. Les problèmes à résoudre lors de la modélisation se présentent chaque fois sous une forme complètement différente, ce qui rend inutile toute tentative d'établir des règles de conduites précises.

L'élaboration d'un schéma facilite l'analyse et fait ressortir les caractéristiques principales du sous-système considéré. Gagsh (5.4) propose d'utiliser

son algorithme. A ce niveau de l'étude, cette technique ne peut être utilisée qu'en tant qu'outil d'aide à la représentation graphique. Elle ne considère que l'existence ou non de flux et ne tient pas compte de l'intensité des volumes ni de la fréquence. Par conséquent, nous préférons une représentation graphique à deux niveaux (voir figure 5.2 a et b). Les principes utilisés lors de l'élaboration du modèle global sont également appliqués. Les schémas exigent en général des commentaires explicatifs ajoutés en tant qu'annexes.

Cette approche évolutive nécessite très souvent des études plus détaillées de certains phénomènes qui n'ont été qu'effleurés ou ignorés à l'étape précédente.

Il est possible de représenter dans un tel schéma les supports d'information selon les objectifs fixés (par exemple: utilisation d'écrans cathodiques) ou selon une décision du concepteur qui considère une telle solution comme appropriée. Ce choix ne prédétermine en aucun cas la suite des travaux, la structure des données et les fichiers sont fixés indépendamment de ce choix, seule une analyse plus approfondie permettra de déterminer la nature des supports d'information et la structure de la base de données.

Une autre méthode de description d'un modèle est présentée par Delobel (5.5) qui propose une autre forme de représentation graphique. Les règles de fonctionnement sont décrites à l'aide de relations fonctionnelles, lesquelles permettent, grâce à un processus de décomposition créé par l'auteur lui-même, d'élaborer le modèle conceptuel.

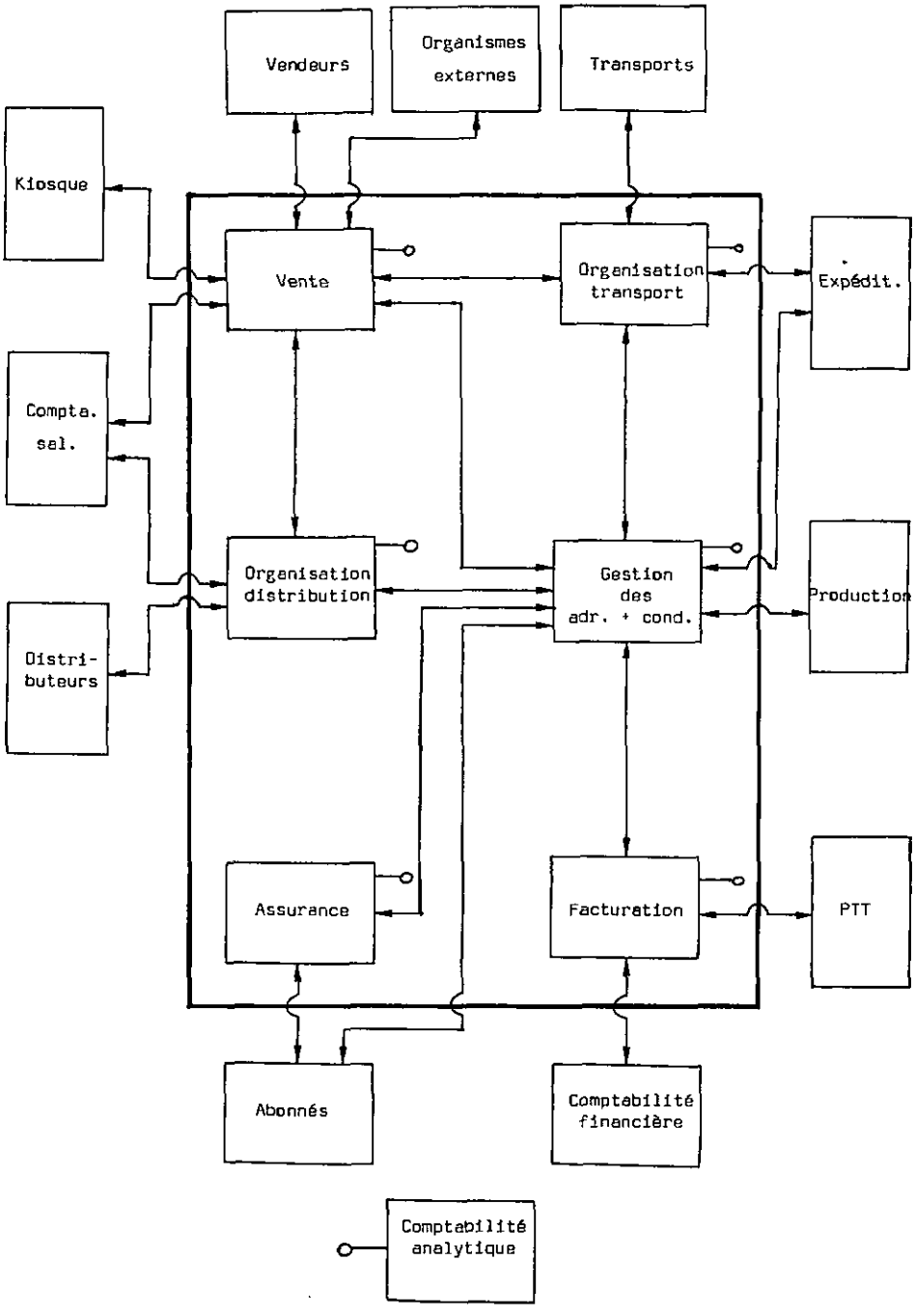


Fig. 5.2.a - Sous-système abonnements (fig. 5.1)

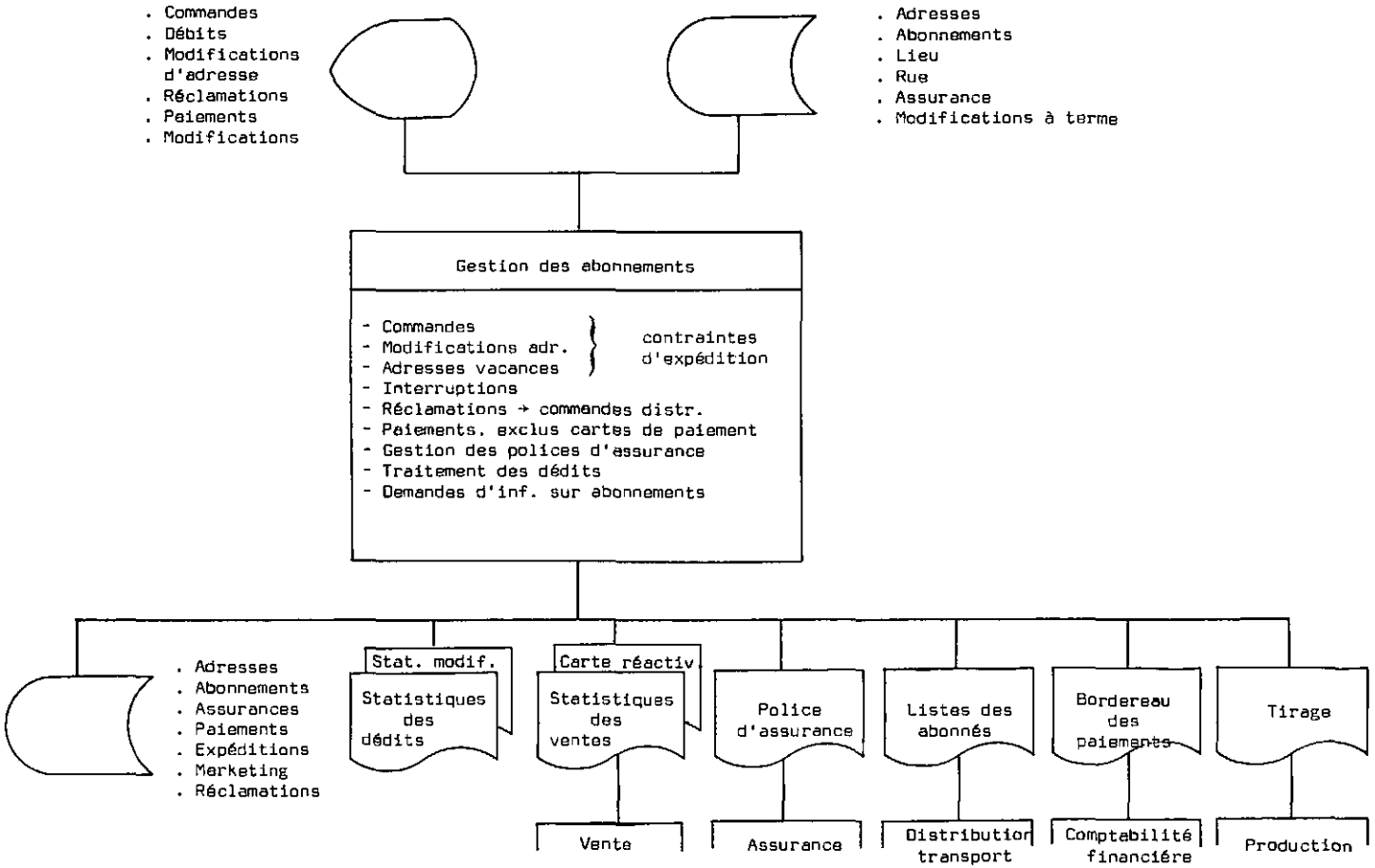


Fig. 5.2.b - Sous-système abonnements (fig. 5.1)

5.3. CHOIX DEFINITIF DU MODELE

A ce stade des études, une prise de décision quant à la suite des travaux et aux sous-systèmes à réaliser est indispensable. Il importe de fixer l'ordre de la réalisation et de déterminer les sous-systèmes à concevoir en parallèle. Une orientation précise des opérations à effectuer est une condition préalable afin de pouvoir respecter les délais et les contraintes de réalisation. Il convient de fixer entre autre:

- le degré d'intégration
  - . quels sous-systèmes seront intégrés en un seul système informatique de gestion ? Très souvent, une analyse plus approfondie des problèmes relatifs aux interdépendances entre les sous-systèmes concernés est nécessaire pour pouvoir répondre à cette question.
  - . réalisera-t-on une banque de données par sous-système ou une intégrant tous les sous-systèmes ? Les critères à considérer sont
    - + la nature des données traitées est-elle fondamentalement différente ?
    - + respect de l'autonomie des centres d'activités
    - + temps de réalisation
    - + problème de coûts
    - + décision arbitraire.

Le groupe de concepteurs fera des propositions dans ce sens aux instances responsables.

- les règles de gestion
  - . les règles actuelles seront-elles modifiées ?
  - . description sommaire des principes de fonctionnement du système de gestion et le déroulement des activités

  - décrire la façon dont les besoins en information des différents utilisateurs seront satisfaits et le contenu de ces informations
  - fixer les mesures à prendre en fonction des changements que provoquent l'implantation du nouveau système sur la structure de l'entreprise

- les investissements
  - les investissements nécessaires
  - problèmes de financement
  - personnel supplémentaire à mettre à disposition
  - formation exigée du personnel

- la relation "cost/benefit"
  - déterminer les coûts et les gains de fonctionnement et d'investissement prévus
  - déterminer les avantages et inconvénients non quantifiables que doit procurer le système

- les moyens techniques
  - pour la saisie des données
  - mode de traitement des données: temps réel, par lot ou mixte
  - organisation des données conventionnelle ou utilisation d'un SGBD

Le choix du SGBD, processus décrit au chapitre 9.1, peut se poser. Arrivé à ce stade des études de développement, plusieurs alternatives se présentent selon que l'on décide de réaliser une banque de données intégrée ou une banque de données par sous-système :

a) Banque de données intégrée

Dans ce cas, il est nécessaire de déterminer les principales caractéristiques des différents sous-systèmes et de déclencher le processus d'évaluation de SGBD en effectuant de façon globale les 3 premières phases de la méthode d'approche pour la conception d'une banque de données, afin de déterminer les caractéristiques des applications à développer.

b) Banque de données par sous-système

. Le choix peut se faire en fonction des exigences du sous-système le plus important du point de vue exigences, volumes des données, nombre d'utilisateurs. Le choix s'effectuera après la troisième phase de la méthode d'approche pour la conception d'une BD.

. L'autre possibilité est identique à celle décrite en a).

REFERENCES

[5.1] GAGSH S., "Eine Methode zur computer-gestützten Vorbereitung der graphischen Darstellung ..." p. 91 ss.

[5.2] L'article de GERBER F. est très représentatif dans cette optique, "Das Soll-Konzept. Die Qual der Wahl" p. 56 ss. qui propose d'étudier toutes les possibilités qui peuvent entrer en ligne de compte.

KIRSCH W., dans "Probleme der Unternehmungsführung ..." p. 173 ss., émet des idées semblables à la méthode que nous proposons.

[5.3] HABERFELLNER R., "System Engineering" p. 373 ss.

[5.4] GAGSH S., "Probleme der Partition und Subsystembildung" p. 623 ss.

[5.5] DELOBEL C., "Contributions théoriques à la conception et à l'évaluation d'un système d'informations appliqué à la gestion" p. 6.3 ss.

CHAPITRE 6

ANALYSE SEMANTIQUE DU MONDE REEL

Afin de pouvoir décrire le monde réel du système entreprise ou du sous-système considéré, il est nécessaire d'en étudier les interprétations des utilisateurs pour aboutir, à l'aide d'un processus de formalisation, à une représentation précise du monde réel (figure 6.1). Le modèle doit pouvoir satisfaire aux besoins des différents utilisateurs. Par conséquent, il importe d'étudier les vues partielles afin d'obtenir une représentation cohérente, non ambigue et sémantiquement non redondante du monde réel considéré.

6.1. RECENCEMENT DES BESOINS EN INFORMATION

Une fois effectué le choix du sous-système à automatiser, il faut entreprendre un recencement systématique des besoins en information, dans le but d'établir un inventaire aussi complet que possible des données indispensables au bon fonctionnement du système.

Les informations recueillies lors de la phase de modélisation du système de gestion serviront de base de départ. Elles seront étudiées plus en détail, d'où nécessité de recherches supplémentaires auprès des utilisateurs, afin de cerner au mieux possible les vues partielles.

Le but de cette activité consistera à déterminer:

- les informations dont la structure sera ensuite représentée sous forme de modèle
- préciser les processus de traitement de l'information, analyse nécessaire à la réalisation des programmes d'application.

L'analyste sera confronté aux problèmes suivants:

- problèmes de nature psychologique
  . disposition de l'utilisateur face au

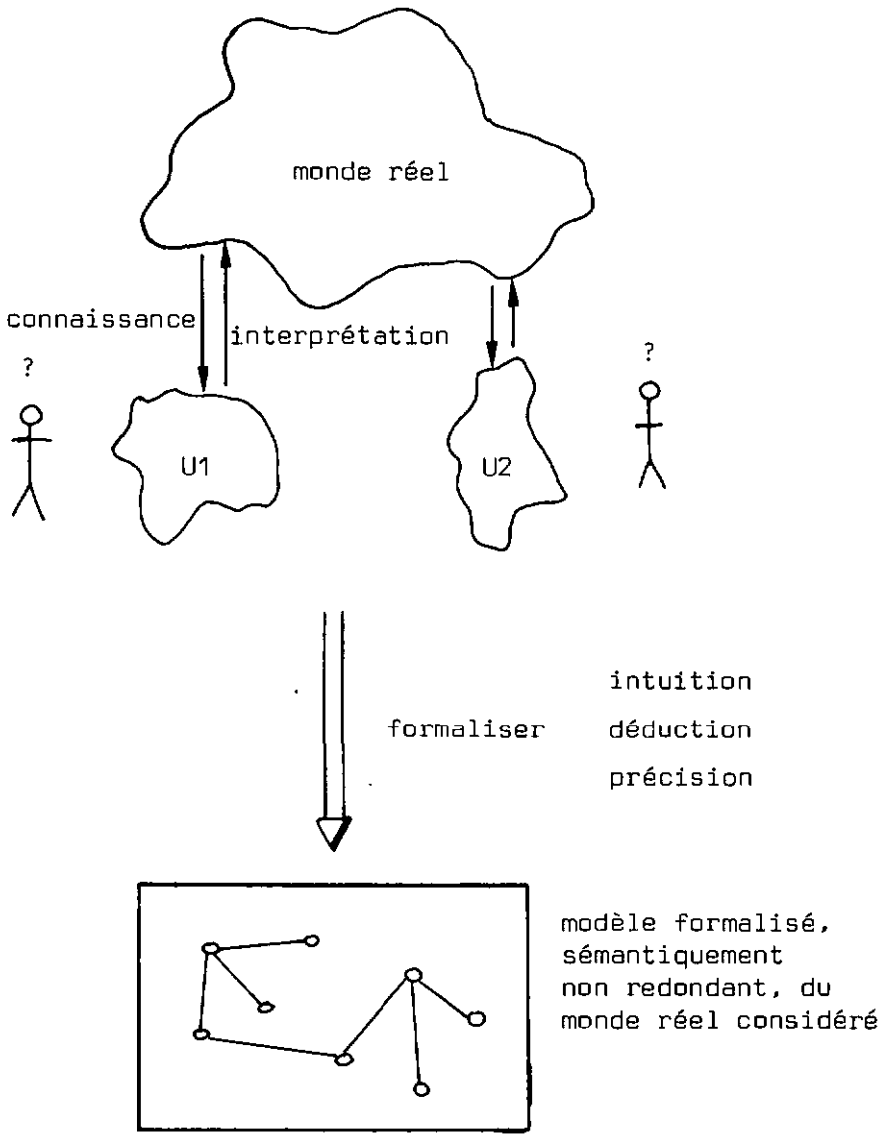


Fig. 6.1 - Représentation du monde réel

projet, méfiance ou même opposition passive
  . comportement de l'analyste vis-à-vis des utilisateurs
  . craignant les critiques, l'utilisateur aura tendance à défendre les faiblesses du système actuel
  . l'utilisateur donnera souvent une importance exagérée à certains problèmes particuliers

- différencier la nature des besoins en information
  . besoins objectifs qui dépendent d'une activité donnée et qui sont donc indépendants de la personne chargée de l'exécution
  . besoins subjectifs qui dépendent de la personne exécutant une tâche donnée

- les besoins futurs.

En général, l'utilisateur n'est pas capable d'énumérer exhaustivement ses besoins. Il aura tendance à décrire les activités dont il est responsable, les processus de traitement qui en découlent et les informations manipulées. Seul un dialogue intensif entre analyste et utilisateur permettra de définir les besoins réels en information. Cette analyse des besoins implique l'examen des points suivants:

- la logique des procédures de traitement
- le volume des données
- la fréquence des besoins
- les contraintes de performance
- la provenance et la destination des informations
- l'utilisation des informations
- les contraintes de sécurité

Différentes méthodes existent en vue de fixer les besoins en information (6.1). On peut les grouper en cinq catégories:

1. enquête
   - interview
   - questionnaire écrit
   - mixte
2. observation
   - structurée, analyse des circuits de documents, analyse quantative (volume, fréquence)
   - non-structurée
3. auto-analyse
4. observation indirecte
5. déduction et adaptation à partir d'un modèle (6.2).

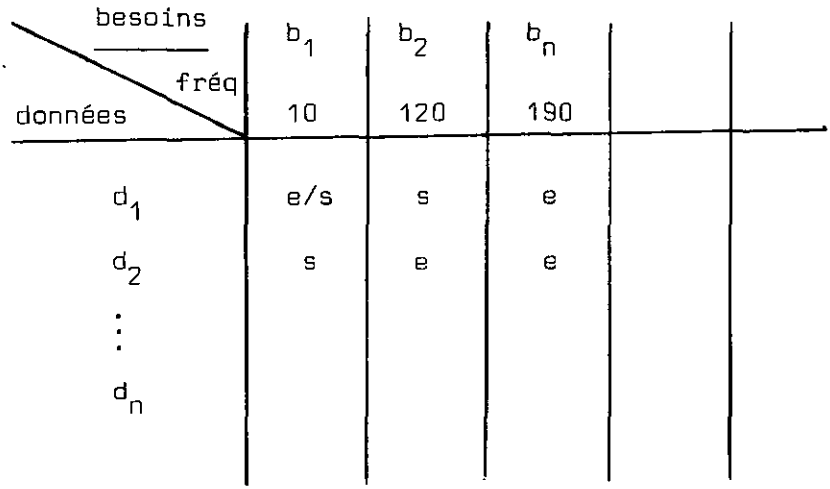
Seule une analyse approfondie à l'aide des critères cités ci-dessus permettra de cerner le problème de façon satisfaisante. Il n'existe aucune technique garantissant une qualité optimale des résultats, celle-ci dépendant uniquement de l'efficacité de l'analyste.

L'objectif de tout système de traitement de l'information consiste à satisfaire les besoins en information. A chaque demande est associé un ensemble ordonné de données qui peuvent apparaître dans plusieurs besoins. Pour des raisons de cohérence et de maintenance, le système n'enregistrera les données et les relations entre elles qu'un nombre limité de fois. L'organisation de la banque de données doit permettre au système de fournir les messages exigés selon des procédures de traitement optimales et conformément aux contraintes fixées.

D'autre part, la relation coût/valeur doit être étudiée afin de décider si les faits réels considérés seront représentés au niveau du modèle. On constate que la grande partie des informations utilisées par les gestionnaires, en premier lieu au niveau stratégique, sont

- non structurées
- de nature non répétitives
- très peu utilisées
- non soumises à des traitements de masse.

Une bonne méthode de structuration et de recencement des données consiste à représenter, à l'aide d'une matrice, les données utilisées par chaque besoin (fig. 6.2). Il importe de distinguer les données élémentaires des données composées obtenues par calcul.



Légende : e = entrée, s = sortie

Fig. 6.2 - Matrice des besoins en information

6.2. ANALYSE SEMANTIQUE DES DONNEES

6.2.1. Objectifs

L'objectif de cette analyse consiste à élaborer la structure des données à partir de l'inventaire effectué à l'étape précédente. La technique d'analyse utilisée devra satisfaire aux conditions suivantes:

- ête indépedante de toutes contraintes d'implantation
- ne pas favoriser certains besoins en information, c'est-à-dire être neutre face aux procédures de gestion
- être un outil d'analyse critique qui incite l'analyste à étudier en détail la structure et la signification des concepts représentés
- permettre une représentation simple de la structure des données et faire ressortir le type de relation
- faciliter la communication entre l'utilisateur, le concepteur et le spécialiste chargé d'implanter la banque de données
- être dynamique, c'est-à-dire offrir des possibilités d'ajouter de nouvelles données ou relations sans remettre en cause tout le modèle
- approche "top down", c'est-à-dire du général au particulier, de telle sorte qu'à un stade donné on ne remette plus en question les stades précédant
- garantir un modèle cohérent et sémantiquement non redondant
- favoriser une définition précise et non ambigue des concepts représentés

6.2.2. Comparaisons des modèles de données existant

La matrice établie à l'étape précédente ne satisfait en aucun cas aux conditions formulées. Elle permet au plus d'étudier la fréquence d'utilisation de chaque données et les besoins redondants.

Nous examinerons d'abord les principaux modèles de données (6.3) en relation avec les conditions formulées au chapitre précédent. Nous nous limiterons aux modèles Codasyl, relationnel et DIAM (relation binaire) de Senko, pour conclure avec le modèle Data Semantics d'Abrial. Il faut signaler qu'il existe d'autres modèles (6.4).

a) Codasyl

Les langages de description de SGBD, tels ceux basés sur le DBTG du Codasyl (IDMS, IDS, DMS 1100, etc) ou ceux qui en sont dérivés, tel IMS, sont des langages dépendant directement de critères d'implantation physique. IMS, système hiérarchique, ne permet de définir que des relations de type 1:n et un élément donné, le segment, ne peut posséder qu'un précédent. Pour remédier à cet inconvénient, la notion de "pairing" a été introduite, grâce à laquelle il est possible de lier des hiérarchies entre elles. Le schéma ainsi décrit implique le choix des méthodes d'organisation, des formats des données, le groupement logique des données en segments et la délimitation de zones physiques de données. L'utilisateur, quant à lui, ne peut traiter que des hiérarchies. D'autre part, les relations de type n:m ne peuvent être décrites qu'en introduisant un nouveau segment. Les mêmes problèmes se posent avec les systèmes de type Codasyl, à la différence que la vue de l'utilisateur peut se définir à l'aidede réseaux. Dans les deux cas, ce sont les accès exigés par les différentes applications qui déterminent la structure. Des modi-

fications de structure peuvent remettre en cause tout le schéma. Nous proposons de prouver cette affirmation à l'aide d'un exemple (figure 6.3).

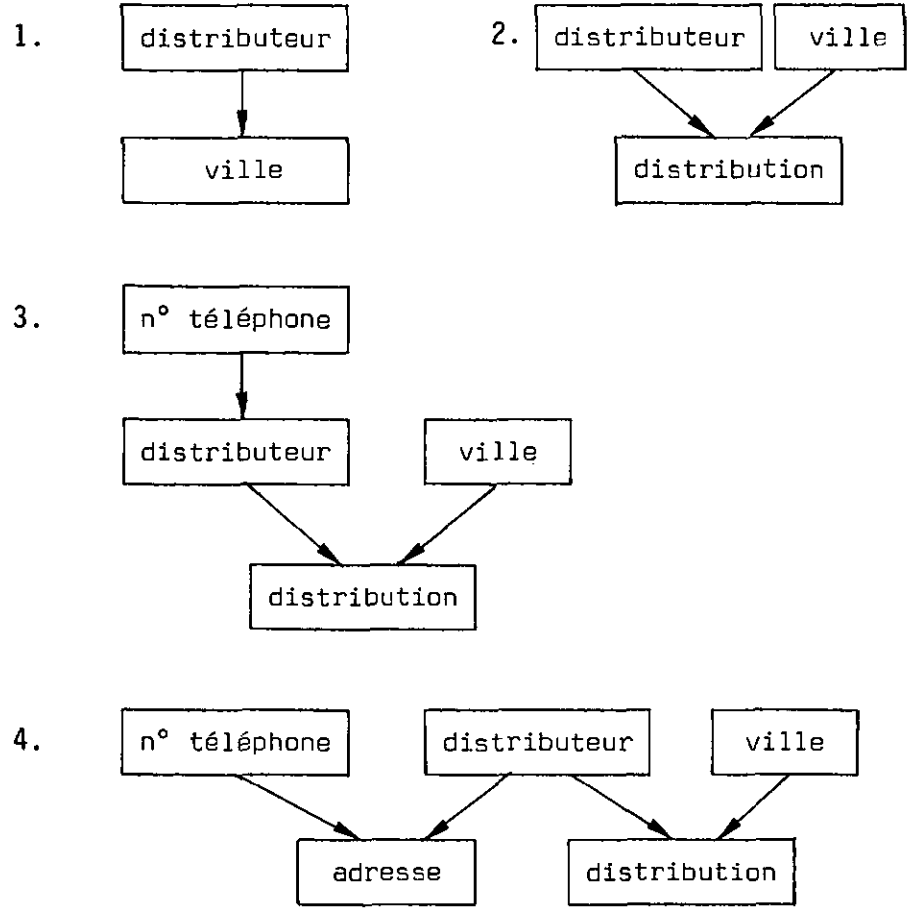


Figure 6.3 - Flexibilité du modèle Codasyl

a) La représentation du fait qu'un distributeur peut travailler dans plusieurs villes et inversement nécessite (passage de 1 à 2) :

- la création d'un nouveau "record"
- la création de nouveaux sets.

b) Un nouveau besoin d'information, accéder au distribueur, implique (passage de 2 à 3) :

- l'extraction de constituants (n° tél.)
- la création d'un "record"
- la création d'un nouveau set.

c) Vouloir représenter le fait que le distributeur peut avoir plus d'une adresse a pour conséquence (passage de 3 à 4) :

- l'extraction de constituants
- la création d'un nouveau "record"
- la suppression et la création de nouveaux sets.

L'exemple ci-dessus démontre que le modèle de données Codasyl ne satisfait en aucun cas aux conditions de stabilité et de flexibilité requises pour la représentation du monde réel, la structure dépendant en premier lieu de critères d'accès et de performance.

D'autre part, il existe plusieurs façons de représenter le monde réel, les contraintes d'accès déterminent la choix à faire.

L'utilisation du langage de description de données exige des connaissances approfondies des caractéristiques techniques du système considéré (voir référence 6.5).

b) Modèle relationnel

Afin de remédier à ces inconvénients, en particulier l'influence de contraintes d'implantation au niveau de la structure logique, Codd a développé le modèle relationnel (6.6), qui se caractérise entre autre par sa simplicité, l'indépendance des données ,la facilité dans la manipulation des données (6.7) et une plus grande flexibilité face aux changements. Reprenons l'exemple précédent (figure 6.5) et représentons la structure des données selon le modèle relationnel:

1. Distributeur (# personnel, nom, adresse, # tél, nomville)

2. Distributeur (# personnel, nom, adresse, # tél)
   Ville (# ville, nomville)
   Distribution (# ville, # personnel)

3. Identique à 2

4. Distributeur (# personnel, nom)
   Adresse (# personnel, type d'adresse, adresse, tél)
   Ville (# ville, nomville)
   Distribution (# ville, # personnel)

De cet exemple, l'on peut en déduire

- l'indépendance de ce modèle face aux besoins d'accès, en effet un nouveau besoin (passage de 2 à 3) ne nécessite aucune modification de la description du monde réel
- une modification du monde réel (passage de 3 à 4) a eu pour conséquence l'introduction de nouvelles relations fonctionnelles et de nouveaux constituants

- les critères de performance n'entrent pas en ligne de compte
- la notion de "set", relation de dépendance entre records, n'existe pas
- une telle description n'exige aucune connaissance technique.

L'approche par le modèle relationnel peut se faire selon deux optiques fondamentalement différentes :

a) Les exigences d'accès, d'insertion, de modification et de suppression peuvent être formulées à l'aide de relations. La relation ci-dessous représente une telle requête (R_1). En premier lieu, les relations ainsi déterminées devront être normalisées, en ce sens qu'une relation ne devra être composée que d'attributs élémentaires. Ce processus doit s'effectuer sans perte d'information (R1'). Le concepteur devra ensuite se poser la question de savoir si l'ensemble des relations contient des redondances et est cohérent. Dans ce but, il devra déterminer la forme des relations et si nécessaire les transformer selon un processus de normalisation (6.8), en troisième forme normale (3FN). Une relation se trouve en première forme normale (1FN) si elle ne contient pas de groupes répétitifs, ce qui est le cas des relations R_1' a) et R_1' b). La relation R_1' contenant un groupe répétitif, elle n'est donc pas en 1FN. Une relation en 1FN est également en 2FN si chaque attribut de complément de l'index dépend de ce dernier, c'est le cas de la relation R_1' b). L'on s'aperçoit qu'une telle relation contient des attributs décrivant deux entités différentes, l'entité commande et l'entité article, ce qui pose des problèmes au niveau de la maintenance.

R_1 (#client, adresse, #commande, #ligne, #article, texte, qté)

R_1' (#client, nom, ville, #commande, #ligne, #article, texte, qté)

R'_1 a) (#client, nom, ville)

b) (#commande, #ligne, #client, #article, texte, qté)

R_A (#client, nom, #ville)

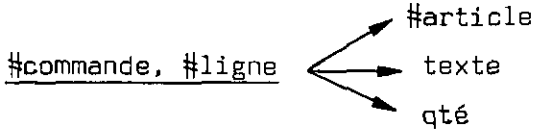
R_B (#ville, ville)

R_C (#commande, #ligne, #article, qté)

R_D (#article, texte)

R_E (#commande, #client)

Une relation est en 3FN si pour tout attribut relié fonctionnellement à un index la relation fonctionnelle est élémentaire directe. Dans notre exemple, la relation R'_1 b n'est pas en 3FN, car le "texte" ne dépend pas directement de l'index {#commande, #ligne}, mais directement de l'attribut "#article".



Il existe tout de même une relation fonctionnelle entre l'index et le complément, car pour une valeur donnée {#commande, #ligne}, il n'existe qu'une seule valeur du complément {#article, texte, qté} correspondant, mais la dépendance est transitive.

Les relations R_A, R_B, R_C, R_D et R_E sont en 3FN, car les relations entre les différents attributs sont des RFE directes.

b) La deuxième approche consiste à décrire le monde réel, non à l'aide de relations, mais en inventoriant les attributs et les relations fonctionnelles élémentaires qui les relient (6.9). Le concepteur doit décrire la réalité à l'aide de relations fonctionnelles élémentaires (RFE), si possible de manière exhaustive, une certaine redondance n'étant pas exclue.

Celle-ci se manifeste en ce sens qu'une RFE peut être engendrée à l'aide d'autres RFE. Une fois la liste de départ établie, il s'agira d'en déduire de nouvelles RFE, afin d'obtenir un ensemble de RFE aussi complet que possible, appelé la fermeture. Les redondances devront être éliminées en vue d'obtenir le noyau qui est l'ensemble des RFE qui permet d'engendrer la fermeture. On ne peut sans autre affirmer que les RFE du noyau sont en 3FN. Représentons les RFE de la fermeture à l'aide d'un graphe orienté qui ne contiendra aucun cycle si le noyau est en 3FN (6.10).

Les mêmes critiques peuvent être formulées envers l'approche relationnelle qui, au départ, se base sur les requêtes de l'utilisateur. Deux optiques différentes sont possibles. La première consiste à formuler les requêtes sous forme de relations qui seront ensuite transformées, selon un processus de normalisation, en 3FN ou de formuler les requêtes à l'aide de hiérarchie qui seront également ramenées à des relations de 3FN. La deuxième part de l'idée que le monde réel soit décrit à l'aide d'attributs et de relations fonctionnelles élémentaires qui les relient et d'en déduire un ensemble de relations en 3FN. Un attribut est toujours étudié en fonction de l'index d'une relation donnée et le fait que la signification d'un attribut peut être différente, selon le contexte dans lequel il est utilisé, n'apparaît pas au niveau du modèle. Dans l'exemple ci-dessous, l'ambiguïté est évidente.

#personne	nom	âge
101	Dupont	54
103	Durand	29

#voiture	marque	âge
596181	Renault	3
596213	VW	29

Si l'on analyse les relations entre tous les attributs, il sera facile de faire ressortir le type de relation qui les relie et de décrire les différentes significations rattachées à un certain attribut. La description du monde réel à l'aide de relations fonctionnelles se borne à étudier les relations entre un index et les attributs qui en dépendent et néglige l'analyse des relations entre attributs, qu'ils appartiennent ou non à un index. Le modèle est-il complet et non redondant ? Une réponse à cette question ne peut être donnée qu'après avoir étudié les relations fonctionnelles de la liste de départ afin d'en déduire des relations en 3FN.

D'autre part, l'on ne peut déduire le type de relation représenté par les relations fonctionnelles avant de les étudier en détail. En effet, la liste de départ peut contenir des relations fonctionnelles redondantes, qui peuvent être engendrées à partie d'autres relations fonctionnelles ou des relations fonctionnelles non en 3FN.

Cette approche qui consiste à établir au départ une liste de relations fonctionnelles et d'en déduire des relations en 3FN peut être qualifiée d'approche de bas en haut (buttom up), alors que la méthode proposée, impliquant une analyse sémantique pour aboutir ensuite à des relations en 3FN, est plutôt du type hiérarchique, c'est-à-dire "de haut en bas" (top down).

Un autre inconvénient du modèle relationnel provient du fait que la représentation des faits élémentaires à l'aide de relations en 3FN peut conduire à des incohérences. L'exemple ci-dessous prouve que l'on ne peut dans tous les cas garantir que toutes les manipulations de données n'aient aucune répercussion sur d'autres relations

R_1 : Ville (#ville, nomville)

R_2 : Distr (#ville, #personne)

R_1

	#ville	nomville
t_{11}	1000	Lausanne
t_{12}	1800	Vevey

R_2

	#ville	#personne
t_{21}	1000	C 128
t_{22}	1000	D 321
t_{23}	1800	C 143

La suppression du tuple t_{12} met la banque de données dans un état inconsistant.

Un autre problème est celui posé par l'équivalence de différents systèmes de relations qui nécessite le choix de l'ensemble permettant de satisfaire à toutes les contraintes du système. Toute manipulation doit maintenir un état consistant du modèle. Le choix correct nécessite une étude détaillée des relations fonctionnelles (6.11).

En résumé, le modèle relationnel présente les inconvénients suivants:

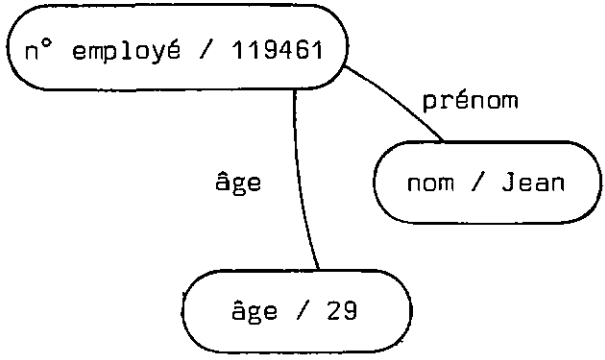
- il conduit dans certains cas à des incohérences
- il manque de précision dans la description, les constituants n'étant étudiés qu'en fonction d'un index
- il ne favorise pas l'analyse de la signification des constituants, puisqu'on étudie en premier lieu les relations fonctionnelles.

Mais par rapport au modèle Codasyl, on peut relever plusieurs avantages:

- indépendance envers toutes contraintes d'implantation physique
- simplicité du modèle, compréhensible par n'importe quel type d'utilisateurs
- les chemins d'accès ne sont pas décrits au niveau du modèle.

c) DIAM

Dans le modèle DIAM de Senko (6.12), les valeurs telles que 'Jean', '29' et '119461' sont considérées comme étant des ENTITY NAMES. L'ensemble des valeurs de même nature sont appelés les ENTITY NAME SETS 'nom', 'âge' et 'n° d'employé'. Une entité du monde réel est décrite à l'aide d'ENTITY NAME SETS et du rôle (ROLE NAME) que ces derniers jouent dans la description (DESCRIPTION SET NAME) de l'entité considérée. Chaque entité sera identifiée (IDENTIFIER) par un ou plusieurs ENTITY NAME SET. Dans l'exemple ci-dessous représentant l'entité 'employé', 'Jean' est un ENTITY NAME de l'ENTITY NAME SET 'nom' qui, dans la description de l'entité (DSN) endosse le rôle (RN) de 'prénom' (fig. 6.4).



Nom	Type de nom	Identifier	Name set Name
Employé	DSN	(n° empl.)	
Employé/prénom employé	RN		prénom
Employé/âge employé	RN		âge
Employé/n° employé	RN		n° employé

Fig. 6.4 - Modèle selon DIAM

La description du monde réel consiste donc à déterminer des entités, à les décrire en indiquant les ENTITY NAME SETS et les rôles qu'ils jouent dans la description de l'entité concernée. Le but de cette activité est l'obtention de la structure logique. Ce n'est donc pas à partir d'une analyse des liaisons possibles et du type de liaison entre les différents ENTITY NAME SETS inventoriés que la structure logique est déduite. Seules les vues partielles sont prises en considération. Ce manque de systématique dans l'analyse des faits réels peut conduire à certaines redondances et incohérences. Il n'est pas exclu qu'une association entre des ENTITY NAME SETS de deux entités différentes soit ignorée lors de l'analyse, le processus de gestion n'y faisant peut-être pas référence.

D'autre part, le type de relation (1:1, 1:n, n:m) qui existe entre deux ENTITY NAME SET n'est pas mis en évidence, quoique celui-ci détermine de manière prépondérante la structure logique du modèle. Le concepteur n'analysant pas explicitement les associations possibles entre les différents ENTITY NAME SETS et le type de relation, il ne distinguera le type de relation d'une association entre deux ENTITY NAME SETS donnés qui peut être de type 1:1 ou 1:n, ce qui posera des problèmes lors du passage à la représentation physique du modèle. Dans l'exemple ci-dessous (fig. 6.5), le type des relations entre les ENTITY NAME SETS n'est pas évident et il est fort possible que le 'complément' puisse exister 0, 1 ou 2 fois pour une adresse donnée. Il en déduira la relation R_1. Si seulement 10 % des adresses ont un complément, 1 % deux, le reste aucun et que l'emplacement mémoire nécessaire pour représenter un seul complément soit de 30 caractères, la relation R_1 ne permettra que l'enregistrement de 0 ou 1 complément, donc incohérence, et le taux d'occupation mémoire sera relativement défavorable. Dans un tel cas, la structure logique devrait correspondre aux relations R_1a et R_1b. Ce n'est qu'au niveau

physique que sera prise la décision de la forme de la réalisation physique des relations. Les critères performance, volume des données, emplacement mémoire nécessaire, possibilité du SGBD seront déterminants. Il est fort possible que la représentation de l'adresse soit réalisée selon la relation R_1.

R_1 (n° adresse, nom, rue, complément, ville)

R_1a (n° adresse, nom, rue, ville)

R_1b (n° adresse, complément)

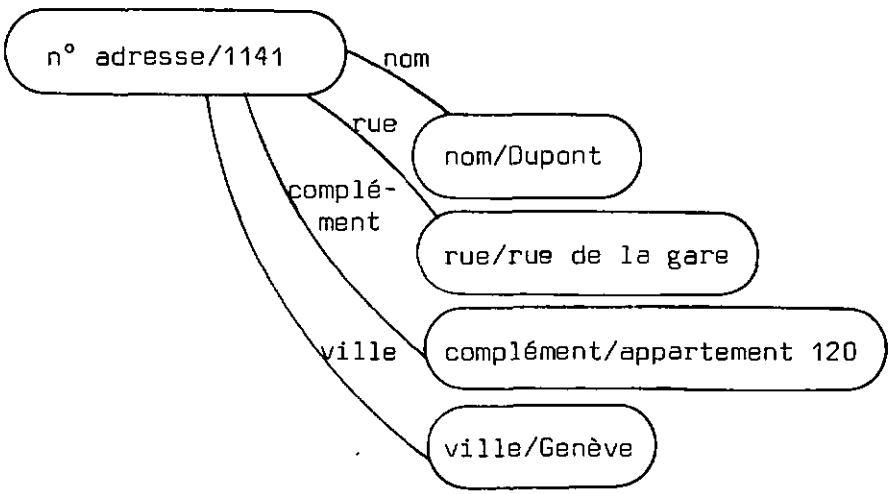


Fig. 6.5 - Entité 'adresse' représentée selon le modèle DIAM

En résumé, les inconvénients majeurs de l'approche DIAM sont les suivants:

- des redondances ne sont pas exclues, les entités étant déterminées sans étudier les relations qui

peuvent exister entre elles
- manque de systématique dans l'analyse des faits réels
- introduit une terminologie relativement complexe
- le modèle n'est pas assez précis, des informations supplémentaires sont nécessaires.

D'importants travaux sur la conception de banque de données ignorent l'importance et la nécessité d'effectuer une analyse sémantique des données. Un des premiers à aborder le sujet fut Lyon (6.13) qui insista sur la distinction entre la structure physique et logique, mais cette dernière ne résulte pas d'une analyse systématique: elle est déduite intuitivement. Lutz, quant à lui, définit la structuration des données comme étant "un processus de conception" destiné à grouper les données afin d'obtenir une banque de données en tout point économique et qui permette une exécution optimale des manipulations de données. Pour lui, ce problème ne peut être résolu qu'accidentellement (6.14). D'autres ne font qu'à peine allusion à ce problème, le processus de structuration se résumant en fait à assembler les données reliées logiquement en une unité dénommée segment (6.15).

Une autre approche consiste à déduire la structure logique des besoins en information quantifiés en fonction d'une unité de temps prédéterminée. Les relations entre les données dépendent en premier lieu d'aspects statistiques. La signification des données et des liaisons entre elles ne sont pas prises en considération (6.16). D'importants travaux dans cette direction ont été effectués au "BIFOA", institut dépendant de l'Université de Cologne (6.17).

6.3. METHODE D'ANALYSE SEMANTIQUE PROPOSEE

La méthode d'analyse sémantique proposée se base sur le modèle DATA SEMANTICS de J.R. Abrial (6.18). La description du monde réel s'effectue à l'aide d'ensembles de valeur et de connexions entre ces différents ensembles, ceux-ci reflétant d'une part les objets et d'autre part les faits élémentaires ainsi que les règles simples et complexes du monde réel considéré.

6.3.1. Les catégories

Chaque donnée représentée dans la matrice établie lors de l'inventaire des besoins en information peut être considérée comme étant une catégorie qui, selon la définition d'Abrial, correspond à un ensemble de valeurs de même nature auquel est associé un nom.

Exemple : '1.7.76', '15.8.76' sont des valeurs de même nature auxquelles on associe le nom 'date', 'Dupont' et 'Durand' font partie de la catégorie 'personne'.

Chaque catégorie sera représentée dans un schéma comme étant un noeud d'un graphe non-orienté. La notion de catégorie est semblable à celle de champ élémentaire.

Les données inventoriées doivent être analysées en fonction des ensembles de valeurs qu'elles représentent, des redondances n'étant pas exclues. Il arrive très souvent qu'un même ensemble de valeurs apparaisse plusieurs fois dans la matrice, selon la signification qu'il endosse dans un contexte donné. C'est le cas par exemple pour les données 'no d'abonné' et 'no de facture' qui référencent le même ensemble de valeurs 'no' qui peut prendre différentes significations, en fonction du contexte dans lequel il est utilisé.

Il est, d'autre part, très important d'étudier la signification des ensembles de valeurs qui, dans un même sens, peuvent être dénommés différemment. Cela peut se produire lorsqu'ils apparaissent dans des besoins en information d'utilisateurs différents. Ceci est d'autant plus probable que les sous-systèmes d'un système de gestion sont relativement peu intégrés et que pour chacun d'eux, il existe des concepts propres à chaque application. Un exemple tiré de la gestion des abonnements d'un journal illustre bien ce problème:

> la gestion du personnel est séparée entre celle du personnel de distribution et celle pour le reste des employés de l'entreprise. L'abonnement gratuit auquel a droit chacun d'entre eux est dénommé dans un cas comme étant un abonnement pour le personnel, alors qu'il est considéré dans le secteur de la distribution comme un exemplaire supplémentaire du journal ajouté lors de la production.

Lors de l'inventaire des besoins en information, deux données différentes pourront être relevées et seule une analyse de la signification des données permettra de déceler cette incohérence. Les gestionnaires devront s'entendre sur une désignation et représentation unique, activité qui ne doit en aucun cas être sous-estimée.

La représentation d'objets pose le problème de leur identification. En effet, la connaissance que l'on possède d'un objet, décrite par une ou plusieurs catégories, peut référencer plus d'un objet.

Exemple : le 'nom', la 'rue' et la 'ville' ne suffisent pas à identifier une personne. En effet aux valeurs 'Dupont', 'avenue de la gare' et 'Genève' peuvent correspondre plusieurs personnes.

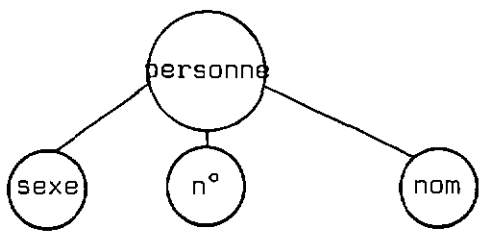
Une nouvelle catégorie sera introduite dans le but d'identifier un objet. Dans l'exemple ci-dessus, cela pourrait

être le 'matricule'. L'identification d'une entité pourrait être déléguée au SGBD, mais pour des raisons d'indépendance, il est nécessaire de résoudre ce problème au niveau de l'analyse sémantique, car tous les SGBD ne sont pas capables d'associer automatiquement un identificateur à chaque entité enregistrée (6.19).

Afin de pouvoir représenter la vue partielle des objets du monde réel, il est nécessaire de distinguer les catégories élémentaires des catégories composées, concept auquel est associé un ensemble de catégories elles-mêmes élémentaires ou complexes. Par exemple, l'objet personne peut être représenté à l'aide de la catégorie 'personne' en tant qu'ensemble des noms de personne

{Dupont, Durand, Dumont}

ou en tant que catégorie composée



{m,f} {4711,9565..}{Dupont,Durand, Dumont ...}

Fig. 6.6 -.Catégorie composée

La matrice de départ doit donc être analysée afin de

- déterminer les ensembles de valeurs,
- différencier les catégories selon la vue partielle que l'on veut décrire.

Cette analyse favorisera la détection d'oublis de certaines catégories et d'imprécisions. Par exemple :

> l'objet 'adresse' posera le problème de sa représentation au niveau du modèle. Le 'nom' et la 'ville' suffiront-ils ou devrons-nous introduire de nouvelles catégories 'rue' et 'complément' ?

Suivant leurs préoccupations, les utilisateurs peuvent percevoir de manière fort différente le même objet. Dans un tel cas, ils devront décider de la vue partielle de cet objet.

6.3.2. Les relations entre catégories

a) les relations binaires

Une association entre deux catégories indique la signification de l'une par rapport à l'autre, elle est de type binaire et inverse (figure 6.7).

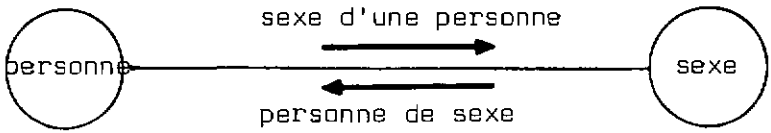


Fig. 6.7 - Relation entre catégories

Entre deux catégories, il peut exister, en tant que vue partielle du monde réel, aucune, une ou plusieurs associations (figure 6.8).

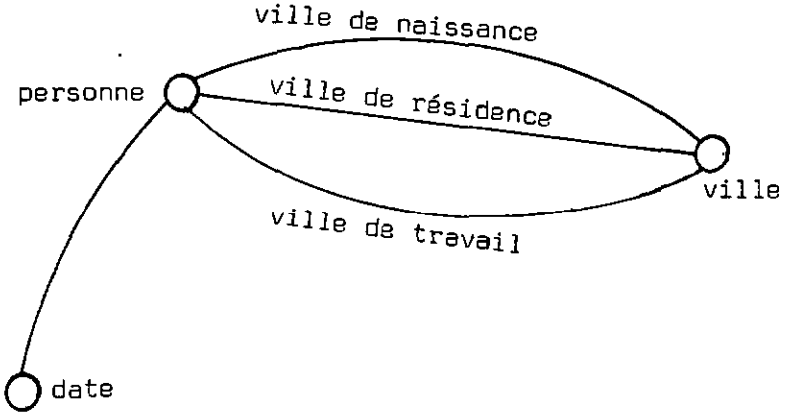


Fig. 6.8 - Plusieurs relations entre catégories

Le graphe de la figure 6.9 n'est pas exact, car les différentes catégories 'ville' représentent en réalité un seul ensemble de valeur : 'noms de ville'. Donc, il n'existe qu'une catégorie 'ville' reliée par trois fonctions d'accès à la catégorie 'personne' (figure 6.8).

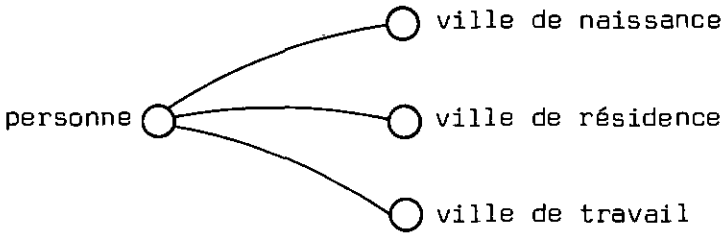


Fig. 6.9 - Catégories représentant un seul ensemble de valeurs

- Comment déterminer les relations ?

Une fois les catégories fixées, il existe deux méthodes d'analyse distinctes qui permettent la définition des relations:

1. étudier toutes les relations imaginables entre catégories, tout en comparant leur signification en fonction du monde réel que l'on désire représenter. Les associations énumérées à l'exemple précédent pourraient correspondre à la vue partielle d'un système de gestion du personnel, alors que seule la relation 'ville de résidence' seraient d'intérêt dans un système de gestion d'abonnements.

2. traduire les données de la matrice en termes de relations entre catégories, sans soumettre les résultats à une analyse plus approfondie, en se basant donc uniquement sur la matrice de départ.

Cette dernière méthode a un inconvénient, elle se base exclusivement sur les données de la matrice, alors que celle-ci ne reflète en aucun cas toutes les liaisons possibles entre les données et qui, en plus, n'est pas forcément complète et peut même contenir des incohérences. Par contre, la première méthode propose une analyse du monde réel considéré, ce qui permet de détecter certaines imprécisions et incohérences qui sont inévitablement contenues dans la matrice de départ. L'exemple ci-dessous, bien que très simple, illustre la difficulté.

besoins / données	Liste de distributeur	Distributeur d'une ville	Liste des rues d'une ville
distributeur	x	x	
rue	x		x
ville	x	x	x

De la matrice, on peut déduire trois catégories: 'distributeur', 'rue' et 'ville' et selon la deuxième méthode citée ci-dessus les relations :

rue d'une ville (1)
ville du distributeur (2)
rues du distributeur (3)

qui peuvent être représentées sous forme de graphe (figure 6.10).

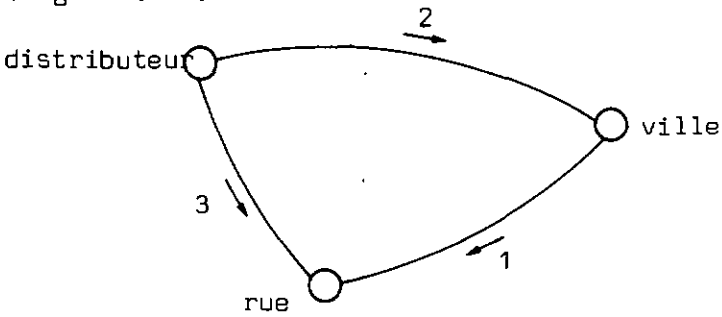


Fig. 6.10 - Modèle incomplet

Mais en réalité, seule une analyse approfondie de la signification des relations entre catégories, en fonction du monde réel que le concepteur a l'intention de décrire, permettra d'éliminer les incohérences, voire les redondances, qui peuvent s'infiltrer au niveau du modèle. Dans l'exemple ci-dessus, la matrice ne représente en aucun

cas les faits réels et la structure des informations qui en découle. En effet, le fait qu'un distributeur distribue dans plusieurs rues de villes différentes et que plus d'un distributeur puisse être affecté à une rue donnée, ne ressort à priori de la matrice. Seule une analyse sémantique, conformément à la première méthode proposée, aboutira à un modèle cohérent, ce qui, dans notre exemple, implique la création d'une nouvelle catégorie : 'secteur de distribution' (voir figures 6.11 a, b et c).

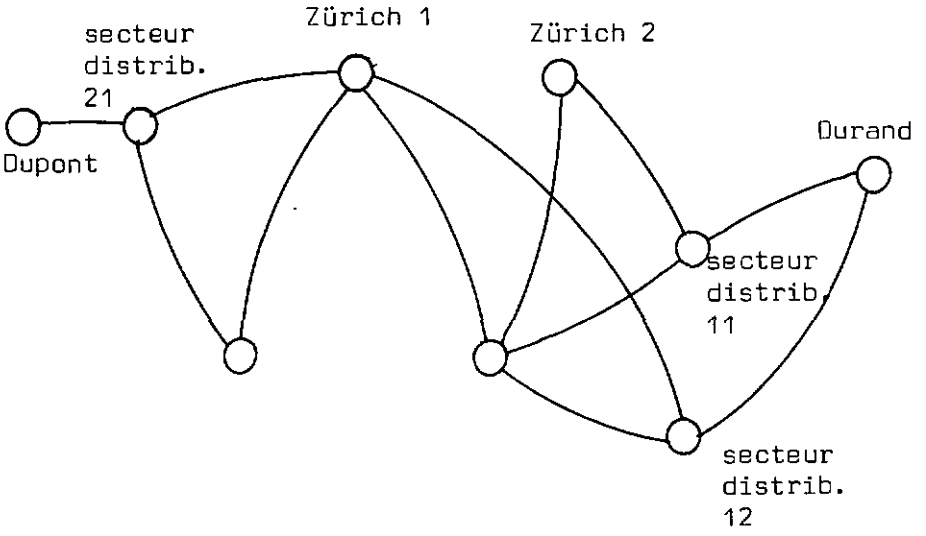


Fig. 6.11 a - Graphe des réalisations

Les relations binaires suivantes permettent, en associa-

tion avec les catégories déterminées, de définir le monde réel :

rues d'une ville (a)
secteurs de distribution du distributeur (b)
ville du secteur (c)
rue du secteur (d)

b	
Dupont	secteur 1
Durand	secteur 11
Durand	secteur 12

b'	
secteur 1	Dupont
secteur 11	Durand
secteur 12	Durand

c	
secteur 1	Zürich 1
secteur 11	Zürich 2
secteur 12	Zürich 2

c'	
Zürich 1	secteur 1
Zürich 2	secteur 11
Zürich 2	secteur 12

d	
secteur 1	albi
secteur 11	gare
secteur 12	gare

d'	
albi	secteur 1
gare	secteur 11
gare	secteur 12

a	
Zürich 1	albi
Zürich 1	gare
Zürich 2	gare

a'	
albi	Zürich 1
gare	Zürich 1
gare	Zürich 1

Fig. 6.11 b - Tables binaires des réalisations

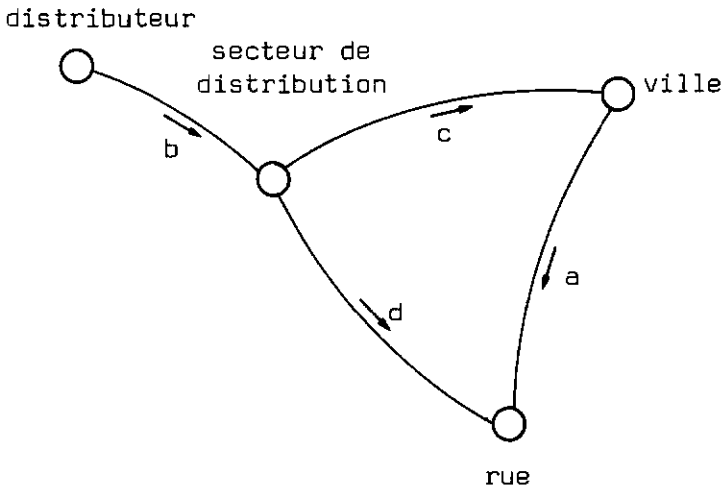


Fig. 6.11 c - Graphe logique

b) *Les relations n-aires*

Des relations de type n-aire ne peuvent être décrites à l'aide de relations binaires qu'en introduisant une nouvelle catégorie. L'exemple classique d'une association n-aire est le problème de la nomenclature dans un système de gestion de la production (6.20).

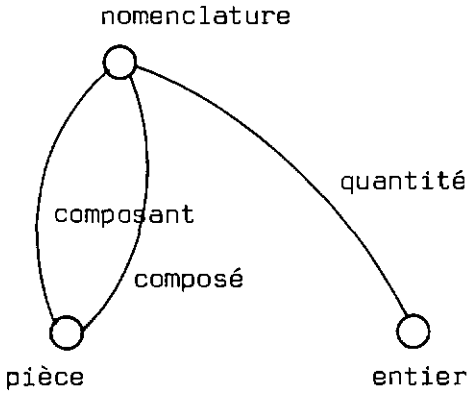


Fig. 6.12 - Représentation de relations n-aires

relation :

composant	composé	quantité
auto	chassis	1
auto	carrosserie	1
chassis	roue	4
roue	jante	1
roue	pneu	1

R_1 : (composant, composé, quantité)

Le modèle ne contiendra donc que des associations d'un seul type, des relations binaires, ce qui n'est pas un inconvénient, au contraire

- le passage à tout autre modèle en sera facilité (voir chapitre 7.3), le degré d'indépendance

plus élevé

- il en résulte une plus grande rigueur mathématiqu car les significations différentes affectées à un même ensemble de valeurs et ces dernières apparaissent au niveau du modèle, alors que dans le modèle relationnel un même ensemble de valeurs pe apparaître plusieurs fois sous des noms différent Dans la relation R_1, il n'est pas évident que les constituants 'composant' et 'composé' se réfèrent au même ensemble de valeurs 'pièce'.

c) *problème de la redondance*

Il est fort possible que des relations redondantes s'infiltrent au cours de l'analyse. Une attention particulièr doit être vouée à ce problème en étudiant attentivement la signification de l'enchaînement des relations et si celle-ci est conforme aux faits réels que l'on veut représenter dans le modèle.

Le graphe de la figure 6.13 a peut représenter deux faits réels différents selon la signification des fonctions d'accès :

- si c correspond au montant d'une facture d'un abonné, conformément au graphe des réalisations de la figure 6.13 b, alors le graphe logique est redondant
- si c représente le solde dû d'un abonné, ligne pointillée dans le graphe de réalisation, il n'y a pas redondance.

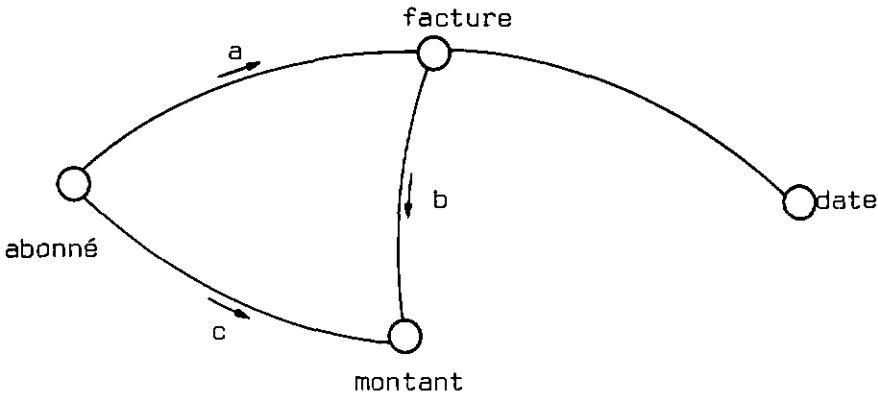


Fig. 6.13 a - Graphe logique partiel d'un abonné

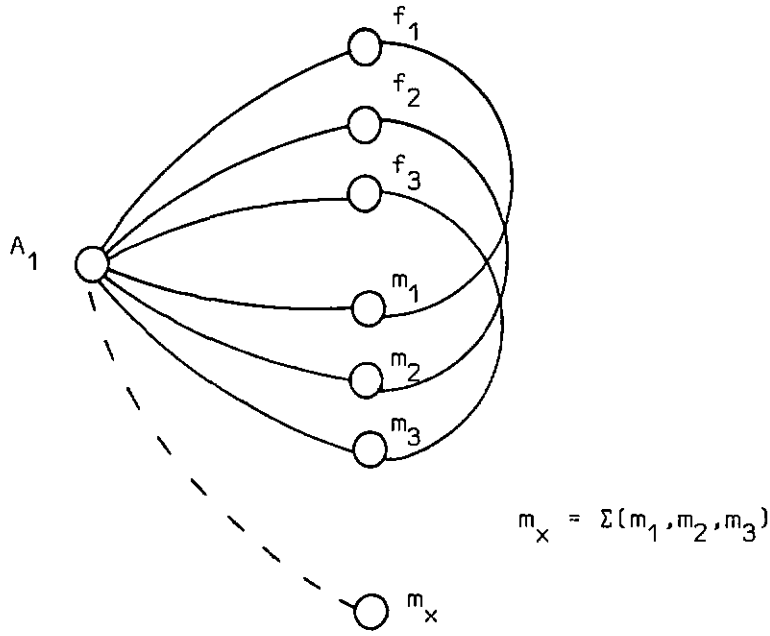


Fig. 6.13 b - Graphe des réalisations

Par contre, le cas illustré par la figure 6.14 est, malgré son apparence, non redondant. En effet, les faits réels indiquent qu'une police d'assurance dépend toujours d'un abonnement mais que les personnes assurées ne sont pas forcément abonnées et inversement. Le graphe est donc concret, a, b ≠ c' (c' étant l'inverse de c).

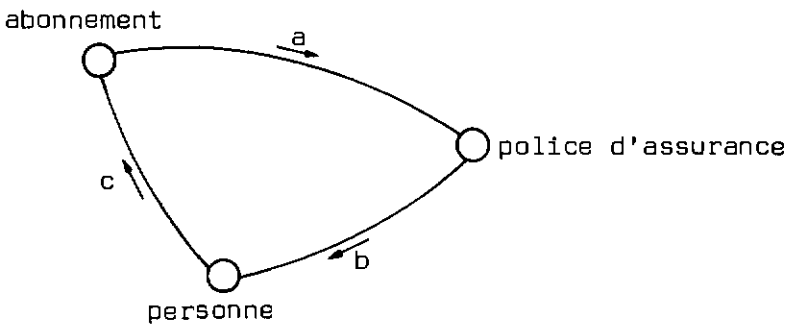


Fig. 6.14 - Graphe logique partiel

d) *L'aspect temporel*

Il arrive très souvent qu'un fait réel représenté au niveau du modèle puisse être de valeur différente dans le temps. Par exemple, un abonnement est formé de la catégorie 'conditions' et 'facture'. Les conditions d'un abonnement peuvent changer dans le temps, de même le concept 'facture' devra être placé dans l'espace en indiquant la date, ou un numéro d'ordre. Pour représenter l'aspect temporel au niveau du modèle, nous introduirons une catégorie choisie en fonction du fait réel. L'exemple cité ci-dessus correspondra alors, sous forme de graphe, à la figure 6.15.

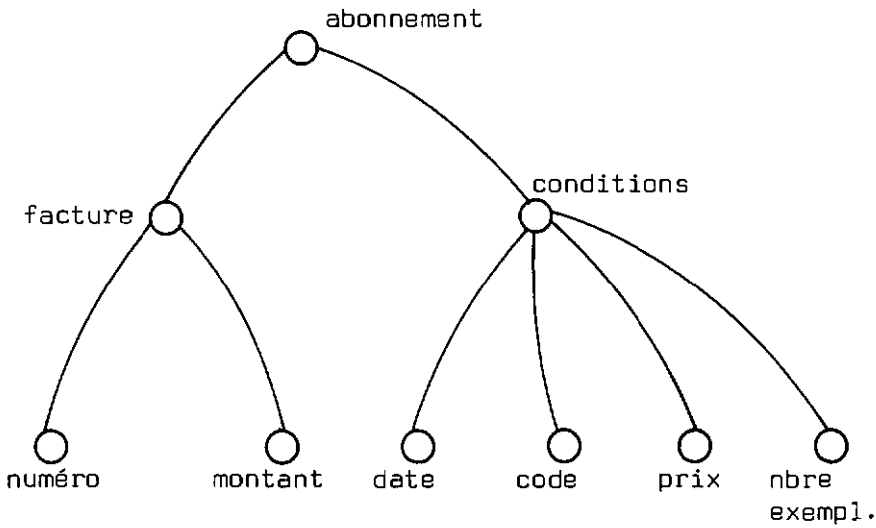


Fig. 6.15 - Graphe logique partiel d'un abonnement

6.3.3. Etablir un graphe

Le nombre des données inventoriées est généralement très élevé. Il est alors très difficile d'élaborer un schéma lisible qui puisse guider la conception et faciliter la communication. En parcourant la matrice des besoins en information, il est relativement facile de détecter certains groupements logiques de données, par exemple l'ensemble 'adresse', 'facture' ou 'police d'assurance'. Ceux-ci seront représentés comme étant les noeuds du graphe, l'analyse des relations entre catégories s'effectuera selon le procédé des "essais-erreurs" jusqu'à l'obtention d'un degré de cohérence maximum et seront représentés par des arcs non orientés. Au cours de cette procédure de nouvelles catégories devront généralement être introduites pour représenter des relations n-aires ou parce qu'elles ont été ignorées. Ensuite, pour chaque noeud ainsi déterminé, analyser l'ensemble des catégories et des associations

entre catégories qui peuvent être du type 1 : n, n : m ou 1 : 1, car il est fort probable que certaines relations aient été oubliées au niveau global, le graphe serait dans ce cas incomplet. D'autre part, l'analyse détaillée des catégories facilitera la détection de redondances, de contradictions ou de simples oublis, ce qui augmentera la qualité du modèle.

Ce processus itératif et hiérarchique est représenté à l'aide d'un exemple extrait d'un cas pratique. En analysant la matrice des besoins en information (le catalogue) il est sans autre possible de détecter les catégories et les relations représentées à la figure 6.16 a et b. Le résultat d'une succession d'analyses critiques du modèle et des faits réels que le modèle doit décrire nous mène à la figure 6.16 c.

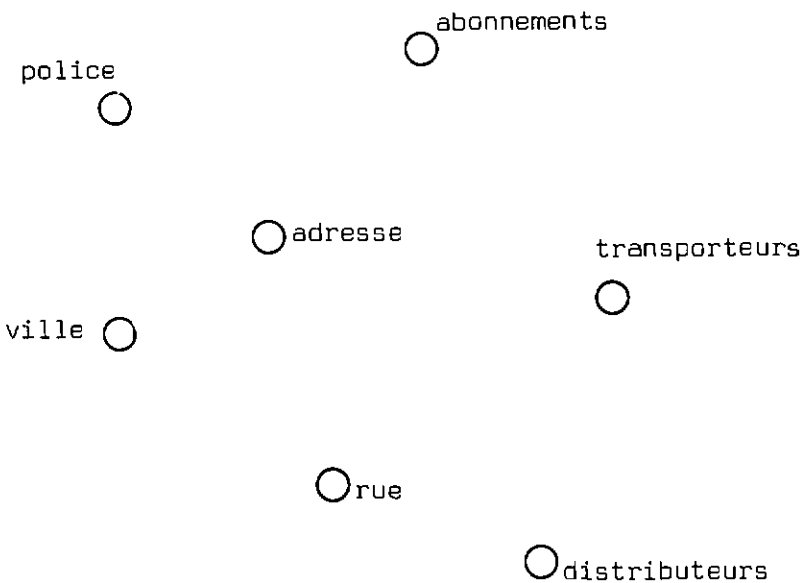


Fig. 6.16 a - Catégories principales

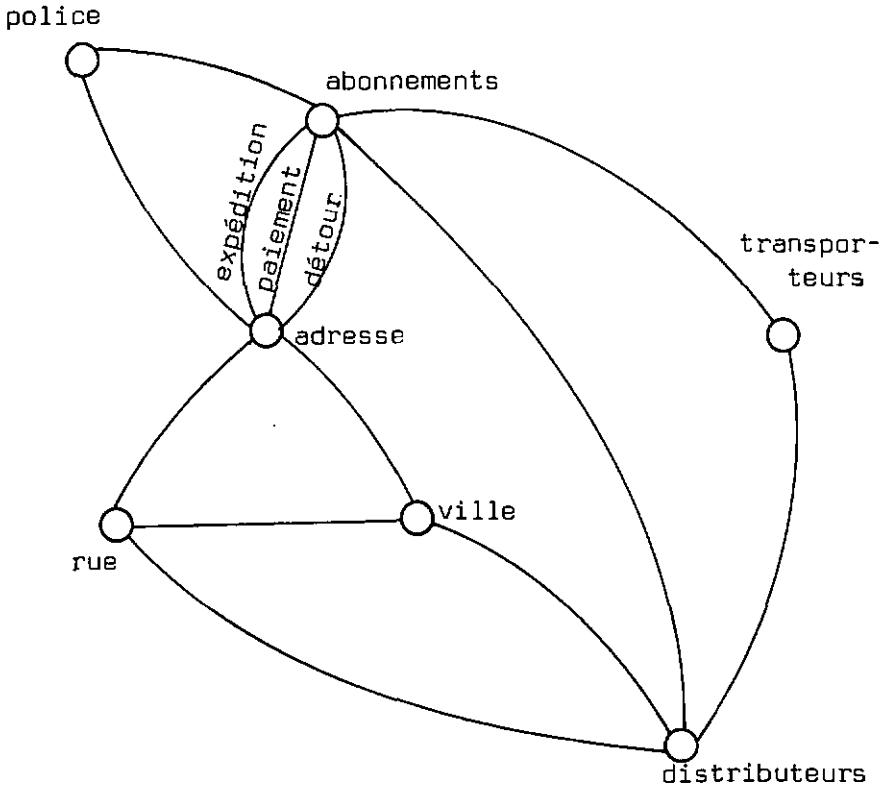


Fig. 6.16 b - Graphe logique après l'itération i_n

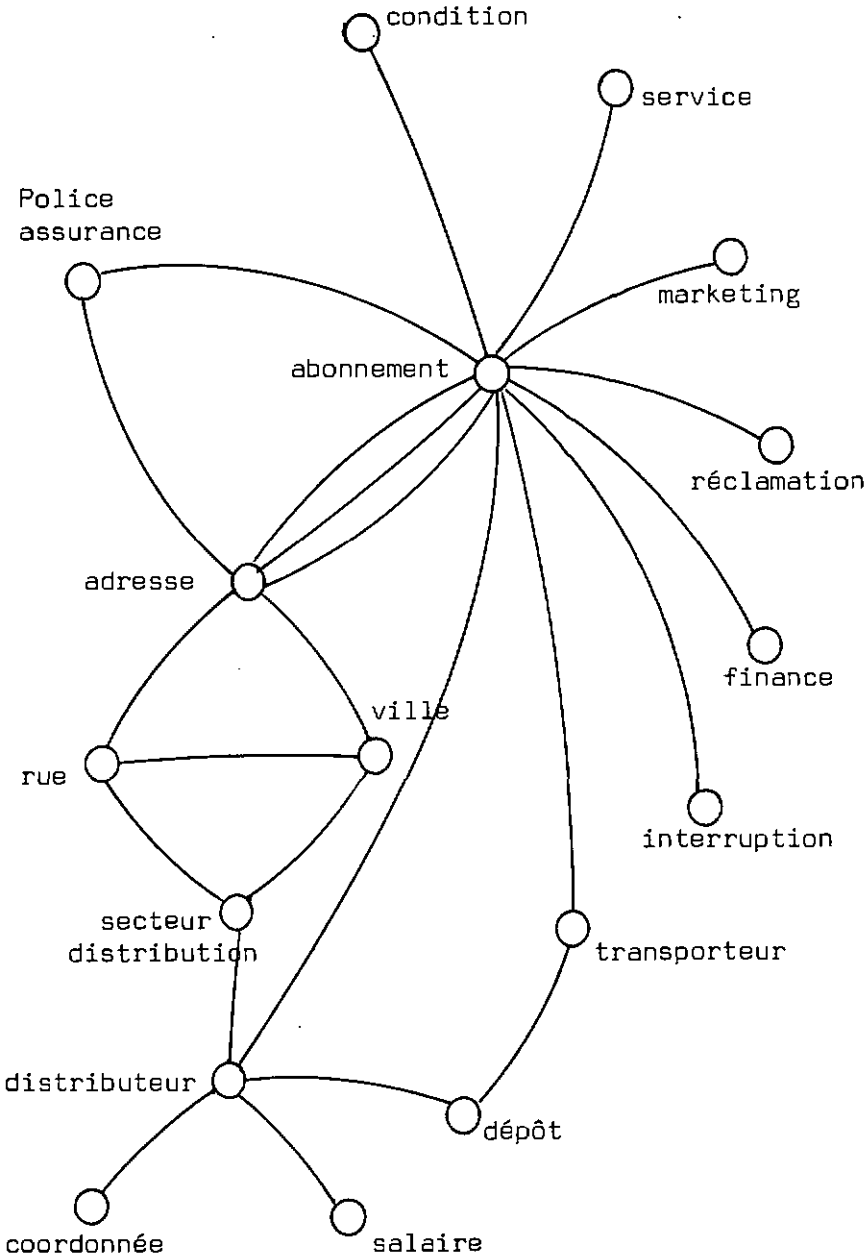


Fig. 6.16 c - Graphe logique après l'itération i_m, m > n

Reprenons, à titre d'illustration, l'exemple de la figure 6.16 c afin d'analyser en détail le noeud 'conditions' et aboutir au graphe de la figure 6.17.

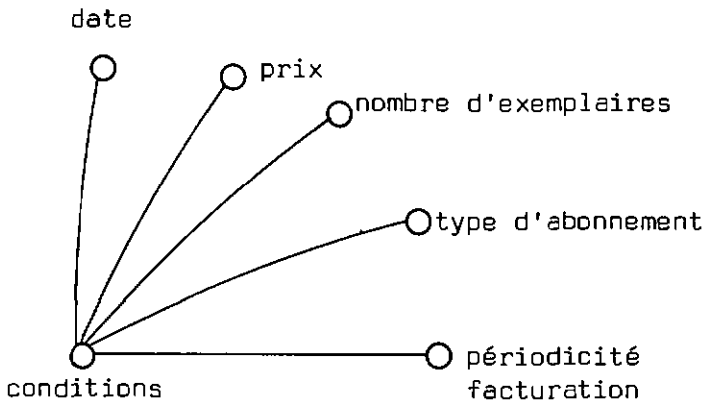


Fig. 6.17 - Graphe logique partiel d'un noeud du graphe de la figure 6.16 c

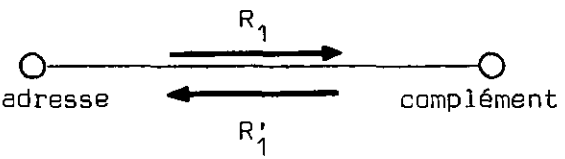
Il est fort possible qu'une catégorie représentée au niveau du modèle englobe des ensembles de valeur dont la représentation s'effectue selon des étalons de mesure différents. La catégorie 'n°' du graphe de la figure 6.18 est un exemple illustrant ce phénomène, le 'n° de personne' étant un ensemble de 6 chiffres, alors que 'n° de téléphone' en est un de 9 chiffres. Un regroupement de ce genre devra être spécifié au niveau de la description du graphe (chapitre 7, figure 7.1). Cette interprétation de la notion de catégorie peut être tolérée, car elle n'influence en aucun cas l'approche proposée.

6.3.4. Description du graphe

Chaque relation, représentée dans le schéma, sera numérotée, car il est impossible d'introduire la dénomination des fonctions d'accès dans le graphe, tout en garantissant une bonne lisibilité, lorsque le nombre des catégories et des relations est élevé. Le modèle sera ensuite décrit en indiquant, pour chaque relation entre catégories, un certain nombre de paramètres.

R_x = rel (noeud d'entrée, noeud de sortie, nom de la relation = afn (les paramètres d'existence minimum, maximum, moyenne), inv (paramètres d'existence))

Les paramètres d'existence indiquent le nombre de réalisations possibles d'une relation donnée. Par exemple, dans le cas de la relation 'adresse → complément', celle-ci peut ne pas exister ou au grand maximum trois fois pour une adresse donnée. Mais en moyenne, dans le cas du monde réel représenté, seuls environ 10 % des adresses ont un complément, la moyenne sera donc représentée par la valeur 0,1.



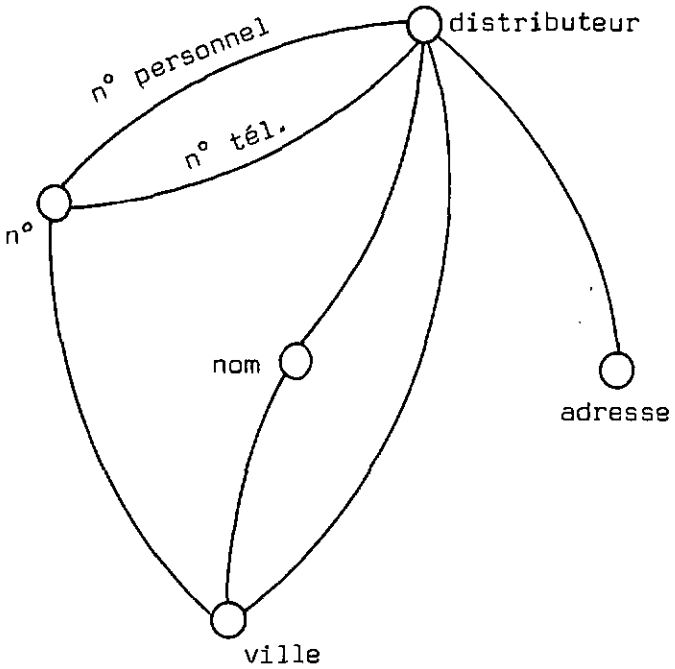
R_1 = rel (adresse, complément, complément-de-adresse = afn (1, 3, 0, 1) inv (1, 1, 1)

Les fonctions d'accès ainsi déterminées seront avantageusement représentées sous forme d'un tableau, à titre d'exemple consultez la figure 7.1 (chapitre 7).

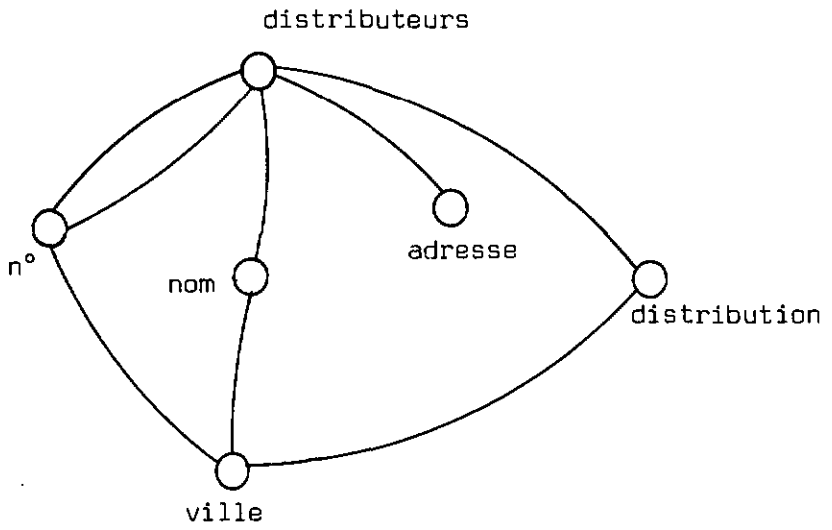
6.3.5. Flexibilité du modèle

Comment le modèle réagit-il face à des modifications dans la représentation du monde réel, entre autre la définition de nouveaux besoins d'accès ou la représentation de nouveaux faits réels. Illustrons le comportement du modèle à l'aide de l'exemple utilisé au chapitre 6.2.2, figure 6.3.

1.



2.



3. identique à 2

4.

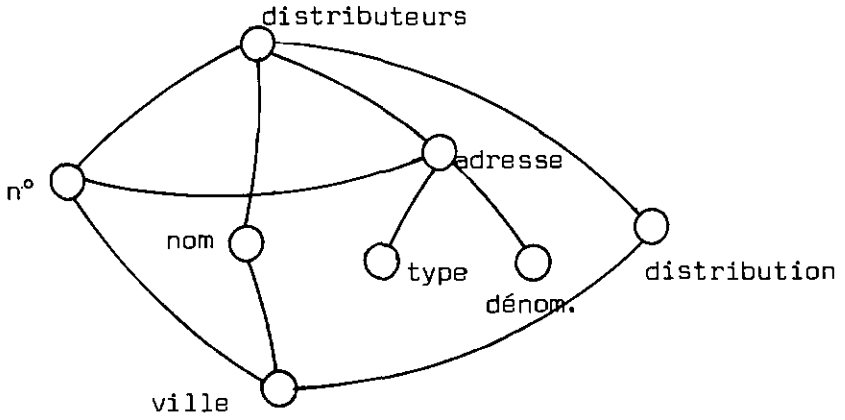


Fig. 6.18 - Flexibilité du modèle

Le modèle est très stable car une modification des besoins en accès ne modifie en aucun cas le graphe. D'autre part, la représentation de nouveaux faits réels ne remet pas en cause le schéma ; seules de nouvelles catégories et des modifications de relation sont nécessaires. Par exemple, le passage de 1 à 2 nécessite la modification de la relation 'distributeur → ville' qui devient ensuite 'distributeur → distribution', 'distribution → rue' et 'distribution → ville'. Cette stabilité est obtenue grâce au non-regroupement des catégories en record ou segment, ce qui ne nécessite pas la restructuration de ceux-ci lors de modifications.

En reprenant l'exemple de la figure 6.18, il est possible, à partir de la catégorie distributeur, de connaître son n° de personnel, son nom, son adresse et sa distribution. Dans le cas de la représentation sous forme relationnelle du même exemple (chapitre 6.2.2), les relations fonctionnelles nous permettent de connaître, à partir du n° de personnel, le nom du distributeur.

R_1 Distributeur (n° de personnel, nom)

R_2 Adresse (n° de personnel, type d'adresse, adresse, n° téléphone)

R_3 Ville (n° ville, nom)

R_4 Distribution (n° personnel, n° ville)

L'interprétation est donc limitée par la description du modèle lui-même. A l'opposé, le modèle proposé permet différentes représentations des vues partielles.

En résumé, l'on peut conclure que la méthode proposée dans ce chapitre satisfait aux exigences formulées au chapitre 6.2.1, en effet:

- elle est indépendante de toutes contraintes de performance ou d'accès ;
- elle est un outil d'analyse critique ;
- elle représente les faits réels de façon précise et non ambiguë ;
- elle propose un modèle stable ;
- elle élimine les redondances sémantiques ;
- elle facilite la communication avec les utilisateurs, utilisant des termes simples et non techniques ;
- elle permet une représentation simple et par niveau ;
- elle facilite le passage à d'autres modèles et procure ainsi un degré élevé d'indépendance quant à la façon de représenter les vues des utilisateurs.

Le but de l'étape suivante consistera à élaborer le modèle complet décrivant la vue du monde réel que nous appellerons la structure logique, terme synonyme de modèle global ou conceptuel.

REFERENCES

[6.1] HABRIAS H., "Méthode d'enquête ...", p. 90 ss.
KOREIMANN D.S., "Methoden der Informations-bedarfsanalyse ..." p. 82 ss.
LESCA H., "Introduction à la gestion automatisée" p. 47 ss.
PEAUCELLE J.L., "Les méthodes de recherche en informatique de gestion ..." p. 210 ss.
SCHMIDT G., "Organisation-Methode und Technik" p. 29 ss.

[6.2] Cette idée est proposée par GROCHLA E. dans "Integrierte Gesamtmodelle der Daten-verarbeitung".

[6.3] Plusieurs modèles sont comparés dans BENCI G. et a. "Introductory Report", p. 1 ss.

[6.4] Entre autre CHEN P.P.S., "The Entity Relation-ship-Model ...", p. 9 ss.
DENEHEFFE C. et a., "The Individual Model" p. 89 ss.
RICHTER G., "On The Relationship ..." p. 21 ss.

[6.5] DOUQUE B.C.M., NIJSEEN G.M. (ed.), "Data Base Description".

[6.6] CODD E.F., "A Relational Model ..." p. 377 ss.
DATE C.J., "An Introduction ..., p. 61 ss.
DELOBEL C., "Cours ...", p. 21 ss.
WEDEKIND H., "Datenbanksysteme I", p. 41 ss.

[6.7] Cette affirmation n'est pas toujours valable, par exemple le langage "alpha" développé par Codd n'est facilement accessible que pour un utilisateur familiarisé avec la logique et la notation mathématique.

CODD E.F., "A Database Sublanguage ..." p. 35 ss.
"Relational Completeness ..." p. 65 ss.

DELOBEL C., cité en [6.6], p. 34 ss.

WEDEKIND H., cité en [6.6], p. 115 ss.

Par contre, cette affirmation est déjà plus valable en ce qui concerne le langage "Sequel"

CHAMBERLIN D.D. et a., "Sequel ...", p. 249 ss.

WEDEKIND H., cité en [6.6], p. 163 ss.

[6.8] CODD E.F., "Further Normalization ...", p. 33 ss.

DATE C.J., cité en [6.6], p. 95 ss.

DELOBEL C., cité en [6.6], p. 45 ss.

WEDEKIND H., cité en [6.6], p. 44 ss.

[6.9] WANG C.P., WEDEKIND H., dans "A Segment Synthesis in ...", p. 71 ss., proposent une telle approche.

[6.10] Les notions de fermeture et de noyau, ainsi qu'une méthode d'analyse appropriée, ont été citées pour la première fois par Delobel dans

DELOBEL C., CASEY R.G., "Decomposition of a Data Base ...". Consultez également

DELOBEL C., cité en [6.6], p. 45 ss.

WEDEKIND H., cité en [6.6], p. 73 ss.

[6.11] Consultez les explications données dans ce chapitre aux alinéas a) et b) et les références citées.

[6.12] SENKO M.E. et a., "Data Structures ...", p. 30 ss.

SENKO M.E., "The Data Independent Accessing Model", p. 81 ss.

[6.13] LYON K., "An Introduction to Data Base Design".

[6.14] Traduction libre tirée de LUTZ T., "Die Stellung der Datenbank ...", p. 28.

[6.15] MERTEN H., "Datenbankorganisation", p. 37, définit un segment comme étant une unité logique composée de une ou plusieurs données.

IBM "IMS/VS General Information ..." p. 4.10 ss. et "IMS/VS Application ...", p. 5.10 ss. ; ne donne plus de précision.

[6.16] Une critique de cette méthode est présentée au chapitre 7.4.

[6.17] GROCHLA E. et a., "Gestaltungskriterien ..." p. 81 ss.

GILLNER R., "Die Strukturierung ...", p. 118 ss.

[6.18] ABRIAL J.R., "Data Semantics", p. 3 ss.

DELOBEL C., cité en [6.6], p. 11 ss.

[6.19] Adabas est un SGBD qui alloue à chaque enregistrement un numéro interne logique (ISN) unique, Adabas "Einführung", p. III-5.

[6.20] L'exemple précédent représenté à la figure 6.11 c illustre le même phénomène.

Chapitre 7

Conception de la Structure logique

7.1. LES CONTRAINTES D'INTEGRITE ET DE SECURITE

7.1.1. Définition

Le maintien de la cohérence du modèle après des opérations de manipulation est un objectif primordial. En effet, puisqu'un utilisateur quelconque ne connaît qu'une partie du modèle, il peut, en effectuant des manipulations sur sa vue partielle, altérer la signification du contenu de la banque de données ou effectuer des modifications non conformes à la représentation du monde réel décrit par le modèle global.

D'autre part, tout utilisateur doit pouvoir se fier à la qualité des données qu'il va utiliser et être certain qu'aucun autre utilisateur ne peut les détériorer. Indirectement se pose donc le problème de sécurité, c'est-à-dire du contrôle des accès et des opérations formulées par les utilisateurs.

L'intégrité est un problème central d'un système de gestion de banque de données. Dans les systèmes conventionnels, les problèmes ne se posent pas de façon aussi cruciale, puisque l'accès aux informations est en général séquentiel et les données sont propres aux applications concernées.

La notion d'intégrité recouvre le concept de maintien de la qualité des données et de leur existence, notion à distinguer de la sécurité qui recouvre toutes mesures capables de protéger les données contre des accès non autorisés. Les procédures déterminant les règles d'autorisation d'accès et les opérations exécutables se nomment contraintes de confidentialité.

Remarque: Il est nécessaire de distinguer les problèmes relatifs à la sécurité physique tels que prévention contre les incendies, accès à la salle machine,

archivage, procédure de recouvrement des fichiers, vols, etc ...

L'intégrité est liée étroitement à la sécurité puisque le maintien de la qualité des informations exigent des mesures de protection. Cette notion d'intégrité soulève les problèmes relatifs aux:

- partage de l'information et de protection qui en découlent
- autorisations d'accès (confidentialité)
- conservation des informations.

A l'intégrité des données est également liée la notion d'intégrité des programmes (déroulement des opérations), du matériel et du logiciel d'exploitation.

L'étude de l'intégrité doit s'effectuer au niveau logique, car elle découle directement de la signification des faits réels représentés au niveau du modèle, signification qui doit être maintenue à tout prix. D'autre part, elle influencera directement la réalisation physique puisqu'elle détermine les mesures de protection à prendre en vue de respecter l'intégrité de la banque de données. Les mesures et les différentes possibilités qui existent seront analysées au cours de l'étape de la conception physique.

Remarque: Un SGBD doit pouvoir vérifier les contraintes d'intégrité et de confidentialité à l'aide d'un langage spécifique. Ce n'est pas le cas des SGBD actuellement commercialisés (7.1), mais des prototypes tels que INGRES (7.2) ou System R (7.3) prévoient ce contrôle centralisé. Dans les autres cas, des procédures adéquates de contrôle du respect des contraintes seront exécutées lors de l'initialisation d'une requête.

7.1.2. Les contraintes d'intégrité

Lorsqu'un utilisateur désire effectuer une des trois opérations: insertion, modification ou suppression, il faut s'assurer que les contraintes d'intégrité soient toujours vérifiées, quelle que soit la valeur des données. Ce problème d'intégrité des données se pose à différents niveaux:

a) le type de données
   . contraintes quant aux valeurs possibles d'une donnée et ceci sans être conditionnées par un autre ensemble de valeur (=plage de valeur)

     Ex: sexe = M, F
         état-civil = Marié, célibataire, veuf, divorcé

   . format des données

     Ex: caractère ou numérique

   . longueur des données
     spécifier la longueur en indiquant le nombre de caractères nécessaires à la représentation de la donnée concernée

   . codification
     afin de faciliter la représentation de certaine donnée

b) les relations

b1) type de relation
   . cardinalité
     Considérons le modèle binaire où toute catégorie appartient à une ou à plusieurs relations. La cardinalité de la relation est indiquée par les paramètres d'existence. Reprenons l'exemple de la figure 6.18 où la relation R_1 pourrait être:

R_1 = rel (distributeur, adresse, adressedistributeur, afn = (1, 3, 2,1) inv (1, 1, 1))

ce qui signifie que chaque distributeur doit avoir au moins une adresse et inversément qu'une adresse donnéee ne référence qu'un seul distributeur.

. plage de valeur

Ex: Numéro - ville (entre 1000 et 9000)
Empl - sal ≠ 0

. rapport entre une occurence donnée et l'ensemble des réalisations d'un type de relation donné
L'existence unique d'une valeur est garantie par les paramètres d'existence

Ex: R_4 = rel (no, client, clientdenuméro, afn (0, 1, 1) inv (1, 1, 1)).

b2) interdépendance entre différentes relations

Il existe une foule de cas dans lesquels l'existence d'une occurence donnée implique le respect de contraintes sur d'autres relations.

. cardinalité
Lorsqu'une catégorie appartient à plusieurs relations, il est nécessaire de vérifier si une manipulation sur l'une d'entre elle ne contredit les autres. Considérons à nouveau l'exemple de la fig. 6.18 et admettons les faits réels suivants:

R_2 = rel (adresse, dénomination, descriptionadr = afn (1, 1, 1) inv (1, 1, 1))

R_3 = rel (adresse, type, typeadresse = afn (1, 1, 1) inv (1, 1000, 50)).

Une insertion d'une occurence de type R_2 nécessite automatiquement l'insertion d'une occurence de type R_3, puisqu'à toute adresse doit correspondre

au moins une valeur de la catégorie type.

. valeurs

Ex: - (figure 6.13a)
Le total des factures d'un abonné (a x b) doit être égal au solde dû (c)

- la quantité livrée doit être inférieure à la quantité en stock.

Remarques : - *Il ne peut être garanti qu'une occurrence de l'ensemble de valeurs des noms de personne ne soit enregistrée dans une occurrence de la relation 'nom-de-machine'.*

- *Le problème du décalage dans l'espace entre l'apparition d'un état de fait dans le monde réel et son enregistrement dans la base n'est pas résolu. L'utilisateur doit en tenir compte lors de la déclaration de requêtes et d'opérations de mise à jour.*

- *Il est intéressant de souligner que les paramètres d'existence expriment un certain nombre de contraintes d'intégrité.*

Les contraintes d'intégrité seront représentées

. par les paramètres d'existence dans le tableau des relations (figure 7.1) ;

. en déterminant pour chaque catégorie les plages de valeu la longueur maximale et le format, par exemple sous forme de tableau (figure 7.2) ;

. les contraintes d'intégrité relatif à une relation donnée seront représentées dans une colonne 'observations' du tableau des relations (figure 7.1) ;

. lorsque des catégories équivalentes ont été regroupées en une seule catégorie, les différentes unités de valeur seront décrites dans la colonne 'observations' du tableau des relations. La catégorie 'numéro' dans le tableau de la figure 7.1 en est un exemple ;

. en indiquant dans un tableau le numéro de la relation et les contraintes incluant d'autres relations. Il existe différentes façons de décrire ces contraintes:

  ** procédures programmées comme le propose Abrial dans le modèle Data Semantics (7.4)

  ** à l'aide de tables de décision qui pourront ensuite être directement intégrées dans les programmes d'application, à condition d'avoir un processeur de tables de décision (figure 7.3)

  ** verbalement (figure 7.2)

  ** sous forme de prédicats à l'aide d'un langage de haut niveau: Alpha (7.5) ou Sequel (7.6). Les SGBD Ingres et System R proposent une telle solution (7.7).

La méthode que nous préconisons, présentée à l'aide d'un exemple aux figures 7.1 et 7.2, est d'utilisation plus pratique à ce niveau de l'approche, car

  . elle minimise le travail d'écriture

  . elle facilite l'exécution de corrections rendues nécessaires par des modifications, le passage d'un tableau à un autre étant facilité par la référence 'no de relation'.

D'autre part, il ne faut pas oublier qu'aucun des SGBD commercialisés actuellement n'offre de langages permettant la description de toutes les contraintes d'intégrité et qui seraient ensuite vérifiés par le système lors des opérations de manipulation de données. En ce qui concerne le problème du partage des ressources, mécanisme nécessaire au maintien de l'intégrité de la banque de données, il n'est de loin pas résolu (7.8), des algorithmes en vue de résoudre les problèmes posés sont proposés en (7.9).

n°	nom de la relation	catégories		afn			afn inverse			observations
		noeud d'entrée	noeud de sortie	min	max	$\bar{x}$	min	max	$\bar{x}$	
R1	n° distributeur	distributeur	n°	1	1	1	1	1	1	4, n, 0-4999
R2	nomdistributeur	distributeur	nom	1	1	1	1	1	1	
R3	adressedistributeur	distributeur	adresse	1	3	2,1	1	1	1	
R4	n°teldeadresse	adresse	n°	0	1	0,9	1	1	1	10, n
R5	typed'adresse	adresse	type	1	1	1	1	1000	500	
R6	dénominationadresse	adresse	dénomination	1	1	1	1	1	1	
R7	secteurdistribution	distributeur	distribution	1	50	5	1	1	1	
R8	villededistribution	distribution	ville	1	1	1	0	300	10	
R9	ruededistribution	distribution	rue	1	1	1	1	100	5	
R10	nomdeville	ville	nom	1	1	1	1	1	1	
⋮										
R_{n-2}	salairedistributeur	distributeur	montant	1	1	1	1	1000	400	
R_{n-1}	typeabonnement	conditions	codeabo	1	1	1	0	200000	25000	
R_n	prixabonnement	conditions	codeabo	1	1	1	0	200000	10000	

Figure 7.1 - Tableau descriptif du graphe logique

Catégories (relation)	Longueur	Format	Plage de valeur
nom (R_2)	30	A	
type (R_5)	2	A	NO, PA, PR, PU
⋮			
montant (R_{n-2})	4,2	N	0 < X < 2000

Fig. 7.2 a) - Tableau des contraintes relatives aux catégories

n° de contrainte	n° de relation	contraintes
Ci_1	R_7	si nombre occurrences > 2 alors r_{n-2} > 800
Ci_2	R_{n-2}	égale à contrainte C_1
Ci_3	R_{n-1}	si r_{n-1} ≠ 'NO' alors r_n > 1
Ci_4	R_n	égale à C_3

Fig. 7.2 b) - Tableau des contraintes interrelations

règles / conditions/actions	1	2	3	
. nombre secteur distribution > 2	-	Y	-	
. salairedistributeur > 800	-	N	-	
. typeabonnement ≠ NO	-	-	Y	
. prixabonnement > 1	-	-	N	
. refus	-	X	X	
. exécuter modification	X	-	-	

Fig. 7.3 - Contraintes d'intégrité décrites à l'aide d'une table de décision

Remarques au sujet du modèle relationnel :

Les relations fonctionnelles sont des contraintes d'intégrité.
Mais il existe un nombre important de contraintes qui ne sont pas décrites dans le modèle :

- *lorsqu'un index apparaît dans plusieurs relations, par exemple dans les relations*
  R_1 (#*fournisseur*, *nom*)
  R_2 (#*article*, *nom*)
  R_3 (#*fournisseur*, #*article*)
  R_4 (#*client*, #*commande*, #*article*, *quantité commandée*, *quantité livrée*)
- *lorsqu'il existe un lien entre deux attributs (non-index), comme dans* R_5(#*client*, *nom*, *ville*,

dette, statut), il n'est pas évident que l'existence d'une dette nécessite l'existence d'un statut. Le même cas apparaît dans la relation R_4 où quantité livrée ≤ quantité commandée.

. *dans le cas d'une relation fonctionnelle non élémentaire*
R_6 (*#client, #commande, #article, texte, quantité*)

En comparant le modèle relationnel avec les conditions formulées dans ce chapitre, il apparaît qu'un plus grand effort doit être fait lors de la description des contraintes d'intégrité, puisque peu d'entre elles sont explicites au niveau du modèle. Illustrons à l'aide du langage Sequel l'expression de telles contraintes

a) Select #client, #commande from R_4
where Qtécommande ≤ Qtélivrée

b) (Select #article from R_4) ⊆ (Select #article from R_2) (article doit figurer dans l'ensemble R_2)

Remarque au sujet du modèle Codasyl :

Le set représente une fonction d'intégrité, car un suivant a toujours un précédent. Une contrainte n'existe donc que si un chemin d'accès est défini. De même, il est possible de déclarer la clé comme devant être unique dans l'ensemble des occurrences d'un 'record' donné. Pour pouvoir considérer le set comme contrainte d'intégrité au niveau du modèle conceptuel, il est nécessaire de spécifier le 'location mode Mandatory Automatic'.

7.1.3. Les contraintes de confidentialité

Afin de garantir l'intégrité de la banque de données, il est nécessaire de contrôler les demandes d'accès et les opérations qui en découlent. On dira d'un système qu'il est sûr s'il propose des mesures adéquates de

contrôle.

Il s'agit donc de déterminer les contraintes en définissant pour l'utilisateur concerné :

- les relations
- les plages de valeur ou conditions
- les opérations

qu'ils osent effectuer, ses droits d'accès étant ainsi délimités. Le degré de confidentialité, beaucoup ou peu de restrictions d'accès, dépend en premier lieu de la nature des informations contenues dans la banque de données, raison pour laquelle les contraintes de confidentialité sont à fixer à ce niveau de l'approche.

Remarque : Le fait de déterminer un sous-schéma pour un utilisateur donné consiste à fixer les données auxquelles l'utilisateur pourra accéder et à décrire les opérations qu'il osera effectuer. Dans certains SGBD, par exemple IDMS, les contraintes sont définies dans le sous-schéma.
Par contre, le problème se pose lorsque les accès à la banque de données s'effectuent par l'intermédiaire d'un langage non procédural, car il n'y a pas de passage par le filtre sous-schéma.

Le contrôle du respect des contraintes de confidentialité pose le problème de l'identification de l'utilisateur, lequel indique au système à l'aide d'un mot de passe, badge, etc ..., son intention d'accéder à la banque de données et celui de l'autenticité qui consiste à vérifier que l'utilisateur correspond à l'identification en l'obligeant à donner des informations supplémentaires, par exemple en donnant son ancien mot de passe (7.10).

La description des contraintes de confidentialité peut se faire de différentes manières :

. modèle descriptif non-procédural
. modèle procédural.

a) modèle descriptif

Les contraintes sont décrites à l'aide d'un langage, par exemple Alpha, Square ou Sequel, qui permettent de formuler des conditions dont le résultat d'une qualification correspond à une requête, résultat qui peut être considéré comme une vue partielle. Cette vue partielle est une relation, non enregistrée, mais définie logiquement à partir des relations de base ou élémentaires. Une autorisation consiste donc à définir :

. une vue partielle
. des conditions
. les opérations autorisées.

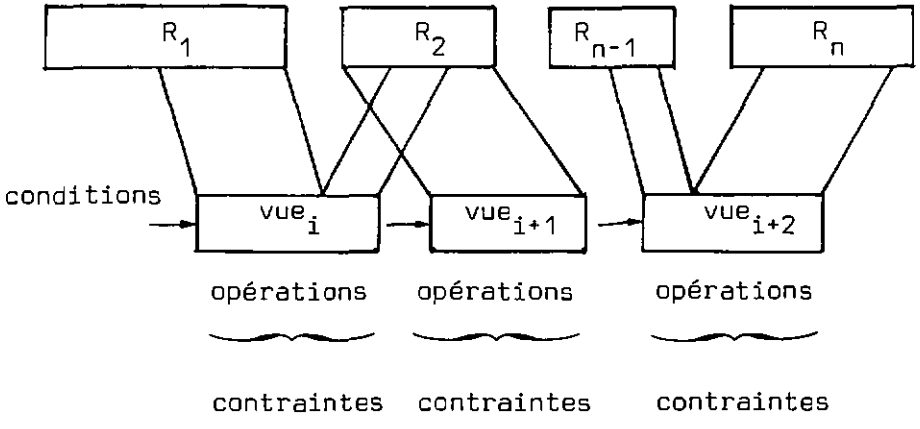


Fig. 7.4 - Les contraintes de confidentialité (7.11)

Prenons un exemple pour illustrer le phénomène de la figure 7.4 et considérons les relations

Article (#article, nom, prix)

Commande (#commande, date, # article, quantité)

Définissons d'abord une vue partielle qui peut être soumise à une condition (a1) ou non (a2).

```
a1  Define Liste 1 Table as :
    Select #article, nom
    From Article
    Where prix < 100 ;

a2  Define Liste 2 Table as :
    Select Article, Commande
    Where Article, #article = Commande,#article
```

Ensuite, les opérations que l'utilisateur est autorisé à effectuer sont à décrire. Ce droit peut être plus ou moins restrictif

- seulement lire Article
- lire et modifier Article
- lire Article et modifier le constituant Article, nom
- lire Article et modifier le constituant prix si celui-ci est < 100
- lire seulement le salaire propre à l'utilisateur mais pas les salaires d'autres personnes.

Reprenons l'avant-dernier cas et décrivons-le à titre d'illustration en se référant aux propositions de Chamberlin (7.12).

```
Define Liste 3 Table as :
Select Article
Grant Liste 3 To Utilisateur n° = 'A' :
    (Grant = 'NO', Revoke = 'NO', Destroy = 'NO',
     Insert = 'NO', Delete = 'NO', Update = 'NO') ;
```

```
Grant Liste 1 To Utilisateur n° = 'A' :
   (Grant = 'NO', Revoke = 'NO', Destroy = 'NO',
    Insert = 'NO', Delete = 'NO', nom.Update = .
    'YES') ;
```

La méthode décrite ci-dessus a été adoptée dans le système R, une autre application est celle proposée par le système Ingres (7.13). Les contraintes de sécurité seront donc décrites avec un tel langage si on a la possibilité d'utiliser un tel système.

Remarque : Un problème de consistance apparaît lorsque des opérations de modification, suppression ou insertion sont autorisées sur des vues partielles ne se référant pas à des relations élémentaires et à des relations ne contenant pas de champs membres de l'index que l'on retrouve dans d'autres relations.

b) *modèle procédural*

Dans le modèle 'Codasyl', le sous-schéma détermine les données auxquelles un utilisateur peut accéder (7.14).

Au niveau du schéma, l'administrateur de la banque de données définit des verrous (Privacy locks) en indiquant par type d'opération les clés nécessaires pour ouvrir le verrou. Dans le programme d'application, l'utilisateur transmettra la valeur de sa clé (Privacy key). La clé pourra être soit :

- un mot de passe
- le nom d'une procédure, par exemple

```
Privacy lock for update is Proc1
Proc1 : si prix < 100 et utilisateur n° = 140
        alors rejet
```

Les contraintes peuvent se référer à plusieurs niveaux: le 'schema', 'l'area' en tant que sous-ensemble de l'espace physique, le 'record', le 'set' et le champ.

L'inconvénient avec ce modèle est le fait que l'on se réfère à des notions d'implantation physique.

Dans un but d'indépendance, nous proposons dans notre approche de définir dans un tableau, pour chaque relation, l'existence de contraintes en indiquant

- le n° de la relation
- l'utilisateur
- les opérations (l = lire, i = insérer, s = supprier, m = modifier)
- les conditions

comme le montre l'exemple de la figure 7.5, qui est une illustration des contraintes de confidentialité du modèle décrit par la figure 7.1.

Une autre possibilité consiste à décrire les contraintes à l'aide de tables de décision. Cette solution est à rejeter car, au niveau des réalisations, une partie du processus de contrôle peut être effectuée par différents éléments :

- le moniteur de télétraitement
- le SGBD qui n'est capable que de contrôler des contraintes sans conditions
- des procédures de programme dans certains cas, avec ici la possibilité de les réaliser à l'aide de tables de décision
- le SGBD qui est capable de couvrir tous les cas.

L'indépendance est obtenue d'une part par la relative facilité de traduire les contraintes :

- dans un langage du type Sequel ou autre
- dans le DDL d'un système Codasyl ou de IMS
- dans des procédures programmes adéquates

et d'autre part par l'élimination du problème lié aux opérations effectuées sur les vues partielles (7.15), puisque nous ne considérons que des faits élémentaires.

rel	utilisateurs	opérations	conditions
R_{n-2}	'A'	l	si montant dans r_{n-2} < 1000
		m, s, i	si montant dans r_{n-2} < 100 et n° dans r_1 ≤ 999
	'B'	l, m, s, i	
R_1 R_2 R_3 ⋮ R_9	'C','O','E'	l	
	'F','G'	l	
	'A','B'	l, m, s, i	
R_{n-1}	'C','O','E'	l, m, s, i	
	'F','G'	l, m, s, i	
R_n	'A','B'	l	

Fig. 7.5 - Tableau des contraintes de confidentialité

Les mécanismes de contrôle d'accès sont faciles à réaliser lorsque l'utilisateur se réfère à une vue partielle définie à l'avance et gérée par le système. Dans le cas du modèle relationnel, l'utilisation d'un langage de manipulation non procédural pose des problèmes qui ne sont pas encore résolus e.a (7.16) :

a) il y a possibilité de détourner les contraintes en formulant d'autres requêtes pour arriver par approche successive aux données dont l'accès est en fait interdit

b) quelle doit être la réaction du système face à des requêtes ne respectant pas les contraintes.

7.2. RELATIONS ENTRE CATEGORIES ET STRUCTURE LOGIQUE

A l'étape précédente, 'Analyse sémantique', nous avons déterminé les catégories et les associations entre catégories, ces dernières indépendamment du type de relation. Nous avons définis les fonctions d'accès en indiquant les paramètres d'existence, mais sans qualifier ceux-ci. D'autre part, lorsque nous avons étudié les problèmes d'intégrité, le fait qu'une fonction d'accès existe une fois au moins et au maximum une fois, signifie qu'à une valeur d'une catégorie n'est associée qu'une valeur de la catégorie en sortie de la fonction d'accès et inversément. Cette contrainte d'intégrité nous autorise donc à regrouper, la relation concernée étant intégrée en une catégorie dite non élémentaire.

L'objectif de cette activité consiste donc à analyser le type des relations entre les catégories en vue de simplifier la représentation graphique du modèle. Le résultat de cette opération permet d'obtenir la structure logique ou en d'autres termes le modèle conceptuel. Le tableau des relations élaboré à l'étape précédente (figure 7.1) constitue le point de départ de notre analyse:

. lorsque les paramètres d'existence des relations sont tous de valeur égale à 1 (afn et afninverse), les relations concernées seront intégrées en une catégorie non élémentaire à laquelle on donnera un nom. Prenons à la fig. 7.1 les relations R_1 à

R_9. Nous constatons que R_1, R_2, R_6 et leur inverse satisfont aux conditions requises. Les catégories concernées, 'distributeur', 'n°', 'nom', seront regroupées en une catégorie dénommée 'distributeur', tandis que 'adresse' et 'dénomination' seront regroupées sous le nom 'adresse'

- dans l'ensemble des liaisons restantes, détecter les relations de type 1 : n ou n : 1 et déterminer une catégorie qui permette d'effectuer la liaison entre les catégories restantes et ceci dans un sens comme dans l'autre. Ces catégories remplissent en fait une fonction d'identification, il s'agira donc de les choisir en conséquence (de longueur restreinte et si possible de format numérique), la relation R_3 de la figure 7.1 est un exemple, le 'n° personne' a été choisi comme identificateur de liaison
- les relations de type n : m devront être représentées en introduisant une catégorie-liaison, la liaison ne pouvant se faire avec un seul identificateur. L'exemple de la figure 7.9 illustre ce phénomène
- représenter graphiquement la structure logique ainsi obtenue en utilisant les symboles suivants :

catégorie non élémentaire

catégorie élémentaire

indique l'appartenance et en même temps les fonctions d'accès (afn et afninverse)

catégorie-liaison

Pour éviter des redondances, les catégories élémentaires appartenant à une catégorie non élémentaire ne seront pas représentées au niveau du groupe. La figure 7.6 est un exemple d'illustration graphique du tableau de la figure 7.1, relations R_1 jusqu'à R_9.

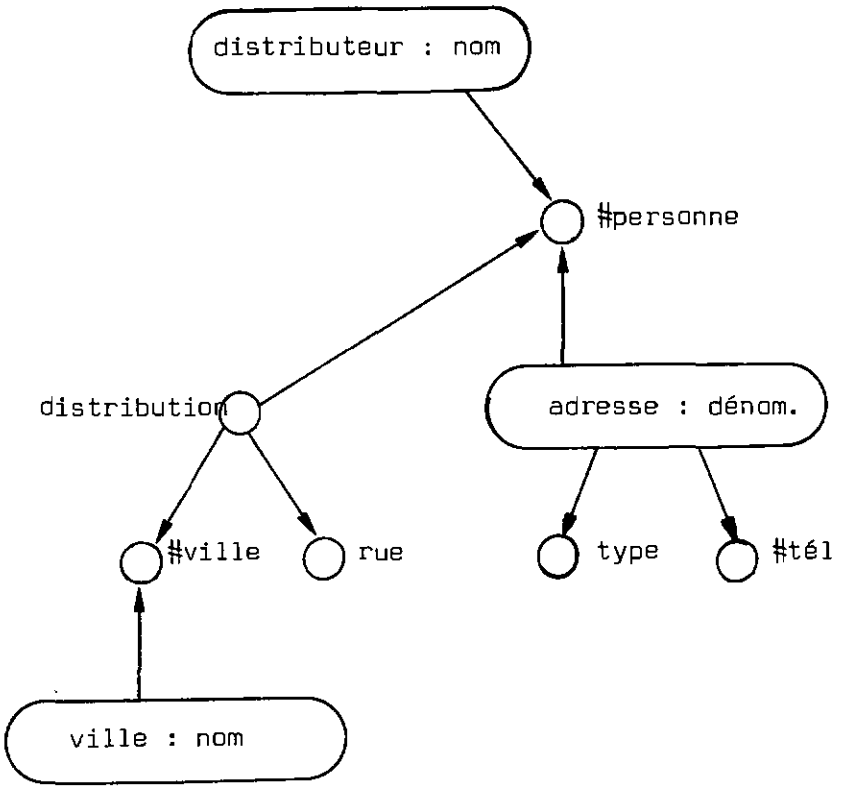


Fig. 7.6 - Structure logique

Un autre exemple est celui de la nomenclature représentée graphiquement à la figure 6.12 et dont le graphe logique serait celui de la figure 7.7.

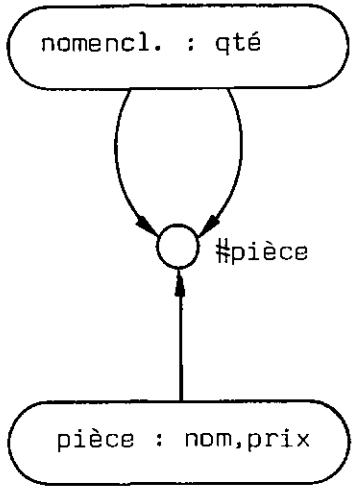


Fig. 7.7 - Structure logique du graphe de la figure 6.12

Quant à l'exemple de la figure 6.11 c, on obtiendrait le graphe suivant (figure 7.8) :

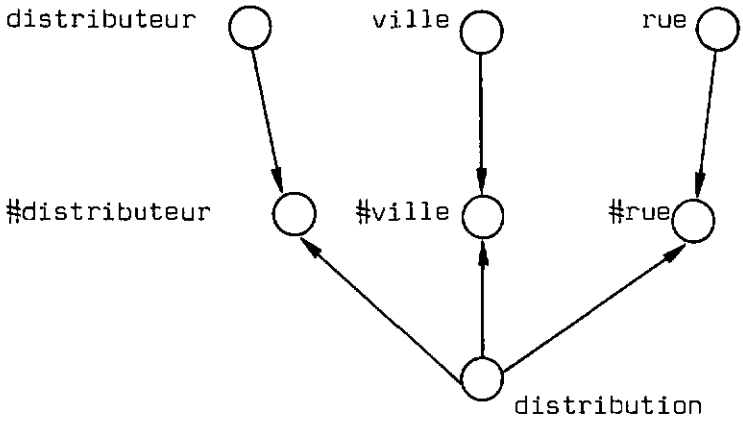


Fig. 7.8 - Structure logique du graphe de la figure 6.11 c

Un autre exemple, avec des liaisons du type n : m est celui de la figure 7.9 dont la structure logique est représentée à la figure 7.10.

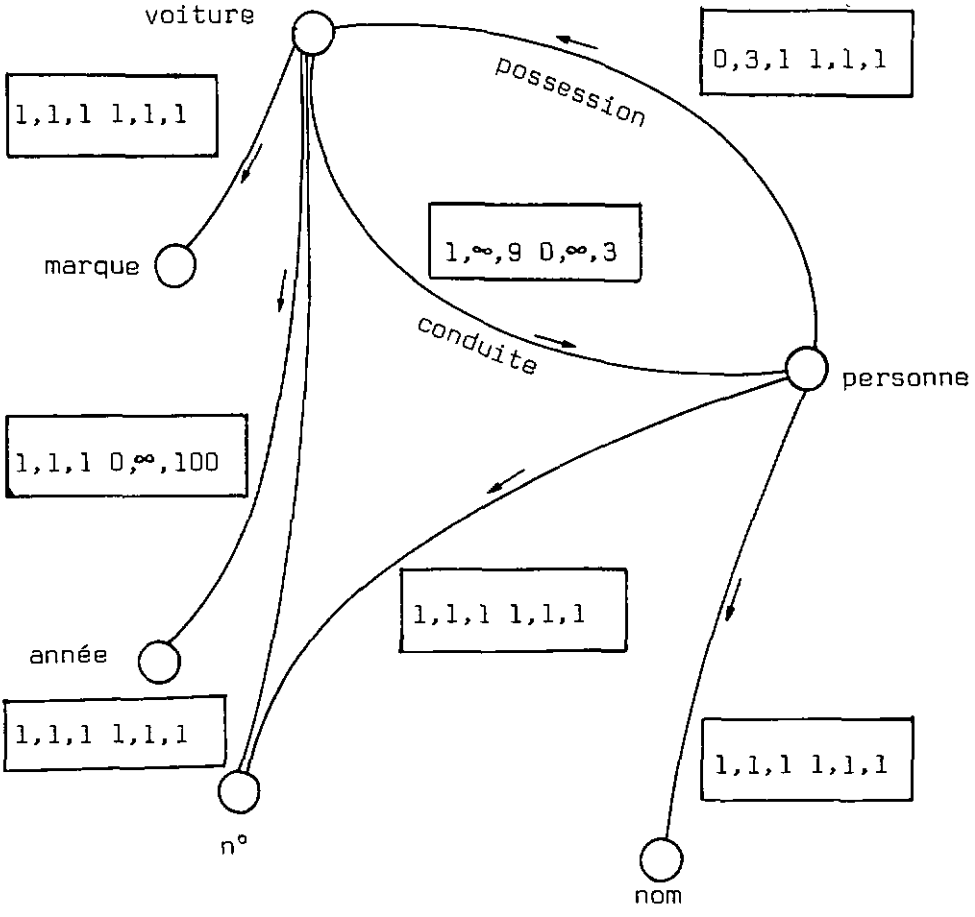


Fig. 7.9 - Graphe logique

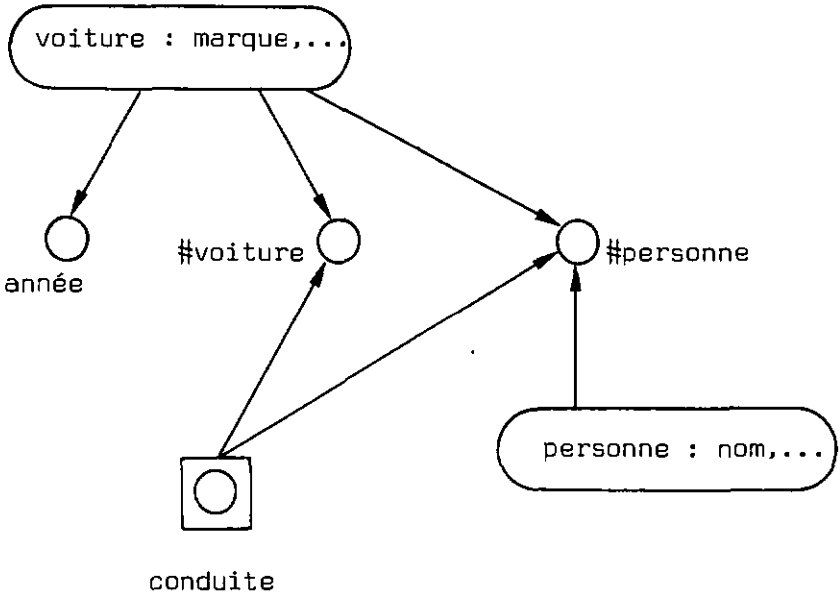


Fig. 7.10 - Structure logique du graphe de la figure 7.9

Cas spécial :

- une relation a comme noeud-cible le noeud de départ, comme c'est le cas de la figure 7.11.

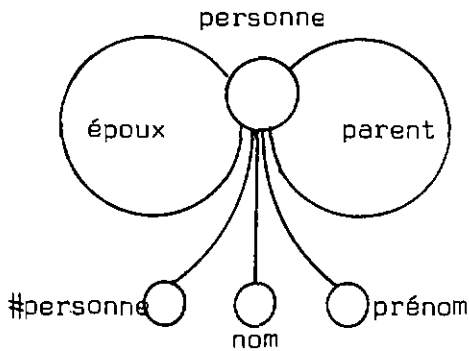


Fig. 7.11 - Graphe logique

Les relations 'prénom', 'nom' et 'n°' seront regroupées, cette dernière étant choisie comme identificateur. Etant donné qu'une relation est toujours composée d'un noeud d'entrée et d'un noeud de sortie et que nous devons choisir l'identificateur qui permette le passage de l'un à l'autre, nous sommes obligés d'introduire une catégorie liaison correspondant à la relation elle-même. Comme les noeuds d'entrée et de sortie sont les mêmes, ils possèdent donc le même identificateur. La figure 7.12 est une illustration de ce phénomène.

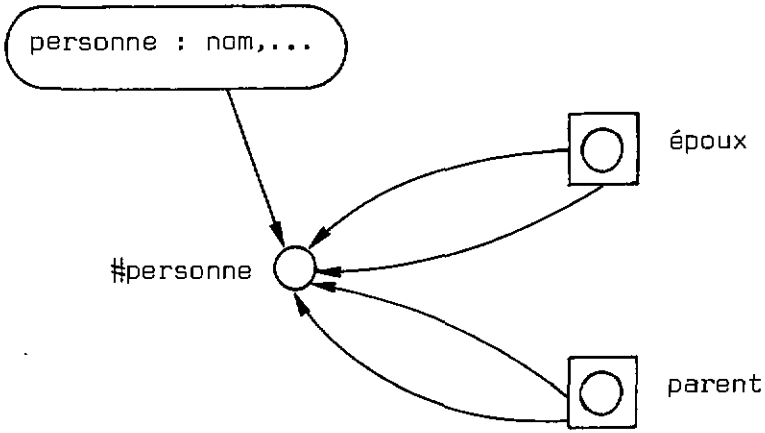


Fig. 7.12 - Structure logique du graphe de la figure 7.11

7.3. PASSAGE A D'AUTRES MODELES DE DONNEES

Un des principes de base de la méthode d'approche (voir chapitre 4.2) est de faciliter le passage à d'autres modèles de données. Nous ne prendrons en considération que le modèle relationnel et Codasyl. Une étude approfondie du passage de modèles hiérarchique et Codasyl vers le modèle relationnel et inversement est présentée en (7.17).

7.3.1. Le modèle relationnel

Il est sans autre possible de passer du modèle proposé au chapitre précédent au modèle relationnel. Il suffit d'englober dans un noeud tous ceux qui y sont reliés par une flèche, afin d'obtenir des relations. A titre d'exemple, nous traduirons les exemples du chapitre 7.2 en relation n-aire.

Figure 7.6 : R_1 : distributeur (#personne, nom)

R_2 : adresse (#personne, type, dénomination, #tél)

R_3 : distribution (#personne, #ville, #rue)

R_4 : ville (#ville, nom)

Figure 7.7 : R_5 : nomenclature (#pièce composant, #pièce composé, quantité)

R_6 : pièce (#pièce, nom, prix)

Figure 7.8 : R_7 : distributeur (#distr., nom, ...)

R_8 : ville (#ville, nom, ...)

R_9 : rue (#rue, nom, ...)

R_{10} : distribution (#distr., #ville, #rue)

Figure 7.10 : R_{11} : voiture (#voiture, année, marque, personne)

R_{12} : personne (#personne, nom)

R_{13} : possession (#voiture, #personne)

Figure 7.12 : R_{14} : personne (#personne, nom, prénom)

R_{15} : époux (#personne, #personneduconjoint)

R_{16} : parent (#personne, #personneparent)

L'index de la relation ainsi obtenue est composé:

- en premier lieu du ou des identificateurs de liaison
- s'il existe d'autres flèches à partir du noeud considéré, il faut vérifier si pour une valeur de l'index obtenu, il existe une seule valeur pour les constituants dépendant de l'index considéré. Si la réponse est négative, une ou plusieurs des flèches menant à un noeud feuille de type 1:1 devra être ajoutée à l'index jusqu'à ce que la dépendance soit fonctionnelle. Dans le cas de la figure 7.6, la catégorie type doit faire partie de l'index.

Les relations ainsi obtenues sont en 3FN, car les redondances ont été éliminées au niveau de l'analyse sémantique, les liaisons entre catégories soigneusement définies et il existe une relation fonctionnelle élémentaire entre l'index et les constituants. La notion de relation fonctionnelle élémentaire a été expliquée au chapitre 6.2.2. Pour prouver que les relations obtenues sont bien en 3FN et élémentaires, il suffit de comparer les résultats de la figure 7.13 avec ceux du chapitre concerné.

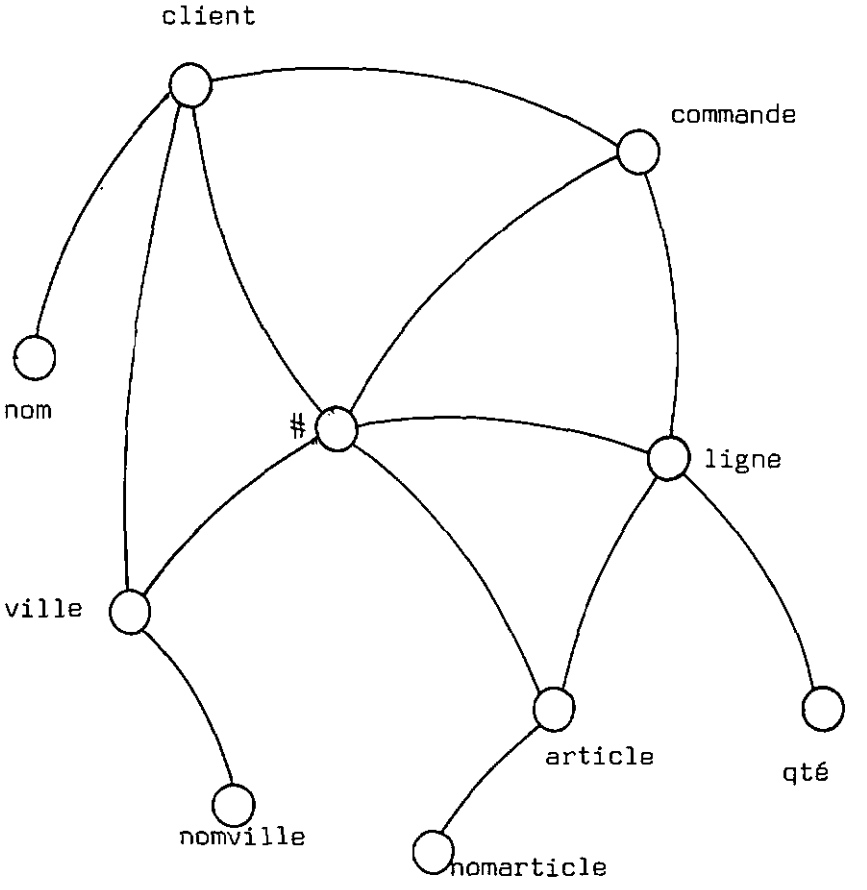


Fig. 7.13 a - Graphe logique

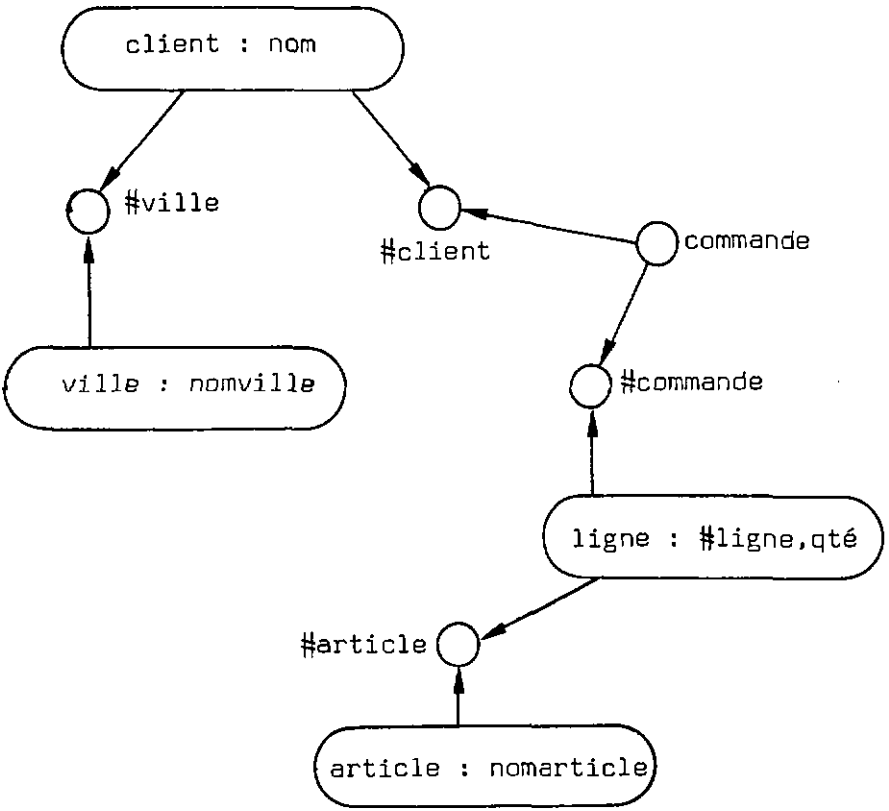


Fig. 7.13 b - Structure logique du graphe de la figure 7.13 a

R_1 client (#client, nom, #ville)

R_2 ville (#ville, nomville)

R_3 commande (#commande, #client)

R_4 ligne (#commande, #ligne, #article, qté)

R_5 article (#article, nom)

Un cas spécial à étudier soigneusement est l'existence de boucles (cycles) dans le graphe, sans qu'il y ait contradiction, ce qui signifie que plus d'un chemin conduit au même résultat. Le problème qui se pose est de choisir un ensemble de relation qui, après n'importe quelle opération de mise à jour, respecte les contraintes d'intégrité. Prenons l'exemple proposé en (7.18) et présentons-le selon la méthode élaborée (figures 7.14 a 7.14 b) et passons au modèle relationnel.

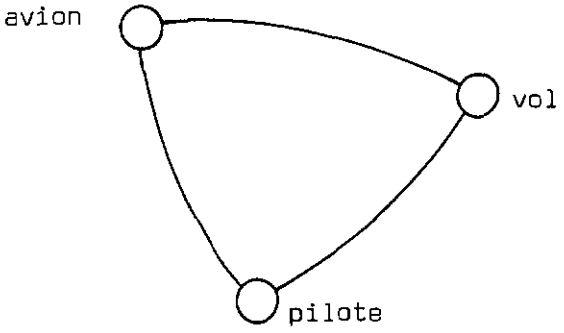


Fig. 7.14 a

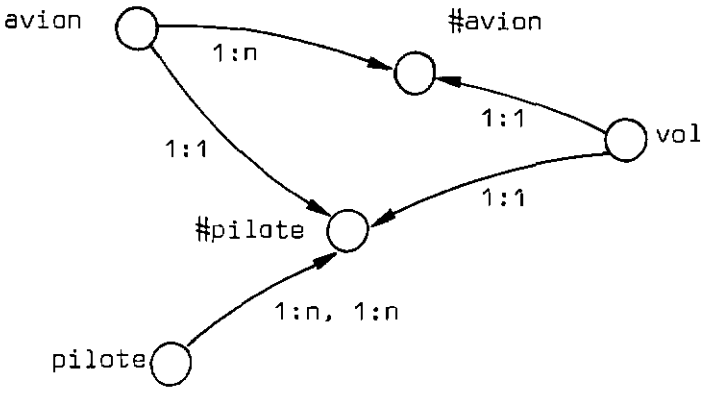


Fig. 7.14 b

Selon le théorème démontré et qui correspond, en terme simple, à choisir le chemin le plus long, les relations à choisir sont donc R_1(vol, avion) et R_2(avion, pilote).

7.3.2. Codasyl

Le modèle Codasyl est caractérisé par la relation père-fils (owner-member) appelé le "set" qui relie entre eux deux records. Le passage se fait de la façon suivante :

- les noeuds, points de départ d'une ou plusieurs flèches, sont des records représentés graphiquement à l'aide d'un rectangle
- le type de liaison déterminera la relation père-fils, l'identificateur de liaison étant transféré dans le record 'père'
- en ce qui concerne les liaisons restantes, la création d'un record et d'un set ou de leur intégration dans un record existant dépend des possibilités d'accès que l'on désire implanter.

Comparons le schéma de la figure 7.14 avec celui de la figure 6.3 qui sont graphiquement différents malgré le fait qu'ils représentent la même vue du monde réel considéré.

Les chemins d'accès étant étudiés ultérieurement, il est avantageux de définir trop de "records" pour ensuite les intégrer au moment de la réalisation physique, ainsi le modèle est moins dépendant de contraintes d'implantation.

A titre d'exemple, le graphe de la figure 7.6 a été tranformé selon les règles décrites ci-dessus pour aboutir au schéma de la figure 7.15

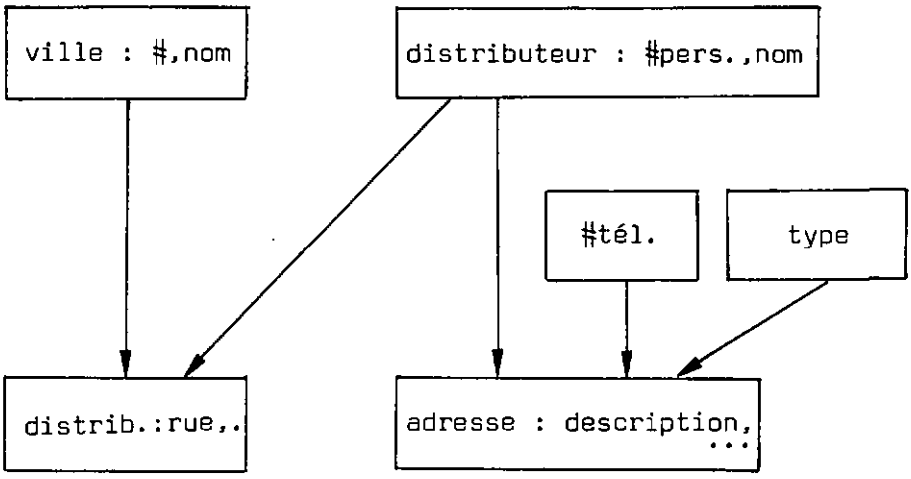


Fig. 7.15 - Exemple de structure Codasyl

Le cas représenté par la figure 7.14 b prendra, selon la méthode proposée, la forme de la figure 7.16.

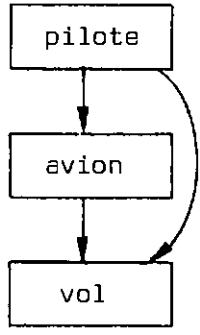


Figure 7.16

Le maintien du set pilote-vol dépendra des contraintes d'accès.

7.4. AUTRE METHODE POUR DETERMINER LA STRUCTURE LOGIQUE

La méthode développée entre autre en (7.19) repose sur le principe fondamental que la structure logique peut se déduire de la fréquence d'utilisation des données nécessaires à la satisfaction des besoins en information. Une telle méthode propose de classer les données par objet en indiquant, sous forme de matrice, le nombre d'utilisation de chaque donnée en une unité de temps. Ensuite, le concepteur groupe les données en segment en se basant sur les résultats fournis par un algorithme d'analyse de cluster, regroupement purement arbitraire car la décision d'intégrer une donnée à un segment plutôt qu'à un autre se fait selon l'appréciation du concepteur.

Les relations entre segments sont déterminées selon le même principe. Pour chaque besoin il s'agit de déterminer la hiérarchie selon laquelle les segments concernés sont utilisés.

Cette méthode ne peut être utilisée car:

- elle ne tient pas compte de faits réels, selon lesquels une liaison entre données (catégorie) existe indépendamment des fréquences d'utilisation
- les données classées dans un groupe 'objet' y sont reliées hiérarchiquement
- elle ne considère que les vues des utilisateurs: des redondances voire des ambiguités peuvent sans autre s'infiltrer
- certains faits réels ne sont pas représentés si

aucun besoin exprimé lors de la conception n'y fait référence

- elle ignore le type de relation entre données (catégorie)
- elle n'implique pas d'analyse sémantique
- les relations entre segments sont étudiés en fonction des utilisations et sont uniquement de type hiérarchique.

Remarque : Nous avons utilisé cette méthode dans le cadre de la conception d'une application pratique. Les résultats obtenus, axés sur différents profils d'utilisation, confirment les critiques formulées ci-dessus.

REFERENCES

[7.1] BAZILLOU P.G., BENCI G.E., "Présentation et analys de SGBD commercialisés ..."

[7.2] STONEBAKER M. et a., "The Design and Implementation of INGRES" p. 208 ss., ainsi que les références citées.

[7.3] ASTRAHAN M.M. et a., "System R ..." p. 108 ss.
KING W.F. "System R".

[7.4] Voir exemple dans ADIBA M. et DELOBEL C., "Les modèles relationnels de bases de données" p. 8.20 ss.

[7.5] DATE C.J., "An Introduction ..." p. 302 ss.
CODD E.F., "A Data Base Sublanguage Founded on ..." p. 35 ss.

[7.6] WEDEKIND H., "Datenbanksysteme II" p. 75 ss.

[7.7] Consultez les références [7.2] et [7.3].

[7.8] ADIBA M., DELOBEL C. cités en [7.4] p. 8.2.

[7.9] GRAY J.N. et a., "Granularity of Locks ..." p. 365 ss.
GARDARIN G., SPACCAPIETRA S., "Integrity of Data Bases ..." p. 395 ss.

[7.10] Pour plus de détail, consultez
WEDEKIND H. cité en [7.6] p. 22 ss.
DATE C.J. cité en [7.5] p. 285 ss.

[7.11] Illustration reprise dans WEDEKIND H. cité en [7.6] p. 51

[7.12] CHAMBERLIN D.D. et a., "Views, Authorization and Locking ..."

[7.13] Pour plus de détail, consultez
WEDEKIND H. cité en [7.6] p. 50 ss. et les références [7.2] et [7.3]

[7.14] DATE C.J. cité en [7.5] p. 292 ss.
WEDEKIND H. cité en [7.6] p. 70 ss.

[7.15] Voir la remarque faite au chapitre 7.1.3 point a).

[7.16] ADIBA M., DELOBEL C. cité en [7.4] p. 8.23 ss.

[7.17] ADIBA M. et a., "An Unified Approach for Modelling ..." p. 636 ss.

[7.18] ADIBA M., DELOBEL C. cité en [7.4] p. 5.1 ss. proposent une méthode permettant un tel choix.
LEONARD M. dans "Aides algorithmiques ..." propose un algorithme.

[7.19] GILLNER R., "Die Struckturierung ..." p. 138 ss.

CHAPITRE 8

LES CHEMINS D'ACCES LOGIQUES

8.1. DESCRIPTION DES PROCESSUS DE TRAITEMENT

L'objectif de toute banque de données consiste à satisfaire au mieux les besoins des utilisateurs. L'analyse de ces besoins permettra de définir les vues partielles spécifiques à chaque utilisation (application). Une application est un ensemble structuré de processus de traitement qui peut se concrétiser par

- une simple demande d'information
- un accès en vue d'effectuer des calculs
- une demande dans le but de transformer le contenu de la banque de données (mise à jour).

Cette phase de description des processus de traitement a pour objectif de décrire de manière plus détaillée les éléments déterminés lors de la phase "modélisation du système de gestion". Ce qui nous intéresse dans le cadre de notre méthode d'approche se résume à:

- définir les chemins d'accès logiques
- vérifier le respect des contraintes d'intégrité et de confidentialité
- compléter la structure logique, si nécessaire.

Au cours de cette étape, l'analyste d'application et l'administrateur de la banque de données étudieront en commun les problèmes cités ci-dessus.

L'objectif de la dernière étape du module "conception logique du système informatique de gestion" (annexe) est d'obtenir d'une part

- la description logique et détaillée des applications

et d'autre part

- la structure logique des données conformément au monde réel pris en considération et capable de

satisfaire les différents besoins en information.

8.2. DESCRIPTION DES REQUETES

Le but de cette activité consiste à décrire les processus de traitement sous forme de requêtes en indiquant pour chacune d'elles les données auxquelles elle désire accéder. Le chemin d'accès logique se réfère aux éléments de la structure logique. Cette activité est importante, car elle permet de vérifier si la structure logique peut satisfaire aux requêtes. L'objectif de tout chemin d'accès logique consiste à:

- former des sous-ensembles permettant de satisfaire aux besoins des utilisateurs
- faciliter les opérations de mise à jour
- favoriser la réalisation des contraintes d'intégrité décrites à la phase précédente
- respecter les contraintes de confidentialité et faciliter leur réalisation
- décrire les aspects logiques influençant la structure physique
- faciliter la collecte des informations

ce qui implique l'étude et la description détaillée des requêtes, une adaptation possible de la structure logique afin que cette dernière satisfasse au mieux aux exigences qui peuvent en résulter. Cette étude doit, en outre, déterminer les paramètres qui influencent la réalisation physique, objectif de l'étape suivante.

Remarque: Un enregistrement ou tuple, que nous avons dénommé au niveau de la structure logique comme étant une catégorie non-élémentaire représentée par le symbole  , *forme un chemin d'accès logique, puisqu'il lie entre eux les catégories et relations décrivant une entité donnée. Par un accès à l'identificateur de valeur quelconque, l'on obtient les réalisa-*

tions des constituants qui en dépendent.

Il importe d'insister sur le fait que cette étude des chemins d'accès logiques doit s'effectuer indépendamment de toutes contraintes liées aux possibilités d'un SGBD quelconque. Lorsque nous aborderons, à l'étape suivante, la réalisation physique de ces chemins d'accès, la plupart des paramètres cités au chapitre suivant seront analysés en fonction des problèmes d'implantation.

8.3. PARAMETRES NECESSAIRES A LA DESCRIPTION DES REQUETES

L'élaboration d'une structure logique optimale exige la prise en compte des paramètres suivants:

a) Description du sous-ensemble des réalisations de la structure exigée par une requête

Ce sous-ensemble pourra être formé

- d'un ensemble de relations
- d'un ensemble de conditions relatives aux relations considérées ou à d'autres
- de conditions relatives à l'existence d'occurences d'une relation donnée.

La définition de ce sous-ensemble permet de vérifier si la structure logique est complète et si les contraintes d'intégrité et de confidentialité ne sont pas violées.

b) La fréquence d'utilisation des relations

En vue de la réalisation physique de la structure logique, il est nécessaire de distinguer les relations utilisées de celles qui ne le sont pas, afin de garantir une structure physique optimale à l'exploitation.

Au préalable, la définition d'une unité de mesure commu-

ne s'impose, par exemple la minute, l'heure ou le jour.

Pour chaque requête, les relations et le nombre d'utilisations de celles-ci par unité de temps devront être déterminés.

c) Le rapport entre sous-ensemble et ensemble des réalisations concernées par la requête

Le choix d'une structure physique adéquate implique la connaissance de l'éventail des valeurs et de la distribution des réalisations par valeur. Au niveau de la réalisation physique, le problème ne sera pas le même si pour un constituant donné, il n'existe que deux valeurs (le sexe: masculin et féminin) ou davantage (les noms de personne).

d) Les points d'entrée

Ceux-ci doivent être définis afin de pouvoir réaliser les accès qu'impliquent les requêtes. Les méthodes d'accès des SGBD actuellement commercialisés exigent la définition d'un constituant, point de départ de tout chemin d'accès, excepté le balayage séquentiel sans critère d'ordre.

e) Les priorités d'accès aux relations

Tout chemin d'accès implique le passage à une ou plusieurs réalisations de relations conformément à la requête formulée, selon un ordre prioritaire qui dépend du besoin formulé, et qui devra être implanté physiquement. A titre d'exemple, une application donnée peut exiger un accès à une rue donnée pour, ensuite, déterminer l'ensemble des villes dans lesquelles existent une telle rue, ou encore accéder à la rue x de la ville y, l'accès partant de ville vers rue. L'impact au niveau physique ne sera pas le même si seulement une des deux requêtes ou les deux doivent être satisfaites. Le choix des bornes du chemin d'accès

est à déterminer, car les SGBD existants ne permettent pas la réalisation automatique de ce choix, excepté le système Adabas qui est capable d'effectuer un choix partiel (voir chapitre 9.3.1.2).

f) Les critères d'ordre

Une requête peut exiger que l'ensemble des réalisations-cibles soit classé selon la valeur d'un constituant donné faisant ou non partie de ce sous ensemble-cible. Au niveau physique, cela peut se traduire par la réalisation d'un chemin physique selon l'ordre ainsi défini ou alors par la création de procédures de tri.

g) Le temps de réponse

La longueur du chemin d'accès logique (nombre de réalisations et de liaisons) influence directement les temps de réponse. Ce critère est d'autant plus critique que la requête représente un des chaînons d'un processus de gestion continu. Le non-respect de cette contrainte peut empêcher l'exécution des opérations. Citons quelques exemples pratiques:

- pour une compagnie aérienne, la confection d'une liste des enregistrements définitifs pour un vol donné
- pour une entreprise de la presse, la préparation des adresses pour l'expédition
- pour une banque, le contrôle du solde d'un compte et de la limite de crédit d'un client lors de l'encaissement d'un chèque

ne tolèrent aucun retard.

Le paramètre temps de réponse peut donc, en fonction du volume des réalisations concernées, influencer le profil du chemin d'accès logique, ceci sans tenir compte de caractéristiques d'implantation du SGBD considéré.

h) Opérations de mise à jour

Celles-ci sont considérées comme étant des requêtes. La vue partielle impliquée doit être décrite, le respect des contraintes d'intégrité et de confidentialité garanti. En outre, certains cas particuliers sont à considérer:

- une opération de modification nécessite d'abord un accès à l'entité concernée, ou à plusieurs, en vue de modifier la valeur de certains constituants ou alors de supprimer la réalisation existante pour en insérer une nouvelle
- une opération de suppression peut s'effectuer soit en modifiant le code "existe" (flag), soit en supprimant l'entité concernée, ce qui peut provoquer une chaîne de suppressions de relation
- le respect des contraintes d'intégrité nécessite souvent l'accès à diverses relations.

Le nombre d'accès nécessaires en vue d'effectuer ce genre d'opération peut croître rapidement et il est fort possible que l'introduction de redondance diminue de façon sensible le nombre des accès, sans compliquer le processus de mise à jour.

i) Le volume des opérations de mise à jour

Le problème du nombre d'opérations se pose sous trois aspects différents:

- le volume à traiter par unité de temps, ou en d'autres termes le nombre d'accès aux relations concernées
- par le rapport nombre de réalisations à mettre à jour et l'ensemble des réalisations enregistrées dans la banque de données
- le taux de mise à jour par constituant, notion qui permet de préciser le paramètre "opérations

de mise à jour" cité auparavant.

j) Le mode de traitement

De l'objectif d'une application, il est nécessaire de déduire le mode de traitement et ceci indépendamment de toutes contraintes d'implantation physique. Par exemple, dans le cas d'une mise à jour des entités 'adresse' en mode transactionnel, l'accès par le constituant "nom de adresse" peut s'avérer plus utile alors qu'un traitement par lot s'effectuera par l'intermédiaire de l'identificateur de l'entité.

k) La nature des informations demandées

Les informations que l'utilisateur désire extraire peuvent se résumer à un nombre relativement restreint de réalisations ou se ramener à un cumul de toutes les réalisations d'un ensemble de constituants, opération qui nécessite l'exécution d'une fonction de calcul. La solution du problème consiste à trouver un optimum entre le nombre d'accès exigé pour satisfaire à une telle requête, la fréquence d'exécution et les contraintes de temps de traitement. Selon les cas, il est avantageux de créer des procédures préparant ces informations avant que le besoin d'accès en soit formulé. Une partie des informations du niveau dispositif et pratiquement toutes celles du niveau stratégique entre dans cette catégorie (8.1).

l) Le rapport coût-valeur

Au niveau logique, il est possible de faire une première sélection afin d'éliminer les requêtes impliquant un nombre trop élevé d'accès ou un coût excessif en maintenance par rapport aux avantages présumés. Cette sélection nécessite une bonne dose de bon sens et un haut degré d'objectivité de la part du concepteur ou du responsable du projet.

L'étape suivante fournira, en cas de doute, une réponse plus précise, puisque l'impact du traitement de telles requêtes pourra être évalué en nombre d'accès disque, emplacement mémoire, temps d'exécution, etc ...

m) Le dynamisme du système de gestion

Tout système de gestion est par nature instable et évolue dans le temps. Ce paramètre est par trop négligé lors de la conception de système informatique, négligence qui peut provoquer de facheuses conséquences, entre autre:

- restructuration de la banque de données, opération qui n'est pas toujours facile à réaliser et selon les SGBD, d'un coût très élevé, parfois même prohibitif
- adaptation d'une grande partie des programmes d'application selon les modifications de structure provoquées.

A l'étape suivante, nous étudierons plus en détail les problèmes posés par une restructuration au niveau de la réalisation physique.

Une étude sérieuse des problèmes liés au dynamisme du système s'impose. Ce phénomène peut se présenter sous deux aspects différents:

- évolution des volumes
- apparition de nouveaux besoins en information se traduisant soit par une adjonction de nouveaux faits réels (catégories et relations), soit par la création de nouveaux chemins d'accès logiques ou les deux à la fois.

La probabilité que les nouveaux besoins et leur impact sur la structure logique deviennent réalité doit donc être évaluée. Dans certains cas, il est à conseiller d'intégrer les besoins futurs au modèle, à condition

que les chances de réalisation soient assez fortes et que l'effort de maintenance le permette, afin de minimiser les efforts d'adaptation futurs.

n) Contraintes liées à la migration du système actuel vers le nouveau système

Des raisons économiques, de sécurité de fonctionnement et de simplicité de réalisation, induiront différentes solutions:

- le SGBD est prévu pour la gestion des données d'un système déjà existant et qui ne sera pas remis en cause en ce qui concerne les applications. Pour l'utilisateur, le changement n'apparaitra pratiquement pas, au plus en ce qui concerne les mesures de reprise en cas de panne. L'effort d'adaptation dépend en premier lieu des possibilités offertes par le SGBD
- Une partie des applications sera traitée selon l'ancien système. Dans ce cas, la création d'un interface est nécessaire en vue de l'échange des informations entre les deux systèmes. Une adaptation de la structure logique peut s'imposer
- le système existant est condamné et remplacé par un nouveau, ce qui ne pose pas le problème d'adaptation de la structure logique, ni de la création d'interfaces.

Dans les deux dernières solutions ci-dessus se pose le problème de la saisie des données. Les faits réels représentés par la structure logique peuvent-ils être engendrés à partir de fichiers existants ou au contraire impliquent-ils des procédures de saisie complexes, longues à effectuer et d'un coût très élevé ? A la limite une adaptation de la structure logique peut s'avérer nécessaire. Un exemple est la classification des abonnés en groupes professionnels: d'une part il est très difficile d'obtenir des

renseignements objectifs et conformes à la réalité et d'autre part se pose le problème de l'actualisation de ces informations.

8.4. METHODES DE DESCRIPTION DES REQUETES

8.4.1. Description utilisant un langage descriptif

Comme dans le cas de la description des contraintes d'intégrité, les chemins d'accès logiques peuvent être spécifiés à l'aide d'un langage descriptif, par exemple Sequel. Le besoin en information est une vue partielle, c'est-à-dire un ensemble d'attributs d'une ou plusieurs relations soumis à certaines conditions. Aux chapitres 7.1.2 et 7.1.3 nous trouvons quelques exemples de vues partielles décrites en Sequel (8.2).

La priorité entre les relations considérées est

a) implicite à la description de la requête

Considérons les deux relations R_1 et R_2

R_1 : client (#, nom, adresse)

R_2 : commande (#, #client, date, #article, qté)

et exprimons la requête 'Quels sont les numéros d'article commandés par le client Dupont' en Sequel

```
select #article
from commande
where #client is
      select #
      from client
      where nom = 'Dupont'
```

La relation entre R_1 et R_2 représentée graphiquement étant la suivante, figure 8.1

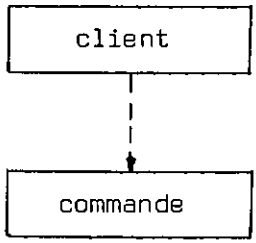


Figure 8.1 - Vue hiérarchique sur 2 relations

Cette liaison n'a rien à voir avec la notion de set du Codasyl, car elle n'est valable que pour la requête décrite et n'a pas de signification sémantique en elle-même.

b) explicite

Dans ce cas, la liaison se décrit en utilisant des opérateurs adéquats. Prenons l'exemple ci-dessus et formulons la liaison entre commande et client que nous nommerons 'vente'

```
Define net vente :
    branch client to commande
    from commande
    where #client is
        select #
        from client
```

Le point d'entrée du chemin d'accès logique est inclus dans la description de la requête. Dans l'exemple ci-dessus, il s'agit du constituant 'nom' de la relation 'client'.

Des critères d'ordre peuvent être formulés en utilisant la fonction

ordered by ... ascending / descending

Par exemple, dans le cas d'une requête exigeant pour les numéros d'article > 1000 les numéros de client et la date de commande, un classement par ordre croissant de la date, nous obtiendrons la description suivante

```
select #client, date
from commande
where #article > 1000
ordered by date asc ;
```

Les opérations de mise à jour se définissent très facilement à l'aide du langage Sequel, mais les problèmes liés à l'exécution de ces opérations, au nombre et à la complexité des accès devront être étudiés séparément. La structure logique (modèle global ou conceptuel) ne contenant que des relations en 3FN, un tuple d'une relation ne décrit donc qu'un objet, ce qui facilite les opérations de mise à jour.

Chaque opération implique un chemin d'accès logique qui devra être minimal, tout en respectant les contraintes d'intégrité, c'est-à-dire le nombre de constituants concernés par ces opérations doit être minimal. Un ensemble de relations en 3FN est caractérisé par le nombre élevé de relations, par une redondance des index et par la non-redondance des constituants ne faisant partie d'aucun index. Les opérations de mise à jour d'une relation donnée peuvent donc impliquer un accès aux relations ayant le même index. Par contre, en ce qui concerne la mise à jour de constituants non membres d'un index, les relations en 3FN sont, du point de vue chemin d'accès logique, optimales.

Un autre type de relation, de nature "événementiel", où toute réalisation n'a de sens que par rapport à un instant donné, par exemple une 'facture' ou un 'bon de commande', où toute relation représentant un fait histo-

rique est un cas particulier. L'index de telles relations est souvent un numéro d'ordre alloué chronologiquement ou une date. Les opérations de mise à jour se limitent, dans la plupart des cas, à des insertions et suppressions de tuples. La représentation d'une entité de cette nature s'effectue donc avantageusement sous la forme d'une relation en 2FN. Cette notion d'aspect temporel est partie intégrante de la méthode proposée et nous avons abordé ce problème au cours du chapitre 6.3.2 (8.3).

En plus de la description des opérations, il est nécessaire de quantifier les opérations de mise à jour, afin de pouvoir annalyser l'efficacité de la maintenance de l'ensemble des relations. Une adaptation de la structure logique peut dans certains cas s'avérer nécessaire. Le passage de relations en 3FN à des relations en 2FN est également une solution possible. A titre de démonstration, nous pouvons consulter l'exemple cité en référence (8.4).

En ce qui concerne le dynamisme du modèle, les besoins futurs peuvent être décrits à l'aide du langage Sequel.

Ce mode de description propre aux SGBD de type relationnel, par exemple le System R ou Ingres, s'imposera certainement lorsque de tels systèmes seront commercialisés et auront dépassé le stade de la recherche.

Quant aux autres paramètres décrits au chapitre précédent, ils ne peuvent être formulés avec le langage Sequel et feront donc l'objet d'une étude spécifique en fonction des objectifs définis au cours du chapitre 8.2.

8.4.2. Description paramétrique des chemins d'accès logiques

Nous proposons de prendre comme base de départ la structure logique élaborée à l'étape précédente et les requêtes qui ont mené à cette structure.

L'approche s'effectuera selon la procédure suivante:

- Pour chaque requête il faut déterminer les relations (fonctions d'accès) de la structure logique nécessaire à la satisfaction de la requête considérée. Le recensement des requêtes est l'objectif principal de l'activité décrite au chapitre 6.1, celles-ci étant fixées à l'aide d'une matrice des besoins en information. En étroite collaboration avec le concepteur de l'application informatique, l'administrateur de la banque de données confrontera cette matrice aux solutions définitives adoptées au niveau de l'application.

Chaque requête ainsi déterminée sera représentée dans une deuxième matrice que nous appellerons 'matrice des requêtes' (figure 8.3).

- L'étude de la fréquence d'utilisation des relations exige en premier lieu de fixer le taux d'activité des requêtes en fonction d'une unité de mesure commune. Les résultats seront notés dans l'en-tête de chaque colonne de la matrice (figure 8.4).

Le nombre d'accès aux relations est le résultat de la multiplication du taux d'activité par le paramètre d'existence moyen (8.5) de la fonction d'accès considérée. La valeur de ce paramètre figure dans le tableau décrivant la structure logique (figure 7.1). Lors de ce calcul, il est très important de respecter l'effet multiplicateur du paramètre d'existence. Illustrons ce phénomène en prenant comme exemple le graphe de la figure 8.2. Considérons

deux requêtes R1 dont le taux d'activité serait 1 et R2 avec un taux de 3, chacune d'elles exigeant un accès aux fonctions f1, f2 et f3. Les fréquences calculées figurent dans le tableau de la figure 8.3.

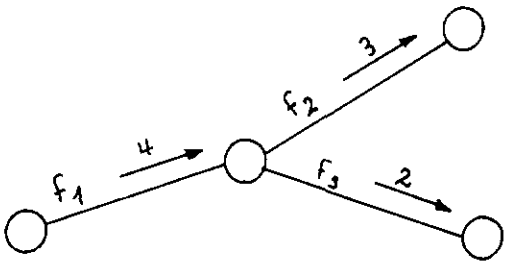


Figure 8.2 - Graphe

requêtes / fonctions d'accès	R_1	R_2	
	1	3	
f1	4	12	
f2	12	36	
f3	8	24	

Figure 8.3 - Matrice des requêtes

Le graphe de la figure 8.2 peut représenter le fait qu'un abonné commande en moyenne 4 objets différents (f1), pour chacun d'eux, il existe en moyenne 3 charges diverses à facturer (f2) et 2 factures (f3). Pour accéder à toutes les réalisations du graphe pour un abonné donné, requête R1, le nombre d'accès correspond bien au résultat de la colonne correspondante de la matrice de la figure 8.3.

- Le point d'entrée du chemin d'accès logique devra être déterminé pour chaque requête. Il correspond à l'une des fonctions d'accès recensée dans la matrice des requêtes. Pour chacune d'elles, le point d'entrée se décrit en soulignant la fonction d'accès concernée dans la colonne de la matrice (figure 8.4).

- En ce qui concerne les priorités d'accès aux relations, en d'autre terme, indiquer le chemin à suivre, l'ordre pourra être précisé en numérotant les étapes d'un chemin donné à côté du nombre d'accès de la fonction concernée. Cette indication n'est obligatoire qu'au cas où le schéma de la structure logique nécessite un tel choix (figure 8.4).
Dans le cas d'un langage du type Sequel, le chemin est implicite à la requête formulée ou est décrit explicitement. Une illustration de ces deux méthodes différentes est donnée au chapitre précédent.

- Un critère important quant à la réalisation physique est la relation entre le nombre de réalisations auxquelles il faut accéder pour répondre à une requête et le nombre total des réalisations de l'ensemble considéré.
Ce rapport sera évalué pour chaque requête et reporté dans la matrice (%) de la figure 8.4.

- Le critère d'ordre, si les réalisations demandées par la requête doivent être classées, peut être une seule fonction repérée par un encadrement dans la colonne correspondante. Si celui-ci est composé d'un ensemble de fonctions d'accès, il pourra être décrit séparément. La direction d'ordre est indiquée à l'aide d'une flèche verticale, ↓ = décroissant et ↑ = croissant.

- En ce qui concerne les contraintes de performance, celles-ci peuvent être indiquées en valeur absolue pour chaque requête ou en indiquant un ordre de priorité. Par exemple, pour des requêtes de type transactionnel, cet ordre pourra prendre la valeur 1 alors que dans le cas de requête dont les contraintes sont moins

critiques, la valeur sera supérieure (statistiques mensuelles ou sporadiques). Ce paramètre apparaît également dans l'en-tête de la colonne concernée.

- Les opérations de mise à jour sont, conformément aux explications du chapitre 8.3, considérées comme des requêtes et par conséquent intégrées à la matrice des requêtes.

- Nous représenterons le mode de traitement à l'aide du symbole  pour transactionnel et  pour le traitement par lot.

- Un autre point important est l'étude, pour toutes les fonctions d'accès, de l'éventail des valeurs possibles et de leur distribution. Par exemple, dans un système de gestion des abonnés, la caractéristique 'type d'abonnement' peut prendre 20 valeurs différentes, mais l'une d'elles apparaît dans 95 % des cas environ et les 19 autres se répartissent, selon une distribution normale, dans les 5 % restant.

Dans la plupart des cas, une telle valeur est unique, par exemple les identificateurs (clé ou index). Ce paramètre sera représenté dans la matrice par le sysmbole 'U' pour unique. Dans les autres cas, une description annexe est plus pratique et empêche un alourdissement inutile de la matrice.

- La nature des informations, les contraintes de performance ou la fréquence d'utilisation impliqueront dans certains cas la création d'entités 'statistiques' correspondant à un cumul des informations enregistrées dans la banque de données.

- A ce stade de l'analyse, sans se référer à des critères d'implantation physique, il est possible d'éliminer des requêtes dont la satisfaction exige des chemins d'accès

requêtes	n°	1	2	3	4
	fréq	100	400	1000	10
	%	0,1	0,04	0,04	0,06
	perf	1	1	1	5
	trait				

fonctions d'accès					
R1	U	100			10
R1'					
R2		100			
R2'					
R3		210			
R3'					
R4		189			
R4'					
R5		210 ↑			
R5'					
R6					
R6'					
R7					50
R7'			2000		
R8					50
R8'			4000(1)		
R9					50
R9'			2000		
R10					50
R10'				1000	
R11				1000	50
R11'					

Fig. 8.4 - Exemple partiel d'une matrice des requêtes

longs par le nombre de bornes à franchir et par le nombre d'accès à exécuter, alors que le besoin dont elles découlent est relativement peu important.

- Le besoin représenté par un chemin d'accès est-il stable ou au contraire son évolution dans le temps est-il très probable (voir chapitre 6.1) ?

Prenons à titre d'illustration les faits élémentaires partiels représentés dans le tableau descriptif de la figure 7.1 et considérons les requêtes suivantes :

R1 : par l'intermédiaire du numéro de personne, accéder au nom, à son adresse privée et au numéro de téléphone correspondant.

R2 : déterminer le numéro de distributeur distribuant dans une ville donnée dont le numéro postal est connu et dans rue de cette ville.

R3 : déterminer le numéro postal d'une ville de nom donné.

R4 : quel est le secteur de distribution d'une personne dont le nom est connu ?

La description des paramètres est représentée par la figure 8.4.

8.5. ADAPTATION DU MODELE DE LA STRUCTURE LOGIQUE

Cette activité a pour objectif d'optimaliser la structure logique élaborée lors de l'étape précédente (chapitre7) en tenant compte des contraintes découlant des chemins d'accès logiques et décrites sous forme paramétrique.

En nous basant sur notre exemple, le tableau descriptif de la figure 7.1, le schéma de la figure 7.6 et la matrice des requêtes établie au cours du chapitre précédent (figure 8.4) sont les outils à disposition dans la réalisation de cette activité. Cette dernière se caractérise par quatre opérations:

1. Minimiser le nombre de liaisons dans le schéma

Si le nombre des accès par relation sont à peu près égaux, nous regrouperons les noeuds-cible dans le noeud source. En d'autre terme, cela signifie que ces fonctions d'accès sont utilisées en même temps.

Le regroupement s'effectuera en respectant les paramètres d'existence et en tenant compte de la longueur des éléments concernés. Par exemple, une liaison entre deux catégories non élémentaires de type 1:n dont les paramètres prennent les valeurs (1, 3, 1,5) et que la longueur de l'entité concernée est grande (une adresse), les deux catégories ne seront pas regroupées. Si la longueur avaient été de 1, comme dans le cas d'un code quelconque, le regroupement s'imposeraient.

Dans le cadre de cette opération, l'algorithme cité en référence (8.6) et basé sur l'analyse des clusters peut être utilisé. Les résultats du traitement serviront de base au regroupement qui s'effectuera selon les principes décrits à l'alinéa précédent.

En général, les résultats obtenus impliquent des solutions

divergentes. Les paramètres performance ou type de traitement décrits dans la matrice peuvent alors forcer la décision.

Les fonctions d'accès utilisées comme point d'entrée des chemins d'accès logiques ne seront pas regroupées. Dans les cas où l'éventail des valeurs est très restreint, par exemple 'sexe' et 'état-civil' de la catégorie personne, le regroupement s'effectue tout de même. Si la relation concernée a été intégrée lors de la phase "conception de la structure logique", le point d'entrée sera représenté par une flèche dans le schéma.

Si un chemin d'accès logique atteint une portion relativement importante de l'ensemble des réalisations concernées, un regroupement s'impose. Le mode de traitement, les contraintes de performance et de mise à jour, ainsi que le volume des données influenceront la solution définitive.

Une certaine redondance peut être introduite, si les opérations de mise à jour n'impliquent pas des procédures trop longues et compliquées à traiter. Le taux de modification auquel sont soumises les relations concernées décidera si l'introduction de la redondance s'impose. Par exemple, la 'rue' et la 'ville' seront regroupées dans L'entité 'adresse, celle-ci étant toujours utilisée comme un tout. D'autre part, des entités 'ville' et 'rue' avec leurs caractéristiques respectives seront générées séparément. Cette solution est défendable, car le nom d'une ville ou d'une rue ne change pratiquement jamais.

2. Influences de la migration vers le nouveau système

Une analyse de la structure des fichiers existants et les faits réels qu'ils représentent est inévitable. Une comparaison avec la description du modèle global permettra ensuite de déterminer:

- la concordance entre les deux modèles
- les règles de passage au nouveau modèle
- si une adaptation du nouveau modèle s'impose.

Le problème de la migration se pose également en fonction de la réalisation du système dans le temps. Dans le cas de grand projet, la réalisation s'effectue en général par étape, il faut donc délimiter celles-ci afin que l'implantation du nouveau système n'exige des modifications trop importantes du système existant. Le découpage devra se faire de manière à minimiser les interrelations entre les éléments des différentes étapes. D'autre part, il est nécessaire:

- de réserver les emplacements nécessaires au niveau de la réalisation physique
- de prévoir la possibilité de réaliser de nouvelles associations dans le futur
- de prévoir les modifications à apporter aux programmes d'application.

3. Dynamisme du système de gestion

Après avoir évalué les besoins futurs et leurs chances respectives de réalisation, ceux-ci seront introduits dans la matrice des requêtes. L'analyse de différentes alternatives permettra de détecter les effets sur le schéma global. Cette méthode tient compte de:

- besoins nouveaux
- l'évolution des volumes
- modifications de tout paramètre nécessaire à la conception
- l'évolution des besoins existants, une attention toute particulière doit être vouée aux informations exigées par le processus décisionnel (chapitre 6.1).

4. Vérifier les contraintes d'intégrité

Ne disposant pas, contrairement à la méthode décrite au chapitre 8.4.1 d'un langage approprié et d'un système vérifiant si toutes manipulations de données respectent les contraintes d'intégrité, il est nécessaire de vérifier si toutes les opérations de mise à jour prévues par les différentes requêtes ne détruisent pas l'intégrité de la banque de données.

L'administrateur de la banque de données effectuera ce contrôle en se basant sur les tableaux descriptifs élaborés (figures 7.1 et 7.2). Il analysera les manipulations prévues en fonction du modèle de données adapté. Cette opération est vitale, car l'introduction de redondances rend le maintien de l'intégrité de la banque de données plus difficile. Référons-nous au cas spécial illustré par la figure 7.14 pour confirmer l'importance de cette activité.

L'introduction de redondance, comme dans le cas de la catégorie 'adresse' de l'exemple cité au point 1 de ce chapitre, implique la création de procédures spéciales de mise à jour. En effet, si le nom d'une rue donnée est modifié, toutes les adresses contenant ce nom devront être identifiées afin de pouvoir effectuer les modifications nécessaires.

Le schéma de la figure 7.6 pourra, suite aux opérations décrites ci-dessus, se présenter sous la forme illustrée par la figure 8.5.

Remarque: L'affirmation ci-dessus laisse la voie ouverte à plusieurs solutions car, une évaluation objective des paramètres n'est pas toujours possible. L'efficacité de l'administrateur de la banque des données conditionne le résultat de l'analyse.

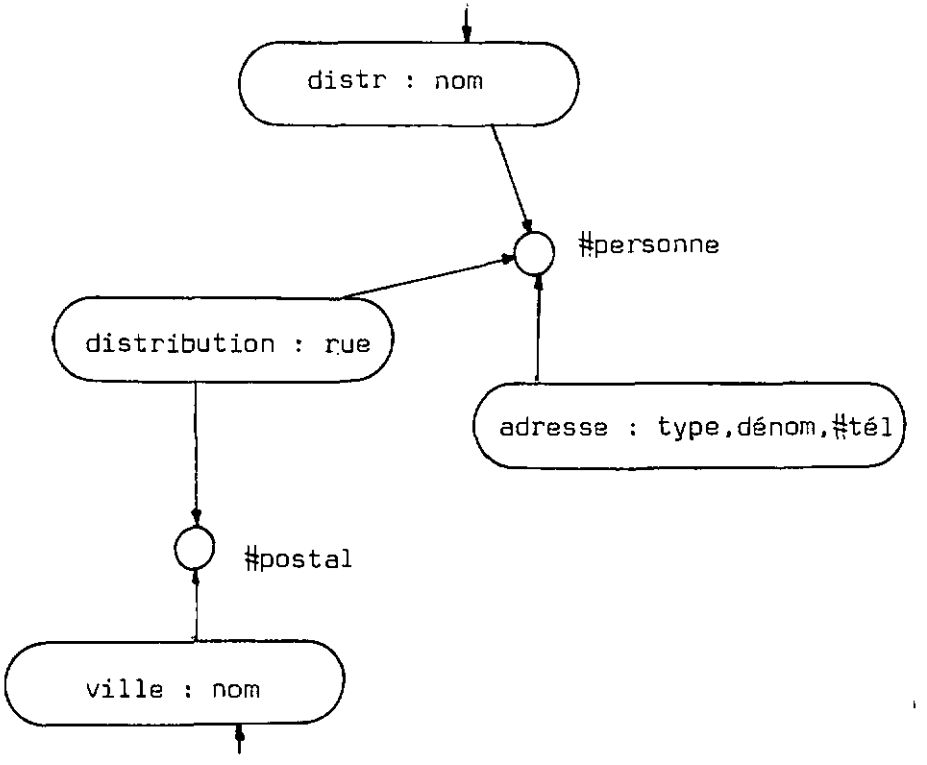


Figure 8.5 - Graphe de la structure logique

Pour la première fois dès le début de notre méthode d'approche, nous avons étudié le modèle global sous l'aspect de son efficacité en fonction de l'exécution des requêtes. L'un des principes fondamental de la méthode, l'indépendance face aux critères d'implantation est respecté puisque la description des chemins d'accès logiques et l'optimalisation de la structure logique ne font références à aucun SGBD et s'adaptent autant à un système hiérarchique ou réseau qu'à un système de type relationnel.

Ce n'est qu'à l'étape suivante que les critères de réalisation physique apparaîtront dans le raisonnement. Les paramètres décrits au cours de la phase décrite dans ce chapitre 8 seront confrontés aux possibilités du logiciel de gestion des données choisi.

REFERENCES

[8.1] Ces notions sont analysées au chapitre 3.1.1, les problèmes liés aux systèmes de décision étant décrits au chapitre 6.1.

[8.2] BOYCE R.F., CHAMBERLIN D.D., "Using a structured query language".

CHAMBERLIN D.D., BOYCE R.F., "Sequel : a structured ...".

[8.3] En ce qui concerne le modèle relationnel, consultez WEDEKIND H., "Normalization of information ...".

[8.4] WEDEKIND H., "Datenbanksysteme I" p. 86 ss.

[8.5] Cette notion a été définie au chapitre 6.3.4.

[8.6] Entre autre celui proposé par GILLNER R. dans "Die Strukturierung ...".

Chapitre 9

Les Chemins d'accès physiques

9.1. CHOIX DU SGBD

9.1.1. Objectifs d'une procédure de choix

Le but de cette procédure ne consiste pas à rechercher des arguments en faveur d'un SGBD, mais à étudier les possibilités d'un tel système en fonction des exigences et des avantages économiques qu'il peut apporter.

Nous déterminerons les critères d'évaluation à partir des résultats obtenus lors de la phase de modélisation et complétés par l'étude plus détaillée des caractéristiques des applications selon le canevas des trois premières étapes de notre méthode d'approche. Il suffit de ne considérer que les traitements principaux, ceux dont les contraintes sont les plus restrictives et les plus exigeantes (performance, volumes à traiter, type de relation, sécurité, etc).

La procédure de choix peut débuter avant d'avoir effectué complètement les trois premières étapes.

Après une première présélection, les systèmes seront évalués en fonction des critères fixés.

Avant la prise de décision, des tests effectués à l'aide d'un modèle réduit permettent de vérifier les comportements des SGBD que l'évaluation présentent comme les plus aptes à satisfaire aux objectifs fixés.

La figure 9.1 illustre le déroulement de cette procédure.

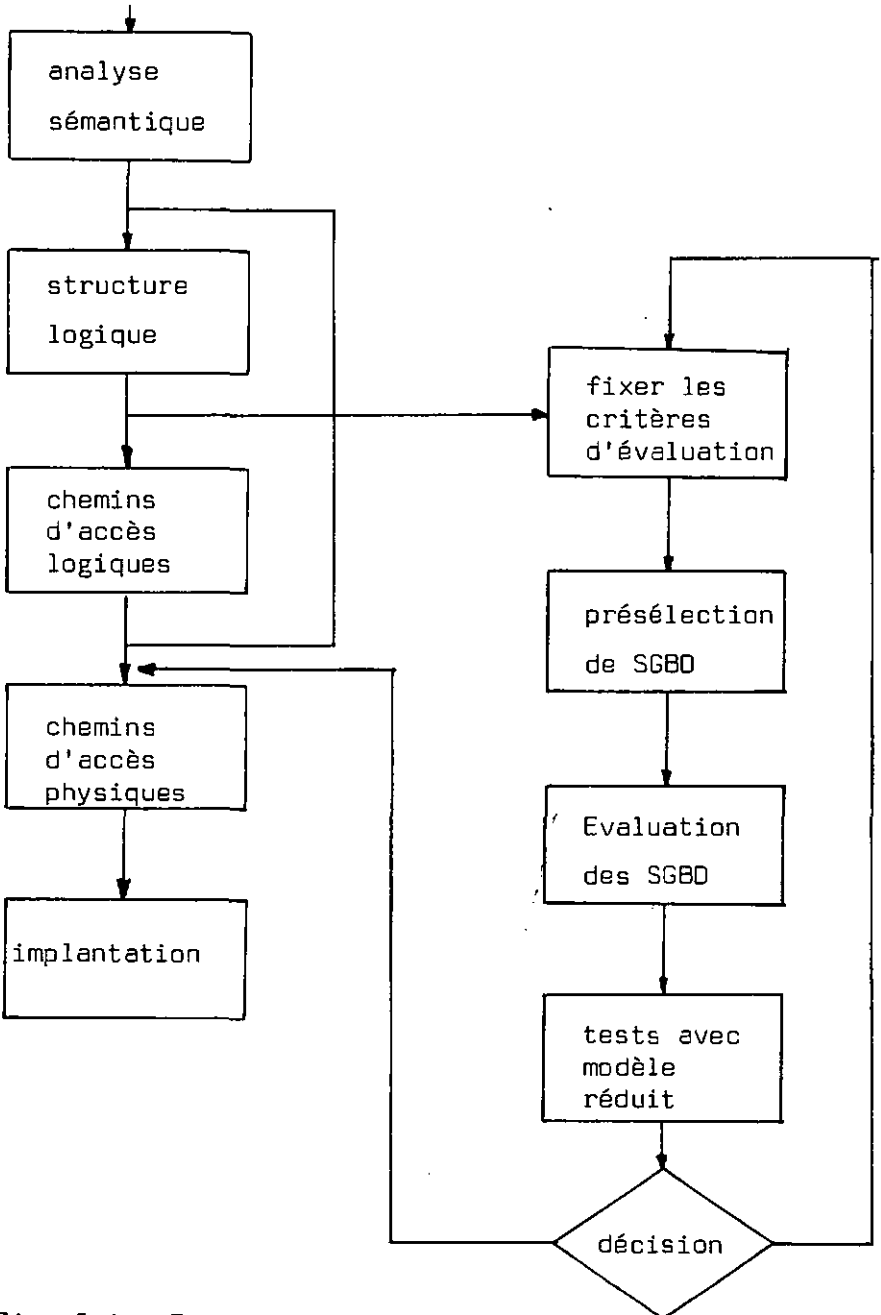


Fig. 9.1 - Etapes de la procédure de choix d'un SGBD

9.1.2. Critères d'évaluation et première sélection

Les critères se classent en général en deux catégories principales (9.1):

- critères quantitatifs
- critères qualitatifs.

La liste énumérées ci-dessous ne prétend en aucun cas être exhaustive (9.2).

a) Critères quantitatifs

- coûts d'installation du logiciel
- coûts de formation
- coûts du logiciel (SGBD, moniteur de télétraitement, interfaces, utilitaires)
- coûts en mémoire centrale et périphérique
- coûts supplémentaires exigés par l'entretien du système
- coûts de réorganisation.

Remarque: Les facteurs de coûts énumérés ci-dessus ne représentent qu'une partie, la plus importante et la partie décisive comprend les coûts liés à la réalisation et à l'exploitation des applications. Leur évaluation est plus délicate, car il faut tenir compte de critères qualitatifs.

b) Critères qualitatifs

Distinguons les critères dépendant directement des applications (b1) et dont la réalisation est prévue avec une organisation banque de données, de ceux de nature plus générale (b2):

b1 - structure des données à traiter
- type de traitement (temps réel, batch ou mixte)
- mises à jour
- contraintes de sécurité

- contraintes d'intégrité (partage des ressources)
- contraintes de performance
- caractéristiques des chemins d'accès logique
- langage de programmation
- langage d'interrogation non procédural
- nature des informations exigées en sortie
- problèmes liés au chargement de la banque de données
- réorganisation des données ou seulement des chemins d'accès
- emplacement en mémoires périphériques
- adaptation des programmes existants
- évolution de l'application.

b2
- indépendance par rapport au matériel
- indépendance par rapport au système d'exploitation
- moniteur de télétraitement dépendant, si oui conforme aux exigences ?
- stabilité du SGBD
- programmes utilitaires à disposition et leur qualité
  - . chargement de la base
  - . procédure de reprise (logging, recovery, restart)
  - . adaptation de la structure
  - . création de nouveaux chemins d'accès
  - . réorganisation
  - . statistique pour l'optimisation ("tuning")
  - . facilités de test pour le programmeur
- réalisation physique des chemins d'accès (pointeur, inversion, indexation)
- modèle de données au niveau logique
- enregistrements de longueur fixe ou variable
- compression des données
- capacité en mémoire centrale nécessaire à une exploitation réaliste
- le code est-il réentrant
- 'dictionary', 'directory'
- facilité d'utilisation, efforts au niveau de la programmation

- taux d'utilisation du système (unité centrale, canaux d'entrées-sorties)
- documentation
- qualité de l'aide fournie par le distributeur
- les développements et adaptations futurs sont-ils garantis ?
- limites maximales du volume de la banque de données
- problèmes liés à la génération du logiciel SGBD
- flexibilité du système (structure, évolution des volumes)
- performance
- facilités d'exploitation
- compatibilité avec le matériel et logiciel à disposition
- puissance du matériel à disposition.

Le recensement des SGBD existant sur le marché et l'obtention de renseignements suffisants pour effectuer une première sélection ne posent pas de problèmes particuliers, il suffit de contacter le constructeur du matériel utilisé et de consulter la littérature spécialisée (9.3).

9.1.3. Evaluation des SGBD présélectionnés

9.1.3.1. Méthode d'évaluation

Le choix d'un SGBD a des conséquences non négligeables sur les coûts, ceux-ci peuvent varier fortement d'un produit à un autre. D'autre part, les investissements effectués dans la réalisation des applications, avant de passer à l'exploitation effective, prennent des dimensions qui rendent le choix irréversible. Par conséquent, la prise de décision doit résulter d'un processus de choix aussi objectif que possible.

Une méthode (figure 9.2) respectant au mieux les exigences

formulées, consiste, une fois les critères d'évaluation déterminés, à établir une structure de préférence en fonction des contraintes découlant des applications à réaliser et des moyens à disposition. Afin d'améliorer le degré d'objectivité, plusieurs personnes doivent effectuer cette classification. Il est préférable de grouper les critères et de leur affecter une valeur relative (taux de préférence) à un entier quelconque, égal à la somme des valeurs relatives, pour établir ensuite les préférences à l'intérieur des groupes de critères. Les résultats de ces classifications seront comparés et analysés en vue d'établir une structure de préférence unique (9.4)

Chaque logiciel présélectionné sera qualifié afin de pouvoir effectuer une comparaison entre les différentes alternatives proposées, ce qui implique la définition d'une échelle de valeur, par exemple 1 à 5 ou 1 à 10.

Pour chaque critère, les logiciels seront comparés entre eux et classés sur l'échelle des valeurs. Cette activité, une des plus délicates de tout le processus de choix, implique le respect des conditions énumérées au chapitre suivant, afin de garantir des résultats réalistes. Ceux-ci seront transférés dans une matrice (figure 9.3).

Ensuite, chaque note adjugée à un SGBD sera multipliée par le taux de préférence. Les résultats ainsi obtenus pour chaque SGBD seront comparés en vue d'une première sélection.

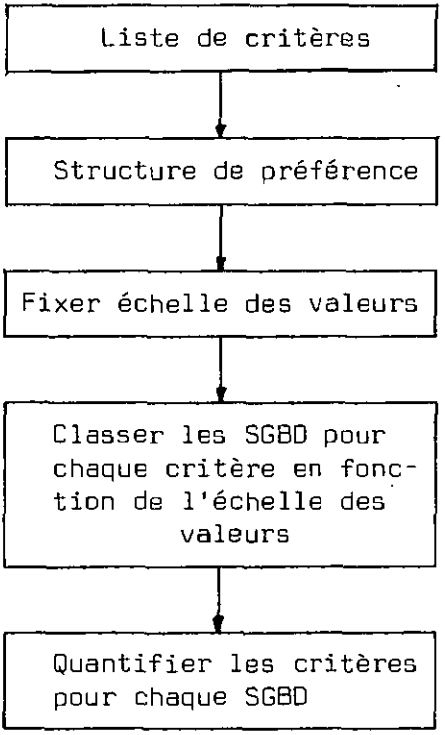


Figure 9.2 - Méthode d'évaluation

Critères	Taux de préfér.	Note max.	Note SGBD		
			S_1	S_2	S_n
C a					
C a_1					
⋮					
C a_n					
C b					
C b_1					
⋮					
C b_n					
Σ c					

Figure 9.3 - Matrice d'évaluation

9.1.3.2. Problèmes liés à l'évaluation

La ou les personnes chargées d'évaluer les SGBD présélectionnés devront résoudre un certain nombre de problèmes, afin de garantir des résultats conformes aux buts fixés.

a) Connaissance des objectifs et du concept banque de données

Une évaluation objective nécessite des connaissances approfondies en banque de données, sinon un constructeur n'aura pas trop de difficulté à imposer son produit.

Ces connaissances s'acquièrent en étudiant la littérature de base en la matière et en suivant des séminaires, si possible non animé par un constructeur. Un spécialiste en la matière, indépendant, pourra donner des conseils, ce qui permettra d'augmenter le degré d'objectivité des résultats.

b) Aptitude du SGBD à satisfaire aux objectifs fixés

Une opération délicate consiste, une fois les objectifs étudiés, à analyser la manière dont ceux-ci sont respectés par les produits à évaluer. Les études comparatives effectuées par certains instituts spécialisés (9.5) facilitent l'étude des caractéristiques des SGBD considérés.

Nous nous limiterons ici à ne citer que les points critiques, leurs influences au niveau de la réalisation étant analysées au cours des chapitres suivants:

- caractéristiques du modèle de données
- réalisation des chemins d'accès logiques (pointeurs, indexation, listes inversées)
- méthodes d'organisation des données
- intégrité des données
- accès multiples
- redondance
- indépendance des données
- sécurité d'exploitation.

c) Etude de la documentation

Il n'est pas exclu que celle-ci contienne des faits n'ayant pas dépassés le stade de projet. Pour éviter de telles mésaventures, il faut:

- demander des références et se renseigner auprès des utilisateurs cités
- effectuer une installation-test et faire des essais.

d) Contacter des utilisateurs du SGBD

Toute collecte d'informations fructueuse implique la préparation d'un plan de déroulement comprenant l'élaboration d'une liste de questions préparée à l'avance dont l'objectif consiste à vérifier l'évaluation et à la rendre plus objective. Il est préférable de sélectionner des utilisateurs dont l'environnement est semblable (grandeur de la machine, volume des données et des requêtes, structure de données de même type, etc).

Certains utilisateurs, avant de prendre une décision, étudient de manière approfondie plus d'un logiciel et effectuent des "benchmark". Il est judicieux d'analyser les résultats obtenus et les conditions dans lesquelles ces études ont été réalisées.

e) Performances du SGBD

L'évaluation des performances d'un SGBD donné est très délicate, car les facteurs qui l'influencent sont nombreux. Une méthode relativement simple à réaliser consiste à créer un modèle réduit représentatif des applications considérées.

Un tel test permettra d'obtenir des renseignements quant aux

- performances
- facilité d'utilisation (programmation, chargement)
- complexité du SGBD
- capacité mémoire nécessaire
- précautions à prendre lors de la conception
- comparaisons entre SGBD
- fiabilité du système.

Ces tests, couramment appelés "benchmark" (9.6), donnent

des résultats assez proches de la réalité. Mais il ne faut pas oublier que le comportement du système varie avec le nombre des utilisateurs et la charge du système.

Les coûts qu'impliquent de tels tests ne sont pas négligeables, mais relativement faibles en comparaison des coûts supplémentaires provoqués par un mauvais choix. Ils offrent en plus les avantages suivants:

- ils constituent une première expérience
- ils permettent de faire ressortir un certain nombre d'erreurs qui n'ont pas de conséquences à ce stade
- ils donnent une bonne connaissance du système en un laps de temps relativement court.

9.2. CARACTERISTIQUES DU SGBD CHOISI

Une fois le SGBD choisi, il faut étudier en détail ses caractéristiques et en déduire les possibilités qu'il offre au niveau de la conception.

La structure de la banque de données s'élabore en fonction des carctéristiques du système et des contraintes découlant des applications à réaliser. Les paramètres en ont été fixés au cours des trois premières phases de la méthode d'approche proposée.

Au cours des chapitres suivants, nous aborderons les problèmes de conception en fonction de SGBD représentant chacun une des grandes tendances dans le domaine des banques de données:

1. système hiérarchique: IMS
2. système réseau de type Codasyl: IDMS
3. système relationnel
   comme il n'existe encore aucun système commercialisé et assez performant pour satisfaire aux

contraintes de la plupart des systèmes informatiques de gestion, nous considérerons dans notre étude un système quasi-relationnel (9.7): Adabas.

Analyser et décrire les caractéristiques de ces SGBD déborderaient le cadre de notre étude. Au cours du chapitre 2, nous avons abordé certains aspects de ces systèmes et cité un certain nombre de références.

9.3. DESCRIPTION DE LA STRUCTURE LOGIQUE DES DONNEES

9.3.1. Conception de la structure logique

La première opération consiste à passer de la structure logique élaborée au cours de la première et deuxième phases de la méthode d'approche proposée (chapitre 6 et 7) au modèle de données pouvant être décrit avec le LDD du SGBD choisi (figure 9.4).

Les méthodes proposées facilitent un tel passage, la preuve en est démontrée au chapitre 7.3.

Il s'agit de confronter le modèle des données et le profil des chemins d'accès logiques aux caractéristiques du SGBD. Pour la première fois au cours de notre approche, nous prenons en compte les aspects de la structure physique (9.8) Une solution optimale ne peut résulter que d'un processus itératif, car le nombre de contraintes à respecter tant du côté des applications que du SGBD est relativement élevé.

Remarque: Il existe des algorithmes ayant comme objectif la conception de la structure logique en fonction d'un SGBD donné. Dans le cas de IMS (9.9), les vues hiérarchiques des utilisateurs doivent être complétées par des critères relatifs aux caractéristiques du SGBD. En ce qui concerne les systèmes de type Codasyl, les résultats

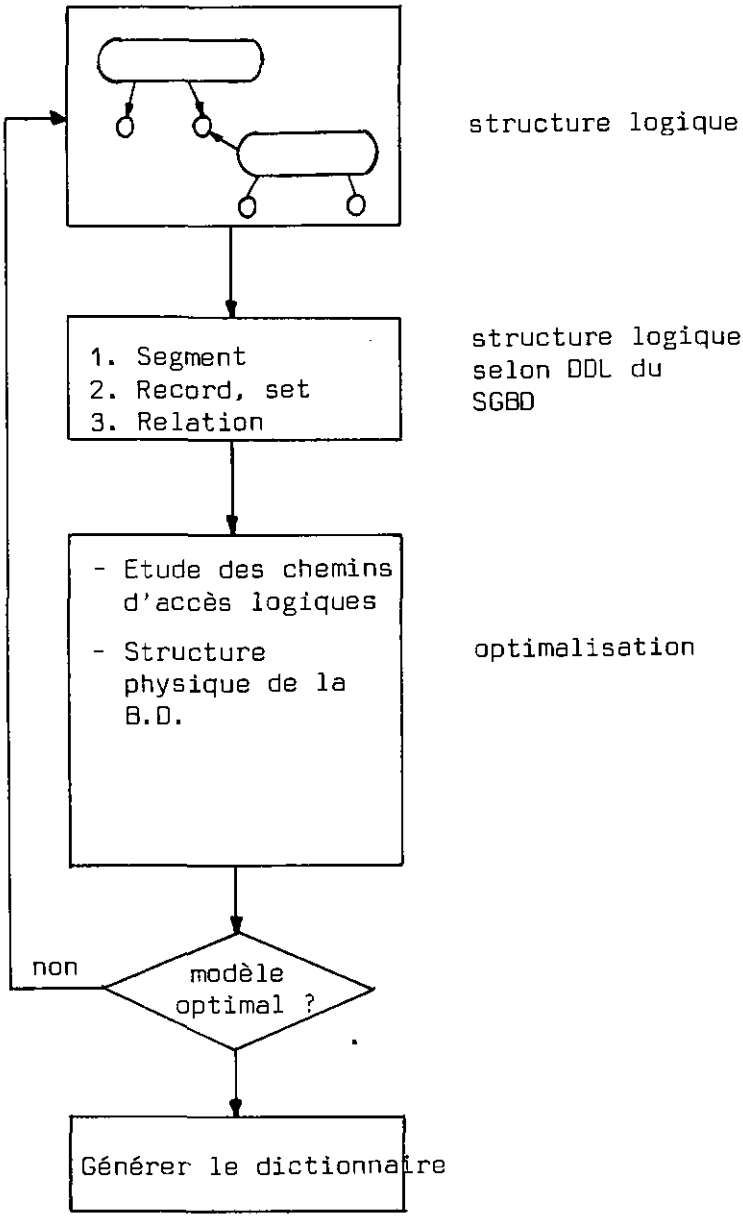


Fig. 9.4 - Description du modèle à l'aide du DDL

obtenus ne sont que partiels, tous les critères à considérés ne pouvant être intégrés à ces algorithmes (9.10).

9.3.1.1. Passage de la structure logique au LDD du SGBD

Le principal critère à respecter est le type de relation associant les différentes catégories entre elles. Le graphe de la structure logique, complété par le tableau descriptif de ce graphe, permet de déterminer la nature des relations.

Dans le cas de IMS, la structure logique est hiérarchique (relation de type 1:n). Une structure de type réseau se réalise à l'aide d'arborescences reliées entre elles.

Les catégories du graphe correspondent à la notion de segment. Les catégories identificateurs seront transférées dans les deux segments qu'elles relient, ceci à l'encontre du principe que la liaison hiérarchique entre deux segments est support d'information et qui permettrait d'éliminer la redondance que nous proposons d'introduire. Les raisons de ce double emploi sont les suivantes:

- possibilité de reconstruire les liens entre segments en cas de destruction des pointeurs
- contrôle d'intégrité - le segment enregistré dans une chaîne dépend-il effectivement du segment père ?
- aide à la programmation - facilite la lecture de "dump" lors de la vérification des tests de fonctionnement des programmes ou lors de la détection d'erreurs
- les segments sont conformes à la définition d'une relation du modèle relationnel, ce qui facilitera le passage à un système de type relationnel.

Le modèle ainsi représenté n'est pas définitif car, comme

nous l'avons signalé à diverses reprises (9.11), il implique la description des chemins d'accès. Il faudra donc les analyser et adapter le modèle en conséquence.

Les mêmes remarques sont valables en ce qui concerne le système IDMS (Codasyl), à la différence que ce dernier soit de type réseau.

Le passage au système Adabas s'effectue selon les règles de transformation valables pour le modèle relationnel, au départ nous avons des relations en 3FN. Pour chaque champ il faut définir les paramètres dictant les mesures de compression des données à prendre par le système. Le choix (FI = pas de compression, NU = pas d'enregistrement physique lorsque le champ est vide) dépend:

- des paramètres d'existence recensés dans le tableau descriptif du graphe logique (figure 7.1)
- de la longueur de la catégorie également indiquée dans un tableau.

Cette procédure de compression permet de regrouper des catégories élémentaires, ceci indépendamment des fréquences d'accès. Les paramètres d'existence et le type de la relation considérée déterminent si un tel regroupement s'impose. Cette possibilité ne procure des avantages par rapport aux deux autres systèmes que si les conditions suivantes sont remplies:

- paramètre d'existence minimum égal à 0
- paramètre d'existence maximum égal à 1 ou 2
- la valeur du paramètre d'existence moyen et la longueur de l'occurence de la catégorie doivent être telles que l'emplacement exigé par les pointeurs reliant les deux catégories puisse être économisé. Ce qui est par exemple le cas de la catégorie 'complément d'adresse' d'une longueur de 30 octets et avec les paramètres d'existence (0, 2, 0,1)

- volume des données: plus il est important et plus la compression est avantageuse.

La possibilité d'introduire des champs multiples (multiple field) dans une relation autorise l'intégration de catégories élémentaires reliées par une liaison de type 1:n à d'autres catégories, à condition que:

- les mises à jour ne soient une suite de suppressions et d'insertions
- le taux de fréquence des modifications ne soit pas trop élevé
- le paramètre d'existence maximum ne soit supérieur à 191.

Les associations entre relations (1:n, n:m) s'effectuent à l'aide de descripteurs, points d'entrée dans la banque de données. Le fait de définir un descripteur ne donne aucun indice quant au type de relation que celui-ci est censé décrire, mais permet à l'utilisateur de formuler des associations dans ses requêtes.

Un descripteur donné exprime implicitement une liaison de type 1:n entre la catégorie définie comme descripteur et la relation à laquelle elle appartient. Par exemple, il est possible de décrire à l'aide d'une primitive d'accès programmée en Cobol ou exprimée à l'aide du langage Adascript, une liaison de type 1:n entre un abonné dont le descripteur 'no abo' est égal à 'X' et les produits auxquels il est abonné.

Quant à la réalisation de relations de type n:m, nous pouvons les représenter à l'aide de relations en 3FN. Le taux de fréquence des mises à jour sera décisif quant aux possibilités de réaliser ces relations de cette façon. La décision définitive sera prise à l'étape suivante.

L'option couplage entre deux relations selon un champ

descripteur commun permet de formuler à l'aide d'une seule primitive d'accès des vues partielles se référant à plus d'une relation. Décrivons une telle requête à l'aide du langage Adascript:

```
Find in file ville with no postal from 8000 thru 8050
   and coupled to file rue with id-rue = 'AA100'
   and sort them by no postal
   and display no postal, 5X, nomrue
(5X = 5 espaces vides).
```

9.3.1.2. Analyse des chemins d'accès logiques et des structures physiques du SGBD

Le modèle élaboré à l'étape précédente représente la description des éléments nécessaires à la satisfaction des besoins en information des différents utilisateurs. La réalisation physique du modèle implique le respect d'un certain nombre de contraintes liées d'une part au profil des chemins d'accès logiques et d'autre part aux possibilités techniques dont disposent le SGBD. L'objectif de cette activité consiste donc à choisir parmi les possibilités de représenter physiquement la structure logique celles qui permettent d'accéder efficacement aux sous-ensembles de données. Les critères à considérer pour évaluer le bien-fondé d'un tel choix sont:

- les performances
- l'emplacement mémoire nécessaire
- les problèmes liés à la mise à jour.

Avec le système Adabas, les chemins d'accès sont séparés des données (fichiers inversés). Il répond donc aux conditions requises par l'indépendance des données, alors que les systèmes réalisant les chemins d'accès en ajoutant les pointeurs aux données provoquent une forte dépendance physique entre les chemins d'accès et les

données, limitant ainsi fortement les libertés dans la formulation des requêtes.

a) Performance

L'efficacité des accès à un sous-ensemble d'enregistrements d'une banque de données et par conséquent le choix du chemin d'accès dépend des facteurs suivants :

- la structure de l'enregistrement physique
- les méthodes d'enregistrement
- le mode de traitement
- le mode d'accès
- les caractéristiques des supports de données
- le mode d'utilisation, simultané ou non
- le volume de la banque de données
- la hiérarchie des mémoires
- l'architecture du système.

L'estimation des performances est relativement simple en ce qui concerne les accès par les clés primaires (9.12), alors que dans le cas des clés secondaires l'évaluation dépend de la distribution des valeurs de la clé, de la méthode d'accès et de la répartition des données sur les supports physiques (facteur de blocage, zone de débordements). Le nombre de facteurs étant très élevé, l'estimation en est d'autant plus difficile (9.13).

La séparation, au niveau physique, entre les chemins d'accès et les données, permet de sélectionner le sous-ensemble des données, répondant aux conditions posées par une requête avant d'accéder aux données elles-mêmes. L'avantage du point de vue performance est évident, la preuve en est donnée en (9.14).

D'autre part, le volume de données transférées vers l'unité centrale est nettement inférieur, les performances en sont améliorées d'autant. Ce n'est pas le programmeur, mais le SGBD lui-même qui détermine le

chemin à suivre. Le programmeur n'a donc pas d'influence sur l'accès aux supports de données.

La difficulté majeure est l'évaluation des performances de l'exécution de requête faisant appel à plus d'un point d'entrée. Une étude détaillée est présentée en (9.15), tandis qu'en (9.16) est analysée l'efficacité des différentes méthodes d'accès à une structure à fichiers inversés.

La limite de ces méthodes analytiques, du fait du nombre très élevé de paramètres et de leurs interdépendances, incite à passer à la simulation en utilisant un système spécialement créé pour l'étude du comportement d'une banque de données, par exemple Forem-Phase II (9.17). Cette technique de simulation permet de mesurer l'influence d'accès simultanés.

Une autre possibilité d'évaluation des performances consiste à réaliser un modèle représentatif de l'application considérée, en utilisant le SGBD choisi, tout en veillant à créer un environnement aussi réel que possible (volume des données, allocation mémoire, utilisation simultanée, etc). Nous avons vu que, lors de la procédure de choix d'un SGBD (chapitre 9.1), il était avantageux de faire des tests à l'aide d'un modèle qui pourra ensuite être complété afin de servir d'aide à la conception. La complexité et les utilitaires du SGBD (langage d'interrogation interactif et séquentiel, chargement de données, modification de structure, etc) seront déterminants quant à l'utilisation de cette méthode.

Une première analyse consiste à distinguer les chemins d'accès selon le type des points d'entrée auxquels ils font appel. Il est possible d'accéder d'une part à une clé primaire, à toute valeur d'une telle clé ne correspond qu'une seule occurrence d'une relation, d'autre part à un autre attribut (champ) de la relation appelé clé secondaire. Très souvent, les requêtes se

réfèrent à une combinaison de ces deux types de clé.

La relation du sous-ensemble de données exigées par une requête par rapport à l'ensemble des données enregistrées dans la banque de données, détermine le mode d'accès, direct ou séquentiel. La matrice des requêtes servira de support pour la prise de décision.

Le type et les caractéristiques des différents modes d'accès impliquent directement la structure d'enregistrement. De la description des chemins d'accès logiques (chapitre 8), nous pouvons en déduire qu'une requête qualifie les enregistrements non en fonction de leur positionnement dans la structure physique, mais selon leurs contenus. La séparation des chemins d'accès des données brutes au niveau physique est une structure qui permettra de déterminer les sous-ensembles avant d'accéder aux données, le système peut donc choisir le chemin d'accès optimal. L'avantage d'une telle séparation est la possibilité de définir des accès par l'intermédiaire d'une combinaison de clés primaires et secondaires, le système étant ensuite capable de déterminer les enregistrements répondant à la demande d'accès, ce qui évite le balayage séquentiel des secteurs de la banque de données concernés.

Dans le cas de IMS et d'IDMS, les chemins d'accès sont enregistrés avec les données, selon différents types de pointeurs. Seul le système Adabas sépare complètement les données des chemins d'accès, ces derniers étant réalisés grâce aux fichiers inversés.

De manière générale, un accès à un enregistrement donné par l'intermédiaire de la clé primaire est plus rapide dans le cas d'une structure HDAM (IMS) ou 'calc' (IDMS) que passer par une table comme dans le cas des structures de fichiers inversés (Adabas). En ce qui concerne un accès combiné de clés et un accès à des éléments de la hiérarchie ou d'un réseau qui n'ont pas de points d'entrée directs, les performances sont nettement meil-

leures avec Adabas, le numéro interne des enregistrements concernés étant déterminé par comparaison de tables. Cette comparaison permet d'optimaliser l'accès en choisissant, en fonction du nombre d'enregistrements du sous-ensemble déterminé par rapport au total des enregistrements des secteurs concernés, le mode de traitement direct ou séquentiel. Le volume des données à transférer est nettement inférieur en comparason à des systèmes où la détermination du sous-ensemble se fait en traitant un enregistrement après l'autre.

Il arrive très souvent que les exigences de deux requêtes soient opposées, la structure physique ne pouvant répondre de manière satisfaisante qu'aux contraintes de l'une des deux requêtes. Illustrons le problème à l'aide d'un cas concret tiré d'une application de gestion d'abonnements. D'un côté l'accès direct aux adresses des abonnés par l'intermédiaire de leur nom, afin de répondre aux questions posées et d'effectuer les mises à jour et de l'autre les contraintes d'expédition qui exigent un accès aux abonnés dans l'ordre des séquences d'expédition. Dans les deux cas, les temps de traitement doivent être le plus faible possible. Pour résoudre ce problème de manière satisfaisante, nous avons introduit de la redondance en enregistrant les informations nécessaires à l'expédition (adresses, nbres d'exemplaires) sur un fichier séquentiel sur bande selon la séquence d'expédition, les raisons de ce choix étant:

- faible fréquence de mise à jour, en moyenne 1 % des enregistrements sont concernés
- seule une partie des données est redondante
- le problème de la sécurité est résolu par la génération des fichiers, en cas de panne pas de problèmes de reprise
- Un taux de blocage très élevé et un accès séquentiel au fichier expédition permettent des temps de traitements excellents.

Pour des raisons de performance, l'accès aux données doit s'effectuer en transférant le minimum d'enregistrements des mémoires périphériques vers la zone de travail en mémoire centrale. En analysant les fréquences d'accès répertoriées dans la matrice des requêtes, certaines relations entre catégories peuvent se réaliser par une représentation redondante qui se traduit au niveau physique par un enregistrement contigü, un accès à un seul enregistrement suffisant alors à satisfaire la requête. Un tel regroupement n'est avantageux que si la fréquence de mise à jour des catégories concernées est relativement faible. Par exemple, dans le cas de la catégorie 'distr' (figure 8.5) la catégorie non-élémentaire 'adresse' peut être regroupée dans la catégorie 'distr', la fréquence de mise à jour de ces deux catégories étant peu importante.

Le choix de la méthode d'organisation, dans le cas de IMS et d'IDMS, détermine les possibilités d'accès et les temps nécessaires aux transferts des données.

Si l'accès peut se faire directement, le rapport nombre d'occurences exigées par rapport au nombre total contenu dans la banque de données doit être faible. A partir du point d'entrée de la requête concernée, celui-ci devra être défini comme clé afin d'obtenir les informations voulues selon une technique d'adressage direct (HDAM ou CALC) ou indirect (HIDAM ou HISAM). Si les données en sortie doivent être présentées dans un certain ordre, l'option 'sort by' devra compléter la définition du chaînage des éléments concernés dans la description du modèle. Par contre, dans le cas de Adabas, cette option n'apparaît pas au niveau du modèle, elle se définit lors de la formulation de la requête et seul un descripteur peut être utilisé comme critère de tri. Cette possibilité implique la prudence, car son utilisation exige des temps de traitements relativement importants. Si le nombre d'occurences pour une valeur donnée du descripteur concerné est faible, elle s'avère efficace.

Analysons un autre type d'accès séquentiel, celui qui doit s'effectuer sur un élément donné de la structure (segment, record ou relation) selon l'ordre logique de la clé principale. Avec IMS, si l'on veut éviter un balayage séquentiel suivi d'un tri, le segment considéré doit être le segment racine ("root segment") de l'arborescence ou d'un "data set group" et avoir comme organisation HISAM ou HIDAM (voir figure 9.5). Avec IDMS, un accès de ce genre n'est possible qu'en effectuant une procédure de tri lors de l'extraction des données. Dans le cas de Adabas, l'opération s'effectue à l'aide d'une primitive d'accès spécifique ("logical sequential read") suivant l'ordre logique de la table d'adresse du champ inversé concerné (figure 9.6).

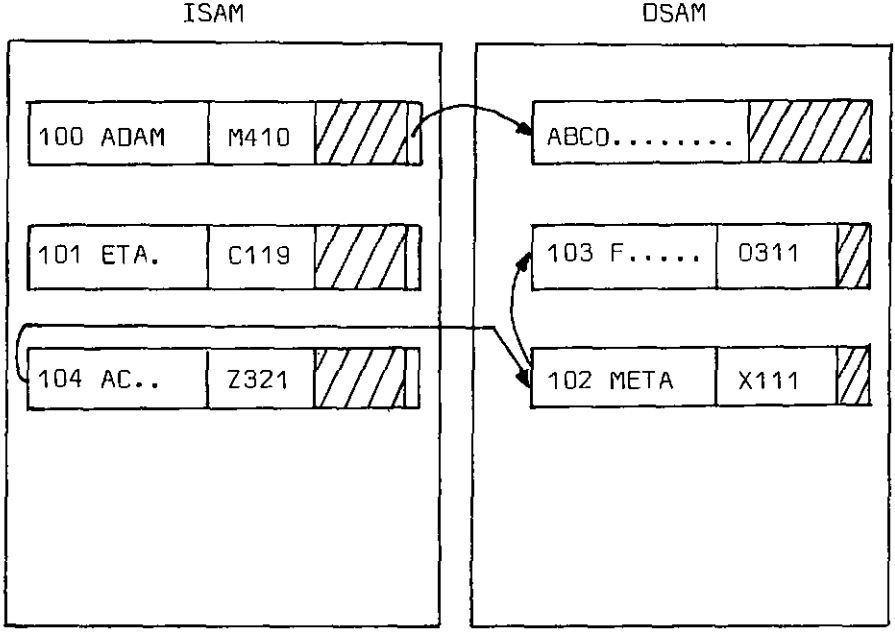


Figure 9.5 - Exemple d'enregistrements HISAM

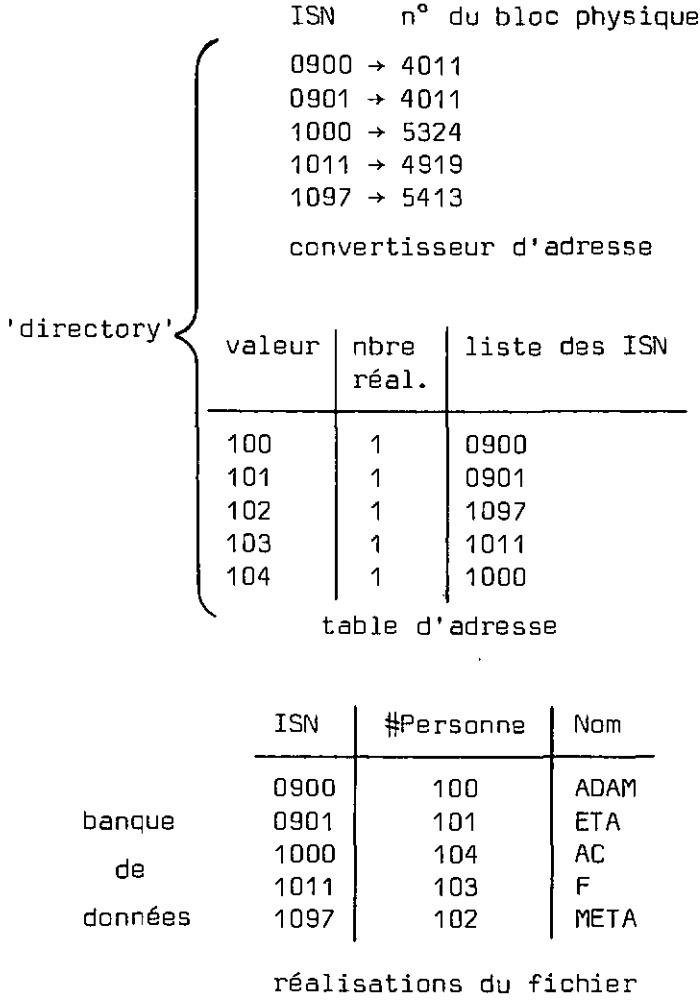


Figure 9.6 - Exemple d'enregistrements Adabas

Dans le cas de IMS, l'accès direct s'effectue toujours par l'intermédiaire du segment racine vers les éléments dépendants. La définition adéquate de "data set group" permet la réalisation d'accès direct à un ou plusieurs segments fils. Comme l'enregistrement des occurences de segments s'effectuent selon la séquence hiérarchique, la fréquence d'accès aux segments détermine donc l'emplacement de ces segments dans l'arbre. De plus, la fréquence des insertions par type de segments pose le problème de la gestion des zones de débordement et influence donc directement les performances (figure 9.5).

L'option "via set" dans le cas de IDMS favorise les accès, puisque le "record" concerné sera enregistré, si l'emplacement nécessaire est disponible, dans la page du "record" défini par ce paramètre. La fréquence d'accès recensée dans la matrice des requêtes sera déterminante dans le choix de cette option.

Dans le cas de Adabas, les performances peuvent encore être améliorées en introduisant des groupes répétitifs aux relations. Ainsi un nombre plus élevé de données est accessible en un seul transfert d'enregistrements dans la zone de travail.

Les performances d'un accès séquentiel physique à un des secteurs d'une banque de données (IMS = physical data base, IDMS = area, Adabas = fichier) sont influencées par les facteurs suivants:

- la suppression physique ou non des enregistrements annulés, en HISAM pas de suppression (IMS)
- les chemins d'accès sont-ils intégrés aux données, dans le cas de IDMS et IMS les pointeurs s'enregistrent avec les données
- la compression des données (Adabas)
- les emplacements libres qu'impliquent
  . la structure d'enregistrement

  - la méthode d'adressage
  - le taux de croissance du volume des données (avec HISAM les emplacements libérés par suppression ne sont pas réutilisables)
- un secteur physique IMS ou IDMS contient les occurences de plus d'un type de segment ou "record", dans le cas d'Adabas un tel secteur ne contient qu'un seul type d'enregistrement, l'espace à balayer est donc plus réduit
- dans le cas de IMS, la séquence physique est interrompue, des accès aux zones de débordement sont nécessaires.

Le choix du type de chaînage, dans le cas de IMS et IDMS, influence également les performances. Par exemple, dans le cas de IDMS, la définition d'une chaîne avec l'option "owner" permet d'accéder directement dans un set de l'enregistrement fils vers l'enregistrement père, sans passer par l'intermédiaire de l'algorithme d'adressage ou de suivre la chaîne.

Dans le cas de IDMS, la définition adéquate des "area" en fonction des fréquences d'accès et l'allocation sur les supports (disques) influencent les performances.

Avec Adabas, la position relative des champs d'une relation détermine les temps d'accès. Les champs les plus utilisés sont à placer en tête, car la décompression est interrompue lorsque tous les champs ont été transférés. D'autre part, l'accès aux champs multiples s'effectuera en lisant le nombre d'occurences correspondant à la distribution normale de l'occupation d'un tel champ, une comparaison avec le compteur des réalisations indiquant l'existence d'autres réalisations et nécessitera dans certains cas une deuxième lecture.

L'architecture du SGBD influence les performances en fonc-

tion des critères suivants:

- qualité des éléments composant le SGBD
- l'exécution d'une requête implique-t-elle des communications interpartition ?
- l'exécution des requêtes en parallèles (multi-threading) est-elle possible ?

b) L'emplacement mémoire

Un des objectifs de réalisation d'une banque de données consiste à minimiser l'occupation en mémoire périphérique. Cet objectif est en concurrence directe avec celui d'obtenir des performances minimales, opposition qui se traduit par une certaine redondance au niveau de l'enregistrement des données.

Les différentes formes de structures physiques exigent des emplacements en mémoire périphérique de dimensions variables. Les méthodes d'organisation directe (HDAM, Calc) ou indirecte (HISAM, HIDAM, tables d'adresses d'un fichier inversé) influencent l'emplacement mémoire par les informations supplémentaires nécessaires pour réaliser les accès et par les réserves exigées dans le cas de l'adressage direct pour réduire les risques de collision.

Le taux de remplissage est également influencé par la longueur des segments et le nombre d'occurences par type de segment dans le cas de IMS. Une illustration de ce phénomène est représentée par la figure 9.5.

Le mécanisme de compression des données du système Adabas permet de réduire, selon les cas, très fortement les emplacements nécessaires. Lorsqu'un champ est défini comme descripteur, la codification de la valeur la plus représentée par la valeur nulle permet d'économiser de l'emplacement au niveau des données et des tables d'adres-

ses (paramètres = 'NU').

Les possibilités de réaliser physiquement les différents types de structure logique implique, selon les cas, des emplacements supplémentaires en mémoire. A titre d'exemple, signalons le cas du système IMS qui ne permet de créer des réseaux qu'en introduisant des pseudo-segments et des pointeurs supplémentaires (logical pointers).

La réalisation de chemins d'accès utilisant des clés secondaires peut nécessiter plus ou moins de places en mémoires périphérique (fichiers inversés, organisation index-séquentielle).

Dans le cas de HISAM en IMS, la suppression de segments ne se traduisant pas par la libération de l'emplacement occupé, il est nécessaire de prévoir des espaces supplémentaires pour accueillir les nouveaux enregistrements si l'on veut éviter de trop fréquentes réorganisations.

Une étude comparative entre d'une part les emplacements nécessaires à la réalisation des chemins d'accès à l'aide de l'inversion et de l'indexation et d'autre part le rapport entre les données brutes et les informations exigées pour la réalisation des accès est présentée en (9.18).

c) Mise à jour

Les opérations de mise à jour consistent d'une part à effectuer des contrôles de validation et d'autre part à réaliser la modification, la suppression ou l'insertion d'un enregistrement donné.

Pour des raisons d'efficacité, comme nous l'avons exposé au point a) de ce chapitre, il est résonnable de réin-

troduire de la redondance au niveau des données primaires, ce qui pose des problèmes pour la mise à jour. Il est souvent nécessaire de prévoir des procédures spéciales afin de respecter l'intégrité de la banque de données.

La structure d'enregistrement physique ainsi que le choix effectué dans la réalisation des chemins d'accès doivent être confrontés aux contraintes résultant des opérations de mise à jour. L'accès exigé pour effectuer une telle opération a été pris en considération dans la matrice des requêtes. L'objectif de l'analyse vise surtout à étudier la structure physique en considérant :

- le temps de positionnement supplémentaire consécutif à une mise à jour, par exemple l'enregistrement dans une zone de débordement (HISAM), le traitement de collisions ou dans le cas d'enregistrements de longueurs variables le déplacement dans un autre bloc (dans le cas de Adabas, un choix adéquat de l'emplacement réservé dans chaque bloc permet de minimiser cet effet) ;
- la mise à jour des chemins d'accès qui, selon leur configuration, exige du temps de traitement supplémentaire (pointeurs, tables d'adresse) ;
- le temps supplémentaire que peut exiger l'exécution d'une requête, une mise à jour provoquant, lors de l'insertion d'un segment ou "record", une plus grande dispersion des données ;
- des réorganisations peuvent s'avérer nécessaires pour des raisons de performance (diminuer les débordements, améliorer l'accès) ou pour libérer des emplacements mémoires. Les réorganisations doivent être limitées quant à leur nombre et il peut s'avérer nécessaire de renoncer à certains chemins d'accès ou de les réaliser sous une autre forme ;
- les problèmes liés à la redondance de données primaires ;

- le respect des contraintes d'intégrité, car la façon d'exécuter les opérations de mise à jour peut violer ces contraintes et les SGBD considérés ne prévoient pas de procédures de contrôle suffisantes.

Les résultats de cette analyse impliqueront très souvent des modifications du modèle de données décrit avec le DDL du SGBD considéré, le critère déterminant étant le taux de modification des données, paramètre déterminé lors de la description des chemins d'accès logiques (chapitre 8).

Une fois le modèle optimal obtenu, par cette approche itérative, qui peut être complétée par des tests à partir du modèle réduit créé lors de la phase du choix du SGBD, celui-ci devra être décrit selon la syntaxe du DDL. Dans le cas de IMS et de IDMS, une telle description implique la définition de paramètres d'implantation physique. Dans le cas de Adabas, ces paramètres sont à générer séparément.

Remarque: *Lors de l'analyse des besoins en information (chapitres 6.1, 8.4 et 8.5), nous avons insisté sur la distinction entre informations de contrôle ou destinées à la prise de décision. Pour ce second type d'information on constate que les SGBD existants sont relativement mal adaptés, car ils n'offrent pas une souplesse d'accès suffisante*

9.3.2. Définition des paramètres d'implantation physique

Conformément aux principes de la méthode proposée, ces paramètres ne devraient pas intervenir au niveau de la description de la structure logique (ce principe est respecté dans le cas du système Adabas).

Un des paramètres à déterminer est la longueur de l'enregistrement physique (IMS = blocs DSAM et ISAM, IDMS = Page size) qui dépend de la longueur des segments, du nombre d'occurrences moyen par type de segments appartenant au secteur considéré (IMS = physical data base, IMS = Area), de contraintes de performance et du type de support.

L'allocation sur les supports disques se fera en fonction des fréquences d'accès aux différents secteurs de façon à minimiser le temps de positionnement des têtes de lecture-écriture.

L'emplacement nécessaire, en nombre de blocs ou pages, à l'enregistrement d'un secteur donné est fonction

- de la méthode d'enregistrement
- de l'allocation ou non des emplacements libérés
- de la croissance du volume des données
- de l'utilisation de zones de débordement.

Le choix de la méthode d'organisation, comme nous l'avons signalé au cours du chapitre 9.3.1.2, dépend des accès aux enregistrements concernés. Dans le cas de Adabas, la structure physique est dictée par les SGBD (organisation directe relative, bloc de longueur fixe).

L'intervention de ces paramètres au niveau de la description de la banque de données est spécifique au SGBD utilisé.

9.4. CONCEPTION DU SYSTEME DE COMMUNICATION

Le mode d'utilisation, simultané ou séquentiel, le type d'opérations à exécuter, les contraintes de sécurité et d'intégrité, ainsi que les procédures de protection dont dispose le SGBD, influencent directement les mesures à prendre lors de la conception du système de communication. Les critères déterminants sont:

- le degré de protection
- les pertes de performance résultant de l'exécution des opérations de contrôle
- les emplacements mémoires supplémentaires.

Les problèmes d'intégrité et de sécurité qui en découlent ont été décrits au cours du chapitre 7. Les mesures à prendre dépendent également des contraintes déterminées lors de la phase d'élaboration des chemins d'accès logiques (chapitre 8).

La première opération consiste à synchroniser le SGBD et le moniteur de télétraitement (si celui-ci n'est pas intégré au SGBD, comme c'est le cas pour IDMS et Adabas). Cette synchronisation impliquera:

- les mesures nécessaires à une reprise en cas de destruction de la consistence de la banque de données (journalisation, point de reprise synchronisé)
- le contrôle des droits d'accès.

Selon les possibilités respectives des deux systèmes et les critères formulés ci-dessus, une adaptation des programmes d'application et/ou de la structure de la banque de données peut s'avérer judicieuse. Elle peut se traduire par

- la création de programmes d'application (transaction) en fonction des données à traiter et des

contraintes de confidentialité et l'adaptation de la structure logique en conséquence. Par exemple, dans le cas d'une application de gestion du personnel, lire et modifier les adresses seront des opérations exécutées par un programme différent de celui de la mise à jour du salaire, les informations relatives au salaire étant enregistrées dans une autre entité
- la création d'un fichier intermédiaire afin de réaliser les mises à jour en différé et par lot
- des mesures organisationnelles à prendre au niveau de l'exécution des tâches par les utilisateurs lorsque les reprises ne sont possibles qu'à partir d'un point fixe.

Une autre décision à prendre est la stratégie à adopter face aux problèmes de la mise à jour simultanée et du phénomène d'interblocage qui peut en résulter. Les solutions dépendent des mécanismes mis à disposition par le SGBD et, comme nous l'avons signalé au cours du chapitre 7.1.2, le problème est loin d'être résolu. Enumérons les stratégies possibles en fonction de ces limites:

- ignorer le phénomène en ne bloquant pas les ressources concernées par une opération de mise à jour,les risques découlant de cette stratégie pouvant être contrés par des mesures au niveau de l'organisation (par exemple allouer les abonnés à gérer par opérateur, mises à jour indirectes et traitées par lot à l'extérieur des heures de bureaux)
- détecter le phénomène en indiquant à l'utilisateur que les ressources demandées sont bloquées et le prier de répéter sa requête tout en libérant les ressources qu'il aurait préemptées. Cette solution est plus simple que celle à adopter dans le cas des traitements par lot et qui consiste à répéter (boucle) la requête un nombre

de fois déterminé et en cas d'insuccès de provoquer une interruption de programme suivie du lancement d'une procédure de reprise automatique ou de libérer les ressources bloquées tout en protocolant les clés primaires des ressources concernées puis de continuer le traitement.

Quant aux droits d'accès, ils ne se définissent qu'en fonction de champs donnés et d'opérations de manipulation. Des conditions relatives à des plages de valeur doivent être contrôlées par des procédures à programmer, à moins que l'on utilise le SGBD Socrate.

La vérification du respect des contraintes d'intégrité par les programmes d'application est une tâche importante dont la responsabilité incombe à l'administrateur de la banque des données.

9.5. PROBLEMES SPECIFIQUES LIES A LA PROGRAMMATION

Il est généralement admis dans un environnement banque de données que les programmes d'application considèrent les données non en tant qu'enregistrement physique, mais en terme d'ensemble logique partiel. Seul ce sous-ensemble est mis à la disposition du programme. On parle dans ce cas de l'indépendance des programmes, c'est-à-dire qu'ils ne doivent pas être modifiés lorsque les paramètres d'implantation physique changent.

Une place importante est occupée dans les programmes par les procédures de contrôles des contraintes d'intégrité, les SGBD considérés ne prévoyant aucun mécanisme approprié (9.19).

La distinction principale entre les différents systèmes (9.20) résulte du degré d'indépendance des programmes,

notion pour laquelle il existe plusieurs interprétations:

a) L'indépendance physique.

b) L'indépendance par rapport aux modifications de structure. Dans le cas de IDMS et de IMS, les évolutions qui n'ont pas d'influences sur les programmes d'application sont limitées, car le modèle de données décrit explicitement les associations entre relations (segment ou record) et les chemins d'accès. Il n'est possible d'ajouter un champ à un segment ou un nouveau dépendant que si les réserves nécessaires ont été prévues. Par contre, une modification des chemins d'accès ou un transfert de champs d'un segment à un autre implique forcément une modification des programmes concernés.

   A titre de comparaison, le système Adabas offre la possibilité d'ajouter et de supprimer des champs à une relation ainsi que des chemins d'accès (descripteurs, couplage et découplage entre relations) à l'aide de programmes utilitaires sans modifier les programmes et sans exiger de réorganisation. De nouvelles requêtes utilisant des opérateurs booléens se référant à des champs descripteurs déjà existants peuvent se formuler sans impliquer au préalable une modification de structure.

c) L'indépendance par rapport aux chemins d'accès, degré d'indépendance le plus élevé, n'implique aucune connaissance de la structure physique lors de la définition d'une requête. Seuls les constituants et les relations doivent être connus.

Quels sont les facteurs qui déterminent cette notion d'indépendance ?

- le modèle de données au niveau logique ne doit

contenir aucun critère d'implantation physique, ni de descriptions des chemins d'accès (ce n'est pas le cas pour IDMS et IMS)

- le langage de manipulation implique-t-il un traitement enregistrement par enregistrement en suivant un chemin d'accès à définir par le programmeur (langage procédural). Le langage DL/1 de IMS et celui de IDMS sont de ce type. Les langages de manipulation relationnels (celui de Adabas en est une application restreinte, le choix des chemins d'accès se limitant aux accès par descripteur) ne nécessitent aucune connaissance de la structure physique et les chemins d'accès sont choisis par le système lui-même. Les données satisfaisant à la requête peuvent être appelées dans la zone de travail par le programme et il n'est pas nécessaire de prévoir une procédure d'accès spécifique. Dans le cas de Adabas, les numéros internes des relations répondant à la requête sont transférés dans la zone de travail, le programme peut ensuite demander les informations en vue des traitements prévus.

- Dans le cas des systèmes IDMS et IMS, la manipulation des données exigent de connaître à chaque instant la position courante sur le chemin d'accès (currency indicator) et les caractéristiques de ce chemin (sort by ..., ascending, etc), d'où une dépendance accrue.

L'impact du degré d'indépendance au niveau de la programmation se traduit entre autre

- dans la structure des programmes - le programmeur "navigue-t-il" d'occurence en occurence ou non ?
- par un effort de programmation qui décroît rapidement plus le degré d'indépendance est élevé, car il y a moins de possibilités de faire des erreurs -

les manipulations sont plus simples à programmer et la découverte d'erreurs de programmation, ainsi que l'exécution des tests sont plus rapides
- l'effort au niveau de la formation varie dans le même sens
- l'accès à la banque de données ne pourra s'ouvrir à des utilisateurs non spécialisés (utilisateur occasionnel) que si le degré d'indépendance est très élevé, ceux-ci ne disposant en aucun cas des connaissances techniques suffisantes
- l'introduction de primitives de manipulation de données plus puissantes que celles des langages hôtes (Cobol, PL/1) permettant de traiter des structures indépendamment de la structure physique se traduit par la diminution du nombre de procédures de tri.

La mise au point des programmes peut être facilitée si le programmeur dispose d'un outil permettant l'analyse des anomalies. Par exemple, le passage des programmes par un pré-compilateur (DML-Processor de IDMS) indiquera aux programmeurs les erreurs de manipulation.

L'existence d'utilitaires de chargement simplifiera la création d'une banque de données test et évitera de provoquer des efforts de programmation supplémentaires. De même, l'existence d'un langage d'interrogation interactif ou d'un simulateur offrant la possibilité de visualiser le transfert des instructions et des données entre le programme et le SGBD permettra une découverte très rapide des anomalies.

REFERENCES

[9.1] Une autre classification est proposée par ADV-Orga dans "Kriterien ..." p. 6.

[9.2] A titre de comparaison, consultez
ADAM B., "Kriterien zur Auswahl ..." p.705 ss.
ADV-Orga cité en [9.1] p. 7 ss.
NIESSING H., "Auswahlkriterien ..." p. 14 ss.

[9.3] BAZILLOU P.G., BENCI G.E., "Présentation et analyse de SGBD ..."
"Dossier sur les bases de données" p.

[9.4] A titre de comparaison, consultez
ADAM B. cité en [9.2] p. 704
ADV-Orga cité en [9.2] p. 68 ss.
NIESSING H. cité en [9.2] p. 17 ss.

[9.5] AUERBACH (G.B.)
DATAPRO (USA)
GMD (RFA)
IRIA (F)

[9.6] KAISER E.V., "Zur Auswahl und Optimierung ...".

[9.7] Les raisons de cette proposition sont spécifiées au chapitre 2.2.3

[9.8] Conformément aux objectifs formulés au chapitre 4.2

[9.9] IBM, "Data Base Design Aid, Designer's Guide".

[9.10] MITOMA M.F., IRANI K.B., "Automatic Data Base Schema Design.

[9.11] Consultez le chapitre 6.2.2. et l'annexe 1.

[9.12] LUM U.Y. et a., "Key-to-Address Transform Techniques ..." p. 228 ss.
NIEVERGELT J., "Binary Search Trees ..." p. 195 ss.
WEDEKIND H., "Datenorganisation".
WEDEKIND H., "Datenbanksysteme II" p. 125 ss.

[9.13] SENKO M.E., "File Organization ..." p. 111 ss.

[9.14] SCHRODER K., "Vergleich der Verweistechniken ..." p. 145 ss.

[9.15] WEDEKIND H., "Datenbanksysteme II" p. 197 ss.

[9.16] CARDENAS A.F., "Analysis and Performance ...".

[9.17] Un exemple de simulation est présenté en [9.15] p. 349 ss.

[9.18] WEDEKIND H. cité en [9.15] p. 178 ss.

[9.19] Le SGBD Socrate permet la description de plages de valeur au niveau du modèle conceptuel, contraintes qui sont vérifiées lors de manipulations de données.

[9.20] Une comparaison de la programmation entre l'approche relationnelle et Codasyl est présentée par CODD E.F. et DATE C.J., "The Relational and Network Approaches ...".

CHAPITRE 10

IMPLANTATION ET EXPLOITATION D'UNE BANQUE DE DONNEES

10.1. CHARGEMENT INITIAL DE LA BANQUE DE DONNEES

Lorsque le fonctionnement du système informatique de gestion et des règles gouvernant l'exécution des tâches auront été vérifiés en étroite collaboration avec les utilisateurs, on pourra réaliser le chargement initial de la banque de données.

La création de l'état initial nécessite des données provenant

- de fichiers classiques existants
- de fichiers gérés par un SGBD
- d'une saisie spécialement prévue .

Cette opération et le coût qui en découle dépendent des possibilités offertes par leSGBD.

a) Utilitaires de chargement

La solution idéale est obtenue si le SGBD fournit un ensemble d'utilitaires permettant d'effectuer cette opération de chargement. Les paramètres suivants sont à prendre en compte:

- description du fichier en entrée
- description de la structure physique
- contrôle de validation des données
- conversion de données
- règles de projection
- procédures de chargement.

Dans le cas des SGBD retenus dans notre étude, seul le système Adabas offre de telles facilités. Une seule restriction est la correspondance entre la structure du fichier en entrée et celle de la structure logique de la banque de données. Le système vérifie la longueur et le format des données. Ce contrôle peut se compléter par l'utilisation du "user exit" qui peut également servir

à la conversion des données.

Un modèle de conversion plus complet est présenté en (10.1) et illustré par la figure 10.1. Il comprend un langage de définition des fichiers en entrée et en sortie, ainsi qu'un langage de conversion permettant de décrire les différentes règles de conversion et les contrôles de validation. L'avantage du modèle réside dans sa flexibilité, un ou plusieurs fichiers tant en entrée qu'en sortie, et dans l'existence d'un langage non procédural. Mais l'exécution du chargement de la banque de données n'est pas résolu et exigera la confection de programmes de chargement spécifiques.

b) Réaliser des programmes de chargement

Si aucun utilitaire n'est à disposition, l'écriture de programmes spécifiques s'avère nécessaire. L'effort à fournir en codifications, compilations et vérifications de fonctionnement est relativement élevé et varie selon les SGBD.

Pour IMS, les données en entrée doivent se présenter sous forme hiérarchique et, dans le cas de HSAM, HISAM et HIDAM, elles doivent être triées selon la clé primaire du segment racine. Comme dans le cas de IDMS, le segment père se charge en premier.

Dans le cas de Adabas, l'enregistrement des données s'effectuera selon l'ordre logique du descripteur (clé primaire ou secondaire) le plus utilisé lors de traitements séquentiels logiques.

Un problème dont l'impact augmente en fonction directe du volume des données est le temps de traitement exigé par les opérations de chargement. Pour éviter des impasses à ce niveau, la mise en route échelonnée des applications

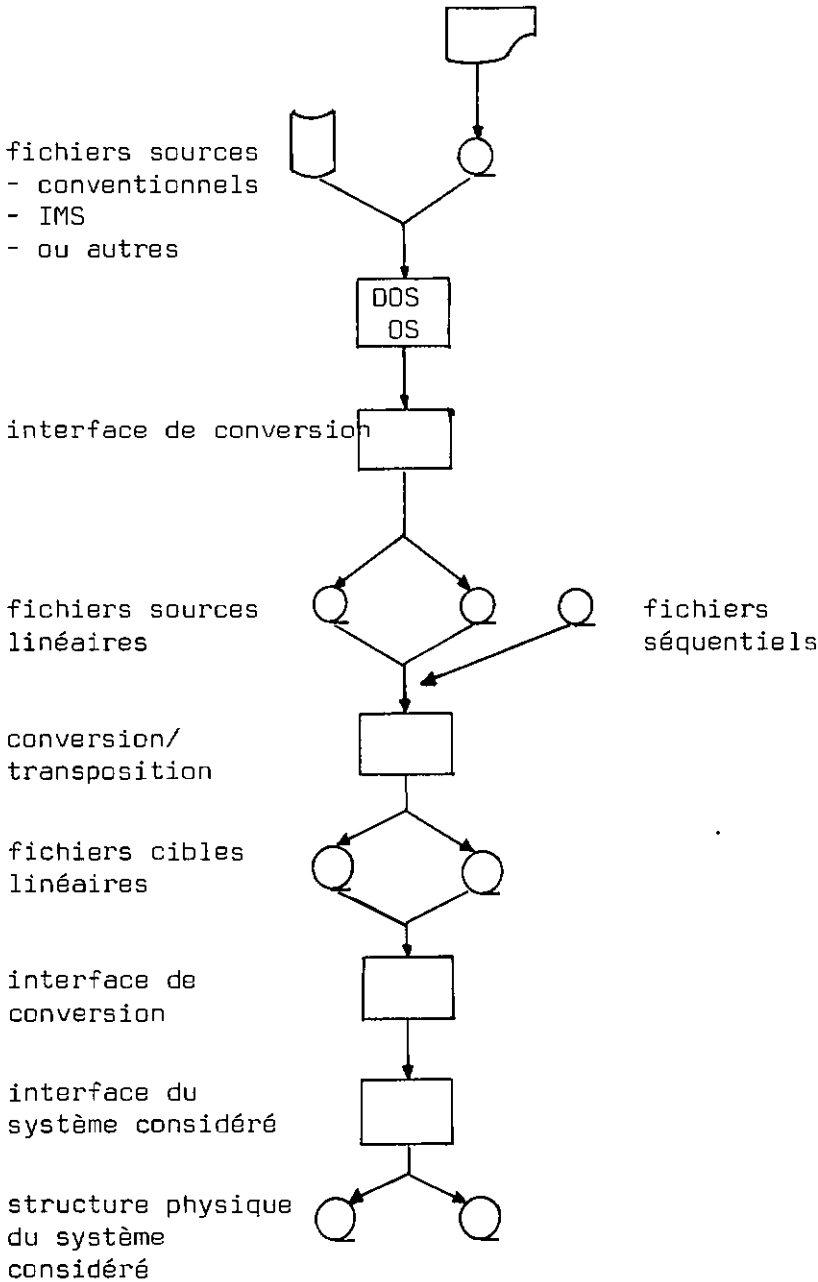


Fig. 10.1 - Eléments d'un système de conversion

s'avère préférable, si cela est possible. Par exemple, Adabas possède un utilitaire qui permet le chargement par bloc et non par enregistrement, d'où gain sensible de temps. D'autre part, l'existence d'utilitaires performants dont le but consiste à ajouter des masses de données importantes à des ensembles déjà existants facilite une réalisation par étape. Le système Adabas offre de telles possibilités.

En ce qui concerne la conversion de programmes existants, les efforts au niveau de la programmation dépendent principalement du degré d'indépendance que procure le SGBD et de la complexité du langage de manipulation des données.

10.2. PROBLEMES SPECIFIQUES A L'EXPLOITATION D'UNE BANQUE DE DONNEES

10.2.1. Pilotage des applications

Le pilotage dépend dans un tel environnement du mode de traitement (10.2). La responsabilité du lancement des programmes d'application sera si possible déléguée à l'utilisateur. Les paramètres relatifs à l'exploitation seront enregistrés dans un fichier spécifique mis à jour par le gestionnaire. La tâche de l'opérateur du centre de calcul se limitera à charger le système et à monter les supports.

Considérons certains problèmes d'exploitation particuliers à chaque mode de traitement:

a) Traitement par lot

Ce type de traitement se pilote en général sans problèmes particuliers. Le planning d'exploitation s'effectue facilement, car il est possible de tenir compte

du taux de charge de la machine. D'autre part, le traitemant par lot présente les avantages suivants:

- la reprise en cas d'incidents est plus simple à maîtriser
- il permet d'éviter les problèmes liés au partage des ressources
- il n'altère pas les performances du système interactif, il suffit d'élaborer un horaire de traitement en conséquence.

L'administrateur de la banque de données édictera, en collaboration avec le concepteur, les règles pour la reprise en cas d'incidents:

- journalisation
- procédure à activer
- fichiers à utiliser (journal, copie)
- messages à transmettre
- destinataires de ces messages
- nombre de génération des fichiers de sauvegarde à archiver.

b) Traitement transactionnel

Dans ce genre de traitement, c'est l'utilisateur qui déclenche l'exploitation des différents programmes en indiquant à son terminal les transactions qu'il désire utiliser.

Le planning se résume donc à fixer les horaires du traitement, tout en évitant les traitements mixtes afin de ne pas trop altérer les performances.

Les mesures à prendre lors d'une interruption dépendent des possibilités de reprises offertes par le SGBD et des contraintes découlant des applications. Des mesures organisationnelles au niveau de l'exécution des tâches administratives ne sont pas exclues.

c) Traitement conversationnel

Ce genre de traitement n'est possible que si le SGBD dispose d'un langage d'interrogation facile à apprendre. Dans le cas des systèmes que nous considérons dans cette étude, seules des interrogations seront autorisées, car ces langages n'effectuent aucun contrôle du respect des contraintes d'intégrité.

Les utilisateurs doivent connaître la structure logique et les contraintes d'intégrité, entre autre:

- plages de valeur
- cardinalité
- règles de codification.

Le problème de reprise en cas d'incident ne se pose pas, à condition que l'on n'autorise que des interrogations. L'utilisateur doit, dès la remise en route du système, reformuler la requête interrompue.

10.2.2. Les activités de l'administrateur de la banque de données

Des études détaillées sur l'activité de l'administrateur sont citées en (10.3). L'envergure des tâches dépend du SGBD utilisé. L'administrateur doit, en étroite collaboration avec le concepteur du système informatique:

- participer à l'analyse sémantique
- déterminer la structure logique
- définir les contraintes d'intégrité et de sécurité
- décrire les chemins d'accès logiques
- adapter la structure logique selon les caractéristiques du SGBD à disposition
- édicter des standards de programmation
- conseillers les réalisateurs

- former les analystes-programmeurs
- superviser la réalisation des programmes
- exécuter la conversion des fichiers
- vérifier si les vues partielles formulées sont conformes à la structure logique
- déterminer les chemins d'accès physiques
- édicter les mesures de sécurité en vue de garantir le bon fonctionnement du système.

D'autre part, il existe un certain nombre d'activités spécifiques à l'implantation et à l'exploitation de la banque de données, Nous en étudierons certains aspects ci-dessous.

a) Réaliser l'implantation de la banque de données

L'administrateur est responsable de la définition des paramètres d'implantation que nous avons décrits au chapitre 9.3.2.

Puisqu'il effectuera le chargement initial, il sera chargé, si aucun utilitaire n'existe, d'écrire les programmes nécessaires à cette opération. De même, il s'occupera de tout chargement ultérieur, car il doit

- vérifier si l'emplacement mémoire nécessaire est disponible
- allouer éventuellement de nouveaux espaces
- si nécessaire effectuer les réorganisations
- coordonner les activités du système.

La responsabilité de la définition des paramètres d'exploitation des différentes applications (job control) lui incombe également.

b) Garantir la sécurité d'exploitation

Puisqu'en phase d'exploitation, les risques d'incidents

ne peuvent être complètement exclus, l'administrateur doit rechercher à en minimiser les effets, ce qui implique la réalisation de procédures de

- prévention
- détection
- correction

d'erreurs qui peuvent survenir en cours d'exploitation.

1. mesures de prévention
   - prévoir la journalisation des mises à jour
   - fixer les points de reprise
   - édicter des règles d'exécution en cas de reprise, tant pour le centre de calcul que pour l'utilisateur
   - planifier le rythme de sauvegardes des fichiers
   - allouer les clés d'accès
   - vérifier le fonctionnement de nouvelles versions du SGBD
   - vérifier les interfaces entre système d'exploitation, moniteur de télétraitement et SGBD lors de modification
   - synchroniser les logiciels utilisés
   - s'assurer qu'un programme a été vérifié avant de donner le feu vert pour l'exploitation
   - surveiller l'évolution des volumes.

2. détection d'erreurs
   L'interruption anormale du système est signalée par un message spécifique, mais le SGBD n'indique pas si l'état de la banque de données est consistent ou pas. La détection des causes d'une interruption et des erreurs qui ont pu s'infiltrer est un travail complexe. La réalisation de procédures d'interruption prévoyant l'élaboration de "dumps" indiquant la valeur des paramètres transmis facilite cette tâche.

Dans le cas des SGBD réalisant les relations entre les éléments de la structure logique à l'aide de pointeurs, la catégorie-liaison doit être enregistrée dans les éléments reliés, afin de pouvoir détecter plus facilement des éventuelles erreurs d'enregistrement. Les erreurs de manipulation qui ne se traduisent pas par une interruption immédiate du programme sont très difficiles à détecter. L'existence d'utilitaires de manipulation ou d'un langage d'interrogation facilite le contrôle des enregistrements.

3. correction d'erreurs
En cas d'interruption due à une erreur de programmes, l'administrateur devra décider des mesures à prendre et en informer les utilisateurs:

- mettre en route une procédure de reprise
- si nécessaire, organiser la correction des enregistrements altérés.

c) Mise à l'épreuve du système

En phase d'exploitation, le système informatique de gestion est mis quotidiennement à l'épreuve. Seule une utilisation intensive permettra de vérifier si le système satisfait aux exigences des utilisateurs. Cette phase d'évaluation du fonctionnement du système implique l'étude des problèmes suivants:

- les temps de réponse sont-ils satisfaisants ?
- le déroulement des procédures administratives fonctionne-t-il correctement ?
- les informations fournies correspondent-elles effectivement aux besoins des utilisateurs ?
- le nouveau système ne provoque-t-il pas de surcharges de travail ?
- les instructions données aux utilisateurs sont-elles adaptées aux compétences des utilisateurs ?
- les avantages prévus sont-ils effectifs ?

- les temps de traitement correspondent-ils aux contraintes fixées ?
- les taux d'activité divergent-ils de ceux établis lors de la conception du système ?

Donner une réponse à ces questions implique l'existence de certains outils (tools) permettant à l'administrateur de surveiller l'efficacité et le comportement du système. Ceux-ci dépendent du SGBD utilisé et en général, ils donnent des renseignements sur

- le nombre de transactions par unité de temps
- le temps de traitement des programmes
  - . durée effective
  - . temps machine
  - . taux d'utilisation mémoire centrale
  - . nombre d'accès physiques par support logique (fichier)
  - . nombre d'accès par point d'entrée dans un fichier (champ inversé)
  - . nombre d'accès par type d'opérations
- le taux de pagination
- la croissance du volume en emplacement mémoire
- le taux d'occupation mémoire
- la fréquence des collisions et longueur des chaînes de collision
- le taux de charge des canaux d'entrée/sortie.

La mesure des activités d'un tel système est très complexe à réaliser du fait de l'exploitation multi-utilisateur. Les informations obtenues sont de nature statique et leur interprétation est relativement difficile (10.4).

Une amélioration des performances exige une analyse détaillée de la dynamique des utilisations du système. L'administrateur analysera entre autre:

- la fréquence d'exécution des requêtes
- les chemins d'accès physiques choisis

- le profil de ces chemins d'accès
  - . longueur
  - . distribution des valeurs
  - . distribution physique de ses éléments sur les supports
  - . rapport entre l'envergure du chemin suivi et le volume des données du sous-ensemble concerné
- l'influence de l'utilisation sur la structure d'enregistrement
  - . fréquence des débordements
  - . évolution du taux de collision
  - . évolution des emplacements libérés mais non utilisables
  - . fréquence des déplacements d'enregistrements dans le cas d'une structure physique de longueur variable
- la structure des programmes
- la réalisation des procédures d'accès par les programmes d'application.

L'utilisation de moniteurs hardware ou software donnera des informations supplémentaires (10.5).

Les mesures qui s'imposent en vue d'améliorer les performances peuvent se traduire soit par des modifications des programmes d'application ou n'exiger aucune adaptation à ce niveau. Plus le degré d'indépendance est élevé, moins les modifications à apporter aux programmes seront importantes. Les mesures à prendre peuvent être, par ordre croissant des efforts à réaliser, les suivantes:

- modifier les priorités d'accès
- modifier la fréquence des réorganisations
- modifier l'allocation mémoire des fichiers
- augmenter la dimension de l'espace physique
- modifier le taux de blocage
- augmenter la réserve par bloc dans le cas des

enregistrements de longueurs variables
- créer ou supprimer des points d'entrées dans un sous-ensemble
- modifier le type de chaînage
- modifier les procédures d'accès au niveau des programmes d'application
- modifier la structure logique.

La durée de cette phase d'adaptation et les efforts nécessaires dépendent de la qualité de l'analyse de conception, des méthodes de travail utilisées, de la qualité du travail fourni lors de la réalisation, de la complexité et de la flexibilité du SGBD utilisé.

d) Surveiller la stabilité du système

Une fois la phase d'adaptation terminée, la surveillance de l'exploitation du système se résume à

- suivre l'évolution de la croissance des volumes de données
- surveiller l'évolution des performances
- effectuer les activités en vue de garantir la sécurité d'exploitation
- évaluer les modifications apportées par les nouvelles versions du SGBD, vérifier leur fonctionnement
- étudier l'évolution technologique
- revoir et améliorer les standards de programmation et la documentation.

e) Réaliser les adaptations qu'implique l'évolution du système

Cette activité, point critique de l'utilisation d'un SGBD, est très souvent sous-estimée. Les problèmes soulevés feront l'objet du chapitre suivant.

10.3. EVOLUTIONS DU SYSTEME

Le problème de l'évolution se pose sous deux angles différents, le premier étant le résultat d'une action voulue et planifiée en conséquence. Comme nous l'avons indiqué au cours du chapitre 4, il est judicieux de réaliser de grands projets par étapes successives. Dans le cadre de notre méthode d'approche, les aspects liés à ce type d'évolution ont été étudiés lors de la phase d'analyse des chemins d'accès logiques (chapitre 8) et soulèvent les problèmes liés à

- la conversion de données
- chargement des nouveaux éléments
- modifications de structures possibles
- aux contraintes d'intégrité
- l'adaptation de programmes existants
- la rupture de la stabilité du système par l'apparition de nouveaux utilisateurs, en particulier la dégradation des performances.

Des mesures adéquates doivent être prises afin de ne pas perturber le déroulement des activités des services utilisateurs. L'exécution des opérations représentées à la figure 10.2 s'effectue plus facilement si des utilitaires

- de conversion et de chargement (chapitre 10.1)
- de modifications de structure (ceux proposés par le système Adabas ou ceux cités en référence (10.6))

existent.

Le deuxième aspect de l'évolution est celui provoqué par l'apparition de nouveaux besoins. Selon l'ampleur des évolutions, différentes situations peuvent se présenter:

1. Les nouveaux schémas externes correspondent au modèle global

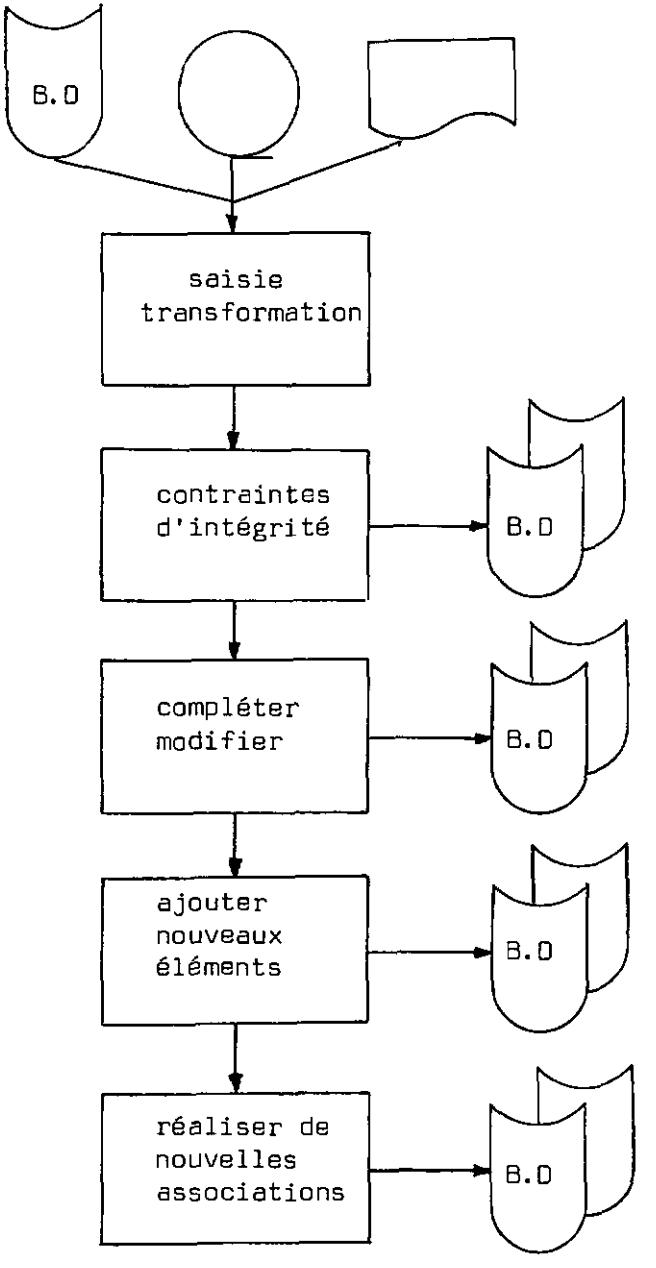


Fig. 10.2 - Evolution de la banque de données

Dans un tel cas, il est possible que la structure physique ne permette de satisfaire les nouveaux besoins de façon efficace, les chemins d'accès nécessaires faisant défaut. Les problèmes posés par une telle situation sont difficiles à résoudre dans le cas de systèmes dont la description des chemins d'accès est explicite au niveau du modèle. Le choix des chemins d'accès étant codifié au niveau des programmes, des adaptations importantes en résultent.

Dans le cas des systèmes réseau ou hiérarchique, de telles modifications sont réalisables avec un minimum d'effort lorsque les conditions suivantes sont remplies:

- la place nécessaire pour réaliser les nouvelles associations à l'aide de pointeurs est prévue au niveau des segments considérés (sinon il faut réorganiser)
- les catégories selon lesquelles les liaisons doivent s'effectuer figurent dans la partie données des segments ou "record" dépendants.

2. Le modèle de la structure globale doit être modifié
   Différentes situations peuvent se présenter:

   - de nouvelles catégories doivent être ajoutées à une relation existante. Cette modification ne pose pas trop de problèmes si l'emplacement nécessaire est prévu. Dans le cas de Adabas, une telle modification s'effectue à l'aide d'un utilitaire qui complète la description de la relation concernée, aucun emplacement de réserve et aucune réorganisation ne sont nécessaires

. de nouvelles entités (segment, "record" ou relation) doivent être ajoutées. Dans le cas des systèmes de type réseau ou hiérarchique, cette modification exige des efforts relativement élevés, les mêmes que ceux découlant de l'extraction de champs d'un élément donné (voir ci-dessous)

. l'extraction de champs d'un élément (segment ou "record") existant exige des efforts importants en
  .. réorganisation
  .. procédures spécifiques de chargement et de réalisation de cette modification
  .. modifications du schéma et des sous-schémas
  .. modifications de programmes existants.

La réalisation pratique des modifications qu'impliquent l'évolution des besoins s'avère dans beaucoup de cas très problématique, voire même impossible si l'on considère

- les temps de réorganisation qui prennent des dimensions énormes plus le nombre de liaisons pointeurs et le volume des données est important
- les temps de chargement des nouveaux éléments
- l'adaptation des programmes d'application que cela implique
- les perturbations dans l'exploitation du système existant que provoque l'implantation de ces modifications.

En fonction des problèmes que posent l'évolution d'un système, il est primordial de les évaluer avec soin déjà au cours de la phase de conception (chapitre 6 et 8), lors du choix du SGBD (chapitre 9.1) et de la réalisation physique (chapitres 9.2 et suivants).

REFERENCES

[10.1.] Lum V.Y. et a. "Data Translation..."

[10.2.] Groupe de travail français de normalisation
dans "Etudes des besoins des utilisateurs" p. 59 ss. décrit les conditions auxquelles un SGBD doit satisfaire en phase d'exploitation.

[10.3.] Sans auteur "The Data Administrator Function"
Guide "The Data Base Administrator To-Day"
Guide "The Space Manager"

[10.4.] Sarzotti A. "Introduction aux techniques d'évaluation et de mesure..."
Svobodova L. "Computer Performance Measurement..."

[10.5.] Sans auteur "Techniques d'évaluation et mesure des performances"

[10.6.] Lum V.Y. et a. cité en [10.1.]

Conclusions

L'objectif de notre étude consistait à proposer une méthode d'approche pour la conception d'une banque de données dans le cadre d'un système informatique de gestion.

D'abord, nous avons démontré la nécessité de disposer d'une méthode

- évolutive
- intégrante et non partielle
- indépendante d'un SGBD donné
- exhaustive, englobant tous les critères à prendre en considération,

afin que le concepteur dispose d'un schéma d'approche le guidant tout au long du cycle de développement.

Notre première préoccupation a été de déterminer les critères nécessaires à la conception d'un système d'information (chapitre 2).

Sur la base d'une analyse critique des méthodes d'approche existantes et des expériences vécues, nous avons été amenés à proposer une méthode d'approche descendante composée de phases successives (chapitres 3 et 4).

Nous avons ensuite insisté sur la nécessité de modéliser le système de gestion et proposé une solution ayant fait ses preuves dans la pratique. Seuls les critères principaux ont été abordés. L'étude ne s'est pas concentrée sur la construction de modèles mais plutôt sur les relations existant entre un modèle global et la conception d'une banque de données (chapitre 5).

Quant à la méthode d'approche proposée en vue de la conception d'une banque de données, nous avons démontré qu'elle satisfait aux objectifs fixés:

- les phases de la méthode n'impliquent pas la résolution de phases ultérieures
- les aspects logiques sont étudiés indépendamment des critères physiques du SGBD
- l'analyse sémantique est une condition impérative à toute conception
- les relations entre les éléments de la structure des données sont étudiées sous tous leurs aspects et indépendamment d'un SGBD quelconque
- la méthode de description paramétrique des applications (requêtes) implique l'analyse des critères conditionnant
    - . le fonctionnement du système
    - . la réalisation des objectifs fixés pour les applications
        - .. performance
        - .. coûts
        - .. fiabilité
        - .. flexibilité
        - .. besoins en information
- les problèmes relatifs à la réalisation physique sont analysés et la méthode permet l'utilisation de n'importe quel SGBD
- la mise en place de la banque des données, ainsi que les problèmes spécifiques à l'exploitation n'ont pas été négligés
- les outils d'aide à la conception proposés sont simples à utiliser et n'impliquent pas l'existence de logiciels spécifiques et coûteux.

Le souci majeur qui nous a guidé dans notre travail peut se résumer par la nécessité de mettre en place une méthode

d'approche présentant un haut niveau de:

- cohérence
- indépendance
- flexibilité
- practicabilité
- adaptabilité
- intégralité.

Au-delà des résultats présentés et des idées émises dans cette étude, nous pensons que certains approfondissements mériteraient d'être réalisés, entre autre:

- développer à chaque phase du cycle de développement des outils automatisés en tant qu'aide à la conception, si possible interactif. Certains travaux ont déjà été réalisés dans cette optique ou sont en cours de développement
- prévoir des possibilités de générer à chaque étape les modules composant le système d'information
    - . modèles de données
    - . fonctions de traitement
- créer un langage de composition capable d'assembler les éléments nécessaires en vue de la mise en route des applications, ce qui permettrait de réduire les efforts au niveau de la programmation
- le problème relatif à la répartition des données sur différents sites est posé, car la décentralisation des capacités informatiques est devenue une réalité et prendra encore à l'avenir de l'importance
- des SGBD modulaires mieux adaptés aux besoins

particuliers des utilisateurs devront encore être crées, entre autre pour répondre aux exigences de la décentralisation

- le dynamisme des applications de gestion, bête noire du concepteur, devrait pouvoir être étudié à l'aide d'outils automatisés d'analyse des chemins d'accès, ceux-ci devant être capables de mesurer l'impact de l'évolution des besoins sur la structure du système
- enfin, les problèmes relatifs à la documentation gagnent en importance, du fait de la complexité et des interdépendances entre les éléments composant un système. Ceux-ci exigent aujourd'hui un effort considérable et la mise à jour n'est pas toujours garantie. Des outils spécifiques seraient à prévoir.

Cette énumération n'est certes pas exhaustive, mais nous pensons que des développements en ces domaines sont souhaitables, afin de faciliter la maîtrise des problèmes posés aux différentes étapes du cycle de vie d'un système informatique de gestion. Les difficultés à surmonter dans cette "aventure" sont encore importantes, mais nous espérons que les recherches en cours permettront d'aboutir à des outils opérationnels. Une coopération industrie-université serait souhaitable et nous ne pouvons qu'encourager toute tentative dans ce but.

ANNEXES

LISTES DES ACTIVITÉS DES DIFFÉRENTES PHASES DU CYCLE DE DÉVELOPPEMENT D'UN SYSTÈME INFORMATIQUE DE GESTION

1. METHODE D'APPROCHE POUR LA MODELISATION D'UN SYSTEME DE GESTION

1.1. Objectifs

1.1.1. Analyse préalable

- analyser les informations de base sur l'entreprise
- interview des-dirigeants concernés
- analyser la situation dans les différents secteurs
- analyser les plans existants.

1.1.2. Fixer les objectifs

- définir les secteurs concernés
- objectifs des systèmes informatiques
  - . qualitatifs
  - . quantitatifs
- moyens à dispositions
- investissements possibles.

1.2. Conception du modèle global

- analyser l'état du système de traitement des informations
- analyser les activités principales
- fixer les priorités pour la suite de l'analyse
- analyser les interdépendances entre activités
- détecter les points faibles
- élaborer un premier modèle

- étudier des systèmes semblables dans des entreprises du même secteur
- déterminer les contraintes
- élaborer définitivement le modèle global
- décision quant à la suite des études de développement.

1.3. Conception des modèles par sous-systèmes

- analyser le modèle global
- concrétiser les objectifs et activités du sous-système
- concrétiser les besoins en information
- concrétiser les procédures de traitement
- déterminer les contraintes
- concrétiser les faiblesses du système actuel
- concrétiser les interrelations entre centre d'activités
- analyser les contraintes socio-psychologiques
- concevoir le modèle
- conséquence pour la gestion des données
- conséquences matériels/logiciels
- conception globale de solutions alternatives
- structure des fonctions principales du système
- opportunité des solutions proposées
- élaborer des dossiers pour la prise de décision.

1.4. Prise de décision

- comparer les solutions aux objectifs fixés
- évaluer les avantages respectifs
- évaluer les conséquences économiques de chaque alternative sur l'entreprise
- évaluer les objectifs économiques
- problèmes du financement
- fixer les priorités quant à la suite des études de développement
- choisir une solution.

2. METHODE D'APPROCHE POUR LA CONCEPTION D'UNE APPLICATION

2.1. Conception logique du système

2.1.1. Etudier le modèle

- analyser les éléments
- étudier les influences sur le déroulement de la phase
- compléter les imprécisions.

2.1.2. Déterminer les exigences du système

- compléter la description des activités
- déterminer les exigences de fonctionnement
- déterminer les besoins en informations
- destination/source des informations
- distinguer les besoins subjectifs et objectifs

- déterminer le volume et la fréquence des besoins
- déterminer les contraintes temporelles et de performance
- évaluer les besoins en information
- déterminer la logique des processus de traitement
- déterminer les contraintes de confidentialité et de sécurité
- déterminer les contraintes de révision
- déterminer l'influence sur l'organisation
- préciser les données en sortie
- déduire les données en entrée
- étudier les problèmes de saisie de données.

2.1.3. Conception globale du système

- étudier le software d'exploitation
- étudier le software d'application
- choix du software
- étude du hardware
- choix du hardware
- choix du mode de traitement
- établir la structure du système
- définir les algorithmes de traitement
- respect des contraintes de performance
- respect des contraintes d'intégrité et de sécurité
- fixer le flux des données.

2.1.4. Plan de réalisation

- planifier la réalisation du système
- planifier l'implantation
- établir une analyse des coûts/valeurs.

2.1.5. Prise de décision

- étudier les objectifs
- étudier les propositions
- décider.

2.2. Conception physique du système

2.2.1. Analyse du modèle conceptuel

- analyser les modules
- étudier l'influence sur la suite de la conception
- compléter les résultats de l'étape précédente.

2.2.2. Structure des programmes

- respecter la restriction du hardware et du software
- fixer la structure définitive des programmes
- formuler des procédures de traitement
- fixer le déroulement des traitements
- fixer les contrôles de validité
- fixer les procédures de traitement des erreurs
- fixer les contrôles de fonctionnement.

2.2.3. Déterminer les chemins physiques d'accès

- consulter l'acheminement de la base de données
- fixer les chemins d'accès
- respecter les contraintes de performances
- déterminer la vue de l'utilisateur
- fixer les commandes d'accès.

2.2.4. Conception des entrées/sorties

- déterminer le type de traitement
- fixer les contraintes de classement
- décrire la logique des procédures de traitements des sorties
- déterminer les clés
- concevoir les supports en sortie
- concevoir les supports en entrée.

2.2.5. Contraintes d'intégrité et de confidentialité

- contraintes d'intégrité
- mesures de sécurité
- mesures garantissant l'exploitation
- fixer le système de confidentialité.

2.2.6. Elaborer le dossier de programmation

- rassembler les informations
- décrire la structure du programme
- décrire les procédures de traitement des modules

- fixer les procédures de test.

3. METHODE D'APPROCHE POUR LA CONCEPTION D'UNE BANQUE DE DONNEES

3.1. Analyse sémantique

3.1.1. Déterminer les besoins en information

- analyser le modèle
- compléter les imprécisions
- préparer l'approche pour déterminer les besoins en information
- déterminer les besoins en information
- établir une matrice des besoins en information.

3.1.2. Analyse sémantique des données

- analyser la matrice et déterminer les catégories
- déterminer les relations entre catégories
- établir un graphe logique
- éliminer les incohérences et redondances
- vérifier si le modèle représente bien la vue du monde réel concerné
- déterminer les fonctions d'accès et les paramètres d'existence
- description du graphe.

3.2. Conception de la structure logique

3.2.1. Les contraintes d'intégrité

- analyser le graphe logique
- déterminer les contraintes d'intégrité en fonction des manipulations de données possibles
- décrire les contraintes.

3.2.2. Les contraintes de confidentialité

- classer les utilisateurs
- déterminer les données accessibles par utilisateur
- déterminer les opérations de manipulation exécutables par utilisateur
- décrire les contraintes.

3.2.3. Etablir la structure logique

- analyser les relations entre catégories
- effectuer des regroupements
- déterminer les identificateurs de liaison
- établir le modèle de la structure logique.

3.3. Chemins d'accès logiques

3.3.1. Déterminer les paramètres

- recenser les requêtes
- sous-ensemble de la structure concernée
- fréquence d'utilisation

- opérations de mise à jour
- fixer les bornes des chemins d'accès logiques
- contraintes de performance
- critères d'ordre
- type de traitement
- nature des données
- rapport coût/valeur
- évaluer l'importance des requêtes.

3.3.2. Décrire les chemins d'accès logigues

- étudier les processus de traitement
- établir la matrice des requêtes
- respecter les paramètres d'existence
- compléter, si nécessaire, la structure logique.

3.3.3. Optimaliser le modèle

- analyser la matrice des requêtes
- comparer les fréquences d'utilisation
- migration de l'ancien vers le nouveau système
- dynamisme du système de gestion
- vérifier les contraintes d'intégrité
- élaborer le modèle optimal.

3.4. Les chemins d'accès physiques

3.4.1. Choix du SGBD

- déterminer les critères d'évaluation
- fixer les objectifs en fonction des applications
- lancer un appel d'offre
- étudier les SGBD à disposition
- présélection
- établir une structure de préférence
- évaluer les SGBD présélectionnés
- choix
- établir une liste des caractéristiques du SGBD choisi ou à réaliser.

3.4.2. Modèle global

- passage de la structure logique au modèle selon DDL du SGBD
- analyser les chemins d'accès logiques
- étudier le tableau descriptif de la structure logique
- respecter les contraintes relatives au DDL
- prendre en considération le problème du passage de l'ancien vers le nouveau système
- réaliser les associations entre catégories.

3.4.3. Réalisation des chemins d'accès

- analyser les chemins d'accès logiques

- étudier les possibilités de réalisation physique
- étudier les contraintes matériel et logiciel (hardware et software)
- définir la réalisation physique des associations entre catégories
- vérifier les contraintes de performance
- modifier le modèle
- étudier les problèmes liés à la mise à jour
- vérifier le respect des contraintes d'intégrité
- améliorer l'occupation mémoire
- analyser les répercussions de l'optimalisation sur les applications
- déterminer le mode d'accès
- déterminer les répercussions pour la programmation.

3.4.4. Définir les paramètres d'implantation

- déterminer le volume d'un enregistrement
- déterminer les secteurs
- allocation des types d'enregistrement aux secteurs
- allocation sur les supports physiques
- définir les zones de débordement.

3.4.5. Réalisation du système de communication

- étudier les mécanismes de sécurité du moniteur de télétraitement
- réaliser un interface moniteur de télétraitement → SGBD
- réaliser le système de sécurité
- déterminer les répercussions au niveau de la programmation des applications
- déterminer les répercussions au niveau de l'organisation du travail.

3.4.6. Programmation

- écrire des modules spécifiques à certaines manipulations de la B.D. (contrôle d'intégrité, statistiques, etc)
- vérifier le respect des contraintes d'intégrité dans les programmes d'application
- conseiller les programmeurs d'application
- prévoir le processus de vérification des programmes
- mettre à disposition les éléments nécessaires à la réalisation de ces vérifications
- générer les utilitaires nécessaires
- préparer les procédures de commande d'exécution ("Job Control").

3.4.7. Planifier la phase suivante

- capacité nécessaire
- matériel à mettre à disposition
- logiciel spécifique à créer
- déroulement de la phase.

3.5. Implantation

3.5.1. Tester l'intégration des applications

- préparer le test
- exécuter le test
- vérifier les résultats
- effectuer les corrections nécessaires.

3.5.2. Pilotage des applications

- fixer les procédures d'exécution
- déterminer le planning de l'exécution des mesures de sécurité
- fixer le déroulement des procédures d'archivage
- édicter les règles à suivre au niveau de l'exploitation en cas de reprise
- élaborer le dossier d'exploitation
- instruire les opérateurs.

4. EXPLOITATION DU SYSTEME INFORMATIQUE DE GESTION

4.1. Plan des opérations

- besoins des utilisateurs
- restrictions Hardware-Software
- temps de traitements nécessaire
- mesures préventives de fonctionnement
- opérations exigées pour des raisons de sécurité
- planifier les réorganisations physiques
- établir plan de charge.

4.2. Exécution

- préparer les supports de données nécessaires
- préparer l'ordre d'exécution
- distribuer les supports de données en sortie aux utilisateurs
- remettre supports dans l'archive
- exécuter les mesures de sécurité
- exécuter les réorganisations physiques

4.3. Evaluation de performance

- contrôler le taux de charge du système
- contrôler performance des applications (tuning)
- contrôler performance du système (monitoring)
- déclencher les mesures adéquates
- effectuer les modifications nécessaires
- contrôler l'efficacité des modifications.

4.4. Réorganisation logique

- modifier la structure logique
- adapter les programmes
- préparer les algorithmes de transformation de la base existante
- étudier le problème de la portabilité des copies de l'ancienne base
- exécuter la réorganisation.

BIBLIOGRAPHIE

ABRIAL J.R.
"Data Semantics" in "Data Base Management" éd p. Klimbie J.W. et Koffemann K.L., Proceedings of the IFIP Working Conference, North-Holland Publishing, Amsterdam, 1974, p. 1-59

ABRIAL J.R., COHEN J.P., FAVRE J.C., PORTAL D., MAZARE G., MORIN R.
"Projet Socrate", Nouvelles spécifications, version 3, Université Scientifique et Méd., Math. appl. inf., Grenoble, sept 1972

ADABAS
"Einführung", Documentation, Software AG, Darmstadt, mars 1976

"Adascript", user manual

ADAM B.
"Kriterien zur Auswahl von Datenbanksystemen", On-line, no 11, 1976, p. 704-706

ADIBA M., DELOBEL C.
"Les modèles relationnels de bases de données", IRIA-Sefi, avril 1976

ADIBA M., DELOBEL C., LEONARD M.
"An Unified Approach for Modelling Data in Logical Data Base Design" in "Modelling in Data Base Management Systems" Preprints, IFIP-TC 2 Working Conference, Freudenstadt 1976, p. 636-665

ADV-ORGA
"Kriterien zur Auswahl eines Datenbank-Systems", séminaire

"Orgware II", Benutzerhandbuch

ANSI/SPARC
"Interim Report", FDT Bulletin of ACM-SIGMOD, vol. 7, no 2, 1975

ANTON J.P., CHRISMENT C.Y., CRAMPES J.B., DEBAISIEUX M.F., LUGUET J.H.
"Scapface - Un système de Conception Automatique et Progressive d'un système informatique Fondé sur l'Analyse Conversationnelles des Etats de sortie" in "Panorama de la nouveauté informatique en France", Congrès de l'AFCET, nov. 1976

ASTRAHAN M.M., BLASGEN M.W., CHAMBERLAIN D.D., ESWARAN K.P., GRAY J.N., GRIFFITHS P.P., KING W.F., LORIE R.A., McJONES P.R., MEHL J.W., PUTZOLU G.R., TRAIGER I.L., WADE B.W., WATSON V.
"System R: Relational Approach to Database Management", ACM Transaction of Database Systems, vol 1, no 2, 1976, p. 97-137

BACHMANN C.W.
"The Programmer as Navigator", Comm. of ACM, vol 16, no 11, 1973, p. 653-658

BAZILLOU P.G., BENCI G.E.
"Présentation et analyse de SGBD commercialisés en France", IRIA-Sti, Paris, 1975

BENCI G.E., BODART F., CABANES A., DEHENEFFE C., HAINAUT J.L., LEROY H., RANDON J., SAVOYSKI S., THULY J.
"Introductory Report" in "Data Structure Models for Information Systems", Proceedings, Presse Universitaire de Namur, Namur 1975, p. 15-56

BLASER A., SCHMUTZ H.
"Data Base Research: A Survey" Report TR 75.10.009, IBM Heidelberg Scientific Center, nov. 1975

BLUMENTHAL S.C.
"Système de gestion informatique MIS", Entreprise Moderne d'Edition, Paris, 1971

BOULENGER J.
"Informatique et Gestion", Sirey, Paris, 1975

BOYCE R.F., CHAMBERLIN D.D.
"Using a structured English query language as a data definition facility", IBM Research Report RJ 1318, San José, 1973

CARDENAS A.F.
"Analysis and Performance of Inverted Data Base Structure", Comm. of ACM, vol. 18, no 5, 1975, p. 253-263

CASTELLANI X.
"Méthode générale d'analyse d'une application informatique", tome 1 et 2, Masson, Paris, 1975

CHAMBERLIN D.D., GRAY J.N., TRAIGER I.L.
"Views, Authorization and Locking in a Relational Data Base System", IBM Research Report RJ 1486, San José, 1974

CHAMBERLIN D., BOYCE R.F.
"Sequel: a Structured English Query Language" in "Workshop on Data Description, Acces and Control" éd p. Rustin R., ACM-SIGMDD, 1974, p. 249-264

CHEN P.P-S
"The Entity-Relationalship Model - Toward a Unified View of Data", ACM Transaction on Database Systems, vol. 1, no 1, 1976, p. 9-36

CHENIQUE F.
"Analyse fonctionnelle et organique", Dunod, Paris, 1974

CLUB BANQUE DE DONNEES
"Rapport du groupe Analyse et Bases de Données", Bulletin de Liaison du Club Banque de Données, IRIA, no 15, 1975, p. I1-I58

CODASYL
"Data Base Task Group Report", ACM, avril 1971, New York

"Feature Analysis of Generalized Data Base Management Systems", IFIP, Amsterdam, mai 1971

CODD E.F.

"A Relational Model for Large Shared Data Banks", Comm. of ACM, vol 13, 1970, p. 377-387

"A Data Base Sublanguage Founded on the Relational Calculus" in "Workshop on Data Description, Access and Control" éd p. Rustin R., ACM-SIGFIDET, 1971, p. 35-68

"Relational completeness of data base sublanguages" in "Data Base Systems" éd p. Rustin R., Prentice-Hall, Englewood Cliffs, 1972

"Further Normalization of the Data Base Relational Model" in "Data Base Systems" éd p. Rustin R., Prentice-Hall, Englewood Cliffs, 1972

"Seven steps to RENDEZ-VOUS with the casual user", IBM Research Report RJ 1333, San José, 1974

CODD E.F., DATE C.J.

"The Relational and Network Approaches: Comparison of the Application Programming Interfaces", IBM Report RJ 1401, Yorktown Heights, 1974

"Interactive Support for Nonprogrammers: The Relational and Network Approaches", IBM Research Report RJ 1400, Yorktown Heights, 1974

COUGER J.D., KNAPP R.W.

"Systems Analysis Techniques", Wiley, New York, 1974

CXP

"Etude comparative des méthodes d'analyse", rapport du Centre d'eXpérimentation des Packages, Paris

DANIELS A., YEATES D.

"Grundlagen der Systemanalyse", Müller Verlag, Köln, 1971

DATE C.J.
"An Introduction to Database Systems", Addison-Wesley,Reading, 1975

DEHENEFFE C., HAINAUT J.L., TARDIEU H.
"The Individual Model" in "Data Structure Models for Information Systems", Proceedings, Presse Universitaire de Namur, Namur, 1975, p. 89-118

DEHENEFFE C., HENNEBERT H., PAULUS W.
"Relational Model for a Data Base" in "Information Processing 74", Proceedings of IFIP Congress 1974 éd p. Rosenfeld J.L., Preprints, Amsterdam, 1974

DELOBEL C.
"Contributions théoriques à la conception et l'évaluation d'un système d'informations appliqué à la gestion", thèse, Université Scientifique et Médicale, Grenoble, 1973

"Les systèmes de banques de données", séminaire de 3ème cycle, Université de Genève, 1975-1976

DELOBEL C., CASEY R.G.
"Decomposition of a data base and the theorey of boolean switching functions", IBM Journal of Research and Development, no 5, 1973, p. 374-386

DELOBEL C., LEONARD M.
"The Decomposition Process in a Relational Model" in "Data Structure Models in Information Systems", Proceedings, Presse Universitaire de Namur, Namur, 1975, p. 57-80

DOUQUE B.C.M., NIJSSEN G.M.
"Data Base Description", Proceedings of the IFIP-TC2 Working Conference on Data Base Description, North-Holland Publishing, Amsterdam, 1975

EDP ANALYZER

"Organizing the Corporate Data Base", EDP Analyzer, vol 8, no 3, 1970

GAGSH S.

"Eine Methode zur computer-gestützten Vorbereitung der graphischen Darstellung von integrierten Gesamtmodellen" in "Integrierte Gesamtmodelle der Datenverarbeitung" éd p. Grochla E., Hanser Verlag, München, 1974, p. 92-105

"Probleme der Partition und Subsystembildung in betrieblichen Informationssystemen" in "MIS - Eine Herausforderung an Forschung und Entwicklung", Gabler Verlag, Wiesbaden, 1971, p. 623-652

GARDARIN G., SPACCAPIETRA S.

"Integrity of Data Bases: A General Lockout Algorithm with Deadlock Avoidance" in "Modelling in Data Base Management Systems" éd p. Nijssen G.M., North-Holland Publishing, Amsterdam, 1976, p. 395-411

GERBER F.

"Das Soll-Konzept: Die Qual der Wahl", Output, no 3, 1973, p. 56-59

GIBERT A.

"Les banques de données", tome 1 et 2, Edition Genti, Paris, 1973

GILLNER R.

"Die Strukturierung informationsbedarfsorientierter betrieblicher Datenbanken in computer-gestützten Informationssysteme", thèse, Universität Köln, 1974

GRAY J.N., LORIE R.A., POTZOLU G.R., TRAIGER I.L.

"Granularity of Locks and Degrees of Consistency in a Shared Data Base" in "Modelling in Data Base Management Systems" éd p. Nijssen G.M., North-Holland Publishing, Amsterdam, 1976, p. 365-394

GROCHLA E.

"Integrierte Gesamtmodelle der Datenverarbeitung. Entwicklung und Anwendung des Kölner Integrationsmodell", Hanser Verlag, München, 1974

GROCHLA E., GARBE H., GILLNER R.

"Gestaltungskriterien für den Aufbau von Datenbanken", Westdeutscher Verlag, Opladen, 1973

GROCHLA E., MELLER F.

"Datenverarbeitung in der Unternehmung, Grundlagen", Rowohlt Verlag, Reinbek, 1974

GROUPE DE TRAVAIL FRANCAIS DE NORMALISATION

"Etudes des besoins des utilisateurs", Bulletin de Liaison du Club Banque de Données, no 16, IRIA, 1976, p. 44-73

GUIDE

"The Space Manager", Performance Evaluation and Management Group, Guide International, Chicago

"The Data Base Administrator To-Day", Data Base Administration Project, Guide Europe, Schindler Informatik, Ebikon, 1975

GUIDE/SHARE

"Data Base Management Systems Requirements", Joint Guide/Share Data Base Requirements Group, New York, 1970

HABERFELLNER R.

"Systems Engineering", Zeitschr. f. Organisation, Nr 7, 1973, p. 373-386

HABRIAS H.

"Méthodes d'enquête pour construire un diagramme de circulation et de traitement des informations", Informatique et Gestion, no 59, 1974, p. 90-98

HAERDER T.

"Auswahl von Zugriffspfadstrukturen in Datenbanksystemen", TH Darmstadt, 1975

IBM

"Interacive Query Facility (IQF) for IMS/360", IBM-Form GH20-1074, 1974

"IMS/VS General Information Manual", IBM-Form GH20-1260-2, 3ème édition, 1975

"IMS/VS System/Application Design Guide", IBM-Form SH20-9025-2, 3ème édition, 1975

"Data Base Design Aid. Designer's Guide", IBM-Form GH20-1267-1, 2ème édition, 1976

IDMS

"IDMS User Manual", Cullinane Corporation, Boston

JOUFFROY C., LETANG C.

"Les fichiers - pratique et choix de l'organisation des données informatiques", Dunod, Paris, 1974

KAISER E.V.

"Zur Auswahl und Optimierung von Datenbank-Management-Systemen (DBMS) - Eine Untersuchung mit Hilfe von Benchmark-Tests", ISAS-Projektbericht Nr 7, Köln, 1973

KING W.F.

"System R", séminaire de 3ème cycle, école de printemps, Anzère, 1976

KIRSCH W.

"Entscheidungssysteme", Gabler Verlag, Wiesbaden, 1971

"Probleme der Unternehmungsführung bei der Entwicklung und Implementierung von Management-Informationssystemen", Die Unternehmung, Nr 3, 1974, p. 173-185

KLIMBIE J.W., KOFFEMANN K.L.
"Data Base Management", Proceedings of The IFIP-TC2 Working Conference on Data Base Description, North-Holland Publishing, Amsterdam, 1974

KOREIMANN D.S.
"Methoden der Informationsbedarfsanalyse", de Gruyter, Berlin, 1976

LE MOIGNE J. L.
"Les systèmes d'information dans les organisations", PUF, Paris, 1973

"Les systèmes de décision dans les organisations", PUF, Paris, 1974

LEONARD M.
"Aides algorithmiques à la conception de bases de données", thèse, Institut National Polytechnique, Grenoble, 1976

LESCA H.
"Les ambiguités de l'analyse", Informatique et Gestion, no 45, 1973, p. 33-38

"Introduction à la gestion automatisée", PUF, Paris, 1974

LESUISSE R.
"Introduction au projet ISODS", Bulletin de Liaison du Club Banques de Données, no 16, 1976, p.163-201

LUM V.Y., YUEN P.S.T., DODD M.
"Key-to-Adress Transform Techniques: a Fundamental Performance Study of Large Existing Formatted Files", Comm. of ACM, vol 14, no 4, 1971, p. 228-239

LUM V.Y., SHU N.C., HOUSEL B.C.
"Data Translation Part 1: A General Methodology for Data Conversion and Restructuring", IBM Research Report RJ 1525, Yorktown Heights, 1975

LUTZ T.
"Die Stellung der Datenbank im Informationssystem" in "Datenbanken im Dienste des Unternehmungsführung" Grochla E., Garbe H., Lutz T., Wedekind H., Borgmann Verlag, Dortmund, 1971, p. 15-31

LYON K.
"An Introduction to Data Base Design", Wiley, New York, 1971

MARTIN J.
"Computer Data-Base Organization", Prentice-Hall, Englewoods Cliff, 1975

MATTHEWS D.
"The Design of the Management Information Systems", Auerbach Publishers, London, 1971

MELESE J.
"L'analyse modulaire des systèmes de gestion", Edition Hommes et Techniques, Paris, 1972

MERTEN H.
"Datenbank-Organisation", Müller Verlag, Köln, 1974

MITOMA M.F., IRIANI K.B.
"Automatic Data Base Schema Design and Optimization" in "Very Large Data Bases" éd p. Kerr D.S., Proceedings of ACM, Framingham, 1975

MZG
"Systementwicklung", Seminarunterlagen, Handelshochschule Sankt-Gallen

NICK F., OHLEN C.P., SCHAEDEL V., SCHAEFER R.
"Systeme der computergestützten Systemgestaltung", BIFOA-Arbeitsbericht 72/7, Wison Verlag, Köln, 1972

NIESSING H.

"Auswahlkriterien für Datenbanksoftware. Dargestellt am Beispiel des Problemkreises Einwohnerwesen", Adl-Nachrichten, Nr 99, 1976, p. 14-18

NIEVERGELT J.

"Binary Search Trees and File Organization", Computing Surveys, vol 6, no 3, 1974, p. 195-207

PEAUCELLE J.L.

"Les méthodes de recherche en informatique des organisations - Les besoins des utilisateurs", séminaire INFORSID, IRIA, 1975, p. 210-224

RENAUD D.

"Le système de base de données Adabas est-il un modèle relationnel", Centre de Calcul Electronique de l'Administration Fédérale, Berne, 1976

RICHTER G.

"On the Relationship between Information and Data" in "Data Base Systems" éd p. Hasselmeier H., Spruth W.G., Lecture Notes in Computer Science no 39, Springer, Berlin, 1976, p. 21-43

ROMERA S.

"Tables de décision et programmation", Informatique et Gestion, no 43, 1972, p. 66-71

RUBIN M.L.

"Introduction to the System Life Cycle", volume 1, Handbook of Data Processing, Brandon System Press, Princeton, 1970

Sans auteur

"The Data Administrator Function", EDP-Analyzer, vol 10, no 11, 1972

"Les méthodes d'analyse", Informatique et Gestion, no 45, 1973, p. 32-61 et no 49, 1973, p. 26-56

"Techniques d'évaluation et mesure des performances", Informatique et Gestion, no 83, 1976, p. 21-64

SARZOTTI A.
"Introduction aux techniques d'évaluation et de mesure des systèmes informatiques, Eyrolles, Paris, 1977

SCHMIDT G.
"Organisation. Methode und Technik", Schmidt Verlag, Giessen, 1974

SCHROEDER K.
"Vergleich der Verweistechniken in Datenbanksystemen", Angewandte Informatik, Nr 4, 1972

SENKO M.E.
"The Data Independent Accessing Model (DIAM)" in "Data Structure Models for Information Systems", Proceedings, Presse Universitaire de Namur, Namur, 1975, p. 81-88

"File organization and management information system", Annual Rev. of Information Sc. and Technology, no 4, 1969, p. 111-143

SENKO M.E., ALTMAN E.B., ASTRAHAN M.M., FEHDER P.L.
"Data Structures and Accessing in Data Base Systems", IBM Systems Journal, vol 12, no 1, 1973, p. 30-93

SKRONN H.J.
"Methoden der Strukturierung von Datenbanken", Angewandte Informatik, Nr 5, 1973, p. 204-210

STONEBAKER M., WONG E., KREPS P., HELD G.
"The Design and Implementation of INGRES", ACM Transaction on Database Systems, vol 1, no 3, 1976, p. 189-222

SVOBODOVA L.
"Computer Performance Measurement and Evaluation Methods: Analysis and Applications", Elsevier, New York, 1976

SYSTOR

"So verwirklichen wir EDV-Projekte", Benutzer-Handbuch

TEICHROEW D., SAYANI H.

"Automation of System Building" in "System Analysis Techniques" éd p. Couger J.D., Knapp R.W., Wiley, New York, 1974, p. 379-391

TEICHROEW D., SIBLEY E.H.

"Isdos Phase I - Report", Isdos Project, Departement of Industrial Engineering, University of Michigan, Ann Arbor, 1969

THURNER R.

"Entscheidungstabellen. Aufbau-Anwendung-Programmierung", VDI Verlag, Düsseldorf, 1972

TOZER E.E.

"Database Systems Analysis and Design", ECI Conference 1976, Proceedings, Lecture Notes in Computer Science, no 44, Springer, Berlin, 1976, p. 193-224

WANG C.P., WEDEKIND H.

"Segment Synthesis in Logical Data Base Design", IBM Journal of Research and Development, vol 19, no 1, 1975, p. 71-77

WEDEKIND H.

"Datenorganisation", de Gruyter, Berlin, 1972

"Systemanalyse. Die Entwicklung von Anwendungssystemen für Datenverarbeitungsanlagen", Hanser Verlag, München, 1973

"Normalization of Information Flow Diagrams in a Data Base", Forschungsbericht, TH Darmstadt, 1973

"Datenbanksysteme I", Bibliographisches Institut, Mannheim, 1974

WEDEKIND H., HAERDER T.
"Datenbanksysteme II", Bibliographisches Institut, Mannheim, 1976

WILL H.J.
"Datenbank-Systeme", Zeitschrift für Betriebswirtschaft, Nr 12, 1971, p. 815-844

YAO S.B.
"Selection of File Organization Using an Analytical Model" in "Very Large Data Bases" éd p. Kerr D.S., Proceedings of ACM, Firmingham, 1975, p. 255-267

YVON P.J., SEMIN C.
"Comment concevoir un système intégré de gestion", Entreprise Moderne d'Edition, Paris, 1970

ZEDA
"Quick-Draw", Dokumentationshandbuch, Wuppertal

ZLOOF M.M.
"Query by Example: The Invocation and Definition of Tasks and Forms", IBM Research Report RC 5115, Yorktown Heights, 1975