MASTER EN SCIENCES INFORMATIQUES

BIGAM-DBMS. Un système de Gestion de Bases de Données "réseau" mis à l'Index

Monsanto, P.

Award date:
1988

Awarding institution:
Universite de Namur

FACULTES
UNIVERSITAIRES
N.D. DE LA PAIX
NAMUR

INSTITUT D'INFORMATIQUE

P. MONSANTO

B I G A M - DBMS

Un Système de Gestion de Bases de Données "réseau" mis à l'Index

Mémoire
1987-1988

Promoteur: J-L. Hainaut

"Vous êtes remerciés", dit-on aux licenciés
Mais seraient-ils fâchés de n'être point cités ?
De moi, humble ouvrier qui n'ai rien à cacher
Il ne faut l'espérer et j'en suis désolé

Bernard, Fry et Teorey, Hainaut et Cadelli
Van Bastelaer, Cherton et membres du CITI
A vous et aux amis, très sincèrement merci...

P. Monsanto

Exemplaire destiné à:

AVERTISSEMENT :

Les "libres commentaires" et le dernier chapitre n'engagent que l'auteur; toute autre personne est réputée en ignorer l'existence.

PLAN

====

"Ce n'est pas le moment de m'interroger: je cherche les moyens de retenir de nouveaux préceptes, bien supérieurs à ceux de Pythagore, de Socrate et du docte Platon."
Horace - Satires, Livre II - IV.

1. INTRODUCTION

Faculté de conserver les idées antérieurement acquises; dissertation scientifique ou littéraire.

Ces définitions au féminin et masculin de "mémoire", empruntées au Larousse Classique, pourraient avantageusement remplacer les usuels mots-clefs jetés en pâture au lecteur désireux de s'épargner de fastidieux efforts face aux élucubrations inintéressantes, voire délétères, de certains essais à caractère informatique. Quelques mécanismes de conservation de données seront effectivement évoqués tout au long de ce document, selon des idées antérieurement acquises; si la démarche suivie peut certes receler quelque originalité, les concepts et techniques mis en oeuvre sont, en effet, connus de longue date. Quoique ne pouvant affirmer les posséder toutes, je commettrai également quelques réflexions sur les Facultés. Quant au côté scientifique ou littéraire de cette dissertation, il vous appartiendra de le déterminer et, le cas échéant, les définitions du vocable hermaphrodite gravées au frontispice de cet essai feront une épitaphe à peu de frais.

Vous l'aurez compris: quoique singulier, ce travail a sans nul doute mauvais genre...

1.1 Synthèse liminaire

Dans le cadre de la démarche de conception de bases de données développée à l'Institut d'Informatique, disposer d'un Système de Gestion de Bases de Données (SGBD) performant apparaît primordial.

Nonobstant leur sophistication extrême, les SGBD disponibles sur le marché ne semblent pas toujours répondre aux critères d'efficacité, rigueur et sécurité requis; en outre, les meilleurs d'entre eux ne peuvent être qualifiés de financièrement abordables. Certaines réalisations pilotes ont cependant démontré (NDBS - Institut d'Informatique - 1987) qu'il serait parfaitement envisageable de construire sous nos latitudes des SGBD soutenant la comparaison avec les produits nord-américains. Subséquemment, l'auteur de ces lignes se propose de commettre, dans un premier temps, un sous-système utile de SGBD répondant provisoirement au barbare acronyme de SPQR (Sophisticated Page Quickening Routines). La préméditation étant établie, nul ne s'étonnera d'une prompte récidive : un SGBD perpétré de sang froid. (Circonstance aggravante: cet esprit rétif et retors purge actuellement les derniers mois d'une peine de 5 ans...).

Le niveau de performance et la qualité du produit logiciel considéré devront rencontrer, autant que faire se peut, les normes professionnelles en vigueur.

1.2 *Libre commentaire*

Il s'agit donc essentiellement d'une réalisation pratique où spécification et programmation firent l'objet d'âpres discussions constructives entre promoteur et chef de chantier: les sempiternelles polémiques nées de problèmes inhérents aux "grands ensembles" forcent au consensus. Un espace de liberté fut dès lors soigneusement délimité, permettant l'observation in vitro des soubresauts désordonnés du chien fou s'ébrouant dans le Mato Grosso des documents et programmes de l'atelier de conception de bases de données. "La végétation luxuriante et putride, principalement constituée de strncpy, rend le déchiffrement/défrichement extrêmement pénible (...) Le chantier progresse trop lentement, à coups de delete meurtriers, et l'hépatite C me gagne..." écrivit-il à un de ses proches éloigné au printemps 1988. Résolu à passer malgré tout, il trahira sans vergogne les spécifications de son promoteur qui, fort heureusement, se montrera magnanime.

Franchissons ensemble le seuil de cette présentation générale pour entrer dans l'édifice chancelant et lézardé de la démarche proposée.

1.3 Les fondements morphogènes des SGBD

L'information exerce sur la gent humaine une fascination indéniable; recueillie, emmagasinée, détenue, désirée, convoitée, exhumée, manipulée, censurée, interdite, dissimulée, propagée, disséminée: cette liste des usages et traitements en la matière est loin d'être exhaustive. En outre, s'il faut en croire les messies technologiques, nous entrons dans une ère qui consacrera la valeur grandissante de cette denrée. Frisant l'impalpable et l'indéfini, cette information n'est bien souvent disponible qu'au travers de représentations, par ailleurs multiformes; ces dernières, organisées en collection, forment une base de données. Un système de gestion de bases de données, du moins dans son acception mécaniste, en régit toutes les interactions.

* SCHEMAS D'UNE BASE DE DONNEES

Cependant, les démarches modernes de conception de Systèmes d'Information permettent, par le biais de modélisations, de lever en partie l'indétermination et s'attachent à dissiper l'équivoque pesant sur toute définition informelle.

Ainsi, une base de données sera perçue comme un ensemble d'entités mutuellement associées, auxquelles seront affectées des valeurs d'attributs.

Les meilleurs auteurs s'accordent à considérer au moins trois niveaux de spécification, correspondant à trois étapes distinctes de description d'une base de données ou schéma .

Schéma conceptuel:

- *Spécification des types d'entités,*
  des types d'associations,
  des attributs qui leur sont attachés.
- *Descriptions en langage naturel.*
- *Contraintes d'intégrité.*

Schéma logique:

- *Origine: schéma conceptuel.*
- *Adjonction de structures d'accès aux données.*

Schéma physique:

- *Origine: schéma logique.*
- *Spécification des caractéristiques physiques du SGBD.*
- *Mise en conformité aux structures admises par un SGBD.*

Les mécanismes fondamentaux permettant la construction d'une base de données à partir de son schéma conceptuel sont les transformations et les enrichissements ; leur examen détaillé sort du cadre de ce document.

La coexistence nécessaire de ces divers schémas suscite bon nombre de réflexions on ne peut plus pertinentes. Elles prenent racine dans quelques épineux problèmes liés à la modélisation, la calculabilité, la technologie et, last but not least, la philosophie. Cette cohabitation présuppose la mise en oeuvre exclusive d'une politique de rigueur ou d'ouverture .
La première conduira au système:

- Schéma conceptuel / modèle conceptuel X
  Schéma logique / modèle logique Y
  Schéma physique / SGBD cible Z
- Etablissement de correspondances X - Y et Y - Z
- Multiplicité de concepts,
  de terminologies.

La seconde approche, réconciliatrice:

- Schéma conceptuel / modèle conceptuel X
  Schéma logique / modèle logique X'
  Schéma physique / SGBD cible Z
- Etablissement d'une correspondance X' - Z
- Adoption d'une terminologie unique.

S'agissant d'un mémoire (travail encadré), avouer son scepticisme face aux deux triptyques conceptuel/logique/physique proposés recèle quantité d'inconvénients:
Que choisir, comment l'exposer et sous quel jour l'illuminer? Que faire quand le promoteur clairvoyant s'illustre dans la seconde approche par un document d'une clarté halogène ? Comment éclairer les lumières d'un obscur étudiant s'il doit marcher à l'ombre ?
En contrepartie, la pluralité des avis en cette matière, divergents ou contradictoires, justifie a priori tout discours original mais étayé, quand bien même ce dernier ne serait pas fruit de célèbres colloques et autres synodes. Sans autres prétentions que celles d'engendrer moult polémiques, le souteneur de cet écrit s'essaiera à présenter, pour sa défense, une liturgie alternative non-schismatique.

* DOCUMENT DE REFERENCE

Cependant, respectueux de la tradition séculaire de conception de bases de données en vigueur aux Facultés, il eut été malséant, sinon périlleux, de contrecarrer l'important effort de simplification et d'unification des concepts, entrepris dans le cadre de l'atelier ORGA. Plus prosaïquement, le client étant roi, le travail à réaliser se devait de satisfaire aux prérequis méthodologiques énoncés dans le document "Conception d'une base de données - Eléments méthodologiques - Equipe bases de données de l'institut d'informatique - 1987 FNDP NAMUR".

Qu'en dire ?

Eliminons, de prime abord la flagornerie (incertaine et plagiaire), le recours à la consultation volontaire du "lecteur intéressé" (pléonastique) et la tautologie (tentante mais guère rémunératrice). Bannissons également le recopiage permuté pour son inévitable incohérence et répudions les résumés tronqueurs: place au texte intégral.

Les différents aspects d'un schéma, ses structures sémantiques (types d'entités, types d'associations, attributs, contraintes d'intégrité) et les structures d'accès sont explicités avec élégance dans ce document que vous trouverez en annexe I. La terminologie adoptée dans la suite de ce mémoire respectant celle détaillée dans le document mentionné, sa lecture est vivement conseillée.

* APERCU DES FONCTIONS D'UN SGBD

Outre la description d'un schéma, la manipulation des données (insertion, sélection, recherche, modification...) et les contraintes d'intégrité évoquées supra, la reprise sur incident, la confidentialité et la gestion de la concurrence font également partie des fonctionnalités souhaitables. Toutefois, n'étant qu'imparfaitement esquissées dans la réalisation pratique proposée à votre évaluation, il ne semble guère souhaitable de leur consacrer plus qu'une mention, à l'endroit où les spécifications pourraient prétendre émarger au monde de la sécurité. Découvrons-les ensemble...

"Mais un auteur qui fait sa cour ne laisse pas de provoquer l'aversion, tandis que le dénigrement et l'envie trouvent des oreilles complaisantes: c'est qu'à l'adulation s'attache le déshonorant reproche de servilisme, alors que la malignité a un faux air d'indépendance."

Tacite - Histoires, Livre I - I.

2. SPECIFICATION LAPIDAIRE

From:uunet.UU.NET!prlb2!fun-cs!jlh
To :citi.umich.edu!prlb2!pedro
Date:Fri, 11 Dec 87 10:14:38 GMT

Quelques caractéristiques du SGBD portable

0. Objectifs

portabilité maximum d'un OS. à l'autre
efficacité (vitesse)
compacité du code
contrôle de la taille des buffers
architecture simple à maintenir

1. Liaisons avec l'OS

possibilité de définir un fichier de pages (direct ou random)
ouverture et fermeture de ce fichier
lire une page
écrire une page
demander une extension
(tout le reste est à gérer explicitement)

2. Modèle de données: MAG restreint utilisé dans l'atelier

3. Langage de manipulation:

ADL-C utilisé dans l'atelier (avec léger lifting syntaxique à la NDBS)
+ transactions (begin, abort, commit, undo, redo)

4. *Structures physiques*

chemins représentés par chaînages (1-N) et pointeur (N-1)

rangement random et clustered

identifiant dans chemin (clé ?)

longueur fixe par type d'articles

5. Recherche page libre: par index des espaces libres (1 byte par page)

6. Gestion buffer

taille paramétrable, gestion LRU

7. Format tables internes du schéma: format actuel récupérable ?

Ceci exclut l'usage d'un ISAM (comme dans la version actuelle) pour des raisons de performances et, paradoxalement, de portabilité.

see you soon
JLH

Si la trentaine de lignes reproduites ci-dessus ne recèle aucun mystère pour vous, ami lecteur, rendez vous directement au chapitre suivant, pour la suite de cette palpitante aventure.
A défaut, les paragraphes suivants vous démontreront l'existence de l'effet redox subi par toute spécification au contact d'un étudiant particulièrement réactif.

2.1 Le modèle initiateur

"Limitée aux structures de données et aux primitives de manipulation de celles-ci, l'analyse montre que (...) les Systèmes de Gestion de Données (...) mettent en oeuvre les mêmes concepts, en très petit nombre, et que ce qui les distingue fondamentalement est l'ensemble des restrictions qu'ils imposent sur les assemblages possibles de ces concepts. Le Modèle d'Accès Généralisé (MAG) (...) est un relevé de ces concepts essentiels (...). Ce modèle décrit les structures de données non seulement sous l'angle de la sémantique qu'expriment ces données, mais aussi sous celui des accès dont elles peuvent faire l'objet. En ce sens, ce modèle ne peut être qualifié de conceptuel (...)."

Also sprach JLH.

La réalisation d'un SGBD présuppose l'existence d'un modèle déterminé transcendant les objectifs et les options d'un processus d'implémentation; l'adoption, mutatis mutandi, du MAG apparaît dès lors judicieuse. En effet, le caractère générique et non-conceptuel de ce dernier rencontre pleinement les objectifs détaillés ci-après.

Cependant, LE document de référence ("Eléments méthodologiques" cfr.chapitre 1) s'efforçant d'encourager une certaine unité terminologique, seules les structures d'accès offertes par le MAG trouvèrent grâce aux yeux du grand réconciliateur, l'élagage printanier venu.

En conséquence, malgré l'affirmation sans équivoque aucune du rôle prépondérant joué par ce modèle dans l'histoire de l'atelier et l'atavisme de certains termes repris dans le document de référence, le MAG ne fera pas, ici, l'objet d'une description même succincte. Quant au modèle Entité/Association, si vous en ignorez les préceptes, camarade lecteur stakhanoviste, dénoncez le dissident de l'Institut qui osa vous transmettre l'ouvrage contre-révolutionnaire que vous tenez entre vos menottes; n'aggravez pas vodka: passez le chapitre trois.

2.2 Les objectifs de BIGAM-DBMS

Bareboned but Induced by the Generalized Access Model
Data Base Management System

Ce programme doit faire partie de la classe PEC:

Portabilité
Efficacité
Compacité

Conséquences:

A. Les liaisons avec le système et l'environnement d'exploitation seront aussi réduites que faire se peut.

B. En l'an de grâce 1988, le langage C, fer de lance du système d'exploitation UNIX, reste (las!) seul en lice, après avoir défait tous ses pairs sur l'un ou l'autre prérequis de la classe PEC.

C. L'adoption définitive d'un algorithme, d'une technique de programmation ou d'un type de données ne pourra s'effectuer qu'après démonstration de son efficacité ou innocuité quant aux objectifs poursuivis.

D. L'insertion dans l'atelier base de données de l'Institut d'Informatique doit être réalisable à moindre frais.

2.3 Insertion dans l'atelier

L'atelier ORGA est un composant de l'atelier logiciel ALCIDE. Dans la phase d'élaboration d'une base de données opérationnelle, il apportera une aide précieuse au concepteur d'un Système d'Information. Disposant d'un schéma conceptuel Entité/Association construit et validé dans l'atelier IDA, ce dernier transformera peu à peu une copie du schéma, préalablement transféré dans la base de spécifications d'ORGA, en un schéma physique . Réintroduit dans IDA, ce dernier sera mis en correspondance avec ses origines conceptuelles; une génération de description exécutable peut s'ensuivre.
Un mode de fonctionnement émancipé est également envisageable; pareille apostasie n'est cependant guère conseillée.
Dans le même ordre d'idées qu'au chapitre précédent, le deuxième chapitre du manuel de référence de l'atelier ORGA est gracieusement mis à votre disposition en annexe II.

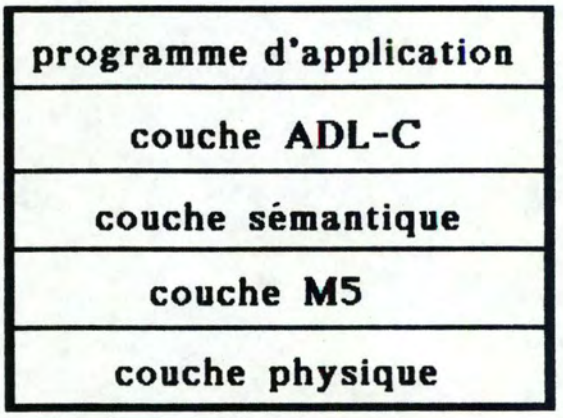


FIGURE 1 : ORGA-88 : HIERARCHIE

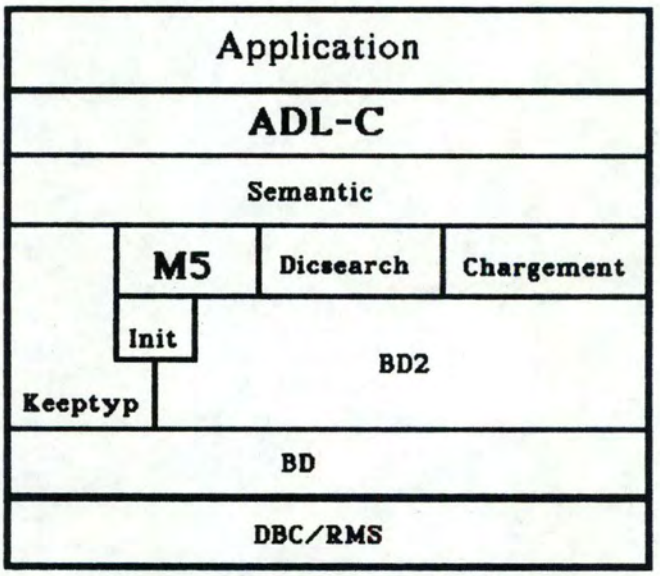


FIGURE 2 : ORGA-88 : DECOUPE MODULAIRE

2.4 Aspects internes de l'atelier

La hiérarchie d'ORGA, dans son implémentation commerciale courante, consacre l'inéluctabilité d'une excision d'importance (cfr. figures 1 & 2).

* LES CINQ COUCHES

Au plus bas niveau, la couche physique, réalisant les accès élémentaires aux données, permet ouverture et fermeture des fichiers, lecture et écriture des enregistrements de typologie fixée par la couche supérieure. Cette dernière, assurant une certaine indépendance de choix du SGBD, porte le nom du modèle qu'elle met en oeuvre: M5. Continuant notre ascension, nous pouvons découvrir la couche sémantique, renfermant l'interface MAG; l'utilisation de ses primitives requiert la domination des concepts du modèle. Elle va, en outre, convertir les schémas MAG en schémas M5 (appelés ainsi parce que le modèle ne connaît que cinq types d'entités). La couche ADL-C fera l'objet d'une description approfondie, dans les paragraphes à venir. L'atelier ORGA, quant à lui, n'est qu'un programme d'application particulier, s'ancrant à la couche supérieure, au même titre que le didactique "commande, ligne, client".

* LES NEUF MODULES

La découpe modulaire des trois niveaux inférieurs trahit la complexité inhérente à l'implémentation des couches cruciales pour les performances de l'atelier. Le module de chargement s'occupe de la lecture des tables de conversion MAG-M5 pour permettre son exploitation par le module semantic ; les opérations de recherche dans ces dernières sont confiées au module dicsearch . Le logique du modèle éponyme est câblée dans M5 . Oublions init et keeptyp . Le module BD2 fournit un ensemble de fonctions élémentaires, compte tenu du courant de chaque fichier; l'interface BD2-DBC est, vous l'aurez deviné, câblée dans l'inoubliable module BD .

Enfin DBC (Lattice) est, n'ayons pas peur des mots, aux SGBD ce que le Canada-Dry est à l'alcool.

On serait tenté d'en déduire que neuf modules pour trois niveaux constituent les douze marches de l'escalier menant à la nécessité du changement. Cependant, pour dissiper toute équivoque, signalons que cette architecture logicielle gothique fut la clef de voûte de la remarquable portabilité et indépendance de l'atelier.

2.5 Aspect prophétique de l'atelier futur

(cfr. Figures 3 & 4)

* LES DEUX COUCHES

Après l'effondrement de l'implémentation mille-feuille, on ne retrouvera dans les décombres que les couches application et ADL-C .

* LES QUATRE MODULES

Au grand regret de l'auteur de ces lignes, il ne sera pas possible de conserver une clarté de programmation exemplaire à tous les niveaux, sous peine de rendre caducs les impératifs plutôt antinomiques de la classe PEC. Après concertation avec les partenaires concernés, une position novatrice fut arrêtée: les modules abscons seront disponibles sous forme de librairie (SGBD.LIB), avec engagement de maintenance logicielle pendant deux ans. L'environnement complet des modules "abordables" (ADL-C) sera communiqué aux responsables de l'atelier; il comprend les sources documentés, les relations de dépendance entre modules et fonctions (cfr. utilitaires MAKE) et les routines de déverminage nécessaires à la validation de toute programmation alternative.

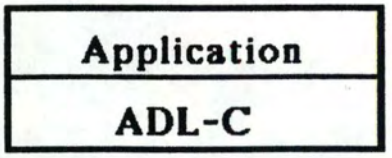


FIGURE 3 : ORGA-89 : HIERARCHIE

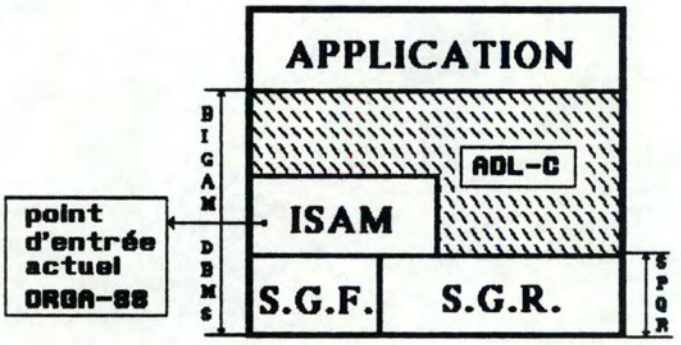


FIGURE 4 : ORGA-88 : DECOUPE MODULAIRE

. Sans anticiper sur le chapitre suivant, voici, en quelques mots, le rôle de chacun des quatre modules:

SGF (Système de Gestion de Fichiers)
Ensemble de fonctions de gestion physique concurrente des fichiers de pages d'entités monotype et des accès par clef.

SGR (Système de Gestion Réseau)
Ensemble de fonctions de gestion physique concurrente des pages d'entités multitypes et des chemins d'accès inter-entités.

ISAM (Indexed Sequential Access Method)
Ensemble de fonctions de gestion conjointe et concurrente des fichiers de pages et des index, par utilisation stricte des modules SGR/SGF (aucun accès physique direct).

ADL-C (Algorithm Description Language)
Définition: Cfr. LES PRIMITIVES, ci-dessous.
Implémentation: stricte utilisation des modules ISAM et SGR.

* LES TYPES DE SCHEMAS

On essaiera, dans la mesure du possible, de ne pas apporter de restrictions contraignantes lors du passage logique-physique; à défaut, celles-ci seront détaillées.

* LES PRIMITIVES

Un langage de programmation, appelé ADL-C (extension du langage C par des instructions du langage ADL) doit permettre l'accès à une base de données à l'aide d'instructions de haut niveau. A cet effet, un pré-processeur transformerait le texte ADL-C en texte C, comprenant des appels à un jeu de procédures dénommées "primitives ADL-C". Ces primitives sont les opérations de base, définies sur les données, et mises à la disposition du programmeur; celles-ci, très succinctement énoncées ci-dessous, constituent la partie émergente de l'iceberg, la frontière de l'espace de liberté.

Base: *Ouverture*
Fermeture

Entité: *Création*
Modification
Suppression
Accès :(premier,suivant):Séquentiel
Par chemin
Par clef dans un chemin
Direct
Chemin: *Modification*

2.6 Les étapes du développement

Les autodafés prononcés à l'encontre de l'architecture actuelle de l'atelier, seraient scandaleux s'ils ne s'appuyaient sur l'expérience pratique d'une solution alternative; qui trop embrase, mal éteint (aphorisme pompier). A cet effet, deux jalons furent posés dans le processus d'implémentation de cette solution:

**** L'ALTERNANCE***

Une transplantation complète de la couche physique (modules: DBC, BD, BD2) par une librairie constituée des modules SGF, ISAM et l'interface ad hoc, s'effectua sans la moindre manifestation de rejet. A la grande satisfaction des parties concernées, l'intervention se déroula en quelques minutes, le temps d'un link. L'atelier supporta parfaitement l'opération et put immédiatement reprendre ses activités. Outre une réduction sensible de l'embonpoint, l'oxygénation périphérique de son anatomie a permis un substantiel accroissement des performances physiques. L'évaluation du facteur d'amélioration par les spécialistes consultés s'établit aux alentours de deux.

Bien que résolument palliative, l'expérience mit en évidence quelques points cruciaux; elle fit la démonstration éclatante de l'indépendance de la couche physique puisque, rappelons-le, pas un iota ne fut changé dans les autres couches. En outre, elle permit de s'affranchir de problèmes résiduels posés par l'utilisation de DBC, dont le côté peu professionnel et vicieux était incompatible avec la crédibilité de l'atelier. Enfin, elle renforça la confiance mutuelle des parties en présence, quant aux chances d'aboutissement du projet.

* ***LE CHANGEMENT***

A l'heure tardive où, au grand dam des lecteurs consciencieux, ces lignes sont bâclées, l'ablation du trio ADL-C, sémantique, physique n'a pu s'effectuer, faute de temps requis par des examens complémentaires. Toutefois, une expérience pilote avec un SGBD commercial (MDBS-III) de très bonne facture, menée récemment par quelques codétenus, permet d'escompter d'ores et déjà un gain décisif en performance pure. Partageant avec MDBS les principaux concepts du modèle réseau, à l'origine de ces excellentes performances, il y a tout naturellement lieu de s'attendre au meilleur. La réalisation du joint entre le module "booster" SGR et ISAM consomma un temps précieux, mais le jeu en valait la chandelle; outre la réconciliation entre les index et les pointeurs, permettant la mise sur orbite de l'accès par clef dans un chemin (s'effectuant à présent en très peu de révolutions du disque), le SGBD "challenger" reste <u>parfaitement</u> <u>compatible</u> avec l'architecture actuelle de l'atelier. En clair, cela signifie que l'alternance et le changement cohabiteront: Rocard battu.

Lors de la défense de ce document, on peut espérer que les navettes effectuées entre le pas de tir du promoteur, le hangar de l'équipe bases de données et la piste de crash du programmeur auront rendu possible le vernissage d'ORGA-89.

* *L'ABANDON*

Il s'agit de l'intégration de l'atelier à l'environnement WINDOWS (Microsoft) qui, malgré l'insistance feutrée du promoteur ("Alors, ça vient ?"), semble totalement hors de portée du programmeur rebelle. D'aucuns, soyons en sûrs, trouveront sans nul doute que la difficulté de cette ultime tâche ne compense pas les risques d'une mutinerie; je les invite à se mettre sans tarder en rapport avec le promoteur. J'implore humblement ce dernier de me pardonner pareille indigence intellectuelle.

* ***LIBRE COMMENTAIRE DES STRUCTURES PHYSIQUES***

Outre un document autographe du maître, que d'aucuns qualifieraient d'apocryphe, n'était la communion d'esprit l'unissant à l'élève, l'énoncé d'un principe fondamental permet de mieux saisir la différence entre l'approche actuelle (cfr.début du chapitre) et la réalisation souhaitée :

L'unicité de l'accès physique lors d'un parcours de chemin.

Il n'entre pas dans les ambitions de ces quelques feuilles d'expliciter les SGBD de type réseau, à l'origine de la réflexion du promoteur de cet ouvrage; quelques graphiques, insérés çà et là dans la prose barbare du chapitre suivant, illustreront quelques fondements de cette technique.

Quant à ceux d'entre vous qui s'inquiéteraient dès à présent de la tournure libertaire de certains propos qui, je le pressens, seront retenus contre moi, la transcription fidèle des souvenirs d'un témoin oculaire (digne de foi) de la dernière scène des spécifications devraient les rassurer: libertin certes, libertaire nullement.

"De même, à la fin du débat, il prit un blanc-seing, le remplit, et le donna à son disciple en disant "prenez mais ne changez pas tout, ceci est mon core livré pour vous". Vous ferez cela en mémoire pour moi".

"L'excès d'objectivité ne pousse pas à l'action. Lorsqu'on a fait l'impossible pour arriver à une décision rationnelle, qu'on a tout pesé et tout envisagé, il vaut mieux fermer son esprit aux nouvelles objections et agir sans hésiter. L'homme qui veut faire un testament parfait meurt intestat".

G.B. Shaw

3. REALISATION

S'il est une appellation à laquelle BIGAM ne peut prétendre, c'est celle de RISC (Reduced Instruction Set Challenger). Soixante fichiers, nonante fonctions, douze mille lignes de code source C: chiffres bruts donnant le frisson quand on entame le délicat chapitre consacré à leur justification; les détenteurs de ce document s'adonnant au passe-temps infamant nommé programmation le comprennent certainement. Quant aux autres, lecteurs intéressés puisque parvenus jusqu'aux portes du troisième chapitre, ils en veulent pour leurs yeux déjà rougis par une prose afflictive. Afin d'amoindrir le risque de somnolence, une approche "journal de bord" fut préférée à toute autre litanie.

Dès le début du projet, il apparut inévitable qu'une connaissance suffisante des objets suivants soit acquise dans les plus brefs délais:

Les systèmes d'exploitation envisagés
Les fichiers
Les modes d'accès

Faute d'un traitement en profondeur, au niveau physique, le risque de problèmes récurrents aux niveaux supérieurs était évidemment plus élevé.

En outre, la terminologie utilisée devait être maniée avec circonspection, sous peine de provoquer quelques quiproquos. Citons, à titre exemplatif, la page: notion héritée de systèmes où pagination rimait avec système d'exploitation, ce vocable n'a pas réellement cours en micro-informatique. Au risque de décevoir certains, elle ne correspond ni au cylindre, ni au secteur, encore moins au "cluster". En ce qui concerne l'une des machines cibles envisagées (PC), on ne voit guère à quelle réalité physique rattacher cette notion et, partant, comment en assurer la portabilité.

3.1 Les fichiers de pages

Si l'on désire ne pas réécrire un système d'exploitation, on considérera la page comme un ensemble de segments d'une structure de données non-typée (binaire), stockée dans un enregistrement appartenant à un fichier. Les primitives offertes par le système d'exploitation seront donc celles usuellement disponibles:

Ouverture et fermeture du fichier
Lecture et écriture d'un enregistrement de position connue

L'implémentation/l'utilisation d'un fichier de pages consistera donc à demander au S.E.:

1. La création d'une entrée dans un répertoire des fichiers.
2. Un pointeur vers la structure de contrôle/d'interface/de description (FIB/FCB/File descriptor/Handle...) créée lors de l'ouverture de ce fichier dont les caractéristiques (longueur d'enregistrement, modes d'accès) lui auront été fournies.
3. D'effectuer les opérations d'écriture/lecture des tampons.
4. La mise à jour du répertoire lors de la fermeture de ce fichier: taille du fichier, table d'allocation des zones (FAT)...

Chacun des points ci-dessus recèle un certain nombre d'inconvénients, plus ou moins sérieux suivant le S.E. considéré, dont la liste mérite d'être commentée. L'impératif de portabilité énoncé dans les chapitres précédents nous impose de n'en négliger aucun. Il faut cependant remarquer que ces inconvénients/avantages devraient, en toute logique, guider le choix d'un S.E.; il est navrant de constater que l'utilisation d'algorithmes de gestion de fichiers simples et robustes soit bridée par un S.E. dont les primitives évoluées manquent de souplesse.

Inconvénients

1. *Nombre limité de fichiers par répertoire.*
2. *Nombre limité de descripteurs simultanés,*
   Manque de gestion dynamique de ces descripteurs,
   Manque de caractéristiques (modes d'accès).
3. *Gestion des tampons n'assurant pas l'intégrité des données,*
   Faiblesses de verrouillage,
   Faiblesses de mise à jour des répertoires.
4. *Condition sine qua non d'intégrité des données*
   (FAT,Size...).

Trois méthodes d'évitement de ces inconvénients s'affrontent:

Truffer le S.E. de routines-rustines.
Minimiser les recours au S.E.
Contourner ces limitations sans violer la portabilité

La première technique conduit rapidement à un S.E. plus miné qu'une plage de Normandie en 1944; en outre, la portabilité n'y est plus. La deuxième est efficace mais peu sélective; on va en effet s'interdire, sans distinction, l'ensemble des fonctions intéressantes du S.E, le réduisant à une table des matières des pistes/secteurs d'enregistrements binaires de taille fixe. De plus, elle ne solutionne que les inconvénients des points 1 et 2, à l'exception des caractéristiques (modes d'accès).

La troisième n'est pas aussi complexe qu'on pourrait le supposer. Elle permet, nous le développerons par la suite, certains enrichissements, tant du point de vue de l'intégrité des données que des performances.

3.2 Civilisation standardisée du S.E.

Serait-il possible de développer des extensions standards, pour S.E. hétéroclites, apportant une réponse significative aux maux précités ?
Reprenons, honnêtement, chacun des points soulevés:

* LES DESCRIPTEURS DE FICHIERS

Le nombre limité de fichiers par répertoire et de descripteurs simultanément actifs ou dynamiquement gérés, suggère la possibilité de regroupement de structures de données différentes, au sein d'un enregistrement logique unique: la page.(Rappel: celle-ci, telle que définie en 3.1, n'est pas une structure physique; seul l'enregistrement dont les caractéristiques ont été fournies au S.E. lors de l'ouverture du fichier, peut prétendre à cette appellation). La consolidation des index (module SGF, nous y viendrons) en constitue une excellente illustration; expérimentée avec succès, la méthode sera dès lors étendue aux entités (module SGR). Mais n'anticipons pas...

Techniquement, quels sont les facteurs influençant significativement le temps d'accès à un enregistrement, de position relative connue, sur un disque ?

- Le temps requis par les opérations d'ouverture du fichier
- Le temps de recherche et de positionnement.
- Le temps de lecture/écriture du nombre d'unités de transfert requis pour l'obtention d'un enregistrement de cette longueur. Cela dépend évidemment de la politique d'allocation ("cluster"), d'alignement et de disposition ("interleave") du S.E.
- Le temps de transfert entre tampons, dépendant également de leurs tailles respectives.
- Le temps nécessaire aux opérations de clôture du fichier.

L'amalgame de structures de données différentes au sein d'un même fichier de pages permet l'unicité des opérations d'ouverture et de clôture, tout en n'utilisant qu'un seul descripteur; cependant, (à moins de supplanter le S.E.) ce seront là les seules durées économisées. Cette solution ingénieuse n'en recèle pas moins quelques inconvénients:

- Opérations d'extraction/insertion des diverses structures de données plus complexes
- Maintien de l'intégrité des données plus délicat
- Contrôle de croissance ardu et non-sélectif
- Reconstruction après incident plus sophistiquée

La raréfaction des descripteurs de fichiers pourrait-elle, seule, justifier le groupement de structures de données multitypes, et le surcroît de traitement en résultant?

Il faut résolument répondre par la négative. Au stade de réalisation décrit dans les lignes environnantes (cfr. chapitre 2, 2.5 - l'alternance), les structures particulières mises en oeuvre dans SGF permettent l'adoption d'une technique mixte, plus avantageuse. Cependant, l'étape suivante du développement, basée sur les SGBD de type réseau, rend obligatoire la gestion des pages d'entités multitypes, SGR pour les intimes (cfr. chapitre 2, 2.5 - le changement). En anticipant quelque peu la démarche de réalisation des modules SGR et ISAM, signalons sans plus attendre que SGF limitait le regroupement aux clefs d'accès, SGR le généralise aux entités. La technique mixte mise en oeuvre dans SGF procède de deux principes:

- *Une gestion dynamique (LRU), automatique et transparente des descripteurs de fichiers de pages.*
- *Une gestion transparente de certains regroupements intéressants de structures de données différentes, en un seul fichier.*

* L'ACCES CONCURRENT

Les fichiers de pages peuvent être accédés, sous hypothèse de concurrence parfaite, par un nombre illimité de processus de consultation et de modification; les contentions sont contrôlées au moyen de verrous. Certains S.E. ne supportent pas le verrouillage en écriture. Le MS-DOS 3.1, par exemple, n'accepte que les verrous exclusifs. Dans le même ordre d'idées, tous les S.E. ne permettent pas la mise en file d'attente ou la réitération automatique des demandes de verrouillage. Le MS-DOS 3.1, encore lui, se contente d'un essai toutes les secondes à concurrence de dix secondes. Des stratégies d'évitement et d'amélioration ont été mises en place. Cependant, l'objectif de portabilité ne pourra être maintenu à ce propos. Les modifications à apporter seront strictement fonction du compilateur et du système d'exploitation, ainsi que de leurs versions respectives. Les adaptations à Lattice 3.0, Microsoft 4.0/5.0 et UNIX System V release 3 sont, à ce jour, disponibles.

* GESTION DES TAMPONS ET REPERTOIRES

Peu de S.E. mettent à jour, avant clôture du fichier, le répertoire et les tables d'occupation connexes. Certains d'entre eux fournissent explicitement une fonction (ex. sync () sous UNIX) de mise à jour; à défaut, différentes astuces ont été développées dans de nombreux programmes où l'intégrité des données ne souffrait guère pareille latence. L'extension d'un fichier de pages par blocs (et non par pages) couplée à une mise à jour ponctuelle des répertoires suffit à la résolution de ce problème.

3.3 Les dessous d'ISAM

Une saine gestion de fichiers nous permet d'envisager la construction d'un ISAM; nous y reviendrons. Les structures à mettre en place doivent, si l'on désire affronter d'illustres SGBD (relationnels, pour ne pas les nommer) offrir souplesse, performance, robustesse et intégrité. Tout un programme...

* INTRODUCTION

Parmi les structures dynamiques de données les plus usitées, les arbres occupent une place de choix. Si les caractéristiques de certaines espèces sont bien connues et firent l'objet de nombreuses études dans le début des années 70, ces constructions sont aujourd'hui au centre de bon nombre de programmes et d'algorithmes où leur utilisation s'est avérée payante. Une généalogie des structures dynamiques arborescentes de données n'entrant pas dans le cadre de ce document, nous nous limiterons à la description de l'espèce sous-jacente aux mécanismes de gestion dynamique de l'espace disque implémentés dans SPQR: les CAB-3 (Concurrently accessed B-trees).

* LES B-TREES

Les arbres binaires sont sans conteste l'espèce la plus intéressante pour la recherche algorithmique; des algorithmes de recherche, d'insertion et de suppression, ainsi que différents critères de contrôle d'extension ont été publiés et commentés. Cependant, la hiérarchie des mémoires et le volume des liens à tisser entre les données limitent l'utilisation des arbres binaires à l'ordonnancement de données simples en mémoire centrale. La gestion de plusieurs centaines de Mb, à haut degré d'interconnexion, disqualifie les arbres binaires où le parcours de chaque noeud se solderait par un (ou plusieurs) accès, à une mémoire périphérique. Une solution consiste à grouper un certain nombre de noeuds en unités d'accès élémentaire; imposons cette condition aux pages définies ci-dessus. On peut alors escompter une réduction très sensible du temps nécessaire à l'obtention du noeud désiré, pour peu qu'un critère de contrôle de croissance soit implémenté. Si l'on impose que toute page contienne entre n et 2*n noeuds (racine exceptée), alors le nombre d'accès nécessaires (lecture/écriture page) à l'obtention d'un noeud parmi un ensemble de P noeuds (cas le plus défavorable) est de l'ordre de log base n (P).

Abandonnons le vocable noeud au profit de celui de valeur d'attribut; cela revient à dire que la recherche d'une valeur d'attribut (clef d'accès) dans un ensemble de 300 Mega valeurs d'attributs organisées en pages de 50 à 100, requiert de l'ordre de 5 accès élémentaires. Ces structures sont appelées des B-trees et l'on trouvera dans (6,7,8) une étude comparative de leurs propriétés.

* **LES CAB-3**

Les caractéristiques intéressantes des B-trees ne peuvent être améliorées et surtout maintenues sans effort. Schématiquement, un certain équilibrage des arbres s'avère nécessaire; ce dernier permet de ramener le nombre moyen d'accès en deçà de log base n (P/2). Ce rééquilibrage peut être considéré comme un processus parallèle intervenant de façon asynchrone lors de l'apparition d'une condition de pénalisation du nombre d'accès. L'une des critiques les plus fréquemment adressées à l'encontre des B-trees concerne leur indisponibilité pendant de telles opérations. Paradoxalement, une des conditions de fonctionnement satisfaisant des arbres en général réside dans la surveillance des statistiques d'accès aux noeuds, et le déclenchement d'actions appropriées. Une dégénérescence des structures due à l'apparition de scénarii d'accès particuliers (ratios insertion/consultation, suppression/consultation hors normes) est extrêmement difficile à gérer simplement.

L'originalité des CAB-3 (Concurrently Accessed B-Trees) réside dans une structure algorithmique permettant l'accès simultané de processus consultants et modifiants, au moindre coût de verrouillage. A titre d'illustration, cet algorithme assure l'éclatement d'un noeud (extension de l'arbre) concurremment avec un nombre quelconque de consultations simultanées de ce même noeud. Moyennant de légères adaptations aux environnements concernés (MS-DOS/OS 2/Unix V/4.2 BSD/Xenix...), SPQR constitue en fait un système de gestion de pages multi-processus/multi-utilisateurs où les impératifs de verrouillage sont minimisés.

3.4 La dot de SPQR

Plusieurs catégories de personnes sont commises à la lecture de mémoires universitaires. Certains, comblés, découvrent chaque année après les lectures faciles et souvent balnéaires, le fleuve noir des mémorands que la sédentarité estivale rendit prolixes. D'autres, pressentis, souffriront le martyre, le temps de l'utilisation du travail réalisé venu; s'il n'est promptement mis à la disposition de la recherche ou de l'assistanat, l'étudiant risque en effet de faire cruellement défaut lors de l'interprétation de sa partition. Dédaignant la prolifique dégaine d'un programmeur COBOL nourri au ginseng, les paragraphes suivants s'attacheront à satisfaire les aspirations légitimes de la première catégorie de lecteurs évoquée. Quoique réductrice et fort peu représentative de l'effort entrepris, cette démarche évite l'insertion de 120 pages du "Vade-mecum du programmeur BIGAM" (épuisé, réédition prochaine) qui n'ont même pas l'excuse d'être pléthoriques.
Chacune des fonctions de SPQR y fait en effet l'objet d'un commentaire en sept points: nom, déclaration C, description en langage naturel, erreurs possibles, limitations, exemple d'utilisation en source C, et références croisées à consulter.
En conséquence, vous ne trouverez ci-après qu'un résumé succinct du patrimoine fonctionnel de ce double mariage morganatique ISAM et ADL-C, SPQR et ADL-C.

* LIBRE TERMINOLOGIE

Il existe dans l'argutie informatique des mots dont l'acception est à ce point vaste qu'ils en deviennent obligés. Ainsi, le vocable "clef", dont le côté galvaudé fut à juste titre relevé dans (2) doit, une fois encore, subir les outrages d'une définition.

On appellera accès par clef aux pages un mécanisme permettant l'organisation puis l'obtention d'une page dans un fichier, sur base de combinaisons non nécessairement identifiantes d'un certain nombre de segments issus de cette page. Par analogie, ces combinaisons seront appelées clefs.

La librairie SPQR (rappel: SGF+SGR) supporte quelques dix types de clefs usuels et permet facilement l'adjonction de types particuliers; compression automatique (préfixe et suffixe) et KLUT (Key Look-Up Table) font également partie du jeu de fonctions. La puissance des CAB-3 est certainement mise en exergue par cette souplesse peu coutumière.

A l'instar des pages, ces clefs sont organisées en collections ordonnées , appelées trousseaux. Conformément au dogme des B-Trees, ces trousseaux seront empagés (néologisme à la page) en fichiers appelés index. Un seul index peut contenir une trentaine de trousseaux différents, cependant afférents au même fichier de pages. Les algorithmes de contention et les structures mises en place afin de permettre le parcours de ces index par un nombre quelconque de processus concurrents feront l'objet d'une description sommaire en fin de chapitre. Enfin, toute entité sera réputée identifiable par un mécanisme d'accès à la page qui l'accueille (extérieur aux segments la composant) appelé adresse d'une page.

Au risque d'être qualifiée de SPQRieuse, cette terminologie se veut autocratique et sans appel; quand bien même les anglo-saxonneries sont à l'informatique européenne ce que le VSOP est au cognac, pareilles lettres de noblesse ne glaneraient que flétrissures à se commettre en compagnie d'un SGBD roturier.

* LA CORBEILLE SGF

On trouvera ci-après une simple évocation des diverses fonctions intéressantes du module SGF , disponibles dans la version actuelle de la librairie SPQR.

Rappel: SGF n'est qu'un ensemble de fonctions de gestion physique concurrente des fichiers de pages d'entité monotype et des accès indexés.

Trousseaux, Index & Clefs

- Création d'un index (fichier); de nombreux paramètres sont requis.
- Création d'un trousseau afférent à un index.
- Insertion d'une valeur de clef dans un trousseau; le mode de décomposition des noeuds saturés des CAB-3 doit être déterminé.
- Suppression inconditionnelle d'une valeur de clef dans un trousseau.
- Suppression d'une valeur de clef dans un trousseau, avec vérification d'intégrité par rapport à la page référencée.
- Recherche dans un trousseau d'une valeur de clef déterminée.
- Recherche dans un trousseau d'une valeur supérieure ou égale à une valeur de clef déterminée.
- Recherche dans un trousseau d'une valeur strictement supérieure à une valeur de clef déterminée.
- Recherche dans un trousseau d'une valeur inférieure ou égale à une valeur de clef déterminée.
- Recherche dans un trousseau d'une valeur strictement inférieure à une valeur de clef déterminée.

- Obtention de la première valeur de clef d'un trousseau.
- Obtention de la valeur de clef suivante, dans l'ordre défini sur un trousseau préalablement objet de recherches.
- Obtention de la valeur de clef précédente, dans l'ordre défini sur un trousseau préalablement objet de recherches.
- Obtention de la dernière valeur de clef d'un trousseau.

- Indication de la population des trousseaux.

Pages

- Lecture d'une page d'adresse connue.
- Ecriture d'une page d'adresse connue.
- Suppression d'une page d'adresse connue.
- Indication de la population des pages.
- Recherche d'une page vierge.

Fichiers

- Ouverture locale ou partagée d'un quelconque fichier.
- Fermeture locale ou globale d'un quelconque fichier.
- Création d'un fichier dont les paramètres de taille, de mode d'extension et d'accès doivent être spécifiés.

Système

- Allocation des zones de mémoire nécessaires (tampons, descripteurs de fichiers, noeuds).
- Pose ou retrait d'un verrou sur une page.

* LE BOUQUET SGR

Les impératifs du modèle "réseau" conduisent à une gestion performante des chemins d'accès inter-entités. L'établissement d'une relation directe , sans index, entre ces dernières présuppose l'utilisation de références au sein de la structure d'accueil d'une entité. La disponibilité de SGF nous pousse tout naturellement à envisager l'adaptation à cet effet de la structure de page (cfr.figures 5 à 10). A l'instar des trousseaux et autres index, la page se verra modelée en structure d'hébergement de plusieurs entités de types différents ; toute entité sera réputée identifiable par:

- l'adresse de la page qui l'héberge (cfr.SGF) et
- un mécanisme d'accès au segment la contenant.

Cette identification sera désignée sous le vocable de position d'une entité, utilisée comme référence dans un chemin d'accès.

SGR donne la dernière touche à l'effort de conglomération entrepris dans SGF qui, par ailleurs, subit sans encombre les modifications découlant de la structure complexe des pages d'entités multitypes. De ce fait, bien que faisant partie stricto sensu de SGR, les adaptations de SGF sont transparentes et seules cinq fonctions supplémentaires devront faire l'objet d'une hâtive description.

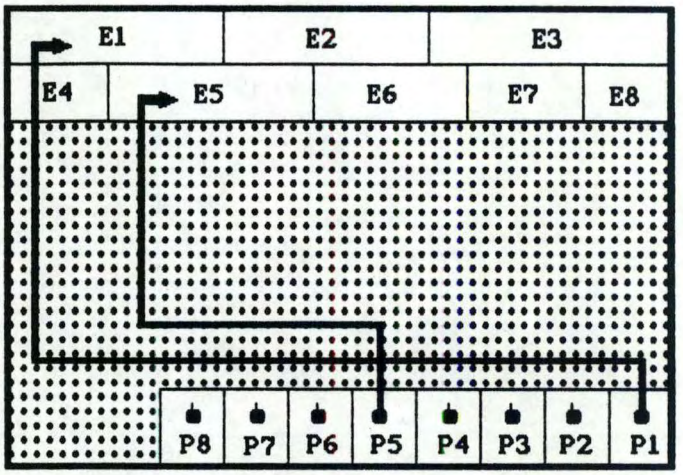


FIGURE 5 : PAGES "SGF"

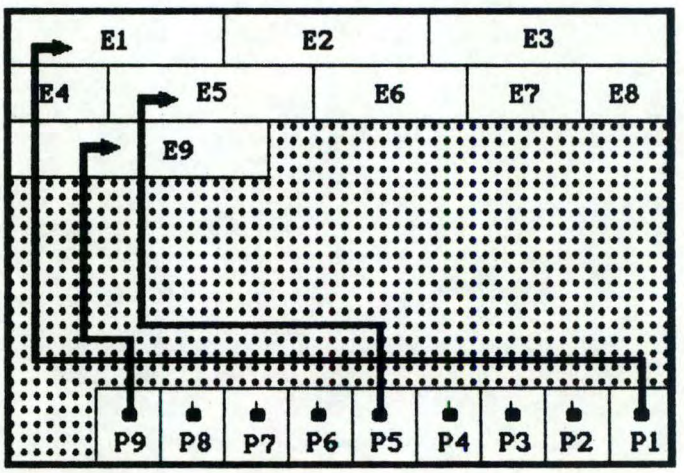


FIGURE 6 : INSERTION D'UNE ENTITE

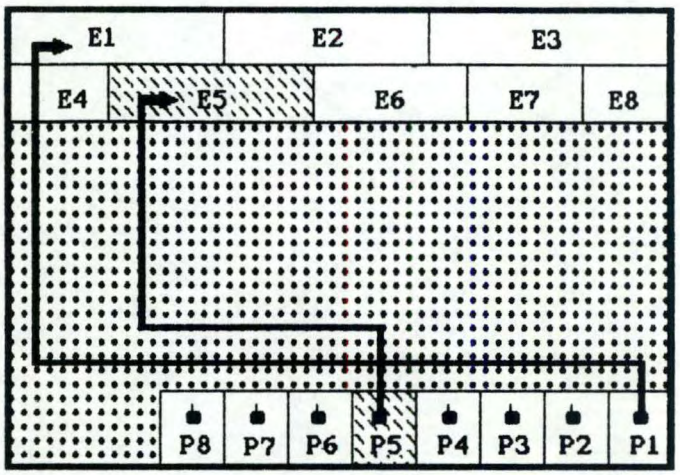


FIGURE 7 : SUPPRESSION D'UNE ENTITE

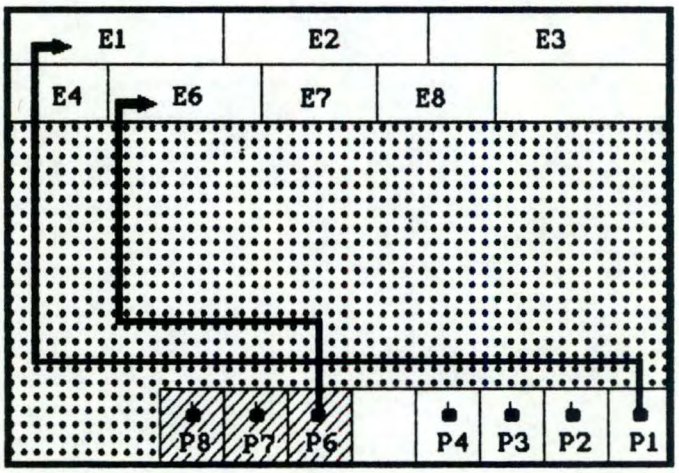


FIGURE 8 : COMPACTAGE

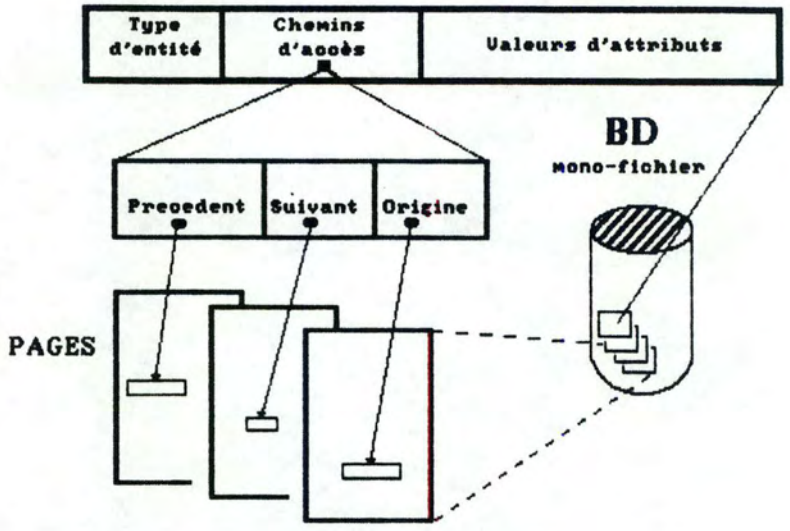


FIGURE 9 : ACCES PAR CHEMIN : CIBLES

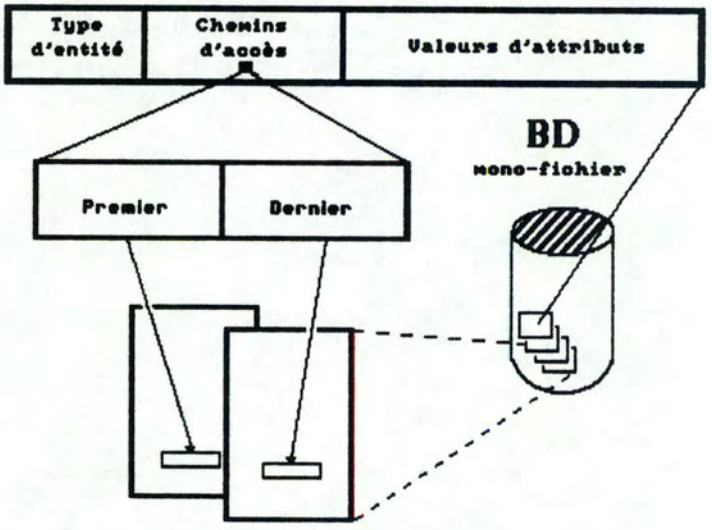


FIGURE 10 : ACCES PAR CHEMIN : ORIGINE

- Insertion d'un segment dans une page d'adresse connue.
- Extraction d'un segment d'une page d'adresse connue.
- Suppression d'un segment d'une page d'adresse connue.
- Indication du peuplement d'une page d'adresse connue.
- Recherche paramétrable d'un segment vierge.

3.5 ISAM

Puisque la majeure partie des problèmes inhérents aux chemins d'accès et à l'accès par clef ont été traités, sainement et en profondeur dès le plus bas niveau, la réalisation d'un ISAM (Indexed Sequential Access Method) s'avère possible et payante (cfr. chapitre 2, 2.5 - l'alternance). Multi-clés, multi-utilisateurs sont les caractéristiques requises à ce niveau. Une librairie comprenant une trentaine de fonctions de haut niveau (dont six seulement utilisées) donne toute satisfaction au sein d'ORGA-88 (cfr. figure 4).

Cependant, le lecteur vigilant pourrait s'interroger avec raison sur les avantages d'un ISAM; si l'on ne peut réfuter qu'il soit théoriquement possible de s'en passer, on pourra tout de moins objecter qu'il devrait permettre de valider efficacement bon nombre d'options techniques de SPQR. En outre, l'ISAM envisagé ci-après crée un conglomérat logique entre la gestion des pages et des accès par clef qui ne peut être que bénéfique pour les niveaux supérieurs. A cet égard, il faut remarquer que l'appellation n'étant guère contrôlée, certains logiciels, bien qu'offrant nettement moins que SGF, s'arrogent avec sérénité cette étiquette d'ISAM.

Une brève description des fonctions importantes vous est proposée ci-dessous; certains termes (ex. fichier de configuration) ne seront détaillés qu'en fin de chapitre (cfr.3.7).

- Insertion d'une page et mise à jour des trousseaux et index concernés.
- Insertion d'un segment et mise à jour des trousseaux et index concernés.
- Suppression d'une page et mise à jour des trousseaux et index concernés.
- Suppression d'un segment et mise à jour des trousseaux et index concernés.

- Mise à jour d'une page, des trousseaux et index concernés.
- Mise à jour d'un segment, des trousseaux et index concernés.

- Construction d'une valeur comparable avec une valeur de clef déterminée.

- Lecture d'une page dont la valeur d'une clef d'accès égale une valeur déterminée.
- Lecture d'une page dont la valeur d'une clef d'accès est supérieure ou égale à une valeur déterminée.
- Lecture d'une page dont la valeur d'une clef d'accès est inférieure ou égale à une valeur déterminée.

- Lecture de la première page dont un préfixe de la valeur d'une clef d'accès égale une valeur déterminée.
- Lecture de la page suivante dans une séquence préalablement déterminée par un préfixe d'une clef d'accès.
- Lecture de la page précédente dans une séquence préalablement déterminée par un préfixe d'une clef d'accès.
- Lecture de la dernière page dont un préfixe de la valeur d'une clef d'accès égale une valeur déterminée.

- Lecture de la première page dans une séquence d'ordre physique ou déterminé par une clef d'accès.
- Lecture de la page suivante dans une séquence d'ordre physique ou déterminé par une clef d'accès.
- Lecture de la page précédente dans une séquence d'ordre physique ou déterminé par une clef d'accès.
- Lecture de la dernière page dans une séquence d'ordre déterminé par une clef d'accès.
- Extraction d'un segment.
- Obtention de la longueur exacte d'un segment.

Fichiers

- Création de tous les fichiers définis par un fichier de configuration.
- Ouverture de tous les fichiers définis par un fichier de configuration.
- Fermeture de tous les fichiers de pages et index.

Système

- Allocation dynamique de courants.
- Désallocation dynamique de courants.
- Sélection de courant.
- Validation de courant.
- Gestion du verrouillage des pages.

Extensions (en cours)

Remarque: il s'agit d'envisager la possibilité de changements dynamiques des structures de données et d'accès, travail confié au mémorand nouveau.

- Allocation des zones mémoire nécessaires au mode de création dynamique.
- Création dynamique de structures ISAM.
- Ouverture d'une structure dynamique ISAM complète (fichier de pages, index).
- Fermeture d'une structure dynamique ISAM complète.
- Reconstruction d'une nouvelle structure dynamique ISAM complète, à partir d'une structure existante.

3.6 ADL-C

Tout SGBD se doit d'offrir au programmeur un modèle de données , un ensemble de primitives permettant de gérer ces données et d'y accéder, ainsi que des règles d'enchaînement de ces primitives. BIGAM offre à la couche "application" une couche unique de primitives; constituées de fonctions rédigées en langage C, ces dernières sont appelées primitives ADL-C (cfr.2.5 - les primitives).

Le modèle de données est un sous-ensemble de celui évoqué au chapitre précédent (cfr.éléments méthodologiques); quelques lignes suffisent à décrire ses principales caractéristiques (cfr. figure 11).

- *Les attributs ne peuvent être répétitifs.*

- *Les types de chemins seront de classe fonctionnelle 1-1, 1-N ou N-1 ; leurs inverses seront d'office créés.*
- *Les origines et les cibles des chemins seront multitypes d'entités.*

- *Un ou plusieurs attributs peuvent constituer une ou plusieurs clefs d'accès.*
- *Il n'y a pas de restrictions particulières au nombre de clefs d'accès.*
- *Toutes les clefs d'accès peuvent l'être au sein d'un type de chemins.*

- *On n'admet, comme identifiant, qu'une combinaison d'attributs attachés à la même entité. Cette restriction pourrait facilement être levée, mais la surveillance de cette contrainte d'intégrité n'incombant pas nécessairement au SGBD, une recherche théorique s'impose.*
- *Plusieurs identifiants par type d'entité sont admis.*

- *Les types d'entités et de chemins, les attributs ont tous des noms distincts.*

- *Quelle que soit la taille de page choisie, toute page peut bénéficier d'une extension transparente ; toutefois elle ne pourra excéder 63 Kb.*

- *La taille maximum d'un fichier ne peut excéder 4 Gb.*

- *La longueur maximale d'une clef d'accès ou d'un identifiant, hors compression est limitée à 240 bytes.*

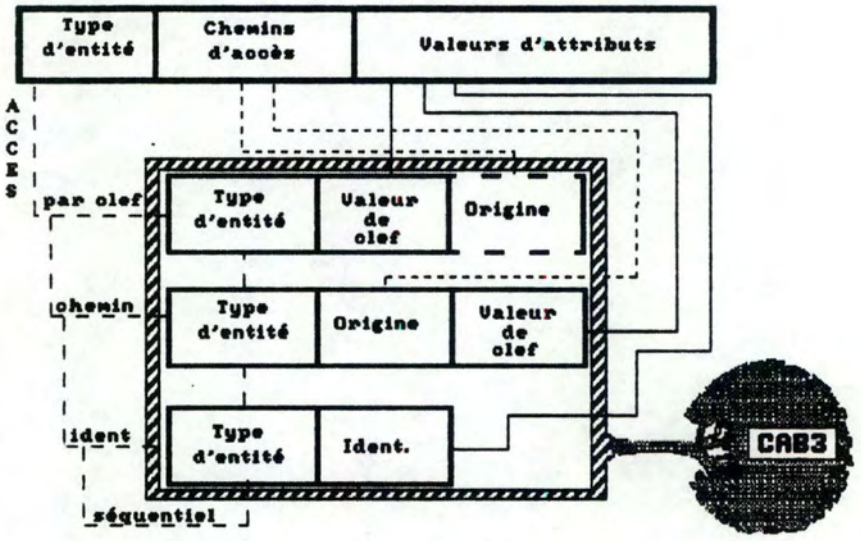


FIGURE 11 : CLEFS ET IDENTIFIANTS

Les règles d'enchaînement des primitives sont réduites au minimum, de par la notion explicite de référence à un article. Elle permet à l'application d'ignorer les "courants" dans les différentes séquences de la base de données, tant en mise à jour qu'en accès. BIGAM prend naturellement en charge la gestion des positionnements; près de deux cents valeurs différentes du témoin de fonctionnement du SGBD, câblées dans SPQR et ISAM, permettent une gestion très fine des problèmes et situations délicates lors de l'exécution de l'application. On retiendra essentiellement les règles élémentaires d'ouverture avant usage de la base de données et de fermeture avant terminaison du programme d'application.

*** L'INTERFACE APPLICATION/ADL-C**

Référencer une entité s'effectue sous la forme d'une variable C dont le contenu, s'il est non vide, désigne un article auquel le programme a accédé antérieurement dans la base de données. Un programme n'est pas limité quant au nombre de variables "de référence"; leur manipulation doit cependant obéir à de strictes règles, sous peine d'incohérence comportementale majeure de BIGAM. En effet, dans certaines situations, une variable réputée contenir une référence contiendra en fait un pointeur vers cette dernière, mémorisée au sein de BIGAM. On comprendra dès lors que toute comparaison, assignation, affectation ou désaffection directe de ces variables soit prohibée; ces opérations seront confiées au SGBD, moyennant disponibilité des fonctions adéquates. Le compilateur de schéma (cfr. figure 12) produira un fichier de déclarations en source C, destiné au programmeur d'application; constantes, références, variables et structures propres au schéma compilé seront ainsi disponibles pour garnir ou réceptionner les divers paramètres des fonctions ADL-C. L'exécution de chaque primitive-fonction, livrera, entre autres, un témoin de fonctionnement de BIGAM; on ne transmettra à la couche supérieure que les témoins pertinents, d'analyse locale impossible et de sémantique compatible avec les concepts manipulés au sein d'une couche application. Une description synoptique de ces fonctions vous est proposée ci-après.

* FONCTIONS ADL-C

Bases de Données

- Ouverture d'une base de données.
- Fermeture d'une base de données.

Accès séquentiel: ordre indéterminé

- Accès à la première entité de type déterminé.
- Accès à l'entité de type déterminé, suivant une entité déterminée du même type.

Accès par chemin

- Accès à la première entité de type déterminé, cible d'une entité déterminée dans un chemin d'accès spécifié.
- Accès à la dernière entité de type déterminé, cible d'une entité déterminée dans un chemin d'accès spécifié.
- Accès à l'entité de type déterminé suivant une entité déterminée, cible d'une entité déterminée dans un chemin d'accès spécifié.
- Accès à l'entité de type déterminé précédant une entité déterminée, cible d'une entité déterminée dans un chemin d'accès spécifié.

Accès par clef

- Accès à la première entité de type déterminé dont la valeur d'une clef spécifiée égale une valeur déterminée.
- Accès à l'entité de type déterminé suivant une entité déterminée, dont la valeur d'une clef spécifiée égale une valeur déterminée.
- Accès à l'entité de type déterminé précédant une entité déterminée, dont la valeur d'une clef spécifiée égale une valeur déterminée.
- Accès à la dernière entité de type déterminé dont la valeur d'une clef spécifiée égale une valeur déterminée.

Accès par clef dans un chemin

- Semblable à l'accès par chemin sauf que l'on ne considère que les entités dont la valeur d'une clef spécifiée égale une valeur déterminée.

Accès direct

- Accès à l'entité de référence déterminée.

Autres accès

Une simple (re)lecture des caractéristiques d'ISAM permet de saisir la richesse des accès disponibles à ce niveau; on pourra ainsi envisager, quasi immédiatement , l'accès trié et, partant, trié dans un chemin. Toutefois, pareille extension requiert une description précise et l'approbation des responsables de l'atelier; peut-être sera-ce là chose faite lors de la défense de ce document.

Modification des données

- Création d'une entité de type déterminé et insertion de celle-ci dans tous les chemins où elle est cible obligatoire d'entités déterminées.
- Suppression d'une entité déterminée et des entités récursivement cibles obligatoires.
- Mise à jour de valeurs d'attributs d'une entité déterminée.
- Insertion d'une entité déterminée comme cible d'une entité spécifiée dans un chemin d'accès déterminé.
- Retrait d'une entité déterminée d'un chemin d'accès spécifié.

Manipulations des variables de référence et divers

- Confère à une variable le statut de référence.
- Retire à une variable le statut de référence.
- Annulation d'une référence.
- Assignation de deux références.
- Test d'égalité de deux références.
- Test d'annulation d'une référence.

- Recopiage des valeurs d'attributs d'une variable de référence dans une autre.
- Recopiage des valeurs d'attributs d'une variable de référence dans une variable d'attribut.
- Détypage d'une variable de référence.

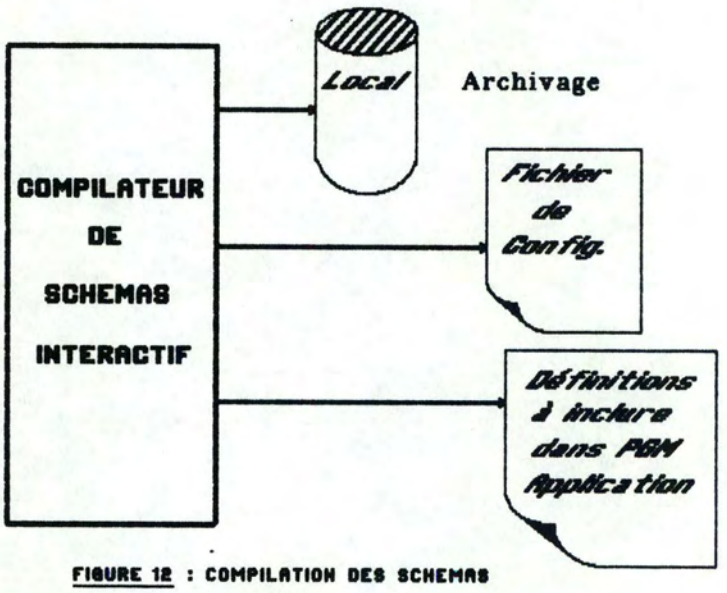


FIGURE 12 : COMPILATION DES SCHEMAS

3.7 Compléments d'explication

Une spécification formelle rigoureuse des divers modules évoqués requerrait, en toute honnêteté, trois cents pages indigestes, certes, mais complètes et cohérentes. Présentées de façon informelle mais progressive, les spécifications et fonctionnalités de BIGAM ne prétendent nullement s'y substituer. De nombreux points sont obscurs et mériteraient dissertation, au détriment du lecteur dont on heurterait ainsi la prévisibilité: il s'attend à un mémoire d'étudiant, alias moine copiste enlumineur, et il se trouverait confronté au manuel technique d'exploitation d'un SGBD parchemin blasphématoire. Le ratio d'entêtement du lecteur (nombre de pages lues/intensité des céphalées) ne s'établirait pas bien haut. Paradoxalement, la plupart des points passés sous silence dans cette optique lénifiante, permettraient d'appréhender mieux encore le travail de l'auteur de ces lignes; les paragraphes suivant s'y attellent de ce pas.

**** TYPOLOGIE DES CLEFS D'ACCES AUX PAGES***

SPQR (et donc BIGAM) permet l'utilisation de clefs:

- *identifiantes (vérification automatique en cas d'insertion) ou non*
- *de longueur fixe ou variable*
- *compulsées de droite à gauche ou de gauche à droite*
- *de représentation binaire, caractère, entière, simple et flottante*
- *avec compression des préfixes et/ou suffixes*
- *transposées par une "Key Look-Up Table byte to byte"*
- *composées de segments de pages épars, de position fixe ou relative (via un délimiteur de segment); ceux-ci peuvent également faire l'objet de transpositions avant assemblage en clef*
- *d'insertion non nécessairement systématique dans les trousseaux (détection de valeurs nulles).*

* *LE VERROUILLAGE INGRAT*

Les fichiers de pages et les index doivent, en cas de concurrence d'accès, faire l'objet de verrouillages.

Les noeuds des CAB-3 sont hébergés dans les pages d'un index; au niveau du système d'exploitation, il s'agit d'un enregistrement. Si deux processus désirent mettre simultanément à jour le même noeud, il faut en faire patienter un. Cependant, les CAB-3 permettent à un nombre quelconque de processus de simple recherche de n'être jamais postposés , s'ils s'exécutent concurremment à un seul processus de mise à jour. En outre, toute situation d'interblocage étant impossible, de par l'algorithme utilisé, le verrouillage des noeuds est transparent et ne devra pas être géré au sein du programme d'application.

En ce qui concerne le verrouillage des pages d'entités, on ne peut bien évidemment gérer entièrement les accès concurrents au sein de BIGAM; il s'agit de transactions. La fonction de verrouillage disponible dans le module ISAM s'inspire du protocole classique "en deux phases". Schématiquement, ce dernier postule qu'un processus ne devra jamais déverrouiller une page avant d'avoir pu verrouiller l'ensemble des pages nécessaires à l'accomplissement d'une transaction. Concrètement, le programme d'application manifeste son désir d'entamer une transaction en appelant au verrouillage préalable de toute page susceptible d'accès dans la suite des opérations (phase 1). Ensuite, la fin d'une transaction se matérialisera par le retrait de tous les verrous posés (phase 2). Ce protocole est cependant trop restrictif; il peut être intéressant de suspendre temporairement le prérequis de verrouillage, à des fins de simple consultation (aucune mise à jour) d'autres pages. Il n'y aura, bien sûr, aucun retrait de verrous posés et l'on pourra reprendre le verrouillage préalable à tout instant.

Aucune stratégie ne permet cependant l'évitement de toute interaction intempestive; citons, à titre documentaire, le processus mettant à jour une page, alors qu'un autre, parcourt un index et se prépare à poser un verrou sur la même page. Deux types de changements requièrent la mise au point d'une routine d'exception au sein du deuxième processus: la page a été supprimée par le premier, ou une valeur de clef a subi de significatives modifications; heureusement ces collisions seront détectées par BIGAM et le processus en sera informé par un témoin de fonctionnement spécial. Dans la majeure partie des cas, il lui suffira de répéter l'opération suspecte.

Quelques règles simples d'écriture d'applications dans un environnement multi-processus doivent être observées. La mise à jour et la suppression d'une page nécessitent la détention d'un verrou posé sur cette dernière. Or, il faut à tout prix éviter de conserver ceux-ci pendant une période prolongée; on envisagera dès lors l'un ou l'autre modus vivendi. Premier scénario possible d'une modification de page:

1. Lecture de la page et duplication des tampons en mémoire centrale (T1,T2).

2. Opérations de modifications dans le duplicata (T2)

3. Demande de verrouillage préalable à tout accès.

4. Relecture de la page dans un troisième tampon (T3)

5. S'il n'y a pas attribution du verrou lors de cette relecture ou si la comparaison des tampons T1 et T3 révèle une discordance, il s'agit là d'une exception à traiter par le processus.

6. Le cas échéant, écriture de la page.

On pourrait également envisager un système de verrouillage par préemption temporaire ("time-out") d'un processus de mise à jour.

Les constructions évoquées ci-dessus sont nettes et compréhensibles, mais, Guy Béart aurait pu le chanter, MS-DOS 3.1 s'en fout. En effet, comme signalé précédemment, la fée Microsoft, penchée sur son berceau n'a pas daigné le munir du verrouillage d'écriture.

Une ruse de guerre s'imposait (infâme bidouillage pour les mauvaises langues): s'agissant d'un environnement multi-processus ou multi-utilisateurs et, pourquoi pas, au sein d'un réseau local, toute table de verrouillage se doit d'être accessible à tous les processus. Elle revêtira donc nécessairement la forme d'un fichier dont on verrouillera, exclusivement, (MS-DOS 3.1 oblige) certaines pages; on établira une correspondance bi-univoque entre l'adresse de ces pages fantômes de taille minimale et les adresses des réelles pages à verrouiller en écriture. Moyennant l'unicité de cette correspondance bi-univoque pour l'ensemble des processus, ce système fonctionne à merveille. Certes, d'autres alternatives sont possibles, mais leur discussion n'a, ici, pas le moindre intérêt.

* EXPLOITATION SUR MACHINE CIBLE

Le schéma d'exploitation (cfr.figure 13), requiert fort peu de commentaires. Le COmpilateur de SChémas INTeractif (COSCINT) harcèle le programmeur d'application en lui demandant avec insistance tous les renseignements pertinents issus d'un schéma logique; l'exécution de ce programme produira deux fichiers de caractères ASCII: l'un, Application.H , contiendra l'ensemble des constantes, variables et autres déclarations en langage C, nécessaires à l'interface Application/BIGAM (cfr.ADL-C). Une simple intromission de ce fichier dans le programme Application.C rendra effective l'intercommunication des couches; l'autre fichier, Application.CFG , contiendra l'ensemble des caractéristiques physiques des fichiers, tampons, pages, index, trousseaux, clefs, segments, modes d'allocation des pages et réservation mémoire nécessaires à BIGAM. Ce fichier détermine entièrement une base de données pour le SGBD; il est utilisé à chaque ouverture et constitue l'interface de référence pour les outils physiques connexes au SGBD (exemple: reconstruction, compactage).

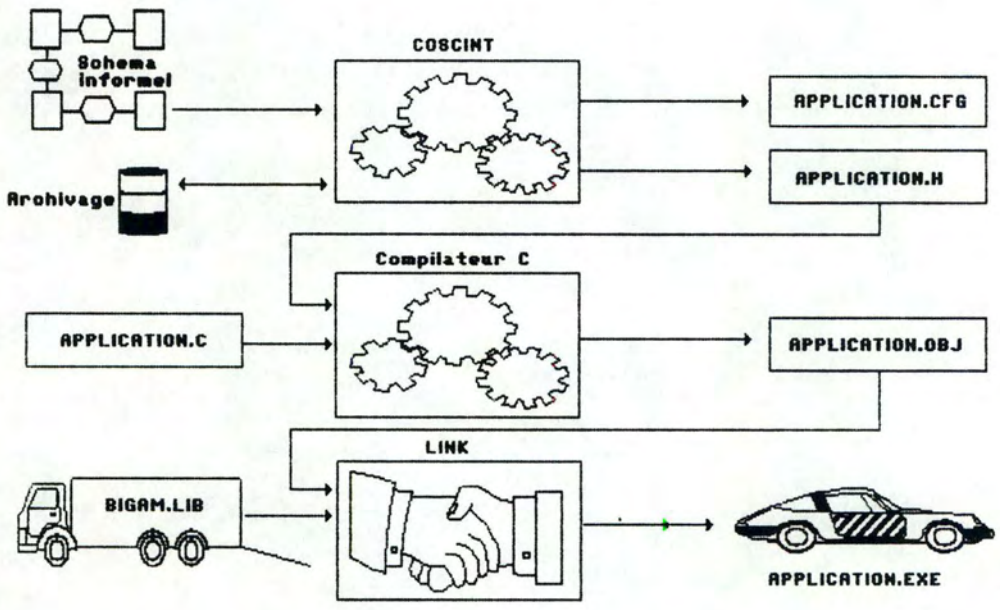


FIGURE 13 : SCHEMA GENERAL D'EXPLOITATION

Pour fixer les idées, une application sur base de données vierge, dans un environnement MS-DOS, se compose de deux fichiers:

Application.EXE

Application.CFG

A l'ouverture de cette base de données, un ou plusieurs fichiers sont créés:

Application.BD

Application.I1,I2,...,In

soit un seul fichier d'entités et un index par 30 trousseaux. En cas de reconstruction, de compactage ou de modification des structures d'index, les seuls fichiers requis sont:

Application.BD

Application.CFG

Enfin, une détection automatique d'incohérence entre la base de données et le fichier de configuration éviteront tout mélange accidentel de versions incompatibles. Outre les outils précités, on pourrait envisager la construction d'outils de chargement et déchargement, fondés sur ce seul fichier. C'est ainsi que l'auteur de ces lignes a réalisé sans trop de difficultés un chargeur de fichiers DBC (ou Dbase III), qui permit la récupération de bases de données de l'atelier ORGA en quelques secondes.

3.8 Libre commentaire

Vous voilà arrivés au terme du voyage autour du SGBD pilote BIGAM; le commandant-programmeur, en bien triste équipage, espère que le survol n'en fut pas trop pénible, et souhaite ne pas vous revoir de sitôt penchés sur ses lignes. Dans le cas contraire, vous ne pourriez que constater le dramatique échec de la mission d'explication dévolue au commandant de bord, pour lequel un des effets du même nom entraînerait la suspension de sa licence.

"La théorie nous prouve que le bourdon ne peut voler parce que le rapport entre la forme et le poids de son corps d'une part, la surface de ses ailes de l'autre, est incorrect. Le bourdon l'ignore, et c'est pourquoi il vole quand même"

Igor Sikorsky

4. CRITIQUE

Le langage C ne permet guère de prendre la hauteur nécessaire à l'élévation du débat; le diplôme de licencié et maître en Informatique ne sanctionne pas, Dieu merci, tout programmeur tâcheron en mal de titres universitaires. C'est pourquoi les chapitres précédents se sont gauchement essayés à présenter, en langage naturel, ce qui aurait pu n'être qu'un de ces "papiers" faisant la joie euphonique de l'Homo sapiens sed néophyte. L'argutie des Technologies de l'Information, d'obédience anglo-saxonne, confère souvent à ses locuteurs, l'aura druidique dont leur ego s'abreuve. La démystification de ces rituels occultes entreprise à l'Institut doit être vigoureusement encouragée; l'absence de cours ex cathedra et de nombreuses lectures critiques ont forgé l'esprit de générations d'étudiants, désormais sensibilisés à une certaine démarche scientifique dans une discipline qui s'y prête mal.

Il convient dès lors de ne pas faillir au devoir d'autocritique que tout apprenti informaticien devrait s'imposer...

4.1 Les bons côtés

Le respect de la spécification énoncée au deuxième chapitre en étonnera plus d'un; le mérite en revient entièrement au patient promoteur dont la vigilance ne peut facilement être mise en défaut.

Les divers impératifs de la classe PEC (cfr.Chapitre 2 - 2.2) sont amplement satisfaits: BIGAM.LIB ne "pèse" qu'une cinquantaine de Kb, toutes options comprises, et les performances s'annoncent conformes à ce que l'on est en droit d'escompter d'un SGBD de type réseau, marié à l'index. Ce n'est cependant pas une première, comme les encarts publicitaires de l'annexe III vous le démontreront. Hormis les points 8, 11 et les sordides questions monétaires, l'extrait de la revue Byte - août 1988

"11 Important Reasons C Programmers Use Our File Manager"

pourrait parfaitement s'appliquer à BIGAM. Si les chapitres précédents n'ont qu'aiguisé votre curiosité, sans jamais la satisfaire, cette simple feuille se révèlera peut-être plus explicite...

4.2 Les mauvais côtés

La qualité de ce document a bien évidemment souffert de la célérité avec laquelle il s'est ébauché; il se serait voulu finement ciselé, il n'est hélas que mal dégrossi et suranné. Faisant fi de toute orthodoxie en la matière, il ne peut dès lors que surprendre les habitués, astreints à la lecture de mémoires universitaires.

4.3 Les à-côtés

Quoique l'étude détaillée des motivations profondes du changement à l'origine de cette réalisation soit hors de propos, il ne faudrait pas hâtivement en déduire que les seuls impératifs de la classe PEC (Cfr.Chapitre 2 - 2.2) ont suscité la création de ce SGBD. Une petite digression s'impose...

Mordu de C.A.O (Conception Assistée par Ordinateur), un programmeur d'applications ne trouvera pas aisément un SGBD rencontrant les spécificités de celle-ci, s'il ne dispose que d'une modeste configuration informatique. La signature spatio-temporelle des structures généralement manipulées est sans commune mesure avec la gestion des amicales colombophiles en DBASE III+ (Ashton-Tate).

Une transaction CAO, opération d'atomicité requise, concernera souvent une importante partie de la base de données pendant un laps de temps qui, lui, n'a rien de transactionnel. Dans le même ordre d'idées, de nombreux domaines prometteurs, la bureautique par exemple, ne peuvent se contenter des classiques SGBD relationnels, mal adaptés à la gestion d'objets complexes, agrégats d'objets élémentaires, appelés également objets structurés (9). Enrichissements des structures de données et élaboration de langages d'accès moins "butineurs de tuples" sont les grands axes de recherche en la matière.

On ne peut nier, soit dit en passant, qu'ADL-C, féru de chemins, appartienne à la classe des SGBD hyménoptères; en effet, le résultat de toute requête d'un programme d'application se traduira par une récolte d'attribut-pollen au gré des chemins parcourus lors des appels de primitives.

Toutefois, c'est au niveau conceptuel que doit en tout premier lieu se porter la réflexion; de plus "riches" modèles sont nécessaires. Des langages de manipulation, algèbres, calculs ou langages d'inspiration "programmation logique", ont été définis avec plus ou moins de réussite, sur ces modèles d'objets structurés. En outre, les bases de données orientées objets procèdent fondamentalement des mêmes objectifs de traitement d'objets complexes sans scission arbitraire; elles représentent de ce fait une voie de recherche prometteuse.

Qu'il s'agisse d'enrichissements du modèle relationnel, muni d'un SQL étendu ad hoc, ou d'approches entièrement différentes, les exposer sortirait totalement de l'objet de ce document.

Pourquoi les mentionner alors ?

On a souvent tendance à gommer un aspect que l'auteur de ces lignes a la fatuité de croire non négligeable; une courte mise en exergue de la conclusion d'un article fort intéressant (9) l'illustrera parfaitement.

"Il est clair que les modèles structurés pourraient être utiles dans de nombreuses applications. Pour qu'ils soient vraiment utilisés, (...) il faudrait surtout implanter des systèmes raisonnablement performants basés sur ces modèles. Peu de systèmes ont été réalisés jusqu'à présent (...) Le succès de ces modèles dépend des performances de ces nouveaux systèmes, et de la qualité des interfaces qui seront proposés".

On l'aura compris, la souplesse, la versatilité, la compacité, la robustesse et la totale indépendance de la couche ADL-C sont des atouts majeurs dans le cadre de développements d'approches alternatives. Les potentialités des recherches théoriques sont en effet trop souvent bridées par l'indisponibilité de réalisations pratiques correspondantes.

Une armée où l'intendance ne suit plus stagne, et le moral des troupes s'en ressent. Chaque informaticien, directeur ou simple exécutant, peut en faire la triste expérience...

4.4 Les bas-côtés

Outre le traitement d'un sujet déterminé, un mémoire constitue également le point d'orgue d'un cycle d'études universitaires; tout au long de ce dernier, nous pouvons nous enorgueillir d'avoir été initiés à de nombreux outils, confrontés à différents aspects et démarches, éclairés quant à la réalité de l'informatique actuelle.

Cette généralité ne nous permet cependant guère d'émettre un avis autorisé sur l'un ou l'autre domaine rencontré au hasard d'un cours.

Ce comportement réservé apparaît d'ailleurs des plus singuliers eu égard aux truculentes certitudes des boucaniers du bit et autres flibustiers des floppies, ecoeurés de constater qu'on n'apprend même pas à l'université comment "déplomber" un programme de jeu. Ces dérapages prêteraient tout au plus à sourire s'ils ne s'accompagnaient d'une lame de fond d'incompétence, ravageant la quasi totalité des Technologies de l'Information. Face aux affirmations péremptoires ahurissantes d'une presse que l'on ne peut même pas qualifier de "certaine", le phénomène se généralisant, on ne s'étonnera plus de l'ignorance démente régnant à tous les niveaux de notre société. De nombreux "professionnels" de l'informatique ne dépassent pas le stade "Mère Denis" quant à la validité de leur argumentation.

Plus grave encore, notre enseignement (à tous ses échelons) se laisse infiltrer par ces histrions mégalomanes, convaincus que la connaissance des commandes de Lotus, Symphony ou Wordstar les propulse au rang d'informaticien.

D'autre part, l'irrépressible spécialisation de la recherche de haut niveau et la complexité inhérente aux domaines étudiés ne facilite pas la dissémination d'un véritable ordre de connaissances informatiques; ainsi, le chercheur-anachorète dans son atelier-thébaïde semble trop souvent intellectuellement inaccessible.

Enfin, de nombreuses "révolutions informatiques" ont conduit la populace bêlante des consommateurs médiatisés à adopter, face à la micro-informatique, des comportements psychasthéniques.

Pour ces quelques raisons, étant parvenu aux confins de l'espace littéraire qui m'était imparti, qu'il me soit permis de souhaiter à l'Institut d'Informatique d'accéder au rôle d'Arbiter Elegantiae en Technologies de l'Information auquel il pourrait légitimement prétendre.

ANNEXE I

CONCEPTION D'UNE BASE DE DONNEES

ELEMENTS METHODOLOGIQUES

Namur - mars 1987

Facultés Universitaires de Namur - Institut d'Informatique
rue Grandgagnage, 21 - B-5000 Namur (Belgique) - tél. (0)81-22 90 65

L'équipe Bases de Données de l'Institut d'Informatique, responsable des travaux en matière de méthodologie et d'atelier logiciels dont il est question dans ce document, est composée des personnes suivantes : A. Delcourt, M. Cadelli, C. Charlot, J-L Hainaut. Ont également participé dans le passé à ces travaux, Y. Delvaux et B. Van Houtte. D'autres personnes ont encore, de près ou de loin, apporté une contribution significative aux résultats obtenus . Nous citerons, sans être certain de n'en oublier aucun, J-P Robert, B. Delcourt, L. Moentack, I. Müller, J. Müller, P. Hoffelt, P. Devil , R. Lazzaroni, G. Modart, P. Maréchal, P. Bock, B. Nizet et P-A. Brès.

Namur, le 20 avril 1987

Chapitre 2

SCHEMA D'UNE BASE DE DONNEES

Ce chapitre décrit les composants de la description d'une base de données. Ces composants constituent un **modèle des données** destiné à servir de fondement à une démarche de conception de bases de données, ainsi qu'à des outils d'aide à leur conception.

2.1 GENERALITES

Les concepts décrits dans ce document sont représentatifs des démarches modernes de conception de Systèmes d'Information. On trouvera dans les références [BODPIG,83/87] et [HAI,86] un exemple de démarche générale qui s'appuie sur de tels concepts.

Une *base de données* est perçue comme un ensemble d'entités auxquelles sont affectées des valeurs d'attributs, et en association les unes avec les autres. La *description d'une base de données*, c'est-à-dire son **schéma**, est constituée principalement de la spécification des types d'entités, des types d'associations ainsi que des attributs attachés à chaque type d'entités et à chaque type d'associations. Ce schéma comprendra également des descriptions en langage naturel de ces composants, ainsi que les contraintes générales que les données doivent respecter à tout instant, c'est-dire les *contraintes d'intégrité*. Ces spécifications constituent le **schéma conceptuel** de la base de données [BODPIG,83/87].

Un **schéma logique** d'une base de données est obtenu à partir de son schéma conceptuel (ou de l'une de ses variantes) par adjonction de structures d'accès aux données. Ces structures d'accès s'expriment sous la forme de *clés d'accès* et de *(types de) chemins d'accès* [HAI,86].

Un **schéma physique** d'une base de données est une variante de schéma logique qui est conforme aux structures admises par un SGBD spécifique, et complété de la spécification des caractéristiques physiques propres à ce SGBD [HAI,86].

Tout schéma peut faire l'objet d'une description quantitative statistique qui donne la taille moyenne des populations des types de données (*quantification statique*). Tout schéma logique ou physique peut en outre faire l'objet d'une description quantitative dynamique qui donne, pour chaque opération élémentaire sur les données, le nombre moyen d'activations par unité de temps (*quantification dynamique*).

La construction d'une base de données à partir de son schéma conceptuel est basée sur deux mécanismes fondamentaux : la *transformation* de structures et l'*enrichissement* de structures. Les transformations permettent de "plier" un schéma aux exigences d'un SGBD ou à tout autre standard ou contrainte qu'impose le concepteur. L'enrichissement permettra la production d'un schéma logique à partir du schéma conceptuel, et d'un schéma physique à partir d'un schéma logique par l'ajoût de spécifications de nature de plus en plus techniques.

2.2 LES ASPECTS D'UN SCHEMA

En première analyse, un schéma décrit des *types d'entités*, des *types d'associations*, les *attributs* qui leur sont attachés, ainsi que les *contraintes d'intégrité* posées sur ces objets. Un objet d'un schéma logique ou physique a généralement pour origine un objet du schéma conceptuel. Il importe donc de spécifier pour chacun l'*origine conceptuelle*. L'ensemble de ces spécifications constituent les **structures sémantiques** du schéma.

Chaque objet peut faire l'objet de descriptions informelles exprimées en langue naturelle. Ces **descriptions textuelles** donnent entre autres choses la *sémantique* de ces objets et reprennent tout *commentaire* jugé utile par le concepteur à leur sujet.

Un schéma de nature logique ou physique contiendra généralement la spécification de **structures d'accès**, telles que des *clés d'accès* ou des *types de chemins d'accès*.

On indiquera pour chaque objet important du schéma une estimation de sa population à un instant déterminé. Ces informations constituent la **quantification statique** du schéma. Elles concernent essentiellement les *types d'entités*, les *rôles des types d'associations*, les *clés d'accès* et les *fichiers*.

La spécification des taux d'utilisation des données de chaque type constituent la **quantification dynamique** du schéma. L'utilisation des données s'exprime en termes d'*accès* (séquentiel, par clé, par chemin), de *création*, de *suppression* et de *modification*. La quantification consiste à spécifier à la fois le *nombre de fois* par unité de temps qu'une opération est effectuée sur les données d'un type, et le *nombre d'éléments* touchés par opération effectuée.

La spécification des caractéristiques et paramètres physiques constituent la **structure physique** du schéma. Elle inclut par exemple la définition des fichiers physiques et de leurs propriétés, des index, des modes d'implantation des données sur les supports, etc.

On trouvera une spécification précise de ces concepts dans [BODPIG,83/87] pour ce qui concerne les structures sémantiques, et dans [HAI,86] pour ce qui concerne les autres aspects.

2.3 LES STRUCTURES SEMANTIQUES

Les structures sémantiques d'un schéma définissent les types d'entités, les types d'associations, les attributs, les contraintes d'intégrité et les origines conceptuelles.

2.3.1 TYPE D'ENTITES

Un type d'entités - ou TE - (le terme est de nature conceptuelle, mais sera conservé quel que soit le niveau du schéma décrit) représente une classe d'objets ou d'individus perçus comme étant dotés d'une existence propre dans le réel perçu. On associera souvent des attributs à un type d'entités. Un type d'entités peut intervenir comme membre d'un ou plusieurs type d'associations. Dans un schéma logique ou physique, un type d'entités représente ce que l'on appellera de manière plus traditionnelle un *type d'articles*, un *type d'enregistrements*, un *type de segments*, etc.

Un type d'entités porte un nom qui l'identifie parmi les TE du schéma. Il est également doté d'un nom court (abréviation) et d'une date (de création par exemple).

2.3.2 TYPE D'ASSOCIATIONS

Un type d'associations - ou TA - représente des groupes d'entités de même nature, dans lesquels chaque entité joue un **rôle** particulier. Le nombre de rôles, identique pour toutes les associations d'un type, s'appelle le **degré** du type d'associations. Chaque rôle peut être joué par des entités d'un seul type (rôle **mono-domaine**) ou au contraire de plusieurs types (rôle **multi-domaines**).

On impose que le degré d'un type d'associations soit de degré égal ou supérieur à 2. En outre, un schéma logique ou physique ne contiendra que des types d'associations de degré 2 (TA binaires). On peut imposer que les rôles soient mono-domaines.

Un type d'associations porte un nom qui l'identifie parmi les TA du schéma. Chaque rôle porte un nom qui l'identifie parmi tous les rôles des types d'associations dans lesquels intervient un même type d'entités. Ce nom de rôle permet de résoudre les problèmes d'ambiguïté qui surgissent lorsqu'un type d'entités intervient dans plus d'un rôle d'un type d'associations, ou dans le cas de rôles multi-domaines.

2.3.3 ATTRIBUT

Un attribut d'un type d'entités ou d'un type d'associations représente une propriété mesurable des individus et associations d'individus du réel perçu.

Un attribut est :

- simple ou répétitif;
- obligatoire ou facultatif;
- élémentaire ou décomposable.

Le caractère simple/répétitif et facultatif/obligatoire d'un attribut est représenté par deux entiers qui spécifient le nombre minimum et le nombre maximum de valeurs de cet attribut que l'on peut attacher à une entité, à une association ou à un attribut décomposable. Ces nombres constituent la **répétitivité** de l'attribut. Ce concept est analogue à la notion de *connectivité* d'un rôle (voir plus loin).

Un attribut élémentaire possède un format (numérique, flottant, caractère, date, etc) et une longueur.

Un attribut possède un nom qui l'identifie parmi les attributs qui dépendent du même type d'entités, du même type d'associations ou du même attribut décomposable.

Dans un schéma logique ou un schéma physique un type d'associations ne peut être doté d'attributs.

2.3.4 CONTRAINTE D'INTEGRITE

Les données structurées en types d'entités, types d'associations et attributs peuvent encore être soumises à des restrictions plus sévères encore, appelées contraintes d'intégrité. Nous en retiendrons trois : l'identifiant, la connectivité et la contrainte référentielle.

Un **identifiant d'un type d'entités** (noté E) est une liste d'attributs et/ou de rôles tels que si l'on désigne une valeur pour chacun de ces attributs et une entité pour chacun de ces rôles, il y correspond au plus une entité du type E dans la base de données qui possède ces valeurs d'attributs et qui soit reliée à ces entités via ces rôles. Chaque attribut est attaché au type d'entités E et y est simple et obligatoire. Chaque rôle (noté R1) appartient à un type d'associations (noté A) tel que :

- A est de degré 2;
- A n'a pas d'attribut;
- E joue dans A le rôle R2, distinct de R1, et dont la connectivité est 1-1.

Le cas où R1 est de connectivité i-1 constitue une dégénérescence qui n'est pas envisagée.

Un **identifiant d'un type d'associations** (noté A) est une liste d'attributs et/ou de rôles tels que si l'on désigne une valeur pour chaque attribut et une entité pour chaque rôle, il y correspond au plus une association du type A dans laquelle on trouve ces valeurs d'atributs et ces entités dans les rôles spécifiés. Chaque attribut est attaché au type d'associations et y est simple et obligatoire. Chaque rôle appartient à A. Le cas où l'un de ces rôles est de connectivité i-1 constitue une dégénérescence qui n'est pas envisagée.

De deux identifiants qui possèdent des attributs en commun, on dira qu'ils *se chevauchent.*

Si un type d'entités est doté d'au moins un identifiant, l'un d'entre eux peut être déclaré *primaire*, les autres étant alors dits *secondaires*. Ce statut représente une hiérarchie de préférence, l'identifiant primaire étant privilégié dans certaines circonstances impliquant le choix d'un identifiant parmi plusieurs. Cette notion est identique à celle de *primary key*, choisie parmi les *candidate keys* dans les bases de données relationnelles.

La **connectivité d'un rôle** (noté R) d'un type d'associations (noté A) est constituée d'un couple d'entiers (noté I-J) tel que toute entité pouvant jouer le rôle R dans des associations A ne peut jouer ce rôle que dans un nombre d'associations A compris entre I et J.

Une **contrainte référentielle** d'un type d'entités B vers un type d'entités A est définie sur une liste *La* d'attributs de A et une liste *Lb* d'attributs de B telles que leurs attributs soient comparables deux à deux. La contrainte spécifie que pour toute entité B (notée EB), il doit exister une entité A dont les valeurs des attributs de *La* soient celles des attributs de *Lb* de EB. Cette contrainte est particulièrement utile dans un schéma du type *relationnel* ou du type *COBOL*. Elle représente partiellement un type d'associations dans un SGBD qui ne les admet pas.

2.3.5 ORIGINE CONCEPTUELLE

A l'exception de certains objets de nature purement technique, un objet d'un schéma logique ou physique dérive, directement ou non, d'un composant du schéma conceptuel initial. L'**origine conceptuelle** d'un objet est constituée du nom et du type de l'objet conceptuel duquel cet objet actuel dérive. Cette notion concerne les types d'entités, les types d'associations et les attributs.

2.4 LES DESCRIPTIONS TEXTUELLES

Les descriptions textuelles sont des textes consacrés chacun à un type de description qui ne peut s'exprimer par un autre aspect du schéma. On distinguera , pour chaque objet du schéma, la description sémantique, le commentaire libre et les contraintes d'intégrité informelles.

2.4.1 DESCRIPTION SEMANTIQUE

La description sémantique d'un type de données (noté D) est un texte qui décrit précisément l'**interprétation** des données du type D en termes des objets de réel perçu. En principe, cette description est entièrement figée lors après l'élaboration du schéma conceptuel. Elle peut cependant migrer vers d'autres objets lors des transformations de structures.

2.4.2 COMMENTAIRE LIBRE

Ce type de description permet au concepteur de consigner des commentaires propres à éclairer les phases de conception ultérieures, la programmation, la révision ou la lecture ultérieure du schéma. On y trouvera par exemple la justification des décisions de conception prises lors de la construction du schéma.

2.4.3 CONTRAINTES D'INTEGRITE INFORMELLE

Un schéma peut faire l'objet d'une grande variété de contrainte d'intégrité. Toute contrainte qui n'aurait pu être exprimée à l'aide des concepts exposés en 2.3.4 peut être décrite dans ce texte spécifique. Certaines contraintes peuvent y être inscrite par le concepteur. D'autres y seront consignées suite à une transformation.

2.5 LES STRUCTURES D'ACCES

Un schéma peut être enrichi de deux mécanismes d'accès : la clé d'accès et le type de chemins d'accès. Etant liés à des notions de performances et de coût, ces concepts se trouveront dans un schéma logique ou un schéma physique, mais jamais dans un schéma conceptuel.

2.5.1 CLE D'ACCES

Une clé d'accès d'un type d'entités (noté E) est une liste constituée d'un ou plusieurs attributs et éventuellement d'un rôle tels que si l'on spécifie une valeur de chaque attribut et une entité (notée e) jouant ce rôle, il est possible (au SGBD) d'accéder sélectivement et rapidement

aux entités qui possèdent ces valeurs d'attributs et qui sont liées à cette entité e. Les attributs appartiennent au type d'entités E. Le rôle est celui d'un type d'associations - noté A - (de degré 2, étant donné le niveau du schéma) dont l'autre rôle est joué par E. En règle général, l'ordre des attributs de la liste n'est pas indifférent.

Si les composants d'une clé sont également constitutifs d'un identifiant, cette clé est dite *identifiante*, et l'accès fournira au plus une entité. Dans le cas contraire, la clé est dite *non identifiante*, et un nombre quelconque d'entités peuvent être obtenues.

Une clé peut être constituée d'un attribut répétitif. La clé est alors qualifiée de *répétitive*.

Si la clé est constituée exclusivement d'attributs, les entités sont obtenues parmi toutes les entités du type E. Si la clé contient un rôle, il est possible de l'interpréter comme étant constituée des attributs de la liste, mais telle que les entités sont obtenues parmi les entités E *attachées à l'entité e via A*.

De deux clés qui possèdent des attributs en commun, on dira qu'elles *se chevauchent*.

2.5.2 TYPE DE CHEMINS D'ACCES

Dans un schéma logique ou physique, un type d'associations (noté A) entre un type d'entités E1 (et jouant le rôle R1) et un type d'entités E2 (et y jouant le rôle R2) peut être interprété également comme un moyen d'accéder, à partir d'une entité E1, aux entités E2 qui lui sont liées par A, et inversément. Nous dirons que le type d'associations A est le siège de chemins d'accès de E1 vers E2 ou de chemins d'accès de E2 vers E1. Pour être plus précis, il convient d'exprimer les chemins non pas entre types d'entités, mais bien entre rôles. On peut ainsi lever les ambiguïtés causées par l'apparition d'un même type d'entités dans les deux rôles du type d'associations. On parlera alors de chemins de R1 vers R2 et de R2 vers R1. Notons enfin qu'en raison du niveau des schémas concernés, A est nécessairement de degré 2.

Un chemin d'accès associé à A, et défini de E1 (de rôle R1) vers E2 (de rôle R2) est une séquence d'entités dont la première, appelée *origine*, est du type E1 et les suivantes, appelées *cibles*, sont du type E2. Les cibles sont les entités associées par A à l'origine du chemin. A ce chemin correspond un *mécanisme* qui permet d'accéder successivement aux cibles du chemin à

partir de la cible. Les chemins de même définition sont réputés appartenir au même *type de chemins d'accès*. A un type d'associations on peut affecter 0, 1 ou 2 types de chemins.

Il est usuel de classer les types de chemins en quatre catégories (dites **classes fonctionnelles**) définies comme suit en fonction de la connectivité des rôles origine et cible :

- si le rôle origine est `i-∞` et le rôle cible est `k-1`, le type de chemins est dit `1-N`;
- si le rôle origine est `i-1` et le rôle cible est `k-∞`, le type de chemins est dit `N-1`;
- si le rôle origine est `i-1` et le rôle cible est `k-1`, le type de chemins est dit `1-1`;
- si le rôle origine est `i-∞` et le rôle cible est `k-∞`, le type de chemins est dit `N-N`.

Ainsi qu'on le démontrera dans l'étude des transformations, il existe une dualité entre clé d'accès et type de chemins d'accès.

2.6 LA QUANTIFICATION STATIQUE

La quantification statique d'un schéma est constituée de statistiques sur les populations des types de données. Ces grandeurs sont en général dérivables de l'observation du réel perçu, et par conséquent exprimable au niveau du schéma conceptuel. On spécifiera la taille moyenne des populations des types d'entités, la cardinalité moyenne de chaque rôle des types d'associations et la taille moyenne des populations des clés d'accès.

2.6.1 POPULATION D'UN TYPE D'ENTITES

La taille (notée N_E) de la population d'un type d'entités (noté E) est le nombre moyen d'entités de ce type présentes dans la base de données à un instant de référence.

2.6.2 CARDINALITE D'UN ROLE D'UN TYPE D'ASSOCIATIONS

La cardinalité moyenne (notée μ_{r1}) d'un rôle (noté R1) d'un type d'associations (noté A) est le nombre moyen d'associations A dans lesquelles une entité joue le rôle R1.

Si R1 est de connectivité i_{r1}-j_{r1}, on a nécessairement :

$$i_{r1} \leq \mu_{r1} \leq j_{r1}$$

Si R1 est de connectivité 1-1, on a également :

$$\mu_{r1} = 1$$

2.6.3 POPULATION D'UNE CLE D'ACCES

La taille (notée N_K) de la population d'une clé d'accès (notée K) associée à un type d'entités (noté E) est le nombre moyen de valeurs distinctes de cette clé présentes dans la base de données à un instant de référence.

Remarquons que si N_E est la taille moyenne de la population de E, on a nécessairement :

$$0 \leq N_K \leq N_E$$

Si K est en outre identifiante, alors on a :

$$N_K = N_E$$

2.6.4 LONGUEURS ET VOLUMES LOGIQUES

Il est possible de calculer la longueur logique totale des valeurs d'attributs d'un type d'entités à partir de la connaissance de la longueur de chaque attribut. Grâce à la population, il est aussi possible d'en déduire le volume logique total relatif au type d'entités. On remarquera que la longueur et le volume physiques (c'est-à-dire relatifs à la base de données réelle) seront généralement différents de leur correspondants logiques. Ils ne pourront cependant être calculés exactement que lorsque l'on disposera de renseignements précis sur les caractéristiques physiques du SGBD et sur les paramètres de représentation et d'implantation des données (voir [HAI,86] pour un traitement détaillé).

2.6.5 POPULATION D'UN TYPE D'ASSOCIATIONS

La taille (notée N_A) de la population d'un type d'associations (noté A) est le nombre moyen d'associations de ce type présentes dans la base de données à un instant de référence. Cette définition n'est pas limitée aux type d'associations de degré 2.

Pour tout rôle R_i (supposé mono-domaine) de A joué par E_i, si E_i est de taille moyenne N_{Ei}, et R_i est de cardinalité moyenne μ_{ri}, alors on a nécessairement :

$$N_A = \mu_{ri} \times N_{Ei}$$

Cette propriété est importante car elle permet de calculer (plutôt que de mesurer) les cardinalités moyennes d'un type d'associations lorsque l'on connait une seule d'entre elles. En effet, on a :

$$\mu_{ri} = \mu_{rj} \times N_{Ej} / N_{Ei}$$

En particulier, si un type d'associations possède un rôle R_j de connectivité 1-1, cette relation se simplifie comme suit :

$$\mu_{ri} = N_{Ej} / N_{Ei}$$

2.7 LA QUANTIFICATION DYNAMIQUE

La quantification dynamique précise le taux d'utilisation des données dans l'hypothèse d'un profil d'exploitation défini. Ce profil est représenté par six classes d'opérations élémentaires appliquées aux entités de chaque type. On précisera, pour chaque type d'entités, le nombre d'activations (que l'on notera **na**) de chaque opération et le nombre moyen d'entités touchées par chaque activation de l'opération (que l'on notera **nea**). Les opérations retenues sont la création, la suppression, la modification, l'accès séquentiel, l'accès par clé et l'accès par chemin. Cette quantification définit la consommation par unité de temps de l'ensemble des traitements en termes d'**opérations logiques** (accès, mise-à-jour) à la base de données. Ces valeurs ne représentent pas la consommation physique (en millisecondes par exemple) mais perment de les obtenir dès que l'on a déterminé les expressions de coût physique de chacune des opération élémentaires. Ces

expressions dépendent ici encore des caractéristiques physiques du SGBD et des paramètres de représentation et d'implantation des données (voir [HAI,86] pour un traitement détaillé).

2.7.1 CREATION D'UNE ENTITE

On indiquera, par type d'entités, le nombre moyen d'entités créées par unité de temps.

2.7.2 SUPPRESSION D'UNE ENTITE

On indiquera, par type d'entités, le nombre moyen d'entités supprimées par unité de temps.

2.7.3 MODIFICATION D'UNE ENTITE

On indiquera, par type d'entités, le nombre moyen d'entités modifiées par unité de temps.

2.7.4 ACCES SEQUENTIEL A DES ENTITES

On indiquera, par type d'entités (noté E), le nombre moyen de fois (noté na_S) qu'un accès séquentiel est demandé par unité de temps. On indiquera en outre le nombre moyen d'entités (noté nea_S) obtenues lors d'un tel accès séquentiel. Le nombre moyen d'entités obtenues par accès séquentiel par unité de temps est de :

$$na_S \times nea_S$$

Sans autre spécification, et si N_E est la taille moyenne de la population de E, on admettra que :

$$nea_S = N_E$$

2.7.5 ACCES PAR CLE A DES ENTITES

On indiquera, par type d'entités (noté E), et pour chaque clé d'accès (notée K) le nombre moyen de fois (noté na_K) qu'un accès par la clé K est demandé par unité de temps. On indiquera en outre le nombre moyen d'entités (noté nea_K) obtenues lors d'un tel accès par clé. Le nombre moyen d'entités obtenues par accès via la clé K par unité de temps est de :

$$na_K \times nea_K$$

Sans autre spécification, et si N_E est la taille moyenne de la population de E et N_K celle de la clé K, on admettra que :

$$nea_K = N_E / N_K$$

2.7.6 ACCES PAR CHEMIN A DES ENTITES

On indiquera, pour chaque type de chemins C d'origine R1 et de cible R2, le nombre moyen de fois (noté na_C) qu'un accès via C est demandé par unité de temps. On indiquera en outre le nombre moyen d'entités (noté nea_C) obtenues lors d'un tel accès par chemin. Le nombre moyen d'entités obtenues par des chemins C par unité de temps est de :

$$na_C \times nea_C$$

Sans autre spécification, et si μ_{r1} est la cardinalité moyenne du rôle R1, on admettra que :

$$nea_C = \mu_{r1}$$

2.8 LES STRUCTURES PHYSIQUES

Les structures physiques sont généralement fonctions d'un SGBD particulier. On notera cependant que certains concepts se retrouvent dans plusieurs SGBD. Ainsi en est-il de la notion de fichier.

2.8.1 FICHIERS

Un fichier est perçu comme une collection d'entités d'un ou de plusieurs types. Une entité peut être stockée dans un seul fichier. Un fichier porte un nom qui l'identifie parmi tous les fichiers liés à un schéma. Pour chaque fichier, on spécifiera les types des entités que l'on peut y stocker.

REFERENCES

[BODPIG,83/87] Bodart, Pigneur, "Conception assistée des applications informatiques - 1. Etude d'opportunité et analyse conceptuelle", Masson 1983 (première édition), 1987 (deuxième édition).

[HAI,86] Hainaut, "Conception assistée des applications informatiques - 2. Conception de la base de données", Masson 1987.

ANNEXE II

Interactive

Design

Approach

L'ATELIER DE CONCEPTION DE BASE DE DONNEES

ORGA

MANUEL DE REFERENCE

Version 1.0

AVRIL 1987

L'atelier ORGA a été réalisé par l'équipe Bases de Données de l'Institut d'Informatique des Facultés Universitaires de Namur (Belgique). Cette équipe est composée des personnes suivantes : A. Delcourt, M. Cadelli, C. Charlot, J-L Hainaut. Ont également participé dans le passé à ces travaux, Y. Delvaux et B. Van Houtte. D'autres personnes ont encore, de près ou de loin, apporté une contribution significative cette réalisation. Nous citerons, sans être certain de n'en oublier aucun, J-P Robert, B. Delcourt, L. Moentack, I. Müller, J. Müller, P. Hoffelt, P. Deville, R. Lazzaroni, G. Modart, P. Maréchal, P. Bock, B. Nizet et et plus particulièrement P-A. Brès.

Namur, le 20 avril 1987

Chapitre 2

FONCTIONS ET COMPOSANTS DE L'ATELIER

Ce chapitre décrit les fonctions et les composants de l'atelier ORGA. On y répertorie les fonctions de consultation, de gestion et d'exploitation de la base des spécifications, ainsi que leur prise en charge par des processeurs de l'atelier.

2.1 LES FONCTIONS DE L'ATELIER

L'atelier est organisé autour d'une base de données qui contient les spécifications relatives à un schéma de base de données. Nous l'appellerons **base des spécifications**. Les principales fonctions concernent l'initialisation de la base des spécifications avec un schéma de base de données, la consulation du schéma, la modification du schéma, la transformation de structures du schéma, la vérification de la conformité du schéma à un standard, la génération du texte DIL correspondant au schéma, la production de rapports.

Initialisation de la base des spécifications

La première fonction de l'atelier est **l'initialisation** de la base des spécifications. Les données sont issues de la base des spécifications IDA et se présentent sous la forme d'un fichier produit par la fonction TG d'IDA. Ces données décrivent un schéma conceptuel Entité/Association. Durant la phase d'initialisation, ce schéma conceptuel est rangé dans la base des spécifications de l'atelier puis transformé en un premier schéma logique. Ce schéma est obtenu de la manière suivante :

- à tout objet est associée son origine conceptuelle;
- tout type d'associations de degré supérieur à 2 est transformé en type d'entités;
- tout type d'associations doté d'attributs est transformé en type d'entités;
- un des identifiants constitué d'attributs est déclaré primaire;
- tout type d'entités reçoit une abréviation;

- à tout type d'associations sont ajoutés deux types de chemins d'accès;
- tout identifiant constitué d'attributs devient également une clé d'accès.

En outre, sur demande explicite de l'utilisateur, les types d'associations dite N-N (de connectivité i - j, k - l avec j > 1 et l > 1) et les types d'associations récursifs sont transformés en types d'entités.

Consultation du schéma

Tout schéma logique peut être parcouru afin de consulter les différents aspects de ses composants : type d'entités, type d'associations, attribut, identifiant, clé d'accès type de chemins d'accès, descriptions textuelles, quantifications, structures physiques.

Modification du schéma

Le schéma présent dans la base des spécifications peut subir des modifications. Cette fonction permet d'augmenter le contenu sémantique et technique du schéma.

Transformation de structures du schéma

Des constructions présentes dans le schéma peuvent faire l'objet de transformations globales qui conservent la sémantique et les structures d'accès. Ces modifications complexes permettent d'atteindre la conformité à un standard défini.

Vérification de la conformité du schéma à un standard

Un schéma peut être évalué en ce qui concerne sa conformité à un standard de structure. On considérera principalement la conformité aux structures permises par des SGBD. Le résultat s'exprime par une liste d'avertissements que l'utilisateur consultera afin de corriger les structures inadéquates.

Production du texte DIL correspondant au schéma

Un schéma peut être traduit en un texte DIL qui permettra l'introduction d'un schéma physique dans l'atelier IDA, ainsi que l'établissement de liens de correspondance entre ce schéma et le schéma conceptuel IDA dont il dérive.

Production de rapports

Un rapport détaillé et un rapport global seront produits à partir d'un schéma. Le rapport détaillé contient toutes les informations du schéma, tandis que le rapport global reprend, pour chaque type d'entités, les principales quantifications.

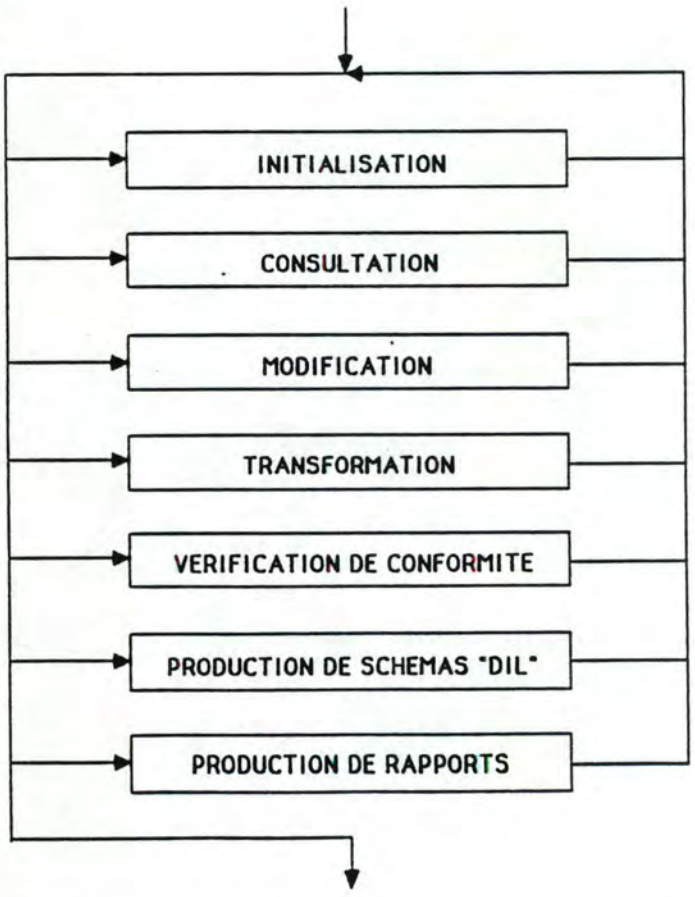


Figure 2.1 - Activation des fonctions de l'atelier ORGA

Scénario d'activation des fonctions

Si l'on excepte les contraintes évidentes (on ne peut transformer qu'un schéma ayant fait l'objet d'une initialisation, par exemple), les différentes fonctions pourront être activées dans un ordre quelconque, selon la dynamique propres à l'utilisateur ou spécifique au problèmes à résoudre. La figure 2.1 illustre les circuits possibles d'activation des fonctions.

2.2 LES COMPOSANTS DE L'ATELIER

Les fonctions sont mises en oeuvre par des processeurs spécialisés, travaillant sur la base des spécifications, et offrant à l'uilisateur des outils regroupés logiquement selon la cohérence de leur utilisation. L'atelier contient ainsi quatre processeurs définis comme indiqué dans la figure 2.2.

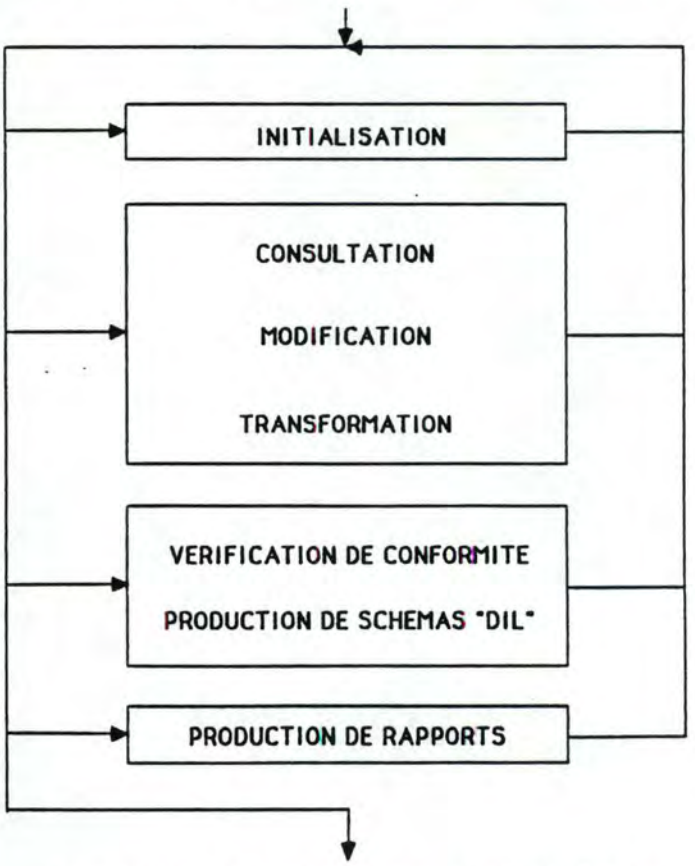


Figure 2.2 - Les processeurs de l'atelier ORGA

Le **processeur d'initialisation** est décrit à la figure 2.3. La fonction de *binarisation* transforme les types d'associations qui ne sont ni de degré 2, ni sans attributs.

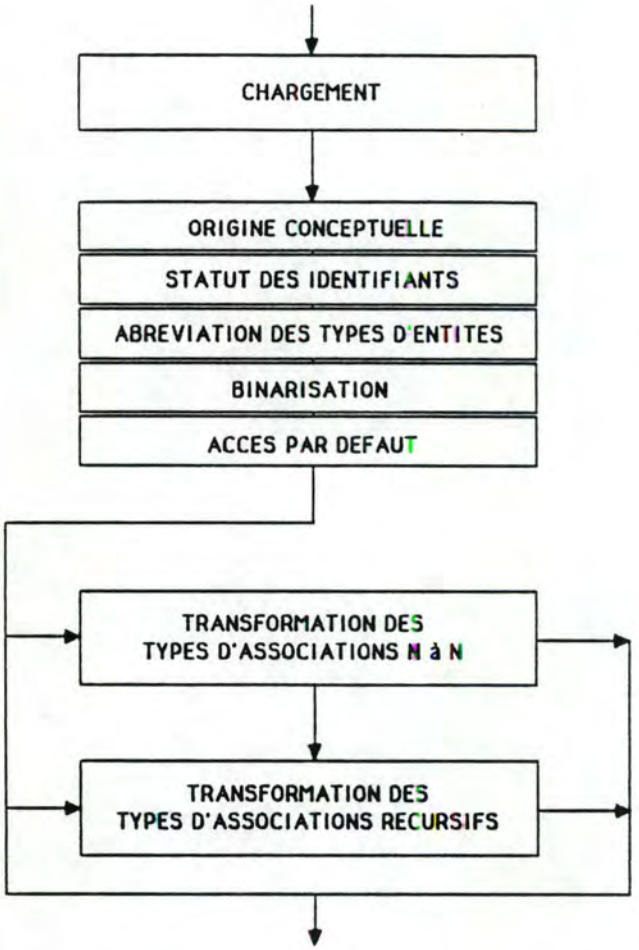


Figure 2.3 - Fonctions du processeur d'initialisation

Le processeur de **consulation-modification-transformation** regroupe les fonctions de consultation, de modification et de transformation d'un schéma. Deux fonctions de service ont été ajoutées : un traitement de textes (simple) et l'édition de la liste des types d'entités et des fichiers. Le traitement de textes donne accès, en consultation et en modification, à tout texte standard. Il permettra en particulier de consulter les avertissements du vérificateur de conformité et la liste des types d'entités. Les fonctions de ce processeur sont illustrées à la figure 2.4.

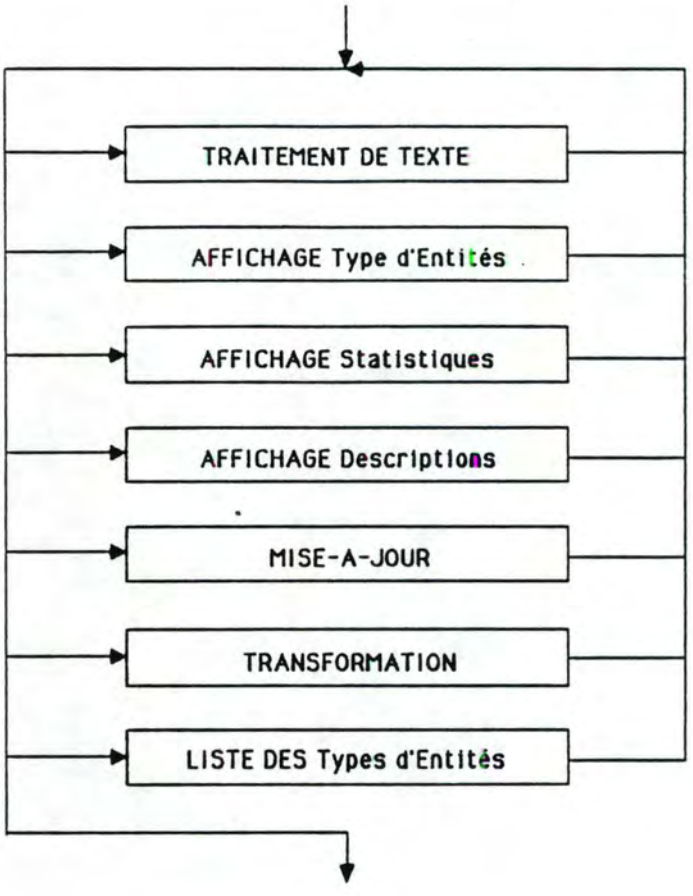


Figure 2.4 - Fonctions du processeur de consulation-modification-transformation

Le processeur de **production de schémas** regroupe les fonctions de vérification de conformité et de production du schéma DIL. Ces fonctions sont illustrées à la figure 2.5.

Le processeur de **production de rapports** offre la sortie du rapport global et du rapport détaillé. Les fonctions de ce processeur sont illustrées à la figure 2.6.

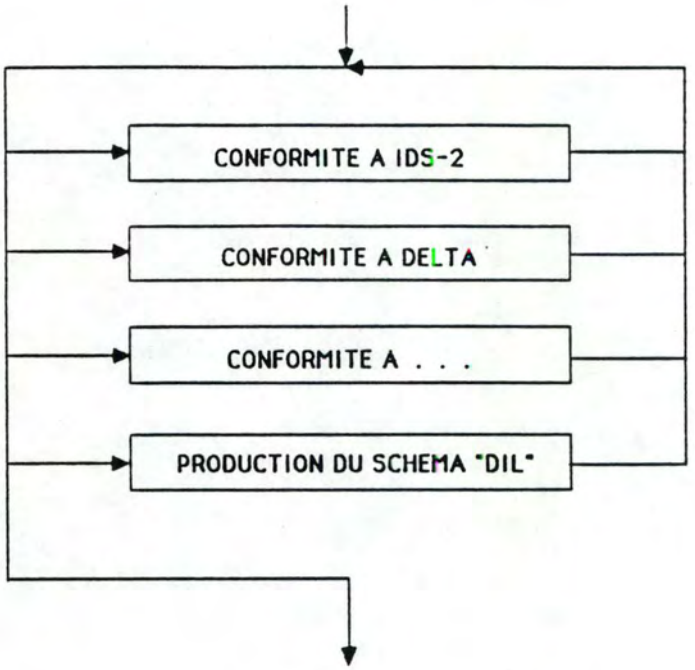


Figure 2.5 - Fonctions du processeur de production de schémas

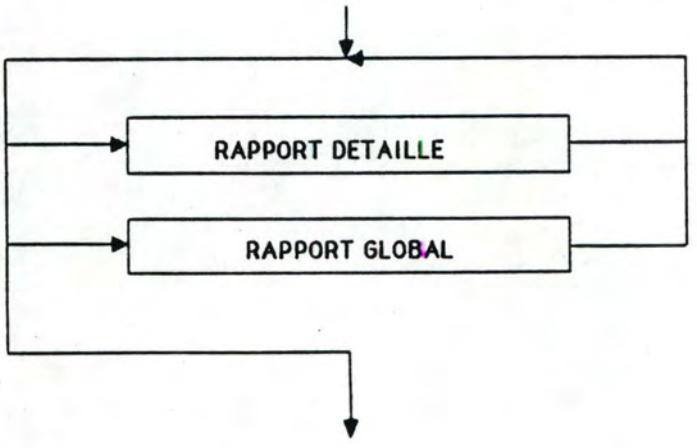


Figure 2.6 - Fonctions du processeur de production de rapports

ANNEXE III

1. Import ... Re... C Programmers Use Our File Manager

1. It's written in C.
Clearly the growing language of choice for applications that are fast, portable and efficient. All of db_VISTA's source code is written in C.

2. It's fast – almost 3 times faster than a leading competitor.
Fast access that comes from the unique combination of the B-tree indexing method and the "network" or direct "set" relationships between records. A winning combination for fast performance.

3. It's flexible.
Because of db_VISTA's combination of access methods, you can program to your application needs with ultimate design flexibility. Use db_VISTA as an ISAM file manager or to design database applications. You decide how to optimize run-time performance. No other tool gives you this flexibility without sacrificing performance.
db_VISTA is also well behaved to work with most any other C libraries!

4. It's portable.
db_VISTA operates on most popular computers and operating systems like UNIX, MS-DOS and VMS. You can write applications for micros, minis, or even mainframes.

5. Complete Source Code available.
We make our entire C Source Code available so you can optimize performance or port to new environments yourself.

6. It uses space efficiently.
db_VISTA lets you precisely define relationships to minimize redundant data. It is non-RAM resident; only those functions necessary for operation become part of the run-time program.

7. Royalty free run-time.
Whether you're developing applications for yourself or for thousands, you pay for db_VISTA or db_QUERY only once. If you currently pay royalties to someone else for your hard work, isn't it time you switched to royalty-free db_VISTA?

db_VISTA™

Features
- **Multi-user** support allows flexibility to run on local area networks
- **File structure** is based on the B-tree indexing method
- **Transaction processing** assures multi-user consistency
- **File locking** support provides read and write locks
- **SQL-based db_QUERY** is linkable
- **File transfer** utilities included for ASCII, dBASE optional
- **Royalty-free** run-time distribution
- **Source Code** available
- **Data Definition Language** for specifying the content and organization of your files
- **Interactive database access** utility
- **Database consistency check** utility

File Management Record and File Sizes
- Maximum record length limited only by accessible RAM
- Maximum records per file is 16,777,215
- Maximum file size limited only by available disk storage
- Maximum of 256 index and data files
- Key length maximum 246 bytes
- No limit on number of key fields per record
- No limit on maximum number of fields per record

Operating System & Compiler Support
- **Operating systems:** MS-DOS, UNIX, XENIX, ULTRIX, Microport, VMS, Macintosh
- **C compilers:** Lattice, Microsoft, IBM, Aztec, Turbo C, XENIX, UNIX and LightspeedC

8. SQL-based db_QUERY
is the query and report writing program that provides a relational view of db_VISTA databases. Use ad hoc or link into your C applications. Royalty-free. Source code available.

9. Free tech support.
60 days of free technical and application development support for every Raima product. Of course, extended support and training classes are also available at your place or ours.

10. Upward database compatibility
Start out with file management in a single-user PC environment—then move up to a multi-user LAN or a VAX database application with millions of records. You'll still be using db_VISTA. That's why so many C programmers are choosing db_VISTA.

11. WKS LIBRARY
The WKS LIBRARY PROVIDES THE MOST EFFICIENT WAY FOR C and BASIC programmers to interface with 1-2-3, Symphony and dBASE.
- Reads & Writes WKS & WK1 Files
- Reads & Writes DBF Files
- 1-2-3, Symphony & dBASE compatible
- Source Included
- No Royalties

30-day Money Back Guarantee!

Price Schedule	db_VISTA	db_QUERY
☐ Single user	$ 195	$ 195
☐ Single user w/Source	$ 495	$ 495
☐ Multi-user	$ 495	$ 495
☐ Multi-user w/Source	$ 990	$ 990
NEW:		
☐ WKS LIBRARY for Lotus 1-2-3	$ 195	

Call Today!
Ordering is easy — simply call toll-free. We'll answer your technical questions and get you started.

1 (800) db_RAIMA
(800) 327-2462 or
(206) 828-4636

For international orders:

In the U.K. call	Systemstar Ltd.:	(0992)500919
	FAX:	(0992)554261
In Switzerland call	Comptronix AG:	01 725 04 10
	FAX:	01 725 87 77
In France call	ISE-CEGOS:	(1) 46 09 2828
	FAX:	(1) 46 09 2800
In Belgium call	Lemnie S.A.:	(02) 720.96.57
	FAX:	(02) 721.12.00
In Germany call	ESM GmbH:	07127/5244

RAIMA™ CORPORATION

3055 - 112th NE, Bellevue, WA 98004 USA
FAX: (206) 828-3131 Telex: 6503018237 MCIUW

BY88

Announcing - the database development system that you designed.

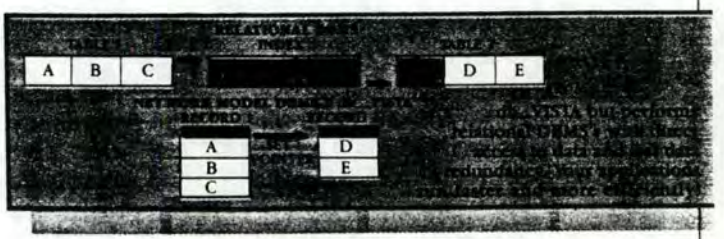
db_VistaIII™

C PROGRAMMERS- We asked what you wanted in a database development system and we built it!

db_VISTA III™ is the database development system for programmers who want powerful, high performance DBMS capabilities ... and in any environment. Based on the network database model and the B-tree indexing method, db_VISTA III gives you the most powerful and efficient system for data organization and access. From simple file management to complex database structures with millions of records. db_VISTA III runs on most computers and operating systems like MS-DOS, UNIX, VAX/VMS and OS/2. It's written in C and the complete source code is available, so your application performance and portability are guaranteed! With db_VISTA III you can build applications for single-user microcomputers to multi-user LANs, up to minis and even mainframes.

RAIMA'S COMMITMENT TO YOU: No Royalties, Source Code Availability, 60 days FREE Technical Support and our 30-day Money-Back Guarantee. Extended services available include: Application Development, Product Development, Professional Consulting, Training Classes and Extended Application Development Support.

HOW TO ORDER: Purchase only those components you need. Start out with Single-user for MS-DOS then add components, upgrade ... or purchase Multi-user with Source for the entire db_VISTA III System. It's easy... call toll-free today!



db_VISTA III™ Database Development System

db_VISTA III™	$595 - 3960
db_QUERY™	$595 - 3960
db_REVISE™	$595 - 3960
db_VISTA™ File Manager	Starts at $195

We'll answer your questions, help determine your needs and get you started.

CALL TODAY!
1-800-db-RAIMA
(that's 1-800-327-2462)

MasterCard VISA

RAIMA™ CORPORATION

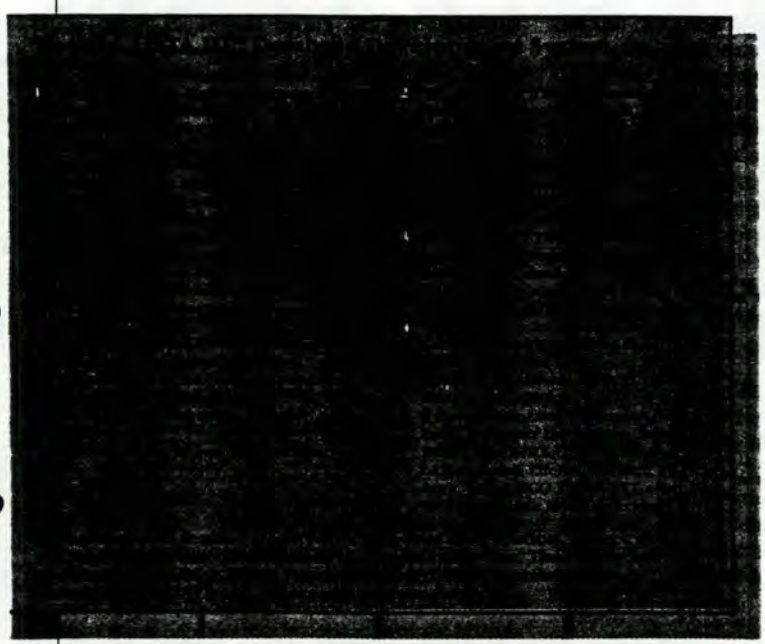
3055 112th Avenue N.E., Bellevue, WA 98004 **(206)828-4636**
Telex: 6503018237MCIUW FAX: (206)828-3131

BY38

Fast database development system with SQL-based db_QUERY and Lotus 123 interface. . . ™

C PROGRAMMERS- We asked what you wanted in a database development system and we built it!

db_VISTA III™ is the database development system for programmers who want powerful, high performance DBMS capabilities ... and in any environment. Based on the network database model and the B-tree indexing method, db_VISTA III gives you the most powerful and efficient system for data organization and access. From simple file management to complex database structures with millions of records. db_VISTA III runs on most computers and operating systems like MS-DOS, UNIX, VAX/VMS and OS/2. It's written in C and the complete source code is available, so your application performance and portability are guaranteed! With db_VISTA III you can build applications for single-user microcomputers to multi-user LANs, up to minis and even mainframes.



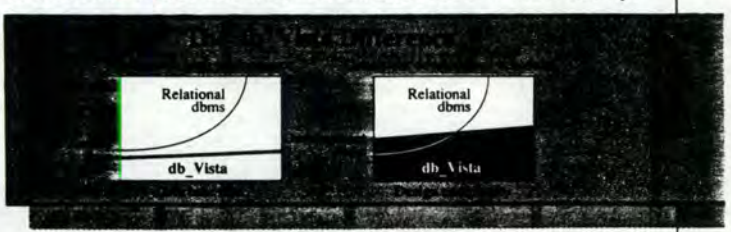
FAST • PORTABLE • ROYALTY-FREE

PROFESSIONAL SERVICES: In addition to 60 days of FREE technical support, we offer complete services to get your development project going and keep it on track:

Training Classes • Extended Support • Applications Development & C Programming Services • Consulting • Database Design & Optimization • Product Modification

We're committed to making your database project a success!

HOW TO ORDER: Call us; we'll help determine your needs and get you started. Add components as you need them. Ask about the new Lotus interface. . .Call today!



db_VISTA III™ Database Development System

db_VISTA III™	$595 - 3960
db_QUERY™	$595 - 3960
db_REVISE™	$595 - 3960
db_VISTA™ File Manager	Starts at $195
WKS Library for Lotus 123	Starts at $195

When high quality data base applications with outstanding performance are important to your company's success:

CALL 1-800-db-RAIMA

(that's 1-800-327-2462)

In the UK call Systemstar Ltd. 0992-500919

RAIMA™ CORPORATION

3055 112th Avenue N.E., Bellevue, WA 98004 **(206) 828-4636**
Telex: 6503018237MCIUW FAX: (206) 828-3131

BY48

BIBLIOGRAPHIE

(1) Conception assistée des applications informatiques
1. Etude d'opportunité et analyse conceptuelle
F.Bodart, Y.Pigneur
Masson - 1987

(2) Conception assistée des applications informatiques
2. Conception de la base de données
J-L.Hainaut
Masson - 1987

(3) Design of Database Structures
T.J.Teorey, J.P.Fry
Prentice-Hall

(4) An Introduction to Database Systems
C.J.Date
IBM Corporation
Addison-Wesley Publishing Company - 1982

(5) Atelier de Conception de Base de Données - ORGA -
Documentation,
Sources,
Spécifications,
Divers
J-L.Hainaut, M.Cadelli
Institut d'Informatique 1986-1988

(6) The ubiquitous B-Tree
D. Comer
ACM Computing Surveys - 11,2
June 1979

(7) Organization and Maintenance of large ordered indexes
R.Bayer, E.M.McCreight
Acta Informatica - 1,3
1972

(8) *Algorithms and Data Structures*
N.Wirth
Prentice-Hall - 1987

(9) *Bases de données et objets structurés*
S.Abiteboul, S.Grumbach
T.S.I. vol.6, No 5 - 1987

(A) *Views, Objects and Databases*
G.Wiederhold
IEEE Computer, December 1986

(B) *Enhancing knowledge representation in engineering databases*
D.J.Hartzband, F.J.Maryanski
IEEE Computer, September 1985

(C) *Topics in C programming*
S.G.Kochan, P.H.Wood
Hayden Books Unix System Library - 1987