ersité de Nancy I
R. de Sciences Mathématiques

Centre de Recherche en Informatique de Nancy

THESE

présentée pour l'obtention du grade de

Docteur de 3e Cycle en Informatique

par

Claude GODART

Sujet

«S G M B»
Un système de gestion multibase

soutenue publiquement le
3 JUILLET 1981

JURY

Présidente	Mme C. ROLLAND
Examinateurs	M. J.J. CHABRIER
	M. J C. DERNIAME
	M. W. LITWIN
	M. D. MEYER

Université de Nancy I
U.E.R. de Sciences Mathématiques

Centre de Recherche en Informatique de Nancy

THESE

présentée pour l'obtention du grade de

Docteur de 3e Cycle en Informatique

par

Claude GODART

Sujet

«S G M B»
Un système de gestion multibase

soutenue publiquement le
3 JUILLET 1981

JURY

Présidente ..... Mme C. ROLLAND
Examinateurs ..... M. J.J. CHABRIER
..... M. J.C. DERNIAME
..... M. W. LITWIN
..... M. D. MEYER

AVANT PROPOS

Il est de bon ton de commencer une thèse par des remerciements et je ne faillirai pas à cette coutume.

"Merci" est l'une des expressions les plus utilisées dans les conversations courantes, comme "bonjour" ou "il fait beau aujourd'hui".

Chacun les prononce par habitude, souvent inconsciemment, et il reçoit d'ailleurs une réponse aussi lourde de sens : "De rien".

C'est pourquoi je ne remercie ici personne, mais je Remercie, je veux dire j'exprime ma gratitude, à toutes les personnes qui ont construit l'environnement dans lequel j'évolue aujourd'hui.

Il s'agit de mes amis, de mes parents, de mes professeurs, des membres du jury, des gens que j'ai cotoyés et de ceux que j'oublie.

Je ne fais pas de distinctions parmi ceux-là parce que je ne peux pas en faire.

Que tous ceux qui se sont reconnus dans cette liste sachent que sans eux cette étude aurait grandi différemment qu'elle ne l'a fait et n'aurait peut être pas vu le jour. C'est pour cela que je leur suis reconnaissant.

SOMMAIRE

CHAPITRE I - "SGMB" : SYSTEME DE GESTION MULTI-BASE

A - INTRODUCTION AUX BASES DE DONNEES REPARTIES

I - Qu'est-ce qu'une base de données

II - La répartition

II-1.- La répartition des accès

II-2.- La répartition des données et des traitements

III - Qu'est-ce qu'une base de données répartie ?

III-1.- Conception ascendante d'une base de données répartie

III-2.- Conception descendante d'une base de données répartie

III-3.- Structure des accès à la base de données répartie

B - PRESENTATION DU PROJET SIRIUS

I - Objectifs

II - Type de SGBDR SIRIUS

II-1.- Problème d'un système réparti

II-2.- Manipulation de données communes

III - "SGMB" dans le cadre de SIRIUS

CHAPITRE II - ARCHITECTURE DE "SGMB"

A - ARCHITECTURE DE "SGMB"

I - Le niveau conceptuel

II - Le niveau externe

III - Le niveau interne

IV - Application aux SGBDR

B - MODELE RELATIONNEL ET "SGMB"

I - Influence de l'intégration de SGBD différents

II - Le modèle relationnel de COOD

II-1.- Notion de relation

II-2.- Relations en troisième forme normale

II-2.1.- Notion de dépendance fonctionnelle

II-2.2.- Notion d'identifiant

II-2.3.- La troisième forme normale

II-2.4.- La normalisation

C - CONCEPTION D'UNE BDR

I - Etat de l'art

II - L'approche Base-Ensemble de base

D - "SGMB" : REALISATION D'UNE BDR

I - Le niveau conceptuel

II - Le niveau interne

III - Le niveau externe

IV - Conclusion

CHAPITRE III - "SGMB" : LE LANGAGE PIVOT

CHAPITRE IV - "SGMB" : ALGORITHME DE DECOMPOSITION DE REQUETES ET DE LOCALISATION DE SOUS-ARBRES

CHAPITRE V : DECOMPOSITION DE REQUETES ET OPTIMISATION

I - Nécessité de l'optimisation

I-1.- Un SGBDR relationnel est d'abord un SGBD relationnel
I-2.- Problèmes inhérents à la répartition

II - Critères d'optimisation de "SGMB"

II-1.- Diminuer le nombre d'opérations élémentaires exécutées par "SGMB"
II-2.- Diminuer le nombre de passages par les interfaces
II-3.- Diminuer la quantité des informations échangées entre les SGBD et "SGMB"

III - Techniques d'optimisation de "SGMB"

III-1.- Réorganisation de l'arbre de décomposition d'une requête
III-1.1.- Principe
III-1.2.- Etude de la commutativité des opérateurs relationnels
III-1.3.- Application
III-2.- Localisation de sous-arbres

IV - Organisation de l'optimisation

CHAPITRE VI - "SGMB" : EXECUTION D'UNE REQUETE

I - Principe

II - Mise en forme d'un sous-arbre localisé

II-1.- Protocole d'échange de "SGMB"
II-2.- Exemple

III - La récupération des résultats associés aux sous-arbres localisés

IV - Exécution des opérateurs globaux

IV-1.- Gestion des relations temporaires
IV-2.- Besoins en synchronisation
IV-2.1.- Hypothèse
IV-2.2.- Synchronisation et opération de G-LECTURE
IV-2.3.- Exécution de l'arbre global

V - Conclusion

CHAPITRE VII - PROBLEMES ET CHOIX DE MISE EN OEUVRE

INTRODUCTION

L'étude faite dans cette thèse a pour but la définition d'un système permettant la gestion simultanée de plusieurs bases de données (BD).

Cela ne signifie pas que ce système gère plusieurs BD, car tous les systèmes de gestion de base de données (SGBD) actuels le font, mais que ce système gère une multibase , c'est-à-dire permet l'accès simultané à plusieurs BD. C'est un système de gestion de multibase de données ("SGMB").

Tous les SGBD actuels permettent seulement à un moment donné l'accès à une BD.

C'est en particulier le cas pour les systèmes de gestion de base de données réparties (SGBDR) développés jusque là. En effet, la base de données répartie (BDR) gérée par ces SGBDR se veut avant tout une BD.

Une BDR est alors une BD physiquement implantée sur plus d'un site. Cela signifie aussi que les données décrites à l'utilisateur le sont à l'aide d'un même schéma, le schéma global (SG). Réaliser l'accès physique aux données de la BDR, c'est assurer la projection du SG en un ensemble de sous-schémas, les schémas locaux (SL). Chaque SL décrit les données stockées sur un site.

L'expérience a prouvé, d'abord la difficulté de définir un SG, ensuite celle de projeter ce SG en SLs. C'est ce qui a conduit à une nouvelle définition d'une BDR [LIT 81] qui est la suivante : un ensemble de BD forme une BDR. Cela signifie que la BDR est décrite par l'ensemble des schémas qui décrivent les BD. Une multibase, ou encore base-ensemble de base, est une BDR et "SGMB" un SGBDR.

Exemple de base-ensemble de base : [LIT 81]

BD LOISIRS
    BD RESTAURANTS
        R(R #,RNOM, RUE, TYPE, TEL) Restaurants

```
      P(P #, PNOM, NCALORIES)        Plats
      M(R #, P #, PRIX)              Menus
   FIN
   RD CINEMAS
      C(C #, CNOM, RUE, TEL)         Cinémas
      F(F #, FNOM, TYPE)             Films
      P(P #, C #, HEURE, PRIX)       Séances
   FIN
FIN
```

Il est à remarquer qu'une BDR au sens base ensemble de base est logiquement répartie, dans le sens où le fait que la BDR est répartie apparaît dès le niveau logique de la BD, et non plus seulement au niveau physique.

L'utilisateur peut alors apporter lui-même les informations sur la localisation des données qu'il veut accéder.

Cela nous permettra d'assurer l'utilisation de la BDR, ce qui n'était pas toujours le cas dans l'approche habituelle (un seul schéma global) et jamais lorsqu'il s'agissait de mise à jour.

Mais il ne suffit pas de décrire des données et de permettre leut utilisation pour définir un SGBD, qu'il soit réparti ou non. Il faut aussi assurer le partage de ces données entre plusieurs utilisateurs, c'est-à-dire gérer les conflits entre ces utilisateurs, dans le but de maintenir la base de données cohérente. Nous ne nous intéressons pas ici à ces problèmes de grande complexité, qui pourront faire l'objet d'une suite à ce travail.

Les concepts développés dans ces quelques lignes sont développés dans les chapitres I et II qui conduisent à la définition de l'architecture de "SGMB".

Nous n'avons pas encore parlé de la façon dont étaient gérées les BD qui forment la BDR. Ces BD sont gérées par leur propre SGBD qui seront tous du type relationnel.

Des SGBD relationnels différents peuvent avoir des langages de définition et de manipulation différents, notre souci a été d'utiliser comme langage d'accès à la BDR le langage qui nous semble le plus représentatif du modèle relationnel, cela dans le but de permettre une intégration plus simple et plus vaste des SGBD relationnels. Le choix de ce langage est expliqué dans le chapitre III.

Permettre l'interrogation d'une base-ensemble de base, c'est avant tout savoir décomposer une requête s'adressant à une ou plusieurs BD. Un algorithme de décomposition de requête est proposé dans le chapitre IV.

Lorsque l'on veut réaliser un SGBD, quelqu'il soit, assurer des performances "acceptables" est fondamental. Des optimisations sont possibles dès la phrase de décomposition de requête et sont décrites dans le chapitre V.

Les actions à entreprendre entre le moment où la requête est décomposée et celui où l'on rend le résultat à l'utilisateur sont analysées dans le chapitre VI.

Le lien entre les bases de données et les abstractions est un des sujets les plus pointus de ces dernières années. C'est dans ce cadre que se fera la mise en oeuvre de "SGMB" en s'appuyant sur un langage développé à l'Université de NANCY I : ATM.

Enfin, je ne peux conclure cette introduction sans préciser que ce travail n'aurait pu avoir lieu hors de l'encadrement du projet SIRIUS de l'INRIA.

CHAPITRE I

"SGMB" SYSTÈME DE GESTION MULTI BASE

A - INTRODUCTION AUX BASES DE DONNEES REPARTIES

I - QU'EST-CE QU'UNE BASE DE DONNEES

C'est une collection structurée d'objets et de relations entre ces objets qui décrit le "monde réel", l'univers d'une organisation.

En opposition à une informatique plus classique, où chaque nouvelle application au sein d'une entreprise engendrait ses propres fichiers et ses propres programmes, donc ses propres données, la base de données, qui décrit toutes les données nécessaires à l'exécution d'une organisation, sera partagée entre les différentes applications développées dans cette organisation.

Le but est le suivant :

- améliorer la cohérence des données
- réduire les redondances
- réduire les efforts de saisie et de mise à jour de l'information.

Cette vue idéale présente cependant des inconvénients liés à la mise à la disposition de plusieurs utilisateurs des mêmes données. Il s'agit de :

- préserver l'intégrité de la base, c'est-à-dire s'assurer qu'au cours de son existence l'information n'est pas déformée.
- développer une procédure de partage de l'information : si un processus acquiert une ressource (une donnée) en lecture/écriture, les autres processus ne peuvent acquérir la ressource qu'en lecture. En particulier, il faut éviter toute situation conduisant à un verrou mortel.
- assurer la sécurité, la confidentialité des données, c'est-à-dire protéger celles-ci contre des utilisateurs indiscrets.

II - LA REPARTITION

II-1.- La répartition des accès

L'histoire de l'informatique a vu l'utilisateur s'éloigner de la machine de traitement. Parti d'une salle proche de l'ordinateur et souvent loin de son lieu de travail habituel, où il travaillait par lot, l'utilisateur a émigré vers des lieux plus fonctionnels et toute personne possédant un téléphone peut maintenant travailler depuis chez elle.

Cet éclatement du lieu d'accès à l'ordinateur est lié à l'épanouissement des techniques de télétraitement et des réseaux de terminaux. Il y a eu répartition des accès.

Mais une telle informatique était toujours très centralisée, dépendante d'un site unique. L'étape suivante fut d'interconnecter plusieurs réseaux de terminaux liés chacun à un ordinateur. Un utilisateur pouvait alors avoir accès à plusieurs ordinateurs ce qui lui apportait une plus grande disponibilité de logiciel.

II-2.- La répartition des données et des traitements

Dans le cas précédent les différents calculateurs étaient totalement indépendants, un programme sur un calculateur ne pouvait communiquer avec un autre calculateur.

Faire communiquer plusieurs calculateurs pour une même application fut l'étape suivante. Cela signifie que deux ordinateurs doivent être capables de s'échanger des données, que deux tâches s'exécutant sur des calculateurs différents puissent communiquer. Les réseaux de terminaux ne suffisaient plus, et on parle alors de réseau général de transport. Les plus connus en France sont CIGALE et TRANSPAC. Ils fonctionnent généralement selon le principe de la commutation de paquet et nécessitent la présence sur chaque calculateur connecté au réseau d'un logiciel d'accès à ce réseau.

On distinguera le cas où seul le site initial exécute des traitements sur les données et que les autres sites se limitent à l'envoi de données vers ce site et à la réception des mêmes données modifiées, du cas où le site de départ initialise des traitements distants sur les données des différents sites, lesquels sites peuvent également déclencher de nouveaux traitements distants.

Dans le premier cas, il y a répartition des données, dans le second répartition des traitements.

Un réseau d'ordinateur est composé, d'une part du réseau général de transport et d'autre part des calculateurs supportant le logiciel d'accès au réseau et le logiciel nécessaire à la mise en oeuvre d'applications réparties (Ex : CYCLADES).

III - QU'EST-CE QU'UNE BASE DE DONNEES REPARTIE ?

Au contact des techniques développées dans les deux paragraphes précédents, une base de données répartie est une base de données composée d'objets représentés et stockés dans des bases classiques. Lesquelles bases peuvent éventuellement se trouver sur des sites géographiquement dispersés. Une base de données répartie est avant tout une base de données, ce qui signifie que l'utilisateur doit pouvoir :

- connaître tous les objets existants dans toutes les bases distantes.
- accéder indifféremment à une base ou à une autre, ou à plusieurs en même temps.

Une base de données répartie se différencie d'une base de données par le fait que l'implémentation et le stockage des données sont réalisés par les bases de données dont elle est une fusion. On dira aussi que le niveau physique de la base de données répartie est supporté par les bases de données locales.

L'influence de la localisation des données sur la perception des objets par le concepteur de la base, mais aussi dans certains

cas par l'utilisateur constitue également une différence entre le fait que la base de données soit répartie ou non.

III-1.- Conception ascendante d'une base de données répartie

Deux cas peuvent se présenter lorsque l'on conçoit une base de données répartie. Ou l'on part sans acquis informatique, c'est-à-dire qu'il n'existe pas encore de base de données et le concepteur va pouvoir décrire la base comme un tout sémantiquement ; on parle alors de la méthode de conception descendante qui est développée dans le paragraphe suivant.

Si on part de bases de données déjà existantes, on parle de méthode de conception ascendante.

Les bases de données sont existantes et chacune possède une description (un schéma conceptuel) des objets qu'elle manipule. Le schéma conceptuel global (de la base de données répartie) sera déduit des schémas conceptuels locaux.

Dans ce type d'approche, le problème de la localisation des données ne se pose pratiquement pas puisque les bases locales existent déjà. Un problème important est la correspondance entre le schéma conceptuel global et les schémas conceptuels locaux, deux bases locales distinctes pouvant utiliser des objets semblables ou en partie semblables, mais représentés différemment.

Un autre problème se situe dans le fait que les SGBD que l'on veut faire coopérer peuvent être de modèles différents (hiérarchique, réseau, relationnel). Un SGBDR développé pour résoudre ce dernier type de conflits est dit hétérogène.

III-2.- Conception descendante d'une base de données répartie

Le concepteur de la base de données ne possède aucun acquis informatique. Les données seront réparties entre les différents sites, les critères de localisation pouvant dépendre d'une analyse de coûts, des besoins des différents sites ou de contraintes psychologiques.

Un utilisateur accèdera aux données par l'intermédiaire du SGBDR.

Les SGBD locaux, a priori tous du même modèle, auront le même rôle que dans l'approche descendante.

III-3.- Structure des accès à la base de données répartie

L'utilisateur accède aux données seulement par l'intermédiaire du SGBDR, lequel SGBDR est chargé de l'accès direct aux données stockées sur les sites distants.

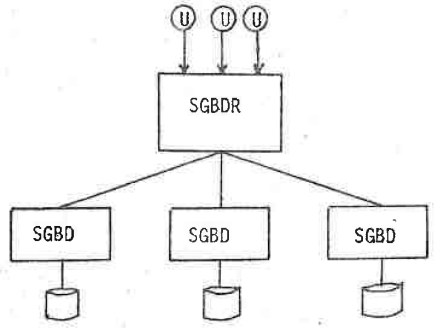


Figure 1

B - PRESENTATION DU PROJET SIRIUS ET INTEGRATION DE "SGMB" DANS CE PROJET

I - OBJECTIFS

SIRIUS est un projet pilote dont le but est d'approfondir les

techniques d'implémentation des systèmes de gestion de base de données répartie (SGBDR). [LEB 80], [SIR 80].

Un tel système est un système de gestion de base de données (SGBD) qui s'exécute dans un milieu réparti, ce qui signifie que les données qu'il gère sont stockées dans plusieurs bases de données gérées par leur propre SGBD.

Les bases de données peuvent se trouver sur une même machine ou sur plusieurs ordinateurs reliés par un réseau, auquel cas les données peuvent être sollicitées depuis différents noeuds de ce réseau.

II - TYPE DE SGBDR SIRIUS : [LIT 78]

SIRIUS s'intéresse plus particulièrement à un type de SGBDR qui peut être caractérisé comme suit :

1) il existe un certain nombre de BD locales, notées BL, qui contiennent toutes les données.

2) chaque BL est gérée par un SGBD propre tel que : IMS, SOCRATE, SYNTEX, Système R.

3) le SGBDR doit permettre à un utilisateur de voir ces bases comme une seule BD qui peut être définie comme un ensemble de BD.

4) l'intégration se fait uniquement par les interfaces habituelles des SGBD. Ex : Interface COBOL pour SOCRATE.

5) le réseau est un réseau de type général (Ex : TRANSPAC) et impose des délais d'accès aux informations plus grands que ceux des accès à une mémoire locale.

Les problèmes posés par un tel système peuvent être divisés en deux ensembles :

- les problèmes dits d'un système réparti

- les problèmes de la manipulation des données mises en commun.

II-1.- Problèmes d'un système réparti

Il s'agit de gérer la synchronisation et la concurrence d'un ensemble de processus qui s'exécutent en parallèle tout en s'échangeant des données. Pour la base de données, ces problèmes sont surtout des problèmes de cohérence globale.

La contrainte (5) introduit des difficultés par rapport à un système multiprocesseur classique (absence d'horloge commune, de contrôle centralisé,...).

Un processus dans un tel milieu doit pouvoir s'exécuter sans une connaissance exacte des autres processus coopérant avec lui.

II-2.- Manipulation de données communes

Il s'agit de trouver des solutions techniques pour la partie du SGBDR qui assure la coopération des bases locales. Les problèmes sont les suivants :

- définition d'une architecture qui offre à l'utilisateur les fonctions habituelles d'un SGBD.
- choix des langages de description et de manipulation des données mises en commun.
- traduction de ces langages en ceux des SGBD locaux.
- traduction des formats de données.
- décomposition de la requête adressant plusieurs bases en requêtes mono-base.
- contrôle global d'intégrité.

III - "SGMB" DANS LE CADRE DE SIRIUS

On a distingué deux types de problèmes, les problèmes dits de systèmes répartis et ceux dits de manipulation de données communes.

On peut constater qu'une étude centralisée, c'est-à-dire faisant l'hypothèse d'un seul point d'accès à la base de données répartie suffit à l'étude de la seconde classe de problèmes.

Ce sont ces problèmes que nous nous proposons d'étudier.

Cette approche ramène les problèmes de concurrences entre processus répartis aux problèmes de concurrence d'accès entre plusieurs utilisateurs accédant à une base de données.

Le problème est alors classique et loin du cas général, nous ne nous y intéresserons pas.

Tout se passe comme si "SGMB" était un système mono-utilisateur.

> "SGMB" : Système mono-utilisateur qui gère l'accès simultanée à plusieurs bases de données, chacune possédant son propre SGBD.

CHAPITRE II

ARCHITECTURE DE "SGMB"

A - L'ARCHITECTURE ANSI-SPARC

Le groupe de travail ANSI-SPARC [ANS 75] a proposé un modèle (une architecture) pour la définition d'un SGBD. Il distingue trois niveaux :

I - LE NIVEAU CONCEPTUEL

Ce niveau doit permettre la description de la base de donnée. Il assure la définition du schéma conceptuel (SC). Ce schéma est défini par l'administrateur de la BD. Ce schéma est la modélisation technique de l'organisation qu'il décrit.

II - LE NIVEAU EXTERNE

Il permet la définition d'un schéma externe (SE) par l'administrateur de chaque application.

Un schéma externe permet à un utilisateur d'avoir une vue propre de la BD, de voir celle-ci à sa manière.

Cette base est une redéfinition d'une partie de la base définie dans le schéma conceptuel.

III - LE SCHEMA INTERNE

Il définit l'implantation physique de la base.

IV - APPLICATION AUX SGBDR

Les architectes de SGBDR ont essayé d'étendre cette architecture [ADI 78], [LIT 78]. Le résultat a souvent été contradictoire avec le modèle de départ.

Le récent travail dans [LIT 81] permet cependant une application directe du modèle. C'est cette architecture que nous retiendrons dans le sous-chapitre D.

B - MODELE RELATIONNEL ET "SGMB"

I - INFLUENCE DE L'INTEGRATION DE SGBD RELATIONNELS DIFFERENTS

Permettre à un utilisateur de concevoir une base de données répartie à partir de bases de données existantes et gérées par leur propre SGBD, c'est prévoir :

- un langage de description permettant la définition d'un modèle global de données englobant les modèles locaux (réseau, hiérarchique ou relationnel).

- un jeu d'interfaces entre le langage de manipulation du SGBDR et ceux des SGBD locaux afin d'assurer l'homogénéisation des différents SGBD.

Ces problèmes sont ceux de l'élaboration d'un SGBDR hétérogène (les SGBD locaux sont de types différents).

De nombreuses études faites sur ce sujet ont montré la grande difficulté du problème, c'est pourquoi nous nous limiterons au cas où tous les SGBD sont du même type et permettent de gérer des données décrites à l'aide du modèle relationnel.

Remarque : Même des SGBD relationnels différents nécessitent la définition d'un protocole assurant l'homogénéisation des différents langages de manipulation utilisés par ces SGBD. (cf chapitre III)

Le langage de description de "SGMB" doit être capable de décrire des relations.
"SGMB" : système de gestion multi-bases relationnel*

II - LE MODELE RELATIONNEL DE CØØD : [COD 70]

L'objectif de ce modèle est d'améliorer l'indépendance entre les programmes et les données, c'est-à-dire entre le contenu de la base de données et les solutions techniques d'accès à ce contenu.

II-1.- Notion de relation

* Définition mathématique d'une relation

Soient $D_1,D_2,\ldots,D_n$ n ensembles.
Une relation R sur ces n ensembles est un sous-ensemble du produit cartésien $D_1 \times D_2 \times \ldots D_n$ tel que :
Pour tout n-uplet, $<d_1,d_2,\ldots,d_n>$ appartenant à R, d_1 appartient à D_1, d_2 à $D_2,\ldots,$ d_n à D_n.

* Une relation est définie par :

- un nom de relation et un ensemble de nom de constituants (les attributs).

Ex : $R(A_1,A_2,\ldots,A_n)$
R est le nom de la relation
$A_1,A_2,\ldots,A_n$ sont les constituants de la relation.

- un ensemble de champs ou domaines dans lesquels les constituants prennent leurs valeurs.

* Nous entendons relationnel au sens de COOD (Cf § II) [COD 70].

Une relation représente l'association entre objets du monde réel.

Ex : PERSONNE (NOM, PRENOM, AGE)

Cette relation représente le fait que l'association entre les objets NOM, PRENOM et AGE définit l'objet personne.

PERSONNE :

NOM	PRENOM	AGE
DUPONT	Pierre	18
DURAND	Paul	40
DUBOIS	Jacques	50
ANDRE	Jean	25

{DUPONT, Pierre, 18} est une réalisation de l'objet PERSONNE, c'est un n-uplet.

Le domaine associé au constituant NOM est l'ensemble des noms possible dans l'univers considéré.

Le domaine associé au constituant AGE est l'ensemble des entiers compris entre 0 et 120 (âge maximum).

Remarque : Cet exemple simple montre la difficulté d'associer à un constituant un domaine. Le domaine du constituant NOM (Idem PRENOM) étant pratiquement indéfinissable.

II-2.- Relations en troisième forme normale

II-2.1.- Notion de dépendance fonctionnelle

Un constituant B est dit fonctionnellement dépendant d'un constituant A dans une relation R (A,B,C,D) si et seulement si à toute valeur a de A n'est associée qu'une et une seule valeur b de B.

Exemples :

- une voiture ne possède qu'une cylindrée.
- à un type de produit commandé par un client ne correspond qu'un montant.
- un bateau n'a qu'un seul port de rattachement.

II-2.2.- Notion d'identifiant

C'est un constituant (ou une suite de constituants) dont les valeurs déterminent de façon unique un n-uplet de la relation. Une relation comporte toujours au moins un identifiant.

II-2.3.- La troisième forme normale

Une relation est en troisième forme normale si et seulement si tous les constituants qui ne sont pas identifiants sont :

- mutuellement indépendants (il n'y a pas de dépendances fonctionnelles entre eux).
- fonctionnellement dépendants de l'identifiant de R.

II-2.4.- La normalisation

C'est un processus structurant qui permet la mise en troisième forme normale de l'ensemble des objets perçus pour la définition du schéma conceptuel.

L'intérêt de cette démarche sera exprimé dans le chapitre suivant (langage pivot-langage de manipulation).

C - CONCEPTION D'UNE BASE DE DONNEES REPARTIE

I - ETAT DE L'ART

Le plus grand nombre des études faites à ce sujet reposaient sur la définition d'une base de données répartie suivante : "Une BDR est une base de données "classique" qui est implantée sur plusieurs sites au lieu d'un".

Cela signifie en particulier que la base de données répartie possède un schéma conceptuel, généralement appelé schéma global qui permet la définition de l'ensemble des données logiques existant sur les différents sites.

Ce schéma global est dit valable si l'on peut définir pour chaque schéma local une fonction assurant la correspondance entre ce schéma global et les schémas locaux.

La définition d'un tel schéma se révèle en général très compliquée, comme l'est d'ailleurs la définition d'un schéma conceptuel classique, c'est-à-dire pour une base centralisée, dans le cadre d'une organisation complexe.

Ce problème est réel quel que soit le type d'approche pour la conception de la BDR.

(a) Approche descendante

On peut penser que des organisations ayant nécessité de l'utilisation de la répartition sont suffisamment complexes pour que la définition d'un schéma conceptuel leur soit difficile.

(b) Cas où la base est est conçue de façon ascendante

Le problème est celui du rapprochement de schémas locaux pour la définition d'un schéma global.

Il s'agit de définir et de créer "les jointures" entre les différents schémas locaux. Ces "jointures" correspondent à des

liens sémantiques entre objets des bases locales.

Ces jointures seront créées intrinséquement par le processus de normalisation grâce à la définition de dépendance fonctionnelle entre relations de bases différentes.

Ces dépendances fonctionnelles doivent être déduites de l'ensemble des schémas locaux en appliquant les propriétés des dépendances fonctionnelles, c'est-à-dire :

- la transitivité

Notation : $\rightarrow$ signifie est en dépendance fonctionnelle.
Soient R1 dans la base B1, R2 dans B2.
Si dans R1 : $E \rightarrow F$
R2 : $F \rightarrow G$
alors dans la base globale, $E \rightarrow G$

- l'additivité

dans R1 : $E \rightarrow F$
R2 : $E \rightarrow G$
alors dans la base globale, $E \rightarrow F,G$

- la pseudo-transitivité

dans R1 : $E \rightarrow F$
R2 : $E,G \rightarrow H$
alors globalement, $E,G \rightarrow H$.

Une étude faite dans [CAL 76] montre l'efficacité des études sur la transitivité et d'additivité, mais prouve que la pseudo-transitivité ne peut pas mettre en évidence de liens sémantiques pour la BDR.

Dans les deux cas, les principaux problèmes sont les suivants :

- Deux objets sémantiquement différents peuvent être connus par le même nom dans deux bases de données différentes (cas de l'approche ascendante) ou simplement dans deux sous-organisations différentes. Il faut alors les distinguer dans le schéma global.

- Inversement, le même objet peut être nommé différemment par deux entités différentes. Il faut alors savoir reconnaître dans le schéma global qu'il s'agit du même objet.

- La normalisation d'un schéma de relations nécessitent la prise en compte des dépendances fonctionnelles qui ne sont pas faciles à appréhender dans le cadre d'une organisation complexe et encore moins lorsque la BDR est conçue à partir de bases de données existantes.

Il semble par conséquent que la définition d'un tel schéma global ne soit possible que dans un cadre restreint et applicable seulement que pour peu d'applications potentielles.

II - L'APPROCHE BASE-ENSEMBLE DE BASE

Les problèmes de faisabilité d'une BDR, définie comme elle l'est dans le paragraphe précédent, ont conduit à une autre définition : "Une base de données est répartie si elle est composée de plusieurs BD, d'un ensemble de base de données, sinon c'est une base de données non répartie (BDN)".

Cela signifie que la BDR sera décrite par l'ensemble des schémas conceptuels des BDN qui la compose (sans définition d'un schéma global, donc sans problèmes de projection de ce schéma en sous-schémas locaux).

Le fait que la base de données soit répartie n'est plus lié à des contraintes physiques, mais à l'expression de la logique d'une organisation (chaque BDN décrit un sous-ensemble de l'organisation). En particulier, les BDN peuvent toutes être stockées sur un même site.

La répartition physique des données n'apparaît qu'au niveau interne de la BDR.

Les principaux avantages de cette approche sont :

- la définition simple d'une BDR : une base de données qui est une base-ensemble de bases est une base de données répartie, sinon c'est une base "classique".

- le schéma global n'est plus nécessaire, on peut concevoir une BDR même si la définition d'un schéma global est impossible.

- la distinction réelle entre la répartition logique et la répartition physique permet l'application intégrale de la terminologie ANSI-SPARC aux BDR.

- une BDR peut être définie dynamiquement dans le sens où on peut adjoindre à un certain nombre de bases locales (au moins deux) d'autres bases locales, c'est-à-dire augmenter la cardinalité de l'ensemble de bases locales qui définit la BDR. Il est évident que cette adjonction devra se faire dans certaines règles de façon à respecter celles qui régissent la gestion des bases déjà existantes.

Des études théoriques concernant l'extension des concepts appliqués jusque là aux bases classiques tels que la définition de l'architecture, le modèle de données, l'intégrité ou la confidentialité d'une base-ensemble de bases sont actuellement en cours dans le cadre du projet B_UA_UBA à L'INRIA. [LIT 80].

Exemple de base-ensemble de base : tiré de [LIT 80]

```
BD LOISIRS
  BD RESTAURANTS
    R(R #, RNOM, RUE, TEL)          restaurants
    P(P #, PNOM, CALORIE)           plats
    M(R #, P #, PRIX)               menus
    C(C #, CNOM)                    types de cuisine
    T(P #, C #)                     types de plat
  FIN RESTAURANTS

  BD CINEMAS
    C(C #, CNOM, RUE, TEL)          cinémas
    F(F #, FNOM, GENRE)             films
    P(P #, F #, DATE, HEURE, PRIX)  projections
  FIN CINEMAS
```



```
  BD METROS
    L(L #, LNOM)                          lignes
    S(S #, SNOM, HOUVERT, HFERMETURE)     stations
    LS(L #, S #)                          lignes-stations
    SR(S #, RUE)                          sorties-stations
  FIN METRO
FIN LOISIRS
```

LOISIRS est une base-ensemble de bases. LOISIRS est donc une BDR. Les bases locales sont RESTAURANTS, METROS, CINEMAS.

L'ensemble des bases METROS, CINEMAS, RESTAURANTS constituent la base LOISIRS.

Il est à remarquer que la définition d'un schéma global associé à une base aussi simple que LOISIRS n'est pas évidente. En particulier, il faut pouvoir différencier les deux objets de même nom C (types de cuisine et cinémas).

Comme toute base de données ensemble de bases, la base LOISIRS peut être interrogée par deux types de requêtes.

- Les requêtes non réparties : qui s'adressent à une seule base.
  Exemple : "Noms de tous les restaurants qui servent des plats chinois".

- Les requêtes réparties : qui s'adressent à plusieurs bases.
  Exemple : "Noms des restaurants qui servent des plats chinois et qui sont dans une rue où un cinéma joue "Délivrance"".

Il est à remarquer que la répartition des données entre les différentes bases étant logique, l'utilisateur peut sans problème apporter des informations de localisation des objets sur lesquels il travaille, c'est-à-dire nommer les bases de données locales.

D - "SGMB" : REALISATION D'UNE BDR

I - LE NIVEAU CONCEPTUEL

Notre système retient l'approche base-ensemble de bases pour la définition de la BDR.

Il s'agit donc de pouvoir décrire chaque base de données locale et de pouvoir signifier qu'un lien existe entre plusieurs de ces bases de données, ce lien permettant la définition d'une base de données répartie.

Nous avons dit précédemment que "SGMB" est du type relationnel, ce qui signifie que chaque base de données locale sera décrite à l'aide de relations.

Plus précisément, ce schéma conceptuel sera un schéma relationnel normalisé.

Le schéma conceptuel de la BDR est une juxtaposition des schémas conceptuels des bases qui la compose. Il est à noter que cette approche permet une distinction réelle entre le niveau conceptuel et le niveau interne de la BDR.

Ce n'était pas le cas dans les autres architectures de SGBDR proposées précédemment [ADI 78] [LIT 78]. Ces architectures qui se voulaient des extensions du modèle ANSI/SPARC étaient souvent en contradiction avec ce modèle.

L'approche multi-base (ou base-ensemble de bases) permet une application directe de ce modèle.

L'exemple du sous-chapitre C est un exemple de schéma conceptuel d'une BDR gérée par "SGMB".

II - LE NIVEAU EXTERNE

Nous supposons, dans un premier temps que l'utilisateur travaille directement sur le schéma conceptuel.

Remarquons tout de même que l'on peut profiter de ce schéma externe pour donner à l'utilisateur une vue globale de sa base de données, lorsque cette vue globale est simple et qu'il agit seulement en consultation.

III - LE NIVEAU INTERNE

Deux cas de présentent :

- ou une base de données locale est stockée sur un site distant auquel cas le niveau interne est réalisé par une "assignation" de cette base de données à ce site,
- ou elle est stockée par "SGMB" et on rejoint le cas classique.

IV - CONCLUSION

Une base de données répartie gérée par "SGMB" est une base du type base-ensemble de base. Chaque base locale sera décrite par un schéma relationnel normalisé.

CHAPITRE III

"SGMB" : LANGAGE PIVOT

I - INTRODUCTION

Le but de ce chapitre est d'expliquer le choix de l'algèbre relationnel comme langage de manipulation de données pour "SGMB".

Nous essayons d'y montrer que ce choix assure à "SGMB" une plus grande généralité en laissant une plus grande liberté de définition d'un langage externe et surtout en permettant l'intégration plus simple de la plupart des SGBD existants.

II - DEFINITION ET NECESSITE D'UN LANGAGE PIVOT

II-1.- Rappels

- Structure des accès à la base de données répartie (cf I.A.III.3)

L'utilisateur accède aux données par l'intermédiaire du SGBDR et du SGBDR seulement, lequel est chargé de l'accès direct aux SGBD locaux.

- Hypothèse (cf I.B.II)

1) Chaque BD locale est gérée par son propre SGBD.

2) L'intégration des SGBD locaux se fait uniquement par les interfaces habituelles du SGBD.

II-2.- Qu'est-ce qu'un langage pivot

Pour une meilleure compréhension de la notion de langage pivot, nous nous plaçons dans le cas où la BDR est conçue de façon ascendante.Les utilisateurs de la BDR existaient avant la création de celle-ci. Ils travaillaient sur les BD locales que l'on veut faire coopérer. Une première réaction peut être, dans le but de faciliter leur adaptation à la répartition, de leur permettre de continuer la manipulation des données dans le langage qu'ils utilisaient jusqu'alors (celui propre à leur SGBD). Les conséquences seraient les suivantes :

- Autant de langages de manipulation que de SGBD locaux différents.
- Pour chaque langage, autant de procédures d'échanges (de traducteurs) que de langages qui lui sont différents.
  Pour n langages différents, on aura n(n-1) traducteurs.
- L'algorithme d'exécution répartie, et en particulier de décomposition de requête devra être capable de s'exécuter à partir de données différentes (de langages différents).

C'est pour ces raisons que l'on a introduit un langage pivot qui sera le seul utilisé pour l'exécution du SGBDR, les autres langages se comportant comme de véritables langages externes qui seront traduits dans le langage pivot avant l'exécution répartie de toute requête. Le logiciel nécessaire à la mise en oeuvre d'une telle architecture sera pour n SGBD différents :

- pour chaque langage, un traducteur de ce langage dans ce langage pivot.
- pour chaque langage, un traducteur du langage pivot dans ce langage.
- soit 2n traducteurs.

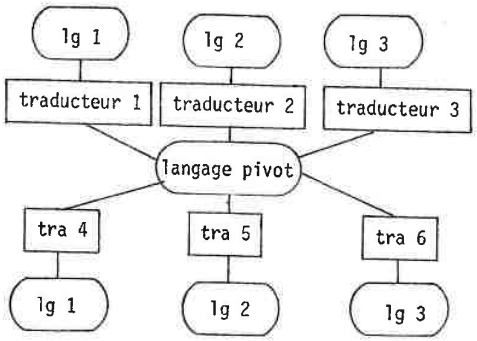


Figure III.I.2 : Couches de langages

II-3.- Type de langage pivot

II-3.1.- Définitions

Langage assertionnel :

Langage dans lequel on exprime globalement l'ensemble des informations que l'on veut obtenir sans indiquer le détail des opérations à réaliser pour les obtenir.

Exemple : Algèbre de COOD, SEQUEL, SOCRATE.

Langage navigationnel :

Langage dans lequel on exprime l'algorithme qui permet, à l'aide d'instructions de manipulation d'ensemble de n uples, de haut niveau, d'obtenir l'ensemble d'informations que l'on désire.

Exemple : DL/1 de IMS, SEQUEL 2.

II-3.2.- Difficulté de traduire un langage navigationnel dans un langage assertionnel

Réaliser un tel traducteur, c'est passer d'une suite d'instruction à une assertion. C'est un problème de grande complexité, connu des gens qui font des preuves de programme.

C'est pourquoi l'utilisation d'un langage pivot navigationnel nous semble mal adaptée.

II-4.- Langage pivot et "SGMB"

Pour les raisons exprimées dans le paragraphe précédent, "SGMB" utilise un langage pivot de type assertionnel qui servira dans une première étape de langage de manipulation à l'utilisateur.

Des traducteurs seront par conséquent nécessaires que dans le sens langage pivot, langage lié à un SGBD.

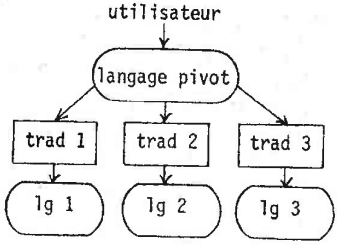


Figure III.1.4 : Couches de langages dans SGMB

III - CLASSIFICATION DES LANGAGES RELATIONNELS

Le problème primordial résolu par les langages permettant de travailler sur une base de données est celui de l'accès aux informations, c'est-à-dire le développement de primitives assurant le recouvrement des informations auxquelles l'utilisateur peut vouloir accéder. Il s'agit du langage d'interrogation.

C'est pourquoi nous limiterons cette étude aux langages d'interrogation.

Nous allons reprendre la classification proposée par D. CHAMBERLIN qui distingue quatre classes de langages relationnels [CHA 76].

1) Le calcul relationnel

Ce type de langage est fondé sur l'observation que le calcul du prédicat du premier ordre peut s'appliquer à l'utilisation de relations normalisées.

CODD a développé un tel langage, appelé ALPHA. Il y distingue

la partie spécification du résultat ("Target") de la partie qualification de ce résultat qui permet la sélection de tuples particuliers vérifiant une condition qu'elle définit. La partie qualification peut utiliser les quantificateurs universel et existentiel.

Exemple :

TARGET {
RANGE relation 1 R1
RANGE relation 2 R2
GET W R.2. Attribut 1 :

∃R1 (condition 1 et condition 2 ou condition 3)

QUALIFICATION

Pour des raisons d'analyse de la requête "RANGE relation 1 R1" renomme la relation 1 par R1.

GET W range le résultat de la requête dans la zone mémoire appelée W.

QUEL, le langage développé pour l'utilisation du SGBD relationnel INGRES appartient à cette classe.

2) Algèbre relationnel

L'algèbre relationnel est une collection d'opérateurs qui, appliqués aux relations, ont pour résultat une nouvelle relation. Les opérateurs relationnels principaux sont les suivants :

PROJECTION :

Cette fonction a pour résultat une relation dont les attributs sont ceux spécifiés en paramètre. Elle élimine de plus les duplications :

Exemple : Soit R1 (attr 1, attr 2, attr 3, attr 4)
PROJECT (R1, attr 2, attr 4) a pour résultat la relation RES (attr 2, attr 4).

Si l'on représente une relation par une table, l'effet de la projection appliquée à cette table est de prendre les colonnes correspondantes en éliminant les duplications.

SELECT :

Cet opérateur sélectionne seulement les tuples associés à une relation qui vérifient la condition donnée en paramètre.

Exemple : Soit R1

A	B	C
a_3	a_1	c_1
a_1	b_2	c_2
a_2	b_3	c_3
a_1	b_1	c_4

SELECT (R1, A = a_1) a pour résultat la relation RES :

RES

A	B	C
a_1	b_2	c_2
a_1	b_1	c_4

JOIN :

Cet opérateur prend deux relations en argument. La relation résultat est formée par la concaténation d'un tuple de la première relation avec un tuple de la seconde chaque fois qu'une certaine condition est vérifiée.

Soit R1 :

A	B
a_1	b_1
a_1	b_2
a_2	b_1
a_2	b_4

R2 :

C	D
a_1	c_1
a_1	c_2
a_2	c_1

Exemple : JOIN (R1, R2, A = C) a pour résultat RES :

RES :

A	B	C	D
a_1	b_1	a_1	c_1
a_1	b_1	a_1	c_2
a_1	b_2	a_1	c_1
a_1	b_1	a_1	c_2
a_2	b_1	a_2	c_1
a_2	b_4	a_2	c_1

OPERATEURS ENSEMBLISTES :

Les opérateurs UNION, DIFFERENCE et INTERSECTION peuvent s'appliquer à deux relations dont les attributs sont compatibles, c'est-à-dire qui prennent leurs valeurs dans le même domaine.

AUTRES OPERATEURS :

D'autres opérateurs sont parfois définis, mais il est toujours possible d'obtenir ceux-ci par combinaison de ceux définis précédemment, c'est pourquoi nous n'en parlerons pas ici.

COMPOSITION :

Il est toujours possible d'appliquer un opérateur algébrique à une relation elle-même obtenue par application d'un tel opérateur à une autre relation. Les opérateurs algébriques sont composables.

3) Langages orientés "mapping"

Dans une requête, on désire obtenir de l'information à partir de données connues, d'attributs définis.

Une relation assure l'association, la correspondance entre attributs sémantiquement liés.

Ce sont ces propriétés qui sont utilisées par les langages de mapping, qui à des attributs connus, spécifiés, fait correspondre des attributs que les spécifications de la requête ne permettaient pas d'obtenir directement.

Exemple : Soit la relation PERSONNE (NOM, SALAIRE).
Soit la requête : "Trouver les noms des personnes qui gagnent plus de 5000 F".
L'attribut connu est SALAIRE. La sélection des noms vérifiant la condition ci-dessus se fera par l'analyse de l'attribut salaire, puis par le mapping de SALAIRE vers NOM.

En général, le résultat d'un "mapping" peut être utilisé dans la spécification d'un autre "mapping" ce qui permet l'expression de questions de grande complexité.

La requête précédente en SEQUEL s'exprimera :

```
SELECT NOM
FROM PERSONNE
WHERE SAL>5000
```

Il est à remarquer que ce type de langage est plus facilement utilisable pour un non informaticien que ceux définis précédemment.

4) langages graphiques

Ce type de langages destiné à des non informaticiens permet à l'utilisateur de faire un choix parmi un ensemble de possibilités que lui offre le système au fur et à mesure qu'évolue leur conversation.

Nous ne nous intéresserons pas plus précisément à ce type de langage à cause de la complexité du système à mettre en oeuvre.

QUERY BY EXAMPLE est le langage de ce type le plus connu.

IV - CHOIX D'UN LANGAGE PIVOT

Nous avons fait au cours des chapitre précédents les hypothèses suivantes :

(1) Le langage pivot sert de langage utilisateur (cf III.I.4) (ou plutôt le langage pivot doit nous laisser libre choix du langage utilisateur).

(2) Chaque BL est gérée par son SGBD propre (cf I.B.II)

(3) L'intégration se fait uniquement par les interfaces habituelles des SGBD (cf I.B.II)

Remarques préliminaires

- Tous les langages étudiés au paragraphe précédent (Calcul algébrique, QUEL, Algèbre relationnel SQUARE, SEQUEL) sont dits relationnellement complets.
  Ce qui signifie que tous ces langages sont capables d'exprimer toutes les requêtes exprimables dans le calcul algébrique [COD 71:1].
- Il ne s'agit pas de faire ici une étude formelle, mais simplement intuitive.

IV-1.- Le langage pivot sert de langage utilisateur

1) Les langages orientés "mapping"

L'utilisateur qui se sert de ce type de langage travaille uniquement avec des noms de relations et d'attributs, c'est-à-dire avec des objets et des qualités d'objets qu'il connaît. C'est par conséquent le type de langage le plus accessible à un non informaticien.

2) Les langages orientés "algèbre relationnel"

Travailler avec ce type de langage, c'est "découper" et "recoller" des relations visualisées sous forme de tables. D'une approche plus délicate que le "mapping", ce type de langage est néanmoins accessible à un utilisateur ayant bénéficié d'un apprentissage informatique minimal.

3) Le calcul algébrique

D'une approche mathématique, ce type de langage peut difficilement être utilisé comme seul langage utilisateur.

On peut cependant remarquer qu'un langage tel que QUEL, pris sans quantificateur, est d'un usage simple très proche de SEQUEL.

IV-2.- Chaque BL est gérée par son SGBD propre et son intégration à la BDR se fait par les interfaces habituelles des SGBD

Définition :

Nous dirons qu'un SGBD est intégrable à "SGMB" si une phrase écrite en langage pivot est traductible en une phrase du langage de ce SGBD.

IV-3.- Etude de la réduction d'une requête orientée "mapping" avant son exécution

IV-3.1.- Etude d'un exemple

Soit le schéma :

Fournisseur (NO #, NOM, VILLE)
Stock (NO # , NOP #, QT-STOCK)

Pièce (NOP #, P-NOM)
Projet (NØ-PR #, VILLE)
Approvisionnement (NØ-PR #,NØ #,NOP #, QT)

Soit la requête QUEL :

```
Q1 = RANGE OF (S,P,V) is (Fournisseur, Pièce, stock)
     RETRIEUE (S.S. norme)
     WHERE (S. VILLE = 'Paris')
     AND (P.P- NOM = 'Ecrous')
     AND (V. NO # = S-NØ #)
     AND (V. NOP # = P.NØP #)
     AND (V.QT-STOCK>1000)
```

qui signifie "trouver les noms de tous les fournisseurs de Paris qui disposent d'un stock de plus de mille écrous.

Il est nécessaire de décomposer cette requête pour son exécution. Pour cela un algorithme a été développé par WONG et YOUSSEFI dans le système INGRES [WON 76]. Cet algorithme décompose la requête comme suit :

Q_{11} :
```
RANGE OF (V,P) is (Stock, Pièce)
RETRIEUE INTO STOCK 2 relation temporaire (V.NØ #)
WHERE (V.NØ # P.NØP #)
AND (V.Q-STØCK>1000)
AND (P.P-NOM = 'Ecrous')
```

Q_{12} :
```
RANGE OF (S,V) is (Fournisseur, STOCK 2)
RETRIEUE (S.NOM)
WHERE (S.NØ # = V.NØ #)
AND (S.VILLE = 'Paris')
```

On voit de façon évidente que Q_{11} est formée d'un JOIN sur les relations STOCK et PIECE suivi de deux SELECTION et que Q_{12} est formé d'un JOIN entre la relation résultat de Q_{11} et la relation PIECE suivi d'une SELECTION.

Il semble par conséquent pertinent de faire le parallèle entre cet algorithme de réduction et la traduction du langage en algèbre relationnel.

L'étude de l'algorithme de décomposition associé au langage SEQUEL nous aurait amené à la même remarque.

On peut remarquer de même que la traduction d'une requête, écrite à l'aide d'un langage orienté "mapping", en une requête, décrite dans un autre langage du même type, est simple si cette traduction se fait à partir de la requête réduite.

On peut affirmer que cette traduction se fait en passant par un état ou la requête de départ est très proche d'une requête en algèbre relationnelle.

IV-3.2.- Conclusion

Le principal problème posé par l'implémentation d'un langage relationnel utilisateur est celui de la décomposition d'une requête utilisateur en sous-ensembles connexes et exécutables, cohérents entre eux.

A l'issue de cette étude, nous avons le sentiment que ce problème est le même que celui de la traduction d'un tel langage en algèbre relationnel.

C'est pourquoi nous choisirons l'algèbre relationnel comme langage pivot, cette solution nous laissant de plus toutes les possibilités de choix d'un langage externe éventuel.

IV-4.- Fondements théoriques

Une définition d'une relation décomposable [DEL 80]

Soit la relation R[X].

Soit Y et Z tels que $X = Z \cup Y$.

R est décomposable si et seulement si :

R = JOIN (PROJECT(R,Z), PROJECT(R,Y)).

Ce qui s'interprète aussi par :

Les opérations JOIN et PROJECT préservent l'intégrité de la base et plus généralement par :

Le modèle relationnel qui donne un mécanisme structurant de décomposition du schéma global d'une base de données fournit les outils nécessaires à l'exploitation de cette base, outils nous assurant de la conservation de l'intégrité de la base.

Par conséquent :

Tout langage désireux d'assurer l'intégrité des données qu'il manipule doit faire référence à ces opérateurs relationnels (ou au calcul algébrique), c'est pourquoi il ne nous semble pas aberrant de dire que :

"Un SGBD de type relationnel est capable d'assimiler une phrase écrite à l'aide des opérateurs relationnels".

V - EXEMPLES DE TRADUCTEURS

Nous allons dans ce paragraphe écrire les principaux opérateurs relationnels (PROJECT, JOIN, SELECT, UNION, INTERSECTION forment un sous ensemble de l'algèbre relationnel dans lequel toute requête est exprimable [COD 71:1]) à l'aide de phrases exprimées dans les principaux langages relationnels connus.

Références bibliographiques de ces langages :

Calcul algébrique : DAT 78
COD 71:2

QUEL : DAT 78

SEQUEL : DAT 78
AST 75

Dans le tableau qui suit :

R,S,r,s sont des relations
A,B représentent des listes d'attributs
a,b un attribut.

Algèbre relationnel	Calcul algébrique	QUEL	SEQUEL
PROJECT (R,A)	r[A] : r ∈R	RANGE of r is R RETRIEUE : r.A	SELECT A FROM R
SELECT (R,a op b)	r:r∈R∧(r[a]op r[b])	RANGE of r is R RETRIEUE r WHERE(r.a op r.b)	SELECT * FROM R WHERE a op b
SELECT (R,a op constante)	r:r∈R∧(r[a] op Cste)	RANGE of r is R RETRIEUE r WHERE(r.a op cste)	SELECT * FROM R WHERE a op Cste
JOIN (R,S,a op b)	r⌢s:r∈R∧s∈S ∧r[a] op r[b]	RANGE of r,s is R,S RETRIEUE r⌢s WHERE(r.a op r.b)	SELECT * FROM r.R,s-S WHERE r.a op r.b
UNION (R,S)	r:r∈R∨r∈S	symbole Union	Symbole Union
INTERSECTION (R,S)	r:r∈R∧r∈S	symbole intersection	symbole intersection

VI - CONCLUSION

Nous avons porté notre choix sur l'algèbre relationnel comme langage pivot. Ce langage est plus qu'un langage permettant la manipulation de données décrites dans le modèle relationnel, il fait partie de ce modèle en assurant l'intégrité des données manipulées. C'est par conséquent une référence pour les autres langages relationnels développés, comme nous avons essayé de la démontrer intuitivement en faisant le parallèle entre la décomposition d'une requête complexe en sous-requêtes connexes avec la traduction du langage QUEL en algèbre relationnel.

CHAPITRE IV

"SGMB" : ALGORITHME DE DÉCOMPOSITION DE REQUÊTES ET DE LOCALISATION DE SOUS-ARBRES

I - INTRODUCTION

S'adresser à la base de données répartie, c'est poser une question en terme du schéma conceptuel.

Une telle question, on parle de requête, peut nécessiter pour son exécution un accès à plusieurs bases de données locales. La requête globale sera par conséquent décomposée en un ensemble de sous requêtes locales.

I-1.- Etat de l'art

Nous avons dans le chapitre II, exposé les problèmes de conception d'un schéma conceptuel global à partir de schémas locaux.

Nous avons dit que les liens sémantiques entre objets de base différentes se faisaient à l'aide de "jointures" mises en évidence par les propriétés des dépendances fonctionnelles.

Les principales études sur la décomposition de requêtes menées jusqu'à maintenant se sont déroulées dans ce cadre [CAL 76].

Une interrogation simultanée de plusieurs bases de données se posera en termes du schéma global et en particulier en termes de "jointures". La mise en évidence d'une jointure dans une requête sera la mise en évidence d'une charnière entre deux bases de données, c'est-à-dire de la nécessité de décomposer la requête.

Décomposer la requête revient à traduire les termes du schéma global en terme de schémas locaux.

Les relations locales sont alors localisées par leur appartenance à tel ou tel schéma local.

Exemple : (inspiré de CAL 76)

Soit dans B_1 : HOPITAL (NOM, ADRESSE, RESPONSABLE)

dans B_2 : CLINIQUE (NOM-CLIN, LITS)

Les constituants NOM et NOM-CLIN sont synonymes.
La vue globale peut être :

CENTRE-DE-SOINS (NOM, ADRESSE, RESPONSABLE, LITS)

CENTRE-DE-SOINS est une jointure entre HOPITAL et CLINIQUE.

Toute requête s'exprimant en terme de CENTRE-DE-SOINS sera décomposée en une requête sur la relation HOPITAL et une autre sur la relation CLINIQUE (l'une de ces deux requêtes peut être vide).

Que se passe-t-il lorsqu'une jointure n'est pas définie explicitement dans le schéma global ?

C'est en particulier le cas lorsqu'une relation est décrite dans deux bases locales avec le même schéma et la même syntaxe.

L'expérimentation de Polyphème [POL 80], [SIR 80] a prouvé que ce problème apparemment simple est un problème réel. (Il est alors nécessaire de renommer globalement chaque relation locale de façon à les distinguer. Peut-on encore parler d'intégration des bases de données préexistantes à la base globale ?).

I-2.- L'approche "SGMB"

Dans l'approche classique, les seules informations de localisation des données étaient celles connues du système, l'utilisateur ne pouvant, d'un schéma global, tirer de telles informations.

Dans l'approche base-ensemble de base l'utilisateur connaît la localisation, logique évidemment, des relations qu'il veut manipuler. Cette localisation se traduit simplement par le fait qu'une relation appartient à une certaine base de données.

Ce sont ces informations de localisation qui seront utilisées

par l'algorithme de décomposition de requête développé dans les paragraphes suivants.

II - DEFINITIONS

- Les opérateurs relationnels dans "SGMB"

Les opérateurs relationnels (sous-ensemble décrit dans le chapitre précédent) sont des opérations qui, appliquées à une ou deux relations, ont pour résultat une autre relation.

- Les relations de base

Les relations de base sont les relations décrites à la conception de la base (dans le schéma conceptuel). Elles donnent à l'utilisateur une vue statique de la base. Notation : RB.

- Les relations temporaires

Une relation temporaire est une relation obtenue dynamiquement par l'application d'un opérateur algébrique, ou d'une composition de ceux-ci à une ou plusieurs relations de base. Le schéma de la relation temporaire est obtenu par déduction (en fonction de la suite d'opérateurs appliqués) à partir des schémas des relations de base mises en cause. Notation : RT.

Remarque : Le schéma d'une relation temporaire ne pouvant être défini qu'à l'analyse d'une requête pose des problèmes de représentation.

Exemple :

Soit la relation Personne (NOM, PRENOM, AGE, ADRESSE).
La requête PROJECT (Personne, NOM, ADRESSE) a pour résultat

une relation temporaire dont le schéma est : RT (NOM, ADRESSE).

Cette relation sera, ou délivrée comme résultat à un utilisateur, ou réutilisée pour la suite de l'exécution de la requête dont ce PROJECT est un sous-arbre, un sous-ensemble.

- Qu'est-ce qu'une requête dans "SGMB" ?

Une requête est une combinaison fonctionnelle d'opérateurs (pris parmi les opérateurs décrits au chapitre précédent) qui s'exécute dans un certain contexte.

Le contexte :

L'utilisateur désirant accéder à la base de données répartie associera dans ce contexte à chaque relation de base qu'il utilise la localisation, c'est-à-dire la base locale, ou est décrite cette relation.

Ce contexte sera utilisé pour différencier deux relations de même nom dans deux bases différentes.

Exemple de requête utilisateur

Reprenons l'exemple du chapitre II, C, § II.

Soit la question : "Trouver tous les cinémas et les restaurants d'une même rue".

La requête associée sera :

Contexte : R pour R dans RESTAURANTS
C pour C dans CINEMAS

Requête algébrique : JOIN (R,C,RUE = RUE).

Il est à noter que de la même façon que l'on associe un contexte à une requête, on peut associer un contexte à chaque opérateur faisant partie de la requête. Ce contexte étant défini par

l'ensemble des bases qu'il est nécessaire d'accéder pour exécuter cet opérateur.

C'est cette propriété qui sera utilisée par l'algorithme de décomposition de requête et en particulier par l'algorithme de localisation de sous-arbres (cf IV-3).

III - REPRESENTATION SOUS FORME ARBORESCENTE D'UNE REQUETE

Structure de l'arbre :

- Chaque noeud de l'arbre représente un opérateur algébrique. Si l'opérateur est binaire, ce noeud aura deux fils, s'il est unaire, il n'en aura qu'un.
- Chaque fil représente le(s) accès aux données nécessaires à l'exécution du noeud père.
- Les feuilles de l'arbre représente des accès aux données stockées dans les bases de données locales (aux relations de base). Dans le cas contraire la requête n'est pas exécutable.

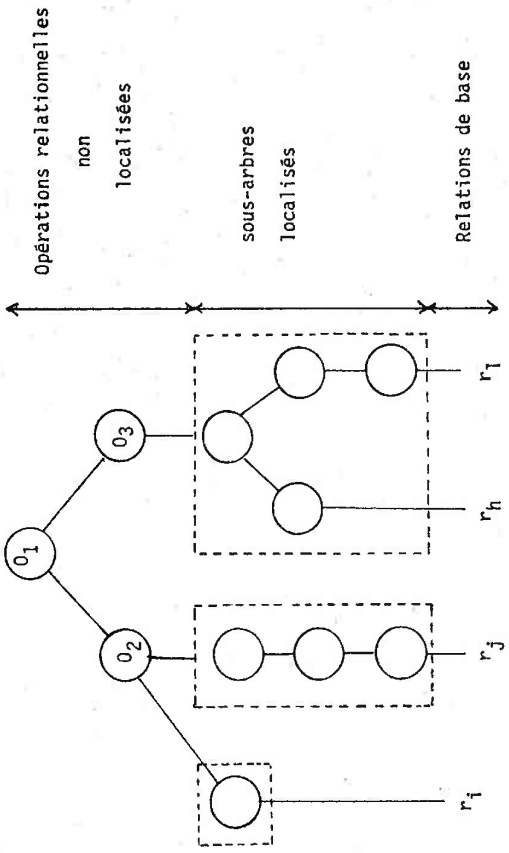
III-1.- Notion de sous-arbre localisé

Un sous-arbre est dit localisé si toutes les données nécessaires à son exécution sont stockées dans une base de données locale et une seule.

C'est un sous-arbre dont le contexte associé à la racine est de cardinalité = 1.

III-2.- Opérations relationnelles non localisées

Ce sont des opérations qui nécessitent des accès à plusieurs bases de données pour leur exécution.



Architecture arborescente d'une requête

III-3.- Architecture arborescente d'une requête

On distinguera trois niveaux :

- Les accès aux relations de base.
- Les sous-arbres localisés : l'intérêt d'une telle localisation sera exprimé dans le chapitre suivant sur l'optimisation d'une requête.
- Les opérations relationnelles non localisées. Ce sont des opérations qui travaillent sur plusieurs bases de données.

IV - OPERATEURS GLOBAUX ET OPERATEURS LOCAUX

IV-1.- L'opération globale de G-LECTURE

Elle fait le joint entre la partie globale d'une requête et un sous-arbre localisé.

Elle signifie un accès logique à une base de données locale.

IV-2.- Règles de décomposition

Ces règles ont pour but de permettre une première distinction entre les opérations relationnelles globales et les opérations localisées. Elles ne tiennent pas compte du contexte dans lequel s'exécute une requête (Opération globale = G-opération).

Notations : OPU = opérateur unaire (SELECT|PROJECT).
OPB = opérateur binaire (JOIN|UNION|INTERSECTION).

1 - Opérations unaires sur des relations de base

OPU (RB) ⊢—— G-LECTURE (OPU(RB))

2 - Opérations unaires sur des relations temporaires

OPU(RT) ⊢—— G.OPU(RT)

3 - Opérations binaires sur des relations de base

OPB(RB_1,RB_2) ⊢—— G-OPB(G-LECTURE(RB_1),G-LECTURE(RB_2))

4 - Opérations binaires sur des relations temporaires

OPB(RT_1,RT_2) ⊢—— G-OPB(RT_1,RT_2)

5 - Opérations binaires composées

Une opération binaire composée est une opération sur une relation de base et une relation temporaire.

OPB(RB,RT) ⊢—— G-OPB(G-LECTURE(RB),RT)

OPB(RT,RB) ⊢—— G-UPB(RT,G-LECTURE(RB))

IV-3.- Règles de localisation de sous-arbres - Gestion du contexte

Rappels :

- Le contexte d'une requête est l'ensemble des bases auxquelles la requête nécessite des accès pour son exécution.

- La localisation d'une relation est la base sur laquelle cette relation est stockée. Cette localisation est donnée par l'utilisateur.

- L'application d'une opération algébrique ou d'une G-LECTURE à une ou plusieurs relations de base a pour résultat une relation temporaire.

D'où les règles

1 : Contexte (OPU,RB) = localisation (RB).

2 : Contexte (G-LECTURE, RB) = localisation (RB).

3 : Contexte (G-OPB, RT_1, RT_2) = contexte ($!T_1$)Ucontexte(RT_2)

4 : Contexte (G-OPU,RT) = contexte (RT).

5 : Contexte (G-LECTURE, RT) = contexte (RT .

Sous-arbre localisé

Tout noeud tel que le contexte qui lui est associé est de cardinalité un est dit localisable et il est localisé par le contenu de ce contexte.

Le contenu du contexte est le nom logique de la base de données à laquelle il nécessite des accès.

Ce sous-arbre peut être exécuté localement, c'est-à-dire que la base locale nommée dans son contexte suffit à son exécution.

IV-4.- Principe de l'algorithme

Phase I : Il s'agit de construire l'arbre représentant la requête. Une requête étant une opération algébrique bien parenthésée, le problème est classique.
Au cours de la même phase, on appliquera les règles de décomposition, il s'agira surtout d'intégrer à la requête les demandes d'accès logiques aux bases locales (opérations de G-LECTURE).
Réaliser cette intégration, c'est surtout faire la distinction entre les relations temporaires et les relations de base.
Syntaxiquement, une relation temporaire se distingue d'une relation de base par le fait qu'elle nécessite

elle-même des accès aux bases locales (PROJECT, SELECT, JOIN, UNION, INTERSECTION).

Exemple : Personne (NOM, PRENOM, ADRESSE)
Voiture (NUMERO, DEPARTEMENT, NOM)

Soit la requête

"Trouver les adresses de tous les propriétaires de voitures immatriculées dans les Vosges"

PROJECT (JOIN (Personne, SELECT (Voiture, DEPARTEMENT = '88'), NOM = NOM), ADRESSE).

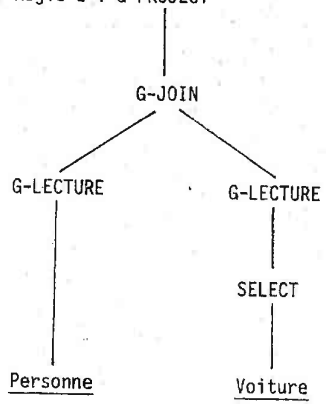
Remarque : Dans cette phrase, nous ne nous servons pas de ce contexte.

L'analyse donnera :

Analyse	Résultat
PROJECT (JOIN (la rencontre du JOIN nous dit que nous allons travailler sur une relation temporaire).	Règle 2 : G-PROJECT
PROJECT (JOIN(Personne, SELECT	Règle 5 : G-PROJECT G-JOIN G-LECTURE Personne

PROJECT (JOIN(Personne, SELECT(Voiture,DEPARTEMENT = 88),

Règle 1 : G-PROJECT

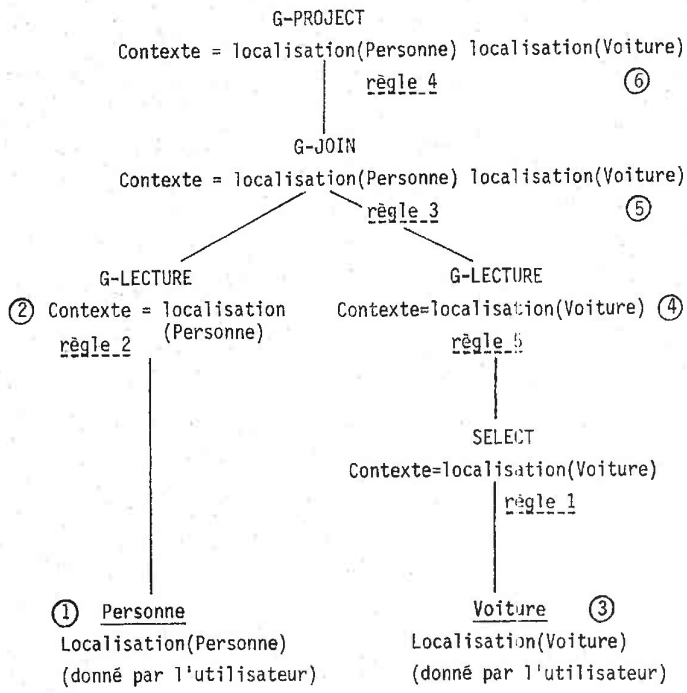


PROJECT(JOIN(Personne, SELECT(Voiture,DEPARTEMENT = 88), NOM = NOM, ADRESSE)

Cette dernière phase permet la connaissance complète des différents critères. Il semble intéressant de mettre la requête sous une forme postfixée dans une phase préliminaire.

Phase II : Il s'agit, en partant des feuilles de décorer l'arbre représentant la requête en associant à chaque noeud un contexte. On associe à chaque feuille la localisation de la relation accrochée puis on applique les règles de localisation de sous-arbre.

Reprenons l'exemple précédent. L'arbre décoré est :



Les chiffres entourés ordonnent [ⓝ] la suite des opérations (des applications des règles de localisation).

- Interprétation de l'arbre décoré

1) Tout opérateur global qui s'exécute dans un contexte de cardinalité = 1 peut s'exécuter localement. Soit

G-OPU(RT) et card(Contexte)=1 ⊢—— OPU(RT)

G-OPB(RT_1,RT_2) et card(Contexte)=1 ⊢— OPB(RT_1,RT_2)

2) Cas particulier des G-LECTURE

Nous avons vu le rôle particulier des G-LECTURE. Une G-LECTURE sert à faire la liaison entre la partie globale de l'arbre de requête et un sous-arbre localisé. Une G-LECTURE signifie en fait un accès à une base locale. Nous devons, à la reconnaissance d'une racine de sous-arbre localisé traduire :

G-OPU(RT) par G-LECTURE

|

OPU(RT)

et G-OPB(RT) par G-LECTURE

|

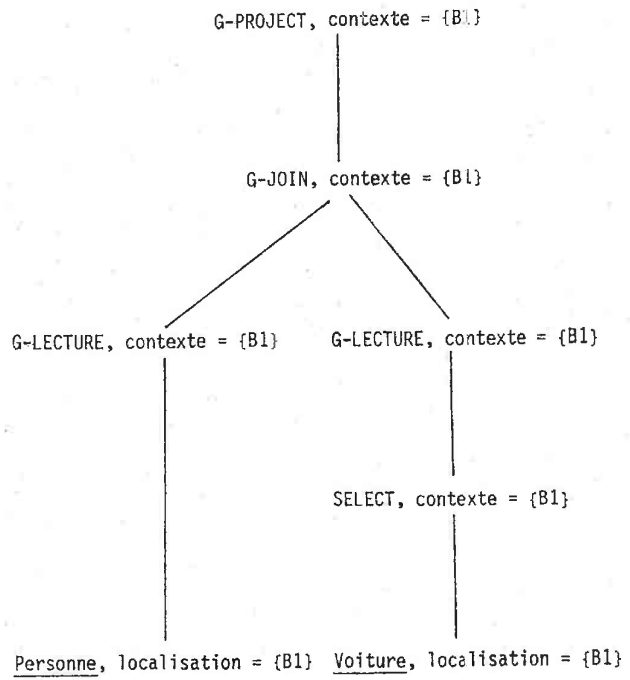
OPB(RT)

Remarque : Une G-LECTURE dans un sous-arbre localisé n'a plus de sens. L'arbre sera raccourci.

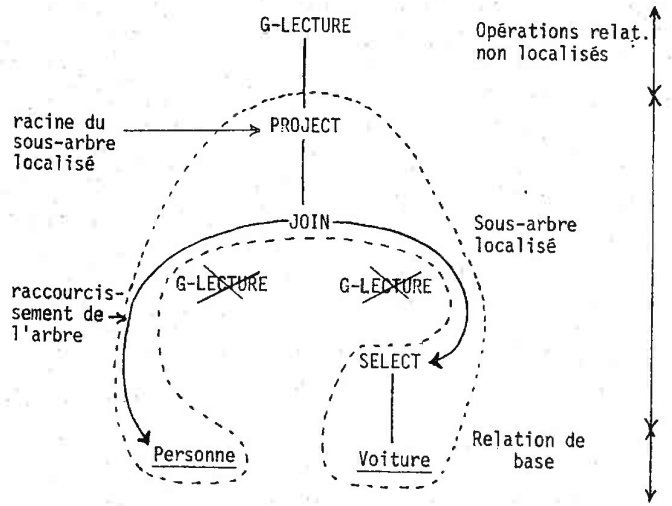
Application

Supposons Personne et Voiture localisés sur la base B1.

* L'arbre décoré de l'exemple précédent est :



* il sera interprété par :



----- limite du sous-arbre localisé.

La requête précédente sera donc totalement exécutée localement, le seul travail de "SGMB" étant de rendre le résultat à l'utilisateur.

Remarque : Le classicisme de la structure utilisée (un arbre binaire) et la simplicité des règles de décomposition et de localisation semble en faire un algorithme facilement implémentable.

CHAPITRE V

"SGMB" : DÉCOMPOSITION DE REQUÊTES ET OPTIMISATION

I - NECESSITE DE L'OPTIMISATION

I-1.- Un SGBDR relationnal est d'abord un SGBD relationnel

L'une des caractéristiques essentielles du modèle relationnel est le principe de "data indépendance", ce qui signifie qu'un utilisateur manipule les données indépendamment de leurs structures internes, donc sans préciser les chemins d'accès nécessaires à l'obtention de ces données.

Qu'est-ce alors qu'un SGBD relationnel ?

Nous dirons que c'est un SGBD classique (logiciel permettant de décrire, manipuler et stocker des données) qui est capable d'inferfacer les opérateurs relationnels algébriques (ou un langage équivalent).

C'est la raison pour laquelle le principal problème dans l'implémentation d'un SGBD relationnel est un problème de performance qui nécessite un souci d'optimisation dans la décomposition et l'exécution d'une requête, l'étude de l'accès aux données et le stockage physique des données.

I-2.- Problèmes inhérents à la répartition

Faire coopérer plusieurs SGBD, c'est assurer un échange important de données entre plusieurs logiciels. Ces échanges coûtant très chers en temps et en argent, il va, par conséquent, s'agir de diminuer leur volume.

II - CRITERES D'OPTIMISATION DE "SGMB"

Il s'agit ici d'optimisation à la décomposition de la requête, c'est-à-dire d'optimisation statique.

II-1.- Diminuer le nombre d'opérations élémentaires exécutées par "SGMB"

Statiquement, en réorganisant l'arbre résultat de l'analyse d'une requête.

II-2.- Diminuer le nombre de passages par les interfaces des différents SGBD

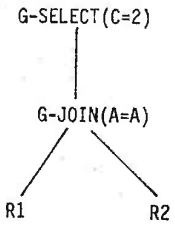
C'est le but de la localisation de sous-arbres (cf chap. IV).

II-3.- Diminuer la quantité d'informations échangées entre les SGBD et "SGMB"

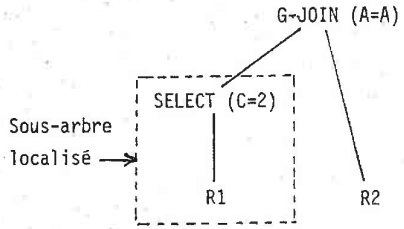
1 - En réorganisant l'arbre de décomposition.

Exemple : Soit R1(A,B,C) dans la base B1
R2(A,E,F) dans la base B2.

La requête SELECT (JOIN(R1,R2,A=A),C=2) est équivalente a :



Lequel arbre est équivalent à :



L'inversion des opérations SELECT et JOIN permet la diminution des transferts de la base B1 vers "SGMB". (On ne transfère plus que les tuples vérifiant le critère de restriction C=2).

2 - en localisant des sous-arbres

III - TECHNIQUES D'OPTIMISATION DE "SGMB"

III-1.- Réorganisation de l'arbre de décomposition d'une requête

III-1.1.- Principe

- L'opérateur PROJECT restreint le schéma de la relation qu'il a en paramètre.
- L'opérateur SELECT restreint le nombre de tuples de la relation qu'il a en paramètres.

- L'opérateur JOIN et les opérateurs ensemblistes sont les opérations qui nécessitent le plus d'opérateurs élémentaires.

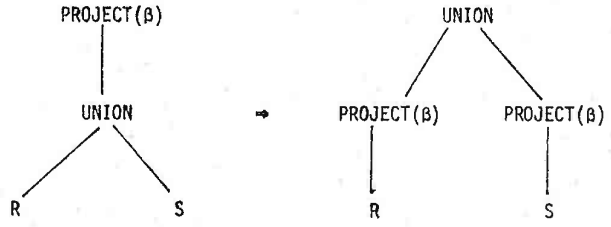
Le but est d'ordonnancer ces opérateurs de façon à minimiser le nombre d'opérations élémentaires tout en s'assurant de ne pas modifier la sémantique de la requête.

III-1.2.- Etude de la commutativité des opérateurs relationnels [SMI 75]

Dans tout ce qui suit :

- α représente l'ensemble des attributs sollicités dans l'expression booléenne associée à une opération SELECT ou JOIN.
- β représente l'ensemble des attributs sollicités par une opération PROJECT.
- $\Rightarrow$ la flèche double représente le sens de transformation de l'arbre, c'est-à-dire le sens où il y a optimisation.

1) Commutativité de PROJECT et UNION



2) PROJECT et INTERSECTION ne sont pas commutatifs

Exemple : R_1

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3

R_2

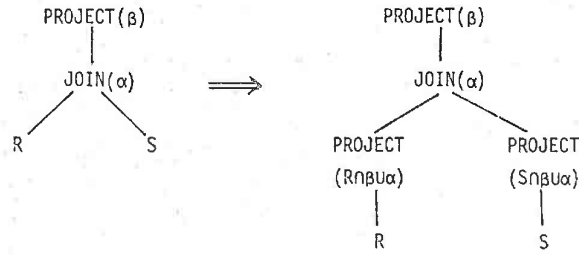
A	B	C
a_2	b_1	c_1
a_2	b_2	c_2
a_2	b_3	c_2

$PROJECT(INTERSECTION(R_1,R_2),B,C) = \{b_2c_2\}$

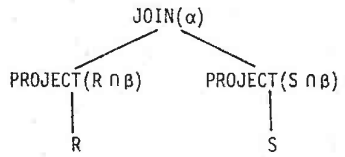
$INTERSECTION(PROJECT(R_1,B,C),PROJECT(R_2,B,C)) = \{b_1c_1,b_2c_2\}$.

Ce contre-exemple le montre.

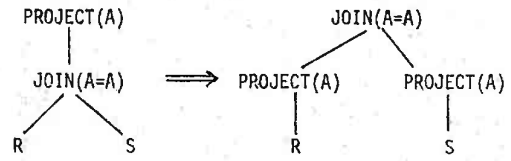
3) Commutativité de PROJECT et JOIN



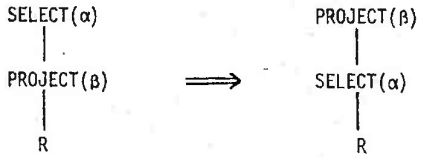
Remarque : Si $\alpha = \beta$ alors l'arbre optimisé se réduit à :



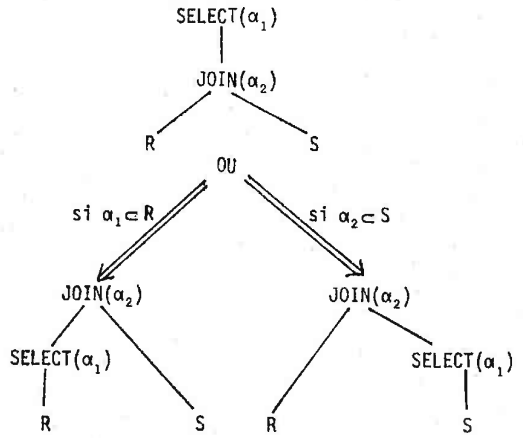
Exemple : Soient R(A,B,C), S(A,X,Y)



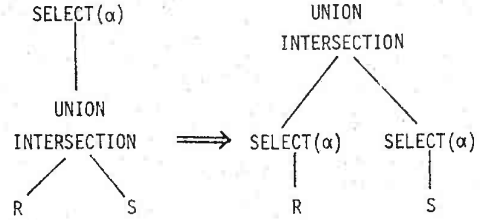
4) Commutativité de SELECT et PROJECT



5) Commutativité de SELECT et JOIN



6) Commutativité de SELECT avec UNION ou INTERSECTION



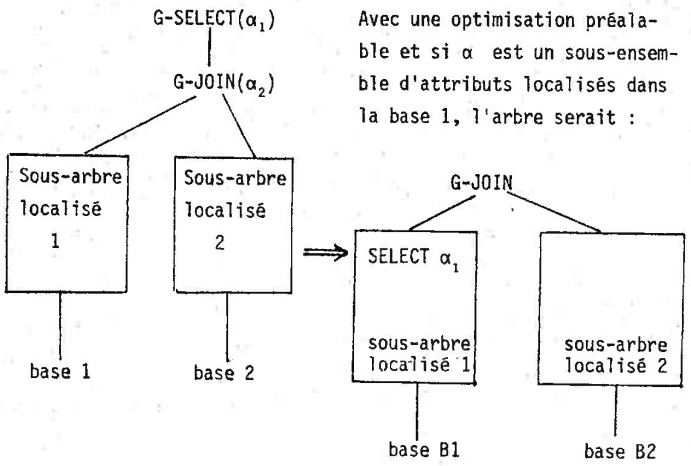
III-1.3.- Application

L'application de ces propriétés des opérateurs relationnels et ensemblistes, qui est développée dans les SGBD relationnels mono-bases, permet une meilleure organisation de l'arbre représentant une requête et devant être exécutée en assurant une amélioration des performances par la minimisation du nombre d'opérations élémentaires.

Elle permet également une minimisation de l'espace mémoire nécessaire à l'exécution d'une requête. C'est le cas lorsque les opérations à exécuter sont des opérations globales.

Elle permet en outre une meilleure localisation des sous-arbres (comme le montre l'exemple ci-dessous) et par conséquent une minimisation des transferts.

Exemple :



Avec une optimisation préalable et si α est un sous-ensemble d'attributs localisés dans la base 1, l'arbre serait :

III-2.- Localisation de sous-arbres

La technique de localisation est développée dans le chapitre IV sur la décomposition de requête.

Cette localisation de sous-arbres permet :

- une minimisation des passages par les interfaces.
- une minimisation de la quantité des échanges entre SGBD locaux et "SGMB" en augmentant le nombre d'opérations exécutées localement.
- une amélioration des performances en assurant un certain parallélisme entre les exécutions des différentes branches de l'arbre représentant la requête.

Il est à remarquer que deux sous-arbres localisés différents peuvent s'exécuter dans le même contexte et qu'une réorganisation

judicieuse de l'arbre représentant la requête aurait pu conduire à la fusion de ces deux sous-arbres localisés, c'est-à-dire diminuer encore les échanges entre la partie globale et la partie locale de la requête.

Cette étude n'a pas encore été entreprise mais pourra servir de cadre à un souci d'optimisation plus approfondi.

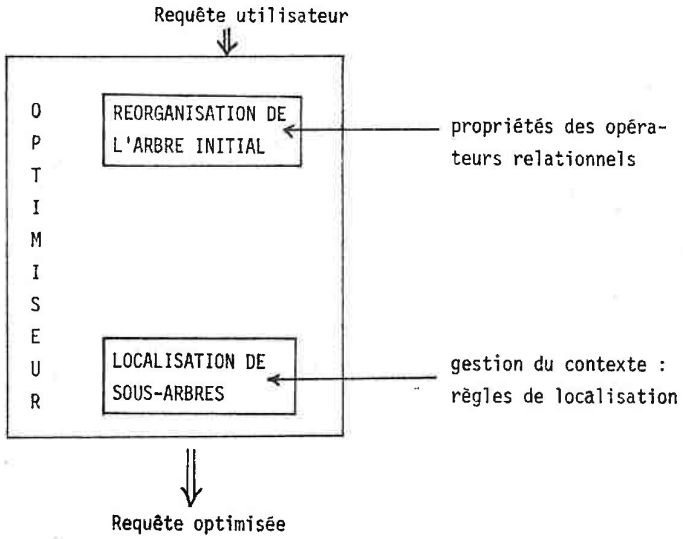
IV - ORGANISATION DE L'OPTIMISATION

(1) La réorganisation de l'arbre par l'étude des propriétés des opérateurs relationnels est indépendante de la répartition c'est la première couche de l'optimisation.

(2) Une première distinction entre opérateurs globaux et locaux se fait grâce à l'utilisation des règles de décomposition.

(3) La localisation de sous-arbre est optimale après les couches (1) et (2) de l'optimisation.

L'organisation de l'optimisation sera donc celle exprimée par la figure suivante :



CHAPITRE VI

"SGMB" : EXÉCUTION D'UNE REQUÊTE

I - PRINCIPE

Au sortir du module de décomposition, une requête se présente comme un arbre constitué d'un arbre global, comprenant des opérateurs globaux, et d'un ensemble de sous-arbre localisés comprenant chacun des opérateurs locaux exécutables par une même machine (ce sera généralement un SGBD) et utilisant des relations localisées dans cette machine.

Au niveau global, celui de "SGMB", l'exécution d'une requête se traduira par :

- Une mise en forme de chaque sous-arbre localisé : une forme transportable par le moyen de communication utilisé et interfaçable par le SGBD distant.
- La récupération de la relation résultat de l'interprétation de ce sous-arbre.
- L'exécution des opérateurs globaux.

II - MISE EN FORME D'UN SOUS-ARBRE LOCALISE

Un sous-arbre localisé est l'image d'un algorithme composé d'oparateurs relationnels s'appliquant à des relations.

Assurer l'exécution d'une requête, c'est en partie permettre l'exécution des sous-arbres localisés, ce qui nécessite le transport d'algorithmes entre "SGMB" et les SGBD locaux.

Un algorithme n'est transportable via un moyen de communication quelconque que sous une forme codée. Cette forme est définie par un accord entre l'émetteur et le récepteur, par un protocole d'échange.

Mettre en forme un sous-arbre localisé, c'est traduire ce sous-arbre en termes du protocole d'échange.

Des procédures d'échange ont été développées précédemment com-

II-1.- Protocole d'échange de "SGMB" (protocole permettant l'accès aux données distantes)

Un SGBD est un monde fermé accessible pour l'extérieur seulement par un sas difficile à franchir, il s'agit de l'interface d'accès.

Il permet aisément le passage de données dans un sens ou dans l'autre, mais difficilement celui de programmes.

Nous avons dit qu'un sous-arbre est un programme et celui-ci doit être exécuté par un SGBD distant.

L'idée est de traduire le protocole d'échange, qui permet le transport d'algorithmes, en termes de relations.

Les SGBD étant relationnels, on peut affirmer que ces relations, donc ces algorithmes seront facilement exploitables, à l'aide de programmes pertinents, par ces SGBD.

En outre, cela peut permettre la définition de procédures sophistiquées donnant l'état des différentes transactions en cours.

Exemple : la transaction numéro X est-elle terminée ?

Nous n'aurons plus qu'un seul type de message en transit sur la communication. Ces messages seront des messages d'insertion d'un n-uple dans une relation.

Objets nécessaires au transport de sous-arbres

NT : numéro de transaction

Classiquement, une transaction est un ensemble de requêtes qui doivent s'exécuter sans interruption de façon à assurer le maintien de la cohérence de la base.

Nous allons utiliser cette opportunité pour faire le lien entre les différents opérateurs d'un même sous-arbre localisé.

NØ : numéro de l'opérateur

Il permet l'ordonnancement des opérateurs appartenant à un même sous-arbre.

COM : nom de la commande

Ex : PROJECT, JOIN,...

R1 : nom de la 1ère relation sur laquelle s'applique la commande.

R2 : nom de la 2ème solution sur laquelle s'applique la commande.

NC : permet l'association d'un critère à une commande.

Ex : Select (R1, A=2). A = 2 est le critère.
Les critères (de sélection ou de jointure) seront distingués entre eux par ce numéro.

AT, AT1, AT2 : numéro d'attribut
Cela signifie que chaque attribut sera globalement distingué.

OP : opérateur
Une requête peut contenir des opérateurs.

Pour pouvoir prendre en compte les algorithmes transmis, chaque SGBD doit avoir à sa disposition la base de données suivante :

TRANSACTION (NT, ETAT)
COMMANDE (NØ, NT, COM, R1, R2)
CRITERE (NØC, NØ, NT, AT1, OP, AT2)
PROJ-ATR (AT, NØ, NT).

Cette base de données permet le stockage de toutes les combinaisons d'opérateurs de l'algèbre relationnelle.

Transport de l'opérateur d'insertion

(format du message)

INSERT	NR	tuple

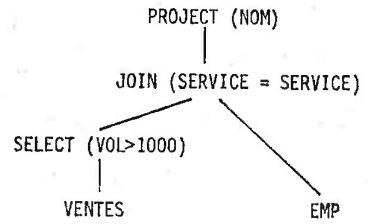
NR : numéro de la relation dans laquelle on veut insérer un tuple.

II-2.- Exemple

Soit le schéma suivant :

EMP (NOM, SAL, MGR, SERVICE)
VENTES (SERVICE, VOL).

Le sous-arbre localisé par "SGMB" suivant



avec la numérotation des attributs :

A1 : NOM
A2 : SAL
A3 : MGR
A4 : SERVICE DE EMP
A5 : SERVICE DE VENTES
A6 : VOL

et la numérotation des relations :

R1 : TRANSACTION
R2 : COMMANDE
R3 : CRITERE
R4 : PRØJ-AT

sera traduit par :

INSERT	R1	1				
INSERT	R2	1	1	SELECT	VENTES	
INSERT	R3	1	1	1	A6	>	1000
INSERT	R2	2	1	JOIN	RT11	EMP	
INSERT	R3	2	2	1	A5	=	A4
INSERT	R2	3	1	PROJECT	RT13		
INSERT	R4	A1	3	1			

RT_{ij} signifie que cette relation est le résultat de l'application de la commande i de la transaction j.

III - LA RECUPERATION DES RESULTATS ASSOCIES AUX SOUS-ARBRES LOCALISES

La terminaison d'une transaction (d'un sous-arbre localisé) par un SGBD se traduira par l'envoi de la relation résultat, via un moyen de communication, à "SGMB". L'association entre la relation résultat de l'exécution d'un sous-arbre et ce sous-arbre se fera par l'intermédiaire du numéro de transaction, connu de façon unique par "SGMB".

Il faut ici noter que l'utilisation d'un SGBD relationnel classique suffit à jouer le rôle qui est alors demandé à "SGMB".

Ce SGBD s'exécute en prenant ses ressources dans une base de données temporaire qui contiendrait les données émises des SGBD distants.

Cette idée est développée dans un projet qui développe un SGBDR sur MULTICS et nous la reprenons en développant notre propre SGBD relationnel autour du système TYP (cf Chapitre VII).

IV - EXECUTION DES OPERATEURS GLOBAUX

Il ne s'agit pas dans ce paragraphe d'étudier une façon de réaliser les opérateurs globaux mais d'étudier les moyens à mettre en oeuvre pour assurer l'enchaînement de ces opérateurs.

IV-1.- Gestion des relations temporaires

Un opérateur relationnel, appliqué à une ou deux relations selon qu'il est unaire ou binaire, a pour résultat une autre relation qui sera,

soit délivrée à l'utilisateur,
soit utilisée par un autre opérateur relationnel.

Le schéma d'une telle relation est une combinaison d'attributs des relations de base et a priori n'importe quelle combinaison est possible. Par conséquent il est impensable de décrire tous les cas de figure possible et la définition des schémas des relations temporaires ne peut se faire qu'à l'analyse d'une requête.

L'utilisation d'une relation universelle [DEL 80], c'est-à-dire d'une relation dont le schéma contient tous les attributs, ou au moins ceux nécessaires à l'exécution totale d'une requête, permet, par un surdimensionnement important certes, le stockage de

toutes les relations temporaires.

Une autre solution consiste à mémoriser seulement les accès nécessaires à la réalisation des opérateurs, cela des feuilles jusqu'à la racine, puis à réaliser ces accès tuple par tuple pour rendre le résultat à l'utilisateur [NAS 81].

Ce problème pas ou peu évoqué jusqu'à maintenant sera repris dans le chapitre VII sur les choix et problèmes d'implantation.

IV-2.- Besoins en synchronisation

IV-2.1.- Hypothèse

Un noeud de l'arbre ne peut s'exécuter que si ses fils ont terminé leur exécution, ce qui signifie qu'un opérateur de "SGMB" ne peut s'exécuter que si l'ensemble des ressources, qui sont les relations temporaires créées par ses fils, et qui lui sont nécessaires sont disponibles.

IV-2.2.- Synchronisation et opération de G-LECTURE

Une opération de G-LECTURE est un noeud particulier de l'arbre, c'est lui qui gère les accès aux données distantes (stockées par l'intermédiaire d'un SGBD).

L'opération de G-LECTURE réalise les actions suivantes :

(1) Envoyer à la base locale du contexte la requête (le sous-arbre localisé attache au noeud).

(2) Récupérer les résultats de cette (ces) requête (s).

L'étape (2) ne peut commencer que lorsque les données émises par la machine locale sont disponibles à "SGMB". Les interfaces d'accès aux communications permettent de gérer ce type de synchronisation.

Dans POLYPHEME, c'est le moniteur d'exécution répartie MER (DEC 80) qui assure la synchronisation entre les machines locales et la machine globale.

L'opération de G-LECTURE sera terminée, ce qui signifie que la ressource obtenue par l'exécution de cet opérateur sera utilisable par le noeud-père, lorsque l'étape (2) sera achevée.

IV-2.3.- Exécution de l'arbre global

Un opérateur global ne peut s'exécuter que si son ou ses fils ont terminé leur exécution.

Une façon d'assurer la séquentialité de ces opérateurs est de parcourir l'arbre en partant des feuilles puis de remonter vers la racine, cela de droite à gauche (ou vice-versa).

Cette solution simple est peu efficace car elle ne permet pas de prendre en compte les possibilités de parallélisme offertes par la localisation de sous-arbres.

Une façon de prendre en compte ce parallélisme est de lancer simultanément les accès aux données distantes (opérations de G-LECTURE), puis de gérer la synchronisation entre les autres opérateurs globaux dont l'ordre d'exécution dépendra de la rapidité des accès distants nécessaires à leur exécution.

Dans un but de simplification, la solution proposée est de lancer de façon simultanée tous les accès aux données distantes, puis d'exécuter l'arbre de façon séquentielle.

Cela signifie que la rapidité d'exécution d'une requête dépendra de(s) accès aux données distantes nécessaire(s) à l'exécution du premier opérateur global : le plus près des feuilles et le plus à droite (ou à gauche).

V - CONCLUSION

Nous avons considéré dans cette étude que tous les SGBD relationnels, et en particulier "SGMB" étaient capables de réaliser tous les opérateurs relationnels décrits précédemment, ce qui nous a permi de définir un algorithme d'exécution d'une requête simple.

Nous ne nous sommes pas attachés à optimiser l'exécution d'une requête tant les critères permettant une telle optimisation dynamique sont difficiles à interpréter (cardinalité des relations temporaires [NUG 78]) [SDD 77] et inexpérimentables dans le cadre centralisé et mono-utilisateur où nous nous sommes placés (Théorie de l'écoulement, charges des différents ordinateurs CEL 80).

CHAPITRE VII

PROBLÈMES ET CHOIX DE MISE EN OEUVRE

I - BASE DE DONNEES ET ABSTRACTIONS

Les bases de données sont issues du besoin d'intégration de grandes quantités d'informations. On modèlise d'abord le monde réel, puis les données sont stockées selon le modèle ainsi obtenu.

Le mécanisme le plus connu de structuration de ce monde réel est le mécanisme de normalisation de CODD, le modèle relationnel étant le seul à assurer une indépendance totale entre les données et leur mode de stockage. Ce besoin de distinction entre la spécification des données et leur implantation est à rapprocher des principes des types abstraits qui permettent de séparer complètement la spécification d'une structure de donnée, c'est-à-dire son comportement, indépendamment des choix d'implantation.

Un type abstrait est un ensemble d'objets défini par un ensemble d'opérations possibles sur chacun d'eux et de règles décrivant le comportement de cet objet (axiomes dans le cas des spécifications algébriques).

Dans une base de données, l'intégration des données est assurée par les contraintes d'intégrité, qui en décrivant une organisation, assurent le lien entre différents types de données. Nous pouvons dire que les contraintes d'intégrité décrivent le comportement sémantique de l'ensemble des données, c'est-à-dire les propriétés qui doivent vérifier à tout moment ces données. On distingue parmi ces contraintes celles que l'on appelle dépendances fonctionnelles et qui permettent la normalisation d'un schéma relationnel de celles qui assurent le maintien de la cohérence de la base de données et qui doivent être vérifiées à chaque mise à jour.

Si l'on regarde les opérations associées à un type abstrait, ces opérations n'ont pas seulement pour but de permettre l'utilisation de ce type, mais aussi de contrôler l'évolution des objets de ce type de façon à rester en accord avec la spécification de ce type.

C'est un autre point commun entre les principes des bases de

données et ceux des abstractions, c'est pourquoi nous allons essayer d'utiliser au maximum les abstractions pour réaliser "SGMB" en nous appuyant sur la langage A.T.M. développé à l'Université de Nancy I.

II - PRESENTATION ET INTERETS D'ATM

Le nom A.T.M. est l'abréviation de "Abstraction Type Modularité".

Le but ici n'est pas de revenir sur ces trois idées, le lecteur pourra se référer à [DER 74], ni même de décrire ce langage (une description détaillée est faite dans [MIN 79]) mais plutôt d'en présenter l'originalité.

Dans A.T.M. l'abstraction des données est réalisée au moyen de deux catégories d'unités de texte compilées séparément. L'une décrit l'aspect externe des objets ; elle est appelée TYPE. L'autre décrit une réalisation de cet objet conforme au TYPE ; elle est appelée CAPSULE.

La manipulation de ces données est faite dans des unités appelées MODULE, analogue au module de PARNAS [PAR 72].

Un module permet de déclarer des variables dont on donne le type. Un module est connu de l'extérieur par son interface (nom, profil de ses opérations, spécification de son comportement) qui constitue une unité séparée appelée MACHINE.

Un module doit être conforme à un certain profil défini dans une autre unité MACHINE.

A.T.M. se présente donc comme un langage modulaire où les spécifications abstraites sont compilées indépendamment des unités de réalisation. Par contre, les unités de réalisation sont compilées dans le contexte des unités d'interface abstrait correspondantes.

II-1.- L'unité TYPE

L'unité TYPE est définie par :

- un nom
- une liste d'opérations caractérisés par leurs définitions syntaxiques et la spécification de leur comportement [MIN 79].

  Cette spécification n'est pas prise en compte par le compilateur qui la considère comme un commentaire.

Exemple : Type décrivant une pile d'entiers.

```
TYPE PILENT
   $ PILE FINIE D'ENTIER
MODIF EMPILER (INTEGER) ;
   $ EMPILER UN ENTIER AU SOMMET DE LA PILE
MODIF DEPILER (INTEGER) ;
   $ PREND L'ENTIER SITUE AU SOMMET DE LA PILE
BUILT INIT ;
   $ INITIALISE UNE PILE VIDE
FUNCTION VIDE RETURNS BOOLEAN ;
   $ REPONSE A LA QUESTION LA PILE EST ELLE VIDE ?
FUNCTION PLEINE ;
   $ REPONSE A LA QUESTION LA PILE EST ELLE PLEINE ?
```

II-2.- L'unité CAPSULE

"Une capsule est une unité de compilation dans laquelle sont précisés les choix de réalisation d'un type abstrait (unité TYPE) c'est-à-dire :

- la structure interne donnée aux objets de ce TYPE
- les algorithmes concrets qui implantent les opérations définies abstraitement dans le type" [MIN 79] p. 36.

Exemple : CAPSULE réalisant le TYPE PILENT

```
CAPSULE MAPILE FOR PILENT ;
REP
   HAUTEUR : INTEGER ;
   PIEU : ARRAY(100) OF INTEGER ;
END
MODIF EMPILER (I : INTEGER) ;
BEGIN
   HAUTEUR := HAUTEUR+1 ;
   PIEU (HAUTEUR):= I ;
END
MODIF DEPILER → I : INTEGER
BEGIN
   I := PIEU (HAUTEUR) ;
   HAUTEUR := HAUTEUR-1
END
BUILD INIT ;
   BEGIN
      HAUTEUR := 0
   END
FUNCTION VIDE RETURNS BOOLEAN ;
   BEGIN
   IF HAUTEUR = 0 THEN RETURN TRUE ;
                  ELSE RETURN FALSE ; ENDIF ;
   END
FUNCTION PLEINE RETURNS BOOLEAN ;
   BEGIN
   IF HAUTEUR > = 100 THEN RETURN TRUE ;
                      ELSE RETURN FALSE ; ENDIF ;
   END
```


II-3.- L'unité MACHINE

"Une machine est une unité de texte définie par
- un nom qui permet au système de fabrication de programmes de l'identifier.
- une liste d'opérations définies par leurs spécifications syntaxiques.
- des spécifications de fonctionnement. Elles sont aussi considérées comme des commentaires par le compilateur [MIN 79, p. 44]."

II-4.- L'unité MODULE

"Un module doit comporter
- un nom qui l'identifie
- le nom de la machine qu'il réalise
- éventuellement la description d'une ressource
- une liste de réalisations d'opérations qui sont conformes aux spécifications données dans la machine qu'il réalise". [MIN 79, p. 53].

II-5.- Intérêts de l'indépendance entre l'interface abstrait et la réalisation d'un type pour les bases de données

L'originalité d'ATM est de séparer la spécification d'un type construit de sa réalisation ; contrairement aux autres langages permettant la création de types nouveaux.

Cela signifie en particulier qu'un TYPE donné peut être réalisé différemment en fonction de la CAPSULE qu'on lui associe. (Resp. pour une MACHINE et un MODULE).

II-5.1.- Définition du schéma conceptuel

Lorsque l'on parle de base de données, on parle de niveau conceptuel et de niveau interne [ANS 75].

Le niveau interne a pour but de réaliser le niveau conceptuel qui décrit les objets de la base de données et les liens entre ces objets, indépendamment de leur implantation.

On peut dire que le niveau conceptuel constitue l'interface abstraite de la base de données, et que le niveau interne réalise la base de données.

Au niveau conceptuel, on parle en terme de TYPE seulement, le niveau interne est composé de CAPSULES réalisant les TYPES de niveau conceptuel.

Le modèle ANSI/SPARC de construction d'une base de données semble donc avoir la même architecture que le système de construction de programme d'ATM [CHA 79].

II-5.2.- Bases de données évolutives

Le concepteur d'une base de données est sensé tenir compte, à la définition de la base de données de toutes les évolutions possibles de celle-ci. Nous savons combien cela est utopique.

Pourtant les bases de données actuelles sont figées. Nous verrons dans le paragraphe V comment nous allons permettre une évolution du schéma de la base de données, ou plutôt comment nous intégrons une nouvelle base de données à la base de données répartie.

La distinction faite par A.T.M. entre la spécification d'un objet et sa réalisation, c'est-à-dire entre un type et une capsule nous permet d'assurer une évolution du niveau interne de la base de données comme le montre l'exemple suivant :

Exemple : Si le TYPE PERSONNE est composé des TYPES NOM et PRENOM, la modification de la CAPSULE associée à NOM n'oblige pas une modi-

fication de la CAPSULE associée à PERSONNE [dans la CAPSULE PERSONNE on importe le TYPE NOM sans faire l'hypothèse sur la façon dont ce TYPE est réalisé].

De même, un programme d'application ne sera pas modifié par la modification d'une capsule associée à un type utilisé dans ce programme.

II-6.- Bibliographie A.T.M.

[CHA 79] - [CHA 80-a] - [CHA 80-b] - [CHA 81]
[HEN 80] - [MIN 79] - [PRO 79] - [TOU 79]

III - DESCRIPTION D'UNE RELATION A L'AIDE D'ATM - Exemple

Nous considérons une relation comme un ensemble d'objets d'un certain type. Ce type est un type construit qui décrit le schéma de la relation. Chaque objet de ce type réalisera un tuple de la relation. Une relation est par conséquent vue comme un ensemble de tuples, ce type ensemble étant paramétré par le type des tuples.

La spécification du type ensemble de tuples est la suivante :

```
TYPE ENS (param type tuple
              muni de Function ACLE returns CLE ;
              # la fonction ACLE rend l'identifiant associée à un tuple # )

BUILD INIT ; # initialise la relation
MODIF INSERT (Tuple) # permet l'insertion d'un tuple.
MODIF DELETE (Clé) # permet la suppression d'un tuple.
CONSULT ACCES (Clé) → TUPLE # permet l'accès à # un tuple
        étant donnée sa clé.
```

```
FUNCTION EXIST (Clé)   boolean
        # teste l'existence d'un tuple.
ITER SELECT YIELDS Tuple
        # SELECT est un itérateur [TOU 79] qui rend un à
          un les Tuples de la relation.
```

Remarque : Cette liste des opérations associé au type relation n'est pas exhaustive.

Exemple : Soit la base

```
FOURNISSEUR (F #, NOM, VILLE)
PIECE (P #, NOM, COULEUR)
FOURNI (F #, P #, QTY)
```

Le schéma de FOURNISSEUR sera décrit grâce au type FOURN :

```
TYPE FOURN
  BUILD CONST (INTEGER, TEXT, TEXT)
       # Ces paramètres correspondent aux domaines des
         différents attributs de la relation.
  FUNCTION F # returns integer ;
  FUNCTION NOM returns text
  FUNCTION VILLE returns text
       # Ces fonctions sont les fonctions d'accès aux
         attributs de la relation.
```

De même pour PIECE :

```
TYPE PIEC
  BUILD CONST (Integer, text, text) ;
  FUNCTION P # returns integer
  FUNCTION NOM returns text
  FUNCTION COULEUR returns text
```


Et pour FOURNI :

```
TYPE FOUR
  BUILD CONST (integer, integer, text)
  FONCTION F # returns integer
  FONCTION P # returns integer
  FONCTION COULEUR returns text
  FONCTION Identifiant returns text
       # Cette fonction rend le couple F #, P #
```

Les relations FOURNISSEURS, PIECES et FOURNIS seront typées comme suit en utilisant le type paramétré ENS.

```
TYPE FOURNISSEURS = ENS with (type FOURN,
     function F # returns integer)
TYPE PIECES = ENS with (type PIEC, function P # returns
     integer)
TYPE FOURNIS = ENS with (type FOUR, function IDENTIFIANT
     returns text).
```

IV - UNE BASE DE DONNEES EST UNE MACHINE

Revenons sur la notion de module. Un module est un ensemble d'opérations. Ces opérations sont définies dans un certain contexte, ce qui signifie qu'elles utilisent des types et utilisent d'autres opérations connues par ce module, cela dans le but de gérer une certaine ressource. Cette ressource est constituée d'objets d'un certain type, dans le cadre des bases de données, ces objets seront du type relation.

On peut alors faire le parallèle, pour un ensemble de relations, entre le fait d'appartenir à une même base de données et celui d'appartenir à la même ressource d'un même module.

Les opérations de ce module sont celles qui décrivent les

règles d'utilisation de cette ressource, c'est-à-dire de cette base de données. Il peut s'agir de règles d'accès ou d'intégrité.

La machine qui interfacera ce module spécifiera cette base de données.

Une machine (et le module qui le réalise) est qualifiée par sa durée de vie : la partie ressource d'un module permet de créer effectivement des objets et éventuellement de les détruire.

Une machine est dite temporaire ou permanente. Si elle est temporaire, la ressource correspondante est créée lors de la première référence à une opération de la machine et détruite à la fin de l'exécution de la machine principale ayant conduit à cet appel.

Si elle est permanente, elle est créée et détruite explicitement.

Une base de données est constituée de machines permanentes (conservant les données) et de machines temporaires travaillant sur la partie permanente (programmes d'interrogation et de manipulation). [CHA 81].

Exemple :

```
MACHINE ENTREPRISE
   # cette machine a pour ressource les relations
     FOURNISSEURS, PIECES et FOURNIS #
   PROC M-INSERT (FOURNIS)
   # cette procédure permet l'insertion d'un n-uplet de la
     relation FOURNIS et vérifie que le fournisseur nommé
     n'existe pas dans la relation FOURNISSEURS #
MODULE ENTREPRISE FOR ENTREPRISE
   RESSOURCE :
     F : FOURNISSEURS
     P : PIECES
     FO : FOURNIS
   END
```



```
PROC M-INSERT (FOURNIS)
     ⋮
END
```

V - BASE-ENSEMBLE DE BASE

Nous savons maintenant comment créer un lien entre différentes relations pour les associer au sein d'une même base de données.

Nous aimerions maintenant pouvoir permettre un accès simultané à plusieurs bases de données, c'est-à-dire définir une machine BD capable d'accéder à la fois à des relations de la machine BD1 et de la machine BD2.

Les relations de la machine BD1, c'est-à-dire la ressource de cette machine, sont accessibles par les opérations de cette machine. Respectivement pour BD2.

Le compilateur A.T.M. nous donne alors l'opportunité suivante :

"Un contexte de compilation est un ensemble de types, de capsules, de machines et de modules".

"Une opération d'une machine est utilisable par une autre machine si les deux machines appartiennent au même contexte de compilation".

Il suffit alors de définir la machine BD dans le même contexte de compilation que les machines BD1 et BD2.

```
MACHINE BD
   # cette machine permet l'accès aux ressources de BD1 et
     BD2 # .
   PROC UTILISER
     # cette opération utilise les opérations UTILISER1 de BD1
```

et UTILISER2 de BD2 #

END

Il est important de noter que les opérations de BD peuvent gérer des règles d'utilisation mettant en jeu des objets de BD1 et de BD2, c'est-à-dire des règles inter-bases.

Une étude plus approfondie a été réalisée dans [CHA 81]

Evolution de la base de données

Intégrer une nouvelle base de données, c'est d'abord définir une nouvelle machine permettant la description de la base de données, c'est ensuite définir de nouvelles machines associant cette base de données à d'autres.

VI - LE PROBLEME DES RELATIONS TEMPORAIRES

Rappel : Tout opérateur relationnel appliqué à une ou plusieurs relations a pour résultat une autre relation.
Nous avons distingué ces relations des relations de base (celles décrites par le schéma conceptuel de la base) : ce sont les relations temporaires.

Il faut maintenant remarquer que le schéma d'une telle relation peut être définie par n'importe quelle combinaison des attributs contenus dans les relations de base.

Ou encore que ces relations peuvent être obtenues par n'importe quelle projection de la relation universelle (celle qui contient tous les attributs de la base - DEL 80).

La première conclusion est de constater qu'il n'est pas possible, en pratique, de prévoir le schéma de toutes les relations temporaires, c'est-à-dire de déclarer un type construit pour tous les schémas de ces relations, étant donné leur nombre potentiellement très important.

Nous aimerions par conséquent ne déclarer que les types effectivement nécessaires à l'exécution d'une requête. Nous avons alors deux solutions :

- L'utilisateur qui accède à la base de données déclare les types dont il a besoin avant de lancer l'exécution d'une requête. Cela signifie que cet utilisateur est capable de définir de tels types. Que dire des performances d'accès à la base de données ?

Cette solution est envisageable et sera réalisée pour l'écriture de programmes servants une application de l'organisation que la base de données décrit. Ces programmes seront alors précompilés. Cependant, si tout "bon" SGBD permet la définition de programmes précompilés, chacun sait que ces programmes ne suffisent pas à l'exploitation d'une base de données, cela parce qu'il est difficile, voir impossible de prévoir toutes les possibilités d'accès à une base de données. Il faut donc prévoir une possibilité d'accès interactif à la base de données. Dans ce cas, la déclaration des types des relations temporaires par l'utilisateur n'est pas crédible.

- Une autre solution consisterait à donner au système la possibilité, à l'analyse d'une requête, de déclarer les types dont cette requête a besoin pour s'exécuter. Aucun système actuel permet une telle déclaration dynamique de types, et si cela était, il semble que les performances d'un tel système seraient prohibitives dans le cadre des bases de données. C'est pourquoi, dans une première étude, les relations temporaires seront, dans le cas où l'accès à la base est conversationnel, stockées à l'aide des relations

de base. Une relation temporaire, composition de relations de base, sera alors décrite par un ensemble de pointeurs associant des n-uples de ces relations de base.

Il est évident que cette approche nécessitera une gestion judicieuse de la mémoire, qui sera très pragmatique et qui, par conséquent, pourra être difficilement abstraite. Cette façon de gérer les relations temporaires est détaillée dans [NAS 81].

VII - LANGAGE DE DESCRIPTION ET REALISATION DES OPERATEURS RELATIONNELS

L'étude des paragraphes précédents nous amène à planifier "SGMB" comme suit :

(a) La description des données : le langage de description des données sera A.T.M. (Il s'agit de la prise en compte informatique du schéma conceptuel).

(b) La manipulation des données : le langage de manipulation des données est l'algèbre relationnelle.
Comment réaliser les opérateurs de cette algèbre ?
On distingue le cas où toutes les relations temporaires nécessaires à l'exécution d'une requête sont définies par un type (accès par programmes précompilés), du cas où ces relations temporaires ne sont a priori pas typées (accès direct à l'aide du langage de manipulation).
Dans le premier cas, les opérateurs relationnels sont interprétés dans un langage relationnel écrit en A.T.M., dans l'autre, ils seront directement exécutés par un programme A.T.M.

CONCLUSION

"SGMB" est un système qui gère de multi-bases. Une multi-base est une base de données répartie au sens base-ensemble de bases.

Le schéma conceptuel de la BDR est la juxtaposition des schémas des bases qui composent la multi-base et est par conséquent simple à définir. Cela est loin d'être le cas lorsque la BDR est conçue de façon classique, par la définition d'un schéma global.

Chaque BD non répartie est gérée par son propre SGBD. De façon à intégrer le plus grand nombre de ces SGBD, le langage de manipulation de "SGMB" est l'algèbre relationnelle. Nous avons montré dans le chapitre III pourquoi cette algèbre nous paraissait être le langage pivot des langages relationnels.

Dire qu'une BDR est du type base-ensemble de bases, c'est dire qu'elle est logiquement répartie. L'utilisateur perçoit alors cette répartition et est capable d'apporter des informations sur la localisation des données. L'algorithme de décomposition de requête s'en trouve très simplifié et réalisable.

Réaliser un SGBD, réparti ou non, c'est assurer des performances acceptables. C'est pourquoi le chapitre V sur l'optimisation a une importance particulière.

Les principes énoncés pour l'exécution d'une requête sont assez classiques, si ce n'est la définition d'un protocole d'échange sous forme relationnelle. Celui-ci est en effet un moyen de forcer la barrière qu'est généralement l'interface d'accès à un SGBD.

L'utilisation des abstractions pour réaliser la partie "SGBD relationnel" de "SGMB" est un autre centre d'intérêt. Nous savons dès maintenant abstraire les fonctions de description et de stockage d'un SGBD. L'abstraction de la fonction de manipulation est la suite la plus immédiate à donner à ce travail.

En conclusion, ce travail est original, d'abord parce que "SGMB" est l'un des tous premiers systèmes gérant une base-ensemble de base, ensuite parce que ce système veut utiliser au maximum les abstractions, le rapprochement entre les BD et les abstractions étant un sujet de recherche des plus actuels.

BIBLIOGRAPHIE

[ANS 75] ANSISPARC interim report, American National standards, Institute Document n° 7514TSO1, (ANSI, Washington, DC), Feb. 75.

[AST 75] ASTRAHAN and CHAMBERLIN
Implantation of a Structured Query Langage
A.C.M. Octobre 1975 - Volume 18 - Number 10.

[CAL 76] CALECA et FORESTIER
Interrogation simultanée de plusieurs bases de données
Rapport de DEA - Université de GRENOBLE.

[CHA 76] CHAMBERLIN
Relational Data-Base Management Systems
Computing Surveys Vol. 8, N° 1, March 1976.

[CHA 79] CHABRIER J.J.
Projet TYP
Ecole d'été de l'AFCET.

[CHA 80:1] CHABRIER J.
Exemple de mise en oeuvre d'une application "base de données" avec le système TYP"
AFCET Journées GROPLAN 1980.

[CHA 80:2] CHABRIER J., PROCH K., MINOT R.
Manuel d'utilisation du langage A.T.M.
Publication CRIN N° 79-r-081.

[CHA 81] CHABRIER J.J.
TYP : Un système de construction modulaire d'applications orientées "Bases de données" utilisant les types abstraits de données.
Thèse d'état NANCY I - A paraître.

[CEL 80] CELLARY W., MEYER D.
A multi query approach to distributed processing.
Distributed Data bases
North Holland Publishing company.

[COD 70] CODD E.F.
A relational Model of Data for Large Shared Data Banks.
A.C.M. Volume 13 - Number 6 - June 70.

[COD 71:1] CODD E.F.
Relational completures of data base sublangages
Courant Computer Science Symposia 6.

[COD 71:2] CODD E.F.
A date base sublangage founded on the relational Calculus.
A.C.M. SIGFIMET Workshop 1971.

[DAT 78] DATE C.J.
An Introduction To Database System.
Addison - Wesley Publishing Company.

[DEC 80] DECITRE P. and ANDRE E.
The DEM distributed execution monitor
Distributed Data base - North Holland Publishing.

[DEL 80] DELOBEL 80
An overview of the relational Data Theory
Ecole INRIA Novembre 80.
Etudes formelles dans les bases de données.

[DER 74] DERNIAME J.C.
Le projet CIVA.
Un système de programmation modulaire.
Thèse d'état NANCY 1974.

[GAR 80] GARDARIN G. et BEGUES
Un protocole de manipulation de données relationnelles.
Congrès AFCET NANCY - Nov. 80.

[GOD 79] GODART C.
Le modèle de répartition "Etoile"
Rapport de Contrat SIRIUS n° 79100.

[GOD-80] GODART C.
THE ETOILE PROJECT
Distributed Databases
North Holland Publishing Company.

[HEN 80] HENRY P.
La connexion dans le projet TYP
Thèse de Spécialité Informatique - Université de NANCY I.

[LEB 80] LEBIHAN J.
SIRIUS a french nationalwide project on Distributed Databases.
6ème VLDB - Montréal - Sep. 80.

[LIT 78] LITWIN W.
Construction d'un SGBDR
Approche SIRIUS, Modèle de référence.
IRIA/SIRIUS 78150 LE CHESNAY (France).

[LIT 80] LITWIN W.
Présentation du projet B A BA
Doc. INRIA GAL-I-042.

[LIT 81] LITWIN W.
A logical Model for the design of a distributed data base
Présenté au II^{ème} séminaire sur les Systèmes Répartis à Partages de Données.
Amsterdam (Juin 81).

[MIN 79] MINOT R.
A.T.M. : Un système de fabrication de programmes basé sur les concepts de modularité et de type abstrait.
Thèse de Spécialité Informatique - Université de NANCY I.

[NAS 81] NASSIF R.
Une méthode d'implantation des opérateurs relationnels.
Rapport de DEA - Université de NANCY I.

[NGU 78] NGUYEN T.
Optimisation de requêtes relationnelles sur les bases de données réparties.
Doc. SIRIUS SCH-I-047.

[PAR 72] PARNAS D.L.
A technique for sofware module specification with examples.
A.C.M. 15, 5 mai 1972.

[POL 80] POLYPHEME :
An experience in distributed databases system design and complementation.
Distributed Databases - North Holland Publishing.

[PRO 79] PROCH K.
Paramétrisation des unités de compilation en A.T.M.
Publication CRIN N° 79-r-058.

[SIR 80] SIRIUS-DELTA
Un prototype de système de gestion de bases de données réparties.
Distributed Databases
North Holland Publishing Company.

[SMI 79] SMITH and CHANG
Optimizing the performances of a Relational Agabra Database Interface.
A.C.M. Vol. 18 - Number 10 - Octobre 75.

[THO 80] THOMAS R.
Process Structure alternative towards a distributed INGRES
Distributed Databases.
North Holland Publishing Company.

[TOU 79] TOUSI N.
Réalisation d'un mécanisme d'itération en A.T.M.
Rapport de DEA - Université de NANCY I.

[WON 76] WONG and YOUSSEFI
Décomposition, a strategy for query processing
ACM-TODS Vol. 1 - N° 3 - Sept 76.

[WON 77] WONG
Retrieving dispersed data from SDD1 : a system for distributed databases.
Computer Comp. of America
CCA 77-03 Mars 77.

NOM DE L'ETUDIANT : GODART Claude

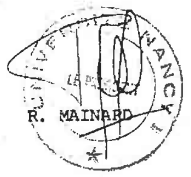
NATURE DE LA THESE : 3° cycle : Informatique

VU, APPROUVE

ET PERMIS D'IMPRIMER

NANCY, le 17 Juin 1981

LE PRESIDENT DE L'UNIVERSITE DE NANCY I,



2269. 22 JUIN 1981