Institut National Polytechnique de Lorraine

Centre de Recherche en Informatique de Nancy

PROJET ADONIS
CONCEPTION ET RÉALISATION D'UN SYSTÈME DE GESTION DE MULTIBASES DE DONNÉES

THESE

soutenue publiquement le **30 septembre 1983**

À L'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

pour l'obtention du grade de
DOCTEUR INGÉNIEUR EN INFORMATIQUE
par
Rodolphe J. NASSIF
devant la Commission d'Examen

Jean-Claude DERNIAME
Jean-Jacques CHABRIER
Antoine COCCO
Daniel COULON
Marion CREHANGE
André FLORY

Institut National Polytechnique de Lorraine

Centre de Recherche en Informatique de Nancy

PROJET ADONIS
CONCEPTION ET RÉALISATION D'UN SYSTÈME DE GESTION DE MULTIBASES DE DONNÉES

THESE

soutenue publiquement le **30 septembre 1983**

À L'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

pour l'obtention du grade de
DOCTEUR INGÉNIEUR EN INFORMATIQUE
par
Rodolphe J. NASSIF
devant la Commission d'Examen

Président: ..... *Jean-Claude DERNIAME*
Examinateurs: ..... *Jean-Jacques CHABRIER*
Antoine COCCO
Daniel COULON
Marion CREHANGE
André FLORY

A ma famille

Je tiens à remercier,

Monsieur Jean-Claude DERNIAME, Directeur du Centre de Recherche en Informatique de NANCY et Professeur à l'Université de NANCY I d'avoir supervisé mon travail, des conseils qu'il m'a prodigués, de la confiance qu'il m'a accordée et de l'honneur qu'il me fait en présidant ce Jury.

Monsieur Jean-Jacques CHABRIER, Professeur à l'Université de DIJON, Ancien Maître Assistant à l'Université de NANCY II, d'avoir dirigé mes recherches et de ses critiques sur une première version de ce travail qui m'ont permis d'améliorer sensiblement la qualité de mon manuscrit.

Monsieur Antoine COCCO, Ingénieur à SYSECA, de s'être intéressé à mon travail en acceptant de participer à ce Jury.

Monsieur Daniel COULON, Directeur du Laboratoire d'Informatique à l'Institut National Polytechnique de la Lorraine et Professeur à l'INPL, de s'être intéressé à mon travail en acceptant de participer à ce Jury.

Madame Marion CREHANGE, Professeur à l'Université de NANCY II, de l'intérêt qu'elle a porté à mon travail et de ses remarques sur une première version de ce travail qui m'ont été très utiles.

.../...

Monsieur André FLORY, Directeur du Laboratoire d'Informatique et Professeur à l'Université Jean Moulin (LYON III), de s'être intéressé à mon travail et des remarques qu'il m'a formulées sur le premier manuscrit et qui m'ont permises d'en améliorer la qualité.

Je tiens également à remercier,

Monsieur Pierre LESCANNE , Attaché de Recherche au C.N.R.S., d'avoir guidé mes premiers pas en CLU,

Mes collègues de Travail Ali AIT HADOU, Claude GODART, Thierry GRISON, Muyung Joon KIM et Najib TOUNSI pour les discussions enrichissantes qu'on a eues durant les réunions de travail; KIM et GRISON ont testé le prototype réalisé ce qui a permis de l'améliorer. D'autre part, ils m'ont aidé avec Monsieur MOHAMED BENNANI à la réalisation matérielle des exemplaires.

Tous les membres du CRIN et l'équipe de l'ANL pour l'ambiance dans le travail et en dehors des heures de travail.

Les secrétaires pour leur sympathie et les différents services qu'elles m'ont rendus.

La DGRST pour son soutien financier.

SOMMAIRE

PAGES

PAGES

NOTATIONS

BD	Base de Données
BDR	Base de Données Réparties
CI	Contrainte d'Intégrité
CS	Contrainte de Sécurité
MB	Multi-Base
RB	Relation de Base
RV	Relation Virtuelle
SC	Schéma Conceptuel
SE	Schéma Externe
SI	Schéma Interne
SG	Schéma Global
SL	Schéma Local
SGBD	Système de gestion de Bases de Données
SGBDR	Système de gestion de Bases de Données Réparties
SGMB	Système de gestion de Multibase de Données

INTRODUCTION

INTRODUCTION

L'architecture actuelle des SGBD Réparties n'a véritablement été conçue que pour manipuler de façon globale des BD réparties sur différents sites.

En effet, bien que ces systémes permettent de stocker les données sur les différents noeuds d'un réseau, ils n'offrent pratiquement pas de possibilité pour manipuler simultanément des données de plusieurs bases. Autrement dit, cette approche requiert que la conception logique des bases s'effectue de façon globale ce qui pose beaucoup de problèmes. En effet, l'intégration compléte de différentes applications dans un schéma conceptuel global exige d'importants compromis.

Par ailleurs, cette approche s'applique mal dans le cas de conception progressive, qui est le plus courant.

L'approche multibase proposée par LIT (79) (8I) présente l'avantage de séparer totalement :

- la répartition logique des données. Le schéma d'une multibase est défini alors comme l'union des schémas de plusieurs bases ou multibases.

- la répartition physique des données sur différents sites.

L'intérêt de l'approche multibase par rapport aux approches classiques se situe selon nous au niveau de la répartition logique des données. Notre contribution dans cette thèse vise à intégrer de façon cohérente les notions de contraintes d'intégrité, de contraintes de sécurité et de vues dans un SGMBD.

Dans les chapitres qui suivent, nous utiliserons souvent un exemple tiré de LIT (79) et que nous décrivons ci-dessous :

```
MB   Loisir
  MB  Restaurant
    BD  R - Luxe
          R (numr, nomr, tel, arrond, st-métro)%restaurant Luxe
          Plats (nump, nomp, type)              %plats
          Menus (numr, nump, prix)              %menus
    END BD

    BD  R - Mode
          R (numr, nomr, tel, arrond, st-métro)%restaurant Modeste
          P (nump, nomp, type)                  %plat
          Menus (numr, nump, prix)              %menu
    END BD
  END MB

   BD   Cinéma
          C (numc, nomc, tel, arrond, st-métro, nbr-salle)% cinéma
          F (numf, nomf, pays, genre, version)          %film
          S (numc, numf)                                % projec
                                                          tion
    END BD

   BD   Métro
          L (numl, noml)                        % ligne
          S (numst, nomst, arrond)              % station
          L-S (numl, numst)                     % ligne-station

    END BD

END MB
```

Cette multibase "Loisir" offre la possibilité de travailler indépendamment sur chacune des bases composantes (Restaurant, Cinéma et Métro) ou sur plusieurs de ces bases simultanément.

Avec l'approche classique, il faudrait que l'utilisateur définisse d'abord un schéma global de "Loisir". Dans ce schéma, certaines entités peuvent avoir des noms ou des structures différentes de celles des entités correspondantes des BD Restaurant, Cinéma et Métro. Par exemple, le concepteur doit d'abord commencer par distinguer la relation R de R - Mode de la relation C de cinéma, ainsi que de la relation de R - Luxe, en donnant des noms différents à ces relations et ensuite définir des schémas externes qui permettent alors d'adapter le schéma global à l'expression initiale des besoins des utilisateurs qui désirent travailler sur l'une des trois bases seulement.

Un SGMB doit maintenir la cohérence des multibases qu'il gère ainsi que les différents niveaux de confidentialité. Il faut donc que ce système permette de prendre en compte des contraintes de sécurité et d'intégrité sur les multibases. Une contrainte pouvant faire intervenir des relations appartenant à des bases différentes.

Dans le prototype que nous avons réalisé, nous avons retenu le modèle relationnel car il permet de manipuler les données indépendamment des structures de stockage et des stratégies d'accès à ces données. En outre, le modèle relationnel offre des langages de haut niveau d'accès aux données

Les SGMB sont des logiciels d'assez grande taille. Il nous est apparu tout à fait essentiel que, pour implanter cette sorte de système, nous puissions utiliser les techniques de développement relevant du génie logiciel. Les concepts qui nous semblent les plus importants du génie logiciel sont la modularité et l'abstraction. Ainsi pour développer notre prototype de SGMB, nous avons choisi un langage de programmation qui supporte ces concepts. Assurément, la maintenance, l'évolutivité et la transportabilité de tels systèmes se feront alors à moindre coût. Plusieurs langages répondent plus ou moins à ces critères ALPHARD, CIVA, CLU, SIMULA, TYP. Après avoir expérimenté une version écirte en TYP, nous nous sommes décidé à réécrire un autre prototype en langage CLU.

Un prototype (MUQUAPOL) de SGMB a été développé à l'INRIA KAB (82). Le langage de manipulation de MAQUAPOL est une extension du langage relationnel QUEL d'INGRES. Le système construit sur le SGBDR POLYPHEME (ADI (78)) ne permet pas d'exprimer des contraintes d'intégrité ou de confidentialité sur les multibases.

Dans notre équipe, une étude faite par GOD (81) a permis de dégager les fonctionnalités d'un SGMB. Ensuite, TOU(83) a développé dans le cadre d'un contrat avec l'INRIA une première maquette d'un SGMB Relationnel appelé TYP-R.

"TYP-R est un système de gestion de bases de données relationnelles et réparties écrit en TYP : un système intégré de programmation modulaire avec utilisation de types abstraits de données". TOU (83).

En paraphrasant TOUNSI, on peut dire que TYP-R a permis de démontrer l'utilité de l'application de nouvelles techniques de génie logiciel pour développer des systèmes de gesti de multibase, l'utilisation des types abstraits permettant de retarder les choix d'implantation.

TYP-R nous a permis aussi de dégager les différents problèmes qui se posent lors de la conception et de l'implantation de SGMB. Certaines caractéristiques de l'approche TYP-R nous ont poussé à implanter un nouveau système en CLU, plutôt que de faire une extension de TYP-R, pour tenir compte des contraintes d'intégrité et de sécurité au niveau de la multibase. Citons trois de ces caractéristiques*:

---

* pour les mots techniques, se référer ch. II

1 - A chaque relation de base, le système TYP-R associe une "machine" (et le "module" correspondant) pour stocker les tuples de cette relation et un "type" (et la "capsule" correspondante) pour accéder ou modifier les différents attributs des tuples. Or, comme pour accéder aux tuples d'une relation de base, il faut appliquer une opération (un itérateur) sur la "machine" correspondante (ex : ω Menus extraire) et comme le nom d'une machine ne peut être passé en paramètre après le "ω", on a choisi d'écrire des clauses de la forme :
si nom-machine = "Menus" alors ω Menus extraire
sinon, si nom-machine = "L" alors ω L extraire
sinon si...

Cette solution n'est pas adaptée à l'introduction dynamique de nouvelles relations en effet, entre autres problèmes, il faut générer les unités "type" "machine" "capsule" et "module" GRI (83) et modifier le module contenant les clauses citées plus haut.

2 - Le langage de définition des données n'est pas intégré au système TYP-R.

3 - L'utilisation des itérateurs dans les types et machines génériques du système TYP nous a posé des problèmes cf. ch. II § VII.

Le premier chapitre de cette thése rappelle des notions essentielles sur les BD, les SGBD, le modèle relationnel ainsi que l'approche classique de BD Réparties.

La dernière partie de ce chapitre est consacrée à l'approche multibase : définition, intérêt et un aperçu de la solution utilisée dans notre implantation.

Dans le deuxième chapitre, les notions de modularité et d'abstraction sont rappelées avant la présentation de deux approches, la première consistant à spécifier une base de données par des types abstraits algébriques, la seconde utilisant un langage de programmation orienté base de données pour définir et manipuler des bases de données.

Notons au passage que notre approche consiste à définir un langage de définition et de manipulation de données indépendant du langage de programmation (langage hôte).

Ensuite, nous présentons rapidement une comparaison entre TYP et CLU systèmes qui ont été successivement utilisés pour la réalisation des systèmes expérimentaux.

Les problèmes de relations virtuelles et des valeurs indéterminées ainsi que les solutions adoptées à ces problèmes sont traités dans le Chapitre III.

Le quatrième chapitre est consacré à la confidentialité et à l'intégrité des données.

Enfin, la maquette est présentée au Chapitre V.

CHAPITRE I

A - BASES DE DONNEES

B - BASES DE DONNEES REPARTIES (BDR)

C - MULTIBASE

0 - INTRODUCTION

Une BD est un ensemble cohérent de données enregistré sur un support informatique et pouvant être partagé par différents utilisateurs pour satisfaire des besoins hétérogénes actuels et potentiels.

Un utilisateur interagit avec la BD par l'intermédiaire d'un logiciel appelé "système de gestion de bases de données (SGBD), qui lui fournit un langage de définition et un langage de manipulation de données et divers moyens de sécurité.

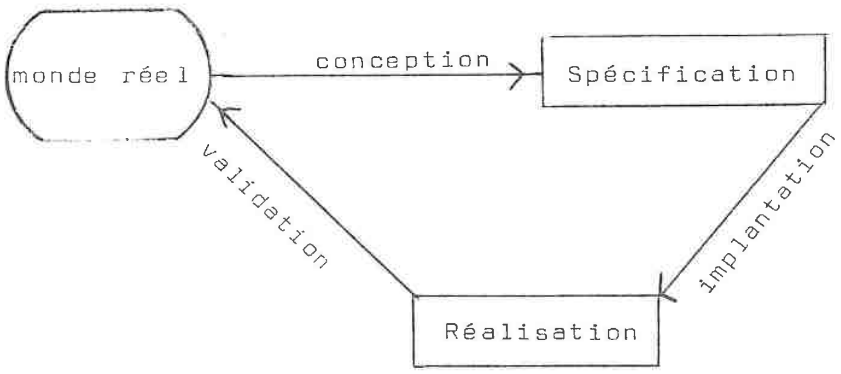
La notion de BD répond au moins à un double besoin :

- partager des informations relevant d'un domaine d'intérêt commun à plusieurs utilisateurs ;

- offrir la possibilité d'enrichir une BD en intégrant de nouvelles données sans être obligé de dupliquer ou bien encore de réorganiser les données et/ou les programmes existants.

I - CONCEPTION D'UNE BD

Dans ce paragraphe, il n'est pas dans notre intention de développer de façon détaillée le processus de conception d'une BD car ceci nous ferait sortir de notre domaine d'intérêt, toutefois, il importe de rappeler que le développement d'une application BD se déroule généralement selon le schéma décrit ci-dessous CHA (82) :



Ce schéma très général peut être rapproché de méthodes de conception de BD développés par FLO (82), WIL (81) REM (79) qui tentent de résoudre un certain nombre de problèmes cruciaux :

I - Définition d'un modèle permettant de représenter les concepts du monde réel notamment la dynamique du système d'information.

2 - Intégration des différentes sortes de besoins des utilisateurs.

3 - Evolution des besoins des utilisateurs.

WIL (8I) propose un outil d'aide à la spécification des besoins en terme d'entité, évènement, relation, attribut et valeur. Cet outil génère un schéma relationnel normalisé (5ième forme normale).

FLO (82) fournit une méthode basée sur un certain nombre de critères permettant au concepteur d'une part d'identifier facilement les entités-associations et d'autre part de définir de façon semi-programmable le schéma de la base.

Le résultat de la conception peut être décrit selon une architecture en trois niveaux définie dans le rapport ANSI-SPARC (75) : niveau conceptuel,niveau externe et niveau interne.

I - Le niveau conceptuel sert à définir le contenu de la BD, c'est-à-dire les classes d'objets qui constituent le réel à représenter.

2 - Le niveau externe permet de restreindre la visibilité sur la BD aux seules données du domaine d'intérêt.

Ainsi, les données qui n'intéressent pas un utilisateur ou celles auxquelles il n'a pas le droit d'accéder ne seront

pas définies dans le schéma externe de cet utilisateur.

Les descriptions des données au niveau externe et au niveau conceptuel peuvent être sensiblement différentes.

3 - Le niveau interne sert à décrire les structures de données, les indexes d'accès, les types d'enregistrements stockés...

L'implantation de la BD s'effectue à partir des informations décrites à ce niveau.

II - MODELES DE DONNEES

Une BD structure les informations tirées du monde réel suivant un certain modèle de données.

Le concept de modèle de données qu'utilise les spécialistes des BD, notamment COD (8I) n'a rien à voir avec le concept classique de modèle des logiciens. Un modèle de données est défini par :

I - Un langage de définitions de schémas de données et de contraintes d'intégrité spécifiques,

Des régles d'intégrité générales sont inhérentes au modèle.

2 - Un langage de manipulation de données.

REMARQUES :

a) Dans le domaine des BD, on a coutume de confondre le schéma d'une base, qui représente sa sémantique, avec son interprétation qui est l'ensemble des données à un instant précis.

b) Les contraintes d'intégrité et de sécurité sont décrites avec le langage d'interrogation.

Rappelons brièvement les principaux modèles de données existants actuellement : hiérarchique, réseau, relationnel et irréductible.

a) Le modèle hiérarchique (AMC(76)) : dans ce modèle, les données sont représentées par des enregistrements logiques reliés entre eux sous forme d'arbre.

Le sommet de l'arbre est appelé la racine.

Un noeud peut avoir plusieurs fils.

Un fils ne peut donc pas exister indépendamment de son père. Par exemple, la BD "Cinéma" pourrait être représentée par un arbre de racine "C" dont le seul fils serait "F" pour une même occurrencede "C".

Si l'on veut connaître le numéro de la salle où la projection d'un film se fait il est alors nécessaire d'ajouter un champ supplémentaire dans le schéma à la définiton de "F".

Le modèle hiérarchique privilégie un type d'accès (du père au fils), les requêtes peuvent devenir très compliquées car il faut indiquer le chemin d'accès logique aux données.

Ce modèle introduit des anomalies de mise à jour.

Par exemple, si l'on modifie la "version" d'un film il faut parcourir tous les cinémas sinon il y aura inconsistance entre des enregistrements sensés représenter le même film dans différents cinémas.

En outre, on ne peut pas retenir des informations sur des films qui ne sont pas projetés.

b) Le modèle réseau (AMC(76)) : dans ce modèle, les données sont représentées par des enregistrements logiques et par des liens entre ces enregistrements.

Par exemple, la BD "Cinéma" sera représentée par trois types d'enregistrement ("C", "F" et un connecteur contenant des données relatives à l'association entre "C" et "F" (le numéro de la salle de projection)).

Les modèles hiérarchiques et réseaux ne permettent pas de réaliser une indépendance logique satisfaisante entre les données et les programmes car des liens entre les données existent et sont référencés dans les programmes d'application.

c) Le modèle relationnel.

Dans ce modèle, les données sont représentées par des relations. Une association entre plusieurs entités du monde réel est alors considérée comme une entité et représentée par une relation.

Une relation s'interprète comme un tableau contenant les valeurs des différentes occurrences d'une même entité.

Par exemple, la BD "Cinéma" sera formée de trois relations "C", "F" et Projection.

La dernière relation contiendra le numéro de la salle de cinéma où s'effectue la projection d'un film donné.

Dans le modèle relationnel, il n'y a pas de liens logiques explicites définis entre les différentes relations.

Les correspondances se font uniquement par les valeurs.

d) Les modèles irréductibles (DAT(83)).

Dans ces modèles, les faits élémentaires sont séparés.

Par exemple, la relation F (numf, nomf, pays, genre, version) qui contient quatre faits élémentaires concernant un film :

nomf, pays, genre et version, sera décomposée en quatre relations F - nom (numf, nomf), F - pays (numf, pays), F - G (numf, genre) et F - V (numf, version).

Différents modèles ont été établis sur ce principe : le modèle binaire, le modèle relationnel irréductible et le modèle fonctionnel.

1 - Le modèle binaire : il n'accepte que des relations binaires.

Il a l'inconvénient d'introduire plusieurs entités fictives pour représenter des "faits" qui ne peuvent pas être réduits à deux caractéristiques.

Par exemple, Projection (numc, numf, num - salle) sera définie dans le modèle binaire par projx (projy (numc, numf), num-salle).

D'autres problèmes se posent dans ce modèle.

Ainsi, le résultat d'une requête n'est pas toujours binaire.

2 - Le modèle relationnel irréductible (DAT(83)): Les relations représentent des faits atomiques. Le degré des relations n'est pas limité à deux.

3 - Le modèle fonctionnel CHA (82) : Les données sont représentées par des objets et des fonctions.

Par exemple "F" serait représentée par une relation unaire caractérisée par un ensemble de fonctions :

numf : F ⟶ entier
nomf : F ⟶ char (50)

pays : F ⟶ char (25)
version : F ⟶ char (1)
genre : F ⟶ char (15)

Les requêtes sont exprimées sous forme de fonction ce qui est très intéressant car la composition de deux fonctions est une fonction.

Dans ce modèle, il n'existe qu'une seule occurence de chaque entité ce qui élimine de nombreuses anomalies de mise à jour.

Par exemple, il est impossible de donner un numéro de salle pour projeter un film qui n'existe pas dans "F".

Il n'en est pas de même pour le modèle relationnel.

Ainsi, dans la relation "Projection", on aurait pu affecter un numéro de salle à un film qui ne figurait pas dans "F".

Notons néanmoins que dans le modèle relationnel, on élimine cette anomalie en introduisant une contrainte d'intégrité de référence.

REMARQUES :

Une relation du modèle binaire ou du modèle relationnel irréductible est, par définition, normalisée car elle ne représente qu'un seul "fait élémentaire".

III - SYSTEME DE GESTION DE BASES DE DONNEES (SGBD)

Un SGBD permet aux utilisateurs d'intéragir avec la BD. Les principaux services qu'il offre sont :

- La définition et la manipulation des données :une BD a trois niveaux de description (conceptuel, externe et interne). Le SGDB doit fournir aux utilisateurs, un langage pour définir, mettre-à-jour ou accéder aux données et ceci, à chacun des trois niveaux.

- La confidentialité : le SGDB doit interdire l'accès à certaines données sensibles ; c'est à dire interdites aux utilisateurs non autorisés.

- Intégrité : le SGDB doit interdire les m-à-j non valides de la base. Les états et les transitions d'états valides sont définis par des règles d'intégrité, écrites sous forme de prédicats, qui doivent être vérifiés par les données de la base.

ex : 1 - L'âge d'un employé est compris entre 15 et 100.

2 - Le salaire d'un employé ne peut pas décroître,

3 - La valeur de l'attribut nbr-employés pour un département donné de la relation département doit être égale au nombre d'employés de la relation employé qui travaillent dans ce département.

N.B. : les problèmes d'intégrité liés à la concurrence d'accès sont développés dans la partie BD Repartie.

- Transaction : une transaction est une unité de travail. Elle doit être exécutée entièrement ou annulée. Une transaction est composée de plusieurs actions élémentaires. Elle s'exécute sur un état cohérent de la base. A la fin de la transaction l'état de la base doit être cohérent mais durant l'exécution de la transaction, les contraintes d'intégrité (qui ne sont pas immédiates) peuvent ne pas être vérifiées. Cette notion est importante car elle permet de retarder l'application de certaines contraintes. Par exemple, si une entreprise embauche un employé, on doit effectuer deux opérations :

insérer l'employé dans la relation employé et,
augmenter de 1 le nombre d'employés pour le département concerné
Quand on exécute l'une de ces deux actions, la 3ème contrainte d'intégrité définit plus haut ne sera plus vérifiée. S'il n'y avait pas la notion de transaction, on se trouverait dans l'impossibilité d'effectuer cette insertion ou bien ce type de contrainte ne pourrait pas être posé :

- Sécurité : un SGDB doit fournir des mécanismes de reprise en cas de panne matérielle ou en cas d'erreurs "logiciel" de façon à pouvoir restaurer l'état de la base. Le principe qui soustend les mécanismes de sécurité est la redondance. Toute information doit pouvoir être reconstruites à partir d'autres informations stockées ailleurs dans le système.

- Dictionnaire des données : le SGDB doit maintenir des informations sur les "types" des données de la base, sur les contraintes attachées à ces "types".

- Autres fonctions : génération de rapports statistiques sur l'utilisation des données.

IV - LE MODELE RELATIONNEL

Le modèle relationnel a été défini pour la première fois par COD (70). Depuis, il y a eu beaucoup de recherches et de travaux sur ce modèle dont les principaux atouts sont :

- La simplicité,

⊠ l'indépendance qu'il offre entre les structures logiques des données et leurs implantations ;

⊠ les langages de haut niveau qu'il supporte et qui permettent d'exprimer des opérations sur des ensembles de données.

⊠ le fondement théorique : calcul relationnel, algébre relationnelle.

X) COMPOSANTS DU MODELE RELATIONNEL

Nous allons détailler les trois composants du modèle relationnel, à savoir :

- les structures de données,
- les opérations sur ces structures
- et les règles d'intégrité générales.

I) Structures de données

On distingue dans le modèle relationnel plusieurs structures de données : attribut, domaine , tuple, relation et clé.

a) attribut : c'est un identificateur appartenant à un ensemble fini $\{A_1, \ldots, A_n\}$

b) domaine : à chaque attribut A_i est associé un domaine dom (A_i) qui est l'ensemble des valeurs possibles de l'attribut A_i.

c) Tuple : soit T le produit cartésien dom $(A_1) \times \ldots \times$ dom (A_n) un tuple $(d_1, \ldots d_n)$ où $d_1 \in$ dom $(A_1) \ldots (d_n) \in$ dom (A_n) est un élément de ce produit cartésien.

d) Relation : Soit R, l'ensemble des parties de T. Un élément r de R est une relation.

Donc une relation est un sous ensemble du produit cartésien dom (A_1)x.... x dom (A_n) c'est-à-dire qu'elle est formée d'un ensemble de tuples.

c) clé : si les valeurs d'un ensemble d'attributs X, pour chaque tuple d'une relation permettent d'identifier ce tuple, et qu'aucun sous ensemble des attributs de X ne permet de le faire, alors x est une clé candidate de la relation.

Une relation peut avoir plusieurs cles candidates. On en choisit une qui sera appelée clé primaire de la relation.

On dit qu'un ensemble d'attributs Y d'une relation R_1 est une clé secondaire si Y est une clé primaire d'une autre relation R_2.

Exemple : Soit le jeu de données suivant pour la base R - Luxe constituée des trois relations R, Plats et Menus.

R

numr	nomr	Tél.	arrond.	St-Métro
5	Cedre	3456254	14	Montparnasse
10	Atlantique	2082720	8	Madeleine
16	La Ciboulette	2717234	4	Hotel de Ville
4	Le Dome	5678345	14	Montparnasse

PLATS

nump	nomp	type
2	Paela	Espagnole
5	Pizza	Italienne
3	Canard laqué	Chinoise
9	Quiche	Lorraine
15	Couscous	Marocaine

MENUS

numr	nump	prix
5	2	30
5	5	25
5	3	30
16	9	20
5	9	15
10	5	23
16	15	25
5	15	25
4	3	30

⊠ Une relation est représentée sous la forme d'un tableau dont les colonnes contiennent les valeurs des différents attributs et les lignes les tuples de la relation.

attributs {numr, nomr, tel, arrond, st-métro} La relation R est définie sur l'ensemble des

L'attribut numr peut être défini par exemple sur le domaine D_1 = (1..10 000), nomr sur CHAR (30), tel sur CHAR (12) arrond. sur (1..21) et St Métro sur CHAR (30).

Dans la relation R, la connaissance de la valeur de numr permet d'identifier un tuple de cette relation.

Si cette propriété est vraie pour toute instantiation de R, alors numr est appelé clé-candidate.

On va supposer que numr est la clé primaire de R, nump celle de Plats, et $\{numr, nump\}$ celle de Menus.

Notons que Menus a deux clés secondaires numr et nump qui sont respectivement les clés primaires de R et Plats.

Une relation dans une BD peut être définie de deux façons : par son intention et par son extension.

⊠ L'intention d'une relation est une propriété sur les tuples indépendante du temps ;

⊠ L'extension d'une relation est l'ensemble des tuples de la relation à un instant donné.

2) Opérations du modèle relationnel

On distingue deux types d'opérations : les mises à jour et les consultations.

a) Les opérations de mise à jour : elles permettent d'insérer, de supprimer ou de modifier des tuples ou des relations

b) Les opérations de consultation : elles permettent de sélectionner certaines données de la base. Dans le cadre du modèle relationnel, ces opérations ont fait l'objet de nombreux travaux.

L'objectif visé est de fournir des opérations de

haut niveau qui travaillent sur des ensembles de données et ainsi dispensent les utilisateurs du recours aux boucles (itération).

Nous allons détailler dans cette partie les opérateurs de l'algébre relationnelle DAT (81) DEL(82) que nous avons retenu dans notre implantation.

Ensuite, nous passerons en revue d'autres types de langage.

On distingue deux groupes d'opérations dans l'algèbre relationnelle :

- les opérations ensemblistes (union, intersection différence et produit cartésien) et les opérations typiquement relationnelles (sélection, jointure, projection et division) que nous allons décrire maintenant.

Notons au passage que le résultat de l'application d'un opérateur relationnel sur une relation est une relation.

a) Projection : cette opération réalise un découpage vertical de la relation. Le découpage est obtenu par sélection de certains attributs. Les tuples redondants sont éliminés du résultat.

ex : PROJECT (R, ARROND) délivre

ARROND.
14
8
4

b) Sélection : Cette opération réalise un découpage horizontal d'une relation. Elle délivre les tuples de la relatior qui vérifient un prédicat donné.

Le prédicat est exprimé sous la forme d'une combinaison de termes booleens faisant intervenir les attributs de la relation.

ex : la requête SELECT (R, St-Métro = Montparnasse)
délivre les tuples de R qui ont Montparnasse comme station de métro la plus proche :

NUMR	NOMR	TEL.	ARROND.	ST- METRO
5	CEDRE	3456254	14	Montparnasse
4	LE DOME	5678345	14	Montparnasse

c) Jointure : Cette opération permet de regrouper des données dispersées sur deux relations.

La jointure d'une relation R_1 avec une relation R_2 sur les attributs A de R_1 et B de R_2 selon un certain opérateur op est l'ensemble des tuples t tel que t est la concatenation d'un tuple $r_1 \in R_1$ et d'un tuple $r_2 \in R_2$, le résultat de la comparaison a op b étant vrai (a étant la valeur de l'attribut A de R_1 et b celle de l'attribut B de R_2).

L'opération de jointure peut aussi être définie sur une liste d'attributs.

Si l'opérateur de comparaison est l'égalité l'opération est alors appelée "équi-join", ou"natural join" si on élimine les colonnes redondantes sur lesquelles s'est effectuée la jointure.

L'opération de jointure peut être réalisée en effectuant le produit cartésien des deux relations suivi d'une sélection

(c'est loin d'être une façon optimale pour réaliser la jointure).

Exemple : Si l'on veut connaître pour chaque restaurant le prix de chacun des plats servis, la requête s'écrira :

JOIN (R, Menus, numr = numr)

Le résultat de cette requête est :

NUM-R	NOM-R	TEL.	ARROND.	ST-METRO	NUM-P	PRIX
5	CEDRE	3456254	14	MONTPARNASSE	2	30
5	CEDRE	"	"	"	5	25
5	CEDRE	"	"	"	3	30
5	CEDRE	"	"	"	9	15
5	CEDRE	"	"	"	15	25
10	ATLANTIQUE	2082720	8	MADELEINE	5	23
16	LA CIBOULETTE	2717234	4	HOTEL DE VILLE	9	20
4	LE DOME	5678345	14	MONTPARNASSE	3	30

d) Division : Dans sa façon la plus simple, la division est une opération entre une relation binaire (le dividende) et une relation unaire (le diviseur) qui produit une relation unaire (le quotient).

Supposons que R_1 ait deux attributs X et Y et que R_2 ait un attribut Z et que Y et Z soient définies sur des domaines compatibles alors DIVISER (R_1, R_2, Y, Z) produit un quotient défini sur le même domaine que X. Une valeur x apparaît dans le quotient si et seulement si le couple(x, y) apparaît dans R_1 pour toutes les valeurs z de R_2.

Exemple : Soient les deux relations R_1 et R_2

R_1 (numr, nump) = PROJECT (Menus, numr, nump)
R_2 (nump) = PROJECT (Plats, nump)

alors DIVISER (R_1, R_2, nump, nump) délivre

numr
5

Cette requête permet d'avoir les numéros de restaurant qui servent tous les plats de la relation Plats.

Opérations de définition de contraintes de sécurité et d'intégrité : ce sont des prédicats de sécurité et d'intégrité. Cf. Ch. III.

3) Régles générales d'intégrité

Il y a deux régles générales d'intégrité dans le modèle relationnel :

- Régle d'identité : tous les composants d'une clé primaire doivent avoir des valeurs bien déterminées.

- Intégrité Référentielle : toute valeur d'une clé secondaire doit figurer comme une valeur d'une clé primaire d'une autre relation.

Exemple : les valeurs de numr de la relation Menus doivent figurer dans la relation R pour l'attribut numr.

Ainsi, on n'a pas le droit de rajouter un tuple dans la relation Menus pour un restaurant qui ne figure pas dans R.

B) LANGAGES RELATIONNELS DAT (81) DEL (82)

Les langages relationnels peuvent être utilisés soit en mode interactif, soit dans un langage de programmation

classique (langage hôte). Dans Système - R, développé par IBM le langage SEQUEL s'utilise aussi bien en mode interactif qu'en mode différé en écrivant un programme PL/1.

La requête de SEQUEL doit alors être préfixée par le symbole "$" ce qui permet à un préprocesseur de la détecter et de la transformer en des appels à des sous programmes PL/1.

Nous allons présenter rapidement trois types de langages relationnels (le langage algébrique a déjà été défini au § IV) : le calcul relationnel, les langages orientés "mapping" et les langages graphiques.

1) Calcul relationnel :

La famille des langages basés sur le calcul relationnel s'est développée du fait que le calcul des prédicats du 1er ordre peut être utilisé comme un sous langage pour les relations normalisées.

Un énoncé de problème exprimé sous forme d'un prédicat est non procédural.

En effet, on caractérise le résultat de la requête sans indiquer la façon de l'obtenir. le langage ALPHA définit par CODD (DAT (8I) est un langage du calcul relationnel.

Une requête en ALPHA a deux parties :

- une cible qui spécifie l'attribut (s) de la relation qui doit être retourné.

- une qualification qui permet de sélectionner les tuples particuliers de la relation cible.

Exemple : Quel est le nom du restaurant de numéro 5 ?

```
RANGE    Rest R
GET W    Rest    NOM-R = (Rest. NUM-R = 5)
```

2) Langages orientés "mapping"

Ce type de langage a été introduit par R.F. BOYCE.

Il offre des possibilités équivalentes à celles du calcul relationnel tout en évitant des concepts logiques comme les quantificateurs.

Le bloc de construction principal des langages orientés "mapping" est composé de trois éléments : la clause SELECT qui définit la liste des attributs résultats, la clause FROM qui détermine les relations nécessaires à la définiton des attributs précisés dans le SELECT et enfin la clause WHERE qui spécifie les conditions devant être vérifiées par les tuples des relations définies dans la clause FROM.

```
Exemple :    SELECT    NOM-R
             FROM       R
             WHERE     NUM-R = 5
```

Le résultat d'un mapping peut être utilisé dans la spécification d'un autre mapping. SEQUEL est un langage orienté mapping.

3) Langages graphiques

Dans ces langages, l'utilisateur exprime ses requêtes en faisant des choix ou en remplissant des cases vides sur un écran. QUERY-BY-EXAMPLE et CUPID appartiennent à cette classe de langages.

Par exemple, avec QBE, l'utilisateur remplit une ou plusieurs lignes de la relation avec un "exemple" du résultat désiré.

Les valeurs connues sont passées directement, les autres sont représentées par des valeurs arbitraires soulignées pour indiquer que ce sont des exemples.

Les attributs qui doivent être imprimés sont identifiés par la lettre P.

8- NORMALISATION

Le processus de normalisation fournit une aide à la conception du schéma de la BD à partir de la spécification des dépendances qui existent entre les données de la base.

Son application aide le concepteur à définir les relations (ainsi que leurs attributs) et ceci en minimisant la redondance de l'information et en prévenant les anomalies de mises à jour.

L'objectif visé est qu'une relation ne doit représenter qu'un seul concept du monde réel.

La théorie de la normalisation a été établie sur le concept de forme normale.

Une relation est dans une certaine forme normale si elle satisfait certaines contraintes d'intégrité liées à cette forme.

Les contraintes principales qui entrent en jeu sont les dépendances fonctionnelles que nous allons introduire avant d'illustrer les différentes formes normales sur des exemples.

Pour une étude plus compléte et plus formelle, se reporter à DAT (81) DEL (82).

I - DEPENDANCE FONCTIONNELLE (DF)

Soient A et B, deux ensembles d'attributs disjoints

d'une relation R, on dit que :

- B dépend fonctionnellement de A si, à chaque valeur a de A dans R, est associée une seule valeur b de B dans R. La DF est notée par " ⟶ "

Exemple : dans la relation client on a :

NUMC ⟶ NOMC
NUMC ⟶ VILLE
NUMC ⟶ taux-r

- B est en DF élémentaire de A si B n'est pas en DF avec unsous ensemble de A

- B est en DF directe de A s'il n'existe pas un ensemble d'attributs C de R tel que A ⟶ C et C ⟶ B.

D'autres types de dépendances existent :

- dépendances multivaluées : Soit R (X,Y,Z) un schéma de relation où X,Y, Z sont disjoints deux à deux.

On dit que Y dépend de X par une dépendance multivaluée si le fait que la relation R contient les tuples r (x,y,z) et r(x,y',z') implique qu'elle contient aussi le tuple r (x,y,z') et r(x,y',z)

- dépendance de jointure, dépendance hiérarchique, dépendance produit.

II - FORMES NORMALES (FN)

Nous allons décrire les 5 formes normales.

a) 1ère Forme normale : une relation est en 1ere FN.

FN si les attributs sont définis sur des domaines simples et si chacun des attributs ne faisant pas partie de la clé dépend fonctionnellement de la clé.

La relation com 1 n'est pas en 1° FN, car le domaine de NUMP-QTE n'est pas simple, mais on peut la transformer facilement pour qu'elle le devienne),

Com 1 :

NUMC	NUMP - QTE	DATE
5	1 200	11.07.82
5	3 300	11.07.82

⟹

NUMC	NUMP	QTE	DATE
5	1	200	11.07.82
5	3	300	11.07.82

b) 2ième forme normale : Une relation est en 2ième FN si elle est en 1ère FN et si les attributs qui ne font pas partie de la clé sont en DF élémentaire de la clé.

La 2ième FN est transgressée si un attribut n'appartenant pas à la clé est un fait sur des attributs de la clé.

Soit la relation qui contient les attributs suivants :

ETUDIANT	UNITE DE VALEUR	SALLE DE COURS	NOTE

clé

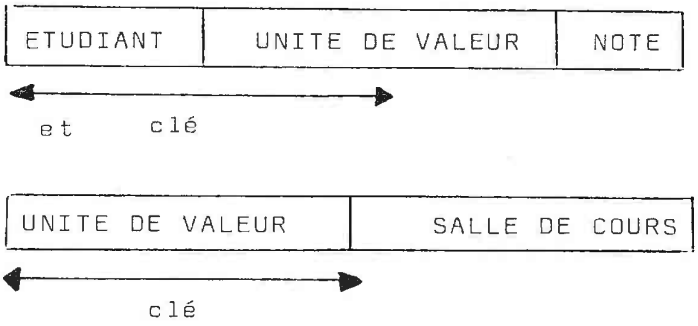
La salle de cours précise l'endroit où l'unité de valeur va être enseignée donc c'est un fait sur une partie de la clé : cette relation qui n'est pas en 2 FN présente certaines anomalies :

⊠ la salle de cours d'une unité de valeur est répétée pour chaque étudiant qui est inscrit dans cette unité de valeur.

⊠ Si la salle change, il faut modifier tous les tuples des étudiants qui suivent l'unité de valeur.

⊠ On ne peut pas stocker une information sur la salle où une unité de valeur va être enseignée s'il n'y a pas déjà un étudiant qui est inscrit à cette dernière.

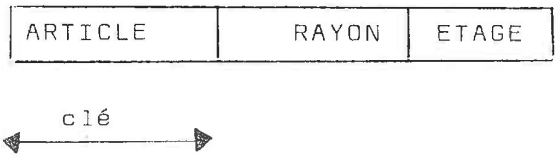
Cette relation peut être décomposée en deux relations qui sont en 2 FN :



c) 3ième forme normale : une relation est en 3ième FN si elle est en 2ième FN et s'il n'y a pas de DF entre les attributs qui ne font pas partie de la clé.

La 3ième FN est transgressée quand un attribut non clé (qui n'appartient pas à la clé) est un fait sur un autre attribut non clé.

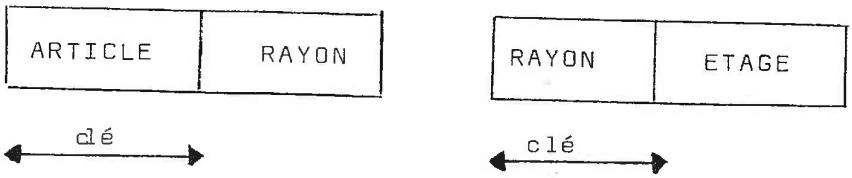
Soit la relation :



Si chaque rayon se trouve dans un seul étage, alors étage est un "fait" sur le rayon.

Cette relation qui n'est pas en 3ième FN présente des anomalies de mises à jour de même nature que celles de l'exemple précédent.

Cette relation peut être décomposée en deux relations qui sont en 3 FN.



d) 4ième Forme normale : une relation est en 4ième FN, si et seulement si l'existence d'une dépendance multivaluée $X \twoheadrightarrow Y$ dans R implique que tous les attributs de R dépendent fonctionnellement de X.

La 4ième FN est transgressée quand une relation contient deux associations (ou plus) de type m-n ou n-1 d'une entité.

Soit la relation

Clé

COURS	PROFESSEUR	LIVRE
Physique	MARTIN	Mécanique I
Physique	MARTIN	Optique - 3
Physique	DUPONT	Mécanique-I
Physique	DUPONT	Optique- 3
Math.	DURAND	Algébre-I
Math.	DURAND	Géométrie-2

Dans cette relation, pour un cours donné toutes les combinaisons entre professeur et livre existent.

Cette rubrique contient de la redondance ce qui implique des anomalies de mises à jour.

Si l'on veut rajouter un livre pour un cours donné il faut rajouter autant de tuples qu'il y a de professeurs pour ce cours de même pour la suppression d'un livre.

Cette relation peut être décomposée en deux relations

qui sont en 4 FN.

clé

COURS	PROFESSEUR
Physique	MARTIN
Physique	DUPONT
Math	DURAND

clé

COURS	LIVRE
Physique	Mécanique I
Physique	Optique 3
Math	Algébre I
Math	Géométrie 2

e) 5 Forme normale : La 5ième forme normale concerne des cas où une information peut être reconstruite à partir d'autres informations moins redondantes.

On peut donc considérer que le 5ième forme normale est la dernière dans le processus de normalisation.

Supposons qu'on ait la relation suivante :

AGENT	COMPAGNIE	PRODUIT
DUPONT	PEUGEOT	VOITURE
DUPONT	PEUGEOT	CAMION
DUPONT	RENAULT	VOITURE
DUPONT	RENAULT	CAMION
MARTIN	PEUGEOT	VOITURE

Cette relation admet la contrainte si un agent représente une compagnie et s'il vend un produit fabriqué par cette compagnie, alors il vend ce produit pour cette compagnie.

Sous cette forme, la relation contient de la redondance et par suite des anomalies de mises à jour.

Elle peut être décomposée en plusieurs relations qui sont en 5ième FN.

AGENT	COMPAGNIE
DUPONT	PEUGEOT
DUPONT	RENAULT
MARTIN	PEUGEOT

AGENT	PRODUIT
DUPONT	VOITURE
DUPONT	CAMION
MARTIN	VOITURE

COMPAGNIE	PRODUIT
PEUGEOT	VOITURE
PEUGEOT	CAMION
RENAULT	VOITURE
RENAULT	CAMION

La décomposition devient intéressante quand il y a un grand nombre de tuples.

Si l'on veut par exemple, introduire qu'un agent vends x produits pour y compagnies, il faut rajouter x + y tuples dans la forme normalisée au lieu de x * y dans la forme non normalisée.

BASE DE DONNEES REPARTIES

INTRODUCTION

Le développement des systèmes répartis est devenu possible (et nécessaire) grâce à la conjugaison de deux facteurs.

1 - Progrès technique qui a conduit à l'augmentation du rapport performance/prix en µélectronique (la technologie des semi-conducteurs a permis de réduire le coût de fabrication d'un élément logique, processeur, mémoire, contrôleur de périphérie, d'un facteur de 10 tous les trois ans depuis 1970.) ainsi qu'à l'apparition des réseaux de transmission.

2 - Besoin des utilisateurs.
L'informatique s'introduit dans toutes les activités. De nouvelles architectures de systèmes sont nécessaires pour répondre aux besoins des utilisateurs.

Les objectifs des systèmes répartis sont d'augmenter les performances, de permettre l'extensibilité des systèmes, d'augmenter la disponibilité, et de faire partager des ressources distantes.

I - DEFINITIONS ET INTERET

Une BD Répartie est une BD où les données sont stocké physiquement sur des sites géographiquement distants. Une requête formulée sur un site peut faire intervenir des données stockées sur d'autres sites.

Chaque site contrôle les données qui y sont stockées. Les différents sites doivent garder un haut degré d'autonomie. Une BD Repartie est considérée comme une union de bases de données centralisée qui coopèrent.

Les différents sites n'ont pas de mémoire commune. Ils ne peuvent communiquer que par des messages envoyés à travers le réseau de communication. Il n'y a pas de sites maitres. En général, aucun site ne connait l'état global de la base à un instant donné.

La répartition permet d'augmenter la disponibilité du système. Si un site est surchargé, des requêtes peuvent alors être exécutées sur d'autres sites. Une panne sur un site ne paralyse pas tout le système. Les requêtes posées à ce site en panne peuvent être éventuellement traitées sur un autre site (à condition qu'il existe une copie de ces données sur ce site).

Il est plus économique et plus facile de rajouter un site à une BD Repartie que de remplacer un système centralisé par un autre système plus puissant, pour traiter un volume d'information qui dépasse la capacité du système actuel.

• on donne dans cette partie, une "vue idéale" d'une BD Répartie.

II - CONCEPTION D'UNE BDR (MEY 79)

Deux approches ont été prises pour la conception des bases de données réparties :

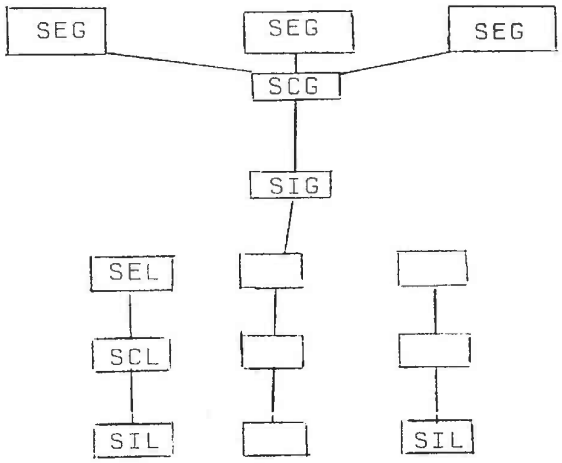
⊠ approche ascendante qui fait coopérer des bases de données pré-existantes

⊠ approche descendante où le concepteur peut décider de la répartition des données sans tenir compte des systémes antérieurs qui auraient pu exister pour l'organisation.

Dansles deux approches, une BDR sera présentée comme une BD centralisée. La répartion et la duplication des données sont transparentes à l'utilisateur.

Le schéma conceptuel (global) de la BDR et les schémas externes ont la même signification que les schémas correspondants d'une BD centralisée.

Par contre, le schéma interne de la BDR contiendra des informations sur la répartition des données ainsi que sur les correspondances entre le schéma conceptuel de la BDR et les schémas externes des différentes bases locales.



S : SCHEMA
E : EXTERNE
G : GLOBAL
L : LOCAL
C : CONCEPTUEL
I : INTERNE

III - PROBLEMES RENCONTRES EN BDR

Nous allons décrire brièvement les principaux pro-

blèmes rencontrés en BDR ainsi que les solutions proposées.

I - Coopération : les différents sites doivent coopérer pour l'éxécution d'une requête.

Il faut décomposer la requête et contrôler son éxecution sur les différents sites en optimisant le coût de l'éxécution (coût des E/S + coût UC + coût Transfert).

Ces problèmes ont été abordés au sein de l'équipe par G OD (8I)

2 - Cohérence des données : on va illustrer ce problème à partir d'un exemple.

Supposons qu'on a les deux transactions :

TI	a) X = X + 100	T2	c) X = X * 2
	b) Y = Y + 100		d) Y = Y * 2

avec la contrainte X = Y

Après l'éxécution de a) ou c), il y a incohérence temporaire car la valeur de X sera différente de celle de Y.

Alors qu'il y a conflit si T2 s'éxécute entre a) et b) car l'état de la base serait incohérent même après la fin de l'éxécution de TI et T2.

Une solution à ce problème est proposée dans ASW (76)

Dans ce papier, l'auteur démontre le théorème suivant : tout ordonnancement légal d'un ensemble de transactions bien formées à deux phases est cohérent, une transaction qui vérouille les entités dans une phase et les libère dans une deuxième phase.

Une fois qu'une entité est libérée il n'est plus possible de verrouiller d'autres entités.

Un ordonnancement des différentes actions des transactions est dit légal si à aucune étape de l'éxécution de cet ordonnancement, il n'y a de conflit de demandes de verrouillage des entités.

3 - CONCURRENCE D'ACCES

Deux types de solution ont été proposées :

A - Verrouillage des entités

⊠ Avant l'éxécution d'une opération de lecture sur un objet la transaction doit poser un verrou partageable sur l'une des copies de l'objet.

⊠ Avant d'éxécuter une opération de mise à jour la transaction doit poser un verrou exclusif sur toutes les copies de l'objet.

⊠ Une fois qu'une transaction a posé un verrou, elle ne le libère qu'à la fin de son éxécution.

Le verrouillage peut conduire à une situation d'interblocage. Une transaction a besoin pour s'éxécuter d'un objet verrouillé par une autre transaction et réciproquemment.

Plusieurs solutions existent pour résoudre le problème de l'interblocage :

a) pré - allocation de toutes les entités à la transaction.

b) allocation contrôlée : on ne permet un verrouillage que s'il n'y a pas de risques d'interblocage.

Ces deux solutions ne sont pas adaptées au contexte BD, car elles sont très restrictives (elles sont utilisées surtout dans les systèmes d'exploitation).

c) Méthodes des classes ordonnées (MAD(74)).

Les entités sont ordonnées en classe $C_1,\ldots,C_n$

Une transaction ne peut verrouiller une entité de classe C_i que si elle a déjà verrouillé toutes les entités des classes $C_1,\ldots C_{i-1}$ qui lui sont nécessaires.

d) Méthode de détection et de recouvrement

B - Estampillage LAM (78).

Chaque transaction a un numéro logique (estampille).

Les mises à jour ne sont effectuées qu'à la fin de la transaction.

Chaque transaction qui effectue une lecture ou une modification d'une entité, conduit à une mémorisation de son estampille dans l'entité en question.

En cas de conflit entre deux transactions, la plus jeune est réinitialisée avec un nouvel estampille.

4 - Copies multiples

Les données d'une BDR peuvent être partitionnées (strictement) ou dupliquées (complétement ou partiellement).

Il faut assurer la cohérence mutuelle des copies multiples.

Plusieurs solutions ont été proposées.

On peut les classer en deux catégories (LNCS 81)

α- Solutions basées sur le vote : le principe est l'échange de messages entre les sites des copies pour se mettre d'accord sur un ordonnancement global des transactions.

On peut distinguer deux types de vote.

a) vote synchrone : tous les sites votent simultanément sur une transaction donnée et procèdent en commun à l'éxécution HOL (74) ELL (78)

b) vote asynchrone : les votes pour les demandes de permission d'éxécution et les éxécutions pour les différentes transactions sont autorisées de façon parallèle THO (7I).

B - Solutions non basées sur le vote : il y en a deux types

a) solution basée sur l'existence d'un site maître.

b) solution basée sur un privilège circulant LE LANN (78)

MULTIBASE

N.B. : Dans cette partie, on expose le concept de multibase qui a été défini dans LIT (78) (80) (81) et KAB (82)

I - DEFINITONS ET EXEMPLES

Une multibase (MB) est une base de données (BD) constituée d'un ensemble de BD.

Le schéma conceptuel(SC) d'une MB est constitué de l'ensemble des schémas conceptuels des bases qui la composent.

Ex : Soient les trois BD suivantes : restaurant, cinéma, et métro

```
MB Restaurant
  BD R - Luxe
     R (numr, nomr, tel, arrond, st métro)   restaurant
     Plats (nump, nomp, type)                plats
     Menus (numr, nump, prix)                menus
  END BD
  BD R - Mode
     R (numr, nomr, tel, arrond, st métro)   restaurant
     P (nump, nomp, type)                    plat
     Menus (numr, nump, prix)                menu
  END BD

END MB

  BD Cinéma
     C (numc, nomc, tel, arrond, st métro,   cinéma
                                 nbr salle)

     F (numf, nomf, pays, genre, version)    film
     S (numc, numf)                          scéance

  END BD
```

```
BD Métro
  L(num1, nom1)                          ligne
  S(numst, nomst, arrond)                station
  LS (num1, numst)                       ligne station
END BD
```

On peut construire à partir de ces trois BD, la multibase LOISIR, dont le schéma conceptuel sera défini de la façon suivante :

```
BD LOISIR
  BD   RESTAURANT

  END BD

  BD CINEMA

  END BD

  BD METRO

  END BD

END BD
```

Le schéma conceptuel d'une multibase est dit hétérogène si les SC des BD composantes sont décrites selon plusieurs modèles (hiérarchique, réseau, relationnel...)

Le SG de la MB LOISIR est homogène car les SG des BD RESTAURANTS, CINEMA et METRO sont décrites selon le même modèle en l'occurrence le modèle relationnel.

Des contraintes peuvent exister entre les données des différentes BD d'une MB.

On en distingue principalement trois types :

I - Contraintes sématiques : Elles définissent des liens entre les schémas des différentes BD.

Par exemple, les liens d'équivalence permettent d'établir que deux objets ont la même"signification". Dans la MB "Restaurant", on peut définir des liens d'équivalences entre les relations R - Luxe, C et R - Mode C ainsi qu'entre les relations R - Luxe - R et R - Mode - R, etc...

Ce sont des équivalences au "nom prés".

D'autres types d'équivalence peuvent exister :

⊠ les valeurs de deux attributs équivalents sont exprimées avec des unités différentes. Le prix peut être exprimé en francs ou en dollars.

⊠ Un attribut est obtenu en combinant plusieurs autres attributs. Une relation EMP peut avoir un attribut salaire alors qu'une autre relation EMPLOYE a deux attributs nb heure et taux hor, dont le produit permet de calculer le salaire.

2 - Contraintes d'intégrité : Pour garantir la cohérence d'une MB on distingue notamment les contraintes qui font intervenir plusieurs BD.

Exemple : les stations de métro qui figurent dans R - Mode.R (ou R - Luxe.R) doivent se trouver dans la relation S de Métro.

3 - Contraintes de confidentialité : pour empécher un utilisateur d'accéder à des informations confidentielles.

On peut interdire par exemple l'accès simultané à certaines relations appartenant à des BD différentes pour empécher le recoupement d'informations confidentielles.

II - SYSTEME DE GESTION DE MULTI BASES DE DONNEES SGMB

Un SGMB doit fournir en plus des services offerts par les SGBD classiques :

- un langage de définition d'une MB et des liens associés à cette MB.

- un langage de manipulation de données permettant d'exprimer des requêtes s'adressant à plusieurs BD.

L'architecture d'un tel système est à 3/4 niveaux.

1) le niveau conceptuel : le SG d'une MB est obtenu en juxtaposant les SC des BD qui la constituent et en y rajoutant ensuite les contraintes.

2) le niveau externe : on peut définir des SE comme étant des vues sur le SG de la MB.

3) le niveau physique : le SP d'une MB est défini par les SP des BD qui la constituent.

4) le niveau interne logique : ce niveau optionnel existe quand le SC d'une BD (au moins) est définie comme une vue sur un certain schéma.

Ce dernier sera appelé schéma interne logique (exemple : un schéma relationnel peut être défini pour interfacer une BD définie selon le modèle réseau)

III - AVANTAGES DE L'APPROCHE MULTIBASE

Parmi les principaux avantages de l'approche multibase, on peut citer :

I - La notion de répartition devient logique et non physique.

Une BD était considérée comme répartie si les données étaient stockées sur des sites géographiquement distants et si les requêtes initialisées sur un de ces sites pouvaient faire intervenir des données stockées sur d'autres sites.

La notion de répartition était donc liée au stockage des données.

Dans l'approche MB, une BD est Répartie si c'est une MB, c'est-à-dire si son schéma conceptuel est un ensemble de SC.

Ainsi, la répartition devient une notion logique et l'utilisateur en est conscient.

2 - La possibilité de manipuler simultanément des informations se trouvant dans plusieurs BD.

Dans les approches classiques, il n'est possible d'accéder qu'à une seule base à la fois.

Par exemple, supposons qu'un utilisateur ait le droit d'accéder aux deux BD Cinéma et Métro. Alors si cet utilisateur désire connaître tous les noms des cinémas qui projettent un film donné et de plus pour chacun d'eux les numéros de ligne des stations de métro qui se trouvent à proximité, il devra :

a) écrire une requête sur la base cinéma pour sélectionner les cinémas (noms, adresses et st métro) qui projettent le film en question.

b) Ensuite se connecter à la base Métro pour connaître les lignes des stations de métro qui se trouvent à proximité

de chacun des cinémas sélectionnés précédemment.

Dans l'approche MB une seule requête aurait suffit.

2 - La définition d'un schéma global (SG) n'est plus nécessaire.

Selon les approches classiques le schéma conceptuel d'une BD répartie ressemble à celui d'une BD centralisée.

Ainsi, pour faire coopérer plusieurs BD existantes il faut définir un schéma global contenant la description des entités décrites dans les schémas des différentes bases (locales).

La définition de ce SG est une tâche difficile et parfois impossible car plusieurs problèmes se posent :

- Les mêmes entités peuvent être représentées différemment dans plusieurs BD,

Par exemple :

a) le salaire d'un employé peut être mémorisé dans une base, alors que dans une autre base, il pourrait être calculé à partir du taux horaire et du nombre d'heures de travail.

b) deux relations peuvent avoir des clés différentes bien qu'elles représentent la même entité.

- Des entités différentes peuvent avoir des noms identiques. Exemple : la relation C désigne les cinémas dans la BD "cinéma" et les cuisines dans les BD R - Luxe et R - Mode.

- Deux relations représentant le même objet peuvent avoir deux tuples de même clé, bien qu'ils décrivent deux occurrences différentes de l'objet .

Exemple : dans la BD R - Luxe, le restaurant de numéro 2

n'est pas le même que le restaurant de numéro 2 de R - Mode.R

- Le processus de normalisation devient très compliqué à mettre en oeuvre car dans les grandes organisations, les dépendances fonctionnelles deviennent très complexes.

Dans l'approche multibase ce SG est obtenu en juxtaposant les SC des différentes BD.

3 - Simplification des requêtes utilisateurs grâce à certains liens établis entre les données.

Exemple : Un utilisateur peut poser une seule requête à un élément d'une classe d'équivalence (la classe étant définie par les liens d'équivalence) au lieu de poser plusieurs fois la même requête pour chaque élément de la classe d'équivalence.

Dans le mode Loisir, si un utilisateur veut connaître les noms et les adresses des restaurants chinois, il lui suffit de poser la requête suivante :

Quels sont les noms et les rues des restaurants chinois de la relation EQUIV (R - Luxe - R)?

Ceci évite de poser deux fois la même requête pour la relation R - Luxe - R et pour la relation R - Mode - R.

La signification de la requête utilisateur est d'autant plus importante que la classe d'équivalence contient plus d'éléments.

Un utilisateur peut n'avoir dans son schéma conceptuel qu'un seul élément de chaque classe d'équivalence qui représente un intérêt pour lui.

4 - Evolutivité : L'intégration d'une nouvelle base de données à une MB ne pose pas de problèmes majeurs au niveau conceptuel contrairement à l'approche classique.

IV - SOLUTION CHOISIE DANS ADONIS

Dans notre projet on dispose d'un langage de définitions de données qui permet de définir le schéma d'une MB.

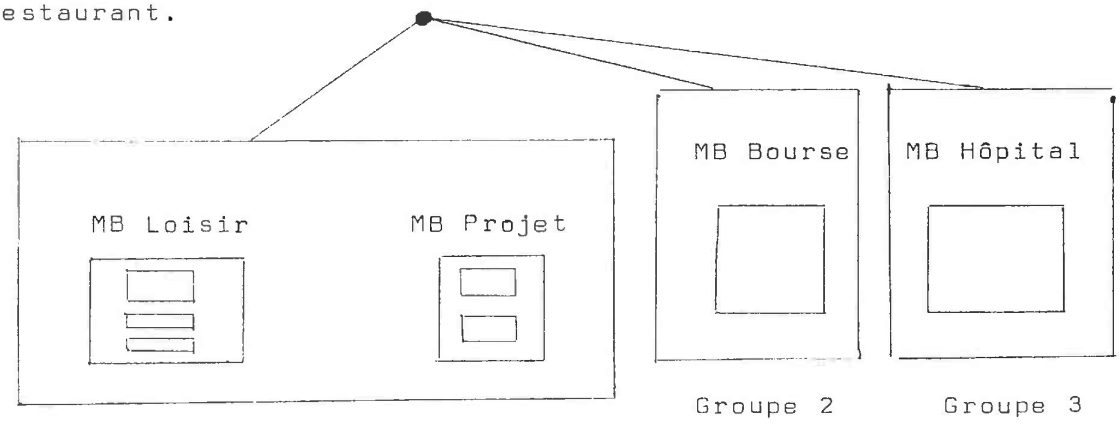
Un utilisateur peut créer plusieurs MB indépendantes :

Exemple : MB "Loisir" et MB "Projet".

Au moment de la connexion il choisit un contexte de travail qui est une de ces multibases.

Ensuite, il peut réduire ce contexte à une multibase ou à une base de la multibase principale de connexion.

Par exemple, un utilisateur peut se connecter à la MB Loisir et ensuite réduire son contexte de travail à la MB Restaurant.



Les différents (groupes d') utilisateurs peuvent partager des relations (et par suite des contextes) en donnant des droits d'accès sur leurs propres relations à d'autres utilisateurs par l'intermédiaire de la commande DONNER (ensemble (droit) SUR relation A utilisateur.

Un utilisateur peut recevoir une relation délivrée par REC - REL en indiquant dans quel contexte, il l'insére.

REC - REL nom base relation DE utilisateur relation corresp

Ce mécanisme peut être comparé aux mécanismes d'importations et d'exportations de types dans les langages de programmation.

En effet, dans une multibase, il peut y avoir un ensemble de relations importées de l'extérieur ainsi qu'un ensemble de relations exportées à l'extérieur d'un contexte de travail.

Pour plus de précision, sur les commandes DONNER, REC - REL se reporter au Chapitre IV § VI

Donc dans une multibase, il peut y avoir des relations de base dont les tuples sont stockées physiquement, des relations virtuelles que l'utilisateur définit sur des relations de son contexte de travail ainsi que des relations virtuelles définies sur des relations importées à partir d'autres contextes.

Une fois que l'utilisateur est connecté à un contexte de travail, toutes ses requêtes vont être interprétées dans ce contexte.

Par exemple, si le contexte de travail est "Cinéma", il ne pourra pas y avoir ambiguité entre la relation C de cinéma et les relations C de R - Mode et R - Luxe.

Le nom d'une relation peut être préfixé par des noms de bases ou de MB qui l'englobent pour lever toute ambiguité avec d'autres relations du même contexte de travail.

Par exemple, si le contexte est Restaurant et si on veut poser une requête sur C de R - Mode, il faut préfixer C par R - Mode.

La confidentialité des données est assurée par le

contexte de travail ainsi que par la définition de contraintes de sécurité et de relations virtuelles.

L'intégrité interne de la MB est assurée par des prédicats d'intégrité qui peuvent être posés sur les attributs les tuples, les relations, les bases et les MB.

On ne dispose actuellement que d'une seule contrainte au niveau de la base ou de la multibase : la contrainte de référence,

On envisage d'introduire une contrainte sémantique d'équivalence au nom près ce qui faciliterait l'expression de certaines requêtes.

Par exemple, pour connaître les noms des restaurants du 5ième arrondis., ainsi que leurs numéros de téléphone, il suffit, s'il y a une contrainte d'équivalence entre R - Luxe . R et R - Mode . R de poser une seule requête au lieu de deux (une sur chaque relation). La syntaxe de cette commande est :

EQUIV relation relation liste corresp. attribut

Exemple : EQUIV R - Luxe . R, R - Mode . R numr : numr, nomr, nomr
tél : tel, arrond arron
nbr salle : nbr salle

Si on pose la requête :

PROJECT (SELECT(EQUIV R-Luxe, R, arrond = 5) nomr, tel)

Elle sera traduite en

UNION (PROJECT (SELECT (R-Luxe.R, arrond = 5), nomr, tel),
PROJECT (SELECT(R-Mode.R, arrond = 5), nomr, tel), corresp)

CHAPITRE 2

ABSTRACTION ET MODULARITE

INTRODUCTION

La conception des SGBD qui sont des logiciels d'assez grande taille nécessite l'utilisation d'une méthodologie et d'un langage formel supportant cette méthodologie.

Le génie logiciel fournit des techniques permettant de réduire la compléxité et d'augmenter la sureté de fonctionnement et la correction des logiciels.

Dans ZEL (78), on trouve une revue des différentes approches utilisées pour résoudre les problèmes rencontrés à chacune des étapes du développement d'un logiciel (analyse des besoins, spécification, conception, codage, test, exploitation et maintenance.)

Nous allons nous contenter de présenter les deux concepts les plus importants utilisés dans le domaine du génie logiciel (l'abstraction et la modularité) ainsi que le concept de type abstrait qui permet de réaliser l'abstraction CHA (82).

Ensuite, nous ferons une comparaison des deux langages fortement typés (ATM et CLU) que nous avons été conduitsà utiliser pour réaliser notre prototype de SGMB.

Puis, nous présentons une méthode de spécification de bases de données avec des types abstraits algébriques MAI (81) et un langage intégré de programmation, de définition et de manipulation de bases de données (RIGEL).

I - ABSTRACTION

Une abstraction est une description simplifiée d'un objet (ou de plusieurs) qui retient certains détails ou propriétés et en cache d'autres.

Si l'objet est très complexe, il est possible de concevoir une hierarchie d'abstraction pour introduire les détails intéressants de façon progressive SMI (77) Les abstractions à un niveau donné de la hiérarchie permettent de comprendre les abstractions du niveau supérieur.

Le concept de programmation structurée est basé sur la reconnaissance des abstractions utiles.

Le problème initial est résolu en exhibant des objets et des opérations.

Si ces objets et ces opérations ne sont pas des primitives du langage de programmation, ils seront considérés comme des abstractions de nouveaux problèmes à résoudre dans les étapes ultérieures.

II - MODULARITE

La notion de modularité permet de résoudre un problème complexe en le décomposant en plusieurs sous-problèmes et en prenant soin de définir les liens logiques qui existent entre eux SOC (80) .

Un module qui résoud un sous problème peut(et doit) être conçu et compris sans faire des références à l'implantation des autres modules.

Une spécification précise du comportement d'un module permet de séparer les programmes qui l'utilisent, des programmes

qui l'implantent.

Ceci, car les premiers ne font référence qu'à la spécification du module.

La spécification d'un module (son interface) constitue une abstraction du module. Cette abstraction plus simple et plus stable que l'implantation va simplifier la conception, l'implantation et la maintenance des programmes.

Le choix d'une implantation pourra être retardé et modifié indépendamment du reste de l'application.

III - ABSTRACTION ET MODULARITE DANS LES LANGAGES DE PROGRAMMATION

La plupart des langages de programmation disposent d'un mécanisme d'abstraction procédural, assez puissant : les fonctions et les procédures.

Quand un utilisateur appelle une procédure, celui-ci n'a en fait besoin que du profil de la procédure en question.

La plupart de ces langages utilisent un mécanisme de compilation indépendante ou même séparée.

Plusieurs systèmes (ALPHARD, CLU, TYP) fournissent des mécanismes puissants d'abstraction et de modularité.

L'intérêt de ces systèmes est de donner la possibilité d'écrire des programmes de façon modulaire et de séparer la spécification de l'implantation.

En fait, ces langages permettent de développer les différents modules de façon indépendante, de retarder les choix d'implantation, de prouver la correction des programmes, de localiser les erreurs, de remplacer un module par un autre ayant la même spécification sans modifier le reste de l'application et enfin de pouvoir réutiliser les modules déjà existants.

PRO (82) présente le prototype ORSEC qui permet de rechercher des spécifications équivalentes de types abstraits algébriques par comparaisons d'exemples.

Dans LOY (82), on trouve une étude détaillée des mécanismes de programmation modulaire et de compilation séparé dans les langages de programmation.

Il a classé les langages en deux catégories : ceux qui utilisent la notion de type abstrait (SIMULA, CLU, ATM, CONCURRENT PASCAL) et ceux qui regroupent dans une même région les objets liés logiquement (MODULA, ADA).

IV - TYPE ABSTRAIT

Un type abstrait est constitué d'un ensemble d'objets et d'un ensemble d'opérations sur ces objets.

Les objets d'un certain type ne peuvent être manipulés que par l'intermédiaire des opérations du type en question.

Le comportement d'un type est complétement défini par le comportement de ses opérations et est indépendant de l'implantation des structures de données et des algorithmes associés aux opérations du type. Ce qui permet d'avoir plusieurs implantations d'un même type.

Deux approches sont utilisées pour spécifier des types abstraits : l'approche modèle abstrait et l'approche axiomatique CHA (82).

a) modèle abstrait : dans cette approche, on utilise un modèle support pour spécifier les opérations du type abstrait.

Pour prouver la correction d'une implantation d'une spécification du modèle abstrait, il faut définir la

correspondance entre la représentation utilisée et la représentation abstraite.

Exemple : Sets Represented by Sequences
WHERE
Create ( ) = Empty Sequence
Has (s,e) = IF e = First (s) THEN True
ELSE Has (Rest(s) e)
Insert (s,e) = IF Has (s,e) THEN 1
ELSE Append (s,e)

b) Approche axiomatique : les structures de données des objets du type sont définies implicitement par un ensemble d'axiomes qui spécifient les relations entre les opérations du type.

Toute structure satisfaisant les axiomes peut être un modèle de l'abstraction.

Si les axiomes sont consistants, il y aura au moins une structure les satisfaisant.

Pour établir la correction d'une implantation, il faut prouver que les axiomes sont satisfaits par les opérations de l'implantation.

Dans cette approche, chaque objet est représenté par un terme en forme normale.

Les opérations sont représentées par des réécritures sur des formes normales.

Exemple : Set IS
SORTS S, E, B
OPERATIONS Create : S
Insert : S x E S
Remove : S x E S
Has : S x E B

AXIOMS Has (Insert(S, i), J)=IF i = J THEN TRUE
ELSE Has (S, J)

Has (create, J) = False

Remove (Insert (S, i), J) = IF i ≠j then remove
(S, J
ELSE S

Remove (create, J) = Create
Insert (Insert(S, i), i) = Insert (S, i)
Insert (Insert (s, i), J) = insert (insert
(S, J), i)

L'approche modèle abstrait permet de vérifier facilement la consistance d'une spécification.

De plus, les changements mineurs du comportement d'une opération sont relativement faciles à décrire.

En effet, si la structure abstraite n'est pas modifiée, le changement de la définition d'une opération n'affecte pas les autres opérations.

Par contre, dans une spécification axiomatique, les opérations sont définies les unes par rapport aux autres.

Le changement d'un axiome peut donc affecter plusieurs opérations.

De plus, l'approche axiomatique élimine les possibilités de déformation du sens des opérations dû à l'implantation (ce qui n'est pas le cas dans l'approche modèle abstrait).

Ajoutons qu'il est aussi plus facile avec cette deuxième approche de prouver les propriétés des types et la correction de leur utilisation.

Remarque : La spécification des types abstraits

nécessite le traitement des exceptions et des erreurs.

Trois approches ont été prises :

- spécification avec définition d'opérations et d'équations avec erreurs qui prennent en compte la propagation des erreurs GOG (78)
  CHA (82)
  CHR (83)

- spécification avec prédicats OK ADJ (76)

- spécification avec préconditions GUT (77)
  REM (81)

Une étude des méthodes de spécification des cas d'exception dans les types abstraits algébriques se trouve dans BID (82).

Le problème des erreurs et des valeurs indéfinies est important dans le contexte des BD.

Une étude détaillée se trouve dans DAT (83), ainsi que dans le Chap. IV de cette thèse.

V - EXEMPLE DE SPECIFICATION ALGEBRIQUE D'UNE BD MAI (8I) WIR (82)

MAIBAUM utilise la théorie des types abstraits algébriques et un concept supplémentaire d'instance de BD (dbi) pour spécifier algébriquement une BD.

Une dbi est un ensemble de valeurs de données (structurées d'une certaine façon) en un instant donné. Pour MAI (8I) :

Une relation est un couple (ensemble d'attributs ensemble de tuples), chaque tuple étant défini sur les attributs du premier argument du couple.

Les opérations sur le type tuple sont les suivantes :

Type Tuple

NEW : set of (attribute) ⟶ tuple
STORE : tuple x attribute x value ⟶ tuple
COLUMNS : tuple ⟶ set of (attribute)
READ : tuple x attribute ⟶ value
PIECE tuple x set of (attribute) ⟶ tuple % projection
CATENATE : tuple x tuple ⟶ tuple % concatenation
COMPOSE : set of (tuple) x tuple ⟶ set of (tuple)
MATCH : tuple x tuple x set of (attribute) ⟶ bool

La sémantique de ces opérations est définie par des axiomes.

Ex: columns (store (t,a,v) = if cont (a, A) ∧ attv = a
then columns (t)
else error

Ensuite, MAIBAUM définit le type BD avec les opérations qui permettent d'ajouter ou de supprimer des tuples dans une relation et avec les opérations qui permettent d'éxécuter les opérateurs de l'algébre relationnelle (union, inters, diff, project, select, joint.)

La sémantique de ces opérations est définie par des axiomes :

type : database
Q ⟶ dbi
#crée une instance de bases de données
addtuple : relnames x tuple x dbi ⟶ dbi
deltuple : relnames x tuple x dbi ⟶ dbi
union : relation x relation ⟶ relation
inters: relation x relation ⟶ relation
diff : relation x relation ⟶ relation

PROJECT : relation x set of (attribute) ⟶ relation

RESTRICT : relation x set of (attribute) x restrictor x set of (attribute)

⟶ relation

JOIN : relation x set of (attribute) x relation ⟶ relation

COURSE : dbi ⟶ relation

COURSE : ⟶ relnames

SECTION : dbi ⟶ relation

⋮

Course et Section sont des relations de base.

Ex : ATTRIBS (CARTESIAN(r,r') =

if EQUIV (INT (ATTRIBS (r), ATTRIBS (r'), EMPTY)

Then ADD (ATTRIBS (r), ATTRIB (r'))

else error

L'intérêt de cette méthode est de fournir un moyen pour définir un BD avec les opérations que l'on peut appliquer sur cette base.

On peut d'ailleurs tenir compte des contraintes d'intégrité dans les axiomes.

MAIBAUM définit donc un outil d'aide à la conception.

Cet outil ne permet pas d'envisager une utilisation opérationnelle comme on pourrait le faire avec un SGBD classique.

Les raisons en sont :

langage trop formel, problèmes de volumes de données, temps de réponse, difficultés pour exprimer des vues ou pour modifier les contraintes d'intégrité (dans ce dernier cas, il faudrait modifier les axiomes).

VI -LE LANGAGE RIGEL (ROW (79))

RIGEL est un langage de programmation général conçu

pour le développement d'applications BD.

Il inclut des constructeurs de relations, de vues et de tuples comme types de base.

L'expression des requêtes relationnelles est intégré aux mécanismes d'itération du langage.

Une facilité d'abstraction de données est fournie, elle permet de spécifier l'interface entre un programme et une BD et de supporter l'utilisation des vues.

On va détailler chacun de ses aspects sur l'exemple de la BD suivante tirée de ROW (79)

```
 1  nameType: type = array 1..NAMESIZE of char;
 2  idNumType: type = integer;
 3  titleType: type = array 1..TITLESIZE of char;
 4  gradeType: type = (A, B, C, D, F, I);

 6  COURSE: relation
 7      c#: idNumType;      /* Course number */
 8      title: titleType;   /* Course title */
 9      p#: idNumType;      /* Teaches course */
10      key c#;             /* Unique courses */
11  end;

13  PROFESSOR: relation
14      p#: idNumType;      /* Prof's employee # */
15      name: nameType;
16      salary: real;
17      rank: (lec, asst, assoc, full, special);
18      yearsService: integer;
19      key p#;
20  end;

22  STUDENT: relation
23      s#: idNumType;      /* Student number */
24      name: nameType;
25      major: nameType;
26      level: (frosh, soph, jr, sr, grad, other);
27      key s#;
28  end;

30  /* A many-many relationship that indicates in
31     what course each student is enrolled */

33  ENROLLMENT: relation
34      s#: idNumType;
35      c#: idNumType;
36      grade: gradeType;
37      key s#, c#;
38  end;
```

Figure 1. Example data base.

I - ITérateur (ou générateur)

L'instruction FOR est utilisée pour générer une séquence de valeurs, son utilisation avec une clause WHERE permet d'exprimer des requêtes relationnelles.

```
Ex :  FOR   P    IN Professor
      WHERE      P. Rank ="asst"D O
       print     (P. name, P. salary)
     END
```

L'exécution de cet exemple entraine l'impression du nom et du salaire de tous les assistants.

La variable d'itération P est liée aux tuples de la relation Professor qui satisfont le prédicat spécifié dans la clause WHERE.

Le passage d'une séquence de valeurs à une procédure est possible :

Ex : AVG (P.salary : P in Professor)

La procédure AVG calcule une moyenne.

L'instruction P.salary : P in Professor délivre à cette procédure les salaires des professeurs.

On peut aussi construire des relations temporaires en utilisant les deux opérateurs { } constructeur de relations et < > constructeur de tuples.

Ex : Construction d'une relation temporaire contenant le numéro, le niveau et le grade des étudiants en économie.

```
temp : = {< S# : S.S#, name : S.name, level : S.level,
           grade : e.grade >:
```

```
S IN STUDENT , C IN COURSE, E IN EROL
WHERE c.title = "ECONI" and C.C# = e.C #
                        and E.s# = S.s# }
```

2 - Mise à jour

Les instructions de modification et de suppression de tuples sont spécifiées à l'aide d'une instruction spéciale. (update)

Des instructions simples (delete, replace) utilisées à l'intérieur de update effectuent les modifications désirées.

Une instruction Append permet d'insérer les tuples.

a) Modification

ex : augmenter de 10 % le salaire des professeurs.

```
Update P in Profess  Do
   replace P by
   P<salary : P.salary x 1.1>
   end
```

Cette instruction parcourt la relation Professor et sauvegarde les modifications dans un fichier.

A la fin du parcours de la relation, les mises à jour sont effectuées.

b) suppression

Ex : Supprimer tous les cours enseignés par DUPONT.

```
Update C in Course for P in Professor
  where P.name = "DUPONT"
  AND P.P# = C .P#  dO
    delete C
  END.
```

c) Insertion

Ex : Ajouter un étudiant

```
append

    s# : 1024, name : "MARTIN", major : "EEcS"
         level : "jr"

    to STUDENT
```

Il est interdit d'utiliser l'instruction append à l'intérieur d'une instruction update.

3 - Module

Les modules de RIGEL permettent de spécifier un interface (schéma externe) entre une BD et un programme.

Un module en RIGEL a trois parties :

- publique : elle définit les objets accessibles à un utilisateur du module.

- privée : elle implante les objets définis dans la partie publique.

- initialisation : elle est exécutée quand le module est défini.

4 - Vues

Une vue est définie dans un module.

Dans la partie publique, on spécifie le schéma de la vue et de certaines opérations.

Dans la partie "privée", on spécifie la correspondance entre la vue et les relations de base par une expression relationnelle définie à l'aide d'un générateur et l'implantation des opérations.

Les mises à jour sur une vue ne peuvent être effectuées qu'à travers les opérations définies dans la partie publique.

Aussi, pour les mises à jour autorisées, le programmeur doit spécifier la translation de l'opération de haut niveau en terme d'opérations sur les relations de base.

Ex : le module suivant définit une vue qui est constituée par le numéro, le nom et le grade des étudiants en économie :

```
 1  courseView: module =
 2  public

 4      ECON1: view
 5          s#: idNumType;
 6          name: nameType;
 7          grade: gradeType;
 8      end;

10      addStudent: procedure(student: nameType);
11      dropStudent: procedure(student: tuple ECON1'type);
12      assignGrade: procedure(student: tuple ECON1'type, courseGrade: gradeType);
13      countClasses: procedure(student: tuple ECON1'type): integer;
14      avgGPA: procedure(grades: generator:gradeType): real;

16  private
17      ...
18  end; /* courseView */
```

Figure 3. Public section of courseView module

```
1  courseView: module =
2  public
3      . . .
4  private

6      /* view mapping specification */

8      ECON1: view =
9        <s#:e.s#, name:s.name, grade:e.grade>:
10       e in ENROLLMENT, s in STUDENT, c in COURSE
11       where c.title = "ECON1" and c.c# = e.c#
12       and s.s# = e.s#;

14     countClasses: procedure =
15     begin
16         return COUNT(x in ENROLLMENT where x.s# = student.s#);
17     end;

19     /* initialize grade point mapping */

21     toGP: array gradeType of real:= [4, 3, 2, 1, 0, 0];

23     avgGPA: procedure =
24     begin
25         return AVG(toGP[s.grade]: s in students where s.grade not= I);
26     end;

28     addStudent: procedure =
29     begin
30         if (s in STUDENT, c in COURSE where s.name = student and c.name = "ECON1")
31         then
32             append <s#: s.s#, name: student, grade: I> to ENROLLMENT;
33         else
34             print("No such course or student");
35         fi;
36     end;

38     assignGrade: procedure =
39     begin
40         replace student'base.e by student'base.e<grade:courseGrade>;
41     end;

43     dropStudent: procedure =
44     begin
45         delete student'base.e;
46     end;

48 end; /* Course view */
```

Figure 4. Private section of courseView module

Exemple de quelques requêtes sur la vue :

I) Lister les noms des étudiants dont le grade est 'I'

```
FO R     S   IN ECONI
  WHERE      S.grade = 'I' DO
    print (S.name)

END
```

2) Inscrire un étudiant dans un cours add student ("DURAND")

3) Mettre un niveau à chaque étudiant

```
Update S in ECONI     DO
   assign Grade (s, get Grade (s .name))

END
```

5 - Critique

RIGEL est le langage qui à notre connaissance intégre le mieux le concept de BD dans un langage de programmation modulaire basé sur les notions d'abstraction et de modularité.

On voudrait néanmoins faire quelques remarques :

- Pour poser une contrainte de sécurité sur une relation, il faut définir une vue sur cette relation ainsi que les opérations ce mises à jour ce qui est un peu lourd.

La remarque est valable si l'on veut introduire une contrainte d'intégrité.

Il est difficile de définir une spécification du programme d'application sans tenir compte de l'implantation des vues, car c'est cette implantation qui définit l'intéraction

entre les vues.

- Deux vues définies sur une même relation de base peuvent être considérées comme deux objets abstraits partageant leurs représentations ce qui pose des problèmes au niveau de la vérification des programmes.

Mais, ceci est inévitable dans le domaines des BD.

- Ce langage est interprété.

VII - SYSTEME TYP

TYP (CHA (82))est un système intéractif qui intégre un compilateur du langage ATM (MIN (79) , un système de paramétrisation PRC (79), un gestionnaire des unités de programme, un connecteur dynamique (HEN (8I)),un éditeur de texte, un interface du SGF SIRIS 8 (CHA (79)) et un automate de dialogue.

Nous allons maintenant décrire ATM, langage basé sur les notions d'Abstraction, Type et Modularité.

Ce langage permet en outre de décrire des applications nécessitant le partage des ressources communes (ex : Base de données, système de documentation).

Dans ce langage, il y a quatre unités syntaxiques : type et machine (unités abstraites), capsule et module (unités d'implantation).

I - L'unité type qui décrit un type abstrait de données. Elle est définie par un identificateur qui en constitue le nom et par une liste d'opérations (pour créer et manipuler les objets de ce type), avec leurs profils syntaxiques.

La sémantique de ces opérations est décrite sous forme de commentaires.

Il y a quatre types d'opérations qui ont des droits d'accès différents sur les objets du type :

DROIT D'ACCES / OPERATION	LECTURE	ECRITURE
BUILD	O	I
CONSULT	I	O
MODIF	I	I
FUNCTION	I	O

ex : TYPE EMPLOYE

BUILD CREER (INTEGER, TEXT, TEXT)

Création d'un employé, son numéro de sécurité
sociale est égale au Ier argument, son nom au
2ième et sa fonction au 3ième.

FUNCTION NUMSS RETURNS (INTEGER)

retourne le numéro de sécurité Sociale de l'employé

FUNCTION NOM RETURNS (TEXT)

FUNCTION SPECIALITE RETURNS (TEXt)

FUNCTION EQUAL (EMPLOYE) RETURNS BOOLEAN

retourne le résultat vrai si les deux employés
ont le même numéro de Sécurité Sociale.

END

2 - L'unité capsule est utilisée pour implanter un type.

Elle contient une représentation des objets concrets du type et les algorithmes qui implantent les opérations abstraites spécifiées dans l'unité TYPE.

Ex : CAPSULE EMPI FOR EMPLOYE

```
REP
 NUMSS : INTEGER ;
 NOM   : TEXT ;
 JOB   : TEX ;

END

BUILD CREER (NSS : INTEGER, NOMSI : TEXT,T: TEXT

BEGIN
 NUMSS = NSS ;
 NOM = NOMS I;
 JOB = T     ;

END
```

3 - L'unité module comprend une liste de procédures et de fonctions partageant éventuellement une ressource commune.

Elle réalise un "morceau" de programme.

C'est un module multi-procédures au sens de Parnas (72). Un module implante une machine.

4 - L'unité"machine" décrit l'interface abstrait d'un module. Elle est définie par un nom et par une liste d'opérations (spécifications syntaxiques).

La sémantique de ces opérations est donnée sous forme de commentaires.

Ex : Machine EMPLOYES

PROC CREER

Création d'un ensemble d'employés vide

PROC AJOUTER (EMPLOYE) ⟶ BOOLEAN

Rajoute un employé
si échec de l'opération BOOLEAN est
positionné à faux sinon à vrai

PROC SUPPRIMER (INTEGER) ⟶ BOOLEAN

supprime un employé dont le numéro de
sécurité Soc. est passé en paramétre.
Si ce numéro ne correspond à aucun emplo-
yé BOOLEAN est positionné à faux sinon à
vrai.

FUNCTION EXISTE (EMPLOYE) returns BOOLEAN

ITER SELECT YIELDS EMPLOYE

itérateur qui fournit les employés un par
un.

Module EMPS FOR EMPLOYES

RESOURCE

F:FIND
les emplcyés sont conservés dans un fichier
indéxé

END

```
PROC CREER

  BEGIN

  @F CRE ('EMP', 138, 1024, 'EMPS', 8, I)

  END

PROC AJOUTER (E : EMPLOYE ) ⟶ B, BOOLEAN

  VAR

    NUMS : INTEGER
     T   : TEXT
     ⋮
  END

  BEGIN

    NUMS : =@E NUMSS

   @F   OPENU
    ⋮
  END
```

Itérateur (TOU (79)): pour pouvoir accéder à des éléments d'une certaine collection, sans faire d'hypothèse sur les structures des données ou les stratégies d'accès à ces données, un mécanisme d'itération est aussi fourni. Un itérateur est spécifié en utilisant le mot clé ITER et peut être appelé soit par l'instruction GOREACH soit par l'instruction FIRST.

```
Ex : GOREACH    EMP IN  $ EMPLOYES    SELECT
     WHERE      @EMP SPECIALITE = 'I'
     THEN WRITE (@EMP NOM)

     ENDFOREC
```

L'utilisation de la clause WHERE permet de ne

traiter que les employés qui vérifient la condition spécifiée.

La variable EMP est déclarée implicitement du type EMPLOYE.

Ex : En ATM il y a la possibilité de paramètrer les types et les machines (PRO (79).

ex : pour définir un type ensemble paramétré par le type des éléments de l'ensemble :

```
TYPEGEN ENSEMBLE
  PARAM TYPE ELEMENT (FUNCTION EQ (ELEMENT )
                 returns BOOLEAN

  CONST  INTEGER N ;
```

Tout type paramétre doit avoir l'opération EQ qui délivre un booléen

```
  END
  BUILD VIDE ;
  MODIF AJOUTER (ELEMENT) ;
  MODIF RETIRER (ELEMENT) ;
  FUNCTION EST DANS ELEMENT RETURNS BOOLEAN ;
  FUNCTION CARD RETURNS INTEGER ;

END
```

VIII LE LANGAGE CLU LIS (81)

En CLU, il y a trois types de module

I - Procédure

Elle réalise des actions sur des paramétres"données" et délivre des résultats (0, I ou plusieurs) ou signale des

exceptions si l'opération ne peut pas s'éxécuter.

L'exception est traitée par le programme appelant.

L'interface abstrait d'une procédure est constituée de son nom, de la liste des types de ses paramétres formels et des "signals".

Le corps de la procédure contient des déclarations de variables et implante l'interface abstrait.

```
Ex: racine - carrée = proc (x : real) returns (real)
                          signals (no-REAL-result)

                    IF x > = o then return ( ... )
                      ELSE signal (no-real-result)

                    END
                    END racine carrée
```

Lors de l'appel de cette procédure, z = racine-carré (y) On doit utiliser l'instruction :

Except when no-real-result : traitement END
où "traitement " décrit ce qu'il faut faire si y est négatif .

2 - Itérateur : Un itérateur définit la façon d'obtenir les éléments d'un certain ensemble.

Il les délivre à l'appelant un par un.

Un itérateur comprend une partie abstraite accessible depuis l'extérieur et une partie concréte.

Il est appelé par l'instruction FOR

```
ex : mots = iter (input : stream) yields (string)
            corps

            END mots
```

L'appel de l'itérateur s'ecrit par un exemple :

```
FOR  mot : string IN mots (inp)
  traitement (mot)

END
```

3 CLUSTER : Le CLUSTER implante un type abstrait de données.

La représentation des objets du type n'est pas accessible depuis l'extérieur de CLUSTER.

Ex : EMPLOYE = Cluster is créer, nums, nom, spécialité
equal.

```
rep = record (nss: INT, name, job : STRING)
créer = proc (N : INT, nam, t : STRING)
            returns (cvt)
          return (rep $ {nss  N, name: nam, job: T}

        END Créer

numss = proc (E : cvt ) returns (INT)
            return (E.nss)

        END numss

nom = proc (E : cvt) returns (INT)
        :
equal = proc (EI, E2 :cvt ) returns (Bool)
        :
```

IX - COMPARAISON ENTRE TYP et CLU

Nous allons essayer de dégager quelques traits de concepticns communs ainsi que les avantages de chacun de ces systèmes.

I - En TYP, il y a un mécanisme de connexion dynamique qui permet de charger les unités au fur et à mesure qu'elles sont référencées à l'éxécution alors qu'en CLU, le chargement des programmes se fait statiquement par l'utilisateur avant le début de l'éxécution de son programme.

La connexion dynamique diminue l'encombrement de l'espace mémoire ce qui est fondamental dans les applications BD.

2 - En TYP et CLU, on peut compiler des interfaces abstraits séparément des modules qui référencent ces interfaces abstraits.

Plusieurs représentations d'un même interface abstrait peuvent être compilées.

Par exemple, en TYP plusieurs capsules peuvent décrire l'implantation d'un même type ou bien plusieurs modules peuvent décrire l'implantation d'une machine.

Le choix d'une représentation est fait avant l'éxécution et est laissé à la charge de l'utilisateur (En TYP, l'option par défaut est la Ière représentation compilée de l'interface).

3 - Les deux systèmes TYP et CLU permettent théoriquement la paramétrisation bien que durant la période où l'on a commence l'implantation de notre projet en TYP, les unités génériques avec itérateurs nous aient posé de sérieux problèmes.

Les procédures, les itérateurs et clusters peuvent être paramétrés.

Ex : le type paramétré ensemble est défini par

ensemble = Cluster (T : TYPE) Is create, insert, delete, elements, equal.

```
                                    Where t has equal :
                                    proctype (t,t) returns (boo

                rep = array (t)
                create = proc ( ) returns (cvt)
                           return (rep S new ( ) ))
                         End Create

                insert = proc (E : cvt, x : t)
                           :
                         End Insert
                  :
                  :
```

4 - Variable propre ("own")

En CLU, il est possible de déclarer des variables propres dans les modules.

La durée de vie de ces variables est égale à celle du programme et non du module où elles sont déclarées.

Ces variables servent à retenir de l'information d'un appel à l'autre du module où elles sont déclarées.

Elles peuvent être initialisées lors du 1er appel.

5 - Ressource commune aux procédures

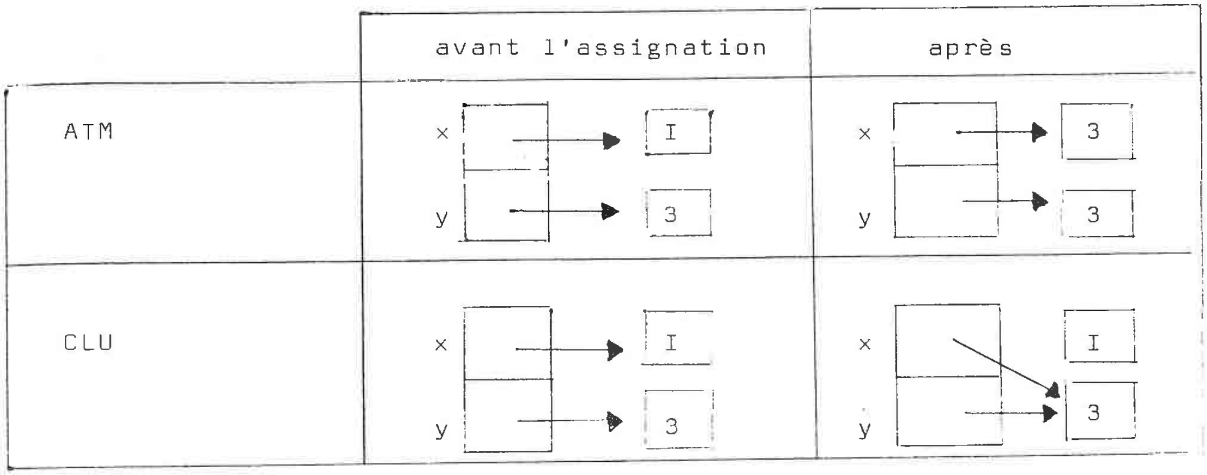
En CLU, un objet ne peut être partagé par plusieurs procédures que s'il est passé en paramétre ou bien s'il est défini dans la partie représentation (ou comme variable propre) d'un cluster qui contient les procédures en question.

En ATM, toutes les procédures d'une machine partagent la ressource de la machine.

Cette ressource étant déclarée dans la partie RESOURCE .., END du module.

6 - Assignation

Une variable en CLU ressemble à un pointeur. Une assignation x := y fait pointer la variable x à l'objet référencé par y, alors qu'en ATM, l'assignation copie y dans x (une variable en ATM "contient" un objet)



7 - Passage de paramétres

En ATM, les paramétres formels "données" référencent les objets des paramétres effectifs.

L'accès en écriture est interdit à ces variables.

Les paramétres résultats seront calculés dans les variables résultats de la procédure appelante.

L'accès en lecture aux variables résultats est interdit.

Dans le cas d'une fonction une variable temporaire est créée pour calculer le résultat qui sera recopié dans la variable de l'appelant.

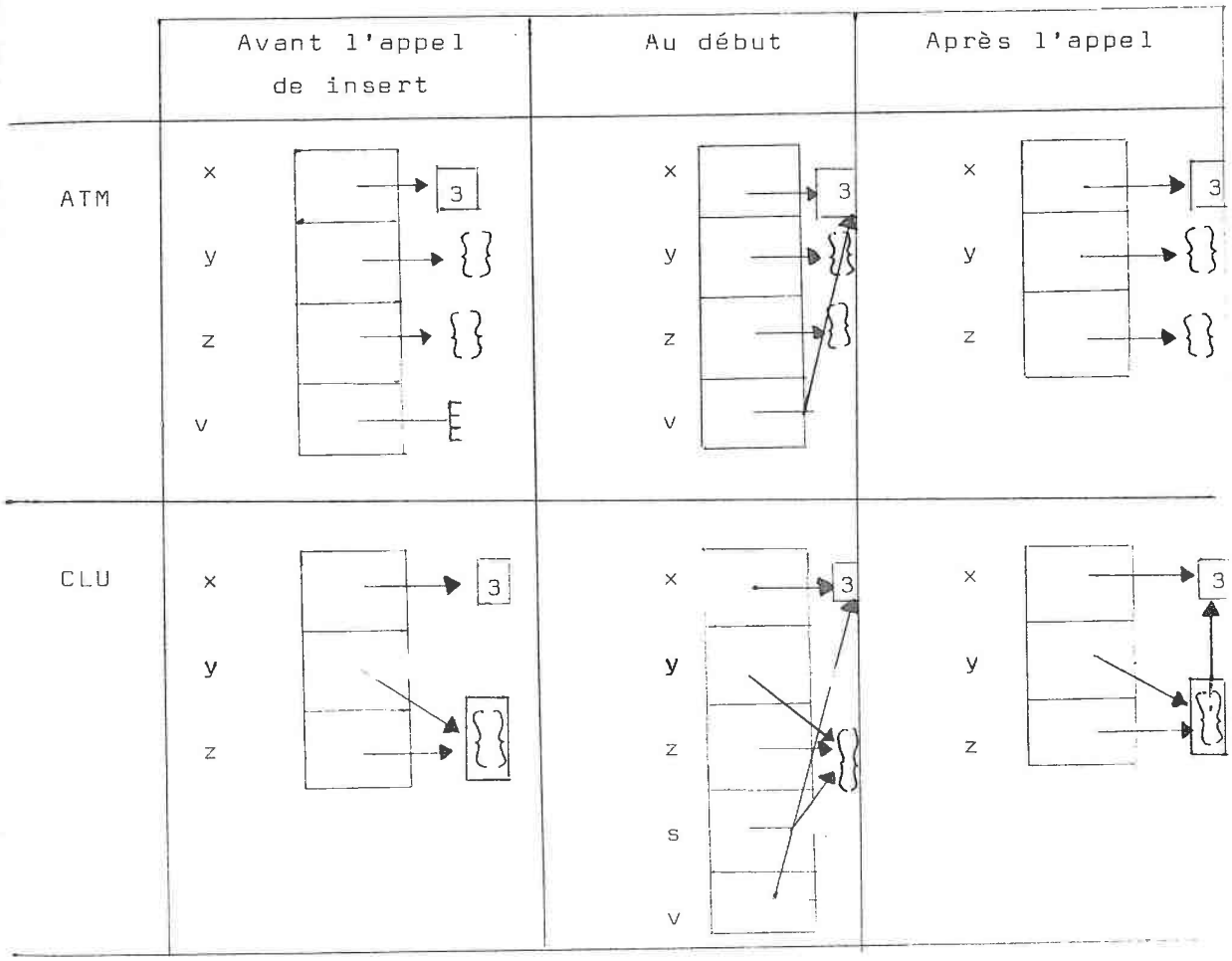
Ex : Supposons qu'on effectue les deux opérations suivantes y et z étant de type ensemble et x de type entier.

I - Assigner z à y

2 - Insérer x dans Y

On va supposer qu'on dispose en CLU du cluster "ensemble" avec l'opération insert (S ; cvt, v : intéger) et qu'on dispose en ATM du type SET et de la capsule correspondante, le profil de l'opération insert étant MODIF Insert (v : intéger)

L'état des variables peut être schématisé ainsi



8 - en CLU, on peut définir récursivement des types ex : liste = cluster is create, insert...

rep = record (c : char, rest: liste)

9 - Fichier

En TYP, le type fichier existe, il y a plusieurs, types de fichiers prédéfinis (CHA (79))- Ex : fichier, séquentiel,

indéxé ...

En CLU, il y a le type "name fichier".

A partir d'une variable de ce type, on peut créer un stream, qui nous donne accès au fichier référencé par le "name fichier".

10 - En CLU, il y a beaucoup plus de types prédéfinis (génériques) qu'en TYP.

Ex : type record, array, one - of, structure, variant, séquence.

Certains de ces types sont immuables, c'est-à-dire qu'une fois créés, on ne peut plus les modifier partiellement.

Par exemple, la séquence est un tableau immuable.

Lorsqu'une séquence est créée on ne peut plus augmenter sa taille ni changer l'un de ses éléments.

Nous avons décidé d'implanter notre SGMB en CLU bien que TYP ait certains avantages, à savoir la connexion dynamique, un contrôle des types d'accès (lecture ou écriture) aux variables passées en paramétre, une facilité de faire partager une ressource par plusieurs procédures (la "machine" d'ATM), ce système a quelques inconvénients citons en quelques uns.

Il ne supporte pas complétement la notion de paramétrisation (généricité des types et des machines avec itérateurs), les erreurs de compilation sont propagées tout au long du programme et certaines erreurs arrêtent la compilation.

De plus, pour retrouver une erreur à l'éxécution,

il faut manipuler des adresses (en hexadécimal) d'implantation des capsules et modules alors qu'en CLU, le "debugger" fournit entre autres, les valeurs des paramètres de la procédure au moment de l'interruption de l'éxécution ainsi qu'une facilité pour suivre l'éxécution d'un module instruction par instruction.

Ajoutons que CLU supporte bien la notion de paramétrisation, permet le traitement des exceptions et fournit plusieurs types prédéfinis et paramétrés ainsi que la possibilité de stocker dans un fichier une image de la mémoire et de la relire après.

CHAPITRE 3

VUES

INTRODUCTION

Une vue (ou relation virtuelle) est une relation dont les tuples n'ont pas d'existence propre dans la BD, contrairement aux tuples des relations de base qui eux sont stockés physiquement sur mémoire secondaire.

La structure d'une vue est définie implicitement par une expression relationnelle dont l'exécution permet de retrouver les tuples de la vue.

Les requêtes d'interrogation sur une vue sont transformées en requêtes d'interrogation sur le schéma conceptuel de la base.

Des problèmes apparaissent lors de la mise à jour des vues car il faut pouvoir définir les mises à jour correspondantes sur les relations de base ce qui est parfois impossible ou ambigü(non unique).

Une vue est une fenêtre dynamique sur la base : les modifications sur les tuoles des relations de base sont visibles à travers les vues définies à partir de ces relations. Ce n'est donc pas une copie de certains tuples de la base à un instant donné ("snapshot").

I - INTERET DES VUES(DAT (8I))

Pour illustrer les principaux avantages d'une vue (ou RV) : simplification de l'interface utilisateur, indépendance logique et protection des données, on va s'appuyer sur la BD Cinéma.

Cinéma (numc, nomc, tel, arrond, nb-salle, St Métro)
Film (numf, nomf, act-princ, type)
Projection (numc, numf)

I - Simplification de l'interface utilisateur.

Le schéma conceptuel (SC) d'une BD peut être très complexe.

Un utilisateur particulier ne s'intéresse qu'à une partie de la BD.

Il faut pouvoir définir un schéma externe qui lui permette d'accéder aux seules données qui l'intéressent.

Ce schéma externe sera constitué par un ensemble de vues.

Ex I : Si un utilisateur s'intéresse aux films de science fiction uniquement on lui définit une vue Film-sf (numf, nomf, act-princ) sur film par l'intermédiaire de la requête : PROJECT (SELECT(film, type = "S - F")
numf, nomf, act-princ)

Ex 2 : Si l'uitlisateur s'intéresse aux films projeté par les cinémas du 5ième arrond., on lui définit une relation cin-film-5 (nomc, tel, nomf, type) qui est le résultat de la requête PROJECT (JOIN (JOIN (SELECT(cinéma, C, arrond = 5),
Projection, numc = numc),
Film, numf = numf),
nomc, tel, nomf, type)

NB: on fait remarquer la simplification des requêtes de l'utilisateur due à l'utilisation de ces relations virtuelles.

En effet, l'utilisateur de l'ex 2 n'aura pour connaître les types des films projetés par chacun des cinémas du 5ième arrond. qu'à faire un PROJECT (cin.film-5, nomc, type)

2 - Indépendance logique : Le schéma conceptuel d'une BD évolue au cours du temps : introduction de nouveaux attributs de nouvelles relations, ou restructuration de la base.

L'utilisation des vues permet dans une certaine mesure de rendre les utilisateurs insensibles à ces changements.

Ex 3 : si on ajoute l'attribut pays à la relation film, l'utilisateur 2 peut toujours utiliser la relation cin-film-5.

Si on décompose la relation cinéma en deux relations
cinéma - arr (numc, nomc, tel, arrond, nb-de-salles)
cinéma-métro (numc, st métro)

Alors il suffit de définir la relation virtuelle cinéma comme JOIN (cinéma-arr, cinéma-métro, numc= numc) pour que les utilisateurs puissent continuer à interroger la base comme si aucun changement n'avait eu lieu.

Nous verrons plus loin que ceci n'est pas toujours vrai pour les opérations de mise à jour.

3 - Protection des données : il suffit que les données à protéger ne figurent pas dans l'extension de la vue (cf ch III)

Ex 5 : pour interdir à quelqu'un d'avoir des renseignements sur les films classés x, il suffit de lui définir une vue sur la relation film par SELECT (film, type = "X")

II - MISE A JOUR DES VUES

Une relation virtuelle peut être obtenue à partir de la combinaison de plusieurs opérateurs relationnels ainsi que des opérateurs d'aggrégation (moyenne, somme ...)

On va essayer de dégager les problèmes qui se posent lors de l'utilisation de chacun de ces opérateurs.

I - Opérateurs d'aggrégation: Il n'est pas normal

d'effectuer des opérations de mise à jour sur des RV définies à partir de ces opérateurs.

En effet, si l'on a par exemple une RV qui contient pour chaque classe le nombre d'étudiants ayant moins que la moyenne, et que l'on veuille diminuer ce nombre de I, le système ne peut pas déduire l'étudiant qui verra sa note augmenter et de combien.

2 - Opérateur SELECT,

L'opérateur de sélection de l'algébre relationnelle ne pose pas de problèmes car à tout tuple de la relation virtuelle, il fait correspondre un seul tuple de la relation de base, donc toute modification de la relation virtuelle peut être traduite en une modification de la relation de base.

3 - Opérateur PROJECT

L'opérateur de projection a la particularité d'éliminer les duplications.

Ainsi, quand la clé de la relation de base (RB) ne figure pas dans la RV, une mise à jour d'un tuple de la RV peut être propagé en mise à jour de plusieurs tuples de la relation de base.

Exemple : Soit la RV type-f = PROJECT (f, type)

a) l'opération supprimer (remplacer) le tuple (science fiction) doit supprimer tous les tuples "f" où figure (science fiction)

b) l'opération insérer (aventure) doit être traduite par l'insertion d'un tuple dans la relation"f" dont on ne connaît pas la clé, ce qui est gnéralement interdit.

D'autre part, il faut donner des valeurs nulles ou par défaut (cf.)aux attributs qui ne figurent pas dans la RV

4 - Opérateur JOIN

C'est l'opérateur qui pose le plus de problèmes. On va les illustrer à partir d'unexemple tiré de KEL (81) :

Soit les deux relations ED (emp, dept), DM (dept,mgr) et la relation virtuelle EM (emp, mgr) :

ED

emp	dept
Baker	Toys
Jones	Toys
Franklin	Art
Shaw	Shoes
White	Shoes

DM

dept	mgr
Toys	Smith
Books	King
Art	King
Stationery	Brown

EM

emp	mgr
Baker	Smith
Jones	Smith
Franklin	King

a) Une modification du tuple (Baker, Smith) de la vue en (Baker, Brown) peut être traduite de deux façons : changer (Baker, Toys) de ED en (Baker, Stationery) ou, changer (Toys, Smith) de DM en (Toys, Brown).

b) Une modification de tuple (Baker, Smith) en (Baker, Black) peut se traduire en modifiant (Toys, Smith) de DM en (Toys, Black) mais ceci change (Jones , Smith) en (Jones, Black

c) Si on veut insérer un nouveau tuple, on ne sait pas quel dept il faut utiliser pour la jointure.

d) Si on veut supprimer le tuple (Baker, Smith) est-ce qu'il faut supprimer (Baker, Toys) de ED ou (Toys, Smith) de DM ou les deux.

5 - Opérateur UNION

L'insertion d'un tuple dans la vue peut être traduite par l'insertion d'un tuple dans une ou dans les deux relations de base. La modification ou la suppression d'un typle ne peut être traduite que d'une seule façon.

6 - Opérateur INTERSECTION

Il ne pose pas de problèmes car il existe toujours une et une seule façon de traduire la mise à jour sur la vue par une mise à jour sur les relations de base.

III - APPROCHES DU PROBLEME DE MISE A JOUR

Deux approches ont été adoptées pour résoudre le problème de mise à jour des vues. La première consiste à considérer le schéma conceptuel et la vue comme des types abstraits. Ainsi, la définition des opérations de mise-à-jour de cette vue en fonction de celle des relations de base (ex : PLAIN - TAXIS). La deuxième approche consiste à définir des procédures générales de correspondance qui à partir de la définition de la vue et de l'opération de cette mise-à-jour et des données essayent de produire une traduction de cette mise-à-jour en terme de mise-à-jour sur le schéma conceptuel de la base.

Certaines approches interdisent les mises-à-jour qui ont des effets de bord (qui changent d'autres tuples de la vue que ceux spécifiés par l'opération de mise-à-jour) d'autres essayent de minimiser les effets de bord.

DAYAL et BERNSTEIN dans DAY (82) développent une théorie pour caractériser les conditions nécessaires pour transformer les mises-à-jour sur les vues en mises-à-jour sur le schéma conceptuel. Ils introduisent la notion de source et de source propre d'une mise-à-jour. Une source est propre si elle ne produit pas des effets de bord. Ils décrivent aussi des conditions syntaxiques pour des traductions correctes des mises-à-jour en terme de deux graphes : un graphe qui représente la définition de la vue et l'autre les dépendances fonctionnelles.

KEL (82) détermine les règles qui permettent d'effectuer des mises-à-jour sur des vues définies à partir d'un Join, où les attributs de jointure figurent dans la vue et où une connexion de référence directe (CDR) ou identité (CRI) existe entre ces attributs. Une CRD existe quand dans l'une des relations (la relation référencée) les attributs qui n'appartiennent pas aux attributs de jointure dépendant fonctionnellement de ces attributs. (il existe une connexion de référence direct entre dept de ED et dept de DM).

Une connexion de référence identité existe quand tous les attributs des deux relations dépendent fonctionnellement des attributs de jointure. KELLER établit que la suppression d'un tuple de la vue doit être effectuée en supprimant le tuple correspondant de la relation de référence si la connexion est CDR alors que si elle est CRI, il faut supprimer les deux tuples correspondants dans les relations de base. Pour la modification : si elle ne fait pas intervenir les attributs de jointure, il faut effectuer les modifications des tuples des relations de base affectées.Si les attributs de jointure changent et si la connexion est CRD, il faut mettre à jour un tuple de la relation de référence et insérer un tuple dans la relation référencée sinon (connexion CRI) il faut changer les deux tuples correspondants dans les deux relations de base. Les mises-à-jour ne pouvant être faites que si elles respectent le contraintes d'intégrité des relations de base.

IV - LES VUES DANS QUELQUES SGBD

On va présenter rapidement les mécanismes utilisés dans INGRES et dans SYSTEM-R pour implanter les mécanismes de vues.

1 - "Vue" en INGRES (STO (76))

Dans INGRES une commande du langage QUEL appelée DEFINE permet de définir une vue à partir des relations de la base. Rappelons qu'une requête QUEL est formée d'une instruction RANGE et d'une ou plusieurs instructions de la forme : commande (liste - cible) WHERE (qualification).

Pour définir une vue contenant les attributs "nomr" et "tel" des restaurants du 5ème arrond de la relation R-Luxe, R il faut écrire la requête :

```
RANGE OF REST IS R-Luxe.R
DEFINE VIEW RL (nom = Rest.nomr,
                tel = tel)
WHERE REST. arrond = 5
```

Une requête sur la vue sera transformée en une requête sur les relations de base par une technique de modification de requêtes (cf. CH IV §).

Ainsi la requête
```
RANGE OF A IS RL
    RETRIEVE INTO W (A· nom)
```
sera transformée en
```
RANGE OF A IS R-Luxe.R
    RETRIEVE INTO W (A.nom)
    WHERE A. arrond = 5
```

2 - "Vue" en SYSTEM-R

Dans SYSTEM-R la commande suivante de SEQUEL (cd. chI § β) permet de définir une vue :

```
DEFINE VIEW view-name
((field-name (,field-name)...))
AS SELECT-statement
```

ex :
```
DEFINE VIEW RL (nom,tel)
AS SELECT nomr, tel
FROM R-Luxe.R
WHERE arrond = 5
```

Le bloc SELECT-statement peut être très compliqué et peut faire intervenir plusieurs relations ainsi que des opérateurs d'aggrégation la mise-à-jour des vues n'est permise que si les attributs de la vue appartiennent à une seule relation de base et que si à chaque tuple de la vue on peut faire correspondre un seul tuple de la relation de base.

L'exécution réussie de l'instruction DEFINE implique le stockage de la définition de la vue dans le dictionnaire de system-R. Les opérations sur la vue seront transformées en opérations sur les relations de base par une technique de modification de requêtes.

3 - Solution choisie en ADONIS

Une relation virtuelle ou vue est définie par l'intermédiaire d'une expression de l'algèbre relationnelle. Comme dans l'approche multibase on ne peut pas créer une relation indépendamment de son contexte, une même commande permet d'insérer une vue dans une base et de la définir :
AJ-RV nom base nom-rv expression relat. corresp.

V - VALEURS NULLES ET VALEURS PAR DEFAUT : DAT (83)

a) Valeur nulle

Une étude du concept de valeur "nulle" (inconnu) et de la notion de valeur par défaut se trouve dans DAT (83) l'auteur préférant l'utilisation des valeurs par défaut. Nous allons reprendre son analyse et on indiquera la solution que nous avons adoptée.

L'introduction du concept de valeur "nulle" dans le domaine des B.D. répond à la nécessité de manipuler des informations incomplètes.

- lors de l'insertion d'un tuple, l'utilisateur ne fournit pas la valeur d'un certain attribut soit parce qu'il ne connait pas cette valeur, soit, parce qu'il manipule une vue où cet attribut ne figure pas,

- lors de l'ajout d'un attribut à une relation. Les valeurs de cet attribut pour les tuples de la relation sont inconnues au départ.

Pour que le système puisse manipuler des valeurs nulles, il faut définir :

1 - Pour chaque attribut de la base, une valeur nulle qui est différente de toutes les valeurs pouvant être prisespar cet attribut : choisir une configuration de bits différente de toutes les configurations légales de l'attribut,et si ceci n'est pas possible, introduire une information complémentaire pour indiquer si la valeur doit être considérée comme nulle ou non.

2 - Le résultat de l'application d'un opérateur de comparaison doit être considéré comme inconnu(?) si l'un des deux opérandes est **"nulle"**.

3 - Une logique à 3 états doit être appliquée

AND	T	?	F
T	T	?	F
?	?	?	F
F	F	F	F

OR	T	?	F
T	T	T	T
?	T	?	?
F	T	?	F

NOT	
T	F
?	?
F	T

Dans cette optique, deux tuples (l1,...,ln) et (d1,...dn) seront considérés comme redondants si pour tout i de 1 à n di et li sont tous les deux non nulles et di = li ou di et li sont tous les deux "nulles".

Les opérateurs de l'algèbre relationnelle peuvent être appliqués de la même façon mais il faut introduire d'autres opérateurs pour mieux tenir compte des tuples qui contiennent des valeurs nulles :

1) MAYBE - SELECT (R, Ai) = Theta - SELECT (R, IS-NULL (Ai))

(theta - SELECT est l'opération de sélection normale IS-NULL(Ai) retourne vrai si la valeur de Ai est nulle)

2) MAYBE - JOIN (R, S, Ai, Bj) = Theta-Select (PRODUCT (R,S), IS-NULL(RAi) OR IS-NULL (SBj))

3) OUTER-JOIN : il rajoute aux tuples du join normal tous les tuples de chacune des deux relations qui n'ont pas participé à la jointure, avec des valeurs nulles pour les attributs de l'autre relation.

La définition d'une dépendance fonctionnelle doit être modifiée légèrement : un attribut R.B dépend fonctionnellement d'un attribut R.A ssi à toute valeur non nulle de A est associée une seule valeur de R.B à un instant donné.

Des problèmes se posent pour définir dans quels cas on doit accepter des valeurs nulles pour certains attributs
Ex : si on a R (A,B, C) avec A → B (A n'étant pas une clé)
est-ce-que A peut avoir des valeurs nulles?
B peut avoir des valeurs nulles?

Des conséquences assez graves peuvent résulter de ces choix. En effet, si on admet que A peut prendre des valeurs nulles, le théorème suivant qui supporte la notion de normalisation ne sera plus vrai : si dans une relation R (A,B,C) on a A → B alors R peut être décomposée sans perte d'information en R1 (A,B) et R2 (A,C)
Ex : R (A, B, C)

a1	b1	c3
?	b2	c2

R1 (A, B)		R2 (A, C)			⟹	A	B	C
a1	b1		a1	c3		a1	b1	c1
?	b2		?	c2				

Dans l'approche valeur nulle, les opérateurs d'aggrégation ignorent les valeurs nulles. D'autre part, en programmant des opérations de la forme x = y doivent être remplacées par (x = y) OR (IS - NULL (x) AND (IS - NULL (y)).

b) Valeur par défaut : la notion de valeur par défaut permet comme le concept de valeurs nulles de manipuler des informations incomplètes bien qu'elle ait moins de portée théorique. Cette notion suppose que pour chaque attribut de la BD qui ne participe pas à une clé primaire, l'utilisateur fournit une valeur par défaut qui sera donnée à cet attribut si l'utilisateur

ne fournit pas de valeur à cet attribut. Les valeurs par défaut ont l'avantage de se comporter comme des valeurs normales. L'opérateur outer-join reste toujours utile contrairement aux opérateurs MAYBE-SELECT et MAYBE-JOIN, Lors de l'application d'une opération d'aggrégation, l'utilisateur doit préciser explicitement qu'il ne veut pas tenir compte des valeurs par défaut ex en SEQUEL : SELECT AVG (EMP : SAL) FROM EMP
WHERE SAL ¬ = DEFAULT (EMP-SAL)

Si l'argument sur lequel s'applique la fonction est vide, la valeur par défaut de l'attribut est retournée.

c) Solution choisie en ADONIS

Dans notre implantation, on a introduit la notion de valeur par défaut ; une valeur par domaine (pour les entiers 999, réels 999, caractères " ", chaines de caractère " "). Ces valeurs par défaut se comportent comme des valeurs nulles. En effet, l'opération de sélection de l'algèbre relationnelle ne retourne pas les tuples dont la valeur de l'attribut sur lequel porte la sélection est indéterminée, de même pour l'opérateur JOINTURE qui ne concatène pas des tuples qui ont des valeurs indéfinies pour lès attributs de jointure.

Deux opérateurs relationnelles SELECT-INCONNU et JOIN-INCONNU ont été rajoutées pour permettre respectivement d'avoir les tuples qui ont une valeur indéterminée pour un attribut et d'effectuer l'opération de jointure normale. Les valeurs par défaut seront considérées comme des valeurs normales.

Les opérateurs de calcul à savoir MAX, MIN, SOMME et MOYENNE ignorent les tuples qui admettent une valeur indéterminée pour l'attribut sur lequel s'effectue l'opération en question.

Une contrainte d'intégrité est considérée comme vérifiée si elle porte sur une valeur indéterminée. ex : si on a la contrainte prix < 300 et que l'on introduit un menu avec une valeur indéterminée pour prix (donc le système affecte au prix la valeur 999) le système accepte d'insertion du tuple.

CHAPITRE 4

SECURITE ET INTEGRITE

A - LA SECURITE DANS LES BD

0 - INTRODUCTION

Assurer la sécurité d'une BD consiste à empêcher l'accès, la modification, la dissémination ou la destruction des données par des utilisateurs non autorisés à le faire. La spécification des autorisations, c'est à dire des règles qui permettent de définir qui a le droit d'effectuer tels types d'opérations sur telles données, incombe soit à l'administrateur de la base de données(approche centralisée) soit, à l'ensemble des utilisateurs (approche répartie).

Dans notre travail, on s'est intéressé uniquement au contrôle d'accès aux données bien que pour maintenir la confidentialité d'une BD il y ait au moins deux autres problèmes qui se posent :

1 - La divulgation de l'information (confinement) :

Contrôler qu'un programme utilise correctement les données auxquelles il a le droit d'accéder. Il importe en effet, d'empêcher un programme de communiquer à un autre programme des données auxquelles il n'a pas droit d'accéder ou bien de communiquer des résultats intermédiaires (problème de référence) (MIN (76)).

2 - L'inférence :

Un utilisateur peut déduire des données confidentielle à partir de données auxquelles il a le droit d'accéder et d'information qu'il a connu depuis l'extérieur. Ce problème se pose surtout dans les BD statistiques DOB (79).

Nous commençons par illustrer sur un exemple la variété des droits d'accès qu'un utilisateur peut avoir sur les données. Ensuite, nous exposons rapidement un modèle de sécurité

basé sur un contrôle du flux de données dans un programme, les solutions proposées pour résoudre le problème de l'inférence avant de parler du contrôle d'accès, des méthodes adoptées dans quelques SGBD et enfin, nous détaillons notre solution.

I - EXEMPLE

Soit, la relation EMP (num, salaire, fonction, num-dir) citons quelques droits qu'un utilisateur peut avoir sur cette relation :

1 - Tous les droits ,

2 - Aucun droit,

3 - Lecture et mise-à-jour de tous les attributs excepté le salaire,

4 - Lecture de tous les employés sous condition que les requêtes ne comportent pas les attributs num et salaire simultanément,

5 - Lecture des données des employés dont le salaire est inférieur à 10.000F,

6 - Lecture des données des employés dont l'utilisateur est le directeur,

7 - Insertion et suppression de tuples.

D'après cet exemple, on peut remarquer que le contrôle d'accès dépend de plusieurs facteurs :

a) Du type d'accès : lecture, insertion, suppression modification.

b) du "nom" (contenant) : A partir du nom de la relation et/ou des noms des attributs on peut décider de permettre ou non l'exécution de la requête. Ce contrôle statique n'est fait qu'une fois avant l'exécution (ex : 1 - 2 - 3 - 4).

c) De la valeur (contenu) : La décision d'autoriser ou non l'accès à la donnée dépend de la valeur de cette donnée (ex : 5 - 6). Ce contrôle est dynamique. Il ne peut être fait qu'à l'exécution et doit être répété pour chaque donnée.

d) Du contexte : ceci permet de restreindre les donnée auxquelles on peut accéder simultanément. Dans l'exemple 4, on empêche l'utilisateur d'accéder au salaire d'un employé particulier.

⊠ Il faut rajouter les problèmes d'inférence et de référence évoqués plus haut.

II - CONTROLE DU FLUX DE L'INFORMATION WOOD (81)

Une première approche du contrôle du flux consiste à associer à chaque objet une catégorie et d'interdire à des sujets (utilisateurs, programmes) appartenant à d'autres catégories d'y avoir accès. Cette approche a été étendu dans le modèle multi-niveau utilisé par exemple, dans le domaine militaire. Dans ce modèle, chaque sujet a un niveau de visibilité et chaque objet (fichier, variable, utilisatuer...) un niveau de classement (ex : secret, top secret). Chaque sujet et chaque objet a un ensemble de catégories (nucléaire, espace). Un niveau de sécurité (nc1, ec1) domine un niveau de sécurité (nc2, ec2) si nc1 $\geqslant$ nc2 et ec1 $\geqslant$ ec2.

L'accès d'un sujet S à l'objet O pour effectuer l'opération t implique :

si t = lecture alors niveau (O) est dominé par niveau de S
si t = modification : niveau (O) domine niveau (S)
si t = écriture niveau (O) = niveau (S).

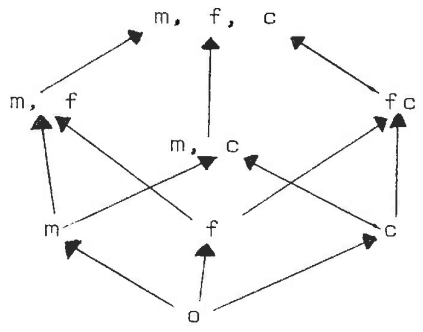
Ceci empêche les utilisateurs de copier des informations d'un niveau supérieur dans un niveau inférieur.

Un autre modèle plus général a été défini par Dening. Ce modèle comprend :

- un ensemble d'objets,
- un ensemble de processus,
- un ensemble de classes de sécurité,
- un opérateur inter classe qui spécifie la classe du résultat d'une opération,
- une relation flux.

Une relation flux entre deux classes A et B (A→B) indique que l'information de la classe A peut aller dans la classe B.

exemple :



m = médical,

f = financier,

c = criminel.

Ces modèles ne sont pas adaptés à l'introduction de nouvelles classes, aux changements dynamiques des classes des objets et à l'expression des règles d'accès dépendants des valeurs des objets ou de leurs contextes.

III - PROBLEME DE L'INFERENCE

Parmi les solutions proposées pour résoudre le problème de l'inférence, on peut citer :

- la détermination du champ potentiel de la requête ainsi que le maintien d'un historique des requêtes. Ceci est très restrictif car tous les utilisateurs n'ont pas l'intention d'accéder à des données confidentielles.

- Ne pas communiquer des résultats dont la cardinalité est inférieure ou supérieure à certaines limites dépendant du volume de l'information. ex : un utilisateur ayant le droit d'accéder à la moyenne des salaires verra sa requête rejetée s'il la pose pour un certain département où il n'y a que deux employés.

- Mélanger les données à condition que le résultat global soit correcte. ex : échanger l'adresse ou le salaire de deux ouvriers dans une BD statistique.

IV - CONTROLE D'ACCES WOOD (81)

Les techniques utilisées dans les systèmes d'exploitation pour contrôler l'accès aux ressources ont dû être étendues pour les adapter au contexte BD. En effet, dans une BD le nombre d'objets à protéger est plus grand et le "grain" de protection est plus fin (niveau multi base, base, relation, tuple, attribut).

Le modèle de protection le plus utilisé dans les systèmes d'exploitation est celui de la matrice d'accès : lignes pour les sujets (utilisateurs,...), colonnes pour les objets (périphérique, fichier), une entrée de la matrice correspond aux droits du sujet sur l'objet (éxécuter, lire, allouer) ces droits étant les règles d'accès.

Un prédicat, définissant les éléments d'un ensemble (ex : P = salaire < 10.000 F), peut être rajouté aux règles d'accès. Ainsi, le modèle peut prendre en compte les contrôles dépendants de la valeur des entités et du contexte. Une règle d'accès sera alors représentée dans ce modèle par le tuple (sujet, objet, type d'accès, prédicat). Les types d'accès dans les BD étant la lecture, insertion, suppression, mise-à-jour ou exécution.

Pour permettre à des utilisateurs de définir des autorisations, de les déléguer à d'autres utilisateurs et de les révoquer, le nom de celui qui a autorisé l'accès ainsi qu'un "flag"pour indiquer si cette autorisation peut être transmise à d'autres utilisateurs, ont été rajoutés aux règles d'accès.

Le modèle a été étendu à l'aide de spécification de procédures où l'on précise les conditions d'activation (ca). Une procédure est exécutée lorsque la règle d'accès correspondante est utilisée et que sa condition d'activation est vraie. Son exécution pouvant avoir lieu avant ou après la prise de la

décision concernant l'autorisation d'accès (l'action d'une procédure peut être la sortie d'un message d'erreur ou la notification au moniteur de sécurité de la décision.). Une règle d'accès aura finalement la forme suivante :

(autorisateur, sujet, objet, type d'accès, flag, prédicats, (cond-activ 1, proc 1), (cond. activ 2, proc 2)...).

V - SYSTEMES DE PROTECTION DANS QUELQUES SGDB

Les systèmes de protection dans les SGDB (hiérarchique et réseaux) reposent sur les notions de sous-schéma et de mot-de passe. Le sous-schéma réduit la visibilité d'une classe d'utilisateurs sur la BD. Les mots-de-passe et les verrous limitent l'accès à certains sous ensembles de données ou l'exécution de certaines opérations. Parmi les inconvénients de ces approches, on peut mentionner la gestion centralisée et la rigidité du mécanisme d'autorisation ainsi que la nécessité pour l'utilisateur de connaitre plusieurs mots-de-passe.

Dans les SGBD relationnels, les mots-de-passe ne sont pas utilisés pour la protection des données. La gestion des autorisations est dynamique. Nous allons présenter les mécanismes d'autorisation utilisés dans INGRES, SYSTEM-R et dans les machines bases de données.

1 - Sécurité en INGRES STO (76) AZR (82)

Le mécanisme d'autorisation en INGRES est centralisé. L'administrateur de la BD est responsable à la fois de la créatior de la destruction des relations partageables entre plusieurs utilisateurs et des autorisations d'accès. Dans ce système, la protection est essentiellement assurée par un mécanisme de vues (cf. ch. III §IV). Les règles d'accès sont stockées dans un tableau PROTECT.

ex : si l'administrateur de la BD veut interdire l'accès, aux employés dont le salaire est supérieur à 10,000 F, pour un utilisateur V1, il écrit la règle d'accès suivante (qui définit une "vue") :

```
RANGE OF E IS EMP
PERMIT E TO U1
RETRIEVE (E) WHERE SAL < 10.000
```

si U1 écrit la requête

```
RANGE OF E1 IS EMP
RETRIEVE (F1) WHERE fonction = ouvrier
```

alors le système transforme cette requête en rajoutant les règles de protection :

RANGE OF E1 IS EMP
RETRIEVE (E) WHERE fonction = ouvrier
AND sal < 10.000

Comme il peut y avoir plusieurs règles d'accès, l'algorithme de modification des requêtes est le suivant :

```
si l'utilisateur est le créateur de la relation alors autoriser
|   l'accès
|
|   sinon pour chaque liste cible lc1 de la requête
|       |   1 - trouver l'ensemble S des attributs de lc1
|       |   2 - trouver l'ensemble T des règles d'accès ayant la
|       |       même commande que celle de lc1 et dont la liste
|       |       cible contient tous les attributs de S.
|       |
|       |   Si T est vide, alors interdire l'accès
|       |     |
|       |     |   Sinon, 3 - ignorer les règles d'accès de T dont
|       |     |              les listes cibles contiennent les listes
|       |     |              cibles d'autres règles. Soient T' l'en-
|       |     |              semble résultat et P1, - Pn les prédicats
|       |     |              des règles qui restent
|       |     |
|       |     |          4 - remplacer le prédicat Pr relatif à la
|       |     |              requête par Pr ∧ (P1 VP2 V... Vpn)
|       |     |
fsi fpour fsi
```

2 - Sécurité en SYSTEM-R WIL (81)

Le mécanisme d'autorisation en SYSTEM-R est reparti. Tous les utilisateurs ont la possibilité de donner et/ou de révoquer les autorisations d'accès sur les objets (relations, programmes) auxquels ils ont le droit d'accéder.

Le concept de vue (ou relation virtuelle) est utilisé pour assurer la sécurité des données. Pour interdire à un utilisateur, l'accès à certaines données, il suffit de définir une vue (cf. ch III §).

ex : soit la relation EMP (num, salaire, dept, num-dir)
Pour interdire à un utilisateur l'accès :

1 - à l'attribut salaire, on lui définit la vue E1 :

```
DEFINE VIEW E1 AS
SELECT num, dept, num-dir
FROM EMP
```

2 - aux employés dont le salaire est supérieur à 10.000 F

```
DEFINE VIEW EMP2 AS
SELECT   *
FROM EMP
WHERE SAL < 10.000
```

3 - au salaire d'un employé particulier mais lui permettre d'accéder à la moyenne des salaires par département

```
DEFINE VIEW EMP3 AS
SELECT (dept, AVG (salaire))
FROM  EMP
GROUP BY dept
```

On remarque néanmoins que la notion de vue ne permet pas de réaliser le contrôle d'accès dépendant du contexte.

En system-R, il y a trois types de sujets ; utilisateur simple, groupe d'utilisateurs et publique (tous les utilisateurs).

Les droits d'accès (privilèges) sur les relations sont : SELECT (lecture et définition de vues), INSERT, DELETE, UPDATE (on peut préciser les différents attributs), EXPAND (pour rajouter des attributs à une relation de base) et INDEX (pour créer des index sur de nouvelles relations).

Le **seul** droit sur un programme est le RUN.

Deux privilèges spéciaux existent :

- " resource autority" pour créer des relations de base,
- DBA autority permet d'avoir tous les privilèges sur tous les objets de la base.

Quand un utilisateur crée une relation de base, il dispose de tous les droits sur cette relation. Lorsqu'il définit une vue il a les mêmes privilèges sur la vue que sur la relation de base à partir de laquelle la vue a été définie (si plusieurs relations figurent dans la définition de la vue, l'utilisateur n'a que le SELECT).

La propagation des privilèges se fait par les 2 commandes

GRANT (privilèges) ON (nom objet) TO (liste utile) WITH GRANT OPTION

REVOKE (privilèges) ON (nom objet) FROM (liste utilisateur.

Toutes les autorisations que les sujets ont sur les objets sont maintenues par le système dans une relation SYSTABAUT

Les lagorithmes des commandes GRANT et REVOKE sont présentés dans WAS (76) et WIL (81).

3 - Sécurité dans les machines BD MEN ( )

Une machine BD est un ordinateur spécialisé capable de manipuler de grands volumes de données. Le mécanisme de sécurité fait partie intégrante de la machine. Il repose sur le concept d'atomes de sécurité : aggrégats de données définis par l'utilisateur en termes de prédicats.

Une requête à la machine BD est une expression booléenne de prédicats en forme normale disjonctive
ex : (Sal $<$ 6.000) $\wedge$ (Sal $>$ 5.000) $\vee$ (fonction = directeur).

Les requêtes sont utilisées pour spécifier des contrôles d'accès ce qui permet de protéger toute donnée accessible par une requête. Ces requêtes sont appelées des conjonctions de prédicats de sécurité. Pour illustrer le mécanisme de sécurité, prenons un exemple : supposons qu'on a les deux conjonctions de sécurité.

C1 : (sal $>$ 5.000) (Sal $<$ 6.000)
C2 : fonction = directeur,
pour le fichier, il y aura quatre atomes de sécurité :

- le premier contenant les directeurs touchants entre 5.000 et 6.000,
- le second contenant les directeurs touchants moins de 5.000 ou plus de 6.000
- le troisième contenant les employés qui ne sont pas des directeurs et qui touchent moins de 6.000 et plus de 5.000,
- le quatrième contenant les employés qui ne sont pas des directeurs et qui touchent moins de 5.000 ou plus de 6.000.

On remarque qu'il n'y a pas d'enregistrement communs entre deux atomes de sécurité.

Chaque utilisateur aura le droit d'accéder à certains atomes de sécurité.

Avec chaque conjonction de sécurité, le créateur du fichier peut spécifier pour chaque utilisateur une liste des accès simultanés non permis sur les enregistrements vérifiant la conjonction de sécurité (pour contrôler l'accès dépendant du contexte ex : (nom, sal) NO - READ).

Les algorithmes utilisés pour contrôler l'accès ainsi que pour répartir les données entre les atomes de sécurité sont détaillés dans (MEN ( )).

Avec une machine BD, tout accès aux données est contrôlé. A l'exécution d'une requête le système n'accède qu'aux enregistrements vérifiant les contraintes de sécurité ; ce qui n'est pas le cas dans les techniques basées sur les vues ou la modification des requêtes.

L'inconvient majeur de cette approche apparait lorsque les conjonctions de sécurité sont modifiées.

VI - SOLUTION CHOISIE EN ADONIS

Parmi les principales caractéristiques du mécanisme d'autorisation du projet, on peut citer :

- La répartition du pouvoir d'autorisation,
- L'utilisation de vues et de prédicats pour protéger les données,
- La gestion dynamique des droits d'accès.

On va détailler chacun de ses aspects :

1 - Répartition du pouvoir d'autorisation

On a adopté la solution utilisée en system-R, à savoir, que tout utilisateur peut avoir le droit de créer des relations, d'autoriser ou de révoquer des droits d'accès (lecture, insertion, suppression, mise-à-jour).

2 - Protection des données par vues et prédicats

L'intérêt des vues pour réaliser un contrôle d'accès dépendant du nom ou de la valeur des entités a déjà été évoquée. Pour nous, une vue sera exprimée par une requête de l'algèbre relationnelle.

ex* : RV EMP1 COMME PROJECT (EMP, num, nom)
CORRESP . . .

* pour la syntaxe exace, cf. annexe 1.

Afin de pouvoir exprimer certaines contraintes de sécurité (ex ; les valeurs d'un attribut dans une relation doivent se trouver dans une autre relation) sans étendre la syntaxe de l'algèbre relationnelle, nous avons décidé d'utiliser des prédicats pour renforcer la protection, ceci permettant de spécifier des protections dépendant du contexte. Ces prédicats de sécurité seront inclus dans la définition du schéma de la vue (de la base ou multi-base). Dans une première étape, cinq types de prédicats sont utilisés. A chaque type de prédicat est associé une procédure qui retourne une relation temporaire contenant les tuples de la relation passée en paramètre et qui vérifient le prédicat associé à cette relation. La syntaxe des prédicats est la suivante :

1 - SELVAL (attribut) (oper-de-comparaison) (valeur)
- C'est équivalent à l'opération SELECT de l'algèbre relationnelle.

2 - (Attribut) ENTRE (valeur) (valeur)
- la valeur de l'attribut doit être dans l'intervalle.

3 - ORDRE (attribut) (oper-de-comparaison) (attribut)

- la relation exprimée par l'opérateur de comparaison doit être vérifiée (pour les tuples auxquels l'utilisateur a le droit d'accéder) entre les valeurs des deux attributs de chaque tuple ; ex : débit < crédit.

4 - MOYENNE (oper-de-comparaison) (attribut)
- Les tuples délivrés à l'utilisateur sont ceux dont la valeur de (l'attribut) vérifie la relation exprimée par l'opérateur de comparaison avec la moyenne des valeurs de cet attribut pour l'ensemble de la relation protégée.

III GESTION DYNAMIQUE DES DROITS D'ACCES

Lorsqu'un utilisateur crée une relation de base, il reçoit tous les droits sur cette relation : lt, st, it, mt (l : lecteur, s : suppression, i : insertion, m : modification et t : transmission).

Quand un utilisateur crée une relation virtuelle, ces droits sur cette relation seront déduits de ces droits sur les relations qui ont servi à la définition de la relation virtuelle.

Trois commandes permettent aux utilisateurs de donner, revoquer et recevoir des droits :

1 - La commande DONNER ensemble (droit) SUR relation
A utilisateur

permet à un utilisateur d'offrir un ensemble de droits sur une relation à un autre utilisateur.

2 - La commande REVOQUER ensemble (droit) SUR relation
DE utilisateur

permet de retirer des droits offerts à un utilisateur sur une relation. Si ce dernier avait à son tour transmis ces droits à d'autres, ils leurs seront révoqués aussi.

3 - La commande REC-REL nom-base nom-rel-receveur
DE nom-donneur nom-rel-donneur
CORREP liste-de-correspondance

permet à un utilisateur d'ajouter une relation virtuelle à l'une de ses bases. Cette relation lui a été offerte par l'instruction DONNER...

Le système maintient un tableau d'autorisation qui contient les informations suivantes :

- nom du donneur,
- nom du receveur,
- nom de la relation par rapport au donneur,
- nom de la relation par rapport au receveur,
- ensemble des privilèges du receveur sur la relation
- type du receveur : utilisateur ou relation virtuelle

Nous allons illustrer le mécanisme d'autorisation à partir d'un exemple :

* supposons qu'un utilisateur U1 crée une multi-base vide Loisir et qu'ensuite U1 rajoute une base cinéma : la relation C et la relation F. Le tableau d'autorisation va contenir les deux tuples suivants :

1 - U1, U1, Loisir-cinéma-C, Loisir-cinéma-C, (LT, IT, ST, MT), utilis
2 - U1, U1, Loisir-cinéma-F, Loisir-cinéma-F (Lt, IT, ST, MT), utilis.

* Si U1 émet la commande :
DONNER LT, IT SUR Loisir-cinéma-C à U2, le système va rajouter le tuple suivant au tableau d'autorisation.
U1, U2, Loisir-cinéma-C, " ", (LT,IT), utilis.

* maintenant, pour que U2 puisse rajouter une vue sur la relation Loisir-cinéma-C à une base qui lui appartient alors U2 doit émettre la commande :
REC-REL exploit cin DE U1 Loisir-cinéma-c liste corresp.

Après cette commande, le système va remplacer le " " dans le tuple précédent par exploit-cin.

* L'utilisateur U2 peut à son tour offrir la relation exploit-cin à d'autres utilisateurs
ex : DONNER 1 SUR exploit-cin A U3
Le tuple suivant sera rajouté au tableau d'autorisation U2, U3, exploit-cin, " ", (1), utilisateur.

A cet instant, le tableau d'autorisation contient les tupes suivants :

U1, U1, Loisir-cinéma-C, Loisir-Cinéma-C (LT, IT, ST, MT), utilis
U1, U1, Loisir-cinéma-F, Loisir-cinéma-F (LT, IT, ST, MT), utilis
U1, U2, Loisir-cinéma-C, exploit. cin, (LT, IL), utilis
U2, U3, exploit-cin, "U", (L), utilis.

* Si U1 émet la commande :
REVOQUER lt, it SUR loisir cinéma.C DE U2
L'état du tableau d'autorisation devient :

U1, U1, Loisir-cinéma-C, Loiisr-cinéma-C (LT,IT,ST,MT) utilis
U1,U1, Loisir-cinéma-F, Loisir-cinéma-F, (LT, IT, ST, MT) utilis
U1, U2, Loisir-cinéma-C, exploit-cin (" "), utilis
U2, U3, exploit. cin, " ", (" "), utilis

* Quand un utilisateur définit une relation virtuelle le système rajoute un tuple pour chaque relation participant à la définition de la relation virtuelle en plus du tuple qui définit les droits de l'utilisateur sur cette relation.

ex : si U1 émet la commande :

AJ-RV Loisir-Restaurant, R-Luxe cin COMME PROJECT (C, nom, tel) corresp.

Les deux tuples suivants seront rajoutés à tabaut :

U1, U1, Loisir-Restaurant - R-Luxe, cin, Loisir-Restaurant. R-Luxe cin, (LT), utilis

U1, Loisir-Restaurant - R-Luxe, cin, loisir-cinéma-C, " ", (LT), rela. virt.

L'algorithme de révoquation est le suivant :

```
REVOKE (tabaut, revoker, revokee, rel-revoker, ensdroit)
   crevoker = revoker          % revoquateur courant
   crevokee = revokee          % revouqé courant
   Pour chaque l : droit de ensdroit faire
      remplacer d par b  dans le tuple t1 = (donneur, receveur, rel-donneur, rel-receveur, (dr),type
      où donneur = crevoker et receveur = crevokee et rel-donneur = rel-revoker et d ∈ (dr)
        si d est transmissible
          % on va révoquer les droits que le revokee aurait transmis
          alors si type = "utilisateur"
                    alors rel-revokee= rel-receveur
                    pour chaque tuple t2 = (donneur, receveur, rel-donneur, rel-receveur, (dr),type)
                        où donneur = crevokee et rel-donneur = rel-revokee et t2 ≠ t1
                         REVOKE (tabuat, crevokee, receveur, rel-revokee, (dr))
                    f pour
          sinon % type = rel. virtuelle
          pour chaque tuple t2 = (donneur, receveur, rel-donneur, rel-receveur, (dr), type)
              si (donneur = crevokee) et (rel-donneur = rel-revokee) et (t2 ≠ t1)
                 alors REVOKE (tabaut, crevokee, receveur, (d))
                 sinon si (donneur = crevoker) et (receveur = crevokee) et (t2 ≠ t1)
                           alors remplacer d par b dans t2
          fpour                fsi
                fsi
        fsi
   fpour
```

B - INTEGRITE DANS LES BD

0 - INTRODUCTION

Une BD contient des informations relatives à une entreprise et il est important que ses informations soient exactes (qu'elles reflètent l'état réel de l'entreprise). Le contrôle d'intégrité interne (sémantique) permet de détecter si les valeurs des données sont plausibles et si les modifications de ses valeurs sont cohérentes. Les règles d'intégrité internes définissent les états et les transitions d'état valide de la BD. Un contrôle d'intégrité externe est nécessaire pour assurer la cohérence de la base lors d'accès concurrents ou lors d'une panne.

Dans le cadre de notre travail, on s'est intéressé aux problèmes liés au contrôle d'intégrité interne. Dans une première partie, nous allons exposer les différents types de CI interne, ensuite, nous présenterons les mécanismes de contrôle d'intégrité utilisés dans quelques SGDB puis, la solution que nous avons adoptée.

I - CONTRAINTES D'INTEGRITE INTERNES FLO (82)

Les C.I. internes sont des assertions sur les éléments du schéma conceptuel. On peut les décomposer en deux classes : celles qui caractérisent les états valides (C.I statique) et celles qui définissent les transitions d'états valides (C.I. dynamique). Les premières peuvent être exprimées par des invariants, les secondes, par des pré-post conditions.

On dit qu'une C.I. est immédiate si elle doit être vérifiée après chaque opération sur la BD. Elle est dite différée si elle est vérifiée après une série d'opérations (transaction), la base peut se trouver dans un état incohérent avant la fin de l'exécution de la transaction, ou à la demande d'un utilisateur.

Les dépendances fonctionnelles(DEL(80))représentent un type important de C.I. Le processus de normalisation permet de concevoir le schéma de la BD de façon à tenir compte implicitement de certaines de ses dépendances.

D'autres types de contraintes ne peuvent être incluses dans le modèle de données et doivent être définies explicitement. Nous allons décrire quelques types de contraintes en les séparant en deux classes : statiques et dynamiques.

1 - Contrainte statique

On va distinguer différents types de C.I. statiques suivant qu'elles portent sur un attribut, un tuple, une relation ou une BD.

a) contrainte attribut : elle est définit sur un seul attribut. Elle est vérifiée en comparant la valeur de cet attribut à un ensemble. Ex : le salaire d'un employé doit être supérieur à 3.000 F.

b) Contrainte tuple : elle porte sur plusieurs attributs du même tuple

Ex : soit, la relation P (num, qte-produite, qte-stockée)

C1 : la valeur de l'attribut qte-produite doit être supérieure à celle de la qte-stockée

C2 : si la valeur de qte-produite est supérieure à 1.000 alors, la valeur de la qte-stockée doit être supérieure à 50.

c) Contrainte relation : elle porte sur un ensemble de tuples. Elle peut faire intervenir un seul ou plusieurs attributs.

Ex : - la moyenne des salaires est supérieure à 4.000 F
- il y a au plus 5 directeurs,
- un directeur ne peut avoir plus de 50 employés sous sa direction immédiate.

d) Contrainte base ou multi-base : elle fait intervenir plusieurs relations appartenant à la même BD ou à une multibase.

Ex : soient les relations F (NUM, NOM, adresse)
P (NUM, NOM, poids)
FP (NUM -F, NUM-P, quantité)

Containte : les NUM-F de FP doivent se trouver dans F (et NUM-P dans P).

Ex 2 : soient les relations Avion (num, type, capacité, places)
Vol (num, num-avion, nombre-places - réserve)

contrainte : le nombre de places réservées sur un vol doit être inférieur à la capacité de l'avion assurant ce vol.

2 - Contraintes dynamiques :

Elles font intervenir la notion du temps. On peut les classer en différents types comme les contraintes statiques. On va se contenter ici, de donner quelques exemples :

- le nouveau salaire d'un employé doit être supérieur à l'ancien salaire

- si l'ancien statut = célibataire alors le nouveau statut ne peut être que marié ou décédé

- la nouvelle moyenne des salaires ne doit pas dépasser de plus de 10 % l'ancienne moyenne.

II - MECANISME DE CONTROLE D'INTEGRITE DANS QUELQUES SGDB

On va décrire brièvement les mécanismes de contrôle d'intégrité dans QBE, INGRES et SYSTEM-R.

1 - Query By Example (DATE (79))

Toutes les opérations dans QBE sont spécifiées en mettant des entrées dans des tables. Il en est de même pour la spécification des contraintes d'intégrité. Elles sont introduites avec les facilités d'insertion normales en ajoutant le mot clé CONSTR. On ne distingue pas les contraintes immédiates des contraintes différées mais comme en QBE les vérifications sont effectuées après que toutes les commandes se trouvant sur l'écran soient exécutées il suffit, pour différer l'application d'une contrainte jusqu'à la fin de la transaction, d'introduire la transaction en une seule étape à partir de la console.

ex :

Fournisseur	num	nom	statut	ville	
I. CONSTR I				Londres Paris Athènes	les valeurs autorisées de l'attribut vill
I. CONSTR(U) I	num 1 num 1		> st st		en cas de mises à jour, le nouveau statut doi avoir une valeu supérieure à l'ancienne

2 - INGRES

Les contraintes d'intégrité sont implantées par la technique de modification de requêtes comme pour le contrôle

d'accès. Les contraintes relatives à un attribut ou à un tuple sont les seules possibles. Les assertions d'intégrité sont spécifiées sous forme de clauses de qualification de QUEL. Elles doivent être appliquées sur les opérations qui portent sur la relation spécifiée dans la partie RANGE. Pour chaque assertion un arbre de la qualification est créé et stocké dans le catalogue INTEGRITY avec une indication de la relation et des domaines concernés. Au moment de l'exécution de la requête, le système cherche les contraintes d'intégrité qui s'y rapportent et les rajoute à la partie qualification de la requête.

Ex : soit la contrainte RANGE OF E IS EMP
INTEGRITY CONSTRASNT IS E.SAL > 8.000

si l'utilisateur pose la requête

RANGE OF E IS EMP
REPLACE E(SAL = 0.9 * E. SAL)
WHERE NAME = DUPONT

elle sera transformée en :

RANGE OF E IS EMP
REPLACE E(SAL = 0,9 * E.SAL)
WHERE NAME = DUPONT
AND 0.9 * E.SAL > 8.000

3 - SYSTEM R

Des assertions du langage SEQUEL sont utilisées pour spécifier les contraintes d'intégrité. La syntaxe est :
ASSERT nom-assertion (ON nom-relation) : prédicat.

Quand une assertion est posée, le système vérifie si elle est respectée par les données. Si c'est le cas, elle est conservée jusqu'à sa suppression par une commande DROP ASSERTION.

Toute modification qui ne vérifie pas les contraintes d'intégrité est rejetée. Les contraintes s'appliquent à des tuples ou à des ensembles de tuples. Quand une assertion décrit les transitions d'états permises, les mots clés OLD et NEW sont utilisés dans SEQUEL pour référer les données avant et

après la modification. Les contraintes d'intégrité ne sont vérifiées qu'à la fin d'une trasaction à moins que les contraintes ne soient déclarées en IMMEDIATE.

Il est possible à l'intérieur d'une transaction de mettre des points d'intégrité. Ainsi, si une transaction ne vérifie pas les contraintes d'intégrité, les données reprennent les valeurs qu'elles avaient au dernier point d'intégrité.

Il est possible de définir des procédures (trigger) pour propager les mises-à-jour.

Nous allons donner quelques exemples de contraintes d'intégrité en SYSTEM-R :

- ASSERT A1 ON EMPLOYEE : SAL > 8.000

- ASSERT A2 =(SELECT NUM-F FORM F-P)
IS IN
(SELECT NUM FROM F)

- ASSERT A3 ON UPDATE OF EMP(SAL) : NEW SAL
>
OLO SAL

pour diminuer le nombre d'employés de 1 un "trigger" sera défini de la façon suivante :

```
DEFINE TRIGGER emp-projet SUP
ON     DELETION OF emp-projet
       UPDATE projet
       SET nbr-employés = nbr-employés - 1
       WHERE num = OLD emp-projet. num- proj.
```

Il faut d'ailleurs pouvoir indiquer à quel moment (avant ou après la mise-à-jour) ces procédures ("trigger") doivent être appelées. Ainsi, la syntaxe d'une procédure d'intégrité ressemble à :

nom de la procédure { AVANT / APRES } { INSERTION / SUPPRESSION / MODIFICATION } DE relation : corps procédure

On peut aussi spécifier l'action à entreprendre si l'opération ne peut pas s'effectuer à cause, par exemple, d'une violation d'une autre contrainte d'intégrité.

III-SOLUTION CHOISIE EN ADONIS

Nous allons détailler les différentes catégories de contraintes retenues dans l'implantation ainsi que le mécanisme de maintien de la cohérence.

A - CATEGORIES DE CONTRAINTES D'INTEGRITE

1 - Contraintes statiques :

a) Sur un attribut : il y a deux contraintes SELV et DE-A

- SELV attribut opérateur valeur
  les valeurs de l'attribut (arg 1) doivent être arg 2 à une certaine valeur (arg 3)

- DE-A attribut (valeur-min) (valeur-max)
  les valeurs de l'attribut (arg 1) doivent être comprise entre une valeur minimale (arg 2) et une valeur maximale (arg 3)

b) Sur un tuple : la contrainte ORDRE attribut op attribut permet d'assurer que pour chaque tuple de la relation, la valeur de l'attribut (arg 1) est arg 2 à la valeur de l'attribut (arg 3).

c) Sur une relation : trois contraintes sont disponibles dont deux de cardinalité :

- MOYENNE (attribut) (opérateur) (valeur)
  ⊠ la moyenne des valeurs de l'attribut (arg 1) de la relation doit être arg2 à une certaine valeur (arg 3).

- NOCCS (attribut) (opérateur) (valeur)
  - le nombre de valeurs différentes de l'attribut (arg 1) doit être arg 2 à une certaine valeur (arg 3).

- NOCCM (attribut) (attribut) (opérateur) (valeur)
  - pour une valeur de l'attribut (arg 1) : le nombre de valeurs différentes de l'attribut (arg 2) doit être arg 3 à la valeur (arg 4)

d) Sur une base ou une multi base : on peut poser une contrainte de référence au niveau d'une base ou d'une multibase pour s'assurer que les valeurs d'un attribut d'une relation doivent être un sous ensemble des valeurs d'un attribut d'une autre relation :
REF (relation) (attribut) (attribut)
Ex : si on pose le prédicat REF PLATS nump nump sur la relation "MENUS" alors, tout numéro de plats de la relation "menus" doit se trouver dans la relation "Plats".

2 - Contrainte dynamique :

On tient compte d'une seule contrainte dynamique.
N-A attribut opérateur
- La nouvelle valeur de l'attrivut (arg 1), pour un tuple donné de la relation sur laquelle on pose la contrainte, doit être arg 2 à l'ancienne valeur de cet attribut pour la même tuple.

B - MECANISMES DE MAINTIEN DE LA COHERENCE

Quand un utilisateur pose une contrainte sur une relation, il peut indiquer après quel type d'opération de mise-à-jour cette contrainte doit être vérifiée. La syntaxe est :

POSER-CI nom-contrainte [(INSERTION) / (SUPPRESSION) / (MODIFICATION)] nom-relation prédicat

Si l'utilisateur n'indique pas le type d'opération, les opérations par défaut prises par le système sont schématisées dans le tableau suivant :

prédicat / Opération	SELV DE-A ORDRE	REF 1°rel	REF 2°rel	NOCCS NOCCM	N N-A
INSERTION	1	1	0	1	0
SUPPRESSION	0	0	1	1	0
MODIFICATION	1	1	1	1	1

* Pour le prédicat référence, on distingue la 1° relation sur laquelle est posée la contrainte de la 2° relation qui est référée dans le prédicat.

A chacun des 8 prédicats est associé deux procédures, la première peut être appelée au moment où l'utilisateur pose sa contrainte pour vérifier que les données de la multibase ne transgressent pas la contrainte (si c'est le cas, le système rejette la contrainte), la deuxième est appelée lors d'une opération de mise-à-jour sur la relation.

On va illustrer le mécanisme de vérification des contraintes à partir de l'exemple suivant :

MENUS (NUM-R, NUM-P, PRIX)

1	2	25
2	4	30
5	3	20

PLATS (NUM-P, NOM-P, TYPE

2	Quiche Lorraine	
3	Couscous Marocaine	
4	Paella Espagnolle	

Supposons que l'utilisateur exécute la commande :
POSER-CI C1 MENUS SELV PRIX < 300

Le système vérifie que les prix des MENUS sont inférieures à 300 et attache cette contrainte aux opérations de modification et d'insertion de tuples dans MENUS (d'après le tableau ci-dessus).

Si l'utilisateur veut insérer un tuple dans MENUS le système va vérifier que le prix du MENUS est inférieur à 300 F. Mais si l'utilisateur supprime un tuple de MENUS aucune procédure de vérification de l'intégrité n'est appelée.

Supposons maintenant que l'utilisateur exécute la commande : POSER-CI C2 MENUS REF PLATS NUM-P NUM-P le système va vérifier que les numéros des plats de MENUS se trouvent dans PLATS. Comme c'est le cas, il va attacher cette contrainte aux opérations d'insertion et de modification de tuples de MENUS ainsi qu'aux opérations de suppression et de modification de tuples de PLATS.

Maintenant, si l'utilisateur veut supprimer le plat numéro 2 de la relation PLATS, le système va refuser l'exécution de cette opération car la contrainte sur MENUS ne serait plus vérifiée.

Si l'utilisateur avait posé la contrainte C2 en spécifiant qu'elle doit être appliquée seulement en cas d'insertion le système aurait attaché la contrainte de référence C2 sur l'opération d'insertion de tuples dans MENUS et n'aurait pas fait des contrôles pour savoir si les numéros de plat de MENUS se trouvent dans PLATS et l'opération de suppression du plat de numéro 2 aurait pu avoir lieu

```
MB Loisir
   MB Restaurant
      BD R-Luxe
         R (numr, nomr, tel, arrond, st-metro)      restaurant
         Plats (nump, nomp, type)                   plats
         Menus (numr, nump, prix)                   menus
      END BD
      BD R-Mod
         R (numr, nomr, tel, arrond, st-metro)      restaurant
         P (num, nomp, type)                        plat
         Menus (numr, nump, prix)                   menu
      END BD
      END MB
      BD Cinéma
         C (numc, nomc, tel, arrond, st-metro,
            nbr-salle)                              cinéma
         F (numf, nomf, pays, genre, version)       film
         S (numc, numf)
      END BD
      BD Metro
         L (numl, noml)                             ligne
         S (numst, nomst, arrond)                   station
         L-S (numl, numst)                          ligne-station
      END BD
      END MB
```

CHAPITRE 5

IMPLANTATION

0 - INTRODUCTION

Le prototype opérationnel de SGMBR que nous avons réalisé sur VAX en CLU comporte un langage de définition et un langage de manipulation de données relationnelles. (cf annexe 1) Ce système permet de définir et de modifier des schémas de multi-base de données, d'interroger et de mettre-à-jour les données tout en assurant l'intégrité interne et la confidentialité des données et en réalisant la "data independence". Ce système est multi-utilisateurs mono-accès.

L'architecture générale de notre système est composée de trois modules principaux :

1 - un automate de dialogue,

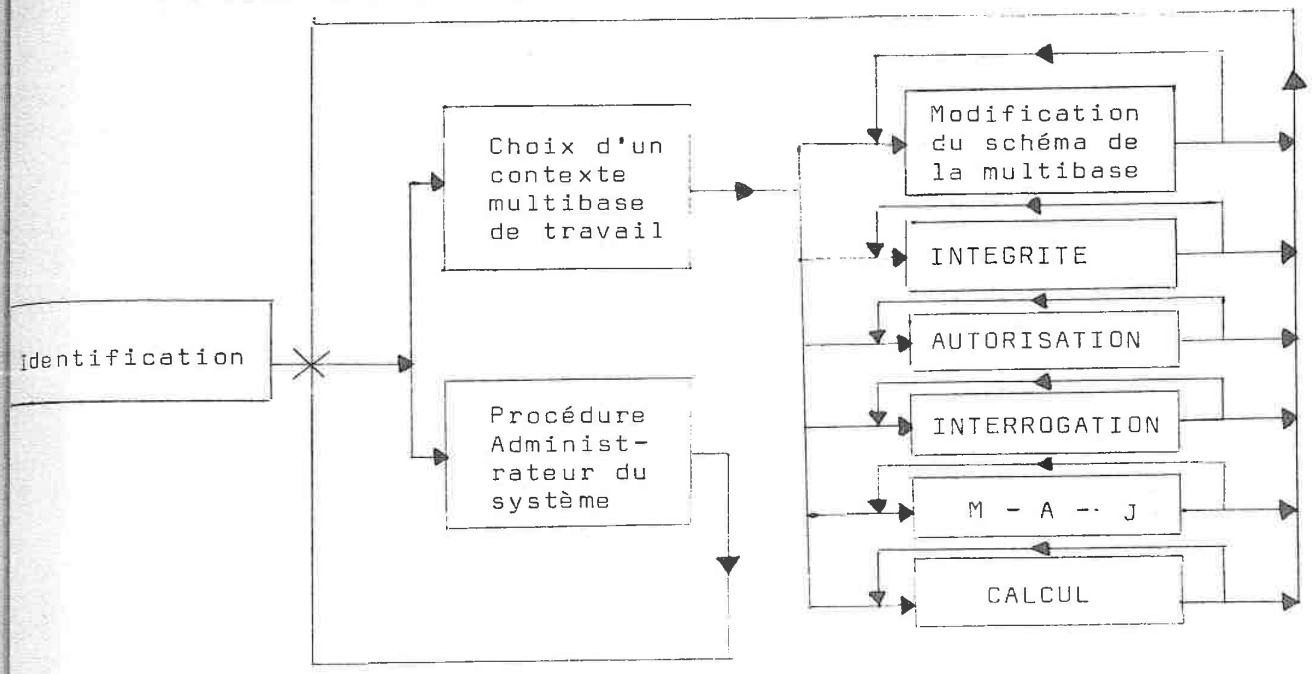
2 - un analyseur de requêtes,

3 - un module d'exécution de requêtes.

Nous allons illutrer les principales commandes du système à partir de l'exemple suivant, tiré du LIT (78).

I - AUTOMATE DE DIALOGUE

Ce module gère le dialogue avec les utilisateurs. Nous allons décrire les principales commandes disponibles pour chacun des états de l'automate. Une commande HELP donne la syntaxe des commandes de l'étape courante. La syntaxe de toutes les commandes se trouve en annexe 1.



1 - Identification

Ce module accomplit trois fonctions :

- Définition de l'identification : il génère un mot de passe lors de l'introduction d'un nouvel utilisateur.

- Modification de l'identification : un utilisateur peut changer son mot de passe par la commande CMP qui lui demande le nouveau mot de passe

- Autentification : quand un utilisateur se connecte au système il doit donner son nom et son mot-de-passe, s'il n'arrive pas à s'identifier correctement au bout de trois essais, le système le rejette.

Remarque : d'autres méthodes d'identifications, utilisant des caractéristiques biologiques des individus (empreintes digitales, voix...), des cartes magnétiques et/ou certains paramètres supplémentaires (type du terminal, lieu, heure de travail), sont utilisées dans les SGDB.

2 - Choix du contexte multibase de travail

Deux commandes sont à la disposition de l'utilisateur créer-mb pour créer une multibase vide et con-mb pour définir la multi-base de travail.

EX 1 : pour créer une multibase vide Loisir
CREER-MB Loisir

EX 2 : pour travailler sur la multibase Loisir qui existe déjà
CON-MB Loisir

Pour réduire son contexte de travail après la commande CON-MB l'utilisateur dispose de deux commandes :

MB	nom multibase	pour travailler sur une MB de la MB de connexion
BASE	nom base	pour travailler sur une base de la MB de connexion

Ex : si après la connexion à la MB Loisir l'utilisateur écrit MB Restaurant son contexte de travail sera la multibase restaurant.

3 - Modification du schéma de la multibase

Les commandes de ce niveau sont :

AJ-MB	ajoute une MB vide
SUP-MB	supprime une MB
AJ-BASE	ajoute une base vide
SUP-BASE	supprime une base
AJ-RB	ajoute une relation de base
AJ-RV	ajoute une relation virtuelle
REC-REL	permet de recevoir une relation (offerte par un utilisateur)
SUP-REL	supprime une relation

Ex 1 : ajouter la MB vide Restaurant à Loisir
AJ-MB Loisir Restaurant
Ex 2 : ajouter la base vide R-Luxe à Restaurant
AJ-BASE Restaurant R-Luxe

Nous allons expliciter les trois commandes : AJ-RB AJ-RV et REC-REL qui sont un peu plus compliquées que les autres :

a) pour ajouter une relation de base, il faut donner son schéma, sa clé et éventuellement la liste des attributs de la relation qui ne doivent pas admettre des valeurs indéterminées.
Ex : AJ-RB Cinema F numf : e, nomf :ch
pays : ch, version : ch,
type : ch
CLE numf
NONUL nomf

NB : les domaines des attributs sont : e pour entier, r pour réel, c pour caractère et ch pour chaine de caractères.
Les valeurs par défaut des attributs sont :
999 pour les entiers,
999 pour les réels
' ' pour les caractères
" " pour les chaines de caractères.

b) Pour ajouter une relation virtuelle, il faut donner l'expression relationnelle de la RV et la correspondance entre le schéma de la RV et celui de la relation résultant de la requête relationnelle.

Ex : pour ajouter à la base R-Luxe la relation Cin (nomcin, tel, arrond) :

```
AJ-RV R-Luxe cin PROJECT (cinéma.C, nomc,
                          tel, arrond)
                corresp nomcin : nomc,
                           tel : tel
                         arrond : arrond
```

c) La commande REC-REL permet de rajouter à une base une relation offerte par un utilisateur

```
Ex : REC-REL R-Luxe Hotel U2. Logem. Hotel.H
                          num : numh
                          nom : nomh
                          categ : categ,
                          tel : tel
```

permet de rajouter à la base R-Luxe, la relation virtuelle Hôtel (num, nom, categ, tel) qui est est une vue sur la relation H de la base Hôtel de la multibase Logem de l'utilisateur U2. Pour que cette requête puisse s'exécuter correctement, il faut que U2 ait au préalable offert la relation à celui qui a émis la requête.

4 - Intégrité

Plusieurs commandes sont disponibles pour lire. supprimer ainsi que pour définir des contraintes d'intégrité. On va donner la syntaxe ainsi que quelques exemples de ce dernier typ de commande :

a) Syntaxe :

POSER-CI nom-contrainte [(INSERTION) / (SUPPRESSION) / (MODIFICATION)] prédicat

prédicat : = nom-relation SELV nom-at op valeur
nom-rel DE-A nom-at val-min val-max
nom-rel ORDRE nom-at1 op nom-at2
nom-rel MOYENNE nom-at op valeur
nom-rel REF nom-rel nom-at nom-at
nom-rel OCCS nom-at op valeur
nom-rel OCCM nomat 1 nomat 2 op valeur
nom-rel N-A nomat op

b) Exemple

. Si on veut interdire l'insertion dans menus d'un plat qui n'existe pas dans la relation plat, il faut poser la contrainte de référence suivante :
POSER-CI C1 INSERTION Menus REF Plats nump nump

. Si le prix des menus ne doit pas diminuer
POSER-CI C2 Menus N-A prix >

. Si sur une ligne de métro, il ne doit pas y avoir plus de 30 stations
POSER-CI C3 L-S OCCM num-l num-s < 30

5 - Autorisation

Deux commandes permettent d'échanger des relations et des droits sur ces relations entre les différents utilisateurs :

DONNER liste de droits SUR nom-relation A nom utilisateur
REVOQUER liste de droits SUR nom-relation DE nom utilisateur

La liste des droits est composée de : l pour la lecture, i pour l'insertion, s pour la suppression et m pour la modification.

A chacun de ces droits, le donneur peut rajouter la lettre t qui autorise le receveur à transmettre ce droit à d'autre

Quand un utilisateur crée une relation de base, il a tous les droits sur cette relation.

Les relations virtuelles ne peuvent pas être mises-à-jour. Ainsi, les seuls droits qu'un utilisateur peut avoir sur une relation virtuelle sont l et lt.

Ex : l'utilisateur U2 peut émettre la commande suivante s'il détient le droit lt sur Log.Hotel.H
DONNER lt SUR Log.Hotel.H A U1

Pour plus de précision sur les mécanismes d'autorisation, se reporter au chapitre III.

6 - Interrogation

Le langage d'interrogation utilisé est l'algèbre relationnelle. Les opérations implantées sont : la projection, la sélection des valeurs d'un attribut par rapport à une valeur bien déterminée ou par rapport aux valeurs d'un autre attribut de la même relation, la sélection des tuples qui ont une valeur inconnue pour un certain attribut, la jointure avec égalité, l'union, l'intersection et la différence :

- PR (relation, liste attribut) : c'est l'opérateur PROJECT de l'algèbre relationnelle.

- SV (relation, attribut, opérateur, valeur de sélection)
  - sélectionne les tuples de arg1 dont la valeur de l'attribut arg 2 vérifie la relation exprimée par arg 3 avec la valeur arg 4. Les tuples dont la valeur de l'attribut sur lequel s'effectue la sélection est indéterminée ne figurera pas dans le résultat :
    Ex : SV (R-Mod.Rest, nbplaces $<$ 100)

- SA (relation, attribut, opérateur, attribut)
  - sélectionne les tuples dont les valeurs des attributs arg 2 et arg 4 vérifient la comparaison exprimée par arg 3. Si la valeur de l'un des attributs arg 2 ou arg 4 est indéterminée pour un tuple, ce tuple ne figure pas dans le résultat

- JN (relation, relation, attribut, attribut) : c'est l'opérateur de jointure de l'algèbre relationnelle.

- SI (relation, attribut) : retourne les tuples de arg 1 dont la valeur de l'attribut arg 2 est indéterminée.

- UN (relation, relation, liste de corresp)
  IN (relation, relation, liste de corresp)
  DF (relation, relation, liste de corresp),

Ce sont les opérations ensemblistes : union, intersection et différence. Une correspondance établit une équivalence entre un attribut de arg 1 et un attribut de arg 2. Le schéma de la relation résultat de l'exécution de l'une de ces trois opérations est égal à celui de arg 1.

Ex : UN (R-Mode.C, R-Luxe.C, numr : numr,
nomr : nomr, type : type)

7 - Mise-à-jour

Trois commandes permettent à l'utilisateur d'effectuer les opérations de mise-à-jour des relations : INSERER DANS, SUPPRIMER-DE, MODIFIER.

- Insertion d'un tuple : l'utilisateur doit fournir les valeurs des attributs qui font partie de la clé ainsi que ceux des attributs définis comme n'admettant pas des valeurs indéterminées :
INSERER-DANS nom rel liste (nom attribut : valeur)

Ex : INSERER-DANS R-Luxe.C (numr : 10, type : Française)

- Suppression d'un tuple : l'utilisateur doit fournir les valeurs des attributs du tuple à supprimer, qui font partie de la clé de la relation.
SUPPRIMER-DE nom relation liste (nom attribut : valeur)

Ex : SUPPRIMER-DE R-Luxe.C (numr : 8)

- Modification d'un tuple : l'utilisateur fournit les valeurs des attributs de la clé ainsi que les nouvelles valeurs des attributs à modifier.

MODIFIER nom rel liste (nom attribut : valeur)

Ex : MODIFIER F (numf : 10, type : western).

8 - Calcul :

Les opérations de calcul sont : MAX, MIN, MOYENNE, SOMME et CARD.

1) MAX (expression relationnelle, attribut) : délivre la valeur maximale de l'attribut arg 2 dans la relation résultat de l'exécution de arg1.

2) MIN (expression relationnelle, attribut) : délivre la valeur minimale de arg 2 dans arg 1,

3) SOMME (expression relationnelle, attribut) : délivre la somme des valeurs de arg 2 dans arg 1.

4) MOYENNE (expression relationnelle, attribut) : pour calculer la moyenne des valeurs de arg 2 dans arg 1.

Ces quatre opérations ignorent les tuples qui ont une valeur indéterminée pour arg 2.

5) CARD (expression relationnelle) : rend la cardinalité de l'expression relationnelle, c'est à dire le nombre de ces tuples.

9) Procédures réservées à l'administrateur du système

Plusieurs procédures sont accessibles uniquement par l'administrateur du système. Elles permettent de contrôler la gestion du système : ut-mp, ut-mb, ajout-ut, fic-rel-base, imp-aut.

1) ut-mp délivre les noms et mots-de-passe des utilisateurs du système.

2) ut-mb délivre, pour chaque utilisateur, les noms de ses multibases et les noms des fichiers où sont stockés les schémas des multibases.

3) fic-rel-base rend les noms des fichiers où sont stockées les tuples des relations de base.

4) Ajout-ut permet d'ajouter un utilisateur à la liste des personnes autorisées à utiliser le système. Cette procédure retourne le mot de passe que le système a donné à l'utilisateur.

5) Imp-aut : imprime le tableau qui contient les autorisations d'accès aux relations.

II - ANALYSEUR DE REQUETES

Les principales fonctions assurées par l'analyseur des requêtes sont : syntaxe, codification, représentation interne et contrôle d'autorisation.

1 - Une analyse syntaxique de la requête est effectuée.

2 - Codification

Les noms des entités qui figurent dans la requête sont transformés en noms interne au SGMB. Le nom interne d'une entité est constituée du nom de l'entité en question concaténé avec les noms des entités qui l'englobent. Par exemple, un utilisateur qui se connecte sur la multibase Loisir R-Mod peut poser une contrainte sur la relation Menus sans mentionner le nom de la base. L'analyseur doit transformer le nom Menus en Loisir · Restaurant. R-Mode. Menus. Si l'utilisateur était connecté à la multibase Loisir l'analyseur aurait signalé que le nom de la relation Menus est ambigü.

3 - Représentation interne de la requête sous forme d'arbre.

4 - Contrôle des autorisations :

Au cours de l'analyse suntaxique, nous avons une étape de contrôle des autorisations d'accès qui utilise le tableau d'autorisation du SGMB. Dans tous les cas, ce contrôle est fait avant l'exécution de la requete.

III - EXECUTION DE REQUETES

Plusieurs types et procédures ont été implantés pour assurer l'exécution des différentes catégories de requêtes interrogation, mis-à-jour, calcul, contrôle, définition...

Nous allons détailler deux de ces types, le type multibase et le type relation temporaire, qui présentent le plus d'intérêt et qui réalisent des fonctions importantes. Ensuite, on présentera trois méthodes qui peuvent être suivies pour l'exécution d'une requête d'interrogation de l'algèbre relationnelle.

1 - Type multibase (mb)

Les opérations du type multibase permettent d'implanter les différentes commandes de définition et de mise-à-jour et d'accès à une multibase :

créer : string ⟶ mb

- crée une multibase vide de nom arg 1
  aj-mb : mb x string ⟶ mb
- rajoute une multibase vide de nom arg 2 à arg 1
  aj-base : mb x string ⟶ mb
- rajoute une base vide de nom arg 2 à arg 1
  aj-rb : mb x string x string x rb ⟶ mb
- rajoute à la base de nom arg 2 de la mb arg 1 la relation de base arg 4 de nom arg 3
  aj-rv : mb x string x string x rv ⟶ mb
- rajoute une relation virtuelle
  ins-tuple : mb x texte x tuple x intégrité ⟶ mb
- insère le tuple arg 3 dans la relation de nom arg 2 de la multibase arg 1. L'insertion aura lieu si l'opération respecte les contraintes d'intégrité arg 4
  sup-tuple : mb x texte x tuple x intégrité ⟶ mb

- supprime le tuple arg 3
  rep-tuple : mb x texte x tuple x tuple x intégrité → mb
- remplace arg 3 par arg 4 de la relation de nom arg 2
  sup-mb : mb x string → mb
- supprime la multibase de nom arg 2 de arg 1
  sup-rel : mb x string → mb
- supprime la relation de nom arg 2 de arg 1
  sup-base : mb x string → mb
- supprime la base de nom arg 2 de arg 1
  +
  opérations de consultation.

2 - Type relation temporaire

Le type relation temporaire sert principalement lors de l'exécution des requêtes relationnelles. Il permet de mémoriser les résultats intermédiaires des exécutions et d'effectuer les opérations de l'algèbre relationnelle sur ces résultats :

type reltemp :
new : → reltemp
- crée une relation temporaire vide
  pschéma : reltemp x set (attribut) → reltemp
- donne un schéma à la relation temporaire
  put-tuple : reltemp x tuple → reltemp
  del-tuple. reltemp x tuple → reltemp
  rep-tuple : reltemp x tuple x tuple → reltemp
- ces trois opérations permettent l'insertion, la suppression et la modification destuples de la relation temporaire
  project : reltemp x set (nom attribut) → reltemp

- cette opération rend une relation temporaire résultat de la projection de arg 1 sur des attributs spécifiés dans arg 2.

  Selecta : reltemp x nom.attribut x opérateur x nom attribut ⟶ reltemp

- effectue l'opération SA décrite au niveau externe c'est à dire l'opération de sélection entre deux attributs.

  Selectv

  Selecti

  Union

  Différence : reltemp x reltemp x set (correspondance ⟶ reltemp

- Toutes les opérations de l'algèbre relationnelle figurent dans le type reltemp

  +

  opérations de consultation

3 - Exécution des expressions relationnelles

Trois principales méthodes peuvent être appliquées Pour exécuter une expression de l'algèbre relationnelle. Nous allons les décrire en s'appuyant sur la requête suivante :

PROJECT (JOIN (R-Luxe.Rest, SELECT (cinéma.C, nbsalles > 3),rue = rue),
nomrest, nomc, rue)

qui permet d'avoir pour chaque cinéma ayant plus de trois salles, le nom de ce cinéma, la rue où est située ce cinéma et les noms des restaurants de luxe situés dans cette rue.

a) 1ère Méthode : elle consiste à exécuter les opérateurs successivement (du plus interne vers le plus externe). Le résultat de l'application d'un opérateur sur une ou deux relations est une relation temporaire sur laquelle serait appliquée les autres opérateurs.

Dans l'exemple, l'opérateur SELECT sera exécuté en premier, le résultat de l'application de cet opérateur sur Cinéma.C va délivrer une relation temporaire formée des cinémas ayant plus de trois salles. Cette relation temporaire sera le 2ème opérande de l'opérateur JOIN qui sera exécuté après le SELECT. L'opérateur PROJECT s'appliquera sur la relation temporaire résultat de la jointure.

b) 2ème Méthode : son principe est d'exécuter toutes les opérations dès que c'est possible sur un tuple avant de passer au tuple suivant. L'algorithme procède ainsi. Il prend un tuple t1 de R-Luxe et cherche les tuples t2 de la relation cinéma.C qui vérifient le critère de sélection (nb.salles de t2 = 3) et le critère de jointure avec t1 (rue de t1 = rue de t2), à chaque fois qu'il trouve un tel tuple t2, il effectue la jointure entre t1 et t2 et ensuite la projection avant de passer au tuple suivant. Quand il n'y a plus de t2, il passe à un autre tuple t1 de R-Luxe et recommence le même processus.

Une première optimisation de cette méthode consiste à sauvegarder les tuples de Cinéma.C qui vérifient le critère de sélection dans une relation temporaire après qu'ils soient sélectionnés lors du traitement du premier tuple de R-Luxe.

La 1ère méthode a le mérite d'être simple à mettre en oeuvre et permet d'éliminer les duplications. Par contre, elle offre moins de possibilité d'exécution en parallèle et nécessite de la place pour stocker les résultats intermédiaires.

La 2ème solution nécessite moins de place que la première pour les résultats intermédiaires. Elle permet l'exécution en parallèle des opérations lorsqu'un processus effectue la projection, un autre processus peut chercher le tuple suivant

vérifiant le critère de jointure dans l'exemple précédent. Par contre, elle ne permet pas d'éliminer les duplications. Cette méthode a été appliquée dans le projet TYP-R (TOU 83).

c) 3ème méthode : cette méthode se situe entre les deux précédentes SMI (75). Elle consiste à lancer l'exécution d'un opérateur dès qu'un nombre "suffisant" de tuples est disponible. Certains opérateurs peuvent s'exécuter dès qu'un tuple est disponible, d'autres nécessitent une relation entière. Cette solution permet d'éliminer les duplications, de faire des exécutions en parallèle et elle nécessite moins de place que la 1ère.

D'après cette présentation, il ressort que la 3ème méthode est la plus optimale. Néanmoins, nous avons choisi la 1ère solution car elle est simple à mettre en oeuvre et parce que les problèmes d'optimisation de l'exécution des expressions relationnelles font l'objet d'une étude approndie réalisée par d'autres membres de l'équipe : KTM (84), GEN (83).

CONCLUSION

CONCLUSION

En guise de conclusion, examinons divers points qui pourraient être améliorés et poursuivis :

1 - Optimisation DEL (82) MIL (75) GEN (83)

Pour réduire le coût total des opérations (E/S + unité centrale + (transfert)), tout en minimisant la place mémoire occupée deux catégories d'optimisation peuvent être effectuées :

a) Optimisation des opérations individuelles

Différents algorithmes peuvent être implantés pour effectuer chacune des opérations de l'algèbre relationnelle. Le choix d'un algorithme particulier pour une exécution dépend de plusieurs facteurs : cardinalité des relations, si les tuples sont triés selon certains domaines ou non, existence d'index.

b) Optimisation globale des requêtes

- Effectuer les opérations de projection et de sélection le plus tôt possible pour réduire le volume d'informations à traiter.

- Regrouper certaines opérations : plusieurs sélections en une seule, une projection et une jointure peuvent être effectuer en même temps.

- etc...

N.B. : L'utilisation de la semantique des contraintes d'intégrité permet dans certains cas, d'effectuer des optimisations substantielles : par exemple, si on a la contrainte que le nombre des étudiants inscrits dans un cours ne peut pas dépasser 35 et que lors de la recherche des noms des étudiants inscrits dans un cours donné, on trouve 35 étudiants, on peut arrêter la recherche car on est sûr qu'il n'y en a plus.

2 - Extension du système

Certaines extensions des fonctions offertes par notre système peuvent être effectuées à court terme, d'autres à moyen ou à plus long terme.

a) Extensions à court terme :

On peut introduire à cout terme :

- La mise -à-jour des relations virtuelles : se limiter dans une première étape au cas où la vue est définie par une (ou des) sélection(s) sur une relation de base,

- Les contraintes sémantiques : notamment les contraintes d'équivalence. Pour cela, on peut s'inspirer des travaux qui sont effectués à l'INRIA par KAB(82).

- La notion de transaction,

- La notion de "trigger" : cette notion permet de spécifier des procédures qui doivent être exécutées à la suite de certaines opérations.

b) Extensions à moyen et à plus long terme

Dans notre système, nous nous sommes intéressés à la description d'une multibase et à l'expression des contraintes. Il est clair que pour réaliser un véritable SGMB complet, il faudrait également intégrer d'autres aspects. Un travail considérable sur ce point reste à faire :

- Concurrence d'accès : ces problèmes ont été largement étudiés dans la litérature scientifique.

- Répartition physique des données. Plusieurs travaux en France ont été effectués dans ce domaine notamment le projet Polyphème à Grenoble et le projet pilote SIRIUS de l'INRIA.

- Aspect temporel : pour conserver l'historique d'un système d'information afin de pouvoir restituer des séquences du passé, de renseigner sur le fonctionnement passé ou présent de l'organisation ainsi que les évolutions futures possible (BAN (79)). Pour cela, on peut profiter de l'expérience acquise par l'équipe REMORA de NANCY FOU (82) KEB (83).

- Réalisation d'un pré-processeur. Afin de pouvoir intercaler dans des programmes écrits en CLU des requêtes adressées au SGMB. Actuellement, une thèse de 3ème cycle KIM (83) a pour objectif de réaliser un pré-processeur pour le langage temps réel LTR.

- Bases de données images : c'est une nouvelle dimension qui se fait jour dans le domaine des BD. A Nancy, des travaux sont effectués par CRE [ ] sur cet aspect

- Spécification : pour aider l'utilisateur à concevoir ses multibases. CHA (82). D'autre part, un outil qui aide l'utilisateur à formuler ses requêtes sur un schéma abstrait d'une BD est en cours de réalisation BOU (83).

- Intelligence artificielle.

BIBLIOGRAPHIE

(ADI 79) ADIBA M. et ANDRADE J.M.
PROJET Polypheme : Description de la maquette et de l'interface **repartie**
Note technique n° NT 39 décembre 1979
IMAG GRENOBLE

(ADI) ADIBA M.
Les bases de données relationnelles : centralisation ou répartition
Laboratoire IMAG - B.P. 53 X - 38041 GRENOBLE CEDEX

(AHO 79) AHO A.V. , BEERI L et ULLMAN J.D.
The theory of joins in Relational Databases
ACM Transactions on Database Systems
vol 4 n° 3 sept. 79 pages 297 - 314

(AHU 79) AHUJA V.
Algorithm to Check Network states for Deadlock
IBM J. RES. DEVELOP Vol 23 n° 1 JANUARY 1979

(ALL) ALLMAN E. STONEBRAKER M. et HELD G.
Embedding a relational Data Sublanguage in a General Purpose Programming Languange
SIGPLAN Not 11 mars 76

(AZR 82) AZROU F.
Confidentialité et Autorisation d'accès dans les systèmes de bases de données
R.R. n° 318 sept. 1982
Université scientifique et médicale de GRENOBLE
MIAGE 38409 GRENOBLE St Martin d'Hères

(BAB 80) BABAD Y.M. et HOFFER J.A.
Data Element Security and its effects on file Segmentation
IEEE Transaction on software engineering
Vol. SE6 n° 5 september 80

(BAL 79) BALZER R. et GOLDMAN N.
Principles of good software specification and their implications for specification languages
IEEE 1979

(BAN 81) BANCILLON F.et SPYRATOS N.
DATA BASE MAPPINGS
PART I : Theory
PART II : Applications
Rapport de recherche n° 62 et 63
INRIA
Domaine de Voluceau Rocquencourt
BP 105
78153 LE CHESNAY CEDEX FRANCE

(BAN 79) BANON D.
PROJET REMORA
Un outil pour la gestion des systèmes d'information
Thèse de 3ème cycle en informatique
université de NANCY 1

(BEC 80) A Generalized Implementation Method for Relational
Data Sublanguages
IEEE Transactions on Softwareengineering
vol SE-6 n° 2 March 1980

(BER 82) BERNSTEIN P.A. et GOODMAN N.
A sophisticated introduction to distributed
data base concurrency control
Proceedings of the 8 th international conference
on very large Data Bases
Mexico city september 1982

(BER 79) BERZINS V.A.
Abstract Model specifications for data abstractions
MIT/LCS/TR-223 July 1979

(BLA 77) BLASGEN M.W. et ESWARAN K.P.
Storage and access in relational data bases
IBM System Journal n° 4 1977

(BOY 75) BOYCE R.F. CHAMBERTIN D. FRANK KING IV W. HAMMER M.M.
Specifying Queries as Relational
Expressions : The SQUARE Data Sublanguage
CACM November 1975
Volume 18 Number 11

(BRA) BRACCHI G. FURTADO A. et PELAGATTI G.
Constraint specification inevolutionary Data
Base Design

(BRO 81) BRODIE M.
On Modelling Behavionral Semantics of Databases
IEEE 1981

(BRO 78) BRODIE M,L. et SCHMIDT J.
What is the use of Abstract Data Types in Data Bases
IEEE 1978

(BUR 81) BURTON F.W. et LINGS B.J.
Abstract data types, subtypes and data independancy
Computer Journal vol. 24 n° 4 1981

(BJO 80) BJORNER P.
Formalization of Data Base Models
LNCS 86 1980

BOU (83) BOUDJLIDA N.
Contribution à l'élaboration de systèmes déductifs
utilisant les types abstraits de données.
Thèse de docteur-ingénieur - Université de Nancy
1983

(CAR 82) CARLSON C.R. ARORA A.K. et CARLSON M.
The Application of functional Dependency
Theory to Relationnel Databases
The Computer Journal vol 25 n° 1 1982

(CHA 75) CHAMBERLIN D.M. GRAY J.M. et TRAIGER C.L.
Views, anthorization, and locking in a relational
data base system
National Computer Conference 1975

(CHA 76) CHAMBERLIN A.
Relational Data Base Management Systems
Computing Surveys vol 8 n° 1 March 1976

(CHE 81) CHEYEL M.H. GASSER M. HUFF G. et MILLEN J.K.
Veriying Security
Computing Surveys vol 13 n° 3 sept. 1981

(COD 74) CODD E.F.
Recent investigations in relational Data Base Systems
Information Processing . North-Holland Publishing compan
PP. 1017 - 1021
(186 Than TOUNSI)

(CON 72) CONWAY R.W. MAXWELL W.L. et MORGAN H.L.
On the implementation of Security Measures
in information Systems
LACM April 1972 Volume 15 number 4

(CUN) CUNIN P.Y
PASCAL Relationnel
CRIN
Chateau du Montet 54500 VANDOEUVRE

(COD 70) CODD E.F.
"A Relational Model of data for large shared data
banks"
CACM, vol 13, n° 6, 1970, pp. 377-387

(COD 71) CODD E.F.
"Further normalization of the database relational
model" Database systems, Prentice Hall, Englwood
Cliffs, N.J. 1971, pp 56-98

(COD 71a) CODD E.F.
"A database sublanguage founded on the relational
calculus". Proc. ACM SIGFIDET, Workshop on data
description access and control. San Diego 1971.

(COD 72) CODD E.F.
"Relational completness of database sublanguages"
Database systems, Curent computer science symposia,
Vol 6, Prentice Hall, 1972, pp 65-93.

(COD 74) CODD E.F.
"Recent investigations in relational database systems" Information processing, North Holland Publishing company, 1974, pp 1017-1021

(COD 79) CODD E.F.
"Extending the database relational model to capture more meaning" ACM TODS, vol 4, n° 4, Dec. 1979, pp 397-43

(COD 81) CODD E.F.
"Data model in database management" ACM SIGMOD RECORD, Vol 11, n° 2, Fev. 1981, pp 112-114

(COD 82) CODD E.F.
"Relational database ; a practical foundation of productivity" CACM, vol 25, n° 2 Fev. 1982 pp. 109-117.

(CHA 82) CHABRIER J.J.
Spécification et construction de systèmes orientés bases de données
Techniques et langages basés sur la conception de type abstrait
Thèse d'état université de NANCY 1 1982

(DAR 79) DARGENT L.
Mise-à-jour de données réparties : contrôle de la concurrence d'accès
Thèse de 3ème cycle
Université de NANCY I

(DAT 81) DATE C.J.
A formal Definition of the Relational Model
ACM SIGMOD 81

(DAV 78) DAVIDA G.
Security and Privacy
IEEE 78

(DAY 82) DAYAL U. et BERNSTEIN P.A.
On the Correct Translation of Update operations on Relational Views
ACM Transactions on Data Base Systems
vol 8 n° 3 sept 1982

(DAC 80) DACITRE P.
POLYPHEME Project :
The DEM distributed execution monitor

(DEL 82) DELOBEL C. ADIBA M.
: Bases de données et systemes relationnels
édition Dunod

(DEL 80) DELOBEL C.
Etudes formelles dans les Bases de Données
19 au 21 nov. 1980 INRIA

(DER) DEREMER F.
Programming in the large versus
Programming in the small

(DER 81) DERNIAME J.C. CHABRIER J.J. GODART C. TOUNSI N.
et AIT HADOU A.
Un système de gestion multibase
rapport de recherche 81-R-53
université de NANCY I

(DER 70) DERNIAME J.C. et VIAULT D.
Sharing of Descriptions and objects in a modular
Programming System

(DER 74) DERNIAME J.C.
Le projet CIVA un système de programmation modulaire
Thèse d'état - Université de NANCY 1 - Juin 1974

(DER 79) DERNIAME J.C. et FINANCE J.P.
Types abstraits de données
spécification, utilisation et réalisation
Ecole d'été de l'AFCET Monastir

(DOS 82) DOSCH W. MOSCARI G. et WIRSING M.
On the algebraic specification of databases
8th International Conference on VLDB
Mexico City sept. 1982

(DAT 81) DATE C.J.
"Referential integrity" Proc. of the 7th intern,
conf. on VLDB, sept. 81, Cannes FRANCE

(DAT 81a)DATE C.J.
"An Introduction to database systems" 3rd édition
1981, ADDISON-WESLEY Pub. Comp.

(DAT 83) DATE C.J.
"An introduction to data base systems"
Vol II
ADDISON - WELSEY 1983

(EHR) EHRIG H. KREOWSKI H.J. THATCHER J. WAGNER E.
et WRIGHT J.
Parameterized Data Types in Algebraic
Specification Languages

(ESW 76) ESWARAN K.P. GRAY J.N. LORIC R.A. et TRUGER I.L.
The Notions of Consistency and Predicate Locks
in a Data Base System
CACM November 76 Volume 19 n° 11

(FER 81) FERNANDEZ F.
Micro Memoire Relationnelle
RR n° 233
IMAG GRENOBLE

(FLO 81) FLORY A.
Bases de données : conception et réalisation
Edition ECONOMICA

(FRY 76) FRY J.M. et SIBLEY E.H.
Evolution of Data Base Management Systems
computing Surveys vol 8 n° 1 1976

(FOU 82) FOUCAULT O.
Modèle et outil pour la conception des systèmes d'information dans les organisations
Projet REMORA
Doctorat d'état - Université de NANCY 1 - Juin 1982

(GAR 80) GARDARIN G. et VALDURIEZ P.
Jointures de relations dans une machine
Bases de données Multiprocesseur
Institut de Programmation université de Paris VI
SIRIUS - INRIA - BP 105 - 78150 LE CHESNAY

(GEH 82) GEHANI N.
Specifications Formal and informal
A case study
Software - Practice and Experience Vol 12 1982

(GOD 81) GODART C.
SGMB : un système de gestion multibase
Thèse 3 : cycle université de NANCY I 1981

(GOG 83) GOGOLLA M. DROSTEN K. LIPECK O. et EHRICH
Algebraic and operational semantics
exceptions and Errors
6th TCS. LNCS 145 1983

(GOG 78) GOGUEN J.
Semantics and Theory of Computation
Report 14 order sorted Algebras. Exceptions and error sorts, coercions and overloaded operators

(GRA 81) GRAY J.
The Transaction Concept : Virtues and Limitations
IEEE 1981

(GRI 77) GRIEES D. GEHANI N.
Some Ideas on Data Types in High - level Languages
CACM June 1977 volume 20 number 6

(GRI 76) GRIFFITHS P. et WADE B.W.
An authorization Mechanism for a relational
Data Base System
ACM TODS Vol 1 n° 3 september 1976

(GUD 80) The Design of a Cryptography Based Secure Fil System
IEEE Transactions on software engineering
Vol SE-6 n° 5 September 1980

(GUT 77) GUTTAG J.
Abstract Data Types and the Development of
Data structures
CACM June 1977 Volume 20 n° 6

(GEN 83) GEND P.
Analyse syntaxique et optimisation des expressions
relationnelles de R-LTR
AESS - Rapport de stage - Université de NANCY 1 1983

(FRI 82) GRISON T.
Génération des textes ATM d'implantation de relations
en TYP-R"
Rapport DEA, université Nancy I, Septembre 1982

(HAI 74) HAINAULT J.L. et LECHARLIER B.
An extensible semantic model of Data Base
and its Data Language
information processing 74
North Holland publishing Company (1974)

(HAR 76) HARRISON M.A. RUZZO W.L. et UNIVERSITY J.
Protection in operating Systems
CACM August 1976 Volume 19 Number 8

(HAS) HASKIN R.L. et LORIE A.A.
On extending the function of a relational
Data Base System
RJ 3182 (38988) 11/11/81
IBM Research Laboratory
San Jose Californie 95193

(HEL 75) HELD G.D. STONEBRAKER M.R. et WONG E.
INGRES - A relational data base system

(HEN 80) HENRY P.
La connexion dans le projet TYP
Thèse du 3ème cycle université de NANCY 1980

(HOR) HORNING J.J.
Some Desirable Properties of Data Abstraction facilities
Computer Systems Rescorch Group
University os Toronto
Toronto Canada M5S 1A4

(LAC 80) LACROIX M. et PIROTTE A.
Associating Types with Domains of Relational Data Bases
1980 ACM

(LEA 79) LEAVENWORTH B.
The use of data abstraction in program design
computer science Departement
Thomas J. Watson Research center
P.O. BOX 238
yorktown Heights N.Y. 10538

(LEA 81) LEAVENWORTH B.
Software technology Project
Technical Memo Number 19
Computer science Department
IBM Corporation
T.J. Watson Research center
Yorktown Heights NY 10598

(LEA 80) LEAVENWORTH B.
A Data Abstraction Approach to Data Base Modelling
1980 ACM

(LED 77) LEDGARD H.F. et TAYLOR R.W.
Two views of Data Abstraction
CACM June 1977 volume 20 n° 6

(LIE 82) LIEN Y.E.
On the Equivalence of Data Base Models
Journal of the Association for Computing Machinery
vol 29 n° 2 April 1982 pp. 333 - 362

(LIT 81) LITWIN
Logical Design of Distributed Data Bases
MOD-I-043
SIRIUS INRIA

(LIT 81) LITWIN W.
Présentation du projet B A BA
GAL-I-042 INRIA

(LIS 77) LISKOV B. SIGYDER A. ATCKINSON R. et SCHAFFERT G.
Abstraction Mechanisms in CLU
CACM August 1977 volume 20 number 8

(LIS 74) LISKOV B. et ZILLES S.
Programming with Abstract Data Types
SIGPLAN NOTICE Vol 9 n° 4 April 1974

(LIS 81) LISKOV B.
CLU reference manual
LNCS 145 1981

(KAB 82) KABBAJ
MUQUAPOL : un SGMB
Communicaciones SEIR 2
Santiago de Compostela 1982

(KAR 80) Le modèle d'accès de la machine Bases de données SABRE
Communication présentée aux journées Machines Bases de Données 10-12 sept. 1980 SOPHIA ANTIPOLLIS

(KEL) KELLER P.M.
Updates to Relational Databases Through views involving Joins
RJ 3282 (39738) 10/27/81
IBM Research Laboratory
SAN JOSE, CA 95 193

(KEN 81) KENT W.
A simple guide to five normal forms in relational Database Theory
Technical Report TR 03.151
Santa Teresa Laboratory
SAN JOSE CALIFORNIA

(KIM 79) KIM W.
Relational Database Systems
Computing Surveys vol 11. n° 3 sept. 79

(KLU 82) KLUG A. et PRICE R.
Determining view dependencies using tableaux
ACM TODS Vol 7 n° 3 sept. 82 pages 361-380

(KEB 83) KEBAILI M.
Une contribution à l'étude du problème de l'évolution de structures dans les systèmes d'informations historiques
Thèse de 3ème cycle - Université de NANCY 1, à paraitre

(KIM 83) KIM M.J. and DERNIAME J.C.
Data type of Relation in a High-Level Programming language : LTR
à paraitre : rapport interne CRIN - NANCY

(LOY) LOYER M.
Les mécanismes de programmation modulaire et de compilation séparée dans les langages de programmation
INRIA

(LOZ) LOZINSKI E,L.
Construction of Relations in Relational Databases

(LUC) LUCAS P.
Practical Programs and Formal Semantics
IBM San Jose Research Laboratory

(LES 79) LESCANNES P.
"Etude algébrique et relationnelle des types abstraits et de leurs représentations"
Thèse d'état INPL

(MAI 81) MAIBUM T.S.E.
Data Base Instances, Abstract. Data Types and Data Base Specification
Univ. of waterloo, computer science dept
CANADA, N2L 3G3, 1983

(MAS) MASRI EL R. et WIEDERHOLD G.
Properties of Relationships and their représentation

(MAR) MARYANSKI F.J. et ROUSA S.
Définition of Data base Transactions by the casual user

(MOH 79) MOHAN C.
An overview of recent Data Base Research
Data Base Fall 1979

(MIC 76) MICHAELS A.S. MITTMAM B. et CARLSON C.R.
A comparison of the relational and Codasyl Approoches to Data Base Management
computing Surveys vol. 8 n° 1 Mars 1976

(MIN 76) MINSKI N. et UNIVERSITY R.
Intentional Resolution of Privacy Protection in Data Base Systems
CACM March 1976 Volume 19 Number 3

(MIR) MIRANDA S,M.
Aspects of Data Security in General Purpose Data Base Management Systems

(MIR 78) MIRANDA S.M.
La sécurité des données dans les SGBD Relationnels
Colloque Modèles relationnel - Institut de Programmation 3/78

(MUS 80) MUSSER R. D.
Abstract Data Type Specification in the Affirm System
Through formal specification
LNCS n° 107 pages 169 - 208

(MEN) MENON M.J. and HSIAO D.K.
The access control mechanism of a data-base computer

(NEU) NEUHOLD E.J. et ANOF TH.
Building Data Base Management Systems
Through formal specification
LNCS n° 107 Pages 169 - 208

(PAI 78) PAIR C.
La programmation de l'énonce au programme
Centre de recherche en informatique de NANCY
78 P 061

(PAO 80) PAOLINI P.
Abstract Data Types and Data Bases
ACM 1980

(PAO 77) PAOLINI P. and PELAGATTI G.
Formal Definition of Mappings in a Data Base
SIGMOD 77 (TORONTO)

(PAR 72) PARMAS D.L.
A technique for software module specification
with examples
CACM May 1972 volume 15 Number 5

(PIR 81) PIROTTE A.
A precise Definition of Basic Relational Notions
and of the relational Algebra
ACM SIGMOD 1981

(PRE 78) PRENNER C.J. et ROWRE L.A,
Programming languages for relational data base
systems
National computer conforence 1978

(PRO 82) PROCH K.
ORSEC : un outil de Recherche de Spécifications
équivalentes par comparaison d'exemples
Thèse 3ème cycle - Université de NANCY 1

(REI 75) REISNER P. BOYCE R.F. et CHAMBERLIN D.D.
Human factors evaluation of two database query languages - square and sequel
National Computer Conference 1975

(RIS 77) RISSANEN J.
Independent component of relations
ACM Transactions on Data Base Systems
Vol 2 n° 4 December 1977 Pages 317 - 325

(ROT 75) ROTHNIE J.B.
Evaluating inter-entry retrieval expressions in a relational data base Management systems

(ROW 80) ROWE A. L.
Issues in the Design of Data Base Programming Languages
1980 ACM

(ROW 79) ROWE A, L.
Data Abstraction, views and updates in RIGEL
ACM 1979

(REM 81) REMY J.L.
"Etude des systèmes de récriture conditionnels et applications aux types abstraits algébriques
Thèse d'état INPL 1982

(SCH) SCHWERMANN P. , SCHIFFNER G. et WEBER H.
Abstraction Capabilities and Invariant properties
Modelling within the entity-relationship Approoch

(SCH 80) SCHMIDT J.
Data Abstraction Tools : Désign, Specification
and Application
1980 ACM

(SCH 77) SCHMIDT J.W.
Some High level language construct for Data of
type Relation
ACM TODS Vol 2 sept 1977 pp. 247-263

(SCH 81) SCHWARTZ R., SCHMIT T MELLIAS P.M.
The finalization opération for abstract types
IEEE 1981

(SEL 80) SELINGER P.G. et ADIBA M.
Access Path selection in distributed database
Management Systems
RJ 2883 (36439) 8/3/80

((SHA 78) SHARMAN G.C.H. et NIWINTERBOTTON
The Data Dictionary Facilitos of NDB

(SHA 80) SHAW M.
Abstraction, Data Types, and Models for software
ACM 1980

(SHO 81) SHORT K.W.
Protection in data type abstractions using constraints
on data values
The Computer Journal vol 24 n° 2 1981

(SHO 81) SHORT K.W.
Protection in data type abstractions using constraints
on data values
Computer Journal vol 24 n° 2 1981

(SMI 75) SMITH J.M. et TANG CHANG P.Y.
Optimizing the Performance of a Relational
Algebra Database Interface
CACM october 75 volume 18 n° 10

(SMI 77) SMITH J.M. et SMITH D.C.P.
Aggregation and Generalization
ACM Iransactions on Data Base Systems
Vol 2 n° 2 June 1977 pp. 103 - 133

(SMI 77) SMITH J.M, et SMITH D.C.P.
Data Base Abstractions : Aggregation
CACM June 77 volume 20 n° 6

(SOK 80) SOK S.C.
Evolutivité du Logiciel
doctorat de 3ème cycle en Informatique
Université de NANCY 1 1980

(STA 78) STANALGY Y.W.S.U. et REYNOLDS M.J.
Conversion of high-level sublanguage queries
to account for data base changes
AFIP vol 47 1978

(STO 76) STONEBRAKER M.
The Design and Implementation of INGRES
ACM TODS Vol 1 n° 3 sept 1976

(TAY 76) TAYLOR W.R.
Codasyl Data-Base Management Systems
Computing Surveys vol 8 n° 1 Moret 1976

(TOM) TOMPA W. F.
A Practical Example of the Specification of
Abstract Data Types
acta informatica 1980

(TOU 82) TOUNSI N. , DERNIAME J.C. ,CHABRIER J.J. et LITWIN W.
TYP-R un systéme de bases de données relationnel
réalisé à l'aide de types abstraits de données.

(TOU 83) TOUNSI N.
TYP-R : réalisation en TYP d'un système multibase
relationnel
Thèse de 3ème cycle
Université de NANCY 1

(TOU 79) TOUNSI N. et CHABRIER J.J.
Traitements itératifs et itérateurs
centre de recherche en inf. de NANCY

(TSI 76) TSICHRITZIS D.C.
Hierarchical Data-Base Management : A survey
computing Survey vol. 8 n° 1 Moret 76

(VAL 80) VALDURIEZ P.
Algorithmes de jointure de relations
Institut de programmation - Université de PARIS VI
SIRIUS - INRIA - BP 105 - 78150 LE CHESNAY

(WAS 79) WASSERMAN A.I.
The Data Management Facilities of PLAIN
ACM 1979

(WEB 78) WEBER H.
A sofware Engineering view of Data Base Systems
IEEE

(WIL 81) WILLIAMS R.
R* : AN onerview of the Architecture
Research Report RJ 3325 (40082) 12/2/81
IBM Research Laboratory
SAN JOSE CALIFORNIA 95193

(WIL 81) WILMS P.F. et LINDSAY B.G.
A Data Base Authorization Mechanism suporting
Individual and Group Authorization
Research Report RJ 3137 (38514)
IBM Research Laboratory
San JOSE CALIFORNIA 95193

(WIL 81) WILSON M.L.
A requirements and Design aidefor Relational
Data Bases IEEE 1981

(WUL 76) WULF W.A., RALPH L. LONDON, and MARY SHAW
An Introduction to the Construction and Verification
of Alphard Programs
IEEE transactions on software engineering
vol SE-2, n° 4. December 76 pp. 253-265

(WOO 81) WOOD C., FERNANDEZ E.B. et SUMMERS R.C.
Database sécurity and Integrity addison Welsey

(ZEL) ZELKOWITZ M.V.
Perspectives on Software Engineering
ACM COMPUTING SURVEYS Vol 10 n° 2 June 1978
(ZOM 80) ZILLES S.N.
Types Algebras and Modelling
RJ 3022 (37387) 12/23/80
Research Report

(ZIL) ZILLES S.
A look at algebraic Specifications
IBM San Jose Research Laboratory
U.S.A.

(ZUR 81) ZURFLUH G.
Le projet PLEXUS
Un système d'Approvisionnement Reparti
Journées de présentation des résultats du projet
pilote SIRIUS
26-27 november 1981

ANNEXE 1 - SYNTAXE DU LANGAGE

```
: start_up()
login:
sbd
mot_de_passe:
007

categorie des operations a effectuer :

help:

   connexion a une multibase      --->  con_mb
   creation d une multibase       --->  creer_mb
   supprimer une multibase        --->  sup_mb
   changer mot_de_passe           --->  cmp
   procedures admin. de la base   --->  fac_sbd
   arret de la session            --->  fin

commande:
 ac_sbd:
requete
help:
   imprimer autorisation                    --->  imp_aut
   nom_util avec mot_passe                  --->  ut_mp
   nom_util avec nom_mb avec nom_fic        --->  ut_mb_fic
   ajouter un utilisateur                   --->  ajout_ut
   noms fichiers relations de base          --->  fic_rb

requete
fin:

categorie des operations a effectuer :

con_mb:

nom multibase:
help:

les noms de vos multibases sont:
loisir
loisir

pour reduire le contexte de travail
    mb   nom_multibase
    base nom_base
    fin  si vous ne voulez plus reduire votre contexte

fin:

commande:
help:

   imprimer le schema d une multibase  --->  imp_scheme
   autoriser des droits d acces        --->  autoriser
   modifier le schema d une multibase  --->  mod_scheme
   interroger:requete relationnelle    --->  interroger
   interroger:proc de calcul           --->  calcul
   mise_a_jour d une relation          --->  maj
   imprimer schema de la relation      --->  sch_rel
   arret                               --->  fin
   ajouter un(des) attribut(s) a relat,--->  aj_attr
   contrainte d integrite              --->  contrainte

commande:
```

```
imp_schema;
votre requete doit etre de la forme
   mb nom multibase
   base nom base
   rel nom relation
  n.b.:si vous voulez imprimer la multibase ou la base  sur laquelle vous etes connect
   il faut taper mb ou base main

requete;
base metro;
   bd  metro
      l(num_l:e:prov:ch:dest:ch)
      s(nom_s:ch:corresp:e)
      ls(num_l:e:nom_s:ch)

commande;
mod_schema;

requete;
help;
  les commandes de modification du schema sont
      ajouter multibase ---> aj_mb nom mbase nom mbase
      ajouter base      ---> aj_base  nom mb    nom base
      ajouter rel_base  ---> aj_rb    nom base  nom relation
      ajouter rel_virt  ---> aj_rv    nom base  nom relation
      supprimer mbase   ---> sup_mb   nom mb
      supprimer base    ---> sup_base nom base
      supprimer rel_base ---> sup_rel  nom relation
      recevoir relation  ---> rec_rel nom_base nom_relation


requete;
fin;

commande;
interroger;

requete;
help;
le syntaxe des operateurs relationnelles est la suivante;
   pr(nom_relation:liste nom_attribut)
   sa(nom_relation:nom_attribut operateur
   sv(nom_relation:nom_attribut operateur valeur)
   si(nom_relation:nom_attribut)
   jn(nom_relation  nom_relation:nom_attribut  nom_attribut)
   un(nom_relation  nom_relation:correspondance)
   in(nom_relation  nom_relation:correspondance)
   df(nom_relation  nom_relation:correspondance)
   range nom_variable nom_relation
   correspondance;:=liste de nom_attribut;nom_attribut
ibut


requete;
fin;

commande;
maj;

requete;
help;
le syntaxe des requetes de mise_a_jour est :
   inserer_dans nom_relation liste nom_attribut;valeur
   supprimer_de nom_relation liste nom_attribut;valeur
   modifier     nom_relation liste nom_attribut;valeur
```

```
requete!
fin!

commande!
contrainte!


requete!
help!

   pour mettre des contraintes d integrite    --->  mettre_ci    (ou m)
   pour supprimer des contr. d integrite      --->  sup_ci       (ou s)
   pour lire des contraintes d integrite      --->  lire_ci      (ou l)
   pour arreter                               --->  fin



requete!
m!

requete !

help!
les requetes doivent etre de la forme suivante !
   nom_contrainte (insertion ou suppression ou modification) nom_relation
    selv nom_attribut operateur valeur  (ex!selv age>16)
    de_a nom_attribut valeur_mini valeur_maxi  (ex!de_a age 18 35)
    ordre nom_attribut operateur nom_attribut  (ex!ordre debit<credit)
    moyenne nom_attribut operateur valeur      (ex!moyenne note>8)
    occs nom_attribut operateur valeur         (ex!occs couleur<5)
    occm nom_attribut nom_attribut op. valeur  (ex!occm classe etudiant<30)
    n_a nom_attribut operateur                 (ex!n_a age<)
    ref nom_relation nom_at1 nom_at2           (ex!ref dept num_dep_serv num_dept)


requete !

fin!


requete!
l!

requete!

help!
pour la lecture !
 des noms des ci simples          --->   nom_relation       ci_rel
 des noms des ci de reference     --->   nom_relation       ci_ref
 du texte d une ci simple de ins  --->   nom_rel texte_ci_rel_ins  nom_ci
 du texte d une ci simple de sup  --->   nom_rel texte_ci_rel_sup  nom_ci
 du texte d une ci simple de mod  --->   nom_rel texte_ci_rel_mod  nom_ci
 du texte d une ci de ref de ins  --->   nom_rel texte_ci_ref_ins  nom_rel nom_ci
 du texte d une ci de ref de sup  --->   nom_rel texte_ci_ref_sup  nom_rel nom_ci
 du texte d une ci de ref de mod  --->   nom_rel texte_ci_ref_mod  nom_rel nom_ci


requete!

fin!


requete!
sup!

erreur de syntaxe! operation inconnue sup
```

```
requete:
s:

requete:
help:
 pour supprimer :
   toutes les cis sur une rel    --->   nom_rel toutes
   une ci simple                 --->   nom_rel rel     nom_ci
   une ci de reference           --->   nom_rel ref     nom_rel nom_ci
   une ci simple d insertion     --->   nom_rel rel_ins nom_ci
   une ci simple de suppression  --->   nom_rel rel_sup nom_ci
   une ci simple de modification--->    nom_rel rel_mod nom_ci
   une ci de ref insertion       --->   nom_rel ref_ins nom_rel nom_ci
   une ci de ref suppression     --->   nom_rel ref_sup nom_rel nom_ci
   une ci de ref insertion       --->   nom_rel ref_mod nom_rel nom_ci


requete:
fin:


requete:
autoriser:

erreur de syntaxe: operation inconnue autoriser


requete:
fin:

commande:
autoriser:

commande
help:
les commandes d autorisations sont:
   donner   liste_des_droits sur nom_relation a  nom_utilisateur
   revoquer liste_des_droits sur nom_relation de nom_utilisateur
    mes_droits nom_relation
   n.b.:les droits sont:l(ecture):i(nsertion):s(uppression):m(odification)
         si on rajoute a un droit le caractere t ce droit devient transmissible
         ex: donner lt:i sur c a charles
              charles aura le droit de lire ainsi que d inserer des tuples
              dans c et il a le droit de permettre a d autres utilisateurs
              de lire c


commande
fin:

commande:
calcul:

requete de calcul:
help:
 les commandes de calcul sont :
      max(expression relationnelle:nom attribut)
      min(expression relationnelle:nom attribut)
      somme(expression relationnelle:nom attribut)
      moyenne(expression relationnelle:nom attribut)
      card(expression relationnelle)\n

requete de calcul:
fin:
```

ANNEXE 2 - EXEMPLES D'EXECUTIONS

Connexion à LOISIR

```
pour reduire le contexte de travail
     mb   nom_multibase
     base nom_base
     fin  si vous ne voulez plus reduire votre contexte

fin

commande!
imp_schema!
votre requete doit etre de la forme
   mb nom multibase
   base nom base
   rel nom relation
  n.b.!si vous voulez imprimer la multibase ou la base  sur laquelle vous etes connect
   il faut taper mb ou base main

requete!
mb main
   mb  loisir
      mb  restaurant
         bd  r_mod
            r(numr!e,nomr!ch,tel!ch,arrond!e,st_metro!ch)
            plats(numpl!e,nompl!ch,type!ch)
            menus(numr!e,numpl!e,prix!e)
         bd  r_luxe
            r(num_r!e,nom_r!ch,tel!ch,arrond!e,st_metro!ch)
            plats(num_pl!e,nom_pl!ch,type_c!ch)
            menus(num_r!e,num_pl!e,prix!e)
            cin_rest(nomc!ch,telc!ch,nomr!ch,telr!ch,metro!ch)
      bd  cinema
         c(num_c!e,nom_c!ch,tel!ch,arrond!e,st_metro!ch,nbr_salle!e)
         film(num_f!e,nom_f!ch,pays!ch,genre!ch,version!ch)
         seance(num_c!e,num_f!e)
      bd  metro
         l(num_l!e,prov!ch,dest!ch)
         s(nom_s!ch,correspl!e)
         ls(num_l!e,nom_s!ch)

commande!
mod_sch!

operation inconnuemod_sch

commande!
mod_schema!

requete!
aj_base main hebergement!

requete!
aj_rb hebergement hotel!

schema relation de base
num_h!e,nom_h!ch,categ!e,nb_ch!e,num_h,categ

requete!
aj_rb hebergement client!

schema relation de base
num_c!e,nom_c!ch,tel!ch,num_c,nom_c!

requete!
aj_rb hebergement reserv!

schema relation de base
```

Modification du schema de LOISIR

Ajouter une base hebergement à LOISIR

Ajouter une relation de base hotel à la base hebergement

Ajouter une relation de base client à la base hebergement

Ajouter une relation de base reserv à la base hebergement

```
num_h!e;num_c!e;debut!ch;fin!ch>num_h;num_c;debut>!
```

```
requete!
fin!
```

```
commande!
imp_schema!
votre requete doit etre de la forme
   mb nom multibase
   base nom base
   rel nom relation
  n.b.:si vous voulez imprimer le multibase ou la base  sur laquelle vous etes connec
   il faut taper mb ou base main
```

```
requete!
mb main!
   mb  loisir
      mb  restaurant
         bd  r_mod
            r(numr!e;nomr!ch;tel!ch;arrondie;st_metro!ch)
            plats(numpl!e;nompl!ch;type!ch)
            menus(numr!e;numpl!e;prix!e)
         bd  r_luxe
            r(num_r!e;nom_r!ch;tel!ch;arrondie;st_metro!ch)
            plats(num_pl!e;nom_pl!ch;type_pl!ch)
            menus(num_r!e;num_pl!e;prix!e)
            cin_rest(nomc!ch;telc!ch;nomr!ch;telr!ch;metro!ch)
      bd  cinema
         c(num_c!e;nom_c!ch;tel!ch;arrondie;st_metro!ch;nbr_salle!e)
         film(num_f!e;nom_f!ch;realv!ch;genre!ch;version!ch)
         seance(num_c!e;num_f!e)
      bd  metro
         l(num_l!e;nomv!ch;destin)
         s(nom_s!ch;correspl!e)
         ls(num_l!e;nom_s!ch)
      bd  hebergement
         hotel(num_h!e;nom_h!ch;categ!e;nb_ch!e)
         client(num_c!e;nom_c!ch;tel!ch)
         reserv(num_h!e;num_c!e;debut!ch;fin!ch)
```

Schema de LOISIR

```
commande!
maj!
```

```
requete!
inserer_dans hotel num_h!2;nom_h!concorde;categ!3;nb_ch!125;
```

insertion de tuples dans hotel

```
requete!
i hotel num_h!4;nom_h!frantel;categ!4;nb_ch!220;
```

```
requete!
i hotel num_h!7;nom_h!pantheon;categ!1;
```

```
requete!
fin!
```

```
commande!
interroger!
```

lister hotel

```
requete!
hotel!
loisir,hebergement,hotel.num_h                                    e
loisir,hebergement,hotel.nom_h                                    ch
loisir,hebergement,hotel.categ                                    e
loisir,hebergement,hotel.nb_ch                                    e
2               concorde        3               125
4               frantel         4               220
```

```
7                pantheon        1               ooo

requete:
fin:

commande:                                  Ajouter un attribut au schema d'Hotel
aj_attr:

donner nom_relation suivi de la liste des nom_attribut:domaine

hotel tel:e

commande:
sch_rel:                                   schema de la relation Hotel

nom de la relation :
hotel
num_h:e:nom_h:ch:categ:e:nb_ch:e:tel:e
cle    : num_h
nonul : categ

nom de la relation :                       schema de la relation reserve
reserv
num_h:e:num_c:e:debut:ch:fin:ch
cle    : num_h          num_c
nonul : debut

nom de la relation :
fin:

commande:
interroger:                                lister hotel

requete:
hotel:
loisir.hebergement.hotel.num_h                                  e
loisir.hebergement.hotel.nom_h                                  ch
loisir.hebergement.hotel.categ                                  e
loisir.hebergement.hotel.nb_ch                                  e
loisir.hebergement.hotel.tel                                    e
2                concorde        3               135             ooo
4                frantel         4               120             ooo
7                pantheon        1               ooo             ooo

requete:
```

```
! start_up()
login!
abd
mot_de_passe!
007

categorie des operations a effectuer !

con_mb!                                          connexion à la multibase
                                                       LOISIR
nom multibase!
loisir

pour reduire le contexte de travail
    mb   nom_multibase
    base nom_base
    fin  si vous ne voulez plus reduire votre contexte

fin!

commande!
imp_schema!
votre requete doit etre de la forme
   mb nom multibase
   base nom base
   rel nom relation
  n.b.!si vous voulez imprimer la multibase ou la base  sur laquelle vous etes conne
   il faut taper mb ou base main

requete!                                         schema de LOISIR
mb main
   mb  loisir
      mb  restaurant
         bd  r_mod
            r(numr!e,nomr!ch,tel!ch,arrond!e,st_metro!ch)
            plats(nump!e,nomp!ch,type!ch)
            menus(numr!e,nump!e,prix!e)
         bd  r_luxe
            r(num_r!e,nom_r!ch,tel!ch,arrond!e,st_metro!ch)
            plats(num_p!e,nom_p!ch,type_p!ch)
            menus(num_r!e,num_p!e,prix!e)
      bd  cinema
         c(num_c!e,nom_c!ch,tel!ch,arrond!e,st_metro!ch,nbr_salle!e)
         film(num_f!e,nom_f!ch,pays!ch,genre!ch,version!ch)
         seance(num_c!e,num_f!e)
      bd  metro
         l(num_l!e,prov!ch,dest!ch)
         s(nom_s!ch,corresp!e)
         ls(num_l!e,nom_s!ch)

commande!
interroger!

requete!                                         lister la relation c
c!
loisir.cinema.c.num_c                                        e   } schema
loisir.cinema.c.nom_c                                        ch  }   de
loisir.cinema.c.tel                                          ch  }   c
loisir.cinema.c.arrond                                       e   }
loisir.cinema.c.st_metro                                     ch  }
loisir.cinema.c.nbr_salle                                    e   }
45              cinoche         6331082         5            champs_elysees
2
23              le marais       2784786         7            st germ des pre
```

```
3
7               ambassade       2974970         15              montparnasse
1
100             pathe           3513130         999             montparnasse
999
5               beaubourg       2715236         5               st_michel
2
12              cameo           6543452         5               st_michel
3
14              gaumont         6789342         5               st_michel
999
17              parc            4553445         1               louvre
999
19              elysee          6621598         1               palais_royal
999
125             escale          4598765         5               luxembourg
4
```

lister R_Luxe.R

```
requete;
r_luxe,r;
loisir,restaurant,r_luxe,r,num_r                                e
loisir,restaurant,r_luxe,r,nom_r                                ch
loisir,restaurant,r_luxe,r,tel                                  ch
loisir,restaurant,r_luxe,r,arrond                               e
loisir,restaurant,r_luxe,r,st_metro                             ch
24              les_acacias     5564601         1               palais_royal
5               europe          5335321         1               louvre
7               la_lorraine     7365585         5               st_michel
1               terminus        4263879         5               luxembourg
11              cedre           4322589         5               st_michel
6               sofitel         6964221         1               louvre
14              le_palais       4452372         2               bourse
22              cornettes       7732001         11              voltaire
25              du_coq_chantant                 5               st_michel

requete;
fin;

commande;
mod_schema;

requete;
aj_rv r_luxe cin_rest;
```

Ajouter une relation virtuelle cin_rest à la base R_Luxe

```
definition  relation virtuelle
pr(jn(c r_luxe,r:arrond arrond):nom_c:c,tel:nom_r:r,tel:r,st_metro)
  nomc;nom_c:telc;c,tel:nomr;nom_r:telr;r,tel:metro;st_metro;

requete;
fin;

commande;
imp_schema;
votre requete doit etre de la forme
   mb nom multibase
   base nom base
   rel nom relation
  n,b,;si vous voulez imprimer la multibase ou la base  sur laquelle vous etes connec
   il faut taper mb ou base main

requete;
mb main;
   mb  loisir
      mb  restaurant
         bd  r_mod
            r(numr;e:nomr;ch:tel;ch:arrond;e:st_metro;ch)
```

```
            plats(numple:nomplch:typelch)
            menus(numrle:numple:prixle)
         bd  r_luxe
            r(num_rle:nom_rlch:tellch:arrondle:st_metrolch)
            plats(num_ple:nom_plch:type_plch)
            menus(num_rle:num_ple:prixle)
            cin_rest(nomclch:telclch:nomrlch:telrlch:metrolch)
      bd  cinema
         c(num_cle:nom_clch:tellch:arrondle:st_metrolch:nbr_sallele)
         film(num_fle:nom_flch:payslch:genrelch:versionlch)
         seance(num_cle:num_fle)
      bd  metro
         l(num_lle:provlch:destlch)
         s(nom_slch:corresple)
         ls(num_lle:nom_slch)

commande:
autoriser:

commande
donner lt sur cin_rest a charles;

commande
fin;

commande:
interroger:

requete:
cin_rest;
loisir,restaurant,r_luxe,cin_rest,nomc                    ch
loisir,restaurant,r_luxe,cin_rest,telc                    ch
loisir,restaurant,r_luxe,cin_rest,nomr                    ch
loisir,restaurant,r_luxe,cin_rest,telr                    ch
loisir,restaurant,r_luxe,cin_rest,metro                   ch
cinoche         6331082         la_lorraine     7365585         st_michel
cinoche         6331082         terminus        4263879         luxembourg
cinoche         6331082         cedre           4322589         st_michel
cinoche         6331082         du_coq_chantant                 st_michel
beaubourg       2715236         la_lorraine     7365585         st_michel
beaubourg       2715236         terminus        4263879         luxembourg
beaubourg       2715236         cedre           4322589         st_michel
beaubourg       2715236         du_coq_chantant                 st_michel
cameo           6543452         la_lorraine     7365585         st_michel
cameo           6543452         terminus        4263879         luxembourg
cameo           6543452         cedre           4322589         st_michel
cameo           6543452         du_coq_chantant                 st_michel
gaumont         6789342         la_lorraine     7365585         st_michel
gaumont         6789342         terminus        4263879         luxembourg
gaumont         6789342         cedre           4322589         st_michel
gaumont         6789342         du_coq_chantant                 st_michel
parc            4553445         les_acacias     5564601         palais_royal
parc            4553445         europe          5335321         louvre
parc            4553445         sofitel         6964221         louvre
elysee          6621598         les_acacias     5564601         palais_royal
elysee          6621598         europe          5335321         louvre
elysee          6621598         sofitel         6964221         louvre
escale          4598765         la_lorraine     7365585         st_michel
escale          4598765         terminus        4263879         luxembourg
escale          4598765         cedre           4322589         st_michel
escale          4598765         du_coq_chantant                 st_michel

requete:
fin;

commande:
```

Donner à Charles le droit de lire (avec transmission) cin_rest

lister la relation virtuelle cin_rest

```
fin;

vous voulez sauvegarder la session;o/n
o

categorie des operations a effectuer ;

fin;

; start_up()
login;
charles;
mot_de_passe;
gh
identification incorrecte
login;
charles
mot_de_passe;
chs

categorie des operations a effectuer ;

con_mb;

nom multibase;
help;

les noms de vos multibases sont;
transport
transport

pour reduire le contexte de travail
    mb   nom_multibase
    base nom_base
    fin  si vous ne voulez plus reduire votre contexte

fin

commande;
imp_scheme;
votre requete doit etre de la forme
   mb nom multibase
   base nom base
   rel nom relation
  n.b.;si vous voulez imprimer la multibase ou la base  sur laquelle vous etes conn
   il faut taper mb ou base main

requete;
mb main
   mb  transport
      bd  poids_lourd
         camion(numc:e;nom_immatr:ch;type:ch;marque:ch;annee:e)

commande;
mod_scheme;

requete;
rec_rel poids_lourd c_r;

reception  relation virtuelle
help;
donner nom_donneur nom_rel_donneur liste de correspondance
  ex; abd loisir,cinema,c liste_correspondance

reception relation virtuelle
abd loisir,restaurant,r_luxe,cin_rest nom_cin;nomc:tel_cin;telc:nom_rest;nomr:tel_re
```

Charles se connecte à sa multibase transport

reception de la relation cin_rest dans la base poids_lourd sous le nom c_r

```
requete:
fin;

commande:
imp_schema;
votre requete doit etre de la forme
   mb nom multibase
   base nom base
   rel nom relation
  n.b.:si vous voulez imprimer la multibase ou la base  sur laquelle vous etes conne
   il faut taper mb ou base main

requete:
mb main                                   schema de transport
   mb  transport
      bd  poids_lourd
         camion(numc:e;num_immatr:ch;type:ch;marque:ch;annee:e)
         c_r(nom_cin:ch;tel_cin:ch;nom_rest:ch;tel_rest:ch;metro:ch)

commande:
interroger:

requete:                                  lister c_r
c_r:
transport.poids_lourd.c_r.nom_cin                                   ch
transport.poids_lourd.c_r.tel_cin                                   ch
transport.poids_lourd.c_r.nom_rest                                  ch
transport.poids_lourd.c_r.tel_rest                                  ch
transport.poids_lourd.c_r.metro                                     ch
cinoche         6331082         le_lorraine     7365585         st_michel
cinoche         6331082         terminus        4263879         luxembourg
cinoche         6331082         cedre           4322589         st_michel
cinoche         6331082         du_coq_chantant                 st_michel
beaubourg       2715236         le_lorraine     7365585         st_michel
beaubourg       2715236         terminus        4263879         luxembourg
beaubourg       2715236         cedre           4322589         st_michel
beaubourg       2715236         du_coq_chantant                 st_michel
cameo           6543452         le_lorraine     7365585         st_michel
cameo           6543452         terminus        4263879         luxembourg
cameo           6543452         cedre           4322589         st_michel
cameo           6543452         du_coq_chantant                 st_michel
gaumont         6789342         le_lorraine     7365585         st_michel
gaumont         6789342         terminus        4263879         luxembourg
gaumont         6789342         cedre           4322589         st_michel
gaumont         6789342         du_coq_chantant                 st_michel
parc            4553445         les_acacias     5564601         palais_royal
parc            4553445         europe          5335321         louvre
parc            4553445         sofitel         6964221         louvre
elysee          6621598         les_acacias     5564601         palais_royal
elysee          6621598         europe          5335321         louvre
elysee          6621598         sofitel         6964221         louvre
escale          4598765         le_lorraine     7365585         st_michel
escale          4598765         terminus        4263879         luxembourg
escale          4598765         cedre           4322589         st_michel
escale          4598765         du_coq_chantant                 st_michel

requete:
pr(c_r:nom_cin:metro);
transport.poids_lourd.c_r.nom_cin                                   ch
transport.poids_lourd.c_r.metro                                     ch
cinoche         st_michel
cinoche         luxembourg
beaubourg       st_michel
beaubourg       luxembourg
cameo           st_michel
```

```
cameo           luxembourg
gaumont         st_michel
gaumont         luxembourg
parc            palais_royal
parc            louvre
elysee          palais_royal
elysee          louvre
escale          st_michel
escale          luxembourg

requete;
fin;

commande;
fin;

vous voulez sauvegarder la session;o/n
o

categorie des operations a effectuer :

fin;

; start_up()
login;
abd
mot_de_passe;
007

categorie des operations a effectuer :

con_mb;

nom multibase;
loisir

pour reduire le contexte de travail
    mb   nom_multibase
    base nom_base
    fin  si vous ne voulez plus reduire votre contexte

fin

commande;
autoriser;

commande
revoquer lt sur cin_rest de charles;

commande
fin;

commande;
fin;

vous voulez sauvegarder la session;o/n
o

categorie des operations a effectuer :

fin;

; start_up()
login;
charles
mot_de_passe;
```

l'Abd se connecte à LOISIR

Revoquation du droit lt antérieurement donné à Charles sur la relation cin_rest

```
chs

categorie des operations a effectuer :

imp_schema;

operation inconnueimp_schema

categorie des operations a effectuer :

con_mb;

nom multibase;
transport;

pour reduire le contexte de travail
    mb   nom_multibase
    base nom_base
    fin  si vous ne voulez plus reduire votre contexte

fin

commande;
imp_schema;
votre requete doit etre de la forme
   mb nom multibase
   base nom base
   rel nom relation
  n.b.:si vous voulez imprimer la multibase ou la base  sur laquelle vous etes connec
   il faut taper mb ou base main

requete;
mb main;
   mb  transport
      bd  poids_lourd
         camion(numc!e:num_immatr!ch:type!ch:marque!ch:annee!e)
         c_r(nom_cin!ch:tel_cin!ch:nom_rest!ch:tel_rest!ch:metro!ch)

commande;
interroger;

requete;
c_r;
vous navez pas le droit de lire transport.poids_lourd.c_r

requete;
fin;

commande;
mod_schema;

requete;
sup_rel c_r;

requete;
fin;

commande;
imp_schema;
votre requete doit etre de la forme
   mb nom multibase
   base nom base
   rel nom relation
  n.b.:si vous voulez imprimer la multibase ou la base  sur laquelle vous etes connec
   il faut taper mb ou base main
```

Charles se connecte à TRANSPORT

Charles n'a plus le droit de lire C_R

suppression de la relation C_R

```
requete;
mb main.
   mb  transport
      bd  poids_lourd
         camion(numc:e;num_immatr:ch;type:ch;marque:ch;annee:e)

commande;
fin;

vous voulez sauvegarder la session:o/n
n

categorie des operations a effectuer :

fin
```

```
% sdonis
login;
sbd
mot_de_passe;
007

categorie des operations a effectuer :

con_mb;

nom multibase;
loisir

pour reduire le contexte de travail
    mb   nom_multibase
    base nom_base
    fin  si vous ne voulez plus reduire votre contexte

fin

commande;
imp_schema;
votre requete doit etre de la forme
   mb nom multibase
   base nom base
   rel nom relation
  n.b.:si vous voulez imprimer la multibase ou la base  sur laquelle vous etes conne
   il faut taper mb ou base main

requete;
mb main
   mb  loisir
      mb  restaurant
         bd  r_mod
            r(numr!e:nomr!ch:tel!ch:arrond!e:st_metro!ch)
            plats(numpl!e:nompl!ch:typepl!ch)
            menus(numr!e:numpl!e:prix!e)
         bd  r_luxe
            r(num_r!e:nom_r!ch:tel!ch:arrond!e:st_metro!ch)
            plats(num_pl!e:nom_pl!ch:type_pl!ch)
            menus(num_r!e:num_pl!e:prix!e)
            cin_rest(nomc!ch:telc!ch:nomr!ch:telr!ch:metro!ch)
      bd  cinema
         c(num_c!e:nom_c!ch:tel!ch:arrond!e:st_metro!ch:nbr_salle!e)
         film(num_f!e:nom_f!ch:pays!ch:genre!ch:version!ch)
         seance(num_c!e:num_f!e)
      bd  metro
         l(num_l!e:prov!ch:gestion)
         s(nom_s!ch:correspl!e)
         ls(num_l!e:nom_s!ch)

commande;
interroger;

requete;
r_luxe.r;
loisir.restaurant.r_luxe.r.num_r                              e
loisir.restaurant.r_luxe.r.nom_r                              ch
loisir.restaurant.r_luxe.r.tel                                cn
loisir.restaurant.r_luxe.r.arrono                             e
loisir.restaurant.r_luxe.r.st_metro                           ch
24              les_acacias     5564601         :             palais_royal
5               europe          5225221         :             louvre
```

connexion à LOISIR

lister R_luxe.R

```
7               le_lorraine     7365585         5               st_michel
1               terminus        4263879         5               luxembourg
11              cedre           4322580         5               st_michel
6               sofitel         6964221         1               louvre
14              le_relais       4452372         2               bourse
22              cornettes       7732001         11              voltaire
25              du_coq_chantant                 5               st_michel

requete:
r_luxe,plats;
loisir,restaurant,r_luxe,plats.num_r                            3
loisir,restaurant,r_luxe,plats.nom_r                            ch
loisir,restaurant,r_luxe,plats.type_r                           ch
1               fourchette_c_orchinois
4               choucroute      francaise
7               foie_gras       francaise
24              fruits_de_mer   francaise
10              couscous        marocain
12              ozelle          espagnol
23              crustaces       francais
30              zarzuela        espagnol
11              gambas          espagnol
14              quiche          francais

requete:
r_luxe,menus;
loisir,restaurant,r_luxe,menus.num_r                            3
loisir,restaurant,r_luxe,menus.num_p                            6
loisir,restaurant,r_luxe,menus.prix                             6
7               7               45
7               10              35
7               23              40
5               14              30
5               24              25
5               10              30

requete:
s;
loisir,metro,s,nom_s                                            20
loisir,metro,s,correspondance                                   6
champs_elysees 0
st_germ_des_pre0
odeon           1
st_michel       1
pigalle         0
opera           0
luxembourg      0
prt_d_orleans   0
pl_d_italie     0
montparnasse    0
palais_royal    1
louvre          1
chateau_d'eau   1
bourse          1
voltaire        0
dugommier       0
place_d'italie  1
rome            0
gambetta        1
vaugirard       1

requete:
fin;

commande:
contrainte:
```

lister R_Luxe.Plats

lister R_Luxe.Menus

lister S

```
requete!
p!

requete !

c1 r_luxe,menus ref r_luxe,r num_r num_r!

requete !

c2 r_luxe,menus ref r_luxe,plats num_p num_p!

requete !

c3 r_luxe,r ref r_luxe,s et_metro nom_s!
nom inconnu! loisir,r_luxe,s

requete !

c3 r_luxe,r ref s et_metro nom_s!

requete !

c4 r_luxe,menus moyenne prix>150!
la contrainte n est pas verifiee


requete !

c4 r_luxe,menus moyenne prix>250!

requete !

c5 r_luxe,menus n_s prix !!

requete !

 in!

requete!
fin!

commande!
maj!

requete!
m r_luxe,menus num_r!7,num_p!7,prix!30!
la contrainte suivante n est pas verifiee

   nouv_ancien  prix !

requete!
m r_luxe,menus num_r!7,num_p!7,prix!65!

requete!
m r_luxe,menus num_p!7,num_p!7,prix!2000!
la contrainte suivante n est pas verifiee

   moyenne   prix < 250

requete!
fin!

commande!
interroger!
```

Poser une contrainte de reference
entre Menus et R de R_Luxe

entre Menus et Plats de R_Luxe

entre R_Luxe.R et S

Poser une contrainte simple :
moyenne (prise) de R_Luxe.Menus

Le nouveau prix d'un Menus
doit être superieur à son
ancien prix

Misc_à_jour

Modification refusée

Modification acceptée

Modification refusée

lister R-Luxe.Menus

```
requete:
r_luxe,menus;
loisir,restaurant,r_luxe,menus,num_r                    e
loisir,restaurant,r_luxe,menus,num_r                    e
loisir,restaurant,r_luxe,menus,prix                     e
5               10              30
7               10              35
7               23              40
5               14              20
5               24              35
7               7               45

requete:
contrainte;
nom inconnu! loisir,contrainte

requete:
fin;

commande:
contrainte;

requete:
1;
```

lister les contraintes :
simples sur R_Luxe.Menus

```
requete:

r_luxe,menus ci_rel;
 noms des ci simples d insertion :

    c4
 noms des ci simples de suppression :

    c4
 noms des ci simples de modification :

    c4
    c5
```

de reference sur R_Luxe.Menus

```
requete:

r_luxe,menus ci_ref;
 noms des ci de ref directe d insertion ainsi que les noms des relations referees:
   c1   loisir,restaurant,r_luxe,r
   c2   loisir,restaurant,r_luxe,plats
 noms des ci de ref directe de suppression ainsi que les noms des relations referees
   c1   loisir,restaurant,r_luxe,r
   c2   loisir,restaurant,r_luxe,plats
 noms des ci de ref directe de modification ainsi que les noms des relations referee
   c1   loisir,restaurant,r_luxe,r
   c2   loisir,restaurant,r_luxe,plats
 noms des ci de ref inverse d insertion ainsi que les noms des relations referees:
 noms des ci de ref inverse de suppression ainsi que les noms des relations referees
 noms des ci de ref inverse de modification ainsi que les noms des relations referee

requete:

r_luxe,r ci_ref;
```

de reference sur R_Luxe.R

```
 noms des ci de ref directe d insertion ainsi que les noms des relations referees:
   c3   loisir,metro,s
 noms des ci de ref directe de suppression ainsi que les noms des relations referees
   c3   loisir,metro,s
 noms des ci de ref directe de modification ainsi que les noms des relations referee
   c3   loisir,metro,s
```

```
noms des ci de ref inverse d insertion ainsi que les noms des relations referees!
  ci   loisir.restaurant.r_luxe.menus
noms des ci de ref inverse de suppression ainsi que les noms des relations referees
  ci   loisir.restaurant.r_luxe.menus
noms des ci de ref inverse de modification ainsi que les noms des relations referee
  ci   loisir.restaurant.r_luxe.menus

requete!

fin!


requete!
fin!

commande!
maj!

requete!
s s nom_s!odeon!

requete!
s s nom_s!bourse!
contrainte de reference inverse non verifiee

   reference   loisir.restaurant.r_luxe.r_st_metro  nom_s

requete!
fin!

commande!
calcul!

requete de calcul!
mav(r_luxe.menus.prix)!
55

requete de calcul!
mov(r_luxe.menus.prix)!
operation inconnue !mov

requete de calcul!
moyenne(r_luxe.menus.prix)!
37.5
```

Mise-à-jour

- suppression du tuple de S dont la clé est odeon

Refus de la suppression du tuple de S de clé bourse

Requêtes de calcul

RESUME

Dans ce travail, nous nous sommes intéressés aux problèmes de confidentialité et d'intégrité dans les multibases de données relationnelles. L'approche multibase(LIT (81)) permettant d'aborder les problèmes de la répartition d'un point de vue logique et non physique. Dans notre système de gestion de multibases de données (ADONIS), la confidentialité est assurée par l'utilisation de relations virtuelles et de prédicats de sécurité. Tout utilisateur peut avoir le droit de donner et de revoquer des autorisations d'accès sur des données à d'autres utilisateurs. ADONIS permet aussi de mettre des contraintes d'intégrité statiques et dynamiques sur les multibases.

ADONIS a été implanté en CLU sur VAX 750, L'utilisation d'un langage supportant les concepts d'abstraction et de modularité (CLU) nous est apparue nécessaire afin de pouvoir utiliser les techniques de développement relevant du génie logiciel lors de la conception et de la réalisation de notre système. Assurément, la maintenance, l'évolutivité et la transportabilité d'ADONIS peuvent alors se faire à moindre coût.

MOTS CLES :

Abstraction - Modularité - Génie logiciel - Multibase de données - Système de gestion de multibases de données - Modèle relationnel - Confidentialité - Intégrité - Sécurité - relation virtuelle - Contrainte.

RESUME

Dans ce travail, nous nous sommes intéressés aux problèmes de confidentialité et d'intégrité dans les multibase de données relationnelles. L'approche multibase(LIT (81)) permettant d'aborder les problèmes de la répartition d'un point de vue logique et non physique. Dans notre système de gestion de multibases de données (ADONIS), la confidentialité est assurée par l'utilisation de relations virtuelles et de prédicats de sécurité. Tout utilisateur peut avoir le droit de donner et de revoquer des autorisations d'accès sur des donnée à d'autres utilisateurs. ADONIS permet aussi de mettre des contraintes d'intégrité statiques et dynamiques sur les multibases.

ADONIS a été implanté en CLU sur VAX 750. L'utilisati d'un langage supportant les concepts d'abstraction et de modul rité (CLU) nous est apparue nécessaire afin de pouvoir utiliser les techniques de développement relevant du génie logiciel lors de la conception et de la réalisation de notre système. Assurément, la maintenance, l'évolutivité et la trans portabilité d'ADONIS peuvent alors se faire à moindre coût.

MOTS CLES :

Abstraction - Modularité - Génie logiciel - Multibase de données - Système de gestion de multibases de données - Modèle relationnel - Confidentialité - Intégrité - Sécurité - relation virtuelle - Contrainte.