HAL
open science

Un système d'autorisation pour une base de données généralisées : projet TIGRE

Fatma Azrou

To cite this version:

Fatma Azrou. Un système d'autorisation pour une base de données généralisées : projet TIGRE. Modélisation et simulation. Institut National Polytechnique de Grenoble - INPG, 1984. Français. NNT : . tel-00311549

HAL Id: tel-00311549

https://theses.hal.science/tel-00311549

Submitted on 19 Aug 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THESE

présentée à

l'Institut National Polytechnique de Grenoble

pour obtenir le grade de

DOCTEUR DE 3ème CYCLE
"Informatique"

par

Fatma AZROU

UN SYSTEME D'AUTORISATION POUR UNE BASE DE DONNEES GENERALISEES.
Projet TIGRE

Thèse soutenue le 29 juin 1984 devant la Commission d'Examen :

Monsieur	C. DELOBEL	: Président
Messieurs	M. ADIBA J. MOSSIERE G.T. NGUYEN	} Examinateurs

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Année universitaire 1982-1983

Président de l'Université : D. BLOCH

Vice-Président : René CARRE
Hervé CHERADAME
Marcel IVANES

PROFESSEURS DES UNIVERSITES :

ANCEAU François	E.N.S.I.M.A.G.
BARRAUD Alain	E.N.S.I.E.G.
BAUDELET Bernard	E.N.S.I.E.G.
BESSON Jean	E.N.S.E.E.G.
BLIMAN Samuel	E.N.S.E.R.G.
BLOCH Daniel	E.N.S.I.E.G.
BOIS Philippe	E.N.S.H.G.
BONNETAIN Lucien	E.N.S.E.E.G.
BONNIER Etienne	E.N.S.E.E.G.
BOUVARD Maurice	E.N.S.H.G.
BRISSONNEAU Pierre	E.N.S.I.E.G.
BUYLE BODIN Maurice	E.N.S.E.R.G.
CAVAIGNAC Jean-François	E.N.S.I.E.G.
CHARTIER Germain	E.N.S.I.E.G.
CHENEVIER Pierre	E.N.S.E.R.G.
CHERADAME Hervé	U.E.R.M.C.P.P.
CHERUY Arlette	E.N.S.I.E.G.
CHIAVERINA Jean	U.E.R.M.C.P.P.
COHEN Joseph	E.N.S.E.R.G.
COUMES André	E.N.S.E.R.G.
DURAND Francis	E.N.S.E.E.G.
DURAND Jean-Louis	E.N.S.I.E.G.
FELICI Noël	E.N.S.I.E.G.
FOULARD Claude	E.N.S.I.E.G.
GENTIL Pierre	E.N.S.E.R.G.
GUERIN Bernard	E.N.S.E.R.G.
GUYOT Pierre	E.N.S.E.E.G.
IVANES Marcel	E.N.S.I.E.G.
JAUSSAUD Pierre	E.N.S.I.E.G.
JOUBERT Jean-Claude	E.N.S.I.E.G.
JOURDAIN Geneviève	E.N.S.I.E.G.
LACOUME Jean-Louis	E.N.S.I.E.G.
LATOMBE Jean-Claude	E.N.S.I.M.A.G.

.../...

LESSIEUR Marcel	E.N.S.H.G.
LESPINARD Georges	E.N.S.H.G.
LONGEQUEUE Jean-Pierre	E.N.S.I.E.G.
MAZARE Guy	E.N.S.I.M.A.G.
MOREAU René	E.N.S.H.G.
MORET Roger	E.N.S.I.E.G.
MOSSIERE Jacques	E.N.S.I.M.A.G.
PARIAUD Jean-Charles	E.N.S.E.E.G.
PAUTHENET René	E.N.S.I.E.G.
PERRET René	E.N.S.I.E.G.
PERRET Robert	E.N.S.I.E.G.
PIAU Jean-Michel	E.N.S.H.G.
POLOUJADOFF Michel	E.N.S.I.E.G.
POUPOT Christian	E.N.S.E.R.G.
RAMEAU Jean-Jacques	E.N.S.E.E.G.
RENAUD Maurice	U.E.R.M.C.P.P.
ROBERT André	U.E.R.M.C.P.P.
ROBERT François	E.N.S.I.M.A.G.
SABONNADIERE Jean-Claude	E.N.S.I.E.G.
SAUCIER Gabrielle	E.N.S.I.M.A.G.
SCHLENKER Claire	E.N.S.I.E.G.
SCHLENKER Michel	E.N.S.I.E.G.
SERMET Pierre	E.N.S.E.R.G.
SILVY Jacques	U.E.R.M.C.P.P.
SOHM Jean-Claude	E.N.S.E.E.G.
SOUQUET Jean-Louis	E.N.S.E.E.G.
VEILLON Gérard	E.N.S.I.M.A.G.
ZADWORNY François	E.N.S.E.R.G.

PROFESSEURS ASSOCIES

BASTIN Georges	E.N.S.H.G.
BERRIL John	E.N.S.H.G.
CARREAU Pierre	E.N.S.H.G.
GANDINI Alessandro	U.E.R.M.C.P.P.
HAYASHI Hirashi	E.N.S.I.E.G.

PROFESSEURS UNIVERSITE DES SCIENCES SOCIALES (Grenoble II)

BOLLIET Louis
Chatelin Françoise

PROFESSEURS E.N.S. Mines de Saint-Etienne

RIEU Jean
SOUSTELLE Michel

CHERCHEURS DU C.N.R.S.

FRUCHART Robert	Directeur de Recherche
VACHAUD Georges	Directeur de Recherche

.../...

ALLIBERT Michel	Maître de Recherche
ANSARA Ibrahim	Maître de Recherche
ARMAND Michel	Maître de Recherche
BINDER Gilbert	
CARRE René	Maître de Recherche
DAVID René	Maître de Recherche
DEPORTES Jacques	
DRIOLE Jean	Maître de Recherche
GIGNOUX Damien	
GIVORD Dominique	
GUELIN Pierre	
HOPFINGER Emil	Maître de Recherche
JOUD Jean-Charles	Maître de Recherche
KAMARINOS Georges	Maître de Recherche
KLEITZ Michel	Maître de Recherche
LANDAU Ioan-Dore	Maître de Recherche
LASJAUNIAS J.C.	
MERMET Jean	Maître de Recherche
MUNIER Jacques	Maître de Recherche
PIAU Monique	
PORTESEIL Jean-Louis	
THOLENCE Jean-Louis	
VERDILLON André	

CHERCHEURS du MINISTERE de la RECHERCHE et de la TECHNOLOGIRE (Directeurs et Maîtres de Recherches, ENS Mines de St. Etienne)

LESBATS Pierre	Directeur de Recherche
BISCONDI Michel	Maître de Recherche
KOBYLANSKI André	Maître de Recherche
LE COZE Jean	Maître de Recherche
LALAUZE René	Maître de Recherche
LANCELOT Francis	Maître de Recherche
THEVENOT François	Maître de Recherche
TRAN MINH Canh	Maître de Recherche

PERSONNALITES HABILITEES à DIRIGER des TRAVAUX de RECHERCHE (Décision du Conseil Scientifique)

ALLIBERT Colette	E.N.S.E.E.G.
BERNARD Claude	E.N.S.E.E.G.
BONNET Rolland	E.N.S.E.E.G.
CAILLET Marcel	E.N.S.E.E.G.
CHATILLON Catherine	E.N.S.E.E.G.
CHATILLON Christian	E.N.S.E.E.G.
COULON Michel	E.N.S.E.E.G.
DIARD Jean-Paul	E.N.S.E.E.G.
EUSTAPOPOULOS Nicolas	E.N.S.E.E.G.
FOSTER Panayotis	E.N.S.E.E.G.

.../...

GALERIE Alain	E.N.S.E.E.G.
HAMMOU Abdelkader	E.N.S.E.E.G.
MALMEJAC Yves	E.N.S.E.E.G. (CENG)
MARTIN GARIN Régina	E.N.S.E.E.G.
NGUYEN TRUONG Bernadette	E.N.S.E.E.G.
RAVAINE Denis	E.N.S.E.E.G.
SAINFORT	E.N.S.E.E.G. (CENG)
SARRAZIN Pierre	E.N.S.E.E.G.
SIMON Jean-Paul	E.N.S.E.E.G.
TOUZAIN Philippe	E.N.S.E.E.G.
URBAIN Georges	E.N.S.E.E.G. (Laboratoire des ultra-réfractaires ODEILLON)
GUILHOT Bernard	E.N.S. Mines Saint Etienne
THOMAS Gérard	E.N.S. Mines Saint Etienne
DRIVER Julien	E.N.S. Mines Saint Etienne
BARIBAUD Michel	E.N.S.E.R.G.
BOREL Joseph	E.N.S.E.R.G.
CHOVET Alain	E.N.S.E.R.G.
CHEHIKIAN Alain	E.N.S.E.R.G.
DOLMAZON Jean-Marc	E.N.S.E.R.G.
HERAULT Jeanny	E.N.S.E.R.G.
MONLLOR Christian	E.N.S.E.R.G.
BORNARD Guy	E.N.S.I.E.G.
DESCHIZEAU Pierre	E.N.S.I.E.G.
GLANGEAUD François	E.N.S.I.E.G.
KOFMAN Walter	E.N.S.I.E.G.
LEJEUNE Gérard	E.N.S.I.E.G.
MAZUER Jean	E.N.S.I.E.G.
PERARD Jacques	E.N.S.I.E.G.
REINISCH Raymond	E.N.S.I.E.G.
ALEMANY Antoine	E.N.S.H.G.
BOIS Daniel	E.N.S.H.G.
DARVE Félix	E.N.S.H.G.
MICHEL Jean-Marie	E.N.S.H.G.
OBLED Charles	E.N.S.H.G.
ROWE Alain	E.N.S.H.G.
VAUCLIN Michel	E.N.S.H.G.
WACK Bernard	E.N.S.H.G.
BERT Didier	E.N.S.I.M.A.G.
CALMET Jacques	E.N.S.I.M.A.G.
COURTIN Jacques	E.N.S.I.M.A.G.
COURTOIS Bernard	E.N.S.I.M.A.G.
DELLA DORA Jean	E.N.S.I.M.A.G.
FONLUPT Jean	E.N.S.I.M.A.G.
SIFAKIS Joseph	E.N.S.I.M.A.G.
CHARUEL Robert	U.E.R.M.C.P.P.
CADET Jean	C.E.N.G.
COEURE Philippe	C.E.N.G. (LETI)

.../...

DELHAYE Jean-Marc	C.E.N.G. (STT)
DUPUY Michel	C.E.N.G. (LETI)
JOUVE Hubert	C.E.N.G. (LETI)
NICOLAU Yvan	C.E.N.G. (LETI)
NIFENECKER Hervé	C.E.N.G.
PERROUD Paul	C.E.N.G.
PEUZIN Jean-Claude	C.E.N.G. (LETI)
TAIEB Maurice	C.E.N.G.
VINCENDON Marc	C.E.N.G.

LABORATOIRES EXTERIEURS

DEMOULIN Eric	C.N.E.T.
DEVINE	C.N.E.T. (R.A.B.)
GERBER Roland	C.N.E.T.
MERCKEL Gérard	C.N.E.T.
PAULEAU Yves	C.N.E.T.
GAUBERT C.	I.N.S.A. Lyon

Je tiens à remercier :

Monsieur Claude DELOBEL, professeur à l'Université Scientifique et Médicale de Grenoble, qui me fait l'honneur de présider le jury de cette thèse.

Monsieur Michel ADIBA, professeur à l'Université Scientifique et Médicale de Grenoble et responsable du projet TIGRE, qui m'a confié ce travail et n'a cessé de s'y intéresser. Je tiens à lui exprimer ma profonde gratitude pour ses nombreux conseils et encouragements.

Monsieur NGUYEN GIA TOAN, ingénieur INRIA, pour les nombreuses discussions que nous avons eu sur l'intérêt de la programmation en logique. Je le remercie également de me faire l'honneur de participer au jury.

Monsieur Jacques MOSSIERE, professeur à l'Institut National Polytechnique de Grenoble et directeur du Laboratoire de Génie Informatique, de me faire l'honneur de participer au jury.

Monsieur Paul WILMS du Laboratoire IBM San JOSE, pour avoir eu la patience de lire en détail la version originale de cette thèse et pour ses nombreux conseils et suggestions.

Monsieur Sid'Ahmed LARIBI, professeur à l'Université des Sciences et de la technologie d'Alger, pour l'intérêt qu'il a témoigné pour ce travail et pour les encouragements qu'il n'a cessé de me prodiguer.

Je tiens aussi à exprimer toute ma reconnaissance à mes collègues des équipes MICROBE et TIGRE ainsi qu'à mes collègues du Département Informatique de l'Université d'Alger et du Centre d'Information Scientifique et Technique d'Alger pour leur soutien amical durant la réalisation de ce travail.

Je remercie également monsieur D. IGLESIAS et le Service de Reprographie pour le soin apporté au tirage de ce texte.

SOMMAIRE

0. INTRODUCTION

1. PRESENTATION DU PROJET TIGRE

1.1 Architecture

1.2 Applications

1.3 Modèle de données

1.3.1 Langage de définition

1.3.2 Langage de manipulation

1.4 Interfaces utilisateurs

1.5 Le système d'autorisation SAGE

2. ETAT DE L'ART

2.1 Protection dans les SGBD hiérarchiques et réseaux

2.1.1 IMS

2.1.2 CODASYL-DBTG

2.2 Protection dans les SGBD relationnels

2.2.1 INGRES

2.2.1.1 Algorithme de contrôle d'accès

2.2.1.2 Optimisation de l'algorithme

2.2.1.3 Conclusion

2.2.2 Le système R

2.2.2.1 Les usagers

2.2.2.2 Les objets

2.2.2.3 Les privilèges

2.2.2.4 Propagation des privilèges

2.2.2.5 Extension à un environnement réparti

2.2.3 Systèmes de WOOD et de BUSSOLATI

2.3 Conclusion

2.4 Notre approche : SACE

2.4.1 Protection des données généralisées

2.4.2 Modélisation de l'environnement par le système

2.4.3 Décentralisation de la fonction d'autorisation

2.4.4 Rôle de l'administrateur de bases de données

2.4.5 Transmission des privilèges

2.4.6 Révocation implicite par suppression d'usagers

2.4.7 Autres caractéristiques de SACE

3. DESCRIPTION DE SACE

3.1 Gestion des utilisateurs

3.1.1 Représentation interne

3.1.2 Les groupes d'utilisateurs

3.1.3 Opérations sur les groupes

3.1.4 Conclusion

3.2 Les objets

3.3 Gestion des privilèges

3.3.1 Privilèges sur les objets

3.3.2 Raffinement

3.3.3 Les catalogues

3.3.4 Accès aux objets structurés

3.3.5 Accès à un objet à partir d'un poste de travail

3.3.6 Propagation des privilèges

3.3.7 Révocation des privilèges

3.3.8 Conclusion

3.4 Réorganisation de l'entreprise

3.4.1 Le problème

3.4.2 Arrivées et départs des utilisateurs

3.4.3 Restructuration de l'entreprise

3.5 Aspects d'implantation

3.5.1 Utilisateurs et groupes

3.5.2 Contrôle et autorisation

3.5.3 Réorganisation de l'entreprise

4. ELEMENTS DE REALISATION

4.1 Le SGBD MICROBE

4.2 L'interface MICROBE-TIGRE

4.3 Position de SAGE dans TIGRE et dans MICROBE

4.4 les catalogues de SAGE

5. CONCLUSION

BIBLIOGRAPHIE

CHAPITRE 0

INTRODUCTION

Le développement actuel des applications utilisant des bases de données nécessite le développement conceptuel et la mise en oeuvre de mécanismes d'autorisation d'accès. Il s'agit de tous les mécanismes destinés à contrôler le droit d'accès et de propagation de chaque privilège, de chaque utilisateur sur telle ou telle donnée. Ces mécanismes doivent non seulement être capables de vérifier l'accès à des données de types classiques (entiers, réels, chaînes de caractères) mais aussi à des données de type bien plus général puisque la tendance aujourd'hui est de développer des Systèmes de Gestion de Bases de Données Généralisées (SGBDG). Ceux-ci sont capables de gérer, outre les données alphanumériques classiques, des informations naturelles de type texte, image, voix... du point de vue de leur saisie, de leur stockage, de leur manipulation et de leur diffusion. Ces données dites généralisées recouvrent divers domaines d'application tels que : la conception assistée, la bureautique, le traitement des images et de la parole et l'ingénierie du logiciel /CRA81/, /JOL81/, /ADI82/, /ADI83/, /TIGR1/, /TIGR2/, /TIGR5/.

Le projet TIGRE, développé à Grenoble, conjointement par le Laboratoire de Génie Informatique de l'IMAG et le centre de recherche BULL, a pour objectif la conception et la réalisation d'un SGBDG. Le système visé est orienté principalement vers la gestion des documents dans un contexte bureautique. Le document est vu comme un objet complexe et structuré, pouvant contenir du texte, des formules et des images. Il est considéré comme

l'unité de base pour la communication et le traitement dans un tel environnement.

Notre travail s'intéresse d'abord au développement d'un mécanisme d'autorisation capable d'assurer la sécurité des objets gérés, en général, et des documents en particulier. Ce mécanisme apparaît comme l'une des fonctions essentielles d'un système capable de gérer des données généralisées. Les travaux les plus récents, développés dans le cadre des SGBD relationnels, ne prennent pas en compte les types de données complexes /WIL80/, /BUS81/. Le problème de la protection d'un objet de structure complexe tel que le document reste ainsi posé. En effet, pour un tel objet il est nécessaire de prévoir sa protection à divers niveaux : tout le document, un chapitre, une section etc. Il nous semble également avantageux de pouvoir décrire au niveau du SGBDG, de façon intégrée les données et les utilisateurs. On décrit ainsi, dans la base de données, l'environnement dans lequel elle est utilisée. Ceci pose le problème de la prise en compte, par le modèle, de l'organisation de l'entreprise. Ainsi, on peut lier étroitement les utilisateurs effectifs à leurs droits d'accès.

Pour résoudre ces problèmes, nous proposons un Système d'Autorisation GEnéralisé (SAGE) qui répond aux deux objectifs précédents, à savoir :

-Gestion des objets de type complexe et notamment de type document.

-Modélisation de l'environnement réel dans lequel le SGBDG est mis en oeuvre.

Le plan de la thèse est le suivant :

-Le premier chapitre présente brièvement le projet TIGRE et en particulier le modèle de données et le langage de définition et de manipulation de ce modèle : LAMBDA.

-Le deuxième chapitre est consacré à un bref aperçu de l'état de l'art en matière d'autorisation d'accès dans les SGBD. Les principaux objectifs de notre modèle d'autorisation d'accès et les justifications de notre approche y sont aussi, brièvement présentés.

-Le chapitre suivant décrit en détail notre solution. Dans la première partie de ce chapitre nous décrivons les concepts de base de notre modèle SAGE, qui sont : (1) les <u>objets</u> à protéger, (2) les <u>utilisateurs</u> ou <u>groupes d'utilisateurs</u> qui créent et manipulent les objets et (3) les <u>privilèges</u> qui peuvent être attribués ou retirés aux utilisateurs, sur les objets. Les particularités de ces concepts y sont bien mises en évidence. Un langage d'autorisation et de gestion des utilisateurs est proposé comme une extension de LAMBDA. Puis, les problèmes posés par la restructuration de l'entreprise et leurs répercussions sur SAGE sont discutés. Enfin, une implantation modulaire de SAGE est proposée, sous forme de primitives. Elle met en évidence la portabilité de celui-ci et son indépendance vis-à-vis de tout SGBD.

-Le quatrième chapitre est consacré à une description de l'aspect

réalisation. Celle-ci est effectuée sur le modèle TIGRE et le SGBD relationnel sous-jacent MICROBE.

-Le dernier chapitre évoque les extensions éventuelles de cette étude sur des aspects qui nous paraissent importants. Ces extensions peuvent être considérées comme une suite de notre travail.

CHAPITRE I

PRESENTATION DU PROJET TIGRE

Les exigences d'applications telles que la bureautique, la CAO, l'ingénierie du logiciel et le traitement graphique mettent en évidence les insuffisances des SGBD actuels. Ces exigences portent en particulier sur les types de données qui ne sont pas seulement alphanumériques mais comprennent aussi des textes, des images et la voix. Actuellement, des efforts importants sont consacrés à essayer de pallier aux insuffisances des SGBD classiques. Par exemple OBE est une extension de QBE pour manipuler des données textuelles dans un environnement bureautique /ZLO77/, /LOR83/.

A Grenoble, des travaux de recherche en ce domaine ont été menés dès 1979. Un système expérimental de bases de données textuelles a été réalisé par extension du SGBD SOCRATE /JOL81/, /KOW82/. Un résultat important de ce travail est l'identification des insuffisances des SGBD classiques, ce qui a permis de préparer un projet de recherche plus vaste dans le domaine des Bases de Données Généralisées (BDG) : le projet TIGRE. Les principaux axes de développement concernent les aspects suivants :

- Architecture système
- Etudes d'applications
- Modèle de données
- Interfaces utilisateurs : éditeur de document, langage PROLOG.

Dans ce chapitre nous présentons rapidement chacun de ces axes. Nous insistons plus particulièrement sur le modèle de données sur

lequel sera construit notre système SAGE.

1.1 L'architecture

Le système TIGRE est conçu pour répondre aux besoins d'applications nouvelles (CAO, bureautique...). Il s'adresse à un groupe de 10 à 100 personnes travaillant sur des données généralisées (par exemple entiers, réels, documents). L'architecture choisie est basée sur des postes de travail élaborés (PT) et des serveurs (base de données, imprimantes...). Les divers constituants de l'architecture communiquent en utilisant un réseau local de type ETHERNET. La structure du système peut être représentée ainsi :

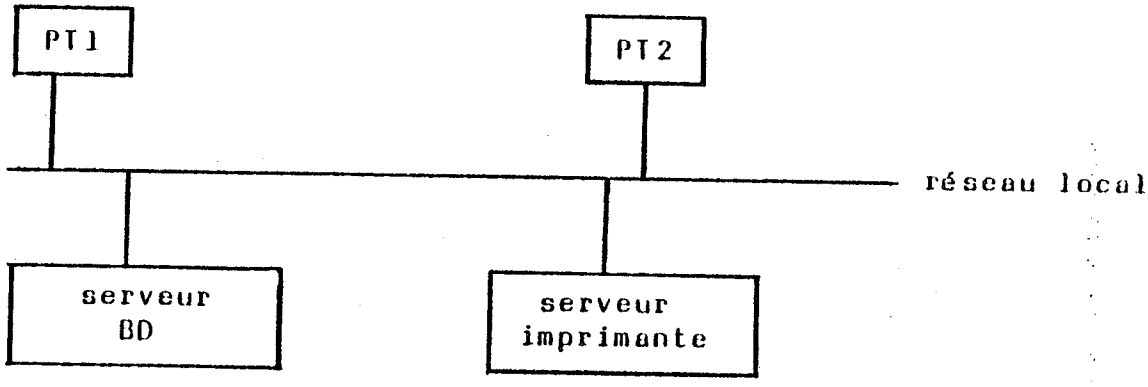


Figure 1 : architecture générale de TIGRE

Le serveur BD offre les services de stockage et de manipulation accessibles à partir des divers postes de travail. Pour réduire le transfert des données sur le réseau et décharger

le serveur BD, les requêtes sont partiellement traitées sur le PT. Ceci est rendu possible grâce à l'existence d'un service local de stockage et de manipulation des données propres à chaque poste. Le serveur imprimante est accessible à toutes les machines connectées sur le réseau y compris le serveur BD.

1.2 Les applications

Dans le projet TIGRE, une importance particulière est accordée à la bureautique. Les objets manipulés dans un contexte de bureautique sont essentiellement des documents. Ces documents ne sont pas atomiques et possèdent une structure interne et une sémantique qui leur sont propres. On définit un "document généralisé" comme un objet pouvant contenir des données alphanumériques, des images et des formules mathématiques /TIGR6/. Il est structuré hiérarchiquement ce qui fournit un chemin d'accès à ses différentes parties. Des éléments non hiérarchiques (notes, figures...) sont également pris en compte. Les éléments hiérarchiques et non hiérarchiques ont des attributs qui leur sont associés. Ces attributs sont utilisés pour identifier une structure, la présentation ou la définition de contraintes d'intégrité ou de confidentialité, par exemple.
Le document est l'unité d'échange dans un contexte bureautique.

Par ailleurs, une étude a été réalisée concernant la CAO, le traitement de la parole et les modèles économétriques /ADI83/, /TIGR9/, /TIG13/. Une recherche sur le développement

d'applications formulaires est également en cours /TIGR9/.

1.3 Modèle de données

Une base de données est décrite par un schéma (conceptuel) et contient plusieurs occurrences d'éléments de ce schéma. Le schéma conceptuel est en général exprimé au moyen d'un modèle de données. Ainsi dans le modèle relationnel, le schéma comporte la description des différents schémas de relations /DEA83/, tandis que la base comporte les différentes relations décrites par le schéma, relations elles-mêmes composées de n-uplets. Le modèle relationnel est cependant pauvre en sémantique, c'est pourquoi on lui préfère des modèles de type "entité-association" (EA) /CHE76/ faisant ressortir davantage cette sémantique. C'est une telle approche qui a été prise dans le projet TIGRE avec la définition d'un modèle de données généralisé /TIGR1/, /TIGR5/. De type EA, ce modèle incorpore également des concepts liés aux types abstraits pour la définition des objets complexes. Cependant la réalisation du SGBD proposant ce modèle s'appuie sur le SGBD relationnel MICROBE /FER81/, /NGU83/. A l'aide du modèle généralisé et du langage de définition et de manipulation (LAMBDA), nous pouvons définir deux types de données appelés types classe : entité et association auxquels sont associées des variables bases de données. Chacune de celles-ci donnant lieu à la création de nombreuses occurrences.

Dans les SGBD de type relationnel /STO76/, /CHA81/, /FER81/,

l'objet est la relation. On peut définir le schéma d'une relation, y insérer des n-uplets, en supprimer, en mettre à jour ou enfin, supprimer totalement la définition de la relation. Dans un SGBD généralisé, il faut étendre et adapter les notions propres au modèle entité-association. L'extension permet la prise en compte de données complexes à partir de constructeurs de types adaptés. Elle comprend également un ensemble d'opérateurs fournissant une capacité d'abstraction (généralisation, agrégation). Parmi les objets complexes, seuls les documents sont pris en compte par TIGRE. Le lecteur intéressé trouvera une étude détaillée du modèle de document proposé dans le cadre du projet TIGRE dans /TIGR3/, /ADI83/ et /TIGR6/. La forme de représentation choisie est l'arborescence.

1.3.1 Définition de données

Dans TIGRE, il existe plusieurs types : types de base, types construits et types classes.

Les types de base sont soit simples : entier, réel, booléen, chaîne de caractères, soit restreints : scalaire et intervalle. On les appelle restreints parce qu'ils spécifient implicitement une contrainte d'intégrité par rapport aux types de base sur lesquels ils sont définis.

Les types construits sont obtenus à partir des constructeurs suivants : "renommage", "enregistrement", "tableau" et

"document". Le constructeur de "renommage" offre la notion forte de type en permettant de renommer un type de base. Ainsi, la distinction entre des types sémantiquement différents définis sur le même type de base est rendu possible en utilisant ce constructeur. Par exemple, si l'on définit les types "renommés" :

```
    type age :  (0..120)
  et type num :  (0..120)
```

aucune opération (arithmétique, algébrique ou de comparaison) n'est possible entre un attribut défini sur le type age et un attribut défini sur le type num.

Les constructeurs "enregistrement" et "tableau" peuvent être illustrés à l'aide des énoncés de définition suivants en LAMBDA :

```
    type hist-travaux :  array(12) of string(20)
  et

    Type adresse : record
                numéro ; integer;
                rue, ville : string(20)
    end
```

Le constructeur document est utilisé pour la définition de ce qu'on appelle "document généralisé" :

```
    type unité : document
            structure
                case nature of
                    1 : texte
                    2 : image
                    3 : graphique
                    4 : formulaire
                  end
    end
```

```
type paragraphe : document
        structure
            contenu : list (1,*) of unité.structure
        end
end
```

En ce qui concerne le type classe, l'idée de base est de définir des ensembles d'objets et des associations entre ces ensembles. Ce niveau est similaire au modèle EA mais inclut les concepts de généralisation et d'agrégation. Une classe est un ensemble de variables ayant des constituants communs. A chacune de ces variables correspondra un ensemble de faits ou occurrences de cette variable. La notion de fait correspond donc à celle de n-uplet dans le modèle relationnel.

```
type T-EMPLOYE :  entity
   key numéro : integer end key
            nom     : string(20)
            salaire: argent
            adresse: adresse
            tâches : hist-travaux
            catégorie : (ingénieur, secrétaire, programmeur)
end
```

Le type association matérialise un lien sémantique entre deux ou plusieurs types entité :

```
type T-LOCALISAT : relationship
        between T-EMPLOYE : employe (1,1)
              and T-SITE : lieu-de-travail (1,*)
           depuis : date
end
```

En supposant le type classe T-SITE déjà défini

Il existe trois types de variables en LAMBDA : variables base de données, variables temporaires et variables de faits.

1) Une variable base de données est un moyen de nommer tous les faits d'un type classe qui sont stockés dans la base. L'instruction :

dbvar : EMPLOYE : T-EMPLOYE

déclare une variable de type T-EMPLOYE.

Actuellement il existe une seule variable base de données associées à un type classe.

2) Une variable temporaire de classe est un moyen de nommer le résultat d'une requête :

var EMP-PRODUCTIFS : T-EMPLOYE

Ceci évite de déclarer un autre type classe associé à cette variable. Les valeurs des variables temporaires ne sont pas stockées dans la base. Elles sont stockées dans la zone de travail d'un programme d'application

3) Une variable de faits est utilisée pour nommer un fait particulier d'un type classe. Sa valeur aussi est stockée dans la zone de travail

var UN-EMPLOYE : fact-of T-EMPLOYE

Des types classes dérivés peuvent être définis par des

opérations de généralisation ou d'agrégation sur les classes existantes. Les opérations de généralisation sont la spécialisation, l'union et l'intersection. La spécialisation permet la définition d'une hiérarchie de classes. Une sous-classe peut avoir ses propres constituants et hérite de ceux de la classe parent. Les faits d'une sous-classe sont ceux de la classe parent qui satisfont un prédicat :

```
type T-SECRETAIRE : specialisation-of T-EMPLOYE
                        where catégorie = 'secrétaire'
                        -
                        -
end
```

L'union de classes existantes produit une nouvelle classe dont les faits peuvent appartenir à l'une d'entre elles. La nouvelle classe peut avoir ses propres attributs :

```
type T-EMPLOYE-BUREAU : union-of T-SECRETAIRE
                        and  T-INGENIEUR
                         no-tel : integer
end
```

Dans le cas de l'intersection, les faits de la nouvelle classe doivent appartenir à l'une et l'autre des classes opérandes. L'agrégation d'entités produit un type entité à partir d'un ensemble de types d'entités arguments. Cette opération permet une modélisation plus naturelle de certaines données dont la complexité repose sur le concept d'agrégation de données plus simples. Par exemple, la définition :

```
type T-DOSSIER : entity-aggregate-of T-LETTRE(1,*)
                                     and  T-CONTRAT(1,1)
                 no-de-dossier : integer
                 responsable   : string(30)
end
```

Exprime qu'un dossier contient des lettres, des contrats et des constituants qui lui sont propres.

1.3.2 Manipulation des données

Comme nous venons de le voir, le modèle TIGRE utilise très fréquemment le concept de type. Ainsi la manipulation des données sera effectuée suivant des opérations élémentaires définies sur ces types. Les services de manipulation de données, offerts par le serveur BD, seront effectués par un interface spécial se trouvant sur les postes de travail.

La notation utilisée dans LAMBDA pour spécifier des opérations de manipulation est basée sur des expressions produisant des séquences de valeurs (une séquence de faits d'une classe). Ces expressions appelées expressions de production ont la forme suivante :

expression de valeur : expression de désignation

par exemple :

e.salaire : employe e

L'expression de désignation correspond à un chemin dans le graphe défini par les entités et les associations du schéma logique. Les variables ('e' dans l'exemple) sont limitées aux faits et

utilisées pour qualifier les constituants sélectionnés dans l'expression de valeurs (e.salaire). L'utilisation des variables permet de faire une nette distinction entre les expressions de valeurs et les chemins d'accès. L'imbrication des chemins et des expressions disparaît et l'utilisation des variables convient à l'intégration de LAMBDA dans un langage hôte tel que Pascal.

Une production de sélection peut résulter en une nouvelle classe qui peut avoir un nom, si c'est nécessaire. Il suffit de lui associer une variable :

```
TEMP := select e.nom, s.site
        from employé e of site s
          where e.categorie ='ingénieur'
          and  s.site = 'France'
end
```

La classe ainsi créée peut être ajoutée à la base de données par utilisation de la commande :

```
restore TEMP as INGENIEUR-FRANCAIS
```

Des extensions sont définies dans LAMBDA pour prendre en compte explicitement la structure de document. Un sous-langage est utilisé dans les expressions de valeurs pour nommer les différentes parties de la structure du document. Par exemple, l'énoncé :

```
SELECT contenu OF clause 3 OF corps OF c-texte-contrat
FROM contrat c PART-OF le-dossier OF employé e
```

WHERE e.nom = 'MARTIN'

permet de retrouver une partie du contrat de Martin : le contenu de la clause 3 du corps du contrat.

Les mises à jour des données sont effectuées à l'aide des commandes insert, delete et replace. Elles sont valables pour les entités et les associations.

Opérations de sélection et de mise à jour sont regroupées dans des transactions.

1.4 Interfaces utilisateurs

Plusieurs interfaces utilisateurs sont en cours de développement pour TIGRE :

i) Un premier interface est fourni par le langage de manipulation LAMBDA que nous venons de présenter brièvement.

ii) En ce qui concerne la manipulation des documents, un éditeur interactif est en cours de développement. Il offre à l'utilisateur une représentation graphique du document, en cours de création ou de modification, afin de lui rendre perceptible visuellement la structure, la présentation, les directives de traitement et le contenu informationnel du document.

L'éditeur vise également à faciliter la restitution homogène des documents et à décharger l'auteur des tâches fastidieuses et répétitives de présentation. Il propose donc des modèles de présentation automatique, adaptés aux différents schémas. Il laisse cependant toute liberté à l'auteur pour faire des choix de présentation personnalisée.

Il est souhaitable que chaque utilisateur soit autonome et évite ainsi de recourir à un site serveur pour les traitements particuliers qui lui sont propres. Les fonctions d'édition des documents sont donc réalisées sur les postes de travail.

iii) Le troisième interface sera fourni par le langage PROLOG. Celui-ci permet la modélisation d'applications développées sur TIGRE /TIGR7/, /TIGR12/, /OLI83/. L'approche adoptée est basée sur la définition en PROLOG des opérations de base sur les types et les classes de LAMBDA. Il est alors possible de manipuler en PROLOG les types de données les plus complexes de LAMBDA, y compris les généralisations et les agrégations de classe. Ceci permet d'adjoindre à TIGRE tous les mécanismes d'inférence associés à la programmation en logique. L'utilisation de PROLOG permet donc l'expression de :

- la sémantique associée au schéma conceptuel d'une application TIGRE.
- l'accès et la manipulation de données au niveau interne, qu'elles soient stockées indifféremment dans le SGBD relationnel MICROBE, sous-jacent à TIGRE ou dans une base de

clauses PROLOG.

-la définition des traitements et la mise en oeuvre d'une application au niveau externe.

1.5 Le système d'autorisation (SAGE)

Ce thème constitue l'objet de cette thèse, il sera introduit de façon plus détaillée au chapitre 2 et largement développé au chapitre 3. Ici nous nous contentons de dire que le système TIGRE étant appelé à gérer des données généralisées dans un contexte bureautique il nous a paru important :

-D'une part de limiter les utilisateurs aux employés de l'entreprise utilisant le système. Pour cela nous gérons données et utilisateurs de façon intégrée.

-D'autre part d'assurer la sécurité des données gérées par le système.

Le terme sécurité comporte en fait deux grands aspects :

-Authentification des utilisateurs : un utilisateur est réellement celui qu'il prétend être.

-Vérification de l'autorisation d'accès : l'utilisateur a effectivement le droit d'exécuter les opérations demandées sur les données référencées.

Dans SAGE nous nous intéressons uniquement à ce dernier aspect.

CHAPITRE II

MECANISMES D'AUTORISATION DANS LES SGBD CLASSIQUES

Dès l'apparition des premiers SCBD, des systèmes d'autorisation visant à assurer la sécurité et la confidentialité des informations stockées dans les bases de données, ont été proposés. Ces systèmes sont basés sur le principe qu'une BD est destinée à être exploitée par plusieurs usagers appartenant à des applications diverses. Ces utilisateurs, pour des raisons évidentes de confidentialité, veulent protéger leurs données et donc ne les rendre accessibles qu'aux utilisateurs en qui ils ont confiance. Typiquement, les utilisateurs ne doivent et ne peuvent pas tous accéder à toutes les données de la base. En outre, l'existence des informations auxquelles les utilisateurs n'ont pas accès devrait leur être cachée. Il doit donc exister, selon les utilisateurs, diverses vues ou perceptions de la BD. la notion de sous-schéma répond à ce besoin. A chaque application est associé un sous-schéma (vue) qui décrit les données auxquelles les usagers de l'application pourront avoir accès et les opérations qu'ils pourront exécuter sur ces données. Donc pour un utilisateur de l'application la BD ne contient que les informations vues par le sous-schéma associé.

Dans les premiers systèmes d'autorisation, l'accès à un sous-schéma se fait en général par un mot de passe. Sous-schémas et utilisateurs qui y accèdent sont définis statiquement par l'administrateur de la BD.

Dans ce qui suit, nous allons présenter l'évolution des systèmes d'autorisation en fonction de celle des SCBD. Nous

prendrons l'exemple des systèmes les plus connus tels que IMS (Information Management System) pour le modèle hiérarchique et CODASYL pour le modèle réseau. Nous insisterons plus particulièrement sur les mécanismes proposés par les SGBD relationnels INGRES et System R; ensuite nous présenterons rapidement les approches de WOOD et de BUSSOLATI. Pour chaque système présenté, nous donnerons les caractéristiques essentielles.

On distingue deux types de mécanismes selon qu'ils sont proposés pour les modèles hiérarchique et réseau ou pour le modèle relationnel.

2.1 Protection d'accès dans les modèles hiérarchiques et réseaux

Elle est basée essentiellement sur les notions de mot de passe, sous-schéma et gestion centralisée. Nous allons illustrer le fonctionnement des mécanismes proposés à l'aide de deux SGBD assez répandus IMS (Information Management System) et CODASYL-DBTG.

2.1.1 Protection dans IMS

Dans IMS la protection est assurée à deux niveaux :

-Niveau segment : Le segment est l'unité de protection dans le système. Vis-à-vis d'un programme P, un segment peut être déclaré "sensitif" ou "non sensitif". Seuls les

"segments sensitifs" (SENSEG) sont accessibles à P.

-<u>Niveau opérations</u> : Il est possible de définir sur les "segments sensitifs" les types d'opérations permises sur ces segments sensitifs à l'aide de l'option PROCOPT (Processing Options). Ces PROCOPT sont déclarées par l'administrateur de la base sous la forme :

G ------> GET = accès en lecture seulement

I ------> INSERT = insertion autorisée

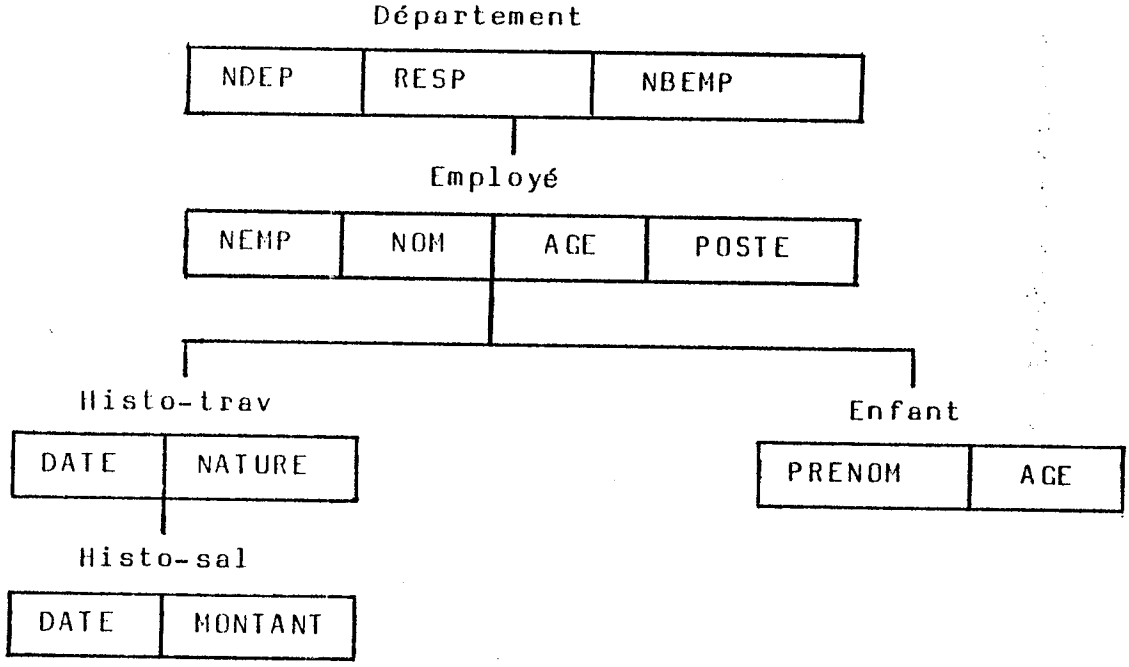
R ------> REPLACE = mise à jour autorisée

D ------> DELETE = suppression autorisée

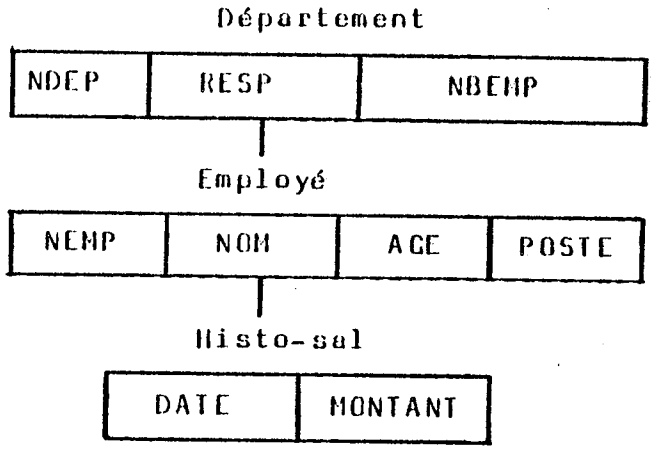
Ces options peuvent être combinées.

Illustrons la mise en oeuvre de ce mécanisme à l'aide d'un exemple :

Soit la structure hiérarchique :



Et soit la vue logique suivante (sous-schéma) :



Cette vue logique sera définie par la déclaration PCB (Program Communication Block) de la façon suivante :

1. PCB type = DB dbname = Entreprise
2. Senseg name = Département, parent = 0, PROCOPT = G
3. Senseg name = Employé, parent = Département, PROCOPT = G, I
4. Senseg name = Histo-trav, parent = Employé, PROCOPT = K
5. senseg name = Histo-sal, parent = Histo-trav, PROCOPT = G

End

Un programme auquel ce PCB est associé n'aura pas accès aux segments Histo-Trav et Enfant et ne pourra accéder aux segments Département et Histo-sal qu'en lecture et au segment Employé qu'en lecture et en insertion. Cependant, à cause de la structure hiérarchique du système, l'accès au segment Histo-sal ne peut se faire qu'en passant par le segment Histo-trav. L'option 'PROCOPT = K' associée au segment Histo-trav, permet au programme auquel ce PCB est associé d'utiliser le segment

Histo-trav pour accéder à Histo-sal, sans toutefois pouvoir prendre connaissance des données qu'il contient.

Si l'on veut restreindre la vue de manière à interdire à l'utilisateur d'avoir accès à la DATE dans Histo-trav, par exemple, alors immédiatement après la défintion de Histo-trav, dans le PCB, on ajoute :

SENFLD name = MONTANT, START = 2

L'option START = 2 indique la position de l'attribut dans le segment.

Tous les PCB d'un programme utilisateur sont regroupés dans un PSB (Program Specification Block). Lors de l'accès à une vue logique, le PCB correspondant est mis en mémoire dans une zone du programme utilisateur.

2.1.2 Protection dans CODASYL-DBTC

Une base de données CODASYL est décrite en termes d'enregistrements (records), de liens entre enregistrements (sets), d'areas et de sous-schémas. Différentes opérations peuvent être effectuées sur ces entités.

Tous les accès à la base se font à partir d'un programme qui fait référence à un sous-schéma. De cette manière l'utilisateur ne "voit" que l'information accessible via ce sous-schéma. En outre, le système DBTC assure la sécurité d'accès et la

confidentialité à travers le mécanisme de clés et verrous. La protection de la base est réalisée de la façon suivante :

- Les verrous sont spécifiés à la définition de la structure et peuvent être des littéraux, data-items ou des noms de procédures.
- Le programme d'application qui veut utiliser une structure de données doit fournir une clé.
- Si le verrou est un littéral il est comparé à la clé. Si c'est un attribut sa valeur est comparée à celle de la clé. Si verrou et clé sont égaux l'accés est autorisé.
- Dans le cas où le verrou est une procédure celle-ci prend la clé comme paramètre et donne un résultat booléen. L'accès est permis si le résultat est "vrai".
- La clé elle-même peut être un littéral, un attribut ou une procédure.

Un exemple d'utilisation de ce mécanisme pourrait être le suivant :

```
Schema :  RECORD NAME IS S
                 "
                 "
                 "
          ACCESS CONTROL LOCK FOR DELETE IS "Autre1"
                 "
                 "

Program
                 "
                 "
                 "
          PROCEDURE DIVISION
                 "
                 "
                 "
```

```
            DECLARATIVES
            ACCESS CONTROL KEY FOR DELETE IS "Autrel"
                  "
                  "
            END DECLARATIVES
                  "
                  "
                  "
            DELETE S
END
```

Dans cet exemple, à l'exécution de l'instruction DELETE S, la clé spécifiée dans la PROCEDURE DIVISION du programme et le verrou associé à l'enregistrement S à la définition de la structure sont comparés, comme ils sont égaux l'opération DELETE est autorisée sur S.

Un verrou peut aussi être spécifié comme une suite de littéraux, de data-items et/ou de procédures connectés par l'opérateur OR. Un tel verrou est satisfait dès qu'un des éléments est satisfait. On peut également avoir un seul verrou d'accès pour plusieurs opérations.

2.2 Protection dans les SGBD relationnels

Les caractéristiques des mécanismes proposés sont :

- Gestion dynamique
- Langage unifié de description, manipulation et autorisation
- Décentralisation de la fonction d'autorisation (pour la plupart des systèmes)
- Indépendance de l'accès aux données proprement dites

vis-à-vis du mot de passe

Les mécanismes les plus caractéristiques sont ceux proposés dans INGRES et le système R. Ces deux SGBD utilisent deux types de relation :

-Les relations de base qui sont efffectivement stockées .

-Les vues qui sont des relations dérivées soit des relations de base soit d'autres vues. Une vue est une relation virtuelle c'est-à-dire non stockée dans la base mais définie comme une requête QUEL ou SEQUEL. Seule sa définition est stockée. Il s'agit alors d'une fenêtre dynamique sur la BD.

Dans les deux systèmes, l'unité de protection est la relation. Cependant la notion de vue est utilisée de façon différente. Dans le système R une vue est le moyen employé pour assurer la protection au niveau d'un sous-ensemble d'attributs et de tuples d'une relation; l'élément protégé est indifféremment une vue ou une relation de base. Dans INGRES cette notion n'est pas utilisée en tant que telle pour gérer les autorisations d'accès. Elle permet seulement de renforcer la sécurité par modification de la requête utilisateur.

2.2.1 INGRES

La définition des contraintes de sécurité s'effectue en deux niveaux :

-Des opérations que l'usager est ou non autorisé à exécuter.

-De la définition des informations sur lesquelles l'utilisateur peut agir. Ces informations peuvent être définies à partir de l'expression de qualification en langage QUEL (vue).

Chaque contrainte de sécurité sera représentée par un tuple de la relation de protection (PROTECT) qui comporte les informations suivantes :

-Nom de l'utilisateur

-Commande

-Nom de la relation et variables entités associées.

-Liste des attributs qui seront transmis.

-Qualification

Le créateur d'une relation a tous les droits d'accès sur cette relation.

2.2.1.1 ALGORITHME DE CONTROLE D'ACCES <FOR74>

Soit R la requête émise par l'utilisateur U; pour chaque variable entité X l'algorithme suivant sera déroulé :

1. Si l'utilisateur demandant l'accès est le créateur de la relation alors autoriser l'accès. Fin
2. Trouver tous les attributs dans R pour la variable X.
   Soit Sx cet ensemble d'attributs.

3. Trouver pour l'utilisateur U tous les tuples tels que :
   - Commande = opération demandée dans R
   - La liste d'attributs contient Sx.

   S'il n'existe aucun tuple vérifiant ces critères alors refuser l'accès. Fin
4. Remplacer la partie qualification de la requête par :

   $Q \cap (Q1 \cup Q2 \cup ... \cup Qn)$.

2.2.1.2 Optimisation de l'algorithme

Elle découle de la constatation suivante : plus la liste des attributs transmis est grande, plus l'accès est restrictif. Par conséquent si la liste d'attributs associée à un tuple de la relation PROTECT contient celle associée à un autre tuple (pour une même relation et une même commande) les deux listes sont dites correctement imbriquées et le deuxième tuple est considéré comme redondant et ne sera pas pris en compte lors de la modification de la requête (étape 4 de l'algorithme).

2.2.1.3 Conclusion

INGRES est le premier système qui a utilisé une technique de sécurité d'accès non basée uniquement sur des schémas externes accessibles via des mots de passe. Ceci a permis d'affiner l'unité de protection (niveau relation) et d'avoir des droits propres à chaque usager.

Cependant l'utilisation d'une gestion centralisée des autorisations peut provoquer un goulot d'étranglement en environnement multi-utilisateurs. Mais le problème majeur de cette approche reste celui provoqué par l'algorithme de modification de requêtes, qui consiste à donner à l'utilisateur uniquement les informations auxquelles il a accès, après avoir modifié sa requête pour y intégrer les contraintes de sécurité. L'information reçue est donc incomplète sans que l'utilisateur s'en rende compte. S'il doit effectuer des calculs sur les données ainsi obtenues ou si sa requête comporte une fonction d'aggrégation (COUNT, AVG...) alors le résultat est invalide et rien ne signale cette anomalie <FOR74>.

2.2.2 Le système R

L'apport le plus original du système R, en matière de sécurité d'accès, par rapport aux systèmes antérieurs, est la répartition de la fonction d'autorisation entre les divers usagers de la BD : tout utilisateur qui crée une relation en devient le propriétaire et détermine les droits qu'il accorde aux autres usagers sur cette relation. Les mécanismes d'autorisation sont dynamiques c'est-à-dire qu'à tout moment le système peut prendre en compte de nouveaux droits ou au contraire certains droits peuvent être supprimés.

2.2.2.1 Les usagers

Les usagers pouvant être autorisés à accéder à la base sont de quatre types :

-Utilisateurs simples.

-Groupes d'utilisateurs.

-Un groupe spécial PUBLIC défini implicitement et représentant tous les utilisateurs de la BD.

-Administrateurs de la BD.

Initialement seuls les utilisateurs et le groupe PUBLIC avaient accès à la BD. Une évaluation du système R a conduit à l'introduction de la notion de groupe d'utilisateurs <CHA81>, <WIL82>. Les opérations pouvant être définies sur un groupe sont :

-Définition de groupe (DEFINE GROUP)

-Ajout d'éléments dans un groupe (INSERT INTO GROUP)

-Suppression d'éléments d'un groupe (DELETE FROM GROUP)

-Suppression de groupes (DROP GROUP)

Les groupes sont stockés dans une table (SYSGROUP) qui définit pour chacun d'eux, les utilisateurs qui en font partie et la date de leur entrée. Un groupe peut lui-même en contenir un autre.

2.2.2.2 Les objets

L'unité de protection est la relation (de base ou vue). La

protection s'étend également aux programmes d'application sur lesquels peut s'exercer la seule commande RUN c'est-à-dire le droit de les exécuter ou non.

2.2.2.3 Les privilèges

Les privilèges sur une relation sont les opérations autorisées par le langage SQL à savoir : lire un ou plusieurs tuples (SELECT), insérer de nouveaux tuples (INSERT), supprimer des tuples (DELETE), modifier la valeur d'un ou plusieurs constituants (UPDATE), ajouter de nouveaux constituants (ALTER) et créer de nouveaux index (INDEX).

2.2.2.4 Propagation des privilèges

La propagation des privilèges est assurée par l'introdution de deux nouvelles commandes dans le langage SQL.

-Commande de gratification de privilège (GRANT)

-Commande de révocation de privilège (REVOKE)

Les droits sur les relations ou les programmes d'application sont stockés dans des catalogues spéciaux acessibles via le langage SQL.

i)Commande de gratification de privilège

Tout privilège transmis peut l'être avec ou sans l'option de de transmissibilité (GRANT). La forme générale de la commande est la suivante:

GRANT <priv.> ON <objet> TO <liste-usag.> [WITH GRANT OPTION]

L'usager transmettant le privilège est appelé le donneur celui qui le reçoit le receveur. Un usager ne peut transmettre, sur un objet, que les privilèges dont il est en possession soit parce qu'il :

- est le créateur de cet objet
- a reçu ces privilèges avec la faculté de les transmettre
- appartient à un groupe possédant ces privilèges avec la faculté de les transmettre.

A tout moment,dans le catalogue d'autorisations, un usager apparaissant comme donneur d'un privilège doit aussi apparaître comme receveur de ce privilège avec l'option GRANT. Le créateur d'un objet est considéré à la fois comme donneur et receveur de tous les privilèges sur cet objet.

ii)Commande de révocation de privilège

La forme générale de la commande est la suivante :

REVOKE <priv.> ON <objet> FROM <liste-usag.>

Un usager ayant reçu un privilège peut le perdre à tout moment. Tout se passe comme si ce privilège ne lui a jamais été accordé. Le receveur perd ce privilège mais aussi les usagers à qui il a pu le transmettre le perdent à moins qu'ils ne l'aient reçu d'une autre source indépendante avant de l'avoir transmis à un autre usager.

Le mécanisme implique une révocation récursive le long de l'arbre (donneur, receveur). Chaque receveur devient à son tour donneur à l'étape suivante de l'algorithme. Pour appliquer ce mécanisme, il est nécessaire de garder trace, dans le catalogue, de l'instant de gratification; à chaque gratification de privilège est associée une estampille.

2.2.2.5 Extension à un environnement réparti (R*)

Dans les SGBD répartis les mécanismes de contrôle d'accès sont un peu plus compliqués que ceux utilisés dans les systèmes centralisés. En outre, à la protection des données localement doit s'ajouter le contrôle de la sécurité de communication. Les informations transmises par les lignes de communication doivent être protégées de telle sorte que les informations sensibles ne soient pas communiquées à des usagers non autorisés, durant leur transfert sur le réseau. Des techniques de cryptographie sont prévues pour cela.

D'autre part, les travaux entrepris dans les SGBDR en matière de contrôle d'accès sont moins abondants qu'en environnement centralisé. A notre connaissance seul le système R* <GRI80>, <WIL82> possède un mécanisme complet présenté comme une extension de celui utilisé dans le système R. Des propositions à ce sujet ont été également faites par Wood <WOO79> et Bussolati <BUS80>. Tous ces travaux n'abordent cependant pas l'aspect sécurité de communication.

Le système R* se présente comme un ensemble de SGBD relationnels, indépendants, coopérant pour partager des données de façon à conserver une indépendance maximale /DAN82/.

Pour préserver l'autonomie des sites , les privilèges de l'ensemble des utilisateurs locaux et distants, accordés sur un objet, sont enregistés sur le site où l'objet est stocké. En effet, le site de stockage de l'objet doit nécessairement être accédé lorsqu'un utilisateur veut avoir accès à l'objet, et il est donc préférable de garder l'ensemble des informations (notamment les privilèges) concernant l'objet sur le même site. De plus, en cas de révocation de privilège, il est indispensable d'avoir sur le même site la description de toute la chaîne des utilisateurs qui ont reçu un certain privilège, sans quoi l'autonomie des sites serait perdue.

Un usager se fait identifier par son nom concaténé à l'identificateur du noeud sur lequel il se trouve. Dans les

catalogues d'autorisation du système R on ajoute les informations concernant les noeuds de localisation :

-Du donneur

-Du receveur

-De la relation.

Dans R*, les différents sites ont le droit d'échanger de l'information; chacun de ces sites doit avoir été introduit au préalable à l'autre au travers d'une commande d'introduction de BD dans laquelle chaque site spécifie le mot de passe qu'il emploiera lorsqu'il ouvrira une session avec le site distant, ainsi que le mot de passe qui devra être employé par le site distant lorsqu'il ouvre une session avec ce site. Cette vérification des mots de passe a lieu une fois pour toute à l'ouverture de la session entre ces deux sites et n'a pas lieu à l'exécution de chaque requête nécessitant une communication entre ces sites.

2.2.3 Systèmes de WOOD et de BUSSOLATI

Ces deux systèmes d'autorisation sont structurés en divers niveaux de sécurité correspondant à l'architecure logique des BD selon le modèle ANSI/SPARC. On a ainsi trois niveaux de sécurité : interne, conceptuel et externe et des associations entre deux niveaux voisins. Cette démarche permet de définir des schémas de sécurité parallèles à ceux des données et de développer des schémas de protection propres aux applications.

Les règles de protection sont obtenues au niveau logique par une association entre quatre éléments fondamentaux :

U:-Usagers

O:-Objets

D:-Droits ou Privilèges

P:-Prédicats

Un quadruplet (U, O, D, P) signifie que l'utilisateur 'U' a le droit 'D' sur l'objet 'O' si la condition exprimée par le prédicat 'P' est vérifiée.

Comme dans le système R la fonction d'autorisation est répartie entre différents usagers de la BD. Mais ici il existe une distinction très nette entre le contrôle d'une ressource et son accès. En effet, il existe deux types de droits pouvant être accordés à un utilisateur sur un objet : (1) les droits d'accès (AC) qui permettent à l'utilisateur de lire un objet ou de le manipuler, (2) les droits d'administration (AD) qui donnent à l'utilisateur le droit de transmettre des AC.

2.3 Conclusion

Les divers systèmes que nous venons d'étudier brièvement se distinguent par un certain nombre de facteurs qui reflètent l'évolution des mécanismes mis en oeuvre pour assurer la protection des données gérées par un SGBD :

-Approche statique ou dynamique : en général, l'évolution

s'est faite de l'approche statique pour les modèles hiérarchique et réseau vers une approche dynamique depuis l'apparition du modèle relationnel.

-Granularité de protection : celle-ci devient de plus en plus fine. Cependant, s'il semble assez facile d'assurer la protection au niveau de l'attribut, la protection dépendant des données (niveau d'une occurrence ou d'un tuple) semble beaucoup plus coûteuse à mettre en oeuvre.

-Gestion des autorisations : elle peut être soit centralisée au niveau d'un seul utilisateur soit répartie entre plusieurs usagers. Là aussi l'évolution s'est opérée progressivement d'une gestion totalement centralisée vers une gestion totalement décentralisée.

-Mécanismes mis en oeuvre pour assurer une granularité dépendante de la valeur aussi fine que possible : vues, prédicats...

La figure suivante donne un tableau comparatif des divers systèmes que nous avons présentés, en fonction des facteurs précédents :

Système	Approche	Gestion	Unité de protection	Granularité de protection	Mécanismes de raffinement
IMS	statique	centra-lisée	segment	attribut	-
CODASYL	statique	centra-lisée	attribut	-	
INGRES	dynami-que	centra-lisée	relation	attribut tuple	modification de requêtes
système R	dynami-que	décent-ralisée	relation	attribut tuple	vues
de WOOD	dynami-que	décent-ralisée	classe de relations	attribut tuple	vues
de BUSSO-LATI	dynami-que	décent-ralisée	classe de relations	attribut tuple	prédicats

Figure 2 : comparaison des divers systèmes d'autorisation

Dans le paragraphe suivant nous donnons les principales caractéristiques de notre approche. Cette approche tient compte de l'évolution des mécanismes d'autorisation et les adapte à de nouveaux contextes pour tenir compte de l'évolution des SGBD.

2.4 Notre approche SAGE

SAGE (Système d'Autorisation GEnéralisé) est développé dans le contexte de bases de données généralisées, dans un environnement bureautique. Et, à notre connaissance, il n'existe pas de système d'autorisation permettant l'expression des contraintes de confidentialité des données stockées dans de

telles bases. Les principaux objectifs de notre approche sont les suivants :

2.4.1 Protection des données généralisées

Mise à part la protection des données alphanumériques classiques, SAGE assure la protection de données structurées telles que les documents généralisés (cf. chapitre 1).

2.4.2 Modélisation de l'environnement par le système

Dans les systèmes de protection d'accès qui existent aujourd'hui seuls les privilèges d'accès des utilisateurs aux données sont décrits; les dépendances hiérarchiques des différents utlisateurs ne sont pas prises en compte; or dans la plupart des entreprises, ces dépendances hiérarchiques ont des implications directes sur les pouvoirs des différents utilisateurs. Les applications de type bureautique sont justement développés par des entreprises, c'est-à-dire, des organisations structurées hiérachiquement par nature. D'autre part, dans un tel environnement les utilisateurs potentiels du système informatique ne peuvent être qu'un sous-ensemble des employés de l'entreprise. Il nous a donc paru important de décrire de façon intégrée les données et les utilisateurs c'est-à-dire de décrire au niveau du modèle l'environnement dans lequel la base est utilisée. La structure qui s'impose dans le cas de la modélisation d'une entreprise est, à notre avis, la

structure arborescente.

2.4.3 Décentralistion de la fonction d'autorisation

Dans les SGBD hiérarchiques et réseaux ainsi que dans le système INGRES il existe un seul administrateur, appelé ABD, qui gère toutes les autorisations. Il définit les utilisateurs de la base et leurs droits d'accès. Les travaux entrepris par la suite dans le cadre des SGBD relationnels ont pour objectif la décentralisation totale de la fonction d'autorisation /WIL82/, /BUS81/. A la limite n'importe quel utilisateur du système peut être nommé administrateur; de plus, plusieurs administrateurs peuvent coexister dans la BD. puisqu'ils gèrent certaines données. Enfin, dans le système R, un utilisateur gère non seulement les relations qu'il crée mais il a aussi le droit de transmettre les privilèges reçus sur d'autres relations, pour autant que ces privilèges lui aient été transmis avec le droit de transmissibilité. Ceci peut conduire à l'autre extrême : un utilisateur peut obtenir par transmissions successives et de façon tout à fait légale, un privilège que le créateur de l'objet aurait voulu lui interdire. Dans le système R, il n'ya pas de possibilité d'interdire à un utilisateur le droit d'avoir tel ou tel privilège; la gestion de ces interdictions est considérée comme extrêmement coûteuse.

Dans SAGE, nous avons opté pour une solution intermédiaire : comme dans System R, tout créateur d'un objet en devient le

propriétaire. Mais dans notre système le propriétaire d'un objet sera le seul à pouvoir transmettre des privilèges sur cet objet. Ceci permet de mieux contrôler la diffusion de l'information et de proposer des algorithmes de propagation et de révocation des privilèges, très simples. Pour des raisons de flexibité du système, un objet peut changer de propriétaire au cours de son existence. Ceci se fera sur demande explicite du propriétaire courant qui a pour ce faire une commande à sa disposition. Cependant à tout moment un objet n'a qu'un seul propriétaire.

2.4.4 Rôle de l'ABD

Dans SACE, l'administrateur ne gère que les utilisateurs et n'a aucun privilège implicite sur les objets. Il a cependant le pouvoir d'accorder aux usagers le privilège de création ou de le leur retirer. L'unicité de l'administrateur se justifie par le fait que son rôle se réduit à la gestion informatique de l'entreprise. Nous verrons au chapitre suivant qu'une originalité de notre système est de considérer ce rôle comme un privilège spécial.

2.4.5 Transmission des privilèges

La structure hiérarchique des utilisateurs et leur description par le modèle confère à la propagation des privilèges certains aspects particuliers. En effet, implicitement tous les supérieurs hiérarchiques du créateur d'un objet ont le droit de

lecture sur cet objet. Cependant, et toujours dans le souci d'assurer une protection efficace, SAGE met à la disposition du créateur une commande lui permettant d'échapper à cette contrainte.

Ce privilège implicite n'est accordé que sur les objets créés par un utilisateur et non sur ceux pour lesquels il reçoit des privilèges. En effet, le créateur d'un objet peut transmettre explicitement des privilèges à n'importe quel utilisateur dans l'arborescence. Les supérieurs de ce dernier n'auront aucun droit implicite sur cet objet.

2.4.6 Révocation implicite par suppression d'utilisateurs

La suppression d'un poste de l'entreprise entraîne implicitement celle de tous les privilèges liés à ce poste. Le code correspondant est supprimé de tous les catalogues où il apparaît. Il est également supprimé de tous les groupes dont il est membre. Cette caractéristique de SAGE est importante. En effet dans un système tel que System R la seule façon de supprimer des privilèges des divers catalogues ou des membres d'un groupe est d'émettre une requête d'autorisation explicite.

2.4.7 Autres particularités de SAGE

Au cours de l'étude des divers systèmes de protection nous avons vu qu'un système d'autorisation est étroitement lié au SGBD

qui l'implante. On construit un système d'autorisation pour un SGBD donné. Dans SAGE, nous avons cherché à éviter cela. C'est pour cette raison que nous lui avons assigné les trois objectifs suivants :

i) Conception modulaire qui le rend indépendant de tout SGBD.

ii) Transportabilité : SAGE est programmé en pascal MULTICS. Nous avons cependant veillé à utiliser le plus possible le pascal standard ce qui permet de le transporter très facilement sur n'importe quelle machine disposant d'un compilateur pascal.

iii) Mise en oeuvre facile : soit à partir d'un langage de haut niveau pour les commandes de propagation et de révocation des privilèges. Soit implicitement pour les opérations concernant le contrôle d'accès.

SAGE est conçu dans le cadre du projet TIGRE, aussi la terminologie employée est celle du modèle entité-association. Cependant il est paramétré de telle sorte qu'il puisse être utilisé par tous les SGBD de type relationnel. En ce qui concerne la terminologie il suffit de remplacer le terme 'entité' par 'relation' et de considérer que toutes les variables sont de type entité et n'ont que des constituants élémentaires.

CHAPITRE III

DESCRIPTION DE SACE

SAGE est un système qui assure un certain nombre de fonctions dont les principales sont les suivantes :

-Gestion des utilisateurs et description, dans la base de données, de l'entreprise utilisant le SGBD. La gestion des utilisateurs se fait de deux manières : (1) par leur apparition dans l'arborescence de modélisation, (2) par leur appartenance à un groupe c'est-à-dire un ensemble d'utilisateurs ayant les mêmes privilèges sur les mêmes objets. Cependant tout utilisateur apparaissant en tant que membre d'un groupe doit aussi apparaître dans l'arborescence.

-Gestion dynamique des autorisations c'est-à-dire des privilèges accordés à tout utilisateur sur les objets de la base. Ainsi, toute demande d'accès non autorisée doit être rejetée.

-Prise en compte de toute modification intervenant dans l'arborescence de modélisation, en perticulier dans le cas de la restructuration de l'organisation de l'entreprise.

-Répercussion implicite, sur les catalogues d'autorisation, de toute modification intervenant dans l'entreprise : suppression d'un poste, changement d'affectation d'un utilisateur etc.

Ces fonctions seront décrites en détail dans la suite de ce chapitre.

3.1. Gestion des utilisateurs

Traditionnellement le terme utilisateur s'applique à toute personne, groupe de personnes ou programme ayant le droit d'accès à la base de données /BUS81/, /WIL82/. Dans notre modèle d'autorisation, nous supposons qu'un système est créé pour répondre aux besoins d'une entreprise. Par conséquent, un utilisateur sera un employé de l'entreprise utilisant le système.

Ainsi l'ensemble des utilisateurs de la BDG est un sous-ensemble des employés de l'entreprise. Celle-ci est représentée par une arborescence reflétant son organisation hiérarchique. Chaque noeud correspond à un poste occupé dans l'entreprise ou plus exactement au rôle joué par l'utilisateur dans celle-ci.

De par sa position hiérarchique, un utilisateur a en principe un droit de lecture sur tous les objets, créés par ses subordonnés, sauf si ceux-ci spécifient le contraire (cf § 4).

Dans cette hiérarchie, et par définition, l'utilisateur associé au noeud-racine correspond au responsable de l'organisation modélisée : il a de ce fait le droit de lecture sur tous les objets de la base. Cette hypothèse est faite dans l'objectif d'avoir un système d'autorisation qui correspond autant que possible à la réalité.

Généralement l'expression administrateur de base de données (ABD) désigne le rôle qu'assume une (ou plusieurs) personne(s) pour gérer les objets de la base. Une particularité de notre système est de considérer que ce rôle est donné comme un privilège à un et un seul utilisateur, à un instant donné. Cependant cet utilisateur peut changer au cours du temps. Dans la suite, l'utilisateur en possession du privilège ABD sera appelé ABD. Il est doté de certains droits spéciaux sur la gestion des utilisateurs qu'il ne peut transmettre à aucun autre usager (sauf en cas de transfert du privilège ABD) tels que :

- Création d'un nouvel utilisateur
- Suppression d'un utilisateur
- Modification du code d'un utilisateur
- Toutes les opérations possibles sur les groupes d'utilisateurs (cf § 2.3)

Pour illustrer cette représentation de l'entreprise et faire ressortir le rôle joué par le responsable de l'organisation et l'ABD, considérons, par exemple, un centre de recherche en informatique. Il est géré par un directeur et est constitué d'un ensemble d'équipes (Langage, CAO, Logiciel de Base etc.). Chacune des équipes étant constituée à son tour d'un certain nombre de projets, par exemple l'équipe Logiciel de Base sera constituée des projets : BD, Réseau, Système d'Exploitation etc. Chaque projet comportera un nombre déterminé de chercheurs. Des secrétaires peuvent apparaître à n'importe quel niveau :

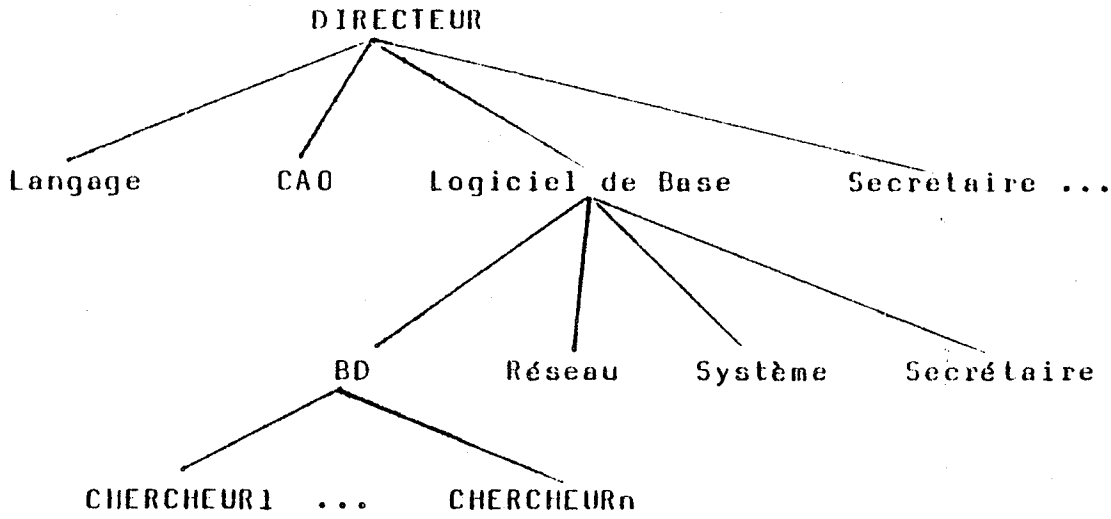


Figure 2 : exemple de structure hiérarchisée

Le directeur est considéré comme le responsable du centre de recherche. Un chef d'équipe aura, par définition, le droit de regard sur tous les objets créés par les chercheurs de son équipe (cf § 4.1). L'ABD sera l'un des utilisateurs apparaissant dans l'arborescence. Il sera chargé de la gestion des autres utilisateurs sans avoir aucun privilège spécial sur les données.

3.1.1 Representation interne

La gestion de l'arborescence est assurée par l'ABD qui affecte, à chaque utilisateur, un code hiérarchique correspondant au poste qu'il occupe dans l'entreprise. Le codage des utilisateurs est introduit pour faciliter la gestion des privilèges et pour des raisons évidentes d'économie de place mémoire. Toute l'arborescence représentant la structure hiérarchique de l'entreprise est ainsi codée. Ceci pourrait

être effectué par une fonction très simple et telle que les codes attribués permettent de vérifier les caractéristiques suivantes :

-Ils reflètent les divers niveaux de l'arborescence :

-Niveau 1 = code à 1 chiffre.

-Niveau 2 = code à 2 chiffres.

- "

-Niveau n = code à n chiffres.

Donc à partir de chaque code on doit pouvoir déterminer le niveau de l'utilisateur (nombre de chiffres dans le code).

-Un groupement horizontal des utilisateurs donne tous les utilisateurs d'un même niveau : les services ou les départements de l'entreprise par exemple.

-Un groupement vertical donne des utilisateurs ayant les mêmes ascendants dans l'arborescence ou tous les ascendants d'un utilisateur.

Le calcul du code se fait de la façon suivante : le code à un certain niveau n se déduit du code précédent dans la hiérarchie par une fonction appelée <u>fonction de codage</u> et pouvant être définie ainsi:

$$F(0) = 0$$

$$F(n) = F(n-1) * 10 + i$$

$$2 <= n <= 9$$

$$i = 1, 9$$

Chaque code est alors unique et identifie un seul utilisateur.

Avec la fonction proposée, un noeud peut avoir au plus 9 fils. dans le cas où l'on a besoin d'un plus grand nombre de fils il suffit de paramétrer la fonction F de façon à pouvoir passer facilement du système décimal au système hexadécimal, par exemple. Il est également possible d'ajouter d'autres paramètres qui permettront d'avoir un nombre quelconque de fils. Nous donnons la fonction F à titre indicatif.

Dans une entreprise, il est courant qu'une même personne occupe plusieurs postes, chacun lui donnant le droit d'accéder à certaines informations; dans ce cas, l'utilisateur sera représenté plusieurs fois et pour chaque poste occupé il aura un code.

A partir de chaque code il est possible de déterminer tous les supérieurs hiérarchiques de l'utilisateur. Pratiquement, ceci est effectué en deux phases. A chaque phase est associée une fonction :

- Dans la première phase la fonction H(F(n)) fait correspondre au code F(n) un niveau dans l'arborescence. Le niveau sera égal au nombre de chiffres dans le code.
- Une fois le niveau déterminé, la fonction G donnera tous les ascendants de l'utilisateur de code F(n), dans l'arborescence.

$$G(j) = \lfloor F(n) / 10 ** (n-j) \rfloor$$
$$j = 1, n-1$$

Chaque valeur de j correspondra à un niveau. G(j) sera le code de l'ascendant associé à ce niveau.

La fonction G sera utilisée par l'algorithme de contrôle d'autorisation. En effet, nous avons vu (cf § 2.1) qu'un utilisateur peut avoir le droit de regard sur les objets créés par un autre s'il en est le supérieur hiérarchique. Etant donné un utilisateur défini par son code, la fonction G permettra d'avoir tous ses supérieurs hiérarchiques et par conséquent de vérifier si un utilisateur donné a, ou non, le droit de regard sur les objets auxquels il veut accéder. Il suffit pour cela de chercher le supérieur du créateur de l'objet au même niveau que l'utilisateur qui demande l'accès et de comparer les deux codes. En cas d'égalité des codes l'utilisateur aura le droit d'accès en lecture à l'objet.

A l'aide de la fonction proposée, l'arborescence de l'exemple précédent serait représentée ainsi :

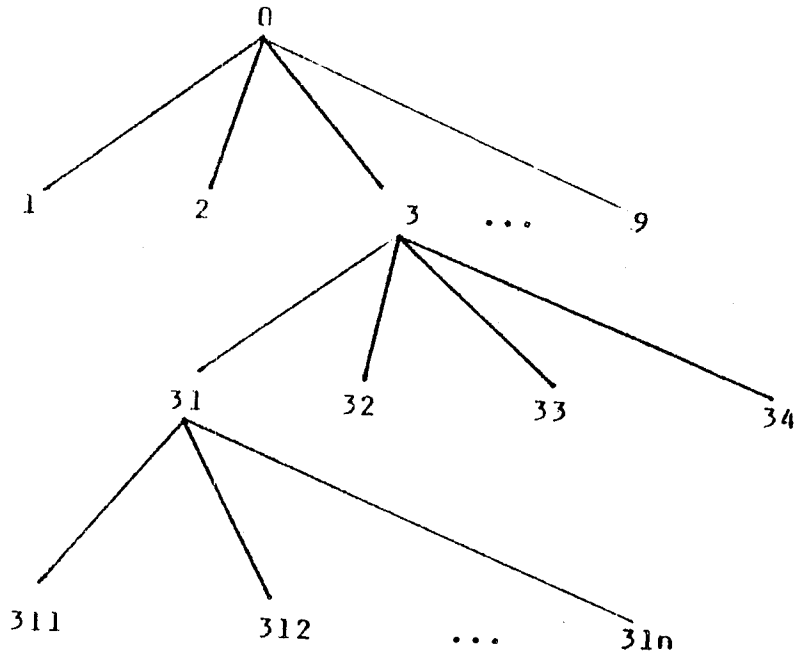


Figure 4 : représentation codée de l'arborescence

Dans cette représentation arborescente le directeur du centre a un code spécial ne déterminant pas sa position dans l'arborescence. Ce code sera donné par la valeur initiale de la fonction de codage (ici 0). Etant donné que l'ABD est un privilège, il ne lui sera associé aucun code spécial. A un moment donné, il correspond à un utilisateur spécifique.

Une fois codée, l'arborescence pourrait être représentée dans la base par un catalogue où chaque ligne décrit un utilisateur en donnant :

- son nom
- son code
- le poste occupé dans l'entreprise.

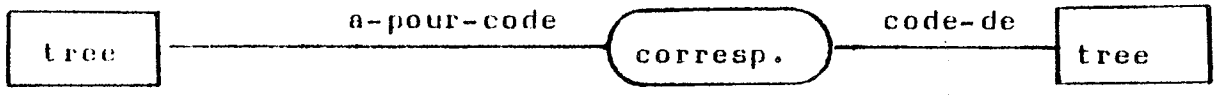
- le fils (premier descendant dans l'arborescence)
- le frère (utilisateur suivant de même niveau et de même père)
- suivant (code suivant si l'utilisateur occupe plusieurs postes)
- création (droit de créer ou non des objets dans la BD). Si l'utilisateur a le droit de créer des objets cette rubrique indique le type des objets qu'il peut créer : entités ou associations.

Ce catalogue doit être géré comme une arborescence, si possible de manière récursive. Actuellement, dans TIGRE qui est une extension du modèle entité-association RM/T /COD79/, le type arbre n'existe pas. Le type qui s'en approche le plus est le type document. Cependant à un document est associée une sémantique qui ne saurait être celle de n'importe quelle arborescence. Pour donner au concepteur du SGBD un formalisme de description adéquat et général, nous représenterons le catalogue comme une entité dans le langage de définition du modèle TIGRE, LAMBDA /TIGR5/ :

```
type tree : entity
        key nom : string (1..30) end key;
                code : integer;
                poste : string (1..30);
                fils : integer;
                frère : integer;
                suivant : integer;
                création : char;
                abd      : integer
end.
```

dbvar ARBRE : tree;

Cette déclaration permet de définir la variable base de données ARBRE comme une entité de type tree décrit précédemment. ARBRE sera le catalogue représentant l'organisation de l'entreprise. L'accès à ARBRE se fera sur le nom de l'utilisateur. Pour pouvoir associer à chaque nom d'utilisateur le (ou les) code(s) correspondant(s), on établit l'association suivante, sur l'entité arbre :



La définition de cette association se fera de la façon suivante :

```
type correspondance : relationship
        between
                tree : a-pour-code (1, *)
        and
                tree : code-de (1, 1)
end
```

dbvar nomcode : correspondance

La notation (1,1) indique qu'un code est associé à un utilisateur unique.

La notation (1,*) indique qu'à un même utilisateur peuvent correspondre un ou plusieurs codes.

L'arborescence est fournie au système par l'ABD.

Au fur et à mesure de l'introduction des informations dans la

table ARBRE, la fonction de codage associe un code à chaque utilisateur. Ce code ne peut être connu que du seul ABD et de l'utilisateur auquel il est associé et est géré par le système.

Un module propre au système de sécurité gère la création et la manipulation de cette arborescence.

3.1.2 Les groupes d'utilisateurs

Au lieu de transmettre des droits individuellement à chaque utilisateur il pourrait être plus avantageux de les transmettre collectivement à tout un sous-ensemble d'utilisateurs à la fois, ce sous-ensemble sera appelé groupe. Si l'on reprend l'exemple précédent un groupe pourrait être constitué de chercheurs appartenant à la même équipe, travaillant sur le même sujet etc. On aura deux types de groupes :

-Ceux qui sont définis implicitement par le niveau ou l'ascendance dans l'arborescence. Par exemple le chef du projet BD et tous les chercheurs appartenant à celui-ci. Dans ce cas un groupe pourra être considéré comme un sous-arbre de l'arborescence initiale. Il suffit alors de donner dans la commande de création du groupe (cf. § 3.1.3) la racine du sous-arbre correspondant et le système associera implicitement au groupe, tous les utilisateurs qui le constituent.

-Ceux qui sont définis explicitement par l'ABD selon un

critère donné par exemple tous les chercheurs travaillant sur les problèmes de confidentialité. Ces utilisateurs ne sont pas forcément à un même niveau hiérarchique ni situés dans un même sous-arbre et ils peuvent donc se trouver n'importe où dans l'arborescence.

Les groupes seront considérés comme des utilisateurs d'un type particulier et n'apparaissent pas dans l'arborescence. Cependant pour rester cohérent avec le mécanisme de gestion des autorisations proposé (cf § 4), où chaque utilisateur est désigné par un code, on affecte à ces groupes un code indépendant de celui attribué par la fonction de codage. Ce code désignera le groupe dans le catalogue d'autorisation.

3.1.3 Opérations sur les groupes

Les groupes sont représentés dans une table appelée GROUPE. En LAMBDA, la définition de la table GROUPE sera exprimée ainsi :

```
type T-GROUPE : entity
          nom : string (1..30);
          code : integer;
          util : array (1..*) of integer
end.

dbvar GROUPE :  T-GROUPE
```

permet de déclarer l'entité GROUPE de type T-GROUPE.

la création d'un groupe se fera à l'aide de la commande :

DEFINE GROUP <nom-du-groupe> [AS <liste-d'utilisateurs>]

où :

<liste-d'utilisateurs> sera l'ensemble d'utilisateurs appartenant au groupe au moment de sa création.

Chaque groupe ainsi défini sera représenté dans la table GROUPE comportant :

-son nom

-son code

-la liste des utilisateurs qu'il contient

Un groupe peut être détruit à tout moment par la commande :

DROP GROUP <nom-du-groupe>

Il est également possible d'ajouter ou de supprimer, à n'importe quel moment, un ou plusieurs utilisateurs à un groupe. Pour ce faire l'ABD dispose des commandes :

ADD <liste-d'utilisateurs> TO GROUP <nom-du-groupe>

et

REMOVE <liste-d'utilisateurs> FROM GROUP <nom-du-groupe>

Si l'utilisateur que l'on veut ajouter existe déjà, l'ajout n'aura pas lieu. Cependant aucun message d'erreur ne sera envoyé à l'ABD.

Il existe d'autres commandes permettant de lancer d'autres opérations sur les groupes telles que :

MERGE GROUP <nom-du-groupe1> <nom-du-groupe2>

qui assure la fusion de deux groupes. Le deuxième groupe est alors supprimé.

MOVE <liste-d'utilisateurs> FROM GROUP <nom-du-groupe1> TO <nom-du-groupe2>

qui permet de transférer des utilisateurs d'un groupe vers un autre. Cette opération revient en fait à une suppression suivie d'une insertion.

Seul l'ABD est habilité à exécuter ces opérations. Le changement d'ABD n'a aucune répercussion sur les groupes existants. Ceux-ci ne peuvent en effet être modifiés que par des opérations exécutées explicitement par l'ABD.

3.1.4 Conclusion

La gestion des utilisateurs proposée permet d'avoir rapidement :

- Le poste occupé par chaque utilisateur, dans l'entreprise.
- Toutes les branches d'activité d'un utilisateur.
- Tous les utilisateurs d'un même service, ou ayant le même domaine d'intérêt, ou les mêmes obligations, etc.

3.2. Les objets

Pour simplifier la discussion, on désignera par objet indifféremment un objet conceptuel (entité et association) ou une occurrence de celui-ci quand il s'agit d'un objet comportant un attribut de type document. Dans notre système, il s'agira d'occurrences d'entité ou d'association. Lorsque l'objet comporte un constituant de type document on parlera d'objet complexe.

Le mécanisme d'autorisation que nous proposons s'applique principalement aux objets non seulement pour leurs aspects de définition (schémas, type) mais aussi pour leurs aspects de manipulation (occurrences).

Dans /TIGR3/, un document virtuel est défini comme étant la réalisation d'un document c'est-à-dire l'association d'informations textuelles, des images ou des formules et d'informations sémantiques. Vis-à-vis d'un objet, la confidentialité peut être considérée comme une information sémantique associée à cet objet, au même titre que les contraintes d'intégrité, par exemple. La confidentialité sera donc associée au document virtuel. Cependant les deux problèmes ne peuvent être traités de la même façon. En effet les contraintes d'intégrité sont les mêmes quel que soit l'utilisateur. Les contraintes de confidentialité varient d'un utilisateur à l'autre. En particulier, les restrictions imposées sur l'accès ne sont pas identiques pour tous les utilisateurs.

Si dans le contexte des SGBD relationnels les travaux concernant la confidentialité des relations ont été nombreux /GRI76/, /DOW79/, /WIL82/ etc. il n'en est pas de même en ce qui concerne les documents. Un de nos principaux objectifs est de prendre en compte ce dernier type d'objet.

3.3 Gestion des privilèges

3.3.1 Privilèges sur les objets

On appelle privilège le droit qu'a un utilisateur d'exécuter une opération particulière sur un objet (définition, manipulation). Les opérations de manipulation à considérer ici sont : suppression, consultation, création, mise à jour. Dans le langage LAMBDA de TIGRE, les opérations sont mises en oeuvre par les commandes suivantes :

- SELECT : pour l'accès en lecture
- INSERT : pour l'ajout de nouvelles occurrences d'un type donné
- DELETE : pour la suppression d'occurrences
- REPLACE : pour la mise à jour d'occurrences.

Notre mécanisme d'autorisation SACE associe à chacune de ces opérations le privilège correspondant. En outre, nous définissons deux autres privilèges particuliers pouvant être accordés aux utilisateurs. Par ailleurs les commandes de création, de destruction et de suppression du langage LAMBDA sont gérées de façon particulières au niveau des privilèges.

- Le privilège ABD qui permet à un utilisateur de gérer toute l'organisation hiérarchisée c'est-à-dire les utilisateurs. Initialement ce privilège appartient au responsable de l'entreprise. Celui-ci peut l'octroyer ultérieurement à n'importe quel utilisateur. Cependant à tout moment il

existe un seul ABD dans le système. La possession du privilège ABD n'implique aucun droit d'accès aux objets de la base.

-Le privilège d'interdiction (FORBID), mis à la disposition du propriétaire d'un objet. Cela lui permet d'interdire à certains de ses supérieurs, dans l'arborescence, d'accéder en lecture à cet objet. Le droit de lecture leur est en principe octroyé implicitement de par leur position hiérarchique (cf § 2). Mais dans certains cas on a besoin d'échapper à cette contrainte imposée par la structure hiérarchique pour des raisons de confidentialité. Ceci est réalisé à l'aide de la commande FORBID.

-Le privilège de création qui donne à son possesseur le droit de créer des objets conceptuels c'est-à-dire des types classes et des variables BD associées à ces types /TIGR5/. Ce privilège n'est pas transmissible. Il peut être octroyé ou retiré par l'ABD. Si le privilège de création est retiré à l'utilisateur, cela n'a aucune répercussion sur les variables et les types crées par celui-ci. Cependant les variables doivent avoir un propriétaire à tout moment, alors que la notion de propriété n'est pas liée à un type classe. Une fois créé, un type peut être utilisé par tout usager ayant le droit de création, pour définir de nouveaux types ou créer des variables. Un type ne peut en aucun cas être supprimé. Initialement, le créateur d'une variable est considéré comme son propriétaire et possède tous les autres privilèges sur cette variable.

Cependant, en ce qui concerne les variables associées à des types classe comportant des constituants complexes, le droit de propriété est également associé à une occurrence de la variable BD. Ainsi, dans le cas d'une variable ayant des constituants de type document, les propriétaires potentiels sont tous les utilisateurs auxquels le créateur de la variable BD accorde le privilège d'insertion. L'exécution de la commande INSERT correspond à la création d'un objet c'est-à-dire d'une occurrence de la variable. L'insertion de parties dans le constituant de type document déjà existant (paragraphes, section...) sera considérée comme une mise à jour de ce document; c'est alors la commande REPLACE qui esr utilisée.

Le privilège INSERT peut être accordé par le créateur d'un objet avec différentes options :

- 'p' : l'utilisateur recevant le privilège devient propriétaire de toutes les occurrences qu'il crée, sauf de celles pour lesquelles il cède le droit de propriété, soit au créateur de la variable associée aux occurrences, soit à un autre usager. Un tel utilisateur est donc en possession de tous les autres privilèges, qu'il pourra éventuellement transmettre à d'autres utilisateurs. Seul le privilège de suppression ne sera pas transmissible.

-'N' : l'utilisateur a le droit de créer de nouveaux objets mais c'est le créateur de la variable qui en devient propriétaire. Il pourra recevoir ultérieurement certains privilèges sur ces objets comme n'importe quel autre utilisateur.

-'T' : quand il s'agit d'une variable composée uniquement de constituants élémentaires, l'utilisateur recevant le privilège d'insertion n'a aucun droit de propriété sur les occurrences qu'il crée. Les droits d'accès qu'il reçoit sont valables sur toutes les occurrences de la variable en respectant évidemment les contraintes qui lui sont imposées. Dans ce cas le propriétaire peut transmettre également le privilège de suppression.

La transmission du privilège d'insertion peut être représentée par le catalogue CAT-INSERT suivant :

COD1	COD2	VAR	INSERT

où COD1 représente le code de l'utilisateur recevant le privilège d'insertion, COD2 le code du créateur de la variable BD sur laquelle est octroyé ce privilège, VAR est le nom de la variable BD. La rubrique INSERT aura la valeur 'P', 'N' ou 'T' selon le

cas.

Le propriétaire d'une variable a tous les droits d'accès sur les occurrences de celles-ci. Si l'utilisateur qui exécute la commande INSERT occupe une position hiérarchique, dans l'arborescence, supérieure à celle du créateur alors la variable BD peut changer de propriétaire (cf § 3.1). Ceci peut être illustré à l'aide de l'exemple suivant : dans un environnement bureautique une secrétaire crée une variable complexe (exécution de la commande CREATE), elle en est ainsi la propriétaire. Par la suite, le responsable du service pourra créer une occurrence de cette variable, y mettre des informations confidentielles et interdire l'accès de cet objet à la secrétaire ou lui permettre d'accéder seulement à certaines parties de celui-ci. Dans ce cas, la secrétaire doit lui céder le droit de propriété et c'est lui qui se charge désormais de la transmission des privilèges sur cet objet ainsi que sur la variable BD.

-<u>Les privilèges de destruction et de suppression</u> : seul le propriétaire d'une variable BD a le droit de la détruire à un moment donné. Ceci est effectué à l'aide de la commande DROP. La destruction d'une variable entraîne celle de tous les privilèges qui ont été accordés sur cette variable et ses occurrences dans le cas où elle contient des constituants complexes. La suppression d'un objet élémentaire (exécution de la commande DELETE) n'a aucune répercussion sur les privilèges octroyés. Rappelons qu'un objet élémentaire correspond à un tuple dans le modèle relationnel. Par contre la suppression d'un

objet complexe entraîne celle de tous les privilèges octroyés sur cet objet.

Dans tous les cas, seul le propriétaire d'un objet (variable ou objet complexe) peut le détruire car il est le seul à pouvoir retirer aux autres utilisateurs les privilèges octroyés sur cet objet. Si le créateur d'une occurrence complexe en devient le propriétaire il sera le seul à pouvoir exécuter la commande DELETE sur cette occurrence. Cependant, nous avons vu que quels que soint les propriétaires des occurrences d'une variable, le propriétaire de celle-ci peut la détruire à n'importe quel moment en exécutant la commande DROP. Pour rester cohérents avec notre mécanisme il serait important d'imposer une contrainte sur le privilège DROP : avant d'exécuter la commande DROP sur une variable, le propriétaire doit retirer le droit de propriété accordé aux autres utilisateurs sur toutes les occurrences de la variable à détruire.

Remarque

Dans ce qui précède nous n'avons pas considéré la cas où un document est partagé c'est-à-dire le cas où une ou plusieurs parties d'un document peuvent être intégrées dans d'autres documents. Le partage peut se faire en lecture et/ou en écriture.

3.3.2 Raffinement

La granularité de protection est affinée (niveau d'une occurrence d'entité, de certaines parties d'un paragraphe ou d'une clause etc.) par l'utilisation d'un système de prédicats. Un prédicat exprime une ou plusieurs conditions limitant les droits qu'un utilisateur peut exercer sur un objet. Par exemple, si l'on considère la variable EMPLOYE, de type entité, n'ayant que des constituants élémentaires et définie ainsi en LAMBDA :

```
type T-EMPLOYE : entity
        key numero : integer end key
              nom     : string(25)
              age     : integer
              salaire: real
              adresse: address
              poste   : string(15)
              dept    : string(20)

        end.

dbvar EMPLOYE :  T-EMPLOYE
```

plusieurs utilisateurs pourront y avoir accès en lecture, par exemple, s'ils possèdent le privilège de sélection (SELECT). Cependant, ils ne le feront pas tous de la même façon. Pour déterminer les restrictions de ce privilège pour chacun des utilisateurs, on emploie des prédicats. Par exemple, un utilisateur U pourra lire seulement les informations concernant les employés du département D, le prédicat associé sera alors : 'DPT = D'. Un autre utilisateur, soit U' pourra avoir accès uniquement aux employés ayant un salaire inférieur à 2000F. On

lui associe le prédicat : 'SALAIRE < 2000' etc. Donc pour une entité un prédicat correspond à une opération de sélection sur cette entité.

Dans Le système R, la même granularité de protection est fournie par utilisation de la notion de vue. Mais une vue est considérée comme une relation virtuelle donc un objet à gérer par le mécanisme de protection. Nous avons préféré l'utilisation des prédicats pour éviter cette multiplication des objets qu'il faut protéger.

3.3.3 Les cataloques

Les droits d'un utilisateur sur un objet sont représentés à l'aide du catalogue PRIVILEGES :

CODE	TYPC	CODP	NOMobj.	TYPE	OP1	...	OPn

Dans ce catalogue, les utilisateurs ne sont connus que par le code qui leur est attribué dans l'arborescence de gestion des utilisateurs ou de par leur appartenance à un groupe.

Chaque ligne du catalogue donne le code de l'utilisateur, possédant un privilège sur l'objet, son type (groupe ou

utilisateur unique), le code du propriétaire de l'objet (cf. § 2), le type de l'objet (document, entité, association...) et toutes les opérations possibles sur cet objet (lecture, modification, supression...). Dans le tableau précédent, une opération est désignée par OPi. Différentes options sont possibles sur chacune des opérations. Elles permettent de restreindre le privilège accordé à l'utilisateur. Ce sont :

OPi = ' I' : opération interdite.

OPi = ' C' : opération permise sur certains constituants sans restriction. Le terme constituant désignera un attribut, dans une entité. Dans le cas d'un document, il désignera un niveau quelconque de l'arborescence représentant ce document (chapitre, section, paragraphe etc.).

OPi = 'CR' : opération permise sur certains constituants avec les restrictions données par les prédicats.

OPi = 'TS' : opération permise sur tous les constituants sans restriction. Dans ce cas il est inutile de passer par la suite du système de protection.

OPi = 'TR' : opération permise sur tous les constituants avec restriction.

La distinction entre 'TR' et 'TS' d'une part et 'C' et 'CR' d'autre part est effectuée dans un souci d'optimisation. Elle permet de sortir plus rapidement du système de contrôle dans le cas où il n'existe aucune restriction, au lieu d'effectuer la vérification sur tous les composants de l'objet.

Par ailleurs il existe des objets ne nécessitant aucune protection particulière en lecture, par exemple des thèses, des rapports de recherche et tous les documents accessibles par tous les services. Pour éviter de passer par le système de contrôle d'accès pour chaque objet et chaque opération sur la BDC on associe un niveau de confidentialité à toute entité et à tout objet complexe :

'AG' : accès général, dans ce cas le contrôle est inutile, quel que soit l'utilisateur.

'AR' : accès restreint auquel cas il faut faire appel au système de sécurité.

Cette protection sommaire n'est évidemment valable qu'en lecture, les modifications ou les suppressions d'objets ne sont effectuées que par le propriétaire ou des utilisateurs autorisés.

Dans le cas où l'objet est de type entité, pour chaque opération possible avec restriction, le catalogue CONSTITUANTS, géré par le système de protection, donne le détail de l'accès par constituant. Chaque ligne donne :

- Le nom interne du constituant
- Le nom interne de l'entité à laquelle il appartient.
- Un pointeur vers la table des prédicats. Chaque prédicat limite l'accès au constituant ou au tuple de la relation considérée.

3.3.4 Accès aux objets structurés

Dans le cas d'objets structurés comme les documents, la structure à prendre en compte est une arborescence. Sur un document, pour chaque opération possible avec restriction, le catalogue STRUCTURE donne le détail des accès selon la structure du document.

Le constituant permettant d'assurer la confidentialité d'un document peut apparaître à n'importe quel niveau de l'arborescence. S'il est placé par exemple au niveau d'une section, cela indique que l'accès de l'utilisateur à toute la section est soit restreint, soit interdit. Si un niveau donné est protégé avec des restrictions représentées par des prédicats, ces restrictions indiquent selon la valeur des prédicats que le niveau est accessible si le texte comporte (ou ne comporte pas) certains mots, certaines phrases, etc. L'expression de ce type de contraintes nécessite l'extension des prédicats aux opérateurs d'inclusion et de non inclusion.
Par exemple, soit une entité contenant un constituant complexe, de type contrat défini ainsi en LAMBDA :

```
type contrat : document;
   structure
      begin
         introduction : texte;
         corps : liste (1..*) of clause.structure;
         conclusion : texte
      end
end.
```

```
type clause : document;
    structure
       begin
           titre : texte;
           contenu : texte
       end

end.
```

la partie du catalogue STRUCTURE, précédemment cité, associé à un utilisateur U ayant un certain privilège sur un objet de ce type, pourrait être la suivante :

IDENT	NOM	PERE	SECURITE
1	contrat	0	I
2	intro.	1	N
3	corps	1	I
4	clause1	3	T
5	clause2	3	N
6	clause3	3	N
7	clause4	3	Pi
8	concl.	1	N

Ce contrat est constitué d'une introduction, de quatre clauses et d'une conclusion. Sa protection vis-à-vis de U est assurée par l'attribut SECURITE selon la valeur prise par celui-ci :

'N' : niveau non protégé

'T' : niveau totalement protégé. L'accès a ce niveau est interdit à l'utilisateur U.

'Pi' : niveau protégé avec restriction. L'attribut SECURITE est alors un pointeur vers la table des

contraintes.

'I' : niveau non protégé, cependant la protection pourrait se faire à un niveau inférieur.

La protection doit être assurée de façon cohérente. Pour cela on établit une hiérarchie entre les valeurs pouvant être prises par l'attribut SECURITE, c'est-à-dire une relation d'ordre telle que :

N <= I < Pi < T

Ainsi si un niveau est protégé avec l'option T tous ses descendants seront protégés avec cette option. Par exemple, si dans un document un chapitre est protégé avec T alors toutes les subdivisions de ce chapitre sont inaccessibles.

Si un niveau est protégé par un système de prédicats c'est-à-dire avec l'option Pi alors ces prédicats s'appliquent également à tous les niveaux inférieurs qui ne pourront être protégés qu'avec l'option T ou un autre système de prédicats. Dans ce dernier cas, les prédicats ne doivent pas être en conflit avec ceux des niveaux supérieurs : soit P1 un système de prédicats s'appliquant au niveau d'une clause, soit P2 un système s'appliquant au niveau d'un paragraphe. Alors, au moment de la définition de P2, le système vérifie que celui-ci n'est pas en conflit avec P1. Si P1 = 'SALAIRE <= 2000' et P2 = 'SALAIRE < 2500' alors P2 ne sera pas pris en considération.

Dans le cas où un niveau est protégé avec un ensemble de prédicats appartenant à divers niveaux supérieurs alors l'accès à ce niveau doit vérifier le prédicat P. P sera constitué de l'intersection de tous les prédicats des niveaux supérieurs :

$$P = P1 \cap P2 \cap \ldots \cap Pn$$

Au fur et à mesure de la définition des prédicats, leur cohérence est vérifiée. Cette vérification peut se faire de façon aisée : le système prend les prédicats à ajouter et vérifie qu'ils sont tous cohérents avec ceux qui existent déjà.

Une autre vérification de la cohérence est celle effectuée à l'exécution des requêtes. Il s'agit de vérifier que les prédicats détenus par le système ne sont pas en conflit avec ceux que comporte éventuellement la requête. Ce problème qui consiste à vérifier si deux prédicats ne sont pas en conflit s'est déjà posé dans le cas des 'predicate locks' /ESW76/. Il a été prouvé en utilisant les résultats de Kleene /KLE52/ qu'il est indécidable dès que les prédicats deviennnent complexes.

<u>Remarque</u>

Pour simplifier la représentation de la protection d'un objet complexe, il est inutile de représenter les niveaux non protégés. Tout niveau n'apparaissant pas dans l'arborescence de protection sera alors accessible à l'utilisateur.

3.3.5 Accès à un objet à partir d'un poste de travail

Les objets de la BDG sont accessibles à partir des postes de travail. Cependant, le système d'autorisation est intégré au SGBDG. Il se trouve donc sur le serveur.

Pour utiliser les possibilités de traitement offertes par les postes de travail et décharger le serveur de certaines opérations de contrôle les objets complexes (documents) de la BDG sont transférés sur un poste de travail dès leur première référence via ce poste.

Toutes les modifications d'un document est répercutée sur le serveur. Par ailleurs, si un document est en mise à jour sur un poste il ne peut être utilisé simultanément sur un autre poste.

Dès qu'un objet est sur un poste de travail il devient nécessaire de s'assurer qu'il ne sera utilisé que par des utilisateurs dûment autorisés et uniquement pour exécuter les opérations qui leur sont permises tout en respectant les restrictions imposées.

Il existe deux possibilités pour assurer la protection d'un objet sur un poste de travail :

- Quand un objet est demandé pour la première fois sur un poste de travail, transférer en même temps que l'objet la liste des utilisateurs qui y accèdent ainsi que leurs

droits sur cet objet.

- Transférer l'objet et uniquement les droits de l'utilisateur qui l'a demandé et à chaque demande d'accès transférer les droits du nouvel utilisateur sur le poste de travail.

L'une ou l'autre technique peut être utilisée selon l'application. Un paramètre permettra de spécifier la méthode employée.

3.3.6 Propagation des privilèges

La structure arborescente de l'entreprise confère à la propagation des privilèges les caractéristiques suivantes :

-Nous avons vu que tous les supérieurs hiérarchiques d'un utilisateur ont le droit de lecture sur tous les objets créés par cet utilisateur. Pour assurer un certain degré de responsabilité des utilisateurs, le droit de modification ou de suppression d'un objet ne peut être accordé implicitement. Le propriétaire d'un objet peut céder à d'autres usagers le droit de le gérer. Seul l'utilisateur ayant le droit de propriété sur un objet peut transmettre des privilèges sur cet objet.

-Quand un groupe reçoit un privilège sur un objet, ce privilège est implicitement accordé à tous les utilisateurs qui appartielent au groupe.

Les utilisateurs sont donc amenés à recevoir des privilèges sur

des objets de plusieurs façons :

-Ceux-ci leur sont accordés explicitement par les propriétaires des objets.

-Ils peuvent leur être accordés implicitement de par leur niveau dans la hiérarchie.

-Ils les détiennent par leur appartenance à un groupe, qui a reçu le privilège.

La propagation explicite des privilèges se fait à l'aide de la commande GIVE. La syntaxe en est la suivante :

GIVE { <liste-de-privilèges> | ALL } TO { <liste-d'utilisateurs> | ALL } ON { <liste-d'objets> | ALL } [clause-where]

où :

<liste-de-privilèges> est l'ensemble des privilèges transmis

<liste-d'utilisateurs> est l'ensemble des utilisateurs destinés à recevoir ces privilèges.

<liste-d'objets> est l'ensemblé des objets sur lesquels sont transmis les privilèges.

Pratiquement ces privilèges ne sont effectivement transmis que si l'utilisateur qui émet la requête est propriétaire des objets référencés.

ALL indique que l'on remplace <liste-de-privilèges>, <liste-d'utilisateurs> ou <liste-d'objets respectivement par <u>tous</u> les utilisateurs, <u>tous</u> les privilèges ou <u>tous</u> les objets.

Si <liste-d'objets> est remplacé par ALL cela signifie que les privilèges sont transmis sur tous les objets dont

l'utilisateur exécutant la commande est propriétaire.

3.3.7 Révocation des privilèges

Elle consiste à retirer à un utilisateur un ou plusieurs privilèges qui lui ont été accordés sur un objet. Après exécution de la requête de révocation, tout se passe comme si l'utilisateur en question n'avait jamais été en possession de ces privilèges. La révocation se produit à la demande explicite du propriétaire de l'objet, il suffit pour cela qu'il émette une requête de révocation. L'algorithme déroulé est alors très simple, un objet ne pouvant avoir qu'un seul propriétaire à un moment donné.

Pour bien faire ressortir l'intérêt de cette simplicité montrons le fonctionnement d'un algorithme utilisé dans un autre système, par exemple celui utilisé dans le système R. Dans ce dernier, si on retire un privilège à un utilisateur soit directement soit indirectement, tous les utilisateurs à qui il a pu le transmettre à son tour, le perdent, sauf ceux qui l'ont reçu d'une autre source indépendante. Le mécanisme implique une révocation récursive le long de l'arbre (donneur, receveur) /GRI76/, /WIL82/. Le donneur est l'utilisateur qui a accordé le privilège, le receveur celui qui l'a reçu. Chaque receveur devient à son tour donneur dans l'étape suivante de l'algorithme. Pour appliquer ce mécanisme il est nécessaire de conserver le temps où ce privilège a été accordé, une technique d'estampille

est utilisée.

Dans notre système, la révocation des privilèges s'effectue en une seule opération. Elle peut se faire implicitement si ceux-ci ont été accordés implicitement (suppression d'un utilisateur d'un groupe possédant le privilège concerné). Généralement elle se fait de façon explicite par utilisation de la commande REMOVE dont la syntaxe est la suivante :

```
REMOVE { <liste-de-privilèges> } FROM { <liste-d'utilisateurs> }
       { ALL                   }      { ALL                   }
ON { <liste-d'objets> }
   { ALL              }
```

Si la révocation explicite porte sur un privilège qui a été accordé implicitement, ceci se traduira par l'octroi du privilège FORBID (voir plus haut) à l'utilisateur dont on veut supprimer le privilège. Evidemment ceci n'est valable que pour le privilège de sélection sur un objet.

Ces diffrérents cas de révocation sont exprimés très simplement par l'algorithme de révocation. Illustrons le fonctionnement de cet algorithme par un exemple :

soit la requête suivante émise par l'utilisateur U1 :

REMOVE 'P' ON 'O' FROM U2

pour rendre cette révocation effective l'algorithme suivant sera déroulé :

1. vérifier que U1 est le propriétaire de 'O'
2. Si U2 a reçu explicitement le privilège 'P' sur l'objet 'O'

alors révocation

3. Si U2 appartient à un groupe ayant reçu 'P' sur 'O' alors le signaler à U1. Fin

4. Si 'P' = SELECT vérifier si U2 est un supérieur hiérarchique de U1 alors octroyer le privilège FORBID à U1. Fin

5. Si aucun de ces points n'est satisfait alors envoyer à U1 le message suivant : "U2 ne possède pas le privilège P sur l'objet O ". FIN

Remarque

Quand un poste (ou un groupe) est supprimé dans l'entreprise (cf. § 3.4.2) tous les privilèges associés à ce poste (ou à ce groupe) sont supprimés implicitement de tous les catalogues. C'est le seul cas de révocation implicite.

3.3.8 Conclusion

La représentation de la structure d'un objet est répétée dans Le catalogue STRUCTURE ou CONSTITUANT pour chaque utilisateur et chaque privilège. Tous les objets accessibles à l'utilisateur sont représentés dans l'un de ces catalogues ou l'autre, en fonction de leur type, sauf ceux dont le niveau de confidentialité est 'AG'. Cette façon de procéder permet de protéger les objets au niveau le plus fin mais demande beaucoup de place en mémoire secondaire. L'utilisation de la notion de groupe permet de remédier à cet inconvénient. Si les utilisateurs sont gérés en groupes le catalogue ne sera pas

répété pour chaque utilisateur mais sera associé au groupe. En fait, dans une entreprise, un objet peut en général être accédé avec les mêmes privilèges par toute une application donc un groupe. Seuls quelques objets confidentiels seront gérés, pour chaque utilisateur, individuellement.

3.4. Réorganisation de l'entreprise

3.4.1 Le problème

L'entreprise est une organisation dynamique caractérisée par l'arrivée et le départ des membres du personnel. Il faut également tenir compte du fait que la situation de ceux-ci, dans l'entreprise, évolue. D'autre part, une entreprise est appelée à être restructurée : fusion de deux ou plusieurs entreprises, éclatement d'une entreprise ou réorganisation de sa structure.

Le système d'autorisation doit tenir compte de toutes ces caractéristiques et répercuter toute modification de l'entreprise sur l'arborescence représentant la position hiérarchique des utilisateurs. Ces modifications doivent également être prises en compte par le système de gestion des privilèges.

3.4.2 Arrivée et départ d'un utilisateur

Dans ce paragraphe les cas que nous considérons se ramènent à l'ajout ou à la suppression de noeuds feuilles. Les

modifications intervenant au niveau des autres noeuds seront considérés comme faisant partie de la restructuration de l'entreprise et pris en compte au paragraphe suivant.

-Le départ d'un utilisateur, si celui-ci n'est pas remplacé par un autre (cas de suppression d'un poste) conduit à la suppression de son code et à la révocation de tous ses privilèges sur tous les objets de la BDG. Le code de cet utilisateur doit également disparaître de tous les groupes dont il fait partie.

-Dans le cas où l'utilisateur partant est remplacé par un autre, le nouvel utilisateur prend le code de l'ancien puisqu'un code correspond non à un utilisateur mais à un poste dans l'entreprise. L'utilisateur, ainsi introduit, acquiert tous les privilèges de l'ancien, sauf ceux qui lui seront explicitement révoqués. L'utilisateur étant représenté par son code dans les divers groupes dont il est membre, ce code continuera d'exister dans les groupes en cas de changement de l'utilisateur qu'il représente. Pour exclure l'utilisateur de certains ou de tous ces groupes il faudra les parcourir et en supprimer explicitement le code.

-L'introduction d'un nouvel utilisateur à un certain niveau, en cas de création d'un nouveau poste, par exemple, conduit à l'attribution d'un code à cet utilisateur qui pourra acquérir ultérieurement des privilèges sur les objets de la base. Si cet utilisateur est doté du droit de création, il

pourra créer de nouveaux types et transmettre sur ceux-ci des privilèges à d'autres utilisateurs.

Pour prendre en compte ces arrivées et départs d'utilisateurs, le système d'autorisation dispose de primitives de bas niveau qui seront mises en oeuvre par le langage de haut niveau LAMBDA ou exécutées directement sur demande.

Il est clair que seul l'ABD est habilité à lancer l'exécution de ces primitives.

3.4.3 Restructuration de l'entreprise

Les cas que l'on vient d'étudier apparaissent comme des cas simples, dans la vie de l'entreprise, puisqu'ils concernent l'arrivée ou le départ d'un seul utilisateur. Cela revient à l'ajout ou à la suppression d'une feuille dans l'arborescence. Le problème le plus important du point de vue de la prise en compte par le système d'autorisation est celui posé par la restructuration de l'entreprise soit en cas de sa réorganisation soit en cas de sa fusion avec une autre. Ceci conduit au déplacement des utilisateurs dans la hiérarchie et/ou à l'introduction de nouveaux sous-arbres dans l'arborescence initiale et nécessite la modification des privilèges accordés aux divers utilisateurs.

Cette réorganisation peut être reflétée de diverses façons

par le mécanisme d'autorisation, en fonction de l'importance de la restructuration. Elle pourrait être prise en compte très simplement par le système si elle consiste en :

-La suppression d'une direction, d'un département...

-La création d'une nouvelle direction...

-La fusion de deux services, départements...

En effet ces restructurations seront interprétées par le système d'autorisation comme des suppressions, créations ou modifications de sous-arbres de l'arborescence initiale. Or de telles opérations pourront être exécutées à partir de n'importe quel niveau dans l'arborescence, il suffit de donner comme paramètre la racine du sous-arbre à supprimer, à créer ou à modifier. Nous pouvons illustrer ces cas à l'aide de l'exemple suivant d'organisation de l'entreprise :

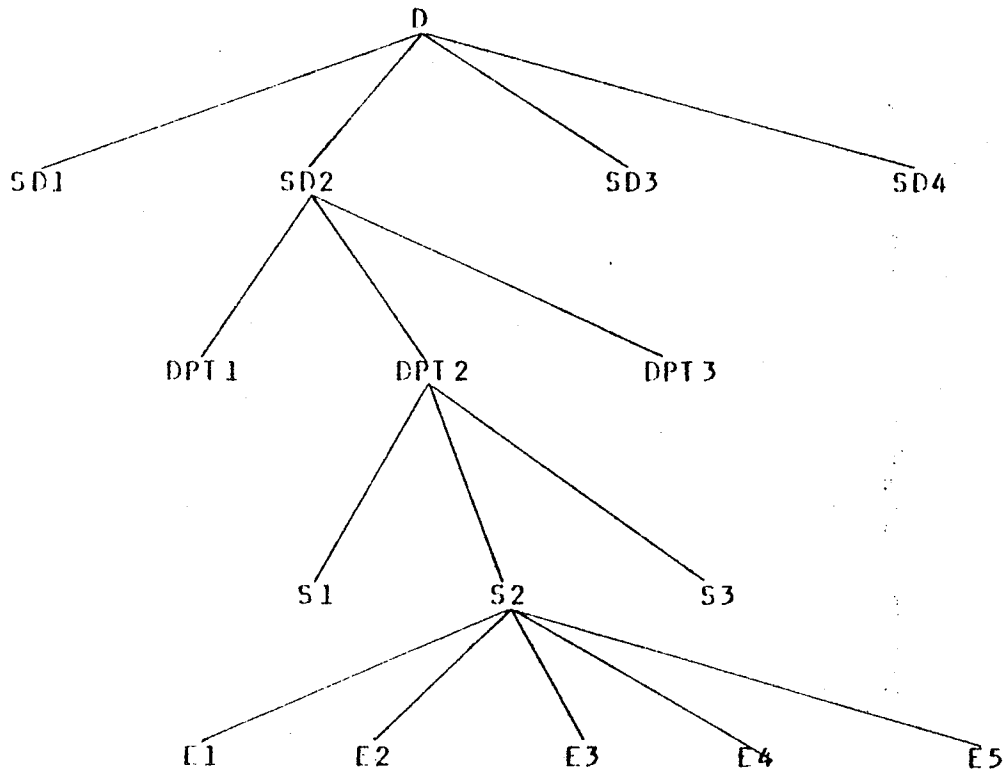


figure 5 : représentation d'une entreprise

où :

D = direction
SDi = sous-directtion i
DPTi = département i
Si = service i
Ei = employé i

La création d'une nouvelle sous-direction soit SD5 peut être présentée comme la création du sous-arbre :

SD5(DPT4(S4,S5(E6,E7,E8),DPT5)

Après création de SD5 l'arborescence initiale deviendra :

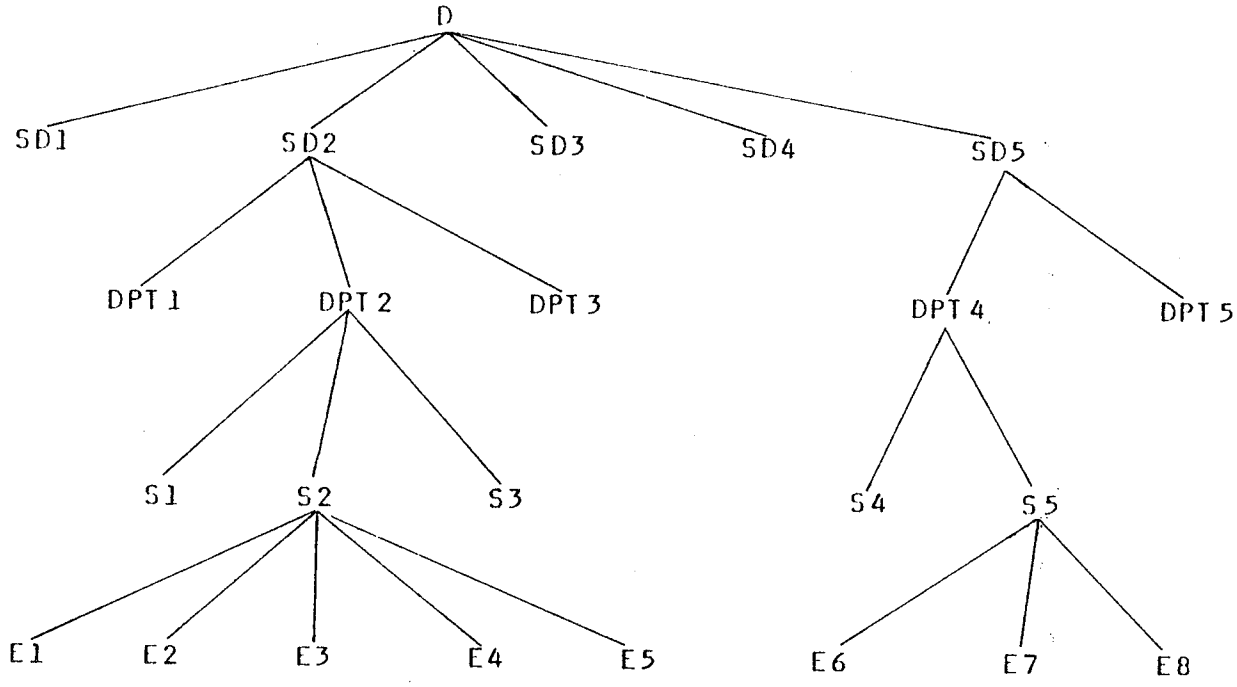


Figure 6 : arborescence après restructuration

La suppression d'un sous-arbre peut également être effectuée très simplement. Il suffit de donner en paramètre, à la primitive qui réalisera la suppression, la racine du sous-arbre à supprimer.

Le transfert d'un service d'un département vers un autre, par exemple, peut se traduire par une opération de modification c'est-à-dire par une suppression suivie d'une création de sous-arbre.

Cependant il existe des cas où l'on a une restructuration plus complexe avec création de niveaux intermédiaires dans

l'arborescence. Si cette création se fait à un niveau assez élevé dans l'arborescence et pour plusieurs branches, cela pourrait conduire à un recodage systématique de tous les utilisateurs. Dans ce cas, il faudrait parcourir la table de gestion des privilèges et modifier le code des utilisateurs et éventuellement leurs droits ce qui est un travail fastidieux. Mais une réorganisation aussi poussée est un cas exceptionnel qui pourrait se produire une ou deux fois dans la vie d'une entreprise. D'autre part, une utilisation judicieuse de la notion de groupe permettrait de réduire considérablement ce travail. En effet, s'il existe une bonne gestion des utilisateurs en groupes, la majeure partie de ce travail consisterait uniquement à changer le code des utilisateurs dans les groupes, le code du groupe lui-même restant inchangé. Le nombre d'accès au catalogue de gestion des privilèges sera ainsi considérablement réduit.

Une autre façon de réduire le nombre d'accès serait d'envisager un accès au catalogue sur le code de l'utilisateur.

Une utilisation combinée de la notion de groupe et de l'accès sur le code, au catalogue permettrait d'atténuer considérablement le problème posé par la restructuration qui apparaît comme l'inconvénient majeur de la gestion hiérarchisée des utilisateurs.

Il serait facile d'intégrer à LAMBDA des commandes de manipulation d'arborescences faisant appel aux primitives. Ces commandes seraient identiques à celles gérant les groupes d'utilisateurs. Cependant, l'arborescence est créée en une seule fois et la restructuration ne se produit que rarement. Il suffit de mettre à la disposition de l'ABD ces primitives de base, qu'il pourra appeler quand cela est nécessaire.

3.5. Aspects d'implantation

Dans l'implantation de tout le système d'autorisation, nous veillons à conserver celui-ci aussi indépendant que possible du SGBD l'implantant. C'est dans cet objectif que nous avons développé SACE sous forme de primitives pouvant être appelées par n'importe quel langage de haut niveau et appelant lui-même des modules du SGBD pour accéder aux informations stockées dans les catalogues de celui-ci. D'un SGBD à l'autre il suffit de changer le nom des modules appelés pour implanter SACE. Dans toutes les primitives de SACE, on adoptera les conventions suivantes pour les paramètres d'entrée et de sortie :

- -NIO : désignera le nom interne de l'objet
- -COD : désignera le code attribué à un utilisateur selon le poste qu'il occupe dans l'entreprise.
- -AC : fera référence à un privilège quelconque
- -PR : désignera un prédicat ou un système de prédicats.
- -CE : sera un code erreur indiquant si la primitive a été exécutée correctement ou non.

3.5.1 Utilisateurs et groupes

L'implantation de l'arborescence initiale, représentant les utilisateurs, se traduit par la création d'une entité dont les attributs ont été décrits au paragraphe 2.1. A chaque utilisateur introduit, la fonction de codage associe un code interne. Dans la suite de sa mise en oeuvre, le système d'autorisation ne connaîtra l'utilisateur que par ce code.

Les opérations sur les groupes d'utilisateurs, exprimées en un langage de haut niveau, se ramènent à l'appel d'un certain nombre de primitives du système d'autorisation permettant d'assurer l'implantation et la manipulation de ces groupes :

-CREER-GROUPE(COD, CE)

Paramètres en entrée : COD

Paramètres en sortie : CE

assure la création d'un groupe qui sera désigné par le code interne COD. Initialement aucun utilisateur n'est affilié à ce groupe.

-DETRUIRE-GROUPE(COD, CE)

Paramètres en entrée : COD

paramètres en sortie : CE

réalise l'opération inverse. Dans ce cas COD n'existera plus dans le système et tous les privilèges qui ont pu lui être octroyés lui sont révoqués.

-AJOUTER-UTILISATEUR(COD1, COD2, CE)

Paramètres en entrée : COD1, COD2

Paramètres en sortie : CE

permet d'introduire l'utilisateur COD1 dans le groupe COD2.

-SUPPRIMER-UTILISATEUR(COD1, COD2, CE)

Paramètres en entrée : COD1, COD2

Paramètres en sortie : CE

réalise l'opération inverse. L'utilisateur COD1 est supprimé du groupe COD2. Dans ce cas de suppression, contrairement au précédent, il ne se passe rien au niveau de la révocation des privilèges et le code COD1 n'est pas détruit. En effet, un utilisateur peut exister autrement que de par son appartenance à un groupe.

-FUSIONNER-GROUPE(COD1, COD2, CE)

Paramètres en entrée : COD1, COD2

Paramètres en sortie : CE

prend tous les utilisateurs du groupe COD2 et les intègre au groupe COD1. Le groupe COD2 n'est pas supprimé et conserve tou les privilèges qui lui ont été accordés. Un groupe ne peut être détruit que par exécution de la commande DELETE-GROUP. Les utilisateurs appartenant au groupe COD2 auront dès lors les privilèges inhérents à leur appartenance au groupe COD1. Cela est effectué sans modification des privilèges puisque ceux-ci sont transmis globalement à tout le groupe.

En ce qui concerne les primitives portant sur l'arborescence c'est-à-dire sur les utilisateurs individuellement, elles sont étudiées en détail au paragrapghe 3.6.3.

3.5.2 Contrôle et autorisation

Le système d'autorisation sera mis en oeuvre par l'intermédiaire d'un langage de primitives de bas niveau qui seront appelées soit implicitement lors de l'accès à un objet, soit explicitement à l'aide des commandes GIVE et REMOVE intégrées au langage de manipulation de TIGRE, LAMBDA.

Dans LAMBDA, comme dans MIQUEL /FRT80/, une requête utilisateur est traduite en arborescence algébrique binaire. Au fur et à mesure de cette traduction, il est possible de vérifier si les objets référencés, quel que soit leur type (entité ou association), sont d'accès général (AG) ou restreint (AR). Dans le premier cas, l'exécution se poursuit sans faire appel aux autres mécanismes du système de protection. Pour les objets protégés avec l'option AR, il faut en vérifier toutes les contraintes d'accès c'est-à-dire passer par toutes les primitives nécessaires du système. C'est la primitive :

VERIFIER-TYPE(NIO, TYPE, CE)

Paramètres en entrée : NIO

Paramètres en sortie : TYPE, CE

qui effectue ce contrôle.

Le paramètre TYPE aura la valeur AG ou AR selon que l'objet est d'accès général ou restreint.

Le contrôle d'accès à une information, proprement dit, est effectué par la primitive :

VERIFIER-ACCES(NIO, COD, AC, RES, CE)

Paramètres en entrée : NIO, COD, AC

Paramètres en sortie : RES, CE

Le paramètre RES est un booléen. Il aura la valeur 'vrai' si l'utilisateur COD peut accéder à l'objet NIO avec le privilège AC et la valeur 'faux' sinon. Dans le premier cas le contrôle se poursuit sur les subdivisions de l'objet, jusqu'au niveau référencé dans la requête. Cela permettra de vérifier si l'accès est effectivement autorisé ou non. Par exemple, dans le cas d'une entité, on accèdera au catalogue CONSTITUANT, défini précédemment (cf § 3.4.3) pour vérifier si l'accès aux attributs apparaissant dans la requête est autorisé ou non.

C'est à ce niveau qu'est également effectué le contrôle de validité des prédicats et de la cohérence entre les prédicats détenus par le système et ceux constituant éventuellement les critères de sélection de la requête. Ceci est réalisé respectivement par les primitives :

-VERIFIER-CONDITION qui vérifie que l'utilisateur n'aura accès qu'aux données restreintes par le système de prédicats associé.

-VERIFIER-COHERENCE dont le rôle est de vérifier que les prédicats stockés par le système sont cohérents avec les critères de sélection de la requête.

Les primitives que nous venons de présenter concernent le contrôle d'accès. Nous savons d'autre part qu'un utilisateur propriétaire d'objets peut transmettre des privilèges sur ces

objets à d'autres utilisateurs. Il peut également leur révoquer des privilèges qu'ils ont préalablement reçus sur ces objets. Ceci sera effectué au niveau utilisateur par les commandes GIVE et REMOVE de LAMBDA qui feront appel à deux primitives de bas niveau :

-AUTORISER-ACCES(COD1, COD2, NIO, AC, PR, CE)

Paramètres en entrée : COD1, COD2, NIO, AC, PR

Paramètres en sortie : CE

COD1 est le code de l'utilisateur qui lance l'exécution de la primitive c'est-à-dire qui transmet le privilège AC. COD2 est l'utilisateur recevant le privilège. PR est l'adresse d'un système de prédicats restreignant la portée du privilège AC. S'il n'existe pas de restriction sur la transmission alors PR prendra la valeur -1.

La première opération effectuée par cette primitive consiste à vérifier que COD1 identifie bien le propriétaire de l'objet. Si c'est le cas des informations sont insérées dans les divers catalogues concernés pour permettre à l'utilisateur COD2 d'avoir accès à l'objet NIO avec le privilège AC.

-RETIRER-ACCES(COD1, COD2, NIO, AC, CE)

Paramètres en entrée : COD1, COD2, NIO, AC

Paramètres en sortie : CE

réalise l'opération inverse : l'utilisateur COD1 retire à l'utilisateur COD2 le privilège AC sur l'objet NIO. Dans ce cas également il faut commencer par vérifier que COD1 est bien le propriétaire actuel de l'objet NIO.

La deuxième étape consiste à suprimer du catalogue les informations permettant à COD2 d'acccéder NIO ave AC.
Dans le cas où COD2 n'a pas reçu explicitement le privilège AC et s'il s'agit du privilège SELECT on vérifie si COD2 est un supérieur hiérarchique de COD1, dans l'arborescence. Si c'est le cas alors COD2 sera doté du privilège FORBID (interdiction de lecture pour un supérieur) dans le catalogue d'autorisation.

3.5.3 Réorganisation de l'entreprise

Le comportement dynamique des utilisateurs dans une entreprise, se traduit par l'ajout, la suppression ou la modification d'un sous-arbre ou d'une feuille dans l'arborescence initiale. Un certain nombre de primitives réalisent ces différentes fonctions :

-SUPPRIMER-UTILISATEUR(COD, CE)

Paramètres en entrée : COD

Paramètres en sortie : CE

permet la suppression d'un noeud feuille.

-REMPLACER-UTILISATEUR(COD1, NOM, CE);

Paramètres en entrée : COD1, NOM

Paramètres en sortie : CE

COD1 est le code sur lequel s'effectue le changement. NOM est le nom de l'utilisateur auquel sera affecté COD1. Cet utilisateur peut déjà avoir un autre code dans l'arborescence. Il hérite de tous les privilèges associés à

COD1 sauf ceux qui lui seront explicitement révoqués. Il en est de même en ce qui concerne son appartenance à divers groupes.

-CREER-UTILISATEUR(COD, CE)

Paramètres en entrée : COD

Paramètres en sortie : CE

permet d'introduire un nouvel utilisateur c'est-à-dire une feuille dans l'arborescence.

Les primitives que l'on vient de citer portent sur un seul utilisateur c'est-à-dire sur un noeud feuille. La restructuration de l'entreprise se traduit par la manipulation de sous-arbres. Ceci peut être effectué grâce aux procédures suivantes :

-CREER-ARBRE(COD, ADR, CE)

Paramètres en entrée : COD, ADR

Paramètres en sortie : CE

permet de créer un sous-arbre dans l'arborescence initiale. COD identifie le noeud à partir duquel sera créé le sous-arbre et ADR est l'adresse du sous-arbre à créer. Celui-ci est fourni au système sous forme de liste.

-SUPPRIMER-ARBRE(COD, CE)

Paramètres en entrée : COD

Paramètres en sortie : CE

réalise l'opération inverse. Le sous-arbre de racine COD est supprimé de l'arborescence initiale.

-MODIFIER-ARBRE(COD1, COD2, CE)

Paramètres en entrée : COD1, COD2

Paramètres en sortie : CE

permet d'effectuer le transfert d'un sous-arbre vers un autre. Ceci pourrait se produire par exemple dans le cas de transfert d'un service d'un département vers un autre. COD1 est la racine du sous-arbre à supprimer. COD2 est la nouvelle racine associée au sous-arbre.

CHAPITRE IV

ELEMENTS DE REALISATION

Le SGBD TIGRE est développé sur le SGBD relationnel MICROBE <FER81>, <NGU83> qui réalise les fonctions de base du système (gestion de la mémoire, accès aux données...). MICROBE gère trois types de relations <FER81> :

-Les relations internes qui représentent les catalogues de la BD.

-Les relations de base qui stockent les données.

-les relations inverses qui sont des index sur les relations de base.

La correspondance entre le schéma de TIGRE et un SGBD relationnel a été décrite dans <TIGR5>. Le schéma de TIGRE sera donc représenté par des relations de base de MICROBE. Une extension de ce dernier, pour la définition et la manipulation des objets généralisés, est en cours.

4.1 Le SGBD MICROBE

Le SGBD relationnel MICROBE comporte quatre modules, indépendants du point de vue de leur architecture et de leur fonctionnement :

-Un traducteur d'un langage de haut niveau MIQUEL en arborescence algébrique binaire <FRT80>.

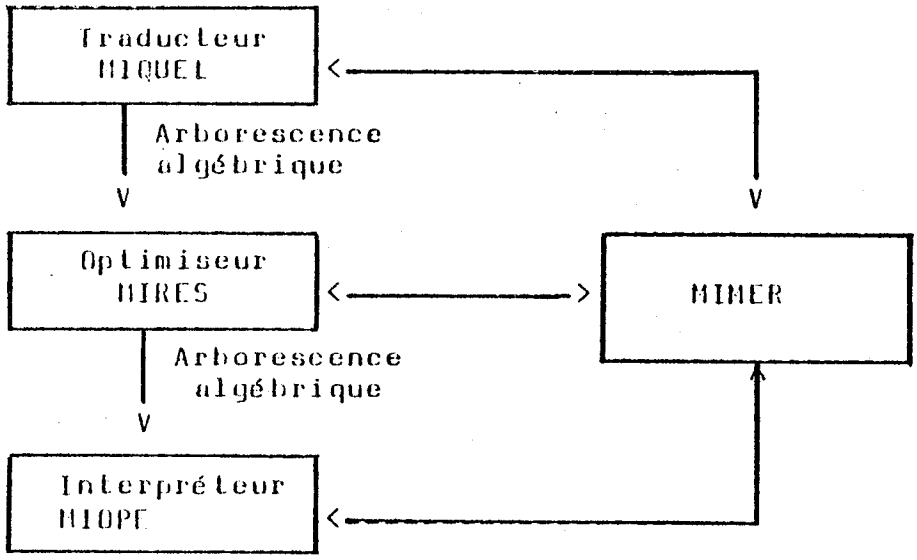
-Une mémoire relationnelle MIMER qui permet la gestion de la mémoire de stockage des informations en offrant des primitives de manipulation de la base et des méthodes d'accès, en particulier celle des B-arbres <FER81>, <LEE83>.

-Un optimiseur MIRES dont le rôle est la restructuration de

l'arborescence algébrique, fournie par le traducteur, de façon à minimiser le volume d'informations inter-opérateurs <GAL83>, <WIN82>. Cette optimisation est basée uniquement sur les propriétés de l'algèbre relationnelle.

-Un interpréteur MIOPE utilisant une bibliothèque d'algorithmes des opérateurs relationnels <LEE81>.

Une architecture fonctionnelle du système MICROBE pourrait être la suivante :



4.2 Interface MICROBE-TIGRE

L'interface offert par MICROBE est représenté par l'arborescence algébrique binaire. Pour que le système TIGRE puisse utiliser MICROBE pour la gestion de la mémoire relationnelle, il suffit que le langage de manipulation de TIGRE

génère une telle arborescence.

MICROBE a été développé sur un micro-ordinateur, le LSI/11-23, sous le système d'exploitation RSX-11M. Actuellement, dans ce type de machine, le temps de réponse est pénalisé par la place disponible en mémoire centrale, ce qui nécessite l'utilisation d'un arbre de recouvrement très contraignant. Pour utiliser MICROBE comme support relationnel de TIGRE nous avons dû le transporter sous le système MULTICS. En effet, celui-ci offre une grande puissance de traitement et de stockage. Il est à noter que seuls les modules constituant MIMER et MIOPE ont été transférés. Il est alors à la charge du traducteur et de l'interpréteur de TIGRE de générer des arborescences algébriques directement interprétables par MIOPE.
Le transfert de MICROBE sous MULTICS a permis de le paramétrer de façon à rendre aisé son transfert sur n'importe quelle machine. D'autre part, toutes les primitives de niveau A de MIMER <FER81> ont été regroupées dans un seul module et réécrites en pascal. Il s'agit des primitives réalisant les opérations de création, d'ouverture et de fermeture du fichier allouant la base de données. Sous le système RSX-11M ces primitives sont écrites en assembleur.

4.3 Position de SAGE dans TIGRE et dans MICROBE

Le système d'autorisation SAGE est constitué d'un ensemble de catalogues et d'un ensemble de primitives permettant de les

gérer et d'assurer la protection des données et la gestion des utilisateurs.

Du point de vue fonctionnel, une requête émise par un utilisateur en LAMBDA fait appel implicitement au système SAGE qui effectue les contrôles suivants :

-Identification de l'utilisateur. Cela revient à vérifier que l'émetteur de la requête correspond à un usager qui apparaît dans l'arborescence modélisant l'entreprise ou à un groupe c'est-à-dire qu'il s'agit d'un utilisateur du système.

-La deuxième vérification concerne le contrôle d'accès. Il s'agit de s'assurer que tout objet référencé est accessible à l'utilisateur, pour les traitements qu'il veut effectuer. Ce contrôle peut avoir lieu à différentes phases du traitement de la requête :

-au moment de la traduction de la requête, si la protection des données est indépendante de leur valeur. Dans ce cas il n'existe pas de prédicats associés aux restrictions éventuelles sur l'accès (cf. § 3.3.3). Si la protection dépend de la valeur des données c'est le contrôle de cohérence des prédicats qui sera effectué à ce niveau : on vérifie que les prédicats stockés par le système ne sont pas en contradiction avec ceux que comporte éventuellement la requête.

-Si la protection des données dépend de leur valeur, le contrôle se poursuit à l'interprétation. Une

occurrence d'entité ou d'association ne sera transmise à l'utilisateur que si elle satisfait toutes les conditions exprimées par les prédicats du système d'autorisation.

Ceci est valable uniquement pour des entités n'ayant que des constituants simples. En effet, ces contrôles se font en accédant aux catalogues de TIGRE qui sont stockés et gérés par MIMER comme des relations de base. Quand il s'agit d'objets complexes la vérification est effectuée en utilisant les structures de données offertes par un module spécial : le Gestionnaire d'Objets Complexes (GOC). Actuellement, parmi les objets complexes nous ne prenons en considération que les documents. Le contrôle d'autorisation d'accès aux documents peut s'effectuer au niveau de deux modules :

-Le GOC

-L'éditeur interactif de documents.

Le contrôle au niveau de l'éditeur n'intervient qu'à la manipulation de l'objet, aprés un contrôle préalable effectué au niveau du GOC par SAGE qui décide alors d'envoyer ou non le document sur le poste de travail pour le mettre à la disposition de l'utilisateur en fonction des droits de celui-ci. Ce contrôle sur un objet complexe est effectué en utilisant la partie de la table STRUCTURE correspondant à l'utilisateur et à l'objet. Cette table sera créée et gérée par SAGE.

Du point de vue architectural TIGRE met en oeuvre MICROBE par l'intermédiaire de deux modules :

- Les routines sémantiques des analyseurs LDD et LMD qui effectuent, par l'intermédiaire de l'interface BD, les appels nécessaires à MICROBE pour tous les accès aux catalogues.
- L'interpréteur qui donne en entrée à MIOPE une arborescence algébrique binaire. Celui-ci l'exécute et renvoie le résultat sous forme d'une page MICROBE. Ce type de résultat est dû au fait que MIOPE exécute en pipe-line l'arborescence algébrique qui lui est soumise. Un opérateur d'un niveau donné commence son exécution dès que l'opérateur de niveau inférieur remplit une page. Cette méthode a été adoptée dans un souci d'optimisation.

D'autre part, les objets complexes sont référencés au niveau de MIMER par un pointeur. Celui-ci est l'attribut correspondant au document dans l'entité où il est défini. C'est cette référence qui assure le lien entre le SGBD MICROBE et la structure de document.

Il existe toujours un interface entre les systèmes ou modules de description et de stockage (MICROBE ou GOC) et les modules les accédant. C'est cet interface qui met en oeuvre SAGE. Dans l'architecture de TIGRE le LDD accède également au GOC pour la génération des catalogues représentant la description des objets complexes.

Le système d'autorisation doit donc être accessible à partir de quatre modules :

-Routines sémantiques du traducteur

-Interpréteur

-Gestionnaire des Objets Complexes (GOC)

-Editeur interactif de documents

Au niveau des deux premiers modules, SAGE contrôle tous les accès à MICROBE. En ce qui concerne le troisième et le quatrième module il assure le contrôle d'accès aux objets complexes. Le stockage des catalogues et des primitives de SAGE se trouve donc réparti de la façon suivante :

-Tout ce qui concerne le contrôle des utilisateurs et celui des entités n'ayant que des constituants simples est stocké par MICROBE.

-Les catalogues de protection des objets complexes et les primitives les manipulant sont stockés par le GOC.

La structure fonctionnelle de SAGE, par rapport à MICROBE et TIGRE, pourrait être représentée ainsi :

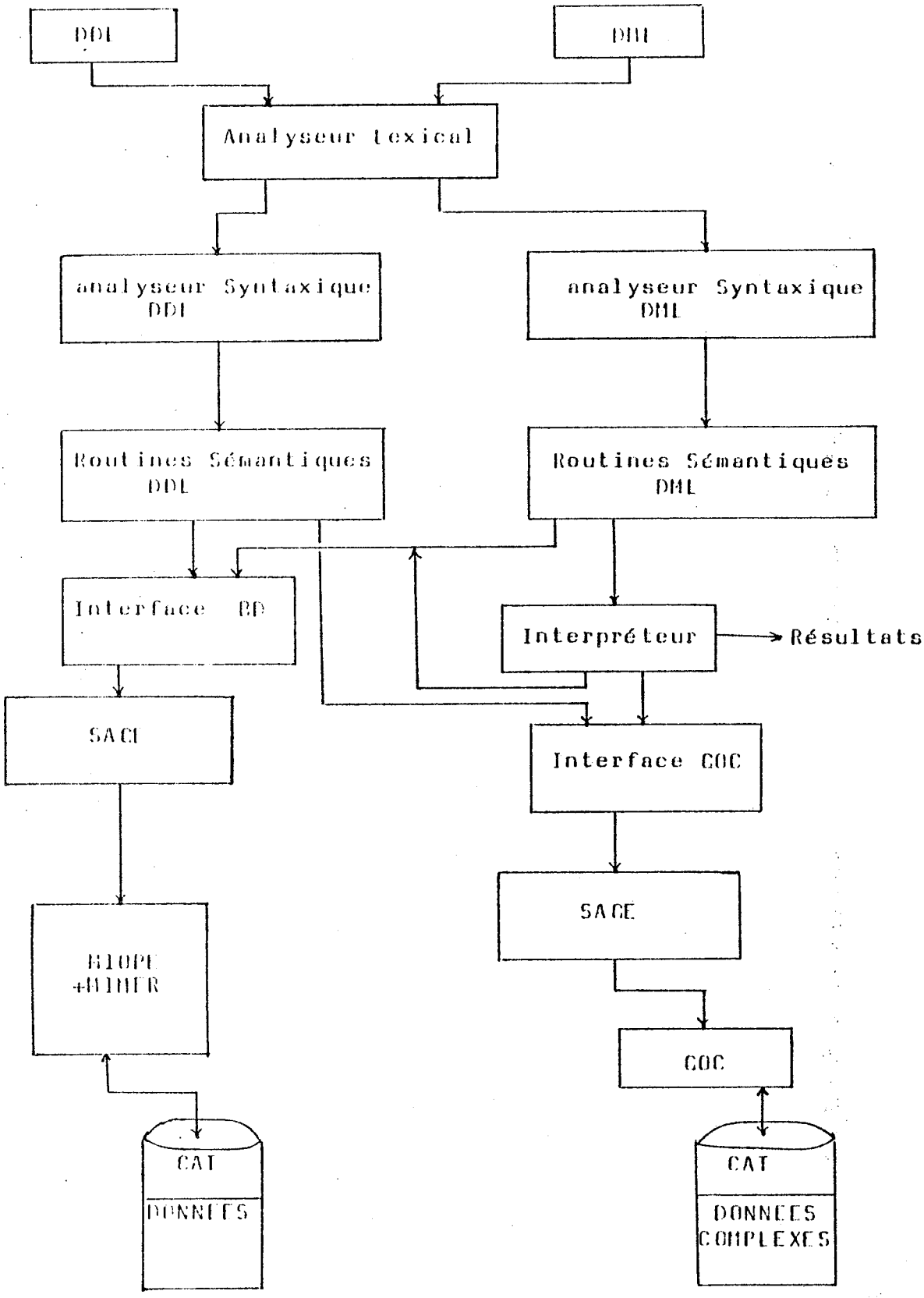


Figure 7 : architecture fonctionnelle de TIGRE

Ainsi toute demande d'accès au schéma ou aux données du serveur est contrôlée par SAGE. S'il existe des demandes d'accès non autorisés la requête est rejetée par le système.

4.4 Le catalogue de SAGE

Comme nous l'avons vu au chapitre précédent, SAGE a besoin d'un certain nombre de relations catalogues pour gérer la structure représentant les utilisateurs et contrôler les droits de ceux-ci sur les objets de la base. Dans ce chapitre nous nous contentons de citer ces relations et leurs constituants, des explications détaillées à ce sujet ayant été données au chapitre trois. Le catalogue de SAGE est constitué des relations suivantes :

-CAT-STRUC(nom, poste, code, fils, frère, suivant, création)
décrit la structure hiérarchique des utilisateurs.

-CAT-GROUPE(nom, code, utilisateurs)
décrit les divers groupes et pour chaque groupe donne la liste de ses membres.

-CAT-INSERT(créateur, utilisateur, variable, insert)
qui enregistre pour chaque utilisateur recevant le privilège d'insertion sur une variable, s'il le reçoit avec le droit de propriété sur les objets complexes ou non.

-CAT-PRIVILEGE(code, typcode, codp, nomobj., privilèges)
contient tous les privilèges d'un utilisateur sur un objet.

-CAT-ATTRIBUT(nom, entité, predicats)
donne le détail de l'accès pour chaque attribut d'une

entité.

-CAT-STRUCTURE(niveau, nom, père, securité)

décrit la protection assurée à chaque niveau d'un document.

-CAT-PREDICAT(opérande1, opérateur, type, opérande2)

Décrit les prédicats associés à la protection d'un attribut ou d'un niveau donné s'il s'agit d'un objet complexe. Type donne le type du deuxième opérande; il peut s'agir d'un attribut ou d'une valeur.

Nous avons deux possibilités pour stocker ces relations :

-En les intégrant au catalogue de TIGRE : dans ce cas, elles seront stockées comme des relations de base de MICROBE, c'est-à-dire des relations devant contenir des données. Les relations à protéger seront alors celles définies par la commande "dbvar" et qui stockent effectivement des données et les relations représentant le catalogue de TIGRE.

-En les intégrant au niveau de MIMER (MICROBE) : elles seront alors gérées comme des relations internes c'est-à-dire des relations catalogues au sens de MICROBE. La génération de ces relations se fera donc à la création de la base. Elles seront générées automatiquement de la même manière que les relations ATTRIBUT, INDEX, ATTRIBUT-CLE, et ARGUMENT <FER81>. Si le catalogue de SAGE est créé à ce niveau il sera placé à un niveau mieux adapté à la protection des relations de base qu'elles soient de types données ou catalogues de TIGRE. La gestion du catalogue de SAGE se fera à l'aide des primitives de MIMER étendues à la

prise en compte de la gestion des autorisations, et des primitives propres à SAGE.

Cette seconde méthode nous paraît plus intéressante dans la mesure où elle permet de compléter le système de base et de le pourvoir d'un système d'autorisation. La protection au niveau généralisé sera considérée comme une extension de celui-ci pour intégrer :

- La syntaxe d'autorisation compatible avec LAMBDA
- les interfaces avec les routines sémantiques, l'interpréteur et le Gestionnaire des Objets Complexes.

Cependant, dans cette première version de SAGE, c'est la première méthode qui sera utilisée. En effet , l'objectif est de tester le fonctionnement du système d'autorisation proposé quelle que soit la méthode adoptée pour la génération des relations catalogues.

CHAPITRE V

CONCLUSION

Nous avons développé un système d'autorisation (SAGE), généralisé aux données de types complexes et plus particulièrement aux documents. De plus l'approche adoptée consiste à intégrer la gestion des données et des utilisateurs.

SAGE a été développé pour des applications bureautiques. Or ces applications sont généralement conçues pour des entreprises. Aussi, il nous a paru important d'être sûrs que le SGBDG représente bien les mêmes stratégies de contrôle que celles qui gèrent l'environnement réel dans lequel le système opère. Il s'ensuit que notre mécanisme d'autorisation modélise et représente l'entreprise ainsi que les processus d'autorisation des organisations réelles avec l'utilisation complexe de la hiérarchie et de la transmission des privilèges. La structure qui nous a semblé la mieux adaptée pour la représentation d'une entreprise est la structure arborescente qui reflète bien la hiérarchie d'une telle organisation.

SAGE se distingue des mécanismes d'autorisation les plus récemment développés par un certain nombre de critères :

-Protection des données généralisées

-Gestion des utilisateurs par le modèle

-Octroi de certains privilèges implicitement aux utilisateurs d'après leur position hiérarchique.

-Révocation implicite des privilèges associés à un utilisateur en cas de suppression de celui-ci.

-Définition de certains privilèges particuliers : (1) le

privilège ABD qui permet la gestion des utilisateurs, (2) le privilège FORBID qui permet de révoquer certains privilèges accordés implicitement, (3) le privilège de propriété qui donne le droit de transmettre des privilèges sur un objet.

Comme suite à notre travail, nous pouvons envisager les extensions suivantes :

-Négociation des autorisations : dans le monde réel, l'octroi d'un privilège sur un objet peut être le résultat d'une <u>négociation</u> entre deux ou plusieurs utilisateurs. Dans ce cas il serait possible d'avoir comme <u>propriétaire</u> d'un objet tout un <u>groupe</u> d'utilisateurs. Un privilège sera transmis à un utilisateur si tous les membres du groupe sont d'accord, Ceci peut être très important pour certaines informations particulièrement confidentielles. De la même façon, un <u>groupe</u> pourrait être administrateur (ABD). Dans ce cas la modification des informations concernant les utilisateurs serait le résultat d'une négociation entre tous les membres du groupe. Ceci permettrait d'avoir une plus grande sécurité.

-Actuellement l'unité de protection est l'entité ou l'association, bien que la granularité soit affinée au niveau de l'occurrence ou de l'attribut. Le droit de création qui initialement implique celui de propriété, c'est-à-dire de la capacité à transmettre des privilèges, n'est accordé que sur les variables bases de données (entité et association). Il serait intéressant d'étendre la

protection au niveau des types puisque les variables BD sont définies sur ceux-ci.

-La transmission d'un privilège quelconque sur un certain objet pourrait être conditionnée par la possession, par l'utilisateur à qui on veut le transmettre, de ce privilège sur un autre objet. Par exemple, le propriétaire d'un document accordera le privilège P1, sur l'objet O1, à l'utilisateur U1 si U1 est déjà en possession de ce privilège sur l'objet O2. La transmission pourrait également être conditionnée par la possession d'un autre privilège sur le même objet.

-Une autre extension envisageable est la programmation de SACE en Prolog. En effet, ce langage est actuellement très utilisé pour la réalisation de prototypes. D'autre part il est bien adapté à la gestion des arborescences, ce qui permettrait de gérer très facilement la structure représentant les utilisateurs. Nous avons vu (§ 3.3.4) que le problème de la vérification des prédicats devient indécidable dès que ceux-ci deviennent complexes. Prolog permettrait de traiter plus efficacement ce problème.

BIBLIOGRAPHIE

Bibliographie Générale

/ADI82/ : M. ADIBA
New trends in database systems
Nat. conference on databases, Bristol (GB), 1982

/ADI83/ : M. ADIBA, M. LOPEZ, J. PALAZZO
TIGER : a project on Generalized Data Management
IMAG, 1983

/ADI84/ : M. ADIBA, F. AZROU
An authorization mechanism for a generalized DBMS.
3rd Seminar on Distributed Data Sharing Systems
Parma, March 28-30, 1984

/ADI4a/ : M. ADIBA, G.T. NGUYEN
Information processing for CAD/VLSI on a generalized data management system.
Very Large Data Bases (VLDB), Singapour, aout 1984

/ADI4b/ : M. ADIBA, G.T. NGUYEN
Knowledge Engineering for CAD/VLSI on a generalized data management system.
IFIP WG 5.2 Working Conference on Knowledge Engineering in computer aided design, Budapest, september 1984

/AZR82/ : F. AZROU
Confidentialité et autorisation d'accès dans les systèmes de bases de données
IMAG, R.R NO 318, 1982

/BUS81/ : U. BUSSOLATI, G. MARTELLA
A data base approach to modelling and managing security information
VLDB, 1981

/CHA81/ : D. CHAMBERLIN et al.
A history of System R and SQL/DATA system
VLDB, Cannes, 1981

/CHE76/ : P.P CHEN
The entity relationship model - Towards a unified view of data
ACM TODS, vol. 1, no. 1, 1976

/COD79/ : E.F. CODD
Extending the database relational model to capture more meaning
ACM TODS, vol 4, no 4, 1979

/CRA81/ : J.B. CRAMPES
Projet BIG : Les Bases d'Informations Généralisées
INFORSID, la Baule, oct. 1981

/DAN82/ : D. DANIELS et al.
An introduction to distributed query compilation

IBM San Jose Research Lab., San Jose, California, 95193, 1982

/DAT82/ : C.J. DATE
An introduction to database systems. Third edition
Addison-Wesley, Muss., 1982

/DEA82/ : C. DELOBEL, M. ADIBA
Bases de Données et Systèmes Relationnels
Dunod, 1982

/DEN79/ : D.E. DENNING, P.J. DENNING
Data security
ACM Computing Surveys, vol. 11, no. 3, 1979

/DOW79/ : D.DOWNS, G.J. POPEK
Data base management security and INGRES
VLDB, 1979

/ESW76/ : ESWARAN et al.
The notions of consistency and predicate locks
CACM, nov. 1976

/FAG78/ : R. FAGIN
On an authorization mechanism
ACM TODS, Vol. 3, No. 3, 1978

/FER81/ : F.FERNANDEZ
MINER : un outil de base pour la construction de SGBD relationnels - Projet MICROBE
Thèse de 3ème cycle, USMG, 1981

/FOR74/ : J. FORD
Access control by query modification
Final report, computer science 244, 1974, Berkeley

/FRT80/ : L. FERRAT
Traduction d'un langage relationnel de haut niveau en arborescence algébrique binaire
Rapport de D.E.A, IMAG, sept. 1980

/GAL83/ : M. GALY
Application de l'intelligence artificielle à l'optimisation des requêtes relationnelles
Thèse de 3ème Cycle, INPG, 1983

/GRI76/ : P.S. GRIFFTHS, B.W. WADE
An authorization mechanism for a relational database system
ACM TODS, vol. 1, no. 3, sept. 1976

/JOL81/ : V. JOLOBOFF, I. KOWARSKI, M. LOPEZ
Projet bases de données textuelles
IMAG, R.R, 1981

/JOL82/ : V. JOLOBOFF, M. LOPEZ, F. VELEZ
A Generalized Data Base Model
IMAG, 1982

/KLE52/ : S.C. KLEENE
Introduction to Mathematics
Van Nostrand, Princeton, N.J., 1952

/KOW82/ : I. KOWARSKI, M. LOPEZ
The document concept in a database
Proc. of the ACM SIGMOD, Orlando, Florida,1982.

/LAN81/ : E.E. LANDWEHR
Formal models for computer security
ACM Computing Surveys, Vol. 11, No. 3, 1981

/LEE81/ : Y. J. LEE
Définition et réalisation d'un interpréteur de requêtes dans MICROBE
Rapport de D.E.A, IMAG, sept. 1981

/LEE83/ : Y. J. LEE
Une approche pour optimiser les performances dans les SGBD relationnels
Thèse de docteur ingénieur, INPG, nov. 1983

/LOR83/ : R. LORIE, W. PLOUFFE
Complex objects and their use in design transactions.
Database Week, San Jose, May 23-26, 1983

/MIN81/ : N. MINSKY
Synergistic Authorization in database systems
VLDB, 1981

/NGU83/ : G. T. NGUYEN, Y. J. LEE, P. WINNINGER
MICROBE: Un système de gestion de bases de données relationnelles
Journées conception, réalisation et utilisation des SGBD relationnels sur micro-ordinateurs
Université Paul Sabatier, Toulouse, fev. 1983

/OLI83/ : J. OLIVARES
Coopération entre une machine Prolog et une base de données généralisées.
Rapport de D.E.A, Lab. IMAG, sept. 1983

/STO76/ : M. STONEBRAKER, E. WONG, P. KREPS
The design and implentation of INGRES
ACM TODS, Vol. 1, No. 3, 1976

/VEL82/ : F. VELEZ
LAMBDA : un langage de définition et de manipulation de base de données généralisées
Séminaire Bases de Données, ADI, Toulouse, Nov. 1982

/WIL82/ : P.F. WILMS, B.G. LINDSAY
A database authorization mechanism supporting individual and group authorization
Distributed Data Base Sharing Systems, Eds(VAn de Riet, Litwin), North Holland, 1982

/WIN82/ : P. WINNINGER
MIRES : Un optimiseur de rquêtes pour le SGBD relationnel MICROBE
Rapport de D.E.A, IMAG, sep. 1982

/WOO79/ : C. WOOD, E.B. FERNANDEZ
Decentralized authorization in a database system
VLDB, 1979

/ZLO77/ : M. ZLOOF, P. DEJONG
The System for Business Automation (SBA)
CACM, vol. 20, no. 6, June 1977

LISTE DES RAPPORTS TIGRE

[TIGR 1] Présentation générale du projet TIGRE
Janvier 1983

[TIGR 2] The Tigre data model
M. Lopez, J. Palazzo Oliveira, F. Velez - Novembre 1983

[TIGR 3] Proposition de modèle pour la normalisation des documents
G. Bogo, H. Richy, I. Vatton - Mars 1983

[TIGR 4] Recommandations pour l'ergonomie d'un poste de travail attaché à un système bureautique
I. Forestier - Mars 1983

[TIGR 5] La correspondance de schémas entre les modèles TIGRE et relationnel
J. Palazzo Oliveira, F. Velez - Novembre 1983

[TIGR 6] Description des éléments du modèle de document
G. Bogo, H. Richy, I. Vatton - Septembre 1983

[TIGR 7] Programmation en logique pour une base de données généralisées
Nguyen G.T., J. Olivares, P. Winninger
Novembre 1983

[TIGR 8] Machine base de données - Etat de l'Art
M. Burnier - Novembre 1983

[TIGR 9] Caractérisation des formulaires pour une base de données généralisées
C. Collet - Novembre 1983

[TIGR 10] Accès concurrent aux documents
S. Baltas - Janvier 1984

[TIGR 11] Définition formelle du langage LAMBDA
F. Velez - Mars 1984

[TIGR 12] Logic programming for a generalized data management system
M. Adiba, Nguyen G.T. - Janvier 1984

[TIGR 13] Modèle et fonctionnalités pour un système intégrant outils BD et outils de conception
D. Rialhe - Janvier 1984

[TIGR 14] SAGE : Un système d'autorisation généralisé
M. Adiba, F. Azrou - Janvier 1984

[TIGR 15] Coopération de Prolog et d'un SGBD généralisé : Principes et Applications
Nguyen G.T., J. Olivarès, P. Winninger - Avril 1984

[TIGR 16] Two-dimensional editing
V. Joloboff, V. Quint - Juin 1984

[TIGR 17] Aspects logiciels de la communication homme-machine sur les postes de travail individuels
V. Joloboff - Juin 1984

AUTORISATION de SOUTENANCE

VU les dispositions de l'article 3 de l'arrêté du 16 avril 1974,

VU le rapport de présentation de Monsieur le Professeur ADIBA,

Mademoiselle AZROU Fatma

est autorisée à présenter une thèse en soutenance pour l'obtention du titre de DOCTEUR de TROISIEME CYCLE, spécialité "Informatique".

Fait à Grenoble, le 06 juin 1984

Le Président de l'I.N.P.-G

D. BLOCH
Président
de l'Institut National Polytechnique
de Grenoble

P.O. le Vice-Président,