Opérations d'accès par le contenu à une base de documents textuels : application à un environnement de bureau

Claudia Lucia Jimenez Guarin

THESE

Présentée par

Claudia Lucía JIMENEZ GUARIN

pour obtenir le titre de DOCTEUR

de l'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE
(arrêté ministériel du 5 juillet 1984)

Spécialité : Informatique

OPERATIONS D'ACCES PAR LE CONTENU A UNE BASE DE DOCUMENTS TEXTUELS. APPLICATION A UN ENVIRONNEMENT DE BUREAU

Thèse soutenue le 5 juillet 1989

Président	:	J. MOSSIERE
Examinateurs	:	Y. CHIARAMELLA
		C. DELOBEL
		M. LOPEZ
		M. SCHOLL

Thèse préparée au sein du Laboratoire de Génie Informatique

UNIVERSITE Joseph FOURIER (GRENOBLE I)

Président de l'Université :
M. PAYAN Jean Jacques

Année Universitaire 1987 - 1988

MEMBRES DU CORPS ENSEIGNANT DE SCIENCES ET DE GEOGRAPHIE

PROFESSEURS DE 1ère Classe

ARNAUD Paul	Chimie Organique
ARVIEU ROBERT	Physique Nucléaire I.S.N.
AUBERT Guy	Physique C.N.R.S
AURIAULT Jean-Louis	Mécanique
AYANT Yves	Physique Approfondie
BARBIER Marie-Jeanne	Electrochimie
BARJON Robert	Physique Nucléaire ISN
BARNOUD Fernand	Biochimie Macromoléculaire Végétale
BARRA Jean-René	Statistiques-Mathématiques Appliquées
BECKER Pierre	Physique
BEGUIN Claude	Chimie Organique
BELORISKY Elie	Physique
BENZAKEN Claude	Mathématiques Pures
BERARD Pierre	Mathématiques Pures
BERNARD Alain	Mathématiques Pures
BERTRANDIAS Françoise	Mathématiques Pures
BERTRANDIAS Jean-Paul	Mathématiques Pures
BILLET Jean	Géographie
BOELHER Jean-Paul	Mécanique
BONNIER Jane Marie	Chimie Générale
BOUCHEZ Robert	Physique Nucléaire ISN
BRAVARD Yves	Géographie
CARLIER Georges	Biologie Végétale
CAUQUIS Georges	Chimie Organique
CHARDON Michel	Géographie
CHIBON Pierre	Biologie Animale
COHEN ADDAD Jean-Pierre	Physique
COLIN DE VERDIERE Yves	Mathématiques Pures
CYROT Michel	Physique du Solide
DEBELMAS Jacques	Géologie Générale
DEGRANGE Charles	Zoologie
DEMAILLY Jean-Pierre	Mathématiques Pures
DENEUVILLE Alain	Physique
DEPORTES Charles	Chimie Minérale
DOLIQUE Jean-Michel	Physique des Plasmas
DOUCE Roland	Physiologie Végétale
DUCROS Pierre	Cristallographie
FONTAINE Jean-Marc	Mathématiques Pures
GAGNAIRE Didier	Chimie Physique
GERMAIN Jean-Pierre	Mécanique,
GIRAUD Pierre	Géologie
HICTER Pierre	Chimie
IDELMAN Simon	Physiologie Animale
JANIN Bernard	Géographie
JOLY Jean-René	Mathématiques Pures
KAHANE André, détaché	Physique
KAHANE Josette	Physique
KRAKOWIAK Sacha	Mathématiques Appliquées

LAJZEROWICZ Jeanine	Physique
LAJZEROWICZ Joseph	Physique
LAURENT Pierre-Jean	Mathématiques Appliquées
LEBRETON Alain	Mathématiques Appliquées
DE LEIRIS Joël	Biologie
LHOMME Jean	Chimie
LLIBOUTRY Louis	Géophysique
LOISEAUX Jean-Marie	Sciences Nucléaires I.S.N.
LUNA Domingo	Mathématiques Pures
MACHE Régis	Physiologie Végétale
MASCLE Georges	Géologie
MAYNARD Roger	Physique du Solide
OMONT Alain	Astrophysique
OZENDA Paul	Botanique (Biologie Végétale)
PAYAN Jean-Jacques	Mathématiques Pures
PEBAY-PEYROULA Jean-Claude	Physique
PERRIER Guy	Géophysique
PIERRARD Jean-Marie	Mécanique
PIERRE Jean-Louis	Chimie Organique
RENARD Michel	Thermodynamique
RINAUDO Marguerite	Chimie CERMAV
ROSSI André	Biologie
SAXOD Raymond	Biologie Animale
SENGEL Philippe	Biologie Animale
SERGERAERT Francis	Mathématiques Pures
SOUCHIER Bernard	Biologie
SOUTIF Michel	Physique
STUTZ Pierre	Mécanique
TRILLING Laurent	Mathématiques Appliquées
VALENTIN Jacques	Physique Nucléaire I.S.N.
VAN CUTSEM Bernard	Mathématiques Appliquées
VIALON Pierre	Géologie

PROFESSEURS de 2ème Classe

ADIBA Michel	Mathématiques Pures
ANTOINE Pierre	Géologie
ARMAND Gilbert	Géographie
BARET Paul	Chimie
BLANCHI J.Pierre	STAPS
BLUM Jacques	Mathématiques Appliquées
BOITET Christian	Mathématiques Appliquées
BORNAREL Jean	Physique
BRUANDET J.François	Physique
BRUGAL Gérard	Biologie
BRUN Gilbert	Biologie
CASTAING Bernard	Physique
CERFF Rudiger	Biologie
CHIARAMELLA Yves	Mathématiques Appliquées
COURT Jean	Chimie
DUFRESNOY Alain	Mathématiques Pures
GASPARD François	Physique
GAUTRON René	Chimie
GENIES Eugène	Chimie
GIDON Maurice	Géologie
GIGNOUX Claude	Sciences Nucléaires
GILLARD Roland	Mathématiques Pures
GIORNI Alain	Sciences Nucléaires
GONZALEZ SPRINBERG Gérardo	Mathématiques Pures
GUIGO Maryse	Géographie
GUMUCHAIN Hervé	Géographie
GUITTON Jacques	Chimie

HACQUES Gérard	Mathématiques Appliquées
HERBIN Jacky	Géographie
HERAULT Jeanny	Physique
JARDON Pierre	Chimie
JOSELEAU Jean-Paul	Biochimie
KERCKHOVE Claude	Géologie
LONGEQUEUE Nicole	Sciences Nucléaires I.S.N.
LUCAS Robert	Physique
MANDARON Paul	Biologie
MARTINEZ Francis	Mathématiques Appliquées
NEMOZ Alain	Thermodymamique CNRS - CRTBT
OUDET Bruno	Mathématiques Appliquées
PECHER Arnaud	Géologie
PELMONT Jean	Biochimie
PERRIN Claude	Sciences Nucléaires I.S.N.
PFISTER Jean-Claude	Physique du Solide
PIBOULE Michel	Géologie
RAYNAUD Hervé	Mathématiques Appliquées
RICHARD Jean-Marc	Physique
RIEDTMANN Christine	Mathématiques Pures
ROBERT Gilles	Mathématiques Pures
ROBERT Jean-Bernard	Chimie Physique
SARROT-REYNAULD Jean	Géologie
SAYETAT Françoise	Physique
SERVE Denis	Chimie
STOECKEL Frédéric	Physique
SCHOLL Pierre-Claude	Mathématiques Appliquées
SUBRA Robert	Chimie
VALLADE Marcel	Physique
VIDAL Michel	Chimie Organique
VIVIAN Robert	Géographie
VOTTERO Philippe	Chimie

MEMBRES DU CORPS ENSEIGNANT DE L' IUT 1

PROFESSEURS de 1ère Classe

BUISSON Roger	Physique IUT 1
DODU Jacques	Mécanique Appliquée IUT 1
NEGRE Robert	Génie Civil IUT 1
NOUGARET Marcel	Automatique IUT 1
PERARD Jacques	EEA. IUT 1

PROFESSEURS de 2ème classe

BOUTHINON Michel	EEA. IUT 1
CHAMBON René	Génie Mécanique IUT 1
CHEHIKIAN Alain	EEA. IUT 1
CHENAVAS Jean	Physique IUT 1
CHOUTEAU Gérard	Physique IUT 1
CONTE René	Physique IUT 1
GOSSE Jean-Pierre	EEA.IUT 1
GROS Yves	Physique IUT 1
KUHN Gérard, (Détaché)	Physique IUT 1
MAZUER Jean	Physique IUT 1
MICHOULIER Jean	Physique IUT 1
MONLLOR Christian	EEA.IUT 1
PEFFEN René	Métallurgie IUT 1
PERRAUD Robert	Chimie IUT 1
PIERRE Gérard	Chimie IUT 1
TERRIEZ Jean-Michel	Génie Mécanique IUT 1
TOUZAIN Philippe	Chimie IUT 1
VINCENDON Marc	Chimie IUT 1

PROFESSEURS DE PHARMACIE

AGNIUS-DELORD Claudine	Physique	Faculté La Tronche
ALARY Josette	Chimie Analytique	Faculté La Tronche
BERIEL Hélène	Physiologie et Pharmacologie	Faculté La Tronche
CUSSAC Max	Chimie Therapeutique	Faculté La Tronche
DEMENGE Pierre	Pharmacodynamie	Faculté La Tronche
FAVIER Alain	Biochimie	C.H.R.G.
JEANNIN Charles	Pharmacie Galénique	Faculté Meylan
LATURAZE Jean	Biochimie	Faculté La Tronche
LUU DUC Cuong	Chimie Générale	Faculté La Tronche
MARIOTTE Anne-Marie	Pharmacognosie	Faculté La Tronche
MARZIN Daniel	Toxicologie	Faculté Meylan
RENAUDET Jacqueline	Bactériologie	Faculté La Tronche
ROCHAT Jacques	Hygiéne et Hydrologie	Faculté La Tronche
SEIGLE-MURANDI Françoise	Botanique et Cryptogamie	Faculté Meylan
VERAIN Alice	Pharmacie Galénique	Faculté Meylan

MEMBRES DU CORPS ENSEIGNANT DE MEDECINE

PROFESSEURS CLASSE EXEPTIONNELLE ET 1ère CLASSE

AMBLARD Pierre	Dermatologie	C.H.R.G.
AMBROISE-THOMAS Pierre	Parasitologie	C.H.R.G.
BEAUDOING André	Pédiatrie-Puericulture	C.H.R.G.
BEZEZ Henri	Orthopédie-Traumatologie	Hopital SUD
BONNET Jean-Louis	Ophtalmologie	C.H.R.G.
BOUCHET Yves	Anatomie	Faculté La Merci
	Chirurgie Générale et Digestive	C.H.R.G.
BUTEL Jean	Orthopédie-Traumatologie	C.H.R.G.
CHAMBAZ Edmond	Biochimie	C.H.R.G.
CHAMPETIER Jean	Anatomie-Topographique et Appliquée	C.H.R.G.
CHARACHON Robert	O.R.L.	C.H.R.G.
COLOMB Maurice	Immunologie	Hopital sud
COUDERC Pierre	Anatomie-Pathologique	C.H.R.G.
DELORMAS Pierre	Pneumophtisiologie	C.H.R.G.
DENIS Bernard	Cardiologie	C.H.R.G.
GAVEND Michel	Pharmacologie	Faculté La Merci
HOLLARD Daniel	Hématologie	C.H.R.G.
LATREILLE René	Chirurgie Thoracique et Cardiovasculaire	C.H.R.G.
LE NOC Pierre	Bactériologie-Virologie	C.H.R.G.
MALINAS Yves	Gynécologie et Obstétrique	C.H.R.G.
MALLION Jean-Michel	Médecine du Travail	C.H.R.G.
MICOUD Max	Clinique Médicale et Maladies Infectieuses	C.H.R.G.
MOURIQUAND Claude	Histologie	Faculté La Merci
PARAMELLE Bernard	Pneumologie	C.H.R.G.
PERRET Jean	Neurologie	C.H.R.G.
RACHAIL Michel	Hépato-Gastro-Entérologie	C.H.R.G.
DE ROUGEMONT Jacques	Neurochirurgie	C.H.R.G.
SARRAZIN Roger	Clinique Chirurgicale	C.H.R.G.
STIEGLITZ Paul	Anestésiologie	C.H.R.G.
TANCHE Maurice	Physiologie	Faculté La Merci
VIGNAIS Pierre	Biochimie	Faculté La Merci

PROFESSEURS 2ème CLASSE

BACHELOT Yvan	Endocrinologie	C.H.R.G.
BARGE Michel	Neurochirurgie	C.H.R.G.
BENABID Alim Louis	Biophysique	Faculté La Merci
BENSA Jean-Claude	Immunologie	Hopital Sud
BERNARD Pierre	Gynécologie-Obstétrique	C.H.R.G.
BESSARD Germain	Pharmacologie	ABIDJAN
BOLLA Michel	Radiothérapie	C.H.R.G.
BOST Michel	Pédiatrie	C.H.R.G.
BOUCHARLAT Jacques	Psychiatrie Adultes	Hopital Sud
BRAMBILLA Christian	Pneumologie	C.H.R.G.
CHIROSSEL Jean-Paul	Anatomie-Neurochirurgie	C.H.R.G.
COMET Michel	Biophysique	Faculté La Merci
CONTAMIN Charles	Chirurgie Thoracique et Cardiovasculaire	C.H.R.G.
CORDONNIER Daniel	Néphrologie	C.H.R.G.
COULOMB Max	Radiologie	C.H.R.G.
CROUZET Guy	Radiologie	C.H.R.G.
DEBRU Jean-Luc	Médecine Interne et Toxicologie	C.H.R.G.
DEMONGEOT Jacques	Biostatistiques et Informatique Médicale	Faculté La Merci
DUPRE Alain	Chirurgie Générale	C.H.R.G.
DYON Jean-François	Chirurgie Infantile	C.H.R.G.
ETERRADOSSI Jacqueline	Physiologie	Faculté La Merci
FAURE Claude	Anatomie et Organogénèse	C.H.R.G.
FAURE Gilbert	Urologie	C.H.R.G.
FOURNET Jacques	Hépato-Gastro-Entérologie	C.H.R.G.
FRANCO Alain	Médecine Interne	C.H.R.G.
GIRARDET Pierre	Anesthésiologie	C.H.R.G.
GUIDICELLI Henri	Chirurgie Générale et Vasculaire	C.H.R.G.
GUIGNIER Michel	Thérapeutique et Réanimation Médicale	C.H.R.G.
HADJIAN Arthur	Biochimie	Faculté La Merci
HALIMI Serge	Endocrinologie et Maladies Métaboliques	C.H.R.G.
HOSTEIN Jean	Hépato-Gastro-Entérologie	C.H.R.G.
HUGONOT Robert	Médecine Interne	C.H.R.G.
JALBERT Pierre	Histologie-Cytogénétique	C.H.R.G.
JUNIEN-LAVILLAUROY Claude	O.R.L.	C.H.R.G.
KOLODIE Lucien	Hématologie Biologique	C.H.R.G.
LETOUBLON Christian	Chirurgie Générale	C.H.R.G.
MACHECOURT Jacques	Cardiologie et Maladies Vasculaires	C.H.R.G.
MAGNIN Robert	Hygiène	C.H.R.G.
MASSOT Christian	Médecine Interne	C.H.R.G.
MOUILLON Michel	Ophtalmologie	C.H.R.G.
PELLAT Jacques	Neurologie	C.H.R.G.
PHELIP Xavier	Rhumatologie	C.H.R.G.
RACINET Claude	Gynécologie-Obstétrique	Hopital Sud
RAMBAUD Pierre	Pédiatrie	C.H.R.G.
RAPHAEL Bernard	Stomatologie	C.H.R.G.
SCHAERER René	Cancérologie	C.H.R.G.
SEIGNEURIN Jean-Marie	Bactériologie-Virologie	Faculté La Merci
SELE Bernard	Cytogénétique	Faculté La Merci
SOTTO Jean-Jacques	Hématologie	C.H.R.G.
STOEBNER Pierre	Anatomie Pathologique	C.H.R.G.
VROUSOS Constantin	Radiothérapie	C.H.R.G.

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Président : Georges LESPINARD

Année 1988

Professeurs des Universités

BARIBAUD Michel	ENSERG
BARRAUD Alain	ENSIEG
BAUDELET Bernard	ENSPG
BEAUFILS Jean-Pierre	ENSEEG
BLIMAN Samuel	ENSERG
BLOCH Daniel	ENSPG
BOIS Philippe	ENSHMG
BONNETAIN Lucien	ENSEEG
BOUVARD Maurice	ENSHMG
BRISSONNEAU Pierre	ENSIEG
BRUNET Yves	IUFA
CAILLERIE Denis	ENSHMG
CAVAIGNAC Jean-François	ENSPG
CHARTIER Germain	ENSPG
CHENEVIER Pierre	ENSERG
CHERADAME Hervé	UFR PGP
CHOVET Alain	ENSERG
COHEN Joseph	ENSERG
COUMES André	ENSERG
DARVE Félix	ENSHMG
DELLA-DORA Jean	ENSIMAG
DEPORTES Jacques	ENSPG
DOLMAZON Jean-Marc	ENSERG
DURAND Francis	ENSEEG
DURAND Jean-Louis	ENSIEG
FOGGIA Albert	ENSIEG
FONLUPTJean	ENSIMAG
FOULARD Claude	ENSIEG
GANDINI Alessandro	UFR PGP
GAUBERT Claude	ENSPG
GENTIL Pierre	ENSERG
GREVEN Hélène	IUFA
GUERIN Bernard	ENSERG
GUYOT Pierre	ENSEEG
IVANES Marcel	ENSIEG
JAUSSAUD Pierre	ENSIEG
JOUBERT Jean-Claude	ENSPG
JOURDAIN Geneviève	ENSIEG
LACOUME Jean-Louis	ENSIEG
LESIEUR Marcel	ENSHMG
LESPINARD Georges	ENSHMG
LONGEQUEUE Jean-Pierre	ENSPG
LOUCHET François	ENSIEG
MASSE Philippe	ENSIEG
MASSELOT Christian	ENSIEG
MAZARE Guy	ENSIMAG
MOREAU René	ENSHMG
MORET Roger	ENSIEG
MOSSIERE Jacques	ENSIMAG
OBLED Charles	ENSHMG
OZIL Patrick	ENSEEG
PARIAUD Jean-Charles	ENSEEG
PERRET René	ENSIEG
PERRET Robert	ENSIEG
PIAU Jean-Michel	ENSHMG
POUPOT Christian	ENSERG
RAMEAU Jean-Jacques	ENSEEG
RENAUD Maurice	UFR PGP
ROBERT André	UFR PGP
ROBERT François	ENSIMAG
SABONNADIERE Jean-Claude	ENSIEG
SAUCIER Gabrielle	ENSIMAG
SCHLENKER Claire	ENSPG
SCHLENKER Michel	ENSPG
SILVY Jacques	UFR PGP
SIRIEYS Pierre	ENSHMG
SOHM Jean-Claude	ENSEEG
SOLER Jean-Louis	ENSIMAG
SOUQUET Jean-Louis	ENSEEG
TROMPETTE Philippe	ENSHMG
VEILLON Gérard	ENSIMAG
ZADWORNY François	ENSERG

Professeur Université des Sciences Sociales (Grenoble II)

BOLLIET Louis

Personnes ayant obtenu le diplôme d'HABILITATION A DIRIGER DES RECHERCHES

BECKER Monique
BINDER Zdenek
CHASSERY Jean-Marc
CHOLLET Jean-Pierre
COEY John
COLINET Catherine
COMMAULT Christian
CORNUEJOLS Gérard
COULOMB Jean- Louis
DALARD Francis
DANES Florin
DEROO Daniel
DIARD Jean-Paul
DION Jean-Michel
DUGARD Luc
DURAND Madeleine
DURAND Robert
GALERIE Alain
GAUTHIER Jean-Paul
GENTIL Sylviane
GHIBAUDO Gérard
HAMAR Sylvaine
HAMAR Roger
LADET Pierre
LATOMBE Claudine
LE GORREC Bernard
MADAR Roland
MULLER Jean
NGUYEN TRONG Bernadette
PASTUREL Alain
PLA Fernand
ROUGER Jean
TCHUENTE Maurice
VINCENT Henri

Chercheurs du C.N.R.S
Directeurs de recherche 1ère Classe

CARRE René
FRUCHART Robert
HOPFINGER Emile
JORRAND Philippe
LANDAU Ioan
VACHAUD Georges
VERJUS Jean-Pierre

Directeurs de recherche 2ème Classe

ALEMANY Antoine
ALLIBERT Colette
ALLIBERT Michel
ANSARA Ibrahim
ARMAND Michel
BERNARD Claude
BINDER Gilbert
BONNET Roland
BORNARD Guy
CAILLET Marcel
CALMET Jacques
COURTOIS Bernard
DAVID René
DRIOLE Jean
ESCUDIER Pierre
EUSTATHOPOULOS Nicolas
GUELIN Pierre
JOUD Jean-Charles
KLEITZ Michel
KOFMAN Walter
KAMARINOS Georges
LEJEUNE Gérard
LE PROVOST Christian
MADAR Roland
MERMET Jean
MICHEL Jean-Marie
MUNIER Jacques
PIAU Monique
SENATEUR Jean-Pierre
SIFAKIS Joseph
SIMON Jean-Paul
SUERY Michel
TEODOSIU Christian
VAUCLIN Michel
WACK Bernard

Personnalités agrées à titre permanent à diriger des travaux de recherche (décision du conseil scientifique)

E.N.S.E.E.G
CHATILLON Christian
HAMMOU Abdelkader
MARTIN GARIN Régina
SARRAZIN Pierre
SIMON Jean-Paul

E.N.S.E.R.G
BOREL Joseph

E.N.S.I.E.G
DESCHIZEAUX Pierre
GLANGEAUD François
PERARD Jacques
REINISCH Raymond

E.N.S.H.G
ROWE Alain

E.N.S.I.M.A.G
COURTIN Jacques

E.F.P.
CHARUEL Robert

C.E.N.G
CADET Jean
COEURE Philippe
DELHAYE Jean-Marc
DUPUY Michel
JOUVE Hubert
NICOLAU Yvan
NIFENECKER Hervé
PERROUD Paul
PEUZIN Jean-Claude
TAIB Maurice
VINCENDON Marc

Laboratoires extérieurs

C.N.E.T
DEVINE Rodericq
GERBER Roland
MERCKEL Gérard
PAULEAU Yves

Je tiens à remercier

Monsieur Jacques MOSSIERE, Professeur à l'I.N.P.G. et Directeur de l'E.N.S.I.M.A.G., qui me fait l'honneur de présider le jury. Je lui suis très reconnaissante pour le soutien qu'il m'a toujours apporté pendant ces années.

Monsieur Claude DELOBEL, Professeur à l'Université d'ORSAY, pour avoir accepté de juger ce travail et participer au jury, et pour ses remarques et ses critiques constructives.

Monsieur Michel SCHOLL, Directeur de Recherche à l'INRIA, pour avoir bien voulu juger ce travail, pour ses critiques et ses remarques qui ont fait progresser ce document.

Monsieur Yves CHIARAMELLA, Directeur du Laboratoire de Génie Informatique et directeur de cette thèse. Qu'il trouve ici toute ma reconnaissance pour avoir bien voulu m'accueillir dans son équipe de recherche et pour la confiance qu'il m'a toujours témoignée. Je tiens à le remercier particulièrement pour les multiples lectures et corrections qu'il a fait de ce manuscrit, ainsi que pour les nombreuses discussions qui ont grandement contribué à la qualité de cette thèse.

Monsieur Mauricio LOPEZ, Ingénieur de Recherche au Centre de Recherche BULL-IMAG et responsable du projet ESPRIT DOEOIS dans le cadre duquel ce travail a été réalisée. Je profite de cette occasion pour lui exprimer ma profonde gratitude pour la confiance, le soutien et l'amitié qu'il m'a toujours témoigné. Son encouragement permanent, ses idées et ses conseils m'ont permis de mener à bien ce travail et ont contribué grandement à la qualité de ce document.

L'ensemble des participants du projet DOEOIS, en particulier Christian LENNE, Esperanza PEDRAZA, Francisca ANTUNES, Jean Pierre MARTIN, José Valdeni DE LIMA, Paul BERARD et Henry GALY, pour les discussions, la collaboration et l'amitié qu'ils m'ont toujours témoigné.

Je remercie Didier TERRAL, Ingénieur à la Société BULL, pour son aide et sa collaboration pendant toutes nos expériences avec le filtre SCHUSS.

Je remercie profondément Jean CHOUANARD, qui pendant plus d'une année a travaillé sur la réalisation du prototype sur SCHUSS. Que sa patience, son esprit de bricoleur et les longues heures passées à découvrir les mystères et les secrets de SCHUSS soient ici largement reconnus.

Je remercie également tous les membres de l'équipe Systèmes Intelligents de Recherche d'Information, pour leur soutien et leur amitié pendant ces années. Un

grand merci en particulier à Patrick, pour son amitié, sa bonne humeur et son esprit toujours conciliant, qui ont fait de notre séjour dans le même bureau une expérience inoubliable.

Enfin, je tiens à remercier toute la colonie latino-américaine, pour son amitié et son appui. Un grand merci à Rafael, Martha, Mario, Jorge, l'Esper, Alejo, Olga, Gustavo, Rodrigo et à tous les autres.

Je ne peux pas oublier ici Germán, sans qui cette thèse n'aurait jamais vu le jour. Merci infiniment pour le soutien continu, les longues heures de discussions et de lectures qu'il a dédié à ce travail. Sans son aide et sa présence je ne serais pas allée au but. Je tien enfin à remercier ma famille, qui même de si loin m'a toujours encouragée et soutenue moralement. Qu'ils trouvent tous ici une récompense à leurs efforts.

TABLE DE MATIERES

1 . LA BUREAUTIQUE ET LA RECHERCHE D'INFORMATION

1.1. Introduction

Le but d'un Système de Recherche d'Information (SRI) est la manipulation et l'accès à des documents d'un domaine d'application. Le contexte le plus fréquent de ces applications est celui des systèmes documentaires. L'ensemble des documents traités par un SRI constitue ce qu'on appelle un corpus documentaire. Les SRI traditionnels ont été développés, en général, sur des corpus relatifs à un thème particulier (le corpus est donc homogène) ou sur des corpus plus larges, couvrant un ensemble de thèmes.

Les SRI deviennent l'outil de base qui permet l'accès au corpus via des requêtes relatives à un sujet donné. Ce type d'accès par le contenu implique, dans un SRI, une modélisation du *contenu sémantique* des documents (modèle de représentation des connaissances). Dans un SRI, les documents gérés sont de type texte et ils sont caractérisés par une forte structuration. Les principales caractéristiques de ces corpus sont l'absence d'évolution du contenu des documents, la prédominance des opérations de sélection et des opérations d'adjonction de nouveaux documents, et une typologie prédéterminée des documents [Chi83].

Un environnement bien différent, celui du **bureau**, fait à l'heure actuelle l'objet de nombreuses études. L'intégration d'outils et de

techniques appliqués dans d'autres systèmes doit être adaptée à ces nouveaux besoins ; ceci est particulièrement vrai pour les Bases de Données qui ont beaucoup apporté en ce qui concerne la gestion de l'intégrité, le stockage, et la reprise. Il reste beaucoup à réaliser, en particulier, l'étude de techniques adaptées à l'accès des nouveaux types d'information : l'information multimédia. Dans ces systèmes, la notion de contenu des documents est classiquement limitée à celle du *contenu textuel* (chaînes de caractères constituant le texte).

Les SRI proposent des techniques d'accès par le contenu sémantique aux bases documentaires; ceci constitue un aspect très important à considérer dans un environnement bureautique. Notre travail a pour objectif l'étude de l'intégration des fonctionnalités d'un SRI dans un **Serveur d'Information Bureautique (SIB)**. Ceci implique, en particulier, la définition de fonctions d'accès par le contenu aux documents manipulés dans un environnement de bureau. La principale caractéristique d'un tel environnement est l'ouverture vers des applications très diverses, où les corpus sont dynamiques, et où le contenu des documents n'est pas nécessairement restreint à un domaine donné.

Ce travail concerne la définition et l'implantation des opérations d'accès par le contenu textuel et le contenu sémantique aux documents du bureau, dans le cadre d'un **Serveur d'Information Bureautique** générique, le **SIB**. Nous avons choisi comme cadre d'étude et d'expérimentation le Serveur OIS, conçu dans le cadre du projet ESPRIT DOEOIS ("Design and Operational Evaluation of Office Information Servers"). L'objet de l'étude faite dans ce projet est la spécification et l'évaluation des services et modèles de données à offrir dans un tel environnement, ainsi que le développement d'un prototype.

Pour conduire notre étude des fonctions d'accès par le contenu aux documents bureautiques, nous avons considéré l'expérience que nous avons acquise dans le contexte de deux projets : le projet IOTA, un Système Intelligent de Recherche d'Informations, et le projet TIGRE, un SGBD conçu pour la manipulation d'information généralisée, en particulier d'information

multimédia. Le premier donne des méthodes pour l'accès par le contenu sémantique à des documents, le deuxième offre des outils de gestion des documents multimédia et d'accès par le contenu textuel.

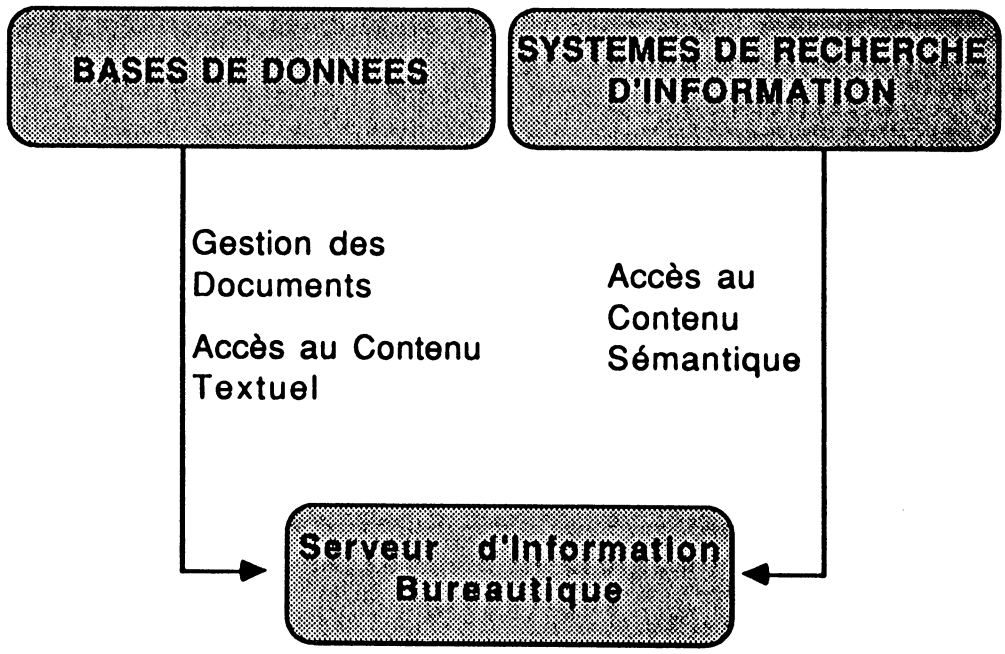


figure 1.1

Dans la suite de ce chapitre, nous faisons tout d'abord une première présentation du contexte dans lequel nous nous situons et les caractéristiques générales du bureau et du SIB. Nous présentons ensuite les généralités des Systèmes de Recherche d'Information, afin de montrer les fonctionnalités qu'ils peuvent apporter.

Finalement, nous donnons une description des fonctions d'accès par le contenu utiles dans le contexte bureautique, dont l'étude et la réalisation constituent l'essentiel de notre travail.

Ce document est organisé de la manière suivante :

Dans le chapitre 2 nous présentons les principaux modèles pour la représentation du contenu d'un corpus documentaire dans le cadre des SRI, ainsi que quelques exemples de systèmes les plus représentatifs de ces approches, les méthodes d'accès et de stockage qu'ils utilisent. Finalement, nous faisons une présentation de l'architecture du système prototype IOTA qui constitue notre système de référence en matière de SRI. Nous détaillons en particulier ses fonctions d'Indexation et d'Interrogation.

Dans le chapitre 3 nous présentons, de manière similaire à celle du chapitre 2, les modèles de documents multimédia utilisés actuellement dans le cadre des applications bureautiques. Nous montrons les techniques de stockage et les fonctions d'accès qui leurs sont associées. Dans ces techniques nous soulignons celles qui utilisent des machines spécialisées pour l'accès aux documents. Nous y détaillons des représentants des principales approches.

Dans le chapitre 4 nous définissons les opérations de recherche par le contenu des documents du Serveur d'Information Bureautique. Nous y définissons les méthodes d'accès et de stockage correspondantes.

Le chapitre 5 est dédié aux réalisations et expérimentations des opérations et techniques que nous avons définies. L'intégration de ces outils, souvent coûteux en place et en temps de calcul, implique l'étude de plusieurs alternatives d'implantation.

Nous avons considéré aussi bien des techniques logicielles que matérielles pour réaliser cette intégration. SCHUSS, un matériel micro-programmable orienté vers le filtrage, est la machine que nous avons expérimentée pour évaluer l'approche matérielle.

Pour mieux situer d'emblée notre problématique, il nous semble utile de situer les principales caractéristiques des systèmes impliqués dans notre étude : le Serveur d'Information Bureautique et les Systèmes de Recherche d'Information.

1.2. L'Information Bureautique

Le SIB est un ensemble de fonctionnalités qui donnent un support au développement d'applications dans un bureau. Ces applications peuvent être aussi variées que les activités qui s'y déroulent : édition de rapports, ventes, manipulation de messages, suivi de clients, etc. Il faut donc fournir des outils généraux, qui permettent la modélisation des **activités de bureau** et de l'information qui y circule.

Les Serveurs de Bureau sont construits autour d'autres serveurs spécifiques, comme les serveurs d'impression, de messages, de stockage, etc. On y trouve une grande variété de types d'information, de complexité variable. Ces types peuvent être des combinaisons d'objets multimédia : voix, textes, images, graphiques. L'unité d'information multimédia est le "document multimédia" [Chr85], qui est composé d'attributs et d'objets multimédia. Les fonctions fondamentales sur ces documents sont la préparation, l'édition, le stockage, la transmission et la récupération.

Pour l'intégration des documents dans le SIB nous avons considéré deux aspects:

- L'intégration des documents et de leurs attributs au niveau du modèle de données.

- L'intégration à niveau fonctionnel, par la spécification des opérateurs propres aux documents, leur expression dans le langage de manipulation de données et les méthodes d'accès et de stockage correspondantes.

Ceci permet d'avoir, au niveau des applications, une vision homogène des objets manipulés par le Serveur.

1.2.1. L'Architecture du SIB

Le Serveur d'Information Bureautique (SIB), est connecté par un réseau local aux stations de travail où se trouvent les applications. Le SIB est l'outil de base pour la modélisation des applications de bureau. Il doit donc supporter aussi bien les aspects statiques que les aspects dynamiques : l'information du bureau est ouverte à tous les domaines et évolue selon les besoins des **Procédures de Bureau** (séquences d'**activités**).

Une activité est une unité non décomposable du travail du bureau, par exemple l'enregistrement d'une vente. Les Procédures de Bureau peuvent être démarrées manuellement ou automatiquement, et peuvent être structurées ou partiellement définies [ESP86b] [ALP89] [Ped88]. L'information gérée par ces procédures et ces activités est fortement dynamique et peut être de type multimédia, comme c'est le cas des documents. Ceci implique une définition statique et dynamique du modèle de données des applications.

Dans le but d'intégrer toutes ces fonctionnalités d'une manière uniforme, le Serveur a son propre modèle de données. Par exemple, dans le cas du serveur OIS, c'est le **Fact Model (FM)** [Sac86], qui est un modèle fonctionnel pour spécifier et manipuler des données. En FM a été intégré le modèle de documents ODA, afin d'offrir aux applications, et aux autres serveurs, une forme standard de communication et de définition des documents. Les données sont manipulées à travers d'un langage appelé FML, divisé en deux composants, FMDL ("Fact Model Definition Language"), pour la définition des aspects statiques, et FMQL ("Fact Model Query Language") [ESP88], pour l'interrogation et la mise à jour de la base. Dans FML ont été intégrés les opérateurs de recherche par le contenu des documents, ce qui permet de réaliser des accès spécifiques à ce type de données, en plus des opérateurs de manipulation des autres entités de FM. Une description sommaire se trouve dans l'Annexe A et dans [Ped88].

Nous pouvons distinguer dans l'architecture de l'OIS les modules suivants (voir fig. 1.2) :

- Le Gestionnaire de Communications, qui est chargé des échanges entre le Serveur et les stations de travail.

ARCHITECTURE DE l'OIS

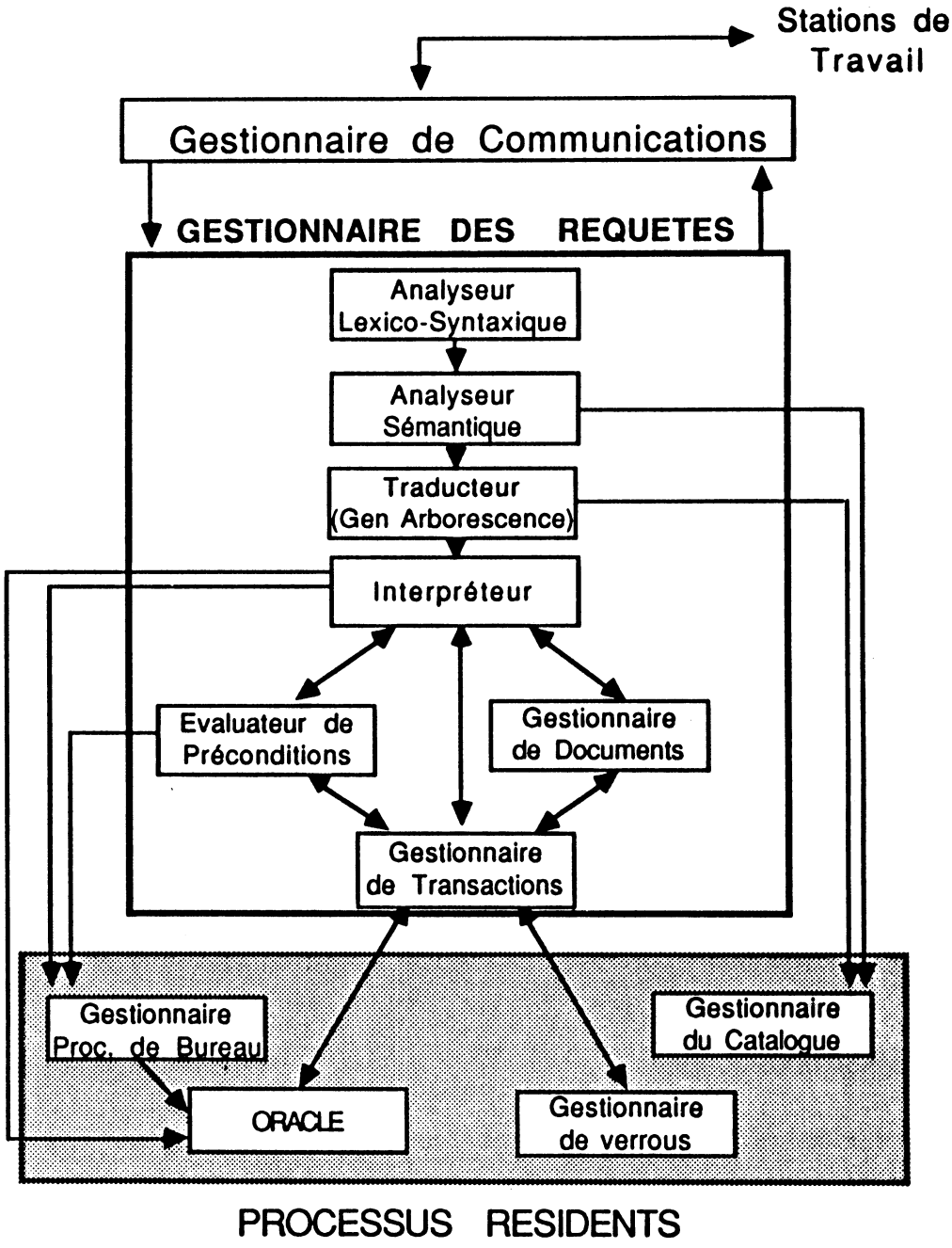


figure 1.2

- L'Analyseur Lexico-Syntaxique pour les instructions en FML.

- L'Analyseur Sémantique, qui vérifie les instructions FML et ajoute les informations pour les phases suivantes.

- Le Traducteur, qui transforme les requêtes FM en un arbre d'opérateurs.

- L'Interprète, qui est chargé de l'exécution des opérations. C'est le noyau du Système, et il contrôle leur bon déroulement. En particulier, il est responsable de l'assignation des tâches.

- Le Gestionnaire de Transactions, qui est responsable de la maintenance de la cohérence et de l'intégrité des données. Il doit garantir aux applications l'exécution complète des activités et la reprise après pannes. Grâce à cette fonctionnalité de l'OIS, l'usager, lorsqu'il définit une application, peut décider quelles sont les opérations sur lesquelles il veut assurer l'indivisibilité.

- Le Gestionnaire de Verrous, qui gère l'accès concurrent aux données par plusieurs transactions.

- Le Gestionnaire d'activités et Procédures de Bureau. Il assure la bonne réalisation des activités définies dans le modèle dynamique des applications. Il peut y avoir des activités entièrement automatiques, partiellement automatiques, ou manuelles.

- Le Gestionnaire de Documents, qui est chargé des opérations définies sur les objets de type Document. Il s'agit de fournir à l'interprète une interface de manipulation de ces objets tout à fait uniforme par rapport aux autres objets du système. Ceci garantit qu'au niveau externe du Serveur, les documents sont vus comme des entités quelconques, et que leur complexité est transparente pour les autres niveaux.

Les opérations sur les documents sont : l'acquisition, la sélection et la restitution. L'opération d'acquisition de documents comprend le décodage, le stockage et la construction des index. L'opération de sélection permet l'accès aux documents par leurs attributs et par leur contenu. L'opération de restitution permet la communication de documents à l'extérieur, par exemple vers des éditeurs ou des imprimantes, au niveau application.

Notre intérêt porte plus particulièrement sur les opérations concernant les documents, et notamment l'accès par leur contenu. Nous décrivons plus en détail les opérations du Gestionnaire de Documents ci-dessous.

1.2.2. Le Gestionnaire de Documents du SIB

Le Gestionnaire de Documents est chargé de gérer dans le SIB le stockage, la manipulation et l'interrogation des objets de type document. Les documents typiques produits dans un bureau sont supposés de taille petite ou moyenne, contenant des informations multimédia et ayant une structure complexe. Le document est considéré comme une unité indivisible, aussi bien pour l'accès que pour le stockage.

Dans le but d'assurer la communication de documents entre différentes applications ou entre serveurs, le SIB offre un support de gestion de documents compatible avec le standard ODA/ODIF [ECM85]. Le modèle ODA (Office Document Architecture) est un standard pour la définition de la structure des documents. Les documents ODA sont codés en ODIF, norme pour l'échange de documents. Dans le SIB tous les documents seront donc conformes à ODA.

Tout objet ODA est composé d'un ensemble d'attributs externes, regroupés dans le **Profil du Document**. Les objets ODA peuvent avoir aussi un contenu, le contenu du document, structuré sous une forme hiérarchique. Il peut y avoir plusieurs manières de

représenter la structure d'un contenu de document : sa structure logique, sa structure de présentation (mise en page, multiples polices, etc) et sa structure générique, définissant la sémantique des composants structurels [ECM85]. Dans ODA des niveaux de conformité entre documents ont été définis, selon l'ensemble des structures utilisées pour décrire le contenu. Dans le SIB nous considérons les documents ODA contenant le Profil, la structure logique, et optionnellement la structure générique.

Les opérations offertes par le Gestionnaire de Documents sont donc des opérations sur des objets ODA [ESP88a] :

- Acquisition d'un objet ODA : quand un nouveau document est transmis au SIB, il est décodé, indexé et stocké. Un Identificateur interne au SIB lui est affecté afin de le distinguer des autres entités, et de permettre ainsi sa récupération lors de l'interrogation. Les détails de cette opération sont décrits dans le chapitre 5, concernant la réalisation.

- Interrogation sur les attributs externes du document (le Profil du document ODA), c'est à dire, auteur, titre, etc. Cette opération correspond à l'accès à des données de type classique.

- Navigation ("browsing") sur la structure logique du document.

- Accès par le contenu des documents, en exprimant un sujet ou un "pattern" de sélection. Dans le cas standard le contenu correspond au texte et les "patterns" de sélection sont des expressions de chaînes de caractères. Pour certaines applications ceci n'est pas suffisant, et l'accès au contenu des documents utilise un modèle qui tient compte de leur sémantique.

- Remplacement et copie des objets ODA.

- Destruction d'objets ODA.

- Restitution de documents, ce qui correspond à l'opération complémentaire à l'Acquisition. Le document est passé par le SIB à l'application via le module de communications.

Dans ce travail nous nous sommes intéressés en particulier aux fonctions d'**Accès par le Contenu** des documents. Pour la définition de ces fonctions il est intéressant d'étudier les principes et apports des services offerts par les Systèmes de Recherche d'Information. Nous décrivons dans la prochaine partie les principes des SRI et ensuite les fonctions d'accès par le contenu qui ont été définies pour le SIB.

1.3. Caractéristiques Générales des Systèmes de Recherche d'Information

Les Systèmes de Recherche d'Information (SRI) ont comme objectif la manipulation d'un ensemble de documents qui constituent un **Corpus Documentaire**. Le Corpus traite un thème particulier ou un ensemble de thèmes qui intéressent un ensemble d'usagers [CDB86] [Cro86] [Ker84]. La Recherche d'Information ("Information Retrieval") est la confrontation entre les besoins des usagers et ce Corpus, afin d'en extraire l'information qui les intéresse [SM 83] [Def86]. Ces besoins sont exprimés par des requêtes, et le Système repère les documents susceptibles de les satisfaire. Le rôle du SRI est donc d'établir une fonction de correspondance entre les éléments du Corpus et les requêtes, et de fournir les mécanismes d'évaluation de cette fonction de correspondance.

Pour réaliser la fonction de correspondance il est indispensable que le SRI connaisse le contenu sémantique du document, pour pouvoir ensuite examiner s'il correspond au contenu de la requête. La nature de cette vérification peut être très simple, comme par exemple l'existence d'une chaîne de caractères, ou plus

compliquée, comme l'équivalence sémantique entre le contenu de la requête (critère de sélection) et le contenu du document [Rij79] [Def86]. L'étape d'acquisition des connaissances sur le contenu d'un document, préalable à l'**étape d'interrogation**, est connue comme l'**Analyse du Document ou Indexation**. Les SRI sont donc composés de deux fonctions principales, la fonction d'Indexation et la fonction d'Interrogation [SM 83] [Ker84].

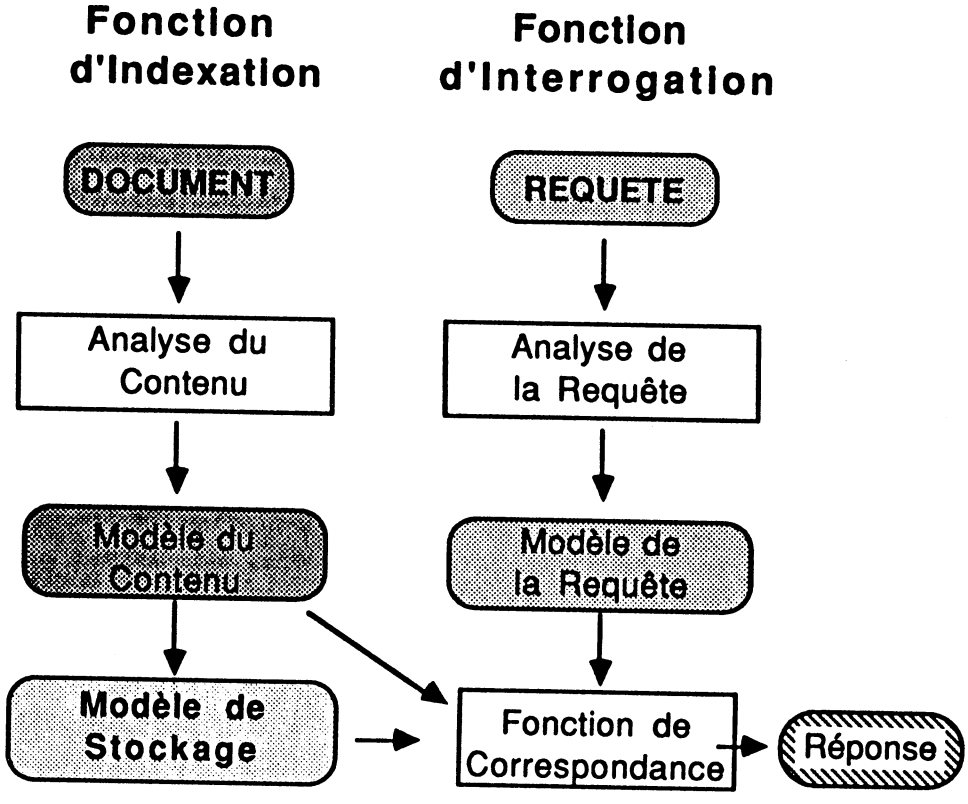


figure 1.3

La fonction d'Interrogation établit la correspondance entre la requête et le modèle sémantique des documents et sélectionne le sous-ensemble d'entre eux qui semblent potentiellement pertinents. Ceci implique des opérations d'accès au modèle de contenu des documents, stocké lors de l'étape d'Indexation. L'étape suivante est celle de l'évaluation des résultats, en

fonction de chaque requête, pour déterminer les références aux documents qui sont réellement pertinents. Le modèle sémantique représentant le contenu des documents doit donc être défini à l'avance et il est relativement figé [Rij79] [Spa72] [Def86].

Il existe donc une très forte interaction entre les deux constituants d'un SRI : la fonction d'Interrogation est aussi performante que le permet l'expressivité du langage d'interrogation et la précision des connaissances acquises sur le contenu des documents lors de l'étape d'Indexation.

Un des problèmes les plus importants est donc la définition du modèle utilisé pour représenter cette connaissance sur les documents. L'accès par le contenu aux documents sera fait à travers une relation associant ce modèle de contenu et des références aux documents correspondants. Cette relation est appelée **Relation d'Indexation.** Les problèmes liés à son stockage et aux méthodes d'accès associées sont décisifs pour les performances de l'Interrogation.

Dans les SRI, le type de recherche proposé est plus large que celui des environnements classiques des SGBD : plus que des résultats de type booléen, vrai (le terme recherché est présent) ou faux (le terme est absent), les SRI s'intéressent aux stratégies de recherche dans lesquelles les documents retrouvés peuvent être plus ou moins pertinents pour la requête [Rij79].

Tous ces modèles sont fondés sur la comparaison (*correspondance*) entre les requêtes et les documents stockés. Plusieurs approches ont été étudiées [Rij79] [SM 83] [Spa72] [CDB86] : modèle booléen, vectoriel, probabiliste, linguistique, etc. Les fonctions de correspondance sont donc plus sophistiquées que de simples opérations booléennes sur des listes de références, comme par exemple des fonctions de similarité entre **clusters** de documents ou entre vecteurs numériques dans un espace représentant l'ensemble de termes connus par le SRI. Cet aspect fondamental des SRI fait l'objet du chapitre 2.

ARCHITECTURE GENERALE D'UN SRI

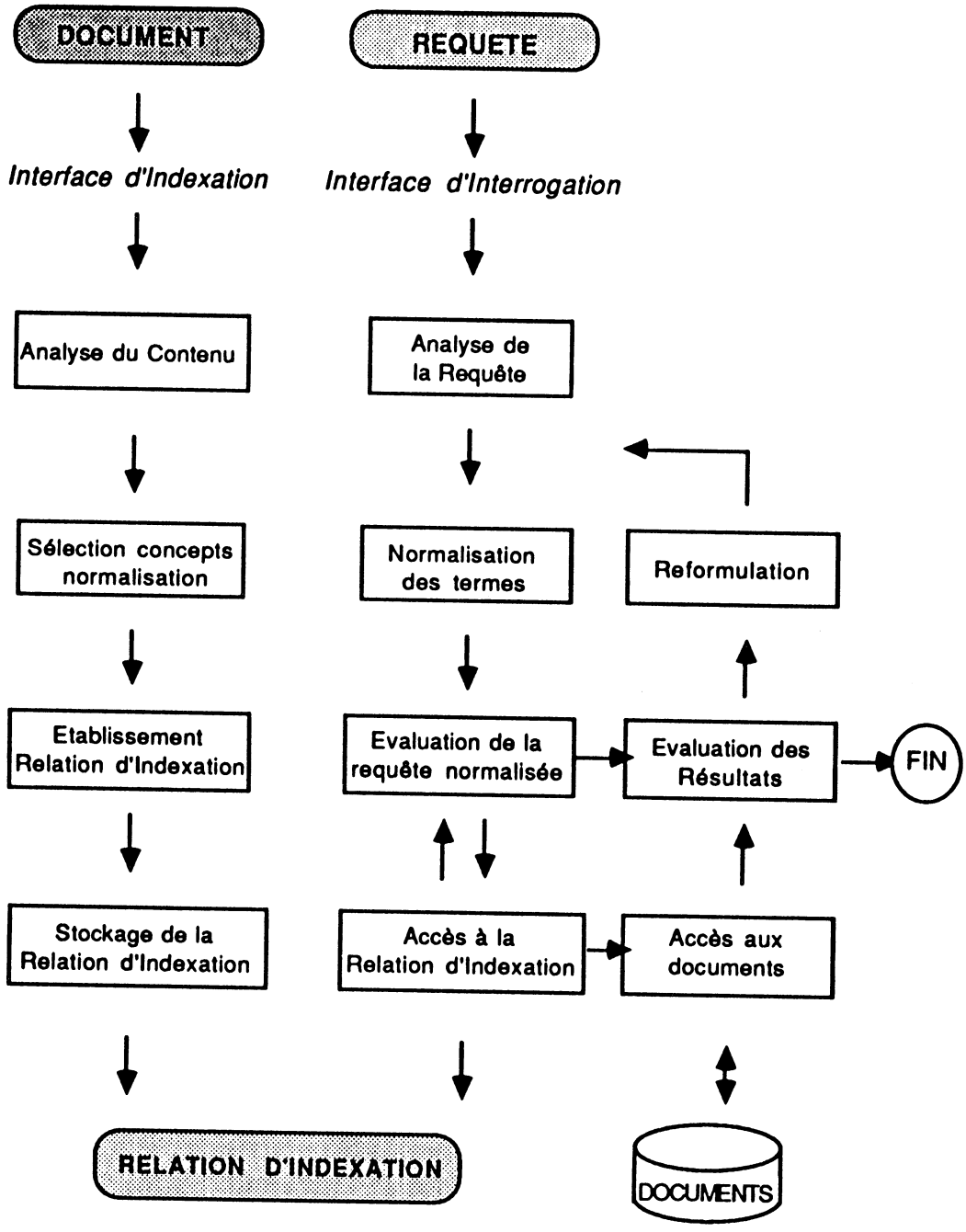


figure 1.4

Dans les SRI une prémisse de base est que l'usager ne connait pas l'information exacte (les documents) qu'il cherche. Il exprime donc, par une requête, une information partielle sur un thème et le SRI recherche les documents les plus pertinents. La requête est analysée et normalisée en fonction des connaissances du système sur le domaine, et elle est évaluée sur l'ensemble du corpus.

Au moment d'évaluer les résultats, la réponse est jugée satisfaisante si les documents retrouvés sont considérés comme pertinents ; sinon il faut aider l'usager à reformuler sa requête pour lui permettre d'accéder aux documents pertinents. Cette approche est aussi valable dans le contexte d'un système bureautique, dont les usagers ne sont pas toujours des experts quant à la connaissance du contenu des corpus gérés par le système.

Il y a de nombreuses approches pour résoudre le problème de la construction des Relations d'Indexation. D'une manière générale, le processus d'Indexation est constitué des étapes suivantes [SM 83] [Ker84] [Pal89] :

- Une étape d'analyse linguistique, qui permet de reconnaître et d'extraire les concepts jugés importants dans le document.

- Une étape de normalisation de ces concepts, et de sélection de ceux qui sont jugés représentatifs du contenu du document.

- Une étape de stockage des concepts retenus, ainsi que des références correspondantes. Ces concepts sont appelés **Termes d'Indexation** et sont ceux qui permettront la sélection ultérieure des documents. L'association de termes d'indexation et des références textuelles correspondantes constitue la Relation d'Indexation.

L'analyse, sélection et normalisation des concepts sont dépendantes des applications. Au niveau du SIB nous sommes

concernés plus particulièrement pour la troisième étape de ce processus : les problèmes de stockage et d'accès à la Relation d'Indexation. Nous nous plaçons donc au niveau le plus bas de l'architecture générale d'un SRI.

Au moment d'intégrer ces fonctionnalités dans un Serveur Bureautique il faut donc tenir compte de deux aspects : l'Indexation, où nous avons étudié les moyens efficaces pour stocker les Relations d'Indexation dans un espace raisonnable, en tenant compte du dynamisme inhérent à l'information du bureau et la diversité des thèmes couverts, tout en assurant des performances d'accès satisfaisantes. Le deuxième aspect est l'Interrogation, en fournissant à l'usager un langage simple pour exprimer ses requêtes et interpréter les résultats. Il est important de souligner que dans le cas d'un **Serveur Bureautique**, "l'usager" est, bien entendu, une **Application Bureautique** qui à son tour fera l'interface avec l'utilisateur final.

Nous allons à présent spécifier les fonctionnalités d'accès aux documents bureautiques, telles qu'elles ont été définies dans le SIB.

1.4. Recherche par le Contenu des Documents dans le SIB

Pour définir les fonctions d'accès aux documents dans le SIB, il faut d'abord tenir compte de la nature des documents et des applications qui y accèdent. Les documents bureautiques ne sont pas en général trop volumineux, il peuvent être structurés, et leur contenu peut varier fortement, même dans le cadre d'une application donnée.

Dans le projet DOEOIS, la méthode d'analyse pour établir les besoins des applications d'un environnement de bureau à été l'étude de cas réels [ESP86a] : Un concessionnaire automobile, un

service de scolarité universitaire, la gestion d'une loterie, un service financier universitaire et une Chambre de Commerce et d'Industrie. Dans le cas du Concessionnaire automobile (choisi comme référence dans la spécification des fonctionnalités de l'OIS), par exemple, nous pouvons trouver des documents aussi différents qu'un contrat de vente, la description technique des véhicules, le courrier avec les clients ou les manuels de références techniques.

Les fonctions d'accès par le contenu à ces documents doivent être uniformes, suffisamment générales et souples, pour permettre aux applications qui le souhaitent, d'effectuer des accès simples, comme par exemple une recherche de chaînes de caractères, ou des accès plus sophistiqués, en tenant compte de la sémantique du contenu des documents, comme cela est fait dans les SRI. Par exemple, dans le cadre du Concessionnaire Automobile, les besoins de prise en compte de la sémantique exprimée dans la documentation technique des véhicules peuvent être différents à ceux de la correspondance interne de l'entreprise.

Nous avons donc défini deux fonctions d'accès au contenu des documents pour le SIB, en tenant compte de ces deux types d'applications :

- Une première fonction qui permet de retrouver les documents dont le contenu correspond à une condition spécifiée par une expression de chaînes de caractères, comme "bases de données ou serveurs d'information". Dans ce cas là, on considère le *contenu textuel* du document, et l'accès s'effectue par des techniques de "pattern matching" entre chaînes de caractères. Ici les documents sélectionnés sont ceux qui contiennent soit "bases de données" ou "serveurs d'information" comme sous-chaînes. Cette fonction a été appelée **Recherche Textuelle**.

- La deuxième fonction tient compte du contenu sémantique du document, tel que cela est fait dans les SRI. Donc, les techniques d'accès portent sur le modèle sémantique du document. Cette fonction a été appelée **Recherche**

Sémantique. Une requête est alors une expression logique de termes d'indexation, et l'accès aux documents correspondants met en œuvre une relation d'indexation préalablement construite.

Ces deux fonctions ont entre elles la même relation qu'on trouve entre la syntaxe et la sémantique : la Recherche Textuelle fait référence à l'expression particulière d'un concept, tandis que la Recherche Sémantique fait référence au concept lui-même. Avec la Recherche Sémantique sont donc résolus les problèmes causés par les expressions textuelles non contrôlées, qui provoquent une faible précision dans les réponses (variations grammaticales, paraphrases, etc).

Pour exprimer les deux types de recherche ainsi définis, nous avons introduit dans le langage de manipulation de données les opérateurs correspondants. Ces opérateurs se situent au niveau de la clause **WHERE** d'un énoncé **SELECT**, dans un langage de type SQL. Une opération de recherche textuelle est exprimé de la forme "**SELECT** ... **FROM** ... **WHERE** ... <document> **TALKS ABOUT** <expression textuelle>". Une opération de recherche sémantique est exprimé de la forme "**SELECT** ... **FROM** ... **WHERE** ... <document> **IS RELEVANT TO** <expression de termes d'indexation>".

De cette manière nous offrons aux applications bureautiques deux méthodes complémentaires pour accéder au contenu des documents. La recherche textuelle ou la recherche sémantique est un choix qui dépend des besoins de précision et de qualité des réponses qui sont requises dans chaque Procédure de Bureau. Dans le cas de la Recherche Sémantique, il faut aussi des outils complémentaires au niveau de l'application pour modéliser la connaissance nécessaire à une bonne normalisation des termes et des expressions d'interrogation (nécessité d'une procédure d'indexation associée).

Aussi bien pour la **Recherche Textuelle** que pour la **Recherche Sémantique** par le contenu des documents du SIB, il est nécessaire de développer des techniques de stockage et d'accès adaptées, en tenant compte de l'environnement propre au SIB. Il faut donc tenir compte des particularités et des différences avec

les systèmes que nous prenons comme référence (les SGBD et les SRI).

Faisons un court rappel des principales caractéristiques de ces systèmes [Chr85], pour montrer les aspects qu'on peut reprendre et ceux qu'il faut développer.

Les SGBD traditionnels gèrent l'information classique, qui est constituée dans la majeure partie des cas de données formatées. Les insertions, les suppressions et les mises à jour sont fréquentes.

Les SRI gèrent de grands volumes de documents, ayant une représentation condensée, comme par exemple des résumés associés à des listes de mots-clés appartenant à des vocabulaires contrôlés. Dans un SRI les insertions de documents sont réalisées par 'lots', par un Administrateur du Système et les suppressions sont très rares, pour ne pas dire inexistantes. Les mises à jour de documents ne se présentent jamais.

Dans le SIB se trouvent, de la manière la plus générale, des données classiques, des documents et de l'information multimédia. Les documents sont structurés et le vocabulaire est ouvert. Les insertions et les mises à jour de données sont réalisées dynamiquement et elles doivent être effectives rapidement. Dans les SIB on peut avoir la suppression systématique de certains types de données alors que pour d'autres, on garde des versions ou des historiques.

Un autre aspect important qui différencie ces trois environnements est l'interrogation. Dans les SGBD l'information est extraite selon des critères de correspondance exacte : tout ce qui répond (sans ambiguïté) à une requête est donné comme réponse. Dans les SRI, étant donné les types d'utilisateurs et les services d'interrogation offerts (voir la partie 2 de ce chapitre), il est toléré d'avoir des réponses partiellement exactes, soit parce que toute l'information pertinente à une requête n'est pas retrouvée (réponses avec un *rappel* inférieur à 100%, phénomène également connu sous le nom de *silence*), soit parce que tout ce

qui est récupéré ne correspond pas complètement à la requête. Ceci est connu comme le *taux de précision*, qui peut donc être également inférieur à 100%.

Dans les SIB nous sommes face à un tout autre type d'applications, où il peut être indispensable d'avoir un taux de rappel et de précision proche de 100%, comme par exemple les applications de ventes, et où il peut être également utile de disposer de la flexibilité d'interrogation offerte par les SRI, comme c'est le cas dans les applications documentaires.

Nous devons donc offrir les méthodes d'accès aux documents qui correspondent aux fonctions définies précédemment, en tenant compte des caractéristiques mentionnées. Nous avons suivi une approche commune, visant à privilégier les aspects de temps d'accès et d'espace nécessaire, ainsi que la dynamicité au moment de l'acquisition des documents.

La technique des Signatures [Chr84] [CF 84] semble la plus appropriée pour les documents des SIB, par sa simplicité, sa généralité et l'économie de place. Par exemple, contrairement aux techniques fondées sur la construction de fichiers inverses, en utilisant les Signatures il n'est pas nécessaire de restreindre le vocabulaire traité. Il est possible de traiter le contenu complet des documents avec des performances acceptables, et elles permettent de construire des index avec un surcoût de stockage réduit. En particulier pour la **Recherche Sémantique**, le stockage de la Relation d'Indexation est un problème critique, car dans les SRI classiques elle est en général très volumineuse [RZ84] [Ker84].

Les performances en temps d'accès étant un critère très important dans le SIB, les méthodes de Signatures peuvent ne pas être suffisamment efficaces par elles mêmes. Nous avons alors étudié l'alternative d'utilisation de matériel orienté filtrage pour améliorer encore l'efficacité de la recherche.

Nous avons réalisé un prototype pour chacune de ces alternatives. L'alternative du filtrage logiciel a été implanté en son

intégralité, et intégrée au Serveur OIS. L'alternative du filtrage materielle a été implantée en une grand partie, sur le filtre SCHUSS. Néanmoins, les mesures sur ce prototype n'ont pas pu être complétées, pour des raisons techniques.

1.5. Conclusions

Nous avons présenté dans ce chapitre l'environnement d'un Serveur d'Information Bureautique générique, ou SIB, les caractéristiques de l'information qui y circulent, et nous avons caractérisé les applications de bureau. Nous avons présenté l'architecture du Serveur qui nous sert d'exemple, l'OIS. Nous avons comparé le SIB à deux environnements qui lui sont proches, les SGBD et les SRI. De chacun d'entre eux nous pouvons reprendre des fonctionnalités et les adapter aux besoins particuliers du **bureau**. Nous nous intéressons notamment à la gestion des documents dans le SIB, et en particulier à l'accès à leur contenu. Nous définissons ainsi deux fonctions, la **Recherche Textuelle** et la **Recherche Sémantique**, qui sont complémentaires pour donner aux applications bureautiques les méthodes d'accès et de stockage des documents et des Relations d'Indexation associées.

Nous nous sommes intéressés à la méthode des signatures comme moyen de stockage des documents et des relations d'indexation, cette méthode étant susceptible de résoudre les problèmes de place classiquement rencontrés, et de résoudre également des problèmes liés à la gestion dynamique des documents dans le SIB.

En ce qui concerne l'implantation des techniques associées à la recherche textuelle et à la recherche sémantique, nous étudions particulièrement les techniques de filtrage sur les textes pleins et sur les signatures des informations textuelles. Pour pousser plus loin l'analyse des performances liées à cette approche, nous étudions des approches logicielle et matérielle de ces techniques de filtrage. Les opérations ainsi définies ont été intégrées au Serveur OIS.

2. LA MODELISATION DES DOCUMENTS DANS LES SRI

2.1. Introduction

Dans le chapitre précédent nous avons décrit l'environnement du bureau, les fonctionnalités du SIB et avec plus de détail le Gestionnaire de Documents. Une fonctionnalité très importante dans les SIB est l'accès par le contenu aux documents, qui constitue l'objet principal de ce travail. Nous avons défini deux fonctions d'accès, la Recherche Textuelle, concernant des opérations de mise en correspondance (*"pattern matching"*) entre chaînes de caractères, et la Recherche Sémantique, qui tient compte du contenu sémantique des documents.

Cette dernière fonction, la Recherche Sémantique, fait appel à des concepts développés dans le contexte des Systèmes de Recherche d'Information, qui abordent le problème de la représentation et de la modélisation du contenu des Corpus Documentaires. A la base des SRI nous trouvons la définition d'une *fonction de correspondance requête-documents*, avec ses propres stratégies de stockage et d'accès, selon le degré de finesse désiré dans le modèle des documents. Tous ces éléments sont classiquement regroupés sous le terme de *Modèle de Recherche d'Information.*

Dans ce chapitre nous faisons un rapide état de l'art sur les différents modèles. Nous consacrons une partie de ce chapitre à une présentation du système qui sert de point de départ à notre

démarche, IOTA [Def86] [Ker84] [CDB86], un Système Intelligent de Recherche d'Informations développé au Laboratoire de Génie Informatique de l'IMAG. Nous en présentons les points fondamentaux qui seront repris dans la définition des fonctions d'accès par le contenu dans le SIB, et nous décrivons les caractéristiques de l'environnement pour lequel IOTA à été conçu.

2.2. Les Modèles de Recherche d'Information

Dans les SRI la modélisation du contenu du corpus documentaire a deux composants. D'une part, la description du domaine couvert par le corpus, d'autre part un modèle du contenu des documents par rapport à ce domaine. Dans la plupart des SRI existants, l'outil utilisé pour la description de l'ensemble des concepts d'un domaine est le thésaurus [Chi83] [Cro83] [Ker84]. Il est constitué classiquement d'un ensemble de concepts et des relations sémantiques ou contextuelles (synonymie, généricité, proximité contextuelle...) entre ces concepts. Deux approches sont possibles pour sa construction :

- Une approche manuelle, qui implique qu'un spécialiste dans le domaine définisse le contenu du thésaurus. Dans ce cas, les connaissances non explicites des documents peuvent être incluses, mais par contre, il y a un très grand degré de subjectivité. Les thésaurus construits par deux spécialistes sur le même corpus seront sûrement différents, ce qui entraîne des difficultés lors de l'indexation et de l'interrogation. En fait, le plus grand problème est la cohérence et la normalisation des concepts et relations retenus.

- Une approche automatique, qui implique que seuls les concepts explicités dans le corpus pourront être traités par le SRI. En général, les relations sémantiques entre termes

sont fondées sur des critères statistiques [SM 83] [Ker84] [Bru82] [Sal71] [Spa72].

Durant ces dernières années, le domaine de la Recherche d'Information s'est ouvert à diverses techniques propres à l'Intelligence Artificielle (IA). Les deux principaux composants d'un SRI sont concernés. D'une part, pour l'analyse et la représentation du contenu du corpus documentaire (analyse de la langue naturelle, réseaux sémantiques, règles de production, etc), d'autre part pour la phase d'interrogation (analyse de la langue naturelle, systèmes experts d'interrogation, fonctions de correspondance fondées sur des mécanismes de déduction, etc).

La coopération entre l'IA et les SRI a permis d'introduire dans les modèles de représentation de documents un élément jusqu'alors très peu considéré : la sémantique. Les SRI ont pu s'enrichir avec des outils linguistiques et des opérateurs de recherche plus puissants (règles de recherche, équivalence sémantique, utilisation de logiques d'évaluation non booléennes), permettant de considérer les données acquises sur le contenu d'une base documentaire comme une *base de connaissances sur le domaine*. Ainsi, les modèles peuvent être augmentés grâce à des possibilités d'apprentissage en fonction de l'interaction avec les utilisateurs pendant la phase d'interrogation.

En ce qui concerne la modélisation du corpus documentaire, l'utilisation d'outils linguistiques permet une identification automatique partielle du contenu des documents. Les techniques actuelles ne sont cependant pas suffisamment développées pour résoudre tous les problèmes sémantiques des langues naturelles, comme les paraphrases, les ambiguïtés contextuelles, les références pronominales dans toute leur généralité. Elles nous permettent cependant de réaliser avec une bonne fiabilité des opérations telles que l'analyse morpho-syntaxique, qui permet d'identifier les éléments grammaticaux dans une phrase, la reconnaissance de syntagmes particuliers, ou l'identification des opérateurs logiques implicites ou explicites dans une requête.

Dans l'analyse d'un document il est "classiquement" admis qu'une analyse morpho-syntaxique *de surface* est suffisante, pour

identifier les portions de texte jugées intéressantes quant au contenu sémantique. D'après les études sur ce sujet [BP 86] [Pal86], il semble que, dans la plupart des langues, les éléments les plus porteurs d'information en vue d'un processus postérieur d'interrogation sont les syntagmes nominaux. Les syntagmes verbaux sont donc pour l'instant ignorés dans un processus d'indexation [Ker84].

Dans les approches où le contenu des documents est considéré comme la *connaissance* sur le domaine du corpus, les modèles de représentation ont tendance à admettre un certain degré de souplesse, pour tenir compte des aspects dynamiques du corpus, et une certaine liberté pour la définition de la structure des documents, souvent bien différente selon l'application.

Pour définir un modèle de représentation de ces connaissances, et les méthodes d'indexation associées on peut distinguer plusieurs approches :

- Le modèle booléen
- Le modèle vectoriel,
- Les modèles fondés sur la logique,
- Les modèles à base de règles de production,
- Les représentations par des réseaux sémantiques,

Nous allons maintenant décrire des SRI représentatifs de ces tendances, à la fois du point de vue de la structure de leurs relations d'indexation, et des méthodes de stockage utilisés ainsi que les méthodes d'accès qui leur sont associées. Nous ne prétendons pas être exhaustifs, mais seulement donner une vision générale du domaine.

2.2.1. Le Modèle Booléen - Le Système STAIRS

Les systèmes fondés sur cette approche ont été les plus commercialisés dans le cadre de Systèmes de Recherche documentaire. Les plus connus d'entre eux sont DIALOG [BA 79], STAIRS [IBM79], MEDLARS [NLM79]. Nous détaillons à titre d'exemple le système STAIRS.

Dans ce type de système de recherche sur des textes complets, un répertoire complet de tous les mots contenus dans les textes doit être maintenu, ainsi que toutes les références à tous les documents contenant ce mot, voire à toutes les phrases contenant ce mot. Ces informations sont stockées dans une structure de fichier inverse.

Pour une requête il est nécessaire d'établir la liste des références à tous les documents concernés. La résolution d'une requête revient donc à combiner par les opérations ensemblistes associées aux opérateurs booléens (union pour le *OU*, intersection pour le *ET*, différence pour le *SAUF*), les ensembles de références correspondant à chaque terme de la requête. Dans ce cas là, une stratégie de *correspondance exacte* doit est appliquée pour chaque terme demandé.

Le grand avantage de ces méthodes est l'efficacité, vu que les requêtes peuvent être résolues avant d'accéder aux contenus des documents proprement dits. Seul les documents sélectionnés sont accédés.

L'utilisation d'un vocabulaire libre pour décrire les documents, pose un problème dans la mesure où deux documents contenant des termes voisins (des synonymes, par exemple), ne seront retrouvés que si ces termes sont explicités dans la requête. L'utilisation d'un vocabulaire contrôlé minimise le problème, mais n'est pas suffisante, la cohérence n'étant pas garantie dans la mesure où les termes peuvent être considérés comme différents par des personnes différentes. Cette difficulté, provenant de la subjectivité de l'indexation, met en évidence la

nécessité des critères automatiques de sélection de termes, aussi bien pour l'indexation que pour l'interrogation.

Cette solution fondée sur des fichiers inverses est donc utilisable pour des systèmes ayant un vocabulaire de recherche assez statique, et limité à un nombre donné de termes. C'est dans ces conditions que la taille et l'organisation de l'index sont gérables à un coût raisonnable. Malheureusement, ces conditions sont rarement remplies dans les applications réelles : les mises à jour sont fréquentes, les vocabulaires sont assez volumineux, et il est nécessaire de stocker des très grands index, coûteux à maintenir.

STAIRS consiste en deux parties, l'une consacrée au stockage et à la maintenance de la base documentaire, et l'autre à la consultation interactive. Il utilise un dictionnaire de mots (l'index inverse), un index sur les textes et les fichiers textes proprement dits.

Les fichiers de texte contiennent les documents, formatés de la manière dont ils seront présentés aux utilisateurs. L'index des textes contient des pointeurs vers les composants des documents, ainsi que de l'information additionnelle, comme par exemple, les droits d'accès. Le dictionnaire de mots contient une entrée pour chaque mot connu, et la liste de toutes ses occurrences. Les mots synonymes sont reconnus à l'aide d'un dictionnaire auxiliaire. Le dictionnaire de mots est un fichier inverse organisé en deux niveaux : le premier donne l'accès aux deux premiers caractères du mot, le deuxième permet l'accès au mot complet, ses synonymes et ses références. Les références aux occurrences de mots consistent en un triplet, (code de paragraphe, numéro de phrase, numéro de mot).

STAIRS permet la recherche sur des textes complets ou sur des résumés. Les termes d'une requête peuvent être spécifiés en mode *RECHERCHE*, à l'aide des opérateurs logiques, des opérateurs de contiguïté entre mots et d'inclusion dans une phrase ou dans un paragraphe, d'un opérateur qui permet de considérer deux termes comme synonymes, et d'un joker à la fin d'un terme de recherche (par exemple 'system$'), qui permet de spécifier

seulement les premiers caractères des termes, avec des suffixes variables.

Avec l'opérateur *RANK* les résultats peuvent être ordonnés, en affectant des poids aux termes associés aux documents. L'ordre est donné par la somme des poids individuels des termes qui correspondent à ceux contenus dans la requête.

Le poids d'un terme par rapport à un document est décidé par l'utilisateur. Il peut combiner, dans son algorithme de calcul, par exemple, les fréquences du terme dans le document, du terme dans l'ensemble des documents retrouvés et le nombre de documents retrouvés dans lesquels le terme apparaît.

D'autres systèmes proposent des organisations et fonctionnalités similaires à celles de STAIRS, dans des environnements différents. Un exemple est montré dans [MMN86], où STAIRS est repris dans un SRI réparti, construit sur un réseau d'ordinateurs individuels type PC.

Une autre méthode d'implantation, ayant à la base une inversion des mots contenus dans les textes, a été étudié par Y. Choueka, A. Fraenkel et S. Klein, dans le cadre de la représentation des **concordances** du texte (les mots et les listes des références qui leurs sont associées) par des chaînes de bits ("bit maps"), stockées en utilisant des méthodes de compression [CFK88]. Ces travaux seront détaillés dans le chapitre suivant, dans l'étude sur les méthodes de compression.

Les SRI fondés sur le modèle booléen présentent le gros avantage d'une implantation relativement simple et donc très performante sur le plan quantitatif. Le gros inconvénient de ce modèle tient cependant à la fonction de correspondance qui est, dans ce cas, exacte. Malgré les assouplissements procurés par différents opérateurs (proximité, synonymie, joker, ...), les performances qualitatives de ces systèmes demeurent peu satisfaisantes. Par ailleurs, l'*overhead* représenté par les fichiers inverses est très important ; il peut représenter plusieurs fois la taille des documents eux-mêmes.

2.2.2. Le Modèle Vectoriel - Le Système SMART

Ce modèle a été proposé par G. Salton et son équipe de Cornell University [Sal71] [Sal86]. Il introduit pour la première fois dans les SRI des critères de correspondance non stricte entre documents et requêtes, fondée sur une représentation vectorielle des documents et une évaluation statistique de l'importance des termes dans la représentation du contenu sémantique des documents.

Dans SMART le domaine correspondant à un corpus est représenté par l'ensemble T des termes d'indexation. Chaque document D du corpus est représenté donc par un *vecteur document* de dimension T, où T est le nombre de termes acceptés par le processus d'indexation :

$$D = [\ w_1, w_2, \ldots, w_T\ ]$$

l'élément w_i représentant l'importance (on dit aussi le poids) du terme t_i dans le document. A partir de cette relation terme-document, est établie la Relation définissant l'espace documentaire comme la réunion des vecteurs documents. Cette relation est donc une matrice associative où chaque élément (i, j) est le degré d'association entre le document i et le terme j.

Une requête peut être représentée selon le même modèle :

$$R = [w_1, w_2, \ldots, w_T\ ]$$

Les poids assignés peuvent être de nature binaire (0 ou 1 selon que le terme est absent ou présent dans la requête) ou avoir des valeurs continues entre 0 et 1 si l'on veut différencier l'importance de certains termes par rapport à d'autres dans la requête.

SMART utilise des critères linguistiques et statistiques pour l'indexation. Les documents sont analysés en deux étapes : une analyse morpho-syntaxique d'abord, et une analyse statistique

suivie d'une autre étape syntaxique qui permet le regroupement de certains termes en 'pseudo-syntagmes'. On passe ensuite à l'analyse sémantique, où l'on utilise un thésaurus manuel et des procédures sémantiques propres au domaine.

Ces poids w_{ij} sont calculés pour chaque document, diverses approches sont possibles pour cela et font l'objet de méthodes statistiques particulières. Le corpus peut être vu comme un ensemble de vecteurs dans un espace à N dimensions. La *similarité* entre documents et requêtes (ou entre documents) est alors vue comme une comparaison de vecteurs dans cet espace.

Une méthode possible pour comparer ces vecteurs est d'en mesurer le cosinus de l'angle qu'ils définissent, selon la formule ci-dessous :

$$S(D_i, R) = \frac{\sum_{i=1}^{N} w_{ik} \times w_{rk}}{\sqrt{\sum_{i=1}^{N} w_{ik}^2 \times \sum_{i=1}^{N} w_{rk}^2}}$$

où N est le nombre de documents considérés, et w_{rk} est le poids du terme t_k dans la requête.

La qualité d'un terme *t* comme descripteur dans le domaine est estimée à partir des mesures contenues dans la Relation d'Indexation, et par des critères empiriques. Les termes ayant une très faible valeur discriminante sont considérés comme trop génériques du domaine; ils diminuent la précision par leur mauvaise discrimination entre documents. Inversement, les termes ayant une valeur discriminante très forte sont ceux qui sont très spécifiques dans le domaine, et sont importants pour obtenir une bonne précision lors de l'interrogation. Ces termes sont regroupés en syntagmes de descripteurs.

Dans [Spa72] il est montré qu'*un bon terme doit être très fréquent dans un document en particulier, mais il doit être peu*

fréquent dans le reste du corpus documentaire. Ceci a permis de définir une formule de pondération de termes connue comme *"tf * idf"*, la multiplication entre la fréquence du terme *dans* le document et la *fréquence inverse du document*, représentant une fonction inverse du nombre de documents indexés par le terme [SM 83] [Sal86].

De cette manière, la Relation d'Indexation fournit les pondérations nécessaires pour avoir les deux aspects fondamentaux pour une bonne recherche documentaire, le rappel et la précision. Cette notion de pondération et la classification des termes en génériques et spécifiques ont été retenues dans la méthode d'indexation documentaire de IOTA [Ker84].

SMART offre une méthode de regroupement des documents ("clustering"), ainsi qu'une méthode de reformulation de requêtes en fonction des résultats obtenus ("Relevance Feedback"). Une description détaillée de ces caractéristiques est présentée dans [SM 83], et elles sont analysées dans [Def86].

La représentation du contenu sémantique du corpus est ainsi faite à travers l'ensemble de vecteurs de documents. SMART a apporté aux SRI classiques les notions d'indexation automatique et de calcul de la fonction de similarité. Ces processus font néanmoins peu d'appel à la sémantique, notamment au moment de l'établissement de la fonction de correspondance et la reformulation de requêtes.

Pour des corpus dynamiques, la représentation vectorielle est très difficile à maintenir en raison, notamment, de la forme statique comme les termes d'indexation sont considérés. En fait, l'insertion d'un nouveau terme implique de recalculer tous les vecteurs et tous les poids. Par ailleurs, la structuration logique des documents n'est pas prise en compte.

2.2.3. Le Modèle Probabiliste

Ce modèle de SRI a été proposé par K.J. van Rijsbergen [Rij79]. Il est fondé sur les notions de classification automatique des documents ("clustering") et des concepts, pour la représentation du contenu du corpus documentaire. La classification d'un document dans un *cluster* est décidée par le calcul d'un coefficient de similarité, dont l'algorithme dépend des besoins spécifiques du SRI en question.

En fait, le modèle vectoriel proposé par Salton (cf. ci-dessus), utilise un cas particulier de ces coefficients, le cosinus entre deux vecteurs documents donnés. La Recherche d'Information dans ce contexte fait l'hypothèse que deux documents fortement associés (donc dans le même cluster) ont tendance à être pertinents relativement aux mêmes requêtes. Diverses méthodes de classification ont été étudiées, qui conduisent à des classifications hiérarchiques, ou non hiérarchiques.

Le corpus étant représenté par des clusters, les techniques de stockage et de recherche sont fondées sur le *représentant du cluster*, qui caractérise ses éléments. Donc, une requête va être évaluée par rapport à chaque cluster, en fonction de la pertinence du représentant, supposé représentatif du contenu des documents qui lui sont associés. La sélection de ce représentant "idéal" du cluster reste cependant un problème ouvert ; elle est pour l'instant déterminée de manière empirique.

Le processus d'interrogation commence par évaluer la correspondance cluster-requête à un niveau donné (par exemple, la racine, si le cluster est organisé de manière hiérarchique). Puis il se poursuit en appliquant des règles de décision sur le 'balayage des clusters', et d'arrêt, par comparaison avec les descripteurs des documents. Les règles de décision peuvent inclure des retours arrières, et des stratégies descendantes ou ascendantes pour l'évaluation dans les clusters.

L'interrogation et la recherche sont accompagnées également de stratégies de reformulation interactive de requêtes et d'enrichissement pour améliorer les performances.

Le modèle probabiliste est fondé sur l'hypothèse que la distribution des termes d'indexation sur une collection de documents, ou une partie d'entre eux, est un indicateur de la pertinence d'un document donné [Rij79]. La pertinence d'un document est déterminée par l'évaluation d'une fonction de correspondance fondée sur cette distribution des termes d'indexation.

La fonction de correspondance permet donc d'établir si un ensemble de documents répond, et à quel degré, à une requête, d'après la pertinence des termes d'indexation associés. Il est supposé que la pertinence d'un document par rapport à une requête est indépendante des autres documents de la collection. Donc, l'intérêt est d'estimer la *probabilité de pertinence d'un document par rapport à une requête,* P_Q*(pertinence/document)*, et la probabilité de pertinence d'un terme d'indexation dans un document. Dans [Rij79] sont développés les méthodes d'estimation de ces paramètres, qui permettent de déduire le *poids* avec lequel les résultats répondent à la requête.

Cette théorie a été reprise dans beaucoup de SRI, dans le calcul de la Relation d'Indexation. La représentation du contenu en clusters pose cependant le problème de l'établissement de critères automatiques de classification. Comme indiqué plus haut, des nombreux algorithmes de classification ont été proposés. (cf. par exemple l'algorithme de classification hiérarchique de clusters est proposé dans [EW 86]).

Si l'approche formelle du modèle probabiliste est très intéressante, la mise en œuvre de ce modèle sur des applications réelles pose de très gros problèmes liés à l'empirisme des évaluations des probabilités (la condition d'indépendance est très approchée), et aux algorithmes de classification très coûteux.

2.2.4. Les Modèles fondés sur des Règles de Production

Le système RUBRIC, proposé par [Ton85] [Ton87], utilise un modèle de règles de production pour établir la fonction de correspondance entre la requête et la base documentaire. La connaissance du système est représentée par des règles qui définissent les *concepts* du système. La relation d'indexation est représentée, comme pour le modèle booléen, par un fichier inverse des mots présents dans les documents, avec l'information contextuelle permettant la recherche avec des opérateurs positionnels (même phrase, même paragraphe, etc). Du point de vue de la représentation du contenu sémantique des documents, ce système est donc très classique. La différence par rapport aux autres modèles se trouve au niveau de l'interrogation.

La requête est exprimée par des règles, qui définissent un filtre de recherche sur les documents. Les règles permettent de définir une hiérarchie entre les concepts, donnant une stratégie d'évaluation des réponses et par conséquent une classification en termes d'une pertinence estimée. Les règles sont définies à l'aide d'une syntaxe étendue des règles de production :

IF <condition1>
THEN <conclusion1>
BUT IF <condition2>
THEN <conclusion2>

Exemple :

La règle **EVIDENCE** permet d'associer des expressions du texte avec des concepts. Par exemple,

(**EVIDENCE** ***france***

((*OR "France" "Paris" "Mitterand") 0.7)

permet d'associer le concept ***france*** avec un degré de pertinence de 0.7 aux documents contenant les chaînes de caractères "France", "Paris" ou "Mitterand".

La stratégie d'évaluation est exprimée par des règles de la forme :

IF (*evidence (terme, réf)* < O.OO1)
THEN ignorer *(terme, réf)* dans la réponse
BUT IF nombre de réponses > 100
THEN reformuler la requête

D'autres extensions de ce type de formalisme [Def86] sont des règles de la forme :

SI *<condition1>*
ALORS <conclusion1> **AVEC CERTITUDE** *p1*
ET SI *<condition2>*
ALORS <conclusion1> **AVEC CERTITUDE** *p2*

pour définir, par exemple, des heuristiques d'évaluation.

Les performances du système semblent fortement liées à la qualité de la formulation en termes de règles et à la définition des coefficients de recherche, que seul un utilisateur spécialiste du système pourrait bien exploiter. Du point de vue de la représentation même du contenu sémantique des documents cette approche a des caractéristiques similaires à celles du modèle booléen.

2.2.5. Les Modèles Construits sur des Réseaux Sémantiques

Ces modèles de documents sont fondés sur la notion de **concepts** présents dans le corpus. Un concept reconnu par le système a donc un schéma associé. Deux systèmes, assez différents, illustrent cette approche :

Le système proposé dans [BCB86], COREL, est un prototype d'un SRI où les documents sont des articles du code civil. Ces documents sont *indexés conceptuellement* en utilisant des

structures de représentation des connaissances basées sur des *schémas*.

Le domaine du corpus (l'ensemble d'articles légaux) est représenté par un ensemble de *structures de connaissance*, où chacune d'entre elles représente un concept légal. Chaque concept possède plusieurs attributs conceptuels qui le définissent, et qui font partie de la structure du schéma. Le processus d'interrogation est supporté par un mécanisme de réseaux de discrimination, qui aide l'usager dans son parcours des concepts et de relations entre concepts. Ces réseaux permettent une détermination automatique du sous-ensemble de concepts auquel l'usager s'intéresse, sans avoir besoin de construire des requêtes booléennes.

Le réseau de discrimination est composé par des noeuds de plusieurs types. Le premier niveau permet de décrire les principaux thèmes traités (famille, propriétés, etc). D'autres types de noeuds décrivent les relations, d'autres les définitions. Ce réseau est présenté à l'usager sous forme de menus, ce qui lui permet de naviguer dans les structures, le système le guidant dans la spécification des termes recherchés.

Ce système est proposé pour des domaines bien définis, où les concepts sont clairement identifiables. La performance du système dépend fortement de l'organisation de la base, tâche réalisée par un spécialiste dans le domaine. En particulier, le réseau de discrimination doit être construit manuellement.

Un autre SRI utilisant des réseaux sémantiques pour modéliser les documents est proposé par W. B. Croft dans [Cro83] et [Cro86]. Son système, *I³R*, a deux grands composants : la base de connaissances sur le domaine, qui est un réseau d'inférences représenté comme une collection de schémas contenant des règles de conditions de reconnaissance des concepts et de leurs relations, et un système expert qui réalise les opérations de recherche et d'évaluation.

Pour permettre le fonctionnement du système avec des bases de connaissances non pré-établies et incomplètes lors de l'interrogation, les connaissances sur le domaine sont fournies par l'utilisateur, ceci en vue d'affiner l'interprétation des requêtes. Le système est alors un intermédiaire de recherche [Cro86] qui l'assiste dans les tâches d'évaluation et de reformulation.

Comme dans le cas des représentations probabilistes des documents, leur contenu est, à son plus bas niveau, représenté comme un ensemble de mots isolés, l'indexation étant donc une sélection de mots clés. Le système offre deux modes d'accès au contenu, les stratégies probabilistes et le balayage ("browsing").

L'architecture du système est fondée sur la coopération de plusieurs systèmes experts utilisant des mécanismes de *tableaux* ("blackboard") :

- Le système de modélisation des usagers, qui gère des définitions partielles du domaine, et des historiques des sessions.

- Le système de modélisation des requêtes. qui gère l'information actuellement utilisée.

- Le module expert en Indexation, qui analyse initialement les documents et les requêtes.

- Le module exploitant le thésaurus.

- Le Contrôleur de requêtes, qui gère les stratégies d'accès à suivre pour une requête spécifique.

- Le module de balayage, qui permet à l'utilisateur de naviguer dans la base de connaissances pour ensuite pouvoir formuler ses requêtes.

- Un module assistant l'usager quant à l'utilisation du système.

Un concept dans le domaine est représenté par un schéma contenant le nom du concept, une information sur la manière dont il peut être reconnu dans le texte (règles "*if <chaîne> then <concept>*" avec un coefficient de certitude) et ses relations avec d'autres concepts (synonymie, généricité, composition, etc). Après l'interprétation de la requête pour en extraire les concepts présents, le thésaurus fournit une liste de concepts qui seront utilisés pour retrouver les documents, ceci d'après les techniques des modèles probabilistes.

Dans [DeJ86] et [BM 88] on trouve aussi d'autres SRI où la connaissance du système et les descripteurs de documents (Relation d'Indexation) sont représentés à l'aide de réseaux sémantiques.

Dans cette approche le contenu sémantique des documents est représenté dans le réseau sémantique, limité à des mots clés choisis par des critères statistiques, sans tenir compte ni de la structure logique des documents ni de la structure de la langue. La complexité de ce réseau nécessite que la plupart des opérations de mise à jour, et l'évolution du corpus, soient assistées par l'utilisateur, de même que la reformulation des requêtes.

2.2.6. Les Modèles Fondés sur la Logique

Ces modèles font appel aux mécanismes de déduction et de résolution de la logique de premier ordre pour l'établissement de la fonction de correspondance. En général, ces systèmes modélisent les connaissances sur le domaine par un ensemble de règles et de faits. A partir de ces faits et règles, les requêtes de l'utilisateur sont résolues par des mécanismes d'inférence en sélectionnant les documents pertinents. PROLOG est d'habitude à la base de tels systèmes.

Un autre domaine assez proche qui fait appel à la logique pour la représentation des connaissances est celui des Bases de Données déductives, où un ensemble de faits et de règles et un mécanisme d'inférence logique permettent de résoudre les requêtes. Cette représentation permet aussi l'implantation de mécanismes de contrôle de la cohérence et d'intégrité de la base, ainsi que le traitement de connaissances incomplètes [Oli86]. Par des axiomes, il est possible de déduire de nouveaux faits à partir des faits explicités dans la base, ou d'autres également dérivés.

Dans ce cadre, les requêtes d'un SRI sont vues comme des formules du calcul de prédicats, et la représentation des données du SRI est fondée sur des axiomes (pour la Relation d'Indexation), et sur des prédicats pour les relations sémantiques qui constituent traditionnellement le thésaurus. La fonction de correspondance est alors définie par des règles de déduction [Def86]. Néanmoins, la définition statique de ces relations, ainsi que la vérification de l'intégrité complète des données du SRI posent encore beaucoup de problèmes d'implantation. A l'heure actuelle, les systèmes existants pour le stockage et la manipulation de données (bases de données ou autres), fondés sur la logique, ne sont pas suffisamment efficaces.

K. J. van Rijsbergen a proposé dans [Rij86] un nouveau *paradigme* pour traiter les problèmes de la Recherche d'Information, fondé sur des logiques non-classiques, pour gérer l'incertitude inhérente dans l'application d'inférences, en définissant le principe logique de **l'incertitude**.

Son point de départ est la mise en évidence du fait que, par des mesures statistiques de la présence ou l'absence de mots dans un document, la notion de **sens** n'est pas atteinte. Les méthodes classiques, telles celles décrites précédemment, même si elles représentent ce qui se fait de mieux dans le domaine, ne sont pas satisfaisantes pour résoudre les problèmes de Recherche d'Information. Le problème réside non seulement dans la représentation de la sémantique d'un texte, mais également dans son utilisation pour l'obtention de documents en réponse à une requête [Rij86].

Sa réponse est l'utilisation d'une logique appropriée, pour **interpréter** le **sens** représenté dans le **modèle sémantique** associé. Les documents, ainsi que les requêtes, sont vus comme des phrases logiques. Les documents sont donc vus comme des ensembles de phrases interprétées dans la sémantique, une requête comme une phrase, et une opération de recherche est considérée comme l'évaluation d'une implication incertaine.

Donc, un document est retrouvé s'il implique logiquement la requête, avec une notion de plausibilité dans l'implication. La plausibilité est résolue en termes de mesures de probabilité d'occurrence d'événements, ce qui rejoint, d'une certaine manière, le modèle probabiliste. L'inférence est faite toujours en fonction du modèle sémantique du document.

Dans le cadre du projet RIME, mené actuellement dans notre Laboratoire, un modèle sémantique des documents, et un module d'interrogation en langue naturelle applicant cette théorie sont en cours de développement [Ber88] [Nie88] [Nie89].

2.3. Le Prototype IOTA

Dans cette partie nous faisons une courte description de IOTA, un prototype de Système Intelligent de Recherche d'Information, développé dans le groupe SIRI du LGI [Bru87] [Ker84] [Def86] [CDB86] [CD 87][BP 86] [Pal89].

IOTA est un SRI construit autour des deux fonctions fondamentales d'un Système documentaire. Une fonction d'Indexation, qui analyse et extrait du contenu du corpus documentaire les concepts sémantiquement les plus importants de chaque document, et une fonction d'interrogation, à l'aide de laquelle l'usager émet des requêtes, le système retournant les documents (ou les références aux documents) qui *correspondent* le mieux à celles-ci. Ces deux fonctions utilisent un thésaurus

dans lequel sont stockés les termes du domaine du corpus, ainsi que leurs liaisons sémantiques.

Le corpus documentaire à analyser est, évidement, écrit en langue naturelle (le français en l'occurrence), de même que les requêtes. Pour l'analyse de leur contenu il est donc nécessaire d'avoir un analyseur syntaxique. La connaissance du système sur le domaine est contenue dans le thésaurus, et le contenu des documents est indexé de manière automatique, pour construire la Relation d'Indexation.

Les modules du système sont donc : le module d'analyse syntaxique [BP 86] [Pal89], le module de construction du thésaurus [CDB86] [Bru85], le module d'Indexation automatique [Ker84] et un système expert pour la fonction d'interrogation [Def86] [CD 87].

Le corpus sur lequel IOTA a été testé porte sur un domaine technique (les NEF, Normes d'Exploitation et de Fonctionnement du CNET) ce qui spécialise, d'une certaine manière, le vocabulaire et le domaine sémantique à traiter. Dans le cadre du projet CONCERTO [Ker84], ont été testés les modules d'indexation automatique et de construction du thésaurus. Le corpus consiste d'environ 30.000 mots significatifs, correspondant à environ 4000 formes lexicales différentes (cf. 2.3.2), et organisés en 15 chapitres avec plus d'un millier d'entités structurelles. La maquette a été réalisée en LELISP, en utilisant MENTOR (dévélopé à l'INRIA) pour manipuler les textes des documents. Le prototype IOTA, qui intègre le système expert d'interrogation, a été développé sur UNIX. Des tests ont été effectués pour des raisons pratiques sur un sous-ensemble du corpus [Def86].

Notre travail s'appuyant sur ce système, nous donnons ci-dessous une description générale de ses composants, nécessaires à une bonne compréhension des éléments présentés dans les chapitres suivants.

2.3.1. Le Module d'Analyse Syntaxique

Le prototype IOTA a été conçu comme un SRI intelligent, capable d'interpréter les requêtes de l'usager et de traduire leur contenu en des termes connus du système. Il est donc indispensable dans cette démarche, que le système soit capable de comprendre la langue la plus naturelle pour les utilisateurs, à savoir le français. Les approches fondées sur l'interrogation sur des mots isolés s'étant avérées insuffisantes, il faut donc pouvoir traiter des composantes syntaxiques plus complexes tant dans l'analyse des textes que dans celle des requêtes, afin de mieux cerner la sémantique des documents et des requêtes.

L'analyse de la langue naturelle est importante aussi bien au niveau de l'interrogation que de l'indexation. Dans IOTA a été développé un module d'analyse syntaxique du français, indépendant du domaine d'application [BP 86] [Pal89]. L'objectif n'étant pas de *comprendre* du français, mais de reconnaître certaines formes syntaxiques prédéfinies dans les corpus, il n'est pas nécessaire de procéder à une analyse syntaxique complète des phrases. Une **analyse de surface** est suffisante pour regrouper et classer les mots, l'objectif étant l'extraction des termes d'indexation potentiels (cf. Paragraphe suivant). La portée de la reconnaissance est limitée à la phrase, ce qui n'est pas trop restrictif dans le contexte d'une indexation ultérieure. Si l'on voulait une interprétation plus poussée, il serait indispensable d'effectuer une analyse syntaxique et sémantique plus complexe (comme dans c'est le cas dans le système RIME), pour lequel un module de compréhension de la langue naturelle et d'interprétation des requêtes est actuellement en cours de développement [Nie89] [Nie88].

Le module d'analyse syntaxique de IOTA est formé par deux composants, un sous-module d'analyse morphologique chargé de la segmentation du texte en morphèmes, décomposés à leur tour en racine et désinence [Pal89], en y ajoutant l'information grammaticale comme la catégorie, le genre, etc. Le sous-module d'analyse syntaxique proprement dit, résout les ambiguïtés grammaticales et regroupe les mots en syntagmes (groupes nominaux considérés comme porteurs de l'essentiel de

l'information sémantique) selon un modèle conforme aux règles syntaxiques du français.

Les groupes nominaux sont ensuite transformés sous une forme canonique (appelés **groupes conceptuels**), d'après des règles de formation linguistiques. Les éléments grammaticaux retenus sont les substantifs, les adjectifs qualificatifs et numéraux, les participes passés, les verbes à l'infinitif et certaines prépositions. Pour des exemples de formation des groupes conceptuels, il faut se rapporter à [Pal89] et [Ker84].

Le module d'Indexation Automatique fonctionne sur la donnée des groupes conceptuels ainsi produits.

2.3.2. Le Module d'Indexation Automatique

Le but de ce module est l'extraction des concepts *"intéressants"* d'un document, c'est à dire, qui peuvent être demandés lors de l'interrogation. La méthode d'Indexation Automatique pour des textes en langue naturelle qui a été définie dans IOTA [Ker84] utilise le thésaurus comme modèle de connaissances pour normaliser les **concepts** (groups conceptuels) sélectionnés lors de la phase d'analyse et les transformer en **Termes d'Indexation**.

Le processus d'Indexation tient compte des propriétés structurelles des documents : le corpus est considéré comme un ensemble d'**entités textuelles**, qui correspondent à des sous-arbres des structures logiques des documents (qui sont toujours arborescentes). Les éléments spéciaux dans la structure, comme les titres, qui véhiculent une information sémantiquement importante, sont traités spécifiquement.

Les termes d'indexation sont normalisés via un processus de *"pattern matching"* avec les éléments du thésaurus, ce qui permet de contrôler la taille des termes. Après ce processus, à chaque élément dans la structure logique est associé un ensemble de

termes d'indexation. Cette association sert de base à la construction de la **Relation d'Indexation**, qui est définie comme un ensemble de triplets de la forme :

(Terme d'Indexation, référence textuelle, poids)

Le poids correspond à l'estimation de la représentativité du terme par rapport au contenu de l'unité textuelle analysée. Cette estimation est faite, comme dans la plupart des SRI, par des critères statistiques. Elle tient compte de l'importance du terme, aussi bien au niveau local (par rapport aux autres termes du document) qu'au niveau global (par rapport aux autres documents du corpus) [Rij79] [SM 83] [Ker84].

Par exemple, les triplets :

("structure de documentation", chapitre_1, 0.2) et

("structure de documentation", chapitre_2, 0.5)

indiquent que le terme "structure de documentation" est associé au chapitre_1 et au chapitre_2 du document, mais que c'est dans le chapitre_2 qu'il est le plus représentatif.

Dans IOTA ces évaluations correspondent à la *représentativité du concept par rapport au contenu de l'unité textuelle*, et à la *représentativité de l'unité textuelle par rapport au concept* [Ker84]. La référence textuelle est la manière unique dans le système d'identifier l'unité textuelle concernée, parmi l'ensemble du corpus. Il s'agit en fait d'un chemin d'accès dans une forêt d'arborescences.

Partant des unités textuelles du niveau le plus bas, le processus d'indexation est appliqué de manière récursive aux niveaux supérieurs (stratégie de **remontée** sélective des termes d'indexation, et de réévaluation des critères de représentativité). Ceci permet notamment d'associer à chaque terme d'indexation les unités textuelles pour lesquelles ces mesures sont les meilleures dans la hiérarchie [Ker84].

2.3.3. Le Thésaurus

IOTA comporte un module chargé de l'acquisition automatique de toute l'information connue par le système sur le domaine sémantique du corpus [Bru82] [Bru85] [Bru87]. Le thésaurus est construit à partir d'un ensemble de documents jugés caractéristiques de ce domaine. Il est cependant irréaliste de penser à construire *à priori* un thésaurus décrivant *tout* un domaine ; il ne représente donc qu'un modèle de *connaissances partielles* qui peut être enrichi manuellement par la suite.

Le thésaurus est un graphe qui exprime les relations possibles entre les concepts. Ces relations sont de type hiérarchique, comme par exemple *générique*, *spécifique*, et non hiérarchiques comme *voir aussi* et la *synonymie*. L'identification des relations sémantiques nécessite un processus assisté, dans la mesure où l'identification des relations sémantiques est extrêmement complexe et dépendante du contexte.

La construction du thésaurus passe par une étape préliminaire consistant à construire une matrice terme-terme dont les éléments expriment une force de liaison contextuelle des mots (mesure fondée sur la fréquence de cooccurrence des mots, leur distance dans un texte). On considère ensuite le graphe associé à cette matrice, et on en extrait les sous-graphes maximaux complets (**cliques**).

Par construction, cette extraction est limitée aux cliques pouvant dériver des groupes nominaux (sélection des catégories lexicales des mots). Chaque clique est considérée comme le représentant d'une classe de concepts : l'ensemble de groupes nominaux pouvant être construits à l'aide de ses composants. Les relations sémantiques sont ensuite identifiées de manière assistée par une analyse des contextes communs à chaque clique. Une présentation détaillée de cette approche peut être trouvée dans [Bru85], [Bru87].

2.3.4. Le Module d'Interrogation

La fonction d'Interrogation de IOTA est assurée principalement par un système expert [Def86], qui donne à l'usager un maximum de flexibilité sur le langage dans lequel sont exprimées ses requêtes, et qui vise à obtenir l'"intelligence" nécessaire pour implanter, à partir des requêtes et des connaissances du système, une fonction de correspondance satisfaisant au mieux les besoins de l'usager.

Le système expert établit une distribution des tâches entre la partie procédurale d'un SRI, et la partie déclarative qui concerne la mise en œuvre des connaissances expertes utilisées dans certaines étapes du processus d'interrogation (évaluation des réponses et reformulation de la requête essentiellement).

Le module d'Interrogation fournit une interface conviviale, en langue pseudo-naturelle, pour sélectionner des documents. Pour donner une bonne interprétation des requêtes, et assister l'usager dans sa démarche, la reformulation automatique des requêtes en fonction de la typologie de l'utilisateur, de la qualité estimée des réponses obtenues, et des connaissances de synthèse sur le domaine (thésaurus) a été introduite [Def86] [CD 87]. Le système gère les documents textuels structurés de IOTA, ce qui permet d'obtenir comme réponse les unités textuelles les plus pertinentes. Les réponses peuvent être ordonnées d'après les mesures de représentativité (cf. module d'indexation).

Dans ce module on peut distinguer :

- Un préprocesseur de requêtes, qui extrait les concepts exprimés en langue naturelle, et détermine à partir d'une analyse morphologique et syntaxique les connecteurs logiques entre eux.

- Un sous-module de *pattern matching* qui effectue deux étapes : la première transforme les concepts extraits de la requête vers ceux connus par le système (du thésaurus). Le thésaurus est alors utilisé comme un "index flou", où le

critère de sélection est l'inclusion du concept dans une clique, ou bien sa décomposition en une expression de termes syntaxiquement plus simples pour la reformulation de la requête. La deuxième étape est l'accès à la relation d'indexation, pour sélectionner les références aux documents satisfaisant la requête [CDB86]. Cette opération implique un processus d'évaluation de la pertinence des références sélectionnées (à partir des mesures de représentativité fournies par la relation d'indexation).

- Un sous-module d'évaluation qui sélectionne, dans les références retrouvées, celles qui correspondent le mieux à la requête initiale.

2.4. Conclusions

Plusieurs modèles de recherche d'information ont été présentés, ainsi que des exemples de SRI fondés sur ces approches, afin d'illustrer la variété des stratégies utilisées dans le domaine de la Recherche d'Information en vue d'obtenir des outils d'accès aux documents suffisamment puissants, prenant en compte de manière efficace la sémantique des requêtes et le contenu sémantique des documents.

En général, ces méthodes font appel à des mesures statistiques et des critères linguistiques ou positionnels pour établir une *pondération* de l'importance d'un concept par rapport à un document en particulier, et au corpus en général. Ces mesures sont utilisées au moment de l'interrogation pour donner à chaque document retrouvé une mesure de sa *pertinence par rapport à la requête*. Les algorithmes de calcul des coefficients de pondération, et les critères de classification et d'ordonnancement des réponses sont très variés selon les modèles considérés. En fait, on trouve donc beaucoup de variantes des grands axes présentés dans ce chapitre.

Une bonne comparaison qualitative des deux approches les plus classiques, les fichiers inverses et le regroupement ("clustering") est développée en [Voo86] : elles sont comparées quant à l'espace nécessaire pour leurs méthodes d'accès ("overhead") et le temps de réponse moyen sur quatre collections de documents et un ensemble de requêtes de test. Les méthodes de regroupement semblent assez sensibles à la définition des représentants (centroïdes), aussi bien pour l'espace utilisé que pour le temps de réponse. L'inversion de termes semble être en tous les cas plus économique en place et en temps de réponse, mais plus rigides par rapport à des extensions vers des termes d'indexation en langue naturelle.

Une caractéristique commune à pratiquement toutes les approches présentées est le fait de modéliser les connaissances du domaine d'application de manière statique. Cette connaissance doit être connue au moment d'indexer les documents et, évidemment, lors de l'interprétation des requêtes. Sa création, que ce soit par des méthodes manuelles, statistiques ou linguistiques est toujours très coûteuse, et nécessairement partielle pour des domaines non fermés.

Dans le cas de IOTA, un problème critique réside dans la taille du corpus effectivement gérable dans l'état actuel du prototype, et dans celle des structures nécessaires à l'indexation et à l'interrogation. Un autre problème est lié aux aspects dynamiques qui ne sont actuellement pas gérés : la croissance du thésaurus et de la Relation d'Indexation liées à l'évolution du corpus. Une voie à explorer est donc l'étude des techniques de compactage pour permettre une gestion dynamique et plus efficace du thésaurus et de la Relation d'Indexation, qui constituent les éléments les plus sensibles d'un SRI.

Rappelons enfin que cette évolution dynamique du corpus est fondamentale dans les cas des SIB, dans lesquels il peut y avoir des suppressions ou des modifications de documents, fait non considéré dans les SRI classiques.

3 . LA MODELISATION DES DOCUMENTS DE BUREAU

3.1. Introduction

Dans le chapitre précédent nous avons présenté la manipulation d'un corpus documentaire dans le cadre des Systèmes de Recherche d'Information. Dans ce chapitre nous présentons les principales orientations de la modélisation des documents dans le cadre des Systèmes d'Information Bureautique, afin de montrer les opérations qui sont proposées et l'environnement dans lequel elles sont réalisées. Nous nous concentrons notamment sur celles qui tiennent compte de la gestion des documents, de leur stockage et des méthodes d'accès associées. Nous dédions une attention particulière aux modèles qui mettent en œuvre du matériel spécialisé, car notre intérêt est l'intégration des fonctionnalités des SRI, souvent coûteuses en temps et en espace, points critiques dans le SIB.

Les documents ont été traités au départ dans le domaine des Bases de Données, comme un type d'information multimédia. La manipulation de ces types d'information, ainsi que les nouvelles technologies (grandes mémoires, disques optiques, terminaux à haute résolution, processeurs dédiés), ont ouvert les portes vers le développement de systèmes de support à l'activité bureautique. De la même manière que pour les SRI, plusieurs approches font actuellement l'objet de recherches. Nous pouvons remarquer parmi toutes les "pistes" suivies les Systèmes de Bases de Données Multimédia, les modèles Orientés-Objets et les Bases de

Données Déductives. Du point de vue des méthodes d'accès aux données textuelles, nous avons exploré en détail les méthodes de compression et les accélérateurs de filtrage de données.

L'approche des Bases de Données Déductives pour la modélisation et l'accès au contenu des documents s'appuie sur la logique de prédicats de premier ordre. Pour l'accès par le contenu des documents, et à notre connaissance, les systèmes proposés actuellement n'apportent rien de nouveau par rapport aux SRI fondés sur des règles de production et sur des réseaux sémantiques (cf. 2.2.4 et 2.2.5).

Nous présentons par la suite la gestion des documents dans les autres approches de Bases de Données, et les méthodes d'accès par le contenu des documents.

3.2. La Gestion de Documents Multimédia - Etat de l'Art

Dans plusieurs domaines, comme le génie logiciel, la CAO ou la bureautique, il est nécessaire d'intégrer, à la gestion de données traditionnelles, celle des types de données dits *multimédia*, comme les textes, les images, la voix, les graphiques. Les nouvelles technologies ont ouvert la voie à l'exploitation de gros objets, permettant la conception de systèmes de manipulation de types complexes de données et des liens entre ces données.

La notion d'*objet multimédia*, est apparue dans les contextes de *systèmes orientés objet* et des *bases de données généralisées*, comme l'entité élémentaire contenant de l'information multimédia. Leur manipulation implique des méthodes d'accès et de stockage spéciales, ces objets étant en général gros et complexes [Ban88], [Chr83], [CF 84b], [Chr84], [Tsi84]. La combinaison de ces objets et des données de types classiques apparaît dans le contexte bureautique notamment dans la spécification des *documents multimédia*. La définition du type de

données **document** se trouve dans des modèles de SGBD généralisés, parmi lesquels nous distinguons le modèle TIGRE [LPV83] [Vel84] [Vel85a] [Vel85b], et dernièrement dans les systèmes Orientés Objets [BDN88].

3.2.1. Les documents structurés dans les SGBD Généralisés - Le modèle TIGRE

Le SGDB Généralisé TIGRE [LPV83] [Vel84], développé en coopération entre le L.G.I et le Centre de Recherches BULL, est un serveur de données qui offre des outils pour manipuler des documents, les retrouver, les insérer et les diffuser vers des postes de travail. Dans TIGRE, un des types de base est le type **Document**, qui permet de modéliser les documents dont ont besoin des applications (article, lettre, etc), en tenant compte de leur structure logique et de la présence d'éléments de types de données classiques et d'éléments multimédia. L'intégration de ces éléments est une entité appelée **document généralisé**.

Le modèle TIGRE est une extension du modèle Entité- Relation [Che76], pour l'enrichir de la sémantique des schémas des applications et de la manipulation des données. Pour la modélisation, nous trouvons les concepts d'agrégation et de généralisation, ainsi que la définition de nouveaux types de données à l'aide de constructeurs. En particulier, un type document est construit par le constructeur **Document.** Les langages de définition et de manipulation de données présentent la particularité d'être associés dans un seul langage appelé LAMBDA. La définition de données est fortement typée et les énoncés d'interrogation sont de type ensembliste. L'utilisateur exprime le contexte de sa requête et les conditions à satisfaire.

Les entités de type **Document** sont construites à l'aide d'opérateurs de composition qui définissent les éléments de structure (nœuds ou feuilles) dans la hiérarchie. Cette définition est faite à partir d'objets déjà définis, qui a leur tour peuvent être soit des composants élémentaires (appelés *unités*), soit des

éléments de structure, soit des structures de type **Document** déjà définies.

Les *unités* peuvent être des objets de type Texte, Image, Graphique, Symbole Mathématique, Référence à un élément associé ou à une partie de la structure. Les éléments associés peuvent être des références bibliographiques, des commentaires, des paramètres, des fonctions, des notes de pied de page, etc. Les paramètres et les fonctions permettent l'échange de données entre les documents et d'autres éléments de la base. Par exemple, au moyen des paramètres il est possible d'accéder directement (par son nom) une partie variable du document, sans avoir à passer par la structure [Vel85b].

Les constructeurs de structure sont :

- L'agrégat d'objets, mis en séquence : **BEGIN** ... **END**
- La liste d'objets d'un même type : **List** (min, max) **of** ...
- Le choix entre plusieurs objets : **case** s **of** $v_1 : e_1, \ldots, v_n : e_n;$

  s est appelé le sélecteur, v_i sont les valeurs du sélecteur et e_i sont les éléments correspondants dans chaque cas.

Exemple :

```
type article : Document
structure
     Begin
     titre : string;
     auteurs : List (1, *) of string;
     date_création     : date;
     date_publication : date;
     corps : cas type_report of
          rapport_complet : List (1, *) of Chapitre;
          résumé              : List (1, 5) of Paragraphe;
     end;
```

```
type conférence : entity
     nom   : string;
     ville  : string;
     articles_soumis : List (1, *) of article;
end;
```

Les types "**date**", "**Chapitre**" et "**Paragraphe**" doivent être définis auparavant, comme c'est le cas dans des langages fortement typés comme PASCAL.

TIGRE permet la sélection de documents par leur contenu en donnant comme clause de recherche une expression conjonctive de chaînes de caractères. Ces chaînes sont des expressions régulières pouvant contenir des caractères "*joker*" (au début, à la fin ou à l'intérieur même de la chaîne). Cette opération a été implantée à l'aide d'automates d'états finis avec mémoire [Jim85], qui seront décrits brièvement dans le chapitre suivant.

LAMBDA est un langage conçu pour être intégré dans un langage de programmation typé. Pour l'interface avec l'utilisateur final, un module graphique conversationnel a été développé. Du point de vue de l'interrogation, LAMBDA permet :

- De retrouver des ensembles de documents à partir de leurs liaisons sémantiques avec d'autres éléments de la base ou par leur contenu textuel.

- L'utilisation de trajets et de variables dans les prédicats d'interrogation. Ceci permet d'exprimer les liaisons sémantiques entre les données (associations, agrégations, généralisations). Les trajets peuvent être linéaires ou arborescents.

- La possibilité de manipuler de manière homogène des ensembles de données factuelles et des ensembles de données textuelles.

Exemple :

```
SELECT   art.titre
```

```
FROM    articles_soumis 1..* art of conférence c
WHERE   c.ville = "Grenoble"
 AND    art.corps.type_report = résumé
 AND    *docum* = Paragraphe 1..* of art.corps
```

permet de retrouver les titres des articles soumis aux conférences de Grenoble, dont le résumé parle de "document", "documentation", etc. Le caractère "*" est un "*joker*", permettant de spécifier n'importe quelle chaîne de caractères dans les paragraphes du résumé.

3.2.2. Les documents dans les Systèmes Orientés Objet (SOO)

L'approche des modèles *Orientés-Objet* est particulièrement intéressante pour le développement des applications de bureau, en raison de la facilité d'introduction des traitements spécifiques aux objets complexes et de longue-durée (*persistants*). Dans [NT 85] nous trouvons une analyse complète des caractéristiques de l'environnement bureautique, dont les procédures de bureau, les événements (*activités* dans le cadre du SIB), les mécanismes transactionnels nécessaires, etc.

Dans les systèmes orientés objets les documents sont généralement traités comme *objets complexes*, ou tout simplement comme n'importe quel autre type d'objet. A l'heure actuelle les SOO sont en cours de définition, la notion même d'**objet** n'étant pas encore standardisée [Ban88]. Nous pouvons dire que dans la plupart des cas, les objets sont vus comme une définition, un ensemble d'opérations associées (appelées aussi "méthodes"), une instantiation (valeur), et une implantation (la manière dont les opérations sont réalisées), avec des notions d'encapsulation (types abstraits de données) et d'héritage de propriétés. En particulier, le système doit offrir la facilité de définir les méthodes d'accès appropriées à chaque type d'objets. La façon de gérer les documents n'est pas toujours explicite, mais ils seraient des objets dont la définition et l'implantation

font l'abstraction de leur complexité, de manière à les rendre semblables à tout autre objet du modèle.

A l'heure actuelle, les travaux fondamentaux développés dans les SOO et les SGDB Orientés Objet sont centrés sur la définition de systèmes extensibles et modulaires, pour tenir compte des besoins et des caractéristiques particulières des objets des applications multimédia. Les sujets abordés sont donc le nommage, l'adressage, la persistance et l'accès aux objets, en tant qu'entités de base du modèle. Très peu de travaux cependant font une référence explicite à la gestion de documents et à leurs opérations associées.

En ce qui concerne la gestion des objets, dans [BDN88] est discutée la problématique de l'adressage, du nommage et de la persistance. La taille est la caractéristique qui différencie les documents, les images et les objets de CAO des autres objets. Par exemple, la gestion de la mémoire doit tenir compte de gros et de petits objets, ce qui a des fortes implications sur la manière dont sont effectuées des opérations telles que la pagination et les échanges avec les dispositifs physiques de stockage.

Dans [PTS88] est présenté le système GEODE, un serveur d'opérations sur objets complexes avec une architecture orientée grandes mémoires. Les problèmes abordés sont notamment le stockage, l'accès concurrent et la gestion de versions, vus sous une optique générale à tous les types d'objets.

A notre connaissance, les opérations d'accès par le contenu des objets "document" n'ont pas été discutés de manière spécifique. Elles seraient définies dans chaque système ou application, et en général les travaux à ce sujet mentionnent seulement des opérations sur les attributs externes des documents. C'est le cas, par exemple, de l'algèbre de documents structurés de bureau dans [GZC87]. Les discussions portent toujours sur les notions de modélisation des objets et de l'environnement en question (CAO, génie logiciel, bureautique, etc). Les suggestions faites quant à l'accès aux documents portent sur l'exploitation de leur structure logique (chapitres, titres, etc), le partage et la gestion de

versions et historiques (comme pour les autres objets), mais les opérateurs sur le contenu même du document constituent encore un problème ouvert.

3.2.3. Les Méthodes de Signatures pour l'accès aux documents

Pour les systèmes gérant de l'information multimédia, en particulier des messages et des documents textuels, les équipes de recherche dirigées par D. Tsichritzis, S. Christodoulakis et C. Faloutsos, entre autres, ont proposé des méthodes d'accès au contenu des textes fondées sur des méthodes de hachage, appelées des **Méthodes de Signatures** [CF 84], [Lar83], [BM 85], [Chr84], [CT 83].

La *signature* d'une donnée est une **image compressée** de son contenu : dans un premier temps, elles ont été conçues pour des données classiques, afin de coder dans très peu de place une intervalle de valeurs. Les opérations d'accès associées sont des *recherches partielles*. Ces méthodes ont été proposées, entre autres, dans [Knu73] [PB 80].

Exemple :

Pour coder le nombre d'enfants des employés d'une entreprise, on peut établir, sur 2 bits, le codage suivant :

0 enfants	:	0 0
1 - 2 enfants	:	0 1
3 - 5 enfants	:	1 0
6 enfants ou plus	:	1 1

au lieu de l'avoir sur 32 bits qui correspondent à une valeur entière.

Evidemment, ce codage entraîne une perte d'information, mais au niveau de l'application elle peut être considérée comme non dommageable.

Une autre méthode de calculer la *signature* d'une donnée est le **codage superposé** ("Superimposed Coding" en anglais), qui consiste en la superposition des images de plusieurs valeurs dans un seul ensemble de bits. Nous discuterons cette technique un peu plus loin. Elles sont présentées dans [FH 69], [Rob79].

L'utilisation des méthodes de signatures a été reprise par Tsichritzis et Christodoulakis dans [TC 82] pour les appliquer à des données textuelles, et permettre donc un accès rapide à l'information multimédia (des messages) par leur contenu, avec un surcoût de stockage ("overhead") raisonnable. Leur proposition est basée sur une méthode de *sélection des messages d'après un filtre de recherche*. Ce filtre permet d'écarter les documents non intéressants pour l'utilisateur, et d'afficher les messages sélectionnés pour qu'il puisse y choisir ceux qui sont les plus pertinents. L'hypothèse fondamentale dans la définition du filtre, est qu'il peut ne pas être exact, car c'est l'utilisateur, grâce à l'affichage, qui fait la sélection définitive. Une autre raison est que l'utilisateur, en général, ne connaît que partiellement le contenu des messages qu'il recherche, et ce n'est que pendant l'affichage qu'il les reconnaît. Nous remarquons que cette problématique rejoint tout à fait celle, plus générale, des SRI.

Cette sélection de messages implique un balayage séquentiel, pour lequel il faut une méthode d'archivage et d'accès appropriées. Par contre, elle évite la tâche, assez difficile et couteûse, de maintenir un classement des messages par leur contenu, ainsi que des index par mots clés. En fait, les systèmes classiques de classement et d'index par mots clés (cf. chapitre 2) imposent des lexiques prédéfinis, ils sont très peu flexibles, et l'interrogation sur des textes pleins est imposible, tout en ayant des grands surcoûts en place. Ces contraintes sont très fortes dans un environnement ouvert et dynamique comme celui du bureau, et les méthodes de signatures permettent de trouver un équilibre entre ces objectifs et le temps d'évaluation de requêtes.

Dans le cadre des environnements bureautiques, on peut généraliser ces concepts aux documents textuels en général, l'objectif étant de présélectionner les documents pertinents à une requête en utilisant une image compacte de leur contenu, la **signature du document**. Elle est constituée d'une chaîne de bits calculée par un algorithme de transformation des chaînes de caractères.

Lors de l'interrogation, le même algorithme est appliqué sur la requête, ce qui permet de faire l'évaluation de la correspondance requête - documents, à travers leur signature, donc par des opérations sur des chaînes de bits. Ceci simplifie aussi l'évaluation des opérateurs logiques dans la requête. Ce processus de comparaison de chaînes de bits pour présélectionner les documents, constitue donc un *filtre*, qui par sa nature (transformation non bijective du contenu des documents) garantit que, au moins, tous les documents pertinents sont retrouvés.

Pour établir la fonction de transformation, plusieurs algorithmes sont proposés . Dans [TC 82] est proposée une méthode de calcul des signatures du contenu d'un message, les **signatures par mot** ("Word Signatures"). Dans [CF 84] cette méthode est comparée aux **signatures par codage superposé** ("Superimposed Coding Signatures" [CT 83] [Tsi84]) et il est fait une étude analytique des deux méthodes. Dans [RZ 84] est faite aussi une analyse complète dans le contexte de systèmes de bureau des méthodes de signatures ainsi qu'une comparaison avec les méthodes d'inversion des mots d'un texte.

Nous allons décrire en détail le principe des méthodes de calcul de signatures : les signatures par mot, puis les signatures par codage superposé, et nous terminons par une courte discussion.

3.2.3.1. La méthode de Signatures par Mot (WS)

Dans cette méthode de calcul de signatures, on extrait les mots significatifs par élimination des mots non significatifs qui sont stockés dans une liste prédéfinie de "mots vides". Chaque mot est

transformé en une chaîne de bits, qui peut être, soit de longueur fixe, soit de longueur variable avec un entête. Cette chaîne constitue la *signature d'un mot*. Toutes les chaînes sont concaténées, les unes après les autres (dans l'ordre du document) pour former la *signature du document*.

Exemple :

Texte	**Signature** (Signatures par mot, de longueur fixe = 5 bits)
Processus	01001
Communicant	10100
Parallèle	00101
Signature du Texte :	01001 10100 00101

Afin de permettre la recherche de parties de mots, il faut que la fonction de transformation préserve l'ordre des sous-chaînes de caractères. Dans [Fal86] est proposée une méthode fondée sur des triplets de caractères consécutifs dans le mot. La signature d'un mot est ainsi générée par la concaténation des signatures des triplets.

Exemple :

Pour une signature de longueur 3 bits par triplet de caractères, la signature du mot *Expert* serait :

Triplets possibles :	"_Ex"	"Exp"	"xpe"	"per"	"ert"	"rt_"
Signature des Triplets :	100	001	101	110	010	110

Signature de "Expert" : 100001101110010110

Une qualité de cette méthode de calcul de la signature d'un texte, est qu'elle préserve les images des mots individuels et l'ordre de leurs composants. Malgré des bons rapports de compression, la taille du fichier des signatures reste proportionnelle à celle du fichier texte original. En fait, les mots qui apparaissent plusieurs fois occasionnent la génération d'autant de signatures

répétées. Ceci pourrait être évité par l'élimination des doubles, donc par la transformation des *mots différents* dans la construction de la signature du texte. Le taux de place mémoire nécessaire est considérablement réduit, mais on perd la séquence entre les mots, et l'interrogation comportant des "*joker*" au milieu des mots devient impossible.

3.2.3.2. La Méthode de Signatures par Codage Superposé (CS)

Le codage superposé comme méthode de compression est fondé sur le fait de faire correspondre, à un ensemble de données, une image sur un bloc de bits de taille fixe (donc, contrôlée). Utilisé d'abord sur les données classiques, elle a été étendue aux données textuelles dans [CF 84b].

Le principe de la méthode est la superposition d'un ensemble de signatures individuelles de mots, dans un seul bloc de bits. Le texte est divisé en blocs d'un nombre fixe *D* de mots non-vides (une liste des mots non porteurs d'information doit être construite, et ces mots éliminés du bloc en question). A chaque bloc ainsi défini est associée une chaîne de bits de taille fixe, *N*, initialisée à 0. Pour chaque mot dans le bloc de texte, un nombre *M* de bits dans le bloc est mis à 1, avec $M << N$. La signature du bloc est formée par le *OU* logique des signatures individuelles des mots.

Exemple :

Si le texte est "Bases Bibliographiques Intégrées", sa signature CS avec les paramètres D = 3, N = 8, M = 2, est :

Texte	**Signature (CS)**
Bases	00101000
Bibliographiques	01000010
Intégrées	10000100

Signature du Texte : 11101110

Pour l'interrogation, la signature de la requête S(Q) est calculée avec le même algorithme de transformation, donc un bloc de N bits initialement à 0, et M bits par mot sont mis à 1. S(Q) est comparée aux signatures des blocs du texte, par des simples opérations logiques sur des chaînes de bits : le bloc du texte est sélectionné si tous les bits "1" dans le bloc signature de la requête sont aussi mis à "1" dans le bloc signature du texte. Les opérations logiques exprimées dans la requête peuvent aussi être examinées sur les signatures, par des opérations sur les bits.

Exemple :

Dans l'exemple précédent, si la requête porte sur le mot "bibliographique", la signature de la requête est:

Q = "Bibliographiques" **S(Q)** = 01000010

et le contenu du texte a comme signature :

Signature du Texte = 11101110

Les positions du bloc de bits à examiner sont la 2ème et la 7ème, car elles sont "1" dans la signature du bloc de texte, donc il est sélectionné.

Q = "Intelligent AND Expert"

S (Intelligent) = 0001001100

S (Expert) = 0100010100

S (Q) = 0101011100

Signature du Document:

S (Bloc1) = 0001011110

S (Bloc2) = 0101011101

Dans ce cas, le Bloc2 est sélectionné, mais pas le Bloc1, son deuxième bit étant "0".

Il faut noter que, pour un mot donné dans la requête, sa signature peut former une configuration de bits S(Q) telle que un bloc soit sélectionné même si le mot n'est pas dans le texte. Ceci est appelé une collision, et elle est due à la méthode de transformation et par le *OU* logique des signatures individuelles.

Exemple :

Dans le même exemple, si:

Q = "Réparties" S(Q) = 10100000

Signature du Texte = 11101110

La signature de "Réparties" est reconnue dans la signature du bloc de texte, même s'il n'y apparaît pas. Le bloc étant sélectionné, c'est une **collision**.

Les collisions peuvent être, soit tolérées par l'application, soit éliminées dans une étape postérieure de balayage ("Text scanning") séquentiel des blocs de texte sélectionnés. Dans [CF84] il est montré que, pour cette méthode, les paramètres qui définissent la performance d'évaluation sont N et M (la taille du bloc signature et le nombre de bits mis à 1 par mot). Avec un bon choix de ces paramètres, le CS produit une signature très compacte, d'environ 10% de la taille du texte original, avec environ 10% de collisions. Une bonne caractéristique de CS est que les mots répétés sont automatiquement transformés vers le même ensemble de positions dans le bloc signature, donc leurs signatures ne sont stockées qu'une seule fois au moment de faire le *OU* logique pour former la signature du bloc.

Pour permettre la recherche de sous-chaînes de caractères (parties de mots), de manière similaire à la méthode WS, les mots sont considérés comme un ensemble de triplets de caractères consécutifs, et les M bits de la signature d'un mot donné sont calculés par la mise à "1" de un bit par triplet. Si un mot contient moins de M triplets, les bits qui manquent peuvent

être ajoutés par des méthodes de hachage. Si le mot contient plus de M triplets, seul les premiers M seraient pris en compte [CF84].

Exemple :

Pour un bloc de signature de taille N = 20, et M = 6, la signature du mot *Expert* serait :

Triplets possibles	"_Ex"	"Exp"	"xpe"	"per"	"ert"	"rt_"
Signature des Triplets	bit 5	bit 12	bit 8	bit 2	bit 18	bit 15

Signature de "Expert" : 01001001000100100100

La taille du fichier signature, ainsi que la performance globale du système, sont fortement liées au bon choix des paramètres de la méthode. Néanmoins, le processus de filtrage est toujours proportionnel en temps à la taille des documents [CF84].

3.2.3.3. Autres Méthodes de Calcul de Signatures

Dans [FC 87], [Fal86] sont présentées, de manière assez claire et synthétisée, plusieurs autres méthodes de calcul de la signature d'un document, toujours dans la même philosophie de création d'images compressées de son contenu. Nous y retrouvons les principes exposés dans les deux méthodes précédentes, avec des optimisations du point de vue stockage ou des taux de collisions.

Les signatures étant des vecteurs de bits creux (en moyenne, pour les meilleurs performances, la moitié du vecteur est mise à "1", l'autre moitié est "0", car ainsi le rapport collisons/place est optimal), une première méthode pour gagner en place requise par le fichier signature est l'utilisation de méthodes de compression de bits. Nous avons :

- Le codage par "Run Length" (RL) [FC 87] [Sed84], ou ne sont stockés que les distances entre "1"s successifs ("runs").

Exemple :

Pour le vecteur "0010000100100001"

son codage RL est : "3535"

Le contenu du document, comme pour la méthode CS, est transformé vers un ensemble de blocs de bits. Ces blocs étant assez longs, les collisions sont minimisées et le codage permet un gain considérable de place. D'après [FC 87], avec les bons paramètres, la redondance causée par la signature est de 0.086 bits par mot. Le principal problème de ce codage est le temps d'évaluation de requêtes : étant donné que pour décoder un ensemble de positions de bits dans un bloc RL il faut décoder et sommer tous les intervalles antérieurs ("runs"). Si on assume une distribution uniforme des requêtes, il faut décoder environ la moitié des intervalles ("runs") à chaque fois.

- Le codage par compression de blocs de bits (Bit-Block Compression) (BC) est fondé sur la partition du vecteur creux de bits en groupes consécutifs disjoints de bits. Chaque bloc ainsi formé est codé séparément, ce qui permet d'accélérer les recherches. Le codage de chaque bloc est formé de trois parties :

  Partie I : Un bit qui indique s'il y a des "1" dans le bloc,

  Partie II : Le nombre de "1" moins 1 dans le bloc (codé en binaire)

  Partie III : Les distances des "1" par rapport au début du bloc (codées en binaire).

  Si le bloc est tout à "0" seul la première partie est nécessaire.

  Exemple :

  Pour b = 4 bits, et le vecteur "00100000100000000011"

La division est : 0010 0000 1000 0000 0011

Son codage BC est:

Partie I :	1	0	1	0	1			
Partie II :	0		0		01			
Partie III :	10		00		10	11		

Concaténation des codes par bloc :

1 0 10 | 0 | 1 0 00 | 0 | 1 01 10 11

Vecteur Codé : "10100100001011011"

Concaténation par parties :

1 0 1 0 1 | 0 0 01 | 10 00 10 11

Vecteur Codé : "10101000110001011"

Au moment de l'interrogation il est nécessaire de décoder le vecteur en entier pour pouvoir le comparer avec la signature de la requête. Le stockage du vecteur codé peut se faire, soit en regroupant par bloc, soit en regroupant par Partie de codage, tel qu'il est montré dans l'exemple. C'est la deuxième possibilité de stockage qui est la plus performante lors de l'interrogation, parce qu'elle "résume" dans les bits de la partie I le contenu de tout le vecteur. C'est, de toutes les manières, un décodage laborieux à faire lors de chaque comparaison.

Les méthodes CS et WS sont des cas particuliers de BC, comme cela est montré dans [FC 87].

- Le codage par compression de blocs variables de bits (Variable Bit-Block Compression) (VBC) [FC 87] élimine les problèmes de définition de blocs de texte, et donc la considération du nombre de mots, D, qui les forment. L'idée est de calculer *b*, le nombre de bits par bloc du vecteur creux, en fonction de la densité de "1" qu'il contient. Cette valeur est stockée au début de la signature en question.

3.2.3.4. Discussion

Les méthodes de signatures permettent l'accès aux documents en utilisant une image compressée de leur contenu. Indépendamment de l'algorithme de calcul de cette image, ou de la manière de l'accéder et de la stocker, les signatures offrent une solution en deux étapes :

- Une première étape de **génération** des signatures, dans laquelle est construit le fichier image des contenus des documents.

- Une deuxième étape d'interrogation. Elle est à son tour composée de plusieurs pas :

  - La requête est codée suivant le même algorithme que pour la première étape.

  - Les signatures des documents sont comparées à la signature de la requête. Une première phase de présélection des documents (ou parties de documents) qui peuvent éventuellement répondre à la requête est faite.

  - L'élimination des collisions dans cet ensemble.

  Ce dernier pas est optionnel, dans la mesure où les applications peuvent accepter un pourcentage, relativement bas, de bruit dans les réponses. Si des réponses exactes sont nécessaires il est indispensable. L'élimination de collisions peut être évitée aussi, évidemment, par la spécification d'un algorithme de transformation qui ne produise pas de collisions, au prix d'une moins bonne performance dans les ratio de compactage.

Nous pouvons résumer ces deux opérations par le schéma ci-dessous :

Génération des Signatures :

Texte (T)
|
Calcul du bloc S (Bi) pour chaque bloc Bi du texte T.
|
Stockage de S(Bi) dans le Fichier Signature.

Etape d'Interrogation :

Requête (Q)
|
Calcul de la Signature de la Requête S (Q) avec le même algorithme.
|
Comparaison entre S(Q) and S(Bi) pour chaque bloc Bi
|
Présélection des blocs qui répondent a la requête.
|
Balayage des blocs présélectionnés pour éliminer les collisions.

3.2.4. Conclusions

Nous avons présenté brièvement les approches suivies dans différents domaines d'application pour traiter le problème de la manipulation et de l'accès aux documents.

Une caractéristique commune à la plupart des systèmes de gestion de documents bureautiques est le fait de présenter un modèle de documents, et la gestion d'attributs externes et de leur stockage. Les opération d'accès par le contenu aux éléments multimédia sont pratiquement absentes de ces systèmes. Nous faisons référence à la récupération de ces éléments en tant qu'objets complets du système, pour les passer ensuite aux applications telles que des éditeurs, le courrier électronique, des serveurs d'impression, etc. Ainsi, pour une donnée de type *graphique*, nous pouvons l'afficher, l'imprimer, mais nous n'avons pas de méthodes d'accès à son contenu.

Nous pouvons identifier trois approches fondamentales pour les méthodes d'accès associées aux documents, et en particulier à leur contenu textuel.

- **Le Balayage du Texte** : c'est la méthode la plus simple et générale, d'habitude fondée sur la théorie des automates d'états finis. Elle n'a pas besoin d'espace additionnel à celui nécessaire pour le stockage du texte. Par contre, le temps nécessaire aux opérations de recherche est proportionnel à la taille du texte, et donc prohibitif dans le cas de bases de documents de grosse ou moyenne taille. Les machines spécialisées en filtrage de texte sont souvent utilisées pour améliorer ces temps d'évaluation.

- **Les fichiers Inverses** : c'est la méthode la plus traditionnelle dans les systèmes commerciaux et beaucoup de prototypes de recherche. C'est la plus performante lors de l'interrogation par mots clés, mais elle présente trois grands inconvénients dans le cadre des environnements bureautiques : l'index de mots clés peut occuper jusqu'à 300% de la taille de la base de documents, le vocabulaire indexé est contrôlé, ou du moins très coûteux à mettre à jour, et la recherche par parties de mots n'est pas toujours possible (avec des performances acceptables).

- **Les fichiers Signatures** : cette méthode est fondée sur la construction, pour chaque fichier de texte (ou partie textuelle d'un document), d'une image compressée de son contenu. L'évaluation d'une requête est faite sur cette image, ce qui permet une sélection des documents plus rapide et, selon l'algorithme de construction de la signature, peut aussi faciliter l'évaluation des expressions logiques. Beaucoup d'algorithmes de construction de signatures sont proposés dans la littérature, les plus connus étant les Signatures par Mot (WS) et par Codage Superposé (CS). Un autre intérêt de ces méthodes est le peu de place nécessaire pour stocker les signatures, et les gains dans les temps de réponse. Il est à noter que ce temps reste, de toutes façon, moins performant que celui des fichiers inverses.

Dans le cas des parties textuelles de documents, peu de méthodes sont proposées dans le cadre des environnements bureautiques. Les méthodes de signatures sont les plus connues et, jusqu'à aujourd'hui les seules qui à notre connaissance aient répondu à

ses exigences. En particulier, les Signatures calculées selon la méthode CS semblent être les plus performantes. Néanmoins, elles ont des propriétés qui ne sont pas idéales au moment de faire une recherche par le contenu dans le sens défini pour cette opération dans les SRI.

Le balayage du texte plein sur tout le corpus est impraticable, en raison des longs temps d'évaluation des requêtes. Les méthodes d'inversion de fichiers, non plus, en raison de la nature de l'information bureautique, où le(s) domaine(s) couvert(s) par la base documentaire est(sont) ouvert(s). Par conséquent les connaissances sur le domaine d'application ne peuvent être complètement connues d'avance. Or, il est indispensable d'avoir des interfaces utilisant ces connaissances pour des usagers non spécialistes. Ceci marque aussi la grande différence avec les critères traditionnels des SRI.

Les méthodes de signatures semblent donc donner une perspective nouvelle pour la conception de méthodes d'accès aux parties textuelles de documents, et elles commencent aussi à être utilisées pour l'accès à d'autres éléments multimédia [Gib86] [GT 83], bien que ce ne soit pas encore très établi. Notre intérêt se centre donc, sur l'exploitation des qualités des Signatures pour les intégrer dans la gestion des documents dans le SIB, ainsi que pour améliorer d'autres fonctionnalités des SRI.

Une autre perspective à explorer pour effectuer un accès performant aux parties textuelles des documents est l'utilisation de machines dédiées, qui permettent l'évaluation rapide de requêtes. Nous traitons ce sujet ci-dessous.

3.3. Les Machines Spécialisés pour l'accès aux documents

La gestion de documents, et en général d'objets multimédia, pose des problèmes nouveaux par rapport à la gestion des objets

classiques. Ils sont gros, à structure complexe, et exigent donc des méthodes de stockage, d'accès, d'indexation, de mise à jour, qui soient appropriées, afin de préserver des performances acceptables. Certaines nouvelles technologies offrent une réponse à la plupart de ces problèmes : les grandes mémoires, les machines parallèles, les disques optiques, les machines dédiées ou les circuits à haut niveau d'intégration en sont quelques exemples.

L'utilisation de matériel spécialisé pour gérer des documents multimédia semble une voie qui permet de pallier les insuffisances des solutions de type logiciel. En particulier, un des plus graves problèmes dans la conception de méthodes d'accès est le temps d'évaluation des opérations. Nous faisons dans cette partie un survol de quelques-unes des machines dédiées au filtrage de texte. Ceci dans l'intention de donner un aperçu des technologies, qui sont souvent des prototypes de recherche.

Dans [Hol79] Hollaar présente de manière très claire les principaux problèmes associés au filtrage de grandes bases textuelles, ainsi que les différents types de technologies spécialisées pour les résoudre. Il expose d'abord le type de requêtes qu'il est souhaitable de pouvoir résoudre dans un SRI, et comment les méthodes traditionnelles s'avèrent inefficaces. Ensuite, il présente les principes de chaque type de machine et comment y implanter les méthodes d'accès nécessaires. Une première approche est l'utilisation de processeurs parallèles, tel qu'ils sont utilisés pour les données formatées. Néanmoins, le fait d'avoir, par exemple, des *jokers* dans les expressions de recherche, ou des termes de recherche (*patterns)* imbriqués dans le texte, pose des problèmes nouveaux au moment de l'évaluation, dans ce type de machines.

Une machine dédiée au filtrage de texte doit permettre [Hol79] : la recherche simultanée de l'occurrence d'un nombre non prédéterminé de termes, la spécification de caractères *joker*, et la possibilité de déterminer si une expression complexe a été satisfaite. En général, l'architecture de ces machines comprend :

- Un processeur **Hôte**, qui identifie une opération de Recherche sur des données textuelles. Il donne le contrôle au **Contrôleur de Recherche**.

- Le **Contrôleur de Recherche** qui est le responsable de la coordination des opérations entre le **Comparateur de termes** et l'**Evaluateur de Requêtes**. Il contrôle, éventuellement, les opérations d'entrée/sortie de données vers le Comparateur de termes.

Pour implanter le Comparateur de termes plusieurs techniques sont proposées :

- L'utilisation de comparateurs discrets ou des mémoires associatives,

- Une architecture cellulaire, dans laquelle chaque élément compare un seul caractère et transmet un signal à son voisin. En général les comparaisons possibles sont limitées, ce qui diminue la puissance dans la résolution de requêtes. Par exemple, dans le cas de reconnaissance de délimiteurs de mots ou de caractères "*joker*" [Has81]. Le nombre de cellules détermine aussi le maximum de termes dans une requête,

- L'utilisation d'un automate d'états finis universel, solution qui implique l'utilisation d'espace mémoire pour stocker la table des états, mais permet de reconnaître des expressions régulières de caractères. Une implantation en circuits VLSI de cette technique permettant l'interrogation de grosses bases de données est proposée dans [Has81].

L'Evaluateur de Requêtes détermine, dans tous les cas, si la requête est satisfaite d'après les résultats du comparateur de termes. Il transfère le résultat final à l'Hôte, si la requête est satisfaite. Une manière d'accélérer le temps de ce processus est l'utilisation en parallèle de plusieurs machines, combinée avec un partitionnement approprié de la base de données. Dans ce cas, il faut aussi prévoir un mécanisme de mise en forme des résultats.

Dans la suite de ce chapitre nous décrivons de manière très générale, un certain nombre de machines qui illustrent les diverses approches technologiques dans la construction de matériel spécialisé pour le filtrage de texte.

3.3.1. Les Machines de Traitement Parallèle - Le Processeur DAP

Plusieurs machines ou circuits dédiés au traitement parallèle de données sont proposées dans la littérature. Beaucoup d'entre elles sont générales, d'autres orientées vers des domaines précis : accélérateurs des opérations d'inférence, accélérateurs d'opérations de bases de données, distribution de tâches, etc. Nous sommes intéressés par celles qui offrent des traitements sur les données de type texte, permettant donc l'implantation de fonctions d'accès par le contenu aux documents.

Un exemple d'une telle machine est le processeur DAP (*Distributed Array Processor*), de ICL [CPW86], pour le traitement en parallèle des opérations de comparaison de termes de recherche ("*pattern matching*") dans les Systèmes Bibliographiques. DAP est intégré dans une machine hôte (ICL2900), dans un des modules mémoire (et non pas comme un processeur associé). Les données sont chargées une seule fois pour les traitements dans les deux machines. Le contrôle et les opérations d'entrée/sortie sont laissées à la charge de l'hôte, l'interface étant faite par une procédure. Le traitement est alterné, soit dans l'hôte, soit dans le DAP.

DAP est un processeur matriciel bidimensionnel, de 64*64 processeurs élémentaires, PEs. Chaque PE a sa propre mémoire (16k octets), son UAL (avec 3 registres d'un bit chacun) et des bus de données d'un bit qui le relient à ses voisins.

Il permet le traitement en parallèle de séries de bits. La longueur de la série est décidée par les applications en fonction de leurs besoins. Il permet les opérations logiques sur les files de la

matrice de PEs, ce qui facilite l'implantation des algorithmes de comparaison de termes.

Les données peuvent être vues comme une matrice de 4096 éléments, ou comme un vecteur unidimensionnel de 4096 bits. Chacun de ces éléments tient dans un seul PE, tout en respectant les séquences de position. Les opérations sur les tableaux ou les matrices peuvent être faites, en parallèle, dans une seule instruction. Des opérations utilisant des techniques de masques de bits peuvent être faites en utilisant des matrices booléennes.

La sélection d'un algorithme de comparaison de termes dépend de la manière dont sont stockés les documents dans la mémoire du processeur. La stratégie adoptée par [CPW86] a été le stockage d'un seul caractère de chaque document dans le même octet relatif dans chaque PE. Comme tous les 4096 peuvent être inspectés en parallèle, toutes les occurrence d'un terme de recherche dans un enregistrement peuvent être inspectées simultanément. Ceci limite, de fait, la taille d'un enregistrement à 4096 caractères maximum.

Le temps de comparaison est proportionnel à la longueur du terme de recherche, et il est indépendant de la taille du document. Néanmoins, pour D documents et M termes, il faut un temps de comparaison proportionnel à D*M. Il faut aussi, en raison des opérations d'entrée/sortie, aligner les données sur les enregistrements, ce qui fait que, pour les documents de moins de 4096 caractères, le reste de la mémoire sur le PE n'est pas utilisé. Des méthodes de compactage en paquets sont proposées.

La comparaison de termes est faite par des diffusions (*"broadcast"*) successives de "masques" d'évaluation. Pour chaque caractère à comparer un masque est généré et comparé dans tous les PEs. Si le résultat est *faux* dans tous les PEs, la comparaison échoue et le processus termine.

Les algorithmes proposés permettent les comparaisons avec des caractères *"joker"* au début ou à la fin des termes. Des masques sont utilisés pour la détection des délimiteurs de mots. En

construisant les masques appropriés et des opérations de décalage sur les PEs, il est possible d'évaluer des "*jokers*" à l'intérieur des termes, ceci pour des longueurs variables ou d'un seul caractère. Le traitement est néanmoins plus laborieux.

L'expérimentation faite porte sur une base de presque 923K caractères majuscules, correspondant à des listes de mots clés non répétés de 2472 titres et résumés de la Base de Données INSPEC. Par les raisons d'alignement de blocs, la taille de la base avait un "*overhead*" de 357%. Les documents avaient entre 50 et 1331 caractères, avec une moyenne de 384.8 caractères/document. Il faut bien noter que ces données ne correspondent pas à une base d'un système de recherche d'information, aucune pondération des mots clés n'étant faite.

Ces mesures sont faites en termes de (*nombre d'instructions * temps d'une instruction DAP*). Ceci donne 77 secondes si le terme de recherche n'a pas de "*jokers*" à l'intérieur, et 29.2% de plus s'il y en a. Des améliorations sont possibles si tout un ensemble de termes de recherche est évalué, et il est ordonné pour optimiser la génération et l'évaluation des masques. Dans le cas de 898 termes simples, le temps de réponse était de 156.7 secondes.

Des comparaisons ont été faites avec les algorithmes de filtrage utilisant les Automates d'Etats Finis proposés par Aho et Corasick, le DAP étant 6 fois plus rapide. Il faut bien noter que ces résultats ne tiennent pas compte du temps des opérations d'entrée/sortie, des algorithmes d'alignement, ni de l'évaluation d'expressions logiques de termes. Le temps d'évaluation est dépendant de l'ordre dans l'ensemble de requêtes et de leur complexité.

3.3.2. Les Machines Dédiées au Filtrage de Bases de Données

Un tout autre type de technologie dans le cadre des machines spécialisés pour le traitement de texte, est celui des *machines*

bases de données, dont nous trouvons des nombreux exemples. Parmi elles nous pouvons citer la machine DIRECT [DEW79], multiprocesseur pour le support de SGBD relationnels, VERSO [Sch85] [VER86], machine pour le filtrage au vol de bases de données relationnelles en forme Non NF1 développée à l'INRIA, GAMMA [DGG86], µSyc [CGM88] de BULL, CAFS (Content-Addressable File Store) de ICL [ICL85] et SCHUSS de BULL [GRT84].

Nous décrivons par la suite, à titre d'exemple, trois d'entre elles, CAFS, SCHUSS, et le coprocesseur µSyc. En raison des fonctionnalités qu'elles offrent, et pour des raisons de disponibilité, CAFS et SCHUSS ont été choisies pour l'implantation des méthodes d'accès aux documents ODA dans le cadre du projet ESPRIT DOEOIS.

3.3.2.1. La machine CAFS

L'architecture de CAFS [ICL85] est fondée sur la coopération entre un contrôleur de disques à haute vitesse et un filtre. Le contrôleur de disques coordonne les opérations de transfert de données entre le processeur hôte et le filtre. Le filtre a une architecture à cinq composants principaux fonctionnant en mode "pipe-line" :

- L'unité de formats logiques ("Logical Format Unit"), qui prend les données du transfert géré par le contrôleur, et les passe en entrée aux autres composants du filtre, en ajoutant les informations nécessaires pour reconnaître les champs pertinents à la requête. Ceci est fait en examinant et en identifiant chaque élément dans la chaîne d'entrée.

- Les canaux de clés ("Key Channels"), au nombre de 16, qui permettent l'entrée au filtre des critères de recherche. Ils traduisent ces critères en masques correspondant à chaque attribut du document. Chaque canal examine une condition atomique, sans faire d'évaluations logiques. Le nombre

maximum de canaux utilisés est néanmoins dépendant de la configuration du filtre pour une application donnée.

- L'unité d'évaluation de recherche ("Search Evaluation Unit") évalue les conditions logiques sur les résultats des filtrages individuels. Grâce à un autre processeur associé à cette unité, cette évaluation peut être pondérée.

- L'unité de recherche ("Retrieval Unit") stocke les enregistrements sélectionnés dans un autre processeur, avec une mémoire de 32K octets qui fait l'évaluation finale des résultats.

- Le processeur de recherche ("Retrieval Processor") produit, à partir des résultats stockés, des données en format transmissible vers le processeur hôte et arbitre ce transfert.

En fait, les canaux de clés et l'évaluateur de recherche sélectionnent les données, tandis que l'unité de recherche fait la projection de celles qui sont demandées dans la requête. Avec cette architecture, l'évaluation de requêtes est améliorée entre 10 et 100 fois par rapport à une solution purement logicielle, et le processeur hôte est déchargé de la plupart des tâches d'évaluation.

Dans les domaines d'application de CAFS nous trouvons par exemple, la gestion des index secondaires dans des grosses bases de données, et un système de recherche d'information, *Textmaster* [ICL85], en utilisant le matériel CAFS-ISP. Le modèle de documents de *Textmaster* est compatible avec le standard ODA.

Dans *Textmaster* CAFS-ISP est utilisée pour retrouver les documents par leur contenu sans avoir recours à l'inversion, ce qui permet de traiter des collections de documents dynamiques.

La structure d'un document peut être reconnue à trois niveaux :

- D'après la définition de la structure des documents ODA.

- Par les marqueurs d'édition, pour les documents non conformes à la norme, par exemple ceux acquis par lecteur optique.

- Le document est vu comme du texte plat, sans structure. Dans ce cas, la sélection du document n'est possible que par ses attributs externes (auteur, titre, etc), donc ceux qui sont dans le profil du document ODA.

Seules certaines des fonctionnalités de *Textmaster* ont été implantées en utilisant CAFS, le reste étant réalisé par logiciel. Les problèmes liés à l'utilisation d'une solution purement matérielle peuvent être résumés ainsi :

- Les documents sont stockés en double. Une première fois dans le format ODA, ce qui permet des bonnes performances au moment de l'affichage et l'édition, et une deuxième fois dans le format CAFS, avec le contenu normalisé via un dictionnaire et sans les spécificités de formattage ou présentation. En raison des contraintes d'évaluation de l'opérateur logique *and* par CAFS, cette opération booléenne a été implantée par logiciel. Le contraire aurait impliqué des restrictions sur la taille des documents traités.

- La gestion performante de grosses bases de données utilisant seulement CAFS n'est pas toujours possible, les temps de recherche étant éventuellement trop grands.

- La conception du logiciel et des fonctions nécessaires pour suppléer aux possibilités de CAFS.

Le format de stockage en CAFS a deux parties, une de taille fixe, où se trouvent les attributs de gestion du document (identificateur, classe, propriétaire, etc), et une partie variable qui contient le corps du document. Cette partie est formatée, chaque élément étant représenté par un triplet de la forme (type

d'élément, longueur, contenu). Le contenu est la suite de termes normalisés correspondant aux mots trouvés dans le document.

Exemple :

Pour le texte "La bibliothèque est complète"

la représentation CAFS est :

[(t, 12, BIBLIOTHEQUE) (t, 4, ETRE) (t, 8, COMPLETE)]

t indiquant le type texte et le verbe étant normalisé à l'infinitif.

Les bases de documents trop grandes posent des problèmes de performance dans l'évaluation de requêtes. Plusieurs alternatives sont analysées [ICL85], leur choix dépendant fortement de l'application. Les résultats mentionnent des recherches sur 10000 pages format A4 en 30 secondes, sans utiliser ni partitions ni inversion de la base. CAFS est toujours chargé du filtrage de la classe de documents concernés par une requête (ceci est déterminé par logiciel) et de l'évaluation de l'opérateur logique *or*. La normalisation, l'évaluation et la mise en forme de résultats est faite par logiciel.

3.3.2.2. La machine SCHUSS

SCHUSS est le prototype résultat de recherches de BULL [GRT84], dans le cadre de la conception d'un accélérateur de filtrage de données. C'est une machine qui prend les données directement du disque pour les traiter au vol. Du point de vue des possibilités de filtrage, SCHUSS peut traiter des données numériques ou textuelles à la vitesse de transfert. Ces données sont stockées séquentiellement.

Le principe essentiel dans la conception du filtre est le fait de considérer les données comme un langage, et les requêtes comme une grammaire [GRT84]. Le filtrage revient donc à la sélection des données qui appartiennent au langage généré par cette

grammaire. Les opérations à réaliser lors d'une requête de filtrage sont :

- La requête est compilée, dans la machine hôte, en une séquence d'instructions exécutables par le filtre, et qui constituent un automate de filtrage.

- L'automate est chargé dans la mémoire du filtre, qui l'exécute en traitant les données au vol.

- Le filtre envoie les résultats de filtrage vers la mémoire du processeur hôte.

Les principales caractéristiques de l'architecture de SCHUSS sont le parallélisme entre les opérations d'entrée/sortie et l'exécution de l'automate de filtrage, ainsi que les facilités de microprogrammation qui permettent d'étendre son utilisation. SCHUSS a son propre processeur, mémoire et disque de stockage de données. Une description détaillée de cette architecture est donnée en Annexe B. Avec l'utilisation de SCHUSS comme coprocesseur de filtrage, la machine hôte est complètement libérée des tâches d'accès au disque et de filtrage.

L'environnement logiciel de SCHUSS est écrit en langage micro-assembleur et il fournit un ensemble de micro-instructions de filtrage. Cette ensemble peut être modifié par les applications. L'environnement de microprogrammation a été développé en APL. Le système de filtrage de fichiers est composé à par deux modules, écrits en TURBO_PASCAL :

a. Module de Génération de Bases : il correspond à la gestion de l'espace sur le disque de SCHUSS. Une "base" (c'est à dire, un ensemble de données correspondant à une table relationnelle) doit être créée dans la machine hôte, dans un format simple (un nuplet par ligne). Ce module est chargé de le stocker dans un format reconnaissable par SCHUSS, dans son disque, pour être ensuite filtré.

b. Module d'Interrogation de Bases : ce module offre l'ensemble des opérations de filtrage sur les bases de données stockées sur le disque de SCHUSS. Il traite aussi l'ensemble des données de réponse aux requêtes. Certaines de ces opérations sont micro-programmées, par exemple l'interprétation des automates de comparaison ou les opérations relationnelles. Les micro-instructions peuvent être modifiées à partir de l'environnement APL.

A la base des opérations de filtrage se trouve un micro-programme qui représente un automate d'états finis. Au lieu de le représenter comme une matrice creuse, ce qui implique des grandes quantités de mémoire et un cycle de machine pour passer d'un état à l'autre, l'automate est représenté d'une manière très compacte, comme un arbre. Chaque élément de l'automate devient une instruction de filtrage micro-programmée dans SCHUSS. Pour évaluer des expressions de filtrage contenant des opérateurs logiques (des expressions en forme disjonctive), une table de vérité calculée et chargée avant le filtrage (tout comme l'automate) est ajoutée.

La conception de l'ensemble des opérations de filtrage a été guidée par le soucis d'optimisation de place et de temps d'exécution. Les quatre opérations implantées sont :

- Les états n'ayant qu'un état successeur sont compilés en utilisant l'instruction *Linéaire* :

  <LINEAR> , N, M, Car_1, Car_2, ... , Car_N

  Cette instruction permet de comparer les prochains N caractères en entrée avec Car_1, Car_2, etc. Si les N caractères sont reconnus, le bit M de la table de vérité est activé. Sinon, l'exécution de l'automate se termine par un échec.

- Les états n'ayant qu'un petit nombre d'états successeurs (4 au maximum) sont compilés en utilisant l'instruction *Séquentiel* :

<SEQUENTIAL> , N, M < 0, Car_1, M = 1, Adr_1, ..., Car_N, M = N, Adr_N

Cette instruction permet de comparer le prochain caractère en entrée avec N caractères successeurs possibles. Si le caractère est égal à Car_i, le bit M = i est activé et l'adresse de la prochaine instruction est Adr_i. Les caractères sont comparés dans l'ordre, si aucun ne produit le branchement l'exécution de l'automate termine en échec.

- Les états qui ont beaucoup d'états successeurs (plus de 4) sont compilés utilisant l'instruction *Indexé* :

<INDEXED> , M = 1, Adr_1, ..., M = N, Adr_N

Cette instruction permet de comparer le prochain caractère en entrée avec tous les caractères possibles dans le vocabulaire. Il sera égal à celui d'une position i, et donc le bit M = i est validé et l'adresse de la prochaine instruction est Adr_i.

- Pour trouver un compromis entre les instructions *séquentiel* et *indexé*, l'instruction *dichotomique* permet de réaliser plusieurs test avec un seul caractère en entrée.

<DICHOTOMIC> , Car , $Adr_<$, $Adr_>$, $Adr_=$, M

Si le caractère en entrée est inférieur à Car, l'adresse de la prochaine instruction est $Adr_<$. S'il est supérieur, c'est $Adr_>$, s'il est égal c'est $Adr_=$ et le bit M est validé.

Ce jeu d'instructions de filtrage permet de faire, avec les meilleurs rapports place/temps d'exécution, la comparaison et la sélection des données obéissant aux critères de recherche. D'après les mesures effectuées, SCHUSS filtre "au vol" (c'est à dire, à la vitesse de transmission du disque) jusqu'à environ 3 méga-octets par seconde). Quelques exemples et plus de détails se trouvent dans [GRT84] [BUL86] et dans l'Annexe B.

3.3.2.3. Le Coprocesseur µSyC

Après la conception de la machine SCHUSS, BULL a conçu sa "version" VLSI, le coprocesseur µSyC [CGM88] [CG 88], *"Microprogrammable Symbolic Coprocessor"*. µSyc est un circuit micro-programmable, destiné à accélérer les opérations élémentaires dans les bases de données, aussi bien au filtrage de données qu'aux traitements symboliques. Les micro-programmes destinés au filtrage de données sont proches de ceux existants sur SCHUSS.

Contrairement à la stratégie de conception de SCHUSS, décrite auparavant, dont l'intérêt portait sur le rapprochement de l'accélérateur vers les disques, ce circuit est conçu comme un accélérateur rapproché de la mémoire. Il est destiné à être placé dans chaque noeud élémentaire de la machine DDC, de BULL, d'architecture modulaire, multi-processus à mémoire distribuée. DDC est une machine destinée aux traitements sur des bases de données relationnelles et des bases de données déductives. Une description détaillée de son architecture se trouve dans [CGM88] [CG 88].

Chaque noeud de la machine DDC a des possibilités de traitement, de communication et une mémoire (PCM). Le traitement de données est réalisé par un processeur MC68020 et le coprocesseur spécialisé µSyC, qui lui apporte un facteur d'accélération de l'ordre de 5. Les mémoires locales sont dimensionnées en accord avec la taille de la base de données, en tenant compte de sa répartition sur l'ensemble de noeuds. Les algorithmes utilisés sont fortement dépendants des structures de données, afin d'optimiser le rapport place mémoire/vitesse d'exécution. La stratégie choisie est celle d'automates d'états finis compactés [CGM88].

Les trois primitives principales que le DDC confie à µSyC sont *search*, *select* et *new_add*. La primitive *search* permet de vérifier si un nuplet donné est présent ou non dans une relation. *select* fait la même comparaison, mais son résultat est l'ensemble de nuplets qui vérifie le critère de sélection. *new_add* ajoute un

nouveau nuplets dans une relation, tout en évitant les duplications.

Avec l'opération *search* il est possible de chercher une chaîne de caractères dans la base. Pour l'exécuter dans un temps minimum, les relations sont stockées en mémoire sous forme d'automates d'états finis. Pour éviter la représentation des automates comme des matrices creuses, une représentation compacte, dont le format dépend du nombre de transitions de chaque état est utilisé. Cette représentation, similaire à celle utilisée dans SCHUSS, est directement interprétable par le jeu de micro-instructions. Les chaînes de caractères sont accédées en "mots" de 32 bits, et l'on dispose d'opérateurs spécialisés pour en extraire, comparer et insérer des octets. Le comparateur peut fonctionner en parallèle avec les deux extracteurs et l'"injecteur" d'octets permet de préparer les résultats avant de les écrire en mémoire.

µSyC est vu par le processeur hôte, le 68020, comme un périphérique de type DMA. Il communique avec la mémoire via un bus d'adresses et un bus de données de 32 bits chacun. Pour l'interface avec la mémoire de micro-programmes il comporte un bus de micro-adresses de 12 bits et un bus de micro-instructions de 69 bits [CGM88]. Le circuit est composé par un chemin de données, un séquenceur de micro-instructions, un contrôleur de la mémoire, une unité arithmétique et logique et des registres pour la communication avec le processeur hôte. La mémoire de micro-programmes se trouve à l'extérieur du circuit en raison de sa nature de prototype, et pour permettre de faire facilement des modifications. L'architecture interne de µSyC est décrite plus en détail en [CGM88] [CG 88].

3.4. Conclusions

Dans ce chapitre nous avons présenté de manière générale les approches prises dans les domaines de base de données pour

l'accès par le contenu de documents. Les principales différences entre la gestion des documents de bureau et celle des données classiques sont la taille, la complexité de leur structure et leur dynamicité.

La plupart des études que nous avons présentées traitent le problème de la gestion des documents du point de vue de leur intégration dans un modèle de données. La tendance générale est de les manipuler de manière homogène avec le reste des objets du modèle, tout en préservant des sémantiques bien précises.

Du point de vue des opérations d'accès à ces objets, complexes, structurés, multimédia, et dynamiques, très peu de possibilités sont proposées. Il est clair que pratiquement tous offrent l'accès par les attributs externes du document, par exemple les auteurs, date, titre, etc, ce qui constitue, d'une certaine manière, un ensemble de données classiques associées au contenu du document. Pour l'accès au corps proprement dit, donc la structure logique et le contenu même du document, nous avons trouvé trois stratégies principales : le balayage de texte (filtrage d'après une condition de sélection), l'indexation par inversion de mots clés et les méthodes de compression en utilisant les signatures.

Les deux premières s'avèrent insuffisantes du point de vue d'un environnement bureautique, le balayage d'un grand ensemble de documents étant impraticable par les coûts importants en temps d'évaluation, et l'inversion par mots clés à cause des restrictions que cette méthode impose sur le vocabulaire, sur la taille de l'index et sur la dynamicité de l'évolution du corpus documentaire. Les méthodes de signatures semblent donner un bon compromis entre la taille de la place mémoire additionnelle pour les supporter et le temps d'accès nécessaire pour l'évaluation de requêtes. Elles offrent des bonnes performances du point de vue des mises à jour. L'inconvénient de l'utilisation des méthodes de signatures est la possibilité de génération de bruit dans les réponses. Ce problème, dû aux méthodes de hachage, peut être pallié par une résolution de requêtes combinée avec le filtrage de l'ensemble des réponses présélectionnées.

Le filtrage de texte est toujours limité par les temps des opérations d'entrée/sortie. Nous avons donc exploré les possibilités offertes dans le domaine du matériel spécialisé. Nous avons présenté les machines que nous avons considéré comme les plus représentatives dans la technologie disponible des accélérateurs de filtrage de données, et nous avons montré leur application au filtrage de documents.

Dans le chapitre suivant nous présentons notre approche pour la gestion et l'accès par le contenu aux documents manipulés par le SIB. Nous combinons les solutions logicielles, en particulier celle des méthodes de signatures, et les solutions matérielles, avec la machine SCHUSS, qui était à notre disposition. Dans les solutions proposées nous étudions l'intégration de fonctionnalités d'accès au contenu d'un corpus documentaire tel que cela est offert par les SRI.

4 . L'ACCES AU CONTENU DES DOCUMENTS DANS LE SIB

4.1. Introduction

L'information circulant dans un bureau se caractérise par la grande variété de domaines qu'elle peut couvrir. Nous trouvons aussi bien des applications qui traitent des domaines totalement ouverts, telles que le courrier électronique, que des applications spécifiques et à domaine fermé, du type gestion ou production de factures et de contrats. Contrairement aux applications manipulées sur des systèmes traditionnels, les applications bureautiques sont de nature dynamique et en général peu structurées [ESP86a] [ESP86b].

L'activité bureautique est constituée de tâches spécifiques au fonctionnement d'une organisation [Tsi85] [Ped88], et de l'information qui y circule. Un des aspects fondamentaux de ces activités est la production, le stockage, la récupération et la transmission de documents de natures diverses, qui constituent la matérialisation des informations du bureau.

Comme nous l'avons présenté auparavant, plusieurs approches sont proposées pour la manipulation des documents. Dans les systèmes de Bases de Données l'effort porte sur l'intégration des documents dans des modèles de données généraux. Dans les systèmes de Recherche d'Information l'accent est mis sur la construction d'outils spécifiques visant à modéliser la sémantique de leur contenu.

Notre intérêt porte sur la définition des outils de base pour l'accès au contenu des documents dans un serveur d'information bureautique générique, que nous avons appelé le **SIB**. Cette définition comprend deux aspects :

- L'intégration des documents au niveau du modèle de données, comme une entité équivalente aux autres entités, tout en preservant leur sémantique. Cette intégration implique la définition homogène des attributs et du contenu des documents, ainsi que, au niveau interne, des structures de données appropiées.

- L'intégration au niveau fonctionnel, qui est faite à trois niveaux :

  - Du point de vue fonctionnel, les opérateurs sur les documents sont tous les opérateurs définies sur les autres entités plus certains permetant d'exprimer une sémantique particulière.

  - L'intégration au niveau du langage de manipulation de données, qui doit permetre l'expression de ces opérateurs.

  - L'intégration au niveau de la réalisation même des opérateurs par la définition des méthodes d'accès et de stockage associées.

L'intégration est faite en tenant compte des besoins des applications qui peuvent être construites autour du SIB. Ces besoins incluent les cas simples, où l'utilisateur cherche simplement la présence d'une chaîne de caractères dans un document, et les cas plus complexes où il est intéressé par la recherche des documents qui traitent d'un sujet en particulier. Ce dernier cas rejoint les applications de recherche documentaire. Notre approche consiste en la définition des opérations de recherche par le contenu, en offrant la souplesse nécessaire pour chaque cas, et l'étude des méthodes d'accès et de stockage appropriées.

Afin de montrer comment sont concrétisés les concepts que nous développons, notre présentation fait référence, à titre d'exemple, à un serveur en particulier, l'**OIS**, réalisé dans le cadre du projet ESPRIT-DOEOIS. L'OIS a été conçu pour la gestion de l'information dans un environnement de bureau. Nous utilisons dans les exemples une syntaxe voisine de celle de son modèle de données, le Fact Model (cf. chapitre 1, Annexe A).

4.2. La Gestion de Documents dans le SIB

Le modèle de données du SIB permet la définition de schémas conceptuels des applications de bureau. Du point de vue externe au serveur, les documents sont traités comme une entité quelconque. Par exemple, le traitement de l'entité "contrat" est homogène à celui de l'entité "facture". La manipulation de documents, en raison de leur taille et de leur structure complexe, exige néanmoins des considérations spéciales à l'intérieur du serveur. Nous regroupons cet ensemble de tâches et de considérations particulières au traitement des documents dans un *Gestionnaire de documents.*

Les documents traités par le SIB sont composés de deux parties : les **attributs externes**, qui constituent leurs "propriétés", (par exemple la date de création, le(s) auteur(s), le titre, le résumé, etc), et le **contenu**. Le contenu est structuré, de façon arborescente, dans la **structure logique** du document.

La plupart des modèles de documents considèrent ces deux éléments, en particulier, le modèle défini comme standard pour l'échange de documents de bureau, ODA ("*Office Document Architecture*"). Le formalisme de codification de ces documents est ODIF [ECM85]. Sans perte de généralité, nous adoptons dans le SIB la définition ODA des attributs externes et de la structure logique.

Les fonctionnalités du Gestionnaire de documents du SIB sont :

- L'intégration homogène des documents, en étant une entité à structure complexe, aux autres composants du SIB.

- Préserver la ***visibilité*** des documents dans le SIB, c'est-à-dire, la manière dont ils sont perçus, au niveau de l'interface, par les applications. Dans notre cas, elle est définie comme étant le document complet. Donc, toutes les opérations offertes sont définies au niveau d'un document ou d'un ensemble de documents. Ceci implique, entre autres, que les opérations sur des parties d'un document ne sont pas définies dans le SIB (par exemple, la récupération d'un chapitre d'un document).

- Offrir les services de stockage et d'accès aux documents. Les applications qui veulent traiter ces documents, ou leur contenu, (éditeurs, serveurs d'impression, etc.) sont externes au SIB et se trouvent au niveau des postes de travail.

4.2.1. Les Documents dans le SIB

Par souci de compatibilité entre le SIB et d'autres systèmes ou serveurs, les documents manipulés par le SIB sont conformes à la norme ODA, tout en préservant la sémantique des données propres au modèle du serveur.

L'unité minimale dans la structure logique d'un document ODA est appelée une **portion de contenu** ("*Content Portion*"), et c'est seulement dans ces éléments que l'on trouve le contenu textuel des documents. Les portions de contenu se situent toujours au niveau des feuilles de l'arborescence. Elles peuvent être de type multimédia (texte, image, graphique, etc.) selon le type de données qu'elles contiennent. Tous les éléments de la structure logique d'un document ODA ont un identificateur qui permet de situer leur niveau et leur ordre dans l'arborescence (sous la forme d'un chemin d'accès).

La structure des documents traités par le SIB est la suivante :

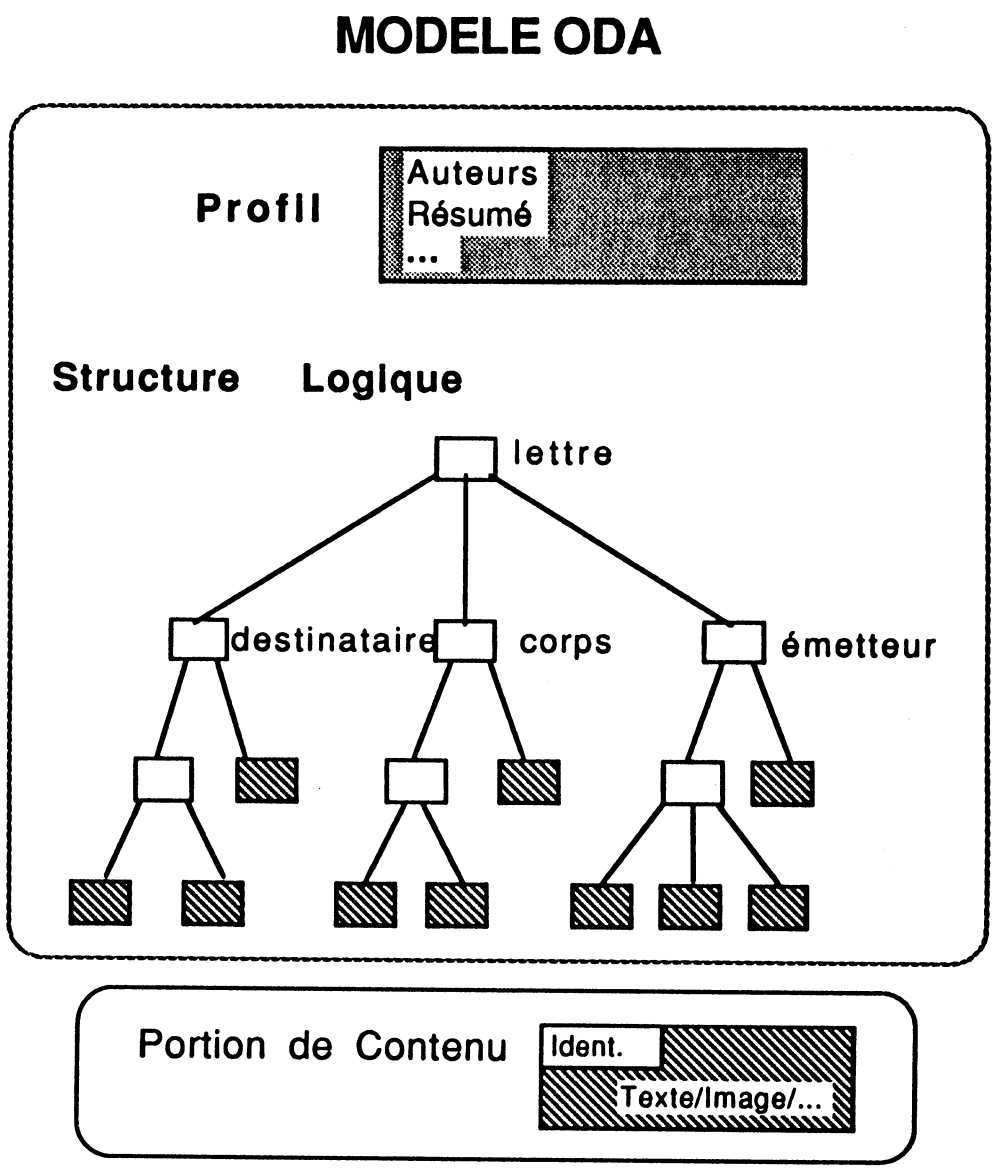


figure 4.1

La norme ODA établit différents **niveaux de conformité**, selon les structures présentes et la manière dont elles sont codées. En raison des traitements que nous voulons offrir sur les documents (accès aux attributs et au contenu des documents avec un haut niveau de sémantique), nous nous plaçons aux niveaux de

conformité à la norme ODA des documents contenant une structure logique (format PODA/H87 [BJM88]). Ceci signifie que tout document dans le SIB doit avoir une structure logique.

Le premier problème qui se pose est l'intégration d'un tel modèle de documents dans le modèle de données du SIB. Par exemple, la solution adoptée dans l'OIS a été de définir un type de base ODA (cf. Annexe A), avec une sémantique particulière, qui respecte les conditions d'atomicité et de visibilité définies au niveau de l'interface du serveur. Les objets du type de base ODA sont codés en ODIF, et ils sont des objets multimédia possédant une structure logique.

Dans FM, les composants du document, à savoir ses attributs externes et son contenu, sont considérés comme des propriétés de l'entité **DOCUMENT**. Les propriétés les plus courantes dans les applications de bureau (par exemple "auteur du document", "date de création", "résumé", etc.) sont prédéfinies. Par exemple le **contenu**, prédéfini comme du type de base ODA [Mar87].

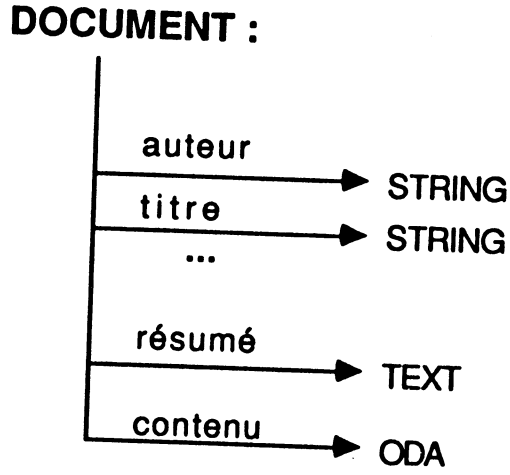


figure 4.2

Pour intégrer à FM la structure logique des documents un type d'entité composé a été défini [LG 89]. Elle permet d'exprimer la hiérarchisation des composants du document.

Exemple :

> Un livre peut être défini comme un document dont le contenu est composé par un titre et des chapitres, composés d'un titre et de sections, elles mêmes composées à leur tour d'un titre et de paragraphes.

4.2.2. Les Services du Gestionnaire de Documents

Pour manipuler les documents dans le SIB nous avons identifié quatre aspects principaux à traiter [BJM88] [Mar87] :

4.2.2.1. L'INTERNALISATION des Documents.

Les documents qui arrivent au SIB sont codés en ODIF. Il faut les décoder, en extraire l'information nécessaire pour établir toutes les relations qui leur sont associées dans le modèle de données du SIB, et finalement les stocker. Le stockage des documents doit préserver l'atomicité des objets, de manière à pouvoir les restituer aux applications, et aussi à pouvoir effectuer les opérations de sélection sur les valeurs de leurs attributs ou sur leur contenu.

Ce service est sollicité par des énoncés de la forme :

INSERT <doc>

WITH <doc>.content = <identificateur_objet_ODA>

4.2.2.2. L'INDEXATION des Documents.

Lors de la consultation sur les attributs ou sur le contenu des documents il faut avoir de bonnes performances, en raison des exigences de l'environnement de bureau. Si la base de documents

est volumineuse, même si les documents sont de taille moyenne, une indexation est indispensable. L'indexation des attributs externes du document, définis sur des types de données "classiques", est effectuée par des méthodes traditionnelles des bases de données. Pour le contenu du document, où l'on trouve des données multimédia, et une structure complexe, d'autres techniques s'imposent.

Ces techniques doivent offrir les bases suffisantes pour préserver la sémantique du contenu du document, et des méthodes d'accès appropriées. Elles sont définies en fonction du type d'opérations de recherche que nous voulons effectuer : pour la recherche textuelle, un **index du contenu textuel** des documents est nécessaire, tandis que pour la recherche sémantique il faut définir une technique d'accès au modèle sémantique du contenu. Un aspect fondamental est la taille des index, car le serveur constitue le support de multiples applications dans des domaines généralement ouverts et différents.

4.2.2.3. L'ACCES aux Documents

Le service d'accès aux documents fournit les outils pour interroger la base de documents d'une application. Les opérations d'interrogation sont spécifiées par des énoncés de type **SELECT**. La sélection peut porter sur des attributs ou sur le contenu du document. Pour l'accès au contenu du document ils est possible de spécifier des thèmes de recherche en tant qu'expressions de sélection.

Exemple :

```
SELECT  e.date
FROM    lettre e
WHERE   e.auteur = "Dupont"
```

Cette requête définit une sélection sur un attribut externe des lettres, leur date, à partir de la valeur d'un autre attribut externe, leur auteur.

```
SELECT   rapport.titre
WHERE    rapport.auteur = "Dupont" OR
         rapport.content TALKS ABOUT "Bases de Données"
```

Cette requête définit une sélection sur un attribut externe des rapports (leur titre), à partir de la valeur d'un autre attribut externe (leur auteur) et de leur contenu (le thème "Base de Données").

4.2.2.4. L'EXTERNALISATION des Documents.

Ce service permet de rendre aux applications, ou aux autres serveurs externes au SIB, un document stocké dans la base. Par définition du serveur, ce document doit être conforme au standard ODA. C'est le service complémentaire à celui de l'internalisation de documents.

Les opérations d'accès par le contenu aux documents, ainsi que les méthodes d'indexation et d'accès que leur sont associées, font l'objet principal de ce travail. Nous les présentons en détail dans la suite de ce chapitre.

4.3. La Recherche par le Contenu des Documents

En général, une opération de recherche par le contenu d'un document vise à sélectionner dans la base documentaire ceux qui traitent d'un sujet donné. C'est-ce que nous appelons un **thème de recherche**. Ce thème peut s'exprimer en termes d'une chaîne

de caractères spécifique (dans le cas du courrier électronique, par exemple) ou d'un sujet connu par l'application à laquelle appartient le document (dans le cas des rapports de recherche).

Nous avons défini (cf. chapitre 1) deux opérations de recherche qui correspondent à ces différents types de sélection, afin de donner aux applications le choix entre une méthode simple et une méthode plus sophistiquée de consulter les documents.

La première, la **RECHERCHE TEXTUELLE,** permet une comparaison entre une chaîne de caractères qui constitue l'expression de recherche, et le *contenu textuel* des documents. Cette méthode est donc fondée sur des techniques de mise en correspondance de chaînes de caractères.

La deuxième, la **RECHERCHE SÉMANTIQUE,** tient compte du contenu sémantique des documents. L'expression de recherche exprime donc un **thème** connu par l'application. La sélection de documents est basée sur l'évaluation d'une fonction de correspondance entre le thème de recherche et un modèle sémantique du contenu des documents.

Ces deux approches sont complémentaires, dans la mesure où dans la grand variété d'applications bureautiques certaines peuvent se contenter d'une sélection simple, rapide et dynamique, tandis que d'autres ont besoin d'un support plus élaboré.

D'une certaine manière, nous pouvons dire que ces deux approches sont dans le même rapport que la syntaxe et la sémantique dans l'analyse de la langue : la **RECHERCHE TEXTUELLE** est fondée sur une expression linguistique particulière d'un concept (la chaîne de caractères), tandis que la **RECHERCHE SÉMANTIQUE** s'appuie sur une expression sémantique du concept lui-même. C'est donc aux applications de choisir le type de recherche qui leur convient le mieux.

L'introduction de ces opérations dans le SIB implique un élargissement du langage de manipulation de données, afin d'exprimer le type de recherche par le contenu à effectuer, le

thème défini comme critère de sélection, et l'ensemble de données sur lequel porte l'évaluation. Une opération de recherche par la contenu est exprimée par un opérateur de recherche (ODR) dans la clause **WHERE** de l'énoncé **SELECT**, de la manière suivante :

SELECT <doc>.<attribut>
FROM <doc>
WHERE <doc>.<ElémentLogique> **<ODR>** <thème_de_recherche>

L'attribut sélectionné, <doc>.<attribut>, peut être un attribut quelconque défini sur le document <doc>, ou d'autres entités associées à <doc>. Leur projection est réalisée ensuite par les méthodes classiques. La clause **FROM**, sert à spécifier le contexte d'évaluation des attributs. Le niveau de la structure logique du document sur lequel s'évalue le thème de recherche est indiqué par <doc>.<ElémentLogique> de la clause **WHERE**.

Exemple :

Pour le document montré dans la figure 4.1, l'énoncé de sélection :

SELECT l.auteurs, l.content
FROM lettre l
WHERE l.corps **<ODR>** "délai de livraison"

indique que l'on veut, dans l'ensemble de lettres, l'auteur et le contenu de celles qui parlent de "délai de livraison" dans le corps. Le contenu des autres éléments logiques des lettres, à savoir, destinataire et émetteur, ne sont pas concernés par la requête.

Dans le processus d'évaluation des clauses de sélection, nous nous intéressons ***seulement*** à l'évaluation de l'opérateur **<ODR>** dans la clause **WHERE**. C'est à ce niveau que la recherche par le contenu est demandée.

Notre requête est ainsi constituée par trois éléments :

- Le type de recherche à effectuer, spécifié par l'opérateur **<ODR>** (recherche textuelle ou recherche sémantique).

- Le critère de sélection, spécifié par le <thème_de_recherche>.

- L'espace de recherche, défini par <doc>.<ElémentLogique>, que nous appelons par la suite la **Unité Logique Designée (ULD)**.

Notre réponse est ainsi constituée par l'ensemble des identificateurs des ULD qui vérifient le thème de recherche. Cette réponse est communiquée aux autres modules du SIB afin de constituer la réponse finale à l'énoncé **SELECT** correspondant.

Aussi bien pour la recherche textuelle que pour la recherche sémantique, le point critique dans la définition de méthodes d'accès associées, est le rapport temps d'accès / place nécessaire. Le problème d'espace de stockage est particulièrement important dans le cas de la recherche sémantique, pour laquelle nous nous inspirons des concepts du prototype IOTA (cf. chapitre 2). Notre approche est fondée sur l'adaptation des techniques de signatures et de filtrage adaptées à l'environnement bureautique, ainsi que sur l'utilisation de matériel spécialisé. L'intérêt du recours à ces techniques a été présenté dans le chapitre précédent.

4.3.1. La Recherche Textuelle

La **RECHERCHE TEXTUELLE** permet un filtrage direct sur le contenu textuel des documents du SIB. Elle est définie comme la comparaison de thèmes de recherche exprimés par des expressions de chaînes de caractères, et le contenu des parties textuelles des documents. Ainsi, pour chaque document, elle porte sur un ensemble de **portions de contenu.** Cet ensemble comprend toutes les portions de contenu d'un objet ODA, ou

seulement celles qui dépendent des éléments de sa structure logique spécifies par l'ULD (par exemple, un chapitre donné).

Pour exprimer une recherche textuelle nous avons défini l'opérateur de recherche **TALKS ABOUT** dans la clause **WHERE**. Il permet la sélection des documents dont le contenu correspond à une condition donnée par une **expression de filtrage**.

La syntaxe générale d'une expression de filtrage est la suivante :

<expression_de_filtrage> ::= (<expr_and> { OR <expr_and> }*)
<expr_and> ::= <terme> { AND <terme> }*
<terme> ::= { NOT } <chaîne>
<chaîne> ::= {"*"}<sous-chaîne>{<joker><sous-chaîne> {"!"{[n]}} }* {"*"}
<joker> ::= "*", "$", "!"{[n]}

Dans une expression de filtrage nous pouvons donc spécifier des combinaisons logiques de chaînes de caractères via les opérateurs **AND**, **OR** et **NOT**. Ces expressions logiques sont en forme disjonctive, la précédence des opérateurs étant implicite. Nous pouvons aussi spécifier partiellement une chaîne, en utilisant des opérateurs "joker". Trois types de joker sont définis :

- l'opérateur **joker** "*", qui peut être placé n'importe où dans la chaîne, au début pour avoir des préfixes variables, à la fin pour des suffixes variables ou à l'intérieur même de la chaîne. La portée du "*" est défini comme étant limitée à une portion de contenu.

  Exemple :

```
SELECT  c.adresse
FROM    chercheur c, rapport r
WHERE   r.auteur = c.nom AND
    r.content TALKS ABOUT ("syst*infor*" AND "*expert*")
```

Dans cet énoncé, nous voulons les adresses des chercheurs qui ont écrit des rapports qui parlent de "systèmes d'information pour experts", "systèmes experts d'information", "experts en systèmes d'information", etc.

La limitation de la portée d'évaluation de "*" à une portion de contenu, évite la sélection de documents qui contiennent, par exemple, "syst*" dans le premier paragraphe et "*infor*" dans le dernier, ce qui ne correspond pas à un texte pertinent par rapport à la requête. Donc, si "syst*" et "*infor*" ne sont pas dans la même portion de contenu d'un rapport, il n'est pas sélectionné.

- L'opérateur **joker "$"**, qui peut être placé entre deux sous-chaînes. La portée du "$" est défini comme étant limitée à une phrase dans une portion de contenu. Une phrase est une unité syntaxique délimitée par un ensemble de délimiteurs classiques de la langue (".", "!", "?", ...). Le joker "$" ne peut pas être placé qu'à l'intérieur d'une chaîne, "*" étant suffisant pour décrire les suffixes et préfixes variables.

Exemple :

```
SELECT  c.adresse
FROM    chercheur c, rapport r
WHERE   r.auteur = c.nom  AND
        r.content TALKS ABOUT ("*syst$infor*")
```

La réponse à cet énoncé, est constituée par les adresses des chercheurs qui ont écrit des rapports qui parlent de "systèmes informatisés", "systèmes et informatique", "systèmes d'information" etc.. Les rapports dont le contenu est, par exemple : "Etude de la *systématisation* des processus industriels. L'*information* de l'entreprise est ...", ne vérifient pas le critère de sélection.

- L'opérateur **joker** "!"[n], qui peut aussi être placé entre deux sous-chaînes, ou à la fin de la chaîne de caractères. La portée du "!" est définie comme étant limitée à maximum n caractères dans un mot de la portion de contenu. Si n n'est pas spécifié, la valeur n = 1 est prise par défaut. Un mot est une unité syntaxique délimitée par des séparateurs (" ", ".", ":", ";", etc). Le joker "!"[n] ne peut pas être placé au début d'une chaîne, le "*" étant considéré suffisant pour spécifier des préfixes variables.

  Exemple :

  SELECT revue.nom, revue.date
  WHERE revue.content **TALKS ABOUT** ("impressionnis!3 ")

  permet de retrouver les noms et les dates des revues qui parlent d'"impressionnis*me*" ou d'"impressionnis*tes*"

La syntaxe générale d'un énoncé de recherche textuelle est :

SELECT <doc>.<attribut>
WHERE<doc>.ULD **TALKS ABOUT** <expression_de_filtrage>

<doc> étant un document, ou une spécialisation de documents.

Notre requête est donc formée par la partie **TALKS ABOUT** de la clause **WHERE**. La réponse est l'ensemble des identificateurs (id_doc, id_ULD), des ULD des documents <doc> qui vérifient la <expression_de_filtrage>.

4.3.2. La Recherche Sémantique

La **RECHERCHE SÉMANTIQUE** permet la sélection de documents via une représentation sémantique du contenu des documents du SIB. Elle porte donc sur des thèmes de recherche exprimés par des descripteurs du domaine couvert par l'ensemble de documents. A

travers ce type de recherche nous offrons **au niveau du serveur** des services de stockage et d'accès au contenu sémantique des documents, qui sont à la base de l'architecture des SRI qui pourraient être construits au niveau des applications.

L'accès au contenu sémantique d'un document est réalisé via une **relation d'indexation.** Elle contient un ensemble de triplets composés de références aux composants logiques des documents, des termes d'indexation et d'une pondération de la pertinence relative entre ces termes et ces composants logiques. Les termes d'indexation peuvent être des mots simples ou des mots composés. Ceci est décidé par le module d'indexation au niveau des applications.

Par exemple, dans IOTA les termes d'indexation sont de groupes conceptuels, extraits par des analyses linguistiques et par un processus de normalisation (qui correspond à une *lemmatisation* dans le contexte linguistique) qui vise à représenter les mots sous une forme canonique, indépendante de leurs déclinaisons dans la langue (pluriels, conjugaisons,...) (cf chapitre 2).

Dans le cas du SIB cette relation d'indexation est *dynamique*, dans la mesure où l'environnement (c'est à dire, le **domaine**) dans lequel elle est établie évolue. Cet aspect, propre au bureau, est justement l'opposé de ce qui est classiquement supposé dans les systèmes de documentation et les SRI, qui considèrent des corpus où *l'ensemble de termes d'indexation est relativement stable et connu.* Les mises à jour de cet ensemble peuvent entraîner, dans certains cas, une réévaluation de la relation d'indexation ou même la ré-indexation du corpus.

L'indexation peut être établie à plusieurs niveaux dans la structure logique d'un document. Dans IOTA une **unité textuelle** est définie comme un sous-arbre correspondant à un élément donné de la structure logique du document. Le processus d'indexation établit une relation entre un terme et une pondération, avec une unité textuelle du document. Par exemple, l'indexation d'un livre peut indiquer que le premier chapitre traite du thème "structures de données" avec un certain poids tandis que le troisième traite celui des "méthodes

d'ordonnancement" avec un certain poids. Dans les deux cas, le poids exprime l'importance du thème dans l'unité textuelle considérée.

L'unité textuelle la plus générale (de plus haut niveau dans la hiérarchie) prise en compte dans le processus d'indexation est appelé **l'unité maximale d'indexation**, et la plus simple (de niveau le plus profond dans la structure logique) est **l'unité minimale d'indexation**. Une méthode d'indexation dynamique, qui commence par les unités minimales, assure la remontée des termes d'indexation et le calcul des pondérations sémantiques associées vers les unités supérieures [Ker84].

Une opération de recherche sémantique vise donc à interroger cette représentation du contenu d'un ensemble de documents. Les opérations définies sur la **RI (Relation d'Indexation)** sont suffisamment générales pour permettre, par exemple, la définition de critères différents de pondération ou d'évaluation propres aux diverses applications documentaires construites sur le SIB.

Il est clair qu'au contraire de la recherche textuelle, qui peut être effectuée sur toute la base de documents, la recherche sémantique ne sera effectuée que sur les documents pour lesquels la RI existe. Lors de la déclaration d'un type de documents il faut donc indiquer qu'ils peuvent avoir une RI associée.

Par exemple :

DEFINE LIVRE : **DOCUMENT indexed;**

La sémantique de cette déclaration est la suivante :

- Seules les données de type document ou ses spécialisations peuvent être indexées.

- Les documents déclarés indexés (par exemple, des livres), ainsi que leurs unités textuelles (identifiées par l'élément

de la structure logique correspondant) peuvent avoir une RI associée.

La RI d'une unité textuelle est constitué par les termes d'indexation pondérés que lui sont associés, et ceux qui se trouvent dans le sous-arbre dont cette unité textuelle est la racine.

Exemple : Dans la figure 4.3, la RI associée à ARTICLE.CORPS correspond aux termes associés au nœud CORPS (t_3 avec le poids p_3) et les termes associés à PARTIE I et à PARTIE II, ainsi que ceux associés à leur fils.

La RI des unités textuelles est dénotée comme tout autre attribut associé aux éléments logiques des documents : <document>.<élément_logique>.RI. Cette expression désigne les termes d'indexation du sous-arbre de la structure logique correspondant, ainsi que leurs pondérations.

- Une RI établit une relation entre les documents (ou une unité textuelle de document), des terme d'indexation et leurs pondérations. La définition d'une RI s'exprime donc par une relation ternaire :

  [<réf> : référence_élém_logique, terme: string, poids: real]

  Le choix des termes, ainsi que leur pondération sont évidemment établis par l'application. Il n'est pas indispensable que tous les composants logiques d'un document aient des termes associés, et de même, un terme peut indexer plusieurs composants d'un même document (voir figure 4.3). Les occurences ainsi définies expriment les liens entre les concepts et les composants du document.

- La RI d'un document peut être mise à jour par l'application, à tout moment, indépendamment du déroulement des autres opérations du serveur.

Relation d'Indexation d'un Document

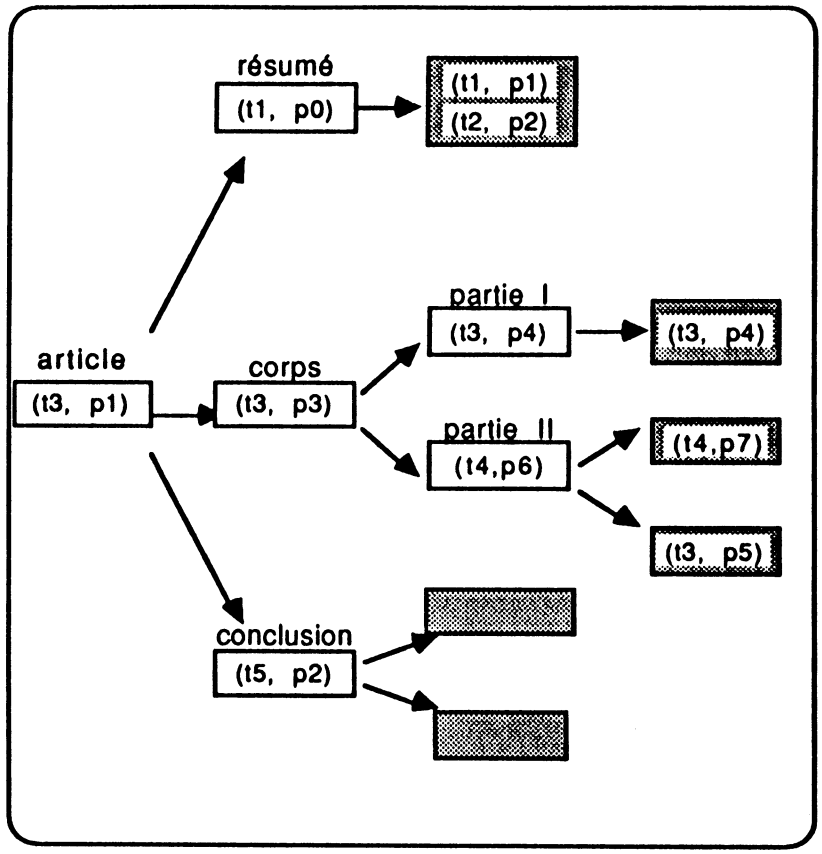


figure 4.3

Nous montrons dans la figure 4.3 un exemple d'une telle RI. Nous pouvons remarquer que la conclusion a été indexée seulement par le terme t_5, tandis que le corps du document est indexé à plusieurs niveaux. De même, la RI du résumé de l'article (article.résumé.RI) est constitué par l'ensemble de termes : $\{(t_1,p_0), \{(t_1, p_1), (t_2,p_2)\}\}$.

A partir de la liste de triplets *établis par l'application*, et envoyés au SIB, le Gestionnaire de Documents construit la relation d'indexation du document. Le processus de construction de l'arborescence procède par comparaison entre les références

trouvées dans les triplets et les identificateurs de la structure logique du document.

Pour exprimer une opération de recherche sémantique nous avons défini l'opérateur **IS RELEVANT TO** dans la clause **WHERE** des énoncés de sélection (cf. début de la section 4.3). Il permet la sélection des documents dont le contenu des ULD (unités logiques désignées comme espace de recherche) correspond à une condition donnée par une expression appelée **thème de recherche**.

Notre requête est donc formée par la partie **IS RELEVANT TO** dans la clause **WHERE**. Son évaluation est réalisée par la comparaison entre ce thème et la RI des ULD concernées, c'est à dire, les termes associés à touts les sous-arbres dépendant de chaque ULD. La réponse est la liste de triplets ((Id_doc,Id_élém_logique), terme, poids) vérifiant le thème de recherche.

Un thème de recherche est défini par une expression logique disjonctive de termes d'indexation via des opérateurs **AND**, **OR** et **NOT** (la précédence des opérateurs étant implicite). Nous pouvons aussi spécifier partiellement un terme d'indexation, en utilisant l'opérateur joker "*". La portée du joker "*" est limitée à un terme d'indexation. Il est possible de limiter le nombre de caractères des jokers intérieurs à un terme à un maximum de n caractères, en le définissant de la forme "*"[n]. Si n n'est pas spécifié la portée par défaut de "*" est le terme d'indexation complet.

La syntaxe générale d'un thème de recherche est la suivante :

< sujet_de_ recherche > ::= (<partie_and> { OR <partie_and> }*)

<partie_and> := <terme> { AND <terme> }*

<terme> ::= { NOT } <terme_index>

<terme_index> ::= {"*"}<chaîne_car>{{"*"{[n]}}<chaîne_car>}*{"*"}

Exemple :

```
SELECT   c.adresse
FROM     chercheur c, rapport r
WHERE    r.auteur = c.nom AND
         r.chapitre IS RELEVANT TO ("informatique*gestion")
```

Dans cet énoncé, nous voulons les adresses des chercheurs qui ont écrit des rapports dont les chapitres sont considérés pertinents, par exemple, relativement aux sujets "informatique de gestion" et "informatique appliquée à la gestion", définis par des termes d'indexation de la RI.

Dans l'exemple, la réponse est l'ensemble de références (donc, des unités textuelles) indexés par l'un des termes correspondant à "informatique de gestion". Aucun traitement n'est fait sur cette liste de références, afin de laisser aux applications toute la liberté d'interprétation (par exemple, définition de seuils, reformulation des requêtes, visualisation, etc.).

La syntaxe générale d'un énoncé de recherche sémantique est :

```
SELECT   <doc>.<attribut>
WHERE<doc>.ULD IS RELEVANT TO <thème_de_recherche>
```

<doc> étant un document, ou une spécialisation de documents.

Les triplets qui conforment la réponse à cette clause **WHERE** sont de la forme ((id_doc,id_élém_logique), terme, poids), où les couples (id_doc,id_élém_logique) sont les identificateurs des éléments logiques dépendants des ULD des documents <doc>, qui vérifient <thème_de_recherche>.

4.4. Méthodes d'Accès et de Stockage Associées

Les méthodes de recherche par le contenu de documents que nous envisageons peuvent être très coûteuses en temps d'évaluation, et les méthodes classiques qui permettent de les améliorer (inversion de mots ou des termes d'indexation) impliquent des gros investissements en mémoire de stockage et en pré-traitement de la base documentaire.

L'approche que nous avons choisie pour la définition des méthodes d'accès et de stockage vise à éviter ces inconvénients :

- Le surcoût en place dû aux structures qui permettent d'accélérer les recherches doit être relativement contrôlé. Par exemple, il ne devrait pas en aucun cas dépasser la taille des documents eux-mêmes.

- Le temps d'évaluation d'une requête par le contenu doit rester acceptable dans le cadre interactif d'un environnement de bureau.

- Les opérations de mise à jour doivent se faire en temps réel. Donc, il est inacceptable d'avoir des mises à jour différées ou qui impliquent un arrêt des applications.

Les méthodes d'accès et de stockage qui s'adaptent le mieux à ces conditions sont, comme nous l'avons décrit dans les chapitres 2 et 3, les méthodes de signatures. Nous avons vu par ailleurs (cf. chapitre 3) que l'utilisation de matériel spécialisé pourrait encore améliorer le filtrage de textes. Chaque méthode de recherche par le contenu (textuel ou sémantique) a ses caractéristiques propres qui impliquent la nécessité de définir des méthodes d'accès spécifiques, bien que guidées par les mêmes principes.

4.4.1. Accès Textuel aux Documents

Pour effectuer la RECHERCHE TEXTUELLE sur les documents il nous faut disposer au minimum :

- De l'accès à la structure logique des documents, afin de pouvoir en distinguer les portions de contenu textuel.

- De méthodes de comparaison de chaînes de caractères et, plus généralement, de méthodes d'évaluation d'expressions de filtrage.

Le premier aspect est assuré par le service d'INTERNALISATION du Gestionnaire de Documents. Les détails et les choix retenus dans l'OIS à ce sujet sont décrits dans [Mar87], [BJM88], [LG 89] et, dans la mesure où cela nous concerne directement, dans la description de la réalisation de ce travail (cf. chapitre 5).

Pour l'évaluation des expressions de filtrage nous avons identifié deux aspects : l'évaluation de l'expression logique, et la comparaison des chaînes de caractères ("pattern matching") dans le contenu textuel des documents.

Pour la comparaison des expressions de filtrage, un premier pas dans la définition de la méthode d'accès textuel aux documents a été la définition d'un **générateur d'automates d'états finis augmentés d'une mémoire** [Jim85]. Cette méthode n'as pas besoin de place de stockage supplémentaire, mais elle est, évidemment, très coûteuse en temps d'évaluation.

Nous avons envisagé deux alternatives pour améliorer le taux temps d'évaluation/surcoût de stockage : la définition d'un index textuel du document et l'utilisation de matériel spécialisé afin d'accélérer l'exécution des automates de filtrage. Des combinaisons de ces alternatives sont aussi possibles.

4.4.1.1. L'Index Textuel des documents

En raison des qualités (facilité de mises à jour, facilité de calcul et d'évaluation, peu de place additionnelle nécessaire) nous définissons l'**INDEX TEXTUEL** des documents du SIB comme *l'ensemble des signatures individuelles des documents* (voir figure 4.4).

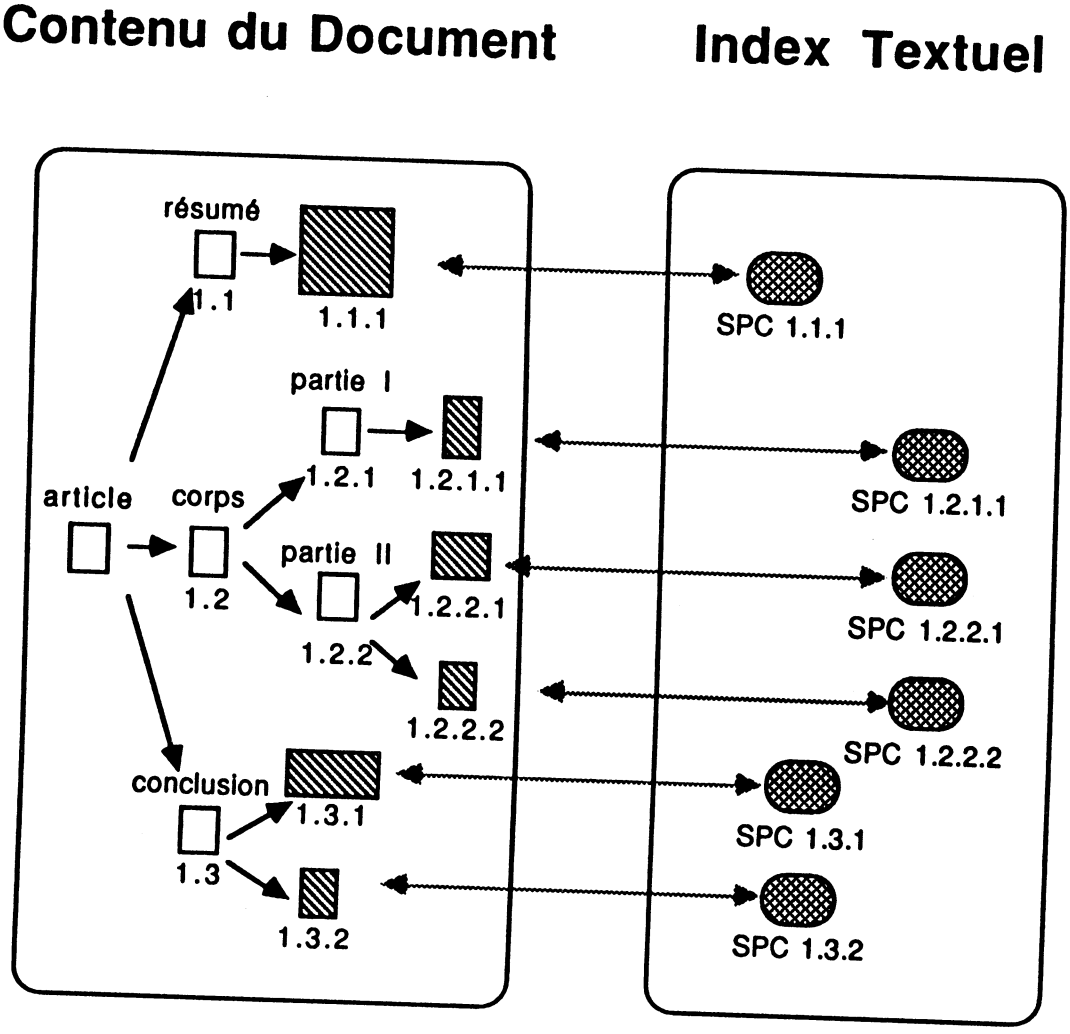


figure 4.4

L'index textuel d'un document est une image compressée de son contenu qui conserve sa structuration logique. Il est ainsi formé par l'ensemble de **signatures des portions de contenu, (SPC)**, chacune d'elles étant associée de manière biunivoque avec

la portion de contenu originelle. Cette association est symbolisée par les flèches ←→ de la figure 4.4.

Pour mettre en œuvre la **RECHERCHE TEXTUELLE** dans le SIB en utilisant l'index textuel des documents nous devons donc considérer deux types de traitement :

1. PHASE D'INDEXATION TEXTUELLE DES DOCUMENTS

   Lors de l'INTERNALISATION d'un document, son décodage permet d'identifier les éléments de la structure logique. L'index textuel du document est calculé, par la création et le stockage de la Signature de chaque Portion de Contenu (SPC).

2. PHASE DE SÉLECTION

   Lors de l'évaluation d'une opération de recherche textuelle on analyse l'expression de filtrage afin d'identifier ses éléments logiques. On calcule la signature des éléments de cette expression par le même algorithme de transformation utilisé à l'indexation. La sélection des documents qui correspondent à l'expression de filtrage est faite par la comparaison entre sa signature et celles des portions de contenu des documents (ou des composants des documents) concernés par la requête.

L'ensemble de portions de contenu ainsi sélectionné peut contenir des collisions. Pour donner des réponses exactes il faut procéder ensuite au filtrage des chaînes de caractères des portions de contenu sélectionnées, par l'automate correspondant (cf. 4.4.1.3 et 4.4.1.4).

De cette manière nous assurons que l'espace de recherche dans la base de documents du SIB est restreint, en éliminant dans l'étape de pré-sélection les portions de contenu qui ne correspondent certainement pas à l'expression de filtrage.

4.4.1.2. Calcul de la Signature d'une Portion de Contenu

La méthode de calcul de Signatures qui semble la plus adaptée pour l'établissement de l'index textuel est celle du **codage superposé** (cf. 3.2.3.2), en raison de la simplicité des opérations nécessaires, tant pour dans la phase d'indexation que pour la phase d'évaluation. Elle offre des très bons taux de compactage et, en raison de la superposition des codages, la signature des mots répétés dans une portion de contenu n'étant stockée qu'une seule fois. Les mises à jour de l'index textuel ainsi construit sont aussi assez simples, se limitant au calcul de la signature des nouvelles portions de contenu. L'évaluation de l'expression logique de sélection et l'évaluation des "joker" peut se faire par de simples opérations sur des chaînes de bits.

Dans le SIB la **signature d'une Portion de Contenu (SPC)** est formée par la superposition des signatures individuelles des mots qui le composent. Une SPC est un bloc de bits de taille fixe représentant le contenu d'une Portion de Contenu (celle-ci, de taille variable). Dans le calcul de la SPC sont codés tous les mots, ainsi que le séparateur entre mots (contrairement aux méthodes classiques, qui prennent des blocs de texte de taille fixe, éliminent les mots non-significatifs de la langue et ne gardent pas les séparateurs).

Cette définition de la SPC nous garantit que dans la structure même de l'index textuel la structuration logique du contenu du document est préservée, ainsi que **tout** son contenu. La réponse à une requête de recherche textuelle est l'ensemble d'éléments logiques de documents ULD, dont au moins une des PC_i à été sélectionnée.

Exemple :

Pour la document de la figure 4.4, si la requête est de la forme :

```
SELECT ...
WHERE article.conclusion TALKS ABOUT "recherche"
```

les SPC considérées dans l'évaluation du TALKS ABOUT sont SPC1.3.1 et SPC1.3.2. Si l'une d'entre ces deux signatures est sélectionnée par rapport à la requête "recherche", la conclusion de l'article est incluse dans la réponse de la clause **WHERE**.

Si F est la taille en nombre de bits d'une SPC, et M est le nombre de Portions de Contenu de la base de documents, la taille de l'index textuel est donnée par F * M bits.

Pour chaque terme de l'expression de filtrage (c'est à dire, chaque chaîne de caractères dans l'expression logique de la requête), et avec la même fonction de transformation, un bloc de signature S, de la même taille F est construit. Une Portion de Contenu PC_i vérifie le terme si tous les bits "1" de S sont "1" dans SPC_i. C'est à dire, $S \subseteq SPC_i$.

F doit être choisi de façon à optimiser à la fois la place occupée par l'index et la probabilité de collisions (le bit i de SPC et mis à "1" par au moins deux mots différents). Evidemment, si F est grand il y aura peu de collisions, mais la SPC sera très grande et très dispersée. Si F est trop petite beaucoup de bits sont "1" et la signature n'est pas assez discriminante.

Un autre type de collisions, indépendant de F, peut se produire en raison de la superposition même des codages des mots. En fait, l'ordre dans les chaînes de caractères constituant la requête est perdu dans certains cas dans le calcul de sa signature. Par exemple, la signature de "*inf*syst*" est calculée par la superposition de Sign(*inf*) et de Sign(*syst*) dans un même bloc de bits, et elle sera égale à la signature de "*syst*inf*". Ce type de collisions est inhérent à la méthode de codage.

Le problème principal des méthodes de signatures est donc le contrôle des collisions [CF 84]. On cherche donc à trouver, pour une taille F, quelle est la probabilité de collision lors de l'interrogation. La **probabilité de collision** PC_{CS} est définie

comme la *probabilité qu'une SPC_i soit sélectionnée alors que la portion de contenu associée PC_i ne l'est pas.*

Dans la méthode classique de SC [CF 84], on ne considère que les mots outils de la langue (les mots "vides" sont éliminés). Le texte est divisé en blocs contenant un nombre fixe **D** de mots outils, et pour chaque bloc est construit un bloc de signature de **F** bits. A chaque mot retenu on associe un nombre **m** de bits dans le bloc signature par une fonction de hachage.

Dans [CF 84] est montré analytiquement que les seuls paramètres qui déterminent les performances de la méthode CS sont F et D, et qu'elles sont indépendantes de la taille du vocabulaire de la base textuelle, de la taille de la base, de la distribution des occurrences des mots et de la fréquence des requêtes.

Il est montré que, pour un bit dans SPC_i, et avec une fonction de transformation uniforme,

$$P\ (\ SPC_i = \text{"1"}) = 1 - [\ 1 - 1/F]^m \ \sim\ m/F$$

m étant le nombre de bits résultant de la transformation d'un mot, et F la taille du bloc de signatures. La Probabilité de Collision dans une requête contenant un mot simple est [CF 84] :

$$PC_{CS} = [\ 1 - e^{-(mD/F)}\ ]\ m$$

où D est le nombre de mots retenus par bloc de texte. La valeur optimale de $m = F \ln(2) / D$

Pour le cas de blocs de texte de taille variable, l'analyse est faite en termes d'un taux de surcoût de mémoire constante, ***v***, qui est donné par le rapport entre la taille de la signature en bits et la taille du bloc de texte en bits. Celle-ci est estimée en fonction de la taille moyenne des mots μ en bits (prévoyant un caractère séparateur entre mots) et du nombre total **N** des mots par bloc.

Alors : $v = F / N * \mu$

$$N = D \ln (2D)$$

et la probabilité de collision est donnée par [CF 84] :

$$\ln PC_{CS} = - (\ln 2)^2 v\mu \ln(2D)$$

La probabilité de collisions est donc une fonction du surcoût de mémoire, de la taille moyenne des mots et du nombre de mots outils retenus par bloc de texte.

Dans les cas du SIB, les Portions de Contenu sont de taille variable et non prédéterminée (elles peuvent correspondre à un titre ou à un paragraphe complet). Donc D n'est plus une constante mais une variable dont la valeur dépend des tailles des Portions de Contenu de l'ensemble de documents.

Nous avons deux possibilités pour traiter ce problème :

- Calculer SPC sur des blocs de bits de taille variable.

- Définir une SPC de taille fixe, et estimer cette taille en fonction de la minimisation des collisions et de la valeur estimée de la taille des PC dans le SIB. Cette valeur correspond à l'estimation de la moyenne statistique dans la distribution des longueurs des portions de contenu de la base de documents.

Nous avons adopté la deuxième possibilité, puisque le fait d'avoir des signatures de taille variable implique beaucoup plus de complexité de traitement lors de l'évaluation de requêtes. En plus, le fait d'avoir des signatures de taille fixe implique que la taille de l'index textuel est indépendante de la taille des Portions de Contenu, même si elle reste proportionnelle en nombre .

Pour permettre la spécification de parties de mots dans les expressions de filtrage, nous avons adopté comme technique de calcul de la fonction de transformation, le découpage des mots en triplets superposés (cf. chapitre 3). Les triplets sont formés en

tenant compte de tous les caractères dans la Portion de Contenu (les caractères multiples éventuels de séparation entre mots sont réduits à un seul). Ceci permet aussi de préserver l'ordre entre les mots, en utilisant le triplet avec le séparateur et les caractères de fin et début de mot.

Exemple :

> Les triplets dans "*yeux noirs*" sont
>
> "_ye", "yeu", "eux", "ux_", "x_n", "_no", "noi", "oir", "irs", "rs_".
>
> C'est le triplet "x_n" qui donne l'ordre entre les deux mots. Si le texte est "*noirs yeux*" il est remplacé par "s_y", et on produit donc pour *"yeux noirs"* une signature différente de celle de *"noirs yeux"*.

La fonction de transformation calcule le codage de **chaque triplet** dans la portion de contenu PC, vers **un** bit dans SPC. Par définition, le nombre de triplets N_{PC} dans une portion de contenu PC est égal au nombre de caractères outils dans cette PC. Nous pouvons supposer que $N_{PC} \approx$ taille de PC. Soit N l'espérance mathématique de la valeur de cette taille dans l'ensemble des PC de la base. Le vocabulaire que nous traitons est donc l'ensemble de triplets possibles (et non pas des mots de texte comme c'est le cas classique).

Pour calculer la probabilité de collisions dans une requête, PC_{SPC}, nous nous inspirons du cas classique décrit en [CF 84]. Nous faisons les suppositions suivantes :

- Soit T le nombre espéré de triplets différents dans une PC. Donc, $T < N$
- Le nombre possible de signatures dans SPC $\gg$ T
- Le vocabulaire (triplets possibles dans la base) $\gg$ T
- Le nombre de portions de contenu dans la base M $\gg$ 1

- La sélectivité des triplets « M (le nombre de PC dans lesquels un triplet apparaît est petit par rapport à la taille de la base)

- La fréquence d'occurrences de triplets obéissent à une distribution géométrique.

- L'apparition d'un triplet sélectionné de manière aléatoire est indépendante des autres triplets dans les PC de la base.

- La fonction de transformation est uniforme.

Une requête est traduite en un ensemble de triplets, connectés éventuellement par des opérateurs logiques. Pour estimer les paramètres de la méthode, nous voulons calculer la Probabilité de Collision d'une requête constituée d'un terme t, $P_t(t)$. Supposons que t est constitué de m triplets. La signature de t est donc formée par un ensemble de m bits.

Soit $P(n)$ = Prob (n triplets ayant le bit ß comme signature)

Par hypothèse d'indépendance de distribution des triplets, pour un bloc de signatures de taille F,

P (pour un triplet quelconque ß = "1") = 1/F

P (m triplets ayant une configuration donnée) = m/F

et $P(n) = \binom{V}{n} (1/F)^n (1 - 1/F)^{V-n}$

Suivant le raisonnement de [CF 84],

$$P_t(t) = (Q/M)^m$$

où Q est le nombre espéré de SPC dans lesquelles ß="1", et M est le nombre total de blocs dans la base.

$$Q \leq M [1 - e^{-A/M}]$$

où A est le nombre de blocs dans lesquels se trouvent les n triplets.

Dans notre cas,

A/M = (m/F) * Esp (T), où Esp (T) est la valeur espérée de T.

et $P_t(t) = [1 - e^{-m*Esp(T)/F}]$

Alors, si la valeur espérée du nombre des triplets différents par PC est par exemple 120, et la taille choisie pour le bloc de signatures est 600 bits, la probabilité de collisions pour une requête d'un triplet est d'environ 18%. Evidement, si F est plus grand, la probabilité de collisions diminue, mais la taille de l'index augmente, et le temps de vérification des signatures est aussi plus grand.

4.4.1.3. Evaluation des Expressions de Filtrage Utilisant les Signatures

Pour évaluer une expression de filtrage (EdF) il faut comparer sa signature avec celles des Portions de Contenu des éléments logiques des documents concernés, les ULD[1].

Cette comparaison produit, pour chaque ULD concernée, la liste de toutes les identificateurs des SPC qui correspondent au critère de sélection. A cause des collisions, cette liste peut contenir des identificateurs dont les PC correspondants ne répondent pas à la

---

[1] Pour faciliter l'exposé nous parlerons par la suite, du document, bien que l'espace de recherche est, bien sûr, constitué des portions de contenu des éléments logiques à un certain niveau de la structure logique du document. Ce niveau est spécifié par l'ULD dans la clause WHERE (cf. 4.3)

requête. Pour avoir des réponses exactes il faut effectuer l'étape de filtrage par l'automate de recherche sur cet ensemble de PCs.

Nous avons intérêt à réaliser le plus précisément possible l'évaluation des SPC, afin d'éviter des collisions ou des présélections inutiles. Ceci dépend de la méthode de calcul et d'évaluation de la **signature de l'expression de filtrage, SEdF**. Nous avons quatre cas :

1. L'EdF est une chaîne de caractères.

La SEdF est un bloc de bits de taille F, tel qu'à chaque triplet de EdF correspond un bit dans SEdF. De la même manière que pour SPC, on considère un caractère séparateur entre mots, un séparateur avant le premier mot et un autre après le dernier mot.

Si la chaîne de caractères contient des "joker", ils doivent être évalués dans chaque PC (par définition de la portée de l'opérateur "*"). Dans un bloc de signature d'une portion de contenu est perdue la notion de phrase et de mot ; en fait, touts les triplets sont codées dans le même ensemble de bits, de manière homogène. Les joker "$" et "!" (cf. 4.3) sont donc traités, au niveau de l'évaluation des signatures, de la même manière que "*". C'est seulement lors de l'élimination de collisions par l'automate qu'ils peuvent être distingués. Donc, la signature de toutes ses sous-chaînes doit apparaître dans la même SPC pour que la PC soit sélectionnée. Les triplets contenant les séparateurs autour des sous-chaînes ne sont pas générés.

Nous pouvons résumer ce cas de sélection d'une ULD U par :

$$U \text{ est sélectionnée} \Leftrightarrow \exists SPC_i \in U \mid SEdF \subseteq SPC_i$$

Exemple :

EdF_1 = "bras droit"

EdF_2 = "bras*droit*"

Les triplets dans EdF_1 sont :

"_br" "bra" "ras" "as_" "s_d" "_dr" "dro" "roi" "oit" "it_"

Les triplets dans EdF_2 sont :

"_br" "bra" "ras" "dro" "roi" "oit"

Pour F=20

EdF_1 = "10000101100111000110"

EdF_2 = "10000100100101000010"

2. L'EdF est la négation d'une chaîne de caractères.

Dans ce cas, un document n'est sélectionné que si aucune de ses SPC ne correspond à l'EdF. Le cas échéant, la liste des PC peut être formée seulement de collisions, ou de PC qui contiennent des chaînes de l'EdF. Pour obtenir des réponses exactes il faut donc filtrer par l'automate cette liste. Si les collisions sont tolérées, et étant donné que la probabilité de collisions est faible, on peut ne sélectionner que les documents dont la liste de PC présélectionnées est vide. C'est le seul cas où l'évaluation de l'EdF peut produire du silence (non-sélection de documents qui répondent à la requête). Le taux de silence est alors égal au taux de collisions.

Nous pouvons résumer ce cas de sélection d'une ULD U par :

$$U \text{ est sélectionnée} \Leftrightarrow \nexists SPC_i \in U \mid SEdF \subseteq SPC_i$$

La SEdF est calculée de la même façon que dans le cas sans négation, c'est seulement le critère de sélection des SPC qui change.

3. L'EdF est une conjonction de chaînes de caractères.

Dans ce cas EdF est une *expr_and*, de la forme

EdF = "$Chaîne_1$ AND $Chaîne_2$ AND ... $Chaîne_n$"

La portée de l'opérateur AND est définie comme étant toute l'ULD. Pour n = 2, si $Chaîne_1$ est trouvée dans PC_i et $Chaîne_2$ est trouvée dans PC_k, où PC_i et PC_k sont des portions de contenu de l'ULD U, U doit être sélectionnée.

Ceci doit être reflété dans la formation de la signature de l'expression, SEdF, qui est formée par **n blocs** (le nombre de chaînes dans l'expression conjonctive). Lors de l'évaluation, chaque SPC est comparée à tous ces **n** blocs. Si elle correspond à au moins une des signatures de chaînes, la SPC est sélectionnée. Une ULD est sélectionnée si au moins une de ses SPC a été sélectionnée et si chacune des signatures des chaînes a été vérifiée par au moins une de ses SPC.

Nous pouvons résumer ce cas de sélection d'une ULD U par :

U est sélectionnée $\Leftarrow\Rightarrow$

$\exists SPC_i \in U, ..., SPC_k \in U \mid SEdF_1 \subseteq SPC_i$ et ... et $SEdF_n \subseteq SPC_k$

où $SEdF_1$, ... $SEdF_n$ sont respectivement les signatures des n arguments de l'expression conjonctive. Les SPC_i, ..., SPC_k ne sont pas forcément distincts.

Si SEdF était calculé sur un seul bloc de bits, par superposition des signatures des chaînes individuelles comme dans le cas 1, nous perdrions l'information sur l'individualité des chaînes dans la signature. Lors de la comparaison avec les SPC, nous ne pourrions en effet plus distinguer quels sont les bits qui correspondent à chacune des chaînes de départ. Par ailleurs, la portée du AND serait restreinte à une seule PC.

Exemple :

Si EdF = "toit AND porte"

Signature (toit) = "111100000000000"

Signature (porte) = "0000000011110000"

et si nous construisons SEdF = "1111000111100000"

Supposons SPC_k = "1111001100100001"

SPC_K ne sera pas sélectionnée, même si elle contient "toit", parce que dans l'évaluation ne tient pas compte seulement du test d'inclusion de S(toit) dans SPC_k .

4. L'EdF est une disjonction d'expressions conjonctives.

Dans ce cas EdF est une *expr_or,* de la forme

EdF = "*expr_and*$_1$ OR *expr_and*$_2$ OR ... OR *expr_and*$_m$"

La portée de l'opérateur OR est aussi définie comme étant toute l'ULD. Par le même raisonnement que dans le cas des expressions conjonctives, la signature de EdF doit permettre l'évaluation de chaque *expr_and*$_k$ de manière indépendante. Si au moins une des *expr_and* est vérifiée, l'ULD est sélectionné.

Nous pouvons résumer ce cas de sélection d'une ULD U par :

U est sélectionnée $\Leftarrow\Rightarrow$

$\exists SPC_i \in U, \mid SEdF_1 \subseteq SPC_i$ ou ... ou $SEdF_m \subseteq SPC_i$

où $SEdF_1, \ldots SEdF_m$ sont respectivement les signatures des m arguments de l'expression disjonctive.

Exemple :

Si EdF = "toit OR porte"

Signature (toit) = "1110011000000000"

Signature (porte) = "0001000011100000"

Si nous trouvons SPC_k = "1110011001000011"

l'ULD est sélectionnée, même si S(porte) n'est pas présent dans SPC_k.

SEdF est donc formée par un arbre, où chaque fils de la racine représente la signature d'une expression conjonctive, celle-ci étant à son tour composée d'autant de blocs qu'elle contient de chaînes de caractères (termes).

Exemple :

EdF = "syst*intelli*" AND "expert*" OR "base* déduct*"

expr_and 1 *expr_and 2*

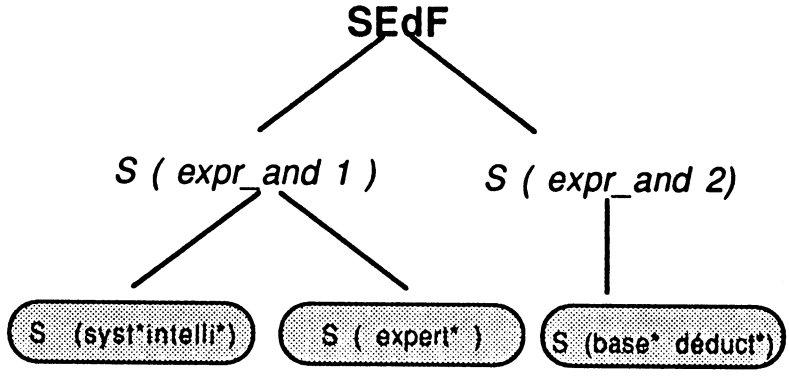


figure 4.5

En applicant les règles précédentes, sont sélectionnées les ULD ayant au moins une portion de contenu satisfaissant *expr_and1* ou *expr_and2*.

4.4.1.4. Le Générateur d'Automates

La méthode d'accès textuel que nous avons définie par la création d'un index de signatures, nous permet d'obtenir un ensemble de réponses qui peuvent contenir des collisions. Pour obtenir des réponses exactes il est donc nécessaire de procéder au filtrage du contenu textuel lui-même, sur les portions de contenu pré-sélectionnées.

A chaque expression de filtrage correspond un automate augmenté d'une mémoire [Jim85] qui permet de la reconnaître dans l'ensemble de portions de contenu. Cet automate est généré en plusieurs étapes, schématisées par la figure 4.6 :

Phase d'analyse syntaxique

Elle prend en entrée une expression de filtrage, identifie les éléments de l'expression logique et les sous-chaînes. Les caractères joker sont identifiés et les séparateurs de mots nécessaires (début, fin ou entre mots) sont ajoutés. Cette phase produit comme résultat une liste de sous-chaînes à reconnaître, avec un identificateur interne qui permet de retrouver sa position dans l'expression originale.

Exemple :

Pour l'expression "a*b AND c*d" la liste résultat est :

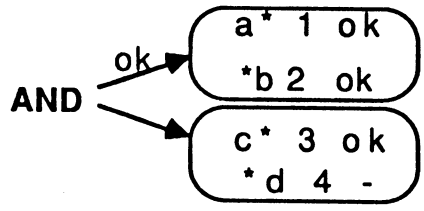
{ (a*, 1), (*b, 2), (c*, 3), (*d, 4) }

Phase de création de la structure de mémoire de reconnaissance.

Cette phase initialise la structure qui permet d'évaluer l'expression logique de recherche, ainsi que la séquence d'occurrences des sous-chaînes dans le texte.

Exemple :

Pour l'exemple précédent, et si une PC contient "adbc", la structure de reconnaissance "mémorise" que "a*b" a été reconnu, tandis que "c*d" ne l'a pas été. Ceci est montré dans la figure ci-dessous :



La structure de mémoire permet aussi la détection avancée de fin de reconnaissance, lorsque l'expression logique peut être évaluée avant la fin de l'analyse du document.

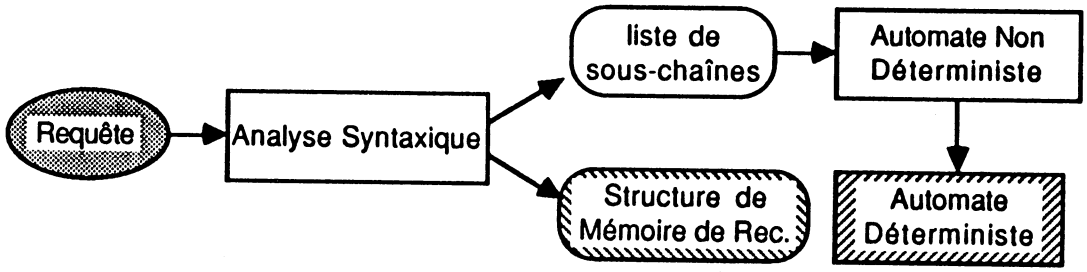


figure 4.6

Phase de création d'un automate d'états finis non déterministe.

A partir de la liste de sous-chaînes produite par l'analyse syntaxique, un automate d'états finis est créé [HU77] [AU 77]. Il a comme vocabulaire les caractères présents dans l'ensemble des

sous-chaînes. Il est formé par la concaténation des automates correspondant à chaque sous-chaîne (voir figure 4.7).

Il a trois types d'états :

- Un état initial permet les transitions vers les débuts de toute les sous-chaînes. Cet état sert aussi d'état d'échec dans la reconnaissance.

- Les états qui sont générés pour la reconnaissance d'une suite de caractères.

- Les états de reconnaissance d'une sous-chaîne.

Tout les états comportent 4 types de transitions :

- Une transition "avant", suite à la reconnaissance d'un caractère d'une sous-chaîne.

- Une transition "arrière", vers le premier caractère de la sous-chaîne.

- Une transition "saut" vers chacun des états reconnaissant le début des autres sous-chaînes.

- Une transition "échec" vers l'état initial, si le caractère en entrée n'est pas reconnu.

Un état peut accepter une sous-chaîne, identifiée par sa position dans l'expression de filtrage. Les transitions "arrière" et "saut" peuvent engendrer des non-déterminismes.

Exemple :

EdF = "*syst*" AND *tous*

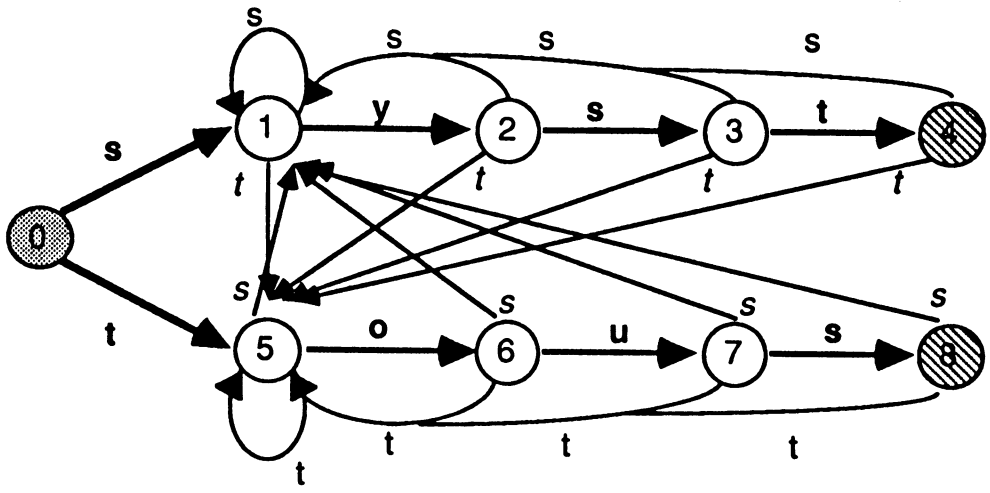


figure 4.7

L'état 0 est l'état initial, les états 4 et 8 sont les états finaux accepteurs. Les transitions "échec" ne sont pas marquées afin de ne pas encombrer la figure, elles renvoient toutes à l'état 0. Les états 2, 3 et 7 sont non-déterministes.

Phase de conversion vers un automate d'états finis déterministe.

L'automate créé lors de la phase précédente est converti, par des algorithmes classiques [LRS86], [HU 77], vers un automate d'états finis déterministe. Par construction, les états accepteurs peuvent reconnaître plusieurs sous-chaînes. C'est cet automate qui sera activé lors du balayage des portions de contenu sélectionnées.

Phase d'interprétation

Cette phase correspond à l'analyse, par l'automate déterministe, d'un ensemble de portions de contenu. Cette évaluation prend en

compte la portée des opérateurs joker, et l'ordre de reconnaissance de sous-chaînes lors de l'arrivée à un état accepteur. L'action de l'automate se termine soit à la fin du balayage de l'ensemble de portions de contenu, soit avant, dès que l'expression logique est vérifiée.

4.4.2. Accès Sémantique aux Documents

Pour effectuer la **RECHERCHE SÉMANTIQUE** il est nécessaire :

- d'avoir accès aux éléments de la RI des documents pour lesquels elle a été définie, ainsi qu'à leur structure logique.

- de disposer de méthodes de stockage et de comparaison de termes d'indexation pour évaluer les thèmes de recherche qui sont exprimés dans l'expression de sélection.

Pour stocker la RI des documents du SIB nous avons plusieurs possibilités :

- Stockage par inversion des termes d'indexation.

- Stockage de manière associative, par élément logique du document.

L'inversion de termes n'est pas appropriée pour l'environnement de bureau (cf. chapitres 2 et 3). Par contre, le principe de stockage associatif par document (comme c'est le cas pour les méthodes de signatures) offre des bons rapports de performances traitement des mises à jour / complexité et temps d'accès. Nous avons choisi d'associer les éléments de la RI, non pas par terme d'indexation, comme cela à été fait dans les méthodes classiques, mais par élément logique dans la structure des documents : à chaque élément de la structure est associé l'ensemble de termes d'indexation pondérés qui décrivent son contenu.

En la considérant comme ceci, la RI est vue comme un arbre (cf. figure 4.3), dont chaque nœud contient les couples *(terme d'indexation, poids)* . La *référence à l'élément logique* est donnée de manière implicite par sa position dans l'arbre. Les nœuds correspondent aux éléments de la structure logique du document. Il peut évidement y avoir des nœuds qui n'ont pas de termes d'indexation associés. Par contre, un terme n'apparaît qu'une seule fois dans un nœud avec un poids associé, exprimant l'importance du concept dans l'unité correspondante.

Une expression de recherche sémantique est évaluée dans toute la sous-arborescence, ayant comme racine le niveau indiqué dans la requête, ULD (c'est à dire, l'*unité textuelle* associée). En fait, un terme peut ne pas indexer directement un chapitre, mais une de ses sections. La réponse est formée par les triplets (référence, terme, poids) sélectionnés dans la RI de l'ULD, c'est à dire, dans tout le sous-arbre correspondant de la RI (cf. 4.3.2). L'application décide, en fonction de ses propres critères, des seuils ou des réévaluations à partir de cette liste de références et de poids.

Le stockage de la RI est toujours très coûteux. Nous introduisons donc des méthodes de compression des termes d'indexation en utilisant des signatures. Cette approche rejoint le principe de stockage et d'accès que nous avons défini pour l'index textuel des documents.

L'**index sémantique** de documents dans le SIB est défini comme l'ensemble des signatures des relations d'indexation des documents pour lesquels elle a été définie. Lors d'une mise à jour d'un document dans le serveur, ou sur sa RI, seul l'index correspondant à ce document est concerné, la signature de la nouvelle RI est recalculée.

La Signature d'une RI est un arbre dont les nœuds contiennent les signatures des termes d'indexation associés à chaque niveau dans la structure logique (cf. figure 4.8).

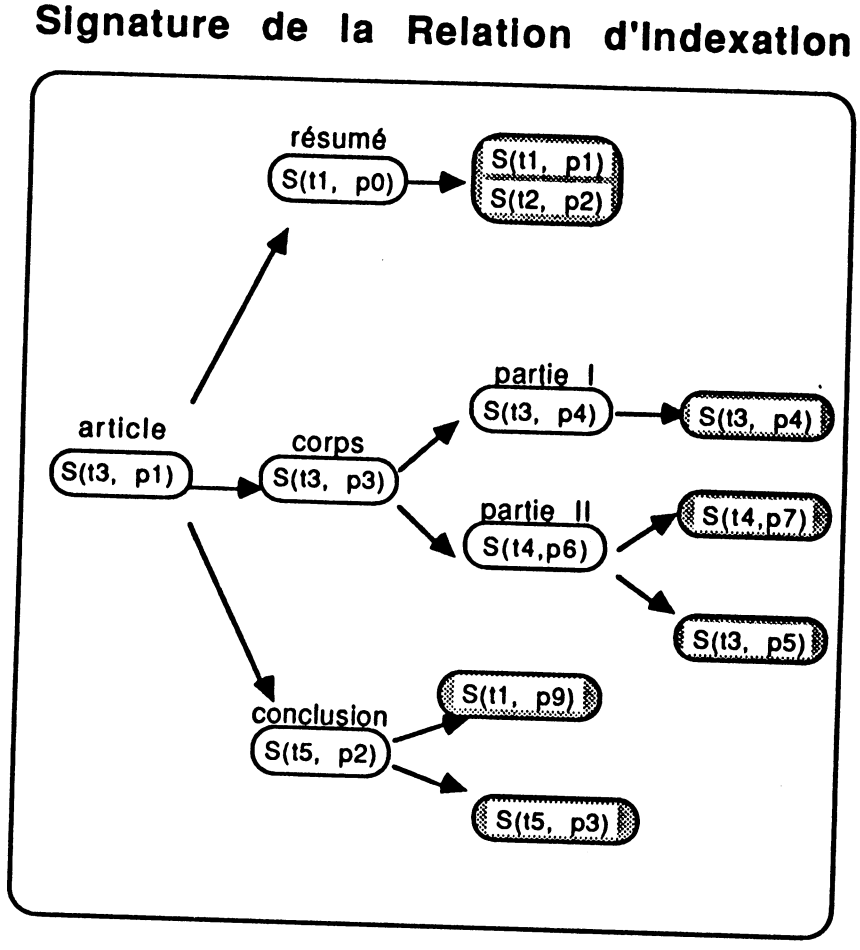


figure 4.8

4.4.2.1. Signature d'une Unité d'Indexation

La méthode de calcul par superposition de signatures utilisée pour l'index textuel a l'avantage de donner une vision de bloc de portion de contenu, mais on perd l'individualité de chaque concept indexé. Les méthodes de signatures existantes offrent très peu de possibilités pour exprimer les relations sémantiques de la RI. Les termes d'indexation correspondent à une occurrence pondérée d'un concept, et la superposition de leurs signatures n'a pas de sens. Même si les rapports de mémoire nécessaire ne sont pas aussi performants, la signature des unités d'indexation de la RI doit

avoir les caractéristiques de séquence que l'on trouve dans des méthodes de calcul comme les Signatures par Mot (cf. chapitre 3).

La signature de la RI est l'arborescence (isomorphe à celle de la structure logique du document) des signatures de ces unités d'indexation (celles-ci correspondant aux éléments de la structure logique). La **signature d'une unité d'indexation** est constituée par la séquence des signatures individuelles des termes d'indexation qui lui sont associés et de leurs poids . Chacun des termes d'indexation a une seule occurrence dans cette séquence.

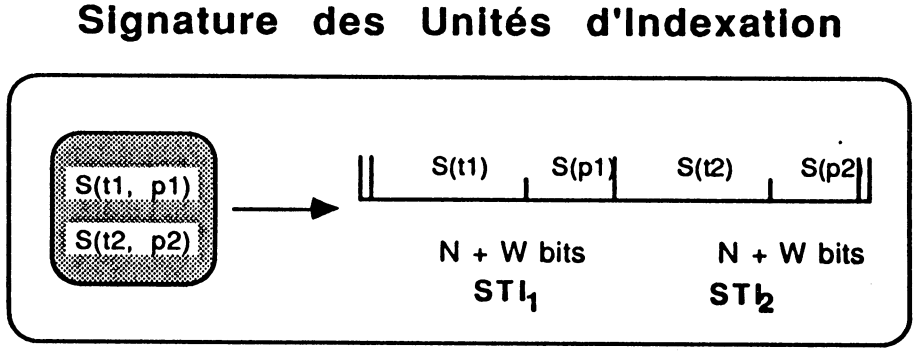


figure 4. 9

La **Signature de Terme d'Indexation, STI**, est donc un doublet comprenant (voir exemple figure 4.9) :

- Une image du terme d'indexation, calculée par une fonction de hachage sur un bloc de bits de taille fixe N.

- La pondération associée au terme. Elle est définie sur un nombre W de bits.

Les termes d'indexation sont les porteurs de la sémantique du contenu du document. Ils ont dû être sélectionnés et normalisés par un traitement d'indexation propre **à l'application**. Ce traitement peut, par exemple, être analogue à celui utilisé dans le système IOTA, mettant en œuvre des traitements

essentiellement linguistiques. Nous pouvons imaginer de considérer d'autres informations associées aux termes d'indexation. Par exemple, la catégorie grammaticale de chaque mot qui le compose (substantif, adjectif, groupe nominal composé, etc) [Ker84], ou des informations sur la conjugaison, le nombre. Nous avons décidé de ne conserver que le terme lui même et sa pondération dans la STI, parce qu'ils constituent les éléments fondamentaux de la définition de la RI. Les autres attributs pourraient être ajoutés au moment de la réalisation d'un serveur en particulier, tout en considérant l'impact sur le volume additionnel de données à stocker.

4.4.2.2. Evaluation des Expressions Logiques

Le thème de recherche dans une opération de recherche sémantique peut être exprimé par une expression logique de termes d'indexation. Pour évaluer cette expression (**EdR**), il faut comparer ses arguments avec ceux de toutes les Signatures des unités d'indexation des documents concernés. Il nous faut donc calculer la Signature de l'expression logique, comme cela a été fait aussi pour l'évaluation des expressions de recherche textuelle. Néanmoins, le calcul de la signature change un peu, pour tenir compte de la portée des opérations et de la forme de stockage des signatures des unités textuelles.

De façon analogue à la signature des expressions de filtrage pour la recherche textuelle (cf. 4.4.1.3), la signature de l'expression de recherche (**SEdR**) est un arbre qui reflète la structure de l'expression logique, ayant au plus bas niveau les signatures de termes d'indexation.

Pour montrer le calcul de SEdR nous considérons cinq cas :

1. L'EdR est un terme d'indexation sans opérateurs joker.

Dans ce cas, la SEdR est un bloc de bits de taille N, le même que celui des images des termes d'indexation. La SEdR est calculée par le même algorithme de construction des STI, est l'on compare

l'**égalité** de SEdR avec chaque élément (cf. figure 4.7) de *toutes* les unités d'indexation qui forment la RI des ULD (cf. 4.3.2). Les STI constituant une signature d'unité d'indexation sont comparés séquentiellement jusqu'à l'égalité avec la SEdR (par définition, une STI est unique dans une unité d'indexation, puisque le terme d'indexation est unique pour cette unité).

Si SEdR = STI_{km}, pour le terme T_k de l'unité U_m, l'identificateur de U_m et le poids associé a T_k sont ajoutés à la liste de références qui constituent la réponse.

Nous pouvons résumer ce cas de sélection d'une ULD U par :

$$U \text{ est sélectionnée} \Leftrightarrow \exists U_j \in U, STI_i \in U_j, \mid SEdF = SPC_i$$

Exemple :

Si l'unité U = { (T_1 = "protocole", 0,3), (T_2= "interface", 0,8) }

et EdR = "interface",

nous avons SEdR = STI_2, l'unité U est sélectionnée et l'on ajoute à la liste de réponses le triplet (U, T_2, 0,8).

2. L'EdR contient des opérateurs joker.

Dans ce cas, la SEdR est aussi un bloc de bits de taille N et la SEdR est calculée par le même algorithme de construction des STI. La comparaison est faite sur l'**inclusion** (en termes de comparaison d'ensembles de bits) de la SEdR dans chaque STI dans *toutes* les unités d'indexation des ULD concernés. La comparaison avec une STI est similaire à celle faite sur des blocs de signatures des portions de contenu lors de l'évaluation d'expressions de filtrage textuel.

Si SEdR $\subseteq STI_{km}$, pour le terme T_k de l'unité U_m, l'identificateur de U_m et le poids associé a T_k sont ajoutés à la liste de références qui constitue la réponse (voir exemple précédent).

Nous pouvons résumer ce cas de sélection d'une ULD U par :

$$U \text{ est sélectionnée} \Leftrightarrow \exists U_j \in U, STI_i \in U_j, \mid SEdF \subseteq SPC_i$$

3. L'EdR est la négation d'un terme d'indexation.

Son évaluation implique le balayage de toutes les RI des ULD de manière équivalente aux cas précédents, et la sélection ULD pour lesquels aucune des unités d'indexation n'a été pré-sélectionnée.

Nous pouvons résumer ce cas de sélection d'une ULD U par :

$$U \text{ est sélectionnée} \Leftrightarrow \nexists U_j \in U \mid STI_i \in U_j, SEdF = SPC_i$$

4. L'EdR est une conjonction de termes d'indexation.

La construction de SEdR est faite par la création d'un ensemble de signatures individuelles des termes de la conjonction. Chacune est évaluée par égalité ou inclusion, comme nous l'avons décrit dans les cas précédents.

Exemple :

Si les unités

$U = \{ (T_1 = \text{"protocole"}, 0.3), (T_2 = \text{"interface"}, 0.8) \}$

et $V = \{ (T_1 = \text{"protocole"}, 0.9), (T_3 = \text{"communication"}, 0.5) \}$

et EdR = "interface" AND "protocole",

nous avons SEdR = [S(interface), S(protocole)]

Dans la liste de réponses sont ajoutés les triplets

$\{ (U, T_1, 0.3), (U, T_2, 0.8), (V, T_1, 0.9) \}$

Le triplet (V, T_1, 0.9) parce que la portée d'évaluation de l'opérateur logique AND n'est pas limité à une seule unité, mais à toute l'ULD.

Nous pouvons résumer ce cas de sélection d'une ULD U par :

U est sélectionnée $\Leftarrow\Rightarrow$

$\exists U_j \in U \mid STI_j \in U_j$ et $SEdF_i \otimes SPC_i$, ...,

$STI_k \in U_j$, $SEdF_n \otimes SPC_n$

où $SEdF_1$, ... $SEdF_n$ sont respectivement les signatures des n arguments de l'expression conjonctive. Les SPC_i, ..., SPC_k ne sont pas forcément distincts. Le symbole $\otimes$ représente soit l'égalité, soit l'inclusion selon que $SEDF_i$ représente la signature d'un argument sans ou avec des joker.

5. L'EdR est une disjonction de termes d'indexation.

La SEdR dans ce cas est, comme pour les expressions de recherche textuelle, un arbre qui reflète la structure logique de l'EdR. La réponse est constituée par toutes les références pondérées et ordonnées des unités d'indexation sélectionnées.

Nous pouvons résumer ce cas de sélection d'une ULD U par :

U est sélectionnée $\Leftarrow\Rightarrow$

$\exists U_j \in U \mid STI_j \in U_j$ $SEdF_i \otimes SPC_i$ ou , ...,
ou $STI_k \in U_j$, $SEdF_n \otimes SPC_m$

où $SEdF_1$, ... $SEdF_m$ sont respectivement les signatures des m arguments de l'expression disjonctive. Les SPC_i, ..., SPC_k ne sont

pas forcément distincts, et le symbole ⊗ a la même signification que dans le cas précédent.

4.4.2.3. L'Indexation Dynamique dans le SIB

Notre définition de l'index sémantique des documents dans le SIB permet l'exploitation indépendante de la RI par des applications de recherche d'information développées sur le serveur. Il est néanmoins intéressant d'étudier comment y intégrer, au niveau du serveur, plus d'éléments de support aux processus d'indexation et d'interrogation de ce type d'applications. Le fait d'introduire dans l'opération de recherche sémantique des critères spécifiques d'exploitation de la RI peut entraîner une certaine perte de généralité par rapport à d'autres méthodes, mais donne plus de richesse aux opérations primitives d'accès et de sélection de documents.

Nous prenons par exemple le cas de IOTA, où l'indexation de documents est faite de manière dynamique [Ker84] [Def86]. Les termes d'indexation sont choisis et normalisés d'après les informations contenues dans le thésaurus (cf. chapitre 2). Ils ont deux poids associés :

- La représentativité du terme par rapport au document, évaluée comme la fréquence relative du terme dans le document (ou l'unité textuelle),

- et la représentativité du document par rapport au terme, évaluée comme le rapport entre la fréquence relative du terme dans le document, et sa fréquence globale dans tout le corpus.

Une fois que les termes sont choisis au plus bas niveau dans la structure logique du document, ils sont *remontés* (associés aux niveaux supérieurs dans la hiérarchie). Le critère pour décider qu'un terme monte dans la structure à un niveau *i* est qu'il indexe tous les éléments du niveau *i+1*, et les nouveaux coefficients de

pondération sont calculés en fonction de ceux existant au niveau *i+1*.

Dans IOTA la RI est établie une fois pour toutes, et elle est stockée par terme d'indexation, à chacun desquels on associe toutes les références pondérées. Le corpus traité est connu dès le démarrage du système, et il correspond à un domaine d'application fermé. Le corpus est considéré comme stable, et donc il ne fait pas l'objet de mises à jour.

Dans l'intégration des opérations d'indexation dynamique dans le SIB, deux aspects sont à considérer :

- La prise en compte du caractère dynamique du corpus.

- Le calcul dynamique de la RI pour un document donné.

La sélection et normalisation des termes d'indexation d'un document sont toujours considérées comme étant à la charge des applications.

Le fait de que le corpus soit dynamique implique que, même si les connaissances sur le domaine qu'il traite sont bien établies dès le début, ils est pratiquement impossible de maintenir des mesures de pondération qui prennent en compte le corpus complet, comme c'est le cas dans IOTA. Il faudrait les recalculer, pour l'ensemble des termes de tous les documents, à chaque fois qu'une mise à jour, insertion ou suppression est faite, tâche très coûteuse même si le corpus n'est pas volumineux.

Dans IOTA, la mesure de *représentativité du document par rapport à un terme* fait appel aux occurrences de ce terme dans l'ensemble des documents. Cette mesure n'est pas économiquement réévaluable dans le contexte du SIB, où cet ensemble est dynamique.

Dans le SIB, il n'est donc envisageable de ne garder comme pondération de la RI, que la mesure de la *représentativité du*

terme par rapport au document, qui est calculée localement à l'ensemble des termes d'indexation du document.

Le calcul dynamique de la RI d'un document comprend deux phases:

1. La sélection des termes et leur normalisation, qui est toujours à la charge de l'application. C'est elle qui dispose des connaissances nécessaires sur le domaine traité par le document.

2. Le calcul de la remontée des termes d'indexation dans la structure logique du document.

Le calcul dynamique de la remontée des termes peut être réalisé au niveau du SIB. Ceci implique que l'application n'a besoin de spécifier que les termes et pondérations qui indexent les éléments au plus bas niveau dans la structure logique. Dans IOTA ceux-ci sont appelés les unités minimales d'indexation. Cette rédéfinition de l'indexation sémantique dans le SIB nous permet d'opérer des simplifications au niveau de la définition même de la RI, et de construire un index sémantique de documents moins volumineux. Nous l'appelons la **RID, Relation d'Indexation Dynamique**. Il est clair qu'une application utilise, soit la RI définie auparavant, soit la RID, cas dans lequel elle décharge une partie de la tâche d'indexation sur le SIB. En fait, par des raisons d'homogénéité du Gestionnaire de Documents, seule une de ces alternatives doit être offerte.

Dans le cas de calcul dynamique des termes d'indexation, l'index sémantique du SIB est rédéfini comme l'ensemble des RIDs des documents pour lesquels l'application a prévu une indexation. La RID est une RI (donc un ensemble de triplets *(référence, terme, poids)*) qui n'est **instantiée qu'au niveau des portions de contenu**. Pour les documents du SIB elles correspondent aux unités minimales d'indexation.

Relation d'Indexation Dynamique

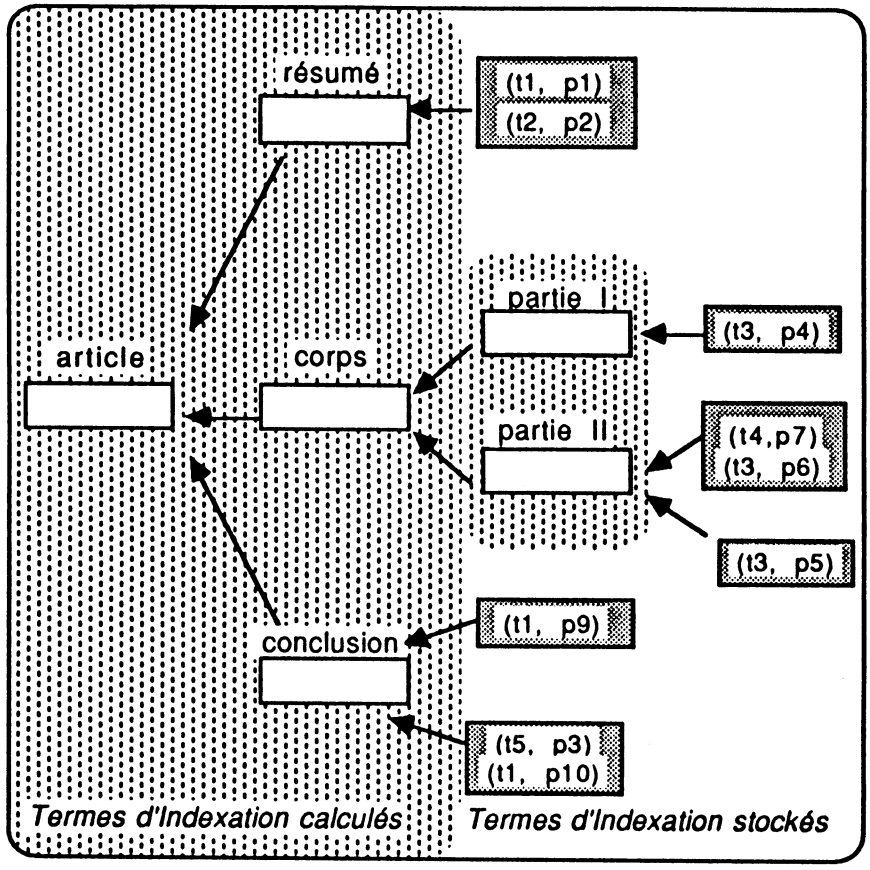


figure 4.10

L'index sémantique des documents ainsi construit n'occupe que la place nécessaire à l'ensemble de termes et pondérations correspondant aux portions de contenu. C'est lors de l'évaluation des requêtes que le SIB prend en charge la tâche de remontée des termes et des poids correspondants dans la structure logique, pour les termes concernés dans la requête. Le temps de traitement des requêtes est par conséquent augmenté par cette sélection des termes et par le calcul des poids correspondants aux niveaux supérieurs de la structure logique du document.

Le critère de remontée des poids est inspiré de celui de IOTA [Ker84]. La pondération qui nous intéresse dans le calcul des poids des termes est celle de la représentativité d'un terme dans le document.

Nous les adaptons de la manière suivante :

- Tous les termes qui indexent l'introduction, le résumé et la conclusion du document sont remontés au niveau supérieur de la hiérarchie. Il en est de même pour les titres des différents niveaux de la structure, car ils sont considérés comme résumant l'information véhiculée par l'unité à laquelle ils sont attachés [Ker84].

- Pour les autres éléments de la structure logique, sont remontés vers le niveau supérieur tous les termes d'indexation communs à tous les éléments fils de ce nœud. La pondération qui leur est associée est fonction de celle des nœuds fils.

Soit :

$E_i = \{e_i\}$ est l'ensemble des nœuds de niveau i dans la structure logique du document, fils du nœud $e_{i-1} \in E_{i-1}$. Soit $N = \| E_i \|$, et n_i le nombre de termes d'indexation associés à e_i.

$T_i = \{t_k\}$ est l'ensemble de termes d'indexation communs aux éléments de E_i. Alors, $n_{i-1} = \| T_i \|$

Si repr (t_k, e_i) est la représentativité de t_k par rapport à e_i, la remonté des termes T_k consiste à calculer, pour le niveau i-1, repr (t_k, e_{i-1}).

$$repr\ (\ t_k,\ e_{i-1}) = \sum_{j=1}^{N} \frac{n_j}{\sum_{j=1}^{N} n_j} * repr\ (\ t_k,\ e_{ij}\ )$$

Cette relation exprime que la représentativité d'un terme par rapport à un élément logique en fonction de la représentativité du terme dans chacun de ses fils.

Une expression de recherche sémantique, dans le cas d'utilisation d'une indexation dynamique est évaluée de la manière suivante :

1. Identification des termes à évaluer dans l'expression de sélection.

2. Pour l'ensemble des ULD concernés, identification de l'ensemble des unités textuelles associées aux Portions de Contenu qui les forment. Sur cet ensemble, sélection des unités textuelles dont les termes d'indexation répondent à la requête.

3. Pour ces unités textuelles, calcul de la remontée des termes jusqu'au niveau de la structure logique spécifié par ULD.

4. Production de la liste complète des références sélectionnées. Sont sélectionnées les références correspondant au niveau de réponse demandé (ULD), avec leurs mesures de représentativité, et les référence qui se trouvent dans les niveaux inférieurs de la hiérarchie ayant un poids supérieur.

   Exemple : Si un sous-arbre S d'une ULD donnée a 2 unités textuelles U et V au niveau n de la structure logique, et si la RID contient :

   $U = \{ (T_1, 0.6), (T_2, 0.8) \}$ $V = \{ (T_3, 0.3), (T_2, 0.3), (T_4, 0.5) \}$

le mécanisme de remontée fournit au niveau n-1 l'unité textuelle S, avec un seul terme associé, T_2. La pondération de T_2 pour S est (0.8 * 2 + 0.3 * 3) / 5 = 0.5

Donc, S = { (T_2, 0.5) }

Pour une requête R = T_2 ayant comme portée d'évaluation ULD, la liste de réponses est :

{ (S, T_2, 0.5), (U,T_2, 0.8) (V, T_2, 0.3)}

La liste ainsi formée est donc la réponse à notre requête, et elle est passée aux autres modules du SIB pour constituer la réponse finale à la clause **SELECT** correspondante.

L'évaluation d'un énoncé de recherche sémantique utilisant la RID est, évidemment, plus coûteuse qu'en utilisant une RI complète. Le temps de traitement est plus important en raison de la complexité considérable des traitements à effectuer. En fait, il faut traiter l'ensemble de termes d'indexation de chaque document pour faire le calcul dynamique des pondérations.

Par contre, l'index sémantique constitué par une RID est beaucoup moins coûteux en place qu'un index sémantique constitué par une RI complètement instantiée.

La différence fondamentale entre les deux alternatives pour l'établissement de l'index sémantique dans le SIB est donnée par le degré de généralité qu'on veut apporter au SIB. La RI est utilisable par la plupart des approches de SRI, la RID impose le critère d'évaluation, mais les opérations réalisés dans le SIB offrent plus de sémantique.

Pour stocker la RID nous adoptons des méthodes de compression similaires à celles définies auparavant pour la Signature de la Relation d'Indexation

4.5. Conclusions

Dans ce chapitre nous avons défini deux opérations de recherche par le contenu des documents. La **RECHERCHE TEXTUELLE,** qui tient compte uniquement du contenu textuel des documents, constitué des chaînes de caractères qui le composent. La **RECHERCHE SÉMANTIQUE,** qui est fondée sur une représentation du contenu sémantique du document, et qui utilise une relation d'indexation qui peut leur être associée à la demande. Nous avons ensuite défini les méthodes d'accès et de stockage que nous avons considérées comme les plus appropriées à chacune d'entre elles, en tenant compte des exigences propres à un serveur d'information bureautique : facilité de mises à jour, corpus documentaire dynamique, espace de stockage limité et réponses en temps réel.

Nous avons défini des structures d'indexation du contenu des documents pour chaque type de recherche. Pour la recherche textuelle, l'index est calculé de manière systématique pour tous les documents du serveur au moment de leur internalisation. Pour la recherche sémantique l'index n'est créé que pour les documents pour lesquels il a été demandé ; le serveur stocke les relations d'indexation définies par les applications.

Nous avons adopté les méthodes de signatures comme stratégie de stockage de ces index. Nous avons adapté la méthode de codage superposé pour le calcul de l'index textuel, en tenant compte de la structuration logique des documents et des expressions de filtrage que nous avons définies pour le SIB. Pour l'index sémantique, nous avons défini la signature de la relation d'indexation par l'association des termes par unités d'indexation. Celles-ci correspondent aux différents niveaux dans la structure logique du document. Nous avons défini dans ce contexte la signature d'une unité d'indexation comme un séquence de signatures individuelles de termes d'indexation pondérés.

Les méthodes d'accès ainsi définies restent dépendantes de la taille du corpus documentaire, la taille des index étant

proportionnelle au nombre d'éléments logiques et de termes d'indexation.

Nous présentons les différents alternatives d'implantation de ces fonctions d'accès par le contenu dans le chapitre suivant.

5 . IMPLANTATION DES OPERATIONS DE RECHERCHE PAR LE CONTENU

5.1. Introduction

L'évaluation d'un énoncé de recherche textuel implique deux types d'accès : l'accès à l'index textuel pour comparer les SPC et la signature de la requête, et l'accès aux documents eux mêmes pour éliminer les collisions dans l'ensemble de références pré-sélectionnées. L'évaluation d'un énoncé de recherche sémantique implique l'accès aux signatures des unités d'indexation existantes pour retrouver les références pertinentes aux sujets de recherche.

Le temps d'accès reste néanmoins proportionnel à la taille de la base de documents. En fait, la taille de l'index textuel est proportionnelle au nombre de portions de contenu des documents, et celle de l'index sémantique est proportionnelle au nombre d'unités et de termes d'indexation définis. En utilisant SCHUSS, pour le filtrage au vol de données, nous pouvons améliorer les temps d'accès à ces index.

La figure 5.1 montre les principaux modules de traitement que nous avons identifiés. Nous avons plusieurs alternatives de partage de ces tâches entre le Serveur et la machine de filtrage :

Alternatives d'Implantation

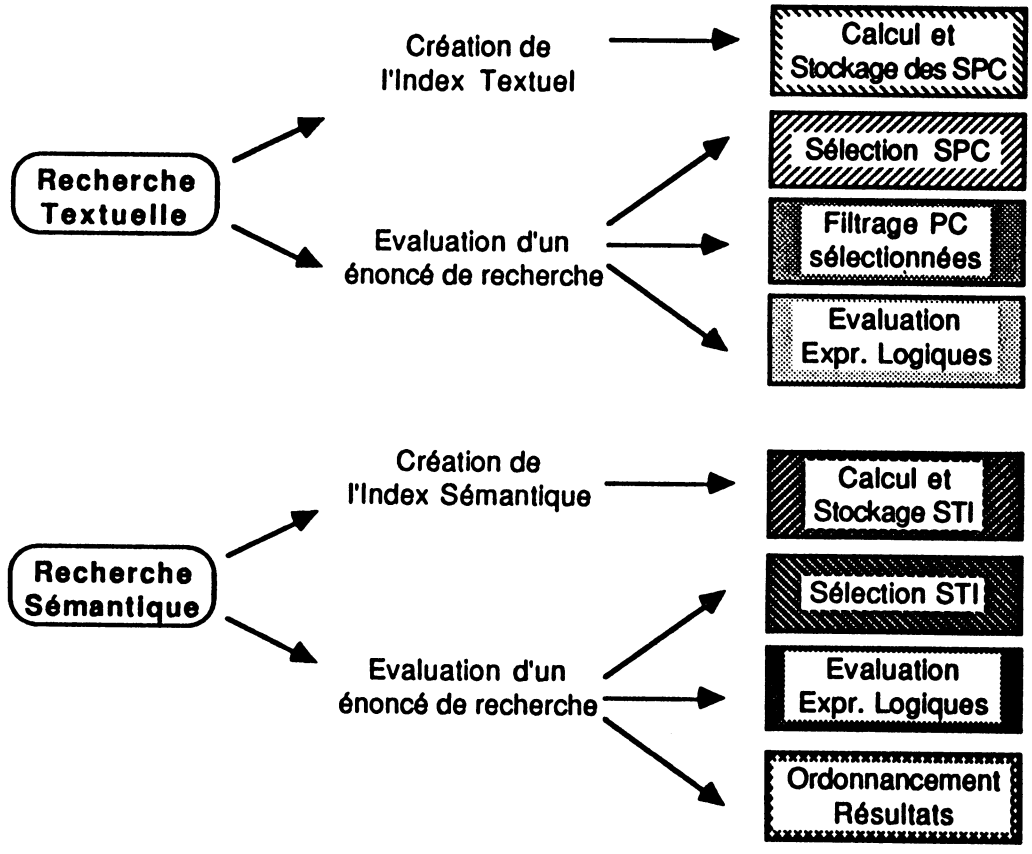


figure 5.1

1. IMPLANTATION PUREMENT LOGICIELLE.

   Dans ce cas, les quatre opérations de chaque type de recherche sont réalisées par le serveur.

2. IMPLANTATION PUREMENT MATÉRIELLE

   Le serveur est déchargé des tâches de recherche par le contenu des documents, et les quatre opérations de chaque type de recherche sont réalisées dans le filtre matériel.

3. IMPLANTATION COMBINÉE LOGICIELLE ET MATÉRIELLE

Les tâches de calcul et de sélection sont partagées entre le serveur et la machine de filtrage. Plusieurs schémas sont possibles. Par exemple, nous pouvons envisager tout le travail de calcul, stockage et sélection des signatures dans le Serveur, et utiliser le filtre matériel uniquement pour les opérations de filtrage du texte des portions de contenu, qui sont les plus coûteuses en temps d'exécution.

Si l'architecture matérielle du filtre le permet, nous pouvons envisager d'effectuer aussi l'évaluation logique des expressions, et même lui assigner des tâches de calcul ou de sélection de signatures.

Le choix entre ces alternatives est fortement dépendant des caractéristiques du matériel à utiliser. Dans notre cas, la machine à considérer est SCHUSS (cf. chapitre 3 et Annexe B), par les facilités de filtrage de données alphanumériques qu'elle offre, ainsi que par la possibilité d'adaptation de l'environnement de filtrage à nos besoins spécifiques. Ceci est possible car SCHUSS offre des facilités de micro-programmation des opérations spécifiques dont nous avons besoin.

De toutes les opérations (cf. figure 5.1), la plus coûteuse en temps d'évaluation est le filtrage des portions de contenu sélectionnées. L'intérêt de sa réalisation par la machine de filtrage nous paraît évident. Les autres opérations qui concernent le module d'évaluation des énoncés de recherche, impliquent des opérations sur des blocs de bits (sélection de signatures) et des tables d'évaluation logique (tables de vérité).

En élargissant l'environnement de SCHUSS, par la microprogrammation de nouvelles micro-instructions de comparaison de bits, il est possible de décharger aussi le serveur des tâches de sélection dans les index. Ceci s'applique tant pour la recherche textuelle que pour la recherche sémantique.

Par contre, les opérations de calcul des signatures, qui impliquent un traitement particulier du contenu des documents (formation de triplets de caractères, calcul des fonctions de transformation, analyse des requêtes) sont plus complexes. Même si les possibilités de calcul du processeur de SCHUSS permettent des opérations arithmétiques, nous considérons que ces opérations, dans un premier temps, peuvent être réalisés par le gestionnaire de documents du serveur, en ne dédiant au filtre matériel que les opérations de sélection.

Nous avons expérimenté deux des alternatives d'implantation : d'une part une implantation complète par logiciel, et d'autre part une implantation combinée entre le serveur et la machine de filtrage. Dans le cas d'implantation mixte, SCHUSS est dédié au filtrage des signatures et au filtrage de texte, ainsi qu'à l'évaluation des expressions logiques. Les opérations de calcul des signatures et de analyse des requêtes sont réalisées dans le serveur. Nous détaillons ces deux schémas d'implantation par la suite.

5.2. Mise en Œuvre Logicielle

L'implantation des opérations de recherche par le contenu se situe à deux niveaux différents dans le SIB : d'une part, il faut donner aux applications, dans le langage de manipulation de données, une forme d'expression des énoncés de recherche. Tout énoncé est analysé et transformé en une requête transmise au Gestionnaire de Documents du serveur, afin de faire sélectionner dans un ensemble de documents, ceux qui répondent au critère de recherche.

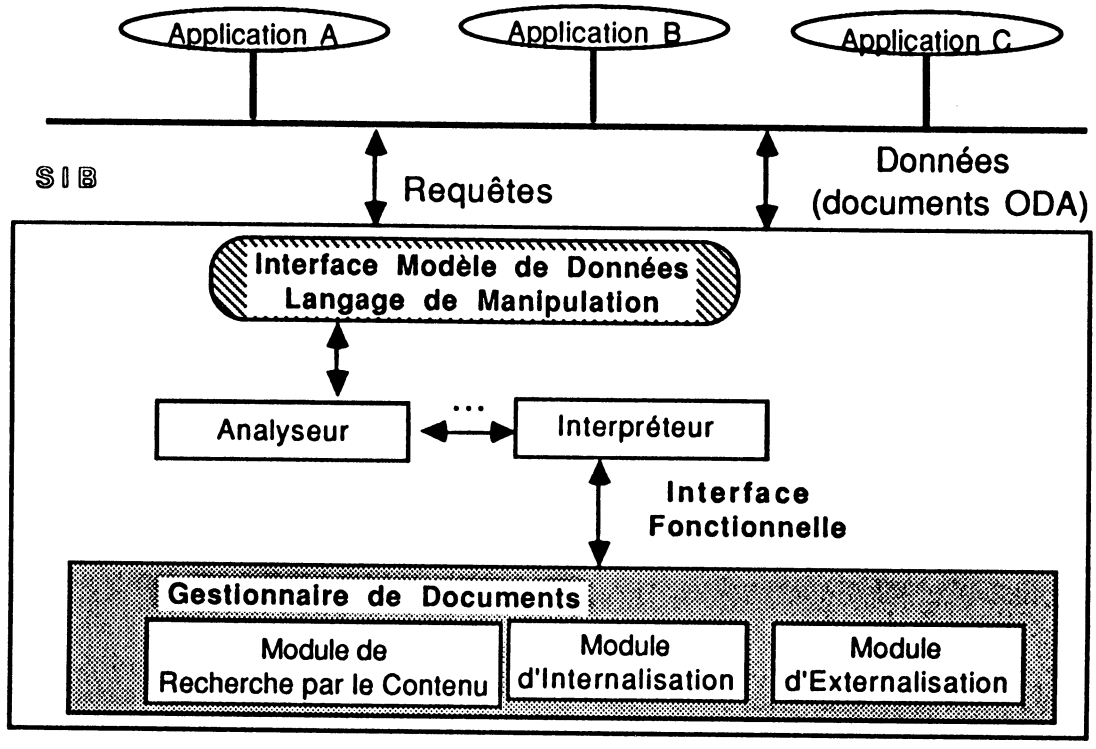


figure 5.2

Nous avons ainsi défini deux niveaux d'interface (voir figure 5.2) : les énoncés au niveau du langage, au niveau interface du SIB, et les fonctions d'accès correspondantes au niveau de l'interface du gestionnaire de documents. A ce niveau se situe aussi la définition des structures de stockage nécessaires à l'exécution des fonctions d'accès.

Nous détaillons par la suite l'implantation des deux opérations de recherche par le contenu que nous avons définies, d'abord en ce qui concerne la recherche textuelle, et ensuite en ce qui concerne la recherche sémantique.

5.2.1. Recherche Textuelle

Nous avons réalisé l'expérimentation de cette alternative sur une machine BULL-SPS7, avec le système d'exploitation SPIX 31.3. Toute les fonctions ont été programmées en C. Dans la section 5.4.1 nous détaillons le volume de programmation que cette réalisation a représenté.

Pour l'implantation de la recherche textuelle sur les documents du SIB nous avons défini, au niveau du modèle de données, une syntaxe permettant d'exprimer les opérations et les objets sur lesquels porte la requête (cf. chapitre 4). Ceci est réalisé par l'introduction de l'opérateur **TALKS ABOUT** dans le langage de manipulation de données.

Au niveau du Gestionnaire de Documents quatre fonctions sont définies : une pour créer l'index textuel d'un document lors de son internalisation, et trois fonctions d'évaluation d'un énoncé de recherche. Ces trois fonctions d'évaluation correspondent en fait à trois formes différentes d'accès pour répondre à une même requête, nous permettant ainsi d'évaluer le comportement de l'utilisation de la méthode de signatures choisie pour l'implantation de l'index textuel.

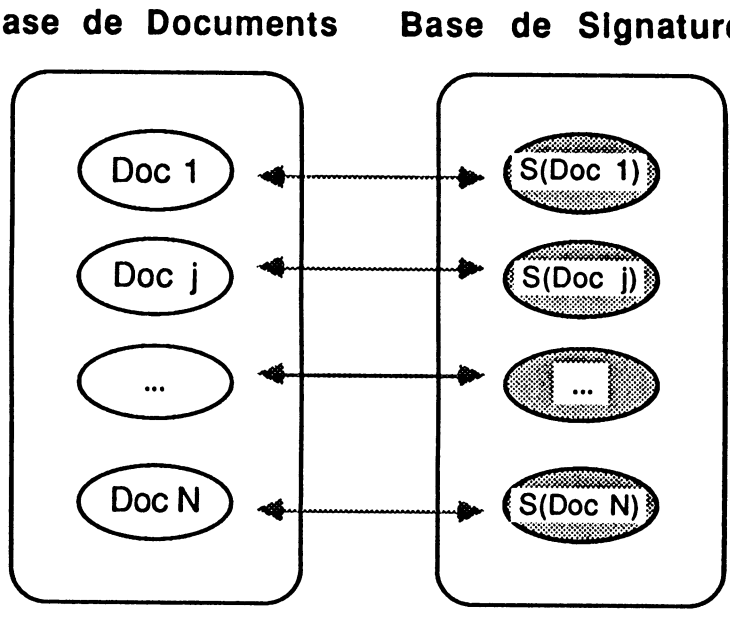
Les fonctions de base dans le SIB sont :

1. **CRÉER_INDEX** (*identificateur_document*)

   Cette fonction nous permet de créer, l'index textuel d'un document dont le contenu est identifié par *identificateur_document*. Ceci peut correspondre, par exemple, au nom du fichier dans lequel se trouve le document.

   Cette fonction doit être appelée de manière systématique pour tout document qui arrive au SIB. Elle identifie toutes les portions de contenu du document, calcule leur signature et les stocke dans une "base_de_documents". Le stockage des signatures (qui constituent donc l'index textuel du document)

est effectué dans une "base_de_signatures" associé aux documents. Cette base contient autant de blocs signatures qu'il y a de portions de contenu dans le document. Ceci est illustré par la figure ci-dessous :



L'algorithme général est le suivant :

CRÉER_INDEX (*id_document*)
| Pour toute PC_i dans Contenu (*id_document*)
| SPC_i = signature (PC_i)
| Stocker SPC_i dans IndexTextuel(*id_document*)
fin

2. **FILTRER** (*document, requête*)

Cette fonction nous permet d'évaluer une expression de filtrage *requête* directement sur le **contenu textuel** d'un document identifié par *document,* sans utiliser l'index correspondant. Elle consiste donc en l'exécution de l'automate généré à partir de la requête, sur les portions de

contenu du document et en l'évaluation de l'expression logique correspondant à l'expression de recherche.

L'algorithme général est le suivant :

```
FILTRER (id_document, requête )
|   Analyse_syntaxique ( requête )
|   Génération_Automate ( requête )
|   rés = Exécution_Automate ( requête , id_document )
| si ( rés = VRAI ) return (id_document  est sélectionné)
| si non     return (id_document   n'est pas sélectionné)
fin
```

3. **FILTRER_INDEX** (*document, requête*)

Cette fonction nous permet de restreindre l'évaluation de l'expression de filtrage *requête* à l'index textuel associé au contenu du document identifié par *document*, en admettant dans la réponse les collisions dûes à la méthode de signatures. Elle comprend donc le filtrage des signatures des portions de contenu du document et l'évaluation de l'expression logique de l'énoncé de recherche.

Dans l'exécution de cette fonction aucun accès n'est donc fait directement sur le contenu textuel des documents. L'algorithme général est le suivant :

```
FILTRER_INDEX (id_document, requête )
|   Analyse_syntaxique ( requête )
|   Sreq = Génération_SignReq( requête )
|   Liste_PC = Eval_signatures ( Sreq , id_document )
| si ( Liste_PC est vide )
|       return (id_document   n'est pas sélectionné)
| si non  return (id_document  est sélectionné)
fin
```

4. **MATCH** (*document, requête*)

Cette fonction nous permet d'évaluer une expression de filtrage *requête* sur l'index textuel associé au contenu du document identifié par *document*, en effectuant aussi le processus d'élimination des collisions. Elle consiste en un filtrage des signatures des portions de contenu du document et en une évaluation de l'expression logique de l'énoncé de recherche, puis en l'exécution de l'automate correspondant sur l'ensemble de portions de contenu pré-sélectionnées. Elle correspond donc à une combinaison des deux fonction précédentes : **FILTRER_INDEX**, puis **FILTRER**.

L'algorithme général est le suivant pour le traitement d'un document :

```
MATCH (document, requête)
|
|   Analyse_syntaxique ( requête )
|   Sreq = Génération_SignReq( requête )
|   Liste_PC = Eval_signatures ( Sreq , id_document )
|  si ( Liste_PC est vide )
|       return (id_document    n'est pas sélectionné)
|  si non
|     Génération_Automate ( requête )
|     rés = Exécution_Automate ( requête , id_document )
|     si ( rés = VRAI )
|     |    return (id_document   est sélectionné)
|     si non
|     |    return (id_document   n'est pas sélectionné)
|     fin
|  fin
fin
```

Les fonctions **FILTER_INDEX** et **FILTER** définies auparavant ont un intérêt de comparaison et d'évaluation de la méthode de signatures, mais elles pourraient aussi être visibles au niveau des énoncés de recherche. Ceci implique l'introduction de nouveaux éléments au niveau de l'expression de l'opérateur **TALKS ABOUT**, et par conséquent un élargissement du langage de manipulation de données du SIB. Nous introduisons ainsi au niveau de la syntaxe les opérateurs **NOISY TALKS ABOUT** et **FILTER TALKS ABOUT**, le premier pour un filtrage limité aux signatures, et l'autre pour un filtrage direct du texte.

Un énoncé de la forme :

SELECT < doc >.<attribut>
WHERE < doc >.content **NOISY TALKS ABOUT** <expression_de_filtrage >

est traduit comme un appel de fonction :

FILTRER_INDEX (*document, requête*)

dont le résultat indique si le document vérifie au niveau de l'index textuel, l'expression de filtrage donnée dans *requête*. Cette réponse peut, par conséquent, contenir des collisions, mais son évaluation est sensiblement plus rapide.

Un énoncé de la forme :

SELECT < doc >.<attribut>
WHERE < doc>.content **FILTER TALKS ABOUT** <expression_de_filtrage >

est traduit comme un appel de fonction :

FILTRER (*document, requête*)

dont le résultat indique si le document vérifie l'expression de filtrage donnée dans *requête* par examen direct du texte, sans passer par l'index.

5.2.1.1. Calcul de la Signature d'une Portion de Contenu

Le calcul de la signature d'une portion de contenu, tel qu'il est décrit dans le chapitre 4, comporte plusieurs étapes :

- Identification des triplets de caractères:

  La formation des triplets de caractères tient compte de tous les mots dans la portion de contenu. Les caractères séparateurs entre mots sont traduits vers un seul caractère séparateur générique. Un séparateur est introduit au début et à la fin de la portion de contenu. Les triplets sont formés sur les caractères consécutifs dans le texte, comme il a été montré dans le chapitre 2.

- Application de la fonction de hachage :

  Pour chaque triplet, une fonction de transformation est appliquée de manière à lui faire correspondre une position dans un bloc de bits de taille N, pour un N choisi en fonction des longueurs typiques des portions de contenu des documents traités (cf. chapitre 4).

  Tous les caractères sont convertis en majuscules non-accentuées. Ceci garantit une même image après la transformation, en permettant de gérer à ce niveau la formation de collisions. Par exemple, les mots "*Evaluation*" et "*évaluation*" sont transformés de manière équivalente, et lors de l'interrogation seul les triplets "_EV" et "EVA" sont à vérifier. En fait, ceci est équivalent à une réduction contrôlée du nombre possible de triplets et de collisions.

  Pour le calcul de la fonction de transformation nous utilisons les algorithmes polynomiaux et de division présentés dans [Knu73], à partir de la représentation hexadécimale de chaque caractère : chaque triplet est

considéré comme un chiffre d'un nombre en base 127 (2^7-1), puis il est transformé en base décimale et transformé enfin en un entier dans l'intervalle [0 .. N-1] en calculant son modulo à N. Le chiffre ainsi obtenu est l'image du triplet, c'est-à-dire, la signature du triplet.

La fonction ainsi calculée garantit que la transposition de caractères (par exemple les triplets "aru" et "aur") produit des images différentes.

Exemple :

Pour une taille de bloc N = 199 bits,

l'image du triplet "abc" est 1056834 mod 199 = 144

et l'image du triplet "bca" est 1073088 mod 199 = 80

- Formation du Bloc de Signatures :

Le bloc signature d'une portion de contenu est initialisé à "0", et pour chaque triplet, le bit correspondant à la signature de ce triplet est mis à "1". Ceci constitue, en fait, le OR des signatures des triplets individuels.

Dans l'exemple précédent, les bits des positions 80 et 144 dans le bloc de signatures ont alors la valeur "1".

La signature ainsi formée est prête à être stockée et filtrée lors de l'évaluation des expressions de recherche.

5.2.1.2. La Prise en Compte de la Structure du Document

La structure logique du document est prise en compte à deux moments : lors de la création de l'index textuel du document, par l'identification des portions de contenu et lors de l'interrogation, par la définition des chemins d'accès aux éléments logiques, par exemple, les chapitres ou les sections dans des documents de type *livre*.

La définition des chemins d'accès est faite lors de la traduction d'un énoncé **TALKS ABOUT** (ou **NOISY TALKS ABOUT** ou **FILTER TALKS ABOUT**), par la correspondance entre le nom donné aux éléments de la structure logique au niveau du modèle de données (ULD), et l'identificateur interne de ces éléments dans le contenu du document. Cet identificateur interne permet de délimiter l'ensemble de portions de contenu concernées par la requête.

Une manière d'identifier facilement cet ensemble, sans avoir besoin de parcourir les arborescences de la structure logique des documents, est l'utilisation de la notation Dewitt pour l'identification interne des portions de contenu. Par exemple, si les chapitres se trouvent au niveau 2 de la structure logique, les portions de contenu identifiées par 1.3.2.1 et 1.3.3.1 font toutes les deux partie du troisième chapitre. Dans les documents ODA, les portions de contenu sont identifiées suivant ce même principe.

La prise en compte dans le SIB de la structure logique des documents implique deux aspects à considérer : d'une part l'introduction dans le modèle de données d'une syntaxe permettant de nommer ces éléments, et d'autre part l'introduction des fonctions de correspondance entre ces noms des éléments et les éléments présents dans les documents ODA. Ceci fait l'objet d'un autre travail [LG 89] mené en parallèle avec le nôtre.

Dans le cas général, un énoncé de recherche textuelle (cf. chapitre 4), de la forme suivante :

SELECT < doc >.<attribut>

WHERE <doc>.ULD **TALKS ABOUT** <expression_de_filtrage>

il est traduit comme un appel la de fonction :

MATCH_ULD (*document, ULD, expression_de_filtrage*)

le couple *(document, ULD)* étant la forme interne au SIB qui permet d'identifier les éléments logiques de niveau ULD dans les documents <doc>. La valeur de cette fonction "vrai" ou "faux", est calculée en fonction de l'évaluation de l'expression de filtrage dans l'ensemble des signatures et des portions de contenu appartenant à chaque ULD concerné. Cette fonction comprend l'étape d'élimination des collisions.

De manière similaire sont définies et traduites les fonctions **FILTER_ULD** et **FILTER_INDEX_ULD.**

5.2.1.3. Les Portions de Contenu

Une définition de la structure d'une Portion de Contenu est nécessaire, aussi bien pour la création de l'index textuel que pour les opérations de filtrage de texte.

Une portion de contenu est constituée par un identificateur qui permet la reconstruction du chemin d'accès dans la structure logique du document, et par le texte correspondant. Le texte peut être délimité de deux manières : soit par une position de début dans le document et un marqueur de fin, soit par une position de début dans le document et le nombre de caractères qui le composent. L'identificateur de la portion de contenu sert à sélectionner, pour une requête donnée, l'ensemble de portions sur lesquelles porte l'évaluation, c'est à dire, l'ensemble de PC associées à ULD (cf. chapitre 4).

Pour délimiter le texte, le choix entre les deux manières décrites dépend uniquement du formattage du contenu des documents. Nous avons retenu, pour le cas général, de délimiter une portion de contenu par une position de début dans le document et un marqueur de fin. Néanmoins, la forme la plus naturelle pour le cas de documents ODA codées en ODIF, est l'utilisation de la position de début dans le contenu et de sa longueur. Pour le cas d'intégration au traitement des documents ODA nous retenons cette deuxième solution (cf. section 5.3). Le choix entre ces deux formes de délimitation du texte est en réalité un détail qui détermine simplement une réalisation particulière des opérations de parcours de la portion de contenu.

5.2.1.4. Stockage et Sélection des Signatures

Les signatures des portions de contenu sont stockées dans une "base de signatures", qui contient, pour chaque portion de contenu dans le document, l'identificateur de portion de contenu et son bloc de signatures.

Lors de la sélection, la signature de la requête est générée (cf. chapitre 4), et pour tous les ULD concernées, chaque SPC est comparée avec tous les blocs de signature de la requête. Si la comparaison est positive (l'ensemble de bits du bloc de signature de la requête est un sous-ensemble de l'ensemble de bits de la signature de la portion de contenu), l'identificateur de cette PC est inclus dans une liste de blocs associée au terme correspondant dans la requête .

A la fin du filtrage des signatures de chaque ULD, les listes associées à chaque terme de la requête sont passées à l'étape de vérification des opérateurs logiques. Ceci signifie des opérations d'union et d'intersection de ces listes. La réponse à la requête à ce stade de l'évaluation (filtrage des signatures), pour chaque ULD, est constituée par la liste finale d'identificateurs des PC pré-sélectionnées.

Si l'on accepte les collisions, les ULD sélectionnés sont ceux pour lesquels la liste finale contient au moins une PC pré-sélectionnée.

Si l'élimination des collisions est demandée, nous passons à l'étape de filtrage du texte des PC. Ce filtrage est réalisé sur les ULD pour lesqueles au moins une PC a été pré-sélectionnée. Pour chaque ULD, le filtrage de texte portera seulement sur l'ensemble de portions de contenu dont les identificateurs se trouvent dans la liste. L'ULD est finalement donnée comme réponse à la requête si dans l'étape de filtrage de texte l'expression logique est vérifiée.

5.2.1.5. Filtrage de Portions de Contenu

Le filtrage des PC est effectué par l'exécution de l'automate de filtrage correspondant à la requête (cf. chapitre 4). Cet automate est physiquement représenté comme une matrice, ayant dans les lignes les états et leur description, et dans les colonnes les caractères du vocabulaire de la requête. La description d'un état indique s'il est non-accepteur ou accepteur, et dans ce dernier cas, de quelles chaînes. Une structure auxiliaire contient la la *mémoire* de l'ordre de reconnaissance des sous-chaînes, ainsi que l'information nécessaire à l'évaluation des expressions logiques.

5.2.1.6. Evaluation des Expressions Logiques

L'évaluation des expressions logiques se fait à deux niveaux : d'abord lors du filtrage des signatures (cf. 5.4.1.4), puis pendant le filtrage des portions de contenu (exécution de l'automate de filtrage). Nous avons inclus cette seconde évaluation logique dans l'exécution de l'automate pour des raisons de performance, afin de pouvoir décider, éventuellement, la terminaison du processus de filtrage de texte avant la fin de la vérification complète des

portions de contenu : l'analyse termine dès que l'expression logique a pu être complètement évaluée. Ceci est vérifié d'après l'état courant de la structure de mémoire de reconnaissance (cf. chapitre 4).

5.2.2. Recherche Sémantique

Les données nécessaires à la création de l'index sémantique d'un document dans le SIB sont des triplets contenant un terme d'indexation, sa pondération et la référence à l'unité textuelle correspondante (cf. chapitre 4). Pour intégrer ces données au serveur (étant donné qu'elles proviennent de l'application), nous avons choisi un stratégie similaire à celle de l'obtention des documents. La RI d'un document est donc fournie au SIB dans un fichier contenant les triplets qui lui sont associés. Les références aux unités textuelles ont une structure similaire à celle de l'identificateur des portions de contenu, c'est-à-dire, une notation Dewitt permettant de retrouver le chemin dans la structure logique.

L'énoncé d'internalisation de la RI d'un document est de la forme :

INSERT < doc >.RI
FROM < doc > .< attribut > = ...
WITH < doc >.RI = <"nom_de_fichier">

Cet énoncé n'est applicable qu'aux documents < doc > qui ont été déclarés **INDEXED** (cf. chapitre 4). Il est traduit vers une fonction qui prend les triplets du fichier, calcule la signature des termes d'indexation et associe les triplets référence, afin de former la signature de chaque unité textuelle une par une. Chaque signature d'unité textuelle est associée à l'élément correspondant de la structure logique du document, et elle est stockée. La RI est ainsi vue est traitée comme un attribut quelconque des éléments logiques du document.

5.2.2.1. Calcul de la Signature des Termes d'Indexation

Le calcul de la signature des termes d'indexation, comme celui des signatures des portions de contenu, comporte trois étapes, à savoir : formation des triplets de caractères dans le terme d'indexation, application de la fonction de transformation, et formation du bloc de signature par la superposition de bits correspondants aux images des triplets de caractères. Nous pouvons utiliser le même algorithme de calcul de la fonction de hachage pour une taille de bloc de bits calculée en fonction de la longueur typique des termes d'indexation (cf. chapitre 4). La signature de l'unité textuelle est formée par la juxtaposition des signatures individuelles des termes d'indexation (STI, présentées ci-dessous).

5.2.2.2. Stockage et Sélection des STI

Les signatures des termes d'indexation sont stockées de manière associative par unité textuelle, constituant ainsi ce que l'on appelle la *signature d'une unité textuelle* (cf. chapitre 4). La liste de triplets composant la RI du document est ordonnée par identificateur d'unité textuelle (c'est-à-dire, l'identificateur de l'élément logique du document), les signatures des termes d'indexation sont calculées de la manière habituelle, et l'on stocke, pour chaque unité textuelle, la suite de STI et de poids associés (cf. figure 4.6).

Pour l'évaluation d'un énoncé de recherche sémantique, seul l'ensemble de signatures des unités d'indexation est accédé. La sélection est réalisée selon le principe décrit dans le chapitre 4.3.2. Le fait d'admettre ces collisions dans l'ensemble de réponses fait diminuer la *précision* (cf. chapitre 2 et 4) de l'évaluation de la *fonction de correspondance* "terme-document" pour l'application. Si la taille des signatures des termes d'indexation est suffisamment grande pour avoir un taux de collisions très faible, le fait d'admettre les collisions n'a pas

des répercussions fondamentales sur le comportement de la fonction d'interrogation du SRI en question.

Comme c'est le cas pour la recherche textuelle, la notation Dewitt pour les identificateurs permet une évaluation rapide des unités textuelles concernées par une requête.

5.2.2.3. Evaluation des Expressions Logiques

L'évaluation des expressions logiques qui constituent le thème de recherche se fait à partir de l'ensemble de références des unités textuelles sélectionnées. Les références sont formées par le couple (identificateur du document, identificateur de l'élément logique correspondant). Comme dans le cas de la recherche textuelle, cette opération comporte des unions et des intersections des listes de références. La réponse est constituée par la liste des triplets (référence, terme, poids) sélectionnés, selon les principes de sélection définis en 4.3.2 et 4.4.2.2.

Par exemple :

Si la requête est "T1 AND T2",

et l'unité textuelle U et V contient les termes :

$U = \{ (T_0, p_0), (T_1,p_1), (T_2,p_2), (T_3, p_3) \}$

$V = \{ (T_4, p_4), (T_1,p_6), (T_2,p_7), (T_5, p_5) \}$

la réponse est constituée par la liste :

$\{ (U, T_1,p_1), (U, T_2,p_2), (V, T_1,p_6), (V, T_2,p_7) \}$

Dans un SRI la réponse à une requête est de la forme (*réf_unité_textuelle, pondération*), où la pondération est calculée en fonction des poids des termes d'indexation retrouvés

dans cette unité. Cette pondération globale donne la mesure de la *correspondance* entre le résultat et la requête (cf. chapitre 2). Dans l'exemple, elle serait de la forme

$$\{ (U, f(p_1,p_2)), (V, f(p_6,p_7)) \}.$$

Nous avons décidé de rendre la liste complète de références (le sous-arbre de la RI satisfaissant la requête), sans aucune interprétation additionnelle, afin de laisser à l'application le choix d'y appliquer ses propres critères de pondération (c'est à dire, la fonction *f*). Par exemple, une application peut établir des évaluations basées sur des logiques floues et une autre application décider de les calculer simplement sur des moyennes.

La liste de réponses constituée par des triplets nous permet de garder aussi une homogénéité du point de vue de l'interface du SIB au niveau du modèle de données. En fait, le format des réponses aux énoncés de recherche sémantique est similaire à celui des données obtenues de l'application au moment d'internaliser la relation d'indexation des documents.

5.2.2.4. Ordonnancement des Résultats

La liste finale de références, produite par l'étape d'évaluation de l'expression logique, est ordonnée en gardant l'ordre des références dans la structure logique. Ceci permet de passer au module d'interrogation de l'application l'information nécessaire pour évaluer facilement ces résultats en fonction des éventuels seuils sur les pondérations et guider d'éventuelles reformulations de la requête. C'est après ce processus d'évaluation que les autre modules du SIB continuent l'évaluation de l'énoncé **SELECT**.

5.2.3. Intégration au Serveur OIS

L'intégration sur un serveur particulier des opérations de recherche par le contenu que nous avons définies, comprend deux aspects : d'une part l'intégration des opérations au niveau du modèle de données et d'autre part la réalisation de ces opérations au niveau des fonctions et des structures de stockage nécessaires (cf. figure 5.2).

L'intégration au niveau du modèle de données implique une syntaxe de définition de la structure logique des documents, ainsi qu'une manière de définir la relation d'indexation de l'ensemble de documents pour lesquels elle peut exister.

Les critères de stockage et les algorithmes d'accès nécessaires à la réalisation des opérations de recherche par le contenu, sont définis de manière à permettre un fonctionnement performant à tous les niveaux du gestionnaire de documents du serveur en question.

Nous présentons par la suite l'intégration que nous avons réalisée sur le serveur OIS (cf. chapitre 1). Nous présentons d'abord de manière générale les problèmes posés par l'intégration dans le gestionnaire de documents de l'OIS. Nous détaillons ensuite les fonctions et les structures de stockage qui interviennent dans cette intégration.

Le serveur OIS a été développé aussi sur la machine BULL-SPS7, avec le système d'exploitation SPIX 31.3. Il est construit comme un frontal du SBGD relationnel ORACLE, qui est utilisé essentiellement comme gestionnaire de stockage. Comme toutes les données de l'OIS, les documents ODA sont stockées sur des relations ORACLE. L'ensemble de fonctions pour la recherche par le contenu textuel de documents sur ces documents est écrit en C. Le volume de programmation qu'elles représentent est détaillé dans 5.4.1.

5.2.3.1. Intégration au Gestionnaire de Documents de l'OIS

Le gestionnaire de documents de l'OIS traite des document ODA, en intégrant ses éléments au modèle de données FM. Dans l'Annexe A est décrite la modélisation des documents dans l'OIS par rapport aux autres types d'entités et aux langages de manipulation et d'interrogation. En ce qui concerne les gestionnaire de documents de l'OIS, nous trouvons quatre fonctions principales :

- Internalisation des documents ODA.
- Externalisation de documents ODA.
- Sélection de documents par les attributs du profil.
- Sélection de documents par leur contenu textuel.

Les tâches qui concernent une opération de recherche par le contenu textuel (cf. figure 5.1) interviennent dans deux de ces fonctions : la création de l'index textuel est réalisée comme une partie des opérations d'internalisation du document, et l'évaluation d'un énoncé de recherche est faite par la sélection de documents par leur contenu textuel.

Nous décrivons ici l'intégration qui concerne l'opération de recherche textuelle. L'intégration des opérations pour réaliser la recherche sémantique n'a pas été réalisée dans la version actuelle du prototype, et fait partie des extensions en cours.

Lors de l'internalisation du document, un processus de décodage permet d'identifier tous les éléments de la structure logique du document, et en particulier les portions de contenu. Pour chacune des portions de contenu sa signature est calculée et stockée. Ceci est fait pour tous les documents.

La fonction de base du SIB **CRÉER_INDEX** (*id_document*) décrite auparavant, dévient dans l'OIS un ensemble d'instructions de la forme :

```
INTERNALISER_DOCUMENT ("nom_fichier_ODIF")
|    Obtenir (identificateur_document)
|    Identifier_attributs_profil (identificateur_document)
|    Jusqu'à la fin du fichier ODIF faire :
|         i = IDENTIFIER_PC
|         CRÉER_SIGNATURE (i)
|         STOCKER_SIGNATURE (i)
fin_internaliser
```

L'intégration de la recherche textuelle implique l'introduction dans FML (cf. Annexe A) d'une syntaxe pour exprimer cette opération. Nous avons introduit ainsi l'opérateur **TALKS ABOUT** dans la clause **WHERE** des énoncés de sélection en FML, avec une syntaxe similaire à celle décrite pour le SIB. La tâche d'évaluation d'un énoncé de recherche consiste donc en l'interprétation d'un énoncé de sélection FML où la clause **TALKS ABOUT** est présente. Dans ce cas l'interprète de l'OIS construit l'ensemble d'identificateurs de documents concernés par la requête et pour chacun d'entre-eux l'expression de filtrage est évaluée par un appel de la fonction

MATCH (*identificateur_document, expression_de_filtrage)*

qui correspond à une des fonctions de base définies pour le SIB. Le résultat de cette fonction est "vrai" si le document vérifie l'expression de filtrage, "non" s'il ne la vérifie pas, ou "code_d'erreur" si il y a eu des problèmes dans l'exécution.

La différence entre la réalisation de la fonction **MATCH** dans le cas général du SIB et le cas particulier de l'OIS se situe au niveau de la façon d'accéder les données : dans le SIB les données se trouvent dans un fichier de texte normal, et leur obtention est réalisée par des opérations d'entrée/sortie Unix. Dans l'OIS les

données se trouvent sur ORACLE et leur obtention est réalisée par des opérations de sélection en SQL. Dans le SIB les portions de contenu sont délimités par leur début et un marqueur de fin, dans l'OIS elles sont délimitées par leur début et leur longueur.

Nous avons voulu rendre visible au niveau du modèle de données de l'OIS le fait qu'il existe un index textuel sur les documents. Comme nous l'avons déjà indiqué, l'application peut l'utiliser en fonction de ses besoins. Par exemple, elle peut décider d'accepter le bruit généré par les méthodes de signatures et de ne pas effectuer de filtrage du texte. De la même façon, elle peut décider de réaliser l'accès au contenu textuel sans passer par l'index de signatures du document.

Nous avons donc introduit dans FML l'énoncé de recherche **NOISY TALKS ABOUT**, qui est traduit, de manière équivalente à un énoncé **TALKS ABOUT**, vers un appel de la fonction

MATCH_SIGN*(ident_document,expression_filtrage)*

Cette fonction correspond à la fonction de base **FILTRER_INDEX** définie pour le SIB. Leur différence est, comme dans le cas précédent, au niveau des opérations d'obtenion de données.

De la même façon, nous avons introduit dans FML l'énoncé de recherche **FILTER TALKS ABOUT**, qui est traduit vers un appel de la fonction

FILTER*(ident_document,expression_filtrage)*

Cette fonction correspond à la fonction de base **FILTRER** définie pour le SIB, les différences étant les mêmes que dans le cas des fonctions **MATCH_SIGN** et **FILTRER_INDEX.**

Pour permettre aussi la recherche textuelle sur les résumés des documents, un processus de décodage et de stockage similaire est défini sur les résumés. Au niveau FML, ceci est traduit par la

possibilité de spécifier <doc>.abstract dans la clause **WHERE** correspondant à un énoncé **TALKS ABOUT**, **NOISY TALKS ABOUT** ou **FILTER TALKS ABOUT**.

5.2.3.2. Les Structures de Stockage

Le module d'internalisation d'un document réalise le décodage du format ODIF (norme de codage des documents ODA) et le stockage de manière à permettre les opérations de sélection et d'externalisation des documents. Nous avons trois alternatives de stockage de cette information :

- Stocker le document tel quel et en extraire l'information pertinente lors de chaque opération de sélection. C'est la solution la plus rapide pour l'acquisition et l'externalisation des documents, mais elle pénalise l'exécution des opérations de sélection.

- Stocker le document décodé : toutes les informations contenues dans le fichier ODIF sont traduites vers des structures de stockage propres à l'OIS de manière à favoriser les opérations de sélection. Cette solution évite les décodages multiples lors de l'interrogation, mais elle a l'inconvénient de pénaliser l'opération d'externalisation, pour laquelle il faudrait ré-coder le contenu du document en ODIF.

- Stocker à la fois le document original sous forme codée en ODIF et sous forme décodée (on extrait au moment de l'internalisation les informations nécessaires aux opérations de sélection). Ceci implique une duplication d'information, dont l'intégrité doit être garantie. L'externalisation du document se limite à la re-transmission vers l'application du codage ODIF du document.

Nous avons choisi la dernière de ces options, même si une partie de l'information est dupliquée, afin de minimiser l'interaction avec le codage ODIF. Le document est décodé et son contenu est indexé une fois pour toutes au moment de l'internaliser, et c'est le seul point de liaison avec le modèle ODA. A partir de là, tout le contenu du document est gérable sans faire appel au décodage.

La duplication du contenu complet de la structure logique du document (voir figure 5.3) est très coûteuse en place.

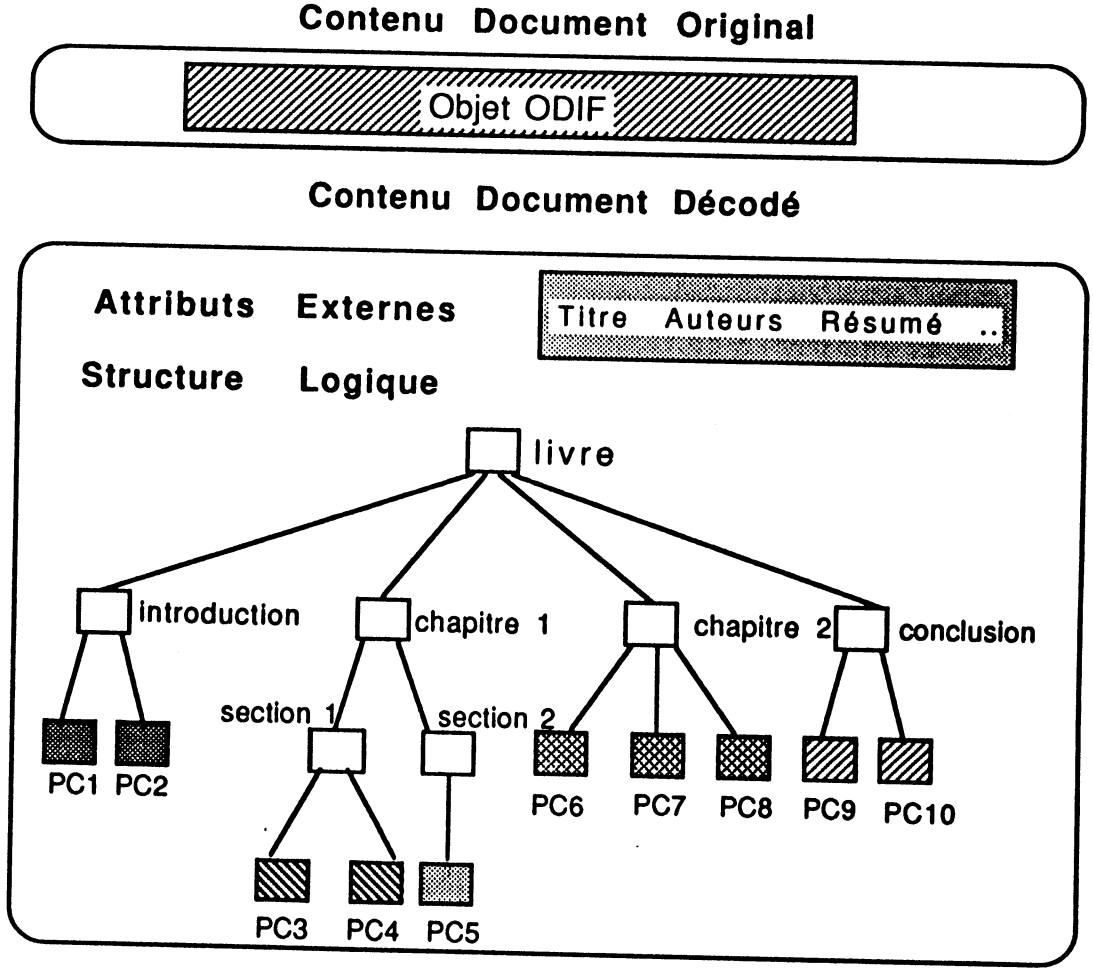


figure 5.3

L'alternative que nous avons choisi consiste à construire une structure auxiliaire décrivant le contenu du document (structure logique, attributs, index) et d'autre part le contenu dans le fichier ODIF original (voir figure 5.4). Cette structure est formée par l'arbre logique du document, les noms des éléments logiques et leurs identificateurs, et des "pointeurs" vers le contenu même des portions de contenu. Les PC ne sont donc pas dupliquées et pour y accéder nous utilisons les "pointeurs" sur l'objet ODIF.

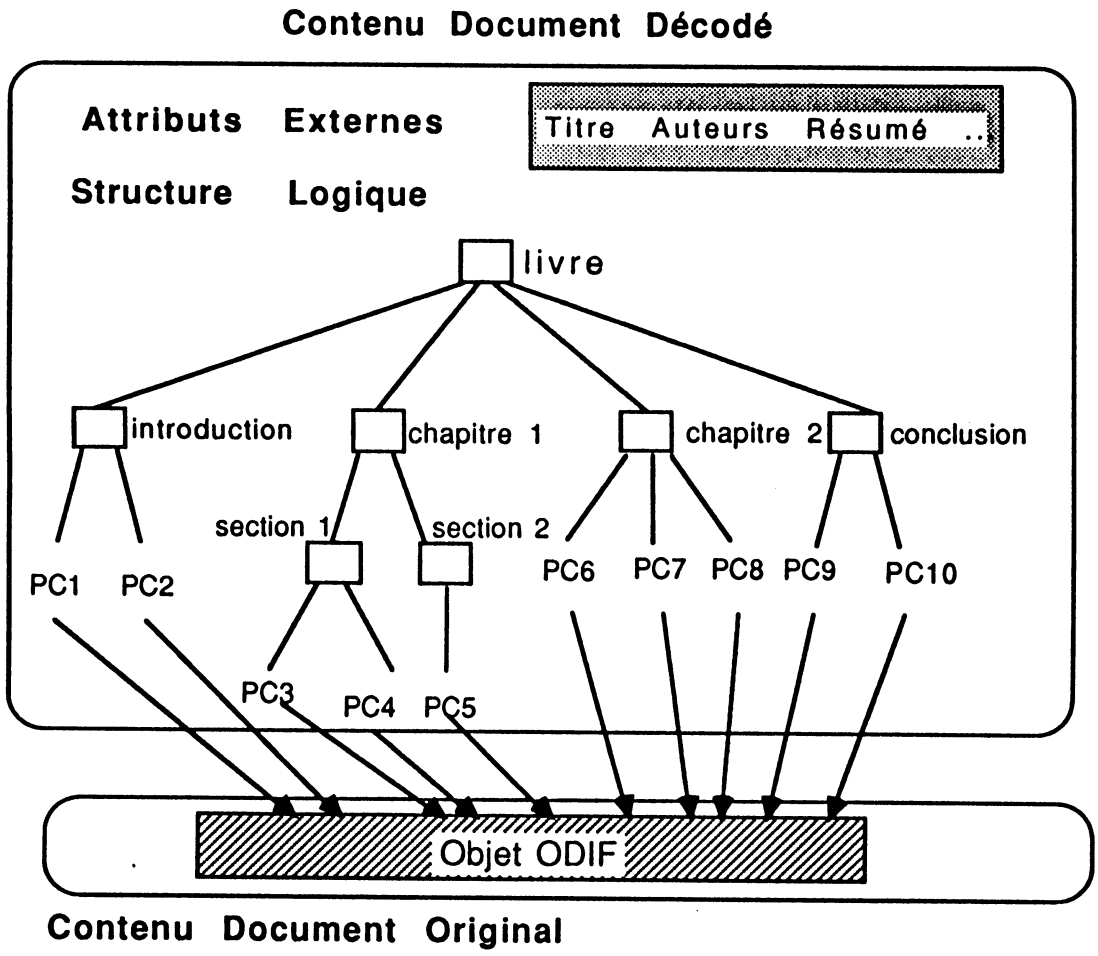


figure 5.4

L'OIS utilise un SGBD relationnel comme serveur de stockage [ESP88a] [Ped88]. Les schémas conceptuels des applications sont traduits vers des relations ORACLE, et les énoncés FM vers des clauses SQL. La représentation relationnelle des structures de stockage dans l'OIS est montrée dans la figure 5.5

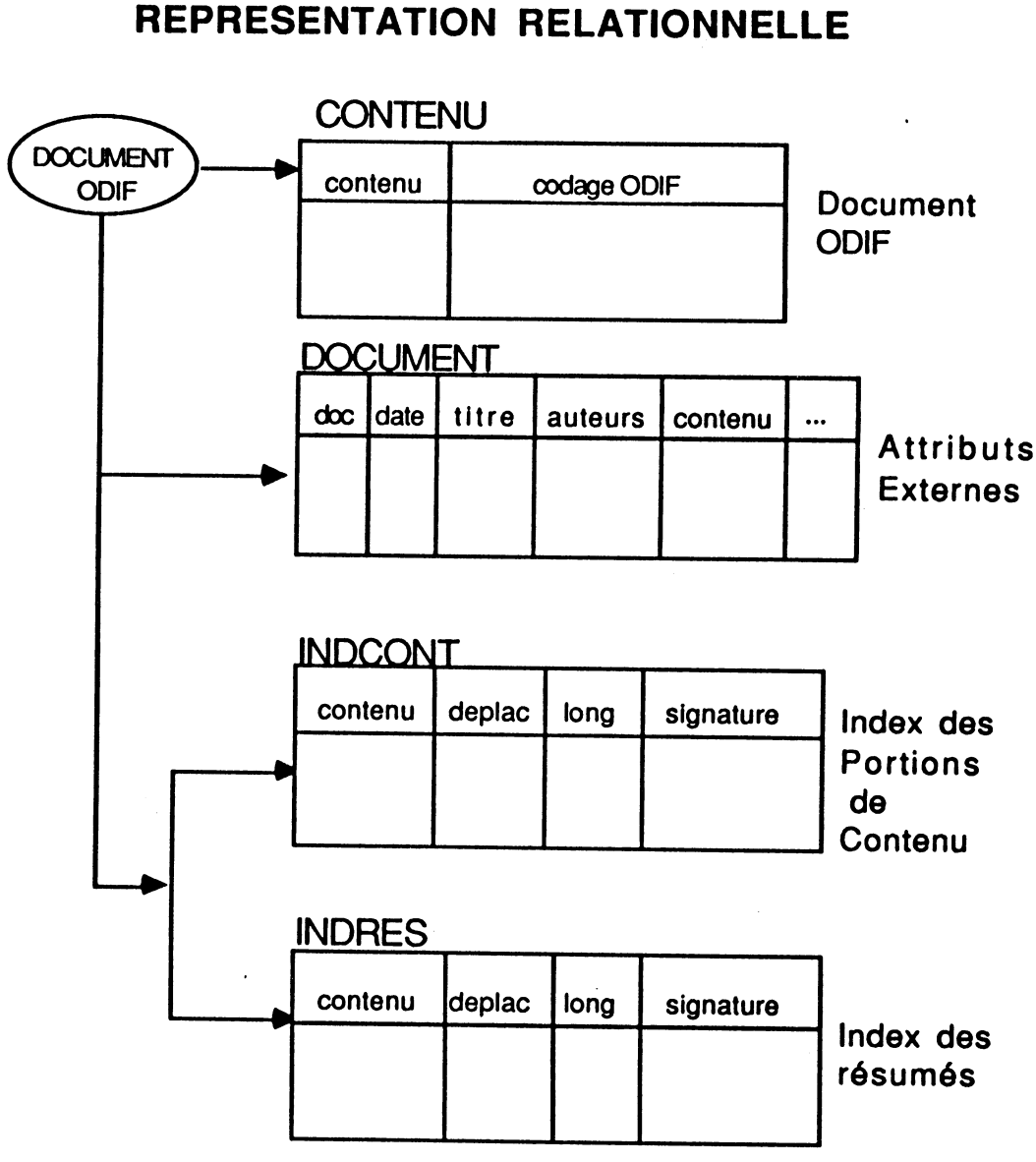


figure 5.5

A partir du document ODIF original est créé un tuple dans la table DOCUMENT, qui contient les attributs du profil. La table INDCONT contient l'index du contenu des documents, où les portions de contenu sont identifiées par leur déplacement et longueur dans le fichier ODIF original, stocké dans la table CONTENU. Dans cette même table est stockée, pour chaque PC, sa signature calculée lors de l'internalisation. Afin de permettre aussi les opérations de recherche textuelle sur les résumés des documents, la table INDRES contient, de manière similaire à INDCONT, le déplacement et longueur du résumé dans l'ODIF, et sa signature.

Une opération **MATCH** sur le contenu d'un document réalise donc un accès d'abord à la table INDCONT (ou INDRES s'il s'agit des résumés), sélectionne les tuples qui décrivent les portions de contenu du document en question, pour chacun vérifie la signature, et construit une liste de couples (déplacement, longueur), correspondant aux PC pré-sélectionnées. Si la liste n'est pas vide, le fichier ODIF est lu dans la table CONTENT, et l'on filtre une à une les portions de contenu de la liste. Dès que l'expression logique est validée, le processus de filtrage de texte se termine. Dans le cas de la fonction **FILTRER_INDEX**, la table CONTENT n'est pas utilisée.

5.2.3.3. Prise en Compte de la Structure Logique des Documents

Dans la première version de l'OIS le seul aspect de la structure logique des documents qui a été prise en compte était l'existence des portions de contenu. Néanmoins, reste à considérer de manière complète la structuration hiérarchique de ces éléments. Ceci fait à l'heure actuelle l'objet d'une étude dans le cadre des extensions par rapport à la version actuelle du prototype. Les structures de stockage et les fonctions d'accès, ainsi que l'intégration des éléments logiques composés au niveau du modèle de données ne sont pas complètement définis. Des propositions à ce sujet se trouvent dans [LG 89].

En ce qui nous concerne, l'évaluation des énoncés de recherche textuelle prennent en compte, non pas toutes les portions de contenu du document, mais les ensembles des portions de contenu associées à chacun des éléments logiques du document impliqués dans la requête, ULD. Ceci se traduit par un appel de la fonction de recherche textuelle **MATCH_SL**, qui doit avoir comme paramètre l'ensemble des identificateurs des PC sur lesquels porte l'évaluation. Les identificateurs sont, comme dans le cas précédent, des couples *(déplacement, longueur)* dans l'objet ODIF.

MATCH_SL *(ident_document, liste_PC, expression_logique)*

Cette fonction est équivalante à la fonction de base du SIB **MATCH_ULD**, avec les différences habituelles au niveau de l'accès aux données. En plus, la liste des PCs de l'ULD en question est complètement spécifiée lors de l'appel de cette fonction.

Par exemple, pour le document de la figure 5.6, la requête en FML

SELECT livre.auteur
WHERE livre.introduction **TALKS ABOUT** "informatique"

se traduit par un appel de fonction

MATCH_SL (livre$_i$, liste_PC, "informatique")

où liste_PC contient {PC1, PC2} avec PC$_i$ = (déplac$_i$, longueur$_i$)

La requête FML

SELECT livre.auteur
WHERE livre.chapitre **TALKS ABOUT** "informatique"

se traduit par un appel de fonction similaire,

MATCH_SL (livre$_i$, liste_PC, "informatique")

où liste_PC contient { {PC3, PC4, PC5}, {PC6, PC7, PC8} }

Si l'on trouve "informatique" (c'est-à dire, si l'expression de filtrage est validée) soit dans une des portions de contenu {PC3, PC4, PC5}, soit dans l'ensemble {PC6, PC7, PC8}, le document identifié par $livre_i$ est sélectionné.

Livre : document

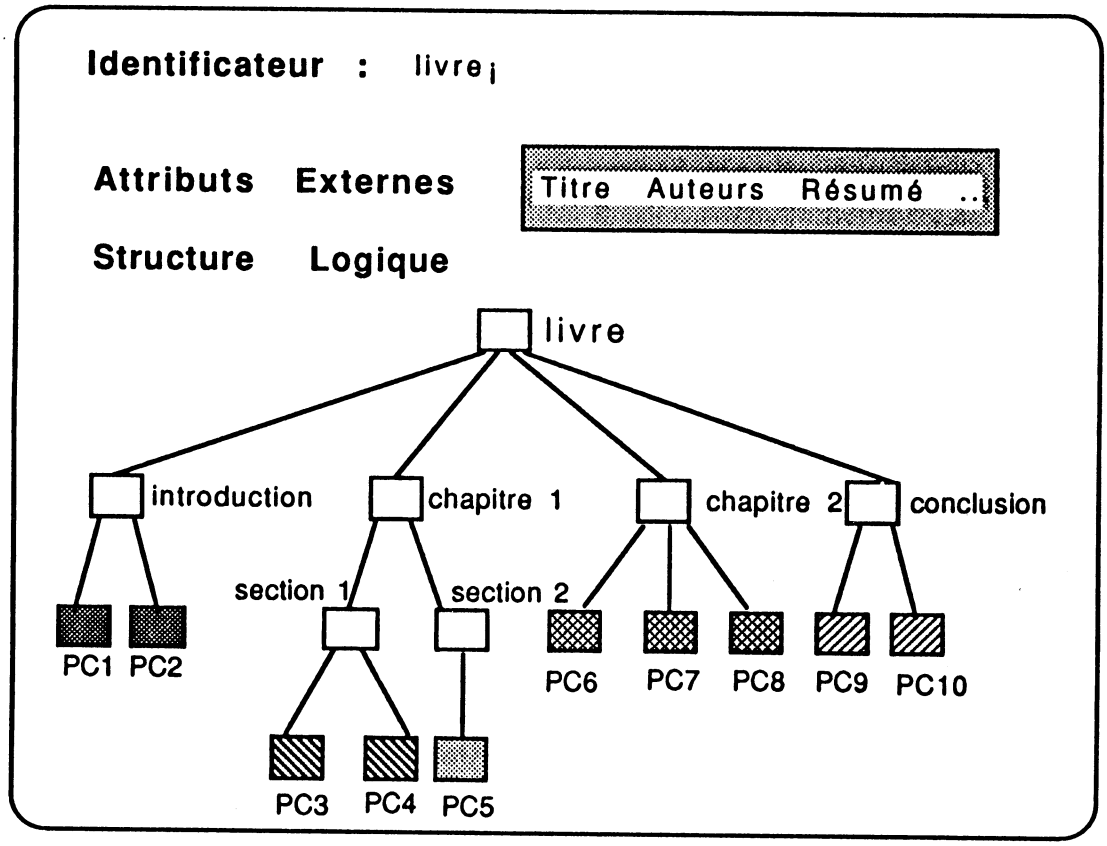


figure 5.6

La liste de portions de contenu est donc un ensemble d'ensembles. Une manière de représenter cette liste d'ensembles est d'utiliser un *numéro_élément*, qui permet de distinguer les PC appartenant à chacun des éléments logiques concernés. Les éléments de la liste sont ainsi des couples (*numéro_élément*, *(déplacement, longueur)*).

Dans l'exemple précédent, la liste de portions de contenu sur lesquelles porte l'évaluation est :

{(1, PC3), (1,PC4), (1,PC5), (2, PC6), (2,PC7), (2,PC8)}

Cette représentation permet l'implantation des modules de recherche par le contenu indépendants de toute implantation et stockage de la structure logique, et profite des stratégies de stockage qui sont déjà en place. En fait, les mêmes structures de stockage que nous avons présentées auparavant sont suffisantes pour évaluer les opérations de recherche textuelle. La construction de la liste d'entrée, par contre, a forcement besoin de structures de données additionnelles pour le stockage de la description de la structure logique.

5.3. Mise en Œuvre utilisant SCHUSS

L'environnement de SCHUSS est composé de deux modules principaux, développés en C (cf. Annexe B) : un gestionnaire de fichiers et un système de filtrage qui fournit les opérations d'accès à une base de données relationnelle . Cet environnement profite des possibilités de micro-programmation des opérations à travers un environnement APL.

Les opérations de recherche par le contenu que nous avons définies peuvent profiter du filtrage de données réalisé "au vol" par SCHUSS afin d'accélérer les temps d'évaluation des requêtes. Ceci est possible grâce à la facilité d'adaptation du filtre par la micro-programmation des opérations qui nous sont nécessaires.

Nous avons, d'une part, les opérations de filtrage de texte par l'automate, et d'autre part les opérations de filtrage de signatures. Leur implantation implique des modifications à l'environnement de filtrage existant et la définition des

nouvelles opérations, notamment en ce qui concerne la gestion de blocs de bits. Nous avons deux possibilités pour réaliser cette implantation : modifier le module de filtrage de données existant en fonction de nos besoins, ou construire un nouvel environnement en profitant de la gestion de l'espace disque offerte par le module de gestion de fichiers. Il faut rappeler que ces deux modules ne sont pas compatibles entre eux. Le module de filtrage fait sa propre gestion du disque de SCHUSS.

Nous avons considéré dans un premier temps l'alternative d'adaptation de l'environnement de filtrage existant. Elle comprend l'adaptation du compilateur de requêtes, l'adaptation du format de stockage de données sur le disque de SCHUSS (les données ne sont plus des champs formatés de taille inférieure à 256 octets) et les micro-instructions de filtrage. Cette alternative, qui nous semblait au début la plus simple, s'est montrée en fait très difficile du fait de la complexité interne du module de filtrage, même au niveau des parties logiciel.

Nous avons donc décidé d'utiliser le module de gestion de fichiers et de construire un nouveau module de compilation de requêtes, tout en gardant l'ancien module de filtrage en manière d'exemple. Nous avons considéré dans un premier temps les opérations de recherche textuelle, qui incluent aussi bien le filtrage de texte que la gestion des signatures. L'implantation des opérations de recherche sémantique obéissent aux mêmes principes, la différence étant au niveau des formats de données et des critères de sélection des résultats.

5.3.1. Architecture du Système

L'exécution d'une opération de recherche textuelle utilisant SCHUSS est fondée sur une coopération entre le système hôte, qui prépare la requête de filtrage, et SCHUSS qui effectue l'accès et

la sélection de données "au vol" (c'est-à-dire, à la vitesse des opérations d'entrée/sortie).

Les données doivent être chargées initialement sur un des disques du kit, les seuls auxquels le filtre peut accéder. Elles sont stockées dans un format reconnaissable par les micro-instructions de filtrage.

Le fonctionnement général du système est schématisé dans la figure 5.7. Le système hôte détermine les fichiers à filtrer, crée un micro-programme à partir de l'expression de filtrage, charge ces données sur la mémoire de SCHUSS et lance le micro-programme (en utilisant l'interface parallèle SCSI qui les relient). Ensuite, il attend les résultats, le processeur étant libre entre-temps pour réaliser d'autres tâches. Lorsque les résultats arrivent, l'exécution dans le système hôte reprend.

La répartition des opérations entre le système hôte et SCHUSS est la suivante :

PHASE D'INTERNALISATION DE DOCUMENTS :

Hôte :

Identification des portions de contenu.

Calcul des signatures des portions de contenu.

Envoi des données vers le kit : signatures et texte des portions de contenu.

SCHUSS :

Formattage et stockage des données.

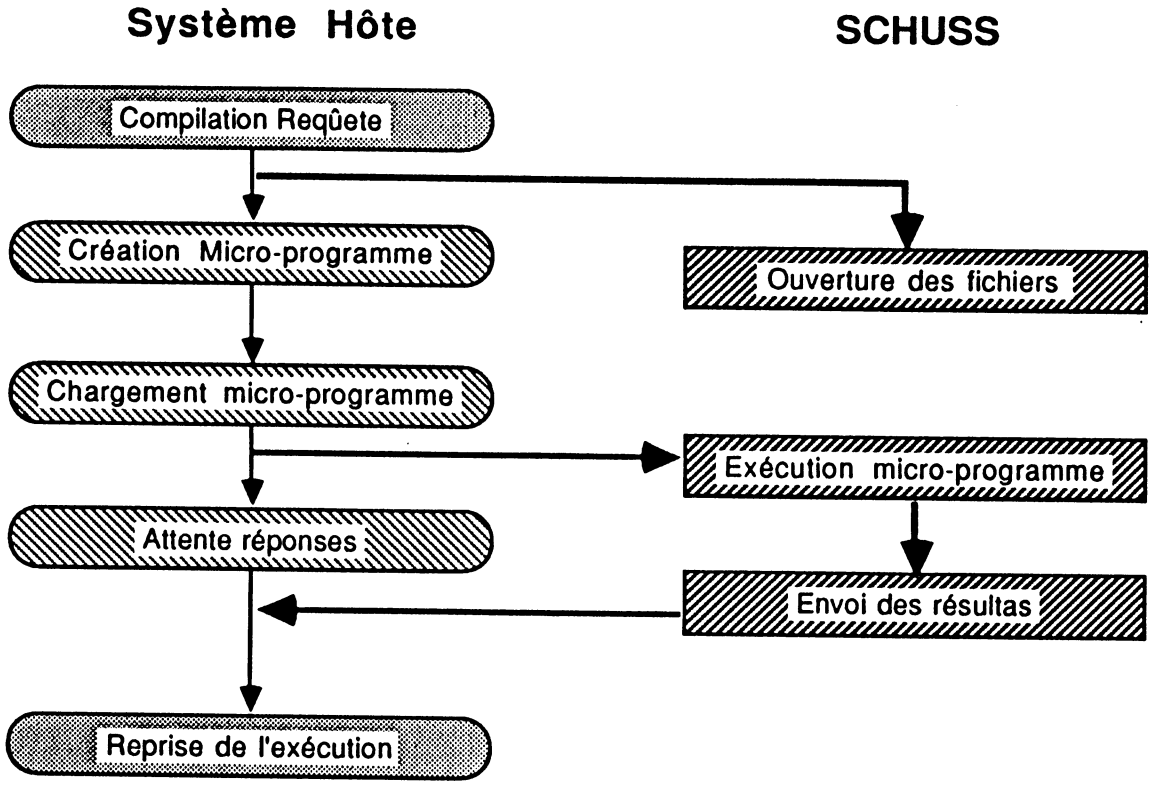


figure 5.7

PHASE DE SÉLECTION

Hôte :

Analyse syntaxique de la requête.

Création de la signature de la requête.

Création de l'automate de filtrage sous forme matricielle.

Traduction de l'automate vers un micro-programme.

Chargement sur le filtre de la signature de la requête, du micro-programme de filtrage, de la structure de mémoire de reconnaissance (cf. section 5.4.3, section 4.4.1.4).

Réception des réponses.

Schuss :

Comparaison entre la signature de la requête et les signatures des portions de contenu.

Décision de filtrer ou non le texte d'une portion de contenu en fonction du résultat de la comparaison des signatures.

Exécution du micro-programme de filtrage.

Evaluation de l'expression logique de la requête.

Mise en forme des résultats et envoi de ceux-ci vers le système hôte.

5.3.2. Stockage des Données

Les structures de données de SCHUSS sont adaptées aux opérations classiques des bases de données relationnelles. Les fichiers sont constitués de tuples, contenant un nombre fixe d'attributs (cf. Annexe B). Nous préservons cette structure dans la mesure ou nous voulons profiter au maximum de l'environnement de micro-instructions existant, ce qui est impossible si nous changeons complètement la stratégie de stockage.

En raison du traitement des données au vol par SCHUSS, la stratégie de stockage des portions de contenu et des signatures, doit exploiter la localité des données. Ainsi, le contenu du document est vu comme une séquence de signatures et de portions de contenu, qui forment les tuples des bases de données de SCHUSS.

Au contraire du critère de stockage pour un accès purement logiciel, l'ensemble de signatures n'est pas dans un fichier séparé, mais chaque portion de contenu est précédée de sa signature. Lors du filtrage, si la signature de la portion de

contenu qualifie un des blocs de signature de la requête, le texte correspondant est immédiatement filtré sans accès disque supplémentaire. L'évaluation de l'expression logique se limite donc à l'étape de filtrage de texte. Dans ce cas, les portions de contenu pré-sélectionnées ne forment pas une *liste*, comme dans le cas de l'évaluation par logiciel, mais elles déterminent le traitement des données qui arrivent ensuite du disque : soit elles sont passées sans tenir compte de leur contenu, soit elles sont filtrées.

Les tuples sont formés par quatre attributs : deux entiers pour identifier le document et la portion de contenu qui suit, la signature de la portion de contenu et le texte lui-même.

(IdDoc, IdPC, signature, texte)

Comme c'est le cas de tous les tuples en SCHUSS, chaque tuple, et chaque attribut du tuple, est précédée de sa longueur (cf. figure 5.8)

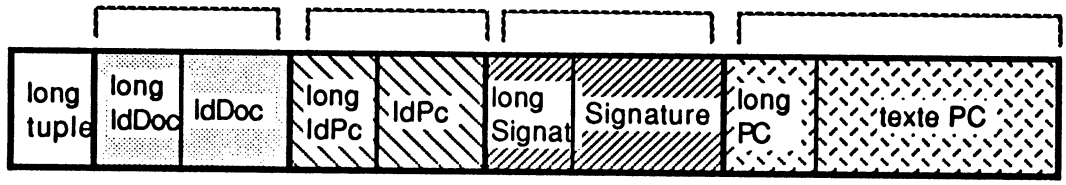


figure 5.8

Toutes les longueurs (tuple, attributs, partitions de la base) sont stockées sur deux octets. Ceci constitue un changement par rapport au modèle de stockage des bases de données original, pour lequel la longueur des attributs tient sur un octet. Nous avons changé cette longueur afin de pouvoir stocker des portions de contenu dont la longueur dépasse 256 octets. Ce changement a impliqué des modifications au niveau des micro-instructions d'accès aux tuples. Il reste néanmoins une restriction, la longueur totale du tuple ne peut pas dépasser les deux octets. Admettre des longueurs plus grandes implique des changements plus

importants et relativement complexes au niveau des micro-instructions.

Le format des réponses est une projection des tuples sur les deux premiers attributs. Elles sont donc des tuples de la forme :

(IdDoc, IdPC)

avec les longueurs devant. Ces tuples peuvent être stockés à nouveau dans le disque du filtre afin de pouvoir réaliser d'autres opérations de filtrage. Ceci n'était pas possible dans le système de filtrage de bases de données de SCHUSS, dans lequel les tuples résultats n'étaient pas "re-filtrables".

5.3.3. Micro-programmes de Filtrage

Les micro-instructions de filtrage existantes dans l'environnement de SCHUSS (cf. chapitre 3) ne sont pas tout-à-fait adaptées à nos besoins : le branchement en cas d'échec vers un état donné n'était pas prévu dans les micro-instructions de SCHUSS, et pratiquement tous les états de notre automate ont au moins trois successeurs (cf. chapitre 4).

Pour un état donné, la matrice de filtrage contient autant d'états successeurs que de caractères dans le vocabulaire de la requête. Il y a au moins trois cas qui déterminent un branchement dans l'automate (cf. chapitre 4) : le caractère suivant est un séparateur, le caractère suivant est dans la matrice, ou il conduit à l'état d'échec (état 0).

Nous avons donc décidé de construire une nouvelle micro-instruction de filtrage, *filtrer* :

<FILTRER> , N, $Adr_{échec}$, $Adr_{sépar}$, Car_1, Adr_1, Car_2, Adr_2,.... , Car_N, Adr_N

Cette instruction permet de comparer le prochain caractère en entrée avec les N caractères Car_1, Car_2, etc. Si le caractère en entrée est un séparateur, l'adresse de l'instruction suivante est $Adr_{sépar.}$ Sinon, il est comparé successivement à Car_1, Car_2, etc. S'il est égal à Car_i, l'adresse de la prochaine instruction est Adr_i. S'il n'est égal à aucun des N caractères, l'adresse de la prochaine instruction est $Adr_{échec}$. La reconnaissance des séparateurs, ainsi que le traitement des majuscules/minuscules sont faits par un tableau de transcodage qui a été chargé une fois pour toutes dans la mémoire PROM du filtre.

Si l'état reconnait une sous-chaîne de l'expression de filtrage, avec un caractère d'entrée, l'adresse suivante correspond aux instructions de vérification de l'ordre de reconnaissance dans la table de mémoire, et une mise à jour de la table d'évaluation des expressions logiques. Si l'expression logique n'est pas encore vérifiée, mais la portion de contenu permet de reconnaître un de ses arguments, le couple (IdDoc, IdPC) est projeté dans l'ensemble de résultats. Si l'expression logique est vérifiée, le filtrage s'arrête, en faisant un *avaler* du reste de la portion de contenu et la transmission de résultats vers le système hôte. Le contenu même de la portion de contenu n'est pas mis dans l'ensemble de réponses. Ceci constitue une modification par rapport au système de filtrage original de SCHUSS.

Pour la vérification de signatures, la micro-instruction *EvalSignature* permet de vérifier si une signature de portion de contenu correspond à la signature de la requête :

<EVALSIGNATURE> , M, TSign, Adr_{avaler}, $Adr_{filtrer}$, Adr_{sign}

La signature de la requête est constituée de M blocs de signature, de taille TSign octets. Les M blocs se trouvent à partir de l'adresse Adr_{sign}. Si un des blocs signature est vérifié, l'adresse suivante est celle du début de l'automate de filtrage, $Adr_{filtrer}$. Si aucun des blocs n'est vérifié, à partir de Adr_{avaler} se trouvent les instructions qui permettent *d'avaler* la portion de contenu sans la filtrer.

Il est à remarquer que si M est très grand, la performance du filtrage de données au vol décroît, puisque l'évaluation de plusieurs blocs de signatures peut prendre plus de temps que l'entrée/sortie des nouvelles données venant du disque.

Un automate de filtrage est ainsi traduit vers un micro-programme constitué d'un ensemble d'instructions *EvalSignature* et *filtrer*. La traduction nécessite deux passages sur la matrice, un pour calculer le nombre des successeurs et leur réserver la place, un autre pour compléter les adresses. La structure de mémoire de reconnaissance est stockée comme un tableau auquel l'accès est fait directement par des adresses calculées au moment de la traduction.

5.4. Expérimentation

Nous présentons par la suite les résultats des expérimentations des méthodes d'accès et de stockage que nous avons faites et décrites auparavant.

Nous avons réalisé deux des alternatives présentées au début de ce chapitre, la solution purement logicielle (cf. sections 5.2.1 et 5.3) et la solution combinée entre le logiciel et l'utilisation de SCHUSS (cf. section 5.4). La réalisation des méthodes d'accès et de stockage pour effectuer la recherche sémantique sont en cours de développement.

5.4.1. Expérimentation par logiciel

Comme nous l'avons précisé auparavant, les expérimentations de cette alternative ont été réalisées sur une machine BULL-SPS7, avec le système d'exploitation SPIX 31.3. Le serveur OIS a été

développé aussi sur cette machine, utilisant le SBGD relationnel ORACLE.

Nous avons considéré deux types de données : les documents ODA, qui sont stockés sur ORACLE, et des documents stockés sous forme de fichiers de texte UNIX. L'ensemble des fonctions pour la recherche par le contenu textuel de documents sur ces données est écrit en C. Elles représentent un total de 4154 lignes distribuées ainsi :

Analyse Syntaxique de requêtes :	286 lignes
Création des Automates de Filtrage :	538 lignes
Exécution des Automates de Filtrage :	179 lignes
Création des Blocs de Signature :	166 lignes
Fonctions de base (gestion bits, E/S, etc) :	441 lignes
Programmes principaux :	283 lignes
Déclarations :	212 lignes
Sur les documents UNIX :	
Filtrage de Signatures :	669 lignes
Filtrage de Textes :	255 lignes
Sur les documents ORACLE :	1125 lignes

L'évaluation de l'implantation a été faite à partir de l'ensemble de documents stockés sur UNIX, constituant un échantillon de texte de volumen représentatif. Les modules de recherche par le contenu sur les documents stockés sur ORACLE sont similaires, les temps d'évaluation étant dépendants des performances du SGBD.

5.4.1.1. Les Données

Les données que nous avons utilisées sont de deux types :

- Des documents ODA, stockés sur ORACLE, qui correspondent à une application de courrier électronique.

- Un ensemble de documents UNIX.

Les documents UNIX utilisés sont deux thèses, un article, et un ensemble de Comptes Rendus Radiologiques. L'article est un document écrit en anglais, les autres sont écrits en français. Leur volume de données textuelles est :

Fichier THESE1 : 315350 octets	2150 Portions de Contenu
Fichier THESE2 : 204011 octets	748 Portions de Contenu
Fichier ARTICLE : 492312 octets	274 Portions de Contenu
Fichier MEDICAL : 23637 octets	172 Portions de Contenu

Les paramètres qui définissent la performance des opérations de recherche sont, d'une part la taille des blocs de signatures, et d'autre part le nombre de triplets par PC. Un bloc de signatures trop grand implique une perte de temps pendant la phase d'évaluation de signatures (le fichier de signatures est trop grand), mais il y aura très peu de collisions. De même, si le nombre de triplets par PC est trop grand par rapport à la taille du bloc de signatures, il y aura beaucoup de collisions et par conséquent plus de PC à filtrer dans l'étape secondaire.

Nous avons étudié le comportement de la taille des PC dans les documents. Les résultats sont résumés dans les figures 5.9 et 5.10. Nous remarquons que, même sur des types de documents différents, la longueur des PC suit en gros la même distribution.

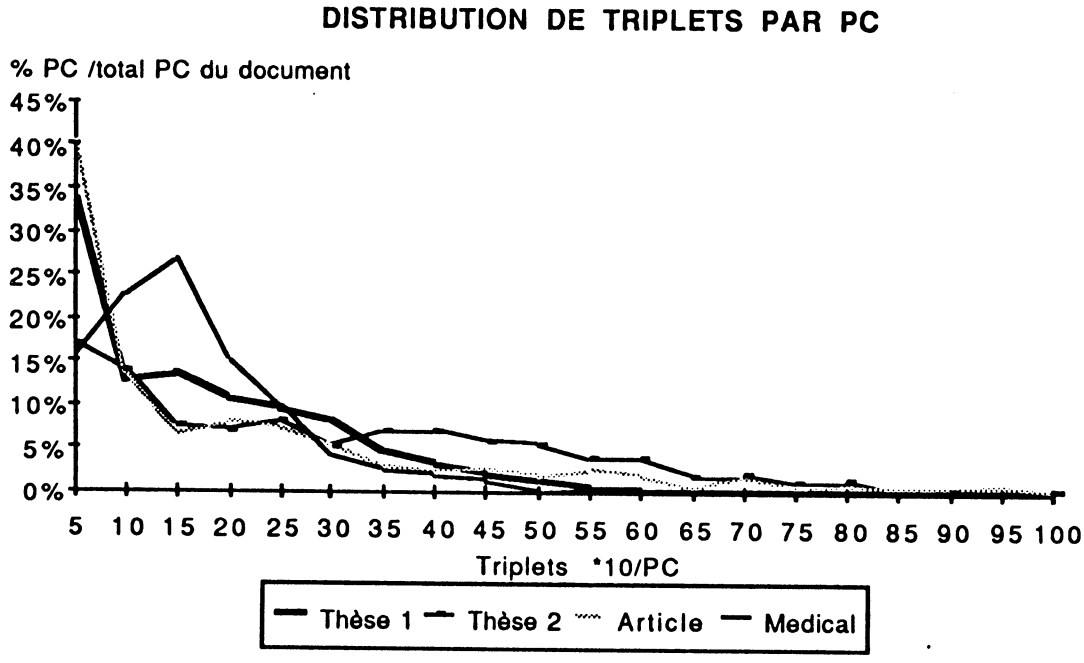


figure 5.9

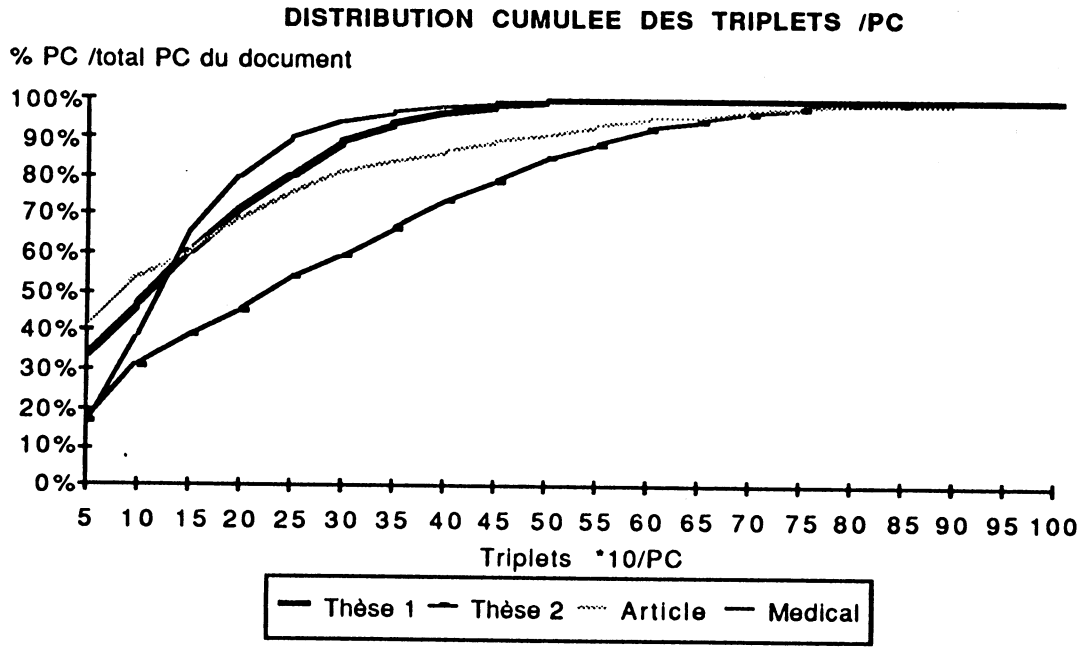


figure 5.10

Nous pouvons voir qu'au moins 15% des PC ont au maximum 50 triplets, et que dans tous les documents, 50% des PC ont moins de 230 triplets. Dans le tableau 5.1, et la figure 5.11 nous détaillons les données du document THESE1. Les mêmes données, pour les autres documents se trouvent dans l'Annexe C.

TABLEAU DE DISTRIBUTION DE TRIPLETS/PC Doc: THESE1

Taille (Tripl./PC)	Fréquence (%PC)	Cumulé
1-50	33,767%	33,767%
51-100	12,884%	46,651%
101-150	13,581%	60,233%
151-200	10,698%	70,930%
201-250	9,581%	80,512%
251-300	8,372%	88,884%
301-350	4,465%	93,349%
351-400	2,977%	96,326%
401-450	1,907%	98,233%
451-500	1,163%	99,395%
501-550	0,372%	99,767%
551-600	0,140%	99,907%
601-650	0,047%	99,953%
651-700	0,047%	100,000%

Tableau 5.1

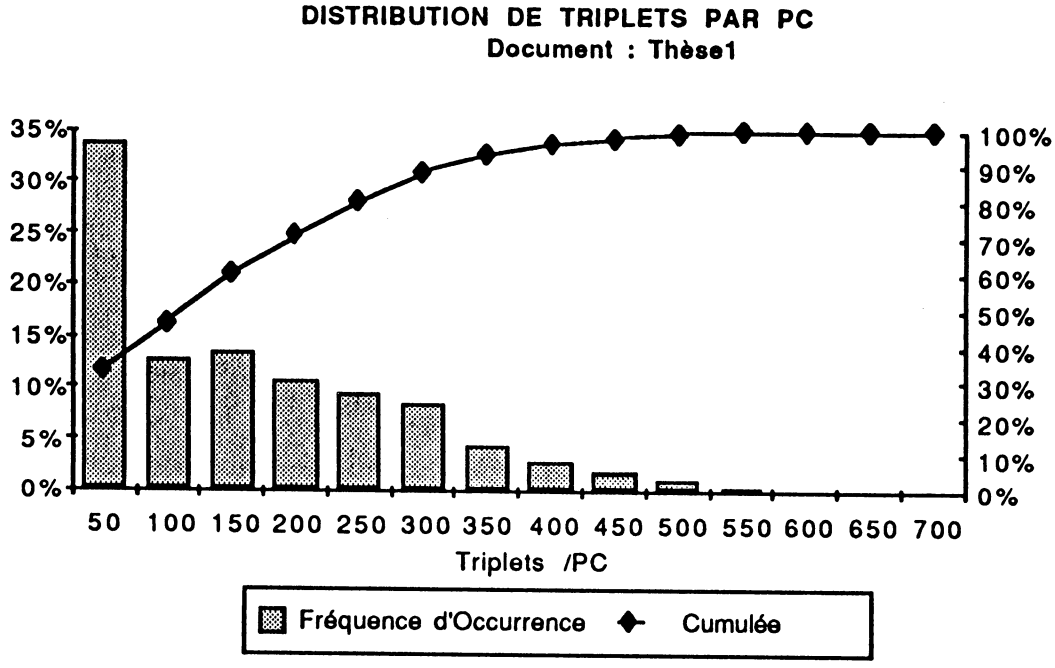


figure 5.11

Nous remarquons que, dans le document correspondant à l'article, ainsi que pour une des thèses, il y a un grand nombre de PC contenant très peu de triplets (moins de 50), comparé aux Comptes Rendus Médicaux. Ces PC de petite taille proviennent en général des titres, des exemples et des énumérations. Nous avons analysé l'influence des titres dans la distribution de triplets/PC pour les trois documents en question. La proportion de portions de contenu correspondant aux titres dans les documents est la suivante :

Influence des Titres dans le total de PC/Document

Document	PC titres	total PC
ARTICLE	27	274
THESE 1	217	2150
THESE 2	69	748

tableau 5.2

Ils correspondent environ à 10% du total des PC des documents. Dans la figure 5.12 nous montrons la distribution des tailles des PC pour le document THESE 1 complet et pour le même document en lui enlevant les titres.

La fréquence moyenne dans le document complet est dans l'intervalle 130-140 triplets/PC, et dans le document sans titres elle est dans l'intervalle 150-160 triplets/PC. Les analyses sur les autres documents se trouvent dans l'Annexe C. Nous pouvons remarquer que le fait d'enlever les titres pour analyser la distribution fait décaler un peu les courbes, mais que ce changement n'est pas très important. En fait, la tendance de la distribution se conserve et le déplacement de la moyenne est minimal. Nous pouvons donc conclure que pour l'analyse du comportement de la taille des PC dans une grande base de documents, il n'est pas indispensable de distinguer les titres des autres PC des documents.

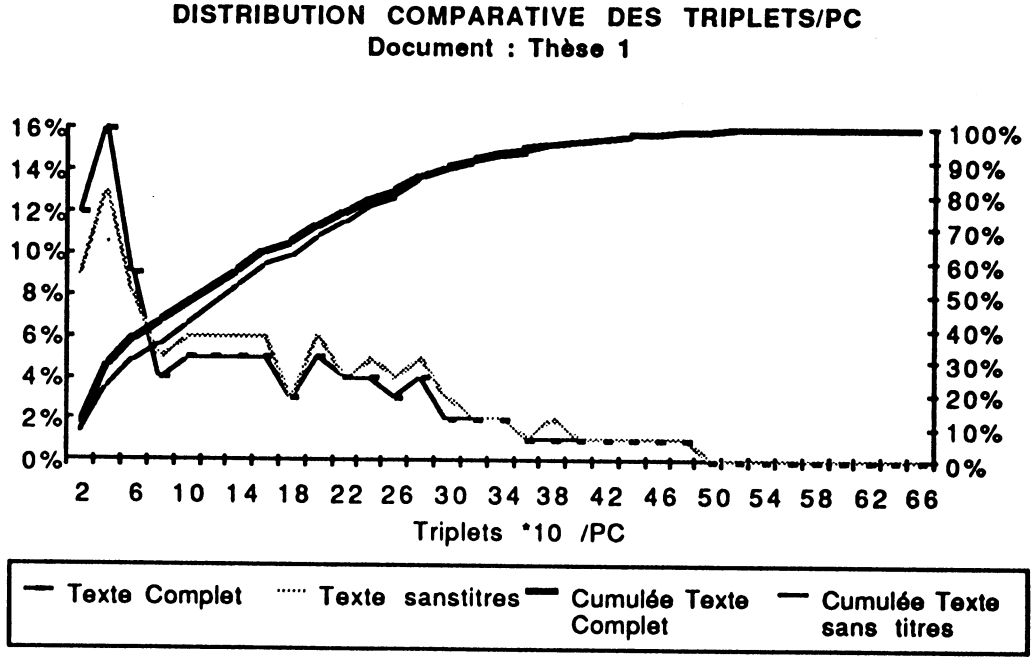


figure 5.12

La distribution statistique des tailles des PC (mesurées en termes de nombre de triplets), de même que celle de la fréquence de mots dans la langue, ne peut être établie que de manière empirique [Mul68]. Nous constatons qu'elles suivent une distribution similaire, comme nous l'avons supposé dans nos hypothèses du chapitre 4. Néanmoins, nous sommes conscients, néanmoins, que les paramètres de cette distribution dépendent du type de document et de la structure de la langue.

5.4.1.2. Evaluation de Requêtes

Nous avons analysé les performances de la méthode de stockage et d'accès que nous avons définie, en termes de trois paramètres :

- l'*overhead* causé par l'index des signatures,
- le taux de collisions introduit pour une longueur de bloc de signatures donnée, et
- le temps d'évaluation des requêtes.

Taux de Compactage

L'*overhead* causé par le stockage de l'index de signatures est directement proportionnel au nombre de PC du document correspondant. Il y a autant de blocs de signature que de PC, et la manière de contrôler cet "overhead" est donné par la taille des blocs de signatures. Le rapport *taille du document / taille de l'index de signatures*, est appelé **taux de compactage**, et c'est cette mesure qui définit l'espace de stockage nécessaire pour les documents du SIB.

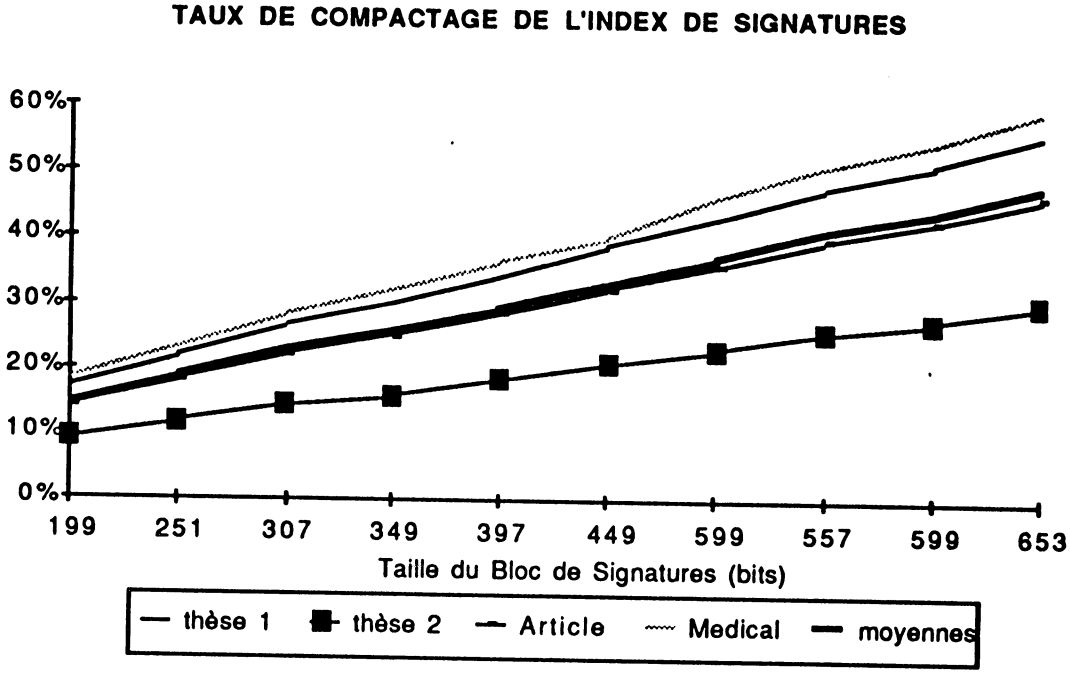


figure 5.13

Nous montrons dans la figure 5.13 les résultats obtenus pour 10 tailles de blocs de signatures sur les 4 documents de test, et leur moyenne. Nous pouvons remarquer que l'index prend plus de place que la moyenne pour le document MEDICAL, tandis que pour le document THESE 2 nous obtenons les meilleurs résultats. Par exemple, pour des blocs de signatures de 499 bits, l'index de THESE 2 occupe 21% de la taille du document original, tandis que l'index de MEDICAL occupe 40%.

Ceci est expliqué par la distribution des tailles des PC de ces documents (voir figure 5.9). En fait, un document qui a beaucoup de PC de petite taille aura un taux de compression moins intéressant qu'un document avec le même nombre de PC, mais plus grandes, les deux ayant un même nombre de blocs de signatures de même taille. Pour les petits blocs de signatures, les différences entre types de documents ne sont pas importantes. En fait, les PC très grandes sont compressées vers des blocs de petite taille. Pour des blocs de signatures très

grands, les PC les plus petites ne sont pratiquement pas compressées.

Taux de Collisions

Nous avons utilisé pour cette expérimentation un jeu de 23 requêtes et 10 tailles différentes pour les blocs de signatures. Nous montrons ci-dessous les résultats obtenus pour le document THESE 1 (315350 octets et 2150 PC).

Le taux de collisions est mesuré comme la proportion de blocs sélectionnés dans l'ensemble total de blocs qui ne répondent pas à la requête [CF 84] (cf. chapitre 4). Soient N le nombre total de blocs considérés, S le nombre de blocs qui répondent effectivement à la requête et C le nombre de blocs sélectionnés qui ne répondent pas à la requête, alors

$$\text{Taux de Collisions} = \frac{C}{N - S}$$

Dans le tableau 5.3 et la figure 5.14 nous présentons le comportement du taux de collisions et du taux de compression pour chaque taille de bloc de signatures.

Taux de Collisions et de Compactage par taille de Bloc Signatures

Taille Bloc	Moyen. Collis.	Max Collis.	Min. Collis	Taux Compact.
199	8,13%	31,05%	0,39%	17,04%
307	3,35%	10,41%	0,42%	26,59%
397	1,23%	2,59%	0,00%	34,09%
499	1,44%	5,79%	0,05%	42,95%
599	0,74%	2,84%	0,00%	51,13%

tableau 5.3

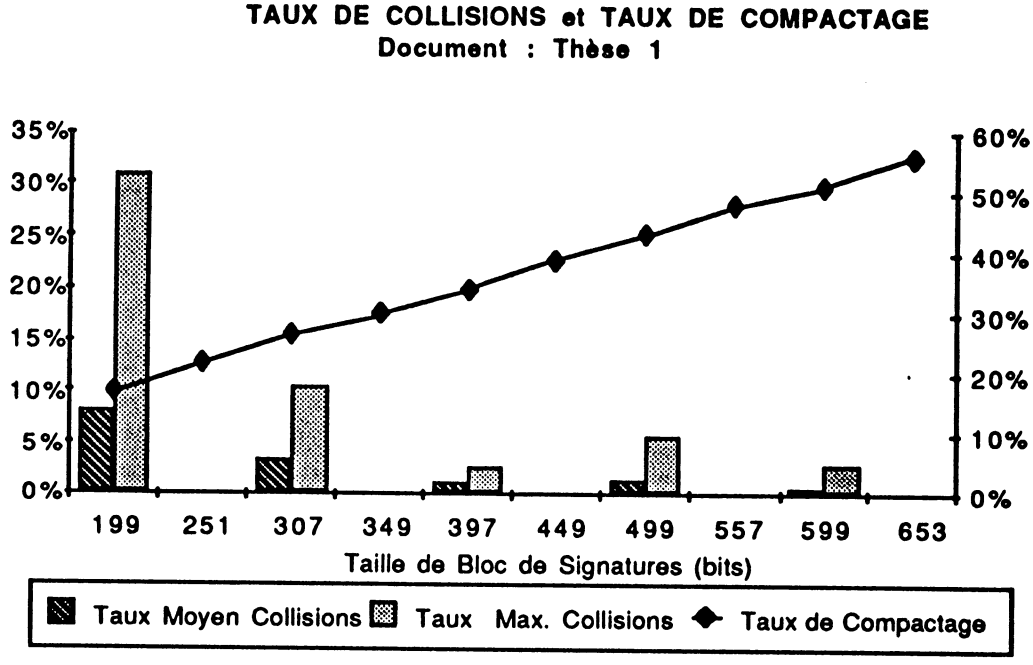


figure 5.14

Pour ce cas, il faut remarquer le comportement du taux de collisions dans le cas des blocs de signatures de 199 bits, pour lesquels la variance est très grande. En fait, la moyenne étant à 8%, il y a des cas de requêtes pour lesquelles nous trouvons jusqu'à 31% de collisions. Ceci s'explique par le fort taux de compression des PC de taille importante (voir figure 5.9), leurs signatures contient un grand nombre de bits à "1" et elles ne sont pas discriminantes.

Au fur et à mesure que la taille des blocs de signature augmente l'écart par rapport à la moyenne des collisions est plus petit, et nous pouvons dire que la performance du système est plus uniforme. A partir de 397 bits/bloc de signature nous trouvons un taux maximal de collisions du 5%, la moyenne étant à inférieur à 1.5%.

Temps d'Evaluation

Les temps d'évaluation ont été calculés sur 20 exécutions par requête et par taille de bloc de signatures. Ceci nous donne un temps moyen d'évaluation établi sur la moyenne de 460 tests pour chaque cas.

Le temps d'évaluation est proportionnel à la taille de l'index de signatures et à la taille des PC présélectionnés, sur lesquels le filtrage de texte par l'automate est effectué. Deux processus sont néanmoins indépendants de ces tailles : l'analyse de la requête (analyse syntaxique et création de la signature de la requête) et la création de l'automate de filtrage.

Nous montrons dans les tableaux 5.4 et 5.5 les résultats que nous avons obtenus pour ces deux processus et pour l'ensemble des requêtes de test.

Le temps de création de la signature de la requête est pratiquement négligeable, la précision donnée par le système étant 1/50 sec = 0,02 sec. Par contre, le temps de création de l'automate est plus important, allant jusqu'à 1/2 sec. Le temps minimal est celui des expressions contenant une seule chaîne entourée de jokers ; ensuite se trouve celui des mots sans jokers ; puis celui des chaînes contenant des jokers à l'intérieur des chaînes ; finalement les temps maximaux ont été observés pour les requêtes contenant des opérateurs logiques.

	Création Sign. Requête	**Création Automate**
Temps Moyen	0,007 sec	0,366 sec
Temps Max	0,016 sec	0,909 sec
Variance	0,001%	5,7968%

tableau 5.4

Requête	Créat. Sign.	Créat. Automate
document	0,007 sec	0,216 sec
document	0,007	0,43
relevance	0,005	0,214
relevance	0,006	0,424
boolean	0,003	0,163
boolean	0,005	0,315
maison	0,004	0,141
maison	0,005	0,282
Office	0,006	0,138
*Office*Inf*	0,007	0,233
*Office*Inf*Serv*	0,004	0,397
index	0,006	0,116
*index*term*	0,007	0,237
attributs	0,008	0,426
structure	0,008	0,385
attributs OR structure	0,013	0,54
inform*	0,003	0,158
retrie	0,003	0,137
inform*retrie*	0,006	0,373
analyse AND *semant*	0,009	0,397
*analys*semant* AND *linguis*	0,006	0,878
*analys*semant* OR *base*connais*	0,01	0,909
*analys*semant* AND *base*connais*	0,016	0,898

tableau 5.5

Dans le tableau 5.6 et dans les figures 5.15 et 5.15a nous montrons les temps moyens d'évaluation pour ces requêtes sur le document THESE 1, et l'extrapolation pour une base de texte de 1 Méga-Octet. Nous montrons la variation de ces temps par rapport à la taille des blocs de signature, et finalement une comparaison avec l'option de filtrage seul de texte, sans utiliser l'index de signatures, et avec l'option de filtrage de signatures sans élimination de collisions.

Filtrage Texte

	Temps Moyen	Temps Max	Moyen 1MO	Max 1MOctet
	6,98 sec	20,90 sec	22,34sec	68,55 sec

Filtrage Signatures et Texte

199	2,28 sec	4,06 sec	6,72 sec	13,17 sec
307	2,39	3,72	7,07	11,37
397	2,29	3,36	6,75	10,82
499	2,66	3,89	7,99	11,98
599	2,76	3,88	8,32	11,77

Filtrage Signatures Seules

199	1,07 sec	2,18 sec	2,69 sec	6,31 sec
307	1,37	2,26	3,68	6,58
397	1,68	2,59	4,71	7,67
499	2,00	3,33	5,77	10,11
599	2,33	3,36	6,89	10,22

tableau 5.6

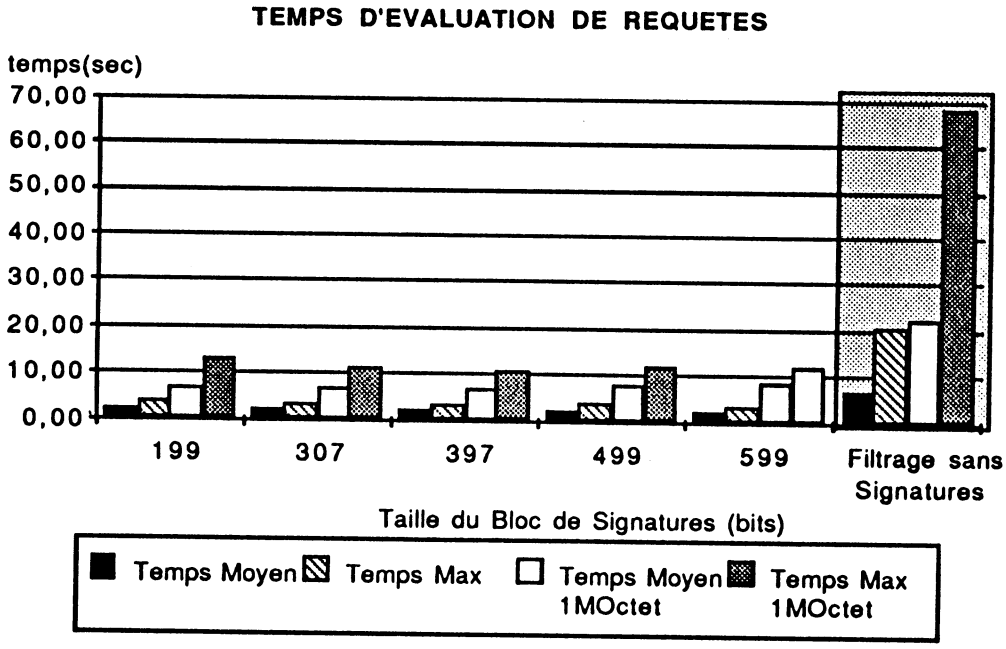


figure 5.15

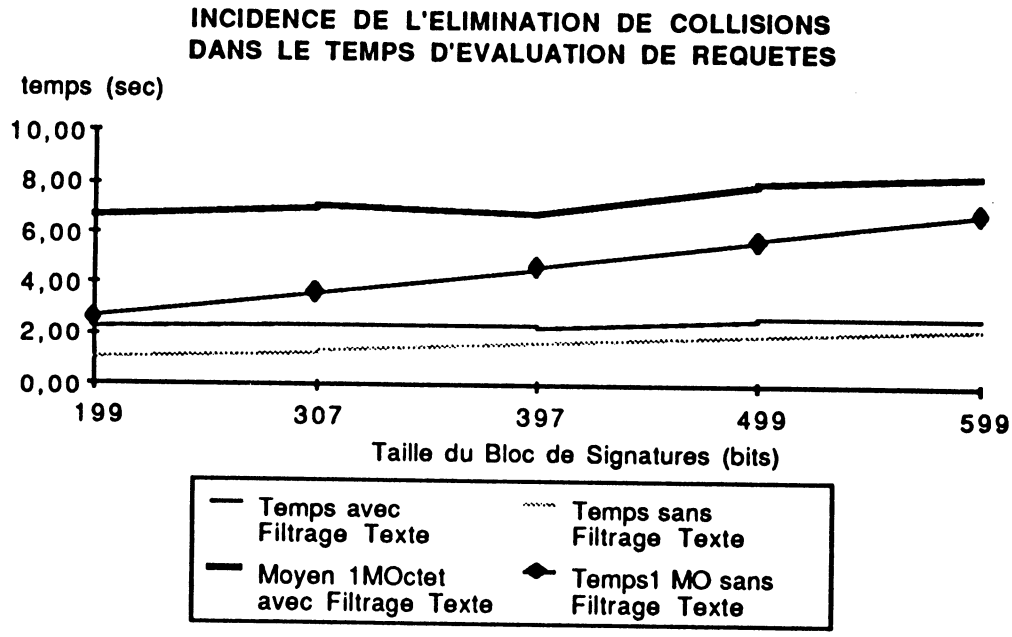


figure 5.15 a

Pour l'évaluation de requêtes éliminant les collisions, les temps relatifs aux différentes tailles de blocs de signatures ne varient pas de manière significative. Ils se situent entre 30 et 35% du temps de filtrage de texte sans utiliser l'index. En fait, pour les blocs de signatures petits, le temps de filtrage de l'index est plus court que pour les grands blocs, mais il y a plus de PC présélectionnés, donc plus de filtrage de texte à faire par l'automate.

Par contre, dans le cas du filtrage de l'index sans élimination de collisions, le temps d'évaluation est proportionnel à la taille du bloc de signature, tel qu'il était prévu. Dans ce cas les temps d'évaluation se situent entre 15 et 33% du temps de filtrage sans utiliser l'index.

La figure 5.15a montre la comparaison du temps d'évaluation des requêtes en utilisant l'index de signatures avec ou sans élimination de collisions. Dans le cas des blocs de signatures

petits, pratiquement 50% du temps d'évaluation est dû au filtrage de texte, contrairement au cas des grands blocs, où il ne représente que 13%.

Dans l'Annexe C nous montrons les résultats détaillés de ces expérimentations. Il est intéressant de noter l'influence des expressions logiques dans les requêtes. En fait, pour l'évaluation avec ou sans élimination de collisions, le temps moyen est augmenté de 70% par rapport au temps d'évaluation de requêtes simples. Par contre, le fait d'avoir des jokers n'a pas des effets significatifs sur ces temps.

Les meilleures performances de la méthode, en ce qui concerne les temps d'évaluation, se trouvent pour les cas où la réponse à la requête est négative (le cas le plus coûteux pour le filtrage de texte sans utiliser l'index). Le temps d'évaluation utilisant l'index avec élimination de collisions est d'environ 14,3% et sans élimination de collisions il est d'environ 8% (cf. Annexe C).

Ceci n'est pas toujours vrai pour les réponses affirmatives. En fait, sans utiliser l'index, le temps de filtrage est proportionnel à la position de l'expression recherchée dans le texte, ce qui produit un grand écart entre les temps d'évaluation de requêtes. Dans le cas général, une des avantages de l'utilisation de l'index de signatures est la homogénéisation des temps d'évaluation (cf. figure 5.15).

Dans la figure 5.16 nous montrons le comportement général du système, simultanément pour tous les paramètres que nous avons considérés.

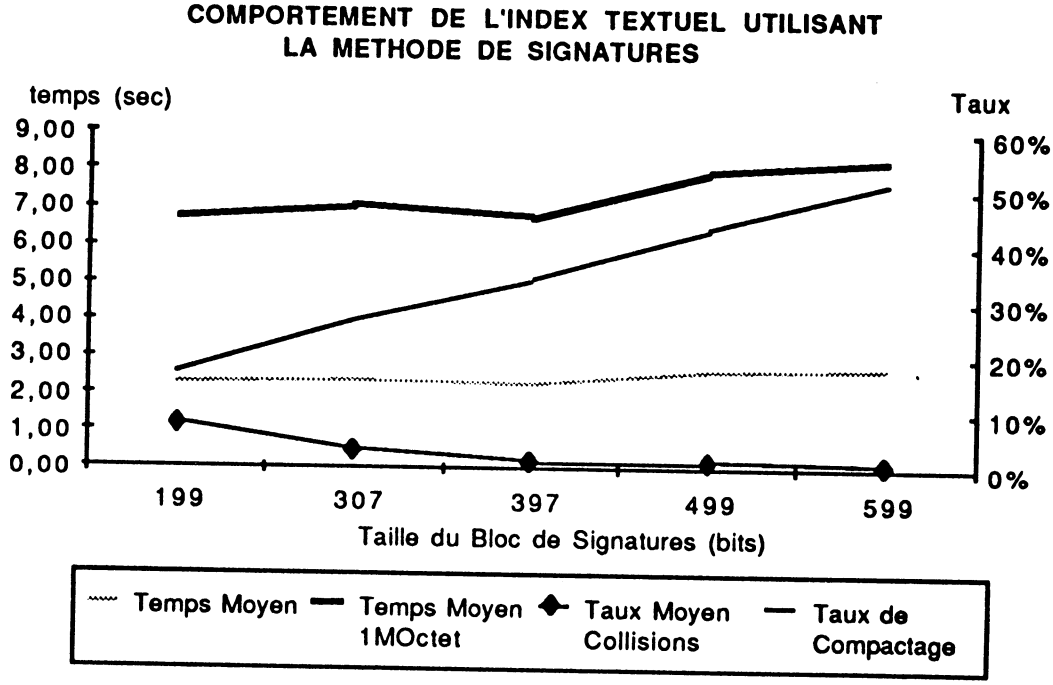


figure 5.16

Nous en déduisons que, pour des documents du même type que le document THESE 1, la taille de bloc de signatures la plus adéquate est 397 bits. C'est pour cette taille que nous trouvons les meilleurs temps d'évaluation, avec environ un 1% de collisions et un taux de compression de 34%.

5.4.1.3. Commentaires

Les résultats que nous avons montrés correspondent à l'expérimentation sur des documents stockés dans des fichiers Unix. La différence par rapport à l'implantation sur ORACLE, pour les documents ODA, se trouve dans la manière d'obtenir, dans une mémoire tampon ("buffer"), le texte correspondant à chaque portion de contenu.

Pour les documents Unix l'obtention des PC est réalisée par une opération de lecture sur un fichier. Pour ORACLE c'est une

opération de sélection en SQL, qui permet de retrouver dans la table des objets ODIF le contenu du document.

Pour le cas des fichiers Unix les portions de contenu sont identifiées par leur position dans le fichier et par un marqueur de fin. Pour le cas de ODA, les PC sont identifiées par leur position dans l'objet ODIF et par leur longueur.

L'implantation que nous avons testée est ainsi indépendante de la codification ODIF, et elle est transportable vers d'autres modèles de documents. Ceci est possible dans la mesure où les portions de contenu sont identifiables au moyen d'une des deux formes proposées, et de disposer des fonctions qui permettent le regroupement des PC par élément logique du document.

Les fonctions de recherche textuelle au niveau du Gestionnaire de Documents du SIB forment un bibliothèque sous Unix. L'intégration dans un autre système (un gestionnaire de références bibliographiques), afin de permettre la recherche textuelle sur les résumés de ces références, est en cours de réalisation.

5.4.2. Expérimentation utilisant SCHUSS

Pour la réalisation sur SCHUSS il a été nécessaire de transporter les modules de recherche textuelle vers le PC de BULL, le BM-30. C'est la seule machine hôte possible pour le kit dans l'état actuel du prototype. L'intégration avec le reste du serveur implique la connexion du kit avec le SPS7, ce qui nécessite le développement d'un nouveau contrôleur.

Le premier pas pour l'implantation utilisant SCHUSS a été donc le transport sur MS/DOS des modules logiciel, tel qu'ils ont été décrits. Pour faciliter ce transport, ainsi que pour pouvoir

l'intégrer au Gestionnaire de Fichiers de SCHUSS (écrit en Latice-C), nous avons tout installé en Turbo-C.

Nous avons modifié et augmenté l'environnement de micro-instructions afin de permettre :

- Le stockage des fichiers (les signatures et les portions de contenu).

- L'exécution d'un automate de filtrage, par l'implantation de la micro-instruction FILTRER.

- La vérification des signatures par l'implantation de la micro-instruction EVAL_SIGN.

- Le transcodage de caractères séparateurs et majuscules/minuscules.

- La vérification des expressions logiques et de l'ordre de reconnaissance des chaînes de la requête.

- Les mécanismes de décision de filtrage d'une PC en fonction du résultat de la vérification des signatures.

Nous avons effectué les tests des trois premières modifications. Les trois dernières sont déjà réalisées, mais malheureusement elles n'ont pas pu être testés en raison d'un panne du matériel. La maintenance du kit, en tant que prototype de recherche, est difficilement assurée. Dès que ces problèmes seront résolus nous pourrons continuer l'expérimentation et les test correspondants.

Les difficultés dans la tâche de développement sur SCHUSS sont augmentées par le caractère expérimental du matériel. En fait, nous avons souvent dû reprendre tout le travail en raison des changements de versions, tant du logiciel que du matériel. La documentation n'est pas toujours à jour et il y a des ambiguïtés.

Par exemple, la différence entre la valeur "0" et la valeur "null" dans les paramètres des micro-instructions n'est pas toujours claire et exposée de manière uniforme. Il faut ajouter le manque d'un débugger au niveau des micro-programmes, la version actuelle n'est pas à jour.

5.5. Conclusions

Dans ce chapitre nous avons montré l'implantation des opérations de recherche par le contenu. Nous avons présenté deux alternatives d'implantation, d'une part une solution purement logicielle, aussi bien pour la recherche textuelle que pour la recherche sémantique. D'autre part, nous avons considéré la combinaison d'opérations réalisées par une machine spécialisée en filtrage, SCHUSS, avec des opérations réalisées par logiciel dans le système hôte, le SIB. Nous avons présenté aussi l'intégration que nous avons faite dans le cas d'un serveur d'information bureautique (l'OIS, construit sur un SGBD relationnel). Finalement dans ce chapitre nous avons présenté l'état actuel de cette réalisation et les résultats que nous avons obtenus.

Les modules de recherche textuelle sont en cours d'intégration à un autre système, la gestion d'une base de références bibliographiques. L'intérêt de les utiliser dans ce système est la possibilité de consultation des résumés des références. Nous sommes aussi en train de construire un Gestionnaire de Documents indépendant d'un serveur ou d'une application particulière, comme une extension au SGBD relationnel. Les modifications à faire dans ce cas sont minimales par rapport à l'implantation que nous avons intégrée au serveur OIS. Il s'agit notamment de généraliser l'accès aux tables relationnelles.

L'expérience d'implantation utilisant SCHUSS a été intéressante et enrichissante. La machine apporte un grand potentiel de développement. Néanmoins, nous avons rencontré beaucoup de problèmes d'ordre technique. Même si en théorie il est facile de modifier l'environnement de filtrage, le caractère expérimental du prototype, pour lequel la documentation n'est pas toujours complète, nous a causé beaucoup de retards. Cette remarque est aussi valable du point de vue matériel, pour lequel pratiquement aucune assistance technique n'est assurée. Les pannes du matériel deviennent ainsi de graves problèmes dont la solution est difficile et coûteuse. L'implantation des opérations pour effectuer la recherche sémantique n'a pas pu être complétée en raison de ces problèmes.

Evidement, des améliorations sont possibles sur la réalisation des modules des prototypes logiciels. Pour l'utilisation des opérations de recherche dans le cadre d'une base importante de documents, on pourrait diminuer les temps d'évaluation des requêtes, aussi bien pour les signatures que pour le texte. Nous pouvons penser, par exemple, à l'utilisation de fonctions plus efficaces, spécifiques au système d'exploitation ou au niveau du matériel, pour la gestion des blocs de bits et le transcodage de caractères.

6 . CONCLUSIONS

Dans ce travail nous avons abordé le problème de la définition de méthodes d'accès au contenu de documents multimédia dans le cadre d'un Serveur d'Information Bureautique, le SIB. Dans le SIB nous regroupons les opérations concernant les documents dans un module que nous appelons le Gestionnaire de Documents. Il s'occupe de l'acquisition, de la sélection, du stockage, de l'accès et de l'externalisation des documents.

Une grande diversité d'applications peuvent être construites sur un SIB. Les documents gérés sont ainsi de nature très différente, et les opérations d'accès qu'il est souhaitable de réaliser sur leur contenu ne sont pas toujours les mêmes. Dans certains cas il est suffisant de faire une sélection par des critères de "pattern matching", dans d'autres cas il est préférable d'utiliser les facilités offertes dans le cadre d'un système de recherche documentaire, à savoir, pouvoir accéder à un modèle sémantique de ce contenu.

Nous adoptons comme modèle de structure logique des documents dans le SIB celui du standard ODA. Le contenu textuel des documents se trouve toujours au niveau des feuilles de l'arbre, dans les Portions de Contenu. Elles peuvent contenir des données multimédia : texte, graphiques, images, etc. Nous nous occupons de l'accès aux portions de contenu textuelles dans le document.

Nous considérons dans le SIB la co-existence de deux modèles du contenu de ces documents : d'une part, le *contenu textuel*, qui est formé par l'ensemble de chaînes de caractères qui se trouvent au niveau des portions de contenu des documents ; d'autre part, nous considérons le *contenu sémantique*, qui est le résultat d'une analyse du contenu du document, une normalisation des termes et

finalement une pondération de la pertinence des concepts retrouvés. Ce modèle du contenu est connu comme la relation d'indexation du document.

Nous avons défini deux opérations de recherche, la **RECHERCHE TEXTUELLE** et la **RECHERCHE SÉMANTIQUE**. La recherche textuelle utilise comme modèle du contenu des documents leur structure logique et leur contenu au niveau des portions de contenu ; elle peut être réalisée sur tous les documents du SIB. La recherche sémantique utilise comme modèle du contenu du document sa relation d'indexation ; elle est possible seulement pour l'ensemble de documents pour lesquels la relation d'indexation a été créée.

L'intégration de ces opérations est faite à deux niveaux : d'une part, par l'expression dans le langage de manipulation de données du serveur, des opérations et des critères de sélection ; d'autre part, par une interface fonctionnelle entre le Gestionnaire de documents, qui réalise effectivement les opérations d'accès, et les autres modules du serveur.

Pour effectuer les opérations d'accès à ces modèles du contenu nous avons défini deux index, l'un sur les portions de contenu (l'index textuel) et l'autre sur les éléments de la relation d'indexation (l'index sémantique). L'environnement de bureau, par son caractère dynamique et ouvert, ainsi que pour le grand nombre d'applications auxquelles il peut donner support, nous impose certaines conditions sur la structure et sur la maintenance de ces index. Nous avons adopté les méthodes de signatures pour le calcul et le stockage de ces index. Elles nous offrent l'avantage d'avoir des surcoûts de place de stockage raisonnables et elles facilitent les mises à jour des index, et nous permettent la sélection rapide des documents demandés. La taille des index, ainsi que le temps d'évaluation restent, néanmoins, proportionnels à la taille des documents. Nous avons expérimenté l'utilisation d'une machine orientée au filtrage de données pour accélérer ces opérations, le prototype SCHUSS.

L'apport de ce travail se situe par rapport aux domaines qui nous ont servi de cadre :

Du point de vue des Systèmes de Recherche d'Information il constitue un pas pour l'ouverture de ces systèmes vers des environnements dynamiques. Nous avons défini les éléments d'un SRI pouvant être traités au niveau d'un SIB, en ce qui concerne le stockage et l'accès aux documents et à la relation d'indexation, en les distinguant des aspects plus spécifiques qui sont dépendants des applications. Notre expérimentation a montré que l'on pouvait concilier ces objectifs avec des meilleures performances en place et en temps d'accès, notamment pour le problème critique du stockage et de l'utilisation des index.

Du point de vue des systèmes de gestion de bases de données, et d'information multimédia, nous avons défini deux opérations et leur méthodes d'accès sur les parties textuelles des documents. Nous avons montré l'intégration de ces opération dans un modèle de données, le Fact Model, et l'intégration au niveau des structures de stockage et des fonctions d'accès sur un SGBD relationnel. Nous avons montré l'intérêt d'un filtre matériel tel que SCHUSS pour réaliser le filtrage de données, aussi bien du texte que des signatures. Nous avons intégré ces opérations dans un serveur particulier, l'OIS, développé dans le cadre du projet ESPRIT DOEOIS, et dans la construction d'un Gestionnaire de Documents construit sur un SGBD relationnel.

La suite de ce travail se situe notamment sur deux aspects :

- D'une part, la considération de signatures de taille variable, en fonction soit du type de document, soit de la taille des portions de contenu et des termes d'indexation.

En fait, la taille typique des portions de contenu est assez difficile à estimer, seule l'expérimentation permet de le faire. Les études statistiques classiques que nous avons trouvées font toujours référence au comportement des mots dans la langue. Il n'en existe aucune sur le comportement des triplets de caractères, ou de celui des tailles d'unités textuelles (qui correspondent, de manière intuitive, aux portions de contenu).

La taille des Portions de Contenu varie aussi en fonction du type de document, et plus généralement, des critères de définition des documents dans les applications. Une méthode de signatures qui s'adapte à ces changements permettrait de maintenir les paramètres de performance (taux de compression, taux de collisions) uniformes à toutes les applications du Serveur.

- D'autre part, sur l'étude de méthodes de transformation sans perte d'information ; ceci est particulièrement important dans le cas de la recherche sémantique.

Les méthodes de transformation que nous avons utilisées pour la constitution des blocs de signature nous obligent à réaliser l'étape de filtrage secondaire afin d'obtenir des réponses exactes. Cette étape serait évitée, et seul l'accès aux index serait nécessaire lors de l'interrogation.

Sur le plan de l'expérimentation, il reste à valider l'approche de la recherche sémantique, dont le développement n'a pas pu être terminé du fait de difficultés techniques insurmontables (pannes difficilement réparables). Il serait notamment intéressant de reprendre notre approche dans le contexte de nouveaux outils tel que le coprocesseur µSyc, développé par BULL.

Finalement, il est intéressant d'étudier le couplage des opérations que nous avons définies sur le contenu textuel et sémantique des documents, avec des méthodes d'accès aux autres composants multimédia dans les documents : les graphiques, les images, la voix, ainsi que sur d'autres approches de gestion de documents.

BIBLIOGRAPHIE

Bibliographie par Thème

Systèmes de Recherche d'Information

[BCB86] Di BENIGNO M. K., CROSS G. R., DEBESSONET C. G.
"COREL - A Conceptual Retrieval System".
Proceedings of the 1986-ACM Conference on Research and Development in Information Retrieval. Pages 144 - 148. Pisa, September 1986.

[BP 86] BERRUT C., PALMER P.
"Solving grammatical Ambiguities within a surface syntactical parser for automatic indexing"
Proceedings of the 1986-ACM Conference on Research and Development in Information Retrieval. Pages 123 - 213. Pisa, September 1986.

[Ber88] BERRUT Catherine.
"Une Méthode d'Indexation Fondée sur l'Analyse Sémantique de Documents Spécialisés. Le Prototype RIME et son Application à un Corpus Médical".
Thèse de Docteur de l'Université Joseph Fourier - Grenoble I. Décembre 1988.

[BM 85] BLAIR D., MARON M. E.
"An Evaluation of Retrival Effectivenees for a Full-Text Retrival System".
Communication ACM Vol. 28 No. 3. March 1985. Pag. 289 - 299.

[BA 79] BOURNE C., ANDERSON B.
DIALOG Lab Workbook,
2nd. ed., Lockheed Information Systems, Palo Alto, California, 1979.

[BM 88] BRACHMAN R., McGUINNESS D.
"Knowledge Representation, Connectionism and Conceptual Retrieval"
Proceedings of the 11th. International ACM SIGIR Conference on Research & Development in Information Retrieval Systems. Pages 161 - 174. June 1988.

[Bru82] BRUANDET Marie-France.
"Concept Notion for Automatic and Dynamic Thesaurus Adapting"
Proceedings of SIGOA - SIGDOC, International Conference on Systems Documentation, Los Angeles, California. 22 - 23 jan. 1982

[Bru85] BRUANDET Marie-France.
"Partial Knowledge Base Model for an Intelligent Information Retrieval System"
Conference RIAO85, Grenoble, Mars 1985

[Bru87] BRUANDET Marie-France
"Outline of a knowledge base model for an Intelligent Information Retrieval System"
Proceedings of the 10th. Annual International ACMSIGIR Conference on Research & Development in Information Retrieval. Pages 33 - 43. New Orleans, June 1987.

[Chi83] CHIARAMELLA Yves.
"Un état de l'art de la Recherche en Informatique Documentaire"
Colloque CID. Paris, octobre 1983

[CDB86] CHIARAMELLA Y., DEFUDE B., BRUANDET M. F., KERKOUBA D.
"IOTA: A Full Text Information Retrieval System"
Proceedings of the 1986-ACM Conference on Research and Development in Information Retrieval. Pages 207 - 213. Pisa, September 1986.

[CD 87] CHIARAMELLA Y., DEFUDE B.
"A prototype of an Intelligent System for Information Retrieval : IOTA"
Information Processing and Management, IP&M vol 23 No.4. 1987. pag 285-303.

[Cro83] CROFT W. B.

"A network organisation used for document retrieval"

Proceedings of the 6th ACM SIGIR Conference, Research and Development in Information Retrieval. Bethesda, 1983.

[Cro86] CROFT W. B.

"User-Specified Domain Knowledge for Document Retrieval".

Proceedings of the 1986-ACM Conference on Research and Development in Information Retrieval. Pages 201 - 206. Pisa, September 1986.

[Def86] DEFUDE Bruno.

"Etude et Rélisation d'un Système Intelligent de Recherche d'Informations: Le prototype IOTA".

Thèse de Docteur de l'Institut National Polytechnique de Grenoble. Juillet 1986.

[DeJ86] De JACO Dario

"An Information Retrieval System Based on Artificial Intelligence Techniques"

Proceedings of the 1986-ACM Conference on Research and Development in Information Retrieval. Pages 214 - 219. Pisa, September 1986.

[EW 86] EL-HAMDOUCHI A., WILLETT P.

"Hierarchic Document Clustering Using Ward's Method".

Proceedings of the 1986-ACM Conference on Research and Development in Information Retrieval. Pages 149 - 156. Pisa, September 1986.

[IBM79] IBM WORLD TRADE CORPORATION.

IBM System/370 (OS/VS). "Storage and Information Retrieval System/Vertical Storage (STAIRS/VS)"

Reference Manual.

[Ker84] KERKOUBA Dalila.

"Une Méhode d'Indexation Automatique des Documents Fondée sur l'Exploitation de Leurs Proprietés Structurelles".

Thèse de Docteur de Troisième Cycle Informatique, INPG, Nov. 1984.

[MMN86] MARTIN P., MACLEOD I., NORDIN B.

"A Design of a Distributed Full Text Retrieval System".

Proceedings of the 1986-ACM Conference on Research and Development in Information Retrieval. Pages 131 - 137. Pisa, September 1986.

[NLM79] NATIONAL LIBRARY OF MEDICINE.
"MEDLARS, The Computerized Literature Retrieval Services of the National Library of Medicine".
Departement of Health, Education and Welfare, Publication NIH 79-1286, January 1979.

[Nie88] NIE Jianyun.
"An Outline of a General Model for Information Retrieval Systems"
Proceedings of the 11th. International ACM SIGIR Conference on Research & Development in Information Retrieval Systems. Pages 495 - 506. June 1988.

[Nie89] NIE Jianyun.
"Interrogation en Langue Naturelle des Systèmes de Recherche d'Information"
Thèse de Docteur de l'Université Joseph Fourier. A apparaître.

[Pal89] PALMER Patrick.
"Analyse de Surface de la Langue Naturelle. Application à l'Indexation Automatique de Documents"
Thèse de Docteur de l'Université Joseph Fourier. Grenoble. A apparaître.

[Rij79] van RIJSBERGEN C. J.
"Information Retrieval".
Second Edition, Butterworths. London 1979.

[Rij86] van RIJSBERGEN C.J.
"A New Theoretical Framework for Information Retrieval"
Proceedings of the 1986-ACM Conference on Research and Development in Information Retrieval. Pages 194 - 200. Pisa, September 1986.

[RIAO85] Conférence RIAO85
Actes. Grenoble, Mars 1985

[Sal71] SALTON Gerard.
"The SMART Retrieval System - Experiment in Automatic Document Processing"

Prentice - Hall, Englewood Cliffs, New Jersey. 1971

[SM 83] SALTON G., McGIL M.
"Introduction to Modern Information Retrieval".
International Student Edition. McGraw-Hill. 1983

[Sal86] SALTON Gerard.
"Recent Trends in Automatic Information Retrieval"
Proceedings of the 1986-ACM Conference on Research and Development in Information Retrieval. Pages 1 - 10. Pisa, September 1986.

[Spa72] SPARCK JONES Karen.
"A statistical Interpretation of Term Specificity and its Application in Retrieval.
Journal of Documentation, 28:1, pages 11-21. March 1972.

[Ton85] TONG R. M. et al.
"RUBRIC: An Environment for Full Text Information Retrieval"
Proceedings of the 8th. International ACM SIGIR Conference on Research and Development in Information Retrieval. Montreal, June 1985.

[Ton87] TONG R. M. et al.
"Conceptual Information Retrieval using RUBRIC"
Proceedings of the 10th. Annual International ACMSIGIR Conference on Research and Development in Information Retrieval. New Orleans, June 1987.

[Voo86] VOORHES Ellen M.
"The efficiency of Inverted Index and CVluster Searches"
Proceedings of the 1986-ACM Conference on Research and Development in Information Retrieval. Pages 164 - 174. Pisa, September 1986.

Systèmes de Bases de Données et d'Information Bureautique

[ALP89] ANTUNES F., LOPEZ M., PEDRAZA E.
"Evaluation de Préconditions des Activités de Bureau"

Actes du 4 Simposio Brasileiro de Banco de Datos. Campinas, Brasil, 5-7 avril 1989.

[Ban88] BANCILHON François

"Object-Oriented Database Systems"

Rapport Technnique Altaïr 16-88. 19 janvier 1988. Invited Lecture 7th. ACM SIGART - SIGMOD - SIGACT Symposium on Principles of Database Systems. Austin, Texas. March 1988.

[BDN88] BENZAKEN V., DELOBEL C., NDALA J. B.

"Gestionnaires de Mémoire et d'Objets"

Actes des Quatrièmes Journées Bases de Données Avancées. Pages 233 - 266, Tome 2. Benodet, 17 - 20 mai 1988.

[BJM88] BERARD P., JIMENEZ C., MARTIN J. P., DE LIMA J. V.

"Traitement des documents dans l'OIS. Spécifications Version1"

Projet ESPRIT 231 -D.O.E.O.I.S. Document BUL-DH-10.01. Mars 1988.

[Che76] CHEN Peter.

"The Entity-Relationship Model - Toward a Unified View of Data"

ACM Transactions on Database Systems. Vol 1 No.1, march 1976. pag 9-36

[Chr83] CHRISTODOULAKIS S. et al.

Computer System Research Group, University of Toronto.

"A Multimedia Office Filling System".

Proceedings of the 9th. conference on VLDB, Florence 1983. Pag 2 - 7.

[Chr84] CHRISTODOULAKIS S.et all. University of Toronto.

"Development of a multimedia Information System for an Office Environment".

Proceedings of the 10th. conf. on VLDB, Singapour. 1984.

[Chr85] CHRISTODOULAKIS Stavros.

"Office Filing"

Article 4 In D. Tsichritzis (ed), Office Automation, Springer Verlag, 1985.

[CF 84] CHRISTODOULAKIS S., FALOUTSOS Ch.

"Signature Files: A Access Method for documents and its Analytical Performance Evaluation".

ACM Transactions on Office Information Systems, Vol. 2, No. 4. October 1984. Pag 267 - 288.

[CF 84b] CHRISTODOULAKIS S., FALOUTSOS Ch.

"Design Considerations for a Message File Server".

IEEE Transactions on Software Engeneering, Vol. SE-10, N0. 2. March 1984. Pag 201 - 210.

[CT 83] CHRISTODOULAKIS S., TSICHRITZIS D.

"Message Files".

ACM Transactions on Office Information Systems, Vol. 1. No. 1. January 1983. Pag 88 - 98.

[ECM85] EUROPEAN COMPUTER MANUFACTURE ASSOCIATION.

"Standard ECMA-101. Office Document Architecture".

September 1895.

[ESP86a] ESPRIT PROJECT DOEOIS

"Results of the Data Analysis"

Esprit Project No.231 Design and Operational Evaluation of Office Information Servers. Document IAO-MCR-8607-2. july1986.

[ESP86b] ESPRIT PROJECT DOEOIS

"Analysis on Information Handling and Manipulation Activities in Offices"

Esprit Project No.231 Design and Operational Evaluation of Office Information Servers. Deliverable2A2. august1986.

[ESP87] ESPRIT PROJECT DOEOIS

"Office Procedures: A Formalism and Support Environlent"

Document OIS-OP-02.01. ETW Conference Proceedings. sept 1987.

[ESP88] ESPRIT PROJECT DOEOIS.

"BNF Specification for Complete OP and FMQL languages".

BULL report. January 1988.

[ESP88a] ESPRIT PROJECT DOEOIS.

"Traitement des documents dans l'OIS - Spécifications version 1".
BULL report. Document BUL-DH-04.01 Avril 1988.

[Fal86] FALOUTSOS Christos.
"Integrated Access Methods for Messages using Signature Files".
Methods and Tools for Office Systems. IFIP. Edited by G. BRACCHI and D. TSICHRITZIS. Pisa, 22 -24 October 1986.

[FC 87] FALOUTSOS C., CHRISTODOULAKIS S.
"Description and Performance Analysis of Signature File Methods for Office Filing"
ACM Transactions on Office Information Systems. Vol. 5, No.3, july 1987. pag 237-257.

[Gib86] GIBBS Simon et all. Research Center of Crete.
"Muse: A Multimedia Filing System"
IEEE Software, March 1987. pag 4 - 15.

[GT 83] GIBBS S., TSICHRITZIS D.
"A Data Modeling Approach for Office Information Systems"
ACM Transactions on Office Information Systems, Vol 1, No.4. October 1983. pag 299-319.

[GZC87] GÜTING R., ZICARI R., CHOY D.
"An Algebra for Structured Office Documents"
IBM Almaden Research Center. Research Report 5559. March 1987. San José, California.

[Jim85] JIMENEZ GUARIN Claudia.
"Recherche par le Contenu de Textes Structurés dans un Environnement Bureautique".
Rapport de D.E.A., ENSIMAG, sept. 1985.

[Jim87a] JIMENEZ GUARIN Claudia.
"Specification of Search by Content Utilities in an Office Information Server".
Esprit Project No.231 Design and Operational Evaluation of Office Information Servers. Deliverable 2D3. March 1987.

[Jim87b] JIMENEZ GUARIN Claudia.

"Report on Implementation of Search by Content Utilities".

Esprit Project No.231 Design and Operational Evaluation of Office Information Servers. Document BUL-DH-01.01. Sept 1987.

[Jim88] JIMENEZ GUARIN Claudia.

"Search By Content on Documents in an Office Information System".

Proceedings of the 11th. International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM- SIGIR June 1988. Grenoble.

[KKW83] KING R., KORTH H., WILLNER B.

"Design of a Document Filing and Retrieval Service".

Proceedings of the ACM conf. on Office Information Systems. May 1983. Pag. 96 - 101.

[Lar83] LARSON Per-Ake.

"A method for Speeding Up Text Retrieval".

Proceedings of the ACM conf. on Office Information Systems. May 1983. Pag 117 - 123.

[LG 89] DeLIMA J., GALY H.

"L'Intégration de la Structure Logique des Documents Multimédia aux SGBD"

Soumis aux Vèmes Journées Base de Données Avancées" . fevrier 1989.

[Lop88] LOPEZ Mauricio.

"Data Definition and Manipulation Services".

ESPRIT PROJECT D.O.E.O.I.S. BULL report. Document BUL-DM-01.02. Avril 1988.

[LPV83] . LOPEZ M., PALAZZO OLIVEIRA J., VELEZ F.

"The TIGRE Data Model".

Rapport de recherche No. 2. Nov. 1983, Centre de Recherches BULL, Laboratoire IMAG, Grenoble.

[Mar87] MARTIN Jean Pierre

"Document Representation in an Office Information Server".

Esprit Project No.231 Design and Operational Evaluation of Office Information Servers. Deliverable 2B3.March 1987.

[NT 85] NIERSTRASZ O. M., TSICHRITZIS D. C.

"An Object-Oriented Environment for OIS Applications"

Proceedings of the 11th. conference on VLDB, Stockholm 1985. Pag 335 - 345.

[Oli86] OLIVARES Judith.

"Traitement logique de l'intégrité et de l'organisation sémantique des connaissances dans les systèmes de gestion de Bases de Données"

Thèse de Docteur de l'Institut National Polytechnique de Grenoble. Juin 1986.

[Ped88] PEDRAZA Esperanza.

"SGBD Sémantiques pour un Environnement Bureautique : Integrité et Gestion de Transactions". Chapitre 5 : "Serveur d'Information pour la Bureautique (SIB)". Projet DOEOIS.

Thèse de Docteur Ingenieur. Université Joseph Fourier. Novembre 1988. Grenoble.

[PTS88] PUCHERAL P., THEVENIN J. M., STEFFEN H.

"Géode : Un Gestionnaire d'Objets Extensible Orienté Grande Mémoire"

Actes des IVèmes Journées Bases de Données Avancées. Benodet, 17-20 mai 1988. Tome 2, pag. 267-294.

[Rab85] RABITTI Fausto.

"A Data Model for Multimedia Documents

Article 10 In D. Tsichritzis (ed), Office Automation, Springer Verlag, 1985.

[RZ 84] RABITTI F., ZIZKA J.

"Evaluation of Access Methods to Text Documents in Office Systems".

Proceedings 3rd. Joint ACM-BCS Symposium on Research and Development in Information Retrival. 1984.

[Sac84] SACCO Giovanni Maria.

"OTTER- An information Retrival System for Office Automation".

Proceedings of the ACM conf. on Office Information Systems. June 1984. Pag. 104 - 112.

[Sac86] SACCO Giovanni Maria.

"The Fact Model: A semantic Data Model for Complex Databases"

Esprit Project No.231 Design and Operational Evaluation of Office Information Servers. Deliverable1B2. feb1986.

[TC 82] TSICHRITZIS D., CRHISTODOULAKIS S.
"Message Files"
Proceedings of the ACM COnference on Office Information Systems, june 21-23 1982. Philadelphia, Pennsylvania. pag. 110 - 112

[Tsi84] TSICHRITZIS Dennis.
"Message Adressing Schemes".
ACM Transactions on Office Information Systems, Vol. 2. No. 1. January 1984. Pag 58 - 77.

[Tsi85] TSICHRITZIS Dennis.
"Office Automation Tools"
Conférence dans Bases de Données Avancées. INRIA, St. Pierre en Chartreuse, mars 1985

[Vel84] VELEZ Fernando.
"Un Modèle et un Langage pour les Bases de Données Generalisées". Projet TIGRE.
Thèse de Docteur Ingénieur INPG. Septembre 1984.

[Vel85a] VELEZ Fernando.
"La prise en compte des Documents Structurés dans un SGBD: Aspects Modèle, Langage et Architecture du Système".
Journées "Bases de Données Avancées", INRIA-IMAG, Saint Pierre de Chartreuse. Mars 85.

[Vel85b] Velez Fernando.
"LAMBDA: An Entity-Relationship based query language for the retrival of structured documents".
4th. Int. Conf. on Entity-Relationship approach Chicago. Oct. 1985.

Méthodes de Compression

[AF 87] APOSTOLICO A., FRAENKEL A.

"Robust Transmission of Unbounded Strings Using Fibonacci Representations".

IEEE Transactions on Information Theory, Vol IT33, No. 2, March 1987.

[CFK88] CHOUEKA Y., FRAENKEL A., KLEIN S.

"Compression of Concordances in Full Text Retrieval Systems"

Proceedings of the 11th. International ACM SIGIR Conference on Research & Development in Information Retrieval Systems. Pages 597 - 612. June 1988.

[FH 69] FILES J. R., HUSKEY H. D.

"An Information Retrieval System based on Superimposed Coding".

Proceedings AFIPS Fall Joint Computer. Conference. Vol 35. AFIPS Press, Arlington, Va. 1969.

[Knu73] KNUTH, D.E.

"The Art of Computer Programming, Vol 3".

Addison-Wesley, Reading, Mass. 1973

[PB 80] PFALTZ J. L., BERMAN W. J.

"Partial-match Retrieval Using Indexed Descriptor Files".

CACM, September 1980, Vol 23, No. 9. pag 522 - 528

[PW 87] POGUE C., WILLETT P.

"Use of Text Signatures for a Document Retrieval in a Highly Parallel Environment".

Parallel Computing 4. 1987. Pag. 259 - 268.

[Rob79] ROBERTS C. S.

"Partial-match Retrieval via the method of Superimposed Codes".

Proceedings IEEE, Vol. 67, No. 12. Dec. 1979, pag 1624 - 1642.

[Sed84] SEDGEWICK Robert.

"Algorithms"

Addison-Wesley Series in Computer Science. Addison-Wesley Publishing Company. 1984.

Machines Spécialisées

[BUL84] CENTRE DE RECHERCHE BULL.
"Le Filtre SCHUSS".
Juin 1984. Paris.

[BUL85] CENTRE DE RECHERCHE BULL.
"Le Projet SCHUSS du Service Intelligence Artificielle".
Février 1985. Paris.

[BUL86] CENTRE DE RECHERCHE BULL.
"Le Projet SCHUSS. Documentation Technique, Manuel d'Utilisation ".
Paris.

[CGM88] COUPRIE M., GARCIA J., MARECHAL T., TERRAL D.
"µSyC : Un coprocesseur pour la machine parallèle DDC"
Actes des 4èmes Journées Bases de Données Avancées BD3. Tome 2, pag. 97 - 122. 17 - 20 Mai 1988. Benodet.

[CG 88] COUPRIE M., GONZALEZ-RUBIO R.
"A Parallel Multiprocessor Machine Dedicated to Relational and Deductive Databases"
Proceeding of the 11th. International ACM SIGIR Conference on Research & Development in Information Retrieval. Pag. 417 - 431. 13 - 15 June 1988. Grenoble.

[CL 80] CROZET J. M., LYON-CAEN R.
"Microprocesseurs et microordinateurs".
MASSON. 1980.

[CPW87] CARROLL D., POGUE C., WILLET P.
"Bibliographic Pattern Matching Using the ICL Distributed Array Processor"
JAMSocInfSci, 1987

[DGG86] DeWITT D., GERBER R., GRAEFE G., HEYTENS M., KUMAR K., MURALIKRISHNA M.
"GAMMA: a High Performance Dataflow Database Machine".
Internal report, CSD, University of Wisconsin, 1986.

[DEW79] DeWITT David.
"DIRECT - A Multiprocessor Organisation for Supporting Relational Database Management Systems"
IEEE Transactions on Computers. Vol c-28, No. 6, june 1979.

[GRT84] GONZALEZ RUBIO R., ROHMER J., TERRAL D. Centre de Recherche BULL.
"The SCHUSS Filtre: A Processor for Non-numerical Data processing".
Proceedings of the 11th. International Symposium on Computer Architecture. Ann Arbor, 1984.

[Has81] HASKIN Roger.
"Special-Purpose Processors for Text Retrival".
Database Engineering Vol. 4. Sept. 1981. Pag 16 - 29.

[Hol79] HOLLAAR LEE A.
"Text Retrival Computers".
IEEE Computer. March 1979 Pag 40 - 50.

[ICL85] ICL Technical Journal
"Description of the CAFS system and its applications".
Volume 4, Issue 4. November 1985

[Sch85] SCHOLL Michel.
"Architecture pour le Filtrage dans les Bases de Données Relationnelles"
Thèse de Docteur d'Etat ès Sciences de l'Institut National Polytechnique de Grenoble. Juin 1885.

[VER86] VERSO Jules.
"VERSO : A Data Base Machine Based on Non 1NF Relations"
INRIA, Rapport de Recherche International No. 523. Mai 1986.

Autres Domaines

[AU 77] AHO A., ULLMAN J.
"Principles of Compiler Design"
ADDISON-WESLEY, 1977.

[HU 77] HOPCROFT J., ULLMAN J.
"Formal Languages and Their Relation to Automata"
ADDISON-WESLEY, 1977.

[LRS76] LEWIS II P.M., ROSENKRANTZ D.J., STEARNS R.E.
"Compiler Design Theory".
ADDISON-WESLEY 1976.

[Mul68] MULLER Charles.
"Initiation à la Statistique Linguistique"
Larousse, "Langue et Langage". 1968.

Bibliographie par Ordre Alphabetique

[ALP89] ANTUNES F., LOPEZ M., PEDRAZA E.
"Evaluation de Préconditions des Activités de Bureau"
Actes du 4 Simposio Brasileiro de Banco de Datos. Campinas, Brasil, 5-7 avril 1989.

[AF 87] APOSTOLICO A., FRAENKEL A.
"Robust Transmission of Unbounded Strings Using Fibonacci Representations".
IEEE Transactions on Information Theory, Vol IT33, No. 2, March 1987.

[AU 77] AHO A., ULLMAN J.
"Principles of Compiler Design"
ADDISON-WESLEY, 1977.

[BA 79] BOURNE C., ANDERSON B.
DIALOG Lab Workbook,
2nd. ed., Lockheed Information Systems, Palo Alto, California, 1979.

[Ban88] BANCILHON François
"Object-Oriented Database Systems"

Rapport Technnique Altaïr 16-88. 19 janvier 1988. Invited Lecture 7th. ACM SIGART - SIGMOD - SIGACT Symposium on Principles of Database Systems. Austin, Texas. March 1988.

[BCB86] Di BENIGNO M. K., CROSS G. R., DEBESSONET C. G.

"COREL - A Conceptual Retrieval System".

Proceedings of the 1986-ACM Conference on Research and Development in Information Retrieval. Pages 144 - 148. Pisa, September 1986.

[BDN88] BENZAKEN V., DELOBEL C., NDALA J. B.

"Gestionnaires de Mémoire et d'Objets"

Actes des Quatrièmes Journées Bases de Données Avancées. Pages 233 - 266, Tome 2. Benodet, 17 - 20 mai 1988.

[Ber88] BERRUT Catherine.

"Une Méthode d'Indexation Fondée sur l'Analyse Sémantique de Documents Spécialisés. Le Prototype RIME et son Application à un Corpus Médical".

Thèse de Docteur de l'Université Joseph Fourier - Grenoble I. Décembre 1988.

[BJM88] BERARD P., JIMENEZ C., MARTIN J. P., DE LIMA J. V.

"Traitement des documents dans l'OIS. Spécifications Version1"

Projet ESPRIT 231 -D.O.E.O.I.S. Document BUL-DH-10.01. Mars 1988.

[BM 85] BLAIR D., MARON M. E.

"An Evaluation of Retrival Effectivenees for a Full-Text Retrival System".

Communication ACM Vol. 28 No. 3. March 1985. Pag. 289 - 299.

[BM 88] BRACHMAN R., McGUINNESS D.

"Knowledge Representation, Connectionism and Conceptual Retrieval"

Proceedings of the 11th. International ACM SIGIR Conference on Research & Development in Information Retrieval Systems. Pages 161 - 174. June 1988.

[BP 86] BERRUT C., PALMER P.

"Solving grammatical Ambiguities within a surface syntactical parser for automatic indexing"

Proceedings of the 1986-ACM Conference on Research and Development in Information Retrieval. Pages 123 - 213. Pisa, September 1986.

[Bru82] BRUANDET, Marie-France.
"Concept Notion for Automatic and Dynamic Thesaurus Adapting"
Proceedings of SIGOA - SIGDOC, International Conference on Systems Documentation, Los Angeles, California. 22 - 23 jan. 1982

[Bru85] BRUANDET Marie-France.
"Partial Knowledge Base Model for an Intelligent Information Retrieval System"
Conference RIAO85, Grenoble, Mars 1985

[Bru87] BRUANDET Marie-France
"Outline of a knowledge base model for an Intelligent Information Retrieval System"
Proceedings of the 10th. Annual International ACMSIGIR Conference on Research & Development in Information Retrieval. Pages 33 - 43. New Orleans, June 1987.

[BUL84] CENTRE DE RECHERCHE BULL.
"Le Filtre SCHUSS".
Juin 1984. Paris.

[BUL85] CENTRE DE RECHERCHE BULL.
"Le Projet SCHUSS du Service Intelligence Artificielle".
Février 1985. Paris.

[BUL86] CENTRE DE RECHERCHE BULL.
"Le Projet SCHUSS. Documentation Technique, Manuel d'Utilisation ".
Paris.

[CDB86] CHIARAMELLA Y., DEFUDE B., BRUANDET M. F., KERKOUBA D.
"IOTA: A Full Text Information Retrieval System"
Proceedings of the 1986-ACM Conference on Research and Development in Information Retrieval. Pages 207 - 213. Pisa, September 1986.

[CD 87] CHIARAMELLA Y., DEFUDE B.
"A prototype of an Intelligent System for Information Retrieval : IOTA"
Information Processing and Management, IP&M vol 23 No.4. 1987. pag 285-303.

[CF 84] CHRISTODOULAKIS S., FALOUTSOS Ch.

"Signature Files: A Access Method for documents and its Analytical Performance Evaluation".

ACM Transactions on Office Information Systems, Vol. 2, No. 4. October 1984. Pag 267 - 288.

[CF 84b] CHRISTODOULAKIS S., FALOUTSOS Ch.

"Design Considerations for a Message File Server".

IEEE Transactions on Software Engeneering, Vol. SE-10, N0. 2. March 1984. Pag 201 - 210.

[CFK88] CHOUEKA Y., FRAENKEL A., KLEIN S.

"Compression of Concordances in Full Text Retrieval Systems"

Proceedings of the 11th. International ACM SIGIR Conference on Research & Development in Information Retrieval Systems. Pages 597 - 612. June 1988.

[CG 88] COUPRIE M., GONZALEZ-RUBIO R.

"A Parallel Multiprocessor Machine Dedicated to Relational and Deductive Databases"

Proceeding of the 11th. International ACM SIGIR Conference on Research & Development in Information Retrieval. Pag. 417 - 431. 13 - 15 June 1988. Grenoble.

[CGM88] COUPRIE M., GARCIA J., MARECHAL T., TERRAL D.

"µSyC : Un coprocesseur pour la machine parallèle DDC"

Actes des 4èmes Journées Bases de Données Avancées BD3. Tome 2, pag. 97 - 122. 17 - 20 Mai 1988. Benodet.

[Che76] CHEN Peter.

"The Entity-Relationship Model - Toward a Unified View of Data"

ACM Transactions on Database Systems. Vol 1 No.1, march 1976. pag 9-36

[Chi83] CHIARAMELLA Yves.

"Un état de l'art de la Recherche en Informatique Documentaire"

Colloque CID. Paris, octobre 1983

[Chr83] CHRISTODOULAKIS S. et al.

Computer System Research Group, University of Toronto.

"A Multimedia Office Filling System".
Proceedings of the 9th. conference on VLDB, Florence 1983. Pag 2 - 7.

[Chr84] CHRISTODOULAKIS S.et all. University of Toronto.
"Development of a multimedia Information System for an Office Environment".
Proceedings of the 10th. conf. on VLDB, Singapour. 1984.

[Chr85] CHRISTODOULAKIS Stavros.
"Office Filing"
Article 4 In D. Tsichritzis (ed), Office Automation, Springer Verlag, 1985.

[CL 80] CROZET J. M., LYON-CAEN R.
"Microprocesseurs et microordinateurs".
MASSON. 1980.

[CPW87] CARROLL D., POGUE C., WILLET P.
"Bibliographic Pattern Matching Using the ICL Distributed Array Processor"
JAMSocInfSci, 1987

[Cro83] CROFT W. B.
"A network organisation used for document retrieval"
Proceedings of the 6th ACM SIGIR Conference, Research and Development in Information Retrieval. Bethesda, 1983.

[Cro86] CROFT W. B.
"User-Specified Domain Knowledge for Document Retrieval".
Proceedings of the 1986-ACM Conference on Research and Development in Information Retrieval. Pages 201 - 206. Pisa, September 1986.

[CT 83] CHRISTODOULAKIS S., TSICHRITZIS D.
"Message Files".
ACM Transactions on Office Information Systems, Vol. 1. No. 1. January 1983. Pag 88 - 98.

[Def86] DEFUDE Bruno.

"Etude et Rélisation d'un Système Intelligent de Recherche d'Informations: Le prototype IOTA".

Thèse de Docteur de l'Institut National Polytechnique de Grenoble. Juillet 1986.

[DeJ86] De JACO Dario

"An Information Retrieval System Based on Artificial Intelligence Techniques"

Proceedings of the 1986-ACM Conference on Research and Development in Information Retrieval. Pages 214 - 219. Pisa, September 1986.

[DGG86] DeWITT D., GERBER R., GRAEFE G., HEYTENS M., KUMAR K., MURALIKRISHNA M.

"GAMMA: a High Performance Dataflow Database Machine".

Internal report, CSD, University of Wisconsin, 1986.

[DEW79] DeWITT David.

"DIRECT - A Multiprocessor Organisation for Supporting Relational Database Management Systems"

IEEE Transactions on Computers. Vol c-28, No. 6, june 1979.

[ECM85] EUROPEAN COMPUTER MANUFACTURE ASSOCIATION.

"Standard ECMA-101. Office Document Architecture".

September 1895.

[ESP86a] ESPRIT PROJECT DOEOIS

"Results of the Data Analysis"

Esprit Project No.231 Design and Operational Evaluation of Office Information Servers. Document IAO-MCR-8607-2. july1986.

[ESP86b] ESPRIT PROJECT DOEOIS

"Analysis on Information Handling and Manipulation Activities in Offices"

Esprit Project No.231 Design and Operational Evaluation of Office Information Servers. Deliverable2A2. august1986.

[ESP87] ESPRIT PROJECT DOEOIS

"Office Procedures: A Formalism and Support Environlent"

Document OIS-OP-02.01. ETW Conference Proceedings. sept 1987.

[ESP88] ESPRIT PROJECT DOEOIS.
"BNF Specification for Complete OP and FMQL languages".
BULL report. January 1988.

[ESP88a] ESPRIT PROJECT DOEOIS.
"Traitement des documents dans l'OIS - Spécifications version 1".
BULL report. Document BUL-DH-04.01 Avril 1988.

[EW 86] EL-HAMDOUCHI A., WILLETT P.
"Hierarchic Document Clustering Using Ward's Method".
Proceedings of the 1986-ACM Conference on Research and Development in Information Retrieval. Pages 149 - 156. Pisa, September 1986.

[Fal86] FALOUTSOS Christos.
"Integrated Access Methods for Messages using Signature Files".
Methods and Tools for Office Systems. IFIP. Edited by G. BRACCHI and D. TSICHRITZIS. Pisa, 22 -24 October 1986.

[FC 87] FALOUTSOS C., CHRISTODOULAKIS S.
"Description and Performance Analysis of Signature File Methods for Office Filing"
ACM Transactions on Office Information Systems. Vol. 5, No.3, july 1987. pag 237-257.

[FH 69] FILES J. R., HUSKEY H. D.
"An Information Retrieval System based on Superimposed Coding".
Proceedings AFIPS Fall Joint Computer. Conference. Vol 35. AFIPS Press, Arlington, Va. 1969.

[Gib86] GIBBS Simon et all. Research Center of Crete.
"Muse: A Multimedia Filing System"
IEEE Software, March 1987. pag 4 - 15.

[GRT84] GONZALEZ RUBIO R., ROHMER J., TERRAL D. Centre de Recherche BULL.
"The SCHUSS Filtre: A Processor for Non-numerical Data processing".
Proceedings of the 11th. International Symposium on Computer Architecture. Ann Arbor, 1984.

[GT 83] GIBBS S., TSICHRITZIS D.

"A Data Modeling Approach for Office Information Systems"
ACM Transactions on Office Information Systems, Vol 1, No.4. October 1983. pag 299-319.

[GZC87] GÜTING R., ZICARI R., CHOY D.
"An Algebra for Structured Office Documents"
IBM Almaden Research Center. Research Report 5559. March 1987. San José, California.

[Has81] HASKIN Roger.
"Special-Purpose Processors for Text Retrival".
Database Engineering Vol. 4. Sept. 1981. Pag 16 - 29.

[Hol79] HOLLAAR LEE A.
"Text Retrival Computers".
IEEE Computer. March 1979 Pag 40 - 50.

[HU 77] HOPCROFT J., ULLMAN J.
"Formal Languages and Their Relation to Automata"
ADDISON-WESLEY, 1977.

[IBM79] IBM WORLD TRADE CORPORATION.
IBM System/370 (OS/VS). "Storage and Information Retrieval System/Vertical Storage (STAIRS/VS)"
Reference Manual.

[ICL85] ICL Technical Journal
"Description of the CAFS system and its applications".
Volume 4, Issue 4. November 1985

[Jim85] JIMENEZ GUARIN Claudia.
"Recherche par le Contenu de Textes Structurés dans un Environnement Bureautique".
Rapport de D.E.A., ENSIMAG, sept. 1985.

Esprit Project No.231 Design and Operational Evaluation of Office Information Servers. Deliverable 2D3. March 1987.

[Jim87b] JIMENEZ GUARIN Claudia.

"Report on Implementation of Search by Content Utilities".

Esprit Project No.231 Design and Operational Evaluation of Office Information Servers. Document BUL-DH-01.01. Sept 1987.

[Jim88] JIMENEZ GUARIN Claudia.

"Search By Content on Documents in an Office Information System".

Proceedings of the 11th. International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM- SIGIR June 1988. Grenoble.

[Ker84] KERKOUBA Dalila.

"Une Méhode d'Indexation Automatique des Documents Fondée sur l'Exploitation de Leurs Proprietés Structurelles".

Thèse de Docteur de Troisième Cycle Informatique, INPG, Nov. 1984.

[KKW83] KING R., KORTH H., WILLNER B.

"Design of a Document Filing and Retrieval Service".

Proceedings of the ACM conf. on Office Information Systems. May 1983. Pag. 96 - 101.

[Knu73] KNUTH, D.E.

"The Art of Computer Programming, Vol 3".

Addison-Wesley, Reading, Mass. 1973

[Lar83] LARSON Per-Ake.

"A method for Speeding Up Text Retrieval".

Proceedings of the ACM conf. on Office Information Systems. May 1983. Pag 117 - 123.

[LG 89] DeLIMA J., GALY H.

"L'Intégration de la Structure Logique des Documents Multimédia aux SGBD"

Soumis aux Vèmes Journées Base de Données Avancées" . fevrier 1989.

[Lop88] LOPEZ Mauricio.

"Data Definition and Manipulation Services".

ESPRIT PROJECT D.O.E.O.I.S. BULL report. Document BUL-DM-01.02. Avril 1988.

[LPV83] LOPEZ M., PALAZZO OLIVEIRA J., VELEZ F.
"The TIGRE Data Model".
Rapport de recherche No. 2. Nov. 1983, Centre de Recherches BULL, Laboratoire IMAG, Grenoble.

[LRS76] LEWIS II P.M., ROSENKRANTZ D.J., STEARNS R.E.
"Compiler Design Theory".
ADDISON-WESLEY 1976.

[Mar87] MARTIN Jean Pierre
"Document Representation in an Office Information Server".
Esprit Project No.231 Design and Operational Evaluation of Office Information Servers. Deliverable 2B3.March 1987.

[MMN86] MARTIN P., MACLEOD I., NORDIN B.
"A Design of a Distributed Full Text Retrieval System".
Proceedings of the 1986-ACM Conference on Research and Development in Information Retrieval. Pages 131 - 137. Pisa, September 1986.

[Mul68] MULLER Charles.
"Initiation à la Statistique Linguistique"
Larousse, "Langue et Langage". 1968.

[Nie88] NIE Jianyun.
"An Outline of a General Model for Information Retrieval Systems"
Proceedings of the 11th. International ACM SIGIR Conference on Research & Development in Information Retrieval Systems. Pages 495 - 506. June 1988.

[Nie89] NIE Jianyun.
"Interrogation en Langue Naturelle des Systèmes de Recherche d'Information"
Thèse de Docteur de l'Université Joseph Fourier. A apparaître.

[NLM79] NATIONAL LIBRARY OF MEDICINE.
"MEDLARS, The Computerized Literature Retrieval Services of the National Library of Medicine".

Departement of Health, Education and Welfare, Publication NIH 79-1286, January 1979.

[NT 85] NIERSTRASZ O. M., TSICHRITZIS D. C.
"An Object-Oriented Environment for OIS Applications"
Proceedings of the 11th. conference on VLDB, Stockholm 1985. Pag 335 - 345.

[Oli86] OLIVARES Judith.
"Traitement logique de l'intégrité et de l'organisation sémantique des connaissances dans les systèmes de gestion de Bases de Données"
Thèse de Docteur de l'Institut National Polytechnique de Grenoble. Juin 1986.

[Pal89] PALMER Patrick.
"Analyse de Surface de la Langue Naturelle. Application à l'Indexation Automatique de Documents"
Thèse de Docteur de l'Université Joseph Fourier. Grenoble. A apparaître.

[PB 80] PFALTZ J. L., BERMAN W. J.
"Partial-match Retrieval Using Indexed Descriptor Files".
CACM, September 1980, Vol 23, No. 9. pag 522 - 528

[Ped88] PEDRAZA Esperanza.
"SGBD Sémantiques pour un Environnement Bureautique : Integrité et Gestion de Transactions". Chapitre 5 : "Serveur d'Information pour la Bureautique (SIB)". Projet DOEOIS.
Thèse de Docteur Ingenieur. Université Joseph Fourier. Novembre 1988. Grenoble.

[PTS88] PUCHERAL P., THEVENIN J. M., STEFFEN H.
"Géode : Un Gestionnaire d'Objets Extensible Orienté Grande Mémoire"
Actes des IVèmes Journées Bases de Données Avancées. Benodet, 17-20 mai 1988. Tome 2, pag. 267-294.

[PW 87] POGUE C., WILLETT P.
"Use of Text Signatures for a Document Retrieval in a Highly Parallel Environment".
Parallel Computing 4. 1987. Pag. 259 - 268.

[Rab85] RABITTI Fausto.
"A Data Model for Multimedia Documents
Article 10 In D. Tsichritzis (ed), Office Automation, Springer Verlag, 1985.

[RIAO85] Conférence RIAO85
Actes. Grenoble, Mars 1985

[Rij79] van RIJSBERGEN C. J.
"Information Retrieval".
Second Edition, Butterworths. London 1979.

[Rij86] van RIJSBERGEN C.J.
"A New Theoretical Framework for Information Retrieval"
Proceedings of the 1986-ACM Conference on Research and Development in Information Retrieval. Pages 194 - 200. Pisa, September 1986.

[Rob79] ROBERTS C. S.
"Partial-match Retrieval via the method of Superimposed Codes".
Proceedings IEEE, Vol. 67, No. 12. Dec. 1979, pag 1624 - 1642.

[RZ 84] RABITTI F., ZIZKA J.
"Evaluation of Access Methods to Text Documents in Office Systems".
Proceedings 3rd. Joint ACM-BCS Symposium on Research and Development in Information Retrival. 1984.

[Sac84] SACCO Giovanni Maria.
"OTTER- An information Retrival System for Office Automation".
Proceedings of the ACM conf. on Office Information Systems. June 1984. Pag. 104 - 112.

[Sac86] SACCO Giovanni Maria.
"The Fact Model: A semantic Data Model for Complex Databases"
Esprit Project No.231 Design and Operational Evaluation of Office Information Servers. Deliverable1B2. feb1986.

[Sal71] SALTON Gerard.
"The SMART Retrieval System - Experiment in Automatic Document Processing"

Prentice - Hall, Englewood Cliffs, New Jersey. 1971

[Sal86] SALTON Gerard.
"Recent Trends in Automatic Information Retrieval"
Proceedings of the 1986-ACM Conference on Research and Development in Information Retrieval. Pages 1 - 10. Pisa, September 1986.

[Sed84] SEDGEWICK Robert.
"Algorithms"
Addison-Wesley Series in Computer Science. Addison-Wesley Publishing Company. 1984.

[SM 83] SALTON G., McGIL M.
"Introduction to Modern Information Retrieval".
International Student Edition. McGraw-Hill. 1983

[Sch85] SCHOLL Michel.
"Architecture pour le Filtrage dans les Bases de Données Relationnelles"
Thèse de Docteur d'Etat ès Sciences de l'Institut National Polytechnique de Grenoble. Juin 1885.

[Spa72] SPARCK JONES Karen.
"A statistical Interpretation of Term Specificity and its Application in Retrieval.
Journal of Documentation, 28:1, pages 11-21. March 1972.

[TC 82] TSICHRITZIS D., CRHISTODOULAKIS S.
"Message Files"
Proceedings of the ACM COnference on Office Information Systems, june 21-23 1982. Philadelphia, Pennsylvania. pag. 110 - 112

[Ton85] TONG R. M. et al.
"RUBRIC: An Environment for Full Text Information Retrieval"
Proceedings of the 8th. International ACM SIGIR Conference on Research and Development in Information Retrieval. Montreal, June 1985.

[Ton87] TONG R. M. et al.
"Conceptual Information Retrieval using RUBRIC"

Proceedings of the 10th. Annual International ACMSIGIR Conference on Research and Development in Information Retrieval. New Orleans, June 1987.

[Tsi84] TSICHRITZIS Dennis.

"Message Adressing Schemes".

ACM Transactions on Office Information Systems, Vol. 2. No. 1. January 1984. Pag 58 - 77.

[Tsi85] TSICHRITZIS Dennis.

"Office Automation Tools"

Conférence dans Bases de Données Avancées. INRIA, St. Pierre en Chartreuse, mars 1985

[Vel84] VELEZ Fernando.

"Un Modèle et un Langage pour les Bases de Données Generalisées". Projet TIGRE.

Thèse de Docteur Ingénieur INPG. Septembre 1984.

[Vel85a] VELEZ Fernando.

"La prise en compte des Documents Structurés dans un SGBD: Aspects Modèle, Langage et Architecture du Système".

Journées "Bases de Données Avancées", INRIA-IMAG, Saint Pierre de Chartreuse. Mars 85.

[Vel85b] Velez Fernando.

"LAMBDA: An Entity-Relationship based query language for the retrival of structured documents".

4th. Int. Conf. on Entity-Relationship approach Chicago. Oct. 1985.

[VER86] VERSO Jules.

"VERSO : A Data Base Machine Based on Non 1NF Relations"

INRIA, Rapport de Recherche International No. 523. Mai 1986.

[Voo86] VOORHES Ellen M.

"The efficiency of Inverted Index and CVluster Searches"

Proceedings of the 1986-ACM Conference on Research and Development in Information Retrieval. Pages 164 - 174. Pisa, September 1986.

ANNEXE A - Le Modèle de Données FM

Le modèle de données du SIB est le **Fact Model (FM)** [Sac86] [Lop88] [ESP88] [Ped88]. Il est de type fonctionnel, fondé sur deux concepts: Les ***entités*** et les ***faits***.

Les entités sont des abstractions des objets du monde réel, et elles sont dénotées par un symbole, par exemple PERSONNE, qui représente l'ensemble des personnes possibles dans le monde réel. L'ensemble des instances possibles pour les entités constituent un **type**, dénoté par le même symbole. Ainsi, PERSONNE représente aussi bien l'ensemble possible d'instances (le type) et l'entité. Les ***faits*** représentent les relations entre les entités, et sont manipulés par des fonctions.

Les entités élémentaires, c'est-à-dire, celles qui n'ont qu'une valeur associée, sont appelées **entités littérales.** Par exemple, taille, prix, couleur sont des entités littérales. Les **entités virtuelles** représentent l'ensemble d'instances pour lesquelles il y a des faits associés. Une instance d'une entité virtuelle est donc définie par ces faits.

Exemple :

Par la définition AGE : **integer**

PERSONNE : **ENTITY**

AGE est un entité littérale est PERSONNE est une entité virtuelle. Les instances de AGE existent dans l'ensemble des entiers. Les instances de PERSONNE sont définies par l'ensemble d'instances des faits associés, par exemple le nom, la taille, la date de naissance.

En FM nous disposons de quelques types prédéfinis, parmi lesquels nous distinguons:

- Un ensemble de types de base (entier, réel, chaîne de caractères, etc) utilisés pour caractériser les entités littérales. Les instances des types de base sont atomiques par définition. Pour la modélisation des types nécessaires aux applications bureautiques, il existe des types de base permettant la définition de données multimédia. Par exemple, nous trouvons dans le SIB les types temps, durée, intervalle, ODA.

- **ENTITY**, qui est le type de toutes les entités virtuelles. Le type **DOCUMENT** est prédéfini dans le SIB pour la gestion des objets document.

D'autres types, en particulier des spécialisations d'une entité, peuvent être construits à partir des types prédéfinis par des opérations ensemblistes.

Exemple:

PERSONNE : **ENTITY** EMPLOYÉ : PERSONNE

RAPPORT : **DOCUMENT**

PERSONNE est un sous-ensemble de toutes les entités virtuelles, et EMPLOYÉ est un sous-ensemble de PERSONNEs. De même, RAPPORT est un sous-ensemble des DOCUMENTs. PERSONNE, EMPLOYÉ et RAPPORT sont des entités virtuelles.

Un fait entre n entités est décrit par une notation de la forme:

$$[\text{entité}_1, \text{entité}_2, \ldots \text{entité}_n]$$

Les faits permettent de définir les relations entre entités. Un fait défini entre une entité virtuelle et une entité littérale établit une propriété de l'entité virtuelle. Un fait défini entre deux entités virtuelles définit deux propriétés.

Exemple :

Pour la définition des*entités*

AGE : **integer** PERSONNE : **ENTITY**

NOM : **string** DÉPARTEMENT : **ENTITY**

et la définition des *faits*

[PERSONNE, AGE] [DÉPARTEMENT, NOM]

[PERSONNE, DÉPARTEMENT]

sont définies les fonctions (proprietés) :

PERSONNE -> AGE DÉPARTEMENT -> NOM

PERSONNE -> DÉPARTEMENT DÉPARTEMENT -> PERSONNE

qui indiquent que les personnes appartient à un département, et que les départements ont des personnes. Les personnes ont un age et les départements un nom.

Les relations établies entre entités via les faits peuvent être locales ou globales.

Exemple:

ÉTUDIANT : **ENTITY** COURS : **ENTITY**

[ÉTUDIANT, COURS]

[ÉTUDIANT, AGE : integer]

Le premier fait définit de manière globale que COURS est une propriété de l'entité ÉTUDIANT, et que ÉTUDIANT est une propriété de COURS.

Le deuxième établit une relation locale entre les entités ÉTUDIANT et AGE. Ici AGE est localement défini comme un sous-ensemble des entiers. Ce fait définit que AGE est une des propriétés de l'entité ÉTUDIANT. Cette définition est indépendante d'autres définitions d'AGE données ailleurs dans le schéma conceptuel de l'application.

La définition de faits peut avoir des restrictions de cardinalité, exprimées dans la forme ($Card_{min}$, $Card_{max}$). $Card_{min}$ peut être 0 ou 1, $Card_{max}$ peut être 1 ou * (un nombre quelconque)

Exemple: Les faits

[ÉTUDIANT, NUMÉRO_INSCRIPTION (1,1)]

[COURS, ÉTUDIANT(1,*)]

expriment qu'un étudiant doit avoir un numéro d'inscription et un seul, et qu'un cours doit avoir au moins un étudiant.

Les contraintes de cardinalité établissent quatre types de fonctions entre les entités concernées:

- Si $Card_{min} = 0$, la fonction est partielle.
- Si $Card_{min} = 1$, la fonction est totale.
- Si $Card_{max} = 1$, la fonction est mono-valuée.
- Si $Card_{max} = *$, la fonction est multi-valuée.

Dans FM il est aussi possible de définir des contraintes de participation aux associations. Donc, il est possible d'indiquer qu'un fait est obligatoire. Ceci se fait par une étoile ("*") avant le nom de l'entité.

Exemple:

[*EMPLOYÉ, SALAIRE : real] [ÉTUDIANT, *RAPPORT]

Définit que tout employé doit avoir un salaire ou que tout rapport doit appartenir à un étudiant. Dans le dernier fait les étudiants sont indépendants des rapports. Cette définition est équivalante à l'utilisation des contraintes de cardinalité

[EMPLOYÉ, SALAIRE (1,1) : real] [ÉTUDIANT, RAPPORT (1,*)]

Le Langage de Manipulation de Données FMQL

Le **Fact Model Query Language**, FMQL, permet la manipulation de données de FM d'une manière ensembliste. Il a, comme SQL, deux types d'énoncés de manipulation:

- L'énoncé d'interrogation **SELECT.**

- Les énoncés de mise à jour: **INSERT, DELETE, ASSERT.**

Ils portent toujours sur des faits, les entités n'existant que si elles ont au moins un fait associé [Ped88]. Au moment de l'insertion d'une entité au moins un fait lui est associé. La mise à jour modifie les faits qui lui sont propres et la suppression d'une entité a pour conséquence la suppression de tout les faits qui lui sont associés.

Par des exemples et de manière rapide nous présentons la syntaxe de ces énoncés. Une description plus détaillée se trouve dans [Lop88], [Ped88] et [ESP88] :

- L'Insertion (d'une nouvelle entité virtuelle).

```
INSERT étudiant
WITH nom = "TOTO" AND inscription = "INP889432"
```

- La Suppression (d'un fait entre deux entités existantes)

  DELETE [étudiant e, cours c]

  FROM étudiant e, cours c

  WHERE e.nom = "TOTO" **AND** c.nom = "Bases de Données"

- Mise à jour d'un fait. Elle peut soit modifier les valeurs d'un fait existant, soit en créer un nouveau.

  ASSERT étudiant.montant_bourse = 3000

  WHERE étudiant.age < 18

- L'interrogation permet d'obtenir des valeurs associés à une entité. Ces valeurs peuvent être obtenues, par l'évaluation des fonctions définies sur l'entité, qui correspondent à l'évaluation des faits que lui sont associés. Par exemple:

  SELECT e.nom

  FROM étudiant e. cours c

  WHERE c . professeur.nom = "Dupont"

  La forme générale d'un énoncé d'interrogation est:

  SELECT <entité>

  FROM <définition du contexte d'évaluation>

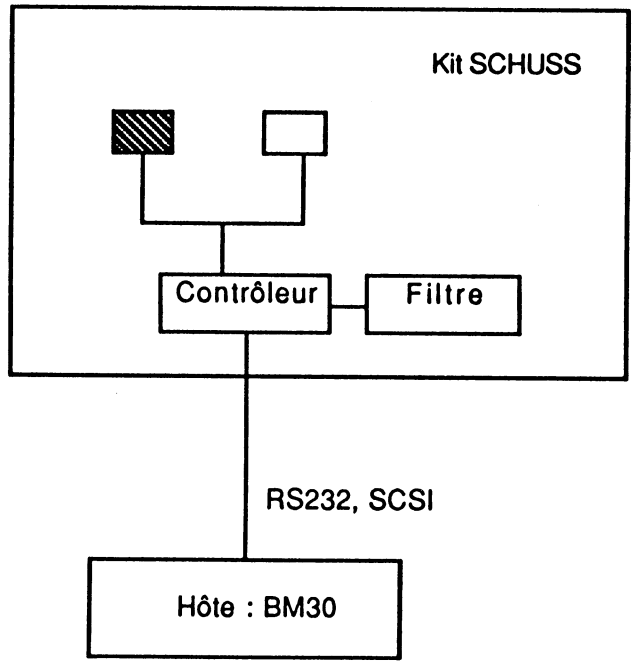
  WHERE <condition d'évaluation>

La clause **FROM** est optionnelle et la clause **WHERE** peut contenir les opérateurs ensemblistes traditionnels.

C'est de cette manière que sont définis et manipulés les schémas conceptuels des applications bureautiques dans les SIB, en utilisant le Fact Model.

ANNEXE B - Architecture de SCHUSS

SCHUSS est un prototype de machine orienté au filtrage de données réalisé par BULL [BUL84] [BUL85] [BUL86] [GRT84] [CL 80]. Il est disponible comme un "kit" avec les composants suivants :



Le "kit" SCHUSS est composé de deux disques durs, une carte contrôleur de disques et une carte du processeur SCHUSS, le filtre. Le contrôleur est constitué par un processeur Z80 et par une mémoire extensible jusqu'à 256K octets pour implanter les

mémoires tampons d'entrée/sortie et les effets de cache. Le filtre, qui est microprogrammable, a deux UAL, une mémoire de microprogrammes (PROM) de 32K octets, des microinstructions de 112 bits de longueur et une mémoire RAM de 64K octets pour le stockage des automates de filtrage. Il est vu, par la machine hôte, comme une "boîte noire" qui se comporte comme un autre périphérique, un disque par exemple. Les deux processeurs communiquent via deux interfaces normalisées, une série RS232 et une parallèle SCSI. L'interface SCSI sert à la transmission de données et d'instructions, et la RS232 sert à la mise au point de microprogrammes.

Pour l'interface logicielle entre l'hôte et le filtre il existe une couche réalisée en C. Elle est composée de deux modules séparés, un gestionnaire d'espace disque ("File Manager") et un système de filtrage de fichiers qui fournit les opérations de base pour l'accès à une base de données de type relationnelle [BUL86]. Le système de filtrage utilise sa propre gestion du disque, qui n'est pas compatible avec l'autre module.

Dans le système de filtrage de données, l'unité minimale de transfert d'une opération entrée/sortie, à partir du disque SCHUSS, est un secteur (256 octets). Les secteurs sont regroupés en pistes, avec 64 secteurs/piste (16k octets). Lors de la création de chaque base un nombre entier de pistes lui est réservé.

La structure logique d'une Base est ainsi définie :

- Une base est un ensemble de nuplets, qui peuvent avoir une longueur variable.

- Les nuplets sont formés par un nombre fixe d'attributs (maximum 16). Les attributs peuvent être de longueur variable, jusqu'à 256 octets.

- Le stockage des nuplets est organisé en partitions, qui déterminent la taille de la mémoire tampon d'entrée/sortie. La partition est toujours un nombre entier de secteurs du

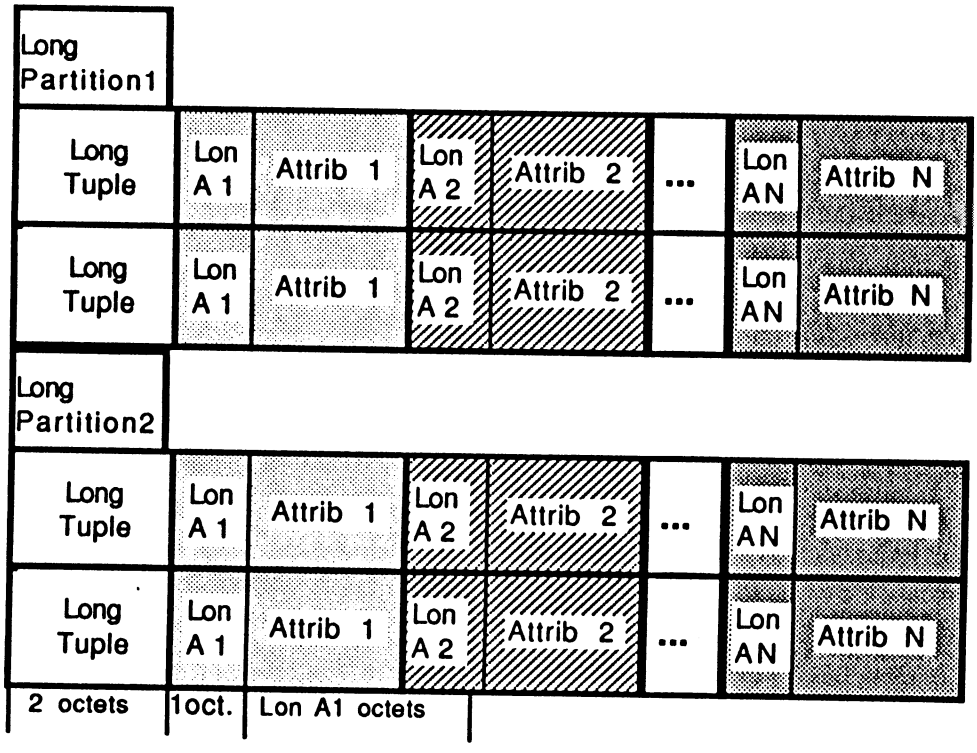
disque, et elle est déterminée lors de la création de la base. Sa longueur maximale est 32k octets. Il est impossible de stocker deux partitions "à cheval" sur un même secteur.

- Chaque base est donc considérée comme un ensemble de partitions.

Pour gérer les longueurs variables tous les composants d'une base sont stockés, précédés de leur longueur.

Exemple :

Une base définie sur deux partitions a la structure suivante :



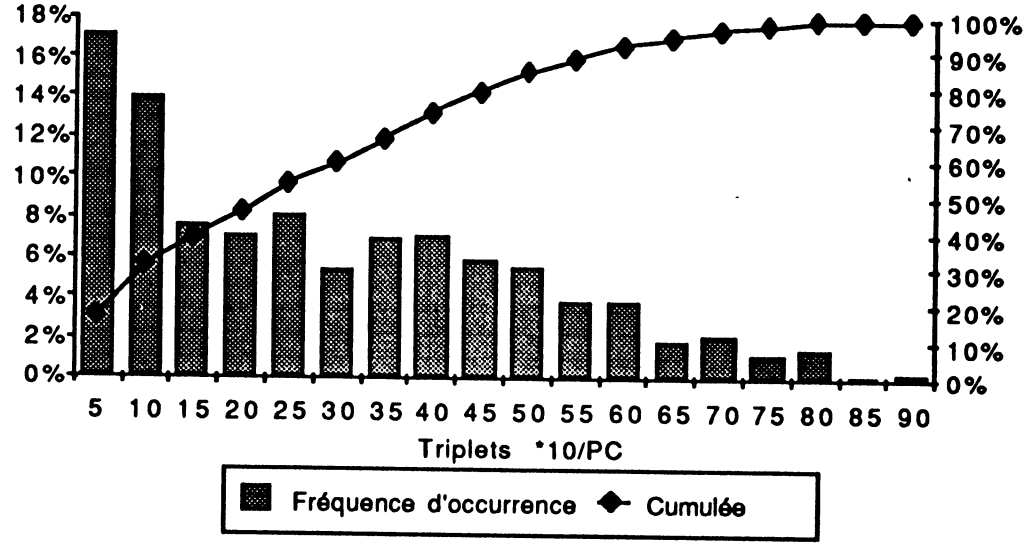
ANNEXE C - Résultats d'Expérimentation

TABLEAU DE DISTRIBUTION DE TRIPLETS/PC

Document : THESE 2

Triplets/PC	Fréquence	Cumulée
1- 50	17,112%	17,112%
51-100	14,037%	31,150%
101-150	7,620%	38,770%
151-200	7,086%	45,856%
201-250	8,155%	54,011%
251-300	5,348%	59,358%
301-350	6,952%	66,310%
351-400	7,086%	73,396%
401-450	5,882%	79,278%
451-500	5,481%	84,759%
501-550	3,877%	88,636%
551-600	3,877%	92,513%
601-650	1,872%	94,385%
651-700	2,139%	96,524%
701-750	1,203%	97,727%
751-800	1,471%	99,198%
801-850	0,134%	99,332%
851-900	0,267%	99,599%
901-950	0,134%	99,733%
951-1000	0,267%	100,000%

DISTRIBUTION DES TRIPLETS PAR PC
Document : Thèse 2



DISTRIBUTION COMPARATIVE DES TRIPLETS/PC
Document : Thèse 2

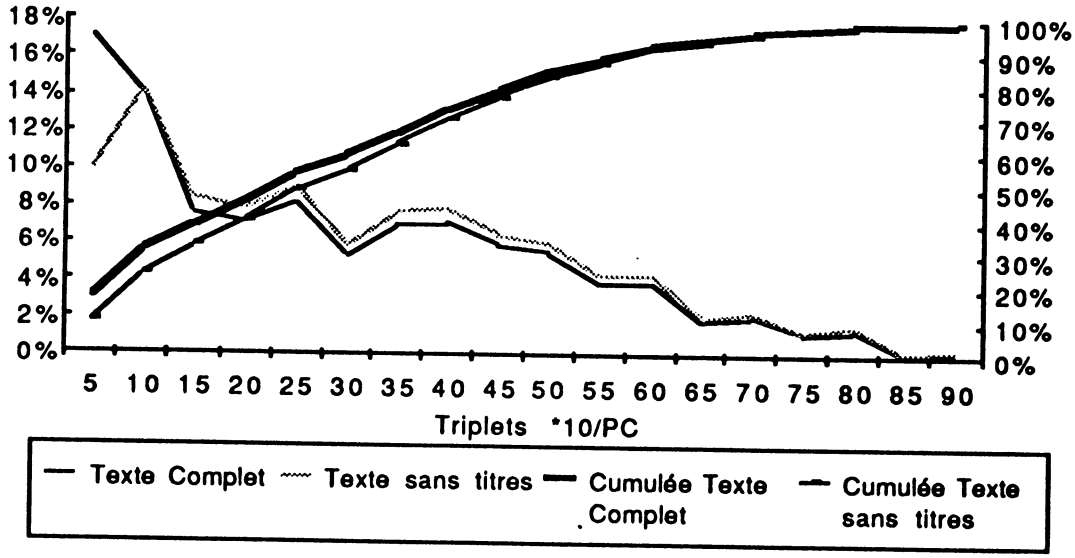


TABLEAU DE DISTRIBUTION DE TRIPLETS/PC

Document : MEDICAL

Triplets/PC	Fréquence	Cumulée
1-50	15,698%	15,698%
51-100	22,674%	38,372%
101-150	26,744%	65,116%
151-200	15,116%	80,233%
201-250	9,884%	90,116%
251-300	4,070%	94,186%
301-350	2,326%	96,512%
351-400	1,744%	98,256%
401-450	1,163%	99,419%
451-500	0,000%	99,419%
551-600	0,581%	100,000%

DISTRIBUTION DES TRIPLETS PAR PC
Document : Medical

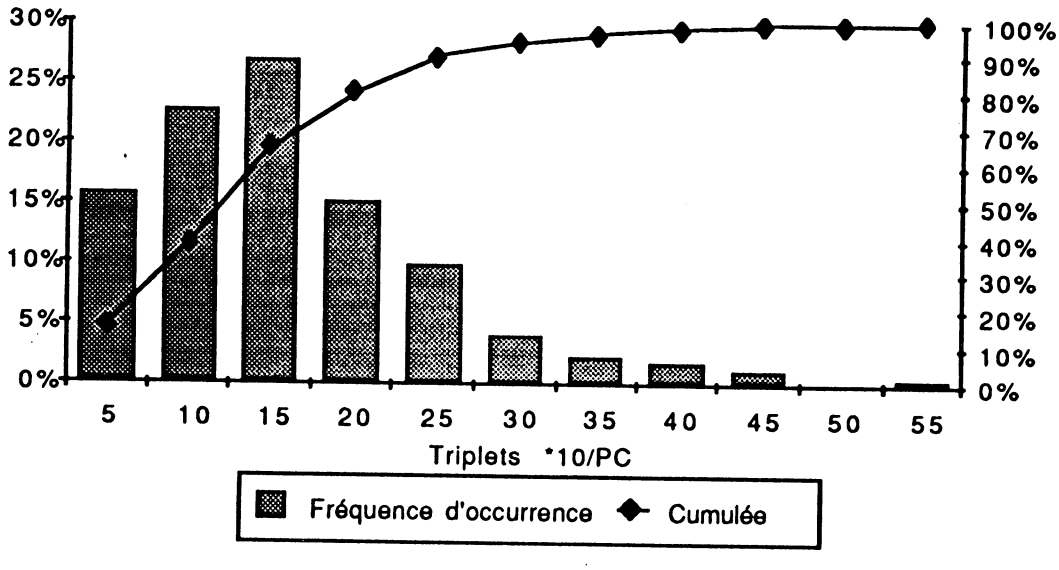
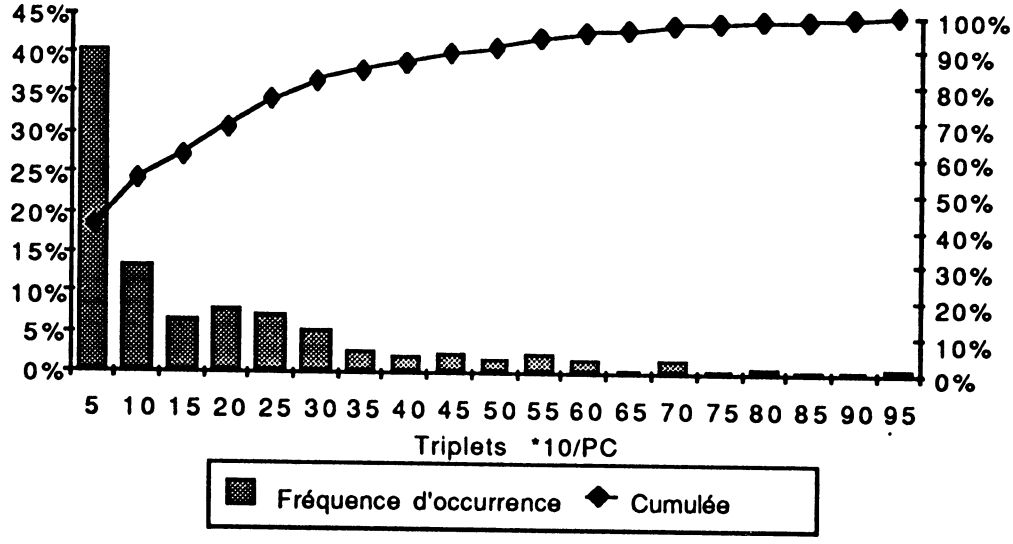


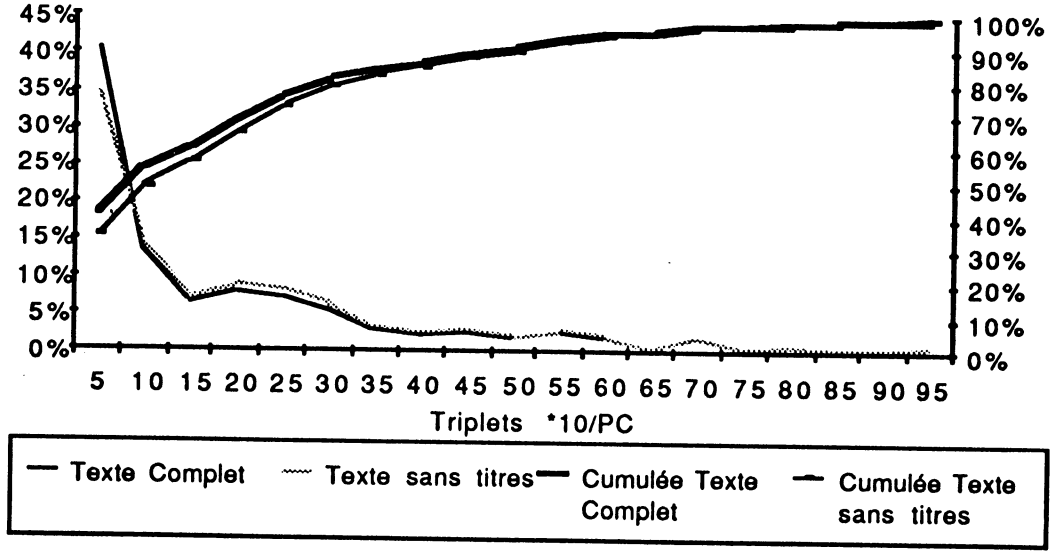
TABLEAU DE DISTRIBUTION DE TRIPLETS/PC Doc : ARTICLE

Triplets/PC	Fréquence	Cumulée
1-50	40,511%	40,511%
51-100	13,504%	54,015%
101-150	6,569%	60,584%
151-200	8,029%	68,613%
201-250	7,299%	75,912%
251-300	5,474%	81,387%
301-350	2,920%	84,307%
351-400	2,190%	86,496%
401-450	2,555%	89,051%
451-500	1,825%	90,876%
501-550	2,555%	93,431%
551-600	1,825%	95,255%
601-650	0,365%	95,620%
651-700	1,825%	97,445%
701-750	0,365%	97,810%
751-800	0,730%	98,540%
801-850	0,365%	98,905%
851-900	0,365%	99,270%
901-950	0,730%	100,000%

DISTRIBUTION DES TRIPLETS PAR PC
Document : Article



DISTRIBUTION COMPARATIVE DES TRIPLETS/PC
Document : Article



TEMPS D'EVALUATION DE REQUETES

(temps donné en secondes)

Requête	Création SignReq	Création Autom.	Filtrage Texte	Rép. Requête
document	0,007	0,216	0,28	oui
document	0,007	0,43	0,645	oui
maison	0,004	0,141	5,767	oui
maison	0,005	0,282	5,873	oui
inform*	0,003	0,158	0,229	oui
retrie	0,003	0,137	0,202	oui
inform*retrie*	0,006	0,373	0,438	oui
index	0,006	0,116	0,139	oui
*index*term*	0,007	0,237	0,609	oui
attributs	0,008	0,426	1,093	oui
structure	0,008	0,385	1,074	oui
attributs OR structure	0,013	0,54	1,26	oui
analyse AND *semant*	0,009	0,397	0,749	oui
*analys*semant* AND *linguis*	0,006	0,878	1,564	oui
*analys*semant* OR *base*connais*	0,01	0,909	1,072	oui
*analys*semant* AND *base*connais*	0,016	0,898	1,638	oui
relevance	0,005	0,214	19,83	non
relevance	0,006	0,424	19,976	non
boolean	0,003	0,163	19,363	non
boolean	0,005	0,315	19,597	non
Office	0,006	0,138	18,787	non
*Office*Inf*	0,007	0,233	19,406	non
*Office*Inf*Serv*	0,004	0,397	20,895	non

Temps d'évaluation utilisant l'index de Signatures avec élimination de Collisions

Requête	Taille Bloc de Signatures (temps en secondes, taille bloc en bits)				
	199	307	397	499	599
document	1,04	1,478	1,665	2,037	2,312
document	1,34	1,784	1,993	2,314	2,613
maison	4,01	2,588	2,044	2,372	2,815
maison	3,77	2,425	2,231	2,463	2,809
inform*	1,16	1,463	1,643	2,012	2,32
retrie	1,17	1,403	1,608	1,947	2,309
inform*retrie*	1,21	1,58	1,818	2,145	2,476
index	1,74	1,569	1,795	2,085	2,294
*index*term*	1,34	1,576	1,924	2,14	2,452
attributs	1,32	1,701	1,905	2,22	2,569
structure	1,32	1,678	1,948	2,198	2,508
attributs OR structure	1,81	2,345	2,589	3,01	3,433
analyse AND *semant*	2,65	2,789	3,107	3,886	3,825
*analys*semant* AND *linguis*	2,41	2,682	3,02	3,534	3,876
*analys*semant* OR *base*connais*	2,30	2,637	3,054	3,413	3,73
*analys*semant* AND *base*connais*	2,51	2,749	3,075	3,514	3,78
relevance	2,36	3,573	2,871	2,636	2,654
relevance	2,32	3,72	3,042	2,9	2,707
boolean	3,24	3,046	1,758	2,113	2,528
boolean	2,82	2,817	1,386	2,312	2,019
Office	4,06	3,511	3,356	3,333	3,397
*Office*Inf*	3,26	2,916	2,421	3,253	2,047
*Office*Inf*Serv*	3,36	2,888	2,457	3,399	2,086

Temps d'évaluation de reqûetes utilisant l'index de Signatures sans élimination de Collisions

Requête	Taille Bloc de Signatures (temps en secondes, taille bloc en bits)				
	199	307	397	499	599
document	0,83s	1,23 s	1,51s	1,81s	2,07s
document	0,81	1,25	1,492	1,801	2,122
maison	1,26	1,297	1,553	1,77	2,18
maison	1,12	1,257	1,539	1,773	2,158
inform*	0,96	1,284	1,524	1,835	2,176
retrie	1,04	1,204	1,512	1,79	2,193
inform*retrie*	0,85	1,215	1,485	1,755	2,16
index	1,66	1,419	1,723	1,956	2,264
*index*term*	1,05	1,295	1,704	1,896	2,232
attributs	0,82	1,228	1,492	1,744	2,151
structure	0,85	1,24	1,518	1,78	2,11
attributs OR structure	1,19	1,726	2,111	2,402	2,939
analyse AND *semant*	2,18	2,264	2,592	3,325	3,358
*analys*semant* AND *linguis*	1,39	1,763	2,115	2,586	3,008
*analys*semant* OR *base*connais*	1,32	1,672	2,067	2,475	2,902
*analys*semant* AND *base*connais*	1,39	1,687	2,057	2,512	2,881
relevance	0,82	1,25	1,619	1,795	2,132
relevance	0,79	1,255	1,553	1,792	2,102
boolean	0,86	1,178	1,445	1,803	2,082
boolean	0,82	1,17	1,45	1,781	2,054
Office	0,89	1,199	1,493	1,842	2,121
*Office*Inf*	0,88	1,206	1,503	1,825	2,116
*Office*Inf*Serv*	0,83	1,188	1,497	1,848	2,137

Comparaison des Temps Moyens d'Exécution

Filtrage Signatures et Texte (temps en secondes)

	199	307	397	499	599
Total requêtes	2,28	2,39	2,29	2,66	2,76
Sans Expr Log rép. OUI	1,76	1,75	1,87	2,18	2,50
Avec Expr Log rép. OUI	2,34	2,64	2,97	3,47	3,73
Sans Expr Log rép. NON	3,06	3,21	2,47	2,85	2,49

Filtrage Signatures Seules (temps en secondes)

	199	307	397	499	599
Total requêtes	1,07	1,37	1,68	2,00	2,33
Sans Expr Log rép. OUI	1,02	1,27	1,55	1,81	2,17
Avec Expr Log rép. OUI	1,50	1,82	2,19	2,66	3,02
Sans Expr Log rép. NON	0,84	1,21	1,51	1,81	2,11

Filtrage Texte (temps en secondes)

Total requêtes	6,98
Sans Expr Log rép OUI	1,49
Avec Expr Log rép OUI	1,26
Sans Expr Log rép NON	19,69

Temps du Filtrage Signatures et Texte / Temps de Filtrage de Texte (en pourcentage)

	199	307	397	499	599
Toutes requêtes	32,71	34,22	32,84	38,16	39,60
Sans Expr Log rép. OUI	118,7	117,7	125,8	146,3	168,0
Avec Expr Log rép. OUI	185,9	210,1	236,2	276,2	296,7
Sans Expr Log rép. NON	15,53	16,30	12,54	14,47	12,65

Influence Expr Logiques	1,57	1,79	1,88	1,89	1,77

Temps du Filtrage Signatures Seules / Temps de Filtrage de Texte (en pourcentage)

	199	307	397	499	599
Toutes requêtes	15,34	19,61	24,02	28,60	33,43
Sans Expr Log rép. OUI	68,8	85,1	104,3	121,8	145,7
Avec Expr Log rép. OUI	119	145	174,1	211,7	240,1
Sans Expr Log rép. NON	4,27	6,13	7,66	9,20	10,70

Influence Expr Logiques	1,73	1,70	1,67	1,74	1,65

département des études doctorales

AUTORISATION de SOUTENANCE

VU les dispositions de l'Arrêté du 23 novembre 1988 relatif aux Etudes doctorales

VU les rapports de présentation de Messieurs

- DELOBEL Claude, Professeur
- SCHOLL Michel, Directeur de Recherche

Madame JIMENEZ GUARIN Claudia Lucia

est autorisé(e) à présenter une thèse en soutenance en vue de l'obtention du diplôme de DOCTEUR de L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE, spécialité " Informatique "

Fait à Grenoble, le 16 juin 1989

Pour le Président de l'I.N.P.G.
et par délégation
Le Vice-Président

C. GAUBERT