# Les bases de données textuelles : étude du concept de document et application à deux réalisations

Irène Kowarski

# THESE

*présentée à*

## l' Institut National Polytechnique de Grenoble

*pour obtenir le grade de*
DOCTEUR DE 3ème CYCLE
«Informatique»

*par*

KOWARSKI Irène

# LES BASES DE DONNEES TEXTUELLES:

# ETUDE DU CONCEPT DE DOCUMENT ET APPLICATION

# A DEUX REALISATIONS.

Thèse soutenue le 5 juillet 1983 devant la commission d'examen.

| | |
|---|---|
| L. BOLLIET | Président |
| M. ADIBA | Examinateurs |
| C. DELOBEL | |
| J. COURTIN | Rapporteur |
| N. NAFFAH | Invité |

**INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE**

**Année universitaire 1982-1983**

**Président de l'Université : D. BLOCH**

**Vice-Président : René CARRE**
**Hervé CHERADAME**
**Marcel IVANES**

**PROFESSEURS DES UNIVERSITES :**

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

.../...

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

**PROFESSEURS ASSOCIES**

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

**PROFESSEURS UNIVERSITE DES SCIENCES SOCIALES (Grenoble II)**

BOLLIET Louis
Chatelin Françoise

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

RIEU Jean
SOUSTELLE Michel

**CHERCHEURS DU C.N.R.S.**

| | |
|---|---|
| FRUCHART Robert | Directeur de Recherche |
| VACHAUD Georges | Directeur de Recherche |

.../...

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

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

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

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

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

.../...

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

.../...

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

**LABORATOIRES EXTERIEURS**

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

En présentant ce travail, je tiens à remercier vivement

- Monsieur le Professeur L. Bolliet qui m'a fait l'honneur de présider le jury;

- Monsieur le Professeur M. Adiba, qui a bien voulu lire cette thèse et me faire part de ses remarques, puis participer au jury;

- Monsieur N. Naffah et Monsieur le Professeur C. Delobel qui m'ont fait l'honneur de faire partie du jury;

- Monsieur le Professeur F. Peccoud, dont le soutien à l'IFCI a permis la réalisation du projet MIDOC;

- Monsieur V. Joloboff, qui m'a accueillie dans son groupe de recherche à l'IMAG;

- Monsieur M. Lopez, qui, dans ce groupe, m'a mise au courant et guidée, toujours avec disponibilité et gentillesse;

- les étudiants stagiaires du Département Informatique de l'IUT de Grenoble, qui ont contribué efficacement à l'élaboration de MIDOC;

- Madame M. Malé, qui a participé à la réalisation matérielle de cette thèse;

et, enfin et surtout,

- Monsieur C. Michaux: son travail et sa bonne humeur ont fait de lui un irremplaçable collaborateur pour l'étude et la réalisation du projet MIDOC;

et

- Monsieur le Professeur J. Courtin: sans ses conseils judicieux, ses critiques constructives, et son soutien amical, je n'aurais certainement pas mené à bien ce travail. Qu'il sache ici combien je lui suis reconnaissante.

...aucun écrivain sérieux ne peut se passer d'un système de traitement de texte qui accouple la machine à écrire à un ordinateur, un écran de télévision et sans doute une cafetière.

John Harris, Le Monde, 6 mars 1983

TABLE DES MATIERES

## INTRODUCTION

Les systèmes informatiques qui permettent de gérer et de traiter des textes se répartissent à l'heure actuelle en trois grandes catégories :

I - Les systèmes de "traitement de textes", dérivés à l'origine des éditeurs de programmes (WORD80),(MENA79).
Ce sont des systèmes qui considèrent un texte comme une chaîne de caractères ("texte au kilomètre"). Certains de ces caractères ne font pas partie du "contenu" du texte, mais indiquent les commandes de mise en page, formattage, impression, etc, destinés à être prises en compte lors de l'impression du texte. Ces commandes sont implantées dans le texte à des positions quelconques, indépendantes du contenu du texte. Il peut de plus y avoir des options de mise en page externes au texte, valables pour l'ensemble du texte.

Certains systèmes permettent une structuration du texte en quelques niveaux prédéfinis et figés : par exemple "section" et "paragraphe".

Néanmoins les unités fondamentales restent le caractère et le mot, c'est ce qu'indique le nom anglais de ces systèmes :"word processing systems".

Les textes sont sauvegardés dans des fichiers, et la recherche d'un texte se limite à l'appel du fichier, dont on doit connaître le nom.

Ces systèmes présentent donc de nombreuses lacunes pour l'utilisation dans un environnement bureautique :

- Ils ne tiennent pas généralement pas compte de la structure interne des documents, structure liée à leur contenu sémantique (sauf certains systèmes, d'origine universitaire: voir (ALLE81), (REID80), (STRO81)).

- Aucune tentative de liaison des documents à leur environnement n'est réalisée, c'est à dire qu'il n'est pas possible de classer ni de retrouver un document sur d'autres critères que son nom.

Par ailleurs, ces systèmes ne permettent pas le partage de parties de texte entre plusieurs documents, ils se bornent à la possibilité de recopie totale ou partielle d'un document dans un autre.

Les systèmes de traitement de textes sont généralement destinés à des utilisateurs non informaticiens, et leur "interface usager" doit donc permettre un apprentissage et une utilisation faciles et agréables. Bien que des progrès notables aient été accomplis en ce domaine, ce n'est pas toujours le cas (FOBE82),(GOOD81).

II - les systèmes documentaires (BOUR81),(CXP75),(MIST75),(TEXT79)

Il s'agit de systèmes qui permettent d'obtenir des références à des documents choisis en fonction des mots clés sur lesquels les documents sont indexés, et éventuellement de certaines caractéristiques telles que la longueur, la langue, la date de publication.

Mais ces systèmes ne fournissent généralement qu'un résumé succint (abstract) du document, quelquefois le document lui-même s'il est court, laissant au lecteur le soin de trouver ailleurs le document complet.

Certains systèmes (banques de données spécialisées juridiques, économiques, par exemple) comportent la saisie de documents complets, et donc la possibilité de les consulter directement à travers le système documentaire. Mais ces documents complets sont traités comme des chaines de caractères de très grande longueur, aucune structuration n'est envisagée.

Les textes ainsi fournis (résumés ou documents complets) ne peuvent pas être modifiés : ces systèmes, destinés uniquement à la consultation, considèrent les documents comme des entités statiques. Les fonctions de saisie et d'archivage existent, mais pas les fonctions de création et de mise à jour.

III - Les systèmes de gestion de bases de données (SGBD).
(CODD70),(DATE77),(DELO82),(GARD83),(MART77)

Issus des systèmes de gestion de fichiers, les SGBD apportent en principe

- La non redondance des informations, qui ne sont pas reproduites dans plusieurs fichiers;

- Les relations (associations) entre toutes les informations;

- L'accès possible à toute les informations stockées dans la base, et pas seulement à des enregistrements à travers une clé;

- La possibilité de définir des vues sur l'ensemble des données, selon les utilisateurs.

Ces fonctions sont réalisées à travers différents langages (langage de définition de données, langage de manipulation de données ou langage de requêtes, éventuellement programmes en langage évolués, etc).

Les SGBD permettent l'introduction, la modification, la suppression de toutes les données qu'elles traitent, mais ces données apppartiennent généralement à quelques types élémentaires: numériques ou chaine de caractères. On dit souvent qu'il s'agit de données "factuelles".

Les données de types numériques peuvent être entières ou réelles, binaires ou décimales codées binaire, mais pour chaque représentation un ensemble de traitements est prédéfini. De même, les chaines de caractères peuvent être de longueur fixe ou variable, mais sont alors limitées à une longueur maximale. Un texte n'est considéré que comme une chaine de caractères, sans structuration interne.

On peut envisager d'ajouter aux données numériques et chaines, des données de types nouveaux : textuelles, graphiques, phoniques... C'est là l'objectif des bases de données généralisées actuellement au stade de la recherche : projets TIGRE(BOGO83) et BIG(BELT82),(CHRI83). Certains systèmes, qui font également l'objet de travaux de recherche, envisagent l'intégration des fonctionnalités de systèmes documentaires dans des SGBD, mais s'intéressent plutôt à la recherche de références aux documents, qu'au traitement des documents eux-mêmes (FLOR82),(MIRA83), (LEMA83).

Dans le travail présenté ici, nous nous limiterons aux traitement de données de type classique et des textes, en nous intéressant particulièrement au texte.

Notre but est donc :

- d'ajouter aux types classiques de données d'une SGBD, les données textuelles;

- de permettre l'exécution de toutes les fonctions d'un système classique de gestion de base de données;

- de créer, pour les données textuelles, des fonctions particulières: certaines possibilités d'un système de recherche documentaire, ainsi que les facilités de saisie, mise à jour et impression fournies par les systèmes de traitement de texte.

Dans le chapitre I, nous cherchons à préciser les objectifs généraux d'une base de données étendue aux textes (BDT), en fonction des différentes contraintes que l'on peut envisager. Au chapitre II, nous présentons le modèle choisi pour la BDT, et nous précisons en particulier tous les aspects du modèle de document dans une BDT. Les chapitres III et IV décrivent les deux implémentations de BDT auxquelles nous nous sommes intéressés (maquette sur Solar et projet MIDOC), et tentent d'évaluer pour chacun d'eux dans quelle mesure les objectifs des chapitres I et II ont pu être atteint. Pour rédiger les chapitres I et II, nous avons largement emprunté au rapport CETIB-IMAG (CETI80) et au Rapport de Recherche IMAG no 273 (JOL082).

# CHAPITRE I

## LES OBJECTIFS D'UNE BASE DE DONNEES TEXTUELLES

### I.1 - OBJECTIF GENERAL

L'objectif général d'une base de données de données étendue aux textes (BDT) est la communication entre individus d'une organisation; cette communication doit pouvoir se faire :

- sans intermédiaire : entre utilisateurs finaux;
- indépendamment de la localisation des communiquants;
- sous le contrôle de processeurs d'accès à l'information;
- en sécurité maximum sur la cohérence des informations échangées ou obtenues;
- à un coût minimal.

De l'objectif général précédent découlent un certain nombre de contraintes:

- contraintes liées au traitement de textes
- contraintes liées à l'environnement d'exploitation
- contraintes fonctionnelles liées aux besoins des utilisateurs.

La prise en compte de ces contraintes simultanément avec l'objectif général nous permettra au paragraphe I.6 la fixation des objectifs techniques du projet.

### I.2 - LES CONTRAINTES LIEES AU TRAITEMENT DE TEXTES

Les problèmes liés à l'archivage des informations textuelles et à leur traitement proviennent :

- des volumes
- des structures
- de la nature des textes.

## LES VOLUMES

Donnons quelques chiffres :

- 1 ligne contient environ 70 caractères.

- 1 page contient environ 60 lignes, soit 4000 caractères.

- 1 document technique peut contenir environ 250 pages, soit $10^6$ caractères.

En plus du texte lui-même, un grand nombre d'informations concernant l'identification, la mise en page, l'impression, la visualisation, etc... du texte restent à gérer .

La gestion d'une telle quantité d'informations pose les problèmes
- de la gestion d'espaces d'adressage suffisants ;

- de la gestion d'une hiérarchie de mémoire (mémoire centrale et différents types de mémoires périphériques);

- de l'existence de mécanismes permettant de limiter au strict nécessaire les transferts de texte d'un point à un autre du système.

## LES STRUCTURES.

Les structures d'informations numériques interprétables par un système donné (codages et bases de numération différents) restent en nombre limité. Sur ces structures sont définies directement les logiques de traitement et de manipulation ad hoc.

Il n'en va pas de même pour les informations textuelles, pour lesquelles on peut dire qu'il existe pratiquement autant de structures différentes que d'applications possibles. Par exemple, un même document ne sera pas vu de la même manière par son créateur, son éditeur, le typographe, le documentaliste ou celui qui le consulte sur telle ou telle information.

Ici réside la difficulté principale du traitement de textes: l'absence de structure(s) priviligiée(s) entraîne le recours habituel à des

logiques définies sur la structure la plus élémentaire : le caractère.

Aux logiques de traitement de textes on a substitué, faute de mieux, une logique de traitement de chaînes de caractères. Notons tout de même que certains systèmes récents prennent en compte les structures: PEN (ALLE81) ou SCRIBE (REID80) par exemple.

LA NATURE DES INFORMATIONS TEXTUELLES.

La nature d'un texte c'est d'être communicable sous forme graphique.

Pour être communicable, il doit comporter un certain nombre de renseignements extérieurs à son texte: le nom de l'auteur (ou du service qui l'a rédigé), les personnes à qui il est destiné, l'utilisation qui devra en être faite, son degré de confidentialité, son prix... Ces renseignements permettront de le situer dans un contexte, de le classer, de l'utiliser judicieusement, etc. Nous les nommerons "caractéristiques externes".

Pour pouvoir être communiqué sous forme graphique, il faut adjoindre au texte brut des renseignements concernant sa présentation: format du papier, tailles des marges, polices de caractères, interlignes... Ces renseignements pourront d'ailleurs varier, pour un même texte, d'une part selon les destinataires envisagés, d'autre part tout au long du document, selon les différentes parties. Ils seront donc liés tant aux renseignements externes qu'à la structure.

Il en résulte qu'un grand nombre d'informations destinées à gérer le texte doit accompagner celui-ci. Ces informations secondaires aux textes doivent elles-mêmes être gérées par le système.

## I.3. - LES CONTRAINTES LIEES A L'ENVIRONNEMENT.

L'environnement de travail d'un système de communication complet pourrait, idéalement, être :

- multi-utilisateur

- multi-base (bases réparties, privées et communes).
- multi-étage, c'est-à-dire reflétant l'organisation hiérarchique plus ou moins complexe de l'accès aux informations : sélection, contrôle, et optimisation des accès.
- multi-site avec en corollaire l'éloignement plus ou moins grand des postes de travail et des systèmes serveurs: réseaux locaux, privés, publics.
- multi-exploitation : traitements conversationnels, ou par lot.
- orienté utilisateur final ( ergonomie, langages, fonctions).

De plus le système doit être exploitable à un coût raisonnable, c'est-à-dire en rapport avec ses services.

## I.4. - LES CONTRAINTES FONCTIONNELLES.

Le système étant orienté utilisateur final, il s'agit de restituer à l'utilisateur son contexte normal de travail qui est en général :

- multi-fonction : simultanéité des fonctions.
- multi-structure d'informations
- varié : diversité des tâches liées à l'organisation et l'environnement du travail.

## I.5. - EXEMPLE.

Pour illustrer la nécessité des structures et des fonctions multiples d'information au niveau de l'utilisateur final nous donnons fig I.1 un exemple d'utilisation de base de données généralisée (c'est-à-dire qui traite à la fois les données factuelles et les textes).

Une société possède plusieurs AGENCES qui ont elles-mêmes des CLIENTS et des produits à vendre dans des CATALOGUES et décrits dans une DOCUMENTATION TECHNIQUE. Chaque Agence enregistre des COMMANDES de CLIENTS sur les CATALOGUES et établit des CONTRATS de vente à partir d'un CLAUSIER (document qui contient toutes les clauses susceptibles de figurer dans les contrats) et de la DOCUMENTATION TECHNIQUE/COMMERCIALE associée à chaque produit.Elle établit ensuite une facture envoyée au CLIENT et enregistrée en COMPTABILITE CLIENT et COMPTABILITE GENERALE.

EXEMPLE D'UTILISATION D'UNE B.D.G.

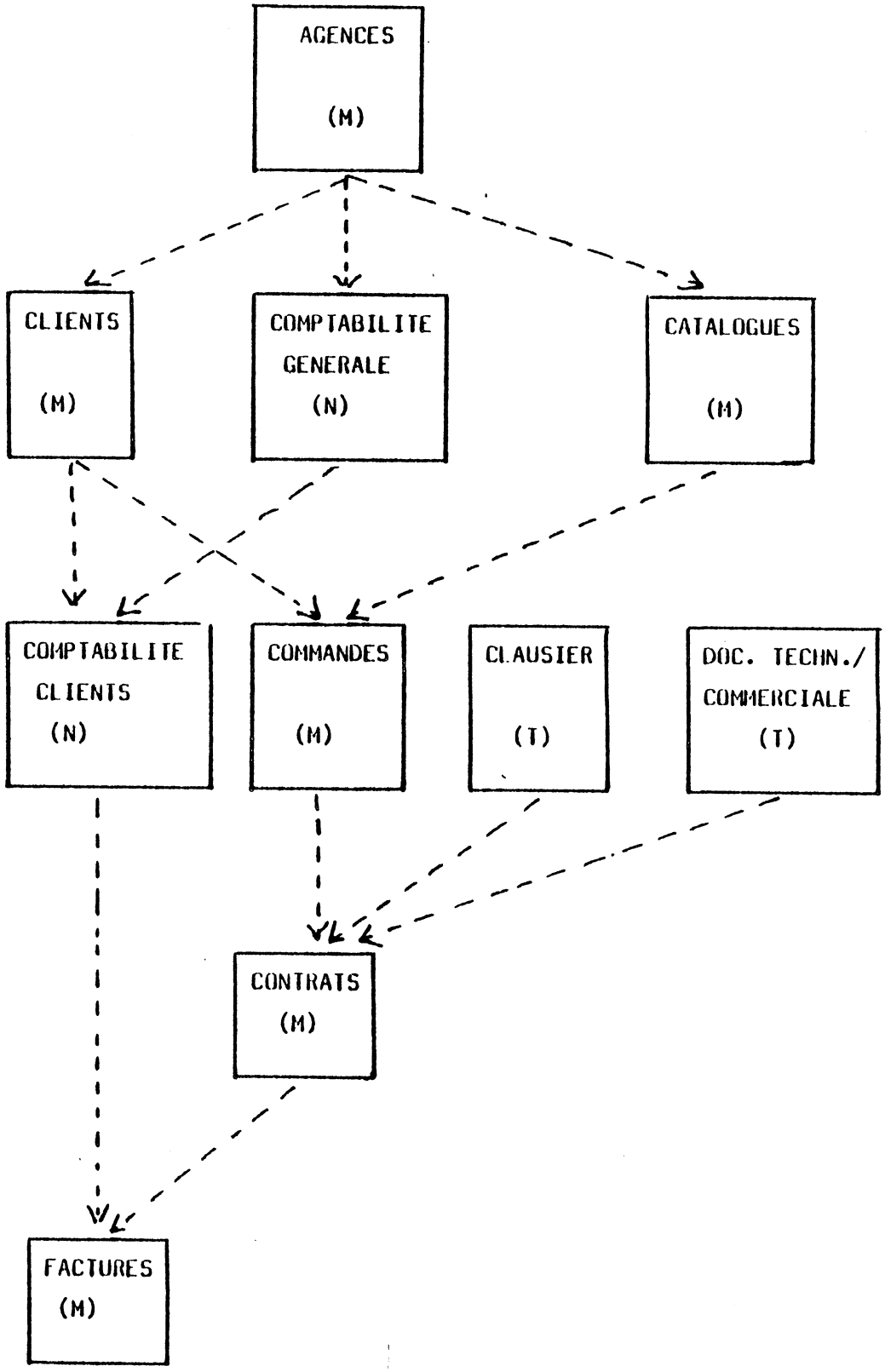



figure I.1

On constate dans la figure I.1 que les " entités " représentées sont :

- numériques (N) en minorité

- textuelles (T) en majorité relativement au volume de stockage.

- mixtes (M) en majorité relativement à la fréquence d'apparition .

Les fonctions définies sont aussi diverses que :

- des fonctions commerciales en mode interactif ( prises de commandes)

- des fonctions de traitement de textes (établissement du contrat)

- des fonctions comptables diverses

- la facturation ( édition en traitement par lots)

- des fonctions de contrôle diverses pour la validation des informations saisies à différents niveaux.

## I.6 - LES OBJECTIFS TECHNIQUES

De l'objectif général, la communication, et des contraintes supportées, nous pouvons tirer un certain nombre d'objectifs techniques pour une BDT:

a)
L'introduction de nouveaux textes dans la base de données doit être réduite au minimum indispensable. En effet un texte géré par une B.D.G est une information coûteuse à saisir, coûteuse à contrôler, coûteuse à traiter. On doit éviter autant que possible les redondances inutiles.

Il faut donc offrir à l'utilisateur le moyen d'utiliser l'information déjà enregistrée dans un environnement fonctionnel acceptable.

b)

On s'attachera à distinguer chaque fois que possible l'archivage du texte, suite de caractères indifférenciés, des traitements proprement dits qui portent en général sur des propriétés du texte plutôt que sur le texte lui-même.

c)

Le contrôle d'accès à l'information par les utilisateurs ne doit pas constituer une entrave sérieuse à l'atteinte de l'objectif (a): partage des textes entre les documents. En d'autres termes, les contraintes de confidentialité doivent porter sur le texte proprement dit, plutôt que sur ses propriétés, ou attributs.

d)

Un même texte doit pouvoir être vu par chaque utilisateur ou groupe d'utilisateurs comme un ensemble distinct d'informations.

En d'autre termes, si un texte est bien composé de la même séquence de caractères pour tout utilisateur, son contenu informationnel est variable suivant la "structure" que lui applique le lecteur.

e)

Le système doit être en apprentissage constant et capable d'aller rechercher les informations qui lui manquent dans un délai compatible avec l'environnement d'exploitation et à un coût acceptable. Ceci doit être vrai aussi bien pour les informations liées à la séléction des textes (indexation de texte) qu'à leur structuration (traitement et manipulation de texte). Ces informations ne sont pas prédéfinies, et stockées "a priori".

f)

"L'univers" d'exploitation des textes doit être aussi complexe et varié que souhaitable, ceci implique que cet univers soit descriptible par l'utilisateur suivant des schémas qui lui sont propres.

Cet univers implique généralement l'utilisation conjointe ou simultanée de différentes fonctions, ce qui parait exclure l'application centrée autour d'une fonction unique. Ainsi, une application qui se bornerait à la saisie de gros volumes de texte, ou bien à la recherche documentaire, aurait intérêt à utiliser un système spécialisé plutôt qu'une BDT.

g)

Les fonctions de manipulation logique des textes (création, modification, consultation) et celles de présentation physique soit à l'entrée soit à la sortie du système, doivent être séparées et accessibles par des interfaces spécifiques.

La présentation d'un texte liée à un style ou à un organe phériphérique, ne doit pas être une cause de rejet par le système.

h)

Tout texte est paramétrable, c'est-à-dire qu'il est possible d'identifier des zones de texte pour lesquelles il est possible d'insérer ou de substituer des valeurs quelconques.

i)

Tout texte est accessible tant par son contenu (défini par des mots-clés), que par ses caractéristiques externes.

j)

La gestion de "l'espace de texte" doit rester indépendante de celle des autres "espaces documentaires". Cet objectif vient logiquement compléter l'objectif (b) du coté de l'exploitation du système.

# CHAPITRE II

# LE MODELE CONCEPTUEL D'UNE BASE DE DONNEES TEXTUELLES

## II.1 - BASE DE DONNEES CLASSIQUE

La figure II.1 rappelle le modèle général d'une base de données classique. On y trouve :

- Le schéma conceptuel par lequel le créateur de la base de données "modélise" l'univers des informations auquel il s'intéresse. Un langage approprié permet la description des objets de cet univers et des relations qui les lient. Cette description reste indépendante de la configuration physique du système hôte.

- Les différents schémas externes qui offrent des visions logiques particulières du schéma conceptuel et sont par conséquent orientés vers les applications.

- Le schéma interne qui permet la description de l'organisation physique des informations du schéma conceptuel dans le cadre d'un système hôte déterminé.

- Les données, ou réalisations, c'est-à-dire les valeurs réelles qui sont introduites dans la base et dont la description est faite dans le schéma conceptuel.

Les services offerts par de tels systèmes de base de données sont d'une manière générale :

- Un langage de description des données au niveau logique (schéma conceptuel).

- Un langage de description des données au niveau physique (schéma interne).

- Un langage de requête pour l'accès à la base avec des services plus ou moins évolués.

- Un ou plusieurs interfaces avec des langages de programmation classiques.

- Un ensemble d'utilitaires ou processeurs spécialisés facilitant la gestion de la base de données.

- Un système hôte fournissant l'infrastructure générale et les fonctions d'accès (gestion des tâches, des terminaux, des utilisateurs etc...)

BASE DE DONNEES CLASSIQUE

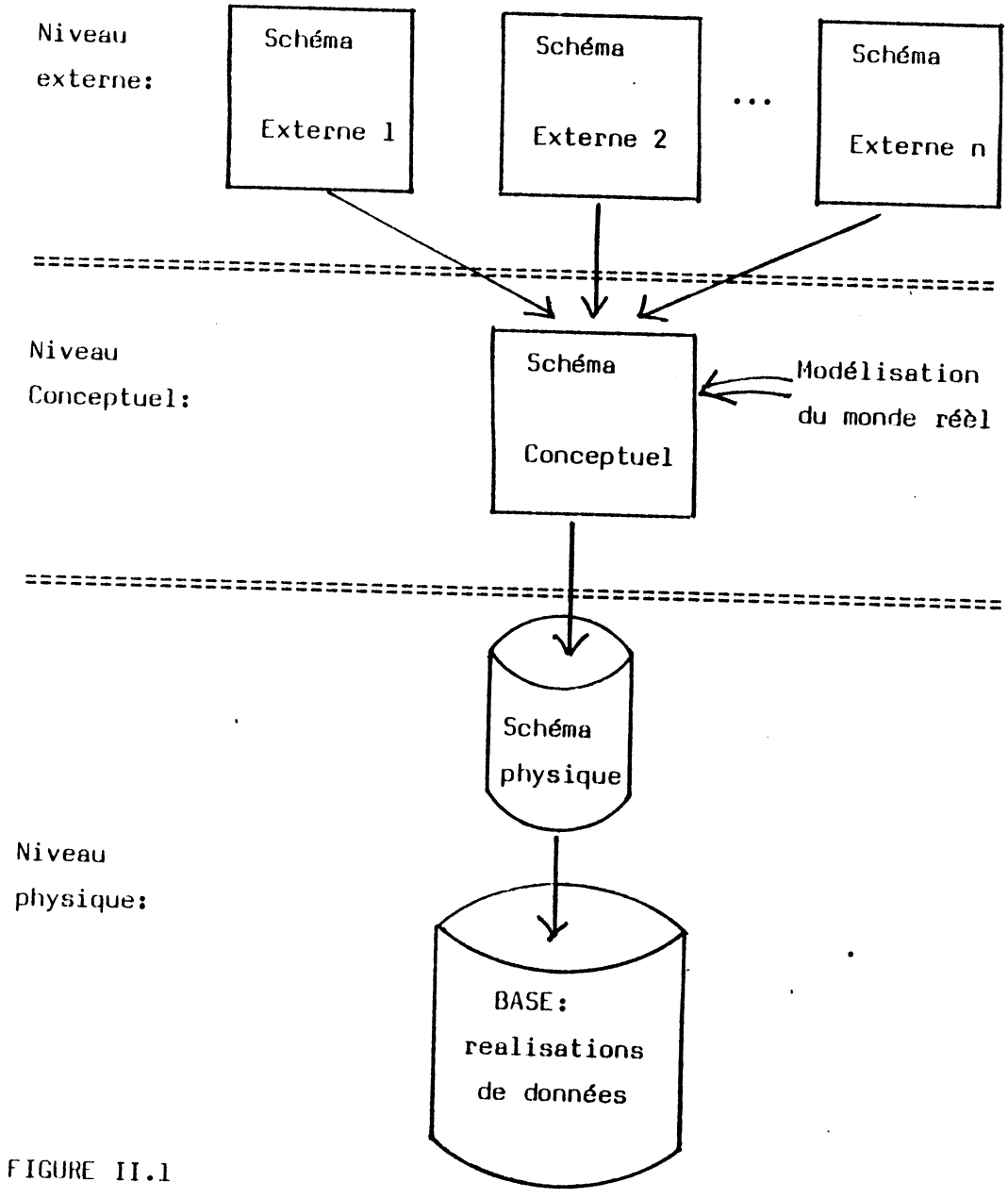



FIGURE II.1

## II.2 - BASE DE DONNEES TEXTUELLES: LE MODELE ENTITE-DOCUMENT

La figure II.2 donne un modèle général d'une base de données textuelles. Elle se dérive de la précédente par l'adaptation au niveau du schéma conceptuel d'un nouveau schéma (ou extension du schéma conceptuel) permettant la description des structures textuelles ou documentaires auxquelles l'utilisateur s'intéresse: c'est le schéma documentaire.

On aurait pu envisager simplement de créer un nouveau type de données, le texte, et de le décrire comme les autres données dans un schéma conceptuel unique. Il nous a semblé que les fonctionnalités propres aux documents impliquent un traitement particulier de ces données, et par conséquent leur description dans un schéma distinct.

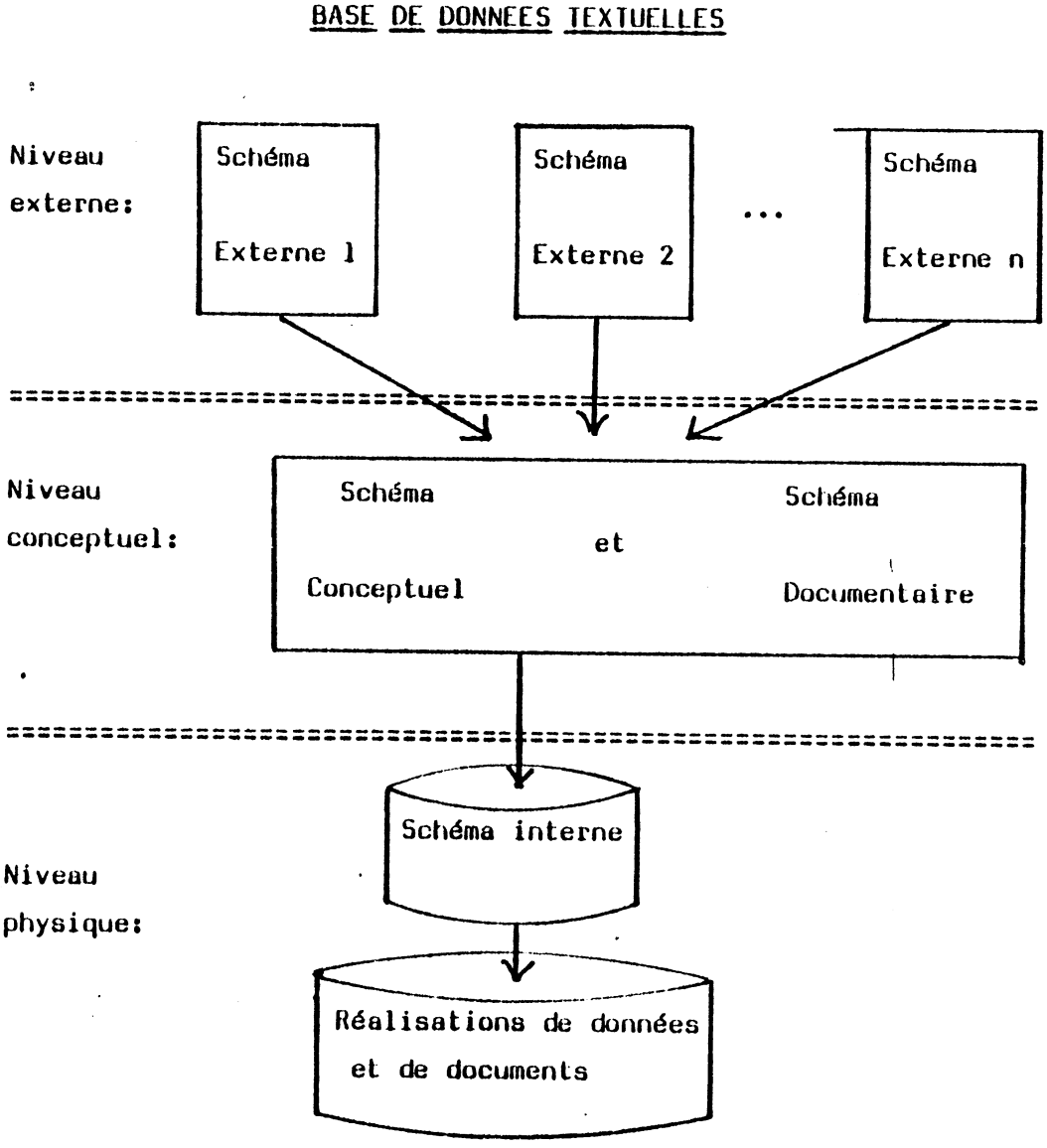



FIGURE II.2

Le schéma documentaire comporte donc l'ensemble des services, langages et utilitaires permettant de décrire et de gérer de manière appropriée les structures textuelles dont les chapitres suivants feront apparaître les propriétés tout à fait particulières par rapport aux informations numériques ou pseudo-numériques classiques.

Ces informations classiques décrivent des objets qui sont extérieurs à l'environnement de l'ordinateur: personnes, automobiles, commandes, projets... La base de données utilise des nombres et des chaînes de caractères pour construire des descriptions de ces objets. Nous dirons que ce sont des "objets généraux" (figure II.3). Mais d'autres objets appartiennent intrinsèquement à l'environnement de l'ordinateur: un fichier ou un programme par exemple. Ces objets peuvent donc faire partie de leur propre description. Il en est de même des représentations electroniques d'objets complexes tels que le texte, l'image, la voix, etc (figure II.4).

Nous pouvons ainsi distinguer les descriptions qui contiennent l'objet décrit (objets-ordinateur) de celles qui ne le contiennent pas (descriptions d'objets généraux). Dans une base de données textuelles les seuls objets electroniques que nous étudierons seront ceux qui décrivent des textes, et que nous appelerons DOCUMENTS. Tous les autres objets seront des objets généraux et nous dirons que leurs descriptions sont des ENTITES.

OBJETS GENERAUX

(L'objet décrit ne figure pas dans sa description)

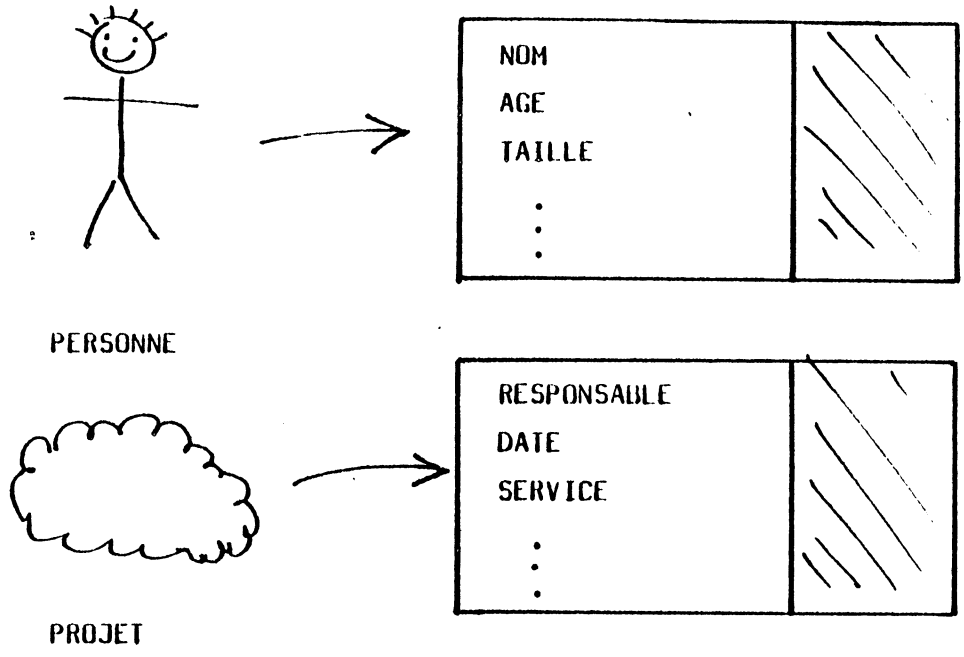



figure II.3

OBJETS ELECTRONIQUES

(L'objet décrit fait partie de sa description

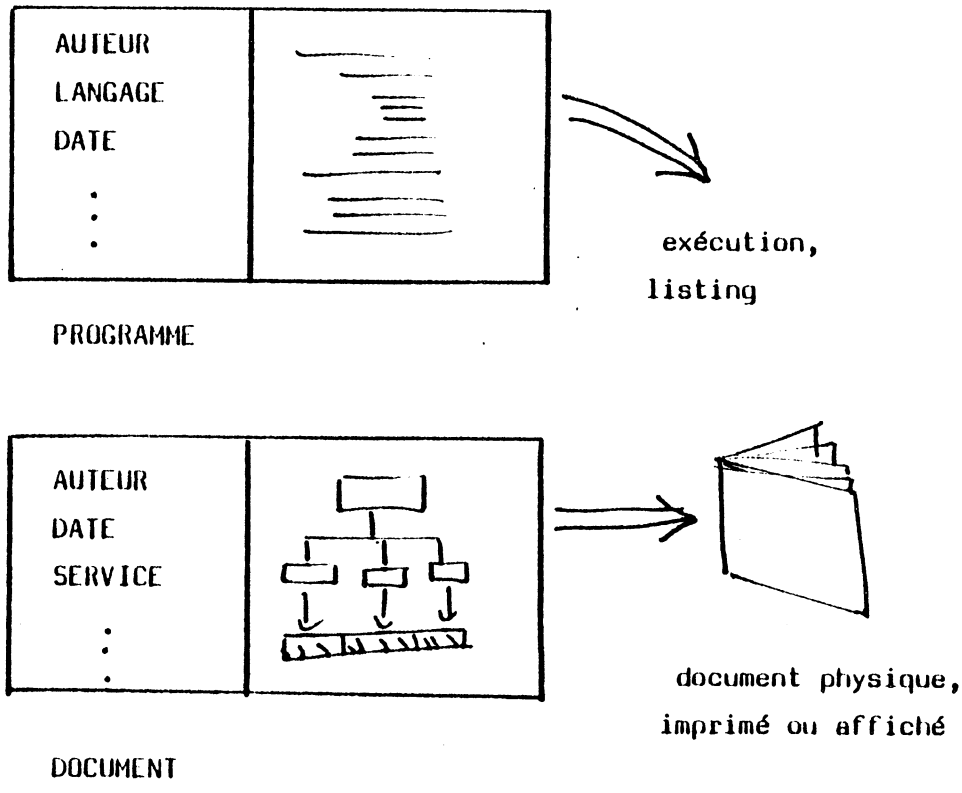



figure II.4

L'objet electronique peut être vu de l'extérieur, au moyen d'une "matérialisation": exécution ou listing pour un programme, document physique imprimé ou affiché pour un document. Le document sur papier ainsi obtenu peut être classé, communiqué, etc..., mais il n'appartient plus à l'environnement de la BDT.

Dans toute la suite de ce travail, lorsque nous parlerons de "documents" il s'agira bien de "documents electroniques" que nous appellerons souvent "documents virtuels", par opposition à leur matérialisation, qui sera nommée "document physique" ou "document imprimé".

En résumé, le schéma conceptuel classique se rapportera aux entités, tandis que le schéma documentaire permettra de décrire plus précisément les documents. Du point de vue d'un utilisateur, cette distinction doit disparaître, car la manipulation de toutes les données, aussi bien factuelles que textuelles, doit se faire à travers un schéma conceptuel unique.

Les classes d'objets que traite une base de données classique sont donc décrites dans le schéma conceptuel, qui sont des modèles d'objets du monde réel, et qui se traduiront à leur tour par des réalisations, ou occurrences, qui sont les données finalement manipulées par les programmes.

Par exemple, la classe des "personnes" pourrait être décrite en utilisant un formalisme dérivé de celui de SOCRATE (SOCR79) par :

```
entite    PERSONNE
début     NOM : alpha ;
          AGE : entier ;
          TAILLE : reel ;
          ADRESSE : alpha ;
fin
```

On dira que "NOM", "AGE", "TAILLE" et "ADRESSE" sont des caractéristiques formelles de la classe des "PERSONNES".

Chacune de ces caractéristiques est elle-même définie comme appartenant à un domaine déterminé, connu du SGBD (ici "entier" "reel" et "alpha").

Il existe aussi des caractéristiques "références", dont la valeur est une entité (de la même classe ou d'une autre classe). Par exemple, on pourrait définir une classe de "voitures" :

```
entite VOITURE
début
     MARQUE : alpha;
     IMMATRICULATION : alpha;
     PROPRIETAIRE : référe une PERSONNE;
fin
```

Une réalisation de l'entité PERSONNE consiste à affecter des valeurs effectives aux caractéristiques formelles . Par exemple :

"Dupont Pierre",
"25",
"1.76",
"10 rue Pasteur - Grenoble".

De la même façon nous décrirons des classes de documents par une liste de caractéristiques formelles. Mais de plus, à toute réalisation d'un document sera implicitement associée un texte noyau de la représentation, structuré en fonction d'un type, on pourrait décrire les documents de la classe des manuels techniques" par le formalisme suivant:

```
document   MANUEL-TECHNIQUE
début
     TITRE : alpha ;
     AUTEUR : alpha ;
     PARUTION : date ;
     MISE-A-JOUR : date ;
     NUMERO-REFERENCE : entier ;
     TYPE : MANUEL ;
fin
```

en utilisant les domaines "alpha", "date" et "entier", et le type de document "MANUEL".

Toutes les réalisations de documents de la classe "MANUEL-TECHNIQUE" seraient donc identifiées par des valeurs effectives des caractéristiques, par exemple

"PASCAL VII.0",
"Softech",
"1.3.82",
"2.5.83",
"12345" (cf figure II.5)

et seraient structurées selon le type "MANUEL" (cf ci-dessous).

On pourra faire des références d'entités aux documents. Ainsi, la classe des personnes pourrait être complétée :

```
entite PERSONNE
début
     NOM : alpha;
     AGE : entier;
     TAILLE : reel;
     ADRESSE : alpha;
     PUBLICATION : réfère un LIVRE;
fin

document LIVRE
début
     AUTEUR : alpha;
     PRIX : entier;
     PUBLICATION : date;
     TYPE : BOUQUIN;
fin
```

Mais la sémantique de la déclaration de document est beaucoup plus riche que celle de la déclaration d'entité. En effet, elle postule implicitement que tous les documents d'une même classe sont structurés d'une manière semblable. Cette structure commune doit donc être définie en même temps que la classe des documents concernés au moyen de la caractéristique obligatoire "type", qui renvoie à une déclaration du type (et donc d'une structure) "MANUEL".

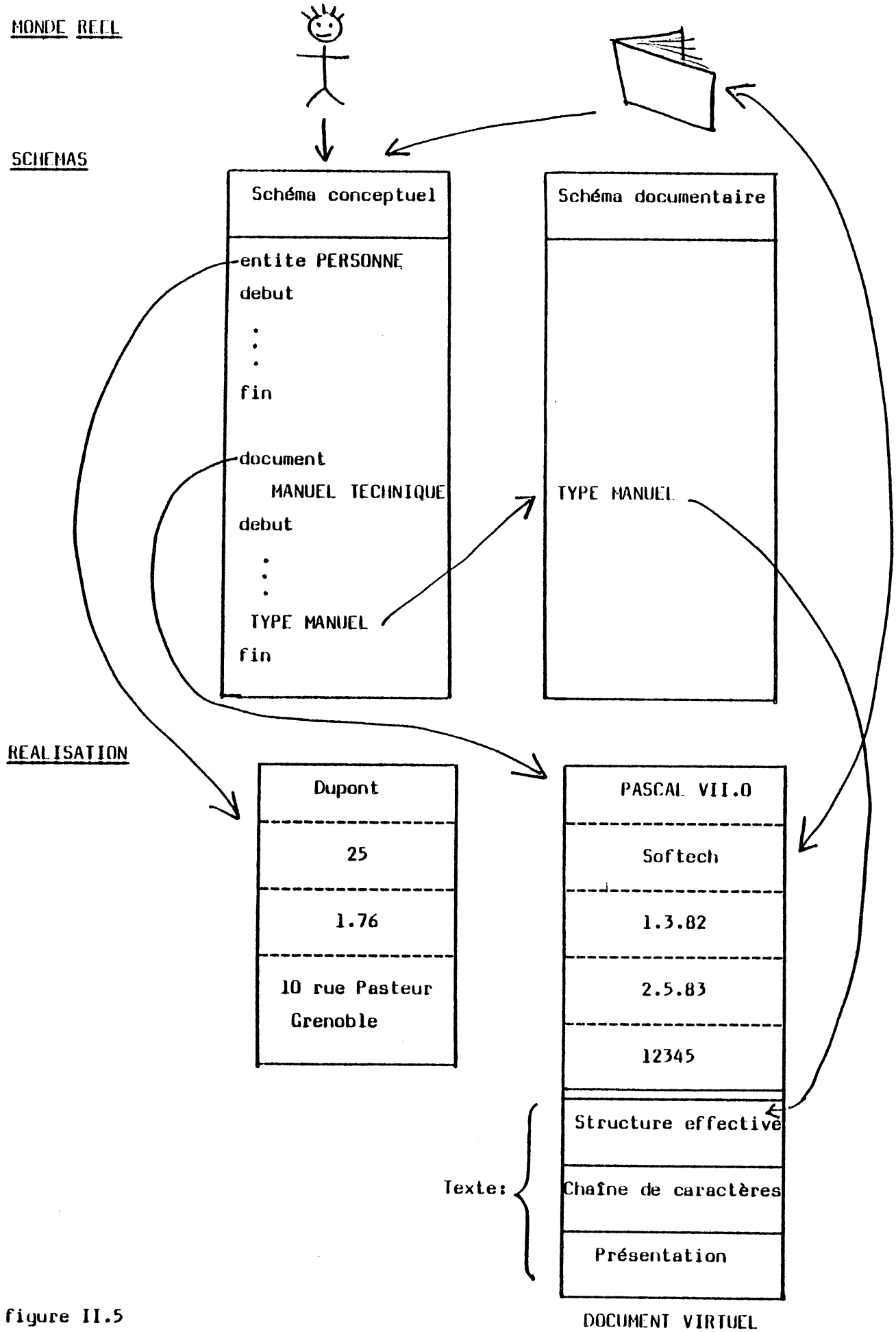

MONDE REEL
SCHEMAS
Schéma conceptuel
entite PERSONNE
debut
fin
document
MANUEL TECHNIQUE
debut
TYPE MANUEL
fin
Schéma documentaire
TYPE MANUEL
REALISATION
Dupont
25
1.76
10 rue Pasteur
Grenoble
PASCAL VII.0
Softech
1.3.82
2.5.83
12345
Structure effective
Texte:
Chaîne de caractères
Présentation
DOCUMENT VIRTUEL

figure II.5

Finalement, une déclaration de document correspond donc à la déclaration

- d'une part d'une liste de caractéristiques qui sont des propriétés extrinsèques, non liées au texte et à sa représentation, et pouvant être saisies et manipulées indépendamment du texte lui-même, au même titre que les caractéristiques des entités .

- d'autre part de propriétés intrinsèques, telles que sa structure logique et sa présentation visuelle.

Nous expliciterons ci-dessous les notions de structure et de type, ainsi que les possibilités de traitement de la présentation du document physique associé au document virtuel.

## II.3 - LE SCHEMA DOCUMENTAIRE

### II.3.1 - NOTIONS GENERALES

Nous présentons dans cette partie la description détaillée du schéma documentaire qui permet de modéliser les documents sous son aspect logique. Nous n'aborderons pas ici les représentations associées, qui seront étudiées dans les chapitres III et IV.

On peut établir un certain parallèle entre le schéma conceptuel et le schéma documentaire (fig II-6)

- Le schéma conceptuel définit des modèles d'entités et de documents au moyen de caractéristiques, le schéma documentaire définit des types de documents au moyen d'éléments hiérarchiques (parties de texte, organisées en arborescence)

- les valeurs des caractéristiques sont prises dans des domaines de valeurs, connus du système, alors que l'on attache aux éléments hiérarchiques des valeurs d'attributs également connus du système (justification, marges, etc...). Mais la valeur principale d'un élément hiérarchique est la chaîne de caractères qui lui est associée.

Modélisation

| Schéma conceptuel | Schéma documentaire |
|---|---|
| Entité = ensemble de caractéristiques | Type = ensemble d'éléments hiérarchiques |
| Valeur caractéristique ∈ domaine | Valeur d'un élément hiérarchique = chaîne de caractères et attributs |

Réalisation

Personne : nom, age...

Livre { Introduction, Développement, Conclusion }

age ↓ 25 entier

Introduction
marges : 5 - 60
numérotation : centrée
↓ chaine de caractères

figure II-6

Un type définit les propriétés logiques d'une classe de documents. Un document sera donc défini au niveau du schéma conceptuel par

- un nom de classe

- un ensemble de caractéristiques externes

- le nom du type auquel correspond le texte du document.

(exemple : cf le manuel technique défini ci-dessus chap II.2)

N.B. Le type étant défini indépendamment de la classe, plusieurs classes de documents peuvent se référer au même type, seules les caractéristiques externes seront différentes.

De même, un même texte pourra éventuellement servir à plusieurs documents, puisque la chaîne de caractères est indépendante du document. Il nous semble tout de même souhaitable que les documents utilisent dans ce cas le même type, sinon les difficultés de structuration paraissent très grandes.

Les documents peuvent être indexés, et par la suite retrouvés, à l'aide de mots-clés. Le schéma documentaire contient donc les descriptions d'un ou de plusieurs lexiques de mots-clés (cf II.3.4)

Les entrées et sorties de documents sont réalisées par des processeurs spécialisés. La présentation d'un document physique pourra être spécifiée dans un "canevas" associé au "type" du document de la base (cf II.3.3)

Enfin les documents peuvent contenir des "parties variables" différentes pour chaque document physique issu d'un même document virtuel. Ces parties seront indiquées dans le document comme des paramètres formels (cf II.3.5)

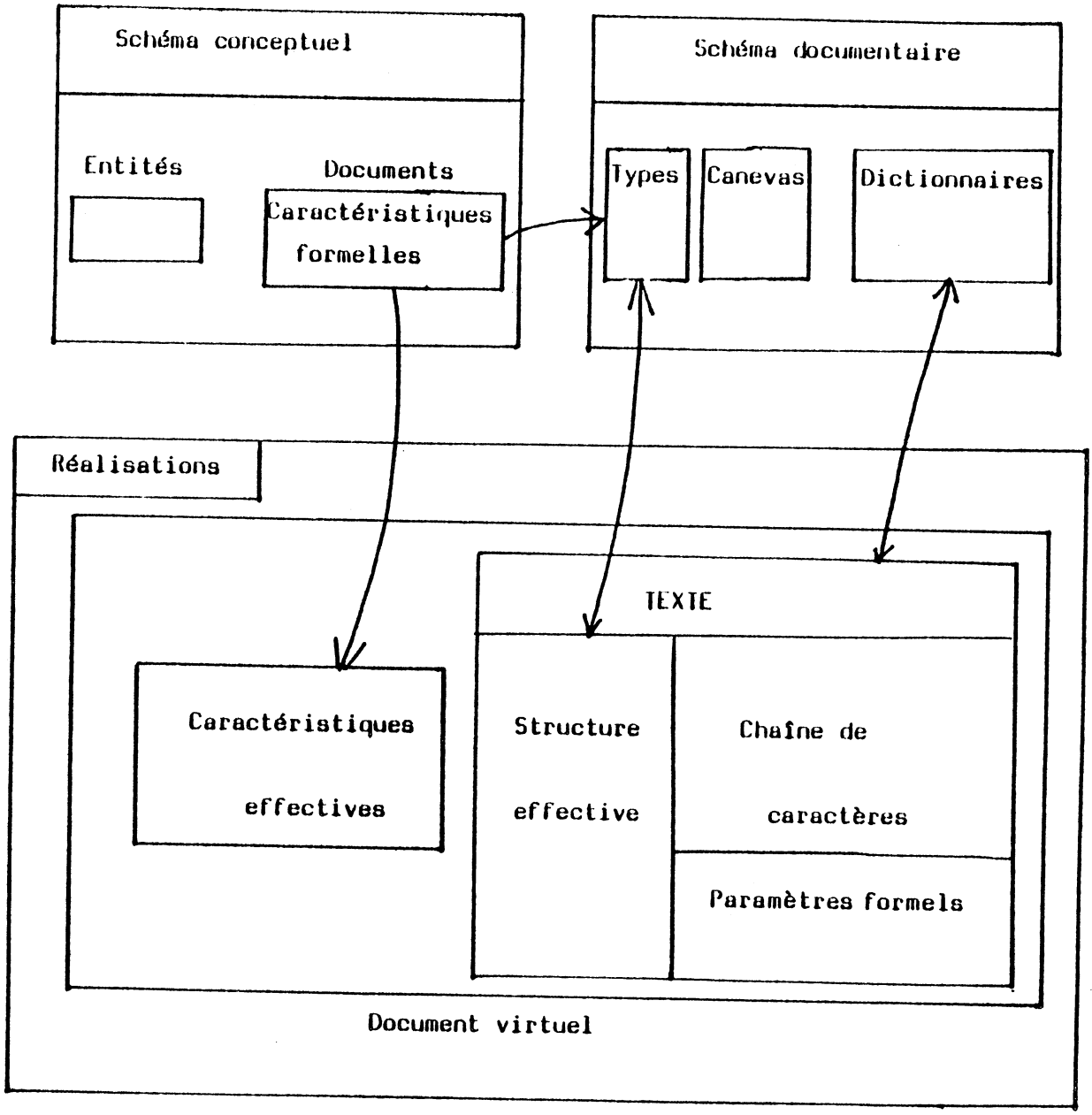



figure II.7

Les questions de confidentialité des documents ne sont pas étudiées dans cette partie, car on peut les traiter globalement au niveau du schéma conceptuel de la base. Il aurait été intéressant d'envisager un schéma de droits d'accès liés au type des documents , et même à leurs éléments hiérarchiques, mais on peut se demander si un tel schéma serait applicable en pratique (BUSS81).

II.3.2 LES STRUCTURES

Tous les textes de la base sont structurés selon une arborescence. On associe donc à la chaîne de caractères qui compose le texte, une structure arborescente qui définit une hiérarchie. Celle-ci se projette sur la chaîne, et à chaque feuille de la structure correspond une chaîne élémentaire, dite "segment de texte". Tout noeud de la structure est identifié par un nom, celui-ci désignera en même temps la suite des segments de texte qui correspondent aux feuilles du sous-arbre issu de ce noeud. L'accès à une partie de texte pourra donc se faire au moyen du nom du noeud concerné, puis du parcours des feuilles du sous-arbre concerné. Les noms et les liens hiérarchiques sont définis dans le type (fig II-8).

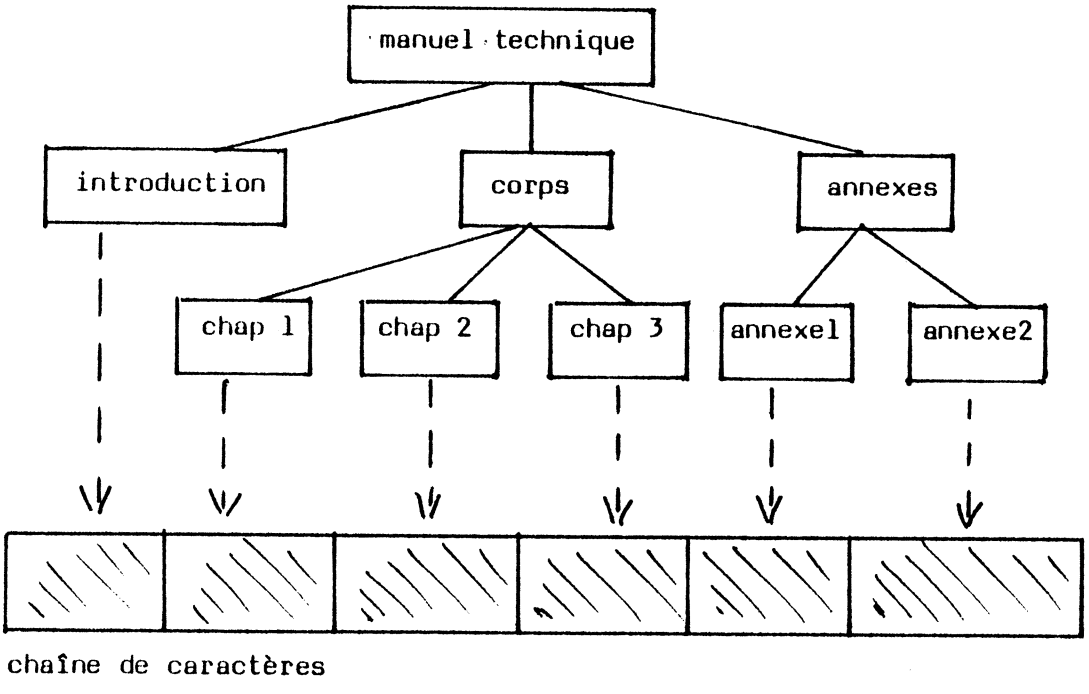



figure II-8

Afin d'éviter d'avoir des structures trop rigides, le type permettra de représenter une famille de structures, variantes autour d'une forme générale. Ainsi, certaines parties pourront être facultatives, d'autres pourront être répétitives en nombre quelconque. (Ce nombre peut être nul, les parties répétitives sont donc ipso facto facultatives).

Une structure générale serait donc de la forme :

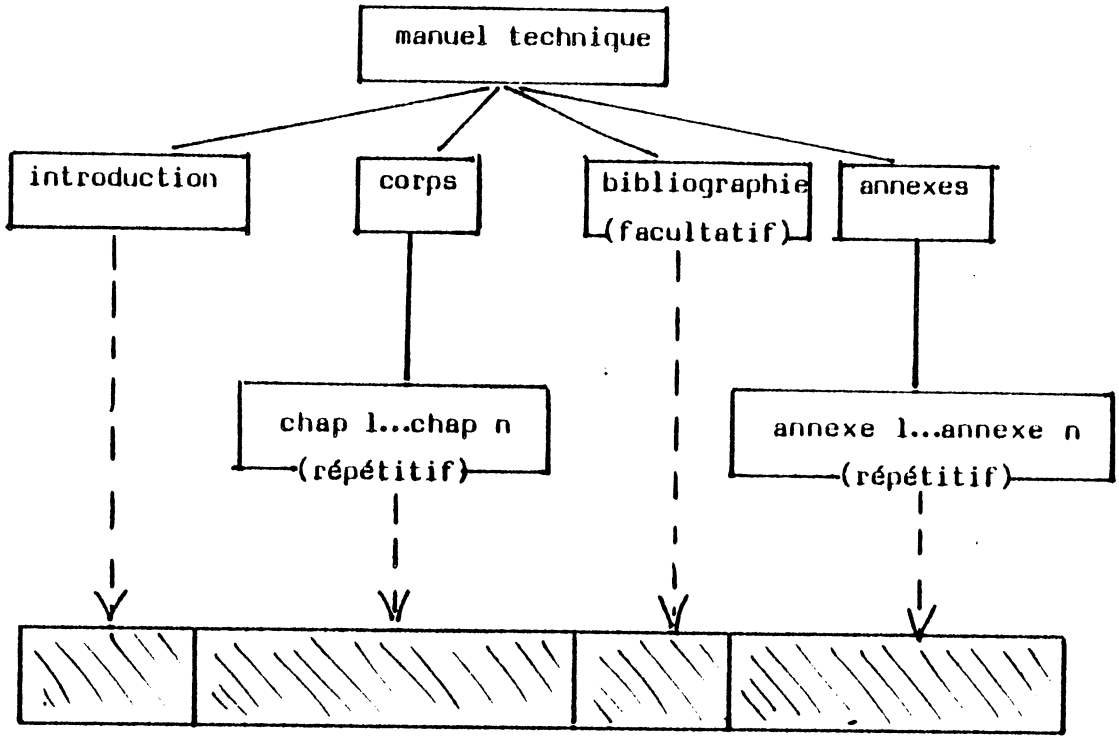



figure II-9

La structure effective minimale qui correspond à cette structure générale est la suivante :

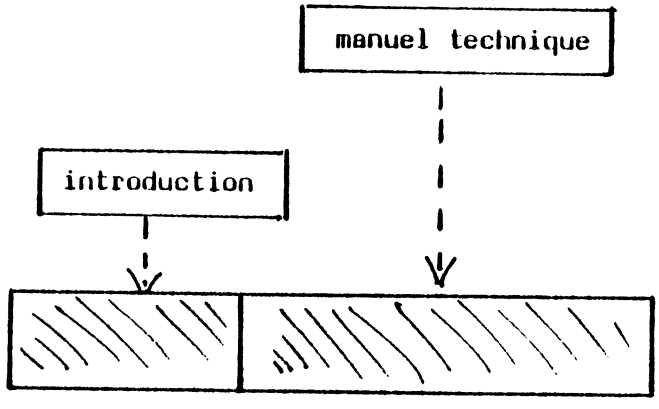



figure II-10

Le manuel correspondant semble se limiter à une présentation générale d'un sujet, sans chercher à détailler diverses parties...

Toute la structuration générale du texte doit être définie avant la création de la chaîne de caractères ; il s'agit donc bien d'une réflexion sur la structure sémantique d'un document, préalablement à sa construction effective.

En cela, notre point de vue diffère de celui d'autres systèmes. Par exemple, certains systèmes de traitement de textes découpent une chaine de caractères en paragraphes, sous-sections et sections au moyen de caractères de contrôle insérés dans le texte. Ce mode de structuration, figé, ne permet pas d'établir des classes d'equivalence sur les documents, ni d'obtenir des arborescences de hauteur supérieure à une valeur figée.

Certains systèmes, par exemple (BEN078), analysent et structurent le texte en fonction de sa présentation physique et permettent en outre d'y inclure des tableaux et des "expressions libres". Mais le but de ce type de structuration est uniquement d'aboutir à une présentation typographique soignée du texte. Ce sont des systèmes de "traitement de textes" évolués, mais non des bases de données textuelles.

### II.3.3 - ATTRIBUTS

Nous nommerons attributs les aspects d'un document qui sont liés à la présentation que l'on souhaite donner au document physique : numérotation, marges, polices, etc...

Les attributs peuvent être partagés en deux classes :

- Les attributs globaux marquent une propriété soit de l'ensemble du document, soit d'un sous-arbre. Ils sont donc liés à la hiérarchie de la structure logique.

Notons bien cependant que nous considérons que ces attributs découlent de la structure, contrairement aux systèmes qui "plaquent" une structure sur un texte en fonction d'une présentation.

Ainsi nous pouvons décider de numéroter les pages par chapitre, et de passer à une nouvelle page à chaque début de chapitre. Mais la structuration du texte en chapitres existera indépendamment de ces

attributs: le passage au chapitre suivant ne sera pas décidé en fonction d'une reprise de la numérotation, ou bien d'un changement de page.

D'autres attributs globaux pourraient être :

- le rappel du titre d'un chapitre en haut de chaque page de ce chapitre

- les marges gauche et droite

- l'indexation ou marge paragraphe : décalage de la première ligne de chaque paragraphe par rapport à la marge gauche

- l'interligne etc...

Si un attribut global est affecté à un noeud de la structure, il est bien entendu valable pour tous les noeuds du sous-arbre associé à moins qu'une nouvelle valeur ne soit affectée à cet attribut sur un des noeuds descendants.

On pourra considérer que tous les attributs globaux ont une valeur pour chaque noeud de la structure :

- soit de la valeur affectée au noeud lui-même

- à défaut, la valeur affectée au noeud "père"

- et à défaut encore, si aucun noeud "ascendant" n'a reçu une valeur explicite, une valeur affectée par les système (valeur dite habituellement "par défaut").

Les valeurs des attributs globaux ne peuvent donc pas être définies indépendamment de la connaissance de la structure logique. On peut envisager soit de les associer directement aux noeuds de la structure effective de la réalisation d'un document, soit de les consigner dans un "canevas" associé au type du document.

Dans le premier cas, chaque réalisation d'un document possède des attributs de présentation globaux qui lui sont propres, dans le deuxième cas tous les documents d'un même type auront les mêmes attributs globaux de présentation. Les deux possibilités peuvent d'ailleurs coexister :

dans ce cas les options définies dans le canevas seront les options "par défaut", et pourront être remplacées par celles qui sont définies aux noeuds d'une structure effective.

- Les attributs locaux sont ceux qui marquent une propriété d'une sous-chaîne de caractères indépendamment de la structure logique : soulignement, changement de police de caractères, tabulation, etc... Ces attributs peuvent être insérés directement dans le texte, comme le font tous les systèmes de traitement de textes. Mais ils peuvent aussi être "extraits" de la chaine d'entrée, mémorisés à part, et réinsérés dans la chaîne de sortie. Le texte interne sera alors dit "normalisé", et tous les textes de la BDT auront cette même forme normalisée.

L'avantage de cette solution est de rendre la BDT indépendante des périphériques d'entrée et de sortie qu'elle utilise. Il suffira de lui adjoindre, pour chaque type de périphérique, un interface spécialisé qui indique la forme des attributs spécifiques à ce périphérique (cf figure II.11).

Le stockage des attributs locaux dans la BDT consistera alors à associer à chaque noeud terminal d'une structure effective, c'est-à-dire à chaque "feuille" de la réalisation d'un document, une table qui indique pour chaque attribut local de cette feuille: sa nature et l'adresse de la sous-chaîne concernée (ou les adresses éventuellement). Cette implantation est relativement lourde, elle ne se justifie que si la BDT est destinée à être utilisée avec des phériphériques nombreux et divers. C'est cette méthode qui a été utilisée dans la maquette SOLAR.

Des solutions intermédiaires existent aussi : par exemple paramétrer les processeurs d'entrée et de sortie en fonction des périphériques utilisés, afin d'obtenir un texte interne qui soit normalisé, mais avec les attributs locaux insérés dans la chaîne de caractères de ce texte (sous forme normalisée) et non stockées séparément. C'est cette dernière solution qui nous parait la plus simple à mettre en oeuvre, tout en restant générale, et que nous avons adoptée dans le projet MIDOC (voir chapitre IV.2.2).

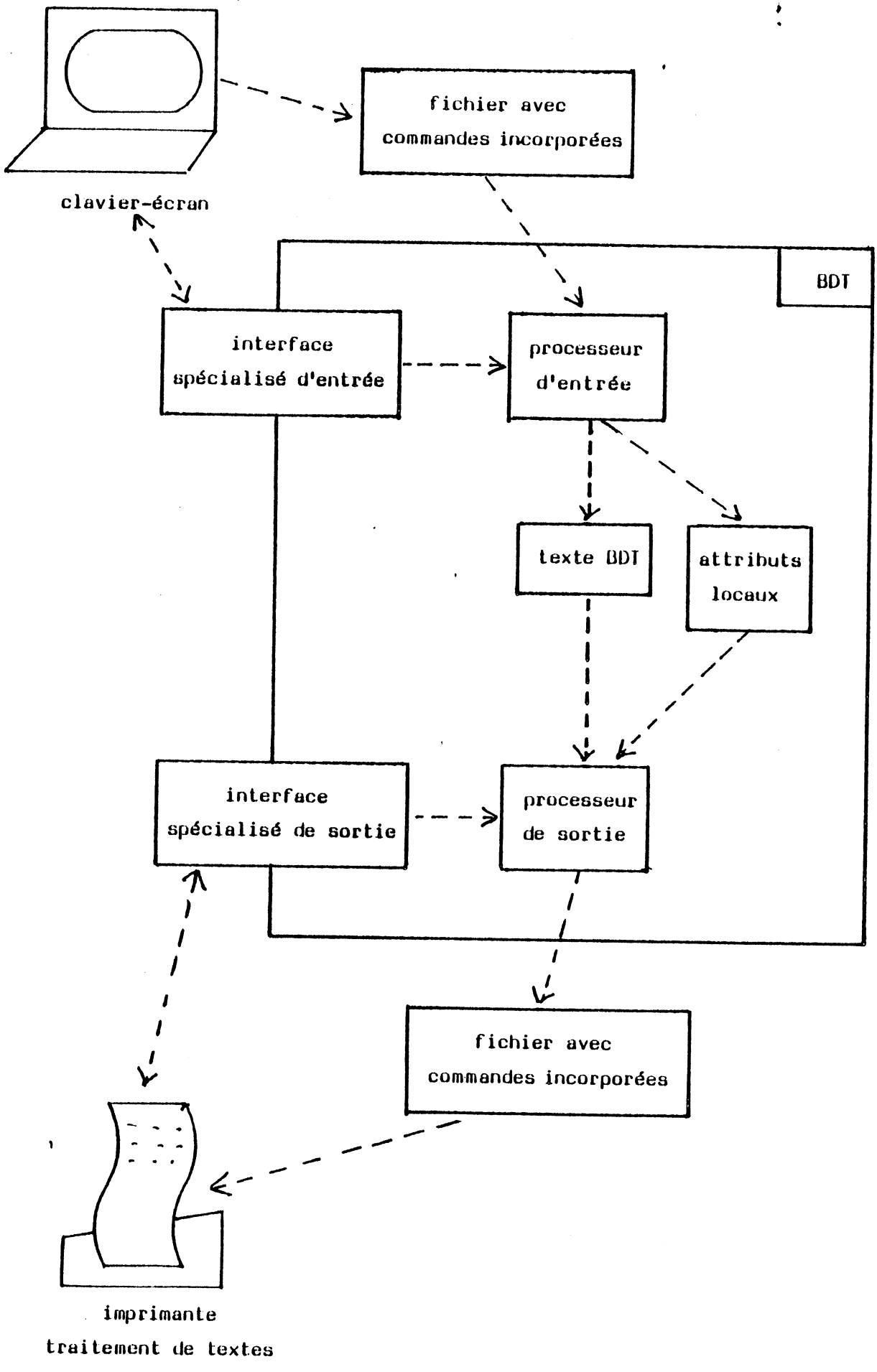

fichier avec
commandes incorporées
clavier-écran
BDT
interface
spécialisé d'entrée
processeur
d'entrée
texte BDT
attributs
locaux
interface
spécialisé de sortie
processeur
de sortie
fichier avec
commandes incorporées
imprimante
traitement de textes

figure II.11

II.3.4 - MOTS-CLES

Pour pouvoir rechercher des documents en fonction de leur contenu, et sans se préoccuper de leur type, il est possible de les indexer sur un certain nombre de mots-clés, puis de constituer un lexique de ces mots-clés avec des références aux documents concernés.

Il semble nécessaire d'autoriser une indexation "dynamique", c'est-à-dire modifiable au cours de la vie du document.

Parmi les problèmes posés par l'indexation, signalons

- l'inflation de la taille du lexique par l'emploi de mots-clés très proches, ou quasi-synonymes, ce qui réduit en même temps l'interêt d'une recherche sur l'un de ces mots. Ainsi dans un système de bibliothèque, on trouve les mots-clés :

  - programme
  - programmes
  - programmation
  - langage de programmation

et il devient très difficile d'y retrouver un ouvrage se rapportant à la programmation sans le rechercher à travers tous ces mots-clés. Il faut donc pouvoir consulter le lexique existant au moment de la création des index, afin d'éviter d'y ajouter des mots-clés trop proches de ceux qui y figurent déjà. Une solution plus élégante serait bien entendu la constitution automatisée de thésaurus, sur laquelle des travaux se poursuivent actuellement (BRUA82).

- Les erreurs dues aux homonymes, c'est-à-dire aux mots qui ont deux ou plusieurs sens différents selon le contexte : par exemple "arbre" en tant que végétal, et "arbre" en tant que structure hiérarchique. Une solution consiste à disposer d'une indexation à deux niveaux : le premier indique un champ d'interêt, le second un mot dans ce champ. Il faut alors disposer de lexiques spécialisés pour chaque champ d'interêt, et préciser tant au moment de la création des index que de la recherche d'un document, de quel champ d'interêt il s'agit. On pourrait ainsi distinguer BOTANIQUE/ARBRE de INFORMATIQUE/ARBRE, en s'interessant au lexique soit de la BOTANIQUE, soit de l'INFORMATIQUE.

II.3.5 - PARAMETRES

Nous considérons comme "paramètre" une chaîne de caractères qui n'est pas saisie en même temps que son contexte, mais qui devra être insérée au moment de la sortie du document physique. La place et la nature de cette chaîne seront indiquées dans la chaîne de caractères fixe (le contexte) par la présence d'un "paramètre formel". Celui-ci sera remplacé ultérieurement par une valeur, le "paramètre effectif".

Par exemple, on pourrait trouver dans un document de type "lettre personnalisée" :

Monsieur $Nom
$ rue
$ Ville Paris le 1-1-83

Cher Monsieur $Nom,
Nous avons le regret de constater que parmi nos lecteurs de la ville de $Ville, vous êtes un des rares à ne pas avoir renouvelé votre abonnement....

Les paramètres formels sont tous précédés ici du symbole réservé "$", afin de les différencier des mots "nom", "rue", "ville". Les paramètres effectifs destinés à les remplacer pouraient être :

$Nom <-- Dupond
$Rue <-- 25 Cours Jean-Jaurès
$Ville <-- Meylan

La liste de paramètres formels doit être associée au document. On peut choisir d'associer à chaque type une liste de paramètres formels, dans ce cas tous les documents d'un même type auront une même liste de paramètres formels (sans nécessairement les utiliser tous). Il y a là une analogie avec la notion de canevas, également associé au type. Mais on peut aussi considérer que la liste des paramètres formels est une caractéristique externe associée à chaque document virtuel.

En ce qui concerne la position des paramètres dans le texte, on peut gérer une table des paramètres, associant chaque paramètre formel à son adresse dans la chaîne de caractères (comme les attributs locaux, mais au niveau de l'ensemble du document et non de chaque feuille), ou bien on peut laisser le soin au processeur de sortie de "repérer" les paramètres formels dans la chaîne. Ces deux solutions sont analogues à celles qui sont proposées pour les attributs locaux.

L'opération d'affectation de valeurs aux paramètres permettra de créer des documents, destinés à être imprimés.

Plusieurs types différents de paramètres peuvent être envisagés :

- chaînes de caractères (exemple ci-dessus)
- paramètres calculés : ils renvoient à une procédure qui délivre à son tour une chaîne de caractères
- paramètres reférences : ils fournissent un élément de la structure d'un autre document, et c'est la chaîne de caractères correspondante qui sera prise en compte (analogue à un adressage indirect)
- paramètres structurés : divisés en champs, chaque champ étant lui-même un paramètre de type quelconque
- paramètres tableaux : ensemble de lignes et de colonnes nommés et pourvus de légendes, et dont chaque élément est une chaîne de caractères.

On pourra soit saisir ces valeurs interactivement une à une, soit les prendre successivement dans un fichier ou une "entité" de la base. On réalise dans ce cas automatiquement un ensemble de documents où seuls les paramètres diffèrent.

Les valeurs des paramètres peuvent éventuellement jouer le rôle de mots clés pour l'accès aux documents

ex: si $Ville est un paramètre-mot-clé, on pourra retrouver tous les documents ayant le paramètre $Ville = Meylan.

Cela implique la nécessité au préalable de créer un lexique des paramètres-mots-clés, et d'indexer les documents sur ces mots, puis de faire une recherche (filtrage) parmi les documents concernés pour retrouver ceux dont le paramètre-mot-clé possède la valeur souhaitée.

## II.4 - REALISATIONS PRATIQUES DE CE MODELE

On peut à priori envisager deux manières de construire un système de gestion de données textuelles :

- soit réaliser un système de gestion de base de données spécifique dont le modèle permette de représenter directement les objets (textes structurés en particulier) qui nous intéressent.

- soit bâtir à partir d'un système de gestion de base de données préexistant, en lui ajoutant une "couche" qui masque dans une certaine mesure le modèle initial.

C'est la deuxième option que nous avons choisie pour plusieurs raisons:

-l'utilisation d'un SGBD qui manie déjà les entités classiques, permet d'ajouter les entités documentaires et d'utiliser les deux;

-l'intégration de plusieurs applications, tant classiques que documentaires, devient possible sans que la taille des logiciels devienne prohibitive.

Par ailleurs, la difficulté de réaliser un mécanisme efficace et sûr de gestion de la mémoire est grande, et nous avons profité de travaux déjà effectués dans ce domaine, qui est à la base de tout SGBD.

Notre travail a donc consisté à ajouter à un SGBD classique:

- le modèle documentaire
- les fonctionnalités concernant les documents
- l'interface utilisateur qui permet de le manier.

Le choix du SGBD qui servira de base dépend principalement de sa disponibilité sur le type de matériel dont on dispose, et de sa facilité d'utilisation.

Nous avons travaillé à deux réalisations distinctes, dans des environnements très différents, et avec des objectifs qui ont évolué.

La première (maquette "Solar" voir chapitre III) visait principalement à vérifier et clarifier les hypothèses sur le schéma documentaire exposées ci-dessus. La plupart des concepts de ce schéma documentaire y ont été effectivement implantés, quoique quelquefois sous une forme difficile à utiliser. Elle intégrait données classiques et données textuelles, sans grandes contraintes de place mémoire ni de temps d'exécution. Elle ne visait pas particulièrement, dans un premier temps, une catégorie déterminée d'utilisateurs. Elle a permis de montrer la faisabilité d'une BDT, et devrait servir de point de départ à des travaux bien plus évolués, tels que le projet TIGRE.

La seconde (projet MIDOC voir chap IV) reprend de nombreux concepts de la première et cherche à les mettre en oeuvre dans un environnement différent : le microordinateur. Les contraintes d'espace mémoire deviennent alors très fortes. Comme par ailleurs, nous cherchons à réaliser un interface utilisateur qui soit très facile à utiliser, nous nous bornons dans un premier temps à un système qui manie les données textuelles, sans chercher à y intégrer les données classiques, pour lesquelles il faudrait étendre l'interface.

# CHAPITRE III

# LA MAQUETTE SUR SOLAR

## III.1 - OBJECTIFS ET MOYENS .

La réalisation d'une maquette de base de données étendue aux textes était initialement la première étape d'un projet : cette maquette devait servir de "banc d'essai" pour la réalisation ultérieure d'un prototype industriel, qui utiliserait un processeur autonome, l'"analyseur structurel de textes" (AST) (cf chap III.6).

Cet AST n'existait pas encore (et à notre connaissance n'existe pas à l'heure actuelle), et devait donc, pour la première étape, être simulé par logiciel. Le but de la maquette était donc de préciser, en les réalisant, tous les aspects du modèle de document dans une BDT.

La maquette a été réalisée sur un SOLAR 16-65, disposant d'une mémoire centrale de 192 KO et de deux disques durs amovibles de 5 MO, soit 10 MO au total, sous le système BOS-C. Le SGBD qui a servi de couche de base était Socrate (version 1.7).

Le système Socrate comporte un langage de requêtes avec des variables et des structures de contrôle qui permettent, à partir de citations et requêtes élémentaires, de bâtir des requêtes complexes équivalentes à des programmes. Celles-ci peuvent être stockées dans des macro-instructions et/ou des programmes précompilés, puis utilisées à leur tour dans d'autres requêtes ou programmes. Il est aussi possible d'avoir des programmes écrits en assembleur (programmes IMT). Socrate nous a donc fourni à la fois le SGBD et le langage de programmation pour la couche BDT.

Nous disposions de plusieurs terminaux, mais notre version de SOCRATE ne prévoyait pas de sécurités pour l'usage en multi-postes, la version multi-utilisateurs de la BDT n'a donc pas été très convaincante.

L'éditeur sous BOS-C est un éditeur de programmes, à adressage par lignes, très rudimentaire, il n'est en rien comparable à un traitement de textes. Par contre, nous disposions d'une imprimante traitement de textes (DIABLO).

## III.2 - LE MODELE ENTITE-DOCUMENT DANS LA MAQUETTE

Rappelons l'exemple de modèle entité-document présenté dans le chapitre II.2 :

```
entite PERSONNE
début
     NOM : alpha ;
     AGE : entier ;
     TAILLE : réel ;
     ADRESSE : alpha ;
     PUBLICATION : réfère un LIVRE;
fin

document LIVRE
début
     AUTEUR :alpha ;
     PRIX : entier ;
     PUBLICATION : date ;
     TYPE : BOUQUIN ;          ==========> renvoi au schéma
fin                                        documentaire
```

Dans la maquette, nous avons un peu modifié ce schéma, en créant un nouveau domaine de définition de caractéristique, destiné à relier le schéma conceptuel et le schéma documentaire. Ce nouveau domaine s'écrit: DOC (<liste de noms de types>).

Ainsi, on pourrait avoir :

```
entité LIVRE
début
     AUTEUR : alpha ;
     TITRE : alpha ;
     PUBLICATION : date ;
     CONTENU : DOC (BOUQUIN)
fin
```

Nous dirons que toute entité qui possède une caractéristique du domaine DOC est une ENTITE DOCUMENTAIRE. Il peut y avoir plusieurs types, parmi lesquels on choisira un type pour la réalisation de chaque document.
Le type sera représenté dans le système par un FORMAT, c'est le terme que nous emploierons donc désormais dans la description de la maquette.
Pour simplifier, nous admettrons qu'une entité documentaire ne peut contenir qu'une caractéristique DOC, et nous dirons donc qu'on associe :

- une entité documentaire : ici LIVRE

à - un document : CONTENU de LIVRE,

- ce document étant conforme au format BOUQUIN.

Dans le langage SOCRATE employé dans la maquette, la formulation devient plus précisément:

```
ENTITE (50) LIVRE
DEBUT
     AUTEUR  MOT(20)
     TITRE MOT(30)
     AN-PUBLICATION DE 1500 A 2000
     CONTENU DOC (BOUQUIN)
FIN
```

En effet, on indique le nombre maximum de réalisations de cette entité (ici 50 LIVRES) ainsi que le domaine de définition de chaque caractéristique (on peut indiquer la longueur des mots, et un intervalle pour la date).

Une entité du schéma conceptuel peut faire référence à plusieurs entités documentaires.

Exemple:

```
ENTITE (10) EMPLOYE
DEBUT
     NOM MOT(20)
     SERVICE MOT(20)
     CONTRAT REFERE UN CONTRAT-TRAVAIL
     CV REFERE UN CURRICULUM-VITAE
FIN

ENTITE (10) CONTRAT-TRAVAIL
DEBUT
  TEXTE DOC (CONTRAT)
FIN

ENTITE (10) CURRICULUM-VITAE
DEBUT
  CURRVIT DOC (C-V)
FIN
```

## III.3 - FONCTIONNALITES ET LANGAGES

### III.3.1 - LES ACTIONS SUR LES ENTITES

#### III.3.1.1 - DEFINITION D'ENTITES

Lorsque l'on définit une entité ayant la caractéristique DOC, une macroinstruction générera automatiquement deux définitions: celle d'une entité Socrate classique, et celle d'une entite DOCUMENT, ainsi que les références qui les lient l'une à l'autre. Mais cette structure réèlle est invisible pour l'utilisateur qui ne voit que les entités "documentaires" qu'il a définies.

SAISIE DIRECTE

La requête :

DEFM

permet de saisir une définition d'entite, classique ou documentaire, en la frappant directement à la console.

Cette définition doit être précédée de 'D' et suivie de 'FIN ?'

Exemple:

```
D
ENTITE (40) MANUEL
DEBUT
  TEXTE DOC (MAN-UTILISATEUR, MAN-OPERATIONS)
FIN
FIN?
```

N.B. Dans cet exemple, on propose deux formats possibles pour une même classe de documents.

SAISIE DANS UN FICHIER

La requête :

DEFF <NOM-FICHIER>

permet de saisir une définition d'entite figurant dans le fichier désigné par <NOM-FICHIER>. Le contenu de ce fichier doit être identique à celui qu'on aurait pu frapper à la console.

### III.3.1.2 LE LANGAGE DE CITATION EXTERNE

Les entités du schéma conceptuel sont des entités du langage Socrate, le langage de citation utilisé à ce niveau est donc le langage de citation Socrate. Rappellons qu'il permet l'utilisation de variables Xi (type référence à une entité), Yi (type numérique) et Zi (type chaîne de caractères). Nous avons introduit des variables de type 'REFERENCE A UN DOCUMENT' nommées

Di (1<= i <= 9)

qui permettent d'établir la liaison entre l'entité documentaire dans le schéma conceptuel et le document correspondant.

### III.3.1.3 - REQUETES SUR LES ENTITES CLASSIQUES

Ce sont les requêtes de Socrate qui permettent respectivement de:

Générer : G
Modifier : M
Imprimer : I
Supprimer: S

Exemples

```
G UN EMPLOYE X1
M SERVICE DE UN EMPLOYE X1 = 'INFORMATIQUE';
I NOM DE TOUT EMPLOYE AYANT SERVICE = 'INFORMATIQUE';
S UN EMPLOYE AVEC NOM = 'DURAND';
```

Nous avons aussi fourni la possibilité d'obtenir une liste d'entités triées selon les valeurs d'une caractéristique, au moyen d'une des deux requêtes suivantes :

TRILISTE <NOM D'ENTITE> SELON <NOM DE CARACTERISTIQUE > ;

TRILISTEF < NOM D'ENTITE FILTREE> SELON <NOM DE CARACTERISTIQUE> ;

Exemples:

TRILISTE LIVRE SELON AUTEUR;

TRILISTEF LIVRE AYANT AUTEUR = 'DUPOND' SELON AN-PUBLICATION;

III.3.1.4 REQUETES SUR LES CARACTERISTIQUES 'DOCUMENT'

CREATION

La requête

GEND <NOM D'ENTITE DOCUMENTAIRE>

permet de créer ("générer") une entité documentaire et le document de sa caractéristique 'DOCUMENT'.

Le système demande à l'utilisateur : le nom du format du document et le nom du canevas d'entrée. Un contrôle de cohérence est effectué entre nom de format et nom de canevas ( en effet, un canevas est toujours défini en liaison avec un format, cf chapitre III.3.7).

Deux possibilités sont offertes à l'utilisateur pour cette génération :

- analyse du texte global du document, à l'aide de l'AST (cf chapitre III.6). Ce texte peut exister ou être créé à la génération. Il doit contenir les séparateurs prévus dans le canevas d'entrée, afin de différencier les parties du document.

- génération interactive. Dans ce cas le document est créé à partir du format nommé et chaque partie fait l'objet d'un dialogue : elle peut être vide, saisie à la console ou être copiée à partir d'un autre document. Le système est guidé par le format pour demander la saisie au niveau des feuilles. Les parties facultatives peuvent ne pas être créées, les parties itératives peuvent être créées en nombre quelconque.

AFFECTATION DES VARIABLES Di

La requête

AFFD <Di> = <CITATION SOCRATE D'UNE ENTITE DOCUMENTAIRE>

permet de pointer, par le Di, sur la caractéristique 'document' de l'entité documentaire, c'est-à-dire sur le document correspondant.

Exemples :

AFFD D1 = CONTENU DE UN LIVRE 4 ;
AFFD D4 = CONTRAT DE UN EMPLOYE;

MISE A JOUR

La requête

MAJD <CITATION SOCRATE D'UNE ENTITE DOCUMENTAIRE> = <Di>;

permet de mettre à jour la caractéristique 'document' de l'entité documentaire désignée, avec le document pointé par le Di.

Exemple :

MAJD CONTENU DE UN LIVRE 3 = D5;

SUPPRESSION

La requête

SUPD <CITATION SOCRATE D'UNE ENTITE DOCUMENTAIRE>;

permet de supprimer l'entité documentaire ainsi que la caractéristique 'document' (le document correspondant) si celle-ci est referencée uniquement par cette entité.

Exemple:

SUPD UN CONTRAT-TRAVAIL;

## III.3.2 ACTIONS SUR LES DOCUMENTS

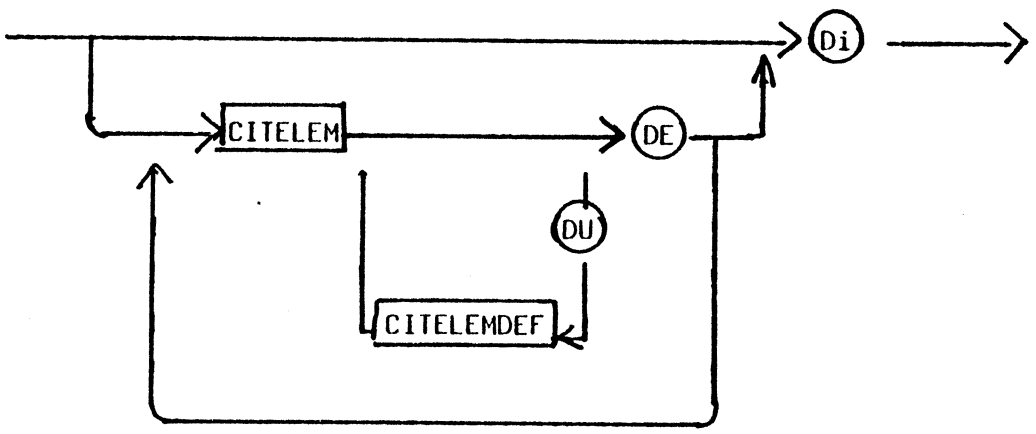
### III.3.2.1. LE LANGAGE DE CITATION INTERNE

C'est le langage qui permet de citer tout ou partie de document, ce document étant pointé par un Di.

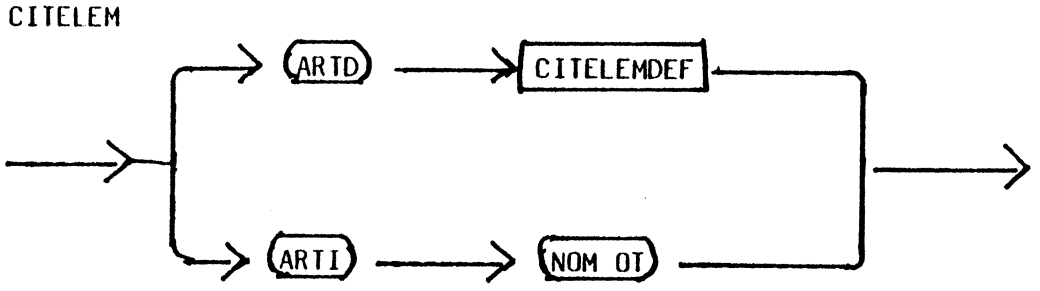
S'il s'agit d'un document complet la citation est simplement le Di. S'il s'agit d'une partie de document, la citation sera formée d'une suite de citations élémentaires 'CITELEM' séparées par 'DE', suivie de 'DE Di'.

Le langage de citation peut être formalisé par la grammaire suivante :

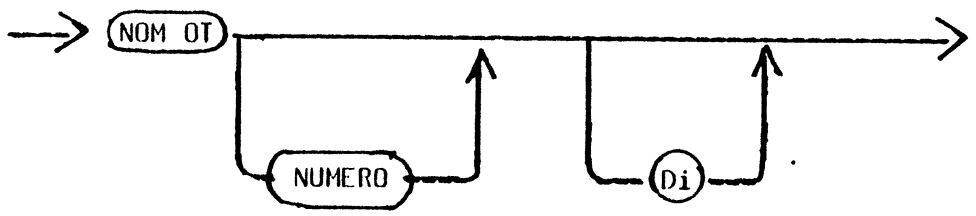
CITATDOC




CITELEM




CITELEMDEF



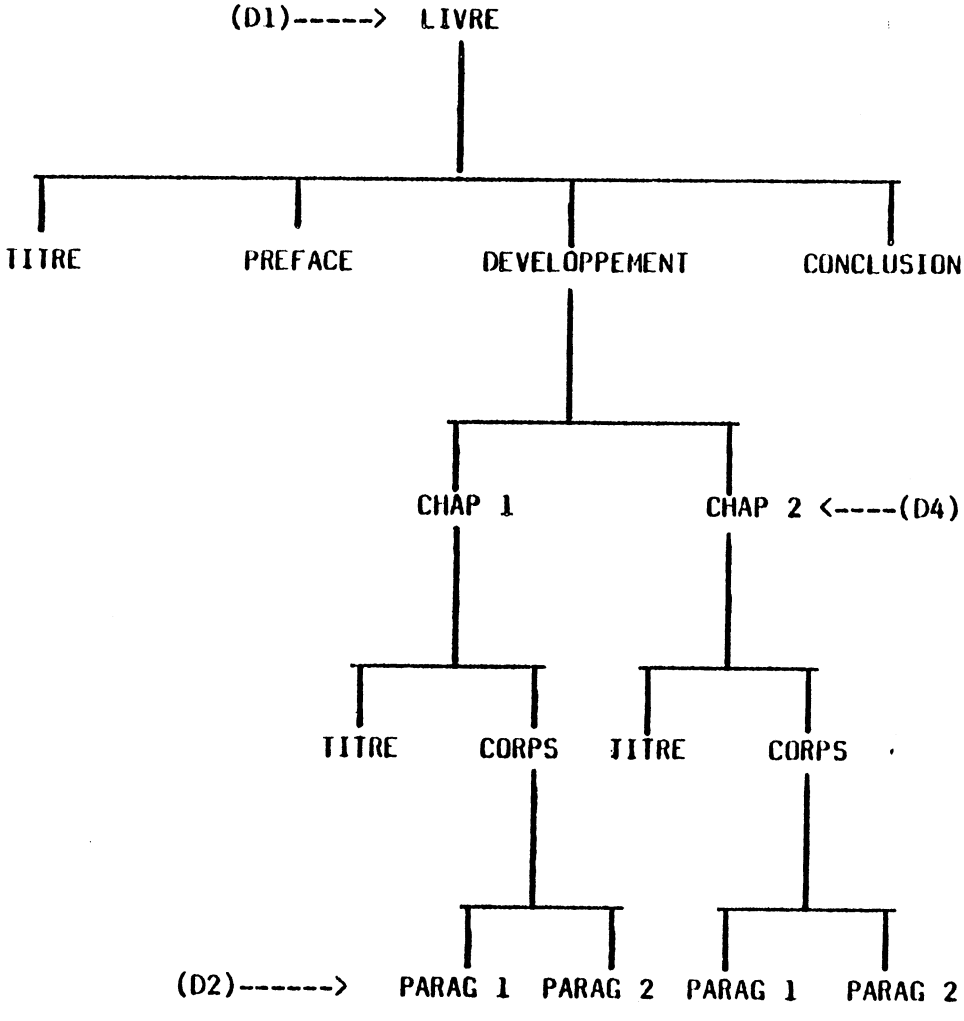
(ARTI) représente "UN" ou "UNE"
(ARTD) représente "LE" ou "LA" ou "L'"
(DE) représente "DE" ou "D'"
(NOM OT) représente l'identificateur d'un noeud de la structure ("objet textuel" ou "OT")

Quand une citation élémentaire se termine par un Di la partie de document désignée par cette citation élémentaire est affectée au Di.

La citation élémentaire <u>définie</u> (CITELEMDEF) est toujours précédée d'un article défini ou de DU. Elle s'applique lorsque l'OT en question est non répétitif, ou bien s'il porte un numéro. La citation élémentaire <u>indéfinie</u>, qui commence par l'article indéfini, s'applique lorsque l'OT en question est répétitif, et signifie qu'on s'intéresse au numéro 1.

EXEMPLE :

Si on a le descripteur suivant



et si D1 est un pointeur sur ce document, les citations suivantes sont possibles :

LA PREFACE DE D1

LE TITRE DU CHAP 2 D4 DE D1 : D4 est affecté au CHAP 2 de D1

LE PARAG 3 DE D4

UN PARAG D2 D'UN CHAP DU DEVELOPPEMENT DE D1 : D2 est affecté au PARAG 1 du CHAP 1 du DEVELOPPEMENT de D1

Remarques :

1) Le parcours n'est pas toujours donné explicitement: ainsi dans la seconde citation donnée en exemple cidessus on a sauté l'étape DEVELOPPEMENT.

2) S'il y a équivoque (même identificateur sur plusieurs noeuds) et si le parcours n'est pas donné explicitement, c'est la première occurrence dans un parcours 'par niveau' du descripteur qui est prise. Il faudra donc dire : LE TITRE DU CHAP 2 DE D1 pour éviter la confusion avec LE TITRE DE D1.

Notons que ce langage de citation ne peut pas être directement interprété en Socrate. Il a donc fallu écrire un petit compilateur qui le transforme en une suite d'instructions élémentaires de Socrate. Ce compilateur détecte les erreurs éventuelles dans les citations, tant dans la phase d'analyse lexicale puis syntaxique (erreurs de syntaxe), qu'à l'exécution (citation d'un OT inexistant ou d'un Di indéfini), et affiche les messages correspondants (GREN80) (KOWA80).

L'AFFECTATION D'UN DI

La requête

AFFECTER <CITATDOC> A <Di>

permet d'affecter un document ou une partie de document désigné par <CITATDOC> à un Di.

Exemples :

AFFECTER LE CHAP 2 DE D3 A D4

AFFECTER LE PARAG 2 DU CHAP 1 D4 DE D2 A D5

Dans ce dernier exemple, deux affectations sont faites:

D4 pointe sur le CHAP 1 de D2
D5 pointe sur le PARAG 2 de D4

L'affectation d'un document complet n'est en fait que la copie de référence d'un Di dans un autre puisque la citation d'un document est toujours un Di :

AFFECTER D2 A D4

III.3.2.2. LA VISUALISATION D'UN DOCUMENT

La requête

VISUALISER <CITATDOC>

permet de visualiser à l'écran le texte de la partie de document désignée par <CITATDOC>. Le système demande à l'usager s'il souhaite connaitre seulement la structure du document, seulement le texte ou les deux.

Exemple :

VISUALISER LE CHAP 1 DE D1

1) Avec la demande de structure seule, on obtiendra :

CHAP 1 = TITRE CORPS
CORPS = PARAG 1 PARAG 2

2) Avec la demande de texte seul on obtiendra le texte correspondant, feulle par feuille, c'est à dire successivement le titre, le parag 1 et le parag 2.

3) Avec la demande de structure et de texte on obtiendra l'édition de structure comme ci-dessus, puis l'édition du texte.

### III.3.2.3. LA MISE A JOUR DE DOCUMENT

Les requêtes suivantes permettent de modifier un document (modification de texte ou de descripteur), mais toujours en se conformant au format de ce document.

#### EFFACEMENT DE TEXTE

La requête

EFFACER <CITATDOC>

permet de supprimer les références au texte.

Tout se passe comme si le texte associé au document ou la partie du document avait été effacé, bien qu'en réalité le texte ne soit pas détruit ni modifié. En effet il est possible que d'autres documents se réfèrent à ce même texte. Par la suite on dira que les parties de document effacées sont vides.

## SAISIE DE TEXTE

La requête

SAISIR <CITATDOC>

permet de saisir le texte d'une partie seulement de document. Si cette partie de document est une feuille et possède déjà un texte, on demande à l'utilisateur s'il veut le modifier.

Si aucun texte n'est rattaché à cette partie, le système suit le format du document et demande la saisie du texte au niveau des parties terminales (feuilles). Le dialogue est le même que pour la requête GEND donnée ci-dessus.

## SUPPRESSION D'UNE PARTIE DE DOCUMENT

La requête

SUPPRIMER <CITATDOC>

permet de supprimer une partie de document, à condition que cette partie soit :

- un élément facultatif

- un élément itératif dont le nombre peut être quelconque.

Si la partie du document citée est un élément obligatoire permanent ou bien si la citation correspond à un document complet, aucune suppression n'est effectuée mais un message indique les possibilités offertes pour ces cas là.

Exemple: SUPPRIMER LE PARAG 2 DU CHAP 1 DE D1 effectuera cette suppression, alors que

SUPPRIMER LE TITRE DU CHAP 1 DE D1 fera afficher le message: "Suppression impossible, mais vous pouvez effecer cette partie".

## REMPLACEMENT D'UNE PARTIE DE DOCUMENT

La requête

REMPLACER <CITATDOC>

permet de remplacer une partie de document. Le système propose le choix entre :

- L'effacement de texte. Tout se passe alors comme dans EFFACER <CITATDOC>

- La saisie d'un nouveau texte. On crée alors un nouveau descripteur selon le format associé et on saisit le texte à chaque niveau terminal, comme dans les requêtes SAISIR et GEND.

- Le remplacement du descripteur et de son texte par la copie d'une autre partie de document. Le système demande alors de fournir la CITATDOC indiquant la partie remplaçante.
La copie n'est effectuée que si les deux parties ont le même type et la même structure.

## INSERTION D'UNE PARTIE DE DOCUMENT

La requête

INSERER <CITELEM> DANS <CITATDOC>

permet d'insérer une partie dans un document, à condition que cette partie soit :

- un élément facultatif non existant

- un élément répétitif

Dans tous les autres cas, un message d'erreur est émis par le système.

Si la citation <CITELEM> se termine par un DI, et si l'insertion est faite, alors le DI devient un pointeur sur la partie insérée.

Pour cette insertion l'utilisateur a le choix entre :

- la création d'une partie vide

- la copie d'une autre partie de document ayant même type et même structure.

- la création d'une partie dont le texte est entré par saisie directe, en tout ou en partie.

### III.3.3 LES PARAMETRES

#### III.3.3.1 LES TABLEAUX

Les tableaux qui peuvent figurer dans les documents sont manipulés à l'aide de variables de type "référence à un tableau", nommées

Ti ( 1<= i <=9 )

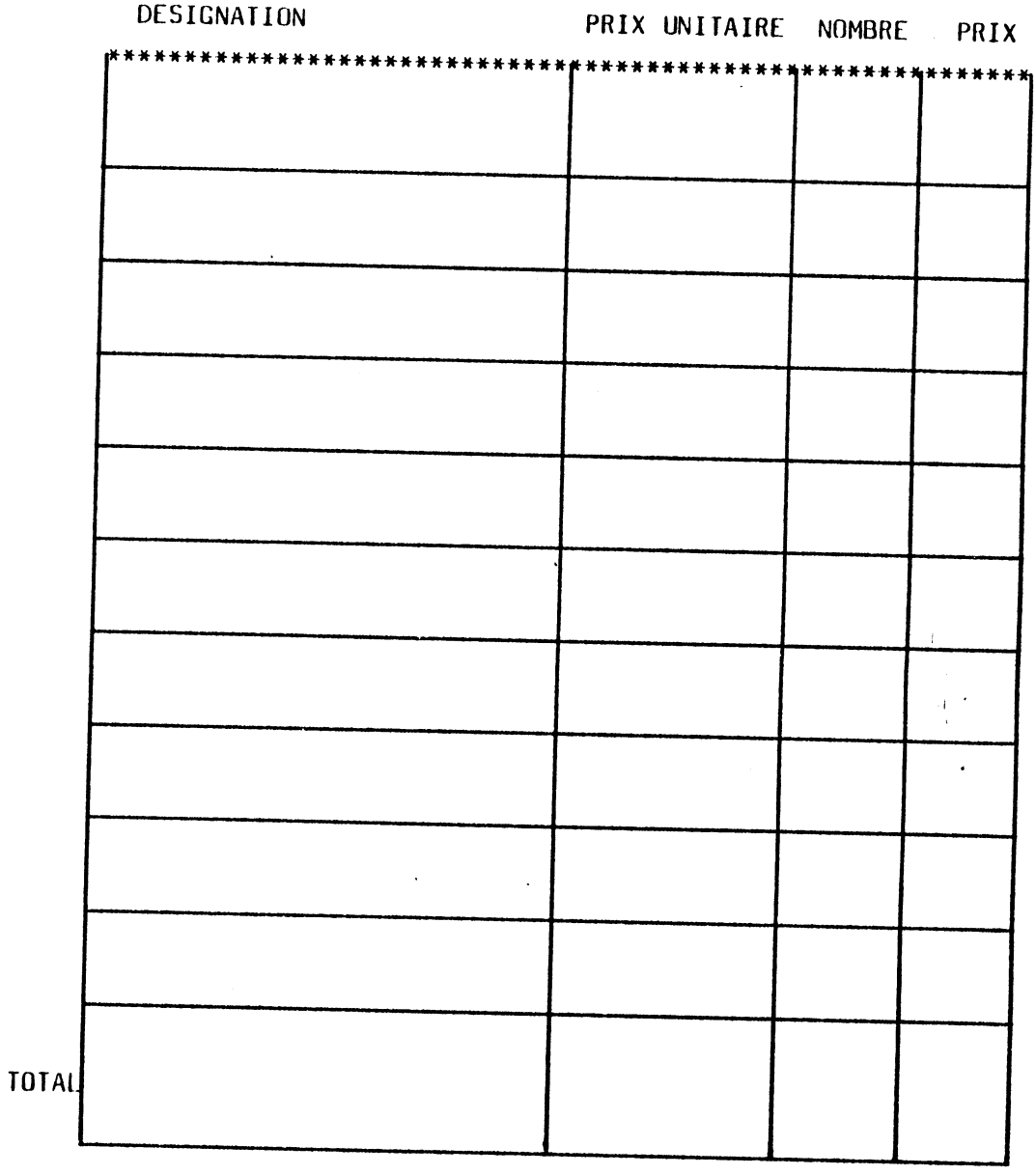
analogues aux variables Di définies pour les documents.

Les tableaux sont formés de lignes et de colonnes, chaque ligne et chaque colonne porte un nom, qui peut être soit un identificateur, soit un numero entre 1 et le nombre de lignes ou colonnes du tableau (cf chapitre III.3.6 et Annexe 3). De plus chaque tableau, chaque ligne, et chaque colonne, peuvent avoir une légende destinée à l'aspect externe du tableau, distincte du nom.

Exemple

L'on veut faire imprimer un tableau de la forme suivante:

COMMANDE




Ce tableau sera conforme à un type :

```
Commande  LIGNES DE 1 A 11;
          COLONNES nom pu nb prix;
```

et porte les légendes suivantes: COMMANDE pour l'ensemble du tableau, DESIGNATION, PRIX UNITAIRE, NOMBRE et PRIX pour les colonnes, et TOTAL pour la ligne numéro 11.

La requête

ATAB <Ti> = <NOM PARAM> DE <Di>

permet d'affecter un paramètre de type tableau désigné par <NOM PARAM>, appartenant au document désigné par <Di>, à la variable <Ti>.

Un certain nombre de requêtes permettent de saisir, mettre à jour et visualiser tant les valeurs des éléments d'un tableau que les légendes. Leur énumération détaillée a été reportée à l'Annexe 2. Nous en donnerons ici quelques exemples, portant sur le tableau qui figure ci-dessus, que l'on suppose désigné par T2.

La requête

SLIG 3 DE T2

permet de saisir les valeurs de la ligne 3 du tableau, au moyen d'un dialogue suivant (les réponses frappées par l'utilisateur sont soulignées):

ELEMENT: 3 NOM ? Crayons HB

ELEMENT: 3 PU ? 2,50

ELEMENT: 3 NB ? 100

ELEMENT: 3 PRIX ?250

La requête

LCOL NB DE T2 = QUANTITE

permet de modifier la légende qui figure audessus de la troisième colonne.

La requête

VELE 11 PRIX DE T2

permet d'afficher à l'écran la valeur de l'élément de la ligne 11 et de la colonne PRIX, c'est à dire le montant total de la commande.

III.3.3.2. VALORISATION DES PARAMETRES

La requête FIXER <NOM PARAM> DE <Di> = <VALEUR PARAM>

affecte la valeur <VALEUR PARAM> au paramètre désigné par <NOM PARAM> dans le document désigné par <Di>.

Pour un paramètre 'chaîne' la valeur doit être une chaîne de caractères, pour un paramètre 'tableau' ce doit être un <Ti>.

Ce Ti doit auparavant avoir recu des valeurs au moyen des requêtes ATAB et STAB (Cf Annexe 2).

Exemple:

Si un document D3 possède un paramètre tableau nommé $COMM1, on exécutera successivement les requêtes

```
ATAB T1 = $COMM1 DE D3
STAB T1      ==>      saisie interactive
FIXER $COMM1 DE D3 = T1
```

La requête FIXERPARAM DE <Di>

permet de valoriser tous les paramètres du document désigné par <Di> de manière interactive. Le système propose chaque nom de paramètre, et attend la frappe d'une chaîne de caractères pour les paramètres chaîne, d'un Ti pour les paramètres tableau. Un retour-chariot donne une valeur indéfinie au paramètre.

La requête LISTERPARAM DE <DI>

affiche la liste des paramètres du document désigné par <DI> avec leur valeur si elle existe. Pour les paramètres de type 'TABLEAU' la valeur est représentée par la mention

TABLEAU : <NOM TABLEAU>.

(NB : Le tableau peut exister mais être vide)

## III.3.4 LA SORTIE DE DOCUMENTS

### VISUALISATION

La requête VDOC <Di>

permet de visualiser le document désigné par <Di> selon un canevas de sortie dont le nom est demandé interactivement. Le document sera affiché à l'écran sous la même présentation qu'une impression sur papier.

### VISUALISATION ET AFFECTATION SIMULTANEES DES PARAMETRES

La requête RDOC <Di>

permet de visualiser le document désigné par <Di> selon le canevas de sortie dont le nom est demandé interactivement, et pour chaque paramètre non affecté dans ce document, de le valoriser interactivement, de la même manière que dans la requête FIXERPARAM.

Le document est affiché à l'écran sous la même présentation qu'une sortie sur imprimante.

### IMPRESSION DE DOCUMENT

La requête EDITER <Di> SELON <NOM CANEVAS SORTIE>

permet d'imprimer sur papier le document désigné par <DI> selon le canevas de sortie désigné par <NOM CANEVAS SORTIE>.

### LE MAILING

Une opération de mailing consiste à faire imprimer en série un certain nombre d'exemplaires d'un document, en y faisant varier éventuellement quelques paramètres. Il s'agira par exemple de lettres dites "personnalisées". On appellera "cible" une entité qui contient les noms et adresses des destinataires du document.

La requête

DEFMAIL <NOM MAILING> DE <Di> SUR <NOM ENTITE> SELON <NOM CANEVAS SORTIE>

permet de définir une procédure de mailing au moyen d'un dialogue avec le système.

- <NOM MAILING> est l'identificateur de la procédure

- <Di> est le pointeur sur le document à diffuser

- <NOM ENTITE> est le nom de l'entité dont les réalisations seront la cible de la procédure de MAILING.

- <NOM CANEVAS SORTIE> désigne le canevas de sortie choisi pour la présentation externe du document.

Il est possible de définir

- un filtre sur l'entité cible du MAILING. Ce filtre s'exprime par une valeur particulière , ou bien une plage de valeurs, d'une caractéristique de cette entité. Il permet de sélectionner un sous-ensemble particulier de destinataires du mailing. Ce filtre est demandé interactivement, si l'utilisateur n'en veut pas il tape un retour-chariot

- des paramètres 'externes' c'est-à-dire valorisés au moment de lancer le MAILING.

- des paramètres 'autres'. Ils peuvent être

    * VARIABLES : la valeur dépend d'une caractéristique de l'entité cible.

La syntaxe de déclaration est :

    <NOM PARAM> SELON <NOM CARACTERISTIQUE> :
    <VALEUR CARACTERISTIQUE> => <VALEUR PARAM> ; ... ;

* FIXES : leur valeur est fixée directement par l'utilisateur.

La syntaxe de déclaration est:

<NOM PARAM> = <NOM CARACTERISTIQUE>.

Exemple

Soit la requête:

DEFMAIL RAPPEL-ABONNEMENTS DE D5 SUR ABONNE SELON C-LETTRE

On suppose que le document D5 contient une lettre de relance, que l'entite ABONNE contient les noms et adresses des abonnés, et que le canevas de sortie C-LETTRE est conforme au format de la lettre.

Le système demandera s'il y a un filtre sur ABONNE. On pourra par exemple répondre

38000 <= CODE-POSTAL de ADRESSE de ABONNE< 39000

si l'on ne s'intéresse qu'aux abonnés de l'Isère.

On donnera ensuite les paramètres, par exemple:

```
externes $DATE $LIEU
autres   $MMM selon SEXE de ABONNE
             'F' => 'MADAME';
             'M' => 'MONSIEUR';
         $NOM = NOM de ABONNE;
```

Ainsi, les paramètres $DATE et $LIEU recevront une valeur quand on lancera le mailing. Le paramètre $MMM vaudra 'MADAME' si la valeur de la caractéristique 'SEXE' de l'entité cible est 'F', 'MONSIEUR' si elle est 'M'. Le paramètre $NOM prendra la valeur de la caractéristique 'NOM' de l'entité cible.

Pour lancer le mailing, il suffira de frapper la requête :

M:<NOM MAILING>

soit, dans ce cas: M: RAPPEL-ABONNEMENTS

On peut également obtenir l'historique d'une procédure de MAILING par la requête

TRACER <NOM MAILING>

qui imprime la date et le nombre d'exemplaires produits, pour chaque utilisation du mailing.

### III.3.5 INDEXATION ET RECHERCHE DOCUMENTAIRE

Rappellons le principe que nous avons adopté pour l'indexation (cf chapitre II.3.4): les mots-clés peuvent être préfixés par le nom d'un dictionnaire, qui indique à quel centre d'intérêt le mot se rattache. Cette méthode doit permettre d'éviter certains problèmes d'homonymie, d'une part, et de restreindre les recherches à une liste limitée de documents, d'autre part.

#### III.3.5.1 LES MOTS-CLES

##### LES DICTIONNAIRES DE MOTS-CLE

La requête CREDICT <NOM DICTIONNAIRE>

permet de créer un dictionnaire ayant le nom <NOM DICTIONNAIRE> à moins qu'il n'existe déjà.

La requête IDICT <NOM DICTIONNAIRE>

permet d'imprimer la liste des mots-clé figurant dans le dictionnaire désigné par <NOM DE DICTIONNAIRE>

Exemples: CREDICT INFORMATIQUE
IDICT BOTANIQUE

La requête DEFMC DANS <NOM DICTIONNAIRE>

permet d'insérer une liste de mots-clé dans le dictionnaire désigné par <NOM DICTIONNAIRE>. La saisie des mots-clé s'interrompt par un retour-chariot.

L'INDEXATION D'UN DOCUMENT

La requête INDEXER <DI>

permet d'indexer le document désigné par <DI> en indiquant le nom des dictionnaires, chaque dictionnaire étant suivi de la liste des mots-clé que l'on a choisis pour ce document.

La saisie des mots-cle et des dictionnaires s'interrompt par un retour-chariot. L'indexation se fait dans tous les cas, la présence du mot-clé dans le document n'étant pas vérifiée.

Exemple: INDEXER D3; le dialogue suivant s'instaure (les réponses frappées par l'utilisateur sont soulignées):

DICTIONNAIRE? BASE DE DONNEES

MOTSCLES? RELATIONNEL
TEXTUEL
<rc>

DICTIONNAIRE? PROGRAMMATION

MOTSCLES? SOCRATE
<rc>

III.3.5.2 LA RECHERCHE DOCUMENTAIRE

La requête RECHERCHE

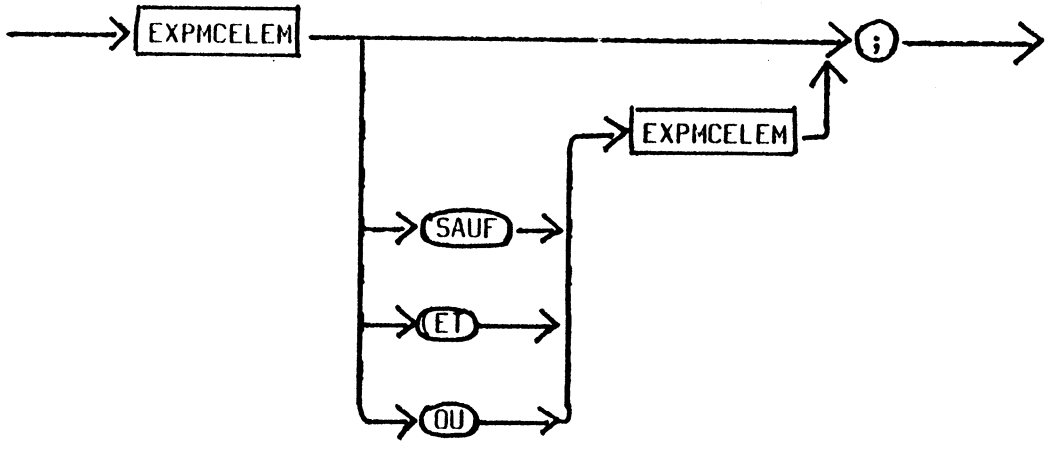
permet de rechercher des documents qui ont été, au préalable, indéxés avec des mots-clé. Le fonctionnement d'une session de recherche est interactif.

Le système demande d'abord à l'utilisateur s'il connait le format des documents recherchés. Si l'utilisateur frappe un nom de format, la recherche se limitera aux documents de ce format. Sinon, la recherche portera sur tous les documents, quel que soit leur format.

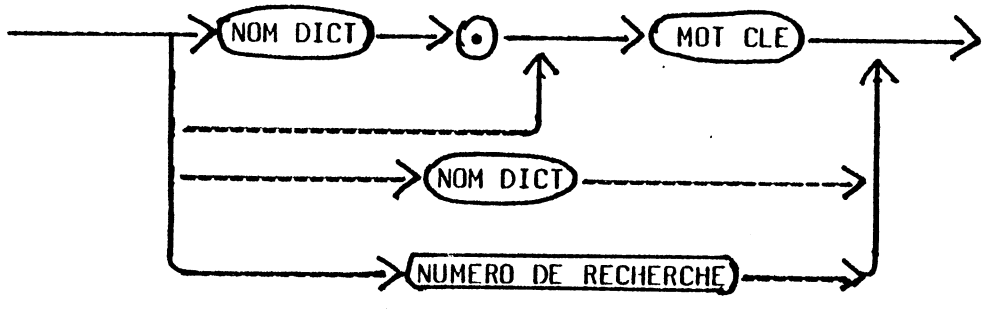
Ensuite, le système affiche le numéro qu'il a affecté à la recherche, l'utilisateur pourra se servir de ce numéro pour affiner la recherche ultérieurement.

L'utilisateur doit ensuite frapper une expression de mots-clé <EXPMC> dont la syntaxe est la suivante :

EXPMC




EXPMCELEM



L'expression élémentaire de mots-clé <EXPMCELEM> indique donc :

- un mot-clé, en précisant ou non à quel dictionnaire il appartient;

exemples: PROGRAMMATION.SOCRATE ou bien SOCRATE tout seul.

Le choix de préfixer ou non dépend évidemment du champ sémantique couvert par l'ensemble des documents de la base. S'il n'y a aucun document philosophique, dans cet exemple, il peut être inutile de préfixer.

- l'ensemble des mots-clé d'un dictionnaire;

on s'intéresse alors à tous les documents couvrant un centre d'intérêt.

- une expression de mots-clé déjà utilisée au cours de la même session de recherche, désignée par son numéro de recherche.

L'expression de mots-clés <EXPMC> permet d'obtenir l'intersection, l'union ou la différence de deux expressions élémentaires. En utilisant les numéros de recherche il est possible de former des expressions plus complexes.

Comme résultat de cette recherche le système donne le nombre de documents trouvés et demande à l'utilisateur s'il en veut la liste, auquel cas il obtient les noms et numéros des entités documentaires qui contiennent ces documents.

Exemple:

```
recherche numero 1: ARBRE OU STRUCTURE        ==> 28 documents
recherche numero 2: 1 ET FEUILLES             ==> 14 documents
recherche numero 3: 2 SAUF BOTANIQUE.         ==> 8 documents
recherche numero 4: LANGAGES.SOCRATE ET 3     ==> 3 documents.
```

On peut alors demander au système les références précises de ces trois derniers documents. Ils traiteront tous, du moins peut-on l'espérer, des feuilles de structures arborescentes, programmées en Socrate.

Notons que les quatre expressions de mots-clés successives que nous avons employés dans cet exemple sont équivalentes à l'expression:

(LANGAGES.SOCRATE) ET
(((ARBRES OU STRUCTURE) ET FEUILLES) SAUF BOTANIQUE)

Ce type de dialogue permet donc de construire des expressions complexes, sans avoir à fournir plus d'un opérateur logique à la fois.

### III.3.6 LES FORMATS

Le langage utilisateur pour les formats permet de définir les paramètres formels, chaînes ou tableaux, et les structures formelles.

Ces formats-source comportent trois parties: déclaration de tableaux, de paramètres, et de structure. Les deux premières sont facultatives.

La partie "déclaration de tableaux" permet de définir des types de tableaux, qui seront utilisés pour des déclarations de parramètres-tableaux. Le type de tableau doit exister lorsqu'on déclare un paramètre-tableau, mais il peut avoir été défini auparavant, dans un autre format. En effet, les types de tableaux sont mis en mémoire dans des entités particulières, indépendantes des formats, et les formats contiennent une caractéristique "référence" aux types de tableaux nécessaires .

La déclaration de chaque tableau comporte donc son nom, suivi de la déclaration soit du nombre de lignes, soit du nom de chaque ligne, puis de la déclaration soit du nombre de colonnes, soit du nom de chaque colonne. Ainsi chaque ligne et chaque colonne est nommée :

- soit par un numéro entre 1 et le nombre total de lignes ou colonnes

- soit par un identificateur.

La partie "déclaration de paramètres" permet de donner la liste de tous les paramètres formels que l'on peut rencontrer dans les documents associés à ce format, c'est-à-dire les identificateurs qui devront être

remplacés par les paramètres effectifs pour produire des documents physiques. Pour chque paramètre, on déclare si le paramètre effectif sera une chaîne de caractères ou bien un tableau, et dans ce dernier cas, de quel type de tableau il s'agira. Les types de tableaux constituent donc en quelque sorte des formats à un second niveau par rapport aux formats de documents :

document -----> format ------> types de tableaux

Enfin, la "déclaration de structure" permet de définir chaque élément de la structure, en indiquant s'il s'agit d'un "bloc" formé de plusieurs éléments différents (entre DEBUT et FIN), d'un "repete" formé d'éléments de même nom, ou d'une feuille, au niveau terminal. Pour indiquer qu'un élément est facultatif, on fait précéder son nom du signe $.

Un élément itératif est ipso facto facultatif, le $ n'y a donc pas de sens.

Exemples de définitions de format :

```
1.  FORMAT Carnet-commande
    TABLEAUX
        Commande LIGNES DE 1 A 20 ;
                COLONNES désignation prix quantité total ;

    PARAMETRES
        $DATE CHAINE ;
        $COMM1 TABLEAU commande ;
        $NOM CHAINE ;
        $ADRESSE CHAINE ;
        $COMM2 TABLEAU commande ;

    STRUCTURE
        DEBUT
            informations-diverses ;
            corps :
                REPETE client ;
            récapitulation
        FIN
    FIN-FORMAT
```

2. FORMAT manuel-technique

```
STRUCTURE
    DEBUT
        Titre ;
        Résumé ;
        Introduction ;
        Développement :
            REPETE chapitre :
                DEBUT
                    Titre ;
                    Corps :
                        REPETE Section
                FIN ;
        Conclusion
    FIN
FIN-FORMAT
```

La syntaxe décrite ci-dessus est celle qui est utilisée pour la saisie interactive d'un format :

le système demande la définition de chaque élément de la structure, et ne s'arrête que lorsque tous les éléments ont été définis jusqu'au niveau des feuilles (éléments vides). De même, l'affichage et la modification des formats suivent la même syntaxe.

Il s'agit donc d'une forme externe, celle du format-source. Les requêtes qui l'utilisent sont les suivantes (GREN81):

DFOR <NOM FORMAT> : saisie interactive

IFOR <NOM FORMAT> : affichage à l'écran

SFOR <NOM FORMAT> : destruction après vérification qu'aucun document n'utilise ce format

MFOR <NOM FORMAT> : modification interactive, également après vérification qu'aucun document n'utilise ce format. La modification (suppression ou insertion) peut s'appliquer à toutes les parties de format.

La saisie d'un format-source est accompagnée de sa transformation en un format-objet, c'est ce format-objet qui sera stocké dans la base et utilisé par l'ASI.

La forme interne de format, ou format-objet est formée de trois tables :

- les paramètres formels

- les modèles-OI

- Les def-OI.

La table des paramètres formels contient, pour chaque paramètre formel :

- son identificateur

- son type (chaîne ou tableau)

- une référence à une définition de tableau.

Les tables de modèles-OI et de def-OI contiennent pour chaque élément de la structure ( "objet textuel" ou "OI")(MIN080):

- son identificateur

- son mode : bloc, répète ou feuille (fils différents, fils itératifs, ou pas de fils)

- l'indication du fils aîné

- l'indication du frère cadet

- une variable booléenne qui indique si l'OI est itératif (père de mode répète) ou non

- une variable booléenne qui indique si l'OI est optionnel ou obligatoire.

### III.3.7 LES CANEVAS

Rappellons que la BDT propose que l'entrée et la sortie de documents soient gérées par des "canevas", liés aux formats, tant pour les paramètres que pour la structure. Chaque canevas se réfère donc explicitement à un format. Toutes les requêtes qui portent sur les canevas vérifient l'existence du format désigné et la cohérence du canevas avec ce format.

Le canevas d'entrée contient les identificateurs de paramètres (paramètres formels), ainsi que les chaînes séparatrices permettant de trouver le début et la fin du texte de chaque élément terminal de la structure. Ce sont ces données qui permettront à l'AST d'analyser le texte en entrée, de le structurer, et de construire les tables d'adresses de paramètres formels.

Il est traité à l'aide des requêtes

DCANVE <NOM CANEVAS ENTREE> saisie interactive

ICANVE <NOM CANEVAS ENTREE> impression

SCANVE <NOM CANEVAS ENTREE> suppression

MCANVE <NOM CANEVAS ENTREE> modification interactive (on demande l'ancien délimiteur, puis le nouveau qui doit le remplacer).

Le canevas de sortie contient toutes les indications pour la présentation: attributs de présentation générale, attributs de présentation liés à chaque OT.

Ces attributs sont les suivants:

- nombre de lignes par page, nombre de lignes blanches en haut et en bas de page

- marge gauche et droite

- interligne
- police de caractères

- justification: à droite, centrée, ou tel-quel

- numérotation des pages: absente ou absolue ou relative, en chiffres arabes ou romains

- cadrage des numéros sur chaque page

- entêtes éventuels, à imprimer en haut ou en bas de page, à gauche ou à droite

Ces attributs peuvent être définis au niveau du document complet, et s'il y a lieu modifiés pour chaque OI qui en fait partie. Des valeurs par défaut sont prévues pour toutes les options.

Pour les paramètres et les entêtes, on peut définir le nombre de lignes à sauter avant ou après, et la police de caractères si elles diffère de celle qui est employée ailleurs. On peut aussi demander l'impression d'un sommaire, au début ou à la fin du document.

Certaines caractéristiques peuvent être demandées explicitement: soulignement de titres, passage du ruban à l'encre rouge...

Les requêtes qui manipulent le canevas de sortie sont:

DCANVS <NOM CANEVAS SORTIE> saisie interactive du format du papier qui sera employé, puis de tous les attributs énumérés ci-dessus, pour l'ensemble du document et éventuellement chaque OI, en liaison avec la définition correspondante de format.

ICANVS <NOM CANEVAS SORTIE> impression

SCANVS <NOM CANEVAS SORTIE> destruction

MCANVS <NOM CANEVAS SORTIE> modification interactive de tout élément du canevas.

De plus, la requête LISTER <ID> permet de lister les noms de formats, canevas d'entrée, ou canevas de sortie, selon la valeur de <ID>: 'F', 'E', ou 'S'.

## III.4 L'INTERFACE UTILISATEUR

Dans tous les exemples de commandes du langage de requête donnés ci-dessus, nous pouvons remarquer que le format le plus fréquent et le plus simple est le suivant :

<requête> <citation>

par exemple :

REMPLACER <CITATDOC>

ou CREDICT <NOM DICTIONNAIRE>

Souvent le format devient plus complexe, les parties variables de la requête doivent être séparées par des séparateurs imposés :

EDITER <Di> SELON <NOM CANEVAS SORTIE>

ou FIXER <NOM PARAM> DE <DI> = <VALEUR PARAM>

Pour aider l'utilisateur, la requête

MENU ?

permet d'obtenir l'affichage de la syntaxe de toutes les commandes du système, au moyen d'une arborescence selon les catégories des commandes: il faut ensuite, selon la catégorie choisie, frapper un numéro pour obtenir un sous-menu, et ainsi de suite jusqu'à obtention du menu effectif des requêtes choisies.

Le premier menu affiché est le suivant:

MENU GENERAL
1 - ENTITES
2 - DOCUMENTS
3 - RECHERCHE DOCUMENTAIRE
4 - CANEVAS ET FORMATS
5 - STRUCTURE
6 - PRODUCTION

FRAPPEZ LE NUMERO DE LA LIGNE QUI VOUS INTERESSE:

Si l'on frappe 6, le menu affiché devient:

PRODUCTION DE DOCUMENTS
1 - PARAMETRES CHAINES
2 - PARAMETRES TABLEAUX
3 - MAILING
4 - SORTIE DE DOCUMENTS

FRAPPEZ LE NUMERO DE LA LIGNE QUI VOUS INTERESSE:

En frappant 1, on obtient:

PARAMETRES DE TYPE "CHAINE"
FIXERPARAM DE <DI> ?... VALORISATION DE TOUS LES PARAMETRES
LISTERPARAM DE <DI> ?... LISTE DES PARAMETRES
FIXER <NOM PARAM> DE <DI> = <VALEUR PARAM> ?... VALORISATION D'UN PARAMETRE

Ces menus permettent de rappeller à l'utilisateur la syntaxe des commandes, mais les explications fournies sont très suuccintes, et surtout, l'utilisateur devra toujours frapper lui-même le libellé complet de chaque requête, en effet les menus ne permettent pas l'appel interactif de celles-ci. Les messages d'erreur obtenus en cas de requête incorrecte sont ceux de Socrate.

Le système n'est donc pas, dans l'état de la maquette, orienté vers un utilisateur non informaticien. Pour pouvoir l'utiliser avec une certaine efficacité, il semble nécessaire de connaitre Socrate.

En effet : - les variables Di et Ti sont proches des variables classiques Xi de Socrate (référence à des entités : Xi, référence à des documents ou parties de documents : Di), et l'emploi de ces variables est nécessaire pour relier schéma conceptuel et schéma documentaire.

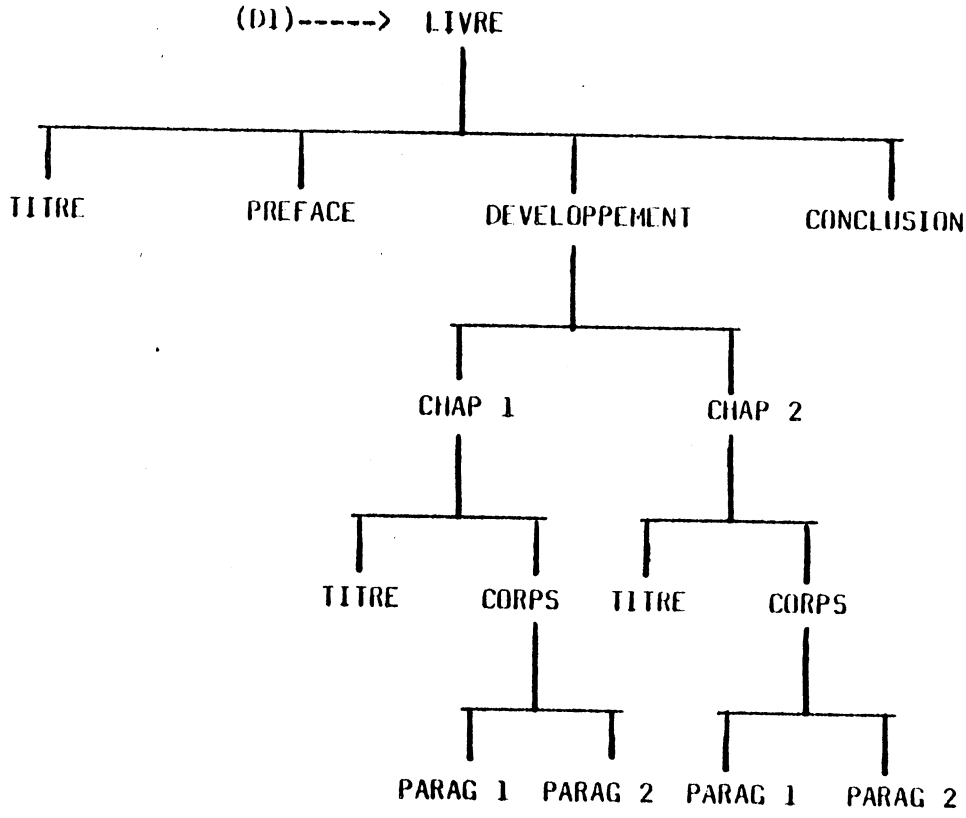
- toutes les requêtes sur les entités factuelles et les caractéristiques externes sont celles de Socrate, avec son langage de citation habituel : il est donc possible de définir des macro-instructions et des fonctions utilisant ces requêtes (langage procédural)

- le langage de citation interne est semblable au langage de citation externe de Socrate.

Malgré ces difficultés, le langage de requêtes spécifique aux documents est tout de même bien plus simple que celui de Socrate: il est non-procédural, et un grand nombre de requêtes ont un déroulement interactif.

Pour pouvoir proposer la maquette de la BDT à des utilisateurs non-informaticiens (personnel de bureau par exemple), il faudrait lui ajouter une "couche" de logiciel qui lui propose des choix simples, et des récupérations d'erreur sans messages sibyllins. Une première étape dans cette direction a été faite : elle consiste à programmer certaines "touches fonction" disponibles sur les terminaux pour qu'elles exécutent automatiquement des parcours dans l'arborescence d'un descripteur : ainsi l'utilisation de touches "fils ainé" et "frère cadet" peut permettre d'éviter le langage de citation interne.

Par exemple, supposons que la variable D1 désigne la structure du chapitre III.3.2.1, que nous rappellons ici:




La citation : LE TITRE DU CHAP 2 DE D1

pourrait être remplacée par la frappe successive de :

D1 --------> livre

fils ainé --------> titre

frère cadet --------> préface

frère cadet --------> développement

fils ainé --------> chap 1

frère cadet --------> chap 2

fils ainé --------> titre

A chaque frappe, le système afficherait le nom de la partie "pointée", avec éventuellement un menu pour les choix suivants, par exemple la liste des frères cadets et des fils pour la partie en question.

## III.5 - REPRESENTATION INTERNE DES DOCUMENTS

Chaque document est représenté par une entité qui comporte les caractéristiques suivantes :

- un nom et un numéro

- le nom du format

- un pointeur vers la racine de la structure effective (cf ci-dessous)

- une table des paramètres de ce document, faisant correspondre à chaque paramètre formel une valeur effective (TABPAR)

La structure effective d'un texte est représentée par un arbre dont chaque noeud est un "objet textuel" (OT).

Chaque OT est un enregistrement composé des éléments suivants :

- pour identifier l'OT :

  - un numéro
  - un nom, celui du noeud correspondant dans le type
  - son mode : - bloc, s'il a des fils de noms différents
    - répète, s'il a des fils "répétitifs"
    - ou feuille

- pour repérer sa position hiérarchique :

  - le numéro de son père
  - son rang parmi ses frères (à partir de 1 pour le fils aîné)

- pour l'associer à une portion de texte :

  - les numéros de la première et de la dernière unité textuelle sur lesquelles il se projette (cf figure III.1, pointeurs en pointillé).

Les unités textuelles, ou UT, sont les représentations logiques des segments élémentaires de texte et de leurs caractéristiques.

Chaque unité textuelle comporte :

- des pointeurs vers le début et la fin de la chaîne de caractères correspondante

- des pointeurs vers l'UT précédent et suivant, selon l'ordre des segments de texte dans le document

- une table des attributs locaux et paramètres (TABUT), donnant pour chacun d'eux : un code, une position dans le texte, la longueur pour les attributs locaux, et un pointeur vers l'attribut ou paramètre suivant. Le code d'un paramètre se réfère à la TABPAR du document.

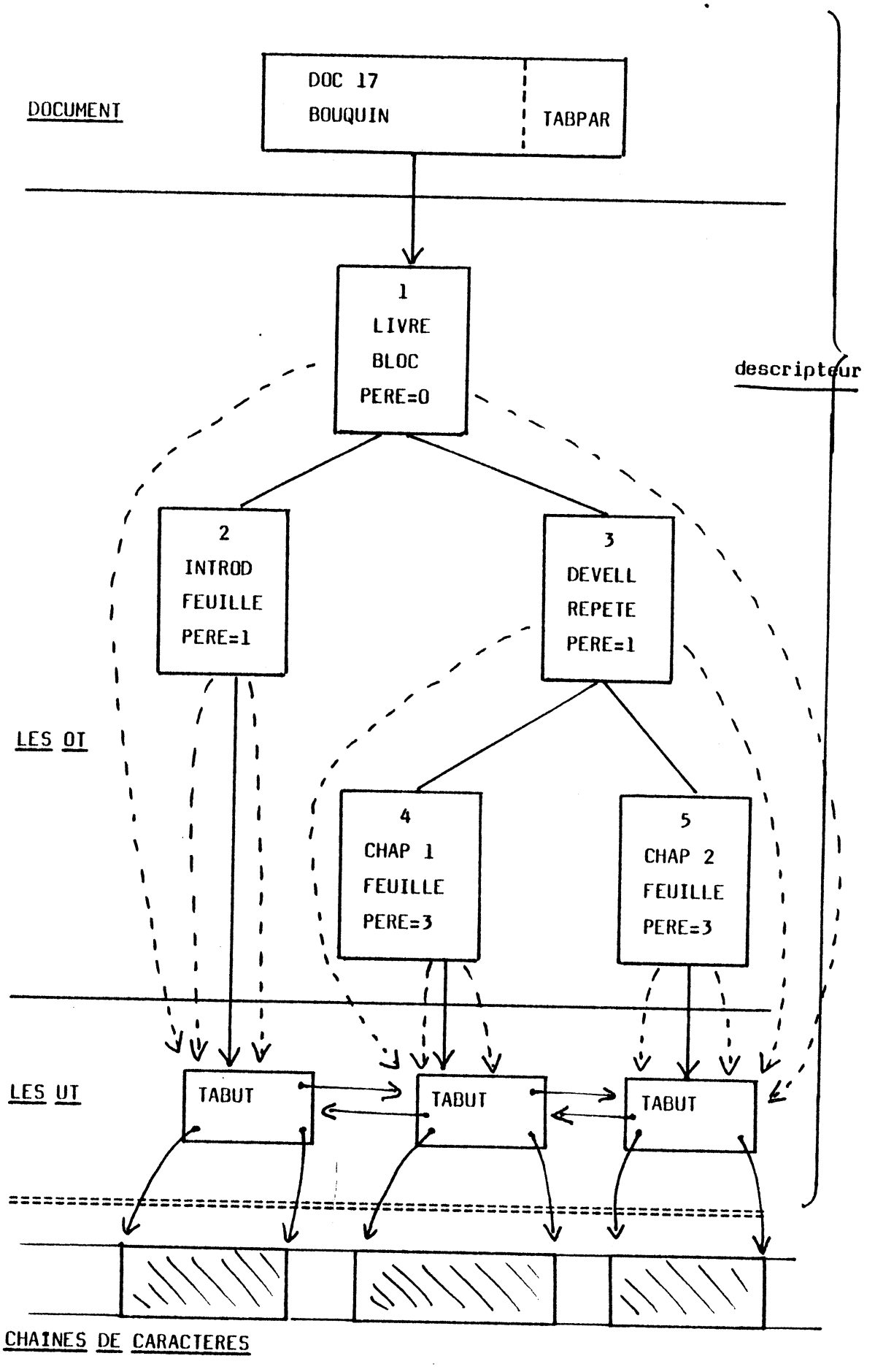

DOCUMENT
DOC 17
BOUQUIN
TABPAR
1
LIVRE
BLOC
PERE=0
descripteur
2
INTROD
FEUILLE
PERE=1
3
DEVELL
REPETE
PERE=1
LES OI
4
CHAP 1
FEUILLE
PERE=3
5
CHAP 2
FEUILLE
PERE=3
LES UI
TABUT
TABUT
TABUT
CHAINES DE CARACTERES

figure III.1

L'ensemble formé de l'entité "document", de l'arborescence d'OT, et des UT, constitue le descripteur d'un document, indépendant de la chaîne de caractères (cf figure III.1) (GREN80).

Le chaînage des UT entre eux, ainsi que les pointeurs d'un OT vers le premier et le dernier UT, sont destinés à rendre plus rapide l'accès à tout le texte concerné par un OT. En effet, il n'est plus nécessaire de parcourir l'arborescence en s'arrêtant à chaque feuille.

En contre-partie, toute modification sur cette structure effective nécessite un nombre important de mises à jour de pointeurs.

La notion d'attribut de présentation global, attaché à tout ou partie du texte, a été éliminée de la structure du document pour être envoyée vers les canevas (cf chap II.4.2).

Par contre, les attributs locaux sont attachés à chaque UT (donc à chaque OT-Feuille) et peuvent donc différer, pour une même chaîne de caractères, selon le descripteur utilisé.

## III.6 - L'ANALYSEUR STRUCTUREL DE TEXTES, OU "AST"

Lorsqu'on veut entrer un document dans la BDT, il faut construire le descripteur associé au texte, qui correspond à la hiérarchie définie dans le format. Pour construire ce descripteur, il faut effectuer un découpage du texte en segments, et créer les OT et UT (cf chap III.5.1) qui se projettent sur ces segments, ainsi que les tables d'attributs locaux et paramètres.

Le découpage du texte en segments peut être effectué de deux manières :

- interactivement : le système parcourt la hiérarchie de la structure formelle du format, et demande à l'utilisateur de lui fournir le texte de chaque feuille successivement.

- automatiquement : le texte contient des chaînes de caractères spéciales qui séparent les différents segments. Ces chaînes séparatrices, ou "constructeurs" seront spécifiées dans le "canevas d'entrée" (cf chap III.3.7), pour le format donné.

Les tables d'attributs locaux et de paramètres (tables liées aux unités textuelles), seront créées par analyse du texte, et recherche des positions dans ce texte des carazctères d'attributs locaux, et des chaînes de caractères des paramètres formels.

Donc, que le texte soit créé de manière interactive ou automatique, il faudra au moins une analyse automatique pour créer les tables. Le module chargé de cette analyse s'appelle l'Analyseur structurel de textes ou AST. C'est un processeur opérant sur un texte, les opérations étant paramétrées par le format et le canevas d'entrée.

On peut aussi envisager que l'AST soit chargé de réaliser l'indexation automatique de textes, c'est à dire le recherche de mots-clés dans le texte d'entrée.

L'environnement de l'AST est alors donné par la figure III.2 :

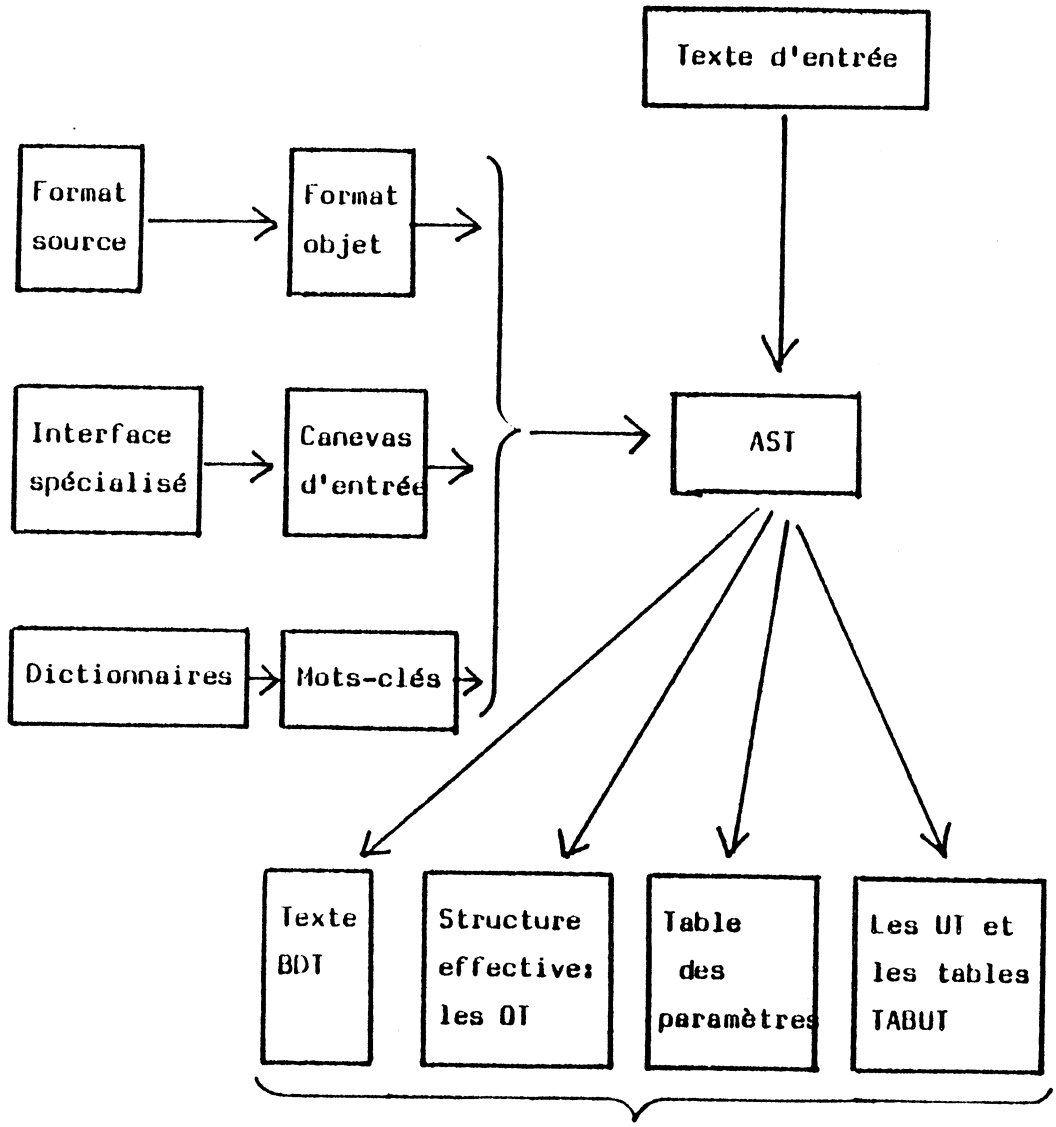



figure III.2

Remarquons d'une part que le format fourni par l'utilisateur (format source) doit être transformé par compilation en un format objet pour l'AST et d'autre part que le canevas d'entrée dépend de l'interface spécialisé d'entrée lié au périphérique d'entrée que l'on utilise.

La construction d'un document virtuel, à l'aide de l'AST, et son insertion dans le schéma conceptuel et le schéma documentaire, sont données dans la figure III.3 :

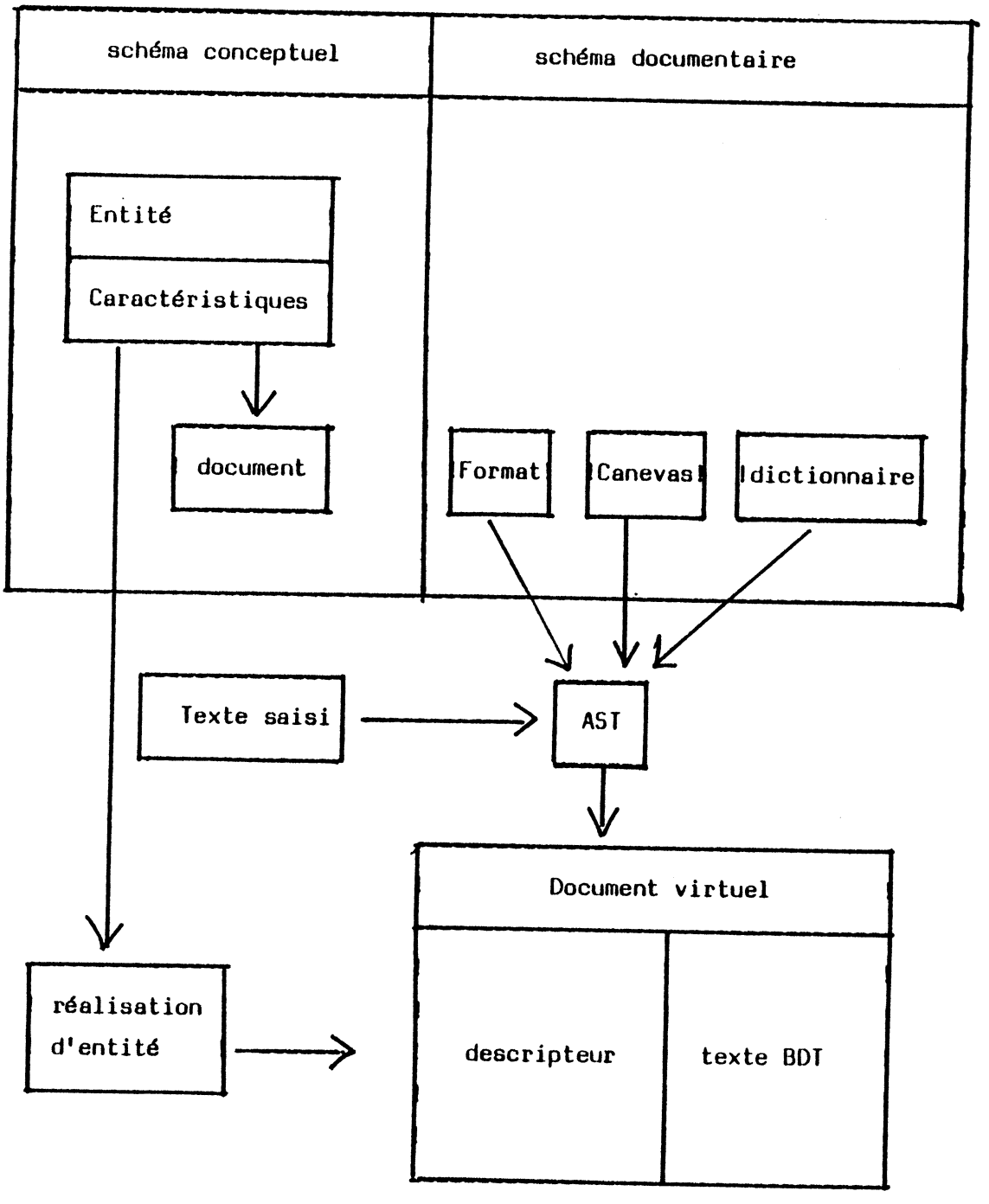



figure III.3

L'AST est donc un automate qui reconnait des sous-chaînes données dans une chaîne en entrée et construit des tables (le descripteur) en sortie. Son fonctionnement est détaillé dans (CETI80) et (MINO80). Dans la maquette Solar, l'AST a été réalisé en Socrate, en utilisant certains sous-programmes en Assembleur pour la recherche de sous-chaînes dans une chaîne de caractères. Il est clair que ce n'était qu'une solution provisoire, destinée à tester les algorithmes de fonctionnement, mais qu'un prototype industriel aurait eu intérêt à utiliser un AST cablé.

III.7 - BILAN

La maquette de BDT a été entièrement programmée et a montré que tous les objectifs énoncés au chap II pouvaient être atteints. Le SGBD et le langage Socrate ont permis d'adjoindre le traitement spécifiquement textuel à une base de données factuelles, réalisant ainsi la première étape vers une base de données généralisée, qui traiterait aussi les données graphiques, vocales....

Parmi les avantages de Socrate, l'un des plus utiles a été la gestion d'une mémoire virtuelle tant pour les données que pour les programmes. Aucun problème de place mémoire ne s'est donc posé, et les délais de réponse étaient généralement acceptables. L'aspect multi-utilisateurs de la BDT n'a pas pu être testé sérieusement en raison de la version de Socrate dont nous disposions.

Les principales difficultés que nous avons rencontrées sont :

- la lourdeur du traitement des fonctions récursives en Socrate, pourtant celles-ci sont indispensables pour les structures arborescentes ( un exemple figure à l'annexe 1).

- Les problèmes de sécurité : reprise sur pannes, et accès concurrents, reposaient uniquement sur Socrate et n'ont pas toujours été résolus.

Au passif de la maquette, on peut noter qu'il manque des fonctionnalités simples de traitement de texte en entrée : la saisie des textes se faisant sous l'éditeur de programmes, tout à fait insuffisant pour cette tâche. Les ajouts au schéma conceptuel (définition de nouvelles entités) se faisant sous Socrate posent un gros problème : toute erreur de syntaxe dans la partie ajoutée entraine la destruction complète de la structure. Celle-ci peut être sauvegardée, mais les programmes doivent tous être recompilés.

Enfin, aucune étude de contrôle de droits d'accès propre à la BDT n'a été faite, seul l'accès à Socrate à travers un mot de passe permet un contrôle.

La maquette de BDT a donc permis d'envisager la réalisation de bases de données textuelles plus ambitieuses :

- réellement multiutilisateurs, liées à un réseau

- avec des terminaux traitement de texte en entrée et en sortie

- en utilisant un analyseur de texte rapide plutot que notre AST en Socrate

- munies d'interfaces utilisateur de différents niveaux : langage procédural permettant de bâtir des applications spécifiques, interfaces graphiques inspirés de QBE, ou encore interfaces interactifs d'utilisation très facile.

# CHAPITRE IV

# LE PROJET MIDOC

## IV.1 - OBJECTIFS DE MIDOC (MIcro-ordinateur-DOCument)

La diffusion des micro-ordinateurs se développe actuellement de manière très importante, tant pour l'utilisation personnelle que dans des environnements de bureaux et de laboratoires. On estime que d'ici 1984 plus d'un million de micro-ordinateurs seront installés en Europe, dont les 2/3 seront utilisés pour la gestion des PME/PMI. Leur technologie évolue très rapidement : du 8 bits, encore le plus répandu, au 16 bits et bientôt au 32 bits, du micro-ordinateur isolé au multimicro et aux réseaux, des mémoires centrales de 32KO aux 64KO et 128KO, des écrans de 40 caractères par ligne aux écrans pleine page et bientôt aux écrans "multifenêtres" des stations de travail de l'avenir (bureauviseur (KAYA81), machine Lillith (WIRT81), "Lisa"... ) des mémoires mortes sur cassettes aux disquettes souples, puis aux disques de capacités toujours croissantes.

Parallèlement, les logiciels proposés se perfectionnent : systèmes, langages et progiciels sans cesse plus performants. Néanmoins, les systèmes de gestion de bases de données sur micro ordinateurs commercialisés sont encore peu nombreux: RTFile (NATA81), dBASE II (EDEL81), MDBS (MDBS80), par exemple.

Assez souvent les systèmes vendus sous ce nom ne sont en réalité que des systèmes de gestion de fichiers en séquentiel indexé (MERR83).

Même lorsqu'il s'agit de véritables SGBD, il leur manque souvent les contrôles d'accès et d'intégrité, ainsi que les mécanismes pour la sécurité en cas de pannes que l'on est habitué à trouver sur les SGBD destinés à de plus grosses machines.

Plusieurs centres universitaires ont mis au point de véritables SGBD sur micro-ordinateurs, qui commencent seulement à se répandre: (ANDE78), (BOUC83), (MERR83), (MARY83), (NGUY82), (REBS83).

Ces systèmes paraissent tous être basés sur le modèle relationnel. En effet celui-ci se présente actuellement comme le modèle le plus facile à utiliser, tant du point de vue de la définition du schéma de données, que du point de vue des langages de manipulation des données. Les micro-ordinateurs apparus sur le marché bien après les minis et les gros, ont pu "sauter" l'étape des SGBD hiérarchiques et réseaux largement répandus sur ceux-ci, et passer directement de la gestion de fichiers aux SGBD relationnels.

Cependant, à notre connaissance, aucun de ces SGBD ne traite spécifiquement les données textuelles. Seuls des travaux de recherche vont actuellement dans ce sens (MIRA83). Nous avons donc tenté de réaliser un système de gestion de base de données adapté aux données textuelles, qui soit implanté sur micro-ordinateur.

Plus précisément, les objectifs du projet MIDOC étaient les suivants :

- Réaliser un système qui ne soit plus une maquette, mais qui soit réellement opérationnel

- Fournir un interface utilisateur interactif qui soit agréable et facile à employer tant par des informaticiens que par des personnes non spécialisées("user - friendly")

- Disposer d'un mécanisme efficace de traitement de texte pour la saisie et l'impression

- Inclure dans le système des contrôles d'accès concernant la confidentialité des documents

- Mettre en oeuvre des mécanismes de sécurité pour éviter les pertes d'informations en cas de panne ou de fausse manoeuvre

- Rendre ce système le plus portable possible sur différentes machines.

Nous étions soumis à des contraintes :

- de temps : réaliser ce système en un temps raisonnable, à deux chercheurs et avec l'aide de quelques stagiaires occasionnels.
- de matériel : au moment où ce projet a commencé les machines 16 bits étaient encore peu disponibles en France.
Nous avons donc décidé de travailler avec une machine 8 bits , de 64 KO de mémoire centrale (cf chap IV.5)

- Par conséquent, les problèmes de place en mémoire centrale se posaient de manière aigüe.

Ces contraintes nous interdisaient de transposer purement et simplement la maquette Solar en la programmant sur un micro-ordinateur. Dès le départ, nous avons décidé de deux limitations, tout au moins pour la première phase du projet :

- ne pas chercher à faire un système multi-poste

- laisser de côté le traitement des données classiques (numériques et factuelles) pour nous préoccuper en premier lieu des documents.

Nous avons apporté certaines modifications, dans le sens tantôt de la simplification, tantôt d'une utilisation plus commode, au modèle de document décrit au chapitre II et mis en oeuvre dans la maquette Solar (cf chap IV.2).

## IV.2 - LE MODELE DE DOCUMENT

Puisque nous avons choisi de ne pas traiter d'entités non documentaires, dans la première phase de ce projet, le schéma conceptuel se réduit à la modélisation des documents. Nous gardons les concepts de classes de documents, de caractéristiques externes, de structure formelle définie dans un "type", de paramètres formels, d'attributs de présentation, et de mots-clés, mais nous remanions leurs liaisons.

### IV.2.1 - CARACTERISTIQUES EXTERNES ET TYPES

En premier lieu, nous faisons coïncider les notions de "classe" et de "type". En d'autres termes, une classe de documents est simplement définie par leur appartenance au même type. Les caractéristiques externes associées à tout document seront de deux sortes :

- les unes seront communes à tous les documents, quelque soit leur type (caractéristiques externes générales)

- les autres seront précisées dans le type (caractéristiques externes particulières).

Les caractéristiques externes générales seront les suivantes:

- titre

- auteur

- date

- numéro de référence.

En effet, la recherche d'un document sur caractéristiques externes, dans la maquette Solar, dépend de la connaissance de sa classe : ainsi la requête

I tout Roman ayant auteur = 'V. HUGO' ?

est possible si la classe ROMAN a été définie :

```
document ROMAN
début
     AUTEUR : alpha;
       .
       .
       .
     TYPE : LIVRE;
fin
```

Mais on peut fort bien ne pas connaitre la "classe" de chacun des livres de V. HUGO (ROMAN, ou HISTOIRE ou SOUVENIRS...) et néanmoins souhaiter tous les connaître. Plus prosaïquement, dans un environnement de bureau, des documents seront rangés dans les classes "CIRCULAIRE", "RAPPORT", "NOTE", etc et l'on peut souhaiter retrouver rapidement un document dont on connait l'auteur, la date ou le n° de référence mais non la classe. Notons que deux documents de la même classe ne peuvent pas avoir le même titre.

Par contre d'autres caractéristiques externes sont propres à un ou plusieurs types. Ce seront, par exemple:.

- Service émetteur
- date de dernière mise à jour
- prix de vente
- langue

Ce seront les caractéristiques externes "formelles" propres au type.

Le type se composera donc de deux parties :

- une liste de caractéristiques externes particulières "formelles"
- une définition de structure formelle.

Une réalisation de document comportera alors :

- une structure effective
- un texte associé (chaîne de caractères)
- une liste de caractéristiques externes effectives

Exemple : soit le type LIVRE défini ainsi :

```
type LIVRE
Structure
      PREFACE
      CORPS
         repete CHAPITRES
      %POSTFACE                    ('%' dénote une partie facultative)
Car-ext
      LANGUE : alpha
      NB-TOMES : entier
      EDITEUR : alpha
Fin-type
```

Dans le classe des LIVRES on pourrait trouver les deux réalisations suivantes (figure IV.1):

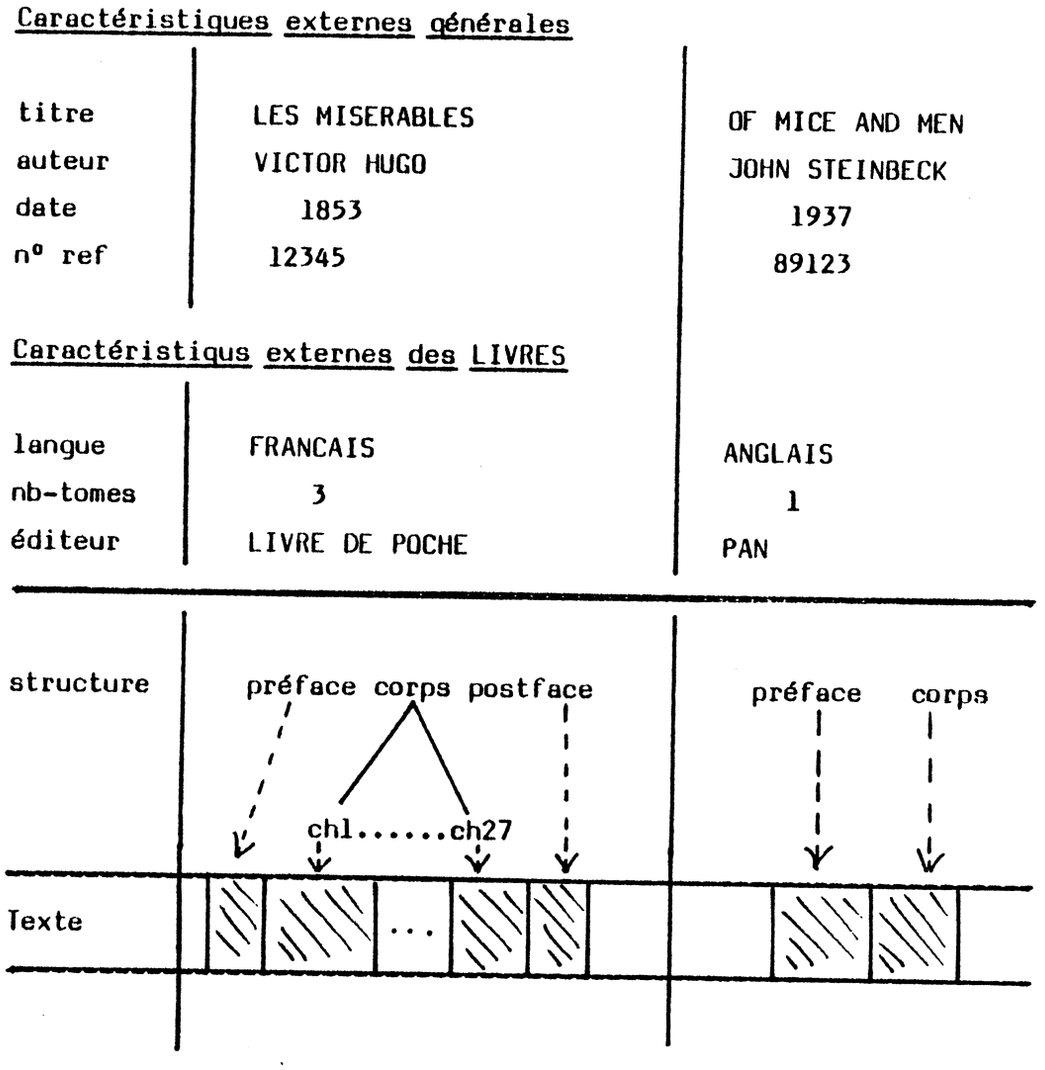



figure IV.1

## IV.2.2 - LES ATTRIBUTS DE PRESENTATION

Pour les attributs locaux que nous traitons

- soulignement

- caractères gras

- indication de changement de police de caractère

nous adoptons la solution "intermediaire" envisagée à la fin du chapitre II.3.3.: ces attributs sont saisis par le processeur d'entrée, et sont rangés dans le texte, sous une forme normalisée indépendante du terminal d'entrée. A la sortie, ils sont à nouveau traduits par le processeur de sortie, dans une forme qui convient au terminal utilisé (écran ou imprimante). Le schéma est donc celui de la figure IV.2:

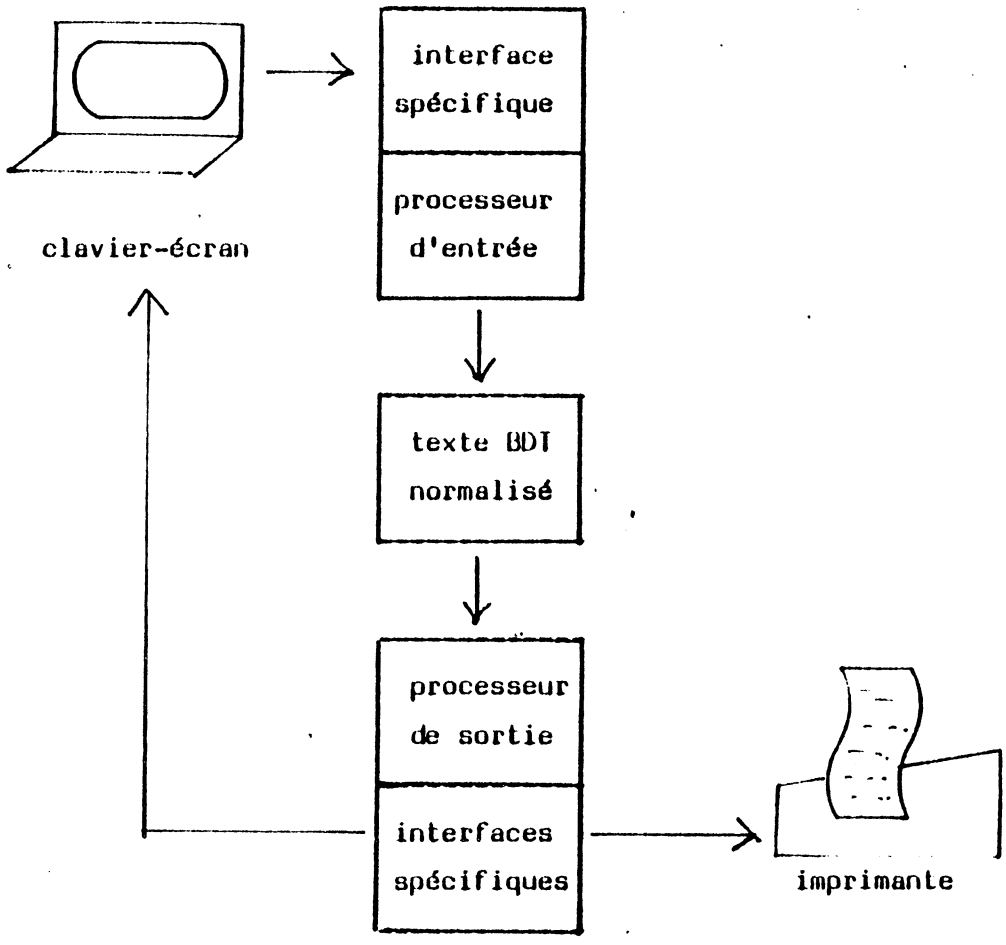



figure IV.2

Quant aux attributs globaux que nous traitons :

- marges et indentation

- marges haut et bas

- numérotations de page

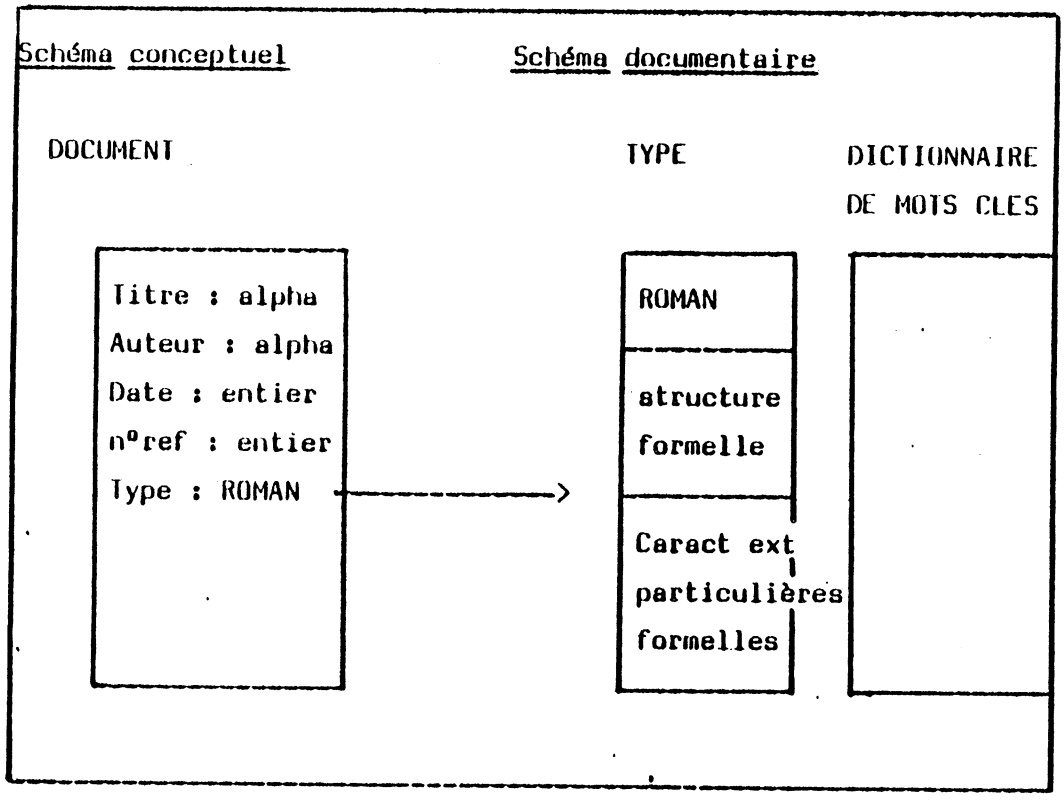
- changements de page

- entêtes et bas de page

- interlignes

ils sont associés soit à l'ensemble du texte, soit aux feuilles (éléments terminaux) de la structure. Les marges et l'indentation (marge paragraphe) sont déterminés dès la saisie, mais sont toujours modifiables, soit localement, soit sur tout un segment de texte (texte qui correspond à un OT). Les autres attributs destinés plus particulièrement à l'impression des documents sur papier, sont déterminés au moyen du processeur d'impression. Ils sont stockés avec le texte, mais peuvent aussi être modifiés sans difficulté. C'est le module de "traitement de texte" qui est chargé de gérer les attributs, d'une manière beaucoup plus souple que par des "canevas" associés aux types. L'implémentation détaillée de ces attributs figure au chap IV.3.5.

Finalement, nous pouvons résumer notre modèle dans le schéma ci-dessous (figure IV.3):




Réalisations

| Caractéristiques externes effectives | Mots clés |
|---|---|
| Auteur, Créateur, Catégorie => (droits d'accès) | |
| Structure effective | |
| attributs globaux | TEXTE |

figure IV.3

## IV.3. - FONCTIONNALITES

### IV.3.1 - ACTIONS SUR UN DOCUMENT

- La création d'un document consiste à demander le type souhaité, puis les valeurs des caractéristiques externes générales et particulières (toutes peuvent rester indéfinies sauf le titre, obligatoire pour l'accès ultérieur au document).

L'étape suivante consiste à créer une structure effective d'objets textuels (OT), en se guidant sur la structure formelle du type. Pour cela, le système parcourt la structure formelle, et crée un OT pour chaque noeud obligatoire de celui-ci. Pour chaque partie optionnelle, il demande si l'usager souhaite la créer : si la réponse est négative, il ne créera pas le sous-arbre correspondant, mais crée néanmoins l'OT lui-même, afin de ne pas modifier la liste des alternants à son niveau. Pour les parties répétitives, le système demande le numéro d'ordre souhaité, et crée tous les OT intermédiaires nécessaires pour sauvegarder la numérotation consécutive à partir de 1.

Là encore, il ne crée pas les sous arbres des parties non souhaitées. On dira que ces parties sont vides.

Si une partie n'a pas de descendants (les descendants possibles sont tous optionnels, et l'on ne souhaite pas les créer, ou bien ils sont itératifs en nombre nul) nous avons choisi de la traiter comme une feuille de la structure effective, même si elle ne correspond pas à une feuille de la structure formelle. Cela permet en effet d'obtenir une grande souplesse dans l'emploi de la structure.

Exemple:

Soit la structure formelle suivante:

```
LIVRE =DEBUT
          INTRODUCTION
          DEVELOPPEMENT
          CONCLUSION
      FIN

DEVELOPPEMENT =REPETE CHAPITRE

CHAPITRE =REPETE SECTION

SECTION =REPETE SOUSSECTION

SOUSSECTION =REPETE PARAGRAPHE

CONCLUSION =DEBUT
               %POSTFACE
               %ANNEXE
            FIN
```

Les feuilles de la structure formelle sont: INTRODUCTION, PARAGRAPHE, POSTFACE, ET ANNEXE. Néanmoins, d'autres élements pourront jouer le rôle d'éléménts terminaux lors de la création d'une structure effective, si l'on décide de ne pas leur créer de descendants. Ainsi, on pourrait créér un document du type LIVRE avec la structure effective suivante (donnée sous forme d'une "table des matières"):

Introduction
Chapitre 1
    Section 1
        Soussection 1
        Soussection 2
            Paragraphe 1
                .
                .
            Paragraphe 10
    Section 2
Chapitre 2
    Section 1
    Section 2
        Soussection 1
            .
            .
        Soussection 6
    Section 3
Chapitre 3
Conclusion

Dans cet exemple, les différents chapitres, sections, etc, ont une structure plus ou moins finement détaillée, alors que la conclusion est traitée comme un tout.

Ainsi, le fait que la structure formelle soit fixée a priori n'entraine pas l'obligation de calquer chaque partie complètement sur celle-ci. La structure formelle est une structure "maximale", il est possible de l'utiliser partiellement, et de manière variable.

Lorsque le parcours de l'arbre parvient jusqu'aux feuilles, on pourra

- soit saisir le segment de texte correspondant (nommé texte-feuille) au moyen du processeur d'entrée,

- soit le laisser (provisoirement) vide,

- soit copier une feuille, ou même un sous-arbre, à condition que la partie à copier appartienne à un document de même type, et que la racine soit un noeud de même nom (sous-arbres de structure identique).

Nous avions aussi envisagé la possibilité de partage d'une feuille entre deux ou plusieurs documents. Le partage permettrait en effet d'éviter la recopie de morceaux de texte destinés à être réutilisés. Il serait conforme à l'"esprit" d'une base de données: non redondance des informations.

Cependant, le problème des modifications se pose alors de facon aigüe: si un texte partagé est modifié pour un document, tous les autres documents qui s'y réfèrent seront modifiés, quelque soient les droits d'accès de la personne qui effectue les modifications sur les autres documents.

Par ailleurs, les exemples que nous avons pu trouver où le partage serait intéressant portent soit sur

- des morceaux de texte très courts: clausier, formules de politesse...qu'il est sans inconvénient de recopier

- des documents en très petit nombre: plusieurs manuels pour un même matériel pourraient partager des chapitres.

Nous avons donc finalement choisi de ne pas prévoir de mécanisme de partage.

L'opération de création d'un document est donc entièrement guidée par la structure du type, mais elle se fait interactivement. En particulier, la saisie ne peut se faire que sur un texte-feuille à la fois.

En cela MIDOC diffère de la maquette Solar qui pouvait analyser un texte d'entrée, et le découper en segments délimités par des caractères spéciaux, au moyen de l'AST.

La mise à jour d'un document est une opération semblable. Le système parcourt l'arborescence de la structure effective, et demande, pour chaque OT, si on veut le supprimer, le modifier, le saisir, ou le laisser inchangé. Là encore, la saisie effective, ainsi que la mise à jour du texte lui-même ne pourra se faire qu'au niveau des textes-feuilles.

La suppression d'un document consiste simplement à supprimer, récursivement, tous les OT de sa structure effective. Les textes-feuilles correspondants seront également supprimés.

Enfin, la fonction de visualisation d'un document consiste à traiter successivement tous les textes-feuilles concernés, en parcourant l'arborescence de la structure effective. On pourra s'interesser soit à un document complet, soit à une partie (sous-arbre). On pourra soit afficher ce texte à l'écran, page par page, soit le faire imprimer. C'est le module de traitement de texte qui est mis en oeuvre pour l'affichage et l'impression, comme pour la saisie et la mise à jour.

### IV.3.2 - RECHERCHE DE DOCUMENTS SUR CARACTERISTIQUES EXTERNES

Deux mécanismes entièrement distincts permettent la recherche de documents:

- la recherche documentaire, sur mots-clés, que nous décrivons ci-dessous, au chapitre IV.3.3.2

et - la recherche sur caractéristiques externes et types.

Pour celles-ci, dans un premier temps, le système demandera à l'usager de lui indiquer les valeurs des caractéristiques externes générales qu'il connait (auteur, titre, date, numéro de référence). Si les renseignements donnés conduisent à un seul document, le système indique le titre et le type. Si plusieurs peuvent convenir, le système demande le type. Il peut, sur demande, fournir la liste des types afin d'aider l'utilisateur.

Une fois le type connu, le système demande à l'usager de lui indiquer les valeurs des caractéristiques externes particulières qu'il connait.

Finalement, on doit donc aboutir à la connaissance du titre et du type d'un document,et c'est ce couple qui va servir au système comme clé d'accès. Rappelons qu'à l'intérieur d'une même classe de documents, deux documents ne peuvent pas porter le même titre.

## IV.3.3 INDEXATION ET RECHERCHE DOCUMENTAIRE

### IV.3.3.1 - LES MOTS-CLES

Nous reprenons le principe d'indexation par mots-clés, mais estimons que dans le contexte d'une application "micro-ordinateur", la nécessité de disposer de plusieurs lexiques spécialisés par centre d'interêt ne s'impose pas. Nous aurons donc un seul lexique. Pour éviter que sa taille ne devienne trop importante par suite de surcharge en formes très voisines ou quasi-synonymes, nous proposons simplement d'inciter le créateur d'un document à utiliser des mots-clés déja présents dans le lexique. En effet, l'indexation, comme toutes les autres fonctions de MIDOC, est interactive. Cela nous permet de signaler à l'utilisateur que le mot-clé qu'il envisage est ou n'est pas déjà dans le lexique, et dans ce dernier cas lui proposer :

- soit des mots morphologiquement très voisins du lexique

- soit la liste complète des mots du lexique afin qu'il modifie éventuellement son choix.

Nous maintenons le principe d'indexation dynamique, cela peut permettre de modifier les critères de recherche d'un document, en fonction, par exemple, d'un lexique qui s'est enrichi.

### IV.3.3.2 - LA RECHERCHE DOCUMENTAIRE

Dans le système MIDOC, nous avons cherché à réaliser des fonctions de recherche documentaire relativement simples, et faciles à utiliser. Nous nous n'avons donc pas prévu de langage d'interrogation, mais une recherche interactive.

L'usager peut à tout moment consulter le lexique des mots-clés ("dictionnaire") du système. La recherche se fait alors au moyen du dialogue suivant :

L'usager frappe une liste de mots-clés, le système lui retourne le nombre de documents indexés sur l'un au moins de ces mots. Si ce nombre est trop important, il peut affiner sa recherche en frappant une nouvelle liste (éventuellement vide) de mots-clés. Il obtient alors le nombre de documents, parmi ceux qui ont déjà été sélectionnés, qui sont indexés sur tous les mots de la deuxième liste. Dans une troisième étape possible, il indique une liste de mots-clés qu'il souhaite exclure de la recherche. Il a ensuite accès au titre et au type de tous les documents, à moins qu'il ne trouve la liste encore trop longue. Dans ce cas, il reprendra à la deuxième ou à la troisième étape. Si la liste devient trop courte (ou même vide), il reprendra au début.

Soient L1 = (m1,.....,mi)

L2 = (n1,.....,nj)

L3 = (p1,.....,pk)

Les listes de mots-clés fournies dans les trois étapes décrites ci-dessus. La recherche documentaire s'effectuera selon l'expression logique suivante :

(m1 ou ... mi) et (n1 et ... nj) et ((non p1) et ... (non pk))

L'articulation entre les trois phases peut s'exprimer ainsi :

```
fonction NBDOC (OPER : opérateur booleen;
                                var LISTEDOC : liste de documents) : entier;
(* cette fonction lit une liste de mots, puis modifie LISTEDOC en
l'élargissant (cas OPER = "ou") ou en le restreignant (OPER = 'et' et
'sauf') aux documents concernés. La fonction délivre le nombre de
documents dans la nouvelle LISTEDOC *)

fonction N_TROP_GRAND (N : entier) : booléen;
(* cette fonction affiche la valeur de N, et demande à l'utilisateur de
lui indiquer s'il juge cette valeur trop élevée; si la réponse est
positive, la fonction délivre la valeur "vrai" *)
```

```
fonction N_SATISFAISANT (N : entier) ; booleen;
(* analogue à la précédente, délivre "vrai" si l'utilisateur juge la
valeur de N satisfaisante *)


procedure RECHDOC (var LISTEDOC : liste de documents);
var N : entier;
début
     LISTEDOC := vide;
     répéter
          N := NBDOC ('ou', LISTEDOC);
          tantque N_TROP_ELEVE (N) faire
               début
                    N := NBDOC ('et', LISTEDOC);
                    si N_TROP_ELEVE (N) alors
                    N := NBDOC ('sauf', LISTEDOC);
               fin
     jusque N_SATISFAISANT (N)
fin;
```

Le dialogue que nous venons de décrire fournit une expression logique sur les mots-clés qui n'est pas générale. Il présente cependant deux avantages : il est plus facile à utiliser que la formulation logique, souvent proposée dans les systèmes documentaires actuels et il correspond à une démarche de l'esprit assez naturelle : indication d'un contexte large, puis restrictions successives, ou ré-élargissements si nécessaire.

IV.3.4 - LES PARAMETRES

Dans la maquette Solar, la liste de paramètres formels faisait partie du format, tous les documents de même type avaient dont la même liste de paramètres formels. Il nous semble que cette association est trop rigide, et que c'est au contraire à chaque réalisation d'un document qu'il convient d'associer, s'il y a lieu, une liste de paramètres formels, de la même façon que les valeurs de caractéristiques externes.

Nous nous limitons au traitement des paramètres formels "chaîne de caractères". En effet, les autres types de paramètres (cf chap II.3.5) sont des manières indirectes de spécifier des chaînes de caractères, et nous doutons que leur emploi puisse être aisé.

L'affectation de valeurs aux chaînes de caractères pourra se faire de deux manières :

- soit interactivement

- soit à partir de listes de valeurs effectives, stockées dans la base (opérations genre "mailing").

On disposera pour cela d'un type prédéfini de document: "LISTE-PARAM", de structure très simple : Repete PARAM. Chaque élément de ce document aura la forme

NOM-PARAM-FORMEL = CHAINE

Supposons que nous ayons le document suivant :

Monsieur NOM
ADRESSE
VILLE

Cher Monsieur NOM,

Nous vous prions de trouver ci-joint un exemplaire de notre catalogue pour l'année DATE, et espérons qu'il vous donnera satisfaction.

Dans l'attente de votre prochaine commande, nous vous prions d'agréer nos meilleurs sentiments.....

accompagné de la liste de paramètres formels :

NOM/ADRESSE/VILLE/DATE

Au moment de la sortie de multiples exemplaires de cette lettre, on pourra décider d'affecter DATE interactivement à la même valeur pour tous les exemplaires ( par exemple DATE = 1983), puis de prendre les autres valeurs dans une LISTE-PARAM qui pourrait commencer ainsi :

NOM = DUPONT / ADRESSE = 15 rue du Vercors / VILLE = GRENOBLE /
NOM = DURAND / ADRESSE = 1 ave Lafayette / VILLE = MEYLAN / ....

Ainsi, la première lettre sortie serait:

Monsieur DUPONT
15 rue du Vercors
GRENOBLE

Cher Monsieur DUPONT,

Nous vous prions de trouver ci-joint un exemplaire de notre catalogue pour l'année 1983 et espérons qu'il vous donnera satisfaction.

Dans l'attente de votre prochaine commande, nous vous prions d'agréer nos meilleurs sentiments.....

## IV.3.5 - LE TRAITEMENT DE TEXTES

### IV.3.5.1 GENERALITES

Le module de traitement de textes, base d'un système de traitement de document, comporte de si nombreuses fonctions que le programme est relativement volumineux. Par ailleurs, lorsque ce programme est appellé, il travaille sur un texte qui occupe une grande partie de la mémoire centrale. Nous avons donc partagé le module de traitement de textes en deux parties à la fois physiquement et fonctionnellement distinctes, le "processeur d'édition" chargé de la saisie, de la mise à jour, et de l'impression avec une présentation simple et le "processeur de formattage chargé de l'impression avec des attributs de présentation plus sophistiqués. Chacun de ces processeurs a été le plus possible segmenté, afin de laisser le maximum de place disponible en mémoire pour le texte.

La chaîne de caractères du texte est rangée dans une zone dite "buffer variable" qui peut occuper toute la place laissée disponible en mémoire par les programmes. Lorsqu'une chaîne de caractères doit être provisoirement sauvegardée (après insertion ou destruction, ou bien les entêtes et bas de pages), elle est rangée à la fin de ce buffer :

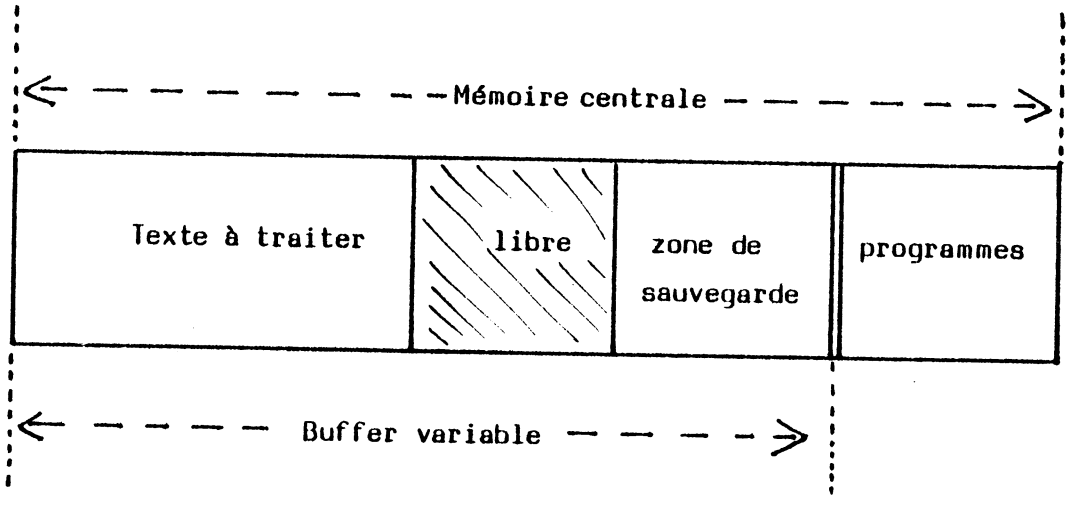



fig IV-4

Pour une utilisation agréable de MIDOC, il faut disposer :

- d'un écran qui possède au moins un attribut video, et de préférence les trois attributs video suivants : soulignement, inverse video (que nous utilisons pour figurer les caractères "gras" de l'imprimante) et intensité réduite (utilisé pour figurer le passage à une autre police de caractères : l'imprimante s'arrête et attend le changement de marguerite, par exemple).

Le clignotement peut remplacer l'un de ces trois attributs video, mais s'avère fatiguant pour l'utilisateur. Il faut de plus que la mise en fonction et hors fonction de ces attributs ne soit déclanchée que par l'envoi de séquences particulières de caractères, et n'occupe pas une place à l'écran. C'est l'interface d'entrée qui est chargé de transformer les caractères de contrôle des attributs video d'un terminal donné en caractères normalisés dans le texte interne à la base de données.

- d'un clavier-écran et d'une imprimante adaptés à la langue utilisée. Les claviers AZERTY posèdent déjà les lettres accentuées les plus fréquemment utilisées en français. Les autres (circonflexe, trema et tilda le plus souvent, mais la liste peut être spécifiée dans l'interface d'entrée) sont obtenues par surimpression lors de la sortie sur papier. Comme la plupart des écrans ne permettent pas de faire apparaître les caractères surimprimés, nous avons choisi d'afficher ces lettres en "souligné", et de permettre la visualisation provisoire de l'accent à la place de la lettre.

Contrairement à de nombreux systèmes de traitement de textes existants, nous avons choisi d'afficher à l'écran une image aussi fidèle que possible du document physique. En particulier, aucun "caractère de contrôle" ni "commande de présentation n'apparaîtra dans le texte. La position du curseur indique l'endroit dans le texte ou s'appliqueront toutes les commandes (il n'y a pas d'adressage par numéro de ligne).

### IV.3.5.2 LE PROCESSEUR D'EDITION

Ce processeur permet en premier lieu la saisie et la mise à jour d'un texte-feuille. Il est basé sur l'éditeur de programmes du système UCSD, dont il reprend le principe des menus arborescents, et la plupart des fonctions. Nous avons éliminé quelques rares fonctions qui nous paraissaient d'un emploi trop peu commode, ou inutile, et avons ajouté un certain nombre de fonctions supplémentaires. De plus, nous avons ajouté toutes les fonctions concernant l'impression, qui n'existaient pas au niveau de l'éditeur de programmes.

La saisie se fait normalement "au kilomètre", c'est-à-dire que les changements de ligne sont placés automatiquement, selon les marges choisies. Les changements de paragraphe, avec une indentation choisie, s'obtiennent en frappant deux "retour-chariots". Si l'on souhaite changer les valeurs choisies pour les marges gauche, droite et paragraphe, le texte peut être rejustifié aux nouvelles marges très facilement, localement ou partout.

La justification à droite permet d'obtenir des lignes qui se terminent exactement sur la marge droite (sans espaces), au moyen d'insertion d'espaces entre les mots de chaque ligne. Cette insertion se fait en alternance à partir de la gauche et de la droite, afin d'éviter la formation de "rivières" de blancs (ACHU81).

Pour éviter d'avoir à insérer un trop grand nombre d'espaces, un algorithme de coupure automatique a été mis en oeuvre. Basé sur les règles classiques de coupures en syllabes de la langue française (GREV75), cet algorithme donne d'excellents résultats, sans demander un temps d'exécution exagéré (cf annexe 5).

La justification, avec coupure de mots, est mise en oeuvre automatiquement à chaque insertion de texte, et sur demande en cas de changement de marges.

Parmi les fonctions de modification de texte, on trouve :

- le déplacement du curseur dans le texte : à l'aide des quatre flèches, vers le début d'une ligne, vers une position de tabulation, vers le début ou la fin d'un texte, vers un marqueur (repère nommé), vers un

signe de ponctuation, vers un espace, vers une chaîne de caractères ou un mot spécifié...;

- le réaffichage du texte avec la ligne courante au centre de l'écran;

- l'insertion de chaînes de caractères avec rejustification automatique;

- le remplacement de caractères;

- la destruction de caractères, de mots, de phrases, de tabulations, d'écrans...;

- la copie d'une partie du texte à un autre endroit, avec ou sans destruction de la partie émettrice;

- le remplacement d'une, plusieurs, ou toutes les occurrences d'une chaîne de caractères (ou d'un mot) par une autre (remplacement automatique ou contrôlé);

- le déplacement horizontal de lignes ou de segments de texte, vers la gauche ou la droite, ou leur centrage;

- la modification de nombreux paramètres : marges et indentation, positions de tabulation, options concernant la frappe au kilomètre ou non, la recherche par chaînes de caractères ou par mots, etc..

- le positionnement des marqueurs, nommés librement, invisibles à l'écran.

Certaines commandes peuvent être exécutées soit un nombre déterminé de fois, soit jusqu'à ce que la fin du document soit atteinte. Il suffit pour cela de frapper un nombre ou un '/' avant le caractère identifiant la commande.

Donnons un exemple d'utilisation du processeur d'édition :

- lorsque le menu principal s'affiche sur la première ligne de l'écran, un certain nombre d'options sont offertes, dont l'option S(aute. Si l'on frappe le caractère 'S' (sans écho sur l'écran), le menu principal est remplacé par le menu suivant ,

>Saute : D(ebut F(in M(arqueur < séparateur><X><esc>

sans que le texte affiché sur le reste de l'écran soit affecté.

La frappe de l'un des caractères proposés permet de repositionner le curseur dans le document :

-'D' ou 'F' : début ou fin du document

-'M' : provoque l'affichage du menu

Saute à quel marqueur ?

(le marqueur doit avoir été préalablement positionné et nommé)

-<séparateur> : saut vers un séparateur (signe de ponctuation ou espace), en amont ou en aval, selon la direction choisie (premier caractère du menu, modifiable à volonté)

-'X' : provoque l'affichage d'un écran d'aide détaillant les différentes options de la commande Saute et en particulier la liste des séparateurs; après l'avoir lu on frappera <esc> pour revenir au niveau Saute.

-<esc> : retour au menu principal, sans déplacement du curseur.

Le graphe de l'automate des affichages de menus (états), en fonction des caractères que l'on frappe (entrées) est donc le suivant (figure IV.5):

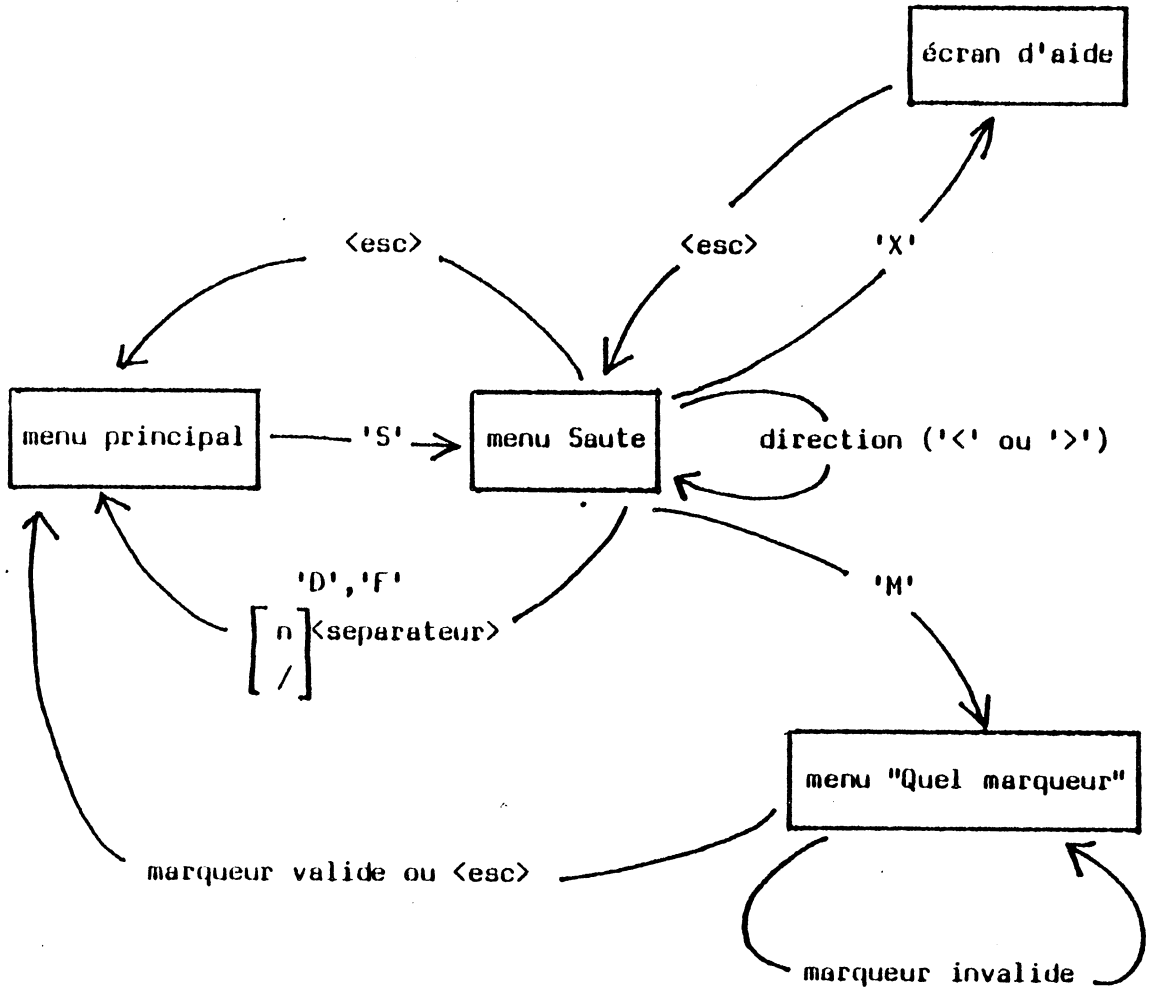



figure IV.5

La plupart des autres commandes obeissent à des schémas analogues: écrans d'aide appelés par 'X', possibilité avec <esc> "d'echapper" en annullant la commande en cours d'exécution et en revenant à l'état antérieur.

En ce qui concerne l'impression des textes, le processeur d'édition offre les choix suivants :

- nombre de lignes par page, nombre de lignes blanches en haut et en bas

- numérotation des pages ou non, choix du numéro de la première page (les numéros sont centrés en bas de page)

- impression en continu ou en page à page (avec arrêt pour remettre une

feuille à chaque page).

Lorsque l'impression est lancée, elle peut à tout moment être interrompue (en cas de "bourrage" ou de manque de papier, par exemple). Elle peut ensuite soit être reprise au point d'arrêt ou en haut de la page en cours, soit être abandonnée.

### IV.3.5.3 - LE PROCESSEUR DE FORMATTAGE

Ce processeur prend en entrée un texte fourni par le processeur d'édition, et le prépare pour l'impression avec un certain nombre d'options supplémentaires :

- choix de l'interligne par logiciel

- mise en place automatique de marques de changement de page, visulisées à l'écran, avec possibilité de déplacer ces marques à volonté dans le texte, d'en insérer et d'en supprimer

- insertion et destruction de lignes blanches

- numérotation des pages en haut ou en bas de page, au centre ou alternée gauche-droite

- mise en place ou enlèvement de soulignements

- création d'entêtes et/ou de bas de pages, saisis sous processeur de formattage (avec toutes les fonctions du processeur d'édition)

- insertion de ces entêtes et bas de pages sur la première page, ou sur toutes les pages.

- choix du nombre d'exemplaires à imprimer.

En résumé, ce sont ces deux processeurs : édition et formattage qui correspondent à la réalisation concrète des attributs de présentation.

## IV.3.6 LA PROTECTION DES DOCUMENTS

### IV.3.6.1 LES DROITS D'ACCES

La question des droits d'accès aux documents (confidentialité) ne s'était pas posée pour la maquette Solar. En effet, celle-ci était bâtie sur le SGBD Socrate, et bénéficiait des contrôles d'accès de ce système.

Par contre le SGBD Pepin que nous avons utilisé (cf chap IV.5) n'envisage pas de contrôles d'accès, car il suppose que seules les personnes "autorisées" auront physiquement accès au micro-ordinateur. Il nous a semblé que dans un environnement de bureau, de nombreuses personnes auraient accès à la machine et au système de gestion de documents, et que certains documents devraient alors être "protégés" des regards et des manipulations intempestives.

Nous proposons donc le mécanisme suivant :

Les utilisateurs sont identifiés, à l'entrée dans le système, par la frappe de leur nom et de leur mot de passe. Ils auront donc préalablement été enregistrés, et affectés à une classe de droits

1 : droits les plus faibles (personnel d'exécution)
2 : droits plus étendus
3 : tous les droits - La classe 3 correspond à la notion d'administrateur de la base. Ce sont en particulier les personnes de classe 3 qui peuvent enregistrer de nouveaux usagers, et les affecter à une classe.

Par ailleurs, une personne qui a effectivement saisi un document est dite "créateur" de ce document, et possède tous les droits de regard et de manipulation de celui-ci, quelque soit sa classe et le degré de confidentialité du document : la secrétaire qui a frappé un document, même ultra-secret, doit pouvoir y accéder pour corriger ses fautes de frappe...

De même, l'auteur d'un document, même s'il ne l'a pas effectivement saisi, doit pouvoir y accéder. On adjoindra donc à chaque document, au moment de la création, deux numéros d'utilisateur: celui du créateur

(connu du système dès son entrée), et celui de l'auteur, demandé au créateur. Le numéro d'auteur peut être vide si l'auteur saisit lui-même son document.

Les documents sont alors répartis en trois catégories :

- libres : aucune protection n'est prévue
- protégés : peuvent être consultés librement, mais le droit de modification est restreint
- confidentiels : le droit de consultation aussi est restreint

Le tableau de la figure IV.6 résume alors les droits accordés à chaque usager, selon sa "classe", sur les documents, selon leur catégorie :

| Documents / Usager | Libres | Protégés | Confidentiels |
|---|---|---|---|
| Usager de classe 1, pour les documents dont il n'est ni créateur, ni auteur | Consultation | Consultation | |
| Usager de classe 2, pour les documents dont il n'est ni créateur, ni auteur | Modification | Consultation | |
| Usager de classe 1 ou 2, pour les documents dont il est créateur ou auteur; usager de classe 3 pour tous les documents | Modification | Modification | Modification |

figure IV.6

Le droit de consultation implique le droit de copie, de même le droit de modification implique bien entendu les droits de consultation et de copie.

### IV.3.6.2 - LA GESTION DES UTILISATEURS

Il s'agit d'un ensemble de fonctions que nous avons inclus dans la BDT, alors que celles-ci auraient aussi bien pu exister dans le SGBD qui nous a servi de support. Celui-ci ne prévoyant rien, tout au moins dans sa version mono-poste, nous en avons profité pour lier la gestion des utilisateurs et la fixation des droits d'accès (cf ci-dessus chapitre IV.3.6.1).

Nous avons donc prévu :

- l'enregistrement de nouveaux usagers (nom, mot de passe, et classe de droits d'accès)

- la modification d'un usager (mot de passe, droits d'accès)

- la suppression d'un usager

- le listage des usagers

Ces fonctions ne sont accessibles qu'aux usagers de classe 3.

A la création de chaque document, une zone particulière reçoit :

- le nom du créateur

- la catégorie du document (libre, protégé, ou confidentiel).

Seuls les usagers de classe 3 peuvent modifier cette catégorie.

Au moment de l'accès à un document, le système, connaissant le nom de l'usager qui demande l'accès, pourra l'autoriser ou l'interdire en utilisant ces renseignements et la table des droits d'accès.

## IV.3.7 - LES TYPES

Rappelons (cf chap IV.2.1) qu'un type est composé de :

- la définition d'une structure logique arborescente formelle

- la liste des caractéristiques externes particulières formelles (6 au plus)

Les fonctions implémentées en ce qui concerne les types sont les suivantes :

- création, par saisie interactive ou par copie

- modification

- destruction (seulement si aucun document de la base n'utilise le type)

- affichage ou impression du contenu d'un type

- affichage d'une liste des noms de types existant dans la base

Les fonctions portant sur les caractéristiques externes consistent simplement à saisir ou afficher une suite d'au plus 6 identificateurs, non triés.

Le traitement des structures est plus complexe. Chaque structure existe sous quatre formes différentes, en d'autres termes quatre grammaires différentes peuvent décrire les structures.

Montrons ces quatre formes sur un exemple :

Soit à représenter la structure formelle suivante :

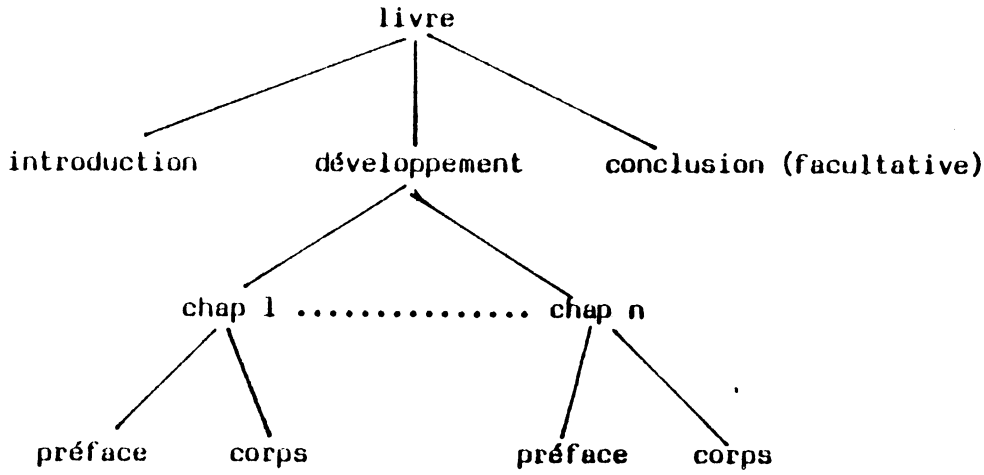



figure IV-7

La saisie initiale se fera au moyen d'un dialogue : (les réponses que doit frapper l'utilisateur sont soulignées)

Définition de structure : Nom ? LIVRE

LIVRE = D
introduction
développement
%conclusion
F

INTRODUCTION = < RC >

DEVELOPPEMENT = R chap

CHAP = D
préface
corps
F

PREFACE = <RC>

CORPS = <RC>

CONCLUSION = <RC>

C'est la forme de saisie d'une structure.

Au cours de ce dialogue, à chaque étape, le système affiche le nom de la partie à définir, suivi du signe = , ainsi qu'un menu indiquant les réponses possibles :

- D <liste de parties> F, s'il s'agit d'une partie composée de parties ayant des noms différents (bloc)

- R <nom d'une partie>, s'il s'agit d'une partie répétitive

- retour chariot symbolisé par <RC> s'il s'agit d'une feuille, élément terminal de la structure.

Le nom d'une partie peut être précédé du caractère '%' pour indiquer qu'il s'agit d'une partie facultative. Dans la pratique, nous avons limité le nom d'une partie à 15 caractères, plus éventuellement le '%'. Le système tronque automatiquement et passe à la ligne après cette limite. De plus si l'on s'aperçoit que l'on s'est trompé, à tous les niveaux de la définition on pourra frapper "escape" ( <ESC>) pour revenir au niveau supérieur.

Remarquons sur cet exemple l'algorithme de parcours d'arbre que nous avons adopté :

- Pour définir une partie, on donne la liste de ses fils

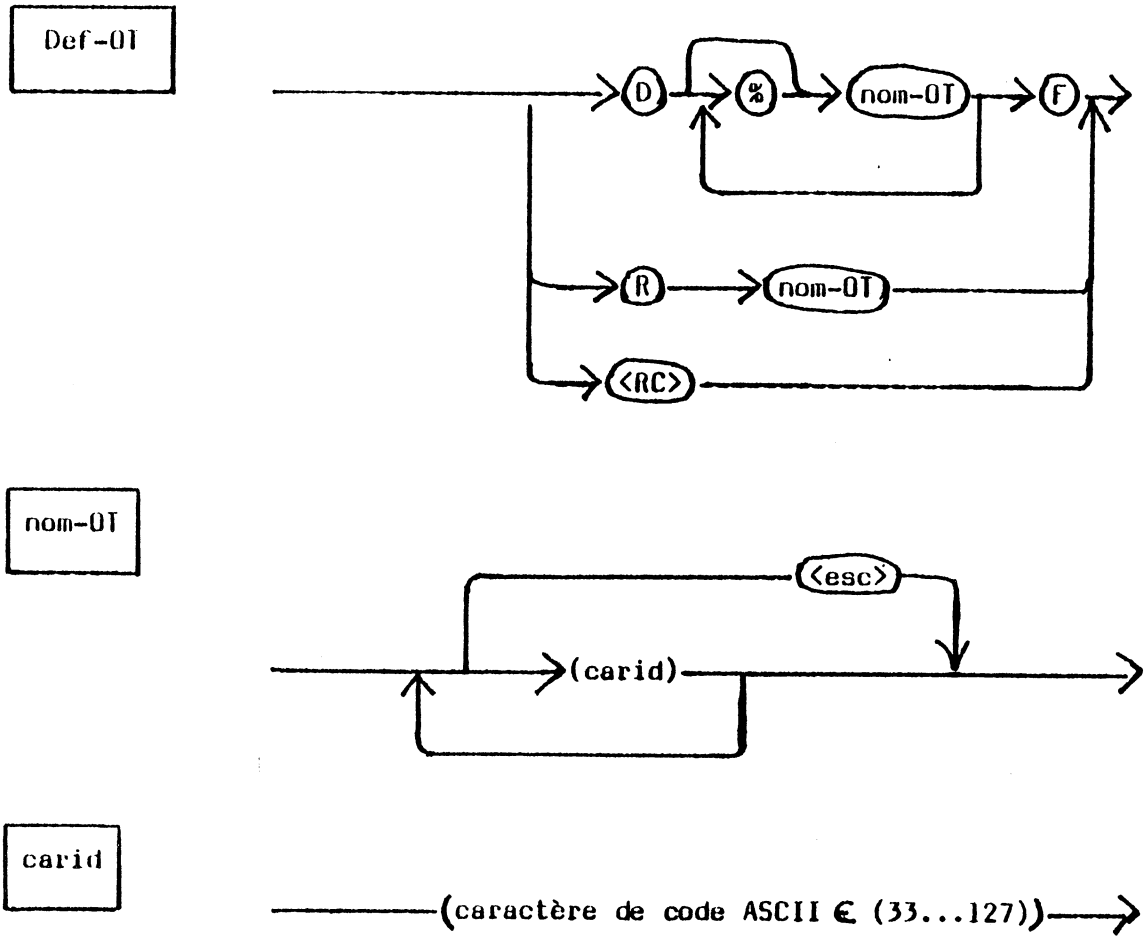
- Puis on définit de la même manière, récursivement, chacun de ses fils.

Il ne s'agit donc ni d'un parcours classique préfixé, ni d'un parcours "par niveaux". Ce type de parcours d'arbre sera repris tout au long de MIDOC, nous le détaillons ci-dessous (page 118).

La grammaire de cette forme de saisie, telle que l'usager doit la frapper, peut se représenter en Forme normalisée de Backus (BNF) ainsi :

<def-OT> ::= 'D' <liste-OT> 'F' / 'R' <nom-OT> / <RC>
<liste-OT> ::= <liste-OT><OT> / <OT>
<OT> ::= <nom-OT> / '%' <nom-OT>
<nom-OT> ::= <carid> / <nom-OT><carid> / <esc> / <nom-OT><esc>
<carid> ::= tout caractère imprimable (code ASCII dans (33...127))

Sous forme de graphe syntaxique, la grammaire de la forme de saisie devient :




Une fois que toutes les parties de la structure ont été définies, et qu'on a abouti à des feuilles partout, le système affichera la structure qu'il vient de saisir, sous la forme suivante (forme d'affichage):

```
LIVRE = DEBUT
          INTRODUCTION
          DEVELOPPEMENT
          %CONCLUSION
        FIN
DEVELOPPEMENT = REPETE CHAP
CHAP = DEBUT
          PREFACE
          CORPS
       FIN
```

La forme d'affichage ne diffère de la forme de saisie que sur des points mineurs :

- troncation à 15 caractères et passage en majuscules, s'il y a lieu, des noms de parties

- affichage complet des mots "DEBUT", "FIN" et "REPETE"

- et surtout, le fait que pour les feuilles l'on n'affiche rien.

Le parcours de l'arborescence est exactement le même que ci-dessus.

Une fois la structure saisie et validée par l'usager, le système la connait sous deux formes :

- la forme condensée, utilisée par le système pour le stockage dans la base sous forme d'une chaîne de caractères. C'est une forme préfixée tout à fait classique, avec pour seule particularité le marquage des parties répétitives et optionnelles en les faisant précéder respectivement des caractères "&" et "%". Dans l'exemple , on obtiendrait la chaine suivante:

LIVRE (INTRODUCTION DEVELOPPEMENT (&CHAP (PREFACE CORPS ))%CONCLUSION)*

* : marque de fin

- la forme interne utilisée par le système pour tous les parcours d'arborescences. C'est une transformation de l'arbre n-aire initial en un arbre binaire, où chaque noeud est une structure Pascal définie ainsi:

```
type    OT = structure
               NOM : NOM-OT;
               OPT : booleen;    (vrai pour noeud
                                      optionnel ou répétitif)
               SUCC,ALT : ↑OT;   (pointeurs sur OT)
             fin;
```

Le même exemple donnerait la forme interne suivante (fig IV.8):

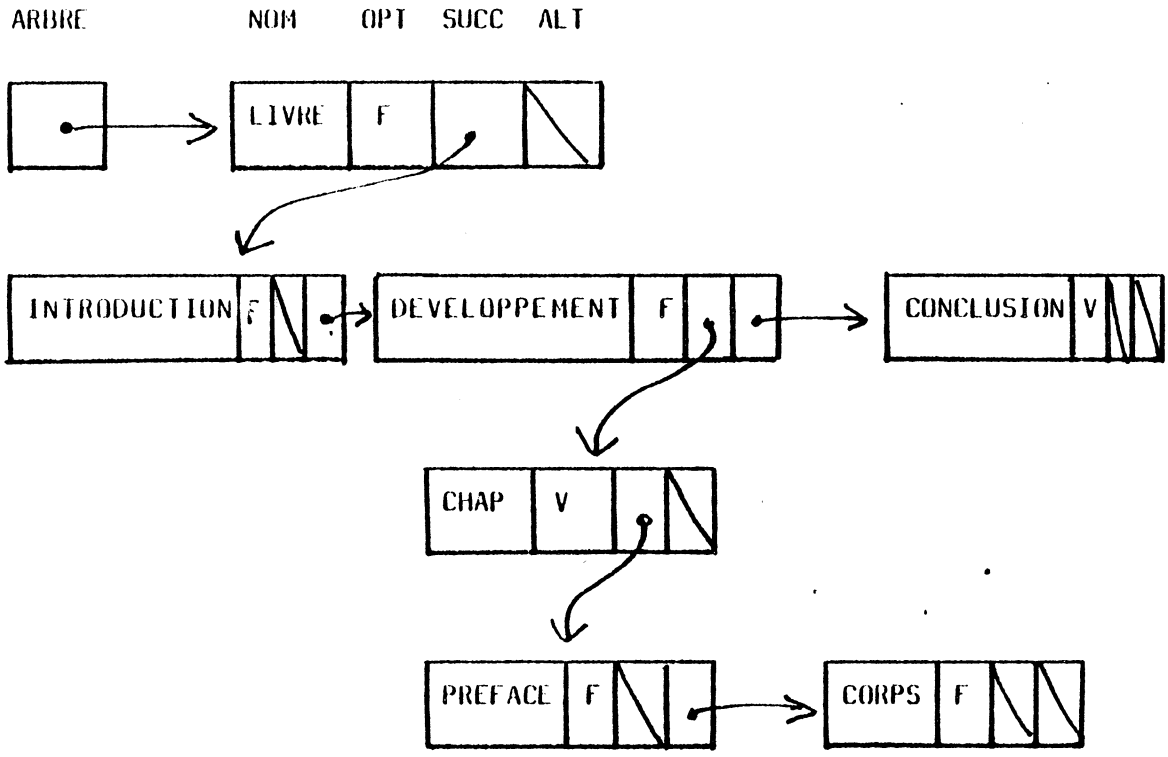



figure IV.8

Notons que les parties répétitives sont marquées comme étant leur propre alternant. De plus, elles sont optionnelles (hypothèse initiale).

Le système dispose d'utilitaires permettant le passage d'une forme à l'autre, selon les besoins (figure IV.9):

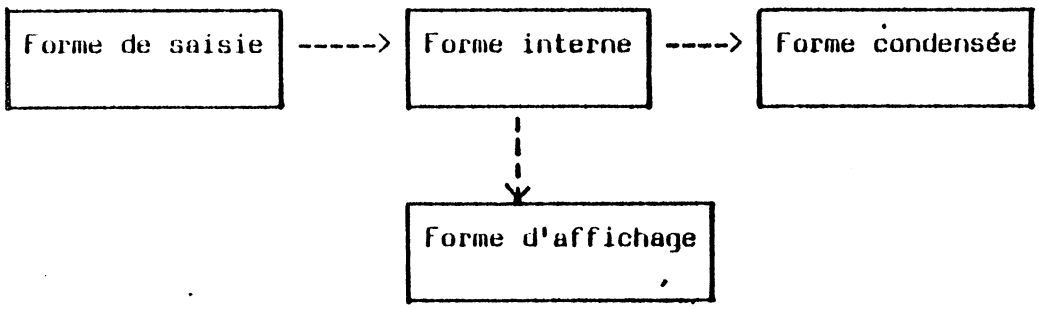



figure IV.9

L'algorithme de parcours d'arbre que nous avons constamment employé pour les passages d'une forme à l'autre, ainsi que pour le traitement des structures effectives, peut être formalisé en deux procédures :

```
procedure PARCOURS (R : ^OT);          (*R = racine de l'arbre*)
début
  VISITER (R);
  TRAVERSER (R)
fin;

procedure TRAVERSER (R : ^OT);
var R1 : ^OT;
début
  si R ≠ nil alors
  début

      R1 := R^.SUCC
      tantque  R1 ≠ nil faire      (*VISITER la racine de tous*)
      début                         (*  les sous-arbres de R   *)
         VISITER (R1);
         R1 := R1^.ALT
      fin;

      R1 := R^.SUCC                    (*TRAVERSER tous les*)
      tantque  R1 ≠ nil faire         (* sous-arbres de R *)
      debut
         TRAVERSER (R1);
         R1 := R1^.ALT
      fin

  fin
fin;
```

Par "visiter" nous entendons un traitement quelconque appliqué à un noeud de l'arbre : affichage, création,...Un exemple d'application de cet algorithme figure à l'annexe 4.

Montrons que cet ordre de parcours n'est pas classique, sur l'exemple d'arbre donné par KNUTH (KNUT69),page 334:

La forêt de la figure IV.10

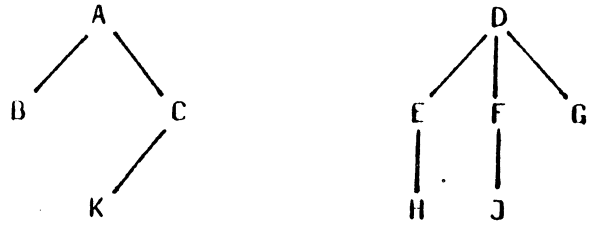



figure IV.10

donne l'arbre binaire suivant : (en ajoutant une racine unique R) (figure IV.11)

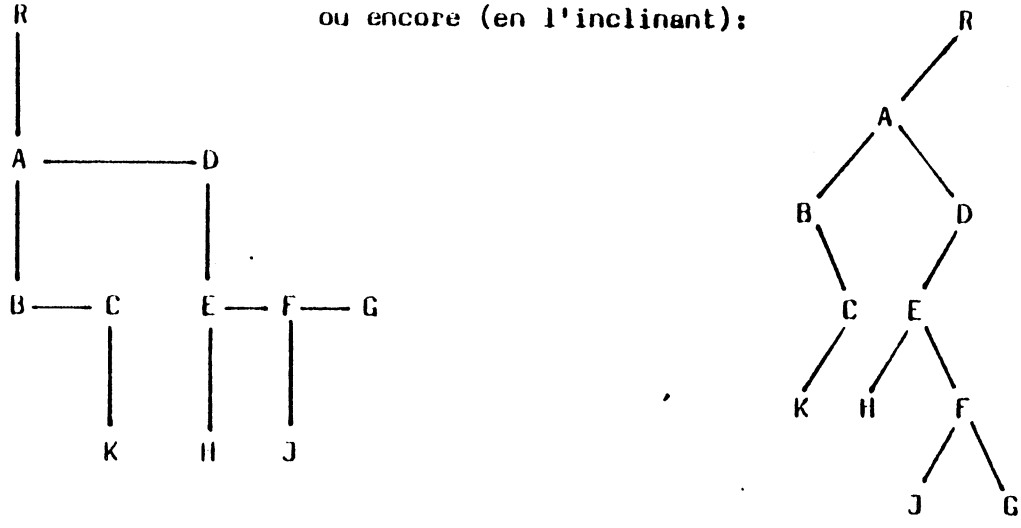



Figure IV.9

Les ordres de parcours "visite" proposés par KNUTH sont les suivants :

Kn1 Pre order (ou prefixé): R A B C K D E H F J G
Kn2 Post order (ou infixé): B K C A H E J F G D R
Kn3 End order (ou postfixé): K C B H J G F E D A R
Kn4 Reverse end order : R A D E F G J H B C K

L'ordre proposé par SALTON (level-order) (SALT62) donne :

S: R A D B C E F G K H J

alors que notre algorithme donne :

K: R A D B C K E F G H J

L'interêt de cet ordre est qu'il vérifie deux propriétés, qu'aucun des autres ordres proposés ne réunit : conservation de l'ordre de traversée des feuilles, et groupage des fils

1) Conservation de l'ordre de traversée des feuilles :

Rappellons que les arbres que nous étudions sont destinés à être "projetés" sur un texte, chaque feuille de l'arbre correspondant à un segment élémentaire de texte :

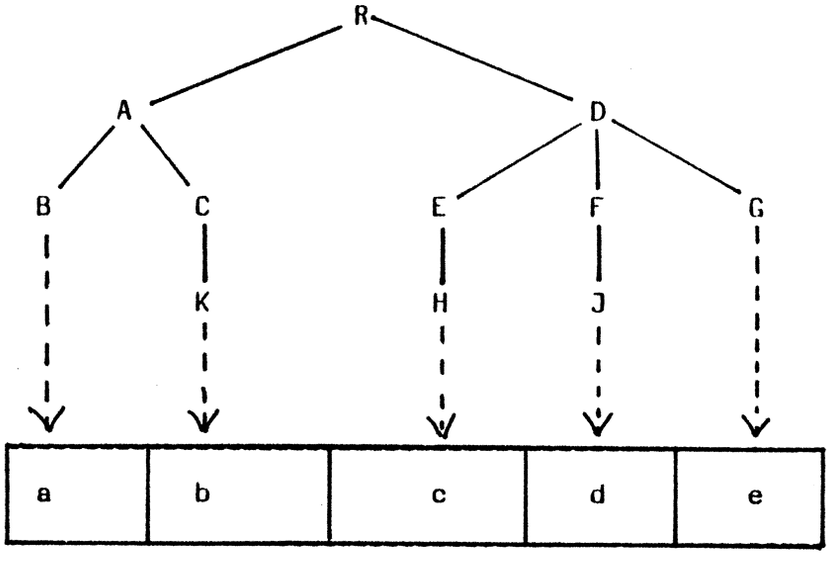



figure IV.12

Le traitement de chaque segment de texte (saisie ou impression) se fera lorsque la feuille correspondante est "traversée", c'est-à-dire en considérant que l'arbre est devenu le suivant (figure IV.13):

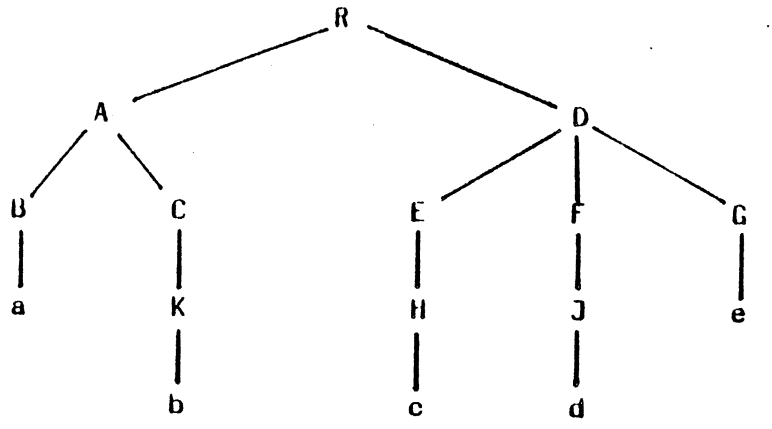



figure IV.13

Lorsque ces nouvelles feuilles sont "visitées", il est évident que la saisie et l'impression doivent se faire dans l'ordre : a b c d e. Or les ordres étudiés ci-dessus donnent :

Kn1 : R A B a C K b D E H c F J d G e

Kn2 : a B b K C A c H E d J F e G D R

Kn3 : a b K C B c H d J e G F E D A R

Kn4 : R A D E F G e J d H c B C K b a

S : R A D B C E F G a K H J e b c d

K : R A D B C a K b E F G H c J d e

On constate que Kn4 énumère les feuilles de droite à gauche et que S détruit leur ordre.

2) Groupage des fils.

Dans le traitement d'un arbre n-aire, il est souhaitable que les fils d'un même père soit visités, dans l'ordre, l'un après l'autre, sans que d'autres noeuds viennent s'intercaler dans le traitement. Cela est particulièrement intéressant dans le cas des parties "répétitives", numérotées consécutivement à partir de 1.

Dans l'exemple on doit trouver groupés :

A et D

B et C

E, F et G

Or on constate que Kn1 et Kn2 ne respectent pas ce groupage, alors que Kn3 le respecte, mais en énumérant les "frères" de droite à gauche.

En résumant la vérification de chacune des propriétés 1 et 2 pour chacun des ordres de parcours :

| | 1 | 2 |
|---|---|---|
| Kn1 | oui | non |
| Kn2 | oui | non |
| Kn3 | oui | inversé |
| Kn4 | inversé | oui |
| S | non | oui |
| K | oui | oui |

Nous constatons que seul K vérifie les deux propriétés de manière satisfaisante.

## IV.4 L'INTERFACE UTILISATEUR

Tout au long du projet MIDOC, nous nous sommes efforcés de partir du point de vue de l'utilisateur, et de rendre la mise en oeuvre du système aussi facile que possible. Nous avons donc éliminé les "langages", et les avons entièrement remplacés par des "menus". Dans la mesure du possible ces menus sont composés de mots suffisamment expressifs pour que leur sens soit évident, et l'appel d'une fonction se résume à la frappe d'une lettre (clairement précisée à l'écran) du mot choisi.

La possibilité d'erreurs de frappe et de syntaxe est donc très limitée. Dans certains cas, l'utilisateur doit frapper un mot (titre, nom d'un type, marqueur...). Il peut corriger sa frappe avant de la valider par un retour chariot. S'il ne sait pas quel mot frapper, le système peut généralement, sur demande, lui fournir la liste complète des mots possibles à ce niveau.

Dans beaucoup de menus nous avons prévu la lettre 'X', réservée à "l'appel au secours". En frappant 'X' on obtient l'affichage provisoire d'un écran qui tente d'expliquer les choix offerts. Ces écrans sont stockés à l'extérieur de MIDOC, dans un fichier spécial, et peuvent être modifiés (améliorés ou mis à jour) à l'aide de commandes spéciales de MIDOC. On pourrait en particulier songer à les traduire dans une autre langue, si le besoin s'en faisait ressentir, sans toucher à MIDOC lui-même.

D'autre part nous avons prévu à tous les niveaux de menus la possibilité de retourner au niveau antérieur sans avoir effectué une quelconque action, et même en "défaisant" une action erronée. En particulier, la destruction de chaînes de caractères est annulable, cela pour permettre d'éviter des effacements intempestifs.

Nous estimons donc que MIDOC doit pouvoir être utilisé par un personnel qui n'a aucune notion de base de données ou de langages de requêtes. Nous avons cherché à le rendre aussi facile à apprendre qu'un système de traitement de textes classique.

## IV.5 - LES MOYENS TECHNIQUES

L'ordinateur dont nous disposions était une Micromachine 2000, basée sur un Z80A, avec 64 KO de mémoire centrale. Depuis le début du projet (fin 1981) jusqu'à maintenant la mémoire secondaire est composée de disquettes souples 8", utilisées sur deux lecteurs de disquettes. Dans un proche avenir, l'un de ces lecteurs de disquettes sera remplacé par un disque dur 10 MO fixe, indispensable pour une gestion commode de gros volumes de textes.

Parmi les systèmes et langages disponibles sur cette machine :

- système CP/M : langage Basic, Cobol, Assembleur Z80, Pascal

- système UCSD : langages Pascal UCSD, Fortran et Assembleur Z80, nous avons choisi d'adopter le Pascal UCSD, version IV.0 (UCSD81).

Ce système nous a semblé offrir à la fois

- un maximum de cohérence entre le système et le langage (interface utilisateur semblable, gestion des fichiers incorporée au langage de programmation...)

- une très grande portabilité

- un éditeur de programmes remarquable, dont les principes ont servi de base à notre traitement de textes.

Au départ le chargement du système UCSD à travers le système CP/M pouvait paraître assez laborieux, mais nous avons pu bénéficier du travail de J.P. Paillard (PAIL83) sur l'écriture d'un système UCSD adaptable. Dans un premier temps, il nous a fourni un système UCSD autochargeable, puis il a écrit un BIOS (Basic Input-Output System) qui optimise les entrées/sorties sur disquettes, enfin il a porté ce BIOS sur disque dur.

Le choix d'un SGBD qui servirait de "couche de base" à MIDOC a été plus difficile. Nous souhaitions disposer d'un SGBD qui offrirait

- un interface procédural utlisable à partir de Pascal UCSD
- quelques facilités pour le traitement de chaînes de caractères.

Un certain nombre de SGBD sont en cours de dévelloppement sur micro-ordinateurs ( Minisequel, Microbe...), peu sont écrits en Pascal UCSD. C'est le système "PEPIN", développé à l'Inria par P. Bouchet et son équipe (BOUC83) que nous avons retenu. Les mécanismes de gestion de la mémoire virtuelle, ainsi que de sécurité dans les transactions qui y sont mis en oeuvre nous ont paru particulièrement interessants. Dans ses premières versions, Pépin avait du mal à réunir toutes les fonctions souhaitables, en raison du manque de place en mémoire centrale. Les versions ultérieures, mieux segmentées, semblent en voie de résoudre ces problèmes.

PEPIN est un SGBD relationnel, cela nous a permis de concrétiser nos choix fonctionnels dans un schéma relationnel d'une manière très commode et naturelle.

Par ailleurs l'interface de procédures en Pascal qui est fourni (recherche d'une sous-relation, sélection d'un n-uplet, mpdification d'un n-uplet, création de nouveaux n-uplets, extraction de valeurs d'attributs, validation de transactions...) est suffisant pour que nous puissions bâtir notre application BDT.

En tant que périphériques, nous disposions d'un clavier-écran (Dialogue80 d'Ampex) à clavier "QWERTY", nécessaire pour la programmation en Pascal, mais nous pouvions églement employer un clavier "AZERTY" adapté à la frappe de textes en langue française.

L'écran ( 24 lignes et 80 caractères par ligne ), dispose de fonctions assez riches (adressage direct du curseur, quatre attributs vidéo possibles, etc) mais malheureusement pas de touches fonctions programmables qui soient d'un emploi commode. Bien entendu, un écran "pleine page" aurait été plus agréable. Enfin nous pouvions tester les impressions sur deux imprimantes : une Lear-Siegler, à points, pour les sorties "test", et une Qume, à marguerite, dite de "qualité courrier" pour les impressions soignées.

La réalisation de la maquette Solar avait reposé sur l'hypothèse de l'utilisation ultérieure d'un A.S.T. (analyseur structurel de textes) cablé, chargé de découper le texte en segments selon le format et d'y repérer les positions d'attributs locaux et de paramètres formels. En attendant un AST cablé, la maquette a utilisé un AST simulé par logiciel, relativement lent et inefficace. La notion d'AST cablé étant tout à fait exclue dans le projet MIDOC, cela explique :

- nos choix de ne pas faire de tables de positions d'attributs locaux et de paramètres, mais de les traiter au fur et à mesure des besoins (cf chapitre IV.2.2 et IV.3.4)

- notre choix fonctionnel de saisie feuille par feuille, guidée par le type et non automatique (cf chap IV.3.1)

## IV.6 - IMPLANTATION

### IV.6.1 - LE SCHEMA RELATIONNEL

L'organisation interne des données traitées par MIDOC s'appuie sur un schéma relationnel que nous détaillons ci-dessous.

Notons que dans ce schéma l'on peut définir tant les documents eux-mêmes que des éléments auxiliaires tels que les types, le dictionnaire des mots clés, et même la liste des utilisateurs.

Pour chaque relation, nous donnons la liste de ses attributs ("titre" de chaque colonne dans la représentation classique sous forme d'un tableau à deux dimensions).

Il faut noter que certains de ces attributs ne servent que de code interne, plus commode pour des sélections ultérieures, qu'une clé formée de plusieurs identificateurs. Ainsi le "NUM-DOC" fourni par la relation RECH-DOC permettra de selectionner certains n-uplets de la relation OBJET-TEXTUEL.

Pour chaque attribut nous donnons entre parenthèses son domaine de définition, pris parmi les suivants :

Cn : chaîne de n caractères au plus
E : entier

Précisons tout d'abord que le SGBD Pepin ne peut pas traiter de chaînes de plus de 512 caractères. Le segment de texte qui correspond à une feuille de la structure effective, que nous appelons "texte-feuille", sera donc formé d'autant de chaînes de 512 caractères, ou "pages" qu'il est nécessaire, plus une dernière chaîne plus courte pour compléter. De plus, chaque document et chaque texte-feuille sera accompagné d'une page spéciale, nommée "page-zéro", qui contient un certain nombre de renseignements externes au texte : date de création, longueur totale du texte, positions des marqueurs, options de justification, tabulations, ainsi que des attributs de présentation : marges, numérotation, etc.

Relation DOCUMENT

NUM-DOC (E) ..... code interne de document

AUTEUR (C20) ..... Valeurs des caractéristiques externes
DATE (C6) ..... générales, autre que le titre. Notons
NUM-REF (E) ..... que le numéro de référence, lié à l'organisation externe, n'est pas une clé priviligiée

VAL-CAR-EXT1 (C20)... Valeurs des caractéristiques externes particulières, dont les noms sont
. donnés dans le type
.
VAL-CAR-EXT6 (C20)...

LISTE-PARAM (C512)... Liste des paramètres formels

CATEGORIE (C1) ..... L pour libre, P pour protégé, C pour confidentiel

NUM-CREATEUR (E) .... numéro d'utilisateur de la personne qui a créé (saisi) le document. Ce n'est pas nécessairement la même personne que l'auteur de RECH-DOC.

Relation RECH-DOC

NOM-TYPE (C10) ..... nom du type

TITRE (C20) ..... caractéristique externe générale formant avec NOM-TYPE une clé d'accès

NUM-DOC (E) ..... code interne de document

Relation PAGEZERO

NUM-DOC (E) ..... Code interne du document

PAGE (C512) ..... Texte de la page zéro de ce document

Relation OBJET-TEXTUEL ou OT

(Rappellons qu'un OT est un noeud de la structure effective d'un document.)

NUM-DOC (E) ..... Code interne du document

NUM-OT (E) ..... Code interne de l'OT (numéros à partir de 1 pour un document)

NOM-OT (C15) ..... Nom de l'OT (tiré de la structure formelle du type)

OT-ALT (E) ..... numéros des OT alternant et successeur dans la structure;

OT-SUCC (E) ..... 0 pour NIL.

NUM-TF (E) ..... Numéro du segment de texte (texte feuille) pour les OT feuilles non vides, 0 pour les OT non feuilles ou vides.

Relation TEXTE-FEUILLE ou TF

NUM-TF (E) ..... code interne d'un texte feuille

ORIGINE (C1) ..... origine du TF :
'E' pour processeur d'édition
'F' pour processeur de formattage

NB-PAGES (E) ..... nombre de pages du texte-feuille page zéro non comprise.

NUM-PAGE (E) .....numéro d'ordre de la page dans le texte-feuille
(de 0 à NB-PAGES)

PAGE (C512) ..... Texte de la page

Rappellons que :

- chaque texte-feuille peut exister en deux "versions" : l'une issue du processeur d'édition, l'autre du processeur de formattage

- chaque texte-feuille est composé d'autant de pages qu'il est necessaire, y compris la page zéro. Pour un numéro de texte-feuille, il pourra donc y avoir deux ensembles de n-uplets dans la relation TF, l'un d'origine 'E', l'autre d'origine 'F', et dans chaque ensemble les numéros de page iront de zéro jusqu'au nombre nécessaire.

Relation TYPE

NOM-TYPE (C15) ..... identificateur du type

STRUCTURE (C512) .... structure formelle sous la forme
"condensée" (cf IV.3.1)

NOM-CAR-EXT1 (C15) ..
:
NOM-CAR-EXT6 (C15) ..
} Noms des caractéristiques externes particulières, propres à ce type

Relation MOTS-CLES

(Le dictionnaire ou lexique de mots-clés équivalent à un fichier inverse)

MOT-CLE (C20) ..... Libellé du mot-clé

REFERENCES (C50) ..... Chaîne de 50 caractères traitée comme une chaîne de bits, indiquant la présence ou l'absence du document correspondant (limitation arbitraire du nombre de documents à 50 x 8 = 400)

Il aurait peut-être été plus simple d'avoir un n-uplet par référence, de la forme (MOT-CLE, NUM-DOC), mais il aurait alors fallu utiliser les opérations de sélection de sous-ensembles de n-uplets, prévues dans PEPIN, pour récupérer l'ensemble des références pour un mot-clé. L'obtention de listes de références pour une expression logique sur mots-clés aurait à son tour fait appel aux opérations logiques sur les relations temporaires ainsi sélectionnées. Ces opérations demandent plus de temps que des simples opérations logiques sur des chaînes de bits.

Une autre solution possible aurait consisté à prévoir une limitation sévère du nombre de références pour un document, et inclure dans le schéma autant d'attributs "NUM-DOC i" que de références possibles: (MOT-CLE, NUM-DOC1, NUM-DOC2,... NUM-DOCn) si on autorise n rérérences.

Mais cette limitation est par trop restrictive, et par ailleurs PEPIN nous oblige à "économiser" le nombre total d'attributs. C'est pourquoi la solution des chaînes de bits nous a paru la meilleure.

Relation UTILISATEUR

NOM-UTIL (C10)..... nomd'unutilisateur, telqu'ilest enregistré

NUM-UTIL (E) ..... Numéro (repris dans NUM-CREATEUR de la relation DOCUMENT)

MOT-PASSE (C5) ..... Pour le contrôle d'accès au système

CLASSE (E) ..... 1,2 ou 3 selon les droits d'accès aux documents

Remarquons que les dépendances fonctionnelles élémentaires que l'on peut définir sur ces attributs sont les suivants :

DOCUMENT
- NUM-DOC ------> AUTEUR
- " ------> DATE .
- " ------> VAL-CAR-EXT1 à 6
- " ------> NUM-PAGEZERO
- " ------> LISTE-PARAM
- " ------> CATEGORIE
- " ------> NUM-CREATEUR

RECH-DOC
- NUM-DOC ------> NOM-TYPE
- " " ------> TITRE
- (TITRE, NOM-TYPE)------> NUM-DOC

PAGEZERO
- NUM-DOC ------> PAGE

OT
- (NUM-DOC,NUM-OT) ------> NOM-OT
- " ------> OT-ALT
- " ------> OT-SUCC
- " ------> NUM-TF

TF
- (NUM-TF,ORIGINE,NUM-PAGE) ------> PAGE

TYPE
- NOM-TYPE ------> STRUCTURE
- " ------> NOM-CAR-EXT1 à 6

UTIL
- NOM-UTIL ------> NUM-UTIL
- " ------> MOT-PASSE
- " ------> CLASSE
- NUM-UTIL ------> NOM-UTIL
- " ------> MOT-PASSE
- " ------> CLASSE

MOTCLE
- MOT-CLE ------> REFERENCE

Il y a donc dans chaque relation, une clé (au moins), formée d'un ou plusieurs attributs tel que tous les autres attributs sont pleinement dépendants de cette clé et ne dépendent d'aucun autre attribut. Ce schéma relationnel est donc conforme à la troisième forme normale de Codd (CODD70)(DEL082)(GARD83), et ne devrait pas poser de problèmes d'anomalies de mise à jour.

### IV.6.2 LES MODULES ET FICHIERS

Afin de réduire le volume du code présent en mémoire à un moment donné, le logiciel de Midoc se présente sous la forme de plusieurs programmes ("modules") différents, le passage de l'un à l'autre se faisant au moyen d'une commande d'enchainement du moniteur. La transmission des données d'un programme à l'autre ne peut se faire que dans des fichiers. Par ailleurs, chaque programme est lui-même segmenté, les différents segments pouvant communiquer sans difficulté entre eux à l'intérieur d'un même programme. Les liaisons entre les modules et fichiers se présentent ainsi (figure IV.14):

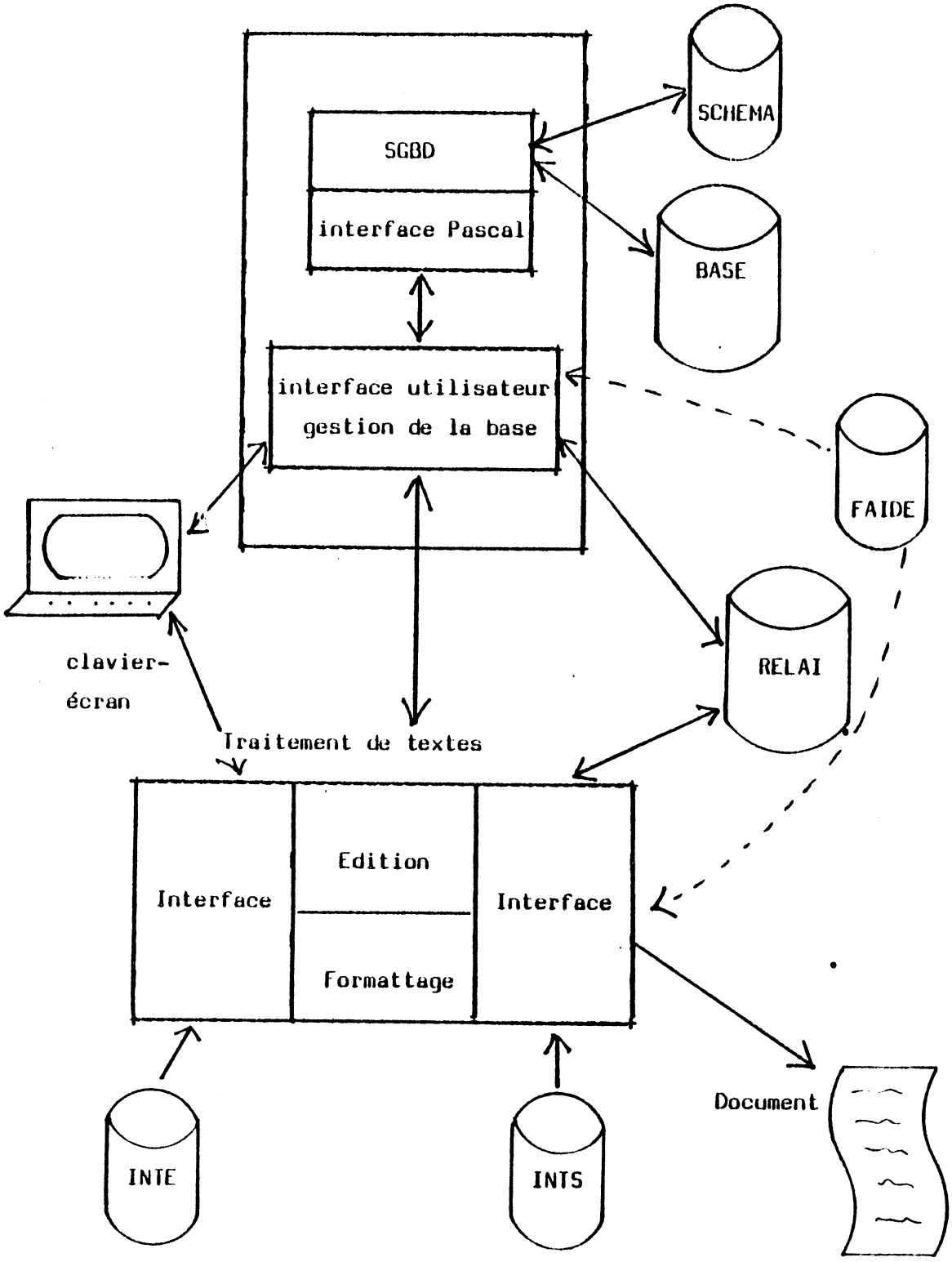



fig IV.14

Remarquons que le SGBD Pépin, chargé de gérer directement le schéma interne et la base, pourrait être remplacé par un autre SGBD relationnel sans grandes difficultés, à condition que celui-ci possède aussi un interface en Pascal UCSD. En effet, à aucun moment notre logiciel n'agit directement sur la base sans passer par le SGBD.

Le module "interface-gestion de la base" a plusieurs fonctionnalités :

- identification/gestion des utilisateurs
- gestion des types (structures formelles et caractéristiques externes particulières)
- consultation de la base de documents
- mise à jour de la base (textes-feuilles, structures effectives, valeurs de caractéristiques externes, mots-clés, etc).

Le module de "traitement de textes" est composé de deux programmes : le processeur d'édition et le processeur de formattage, qui ont les mêmes interfaces d'entrée et sortie pour la normalisation des commandes des périphériques.

Ces interfaces sont constitués à partir des fichiers INTE et INTS, qui contiennent les codes des différentes commandes propres aux périphériques (clavier-écran et imprimante) utilisés. Il peut y avoir autant de fichiers INTE et INTS que de périphériques différents.

Bien que Pépin autorise, dans ses dernières versions tout au moins, la modification dynamique du schéma par l'utilisateur, nous n'utilisons pas cette possibilité. Notre logiciel est en effet bâti sur un schéma figé, (cf IV.6.1), inaccessible et invisible pour l'utilisateur. Par contre il est possible qu'à l'avenir nous puissions élargir MIDOC au traitement de données autres que textuelles,il faudra alors fournir à l'utilisateur des outils commodes, aussi interactifs que possible, pour la définition de nouvelles relations. En attendant, le fichier SCHEMA sera fourni avec le système MIDOC et ne sera utilisé qu'en lecture.

Le fichier BASE contient tous les catalogues et les valeurs des attributs des n-uplets de la base, y compris les pages de texte.

Le fichier FAIDE, utilisé à partir des deux grands modules de MIDOC, contient le texte des écrans d'aide (cf IV.4). Comme le fichier SCHEMA, il est en principe figé et inaccessible à l'utilisateur. Néanmoins, un programme utilitaire, disponible dans le processeur d'édition, permet éventuellement de modifier le texte des écrans d'aide. Cet utilitaire est appelé en frappant une lettre qui ne figure pas au menu, seules les personnes compétentes en auront connaissance.

Le rôle du fichier RELAI est de servir d'intermédiaire entre la gestion de la base proprement dite et les programmes de traitement de texte. Rappellons qu'au niveau conceptuel, l'unité de texte est le "texte-feuille", segment de texte qui correspond à une feuille de la structure. Nous avons vu (relation TF du chapitre IV.6.1) que ce texte feuille est constitué physiquement dans la base, d'une suite de pages de 512 caractères. Or le traitement de texte doit pouvoir disposer, au minimum, d'un texte-feuille complet.

Au moment de l'appel du traitement pour un OT, plusieurs cas peuvent se présenter :

- a) Il s'agit d'un OT feuille (ou "OT terminal") encore vide, que l'on veut saisir: le texte est saisi en mémoire centrale sous traitement de texte, rangé dans le fichier RELAI, puis découpé en pges qui sont stockées dans la base comme autant d'attributs "PAGE" des n-uplets de la relation TEXTE-FEUILLE.

- b) Il s'agit d'un OT feuille ayant déjà un texte : les pages successives pour la reconstitution du texte-feuille correspondant sont prises dans la base, et recopiées dans le fichier RELAI. Le traitement de texte charge le texte en mémoire à partir du fichier RELAI, modifie le texte, le range à nouveau dans RELAI, d'où il est redécoupé en pages pour être stocké dans la base.

- c) Il s'agit d'un OT non terminal encore vide : le système demandera toujours la saisie feuille par feuille, se ramenant ainsi au cas (a).

- d) Il s'agit d'un OT non terminal, ayant déja un texte : MIDOC va parcourir le sous-arbre correspondant à cet OT, et charger les textes feuilles de celui-ci consecutivement dans RELAI, comme en (b). Un marqueur particulier sera inséré à la fin de chaque texte-feuille, dans RELAI, afin de conserver les "limites" entre chaque texte-feuille.

Le traitement de texte chargera alors en mémoire autant de pages que possible, correspondant à un ou plusieurs textes-feuilles entiers consécutifs. Une fois ceux-ci traités, ils sont rangés dans le fichier RELAI, et remplacés en mémoire par les suivants, jusqu'à épuisement du contenu du fichier RELAI. Celui-ci redécoupe alors ses feuilles en pages, pour stockage dans la base.

Exemple:

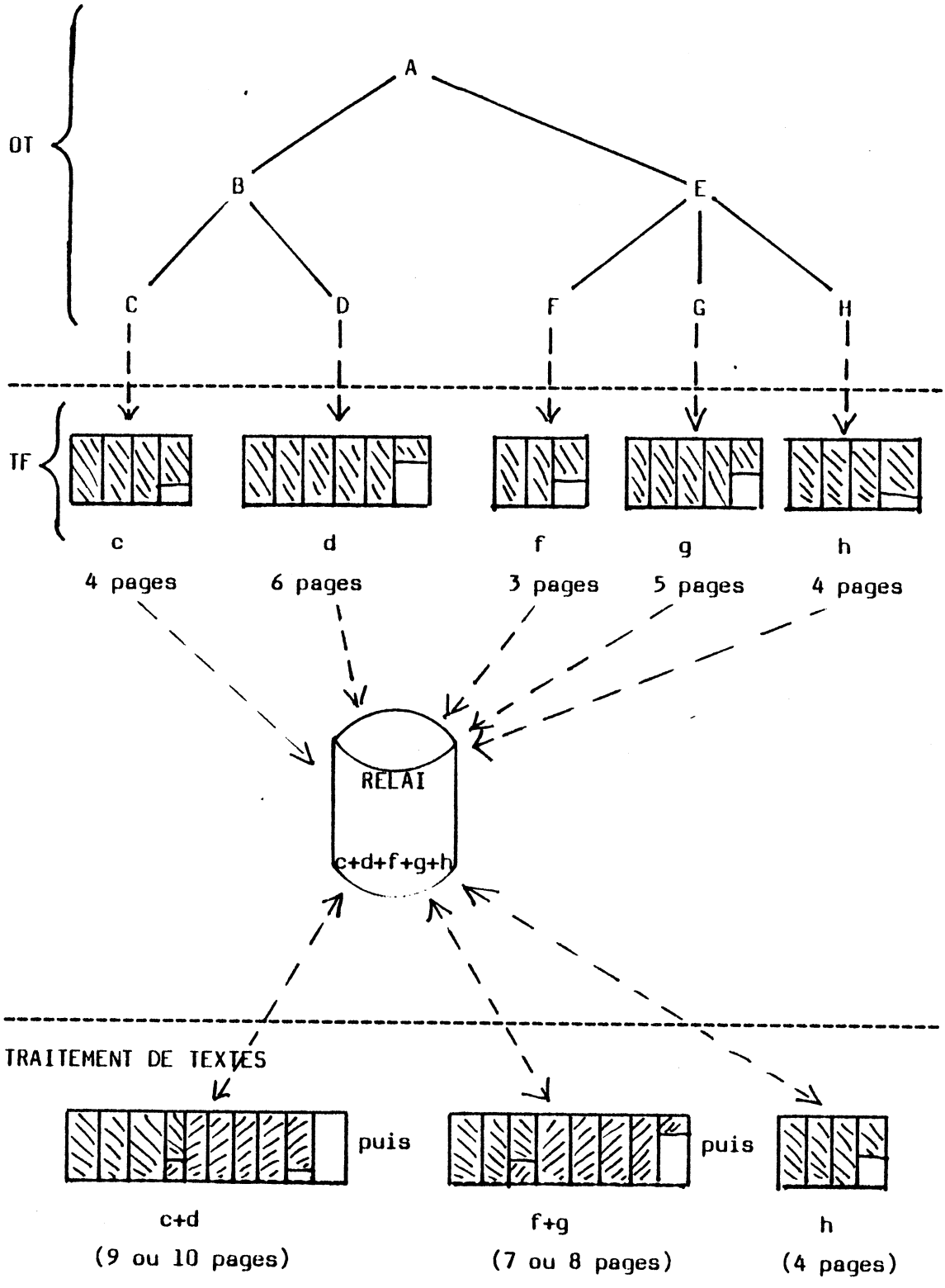



figure IV.15

Dans la figure IV.15 nous supposons que la taille mémoire est de 10 pages. Comme c occupe plus de 3 pages et moins de 4 pages complètes, d occupe plus de 5 pages et moins de 6 pages complètes, leur concaténation occupera plus de 8 pages et moins de 10 pages complètes. On pourra donc charger c+d en mémoire centrale, mais pas c+d+f qui dépasserait les 10 pages. De même, on pourra charger f et g en même temps, mais h ne pourra y être joint.

Dans le système MIDOC actuel, avec 64KO de mémoire centrale, le texte-feuille est limité à 18K caractères, soit 36 pages de 512 caractères. Bien entendu, toute augmentation de taille de la mémoire centrale profitera intégralement à la taille du texte que l'on peut traiter.

Nous n'avons pas envisagé de mettre en oeuvre un mécanisme de pagination ("swapping") au niveau des textes-feuilles. En effet, d'une part, celui-ci ralentirait considérablement le traitement de textes, d'autre part nous pensons que la structuration de textes en parties élémentaires (éventuellement répétitives en nombre illimité) permet toujours de se ramener à des longueurs suffisamment petites.

### IV.6.3 - SECURITE

Les questions de sécurité (cohérence) dans un SGBD peuvent de manière classique se répartir en plusieurs catégories (DEL082):

- la synchronisation des accès concurrents : le problème ne se pose pas pour le système MIDOC tant qu'il demeure mono-poste. A l'avenir, si l'on envisageait de le rendre multiposte, le SGBD Pépin prévoit un algorithme de synchronisation par verrouillage des transactions au niveau des relations.

- la sécurité d'utilisation (droits d'accès). Le problème est ignoré dans la version monoposte de Pépin: on y suppose que chaque machine ne sera utilisée que par une personne, ou tout au moins qu'une personne pourra toujours contrôler l'accès physique à la machine. Nous pensons que ces conditions ne seront pas toujous vérifiées, et qu'il est donc préférable de prévoir un contrôle par logiciel des droits d'accès. Nous l'avons donc traité dans MIDOC (cf chapitre IV.3.6).

- les contraintes d'intégrité statiques se bornent à la vérification:

- de l'unicité des clés des relations, et

- de l'appartenance de certains attributs à des domaines de définition. PEPIN peut être chargé de ces tâches, dans certains cas il sera plus simple d'effectuer des contrôles dès la saisie, au niveau de MIDOC, afin d'émettre aussitôt des messages d'erreur appropriés et resaisir les valeurs en cause. Citons par exemple la classe d'un utilisateur qui doit être égale à 1, 2, ou 3, ou la liste des paramètres formels qui doit obéir à une certaine syntaxe (cf chap IV.3.4).

- Les contraintes d'intégrité dynamiques seront programmées dans MIDOC : par exemple à la destruction d'un document il faudra détruire tous les n-uplets des relations RECH-DOC,OT,TF et MOTS-CLES portant le numéro du document détruit; puis il faudra aussi détruire le n-uplet de la relation PAGEZERO qui contient la pagezero du document détruit.

- enfin : la sureté du fonctionnement et la reprise après pannes. Plusieurs cas de figure peuvent se présenter :

- en cas de fausse manoeuvre pendant le traitement de texte, par exemple une destruction massive erronée, il suffit de frapper la touche <escape> pour "défaire" la destruction;

- si un incident se produit en cours de traitement de textes, effaçant la version en mémoire centrale d'un texte, la version stockée sur disque dans le fichier RELAI est conservée, on peut ainsi perdre, au pire, le contenu d'un texte-feuille que l'on vient de saisir ;

- en cas d'accident matériel sur un support magnétique, les utilitaires de Pascal UCSD permettent souvent de récupérer le contenu de fichiers défectueux. Cette possibilité, qui peut s'avérer utile si l'on travaille avec des disquettes souples, support relativement peu fiable, n'est néanmoins pas à la portée de tous les utilisateurs, car elle nécessite une bonne connaissance du gestionnaire de fichiers de Pascal UCSD. La méthode la plus prudente consiste bien sûr à effectuer des sauvegardes régulières de la base sur d'autres disquettes.

- Si un incident se produit au cours d'une transaction sur un n-uplet sous Pépin, la reprise est effectuée par le mécanisme de mise à jour différée de celui-ci : les modifications sont toujours faites sur

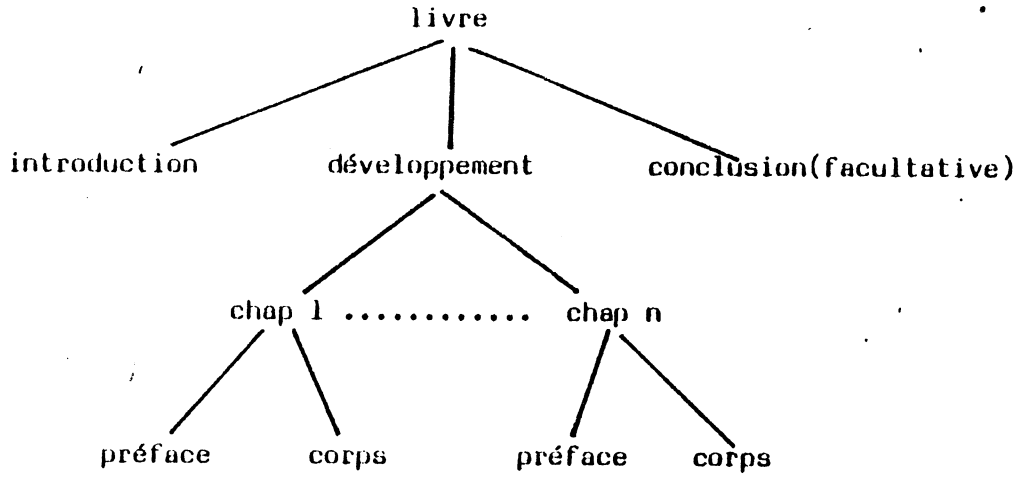
des pages copies des pages initiales ; à la validation d'une transaction, les pages d'origine sont effacées, et les pages copies les remplacent. En cas d'incident, la transaction est avortée et les pages copies sont effacées, la base garde donc son état initial.

La notion de transaction sous Midoc est plus large que sous Pépin: une création ou une mise à jour de document entrainent l'exécution de plusieurs transactions de Pépin, une pour chaque OI et chaque IF créé ou modifié. Si une panne se produit pendant ces opérations, Pépin ne pourra annuller que la dernière transaction Pépin en cours, mais la transaction MIDOC demeurera incomplète. Il n'est pas pensable de "défaire" complètement la transaction entamée sous MIDOC et de revenir à l'état antérieur, car tout le travail de saisie déjà effectué serait à refaire.

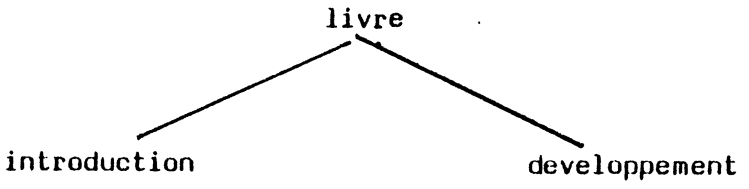
Par ailleurs, le même problème se pose si l'on souhaite créer un document en plusieurs séances de travail, même si aucune panne ne s'est produite.

La solution que nous envisageons de mettre en oeuvre est la suivante :

La création d'un document commencera par la création d'une structure effective minimale, selon le type choisi, avec des OI-Feuilles vides. Rappelons que par structure minimale, nous entendons une structure dont toutes les parties optionnelles (facultatives ou itératives) sont absentes. Ainsi, à la structure formelle suivante:



correspondrait la structure effective minimale suivante :




Toutes les autres opérations (création d'OT supplémentaires et de textes-feuilles) seront considérées comme des mises à jour. On est donc ramené à des problèmes de reprises sur des opérations plus petites :

- création de structure effective minimale

- création ou modification d'un OT

- création ou modification d'un texte-feuille.

- La création ou modification d'un OT est une opération portant sur un n-uplet, donc prise en charge par Pépin.

- La création d'une structure effective minimale, se fera en trois étapes :

- affectation d'un numéro interne NUM-DOC

- création des n-uplets OT nécessaires

- création du n-uplet RECHDOC comportant le numéro (clé unique) NUM-DOC.

C'est cette dernière étape qui servira de validation. En effet, si elle n'a pas lieu en raison d'une panne, les n-uplets OT créés ne pourront pas être atteints, puisque leur attribut NUM-DOC ne correspondra à aucun document dans la relation RECH-DOC. Il faudra prévoir un programme utilitaire ramasse-miettes qui récupérera les OT inutiles, lorsque le besoin de place se fera sentir.

- La création ou la modification d'un texte-feuille comporte la création ou la modification de plusieurs n-uplets, à raison d'un par page. Si une panne se produit pendant un transfert entre le fichier RELAI et le fichier BASE, une solution simple consiste à laisser à l'utilisateur le soin de s'apercevoir que ce texte-feuille est incomplet ou erroné, et qu'il doit être modifié par une nouvelle saisie. Le travail à refaire éventuellement ne porte que sur un texte-feuille.

## IV.7 - BILAN DU PROJET MIDOC

Il est trop tôt pour tirer un bilan définitif du projet MIDOC, car celui-ci n'est pas encore complètement programmé.

Les modules de traitement de textes fonctionnent de manière opérationnelle, avec des temps de réponse satisfaisants, mais indépendamment de la base de données. Le module de saisie, affichage, mise à jour, et transformation des structures formelles d'une forme à une autre (cf chapitre IV.3.7) fonctionne aussi parfaitement.

L'interface de gestion de la base de données est actuellement en cours de programmation, en effet nous n'avons pu disposer que tout dernièrement des versions de Pépin (chaînes variables et opérations relationnelles) qui nous étaient nécessaires. Nous pensons que l'étude préalable exposée ci-dessus et les algorithmes déjà écrits sont suffisamment précis pour que cette programmation effective puisse se faire assez rapidement et sans trop de difficultés.

Si l'on se réfère aux objectifs exposés dans le chap I.6, on peut dire que les objectifs (b) (c) (e) (f) (g) et (h) seront atteints :

- séparation des caractéristiques externes et du texte

- accès à l'ensemble d'un document

- système dynamique (types et mots-clés définis par l'utilisateur)

- recherche sur caractéristiques externes ou sur mots-clés.

Par contre l'objectif (d) : plaquer plusieurs structures sur un même texte, ne nous paraît pas indispensable dans un petit système, et l'objectif (a), le partage des textes, n'a pas été retenu. Nous expliquons les motifs de ce choix au chapitre IV.3.1.

En ce qui concerne le modèle général d'une base de données étendue aux textes, tel que nous le décrivons dans le chapitre II, il est certain que MIDOC souffre d'abord d'être limité aux documents, et de ne pas traiter d'autres entités. Il sera tout à fait possible par la suite d'inclure la possibilité d'agir sur le schéma, et donc de créer de nouvelles entités, non documentaires.

Au niveau des documents eux-mêmes une des lacunes de MIDOC est l'absence de traitement spécifique de tableaux, paramétrés ou non. Par ailleurs le traitement de textes ne permet pas, pour l'instant, l'utilisation d'indices et d'exposants.

Malgré ces imperfections, nous espérons avoir démontré qu'il est possible de réaliser un système complet de gestion de documents, accessible même à des personnes non spécialistes, sur un micro-ordinateur.

## CONCLUSION

Pour conclure, nous reprendrons le bilan des deux réalisations auxquelles nous avons travaillé:

- la première a montré qu'il était possible de traiter des documents à travers leur structure propre, en rattachant à cette structure toutes les notions de présentation physique de documents imprimés; par ailleurs elle a permis d'intégrer les documents aux autres types de données traitées par une SGBD, tout en prévoyant les traitements spécifiques que requièrent les documents. Elle a ouvert la voie aux bases de données généralisées, qui traiteront aussi bien les données graphiques et vocales que les données textuelles et factuelles. Celles-ci trouveront leur place au sein de grandes organisations, implantées sur des réseaux locaux.

- la seconde montre la possibilité de réaliser un système intégré de traitement de documents structurés, à l'échelle d'un micro-ordinateur, incorporant les possibilités d'un système de traitement de textes, d'une base de données textuelles, et d'un système de recherche documentaire. La préparation de ce mémoire, à l'aide d'un système classique de traitement de textes, nous a permis d'en mesurer les lacunes, et nous a montré combien la prise en compte de la structure serait utile. Un système tel que MIDOC aura sa place dans de petites entreprises, laboratoires, etc, et devrait leur permettre de gérer plus facilement leur documents que les systèmes dits "bureautiques" actuels.

# ANNEXES

## 1- MAQUETTE SOLAR : IMPRESSION D'UNE STRUCTURE DE DOCUMENT

### Définition d'entité Socrate employée dans les programmes:

```
ENTITE (500) OT
DEBUT
  TYPE MOT (30)
  NUMPERE BINAIRE DE 1 A 500
  NBFILS BINAIRE DE 0 A 500
  RANG BINAIRE DE 0 A 500
  MODE (5 32) (BLOC REPETE FEUILLE)
  MODELE REFERE UN MODELE-OT DE UN FORMAT-OBJET
  PREMUT REFERE UN UT
  DERNUT REFERE UN UT
FIN
```

### Programmes et macro-instructions:

```
:DEFMAC IMPROT :
:EXP
SI MODE DE :1: != 'FEUILLE' ALORS
 I (1 10) TYPE DE :1:
 M Y25 = NUMPERE DE :1:
 SI EXISTE Y25 ALORS
  SI MODE DE UN OT Y25 = 'REPETE' ALORS
   I (+4 -3) RANG DE :1:
  FIN
 FIN
 M Y25 = NUMDE :1:
 M Y24 = 1
 M Y23 = 0
 M Y22 = NBFILS DE :1:
 I (+1) '='
 SI Y22 = 0 ALORS
   ECRIRE
 SINON
  FAIRE
```

```
    SI Y24 > Y22 ALORS SORTIE FIN
    M X8 = UN OT AVEC NUMPERE = Y25;
                AYANT RANG = Y24;
    SI EXISTE X8 ALORS
     I (+3 10) TYPE DE X8
     SI MODE DE :1: = 'REPETE' ALORS I (+3 -2) Y24 FIN
     M Y23 = Y23 + 1
     SI Y23 = 4 ALORS
      ECRIRE
      I (1 5) '      '
      M Y23 = 0
     FIN
    FIN
    M Y24 = Y24 + 1
    REFAIRE
   FIN
  FIN
  SI Y23 > 0 ET Y23 < 5 ALORS ECRIRE FIN
FIN
:FDEF?

:DEFPRO IMPARB
/* AUXILIAIRE PERMETTANT LA COMPILATION DE IMPARBRE */
:CONTXT
D X1 = UN OT
D X2
D X8
:EXP
I 'TOTO'
:FDEF

:DEFPRO IMPARBRE
/* AUXILIAIRE POUR LA RéCURSIVITE DE IMPARB */
:CONTXT
D X1 = UN OT
D X2 = UN OT
D X8
:EXP
EXEC IMPARB ( X2)
:FDEF
```

```
:SUPEXP IMPARB
:DEFPRO IMPARB
/* X1 = RACINE DE L'ARBRE A IMPRIMER */
:CONTXT
D X1 = UN OT
D X2
D X8
:EXP
POUR X0
 IMPROT X1
 M Y25 = NUMDE X1
 M Y24 = 1
 M Y23 = NBFILS DE X1
 SI MODE DE X1 ]= 'FEUILLE' ET Y23 = 0 ALORS
  I '       ..VIDE..'
 FIN
 FAIRE
  SI Y24 > Y23 ALORS SORTIE FIN
  M X2 = UN OT AVEC NUMPERE = Y25 ;
              AYANT RANG = Y24 ;
  SI EXISTE X2 ALORS
   EXEC IMPARBRE
  FIN
  M Y24 = Y24 + 1
  REFAIRE
 FIN
FIN
D X2
D X8
:FDEF
```

## 2 - MAQUETTE SOLAR : REQUETES CONCERNANT LES TABLEAUX

### SAISIE DE TABLEAUX

Les requêtes suivantes permettent de saisir ou de modifier tout ou partie d'un tableau désigné par un Ti. Celui-ci doit auparavant avoir été affecté à un tableau.

Pour chaque élément à saisir, la mention

ELEMENT : <NOM LIGNE> <NOM COLONNE>

est affichée, et l'utilisateur peut alors frapper une valeur suivie d'un <rc>, ou <rc> seul s'il souhaite laisser la valeur indéfinie.

La requête STAB <Ti>

permet de saisir toutes les valeurs d'un tableau, ligne par ligne. En frappant 'U', on peut passer directement à la ligne suivante.

Les requêtes SLIG <NOM LIGNE> DE <Ti>
et SCOL <NOM COLONNE> DE <Ti>

permettent de saisir une ligne ou une colonne du tableau désigné par <Ti>, après avoir vérifié l'appartenance de la ligne ou de la colonne à ce tableau.

### RECOPIE DE TABLEAU

La requête CTAB <Ti> = <Tj>

vérifie que les tableaux Ti et Tj sont de même type, et recopie les éléments de Ti dans Tj.

Les requêtes CLIG <NOM LIGNE> DE <Ti> = <NOM LIGNE> DE <Tj>
et CCOL <NOM COLONNE> DE <Ti> = <NOM COLONNE> DE <Tj>

recopient de même une igne ou une colonne, après avoir vérifié de plus que les lignes ou colonnes citées appartiennent bien aux tableaux.

La requête

CELE <NOMLIGNE><NOMCOLONNE>DE<Ti>=<NOMLIGNE><NOM COLONNE> DE <Tj>

recopie l'élément cité de Tj dans l'élément cité de Tj, même si ces tableaux sont de types différents.

LEGENDES

Les requêtes LTAB <Ti> = <LEGENDE>
LLIG <NOM LIGNE> DE <Ti> = <LEGENDE>
LCOL <NOM COLONNE> DE <Ti> = <LEGENDE>

permettent de saisir ou modifier la légende d'un tableau complet, d'une ligne ou d'une colonne. Si la ligne ou la colonne citée n'appartient pas au tableau désigné par <Ti>, cette ligne ou cette colonne est alors générée dans ce tableau.

VISUALISATION DE TABLEAUX

La requête VTAB <Ti>

affiche à l'écran la mention

TABLEAU <NOM DU TABLEAU> : <LEGENDE>

puis pour chaque ligne du tableau la mention

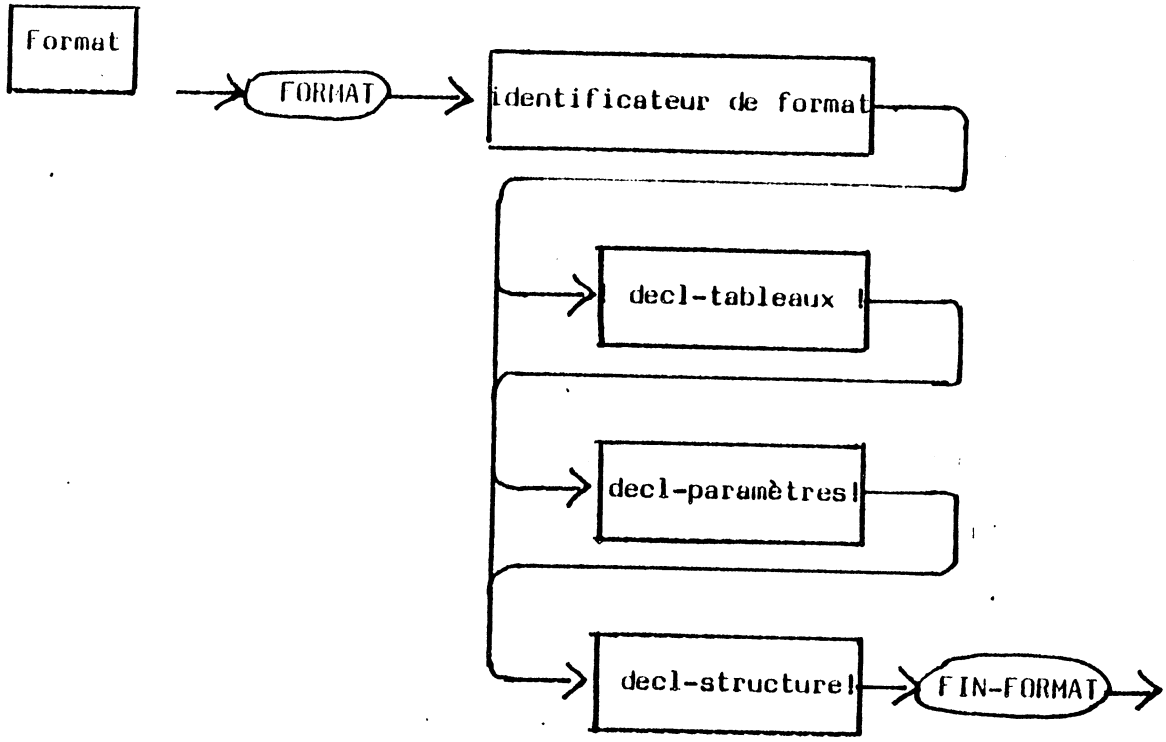
LIGNE <NOM LIGNE> : <LEGENDE>

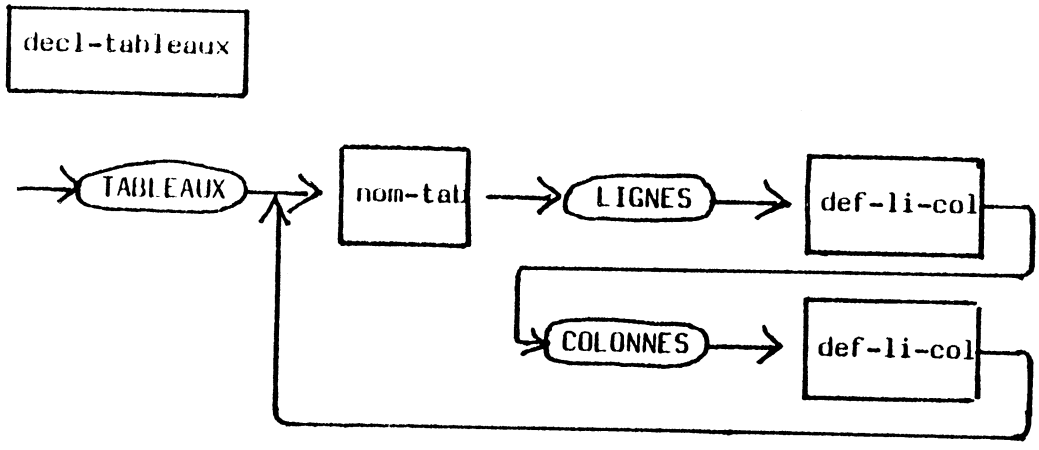
suivie de la valeur de tous les éléments de cette ligne.

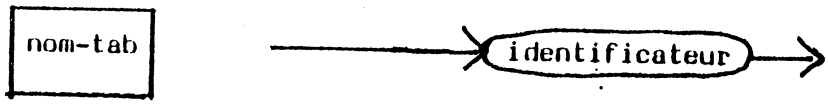
Les requêtes VLIG <NOM LIGNE> DE <Ti>
VCOL <NOM COLONNE> DE <Ti>
et VELE <NOM LIGNE> <NOM COLONNE> DE <Ti>

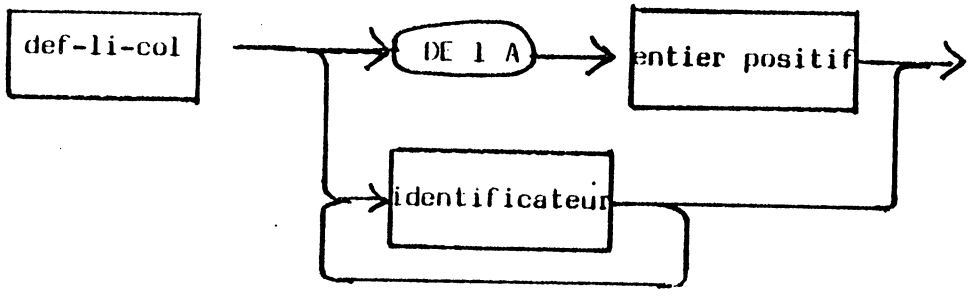
affichent de même la valeur des éléments d'une ligne ou d'une colonne, ou d'un seul élément.

3 - MAQUETTE SOLAR : SYNTAXE DES FORMATS

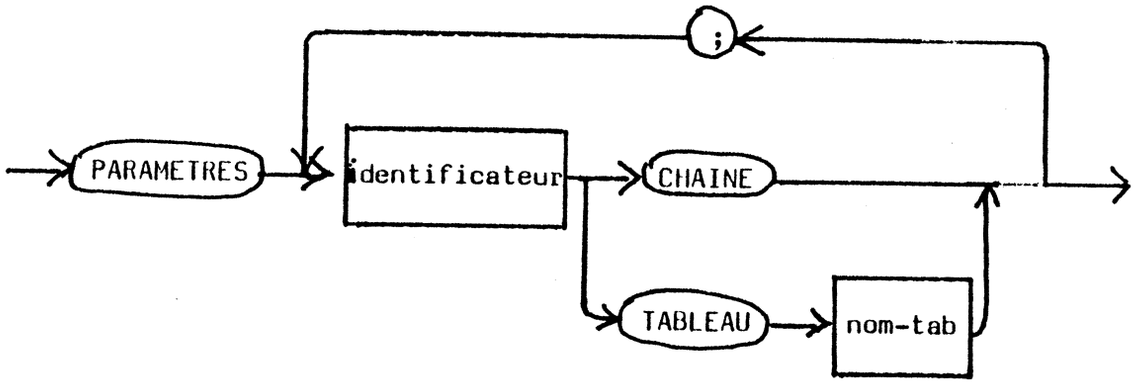


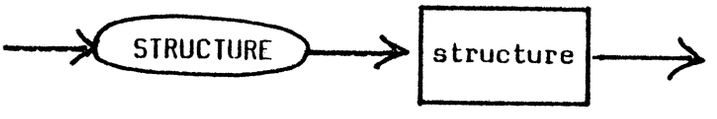



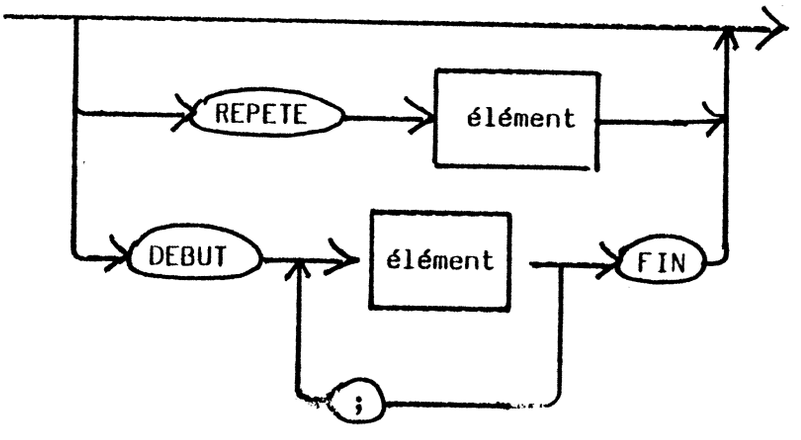
décl-paramètres



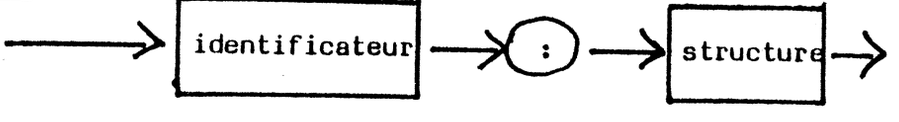

decl-structure




structure




élément




La structure est donc définie récursivement, par l'intermédiaire de "élément". Notons qu'un bloc début-fin ne peut pas être vide.

## 4 - MIDOC : IMPRESSION D'UNE STRUCTURE FORMELLE

```
type POT = ↑OT;
     OT  = record
             NOM : string[15];
             OPT : boolean;
             SUCC, ALT : POT;
           end;

procedure AFFSTRUC ( R : POT; var F : TEXT);
var S, R1 : POT;
    ID : string[15];

begin
  if R <> nil then
    begin

      (* affichage des noms d'une partie et de tous ses fils *)

      R1:=R↑.SUCC;
      if R1 <> nil then
      begin
        write(F,R↑.NOM:15,' = ');
        if R1↑.ALT = R1 then begin
                                ID:=R1↑.NOM;
                                writeln(F,'REPETE ',ID);
                             end
        else
          begin
            writeln(F,'DEBUT ');
            repeat
              ID:=R1↑.NOM;
              if R1↑.OPT then ID:=concat('%',ID);
              writeln(F,' ':17,ID);
              R1:=R1↑.ALT;
            until R1 = nil;
            writeln(F,' ':14,'FIN');
          end;
      end;
```

```
        (* recommencer pour chacun des fils *)

        R1:=R↑.SUCC;
        while R1 <> nil do
          begin
            AFFSTRUC(R1,F);
            S:=R1↑.ALT;
            if R1 = S then R1:=nil else R1:=S;
          end;
      end;
end;
```

## 5 - MIDOC: JUSTIFICATION ET COUPURE DE MOTS

Les critères retenus pour la justification à droite d'un texte sont les suivants:

- insertion d'espaces entre les mots: au plus deux espaces consécutifs, et les lignes comporteront en alternance les espaces supplémentaires à partir de la gauche et de la droite; si cette insertion ne permet pas de cadrer à droite le dernier mot de la ligne, on doit passer à la coupure du dernier mot

- la coupure de mots doit être faite sur un changement de syllabe; pour cela, on teste le mot à scinder quatre caractères par quatre caractères. Sept configurations de quatre caractères (voyelles ou consonnes) permettent la coupure. Par exemple, CVVC.. et CVCC... seront coupés après la troisième lettre ( "cou-per", "con-son-ne"), alors que VVCV... et CVCV... seront coupés après la deuxième lettre ("au-to-ma-ti-que"). Les suites de deux consonnes sont testées, certaines étant insécables: "ch" ou "gn" par exemple. Il en est de même de "qu". Si le mot comporte un tiret, la coupure se fait sur celui-ci. On ne coupe jamais un mot en laissant une lettre seule.

- si la coupure est impossible, on insère deux espaces entre chaque mot, et on renvoie le dernier mot à la ligne suivante; la ligne n'est donc pas justifiée. D'après nos expériences, ce cas ne se présente que très rarement.

En insertion:

- on commence par afficher le texte frappé sans justification. Si le mot courant ne tient pas sur la ligne, il est renvoyé à la ligne suivante (frappe dite "au kilomètre"). Lorsque l'insertion est validée, le texte est automatiquement justifié depuis le début de l'insertion jusqu'à la fin du paragraphe concerné, avec coupure de mots si nécessaire.

En cas de changement de marges, on peut obtenir la rejustification en frappant

- J : justification du paragraphe dans lequel se trouve le curseur
- nJ : justification de n paragraphes à partir de la position du

curseur

- /J : justification depuis la position du curseur jusqu'à la fin du texte.

Un changement de paragraphe s'obtient par la frappe d'un retour chariot "volontaire", c'est à dire non fourni par le système. On peut ainsi facilement protéger des séquences de lignes, afin qu'elles ne soient pas justifiées, il suffit pour cela de terminer chaque ligne par un retour chariot. Ce sera le cas des figures, par exemple.

BIBLIOGRAPHIE

(ACHU82) ACHUGBUE J., "On the line breaking problem in text formatting", Proc. of the ACM Sigplan Sigoa symposium on text manipulation, Portland (Ore), juin 1981, pp 117-122

(ALLE81) ALLEN T., NIX R., PERLIS A., "PEN, a hierarchical document editor", Proc. of the ACM Sigplan Sigoa symposium on text manipulation, Portland (Ore), juin 1981, pp 74-81

(ANDE78) ANDERSON N., BURKHARD W., "Minisequel relational data management system", in "Databases, improving usability and responsiveness", Academic Press, New York, 1978, pp 57-76

(BELT82) BELTRAN X., CHRISMENT C., "Schéma conceptuel dans les bases d'information généralisées", Séminaire Base de Données, Toulouse, nov 1982

(BENO78) BENOLIEL J., MANUGUERRA M., "Mise en page et impression automatique de textes fondées sur la reconnaissance de leur structure. Système AURORE", Thèse 3ème cycle, Nice, juin 1978

(BOGO83) BOGO G., RICHY M., VATTON I., "Un modèle de représentation des documents généralisés", Journées sur la manipulation de documents, Rennes, mai 1983, pp 212-226

(BOUC81) BOUCHET P., CHENAIS A., FEUVRE J.M., JOMIER G., KURINCK A., "Database for microcomputers: the PEPIN approach", ACM Sigmod Sigsmall, Orlando (Fla), oct 1981, Sigsmall Newsletter vol 7 no 2, pp 145-151

(BOUC83) BOUCHET P., CHENAIS A., FEUVRE J.M., JOMIER G., SZULMAN S., "PEPIN: un SGBD relationnel pour micro-ordinateurs", Journées sur la conception, l'implantation et l'utilisation des SGBD relationnels pour micro-ordinateurs, INRIA, Toulouse, fev 1983

(BOUR81) BOURELLY L., CHOURAKI E., "La représentaton et le traitement de données documentaires par le logiciel SATIN 1", in Banque d'Information dans les sciences de l'homme, Ed. Hommes et techniques, Paris 1981, pp 73-90

(BRUA82) BRUANDET M. F., "Concept notion for automatic and dynamic thesaurus updating", Itnl Conf. on Systems documentation, ACM SIGDOC SIGOA, Los Angeles, janv 1982

(CETI80) Etude CETIB-IMAG, "Les bases de données textuelles", Grenoble, mai 1980

(CHRI83) CHRISMENT C., CRAMPES J.B., ZINA M., ZURFLUH G., "Production de documents dans un contexte de base d'information généralisée", Journées sur la manipulation de documents, Rennes, mai 1983, pp 164-184

(CHUP81) CHUPIN J.C., JOLOBOFF V., "A data model for office systems", Proc. of the International Workshop on Office Information Systems, INRIA, St Maximin, 1981

(CODD70) CODD E.F., "A relational model of data for large shared data banks", CACM Vol 13 no 6, juin 1970, pp 377-387

(COUR75) COURTIN J., VOIRON J., "Introduction à l'algorithmique et aux structures de données", IUT B, Département Informatique, Grenoble, 1975

(CXP 75) "Logiciels et systèmes documentaires", Etude CXP, Les Cahiers de l'ADBS, Paris, 1975

(DATE77) DATE C.J., "An introduction to database systems", 2nd ed., Wesley, Reading (Mass), 1977

(DELO82) DELOBEL C., ADIBA M., "Bases de données et systèmes relationnels", Dunod Informatique, Paris, dec 1982

(EDEL81) EDELSON R. H., "dBASE II, a relational DBMS for CP/M", Interface Age, 60-3, aout 1981

(FLOR82) FLORY A., METZGER J.P., "Exemple d'application du modèle relationnel à des BD textuelles", Séminaire INFORSID, Toulouse, mai 1982

(FOBE82) FOBES R., "Program your own text editor; Part 1: Avoid complex commands by using instant updating", BYTE, sept 1982, pp 476-489

(GARD83) GARDARIN G., "Bases de données. Les systèmes et leurs langages", Ed. Eyrolles, Paris 1983

(GOOD81) GOOD M., "ETUDE and the folklore of user interface design", Proc. of the ACM Sigplan Sigoa symposium on text manipulation, Portland (Ore), juin 1981, pp 34-43

(GREN80) GRENIER F., KOWARSKI I., LOPEZ M., "Maquette base de données textuelles. Les descripteurs: création et manipulation", Rapport interne, août 1980

(GREN81) GRENIER F., KOWARSKI I., LOPEZ M., "Maquette base de données textuelles. Manuel d'utilisation", Rapport interne, mars 1980

(GREV75) GREVISSE M., "Le bon usage", 10ème ed., Editions Ducolot, Gembloux (Belgique), 1975

(HUNT82) HUNTER J., HALL N., "A network screen editor implementation", SOFTWARE, Practise and Experience, Vol 12, no 9, sept 1982, pp 843-856

(JOLO80) JOLOBOFF V., "Spécifications de la maquette base de données textuelles", Rapport interne, juillet 1980

(JOLO81) JOLOBOFF V., KOWARSKI I., LOPEZ M., "Projet base de données textuelles", Rapport de Recherche IMAG no 273, Grenoble, Nov 1981

(JOLO82) JOLOBOFF V., KOWARSKI I., LOPEZ M., "Un système de bases de données textuelles", Séminaire Inforsid, Toulouse, mai 1982

(KAYA81) KAYAK, Actes des journées sur la bureautique, ADI, INRIA, Paris, mars 1981

(KERN81) KERNIGHAN B., PLAUGER P., "Software tools in Pascal", Addison-Wesley, Reading (Mass), 1981

(KNUT69) KNUTH D., "The art of computer programming, Vol 1: Fundamental algorithms", Addison-Wesley, Reading (Mass), 1969

(KOWA80) KOWARSKI I., LOPEZ M., "Maquette base de données textuelles. Traitement des requêtes", Rapport interne, août 1980

(KOWA82a) KOWARSKI I., LOPEZ M., "The document concept in a database", Proc. of the ACM Sigmod conference on management of data, Orlando (Fla), juin 1982, pp 276-283

(KOWA82b) KOWARSKI I., MICHAUX C., "MIDOC: un système de gestion de documents structurés sur microordinateur", Rapport de recherche IMAG no 326, Grenoble, nov 1982

(KOWA83) KOWARSKI I., MICHAUX C., "MIDOC: a microcomputer system for the management of structured documents", Congrès IFIP, Paris, sept 1983

(LEMA83) LEMAITRE J., "Le système de gestion de base de données SOFIA et son langage de manipulation SOLANGE", Journées sur la conception, l'implantation et l'utilisation des SGBD relationnels pour micro-ordinateurs, INRIA, Toulouse, fev 1983

(MART77) MARTIN J., "Computer database organization", 2nd ed., Prentice-Hall, Englewood Cliffs (NJ), 1977

(MARY83) MARYANSKI F., "Design, implementation, and use of relational DBMS on microcomputers", Journées sur la conception, l'implantation et l'utilisation des SGBD relationnels pour micro-ordinateurs, INRIA, Toulouse, fev 1983

(MENA79) MENAGER E., CHAPELIER J.L., "Bureautique: les systèmes multipostes de traitement de textes", Etude no. 58 du CXP, Paris, 1979

(MERR83) MERRET T., "A comparison of relational database systems for microcomputers", Journées sur la conception, l'implantation et l'utilisation des SGBD relationnels pour micro-ordinateurs, INRIA, Toulouse, fev 1983

(MEYE78) MEYER B., BAUDOUIN C., "Méthodes de programmation", Coll. de la Direction des Etudes et Recherches EDF, Eyrolles, Paris, 1978

(MINO80) MINOT R., "Maquette base de données textuelles. Spécification et définition de l'AST", Rapport interne, dec 1980

(MIRA83) MIRANDA S., VINTROU L., "MINIDOC: un système documentaire construit à partir d'un SGBD relationnel", Rapport de recherche, jan 1983

(MIST75) MISTRAL, Recherche documentaire, Manuel de présentation, CII, 1975

(NAFF80) NAFFAH N., "KAYAK: a national project for office automation", International conference for data processing, Berlin, oct 1980

(NAFF81) NAFFAH N., "Office information systems", International workshop, St Maximin (Var), oct 1981

(NATA81) NATALE R., "RTFILE: Data management for small DEC systems", Proc. ACM SIGMOD/SIGSMALL Workshop, pg 19, oct 1981

(NGUY82) NGUYEN G.T., FERNANDEZ F., FERRAT L., LEE Y.J., "MICROBE: Manuel de référence", Laboratoire IMAG, janvier 1982

(PAIL83) PAILLARD J.P., "Installation paramétrée du système UCSD", à paraître

(PERR81) PERRET R., "Maquette base de données textuelles. Spécification du processeur d'édition", Rapport interne, jan 1981

(REBS83) REBSAMEN J. et al, "LIDAS - The database system for the personal computer LILITH", Journées sur la concepton, l'implémentaton et l'utilisation de SGBD relationnels sur micro-ordinateurs, INRIA, Toulouse, fev 1983

(REID80) REID B., "SCRIBE, a document specification language and its compiler", Carnegie-Mellon University, Dept. of Computer Science, oct 1980

(SALT62) SALTON G., "Manipulation of trees in information retrieval", CACM Vol 5, 1962, pp 103-114

(SIBE83) SIBERTIN-BLANC C., "L'utilisation des SGBD relationnels en burotique. L'exemple du projet KAYAK", Journées sur la conception,l'implantation et l'utilisation des SGBD relationnels sur micro-ordinateurs, INRIA, Toulouse, fev 1983

(SOCR79) SOCRATE: Manuel de référence, ECA Automation, Grenoble 1979

(STRO81) STROMFORS O., JONESJO L., "The implementation and experiences of a structure oriented text editor", Proc. of the ACM Sigplan Sigoa symposium on text manipulation, Portland (Ore), juin 1981, pp 22-27

(TEXT79) TEXTO, Manuel d'utilisation, Chem-data SARL, Lyon, 1979

(TSIC80) TSICHRITZIS D. (Ed), "A panaché of DBMS ideas III", Technical report CSRG-111, Computer Systems Research Group, University of Toronto, avril 1980

(TSIC81) TSICHRITZIS D., "Integrating database and message systems", Proc. of the VLDB, Cannes, sept 1981, pp 356-362

(UCSD81) UCSD p-SYSTEM and UCSD PASCAL, Version IV.0, Users manual and Installation guide, Softech Microsystems, San Diego (Cal), 1981

(WIRT81) WIRTH N., "LILITH: a personal computer for the software engineer", Proc. of the 5th International Conference on software engineering, San Diego (Cal), mars 1981, pp 2-15

(WORD80) WORDSTAR: User's guide for release 2.1, Micropro Int. Corp., 1980

(ZLOO82) ZLOOF M.M., "Office by example: a business language that unifies data and word processing and electronic mail", IBM Systems Journal, vol 21, no 3, 1982

# AUTORISATION DE SOUTENANCE

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

VU le rapport de présentation de Monsieur J. COURTIN, Professeur

**Madame KOWARSKI Irène**

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

Fait à Grenoble, le 15 juin 1983

Le Président de l'I.N.P.-G

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

*P.O. le Vice-Président,*