# Étude et évaluation d'une architecture de système pour les bases de données généralisées

Marc Burnier

# THESE

*présentée à*

**l' Université Scientifique et Médicale de Grenoble**

*et à*

**l' Institut National Polytechnique de Grenoble**

*pour obtenir le grade de*

**DOCTEUR DE 3ème CYCLE**

**«Informatique»**

*par*

**Marc BURNIER**

## ETUDE ET EVALUATION D'UNE ARCHITECTURE DE SYSTEME POUR LES BASES DE DONNEES GENERALISEES.

**Thèse soutenue le 21 septembre 1984 devant la commission d'examen.**

| | |
|---|---|
| C. DELOBEL | Président |
| M. ADIBA | Examinateurs |
| G. VIGLIANO | |
| G. MAZARÉ | |

**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 |

# ECOLE NATIONALE SUPERIEURE DES MINES DE SAINT-ETIENNE

**Directeur** : **Monsieur M. MERMET**
**Directeur des Etudes et de la formation** : **Monsieur J. LEVASSEUR**
**Directeur des recherches** : **Monsieur J. LEVY**
**Secrétaire Général** : **Mademoiselle M. CLERGUE**

## Professeurs de 1ère Catégorie

| | | |
|---|---|---|
| COINDE | Alexandre | Gestion |
| GOUX | Claude | Métallurgie |
| LEVY | Jacques | Métallurgie |
| LOWYS | Jean-Pierre | Physique |
| MATHON | Albert | Gestion |
| RIEU | Jean | Mécanique - Résistance des matériaux |
| SOUSTELLE | Michel | Chimie |
| FORMERY | Philippe | Mathématiques Appliquées |

## Professeurs de 2ème catégorie

| | | |
|---|---|---|
| HABIB | Michel | Informatique |
| PERRIN | Michel | Géologie |
| VERCHERY | Georges | Matériaux |
| TOUCHARD | Bernard | Physique Industrielle |

## Directeur de recherche

| | | |
|---|---|---|
| LESBATS | Pierre | Métallurgie |

## Maîtres de recherche

| | | |
|---|---|---|
| BISCONDI | Michel | Métallurgie |
| DAVOINE | Philippe | Géologie |
| FOURDEUX | Angeline | Métallurgie |
| KOBYLANSKI | André | Métallurgie |
| LALAUZE | René | Chimie |
| LANCELOT | Francis | Chimie |
| LE COZE | Jean | Métallurgie |
| THEVENOT | François | Chimie |
| TRAN MINH | Canh | Chimie |

## Personnalités habilitées à diriger des travaux de recherche

| | | |
|---|---|---|
| DRIVER | Julian | Métallurgie |
| GUILHOT | Bernard | Chimie |
| THOMAS | Gérard | Chimie |

## Professeur à l'UER de Sciences de Saint-Etienne

| | | |
|---|---|---|
| VERGNAUD | Jean-Maurice | Chimie des Matériaux & chimie industrielle |

**********

A mes Parents,

Je tiens à remercier

- **CLAUDE DELOBEL**, professeur à l'USMG, de l'honneur qu'il m'a fait en acceptant de présider le jury de ma thèse et d'avoir su me passionner, dès la maitrise, au domaine des bases de données,

- **MICHEL ADIBA**, professeur à l'USMG, de m'avoir accueilli dans son équipe, encouragé tout au long de mon travail et encore plus spécialement dans les derniers moments de rédaction ainsi que de sa participation au jury,

- **GEORGES VIGLIANO**, directeur recherche et développement du département produit de la Société SYSECA, d'avoir assuré l'encadrement de la réalisation pratique de ce travail, pour ses encouragements et sa présence au jury,

- **GUY MAZARE**, professeur à l'ENSIMAG, pour son aide apportée tout au long de la première phase de ma thèse ainsi que de sa participation au jury de soutenance.

Je tiens également à remercier Monsieur **GEORGES BEAUME**, directeur de l'agence RHONE-ALPES de la Société SYSECA, de m'avoir accueilli au sein de sa société durant la plus grande partie de mon travail et d'avoir mis à ma disposition tous les moyens nécessaires au bon déroulement de la partie expérimentation.

Je suis également reconnaissant à Monsieur **ROBERT MORIN**, directeur technique du département produit de la Société SYSECA, qui a grandement facilité ma familiarisation avec le système SOCRATE/CLIO.

J'adresse également mes remerciements à Messieurs **GERARD DENIEL** (RUNGIS, FRANCE) et **JAN BUELENS** (SWINDON, GRANDE-BRETAGNE) de la société INTEL pour la mise à disposition de la Machine Bases de Données iDBP ainsi que pour les nombreux échanges qui ont eu lieu durant la période d'expérimentation.

Je ne voudrais pas oublier l'ensemble des membres des équipes MICROBE et TIGRE ainsi que ceux de la société SYSECA avec qui les échanges ont été tout autant amicaux que fructueux.

Je ne pourrais conclure sans adresser tous mes profonds remerciements à **MARTINE**, mon épouse, pour toutes la patience et la compréhension dont elle a fait preuve tout au long de ces années afin que je puisse présenter ma thèse.

# TABLE DES MATIERES

# PRESENTATION

## 1 DESCRIPTION DE LA SITUATION

L'informatique apparait aux yeux d'une grande partie de la communauté des non spécialistes comme un outil consacré exclusivement à la résolution des problèmes de gestion des entreprises (comptabilité, gestion du personnel, gestion de production ...). Les progrès réalisés aussi bien au point de vue matériel que logiciel ont poussé l'ordinateur à prendre place dans des domaines aussi bien traditionnels telles la physique, la mécanique, que de pointe, le traitement de la parole et de l'image, la conception de circuits. Ainsi nous en sommes au stade où l'on ne peut concevoir des ordinateurs sans l'assistance d'autres ordinateurs et de logiciels spécialisés.

Ces nouveaux domaines sont d'une complexité telle que l'on doit être en mesure de leur offrir des outils en rapport avec leurs exigences. Il est vrai qu'il est plus aisé de trouver un nouveau domaine d'application que l'outil informatique associé. Il en résulte que pour ne pas paraître un frein, on propose à ces nouveaux utilisateurs les mêmes solutions que celles retenues pour les autres secteurs. Indéniablement nous convergeons vers une inadéquation des systèmes proposés. Ce retard se situe aussi bien au niveau

du matériel qu'à celui de la gestion des données manipulées qui apparaissent comme de plus en plus volumineuses et complexes au point de vue structure.

Le but de cette thèse a été, dans un premier temps, de mettre en évidence les exigences de ces nouveaux domaines d'applications quant à la gestion des données manipulées. De là, nous avons extrait les problèmes de performances d'accès inhérents à la fois aux architectures de systèmes existants et à la nature de ces nouvelles informations. Nous avons fait le panorama des nouvelles configurations proposées et des prototypes réalisés en vue de pallier les insuffisances des outils classiques. Nous avons alors effectué un comparatif entre un système traditionnel et un des rares prototypes, d'accélération de recherche de l'information, actuellement commercialisé. Enfin, une nouvelle architecture de système pour les bases de données généralisées a été esquissée comme la synthèse des besoins de ces nouveaux domaines d'utilisation et des solutions proposées jusqu'à aujourd'hui.

## 2 DEROULEMENT DES TRAVAUX

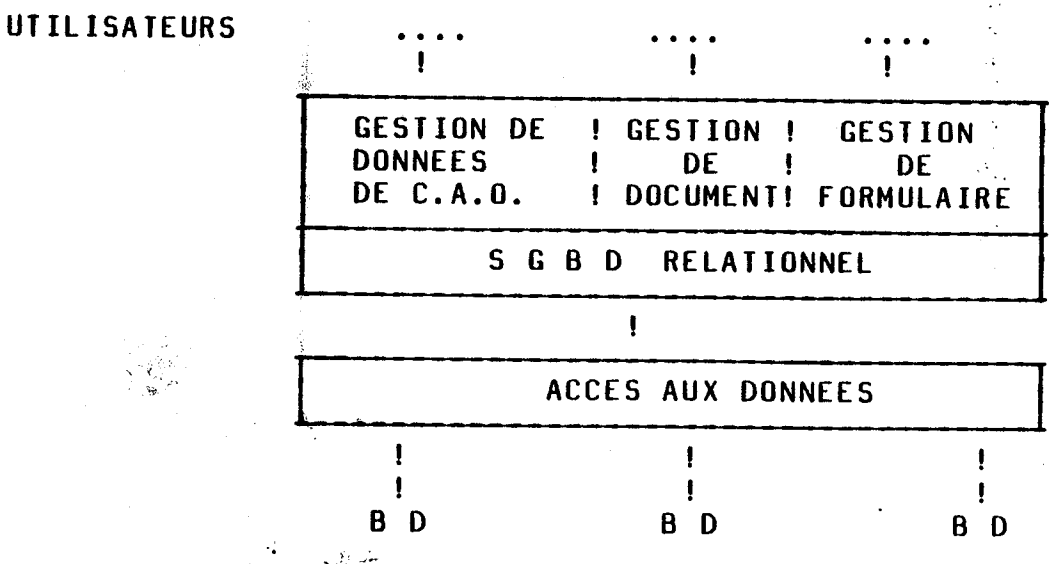
Nous avons limité volontairement notre domaine d'investigation aux secteurs du traitement de la parole et de l'image et à celui plus général de la Conception Assistée par Ordinateur (C.A.O.). Le son et l'image présentent un intérêt double puisqu'ils sont à la fois le coeur de recherches fondamentales intensives mais aussi les composants que l'on voudrait intégrer comme éléments à part entière de nombreux systèmes plus généraux: système de gestion de documents intégrant l'image <PALA 84> <VELE 84>. Derrière le terme de C.A.O. se profile une population d'usagers diversifiée. On parle aussi bien de conception assistée par ordinateur dans le secteur de la chaussure que dans celui des circuits intégrés sans omettre les domaines de la mécanique et de

l'architecture. Nous n'aborderons que la conception architecturale et celle des V.L.S.I. ou circuits à très haute intégration, parce que les exigences rattachées à la C.A.O. se retrouvent très bien au travers de ces deux exemples.

Nous décrirons chacune de ces spécialités et listerons leurs exigences qui se distinguent de celles des applications plus classiques de gestion. Nous ferons référence aux besoins liés à la gestion de l'information.

Les systèmes de gestion de fichiers ont révélé de nombreuses insuffisances dans le domaine de la gestion. De là sont nés les premiers Systèmes de Gestion de Bases de Données (SGBD). Nous expliquerons en quoi ces SGBD constituent aussi une première réponse à ces nouvelles exigences mais aussi pourquoi ils ne donnent pas entière satisfaction.

De là plusieurs centres de recherche se sont dessinés et ont été pris en compte dans le cadre du projet de Traitement d'Informations Généralisées REparti <TIGRE 1 83>. La gestion des données du projet TIGRE peut être schématisée de la façon suivante:



La prise en compte des insuffisances du modèle relationnel, qui a été retenu comme modèle de base, sera réalisée par la spécification de nouveaux modèles de données rattachés à chacune des nouvelles aplications <TIGRE 9 83> <TIGRE 13 84> <PALA 84> <VELE 84>. Cette superposition de niveaux et les côtés volumineux et complexes des données manipulées font apparaître des problèmes réels de performances d'accès à l'information. Cet aspect sera pris en considération, toujours dans le cadre du projet TIGRE, à travers la notion de machine bases de données. Pour notre part, nous avons porté l'intérêt de nos investigations sur l'aspect performances et architecture des SGBD.

Une première forme de proposition, liée à la résolution des problèmes de coûts d'accès aux données, est apparue à travers les machines bases de données. Peu de prototypes ont vu le jour et encore moins de machines ont pris place sur le marché. On les présente comme LA solution à ce problème bien spécifique. Avant d'investir dans la spécification d'une n-ième machine, il nous parait opportun d'évaluer de manière concrète ce qu'une machine bases de données existante pouvait apporter à un SGBD exécuté sur une architecture traditionnelle. Cette approche est encore très controversée et il n'y a eu, à notre connaissance, que très peu d'études concrètes sur l'apport d'une machine bases de données aux performances globales d'un SGBD <DEWBITT 83> <DEWBOR 84>. Ce comparatif a été basé sur la réalisation d'un test entre la machine bases de données d'INTEL, l'iDBP (iNTEL DATA BASE PROCESSOR), et le SGBD SOCRATE/CLIO de la société SYSECA.

Les résultats de ce test nous ont incités à investir dans une nouvelle architecture de plus grande envergure que celle proposée par l'utilisation d'un dorsal. Nous esquisserons les grands traits de ce nouveau type de configuration associée aux SGBD.

## 3 PRESENTATION DE LA THESE

Tout au long du chapître 1, nous décrivons chacune des applications abordées et nous mettons en évidence leurs exigences et la façon dont elles ont été satisfaites, entièrement ou partiellement, par les systèmes de gestion de bases de données actuels.

Le chapître 2 développe le pourquoi de la conception de SGBD sur de nouvelles architectures et fera un panorama sur les travaux des années antérieures, présentes et sur les tendances actuelles pour les machines bases de données de demain.

Avec le chapître 3, nous abordons la phase d'expérimentation et décrivons, dans un premier temps, le contexte matériel et logiciel de notre réalisation. De là, nous développons la traduction du modèle de données de SOCRATE/CLIO en termes du modèle relationnel tel qu'il a été étendu et développé sur l'iDBP. On évoque leur compatibilité, l'amélioration de certains aspects et enfin, les points où la traduction en relationnel nous a paru plus couteuse. Nous décrivons ensuite les différentes requêtes exécutées pour le test. Ces requêtes ne sont pas totalement aléatoires et tentent de recouvrir l'ensemble des accès possibles que l'on peut effectuer sur de tels systèmes. Enfin, nous donnons nos résultats et les analysons eu égard aux performances obtenues sur une configuration classique. Nous clôturons ce chapître par la spécification d'une méthodologie portant sur les comparatifs de systèmes de gestion de bases de données.

Grâce aux résultats de notre expérimentation, nous pouvons alors proposer, dans le chapître 4, une nouvelle architecture de SGBD associée à ces nouveaux domaines d'applications.

# CHAPITRE 1

# EVOLUTION DES APPLICATIONS

# DES BASES DE DONNEES ET

# DES SYSTEMES DE GESTION DE BASES DE DONNEES

## 1.0 PREAMBULE

Ce chapître est consacré à la description de l'évolution des applications des bases de données et des systèmes de gestion de bases de données. Nous nous attachons dans un premier temps à la Conception Assistée par Ordinateur appliquée à l'architecture et aux VLSI. Après avoir défini ce qu'est un outil de C.A.O., nous ferons la synthèse des besoins de leurs utilisateurs et le point sur la situation actuelle en matière de système intégré de C.A.O.. Il sera temps alors de décrire en quoi l'approche base de données peut être une solution aux problèmes de gestion des données de cette nouvelle classe d'utilisateurs de l'outil informatique et sur quels points elle présente certaines insuffisances. Nous reprendrons chacune des particularités des SGBD qui nous paraissent être un "plus" vis à vis des outils classiques et en fonction précisément de ces nouvelles exigences.

A l'issue de la description de la situation en C.A.O., nous abordons les applications de traitement de la parole et de la synthèse d'images. Ces deux applications n'ont pas été dissociées volontairement. Nous reprendrons le même plan que pour la partie C.A.O..

## 1.1 APPLICATIONS C.A.O.

### 1.1.1 PRESENTATION

La Conception Assistée par Ordinateur correspond à une utilisation récente de l'outil informatique. L'idée de base est simple: faire coopérer de la façon la plus efficace un homme et une machine pour résoudre des problèmes.

Elle pénètre actuellement de plus en plus de domaines techniques et ne cesse de s'ouvrir vers de nouvelles applications. On l'utilise fréquemment en construction mécanique. Elle commence à faire son entrée en conception de circuits intégrés et on la voit apparaître dans l'industrie de la chaussure.

Dans le cadre de cette étude, il est évident que nous ne pouvons nous attacher à explorer chacune des industries de fabrication au sein de laquelle les concepteurs font appel à l'outil informatique dans leur tâche journalière, tant la multiplicité des contextes est grande.

Par conséquent, nous ne retiendrons que deux domaines où la C.A.O. est susceptible d'apporter beaucoup à l'homme: la conception architecturale et la conception de circuits intégrés de type VLSI.

### 1.1.2 BUTS RECHERCHES PAR L'UTILISATION D'UN SYSTEME DE C.A.O.

Les exigences des futurs utilisateurs de systèmes de C.A.O. se retrouvent assez bien lorsque l'on passe du domaine de l'architecture au domaine des circuits intégrés. Elles seront de nature différente selon que l'on se place au niveau des besoins de l'utilisateur final ou au niveau des besoins informatiques indispensables à la réalisation de ces outils

spécifiques.

Les désirs qui reviennent le plus fréquemment sont les suivants:

- réduire le temps entre la description initiale de l'objet (cahier des charges) et sa version finale (ensemble des plans décrivant l'objet conçu)

- permettre au concepteur d'explorer efficacement différentes alternatives. En effet, tout système informatique est d'autant plus apprécié que le nombre de ses paramètres est grand. Ainsi, pour l'élaboration d'un objet complexe, la qualité du produit final sera fonction du nombre d'alternatives que le concepteur aura pu explorer à chacun des stades de la conception. Ces différentes versions devront être prises en compte par le système et traitées indifféremment les unes des autres.

- réduire la complexité de la conception. Selon <TRIM 81>, pour concevoir un circuit intégré de 10 millions de transistors, 6000 hommes/année, environ, seraient nécessaires.

- accroître les tests, les simulations et réduire leur durée. A l'issue de chaque étape du processus de conception, le concepteur doit pouvoir être en mesure de certifier que son objet est correct vis à vis à la fois du cahier des charges et des contraintes d'ordres technologiques. Par contre il est vrai qu'un programme de test ou de simulation ne peut garantir la perfection de l'objet. Ainsi <TRIM 81> part de l'hypothèse de DIJKSTRA qui affirme que le test peut être utilisé pour détecter d'éventuelles erreurs et non pour attester de l'infaillibilité de l'objet.

- les utilisateurs de systèmes de C.A.O. faisant appel à une gestion classique des données sont quasi unanimes pour déclarer que ces outils réalisent de façon insuffisante chacun des aspects suivant:

  * la sécurité des données (comme les problèmes de reprises après pannes)

  * la confidentialité (protection de l'information)

  * l'intégrité de l'information (contraintes d'intégrité, accès concurrents)

  * l'interface homme-machine

- un processus de conception se découpe, très souvent, en étapes (logique, électrique et implantée pour la conception de VLSI). Les objets à élaborer sont de nature complexe, d'où la nécessité de décomposer cet objet en éléments atomiques. De plus, on ne pourra toujours garantir une correspondance biunivoque entre chaque élément décomposé de chaque phase. Ainsi nous en arrivons à une décomposition hiérarchique de l'objet qui se répétera à chacune des étapes de l'élaboration de l'objet final. Par conséquent, le système de gestion des données devra permettre une définition aisée d'une hiérarchie et une manipulation efficace de chacune des parties de cette structure arborescente.

- l'utilisateur devra, avec un coût minimal, pouvoir intégrer à tout moment de nouveaux programmes de C.A.O.

- l'utilisateur préfèrerait à une configuration centralisée un système réparti.

- en C.A.O. il n'est pas rare de rencontrer des ingénieurs travaillant plusieurs heures voire plusieurs

jours sur les mêmes données, les opérations exécutées étant de type mise à jour. Dans un souci de cohérence on ne peut autoriser d'autres utilisateurs à manipuler les mêmes occurences objets. Le choix de la taille de l'objet à verrouiller est un problème fondamental dans la réalisation d'un SGBD multi-usagers. Plus cet objet est petit et plus la concurrence pourra être grande mais plus lourde sera la tâche du système. A l'inverse, si les verrous ne peuvent être placés que sur de gros objets, cela va diminuer le nombre de transactions qui pourraient accéder concurremment à des composants plus élémentaires. Ces situations d'interblocage sont acceptables tant qu'elles sont de l'ordre de la seconde mais à cette échelle on ne peut retenir de telles solutions.

- le système devra être portable.

### 1.1.3 DEFINITION DE L'OUTIL DE C.A.O.

D'importants efforts ont été accomplis aussi bien dans le monde industriel qu'universitaire afin de développer de véritables systèmes de C.A.O.. En effet, pendant trop longtemps et encore maintenant, on assimile les outils de C.A.O. à des outils purement graphiques (CALMA pour la représentation des masques en C.A.O. de VLSI,...).

Ces logiciels graphiques sont nés du besoin de stocker des dessins et avec le temps, quelques opérateurs (rotation, translation, homothétie) ont été élaborés, la phase de conception demeurant toujours le résultat du travail du technicien. Malgré tout, les techniques graphiques interactives seront une des composantes essentielles des futurs systèmes de Conception Assistée par Ordinateur.

Signalons en tout premier lieu que les techniciens désirent faire de la conception assistée et non de la conception

automatisée. Ceci impliquera, au niveau de l'outil proposé, de laisser à son utilisateur un degré de liberté suffisant de telle sorte qu'il puisse à tout moment modifier l'objet conçu.

On parle indifféremment, mais à tort, de programmes de C.A.O. et de systèmes de C.A.O. . Avant de poursuivre, il serait bon de mettre en évidence la différence qui existe entre ces deux champs d'investigations.

Un programme de C.A.O. réalise une fonction technique précise et sa réalisation est uniquement entre les mains du spécialiste. Ces programmes accomplissent une action ponctuelle et déjà à ce stade, on peut voir apparaître les problèmes d'enchainement de ces programmes quant aux transferts de données entre deux unités de ce type <DAVID 81>.

Par contre, derrière la notion de système de C.A.O., on voit apparaître un "tout", constitué à la fois de modules de gestion et de manipulation de données ainsi que d'outils de véritable conception qui seront les programmes de C.A.O. précédemment définis <FOISSEAU 82>.

Pour la phase de modélisation, de description, de gestion et de représentation graphique de l'information, il faudra faire appel à des spécialistes informaticiens alors que les programmes de conception seront réalisés par des spécialistes du domaine pour lequel est développé le système.

### 1.1.4 OU EN SONT LES OUTILS DE C.A.O. ?

La jeunesse de ce domaine a conduit les concepteurs de tels systèmes à s'intéresser, dans un premier temps, davantage aux outils d'aide à la conception qu'à la gestion des données manipulées tout au long du processus de conception. Il semblerait, à juste titre, que les travaux actuels visent à ne pas adapter le problème aux produits connus mais plutôt

à développer de nouveaux outils répondant aux exigences spécifiques de la C.A.O. <FOISSEAU 82>.

Avant de développer les tendances nouvelles en matière de C.A.O., il serait bon de faire un historique de cet outil.

Les premiers systèmes étaient réduits à un ensemble de programmes autonomes, c'est à dire totalement indépendants les uns des autres au point de vue modélisation de l'application et structuration des données. Ces programmes étaient, à l'origine, très spécifiques aux problèmes traités, c'est à dire étroitement dépendants des modélisations effectuées et donc, parfois, de la vue de son créateur sur le problème à résoudre. Nous étions bien à l'époque de l'outil créé et utilisé uniquement par le spécialiste. On ne cherchait pas à définir des programmes de C.A.O. à mettre à la portée de tous. Cette approche est caractéristique d'une époque où l'utilisation de l'ordinateur était essentiellemnt réservée aux traitements numériques.

L'intérêt de ces programmes ne pouvait être que ponctuel car ils étaient difficilement intégrables à un véritable système de C.A.O.. Ensuite le concepteur a voulu enchaîner ces programmes, un reformattage voire une restructuration complète des données devenait en général indispensable. Ces enchainements nécessitaient l'écriture de nouveaux programmes de transition qui très souvent étaient sources d'erreurs, devenaient de plus en plus, pour les concepteurs, une tâche fastidieuse et contraignante.

On peut imaginer très facilement la charge supplémentaire dûe à une insertion d'un ou plusieurs nouveaux programmes. A l'inverse, il se pouvait très bien que le concepteur eu désiré "sauter" un ou plusieurs modules. De nouveau, il lui aurait été nécessaire de générer un reformattage de ses données. Il est vrai que ces programmes de transitions pouvaient déjà être réalisés par d'autres concepteurs.

En résumé, chaque système conçu autour d'un enchainement de programmes autonomes a un aspect figé qui devient vite contraignant. Il est évident que les concepteurs de tels systèmes auraient voulu élaborer des outils plus souples, mieux adaptés aux différents processus de conception dont ils avaient la charge de simplifier le mécanisme.

Détaillons le contenu d'un programme de C.A.O. afin de localiser la ou les fonctions à revoir dans le but de concevoir un véritable système intégré. L'analyse informatique de ces programmes fait apparaître qu'ils assurent quatre fonctions essentielles:

- une fonction de calcul via un ensemble de sous-programmes

- une aide au dessin et au dialogue homme/machine

- une fonction de stockage et de gestion des données utilisées par ces sous-programmes

- une fonction de description des données du problème étudié

Si la première fonction restera, pour toujours, à la charge du spécialiste du domaine couvert, les trois dernières fonctions sont à l'évidence de nature purement informatique et peuvent donc être étudiées par des informaticiens qui sont sensés, d'une part, mieux connaître les outils actuels dédiés au traitement de l'information et, d'autre part, être prêts à proposer de nouvelles méthodes répondant aux besoins nouveaux des utilisateurs. Celles-ci pourraient être réalisées sans se préoccuper du domaine d'application du système de C.A.O. et on aborde ainsi les systèmes généraux.

## 1.1.5 NOUVELLE APPROCHE DE LA C.A.O.

Cette nouvelle approche résulte de l'analyse précédente en proposant la notion de système. Il ne s'agit plus de faire des logiciels spécifiques mais d'offrir des outils permettant de constituer un cadre de construction de systèmes qui soient:

- modulaires afin de pouvoir offrir un outil pour chaque niveau du processus de conception

- extensibles. Il ne faudra plus que le seul fait d'ajouter de nouveaux programmes crée une charge de travail supplémentaire au niveau de la gestion des données.

- interactifs. Le concepteur veut contrôler de bout en bout la conception du nouveau produit et pouvoir y apporter toutes modifications à tout moment.

L'informaticien concentrera donc ses efforts sur les trois pôles suivants:

- modélisation et description des informations

- gestion de l'information

- représentation graphique de l'information

Il va de soi que les solutions apportées à ces exigences sont le fruit d'investigations développées sur de nombreuses années et font appel à des notions nouvelles du traitement de l'information.

### 1.1.5.1 Les bases de données

Les bases de données ont permis de faire un premier pas vers une meilleure gestion de l'information. Nous allons tenter de définir l'apport supplémentaire des Systèmes de Gestion de Bases de Données (ou SGBD) indépendamment de leurs modéles de données respectifs.

#### 1.1.5.1.1 Centralisation, coordination et diffusion de l'information

Cet aspect est à l'origine de l'utilisation des bases de données aussi bien dans les nouvelles applications (CAO, traitement de la parole,...) que dans les secteurs classiques de gestion de l'information (gestion, économie,...).

Le fait de partager l'information est le reflet d'une évolution dans l'utilisation des systèmes informatiques. Historiquement, chaque nouvelle application engendrait ses propres fichiers et programmes <DELADI 83>. Tant que les utilisateurs se trouvaient isolés au point de vue partage de l'information, les outils classiques leur convenaient. Mais dès l'instant où ils ont été intégrés dans un contexte unifié, les outils qui leur avaient été proposés jusqu'à présent ne répondaient plus à leurs nouvelles exigences.

Un premier effort a été réalisé dans le sens d'une meilleure utilisation de l'outil informatique avec l'apport de la notion de fichiers partageables entre utilisateurs. Malgré cela, il était encore difficile à l'utilisateur de localiser les éléments d'information entrant en compte dans son application et déjà existant dans un fichier d'un autre utilisateur. Il parait opportun de relever le fait que la notion de propriété de fichier était encore fortement marquée au cours de cette étape de l'évolution de la représen-

tation de l'information. Cet état de fait engendrait encore trop souvent la duplication des enregistrements.

L'approche base de données propose à travers le schéma conceptuel de la base, une vision globale de l'application. Pour chaque nouvelle application, nous définirons plusieurs groupes d'utilisateurs qui n'auront pas la nécessité de manipuler l'ensemble de la base. A chacun d'entre eux, nous associerons une vue réduite de la base: un schéma externe.

Cette centralisation permet de rationaliser à la fois la formalisation de l'environnement de l'application mais aussi les efforts des utilisateurs. Si la recherche informatique investit beaucoup dans le domaine des performance de ces systèmes, c'est aussi dans un souci d'offrir un outil puissant et plus proche de la façon de travailler de chacun des utilisateurs.

1.1.5.1.2 Indépendance physique et logique

Une base de données ne peut être utilisée en tant que telle sans l'adjonction d'un système de gestion. Le SGBD offrira entre autre un langage de définition des données ainsi qu'un langage de manipulation des données. Ces deux langages seront totalement dépendants du modèle de données utilisé.

Ces modèles sont au nombre de trois, Hiérarchique, Réseau et Relationnel <DELADI 83> et permettent de formaliser, selon de nouveaux concepts, la structuration des fichiers classiques. La phase de modélisation peut être assimilée à un processus d'arrangement des données, utilisées par un groupe de personnes, qui reflètera un certain univers et où les technologies informatiques de stockage et de méthodes d'accès ne doivent pas apparaître. Cette étape dans la vie d'une B.D. revêt toute son importance dans la mesure où n'étant pas mise en oeuvre à bon escient, elle influencera

à la fois les performances du système et l'intégrité des données.

Dés l'instant où l'on s'attache à modéliser un univers, il faudrait pouvoir faire abstraction de l'ensemble des traitements que l'on désire effectuer sur cette application (modélisation descendante). S'il n'en est pas ainsi, il va de soi que l'on privilégiera, inconsciemment, certaines opérations par rapport à d'autres, connues dès l'initialisation ou nécessitées par l'apparition de nouveaux besoins (modélisation ascendante).

A priori il est très difficile de comparer ces deux approches si l'on considère qu'à chaque nouvelle modélisation on se retrouve devant un environnement, un passé et des besoins très différents. Par rapport aux différents schémas que l'on peut produire pour une même base, quelle est la part de diversité apportée par chacun des trois modèles?

Un modèle de données dont le formalisme de description des données ne tient nullement compte de l'organisation physique de celles-ci, tel que le modèle relationnel, limitera notablement cette diversité. Un tel outil fera abstraction des contraintes et des souplesses de l'informatique et sera totalement orienté vers la représentation du monde à modéliser. cette indépendance physique sera d'autant plus appréciée dans une modélisation descendante où à priori on ne peut, par exemple, prévoir les critères de recherche d'informations. Ainsi à tout moment on pourra déclarer de nouvelles tables d'index sans modifier les programmes existant afin qu'ils profitent de ces nouveaux dictionnaires.

L'indépendance logique vise à pouvoir modifier le schéma conceptuel de la base, par exemple, en ajoutant de nouvelles classes d'objets ou bien de nouvelles associations entre classes d'objets <DELADI 83>. Si une base est un ensemble structuré de données, elle ne doit pas pour autant

être figée quant à la définition de son schéma conceptuel. En C.A.O., si l'on connait très bien les différentes étapes du processus de conception, l'utilisateur doit être en mesure, et ceci à tout moment, de pouvoir introduire de l'information complémentaire dans un souci de raffinement de l'objet à concevoir.

1.1.5.1.3 Non redondance de l'information

Dés l'apparition du modèle réseau, les concepteurs de SGBD et/ou de modèles de données ont posé la non redondance des données comme une condition sine qua non. Cet état de fait est le fruit de plusieurs exigences.

On ne peut pas, en effet, admettre qu'un seul et même objet puisse être décrit en plusieurs endroits dans un souci de:

- cohérence des données: pour éviter d'avoir à répéter les mises à jour d'un même objet sous prétexte qu'on a plusieurs descriptions de celui-ci.

- protection des données: il est toujours plus simple de protéger un objet qui est décrit de façon unique.

- gain de place: dés que l'on manipule des milliers d'objets (architecture, conception de circuits intégrés) on ne peut se permettre de dupliquer ces éléments si l'on veut conserver un degré de performance acceptable pour l'utilisateur.

Cette non redondance de l'information est obtenue par une modélisation plus rigoureuse de l'application et, également, par une utilisation plus poussée de la notion d'association entre éléments. Cette association sera explicite dans le modèle réseau (inverse) et implicite dans le modèle relationnel (rappel d'identificateur de n-uplet).

On ne pourra parler de redondance d'informations lorsque l'utilisateur manipulera plusieurs représentations d'un même objet mais de contenus différents. Cette situation est classique en C.A.O. de circuits où le même circuit aura une description logique, électrique, symbolique et un schéma des masques.

1.1.5.1.4 Intégrité des données

L'intégrité des données est une des fonctions non moins caractéristique des SGBD sans pour autant qu'elle soit prise en compte par la majorité des systèmes existants.

L'intégrité des données d'une base recouvre plusieurs notions:

- contrôle de concurrence: le contrôle d'accès aux données a pour but de coordonner les actions des usagers d'une base quant au partage de l'information et afin que celle-ci demeure cohérente.

- protection des données: c'est la possibilité de ne donner l'accès aux informations qu'à des personnes autorisées.

- sécurité de la base: elle consiste à maintenir l'existence et la cohérence de la base en cas d'incidents techniques de nature logicielle ou matérielle. Différents mécanismes de reprises ont été étudiés et sont actuellemnt utilisés pour revenir à un état cohérent de la base, plus ou moins éloigné.

- intégrité sémantique: on s'intéresse ici à la qualité du contenu de la base. Il faut être en mesure de garantir que les données stockées dans la base soient cohérentes entre elles <FERRAT 83>. L'aspect "intégrité sémantique" est systématiquement abordé dans la

phase de définition des spécifications d'un SGBD mais très rarement réalisé, tout au moins de façon satisfaisante, dans les différents prototypes et produits commercialisés. L'apport de la logique (PROLOG, LISP... ) sera un atout à ne pas négliger pour les SGBD du futur <TIGRE 15 84>.

Il est évident qu'à partir du moment où un système possède un degré de sophistication élevé, la mise en oeuvre de modules supplémentaires pénalisera, en terme de temps de traitement, la part de travail consacrée à l'exécution, proprement dite, des programmes d'application. Mais, au même titre que pour la non redondance des données, les systèmes futurs ne pourront plus se contenter d'offrir les fonctions classiques d'archivage et de recherches des données. Ils devront être en mesure de garantir l'intégrité des données. Un signe évident de cette exigence est l'apparition des systèmes déductifs et l'utilisation de plus en plus fréquente des langages de programmation logique <NICO>. Ces derniers offrent de très grandes possibilités et facilités quant à l'expression et la vérification des contraintes d'intégrité sémantique.

Dans un domaine d'application telle que la C.A.O. où l'on a recours, à l'issue de chaque étape du processus de conception, à des programmes de simulation et/ou de test, on sent très bien ce qu'un module garantissant l'intégrité sémantique des données peut apporter quant à l'allégement de la charge de travail de ces programmes de validation. Le contrôle de la cohérence de l'information ne doit pas être réalisé au niveau des programmes de CAO afin:

- de ne pas surcharger la tâche des concepteurs de ces programmes
- de s'assurer un meilleur respect des règles de cohérence.

Dans le futur, un utilisateur ne devra plus pouvoir introduire au sein du système, des données qui soient incohérentes par rapport au contexte. Les modèles de données classiques utilisés dans les différents langages de programmation permettent déjà de définir des contraintes locales à l'aide de types enrichis: liste de valeurs, intervalles bornés. La puissance de ces outils demeure malgré tout très restrictive. Le point crucial devient la spécification de ces contraintes d'intégrité. Elle doit comporter la spécification des objets et des opérations sur ces objets. Elle nécessite un outil puissant de spécification formelle offrant une validation qui garantirait que ces spécifications soient complètes et cohérentes entre elles. La programmation logique dont l'étude est effervescente pourrait être cet outil.

1.1.5.1.5 Structure hiérarchique des objets

Cette notion de structure hiérarchique fait référence implicitement à la notion de modèle. Bien que cette partie ne se veuille pas descriptive des différents modèles, nous allons être tenus de faire allusion dans les lignes qui vont suivre à leurs principes fondamentaux.

Par définition, le modèle hiérarchique a pour élément de base des enregistrements logiques qui sont reliés entre eux pour constituer une arborescence. Par conséquent, ce modèle est parfaitement adapté pour représenter une structure hiérarchique des éléments de la base. Les contraintes du modèle hiérarchique (la racine d'un arbre ne reçoit aucune branche, un noeud n'a qu'un père) entrainent implicitement une redondance de l'information que l'on cherche précisément à éviter.

Le modèle réseau fait appel à l'idée d'entités et de liens qui permettront, notamment, de représenter des structures hiérarchiques avec l'hypothèse qu'une même entité puisse

appartenir à plusieurs hiérarchies sans duplication de l'information.

Le modèle relationnel est quant à lui tout à fait mal adapté pour formaliser de telles structures. Leur représentation reste malgré tout possible en ayant recours à des identificateurs de tuples mais leur manipulation fait appel à de fréquentes opérations de jointure, coûteuses en entrées/sorties.

### 1.1.6 INSUFFISANCES DES SGBD EN C.A.O.

Si les SGBD peuvent paraître comme une première solution aux problèmes de gestion des données en conception assitée par ordinateur, ils présentent malgré tout de nombreuses insuffisances et ceci plus précisément au niveau de la représentation des données.

#### 1.1.6.1 INSUFFISANCE SEMANTIQUE

La sémantique contenue dans les LDD des SGBD actuels est encore trop restreinte pour modéliser des objets aussi complexes que des circuits ou même des pièces mécaniques. Pour représenter de telles entités, la solution passe par la réalisation d'un langage permettant à l'utilisateur de définir des types beaucoup plus riches que les types de base actuellement disponibles. La tendance actuelle s'oriente vers des modèles étroitement associés à des champs d'applications bien définis et qui posséderaient à priori tous les types nécessaires à la représentation des données intervenant dans des secteurs d'activités particuliers.

#### 1.1.6.2 INSUFFISANCE DES OUTILS DE DEFINITION DES CONTRAINTES D'INTEGRITE

Nous avons mis en évidence l'opportunité de pouvoir définir, au moment de la description de la structure de la base,

des contraintes d'intégrité sur les éléments du schéma. Les outils proposés sont encore trop restrictifs pour spécifier, dans le cas de la conception de circuits, des contraintes d'ordre technologiques. Maintenant il apparait comme de plus en plus évident qu'il faille faire appel à la programmation logique pour définir ces nouveaux types de contraintes. C'est ainsi que dans le cadre du projet TIGRE <TIGRE1 83> on propose une collaboration entre le langage LAMBDA et PROLOG <TIGRE15 84> afin de spécifier d'une part les contraintes associées à toutes données LAMBDA et d'autre part, les programmes qui permettront de vérifier que ces contraintes sont bien remplies par tout programme d'application.

### 1.1.6.3 INSUFFISANCE DES CONTROLES D'ACCES

Le contrôle d'accès à l'information assure la confidentialité et la conservation de l'intégrité des données. Au paragraphe 1.1.2 nous avons défini la particularité des transactions en C.A.O. et leurs conséquences au niveau des interblocages entre utilisateurs. Ainsi, le fait de verrouiller les éléments au niveau de la relation ou du fichier ne pourra être retenu dans un systeme de conception. En affinant la granularité de l'élément à verrouiller jusqu'au stade de l'instance de l'objet complexe, les autres occurences de cet objet demeureront accessibles et ceci aussi bien en lecture qu'en mise à jour. Cette proposition n'est pas dénuée d'intérêt dans le sens où le fait d'appartenir à la même classe ne laisse rien présager sur la nature des rapports entre éléments de ce même ensemble. Cette stratégie existe déjà dans certains SGBD.

### 1.1.6.4 INSUFFISANCE QUANT A LA GESTION DES VERSIONS D'UN OBJET

La complexité des domaines d'application de la C.A.O. est telle que les concepteurs sont tenus de segmenter leurs

tâches en de nombreuses étapes. Si bien qu'ils ne peuvent décider, à l'issue de chaque stade intermédiaire, que telle ou telle solution est meilleure qu'une autre. Nous sommes alors tenus de leur proposer une gestion de ces différentes versions. Au niveau des modèles jusqu'alors disponibles, rien n'est prévu explicitement pour gérer ces alternatives. Seules des solutions temporaires sont actuellement proposées.

A ces problèmes de modélisation viennent se greffer des contraintes de performances résultant de systèmes de plus en plus complexes et des bases associées qui voient leur taille croître sans cesse.

### 1.1.7 COUT DE CES NOUVEAUX OUTILS EXIGES PAR LES NOUVELLES APPLICATIONS

Chacun des différents points abordés et développés dans le § 1.1.4 résulte d'exigences mises en évidence par de nouveaux utilisateurs. Ces degrés de sophistication des SGBD introduisent des couches supplémentaires par rapport aux S.G.F. classiques.

Si l'on s'attache à développer un modèle avec plus de sémantique, on va au devant d'un outil de mise en oeuvre plus complexe. La gestion et la manipulation de ces outils supplémentaires dégradera de façon évidente les performances du système.

Cet aspect sera d'autant plus regrettable que cette même notion de performances fait partie intégrante des besoins de ces nouveaux utilisateurs. De plus, à cette lacune, se superpose un problème évident lié aux accès à une masse d'informations de plus en plus conséquente. Lorsque l'on sait que dans un système tel que SOCRATE/CLIO, 30% du temps de traitement des transactions est consacré aux opérations de lecture/écriture, on peut imaginer très facilement les

conséquences d'une telle situation dans l'hypothèse d'utilisation de ce SGBD pour un système graphique. Pour gérer les données graphiques, il faudra s'orienter vers d'autres méthodes d'accès.

En résumé, nous avons mis en évidence une dégradation des performances à deux niveaux:

a) dégradation du temps de traitement

b) dégradation des entrées/sorties

Le premier niveau pourra être abordé sous deux aspects:

- matériel

- logiciel

La solution la plus évidente demeure celle qui consisterait à développer des unités de traitement plus rapides. D'autre part, on peut s'attacher à développer des optimiseurs aussi bien au stade de la traduction des requêtes qu'à celui de la recherche d'information.

Le point (b) est totalement indépendant de (a) puisqu'il se trouve être lié à la nature des applications nouvelles traitées par l'informatique. Avec l'apparition des processeurs 32 bits, on se dirige vers une génération de calculateurs dédiés au traitement d'applications couteuses en temps de traitement pur et volumineuses au point de vue masse d'informations à gérer. C'est précisément ce second aspect qui tend à pénaliser les SGBD vis à vis des utilisateurs potentiels de ce type de produit et pour lesquels nous tenterons d'apporter une première forme de solution au cours du chapitre 2.

## 1.2 APPLICATIONS SON ET IMAGE

### 1.2.1 PRESENTATION

Les applications de traitement du son et de la parole ont trop de similitudes à la fois par la nature des informations manipulées et par les traitements réalisés pour faire l'objet de parties distinctes. Nous décrirons chacune d'elles, rapprocherons leurs besoins et définirons leurs positions vis à vis des solutions proposées par les systèmes de gestion de bases de données.

### 1.2.2 TRAITEMENT DE LA PAROLE

#### 1.2.2.1 DESCRIPTION

Le travail des phonéticiens consiste principalement à étudier l'élocution d'un corpus de mots prononcés par des locuteurs d'origines géographiques et sociales différentes. Par conséquent chaque prononciation de phrase sera segmentée en phonèmes vus comme éléments atomiques de l'application et repérés par des marqueurs de début et de fin de prononciation. A ce doublet de marqueurs sera associée la chaîne orthographique qui correspond à cette découpe du son prononcé. Il faudra être, par exemple, en mesure de distinguer deux élocutions, du même phonème, prononcées par une personne.

De là on peut mettre en évidence les différentes étapes de réalisation d'une base du son et de ses traitements :

- acquisition de l'information : digitalisation de la parole

- échantillonage et indexation des phonèmes

- traitement proprement dit : reconnaissance, synthèse,...

L'acquisition de l'information correspond à la digitalisation d'un signal analogique par échantillonage à une fréquence donnée, 20khz par exemple, c'est à dire 20000 points à la seconde. Chacun de ces points pourra être codé sur un mot de 16 bits. Pour notre exemple, une seconde de son sera stockée sur 40 Ko <BURRIA 82>.

Dans une seconde étape, cet enregistrement sera visualisé afin d'être échantilloné. A chacun de ces échantillons, nous associerons le nom du locuteur, la référence de la phrase à laquelle appartient ce phonème et enfin la coresponsdance orthographique du son traité. Enfin, l'utilisateur pourra, par exemple, demander au système de lui restituer la prononciation d'une phrase ou d'une portion de phrase prononcée par un locuteur à une date donnée. Ensuite, il devra être en mesure d'effectuer les différentes opérations liées à la reconnaissance de la parole.

### 1.2.2.2. LES BESOINS DU TRAITEMENT DE LA PAROLE EN MATIERE DE GESTION DES DONNEES

Jusqu'à présent, les phonéticiens et les acousticiens ont eu recours aux outils classiques de gestion de fichiers pour la manipulation de leurs données. Cette utilisation de l'informatique dans ce domaine étant récente, il comptait beaucoup plus à leurs yeux de voir aboutir la puissance de calcul offerte par cette nouvelle approche plutôt qu'une gestion sophistiquée des objets manipulés. Une fois que ces techniques de reconnaissance ont été suffisamment développées, de nouvelles exigences sont apparues avant d'aborder la phase d'implantation sur un site opérationnel.

Ces nouveaux besoins ne seront pas aussi exigeants que ceux relevés dans les applications de Conception Assistée par Ordinateur (cf § 1.1). On peut difficilement parler de confidentialité de l'information de même que l'intégrité des données n'est pas évidente dans la manipulation de phonèmes si l'on considère que les opérations effectuées comportent peu de mises à jour.

Ce type d'application fait appel à deux classes d'informations bien distinctes:

- les phonèmes prenant la forme de son digitalisé

- la description des phonèmes i.e. leur chaîne orthographique, le nom du locuteur, les pointeurs de début et de fin de prononciation...

Les phonèmes sont représentés par une chaîne de bits au sein de laquelle on ne peut déceler aucune structure et pour laquelle on ne connait pas à priori sa longueur. Par contre les rubriques de leurs descriptions ont été figées par l'utilisateur et pour chacune d'entre elles on connait son type et sa longueur. La structure de ces descriptions est plane et ne présente aucune complexité.

Ces deux classes d'éléments considérées indépendamment les unes des autres ne sont d'aucune signification. D'où la nécessité de pouvoir établir un lien, tout au moins unidirectionnel, entre ces données.

La diversité des informations manipulées par le traitement de la parole se réduit donc à des:

- éléments non structurés de longueur variable

- éléments structurés de longueur fixe associés aux précédents.

Les utilisateurs de systèmes de reconnaissance de la parole souhaitent avoir un langage de manipulation du son souple et interactif. Pour eux il est important d'obtenir facilement et rapidement la prononciation d'un phonème prononcé par un locuteur à un instant précis, d'où la nécessité d'un langage de requête puissant par la variété des opérations qu'il est en mesure de traiter et simple de mise en oeuvre afin d'en rendre son utilisation plus aisée.

Remarque : ce langage de manipulation de données sera utile au phonéticien dans un but d'interrogation mais non d'insertion. En effet, pour l'ajout d'informations dans le systeme, il faudra avoir accès aux primitives situées sous la couche du langage de requête tant la masse d'informations à insérer est volumineuse.

Nous avons avancé précédemment le chiffre de 40 Ko pour une seconde d'enregistrement. L'application de la reconnaissance de la parole dans une réalisation pratique nécessitera le stockage, la manipulation de plusieurs secondes, voire de minutes de son.

Le stockage de grosses masses d'informations intéresse la recherche informatique depuis de longues années et on peut commencer à voir paraître ses premiers résultats: le disque optique numérique. La principale différence avec le disque magnétique vient de la non-réinscription du support puisque le principe du stockage d'une donnée est liée selon la technique à la présence ou non d'une aspérité ou d'une cavité à la surface du support. Les caractéristiques de capacité sont de:

- 40000 pistes de 16 secteurs de 2 Ko soit 1 giga octets par face.<BD3>

On cherche maintenant à utiliser ces périphériques en batteries de disques optiques numériques comprenant jusqu'à 64 "Giga Disques", soit une capacité de 128 millards d'octets (juke box avec des disques double faces). Les temps d'accès seraient de l'ordre de 100 ms, c'est à dire du même ordre d'accès qu'un disque magnétique de bas de gamme mais malgré tout beaucoup plus lent que les disques d'utilisation courante.

Ainsi le problème de performances d'accès est lié à deux facteurs:

- périphériques plus lents

- masses d'informations échangées plus volumineuses.

## 1.2.3 ANALYSE D'IMAGES

### 1.2.3.1 DESCRIPTION

L'analyse d'images présente de nombreuses analogies avec le traitement de la parole. C'est pourquoi nous avons décidé d'évoquer sommairement ce type d'application dont la description est extraite du rapport <BD3>.

On peut mettre en évidence trois types d'application faisant appel à l'analyse d'images:

- les applications géographiques

- les applications médicales

- les systèmes graphiques.

Elles ont un domaine d'utilisation très vaste: cela va de l'étude de la pollution atmosphérique à celle de l'implan-

tation d'usines ou de routes dans une région donnée en passant par l'étude de l'état d'avancement des cultures.

Ces applications manipulent aussi bien des données d'ordre alphanumériques que des images. Celles-ci pourront être construites à partir d'objets élémentaires tels que le point, la ligne, la région. De là naissent des objets plus complexes.

Sur ces objets, élémentaires ou complexes, on effectuera un ensemble d'opérations dont nous allons fournir une liste non exhaustive:

- opérations ensemblistes

- opérations aggrégats (surface, longueur,...)

- zoom,...

- symétrie, rotation,...

Les opérations que l'on pourra effectuer sur ces objets géographiques sont dépendantes de l'implantation physique choisie:

- Représentation polygonale: une carte sera représentée par des régions, des lignes et des points, les régions par leur contour, les lignes par des segments de droite, les droites par les coordonnées de l'origine et de l'extrémité. Cette représentation ne sera pas aussi fine et aussi précise que la seconde. Les opérations topologiques, telles que l'inclusion, conviennent très bien à ce genre de représentation alors que les opérations logiques sont plus difficiles à effectuer.

- Représentation cellulaire: chaque image est découpée en cellules ou pixels, auxquels sont affectées des va-

leurs. Une carte est donc représentée par le contenu de ses régions et non plus par leur contour. Cette représentation est intéressante pour les applications qui s'attachent à l'étude des valeurs différentes d'un même attribut. Par contre cette solution sera très couteuse en place mémoire d'autant plus que l'on va vers des systèmes à très haute résolution (nombre très élevé de pixels par image). Cette fois-ci les opérations topologiques seront plus complexes que les opérations de nature logique.

### 1.2.3.2 OUTILS DE GESTION D'OBJETS GEOGRAPHIQUES

L'analyse d'image est suffisamment récente pour que ses adeptes aient pu profiter des outils les plus sophistiqués en matière de gestion de l'information. Nous allons mettre en évidence ce que doit ou devra satisfaire les systèmes de gestion d'images.

Par analogie avec le traitement de la parole, l'analyse d'image fait appel à deux classes d'informations. La première, et du même coup la plus évidente, est celle qui constitue l'essence des données traitées: c'est à dire l'image. Elle pourra être digitalisée ou encore stockée sous forme analogique. Elle ne révèle aucune structure. A l'issue de cette phase d'acquisition, elle pourra être analysée. Mais les résultats obtenus ne seront d'aucune signification si on ne peut les rattacher à un descriptif de l'image traitée. Alors intervient la seconde classe d'information, celle qui décrit l'image. Ce descriptif pourra être très varié selon la nature de l'application Il sera assimilé la fiche d'identité de la photo (date de la prise de vue, position géographique,...).

Ces deux classes d'informations seront en étroites associations dans la mesure où, isolées, elle n'ont aucune signification. Ainsi à partir de la description alphanumérique

d'une image, il faudra être en mesure de pouvoir retrouver sa forme numérique.

La spécificité et la complexité de la nature de l'analyse d'image rend nécessaire la définition d'un modèle de données géographiques propre alors que la simplicité de l'information manipulée dans le traitement de la parole n'impliquait en aucune mesure une telle exigence.

De la même manière que nous l'avions définie pour le traitement de la parole, le système retenu devra offrir à l'utilisateur un langage de manipulation des données souple, interactif et d'utilisation aisée. A l'aide de ce langage, il devra être en mesure de demander la visualisation de la partie du globe de latitude 50° N et de longitude 63° E photographiée le 15/02/84 et demander l'affichage de la température qui y régnait ce jour là. Les outils classiques associés aux systèmes de gestion de fichiers sont trop figés pour de telles requêtes.

En dehors de ces aspects de convivialité, il demeure un aspect déjà largement abordé pour l'application de C.A.O. et celle du traitement de la parole qui domine de très haut chacune des exigences mentionnées précédemment: la gestion d'une masse d'informations volumineuse. Cette gestion implique des périphériques capables de stocker des matrices de (3240 x 2430) octets par image, des méthodes d'accès sophistiquées et enfin, la mise en oeuvre de canaux d'échanges à haut débit. Ce volume d'information aura une incidence directe sur les coûts d'accès aux données puisqu'il entrainera une dispersion de l'information entre les différentes unités de stockage.

## 1.2.4 EN QUOI L'APPROCHE B.D. APPORTE-T-ELLE UNE SOLUTION PARTIELLE AUX UTILISATEURS DE TELLES APPLICATIONS ?

Nous allons tenter de définir en quoi l'approche base de données a attiré les phonéticiens et les analyseurs d'images pour la gestion de leurs données. Nous mettrons en évidence, également, les exigences qui n'ont été que partiellement satisfaites par cette solution.

Nous ne reviendrons pas sur les avantages, des systèmes de gestion de bases de données, déjà mentionnés pour l'application de C.A.O. (structure hiérarchique, centralisation de l'information,...).

### 1.2.4.1 LANGAGE DE DEFINITION DE DONNEES

Les complexités structurelles des données vocales et des données visuelles ne sont pas de même degré. Une façon simple et évidente de prendre conscience de ce phénomène est de comparer les travaux effectués dans ce domaine (la modélisation des données) pour ces deux types d'application. Lorsque l'on aborde une application de traitement de la parole, la simplicité des objets manipulés n'astreint pas l'acousticien à investir dans des outils sophistiqués pour représenter ces phonèmes. Leur structure est plane.

Par contre, les images géométriques sont construites à partir d'objets géographiques élémentaires qui pourront être le point, la ligne ou la région. A partir de ces objets élémentaires, on construit progressivement des objets plus complexes, par exemple une rue, un quartier, une ville <BD3>.

La sémantique des modèles de données des SGBD a été développée pour répondre aux besoins des applications classiques de gestion ne faisant intervenir que des données bâties sur des types de base et à structure simple. Dés que

l'on veut modéliser à l'aide de tels outils des objets complexes telles que les images géométriques, on atteint très vite leurs limites.

De là, commencent à paraître différentes approches d'élaborations de modèles dont l'essence même est bâtie autour de la notion d'entités et d'associations. Ensuite, chaque modèle traduira, selon ses propres concepts et son niveau de sémantique ces éléments de base <PALA 84> <VELE 84>.

De la même manière qu'en reconnaissance de la parole, l'analyse d'images manipule des objets qui sont la combinaison d'objets classiques (alphanumériques) et d'objets spécifiques (média digitalisé).

La prise en compte d'objets non structurés, d'associations entre éléments structurés et non structurés ne sont pas réalisées. Nous sommes alors tenus de développer des artifices pour passer outre ces limitations et on obtient en résultat un logiciel qui ne peut être assimilé à un système homogène de gestion de données.

Jusqu'à présent les types de bases étaient les entiers, réels, décimaux et les caractères. Mais on s'oriente de plus en plus vers des contextes où les objets élémentaires seront beaucoup plus complexes, ce seront les phonèmes ou même l'image. Il s'ensuit que nos modèles devront être enrichis non seulement au niveau des types de données mais aussi au point de vue des opérateurs de manipulation associés à ces nouveaux éléments de base.

### 1.2.4.2 LANGAGE DE MANIPULATION DE DONNEES

Le besoin d'un langage de manipulation des données souple et performant n'est pas le propre des applications du traitement de la parole ou des images. On veut pouvoir accéder, d'une façon qui soit de plus en plus voisine du langage

naturel, l'ensemble de l'information stockée et ceci quelque soit la complexité des requêtes. Ces langages seront principalement orientés utilisateurs finaux et devront être non procéduraux. Dans un proche avenir, il faudra dépasser le stade des langages relationnels en vue du développement d'outils de manipulation de type langage naturel.

On peut voir paraître certaines difficultés dans la mise au point de tels outils. On reconnait que l'on sera amené à manipuler des objets de complexité croissante et que de nouveaux types, plus riches, devront être définis. Il s'ensuit que de nouveaux opérateurs, associés à ces nouveaux types, devront être définis. Pourra-t-on alors conserver cet aspect de convivialité avec ces nouveaux langages ? Pour les concepteurs d'outils dédiés à ces nouvelles applications, il serait bon de tenter une nouvelle approche dans la spécification du niveau de complexité des LMD. Ces concepteurs ont un rayon d'activités bien défini au sein du domaine sur lequel ils travaillent. Par conséquent, il serait préférable de pouvoir leur offrir un langage sophistiqué bien que complexe plutôt qu'un outil simple mais couteux en mise en oeuvre. D'autant plus que les occasions d'engorgement seront suffisamment nombreuses par ailleurs.

### 1.2.4.3 LES SGBD REPARTIS

Dans le cas de SGBDR (R:réparti), l'information est dispersée sur plusieurs sites et on cherche précisément à rendre ce fait totalement transparent à l'utilisateur. On veut lui donner l'impression d'opérer sur un site unique avec un langage unique. Derrière cette façade, le SGBDR met en oeuvre tout un ensemble de mécanismes complexes de recherche de l'information et d'interprétation des requêtes de l'utilisateur.

Il faut savoir que cette répartition de l'information ne va pas sans poser des problèmes de performance quant à la lo-

calisation et à l'accès de l'information. Certains systèmes tels que R* d'IBM offrent des outils pour recalculer une meilleure localisation de l'information en fonction de la situation géographique des sites qui l'accèdent le plus fréquemment. L'utilisateur pourra demander à avoir sur son site des copies de relations afin de réduire les coûts inhérents à cette dispersion.

De telles architectures rentrent parfaitement dans la lignée des besoins de distribution de l'information telle qu'on l'a spécifiée dans les exigences des accousticiens et des analyseurs d'images. Par contre elles nécessitent une certaine homogénéité au niveau des outils disponibles sur chacun des sites. On ne pourra réaliser directement une telle répartition avec des calculateurs supportant différents modèles de données <ADI 78>. Les interfaces à mettre en oeuvre seront multiples et très complexes. Il en résulte une tendance très marquée en faveur de la notion de serveur de données qui va dans le sens d'un partage de l'information mais cette fois-ci sur un site privilégié et dédié à la manipulation de la base de données générale. Les avantages d'une telle configuration seront repris au cours du chapître 2.

## 1.3. SYNTHESE DES NOUVELLES CARACTERISTIQUES DES SGBD DU FUTUR

A travers l'analyse de ces trois nouveaux domaines d'utilisation de l'informatique, nous allons pouvoir dresser une liste, non exhaustive, des nouvelles spécifications qui devront être prises en compte par les futurs systèmes de gestion de bases de données afin que ceux-ci demeurent le véritable outil de gestion de l'information. Ceux-ci devront pouvoir:

- décrire et manipuler des objets structurellement complexes ainsi que des éléments non structurés et de longueur variable

- établir des associations entre éléments structurés et non structurés

- spécifier et vérifier des contraintes d'intégrité

- offrir une nouvelle gestion des concurrences pour permettre la mise en oeuvre de transactions de longue durée

- autoriser la modification du schéma de la base à tout instant

- gérer différentes versions d'un même objet

- offrir de nouvelles interfaces de haut niveau pour des non informaticiens: langage naturel,...

- offrir une puissance de traitement en rapport avec la complexité des applications.

Nombre de ces exigences seront résolues par la spécification de nouveaux modèles de données intégrant les notions de types abstraits, l'utilisation de la programmation logique ou la mise en oeuvre de nouveaux concepts de base (gestion des concurrences d'accès,...).

Jusqu'à présent l'intérêt porté sur l'aspect performance des SGBD a été très modeste. On ne cherchait à optimiser que des parties bien précises des différents systèmes telles que la réduction du volume des résultats intermédiaires des transactions dans un système de bases de données réparties <CAL *78> ou encore la projection des espaces virtuels sur les espaces réels pour les systèmes à mémoire virtuel-

le. Ces besoins ne résultaient pas de la spécificité des applications prises en considération par les SGBD mais davantage d'études ponctuelles réalisées au niveau de différents modules du système. En fait, on n'était pas confronté à des situations d'engorgement mettant en cause l'utilisation des SGBD.

La nécessité actuelle de développer des systèmes performants pour des nouvelles applications résulte :

- de la mise en oeuvre de types plus riches et d'opérateurs associés complexes

- de traitements très couteux en CPU

- d'une masse d'information de plus en plus volumineuse et à accéder rapidement en raison d'exigences des utilisateurs ou de la spécificité de l'application

- d'une tendance très marquée visant à rendre le modèle relationnel comme modèle de référence pour les SGBD du futur.

Une première forme de solution passe par une remise en cause plus profonde des systèmes classiques, tout au moins de leur architecture. Et c'est précisément sur cet aspect que nous allons porter notre dévolu au cours des chapîtres suivants en commençant par présenter une première forme de solution, les machines bases de données, qui sera testée par la suite.

# CHAPITRE 2

# EVOLUTION DES ARCHITECTURES DES SGBD

# VERS LES MACHINES BASES DE DONNEES

## 2.1 PREAMBULE

Nous avons pu nous rendre compte à la lecture du CHAPITRE 1 que pour répondre aux exigences des nouvelles applications, les SGBD ont à manipuler des bases de taille sans cesse croissante et structurellement beaucoup plus complexes que les bases de gestion. Il en résulte indéniablement une dégradation des performances. Si sur le plan stockage cela ne pose en général plus de problèmes, il en n'est pas de même pour les temps d'accès.

Il semblerait que les limites de l'optimisation du logiciel aient été atteintes et qu'un gros travail reste à faire au niveau de l'architecture des futurs systèmes de gestion de bases de données. Alors apparurent les premieres propositions, intégrant matériel et logiciel, avec les Machines Bases de Données (ou MBD) et les processeurs dorsaux.

## 2.2 POURQUOI DES MACHINES BASES DE DONNEES ?

### 2.2.1 MOTIVATIONS

Avant de définir les concepts matériels contenus dans les MBD, il serait bon de reprendre plus en détail les motivations qui ont conduit au développement de tels outils.

Le besoin majeur qui a été repris dans la plupart des cahiers des charges des concepteurs de MBD est d'alléger l'unité centrale. Cet objectif pourra être atteint en déportant les fonctions de base du SGBD vers les modules qui composeront la MBD. Ces opérations seront distribuées sur le site de stockage des données où l'on pourra améliorer sensiblement leurs performances. Une telle approche a de multiples conséquences :

- CPU plus disponible: il est bien connu que sur une machine conventionnelle le SGBD passe le plus gros de son temps à interpréter les appels au système, à exécuter un grand nombre de modules pour traiter ces appels et à déterminer la localisation des données, à transférer de gros volumes d'information de la mémoire secondaire vers la mémoire principale

- Mémoire centrale libérée et allégement des canaux d'E/S: si par exemple,on distribue les opérations de recherche de tuples d'une relation, par exemple, sur le site de stockage, on n'aura plus à transférer toutes les pages de la relation à traiter en mémoire centrale du site hôte. On libérera ainsi de la place mémoire en vue d'autres applications et on allégera du même coup le taux d'occupation des canaux d'entrées/sorties. Ceux-ci ne serviront plus qu'au transfert des données satisfaisant les critères de sélection contenus dans les requêtes adressées à la MBD sur le même canal.

- Temps de réponses améliorés: à première vue, on pourrait croire qu'en exportant certaines fonctions de l'U.C. d'un gros calculateur vers celle d'un plus petit on augmenterait le temps de calcul (en comparant la puissance de traitement relative de chacun d'entre eux). Il faut garder à l'esprit que la MBD pourra être construite autour de processeurs spécialisés pour le traitement de fonctions spécifiques (processeurs de jointure,...). Par contre la limitation du nombre d'échanges au niveau des canaux améliorera les temps de réponse d'autant plus que l'on va vers des BD de taille gigantesque où le nombre d'enregistrements à tester pour chaque requête est de plus en plus élevé. Sans rentrer dans les spécifications techniques de certaines MBD, on peut avancer le côté attirant du parallélisme inter et intra-requête. Une façon de le réaliser serait de multiplier le nombre de processeurs traitant une requête sur de petites unités de stockage. Avec cette configuration, le parallélisme croît avec le nombre de processeurs alloués à chaque requête d'où une réduction notoire du temps de traitement.

On vient d'introduire la notion de processeurs spécialisés. Les SGBD sont composés de modules réalisant des fonctions particulières. Tous ces modules son implantés sur le même matériel et on peut déjà sentir une inadéquation de celui-ci vis à vis de ces fonctions. C'est pourquoi il y a beaucoup à gagner en tentant non seulement de distribuer ces fonctions mais aussi en les transférant sur des processeurs spécialisés qui auront été développés pour traiter explicitement ces tâches. On peut déjà sentir que durant les années à venir, les SGBD évolueront vers des sous-systèmes modulaires plutôt que vers des systèmes intégrés comme les SGBD actuels.

A l'issue de ces observations nous pouvons donner une première définition des MBD: une MBD est composée de modules matériels et logiciels réalisant la totalité des fonctions du SGBD.

Le principe général des MBD se retrouve dans la notion de "frontal". Un calculateur "frontal" est utilisé comme interface entre un calculateur hôte et un réseau d'ordinateurs ou de terminaux. La MBD servira d'interface entre le calculateur hôte et la BD.

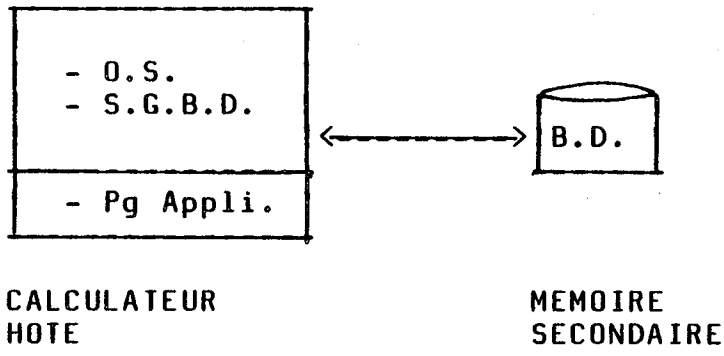



Figure 2.1 - Système conventionnel

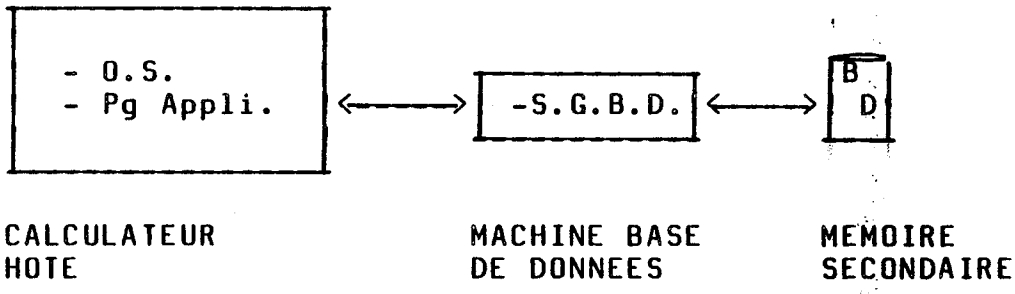



Figure 2.2 - Système comportant une MBD

Si l'on considère la configuration de la figure 2.2, il faudra interfacer le calculateur hôte avec la MBD. Cette interface sera chargée de récupérer les requêtes des programmes d'application et de les transmettre au dorsal. En

retour elle acceptera les réponses du dorsal et les distribuera aux programmes d'application.

## 2.2.2 NOUVEAUX AVANTAGES DE CETTE APPROCHE

Lorsque l'on travaille sur une architecture répartie, nous pouvons accéder à plusieurs bases situées sur chacun des calculateurs. Avec l'approche conventionnelle, nous sommes confrontés à deux types de problème:

- verrouillage adéquat des bases entre les calculateurs
- mise au point d'interfaces sur chacun des calculateurs dans le cas de non correspondance des modèles.

Avec l'approche MBD, le fait que la gestion des données se fasse au niveau du site de stockage solutionne le premier point et simplifie le second dans le sens où chaque calculateur se préoccupera uniquement de son interface avec le dorsal. Ainsi on aura à notre disposition l'opportunité d'assurer localement la compatibilité des modèles de données: compatibilité logique et physique (figure 2.3).

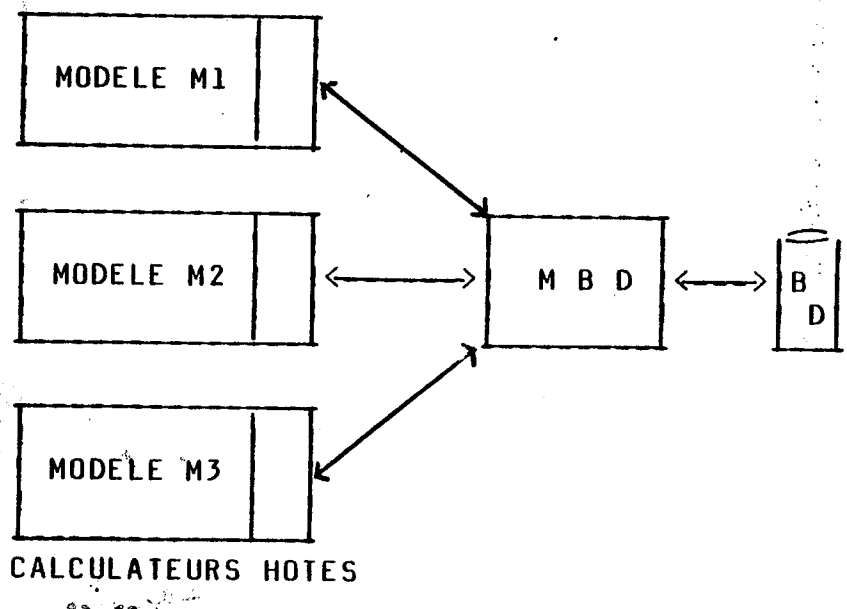



Figure 2.3 - Partage de la base

Supposons qu'une MBD ait été développé pour une machine hôte spécifique et que l'on désire l'utiliser depuis un nouveau site hôte. Il ne reste plus qu'à écrire une interface sur la nouvelle machine afin d'atteindre la base gérée par la même MBD.

Certains auteurs commencent à introduire une nuance fondamentale entre machine base de données et processeur dorsal. En effet ils définissent une MBD comme un calculateur sur lequel a été entièrement déporté le SGBD (cf définition précédente). Avec cette définition nous retrouvons tous les avantages mentionnés plus haut, notamment ceux faisant intervenir un ensemble de machines hôtes accédant à une base unique gérée par la MBD. Par contre, vouloir faire effectuer le plus d'opérations à une seule machine, ne fait que recréer des situations d'engorgement que l'on veut précisément éviter. D'un autre côté, ils définissent le processeur dorsal comme une unité sur laquelle ont été déportées uniquement certaines fonctions de base du SGBD et non le SGBD dans son intégralité. Ainsi les accélerateurs disques pourront être assimilés à des processeurs dorsaux <COPER2>. Ils assistent le calculateur hôte pour l'exécution des opérations d'accès à la base.

Du même coup cette seconde approche perd de nombreux avantages au niveau de la BD partagée mais par contre peut conduire à une plus grande spécialisation du matériel vis à vis des tâches qu'il aura à remplir.

Ainsi en résumé, une MBD implique un processeur dorsal mais l'inverse est faux. Le dorsal est utilisé avec une machine hôte qui recevra et compilera les requêtes alors que la MBD pourra être connectée à un hôte (figure 2.4) ou au contraire être utilisée de façon autonome (figure 2.5).

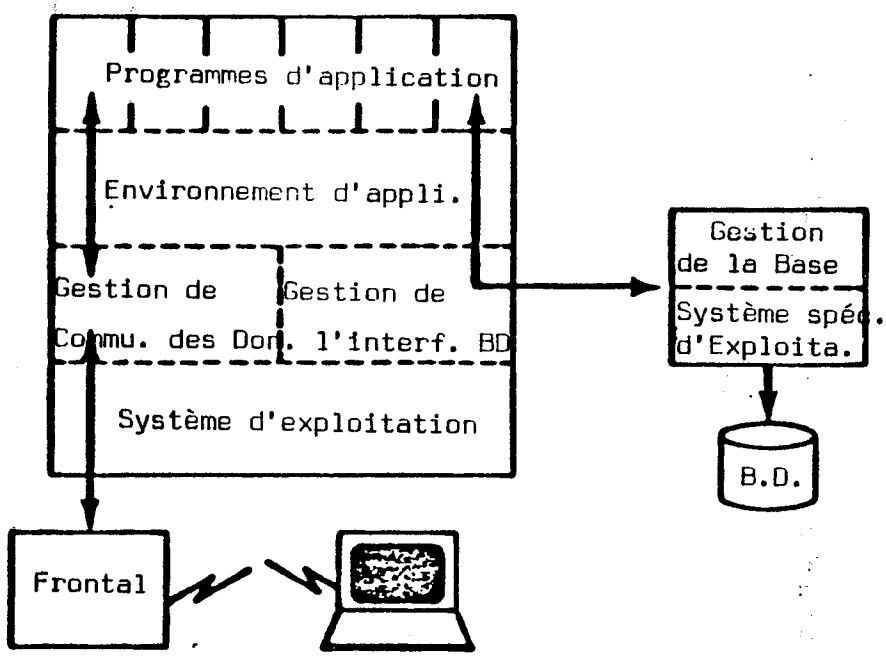



Figure 2.4 - MBD et calculateur hôte

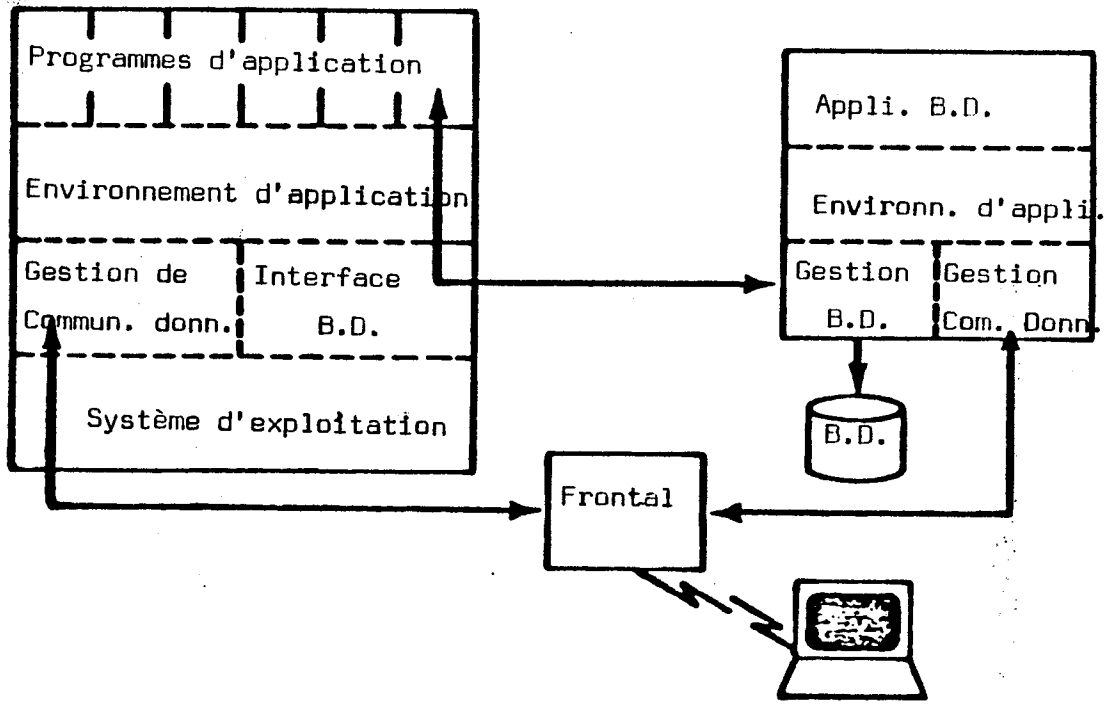



Figure 2.5 - MBD spécifique

Le degré d'indépendance vis à vis de la machine hôte est fortement lié à la solution retenue: dorsal ou MBD. Dans le cas d'utilisation de MBD, il faudra prendre en considération la transformation des requêtes, issues de la machine hôte, en un format compréhensible par la MBD.

## 2.3 OUTILS DISPONIBLES POUR LA REALISATION DES MBD

Le très grand pouvoir d'intégration de la logique sur les circuits (VLSI) ont permis une meilleure distribution de cette logique sur les lieux de mémorisation. Certains auteurs avaient avancer l'idée d'introduire la logique et les données sur un "chip". Mais cette solution parut trop onéreuse pour de gros volumes de données. Par contre l'idée d'une distribution de micro-processeurs ne serait pas à négliger. Pour le moment, l'association de la logique au mécanisme de lecture-écriture du disque est certainement la solution la plus couramment avancée.

D'autre part il a été reconnu que les SGBD se sont sentis très vite limités par le temps d'accès disque d'où toutes les recherches sur les mémoires caches et les accélérateurs disque. Ces nouvelles techniques permettent d'accroitre sensiblement le nombre de postes de travail et d'ouvrir une ère nouvelle aux applications manipulant de gros volumes d'informations (CAO, DAO, animation d'images...).

La réalisation des MBD ne repose pas uniquement sur des concepts matériels. L'état de l'art dans ce domaine fait état d'une attirance certaine pour l'adressage par contenu. D'où la nécessité de développer une représentation des données adéquate. Les performances de l'application dépendront notamment du choix de la représentation physique des données.

Dans le cas d'un système relationnel, on aura à choisir la façon avec laquelle seront repérées les relations et leurs attributs. Ainsi les valeurs des attributs pourront être à des places fixes ou au contraire, précédées du label de l'attribut. De plus il faudra savoir où et comment identifier les tuples qui auront satisfait les critères de sélection. Les réponses à ces questions seront le fruit de stra-

tégies que le concepteur voudra prvilégier par rapport à d'autres.

Il peut être intéressant de mentionner l'effort consenti par quelques constructeurs japonais pour intégrer le SGBD dans le système d'exploitation de la machine.

Après avoir défini les besoins qui ont motivé l'élaboration des MBD et les techniques nouvelles qui leur donneront un second souffle, il serait bon de lister les trois grandes classes de MBD existant ou en cours d'élaboration et de spécifier leur mode de fonctionnement.

## 2.4 LES DIFFERENTES APPROCHES EN MATIERE DE MBD

Préambule: La structure de cette partie a été fortement inspirée de l'article "Machines Bases de Données: une introduction" de François BANCILHON, Septembre 1982 <BANCILHON>.

La classification qui va suivre correspond aux grandes lignes de recherche en matière de Machines Bases de Données:

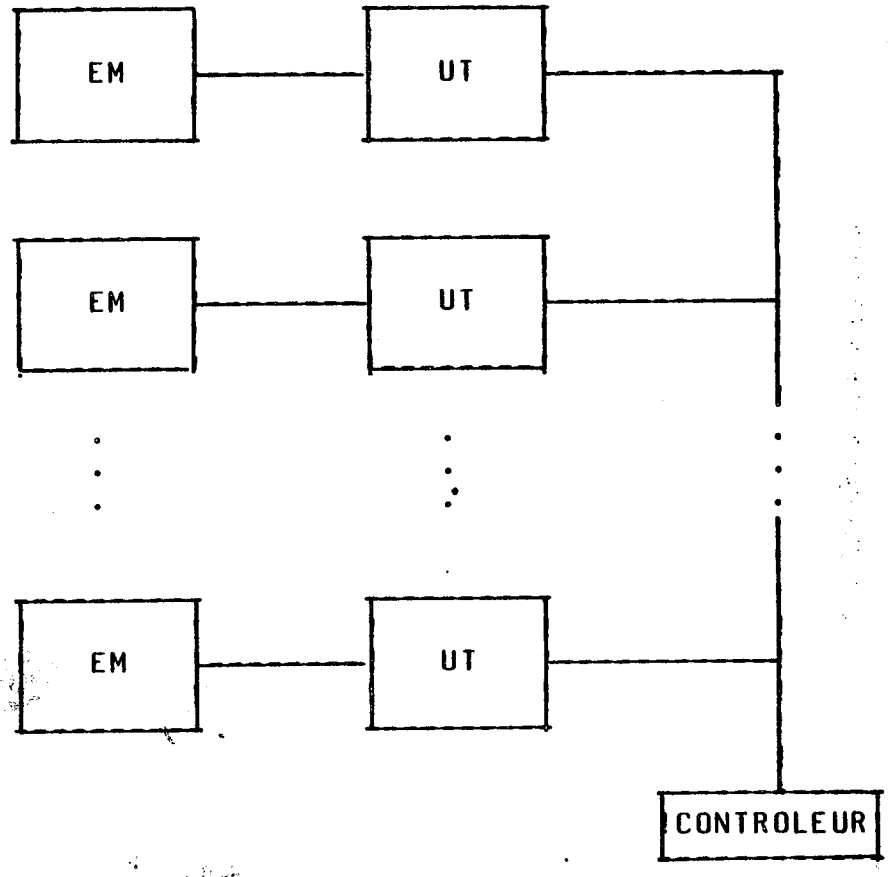
- les processeurs associatifs
- les mémoires associatives
- les machines MIMD

## 2.4.1 MACHINES A PROCESSEURS ASSOCIATIFS (OU CALCULATEURS ASSOCIATIFS)

### 2.4.1.1 PRESENTATION

Il existe depuis longtemps des calculateurs associatifs pour le traitement d'applications spécifiques (traitement du signal, de l'image). Les plus connus d'entre eux étant STARAN et PROPAL 1 et 2. Ainsi est née l'idée d'utiliser la puissance de ces machines afin de traiter des problèmes de type Bases de Données.

Définition: "Un calculateur associatif est formé d'éléments de mémoires (ou EM) auxquels sont associés des unités de traitement (ou UT) relativement simples."



Cette structure est de type SIMD ("Single Instruction Multiple Data"), c'est à dire qu'à un instant donné toutes les UT traitent la même instruction sur leur EM respectifs. Le nombre de couples (EM UT) détermine la puissance de la machine et son degré de parallélisme.

Illustration: prenons le cas du modèle relationnel et de l'opérateur de sélection d'un n-uplet d'une relation. On chargera les n-uplets dans les EM et chaque UT effectuera l'opération de sélection. Ainsi la vitesse d'exécution est multipliée par le nombre de couples (EM UT).

Cette structure SIMD nécessitera que tous les n-uplets soient formattés de la même façon d'où l'obligation de représenter tout en format fixe. D'autre part, toute la base ne pourra être transférées dans les EM. Ces derniers devront être alimentés depuis une mémoire de masse à un débit impressionant.

Il semblerait que ce soit pour cette raison que les recherches en matière de processeurs associatifs aient été abandonnées à ce jour. Mais malgré tout l'idée de base des EM et des UT a été reprise dans certaines réalisations et sera très certainement à prendre en compte dans les futures réalisations.

Exemples de Processeurs Associatifs: PROPAL 2, STARAN.

### 2.4.1.2 UN PROCESSEUR PARALLELE ASSOCIATIF: PROPAL 2

La nature des traitements envisagés pour PROPAL 2 nécessite une très grande puissance due à un nombre élevé d'informations à prendre en compte et à traiter dans un laps de temps extrememement court. Une architecture SIMD permettant de gérer un très grand nombre de processeurs a donc été retenue pour PROPAL 2. Il comporte deux éléments principaux (cf figure 2.6)

- un processeur parallèle

- une unité de gestion micro-programmée.

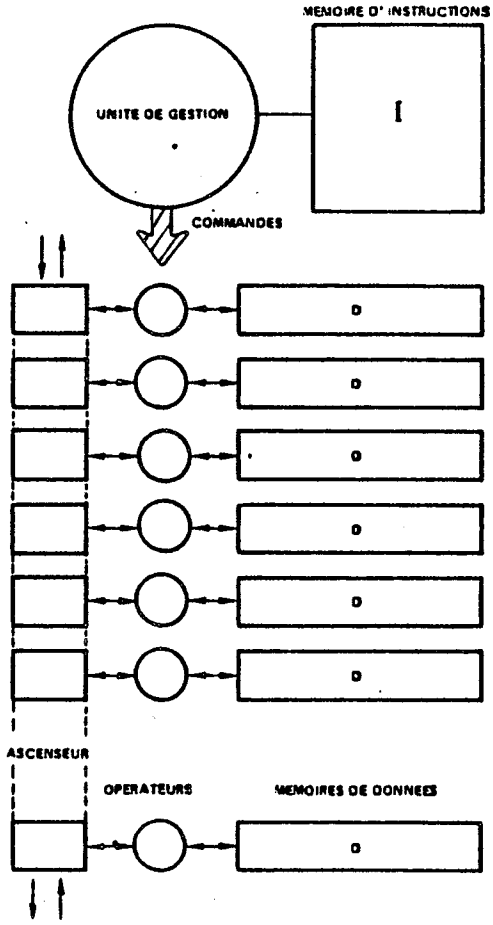



Figure 2.6 - Architecture de PROPAL 2

Le processeur parallèle de PROPAL 2 est constitué de processeurs élémentaires (PE) exécutant simultanément les mêmes opérations sous le contrôle de l'unité de gestion. Leur nombre P est compris entre 8 et 2048. Chaque processeur élémentaire comporte :

- une mémoire de données M de N bits

- une mémoire de travail A, dite ascenseur, de 16 bits utilisée principalement comme élément de communication avec les autres PE et avec l'extérieur

- des opérateurs booléens et arithmétiques OP capables de réaliser des opérations bit à bit sur des tranches de mémoire et d'ascenseur.

L'unité de gestion de PROPAL 2 a pour principale fonction le contrôle du processeur parallèle et aura également des échanges avec l'extérieur. Elle comporte deux principaux éléments:

- un micro-ordinateur et sa mémoire de commande dans laquelle sont rangés les micros-programmes permettant le séquencement et la commande du processeur parallèle

- une interface avec le processeur parallèle au niveau de laquelle sont générés tous les signaux nécessaires au fonctionnement de ce dernier.

PROPAL 2 peut être utilisé selon trois principaux modes d'exploitation:

- processeur "autonome", lié à des appareils non informatiques et constituant avec eux un équipement. Exemple: acquisition rapide de mesures à l'aide des E/S parallèles et stockage sur bandes magnétiques

- processeur connecté à un ordinateur par des liaisons "faibles" pouvant être utilisé au point de vue fonctionnel comme un périphérique intelligent

- processeur connecté à un ordinateur par des liaisons "fortes" pouvant être utilisé comme opérateur spécialisé. Exemple: calcul matriciel,...

Les domaines d'application de PROPAL 2 résultent directement des possibilités offertes par son architecture particulièrement bien adaptée au traitement des tableaux.

Ses applications peuvent se classées en deux grandes catégories:

- les applications numériques caractérisées par un nombre élevé d'informations, généralement acquises en temps réel et donnant lieu à des traitements mathématiques complexes bien adaptés au parallélisme :

  - traitement du signal acoustique

  - traitement des images.

- les applications non numériques caractérisées par l'utilisation de fichiers volumineux dont l'adressage par le contenu permet de faire des recherches rapides. C'est le cas des applications du type :

  - interrogation de B.D. Relationnelles

  - traitement de textes.

## 2.4.2 MEMOIRES ASSOCIATIVES

### 2.4.2.1 DESCRIPTION

La gestion des supports physiques d'information et les moyens d'accès sont actuellement, étant donné leur impact sur les problèmes de performances et de fiabilité, deux points fondamentaux de l'activité de recherche dans le domaine des MBD.

L'état de l'art est directement lié aux possibilités technologiques. Une des approches est la mémoire associative. Elle consiste à mettre en oeuvre des disques intelligents qui traitent les informations à la volée et permettent la sélection par leur contenu. Cette approche ne diminue pas

le temps d'accès aux informations mais réduit globalement le temps de réponse à une requête et la quantité d'informations véhiculées par le canal mémoire.

Ceci peut être réalisé par un filtrage séquentiel des données. SLOTNICK (1970) semble avoir été le premier à utiliser la rotation du disque pour réaliser de telles mémoires <SLO 70>.

En connectant un ou plusieurs filtres à une mémoire de masse (qui sera le disque) on obtient divers types de mémoires associatives.

Avec cette appproche on a un premier aperçu de la façon dont sera réalisée le rapprochement de la logique sur le site de stockage.

2.4.2.2 CLASSIFICATION

Notre classification des machines à mémoires associatives se fera autour de trois critères:

- nombre de filtres
- mode de connexion des filtres
- type de parallélisme

2.4.2.2.1 Nombre de filtres

On distingue trois grandes catégories:

PROCESSEUR PAR PISTE (PPP)

Chaque piste est munie d'un processeur de filtrage, ce qui nécessite à priori une mémoire de type disque à tête

fixe (en voie de disparition) ou des mémoires à bulles (performances meilleures qu'avec un disque à tête fixe).

Les tuples résultats sont stockés dans des buffers (au niveau de chaque processeur) et dés que l'un d'entre eux est plein, il est envoyé à l'hôte. Mais si le bus est occupé, le traitement des pistes est stopé jusqu'à ce que tous les buffers pleins aient pu être émis.

L'aspect positif de cette technique est qu'en une révolution on parcourt la base mais à quel prix:

- quantité de logique énorme pour une base de taille normale (logique déportée sur chaque piste, IBM 3330: 404 cylindres, 19 pistes/cylindre ----> 7676 processeurs)

- mécanisme puissant de gestion du flux des résultats. Exemple: taille de la base : 100 millions d'octets
  taille d'une piste : 10 Ko, notre requête sélectionnera 5% de la base soit 5 millions d'octets
  temps de rotation : 20 ns

  ---> débit d'arrivée des résultats: 250 Mo/s.

Il faudrait avoir recours à des tampons ou d'autres techniques.

Dans le cas des systèmes relationnels, les PPP obtiennent, de manière théorique, de meilleurs résultats que les architectures SIMD pour le traitement de la restriction car la seconde solution nécessite un transfert des données de la mémoire de masse vers une mémoire cache.

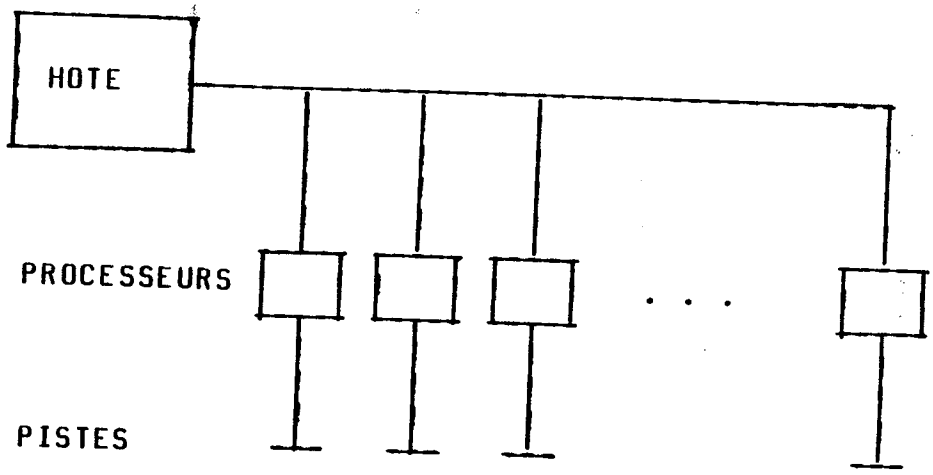



Figure 2.7 - Processeurs par piste

Exemples de machines PPP: RAP 1 et 2, CASSM.

## PROCESSEURS PAR TETE (PPT)

Cette technologie s'associe aux mémoires telles que les disques à tête mobile. Chaque tête de lecture est équipée d'un processeur capable de filtrer un cylindre en une révolution. La synchronisation entre l'ensemble des têtes n'est pas toujours triviale. On aura besoin sur certaines machines d'effectuer plusieurs rotations pour obtenir les données satisfaisant à un critère de sélection: une rotation pour les repérer, une ou plusieurs rotations pour les récupérer physiquement.

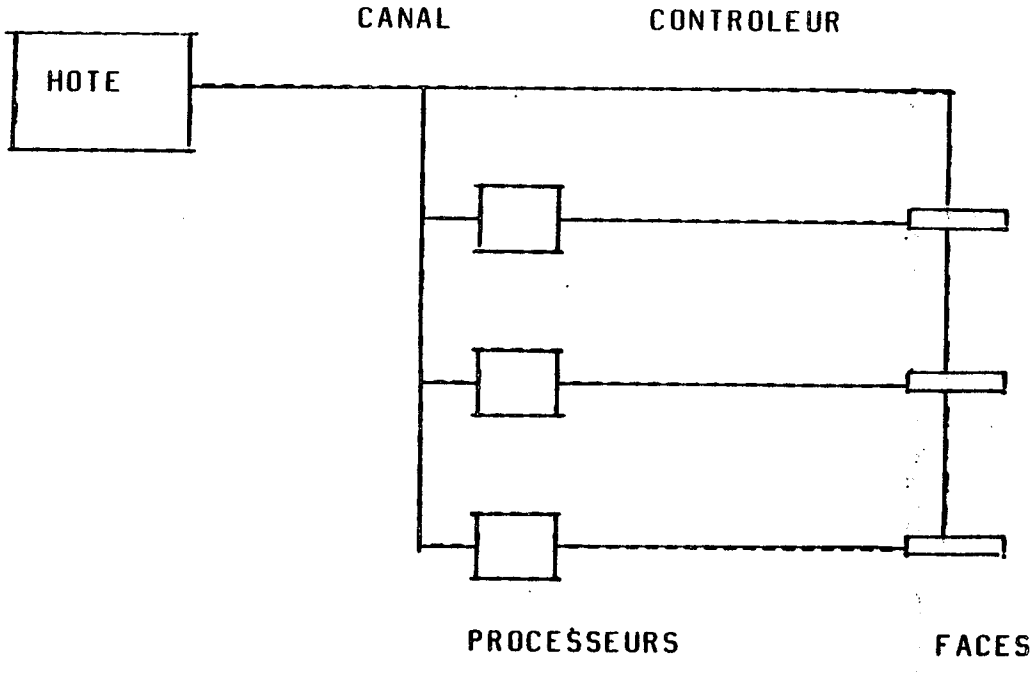



Figure 2.8 - Processeurs par tête

Exemples de machines PPT: DBC, SURE, SARI.

## PROCESSEUR PAR DISQUE (PPD)

On arrive ici à la réalisation la moins couteuse puisque l'on associe un (ou plusieurs) processeur à une unité de disque (éventuellement un filtre pour plusieurs disques). Ce genre d'approche est relativement intéressante lorsque l'on sait que pour réaliser un processeur de filtrage il faut entre deux à quatre cartes. Très vite on peut se rendre compte du coût inhérent à une structure de type PPT ou PPP.

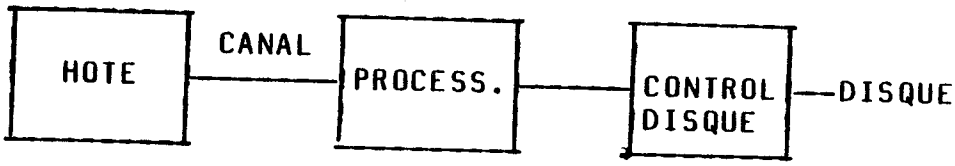
Des études d'évaluation ont montré qu'avec une utilisation judicieuse des index, les systèmes PPD peuvent avoir des performances proches de celles des systèmes PPT, ils sont en outre plus simples d'emploi (en effet pour tirer avantage d'une structure PPT, il faut une gestion de

l'espace disque allouant chaque cylindre à une relation).

Remarque: la recherche par contenu effectuée directement sur les têtes de lecture présente quelques inconvénients:

- l'étude doit être refaite en grande partie si le modèle de disque change. Les évolutions technologiques sont trop rapides pour que les constructeurs fassent de tels investissements.

- il doit y avoir adaptation permanente des performances entre le débit du disque et la capacité de traitement du dispositif associatif et ces deux technologies évoluent de manière indépendante.

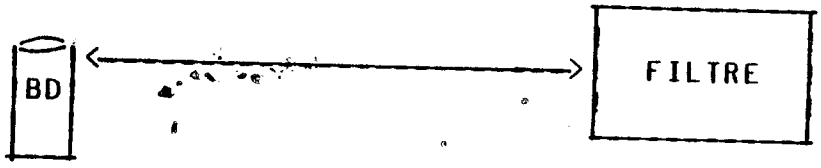



Exemples de machines PPD: IDM 500, CAFS, iDBP, VERSO, DORSAL 32.

2.4.2.2.2 Mode de connexion: processeurs / mémoire secondaire

Filtrage au vol pur

Les données sont directement traitées au vol, i.e. à la sortie du controleur disque.



Le résultat du filtrage est disponible en mémoire à la fin du transfert disque et ainsi on n'a pas besoin de tampon. Le filtre devra traiter les données à leur vitesse d'arrivée du disque et ceci quelque soit la complexité de la requête.

Par contre nous nous trouvons face à un problème de synchronisation entre deux processeurs à haut débit. Si celle-ci ne peut être réalisée, cela suppose l'existence d'une FIFO entre le controleur disque et le filtre.

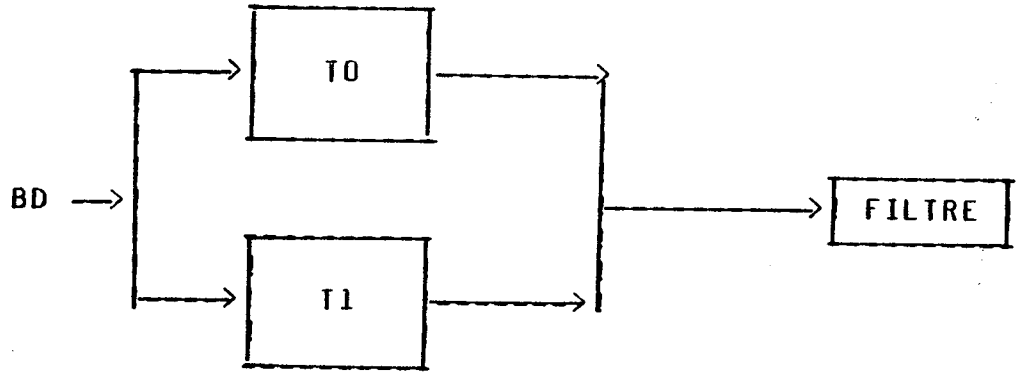
De plus la technologie du filtre est fortement liée à celle du disque puisqu'un changement de débit de ce derier nécessite une refonte du filtre.

Finalement il semblerait que les expériences de filtrage au vol pur aient été abandonées.

Filtrage avec tampon

Pour palier aux inconvénients du filtrage au vol pur, on introduit une zone tampon entre le disque et le filtre.

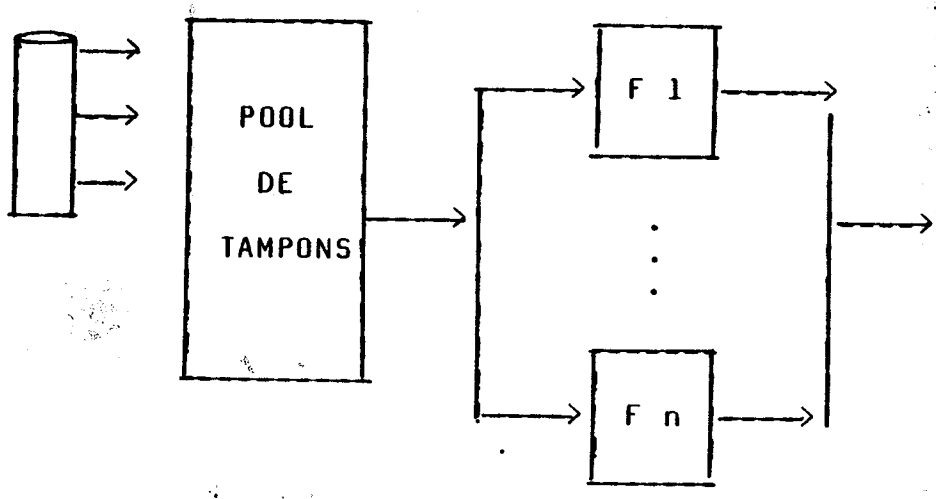

Le filtre devra attendre que le tampon soit plein (taille d'une piste ou d'un secteur) pour le vider et pendant ce temps le disque ne peut plus transmettre. Il en résulte un débit divisé par deux d'où la solution suivante:




TO est rempli par le disque pendant que T1 est vidé par le filtre et vice versa. Si les débits du disque et du filtre sont synchronisés, on aboutit à un transfert continu après un retard global égal au temps de lecture d'un tour de piste (taille du tampon).

Une autre méthode consisterait à avoir un tampon à double accès où le filtre pourrait lire pendant que le disque écrirait. Un mécanisme de verrouillage vérifierait alors que le filtre n'aille pas plus vite que le disque.

Des méthodes de gestion de "pool" de tampons ont été suggérées dans le cas multiprocesseurs.



Un ensemble de filtres se sert dans un "pool" de tampons rempli par le disque. Ce système offre la souplesse et la complexité de toute gestion dynamique de tampons.

2.4.2.2.3 Type et nature du parallélisme

Intra ou Inter requête

Seules les machines multi filtres sont capables de faire du traitemen parallèle:

- à l'intérieur d'une même requête, cas des PPP où tous les filtres traitent la même requête en parallèle sur un ensemble de pistes

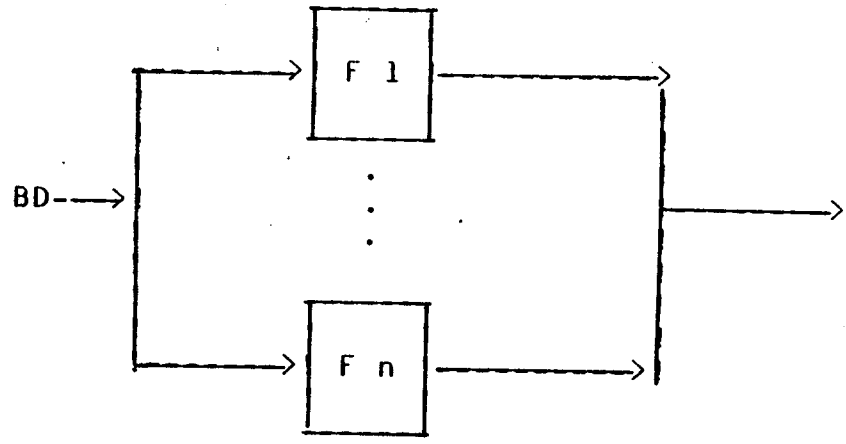
- entre deux requêtes, deux filtres exécuteront des traitements différents sur les mêmes données (MISD) ou sur des données différentes (MIMD).

(S ou M) I (S ou M) D

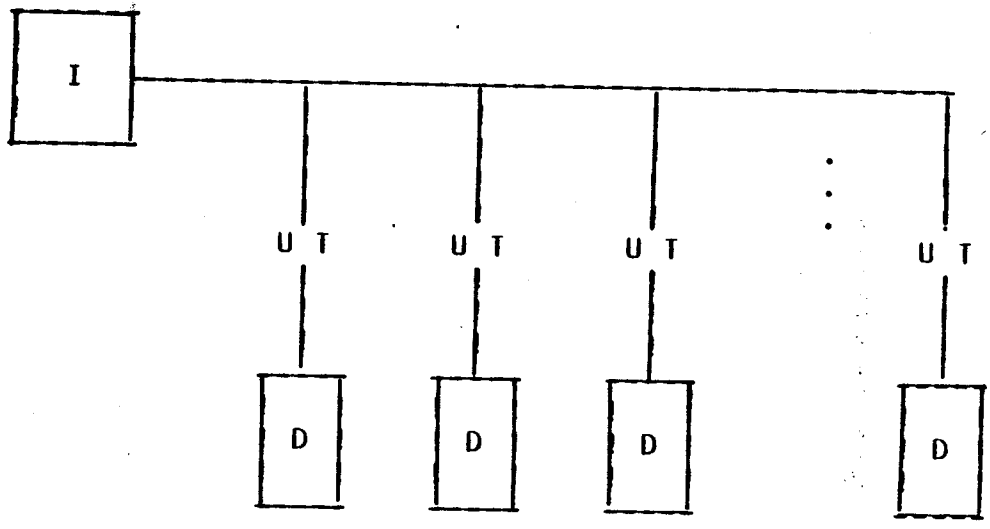
Tous les types classiques de parallélisme se retrouvent dans les machines à filtres:

SISD: machines PPD traitant une requête à la fois

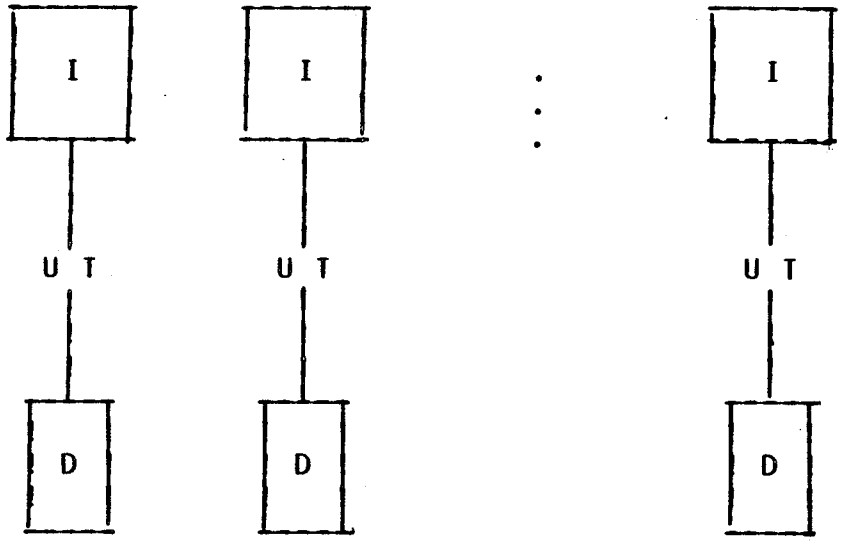
MISD: un même flot traité par plusieurs filtres effectuant des opérations différentes




SIMD: cas des machines traitant les mêmes opérations sur plusieurs pistes. Les machines PPT traitent une seule requête sur un cylindre. Cette architecture n'est efficace que sous des applications permettant d'exploiter la parallélisme (traitement de tableaux).




MIMD: chaque filtre traite une opération différente sur une piste différente. A l'opposé de l'architecture SIMD, il y a un problème de répartition des processeurs entre les tâches.




Remarque: la complexité de ces machines est croissante avec le nombre de modules de traitement.

2.4.2.3 EXEMPLES

MACHINES A PROCESSEURS PAR PISTES

**RAP**

RAP: Relational Associative Processor
Université de TORONTO
SCHUSTER, H.B. NGUYEN, OZKARAHAN, SMITH

RAP 1 et RAP 2 sont des dorsaux bâtis autour de l'idée des machines à processeurs par piste. L'architecture de base de RAP comprend un ensemble de cellules, un contrôleur central et une unité arithmétique.

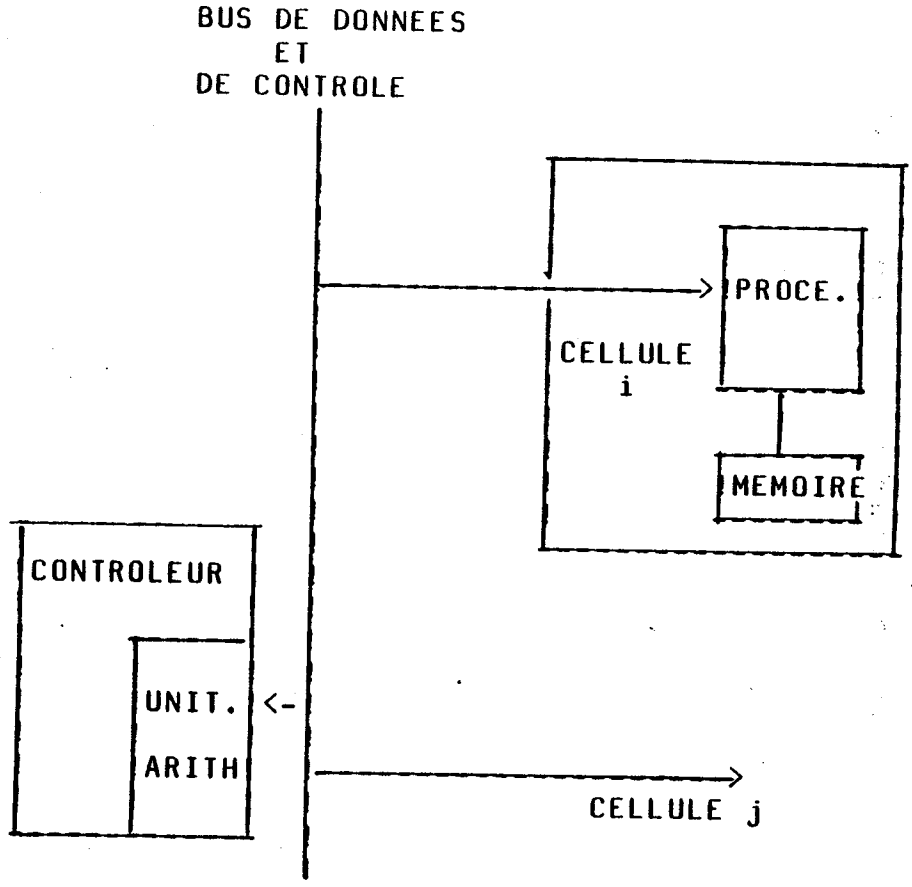



Figure 2.9 - ARCHITECTURE DE RAP

Chaque cellule est composée d'un processeur et d'une mémoire (avec une tête de lecture et une tête d'écriture). Ces processeurs exécuteront les primitives de recherche, d'insertion, de suppression et de mise à jour. La mémoire de chaque cellule pourra être considérée comme une antemémoire vis à vis de la mémoire secondaire. Elle peut être assimilée à une piste de disque, un CCD ou encore à une mémoire à bulles. L'unité arithmétique est destinée à exécuter les fonctions aggrégats sur les données contenues dans les mémoires des cellules.

Le contrôleur reçoit les instructions de la machine hôte dans un format RAP, les décode, les diffuse aux cellules et retourne les résultats à la machine hôte.

Chaque instruction est exécutée dans les cellules qui travaillent en parallèle directement sur leurs données.

Chaque mémoire contient les données ordonnées en séquence d'enregistrements contenant les valeurs des attributs.

Une cellule est composée de plusieurs unités logiques dont la plus importante est chargée de la recherche. Plusieurs éléments de comparaison pourront constituer la base de l'adressage par contenu de cette architecture. Les comparateurs peuvent tester le contenu d'un ou plusieurs attributs. Ils pourront exécuter des conjonctions ou des disjonctions.

Les utilisateurs peuvent communiquer avec RAP via un conversationnel ou un langage de programmation contenant des appels à RAP. C'est le frontal qui se charge de la transformation des différents langages de requêtes en un format RAP. Les fonctions du SGBD demeurent sur la machine hôte.

STRUCTURE DE DONNEES RAP

Du point de vue programmation, RAP stocke les données comme des occurences non ordonnées d'enregistrements définis par une "relation RAP".

| NOM RELATION | | | | | | | |
|---|---|---|---|---|---|---|---|
| M1 | M2 | .... | M6 | val. attr. | val. attr. | | val. |
| | | | | | | | |
| 1 | 0 | | 0 | xxxx | yyyy | | zzzz |
| | | | | | | | |

TUPLES

Chaque occurence de relation possède des bits de marquage Mi qui peuvent être positionés par des instructions de mar-

quage spécifiques ou par des opérations intermédiaires en vue de traitements futurs des tuples marqués. Ils sont utilisés entre autre comme zone de travail pour identifier les enregistrements qui ont satisfait une première opération et qui pourront être traités par les opérations suivantes. Ces Mi seront traités comme des attributs de tuple. Un Mi spécifique permet de repérer les enregistrements à détruire.

Les tuples d'une relation peuvent occuper une ou plusieurs mémoires mais chaque cellule ne peut contenir qu'une relation. La machine RAP pourra donc contenir dans les deux cas extrême soit une très grande relation soit N relations réparties sur les N cellules.

Le contrôleur dispose de plusieurs registres où il pourra stocker les données susceptibles d'être retraitées par d'autres instructions pour l'exécution de requêtes complexes.

## RAP ET SON ENVIRONNEMENT

Nous allons définir deux approches sur la façon de partager les données entre RAP et sa mémoire secondaire.

### Partionnement de la base

Cette approche part du principe qu'à un instant précis toutes les données n'ont pas la même probabilité d'être traitées après une requête quelconque. On rejoint ici les raisons qui ont motivé la création des mémoires cache dans les premiers accélérateurs disques. Ainsi les données pourront être stockées soit sur la mémoire secondaire soit dans RAP. Ceci impose de réaliser un mécanisme de transfert, soit automatique soit controlé par l'utilisateur, des données entre la mémoire secondaire et RAP.

Les requêtes seraient alors décomposées en deux sous requêtes, une pour RAP et une pour la mémoire secondaire. La sous requête RAP serait exécutée en premier et si certaines données manquent on exécutera la seconde sous requête sur le disque tout en se servant des résultats de la première afin de minimiser les recherches disque.

Pagination

Cette solution suppose que toutes les données soient stockées dans un format RAP. Celles-ci seront divisées en pages de taille égale à celle de la mémoire d'une cellule. Avant l'exécution, chaque requête est transférée dans un moniteur (sur le frontal). Celui-ci gère une table qui donne la localisation des pages pour chaque relation, analyse la requête pour repérer les pages à traiter et génère l'ordre de transfert entre la mémoire secondaire et RAP. La requête est alors transférée à RAP. Il serait préférable d'avoir une ligne de communication directe entre les disques et RAP afin d'éviter un passage des données par le frontal. A l'inverse du cas précédent, la requête est entièrement exécutée dans RAP.

PERFORMANCES

Dans le cas de requêtes mettant en cause plusieurs relations, les résultats dépendent fortement du fait qu'une des relations puissent être ou non stockée entièrement dans les CCD. On retrouve ici un des aspects contraignant des mémoires cache qui ne peuvent contenir toutes les données à traiter .

## MACHINE A PROCESSEUR PAR TETE

## DBC

DBC: DATA BASE COMPUTER
Université de l'OHIO DAVID K. HSIAO

DESCRIPTION

DBC pourra être également considéré comme une machine à processeurs spécialisés.

Il y avait quatre raisons principales à l'origine de la conception de DBC:

- gérer une base volumineuse en temps réel
- aboutir à un prototype avec les moyens technologiques de l'époque
- développer une machine avec un mécanisme de sécurité (protection, confidentialité) faisant partie intégrante du système
- posséder un langage de commande de haut niveau afin de s'interfacer avec différentes machines hôtes et de supporter les trois grands modèles de données.

DBC se sert de deux boucles pour l'exécution des commandes:

- la boucle de structure utilisée pour limiter l'espace de recherche, pour déterminer les autorisations d'accès aux enregistrements et pour mettre en oeuvre le mécanisme de "clustering" lors de l'insertion de

nouveaux enregistrements

- la boucle de données qui est utilisée pour stocker et accéder la base.

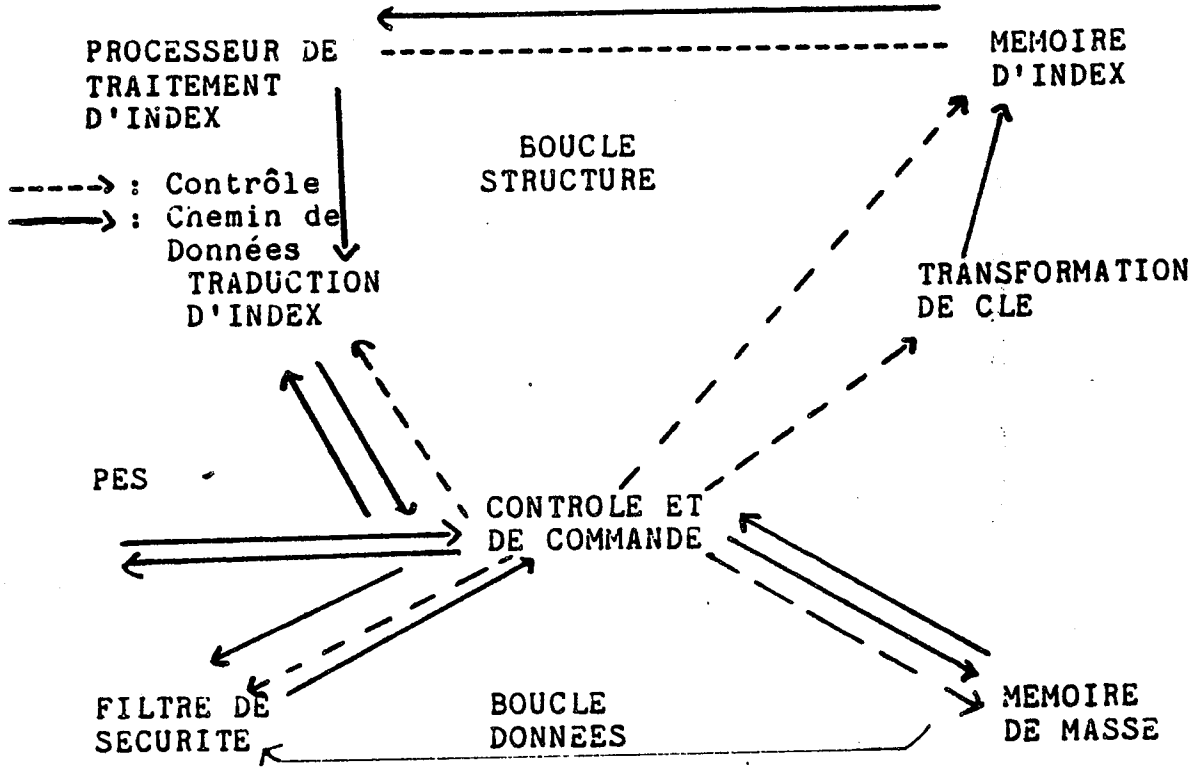



Figure 2.10 - Boucle de structure et boucle de données

DBC est conçu autour d'un disque à tête mobile afin de pouvoir lire un cylindre en une révolution. Les données seront adressées par leur contenu par un ensemble de processeurs dans la même révolution.

Prendre le cylindre comme unité semblait satisfaire les concepteurs de DBC puisqu'ils estimaient que le volume des transactions ne dépassait guère le cylindre. Tant que les enregistrements ne sont pas trop dispersés, on limitera le nombre de cylindres accédés. Cette dispersion pourrait être prévenue par un mécanisme de "clustering" qui aurait recours à certaines informations fournies par les créateurs de la base via l'interface hôte-dorsal.

En dehors du fait que DBC soit resté au stade de projet, son approche (dissociation du repérage des données et de l'accès de l'information) était suffisamment originale pour que l'on se soit attaché à la décrire. Elle fut l'une des rares machines à vouloir accorder de l'importance au rapprochement physique et à avoir proposeé une solution, bien que partielle, à cet aspect pour lequel on ne porte encore que trop peu d'intérêt aussi bien pour les SGBD que les MBD d'aujourd'hui.

## MACHINES A PROCESSEUR PAR DISQUE

Nous allons décrire l'IDM 500 car elle se trouve être la principale concurrente de l'IDBP d'INTEL, qui sera également décrite, sur laquelle sera réalisé l'ensemble de nos tests. De plus nous aborderons le projet de l'INRIA qui est en cours de réalisation et qui présente quelques originalités intéressantes: VERSO, ainsi que la machine de gestion de bases de données navigationnelles et relationnelles, le DORSAL 32.

### IDM 500

IDM 500: INTELLIGENT DATABASE MACHINE
BRITTON-LEE INC.
USA

L'IDM 500 est une machine base de données conçue et commercialisée par BRITTON-LEE Inc. . Elle fait partie des rares machines qui ont vu le jour et qui par la suite ont été introduites sur le marché des machines spécialisées.

Ses concepteurs ont voulu s'adresser à un marché intermé-

diaire entre les petites applications de gestion et les très grosses applications scientifiques. En choisissant ce marché, il était indispensable d'obtenir un bon compromis coût/performance et d'être indépendant de la machine hôte.

L'IDM 500 est un système qui prend en charge toutes les tâches spécifiques de la gestion des bases. Ces performances sont obtenues par l'utilisation de matériel spécialisé. Ses composants sont:

- un processeurs (10 MIPS) qui réalise les fonctions du SGBD
- un contrôleur disque
- une mémoire RAM
- un processeur d'E/S
- un bus à grande vitesse
- un accélérateur disque.

L'IDM 500 supporte le modèle relationnel. Il pourra être utilisé seul (éventuellement avec des terminaux intelligents) comme système autonome ou avec des machines hôtes comme dorsal. Dans le second cas, plusieurs machines hôtes pourront se partager l'accès à la base étant donné que la gestion des accès concurrents est réalisée sur le dorsal.

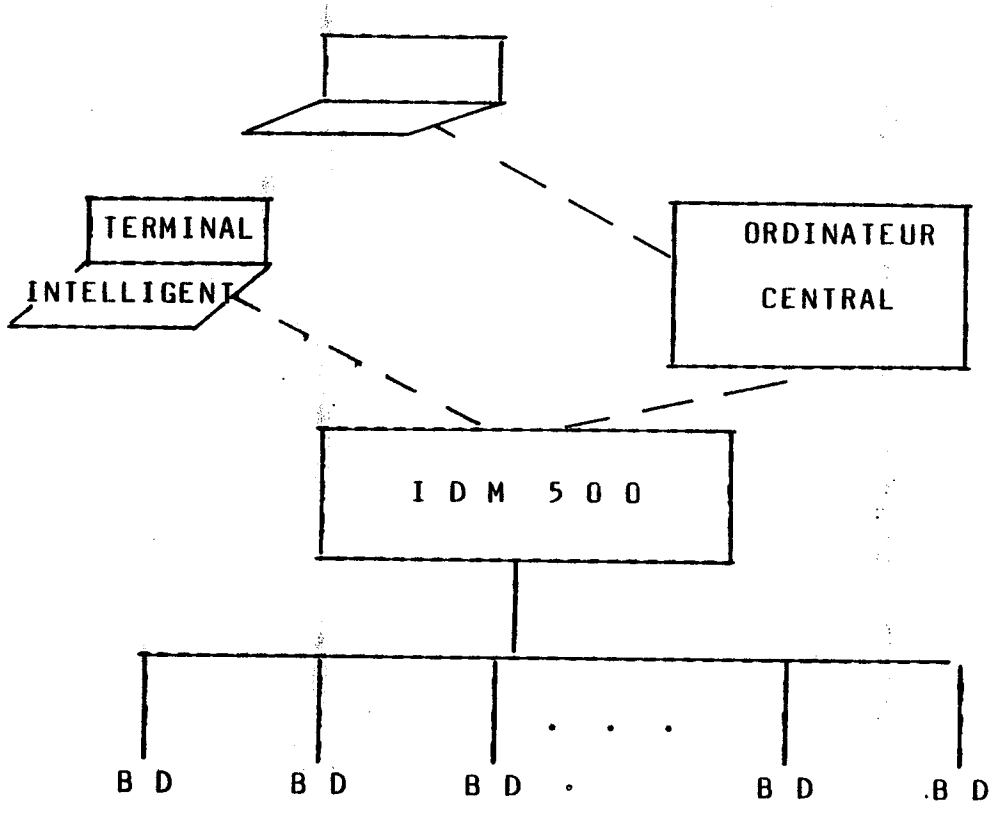



Figure 2.11 - CONFIGURATION D'UTILISATION DE L'IDM 500

L'IDM 500 comporte un SGBD complet. Les commandes d'accès à la base sont pré-traitées dans la machine hôte (dans le cas où le dorsal est utilisé avec un frontal) et envoyées au dorsal. Ce dernier les exécute et mémorise les résultats jusqu'à ce que l'hôte soit prêt à les recevoir. L'utilisateur ne peut plus intervenir à partir du moment où la commande se situe dans le dorsal.

L'architecture physique de l'IDM 500

Ce dorsal est organisé autour d'un bus à très grande vitesse. Il est divisé en modules ayant chacun leur fonctionalité propre. Les composants matériels sont:

- le processeur central: il gère toutes les ressources du système et exécute la majeure partie du logiciel

- le processeur d'E/S: il accepte les commandes de l'hôte et lui retourne les résultats. Il controle que tout a été bien envoyé et bien reçu. Il convertit les types de données du frontal dans un format IDM.

- le contrôleur disque: il transfert les données entre les disques et la mémoire principale de l'IDM. La page a une taille de 2 Ko. Un contrôleur gère entre 1 et 4 unités de disques.

- un contrôleur disque: c'est un processeur muni d'un ensemble d'instructions spécialement conçues pour les BD relationnelles. Il travaille sous les ordres du processeur central et traite les données au débit du disque.

- une mémoire de stockage: l'IDM pourra posséder jusqu'à 12 cartes de mémoire.

L'interface utilisateur

En mode dorsal, l'IDM 500 dépend de son frontal pour communiquer avec les utilisateurs. Les utilisateurs finaux éxécutent une commande derrière leurs terminaux ou en batch. Le programme de translation transforme la requête source en un format compréhensible par l'IDM. Elles est alors envoyée à l'IDM en vue de son exécution. Les résultats sont retournés à l'interface en vue d'être reformatés de façon à être compréhensibles par l'utilisateur.

Conclusion

Une des originalités de l'IDM 500 repose sur le fait que le choix de la configuration est entièrement entre les mains de l'utilisateur: IDM 500 comme MBD ou IDM 500 comme systè-

me autonome. Derrière le choix de BRITTON-LEE on ressent très nettement les impératis commerciaux de la firme californienne. On se tourne de plus en plus vers des machines entièrement indépendantes de la machine hôte. Pour cela, la tendance actuelle opte en faveur d'un déchargement optimal du SGBD sur la MBD afin de ne conserver qu'un noyau, assez réduit, sur la machine hôte qui se chargera de l'interface.

## iDBP

INTEL C.
CALIFORNIE USA

La machine base de données iDBP exécute toutes les fonctions de gestion des bases pour les programmes d'application lancés sur des machines hôtes. Située entre le ou les machines hôtes et les unités de stockage, elle aura des relations de type esclave-maître avec les sites hôtes. Elle pourra être assimilée à un serveur intelligent mais malgré tout, elle aura davantage de fonctionalité.

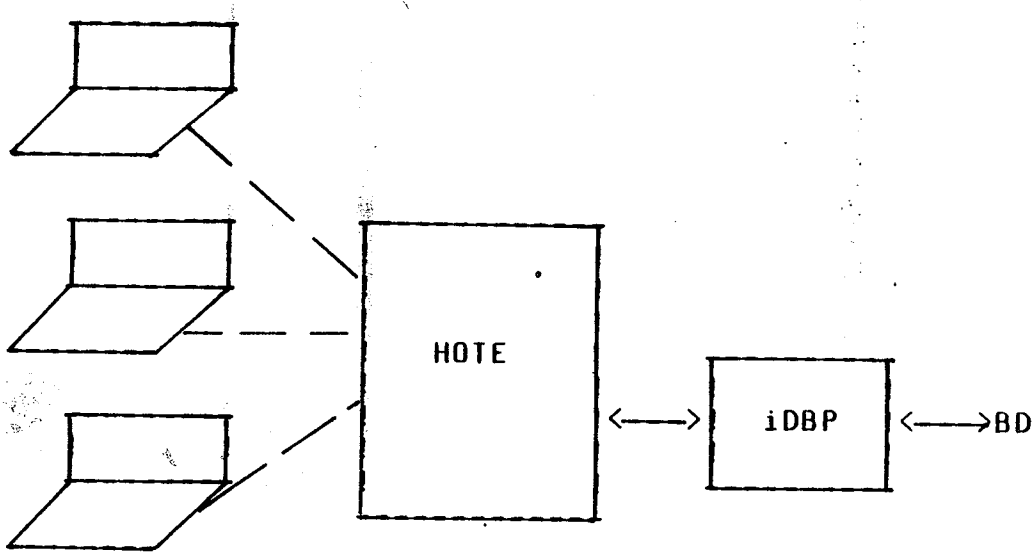



Figure 2.12 - CONFIGURATION D'UTILISATION DE L'iDBP

L'iDBP fournit un noyau de base du SGBD et ce sera à la machine hôte de proposer la ou les couches supplémentaires qui permettront le dialogue avec les utilisateurs finaux selon le type d'application mis en oeuvre.

L'iDBP est soumis aux commandes émises par les machines hôtes. Une séquence de celles-ci sera regroupée dans un module de requêtes. A chaque module de requête, le dorsal répondra par un module de réponses qui contiendra une réponse pour chacune des commandes contenues dans le module de requêtes. Une réponse pourra être un message d'erreur ou d'acquittement spécifiant la réussite de l'opération ou les données répondant aux critères de la requête.

L'interprétation du module de requêtes est à la charge du site hôte.

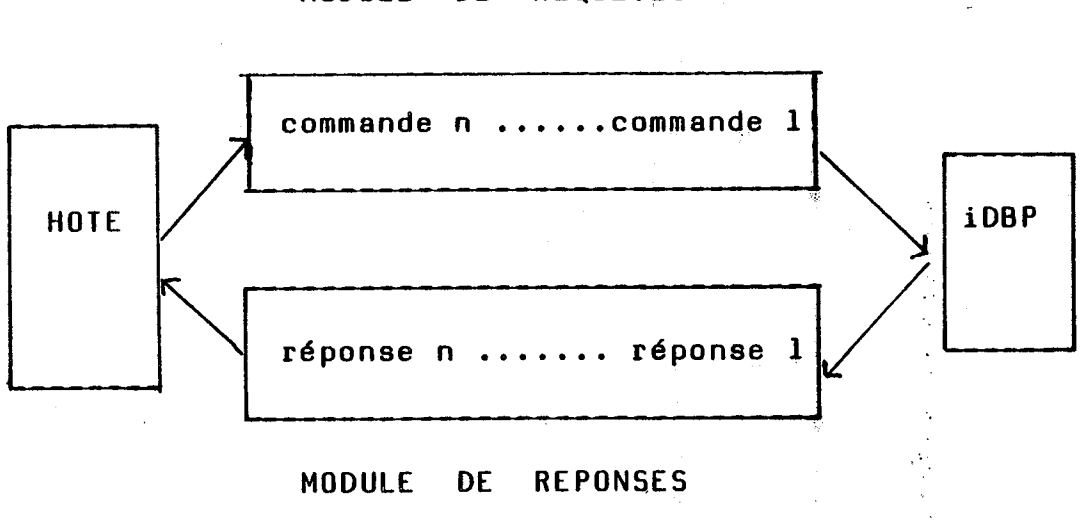



Figure 2.13 - ECHANGES HOTE - iDBP

Au niveau de la conception générale des machines, les ingénieurs de BRITTON-LEE ont fait davantage d'efforts quant à l'architecture de leur machine. L'IDM 500 possède un matériel plus puissant au niveau recherche, en offrant la possibilité d'adjoindre un accélérateur disque. Par contre, l'iDBP propose davantage de souplesse, aux utilisateurs, vis à vis de la gestion des données (fichiers struc-

turés pouvant être liés à des fichiers non structurés,...) et amorce une orientation vers la manipulation de données généralisées.

## DESCRIPTION MATERIELLE DE L'iDBP

L'iDBP ou "INTEL DATA BASE PROCESSOR" est conçu utour du calculateur 86/330A de INTEL et a été renommé 86/440A. L'iDBP 86/440A est composé :

- d'un processeur
- de cartes mémoires
- d'une interface de communications
- d'un controleur disques (1 à 3 disques de 35 Mo chacun)
- d'une unité de disquettes 8 pouces

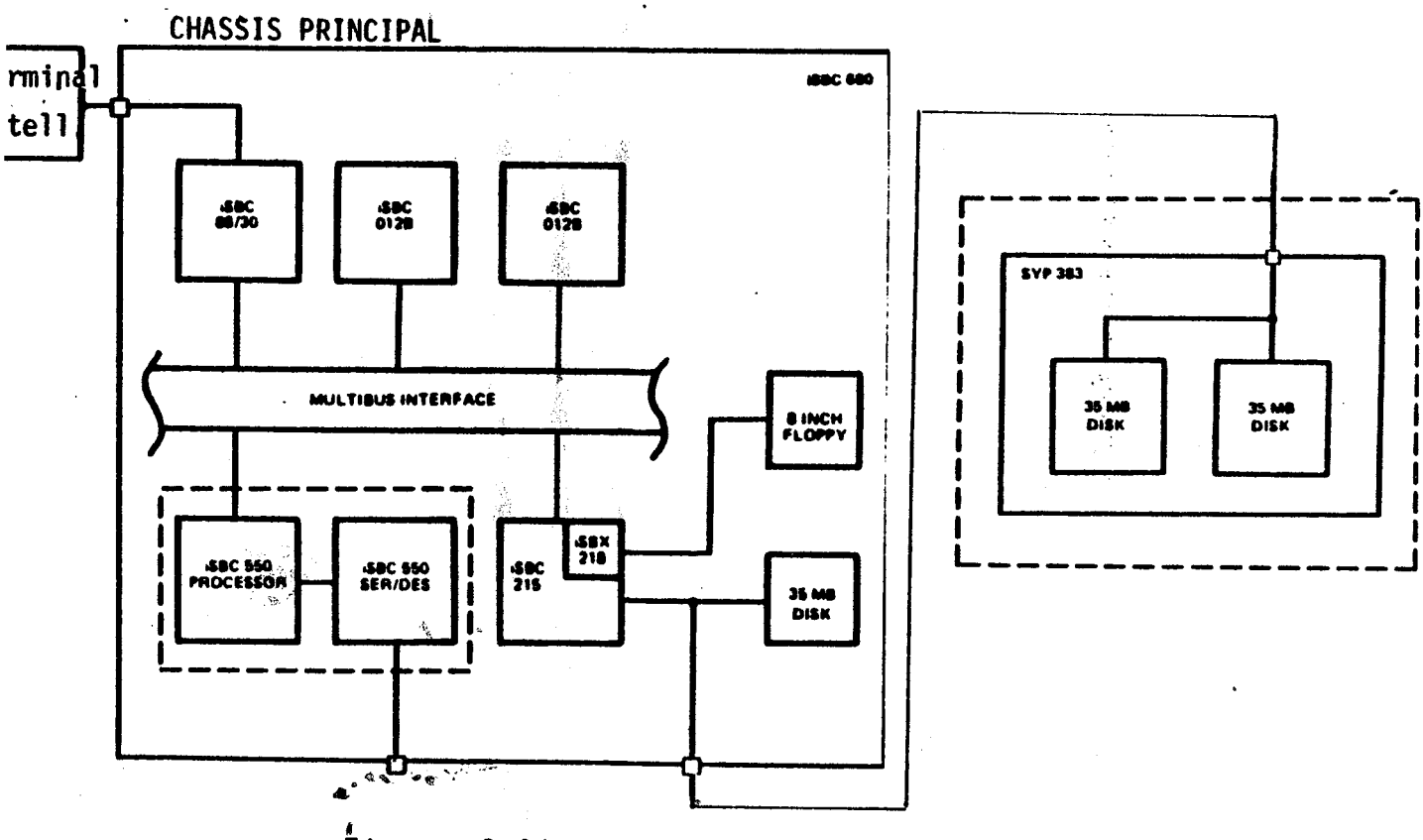



Figure 2.14 - Architecture de l'iDBP

Le 86/440A est organisé logiquement en un ensemble de sous-systèmes qui réalisent chacun une série de fonctions.

LE SOUS-SYSTEME PROCESSEUR

La carte iSBC 86/30 éxécutera toutes les fonctions du système d'exploitation et du SGBD. Elle controlera également les accès à chacun des autres sous-systèmes à savoir la mémoire, la mémoire secondaire, le système de communication avec le calculateur hôte.

L'iSBC 86/30 est bâti autour du microprocesseur 16 bits. 8086 qui devrait être remplacé dans le futur par l'iAPX 286.

LE SOUS-SYSTEME MEMOIRE

La version qui nous a été livrée comportait 1 Mo de mémoire principale.

LE SOUS-SYSTEME MEMOIRE DE MASSE

Ce sous-système est composé d'un controleur disque iSBC 215 et peut comporter jusqu'à trois disques WINCHESTER de 35 Mo dont un est intégré au chassis principal et les deux autres pourront être adjoints au sein d'un chassis périphérique. On pourra ainsi disposer de 105 Mo.

Le temps moyen d'accès est de 45 ms avec un extrême à 90 ms, le temps de déplacement d'une piste à l'autre étant de 10 ms.

LE SOUS-SYSTEME DISQUE SOUPLE

Il est composé d'un controleur iSBC 218 et d'une unité de disquette 8 pouces (double face double densité de 1 Mo).

LE SOUS-SYSTEME DE COMMUNICATION HOTE

Ce sous-système est composé d'un controleur de communication (iSBC 550) ETHERNET qui assurera la liaison avec un réseau de type ETHERNET. L'iSBC 550 comporte entre autre un microprocesseur 8088.

**VERSO**

VERSO: INRIA

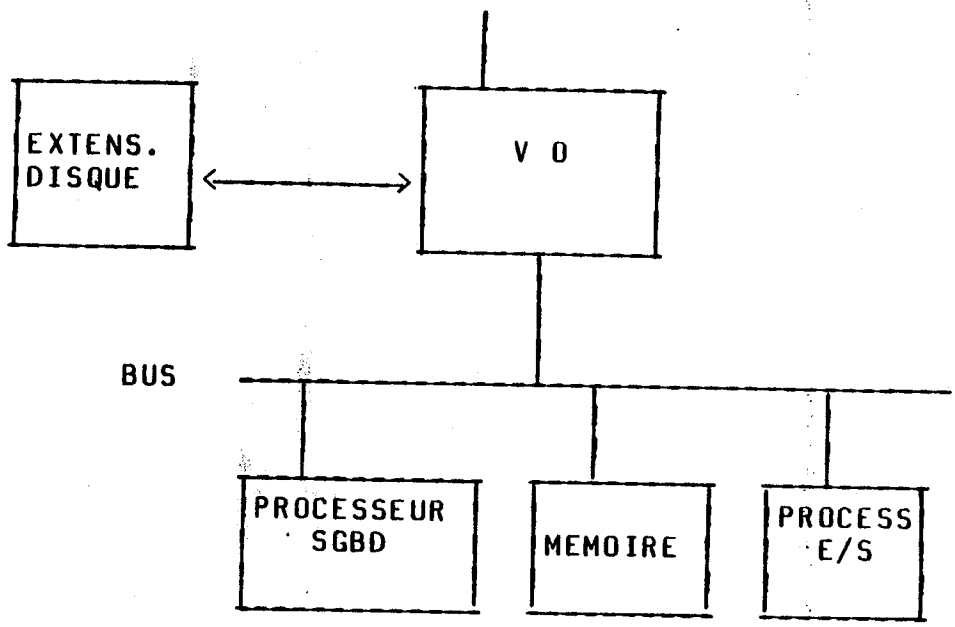



Figure 2.15 - ARCHITECTURE DE VERSO

"La base est répartie entre un disque et une extension disque réalisée en RAM et contenant les index, le schéma et une partie des relations temporaires.

Le processeur VO contient:

- un filtre qui peut effectuer des opérations de sélection, d'insertion et de suppression sur des relations et des opérations de type fusion (jointure...) sur des relations triées,

- la gestion du disque et son extension.

Le SGBD est résident sur un processeur standard (MOTOROLA 68000) communiquant avec VO via un bus (VERSABUS)."

Une des particularités de VERSO repose dans la forme parenthésée des relations. Celles-ci sont stockées dans un format qui permet de limiter la taille des relations. Ceci est fort appréciable à la fois pour le gain de place mais aussi pour le gain de traitement qui en découle. On limite ainsi le temps de parcours des relations en réduisant l'espace des données à traiter. De plus on est assuré que les données sont triées dans l'ordre des éléments mis en facteurs successivement. Cette particularité est à relever car elle fait preuve d'originalité vis à vis des structures de données classiques que l'on retrouve dans chaque nouvelle proposition de MBD. Nous rejoignons le point développé au début de ce chapître où l'on a fait mention du rôle que peut jouer une représentation judicieuse des données.

Ce parenthésage sera étendu aux index dont on cherche à limiter la taille afin de les rendre résident en M.C.. Ils pourront alors être chargé dés l'ouverture de la base.

A ce jour, une émulation de VERSO est opérationnelle sous MULTICS. Un compilateur du filtre a été réalisé afin de pouvoir générer l'automate.

## DORSAL 32

DORSAL 32
COPERNIQUE

Le DORSAL 32 est un ordinateur spécialisé qui prend en charge la gestion physique des disques et les fonctions de gestion des bases de données. Il gère deux modèles externes: le modèle navigationnel et le modèle relationnel. A ces deux modèles externes correspond un seul modèle interne: une même base de données peut être utilisée concurremment en modes navigationnels et relationnels.

### ARCHITECTURE GENERALE DU DORSAL

Le DORSAL 32 comportera deux machines principales, chacune ayant une structure multi-processeurs:

- la machine de gestion logique (MGL)

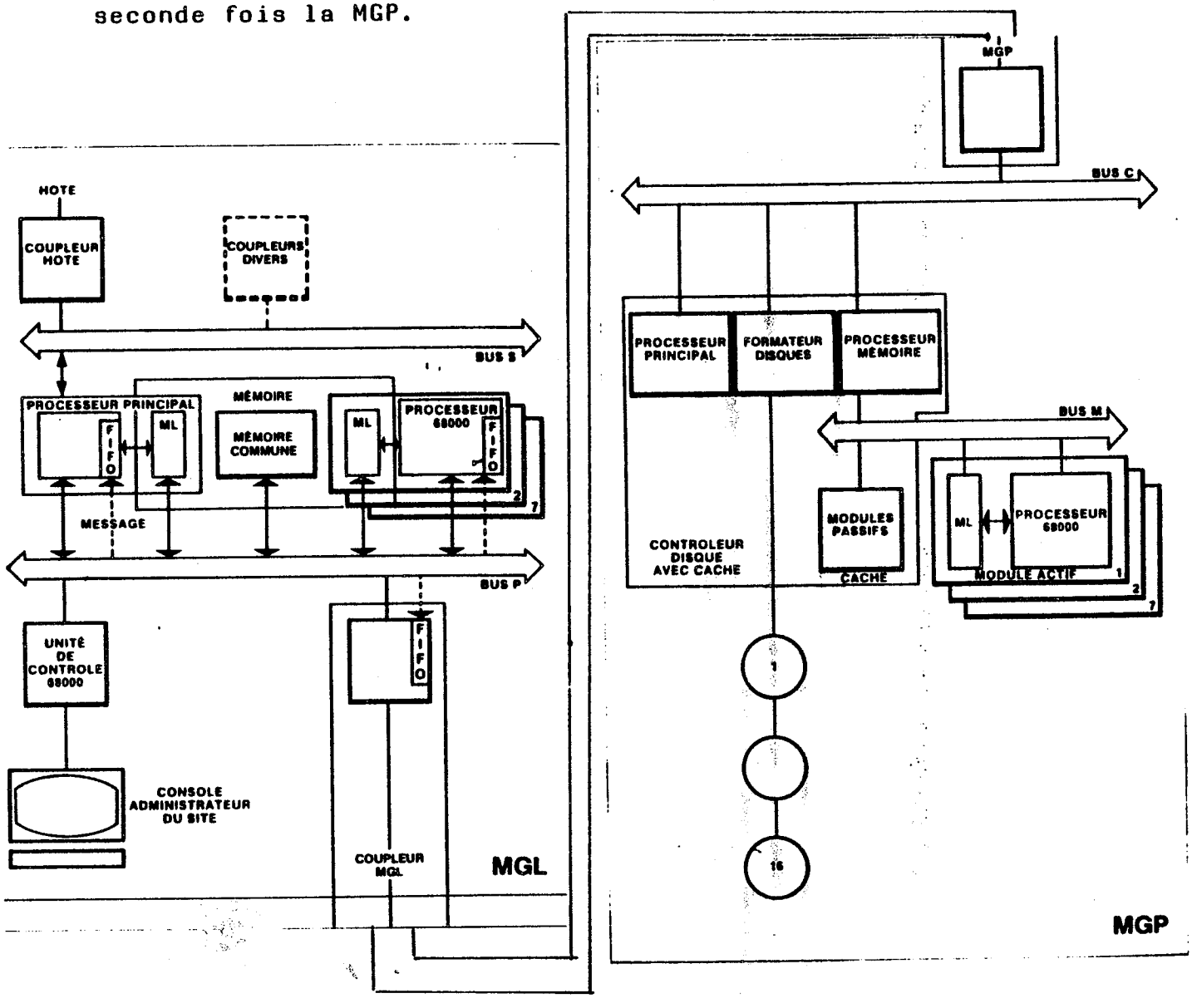
- la machine de gestion physique (MGP).

La MGL sera la machine d'exécution du SGBD et la MGP, la machine de gestion des disques qui sera en fait l'accélérateur disque DIRAM 32 <COPER 2>. Cette dernière sera dotée d'une mémoire cache et de modules de recherches associatives utilisés pour les méthodes d'accès relationnelles.

La MGL sera multi-processeurs, à base de micro-processeurs "68000".

La MGP sera également multi-processeurs et pourra comporter entre 0 et 7 modules actifs, chacun étant bâtis autour d'un "68000" et disposant de 256 KO de mémoire privée.

La mémoire cache sera constituée par les modules passifs(cf schéma). Par contre, les modules actifs ne pourront travailler directement sur ces modules passifs et on perd ainsi, une grande partie des avantages de la mémoire cache. D'un autre côté, les modules actifs travailleront uniquement en lecture. Ainsi lors d'une opération de mise à jour, ils accéderont les données, les transmettront à la MGL qui demandera alors leur mise à jour, en parcourant une seconde fois la MGP.




SCHEMA DU DORSAL 32

Cette machine n'est pas encore disponible dans sa version définitive mais fait preuve d'une certaine originalité.

Elle va plus loin que déporter les fonctions d'un SGBD sur un calculateur situé entre un site hôte et les unités de stockage. Elle a su décomposer d'un côté, tout ce qui était gestion de la base et d'un autre côté, la partie accès à l'information. Elle profite ainsi du parallélisme inhérent à ce partage des tâches. D'autre part, ses concepteurs ont très bien ressenti qu'il ne fallait pas se contenter d'écrire un SGBD sur un calculateur dorsal pour réaliser une machine base de données. Un module matériel d'accélération de recherche, ici le DIRAM 32, est indispensable à l'amélioration des performances.

## 2.4.3 ARCHITECTURES MIMD

### 2.4.3.1 DESCRIPTION

Cette architecture dissocie logique et mémoire de masse et essaie de tirer partie au maximum du parallélisme. L'architecture MIMD regroupe les architectures de type multiprocesseurs, indépendants et effectuant des tâches différentes sur des données différentes. Ces machines sont bâties autour d'une hiérarchie de mémoires.

### 2.4.3.2 CLASSIFICATION DES MACHINES MIMD

BANCILHON a décelé quatre paramètres essentiels pour caractériser les machines MIMD:

- degré de parallélisme
- degré de spécialisation des processeurs
- degré d'hétérogénéité de la machine
- mode de connexion entre processeurs.

2.4.3.2.1. Degré de parallélisme

Pour beaucoup, la solution au problèmes de performances passe par le parallélisme et comme l'amélioration des performances est le but des MBD, il est normal qu'ils se soient tournés vers l'approche machine parallèle.

Les SGBD ont été conçus entre autre pour pouvoir traiter un certain nombre de transactions simultanément. Chacune de ces transactions se décompose en un certain nombre d'étapes dont l'interrogation et la mise à jour. Chaque étape fait appel à de nombreux modules: gestion de la mémoire, des chemins d'accès, de la concurrence, des reprises. Nous pouvons alors dégager quatre degrés de parallélisme:

- entre transactions
- entre étapes d'une transaction
- à l'intérieur d'une même étape entre sous tâches
- en sous traitant une même sous tâche à plusieurs processeurs.

2.4.3.2.2 Degré de spécialisation des processeurs

Les processeurs de la machine peuvent être tous banalisés, toute tâche peut être alors sous traitée à tout processeur.

Ils peuvent aussi être spécialisés fonctionnellement:

- tel processeur gère les relations avec le frontal
- tel processeur gère les problèmes d'intégrité
- tel processeur gère les chemins d'accès.

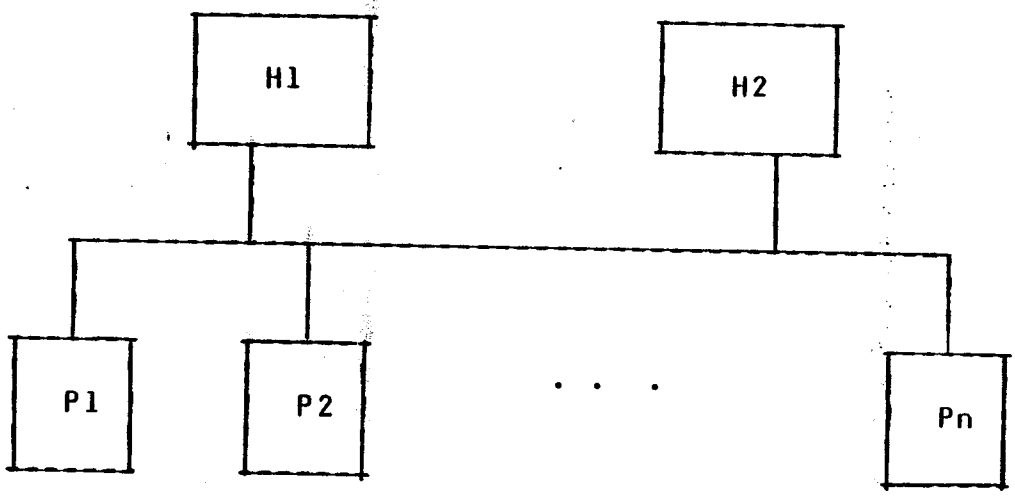
INFOPLEX (MADNICK) a été la première machine basée sur un réseau de microprocesseurs destinés au traitement des fonctions spécifiques de la base. Elle est bâtie autour d'une hiérarchie de mémoires et de microprocesseurs afin d'exploiter le parallélisme inhérent aux accès concurrents à la base. STONEBRAKER a proposé une MBD distribuée où il utilise des mini/micro conventionnels afin de décharger l'ordinateur central des fonctions de la base.

2.4.3.2.3 Degré d'homogénéité des processeurs

"Dans le cas de processeurs spécialisés fonctionnellement, on peut, mais ce n'est pas indispensable, concevoir le processeur en fonction de la tâche qu'il a à accomplir."

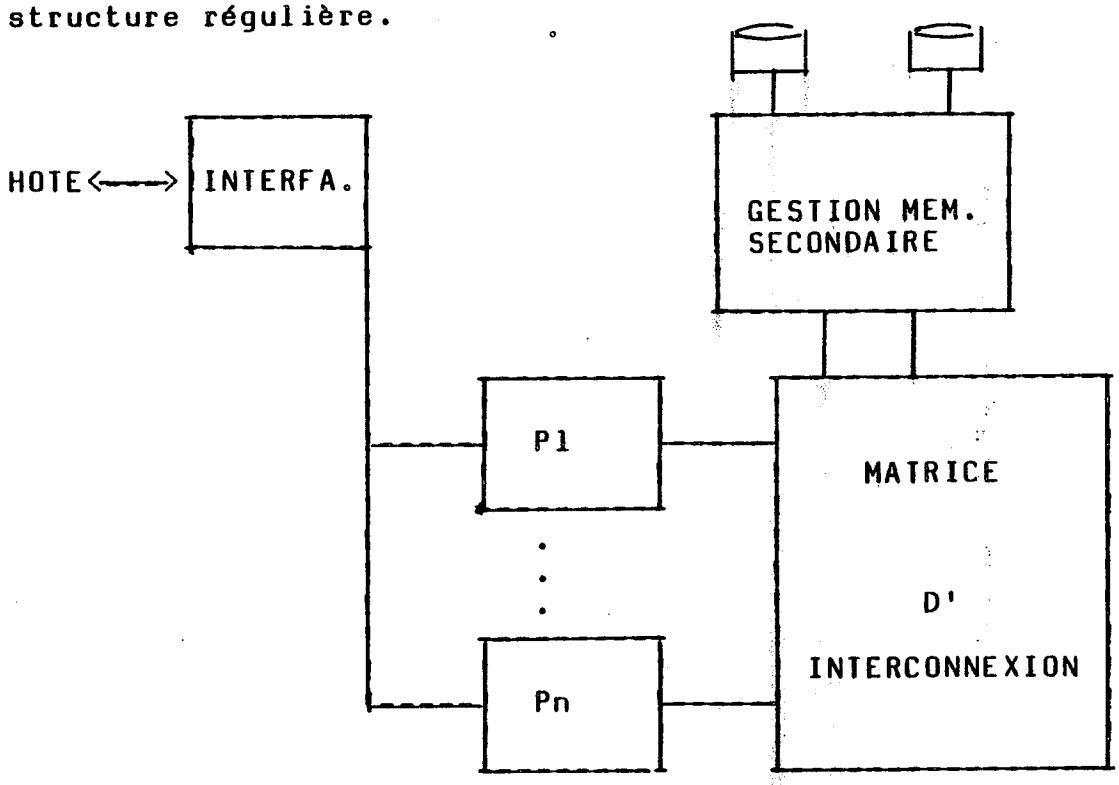
2.4.3.2.4 Mode de connexion des processeurs

On peut concevoir la machine comme un système distribué de machines.



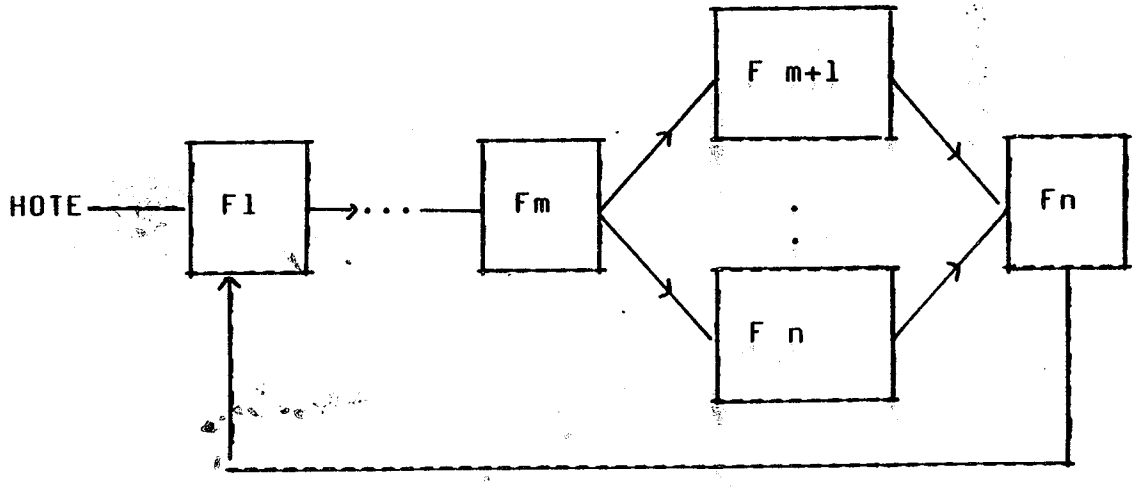

Les hôtes Hi communiquent avec les processeurs Pi base de données via un réseau.

A l'opposé on trouve les systèmes fortement couplés à structure régulière.




Il s'agit d'une structure homogène à processeurs non spécialisés. L'interconnexion peut être réalisée entre autre par un bus.

Enfin, les structures hétérogènes ont souvent donné lieu à des architectures de type "pipeline".



Exemples de machines MIMD:

- à processeurs spécialisés: RDBM, DBC, SABRE

- à processeurs banalisés: DBMAC, DIRECT.

2.4.3.2 EXEMPLES

**DIRECT**

DIRECT
Université du WISCONSIN
D. DEWITT

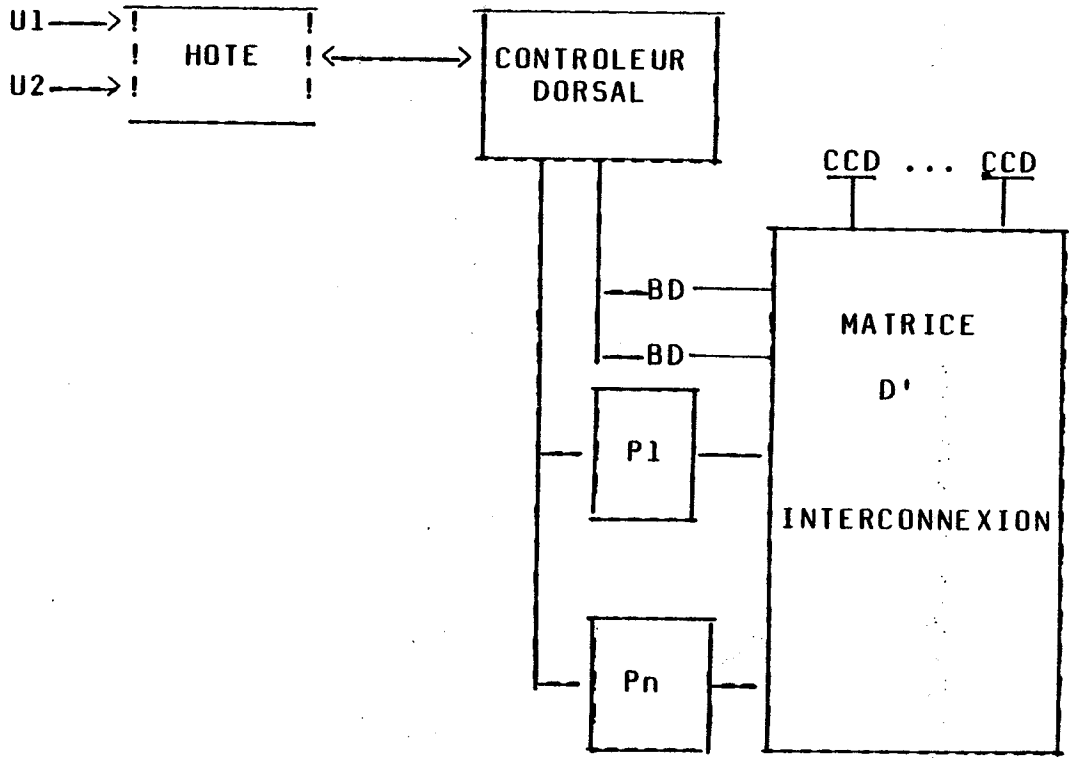



Figure 2.16 ARCHITECTURE GENERALE DE DIRECT

L'architecture générale de DIRECT est bâtie autour de six composants principaux:

- un ordinateur hôte PDP 11/45 avec INGRES

- un controleur dorsal: PDP 11/40

- un ensemble de huit processeurs pour le tratement des requêtes: PDP 11/03, 28 K MOTS

- un ensemble de CCD comme mémoire associative

- une matrice d'interconnexion entre les CCD et les PDP 11/03

- une ou plusieurs mémoires de masse.

L'ordinateur hôte gère les communications avec l'utilisateur. Ainsi lorsque ces derniers voudront accéder à une base, INGRES compilera les requêtes et transmettra la séquence d'opérateurs algébriques relationnels au contrôleur dorsal.

Le contrôleur dorsal est responsable des communications avec la machine hôte et du contrôle des processeurs de requêtes. Lorsqu'il reçoit un paquet de requête, il détermine le nombre optimal de processeurs à lui allouer. Si les relations référencées par les requêtes ne sont pas dans la mémoire associative, c'est à lui de les récupérer dans la mémoire de masse et de transmettre l'ordre de chargement dans les CCD.

Les processeurs de requêtes sont chargés du traitement des requêtes sur les données stockées dans un des CCD.

Puisque l'architecture de DIRECT est de type MIMD, ce système supporte la concurrence inter et intra requête. Pour le parallélisme intra equête, chaque relation est divisée en pages de taille fixe. Chaque processeur effectuera une recherche associative sur les tuples des relations référencées par la requête. Lorsqu'il a fini de parcourir une page il demande au contrôleur la page suivante. Ainsi les opéra-

tions du contrôleur ne doivent pas être partagées afin de garantir que tous les processeurs affectés à une même requête examinent une page différente. Les échanges entre les CCD et les processeurs se font par la matrice d'interconnexion.

Pour faciliter la concurrence inter requête, la matrice doit permettre à deux processeurs traitant deux requêtes différentes de pouvoir accéder la même page d'une relation. Ainsi en réduisant les copies des relations dans les CCD, on économise de la place en mémoire associative, on réduit les accès à la mémoire de masse et on évite les problèmes de mise à jour des copies multiples de relation.

Conclusion

La structure MIMD de DIRECT profite au maximum du potentiel de travail des processeurs de requêtes. Cette architecture est plus que nécessaire dans la mesure où les utilisateurs sont de plus en plus nombreux à partager des ressources communes et à vouloir éxécuter des requêtes complexes (jointure, aggrégat).

Différentes évaluations de performances de ce prototype ont été réalisées. Les concepteurs de DIRECT en sont arrivés à la conclusion suivante: vouloir accroitre le parallélisme entre les traitements sans tenir compte du coût des communications entre processeurs ne conduit pas aux performances escomptées. Ceci est principalement dû à l'architecture de DIRECT avec son contrôleur centralisé.

En effet ils ont montré que les coûts de communication entre processeurs dépassaient le temps consommé par ces mêmes processeurs pour traiter les requêtes. Cette première conclusion pourrait nous pousser à dire qu'une architecture MIMD présente certaines limites en particulier au niveau des communications intra processeurs et de la synchronisa-

tion des tâches. DE WITT et son équipe désiraient arriver à un contrôle distribué des requêtes afin qu'aucun composant ne devienne un goulot d'étranglement.

## SABRE

SABRE
INRIA

En cours de réalisation à l'INRIA, SABRE est une machine à processeurs spécialisés.

L'objectif du projet est de résoudre les problèmes de performances des SGBD et de leur intégration au sein d'un système réparti. Le système utilise des filtres, une mémoire principale partagée et des processeurs spécialisés de jointure, de tri, de décomposition de requête et de journalisation.

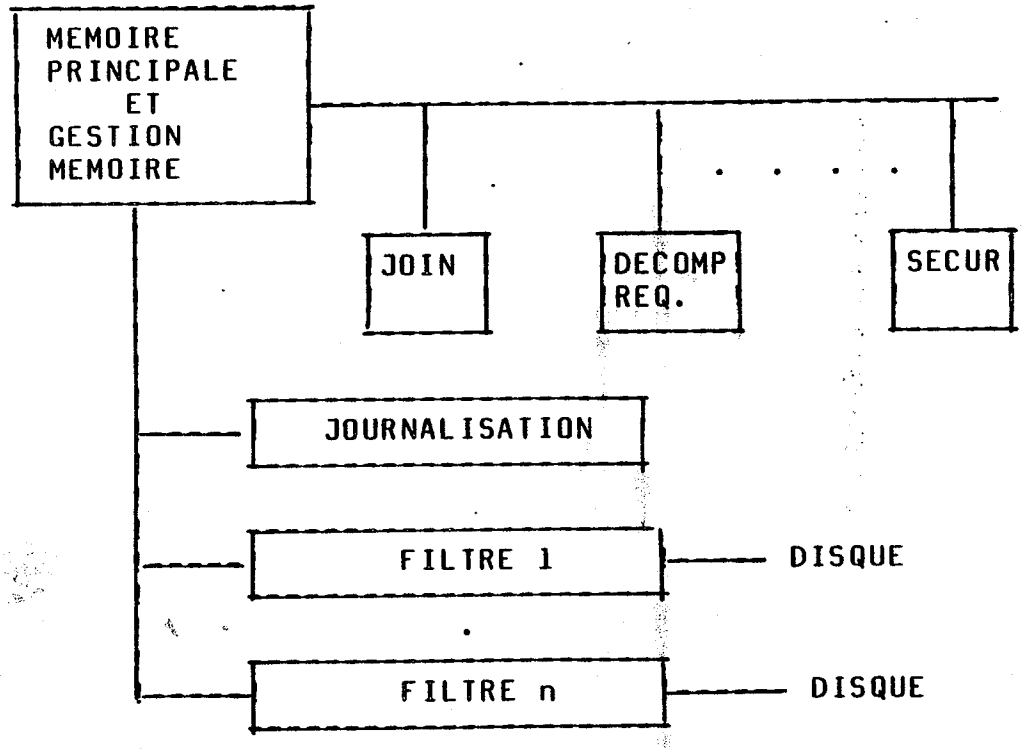



Figure 2.14 Architecture de SABRE

Le système est opérationnel sur MULTICS à l'INRIA. Une

version sur MICROMEGA et SM90 est en cours.

## REMARQUE

Une description plus complète de ces architectures et des prototypes a été faite dans <BUR 83>.

TABLEAU DE SYNTHESE DES MACHINES BASES DE DONNEES

| PROCESSEURS ASSOCIATIFS | MEMOIRES ASSOCIATIVES | MACHINES MIMD |
|---|---|---|
| - STARAN : Goodyear Ine (R)<br>- PROPAL2 : CIMSA (R) | - PROCESSEUR PAR PISTE :<br>RAP 1 (R) : Université de Toronto<br>RAP 2 : " "<br>CASSM (R) : Université de Floride<br>- PROCESSEUR PAR TETE :<br>SARI (R) : CINTRA<br>SURE (R) : Université de Brauschweig<br>DBC : Université de l'Ohio<br>- PROCESSEUR PAR DISQUE :<br>IDM 500 (R) : Britten-Lee Inc.<br>CAFS (R) : ICL<br>IDBP (R) : INTEL C.<br>VERSO (R) : INRIA<br>DORSAL 32 (R) : COPERNIQUE<br>MAGE : IMAG<br>- FILTRAGE AU VOL PUR :<br>TREFLE (R) : FRANCE | - PROCESSEURS SPECIALISES :<br>RDBM (R) : Université de Brauscheig<br>DBC : Université de l'Ohio<br>SABRE (R): INRIA<br>- PROCESSEURS BANALISES :<br>DBMAC (R) : CNR de Rome<br>DIRECT (R) : Université de Wisconsin |

: réalisé ou en cours de réalisation.

# C H A P I T R E  3

# EXPERIENCE AUTOUR DE SOCRATE/CLIO

# ET DE L'iDBP

## 3.1 PRESENTATION GENERALE DE L'EXPERIMENTATION

### 3.1.1 APPROCHE DE L'EXPERIMENTATION

#### 3.1.1.1 MOTIVATIONS

Au cours du Chapître 2, nous avons présenté les principes de base qui conféraient aux machines bases de données un rôle d'accélérateur pour les opérations d'accès aux données. A l'issue de ces années de spécifications de prototypes, nous pensons qu'il est de bon escient, d'évaluer ce qu'une telle architecture, intégrant matériel et logiciel, est susceptible d'apporter à un SGBD purement logiciel. Ces nouvelles configurations sont très controversées et seule, une telle expérience pourra apporter une preuve tangible à la remise en cause, ou non, des architectures des SGBD traditionnels.

### 3.1.1.2 HISTORIQUE

Il n'y a eu, jusqu'à présent et à notre connaissance, que très peu d'études concrètes visant à atteindre un but similaire à celui de notre expérimentation.

Des études d'estimation de performances ont été réalisées par <DEWHAW 81>. Après avoir décomposé chacune des requêtes participant au test, ils estimaient leurs temps sur la configuration retenue et concluaient. D'autre part, les performances de certaines machines telle que l'IDM, ont été évaluées mais sans être comparées à celles d'un SGBD traditionnel <RIE 83> <SCH 83>. L'iDBP n'a, quant à lui, jamais subi de tels tests.

Seules les études de l'équipe de David J. DeWitt, à l'université du Wisconsin, se sont véritablement intéressées à l'apport d'une machine bases de données à une architecture conventionnelle de SGBD <DEWBITT 83>. Leur analyse portait sur les systèmes suivants:

- INGRES - version universitaire sur VAX 750 sous UNIX
  - version commercialisée sur VAX 750 sous VMS
- ORACLE sur VAX 750 sous UNIX
- IDM 500 connecté à un PDP 11/70 sous UNIX. L'IDM a été utilisé avec, puis sans accélérateur disque.
- DIRECT connecté à un VAX 750. La partie multi-processeurs de DIRECT était composée de 4 LSI 11/23.

Les mesures de ce test ont été réalisées dans un contexte mono-utilisateur sur la base de test. Une expérience multi-utilisateurs a été réalisée, toujours par la même équipe,

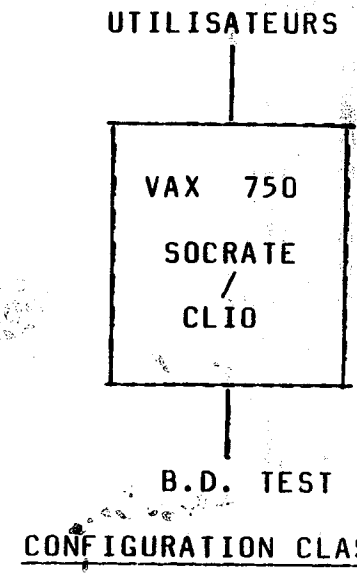
au début de l'année 1984 <DEWBOR 84>.

### 3.1.1.3 PRESENTATION DU CONTEXTE DE L'EXPERIMENTATION

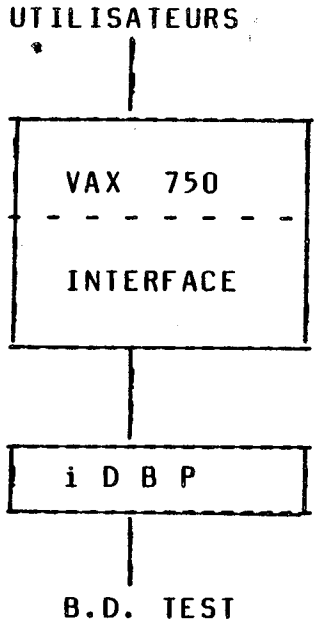
La pré-étude des besoins des nouvelles applications a été réalisée dans le cadre de la première étape du projet TIGRE de l'IMAG <TIGRE1 83>. Par conséquent, il était encore trop tôt pour disposer du futur système de gestion de bases de données généralisées. Par contre, nous avons trouvé un interlocuteur, en l'occurence la société SYSECA, fortement motivé par l'évaluation de l'apport d'une machine base de données à leur propre SGBD. Rappelons que SYSECA assure la commercialisation et le développement du SGBD SOCRATE/CLIO initialement conçu à l'Université Scientifique et Médicale de Grenoble.

Par ailleurs, nous avons disposé de la machine base de données d'INTEL, l'iDBP (iNTEL DATA BASE PROCESSOR).

L'expérimentation se fera en comparant les temps d'exécution de requêtes, judicieusement définies, exécutées une première fois sur une configuration traditionnelle de SGBD et une seconde fois, sur une architecture comportant une machine base de données, selon les schémas suivants.




CONFIGURATION CLASSIQUE




NOUVELLE CONFIGURATION

3.1.1.4 PRESENTATION MATERIELLE

La "configuration classique" est bâtie autour d'un VAX 750, sous VMS, comportant 2 Mo de mémoire centrale. La base de test est stockée sur un disque RA80 de 120 Mo et sera transférée, en fin d'expérimentation, sur un RA81 de 450 Mo. Nous disposons de la version V1.7 de SOCRATE/ CLIO. Les temps moyens d'accès sur le RA80 sont de 33 ms et sur le RA81, de 38 ms.

La seconde configuration dispose du même VAX 750. L'iDBP a 1 Mo de mémoire centrale et un disque winchester de 35 Mo, dont les temps moyens d'accès sont de l'ordre de 45 ms, inclu dans le chassis principal. L'iDBP intègre un microprocesseur "8086" dont la puissance de traitement est inférieure à celle du "68000". Le SGBD de l'iDBP est exécuté sous iRMX 86. La liaison VAX 750 - iDBP est une liaison de type asynchrone à 9600 bauds.

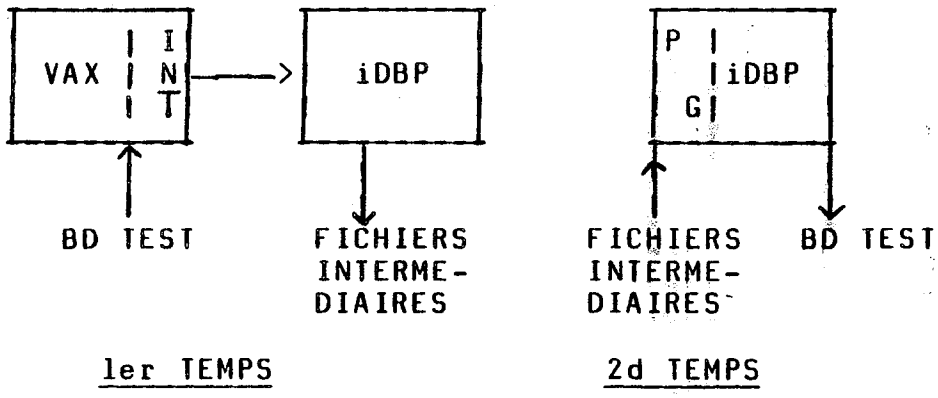
### 3.1.1.5 PRESENTATION LOGICIELLE

L'écriture de l'interface VAX 750 - iDBP a été à notre charge. La base de test est une simulation d'une gestion scolaire. Cette base est une véritable base de test qui a été générée selon des contraintes précises et définies au préalable, en vue d'un comparatif de différents SGBD. Sa structure en SOCRATE/CLIO, et partiellement en iDBP, est présentée en annexe.

L'ensemble des requêtes exécutées ont été spécialement écrites pour le test.

Pour le transfert de la base de test sur l'iDBP, nous avons dû procéder de la façon suivante. Nous avons commencé par traduire tous les concepts du LDD de SOCRATE/CLIO en termes du LDD de l'iDBP. De là, nous avons traduit la structure complète de la base ainsi que la totalité des requêtes sous le format des primitives du LMD de l'iDBP.

Ensuite, l'ensemble de la base a été transféré sur l'iDBP dans des "fichiers intermédiaires". Ceux-ci ont été traités, dans une seconde étape, et cette fois-ci de manière locale à l'iDBP et par des programmes écrits par nos soins, pour générer les fichiers définitifs et les chainages physiques entre enregistrements (simulation des anneaux SOCRATE, des inverses SOCRATE, des entités imbriquées). Ce transfert en deux temps a été nécessité par la nature des pointeurs que l'on peut définir entre n-uplets et qui seront décrits par la suite.




La durée du transfert de la base a été fortement pénalisée par la vitesse des échanges et par certaines limitations du SGBD de l'iDBP qui sortent du cadre de cette thèse pour être davantage approfondies.

Nous donnons davantage de précisions sur les conditions de réalisation du test au cours des § 3.3.1 et § 3.3.2.

Les particularités du LDD et du LMD de l'iDBP nous ont poussés à les décrire au cours du § 3.1.2. Nous pourrons alors aborder la phase de traduction des concepts du LDD de SOCRATE/CLIO en termes du modèle de données de l'iDBP.

## 3.1.2 DESCRIPTION DU SYSTEME DE GESTION DE BASES DE DONNEES DE L'iDBP

### 3.1.2.1 LANGAGE DE DEFINITION DE DONNEES

Le modèle de données de l'iDBP est le modèle relationnel qui a été étendu afin de pouvoir simuler les modèles hiérarchiques et réseaux. On parlera de fichier pour les relations et d'attribut pour les constituants.

#### DEFINITION D'UN FICHIER

La définition d'un fichier se fera avec la commande "DEFINE FILE" où l'on stipulera :

- le nom du fichier
- la taille des pages
- la localisation du fichier (nom logique du volume)
- le nombre minimum de pages réservées
- le nombre maximum de pages réservées (il peut être mis à indéfini si on ne le connait pas).

```
DEFINE FILE <nom fichier> <taille page> <volume>
            <nb-page-mini> <nb-page-maxi>...
```

Exemple:

```
DEFINE FILE ELEVE SMALLPAGE DBPSYS 4500 5500 ;
```

Un fichier pourra résider sur plusieurs volumes mais on ne pourra contrôler, au moment du stockage, la localisation des données.

La définition structurelle d'un fichier se fera avec la commande DEFINE SCHEMA. On spécifie en premier lieu si le fichier est structuré ou non. Un fichier structuré est assimilable à une relation et un fichier non structuré peut être, dans un premier temps, du texte et dans l'avenir du son ou des images. On pourra également contrôler l'utilisation ou la non récupération des espaces libérés par les enregistrements détruits. On definit la structure d'un fichier structuré en décrivant chacun de ses attributs. En entête on donnera la longueur de tous les attributs déclarés comme attributs de longueur variable et la taille de la zone d'expansion qui permettra dans le futur de définir des attributs supplémenataires. Cette zone d'expansion sera réservée pour chaque occurence. La description d'un attribut est composée de quatre éléments:

- son nom
- un qualificatif spécifiant s'il est de longueur fixe
- sa longueur.

Les types disponibles sont les suivants :

- entier (1,2 ou 4 octets) (signé ou non signé)
- chaîne de caractères
- pointeur (4octets)
- pointeur-id (4octets)

- pointeur de chaîne (6 octets).

Un pointeur référencie un enregistrement d'un fichier structuré. Cet attribut pourra pointer soit un enregistrement du fichier où est décrit ce pointeur soit un enregistrement d'un autre fichier. Lorsque ce pointeur est valué, il est composé d'un identificateur de fichier et d'un déplacement dans ce fichier.

La valeur d'un attribut de type pointeur-id d'un enregistrement E contient le déplacement par rapport à l'origine du fichier de cet enregistrement E. Ces pointeurs aideront à la définition et à la manipulation de listes chainées. Ainsi le modèle de données de l'iDBP se démarque du modèle relationnel dans la mesure où il permet de privilégier certains chemins d'accès.

Un pointeur de chaîne référencie une sous-chaîne dans un fichier non structuré. Sur quatre octets nous avons la position de départ de la sous chaîne et sur deux octets, la longueur de la sous chaîne.

Remarque: lorsqu'un attribut est stocké, sa valeur est précédée d'un octet de longueur qui spécifie la longueur effective de l'attribut valorisé. Chaque attribut de longueur fixe occupera donc la longueur déclarée dans le schéma plus un octet. Les attributs de longueur variable seront stockés dans la zone des attributs de longueur variable et précédés également d'un octet de longueur. Par contre on cumulera en fin de cette zone tous les octets non occupés par ces attributs. Ainsi, la façon dont est abordée la notion d'enregistrement de longueur variable ne permet pas de gagner de la place et d'un autre côté, l'accès à de tels attributs sera plus lent que l'accès à un attribut de longueur fixe.

DEFINE SCHEMA <nom fichier> <type> <re-util> <shema>

## FICHIER TEMPORAIRES, FICHIERS DEFINITIFS

A l'issue de la commande DEFINE SCHEMA ...., un fichier est considéré comme temporaire, c'est à dire qu'à la fin de la session il sera détruit. Afin de le conserver de façon définitive, on dispose d'une commande KEEPFILE qui rend permanent un fichier. Si auparavant on a pris soin de définir une base (commande DEFINE DATABASE <nom base>), on peut associer le fichier mentionné dans le KEEPFILE à une base précise ou au contraire le laisser totalement indépendant.

## CONTRAINTES D'INTEGRITES

Les types offerts par le LDD de l'iDBP sont relativement pauvres. Cet état de fait sera partiellement compensé par la possibilité de définir des contraintes d'intégrité sur un attribut d'un fichier. Celles-ci sont définies séparément de la structure de chaque fichier. On pourra définir plusieurs contraintes sur un attribut. Une contrainte d'intégrité (CI) attachée à un attribut d'un fichier peut comporter dans son expression une référence à un autre attribut du même fichier. Une contrainte d'intégrité peut être supprimée et/ou créée une fois que le fichier est déjà peuplé. Cette nouvelle CI ne sera pas testée pour les données déjà stockées.

Une CI est vérifiée lors du stockage d'une occurence. On pourra exiger lors de l'insertion ou de la modification d'une occurence de tester ou non les CI.

Dans une CI on pourra spécifier plusieurs exigences reliées par un OU et/ou un ET. La liste des contraintes est :

- IS VALUED
- IS UNIQUE

- IS ALPHA - IS NUMERIC
- IS NULL

- <attribut> EQ (<valeur> ou <attribut>)
- <attribut> NE (<valeur> ou <attribut>)
- <attribut> LT (<valeur> ou <attribut>)
- <attribut> GE (<valeur> ou <attribut>)
- <attribut> LE (<valeur> ou <attribut>)
- <attribut> GT (<valeur> ou <attribut>)
- <attribut> IS LISTED IN <valeur>.....
- <attribut> IS NOT LISTED IN <valeur>....
- <attribut> SPANS <valeur min> <valeur max>
- <attribut> DOES NOT SPAN <valeur min> <valeur max>

Ces contraintes nous permettront de simuler les types listes de valeurs et intervalles bornés tels qu'on les rencontre dans SOCRATE/CLIO.

Exemple:

IC PROFESSEURS .SALAIRE WHERE-IC SPANS 0 99999 OR-IC NOT-VALUED

DEFINITION D'INDEX

Il est possible de définir des index sur des attributs de fichiers structurés. On est limité à 16 index par fichier. Lorsqu'une sélection porte sur un attribut pour lequel a été défini un index, il sera toujours possible de demander l'exécution d'un tel opérateur sans avoir recours à l'utilisation du dictionnaire associé. Ces index sont bâtis sur la notion de B-arbre. Par opposition à un système relationnel tel que "SYSTEM R", on ne peut construire un index multi-attributs.

Il est possible de supprimer un fichier dés que celui-ci n'est plus utilisé. Tous les programmes, vues et CI définies

sur ce fichier seront détruits du même coup. De même, il est possible de supprimer une base. Ceci impliquera la suppression de tous les fichiers, programmes et vues associées à cette base.

Nous avons vu précédemment qu'il était possible de supprimer les CI, il en est de même pour les index.

### 3.1.2.2 LANGAGE DE MANIPULATION DES DONNEES

#### RESERVATION DES DROITS SUR UN FICHIER

Avant de manipuler n'importe quel fichier, il faut définir les droits de l'utilisateur sur le fichier pour la transaction ou la session en cours (commande ATTACH). On passe en paramètre les droits que l'on s'accorde et ceux que l'on réserve aux autres utilisateurs.

Il existe un droit nommé ADMIN, pour ADMINISTRATEUR, qui est réservé à l'administrateur de la base et qui préserve le fichier de toute autre manipulation. Ainsi pour définir une CI, un index ou détruire un fichier, l'admistrateur doit s'accorder ce droit.

#### INTRODUCTION DE LA NOTION DE CURSEUR

Le SGBD de l'iDBP permet de définir jusqu'à 32 pointeurs, appelés curseurs. Toutes les opérations de création, mise à jour et suppression utilisent ces curseurs. Dés qu'un curseur est positionné sur un enregistrement on peut manipuler cette occurence.

Il existe deux types de curseur :

- les curseurs séquentiels que l'on déplace en avant, en arrière ou que l'on peut positionner au ième enregistrement, on pourra limiter leur sens de déplacement,

- les curseurs aléatoires que l'on positionne à l'aide d'un attribut de type pointeur ou pointeur-id. On effectue alors des accès directs.

IDENTIFICATION D'UN CURSEUR

Nous avons mentionné précédemment que la plupart des opérateurs du LMD avait recours à des pointeurs. Avant leur utilisation, il est indispensable de les déclarer en nommant le nom des fichiers auxquels ils seront associés ainsi que leur mode de positionnement (séquentiel ou aléatoire). Pour cela on dispose de la commande:

START CURSOR <curseur> <nom fichier> <mode>

Le nombre maximum de définition de curseurs est de 32 pour chaque session.

POSITIONNEMENT D'UN CURSEUR

Un curseur séquentiel se positionne uniquement à l'aide de la commande FIND. Au départ, il faut le positionner à une valeur absolue (sur le ième enregistrement). De là, on peut le déplacer (toujours avec FIND) de x enregistrements en avant ou arrière. Il existe une option du FIND qui permet de le positionner sur l'enregistrement que le curseur pointait à l'avant dernier FIND.

FIND <seq-curseur> <mode-posi> (<position>)

Un curseur aléatoire dispose de deux commandes pour se positionner sur un enregistrement. La commande SET CURSOR positionne un curseur soit à l'aide d'un autre curseur, défini sur le même fichier, soit à partir d'un attribut de type pointeur ou pointeur-id. Lorsqu'un curseur repère un enregistrement, il peut être repositionné à partir d'un

attribut de l'enregistrement qu'il pointait. Ainsi on peut se déplacer aisément dans une liste chainée. Il faut savoir qu'un attribut est repérable à l'aide du curseur qui pointe l'enregistrement et du nom ou de la position de l'attribut dans l'enregistrement.

La commande STORE qui permet l'insertion d'occurences requiert un curseur aléatoire. A l'issue de cette commande le curseur est positionné sur l'enregistrement qui vient d'être créé. Si le curseur est positionné avant le STORE, cette position n'affecte en rien la localisation du nouvel enregistrement.

## LIBERATION D'UN CURSEUR

Si on se trouve en fin de session ou si le nombre de curseur est insuffisant, on peut libérer un curseur et le redéfinir sur un autre fichier. Pour la libération on dispose de la commande :

END CURSOR <curseur>

## OPERATION MONO FICHIER

### AJOUT D'ENREGISTREMENT

L'ajout d'un enregistrement se fait à l'aide de la commande STORE qui nécessite l'utilisation d'un curseur aléatoire. Lors de l'appel de cette commande on peut demander au sysème de vérifier les CI définies sur les attributs du fichier. Si une des contraintes n'est pas vérifiée, la création n'est pas réalisée. On donne la liste des valeurs des attributs dans l'ordre où ces derniers ont été déclarés dans la commande DEFINE SCHEMA. Si un attribut n'est pas valué, on lui attribue la valeur NULL qui correspond à l'indéfini. La valeur d'un attribut pourra être une cons-

tante ou une valeur d'un attribut d'un enregistrement existant.

STORE <curseur> <valid.> <littéral/ref attribut>...

MODIFICATION D'UN ENREGISTREMENT

La commande MODIFY nécessite un curseur qui soit positionné. Elle autorise la modification de plusieurs caractéristiques du même enregistrement en vérifiant ou non les CI associées à ces attributs modifiés. On peut également donner le nombre d'enregistrement, à partir de celui pointé par le curseur, pour lesquels on veut modifier les attributs mentionnés dans la commande de modification. Cette option est intéressante lorsque l'on veut mettre à blanc un attribut de tous les enregistrements d'un fichier. Pour réaliser une opération de modification il faut être en lecture écriture sur le fichier.

MODIFY <curseur> <vali.> <compteur> <attribut>
<littéral/ref-attribut>

SUPPRESSION D'UN ENREGISTREMENT .

La commande DELETE nécessite un curseur qui soit positionné. Elle permet de supprimer à partir de l'enregistrement pointé par le curseur un ou plusieurs enregistrements, ce nombre étant passé en paramètre. A l'issue de cette opération, le curseur est positionné sur le dernier enregistrement supprimé, mais il ne sera pas exploitable.

DELETE <curseur> <compteur>

AFFICHAGE D'UN ENREGISTREMENT

Les commandes FIND et SET CURSOR ne font que positionner un curseur et ne restituent pas l'enregistrement à l'utilisateur. Pour récupérer les attributs d'une occurence, on dispose de la commande FETCH. Elle nécessite un curseur qui soit positionné et permet de récupérer à partir de l'enregistrement pointé par le curseur une ou plusieurs occurences, le nombre étant passé en paramètre. A l'issue de cette opération, le curseur est positionné sur le dernier enregistrement restitué.

FETCH <curseur> <compteur>

OPERATION MULTI-FICHIERS

Nous venons de décrire l'ensemble des opérations que l'on peut effectuer sur un fichier. Les utilisateurs ont besoin de manipuler des sous-ensembles d'occurences de fichier, des intersections de fichiers, des sous-ensembles d'attributs à l'intérieur de toutes ou d'une partie des occurences d'un fichier ou encore de naviguer entre plusieurs fichiers. Le dernier point a été abordé dans la partie positionnement de curseurs. Les autres besoins pourront être satisfaits par la mise en oeuvre des opérateurs de l'algèbre relationnelle. L'approche réalisée sur l'iDBP est quelque peu différente de celle effectuée dans les langages classiques de manipulation de données.

On parlera de vues comme le résultat d'une transformation d'autres fichiers. Ces vues auront un schéma qui sera déterminé par l'ensemble des opérations que l'on effectuera sur les fichiers référencés au cours de la definition de celle-ci Les opérateurs que nous allons définir dans les paragraphes suivants doivent être davantage considérées en tant qu'outils de définition des différents schémas concep-

tuels plutôt qu'en tant que véritables opérateurs de l'algèbre relationnelle. Nous aurons l'occasion de prendre conscience des effets de cette remarque lors du comparatif réalisé pour cette étude.

## LA JOINTURE: JOIN

Cet opérateur concatène les occurences, de deux vues, pour lesquelles un attribut dans un enregistrement est égal à un autre attribut dans l'autre enregistrement. Les occurences résultats sont composées de tous les attributs des deux occurences concaténées. L'opérateur de comparaison entre les deux attributs sur lequel repose cette jointure est réduit à l'égalité. Ceci implique une plus forte complexité au niveau de certaines requêtes.

La vue, résultat d'un "JOIN", ne pourra être mise à jour. Par contre on pourra toujours modifier les fichiers sources sur lesquels sont bâtis ces vues.

DEFINE VIEW <nom> JOIN <fichier1> <attribut1>
<fichier2> <attribut2>

## LA SELECTION: SELECT

L'opération de sélection définit un sous-ensemble d'occurences, d'un fichier structuré, qui satisfont le même critère de sélection spécifié dans la description de la commande "SELECT". La structure de la vue résultat est identique à celle de la vue source. Dans la définition de la commande on peut spécifier ou non l'utilisation des index éventuellement définis sur les attributs participant au filtre.

DEFINE VIEW <nom> SELECT <fichier> WHERE-ITEM <op comp>
<USE-HELP/NO-HELP> <attribut> <valeur/attribut>

## LA CONNECTION DE FICHIERS : CONNECT

Cette transformation porte sur un fichier structuré et un fichier non structuré. Le fichier structuré doit comporter un attribut de type pointeur de chaîne où les valeurs pointent des occurences du fichier non structuré.

Les instances de la vue résultat seront composées des attributs du fichier structuré et d'un attribut supplémentaire qui sera la chaîne de caractères du fichier non structuré. Cette dernière sera référencée à l'aide d'un attribut du fichier structuré.

Lorsque l'on ajoute une occurrence à la vue résultat, le système crée automatiquement une occurence dans le fichier structuré et une dans le fichier non structuré et met à jour le pointeur référençant la chaîne de caractères.

On peut modifier n'importe quel attribut excepté le pointeur de chaîne du fichier structuré. La suppression d'un enregistrement supprime l'occurence uniquement dans le fichier structuré.

DEFINE VIEW <nom> CONNECT <fichier struct.>
<attri. pteur.> <fichier non struct.>

## L'EXTRACTION DE CHAINE : SUBSTRING

Cette transformation sur des fichiers non structurés correspond à l'opérateur de sélection sur des fichiers structurés. Elle permet de récupérer toutes les chaînes de caractères d'une certaine longueur ou toutes les chaînes comprises entre deux mots clés. Si on considère un fichier non structuré contenant des documents et que chaque document commence par le mot clé DEBDOC et se termine par le mot clé FINDOC, on peut demander d'extraire toutes les

chaînes comprises entre DEBDOC et FINDOC et l'on obtient isolément tous les documents du fichier.

## LA PROJECTION : PROJECT

Il arrive très fréquemment que l'on cherche à obtenir un sous-ensemble d'attributs d'occurences sans pour autant vouloir afficher ces occurences en totalité. Pour cela on dispose de la transformation PROJECT. Selon la valeur du paramètre d'inclusion, elle conservera ou excluera tous les attributs passés en paramètre. Lorsque l'on supprime ou modifie un enregistrement d'une telle vue, l'enregistrement est supprimé ou modifié dans la vue source. On ne pourra modifier que les attributs contenus dans la projection. Un enregistrement créé dans une vue résultat d'un PROJECT, verra les attributs non projetés prendre la valeur indéfinie (NULL).

DEFINE SCHEMA <nom> PROJECT <INCLUDE/EXCLUDE>

<fichier source> <attribut.....>

## COMBINAISONS DE VUES

On peut construire des vues à partir d'autres vues avec les restrictions suivantes:

| Vues qui sont le résultat de | Type de vue source |
|---|---|
| CONNECT | PROJECT, fichier |
| SUBSTRING | PROJECT, fichier |

| | |
|---|---|
| JOIN | CONNECT, PROJECT, fichier |
| SELECT | JOIN,CONNECT,PROJECT,fichier |
| ORDER | SELECT,JOIN,SUBSTRING,fichier |
| PROJECT | Tous |

Remarque: tous ces opérateurs que l'on peut appliquer sur un ou plusieurs fichiers ou vues et qui donnent en résultat une autre vue, vont nous permettre de définir le niveau externe de notre base de données. Ainsi c'est l'administrateur de la base qui définira les vues de chacun des utilisateurs selon leurs besoins (fichiers, caractéristiques, droits...). Les droits des utilisateurs sur ces vues sont fonction des droits définis sur les fichiers sources sur lesquels sont construites ces vues. Le niveau conceptuel a été défini par l'administrateur de la base lors de la déclaration de structure de chacun des fichiers de la base. Sous le niveau conceptuel, nous retrouvons le niveau interne, orienté davantage vers l'aspect performance sur lequel on pourra agir soit en créant des index soit en jouant sur la localisation physique des données.

### 3.1.2.3 SECURITE DES DONNEES

La première solution, et de loin la plus simple, proposée pour le retour à une base cohérente, à l'issue d'un incident matériel ou logiciel, est la sauvegarde physique de la base. Pour cela on génère un fichier de "BACKUP" dans lequel on videra tous les fichiers, tous les dictionnaires associés, les descriptions des vues, etc... Ensuite, à l'issue de l'incident on disposera de la commande "STORE" afin de restaurer la base dans son intégralité.

Il existe une seconde solution qui permet de restaurer la base dans un état cohérent d'une façon beaucoup plus dynamique. Pour cela il faut introduire la notion de transaction. Une transaction est une série de commandes indissociables. on suppose que la base est dans un état cohérent au début et à la fin de chaque transaction. Durant le déroulement d'une transaction, l'état de la base ne peut être considéré comme cohérent. Cela signifie que si un incident intervient durant l'exécution de cette transaction, il faudra être en mesure de revenir à l'état précédant le début de la transaction.

On dispose de la commande "BEGIN FRAME" qui permet de définir le début de la transaction et de "END FRAME" qui en marquera la fin.

## 3.2 TRADUCTION DU MODELE DE DONNEES DE SOCRATE/CLIO

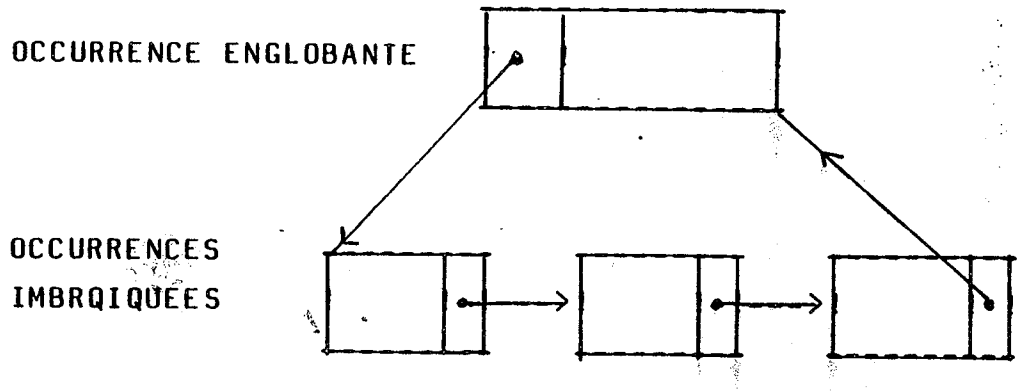
## DANS LE MODELE DE DONNEES DE L'iDBP

Cette partie est consacrée à la traduction des éléments du langage de définition de données de SOCRATE/CLIO en termes du LDD de l'iDBP. Ceci est réalisé en vue de décrire la structure de la base de test, disponible à l'origine sous SOCRATE/CLIO, sur la machine bases de données iDBP.

### 3.2.1 TRADUCTION DES ENTITES "SOCRATE"

A chaque entité de niveau 1, dans SOCRATE/CLIO, nous pourrons associer un fichier iDBP.

Les entités imbriquées seront également représentées par des fichiers iDBP. Elles seront gérées comme des listes chainées. Par contre, le rapprochement physique entité englobante - entité imbriquées ne sera pas conservé comme dans SOCRATE/CLIO. Mais cet effet pourra être atténué dans le sens où l'on peut gérer la localisation physique des fichiers. Ainsi les deux fichiers (de l'entité englobante et de l'entité imbriquée) pourront être situés sur le même volume.



## 3.2.2 TRADUCTION DES ANNEAUX "SOCRATE"

Les anneaux et références sont représentés de la façon suivante:

- l'occurence tête d'anneau pointera sur le premier élément fils de la chaîne

- chacun des éléments de la chaîne pointera sur la tête d'anneau, sur le suivant dans la chaîne et éventuellement sur le précédent. Ce sera à nous de choisir le type de chainage en fonction du type déclaré dans la structure SOCRATE/CLIO (chainage simple ou chainage double).

La mise à jour des listes est entièrement à la charge de l'utilisateur qui ne dispose que de primitives de bas niveau.

Exemple: représentation d'un anneau (simple et double) avec les pointeurs iDBP:

```
*Structure SOCRATE*

        ENTITE 10 ELEVE
           DEBUT
                NOTE ANNEAU
                ADRESSE ANNEAU AVEC CHAINE DOUBLE
                /*ON CONSERVE TOUTES LES ADRESSES D'UN*/
                /*ELEVE*/
           FIN

        ENTITE 100 NOTE
           DEBUT
                NOTE-EL REFERE NOTE DE UN ELEVE
           FIN
```

```
ENTITE 100 ADRESSE
   DEBUT
        AD-EL REFERE ADRESSE DE UN ELEVE
   FIN
```

En "iDBP" nous aurons explicitement:

- pour chaque élève:
  - un pointeur physique sur sa première NOTE : "PTR-FILS-NOTE"
  - un pointeur physique sur sa première ADRESSE : "PTR-FILS-ADRESSE"
- pour chaque NOTE
  - un pointeur physique sur le père : "PTR-PERE-ELEVE"
  - un pointeur physique sur la note suivante : "PTR-FRERE-ELEVE"
- pour chaque ADRESSE
  - un pointeur physique sur le père : "PTR-PERE-ELEVE"
  - un pointeur physique sur l'adresse suivante : "PTR-FRERE-ELEVE"

- un pointeur physique sur l'adresse précédente : "PTR-FREPRE-ELEVE"

Les noms des pointeurs sont déterminés de la façon suivante et ceci afin de lever toute ambigüité dans le cas où il y aurait plusieurs anneaux ou références dans une même entité:

PTR-PERE-X : X = nom de l'entité de la tête d'anneau
PTR-FILS-Y : Y = nom de l'entité composant la chaîne
PTR-FRERE-Z : Z = nom de l'entité tête d'anneau
PTR-FREPRE-T : T = nom de l'entité tête d'anneau

### 3.2.3 TRADUCTION DES INVERSES "SOCRATE"

Les inverses "SOCRATE" permettent de formaliser les associations de type N/M entre deux entités. Ce type d'association est formalisé, dans le modèle relationnel, par la création d'une nouvelle relation.

Nous discernerons deux formes d'inverse:

- 1) les inverses déclarées au plus haut niveau

- 2) les inverses définis au sein des entités.

1) Un inverse de plus haut niveau est formalisé par une seule chaîne de bits. Par conséquent une occurence ne peut appartenir qu'à une seule valorisation d'un inverse de plus haut niveau. La traduction optimale de cette forme d'inverse se retrouve dans l'établissement d'un chainage entre tous les enregistrements appartenant à cet inverse. Lorsque l'on voudra accéder tous les éléments appartenant à cet inverse, il sera nécessaire de parcourir cette liste chainée à partir d'un élément père. Nous sommes alors tenu de déclarer pour chacun de ces inverses un père fictif qui

sera l'unique occurence d'un fichier iDBP qui sera défini pour chaque inverse de plus haut niveau.

2)
Soit la structure "SOCRATE"

```
ENTITE CLUB_SPORTIF (100)
        DEBUT
                NOM_CLUB MOT(15) CLE UNIQUE
                MEMBRE INVERSE TOUTE PERSONNE
                ...
        FIN

ENTITE PERSONNE (1000)
        DEBUT
                NOM_PERS MOT (30) CLE UNIQUE
                ...
        FIN
```

Avec une telle structure, la même PERSONNE X peut être membre de n (n>=0) CLUB_SPORTIF. La solution des listes chainées ne peut être retenue qu'en définissant, dans notre exemple, 100 caractéristiques de type "pointeur" dans la relation "iDBP" PERSONNE. Chacune de ces personnes peut être membre au plus de 100 CLUB_SPORTIF. Par conséquent, pour la traduction en relationnel, nous sommes tenus d'avoir recours à 3 relations:

```
CLUB_SPORTIF (NOM_CLUB,...)
PERSONNE     (NOM_PERS,...)
APPARTENANCE (NOM_CLUB,NOM_PERS)
```

Avec les possibilités offertes par l'iDBP, nous pouvons substituer à NOM_PERS de la relation APPARTENANCE, l'identificateur de type "pointeur-id" qui est défini implicitement pour chaque occurence de toute relation. Ceci permettra de réaliser un accès direct aux occurrences de PERSONNE.

### 3.2.4 TRADUCTION DES TYPES

Type numérique borné

Une caractéristique de type numérique borné comporte, en SOCRATE/CLIO, une borne inférieure et supérieure. Le LDD de l'iDBP n'offre pas de telles possibilités, par contre, il donne à l'utilisateur la possibilité de générer des contraintes d'intégrité. Dans ce cas précis, on pourra créer une contrainte sur la plage de valeurs dans laquelle pourra se situer la caractéristique numérique considérée. Il reste à nous assurer que cette contrainte d'intégrité sera exécutée uniquement dans la mesure où la caractéristique est valuée. On aura une contrainte par caractéristique de type numérique borné. Ces C.I. vont nous permettre de définir des caractéristiques avec des types plus riches que ceux offerts par le LDD de SOCRATE/CLIO. Ainsi on pourra définir des caractéristiques de type numérique qui pourront:

- prendre toutes les valeurs possibles, exceptées celles comprises dans un intervalle

- être supérieures ou inférieures à des constantes ou à d'autres caractéristiques

- être obligatoirement valuées ou non valuées.

Exemple:

STRUCTURE SOCRATE

ENTITE PROFESSEUR (100)

DEBUT

SALAIRE DECIMAL DE 0 A 99999

FIN

Nous aurons en iDBP:

IC PROFESSEURS WHERE-IC SPANS 0 99999 OR-IC NOT-VALUED

Caractéristique de type liste de valeurs:

L'iDBP ne permet pas de gérer les listes de valeurs au sens où elles sont manipulées dans SOCRATE/CLIO. Trois solutions s'offrent à nous:

- on traite ces caractéristiques comme les autres caractéristiques et on ne tient pas compte de la notion de liste de valeurs.

- on tente de se rapprocher de la solution retenue dans SOCRATE/CLIO et on décrira ce type par un pointeur vers une occurence (qui sera une valeur de la liste) d'un fichier structuré.

- à chaque caractéristique de type liste de valeurs, on associe un attribut de type caractère que l'on complète par une contrainte d'intégrité dans laquelle on précisera toutes les valeurs contenues dans la liste définie dans la structure de la base sous SOCRATE/CLIO.

Pour gagner en temps de traitement, on serait tenté de retenir la première solution mais la sémantique du type ne s'y retrouverait pas.

Si l'on cherche à se rapprocher de la solution retenue dans SOCRATE/CLIO, il nous faudrait faire une synthèse des deux dernières solutions. La caractéristique de type liste de va-

leurs serait représentée, sous iDBP, par un pointeur qui nous permettrait d'accéder une occurence à deux attributs dont l'un se verrait limité par une C.I. qui spécifierait l'ensemble des valeurs qu'elle serait en mesure de prendre.

Mais le but de cette traduction n'est pas de se rapprocher de la façon la plus fidèle d'une structure de données physiquement identique à celle de SOCRATE/CLIO mais tout simplement, correspondante sur le plan sémantique. Nous retiendrons donc la troisième solution.

Caractéristiques de longueur variable ou fixe:

Toutes les caractéristiques seront déclarées comme caractéristiques de longueur fixe. Leur chargement en sera facilité et il est reconnu que l'on accède plus rapidement à des champs de longueur fixe. En effet dès que l'on connait la position de début de l'occurence, on atteint très facilement de telles caractéristiques par déplacement. De plus la gestion des caractéristiques de longueur variable ne permet pas de gagner de la place tout au moins telle qu'elle est conçue dans l'iDBP. Les caractéristiques de longueur variable sont stockées dans une zone réservée de l'occurence et tous les octets non utilisés sont cumulés en fin d'occurence.

### 3.2.5 TRADUCTION DES BLOCS

SOCRATE/CLIO permet de déclarer la notion de bloc qui simplifie notablement la manipulation des noms de caractéristique. Le format iDBP n'offre pas de telles facilités mais il sera possible de pallier cet inconvénient par des noms de champ plus complexes donc plus siginificatifs.

## 3.3 RESULTATS

### 3.3.1 PRESENTATION DU CONTEXTE DU TEST

#### 3.3.1.1 REMARQUES GENERALES

A l'origine du test, nous avions l'intention de réaliser notre étude sur les requêtes utilisées par les tests d'évaluations des différentes versions de SOCRATE/CLIO. Nous avons très vite pris conscience que ces requêtes n'étaient pas adaptées au travail que nous avions à réaliser. Ceci en raison du fait que chacune d'entre elles faisaient très souvent intervenir plusieurs opérations spécifiques au modèle de données de SOCRATE/CLIO, parcours d'anneaux combinés avec des parcours d'entités imbriquées. Ce fait est d'autant plus gênant que nous voulions bâtir notre étude sur une comparaison des temps d'exécution des opérateurs pris séparément.

Chacune des requêtes, écrites dans les langages respectifs des deux systèmes, ont été cataloguées dans un code directement interprétable. Les transactions iDBP sont lancées, depuis le VAX 750, par l'intermédiaire de l'interface spécialement écrit pour l'expérimentation.

Il n'était pas possible de relever le temps CPU consommé par les programmes sur la MBD, par conséquent nous ne retiendrons que le temps apparent relevé sur le VAX et qui tient compte d'une vitesse d'échange de 9600 bauds. Pour les requêtes exécutées sur le VAX, nous disposons à la fois du temps CPU et du temps apparent ainsi que du nombre d'appels au module de pagination VSS et du nombre de pages demandées en lecture et écriture. Ces derniers renseignements bien que non disponibles sur l'iDBP, seront conservés pour la plupart des requêtes SOCRATE/CLIO.

Pour les requêtes demandant l'affichage de données, nous releverons le temps d'exécution des requêtes avec et sans affichage des résultats. Il existe en effet des options, à la fois sur l'iDBP et sur le VAX, qui permettent d'offrir la possibilité d'afficher ou non les données accédées.

De même, nous pourrons exécuter les requêtes avec ou sans utilisation de l'index défini sur les critères de recherche déclarés comme clé.

Les temps relevés sont en centième de seconde.

Lorsque l'on parle d'accès à un enregistrement, cela supposera l'impression d'une ou plusieurs caractéristiques. En effet dans un système tel que SOCRATE/CLIO, il ne suffit pas de positionner un Xi pour garantir que l'accès physique a été réalisé, il faut en plus faire appel à une des caractéristiques de l'occurence répérée par ce Xi et, par exemple, en demander l'impression.

### 3.3.1.2 PRECISIONS SUR L'ENVIRONNEMENT UTILISATEUR

Au cours de la première partie de l'expérimentation, les mesures ont été relevées sur un calculateur hôte supportant des charges de travail totalement étrangères à nos transactions et variant considérablement entre deux exécutions de la même requête. Les temps apparents relevés étaient alors composés à la fois des temps CPU et d'entrées / sorties de la requête courante, mais aussi des temps de partage des ressources avec les autres utilisateurs. Il nous était donc impossible d'analyser, de manière pertinente, les temps relevés pour des requêtes accédant à des masses d'informations variées ou de les comparer à ceux de l'iDBP.

Par conséquent, l'ensemble des requêtes ont été reprises, mais cette fois-ci dans un contexte mono-utilisateur à

la fois vis à vis de la base mais aussi de la machine (pas d'autres utilisateurs et pas de traitement batch). Nous présentons certaines mesures avec le label "multi-utilisateurs" afin de montrer l'impact, connu, de ces utilisateurs sur nos résultats vis à vis d'une solution telle que l'iDBP où l'on dispose d'un calculateur totalement dédié à nos besoins.

Lorsque l'on parlera de résultats multi-utilisateurs, cela signifiera:

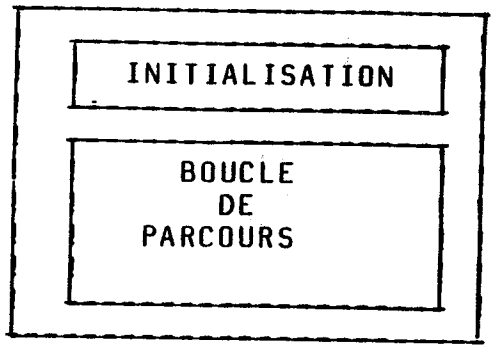
- une personne travaillant sur la base
- N personnes travaillant sur le calculateur hôte mais n'accédant pas à la base de test

et de résultats mono-utilisateurs:

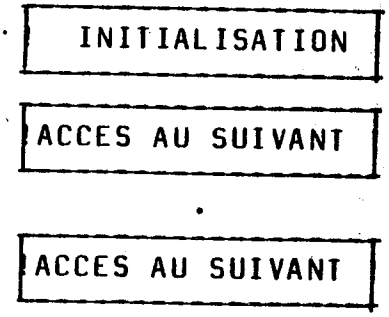
- une seule personne travaillant sur la base.

### 3.3.1.3 STRUCTURE DES REQUETES

La quasi totalité des requêtes spécifiées pour notre test nous restitue, en bloc, un ensemble d'instances ou de caractéristiques d'instances. En comparant leurs temps d'exécution sur les deux systèmes, nous avons été conduits à "éclater" ces requêtes afin de détailler les coûts d'accès à chacune des instances appartenant à l'ensemble résultat. Nous parlerons donc de "parcours global" et de "parcours élément par élément". Ces deux types de requêtes peuvent être schématisés de la façon suivante:




"PARCOURS GLOBAL"




"PARCOURS ELEMENT PAR ELEMENT"

Nous allons donner les textes des programmes d'une requête, REQEXEM, qui réalise une sélection de tous les "ELEVE" de "NOM" égal à "PETIT" (cf strcuture de la base en annexe).

"PARCOURS GLOBAL"

En SOCRATE/CLIO

```
:SUPEXP REQEXEM
:DEFPRO REQEXEM
D X1 = UN ELEVE
:EXP
POUR TOUT ELEVE X1 AVEC NON = "PETIT" ;
   I PRENOM DE X1
   I DATE_NAISSANCE DE X1
FIN
:FDEF?
```

En iDBP

```
MACRO REQEXEM ,
ATTACH ELEVE READ-WRITE ;
VIEW ELE, SELECT ELEVE WHERE-ITEM EQ USE-HELP .NOM
"PETIT" ;
```

```
   START £1 ELE BI ;
   LOOPWHILE VALUED £1 ,
      NEXT £1 ,
   ENDLOOP .
   FREE ELEVES ;
   ENDMACRO ;
```

"PARCOURS ELEMENT PAR ELEMENT"

En SOCRATE/CLIO

```
   M X1 = UN ELEVE AVEC NOM = "PETIT" ;
   I PRENOM DE X1
   I DATE_NAISSANCE DE X1

   M X1 = UN ELEVE AVEC NOM = "PETIT" ;
   I PRENOM DE X1
   I DATE_NAISSANCE DE X1

   M X1 = UN ELEVE AVEC NOM = "PETIT" ;
   I PRENOM DE X1
   I DATE_NAISSANCE DE X1
```

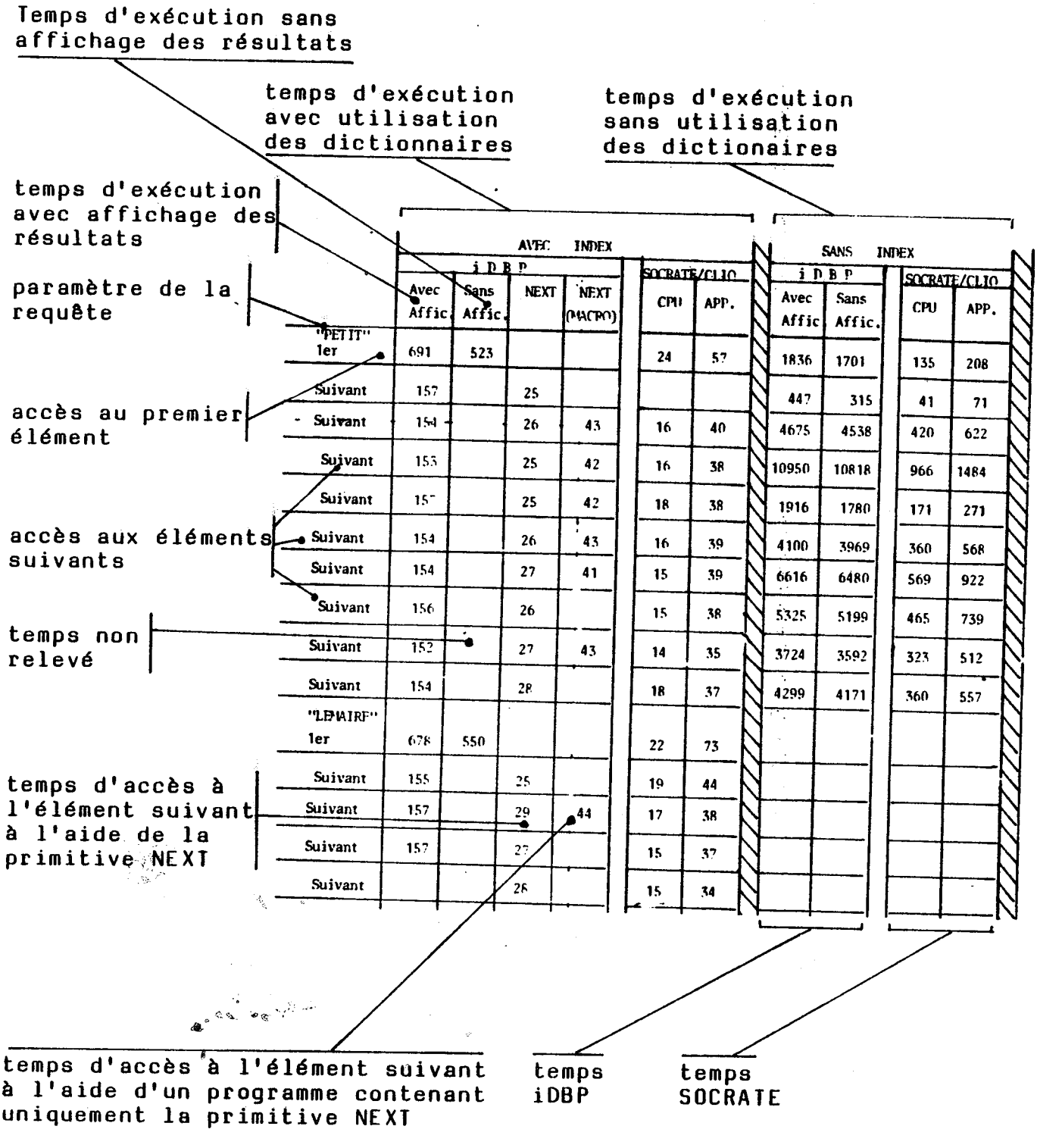
En iDBP

```
   ATTACH ELEVES READ-WRITE ;
   VIEW ELE SELECT ELEVE WHERE-ITEM EQ USE-HELP .NOM
   "PETIT" ;
   START £1 ELE BI ;

   NEXT £1 ;

   NEXT £1 ;

   NEXT £1 ;
```

D'autres requêtes sont fournies en annexe.

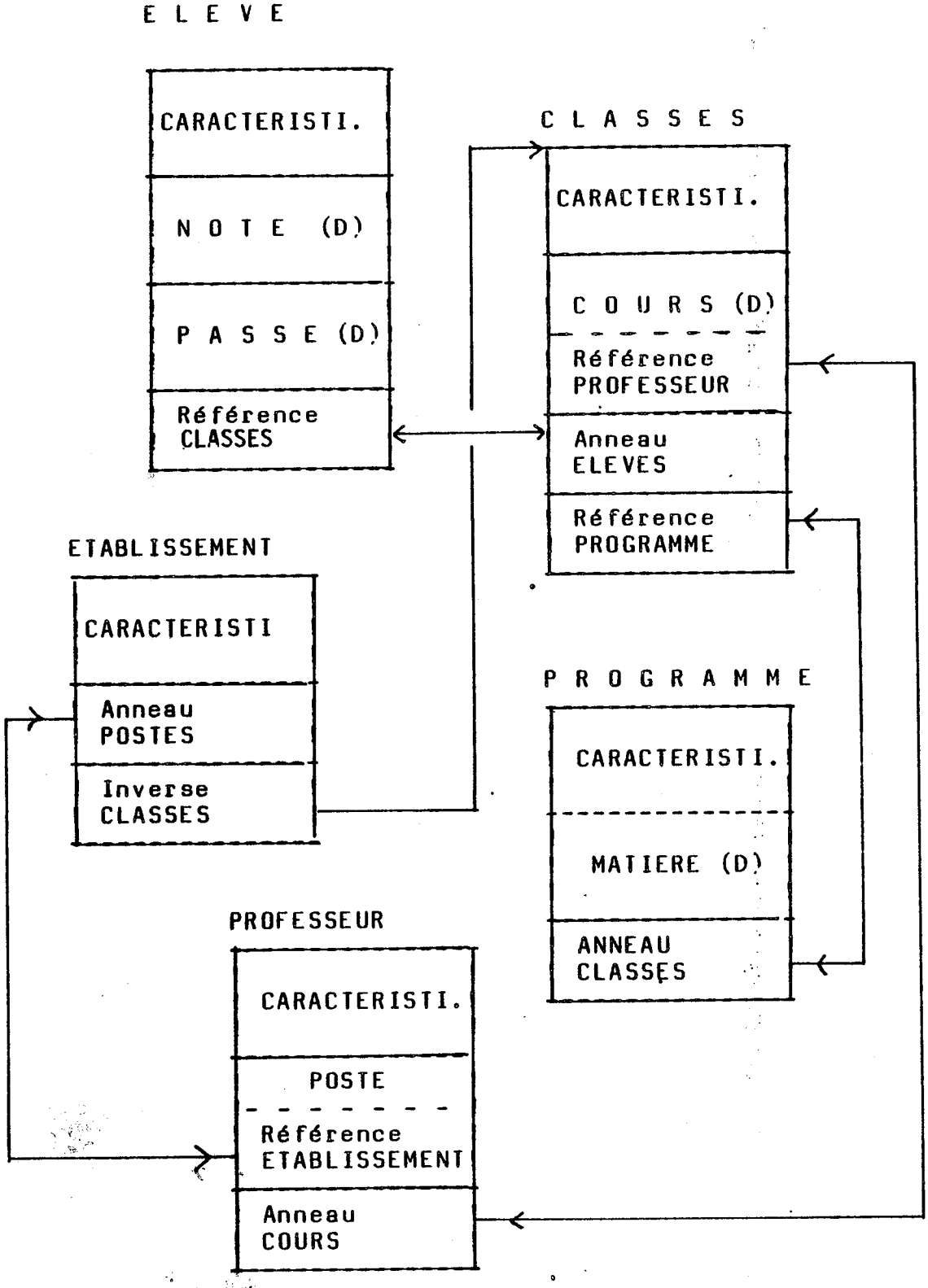
### 3.3.1.4 COMMENT LIRE LES TABLEAUX DE RESULTATS ?

Afin d'aider le lecteur pour une meilleure compréhension des tableaux de résultats, nous allons détailler le contenu du premier tableau: l'exécution de REQUETE 1.




| | AVEC INDEX | | | | | | SANS INDEX | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | iDBP | | | | SOCRATE/CLIO | | iDBP | | SOCRATE/CLIO | |
| | Avec Affic | Sans Affic. | NEXT | NEXT (MACRO) | CPU | APP. | Avec Affic | Sans Affic. | CPU | APP. |
| "PETIT" 1er | 691 | 523 | | | 24 | 57 | 1836 | 1701 | 135 | 208 |
| Suivant | 157 | | 25 | | | | 447 | 315 | 41 | 71 |
| Suivant | 154 | | 26 | 43 | 16 | 40 | 4675 | 4538 | 420 | 622 |
| Suivant | 153 | | 25 | 42 | 16 | 38 | 10950 | 10818 | 966 | 1484 |
| Suivant | 157 | | 25 | 42 | 18 | 38 | 1916 | 1780 | 171 | 271 |
| Suivant | 154 | | 26 | 43 | 16 | 39 | 4100 | 3969 | 360 | 568 |
| Suivant | 154 | | 27 | 41 | 15 | 39 | 6616 | 6480 | 569 | 922 |
| Suivant | 156 | | 26 | | 15 | 38 | 5325 | 5199 | 465 | 739 |
| Suivant | 152 | | 27 | 43 | 14 | 35 | 3724 | 3592 | 323 | 512 |
| Suivant | 154 | | 28 | | 18 | 37 | 4299 | 4171 | 360 | 557 |
| "LEMAIRE" 1er | 678 | 550 | | | 22 | 73 | | | | |
| Suivant | 155 | | 25 | | 19 | 44 | | | | |
| Suivant | 157 | | 29 | 44 | 17 | 38 | | | | |
| Suivant | 157 | | 27 | | 15 | 37 | | | | |
| Suivant | | | 26 | | 15 | 34 | | | | |

### 3.3.1.5 SCHEMA SYNOPTIQUE DE LA BASE



Explications sur le schéma:

(D) : entité désimbriquée physiquement.

<——> : Anneau - Référence

——> : Inverse

| | |
|---|---|
| CARACTERISTI. | caractéristiques de l'entité |
| XXXXXX | entité(s) imbriquée(s) logiquement |
| Anneau<br>YYYYY | caractéristique de type anneau |
| Référence<br>ZZZZZ | caractéristique de type référence |
| Inverse<br>TTTTT | caractéristique de type inverse |

La structure de la base SOCRATE/CLIO est fournie en annexe.

La base est peuplée de la façon suivante:

- 18000 élèves
- 72000 notes
- 2700 passés
- 600 classes

- 10800 cours

- 12 établissements

- 15 programmes

- 1200 professeurs.

La taille des espaces est la suivante:

| | | |
|---|---|---|
| N°4 | FICH | 4154240 octets |
| N°5 | DICO | 305088 octets |
| N°6 | NOTE | 905120 octets |
| N°7 | CLAS | 22176 octets |
| N°8 | COUR | 203200 octets |
| N°9 | PASS | 925728 octets. |

Pour la plupart des mesures, nous avons relevé le nombre de pages accédées, en lecture et en écriture, sur les différents espaces physiques tels qu'ils sont définis dans SOCRATE/CLIO. Ces espaces seront référencés par leur numéro:

- espace n° 1 : espace contenant les programmes catalogués et la structure

- espace n° 4 : espace contenant toutes les données de la base qui n'ont pas été explicitement désimbriquées dans la définition de la structure

- espace n° 5 : espace contenant les dictionnaires

- espace n° 6 : espace "NOTE" contenant les NOTES des ELEVE

- espace n° 7 : espace "CLAS" contenant le PR_PRINC et le CHEF_CL de chaque CLASSES

- espace n° 8 : espace "COUR" contenant les COURS de chaque CLASSES

- espace n° 9 : espace "PASS" contenant le PASSE des ELEVE

### 3.3.1.6 MECANISMES DE LECTURE ET DE GESTION DE LA MEMOIRE CENTRALE

Cette partie sera exclusivement consacrée à la description des mécanismes de lecture et de gestion de la mémoire centrale par le système SOCRATE/CLIO. Nous n'avons pu obtenir les renseignements équivalents pour l'iDBP.

MECANISME DE LECTURE

Le système SOCRATE/CLIO utilise le concept de mémoire virtuelle. Celle-ci sera découpée en espace virtuels qui se projettent sur les espaces réels par blocs ou sous-pages. Les espaces réels et la mémoire centrale se projettent l'un dans l'autre par ensembles de n sous-pages, appelés pages.

Ainsi une demande de lecture se décompose de la façon suivante:

- conversion d'une adresse virtuelle en une adresse réelle

- recherche de l'information sur l'espace réel

- transfert de la page en mémoire.

MECANISME DE GESTION DE LA MEMOIRE CENTRALE

Chaque utilisateur dispose, au niveau de la mémoire centrale, d'un espace (espace des cadres), qui lui est totalement réservé. Cet espace est découpé en cadres de la taille d'une page physique. Sur la version VAX de SOCRATE/CLIO, nous disposions de 20 cadres de 2 Ko chacun. Ceux-ci contiennent notamment les pages physiques accédées sur la base.

Avant de transférer une page en mémoire centrale, on vérifie si elle s'y trouve déjà, à l'aide de la table des cadres. Celle-ci gère l'ensemble des pages situées en mémoire centrale. Si la page recherchée s'y trouve déjà, on n'a pas à la transférer sinon on l'accède et on la charge dans un cadre libre. Si aucun cadre n'est disponible, on recherche le cadre le "moins récemment utilisé" (algorithme dit de "LRU") et on le recopie sur disque s'il a été modifié. On peut alors écrire la page demandée.

MECANISME DE LECTURE ANTICIPEE

SOCRATE/CLIO, sur VAX, dispose d'un mécanisme d'anticipation de lecture. Lorsqu'il détecte un parcours séquentiel sur un espace physique, il anticipe les lectures futures en réalisant des transferts qui pourront atteindre une taille de 16 ko. Les avantages de ce mécanisme seront mis en évidence au cours de notre test.

## 3.3.2 RESULTATS ET ANALYSE DES DIFFERENTES EVALUATIONS

Le but de notre étude était de mettre évidence ce que pouvait apporter une architecture de SGBD bâtie autour d'une machine bases de données vis à vis d'une configuration classique. Afin de réaliser une analyse pertinente, il nous a paru opportun de réaliser nos mesures sur les différents opérateurs rencontrés dans les modèles de données de SOCRATE/CLIO et de l'iDBP à savoir:

- l'opération de sélection

- l'opération de parcours d'une entité imbriquée

- l'opération de parcours d'un anneau

- l'opération de jointure

- les opérations de mises à jour.

Nous nous attacherons donc à définir chacune de ces opérations, à livrer nos résultats et à donner notre analyse des chiffres obtenus.

## 3.3.3 OPERATION DE SELECTION

### 3.3.3.1 PRESENTATION

L'évaluation des opérations de sélection consistera à relever les temps d'exécution des requêtes dont le but est de récupérer une ou plusieurs occurrences d'une entité, ceci de façon séquentielle ou indéxée et sans avoir recours à la notion de curseur pour l'iDBP ni à celle d'anneau ou d'inverse pour SOCRATE/CLIO sur VAX.

L'analyse des résultats permettra de comparer les différentes méthodes d'indexation, les B-arbres pour l'iDBP et le hash-code pour SOCRATE/CLIO sur VAX <SOCREF> ainsi que les parcours séquentiels des deux systèmes.

Pour l'iDBP, l'opération de sélection se fait à l'aide de la primitive de définition d'une vue de type "SELECT". Au cours de cette définition de vue, on donne la ou les caractéristiques qui constituent le critère de recherche. Pour chacune de ces caractéristiques, on précise si on veut ou non profiter de l'index éventuellement défini sur la caractéristique participant au filtre. Certaines mesures seront effectivement réalisées sans index.

A partir de l'instant où la vue est définie, on la parcourt séquentiellement comme une relation classique et on récupère alors chacune des occurrences vérifiant le ou les critères de sélection. L'accès et l'affichage, pas à pas, de chacune de ces occurrences peut se faire à l'aide de la primitive "NEXT".

Pour SOCRATE/CLIO, dans le cas d'utilisation d'index, le système accède au "premier" enregistrement satisfaisant le critère de sélection à l'aide du dictionnaire. Ensuite,

chacun des enregistrements satisfaisant ce filtre est accédé par chainage, totalement transparent à l'utilisateur, existant entre tous les éléments qui ont la même valeur de hash-code appliquée sur la clé <SOCREF>. On pourra également préciser l'utilisation ou non du dictionnaire pour effectuer la recherche.

DEFINITION DES REQUETES DE SELECTION

REQUETE 1 : affichage du NOM, PRENOM et DATE_NAISSANCE d'un ELEVE de NOM X. X prendra la valeur "PETIT" puis la valeur "LEMAIRE". Cette question doit pouvoir être répétée pour trouver successivement tous les ELEVES de NOM X. La caractéristique NOM est une clé.

Q11 : on reprend REQUETE 1.

Q12 : on reprend REQUETE 1 et on ajoute un critère de sélection, non clé, dans le filtre.

Q13 : on reprend REQUETE 1 et on ajoute deux critères de sélection, non clé, dans le filtre.

Q14 : on reprend REQUETE 1 et on ajoute trois critères de sélection, non clé, dans le filtre.

Q15 : on reprend Q14. Cette requête est associée à Q16 et par conséquent ne sera pas exécutée sous SOCRATE/CLIO.

Q16 : on reprend Q14 et on crée un dictionnaire sur une des 3 caractéristiques non clé. Cette requête sera exécutée uniquement sur l'iDBP car il n'est pas possible de définir, sous SOCRATE/CLIO, un filtre utilisant plus d'un dictionnaire.

| | AVEC INDEX | | | | | | SANS INDEX | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | i D B P | | | | SOCRATE/CLIO | | i D B P | | SOCRATE/CLIO | |
| | Avec Affic | Sans Affic. | NEXT | NEXT (MACRO) | CPU | APP. | Avec Affic | Sans Affic. | CPU | APP. |
| "PETIT" 1er | 691 | 523 | | | 24 | 57 | 1836 | 1701 | 135 | 208 |
| Suivant | 157 | | 25 | | | | 447 | 315 | 41 | 71 |
| Suivant | 154 | | 26 | 43 | 16 | 40 | 4675 | 4538 | 420 | 622 |
| Suivant | 153 | | 25 | 42 | 16 | 38 | 10950 | 10818 | 966 | 1484 |
| Suivant | 157 | | 25 | 42 | 18 | 38 | 1916 | 1780 | 171 | 271 |
| Suivant | 154 | | 26 | 43 | 16 | 39 | 4100 | 3969 | 360 | 568 |
| Suivant | 154 | | 27 | 41 | 15 | 39 | 6616 | 6480 | 569 | 922 |
| Suivant | 156 | | 26 | | 15 | 38 | 5325 | 5199 | 465 | 739 |
| Suivant | 152 | | 27 | 43 | 14 | 35 | 3724 | 3592 | 323 | 512 |
| Suivant | 154 | | 28 | | 18 | 37 | 4299 | 4171 | 360 | 557 |
| "LEMAIRE" 1er | 678 | 550 | | | 22 | 73 | | | | |
| Suivant | 155 | | 25 | | 19 | 44 | | | | |
| Suivant | 157 | | 29 | 44 | 17 | 38 | | | | |
| Suivant | 157 | | 27 | | 15 | 37 | | | | |
| Suivant | | | 28 | | 15 | 34 | | | | |

Tableau 3.1 - Requête 1, multi-utilisateurs

| | | 1 er | Suiv. | Suiv. | Suiv. | Suiv. | Suiv. | Suiv. | Suiv. | Suiv. | Suiv. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| i D B P | | 523 | 25 | 25 | 26 | 26 | 25 | 26 | 25 | 26 | 26 |
| | | | | | | | | | | | |
| SOCRATE CLIO | CPU | 23 | 22 | 18 | 16 | 15 | 15 | 13 | 15 | 12 | 16 |
| | APP. | 43 | 40 | 31 | 28 | 29 | 28 | 27 | 31 | 27 | 28 |
| | Nbr Page | 9 | 7 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |

Tableau 3.2 - Requête 1, mono-utilisateur,

recherche avec index

### 3.3.3.2 INTERPRETATION

AVEC INDEX

La recherche du premier élément est l'opération la plus couteuse pour les deux systèmes. En effet pour l'iDBP, cette opération consiste à parcourir tous les niveaux du B-arbre. On sait qu'une caractéristique de ces méthodes d'indexation est de fournir les mêmes temps d'accès et ceci quelque soit l'élément recherché. Dans SOCRATE/CLIO, la première opération de recherche consiste à calculer à l'aide d'une fonction de hash-code une entrée dans la table du dictionnaire. Ensuite, l'accès au premier élément est fonction du taux de collision provoqué par la fonction, ce qui devrait fournir des temps d'accès, au premier élément, différents selon la valeur de la clé. Remarquons que pour les deux valeurs de la clé testée, nous obtenons des temps d'accès identiques.

Pour les occurrences suivantes, la conservation des temps d'exécution est manifeste pour l'iDBP où nous relevons des variations de quelques centièmes de secondes. Ceci est observé quelque soit la charge de la machine hôte, ce qui traduit bien le peu d'influence de ce dernier facteur sur les performances d'exécution des E/S lorsque celles-ci sont déportées sur un calculateur dorsal. Notons que pour l'iDBP, les mesures relevées ne tiennent pas compte de l'affichage des résultats et ceci pour deux raisons:

- la première est directement liée au mode de connection que nous avons mis en oeuvre entre ces deux calculateurs. Une liaison asynchrone à 9600 bauds ne peut être mise en compétition avec une communication de type réseau (une carte de communication ETHERNET était inclue dans le 86/440 testé) au point de vue débit des données. Si l'on considère

que nos requêtes ne restituent que peu d'informations, le temps de transfert de ces résultats pourra être négligé dans le cas d'une liaison de type réseau, bien qu'il soit difficile d'estimer le débit d'un réseau selon le type de liaison calculateur-réseau, selon le type d'application sur lequel on évaluera ce débit.

- la seconde raison est due au fait que nous avons simulé sur le VAX une communication avec l'outil d'aide au développement d'application, DELPHI, <iDBP6> de l'iDBP en remplacement du protocol propòsé par le constructeur et qui restait à mettre en oeuvre sur le calculateur hôte. En réponse à toute transaction, DELPHI restitue davantage d'informations (rappel du nom des caractéristiques demandées, restitution des valeurs en héxadécimal,...) que le protocol nous aurait retournés pour la même requête. Ainsi notre solution était à la fois plus lente et plus couteuse en terme de volume des résultats, d'où le choix de ne pas tenir compte des temps d'affichage.

Sur le VAX, les temps d'accès aux éléments suivants restent également constants au fil de l'exécution instruction par instruction. Ceci est tout à fait compréhensible si l'on se rappelle que les accès aux élément suivants se font par chainage, ce qui signifie un accès direct à l'élément après avoir traduit l'adresse virtuelle, contenue dans la zone de chainage, en une adresse physique <SOCREF>. Ce mécanisme de projection de l'espace virtuel sur l'espace réel implique de gérer les collisions en créant des chainages entre les sous-pages virtuelles qui se sont projetées sur la même sous page réelle. Ainsi le choix de la taille de ces sous-pages ne doit pas être fait de manière totalement aléatoire puisqu'il influera considérablement sur le nombre d'accès disques à réaliser pour accéder l'information. Lorsque l'on étudie le nombre de pages accédées pour obtenir l'élément suivant, en dehors des accès au catalogue, nous avons un accès disque à l'espace fichier pour chacune des occur-

rences. Cela signifie que les sous-pages accédées n'ont pas été mises en collision avec d'autres sous-pages lors du chargement de la base et par conséquent, que la projection espace virtuel sur espace réel a aboutit en un seul accès disque. Cet état de fait est conforté par les statistiques réalisées sur la base et qui font mention, précisément, du nombre de sous-pages chainées pour chaque espace. De plus à travers ces résultats, on peut dire que, pour chaque occurrence, les caractéristiques accédées n'étaient pas à "cheval" sur les sous-pages.

STATISTIQUES ESPACE : FICH

| | | |
|---|---|---|
| NOMBRE TOTAL DE SOUS-PAGES | : | 262144 |
| NOMBRE DE SOUS-PAGES VIDES | : | 129662 |
| NOMBRE DE SOUS-PAGES OCCUPEES | : | 132482 |
| NOMBRE DE SOUS-PAGES SEULES | : | 132482 |
| NOMBRE DE SOUS-PAGES TETE DE CHAINE | : | 0 |
| NOMBRE DE SOUS-PAGES CHAINEES | : | 0 |
| TAUX D'OCCUPATION (EN %) | : | 50,53 |
| TAUX DE SOUS-PAGES CHAINEES (EN %) | : | 0 |
| NOMBRE MOYEN D'ACCES | : | 1 |

Les statistiques de l'espace FICH qui contient les caractéristiques demandées durant l'exécution de la requête 1, nous montrent que pour un taux d'occupation de 50 %, nous n'avons aucun chainage entre les sous-pages. Ainsi, telle que la base est peuplée, chaque caractéristique décrite dans cet espace sera récupérée en un seul accès disque.

De plus l'accès à l'espace des données est réduit à l'unité car chacune des caractéristiques dont on a demandé l'impression se trouve dans le même espace. Si une d'entre elles avait été désimbriquée physiquement, nous aurions eu au moins un accès disque supplémentaire. Cette désimbrication physique aurait eu sa raison d'être. Pour l'iDBP les coûts auraient été conservés.

Il est bon de remarquer que pour cette requête la taille des pages transférées, 2ko pour le VAX, ne rentre pas en

ligne de compte vue la dispersion des occurrences concernées. Il en aurait été différemment si nous avions demandé l'affichage de caractéristiques définies au sein d'entités imbriquées dans l'entité ELEVE, vu que ces dernières sont stockées de façon contiguë à l'occurrence englobante alors que pour l'iDBP celles-ci sont situées dans un autre fichier.

Lorsqu'on aligne les temps de "Requête 1" sur le VAX en mono-utilisateur et en multi-utilisateurs, on obtient des écarts de 25 % sur les temps apparents, les temps CPU étant conservés entre les deux séries. Par contre les temps iDBP ne varient pas entre ces deux états de la machine hôte. Ainsi cette comparaison met en évidence de façon manifeste et chiffrable la pénalisation apportée par une charge supplémentaire de la machine sur les performances d'accès à la base de données. Pour cette requête, l'utilisation d'un dorsal tel que l'iDBP peut être assimilé à une utilisation en mono-utilisateur de la machine hôte.

Nous pouvons donc avancer que pour une recherche sur clé et dans un environnement mono-utilisateur, les performances de l'iDBP sont comparables à celles de SOCRATE/CLIO sur VAX.

SANS INDEX

Les LMD des deux systèmes nous permettent d'effectuer des opérations de sélection en précisant si l'on veut ou non utiliser les index définis sur les caractéristiques participant au filtre.

Pour ce second type d'opération, les résultats obtenus sur les deux systèmes ne sont plus à la même échelle puisqu'il y a un rapport de 7, sur les temps apparents, qui les sépare et ceci en faveur de SOCRATE/CLIO.

Si l'on relève le rapport (temps apparent iDBP/temps CPU SOCRATE/CLIO) pour l'accès à chacun des éléments, on peut faire remarquer la constance de cette valeur (=11). Ce rapport constant peut s'expliquer en partie par le fait que la base située sur le VAX et celle sur le dorsal ont été chargées à partir de la même bande, dans le même ordre de création des entités ou relations et à partir de programmes dont les logiques étaient quasiment identiques. Ainsi pour les deux systèmes, nous avons la même dispersion des données, d'autant plus que pour SOCRATE/CLIO, les 18000 élèves sont stockés de manière totalement contiguë (leurs numéros d'ordre sont consécutifs). L'importance de la différence entre les temps relevés sur les deux systèmes s'explique, notamment, par le mécanisme d'anticipation de lecture décrit au § 3.3.1.5 qui a été spécialement développé pour les parcours séquentiels d'entités.

RECHERCHE MULTICRITERES

Nous avons repris la requête 1 en modifiant le filtre par ajout de un, puis deux et enfin trois nouveaux critères de sélection. A chaque fois nous avons un critère sur lequel a été défini un index. Notons qu'il n'est pas possible, pour chacun des deux systèmes, de définir des index multi-attributs.

La première remarque que l'on peut faire à la vue du Tableau 3.3 porte sur la comparaison des résultats avec 1, 2 et 3 critères. On s'aperçoit, aussi bien pour l'iDBP que pour le VAX, qu'une sélection avec 2 ou 3 critères est plus rapide qu'une sélection avec un seul critère. Si l'on essaie de détailler le déroulement de ces opérations, la différence des temps d'exécution ne doit être que le fait d'une différence de temps CPU si l'on rappelle qu'il n'y a qu'un index d'utilisé pour chacune des trois requêtes. Le nombre de pages accédées sera par conséquent constant, ceci est vérifié par les resultats.

| | iDBP | SOCRATE / CLIO | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Multi-utilisateurs | | | | | Mono-utilisateur | | | | |
| | | CPU | APP | Nb. Pag 1 | 4 | 5 | CPU | APP | Nb. Pag 1 | 4 | 5 |
| Q11 1critère = 1 index 10 occ. résult. | 429 | 75 | 449 | 6 | 13 | 2 | 76 | 162 | 6 | 12 | 2 |
| Q12 2 critères dont 1 index 8 occ. résult. | 375 | 84 | 230 | 6 | 13 | 2 | 65 | 146 | 6 | 12 | 2 |
| Q13 3 critères dont 1 index 8 occ. résult. | 392 | 75 | 199 | 7 | 13 | 2 | 69 | 145 | 7 | 12 | 2 |
| Q14 4 critères dont 1 index 2 occ. résult. | 604 | 57 | 176 | 7 | 12 | 2 | 46 | 120 | 7 | 12 | 2 |
| Q15 4 critères dont 1 index | 606 | | | | | | | | | | |
| Q16 = Q15 avec 2 index | 652 | | | | | | | | | | |

Tableau 3.3 - Requête 1, multi-critères, parcours global

De plus, nous avons exécuté une requête comportant 4 critères de sélection avec un index dans un cas (Q15) et deux dans l'autre (Q16). Ce test n'a été réalisé que pour l'iDBP étant donné que

- l'on ne peut créer dynamiquement des clés sur SOCRATE/CLIO en raison de la nouvelle zone de chainage (pour les synonymes) qui serait utile de créer

- l'on peut définir un filtre en demandant l'utilisation, au plus, d'un dictionnaire.

Si le premier index est suffisamment discriminant, dans notre cas il permet de repérer 10 élèves parmi 18000, l'utilisation d'un second index sera plus couteuse car trop long à parcourir. Cette situation se retrouve précisément pour les programmes Q15 et Q16.

L'utilisateur précisant s'il veut utiliser ou non les index, il lui faut connaitre la puissance discriminante de ses clés lors de la spécification du filtre. Le LMD de l'iDBP se trouve en opposition aux LMD des SGBD relationnels classiques où précisément on ne veut pas que l'usager ait à préciser ou non l'utilisation du dictionnaire. Dans un langage tel que SQL, il existe un optimiseur qui fera le choix, en fonction d'heuristiques, du meilleur dictionnaire à utiliser (voici un des aspects de la notion d'indépendance physique des données). Mais est-ce que la connaissance, de la population de la base par le système pourra être aussi poussée que celle de l'utilisateur pour faire un tel choix ?

### 3.3.3.3 CONCLUSIONS

Pour les opérations de recherche sur clé, les performances de l'iDBP connecté à un calculateur hôte en charge, sont assimilables aux performances de ce même calculateur s'il était dédié totalement à l'exécution de la même requête sous SOCRATE/CLIO. Cette observation n'est valable que dans la mesure où l'on réalise un parcours élément par élément. Si l'on veut accéder à tous les éléments ayant une même valeur de clé, les performances du VAX sont supérieures à celles de l'iDBP. Mais là n'était pas le but de cette requête et nous aurons l'occasion, plus tard, d'apporter quelques explications à ce phénomène qui peut paraître quelque peu surprenant au premier abord.

## 3.3.4 OPERATION DE PARCOURS D'UNE ENTITE IMBRIQUEE

### 3.3.4.1 PRESENTATION

Nous avons étudié précédemment l'opération de sélection qui n'était pas une opération d'accès particulière au modèle réseau mais commune à tous les modèles. Ceci explique très certainement les très bonnes performances obtenues sur l'iDBP dans le cas d'un parcours avec clé en comparaison avec SOCRATE/CLIO sur VAX.

Notre étude a pour but d'étudier et d'évaluer ce qu'une Machine Bases de Données telle que l'iDBP peut apporter à un SGBD tel que SOCRATE/CLIO. Dans ce contexte, il faudra étudier en priorité les opérations propres à SOCRATE/CLIO.

Avec l'opération de parcours d'une entité imbriquée, nous abordons la première d'entre elles.

SOCRATE/CLIO

L'un des avantages des entités imbriquées est un rapprochement physique avec les occurences englobantes et par conséquent un accès plus rapide aux feuilles de cette hiérarchie.

iDBP

Parce qu'il est relationnel, le LDD de l'iDBP ne permet pas la définition de telles hiérarchies. Par contre certaines extensions ont été apportées pour arriver à une solution très semblable à celle de l'anneau SOCRATE. Ainsi nous définirons, pour l'entité imbriquée, une relation où chaque occurence aura un pointeur père vers l'entité englobante et un pointeur frère vers l'occurence "imbriquée" suivante.

Le parcours d'entités imbriquées que nous allons réaliser

nécessitera dans un premier temps un accès, sur clé, à l'entité englobante. La cardinalité de cette dernière, pour notre requête, est faible. Par conséquent, nous pouvons avancer que les temps de recherche dans les dictionnaires sont comparables pour les deux systèmes.

Nous exécuterons également une seconde requête(IMB2) qui effectuera deux fois le parcours des occurences imbriquées. Ce comparatif nous permettra de mettre en évidence l'efficacité des différents mécanismes de gestion de la mémoire centrale.

Les différentes requêtes seront exécutées de manière globale et ensuite élément par élément afin de détailler les accès à chacune des occurences imbriquées.

DEFINITION DES REQUETES DE PARCOURS D'ENTITES IMBRIQUEES

IMB1 : à partir d'un PROGRAMME dont on fournit la clé (son NIVEAU), on parcourt chacune de ses MATIERE.

IMB2 : à partir d'un PROGRAMME dont on fournit la clé (son NIVEAU), on parcourt successivement deux fois toutes ses MATIERE.

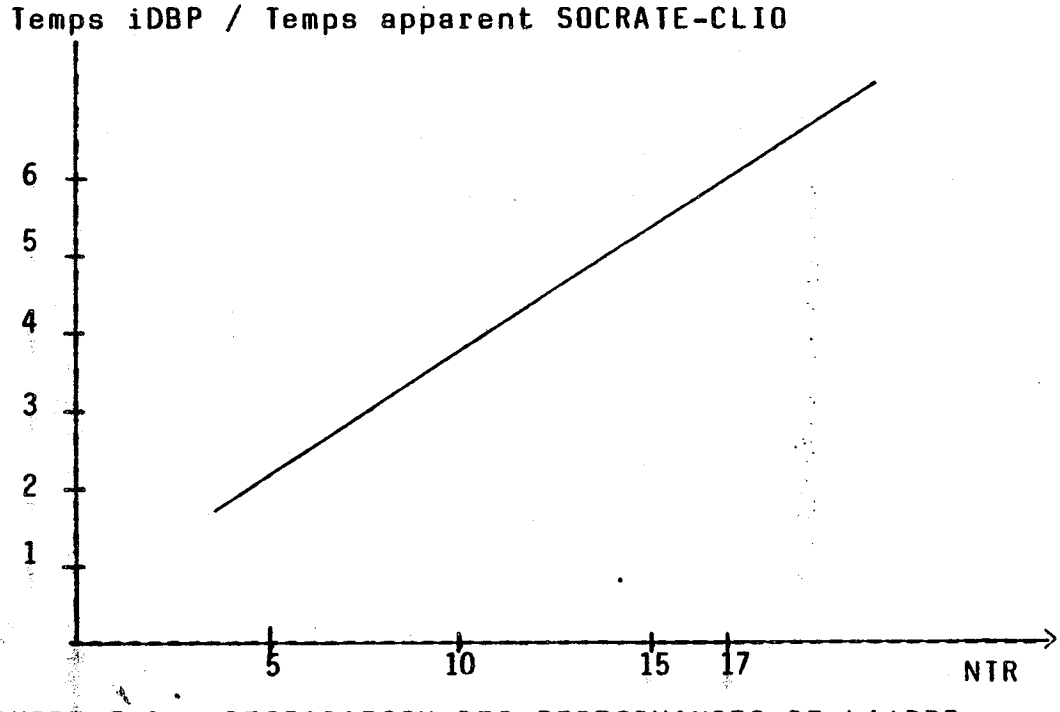
### 3.3.4.2 INTERPRETATION

PARCOURS GLOBAL

Par opposition à la requête précédente, les résultats relevés sur les deux systèmes ne sont plus en rapport. On relève que la dégradation des temps de l'iDBP croît considérablement avec le nombre de tuple retournés (= NTR).

| | IMB1 | | | | | | IMB2 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | iDBP | SOCRATE / CLIO | | | | | iDBP | SOCRATE / CLIO | | | | |
| | | CPU | APP | Nbre Pages 1 | 4 | 5 | | CPU | APP | Nbre Pages 1 | 4 | 5 |
| "2DEA" GLOBAL 15 Matières | 458 | 46 | 85 | 8 | 4 | 1 | 839 | 77 | 118 | 9 | 4 | 1 |
| Elément par Elément | 26 | 15 | 39 | - | 1 | 1 | | | | | | |
| Suivant | 21 | 9 | 31 | 1 | 1 | 1 | | | | | | |
| Suivant | 21 | 8 | 18 | 1 | - | - | | | | | | |
| Suivant | 21 | 8 | 18 | 1 | - | - | | | | | | |
| Suivant | 21 | 6 | 18 | 1 | - | - | | | | | | |
| "1EREC" GLOBAL 17 Matières | 507 | 53 | 89 | 8 | 4 | 1 | 929 | 72 | 111 | 9 | 4 | 1 |
| Elément par Elément | 20 | 18 | 39 | 1 | 1 | 1 | | | | | | |
| Suivant | 30 | 11 | 25 | 1 | 1 | - | | | | | | |
| Suivant | 20 | 8 | 19 | 1 | - | - | | | | | | |
| Suivant | 21 | 9 | 22 | 1 | 1 | - | | | | | | |
| Suivant | 21 | 8 | 19 | 1 | - | - | | | | | | |
| "TERMB" GLOBAL 10 Matières | 309 | 45 | 81 | 8 | 3 | 1 | 533 | 54 | 89 | 9 | 3 | 1 |
| Elément par Elément | 21 | 12 | 33 | 1 | 1 | 1 | | | | | | |
| Suivant | 21 | 12 | 27 | 1 | 1 | - | | | | | | |
| Suivant | 22 | 7 | 18 | 1 | - | - | | | | | | |
| Suivant | 21 | 7 | 18 | 1 | - | - | | | | | | |
| Suivant | 21 | 8 | 18 | 1 | - | - | | | | | | |

Tableau 3.4 - IMB1, IMB2, mono-utilisateur




COURBE 3.1 - DEGRADATION DES PERFORMANCES DE L'iDBP EN FONCTION DU NOMBRE D'OCCURRENCES IMBRIQUEES

La linéarité de la dégradation des performances nous incite à affirmer que la pénalisation est située au niveau de la

lecture de chacun des enregistrements. Sur l'iDBP, leur lecture est réalisée au travers d'un chainage, par conséquent nous avons à faire avec un accès quasi direct, puisque les pointeurs sont des identificateurs de tuples plutôt que des adresses physiques.

Pour SOCRATE/CLIO, à partir de l'instant où l'on pointe sur l'entité englobante, on a accès à une chaine de bits d'existence qui nous renseigne sur les numéros d'ordre des occurences imbriquées existantes. Pour chacune d'entre elles nous avons à réaliser la conversion de l'adresse virtuelle en une adresse réelle et la recherche de la sous-page correspondante dans l'espace physique associé, en l'occurence l'espace "FICH" où nous avons vu que le taux de collision était nul. Ainsi, en raison de l'imbrication et de la valeur de ce taux, nous avons un rapprochement physique des occurences imbriquées très marqué. Ce rapprochement physique est vérifié par le nombre de pages (de l'espace "FICH") accédées au cours de l'exécution pas à pas. C'est ainsi que pour obtenir toutes les "MATIERE" du "PROGRAMME" de "2DEA", on a un accès à une page toutes les quatre occurrences imbriquées. La différence entre les temps d'exécution provient des temps d'entrées/sorties qui seront plus conséquents sur l'iDBP, d'autant plus qu'aucune gestion sophistiquée des pages en mémoire n'est réalisée en contraste avec la gestion des cadres de SOCRATE/CLIO <SOCREF>. On retrouve, de façon pertinente, cet état de fait dans les résultats de IMB2.

Lorsque l'on parcourt deux fois les occurences imbriquées d'une occurence englobante, on double quasiment le temps d'exécution (pour l'iDBP) par rapport à un parcours unique. La différence vient du fait que dans IMB1 on tient compte du temps d'appel de la procédure, des ouvertures de fichiers, de la définition de la vue et que ces trois opérations ne sont pas doublées dans IMB2. Ainsi on peut se rendre compte du coût engendré par une gestion classique de

la mémoire et de la proportion que peut prendre cela pour de très grosses applications.

Par contre, pour SOCRATE/CLIO, nous avons à la fois conservation des temps CPU et du nombre de pages accédées. IMB2 consommera un peu plus de CPU que IMB1 pour recalculer chacune des adresses réelles des occurrences résultats et n'accèdera pas à plus de pages que pour IMB1 Ceci explique la conservation globale des temps vu qu'aucune surcharge d'E/S supplémentaire n'est intervenue.

Nous pouvons maintenant détailler les accès physiques aux pages.

| Paramètre | "2DEA" | "TERMB" | "1EREC" |
|---|---|---|---|
| Espace | | | |
| CATALOGUE | 8/9 | 8/9 | 8/9 |
| FICH | 4/4 | 4/4 | 3/3 |
| DICTION. | 1/1 | 1/1 | 1/1 |

x/y x : nombre de pages accédées pour IMB1
y : nombre de pages accédées pour IMB2

L'espace "CATALOGUE" contient entre autre la structure de la base et les programmes. Il est accédé physiquement (sur disque) à chaque appel car, l'utilisateur ayant été déclaré avec le droit d'écriture (pour lui et les autres utilisateurs) sur cet espace, il y a un "point de contrôle" en début et fin de chaque transaction. Et par conséquent, on ne peut au cours de la transaction suivante réutiliser ces pages car elles auraient pu être modifiées par un autre utilisateur (chaque utilisateur a sa propre zone des cadres dans la version VAX de SOCRATE/CLIO). Pour les autres espaces, l'utilisateur est en lecture partagée ce qui signifie, pour SOCRATE/CLIO, qu'aucune mise à jour ne peut avoir lieu. Par conséquent une page accédée par deux transactions lancées séquentiellement ne peut être modifiée entre temps.

Tant que le nombre de pages accédées est inférieur à 20, pour notre configuration, et si l'on demande les mêmes pages, on n'a pas de nouvelles lectures sur les espaces déclarés en lecture partagée. Ceci explique d'une part le faible accroissement des temps d'exécution entre les programmes IMB1 et IMB2 (sur VAX, moins de 20 pages accédées) et d'autre part, la conservation du nombre de pages lues réellement entre un parcours global et un parcours pas à pas. En effet, pour ce dernier, on accède au même nombre de pages de l'espace "FICH" comme pour le parcours global.

Il est vrai que cet exemple ne demande que peu d'entrées/sorties et que pour une application classique, notre zone des cadres risque d'être rapidement insuffisante. Cette zone des cadres est précisément limitée par la taille mémoire (mémoire centrale) mais cette contrainte ne devrait bientôt plus en être une dans la mesure où ces mémoires centrales vont aller sans cesse croissante avec les nouvelles technologies de très haute intégration.

TEMPS D'E/S

D'un autre côté, cet aspect met en évidence les coûts engendrés par les lectures-écritures sur les périphériques et peut nous donner une première idée de la charge de travail qui sera déportée du calculateur hôte en faveur des autres utilisateurs dans le cas d'une architecture comprenant une machine base de données.

La conjonction de la courbe de "dégradation des performances" (courbe 3.1) et du tableau "IMB1, IMB2, mono-utilisateur", nous montre de manière pertinente que la dégradation des temps d'accès est due, pour ce type de requête, en très grande partie au nombre de pages accédées qui sont plus nombreuses sur l'IDBP. On rejoint ici la thèse avancée par DE WITT qui prétend que les Machines Bases de Données ne seront viables que lorsque l'on sera en mesure d'assurer de

plus forts débits avec les mémoires secondaires <DEWBOR 84>. En effet le goulot d'étranglement se situe précisément à ce niveau et non pas au stade de la vitesse de traitement de l'information. D'ici là, on pourra peut être compenser, partiellement, ce fait par la réalisation de filtres qui sélectionneraient au vol les données demandées par l'utilisateur et éviteraient du même coup, pour certaines requêtes, un transfert considérable d'informations.

Par contre, pour d'autres requêtes telles que IMB1, la solution pour de meilleures performances ne réside pas dans la réalisation de filtres mais davantage dans le développement de véritables méthodes de rapprochement physiques des données tel que cela a été fait dans SOCRATE/CLIO pour un type d'information précis (la hiérarchie d'entités). Avec une telle approche, on réduirait à la fois le nombre de pages transférées et la dispersion de l'information qui finit par être excessivement couteuse en temps de recherche physique.

PARCOURS OCCURRENCE PAR OCCURRENCE

Lorsque l'on décompose la requête IMB1 instruction par instruction et que l'on rapproche les temps de l'iDBP et de SOCRATE/CLIO, on relève que les temps d'accès à chacune des entités imbriquées se conservent d'une machine à l'autre. Pour SOCRATE/CLIO, on passe d'une boucle "FAIRE-REFAIRE" à une sérialisation d'instructions dont la cardinalité sera égale au nombre d'occurences accédées.

De cette exécution pas à pas, on peut relever la régularité des temps d'accès à chacune des occurrences. L'uniformité des temps de réponse de l'iDBP se retrouve sur SOCRATE/CLIO.

La régularité des résultats de l'iDBP vient du fait que le coût de l'opérateur "SET" est quasiment constant. Ceci n'est pas tout à fait exact pour le premier "SET" qui par-

fois sera plus couteux que les suivants qui doivent précisément profiter de ce premier positionnement. L'opérateur "SET" doit pouvoir se résumer à la traduction d'un identificateur de tuple en une adresse physique et l'accès à cet enregistrement physique. Ceci explique cette conservation des temps d'autant plus que la logique du chargement de la base n'a pas "trop" dispersé toutes les "MATIERE" d'un "PROGRAMME", d'où un gain de temps de positionnement.

On peut rappeler à la vue du nombre de pages accédées, pour l'espace "FICH", l'efficacité de la gestion des cadres pour SOCRATE/CLIO. On accède au même nombre de pages que pour l'exécution globale car nous avons défini un droit de lecture partagée sur tous les espaces physiques de données avec mise à jour interdite de ceux-ci. Notons que les mêmes droits ont été définis sur l'iDBP pour les fichiers manipulés lors de l'exécution des requêtes IMBxx. Les effets ne sont pas apparemment comparables.

3.3.4.3 CONCLUSIONS

Le parcours d'une entité imbriquée est une opération propre au système SOCRATE/CLIO. C'est une opération qui se comporte très bien avec la structure physique des données telle qu'elle a été réalisée dans ce SGBD. Par conséquent, il n'est pas étonnant que les performances de SOCRATE/CLIO soient supérieures à un système basé sur le modèle relationnel, même si celui-ci a reçu quelques extensions. Les résultats auraient certainement été plus médiocres avec un "Relationnel pur" où il aurait fallu faire appel à l'opérateur de jointure. Ce dernier sera évalué plus tard.

Les occurrences imbriquées qui ont été parcourues au cours de l'exécution des requêtes IMB1 et IMB2, ont été stockées de manière contiguë dans le temps. Cela signifie que pour

l'iDBP, elles étaient situées côte à côte dans leur fichier. Par contre, si elles avaient été générées de manière totalement aléatoire dans le temps, les occurrences imbriquées d'une même occurrence englobante auraient été dispersées sur le fichier alors que pour SOCRATE/CLIO, la situation précédente aurait été conservée.

Il s'en serait suivi un écart encore plus grand entre les temps de l'iDBP et ceux de SOCRATE/CLIO. D'où l'intérêt, encore mal partagé par les concepteurs de systèmes de gestion de bases de données, d'investir de façon plus poussée vers de nouvelles méthodes de rapprochement physique.

## 3.3.5 OPERATION DE PARCOURS D'UN ANNEAU

### 3.3.5.1 PRESENTATION GENERALE

Avec l'opération de parcours d'un anneau, nous abordons la seconde opération spécifique à SOCRATE/CLIO.

SOCRATE/CLIO

Les anneaux SOCRATE/CLIO sont représentés à l'aide de pointeurs qui sont des adresses virtuelles. Chaque élément fils possède un pointeur sur l'élément suivant, un pointeur sur le père de l'anneau et éventuellement un pointeur sur l'élément précédent. L'élément père pointe sur le premier élément et éventuellement sur le dernier.

L'opération interne de parcours d'un anneau consiste, en SOCRATE/CLIO, à accéder successivement à chacun de ces pointeurs et à les transformer en une adresse physique.

iDBP

Le LDD de l'iDBP nous permet de simuler un anneau mais pas de le définir comme élément du modèle de données du système. Par conséquent, sa gestion sera à la charge de l'utilisateur. Ainsi pour supprimer ou insérer un élément dans un anneau, nous serons tenus de gérer la mise à jour des pointeurs alors que cette opération est totalement transparente pour des manipulations identiques sur SOCRATE/CLIO.

DEFINITION DES REQUETES DE PARCOURS D'ANNEAU

AN1 : à partir d'un PROGRAMME dont on fournit la clé (son NIVEAU), on accède à toutes les CLASSES de ce PROGRAMME.

AN2 : cette requête reprend AN1 mais réalise deux parcours de toutes les CLASSES du PROGRAMME.

Certains des parcours se feront, comme pour les requêtes précédentes, de manière globale et élément par élément.

| | A N 1 | | | | A N 2 | | | |
|---|---|---|---|---|---|---|---|---|
| | IDBP | SCRATE/CLIO | | | IDBP | SCRATE / CLIO | | |
| | | CPU | APP | Page | | CPU | APP | Page |
| "2DEA" GLOBAL 40 Classes | 1184 | 77 | 112 | 15 | 3884 | 128 | 174 | 15 |
| élément par élément | 26 | 12 | 30 | 2 | | 16 | 39 | 2 |
| | 27 | 7 | 58 | 2 | | 10 | 33 | 2 |
| | 21 | 6 | 16 | - | | 8 | 20 | |
| | 22 | 9 | 22 | 1 | | 6 | 24 | 1 |
| | 24 | 7 | 14 | - | | 7 | 20 | |
| | 26 | 8 | 25 | - | | 6 | 21 | |
| | 21 | 8 | 20 | - | | | | |
| | 21 | 7 | 15 | - | | | | |
| "1EREC" GLOBAL 30 Classes | 928 | 62 | 99 | 13 | | 109 | 153 | 13 |
| élément par élément | | 19 | 35 | 2 | | | | |
| | | 9 | 22 | 1 | | | | |
| | | 7 | 15 | - | | | | |
| | | 7 | 14 | - | | | | |
| | | 7 | 15 | - | | | | |
| "6EME" GLOBAL 69 Classes | 1985 | 113 | 169 | 16 | | 192 | 237 | 16 |
| élément par élément | | 18 | 35 | 2 | | | | |
| | | 8 | 20 | 1 | | | | |
| | | 6 | 15 | - | | | | |
| | | 7 | 15 | - | | | | |
| | | 7 | 16 | - | | | | |
| | | 5 | 15 | - | | | | |

Tableau 3.5 - AN1, AN2, mono-utilisateur

Q2 nous donne l'emploi du temps d'un PROFESSEUR dont on fournit le CODE_ENSEIGNANT. Pour chaque COURS effectué, on veut:

- le JOUR et l'HEURE du COURS
- le NOM et la VILLE de l'ETABLISSEMENT
- le NIVEAU de la CLASSES

- le NUMERO de la CLASSE.

| Code Enseignant | iDBP | SOCRATE / CLIO | | | | | |
|---|---|---|---|---|---|---|---|
| | | CPU | APP | Nbre Pages | | | |
| | | | | 1 | 4 | 5 | 8 |
| "20500" | 4601 | 81 | 150 | 8 | 7 | 1 | 5 |
| "20407" | 4578 | 81 | 147 | 8 | 8 | 1 | 4 |
| "20743" | 4569 | 72 | 138 | 7 | 8 | 1 | 5 |

Tableau 3.6 - Q2, mono-utilisateur

Q5 : on veut les NOM, PRENOM et DATE_NAISSANCE de tous les ELEVES des CLASSE de NIVEAU X ("2DEA", "1EREC"). Cette requête sera ensuite modifiée afin de pouvoir parcourir, deux fois de suite, tous les ELEVES de ces CLASSE de NIVEAU X.

| | iDBP | SOCRATE / CLIO | | | | |
|---|---|---|---|---|---|---|
| | | CPU | APP | Nbre Pages | | |
| | | | | 1 | 4 | 5 |
| "2DEA" 1200 Elèves | 52218 | 1340 | 1608 | 7 | 139 | 1 |
| "1EREC" 900 Elèves | 39374 | 1028 | 1251 | 7 | 107 | 1 |

Tableau 3.7 - Q5, mono-utilisateur

| | SIMPLE PARCOURS | | | | | DOUBLE PARCOURS | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | CPU | APP | Nbre pages | | | CPU | APP | Nbre Pages | | |
| | | | 1 | 4 | 5 | | | 1 | 4 | 5 |
| "2DEA" | 1351 | 1755 | 7 | 139 | 1 | 2733 | 3379 | 9 | 277 | 1 |
| "1EREC" | 1020 | 1241 | 7 | 107 | 1 | 2055 | 2478 | 9 | 213 | 1 |

Tableau 3.8 - Q5, mono-utilisateurs, simple et double parcours sur SOCRATE/CLIO

3.3.5.2 INTERPRETATION

PARCOURS GLOBAL : iDBP vs SOCRATE/CLIO

Les requêtes Q2, Q5 et AN1 exécutées de manière globale nous apportent une preuve supplémentaire de la médiocrité des résultats de l'iDBP pour une opération propre au modèle réseau : le parcours d'anneaux.

Le parcours d'un anneau se fait à l'aide de l'opérateur "SET" pour l'iDBP. Sans connaître de façon précise la mécanique interne du SGBD de l'iDBP, nous pouvons avancer que le positionnement sur un élément à partir de cet opérateur "SET" se fait de façon très semblable à ce que l'on peut rencontrer dans SOCRATE/CLIO pour la primitive de parcours d'anneau ("SUIVANT DE"). Dans les deux cas, on récupère à l'origine de l'opération une adresse "virtuelle" qui est pour l'iDBP :

- un identificateur de relation

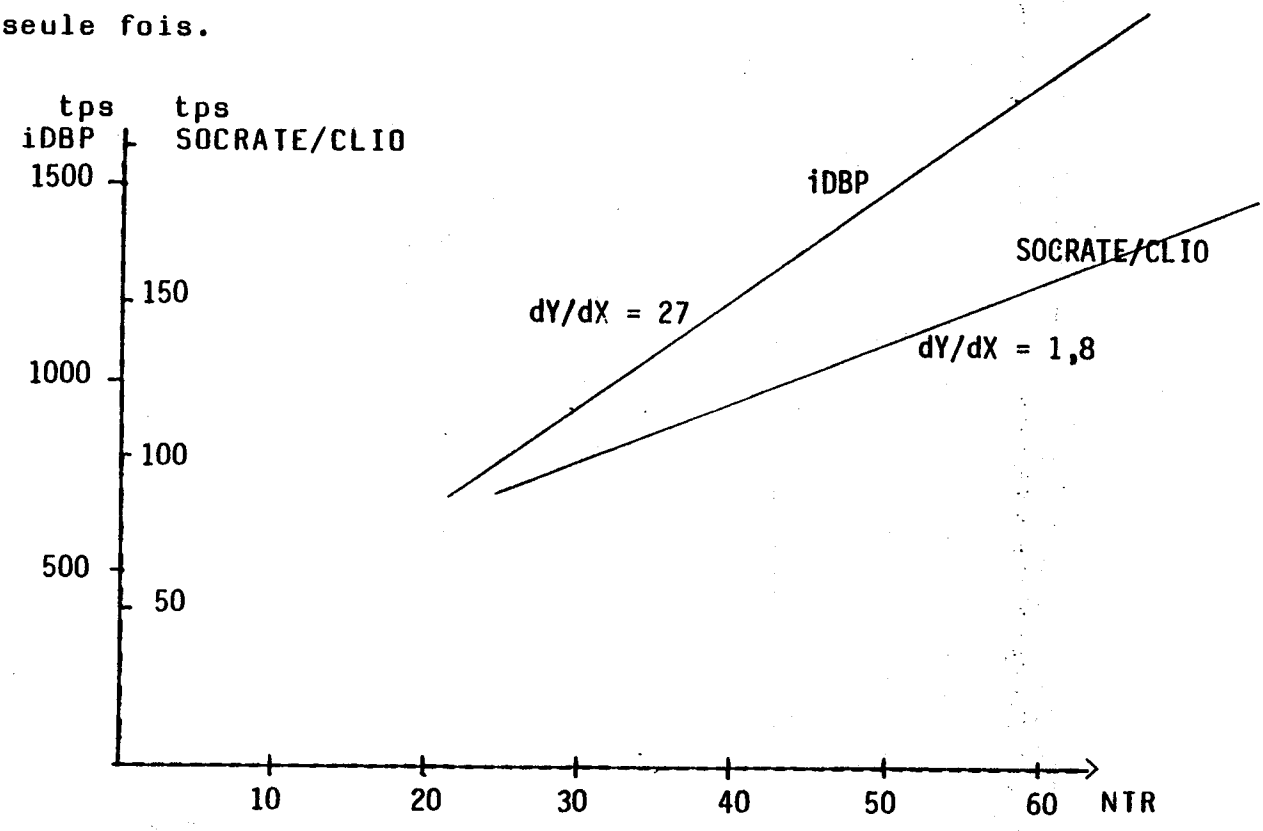
- un déplacement dans la relation

et pour SOCRATE/CLIO, une adresse dans l'espace virtuel. Ensuite le système traduit cette adresse en adresse physique à l'aide d'informations contenues dans le dictionnaire de l'iDBP ou dans les catalogues de SOCRATE/CLIO.

Pour les requêtes accédant à une grosse masse d'informations, les différences de temps apparents entre les deux systèmes ne peuvent être imputées aux vitesses d'accès des deux types de disques. La croissance des temps d'exécution est linéaire avec le nombre d'enregistrements résultats <courbe 3.2> quelque soit le système. Ceci s'explique par la dispersion des occurrences des anneaux, vu qu'aucune méthode n'a été développée pour tenter de rapprocher toutes les instances d'un même anneau. Par contre la croissance de

cette évolution est beaucoup plus forte pour l'iDBP (dy/dx = 27) que pour SOCRATE/CLIO (dy/dx = 1,8) et ceci est d'autant plus inquiétant pour les applications manipulant de très grosses masses d'informations et exécutées sur un système tel que ce dorsal. Malgré tout, ce sera toujours plus satisfaisant qu'une opération de jointure exécutée sur un "véritable relationnel" où les temps d'exécution seraient supérieurs, ceci quelque soit la valeur de NTR.

Pour AN1, nous avons renouvelé l'expérience d'un double parcours tel que cela avait été réalisé pour le parcours d'une entité imbriquée. La requête AN2 effectue deux fois le parcours de l'anneau dont l'accès au père se fait une seule fois.




Courbe 3.2 - EVOLUTION DES TEMPS D'EXECUTION AVEC LA CARDINALITE DE L'ENSEMBLE RESULTAT

Les résultats obtenus sont de très près analogues à ceux relevés pour un double parcours d'entités imbriquées, à savoir que :

- les temps apparents sur l'iDBP sont doublés,

- les temps apparents sur SOCRATE/CLIO sont supérieurs à un parcours simple mais ne sont pas doublés,

- le nombre de pages accédées est conservé pour SOCRATE/CLIO entre un parcours simple et un parcours double.

La seule différence avec le parcours d'entités imbriquées réside dans les temps CPU de SOCRATE/CLIO qui croissent d'une manière plus marquée par rapport à IMB2. Ceci résulte tout naturellement du fait que l'opération de parcours d'anneau est plus couteuse en calcul qu'une opération d'accès à des entités imbriquées. Ainsi le fait de doubler cette recherche se répercute de manière visible sur les temps CPU. On pourra faire la même remarque que pour la partie précédente, à savoir que le nombre de pages accédées n'a pas saturé notre zone des cadres et que ceci, est d'une part très favorable à SOCRATE/CLIO et d'autre part, ne risque pas de se reproduire pour de très grosses applications. La requête Q5 sature très vite la zone des cadres avec 139 pages accédées. Si on modifie Q5 afin de parcourir deux fois l'anneau, on vérifie que l'on double à la fois le nombre de pages lues physiquement sur disques et les temps apparents. De ces chiffres nous pouvons tirer deux nécessités :

- celle de minimiser les transferts disque - mémoire centrale en rapprochant physiquement certaines classes d'informations et en instaurant davantage de chainages physiques

- et celle de chercher à accroitre les tailles des mémoires centrales.

PARCOURS INSTRUCTION PAR INSTRUCTION

iDBP vs SOCRATE/CLIO

Lorsque l'on détaille le parcours de l'anneau et que l'on évalue l'accès élément par élément, on arrive à des résultats fort semblables à ceux obtenus pour le parcours d'entités imbriquées, à savoir que les temps d'accès à l'élément suivant sont identiques d'un système à l'autre (temps apparent iDBP, temps apparent SOCRATE/CLIO) et d'un élément à l'autre, à l'intérieur de chacun des deux systèmes. Le parcours des éléments suivants se fait par accès quasiment direct. Le coût de la conversion adresse virtuelle - adresse réelle reste constant ainsi que le coût de lecture. La petite taille du fichier "CLASSES" limite considérablement les déplacements de bras dont le temps est minime (10 ms pour le winchester de l'iDBP).

Lorsqu'on analyse le coût de l'opérateur "SET" soit pour un parcours d'anneau soit pour accéder à un élément d'une liste de valeurs (cf § "traduction des structures"), on a pu relevé que la première valorisation d'un curseur est plus couteuse que les suivantes. Le coût de ce premier positionnement varie fortement d'un fichier à l'autre alors que pour les valorisations suivantes, il est constant quelque soit l'élément accédé dans un fichier et quelque soit la taille du fichier. Ainsi il semblerait que cet opérateur "SET" s'exécute différemment selon que le curseur, passé en paramètre, ait été ou non déjà positionné.

3.3.5.3 CONCLUSIONS

De la même façon que pour le parcours d'entités imbriquées, le parcours d'anneaux est beaucoup plus rapide sur SOCRATE/CLIO que sur l'iDBP. De plus, la dégradation des performan-

ces en fonction de la taille de l'anneau est nettement plus marquée sur la MBD. Cet état de fait nous conduit à avancer que pour des parcours de structures d'objets hiérarchiquement complexes, on ne pourra retenir une architecture décentralisée bâtie sur un iDBP.

Il ne serait pas inintéressant et inopportun d'investir dans la mise au point de nouvelles méthodes de rapprochement physique des données participant à la même instance d'anneau.

## 3.3.6 OPERATION DE JOINTURE

### 3.3.6.1 PRESENTATION

A l'issue de l'étude des opérations spécifiques au modèle de données de SOCRATE/CLIO, il serait intéressant d'évaluer l'opérateur bien connu du modèle relationnel: la JOINTURE. A travers les requêtes JOIN1 et JOIN2, nous allons comparer cet opérateur avec le parcours d'un anneau afin de chiffrer ce que l'on est en mesure de gagner en privilégiant certaines méthodes d'accès. La jointure que nous allons tester retiendra un élément de la relation externe et n éléments de la relation interne.

iDBP

La jointure est une des opérations spécifiques du modèle relationnel. Pour l'iDBP, elle est le moyen de définir une vue au même titre que la sélection ou la projection. Par contre, on ne peut pas construire une vue à l'aide de l'opérateur JOIN sur des vues définies comme le résultat d'une sélection. Cela implique d'exécuter la jointure et de la compléter en définissant, sur elle, une vue à l'aide de l'opérateur "SELECT". Ceci ne sera pas sans conséquence sur les temps d'exécution. Cette jointure sera opposée à un parcours d'anneau.

SOCRATE/CLIO

La jointure n'étant pas un élément du modèle de SOCRATE/CLIO, nous lui substituerons soit un parcours d'anneau soit une recherche basée sur un accès direct suivi d'une recherche séquentielle. Il peut paraitre surprenant de vouloir évaluer le coût d'une telle opération alors que l'on évolue avec un système réseau d'une part et un système relationel simulant un modèle réseau d'autre part. Il y a

deux raisons fondamentales à ce choix :

- 1) la jointure, sans que ce soit une opération propre au modèle réseau peut se retrouver fréquemment dans ce dernier

- 2) à une époque où l'on polémique beaucoup sur les modèles relationnels et non relationnels, il est intéressant de voir ce que le modèle réseau apporte en plus, au point de vue performances au modèle relationnel et ceci en privilégiant certains chemins d'accès.

DEFINITION DES REQUETES: En fournissant le nom d'un établissement et sa ville, nous voulons récupérer toutes ses classes. Dans l'entité "CLASSES", nous avons le numéro de l'"ETABLISSEMENT" dont elle dépend. L'entité "ETABLISSEMENT" contient également son numéro comme caractéristique.

JOIN1 réalise la jointure sur l'iDBP et sur SOCRATE/CLIO. Elle accède à toutes les classes de l'établissement en question sans se servir de l'anneeau.

JOIN2 fournit les mêmes résultats que JOIN1 mais en se servant de l'anneau sur SOCRATE/CLIO et de la liste chainée sur l'iDBP, définis entre ETABLISSEMENT et CLASSES.

| | | JOIN1 | | | | JOIN2 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | iDBP | SOCRATE / CLIO | | | iDBP | SOCRATE / CLIO | | |
| | | | CPU | APP | Nbre Pages | | CPU | APP | Nbre Pages |
| a) | Accès ETAB. | 59 | 24 | 69 | | 47 | 23 | 60 | |
| | Accès CLA. | 19 | | | | 22 | 14 | 92 | 4 |
| | CLA. Suiva. | 21 | | | | 21 | 11 | 23 | 2 |
| | " " | 21 | | | | 26 | 11 | 19 | 2 |
| | (1) | 5624 | | | | | | | |
| b) | Accès ETAB. | | 21 | 49 | 4 | | 24 | 59 | 5 |
| | Accès CLA. | 19 | 12 | 38 | 3 | | 13 | 41 | 3 |
| | CLA. Suiva. | 20 | 9 | 28 | 2 | | 10 | 25 | 2 |
| | " " | 18 | 11 | 27 | 2 | | 10 | 24 | 2 |
| | (1) | 5216 | | | | | | | |

(1): le dernier élément a été accédé, on cherche le suivant qui, en fait, n'existe pas.

Tableau 3.9 - JOIN1, JOIN2, parcours élément par élément

| | | iDBP | SOCRATE / CLIO | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | CPU | APP | Nbre Pages | | | |
| | | | | | 1 | 4 | 5 | 7 |
| a | Louis le GRAND | 7567 | 248 | 421 | 8 | 20 | 1 | 2 |
| | Paris | 3122 | 210 | 260 | 9 | 5 | 1 | 2 |
| b) | D-Artagnan | 8023 | 391 | 469 | 8 | 21 | 1 | 3 |
| | Paris | 3904 | 262 | 321 | 9 | 7 | 1 | 3 |
| c) | Napoleon | 2865 | 264 | 338 | 8 | 19 | 1 | 2 |
| | Brest | 1357 | 107 | 155 | 9 | 3 | 1 | 2 |
| d) | Aramis | 2415 | 235 | 309 | 8 | 20 | 1 | 1 |
| | Jouy | 722 | 74 | 126 | 9 | 4 | 1 | 1 |
| e) | Athos | 7000 | 307 | | 8 | 20 | 1 | 1 |
| | Paris | | 149 | 198 | 9 | 4 | 1 | 2 |

tps 1 : temps de JOIN1

tps 2 : temps de JOIN2

Tableau 3.10 - JOIN1, JOIN2, mono-utilisateur, parcours global

a) on accède 80 CLASSES

b) on accède 100 CLASSES

c) on accède 32 CLASSES
d) on accède 16 CLASSES
e) on accède 50 CLASSES

Les requêtes suivantes ne seront exécutées que sur l'iDBP.

JOINIF, à partir d'un numéro d'établissement, recherche toutes les classes qui ont le même numéro d'établissement. La recherche se fait à l'intérieur d'une boucle avec l'instruction "IF".

JOINSEL est identique à JOINIF mais la recherche se fait en définissant une vue à l'aide de l'opérateur "SELECT" appliqué au numéro d'établissement des "CLASSES". Il n'existe pas d'index sur le critère de sélection.

JOINSE accepte en entrée le nom et la ville de l'établissement et effectue un "SELECT" pour retrouver l'établissement en question et ensuite applique un autre "SELECT", sur "CLASSES", sur la caractéristique "NUMETAB". Cette requête est très voisine de JOIN1 exécutée sur SOCRATE/CLIO.

| | | JOINIF | JOINSEL | JOINSE | JOIN1 (rappel) |
|---|---|---|---|---|---|
| a) | 80 | 28740 | 5587 | 5723 | 7572 |
| b) | 100 | 28821 | 6184 | 6592 | 8023 |
| c) | 32 | | | 3578 | 2962 |
| d) | 16 | | | 2870 | 2415 |

Tableau 3.11 - JOINIF, JOINSEL, JOINSE

### 3.3.6.2 INTERPRETATION

#### 1) COUT DE L'OPERATION DE JOINTURE SUR LES DEUX SYSTEMES

Dans le modèle réseau, la notion d'anneau sert à privilégier une forme d'accès direct par rapport à un accès purement séquentiel. Ceci implique que l'on ne veut pas privilégier tous les accès et qu'il restera encore beaucoup d'accès séquentiels à effectuer tel que celui de la requête JOIN1.

L'opération de jointure est réputée pour son coût exhorbitant. C'est pourquoi plusieurs algorithmes d'accélérations ont été développés pour les systèmes relationnels (boucles imbriquées, tri-fusion,...). Par contre dans le modèle réseau, elle ne figure pas au même titre que le parcours d'un anneau et sa réalisation revient à la charge du programmeur. La requête JOIN1 a eu recours, pour l'iDBP, à l'opérateur de jointure sur l'attribut "numéro d'établissement" (entre ETABLISSEMENT et CLASSES) suivi d'une sélection sur le nom et la ville de l'établissement. Sur SOCRATE/CLIO, JOIN1 accède à un établissement à l'aide de son nom et de sa ville et ensuite effectue une recherche séquentielle sur les "CLASSES" à l'aide du numéro d'établissement.

A la vue du résultat de la jointure de l'iDBP, il semblerait que l'algorithme retenu soit celui des boucles imbriquées où l'on parcourt autant de fois la seconde relation (de la commande "JOIN") qu'il y a de tuples dans la première. Bien que dans l'opération de jointure de l'iDBP on ne puisse insérer des critères de sélection et que l'on soit tenu de définir successivement une jointure et une sélection, le système réalise dans un premier temps la sélection et ensuite le JOIN sur les éléments résultats du SELECT. Malgré cette optimisation, on obtient un facteur de 20 entre les résultats de l'iDBP et ceux de SOCRATE/CLIO

(la faveur allant à ce dernier).

Quand on analyse les résultats de JOIN1, instruction par instruction pour le cas (b), on s'aperçoit que lorsque l'on a atteint le dernier élément recherché et que l'on cherche le suivant, on passe les 2/3 du temps total à chercher en vain cet élément, qui n'existe pas. Lorsque l'on évalue le temps d'accès à cet élément qui n'existe pas, on remarque qu'il est fonction du nombre de tuples résultats (NTR) (tableau). De plus à l'issue de cette opération, si l'on se positionne de nouveau sur le dernier élément et que l'on demande le suivant, le temps d'accès à ce dernier (qui n'existe pas) varie considérablement avec la première mesure. Par contre cette seconde valeur varie toujours avec NTR mais à un dégré moindre que pour la première. Cela signifierait-il qu'il existe au sein du système un repérage du dernier élément dés l'instant où il a été accédé ? Ce repérage peut être réalisé par comptabilisation des enregistrements au niveau de chacune des vues ouvertes. Ainsi serait expliquée la relative constance de cette valeur. Par contre, la variété des premiers résultats doit être très certainement liée à la manière avec laquelle sont mises en oeuvre la jointure et la sélection.

| ACCES AU DERNIER ELEMENT \ NBRE TUPLES RESULTATS | 16 | 32 | 80 | 100 |
|---|---|---|---|---|
| 1er POSITIONNEMENT | 1063 | 1758 | 5619 | 5216 |
| 2ème POSITINNEMENT | 1062 | 1526 | 1626 | 1446 |

JOINSE vs JOIN1 (iDBP)

L'algorithme d'exécution de JOINSE rappelle fortement la jointure utilisant l'algorithme des boucles imbriquées.

La première remarque relative à JOINSE porte sur la linéarité, par opposition à JOIN1 (cf Courbe 3.4), des résultats en fonction du nombre de tuples retournés. Si l'on déduit le coût supplémentaire engendré par les ouvertures de fichier, les définitions des vues et des curseurs, on estime pour (c) et (d) à 0,4 seconde le temps nécessaire pour obtenir un tuple résultat. Cette valeur est constante quelque soit l'intervalle que l'on considère, ce qui signifie, entre autre, que le nombre d'accès disque est constant pour toutes les exécutions de JOINSE. Ensuite à partir de ce temps et du temps d'exécution de (d), on retrouve, en fonction de NTR, les temps apparents de JOINSE pour chacune des différentes valeurs passées en paramètre.

(3438-2808)/(32-16) = 39,37 = Temps d'obtention d'un tuple résultats

Formule 1 : Tps d'obtention de N tuples = tps d'obtention de N1 tuples + (N - N1) * 39,37

| Nbre tuples résultats | Valeurs relevées | Valeurs estimées par Formule 1 |
|---|---|---|
| 50 | 4380 | 4208 |
| 80 | 5389 | 5723 |
| 100 | 6177 | 6592 |

Ainsi si l'on obtient des temps d'exécution qui évoluent linéairement avec le nombre de tuples résultats, cela laisse sous-entendre que pour chacune des requêtes on parcourt des ensembles de cardinalité variable. Si on sait que la sélection s'effectue systématiquement sur le même ensemble, on en déduit qu'un espace de travail intermédiaire est d'abord créé puis parcouru. Les différences dans les temps d'exécution résultent du parcours de cet espace de travail qui n'est rien d'autre que la vue définie pour effectuer la sélection. Les coûts supplémentaires qui apparaissent dés

que NTR croît ne sont pas dus aux traitements que l'on effectue sur chacun des tuples résultats puisque dans la requête JOINSE on ne demande même pas l'affichage des tuples (commande "FETCH"). Par contre pour JOIN1 exécutée sur SOCRATE/CLIO, les temps évoluent avec le nombre d'occurrences résultats pour deux raisons non associées à des parcours d'ensembles de tailles variables :

- le traitement associé à chaque occurrence résultat, i.e. la demande d'affichage des caractéristiques

- l'accès à des pages supplémentaires alors que pour l'iDBP on accéde toujours au même nombre de page.

Les limitations du LMD de l'iDBP qui ne nous permettent pas de réaliser de façon classique l'opérateur de sélection sans définir de vue, ont donc un impact non négligeable sur les temps de réponse. En effet une opération de sélection ne devrait pas nécessiter la création d'un espace de travail intermédiaire mais au contraire être réalisée au fil de la lecture des enregistrements afin de ne pas être doublement tributaire de la taille des fichiers:

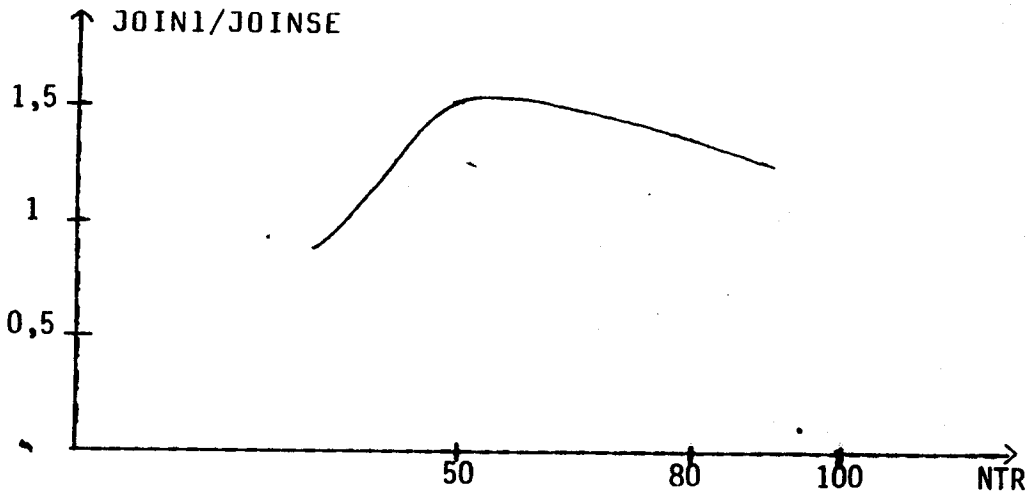
- a) pour le premier parcours

- b) pour le parcours de l'espace résultat.

Ceci implique que pour JOIN1, on créera dans un premier temps la vue établie sur l'opérateur SELECT, dans un second temps la vue sur l'opérateur JOIN que l'on parcourra dans une troisième étape.

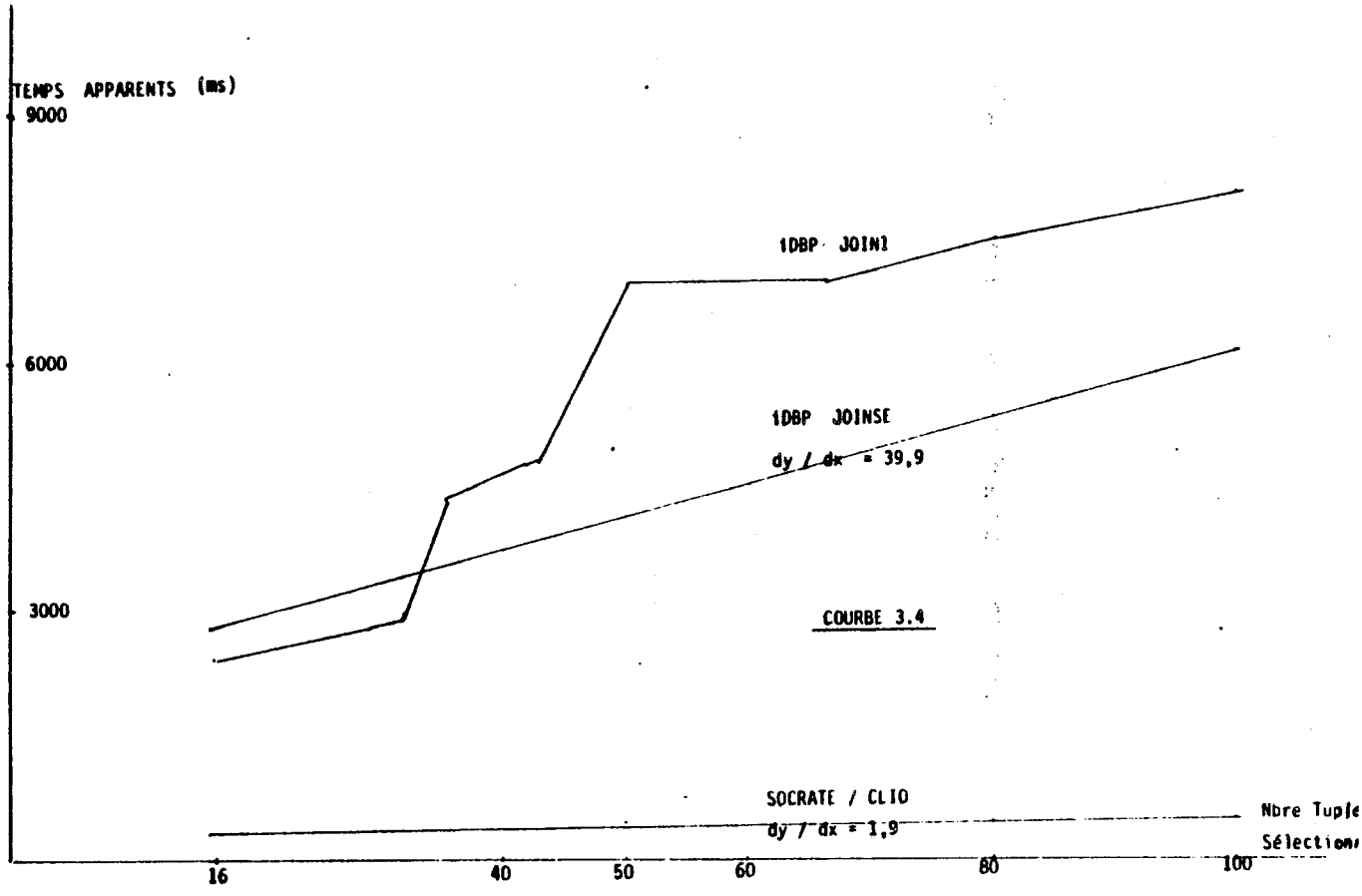
En opposant les chiffres de JOINSE et de JOIN1, on peut estimer le coût supplémentaire introduit par la conjonction du JOIN et du SELECT (cf courbe 3.3). Il est intéressant de remarquer que le facteur d'amélioration croît dans un premier temps avec le nombre de tuples résultats, atteint un

plafond et décroit ensuite. Nos échantillons ne nous permettent pas de poursuivre la courbe d'évolution.




Courbe 3.3

La forme de cette courbe est due principalement au comportement de JOIN1 puisque JOINSE évolue de façon linéaire avec le nombre de tuples résulats (dx/dy = 39,9) (cf courbe 3.4). Il est très difficile d'expliquer ces brusques variations de pente tant que l'on ne pourra fournir plus d'explications sur la non linéarité des résultats de JOIN1. La relation externe du JOIN est toujours ETABLISSEMENTS, par conséquent toutes les vues, résultat du JOIN, ont leurs tuples ordonnées de la même manière. Si l'on évalue l'accès au premier élément pour les différentes valeurs de paramètre, on peut relever que ces temps d'accès sont liés au rang de l'établissement, passé en paramètre, dans le fichier ETABLISSEMENT. Par contre, les temps d'accès au dernier élément sont très aléatoires et on ne peut établir de corrélation entre ceux-ci et la cardinalité des résultats et les temps d'accès au premier élément.




Courbe 3.4

JOINIF vs JOINSEL

Ces requêtes ont été exécutées exclusivement sur l'iDBP.

Les requêtes JOINIF et JOINSEL nous permettent de mettre en évidence la puissance de l'opérateur SELECT vis à vis d'une recherche séquentielle conditionnelle.

Tant que l'on ne connaitra pas la façon dont est réalisée cette sélection, il est très difficile de dire sur quel point nous sommes en mesure de gagner du temps apparent vis à vis d'une boucle avec un "IF" imbriqué.

La caractéristique "NUMETAB", critère de sélection du filtre, a été déclarée de longueur fixe. Ainsi, dans le cas du

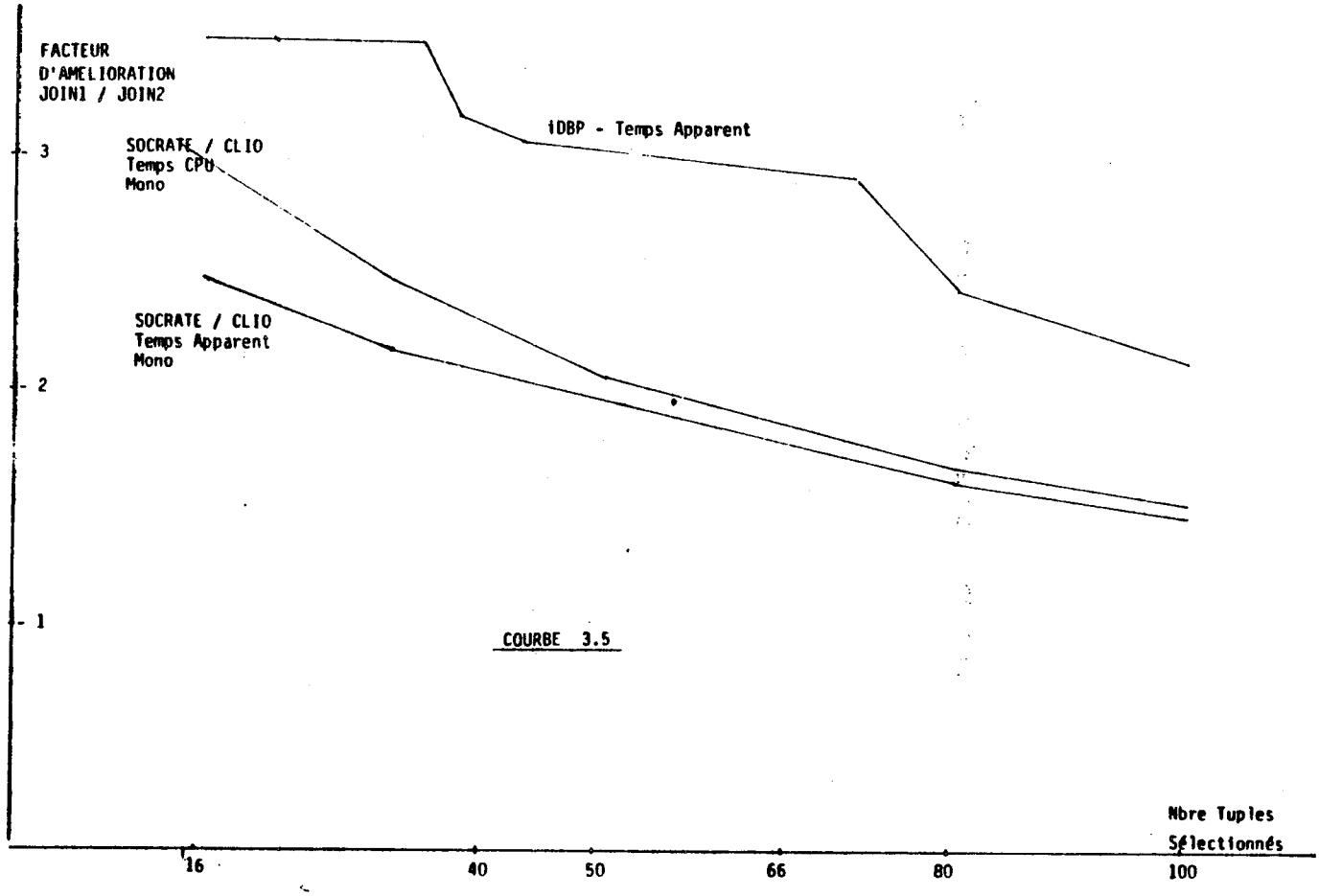
SELECT, l'attribut sur lequel porte le critère de sélection se trouve toujours à la même place dans chacun des enregistrements à parcourir. Voici peut être un des avantages des caractéristques de longueur fixe et cela expliquerait, en partie, pourquoi le système iDBP stocke en début d'enregistrement tous les attributs de longueur fixe et ensuite, ceux de longueur variable et ceci quelque soit l'ordre dans lequel ils ont été déclarés.

Comme pour JOINSE, les temps de JOINSEL évoluent avec le nombre de tuples résultats alors que ceux de JOINIF sont totalement indépendants de ce facteur. Ainsi, cela pourrait conforter l'hypothèse de construction, d'un espace de travail intermédiaire et de parcours de celui-ci pour l'opération de sélection.

De plus il n'est pas étonnant que les résultats de JOINIF soient indépendants de NTR (on ne fait aucun affichage) puisqu'à chaque fois on parcours une seule fois le même ensemble de données.

## 2) ACCES PAR JOINTURE - ACCES PAR ANNEAU

Une des raisons de l'évaluation du coût d'exécution de l'opération de jointure est de pouvoir chiffrer l'apport de la notion d'anneau vis à vis de la gestion uniquement tabulaire du relationnel. Jusqu'à présent, il était difficile d'avancer un facteur d'amélioration apporté par les anneaux de SOCRATE/CLIO ou les pointeurs de l'iDBP (rappelons que ces pointeurs ne sont que des pointeurs virtuels et non réels).




Courbe 3.5

Les facteurs d'amélioration (cf courbe 3.5) obtenus sur le VAX en mono-utilisateur pour les temps apparents décroissent linéairement avec un accroissement du nombre d'occurrences retournées et ceci en conservant la cardinalité et la taille des entités. Pour l'iDBP nous relevons une décroissance de ce rapport qui n'est pas aussi régulière que pour SOCRATE/CLIO et qui marque plusieurs niveaux où ce rapport reste quasiment constant.

De plus si les entités avaient eu 2 ou 3 fois plus d'occurrences, avec le même nombre de tuples résultats, nous aurions conservés les temps d'éxécution de la requête JOIN2 alors que ceux de JOIN1 auraient été accrus du même facteur. Nous aurions alors un facteur d'amélioration bien supérieur à celui avancé précédemment.

Pour une valeur constante de NTR, le facteur d'amélioration croit avec la taille des relations sur lesquelles on effectue la jointure. Pour des fichiers de taille constante, ce même facteur décroit avec un nombre de tuples retournés croissant.

Si on fait abstraction des index, nous avons tout intérêt à utiliser les anneaux pour des facteurs de sélectivité faibles et une structure classique (sans lien) pour les autres cas afin de ne pas pénaliser le système par de couteuses mises à jour de pointeurs.

Les courbes d'amélioration des temps CPU et apparents, en monoutilisateur, se rejoignent lorsque NTR croît, ce qui tend à prouver que le traitement associé à chaque résultat et les temps CPU de demande de pages prennènt le pas sur les temps d'E/S.

| NTR | R | FA |
|---|---|---|
| 16 | 2 | 2,4 |
| 32 | 2 | 2,4 |
| 80 | 1,8 | 1,6 |
| 100 | 1,6 | 1,4 |

Cette remarque est confortée par la comparaison des facteurs d'amélioration FA avec le rapport R (nbre pages accédées pour JOIN2 / nbre pages accédées pour JOIN1). FA est supérieur à R pour des valeurs faibles de NTR et inférieur pour des valeurs plus grandes.

### 3.3.6.3 CONCLUSIONS

L'opération de jointure, telle qu'elle est intégrée dans

le LMD de l'iDBP et telle qu'elle a dû être programmée sur SOCRATE/CLIO, est beaucoup plus rapide sur le système traditionnel que sur la machine bases de données. Vis à vis d'une machine telle que l'IDM 500, l'iDBP a été pénalisée par l'absence d'un accélérateur disque qui devra être considéré comme la partie maîtresse des futures MBD associées au modèle relationnel.

D'autre part, nous avons pu mesuré le gain obtenu par le retour de la notion de pointeurs dans un système comportant, à l'origine une vue des applications uniquement tabulaire. Ces notions de listes chainées ne devront pas pour autant être généralisées. Ainsi la courbe 3.5, nous montre à partir de quand cette solution n'est plus rentable en fonction de la taille des fichiers et du nombre d'occurrences retournées. De plus, nous avons, jusqu'à présent, exécuté uniquement des opérations de recherche. Par conséquent, ces facteurs d'améliorations devront être pondérés avec les coûts de mise à jour associés à la solution des listes chainées.

Ces observations sont révélatrices quant aux conclusions sur l'apport d'une machine base de données à un système de gestion de bases de données tel que SOCRATE/CLIO.

## 3.3.7 OPERATIONS DE MISES A JOUR

### 3.3.7.1 PRESENTATION GENERALE

Avec les opérations de mise à jour nous abordons les derniers tests de notre étude. Nous discernerons trois classes de mise à jour:

- 1) l'insertion
- 2) la mise à jour d'une occurrence
- 3) la suppression.

1) et 3) portent aussi bien sur l'insertion et la suppression d'une occurrence d'une entité que sur l'insertion et la suppression d'une occurrence dans un anneau ou d'une occurrence imbriquée.

Pour ce type de requête, nous insisterons beaucoup sur l'utilisation ou non des index pour évaluer leur apport pour l'accès à l'élément à insérer, supprimer ou mettre à jour. Il faudra tenir compte de la mise à jour des index.

Cette étude qui nécessitait épisodiquement la gestion d'index a pu être réalisée sur l'iDBP grâce à la possibilité que l'on a de supprimer et de créer à tout moment un index sur n'importe quelle caractéristique.

Nous allons analyser successivement l'insertion, la suppression et enfin la mise à jour.

### 3.3.7.2 L'INSERTION

Nous avons généré l'insertion d'occurrences pour des entités de cardinalités très variables (12, 600, 18000) afin d'évaluer le coût de mise à jour des dictionnaires en fonction

de la taille de ceux-ci et de comparer l'évolution du comportement des deux types d'indexation en fonction du nombre d'occurrence instanciées.

D'autre part, nous avons vu que le LMD de l'iDBP comportait de nombreuses restrictions au niveau des types de données, restrictions qui pouvaient être compensées en définissant des contraintes d'intégrité. Par conséquent la vérification de ces contraintes doit être comptabilisée dans le coût d'insertion d'une occurrence. Il nous est possible de demander ou non, dans l'ordre "STORE", le contrôle de cette contrainte. Ainsi nous pourrons évaluer le coût de cette opération supplémentaire engendrée par le LMD de ce système. Nous avons trois opérations d'insertion où pour chacune d'entre elles on demande l'utilisation ou non d'index et où pour l'iDBP on demande ou non la vérification des contraintes.

On insère

- un établissement parmi 12
- une classe parmi 600
- un élève parmi 18000.

DEFINITION DES REQUETES:

INETAV : insertion d'un établissement avec vérification des C.I..

INETSS : insertion d'un établissement sans vérification des C.I..

INCLAA : insertion d'une classe avec vérification des C.I..

INCLSA : insertion d'une classe sans vérification des C.I..

INELAV : insertion d'un élève avec vérification des C.I..

INELSS : insertion d'un élève sans vérification des C.I..

| | iDBP | | SOCRATE / CLIO | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | MULTI | | | | | MONO | | | |
| | | | | | Nbr Pag | | | | | Nbr Pag | |
| | 2 index | 1 ind. | CPU | APP | 1 | 4 | 5 | CPU | APP | 1 | 4 | 5 |
| INETAV avec index | 302 | 265 | 46 | 184 | 6 | 6 | 4 | 33 | 94 | 6 | 6 | 4 |
| INETAV ss index | 189 | | | | | | | | | | | |
| INETSS avec index | 310 | | | | | | | | | | | |
| INETSS ss index | 182 | | | | | | | | | | | |
| INCLAA avec index | 262 | | 39 | 112 | 5 | 6 | - | 25 | 93 | 5 | 6 | - |
| INCLAA ss index | 190 | | 34 | 108 | 5 | 6 | - | 23 | 90 | 5 | 6 | - |
| INCLSA avec index | 261 | | | | | | | | | | | |
| INCLSA ss index | 185 | | | | | | | | | | | |
| INELAV avec index | 429 | 314 | 41 | 133 | 6 | 6 | 2 | 37 | 87 | 6 | 6 | 2 |
| INELAV ss index | 210 | | 25 | 107 | 6 | 6 | - | 27 | 80 | 6 | 6 | - |
| INELSS avec index | 430 | | | | | | | | | | | |
| INELSS ss index | 184 | | | | | | | | | | | |

Tableau 3.12 - insertion d'un enregistrement

INTERPRETATION

Pour SOCRATE/CLIO, nous comparons les coûts d'insertion avec et sans index dans un environnement mono-utilisateur afin d'évaluer les mises à jour des index.

Si l'on compare les coûts d'exécution de cette requête pour les deux systèmes, les temps de l'iDBP sont supérieurs à ceux de SOCRATE/CLIO avec des facteurs de variation allant de 2,11 à 4,93. De plus, on ne relève aucune corrélation

entre ces facteurs et la taille des fichiers dans lesquels on insère les occurrences. Les facteurs les plus élevés sont relevés pour les insertions avec index ce qui peut nous inciter à avancer que le coût de mise à jour des index est plus rapide pour SOCRATE/CLIO que pour l'iDBP pour le contexte dans lequel on effectue ces opérations, à savoir que l'on insère une seule occurrence et que pour l'iDBP on a un index de plus: le rang, qui correspond au numéro d'ordre de SOCRATE/CLIO. Si l'on supprime sur l'iDBP l'index défini sur le rang, la différence dans les rapports s'amenuise mais le coût de mise à jour des index reste toujours supérieur sur l'iDBP.

Si l'on compare les différences de temps d'exécution d'une insertion avec mise à jour de zéro puis 1, puis 2 index, sur l'iDBP, on s'aperçoit que cette mise à jour est plus couteuse pour les fichiers fortement peuplés. Ce qui est tout à fait normal puisque l'index, étant organisé en B-arbre, est plus couteux à parcourir et à mettre à jour pour une insertion (à ne pas confondre avec le coût de recherche d'un élément en comparaison avec l'indexation telle qu'elle est réalisée sur SOCRATE/CLIO).

Pour ce système, l'insertion d'une clé dans le dictionnaire s'apparente à une insertion, en tête, dans une liste chainée <SOCREF>. Ceci explique que cette mise à jour ne soit aucunement liée à la taille du fichier. Cette remarque est confortée par le nombre de pages, de l'espace "DICO", mises à jour dans le cas d'une insertion avec index qui est supérieur pour INETAV (12 éléments) vis à vis de INELAV (18000 éléments). Sans connaître les chiffres correspondant sur l'iDBP, on peut penser que la situation y est tout à fait différente.

A travers cette description, nous avons une explication sur les faibles valeurs des rapports (Tps APP. avec index / Tps APP. sans index), pour SOCRATE/CLIO, en comparaison de

ceux obtenus sur l'iDBP.

INSERTION DE 10 TUPLES

Jusqu'à présent nous avons réalisé des insertions de 1 tuple. Autour de cette opération il y a une série de coûts supplémentaires (ouverture et fermeture de fichiers, définition de curseurs,...) qui nous cache le véritable coût d'insertion d'une occurence. Pour minimiser cet effet nous allons générer des programmes qui insèreront chacune 10 tuples. Nous avons dû nous limiter à 10 occurences en raison du LMD de l'iDBP qui ne nous permet pas de réaliser de la génération aléatoire de nombres ou de chaînes de caractères.

| | iDBP | SOCRATE / CLIO | | | | |
|---|---|---|---|---|---|---|
| | | MONO | | | | |
| | | CPU | APP | Nbr Pa 1 | 4 | 5 |
| avec index INETAV10 | 2089 | 85 | 158 | 6 | 10 | 6 |
| ss index | 1583 | 47 | 71 | 3 | 8 | 0 |
| avec index INELAV10 | 2654 | 99 | 165 | 8 | 8 | 6 |
| ss index | 2497 | 68 | 98 | 6 | 8 | 0 |

Tableau 3.13 - insertion de 10 tuples

Avec l'expérience acquise au cours des requêtes précédentes, ces coûts supplémentaires sont très bien ressentis dans les chiffres de l'iDBP (insertion sans index) où l'on multiplie les temps d'insertion par un facteur supérieur à 10. Cela peut se comprendre très facilement où pour insérer 10 éléments (sans index) on effectue 10 fois l'insertion d'un élément (sans index) d'autant plus qu'aucun outil ne réalise une gestion particulière de la mémoire.

Par contre, pour les insertions avec index, les choses ne sont pas aussi évidentes. En effet nous n'avons pas un facteur multiplicateur de 10 pour les deux opérations d'inser-

tion testées. Cela peut se comprendre lorsque l'on sait que le coût d'insertion de 10 éléments, dans un B-arbre, n'est pas égal à 10 fois celui de l'insertion d'un seul élément. Ce B-arbre a besoin d'être constamment équilibré et il arrivera parfois qu'il faille éclater certains niveaux de cette hiérarchie, d'autant plus que toutes les clés insérées au cours de INxx10 avaient le même préfixe et par conséquent ce B-arbre évolue de manière déséquilibrée. La comparaison des chiffres de INxx10 avec index entre l'iDBP et SOCRATE/CLIO est très éloignée et ceci peut s'expliquer en partie par le fait que la mise à jour d'un B-arbre est une opération très couteuse en CPU ainsi qu'en accès disques alors qu'insérer un élément dans un dictionnaire de type SOCRATE/CLIO est beaucoup moins onéreuse en E/S et en temps CPU.

Pour SOCRATE/CLIO les temps apparents ne sont pas aussi étroitement liés au nombre d'occurences insérées que pour l'iDBP. L'explication se trouve en partie dans le nombre de pages accédées pour cette insertion et dans la manière avec laquelle l'insertion s'est faite. On n'a pas demandé à ce que les occurences soient stockées à des rangs précis. Notre base ne contenant pas de "trous" au sein des entités aussi bien pour "ETABLISSEMENT" que pour "ELEVES", toutes les occurences ont été stockées en fin d'entités, elles sont par conséquent contiguës. Ceci explique que pour insérer 9 occurences supplémentaires on ait eu besoin que de 2 pages en sus, pour l'espace "FICH". Pour la mise à jour du dictionnaire nous n'avons qu'un accès page supplémentaire. Cela est dû au fait que les clés insérées ont même racine et surtout au fait que la manipulation du dictionnaire de SOCRATE/CLIO n'est pas aussi complexe, pour une insertion, que celle d'un B-arbre. Selon la définition de la structure, l'insertion dans la liste chainée du dictionnaire a lieu en tête de liste, d'où le faible nombre de pages accédées pour l'espace "DICO" (espace5). Si l'insertion avait eu lieu en fin de liste, l'opération aurait été, d'une part

plus couteuse en accès au dictionnaire et d'autre part, l'espace des cadres aurait été plus vite saturé, ce qui aurait entrainé davantage de lectures et d'écritures pour les autres espaces.

La comparaison des temps d'insertion d'un élève en multi-utilisateurs et de 10 élèves en mono-utilisateur nous donne un aperçu de l'influence de la charge de travail de la machine hôte sur les opérations d'E/S. En effet dans notre cas, il a été plus couteux d'insérer un élément dans un environnement "multi" que d'en insérer 10 si nous avions eu un dorsal (supportant un système de performance égale à celle de SOCRATE/CLIO) totalement dédié à cette dernière opération. C'est sur de tels chiffres que l'on peut prendre conscience des gains potentiels résultant d'une architecture décentralisée.

3.3.7.3 SUPPRESSION

Les suppressions ont été effectuées sur les occurrences créées lors des opérations d'insertion de 1 et de 10 enregistrements. Le coût des suppressions sera plus spécialement opposé au coût de l'insertion correspondante.

DEFINITION DES OPERATIONS DE SUPPRESSION

SUETR : on supprime un ETABLISSEMENT en l'accédeant sur son RANG (clé sur l'iDBP, nº d'ordre sur SOCRATE/CLIO).

SUELR : on supprime un ELEVE en l'accédant sur son RANG.

SUELN : on supprime un ELEVE en l'accédant sur son NOM.

MACSUET10 : on supprime 10 ETABLISSEMENT à l'aide d'un

programme comportant 10 instructions de sélection.

MACSUEL10 : on supprime 10 ELEVE à l'aide d'un programme comportant 10 instructions de sélection.

MACSUET110 : on supprime 10 ETABLISSEMENT à l'aide d'un programme comportant une instruction de sélection multi-critères. La requête ne pourra être exécutée avec utilisation des dictionnaires sur SOCRATE/CLIO.

MACSUEL110 : on supprime 10 ELEVE à l'aide d'un programme comportant une instruction de sélection multi-critères. La requête ne pourra être exécutée avec utilisation des dictionnaires sur SOCRATE/CLIO.

Pour les insertions de 1 ou 10 éléments, sur SOCRATE/CLIO avec index, il y a conservation des temps d'exécution des insertions et des suppressions malgré des accès disques supplémentaires dans le cas de la suppression pour les "ELEVES". Lors de l'insertion d'un élève on n'a pas valorisé ses occurences imbriquées qui, de plus, sont désimbriquées physiquement. Ainsi pour chaque occurence supprimée, on doit accéder les espaces 6 et 9 des entités imbriquées "NOTES" et "CLASSES" afin de parcourir la chaîne de bits d'existence qui nous renseigne sur la valorisation d'occurences associées hiérarchiquement à celle que l'on veut supprimer. Dans notre cas nous n'avons que des accès en lecture sur ces espaces puisque l'on n'a pas d'occurences à supprimer. On peut se rendre compte que la structure n'a pas été optimisée car nous aurions pu laisser la chaîne de bits d'existence dans l'espace contenant l'occurence englobante, ce qui aurait évité les accès à ces nouveaux espaces. C'est ainsi que dans le cas de la suppression de 10 élèves on a 10 accès à l'espace numéro 9 ("PASSE") et 1 à l'espace numéro 6 ("NOTE"). Cet écart est dû uniquement à la taille des occurences désimbriquées physiquement.

| | iDBP | SOCRATE / CLIO | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | MULTI | | | | | MONO | | | | |
| | | CPU | APP | Nbr Pag | | | CPU | APP | Nbr Pa | | |
| | | | | 1 | 4 | 5 | | | 1 | 4 | 5 |
| SUETR avec index | 363 | 45 | 139 | 6 | 11 | 4 | 28 | 88 | 6 | 11 | 4 |
| SUETR ss index | 292 | 27 | 129 | 6 | 6 | 0 | 23 | 65 | 6 | 6 | 0 |
| SUELN avec index | 435 | 29 | 94 | | | | 28 | 90 | | | |
| SUELN ss index | 86569 | 6212 | 7688 | | | | | | | | |
| SUELR avec index | 431 | 39 | 111 | | | | 31 | 102 | | | |
| SUELR ss index | | 20 | 68 | | | | | | | | |

Tableau 3.14 - suppression de 1 enregistrement

| | iDBP | SOCRATE / CLIO | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | MULTI | | | | | | MONO | | | | | |
| | | CPU | APP | Nbre Pages | | | | CPU | APP | Nbre Pages | | | |
| | | | | 1 | 4 | 5 | 6 | 9 | | | 1 | 4 | 5 | 6 | 9 |
| 10 SELECT MACSUET10 index | 2432 | 97 | 207 | 7 | 10 | 6 | - | - | 89 | 172 | 7 | 10 | 6 | - | - |
| 10 SELECT MACSUET10 ss index | | | | | | | | | | | | | | | |
| 10 SELECT MACSUEL10 index | 2777 | 95 | 206 | 9 | 8 | 6 | 1 | 10 | 106 | 197 | 9 | 8 | 6 | 1 | 10 |
| 10 SELECT MACSUEL10 ss index | | | | | | | | | | | | | | | |
| 1 SELECT MACSUET110 index | 943 | | | | | | | | | | | | | | |
| 1 SELECT MACSUET110 ss index | 828 | 120 | 1570 | 7 | 14 | - | - | - | 79 | 112 | 7 | 10 | - | - | - |
| 1 SELECT MACSUEL110 index | | | | | | | | | | | | | | | |
| 1 SELECT MACSUEL110 ss index | 107564 | 36520 | 39695 | | | | | | 36021 | 37504 | | | | | |

Tableau 3.15 - suppression de 10 enregistrements

Ainsi par désimbrication astucieuse de caractéristiques on peut éviter nombre d'accès disque lorsque l'on s'intéresse à un faible nombre de caractéristiques de beaucoup d'occurences. La notion de désimbrication pourra être considérée comme un outil d'optimisation de la gestion d'une base de données.

Dans le cas de non utilisation d'index, les résultats ne sont plus comparables à ceux où l'on fait appel aux index. Ainsi par opposition à l'insertion où l'on pouvait estimer le coût engendré par un ajout dans un dictionnaire ou dans un B-arbre, on ne peut estimer le coût de suppression d'un élément dans un tel index car la recherche séquentielle prend le pas sur la mise à jour de la structure d'indexation. Si nous faisons un parallèle avec le comparatif de De WITT <DEWBITT 83> pour l'insertion et la suppression d'un tuple, il obtient un rapport entre le coût d'exécution de ces 2 opérations qui est de très loin beaucoup plus faible que ce que l'on obtient sur nos deux systèmes. Il est évident que nous parlons d'opérations portant sur des fichiers de très grande taille, les coûts restant comparables pour des fichiers tels que "ETABLISSEMENTS" où l'on a que très peu d'enregistrements.

Dans le cas de la suppression de 10 occurences, il est intéressant de faire ressortir la possibilité offerte par le LMD de l'iDBP de pouvoir définir dans un filtre l'utilisation de plusieurs index. La réalisation de ce filtre se fait par intersection des résultats des 10 parcours du B-arbre. Ceci pourra être considéré comme un "plus" vis à vis d'un système tel que SOCRATE/CLIO où par la nature même du dictionnaire ce genre d'opération est impossible à réaliser. Ceci est d'autant plus intéressant dans la mesure où chacun des critères de sélection est peu discriminant: c'est précisément notre cas.

Nous pouvons renouveler la même remarque que celle faite pour la comparaison des différentes insertions sur SOCRATE/CLIO, à savoir que supprimer 10 tuples n'est pas 10 fois plus couteux que d'en supprimer un. Nous n'en rappellerons pas les raisons. Cette fois-ci la même remarque peut être faite pour les résultats de l'iDBP et cei grâce à cette caractéristique du LMD que nous avons évoquée précédemment.

3.3.7.4 MISE A JOUR

Les opérations de mises à jour ont été réalisées sur les occurrences créées lors des opérations d'insertion. Nous allons effectué des mises à jour de caractéristiques (clé ou non clé), des opérations d'insertion ou de suppression d'occurrences dans des anneaux et des insertions et suppressions d'occurrences imbriquées. Ces deux dernières opérations seront davantage considérées comme des opérations de mise à jour que comme opération de suppression.

INTERPRETATION

MISE A JOUR D'ANNEAUX

DEFINITION DE LA REQUETE:

MJAN insère un ELEVE dans l'anneau "CLASSES".

| | | IDBP | SOCRATE / CLIO | | | | |
|---|---|---|---|---|---|---|---|
| | | | MONO | | | | |
| | | | CPU | APP | Nbr Pag | | |
| | | | | | 1 | 4 | 5 |
| MJAN | | | | | | | |
| | BURNIER0 | 2488 | 23 | 52 | 4 | 6 | 1 |
| | BURNIER1 | 2481 | 24 | 53 | 4 | 6 | 1 |
| | BURNIER2 | 491 | 27 | 53 | 4 | 6 | 1 |
| | BURNIER3 | 488 | 27 | 58 | 4 | 6 | 1 |
| | BURNIER4 | 490 | 22 | 51 | 4 | 7 | 1 |
| | BURNIER5 | 489 | 30 | 60 | 4 | 7 | 2 |
| | BURNIER6 | 489 | 35 | 68 | 4 | 7 | 2 |

Tableau 3.16 - mise à jour d'anneaux

L'opération de mise à jour d'un anneau consiste à introduire un élément dans une liste chainée. Pour les deux systèmes l'insertion se fera en tête. On insère un élève dans l'anneau "CLASSES" et par conséquent on accède à l'élève par son nom et à la classe par son rang vu que cette dernière n'a pas de clé discriminante. Ce fait ne sera pas un avantage pour l'IDBP où le rang d'une occurence n'est rien d'autre qu'une clé au même titre que n'importe quelle clé. Par contre, pour SOCRATE/CLIO cela permet de faire un accès

quasi direct et on limite du même coup le nombre d'accès au dictionnaire.

Pour SOCRATE/CLIO nous avons systématiquement deux écritures sur l'espace "FICH", une pour la mise à jour du pointeur fils et une pour celle des pointeurs père et frère de l'élève. Etant donné que cette opération d'insertion n'est pas à la charge du programmeur, l'opération de mise à jour est optimisée vis à vis de la programmation que l'on est tenu de réaliser sur l'iDBP. Ces deux facteurs expliquent le rapport de 9 que l'on obtient entre l'iDBP et SOCRATE/CLIO. Les résultats restent constants pour les deux systèmes ainsi que le nombre de pages accédées pour SOCRATE/CLIO. Il faut relever que pour accéder à une occurrence sur 18000 enregistrements on accéde à une page du dictionnaire alors que l'iDBP est tenu de parcourir son B-arbre jusqu'au niveau des feuilles. Pour une seule valeur de clé on est tenu d'accéder à deux pages du dictionnaire, ceci étant dû au hash-code. Les résulats restent constants car l'insertion dans l'anneau est réalisée systématiquement en début de liste.

SUPPRESSION D'OCCURRENCES DANS UN ANNEAU

DEFINITION DE LA REQUETE:

SUPAN supprime un ELEVE dans l'anneau CLASSES. Seul le chainage est mis à jour et l'occurrence ELEVE n'est pas détruite physiquement.

| | iDBP | SOCRATE / CLIO MULTI | | | | | SOCRATE / CLIO MONO | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | CPU | APP | Nbr Pag 1 | 4 | 5 | CPU | APP | Nbr Pag 1 | 4 | 5 |
| SUPAN | | | | | | | | | | | |
| BURNIER0 | 1848 | 26 | 66 | 6 | 5 | 0 | 26 | 51 | 6 | 5 | 0 |
| BURNIER1 | 1111 | 24 | 67 | 6 | 5 | 0 | 26 | 51 | 6 | 5 | 0 |
| BURNIER2 | 1743 | 26 | 66 | 6 | 5 | 0 | 29 | 48 | 6 | 5 | 0 |
| BURNIER3 | 1074 | 27 | 58 | 6 | 6 | 0 | 27 | 53 | 6 | 6 | 0 |
| BURNIER4 | 1028 | 27 | 60 | 6 | 5 | 0 | 25 | 50 | 6 | 5 | 0 |
| BURNIER5 | 1012 | 27 | 61 | 6 | 5 | 0 | 25 | 51 | 6 | 5 | 0 |
| BURNIER8 | 862 | | | | | | | | | | |
| SUPANE | | | | | | | | | | | |
| BURNIER0 | | 27 | 61 | 6 | 5 | 1 | 25 | 51 | 6 | 5 | 1 |
| BURNIER1 | | 22 | 53 | 6 | 4 | 1 | 22 | 47 | 6 | 4 | 1 |
| BURNIER2 | | 24 | 64 | 6 | 4 | 1 | 24 | 52 | 6 | 4 | 1 |
| BURNIER3 | | 27 | 60 | 6 | 4 | 1 | 23 | 34 | 6 | 4 | 1 |
| BURNIER4 | | 27 | 54 | 6 | 4 | 1 | | | | | |

Tableau 3.17 - suppression d'occurrence dans un anneau

Pour la suppression d'une occurrence dans un anneau les résultats de l'iDBP sont fortement liés à la place de l'occurrence dans la liste alors que ceux de SOCRATE/CLIO sont totalement indépendants. Ceci s'explique très facilement par la façon dont est réalisée cette suppression sur chacun de ces deux systèmes. Pour l'iDBP on est tenu d'accéder à l'élève en parcourant l'anneau depuis sa tête ("CLASSES") car cette liste n'a pas été générée avec chainage double et par conséquent c'est le seul moyen d'atteindre son précédent. Par contre chez SOCRATE/CLIO on peut soit demander la mise à jour de la caractéristique "référence" de l'"ELEVE" sans mentionner la classe soit demander à supprimer tel élève dans un anneau dont on précise la tête. La façon dont est réalisée la mise à jour est totalement transparente à l'utilisateur puisque ces deux formes, bien que différentes au premier abord, génère quasiment le même code. Ces deux formes de mise à jour consomment le même temps apparent. En effet on se positionne sur la tête de chaîne et on parcourt ensuite l'anneau. Ce code est optimisé vis à vis d'un parcours programmé par l'utilisateur tel que cela se fait sur l'iDBP. Le nombre de pages accédées en lecture sur l'espace "FICH" reste constant pour

chacun des éléments à supprimer car d'une part les éléments de notre anneau sont contigus et d'autre part la taille de ceux-ci est petite devant la taille des pages. Par conséquent il n'est pas surprenant que les coûts demeurent constants. Si la taille de l'anneau avait été beaucoup plus conséquente et les enregistrements le composant plus dispersés et que l'on supprime des éléments pris au hasard, les temps apparents auraient été étroitement liés à la position de l'enregistrement dans la liste. Il faut garder à l'esprit que pour une telle opération le système SOCRATE/CLIO n'est pas tenu d'accéder entièrement aux enregistrements à modifier. Seules les caractéristiques référencées seront accédées.

## SUPPRESSION D'OCCURENCES IMBRIQUEES

DEFINITION DE LA REQUETE:

SUPIMB supprime physiquement les occurrences imbriquées COURS d'une CLASSES

| | iDBP | SOCRATE / CLIO | | | | |
|---|---|---|---|---|---|---|
| | | CPU | APP | Nbre Pages 1 | 4 | 8 |
| SUPIMB 18 entités | 4828 | 64 | 149 | 7 | 9 | 21 |

Tableau 3.18 - suppression d'occurrences imbriquées

L'opération de suppression d'occurrences imbriquées détruit physiquement chacun de ces enregistrements. Pour l'iDBP, celles-ci sont accédées par chainage et ne sont pas nécessairement contiguës. Pour SOCRATE/CLIO ces données sont stockées dans un espace autre que celui contenant l'occurrence englobante et se trouvent côte à côte (aux "squatters" près). Cet espace est occupé à 77,51 % et le nombre moyen d'accès est 1,24. Ceci explique le nombre de pages accédées qui est somme toute conséquent pour supprimer 18 petites occurrences. Cette fois-ci l'ensemble de l'occur-

rence doit être accédée car on la supprime dans son intégralité.

Rappelons que l'opération de parcours d'occurences imbriquées n'était pas favorable à l'iDBP. La suppression de tels éléments est de nouveau beaucoup plus couteuse avec l'iDBP (avec un rapport de 32). De plus notre suppression porte sur peu d'occurences qui, d'après la logique de chargement de la base de l'iDBP, sont rapprochées physiquement. Mais si l'on avait considéré des occurences valuées tout au long de la vie de la base, la localisation de celles-ci aurait été strictement identique sur SOCRATE/CLIO mais la dispersion aurait été beaucoup plus forte sur le dorsal et par conséquent leur coût de suppression, fortement accru.

Ainsi les machines bases de données qui se disent être multi-modèles arrivent finalement à être fortement pénalisées pour les opérations portant sur le modèle pour lequel elles n'ont pas été conçues à l'origine. Bien souvent elles étaient orientées à leur origine vers le modèle relationnel que l'on a quelque peu revu leur LDD et LMD afin de pouvoir simuler un modèle réseau.

MISES A JOUR DE CARACTERISTIQUES

DEFINITION DES REQUETES

MJELN met à jour une caractéristique non clé de l'entité ELEVE que l'on accède sur sa clé. Cette caractéristique est le NOM de l'ELEVE mais nous avons détruit le dictionnaire associé à celle-ci.

MJELN10 reprend MJELN mais sur 10 occurrences.

MJELNK est comparable à MJELN mais la caractéristique modifiée est le nom de l'élève, le nom étant une clé non discriminante. Le dictionnaire a été reconstruit par rapport à MJELN.

| | | iDBP | SOCRATE / CLIO MONO CPU | SOCRATE / CLIO MONO APP | SOCRATE / CLIO MONO Nbr Pag 1 | 4 | 5 |
|---|---|---|---|---|---|---|---|
| MJELN | BURNIER0 | 212 | 22 | 52 | 6 | 3 | 1 |
| | BURNIER9 | 215 | | | | | |
| | BURNIER5 | 256 | | | | | |
| MJELN10 | BURNIERx | 1806 | 44 | 77 | 7 | 5 | 2 |
| | BURNIER0 | 360 | 33 | 77 | 6 | 5 | 8 |
| | BURNIER10 | 325 | 32 | 86 | 6 | 5 | 8 |
| MJELNK | BURx-- GUIG | 367 | 54 | 1[illegible] | 6 | 4 | 5 |

Tableau 3.19 - mise à jour de caractéristiques

Nous avons évalué deux types de mise à jour de caractéristiques:

- une pour une caractéristique non clé

- une pour une caractéristique clé.

Pour chacune de ces deux situations nous avons effectué l'opération sur une puis sur dix occurences, celles insérées au cours de l'opération de création. Il est vrai qu'elles sont stockées de façon contiguë mais la situation est identique pour les deux systèmes. L'opération effective de mise à jour ne doit pas être plus couteuse d'un système à l'autre, seuls les temps d'accès disque pouvant influer. C'est très certainement la phase de recherche qui a fait la différence en faveur de SOCRATE/CLIO.

La chose est plus évidente dans le cas de 10 mises à jour où, à la part d'"overhead" près, les temps sont multipliés par dix sur l'iDBP alors que pour SOCRATE/CLIO on ne double même pas les temps apparents. Une fois de plus l'explication de cet écart réside dans le nombre d'accès en lecture sur les espaces FICH et DICO. La quasi totalité des requê-

tes testées a profité de cette gestion des cadres.

Pour la mise à jour de la caractéristique clé, nous avons testé le remplacement d'une clé par une clé de même racine et ensuite par une clé totalement différente. Finalement l'effet a été peu influant sur les temps apparents pour les deux systèmes mais on peut relever malgré tout le nombre important de pages du dictionnaire accédées pour ce genre d'opération.

### 3.3.7.5 CONCLUSIONS SUR LES OPERATIONS DE MISES A JOUR

Cette conclusion portera davantage sur la manière avec laquelle a été réalisée l'évaluation des opérations de mise à jour que sur les résultats eux mêmes.

Les requêtes exécutées en mono-utilisateur nous ont permis de mettre en évidence les coûts de mise à jour des dictionnaires pour chacun des deux systèmes. Dans un environnement multi-utilisateurs sur la base, cet aspect aurait été masqué par plusieurs autres coûts supplémentaires sans rapport avec celui de la mise à jour de la méthode d'indexation. Nous n'avons pu réaliser des estimations aussi fines pour l'iDBP que pour SOCRATE/CLIO en raison des renseignements relatifs aux accès pages que nous ne pouvons relever sur ce système. Ces données auraient été très intéressantes à analyser d'autant plus que la méthode des B-arbres comporte des mises à jour très variables selon le degré de remplissage de chacun des noeuds. Nous aurions pu également repérer les opérations d'éclatement du B-arbre et par conséquent évaluer ce type de manipulation qui est reconnu pour être l'opération la plus couteuse pour cette forme de gestion d'index. De là, nous aurions pu jouer sur le nombre de mises à jour à effectuer au cours d'une requête et sur le nombre de niveau du B-arbre.

En dehors de cet aspect mono/multi utilisateurs, nous avons

pu jouer sur le nombre de mises à jour exécutées au cours de chaque requête, sur la dispersion des clés des éléments mis à jour et sur leur localisation physique dans les fichiers iDBP et les espaces SOCRATE/CLIO.

Pour l'opération d'insertion nous avons volontairement laissé de côté l'insertion sur le rang telle qu'elle nous est proposée dans SOCRATE/CLIO afin de ne pas nous placer dans une position trop avantageuse pour ce système, bien que ce genre d'opération pourra être fréquemment utilisée.

De plus, les insertions que nous avons réalisées se sont effectuées de manière contiguë sur les espaces physique étant donné qu'aucun enregistrement n'avait été supprimé sur l'iDBP et que toutes les occurences avaient été stockées avec des numéros d'ordre consécutifs sur SOCRATE/CLIO. Ainsi, toutes les opérations de mise à jour réalisées sur ces occurences nouvellement crées ont nécessité peu de mouvements de pages dans SOCRATE/CLIO.

Quant à l'analyse proprement dite des temps d'exécution des opérations de mise à jour, les résultats sont du même ordre que ceux obtenus jusqu'à présent, à savoir que l'iDBP ne rivalise pas avec les performances de SOCRATE/CLIO. Ce phénomène n'est pas uniquement dû à la gestion des zones des cadres de SOCRATE/CLIO (cf MJELN10 avec les occurences dispersées) et à la vitesse d'accès disque du VAX (les opérations de mise à jour sur SOCRATE/CLIO ont été réalisées sur un disque plus lent que celui utilisé pour les autres requêtes).

Cette grande diversité dans les résultats est, pour une grande part, le fruit de la méthode d'indexation retenue sur l'iDBP, à savoir les B-arbres. Si ceux-ci sont intéressants pour exécuter des filtres sur plusieurs attributs clés (à manipuler avec précaution, cf Q15, Q16), nous payons très cher le coût de mise à jour. Les B-arbres ne sont pas seuls à être mis en cause car une requête telle que MJELAN qui ne modifie pas les index, reste toujours beaucoup plus couteuse sur l'iDBP.

Quoiqu'il en soit, à aucun moment nous n'avons obtenu de résultats comparables. Ceci laisse fortement pressentir que malgré un environnement multi-utilisateurs sur la machine hôte, la décentralisation des fonctions d'accès à la base de données sur une MBD telle que celle étudiée, pénalisera considérablement les usagers de la base. Seuls les utilisateurs ne travaillant pas sur la base bénéficieront de cette décentralisation puisqu'ils se verront attribuer davantage de ressources.

## 3.4 SYNTHESE DES RESULTATS

### 3.4.1 CONCLUSIONS SUR L'APPORT D'UNE MACHINE BASES DE DONNEES

Notre expérimentation s'est voulue être une étude sur l'opportunité d'utilisation d'une machine bases de données en vue d'assister un calculateur hôte pour toutes les opérations d'accès à l'information. Les conclusions que nous allons tirer de ce comparatif sont, dans une première étape, associées à la nature de l'application retenue, en l'occurrence une application de gestion.

La conclusion principale de notre travail est que, pour l'application testée, l'apport d'une machine bases de données telle que l'iDBP n'est pas primordial.

Nous nous sommes attachés à évaluer l'ensemble des opérateurs d'accès à la base fournis par les deux SGBD. Pour aucun d'entre eux, nous n'avons obtenu une amélioration des temps d'accès avec l'utilisation de l'iDBP. Au contraire, d'une manière globale, les temps d'exécution relevés sur la configuration traditionnelle étaient bien inférieurs à ceux de l'iDBP.

Par contre, il est évident qu'en déchargeant toutes les fonctions d'accès aux bases de données sur un second site, on allègera de façon manifeste le calculateur hôte. Par conséquent, tous les utilisateurs, non concernés par ces types d'accès, se verront attribuer davantage de ressources CPU et bénéficieront de l'allègement des canaux d'entrées sorties. Mais il est bon de rappeler que ce nouveau type d'architecture s'adresse en priorité aux utilisateurs des bases de données et que leurs besoins et leurs exigences constituent la priorité de nos recherches.

Les mesures ont été relevées sur une application de gestion alors que l'on s'était intéressé, au cours du Chapître 1, aux problèmes de la gestion de l'information au sein de nouvelles applications. Si l'on avait retenu une application telle que celle de la C.A.O. de V.L.S.I., les résultats obtenus n'auraient fait que confirmer nos conclusions. Dans de telles applications, l'utilisateur est amené à parcourir fréquemment les structures hiérarchiques des éléments de la base. On aurait été, alors, amené à évaluer des transactions encore plus complexes que la requête Q5 (cf § 3.3.5 Opération de parcours d'anneau) où les rapports, en faveur de SOCRATE/CLIO, entre les temps d'exécution étaient supérieurs à 20. Ainsi, il ne semble pas opportun de substituer à un système conventionnel tel que SOCRATE/CLIO, une architecture décentralisée intégrant l'iDBP pour les applications de type C.A.O.. La tendance s'orientera très certainement vers la spécification d'une architecture de plus grande envergure que celle expérimentée. Nous exposerons au cours du Chapître 4, notre vision d'une nouvelle architecture de SGBD consacrée aux applications de C.A.O..

Avant de spécifier une nouvelle proposition, il serait bon de mettre en valeur les points sur lesquels la machine base de donées iDBP présente certaines insuffisances, contraintes ou limitations et qui nous ont conduits à tirer de telles conclusions.

### 3.4.2 iDBP : MACHINE RELATIONNELLE

L'iDBP est présentée comme une MBD relationnelle dont le modèle de données a été quelque peu étendu afin de pouvoir gérer des bases hiérarchiques ou réseaux. Ceci implique qu'elle a été conçue pour répondre en priorité aux exigences du modèle relationnel qui, il est vrai, est le modèle présentant le plus de limitations au niveau des vitesses d'accès aux données.

Il est bon de faire remarquer que les aspects de performances intéressent tout autant les systèmes bâtis sur d'autres modèles que celui de CODD. Les modèles réseaux ou hiérarchiques ne sont pas la panacée quant à ces problèmes.

Le problème est donc global aux trois modèles mais la solution n'est très certainement pas unique. La meilleure façon de s'en rendre compte est d'analyser, sommairement, les types d'accès que l'on réalise.

Les primitives des LMD du modèle relationnel réalisent, en dehors des accès sur clé, uniquement des balayages séquentiels de relations. Alors que celles des modèles hiérarchiques ou réseaux exploitent les différents types de chainage entre occurrences, tout en effectuant également des parcours systématiques d'entités.

Par conséquent, et là nous dépassons quelque peu les possibilités de l'iDBP, les machines bases de données disponibles abordent plus spécialement l'aspect "accélération d'accès aux données" à travers la solution des accélérateurs disques réalisant du filtrage au vol. Il s'ensuit que lorsque l'on interface ce type de matériel à des modèles de données tels que celui de SOCRATE/CLIO, on juge ce genre d'architecture inadéquate.

Il semblerait donc qu'il faille aborder ces problèmes dans des contextes bien définis et développer, à chaque fois, des solutions précises pour une catégorie donnée d'utilisateurs. L'issue ne se trouve certainement pas dans la " boîte noire universelle ".

### 3.4.3 PARTICULARITES DE L'iDBP

En dehors de l'architecture générale de l'iDBP, nous allons développer certains choix qui ont été faits au niveau de

la réalisation du SGBD et qui n'ont pas été sans conséquences sur les résultats obtenus.

### 3.4.3.1 METHODE D'INDEXATION

Les coûts supplémentaires de la méthode d'indexation de l'iDBP, les B-arbres, vis à vis du hash-code sont bien connus. Ils ont été mis en évidence à plusieurs reprises, notamment pour les opérations d'insertion, de suppression et de mises à jour.

Par contre, il est vrai que pour des requêtes de sélection comportant plusieurs critères de recherche définis comme clé, peu discriminantes, nous aurions pu mettre en exergue l'avantage de la solution des B-arbres. Cet avantage n'a pas été évalué au cours de l'étude mais peut être mis au crédit de l'iDBP.

### 3.4.3.2. GESTION DE LA MEMOIRE

Nous avons pu chiffrer l'apport bénéfique de la gestion des cadres en mémoire centrale retenue dans SOCRATE/CLIO vis à vis d'un système traditionnel tel que celui de l'iDBP. Lorsque l'utilisateur ne travaille qu'en consultation, il est intéressant de pouvoir conserver en mémoire les pages récemment utilisées qui demeurent susceptibles d'être redemandées.

Cette gestion peut paraitre comme une première solution aux problèmes inhérents aux coûts des entrées/sorties. Ce facteur de qualité se retrouve très bien dans les résultats des requêtes où l'on effectue des doubles parcours. Sur l'iDBP on double les temps d'exécution alors que ceux-ci sont pratiquement conservés sur SOCRATE/CLIO. Malgré tout, gardons bien à l'esprit que l'efficacité de cette méthode est maximale lorsque les utilisateurs sont déclarés avec des droits de consultation sur les différents espaces phy-

siques.

La solution évoquée précédemment se rapproche beaucoup de la notion de mémoire cache et d'ante-mémoire telle qu'elle est utilisée dans les accélérateurs disques de type DIRAM 32 <COPER.2>. La solution retenue pour ces périphériques va encore plus loin puisqu'au moment du montage du disque, un certain nombre de pages, les plus fréquemment utilisées, sont chargées dans la mémoire cache du périphérique.

DE WITT avait également abordé ce problème de mémoire cache avec le prototype de DIRECT <DIRECT>. Les processeurs de traitement demandaient le transfert de pages de relations de la mémoire secondaire vers des CCD. Ces pages étaient alors disponibles pour tous les processeurs et la même page ne pouvait se retrouver sur plusieurs CCD. Les pages étaient alors verrouillées lors des mises à jour et une fois libérées, les processeurs disposaient de la dernière version. On réduisait ainsi à la fois le nombre des lectures mais aussi celui des écritures. La taille des CCD a dû être judicieusement calculée afin d'aboutir à un équilibre entre les charges de travail des processeurs et des unités d'échange.

### 3.4.3.3 MISE EN OEUVRE DES OPERATEURS DE L'ALGEBRE RELATIONNELLE

Le LMD de l'iDBP a une vision très particulière quant à la mise en oeuvre des opérateurs de l'algèbre relationnelle. Le fait de ne pas pouvoir disposer de ces opérateurs autrement qu'à travers la définition de vues aboutit à une programmation très peu aisée qui ne sera pas sans conséquences sur les résultats.

De plus dès que l'on fait référence à une vue, le système construit, à partir de la définition de la vue, une liste d'identificateurs d'enregistrements, les "pointeurs-id" des

tuples, laquelle liste est parcourue dans une seconde étape pour récupérer les enregistrements de la vue. C'est ainsi que pour une opération de sélection on obtient des temps d'exécution étroitement liés au nombre de tuples retournés, cette association n'étant pas due uniquement à l'affichage des enregistrements de la vue.

Cette approche se trouve être opposée à la tendance actuelle qui s'oriente vers un filtrage au vol afin précisément de réduire la masse d'informations à transférer sur les canaux d'E/S.

Cette mise en oeuvre paraissant très particulière au premier abord, peut avoir ses raisons d'être dans un contexte de travail différent de celui dans lequel l'étude a été réalisée.

### 3.4.4 PERSPECTIVES POUR SOCRATE/CLIO

Rappelons que l'expérimentation a été réalisée notamment dans le but d'évaluer une solution d'accélération d'accès aux données pour le système SOCRATE/CLIO. Nos conclusions vont nous inciter à investir dans d'autres axes de recherches que les accélérateurs de type iDBP.

Tous les accès directs réalisés à travers les parcours d'anneaux et d'inverse SOCRATE ne pourront être accélérés, par la mise en oeuvre d'un filtrage au vol.

Pour ce type de primitive, il ne serait pas inintéressant d'investir davantage dans les notions de mémoire cache, de rapprochement physique des données et de parallélisme.

L'intérêt des mémoires caches a été abordé précédemment et celui du rapprochement physique, évalué avec les requêtes de parcours d'entités imbriquées. Quant à l'aspect paralléli sme intra et inter requêtes, les contraintes techniques

dominent encore trop le sujet.

## 3.5 METHODOLOGIE SUR LES COMPARATIFS DE S G B D

Au cours de cette partie nous voudrions faire quelques remarques sur la façon de mener à bout une opération telle que la comparaison de systèmes de gestion de bases de données sur le plan des vitesses d'exécution des transactions.

### 3.5.1 LA BASE DE TEST

Le contenu du test a été fortement influencé par les éléments dont on disposait avant sa réalisation et le temps qui nous était imparti, à savoir que l'on disposait du matériel pour une durée de quatre mois. Toutes les mesures ont été effectuées sur une base de données simulant une gestion scolaire. Le premier point positif est que cette base avait une taille de 12 Mo, ce qui pour une base de test est fort appréciable et somme toute, assez rare.

De plus on avait beaucoup misé sur le fait que l'on possédait un jeu de requêtes, ainsi que de leurs temps d'exécution, qui avait été élaboré pour comparer les différentes versions de SOCRATE/CLIO. Ces transactions sont adéquates pour évaluer différentes alternatives du même produit mais se sont révélées très vite peu significatives quant à notre comparatif.

Nous avons conçu un nouveau jeu de requêtes que nous avons basé sur les opérateurs dont on disposait aussi bien sur SOCRATE/CLIO que sur l'iDBP. Pour nous il était important de décefer les opérations qui étaient plus couteuses sur tel ou tel système en évitant de combiner plusieurs opéra-

teurs dans la même transaction.

Afin d'enrichir notre analyse, il était important de faire varier les paramètres des requêtes pour suivre l'évolution des temps en fonction du nombre d'accès disques ou d'occurences retournées. Et c'est précisément là que nous avons pu constater l'inadéquation de la population de notre base. En effet, nous n'avions aucune possibilité de controler la cardinalité des résultats. Pour une opération de sélection nous ne pouvions connaître les critères de recherche qui nous auraient permis d'obtenir en retour, par exemple, 10 %, 20 % ou 50 % de la base. Certains résultats et analyses ont souffert de ce manque de souplesse. C'est pourquoi il aurait été opportun de concevoir sa propre base tel que cela a été fait dans <DEWBITT 83>.

Dans une véritable base de test la génération des valeurs des caractéristiques n'est pas totalement aléatoire. La distribution des valeurs d'un attribut est définie en lui affectant une plage de valeurs, lesquelles valeurs étant instanciées le même nombre de fois. De plus pour des opérations aussi complexes que la jointure, il aurait été intéressant de pouvoir travailler sur des relations de taille variable afin de comparer les mises en oeuvre de cet opérateur.

Par contre, dans une étude tel que le comparatif de DEWITT <DEWBITT 83>, nous pourrions regretter que les clés soient des valeurs entières dont la plage de valeurs va de 1 à la cardinalité de la relation. Des clés alphanumériques générées cette fois-ci de manière totalement aléatoire aurait très certainement mis en évidence de façon plus marquante les différences entre les méthodes d'indexation retenues pour chacun des systèmes.

Il aurait été intéressant de pouvoir disposer, au sein de la même relation, de plusieurs caractéristiques discrimi-

nantes déclarées comme clé. Nous aurions pu mettre davantage en relief les avantages de la recherche multi-critères avec clés telle qu'elle est réalisée sur l'iDBP et que l'on ne peut mettre en oeuvre sur SOCRATE/CLIO.

Il est bon de faire remarquer que pour un tel comparatif la spécification de la base n'est pas à dissocier de la définition des transactions. La base devrait être peuplée et même structurée en fonction du type de requêtes que l'on veut exécuter.

### 3.5.2 LA CONFIGURATION MATERIELLE

Notre comparatif portait sur deux systèmes, chacun disponible sur un matériel différent et pour cause... Il est bon de faire remarquer que la machine sur laquelle était exécutée SOCRATE/CLIO était la même qui servait de calculateur hôte pour l'iDBP. Ainsi nous étions dans la meilleure situation pour estimer ce que pouvait apporter une machine bases de données à un système tel que SOCRATE/CLIO puisque nous pouvions nous assimiler à un utilisateur final, détenteur d'un VAX 750 et à qui on proposerait soit SOCRATE/CLIO soit l'iDBP connecté à son calculateur.

Il est vrai que si notre étude avait porté sur l'évaluation de plusieurs S.G.B.D., il y aurait eu peu de chance pour que toutes les mesures aient été relevées sur le même matériel. La taille de la mémoire et celle des pages, le type de disques utilisés sont autant de facteurs influant de façon marquante les temps d'accès aux données. De là, il est très difficile de pondérer les résultats selon la configuration utilisée.

### 3.5.3. L'ENVIRONNEMENT UTILISATEUR

Notre étude a été réalisée uniquement dans un environnement mono-utilisateur à l'égard de la base bien que les deux systèmes soient prévus pour évoluer en multi-utilisateurs. Pour de nombreuses requêtes nous avions tout intérêt à travailler avec un seul usager en accès sur la base. Par contre il aurait été intéressant de pouvoir évaluer ce que pouvait apporter, à un utilisateur ne travaillant pas sur la base, le déchargement sur un dorsal de toutes les demandes d'E/S d'un groupe d'usagers de la B.D.. Une machine tel que l'iDBP est sensée apporter un "plus" à la fois aux personnes qui accèdent les B.D. déportées sur le dorsal mais aussi aux autres utilisateurs qui profiteraient d'un allégement de la charge de travail du calculateur hôte.

Jusqu'à présent une seule expérience d'évaluation dans un environnement multi-utilisateur a été tentée par l'équipe de DEWITT <DEWBOR 84>. La stratégie à adopter sera différente, de celle retenue pour un test en mono-utilisateur, car apparaissent plusieurs facteurs intervenant directement sur les performances dans un environnement multi-utilisateurs:

- le degré de multi-programmation c'est à dire le nombre de transactions exécutées simultanément

- le partage des données: ce facteur fait référence à une gestion sophistiquée de la mémoire et aux différentes méthodes de verrouillage des données

- la diversité des requêtes: on n'exécutera pas au même instant n fois la même requête d'où la nécessité d'affiner le choix des requêtes à lancer simultanément. Ce découpage pourra se faire en classant les requêtes sélectionnées selon le taux d'utilisation de CPU et d'utilisation des disques <DEWBOR 84>.

### 3.5.4. LE CONTENU DES MESURES

Les outils offerts par SOCRATE/CLIO sont très largement suffisants pour effectuer une étude d'évaluation de performances. On peut récupérer le nombre de pages demandées et effectivement transférées, les temps CPU et apparents des exécutions des transactions. Par contre pour l'iDBP nous ne disposions d'aucun outil de mesure et nous nous sommes contentés des temps d'exécution apparents relevés depuis le VAX. Ainsi nous n'avons pu évaluer avec précision les accès disques et les temps réels d'exécution sur le dorsal. Il était donc difficile de conaître l'origine précise de certains résultats.

# CHAPITRE 4

# PROPOSITION POUR UNE NOUVELLE ARCHITECTURE DE S.G.B.D. ADAPTEE AUX EXIGENCES DES NOUVELLES APPLICATIONS

## 4.1. POSITION A L'EGARD DES MACHINES BASES DE DONNEES

### 4.1.1 DESCRIPTION DE LA SITUATION

Notre position à l'égard de l'utilisation de machines bases de données telles que l'on en dispose actuellement sur le marché est clairement définie à travers la synthèse des résultats du chapitre 3. Tant que l'on ne fera pas appel à du matériel spécialisé, à des architectures multi-processeurs ou encore à des machines développées pour des modèles de données précis, rien ne peut laisser entrevoir que l'on obtienne des améliorations vis à vis de systèmes développés et optimisés depuis de nombreuses années (comme SOCRATE/CLIO), et exécutés sur des mini-ordinateurs.

Jusqu'à présent, l'accélérateur disque peut être considéré comme le seul outil en mesure d'offrir une optimisation des accès à la base. Ces machines ne sont pas des M.B.D. mais constituent une étape entre les disques traditionnels et la

machine base de données. Il est vrai que son utilisation est réduite à l'exécution d'opérations simples telle que la sélection. Pour un modèle tel que le modèle relationnel posséder un opérateur de sélection mis en oeuvre sur une base de recherche associative est un atout non négligeable pour l'exécution de la jointure, extrèmement couteuse en E/S. En associant à cette recherche associative une mémoire cache, on réduit encore le taux de transferts mais cette fois-ci entre le disque et le filtre. Ainsi on arrive à réduire notablement la duplication des échanges de la même masse d'informations. L'intérêt le plus marquant d'une telle solution peut être relevé, de nouveau, pour l'exécution de la jointure où l'on stockerait l'opérande interne dans l'ante-mémoire, dans la limite de la taille de celle-ci. Cet accélérateur est la première solution matérielle véritablement opérationnelle.

Nous n'avons cessé de faire remarquer jusqu'à présent, que les solutions aux différents problèmes mis en évidence au cours de cette thèse ne pourront, à chaque fois, être résolus par un outil unique:

- le modèle de données généralisées pouvant décrire et manipuler aussi bien des documents, du son ou des images et des données de C.A.O. n'existe pas encore

- la machine base de données qui permettra d'accélérer les accès à l'information et ceci quelque soit le modèle de données, n'a pas encore été spécifié.

En conséquence, nous allons restés logiques avec nos précédentes conclusions. Et plutôt que définir une nouvelle architecture de SGBD répondant d'emblée à tous les besoins des nouvelles applications, nous allons définir une configuration qui tendra de satisfaire en priorité les besoins de la C.A.O..

Par contre, nous pouvons nous permettre de parler de C.A.O. au sens général, car il a été mis en évidence que les principes du processus de conception et les différentes classes d'informations manipulées étaient conservées d'un domaine d'activité à l'autre. La suite de ce chapître fera donc référence aux applications C.A.O..

Nous gardons toujours à l'esprit le côté performance d'accès qui a guidé l'ensemble de cette thèse. Pour nous, cet aspect pourra être partiellement résolu par la spécification d'une architecture "éclatée" où l'on chercherait à spécialiser chacun des sous-systèmes.

Avant de décrire notre proposition, nous allons reprendre les outils offerts par l'iDBP qui nous paraissent pouvoir faire partie de cette nouvelle architecture.

## 4.1.2 OUTILS NOUVEAUX OFFERTS PAR L'iDBP.

Il faut noter que le non classicisme du SGBD implanté sur l'iDBP n'est pas le fruit du hasard mais une première proposition de réponse aux exigences des utilisateurs des nouvelles applications. On peut dire que la nature profonde de ce système n'est pas novatrice puisque fortement inspirée du modèle relationnel. Par contre un premier pas a été fait le premier pas vers le développement d'un outil tout spécialement orienté utilisateur grâce à ses nouvelles possibilités, les deux les plus marquantes étant la gestion de fichiers non structurés associés à des fichiers structurés et la réapparition des pointeurs.

### 4.1.2.1 FICHIERS STRUCTURES - FICHIERS NON STRUCTURES

Il est possible de définir des fichiers non structurés, pouvant être dans un premier temps des documents, et de les associer à des fichiers structurés à l'aide de pointeurs.

Cette proposition converge parfaitement vers notre analyse des besoins d'applications telles que le traitement de la parole ou encore l'analyse d'images. Pour ces types d'applications il était indispensable de pouvoir établir à tout instant une association entre des éléments tels que du son ou une image et leur "fiche descriptive" bâtie sur une structure figée.

La mise en route d'une application, en grandeur nature, manipulant du son et/ou de l'image ne pourra être effectuée de manière classique. Il ne semble pas raisonable de vouloir stocker sur les mêmes unités à la fois ces éléments non structurés et leur description. Il faut savoir que le son ou les images constituent une nature d'objets qui seront, dans la plupart des situations, non modifiés. Par conséquent il serait opportun de profiter de ces nouveaux périphériques que sont les disques optiques numériques afin:

- de ne pas surcharger les disques conventionnels
- de ne pas disperser inutilement l'information structrée.

Ainsi si l'on arrive à dissocier le stockage d'une certaine catégorie d'éléments, rien ne nous empèchera de dissocier leurs accès et de profiter du parallélisme inhérent à cette distribution.

Cette dissociation entre éléments est basée sur leur nature, structurée ou non structurée. Elle permettra de distinguer les opérations à effectuer selon le type de l'élément:

- opération de sélection, de O-produit sur des éléments structurés
- accès direct sur des enregistrements non structurés.

On arrive ainsi à dégager un degré de spécialisation des accès très marqué qui pourra favoriser leur optimisation. Cette situation implique de dissocier les accès en les déportant soit directement sur l'unité de stockage soit en placant un dorsal entre le périphérique et le calculateur hôte, ce dernier se contentant de répartir les tâches selon les requêtes des utilisateurs et les réponses restituées par les dorsaux.

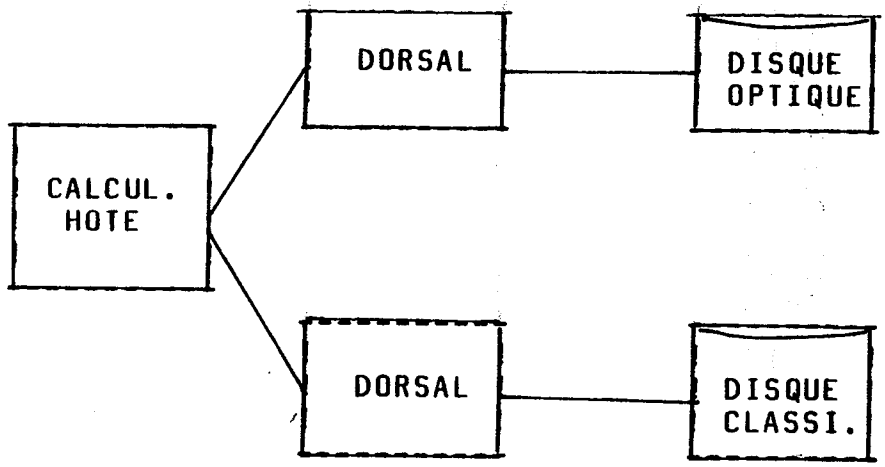



Figure 4.1 - Spécialisation des unités de stockage

4.1.2.2. RETOUR DE LA NOTION DE POINTEURS

La seconde particularité relevée parmi les outils offerts par l'iDBP repose sur le retour de la notion de pointeur. Il est bon de faire remarquer que ceux-ci ont été réintroduits non pas dans le but unique de pouvoir simuler les modèles réseaux ou hiérarchiques mais aussi d'apporter une forme de solution aux remarques que l'on pourrait formuler à l'égard du modèle relationnel quant au coût de certains de ses opérateurs. En introduisant cette notion, on supprime d'emblée la particularité du modèle relationnel. Si l'on veut rester dans le cadre d'un LMD non procédural, donnons à l'utilisateur l'impression de travailler avec un tel langage et développons au niveau de son interprétation les modules permettant de tirer profit des pointeurs existants. Un tel outil n'appartient pas au domaine de l'utopie puique

l'on sent très bien venir l'époque où l'utilisateur ne se préoccupera plus de la forme de stockage de ses données mais par contre exigera un interface souple et performant. Ce sera donc notre tâche de développer cet outil en lui laissant toujours l'impression de manipuler un système relationnel puisque celui-ci passe pour être le plus convivial de tous, mais en tirant profit des avantages du réseau, par exemple. De plus on s'oriente de façon très marquée vers des langages d'interrogation de type graphique ou langage naturel où l'on fait abstraction de toute notion de programmation.

L'iDBP ne pouvait profiter pleinement de ces pointeurs dans la mesure où le programmeur ne disposait que de primitives de bas niveaux pour la manipulation des occurences dans une liste chainée.

Ce besoin de réintroduire les pointeurs dans des systèmes relationnels disponibles correspond à une réalité présente. La preuve en est que dans certains systèmes de C.A.O. de V.L.S.I. en cours de développement, on continue de faire appel aux systèmes relationnels mais en leur apportant les modifications nécessaires à l'introduction et à l'utilisation de tels liens. En effet il parait impensable aux concepteurs de circuits de parcourir une structure de cellule de circuit sans faire référence à des liens physiques préétablis.

### 4.1.3 CONFIGURATION D'UTILISATION DE L'iDBP

L'iDBP est le plus souvent présenté comme une machine base de données, ce qui implique de notre part de le considérer comme un calculateur offrant des accès à l'information plus rapides qu'un système conventionnel. Ses concepteurs lui conférent aussi le rôle de serveur de base de données dans un réseau d'ordinateurs.

L'idée d'un serveur unique est intéressante pour faire opposition aux problèmes inhérents à une configuration répartie intégrant plusieurs SGBD bâtis sur différents modèles. La gestion de l'information en est simplifiée et la localisation des données est unique. Par contre le serveur atteindra plus vite son seuil de saturation car l'intégralité des demandes d'accès lui sera conférée, exceptées celles portant sur d'éventuelles bases locales.

Cette notion de serveur peut être abordée selon deux points de vue:

- serveur pour un ensemble de gros calculateurs hôtes

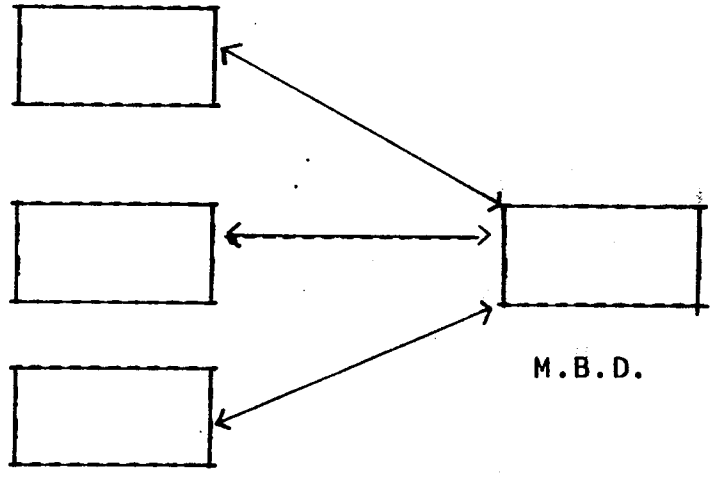



Figure 4.2

- serveur pour des postes de travail et des usagers finaux.

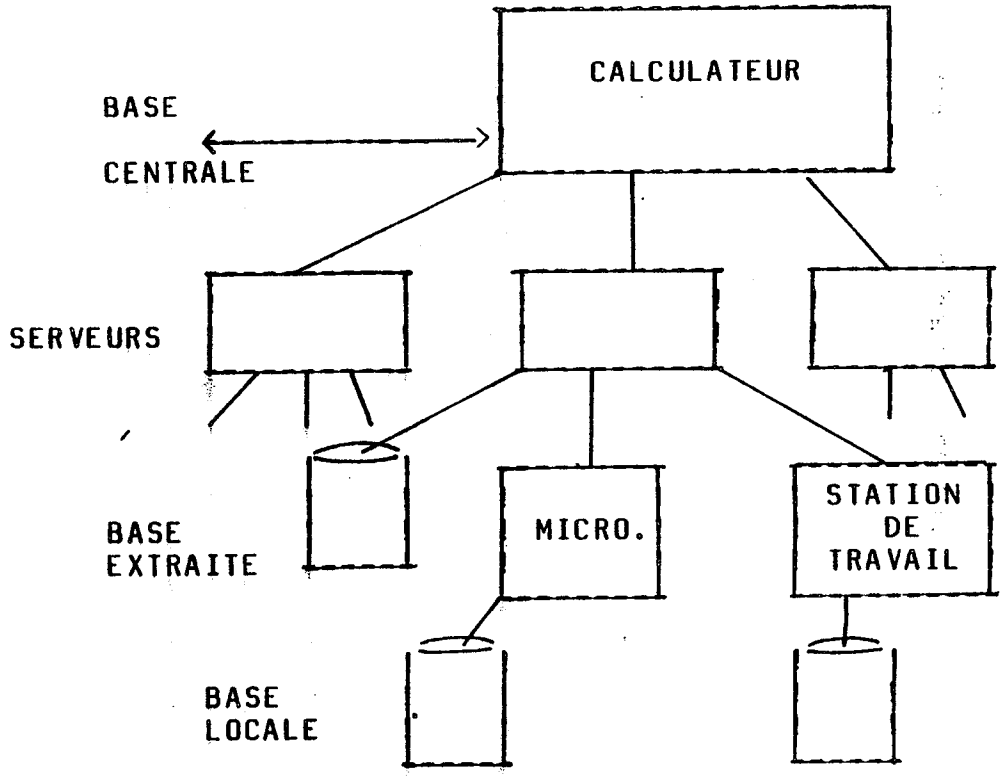



Figure 4.3

La seconde optique est novatrice et pourra constituer une forme de réponse aux exigences de performances et à une convivialité davantage marquée. Elle sera adressée à des groupes humains opérant dans des domaines d'application bien précis et faisant appel à une décentralisation de l'outil informatique sur des postes de travail de plus en plus performants. Nous allons tenté de décrire plus précisément cette seconde "version" de la notion de serveur en abordant le contenu de chacun des niveaux de sa hiérarchie.

## 4.2 NOUVELLE ARCHITECTURE DE SERVEUR

### 4.2.1 EXIGENCES A REMPLIR PAR CETTE NOUVELLE ARCHITECTURE

Jusqu'à présent rien ne peut nous assurer que l'issue à ces problèmes de coût d'accès ne se trouve uniquement dans le développement d'unités de recherche sophistiquées. Si l'on s'attarde un tant soit peu à analyser cet aspect de performance on s'aperçoit que les temps d'E/S prédominent dans les temps globaux de traitement parce que:

- on manipule des objets à structure complexe

- on traite des bases sans cesse croissantes en taille.

La taille ou la représentation informatique d'un objet ne sont pas des paramètres sur lesquels nous pourrons influer pour trouver une parade à ce problème. Au contraire, la technique ne cessera de nous donner la possibilité de travailler avec des descriptions de plus en plus fines de notre environnement.

Il existe un dénominateur commun à ces applications manipulant de tels objets complexes: leurs transactions sont de longue durée et très souvent ne manipulent pas les mêmes instanciations d'objet. Ainsi avec un système de gestion de données classique, de telles transactions amènent les utilisateurs à des situations d'interblocage pénalisantes car l'élément atomique verrouillé est encore trop souvent le fichier ou la relation. Pour les applications de C.A.O., il est fréquent de rencontrer un ensemble de personnes travaillant sur la même nature d'objets: la réprésentation logique de tous les éléments d'un circuit, par exemple. Pour une telle situation, on ne pourra autoriser le verrouillage de toutes les représentations logiques dès

l'instant où un utilisateur veut en modifier une. De plus les traitements réalisés sur ces objets sont de nature de plus en plus complexe et font appel à des postes de travail scientifiques qui sont de super micro-ordinateurs connectés sur un réseau local au calculateur central. Avec une base centralisée, ces stations sont tenues d'effectuer de nombreux échanges de données qui sont tous autant de facteurs pénalisants.

Par cette approche descendante nous sentons très bien se profiler une architecture répartie à plusieurs niveaux de hiérarchie.

### 4.2.2. PREMIER NIVEAU

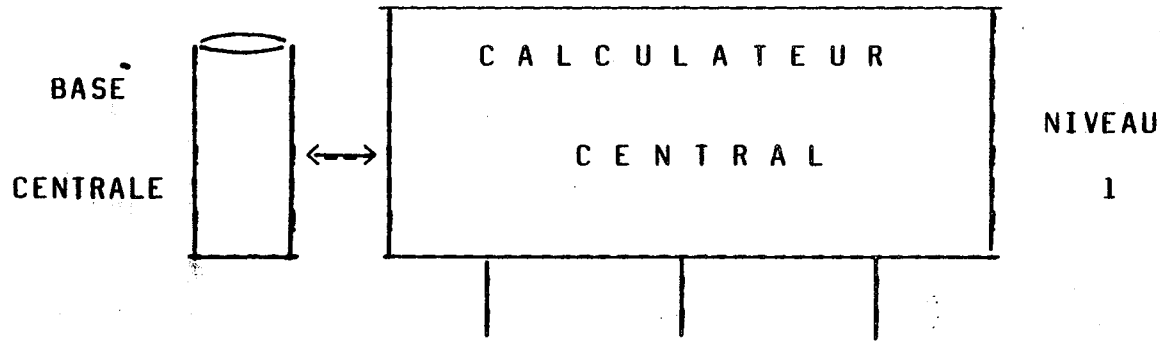



Figure 4.4 - ARCHITECTURE DU PREMIER NIVEAU

Nous conservons au plus haut niveau le calculateur principal avec la base de données centrale. Celle-ci sera basée sur un modèle de données de type hiérarchique ou réseau afin de limiter les coûts d'accès en évitant la mise en oeuvre d'opérateurs couteux d'autant plus que la taille de cette base centrale sera conséquente. Cette base sera dans un état de cohérence permanent. Elle sera principalement accessible, au niveau 1, par l'administrateur de la base de données.

### 4.2.3 SECOND NIVEAU

Au second niveau, nous voudrions instaurer plusieurs extracteurs qui pourraient être des mini-ordinateurs et qui réaliseraient une extraction de la base centrale (figure 4.5). Le contenu de chaque base extraite serait conditionnée par les besoins d'une unité humaine de production. Le découpage de la base principale devra être réalisé avec précaution afin de ne pas aboutir à des situations d'incohérences sur des objets bâtis sur une strcuture complexe où plusieurs éléments de celle-ci auraient pu être modifiés dans des contextes différents. Deux bases extraites pourront contenir la même occurrence et la gestion des verrous se fera au niveau du calculateur central.

Cette base extraite va plus loin que la notion de sous-structure rejoint très bien l'idée des vues que l'on a pu utiliser sur l'iDBP.

Ces bases extraites pourront être accédées par le troisième niveau de hiérarchie que nous décrirons plus tard mais également par les utilisateurs qui s'estimeront satisfaits des possibilités offertes par les différents calculateurs de ce second niveau. Par conséquent, ces bases seront bâties sur le modèle relationnel dans un souci bien évident de convivialité. L'handicap créé par les coûts d'exécutions des opérateurs des LMD du modèle relationnel sera minimisé puisque, par définition, le second niveau travaille sur une base de taille restreinte par rapport à la base centrale. Par conséquent les domaines d'exploration seront déjà considérablement réduits. Rien ne nous empèchera dans le futur de conserver un modèle réseau ou hiérarchique et d'offrir un interface de type relationnel.

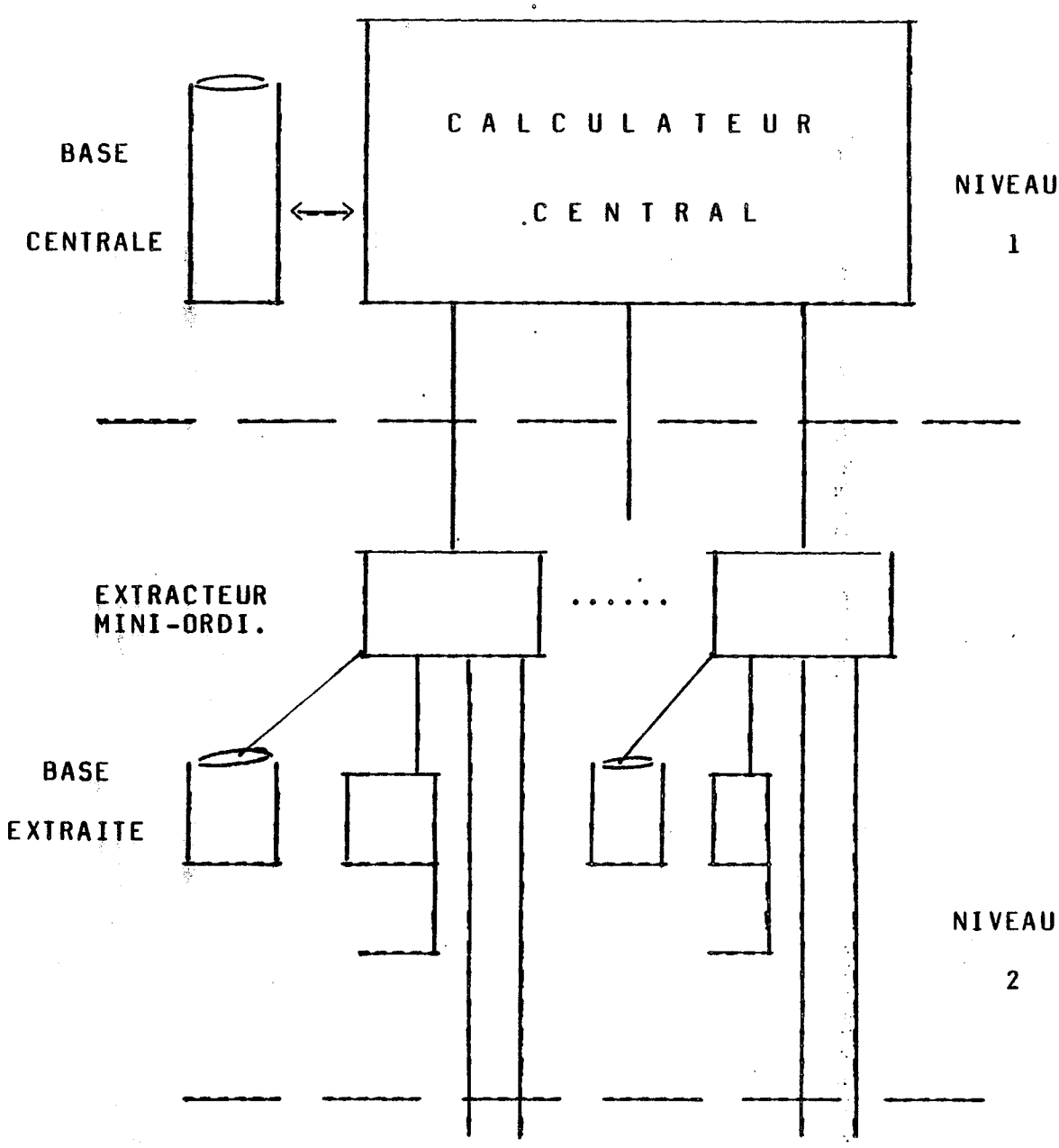



Figure 4.5 - ARCHITECTURE DES PREMIER ET SECOND NIVEAUX

Cette extraction modèle réseau / hiérarchique - modèle relationnel commence à être parfaitement maitrisée, la preuve en est l'annonce faite par IBM de charger une base DB2 depuis un site IMS. Les transferts devront être bi-directionnels afin de pouvoir réintégrer les modifications faites sur la base extraite dans la base centrale. Il reste à déterminer la fréquence de ces réintégrations. Certaines d'entre elles pourront être imminentes lorsque des utilisateurs d'autres bases extraites sont en attente de ressour-

ces afin de poursuivre leurs tâches. Pour les utilisateurs travaillant directement sur ces extracteurs, ces bases extraites ne sont pas assimilables à des bases de travail telle qu'elles sont décrites classiquement.

La gestion des contrôles d'accès pour les utilisateurs des niveaux 2 et 3 se fait au sein de chacun des extracteurs. Le contrôle des mises à jour simultanées des bases extraites sera mis en oeuvre au niveau 1 en fonction des descriptions des bases extraites.

Au niveau de chacun des extracteurs, on pourra bénéficier d'outils spécialement développés pour les utilisateurs que l'on associe à chacune des bases extraites. A ce stade rien ne nous empêcherait de développer des accélérateurs de recherche fortement conditionnés par la nature des travaux à réaliser sur chacun de ces extracteurs. De même, nous pourrions pousser la distinction entre ces groupes de spécialistes encore plus loin en développant pour chacun d'eux un modèle de données plus spécialement adapté à leurs problèmes. Dans le domaine des modèles de données généralisées, on s'aperçoit très vite que l'on arrive à développer un modèle par spécialité: un modèle document, un modèle de C.A.O., un modèle de données géographiques. Si l'on veut pouvoir intégrer dans la même application la gestion des documents, d'objets C.A.O. et d'images géographiques, pourquoi ne pas tenter cette approche multi-modèles? Il est vrai que les choses ne seraient pas aussi simples dans la mesure où les tâches de travail ne se découperaient pas de façon suffisamment rigoureuse pour leur attribuer un modèle précis. Notre architecture ne sera pas complète sans aborder le troisième niveau.

### 4.2.4 TROISIEME NIVEAU

Nous aurions très bien pu figer notre architecture avec deux niveaux mais nous allons présenter trois raisons ma-

jeures nécessitant de créer un troisième niveau:

- la personne qui fait de la C.A.O. s'assure avant de figer une solution, d'avoir explorer toutes les alternatives. Par conséquent, tant que son objet ne lui paraitra pas optimum, elle cherchera à le modifier et voudra conserver ses différents résultats. Cette nouvelle forme de travail s'insère très mal dans un système de gestion de données où rien n'a été prévu pour les différentes versions ou alternatives d'un objet. Par conséquent nous pourrions introduire cette nouvelle notion comme élément du modèle de données de C.A.O.. Cela commence à être réalisé mais néanmoins, nous estimons qu'il n'est pas nécessaire de conserver systématiquement tous les résultats de conception. Par conséquent, la base de données du second niveau dans notre architecture serait surchargée par toute une série de mises à jour qui seraient anihilées par la suite. Cette hypothèse de travail handicaperait tous les autres utilisateurs et dégraderait les performances de la base.

- d'autre part la base extraite, bien que de taille inférieure à la base centrale, peut être conséquente au point de vue place en mémoire secondaire. La représentation logique d'un circuit de 500000 transistors doit avoir une taille non négligeable. Ainsi le concepteur qui travaille sur la représentation de quelques cellules restera fortement pénalisé par la taille de la base extraite.

- enfin, les outils techniques offerts par le calculateur extracteur pourront demeurer trop généraux pour certaines opérations. L'utilisation de véritables postes de travail scientifiques, connectés à l'extracteur, s'avérera indispensable. Une unité de stockage pourra être attribuée à chacun de ces postes.

De ces trois exigences est née l'idée de développer un troisième niveau qui pourra être composé de postes de tra-

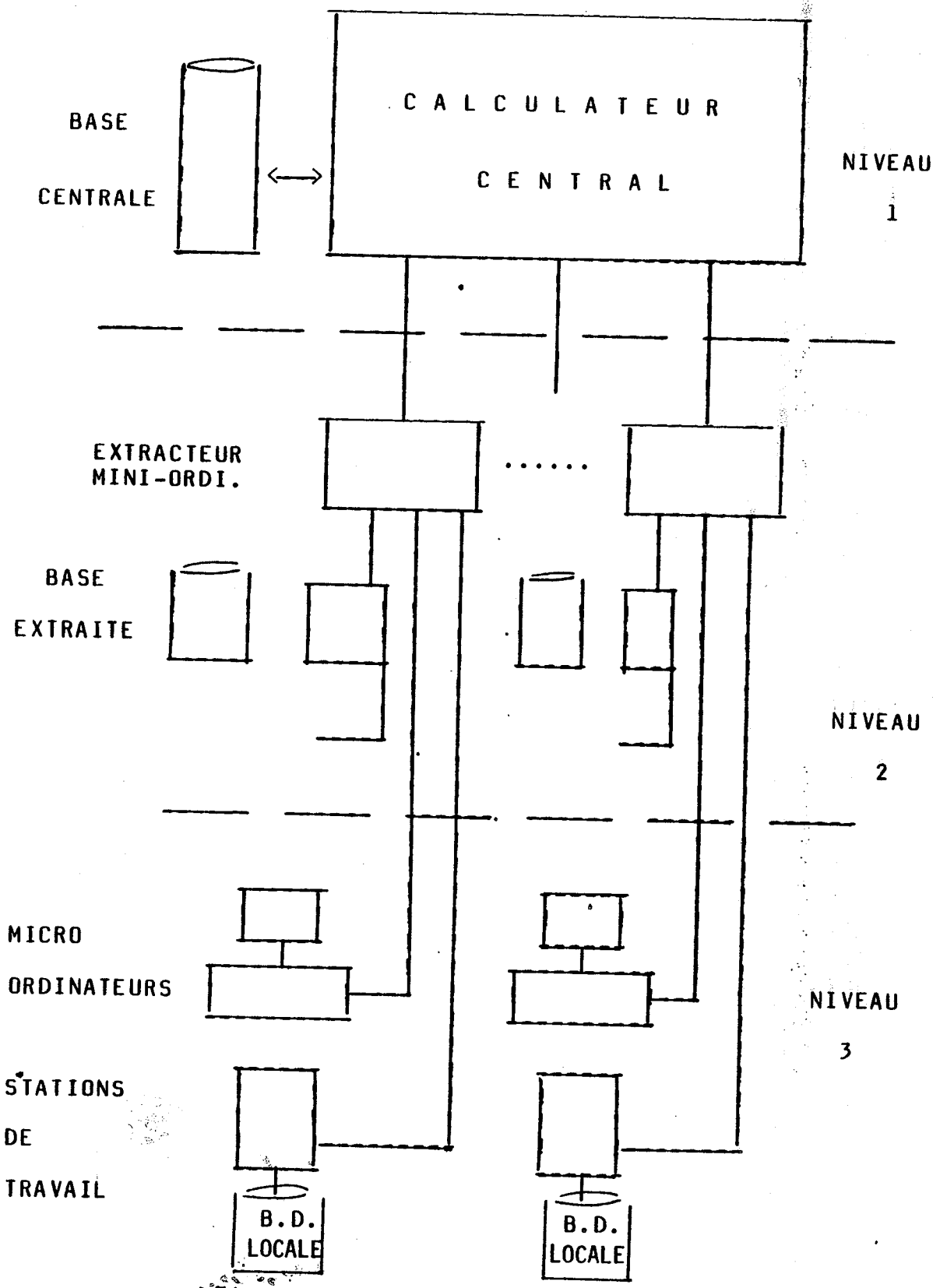



Figure 4.6 -ARCHITECTURE GLOBALE

vail spécialisés et/ou de micro-ordinateurs auxquels serait associée une base locale. Le contenu de ces bases locales sera composé d'éléments issus de la base extraite et de résultats d'opérations appliquées sur ces mêmes éléments. La réintégration des résultats dans la base du niveau 2 est à la charge de l'usager du niveau 3.

Par cette approche on réduit la taille des bases des utilisateurs et on leur fournit une base de travail, indépendante de la base centrale, sur laquelle aucun contrôle n'est effectué et dont l'excroissance ne pénalisera pas les autres usagers.

L'insertion de ce troisième niveau n'implique pas nécessairement qu'aucune application ne soit traitée au second niveau. Ainsi si l'on n'a pas besoin de stations de travail sophistiquées ou si notre travail ne consiste pas à générer et explorer des alternatives, nous pourrons très bien nous contenter des performances de l'extracteur. Il faudra connaître avec précision ses besoins et les possibilités du matériel qui nous est proposé afin de ne pas être pénalisé par une mauvaise spécification de l'architecture.

Une telle proposition nécessitera la mise en oeuvre de toute une gamme de nouveaux outils dont nous allons donner une liste non exhaustive.

## 4.3 OUTILS NOUVEAUX A METTRE EN OEUVRE

### 4.3.1 INTERFACE BASE RESEAU/HIERARCHIQUE - BASE RELATIONNELLE

Nous avons avancé l'idée de retenir le modèle hiérarchique ou réseau pour la base centrale et le modèle relationnel pour la base extraite. Les outils réalisant le passage de l'un à l'autre ne font que leur première apparition. La théorie a été explorée depuis plusieurs années <ADILEO> par

contre peu de logiciels ont été conçus dans leur intégralité.

### 4.3.2 DEFINITION DES BASES EXTRAITES

La notion de base extraite n'est pas comparable à la notion de sous-structure. Par conséquent, des outils permettant de définir le schéma de celles-ci devront être spécifiés et réalisés. Ils seront comparables aux opérateurs qui nous ont permis de définir des vues sur l'iDBP. On définit explicitement les occurrences, les caractéristiques des occurrences que l'on veut retenir pour chacune des bases et les droits d'accès sur ces instances. La structure est alors implicitement décrite. Cette dernière est déduite de la structure des entités qui aident à construire la base extraite et des opérateurs appliqués sur ces entités.

D'autre part il faut savoir que la structure de cette base n'est pas figée et qu'elle pourra à tout moment être modifiée. Le fait que la définition des bases extraites soit réalisée au niveau du calculateur central facilitera la répercussion de ces modifications sur les niveaux 2 et 3.

### 4.3.3. GESTION DES CONCURRENCES D'ACCES

On ne peut proposer aux utilisateurs de bloquer une relation entière dés qu'ils voudront modifier une occurrence d'autant plus que la relation peut être répartie entre les bases extraites. Il nous faudra pouvoir définir, à l'aide d'un formalisme intégré au LDD, l'atomicité de l'élément que l'on aura à verrouiller. Pour en revenir à notre exemple de C.A.O. de circuits, cela pourra être la représentation logique d'une cellule pour ceux qui s'attachent à la conception logique, le dessin du masque d'une porte ou d'une cellule pour ceux qui travaillent sur le "lay out".

### 4.3.4. GESTION DES BASES LOCALES

Au troisième niveau de la hiérarchie de notre structure nous avons fait référence à la notion de base locale, celle-ci sera composée d'éléments issus de la base extraite ainsi que d'un acquis scientifique qui permettra la génération de nouveaux éléments. La base locale pourra donc être assimilée à une base de connaissances à laquelle on aurait intégré des composants de la base de projet. Ces notions de base de connaissances et de base projet sont issues des systèmes de C.A.O. <DAVID 81>.

De cette définition on en déduit que tous les éléments de la base locale ne devront pas être réintégrés dans la base extraite. Il faudra donner la possibilité à l'utilisateur de pouvoir spécifier les éléments à transférer dans la base extraite. Cette opération ne peut être comparée au transfert base extraite - base centrale où là, on pourra demander la réintégration totale de la base extraite puisque l'on sait que son schéma est inclu dans celui de la base centrale. L'utilisateur pourra, malgré tout, demander à réintégrer plusieurs versions de son objet.

## 4.4 CONCLUSION

Cette configuration est issue d'une synthèse des besoins d'une nouvelle gamme d'utilisateurs, ceux de la C.A.O., et des résultats d'un comparatif entre une première forme de solution aux problèmes de coûts d'accès et d'architecture, intégrant matériel et logiciel, et d'un système purement logiciel. L'architecture de ce dernier est actuellement fortement controversée pour son utilisation dans les nouvelles applications.

A elle seule, notre proposition ferait l'objet de nombreuses années de recherche pour une équipe d'informaticiens et de futurs utilisateurs. Les concepteurs de telles architectu-

res devront se soumettre à un effort d'ouverture et de dialogue envers les personnes pour qui ils oeuvrent. Aujourd'hui, ce rapprochement est encore trop peu marqué bien que l'on s'oriente vers des équipes pluridisciplinaire, coopération projet TIGRE - projet CVT (C.A.O. de VLSI) <CVT>. Le premier effort à consentir va dans le sens bien précis d'une coopération entre spécialistes et usagers finaux. L'outil que l'on réalise n'est pas destiné à son concepteur, alors tournons nous vers son utilisateur ?

# CONCLUSIONS

Au cours de la première partie de cette thèse, nous avons mis en évidence les exigences des nouveaux domaines d'application, faisant appel à l'informatique, quant à leurs problèmes de gestion et de manipulation des données.

De là, nous avons montré comment les outils les plus sophistiqués en matière de gestion des informations, les systèmes de gestion de bases de données, satisfont certaines de ces exigences. Par contre, ils demeurent encore mal adaptés à la résolution de nombreux problèmes. Ceux-ci sont d'ordre fonctionnel et matériel.

Les premiers sont pris en compte par la spécification de nouveaux modèles de données plus riches en sémantique et s'orientant, notamment, vers l'intégration de la programmation en logique.

Nous avons porté l'intérêt de nos travaux sur l'aspect optimisation "matérielle" des fonctions d'accès aux bases de données.

La recherche dans le domaine des accélérateurs d'accès aux informations a été effervescente tout au long des 15 dernières années. De là sont nés plusieurs prototypes dont certains commencent à apparaître comme produits sur le marché des ordinateurs.

Jusqu'à présent, très peu d'études concrètes ont réussi à mettre en valeur l'apport de telles machines à un système traditionnel. C'est pourquoi avant de spécifier une nouvelle architecture, il nous a paru plus intéressant d'étudier, dans un pemier temps, une solution existante.

L'expérimentation s'est déroulée dans un contexte d'application de gestion mais avec une perspective d'adaptation aux nouveaux domaines d'applications référencés précédemment et plus précisément à celui de la C.A.O..

La conclusion principale de cette réalisation a été que, pour une application classique de gestion, l'apport d'une machine spécialisée, en l'occurence l'iDBP, à un système traditionnel tel que SOCRATE/CLIO, n'est pas primordial. Etant donné la complexité de certaines requêtes testées, nous pourrions avancer que ce type de configuration ne pourra être retenu comme tel pour des systèmes de Conception Assistée par ordinateur.

De là, nous avons ébauché une architecture de système, plus spécialement orientée vers la C.A.O., pouvant intégrer notamment des machines bases de données.

Enfin, il nous a paru prématuré de vouloir développer une solution universelle pour chacun de ces problèmes, qu'ils soient de modélisation ou de performances. La jeunesse de ces nouveaux domaines d'application doit nous conduire à développer dans un premier temps, des outils spécifiques. Seules, la réalisation et l'utilisation de ces prototypes à venir nous diront s'ils pourront être généralisés à plusieurs domaines d'activités.

# BIBLIOGRAPHIE


<ADI 78> : "UN MODELE RELATIONNEL ET UNE ARCHITECTURE POUR LES SYSTEMES DE BASES DE DONNEES REPARTIS. APPLICATION AU PROJET POLYPHEME"
Adiba Michel
Thèse de doctorat d'état
UNIVERSITE SCIENTIFIQUE ET MEDICALE DE GRENOBLE
Septembre 1978

<BACKEND> : "A BACK-END COMPUTER FOR DATABASE MANAGEMENT"
Canaday, Harrison, Ivie, Ryder,Wehr
BELL TELEPHONE LABORATORIES INC.
PISCATAWAY
NEW-JERSEY

<BANCILHON> : "MACHINES BASES DE DONNEES: UNE INTRODUCTION"
François Bancilhon
Septembre 1982

<BD3> : "BASES DE DONNEES ET NOUVELLES PERSPECTIVES"
INRIA ADI
Janvier 1983

<BER 78> : ETUDE FONCTIONNELLE D'UN PROCESSEUR DE BASES DE DONNEES HIERARCHIQUES: MAGE"
Berger Sabbatel Gilles
Thèse de 3ième cycle
INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE
Juin 1978

<BRITTON-LEE > : "IDM INTELLIGENT DATABASE MACHINE"
PRESENTATION GENERALE
BRITTON-LEE INC.
ALABANY CALIFORNIA

<BRITTON-LEE a>: "DESIGN DECISIONS FOR THE INTELLIGENT DATABASE MACHINE"
Epstein, P. Hawthorn
BRITTON-LEE INC.
ALABANY CALIFORNIA

<BUCH 84> : "CURRENT TRENDS IN CAD DATABASES"
Alejandro P. Buchmann
IIMAS NATIONAL UNIVERSITY OF MEXICO
May 1984

<BUR 83> : "MACHINES BASES DE DONNEES: ETAT DE L'ART ET PERSPECTIVES"
Burnier Marc
IMAG SYSECA
R.R. TIGRE Nº 8
GRENOBLE Octobre 1983

<BURRIA 82> : "BASES DE DONNEES ET NOUVELLES APPLICATIONS"
Adiba, Burnier, Delobel, Rialhe
RAPPORT DE RECHERCHE IMAG

<BURRIA 82 a> : "CONTRIBUTION AUX BASES DE DONNEES GENERALISEES"
Fascicule 1 Fascicule 2
Burnier Marc, Rialhe Dominique
Rapport de DEA
USMG/INPG
Juin 1978

<CAL 78> : "PROJET POLYPHEME: L'EXPRESSION ET LA DECOMPOSITION DE TRANSACTIONS DANS UN SYSTEME DE BASES DE DONNEES REPARTIS"
Caléca
Thèse de 3ième cycle
UNIVERSITE SCIENTIFIQUE ET MEDICALE DE GRENOBLE
Septembre 1978

<CASSM> : "ASSOCIATIVE PROGRAMMING IN CASSM AND ITS APPLICATIONS"
Stanley Y., W. Su
UNIVERSITY OF FLORIDA
GAINESVILLE
FLORIDA

<CASSM a> : "THE ARCHITECTURE FEATURES AND IMPLEMENTATION TECHNIQUES OF THE MULTICELL CASSM"
Stanley, Su, Nguyen, Ehan, Lipowski
IEEE TRANSACTIONS COMPUTERS VOL C-28
Nº 6 June 1979

<COPER 1> : "COPERNIQUE DORSAL 32: MACHINE DE GESTION DE BASES DE DONNEES NAVIGATIONNELLES ET RELATIONNELLES"
BROCHURE TECHNIQUE
COPERNIQUE
78380 BOUGIVAL

<COPER 2> : "DIRAM 32: DISQUE AVEC RECHERCHE ASSOCIATIVE ET ANTE MEMOIRE" BROCHURE TECHNIQUE
COPERNIQUE
78380 BOUGIVAL

<CVT> : "PROJET CVT"
CNET
MEYLAN

<DAVID 81> : "METHODOLOGIE POUR LA CONSTRUCTION DE SYSTEMES CAO, SIGMA-CAO"
DAVID
Thèse d'etat USMG/INPG
Septembre 1981

<DBM> : "DATABASE MACHINE ARE COMING
DATABASE MACHINE ARE COMING"
David K. Hsiao
OHIO STATE UNIVERSITY
HANCH 1979

<DELADI 83> : "BASES DE DONNEES ET SYSTEMES RELATIONNELS"
Delobel, Adiba
DUNOD INFORMATIQUE

<DEWBITT 83> : "BENCHMARK DATABASE SYSTEMS: A SYSTEMATIC APPROACH"
Dina Bitton, David J. Dewitt, Carolyn Turbyfill
COMPUTER SCIENCES DEPARTMENT
UNIVERSITY OF WISCONSIN
MADISON USA

<DEWBOR 83> : "DATABASE MACHINE: AN IDEA WHOSE TIME IS PAST ? A CRITIQUE OF THE FUTURE OF DATABASE MACHINES"
Boral H.
COMPUTER SCIENCES DEPARTMENT
TECHNION ISRAEL
DeWitt J. David
COMPUTER SCIENCES DEPARTMENT

UIVERSITY OF WISCONSIN
MADISON USA

<DEWBOR 84> : "A METHODOLOGY FOR DATABASE SYSTEM PERFORMANCE EVALUATION"
Haran Boral, David J. Dewitt
COMPUTER SCIENCES TECHNICAL REPORT N 532
January 1984
COMPUTER SCIENCES DEPARTMENT
UNIVERSITY OF WISCONSIN
MADISON USA

<DEWHAW 81> : "A PERFORMANCE EVALUATION OF DATABASE MACHINE ARCHITECTURES"
David J. Dewitt, Paula B. Hawthorn
VLDB 81 IEEE 81

<DIRECT> : "PROCESSOR ALLOCATION STRATEGIES FOR MULTIPROCESSOR DATABASE MACHINES"
Boral, Dewitt
UNIVERSITY OF WISCONSIN
ACM TRANSACTIONS ON DATABASE SYSTEMS
VOL 6 N 2
June 1981 PAGES 227 - 254

<DIRECT a> : "QUERY EXECUTION IN DIRECT"
David J. Dewitt
UNIVERSITY OF WISCONSIN
MADISON USA

<DIRECT b> : "IMPLEMENTATION OF THE DATABASE MACHINE DIRECT"
Boral, Dwitt, Friedland, Jarrel
WILKINSON
IEEE TRANSACTIONS ON SOFTWARE ENGINEERING
VOL SE-8 N 6 November 1982

<DIRECT c> : "DIRECT: A MULTIPROCESSOR ORGANIZATION FOR SUPPORTING RELATIONAL DATABASE MANAGEMENT SYSTEMS"
David J. Dewitt
IEEE TRANSACTIONS ON COMPUTERS
VOL C-28 N 6 June 1979

<FERRAT 83> : "EXPRESSION ET CONTROLE DE L'INTEGRITE SEMANTIQUE DANS LES BASES DE DONNEES RELATIONNELLES. PROJET MICROBE"
Ferrat Lounas
Thèse de 3ième cycle
INPG
Mai 1983

<FOISSEAU 82> : "TECHNIQUES INFORMATIQUES ET C.A.O."
Foisseau Jack
INFORMATIQUE ET GESTION n° 138
Décembre 1982

<HSIAO 76> : "THE ARCHITECTURE OF A DATABASE COMPUTER PART III: THE DESIGN OF THE MASS MEMORY AND ITS RELATED COMPONENTS"
Hsiao, Kannan
OHIO STATE UNIVERSITY
Décembre 1976

<HSIAO al 78> : "CONCEPTS AND CAPABILITIES OF A DATABASE COMPUTER "
David K. Hsiao, Jayanta Banerjee
THE OHIO STATE UNIVERSITY
Richard I. Baum
IBM POUGHKEEPSIE LABORATORY
ACM TRANSACTIONS ON DATABASE SYSTEMS
VOL 3 N°4 Décembre 1978 P.347-384

<HSIAO 79> : "DBC-A DATABASE COMPUTER FOR VERY LARGE DATABASES"
Hsiao, Banerjee, Kannan
IEEE TRANSACTIONS ON DATABASE SYSTEMS
VOL 3 N°4 Décembre 1978 P.347-384

<HSIAO 81> : "CONCEPTS AND CAPABILITIES OF A DATABASE COMPUTER"
ACM TRANSACTIONS ON DATABASE SYSTEMS
VOL 3 N°4 Décembre 1978 P.347-384

<iDBP 1> : "SYSTEM 86/330A HARDWARE REFERENCE MANUAL"
INTEL C.
SANTA CLARA CALIFORNIA

<iDBP 2> : "GUIDE TO iDBP"
INTEL C.
SANTA CLARA CALIFORNIA

<iDBP 3> : "iDBP 86/440A HARDWARE REFERENCE ADDENDUM"
INTEL C.
SANTA CLARA CALIFORNIA

<iDBP 4> : "iDBP HOST LINK REFERENCE MANUAL"
INTEL C.
CALIFORNIA

<iDBP 5> : "iDBP OPERATIONS MANUAL"
INTEL C.
CALIFORNIA

<iDBP 6> : "iDBP DELPHI USER'S GUIDE"
INTEL C.
SANTA CLARA CALIFORNIA

<iDBP 7> : "iDBP DBMS COMPLETION CODES"

INTEL C.
SANTA CLARA CALIFORNIA

<KAT 83> : "MANAGING CHIP DESIGN DATABASE"
Katz
UNIVERSITY OF WISCONSIN MADISON
R.R. N°506
WI 53706 May 1983

<LANG> : "DATABASES MACHINES: AN INTRODUCTION"
Glen G., langdon JR
IEEE TRANSACTIONS ON COMPUTER
VOL C-28 N°6 June 1979

<LOWENTHAL AL> : "DATABASE MACHINES AND SOME ISSUES ON DBMS STANDARDS"
Stanley, Chang, Copeland, Fischer, Lowenthal, Schuster

<MUN 83> : "DATABASE MACHINES"
THIRD INTERNATIONAL WORKSHOP ON DATABASE MACHINES
EDITED BY H.O. Leilich and M. Missikoff
MUNICH September 1983

<NICO> : "AN OUT-LINE OF BD-GEN: A DEDUCTIVE DBMS"
Nicolas, Yazdanian
R.R. ONERA CERT
Toulouse
Octobre 1982

<PALA 84> : "LE MODELE DE DONNEES ET SA REPRESENTATION RELATIONNELLE DANS UN SYSTEME DE GESTION DE BASES DE DONNEES GENERALISEES : PROJET TIGRE"

Palazo José

Thèse de docteur ingénieur I N P G

Juin 1984

<PROPAL 2> : "PROCESSEUR PARALLELE ASSOCIATIF: PROPAL 2"
PRESENTATION
CIMSA

<RAP 2> : "AN ASSOCIATIVE PROCESSOR FOR DATABASES"
S.A. Schuster, H.B. Nguyen, E.A. Ozkarahan, K.C. Smith
UNIVERSITY OF TORONTO
THE 5 TH ANNUAL SYMPOSIUM COMPUTER ARCHITECTURE
POLO ALTO CALIFORNIA USA
April 1978

<RAP 2 a> : "RAP 2: AN ASSOCIATIVE PROCESSOR FOR DATABASES AND ITS APPLICATIONS"
Steward, Schuster, Nguyen, Esen, Ozkarahan, kenneth, Smith
IEEE TRANSACTIONS ON COMPUTERS
VOL C-28 N°6 June 1979

<RAP 2 b> : "PERFORMANCE EVALUATION OF A RELATIONAL ASSOCIATIVE PROCESSOR"
Ozkarahan, Schuster, Sevich
UNIVERSITY OF TORONTO
ACM TRANSACTIONS ON DATABASE SYSTEMS
VOL 2 N°2 June 1977 P. 175-195

<RDBM> : "RDBM: A RELATIONAL DATABASE MACHINE"
Auer, Hell, Leilich, Lie, Scweppe, Seehuser, Stiege, Teich, Zeidler
UNIVERSITAT BRAUNSCHWEIG
BRAUNSCHWEIG.
GERMANY

<RIE 83> : "IDM 500 WITHIN A MAINFRAME ENVIRONMENT:
SOME FIRST EXPERIENCES"
Riechman Christian
FGAN, Forschunginstitut fur funk und mathematik
Konigstrase 2, D-5307 Wachtberg
DATABASE MACHINES
INTERNATIONAL WORKSHOP
Munich September 1983

<SCH 83> : "GEI'S EXPERIENCE WITH BRITTON-LEE'S IDM"
Schumacher Gunter
GEI - Gesellschaft fur elektronische
Informationsverarbeitung M.B.H.
Albert Einstein Strase 61
D-5100 Aachen Walheim
DATABASE MACHINES
INTERNATIONAL WORKSHOP
Munich September 1983

<SLO 70> : "LOGIC PER TRACK DEVICES IN"
Slotnick D.L.
ADVANCES IN COMPUTERS
VOL 10 (ed. Ton J.)
ACADEMIC PRESS 1970

<SMITH> : "RELATIONAL DATABASE MACHINES"
Diane C.P. Smith
John Miles Smith
UNIVERSITY OF NEW MEXICO

<SOCREF> : "SOCRATE/CLIO: MANUEL DE REFERENCE"
SYSECA
38000 MEYLAN

<SOCUTI> : "SOCRATE/CLIO: MANUEL UTILISATEUR"
SYSECA

38000 MEYLAN

<STRAW al 83> : "EXPERIMENTS IN BENCHMARKING RELATIONAL DATABSE MACHINES"
R. Bogdanowicz, M. Crohen, David K. Hsiao, P. Strawser
NAVAL POSTGRADUATE SCHOOL
MONTEREY CA 93940
THIRD INTERNATIONAL WORKSHOP ON DATABASE MACHINES
MUNICH September 1983

<TRIM 81> : "A STRUCTURED DESIGN METHODOLOGY AND ASSOCIATED SOFTWARE TOOLS"
Trimberger S., Rowson J., Lang C., Gray J.
IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS
VOL CAS-28 Nº 7
July 1981

<VELE 84> : "UN MODELE ET UN LANGAGE POUR LES BASES DE DONNEES GENERALISEES : PROJET TIGRE"
Velez Fernando
Thèse de docteur ingénieur I N P G
Septembre 1984

<VERSO> : "VERSO: A RELATIONAL BACKEND DATABASE MACHINE"
Bancilhon, Fortin, Gamerman, Laubin, Richard, Scholl, Tusera, Verroust
INRIA LRI

LISTE DES RAPPORTS TIGRE

[TIGRE 1] Présentation générale du projet TIGRE
Janvier 1983

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[TIGRE 18] Information processing for CAD/VLSI on a generalized data management system
G. Nguyen, M. Adiba - Juin 1984

[TIGRE 19] Représentation de la sémantique et des méta-connaissances dans une BD généralisées.
G. Nguyen, M. Adiba - Juin 1984

[TIGRE 20] Description de l'éditeur TIGRE -
1) Fonctionnalités, Architecture, Interfaces.
I. Vatton, H. Richy - Juillet 1984

# ANNEXE

## STRUCTURE

## SOCRATE / CLIO

## DE LA

## BASE DE TEST

```
*****************************************************************************

UT
 ENTITE (20000) ELEVE
 DEBUT
      NOM MOT (15)
          AVEC $DICO CLE DOUBLE AVANT CHAINE SIMPLE
          NON ORDONNE (20000) *DICO FIN
      PRENOM MOT (15)
      SEXE (3 3) (MAS FEM X)
      DATE-NAISSANCE DECIMAL (6V0) DE  500101 A  991231
      ADRESSE MOT (30)
      RESPONSABLE
      DEBUT
          NOM MOT (15)
          PRENOM MOT (15)
          ADRESSE MOT (30)
          TELEPHONE BINAIRE DE  0 A  9999999
      FIN
      ANNEE-ENTREE-6EME BINAIRE DE  50 A  99
      ANNEE-ENTREE-LYCEE BINAIRE DE  50 A  99
      SPORT1 (30 11) (
      GYM VOLLEY FOOT TENNIS SKI NATATION RUGBY VOILE SPELEO PETANQUE JUDO E
      POIS CYCLISME MARCHE MARTEAU SAUT PECHE JAVELOT)
      SPORT2 (30 11) (
      GYM VOLLEY FOOT TENNIS SKI NATATION RUGBY VOILE SPELEO PETANQUE JUDO E
      POIS CYCLISME MARCHE MARTEAU SAUT PECHE JAVELOT)
      HOBBY MOT (15)
      MALADIE-INFANT (10 11) (ROUGEOLE VARICELLE OREILLONS RUBEOLE COQUELUCH
      MALADIE-GRAVE MOT (20)
      CLASSES REFERE  ELEVES DE UN CLASSES
          AVEC CHAINE SIMPLE FIN
 *FICH
 $NOTE
      ENTITE (10) NOTES
      DEBUT
          PLACE BINAIRE DE  1 A  50
          MOYENNE DECIMAL (2V1) DE  0,0 A  20,0
      FIN
 *NOTE
 $PASS
      ENTITE (15) PASSE
      DEBUT
          CLASSES (10 7) (6EME 5EME 4EME 3EME 2EME 1EME TERM)
          RESULTAT (2 7) (ADMIS DOUBLE)
          OBSERVATIONS FILLER (120)
      FIN
 FIN
SS
:H
 ENTITE (30) ETABLISSEMENT
 DEBUT
      VILLE MOT (15)
          AVEC $DICO CLE DOUBLE AVANT CHAINE SIMPLE
          NON ORDONNE (30) *DICO FIN
      NOM MOT (20)
      ADRESSE MOT (30)
      TELEPHONE BINAIRE DE  0 A  9999999
      TYPE (5 11) (LYCEE COLLEGE CES TECHNIQUE)
      DIRECTEUR
      DEBUT
```

```
        NOM MOT (15)
        PRENOM MOT (15)
        AGE BINAIRE DE  20 A  65
        PRISE-FONCTION BINAIRE DE  0 A  80
    FIN
    CAPAC-ELEVES BINAIRE DE  0 A  10000
    NBRE-SALLES BINAIRE DE  0 A  1000
    %-RECU-BAC BINAIRE DE  0 A  100
    NBRE-POSTES-TITULAIRES BINAIRE DE  0 A  1000
    NBRE-POSTES-VACATAIRES BINAIRE DE  0 A  1000
    VISITE-SECURITE BINAIRE DE  90 A  99
    ETAB-DEPENDANTS ANNEAU
        AVEC CHAINE SIMPLE FIN /* MAISON-MERE DE UN ETABLISSEMENT */
    MAISON-MERE REFERE  ETAB-DEPENDANTS DE UN ETABLISSEMENT
        AVEC CHAINE SIMPLE FIN
    POSTES ANNEAU
        AVEC CHAINE SIMPLE FIN /* ETABLISSEMENT DE UN POSTE  DE UN PROF
    CLASSES INVERSE TOUT CLASSES
FIN
ENTITE (3600) CLASSES
DEBUT
    ELEVES ANNEAU
        AVEC CHAINE SIMPLE FIN /* CLASSES DE UN ELEVE */
    PROGRAMME REFERE  CLASSES DE UN PROGRAMME
        AVEC CHAINE SIMPLE FIN
    NIVEAU (20 7) (6EME 5EME 4EME 3EME 2DEA 2DEB 2DEC 1EREA 1EREB 1EREC
    NO-CLASSE BINAIRE DE  1 A  20
    NO-PIECE MOT (5)
    NUMETAB BINAIRE DE  0 A  30
*FICH
$CLAS
    PR-PRINC MOT (15)
    CHEF-CL MOT (15)
*CLAS
$COUR
    ENTITE (18) COURS
    DEBUT
        JOURS (5 11) (LUNDI MARDI JEUDI VENDREDI SAMEDI)
        HEURE BINAIRE DE  1 A  4
        MATIERE (30 15) (
        FRANCAIS MATHS GEO SCIENCE-NAT SPORT CHANT DESSIN ANGLAIS PHIL
        PHYSIQUE CHIMIE TECHNOLOGIE ED-SEXUELLE ATELIER AUTRE)
        PROFESSEUR REFERE  COURS DE UN PROFESSEUR
            AVEC CHAINE DOUBLE FIN
    FIN
FIN
*COUR
$FICH
    ENTITE (20) PROGRAMME
    DEBUT
        CLASSES ANNEAU
            AVEC CHAINE SIMPLE FIN /* PROGRAMME DE UN CLASSES */
        NIVEAU (20 7) (6EME 5EME 4EME 3EME 2DEA 2DEB 2DEC 1EREA 1EREB 1ERE
            AVEC $DICO CLE UNIQUE CHAINE SIMPLE *DICO FIN
        FILPROG FILLER (30)
        ENTITE (20) MATIERE
        DEBUT
            NOM (30 15) (
            FRANCAIS MATHS GEO SCIENCE-NAT SPORT CHANT DESSIN ANGLAIS PHIL
            PHYSIQUE CHIMIE TECHNOLOGIE ED-SEXUELLE ATELIER AUTRE)
            NBRE-COURS BINAIRE DE  0 A  10
```

```
        NBRE-TP BINAIRE DE  0 A  10
        NBRE-TD BINAIRE DE  0 A  10
        MATERIEL FILLER (240)
    FIN
FIN
ENTITE (3000) PROFESSEUR
DEBUT
    CODE-ENSEIGNANT BINAIRE DE  10000 A  99999
        AVEC $DICO CLE UNIQUE CHAINE SIMPLE
        NON ORDONNE (0) *DICO FIN
    NO-SECU-SOCIALE
    DEBUT
        MAIS BINAIRE DE  10000 A  29999
        LIEU BINAIRE DE  0 A  99999
        NUM BINAIRE DE  0 A  999
    FIN
    NOM MOT (15)
        AVEC $DICO CLE DOUBLE AVANT CHAINE SIMPLE
        NON ORDONNE (3000) $DICO FIN
    PRENOM MOT (15)
    SEXE (3 3) (MAS FEM X)
    SIT-FAMILLE (8 11) (CELIBAT MARIE DIVORCE VEUF SEPARE CONCUBIN POLYGAM
    ADRESSE MOT (30)
    DATE-NAISS BINAIRE DE  0 A  991231
    LIEU-NAIS MOT (30)
    MATIERE (30 15) (
    FRANCAIS MATHS GEO SCIENCE-NAT SPORT CHANT DESSIN ANGLAIS PHILO ALLEMA
    PHYSIQUE CHIMIE TECHNOLOGIE ED-SEXUELLE ATELIER AUTRE)
        AVEC $DICO CLE DOUBLE AVANT CHAINE SIMPLE *DICO FIN
    DIPLOME (15 11) (
    CERT.ETUDE ECOLE.BUIS BREVET BAC LICENCE MAITRISE DEA DOCTORAT AGREG N
    I.S.M.A.R)
    BANQUE
    DEBUT
        NOM MOT (5)
        AGENCE MOT (20)
        NUM-COMPTE BINAIRE DE  0 A  9999999
    FIN
    SALAIRE DECIMAL (5V2) DE  0,00 A  99999,00
    COURS ANNEAU
        AVEC CHAINE DOUBLE FIN /* PROFESSEUR DE UN COURS  DE UN CLASSES */
    ENTITE (5) POSTE
    DEBUT
        ETABLISSEMENT REFERE  POSTES DE UN ETABLISSEMENT
            AVEC CHAINE SIMPLE FIN
        FILLER (50)
    FIN
FIN
INV-CLASSE INVERSE TOUT CLASSES
INV-ELEVE INVERSE TOUT ELEVE
INV-PROF INVERSE TOUT PROFESSEUR
INV-COURS INVERSE TOUT COURS  DE UN CLASSES
INV-PROG INVERSE TOUT PROGRAMME
INV-MATIERE INVERSE TOUT MATIERE  DE UN PROGRAMME
INV-POSTE INVERSE TOUT POSTE  DE UN PROFESSEUR
INV-PASSE INVERSE TOUT PASSE  DE UN ELEVE
INV-NOTE INVERSE TOUT NOTES  DE UN ELEVE
INV-ETABLI INVERSE TOUT ETABLISSEMENT
INV-RESP INVERSE TOUT ELEVE
INV-DIRECT INVERSE TOUT ETABLISSEMENT
INV-BANQUE INVERSE TOUT PROFESSEUR

INVI INVERSE TOUT ELEVE
```

# ANNEXE

## EXEMPLE DE PROGRAMMES ET DE STRUCTURES iDBP

## SCHEMA DE LA BASE SOUS iDBP

ELEVE (NOM,..., ↑PERE_CLASSES, ↑FRERE_CLASSES, ↑FILS_NOTES,
↑FILS_PASSE)

NOTE (PLACE,MOYENNE,↑PERE_ELEVE,↑FRERE_ELEVE)

PASSE (CLASSE,...,↑PERE_ELEVE,↑FRERE_ELEVE)

ETABLISSEMENT (RANG,VILLE,...,↑FILS__POSTE)

ETAB_CLAS (RANG,↑CLASSES)

CLASSES ( NIVEAU,...,↑FILS_COURS,↑FILS_ELEVE,...)

COURS (JOURS,HEURE,...,↑PERE_CLASSES,↑FRERE_CLASSES,...)

PROGRAMME (NIVEAUX,...,↑FILS_MATIERE,↑FILS_CLASSES)

MATIERE (NOM,...,↑PERE_PROGRAMME,↑FRERE_PROGRAMME)

PROFESSEURS ( NOM,...,↑FILS_POSTE,↑FILS_COURS)

POSTE (RANG,↑PERE_ETAB,↑FRERE_ETAB,↑PERE_PROF)

## EXEMPLE DE STRUCTURE iDBP

```
FILE ELEVES SMALLPAGE DBPSYS 4500 5500 ;

SCHEMA ELEVES STR-REUSE NULL OW 216W

        ELEVE-PT            SELF-RID
        RANG                INTEGER          4
        NOM                 FIXED-ASCII      15
        PRENOM              FIXED-ASCII      15
        SEXE                FIXED_ASCII      3
          .
          .
          .
          .
        PTR-FRERE-CLASSES   RECPTR
        PTR-PERE-CLASSES    RECPTR
        PTR-FILS-NOTES      RECPTR
        PTR-FILS-PASSE      RECPTR
          .
          .
          .                                    ;

FILE NOTES SMALLPAGE DBPSYS 150 W 250W ;

SCHEMA NOTES STR-REUSE NULL OW 250W

        NOTES-PTR                SELF-RID
        RANG                     INTEGER          4
        PLACE                    INTEGER          1
        MOYENNE                  INTEGER          1
        MOIS                     INTEGER          1
        PTR-FRERE-ELEVES         RECPTR
        PTR-PERE-ELEVES          RECPTR ;

FILE CLASSES SMALLPAGE 45W 50W ;

SCHEMA CLASSES STR-REUSE NULL OW OW

        CLASSES-PT               SELF-RID
        RANG                     INTEGER
        PTR-FILS-ELEVES          RECPTR
        PTR-PERE-PROGRAMMES      RECPTR
        PTR-FRERE-PROGRAMMES     RECPTR
          .
```

;

```
FILE PROGRAMMES SMALLPAGE 4W 6W ;

SCHEMA PROGRAMMES STR-REUSE NULL OW OW

        PROGRAMMES-PT           SELF-RID
        RANG                    INTEGER
        PTR-FILS-CLASSES        RECPTR
        NIVEAUX                 FIXED-ASCII     5
        PTR-FILS-MATIERES       RECPTR ;
```

## DEFINITION DE CONTRAINTES D'INTEGRITE

```
IC ELEVES .DATE_NAISSANCE WHERE-IC SPANS .DATE_NAISSANCE 500101
991231 OR-IC NOT-VALUED;

IC ELEVES .RANG WHERE-IC LE .RANG 20000;
```

## DEFINITION D'INDEX

```
INDEX INDEX-FLAGS ELEVES .RANG DBPSYS 75W OW ;

INDEX INDEX-FLAGS ETABLISSEMENTS .VILLE DBPSYS 1W OW ;
```

## REQUETE 1

PARCOURS ELEMENT PAR ELEMENT

```
ATTACH ELEVES READ-READ ;

VIEW ELE SELECT ELEVES WHERE-ITEM EQ USE-HELP .NOM 'PETIT' ;

START £1 ELE BI ;

FIND £1 ABS 0 ;
FETCH £1 1 ;
FIND £1 AHEAD 1 ;
FIND £1 AHEAD 1 ;
NEXT £1 ;
NEXT £1 ;
FIND £1 AHEAD 1 ;
FIND £1 AHEAD 1 ;

FREE ALL ;
```

## RECHERCHE MULTI-CRITERES

PARCOURS ELEMENT PAR ELEMENT

```
ATTACH ELEVES READ ;

VIEW ELE SELECT ELEVES WHERE-ITEM EQ USE-HELP..2 'PETIT
' AND-ITEM EQ NO-HELP .SEXE 'MA$' ;

START £1 ELE BI ;

FIND £1 ABS 0 ;
FIND £1 AHEAD 1 ;
NEXT £1 ;

FREE ALL ;
```

```
ATTACH ELEVES READ ;

VIEW ELE SELECT ELEVES WHERE-ITEM EQ USE-HELP .2 'PETIT
' AND-ITEM EQ NO-HELP .SEXE 'MAS' AND-ITEM EQ NO-HELP .MALADIE-
GRAVE NULL ;

START £1 ELE BI ;

FIND £1 ABS 0 ;
FIND £1 AHEAD 1 ;
NEXT £1 ;

FREE ALL ;
```

## PARCOURS D'ANNEAU

### PARCOURS ELEMENT PAR ELEMENT

```
ATTACH CLASSES READ ;

VIEW CLA SELECT CLASSES WHERE-ITEM EQ USE-HELP .RANG 000E0000H ;

START £1 CLA BI ;

FIND £1 ABS 0 ;

ATTACH ELEVES READ ;

START £2 ELEVES RANDOM ;

SET £2 £1.PTR-FILS-ELEVES ;
FETCH £2 1 ;
SET £2 £2.PTR-FRERE-CLASSES ;
```

```
FETCH £2 1 ;

SET £2 £2.PTR-FRERE-CLASSES ;
FETCH £2 1 ;

SET £2 £2.PTR-FRERE-CLASSES ;
FETCH £2 1 ;

FREE ALL ;
```

## REQUETE JOIN1

PARCOURS ELEMENT PAR ELEMENT

```
ATTACH ETABLISSEMENTS READ CLASSES READ ;

VIEW ETAB JOIN ETABLISSEMENTS .RANG CLASSES .NUMETAB ;

VIEW CLA SELECT ETAB WHERE-ITEM EQ NO-HELP .NOM 'LOUIS-LE-GRAND'
AND-ITEM EQ USE-HELP .VILLE 'MARSEILLE' ;

START £2 CLA BI ;

FIND £2 ABS 0 ;
FIND £2 AHEAD 1 ;
NEXT £2 ;

FREE ALL ;
```

## REQUETE JOIN2

PARCOURS ELEMENT PAR ELEMENT

```
ATTACH ETABLISSEMENTS READ CLASSES READ ;

VIEW ETAB SELECT ETABLISSEMENTS WHERE-ITEM EQ NO-HELP .NOM 'LOUIS-
LE-GRAND' AND-ITEM EQ USE-HELP .VILLE 'MARSEILLE' ;

START £1 ETAB BI ;

FIND £1 ABS 0 ;

START £2 CLASSES RANDOM ;

SET £2 £1.PTR-FILS-CLASSES ;

SET £2 £2.PTR-FRERE-ETABLISSEMENTS ;

SET £2 £2.PTR-FRERE-ETABLISSEMENTS ;
FETCH £2 1 ;

SET £2 £2.PTR-FRERE-ETABLISSEMENTS ;

FREE ALL ;
```

## APPEL DU PROGRAMME, MJAN, DEPUIS LE VAX

## EN VUE D'UNE EXECUTION SUR L'IDBP

```
INVOKE MJAN 1 'BURNIER6'
```

## DECOMPOSITION DE LA MACRO Q5

```
MACRO Q5 ,
ATTACH PROGRAMMES READ-READ
       CLASSES READ-READ
       ELEVES READ-READ ;

VIEW PROG SELECT PROGRAMMES WHERE-ITEM
EQ USE-HELP .NIVEAUX *1 ;

START £1 PROG BI ;
START £2 CLASSES RANDOM ;
START £3 ELEVES RANDOM ;

FIND £1 ABS 0 ;
SET £2 £1.PTR-FILS-CLASSES ;

LOOPWHILE VALUED £2.CLASSES-PT ,

  IF VALUED £2.PTR-FILS-ELEVES ,

    SET £3 £2.PTR-FILS-ELEVES ,

    LOOPWHILE VALUED £3.ELEVE-PT ,

      FETCH £3 1 ,

      IF VALUED £3.PTR-FRERE-CLASSES ,

        SET £3 £3.PTR-FRERE-CLASSES ,

      ELSE ,

        EXITLOOP ,

      ENDIF ;

    ENDLOOP ;

  ENDIF ;

  IF VALUED £2.PTR-FRERE-PROGRAMMES ,

    SET £2 £2.PTR-FRERE-PROGRAMMES ,

  ELSE ,

    EXITLOOP ;

  ENDIF ;

ENDLOOP ;
FREE ALL ;
ENDMACRO ;
```

# AUTORISATION de SOUTENANCE

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

J le rapport de présentation de Monsieur le Professeur M. ADIBA

**Monsieur Marc BURNIER**

t autorisé à présenter une thèse en soutenance pour l'obtention du titre de
OCTEUR de TROISIEME CYCLE, spécialité "Informatique".

Fait à Grenoble, le 10 septembre 1984

Le Président de l'I.N.P.-G