MODELES, LANGAGES ET SYSTEME POUR LA CONCEPTION ET L'EXPLOITATION DE BASES DE DONNEES

B. LE CHARLIER et J.L. HAINAUT(*)

Résumé : Ce rapport expose quelques résultats obtenus dans la cadre d'une recherche en matière de conception et d'utilisation de bases de données. Les différents outils mis au point s'inscrivent dans un schéma hiérarchisé de niveaux d'indépendance ; ils abordent un certain nombre de problèmes qui vont de la définition de modèles théoriques jusqu'à la génération automatique de maquettes en passant par la définition de langages et la réalisation d'un système de gestion de bases de données.

Mots Clés : Base de données ; système de gestion de bases de données, modèles binaires ; langages de description de données ; langage de manipulation de données.

Les auteurs de cet article ne sont que les interprètes de l'équipe Banque de Données qui a réalisé le projet qu'ils décrivent (**) : F. Bodart (responsable), S. Colussi (1976-1977), C. Deheneffe (1971-1977), D. Guillebaud (1977), J.L. Hainaut (1971-1978), H. Hennebert (1971-1976), B. Le Charlier (1973-1978), W. Paulus (1973-1977).

(*) Institut d'Informatique des Facultés universitaires Notre-Dame de la Paix, rue Grandgagnage, 21, 5000 Namur (Belgique).
(**) Projet subventionné par la C.I.P.S. I.2/15.

1. INTRODUCTION

L'un des aspects essentiels de la problématique des bases de données est l'écart considérable qui existe entre d'une part, les besoins des concepteurs et des utilisateurs de disposer de données stables, décrites simplement en fonction des problèmes à résoudre et d'autre part, les contraintes technologiques imposant des solutions pratiques biaisant la structure des données enregistrées, et susceptible d'évoluer rapidement au gré des changements des besoins en performances. De là est née très tôt l'idée d'isoler la description logique des informations de sa mise en oeuvre sur un ordinateur. On est ainsi amené à distinguer dans une base de données un certain nombre de niveaux de description emboîtés tels qu'un niveau enrichit la description du niveau supérieur de considérations de nature plus technique. L'indépendance d'un niveau par rapport aux niveaux inférieurs assure à l'ensemble une grande stabilité vis-à-vis des choix techniques et de leur modification.

Des travaux de recherche actuels se dégagent actuellement quatre niveaux (voir (10) et (11)) :
- *le niveau conceptuel ou fonctionnel* offre une vue des informations aussi stable que l'organisation qu'elles décrivent ;
- *le niveau des accès logiques* se propose de faire apparaître les accès entre les informations permettant d'assurer l'exécution efficace des applications prévues ;
- *le niveau des structures physiques* permet de fixer les paramètres techniques des méthodes d'accès, les paramètres de stockage liés aux supports, etc. ;
- *le niveau externe* propose aux différents utilisateurs des vues particulières extraites de la description conceptuelle.

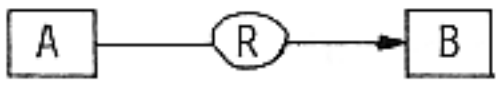
Cette décomposition permet de clarifier les interactions entre les différents éléments constitutifs d'un ensemble intégré d'informations, tant lors de la conception d'une base de données que lors de la construction de systèmes de gestion.

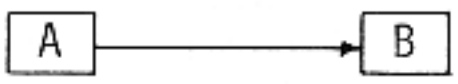
Le présent rapport décrit l'état actuel d'une recherche qui s'inscrit dans ce cadre de référence. Les outils mis au point relèvent à la fois de la modélisation (un modèle conceptuel : le MSI, et un modèle d'accès), des langages opérationnels (langages de description de données, de manipulation pour programmeur et d'interrogation pour utilisateur non informaticien) et de l'aide à la mise au point de bases de données.

2. MODELE DE STRUCTURE DE L'INFORMATION

Destinées à décrire une organisation indépendamment de toute contrainte technique, les informations doivent être structurées de manière à représenter les propriétés intrinsèques de l'organisation et elles seulement. Une approche classique consiste à donner aux informations une structure proche de celle d'une certaine formalisation de l'organisation. Ainsi, s'il est possible de modéliser l'organisation sous forme d'un ensemble d'éléments qui peuvent être groupés en classes et associés deux à deux, il est assez naturel de représenter un élément par une information élémentaire et une association entre deux éléments par une association entre les informations qui les représentent. Groupant en classes informations et associations entre informations, nous sommes amenés à adopter un *modèle binaire* des informations dans lequel nous appellerons OBJET une classe d'informations et RELATION une classe d'associations binaires.

Une relation est définie d'un objet, appelé *origine* , vers un autre, appelé *cible* ; si R est une relation de A vers B, nous la noterons : R(A,B). Cette orientation n'a d'ailleurs d'autre fonction que de distinguer les deux domaines, contrairement à ce qui sera défini dans le modèle d'accès. De manière à alléger la notation, nous admettons les relations dites "sans-nom", notées simplement : (A,B). Nous représenterons graphiquement objets et relations de la manière suivante :

R(A,B) : 

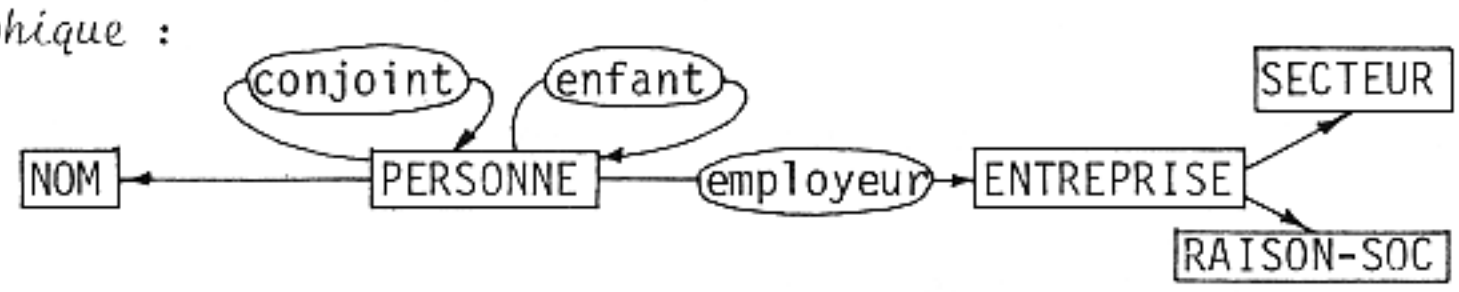
(A,B) : 

Si $a \in A$, $b \in B$ et $(a,b) \in R$, nous dirons que a est une réalisation de l'objet A, b une réalisation de B et (a,b) une occurrence de la relation R.

Parmi les nombreuses propriétés d'une relation qu'il pourrait être utile de spécifier, nous ne retiendrons que celles qui décrivent le nombre de réalisations d'un objet que l'on peut (ou doit) associer par la relation à chaque réalisation de l'autre objet ; nous les exprimerons par les caractéristiques de cardinalité qui, pour une relation R(A,B), sont notées I_R-J_R, K_R-L_R et s'interprètent comme suit : à tout instant, pour tout $a \in A$, le nombre d'occurrences de R dont a est origine est au minimum I_R et au maximum J_R ; à tout instant, pour tout $b \in B$, le nombre d'occurrences de R dont b est cible est au minimum K_R et au maximum L_R.
Soit, par exemple, un système constitué de personnes ayant un nom, un employeur, un conjoint et des enfants, d'entreprises appartenant à un secteur d'activité et dotées d'une raison sociale. Après avoir isolé les classes d'éléments et d'associations, on construira le modèle binaire suivant : .

objets :	PERSONNE, NOM, ENTREPRISE, SECTEUR, RAISON-SOC		
relations :		(PERSONNE, NOM)	: 1-1, 0-∞
	conjoint	(PERSONNE, PERSONNE)	: 0-1, 0-1
	enfant	(PERSONNE, PERSONNE)	: 0-30, 0-2
	employeur	(PERSONNE, ENTREPRISE)	: 1-1, 0-∞
		(ENTREPRISE, SECTEUR)	: 0-∞, 1-∞
		(ENTREPRISE, RAISON-SOC)	: 1-1, 0-1

Représentation graphique :



D'une manière générale des renseignements sur une réalisation sont obtenus grâce aux autres réalisations qui lui sont reliées ; une structure aussi élémentaire que celle de l'exemple précédent peut engendrer une qualité quasi illimitée d'informations sur une personne, une entreprise ou un secteur ; on y trouve par exemple le nombre moyen d'enfants mariés des personnes qui travaillent dans le même secteur d'activité qu'une personne déterminée.

Il y a cependant dans l'organisation des éléments que l'on juge tellement simples qu'ils peuvent être décrits complètement par une valeur unique ; un objet dont les réalisations sont des valeurs s'appellera *objet élémentaire* ; tel serait le cas d'objets décrivant des noms, des prix ou des dates. D'autres éléments par contre ne peuvent être décrits de façon satisfaisante que par leur association à d'autres éléments de l'organisation : une personne est un élément défini comme ayant un nom, un conjoint, des enfants et un employeur ; s'il n'est pas possible de donner d'une personne une description intrinsèque complète sous forme d'une valeur, nous lui associerons dans le modèle une réalisation d'un *objet complexe*. Une telle réalisation apporte par elle-même peu d'informations, hormis son existence et sa discernabilité.

Conclusion : Membre de la famille des modèles relationnels binaires (voir aussi ABRIAL (12), BRACCHI (13) et SENKO (14)) le MSI propose au concepteur un nombre réduit de concepts lui permettant de donner aux informations une structure proche de celle du système à décrire lorsque celui-ci peut se réduire en un ensemble d'éléments associés deux à deux. Ce mode de description permet également d'exprimer à la fois une vue relationnelle n-aire et une vue en réseau (Type CODASYL) d'un ensemble d'informations ainsi que la correspondance entre celles-ci (cfr. (7)).

3. LE MODELE D'ACCES

Le modèle d'accès que nous allons décrire est du niveau de l'implémentation logique : il permet de définir la structure logique globale des accès ainsi que celles du niveau externe (appelées souvent sous-schemas). A ce niveau, un modèle de données doit permettre la définition de types d'informations et de chemins d'accès aux informations, mais sans fournir aucune indication sur la façon dont ces concepts sont réalisés physiquement. Il doit fournir également des primitives permettant de créer, rechercher, modifier ou supprimer les informations contenues dans la base de données. La modèle d'accès utilise des concepts et une terminologie proches de ceux du MSI ; en effet, un modèle binaire constitue un outil privilégié de description de chemins d'accès (un chemin a une origine et un but) ; ces modèles sont cependant différents puisque les univers qu'ils décrivent sont respectivement un système réel et des données enregistrées.

3.1. Les types d'information ou objets

(i) Objets élémentaires. Un tel objet est un type d'informations dont les réalisations sont des *chaînes finies de caractères*. Le domaine de valeurs d'un *objet élémentaire simple* est défini de la manière habituelle (type numérique ou alphanumérique, longueur, ...) ; tel serait le cas des objets SIGLE et VITESSE (voir annexe). Par contre, les valeurs d'un *objet élémentaire composé* sont constituées de la concaténation de valeurs d'autres objets élémentaires ; une valeur de l'objet NUMERO, par exemple, est constituée de la concaténation d'une valeur de GAMME et d'une valeur de MODELE.

(ii) Objets complexes. Cette notion correspond à une idéalisation de la notion d'enregistrement, abstraction faite de son contenu et de son appartenance à divers chemins d'accès. Une réalisation d'un objet complexe n'apporte par elle-même d'autre renseignement que son *existence* et sa *discernabilité* ; elle pourra être créée et supprimée. Exemples : CONSTRUCTEUR, CPU, MEMOIRE.

(iii) Objet racine. Tout schéma de base de données contient un et un seul *objet racine*, représentant la base de données elle-même ; son unique réalisation est un point d'entrée privilégié (accès séquentiel). Exemple : CONFIGURATION.

3.2. Les relations d'accès

La plupart des méthodes d'accès disponibles peuvent se répartir en quatre catégories : accès séquentiel aux enregistrements d'un même type, accès à un ensemble d'enregistrements sur base d'une valeur de clé, accès à un ensemble d'enregistrements à partir d'un autre enregistrement, accès au contenu d'un enregistrement.
Formalisant les méthodes par des relations binaires, nous pouvons définir la relation d'accès. Une *relation d'accès* est un type de données défini à partir d'un couple d'objets appelés *origine et cible de la relation*. A tout couple (o,c) de réalisations de l'origine et de la cible correspond au plus une *occurrence* de la relation d'accès *exprimant l'existence d'un moyen d'atteindre c, à partir de o* (en utilisant la primitive d'accès), o et c sont appelés *origine et cible de l'occurrence*.
Nous décrirons une relation d'accès par trois propriétés logiques :

(i) Caractéristiques de cardinalité. De même que dans le MSI, nous associerons à toute relation un quadruplet noté : $R(O,C) : I_R\text{-}J_R,\ K_R\text{-}L_R$.

(ii) Relations d'accès inverses. Deux relations R(O,C) et S(C,O) seront dites *inverses* si et seulement si il correspond à chaque instant à toute occurrence (o,c) de la première, une occurrence (c,o) de la seconde. Toute relation possède au plus une inverse et on utilisera la notation $R(O,C) = S(C,O)^{-1}$.

(iii) Séquence d'accès aux réalisations cibles. Si, pour les applications, on a intérêt à traiter suivant une certaine séquence, l'ensemble des réalisations cibles qu'on peut atteindre à partir de chaque réalisation origine, on peut joindre à la définition de la relation la règle qui détermine cette séquence.

(ordre chronologique de création, ordre déterminé par les valeurs d'un objet élémentaire, etc.)

3.3. Classes de relations

Nous limitant à l'univers à représenter (les méthodes d'accès), nous retiendrons :

(i) les relations (racine, objet complexe), qui représentent la possibilité d'un accès séquentiel aux réalisations de l'objet complexe. La relation (CONFIGURATION, CPU), par exemple, est du type 0-∞, 1-1.

(ii) les relations (objet complexe, objet élémentaire), qui représentent la possibilité d'accéder au "contenu d'un enregistrement". Exemples : (CONSTRUCTEUR, SIGLE), (CPU, NUMERO).

(iii) les relations (objet élémentaire, objet complexe), qui spécifient que cet objet élémentaire est une "clé d'accès" aux réalisations de l'objet complexe. Exemple : (NUMERO, CPU).

(iv) les relations (objet complexe, objet complexe), qui représentent un mécanisme d'accès à des réalisations d'objets complexes à partir d'une réalisation d'un objet complexe. Une telle relation peut être symétrique, comme la relation COMPATIBLE (PERIPHERIQUE, PERIPHERIQUE) : 0-∞, 0-∞.

3.4. Les primitives

Tout traitement sur une base de données peut être décomposé en une séquence d'actions primitives conservant la cohérence des données.

(i) La primitive d'accès. La notion de relation d'accès permet de réduire à un le nombre de primitives nécessaires à la "recherche de l'information". *L'application de la primitive d'accès à une réalisation o de l'origine d'une relation R(O,C) fournit pour résultat l'ensemble ordonné des réalisations c cibles d'une occurrence (o,c) de la relation.* Les réalisations que l'on peut connaître a priori sont les valeurs d'objets élémentaires et l'unique réalisation de la racine. Elles servent de point de départ à toute recherche d'information dans la base de données.

(ii) Les primitives de mise à jour. Le modèle d'accès fournit 5 primitives permettant de modifier le contenu d'une base de données : *création et suppression d'une réalisation d'objet complexe ; création, modification et suppression d'une occurrence de relation.* Ces actions doivent respecter les règles de cohérence associées aux relations inverses et aux I-J, K-L : elles sont donc plus complexes que des actions ponctuelles. On trouvera dans (8) ou (9) une étude des primitives de mise à jour et de certaines règles de cohérence supplémentaires assurant leur validité.

4. LDD : UN LANGAGE DE DESCRIPTION DE DONNEES ASSOCIE AU MODELE D'ACCES

Langage de déclaration des objets et des relations d'accès d'une base de données, le LDD doit jouir de la simplicité et de l'indépendance du modèle d'accès lui-même. La syntaxe choisie est proche de celle du DDL CODASYL.

La description LDD d'une base de données contient dans l'ordre : la déclaration de l'objet racine, les déclarations des objets complexes et des objets élémentaires associés, et les déclarations des relations entre objets complexes. Un examen attentif révèlera que le modèle que décrit un texte DDL est légèrement différent (plus restreint) que le modèle d'accès qui vient d'être décrit ; la raison en est une plus grande simplicité de réalisation.

Objet racine. La phrase ROOT déclare un nouvel objet racine.

Objet complexe. Dans la phrase COMPLEX-OBJECT, qui déclare un nouvel objet complexe, seront incluses les déclarations des objets élémentaires, s'il en existe, directement reliés à cet objet.

Objet élémentaire. Un objet élémentaire simple est décrit par le type de ses valeurs (d'une manière analogue au COBOL) ; un objet élémentaire composé est décrit par la liste des descriptions de ses composants. Chaque description peut contenir une clause de répétitivité spécifiant les nombres minimum et maximum de valeurs de cet objet pouvant apparaître (OCCURS) ; les différents niveaux de décomposition sont indiqués par un numéro de niveau précédant la description de l'objet. Parmi les objets élémentaires de niveau 1, associés à un même objet complexe, un seul peut être déclaré identifiant mais plusieurs peuvent être déclarés clé d'accès.

Relations entre objets complexes. La déclaration spécifiera son nom (ou le fait qu'elle est sans nom) les noms de l'origine et de la cible, les caractéristiques I-J, K-L ainsi que l'existence et le nom d'une relation inverse.

Exemple
Le MSI donné en exemple précédemment,et auquel on aurait ajouté les objets élémentaires ADRESSE et PRENOM, pourrait être à l'origine du modèle d'accès décrit par le texte DDL suivant :

```
ROOT NAME IS D-B-SOC.
COMPLEX-OBJECT NAME IS PERSONNE ;
ELEMENTARY-OBJECT NAMES ARE :
   1 NOM PIC X(30) ACCESS-KEY,
   1 PRENOM OCCURS (1,5) PIC X(15),
   1 ADRESSE OCCURS (0,1),
      2 LOCALITE PIC X(30),
      2 RUE      PIC X(30),
      2 NUMERO   PIC 9(5).

COMPLEX-OBJECT NAME IS ENTREPRISE ;
E-O NAMES ARE :
   1 RAISON-SOC PIC X(25) IDENTIFIER,
   1 SECTEUR PIC X(40) ACCESS-KEY.

RELATION NAME IS ENFANT FROM (0,2) PERSONNE TO (0,∞) PERSONNE.
RELATION NAME IS CONJOINT FROM (0,1) PERSONNE TO (0,1) PERSONNE IN-
   VERTED BY CONJOINT.
RELATION NAME IS EMPLOYEUR FROM (0,∞) PERSONNE TO (1,∞) ENTREPRISE
   INVERTED BY EMPLOYE.
END.
```

5. LMD : UN LANGAGE DE MANIPULATION DE DONNEES

Ce langage devait permettre la rédaction de programmes d'accès et de modification clairs, bien structurés et aisés à lire. Il devait aussi, tout en libérant le programmeur de contraintes délicates ou fastidieuses (gestion des codes de retour, gestion des "courants", et initialisation de paramètres implicites chez CODASYL), lui laisser, si celui-ci le désire, le contrôle de l'efficacité des accès. Nous n'aborderons ici que l'un des mécanismes fondamentaux du LMD : la boucle d'accès.

A toute boucle d'accès est associée une variable prenant pour valeurs les différentes réalisations obtenues lors des accès. A cette variable correspond à tout instant une réalisation courante, connue seulement dans le corps de la boucle, lequel constitue la portée ou *contexte* de cette variable. Ainsi, si la variable de nom CONSTR correspond à une réalisation de CONSTRUCTEUR, on pourra écrire, dans le contexte de CONSTR, la boucle suivante commandant l'accès aux réalisations décrivant les CPU livrés par le constructeur "CONSTR" et dont la vitesse est inférieure à 500 nsec (cfr. annexe) :

```
@REACH CPU = C1 FROM CONSTR VIA LIVRE IF (VITESSE < 500) GET(NUMERO,MODELE).
                    ‹corps de la boucle C1›
@END C1.
```

L'ordre d'accès spécifie, explicitement ou non, l'objet cible, la variable de boucle, la réalisation origine, la relation à emprunter, des critères restrictifs éventuels, et enfin l'accès simultané à des valeurs d'objets élémentaires (clause GET) ; ces dernières, liées à la variable de boucle, seront nommées selon les modes de désignation en vigueur dans le langage hôte (p. ex. : MODELE OF C1 dans le cas du COBOL) ; elles seront indéterminées en dehors du contexte de C1. Le corps d'une boucle est constitué d'un nombre quelconque d'instructions du langage hôte ou du LMD, et en particulier de boucles d'accès. La boucle REACH-END permettant l'accès aux réalisations d'un objet complexe, on utilisera une boucle d'accès OPEN-CLOSE pour accéder à l'unique réalisation d'un objet racine que constitue une base de données.

A titre d'exemple, écrivons l'architecture d'un programme COBOL qui établit, par CPU du constructeur désigné par la variable IDENT, et pour chaque périphérique supporté par ce CPU, la liste des périphériques qui lui sont compatibles ainsi que leur fournisseur. Signalons au préalable que l'accès par une relation sans nom à partir d'un objet complexe ou d'une racine se signale par l'absence du paramètre VIA, que la clause FROM peut être omise si elle fait référence à la variable de la boucle du niveau le plus proche, et enfin, qu'un ordre CLOSE clôture les boucles de son contexte qui n'auraient pas encore été clôturées.

```
@OPEN CONFIGUR = CONF.
   @REACH CONSTRUCTEUR = C1 IF (SIGLE = IDENT).
      @REACH CPU = CP  VIA LIVRE GET (NUMERO).
           DISPLAY NUMERO OF CP UPON TERMINAL.
           @REACH PERIPHERIQUE = P1 GET ALL.
                @REACH PERIPHERIQUE = P2 VIA COMPATIBLE GET ALL.
                   @REACH CONSTRUCTEUR = C2 GET (NUMERO).
                       DISPLAY MODELE OF P1, '=', MODELE OF P2, 'DE',
                       SIGLE OF C2 UPON TERMINAL.
@CLOSE.
```

On observe que dans ce programme, tout accès quel qu'il soit se commande par une phrase de même syntaxe. D'autre part, le programmeur peut disposer à certains endroits du programme d'un nombre quelconque de "courants" d'un même objet complexe ; tel est le cas du contexte C2 où deux réalisations de CONSTRUCTEUR et deux réalisations de PERIPHERIQUE sont connues simultanément. On observera également que ce programme n'est pas encombré de contrôles de codes de retour ; ceux-ci sont effectués automatiquement sauf dans les conditions anormales, où le programmeur pourra reprendre l'initiative. Enfin, la structure rigide d'un programme permet au compilateur d'anticiper dans une large mesure son comportement et de prévoir certaines optimisations, et une gestion automatique de la concurrence et des points de reprises.

La rigidité de la structure arborescente d'un programme peut aisément être assouplie grâce à l'usage des modificateurs dynamiques NEXT ‹var.boucle› et EXIT ‹var.boucle›, à celui des variables de travail constituant une référence à une réalisation, et à l'ordre ALLOCATE d'extension de portée d'une variable de boucle (voir référence (8)).

6. NUL : UN LANGAGE INTERACTIF DE CONSULTATION D'UNE BASE DE DONNEES

Le langage interactif NUL (Navigational User's Language) (voir (5) et (8)) permet à un utilisateur non spécialiste de construire des requêtes pas à pas et d'extraire d'une base de données des ensembles hiérarchisés de valeurs. Le système NUL dirige le dialogue ; il aide l'utilisateur à formuler ses requêtes (contrôle des messages, rappel de la syntaxe, édition sélective de schémas de bases de données, ...) et permet les modifications de paramètres standards

(milieu de sortie, longueur de ligne, format d'impression, ...).

Dans un premier temps, l'utilisateur construit un contexte, c'est-à-dire qu'il sélectionne un ensemble arborescent de réalisations d'objets complexes. Dans un second temps, il peut, par une série de commandes d'impression, obtenir des ensembles de valeurs d'objets élémentaires associées aux réalisations.

Définition d'un contexte. Après ouverture d'une base de données, l'utilisateur introduit une série de définitions d'un contexte, étiquetées S01, S02, ... D'une manière analogue au LMD, chaque définition sélectionne un sous-ensemble (qui portera le nom de l'étiquette) de réalisations d'un objet complexe selon deux mécanismes : à partir de la racine, ou du sous-ensemble déjà défini et donc étiqueté, on sélectionne, via une relation, un sous-ensemble de réalisations cibles ; ce sous-ensemble peut être réduit par la spécification d'une condition ou produit logique de conditions portant sur les valeurs d'objets élémentaires ou sur la liaison à certaines réalisations d'objets complexes. Supposons, par exemple, que S01 désigne un sous-ensemble de CONSTRUCTEUR. La définition du sous-ensemble S02 de CPU correspondant aux CPU livrés par le constructeur décrit par S01 et dont le prix est supérieur à 100000 s'écrira :

S02 <- For S01 BY LIVRE CPU SUCH THAT PRIX > 100000 ;

Dans l'exemple suivant, plus complet, on construit un contexte correspondant à la requête suivante : "Sélectionner tous les constructeurs ; pour chacun d'eux, sélectionner leurs CPU dont tous les périphériques possèdent au moins 2 compatibles ; pour ces CPU sélectionner leurs mémoires à tores de ferrite".

ENTER DATA BASE NAME/HALT = CONFIGURATION.
S01 <- CONSTRUCTEUR ;
S02 <- FOR S01 BY LIVRE CPU SUCH THAT HAVE ALL PERIPHERIQUE = X01 ;
X01... SUCH THAT BY COMPATIBLE HAVE AT LEAST 2 PERIPHERIQUE ;
S03 <- FOR S02 MEMOIRE SUCH THAT TYPE = 'TORES' ;
S04 <- END.

N.B. : le texte en italique est affiché par le système NUL.

On observera que l'association d'une condition restrictive sur PERIPHERIQUE de S02 est définie grâce au qualificateur "=X01" dont NUL réclame par le suite le développement. Cette méthode de construction évite l'emploi de parenthèses et permet la mise au point du contexte niveau par niveau.

Impression d'une hiérarchie extraite du contexte. Le contexte détermine une ou plusieurs hiérarchies de réalisations, issues de la racine. La commande PRINT permet d'imprimer les valeurs d'objets élémentaires associés à un sous-ensemble arborescent d'une hiérarchie du contexte. Les parties de ce sous-ensemble, appelé lui-même hiérarchie d'impression, sont désignées, dans la commande, par les étiquettes S_i et sa structure arborescente est établie par l'utilisation de parenthèses. L'expression S_i.⟨nom-OE⟩ signifie que, pour chaque réalisation S_i, on veut seulement imprimer les valeurs de l'objet élémentaire ⟨nom-OE⟩.
Exemple (relatif au contexte ci-dessus) :
"Imprimer les CPU et pour chacun d'eux, le nom de leur constructeur et le prix de chaque mémoire".

ENTER COMMAND - print S02 (S03.PRIX), S01.NOM.

7. SPHINX : SYSTEME DE BASE DE DONNEES BASE SUR LE MODELE D'ACCES

La réalisation d'un système de base de données suppose la définition d'un mode de représentation physique des concepts du modèle choisi. Cette représentation s'exprime, au niveau le plus bas, en termes de concepts d'un modèle plus primitif tel que celui qui est sous-jacent aux techniques d'accès physique, aux méthodes d'accès du système d'exploitation ou même à un autre système de gestion de bases de données. Se pose alors le problème de la correspondance entre les deux niveaux, et plus particulièrement de la traduction d'un texte LDD en une

description relative au niveau inférieur, et de la réalisation d'une expression LMD ou NUL par l'exécution d'une combinaison de primitives du niveau inférieur. La difficulté de réaliser cette correspondance est fonction de "l'éloignement" des concepts des deux niveaux.

Le système choisi pour la réalisation de SPHINX est le système SESAM de SIEMENS. Conçu pour gérer une collection de bases de données dans toutes ses fonctions essentielles SESAM présente les données sous une forme très libre et peu structurée (format du type relationnel ; accès séquentiel, indexé, direct et inversé).
La correspondance entre les descriptions des données est assurée par un *compilateur LDD*. Les liaisons entre les programmes LMD sont moins simples cependant. Il est possible en effet de réaliser une correspondance *compilée* ou une correspondance *interprétée*. La première solution, qui avait été adoptée pour le prototype réalisé en 1974, consiste à traduire tout ordre LMD en une séquence d'appels à SESAM ; cette technique conduit à générer des programmes d'applications assez volumineux (mais rapides) mais surtout elle rend impossible une gestion satisfaisante de la concurrence, SESAM étant insensible aux concepts du modèle d'accès. La deuxième solution, qui a conduit à la réalisation du second prototype (8), consiste à établir une communication entre chaque programme d'application (et occurrence de l'interpréteur NUL) et un noyau central qui exécute les tâches demandées par ces programmes. Connaissant à la fois la structure des programmes qui dialoguent simultanément avec lui et les descriptions des bases de données selon le modèle d'accès et selon SESAM, ce noyau est capable d'interpréter les commandes en termes d'opérations SESAM et de contrôler strictement les programmes de manière à en assurer l'exécution concurrente sans interblocage.

Le travail sur les bases de données peut donc être assuré par :
- le noyau qui communique avec les processus demandeurs et avec le processus fournisseur SESAM ;
- le compilateur LMD générant des programmes d'application en COBOL pur ;
- l'interpréteur NUL dont une copie peut dialoguer avec un terminal.

A ces composants de base s'ajoutent des utilitaires de restauration, de sauvetage, et de réorganisation. Une dernière fonction intéressante de SPHINX est l'existence d'un dictionnaire de données décrit lui-même selon le modèle d'accès (par un *métaschéma*) et qui permet la mise au point conversationnelle de schémas, puis la génération automatique de bases de données et de programmes de chargement conversationnels. Associée à NUL, cette fonction constitue un outil très simple de constitution de prototype ; les programmes sont réalisés en DML et sont donc aisément extensibles.

8. CONCLUSIONS

Les modèles et les langages sont, malgré leur puissance, d'une extrême simplicité, essentiellement grâce à l'application des principes de niveaux de description indépendants. Si bon nombre de principes adoptés existent par ailleurs (décomposition en niveaux (10), (14), (17), modèles binaires (12), (13), (14), langage de boucles (15), (16), dictionnaires de données et maquettes (19) et IMS, langages interactifs, ...), il semble qu'ils ont rarement été intégrés de cette manière dans une démarche à la fois théorique et pragmatique assurant à l'ensemble une grande simplicité et une grande cohérence.

A l'heure actuelle, les travaux se poursuivent dans trois directions :
- mise au point d'un LMD interactif portable acceptant toute base de données CODASYL ;
- étude probabiliste de programmes d'application ((6), (7)) ;
- principes généraux d'indépendance dans les correspondances entre structures.

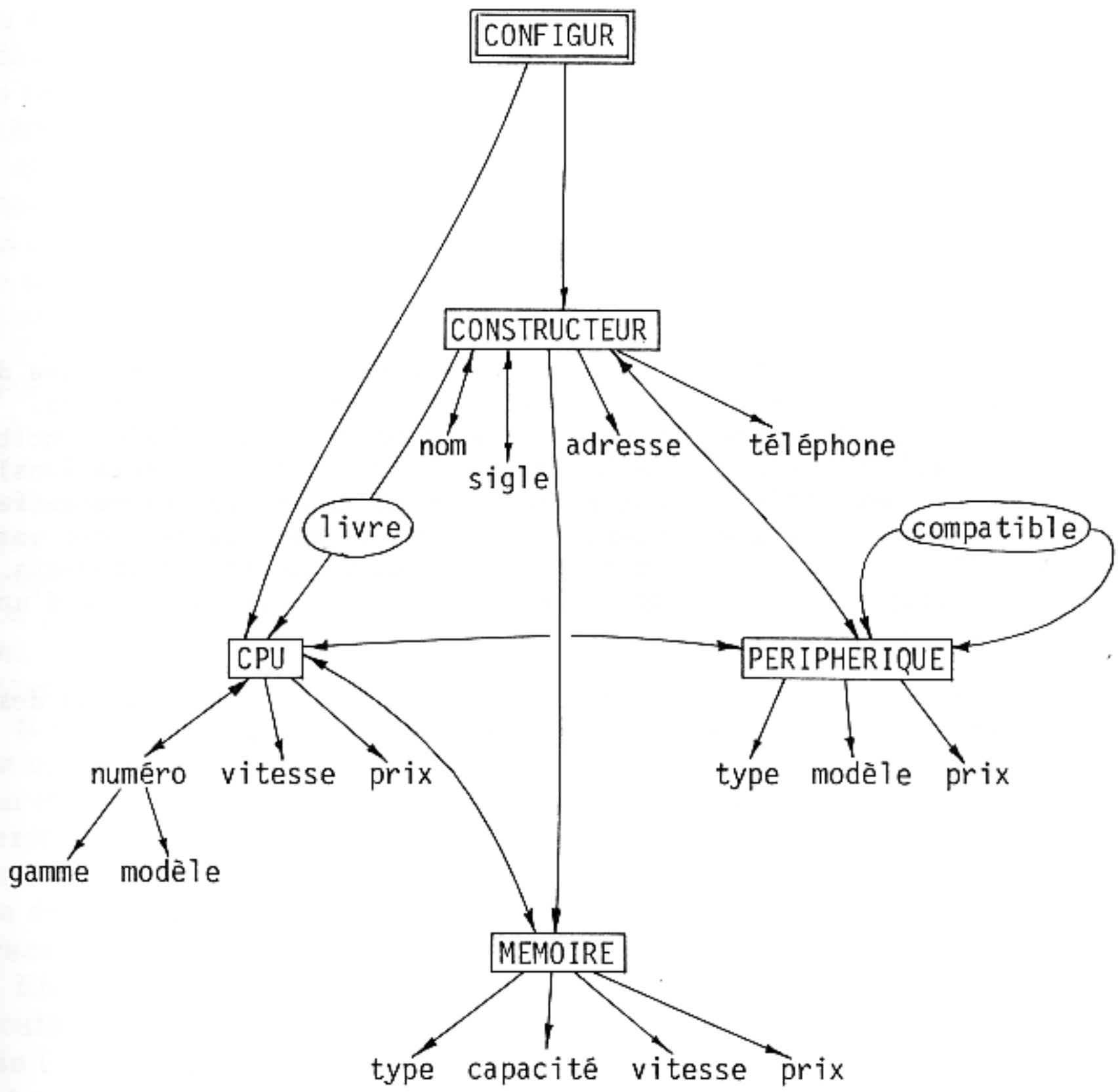
9. BIBLIOGRAPHIE ET REFERENCES

(1) HAINAUT, J.L. : *A Semantic Formal Model of an Implemented Data Base and*

its Data Manipulation Language, Institut d'Informatique de Namur, July 73.
(2) HAINAUT, J.L. and LE CHARLIER, B. : *Présentation et spécification du modèle d'accès et du langage rigide*, Internal Papers, Institut d'Informatique, Feb. 1974.
(3) HAINAUT, J.L. and LE CHARLIER, B. : *An Extensible Semantic Model of Data Base and its Data Language*, Proc. IFIP Congress 1974, North-Holland Publish. Co, 1974.
(4) DEHENEFFE, HENNEBERT, PAULUS : *Relational Model for Data Base*, Proc. IFIP Congress 1974, North-Holland Publish Co, 1974.
(5) DEHENEFFE, Cl., HENNEBERT, H. : *NUL, a Navigational User's Language for a Network Structured Data Base*, Proc. ACM Sigmod Conference 1976, Washington, ACM Publish.
(6) HAINAUT, J.L. : *Evaluation des performances d'une base de données par modèle probabiliste*, Proc. Séminaire INFORSID, Grenoble, 1976 - IRIA.
(7) HAINAUT, J.L. : *Some Tools for Data Independence in Multilevel Data Systems*, in "Architecture and Models in Data Base Management Systems" - Nice 1977, North-Holland Publish. Co.
(8) EQUIPE BASE DE DONNEES : *Système de conception et d'exploitation de bases de données, première partie : Modèles et langages*, Rapport final du projet C.I.P.S. I.2/15, Institut d'Informatique, février 1978.
(9) LALOUX, J.P. : *Implémentation du modèle d'accès par lui-même*. Mémoire de fin d'études, Institut d'Informatique de Namur, juin 1975.
(10) *Interim Report ANSI/X3/SPARC Study Group on Data Base Management Systems*, ANSI, Feb. 1975.
(11) *Introductive Report*, Proc. Intern. Workshop on Data Structure Models. Presses Universitaires de Namur, 1975.
(12) ABRIAL, J.R. : *Data Semantics*, in "Data Base Management", IFIP WC. 1974, North-Holland Publish. Co, 1974.
(13) BRACCHI, PAOLINI, PELAGATTI : *Binary Logical Association in Data Modelling*, Internal Report n° 75-12, Sept. 75, Ist. Elettrotecnica et Elettronica del Politecnico di Milano.
(14) SENKO, M.E. : *Data Description Language in the Concept of a Multilevel Structured Description : DIAM II with FORAL*, in "Data Base Description", IFIP WC 1975, North-Holland Publish. Co., 1975.
(15) BACHMAN, Ch., BREWER, S. : *A Direct Access System with Procedurally Generated Virtual Data Structuring Capability*, Honeywell Comp. Journal, 1974, Vol. 8, n° 1.
(16) ABRIAL, BAS, BEAUME, HENNERON, MORIN, VIGLIANO : *Projet Socrate*, Institut de Mathématique Appliquée de Grenoble, 1970.
(17) CABANES, A. : *Rapport Indépendance dans les SGBD*, Mars, 1977, Institut d'Informatique d'Entreprise, C.N.A.M., Paris.
(18) CODASYL : *Data Base Task Group 1971 Report*, ACM, 1971.
(19) PECCOUD, F. : *Projet MAXI-P*, Université de Grenoble, 1977.
(20) *SIEMENS-SYSTEM 4004/7000 - SESAM (BS2000)*, 1976.

°

° °

ANNEXE

MODELE D'ACCES D'UNE BASE DE DONNEES DE CHOIX D'UNE CONFIGURATION