Systèmes d'Aide à la Décision : une approche méthodologique intégrée pour l'utilisateur final

Jean-Luc Hainaut
Institut d'Informatique
Facultés Universitaires de Namur
rue Grandgagnage, 21 B-5000 Namur (Belgique)
tél. (81) 72.49.96

Résumé. L'article propose un modèle de spécification de bases de connaissances pour l'aide à la décision. Ce modèle doit être accessible à l'utilisateur final et pas seulement au développeur professionnel. Il doit permettre une implantation simple et aisée dans des outils largement répandus tels que les tableurs, les gestionnaires de données et les petits systèmes de règles de production. La base de connaissances est décomposée en base de données, spécifiée par son schéma conceptuel Entité/Association, et par une base de règles qui décrit des grandeurs dérivables à partir de données externes, issues notamment de la base de données. L'article se concentre plus particulièrement sur l'expression de la base de règles, constituée de la spécification de grandeurs et de leurs règles de définition. On décrit successivement les grandeurs simples, puis les grandeurs dimensionnées (indicées à l'aide d'une grandeur simple), enfin les grandeurs qui sont des objets de la base de données. On propose également une notation destinée à l'écriture des grandeurs et des règles. L'exposé suit une progression pédagogique parallèle à l'expérience que pourrait avoir un utilisateur développeur, plutôt qu'un développement axiomatique.

Mots-clés Aide à la décision, Base de données, Base de connaissances, Base de règles, Modèle de spécification

Abstract. The paper puts forward a specification model for Decision Support Knowledge Bases. That model is intended for end-users as well as for professional developers. Specifications in this model must be easily mapped into widespread decision support tools such as spreadsheet processors, data managers and small rule-based systems. The knowledge base is split into a data base and a rule base. The data base is described through its E/R conceptual schema, while the rule base consists of a set of variables and a set of rules that tell how to derive the values of the variables from external data (e.g. from the data base). The paper concentrates on expressing the rule base. It develops simple variables and their defining rules, then *dimensioned* variables, which are indexed with simple variable(s), and finally variables which take their values in the data base. A specification language is suggested as well. The text is not an axiomatic presentation of abstract concepts but follows a pedagogical structure that best matches experience of end-users.

Keywords. Decision support, Data base, Knowledge base, Rule base, Specification model

INFORSID 90 BIARRITZ 15-18 MAI 90 Pages: 7-34 Editeur: INFORSID

1. INTRODUCTION

Le domaine de l'aide à la décision est en continuelle évolution particulièrement en raison de la disponibilité sur ordinateurs personnels d'outils qui permettent à l'utilisateur final de réaliser lui-même des applications d'une certaine complexité. Dans les années 80, les deux composants essentiels de ces outils ont été le **tableur** et le **gestionnaire de données** locales.

Les raisons du succès des tableurs sont bien connues [RONEN,89] : programmation visuelle et manipulation directe, langage non procédural, similitude entre modèle de l'outil et mode d'expression des problèmes par l'utilisateur. Outils faiblement couplés au départ, tableurs et gestionnaires de données ont progressivement vu ce couplage consolidé. Souvent même, leurs fonctions ont été **intégrées** pour offrir une synergie maximum (K-Man de MDBS, SQL*CALC d'ORACLE, EXCEL+Serveur SQL). S'y ajoutent parfois des composants procéduraux et déductifs, comme dans GURU de MDBS.

L'intégration s'est également développée selon un autre axe, qui n'est pas seulement technologique : le couplage *local/ central* ou encore *Aide à la Décision/ Informatique opérationnelle*, dont l'aspect le plus populaire et le plus immédiat est l'accès, à partir d'un ordinateur personnel, à la base de données du Système d'Information central.

Si l'état de l'art en matière d'outils de développement d'applications d'aide à la décision évolue très rapidement, on ne peut en dire autant des **aspects méthodologiques**. L'utilisateur-développeur, qui n'est pas un professionnel de l'informatique, est confronté à des logiciels de plus en plus puissants, de plus en plus difficiles à maîtriser, et surtout aborde des problèmes de plus en plus complexes. Au contraire du développeur professionnel de Systèmes d'Information, il ne dispose ni de modèles, ni de méthodes, ni a fortiori d'outils d'aide au développement. La situation est donc proche de celle du programmeur du début des années 60, peu à peu conscient que la maîtrise de FORTRAN ou COBOL ne suffit plus à la construction systématique de programmes fiables et maintenables.
Si les travaux sont nombreux dans de domaine des systèmes d'aide à la décision [DSS:TA,87] [BEULENS,88] [KARAGIANNIS, 87] [VANHEE,88], peu ont été adaptés aux problèmes méthodologiques des développeurs. Citons par exemple [RONEN,89] ou encore [HOLSAPPLE,87] et [KONOPASEK,84], ces derniers cependant dans le cadre de logiciels spécifiques. Tout au plus commence-t-on à poser le problème de la qualité et de la fiabilité des applications [RONEN,89], pour y relever une situation plus que préoccupante. Cette observation contraste avec l'état de l'art et de la pratique dans le développement des Systèmes d'Information, où

modèles, méthodes et outils foisonnent, tant dans les laboratoires que sur le terrain [TARDIEU,83] [BODART,89] [JACKSON,83].

Nous voudrions dans cet article apporter une contribution méthodologique au problème du développement, notamment par des **personnes non professionnelles**, d'applications d'aide à la décision exécutables par les logiciels actuellement disponibles sur ordinateurs personnels.
Nous nous limiterons au problème de la spécification d'applications en proposant des concepts, et en suggérant une notation d'expression, qui permettent de décrire simplement, précisément et complètement une large classe de problèmes d'aide à la décision. Ces concepts sont tout à la fois indépendants des outils opérationnels, et proches de ceux-ci. Ils sont aussi proches du modèle mental de l'utilisateur.
Nous ne proposerons cependant pas de démarche qui guiderait l'utilisateur dans l'expression d'un problème à l'aide de ces concepts (des propositions partielles sont proposées dans [RONEN,89] et [HAINAUT,89]).

D'un point de vue macroscopique, la solution à un problème s'exprime sous la forme d'une **base de connaissances** qui contient les structures pertinentes d'un domaine d'application, sous la forme d'*objets* et de *relation primitives ou d'inférence* entre ceux-ci. Eu égard non seulement à l'état de l'art en matière de modélisation, mais aussi à l'architecture des outils actuels qui induit encore chez l'utilisateur la dichotomie *données/ traitements*, nous décomposerons cette base de connaissances en **base de données** et **base de règles**. Spécifier une base de données sous forme d'un schéma conceptuel, et exprimer ce dernier selon les structures d'un SGBD, sont des problèmes désormais bien maîtrisés [TARDIEU,83] [BODART, 89] [HAINAUT,86]. Nous nous tournerons donc vers le problème de la spécification d'une base de règles, que nous appellerons plus généralement *modèle*, et de son couplage avec le schéma d'une base de données.

Evitant volontairement une approche axiomatique, nous avons préféré un exposé progressif, parallèle à une démarche pédagogique, dans lequel nous aborderons successivement la notion de *modèle simple* (constitué de grandeurs monovaluées et de règles), puis la notion de *modèle dimensionné* (dans lequel grandeurs et règles peuvent être indicées selon une ou plusieurs dimensions du problème), enfin la notion de modèle *attaché à un schéma* de base de données (où des objets du schéma constituent des dimensions du problème). Le sujet aurait pu être abordé selon l'angle d'une extension déductive du schéma d'une base de données. On trouvera une telle approche dans [LAZIMY,89] ET [CHEN,88]. Nous avons cependant écarté cette approche en raison de l'objectif spécifiquement pédagogique des propositions, tant en ce qui concerne l'élaboration des spécifications que de leur expression dans des outils populaires.

Le présent texte suppose de la part du lecteur quelques notions de modélisation de bases de données et quelqu'expérience en matière d'utilisation de tableurs.

2. NOTION DE MODELE SIMPLE

Un modèle simple est constitué d'un ensemble de *grandeurs simples* et de *relations* ou *règles* qui représentent certains aspects d'un système réel, que l'on appellera le *domaine d'application*.

2.1 Grandeurs simples

Une grandeur simple représente une propriété mesurable du domaine d'application. A un instant déterminé, on peut associer à chaque grandeur une **valeur** qui représente l'état de la propriété. L'ensemble des valeurs prises par chaque grandeur représente un état du domaine d'application. Cette valeur sera *numérique* (montant des exportations, quantité en stock, distance), *logique* (vrai/faux, grand/petit, actif/inactif), *temporelle* (date, heure, mois, quinzaine) ou encore *qualitative*, et représentée par un libellé (nom d'un département, marque d'un véhicule). Plus généralement, une grandeur prend ses valeurs dans un ensemble dénommé *domaine de valeurs* de la grandeur. On admet qu'un domaine de valeurs contienne la valeur conventionnelle ω, dite *indéfini*, qui représente l'absence de valeur (dans un but de simplification, nous ne parlerons pas des règles particulières de propagation de cette valeur via les relations de définition). Une grandeur est spécifiée par son nom, son domaine de valeurs, son unité de mesure éventuelle ainsi que, si nécessaire, une définition précise en langue naturelle. Exemples :

```
V : réel; en m/sec;  vitesse du véhicule
D : réel; en m;      distance parcourue par le véhicule
T : réel; en sec;    temps de déplacement du véhicule
```

2.2 Relations

Dans le domaine d'application, les propriétés ne sont généralement pas indépendantes. Nous admettrons donc que pour tout état du domaine d'application, la valeur d'une grandeur puisse dépendre de celles d'autres grandeurs. Un ensemble de valeurs des grandeurs qui respecte ces dépendances constitue la représentation d'un état valide du domaine d'application. A titre d'exemple, la relation suivante exprime la dépendance entre les grandeurs V, D et T définies ci-dessus :

```
D/V = T
```


Le plus souvent, une telle relation peut prendre plusieurs formes strictement équivalentes :

```
          D/T - V = 0;
          D = T*V
ou encore D/V/T = 1.
```

2.3 Règles

Lorsque dans une relation la partie gauche est constituée du nom d'une grandeur, et que le but de la relation est de définir l'expression d'évaluation de cette grandeur, on dira que cette relation est une *règle de définition* de la grandeur. Une règle est donc une *relation directionnelle*, qui ne définit que la grandeur indiquée en partie gauche. Dans certains cas, il est possible de dériver d'une même relation autant de règles qu'il y a de grandeurs qui interviennent dans la règle. La forme générale d'une règle est,

```
g = f(H)
```

où `g` est le nom de la grandeur définie, `f` une fonction monovaluée dont on donne soit le nom (si elle définie par ailleurs ou si on ne désire pas la décrire à cet endroit) soit l'expression d'évaluation, et **H** une liste de grandeurs. On admettra que l'expression d'une fonction fasse appel à tout opérateur ou fonction mathématique, financière ou autre, que l'on jugera utile, selon une syntaxe adéquate. La définition précise d'une telle syntaxe n'offrirait pas d'intérêt particulier dans le cadre de cet article.

2.4 Grandeur à définition multiple

On n'admettra pas dans un modèle qu'une grandeur soit définie par plus d'une règle. Si une grandeur est définie par plusieurs expressions selon le contexte dans lequel elle est considérée, on utilisera une règle de définition multiple telle que la suivante :

```
A =  B + C   si D < 0
     B       si D = 0
     B - C   si D > 0
```

2.5 Modèle directionnel

Un modèle directionnel est constitué des définitions d'un ensemble de grandeurs et d'un ensemble de règles. Parmi les grandeurs, certaines sont définies par une règle, d'autres non. Ces dernières grandeurs sont appelées *données* de modèle. Connaissant la valeur de chacune des données, il est possible de déterminer les valeurs des autres grandeurs, qui jouent alors le rôle de *résultat*. L'ensemble de toutes ces valeurs représente un état valide

du domaine d'application. Les spécifications suivantes définissent l'un des trois modèles directionnels qui pourraient être dérivés à partir des définitions précédentes :

```
Données   V : réel; en m/sec;  vitesse du véhicule
          T : réel; en sec;    temps de déplacement du véhicule
Résultat  D : réel; en m;      distance parcourue par le véhicule
Règle     D = T*V
```

La notion de directionnalité est importante dans la mesure où les logiciels d'évaluation de modèles se distinguent quant au moment où le rôle des grandeurs doit être défini. Les **processeurs d'équations** (TK!Solver, EUREKA, FORMULA-1, MATHEMATICA, MACSIMA) permettent généralement que l'on définisse au départ les grandeurs et les relations, puis qu'a **posteriori** on choisisse parmi ces grandeurs les données et les résultats. On peut d'ailleurs changer ce rôle à tout moment. On retrouvera ce caractère de non directionnalité dans certains prédicats exprimés en PROLOG. Pour ces logiciels, la forme - ou **direction** - de la relation est indifférente. Par contre, les **tableurs** et les systèmes simples de **règles de production** n'admettent que des modèles dans lesquels on a choisi **a priori** et définivement les données et les résultats. C'est à ces derniers modèles que nous nous limiterons dans la suite de l'exposé.

2.6 Vue externe d'un modèle directionnel

Cette notion est liée à l'objectif que l'on assigne au modèle : étant donné des valeurs connues des données, on désire connaître la valeur de certaines grandeurs dérivées, les résultats. Si l'on désigne par G l'ensemble des grandeurs, par D le sous-ensemble des données, par R le sous-ensemble des résultats et par F l'ensemble des règles, un modèle peut s'écrire symboliquement sous la forme,

D
R
R = F(G)

Une telle description correspond à la vue externe du modèle, celle qui est perçue par l'utilisateur.

2.7 Vue interne d'un modèle directionnel

Si le sous-ensemble I=G-D-R n'est pas vide, il contient des grandeurs qui sont définies par une règle, mais qui ne sont pas pour autant un résultat attendu par l'utilisateur du modèle. Nous les appellerons *grandeurs internes*. Elles correspondent à des grandeurs représentatives du domaine d'application qui ont été prises en compte afin de simplifier l'élaboration des règles. La définition et les règles relatives aux grandeurs internes sont

ignorées de l'utilisateur du modèle. On distinguera ces trois classes de grandeurs dans la spécification complète du modèle (RI désigne l'ensemble des résultats et des grandeurs internes) :

D
R
I
RI = F(G)

L'exemple de la figure 1 est un développement du modèle suivant,

```
{RETENUE-FISCALE,NET-PAYE} = F(ANCIENNETE,NIVEAU,PRIMES,INDEX)
```

qui donne la retenue fiscale et le montant net payé à un employé dont on connaît l'ancienneté, le niveau barémique, le montant des primes et connaissant le niveau de l'index des prix. Pour simplifier, on n'a pas indiqué le domaine de valeur, l'unité ni la description sémantique des grandeurs. On admet que les fonctions BR, CS et RF, soit sont déjà définies, soit seront définies ultérieurement. Elle seront vraisemblablement évaluées par consultation d'une table de barèmes ou par l'exécution d'un algorithme dont on ne désire pas préciser le détail. Une telle fonction est en fait un sous-modèle, constituant une sorte de module de connaissances autonome; une approche méthodologique doit proposer une décomposition modulaire des connaissances, par exemple basée sur la notion d'objet (voir section 4). Observons enfin que l'énoncé des règles d'un modèle est de nature purement **déclarative**, et non **algorithmique**, l'ordre des règles étant strictement indifférent.

```
Données
    ANCIENNETE, NIVEAU, PRIMES, INDEX
Résultats
    RETENUE-FISCALE, NET-PAYE
Grandeurs internes
    BRUT, COTISATION-SOCIALE, NET-IMPOSABLE
Règles
    BRUT = (BR(ANCIENNETE, NIVEAU) + PRIMES)*INDEX
    COTISATION-SOCIALE = CS(BRUT, NIVEAU)
    NET-IMPOSABLE = BRUT - COTISATION SOCIALE
    RETENUE-FISCALE = RF(NET-IMPOSABLE)
    NET-PAYE = NET-IMPOSABLE - RETENUE-FISCALE
```

Figure 1 - *Un modèle de calcul du traitement d'un employé*

2.8 Grandeurs et règles logiques

Les grandeurs logiques ou à valeur boléenne constituent un concept essentiel pour l'expression des modèles d'aide à la décision. Nous illustrerons ce concept par le modèle de la figure 3 qui détermine la durée et le montant d'un prêt accordé à un client dont on connaît le salaire, l'âge, le montant du compte et la fiabilité (grandeur booléenne). Un tel modèle se

traduira aussi aisément dans un tableur que dans un système de règles de production [HAINAUT89].

Données
SALAIRE, AGE, COMPTE, FIABLE
Résultats
DUREE, MONTANT
Grandeurs internes
PRET-ACCORDE, CLIENT-JEUNE, CLIENT-MOYEN, CLIENT-AGE,
PRET-MODERE, PRET-ELEVE, BON-CLIENT, HAUT-SALAIRE
Règles

```
DUREE    =  0  si non PRET-ACCORDE
           10  si PRET-ACCORDE et CLIENT-JEUNE
            8  si PRET-ACCORDE et CLIENT-MOYEN
            6  si PRET-ACCORDE et CLIENT-AGE
MONTANT  =  0            si non PRET-ACCORDE
            5 x SALAIRE  si PRET-MODERE
           10 x SALAIRE  si PRET-ELEVE
PRET-ACCORDE =  FIABLE ou (HAUT-SALAIRE et COMPTE > 0)
PRET-MODERE  =  PRET-ACCORDE et non BON-CLIENT
PRET-ELEVE   =  PRET-ACCORDE et BON-CLIENT
BON-CLIENT   =  FIABLE et HAUT-SALAIRE
HAUT-SALAIRE =  SALAIRE > 1000.000
CLIENT-JEUNE =  AGE ≤ 30
CLIENT-MOYEN =  AGE ≤ 60 et non CLIENT-JEUNE
CLIENT-AGE   =  non (CLIENT-JEUNE ou CLIENT-MOYEN)
```

Figure 2 - *Exemples de définitions multiples et de règles logiques*

3. NOTION DE MODELE DIMENSIONNE

Il arrive fréquemment qu'un domaine d'application contienne **plusieurs objets similaires** obéissant aux mêmes règles de comportement, ou encore, que l'on désire considérer non pas un état de ce domaine, mais une **suite d'états**. Le modèle descriptif de telles situations utilisera alors des *grandeurs dimensionnées*.

3.1 Grandeurs dimensionnées

On associe à une grandeur dimensionnée, non pas une valeur, mais **une suite de valeurs**. Considérons par exemple le domaine des *salaires et traitements du personnel* dans une entreprise. Nous avons établi à la Figure 1 un modèle qui décrit l'état d'un employé pour un période déterminée. Fixer ou calculer une valeur pour chaque grandeur donne une représentation de l'état du traitement d'**une** personne pour **cette** période. Admettons que l'on désire décrire ce domaine d'application dans son évolution temporelle. En d'autre termes, et selon une échelle adéquate, on se propose de prendre en compte la dimension du temps, que l'on représentera par la grandeur T. Il serait aisé de généraliser l'approche

suivie jusqu'à présent en traitant cette nouvelle grandeur comme les autres, et en ajoutant de nouvelles règles telles que INDEX = I(T), où I est une fonction délivrant la valeur de l'index du mois T. Cependant, la spécificité du traitement et surtout de l'interprétation de ce type de grandeurs multi-valuées (une telle grandeur peut être considérée comme une **dimension** du domaine) suggère une représentation qui leur soit propre. On proposera par conséquent la notation indicée suivante : $INDEX_T$, dans laquelle on considère que T désigne les mois de t1 à tn, selon une séquence quelconque, ce que l'on écrira sous la forme T : t1..tn.
Précisons encore que la notation "$X_T=f(Y_T)$" est mise pour "$X_t=f(Y_{ti})$ (i=1..n)" et représente donc les **n** règles similaires : "$X_{t1}=f(Y_{t1})$, $X_{t2}=f(Y_{t2})$,..., $X_{tn-1}=f(Y_{tn-1})$, $X_{tn}=f(Y_{tn})$". En outre, l'expression "$\Sigma_T\ Y_T$", est une forme condensée de "$\Sigma_{t=t1..tn}\ Y_t$".

3.2 Modèle dimensionné

Considérons à titre d'illustration de la notion de dimensionnement la spécification de la figure 3 comme une extension du modèle de la figure 1. On y a en outre enrichi le modèle en considérant de nouvelles grandeurs qui sont le total des cotisations et le total des retenues fiscales. Celles-ci ne dépendent plus de T.

Données
T : t1..tn
$ANCIENNETE_T$, $NIVEAU_T$, $PRIMES_T$, $INDEX_T$
Résultats
$RETENUE\text{-}FISCALE_T$, $NET\text{-}PAYE_T$
TOTAL-RETENUES-E, TOTAL-COTISATIONS-E
Grandeurs internes
$BRUT_T$, $COTISATION\text{-}SOCIALE_T$, $NET\text{-}IMPOSABLE_T$
Règles

$BRUT_T = (BR(ANCIENNETE_T, NIVEAU_T) + PRIMES_T) * INDEX_T$
$COTISATION\text{-}SOCIALE_T = CS(BRUT_T, NIVEAU_T)$
$NET\text{-}IMPOSABLE_T = BRUT_T - COTISATION\text{-}SOCIALE_T$
$RETNUE\text{-}FISCALE_T = RF(NET\text{-}IMPOSABLE_T)$
$NET\text{-}PAYE_T = NET\text{-}IMPOSABLE_T - RETENUE\text{-}FISCALE_T$

$TOTAL\text{-}RETENUES\text{-}E = \Sigma_T\ RETENUE\text{-}FISCALE_T$

$TOTAL\text{-}COTISATIONS\text{-}E = \Sigma_T\ COTISATION\text{-}SOCIALE_T$

Figure 3 - *Un modèle de calcul du traitement d'un employé selon le mois*

Considérons à présent que le domaine d'application s'étende en outre à **tous les employés** de l'entreprise. En traitant cet aspect comme l'ajout d'une dimension supplémentaire du problème, on obtient le modèle présenté dans la figure 4. La grandeur E représente un employé

(admettons pour l'instant que son domaine est constitué d'entiers qui désignent les employés) et est traité comme l'était Tdans le modèle de la figure 3. Nous avons aussi considéré de nouvelles grandeurs qui sont le total des cotisations par mois,et le total de toutes les cotisations.

Données

T : t1..tn
E : e1..em
$ANCIENNETE_{T,E}$, $NIVEAU_{T,E}$, $PRIMES_{T,E}$, $INDEX_T$

Résultats

$RETENUE\text{-}FISCALE_{T,E}$, $NET\text{-}PAYE_{T,E}$
$TOTAL\text{-}RETENUES\text{-}E_E$, $TOTAL\text{-}COTISATIONS\text{-}E_E$
$TOTAL\text{-}COTISATIONS\text{-}M_T$, TOTAL-COTISATIONS

Grandeurs internes

$BRUT_{T,E}$, $COTISATION\text{-}SOCIALE_{T,E}$, $NET\text{-}IMPOSABLE_{T,E}$

Règles

$BRUT_{T,E}$= (BR($ANCIENNETE_{T,E}$, $NIVEAU_{T,E}$) + $PRIMES_{T,E}$)*$INDEX_{T,E}$
$COTISATION\text{-}SOCIALE_{T,E}$ = CS($BRUT_{T,E}$, $NIVEAU_{T,E}$)
$NET\text{-}IMPOSABLE_{T,E}$ = $BRUT_{T,E}$ - $COTISATION\text{-}SOCIALE_{T,E}$
$RETENUE\text{-}FISCALE_{T,E}$ = RF($NET\text{-}IMPOSABLE_{T,E}$)
$NET\text{-}PAYE_{T,E}$ = $NET\text{-}IMPOSABLE_{T,E}$ - $RETENUE\text{-}FISCALE_{T,E}$

$$TOTAL\text{-}RETENUES\text{-}E_E = \sum_T RETENUE\text{-}FISCALE_{T,E}$$

$$TOTAL\text{-}COTISATIONS\text{-}E_E = \sum_T COTISATION\text{-}SOCIALE_{T,E}$$

$$TOTAL\text{-}COTISATION\text{-}M_T = \sum_E COTISATION\text{-}SOCIALE_{T,E}$$

$$TOTAL\text{-}COTISATIONS = \sum_T TOTAL\text{-}COTISATIONS\text{-}M_T$$

Figure 4 - *Un modèle de calcul du traitement des employés selon le mois*

4. MODELES ET BASES DE DONNEES

4.1 Origine des grandeurs d'un modèle et puissance d'abstraction

Nous ne nous sommes jusqu'à présent jamais préoccupés de l'origine des données d'un modèle. Tout au plus a-t-on admis qu'une valeur de chacune d'elles devait être fournie lors de l'évaluation du modèle. Il est clair cependant que certaines données relèvent d'une description générale du domaine d'application, et qu'elles ont une portée beaucoup plus large que celle du modèle élaboré, englobant par exemple des applications traditionnelles. La remarque vaut également pour les résultats du modèle, qui vont à l'occasion servir de données à d'autres modèles. Il importe donc de considérer non seulement l'interdépendance des modèles d'aide à la décision relatifs à un même domaine d'application, mais aussi les relations étroites entre aide à la décision et informatique opérationnelle.
Il apparaît également que les grandeurs et les relations d'un modèle ne peuvent constituer une abstraction que de certains aspects seulement du domaine modélisé : propriétés élémentaires et relations d'inférence. En particulier, la structuration du domaine selon des classes d'objets que caractérisent ces propriétés et d'associations entre objets ne peut y être représentée.
Il est donc nécessaire d'étendre les concepts étudiés jusqu'ici à l'aide de *structures statiques d'objets*, domaine couvert aujourd'hui par les approches sémantiques de spécification de bases de données (une extension selon l'approche orientée-objet constituerait une variante intéressante, mais probablement plus délicate en ce qui concerne l'utilisateur final). En d'autres termes, nous chercherons à intégrer spécification de modèles et structures de bases de données. Notons qu'on retrouve un tel mouvement dans l'évolution de certains environnements de développement en IA, où des systèmes de règles de production ont été tardivement enrichis de structures d'objets (NExpert de Neuron Data, LEVEL-5 d'Information Builders, extensions objets de PROLOG) [FIKES,85].
A titre d'exemple, et adoptant une représentation graphique du formalisme de spécification Entité/Association, le schéma de la figure 5 spécifie sommairement une base de données dans laquelle le modèle de la figure 4 trouverait toutes les données nécessaires à son évaluation. Une entité ETAT-EMPLOYE décrit les caractéristiques variables d'un employé à une date (mois) déterminée. Pour le reste, l'interprétation du schéma ne pose pas de problème particulier.

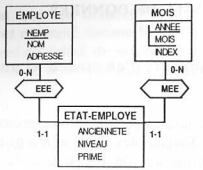

Figure 5 - *Schéma E/A partiel d'une base de données qui décrit les caractéristiques signalétiques et mensuelles d'un ensemble d'employés liées au calcul de leurs traitements.*

4.2 Extension des grandeurs d'un modèle aux objets d'une base de données

Relativement à cette base de données, la grandeur E définie dans le modèle de la figure 4 désigne de toute évidence une entité EMPLOYE tandis que T désigne une entité ETAT-EMPLOYE (qui aurait hérité de l'attribut INDEX du MOIS qui lui est associé). Dans ce cas donc, une **dimension d'un modèle** correspond a un **ensemble d'entités** de la base de données. En outre, une **grandeur dimensionnée** peut immédiatement être assimilée aux **valeurs d'un attribut** attachées à ces entités, attribut réel (s'il s'agit d'une donnée) ou dérivé. Nous exploiterons cette interprétation pour étendre les concepts *externes* des modèles aux objets d'une base de données.
Le modèle Entité-Association s'imposant à l'heure actuelle comme standard de spécification du schéma conceptuel d'une base de données [CHEN,76] [TARDIEU,83], [BODART,89], nous l'adopterons pour décrire les grandeurs externes d'un modèle.

4.3 Variables d'entité et ensemble d'entités

De la même manière qu'une grandeur jouant le rôle de dimension prend ses valeurs dans une suite définie explicitement, une grandeur représentant une entité prendra ses valeurs dans un ensemble d'entités que l'on définira au moyen d'un prédicat, ou plus généralement d'une expression de requête.

Variable entité

Nous appellerons **variable entité** une grandeur du modèle (donnée, résultat ou interne) jouant le rôle de dimension et définie sur un ensemble d'entités. Avant même de définir la structure d'un modèle qui fait usage de telles grandeurs, il nous faut choisir un langage de requête qui permette

de décrire des ensemble d'entités. Visant essentiellement la simplicité nous proposerons un langage dédié à une restriction binaire du formalisme E/A dans laquelle les types d'associations sont simples, c'est-à-dire de degré 2, et sans attributs, et les attributs d'un type d'entités sont obligatoires, monovalués et indécomposables. Cette restriction du modèle E/A n'a pour but que de simplifier le langage de désignation d'ensembles d'entités. Un schéma qui comporte des structures plus complexes peut toujours être transformé en un schéma sémantiquement équivalent qui respecte les restrictions ci-dessus [HAINAUT,89].

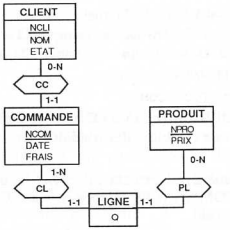

Figure 6 - *Exemple de schéma E/A*

Désignation d'un ensemble d'entités

Nous indiquerons au travers de quelques exemples représentatifs les principales caractéristiques du langage de désignation de données que nous utiliserons. Le lecteur intéressé trouvera dans [HAINAUT,86] une description formelle plus complète de ce langage. Les exemples ci-dessous sont basés sur le schéma de la figure 6.

a) l'ensemble de toutes les entités d'un type est désigné par le nom de ce type. Ainsi, l'expression,

```
COMMANDE
```

désigne l'ensemble des entités du type COMMANDE.

b) le nom d'une variable entité désigne une entité particulière. Si la variable CLI est du type CLIENT, l'expression,

```
CLI
```

désigne une entité CLIENT particulière.

c) l'ensemble des entités dont la valeur d'un attribut vérifie une condition déterminée est décrit par une expression telle que,

```
CLIENT(: ETAT > 0)
```

qui désigne les entités CLIENT dont la valeur d'ETAT est supérieure à 0.

d) l'ensemble des entités associées via des associations d'un type déterminé à une entité désignée par une variable est décrit par une expression telle que la suivante,

```
COMMANDE(CC: CLI)
```

qui décrit les entités COMMANDE associées via CC à l'entité CLI.
Inversement, si COM est une variable désignant une entité COMMANDE, l'expression

```
CLIENT(CC: COM),
```

désigne l'entité CLIENT associée via CC à l'entité COM.
Enfin, il est permis de combiner des conditions du type c et d, comme dans l'expression

```
COMMANDE(CC: CLIENT(: ETAT > 0))
```

qui désigne les COMMANDES (associées via CC à) des CLIENTS dont l'ETAT est positif.

e) un ensemble d'entités peut être décrit par une expression booléenne de conditions :

```
COMMANDE((: FRAIS = 0) et (CC: CLI))
```

Cette expression désigne les COMMANDES du CLIENT CLI dont les FRAIS sont nuls.

4.4 Utilisation des objets d'une base de données dans un modèle

Définition d'une variable entité et de son domaine

Comme toute dimension, une variable entité sera déclarée comme grandeur données, résultat ou interne. On indiquera son type, qui un type d'entités du schéma de la base de données :

```
CLI : CLIENT
```

Si une variable entité prend ses valeurs dans un *sous-ensemble* seulement des entités d'un type, on indiquera ce sous-ensemble par une expression telle que la suivante (que l'on considérera pour l'instant comme une règle) :

```
COM = COMMANDE(CC: CLI)
```

qui spécifie que COM, variable déclarée du type COMMANDE, prend ses valeurs dans l'ensemble COMMANDE(CC:CLI) et donc que COM, dans toute règle où elle apparaît en partie droite, désigne une entité COMMANDE associée via CC à l'entité CLI. Si CLI est une variable qui peut prendre plusieurs valeurs (c-à-d prenant ses valeurs dans un ensemble de cardinalité éventuellement supérieure à 1), alors l'ensemble COMMANDE(CC:CLI) est *paramétré* selon CLI. La variable COM est par conséquent considérée comme étant implicitement *indicée* par CLI.

Valeurs d'attributs d'entités

La désignation dans une règle d'une *valeur d'attribut* d'une entité se fait comme pour une grandeur dimensionnée traditionnelle. Par exemple, $ETAT_{CLI}$ désigne la valeur d'ETAT de l'entité CLI.
Supposons que CLI désigne une entité CLIENT, et que COM désigne une entité COMMANDE liée à l'entité CLI selon la définition COM = COMMANDE(CC:CLI). Selon les conventions d'indiçage dimensionnel proposées en 3.2, on devrait dénoter la valeur de FRAIS de cette entité COM sous la forme $FRAIS_{CLI,COM}$. Cependant, la variable COM étant paramétrée selon CLI, la notation $FRAIS_{COM}$ est parfaitement définie, et sera utilisée désormais.

Fonctions d'agrégation et règles de domaine

La définition d'un ensemble d'entités sera souvent utilisée pour spécifier l'intervalle d'une fonction d'agrégation, telle que la sommation :

$$TOTAL_{CLI} = \Sigma_{COM=COMMANDE(CC:CLI)} FRAIS_{COM}$$

qui définit la grandeur $TOTAL_{CLI}$ comme la somme des frais des commandes du client CLI. Si cet intervalle intervient dans plusieurs expressions, ou si l'on désire alléger l'écriture des fonctions d'agrégation, on définira globalement le domaine de la variable COM sous la forme d'une **règle de domaine**, comme dans le fragment ci-dessous, équivalent à la dernière expression :

$$COM = COMMANDE(CC:CLI)$$

$$TOTAL_{CLI} = \Sigma_{COM} FRAIS_{COM}$$

Dans ce cas cependant, selon la convention d'unicité de définition d'une grandeur, la variable COM est définitivement liée à CLI par la règle de domaine, alors qu'avec l'expression précédente, COM est une variable libre du type COMMANDE qui peut apparaître dans d'autres expressions utilisant d'autres domaines sous-ensembles de COMMANDE. Observons qu'une règle de domaine définit à la fois une variable et son domaine. On sera d'ailleurs amené à utiliser le nom de cette variable pour désigner son domaine (voir *Fonctions de domaine* ci-dessous)

Indiçage par l'expression d'un ensemble singleton

Lorsque l'ensemble désigné est de cardinalité 1 par construction, son expression elle-même peut être utilisée pour indicer une grandeur (telle qu'un attribut). C'est ainsi que le PRIX de l'unique PRODUIT de la LIGNE L sera désigné par l'expression,

$$P = PRIX_{PRODUIT(PL:L)}$$

Ici encore l'entité PRODUIT pourrait être définie par une règle de domaine sous la forme équivalente suivante,

$$PRO = PRODUIT(PL:L)$$
$$P = PRIX_{PRO}$$

Domaines paramétrés et relation d'ordre

Nous avons vu qu'un domaine pouvait être lié à une variable entité, qui en constitue donc un *paramètre*. Ce paramètre n'apparaît explicitement que dans la règle de domaine. Il est possible de donner un paramètre externe à la définition d'un domaine. Dans ce cas, il est permis de le spécifier explicitement dans le nom de la variable associée au domaine. Supposons que toutes les valeurs de l'attribut NCLI de CLIENT constituent le domaine [1..1000]. On peut écrire :

$$CLI(I) = CLIENT(:NCLI=I)$$

$$TOTAL_{CLI} = \Sigma_{I=1..5} ETAT_{CLI(I)}$$

Cette notation est particulièrement utile lorsqu'il est nécessaire de désigner des éléments par leur position selon un ordre déterminé. On pourrait par exemple écrire,

$$\ldots \text{ si } ETAT_{CLI(I-1)} < ETAT_{CLI(I)}$$

On trouvera en section 5 un cas représentatif de domaine à paramètre explicite.

Fonctions de domaine

Un domaine étant un ensemble, il est possible d'en déduire des grandeurs par application de fonctions d'agrégation telles que *taille*, *maximum*, *moyenne*, etc, selon le type des éléments.
L'expression ci-dessous définit N comme le nombre de commandes du client CLI. N est implicitement paramétré selon CLI.

$$COM = COMMANDE(CC:CLI)$$
$$N = taille(COM)$$

4.5 La base de données, source et destination d'un modèle

Nous avons montré comment les informations d'une base de données pouvaient être utilisées dans un modèle en tant que **données**. Rien n'empêche de construire des règles numériques, logiques ou plus généralement de *calcul* (incluant par là des traitements sur des types de données plus étendus tels que les chaînes de caractères ou les types temporels) qui définissent, non pas des grandeurs locales au modèle, mais des valeurs d'attributs d'entités de la base de données. De telles règles définissent des attributs comme grandeurs **résultats** du modèle. En d'autre terme, elles définissent certains objets de la base de données comme étant calculables ou déductibles. Si certaines données du modèle ne sont pas extraites de la base de données, mais proviennent par exemple de l'utilisateur, de telles règles définissent des **mises à jour** des données de la base. Enfin, une grandeur indicée par une variable entité, mais qui ne correspond pas à un attribut du type d'entité de la variable, représente un attribut virtuel de ce type d'entités. Le modèle définit alors les **règles d'inférence** qui étendent le schéma de la base de données. On rejoint là le domaine des bases de données déductives [GALLAIRE,84]. Rappelons cependant que notre démarche vise à étendre les notions d'un modèles aux concepts d'une base de données plus qu'à étendre les concepts de cette dernière par des fonctionnalités déductives. Nous ne développerons donc pas plus avant cet aspect, qui déborde du cadre limité de cet exposé.

4.6 Trois exemples de modèles

Les deux premiers exemples sont relatifs au calcul du montant dû par un (ou plusieurs) client pour des commandes qu'il a passées. On dispose d'une base de données obéissant au schéma 6.
Le premier modèle établit les règles de calcul du montant dû par le client dont on donne le numéro de client. On y précise que le montant dû est égal à la somme des montants de ses commandes de laquelle on retire l'état du crédit dont il dispose (ETAT). En outre, le montant d'une commande est égal au total des montants de ses lignes augmenté des frais. Enfin, le montant d'une ligne est obtenu en multipliant la quantité de la ligne par le prix du produit de la ligne. On notera qu'on a choisi de ne pas indicer MONTANT-DU selon CLI, ce dernier désignant un singleton par construction.

Données
N : entier ; numéro d'un client
Résultats
MONTANT-DU : réel
Grandeurs internes
CLI : CLIENT
COM : COMMANDE
L : LIGNE
$MONTANT\text{-}COM_{COM}$: réel
$MONTANT\text{-}LIGNE_{L}$: réel
Règles
CLI = CLIENT(: NCLI = N)

$$MONTANT\text{-}DU = (\sum_{COM=COMMANDE(CC:CLI)} MONTANT\text{-}COM_{COM}) - ETAT_{CLI}$$

$$MONTANT\text{-}COM_{COM} = (\sum_{L=LIGNE(CL:COM)} MONTANT\text{-}LIGNE_{L}) + FRAIS_{COM}$$

$$MONTANT\text{-}LIGNE_{L} = Q_{L} \times PRIX_{PRODUIT(PL:L)}$$

Figure 7 - *Modèle du calcul du montant dû par le client dont on donne le numéro.*

Le deuxième modèle est obtenu en *indiçant* la grandeur MONTANT-DU du modèle 7 selon les entités CLIENT représentées par CLI. La grandeur CLI est ici proposée comme donnée. On observe une propriété remarquable qui est *l'invariance* des règles de définition de $MONTANT\text{-}COM_{COM}$ et de $MONTANT\text{-}LIGNE_{L}$ lors du processus d'*indiçage* du modèle. Nous avons en effet défini ces deux concepts comme attachés respectivement à toute entité COMMANDE et toute entité LIGNE, indépendamment du contexte dans lequel ils sont utilisés. Cette propriété était absente des premiers modèles à indiçage comme on peut l'observer dans le processus d'*indiçage* du modèle 3 en modèle 4 par exemple. On peut admettre que cette propriété découle de l'approche Entité/Association.

Données
CLI : CLIENT
Résultats
$MONTANT\text{-}DU_{CLI}$: réel
Grandeurs internes
COM : COMMANDE
L : LIGNE
$MONTANT\text{-}COM_{COM}$: réel
$MONTANT\text{-}LIGNE_{L}$: réel
Règles

$$MONTANT\text{-}DU_{CLI} = (\sum_{COM=COMMANDE(CC:CLI)} MONTANT\text{-}COM_{COM}) - ETAT_{CLI}$$

$$MONTANT\text{-}COM_{COM} = (\sum_{L=LIGNE(CL:COM)} MONTANT\text{-}LIGNE_{L}) + FRAIS_{COM}$$

$$MONTANT\text{-}LIGNE_{L} = Q_{L} \times PRIX_{PRODUIT(PL:L)}$$

Figure 8 - *Modèle du calcul du montant dû par chaque client. Ce modèle est obtenu par indiçage du modèle 7 selon CLI.*

Le troisième exemple est une réécriture du modèle 4 couplé avec une base de données de description du personnel obéissant au schéma 5. La distinction entre les concepts de MOIS et de ETAT-EMPLOYE mise en évidence dans le schéma de la base de données nous amène a distinguer les variables T et M qui étaient confondues dans le modèle 4.

Données
E : EMPLOYE
M : MOIS
Résultats
$TOTAL\text{-}RETENUES\text{-}E_{E}$, $TOTAL\text{-}COTISATION\text{-}E_{E}$
$TOTAL\text{-}COTISATION\text{-}M_{M}$, TOTAL-COTISATION
Grandeurs internes
T : ETAT-EMPLOYE
BUT_{T}, $COTISATION\text{-}SOC_{T}$, $NET\text{-}IMPOSABLE_{T}$,
$RETENUE\text{-}FISCALE_{T}$, $NET\text{-}PAYE_{T}$
Règles

$$BRUT_{T} = (BR(ANCIENNETE_{T}, NIVEAU_{T}) + PRIMES_{T}) * INDEX_{MOIS(MEE:T)}$$

$$COTISATION\text{-}SOC_{T} = CS(BRUT_{T}, NIVEAU_{T})$$

$$NET\text{-}IMPOSABLE_{T} = BRUT_{T} - COTISATION\text{-}SOC_{T}$$

$$RETENUE\text{-}FISCALE_{T} = RF(NET\text{-}IMPOSABLE_{T})$$

$$NET\text{-}PAYE_{T} = NET\text{-}IMPOSABLE_{T} - RETENUE\text{-}FISCALE_{T}$$

$$TOTAL\text{-}RETENUES\text{-}E_{E} = \sum_{T=ETAT-EMPLOYE(EEE:E)} RETENUE\text{-}FISCALE_{T}$$

$$TOTAL\text{-}COTISATION\text{-}E_{E} = \sum_{T=ETAT-EMPLOYE(EEE:E)} COTISATION\text{-}SOC_{T}$$

$$\text{TOTAL-COTISATION-M}_M = \sum_{T=\text{ETAT-EMPLOYE(MEE:M)}} \text{COTISATION-SOC}_T$$

$$\text{TOTAL-COTISATION} = \sum_M \text{TOTAL-COTISATION-M}_M$$

Figure 9 - *Révision du modèle de calcul du traitement des employés selon le mois, (figure 4). Les données sont extraites de la base de données décrite par le schéma de la figure 5.*

On notera ici encore l'invariance des définitions telles que celles de BRUT, COTISATION-SOC ou NET-PAYE qui sont exprimées comme s'il s'agissait d'attributs d'ETAT-EMPLOYE, indépendants du contexte dans lequel ils sont employés.

5. UNE ETUDE DE CAS

Ce chapitre est consacré à l'énoncé d'un problème simple d'aide à la décision, et à une proposition de solution sous la forme d'une base de données et d'une base de règles. Nous n'en proposerons que la spécification sous la forme d'un schéma conceptuel et d'un modèle. La démarche *d'élaboration* de cette solution et son *implantation* dans un outil opérationnel déborde du cadre de cet article (voir [TARDIEU,83], [BODART,89], [HAINAUT,86] pour les bases de données et [RONEN,89] [HAINAUT,89] pour les modèles). Le problème posé étant relativement simple, l'implantation de la solution proposée dans des logiciels intégrés tels que GURU, SQL*CALC ou EXCEL/SQL-Server ne poserait pas de problèmes importants.

5.1 Enoncé : étude du coût d'une liaison aérienne

On propose de construire la base de connaissances d'un système d'aide à la décision qui permet d'étudier les coûts de vols aériens. Ce système doit aider le service de planification des vols tant en ce qui concerne les choix à court terme (tels que les charges utiles à transporter et les quantités de carburant à acheter à chaque escale) que les choix à moyen terme (relatifs, par exemple, au choix des escales d'un vol ou des appareils). Le domaine d'application se présente comme suit.

"On considère des vols aériens dont chacun relie deux villes en passant par un certain nombre d'escales, qui sont des aéroports. Un même vol peut être effectué à des dates différentes par des appareils différents. Un appareil est caractérisé par la capacité de ses réservoirs (en kilos de carburant) ainsi que la consommation à vide, équipage compris (en kilos de carburant par Km).
On connaît aussi sa charge utile maximale et sa consommation supplémentaire par kilo de charge (en kilos de carburant par kilo de charge et par km). Il est à noter cependant que la consommation à vide n'inclut pas le transport du carburant lui-même. On admet que la charge utile (fret et passagers) est constante pour toute la durée du vol, mais qu'elle peut varier d'une date à l'autre. On connaît la longueur de chaque tronçon du vol, c'est-à-dire la distance entre deux villes ou escales consécutives de ce vol. On supposera que la consommation en vol est une fonction linéaire de la charge emportée (charge utile + carburant).
A chaque escale, l'appareil est ravitaillé en carburant. Celui-ci est acheté au tarif local (en dollars par kilo). Le tarif local dépend de la date.

Lorsque l'appareil atterrit, ainsi que dans l'aéroport de départ, ses réservoirs peuvent contenir une quantité résiduelle non consommée lors du parcours du tronçon précédent. Pour effectuer le tronçon suivant, il est généralement nécessaire d'ajouter au réservoir une quantité qui permet d'atteindre l'escale ou la ville suivante. Il est cependant possible d'emporter une quantité supérieure à ce qui est strictement nécessaire. Ce supplément peut être intéressant si le tarif local est particulièrement bas, et si le tronçon suivant n'est pas trop long. On fera l'hypothèse que la valeur financière d'une quantité résiduelle est à calculer au tarif de l'endroit où cette quantité est observée, donc à l'atterrissage (= valeur de revente). On notera que l'appareil est présumé avoir consommé la quantité résiduelle à l'aéroport de départ, et qu'il faut donc lui imputer, mais qu'il n'a pas consommé celle qui subsiste après l'atterrissage final, et qu'il ne faut donc pas lui imputer puisqu'elle n'aura pas servi au vol."

5.2 Schéma de la base de données

On propose une base de données correspondant au schéma conceptuel de la figure 10. Les types d'entités APPAREIL, VOL-TYPE, TRONCON-TYPE et AEROPORT définissent les propriétés communes à tous les vols effectués. Ils représentent des informations structurelles indépendantes du temps, du moins à court terme. Les types d'entités VOL, TRONCON et ESCALE-VOL par contre représentent les informations journalières concernant les vols effectués et les tarifs locaux du carburant. Ils constituent en quelque sorte des *instanciations* temporelles respectivement de VOL-TYPE, TRONCON-TYPE et AEROPORT, auxquels ils sont associés par des types d'associations **type-de**; à ce titre, ils héritent des caractéristiques (attributs, associations) des objets qui représentent leur type : on peut parler de l'ID-VOL d'un VOL ou de la DISTANCE d'un TRONCON. On trouvera une description de cet important type de lien inter-entité et des règles d'héritage qui lui sont spécifiques dans [DAVIS,89] (relation *member-of* du modèle EARL), [HAINAUT,89] ainsi que dans [BRACHMAN,83].

Une entité **APPAREIL** représente un modèle d'appareil. Elle est caractérisée par la dénomination du modèle (MODELE), la capacité maximale des réservoirs (CAP-RESERVOIR), la consommation en vol d'un appareil à vide, équipage compris, mais sans passagers, ni fret, ni carburant, exprimée en kilos de carburant par km (CONS-VIDE), la consommation en vol nécessaire au transport d'un kilo de charge sur un km (CONS-

CHARGE), la charge maximale qu'un appareil peut emporter (CH-UT-MAX).

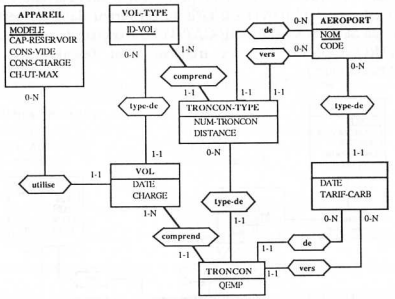

Figure 10 - *Schéma E/A de la base de données des vols aériens. Dans un but de simplification, certaines contraintes d'unicité n'ont pas été mentionnées.*

Une entité **VOL-TYPE** représente un type de vol caractérisé par le code identifiant du vol-type (ID-VOL), la suite des tronçons dont le type de vol est constitué (TRONCON-TYPE via *comprend*).

Une entité **AEROPORT** représente un aéroport. Elle est caractérisée par le nom en clair de l'aéroport (NOM), le nom de code de l'aéroport (CODE), les tronçons dont l'aéroport est le départ (TRONCON- TYPE via *de*), les tronçons dont l'aéroport est l'arrivée (TRONCON-TYPE via *vers*).

Une entité **TRONCON-TYPE** représente une section ininterrompue d'un vol-type. Elle est caractérisée par son type de vol (VOL-TYPE via *comprend*), le numéro de séquence du tronçon dans le type de vol (NUM-TRONCON), la longueur du tronçon en km (DISTANCE), l'aéroport de départ du tronçon (AEROPORT via *de*), l'aéroport d'arrivée du tronçon (AEROPORT via *vers*).

Une entité **VOL** représente un vol réel effectué à une date déterminée. Elle correspond à une entité VOL-TYPE qui en définit les propriétés stables. Elle est caractérisée par le type du vol (VOL-TYPE), le modèle de l'appareil effectuant le vol (APPAREIL via *utilise*), la date du vol (DATE), la charge utile transportée (CHARGE), les tronçons dont le vol est constitué (TRONCON via *comprend*).

Une entité **ESCALE-VOL** représente une escale constituée d'un aéroport à une date déterminée auquel tout appareil d'un vol peut atterrir ou décoller. Elle est caractérisée par l'aéroport (AEROPORT), la date à laquelle l'aéroport est considéré (DATE), le tarif du carburant dans cet aéroport à cette date, en $US, par kilo (TARIF-CARB), les tronçons dont l'escale est le départ (TRONCON via *de*), les tronçons dont l'escale est l'arrivée (TRONCON via *vers*).

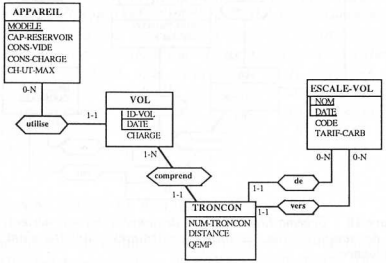

Figure 11 - *Vue E/A de la base de données des vols aériens destinée au modèle de calcul du coût des vols.*

Une entité **TRONCON** représente une section effectué par un vol réel. Elle correspond à une entité TRONCON-TYPE qui en définit les propriétés stables. Elle est caractérisée par le type de tronçon (TRONCON-TYPE), le vol réel durant lequel le tronçon a été effectué (VOL via *comprend*), la quantité de carburant emportée (achetée) à l'escale de départ du tronçon (QEMP), l'escale de départ du tronçon (ESCALE-VOL via *de*), l'escale d'arrivée du tronçon (ESCALE-VOL via *vers*).

Afin de simplifier l'élaboration du modèle de calcul du coût des vols, on dérivera du schéma 10 une vue obtenue par application des règles d'héritages aux types d'associations **type-de**, puis par abandon des types d'entités sources de cet héritage. Cette vue est représentée par la figure 11.

5.3 La base de règles de l'évaluation du coût d'un vol

Remarques. On admet que la quantité résiduelle au départ d'un vol (grandeur QRES-INIT) est donnée par l'utilisateur du modèle. En outre, le

modèle ne devant évaluer que le coût d'un seul vol, on a choisi de ne pas dimensionner COUT-VOL selon V.

Données

```
ID: entier                  ; numéro du vol
D : date                    ; date du vol
QRES-INIT: réel,en kg       ; quantité résiduelle de carburant à l'aéroport de départ
```

Résultats

```
COUT-VOL : réel,en $        ; coût du vol numéro ID à la date D
```

Grandeurs internes

```
V: VOL                      ; vol dont on calcule le coût
APP: APPAREIL               ; appareil effectuant le vol V
T: TRONCON                  ; tronçons du vol V
N: entier
I: entier
T(I): TRONCON               ; tronçon numéro I du vol V
E: ESCALE-VOL               ; escale de départ d'un tronçon du vol V
COUT-INIT: réel,en $        ; valeur du carburant résiduel à l'escale de départ
                              du vol V
VALEUR-RES: réel,en $       ; valeur du carburant résiduel à l'escale d'arrivée
                              du vol V du tronçon T
QINIT_T: réel,en kg         ; quantité de carburant résiduel avant achat à l'escale
                              de départ du tronçon T
QDEPART_T: réel,en kg       ; quantité de carburant au décollage du tronçon T
QRES_T: réel,en kg          ; quantité de carburant résiduel à l'atterrissage
                              du tronçon T
QCONS_T: réel,en kg         ; quantité du carburant consommée durant le tronçon T
QCONS/km_T: réel,en kg      ; quantité du carburant consommée par km durant
                              le tronçon T
CHARGE-MOY_T: réel,en kg    ; poids total moyen de l'appareil durant le tronçon T
QMOY_T: réel, en kg         ; quantité moyenne de carburant durant le tronçon T
```

Règles

```
V = VOL((:ID-VOL=ID) and (DATE=D))
APP = APPAREIL(utilise:V)
T = TRONCON(comprend:V)
N = taille(T)
T(I) = T(:NUM-TRONCON=I), (I = 1..N)
E = ESCALE-VOL(de:T)

COUT-VOL     = COUT-INIT + Σ_T COUT-ESC_T - VALEUR-RES
COUT-INIT    = QRES-INIT x TARIF_ESCALE-VOL(de:T(1))
VALEUR-RES   = QRES_T(N) x TARIF_ESCALE-VOL(vers:T(N))
COUT-ESC_T   = QEMP_T x TARIF_E
QRES_T       = QINIT_T + QEMP_T - QCONS_T
QINIT_T(1)   = QRES-INIT
QINIT_T(I)   = QRES_T(I-1),   (I = 2..N)
QCONS_T      = QCONS/km_T x DISTANCE_T
QCONS/km_T   = (CONS-VIDE_APP + CONS-CHARGE_APP x CHARGE-MOY_T)
CHARGE-MOY_T = CHARGE_V + QMOY_T
QMOY_T       = (QDEPART_T + QRES_T) / 2
QDEPART_T    = QINIT_T + QEMP_T
```

Figure 12 - *Modèle du calcul du coût du vol ID-VOL = ID à la date DATE = D*

6. CONCLUSIONS

A la connaissance de l'auteur, le problème auquel s'adresse cet article semble n'avoir suscité que peu de travaux jusqu'ici, malgré la nécessité croissante d'une plus grande qualité dans le développement d'applications d'aide à la décision. Disposer d'un jeu de concepts abstraits mais intuitifs est essentiel dans un domaine où l'utilisateur reste le premier, sinon le seul acteur à chaque stade de ce développement.

Utilisés en pratique lors de sessions de formation en entreprise et par des étudiants en Sciences Economiques, les concepts présentés dans cet article (à l'exclusion cependant du couplage avec la base de données, dont l'enseignement n'a pas encore été expérimenté) se sont avérés d'une aide efficace dans la formulation de modèles complexes. Il est également apparu que cette formulation était traduite sans difficulté dans le langage d'un tableur simple. Au-delà de cette validation pragmatique, cet article propose à la fois une formalisation des concepts et leur extension aux bases de données, dans le cadre général de l'architecture des bases de connaissances.

Outre une définition formelle complète, trois points sont encore à l'étude. Le premier concerne les notions de complétude et de cohérence d'un modèle : on cherche par exemple à établir dans quelles conditions un modèle est évaluable, ou deux règles sont distinctes, ou non déductibles l'une de l'autre, ou non contradictoires, des règles de récurrence sont complètes. Le deuxième point est relatif à une démarche de conception d'un modèle, qui proposerait d'une part la formalisation du processus d'analyse d'un problème, et d'autre part la traduction systématique d'un modèle selon les langages d'outils coordonnés ou intégrés. Une démarche pragmatique existe, mais doit encore être formalisée et validée. Le troisième point est celui d'outils de conception et de traduction de modèles.

Remerciements

Je remercie toutes les personnes qui ont bien voulu lire les premières versions de cet article et y apporter leurs commentaires, en particulier Karine Becker ainsi que les évaluateurs du comité de programme d'INFORSID 1990.

7. REFERENCES

BEULENS,88,
BEULENS, VAN NUNEN, "The use of Expert Systems Technology in Decision Support Systems", in Decision Support Systems, Vol. 4, N° 4, Dec. 88, North-Holland.

BLANNING,87,
BLANNING, "A Relational Theory of Model Management", in [DSS:TA,1987]

BODART,89,
BODART, PIGNEUR, "Conception assistée des applications informatiques - 1. Etude d'opportunité et analyse conceptuelle", Masson, Paris, 1989.

BRACHMAN,83,
BRACHMAN, "What IS-A is and isn't : An analysis of Taxonomic Links in Semantic Networks", IEEE Computer, Oct. 1983.

CHEN,76,
CHEN, "The entity-relationship model - toward a unified view of data", ACM TODS, Vol. 1, N° 1, 1976.

CHEN,88,
CHEN, Y-S, "An Entity-Relationship Approach to Decision Support and Expert Systems", in Decision Support Systems, Vol. 4, N° 2, June 88, North-Holland.

DAVIS,89,
DAVIS, BONNELL, "Modeling Semantics with Concept Abstraction in the EARL model", in Proc. 8th Intern. Conf. on Entity-Relationship Approach, ER Institute/ACM/IEEE, October 1989.

DSS:TA,87,
--, "Decision Support Systems : Theory and Application", Holsapple & Whinston Ed., Springer-Verlag, 1987.

FIKES,85,
FIKES, KEHLER, "The role of frame-based representation in reasoning", CACM, Vol. 28, N° 9, Sept. 1985.

GALLAIRE,84
GALLAIRE, MINKER, NICOLAS, "Logic and Database : a deductive approach", ACM Computing Surveys, Vol.16, n°2, June 1984

HAINAUT,86,
HAINAUT, "Conception assistée des applications informatiques - 2. Conception de la base de données", Masson, Paris 1986.

HAINAUT,89,
HAINAUT, "Bases de données et Base de connaissances en gestion des organisations", notes du cours présenté à la 5ème Ecole d'Automne de Bases de Données, AFCET, novembre 1989.

HOLSAPPLE,87,
HOLSAPPLE, WHINSTON, "Business expert systems", IRWIN, 1987.

JACKSON,83,
JACKSON, "System Development" Prentice-Hall, 1983.

KARAGIANNIS,87,
KARAGIANNIS, SCHNEIDER, "Data- and Knowledge- base Management systems for Decision Support", in [DSS:TA,1987].

KONOPASEK,84,
KONOPASEK, JAYARAMAN, "The TK!Solver Book - A Guide for Problem Solving", Osborne-McGraw-Hill, 1984.

LAZIMY,89,
LAZIMY, "E2R Model and Object-oriented Representation for Data Management, Process Modeling and Decision Making", Proc. 8th Intern. Conf. on Entity-Relationship Approach, ER Institute/ACM/IEEE, October 1989

RONEN,89,
RONEN, LUCAS, "Spreadsheet Analysis and Design", in CACM, Vol. 32, N°1, Jan. 89.

TARDIEU,83,
TARDIEU, ROCHFELD, COLETTI, "Méthode MERISE : principes et outils", Editions d'Organisation, Paris 1983.

VANHEE,88,
VAN HEE, LAPINSKI, "OR and AI Approaches to Decision Support Systems, Vol. 4, N° 4, Dec. 88, North-Holland.