L'interrogation de bases de données comme application des classes d'objets

Béatrice Bouchou, Julien Lerat, Denis Maurel

LI, Université François Rabelais
E3i, 64 avenue Jean Portalis, 37200 Tours
(bouchou+maurel)@univ-tours.fr

Résumé – Abstract

En travaillant sur l'interrogation de bases de données en langue naturelle, nous sommes amenés à exploiter les propositions du Laboratoire de Linguistique Informatique (LLI) en matière de représentation de la langue : les classes d'objets. Un outil d'interrogation définit une application du langage vers le modèle de l'information stockée. Ici les classes d'objets et leurs prédicats appropriés modélisent le langage source, tandis que le modèle relationnel sert pour les données interrogées. Nous présentons d'abord ce contexte d'application, puis comment nous utilisons les classes d'objets et prédicats appropriés dans ce cadre.

We investigate how to use natural language to query a database from both the linguistic and database points of view (but without AI considerations). In order to achieve this goal, we need a natural language model which we can map on to a relational database model. We have chosen to use the word classification called « classes d'objets » as proposed by the Laboratoire de Linguistique Informatique (LLI). We present here the first results of this work.

Mots clés – keywords interrogation de BD en langage naturel, modèle relationnel, classes d'objets – natural language database query, relational model, « classes d'objets »

1. Introduction

Notre système est destiné à interpréter la langue naturelle dans le cadre précis d'une interrogation de bases de données. Ses grandes lignes sont présentées entre autres dans (Bouchou, Maurel, 1999). En phase opérationnelle, l'utilisateur écrit sa question en langage naturel, puis le système lui fournit une réponse issue de la base de données. Avant d'être opérationnel, le système est "installé" sur la base de données cible. Le lien entre le sens de la question et le sens des données stockées est établi au cours de cette installation, sous la forme d'un dictionnaire électronique de mots clés.

La phase de configuration du système d'interrogation pour une base de données précise est l'une des principales pierres d'achoppement des systèmes existants (Kaplan, 1984), (Sabah, 1997) : il faut qu'elle soit rapide, et surtout qu'elle ne nécessite pas l'intervention d'un expert en linguistique, ni en intelligence artificielle. C'est ici que nous faisons intervenir les classes

d'objets et prédicats appropriés, concepts mis au point par l'équipe de Gaston Gross au LLI (Gross, 1994), (Gross, 1998), (Le Pesant, Mathieu-Colas, 1998), (Le Pesant, 2000). Les classes d'objets découlent de la théorie des opérateurs linguistiques, développée par Z.S. Harris (Harris, 1976) et décrite, pour le français, par les nombreux travaux qui ont suivi (Gross, 1975). Ces classements définis en fonction des opérateurs linguistiques spécifient les conditions (syntaxiques) que ces mots doivent remplir pour *faire sens* avec les autres mots de la question. Notre système s'attache à associer ainsi à la question le sens qu'impose la base.

La phase d'installation consiste à construire un dictionnaire de mots clefs :

- En entrée sont les mots susceptibles d'être reconnus dans une question sur la base : ces mots sont rattachés à des classes d'objets et/ou à des prédicats (ou classes de prédicats).
- En sortie sont des codes qui indiquent :
    - d'une part les éléments de la base auxquels il est fait référence dans la question,
    - et d'autre part les liens entre ces éléments dans la base, établis à partir des liens reconnus entre les mots dans la question.

Nous disposons alors d'un système de mots clés, combiné à la connaissance apportée par les opérateurs linguistiques. Pour ce dictionnaire nous travaillons sur des transducteurs à nombre fini d'états, minimaux, compactés (Revuz, 1991), (Mihov, 2000). Au contraire d'un système développé avec ILLICO (Pasero, 1999), nous n'utilisons pas de logique : d'une part, la connaissance de la langue (syntaxe/sémantique) se trouve dans les classes d'objets, et d'autre part les modèles conceptuel et contextuel sont présents dans la base interrogée.

Dans la section 2 de cette communication nous exposons ce que sont les liens entre données dans une base de données relationnelle. Cela permet, dans la section 3, de préciser ce qui est recherché dans la base lors d'une interrogation. Après cette spécification de la cible, la section 4 revient sur la théorie utilisée pour représenter la source (la question en langue naturelle), puis présente la correspondance établie entre les deux. Enfin, nous rapportons dans la section 5 des exemples de l'utilisation des classes d'objets lors de la construction du dictionnaire.

2. Le "sens" des données stockées

Une base de données représente un système d'information. Le sens des données de la base est fondamental pour concevoir un système d'interrogation : c'est *de cela* que parlera la question.

2.1 Le modèle conceptuel d'une base de données

Rappelons comment est conçue une base : la première étape consiste à modéliser le système d'information, c'est le modèle conceptuel (MCD). La figure 1 donne un exemple de résultat avec le formalisme E-A (« Entités-Associations ») : on a 3 entités, *Station*, *Skieur* et *Compétition*, avec leurs attributs, ainsi que 2 associations, *est classé* et *A lieu à*, la première ayant pour attribut le rang de classement.

Le système d'information ne saurait être réduit aux seules entités et associations. *L'ensemble des contraintes* est également une composante essentielle de sa description. Les principaux types de contraintes conceptuelles sont :

- Comment un sous-ensemble d'attributs détermine les valeurs d'une occurrence d'entité, ce qui se traduit par la notion d'identifiant (la "clef" dans les BD relationnelles).

- Comment certains attributs d'une entité A sont liés à certains attributs d'une entité B, ce qui se traduit par la notion de cardinalité des associations.

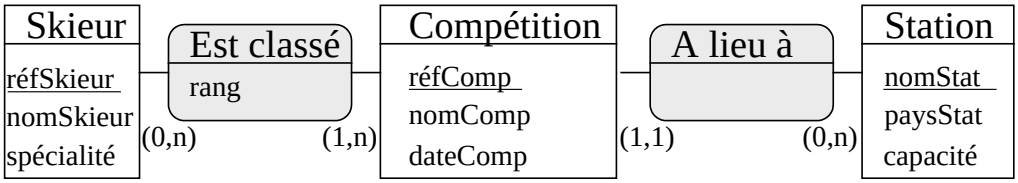

Figure 1. Un schéma E-A.

La cardinalité consiste en un couple d'informations : le nombre minimum d'occurrences de l'entité dans l'association (choix restreint à 0 ou 1), et le nombre maximum d'occurrences de l'entité dans l'association (choix restreint à 1 ou "plusieurs", le second cas étant noté "n").

Revenons à l'exemple de la figure 1 (notez que les identifiants des entités ont été soulignés) : le couple *(0,n*) du côté *Skieur* indique qu'un skieur participe à éventuellement 0 et en général plusieurs compétitions, tandis que *(1,n)* côté *Compétition* dénote qu'une compétition accueille au moins 1 et en général plusieurs skieurs. La cardinalité *(1,1)* côté *Compétition* exprime qu'une compétition a lieu dans au moins une station et au plus une station.

Une fois spécifié le schéma conceptuel du système d'information, il faut le traduire dans une représentation opérationnelle du point de vue informatique : à ce stade on peut utiliser soit un modèle orienté objet, soit un modèle hiérarchique, soit encore un modèle relationnel. Dans le cas du modèle relationnel, les tables de la figure 2 sont dérivées du schéma de la figure 1. On voit qu'un certain nombre de règles président à la traduction :

- Une entité se traduit par une table, chacun de ses attributs devenant une colonne de la table, et son identifiant devenant la *clef* (primaire) de la table.

- Une association avec un maximum égal à *n* des deux côtés se traduit par une table, dont la clef est formée des clefs de chaque table correspondant aux entités associées. Chacun de ses attributs devient une colonne de la table, cf. la table **Classement**.

- Une association avec maximum égal à *1* d'un côté se traduit par l'ajout, dans la table "dépendante", de la clef de l'autre table, cf. **nomStat** dans **Compétition**.

Compétition(réfComp, nomComp, dateComp, **nomStat**)
Skieur(réfSkieur, nomSkieur, spécialité)
Station(nomStat, paysStat, capacité)
Classement(réfComp, réfSkieur, rang)

Figure 2 : Tables dérivées du schéma E-A.

2.2 Retrouver le sens des données d'une base en exploitation

Le modèle relationnel est le modèle le plus largement répandu dans les bases de données actuelles. En relationnel « pur », on ne dispose que des relations (les tables avec leurs colonnes) : cela seul ne permet pas de retrouver le « sens » des données, c'est-à-dire *les liens* qui existent entre elles. Mais il a très vite été adjoint à la théorie relationnelle de quoi

exprimer ce sens : ce sont *les contraintes* (Abiteboul et al., 1995). Dans notre problématique, les contraintes exprimant le modèle conceptuel à l'origine de la base, nous nous en servons pour déterminer *l'image dans la base de données* des liens sémantiques trouvés entre les mots de la question (liens exprimés par les prédicats).

Il est possible de retrouver automatiquement les contraintes d'une base de données en exploitation : soit dans son catalogue (lequel contient les « méta-données ») si le concepteur a explicitement posé des contraintes sur la base, soit par des calculs sur le contenu des tables (Novelli et al., 2000). C'est le cas en particulier pour les dépendances fonctionnelles.

Dépendance fonctionnelle : Il y a une dépendance fonctionnelle lorsque la valeur d'un attribut (ou d'un groupe d'attributs) détermine de façon unique celle d'autres attributs. Nous appelons ***df*** un tel lien entre un attribut et les autres attributs *de la même table*. Dans notre exemple, la référence de la compétition en détermine la date et la station. De même la référence du skieur en détermine le nom et la spécialité, etc.

Dépendance d'inclusion : Il y a une dépendance d'inclusion lorsque les valeurs d'un attribut d'une table doivent appartenir à l'ensemble des valeurs d'un attribut *d'une autre table*. Nous appelons ***di*** un lien de dépendance d'inclusion. Par exemple, le nom de station de la compétition (dans *Compétition*) doit apparaître dans la colonne *nomStat* de *Station*. Cette information permet de déduire que, la référence de la compétition déterminant la station (dans *Compétition*), elle en détermine également le pays et la capacité d'accueil (dans *Station*).

Les contraintes n'expriment pas à elles seules tous les liens entre les données de la base : un certain nombre de ces liens ne sont pas *stockés* (statiques), mais *calculés* (dynamiques). Ainsi il existe dans tout SGBD des ***fonctions*** prédéfinies, appelées fonctions d'agrégat, pour calculer une somme, une moyenne, un maximum ou un minimum, ou encore compter un ensemble de valeurs. Elles vont servir par exemple à trouver la station offrant la plus grande capacité d'accueil, à déterminer la capacité moyenne des stations françaises, etc.

Les fonctions d'agrégat correspondent à des prédicats généraux de la langue : nous avons développé à ce sujet un certain nombre d'exemples dans (Bouchou et al., 1999). Il y a également des fonctions programmées par le concepteur de la base : par exemple dans une base d'inventaire de marchandise, il pourra y avoir une fonction pour calculer le stock (produits reçus - produits vendus) (Bouchou, Maurel, 1999). Qu'elles soient prédéfinies ou programmées, on peut trouver les fonctions d'une base dans de ses « méta-informations ».

En résumé, les informations que nous récupérons dans la base sont :

- les données : le contenu des tables,
- la structure générale : les noms des tables, les noms des colonnes,
- les liens entre les données : les dépendances (*df* et *di*), ainsi que les *fonctions* de calcul.

3. Le "sens" des données pour l'utilisateur qui interroge la base

3.1 Questions sur les « entités »

C'est un fait établi en IHM que tout utilisateur se forge un *modèle mental* de l'application informatique qu'il utilise (cf. par exemple l'ouvrage (Norman, 1986)). Ainsi l'utilisateur de la

base a son propre modèle du système d'information qu'elle représente lorsqu'il l'interroge (plus ou moins clair, complet, correct vis-à-vis de l'implantation effective de la base…).

En particulier, il en imagine les « entités » : c'est à leur propos qu'il va poser ses questions, lesquelles vont donc contenir des références à ces entités. Nous devons donc savoir comment l'utilisateur va faire référence à l'entité « *skieur* ». Il va peut être utiliser le mot *skieur* :

Quels skieurs participèrent au Championnat d'Europe 2000 ?

Pour le même genre de question, il peut aussi utiliser un terme représentant des compétiteurs, comme *concurrent* par exemple. Enfin, pour parler d'une occurrence particulière de skieur, il utilisera essentiellement les noms et prénoms de la personne :

Quelles compétitions ont été remportées par Franck Picard ?

Nous posons une définition pour prendre en compte cette dernière possibilité : nous appelons l'ensemble des valeurs caractéristiques utilisées dans la question pour parler d'une occurrence précise d'une entité le « *représentant* » de l'entité. Cela se traduit dans la base par un attribut, ou un ensemble d'attributs d'une table. Seules les tables qui correspondent à des entités ont un représentant[1]. Dans notre exemple, une station est représentée par son nom, une compétition par son nom et sa date, un skieur par son nom (qui regroupe nom et prénom).

Cette notion de représentant est nécessaire pour repérer dans la question *toute référence* à une table, soit directe, par ce qu'elle représente (« *la **station** qui…* » fait référence à la table *Station*), soit par une valeur de son représentant (« *la capacité d'accueil de **St Moritz*** » fait également référence à la table *Station*).

3.2 Questions sur les caractéristiques des entités

L'interrogation porte donc sur les tables qui correspondent aux entités que peut imaginer l'utilisateur. Plus précisément, elle porte sur l'une ou l'autre des *caractéristiques* des entités.

- Dans notre exemple, on interrogera sur le pays d'une station, sa capacité, la spécialité d'un skieur, la date d'une compétition, etc. On en déduit ainsi que *chaque attribut d'une table T correspond à une des caractéristiques possibles de l'entité représentée par T*.

Formellement, il y a dépendance fonctionnelle (*df*) entre chaque attribut non-clef et la clef de la table ; nous identifions la table (et son représentant) à sa clef. Ainsi, connaissant les *df*, nous connaissons une première partie des caractéristiques de l'entité qui pourront être interrogées.

- Reprenons encore notre exemple : d'autres caractéristiques de l'entité sont stockées en dehors de la table : le pays qui accueille la compétition, ou la performance (le rang) d'un skieur dans une compétition.

Ces caractéristiques-là sont "rapportées" par les dépendances d'inclusion (*di*).

---

[1] Parenthèse technique : cet ensemble d'attributs est souvent « clef » pour la table, mais c'est rarement la clef primaire effective. Pour l'instant, ce représentant doit encore être identifié par l'installateur.

- Enfin, il y a les caractéristiques qui ne sont pas stockées, mais calculées, à savoir les fonctions : le gagnant d'une compétition, par exemple, s'obtient par un MIN sur le rang.

Ainsi les images dans la base des caractéristiques interrogées s'expriment en termes de *df*, *di* et *fonctions*, lesquelles mettent en relation tables, attributs, et valeurs.

4. Projection de la question sur la base

4.1 Classes d'objets et prédicats appropriés pour représenter la question

Nous sélectionnons dans la question un certain nombre de mots qui fournissent assez d'informations pour construire une requête SQL pertinente vis-à-vis de la base. Pour cela, il faut non seulement les mots, mais leurs liens, soit encore l'ensemble des conditions qu'ils doivent remplir pour faire sens les uns avec les autres dans la question, ceci dans le contexte de la base. Ces liens, les travaux de Zellig Harris, puis de Maurice Gross (LADL) et, enfin, de Gaston Gross (LLI) montrent qu'ils sont donnés par les prédicats linguistiques : des opérateurs sur les phrases simples, dotés d'arguments (le sujet, les compléments)...

Les classes d'objets sont présentées par leurs auteurs comme « des classes *sémantiques* construites à partir de critères *syntaxiques* »[2] (Gross, 1998), (Le Pesant, Mathieu-Colas,1998), (Le Pesant, 2000). Il n'est pas dans notre propos de détailler leur cadre théorique, élaboré au LLI depuis une dizaine d'années, aussi nous ne faisons que rappeler leur définition. Une ***classe d'objets*** est définie comme représentant le « type » d'un argument de prédicat : en d'autres termes, le prédicat sélectionne son argument dans telle ou telle classe d'objets. Ainsi, on peut décrire le sens d'un mot prédicatif (verbe, nom, adjectif, ou adverbe) en indiquant les classes d'objets qu'il sélectionne. Et inversement, une classe d'objets est définie (en partie) par les prédicats qui lui sont spécifiques : ce sont ses ***prédicats appropriés***. Cela permet de décrire le sens d'un mot non prédicatif par l'énoncé des classes d'objets auxquelles il appartient.

Par exemple, *compétition* appartient à la classe des <*événements*> : ce mot est sélectionné par « *se dérouler* », « *être organisé* », « *avoir lieu* », etc.

(i) N0 *a eu lieu à* N1 *le* N2 ici N1 est de la classe des <toponymes> (auquel appartiennent les stations de ski, mais également les pays), et N2 de la classe des dates.

(ii) De même, le mot *skieur*, de la classe des <*humains*> (à ce titre il a un nom, donc il peut être sélectionné par « *être nommé* », « *avoir pour nom* »,…) est aussi de la (sous-) classe des <*sportifs*>, ce qui le rend sélectionnable par :

(iii) N0 *a pour spécialité* N1 ; *la spécialité de* N0 *est* N1.

(iv) Il est également dans la classe des <*compétiteurs*>, qualifiée par « *être vainqueur* », « *perdre* », « *se classer* », « *avoir pour rang* », etc.

(v) Le mot *station* relève (entre autres) de la classe des <*villes*>, sélectionnées en particulier par : *N0 est dans <pays>*, *N0 en <pays>*, *N0 de <pays>*. Il appartient

[2] C'est en ceci nous semble-t-il que la démarche du LLI diffère de celle du DEC de Mel'cuk (Mel'cuk, 1995), dans lequel la plupart des propriétés lexicographiques d'une lexie *découlent* des définitions sémantiques.

également aux lieux géographiques, caractérisés entre autres par leur altitude : *N0 se situe à <altitude>*.

Dans les faits, les classes d'objets sont définies en extension, par l'ensemble des mots qui les composent, et l'ensemble de leurs prédicats appropriés (verbes, adjectifs, noms ou adverbe). Chaque mot est lui-même associé à des caractéristiques lexico-sémantiques : genre, trait, domaine, etc. Les prédicats sont décrits également par leurs caractéristiques : catégorie grammaticale, classes des arguments , trait, domaine, etc.

4.2 L'image dans la base des classes d'objets et prédicats appropriés

Les classes d'objets nous servent à construire le dictionnaire, en même temps que les informations issues de la base de données : ce dictionnaire met en relation des mots avec des ensembles de codes, comme le schématise la figure 3.

Figure 3 : À un mot correspond une liste de codes

Quels sont les mots en entrée du dictionnaire, mots qu'il faudra reconnaître dans la question ? C'est la base de données qui détermine ce vocabulaire : on doit pouvoir reconnaître :

- Toute référence aux entités (tables).
- Toute référence aux instances (valeurs stockées).
- Toute référence aux caractéristiques de telle ou telle entité (relations entre attributs d'une table *–df–*, ou relations entre attributs de différentes tables *–di–*, ou *fonctions*).

Reprenons les exemples *(i)* à *(v)* précédents :

- Les mots qui parlent d'une **table** appartiennent à une (ou plusieurs[3]) classe(s) d'objets linguistiques. De même, les **valeurs** (textuelles) prises par un attribut appartiennent à une (ou plusieurs) classe(s) d'objets : *St Moritz*, *Genève* ou *Lausanne* sont des éléments de la classe des noms de villes.
- Considérons maintenant les **attributs** d'une table T : chacun correspond à une caractéristique de l'entité représentée (même partiellement) par T. Chacun représente donc un mot prédicatif, qui sélectionne au moins deux arguments : le premier dans la classe d'objets de la table, le deuxième dans la classe d'objets des valeurs prises par l'attribut. Voyez par exemple *(i)* : *N0 a lieu à N1*, ou encore *(iii)* et *(iv)*.
- Lorsqu'une entité est représentée par un ensemble de tables, les attributs d'une table T1 qui dépendent des valeurs d'une table T2 correspondent également à des mots prédicatifs, lesquels sélectionnent un argument dans la classe d'objets de la table T2, et les autres dans les classes des valeurs prises par l'attribut de T1. Ainsi la relation entre

[3] Du fait de la hiérarchie entre classes d'objets, un mot peut apparaître dans une classe et une sous-classe.

l'attribut *pays* de la table *Station* avec la table *Compétition* représente la forme *N0 a lieu en N1* : *le championnat d'Europe a lieu en Autriche*.

- Par ailleurs les fonctions (nommées selon le « sens » de leur résultat : *minimum*, *somme*, *stock*, *vainqueur*…, et caractérisées par leurs arguments et leur type) sont, elles aussi, référencées dans la langue par des mots prédicatifs. Par exemple la fonction qui calcule un minimum représente la forme superlative *le moins*, ou *le plus petit* : *l'altitude la moins élevée, la plus petite capacité d'accueil.*

Ainsi les informations (valeurs stockées) et les méta-informations (tables, attributs, contraintes et fonctions) de la base de données conditionnent l'ensemble des mots à reconnaître dans la question. À partir de ces conditions posées par la base, notre démarche consiste ensuite à *consulter la langue*, structurée en classes d'objets et de prédicats, pour déterminer les mots qui seront utilisés pour l'interrogation.

Spécifions maintenant les codes à associer aux mots du dictionnaire. Un mot est en entrée du dictionnaire en tant qu'élément d'une classe d'objet et/ou élément d'une classe de prédicats.

- Si c'est un élément d'une classe d'objets, alors c'est une référence soit à une table, soit à une valeur d'un attribut d'une table[4]. Donc au moment où on ajoute le mot au dictionnaire, on connaît la table ou la valeur d'attribut en question : on associe alors à ce mot une référence à cette table (code *T.nom_interne_de_la_table*), ou à cette valeur (de cet attribut de cette table : code *I.nom_de_la_table.nom_ de_l'attribut.valeur*).

- Si c'est un mot prédicatif, alors il est en entrée du dictionnaire du fait d'une *df*, d'une *di*, ou d'une *fonction*. De la même façon on peut lui associer un code qui représente cette information dans la base. Par exemple, « *se déroule* » est associé au code *df(T.Compétition, A.Compétition.nomStat)* au moment où cette *df* est traitée, puis au code *di(T.Compétition, A.Station.paysStat)* lorsque cette *di* est traitée à son tour. Ce terme, « *se déroule* », est donc en entrée du dictionnaire du fait des relations *df* et *di*, ET grâce aux classes d'objets et de prédicats associés, qui fournissent *les mots de la langue*[5] qui expriment ces relations.

On voit donc qu'à chaque mot est associée *une liste* de codes, ne serait-ce que parce qu'il est rare qu'un mot n'ait qu'un emploi, même dans un certain domaine. Le traitement d'une question par le premier transducteur (qui applique le dictionnaire) génère *une liste de listes* de codes. L'essentiel du traitement ultérieur consiste à mettre en regard les sous-listes : par exemple, s'il y a le code *df(T.Compétition, A.Compétition.nomStat)* dans une sous-liste S1, et s'il y a le code *T.Compétition* dans une sous-liste S2 et enfin le code *A.Compétition.nomStat* dans une sous-liste S3, alors *df(T.Compétition, A.Compétition.nomStat)* est retenu, et ce code n'est retenu qu'à cette condition : nous ne retenons que les dépendances et fonctions « *instanciées* » par une partie ou une autre de la question.

Bien évidemment, les principes qui viennent d'être décrits ne sont applicables qu'en cas d'attributs textuels : les booléens, les nombres et les dates ne sauraient se trouver en entrée du dictionnaire, et portent peu de sémantique en eux-mêmes. Ces types d'attributs font l'objet d'un traitement distinct que nous n'aborderons pas ici.

---

[4] Table ou valeur qui sont à l'origine de la présence de ce mot en entrée du dictionnaire.

[5] Et les règles qui régissent leur emploi, donc leur sens…

5. Construction du dictionnaire avec les classes d'objets

Dans la phase d'installation, il faut que les choses soient simples. Dans (Bouchou et al., 1999), il est montré que notre système fait grosso-modo ce que le système TEAM faisait déjà en 1986 avec une approche de type intelligence artificielle (Grosz, 1986). Mais TEAM demandait trop d'efforts et de compétences pour son adaptation à chaque nouvelle base. Pour nous, compte-tenu de ce que nous extraions de la base elle-même (dont les dépendances et les fonctions), et de la connaissance de la langue fournie par les classes d'objets, il faut peu d'efforts et peu de temps à l'installateur. Voici quelques points qui tendent vers cet objectif :

Au départ, l'installateur donne :

- un mot de la langue courante pour désigner chaque table (le nom interne des tables est rarement parlant),
- le « représentant » des tables qui correspondent à des entités.

Pour chaque nom de table donné, *T*, les classes d'objets auxquelles il appartient sont recherchées, et soumises à l'installateur pour qu'il désigne la plus pertinente, *CoT*[6].

Par ailleurs, les classes d'objets auxquelles correspondent les valeurs de chaque attribut – textuel– de chaque table sont également recherchées. Chaque attribut A_i de *T* correspond à une colonne de valeurs : on recherche la plus petite classe d'objets qui contienne toutes ces valeurs. Nous avons ici besoin de la hiérarchie des classes : si au plus petit niveau (le plus précis) aucune classe ne contient toutes les valeurs de la colonne, alors il faut sélectionner une classe d'un niveau supérieur. Soit $CoT.A_i$ la classe ainsi déterminée pour l'attribut A_i de *T*.

Des techniques similaires sont utilisées pour les *df*, *di* et les *fonctions* : ainsi pour les *df*, on recherche la classe de prédicats à laquelle correspond un attribut A_i de la table *T*. Il s'agit de l'ensemble des prédicats qui sélectionnent *à la fois* un élément de la classe *CoT* et un élément de la classe $CoT.A_i$. On note cet ensemble : $CpT.A_i = \{P/(CoT, CoT.A_i , P)\}$.

Grâce à de tels calculs, opérés automatiquement sur les fichiers qui décrivent les classes d'objets et les prédicats, l'installateur, après avoir « amorcé » l'installation avec un nom pour chaque table, n'a plus qu'un simple rôle de sélection ou de validation.

6. Bilan et perspectives

Nous ne pensons pas avoir épuisé les ressources offertes par les classes d'objets et prédicats associés. Les techniques qui viennent d'être présentées sont implantées dans notre prototype, développé en Java. Celui-ci est testé sur plusieurs bases, de domaines différents (la gestion de stock, les vins, …), et l'expérience mérite d'être prolongée pour intégrer des propositions concernant l'organisation de la classification, comme l'usage des domaines et sous-domaines.

Une perspective importante ouverte par ce travail, pour l'instant focalisé sur les bases de données relationnelles, consiste à transposer les réflexions qui le sous-tendent vers des

---

[6] Les classes sont "soumises" à travers un échantillon de leurs prédicats appropriés.

collections de données semi structurées, par exemple en XML, et à ouvrir notre application, de ce fait, à la recherche d'informations sur le Web.

Remerciements

Les auteurs tiennent à remercier M. Gaston Gross pour leur avoir permis de tester les idées présentées ici sur des fichiers de classes d'objets et de prédicats développés au LLI.

Références

Abiteboul S., Hull V., Vianu V. (1995), *Foundations of Databases*, Addison-Wesley.

Bouchou B., Maurel D. (1999), Une bibliothèque d'opérateurs linguistiques pour la consultation de base de données en langue naturelle, *6ème Conférence sur le Traitement Automatique des Langues naturelles (TALN 1999*), Cargèse, actes pp 65-74.

Bouchou B., Maurel D., Kaltz B. (1999), Prédicats logiques / prédicats linguistiques pour la consultation de base de données en langue naturelle, *RISSH*, U. de Liège, Vol.35, pp 127-149.

Gross G. (1994), Classes d'objets et description des verbes, *Langages* 115, Larousse.

Gross G. (1998), Pour une véritable fonction synonymie dans un traitement de texte, *Langages* 131, Larousse.

Gross M. (1975), *Méthodes en syntaxe*, Hermann, Paris.

Grosz B., Appelt D., Martin P., Peirera F.(1987), TEAM : an experiment in the design of transportable natural language interfaces, *Artificial Intelligence 32*, pp 173-243.

Harris Z.S. (1976), *Notes du cours de syntaxe*, Paris, Le Seuil.

Kaplan S. J. (1984), Designing a Portable Natural Language Database Query System, *ACM Transactions on Database Systems*, vol. 9, n°1, march 1984, pp. 1-19.

Le Pesant D., Mathieu-Colas M. (1998), Introduction aux classes d'objets, *Langages* 131, Larousse.

Le Pesant D. (2000), Introduction aux classes d'objets, *Thèse d'habilitation à diriger des recherches*, Université de Villetaneuse, décembre 2000.

Mel'cuk I. A., Clas A., Polguere A. (1995), *Introduction à la lexicologie explicative et combinatoire*, "champs linguistiques", Louvain La Neuve, Duculot.

Mihov S., Maurel D. (2000), Direct Construction of Minimal Acyclic Subsequential Transducers, First Conference on Implementing and Application of Automata (CIAA'2000).

Norman, D. A., Draper, S. (Eds.), (1986). *User Centered System Design: New Perspectives on Human-Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Novelli N., Cicchetti R. (2000), Mining functional and embedded dependencies using free sets, *16ème Conférence Bases de Données Avancées (BDA 2000), Blois*.

Pasero R., Sabatier P. (1999), ILLICO : un système générique pour la compréhension d'un sous-ensemble du français, rapport Laboratoire d'Informatique de Marseille, 1999.

Revuz D. (1991), *Dictionnaires et lexiques - Méthodes et algorithmes*, Thèse de Doctorat en Informatique (Université Paris7).

Sabah G. (1997), Le sens dans le traitement automatique des langues, *T.A.L.* 38-2, pp. 91-133.