

# Une Ontologie De Domaine Pour L'enrichissement Sémantique D'une Base De Données

Ines FAYECH\* et Habib OUNALLI\*

\*URPAH, Faculté des sciences de Tunis Université de Tunis El Manar, TUNISIE

inesfaiech@yahoo.fr

Habib.ounalli@fst.rnu.tn

**Résumé:** Cet article traite un problème dans le domaine de la gestion des bases de données classiques. Il s'agit d'exploiter une ontologie de domaine pour aider l'utilisateur d'une base de données relationnelle dans sa recherche et de lui permettre une interrogation transparente de la base de données. Pour cela, nous proposons une approche d'expansion automatique de requêtes SQL lorsque celles-ci n'ont pas de réponses. Notre approche est décrite par un algorithme défini de manière générique afin d'être utilisé pour une base de données quelconque.

**Mots clés:** Base de données relationnelle, Approche d'interrogation, Ontologie de domaine, Expansion de requêtes.

## INTRODUCTION

Les systèmes de gestion des bases de données relationnelles permettent de structurer les données mais, leurs pauvretés sémantiques constituent un handicap pour répondre aux nouveaux besoins des utilisateurs. En effet, pour exprimer son besoin, l'utilisateur d'une base de données relationnelle doit spécifier, dans une syntaxe précise, les éléments du schéma qui y sont impliqués et savoir exploiter les jointures existantes entre eux. Pour des utilisateurs non initiés, qui ignorent la structuration des données, il est nécessaire de trouver d'autres moyens pour rendre l'interrogation plus conviviale.

Avec l'avènement des ontologies, un progrès important a pu être réalisé grâce à la représentation explicite de la signification des données [GRU 93]. Dans ce cadre, nous nous intéressons à leur exploitation pour aider l'utilisateur d'une base de données relationnelle dans sa recherche et lui permettre une interrogation transparente de la base de données.

Dans [FAY 08], nous avons proposé une approche ontologique d'expansion de requêtes SQL mono-tables lorsque celles-ci n'ont pas de réponses. L'utilisation d'une telle ontologie pour le raffinement des requêtes utilisateurs, nous a permis de traiter le problème de la spécification de la requête. En effet, elle permet à l'utilisateur d'exprimer son interrogation de façon relativement libre, sans être obligé d'utiliser les mêmes termes contenus dans la base de données.

Dans ce papier, nous étendons cette approche pour

reformuler des requêtes SQL multi-tables et nous abordons le problème de la construction de l'ontologie utilisée. Si la requête initiale n'admet pas de réponse, elle sera reformulée par notre approche afin de mieux satisfaire l'utilisateur. Nous utilisons une expansion automatique qui est réalisée sans que l'utilisateur soit sollicité. Elle est décrite par un algorithme que nous avons élaboré de manière générique afin d'être utilisée pour une base de données relationnelle quelconque.

Le reste du papier est organisé comme suit : Nous commençons par justifier, en section 1, le choix de l'approche d'interrogation. La section 2 décrit les techniques d'expansion de requêtes tout en présentant leurs différentes classifications ainsi qu'une synthèse des différents travaux d'expansion de requêtes en utilisant les ontologies. La section 3 est consacrée à la présentation de l'ontologie de domaine que nous utilisons dans notre approche. Dans la section suivante, nous détaillons l'approche proposée. Des exemples d'application de notre proposition sont cités dans la section 5. Enfin, nous terminons avec une conclusion et quelques directions de recherches pour les travaux futurs.

## 1. Approche d'interrogation : Pourquoi le langage SQL ?

Dans la littérature, de nombreux travaux ont investi la mise en œuvre de stratégies afin d'aider l'utilisateur dans sa recherche. Ces solutions peuvent être classées selon l'approche d'interrogation utilisée. Nous distinguons les travaux utilisant une approche visuelle basée sur des éléments graphiques, ceux qui utilisent une approche d'interrogation basée

sur un langage naturel et ceux exploitant un langage de requêtes tel que SQL.

L'approche basée sur les langages naturels reste difficile à être efficacement exploitable du fait qu'elle présente deux limites relatives à l'imprécision et aux ambiguïtés sémantiques.

Pour l'approche visuelle, l'interrogation se construit, progressivement, par la manipulation des objets présentés au niveau de l'interface sous différentes formes graphiques. Plusieurs systèmes visuels d'interrogation ont été proposés dans la littérature avec différentes représentations visuelles.

Les systèmes QBE [ZLO 77] et TABLETALK [EPS 91] utilisent une représentation tabulaire, les systèmes ICONIC BROWSER [TSU 90] et MEDIABENCH [TON 89] utilisent les icônes et les systèmes QBD\* [ANG 90] et SUPER [AUD 91] utilisent les diagrammes. D'autres systèmes tels que SICON [GRO 88] utilisent une composition de ces représentations.

L'évaluation d'une approche d'interrogation dépend de son utilisabilité. Dans la littérature, peu de travaux ont été proposés, visant à tester et à valider l'efficacité et la facilité d'utilisation des systèmes d'interrogation. Nous distinguons les travaux visant à évaluer la facilité d'utilisation d'un langage donné et celles visant à établir une étude comparative entre plusieurs langages. Parmi ces derniers, nous citons l'étude comparative entre le système QBD\* et le système QBI [BAD 96], le système QBE et le langage SQL [YEN 93] et celle entre le système QBD\* et le langage SQL [CAT 95].

Cette dernière expérience a été réalisée par différentes classes d'utilisateurs (simple utilisateur, niveau intermédiaire, experts) qui ont été appelés à formuler une liste d'interrogation écrite en langage naturel, en utilisant QBD\* et SQL. Les résultats de l'expérience ont prouvé que les utilisateurs ont trouvé que l'environnement QBD\* est plus attractif et plus facile à utiliser. En effet, l'utilisateur n'a pas à se rappeler des noms des tables et de leurs attributs mais il suffit de les choisir en naviguant dans le schéma.

Bien que le système QBD\* présente des avantages par rapport au langage SQL, il manque de portabilité. En effet, pour être greffé à un système d'information quelconque, il faut que le modèle conceptuel correspondant soit fourni et adapté pour être supporté par QBD\*. Par conséquent, en cas d'évolution de la structure de la base de données, il est indispensable de mettre à jour le modèle afférent afin que ces changements puissent être pris en compte par le système d'interrogation.

Les systèmes visuels ont facilité plusieurs manipulations informatiques jugées ambiguës par un utilisateur non initié. Cependant, leurs utilisations deviennent de plus en plus difficiles au fur et à mesure que le nombre d'objets manipulés devient important. Dans ce cas, l'utilisateur peut se perdre facilement le long du procédé d'interrogation.

Quant au langage SQL, bien qu'il soit supporté par la plupart des produits commerciaux, il ne peut pas définir des descriptions sémantiques de données. En effet, pour écrire une requête SQL, l'utilisateur d'un tel langage doit connaître les noms des tables et ceux de leurs attributs ainsi que les relations qui les relient.

Pour pallier à cette lacune, d'autres langages de requêtes, appelés langages ontologiques, ont été proposés. Ce sont des extensions du langage SQL en ajoutant une couche supplémentaire au dessus d'un SGBD classique par l'exploitation des ontologies qui apportent la signification des données. Parmi ces langages, nous citons le langage d'interrogation du web sémantique OWL-QL et le langage OntoQL [JEA 04] qui est proposé dans l'architecture OntoDB [DAH 03] pour exploiter les bases de données (appelées bases de données à bases ontologique) qu'elle supporte. Le choix de ces langages ontologiques exige que l'utilisateur ait des connaissances sur l'ontologie utilisée.

Afin de définir des solutions qui soient compatibles avec l'existant et de permettre une interrogation transparente de la base de données, nous considérons le langage SQL comme approche d'interrogation et nous proposons une approche d'expansion de requêtes SQL lorsque celles-ci n'ont pas de réponses. Notre approche est basée sur l'utilisation d'une ontologie qui apporte la représentation explicite de la signification des données.

## 2. L'expansion des requêtes

La spécification des éléments d'une requête dépend de la connaissance des données et de leurs structurations. En effet, les utilisateurs qui sont familiarisés avec le système connaissent les données disponibles et sont donc capables d'exprimer leurs besoins en terme d'interrogation suivant une démarche donnée. Quant aux utilisateurs non familiers, qui ont seulement une idée générale à propos du contenu de la base de données, ils peuvent mal formuler leurs requêtes. Ce problème naît du fait qu'une même idée peut être formulée de plusieurs manières différentes. Dans ce cas, il est nécessaire d'avoir un système qui assiste ces utilisateurs pour les aider à extraire le maximum d'informations susceptibles de les intéresser et satisfaisant leurs besoins.

Une des solutions envisagées en recherche d'information pour couvrir la diversité des formulations d'un même besoin, est l'expansion de requêtes. Il s'agit d'une approche classique de reformulation de requêtes par ajout de termes liés sémantiquement à ceux de la requête initiale [HER 05]. Il a été prouvé dans la littérature que la reformulation de requêtes a des effets positifs en recherche d'information [HER 05]. En effet, les mécanismes de reformulation de requêtes permettent d'améliorer la représentation de la requête.

Dans ce qui suit, nous allons présenter les différentes classifications des techniques d'expansion

de requêtes ainsi que certains travaux d'expansion de requêtes en utilisant les ontologies.

### 2.1. Les différents types d'expansion de requêtes

En considérant les relations sémantiques de synonymie, de spécialisation et de généralisation, nous distinguons les trois types d'expansion de requêtes : l'expansion par synonymie, par spécialisation et par généralisation.

- **L'expansion par synonymie** permet d'avoir une requête reformulée qui est équivalente à la requête initiale. En effet, elle se base sur l'utilisation des synonymes ce qui garantit que la reformulation ne change pas la sémantique de la requête initiale.

- **L'expansion par spécialisation** représente une spécialisation de la requête initiale. Par conséquent, l'ensemble des solutions générées à partir de la nouvelle requête est inclus dans l'ensemble des solutions pertinentes pour l'utilisateur. Ce raffinement peut contribuer à l'augmentation du nombre de solutions et par la suite, à une augmentation de la précision. Il a été exploité dans le cas d'une réponse vide à une requête [MES 06], [BID 02] ou lorsque l'utilisateur obtient trop de réponses à sa requête [SAF 04].

- **L'expansion par généralisation** consiste à remplacer chaque concept de la requête initiale par des concepts plus généraux présents dans l'ontologie. Cette approche a été exploitée dans [BID 02] où un processus de généralisation de la requête initiale est déclenché si elle n'a pas de réponse.

Une requête reformulée par généralisation donne généralement des solutions plus générales et moins précises par rapport à ce qui est souhaitée par l'utilisateur. Ceci s'explique par le fait que dans la requête reformulée, certains concepts généralisés peuvent contenir des informations qui contredisent le concept spécialisé présent dans la requête initiale. Dans ce cas, la requête reformulée s'éloigne du besoin de l'utilisateur. Donc, cette approche peut contribuer à augmenter le nombre de solutions non pertinentes restituées, chose qui s'oppose à notre objectif.

Afin d'éviter ce problème, nous avons choisi de ne pas appliquer ce type de raffinement dans notre travail.

Une autre classification des mécanismes d'expansion a été proposée dans la littérature en fonction de l'implication de l'utilisateur dans la reformulation. Elle aboutit à deux types d'expansion de requêtes qui sont l'expansion interactive faisant intervenir l'utilisateur dans le processus de reformulation et l'expansion automatique qui est réalisé sans que l'utilisateur soit sollicité [KHA 00].

La plupart des travaux d'expansion de requêtes utilisent l'expansion interactive [MES 06], [SAF 04]. Par exemple, dans [MES 06] une ontologie de domaine est utilisée dans une approche interactive d'expansion de requêtes pour résoudre un problème de recherche d'information en bioinformatique. Les

auteurs ne précisent aucun critère de décision de l'enrichissement possible et ils laissent le choix pour l'utilisateur. Ceci n'est pas possible dans plusieurs cas. En effet, l'utilisateur ne peut pas décider du type de raffinement adéquat sans voir le résultat de chaque reformulation.

Afin d'améliorer les résultats de la recherche sans que l'utilisateur ait à intervenir, nous avons choisi d'appliquer une expansion automatique.

### 2.2. L'utilisation des ontologies pour l'expansion de requêtes

Dans la littérature, plusieurs travaux ont été définis pour reformuler des requêtes en exploitant une ontologie. Parmi ces travaux, nous citons ceux qui ont utilisé une ontologie linguistique telle que Sensus [SWA 96], CYC [LEN 95] et WordNet [MIL 95]. Cette dernière est la plus utilisée bien qu'elle n'utilise que la langue anglaise. Dans [BAZ 03a], les auteurs proposent une approche de reformulation de requêtes basée sur les relations linguistiques. Il s'agit d'étendre les termes de la requête initiale à tous les termes de sa famille morphologique. Soualmia et Darmoni ont proposé dans [SOU 03] une approche similaire en utilisant un lexique. Une autre approche de reformulation de requêtes est proposée dans [BAZ 03b]. Elle exploite les liens sémantiques tels que la synonymie, l'hyponymie et l'hyperonymie, pour améliorer la recherche d'information. Une approche similaire est présentée par Voorhees qui a testé différentes combinaisons des relations sémantiques extraites à partir de WordNet pour étendre des requêtes [VOO 94].

Le choix d'une ontologie linguistique présente des limites puisqu'elle est très générale. Par conséquent, les termes générés par enrichissement d'un concept de la requête peuvent être nombreux et ne correspondent pas tous à la sémantique recherchée par l'utilisateur. Ces différents termes, une fois rajoutés à la requête peuvent donner des réponses non pertinentes pour l'utilisateur chose qui s'oppose avec l'objectif de l'expansion de requêtes.

D'autres travaux d'expansion de requêtes ont exploité une ontologie de domaine décrivant le vocabulaire relatif à un domaine générique [MES 06], [SAF 04]. Dans [MES 06], les auteurs ont proposé une approche combinant l'analyse de concepts formels et les ontologies de domaine pour résoudre un problème de recherche d'information en bioinformatique. Ils ont exploité l'expansion par généralisation et/ou par spécialisation pour générer les sources de données dans le cas où la réponse à la requête initiale est vide. Leur approche présente des limites dans le choix du raffinement adéquat. En effet, ils ne précisent aucun critère de décision de l'enrichissement possible (par généralisation ou par spécialisation). En plus, le choix non justifié du raffinement par généralisation peut conduire à des résultats non pertinents pour un utilisateur non intéressé par le résultat de l'élargissement de sa requête.

Le choix d'une ontologie de domaine présente des avantages et peut pallier aux lacunes d'une ontologie linguistique. En effet, les termes considérés dans l'expansion basée sur une ontologie de domaine sont tous issus d'un même domaine ce qui permet de réduire le risque d'avoir des réponses non pertinentes.

### 3. L'ontologie de domaine utilisée

Nous avons choisi d'exploiter une ontologie de domaine de la base de données permettant de retrouver la sémantique attachée aux termes de la requête afin d'identifier leurs équivalents dans la base de données malgré leurs représentations différentes.

Dans la littérature, plusieurs définitions d'une ontologie de domaine ont été proposées [GUA 95], [BAC 05], [BEN 03b]. D'après [GUA 95], une ontologie de domaine vise à représenter d'une manière générique et réutilisable, la sémantique d'un domaine donnée. Elle permet la compréhension du domaine qu'elle modélise. Selon les auteurs de [BEN 03a], un même domaine peut être décrit par plusieurs ontologies de domaines, chacune étant relative à un contexte particulier.

Dans notre travail, nous considérons une ontologie de domaine couvrant les termes qui représentent les noms utilisés dans la base de données pour désigner les tables et leurs attributs. En effet, ces termes peuvent ne pas correspondre à des mots en langage naturel.

#### 3.1. La construction de l'ontologie de domaine :

Bien que le choix d'une ontologie de domaine permette de pallier aux lacunes d'une ontologie générique, il pose le problème de sa construction. Ce dernier, a été traité dans la littérature et différentes solutions ont été proposées en fonction du besoin [BAR 04], [BIZ 03], [CUL 07], [MHI 05]. D'après [BEN 03a], il existe deux approches pour la construction d'une ontologie. Elle peut débuter soit à partir de zéro ou à partir des bases de données déjà existantes afin de capturer une certaine réalité des concepts.

Dans notre travail, nous partons d'une ontologie minimale du domaine de la base de données. Elle représente une hiérarchie de concepts structurés par la relation sémantique de spécialisation. Chaque concept a un nom et peut avoir un ou plusieurs termes qui le désignent.

Par la suite, nous allons enrichir cette ontologie minimale en rajoutant d'autres concepts et termes afin de couvrir tous les termes contenus dans la base de données. Pour cela, nous exploitons le dictionnaire de données relatif au SGBD utilisé afin de dégager la liste des tables de la base de données et leurs colonnes (L-DB) ainsi que la liste des synonymes (L-SYN).

Cette dernière est utile pour décider de l'insertion de chaque terme T dans L-DB comme étant un nouveau concept dans l'ontologie minimale ou comme une mise à jour de la liste des termes d'un concept

existant. Ainsi, lorsque le terme T ne fait pas partie de l'ontologie, nous distinguons les deux cas suivants :

Si, dans la liste L-SYN, il existe un terme synonyme du terme T, qui est présent dans l'ontologie et qui désigne un concept C. Dans ce cas, il suffit de rajouter le terme T comme étant un nouveau terme désignant aussi ce concept C.

Sinon, il faut rajouter un nouveau concept dans l'ontologie. Ce concept est désigné par le terme T et par son synonyme dans la liste L-SYN.

#### 3.2. Spécification formelle de l'ontologie

Nous avons choisi de représenter l'ontologie de domaine par une base de connaissance décrite sous la forme de clauses de Horn. Elle permet la représentation des constituants de l'ontologie qui sont les concepts et les termes et des liens entre eux.

Nous considérons les prédicats suivants pour la représentation des concepts et des termes :

- concept (C) signifie que C est un concept.
- terme (T) signifie que T est un terme.
- désigne(C, T) signifie que le terme T désigne le concept C.

Pour exprimer les liens sémantiques entre les concepts et les termes utilisés dans un domaine, nous définissons les prédicats suivants :

- La relation sémantique de synonymie entre deux termes T1 et T2 exprime que T1 et T2 ont le même sens. Le prédicat binaire synonyme() permet de vérifier cette relation. Il est défini par :

$$\begin{aligned} \text{synonyme}(T1, T1) &\leftarrow \\ \text{synonyme}(T1, T2) &\leftarrow \text{terme}(T1), \text{terme}(T2), \\ &\text{concept}(C), \text{désigne}(C, T1), \text{désigne}(C, T2) \end{aligned}$$

La relation sémantique de synonymie est symétrique et transitive. Par la suite, le prédicat synonyme() vérifie les propriétés suivantes:

La symétrie :

$$\text{synonyme}(T1, T2) \leftarrow \text{synonyme}(T2, T1)$$

La transitivité :

$$\text{synonyme}(T1, T2) \leftarrow \text{synonyme}(T1, T3), \\ \text{synonyme}(T3, T2)$$

- La relation sémantique de spécialisation entre deux concepts C1 et C2 exprime que le concept C1 spécialise le concept C2. Nous utilisons un prédicat C-Spec() pour vérifier cette relation. Ce prédicat vérifie la propriété de transitivité suivante :

La transitivité :

$$\text{C-Spec}(C1, C2) \leftarrow \text{C-Spec}(C1, C3), \text{C-Spec}(C3, C2)$$

- La relation de spécialisation entre deux termes est exprimée par le prédicat T-Spec().

Deux termes sont reliés par une relation de spécialisation, si les concepts qui les désignent les sont aussi. La définition du prédicat T-Spec est alors:

$$\text{T-Spec}(T1, T2) \leftarrow \text{terme}(T1), \text{terme}(T2),$$

concept (C1), concept (C2), designe (C1, T1), designe (C2, T2), C-Spec(C1, C2)

#### 4. L'approche proposée

Notre objectif est d'aider l'utilisateur d'une base de données relationnelle dans sa recherche.

Etant donné une base de données relationnelle BD et une ontologie ONTO couvrant le domaine de la base, nous proposons une approche ontologique d'expansion d'une requête SQL lorsque celle-ci n'a pas de réponse. Nous considérons un enrichissement automatique qui s'effectue par les liens sémantiques de synonymie et de spécialisation présents dans l'ontologie considérée. L'expansion peut s'appliquer sur les tables et/ou sur les colonnes indépendamment de leurs emplacements dans la requête initiale (dans la clause SELECT ou dans la clause WHERE). Elle n'est pas uniforme pour tous les termes de la requête. En effet, nous avons privilégié certaines formes d'expansion en dépend d'autres. Ainsi les relations ontologiques de synonymie entre les termes associés à un même concept sont prioritaires par rapport aux relations de spécialisation.

Avant de présenter l'algorithme proposé, nous allons décrire, ci-après, l'expansion des colonnes ainsi que celle des tables.

##### 4.1. L'expansion des colonnes relatives à une table

L'expansion de l'ensemble Col des colonnes d'une table T consiste à déterminer un ensemble C de termes dont chacun est synonyme ou égale à un élément de Col et représentant un attribut de la table T.

Nous décrivons ce type d'expansion par la procédure `expansion_colonnes()` qui retourne faux si l'expansion échoue (c'est-à-dire qu'il existe au moins un terme de Col qui n'a pas de correspondant dans la base de données).

##### 4.2. L'expansion d'une table

L'enrichissement d'une table Tab consiste à vérifier pour ce terme introduit dans la clause FROM, s'il correspond à une table de la base de données. Si ce n'est pas le cas, il faut rechercher à le remplacer par d'autres termes qui les relient sémantiquement et qui représentent des tables de la base de données. Nous distinguons deux types d'expansion : Le premier est l'expansion de la table Tab par synonymie qui est appliquée dans le cas où Tab ne référence pas une table dans la base de données. Ce raffinement est utilisé par la procédure `trouver_table_correspondante()` qui génère un nouveau terme Tab1, synonyme ou égale à Tab et référençant une table de la base de données.

Le deuxième type d'expansion se base sur la relation sémantique de spécialisation et s'applique dans l'un des deux cas suivants :

1. Il n'existe aucune table, dans la base de données, synonyme ou égale à Tab.

2. Le terme Tab représente (synonyme ou égale) une table Tab1 de la base de données (dans ce cas nous remplaçons Tab par Tab1). Cependant, il n'existe pas de correspondance entre les colonnes de la requête et les attributs de la table Tab1 (c'est-à-dire que la procédure `expansion_colonnes()` retourne la valeur faux).

Ce type d'expansion est appliqué dans la procédure `trouver_table_spécialisés()` qui génère un ensemble LTab1 de termes spécialisant le terme Tab et référençant des tables dans la base de données. L'expansion des colonnes, appliquée à chaque élément de LTab1, peut générer un ensemble de requêtes dont l'exécution satisfait l'utilisateur.

L'expansion d'une table Tab ainsi que celle de ces colonnes Col (Col est un tableau de taille ncol) sont décrites par la procédure `Reformulation_monotable()` suivante :

**Procédure** `Reformulation_monotable` (ncol, Col, Tab, BD, ONTO, LReq)

**Début**

LReq = Nil

`trouver_table_correspondante` (Tab, BD, ONTO, Tab1)

**Si** Tab1 existe **Alors** Tab = Tab1

ok = `expansion_colonnes` (Tab, Col, ncol, ONTO, BD, C)

**Si** ok **Alors**

{/\* insérer une nouvelle requête dans LReq en remplaçant dans ReqInit la table Tab par sa nouvelle valeur et chaque élément de Col par son équivalent dans C/\*}

`Insérer_Requête`(ReqInit, LReq, Tab, C, ncol)

**fin\_si**

**fin\_si**

**Si** LReq = nil **Alors**

`trouver_table_spécialisés` (Tab, BD, ONTO, LTab1)

**Pour** chaque T1 dans LTab1 **faire**

ok = `expansion_colonnes` (T1, Col, ncol, ONTO, BD, C)

**Si** ok **Alors**

`Insérer_Requête`(ReqInit, LReq, T1, C, ncol)

**fin\_si**

**fin\_pour**

**fin\_si**

**Fin**

##### 4.3. L'algorithme proposé

Notre approche automatique d'expansion de requêtes est décrite par un algorithme nommé `Reformulation`. Ce dernier est élaboré de manière générique (indépendamment du contenu de la base de données) afin d'être réutilisable pour une base de données relationnelle quelconque. Il prend en entrées une base de données relationnelle (BD), une ontologie de domaine (ONTO) et une requête initiale (ReqInit), et calcule un ensemble LReq\_Ref de requêtes résultantes de l'expansion de la requête initiale.

Nous commençons par appliquer la procédure `decomposer_Req()`, à la requête initiale ReqInit, afin d'extraire les colonnes relatives à chaque table dans

cette requête. Le résultat est un nuplet:

(ncol[], Col[[]], nTab, Tab[]) tel que :

nTab est le nombre de termes (tables) de la clause FROM de la requête initiale, Tab est un tableau contenant les tables décrites dans la requête ReqInit et Col[[]] est un tableau de dimension 2 tel que chaque Col[i] (i=1..nTab) est un tableau de taille ncol[i] contenant les champs de ReqInit qui représentent des colonnes relatives à Tab[i].

Par exemple, pour la requête ReqInit suivante :

```
SELECT enseignant.nom, enseignant.discipline,
departement.nom FROM enseignant, departement
WHERE enseignant.numerodep =
departement.numerodep
```

La procédure décomposer\_Req() génère les valeurs suivantes pour nTab, Tab, ncol et Col :

nTab=2 (deux tables dans la clause FROM)

Tab :

Enseignant	Département
------------	-------------

ncol[1] = 3, ncol[2]=2

Col :

Nom	Discipline	Numerodep
Nom	numerodep	

Pour chaque terme Tab[i], nous appliquons la procédure Reformulation\_monotable() qui génère un ensemble LReq[i] résultant de l'expansion de la table Tab[i] ainsi que ces différentes colonnes figurants dans la requête initiale. Si nous arrivons à générer tous les ensembles LReq[i] (i= 1..nTab), nous pouvons reformuler la requête initiale en remplaçant chaque couple (Tab[i], col[i]) par leur équivalent dans LReq[i].

Soit alors l'algorithme Reformulation suivant :

**Algorithme** Reformulation

**Entrées :**

ReqInit : la requête initiale

BD : une base de données relationnelle

ONTO : une ontologie de domaine

**Sortie :**

LReq\_Ref : liste de requêtes résultantes de la reformulation de ReqInit

**Debut**

decomposer\_Req ( ReqInit, ncol, Col, nTab, Tab)

i=1

Echec= faux

**Tant que** i<=nTab **et** echec= faux **faire**

Reformulation\_monotable(ncol[i],Col[i],Tab[i], BD, ONTO, LReq[i])

**Si** LReq[i] = nil **alors** Echec = vrai

**Sinon** i :=i+1

**Finsi**

**Fin faire**

**Si** i>nTab **alors**

reconstruire-requête(LReq, ReqInit, nTab, LReq\_Ref)

**Finsi**

**Fin**

## 5. Application

Soit une partie du modèle logique de données relationnel relatif à une base de données BD qui gère les informations d'une école :

Prof(**identifiant**, nom, prénom, spécialité, #Numerodep, .....

Enseignant\_permnt(**matricule**, nom, grade, #Numerodep, ..)

Departement (**Numerodep**, Nomdep, ....)

Soit l'ontologie ONTO couvrant le domaine de cette base de données. Dans cette ontologie, le concept C1 qui définit les enseignants est désigné par les termes contenus dans l'ensemble suivant : {enseignant, professeur, formateur, Prof}. Ce concept est spécialisé en deux concepts C2 et C3, désignés respectivement par les ensembles des termes: {Enseignant\_permnt, permanent} et {enseignant\_vacataire, vacataire}.

Soit alors :

designe (C1, 'Prof') ←

designe (C1, 'professeur') ←

designe (C1, 'enseignant') ←

designe (C1, 'formateur') ←

designe (C2, 'permanent') ←

designe (C2, 'enseignant\_permnt') ←

designe(C3, 'vacataire') ←

designe (C3, 'Enseignant\_vacataire') ←

designe(C4, 'specialité') ←

designe(C5, 'identifiant') ←

C-Spec(C1, C2) ←

C-Spec(C1, C3) ←

synonyme ('spécialité', 'discipline') ←

synonyme ('identifiant', 'matricule') ←

Soient les requêtes Ri, i= {1...3} suivantes :

**R1:** SELECT \* FROM enseignant

WHERE identifiant<100 and spécialité= 'info'

**R2:** SELECT nom, discipline FROM Prof

**R3:** SELECT enseignant.nom, departement.nom, enseignant.discipline FROM enseignant, departement Where enseignant.numerodep = departement.Numerodep

Ces requêtes, telles qu'elles sont formulées par l'utilisateur, n'ont pas de réponses et leurs exécutions génèrent des erreurs. En appliquant l'algorithme Reformulation, nous avons pu les corriger, par expansion, afin de satisfaire l'utilisateur qui, initialement, a mal formulé son besoin.

- R1: Dans la requête R1, le terme 'enseignant' ne

référence pas une table de la base de données. Il faut donc rechercher l'existence d'une table de la base de données qui est reliée sémantiquement à 'enseignant' par la relation de synonymie ou de spécialisation. La procédure `trouver_table_correspondante()` génère, à partir de l'ensemble des synonymes de 'enseignant', le terme 'Prof' représentant une table de la base de données. En plus, cette table ('prof') comprend les attributs 'identifiant' et 'spécialité'.

La nouvelle requête est alors:

```
SELECT * FROM Prof WHERE identifiant <100 and
spécialité= 'informatique'
```

- R2: Bien que la base de données comporte une table nommée 'Prof', cette dernière n'admet pas un champ nommé 'discipline'. Par contre, elle possède un attribut 'spécialité' synonyme de 'discipline'. Dans ce cas, l'application de l'algorithme Reformulation génère une nouvelle requête à partir de R2, en remplaçant le champ 'discipline' par 'spécialité'. La requête générée par notre approche est : `SELECT nom, spécialité FROM Prof`

- R3: Cette requête comprend deux termes dans la clause FROM qui sont 'enseignant' et 'département'. Ce dernier référence une table dans la base de données. En plus, la table 'département' possède les deux attributs nom et numerodep.

L'expansion de la table appliquée au terme 'enseignant' permet de le remplacer, à partir de l'ontologie ONTO, par son synonyme 'Prof' dans la base de données BD. La table 'Prof' comprend les champs nommés 'nom' et 'numerodep' et un champ 'spécialité' qui est synonyme de 'discipline'. La requête générée par l'expansion de R3 consiste alors à remplacer, dans cette requête, les termes 'enseignant' par 'Prof' et 'discipline' par 'spécialité'. La nouvelle requête est: `SELECT Prof.nom, département.nom, Prof.spécialité FROM Prof, departement WHERE Prof.numerodep = departement.numerodep`

## 6. Conclusion

Les systèmes classiques de gestion de base de données présentent des limites relatives à leurs insuffisances pour gérer une masse volumineuse de données ainsi que leurs pauvretés sémantiques.

Les recherches devront donc permettre l'évolution des bases de données actuelles vers des bases de données de plus en plus sémantiques. Pour cela, il est important de proposer des solutions compatibles avec l'existant. Dans ce cadre, nous avons défini une approche automatique d'expansion de requêtes SQL en exploitant une ontologie couvrant le domaine de la base de données considérée.

Notre approche est décrite par un algorithme que nous avons élaboré de manière générique afin d'être appliqué sur une base de données classique quelconque.

L'originalité de notre travail est que la solution proposée est compatible avec l'existant et ne

nécessite pas de modifier SQL. Elle permet de relaxer la requête en exploitant une ontologie afin d'éviter les réponses vides et de fournir la réponse la plus proche possible des souhaits de l'utilisateur.

Nous avons exploité les relations sémantiques de synonymie et de spécialisation présentes dans l'ontologie de domaine que nous avons construit afin de couvrir tous les termes présents dans la base de données. Une extension de notre contribution, prenant en considération la relation sémantique de composition, est actuellement en cours de réalisation.

La détection des causes d'échec est aussi en cours de réalisation. En effet, lorsque notre approche échoue, nous pouvons détecter les termes introuvables qui représentent les raisons d'échec. Dans ce cas, l'ontologie peut être mise à jour en rajoutant chacun de ces termes comme un nouveau concept ou comme étant un nouveau terme à rajouter dans la liste des termes décrivant un concept présent dans l'ontologie.

## REFERENCES

- [ANG 90] Angelaccio M., Catarci T. et Santucci G., QBD\* : A Graphical Query Language with Recursion. *IEEE Transactions on Software Engineering*, 16, 1150-1136, 1990.
- [AUD 91] Auddino A., Dupont Y., Fontana E., Spaccapietra S. et Tari Z., SUPER : A comprehensive Approach to database visual interfaces. In *IFIP 2nd Working Conference On Visual Database Systems*, pages 359,374, 1991.
- [BAC 05] Bachimont B., Baneyx A., Malaisé V., Charlet J. et Zweigenbaum P., Synergie entre analyse distributionnelle et patrons lexico-syntaxiques pour la construction d'ontologies différentielles, *Terminologie et Intelligence Artificielle*, Rouen, France, Avril 2005.
- [BAD 96] Badre A.N., Catarci T., Massari A. et Santucci G., Comparative ease of use of a diagrammatic vs. An iconic query language. In *Interfaces to Databases. Electronic Series Workshop in Computing*, pp 1-14, Springer, 1996.
- [BAR 04] Barrasa J., Corcho Ó. et Gómez-Pérez A., R2O, an Extensible and Semantically Based Database-to-ontology Mapping Language. *Ontology Engineering Group, SWDB*, 2004.
- [BAZ 03a] Baziz M., Aussenac-Gilles, N., et Boughanem, M., Exploitation des liens sémantiques pour l'expansion de requêtes dans un système de recherche d'information. *XXIème Congrès INFORSID*, Nancy, France, p.121-134, janvier 2003.
- [BAZ 03b] Baziz M., Boughanem M. et Nassr N., La recherche d'information multilingue: désambiguïsation et expansion de requêtes basées sur WordNet. Dans : *ISPS'2003 Sixt-h International Symposium on Programming and Systems*, p. 175-186, Algeria, Mai 2003.
- [BEN 03a] Benslimane D., Arara A., Yetongnon K., Gargouri F. et Ben Abdallah H., Two approaches for ontologies building : From-scratch and From existing data sources, *the 2003 International Conference on Information Systems and Engineering ISE 2003*,

- Montreal, Canada, 20-24 July 2003.
- [BEN 03b] Benslimane D., Christelle Vangenot, Catherine Roussey et Ahmed Arara, Multirepresentation in Ontologies, *Advances in Databases and Information Systems*, pages : 4-15. Septembre 2003.
- [BID 02] Bidaut A., Froidevaux C. et Safar B., Similarity between queries in a mediator. *In proc. of ECAI'02*, pages 235-239, 2002.
- [BIZ 03] Bizer C, D2R MAP – A Database to RDF Mapping Language, *The twelfth international World Wide Web Conference, WWW2003*, Budapest, Hungary, 2003.
- [CAT 95] Catarci T. et Santucci G., Diagrammatic vs Textual Query language : A comparative experiem. *In Proc. of the IFIP W.G.2.6 Working Conference on Visual Databases*, Lausanne, pp. 57-74, 1995.
- [CUL 07] Cullot N., Ghawi R. et Yétongnon K., DB2OWL: A Tool for Automatic Database-to-Ontology Mapping. *In Proceedings of the 15th Italian Symposium on Advanced Database Systems*, Italy, pp. 491-494, 2007.
- [DAH 03] Dahainsala H., Conception d'une architecture de Base de Données à base Ontologique, Mémoire de DEA - Stage effectué au LISI/ENSMA, 2003.
- [EPS 91] Epstein R.G., The TableTalk Query Language, *Journal of Visual Language and Computing*, 2, 2, p115-141, 1991.
- [FAY 08] Fayeche I. et Ounalli H., Une approche ontologique d'expansion de requêtes SQL, *In Proceeding of 10TH Magrebian Conference on Software Engineeering and Artificial Intelligence MCSEAI08*, Page 435- 439, Oran Algeria, 28–30 Avril 2008.
- [GRO 88] Groette I.P. et Nilson E.G., SICON : An icon presentation module for an E.R. database. *In Proceeding of the 7th Int. Conf. On Entity-Relationship Approach*, 1988.
- [GRU 93] Gruber T.R., A translation approach to portable ontology specification, knowl. Acquis, vol. 5 num2, p199-220, *Academic Press Ltd*, 1993.
- [GUA 95] Guarino N. et Giarretta P., Ontologies and knowledge bases, towards a terminological clarification, Towards very large knowledge bases: knowledge building and knowledge sharing, *IOS Presse*, pages 25-32, 1995.
- [HER 05] Hernandez N., Ontologies pour l'aide à l'exploration d'une collection de documents, *In Revue des Sciences et Technologies de l'Information (RSTI), Série Ingénierie des systèmes d'information*, 10 (1), pp 11-34, 2005.
- [JEA 04] Jean S., Langage d'exploitation de base de données ontologiques, Mémoire de DEA - Stage effectué au LISI/ENSMA, 2004.
- [KHA 00] Khan Latifur R., thèse de doctorat. Ontology-based Information Selection. Faculty of the graduate school university of southern california, 2000.
- [LEN 95] Lenat D. B., Cyc : A Large-scale investment in Knowledge Infrastructure, *Communications of the ACM*, pp. 33-38, vol. 38, no. 11, 1995.
- [MES 06] Messai N., Devignes M., Napoli A. et Smail-Tabbone M., Treillis de concepts et ontologies pour interroger l'annuaire de sources de données biologiques BioRegistry (version étendue). *Revue ISI, Ingénierie des Systèmes d'Information: Systèmes d'information spécialisés 11(1)*, pages 39–60, 2006.
- [MHI 05] Mhiri M., Mtibaa A. et Gargouri F., Towards an approach for building information systems'ontologies, *1st workshop Formal Ontologies Meet Industry*, 9-10, June, Verona, Italy, 2005.
- [MIL 95] Miller G., WordNet: A Lexical Database for English, *Communications of the ACM*,” vol. 38, no. 11, 1995.
- [SAF 04] Safar B., Kefi H. et Reynaud C., OntoRefiner, a user query refinement interface usable for Semantic Web Portals, Application of Semantic Web Technologies to Web Communities Workshop, August 23rd, *16th European Conference on Artificial Intelligence*, Valencia, Spain, August 22-27, 2004.
- [SOU 03] Soualmia, L. F. et Darmoni, S. J., Projection de requêtes pour une recherche d'information intelligente sur le web. *In RJCIA*, pages 59-72, 2003.
- [SWA 96] Swartout B., Patil R., Knight K., and Ross T., Toward Distributed Use of Large Scale Ontologies, *in Proc. of the Tenth Workshop on Knowledge Acquisition for Knowledge Based Systems*, Banff, Canada, 1996.
- [TON 89] Tonomura Y. and Abe S., Content Oriented Visual Interface Using Video Icons of Visual Databases Systems. *Proc. Of the Int. Workshop on Visual Languages*, Roma, Italia, 1989.
- [TSU 90] Tsuda K., Yoshitaka A., Hirakawa M., Tanaka M. et Ichikawa I., Ionic Browser : An Iconic Retrieval System for Object-Oriented Databases. *Journal of Visual Language and Computing*, 1, 1, pages 59-76, 1990.
- [VOO 94] Voorhees E. M., Query expansion using lexical semantic relations. In SIGIR '94: *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 61-69, 1994.
- [YEN 93] Yen M.Y. et Scamell R.W., A Human Factors Experimental Comparaison of SQL and SBE. *IEEE Transactions on Software Engineering*, 19, 4, page 390-402, 1993.
- [ZLO 77] Zloof M.M., Query-By-Example : A Database Language. *IBM Syst.Journal*, 16, 4, pages 324-343, 1977.