Expansion de requêtes SQL par une ontologie de domaine

Ines Fayech*, Habib Ounalli**
Faculté des sciences de Tunis - Université El Manar Tunis 1060 Tunis
*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.

1 Introduction

Les systèmes classiques de gestion de base de données (SGBD) présentent une limite relative à leur pauvreté sémantique. Les recherches devront donc permettre l'évolution des bases de données actuelles vers des bases de données de plus en plus sémantiques.

Avec l'avènement des ontologies, un progrès important a pu être réalisé grâce à la représentation explicite de la signification des données (Gruber, 1993). Dans ce cadre, nous nous intéressons à leur exploitation dans une approche automatique d'expansion de requêtes SQL afin d'éviter les réponses vides et de fournir la réponse la plus proche possible des souhaits de l'utilisateur. L'approche proposée permet de résoudre certains conflits de données dans la représentation de la requête initiale. Un conflit est une situation dans laquelle deux éléments ont des noms sémantiquement liés, mais présentent des différences dans leurs structures ou dans leurs comportements. Particulièrement, nous étudions les conflits de nom liés à l'ambiguïté des termes utilisés dans la requête et les conflits sémantiques qui se manifestent quand l'intension de l'utilisateur est modélisée à un niveau d'abstraction différent de celui utilisé par le schéma de la base de données. 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 les liens sémantiques de synonymie, de méronymie et de spécialisation décrits par une ontologie de domaine de la base de données.

Le reste du papier est organisé comme suit : Nous commençons par définir, en section 2, l'expansion de requêtes SQL. La section 3 est consacrée à l'ontologie de domaine utilisée. Dans les deux sections suivantes, nous détaillons l'approche ainsi que l'algorithme proposés. Des exemples d'application de notre proposition seront cités dans la section 6. Enfin, nous terminons avec une conclusion et quelques directions de recherches pour les travaux futurs.

2 L'expansion de requêtes SQL

L'expansion d'une requête SQL consiste à la reformuler en remplaçant certains termes dans cette requête par d'autres, dans l'ontologie, qui les relient sémantiquement.

En considérant les liens sémantiques de synonymie, de spécialisation et de méronymie, nous distinguons les techniques suivantes d'expansion de requêtes :

- 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 méronymie exploitée dans différents travaux (Buscaldi *et al.,* 2005), consiste à remplacer certains concepts de la requête initiale par leurs méronymes.

- 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 inclu dans l'ensemble des solutions pertinentes pour l'utilisateur. Ce type d'expansion a été exploité dans le cas d'une réponse vide à une requête (Messai *et al.*, 2006 ; Bidaut *et al.*, 2002) ou lorsque l'utilisateur obtient trop de réponses à sa requête (Safar *et al*., 2004).

3 L'ontologie de domaine utilisée

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 (Swartout *et al.,* 1996) et WordNet (Miller, 1995). Une telle ontologie présente une limite du fait qu'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.

Afin de pallier à cette lacune, nous allons exploiter une ontologie de domaine décrivant le vocabulaire relatif à un domaine générique (Guarino *et al*., 1995). Le choix d'une telle ontologie pose le problème de sa construction. D'après (Benslimane *et al.,* 2003), la construction d'une ontologie peut débuter à partir de zéro ou à partir des bases de données déjà existantes.

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, peut avoir un ou plusieurs termes qui le désignent. Il peut aussi être lié à d'autres concepts par la relation de méronymie. Nous allons par la suite enrichir cette ontologie minimale en rajoutant d'autres concepts et termes afin de couvrir les termes contenus dans la base de données. Ces termes peuvent ne pas correspondre à des mots en langage naturel. 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) si elle est non vide. Cette dernière est utile pour décider de l'insertion de chaque terme T dans L-DB comme étant un nouveau concept dans l'ontologie ou comme une mise à jour de la liste des termes d'un concept existant.

Nous représentons l'ontologie de domaine par une base de connaissance décrite sous la forme de clauses de Horn. 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 et 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 : synonyme(T1, T1) ←
synonyme(T1,T2)←terme(T1),terme(T2),concept(C),designe(C,T1),designe(C,T2)
Ce prédicat vérifie les propriétés suivantes : la symétrie (synonyme(T1, T2) ← synonyme (T2, T1)) et la transitivité (synonyme(T1, T2) ← synonyme(T1, T3), synonyme(T3, T2))

- La relation sémantique de spécialisation entre deux concepts C1 et C2 (représentée par le prédicat C-Spec(C1, C2)), exprime que le concept C1 spécialise le concept C2. Ce prédicat vérifie la propriété de transitivité : C-Spec(C1, C2)← C-Spec(C1, C3), 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: T-Spec(T1, T2)← terme(T1), terme(T2), concept (C1), concept (C2) , designe (C1, T1), designe (C2, T2), C-Spec(C1, C2)

- La relation sémantique de méronymie entre un concept C et des concepts C1, C2,…, Cn (avec n un entier et $n \geq 1$) exprime que le concept C est composé de C1, C2,…, Cn. Nous utilisons un prédicat C-Mer() pour vérifier cette relation.

- Un terme T est relié à des termes T1, T2,…..,Tn (avec n un entier et $n \geq 1$) par une relation de méronymie, si les concepts qui les désignent les sont aussi. Les termes T1, T2,…, Tn, sont les méronymes du terme T.

4 L'approche proposée

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, de méronymie et de spécialisation présents dans l'ontologie considérée. L'expansion peut s'appliquer sur les tables et/ou sur les colonnes. Nous allons décrire, ci-après, l'expansion des colonnes ainsi que celle des tables.

- L'expansion de l'ensemble Col des colonnes d'une table T (décrite par la procédure expansion_colonnes()) se base sur l'application des relations sémantiques de synonymie et de méronymie. Il s'agit de remplacer chaque élément C de Col, qui ne référence aucun attribut de la table T, par des termes qui les relient sémantiquement. Ces termes désignent des attributs de la table T dans la base de données considérée. Si la colonne C est dans la partie SELECT de la requête et s'il n'existe aucun synonyme de C désignant un attribut de T, nous appliquons la relation de méronymie afin de générer, si possible, un sous ensemble de méronymes de C correspondant (synonymes ou égales) à des attributs de la table T.

- L'enrichissement d'une table Tab consiste à vérifier pour le 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. Dans ce cas, nous commençons par appliquer une expansion de la table Tab par synonymie (la procédure trouver_table_correspdante()). Si on génère un nouveau terme Tab1, synonyme ou égal à Tab et référençant une table de la base de données, alors on remplace Tab par Tab1. Sinon, on applique une expansion de la table par spécialisation (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 ses 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_correspdante (Tab, BD, ONTO, Tab1)
Si Tab1 existe alors Tab = Tab1
ok = expansion_colonnes(Tab, Col, ncol, ONTO, BD, C)
Si ok alors Insérer_Requête(ReqInit, LReq, Tab, C, ncol) fin_si
{/* 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/*} 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**

Notons que le remplacement d'une table ou d'une colonne par une autre qui lui est reliée sémantiquement, n'est possible que si l'utilisateur dispose du droit d'accès à ce nouveau terme. En effet, dans une base de données relationnelle, il est possible d'interdire l'accès à une donnée ou de le limiter pour certains utilisateurs.

5 L'algorithme proposé

Notre approche automatique d'expansion de requêtes est décrite par un algorithme (nommé Reformulation) élaboré de manière générique 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 décomposer la requête initiale afin d'extraire les colonnes relatives à chaque table dans cette requête (la procédure decomposer_Req()).

Le résultat est un nuplet : (ncol, Col, nTab, Tab) tel que :nTab est le nombre de termes (tables) dans 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].

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 ses 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 son correspondant dans LReq[i]. Soit alors l'algorithme suivant :

Algorithme Reformulation
Entrées : ReqInit : requête initiale, BD : base de données et ONTO : ontologie
Sortie : LReq_Ref : liste de requêtes résultantes de la reformulation de ReqInit
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 finfaire
Si i>nTab alors reconstruire-requête(LReq, ReqInit, nTab, LReq_Ref) finsi
Fin

6 Application

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

Enseignant_contractuel(identifiant, nom, prénom, spécialité, #Numerodep, ville ……)
Enseignant_permnt (matricule, nom, grade, #Numerodep, ..)
Departement (Numerodep, nom, ….)
Soit une partie de l'ontologie ONTO couvrant le domaine de cette base de données :

designe (C1, 'enseignant') ← designe (C2, 'enseignant_permnt') ←
designe (C2, 'permanent') ← designe(C3, 'contractuel') ←
designe (C3, 'Enseignant_contractuel') ← designe(C4, 'specialité') ←
designe(C5, 'identifiant') ← designe(C6, 'adresse') ←
designe(C7, 'ville') ← designe(C8, 'rue') ←
designe(C9, 'code postale') ← C-Mer(C6, C7, C8, C9) ←
C-Spec(C1, C2) ← C-Spec(C1, C3) ←
synonyme ('spécialité', 'discipline') ← synonyme ('identifiant', 'matricule') ←

Soient les deux requêtes R1 et R2 suivante :

- R1: SELECT nom, adresse FROM contractuel WHERE discipline = 'info'

Cette requête n'a pas de réponse à cause de conflits de noms. Dans cette requête, le terme 'contractuel' ne référence pas une table de la base de données. Il sera remplacé par son synonyme, le terme 'enseignant_contractuel', représentant une table de la base de données. Cette table a un attribut 'identifiant' et un attribut 'spécialité' synonyme du terme 'discipline'.Cependant, elle ne comprend pas un attribut correspondant à l'adresse. L'expansion du terme 'adresse', par méronymie, génère le terme 'ville', un attribut de la table considérée dans la base de données. Dans ce cas, l'algorithme proposé génère la nouvelle requête suivante : SELECT nom, ville FROM enseignant_contractuel WHERE spécialité = 'info'

- R2 : SELECT enseignant.nom, departement.nom FROM enseignant, departement WHERE enseignant.numerodep=departement. numerodep

Cette requête a une réponse vide dûe à un conflit sémantique entre le schéma de la base de donnée et le celui imaginé par l'utilisateur. En effet, dans la base de données, les enseignants sont spécialisés en permanents et contractuels. L'approche proposée génère les deux requêtes suivantes dont l'union est une requête satisfaisant l'utilisateur : R21 (SELECT enseignant_contractuelt.nom, departement.nom FROM enseignant_contractuel, departement WHERE enseignant_contractuel.numerodep=departement.numerodep) et R22 (SELECT enseignant_permnt.nom, departement.nom FROM enseignant_permnt, departement WHERE enseignant_permnt.numerodep=departement. numerodep)

7 Conclusion

Nous avons défini une approche 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'approche proposée permet de résoudre les conflits de nom et les conflits sémantiques. Actuellement, nous travaillons sur l'extension de notre approche afin de résoudre certains conflits structurels. Nous avons un conflit structurel quand la même information est modélisée de façon différente dans la base de données et dans la requête. Ce type de conflits a été rarement étudié dans le domaine des bases données et surtout dans les travaux d'intégration de schéma de base de données.

Références

BENSLIMANE D. & ARARA A. & YETONGNON K. & GARGOURI F. & BEN ABDALLAH H. (2003). Two approaches for ontologies building : From-scratch and From existing data sources. *The 2003 International Conference on Information Systems and Engineering ISE*

BIDAUT A. & FROIDEVAUX C. & SAFAR B. (2002). Similarity between queries in a mediator. *In proc. of ECAI'02*, pages 235-239, 2002.

BUSCALDI D. & ROSSO P. & ARNAL ES. (2005). A WordNet-based Query Expansion method for Geographical Information Retrieval. *In CLEF 05*.

GRUBER T.R. (1993). A translation approach to portable ontology specification, knowl. Acquis, vol. 5 num2, p199-220, *Academic Press Ltd.*

GUARINO N. & GIARETTA P. (1995). Ontologies and knowledge bases, towards a terminological clarification, Towards very large knowledge bases: knowledge building and knowledge sharing. , *IOS Presse*, pages 25-32.

MESSAI N. & DEVIGNES M. & NAPOLI A. & SMAÏL-TABBONE M. (2006). Treillis de concepts et ontologies pour interroger l'annuaire de sources de données biologiques BioRegistry (version étendue). *Revue ISI,* pages 39–60.

MILLER G. (1995). WordNet: A Lexical Database for English. *Communications of the ACM,* vol. 38, no. 11.

SAFAR B. & KEFI H. & REYNAUD C. (2004). OntoRefiner, a user query refinement interface usable for Semantic Web Portals, Application of Semantic Web Technologies to Web Communities Workshop. *16th European Conference on Artificial Intelligence*, Valencia.

SWARTOUT B. & PATIL R. & KNIGHT K. & ROSS T. (1996). Toward Distributed Use of Large Scale Ontologies. *In Proc. of the Tenth Workshop on Knowledge Acquisition for Knowledge Based Systems,* Banff, Canada.

Summary

In this paper, we are interested in the use of domain ontologies as a semantic enrichment for traditional databases. Our first aim is to help the user in his search when his initial query doesn't return any result. So, we propose an ontological approach for SQL query expansion based on different refinement types.