Des documents virtuels pour lire les bases de données

Gilles Falquet[+], Luka Nerima[+], Jacques Guyot[+], Christine Vanoirbeek[++], Yassine A. Rekik[++]

[+] Centre Universitaire d'Informatique, Université de Genève
24, rue Général-Dufour – CH 1211 Genève 4 – Switzerland. Tel +41 22 705 77 72
{falquet, nerima, guyot}@cui.unige.ch

[++] MEDIA Research Group
LITH-DI-EPFL – IN Ecublens – CH 1015 Lausanne - Switzerland. Tel +41 21 693 6782
{Christine.Vanoirbeek, Yassine.Rekik}@epfl.ch

1. INTRODUCTION

Il est bien connu que l'accès aux informations stockées dans une base de données est difficilement praticables, pour un non spécialiste, à l'aide d'un langage d'interrogation de type SQL. D'autre part, l'accès à travers des formulaires de requête nécessite un important travail de développement et aboutit à des solutions rigides qui ne satisfont que des besoins biens précis et définis d'avance. La voie que nous explorons consiste à remplacer la notion d'interrogation des bases de données par la notion de lecture et de navigation dans un hyperdocument virtuel qui représente ces données.

Le développement du World Wide Web a poussé les chercheurs et les éditeurs de logiciels à aller dans ce sens. Pour preuve, tous les éditeurs de base de données proposent désormais des outils pour générer automatiquement des pages HTML à partir du contenu de la base de données. L'approche est généralement procédurale, c'est-à-dire qu'elle nécessite l'écriture de programmes s'appuyant sur des procédures prédéfinies qui génèrent des pages HTML. D'autres systèmes génèrent automatiquement ces pages à partir du schéma de la base de données, selon une méthode fixe.

La recherche dans ce domaine met en avant une autre approche : l'approche déclarative. Elle repose simultanément sur le modèle de données et sur le modèle de l'hypertexte (en l'occurrence le modèle du Web) et propose un langage d'interrogation pour passer de l'un à l'autre [Atzeni 97] [Sindoni 98] [Siméon 98]. Construire un site Web en utilisant un langage déclaratif pour décrire sa structures représente plusieurs avantages. Premièrement, cela permet de construire plusieurs versions d'hypertexte ou vues sur les mêmes données. Deuxièmement, la prise en compte de l'évolution de la structure de données et de celle de l'hypertexte sera facilitée. Nos travaux sont représentatifs de cette approche.

La notion de vue est particulièrement bien adaptée à la personnalisation des hyperdocuments car chaque vue représentera le point de vue d'un utilisateur ou d'un groupe d'utilisateurs. Cette approche n'est cependant praticable que si l'on dispose d'un langage de spécification de vues déclaratif pour qu'il ne soit pas nécessaire d'écrire un programme de génération pour chaque vue.

Ce papier décrit notre expérience de spécification des hypertextes sur les bases de données. Nous présentons brièvement nos modèles et le langage de spécification de l'hypertexte à partir du contenu de la base de données. Nous décrivons aussi le mécanisme de mise à jour de du contenu de la base de données en agissant sur l'hypertexte. Les problèmes de conception sont également abordés.

2. LANGAGE DE DÉFINITION DE VUES HYPERDOCUMENTS

Afin de pouvoir spécifier formellement le langage de définition de vues, nous avons choisi un modèle de base de données et un modèle d'hyperdocument abstraits simples.

Le modèle de base de données est un modèle à objets qui est un sous-modèle du modèle de base de données O2. Chaque objet possède des attributs qui peuvent être soit atomiques (chaînes de caractères, nombres, booléens, etc.), soit des références à d'autres objets. Les attributs peuvent être monovalués (scalaires) ou multivalués (ensembles ou listes). Tout objet est instance d'une classe qui définit les noms et types de ses attributs. Une base de données est formée d'un ensemble de collections d'objets provenant de la même classe. On notera que ce modèle de données contient le modèle relationnel.

Une spécification de vue hyperdocument se compose d'un ensemble de schémas de noeuds. Un schéma de noeud se présente sous la forme suivante:

node <node-name> [<parameters>] is
    <element> ...
from <collection>
selected by <expression>
ordered by <expression>

Les noeuds de la vues hyperdocument sont des instances des schémas de noeuds. L'instantiation d'un noeud consiste à sélectionner les objets de la collection de données qui satisfont l'expression de sélection; ordonner ces objets d'après l'expression d'ordonnancement; pour chacun de ces objets créer un item du noeud, lui-même composé en évaluant successivement tous les éléments de la spécification.

Chaque élément peut être une constante littérale (chaîne de caractères, nombre entier, etc...) ou un nom d'attribut. Les éléments peuvent contenir des fonctions de marquage (*paragraph(), heading1(), emphasis(),* etc...) qui produiront des tags adéquats pour le langage de spécification d'hyperdocument cible (XML, HTML, etc.).

Exemple. Si le schéma *emp_of_dept* est défini comme:

node emp_of_dept[d: int] is
" No: ", bold(no), " => " , bold(name),
break(),
" Hire date: ", hire_date
from EMP selected by deptno = d ordered by emp_name

l'instance *emp_of_dept[10]* contiendra un item par objet de la collection *EMP* ayant *deptno = 10*. La figure ci-dessous montre l'apparence d'une telle instance produite pour le Web.

Liens de référence

Chaque champ d'un item peut être l'ancre de départ d'un lien de référence (href). La référence à un noeud se fait en indiquant le nom de son schéma et la valeur des paramètres.

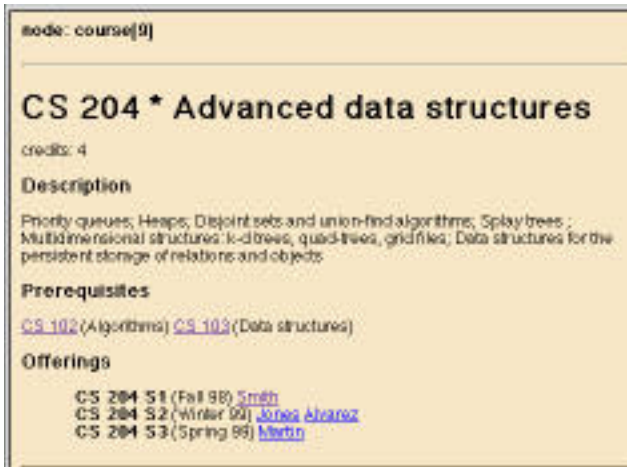
L'exemple suivant montre un schéma de noeud possédant une référence vers un noeud de type *emp_of_dept*.

node dept_in[loc: String] is
no, ": ", bold (name), " ",
href emp_of_dept [no] " Employees: "
...
from DEPT selected by location = loc
ordered by no

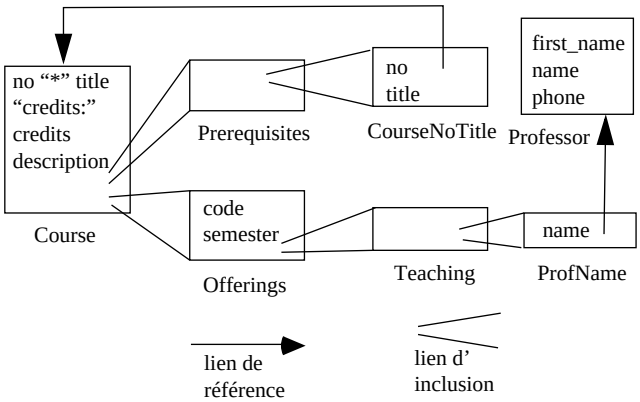
L'item représentant un département de numéro nd aura une ancre " Employees: " qui sera le point de départ d'un lien vers le noeud *emp_of_dept[nd]*.

Liens d'inclusion

Un lien d'inclusion entre deux noeuds détermine une relation composé-composant entre ces noeuds. Le noeud de cible doit être considéré comme un sous-noeud du noeud source du lien. Ce fait devrait normalement être pris en considération par le système d'interface d'hypertexte qui devrait présenter des liens d'inclusion d'une manière particulière (généralement en incluant le contenu de sous-noeud dans la présentation de noeud). Les liens d'inclusion servent en particulier à créer des noeuds hiérarchiques à plusieurs niveaux pour représenter les entités complexes, comme dans l'exemple ci-dessous.



Ce noeud complexe est une instance du schéma de vue hyperdocument suivant:



3. CONCEPTION DES VUES HYPERDOCUMENT

Il faut tout d'abord remarquer que la conception des bases de données et celle des hypertextes n'ont pas les mêmes objectifs. Donc, si l'on peut s'appuyer le schéma de base de données au niveau sémantique, il s'agira ensuite de

créer une structure hypertextuelle efficace du point de vue de la lecture et de la navigation.
Les expériences que nous avons menées sur des cas réels nous ont conduit à développer une méthode de conception des vues hypertextes par raffinements successifs. Cette méthode procède en deux phases: 1) on définit une première structure d'hypertexte calquée sur le schéma de la base de données; 2) on raffine cette structure en appliquant diverses opérations sur les spécifications de noeuds et de liens.

Structure initiale

On obtient la structure initiale en définissant un schéma de noeud pour chaque collection de la base de données. Le contenu du noeud est formé des attributs atomiques de la classe. Pour chaque attribut qui fait référence à un objet d'une autre classe on crée un lien de référence vers le schéma de noeud correspondant. Un attribut multivalué sera traduit par un lien vers un noeud « menu » qui contiendra les ancres des liens vers les objets individuels.
Exemple. Si la classe *Vehicule* possède les attributs atomiques no et marque et un attribut propriétaire qui fait référence à un objet de la classe Personne, on aura un noeud

node NVehicule[me: Vehicule] is
no, marque, href NPersonne[proprietaire] "propriétaire"
from Vehicule selected by self = me
(le pseudo attribut *self* désigne l'identité de l'objet)

Chaque instance d'un tel noeud représentera un objet de la base de donnée Cette première structure représente donc fidèlement le contenu de la base de données, c'est-à-dire que le graphe formé de toutes les instances de noeuds et liens possibles est isomorphe au graphe des objets liés par les attributs de référence.

Opérations de raffinement

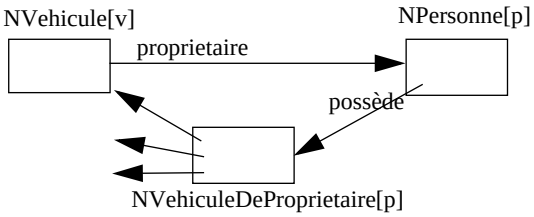
Les opérations de raffinement vont permettre d'améliorer la navigabilité (ou la lisibilité) de la structure initiale. Nous énumérons ci-dessous quelques unes de ces opérations.

Création de liens inverse

Si un noeud *N* possède une référence (*href M[r]*), il s'agit d'ajouter un lien de *M* à *N* qui correspond au parcours en sens inverse de ce lien (on se souvient que les liens sont monodirectionnels). Pour réaliser cette opération on définit un schéma de noeud intermédiaire. Sur l'exemple précédent, pour créer le lien inverse de *proprietaire* on définira un noeud

node NVehiculeDeProprietaire[p: Personne] is
href NVehicule[self]
from Vehicule selected by proprietaire = p

Ce noeud contiendra donc une liste de références à tous les noeuds *NVehicule* qui font référence à la même personne p en tant que propriétaire. Finalement on ajoutera un lien href *NVehiculeDeProprietaire[self]* dans le noeud Personne. Le noeud intermédiaire que l'on a créé joue le rôle du menu de sélection dans les systèmes qui autorisent les liens à extrémités multiples. La figure ci-dessous illustre ce principe.

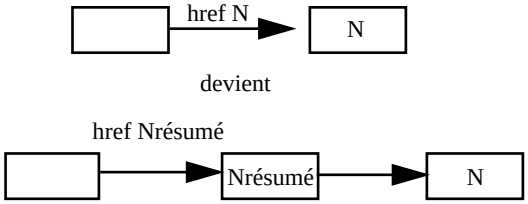


Inclusions

Cette opération consiste à changer la nature d'un lien de référence en lien d'inclusion. Elle permet, par exemple, de représenter des entités complexes sous forme d'un seul noeud comprenant des sous-noeuds. Cette opération est, par exemple, intéressante lorsque le lien possède une sémantique de type « partie-de » ou « composé-de ».

Dérivation

Lorsqu'un noeud représente un objet possédant de nombreux attributs, on peut créer par dérivation un schéma de noeud « résumé » en supprimant certains attributs de la définition initiale. Ce noeud résumé possédera un lien vers le noeud complet. Il faut également décider pour chaque lien qui aboutit au noeud initial s'il faut le « dévier » vers le noeud résumé. Cette opération est représentée schématiquement ci-dessous.



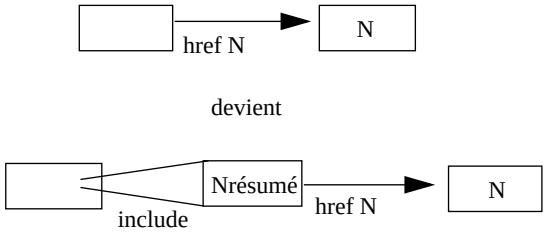
Elargissement

L'élargissement d'un noeud consiste à rendre plus faible sa condition de sélection, ce qui aura pour effet de montrer d'autres objets dans le noeud. Ceci permet de contextualiser une information en la présentant accompagnée d'autres informations considérées comme proches. Par exemple, on pourra élargir un noeud qui représente un livre (donné en paramètre) à tous les livres du même auteur ou à tous les livres placés sur la même étagère.

Prévisualisation

La prévisualisation permet de voir une partie du contenu d'un noeud référencé sans avoir à parcourir le lien. L'objectif étant d'éviter la navigation vers un noeud dont le contenu ne correspond pas à l'information cherchée.

Cette opération a pour effet de créer un noeud résumé (en général avec seulement quelques attributs), comme dans l'opération de dérivation, et de remplacer le lien de référence initial par un lien d'inclusion du noeud résumé.

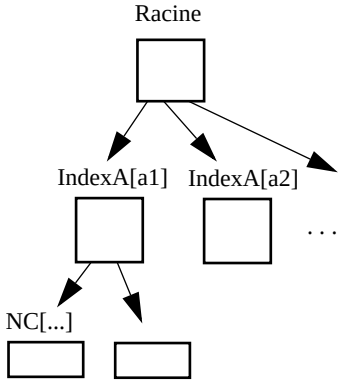


Création de structure d'index

Une structure d'index est un ensemble de noeuds qui permet, par sélections successives à partir d'un noeud racine, d'atteindre un noeud particulier. Un cas particulier simple est la création d'un index sur un attribut A. Il implique à la création de deux schémas de noeuds : 1) un noeud racine présentant toutes les valeurs possibles de A; 2) un schéma de noeud présentant une liste de tous les objets possédant une même valeur pour A.

node Racine[] is
href IndexA[A] A
from C selected by true

node IndexA[val: T] is
href NC[self] ...
from C
selected by A = val



On peut généraliser cette structure pour créer des index à plusieurs niveaux où chaque niveau correspond à un attribut différent. Le parcours depuis la racine revient, à chaque étape, à fixer une valeur d'un attribut et donc à restreindre le choix.

Création de parcours séquentiels

Cette opération crée des liens qui permettent de parcourir tous les noeuds instances d'un schéma dans un ordre prescrit (visite guidée).

4. MISES À JOUR

Si l'interrogation directe d'une base de donnée à l'aide d'un langage de type SQL est compliquée, la mise à jour l'est plus encore, en particulier lorsqu'il s'agit d'établir ou de supprimer des associations entre entités. Le design et le développement d'interfaces de mise à jour simples et intuitifs requiert également un travail considérable. Par contre la plupart des utilisateurs sont capables de mettre à jour un document en utilisant un logiciel de type traitement de texte. D'où l'idée d'utiliser les documents virtuels comme interface de mise à jour d'une base de données.

Les instances de noeuds hypertexte contiennent des éléments de nature atomique ou lien. L'utilisateur va agir sur ces éléments en utilisant les opérations que l'on trouve traditionnellement dans le paradigme de l'hypertexte: couper/coller/insérer/modifier un item et créer/supprimer un lien entre deux noeuds, ces opérations seront ensuite répercutées dans la base de donnée au moyen des opérations de mise à jour habituelles des bases de données. Ceci nécessite d'ajouter à la spécification du document virtuel la manière de traduire les actions sur le document en mises à jour de la base.

Insérer (coller) et supprimer un élément

Sur l'instance d'un noeud ces opérations correspondent respectivement à insérer (ou coller) et à couper un item. La mise à jour de la base suite à une opération d'insertion pose plusieurs problèmes. Ils sont la conséquence de trois causes:

1. L'insertion dans un document attribue un ordre à l'élément inséré (l'endroit où il est inséré dans le document). Dans la base de données, cette ordre sera perdu (les objets font partie de collections). Plusieurs solutions sont possibles: valider l'insertion uniquement si le nouvel item est à sa bonne place (c-à-d l'emplacement conforme à l'instruction de tri du noeud); gérer implicitement l'ordre des items dans le noeud; considérer que l'insertion d'un élément dans le noeud n'attribue pas d'emplacement particulier etc.

2. L'instance de noeud pouvant être une sélection de collection (le sous-ensemble d'une collection), il faut réévaluer la condition de sélection avant de mettre à jour effectivement la collection dans la base de données. Si la condition n'est pas satisfaite, il faut choisir si l'action à entreprendre est d'insérer ou non l'objet dans la base. Dans tous les cas, l'élément n'apparaîtra pas dans le noeud. Ce problème est d'ailleurs plus général et se posera avec tout document virtuel, il correspond au problème bien connu de la mise à jour des vues dans les bases de données [Mas 84].

3. Deux situations peuvent se présenter: (i) lorsque la condition de sélection porte uniquement sur la collection de base du noeud, il suffit d'insérer un objet ou mettre à jour un attribut dans cette collection, (ii) si la condition porte sur d'autres collections que la collection de base du noeud, il faut propager la mise à jour dans les autres collections.

Autres problèmes de la mise à jour:

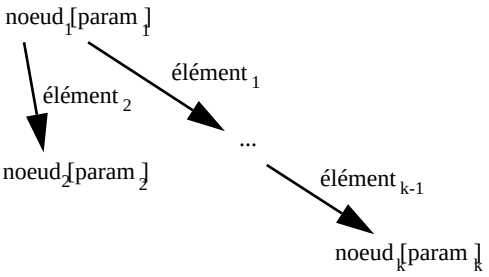
Comme les noeuds hypertexte sont des vues, des problèmes d'ambiguïté vont éventuellement surgir comme lorsqu'il s'agit de mettre à jour une vue de base de don-

nées relationnelle ou objet.

Un exemple bien connu: Considérons la base de données d'une Université; lorsque l'on supprime une personne d'un département faut-il faire le même genre d'opération de mise à jour sur la base que lorsque l'on supprime une personne de l'équipe de football du département ? Dans le premier cas, on voudra sans doute supprimer la personne de la base de données (ou la déplacer dans la collection anciens collaborateurs) alors que dans le deuxième cas on voudra simplement mettre l'attribut membre de l'équipe de football à faux.

L'historique de la navigation dans l'hypertexte a la remarquable propriété de permettre, dans certains cas, de lever ce type d'ambiguïté.

L'historique de la navigation est un arbre dont les noeuds sont des identificateurs d'instance de noeuds hypertexte (nom, valeur des paramètres) et les arcs sont des paires d'instance de noeuds étiqueté par l'élément. La racine de l'arbre est l'instance de noeud d'où l'utilisateur a commencé sa navigation, les feuilles sont les instances de noeud actuellement affichées et les autres noeuds sont des instances de noeuds visitées par la navigation (et éventuellement encore affichées).



Reprenant l'exemple de la base de l'Université, un noeud affichant la liste des membres aura une opération de suppression dont la sémantique dépendra du dernier lien parcouru ("liste des joueurs" ou "liste des membres").

5. CONCLUSION

Nous avons développés plusieurs prototypes de systèmes de vues hyperdocument comprenant un compilateur du langage de spécification et un serveur de noeuds. Ces prototypes fonctionnent sur diverses bases de données (O2, Oracle, etc.) et produisent des documents HTML. Ces systèmes ont été utilisés sur plusieurs bases en exploitation (données financières, catalogue de produits, programmes d'étude.

Le prototype le plus récent permet de stocker les définitions de vues dans la base de données elle-même. On obtient ainsi un véritable dictionnaire de spécifications de documents virtuels. Il est donc possible de modifier dynamiquement les définitions des vues pour en améliorer la qualité ou pour les adapter aux nouveaux besoins.

Actuellement nous développons les concepts théoriques nécessaires à la mise à jour à travers les hyperdocuments virtuels, dans le but d'étendre le prototype pour répercuter la mise à jour des hyperdocuments sur les données de la base. Nous étudions également la possibilité de créer des vues par extension de vues existantes.

BIBLIOGRAPHIE

[ACM 95] Special section: Hypermedia Design, Communications of the ACM, Vol. 38, No. 8, 1995.

[AG 97] P. Atzeni, G. Mecca. "Cut and Paste". In PODS'97, Tucson (Arizona), 1997.

[AMM 97] P. Atzeni, G. Mecca, P. Merialdo. "To Weave the Web". In International Conf. on Very Large Databa Bases (VLDB'97), Athens, 1997.

[FGN 99] G. Falquet, L. Nerima, J. Guyot. "Languages and Tools to Specifiy Hypertext Views on Databases". In The World Wide Web and Databases, P. Atzeni, A. Mendelzon, G. Mecca (Eds.), (LNCS Vol. 1590), Springer, 1999.

[FFK+98] M. Fernandez, D. Florescu, J. Kang, A. Levy, D. Suciu. "Catching the Boat with Strudel: Experiences with a Web-Site Management System". In Proc. ACM SIGMOD Conf., Seattle, pages 414-425, 1998

[LRV 88] C. Lécluse, P. Richard, F. Velez. "O2, an Object-Oriented Data Model ". In Proc. ACM SIGMOD, Chicago, 1988.

[Mas 84] Y. Masunaga. "A Relational Database View Update Translation Mechanism". In Proc. VLDB Conf., Singapore, 309-320, 1984.

[NS 96] T. Nguyen, V. Srinivasan. "Accessig Relational Databases from the World Wide Web". In Proc. ACM SIGMOD Conf., 529-540, 1996.

[PV 98] F. Paradis, A-M. Vercoustre. "A Language for Publishing Virtual Documents in the Web". In Proc. of the WebDB'98 Workshop, Valencia, 1998.

[Sindoni 99]G. Sindoni. "Incremental Maintenance of Hypertext Views". In The World Wide Web and Databases, P. Atzeni, A. Mendelzon, G. Mecca (Eds.), (LNCS Vol. 1590), Springer, 1999.

[SC 99]J. Siméon, S. Cluet. "Using YAT to Build a Web Server". In The World Wide Web and Databases, P. Atzeni, A. Mendelzon, G. Mecca (Eds.), (LNCS Vol. 1590), Springer, 1999.