L'arbre recouvrant dans l'interrogation de documents XML

Abdeslame ALILAOUAR

Institut de recherche en informatique de Toulouse
Université de Paul Sabatier. 118 route de Narbonne
31062 Toulouse, cedex 4
laouar@irit.fr

RÉSUMÉ.: Dans cet article nous présentons un modèle d'interrogation flexible de données semi-structurées (DSS) en général et de documents XML en particulier, en prenant en compte non seulement le contenu mais aussi la structure de ces documents. La notion d'arbre recouvrant de taille minimale est employée pour déterminer les fragments de documents qui coïncident le plus possible avec une requête et la théorie des sous-ensembles flous sera utilisée pour représenter les critères flexibles sur le contenu à travers des formules pondérées.

ABSTRACT. In this paper, we propose a fuzzy model to querying the XML documents, by taking into account not only the document contents, but also their structure. The concept of minimal size spanning tree is employed to determine the fragments of documents which coincide as much as possible with a query tree and the fuzzy logic framework is used to represent a user's criteria through weighted formulas according to the level of user's knowledge in documents structures

MOTS-CLÉS : Données semi-structurées, Interrogation flexible

KEYWORDS: semi-structured data, flexible querying.

CORIA 2007, pp. 491–496
Saint-Étienne, 28–30 mars 2007

1. Introduction

La popularité croissante et l'évolution flagrante de l'utilisation de documents Xml ont amené XML (eXtensible Markup Language) à devenir le standard incontournable de description, et d'échange de données semi-structurées(DSS). La nature de ces données dite semi-structurées est différente de celles manipulées par les systèmes classiques de gestion de bases de données ou encore celles utilisées dans les systèmes documentaires usuels. Il s'agit, pour la plupart, de données qui ont une certaine régularité mais dont la structure sous-jacente reste limitée, implicite, inconnue a priori et hétérogène d'une source à une autre, voire dans la même source. En effet, pour manipuler ces données et en extraire les informations pertinentes en termes de structure et/ou de contenu pour l'utilisateur, de nombreux langages d'interrogation ont été développés. Dans ce papier, nous proposons un modèle d'interrogation flexible pour les DSS en général et pour les documents XML en particulier. Le cadre de la logique floue sera utilisé, pour représenter les critères d'interrogation au travers de formules pondérées. En outre, à la fin du processus d'interrogation, chaque réponse est associée à un degré de pertinence compris dans l'intervalle]0,1].

2. Interrogation de documents XML

Avant d'explorer le problème de l'interrogation de documents XML, notons que ces documents peuvent être classés en deux catégories : "orientés texte" et "orientés données". Les documents "orientés texte" sont caractérisés par une structure irrégulière, des données qui présentent une granularité plus grosse et beaucoup de contenus mixtes. Citons comme exemple les livres et les e-mails. Les documents "orientés données" sont caractérisés par une quasi-absence de contenu mixte, une structure assez régulière ainsi que des données qui présentent une granularité fine. Un exemple d'utilisation de ces documents est l'échange de données numériques entre applications. Cette dernière catégorie de documents est le centre intérêt de ce papier. l'interrogation de documents XML est un sujet de recherche qui a rencontré un intérêt croissant depuis de nombreuses années et pour lequel plusieurs langages ont été proposés. En fait, interroger des données semi-structurées conduit à comparer deux structures : celle de la requête et celle du document cible.

```
<?xml version="1.0" encoding="UTF-8"?>
<Résidences>
<appartement numéro="123">
  <description>
     <chambre> <nlits>3</nlits> </chambre>
     <chambre> <nlits>2</nlits> </chambre>
  </description>
  <prix>1400</prix>
</appartement >
<appartement numéro ="140">
  <description>
    <nlits>4</nlits>
  </description>
    <prix>
      <horssaison>1400</horssaison>
      <été>1700</horssaison>
     </prix>
</appartement >
</Résidences>
```

Figure 1 : Un document XML orienté données

Cependant, en raison de la nature irrégulière (hétérogène, incomplète, inconnue a priori) de la structure de ces données, les méthodes classiques d'interrogation basées sur la mise en correspondance exacte sont inadaptées et risquent de conduire à des réponses incomplètes ou vides. A titre d'exemple, soit le document XML de la figure 1 qui représente un extrait du fichier de l'ensemble des appartements d'une agence de tourisme décrits en termes de prix, et de nombre de chambres et de lits. Dans ce document, on peut constater facilement que les deux descriptions des appartements ont deux structures logiques différentes à un certain degré. En effet, cette différence a le plus souvent lieu dans des sources hétérogènes. On peut dire aussi que les deux appartements proviennent de deux bases de données conçues par deux concepteurs différents. Supposons maintenant qu'un utilisateur cherche une mais appartement on à quatre couchages pour passer ses vacances. Pour cela, il formule la requête XQuery [W3C1] suivante:

```
FOR $lgs IN doc("résidences.xml")//Résidences
WHERE  $lgs//appartement/nlits = 4
RETURN  <maison>
             {$lgs/@numéro}
        </maison>
```

L'exécution de cette requête conduit à un ensemble vide de réponses, dû à la relation de descendance directe ("/") définie entre la balise "appartement" et la balise "nlits" dans la clause "WHERE" de cette requête. Dans le document exemple "nlits" est descendant direct et indirect de "description" et descendant indirect de "appartement ". Une syntaxe qui peut se voir comme une solution à ce problème est le remplacement de ("/") par ("//"). Malheureusement ce n'est pas le cas, car cette solution ne fait pas la différence entre le nombre de lits par chambre et par appartement.

Dans ce cas, le recours aux techniques d'interrogation flexible et la réponse sous forme d'une liste ordonnée de réponses selon la correspondance aux critères de l'utilisateur, représentent une opportunité intéressante.

3. Les sous-ensembles flous et l'interrogation flexible

La théorie des sous-ensembles flous a pour objet d'étudier la représentation des connaissances imprécises ainsi que la mise en œuvre d'un mécanisme de raisonnement approximatif, capable d'utiliser et de prendre en compte des connaissances imprécises et incertaines, tel que l'être humain est capable de le faire. En effet, cette logique peut être appliquée ici dans deux cas, premièrement pour exprimer des préférences dans les requêtes en utilisant des prédicats flous (forme assouplie des critères booléens habituels), deuxièmement pour ordonner l'ensemble des réponses d'une requête. En effet, la nature de ces documents n'est pas la seule raison de l'insuffisance de méthodes d'interrogation basées sur l'appariement strict. Il est parfois souhaitable de permettre aux utilisateurs d'interroger ces données en n'exigeant qu'un minimum de pré-requis, autrement dit : permettre aux utilisateurs de formuler des requêtes qui contiennent ou peuvent contenir des éléments plus ou moins obligatoires.

Ceci peut être modélisé en logique floue par un ensemble de critères flous (i.e. préférences) $\{C_i : i = 1, \ldots, m\}$ sur les données avec leurs niveaux d'importance w_i: i=1,…,m l'un par rapport à l'autre avec $w_i \in]0,1]$. La théorie des possibilités est utilisée pour représenter le degré d'importance par le degré de nécessité $N(C_i)$ évaluant combien le critère C_i est impératif, on écrit donc $N(C_i) = w_i$.

La formule (C_i, w_i) peut être représentée par une sorte de contrainte stricte en utilisant le degré d'appartenance μ_{Ci}* comme suivant :

$$\mu_{C_i*}(\alpha) = \begin{cases} 1 & \text{si } \alpha \text{ satisfait } C_i \\ 1 - w_i & \text{sinon (i.e. } \alpha \text{ viole } C_i) \end{cases} \qquad (1)$$

α est une solution. w_i représente dans quelle mesure il est nécessaire de satisfaire C_i et 1- w_i indique dans quelle mesure il est possible de le violer. Si C_i est lui même un critère flou, nous pouvons écrire :

$$\mu_{C_i*}(\alpha) = \max\left(1 - w_i, \mu_{C_i}(\alpha)\right) \qquad (2)$$

$\mu_{Ci}(\alpha)$ est le degré de faisabilité de la réponse potentielle α selon le critère C_i, tels que $\mu_{Ci}(\alpha)=1$ indique que la réponse α est totalement satisfaite pour le critère C_i, tandis que $\mu_{Ci}(\alpha) = 0$ indique que le critère C_i est violé totalement

4. Représentation de requêtes

Une requête formulée par un utilisateur qui ne possède pas de connaissance exhaustive ou éventuellement qui possède une incertitude sur la structure logique de données à interroger est considérée comme une requête structurellement imprécise. Par conséquent, nous ne nous contentons pas de ne chercher que les données dont la structure hiérarchique correspond exactement à la structure donnée dans la requête. En effet, les requêtes formulées dans ce cas-ci sont modélisées sous forme d'arbre modèle pondéré (weighted tree patterns). Un arbre modèle (tree pattern) est un arbre dont les nœuds sont étiquetés par des noms de variables (nom de balise ou attribut) ou par des valeurs. Un arbre modèle pondéré est un arbre modèle dans lequel des poids approximatifs sont associés aux nœuds et aux arcs du d'arbre modèle. Le poids représente le degré d'importance des informations associées aux nœuds. Sa valeur peut être calculée automatiquement comme dans (Da et al., 2000), ou donnée directement par l'utilisateur.

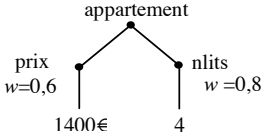

Figure 2. : Une requête sous forme d'arbre pondéré

A titre d'exemple, supposons qu'un utilisateur mal informé sur la structure de document présenté en Fig.1 (résidences.xml) cherche à louer une maison à quatre lits, avec un budget d'environ 1400€, avec une importance égale à 0,8 attribuée au nombre de lits alors que le critère sur le prix est moins important (correspondant par exemple à un poids de 0,6). Cette requête peut être représentée par l'arbre pondéré de la figure 2. Nous pouvons dire que la structure hiérarchique de cette requête est approximative, et structurellement imprécise comparée à celle du document exemple (Fig. 1) du fait que la maison numéro "140" a deux prix (il y a plus de détails dans le document que dans la requête). Ce problème, appelé problème des nœuds éclatés (décomposés), représente le fait qu'un sous-arbre du document à interroger est remplacé par un simple nœud dans l'arbre pondéré de la requête. L'une des solutions envisageables à ce problème est l'agrégation. Il existe plusieurs façons pour agréger (la moyenne, la somme, etc.). On ne peut pas généraliser un seul type d'agrégation dans une solution. Par exemple, dans le cas du prix, on peut utiliser le "prix moyen", dans le cas de "nblits" on peut utiliser la somme de "nblits" de toutes les chambres, etc. cela dépend du contexte. Cependant, pour une meilleure interopérabilité de notre modèle, en cas de problème de ce type, on se contente de retourner le sous-arbre entier à l'utilisateur

5. Algorithme d'appariement

L'algorithme que nous proposons (figure 3) pour chercher les sous-arbres recouvrants est un algorithme bottom-up qui construit les sous-arbres recouvrants de l'arbre de requête progressivement en parcourant l'arbre ce dernière en postordre. Il se compose de deux module : le premier consiste à chercher pour un nœud tout les nœuds qui sont suffisamment similaires (i.e. le poids est plus grand ou égal à un seuil donnée) dans l'arbre de document cible en utilisant les technique données dans la sous section précédente. Le second concerne la validation de la structure. Il prend l'ensemble des nœuds renvoyés par le premier module et essaie de valider leur structure hiérarchique avec celle de l'arbre de requête en faisant des jointures successives (cf. Jag et al., 2002) avec les sous arbres partiels construits à partir des nœud de requête déjà parcourus, de telle sorte que la structure d'arbre de requête soit conservée. Si, au contraire, la structure d'arbre de requête n'est pas conservée, et qu'il existe plusieurs choix de nœuds, on prend les nœuds v qui construisent un sous-arbre de taille minimale en termes de nombre de nœuds.

```
1 :  Arbre_recouvrent(Q, T)
2 :     Trouver_match(v1)
3 :      MT = {{u1},{u2}, .  .  . ,{um}}
4 :      Pour i allant de 2 à n faire
5 :                  Mv = Trouver_match(vi)
6 :                    MT =MT ⋈ Mv ;
7 :      Fin Pour
8 :      Return (MT)
9 :    Fin Arbre_recouvrent
```

Figure 3 : L'algorithmes d'appariement

La ligne 3 permet de initialiser l'ensemble de réponses M_T, Chaque ensemble de l'ensemble M_T un noeud de l'ensemble Mv_1 de nœuds de T similaire à v_1 le premier nœud visité dans le parcours de l'arbre Q en postordre. La boucle de la ligne 4 parcours l'arbre de requête P en postordre. Dans la ligne 6 nous faisons une jointure entre l'ensemble M_T l'ensemble de réponses intermédiaires et l'ensemble M*v* de nœuds de T similaire à v_1

```
1 : Trouver_match(vi)
2 :   Mv = Ø
3 :     Pour j allant de 1 à n faire
4 :       Si (match(vi ,uj) ≥ α) alors
5 :                w(ui) = match(vi, uj)
6 :                Mv = Mv ∪ {uj}
7 :       Fin Si
8 :    Fin Pour
9 :   Return (Mv);
10:  Fin Trouver_match;
```

Figure 4 : La fonction Trouver_match(v_i)

5. Conclusion

Dans ce papier, nous avons abordé le problème de l'interrogation de Données Semi-Structurées en général et de documents XML en particulier. Dans le cadre de la logique possibiliste, nous avons développé un modèle d'interrogation flexible basé sur la notion d'arbre recouvrant de taille minimale. En effet, l'idée de l'application de l'arbre recouvrant dans l'interrogation des données semi-structurées découle du rapprochement fait entre l'approche par relaxation et l'approche basée sur la distance d'édition. L'arbre recouvrant peut être vu comme un cas de la distance d'édition si on n'autorise qu'une seule opération élémentaire : l'ajout de nœud. Quant à la relation avec l'approche par relaxation, lorsqu'on relaxe un critère de descendance directe vers un critère de descendance indirecte, on est dans le contexte de l'arbre recouvrant autrement dit chemin recouvrant (un arbre est un ensemble de chemins) mais il n'y a pas la condition de minimum.

Bibliographie

Amer-Yahia S., Cho S., Srivastava D.. "Tree Pattern Relaxation", In EDBT, Prague, Czech Republic, 2002.

Damiani E., Tanca L., Arcelli F., "Fuzzy XML queries via context-based choice of aggregation", Kybernetika Vol.36, 2000

De Calmès M., Prade H., Sèdes F.. Flexible querying of semi-structured data: a fuzzy set-based approach. Inter. J. of Intelligent Systems, to appear, 2006.

Dubois D., Lang J., Prade H.. Possibilistic logic. *In Handbook of Logic in AI and Logic Programming*, (D. Gabbay et al., eds, 3, Oxford University Press 1994, p. 439–513.

Jagadish H., Al-Khalifa S. , Srivastava D., Structural Joins: A Primitive for Efficient XML Query Pattern Matching, In ICDE, 2002 p.141-153,

Shasha D., Zhang K., Approximate tree pattern matching in pattern matching string, chapter 14. Oxford University Press, 1997.