Bases de données: Examen

Pierre Senellart (pierre.senellart@ens.fr)

29 mai 2019

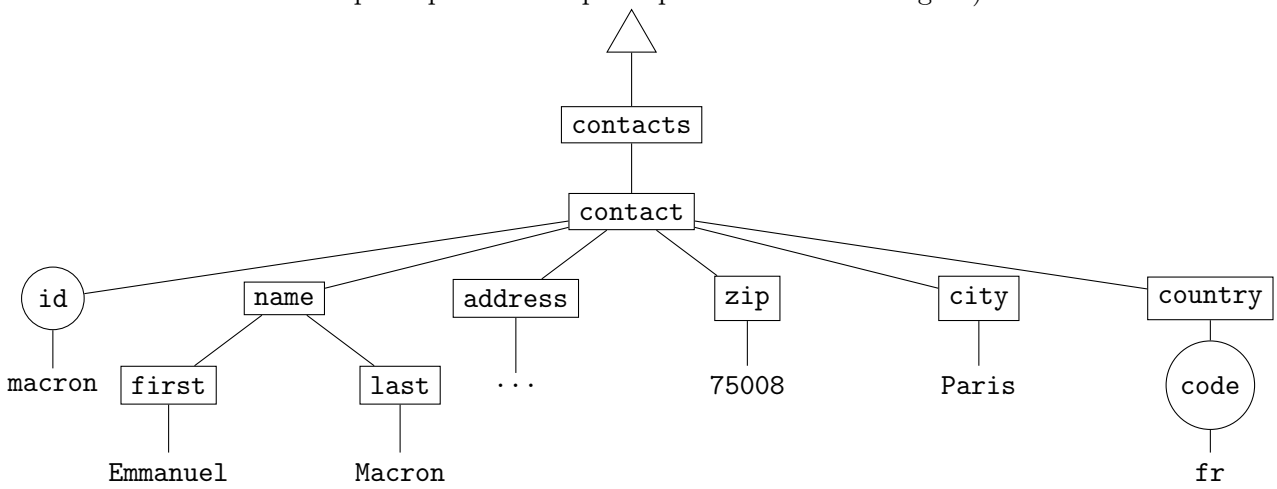
Les seuls documents autorisés sont 2 feuilles recto-verso A4 (4 pages), avec le contenu de votre choix, lisible à l'œil nu. Cet examen dure deux heures et est constitué d'un unique problème de 10 questions, noté sur 20 points.

Encodage de documents XML dans un SGBD relationnel

Un *document XML* est un document mêlant texte et balises (de la forme `<balise>` ou `<balise attribut="valeur">` pour les *balises ouvrantes* et `</balise>` pour les *balises fermantes*), permettant de représenter de l'information semi-structurée. Les balises sont supposées être bien imbriquées (c'est-à-dire qu'à toute balise ouvrante `<balise>` correspond une balise fermante `</balise>` et réciproquement, et une balise ouverte après une autre balise doit être fermée avant celle-ci). Un document XML commence toujours par une balise ouvrante dont la balise fermante correspondante termine le document. Un exemple d'un tel document est le suivant, que l'on notera d_0 (les espaces et retours à la ligne entre balises sont juste pour la clarté de l'affichage, ils ne font pas partie du document) :

```
<contacts>
  <contact id="macron">
    <name>
      <first>Emmanuel</first>
      <last>Macron</last>
    </name>
    <address>55, rue du Faubourg Saint-Honoré</address>
    <zip>75008</zip>
    <city>Paris</city>
    <country code="fr"></country>
  </contact>
</contacts>
```

Un tel document XML peut se représenter de manière graphique comme un *arbre XML*, comme suit (le texte de l'adresse a été remplacé par « ... » pour qu'il tienne dans la figure) :



Un arbre XML est ainsi un arbre (fini) étiqueté, et dans lequel chaque nœud a un type :

— un unique nœud de type *document* (représenté par un triangle) est à la racine de l'arbre XML, et ne comporte pas d'étiquette ;

— les nœuds de type *élément* (représentés par des rectangles) correspondent aux balises du document, et ont pour enfants une représentation sous forme de forêt du contenu entre chaque balise ouvrante et fermante correspondante ;

— les nœuds de type *texte* (sans cadre dans la figure) correspondent aux fragments de texte du document ;

— les nœuds de type *attribut* (représentés par des cercles) correspondent aux attributs pouvant figurer sur les balises ouvrantes), et ont pour unique fils un nœud texte comportant la valeur de l'attribut.

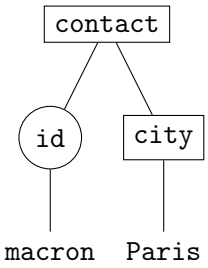
Pour simplifier, dans ce problème, nous supposerons que cet arbre est *non-ordonné*, c'est-à-dire que l'ordre des enfants de chaque nœud n'est pas spécifié et n'a pas d'importance.

On souhaite utiliser un SGBD relationnel pour stocker et interroger de manière efficace une collection de documents XML arbitraires (ne ressemblant pas forcément à d_0), représentés sous la forme d'arbres XML. Chaque document de la collection a un identifiant, qui est un entier naturel.

1. (2 points) Proposer un diagramme entité–association permettant de représenter une collection d'arbres XML ; ce diagramme devra faire en particulier figurer une entité *Nœud*.
2. (2 points) En déduire un schéma relationnel. Faire figurer sur ce schéma relationnel l'ensemble des contraintes pertinentes, en particulier clefs primaires, clefs étrangères, possibilité pour les attributs d'être `NULL`, autres contraintes `CHECK`.
3. (1 point) Le schéma obtenu à la question précédente est-il en forme normale de Boyce–Codd ? Prouver votre affirmation. S'il ne l'est pas, décomposer le schéma sans perte pour obtenir un schéma en forme normale de Boyce–Codd.
4. (1 point) Donner l'*encodage relationnel* (une base de données dans ce schéma) d'une collection comportant un unique document, d_0.

   On souhaite maintenant permettre d'effectuer des requêtes sur des documents XML stockés dans notre base. Pour cela, on utilise en général des langages de requêtes pour XML tels que XPath ou XQuery. On va donner une version très simplifiée d'un tel langage, les *requêtes de motif d'arbre*. Pour simplifier également, toutes les requêtes que l'on va considérer sont booléennes : elles retournent soit Vrai soit Faux.

Une requête de motif d'arbre est un arbre étiqueté et dans lequel chaque nœud a un type (document, élément, texte, attribut). Contrairement à un arbre XML, une requête de motif d'arbre ne comporte pas nécessairement de nœud de type « document ». L'arbre suivant est par exemple une telle requête q_0 :



Une requête de motif d'arbre q s'évalue à Vrai sur un document XML d (ce que l'on note $d \models q$) si et seulement s'il existe un homomorphisme de q vers l'arbre XML t correspondant à d (c'est-à-dire une fonction h qui envoie les nœuds de q vers les nœuds de t en préservant les étiquettes, les types, et la relation parent–enfant). Par exemple, la requête q_0 s'évalue à Vrai sur le document exemple d_0.

Une requête de motif d'arbre q s'évalue à Vrai sur une collection de documents XML C s'il existe $d \in C$ tel que $d \models q$.

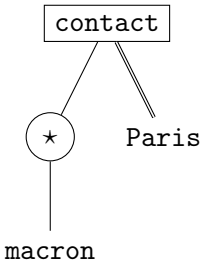
5. (1 point) Exprimer la requête q_0 sous la forme d'une requête conjonctive Q_0 sur l'encodage relationnel D d'une collection C de documents XML, telle que $D \models Q_0$ si et seulement si $C \models q_0$.
6. (1 point) Exprimer Q_0 en SQL.

On ajoute aux requêtes de motif d'arbre deux fonctionnalités supplémentaires :

— L'étiquette d'un nœud peut prendre la valeur spéciale $\star$ qui sert de joker : cette étiquette est ignorée par l'homomorphisme.

— Les arêtes d'un nœud n de l'arbre à son fils n' peuvent être doubles, on parle d'arêtes *descendantes*. Dans ce cas, la condition sur l'homomorphisme h devient que $h(n)$ doit être un *ancêtre strict* de $h(n')$, pas forcément son parent.

Ainsi, q_1 est la requête suivante exprime le fait qu'il existe un élément `contact` dont un attribut a la valeur `macron` et qui a un descendant textuel ayant pour étiquette `Paris` :



On voit immédiatement que $d_0 \models q_1$.

7. (1 point) Exprimer la requête q_1 sous la forme d'une requête Datalog Q_1 sur l'encodage relationnel D d'une collection C de documents XML, telle que $D \models Q_1$ si et seulement si $C \models q_1$.
8. (1 point) Exprimer Q_1 en SQL.
9. (2 points) Proposer un algorithme permettant de traduire n'importe quelle requête de motif d'arbre en une requête Datalog sur l'encodage relationnel d'une collection de documents XML. Quelle est la complexité de cette traduction ?

10. (1 point) En déduire une borne supérieure sur la complexité de l'évaluation de requête de motif d'arbre (en complexité en les données, et en complexité combinée).
11. (2 points) Discuter du nombre de pages disque qu'il est nécessaire de parcourir pour évaluer une requête de motif d'arbre via cette traduction vers Datalog. Quelle technique de stockage peut-on imaginer pour réduire ce nombre de pages disque ?
12. (2 points) Exhiber une requête conjonctive sur l'encodage relationnel d'une collection de documents XML qui n'est équivalente à la traduction d'aucune requête de motif d'arbre. Justifier (mais sans preuve formelle) votre affirmation.
13. (1 point) Sans passer par la traduction vers Datalog, proposer un algorithme direct d'évaluation de requêtes de motif d'arbre qui s'exécute en $O(n^k)$ où n est la taille de la collection et k est le nombre de nœuds de la requête.
14. (2 points) Proposer un algorithme pour ce même problème qui soit linéaire en la taille de la collection (c'est-à-dire, linéaire en complexité en les données).