2ème année EPUL - BDAV
18 janvier 2012

Documents autorisés

Nom : **Prénom :**

N° d'étudiant(e):

Exercice1 (Datalog)

On considère les trois relations suivantes:

Employé(x)
A-Comme-Responsable(x,y)
Responsable (y)

Question 1 : Ecrire, en Datalog (sous forme de clause(s)), la requête permettant de retourner tous les employés ayant le même responsable que « Smith »

Question 2 : Ecrire, en Datalog (sous forme de clause(s)), la requête permettant de retourner toutes les personnes (les personnes regroupent les employés et les responsables)

Question 3 : Ecrire, en Datalog (sous forme de clause(s)), la requête permettant de retourner tous les responsables qui sont des employés avec un niveau de responsabilité supérieur ou égal à deux (c'est-à-dire que chaque responsable retourné est responsable au moins d'un autre responsable).

Exercice 2 (inclusion de requêtes)

Soient deux (2) requêtes q_1 et q_2, spécifiées sur le même schéma de base de données.

Question 1 : Quand q_1 est incluse dans q_2 (que l'on écrit $q_1 \subseteq q_2$) ?

Question 2 : Pour chacun des cas suivants, indiquez (en entourant OUI ou NON) si $q_1 \subseteq q_2$?

$q_1(x)$:- R(x,u), R(u,v), R(v,w) $q_2(x)$:- R(x,u), R(u,v) OUI / NON	$q_1(x)$:- R(x,u), R(u,v), R(v,x) $q_2(x)$:- R(x,u), R(u,x) OUI / NON
$q_1(x)$:- R(x,u), R(u,u) $q_2(x)$:- R(x,u), R(u,v) OUI / NON	$q_1(x)$:- R(x,u), R(u,"Smith") $q_2(x)$:- R(x,u), R(u,v) OUI / NON

Exercice 3 (réécriture de requêtes)

Question 1 : On considère les trois relations suivantes:

Employé(x)
A-Comme-Responsable(x,y)
Responsable (y)

Soient les deux vues suivantes

L(x,y) :- ***A-Comme-Responsable*** (x,u), ***A-Comme-Responsable*** (u,y)
E(x,y) :- ***A-Comme-Responsable*** (x,y), ***Employé*** (y)

Soit la requête suivante :

Q(x,y) :- ***A-Comme-Responsable*** (x,u), ***A-Comme-Responsable*** (u,v),
A-Comme-Responsable (v,w), ***A-Comme-Responsable*** (w,y), ***Employé***(y)

Peut-on répondre (et comment) à la requête ***Q*** si nous ne disposons que des vues ***L*** et ***E***?

Exercice 4 (Contraintes d'intégrité)

On considère la base de données déductive suivante, composée d'une partie EDB, d'une partie IDB et de contraintes d'intégrité :

EDB :

E1 : ***Employé(Nom, Salaire, Nom_département)*** /* un employé est identifié par un nom, son salaire et le département où il exerce*/

E2 : ***Dept(Nom_département, Fonction, Nom_responsable)*** /*un département est identifié par un nom, une activité (fonction), et le nom du responsable*/

E3 : ***Emp-Véhicules(Numéro-immatriculation, appartient-à, Couleur)*** /* les véhicules personnels sont identifiés par un numéro, le propriétaire (qui est un employé), et une couleur)*/

E4 : ***Entreprise-Véhicules(Propriétaire, Numéro-immatriculation, Dépt, Couleur, Kilométrage)*** /* un véhicule d'entreprise est identifié par l'entreprise d'appartenance, un numéro, un département au sein de l'entreprise, une couleur, et le nombre de kilomètres*/

(IC1) Les salaires de tous les employés sont inférieurs à 70000 et supérieurs à 10000.

Question 1 : Exprimez sous forme de clause(s) cette contrainte d'intégrité.

(IC2) Le nom de l'employé détermine fonctionnellement (dépendance fonctionnelle) le salaire et le nom du département.

Question 2 : Exprimez sous forme de clause(s) cette contrainte d'intégrité.

(IC3) Tout véhicule dans la relation ***Emp-Véhicules*** appartient à un employé.

Question 3 : Exprimez sous forme de clause(s) cette contrainte d'intégrité.

Question 4 : Ecrire, sous forme de clause, la requête suivante :

La liste des propriétaires (nom) de voitures rouges qui travaillent dans un département publicité (fonction).

Exercice 5 (XQuery et SQL)

On considère la DTD suivante :

```
< !ELEMENT carnet (personne*)>
< !ELEMENT personne (nom,prenom,adresse*)>
< !ATTLIST personne id CDATA #REQUIRED>
< !ELEMENT adresse (numero, rue, ville)>
< !ELEMENT nom (#PCDATA)>
< !ELEMENT prenom (#PCDATA)>
< !ELEMENT numero (#PCDATA)>
< !ELEMENT rue (#PCDATA)>
< !ELEMENT ville (#PCDATA)>
```

On considère également le schéma relationnel suivant :

personne(id, nom, prenom)
adresse(numero, rue, ville, pers)

On souhaite faire établir une correspondance entre les données stockées sous ces deux formats :

1. Ecrire une requête SQL permettant de générer un document XML conforme à la DTD à partir de données stockées dans une base respectant le schéma relationnel fourni. On pourra utiliser les fonctions SQL/XML comme XMLElement, XMLAttribute, XMLAgg etc. On pourra si besoin créer une vue intermédiaire.

2. Ecrire une fonction en pseudo PL/SQL permettant d'insérer les données d'un document (plus précisément le nœud racine du document) conforme à la DTD dans une base respectant le schéma relationnel fourni. Pour parcourir le document, on dispose des fonctions suivantes :
    - eval(xpath,nœud)
      Cette fonction envoie la liste des nœuds correspondant à l'évaluation de l'expression XPath fournie (sous forme de chaîne de caractères) à partir du nœud fourni.
    - val(noeud)
      Cette fonction donne la valeur d'un nœud texte.
    - nom(nœud)
      Cette fonction donne le nom d'un élément.
    - for n in (une liste) loop … end loop ;
      Cette construction est ajoutée au langage et permet d'itérer sur des listes de nœuds.

3. On considère l'expression XQuery suivante :

```
<voisinages>
{ for $p in //personne, $a in $p/adresse,
  $p2 in //personne, $a2 in $p2/adresse
   where $p/@id != $p2/@id
     and $a/ville/text() = $a2/ville/text()
     and $a/rue/text() = $a2/rue/text()
   return
     <voisins>{$p/nom}{$p2/nom}</voisins>
  }
</voisinages>
```

Ecrire une requête SQL telle que cette requête renvoie le même résultat que la requête XQuery ci-dessus appliquée au résultat de la requête de la question 1.

Exercice 6 (Requêtes hiérarchiques, XML et indexation)

On considère des documents XML constitués uniquement d'éléments et de nœuds texte. On considère le schéma relationnel suivant pour stocker ces documents :

noeud(id, iddoc, type, pere, rang, nom, valeur)

- id est l'identifiant du nœud
- iddoc est l'identifiant document du nœud
- type est soit 'E' pour les éléments, soit 'T' pour les nœuds texte
- pere est une clé étrangère vers l'identifiant du nœud père (peut être NULL)
- rang est le rand du nœud parmi les enfants du père (vaut NULL si pere vaut NULL)
- nom est le nom de l'élément (vaut NULL pour les nœuds texte)
- valeur est la valeur du nœud texte (vaut NULL pour les éléments)

Traduire les expressions XPath suivantes en SQL (avec les extensions CONNECT BY) sur ce schéma relationnel, en supposant que ces expressions sont évaluées à partir de la racine du document n°1 (la requête devra renvoyer l'ensemble des numéros des nœuds qui auraient été renvoyés par la requête XPath):

a) /carnet//adresse/text()

b) //wagon[type/text()=’cargo’]

Lister les attributs du schéma relationnel qu’il pourrait être intéressant d’indexer pour optimiser la requête (b) et expliquer brièvement pourquoi l’indexation des autres est inutile.