

Université Paris 7
Master 1 Informatique, Bases de données avancées.
10 janvier 2014

Durée : 2h30 Documents manuscrits, notes de cours, notes de TD / TP

Exercice 1 - Dépendances - 1.5 points

On considère une relation $R(A, B, C, D, E)$ qui satisfait les dépendances fonctionnelles suivantes :

$$\begin{aligned} A, B &\rightarrow C \\ D &\rightarrow C \\ D &\rightarrow E \\ E &\rightarrow A \end{aligned}$$

Question 1: Donner toutes les clés candidates de la relation R . Justifier. (Justifier ne veut pas dire du'il faut calculer systématiquement toutes les clôtures possibles.)

Exercice 2 - Triggers et Fonctions - 2 points

Le site de vente aux enchères gère une base de données qui contient les tables suivantes :

```
create table client(  
    id_client int primary key check(id_client > 0),  
    nom varchar(30) not null,  
    prenom varchar(30)  
);  
  
create table objet_en_vente(  
    id_objet int primary key check(id_objet >= 0),  
    id_vendeur int not null references client,  
    date_mise_en_vente timestamp not null,  
    date_fin timestamp not null check(date_fin > date_mise_en_vente),  
    prix_reserve decimal(12,2) check(prix_reserve > 0)  
);  
  
create table objet_vendu(  
    id_objet int primary key check(id_objet >= 0),  
    id_vendeur int not null references client,  
    id_acheteur int not null references client,  
    date_fin timestamp not null,  
    prix_vente decimal(10,2) check(prix_vente > 0)  
);  
  
create table objet_non_vendu(  
    id_objet int primary key check(id_objet >= 0),  
    id_vendeur int not null references client,
```

```

    date_fin timestamp not null ,
    prix_reserve decimal(12,2) check(prix_reserve > 0)
);

create table offre (
    id_offre int primary key,
    id_objet int not null references objet_en_vente ,
    id_acheteur int not null references client ,
    date_offre timestamp not null ,
    prix_offert decimal(12,2) not null check(prix_offert > 0),
);

```

client La table client regroupe les vendeurs et acheteurs.

objet_en_vente La table objet_en_vente contient tous les objets actuellement en enchère. L'attribut date_fin donne la date et temps (timestamp) de fin d'enchère. L'attribut prix_reserve

objet_vendu La table objet_vendu contient tous les objets vendus.

objet_non_vendu La table objet_non_vendu contient tous les objets non vendus pour lesquels l'enchère est terminée. L'objet peut être non vendu pour deux raisons :

- il n'y avait aucune offre pour cet objet,
- tous les offres étaient inférieures au prix_reserve fixé par le vendeur.

offre La table offre sert à enregistrer les prix proposés par les acheteurs. La table contient les offres des acheteurs pour les enchères en cours. Chaque nouvelle offre est ajoutée dans la table sans qu'on change les offres précédentes.

Question 1: Écrire un trigger qui sera déclenché à la suppression d'un objet de la table objet_en_vente.

Dans le cas (1) le trigger implémente les actions suivantes :

prix_reserve alors l'objet
la table objet_vendu

(A) si le plus grand prix offert pour l'objet supprimé est supérieur au prix_reserve alors l'objet est vendu au plus offrant est le trigger doit enregistrer la vente dans

- (B) si le plus grand prix offert pour l'objet est inférieur ou égal au prix_reserve ou s'il n'y a pas d'offres pour l'objet alors l'objet le trigger doit enregistrer l'objet dans la table objet_non_vendu,

Finalement aussi bien dans le cas (2) d'annulation d'enchère que dans le cas (1) de fin d'enchères le trigger doit supprimer de la table offre toutes les offres concernant l'objet.

Question 2: Écrire un trigger qui sera déclenché par l'opération d'insertion dans la table

le même objet,

soit au moins 1% plus grand que la meilleure offre précédente),

(B) la date `do` est plus grande que la date de la dernière offre pour

(C) la date `do` est inférieure à la date `date fin` pour l'objet

Bien sûr les conditions (A) et (B) sont à vérifier uniquement s'il y a déjà des offres pour l'objet

i.

Si une de condition (A)-(C) n'est pas satisfaite alors l'opération d'insertion de l'offre doit être

<AUTEURS>

<AUTEUR identou

nal complexity: a modern approach</TITRE>

e University Press</EDITEUR>

2425-4</ISBN>

E>

<TITRE>Computatic

<EDITEUR>Cambridg

<ISBN>978-0-521-4

<ANNEE>2009</ANNE

</LIVRE>

<LIVRE lang=" fr ">

```

<AUTEURS>
  <AUTEUR idauteur="5"/>
  <AUTEUR idauteur="6"/>
</AUTEURS>
<TITRE>
  Analyse mathématique.
  Grands théorèmes du vingtième siècle.
</TITRE>
<EDITEUR> Calvocoressi, Marcello / <EDITEUR>

```

```

  <ANNEE>2009</ANNEE>
</LIVRE>

  <LIVRE lang="fr">
    <AUTEURS>
      <AUTEUR idauteur="7"/>
    </AUTEURS>
    <TITRE>
      Les liaisons dangereuses
    </TITRE>
    <EDITEUR> Gallimard</EDITEUR>
    <ANNEE>1972</ANNEE>
  </LIVRE>
</LIVRES>

```

```

  <PRENOM>Sanjeev</PRENOM>
  <NOM>Arora</NOM>
  <AnneeNaiss>1968</AnneeNaiss>
</AUTEUR>
  <AUTEUR idauteur="4">
    <PRENOM>Boaz</PRENOM>
    <NOM>Barak</NOM>
  </AUTEUR>
  <AUTEUR idauteur="5">
    <PRENOM>Denis</PRENOM>
    <NOM>Choimet</NOM>
  </AUTEUR>
  <AUTEUR idauteur="6">
    <PRENOM>Hervé</PRENOM>
    <NOM>Queffélec</NOM>
  </AUTEUR>
  <AUTEUR idauteur="7">
    <PRENOM>Pierre</PRENOM>
    <NOM>Choderlos de Laclos</NOM>
    <AnneeNaiss>1741</AnneeNaiss>
    <AnneeMort>1803</AnneeMort>
  </AUTEUR>
</AUTEURS>

</CATALOGUE>

```

Le fichier complet contient beaucoup plus de livres et d'auteurs et les questions qui suivent concernent le fichier complet catalogue.xml et non le petit extrait ci-dessus.

Comme vous ne pouvez pas constater, catalogue.xml est un catalogue de livres, le noeud CATALOGUE contient deux fils : LIVRES et AUTEURS.

Le fils LIVRES du CATALOGUE contient plusieurs livres. Chaque livre peut avoir plusieurs auteurs¹.

Question 1: Écrire une expression XPath qui donne les titres de livres écrits en français et publiés après 1970.

Question 2: Expliquer ce que affiche la requête XQuery suivante :

```
for $x in distinct-values(//LIVRE[EDITEUR='Gallimard']//AUTEUR/@idauteur)
for $y in /CATALOGUE/AUTEURS/AUTEUR
where $y/@idauteur = $x
return <li> { $y/NOM/text(), ' ', $y/PRENOM/text() } </li>
```

Remarque : La fonction distinct_values supprime les doublons.

Question 3: Écrire une requête XQuery qui affiche les titres de tous les livres d'Alexandre Dumas. Le résultat doit sortir sous la forme :

```
<tr> <td> titre 1 </td> <td> lang 1 </td> </tr>
<tr> <td> titre 2 </td> <td> lang 2 </td> </tr>
...
<tr> <td> titre n </td> <td> lang n </td> </tr>
```

où lang est la valeur de l'attribut lang

Exercice 4 - transactions (1.5 point)

Question 1: On exécute la transaction suivante :

```
start transaction;
insert into T values (1,2);
savepoint A;
insert into T values (2,3);
savepoint B;
insert into T values (3,4);
rollback to A;
insert into T values (5,6);
rollback to A;
insert into T values (5,6);
commit;
```

En supposant que la table T est initialement vide, quel est le contenu de T à la fin de la transaction ?

1. Noter que les éléments AUTEUR qui apparaissent comme les descendants d'un LIVRE donnent les auteurs de ce livre et contiennent uniquement l'attribut idauteur qui permet d'identifier l'auteur. L'auteur lui-même est présenté en détail dans la partie CATALOGUES/AUTEURS du catalogue. En particulier il ne faut pas confondre LIVRE avec AUTEURS et AUTEUR qui sont le fils et petit-fils d'un LIVRE et petit-fils de CATALOGUE

Question 2: On exécute en parallèle deux transactions :

```
start transaction
isolation level serializable;

insert into s values(1,1);
select * from S;
```

commit;

```
START transaction
isolation level read committed;
```

```
select * from S;
```

```
select * from S;
commit;
```

La transaction à gauche c'est la transaction $T1$, à droite $T2$. L'ordre d'exécution de différentes commandes est celui de haut vers le bas, par exemple commit de $T1$ est entre deux select de $T2$. Initialement la table S est vide. Indiquer ce que affiche chaque select.

Question 3: Est-ce que quelque chose change dans l'exercice précédent si $T2$ était ouvert avec isolation level serializable? (On garde l'ordre d'exécution de différentes commandes et

```
explain
select *
from X,Y
where X.A = 4 and X.B=Y.B;
```

et on obtient le plan d'exécution suivant :

Hash Join

Hash Cond: (x.b = y.b)

-> Hash

-> Seq Scan on x

Filter: (a = 4)

-> Seq Scan

Il y a un index B-arbre sur l'attribut A de

Exécutera cette requête select ?

cette requête select sera réalisée (quelle

(les estimation de coût sont supprimées). On sait qu'il y a un index B-arbre sur l'attribut A de la table X .

Question 1: Est-ce que cet index est utilisé si on exécute cette requête select ?

Question 2: Expliquer très brièvement comment sont réalisées les opérations élémentaires dans quel ordre ?