

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

15 janvier 2013

Durée : 2h30 Documents manuscrits, notes de cours, notes de TD/TP  
autorisés. Livres, ordinateurs, téléphones portables interdits.

Le sujet comporte 4 pages.

Le barème est donné à titre indicatif.

### Exercice 1 - Dépendances - 6 points

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

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

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

**Question 2:** Pourquoi cette relation n'est pas BCNF ?

**Question 3:** Décomposez cette relation en relations BCNF.

**Question 4:** Est-ce que la même relation  $R(A, B, C, D, E)$  est 3NF ? Justifier la réponse.

**Question 5:** Si  $R$  n'est pas 3NF alors décomposez  $R$  en relation 3NF.

### Exercice 2 - Triggers et fonctions - 7 points

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

```
create table vendeur(  
  id_vendeur int primary key check(id_vendeur >= 0),  
  nom_vendeur varchar(30) not null,  
  prenom_vendeur varchar(30),  
);  
  
create table acheteur(  
  id_acheteur int primary key check(id_acheteur >= 0),  
  nom_acheteur varchar(30) not null,  
  prenom_acheteur varchar(30),  
);  
  
create table objet(  
  id_objet int primary key check(id_objet >= 0),
```

```

type_objet varchar(50) not null,
id_vendeur int not null references vendeur,
date_limite timestamp not null,
prix_reserve decimal(10,2) check(prix_reserve > 0)
);

```

```

create table offre(
id_objet int references objet,
id_acheteur int references acheteur,
date_offre timestamp not null,
prix_achat decimal(12,2) not null check(value > 0),
primary key(id_objet, id_acheteur)
);

```

**vendeur et acheteur** Les tables vendeur et acheteur permettent d'enregistrer tous les vendeurs et acheteurs.

**objet** La table objet contient tous les objets actuellement en vente. L'attribut date\_limite donne la date et temps de fin d'enchère pour l'objet donné. L'attribut prix\_reserve c'est le prix minimal fixé par le vendeur, le vendeur refuse de céder l'objet pour un prix inférieur à prix\_reserve.

**offre** La table offre sert à enregistrer les prix proposés par les acheteurs. Pour un objet O et un acheteur A la table contient uniquement la dernière offre de l'acheteur A concernant O (pour cette raison le couple id\_objet, id\_acheteur sert de clé pour cette table). L'attribut prix\_achat donne le prix proposé par acheteur et l'attribut date\_offre indique quand cette offre a été reçue.

**Question 1:** Cet exercice consiste à écrire un trigger qui sera déclenché par l'action INSERT sur la table offre. Écrire un trigger cela veut dire écrire la fonction PL/pgSQL qui implémente le trigger et écrire CREATE TRIGGER approprié.

Ce trigger sera déclenché **avant** l'exécution de INSERT (cette remarque doit être prise en compte dans votre CREATE TRIGGER).

Soit

```
INSERT INTO offre VALUES (O,A,T,V) (1)
```

l'action qui déclenche le trigger.

Le trigger vérifie si l'acheteur A a déjà fait une offre pour l'objet O, c'est-à-dire si la table offre contient déjà un enregistrement (O, A, T1, V1) avec les mêmes valeurs O et A.

Plusieurs cas sont possibles.

(A) L'offre (1) est la première offre de l'acheteur A pour l'objet O.

Dans ce cas l'enregistrement (O, A, T, V) sera ajouté dans la table offre à condition que le prix V proposé est supérieur ou égal au prix prix\_reserve fixé par le vendeur

(voir la table objet)

- (i) si  $V \leq V1$  alors la nouvelle offre est illégale<sup>1</sup> La nouvelle offre ne sera pas prise en compte (le contenu de la table offre ne doit pas changer).
- (ii) si  $V > V1$  alors la nouvelle offre (O, A, T, V) doit remplacer l'ancienne (O, A, T1, V1) dans la table offre. Donc l'action INSERT dans la table offre qui déclenche le trigger ne sera pas exécutée, à la place de cette action le trigger lance l'exécution de l'action UPDATE sur la même table qui met à jour l'offre existante (et le timestamp).

Exercice 9

Les questions ci-dessous sont basées sur le document films.xml suivant :

```
<?xml version="1.0" encoding="iso-8859-1"?>
<FILMBASE>
```

```
<FILMS>
```

```
<FILM annee="1958" idfilm="3">
```

```
<TITRE>Vertigo</TITRE>
```

```
<GENRE>Drame</GENRE>
```

```
<PAYS>GB</PAYS>
```

```
<ROLES>
```

```
<ROLE idact="1">
```

```
John Ferguson
```

```
</ROLE>
```

```
<ROLE idact="2">
```

```
Madeleine Elster
```

```
</ROLE>
```

```
</ROLES>
```

```
</FILM>
```

```
<FILM annee="1997" idfilm="4">
```

```
<TITRE>Titanic</TITRE>
```

```
<GENRE>Drame</GENRE>
```

```
<PAYS>USA</PAYS>
```

```
<ROLES>
```

```
<ROLE idact="3">
```

```
Rose DeWitt Bukater
```

```
</ROLE>
```

```
<ROLE idact="4">
```

```
Jack Dawson
```

```
</ROLE>
```

```
</ROLES>
```

```

        <PRENOM>Kate</PRENOM>
        <NOM>Winslet</NOM>
    </ACTEUR>
    <ACTEUR idact="4">
        <PRENOM>Leonardo</PRENOM>
        <NOM>DiCaprio</NOM>
    </ACTEUR>
</ACTEURS>

</FILMBASE>

```

On suppose aussi que nous avons un document plus complet `films_full.xml` qui contient plus de noeuds `<FILM>` et `<ACTEUR>`, par contre la structure de deux documents est la même, c'est-à-dire `<FILMBASE>` à la racine avec deux enfants `<FILMS>` et `<ACTEURS>`. Le noeud `<FILMS>` contient plusieurs balises `<FILM>` et le noeud `<ACTEURS>` contient plusieurs balises `<ACTEUR>`. L'attribut `idfilm` permet d'identifier un film, l'attribut `idact` identifie l'acteur et fait le lien entre les films et les acteurs. Par exemple pour le premier film la balise `<ROLE idact="1"> John Ferguson </ROLE>` indique que dans le rôle de John Ferguson joue l'acteur avec `idact="1"`, c'est-à-dire James Stewart.

**Question 1:** Quel est le résultat d'évaluation de l'expression XPath

```
//PAYS[text()='USA']/parent::node()/TITRE/text()
```

sur le document `films.xml` donné au début de l'exercice.