Université Paris 7
Master 1 Informatique, Bases de données avancées.
15 janvier 2013

Durée : 3 heures. Documents manuscrits, notes de cours, notes de TD/TP autorisés. Livres, ordinateurs, téléphones portables interdits.

Le sujet comporte 6 pages.

Dépendances

Exercice 1 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.

Triggers - déclencheurs

Exercice 2

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 pari(
    id_objet int references objet ,
    id_acheteur int references acheteur ,
    quand timestamp not null ,
    mise decimal(12,2) not null check(value > 0) ,
    primary key(id_objet , id_acheteur)
);

```

```

create table pari_max(
    id_objet int references objet ,
    id_acheteur int references acheteur ,
    prix_max decimal(12,2)
    primary key(id_objet , id_acheteur , valeur_max)
);

```

```

create table message(
    id_acheteur int references acheteur not null ,
    temps timestamp default current_timestamp not null ,
    msg varchar(100) not null
);

```

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`.

pari La table `pari` sert à enregistrer les mises des acheteurs. Pour un objet `O` et un acheteur `A` la table contient uniquement la dernière mise de l'acheteur `A` sur `O` (cela implique que le couple `id_objet`, `id_acheteur` sert de clé pour cette table). L'attribut `quand` donne la date et le temps de la mise, l'attribut `mise` donne le prix proposé par l'acheteur.

message La table `message` permet d'enregistrer les messages que le système décide à envoyer aux acheteurs. La valeur de `id_acheteur` permet d'identifier l'acheteur qui recevra le message, `temps` sera toujours égal à `current_timestamp` au moment d'insertion de message et `msg` est le contenu du message. (L'envoi de message sera assuré par un programme spécifique qui ne fait pas partie de notre BD).

pari_max et increment Les tables `pari_max` et `increment` sont utilisées par un système d'enchères automatique.

Le système d'enchères automatique L'acheteur qui n'a ni de temps ni d'envie de rester devant ordinateur pour augmenter ses mises peut utiliser le système d'enchères automatiques. Pour mettre en oeuvre le système d'enchères automatiques pour l'acheteur `id_acheteur = A` et l'objet `id_objet = 0` il suffit d'enregistrer dans la table `pari_max` un triplet

$$(\text{id_objet}, \text{id_acheteur}, \text{prix_max}) = (0, A, P)$$

où P le prix maximal que l'acheteur est prêt à payer pour l'objet `verb+O+`. L'acheteur est le seul à connaître son prix max.

Le système augmente automatiquement la mise de l'acheteur A pour l'objet O dans la table `pari` de qu'il détecte qu'un autre acheteur propose une mise supérieure à celle de A .

Supposons que M_0 soit la valeur maximale de l'attribut `mise` pour l'objet O enregistrée dans la table `pari` (c'est le prix de vente si l'enchère devrait s'arrêter en ce moment), on dit que M_0 est la mise gagnante¹

Soit V_0 la valeur de l'attribut `mise` dans la table `pari` donné par l'acheteur A pour le même objet O (la mise courante de l'acheteur A pour cet objet).

Nous allons dire que l'acheteur `id_acheteur=A` est gagnant² pour l'objet O si $V_0 = M_0$, c'est-à-dire sa mise est la plus élevée (s'il y a plusieurs acheteurs qui satisfont cette condition alors celui avec la valeur de l'attribut `quand` la plus petite est gagnant).

L'algorithme de l'enchère automatique Entrées : l'acheteur A et l'objet `\verb+O+`.

Si A n'est pas enregistré dans la table `pari_max` alors terminer sans rien faire.

Supposons que A soit enregistré dans la table `pari_max` (donc A participe aux enchères automatiques). Si A est gagnant pour l'objet O (dans le sens expliqué ci-dessus) alors terminer sans rien faire.

Si A n'est pas gagnant alors mettre à jour la mise dans la table `pari` pour l'objet O et acheteur A (lancer un `UPDATE` approprié dans la table `pari`).

Intuitivement la nouvelle mise doit être la plus petite mise qui rend A gagnant à condition de ne pas dépasser `pari_max` que l'acheteur a fixé pour l'objet O dans la table `pari_max`.

Cependant il faut satisfaire une condition supplémentaire. On suppose qu'il existe une fonction³

```
FUNCTION increment(mise decimal(12,2)) returns decimal(6,2)
```

qui donne l'augmentation minimale de la mise en fonction du prix M_0 actuel (M_0 est la mise gagnante).

Le système calcule $D = \text{increment}(M_0)$ et la nouvelle mise V'_0 de l'acheteur A en utilisant la formule suivante :

1. si $V_0 + D > M_0 + 0.01$ alors $V'_0 = V_0 + D$,
2. dans le cas contraire $V'_0 = M_0 + 0.01$, c'est-à-dire on mise juste un centime de plus que l'ancienne mise gagnante.

Si V'_0 calculé de cette façon est supérieur à `pari_max` que l'acheteur a fixé dans la table `pari_max` pour cet objet alors la nouvelle mise de A sera égale à `pari_max` et un message dont le contenu est `'prix max insuffisant'` sera ajouté dans la table `message` à l'attention de l'acheteur A .

1. gagnante juste à ce moment là, les enchères ne sont pas finies
2. gagnant à un moment donné, il n'est pas encore le gagnant définitif tant que l'enchère n'est pas terminée
3. on ne demande pas de l'écrire, par contre elle sera utilisée dans votre programme

Si $V_0 \leq \text{prix_max}$ la nouvelle mise sera égale à V_0 et le message 'nouvelle mise gagnante val' est ajouté dans la table de messages (verb+val+ est la valeur de la nouvelle mise) à l'attention de l'acheteur A.

Question 1: Le système d'enchères automatiques sera implémenté à l'aide de la fonction PL/pgSQL

```
CREATE FUNCTION enchere_aut(a acheteur.id_acheteur@TYPE,  
                           o objet.id_objet@TYPE) RETURNS BOOLEAN
```

Cette fonction met à jour l'augmentation automatique de la mise de l'acheteur a pour l'objet o en utilisant l'algorithme décrit ci-dessus. Écrire la fonction `encher_aut`.

Question 2: Le deuxième exercice consiste à écrire un trigger qui sera déclenché par l'action INSERT sur la table `pari` (écrire un trigger veut dire écrire la fonction PL/pgSQL qui implémente le trigger et CREATE TRIGGER approprié).

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

Soit

$$(O, A, T, V) \tag{1}$$

l'enregistrement donc l'insertion a déclenché le trigger (le valeur de quatre attributs de la table `pari` sont donnée dans l'ordre).

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

Plusieurs cas sont possibles.

(A) La mise (1) est la première mise de l'acheteur A pour l'objet O.

Dans ce cas l'enregistrement (O, A, T, V) sera ajouté dans la table `pari`.

(B) La mise (1) n'est pas la première mise de A pour l'objet O, c'est-à-dire la table `pari` contient déjà une autre mise $(O, A, T1, V1)$ du même acheteur A pour l'objet O.

Dans ce cas

- (i) si $V \leq V1$ alors la nouvelle mise est illégale⁴ La mise nouvelle mise ne sera pas pris en compte (le contenu de la table `pari` ne doit pas changer) et on ajoute dans la table `message` un message destiné à l'acheteur A dont le contenu est `mise V illegale`,
- (ii) si $V > V1$ alors l'enregistrement (O, A, T, V) doit remplacer l'enregistrement $(O, A, T1, V1)$ dans la table `pari`. Donc l'action INSERT dans la table `pari` qui déclenche le trigger ne sera pas exécutée, à la place de cette action on lance l'action UPDATE sur la même table qui met à jour la mise (et le timestamp).

Dans les cas (A) et (Bii) la fonction de trigger ajoute un message dont le contenu `msg` est

- soit 'votre mise V gagnante' si l'acheteur A devient gagnant avec la nouvelle mise (gagnant dans le sens de l'exercice précédent).
- soit 'votre mise V insuffisante' dans le cas contraire.

XPath et XQuery

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

4. On ne peut miser que plus qu'avant.

```

<?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>
  </FILM>
</FILMS>

<ACTEURS>
<ACTEUR idact="1">
  <PRENOM>James</PRENOM>
  <NOM>Stewart</NOM>
</ACTEUR>
<ACTEUR idact="2">
  <PRENOM>Kim</PRENOM>
  <NOM>Novak</NOM>
</ACTEUR>
<ACTEUR idact="3">
  <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 `<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.

Exercice 3

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

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

sur le document `films.xml` ci-dessus. Justifier la réponse en indiquant ce qui est sélectionné à chaque étape.

Question 2: Écrire une expression XPath qui trouve les genres de tous les films qui ont été produit en 1958 (l'expression sera évaluée sur le document `films_full.xml` qui contient plusieurs films qui sont produits cette année).

Question 3: Expliquer ce que fait la requête XQuery suivante si elle est évaluée sur le fichier complet `films_full.xml`.

```
for $x in //FILM
where $x/PAYS/text()='USA'
return <tr>
    <td> { $x/TITRE/text() } </td>
    <td> { count($x//ROLE) } </td>
</tr>
```

Qu'est-ce que retourne cette requête si elle est évaluée sur le fichier abrégé `films.xml` donné au début de l'exercice.

(La fonction `count(expr)` retourne le nombre d'éléments qu'on obtient en évaluant l'expression XPath `expr`).

Question 4: Faire une requête XQuery qui donne les titres de tous les films où a joué Kim Novak (on suppose que la requête sera évaluée sur le fichier `films_full.xml` donc il y aura plusieurs films dans la réponse). Chaque titre doit être entouré par les balises `<p>` et `</p>` :

```
<p>titre 1</p>
<p>titre 2</p>
<p>titre 3</p>
...
```

Bien sûr on préfère une réponse où la valeur de l'attribut `idact` de Kim Novak est calculée à l'intérieur de la requête au lieu d'être mise en dur.