Examen de bases de données – APP 2A - IR
Novembre 2015

Documents Autorisés

Exercice 1

Une base de données est utilisée pour représenter les clients d'une très grande entreprise. Chaque client est décrit par les attributs suivants :

NoC : Identification du client **NomC** : Nom de famille du client **Adr1** : 1ère adresse du client
Adr2 : 2ème adresse du client **VilleC** : Ville du client **Cp** : code postal du client

Soit F1 l'ensemble de dépendances fonctionnelles suivant:

1. NoC → NomC, Adr1, Adr2, VilleC, Cp
2. Adr1, Adr2 → VilleC
3. Cp → VilleC

1. Interpréter les dépendances (2) et (3)
2. Construire un schéma en troisième forme normale correspondant à l'ensemble des dépendances fonctionnelles de F1.
3. Illustrer les anomalies de mises à jour dans la relation initiale **Client**. Le schéma normalisé élimine-t-il ces anomalies ? Justifier votre réponse.

Soit F2 un ensemble de dépendances fonctionnelles obtenu en supprimant la dépendance fonctionnelle 2 de l'ensemble initial F1. On obtient

F2 = {NoC → NomC, Adr1, Adr2, VilleC, Cp ; Cp → VilleC}.

4. Interpréter cette suppression.

Exercice 3

Considérons un système de gestion de bases de données acceptant des transactions. Ces transactions sont réalisées par des utilisateurs qui consultent et modifient une base de données comprenant les tables R, S, T et W.
On se propose de définir une politique de transactions dans la situation d'utilisation suivante :

- les utilisateurs modifient les N-Uplets de *R, S* et *T*
- les utilisateurs accèdent (lisent) les n-uplets de *R, S, T* et *W*.

De plus :

- aucune modification de *R* ne peut intervenir lorsque *T* est en cours de modification.
- la modification de *R* entraîne une modification de W car certaines valeurs des attributs de *W* dépendent de valeurs des attributs de *R*

Questions

1- Identifier les problèmes engendrés par les lectures et modifications ainsi définies.
2- Définir les types de verrou puis les verrous à poser pour que cette politique de transactions soit mise en œuvre et conserve la cohérence de la base de données.
3- Définir deux transactions différentes possibles ainsi que leur déroulement. Vous donnerez une transaction de lecture sur les relations ainsi qu'une transaction de modification. Pour décrire ces transactions, on utilisera les notation (`update, read, write, commit`, etc.) vues en cours.

Exercice 3

On considère le modèle logique suivant

```
Personne(id : integer, frere : integer)
```

Questions

1- Indiquer en quelques phrases, la contrainte que chacune des expressions suivantes spécifie

```
a- CREATE ASSERTION A
            CHECK (NOT EXISTS(
      SELECT id FROM Personne GROUP BY id HAVING COUNT(*) > 1
                        ))
b- CREATE ASSERTION B
          CHECK (NOT EXISTS(
      SELECT frere FROM Personne WHERE frere NOT IN
           (SELECT id FROM Personne)
                        ))
c- CREATE ASSERTION C
            CHECK (NOT EXISTS(
      (SELECT * FROM Personne) EXCEPT
      (SELECT * FROM Personne WHERE id = id)
                        ))
d- CREATE ASSERTION D
            CHECK (NOT EXISTS(
   (SELECT id, frere FROM Personne) EXCEPT
   (SELECT frere, id FROM Personne)
            ))
```

2- Les expressions précédentes décrivent des contraintes sur le modèle logique. Pour chacune de ces expressions, expliquer en quelques phrases comment on pourrait les implanter dans une base de données (clés, contrainte, déclencheurs, etc.)

Exercice 4

Considérons le modèle logique suivant permettant de stocker les utilisateurs d'un forum et les messages qu'ils ont écrits. Nous supposons qu'un message peut être écrit par plusieurs utilisateurs.

```
Utilisateur(    id : integer,
                name : varchar)

Message(    id : integer,
            titre : varchar)

Auteurs(    idUtilisateur integer,
            idMessage integer,
        Foreign key idUtilisateur references Utilisateur(id),
        Foreign key idMessage references Message(id))
```

1. Donner un modèle orienté objets associé à ce modèle logique. On pourra utiliser pour cela un diagramme de classes UML sans préciser les méthodes associées aux classes.
2. Décrire une interface de programmation (API) associée au modèle orienté objets obtenu en question 1.
3. Définir une table de correspondances entre les tables et/ou attributs du modèle logique précédent et les classes et/ou attributs du modèle orienté objets obtenus en question 1.
4. Considérons la requête suivante « *Donner la liste des identifiants de messages rédigés par l'auteur de name 'TOTO'* ». On suppose que cette requête est implantée dans le modèle orienté objets obtenu en question 1. En utilisant la table de correspondances précédente, décrire les accès aux tables du modèle logique réalisés par cette méthode.