M1 Informatique

Bases de données avancées

Examen de session 2 (durée 2h)

27 juin 2017

Documents autorisés : dix feuilles A4 recto-verso manuscrites ou imprimées

Modélisation E/R et formes normales

Question 1 1. On souhaite représenter les données (très simplifiées) d'une agence immobilière. L'agence gère plusieurs appartements, disponibles à la vente ou à la location. Tous les appartements de l'agence ont une adresse, un prix (de location ou de vente) et un propriétaire. Pour les appartements à vendre on représente également les frais d'agence, et pour les appartements à louer, on représente les charges mensuelles. Tous les clients de l'agence (y compris les propriétaires) ont un nom, une adresse et un numéro de telephone. Pour chaque appartement il faut stocker un ensemble de rendez-vous. Un rendez-vous est pris par un client à une date pour visiter un appartement. (Attention : un même client doit pouvoir prendre plusieurs rendez-vous pour le même appartement, y compris à la même date)

- Produire un diagramme E/R pour représenter ces données.
- Spécifier toutes les contraintes d'identification et de cardinalité, ainsi que les éventuelles contraintes externes.
  Toute hypothèse faite sur les données, qui n'est pas spécifiée dans ce texte, doit être clairement indiquée.
- Traduire le diagramme E/R dans le modèle relationnel.
- Exprimer les dépendances fonctionnelles sur le schéma obtenu. Vérifier que il est bien en forme normale de Boyce-Codd, par rapport à ces dépendances.

Question 2 Soit $R(A, B, C, D, E, F, G, H, I)$ un schéma de relation avec dépendances fonctionnelles :

$\mathcal{F} = \{$
$E \to C$
$G \to C$
$A \to B$
$F \to C$
$C \to G$
$D \to B$
$C \to AB$
$B \to C$
$H \to I$
$\}$

- Donner toutes les clefs de R.
- Trouver une décomposition de R en *forme normale de Boyce-Codd* sans perte d'information. Est-ce que cette décomposition préserve toutes les dépendances fonctionnelles ?
- Trouver une décomposition de R en *troisième forme normale* sans perte d'information et sans perte de dépendances fonctionnelles. (Se rappeler que il est nécessaire d'abord de minimiser $\mathcal{F}$, c'est-à dire d'abord minimiser toutes les parties gauches, et ensuite éliminer toutes le dépendances fonctionnelles redondantes.)

Indexation

Question 3 On considère un *index hash extensible* construit sur l'attribut A d'une table R. Supposer que l'index utilise des valeurs de hachage à 4 bits. Supposer qu'une valeur de A occupe 40 bytes, qu'un pointeur occupe 8 bytes et que qu'un bloc de la table de hachage a une capacité de 100 bytes (hypothèse irréaliste pour faciliter les calculs). Simuler l'insertion de six enregistrements avec valeurs de hachage de A : 0000, 0001, 0101, 0111, 0100, 1000, dans cet ordre. Supposer l'index hash initialement vide. Montrer l'état de l'index hash après chaque insertion. En particulier montrer, après chaque insertion :

- le contenu des blocs de l'index (indiquer les valeurs de hachage au lieu des valeurs de A),
- la structure des pointeurs aux blocs,
- les valeurs des compteurs de chaque bloc et du compteur global i.

Optimisation

Question 4 Soit la requête suivante

```
SELECT Client.nom, Produit.type
FROM Client, Achat, Produit
WHERE Achat.date = '27/06/2017'
AND  Client.id_client = Achat.id_client
AND  Produit.num_prod = Achat.num_prod
AND Produit.prix > 100
```

Sur le schéma :

Achat (num_prod, id_client, date, quantite)
Client (id_client, nom, ville)
Produit (num_prod, type, prix)

en présence des index I1 sur Client.id_client et I2 sur Produit.num_prod.

- Produire un plan logique, obtenu par traduction directe de la requête en algèbre relationnelle (utiliser uniquement les opérateurs de sélection, projection et produit cartésien, et afficher le plan sous forme d'arbre).
- Modifier ce plan en appliquant des règles d'équivalence algébrique, en suivant les étapes suivantes :
  - faire apparaitre les jointures naturelles dans le plan ;
  - générer un plan différent pour chaque ordre possible des jointures ;
  - dans chacun de ces plans pousser les sélections le plus bas possible, tant que cela rend le plan plus efficace (attention à ne pas pousser trop bas dans le plan) ;
- À l'issue du point précédent on obtient plusieurs plans logiques possibles.
  - Lequel est le plus efficace ? Motiver la réponse.
  - Proposer un plan physique pour le plan choisi.

Transactions

Question 5 On considère les ordonnancements suivants de deux transactions :

```
(a) R1(A), W2(A), W1(A)
(b) R1(A), R2(A), W1(A)
```

Pour (a) et (b) répondre aux questions suivantes :

- Est-ce que l' ordonnancement est possible sous verrouillage à deux phases avec un seul type de verrou (Lock/Unlock) ?
- Est-ce qu'il est possible sous verrouillage à deux phases avec verrous partagés (S-Lock, X-Lock / Unlock) ?
- Dans les cas où la réponse est oui ajouter les verrous aux transactions qui permettent cet ordonnancement sous verrouillage à deux phases.

Triggers

Question 6 On considère le schéma de base de données suivant pour la gestion de réservations de vols :

Info-Vol (num-vol, ville-depart, ville-arrivee, heure-depart, heure-arrivee)
Vol (id-vol, num-vol, date, complet)
Passager (id-pass, nom)
Places (id-vol, categorie-place, num-places, prix)
Reservation (id-vol, id-pass, categorie-place)

Les clefs primaires sont soulignées. Les clefs étrangères sont les suivantes :
Vol(num-vol) $\subseteq$ Info-Vol(num-vol) ; Places(id-vol) $\subseteq$ Vol(id-vol) ;
Reservation(id-pass) $\subseteq$ Passager(id-pass) ;
Reservation(id-vol, categorie-place) $\subseteq$ Places(id-vol, categorie-place) ;

La table **Places** liste le nombre de places dans chaque vol pour chaque catégorie de place ("buisness", "economy", etc) , ainsi que leur prix (il s'agit du prix par place).

L'attribut *complet* de la table **Vol** est booléen ; il a la valeur **true** s'il ne reste plus de places disponibles (dans aucune catégorie) dans le vol. (Remarquer qu'il s'agit d'une information redondante.)

Une ligne *(id-vol, id-pass, categorie-place)* de la table **Reservation** indique qu'une place de la catégorie *categorie-place* est réservée dans le vol *id-vol* pour le passager *id-pass*.

Écrire le ou les triggers nécessaires pour réaliser les actions suivantes (il faut écrire aussi bien les fonctions `PL/pgSQL` qui implémentent les triggers, que les commandes `create trigger`.).

À l'ajout d'une nouvelle ligne dans la table **Reservation**, on vérifie si le vol est complet, et si cela n'est pas le cas, on calcule le nombre de places déjà réservées dans le vol pour la catégorie demandée. (Remarquer qu'une ligne de la table **Reservation** correspond à une réservation d'une seule place)

- S'il y a une place disponible dans la catégorie demandée, la reservation est effectuée. Dans ce cas l'attribut *complet* est mis à jour si besoin.
- S'il n'y a pas de places disponibles dans la catégorie demandée, la reservation est annulée. Dans ce cas un message d'erreur est affiché.