Université Paris 7

Master 1 Informatique, Bases de données avancées.

14 janvier 2009

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

Le sujet comporte 4 pages.

Dépendances

Exercice 1 On considère la relation $R(A, B, C, D, E, F)$ sur laquelle sont définies les dépendances fonctionnelles suivantes :

$$A, B \rightarrow C$$
$$D \rightarrow C$$
$$D \rightarrow E$$
$$C, E \rightarrow F$$
$$E \rightarrow A$$

Question 1: Compléter la table ci-dessus en supposant qu'elle satisfait les dépendances ci-dessus.

A	B	C	D	E	F
	1		110		54
x	2	j	100	n	52
w	1	i	110	m	
	2		100		

Question 2: Calculer la fermeture des ensembles d'attributs : $(D)^+$ et $(CE)^+$.

Question 3: Donner une clé candidate de la relation R.
Est-ce que il y a d'autres clés candidates que celle que vous avez donnée ? Justifier (sans faire le calcul).

Exercice 2 Nous avons une relation $R(A, B, C, D)$ avec les dépendances :

$$A, B \rightarrow C,$$
$$A, B \rightarrow D$$
$$D \rightarrow B.$$

Question 1: Est-ce qu'elle est 3NF ? Si elle ne l'est pas 3NF alors décomposer pour avoir les relations 3NF.

Question 2: Est-ce qu'elle est BCNF ? Si elle n'est pas décomposer pour avoir les relations BCNF.

Question 3: Si vous avez effectué une décompositions soit dans la première soit dans la deuxième question, est-ce que cette décomposition préserve information (c'est-à-dire est-ce que par la jointure on pourra reconstituer la relation initiale) ? Justifier votre réponse (une phrase suffit).
Votre décomposition préserve-t-elle les dépendances fonctionnelles ?

Modélisation

Exercice 3 Vous avez décidé de mettre un peu d'ordre dans vos bibliothèque de fichiers musicaux. Les fichiers sont stockés sur différents supports : disque de l'ordinateur portable, disque de l'ordinateur bureaux, disque externe, disque de la freebox etc, sur chaque disque dans différents répertoires.
Par la suite on distingue
- les fichiers dans le sens informatique,
- répertoires – désignent l'emplacement, le chemin d'accès, (on peut inclure le support dans le répertoire, comme une sorte d'attribut par exemple),
- le morceaux désigne un oeuvre musical.

On sait que :
Le même fichier peut être stocké dans plusieurs endroit (on peut avoir plusieurs copies du même fichier).
Le même morceau de musique peut-être stocké dans différents formats : mp3, flac, wma etc. ce qui donne bien sûr les fichiers différents chaque fois (mais le même morceau = le même oeuvre).
Pour chaque morceaux de musique les informations pertinentes minimales sont : album, titre, artiste, genre. On aimerais associer les notes aux morceaux. Si la place de disque nous manque cela permettra plus facilement effacer les morceaux sans grand intérêt pour nous (les morceaux que nous avons mal notés).
Faites une modélisation à l'aide de UML. N'oubliez pas les cardinalités.
Quels tables on obtient à partir de votre modèle ? (Si vous avez trop de tables il suffit de donner des exemples pertinents et indiquer au moins les clés et les clés étrangères.) Si à l'intérieur d'une table vous avez une dépendance fonctionnelle non triviale et dont la partie gauche n'est pas une clé indiquez la.
Remarque : C'est un exercice qui apporte des points s'il est bien fait. Par contre si mal fait cela ne rapporte pas grand chose. Et pour bien le faire il faut du temps. Peut-être à faire à la fin si le temps reste ?
En tout cas il vaut mieux bien modéliser sans faire de tables que mal modéliser et faire mal les tables par la suite (moins mais bien fait c'est mieux que plus mais mal fait).

Triggers - déclencheurs

Exercice 4

Un courtier gère les actions de ces clients. Parmi toutes les tables stockées dans sa base de données il y en a trois qui nous intéressent :

```
COURS(action_id, valeur, quand )
POSSEDE(client_id, action_id, quantite, min)
MISE_EN_VENTE(client_id, action_id, quantite, date )
```

`action_id` – l'identifiant de l'action de type INT,
`client_id` – l'identifiant client de type INT,
`valeur` – la valeur de l'action `action_id` de type `DECIMAL(10,2)` au moment `quand`
`quand` – de type `timestamp`,
`quantite` – le nombre d'actions de type `INT`,
`min` – la valeur minimale d'action de type `DECIMAL(10,2)`,

Dans `COURS` on stocke les valeurs des actions. Les anciens enregistrements ne sont jamais supprimés, si une action dont l'identifiant est `action_id` change la valeurs alors on ajoute

dans COURS une ligne avec la nouvelle valeurs et le moment (l'attribut quand) où le changement de cours avait lieu.
POSSEDE stocke les informations sur les actions possédées par les clients. Les attributs client_id et action_id donnent l'identifiant du client et l'identifiant des actions qu'il possède. Il y a autant lignes par clients que des actions différentes que le client possède. quantite donne le nombre d'actions (INT). La valeur de l'attribut min est spécifiée par le client. C'est un attribut de type DECIMAL(10,2). Si cette valeur est non NULL alors cela veut dire que le client demande que ses actions action_id soient automatiquement mises à la vente que si le cours d'action tombe au-dessous de min. (Le client veut arrêter les pertes si l'acion part à la baisse.)
L'opération de la mise à la vente implique les opérations suivantes :
- ajouter une ligne dans la table MISE_EN_VENTE avec les renseignements appropriés. L'attribut date de type timestamp donne le moment de la mise à la vente (le timestamp pour le moment "maintenant" est donné par la fonction CURRENT_TIMESTAMP), quantite donne le nombre d'action mises à la vente.
- si on a mis a actions à la vente alors on soustrait a de la valeur de l'attribut quantite de la table POSSEDE. En particulier si toutes les actions sont mises à la vente la valeur de l'attribut quantite dans POSSEDE devient 0.

Il faut écrire un déclencheur (trigger) qui sera associé à COURS et qui implémente la vente automatique demandée par les clients pour les actions qui perdent la valeurs.
Il faut donner une implémentation complète avec CREATE TRIGGER et avec une fonction pgsql correspondante.
Donc le trigger est déclenché par un ajout de ligne dans COURS mais il modifie, si besoin, les tables POSSEDE et MISE_EN_VENTE.

Transactions

Exercice 5 On suppose que dans cet exercice on travaille avec les mêmes tables que dans l'exercice précédent.
Vous êtes un directeur de banque et un client très important (client_id=3) vous appelle pour vous demander de vendre immédiatement 100000 actions de Google (action_id=1005). Il attend au téléphone, vous devez le faire tout de suite.
Alors vous vérifiez dans POSSEDE s'il a ces actions, vous modifiez dans POSSEDE la ligne appropriée et vous ajoutez une ligne (avec INSERT) dans MISE_EN_VENTE.
Pour éviter des problèmes (peut-être la femme de votre client demande la même chose à un de vos collaborateur et vendre deux fois les mêmes actions n'est pas une bonne idée ou peut-être vous craignez que le système tombe en panne après la modification d'une table mais avant la modification de l'autre) vous allez utiliser une transaction.
Écrire pas à pas les commandes SQL que vous allez exécuter. Il faut être très précis en ce qui concerne les commandes, écrire juste "je commence la transaction" et "je termine la transaction" ne donne pas de points.

Et autres

Exercice 6 Vous êtes l'administrateur de BD et vous devez donner le privilège d'effectuer SELECT et INSERT sur la table ACTIONS à un dénommé Toto. De plus vous autorisez Toto à donner les mêmes privilèges à qui il veut. Écrire la commande SQL appropriée.

Exercice 7 On reprend la table COURS de l'exercice 4.
On constate qu'on effectue de nombreuses requêtes SELECT sur cette table avec le critère de sélection de la forme

`WHERE quand > DATE '2005-05-01' AND quand < DATE '2006-01-01'`
Les dates sont données juste à titre d'exemples pour indiquer que les requêtes SELECT les plus utilisées demandent des intervalles de temps.
Ces requêtes sont très longues à exécuter vu la taille de la table.
Pour améliorer la vitesse de l'exécution de ces requêtes on construit un index sur l'attribut quand. Quel type d'index on préférera (hachage, B-arbres, pas d'importance) ? Justifier (très brièvement).

Et pour le requêtes avec le critère
`WHERE action_id=5`
c'est-à-dire égalité sur l'attribut `action_id`, quel type d'index est approprié (hachage, B-arbres, pas d'importance) ? Justifier (très brièvement).
Est-ce que ces indexes aident pour une requêtes de type :

```
WHERE (action_id=5) AND
      (quand > DATE '2005-05-01') AND (quand < DATE '2006-01-01')
```

Sinon quel index vous proposez pour ce type de requêtes ?

Exercice 8 Expliquer les termes "fragmentation horizontale, verticale, mixte" dans le contexte de BD distribuées. (Réponse courte, avec un petit exemple suffit.)
Expliquer le terme "réplication de données". A quoi sert la réplication de donnée dans le BD distribuées ?