Université de Versailles Saint-Quentin

M1 Informatique

BASES DE DONNEES – P. Pucheral — 12 janvier 2012

Durée : 2h. Tous (vos) documents autorisés.

Soyez précis dans vos réponses, toute réponse non justifiée sera considérée comme fausse.

Exercice 1 : SQL

La société Ama-Zone met en place une bourse d'échange permettant à tous ses clients d'échanger entre eux les cadeaux de Noël qui ne les ont pas satisfait. Cette bourse d'échange est gérée par la base de données présentée ci-dessous. L'échange est non lucratif (il s'agit de troc), c'est-à-dire qu'un client peut échanger un produit par un autre proposé dans la bourse d'échange même si les prix diffèrent (ces derniers sont donc publiés à titre purement indicatif). Dans la table `CLIENTS`, l'attribut `Reputation` est une estimation de la fiabilité du client variant de 0 à 5 (étoiles). Dans la table `ECHANGE`, un tuple représente le fait qu'un client de numéro `NumCli` propose d'échanger le produit `NumProd1` contre le produit `NumProd2` et que cette proposition est valable pour une période de `Durée` jours. La durée proposée pour l'échange ne peut dépasser 15 jours.

```
CLIENTS (NumCli, NomCli, Mail, Tel, Reputation)
PRODUITS (NumProd, NomProd, Type, Couleur, PrixCatalogue)
ECHANGE(NumCli, NumProd1, NumProd2, Durée)
```

Question 1.1 : Ecrire la requête SQL de création de la table `ECHANGE` en précisant l'ensemble des contraintes d'intégrité nécessaires.

Question 1.2 : Ecrire la requête SQL donnant la liste des noms des produits désirés (par échange) par le client de nom 'Ducerf'.

Question 1.3 : Ecrire la requête SQL donnant la liste des noms des produits que le client de nom 'Ducerf' est prêt à échanger avec d'autres produits moins chers que ces derniers.

Question 1.4 : Ecrire la requête SQL qui retrouve la valeur totale (i.e., cumul des Prix catalogue) de tous les produits proposés à l'échange groupés par Type, en ne considérant que les types pour lesquels il existe plus de 5 échanges proposés.

Question 1.5 : Ecrire la requête SQL qui met à zéro la réputation des clients qui ne proposent que des échanges dans lesquels leurs produits valent moins chers que ceux avec lesquels ils sont prêts à faire l'échange.

Exercice 2 : Indexation

La table `PRODUITS` est hachée sur `Type` (5000 valeurs possibles) et `Couleur` (256 valeurs possibles) par hachage multi-attributs extensible. Le numéro de paquet de hachage est composé de p1 bits résultant de l'application d'une fonction de hachage sur `Type` et de p2 bits sur `Couleur`.

Question 2.1 : Dessiner l'état du fichier contenant la table dans l'hypothèse où le nombre initial de paquets était fixé à 16 et où le paquet numéro 2 a éclaté une fois et le paquet 5 a éclaté deux fois. Ne représenter qu'un fragment du fichier, à savoir les pages disque correspondant aux paquets 0, 1, 2, 5 et 15 après les éclatements ainsi que les entrées associées dans le catalogue.

Question 2.2 : Supposons que les bits soient rangés dans l'ordre suivant: les p1 bits de l'attribut `Type` poids fort et les p2 bits de `Couleur` en poids faible. En considérant l'état du fichier tel que présenté dans la question 2.1, combien d'E/S faut-il pour retrouver tous les tuples satisfaisant le prédicat (`Couleur` = 'rouge') ? Même question avec le prédicat (`Type` = 32). Combien faut-il donc d'éclatements successifs d'un même paquet pour que le bénéfice du hachage sur `Type` se fasse sentir ?

Question 2.3 : Si l'on avait intercalé les p1 bits de `Type` et les p2 bits de `Couleur` (1 bit de `Couleur` puis un de `Type` et ainsi de suite), combien d'entrées de catalogue aurait-on sélectionnées pour la recherche (`Type` = 32). Même question pour ((`Couleur` = 'rouge') and (`Type` = 32)).

Exercice 3 : Vrai ou faux ?

Question 3.1 : Les affirmations suivantes sont-elles vraies ou fausses ? Toute réponse non justifiée sera considérée comme fausse.

1. Dans un B-Tree d'ordre m, chaque nœud (hormis la racine) contient entre m et 2m clés. (a) cette règle à pour objectif d'assurer un taux de remplissable d'au moins 50% de chaque nœud, (b) cette règle est indispensable pour que l'arbre soit équilibré.
2. Pour réaliser un index non plaçant, un index bitmap est toujours plus volumineux qu'un B-Tree.
3. Un algorithme de sérialisation par estampillage abandonne plus de transactions qu'un algorithme par verrouillage.
4. Une transaction s'exécutant avec un degré d'isolation 'Repeatable Read' pose plus de verrous que si elle s'exécutait en mode 'Read Committed'.
5. La table T contient 50.000 tuples répartis dans 1000 pages. Si la sélectivité d'un prédicat est inférieure à 5%, j'ai tout intérêt à utiliser un index secondaire.

Barème indicatif :

Exercice 1 : 10 points (2 points par question)
Exercice 2 : 6 points (2 points par question)
Exercice 3 : 5 points (1 point par sous-question)
(donc, 1 point de bonus)