M1 Informatique

Bases de données avancées

Examen de session 2 (durée 2h)

28 juin 2018

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

Évaluation de requêtes

Question 1 Supposer qu'une jointure $R \bowtie S$ est calculée par *nested-loop join*. Supposer que R occupe 200 blocs et S occupe 500 blocs. Il y a 22 blocs en mémoire centrale qui peuvent être utilisés pour calculer cette jointure, dont 1 sera utilisé pour écrire le résultat un bloc à la fois. Au plus 10 lignes de R (de S) rentrent dans un bloc.

Le coût de cette jointure dépend de la table choisie pour la boucle externe et du nombre de blocs utilisés en mémoire centrale. Calculer le coût optimal (i.e. le plus petit nombre d'accès à disque) pour cette jointure, et dire quels choix permettent de l'obtenir.

Question 2 La table R a 100 000 lignes, dont 100 rentrent dans un bloc. La table S a 20 000 lignes, dont 200 rentrent dans un bloc. Il y a 11 blocs disponibles dans le buffer en mémoire centrale. Supposer qu'il ait au plus 30 lignes de S qui peuvent être en jointure avec une même ligne de R. Calculer le nombre d'accès à disque nécessaires pour effectuer un *merge join* entre R et S. Inclure le coût du tri des deux tables ainsi que le coût de la matérialisation des tables triées ; ne pas inclure le coût de l'écriture du résultat de la jointure.

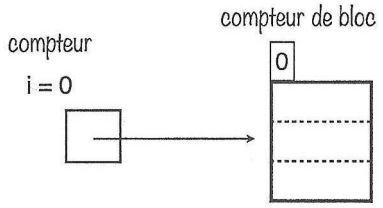
Indexation

Question 3 Nous considérons ici un index hash sur une clef de type entier. Faire l'hypothèse (irréaliste) qu'un bloc du disque puisse stocker jusqu'à trois entrées de l'index (i.e. trois couples <clef, pointeur à enregistrement>). La fonction de hachage est $h(x) = x \bmod 8$.

Supposer que les clefs suivantes sont insérées dans l'index hash initialement vide, dans l'ordre :
2, 3, 5, 7, 11, 12, 14, 16.

Montrer l'état de l'index hash après chaque insertion. En particulier montrer, après chaque insertion : les blocs utilisés par l'index, leur contenu, la structure des pointeurs et la valeur de tous les compteurs (compteur global et compteurs de bloc).

Se rappeler que dans un index hash vide il y a un seul pointeur à un bloc du disque vide, et tous les compteurs ont valeur 0, comme le montre la figure ci-dessous.



Formes normales

Question 4 Soit $R(A, B, C, D, E, F, G)$ un schéma de relation avec dépendances fonctionnelles :

$\mathcal{F} = \{$
$BAE \rightarrow F$
$BCA \rightarrow F$
$E \rightarrow G$
$C \rightarrow AG$
$B \rightarrow D$
$FD \rightarrow B$
$CD \rightarrow F$
$AED \rightarrow B$
$AE \rightarrow C\}$

- 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.)

Triggers

Question 5 On considère le schéma de base de données suivant pour la gestion d'un ensemble de lignes de bus :

Arret (id, nom, rue, numero)
Ligne (numero, arret-dep, arret-arr)
Plan (num-ligne, id-arret, ordre)

Les clefs primaires sont soulignées. Les clefs étrangères sont les suivantes :
Ligne(arret-dep) $\subseteq$ Arret(id) ;
Ligne(arret-arr) $\subseteq$ Arret(id) ;
Plan(num-ligne) $\subseteq$ Ligne(numero) ;
Plan(id-arret) $\subseteq$ Arret(id) ;

La table Ligne contient toutes les lignes de bus, et la table Arret liste tous leurs arrêts. La table Plan indique quelle ligne de bus s'arrête à quel arrêt, et dans quel ordre ; en particulier ordre est un attribut entier avec valeur 0 pour l'arrêt de départ, 1 pour le premier arrêt, 2 pour le deuxième etc, jusqu'à n pour l'arrêt d'arrivée, où n est le nombre d'arrêts.

Écrire les triggers nécessaires pour garantir le comportement suivant de la base de données (il faut écrire à la fois les commandes de création des trigger et les fonctions qui implémentent leur actions) :

- À chaque insertion d'une nouvelle ligne de bus, les attributs specifiant l'arrêt de départ et l'arrêt d'arrivée de cette ligne prennent la valeur null avant l'insertion. (Des éventuelles valeur existantes pour ces attributs sont écrasées.)
- Toute tentative de mise à jour du numéro de ligne dans la table Plan sera fait échouer.
- À chaque mise à jour d'un plan de ligne, les arrêts de départ et d'arrivée de la ligne en question sont recalculés et mis à jour dans la table Ligne. Remarquer qu'un plan de ligne peut être mis à jour par modification des attributs id-arret et ordre de la table Plan, mais également par l'insertion ou suppression d'un tuple dans cette table.