M1 Informatique et Mathématiques-Informatique

Bases de données avancées

Examen de session 1 (durée 2h)

7 mai 2021

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

Indexation

Question 1 Ci-dessous est donné le psuedo-code (vu en cours) de l'algorithme de recherche d'une clef c dans un arbre B^+ sans doublons. On considère qu'à chaque itération $K_1, ..K_n$ dénotent les clefs stockées dans le noeud courant l'arbre, dans l'ordre. On suppose aussi l'existence de clefs fictives $K_0 = -\infty$, $K_{n+1} = +\infty$ sur chaque noeud. De cette facon l'enfant i d'un noeud de l'arbre se trouve toujours entre la clef K_{i-1} et la clef K_i.

```
B := le noeud racine;
while le noeud courant B n'est pas une feuille do
    trouver l'enfant i de B tel que K_{i-1} =< c < K_i;
    B := i-eme enfant de B;
end
chercher c dans B et la retourner
```

a. Proposer une modification du code ci-dessus pour chercher toutes les clefs supérieures ou égales à c;

b. Supposer maintenant que les clefs de recherche sont de la forme (a, b). L'ordre parmi les clefs est lexicographique, c'est à dire $(a, b) < (a', b')$ si et seulement si

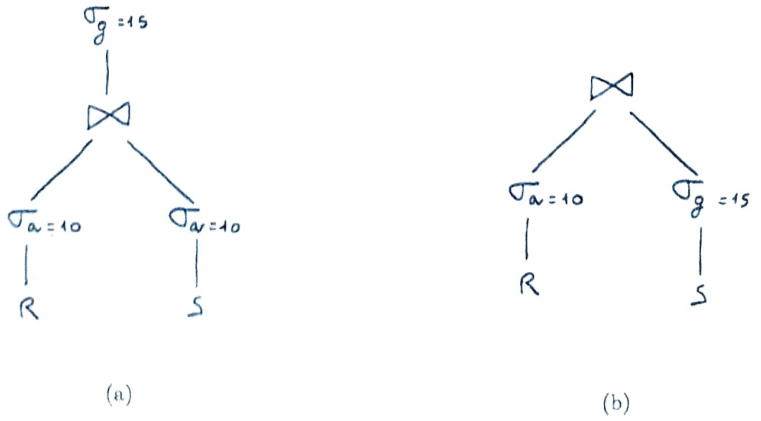
   1. $a < a'$ ou
   2. $a = a'$ et $b < b'$

   Proposer un algorithme pour chercher, donné c, la plus petite clef (a, b) de l'index telle que $a = c$ (sachant qu'une telle clef n'est pas nécessairement présente dans l'index).

Évaluation de requêtes

Question 2 Considérer les deux plans logiques représentés dans la Figure 1, définis sur le schéma $R(a, b), S(a, g, e)$. Supposer les statistiques suivantes :

- taille de $R : r$, taille de $S : s$.
- Il y a 100 valeurs distinctes de a dans R, comme dans S, et chaque valeur apparaît avec la même probabilité.
- Il y a 8 valeurs distinctes de g dans S, chaque valeur de g apparaît dans un même tuple avec chaque valeur de a (donc il y a 800 valeur distinctes de (a, g) dans S). De plus chaque valeur de (a, g) apparaît dans S avec la même probabilité.



(a) (b)

FIGURE 1 – Figure pour la Question 2

Répondre aux questions suivantes :

1. Les deux plans sont ils équivalents ?
2. Estimer le coût de chacun des deux plans en fonction de r et s. La mesure de coût sera la somme des tailles des résultats intermédiaires, y compris la taille du résultat final.
3. Sur la base de cette notion de coût, quel est le meilleur plan si $r = s = 1000$? Et si $r = s = 2000$?

Question 3 Ecrire un algorithme (pseudo-code) qui implémente un itérateur pour le *merge join* entre `R` et `S` sur la condition de jointure `R.A=S.A`. On supposera que `A` soit une clef pour R (mais pas nécessairement pour S). (Remarquer que cette assomption permet, pendant la fusion de R et S, de ne pas devoir bufferiser les tuples de R ayant la même valeur de A.)

Le code doit définir les fonctions standard `open()`, `next()` et `close()` de l'itérateur, ainsi que spécifier quel est son état interne. Se rappeler que `open()` initialise et `close()` finalise l'état interne de l'itérateur, alors que `next()` renvoie le prochain tuple du résultat de la jointure.

On supposera que R et S sont déjà triées sur l'attribut A et qu'un *linear scan* de `R` et un *linear scan* de `S` sont disponibles également en tant qu'itérateurs. C'est-à-dire, on supposera que `R.open()` et `R.close()` s'occupent d'ouvrir et fermer le parcours séquentiel de `R`, et `R.next()` renvoie le prochain tuple de `R`. Les fonctions `S.open()`, `S.close()`, `S.next()` font la même chose sur `S`. Toutes les opérations de `next()` renvoient `null` s'il ne reste plus de tuples à retourner. (Notez qu'il n'est pas demandé d'écrire ces fonctions pour le *linear scan* de `R` et `S`, vous pouvez les supposer données.)

Formes normales

Question 4 Soit $R(A, B, C, D, E, F, G)$ un schéma de relation avec dépendances fonctionnelles :
$\mathcal{F} = \{B \rightarrow CD, \quad F \rightarrow G, \quad A \rightarrow D, \quad CA \rightarrow B, \quad FCA \rightarrow E, \quad EG \rightarrow F, \quad CAG \rightarrow F, \quad FBC \rightarrow E, \quad BG \rightarrow E\}$

- 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 ?
- Donner toutes les clefs de R, puis 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.)

Transactions

Question 5 Considérer 3 transactions concurrentes $T1$, $T2$ et $T3$ définies comme suit :
$T1 : r1(X)\ r1(Y)\ r1(Z)\ w1(Z)$
$T2 : r2(Y)\ w2(X)\ w2(Z)$
$T3 : r3(Z)\ r3(Y)\ w3(Y)$

Supposer que leur ordonnancement se fait par *estampillage* et que les estampilles des trois transactions sont $E(T1) = 70$, $E(T2) = 80$ et $E(T3) = 60$. Supposer que T1, T2 et T3 sont les seules transactions actives actuellement mais que d'autres transactions ont opéré auparavant sur les données X, Y et Z. Par ses autres transactions terminées, les estampilles en lecture (E_R) et en écriture (E_W) des données, avant que $T1$, $T2$ et $T3$ ne commencent, sont les suivantes :
$E_R(X) = 40, \quad E_W(X) = 65$
$E_R(Y) = 55, \quad E_W(Y) = 50$
$E_R(Z) = 55, \quad E_W(Z) = 55$

L'ordonnancement partiel exécuté jusqu'à maintenant est : $S = r1(X), r3(Z), r3(Y)$

1. Quelles sont les estampilles des données, en écriture et en lecture, juste après ces trois opérations ?
2. Pour chacune des deux sequences suivantes dire si elle peut être la suite de l'ordonnancement courant S sous estampillage, sans entraîner de *rollback*. Si ce n'est pas le cas, indiquer quelles sont les opérations qui posent problème.

   (a) $w3(Y), r2(Y), r1(Y), w2(X), w2(Z), r1(Z), w1(Z)$
   (b) $r1(Y), w3(Y), r2(Y), w2(X), r1(Z), w1(Z), w2(Z)$