Nom :	Prénom :	*page 1*

UPMC - UFR 919 - Master d'informatique - M1

Bases de Données Réparties. Examen du 29 mai 2012

Version CORRIGEE remarques intégrées

Documents de cours, TD et TME autorisés – Durée : 2h.Téléphones éteints et rangés dans les cartables.

Répondre aux questions sur la feuille du sujet dans les cadres appropriés. La taille des cadres suggère celle de la réponse attendue. Utiliser le dos de la feuille précédente si la réponse déborde du cadre. Le barème est donné à titre indicatif. Ecrire à l'encre bleue ou noire..Les réponses non-justifiées **ne seront pas** prises en compte.

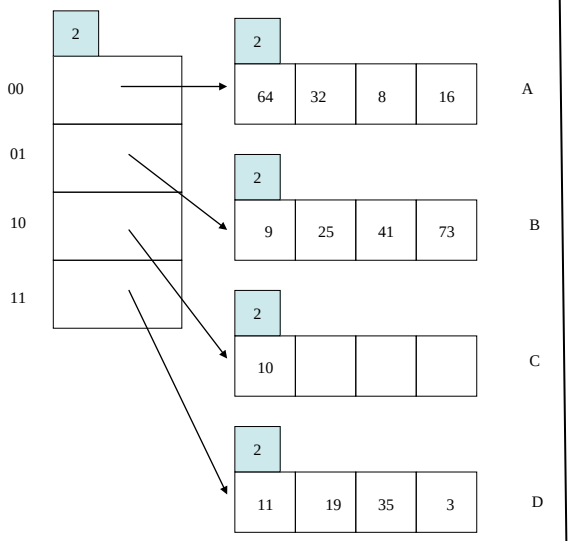
Exercice 1 : Hachage extensible — 5 points

On considère un index utilisant une technique de hachage extensible. La profondeur globale est indiquée au dessus du répertoire. La profondeur locale est indiquée au dessus de chaque paquet. Un paquet peut contenir de 1 à 4 entrées. Les entrées correspondent à des valeurs de clé hachées.

On utilise un algorithme de suppression complète, c'est-à-dire que la suppression d'un paquet entraîne la diminution de la taille du répertoire (division par 2), si cela est possible. On suppose qu'il y a fusion seulement si et seulement si un paquet est vide. Toute insertion dans un paquet plein provoque un éclatement et donc la création d'un nouveau paquet.

Question 1. Soit l'index suivant dans l'état initial *E1*.

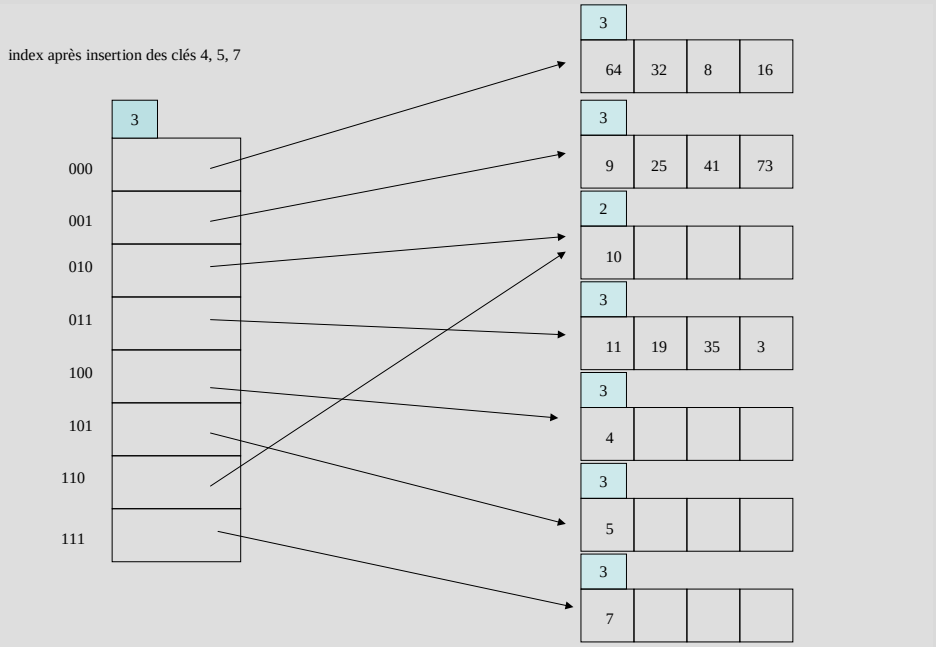
`Index initial (E1):`



`Index après insertion (E2):`

Représenter l'état *E2* de l'index après insertion successive des entrées 4, 5, 7.

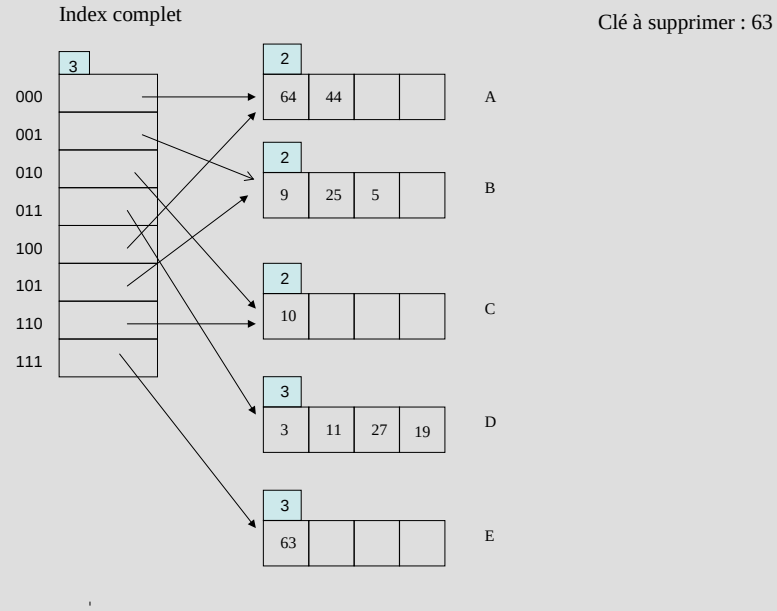
Réponse :



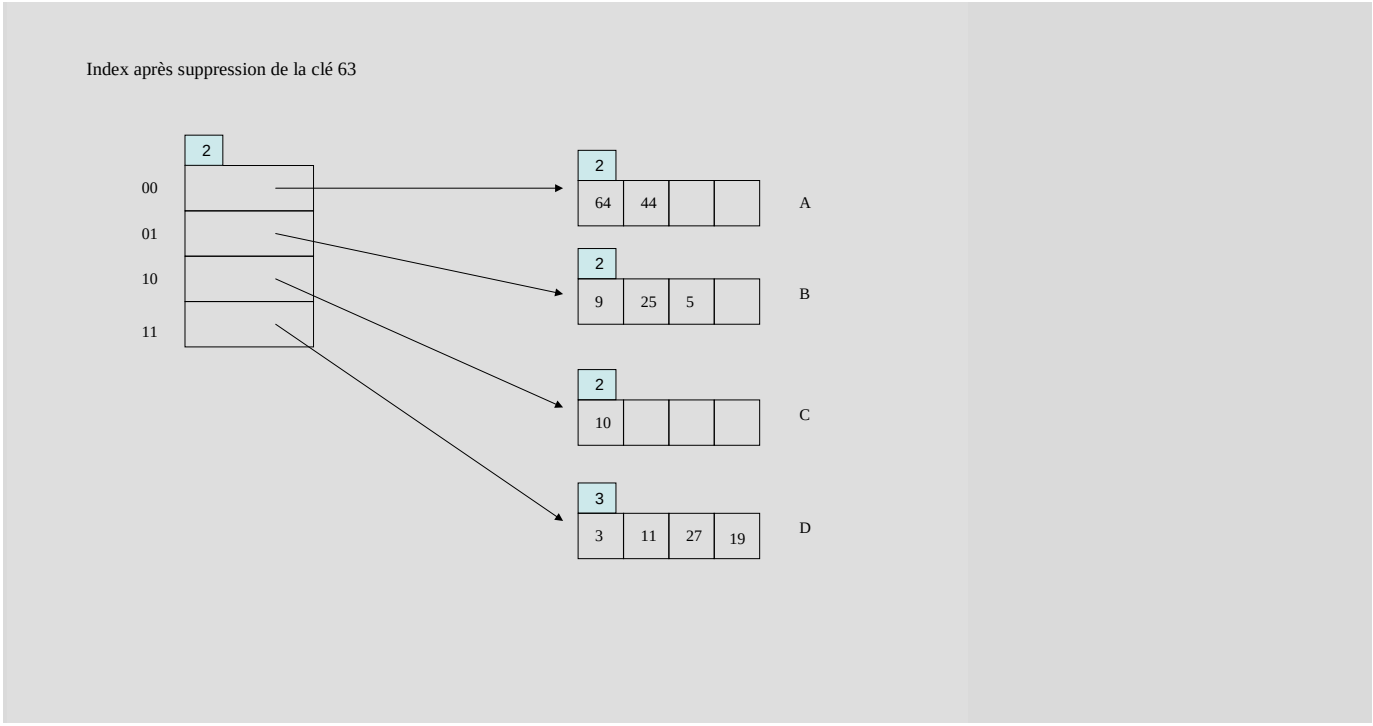
Question 2. On considère l'état suivant d'un index utilisant une technique de hachage extensible.

a) Complétez l'état initial de cet index de telle façon que celui-ci contiennent les entrées (en plus de celle déjà apparentes sur la figure ci-dessous) : 10, 3, 11, 19, 27, 63. On suppose qu'il n'y a eu aucune suppression dans l'index.

Réponse :



b) Donnez l'état de l'index après suppression de l'entrée 63.



Question 3 (bonus)

Reprendre les questions suivantes en illustrant avec un exemple

Soit un index de hachage extensible et T la taille du répertoire de l'index. Quel est le nombre maximum NmaxP et le nombre minimum NminP de paquets de l'index ? Quel est le nombre maximum NmaxE et le nombre minimum NminE d'entrées de l'index ? On suppose qu'il n'y a eu que des insertions dans l'index. Justifiez les valeurs min, en indiquant la suite de valeurs insérées pour arriver au résultat. Indice : une insertion d'une entrée peut provoquer plusieurs éclatements si la répartition est particulièrement non uniforme.

MaxP= T, maxE=4*T, minP = log2(T)+1. Pour y arriver, on insère 5 clés (minE=5). Les 4 premières ont une valeur de hachage se terminant avec log2(T) zéros, la dernière se terminant par 1 un suivi de log2(T) zéros. L'insertion de la dernière clé provoque éclatement et doublement de répertoire jusqu'àa ce que la dernière clé se retrouve dans un paquet distinct des 4 premières, soit lorsque le répertoire atteint une taille de T. Cela se produit donc log2(T) fois, on a donc log2(T)+1 paquets à la fin, donc 1 est plein, 1 contient une entré, les autres sont vides.

Exercice 2 : Transactions réparties 5 points

Soit une base de données répartie telle que les granules A et B sont sur le site S1, les granules C, D, E sont sur le site S2. Soit l'exécution suivante des transactions T1, T2, T3, T4 (Li(g) signifie lecture du granule g par la transaction Ti, Ei(g) signifie écriture par la transaction Ti du granule g). On précise la date d_i à laquelle la i-ème opération est exécutée. On suppose que les transactions demandent leur validation dès qu'elles ont fini leur dernière opération.

L2(A) , L3(C) , L1(A) , E2(A) , L3(A) , L3(D) , L2(D) , E4(D) , L4(C) , L1(D) , L2(B) , E3(B)

d1 d2 d3 d4 d5 d6 d7 d8 d9 d10 d11 d12

On suppose un contrôle de concurrence optimiste où le graphe de précédence global de l'exécution est construit sur S1, progressivement, à chaque nouvelle précédence générée sur l'un ou l'autre des sites. Les messages sont de la forme (Ti, Tj, g) si Ti précède Tj sur le granule g.

Question 1
Quelle est la première transaction qui demande sa validation ? A quelle date cela se produit ?
Quelle est la dernière opération exécutée à cette date ?

Transaction T4. à la date d9

Dernière opération exécutée :L4(C)

Question 2
Représenter ce graphe au moment où la première transaction demandera sa validation. Cette validation sera-t-elle acceptée ? Justifiez votre réponse. Cette décision aurait-elle pu être prise en ne considérant que les graphes de précédences locaux ? Quels messages ont-été envoyés par S2 jusqu'à ce moment ?

Au moment de la première demande de validation

Graphe de précédence sur S1 (indiquer sur chaque arc le(s) granule(s) correspondants(s))

T1 –A-> T2 –A-> T3

Graphe de précédence sur S2 (indiquer sur chaque arc le(s) granule(s) correspondants(s))

T3 –D-> T4
T2 –D-> T4

Graphe de précédence global (indiquer sur chaque arc le(s) granule(s) correspondants(s))

T1 –A-> T2 –A-> T3–D-> T4
Validation acceptée (oui/non) ? Non
Justification : il faut attendre que les transactions qui précèdent T4 aient validé

Décision possible en ne considérant que les graphes de précédence locaux : non
Justification : sérialisation locale n'implique pas sérialisation globale

Messages envoyés de S2 à S1 (dans l'ordre d'émission):

(T3, T4, D) , (T2, T4, D)

Question 3

Représenter le graphe de précédence global à la fin de l'exécution. Celle-ci est elle globalement sérialisable ? Justifier.

A la fin de l'exécution

Graphe de précédence global :

T1 –A-> T2 –AB-> T3–D-> T4 –D-> T1

Exécution sérialisable (oui/non) ? non
Justification : circuit

Exercice 3 : Questions de cours 2 points

Répondre aux questions suivantes en argumentant votre réponse.

Question 1 :
Les systèmes pair à pair structurés (par une table de hachage distribuée) sont adaptés à un type de requête et ne le sont pas pour un autre type. Lesquels ? Justifiez votre réponse en une phrase.

Adapté :
Requête d'égalité

Non-adapté : requête d'intervalle

Question 2 :

Quel type de réplication (synchrone ou asynchrone) est le mieux adapté pour les entrepôts de données. Justifiez votre réponse.

Asynchrone. Pour ne pas bloquer les systèmes de production de données et aussi parce que les requêtes olap des entrepôts sont des statistiques qui la plupart du temps ne nécessitent pas une fraicheur parfaite des données

Exercice 4 : Bases de données parallèles 3 pts

Soit une relation R(A,B) tenant sur 2000 pages

R est initialement triée sur l'attribut A.

On décide d'utiliser une base de données parallèle. Celle-ci est formée de 10 nœuds (N_1 à N_{10}) ayant chacun une mémoire vive (RAM) de 100 pages. R est allouée aux nœuds selon la méthode du tourniquet (chaque nœud disposant d'un disque de taille suffisante).

Question 1

Montrer que le fragment de R alloué à chaque nœud est lui-même trié sur A.

Les nuplets sont envoyés dans l'ordre au différents sites, ils sont donc triés sur les sites

Question 2

Donner la stratégie pour évaluer la requête suivante, posée sur le nœud N_1 et évaluer son coût en entrée/sorties (nb de pages) et son coût en terme de pages de données transférées sur le réseau. Détailler les coûts étape par étape.

Select avg(A)*max(A) from R ;

Stratégie :

Attention, round robin ne garantit pas que tous les sites ont exactement le même nombre de tuples.

La stratégie est donc que chaque site fait une passe en lecture sur son fragment, pour calculer sum, max et count. Tous les sites envoie leurs résults à N1 (une page transférée par site suffit)

Enfin, N1 fait le calcul final

Coût en entrée/sorties (justifier) : 10 * 2000/10 = 2000 pages Lues

Coût en pages transférées (justifier) : 9 pages transférées (N2 à N10)

Exercice 5 : JDBC 5 pts

Nouvel exercice 2011

Soit un BD de tennis contenant des joueurs (J), des gains (G) et des rencontres (R). Son schéma global est

J (joueur, nom, nationalité) // l'attribut joueur est le numéro (nombre entier) du joueur.

G (joueur, lieutournoi, annee, prime, sponsor) // Il existe deux lieux de tournoi.

R (gagnant, perdant, lieutournoi, annee) // gagnant et perdant sont des numéros de joueur

Un joueur participe à au moins une rencontre. La base est répartie sur deux sites, selon de lieu de tournoi : S1 pour Rolland Garros et S2 pour Wimbledon.

Le site Si contient les relations Ji, Gi et Ri (avec i dans {1,2}).

Les relations Gi et Ri contiennent les n-uplets dont le lieu de tournoi est celui du site Si.

La relation Ji représente les joueurs qui ont participé au moins une fois à un tournoi du site Si. Rmq : certains joueurs peuvent être à la fois dans J1 et J2.

On suppose que seul le site S0 peut se connecter aux sites S1 et S2. Le site S0 se limite à exécuter une application java qui accède aux sites S1 et S2 par JDBC. Les programmes java, proposés par la suite, sont des extraits de programmes complets, suffisants pour comprendre ce que fait le programme. Ils s'exécutent sur S0.

Question 1

L'application A1 affiche le résultat de la requête globale : `select joueur from J`
Joueur étant une clé de J, le résultat ne contient donc aucun double.

1) Le programme P1 suivant est-il A1 ? Justifier.

```
1   s1 = DriverManager.getConnection(url1, u1, pw1); // site S1
2   s2 = DriverManager.getConnection(url1, u2, pw2); // site S2
3
4   Statement q1 = s1.createStatement();
5   Statement q2 = s2.createStatement();
6
7   ResultSet r1 = q1.executeQuery("select * from J1");
8   while (r1.next()) {
9         out.println(r1.getInt(1));
10  }
11  r1.close();
12  ResultSet r2 = q2.executeQuery("select * from J2");
13  while (r2.next()) {
14      out.println(r2.getInt(1));
15  }
16  r2.close();
17  q1.close(); q2.close();
18  s1.close(); s2.close();
```

Non, cela affiche en double les joueurs qui ont joué à la fois à Rolland Garros et Wimbledon

2) Le programme P2 suivant est-il A1 ? Justifier.

```
1   s1 = DriverManager.getConnection(url1, u1, pw1); // site S1
2   Statement  q1 =  s1.createStatement();
3   ResultSet  r1  =  q1.executeQuery("select  joueur  from  J1  union
4   select joueur from J2");
5
6   while (r1.next()) {
7         out.println(r1.getInt(1));
8   }
9   r1.close(); q1.close(); s1.close();
```

Non, requête erronée J2 n'existe pas dans le site S1

3) Le programme P3 suivant est-il A1 ? Justifier.

```
1   s1 = DriverManager.getConnection(url1, u1, pw1); // site S1
2   Statement  q1 =  s1.createStatement();
3   ResultSet r1 = q1.executeQuery("select joueur from J);
4   while (r1.next()) {
5         out.println(r1.getInt(1));
6   }
7   r1.close(); q1.close(); s1.close();
```

Non, requête erronée J n'existe pas (ni sur S1 ni sur S2)

4) Compléter les deux trous du programme P4 pour qu'il corresponde à A1.

```
1   s1 = DriverManager.getConnection(url1, u1, pw1); // site S1
2   s2 = DriverManager.getConnection(url1, u2, pw2); // site S2
3   Statement q1 = s1.createStatement();
4   Statement q2 = s2.createStatement();
5   ResultSet r1 = q1.executeQuery("select joueur from J1");
6   while (r1.next()) {
7         out.println(r1.getInt(1));
8   }
9   r1.close();
10  ResultSet r2 = q2.executeQuery("select joueur from J2");
11  while (r2.next()) {
12        r1 = q1.executeQuery("select joueur from J1 where  Trou1  );
13        if(  Trou2  ) {
14              out.println(r2.getInt(1));
15        }
16        r1.close();
17  }
18  r2.close();
19  q1.close(); q2.close();
20  s1.close(); s2.close();
```

Trou1 :

```
executeQuery("select joueur from J1 where joueur = " + r2.getInt(1))
```

Trou2 : `! r1.next()`

Question 2

L'application A2 affiche les numéros de joueurs qui ont gagné (au moins une fois) une prime strictement supérieure à 600 euros à Rolland Garros **et** aussi à Wimbledon.

Compléter les trous du programme ci-dessous pour qu'il corresponde à A2.

```
1   c1 = DriverManager.getConnection(url1, u1, pw1); // Site1
2   c2 = DriverManager.getConnection(url2, u2, pw2); // Site 2
3   Statement s1 = c1.createStatement();
4   ResultSet r1 = s1.executeQuery("select distinct joueur from G1
5   where prime > 600");
6
7   PreparedStatement s2 = c2.prepareStatement("select prime from
8   G2 where  Trou1   ");
9
10  while (r1.next()) {
11        s2.setInt( Trou2 );
12        ResultSet r2 = s2.Trou3;
13        if(r2.next()) {
14          out.println( Trou4 );
15        }
16        r2.close();
17  }
18  r1.close(); s1.close(); s2.close(); c1.close(); c2.close();
```

```
Trou1 : c2.prepareStatement("select joueur from G2 where joueur = ? and prime > 600")

Trou2 : setInt(1, r1.getInt(1));

Trou3 : s2.executeQuery();

Trou4 : out.println(r1.getInt(1));
```

Question 3

L'application A3 affiche la somme des primes pour les joueurs qui ont gagné (au moins une fois) une prime à Rolland Garros et aussi à Wimbledon. Sur chaque ligne, A3 affiche le numéro du joueur, suivi d'un espace, suivi de la somme de ses primes.

Compléter les trous du programme ci-dessous pour qu'il corresponde à A3.

```
1   s1 = DriverManager.getConnection(url1, u1, pw1); // Site 1
2   s2 = DriverManager.getConnection(url2, u2, pw2); // Site 2
3   Statement q1 = s1.createStatement();
4   Statement q2 = s2.createStatement();
5   ResultSet r1 = q1.executeQuery("select joueur, sum(prime) from
6   G1 Trou1 ");
7   ResultSet r2 = q2.executeQuery("select joueur, sum(prime) from
8   G2 Trou2 ");
9
10  boolean c1 = r1.next();   boolean c2 = r2.next();
11  while(c1 && c2) {
12        if(r1.getInt(1) < r2.getInt(1))
13            c1 = r1.next();
14        else if(r1.getInt(1) > r2.getInt(1))
15            c2= r2.next();
16        else {
17            out.println(Trou3);
18            Trou4
19        }
20  }
21  r1.close(); r2.close();
22  q1.close(); q2.close(); s1.close(); s2.close();
```

```
Trou1  :Group by joueur order by joueur

Trou2 : Group by joueur order by joueur

Trou3 : println(r1.getInt(1)+ " " + (r1.getInt(2) + r2.getInt(2)))

Trou4 : c1 = r1.next(); c2= r2.next();
```