Sorbonne Université

Master d'informatique- M1

UE 4IN803

SAM : Stockage et Accès aux Mégadonnées

Support de TD - Partie 1

année 2023

Hubert NAACKE

TABLE DES MATIERES

Arbres B+

Hachage

Optimisation de requêtes

TD 1 : Index , Arbre B+

Notations et conventions

Nombre de valeurs dans un nœuds. Pour un arbre B+ d'ordre d, le nombre de valeurs qu'un nœud peut contenir est :

dans l'intervalle [1, 2.d] pour la racine,

dans l'intervalle [d, 2.d] pour les nœuds intermédiaires et les feuilles.

Certains exercices n'indiquent pas l'ordre de l'arbre mais directement le nombre de valeurs dans un nœud.

Dessin d'un arbre. Pour repérer facilement les nœuds d'un arbre quand on le dessine, on peut attribuer un nom à chaque nœud : R pour la racine, Ni pour les nœuds intermédiaires et Fi pour les feuilles. On peut utiliser la syntaxe suivante pour représenter le contenu d'un nœud : N(v_1, v_2, …) où N est le nom du nœud et les v_i sont les valeurs.

Insertion d'une valeur en cas de débordement : quand la feuille *F* déborde, on garde les *d+1* plus petites valeurs dans *F*, les *d* autres valeurs vont dans une nouvelle feuille. Quand le nœud intermédiaire *N* déborde on garde les *d* plus petites valeurs dans *N*, les *d* plus grandes valeurs vont dans un nouveau nœud. La valeur restante est insérée dans le nœud père.

Suppression d'une valeur. Si le nœud ne contient que *d* valeurs avant la suppression, alors on considère d'abord la redistribution avec le nœud voisin (de même père) situé à gauche, puis avec celui situé à droite. Si aucune redistribution n'est possible, on considère la fusion avec le voisin (de même père) situé à gauche, puis avec celui situé à droite. Une suppression avec redistribution nécessite d'ajuster le contenu du nœud père. Une suppression avec fusion nécessite de supprimer une valeur dans le nœud père.

Décompte du nombre de **nœuds** lus et écrits. Une opération d'insertion ou de suppression peut nécessiter de lire, modifier ou créer des nœuds. On appelle *L* le nombre de nœuds lus pendant une opération, et respectivement *E* le nombre de nœuds écrits ou créés.

Exercice 1 (ref 17-2) : Index composés — pts

On considère la table : **Livre** (titre, num, auteur, résumé, taille, prix, année)

On a les index **I1** sur Livre (année, auteur) **I2** sur Livre (prix, année) **I3** sur Livre(auteur, année, prix)

Quels index peuvent être utilisés pour traiter les requêtes suivantes ? Pour chaque index entourer oui/non, si oui alors préciser quel prédicat sert d'accès (*access*) à l'index : soit toute la clause where, soit **une partie** de la clause where, la partie restante étant évaluée ultérieurement par un *filter*. Justifier très brièvement une réponse *non*.

1) Select * from Livre where année=2016;
2) Select * from Livre where année>2010 and auteur = 'Hugo';
3) Select * from Livre where année>2000 and prix > 50;
4) Select * from Livre where année=2016 and prix=20 and auteur like 'T%';

Exercice 2 (ref 16-1) : Index composé et index couvrant une requête

Soit la relation **Joueur** (nujoueur, nom, prénom, âge, ville, sport)

Tous les index sont non-plaçants. Il y a 3 index :

I1 sur Joueur(ville) **I2** sur Joueur**(**âge, sport) **I3** sur Joueur(sport, âge)

Soit les requêtes :

R1 : select sport from Joueur where âge >20

R2 : select max(age) from Joueur where sport = 'vélo'

R3 : select distinct ville from Joueur

R4 : select avg(age) from Joueur where ville='Paris' and sport='vélo'

R5 : select prénom from Joueur where prénom like 'Ted%' age = 18 and sport = 'judo' order by prénom

a) Est-ce que I2 est utilisable pour R1 ? Entourer `oui` `non.` Justifier.

b) Est-ce que I3 est utilisable pour R1 ? Entourer `oui` `non.` Justifier.

c) Est-ce que I2 est utilisable pour R2 ? Entourer `oui` `non.` Justifier.

d) Est-ce que I3 est utilisable pour R2 ? Entourer `oui` `non.` Justifier.

e) Est-ce que I1 couvre R3 ? Entourer `oui` `non.` Justifier.

f) Peut-on couvrir R4 avec un (ou plusieurs) index parmi ceux existants ? Si oui lesquels ?

g) On sait que R5 peut être évaluée sans accéder à la table Joueur. Expliquer comment évaluer R5 en utilisant I2(âge, sport) et le nouvel index I4(prénom) sans lire aucun nuplet de Joueur.

Exercice 3 (ref 17-1) : Plan et coût d'une requête — pts

On considère la base d'une librairie. **Livre** (titre, num, auteur, résumé, taille, prix, année)

Il y a 100 années de 1917 à 2016 incluses.

Il y a 10 prix (10, 20, ..., 90, 100).

Il y a 200 auteurs.

Les index (non plaçants) sont : Livre(titre), Livre(auteur), Livre (prix) et Livre(année).

Le nom d'un index est I_L_*nom_attribut.*

On pose la requête **R1** : `Select * from Livre where prix <=20 ;`

Le plan est

```
|  Id | Operation             | Name      | Rows      | Bytes     | Cost (%CPU)
|   0 | SELECT STATEMENT      |           | 200       | 127K      |    68   (0)
|*  1 |  TABLE ACCESS FULL    | LIVRE     | 200       | 127K      |    68   (0)
Predicate Information:
    1 - filter(PRIX<=20)
```

On pose la requête **R2** affichant les livre de 2015 et 2016 : `Select * from Livre where annee >=2015 ;`

Le plan est :

```
|  Id | Operation                     | Name     | Rows  | Bytes   | Cost (%CPU)
|   0 | SELECT STATEMENT              |          |    20 | 15694   |  22   (0)
|   1 |  TABLE ACCESS BY INDEX ROWID| LIVRE    |    20 | 15694   |  22   (0)
|*  2 |   INDEX RANGE SCAN            | I_L_ANNEE|    20 |         |   2   (0)
Predicate Information:
    2 - access(ANNEE>=2015)
```

Répondre à partir des informations détaillées ci-dessus en justifiant votre réponse. Le SGBD utilise des index seulement si le coût est inférieur à celui d'un parcours séquentiel.

Question 1 : Quel est le nombre de nuplets de la relation Livre ?

Question 2 : Quel est le coût de la requête ?

`Select * from Livre where résumé = 'Un tutoriel pour tout savoir sur les index';`

Question 3 : Quels sont la cardinalité et le coût de la requête ? `Select * from Livre where année >=2013;`

Question 4 : Quels sont la cardinalité et le coût de la requête ? `Select * from Livre where année >=2000;`

Question 5 : La requête suivante utilise-t-elle un index ? `Select * from Livre where prix=10;`

Question 6 : Détailler le pseudo-code pour exécuter la requête affichant les livres à 80 euros publiés entre 2000 et 2002 (soit 3 années) : `select * from Livre where année between 2000 and 2002 and prix=80;`

Rmq, deux index sont utilisés.

Question 7 : La requête suivante utilise l'index sur l'attribut *prix*. `Select sum(prix) from Livre ;`

L'index contient une racine, un niveau de nœuds intermédiaires et un niveau de feuilles. Chaque valeur dans une feuille est associée à une liste de ROWID. Détailler l'exécution de la requête : quelles lectures (quels nœuds de l'index, quelles listes de ROWID, ...) et quels calculs ?

Exercice 4 (ref 16-3): Arbres B+

Sauf indication contraire, tous les arbres sont de type arbreB+ d'ordre **1** (i.e., il y a 1 ou 2 valeurs par nœud). On utilise la syntaxe suivante pour représenter un nœud de l'arbre : *N (v_1, v_2, …)* où *N* est le nom du nœud et les v_i sont les valeurs. Quand la feuille F déborde, on garde les **2** plus petites valeurs dans F, la plus grande valeur sera dans la nouvelle feuille. S'il faut choisir une valeur pour un nœud intermédiaire, la choisir, autant que possible, identique à une valeur existant dans une feuille. Toutes les valeurs sont des nombres entiers.

Question 1)

a) Au moment d'insérer une nouvelle valeur, quel est l'inconvénient d'avoir un arbre où tous les nœuds sont déjà remplis?

b) Les trois premières feuilles d'un arbre sont **F1**(9) **F2**(10, 15) **F3**(16). Lorsqu'on insère la nouvelle valeur 13, pourquoi décide-t-on de ne **pas** redistribuer avec F1 ou F3 ?

c) Un arbre a 20 feuilles. Les feuilles et les autres nœuds sont le **plus** remplis possible. Combien de niveaux a l'arbre ? Tenir compte du niveau de la racine et de celui des feuilles. Par exemple, un arbre avec une racine et 3 feuilles a 2 niveaux.

d) Un arbre a 17 feuilles. Les feuilles et les autres nœuds sont le **moins** remplis possible. Combien de niveaux a l'arbre ?

e) Un arbre contient dans ses feuilles les valeurs consécutives {10,11, ... ,26}. Les feuilles et les autres nœuds sont le plus remplis possible. Que contient la racine ?

Question 2) Soit l'arbre A0 composé d'une racine **N_1**(8, 21), et des feuilles **F_1**(4), **F_2**(10, 13) et **F_3**(21).

On insère 8. On obtient A1. Dessiner A1.

Question 3) Lors d'une suppression, on considère si possible la redistribution à gauche puis à droite, seulement entre des nœuds ayant le même père. L'arbre initial S_0 est composé :

d'une racine R(27)
des valeurs {21, 25, 100} dans les nœuds intermédiaire nommés N_i
et des valeurs {1, 2, 21, 24, 25, 26, 91, 100} dans les feuilles nommées F_j

a) Dessiner S_0.

b) On supprime 91 dans S_0. Représenter l'arbre S_1 obtenu (dessiner seulement les nœuds modifiés).

c) On supprime successivement les valeurs 24 puis 21 dans l'**arbre initial S_0**. Représenter l'arbre S_2 obtenu.

d) On supprime successivement dans l'**arbre initial S_0** les valeurs, dans l'ordre croissant, 1, 2, 21,... jusqu'à ce que l'arbre perdre un niveau. Représenter l'arbre S_3 obtenu.

Exercice 5 (ref ra5): Effet d'une séquence insertion-suppression

On étudie l'effet de l'insertion d'une valeur *v* suivie de la suppression immédiate de *v*. Cela ne modifie pas l'ensemble des valeurs indexées. Mais est-ce que cela modifie la structure de l'arbre ? Pour répondre à cette question, on considère un arbre B+ d'ordre 2, nommé A1. La racine de A1 contient les clés 13, 17, 24, 30. Ses feuilles contiennent (toutes feuilles confondues) les valeurs 2, 3, 5, 7, 14, 16, 19, 20, 22, 24, 27, 29, 33, 34, 38, 39.

1) Dessiner A1.
2) Donner 4 valeurs de clé telles que leur insertion successive puis leur suppression dans l'ordre inverse résulte dans un état identique à l'état initial
3) Donner une valeur de clé dont l'insertion suivie de la suppression résulte dans un état différent de l'état initial.
4) Montrer l'état de l'arbre résultant, à partir de l'arbre A1, de l'insertion de la clé 30.
5) On veut déterminer le nombre minimal de clés à insérer consécutivement dans l'arbre **A1 initial** pour qu'il gagne deux niveaux en hauteur (c'est-à-dire pour passer de 2 à 4 niveaux) ?
   a) Sachant qu'on veut insérer le moins de clés possible, quelle est, parmi F1 à F5, la première feuille dans laquelle on insère une valeur? Par la suite dans quelles feuilles est-il préférable d'insérer les autres valeurs ?
   b) Quels nœuds de l'arbre doivent être pleins pour que l'insertion d'une clé dans une feuille provoque (en cascade) un éclatement de la racine ?
   c) Représenter l'arbre obtenu **après** la dernière insertion qui fait passer l'arbre de 3 à 4 niveaux.

Exercice 6 : (ref p02-1) Profondeur des arbres B+

Soit la relation **R** (a1, a2, a3) contenant 10 millions de tuples. L'attribut a1 est la clé primaire, il est indexé par un arbre B+. L'index est dense : chaque valeur de a1 correspond à une clé d'une feuille de l'arbre.

Question 1) Le nombre de clé par nœud est *nc* tel que : $nc \in [4,8]$ pour tout nœud sauf la racine, $nc \in [1,8]$ pour la racine
La profondeur *p* d'un arbre correspond au nombre de niveaux, racine incluse, soit:

p = 1 pour un arbre réduit à sa seule racine,
p = 2 pour un arbre ayant seulement une racine et des feuilles,
etc…

Quelle est la profondeur minimale de l'index sur a1? Expliquez brièvement votre réponse.

Exercice 7 (p02-2): Arbres B+ : insertion et suppression de clés pts

Soit un arbre B+ dont le nombre de clé par nœud est *nc* tel que :
$nc \in [2,4]$ pour tout nœud sauf la racine, et $nc \in [1,4]$ pour la racine.
a) Dessiner l'arbre **A1** ayant 2 niveaux tels que :
- la racine a les clés 10, 20, 23, 40.
- les feuilles ont les clés 1, 2, 4, 6, 10, 12, 14, 20, 22, 24, 30, 32, *c*, 42 avec $(32 < c < 42)$.

b) Donner toutes les valeurs possibles pour la clé *c*.
c) Représenter l'arbre **A2** après insertion de la clé 8 dans A1, sans redistribuer les clés avec les voisins. En cas d'ajout d'un nœud, le nœud créé doit contenir le nombre minimal de clés.

d) Représenter l'arbre **A3** après suppression de la clé 8 dans A2. On considère la redistribution éventuelle avec le voisin de gauche d'abord.
e) Combien de clés au maximum peut on supprimer dans l'arbre **A3** sans qu'il perde un niveau ? Donner un exemple de clés que l'on peut supprimer.

Exercice 8 (ref p03) : Arbre B+

Soit un arbre B+ avec 3 niveaux. Le nombre de clés par nœud est tel que :
la racine et les nœuds intermédiaires contiennent de **1** à **2** clés,
les feuilles contiennent de **2** à **3** clés,

Question 1 :

1.1) Dessiner l'arbre **A1** ayant 3 niveaux tels que :
Les feuilles ont les clés 1, 4, 9,16, 25, 36, 49, 54, 61, 70, 81, 84, 87, 88, 95, 99.
Le niveau intermédiaire a les clés 9, 54, 70, 88
La racine contient 2 clés (pour les clés de la racine: choisir les **plus petites** valeurs possibles **parmi les clés des feuilles**).

1.2) Représenter l'arbre A2 après insertion de la clé 32 dans A1, sans jamais redistribuer de clé avec les voisins. Créer un nouveau nœud en cas de débordement d'une feuille ou d'un nœud intermédiaire.

1.3) Soit l'arbre A3 après insertion de la clé 32 dans **A1**. L'arbre A3 est obtenu en redistribuant si possible les clés des niveaux intermédiaires. Quel est le nombre de nœuds de l'arbre A3 (racine, nœuds intermédiaires et feuilles inclus) ?

1.4) Représenter l'arbre A4 après suppression de la clé 16 dans **A1**. L'arbre A4 est obtenu en redistribuant si possible les clés des feuilles avec les voisins.

1.5) Quel est le nombre minimum de clé à supprimer dans A1 pour qu'il perde un niveau ? Justifier votre réponse. Donner un exemple de clés à supprimer (en choisissant des valeurs de clé les plus petites possibles).

Question 2 : Index plaçant et non dense
Soit la relation **Produit** (numéro, prix). Les produits sont stockés dans l'ordre croissant du numéro. L'attribut *numéro* est une clé, il est indexé par un arbre B+. Les attributs sont indépendants et leur distribution est uniforme.
La relation *Produit* a 100 000 n-uplets, une page de données contient 100 n-uplets.
L'index est non dense (i.e., une seule clé par page de données).

2.1) Quel est le nombre minimum de clés au niveau des feuilles ? Justifier votre réponse.

2.2) Quel est le nombre minimal de niveaux de l'index ? Justifier votre réponse.

Exercice 9 : (ref p04 et ra94) Arbre B+

On considère des arbres B+ contenant des données de type entier, tel que les nœuds et les feuilles contiennent au plus quatre valeurs. Les noeuds (sauf la racine) et les feuilles doivent être au moins à moitié pleins (2 valeurs au moins). La hauteur d'un arbre est égale à son nombre de niveaux. Un arbre de hauteur 1 est réduit à sa seule racine.

Question 1. Donnez un exemple d'arbre B+ dont la hauteur passe de 2 à 3 lorsqu'on y insère la valeur 25. Donnez les deux arbres, avant et après l'insertion. Utiliser la trame quadrillée pour dessiner les nœuds et laisser un espace entre deux nœuds. Répondre page suivante.

Question 2. Donnez un exemple d'arbre B+ dans lequel la suppression de la valeur 25 conduit à une redistribution. Donnez les deux arbres, avant et après la suppression.

Question 3. Donnez un exemple d'arbre B+ dans lequel la suppression de la valeur 25 conduit à une fusion de deux nœuds, mais ne modifie pas la hauteur de l'arbre. Donnez les deux arbres, avant et après la suppression.

Exercice 10 : (ref p05 et s02) Arbre B+

On considère un arbre B+ tel qu'un nœud quelconque peut contenir de **1** à **3** clés.

Question 1

a) Dessiner l'arbre résultant de l'insertion successive des 10 clés 1, 2, ..., 10, dans l'ordre croissant, dans un arbre initialement vide. En cas de débordement d'une feuille, l'éclater en 2 feuilles ayant chacune le même nombre de clés. Utiliser la trame quadrillée pour dessiner les nœuds et laisser un seul espace entre deux nœuds.

b) On considère l'insertion successive de N clés 1, 2, ..., N, dans l'ordre croissant, dans un arbre initialement vide. Donner la valeur maximale de N lorsque l'arbre obtenu est de profondeur 3 (un seul niveau intermédiaire).

c) On considère l'arbre de profondeur 3 qui a le nombre maximum de clés (sans se préoccuper de la façon d'obtenir cet arbre), et M le nombre total de clés dans ses feuilles. N_{max} est il égal à M ? Justifier.

Question 2

Donner une suite de 10 clés à insérer successivement, dans un arbre vide afin d'obtenir un arbre de profondeur 2 (pas de niveau intermédiaire), en choisissant la valeur des clés parmi les entiers dans [1, 10]. Puis représenter l'arbre obtenu.

Indication : Répondre de manière à insérer autant que possible les plus petites valeurs en premier. Parmi toutes les réponses possibles, donner celle qui est la plus "proche" d'une suite croissante. C'est-à-dire, donner la suite qui est composée d'un nombre minimum de sous-suites croissantes.

EXERCICES DE RAPPEL

Ex. 11 (ref 15-5-s2) : Insertions successives

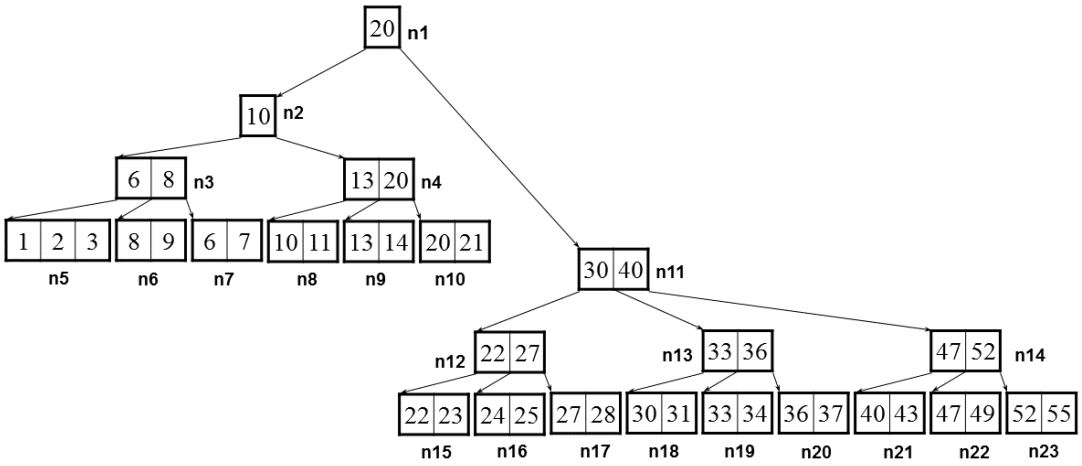
Dans cet exercice, les arbres sont d'ordre 1 (i.e., il y a 1 ou 2 valeurs par nœud).

1) Soit l'arbre A0 composé d'une racine N1(10), et de deux feuilles F1(5) et F2(20). A0 a 2 niveaux. On insère successivement dans A0 les nombres entiers 8 puis 30 puis 15. On obtient A1. Dessiner A1.

2) On insère dans A1 le nombre 10. On obtient A2. Dessiner A2.

Ex.12 (ref p05) : Chercher l'erreur

On considère l'arbre B+ d'ordre 2 suivant :



1) Cet arbre est-il équilibré ? Pourquoi ?

2) Trouver les erreurs dans cet arbre. Indiquer quel nœud n_i est erroné et expliquer brièvement l'erreur. S'il est possible de corriger l'erreur sans restructurer l'arbre, mais en modifiant seulement des valeurs de clés, alors suggérer une correction.

Ex. 13 (ref ra3) : Insertion, suppression, perte d'un niveau

On considère un arbre B+ d'ordre d=2. La racine de l'arbre **A1** contient la valeur 50. Un seul niveau intermédiaire contient (tous nœuds confondus) les valeurs 8, 18, 32, 40 73, 85. Les feuilles contiennent (toutes feuilles confondues) les valeurs 1, 2, 5, 6, 8, 10, 18, 27, 32, 39, 41, 45, 52, 58, 73, 80, 91, 99.

1) Dessinez l'arbre A1.
2) Dessiner l'arbre A2 après l'insertion de la valeur 9 dans A1. Combien valent L et E ?
3) Dessiner l'arbre A3 après l'insertion de la valeur 3 dans A1. Combien valent L et E ?
4) a) Dessiner l'arbre A4 après suppression de la valeur 8 dans A1. Si nécessaire, on envisage une redistribution à gauche. Combien valent L et E ?

   b) Même question mais en considérant uniquement la redistribution à droite si possible, sinon fusionner 2 feuilles.
5) Montrer l'état de l'arbre résultant, à partir de l'arbre A1, de l'insertion de la clé 46 suivie de la suppression de la clé 52
6) Montrer l'état de l'arbre résultant, à partir de l'arbre A1, de la suppression successive des clés 32, 39, 41, 45 et 73.

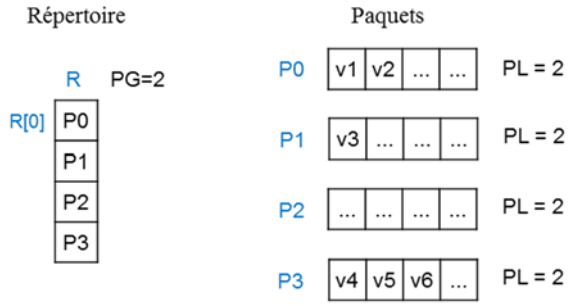
TD 2 : Hachage extensible

Notations et conventions

Pour toutes les tables de hachage étudiées en TD, on limite le nombre de valeurs dans un paquet à **4** au maximum. Ceci permet d'étudier plus facilement sur des petites structures les cas d'insertion avec d'éclatement et de suppression avec fusion. On utilise la notation suivante :

Le répertoire est noté **R [**P_0, P_1, P_2,..., P_{k-1} **] PG=***pg* avec
P_i ... les noms d'un paquet,
pg la profondeur globale.
Rmq : le répertoire contient k cases avec k = 2^{pg}
Un paquet est noté P_i**(**v_j, ..., ...**) PL=***pl* avec
P_i le nom du paquet, par exemple P0, P1, A ou B
V_j. les valeurs que contient le paquet,
pl la profondeur locale du paquet P_i.

On peut aussi préciser le contenu d'une case particulière du répertoire avec R[i]= P_j (avec R[0] étant la 1ère case).



Suppression : Lors d'une suppression, si un paquet devient vide, on tente de le fusionner seulement si sa profondeur locale est égale à la profondeur globale, sinon il reste vide.

Exercice 1 (16-4) : Table de hachage extensible

Question 1. Dans cette question. Un paquet peut contenir au maximum **2** valeurs (rmq : seulement 2, pas 4).

1) On considère une structure de hachage extensible de profondeur globale PG=3. Le répertoire contient 8 paquets ayant tous une profondeur locale PL=3 : R [P0, P1, P2, P3, P4, P5, P6, P7].

On insère les valeurs 1, 6, 12, 15, 19, 32, 42, 81, 808.

Que contiennent les paquets P0 à P7 ?

2) On considère une table de hachage extensible **T1** de profondeur globale PG=2. Le répertoire contient 4 paquets ayant tous une profondeur locale PL=2 : R [P0, P1, P2, P3].

P0(4, 44) P1(13, 17) P2(6) P3(3)

On supprime 6 de la table **T1** en tentant de fusionner les paquets vides. Quel répertoire obtient-on?

Préciser les paquets supprimés et/ou modifiés :

Question 2. Dans cette question un paquet peut contenir jusqu'à **4 valeurs** au maximum.

On considère la table **T2** avec les paquets A(9) et B(10).

a) Le répertoire est-il R[A, B] ou R[B, A] ? On insère 1, 2, 5, 7, 8, 12 dans T2. Représenter la table **T2'** obtenue.

b) On insère 13 dans **T2'** obtenue à la question précédente. Représenter la table **T3** obtenue.

c) On insère 41 dans **T3** obtenue à la question précédente. Représenter la table **T4** obtenue.

Question 3. On a deux tables de hachage H0 et H1 ayant une profondeur PG=1. Chaque table a 2 paquets avec **4 valeurs par paquet**. On veut indexer les nombres pairs dans H0 et impairs dans H1. H0 doit indexer les valeurs {2, 4, 8, 10,12} et H1 les valeurs {1, 5, 9, 11, 19}. Expliquer quelle fonction de hachage utiliser pour qu'on puisse indexer ces valeurs en remplissant les paquets existants, sans créer aucun nouveau paquet.

Exercice 2 (17-5) : Table de hachage extensible pts

Un paquet peut contenir au maximum **2** valeurs (rmq : seulement 2, pas 4).

Question 1. T1 a un répertoire R [P0, P1, P2, P3, P4, P5, P6, P7]. Le contenu des paquets est :

P0 (16, 104)	P4 (4, 12)
P1 (1, 137)	P5 (21)
P2 (10, 42)	P6 (14)
P3 (19)	P7 (15)

On supprime 19 dans T1. On obtient T2. Quel est son répertoire? Quels sont le contenu et la PL de chaque paquet modifié ?

Question 2. Soit R[A,B] le répertoire de **T3**. Les valeurs dans les paquets sont : 1, 23, 24.
Quels sont le contenu et la PL de A et B ?

On insère 3 dans **T3**. On obtient T4. Quel est son répertoire? Quels sont le contenu et la PL de chaque paquet ?

Question 3. On insère 7 dans **T4**. On obtient T5.

Quel est son répertoire? Quels sont le contenu et la PL de chaque paquet ?

Question 4. Donner un exemple de table de hachage T6 tel que :

- PG=2
- T6 a un paquet vide (qui n'est pas fusionné car sa profondeur est inférieure à PG)
- Tous les autres paquets de T6 sont pleins.

Question 5 (bonus) Question sur le hachage **linéaire** vu en cours. Une table de hachage **linéaire** a 4 paquets (P0 à P3) et contient les valeurs 1, 4, 8, 10, 14, 15.

Un paquet peut contenir **2** valeurs au maximum. On suppose que p=0 et que le taux d'occupation ne doit pas dépasser 75%, donc on en déduit qu'il faudra éclater P0 lors de la prochaine insertion.

Représenter la table **après** l'insertion de 22.

Exercice 3 (p15)

1) La table de hachage T0 contient seulement les valeurs 1, 4, 5, 6, 9, 14, 25. De combien de paquets a-t-on besoin au **minimum**? Préciser leur contenu, leur PL, le contenu du répertoire.

2) On insère 7 dans T0. On obtient T1. Représenter T1. Préciser ce que contient chaque case du répertoire R

3) On insère 13 dans **T1** obtenu à la question précédente. On obtient T2. Représenter T2.

4) On considère la table T3 avec les paquets A(10), B(16), C(17,31), D(30), E(36). Représenter T3. Rmq : vous devez préciser le contenu de chaque case du répertoire (R[0] est différent de A, la taille du répertoire est une puissance de 2).

5) On supprime 10 dans T3. On obtient T4. Représenter T4

6) On supprime 30 dans T4 obtenu à la question précédente. On obtient T5. Représenter T5.

Exercice 4 (p14) : Table de hachage extensible pts

Dans toutes les tables de hachage considérées, un paquet ne peut pas contenir plus de 4 valeurs.
Une table de hachage T1 contient *seulement* les 5 paquets suivants:

A (8)
B (9, 29)
C (10, 22, 42)
D (11, 27)
E (23, 39)

1) Structure de T1.

a) Est-ce que le répertoire est tel que R[0] = A et R[4] = E ?

b) Pour chaque valeur *x* de chaque paquet, que vaut *x mod 8* ?

c) Le répertoire a 8 cases, que contiennent-elles ? Préciser le nom du paquet référencé dans chaque case.

9

2) Dans T1, on insère successivement 6 puis 18. Quels sont les paquets créés et/ou modifiés et leur contenu ? Préciser aussi les cases modifiées du répertoire.

3) Dans T1, on insère successivement 3 valeurs dans le paquet E. Est-ce que cela a pour effet de doubler la taille du répertoire ?

4) Dans T1, on supprime 9 puis 29. Quels sont les paquets supprimés et/ou modifiés et leur contenu ?

5) Dans T1, on supprime 8. Quels sont les paquets supprimés et/ou modifiés et leur contenu ?

6) On a un million (10^6) de valeurs à indexer. Quelle sera la taille minimale du répertoire ?

7) Question bonus : index bi-attribut. On considère la relation Restau(a, b, c) contenant 200 triplets. Les valeurs a, b, c sont des entiers. On crée un index avec la commande :

create index IAB on Restau (a, b);

L'index utilise des techniques de hachage avec des paquets contenant au plus 4 valeurs. On veut utiliser un répertoire en forme de matrice pour pouvoir accéder à une case du répertoire en fonction de la valeur de *a* et de celle de *b*. Préciser la (ou les) fonction(s) de hachage à utiliser et expliquer comment accéder au triplet dont les valeurs (a, b) valent (12, 130).

Exercice 5 (p13) : Indexation avec table de hachage — 4 pts

La fonction x modulo y est notée : *x mod y*. Chaque paquet **contient au plus 2 valeurs.**

Question 1. On considère un répertoire de profondeur globale PG=1. Avec 2 paquets P0 et P1 tels que R=[P0,P1]. Initialement les deux paquets contiennent :

P0(4,8) **P1**(1,3)

On insère la valeur 12.

a) Est-ce que cette insertion provoque plus d'un éclatement ? Quelle sera la profondeur globale de la table obtenue après l'insertion ?
b) Détailler la table obtenue après insertion. Préciser le contenu des paquets modifiés ou créés, et leur profondeur locale (PL).

Question 2. On a un répertoire de profondeur PG=3. Les paquets contiennent les valeurs : 1, 7, 8, 10, 11, 16, 20.
a) De combien de paquets a-t-on besoin au minimum? Préciser leur contenu, leur PL, les liens entre les cases et les paquets

b) On insère 19. Détailler la table obtenue en précisant *seulement* les paquets, les liens et les profondeurs modifiés.

Question 3. Répartition d'une table de hachage. Pour indexer un attribut dont le domaine a de nombreuses valeurs, on veut construire une « grande » table de hachage dont le répertoire a 32 cases et autant de paquets. Or, on suppose que la plus grande table tenant dans la mémoire d'une machine a un répertoire de 8 cases et 8 paquets de 2 valeurs. Ainsi, on propose de répartir l'index sur 4 machines M0 à M3 gérant chacune la table de hachage T0 à T3 respectivement. Pour chaque Ti, la profondeur est PG=3.
Initialement, chaque Ti a déjà 8 paquets (PL=3) notés $P_{i,0}$ à $P_{i,7}$. Il y a une valeur et une place libre dans chaque paquet.

On insère la valeur v = 49. Décrire les étapes de l'insertion, et préciser dans quelle Ti et quel paquet est insérée la valeur 49.

EXERCICES DE RAPPEL

Exercice 6 (rappel): Hachage extensible

On considère une base de données contenant la relation Personne (nom, prénom, ville). On veut indexer les personnes sur l'attribut ville. La relation contient 11 villes différentes dont les codes sont 1, 4, 5, 7, 10, 12, 15, 16, 19, 21, 32. Pour cela, on construit une structure de hachage extensible contenant une entrée pour chacune des 11 valeurs.

1) Initialement, on veut construire une table de hachage avec suffisamment de place pour contenir les 11 valeurs. Avec au plus 4 valeurs par paquets, combien faut-il de paquets au minimum? Quelle doit être la taille minimale du répertoire pour atteindre ces paquets ?

10

2) Dessiner la table de hachage, nommée T1, après avoir inséré les 11 valeurs. Représenter le répertoire avec le nom des paquets qu'il contient et sa profondeur globale. Représenter chaque paquet avec les valeurs qu'il contient et sa profondeur locale.

3) Expliquer pas à pas comment retrouver le paquet contenant la valeur 7.

4) Insérer 13

5) Insérer 20

6) Insérer 29

7) On considère la table obtenue. Donner un exemple de plus petit ensemble de valeurs à supprimer pour obtenir une division par 2 du répertoire.

Exercice 7 (rappel) : Hachage extensible

On considère une structure de hachage extensible de profondeur globale 3. Dans l'état initial, nommé **T0**, les paquets du répertoire sont les suivants : R[A, B, C, D, A2, B, C, D] (*i.e.*, R[0]=A, ..., R[7]=D), et PG=3

Les valeurs (tous paquets confondus) sont les suivantes : 1, 4, 5, 7, 10, 12, 15, 16, 20, 21, 36, 51, 64.

1) Représenter T0.

2) Insérer 68 dans T0.

3) On considère la table T0 initiale. Insérer 17 et 69 dans T0.

4) Supprimer 21 de T0.

5) Supprimer 10, puis 64 et 16 deT0.

TD Optimisation de requêtes

Notations et conventions

Dans tous les exercices, on suppose que la distribution des attributs est uniforme. Les attributs sont tous indépendants les uns des autres. On s'intéresse au **coût** des opérations en termes de quantité d'accès aux données. L'unité de coût est la page. P(R) est le nombre de pages de R. Les principales formules de coût sont (*cf* cours pour plus de détails) :

- **Sélection** avec un prédicat *pred* défini sur l'attribut *a*.

(S1) coût(σ_{pred} (R)) = card(σ_{pred} (R)) si index R(*a*)
(S2) OU = P(R) × card(σ_{pred} (R)) / card(R) si R **stockée** et triée sur l'attribut *a* (index **plaçant**)
(S3) OU = coût(R) sinon

- **Lecture séquentielle**

(L1) coût(R) = P(R) si R est une table

- **Jointure par boucles imbriquées**

(J1) coût($R \bowtie_a S$) = coût(R) + card(R) × card($\sigma_{a=v}$ (S)) si index S(a)
(J2) OU = coût(R) + P(R) × P(S) si S est une table **sans** index S(a)

Si S n'est **pas** une table mais est une expression, il faut rajouter son coût d'évaluation et le coût de la stocker.

(J3) coût($R \bowtie_a S$) = coût(S) + P(S) + coût(R) + P(R) × P(S)

- **Jointure par hachage**

(J4) coût($R \bowtie_a S$) = coût(R) + coût(S)

Exercice 1 (17-3) : Jointures pts

On considère une base (différente de la base des exercices précédents) :

Livre (titre, numL, prix) // il y a **10** *prix* parmi (10, 20, ..., 90, 100)

Stock (numL, numM, quantité) // *numL* est le n° du livre, *numM* est le n° du magasin.
// il y a **50** valeurs de *quantité* parmi (1, 2,..., 49, 50)

Magasin (numM, ville) // il y a **10** *villes*

Il y a **2000** livres et **50** magasins. Tous les livres sont en stock dans tous les magasins.

La requête **R1** est :

```
Select li.titre, m.ville
From Livre li, Stock s, Magasin m
Where li.numL = s.numL and m.numM = s.numM
And s.quantité =2 and li.prix=10 and ville <> 'Paris'
```

Le nombre de pages d'une relation est P(A) = card(A)/10 car on suppose qu'il y a 10 nuplets par page, quelle que soit la relation. On *néglige* le coût des opérateurs traités en pipeline.

Question 1 : On dispose des index non plaçants suivants :

Livre(numL), Livre(prix),

Stock(numL), Stock(quantité)

Quel est l'arbre **linéaire à gauche** de R1 tel que l'ordre des jointures soit Livre, Stock, Magasin, et que les sélections et projections soient le plus bas possible sur la branche principale ? Décomposer la réponse en 2 expressions algébriques T1 et T2.

Que valent les cardinalités suivantes ? Card(Stock), card(Stock ⋈ Livre) , card(Stock ⋈ Livre ⋈ Magasin)

Soit T3 = $\sigma_{quantité=2}$ (.Stock) et R3 = T3 ⋈ Livre

Les index Stock(quantité) et Livre(numL) sont utilisés. Donner le coût de T3 et R3.

Soit T4 = $\sigma_{prix=20}$ (Livre) et R4 = T4 ⋈ Stock.

Les index Livre(prix) et Stock(numL) sont utilisés. Donner le coût de T4 et R4.

Question 2 : Les 3 relations sont initialement triées respectivement sur l'attribut suivant (index plaçant) :

Livre (numL) Magasin (numM) Stock (quantité)

On dispose de **M=50** pages de mémoire pour trier une relation intermédiaire.

Expliquer les étapes du plan de la requête **R1** tel que les **jointures sont faites par fusion**. Le coût est le nombre de pages lues et écrites.

a) : traiter E1 = $\sigma_{quantité=2}$ (Stock)

b) : trier E1 sur numL et l'écrire temporairement dans T, puis calculer E2 = $\sigma_{prix=10}$ (T ⋈ Livre).

c) : trier E2 sur numM puis calculer R1 = $\pi_{titre,ville}$ $\sigma_{ville<>'Paris'}$ (E2 ⋈ Magasin)

Exercice 2 (16-2) : Optimisation de requête et plan d'exécution pts

Soit la base :

Chanteur (nomChanteur, âge, style)
Chanson (titre, nomChanteur, durée, année)
Parole (titre, texte, langue)

On a les index **non** plaçants :

Les clés (soulignées) sont indexées par :	Les autres index sont :
IndNom on Chanteur(nomChanteur), *IndCTitre* on Chanson(titre) *IndPTitre* on Parole (titre).	*IndAge* on Chanteur(age) *IndAnnée* on Chanson(année) *IndLangue* on Parole(langue)

Soit la requête **R1** :

```
Select t.nomChanteur, t.style, p.texte
From Chanteur t, Chanson s, Parole p
Where t.nomChanteur = s.nomChanteur and s.titre = p.titre
and p.langue = 'FR' and t.age = 20 and s.annee = 2016;
```

Soit le plan **P1** exécutant R1 :

```
 ------------------------------------------------------
| Id  | Operation                      | Name        |
 ------------------------------------------------------
|   0 | SELECT STATEMENT               |             |
|   1 |  NESTED LOOPS                  |             |
|   2 |   NESTED LOOPS                 |             |
|   3 |    NESTED LOOPS                |             |
|*  4 |     TABLE ACCESS FULL          | CHANSON     |
|*  5 |     TABLE ACCESS BY INDEX ROWID| CHANTEUR    |
|*  6 |      INDEX UNIQUE SCAN         | INDNOM      |
|*  7 |    INDEX UNIQUE SCAN           | INDPTITRE   |
|*  8 |   TABLE ACCESS BY INDEX ROWID  | PAROLE      |
 ------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
  4 - filter("S"."ANNEE"=2016)
  5 - filter("T"."AGE"=20)
  6 - access("T"."NOMCHANTEUR"="S"."NOMCHANTEUR")
  7 - access("S"."TITRE"="P"."TITRE")
  8 - filter("P"."LANGUE"='FR')
```

Question 1) Compréhension de P1.

a) Dans quel ordre les jointures sont-elles traitées ? Répondre en listant, dans le bon ordre, le nom des relations.

b) La sélection âge=20 est-elle évaluée avant d'évaluer le prédicat de jointure entre Chanteur et Chanson ?

c) P1 utilise-t-il l'index sur Chanson(année) ? Pourquoi ?

d) P1 utilise-t-il l'index sur Parole(langue) ? Pourquoi ?

e) Dessiner l'arbre algébrique linéaire à gauche de R1 tel que les jointures sont dans le **même ordre** que dans P1.Préciser les éventuelles projections qui peuvent être faites sans modifier l'ordre des autres opérations. Inscrire les feuilles de l'arbre sur les traits pointillés en bas du dessin, la racine est en haut.

Question 2) Jointure par hachage.
On considère le plan **P2** qui exécute la requête R1 en utilisant seulement des jointures par hachage et autant d'index que possible. Afin de déterminer l'ordre des opérations, on suppose que les paroles françaises ont une taille trop grande pour tenir en mémoire. On suppose que les chanteurs de 20 ans et les chansons de 2016 peuvent tenir en mémoire.

a) Dessiner l'arbre algébrique de la requête telle que les opérations sont dans l'ordre de P2. Ecrire les feuilles de l'arbre sur les traits pointillés, la racine est en haut du dessin.

b) Expliquer brièvement les étapes de l'évaluation de P2

c) Quels index peuvent être utilisés pour P2, parmi les index existants ?

Question 3) Coût d'un plan.
Tous les attributs numériques sont entiers. Les hypothèses vues en cours (uniformité, indépendance) sont vérifiées.
Le coût est exprimé en nombre de lectures de pages.

Il y a 10 000 chanteurs tenant sur 100 pages. L'âge est dans [11, 60]

Il y a 200 000 chansons tenant sur 1 000 pages. L'année est dans [1917, 2016],

Il y a 200 000 paroles tenant sur 20 000 pages. Il y a 200 langues.

a) Quel est le coût minimal pour la requête T1 suivante ? select * from Chanteur where âge between 41 and 50 ;

Préciser si l'index *IndAge* est utilisé ou non.

b) Quel est le coût minimal pour la requête T2 suivante ? select * from Chanson where année > 2010 ;

Préciser si l'index *IndAnnée* est utilisé ou non.

c) Quel est le coût minimal pour la requête **T3** suivante ? select * from Parole where langue = 'FR' or langue ='EN' ;

Préciser si l'index *IndLangue* est utilisé ou non.

d) Quel est le coût de **R2** exécutée en utilisant seulement des jointures par **hachage** et en supposant qu'on dispose d'un espace mémoire libre de taille infinie ?

R2 : select *
from Chanteur t, Chanson s, Parole p
where t.nomChanteur = s.nomChanteur and s.titre = p.titre
and (p.langue = 'ES' or p.langue = 'RU') and (t.âge between 21 and 30) and s.année > 2000 ;

e) Quelle est la taille **minimale** de la mémoire dont on doit disposer, exprimée en nombre de pages, pour exécuter R2 en utilisant seulement des jointures par hachage ?

f) Quel est le coût **minimal** de **R3** en utilisant une jointure par boucles imbriquées avec index ?

R3 : select *
from Chanson s, Chanteur t
where s.nomChanteur = t.nomChanteur ;

Exercice 3 : Jointure entre 2 relations

Soit les relations :
R (a, b) et **S** (c),
Le domaine des attributs est : a∈[1,1000], b∈[1,10], c∈[1,100].
Les cardinalités des relations sont : card(R) = 1000, card(S)= 100
La taille en nb de pages des relations est : P(R) = 500, P(S) = 10

Soit la requête :
select *
from R, S
where a = c and b =1
Pour chaque question, énumérer tous les plans équivalents (notés P1 à P4) et calculer leur coût. Pour cela, représenter un plan d'exécution sous la forme d'un arbre d'opérateurs. Donner le coût et la cardinalité de chaque opérateur de l'arbre, donner le coût total de l'arbre.

1) Il y a seulement un index sur R.a et un index sur R.b,
2) Il y a seulement un index sur S.c
3) Il y a seulement un index sur R.b et un index sur S.c.

Exercice 4 : Optimisation de requêtes de sélection

Soit le schéma relationnel décrivant l'organisation d'un laboratoire en départements contenant des employés et des projets.

Emp (Enum, salaire, age, Dnum)

Dept (Dnum, Pnum, budget, statut)

Proj (Pnum, code, description)

Tous les attributs sont des nombres entiers sauf *statut* et *description* qui sont des chaînes de caractères. Les attributs soulignés forment une clé. La clé de Dept est composée de deux attributs, (l'attribut Dnum seul n'est pas une clé primaire de Dept).

La taille d'un n-uplet est respectivement de 20 octets pour Emp, 40 octets pour un Dept et 2000 octets pour Proj. Le nombre de n-uplets par relation et respectivement de 20 000 pour, Emp, 5000 pour Dept et 1000 pour Proj. La taille d'une page, sur le disque ou en mémoire, vaut 4000 octets. La fonction page(R) retourne le nombre de pages contenant les n-uplets de R.

La distribution des valeurs des attributs est uniforme. Les attributs sont indépendants. Le domaine de l'âge des employés est l'ensemble des nombres entiers de 20 à 69 inclus : {20, 21, …, 69}. Le domaine des budgets est l'ensemble des multiples de 1000 inclus dans]100 000, 600 000].

On suppose que tous les index sont des arbres B+. Un index sur l'attribut A est dit *plaçant* si les données sont **triées** sur le disque dans l'ordre des valeurs de A. Un index sur l'attribut A est dit *non plaçant* si les données ne sont **pas triées** sur le disque dans l'ordre de A.

L'estimation du coût des opérations repose sur le modèle suivant:

Question 1. Soit la requête R1: `select * from Emp where age=30`

a) Quel est le coût pour traiter R1 s'il existe un index non plaçant sur l'attribut *age* ?

b) Quel est le coût pour traiter R1 s'il existe un index plaçant sur l'attribut *age* ?

Question 2. Soit la requête R2 `select * from Proj where code=20`

On suppose que la cardinalité de R2 est identique à celle de R1.

a) Quel est le coût pour traiter R2 s'il existe un index non plaçant sur l'attribut *code* ?

b) Quel est le coût pour traiter R2 s'il existe un index plaçant sur l'attribut *code* ?

Question 3. Soient *n* un nombre entier et la requête R3 dépendant de *n* :

```
select * from Dept where budget > n
```

On suppose qu'il existe un index **non plaçant** sur l'attribut *budget*. Pour quelles valeurs de *n*, la lecture séquentielle de Dept est moins coûteuse que l'accès par index, pour traiter R3 ? Répondre en donnant toutes les valeurs de *n*.

Exercice 5: Jointure entre 3 relations

L'objectif de cet exercice est de comparer deux méthodes pour le choix du plan d'exécution d'une requête. La première méthode est la transformation de plan basée sur des heuristiques. La 2ème méthode est la génération exhaustive de l'espace de recherche associée au choix du plan candidat de moindre coût.

Soit le schéma **R1**(A,B, …), **R2**(A,B, …) et **R3**(A, B, …)

Soit la requête :

```
select R1.A, R3.B
from R1, R2, R3
where R1.A=R2.A and R2.A=R3.A and R1.B=1 and R2.B=2
```

1) Donner l'arbre algébrique, nommé P1, correspondant en respectant l'ordre des prédicats donnés dans la clause *where*.
2) Donner un arbre équivalent, nommé P2, en appliquant les opérations les plus réductrices (sélection puis projection) d'abord.
3) Voir le modèle de coût vu en cours.

On suppose 5 nuplets par page : page(R) = card(R)**/5**

- On suppose que R1, R2, R3 ont les caractéristiques suivantes :

- il y a un **index** sur R1(B) , A est clé primaire de R1, R2 et R3 (il existe un index pour chaque clé primaire)
- card(R1) = card(R2) = card(R3) = 1000. Les domaines de A sont identiques: $\pi_A(R1) = \pi_A(R2) = \pi_A(R3)$
- il y a **10** valeurs possibles pour B, uniformément réparties dans R1 et aussi dans R2 et dans R3.

Questions :

a) Quelle est la cardinalité du résultat de la requête ?

b) Pour simplifier, on ignore les projections (car leur coût est nul). Donner l'expression algébrique de P1' (resp P2') correspondant à P1 (resp. P2) sans aucune projection.

c) Donner le coût de P1' et P2'. Préciser votre réponse en détaillant le coût et la cardinalité des résultats intermédiaires.

d) Quel est l'arbre de coût minimal pour évaluer la requête ? Quel est son coût ?

Exercice 6: Club de joueurs : A faire ou à continuer en TME

Soit le schéma relationnel :

Joueur (licence: integer, cnum : integer, salaire: integer, sport: char(20))

Club (cnum: integer, nom: char(20), division: integer, ville : char(10))

Finance (cnum: integer, budget: real, dépense: real, recette: real)

On considère la requête :

```
SELECT C.nom, F.budget
FROM Joueur J, Club C, Finance F
WHERE J.cnum = C.cnum AND C.cnum = F.cnum
AND C.division = 1 AND J.salaire > 59000 AND J.sport = 'aviron'
```

1) Déterminer un arbre d'opérateurs de l'algèbre relationnel qui reflète l'ordre des opérations qu'un optimiseur de requête peut choisir. Combien y a-t-il de possibilités équivalentes pour ordonner les jointures ?

2) Pour réduire l'espace de recherche exploré pendant l'optimisation, on considère seulement les arbres de jointure qui n'ont pas de produit cartésien et qui sont linéaires à gauche. Donner la liste de tous les arbres de jointure construits. Expliquer comment vous obtenez cette liste.

Les informations suivantes sont extraites du catalogue du SGBD.
Les attributs Joueur.cnum, Joueur.salaire, Club.division, Club.cnum et Finance.cnum sont indexé par un arbre B+.
Le salaire d'un joueur est compris entre 10.000 et 60.000 EUR.
Les joueurs peuvent pratiquer 200 sports différents. Un club est en division 1 ou 2.
La BD contient au total 50000 joueurs et 5000 clubs. Il y a un nuplet d'information financière par club.

3) Pour chaque relation (Joueur, Club et Finance) estimer le nombre de nuplets qui sont sélectionnés après avoir traité les prédicats de sélection et avant de traiter les jointures.

4) D'après la réponse à la question précédente, quel est l'arbre de jointure de coût minimum que l'optimiseur construit ?

UFR 919 - Master d'informatique - M1

EXTRAIT du partiel 2019

TD sur l'OPTIMISATION de requêtes

Exercice 7 (ref ER1-19-1): Plan d'une requête

On considère la base :

Etudiant (nE, prénom, nF*, mention, réside)

Formation (nF, université, diplôme, région)

Stage (nS, nE*, sujet, lieu, note)

//un étudiant peut faire plusieurs stages

INDEX	Relation	Attribut
I_Etu_nE	Etudiant	nE
I_Etu_nF	Etudiant	nF
I_Etu_réside	Etudiant	réside
I_Forma_nF	Formation	nF
I_Forma_diplôme	Formation	diplôme

Pour les questions demandant d'écrire du pseudo code : écrire **très lisiblement sans aucune rature, en indentant les lignes**. Utiliser les méthodes vues en TME. Pour un index, la méthode getRowIds(*condition*) retourne un ensemble de rowids. Pour une table, la méthode lireTuple(*r*) retourne le tuple de rowid *r* et la méthode tuples() retourne l'ensemble des tuples. Utiliser for(*x* : ...), if(..), affiche(...), init *M* pour initialiser une map, M[*clé*].append(*val*) pour ajouter une valeur associée à une clé, etc... Exemple d'instructions : *Etudiant.tuples()* : pour obtenir la liste des étudiants, *I_Etu_Réside.getRowIds('Paris')* pour obtenir une liste de rowids.

Question 1 : Sélection

```
Select  f.nF, f.université
From Formation f
Where f.région='IDF' and f.diplôme = 'Master Info' and université in ('Sorbonne', 'Diderot');
```

```
|  Id| Operation                    | Name             |
|   0| SELECT STATEMENT             |                  |
|   1|  TABLE ACCESS BY INDEX ROWID | FORMATION        |
|   2|   INDEX RANGE SCAN           | I_FORMA_DIPLOME  |
```

a) Préciser quels sont les prédicats évalués par la méthode access ou filter, et l'id de l'opération correspondante :

b) Pseudo code:

Question 2 : Utilisation de 2 index.

```
Select /*+ index_combine( e  I_Etu_nF  I_Etu_Réside ) */  e.prénom
From Etudiant e
Where e.prénom like 'A%' and e.réside = 'Paris' and e.nF = 'F1';
```

Ecrire le pseudo code de cette requête

Question 3 : Jointure par boucle avec index

```
select e.prénom, f.région          from Etudiant e, Formation f
where e.nF = f.nF    and    f.diplôme = 'Licence';
```

```
|Id  | Operation                    | Name          | Predicate Information
|  0 | SELECT STATEMENT             |               |
|  1 |  NESTED LOOPS                |               |
|  2 |   NESTED LOOPS               |               |
|  3 |    TABLE ACCESS FULL         |  ETUDIANT     |
|  4 |    INDEX UNIQUE SCAN         |  I_FORMA_NF   | access(E.NF=F.NF)
|  5 |   TABLE ACCESS BY INDEX ROWID| FORMATION     | filter(F.DIPLOME='Licence')
```

Ecrire le pseudo code de cette requête

Question 4 : Sélection et jointure par boucles imbriquées

```
select e.prénom, f.région          from Etudiant e, Formation f
where e.nF = f.nF   and   f.diplôme = 'Master';
```

```
|Id| Operation                    | Name           |Predicate Information
| 0| SELECT STATEMENT             |                |
| 1|  NESTED LOOPS                |                |
| 2|   NESTED LOOPS               |                |
| 3|    TABLE ACCESS BY INDEX ROWID |FORMATION       |
| 4|     INDEX RANGE SCAN         |I_FORMA_DIPLOME |access(F.DIPLOME='Master')
| 5|    INDEX RANGE SCAN          |I_ETU_NF        |access(E.NF=F.NF)
| 6|   TABLE ACCESS BY INDEX ROWID |ETUDIANT        |
```

Ecrire le pseudo code de cette requête

Question 5 : Jointure par hachage

```
select e.prénom, f.université
from Etudiant e, Formation f
where e.nF = f.nF   and   f.diplôme = 'M2'   and    e.réside = 'Aix';
```

```
|Id| Operation                      | Name           |Predicate Information
| 0| SELECT STATEMENT               |                |
| 1|  HASH JOIN                     |                |access(E.NF=F.NF)
| 2|   TABLE ACCESS BY INDEX ROWID  |ETUDIANT        |
| 3|    INDEX RANGE SCAN            |I_ETU_RESIDE    |access(E.RESIDE='Aix')
| 4|   TABLE ACCESS BY INDEX ROWID  |FORMATION       |
| 5|    INDEX RANGE SCAN            |I_FORMA_DIPLOME |access(F.DIPLOME='M2')
```

Ecrire le pseudo code. Les projections sont poussées autant que possible pour garder seulement les attributs nécessaires dans la Map.

Question 6 : Jointure par hachage entre 3 tables

```
select e.prénom, f.université, s.nS
from Etudiant e, Formation f, Stage s
where e.nF = f.nF and e.nE = s.nE;
```

```
|Id | Operation           | Name      | Predicate Information |
|  0| SELECT STATEMENT    |           |                       |
|  1|  HASH JOIN          |           | access(E.NE=S.NE)     |
|  2|   HASH JOIN         |           | access(E.NF=F.NF)     |
|  3|    TABLE ACCESS FULL| FORMATION |                       |
|  4|    TABLE ACCESS FULL| ETUDIANT  |                       |
|  5|   TABLE ACCESS FULL | STAGE     |                       |
```

Quelle projections sont faites lors des opérations n° 2 et n°3?

a) Ecrire le pseudo code en pensant à pousser les projections avant de remplir les Map.

b) On veut traiter cette requête avec des jointures par hachage et un plan linéaire à <u>droite</u> dont la forme est :

$\pi_{\text{prénom, université, nS}}$ [Stage ⋈ (Formation ⋈ Etudiant)]

Que contiennent les maps Map1 et Map2 ? La table Stage est-elle lue avant ou après la table Etudiant ? Expliquer brièvement en deux phrases ce qu'on fait pour chaque étudiant lu.

Exercice 8 (ref ER1-19-2) : Optimisation de requêtes

On connait les tables suivantes et leur cardinalité :

Etudiant (<u>nE</u>, prénom, nF*, mention)

Formation (<u>nF</u>, université, diplôme, région)

Stage (<u>nS</u>, nE*, sujet, lieu, note, spécialité, durée)

PratiqueLangue (<u>nE*, langue</u>, niveau)

Table	Card
Etudiant	1000
Formation	50
Stage	2000
PratiqueLangue	300

La taille en nombre de pages d'une relation est P(R) = card(R) / 10

On donne les statistiques suivantes, D est le nombre de valeurs distinctes d'un attribut :

Table	Attribut	D	Rmq
Etudiant	**nE**	1000	clé
Etudiant	nF	50	
Etudiant	mention	4	{P, AB, B, TB}
Formation	**nF**	50	clé
Formation	université	10	
Formation	diplôme	2	
Formation	région	5	

Table	Attribut	D	Rmq
Stage	**nS**	2000	clé. Il y a 2 stages par étudiant en moyenne
Stage	nE	1000	
Stage	lieu	100	
Stage	note		dans]0, 20]
Stage	spécialité	8	{ nosql, bigdata, ia, etc...)
Stage	durée	4	{3, 6, 9, 12} mois

Table	Attribut	D	Rmq
PratiqueLangue	nE	100	100 étudiants pratiquent 3 langues. Les autres étudiants ne pratiquent aucune langue.
PratiqueLangue	langue	3	

Question 1 : On appelle S_i un prédicat de sélection et SFi le facteur de sélectivité de l'opération σ_{si}(Stage). Calculer SFi.

S_1 : note >12 AND note ≤ 17

S_2: lieu IN ('Paris', 'Aix', 'Pau', 'Tour')

S_3: nE = 'E13' AND spécialité = 'ia

S_4 : durée = 6 OR durée = 12 OR note > 5

Question 2 : Quelle est la cardinalité card(Ri) des requêtes suivantes ? Justifier votre réponse.

R1 = [$\sigma_{mention=B}$ (Etudiant)] $\bowtie_{nF}$ [$\sigma_{région='idf'}$ (Formation)]

R2 = Etudiant $\bowtie_{nE}$ Stage $\bowtie_{nF}$ Formation

R3 = Stage $\bowtie_{nE}$ PratiqueLangue

Question 3. **Coût d'une requête**

Pour les questions suivantes, Le modèle de coût est celui présenté en cours. Les index sont **non** plaçants. Ils tiennent en mémoire donc le coût de traverser un index est nul : C_{index}=0 et C_{rowid} = 0. Chaque question précise les index qu'on utilise.

Quel est le coût de la requête suivante avec une jointure par boucles imbriquées et en utilisant l'index Formation(nF) ?

```
Select e.mention
From Etudiant e, Formation f
Where e.nF = f.nF and f.diplôme = 'Master';
```

Question 4 : Comparaison des coûts

Soit la requête

```
select *        from Etudiant e, Stage s
where e.nE = s.nE
and e.mention='TB' and s.note > 19 and s.lieu = 'Dol';
```

a) Le plan P1 de cette requête utilise l'index Stage(lieu) et fait une jointure par boucle avec l'index Etudiant(nE). Les autres sélections sont traitées le plus tôt possible et de telle sorte que P1 soit <u>linéaire à gauche</u>. Pour simplifier la réponse, on pose T1 = $\sigma_{note>19}$ ($\sigma_{lieu='Dol'}$ Stage). On connait la sélectivité SF($\sigma_{note>19}$ (Stage)) = 1/20.
Calculer coût(T1) et card(T1) ; exprimer le coût de P1 en fonction de T1 ; calculer le coût de P1.

b) Le plan P2 de cette requête lit la table Etudiant fait une jointure par boucles imbriquées avec l'index Stage(nE). Est-ce que le coût de P2 est inférieur au coût de P1 ?

c) Le plan P3 utilise l'index Stage(lieu) et fait une jointure par **hachage** ? Quel est son coût ?

Question 5 : Ordre des jointures. Soit la requête :

Etudiant $\bowtie_{nF}$ Formation $\bowtie_{nE}$ Stage $\bowtie_{nE}$ PratiqueLangue abrégée en : E $\bowtie_{nF}$ F $\bowtie_{nE}$ S $\bowtie_{nE}$ P

Enumérer les ordres de jointures équivalents (sans aucun produit cartésien) <u>commençant par **S**</u> pour cette requête