Université Paris Nanterre
Master 1- MIAGE Apprentissage
Contrôle 2 BDA

Novembre 2024

1. Questions de cours

1.1. Expliquez brièvement la différence entre stockage primaire et stockage secondaire d'un fichier.
1.2. Indiquez si l'affirmation **'Un index primaire prend moins de place qu'un index secondaire'** est correcte ou non. **Justifiez votre réponse**

3 points

2. Stockage

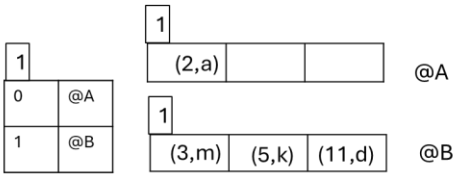
a) Soit la relation Etudiant (id_etu, nom_etu) où id_etud est la clé de la relation

On souhaite stocker cette relation en utilisant une structure de hachage extensible, sachant qu'on peut stocker jusqu'à 3 enregistrements par page, construisez la structure avec les données indiquées ci-dessous, et en ne considérant au départ que le dernier bit de la représentation binaire de la clé de la relation.

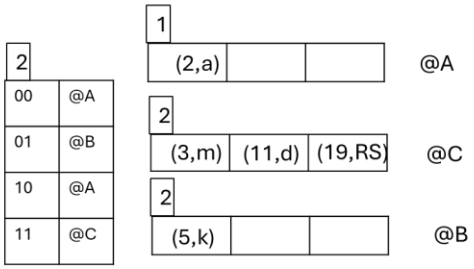
(2,a) ; (3,m) ; (5, k) ; (11, d) ; (19, RS) ; (23, yy).

2	010
3	011
4	100
5	101
11	1011
19	10011
23	10111

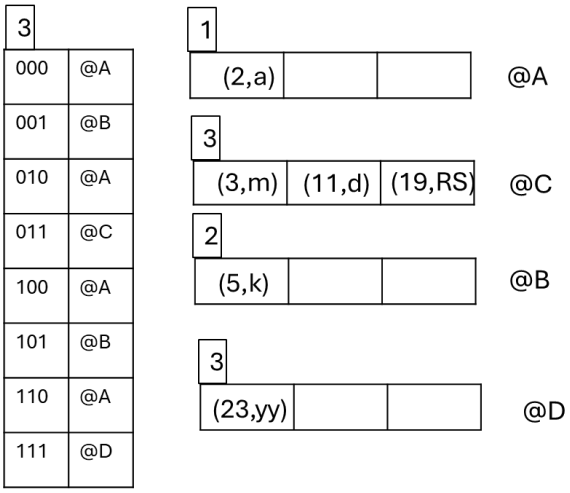
Réponse



L'insertion de (19,RS) doit être réalisée dans la page @B comment il n'y a plus de espace et @B et l'index considèrent la même quantité de bits on doit dédoubler l'index, allouer une nouvelle page et distribuer les donnes entre la page @B et la nouvelle page, on considérant les 2 derniers bits et il faut mettre à jour l'index. **Les autres pages ne sont pas touchées.**



La insertion de (23,yy) doit être réalisée dans la page @C,, il n'y a plus de espace et l'index et @C considèrent la même quantité de bits alors il faut faire la même chose que pour l'insertion de (19,RS). **Les autres pages ne sont pas touchées.**



4 points

b) Soit la base de données Etudiant, ci-dessous, contenant 200.000 étudiants de 20 universités.

Etudiant (ID, nom, adresse, téléphone, université)

La relation Etudiant est organisée en arbre B+ sur l'attribut ID, clé de la relation. On dispose d'un index secondaire organisé en arbre B+ associé à l'attribut université. Les pages de données de cet index sont composées de couples (université, ID).

On suppose que :

- L'attribut ID est stocké sur 7 octets, l'attribut nom sur 30 octets, l'attribut adresse sur 30 octets, téléphone sur 20 octets et université sur 13 octets.
- Une adresse de page est stockée sur 8 octets.
- Les pages de la base ont 520 octets.
- Les valeurs de chaque attribut suivent uniformément réparties.

Complétez le tableau des statistiques contenues dans le catalogue. **Justifiez votre réponse.**

Taille de nuplets : 100 octets
NTuples(Etudiant) = 200.000
BFactor(Etudiant) = 512/100 = 5
NBlocks((Etudiant) = 200.000/5 = 40.000 pages
NDistinct Université (Etudiant) = 20
SC $_{\text{Université}}$ (Etudiant) = 200.000/20 = 10.000

$NLevel_{ID}(I)$ = 4 (3 d'index + les données)
$NLBlocks_{ID}$ (I) = 1.143+ 33 + 1 = 1.177 pages

Où :

- NDistinct Université (Etudiant) = Nombre des valeurs distincts de l'attribut Université dans la relation Etudiant.
- SC Université (Etudiant) = Nombre d'Etudiants par université
- $NLBlocks_{ID}$ (I) = Nombre blocs de l'index sur l'attribut ID
- $NLevel_{ID}(I)$ = Hauteur de l'index s sur l'attribut ID

RELATION ETUDIANT
Chaque nuplet occupe 13+ 30+30 + 20 + 7 = 100 octets
Une page a 520 octets, les données sont stockées dans les nœuds feuilles qui sont organisés en une liste chainée donc chaque nœud doit avoir l'adresse de la page suivante donc on a 8 octets pour l'adresse de la page suivante et 512 octets disponibles pour les données. Par conséquent, chaque page peut stocker 512/ 100 = 5 nuplets.
La relation Etudiant contenant 200.000 nuplets, on peut la stocker sur 200.000/5 = 40.000 pages données.
On a 20 Universités et 200.000 Etudiants avec une distribution uniforme donc on a (SC Université (Etudiant)) 200.000/20=10.000 Etudiants dans chaque Université.

Stockage de la relation Etudiant en arbre B+ :
Les pages de données (nœuds feuilles) contiennent les enregistrements de la relation, donc on a 40.000 pages à indexer. Les nœuds non-feuilles contiennent un enregistrement avec (@ ID @ ID …@) Où @ es l'adresse d'une page du niveau inférieur), une page a 520 octets, on utilise 8 octets pour la dernière @ et 512 octets pour stocker *m* couples (@,ID) dont des couples de 7 + 8 = 15 octets par conséquent une page peut contenir 512/15 = 34 couples donc une page peut avoir 34 ID et 35@, par conséquent une page d'index peut indexer 35 pages du niveau inférieur.
Donc le deuxième niveau contient 40.000/35 = 1142,85 donc 1143 pages d'index qu'il faut indexer. 1143/35 = 32,65 donc 33 pages qu'il faut indexer. Par conséquent on a besoin d'une page pour le troisième niveau.
L'index contient donc :
· $NLevel_{ID}(I)$ = 4 (3 d'index + les données)
· $NLBlocks_{ID}(I)$ = 1.143+ 33 + 1 = 1.177 pages.

Au total pour stocker la table Etudiant on a besoin de 1177 (index) + 40.000 (données) = 41.177 pages

6 points

3. Indexation

Supposons qu'on veut créer un index secondaire organisé en arbre B+ associé à l'attribut université de la table Etudiant de l'exercice 1.b). Cet index est de type (clé de recherche, @) donc de couples (université, adresse d'enregistrement)

a) Décrivez, en français, la structure de l'index.

Il s'agit ici d'une table d'index secondaire de type (clé de recherche, adresse de l'enregistrement) sur l'attribut *universite* donc une table contenant deux attributs :

- la clé de recherche : ***universite*** et
- L'adresse d'enregistrement (@) : l'adresse d'un nuplet de la table Etudiant dont la valeur de leur attribut universite = clé de recherche (universite)

Les enregistrements (universite, @) de cette table d'index sont stockés en utilisant un arbre B+, dont

- Les nœuds feuilles de cet arbre B+ forment une liste chainée contenant des enregistrements (universite, @) où @ est l'adresse du nœud feuille suivant et
- Les nœuds non-feuilles ainsi que la racine contiennent des enregistrements type

  (@ universite @ universite @ … universite @) où chaque adresse pointe vers un nœud du niveau inférieur de l'arbre B+

3 points

b) Supposons que les statistiques contenues dans le catalogue pour cet index sont :

$NLevel_{université}$ (I) = 2 niveaux (en comptant les feuilles)
$NLBlocks_{université}$ (I) = 8000 + 1 = 8.001

Combien de pages faut-il lire pour répondre la requête ci-dessous :

```
SELECT *
FROM Etudiant
WHERE université = 'Nanterre'
```

b.1. En utilisant l'index secondaire
b.2. Sans utiliser l'index secondaire

Justifiez vos réponses

4 points

Réponse
Avec l'index

Il faut traverser l'index en comparant la clé de recherche avec la valeur Nanterre (université =Nanterre) donc il faut lire autant de pages que la hauteur de l'index (1) puis le nœud feuille qui nous donne l'adresse de la **première page** contenant des enregistrements (Université, @) dont le Universite = 'Nanterre', puis il faut récupérer tous les enregistrements de l'index avec Université = 'Nanterre' en parcourant la liste chainée de nœuds feuilles, si besoin.

On a 20 valeurs différentes d'université, on suppose que ces valeurs sont distribuées uniformément dans les valeurs de la table étudiant donc on aura 200.000/20 =10.000 étudiants pour chaque valeur différente d'université, donc pour l'université = 'Nanterre' on aura 10.000 enregistrements dans

l'index contenant les @ de nuplets résultants de la requête, par conséquence il y a 10.000 enregistrements type (Nanterre, $@_i$) dans la table d'index secondaire.

Chaque page feuille de l'index peut stocker **24** enregistrements (université, @), donc 10.000/24 = 416,66, donc pour récupérer les 10.000 enregistrements de l'index avec université = 'Nanterre' on doit lire :

- 417 dans le meilleur des cas, si dans la première page trouvée avec la valeur (Nanterre, @) il y a au moins 24 enregistrements (Nanterre, $@_i$).
- 418 dans le pire de cas, si dans la première page trouvée avec la valeur (Nanterre, @) il y a moins de 24 enregistrements (Nanterre, $@_i$).

Pour chaque enregistrement (Nanterre, $@_i$) de l'index, il faut lire les enregistrements de la table Etudiant en utilisant @i, donc il nous faut **10.000 lectures** pour récupérer les enregistrements de la table Etudiant.

Au total on a besoin de 1 + 417 (nombre de feuilles contenant les adresses des nuplets résultants) + 10.000 (lire les nuplets) = **10.418** dans le meilleur de cas ou bien 1 + 418 + 10.000= **10.419** dans le pire de cas.

Sans l'index

Il faut lire toute la table Etudiant