

Principales structures d'index

UE fondements des bases de données - TD8

1. Supposons une mémoire organisée en blocs de taille de 4k, soit 4096 octets. Un pointeur sera codé sur 8 octets. Soit une relation $FILMS(numfilm, titre, date, numstudio)$ triée sur sa clé primaire ; chaque tuple a une taille de $(8 + 16 + 16 + 8)$ octets. Calculez le nombre de blocs occupés par les objets suivants lorsque la table contient 1.000.000 de tuples, sachant que les blocs ne sont remplis qu'à 80% :
 - (a) La relation
Solution - Un bloc peut contenir 80% de 4096/48 tuples, soit 68 tuples. Donc, la relation est stockée dans $1000000/68 = 14706$ blocs.
 - (b) Un index primaire dense sur les numéros de films
L'index est primaire puisque la table est triée sur les numéros de films. Puisqu'il est dense, il possède une paire (valeur,pointeur) pour chacune des n valeurs de $numfilm$. Une telle paire a une taille de $(8+8)=16$ octets ; ainsi, un bloc peut contenir $80\% \times 4096/16 = 204$ paires. Sa taille est donc de $1000000/204 = 4902$ blocs.
 - (c) Un index primaire creux sur les numéros de films
Solution - L'index est creux, il possède donc une paire (valeur,pointeur) pour chaque bloc du fichier de la relation. Donc sa taille est de $14706/204 = 70$ blocs.
2. Combien de chargement de blocs faut-il effectuer au plus pour retrouver une valeur dans la relation sans index ? Avec l'index primaire creux ?
Solution - Retrouver une valeur, c'est effectuer une recherche dichotomique sur les blocs puisque aussi bien la table que l'index est triée. Donc pour la table sans index : $\log_2(14706) = 14$ blocs chargés. Avec index : $\log_2(70) = 7$ blocs, plus celui contenant le tuple cherché, donc 6 blocs.
3. Même question avec une deuxième niveau sur cet index.
Solution - Un index (forcément creux) sur l'index contient une paire (valeur,pointeur) pour chaque bloc de l'index, soit 70 ici. Ce sur-index est donc contenu dans un seul bloc. A partir de ce bloc, on charge le bloc nécessaire de l'index lui-même, et enfin le bloc du tuple. On aura donc 3 chargements ; toutefois l'index de deuxième niveau, constitué d'un seul bloc, sera en pratique gardé en mémoire principale, conduisant à 2 blocs seulement chargés en mémoire ce qui divise par 7 le temps d'accès par rapport à la relation sans index.
4. On suppose qu'il y a 80% de valeurs distinctes dans les titres. Calculez le nombre de blocs d'un index sur le titre. N'oubliez pas le "bucket" utilisé pour chaque valeur ; rappelons qu'un bucket est une liste de pointeurs elle même organisée en blocs. On supposera que les blocs stockant les buckets sont pleins.
Solution - Il y a 800.000 titres distincts dans la relation, donc autant de couples (valeur,pointeur) dans l'index (étant secondaire, il est forcément dense). Chaque couple a une taille de $16 + 8 = 24$ octets. Un bloc contient $80\% \times 4096/24 = 137$ paires. L'index est donc composé de $800000/137 = 5840$ blocs. En outre, dans les buckets il y a 1000000 de pointeurs, soit 8000000 d'octets, ce qui correspond à $8000000/4096 = 1954$ blocs. Au total, l'index occupe 7794 blocs.
5. Construire un B-Arbre pour les valeurs suivantes : 49, 17, 21, 16, 34, 42, 81, 39, 41, 27, 1, 5, 83, 6.
 - Considérer $N=3$, c'est-à-dire que chaque bloc contient au plus 3 valeurs et 4 pointeurs
 - Insérer les valeurs une à une
 - Faire un nouveau dessin pour chaque nouveau nœud inséré.**Solution - La solution sera donnée en TD.**

6. On se replace dans les conditions de la question 1. Soit un B-arbre sur l'attribut *NumFilm* que l'on supposera dense. Quel est le nombre de niveaux dans l'arbre ? Quel est le nombre de blocs à charger pour retrouver une valeur de la clé ? Notez que chaque nœud contient au moins $\lceil N/2 \rceil$ pointeurs, et donc $\lceil N/2 \rceil - 1$ valeurs.

Solution - Il faut calculer N , c'est à dire le nombre maximal de valeurs dans un nœud. Un nœud contient également de la place pour $N + 1$ pointeurs. On a donc l'inéquation $8N \times 8(N + 1) \leq 4096$. On trouve $16N \leq 4088$ soit $N \leq 255,5$, soit $N = 255$. Dans le cas le plus "gros", on aura alors $\lceil 255/2 \rceil = 128$ pointeurs et 127 valeurs par nœud.

Ainsi, le nombre de bloc-feuilles sera de $1000000/127 = 7875$. Puisque chaque nœud intermédiaire pointe sur 128 blocs, on divise par 128 le nombre de blocs en montant d'un niveau ; ainsi, le nombre de niveaux sera de $\log_{128}(1000000) = 3$.

Une recherche dans l'arbre correspond à une descente, soit 3 chargements de blocs, plus 1 pour accéder au bon tuple. Ici encore, la racine de l'arbre est gardée en mémoire en générale.