LIFBDW2 –Bases de données avancées
TD9 – Principales structures d'index

Licence informatique – Automne 2016–2017

Les questions marquées du symbole (†) sont à préparer pour la séance

Exercice 1 : Structures simples

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% :

1. La relation
2. Un index primaire dense sur les numéros de films
3. Un index primaire creux sur les numéros de films
4. Combien de chargements de blocs faut-il effectuer au plus pour retrouver une valeur dans la relation sans index ? Avec l'index primaire creux ?
5. Même question avec un deuxième niveau sur cet index.
6. 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.

Exercice 2 : B-Arbre

1. Construire un B-Arbre pour les valeurs suivantes : $49, 17, 21, 16, 34, 42, 81, 39, 41, 27, 1, 5, 83, 6, 100, 51$
   — Considérer N=3 (un nœud contient 3 valeurs, 4 pointeurs).
   — Faire un nouveau dessin pour chaque nouveau nœud inséré.

Corrections

Solution de l'exercice 1

1. Un bloc peut contenir 80% de $\frac{4096}{(8+16+16+8)=48}$ tuples, soit 68 tuples. Donc la relation est stockée dans $\lceil \frac{10^6}{68} \rceil = 14706$ blocs.
2. L'index est dit primaire car la table est triée sur les numéros de films (i.e., sa clé primaire). 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 $\lfloor 0.8 \times \frac{4096}{16} \rfloor = 204$ paires. Sa taille est donc de $\lceil \frac{10^6}{204} \rceil = 4902$ blocs.
3. L'index est creux, il possède donc une paire *(valeur, pointeur)* pour chaque bloc du fichier de la relation. Donc sa taille est de $\lceil \frac{14706}{204} \rceil = 73$ blocs.
4. 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 l'index creux, $\log_2(73) = 7$ blocs, plus celui contenant le tuple cherché, donc 8 blocs.
5. Un index (forcément creux) sur l'index contient une paire *(valeur, pointeur)* pour chaque bloc de l'index, soit 73 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 centrale, conduisant à 2 blocs seulement chargés en mémoire, ce qui divise par 7 le temps d'accès par rapport à la relation sans index.
6. 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 de $(16+8) = 24$ octets. Un bloc contient $\lfloor 0.8 \times \frac{4096}{24} \rfloor = 136$ paires. L'index est donc composé de $\lceil \frac{800.000}{136} \rceil = 5883$ blocs. En outre, dans les buckets, il y a 1.000.000 de pointeurs soit 8.000.000 d'octets, ce qui correspond à $\lceil \frac{8.000.000}{4096} \rceil = 1954$ blocs. Au total, l'index occupe 7837 blocs.

Solution de l'exercice 2

1. Voir feuille scannée, figure . Il est important de remarquer la croissance d'un B-arbre par la création d'une nouvelle racine.

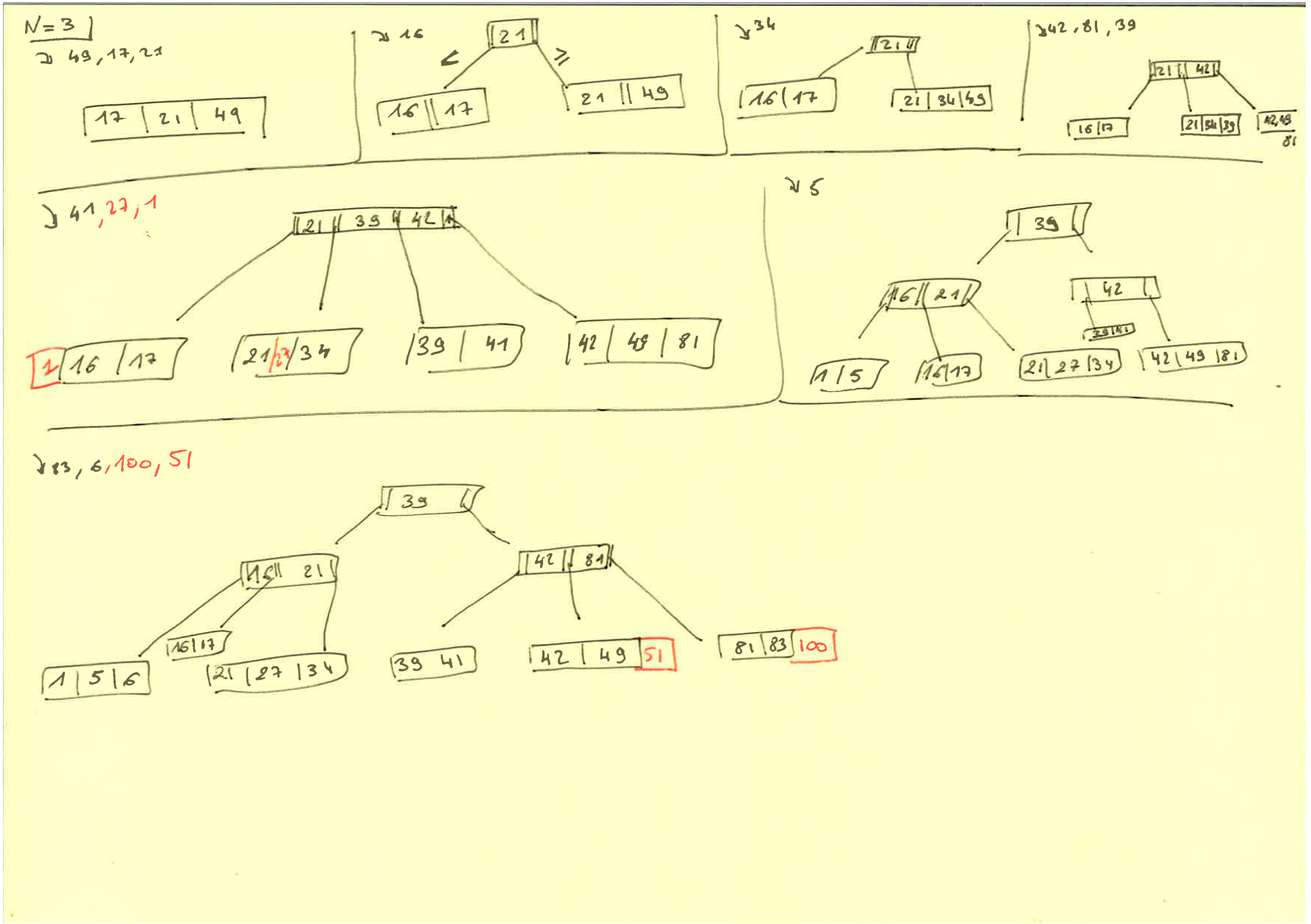

FIGURE 1 – B-Arbres pour l'exercice 2