1 Techniques de stockage

Caractéristique	Performance
Taille d'un secteur	512 octets
Nbre de plateaux	5
Nbre de têtes	10
Nombre de cylindres	100 000
Nombre de secteurs par piste	4 000
Temps de positionnement moyen	10 ms
Vitesse de rotation	7 400 rpm
Déplacement de piste à piste	0,5 ms
Taux de transfert max.	120 Mo/s

TABLE 1 – Spécifications d'un disque magnétique

Exercice 1. *Le tableau 1 donne les spécifications partielles d'un disque magnétique. Répondez aux questions suivantes.*

1. *Quelle est la capacité d'une piste ?, d'un cylindre ? d'une surface ? du disque ?*
2. *Quel est le temps de latence maximal ? Quel est le temps de latence moyen ?*
3. *Quel serait le taux de transfert (en Mo/sec) nécessaire pour pouvoir transmettre le contenu d'une piste en une seule rotation ? Est-ce possible ?*

Solution :

1. – Capacité d'une piste = nbre de secteurs par piste x taille d'un secteur = 4 000 x 512 = 2 048 000 octets (soit approx 2 Mo)
   – Capacité d'un cylindre = taille d'une piste x nombre de têtes = 2 048 000 x 10 = 20 480 000 octets (soit approx 20 Mo)
   – Capacité d'une surface = capacité d'une piste x nombre de cylindres = 2 048 000 x 100 000 = 204 800 000 000 octets (soit approx 204 Go)
   – Capacité du disque = capacité d'un cylindre x nombre de cylindre = 20 480 000 x 100 000 octets = approx 2 To
2. Le temps de latence maximal est le temps pour une rotation. Ici on a $\frac{7400}{60} = 123$ rotations par seconde, soit une rotation toutes les 1/123=8 ms. Le temps moyen est de 4 ms.
3. Il faut donc pouvoir transférer 2 048 000 octets en 8 ms, soit 250 Mo par seconde. Ce taux dépasse les capacités de transfert du disque.

Exercice 2. *Étant donné un disque contenant C cylindres, un temps de déplacement entre deux pistes adjacentes de sms ms, donner la formule exprimant le temps de positionnement moyen. On décrira une demande de déplacement par une paire $[d, a]$ où d est la piste de départ et a la piste d'arrivée, et on supposera que toutes les demandes possibles sont équiprobables.*

Attention, on ne peut pas considérer que les têtes se déplacent en moyenne de $C/2$ pistes. C'est vrai si la tête de lecture est au bord des plateaux ou de l'axe, mais pas si elle est au milieu du plateau.

Il faut commencer par exprimer le temps de positionnement moyen en fonction d'une position de départ donnée, puis généraliser à l'ensemble des positions de départ possibles.

On pourra utiliser les deux formules suivantes :

1. $\Sigma_{i=1}^{n}(i) = \frac{n\times(n+1)}{2}$
2. $\Sigma_{i=1}^{n}(i^2) = \frac{n\times(n+1)\times(2n+1)}{6}$

et commencer par exprimer le nombre moyen de déplacements en supposant que les têtes de lecture sont en position p. Il restera alors à effectuer la moyenne de l'expression obtenue, pour l'ensemble des valeurs possibles de p.

Solution :

On commence par calculer le temps en supposant connue la position courante des têtes de lecture, p, avec $1 < p < C$. Le nombre moyen de déplacements à effectuer vers une piste i avec $1 \leq i < p$ est :

$$\frac{1 + 2 + \ldots + (p-1)}{p-1} = \frac{p \times (p-1)}{2 \times (p-1)} = \frac{p}{2}$$

De même, le nombre moyen de déplacements à effectuer vers une piste j avec $p \leq j \leq C$ est :

$$\frac{1 + 2 + \ldots + (C-p-1)}{C-p} = \frac{(C-p) \times (C-p+1)}{2 \times (C-p)} = \frac{C-p}{2}$$

Noter que pour $p = C/2$, le nombre moyen de déplacements, dans un sens ou dans l'autre, est de $C/4$, alors que quand $p = 1$, le temps de déplacement moyen est de $C/2$.

Pour une position p donnée, la probabilité d'aller en $i < p$ est p/C, et celle d'aller en $j > p$ est $(C-p)/C$. On obtient donc le nombre de déplacements moyen pour une position p :

$$\frac{p^2}{2 \times C} + \frac{(C-p)^2}{2 \times C}$$

Il reste à effectuer la moyenne de cette expression pour l'ensemble des positions de départs possibles, ce qui donne :

$$\frac{1}{C} \times \Sigma_{i=1}^{C}(\frac{i^2 + (C-i)^2}{2 \times C}) = \frac{1}{2 \times C^2} \times 2 \times \Sigma_{i=1}^{C}(i^2) = \frac{1}{C^2} \times \frac{C \times (C+1) \times (2C+1)}{6}$$

On en déduit que le nombre de déplacements moyen est de l'ordre du tiers du nombre total de pistes (ou cylindres).

Exercice 3. *Soit un disque de 5 000 cylindres tournant à 12 000 rpm, avec un temps de déplacement entre deux pistes adjacentes égal à 0,2 ms et 500 secteurs de 512 octets par piste. Quel est le temps moyen de lecture d'un bloc de 4 096 ?*

Solution :

Calculer le temps de transfert (8/400 de rotation).

Calculer le délai de latence : 12000 rotations pour 60s, donc une rotation en 60/12000= 5 ms. Temps moyen de latence=2,5 ms.

Temps moyen de positionnement : environ 1/3 du temps total de balayage du disque, soit $\frac{0,5\times5000}{3} = 1/3s$

Exercice 4. *On considère un fichier de 1 Go et un buffer de 100 Mo.*

- *quel est le hit ratio en supposant que la probabilité de lire les blocs est uniforme ?*
- *même question, en supposant que 80 % des lectures concernent 200 Mo, les 20 % restant étant répartis uniformément sur 800 Mo ?*
- *avec l'hypothèse précédente, jusqu'à quelle taille de buffer peut-on espérer une amélioration significative du hit ratio ?*

Solution :

- Pour la première question le *hit ratio* est évidemment de 0,1.
- Appelons A la partie de la base correspondant aux 200 Mo les plus lus, et B le reste. On remarque tout d'abord que le *cache* contient 80 % d'enregistrements de A, et 20 % d'enregistrements de B.
  Si on fait une lecture d'un enregistrement de A, le *hit ratio* est de $\frac{80}{200}$. Pour B, le *hit ratio* est de $\frac{20}{800}$. On pondère ces valeurs par la probabilité de demander la lecture, respectivement, d'un enregistrement A ou B, soit :
  $$0,8 \times \frac{80}{200} + 0,2 \times \frac{20}{800}$$
  Ce qui est déjà mieux de que le 0,1 obtenu à la première question.
- Jusqu'où peut-on espérer une amélioration ? Notons C la taille du cache. La formule qui donne le *hit ratio* est :
  $$0,8 \times \frac{Max(|A|, 0,8 \times C)}{200} + 0,2 \times \frac{Max(800, 0,2 \times C)}{800}$$
  Une fois que toutes les données de A (ou de B) sont en mémoire, le *hit ratio* ne s'améliore plus. Le premier terme de l'équation bénéficie rapidement d'un accroissement de C, jusqu'à $C = 250$. Tout le fichier A sera alors en mémoire. Ensuite, la petite amélioration du *hit ratio* en fonction de C devient beaucoup plus faible.

Exercice 5. *Soit l'ordre SQL suivant :*

```
DELETE FROM Film
WHERE condition
```

La table est stockée dans un fichier de taille n blocs, sans index. Donner le nombre moyen de blocs à lire, en fonction de n, pour l'exécution de l'ordre SQL dans les cas suivants :

- *aucun enregistrement ne satisfait la condition ;*
- *la condition porte sur une clé unique et affecte donc un seul enregistrement ;*
- *la condition ne porte pas sur une clé unique (et on ne connaît donc pas à priori le nombre d'enregistrements concernés).*

Exercice 6. *Soit la table de schéma suivant :*

```
CREATE TABLE Personne (id     INT NOT NULL,
                       nom    VARCHAR(40) NOT NULL,
                       prenom VARCHAR(40) NOT NULL,
                       adresse VARCHAR(70),
                       dateNaissance DATE)
```

Cettte table contient 300 000 enregistrements, stockés dans des blocs de taille 4 096 octets. Un enregistrement ne peut pas chevaucher deux blocs, et chaque bloc comprend un entête de 200 octets.

1. *Donner la taille maximale et la taille minimale d'un enregistrement. On suppose par la suite que tous les enregistrements ont une taille maximale.*
2. *Quel est le nombre maximal d'enregistrements par bloc ?*
3. *Quelle est la taille du fichier ?*
4. *En supposant que le fichier est stocké sur le disque de l'exercice 1, combien de cylindres faut-il pour stocker le fichier ?*
5. *Quel est le temps moyen de recherche d'un enregistrement dans les deux cas suivants : (a) les blocs sont stockés le plus contigument possible et (b) les blocs sont distribués au hasard sur le disque.*
6. *On suppose que le fichier est trié sur le nom. Quel est le temps d'une recherche dichotomique pour chercher une personne avec un nom donné ?*

Solution :

1. Taille minimale : 4 + 1 + 1 + 0 (null) + 0 (null). Taille maximale : $4 + 41 + 41 + 71 + 8 = 165$.
2. Espace disponible pour les enregistrements : $4096 - 200 = 3896$. Nombre max. d'enregistrements par bloc : $\lfloor 3\,896/165 \rfloor = 23$
3. Taille du fichier : $\lfloor 300\,000/23 \rfloor = 13\,044$ blocs, soit $13\,044 \times 4\,096 = 53$ Mo.
4. Un cylindre contient 20 Mo (soit 5 000 blocs ici) donc 3 cylindres, par exemple 5 000 (cylindre 1) + 5 000 (cylindre 2) + 3 044 (cylindre 3). NB : un piste peut contenir 2 048 000 (capacité piste) / 4 096 (taille bloc) = 500 blocs. Le cylindre 3 n'est pas "plein", on n'utilise que 7 pistes de ce cylindre.
5. En supposant que le fichier est stocké le plus continûment possible : on a :
   (a) positionnement sur le premier cylindre = temps de positionnement moyen) : 10ms
   (b) temps de lecture d'un cylindre (cylindre 1) = 10 pistes / cylindre x 8 ms (temps lecture piste) : 80 ms
   (c) positionnement sur le second cylindre = temps de déplacement de piste à piste = 0,5 ms
   (d) temps de lecture d'un cylindre (cylindre 2) = 10 pistes / cylindre x 8 ms (temps lecture piste) : 80 ms
   (e) positionnement sur le second cylindre = temps de déplacement de piste à piste = 0,5 ms
   (f) temps de lecture d'un cylindre (cylindre 3) = 7 pistes / cylindre x 8 ms (temps lecture piste) : 56 ms

   i+ii+iii+iv+v+vi=227ms, soit approx 0,23 s

   Si les blocs sont distribués au hasard, il faut 10 (temps de positionnement moyen) x 4 (temps de latence moyen) = 14 ms en moyenne pour lire chaque bloc. Pour les 13 044 blocs : 13 044 x 14 = 182 616 ms = approx 183 secondes !
6. Avec une recherche par dichotomie, il faut $log_2(13044)$ = approx 14 lectures de blocs, soit 14 (nb lectures) x 14 ms (temps lecture moyen) = 196 ms.

Exercice 7. *On considère un disque composé de C cylindres qui doit satisfaire un flux de demandes d'entrées-sorties de la forme $[t, a]$ où t est l'instant de soumission de la demande et a l'adresse. On suppose que le SGBD applique de deux techniques de séquencement des entrées-sorties. La première, nommée* PDPS *(premier demandé premier servi) est classique. La second,* balayage *se décrit comme suit :*

- *les demandes d'entrées-sorties sont stockées au fur et à mesure de leur arrivée dans un tableau T_{es} à C entrées ; chaque entrée est une liste chaînée stockant, dans l'ordre d'arrivée, les demandes concernant le cylindre courant ;*
- *les entrées du tableau T_{es} sont balayées séquentiellement de 1 à C, puis de C à 1, et ainsi de suite ; quand on arrive sur une entrée $T_{io}[c]$, on place les têtes de lecture sur le cylindre c, et on effectue les entrées/sorties en attente dans l'ordre de la liste chaînée.*
  On supposera qu'on traite les demandes connues au moment où on arrive sur le cylindre,

L'idée de balayage *est bien entendu de limiter les déplacements importants des têtes de lecture. On suppose que le disque a 250 cylindres, effectue une rotation en 6 ms, et qu'un déplacement entre deux pistes adjacentes prend 0,2 ms.*

1. *On suppose que le flux des entrées/sorties est le suivant :*

$$d_1 = [1, 100], d_2 = [15, 130], d_3 = [17, 130], d_4 = [25, 15],$$
$$d_5 = [42, 130], d_6 = [27, 155], d_7 = [29, 155], d_8 = [70, 250]$$

   Indiquez à quel moment chaque entrée/sortie est effectuée (attention : une E/S ne peut être traitée avant d'être soumise !). L'adresse est ici simplement le numéro de cylindre et les instants sont exprimés en valeur absolue et en millisecondes. On suppose que les têtes de lectures sont à l'instant 1 sur le cylindre 1.

2. *Donnez les instants où sont effectuées les entrées/sorties avec l'algorithme classique PDPS (contraitement à balayage, on traite donc les demandes dans l'ordre où elles sont soumises).*

3. *Mêmes questions avec la liste suivante :*

$$[1, 100], [10, 130], [15, 10], [20, 180], [25, 50], [30, 250]$$

4. *Soit la liste d'E/S suivante :*

$$[1, c_1], [2, c_2], [3, c_3], \ldots, [100, c_{100}]$$

   Supposons que le disque vienne de traiter la demande $d_1 = [1, c_1]$. Donner le temps maximal pour que la demande $d_2 = [2, c_2]$ soit traitée, avec les deux approches (balayage et premier demandé-premier servi).

5. *Soit $n > 250$ le nombre de demandes en attente. On suppose qu'elles sont également réparties sur tous les 250 cylindres. Quel est le temps moyen d'accès à un bloc avec l'agorithme de balayage, exprimé en fonction de n ?*

<u>*Solution :*</u>

1. On note d'abord que le délai de latence moyen est de 3 ms. Pour la première entrée/sortie, on fait donc 100 déplacements (20 ms) et 3 ms, soit 23 ms.

   Une fois qu'on a effectué la première E/S, on s'intéresse à la seconde. Comme on est à l'instant 23, on traite d_2 et d_3 ce qui prend $6 + 3 + 3 = 12ms$, et on est à l'instant $23 + 12 = 35$.

   Ensuite on traite d_6 et d_7. Le temps de déplacement est de 5 ms, donc ça prend 11 ms. On se retrouve à 44 ms.

   On traite d_8 en parcourant 95 pistes, soit 19 ms, plus 3 ms = 21 ms. On se retrouve à l'instant $44 + 21 = 65$.

On revient sur d_5, la demande de la piste 130 soumise à $t = 42$. Il faut parcourir 120 pistes, en 24 ms, et effectuer la lecture. On est à l'instant $65 + 24 = 89ms$.

Il reste d_4 qui prend $23 + 3 = 26ms$ à satisfaire. Finalement on est à $t = 115$.

2. Comptons les temps de déplacement : $20+6+23+23+5+19 =$, soit $96ms$, plus $8_{\times}3 = 24ms$ de temps de latence. Donc ça prend $120ms$: on n'a pas gagné énormément sur cet exemple.
3. $$[1, 100], [10, 130], [15, 10], [20, 180], [25, 50], [30, 250]$$

   Cas 1, déplacements : $20 + 6 + 10 + 14 + 40 + 8 = 98ms$

   Cas 2, déplacements : $20 + 6 + 24 + 34 + 26 + 40 = 150ms$

   La séquence comprend ici beaucoup d'alternances, donc l'algorithme de balayage est préférable.
4. Dans le cas le plus défavorable, le cylindre c_2 est juste *avant* le cylindre c_1 dans le sens courant de balayage, et il faut attendre, dans le pire des cas, que les têtes de lecture aient effectué un aller-retour complet, soit $500 \times 0,2 = 100ms + 98 \times 3 = 294ms$ de temps de latence au pire. Donc $394ms$ de latence, alors qu'on a au pire $250 \times 0,2 + 3 = 53ms$ avec l'algorithme classique.
5. On a $n/250$ demandes par cylindre. Pour effectuer une E/S, il faut le délai de latence de 3 ms, et se déplacer d'une piste à une autre 1 fois sur $n/250$, soit $250/n$ fois (en moyenne). le temps de lecture moyen est donc de

   $$3 + \frac{250}{n} \times 0,2ms$$

   Ce temps est inversement proportionnel à n, ce qui montre bien que plus n est important, et plus le gain du balayage est important en terme de temps moyen d'accès.

2 Indexation

Organisation Physique

Exercice 8.

On prend ici comme exemple la relation **Directeur** *(*nom_directeur, nom_film*).*

1. *Organisation Séquentielle : Expliquer l'organisation séquentielle et représenter un exemple sur la relation Directeur. Montrer le fichier aprés une insertion et aprés quelques suppressions d'articles.*

   Solution :

   Les articles sont stockés les uns derrière les autres dans des pages successives. L' avantage de cette structure physique est sa simplicité. En particulier, il n'existe pas d'ordre entre les articles. La seule façon d'accéder à un article est par *balayage séquentiel* : on parcourt le fichier en séquence, page par page : chaque page est lue en mémoire : les articles dans la page sont alors parcourus séquentiellement. On lit chaque article et le sélectionne si la valeur de ses champs correspond au(x) critËre(s) de recherche. Un exemple d'une organisation séquentielle pour la relation **Directeur** se trouve dans la Figure 1

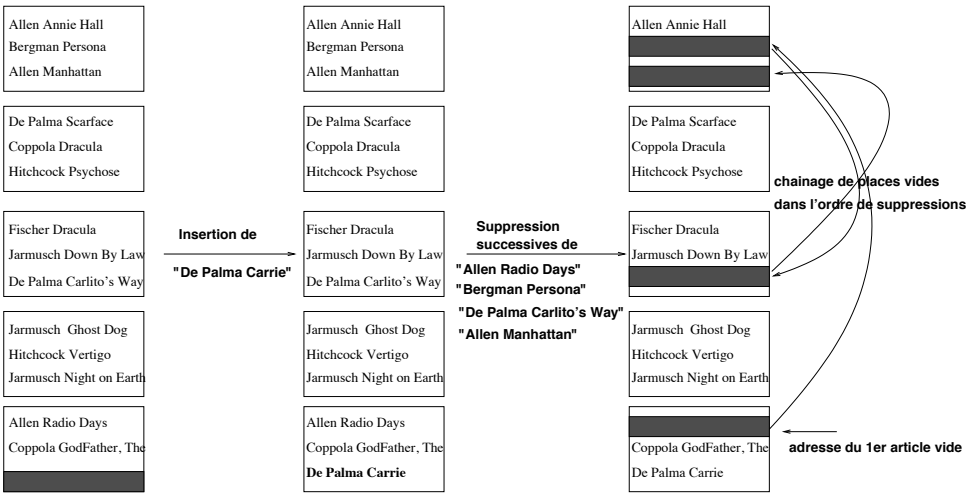


FIGURE 1 – Organisation Séquentielle de la relation **Directeur**

2. *Organisation Indexée : Montrer des exemples d'index non dense (primaire) et dense (secondaire) sur la relation Directeur.*

   Solution :

   On illustre dans les Figures 2 et 3 des exemples d'index non dense respectivement sur l'attribut directeur et sur l'attribut nom de film. L'index est ordonnÈ sur le même attribut que le fichier. Dans la figure 4 on montre deux index denses, l'un sur l'attribut directeur l'autre sur le nom du film. L'index est trié sur la clé alors que l'ordre des articles dans le fichier est quelconque. Deux remarques sont importantes : on a supposé dans ces exemples que l'index tient dans une page (feuille unique). Il faut se rappeler qu'en pratique, l'index a une structure arborescente (Arbre B et des tris). Ensuite, l'exemple dans la figure 4 suppose qu'on associe à une valeur de clé "c" dans l'index une adresse de page (où se trouvent un ou plusieurs nuplets ayant "c" pour valeur de l'attribut clé). En fait la plupart du temps, ce n'est pas une adresse de page, mais une adresse complète du nuplet (adresse de page et adresse dans la page, voir cours) qui est associée à "c" dans l'index. Ceci a pour conséquence, que s'il y a "n" nuplets dans la page ayant pour valeur de clé "c", il y aura "n" couples ("c, adresse de nuplet) dans l'index.

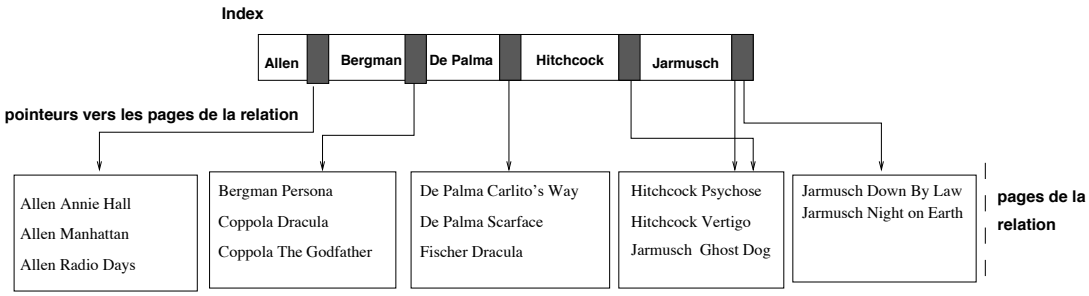


FIGURE 2 – Index non dense sur l'attribut *nom_directeur* de la relation **Directeur**

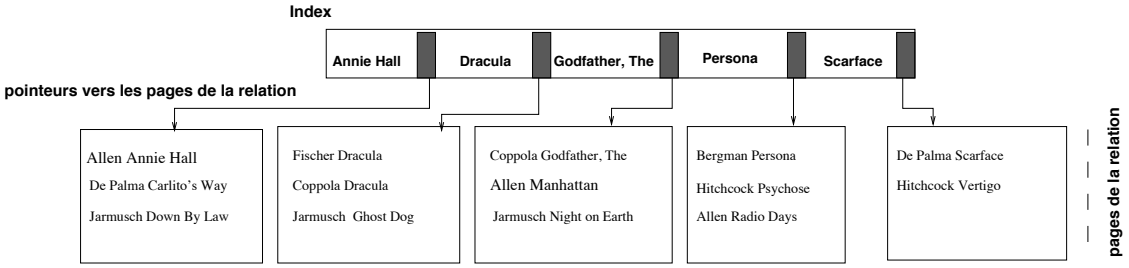


FIGURE 3 – Index non dense sur l'attribut *nom_film* de la relation **Directeur**

Exercice 9. *Soit l'arbre B+ de la figure 5, avec 4 entrées par bloc au maximum. Expliquez les opérations suivantes, et donnez le nombre d'entrées-sorties de blocs nécessaires.*

1. *Recherche de l'enregistrement 41.*
2. *Recherche des enregistrements 17 à 30.*
3. *Insertion des enregistrements 1 et 4.*
4. *Destruction de l'enregistrement 23.*

Exercice 10. *Soit la liste des départements suivants.*

```
3  Allier
36 Indre
18 Cher
75 Paris
39 Jura
9  Ariège
81 Tarn
11 Aude
12 Aveyron
25 Doubs
73 Savoie
55 Meuse
15 Cantal
51 Marne
42 Loire
40 Landes
14 Calvados
30 Gard
84 Vaucluse
7 Ardèche
```

1. *Construire, en prenant comme clé le numéro de département, un index dense à deux niveaux sur le fichier contenant les enregistrements dans l'ordre donné ci-dessus, en supposant 2 enregistrements par bloc pour les données, et 8 par bloc pour l'index.*
2. *Construire un index non-dense sur le fichier trié par numéro, avec les mêmes hypothèses.*

FIGURE 4 – Index dense

3. *Construire un arbre-B sur les noms de département, en supposant qu'il y a au plus 4 enregistrements par bloc, et en prenant les enregistrements dans l'ordre donné ci-dessus.*
4. *On suppose que les départements sont stockés dans un fichier séquentiel, dans l'ordre donné, et que le fichier de données contient deux enregistrements par bloc. Construisez un arbre-B+ sur le nom de département avec au plus 4 entrées par bloc.*
5. *Quel est le nombre de lectures nécessaires, pour l'arbre-B puis pour l'arbre-B+, pour les opérations suivantes, :*
   (a) *rechercher le Cher ;*
   (b) *rechercher les départements entre le Cantal et les Landes (donner le nombre de lectures dans le pire des cas).*

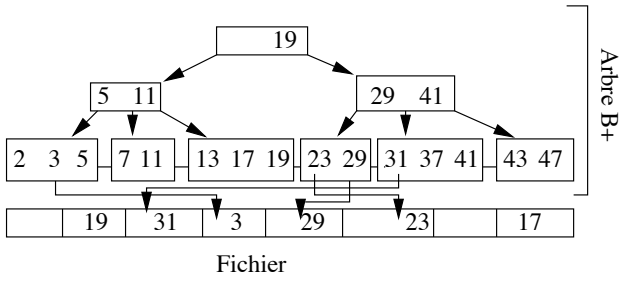


FIGURE 5 – Un arbre B+

Solution :

1. Voir Figure 6.

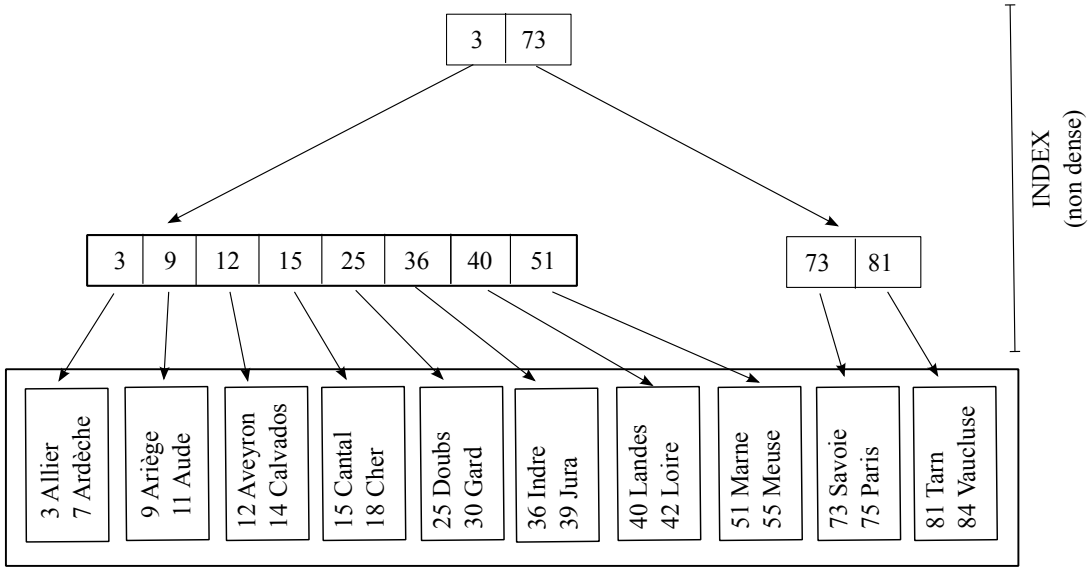


FIGURE 6 – Index non dense sur les départements.

2. Voir Figure 7.
3. Voir Figure 8.

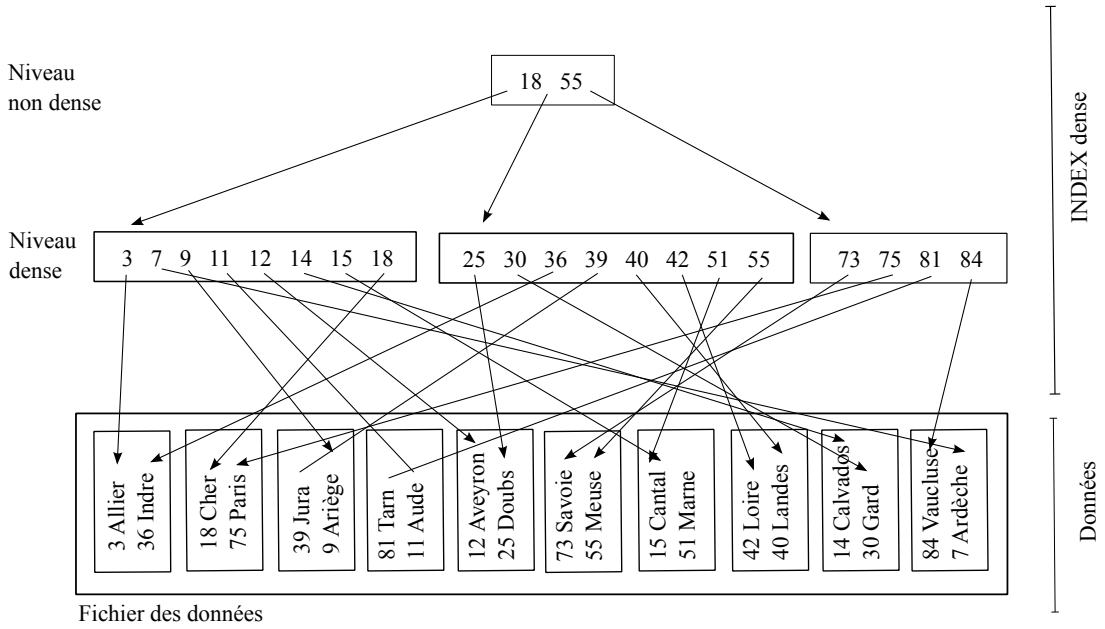


FIGURE 7 – Index dense sur les départements.

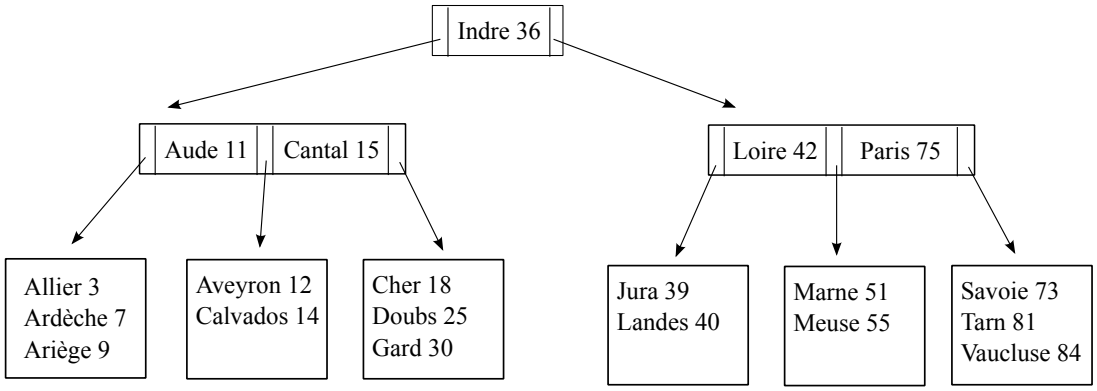


FIGURE 8 – Arbre B sur les départements.

4. Voir Figure 9.

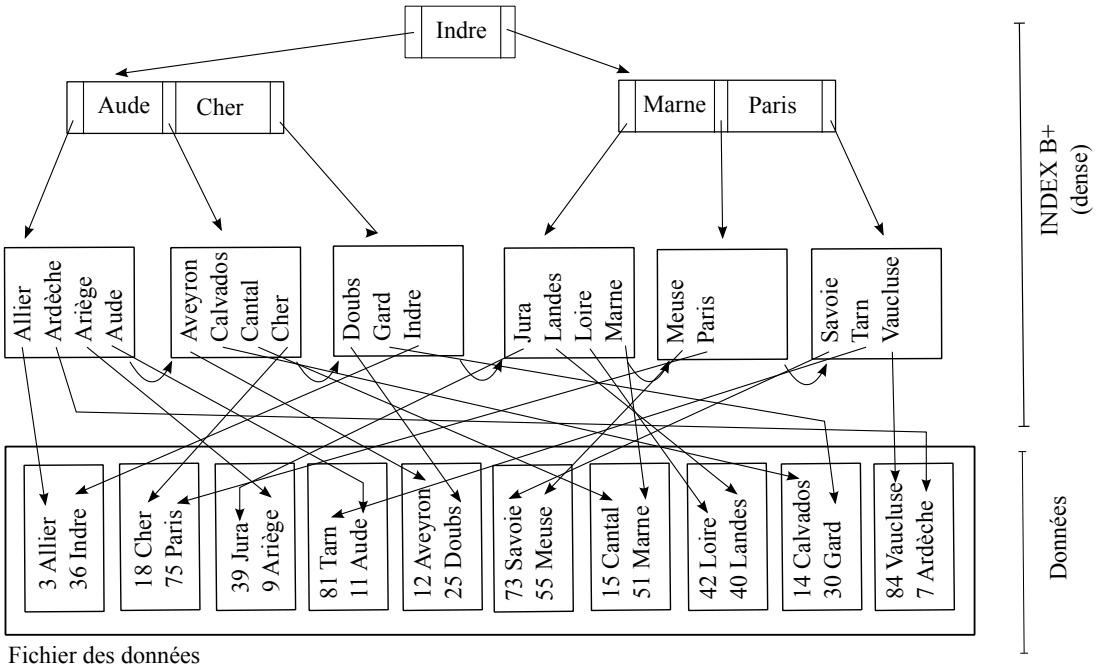


FIGURE 9 – Arbre Bplus sur les départements.

Exercice 11 (Index dense et non-dense)**.** *Soit un fichier de données tel que chaque bloc peut contenir 10 enregistrements. On indexe ce fichier avec un niveau d'index, et on suppose qu'un bloc d'index contient 100 entrées* $[valeur, adresse]$.

Si n *est le nombre d'enregistrements, donnez la fonction de* n *permettant d'obtenir le nombre minimum de blocs pour les structures suivantes :*

1. *Un fichier avec un index dense.*
2. *Un fichier trié sur la clé avec un index non-dense.*

Solution :

(1) $n/10$ et $n/100$, soit $\frac{11\times n}{100}$ blocs. (2) Il faut seulement $n/10$ paires (clé, pointeur), donc $n/1000$ blocs, pour un total de $\frac{101\times n}{1000}$ blocs.

Exercice 12 (Arbre-B+)**.** *On reprend les hypothèses précédentes, et on indexe maintenant le fichier avec un arbre-B+. Les feuilles de l'arbre contiennent donc des pointeurs vers le fichier, et les nœuds internes des pointeurs vers d'autres nœuds. On suppose qu'un bloc d'arbre B+ est plein à 70 % (69 clés, 70 pointeurs).*

1. *Le fichier est indexé par un arbre-B+ dense. Donnez (1) le nombre de niveaux de l'arbre pour un fichier de 1 000 000 articles, (2) le nombre de blocs utilisés (y compris l'index), et (3) le nombre de lectures pour rechercher un article par sa clé.*

   Solution :
   Il y a 100 000 blocs de données. Un million d'articles, donc $\frac{1M}{70} = 14\,286$ blocs pour le dernier niveau de l'arbre B+. On divise encore par 70 : 204 blocs au niveau $l-1$, 3 blocs au niveau supérieur, et 1 bloc pour la racine.
   Il faut donc lire 4+1 blocs pour accéder à un article par sa clé.

2. *On effectue maintenant une recherche par intervalle ramenant 1 000 articles. Décrivez la recherche et donnez le nombre de lectures dans le pire des cas.*

   Solution :
   Recherche sur le plus petit élément (4 blocs), puis lecture de $\frac{1000}{70} = 14$ blocs en suivant le niveau des feuilles de l'arbre B+. Pour chaque pointeur vers un objet il faut lire un bloc dans le pire des cas, soit 1 000 blocs. Total : 1018 blocs.

3. *Mêmes questions que (1), mais le fichier est trié sur la clé, et l'arbre-B+ est non dense.*

   Solution :
   Il faut indexer 100 000 blocs. Donc 1 429 blocs au dernier niveau, puis $\lceil \frac{1429}{70} \rceil = 21$, puis 1. On économise un niveau.
   Pour la recherche par intervalle, on tire parti du fichier trié : 3 lectures dans l'index, + $\lceil 1000/10 \rceil +$ 1 dans le fichier.

Exercice 13. *Soit un fichier de 1 000 000 enregistrements répartis en pages de 4 096 octets. Chaque enregistrement fait 45 octets et il n'y a pas de chevauchement de blocs. Répondez aux questions suivantes en justifiant vos réponses (on suppose que les blocs sont pleins).*

1. *Combien faut-il de blocs ? Quelle est la taille du fichier ?*

   Solution :

   (a) Dans un bloc on met $\lfloor \frac{4096}{45} \rfloor = 91$
   (b) $\lceil \frac{1000000}{91} \rceil = 10990$ blocs, donc 45 Mo.

2. *Quelle est la taille d'un index de type arbre-B+ si la clé fait 32 octets et un pointeur 8 octets ? Même question si la clé fait 4 octets.*

   Solution :
   Une entrée d'index fait 40 octets. Donc on en met $\frac{4096}{40} = 102$ clés par bloc et 103 adresses. Il faut indexer 1 000 000 enregistrements pour le dernier niveau de l'index, soit $\frac{1\,000\,000}{103} \simeq 10\,000$ pages. Ensuite il faut indexer chacun des 10 000 blocs, soit $\frac{10\,000}{100} = 100$ blocs supplémentaires, que pour finir on indexe avec la racine.
   Donc il faut 10 000 + 100 + 1 blocs dans l'index.

3. *Si on suppose qu'un E/S coûte 10 ms, quel est le coût moyen d'une recherche d'un enregistrement par clé unique, avec index et sans index ?*

   Solution :

Recherche avec index : trois lectures dans l'index, puis une lecture dans le fichier. Soit 40 ms.

Sans index, en moyenne il faut lire la moitié du fichier, soit 5 495 blocs, soit plus de 50 secondes.

Exercice 14. *Un arbre B+ indexe un fichier de 300 enregistrements. On suppose que chaque nœud stocke au plus 10 clés et 10 pointeurs. Quelle est la hauteur minimale de l'arbre et sa hauteur maximale ? (Un arbre constitué uniquement de la racine a pour hauteur 0).*

Inversement, on constate que l'arbre B+ a pour hauteur 1. Quel est le nombre minimal de pointeurs par bloc ? Le nombre maximal ?

Solution :

Hauteur minimale : tous les blocs sont pleins, avec 10 pointeurs. Donc il faut 30 blocs au niveau des feuilles, 3 au niveau intermédiaire, et un bloc avec 3 pointeurs pour la racine. Hauteur 2.

Hauteur maximale : il n'y a que 5 pointeurs par bloc. Donc il faut 60 blocs feuilles, 12 blocs au niveau intermédiaire, 3 au dessus, et un pour la racine. Hauteur 3.

Hauteur 1 : le nombre de pointeurs au niveau des feuilles est n^2, où n est le nombre de pointeurs par bloc.

n maximal : le plus grand n tel que $n < 300$ et $n^2 \geq 300$, soit 299.

n minimal : le plus petit n tel que $n^2 \geq 300$, soit $n = 18$.

Exercice 15. *On indexe une table par un arbre B+ sur un identifiant dont les valeurs sont fournies par une* séquence. *À chaque insertion un compteur est incrémenté et fournit la valeur de clé de l'enregistrement inséré.*

On suppose qu'il n'y a que des insertions dans la table. Montrez que tous les nœuds de l'index qui ont un frère droit sont exactement à moitié pleins.

Solution :

Au moment d'un éclatement, on se retrouve avec deux blocs remplis à moitié. Celui qui a un frère droit ne contient que des valeurs de clés inférieures à celui de ce frère droit. Comme on n'insère que des valeurs supérieures à toutes celles existantes, un nœud qui a un frère droit ne sera jamais modifié.

Exercice 16. *Soit un fichier non trié contenant N enregistrements de 81 octets chacun. Il est indexé par un arbre-B+, comprenant 3 niveaux, chaque entrée dans l'index occupant 20 octets. On utilise des blocs de 4 096 octets, sans entête, et on suppose qu'ils sont utilisés à 100 % pour le fichier et à 70 % pour l'index.*

On veut effectuer une recherche par intervalle dont on estime qu'elle va ramener m enregistrements. On suppose que tous les blocs sont lus sur le disque pour un coût uniforme.

1. *Donnez la fonction exprimant le nombre de lectures à effectuer pour cette recherche avec un parcours séquentiel.*
2. *Donnez la fonction exprimant le le nombre de lectures à effectuer en utilisant l'index.*
3. *Pour quelle valeur de m la recherche séquentielle est-elle préférable à l'utilisation de l'index, en supposant un temps d'accès uniforme pour chaque bloc ?*

   En déduire le pourcentage d'enregistrements concernés par la recherche à partir duquel le parcours séquentiel est préférable.

   On pourra simplifier les équations en éliminant les facteurs qui deviennent négligeables pour des grandes valeurs de N et de m.

Solution :

1. Combien d'enregistrements dans chaque bloc : $\lfloor 4096/81 \rfloor = 50$. Nombre de blocs du fichier : $\lceil \frac{N}{50} \rceil$. Il faut lire tout le fichier pour une recherche par intervalle.
2. Il faut lire la racine de l'arbre, puis un bloc du niveau intermédiaire, avant d'arriver à la feuille contenant l'adresse de la borne inférieure de l'intervalle. Il faut alors tenir compte du nombre d'entrées par bloc de l'arbre-B+, égal en moyenne, d'après l'énoncé, à $0,7 \times \lfloor 4096/20 \rfloor = 143$. Il faudra parcourir un nombre de feuilles de l'arbre égal à $\lceil m/143 \rceil + 1$ et pour chaque adresse rencontrée aller lire le bloc correspondant dans le fichier de données. On en déduit le nombre de lectures nécessaires :
$$2 + \lceil \frac{m}{143} \rceil + m$$
3. Il vaut mieux utiliser une lecture séquentielle quand le nombre de blocs à lire en passant par l'index est supérieur à a taille du fichier, soit :
$$\lceil \frac{N}{50} \rceil < 2 + \lceil \frac{m}{143} \rceil + m$$
Pour une grande valeur a, on pose $\lceil a \rceil = a$, cela donne :
$$\frac{N}{50} < 2 + \frac{m}{143} + m$$
Soit
$$m > \frac{1}{1 + 1/143} \times (\frac{N}{50} - 2)$$
On pose $m = p \times N$ où p est le pourcentage d'enregistrements ramenés par la recherche. On a donc :
$$p > \frac{1}{1 + 1/143} \times (\frac{1}{50} - \frac{2}{N})$$
Pour une valeur de N élevée, le facteur $2/N$ (correspondant à la descente dans l'arbre) devient négligeable et on constate qu'au-delà de 2 % des enregistrements ramenés par une recherche, un parcours séquentiel est préférable.
En pratique le placement dans un cache des blocs du fichier de données réduit l'effet de dispersion des lectures aléatoires engendrées par le parcours d'index. Mais en contrepartie la lecture séquentielle du fichier de données est bien plus efficace qu'une série d'accès aléatoires.

Exercice 17. *Soit les deux tables suivantes :*

```
CREATE TABLE R (idR VARCHAR(20) NOT NULL,
                  PRIMARY KEY (idR));

CREATE TABLE S (idS INT NOT NULL,
                idR VARCHAR(20) NOT NULL,
                  PRIMARY KEY (idS),
                 FOREIGN KEY idR REFERENCES R);
```

Indiquez, pour les ordres SQL suivants, quels index peuvent améliorer les performances ou optimiser la vérification des contraintes `PRIMARY KEY` *et* `FOREIGN KEY`.

1. `SELECT * FROM R WHERE idR = 'Bou'`
2. `SELECT * FROM R WHERE idR LIKE 'B%'`
3. `SELECT * FROM R WHERE LENGTH(idR) = 3`
4. `SELECT * FROM R WHERE idR LIKE '_ou'`
5. `INSERT INTO S VALUES (1, 'Bou')`
6. `SELECT * FROM S WHERE idS BETWEEN 10 AND 20`
7. `DELETE FROM R WHERE idR LIKE 'Z%'`

Solution :

1. `SELECT * FROM R WHERE idR = 'Bou'` : index sur R(idR).
2. `SELECT * FROM R WHERE idR LIKE = 'B%'` : index sur R(idR).
3. `SELECT * FROM R WHERE LENGTH(idR) = 3` pas d'index utile.
4. `SELECT * FROM R WHERE idR LIKE '_ou'` pas d'index utile.
5. `INSERT INTO S VALUES (1, 'Bou')` : index sur R(idR).
6. `SELECT * FROM S WHERE idS BETWEEN 10 AND 20` : index sur S(idS).
7. `DELETE FROM R WHERE idR > 10` : index sur S(idR).

Exercice 18. *Écrire l'algorithme de recherche par intervalle dans un arbre-B.*

Exercice 19. *Soit la liste des départements données dans l'exercice 10, la clé étant le numéro de département. On suppose qu'un bloc contient 5 enregistrements.*

1. *Proposez une fonction de hachage et le nombre d'entrées de la table, puis construisez la structure en prenant les enregistrements dans l'ordre indiqué.*
2. *Insérez un ou plusieurs autres départements de manière à provoquer un chaînage pour l'une des entrées.*

Exercice 20. *Même exercice, mais avec une structure basée sur le hachage extensible.*

On prendra une fonction de hachage sur le nom, définie de la manière suivante : $f(nom) = i_1 i_2 \cdots i_4$ *avec* $i_j = 1$ *si la lettre* $nom[i_j]$*est en position impaire dans l'alphabet, et 0 sinon. Donc* $f(Aude) = 1101$*. Voici la liste des valeurs de hachage, en ne prenant que les 4 premiers bits.*

```
Allier  1001    Indre   1000    Cher    1010    Paris   0101
Jura    0101    Ariège          1011    Tarn    0100    Aude    1101
Aveyron 1011    Doubs   0110    Savoie  1101    Meuse   1111
Cantal  1100    Marne   1100    Loire   0110    Landes  0100
Calvados 1100   Gard    1100    Vaucluse 0111   Ardèche 1001
```

Le hachage extensible consiste à considérer les n *derniers bits de la valeur de hachage (en d'autres termes, on prend* $h(v) mod 2^n$*, le reste de la division de la valeur de hachage par* 2^n*). Au départ on prend* $n = 1$*, puis* $n = 2$ *quand une extension est nécessaire, etc. Montrez l'évolution de la structure de hachage extensible en insérant les départements dans l'ordre indiqué (de gauche à droite, puis de haut en bas).*

Exercice 21. *On indexe l'ensemble des pistes de ski du domaine skiable alpin français. Chaque piste a une couleur qui peut prendre les valeurs suivantes : `Verte`, `Rouge`, `Bleue`, `Noire`. On suppose qu'il y a le même nombre de pistes de chaque couleur, et que la taille d'un bloc est de 4 096 octets.*

1. *Donnez la taille d'un index bitmap en fonction du nombre de pistes indexées.*
2. *Combien de blocs faut-il lire, dans le pire des cas, pour rechercher les pistes vertes ?*
3. *Combien de blocs pour compter le nombre de pistes vertes ?*
4. *Que se passerait-il si on utilisait un arbre B+ ?*

Exercice 22. *Soit un arbre-R dont chaque nœud a une capacité maximale de 4 entrées. Quelle est à votre avis la meilleure distribution possible au moment de l'éclatement du nœud de la figure 10 ?*

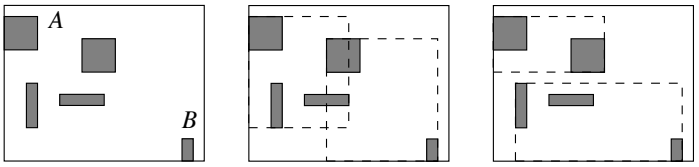


FIGURE 10 – Éclatement d'un nœud d'arbre-R

Exercice 23. *Même hypothèse que précédemment : on insère le rectangle 8 dans le nœud de la figure 11.*

1. *Quel est le résultat après réorganisation ?*
2. *Coment pourrait-on faire mieux, en s'autorisant la* réinsertion dans l'arbre *de un ou plusieurs rectangles d'un nœud avant éclatement.*

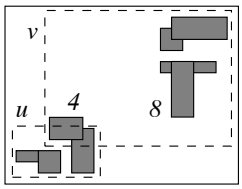


FIGURE 11 – Éclatement avec réinsertion

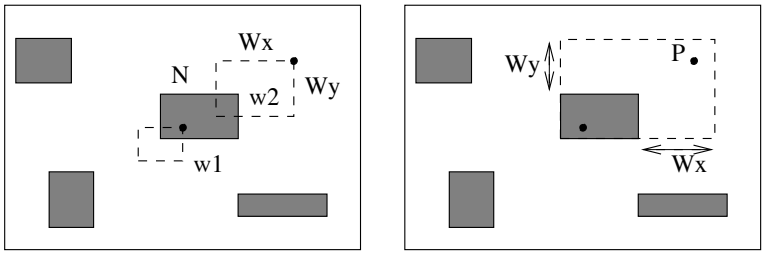
Exercice 24. *Soit un rectangle R de dimension $[h, l]$ dans un espace de dimension $[d, d]$.*

1. *Quelle est la probabilité qu'un pointé $P(x, y)$ ramène le rectangle, en supposant une distribution uniforme ?*
2. *Quelle est la probablité qu'un fenêtrage $W(w_h, w_l)$ ramène le rectangle ?*

Solution :

1. Pointé : $P(N) = \frac{h \times l}{d^2}$
2. Fenêtrage : le coin supérieur droit de W doit tomber dans R ou dans une extension de R de taille $[w_h, w_l]$ (voir figure 12), soit

$$\frac{(h + w_h) \times (l + w_l)}{d^2}$$



Un fenêtrage Le pointé équivalent

FIGURE 12 – Un fenêtrage et le pointé équivalent

Exercice 25. *Écrire l'algorithme de pointé avec un arbre-R.*

3 Évaluation de requêtes

Les premiers exercices sont à faire en TD, et adoptent le modèle d'exécution par itération/pipelinage. Les exercices suivants consistent à expérimenter les concepts proposés avc le SGBD ORACLE, en créant une base de données, en l'alimentant avec un nombre choisi d'enregistrements, et en évaluant l'effet de manipulations diverses sur les performances du système.

3.1 Opérateurs d'accès aux données

Exercice 26. *Soit la liste des départements de l'exercice 10, page 9. On suppose qu'on peut stocker deux enregistrements par bloc. Décrire l'algorithme de tri-fusion sur le numéro de département appliqué à ce fichier dans les cas suivants :*

1. $M = 4$ *blocs ;*
2. $M = 3$ *blocs.*

Exercice 27. *Soit un fichier de 10 000 blocs et un buffer en mémoire centrale de 3 blocs. On trie ce fichier avec l'algorithme de tri-fusion.*

- *Combien de fragments sont produits pendant la première passe ?*
- *Combien de passes faut-il pour trier complètement le fichier ?*
- *Quel est le coût total en entrées/sorties ?*