Date de dernière modification *: 30/6/2015*

Annexe **4**

Technologie des bases de données

Cette annexe comprend des matériaux complémentaire au chapitre 4. En particulier, elle développe plus avant les problématiques de **correction d'erreurs** dans les mémoires RAID, la **recherche séquentielle** dans un fichier, le processus de **tri externe** d'un fichier séquentiel, l'analyse du comportement d'un fichier **séquentiel indexé**, l'analyse du comportement d'un **fichier calculé**, des détails sur l'évaluation des **index bitmap** et une discussion précise sur l'**opportunité de créer un index** en support à une classe déterminée de requêtes.
En outre, elle comprend un jeu d'**exercices**, dont la plupart sont accompagnés d'une solution ou de conseils de résolution.

A4.1 LES DISQUES RAID : RÉCUPÉRATION D'UNE VALEUR CORROMPUE

Dans les architecture RAID 5 et RAID 6, chaque unité de données (mot, secteur, page, etc.) est dupliquée un certain nombre de fois et est accompagnée d'un code permettant de détecter une erreur et, dans certaines conditions, de corriger cette erreur. Montrons par un exemple simplifié le fonctionnement du mécanisme de correction.

Admettons que la mémoire comporte quatre disques de données sans redondance D1, D2, D3, D4 + un disque de code D5 (les données et les codes peuvent être distribués sur les vcinq disques mais ceci ne change rien au raisonnement). Admettant encore qu'un code soit associé à chaque octet et qu'il soit calculé par l'application de l'opérateur binaire xor (voir annexe A24) sur les quatre octets de même rang des

disques de données. Ainsi, si on désigne par O_{dn} la chaîne binaire de l'octet de rang n du disque d, on a :

$$\mathbf{O_{5n}} = O_{1n} \text{ xor } O_{2n} \text{ xor } O_{3n} \text{ xor } O_{4n}$$

Une propriété intéressante de l'opérateur xor est que cette égalité reste vraie si on permute l'octet de gauche avec l'un des octets de droite. Si l'un des octets de droite est perdu, suite à une erreur détectée sur son disque, on peut le reconstutuer à partir du code O_{5n} et des autres octets de données.

Supposons que le code O_{5n} ait été calculé comme suit :

1001 1100 = 0010 1101 xor 1000 1011 xor 0101 1101 xor 0110 0111

Supposons à présent que des erreurs soient détectées sur le disque D2, de sorte que O_{2n} soit considéré comme non fiable. On va le reconstruire comme suit :

$$O_{2n} = O_{1n} \text{ xor } O_{3n} \text{ xor } O_{4n} \text{ xor } \mathbf{O_{5n}}$$

soit,

1000 1011 = 0010 1101 xor 0101 1101 xor 0110 0111 xor **1001 1100**

En pratique, la technique est plus élaborée que ce que cet exemple a montré, mais la raisonnement est similaire.

A4.2 RECHERCHE SÉQUENTIELLE D'ÉLÉMENTS DANS UN FICHIER

Soit un fichier séquentiel de **N** enregistrements parmi lesquels lequel **n** ($0 \leq \mathbf{n} \leq \mathbf{N}$) vérifient une propriété déterminée. Par exemple, un fichier CLIENT contient 10 000 enregistrements, dont 10 sont relatifs aux clients habitant à `'Mirval'`. On admet dans un premier temps que ces **n** enregistrements sont distribués de manière aléatoires dans le fichier.

On pose la question suivante : combien d'enregistrements de ce fichier faut-il lire en moyenne pour retrouver le $\mathbf{k}^e$ de ces 10 enregistrements ? Appelons **m(N,n,k)** ce nombre. Quelques réflexions préliminaires :

- si **N** = 0, il n'y a rien à faire pour constater l'échec de la recherche,
  soit **m(0,0,k)** = **0** ; nous écarterons ce cas désormais et supposerons que ($1 \leq$ **N**)
- si **n** = 0, il faut lire *la totalité du fichier* pour constater l'échec de la recherche,
  soit **m(N,0,k)** = **N** ; nous écarterons ce cas désormais et supposerons que ($1 \leq$ **n** $\leq$ **N**)
- si **k** > **n**, il faut lire *la totalité du fichier* pour constater l'échec de la recherche,
  soit **m(N,n,k)** = **N** ; nous écarterons ce cas désormais et supposerons que ($1 \leq$ **k** $\leq$ **n** $\leq$ **N**)
- si **n** = 1, on sait qu'il faut lire *à peu près la moitié du fichier en moyenne*,[1]
  soit **m(N,1,1)** = (1 + **N**) / 2

– si **n** = **N**, chaque enregistrement vérifie la propriété, soit :
soit **m(N,N,k)** = **k** et **m(N,N,N)** = **N**

En toute généralité, on a, en excluant les trois cas triviaux identifiés ci-dessus :

m(N,n,k) = **k** x (**N** + 1) / (**n** + 1)

Pour N = 100 et n = 10, on obtient :

m(100,1,1) = 50,5	**m(100,2,1)** = 33,7	**m(100,3,1)** = 25,3	**m(100,10,1)** = 9,2
	m(100,2,2) = 67,3	**m(100,3,2)** = 50,5	**m(100,10,5)** = 45,9
		m(100,3,3) = 75,8	**m(100,10,10)** = 91,8

Imaginons à présent que le fichier soit ordonné selon la valeur de la propriété. Dans ce cas, les enregistrements recherchés sont regroupés en une sous-liste de taille **n**. Cette sous-liste occupe, au mieux, les positions [1, **n**] et au pire les positions [**N**-**n** + 1, **N**]. En considérant que les valeurs recherchées sont équiprobables, les sous-listes occupent les positions de **p** à **P**, avec **p** ∈ [1, **N**-**n** + 1] et **P** ∈ [**n**, **N**], les valeurs de **p** et **P** étant équiprobables. Soit les valeurs moyennes

p' = (**N** - **n**)/2 + 1 et **P**' = (**N** + **n**)/2.

On a donc selon ces hypothèses,

m'**(N,n,k)** = (**N** - **n**)/2 + 1 + **k** - 1, soit (**N** - **n**)/2 + **k**

Pour N = 100 et n = 10, on obtient :

m(100,10,1) = 46 et **m(100,10,10)** = 55

Il est évident, dans l'hypothèse d'un fichier trié, qu'il sera préférable de l'implanter sur un support adressable et de procéder à un accès par index, ou, à défaut, à une recherche dichotomique (ou par interpolation), qui exigera dans le cas le plus dévavorable, l'examen de $\log_2 N$ enregistrements.

A4.3 TRI D'UN FICHIER SÉQUENTIEL

Cette section détaille la section 4.6.2 de l'ouvrage consacrée au tri externe d'un fichier séquentiel. On y analyse de manière plus détaillée les principes du tri par fusion-éclatement, du tri interne du tournoi et les diverses optimisations du tri externe.

A4.3.1 Fusion de deux listes triées

La technique que nous décrirons se base sur un algorithme très simple : la *fusion* (ou *interclassement*) de deux listes ordonnées disjointes (figure A4.1). On examine un couple d'éléments, choisis dans chaque liste, en commençant par les premiers éléments de ces listes. On sort le plus petit, qu'on remplace immédiatement par le suivant dans sa liste, puis on examine de la même manière ce nouveau couple.

1. Avec un écart-type de 0,289 x **N** [Knuth,,1998]

Lorsqu'une liste est épuisée, on sort les éléments restant de l'autre liste. La liste des éléments ainsi sortis comporte tous les éléments des deux listes sources et est triée.

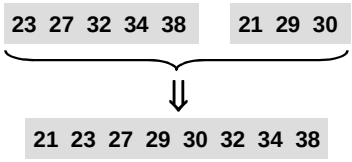


Figure A4.1 - Fusion de deux listes triées

A4.3.2 Fusion de deux listes non triées

Observons plus en détail une liste quelconque, telle que celle de la figure A4.2. Elle n'est pas triée mais on observe qu'elle est constituée de séquences plus ou moins longues d'éléments déjà en ordre : (82, 98), (87), (78, 83), etc. On appelle **monotonie** d'une liste chacune de ses sous-listes maximales d'éléments consécutifs en ordre. Toute liste est donc constituée d'une suite de monotonies.

On peut montrer [Knuth, 1998] qu'une liste de **N** éléments *parfaitement en désordre* comporte en moyenne **N**/2 monotonies. Une liste triée ne comporte qu'une seule monotonie et une liste triée dans le sens inverse en comporte **N**.

Figure A4.2 - Liste arbitraire comportant 8 sous-séquences triées (ou *monotonies*)

Que se passerait-il si nous tentions de fusionner deux listes non ordonnées selon la procédure décrite ci-dessus ? En généralisant celle-ci, nous pouvons lui donner le comportement suivant :

- des deux éléments en cours d'examen, on sort le plus petit *qui soit supérieur à celui qu'on vient de sortir*
- la fusion se poursuit jusqu'à ce que, soit une liste est épuisée, soit les deux éléments en cours d'examen sont *inférieurs au dernier élément sorti*.

Si une liste est épuisée, on sort les éléments restants de l'autre, sinon, on opère une rupture et on redémarre une fusion sans tenir compte du dernier élément sorti. La liste résultante n'est pas nécessairement triée, mais chaque rupture y clôture une monotonie qui a été obtenue par la fusion de deux monotonie, une dans chaque liste source. Idéalement, les deux listes sources contiennent le même nombre de monotonies. La liste résultante contient autant de monotonies que chacune des listes sources. Le nombre de monotonies a donc été divisé par 2. La figure A4.3 illustre l'effet de cette procédure.

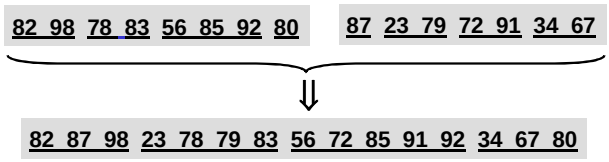


Figure A4.3 - Fusion des monotonies de deux listes non ordonnées. Des 8 monotonies des listes sources, il en reste 4 dans la liste résultante.

A4.3.3 Tri externe d'un fichier

Nous sommes à présent capable de produire une liste de **M** monotonies à partir de deux listes comportant elles-même (à une unité près) **M** monotonies. En parcourant les listes une seule fois, nous avons réduit de moitié le nombre de monotonies. Pour atteindre l'objectif, qui est de trier une (longue) liste non ordonnée, il nous faut encore résoudre un petit problème : comment construit-on, à partir d'une liste initiale, deux listes sources comportant à peu près le même nombre de monotonies ?

Simple jeu d'enfant : on parcourt la liste d'origine et on recopie ses monotonies alternativement sur une liste de sortie ou sur l'autre. Dès qu'on examine un élément plus petit que celui qu'on vient de sortir, on change de liste de sortie et on y copie cet élément avant de poursuivre sur cette même liste. On dira qu'on a **éclaté** la liste source en deux listes de sortie. Si la liste d'origine est déjà triée, elle sera intégralement recopiée dans une des listes de sortie, l'autre restant vide.

Nous disposons de deux processus utiles : la fusion de deux listes (non) triées et l'éclatement d'une liste (non) triée. Il nous suffit de les composer pour obtenir un premier algorithme dont l'effet est illustré à figure A4.4, où chaque barre verticale identifie une monotonie. Ces listes et demi-listes étant lues et écrites séquentiellement, cette procédure est directement applicable aux fichiers séquentiels.

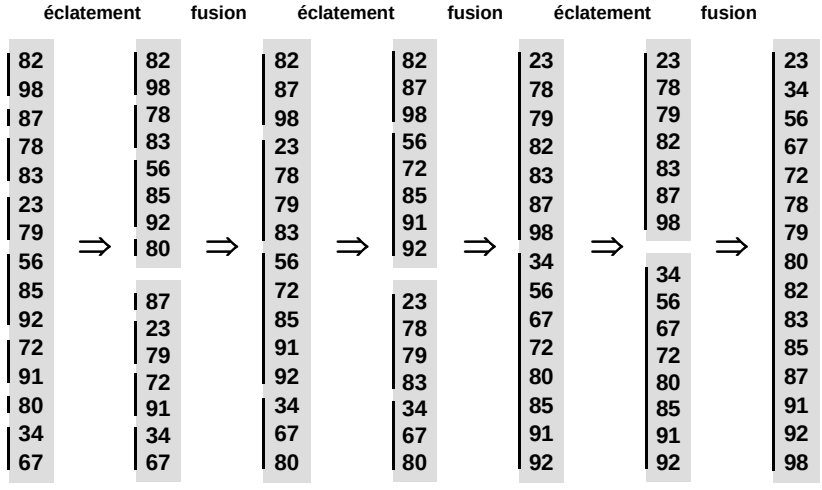


Figure A4.4 - Tri d'un fichier - Exemple d'application

Les processus d'éclatement n'ont pour but que de préparer une fusion. C'est celle-ci qui joue un rôle actif, celui de diminuer de moitié le nombre de monotonies. Une phase complète est constituée d'un éclatement suivi d'une fusion. Chaque phase réalise quatre opérations sur des fichiers : lecture d'un fichier complet, écriture de deux demi-fichiers, lecture de deux demi-fichiers et écriture d'un fichier complet. En considérant que l'écriture se fait sans vérification, on peut admettre que ces quatre opérations ont le même coût, celui de la lecture du fichier complet.

Dans l'enchaînement des phases, on observe qu'on écrit un fichier complet pour le relire immédiatement après. Il serait plus efficace de ne pas écrire le fichier qui résulte d'une fusion mais de procéder directement à la répartition en deux demi-fichiers des éléments qui sont fournis successivement par le processus de fusion. On définit ainsi le processus de *fusion-éclatement*.

Les opérations comprennent dès lors une première phase d'éclatement, suivie d'un certain nombre de phases de fusion-éclatement (figure A4.5). Chaque phase ne réalise plus qu'une lecture et une écriture d'un fichier complet (ou de deux demi-fichiers). Le tri se termine lorsque le deuxième demi-fichier est vide. L'algorithme est décrit à la figure A4.6. A partir d'un fichier Fi, il produit le fichier Fo trié et de même contenu.

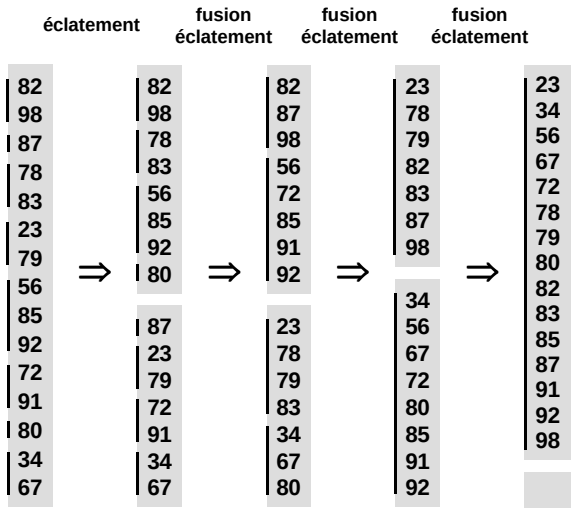


Figure A4.5 - Tri d'un fichier - Procédure améliorée

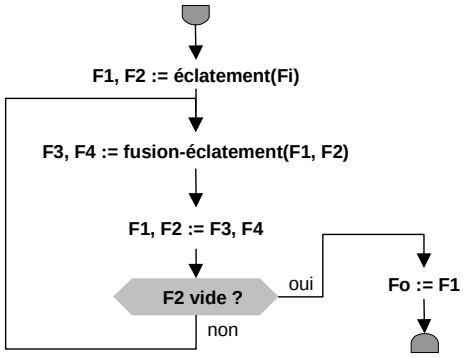


Figure A4.6 - Tri d'un fichier - Algorithme de base

A4.3.4 Calcul du temps de tri d'un fichier

Pour calculer le temps de tri d'un fichier de **Nr** enregistrements, deux questions préalables se posent : quel est le coût d'une phase et combien de phases requiert le tri ?

Chaque phase, y compris la première, que nous appellerons *phase 0*, lit un fichier complet (ou deux demi-fichiers) et écrit un fichier complet (ou deux demi-fichiers). Considérant le scénario de lecture 2, on obtient :

$$\mathbf{t_{ph0}} = 2 \times \mathbf{t_{lsf2}}$$

Le nombre de phases est 1 + **P** , où **P** est le nombre de phases actives de fusion-éclatement. **P** se calcule en considérant que :

- chaque phase active divise le nombre de monotonies par deux
- le nombre de monotonies dans le fichier source est **M**; pour un fichier quelconque en désordre, on estime[2] $\mathbf{M} = \mathbf{Nr} / 2$
- au terme de la dernière phase, le nombre de monotonies est de 1.

La phase **1** produit M/2 monotonies. La phase **k** en produit $\mathbf{M} / 2^{\mathbf{k}}$ et la phase **P** en produit $\mathbf{M} / 2^{\mathbf{P}}$, qui ne doit pas dépasser 1. On a donc :

$$0{,}5 < \mathbf{M} / 2^{\mathbf{P}} \leq 1$$

$$0{,}5 \times 2^{\mathbf{P}} < \mathbf{M} \leq 1 \times 2^{\mathbf{P}}$$

$$\mathbf{P} - 1 < \log_2 \mathbf{M} \leq \mathbf{P}$$

$$\mathbf{P} = \lceil \log_2 \mathbf{M} \rceil$$

Nous obtenons ainsi une première estimation du temps de tri :

$$\mathbf{t_{tri0}} = (1 + \mathbf{P}) \times \mathbf{t_{ph0}} = 2 \times (1 + \mathbf{P}) \times \mathbf{t_{lsf2}} = 2 \times (1 + \lceil \log_2 \mathbf{M} \rceil) \times \mathbf{t_{lsf2}}$$

Supposons à présent que le fichier est en désordre parfait par rapport au critère de tri ($\mathbf{M} = \mathbf{Nr} / 2$ et $\lceil \log_2 \mathbf{M} \rceil = \lceil \log_2 \mathbf{Nr} \rceil - 1$). Il vient :

2. [Knuth,1999], p. 253

$$\mathbf{t_{tri0}} = 2 \times \lceil \log_2 \mathbf{Nr} \rceil \times \mathbf{t_{lsf2}}$$

Puisque $\mathbf{t_{lsf2}}$ est proportionnel à **Nr**, le temps de tri peut aussi s'écrire, K étant un coefficient dépendant du support de mémoire, $\mathbf{t_{tri0}} = K \times \mathbf{Nr} \lceil \log_2 \mathbf{Nr} \rceil$. La complexité de l'algorithme est donc O(**Nr** log **Nr**).

A4.3.5 Optimisation du tri externe

On détaille ci-dessous les différentes techniques d'optimisation du processus de tri externe d'un fichier.

• Tri interne lors de la phase initiale d'éclatement

Lors de la phase initiale, les enregistrements du fichier d'entrée transitent par la mémoire centrale pour être distribués entre les deux demi-fichiers. Cette optimisation consiste à charger un certain nombre d'enregistrements en mémoire, **de les y trier**, puis de les écrire dans un des demi-fichiers. Supposons qu'on réserve en mémoire centrale un espace pouvant accueillir **b** enregistrements. Ceux-ci font l'objet d'un tri interne avant écriture dans un fichier de sortie. On crée ainsi des monotonies de taille **b** au lieu de 2 dans la procédure standard. Ce qui diminuera le nombre **M** de monotonies et donc le nombre **P** de phases. Bien mieux, en appliquant une variante du *tri du tournoi* [Knuth, 1998], on produit des monotonies dont la taille moyenne est de 2 **b**. On recalcule donc le nombre de phases :

$$\mathbf{M'} = \lceil \mathbf{Nr} / 2\,\mathbf{b} \rceil$$

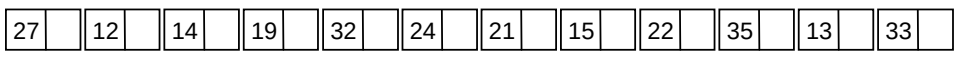
$$\mathbf{P'} = \lceil \log_2 \mathbf{M'} \rceil = \lceil \log_2 \lceil \mathbf{Nr} / 2\,\mathbf{b} \rceil \rceil = \lceil \log_2 \mathbf{Nr} / 2 \rceil - \lfloor \log_2 \mathbf{b} \rfloor = \mathbf{P} - \lfloor \log_2 \mathbf{b} \rfloor$$

$$\mathbf{t_{tri1}} = 2 \times (1 + \mathbf{P'}) \times \mathbf{t_{lsf2}} = 2 \times (1 + \mathbf{P} - \lfloor \log_2 \mathbf{b} \rfloor) \times \mathbf{t_{lsf2}}$$

Par exemple, **b** = 8 200 permet d'économiser 13 phases.

Le tri du tournoi est une technique simple, élégante et efficace, de complexité O(**Nr** log **Nr**). Il consiste à introduire dans un tableau **b** enregistrements du fichier source, à sélectionner dans ce tableau l'enregistrement de clé minimale, à écrire celui-ci dans le fichier de sortie courant, puis à le remplacer par l'enregistrement suivant du fichier d'entrée. On sélectionne ensuite l'enregistrement du tableau dont la clé est la plus petite mais supérieure à la dernière écrite dans le fichier de sortie courant. On l'écrit puis on le remplace comme décrit ci-avant. On constitue ainsi une monotonie beaucoup plus grande (2 fois en moyenne) que celle qui aurait été constituée du contenu initial du tableau. La section critique de cette procédure est la sélection de l'élément minimal. On utilise pour ce faire un arbre binaire dont chaque mise à jour nécessite autant de comparaisons/modifications que l'arbre possède de niveaux, soit $\log_2 \mathbf{b}$.

On l'illustre par un exemple réduit dont le traitement est illustré à la figure A4.7. Le fichier à trier comprend des enregistrements dont la clé de tri forme la séquence 27, 12, 14, 19, 32, 24, 21, 15, 22, 35, 13, 33, etc. On y compte parmi les 12 premiers enregistrements 6 monotonies naturelles de longueur 2 en moyenne.



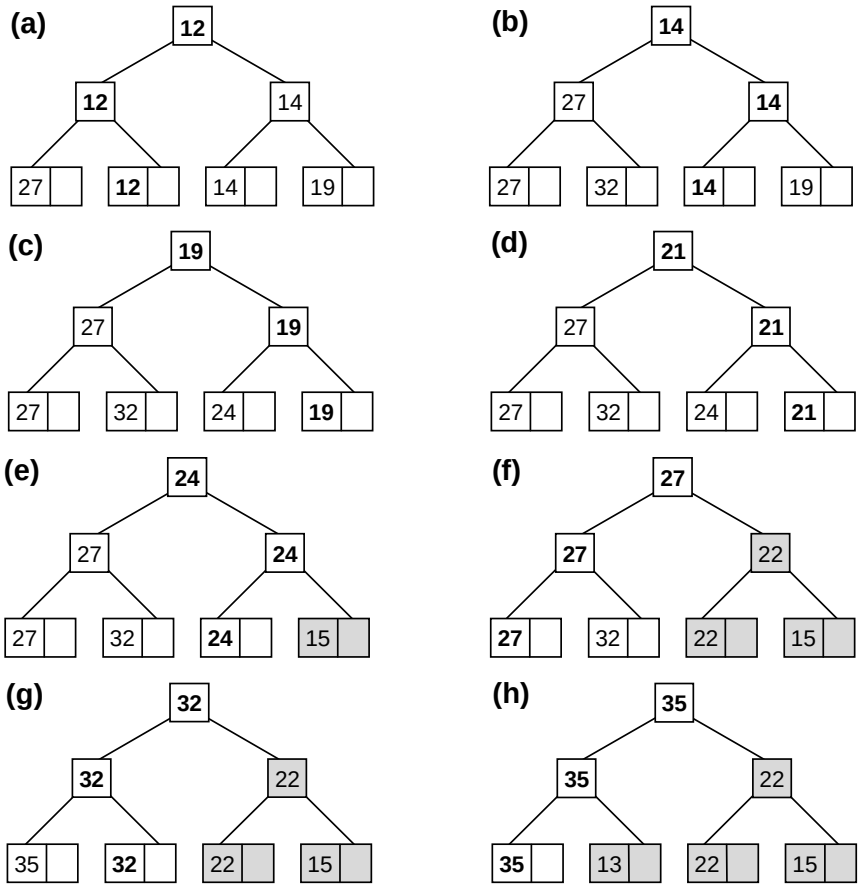
On réserve en mémoire centrale un espace pouvant accueillir **b** = 4 enregistrements.

a) Chargement des 4 premiers enregistrements, de clés 27, 12, 14, 19. On construit sur cette sous-séquence un arbre binaire représentant les gagnants des tournois opposant les enregistrements pris deux à deux. Le gagnant est celui dont la clé est la plus petite. En demi-finale, le combat des clés 27 et 12 donne **12** gagnant et le combat des clés 14 et 19 donne **14** gagnant. En finale, le combat des clés 12 et 14 donne **12** gagnant. La clé **12** est donc gagnante du tournoi et son enregistrement sort sur un des fichiers de sortie.

b) L'enregistrement 12 est remplacé par l'enregistrement suivant du fichier d'entrée, doit 32. On met à jour la branche affectée de l'arbre du tournoi : le combat de 27 et 32 donne **27** gagnant et le combat en finale de 27 et 14 donne **14** gagnant du tournoi. L'enregistrement **14** sort sur le même fichier de sortie.

c) L'enregistrement 14 est remplacé par l'enregistrement 24; **19** gagne et sort.

d) L'enregistrement 19 est remplacé par l'enregistrement **21**, qui gagne et qui sort.



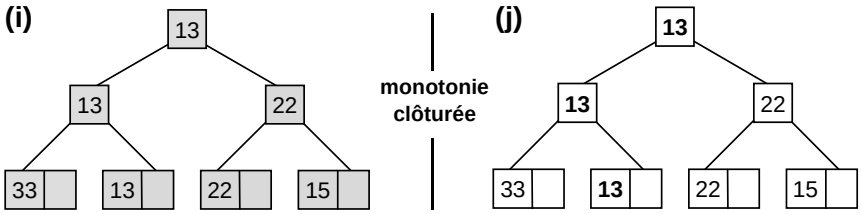


Figure A4.7 - Tri du tournoi produisant de grandes monotonies initiales

e) L'enregistrement 21 est remplacé par 15, valeur qui est inférieure à celle qui vient de sortir (21).L'enregistrement 15 ne peut plus participer à la monotonie courante et est donc désactivé (en grisé); 24 gagne sur 15 par abandon, puis sur 27 en finale. L'enregistrement **24** gagne et sort.

f) L'enregistrement 24 est remplacée par 22 qui est désactivé car sa clé est inférieure à 24. Les deux combattants 15 et 22 étant désactivés, le gagnant l'est aussi. L'enregistrement **27** gagne par défaut et sort.

g) L'enregistrement 27 est remplacé par l'enregistrement 35. L'enregistrement **32** est gagnant et sort.

h) L'enregistrement 32 est remplacé par l'enregistrement 13 qui est désactivé. L'enregistrement **35** sort, et est remplacé par l'enregistrement 33 qui est désactivé.

i) Tous les noeuds de l'arbre sont ainsi désactivés, ce qui clôture la monotonie courante.

j) Une nouvelle monotonie est construite. Tous les noeuds sont réactivés, l'arbre est mis à jour; 13 gagne et sort sur un autre fichier de sortie.

On observe que la monotonie ainsi construite (12, 14, 19, 21, 24, 27, 32, 35) comporte 8 enregistrements au lieu de 2 en moyenne.

• Augmentation du nombre de voies

La procédure de base distribue les monotonies dans deux (demi-)fichiers. On parle alors d'un tri à 2 voies. Rien n'empêche de les distribuer dans un nombre plus important, soit **v**, de fichiers. Lors de chaque phase, on divise le nombre de monotonie par **v**. Le nombre de phases est dès lors plus réduit :

$$\mathbf{P''} = \lceil \log_{\mathbf{v}} \mathbf{M'} \rceil = \lceil \log_2 \mathbf{M'} / \log_2 \mathbf{v} \rceil \cong \lceil \mathbf{P'} / \log_2 \mathbf{v} \rceil$$

$$\mathbf{t_{tri2}} = 2 \times (1 + \mathbf{P''}) \times \mathbf{t_{lsf2}} = 2 \times (1 + \lceil \mathbf{P'} / \log_2 \mathbf{v} \rceil) \times \mathbf{t_{lsf2}}$$

Si on choisit autant de voies que de monotonies (**v** = **M'**), on obtient **P''** = 1. Le tri est alors linéaire en **Nr**.

$$\mathbf{t_{tri2'}} = 4 \times \mathbf{t_{lsf2}}$$

Il existe une variante d'optimisation, le *tri polyphasé*, qui n'utilise que **v** + 1 voies au total au lieu de 2 **v** et qui évite les voies inutilisées lors de l'éclatement. Il consiste à fusionner **v** fichiers d'entrée, garnis d'un nombre approprié de monotonies (selon la suite de Fibonnaci), vers une seule voie de sortie. Lorsqu'une voie d'entrée est vide,

elle devient la nouvelle voie de sortie tandis que la précédente, partiellement garnie, redevient une voie d'entrée. Cette optimisation complexe était importante lorsque chaque voie correspondait à une unité de mémoire, typiquement une unité à bande qui était un appareil très coûteux. Elle est devenue moins critique lorsque chaque voie est devenue un simple fichier sur disque, créé dynamiquement et dont la lecture et l'écriture sont optimisées selon les techniques décrites en 4.3.3.

• Parallélisme lecture/écriture

Si, pour chaque phase, les fichiers d'entrée sont stockés sur un disque et les fichiers de sortie sur un autre, alors les opérations de lecture et d'écriture peuvent s'exécuter simultanément. Le temps d'une phase est dès lors de $\mathbf{t_{lsf2}}$ au lieu de 2 x $\mathbf{t_{lsf2}}$.

$$\mathbf{t_{tri3}} = (1 + \mathbf{P''}) \times \mathbf{t_{lsf2}} = (1 + \lceil \mathbf{P'} / \log_2 \mathbf{v} \rceil) \times \mathbf{t_{lsf2}}$$

Ou, si **v** = **M'** :

$$\mathbf{t_{tri3'}} = 2 \times \mathbf{t_{lsf2}}$$

a) Exemple

On considère un fichier séquentiel de 1 000 000 enregistrements de 400 octets implanté sur notre disque de référence. Les enregistrements possèdent un champ identifiant mais sont en désordre parfait par rapport à celui-ci. L'écriture se fait sans vérification. On calcule le temps du tri du fichier selon les différentes optimisations. En supposant le mode de rangement (a) de la figure 4.9, le fichier comporte 97 657 pages. Le temps de lecture ou d'écriture du fichier est de

$\mathbf{t_{lsf2}} \cong \mathbf{N_p} \times \mathbf{t_{ls1}}$ = 97 657 x 0,184 = 17 970 ms ou 18 s

- **Procédure de base**

  $\mathbf{M} = \lceil \mathbf{Nr} / 2 \rceil$ = 500 000

  $\mathbf{P} = \lceil \log_2 \mathbf{M} \rceil$ = 19

  $\mathbf{t_{tri0}} = 2 \times (1 + \mathbf{P}) \times \mathbf{t_{lsf2}}$ = **720** secondes ou 12 minutes

- **Tri interne initial**

  On réserve un espace de tri de 40 Mo, soit **b** = 100 000

  $\mathbf{M'} = \lceil \mathbf{Nr} / 2\,\mathbf{b} \rceil$ = 5

  $\mathbf{P'} = \lceil \log_2 \mathbf{M'} \rceil$ = 3

  $\mathbf{t_{tri1}} = 2 \times (1 + \mathbf{P'}) \times \mathbf{t_{lsf2}}$ = **144** sec. ou 2 min 24 s.

- **Augmentation du nombre de voies**

  On conserve un espace de tri initial de 50 Mo et on répartit les enregistrements sur 5 voies en entrée et 1 en sortie (théoriquement 5 en sortie mais une seule sera utilisée). 6 fichiers sont donc ouverts durant le processus.

  $\mathbf{M'} = \lceil \mathbf{Nr} / 2\,\mathbf{b} \rceil$ = 5

  $\mathbf{P''} = \lceil \log_{\mathbf{v}} \mathbf{M'} \rceil = \lceil \log_2 \mathbf{M'} / \log_2 \mathbf{v} \rceil$ = 1

  $\mathbf{t_{tri2}} = 2 \times (1 + \mathbf{P''}) \times \mathbf{t_{lsf2}}$ = **72** sec. ou 1 min 12 s.

- **Parallélisme lecture/écriture**

  Dans les mêmes conditions que ci-dessus, les voies d'entrées sont sur un disque et les voies de sortie sur un autre.

  ttri3 = (1 + **P''**) x **tlsf2** = **36** s.

- **Parallélisme des voies en lecture et en écriture**

  Il est possible d'étendre le principe du parallélisme aux différentes voies d'entrée et/ou de sortie. En distribuant les voies sur autant de disques indépendants, celles-ci sont lues en parallèle, de sorte que le coût de lecture et/ou d'écriture des voies se réduit à celui d'une seule voie. Le temps de tri est alors, si toutes les voies sont parallélisées :

  $$\mathbf{ttri4 =} (1 + \mathbf{P''}) \times \mathbf{tlsf2} / \mathbf{v} = (1 + \lceil \mathbf{P'} / \log_2 \mathbf{v} \rceil) \times \mathbf{tlsf2} / \mathbf{v}$$

  Par exemple, la répartition de 5 voies sur 5 disques utilise 10 disques mais divise le temps de chaque phase par 5. Cette technique réclamant un grand nombre de disques sera surtout appliquée dans les machines de base de données.

Cette évaluation ne concerne que le temps des échanges de données avec la mémoire externe. En toute rigueur, il faudrait également prendre en compte d'autres coûts : temps de calcul (notamment comparaison des valeurs de clé), temps d'ouverture et fermeture des fichiers, temps d'allocation d'espace aux fichiers. En ce qui concerne le temps de calcul lors d'une phase de fusion-éclatement, on observera que le nombre d'opérations de comparaison est linéaire en fonction du nombre d'enregistrements. Par exemple, la fusion de deux listes totalisant **n** éléments nécessite au plus **n** - 1 comparaisons.

b) Synthèse des calculs

La figure A4.8 reprend les différents résultats par optimisation. Ces modèles de calcul du temps de tri selon les différentes optimisations ont été traduits dans la feuille de calcul SORT.xls disponible sur le site de l'ouvrage.

Calcul du tri d'un fichier

Procédure de base

$\mathbf{P} = \lceil \log_2 \mathbf{M} \rceil$

$\mathbf{t_{tri0}} = 2 \times (1 + \mathbf{P}) \times \mathbf{t_{lsf2}}$

Tri interne initial

$\mathbf{M'} = \lceil \mathbf{Nr} / 2\,\mathbf{b} \rceil$

$\mathbf{P'} = \lceil \log_2 \mathbf{M'} \rceil$

$\mathbf{t_{tri1}} = 2 \times (1 + \mathbf{P'}) \times \mathbf{t_{lsf2}}$

Augmentation du nombre de voies

$\mathbf{P''} = \lceil \log_{\mathbf{v}} \mathbf{M'} \rceil = \lceil \log_2 \mathbf{M'} / \log_2 \mathbf{v} \rceil$

$\mathbf{t_{tri2}} = 2 \times (1 + \mathbf{P''}) \times \mathbf{t_{lsf2}}$

si $\mathbf{v} = \mathbf{M'}$, $\mathbf{t_{tri2}} = 4 \times \mathbf{t_{lsf2}}$

Parallélisme lecture/écriture

$\mathbf{t_{tri3}} = (1 + \mathbf{P''}) \times \mathbf{t_{lsf2}}$

si $\mathbf{v} = \mathbf{M'}$, $\mathbf{t_{tri3}} = 2 \times \mathbf{t_{lsf2}}$

Figure A4.8 - Temps de tri d'un fichier (échanges avec la mémoire secondaire)

A4.4 ORGANISATION SÉQUENTIELLE INDEXÉE

La section 4.8 consacrée aux fichiers séquentiels indexés présente un modèle de calcul simplifié des volumes et des temps d'accès et de modification des données. La présente section propose un calcul plus précis de ces grandeurs. Il analyse en particulier l'évolution des taux d'occupation et du nombre d'éclatements de pages dans la phase d'exploitation de ces fichiers.

A4.4.1 Caractéristiques et performances d'un fichier séquentiel indexé

Les algorithmes d'insertion et de suppression d'enregistrements préservent l'ordre des enregistrements et des entrées de l'index. Ils garantissent également aux pages du fichier de base et de l'index (sauf la dernière) un **taux d'occupation au moins égal à 50 %**. Lors de la création du fichier, il est habituel de charger chaque page à un taux d'occupation initial, typiquement de l'ordre de 60 à 80 %. En cours d'exploitation, ce taux d'occupation va évoluer selon la taille des pages et du profil des opérations de mise à jour (proportion insertions/suppressions). Progressivement, on va observer une diminution du taux d'occupation et une dégradation des temps d'accès due à la proportion croissante des pages hors séquence. Il est conseillé de réorganiser un fichier séquentiel indexé dès que ces indicateurs ne sont plus satisfaisants. Cette réorganisation est simple et relativement peu coûteuse : il suffit de recopier le fichier à partir de sa version actuelle, ce qui est un processus purement séquentiel et linéaire. Nous étudierons ci-après plus en détail l'évaluation des

volumes, des temps d'accès et des temps de modification. Cette évaluation sera basée sur deux grandeurs principales : le taux d'occupation et le nombre d'éclatement de pages. l'évolution de ce taux d'occupation en fonction de divers paramètres à la fin de cette section.

A4.4.2 Les volumes

L'estimation du volume d'un fichier séquentiel indexé ne pose pas de problèmes majeurs. On a en effet :

$$\mathbf{Np} = \mathbf{Npb} + \mathbf{Npi}$$

où **Npb** et **Npi** sont les nombres de pages respectivement du fichier de base et de l'index. Dans les calculs ci-dessous, on ignore les éventuelles pages vides.

a) Volume du fichier de base

Le fichier de base contient **Nr** enregistrements de taille **Lr** rangés dans une suite de **Npb** pages de taille **Lp**. Pour simplifier, on admet qu'un enregistrement est entièrement compris dans une page. On connaît τ_b, le taux d'occupation moyen des pages de base à un moment donné. Soient **Mrpp** et **Nrpp** les nombres maximum et moyen d'enregistrements par page :

$$\mathbf{Mrpp} = \lfloor \mathbf{Lp} / \mathbf{Lr} \rfloor$$

$$\mathbf{Nrpp} = \tau_b \times \mathbf{Mrpp}$$

Ceci nous permet de calculer la taille du fichier de base :

$$\mathbf{Npb} = \lceil \mathbf{Nr} / \mathbf{Nrpp} \rceil$$

b) Volume de l'index

On connaît τ_i, le taux d'occupation moyen des pages d'index à un moment donné. La longueur d'une entrée d'index est **Li**, celle du champ clé est **Lk** et celle d'un pointeur de page **Lptr**. Soient **Mipp** et **Nipp** les nombres maximum et moyen d'entrées d'index par page. On a :

$$\mathbf{Li} = \mathbf{Lk} + \mathbf{Lptr}$$

$$\mathbf{Mipp} = \lfloor \mathbf{Li} / \mathbf{Lp} \rfloor$$

$$\mathbf{Nipp} = \tau_i \times \mathbf{Mipp}$$

L'évaluation de **Npi** s'effectue le plus simplement du niveau n au niveau 1. Le niveau n comporte **Ni(n)** entrées réparties dans **Npi(n)** pages :

$$\mathbf{Ni(n)} = \mathbf{Npb}$$

$$\mathbf{Npi(n)} = \lceil \mathbf{Ni(n)} / \mathbf{Nipp} \rceil$$

Le calcul se déroule ensuite itérativement selon les expressions de récurrence, pour un niveau **m** quelconque :

$$\mathbf{Ni(m)} = \mathbf{Npi(m+1)}$$

$$\mathbf{Npi(m)} = \lceil \mathbf{Ni(m)} / \mathbf{Nipp} \rceil$$

jusqu'à ce que

$$\mathbf{Npi(m)} = 1$$

auquel cas **m** = 1. On en déduit les valeurs de **n** et de **Npi** :

$$\mathbf{Npi} = \sum_m \mathbf{Npi(}m\mathbf{)}$$

Considérons un fichier de **Nr** = 2 000 000 enregistrements de **Lr** = 400 octets, dont **Lk** = 40 pour la clé K. Les pages sont de **Lp** = 4 Ko remplies à $\tau\mathbf{b} = \tau\mathbf{i} = 0{,}8$. Le fichier de base est formé de **Npb** = 250 000 pages et l'index de **Npi** = 3 408 pages. L'index comporte trois niveaux, respectivement de 1, 46 et 3 361 pages.

c) Calcul simplifié

On observe que la taille de l'index est généralement très faible et que celle du dernier niveau y est largement prédominante. On peut donc admettre :

$$\mathbf{Npi} \cong \mathbf{Npi(n)} = \lceil \mathbf{Npb} / \mathbf{Nipp} \rceil$$

On pourra alors calculer la valeur de **n** comme suit, en ignorant les arrondis,

$$\mathbf{Npi(m)} \cong \mathbf{Npb} / \mathbf{Nipp}^{\mathbf{n-m+1}}$$

Pour le niveau 1, on a :

$$\mathbf{Npb} / \mathbf{Nipp}^{\mathbf{n}} \cong 1$$

dont on tire :

$$\mathbf{n} = \lceil \log_{\mathbf{Nipp}} \mathbf{Npb} \rceil = \lceil \log \mathbf{Npb} / \log \mathbf{Nipp} \rceil$$

A4.4.3 Les temps de lecture

Nous étudierons successivement la lecture séquentielle, la lecture par index et la lecture par intervalle.

a) Lecture séquentielle des enregistrements

Lorsque le fichier est jeune, les pages successives du fichier de base sont physiquement contiguës. La séquence logique (l'ordre de lecture) et la séquence physique (l'ordre des adresses dans le fichier) sont identiques (figure A4.9, haut). On a montré (section 4.6) que le temps de lecture séquentielle $\mathbf{t_{lsf}}$ du fichier, dans l'hypothèse réaliste d'une lecture anticipée des 112 pages d'une piste, était de :

$$\mathbf{t_{lsf}} = \mathbf{t_{lsf2}} = \mathbf{N_p} \times \mathbf{t_{ls1}} = \mathbf{46\ s}$$

Chaque éclatement de page provoque deux ruptures dans la séquence physique car la nouvelle page, dite physiquement *hors séquence*, peut être très éloignée de la page source qui éclate. Celle-ci reste en place mais la nouvelle page qui la précède logiquement est choisie parmi les pages libres. Celle-ci peut être trouvée à proximité de la page source, par exemple dans le cas où une page voisine a été abandonnée suite à une fusion lors de la suppression d'un enregistrement, mais elle peut aussi se trouver en fin de fichier (figure A4.9, bas). Dans ce dernier cas, l'accès à la nouvelle page peut exiger, pour les fichiers occupant une partie importante du disque, un temps proche de $\mathbf{t_{la1}}$ (typiquement 12,3 ms) au lieu de $\mathbf{t_{ls1}}$ (typiquement 0,184 ms). A priori, on devrait compter le coût de deux ruptures (*aller*, de 100 à 99 997, et *retour*,

de 99 997 à 101). Cependant, grâce à la lecture anticipée, la pénalité ne sera que d'une rupture : lors de la lecture des pages d'adresses 99, 100, 101, 99.997, 102, 103, ..., les pages 99, 100, 101, 102, 103, 104, ... sont en général toutes présentes dans le tampon lors de l'accès à la page 99 997. Le retour à la page 101 sera alors de coût nul.

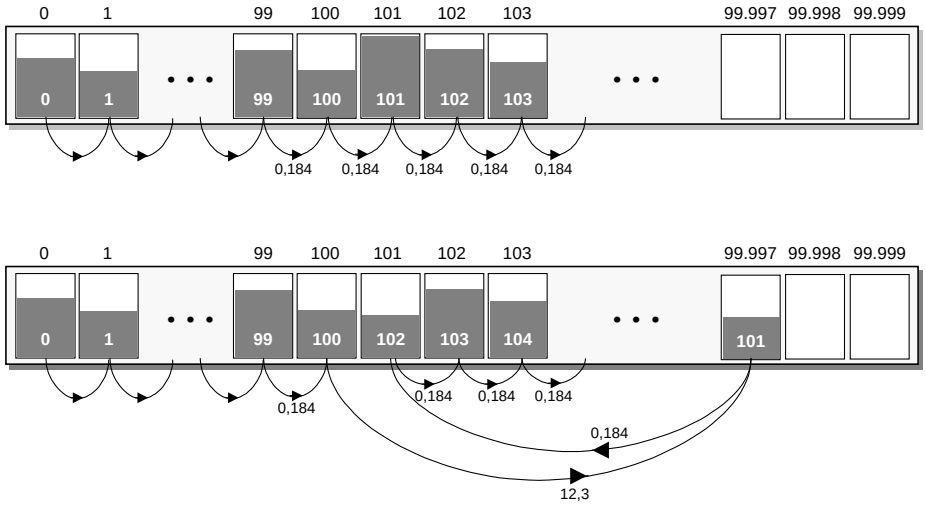


Figure A4.9 - Etat du fichier avant (haut) et après (bas) insertion d'un enregistrement dans la page 101 et estimation des temps d'accès en msec L'ajout d'une nouvelle page 102 se fait en fin de fichier.[3]

En admettant qu'un nombre **h** de pages soient physiquement hors séquence, il faut, en toute généralité, remplacer dans le calcul du coût de lecture **h** lectures séquentielles par **h** lectures aléatoires. Nous apprécions ce phénomène par le **taux de ruptures** $\pi_r = \mathbf{h} / \mathbf{N}_p$, mesurant la proportion de pages *hors séquence*, qui exigeront une lecture aléatoire. $\pi_r = 0$ correspond à un fichier jeune d'un seul tenant, sans ruptures, et $\pi_r = 1$ implique que toutes les pages sont disposées aléatoirement dans le fichier. La lecture séquentielle du fichier se présente comme une suite de lectures de séquences de pages contiguës. Il vient donc, en désignant par $\mathbf{t_{lr}}$ le temps d'accès à une page hors séquence :

$$\mathbf{t_{lsf}} = (1 - \pi_r) \times \mathbf{Npb} \times \mathbf{t_{ls1}} + \pi_r \times \mathbf{Npb} \times \mathbf{t_{lr}}$$

La valeur de $\mathbf{t_{lr}}$ dépend de plusieurs facteurs, qui devront être estimés dans chaque situation particulière. Si le disque est fortement partagé, l'accès à une page hors séquence qui est trop éloignée dans le fichier pour se trouver dans le tampon est un accès aléatoire, qui coûte donc $\mathbf{t_{la1}}$. Si le disque est dédié au processus client, si le fichier est très volumineux et si les pages hors séquence sont placées en fin de fichier, la conclusion est la même. Si les pages libérées lors d'une fusion de deux

3. Le numéro placé au dessus des pages représente l'adresse physique de la page dans le fichier. Le numéro à l'intérieur de la page représente le numéro de séquence logique de la page. Lors d'une lecture séquentielle, les pages sont lues dans l'ordre de leurs numéros de séquence logique.

pages voisines sont réutilisables[4], et si les pages libres sont nombreuses, alors certaines pages hors séquence peuvent être proches de leur page source; le coût de leur accès peut ainsi être inférieur à t_{la1}.

Considérons par exemple un fichier de 250 000 pages dont 20 % sont hors séquence suite à des éclatements ($\pi_r = 0{,}2$). En admettant pour simplifier que $t_{lr} = t_{la1}$ (cas le plus défavorable), le temps de lecture du fichier de base est

$$t_{lsfr} = (1 - 0{,}2) \times 250\,000 \times 12{,}3 + 0{,}2 \times 250\,000 \times 0{,}184 = \mathbf{652\ s},$$

soit **14 fois plus** que le temps de lecture d'un fichier jeune ! Ce rapport de 14 constitue le *facteur de dégradation* **dr** du fichier. Dans cet exemple, il est manifestement excessif et réclame clairement une réorganisation du fichier. Nous étudierons cette question plus loin. Le facteur de dégradation est facile à calculer :

$$d_r = t_{lsfr} / t_{lsf} = 1 + \pi_r \times (t_{lr} - t_{ls1})/t_{ls1}$$

Ce facteur vaut 1 pour un fichier parfaitement en séquence ($\pi_r = 0$) et t_{lr}/t_{ls1} pour un fichier complètement hors séquence ($\pi_r = 1$). Remarquons que ce phénomène n'affecte que l'accès séquentiel et l'accès par intervalle et pas du tout l'accès par index d'un enregistrement.

b) Lecture par index d'un enregistrement

En théorie, le temps de lecture d'un enregistrement dont K = k est celui de la lecture d'une page dans chacun des n niveaux d'index suivie de celle d'une page de base, soit :

$$t_{l1k} = (n + 1) \times t_{la1}$$

Dans notre exemple, $t_{lk} = 49{,}2$ ms. L'index permet donc 20 lectures par seconde. C'est ici qu'intervient le tampon. Supposons que nous maintenions le premier niveau d'index (1 page) dans le tampon. Pour un coût négligeable nous obtenons un gain de 12,3 ms. On a donc $t_{lk} = 36{,}9$ ms, correspondant à une vitesse de 27 lectures par seconde. En maintenant dans le tampon les pages des niveaux 1 et 2, soit 1 + 46 pages d'index, l'accès à l'enregistrement est limité à 2 pages, soit 24,6 ms, pour une vitesse de lecture de 40,6 lectures par seconde. Poussons le raisonnement à la limite : on charge la totalité de l'index dans le tampon. La consommation d'espace est évidemment importante (3 408 pages, soit 13,3 Mo), mais le temps d'accès par l'index n'est plus que de 12,3 ms. La vitesse est dès lors de 81 lectures par seconde. Si nous admettons qu'il soit possible de précharger et de maintenir **m** niveaux d'index dans le tampon[5], il vient :

$$t_{l1k} = (n - m + 1) \times t_{la1}$$

4. Nous verrons cependant que les fusions de pages sont en général très rares.

5. Ce type d'optimisation concerne avant tout les **serveurs** de bases de données, réalisant les accès à une base de données demandés par plusieurs dizaines ou plusieurs centaines d'applications clientes simultanées. Certains index sont fortement sollicités (plus de 10 fois par seconde par exemple), de sorte que le maintien en mémoire centrale de leurs premiers niveaux, voire de tous, permet un gain de temps d'accès très important.

L'espace à réserver dans le tampon est Σ_m **Npi(**m**)** + 1, compte tenu du cadre d'accueil de la page de base

En toute généralité, pour un index fréquemment utilisé, on peut admettre m = 2, conduisant à un temps d'accès de **tl1k** = (n - 1) x **tla1** et exigeant la réservation raisonnable de **Npi(**2**)** + 2 pages dans le tampon[6].

c) Lecture par index des enregistrements d'un intervalle

On rappelle que l'accès par intervalle de clé K ∈ [k1, k2], consiste en un accès par clé K = k1 (en réalité au premier enregistrement[7] dont K ≥ k1) suivi d'une lecture séquentielle des enregistrements dont K ≤ k2. En notant **pik** la probabilité de réalisation de la condition d'intervalle, **nrik** = **pik** x **Nr** enregistrements tombent dans cet intervalle. Ceux-ci occupent **npik** pages consécutives[8] :

$$n_{pik} = \lfloor (n_{rik} - 1) / N_{rpp} + 0{,}5 \rfloor + 1$$

Le temps de lecture de ces enregistrements est celui de la lecture d'une séquence de pages consécutives. Si nous ignorons le phénomène de rupture, il vient, pour un accès séquentiel avec lecture anticipée :

$$t_{lik} = n_{pik} \times t_{ls1}$$

En tenant compte des ruptures, on affine le calcul comme suit :

$$t_{lik} = t_{la1} + (1 - \pi_r) \times n_{pik} \times t_{ls1} + \pi_r \times n_{pik} \times t_{lr}$$

À ces valeurs de **tlik**, il convient d'ajouter le temps de parcours de l'index, soit (**n** - **m**) x **tla1.**

A4.4.4 Les temps de modification des données

Le contenu d'un fichier séquentiel indexé peut faire l'objet des trois primitives de modification : *insertion* d'un enregistrement, *suppression* d'un enregistrement et *modification* de la valeur de certains champs d'un enregistrement. Nous évaluons le coût de chacune de ces opérations.

---

6. Il est possible que cette estimation soit pessimiste. Les niveaux d'index supérieurs qui n'ont pas été préchargés sont néanmoins soumis aux règles de gestion habituelles du tampon. Ainsi, si certaines pages de ces niveaux sont particulièrement sollicitées, elles auront tendance à rester dans le tampon, réduisant la nécessité d'accéder au disque. Si les accès se concentrent sur une petite proportion des enregistrements du fichier, ce qui est assez fréquent (on pense aux *produits* les plus vendus), alors il est possible que le temps moyen d'accès par index soit plus faible encore que ne le suggère notre estimation.

7. On observe que l'enregistrement dont K ≥ k1, s'il en existe un, se trouve dans la page dans laquelle l'enregistrement dont K = k1 aurait dû se trouver.

8. Calcul approché pour **nrik** > 0. Chaque enregistrement peut constituer le début d'un intervalle. On admet qu'en moyenne ce début est au milieu d'une page.

a) Insertion d'un enregistrement

Il existe principalement[9] deux scénarios d'insertion : sans éclatement dans une page offrant un espace libre suffisant et avec éclatement dans une page pleine.

Dans les deux scénarios, la page doit au préalable être rangée en mémoire (dans le tampon) par une lecture classique via l'index (coût : t_{l1k}). Si cette lecture échoue (la valeur de la clé est absente du fichier), alors l'insertion se poursuit normalement, si elle réussit, l'insertion est annulée pour cause de violation de la contrainte d'unicité. Dans les calculs, on ignorera ce dernier cas.

- **Insertion sans éclatement**

L'enregistrement est inséré dans la page en mémoire puis celle-ci est recopiée sur le disque. Comme nous avons considéré que les insertions étaient aléatoires, la probabilité que cette page accueille prochainement un autre enregistrement est très faible, de sorte que sa recopie sur disque (coût : t_{la1} sans vérification) est à imputer entièrement à cette insertion. Le temps requis pour l'insertion d'un enregistrement sans éclatement est donc :

$$t_{ins}0 = t_{l1k} + t_{la1}$$

- **Insertion avec éclatement**

La procédure est plus complexe. Elle comporte les étapes suivantes :

1. *Identifier une page libre dans le fichier*. Généralement, le SGBD gère une liste des pages libres, soit sous la forme d'une chaîne de ces pages, soit sous la forme d'une liste d'au moins N_r bits, chacun d'eux indiquant si la page correspondante du fichier est libre ou utilisée. Ces données techniques sont généralement déjà en mémoire centrale, ce qui permet d'ignorer le coût de cette recherche[10].
2. *Charger cette page dans le tampon*. La page libre repérée est lue et rangée dans le tampon (coût : t_{la1}).
3. *Répartir les enregistrements entre les deux pages*. Cette opération s'effectue par des transfert en mémoire central. Son coût est négligeable.
4. *Recopier ces deux pages sur le disque*. Coût : $2 \times t_{la1}$.
5. *Mettre à jour l'index*. L'ajout d'une nouvelle page implique l'insertion dans l'index d'une nouvelle entrée[11]. La page dans laquelle cette entrée doit être insérée est en principe déjà dans le tampon puisqu'on vient de l'utiliser. Si cette

9. On ignore par exemple les insertions dans un fichier vide et en fin du fichier et les insertions de chargement ou en séquence, qui correspondent à des procédures spécifiques.
10. Correctif : s'il n'existe plus de pages libres, le SGBD doit demander au système d'exploitation d'allouer de l'espace supplémentaire au fichier. Cette opération est coûteuse mais rare si la quantité d'espace alloué à chaque réquisition est importante (typiquement plusieurs milliers de pages).
11. Si la nouvelle page est insérée logiquement devant la page source, l'entrée d'index de cette dernière n'est pas modifiée. Si elle est insérée derrière, cette entrée doit être modifiée mais elle se trouvera en général dans la même page d'index que l'entrée de la nouvelle page. Il n'y aura donc pas de coût supplémentaire.

page offre de l'espace libre, il suffit de la recopier sur disque après insertion de l'entrée (coût : t_{la1}). Si elle est pleine, alors on procède à une insertion avec éclatement dans le niveau d'index précédent. Cependant, ce cas sera beaucoup plus rare que dans le fichier de base, puisque les pages d'index ont le plus souvent une contenance nettement supérieure à celle des pages de base ($N_{ipp} \gg N_{rpp}$). On suggère d'ignorer ce phénomène.

Le temps requis pour l'insertion d'un enregistrement avec éclatement est donc :

$$t_{ins}1 = t_{l1k} + t_{la1} + 2 \times t_{la1} + t_{la1}$$
$$= t_{l1k} + 4 \times t_{la1}$$

- **Insertion d'un enregistrement**

Appelons τ_{split} la proportion des opérations d'insertions qui entraînent un éclatement de pages[12]. On peut dès lors calculer le temps moyen d'une opération d'insertion :

$$t_{ins} = (1 - \tau_{split}) \times t_{ins}0 + \tau_{split} \times t_{ins}1 = t_{l1k} + (1 + 3 \times \tau_{split}) \times t_{la1}$$

b) Suppression d'un enregistrement

Il existe trois scénarios de suppression : suppression dans une page dont le taux d'occupation reste supérieur à 50%, suppression suivie d'un rééquilibrage et suppression avec fusion de pages. Dans ces scénarios, on admet que l'enregistrement concerné a déjà été lu par le programme client[13] et que sa page se trouve en général encore dans le tampon. Si tel n'est pas le cas, on comptabilisera le coût de lecture de cette page ou de son rechargement. Dans les calculs ci-dessous, on ignorera le coût de ces accès.

- **Suppression simple**

L'enregistrement est supprimé de la page en mémoire puis celle-ci est recopiée sur le disque. Les insertions étant aléatoires par hypothèse (cf. insertion), le temps requis pour la suppression d'un enregistrement déjà en mémoire est donc :

$$t_{del}0' = t_{la1}$$

Si l'enregistrement supprimé est le dernier de sa page, il faut en outre modifier l'entrée de cette page dans l'index puis recopier la page d'index ainsi modifiée sur le disque (on suppose que cette page est dans le tampon, puisqu'on vient de l'utiliser). Tous les enregistrements d'une page ont la même probabilité d'être supprimés, soit $1 / N_{rpp}$. Nous devons donc préciser l'expression du coût :

$$t_{del}0 = t_{la1} + 1 / N_{rpp} \times t_{la1}$$
$$= (1 + 1 / N_{rpp}) \times t_{la1}$$

12. to split = éclater, to bal(ance) = équilibrer et to merge = fusionner.
13. Nous verrons dans le chapitre 9 que la requête de suppression `delete` spécifie soit un ensemble de lignes par une condition de sélection (`searched delete`), soit la ligne courante (`positioned delete`).

Cette modification d'une entrée de l'index pourrait, s'il s'agissait de la dernière de sa page, entraîner une modification équivalente d'une entrée du niveau inférieur, et ainsi de suite. La faible probabilité de cette situation ($N_{ipp} \gg N_{rpp}$) nous suggère de l'ignorer.

- **Suppression avec rééquilibrage**

La page (logiquement) voisine est introduite en mémoire. Elle est probablement physiquement voisine (sauf si elle résulte d'un ancien éclatement) mais si l'accès à l'enregistrement à supprimer date d'un certain temps ou si le disque est partagé, il faut compter un accès aléatoire t_{la1}. Les deux pages sont ensuite à réécrire sur le disque, ce qui coûte au maximum 2 accès au disques, soit $2 \times t_{la1}$. Comme une des pages a vu son dernier enregistrement changer, il faut modifier son entrée dans l'index. En supposant que la page d'index est encore dans le tampon, on compte une recopie sur disque, soit t_{la1}. Si cette entrée était la dernière de sa page, alors il faut également modifier une entrée dans une page du niveau précédent de l'index. Événement de faible probabilité que nous ignorerons. Le temps requis pour la suppression d'un enregistrement avec rééquilibrage est :

$$t_{del1} = 4 \times t_{la1}$$

- **Suppression avec fusion**

La page (logiquement) voisine est introduite en mémoire comme ci-dessus (coût : t_{la1}) et le contenu de l'une est transféré vers l'autre dans le tampon. La page conservée, elle est recopiée sur le disque (coût : t_{la1}) et son entrée dans l'index est inchangée. En ce qui concerne la page vide, elle introduite dans la liste des pages libres (on comptabilise un accès : t_{la1}) et son entrée dans l'index est supprimée (coût de recopie de la page d'index : t_{la1}). Ici encore, on ignore l'éventuelle propagation de cette suppression dans les niveaux inférieurs de l'index. Le temps requis pour la suppression d'un enregistrement avec rééquilibrage est :

$$t_{del2} = 4 \times t_{la1}$$

- **Suppression d'un enregistrement**

Appelons τ_{bal} la proportion des opérations de suppression qui entraînent un rééquilibrage de pages et τ_{merge} la proportion des opérations de suppression qui entraînent une fusion de pages. On peut dès lors calculer le temps moyen d'une opération de suppression :

$$\begin{aligned} t_{del} &= (1 - \tau_{bal} - \tau_{merge}) \times t_{del0} + \tau_{bal} \times t_{del1} + \tau_{merge} \times t_{del2} \\ &= (1 + 3 \times \tau_{bal} + 3 \times \tau_{merge} + (1 - \tau_{bal} - \tau_{merge}) / N_{rpp}) \times t_{la1} \end{aligned}$$

c) Modification d'en enregistrement

La modification d'un champ appartenant à la clé de l'index primaire induit la modification de la valeur de l'identifiant de l'enregistrement. Comme la position de ce dernier dépend directement de cette valeur, il est nécessaire de déplacer cet enregistrement. Du point de vue de son coût, cette opération revient à supprimer puis à recréer cet enregistrement[14].

$$t_{updi} = t_{del} + t_{ins}$$

La modification d'un autre champ, non identifiant, se réduit[15] à la recopie de la page sur le disque :

$$t_{updni} = t_{la1}$$

A4.4.5 Synthèse des calculs

La figure A4.10 synthétise les différents modèles de calcul. Ils ont été traduits dans la feuille de calcul ISAM.xls que le lecteur trouvera sur le site de l'ouvrage.

Calcul d'un fichier séquentiel indexé

Taille du fichier

$N_p = N_{pb} + N_{pi}$

$N_{pb} = \lceil N_r / N_{rpp} \rceil$
$N_{rpp} = \tau_b \times M_{rpp}$
$M_{rpp} = \lfloor L_p / L_r \rfloor$

$N_{pi} = \sum_m N_{pi(m)}$
$N_{pi(m)} = \lceil N_{i(m)} / N_{ipp} \rceil$
$N_{i(m)} = N_{pi(m+1)}$
$N_{i(n)} = N_{pb}$
$N_{pi(1)} = 1$
$N_{ipp} = \tau_i \times M_{ipp}$
$M_{ipp} = \lfloor L_i / L_p \rfloor$
$L_i = L_k + L_{ptr}$

Temps de lecture séquentielle du fichier et dégration de ruptures

$t_{lsf} = (1 - \pi_r) \times N_{pb} \times t_{ls1} + \pi_r \times N_{pb} \times t_{lr}$
$dr = t_{lsfr} / t_{lsf} = 1 + \pi_r \times (t_{lr} - t_{ls1})/t_{ls1}$

Temps de lecture d'un enregistrement via l'index

$t_{lk1} = (n - m + 1) \times t_{la1}$ (m niveaux d'index dans le tampon)

Temps de lecture des enregistrements d'un intervalle (proportion p_{ik}) via l'index

$n_{rik} = p_{ik} \times N_r$
$n_{pik} = \lfloor (n_{rik} - 1)/ N_{rpp} + 0{,}5 \rfloor + 1$
$t_{lik} = (n - m + 1) \times t_{la1} + (1 - \pi_r) \times n_{pik} \times t_{ls1} + \pi_r \times n_{pik} \times t_{lr}$

14. Si cet enregistrement est une ligne d'une table que référencent d'autres lignes, le SGBD va exécuter des opérations additionnelle de préservation de l'intégrité référentielle, telle que la modification des clés étrangères (`on update cascade`). L'évaluation du coût de la suppression d'un enregistrement en situation est dès lors plus complexe mais se réduit à une suite d'opérations élémentaires.

15. Si ce champ appartient à un index secondaire, sa modification entraînera des opérations de gestion de cet index que nous examinerons dans une section suivante.

Temps d'insertion d'un enregistrement

$t_{ins} = t_{l1k} + (1 + 3 \times \tau_{split}) \times t_{la1}$

Temps de suppression d'un enregistrement

$t_{del} = (1 + 3 \times \tau_{bal} + 3 \times \tau_{merge} + (1 - \tau_{bal} - \tau_{merge}) / N_{rpp}) \times t_{la1}$

Temps de modification de la clé d'un enregistrement

$t_{updi} = t_{del} + t_{ins}$

Figure A4.10 - Calculs relatifs aux fichiers séquentiels indexés

La figure A4.11 reprend une version simplifiée de ces modèles de calcul, qui ne reprend que le calcul simplifié des volumes et les temps de lecture.

Calcul (simplifié) d'un fichier séquentiel indexé

Taille du fichier

$N_p = N_{pb} + N_{pi}$

$N_{pb} = \lceil N_r / N_{rpp} \rceil$ $\qquad$ $N_{pi} \cong \lceil N_{pb} / N_{ipp} \rceil$

$N_{rpp} = \tau_b \times M_{rpp}$ $\qquad$ $N_{ipp} = \tau_i \times M_{ipp}$

$M_{rpp} = \lfloor L_p / L_r \rfloor$ $\qquad$ $M_{ipp} = \lfloor L_i / L_p \rfloor$

$L_i = L_k + L_{ptr}$

Temps de lecture séquentielle du fichier

$t_{lsf} = (1 - \pi_r) \times N_{pb} \times t_{ls1} + \pi_r \times N_{pb} \times t_{lr}$

$\pi_r = h / N_p$

$d_r = t_{lsfr} / t_{lsf} = 1 + \pi_r \times (t_{lr} - t_{ls1}) / t_{ls1}$

Temps de lecture d'un enregistrement via l'index

$t_{lk1} = (n - m + 1) \times t_{la1}$ (m niveaux d'index dans le tampon)

Temps de lecture des enregistrements d'un intervalle (proportion p_{ik}) via l'index

$n_{rik} = p_{ik} \times N_r$

$n_{pik} = \lfloor (n_{rik} - 1) / N_{rpp} + 0{,}5 \rfloor + 1$

$t_{lik} = (n - m + 1) \times t_{la1} + (1 - \pi_r) \times n_{pik} \times t_{ls1} + \pi_r \times n_{pik} \times t_{lr}$

Temps d'insertion d'un enregistrement

version simplifiée

Temps de suppression d'un enregistrement

version simplifiée

Temps de modification de la clé d'un enregistrement

version simplifiée

Figure A4.11 - Calculs simplifiés relatifs aux fichiers séquentiels indexés

A4.4.6 Collecte des caractéristiques de comportement - Le simulateur ISAM

L'étude mathématique de l'évolution des caractéristiques de comportement d'un fichier telles que le taux d'occupation et le nombre d'éclatements de pages est assez complexe, de sorte que nous ferons appel à un **simulateur** qui fournit diverses mesures décrivant notamment ces valeurs pour une suite de pages (un fichier) qu'on soumet à une suite d'opérations d'insertion et de suppression[16].

a) Le simulateur ISAM

Le simulateur a été développé sous la forme d'une application ISAM-auto.mdb[17]. Il utilise un *fichier de paramètres* qui fournit les paramètres de départ d'une série de simulations à exécuter. Il produit pour chacune un *fichier de statistiques* directement transférables dans une feuille Excel, pour calculs ultérieurs et présentation graphique. Pour exécuter une simulation, on précise :

- N_{p0}, le nombre initial de pages du fichier
- M_{rpp}, la taille des pages (en nombre d'enregistrements)
- N_{rpp0}, le nombre initial d'enregistrements dans chaque page; dans la suite, nous ferons aussi référence à τ_0, le taux d'occupation initial des pages ($N_{rpp0} = \tau_0 \times M_{rpp}$)
- **%ins**, le taux (en %) d'insertions d'enregistrements parmi les opérations appliquées au fichier; la proportion d'opérations de suppression est (100 - **%ins**); cette grandeur définit le *profil des opérations*; aussi exprimable sous la forme $\tau_{ins} = \%ins / 100$.
- N_{op}, le nombre d'opérations à appliquer au fichier; les opérations d'insertion et de suppression sont distribuées aléatoirement selon la proportion **%ins**; plus précisément, on spécifie un nombre N_{pas} de *pas*, chacun constitué de N_{oppas} opérations ($N_{op} = N_{pas} \times N_{oppas}$).
- N_{simul}, le nombre de fois que la simulation est exécutée, les statistiques produites étant les moyennes de ces N_{simul} exécutions.

Outre les valeurs initiales, le simulateur enregistre au terme de chaque *pas* les statistiques suivantes :

- le nombre d'opérations déjà exécutées (n_{op}),
- le nombre d'insertions (n_{ins}),
- le nombre de suppressions (n_{del}),
- le taux d'occupation courant (τ),

16. Comme nous l'avons vu, les modifications n'ont pas d'impact sur le phénomène, sauf la modification de l'identifiant, qui se traduit par une suppression suivie d'une insertion.
17. Il s'agit d'une application MS Access qui remplace le simulateur isam.com mentionné dans la première édition et jugé trop limité.

- le nombre de pages (**N_p**),
- le nombre d'enregistrements (**N_r**),
- le nombre de rééquilibrages de pages (**n_{bal}**),
- le nombre d'éclatement de pages (**n_{split}**),
- le nombre de fusions de pages (**n_{merge}**)
- le nombre d'erreurs (**n_{error}**; par exemple, suppressions dans un fichier vide).

Les valeurs de **n_{ins}**, **n_{del}**, **n_{bal}**, **n_{split}**, **n_{merge}** et **n_{error}** sont relatives au début de la simulation courante. Les valeurs incrémentales se calculent par différence entre deux pas. On précisera dans le fichier de paramètres lesquelles de ces statistiques on désire voir dans les fichiers de statistiques produits.
Le fichier de paramètres doit être présent dans le répertoire "Mes Documents". Son nom est "ISAM-param-<id>.txt", où <id> est un code quelconque à introduire avant les simulations. Par défaut, son nom est "ISAM-param-STD.txt". Le fichier de paramètres est constitué d'au moins 2 lignes :
– un nombre quelconque de lignes de commentaires dont le premier caractère est une apostrophe; ces lignes sont ignorées,
– une ligne spécifiant la liste des statistiques à produire, dans l'ordre désiré (Step tocc Nsplit Nbal Nmerge Nerror Np Nr Nop Nins Ndel),
– une ou plusieurs lignes de paramètres décrivant chacune une simulation.

Le simulateur crée dans le répertoire "Mes documents" un fichier pour chaque ligne de paramètres. Le nom du fichier résultat indique le fichier de paramètres source, un numéro de séquence ainsi que la valeur des paramètres. Le texte ci-dessous est le contenu du fichier de paramètres "ISAM-param-DUNOD01.txt" qui a permis d'obtenir les statistiques τ, **N_p** et **n_{split}** dont les deux premières sont présentées graphiquement dans les figures A4.12 et A4.13.

```
' Courbes des figures A4.12 et A4.13
' Paramètres à spécifier : Np0, Mrpp, Nrpp0, %ins, Npas, Noppas, Nsimul
tocc Np Nsplit
5000, 40, 32, 51, 200, 1600, 10
5000, 40, 32, 90, 200, 1600, 10
```

Le simulateur produit les deux fichiers de statistiques ISAM-Stat-DUNOD01-01-(Np0=5000;Mrpp=40;Nrpp0=32;%ins=51;Nop=320000).txt et ISAM-Stat-DUNOD01-02-(Np0=5000;Mrpp=40;Nrpp0=32;%ins=90;Nop=320000).txt. Le (début du) contenu du premier fichier est donné ci-dessous :

Start time =	28/08/2011 23:46:50	
End time =	29/08/2011 5:17:47	
Np0 =	5000	
Mrpp =	40	
Nrpp0 =	32	
%ins =	51	
NbreSimul =	10	
NbrePas =	200	
NbreOpPas =	1600	
NbreOp =	320000	
tocc	Np	Nsplit
0,8000001	5000	0
0,8000841	5000	0,1
0,800214	5000	0,2
0,800303	5000	0,4
0,8005179	5000	0,5
0,8006319	5000	0,5
0,8007939	5000	0,5
0,8009169	5001	0,7
0,8010718	5001	0,7
0,8012408	5001	0,8
0,8013917	5001	0,9
0,8013386	5002	1,5
etc. (suite des 200 pas)		

Les paramètres et les statistiques enregistrées par le simulateur ne sont pas indépendants. Nous allons montrer en particulier que les statistiques τ, N_p, n_{bal} et n_{split} sont suffisantes pour dériver la plupart des autres (à l'exception de n_{error} bien sûr).
Des paramètres N_{p0}, M_{rpp}, et τ_0 on tire N_{rpp0}, le nombre initial d'enregistrements par page et N_{r0}, le nombre initial d'enregistrements :

$$N_{rpp0} = M_{rpp} \times \tau_0$$
$$N_{r0} = N_{p0} \times N_{rpp0}$$

Des statiques τ et N_p, il vient, pour le *pas* courant :

$$N_{rpp} = M_{rpp} \times \tau$$
$$N_r = N_p \times N_{rpp}$$

Et aussi, en désignant par ΔN_r l'accroissement du nombre d'enregistrements et ΔN_p celui du nombre de pages pour le pas courant :

$$\Delta N_r = n_{ins} - n_{del}$$
$$\Delta N_p = N_p - N_{p0}$$
$$N_r = N_{r0} + \Delta N_r$$
$$n_{op} = n_{ins} + n_{del}$$

Nous disposons ainsi de deux relations impliquant n_{ins} et n_{del}, ce qui nous permet de les exprimer pour le *pas* courant :

$$n_{ins} = (\Delta N_r + n_{op}) / 2$$

$$n_{del} = n_{ops} - n_{ins}$$

Chaque éclatement ajoutant une page et chaque fusion en en supprimant une, on a aussi, au *pas* courant :

$$n_{merge} = n_{split} - \Delta N_p$$

Pour l'étude générale du comportement d'un fichier, ce n'est pas tant les nombres d'éclatements et de fusions qui importent que ces valeurs ramenées à l'unité, c'est-à-dire τ_{split} la proportion d'opérations d'insertion qui provoquent un éclatement, τ_{bal} la proportion de suppressions qui provoquent un rééquilibrage et τ_{merge} la proportion de suppressions qui provoquent une fusion. On définit donc, pour le pas courant[18] :

$$\tau_{split} = n_{split} / n_{ins}$$

$$\tau_{bal} = n_{bal} / n_{del}$$

$$\tau_{merge} = n_{merge} / n_{del}$$

Il nous reste à dériver le taux de ruptures π_r. Chaque éclatement créant une page hors séquence, on peut estimer[19] ce taux comme suit :

$$\pi_r = n_{split} / N_p$$

Les grandeurs discutées jusqu'ici sont *observées* puisqu'elles sont valuées au terme de chaque pas de la simulation. On pourrait en estimer la valeur a priori, avant simulation. On devrait donc avoir :

$$n_{ins} \cong n_{ins}' = \tau_{ins} \times n_{op}$$

$$n_{del} \cong n_{del}' = (1 - \tau_{ins}) \times n_{op}$$

$$N_r \cong N_r' = N_{r0} + (2 \times \tau_{ins} - 1) \times n_{op}$$

Ces relations sont évidemment approchées et ne correspondront sans doute pas exactement avec les valeurs produites par le simulateur[20].

La validité des statistiques produites par le simulateur est soumise à certaines conditions qu'il convient de prendre en compte. Si ces conditions ne sont pas satisfaites, il n'est pas garanti que le comportement du fichier soit correctement simulé.

18. On convient que ces taux valent 0 lorsque le dénominateur est nul.
19. Cette expression n'est pas tout à fait exacte car elle ignore la situation d'une page qui fusionne avec une page (logiquement) voisine qui est elle-même hors séquence. Dans ce cas, la page hors séquence disparaît (par exemple, dans la fusion des pages 102 et 103 à la figure A4.9, la page 102 disparaît). La grandeur n_{split} est donc une estimation par excès du nombre de pages hors séquence.
20. Le choix de l'opération courante étant réalisé par tirage aléatoire, on ne peut espérer qu'à chaque pas, la proportion d'opérations d'insertion déjà réalisées soit **exactement %ins**.

- Les simulations s'appliquent à des fichiers contenant au départ un nombre *raisonnable* de pages et d'enregistrements. On interprétera donc avec prudence le traitement de fichiers vides ou pratiquement vides.
- Il en sera de même de taux d'insertion **%ins** inférieurs à 50%, cas de figure peu réaliste d'un fichier en processus de disparition !
- Toutes les pages (sauf la dernière) sont supposées contenir au moins 50% d'enregistrements ($\tau_0 \geq 0{,}5$).

A savoir également :

- Une opération d'insertion est simulée comme suit : un nouvel enregistrement du fichier est inséré devant un enregistrement de référence choisi de manière aléatoire dans l'état courant du fichier. La page affectée est identifiée par sommation des contenus des pages successives du fichier jusqu'à obtenir un nombre égal ou dépassant le numéro de l'enregistrement de référence[21]. En cas d'éclatement, la nouvelle page est insérée devant la page en débordement, simulant ainsi la séquence logique des pages et non leur séquence physique[22]. Le simulateur n'est pas conçu pour fonctionner en régime de *chargement*, où chaque enregistrement est inséré dernière le dernier inséré, ni pour traiter de manière conforme les suites d'insertions d'enregistrements de valeurs de clé consécutives.
- Une opération de suppression est simulée comme suit : l'enregistrement à supprimer est choisi de manière aléatoire dans l'état courant du fichier. La page affectée est identifiée comme ci-dessus. En cas de rééquilibrage, on utilise la page courante et sa suivante, sauf pour la dernière page du fichier, auquel cas on utilise la page courante et sa précédente. En cas de fusion, la page vide est retirée du fichier (le fichier ne contient pas de pages vides).
- Le choix d'une opération et la position dans le fichier de l'enregistrement cible utilisent le même générateur aléatoire. Les opérations utilisent les tirages impairs et les enregistrements les tirages pairs. Le générateur est réinitialisé de manière aléatoire avant chaque simulation. Les résultats ne sont donc pas reproductibles.
- Aux éventuelles imperfections du générateur de nombres aléatoires utilisé par le simulateur s'ajoutent un certain nombre d'hypothèses simplificatrices (1) sur les règles de gestion du SGBD réel, (2) sur les contraintes technologiques de l'environnement (matérielle et logicielles) dans lequel le SGBD s'exécute et (3) sur les conditions réelles d'exploitation de la base de données au moment où on observerait le comportement du fichier. Dans ces conditions, il faut prendre les résultats des simulations avec un certain recul et rester critique en ce qui concerne la précision des valeurs produites, Par exemple des taux d'occupation de 0,6978 et de

21. Pour les deux opérations, le choix aléatoire de la **page** affectées accélérerait considérablement la simulation mais ne serait pas correct. En effet, la probabilité qu'une page soit affectée dépend du nombre d'enregistrements de cette page.
22. Toutes les pages vides obtenues à la suite d'une fusion se retrouvent à la fin du fichier. Lors d'un éclatement, le simulateur ne peut pas utiliser la plus proche des pages laissées libres à la suite d'une fusion.

0,7013 sont à interpréter comme "*pratiquement 70%*". Plus que la précision absolue des valeurs, c'est l'allure générale de l'évolution des statistiques qui est importante.

A4.4.7 Evolution des taux d'occupation τ_b *et* τ_i

Les taux d'occupation τ_b et τ_i jouent un rôle essentiel dans l'expression des volumes et des temps d'accès d'un fichier séquentiel indexé. Leurs valeurs vont pratiquement osciller entre 50 et 100%, de sorte que le volume, par exemple, tout comme le temps de lecture séquentielle, peuvent varier du simple au double et que l'index peut compter un niveau en plus ou en moins. Il est donc très important de comprendre comment ces taux évoluent au cours de la vie du fichier en fonction de certains paramètres du fichier. On en tirera en particulier des règles de gestion des fichiers séquentiels indexés et de leurs dérivés.

Nous ne distinguerons pas le fichier de base de son index, leur comportement étant strictement identique puisqu'ils sont tous deux soumis aux mêmes algorithmes de gestion.

a) Analyse des premières statistiques

Examinons deux scénarios extrêmes mais réalistes, celui d'un fichier en *forte croissance* (90% d'insertions et 10% de suppressions) et celui d'un fichier en *légère croissance* (51% d'insertions pour 49% de suppressions. On travaille sur un fichier de 5 000 pages chargées initialement à 80% (32 enregistrements par page). On convient d'effectuer les deux simulations pour un nombre d'opérations égal au double du nombre initial d'enregistrements (320 000), en 200 pas de 1 600 opérations. Chaque scénario sera exécuté 10 fois. Les paramètres des simulations sont les suivants :

```
Np0=5000, Mrpp=40, Nrpp0=32, %ins=90, Npas=200, Noppas=1.600, Nsimul=10
Np0=5000, Mrpp=40, Nrpp0=32, %ins=51, Npas=200, Noppas=1.600, Nsimul=10
```

La figure A4.12 montre l'évolution du taux d'occupation moyen du fichier selon ces deux scénarios tandis que la figure A4.13 représente l'évolution de la taille du fichier (qui peut d'ailleurs se déduire de la précédente).

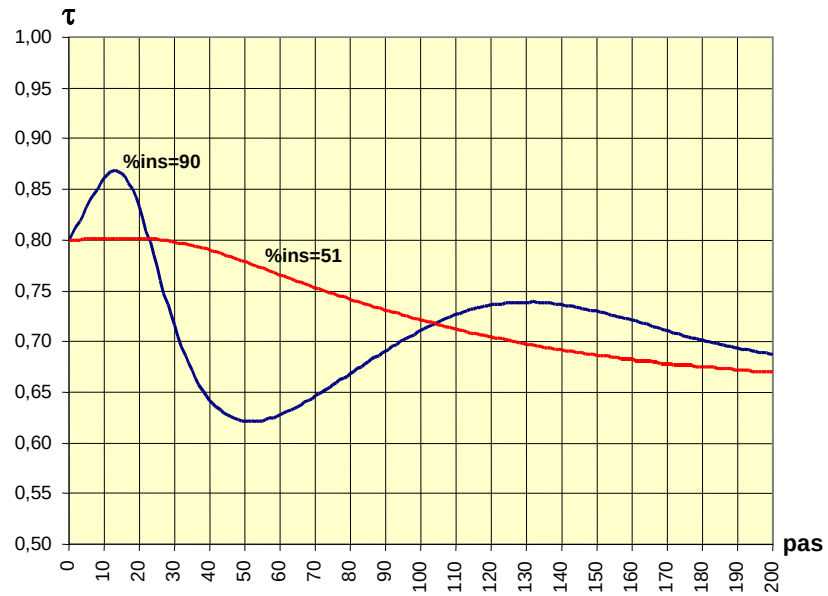


Figure A4.12 - Évolution du taux d'occupation τ d'un fichier [Np0 = 5.000; Mrpp = 40; τ0 = 0,8; %ins = **90** et **51**; Nop = 320.000]

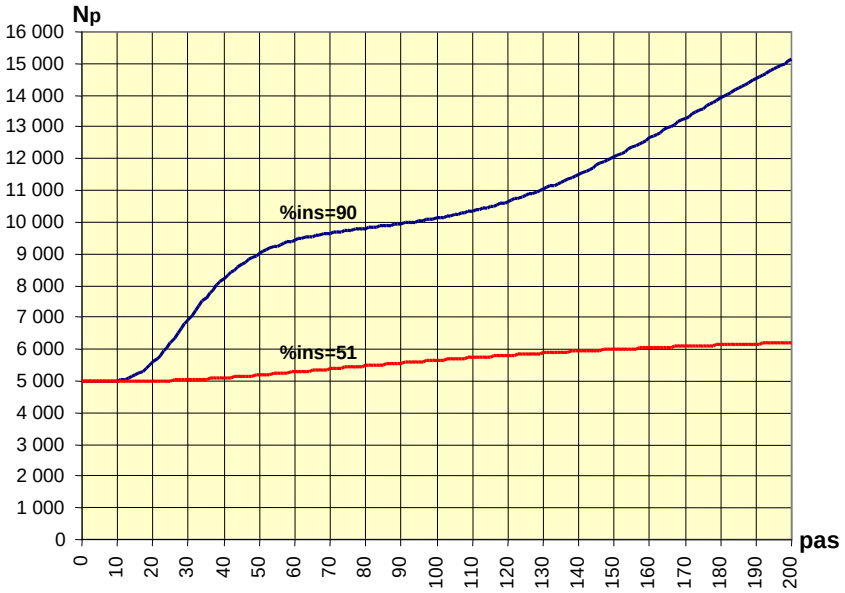


Figure A4.13 - Evolution du nombre de pages **Np** d'un fichier [Np0 = 5.000; Mrpp = 40; τ0 = 0,8; %ins = **90** et **51**; Nop = 320.000]

- **Scénario 1 - %ins = 90**

  Le taux d'occupation évolue selon une courbe oscillante d'amplitude décroissante comprise entre 0,87 et 0,62. Initialement, au **pas** = 0, ou **nop** = 0, le fichier contient 5 000 pages de 32 enregistrements, soit 160 000 enregistrements. Chacun des 200 pas correspond à 1.600 opérations, dont, en moyenne, 1 440 insertions et 160 suppressions conduisant à un accroissement moyen de 1.280 enregistrements. Chaque opération ajoute en moyenne $(2 \times \tau_{ins} - 1) = 0{,}8$ enregistrement.

  - Les graphiques nous disent qu'au **pas** = 50 (**nop** = 80.000), le taux d'occupation (moyen, toujours) est $\tau \cong 0{,}62$ et le nombre de pages $N_p \cong 9\,000$, d'où on tire $N_r \cong 223\,200$. Plus précisément, la simulation indique, au pas 50, $\tau = 0{,}6221738$, $N_p = 9\,261$.
  - À la clôture de la simulation (**pas** = 200, **nop** = 320 000), le taux d'occupation est $\tau \cong 0{,}69$, $N_p \cong 15.130$ et $N_r \cong 416\,000$.

  Bien qu'elle ne décrive qu'un cas spécifique, la courbe de l'évolution du taux d'occupation nous permet de faire des observations intéressantes.

  - On observe d'abord que la partie gauche de la courbe montre un comportement perturbé, caractérisé par les oscillations importantes. Celles-ci s'expliquent facilement. Au départ en effet, toutes les pages disposent d'un espace libre de 20%, soit 8 enregistrements. Les toutes premières insertions tombent dans des pages non pleines, sans pratiquement provoquer d'éclatements et donc sans augmentation sensible de leur nombre. La simulation montre qu'au **pas** = 10, on ne compte que 19 éclatements pour 5 019 pages, soit un taux de ruptures[23] $\pi_r = 0{,}038$. Le taux d'occupation croît linéairement jusqu'à dépasser 0,85. Progressivement, un nombre croissant de pages éclatent, provoquant un fléchissement, puis un effondrement du taux d'occupation qui tombe à 0,62 au **pas** = 52, où 4 142 éclatements se sont produits conduisant à $N_p = 9\,122$, soit $\pi_r = 0{,}45$. Au cours de cette chute, un nombre croissant de pages retrouvent de l'espace libre, susceptible d'accueillir des enregistrements sans éclatements. Le taux d'occupation se redresse pour atteindre un nouveau maximum en **pas** = 130.
  - Ces oscillations vont en s'atténuant au fur et à mesure des opérations. Ce phénomène est dû au fait qu'au départ **toutes** les pages disposent d'espace libre et que les phénomènes d'insertion sans éclatement, puis d'insertions avec éclatement surviennent à peu près en même temps. Plus tard, cet espace libre, qui est compris entre 50% et 0%, est distribué de plus en plus inégalement entre les pages, de sorte que cette conjonction de phénomènes ne se produit que de plus en plus rarement.
  - Dans la partie droite, la courbe semble tendre asymptotiquement vers l'horizontale $\tau \cong 0{,}7$. On atteint après un certain nombre d'opérations un équilibre

23. Pour rappel, proportion des pages qui ne sont pas implantées à l'adresse suivant celle de leur page logiquement précédente.

entre les deux scénarios d'insertion. Il reste d'ailleurs à vérifier que le taux converge vers une valeur stable.

- Nous avons ignoré dans ces explications l'effet des suppressions. Elles sont en effet négligeables, puisqu'on n'en compte qu'une pour neuf insertions. Par exemple, au **pas** = 52, on a observé n_{merge} = 22 fusions de pages pour 4 142 éclatements ($n_{merge} = (N_{p0} + n_{split}) - N_p$). Il faudra cependant observer ce qui se passe quand leur nombre devient plus important, comme dans le second scénario.

- **Scénario 2 - %ins = 51**
  La courbe d'évolution de τ présente une tout autre allure qui témoigne de la part beaucoup plus importante des suppressions : 49 suppressions pour 51 insertions.
  - Le taux d'occupation reste pratiquement constant entre les **pas** 0 à 26 (avec cependant un léger maximum de 0,8018 au **pas** 16). Dans cette plage, on observe, pour N_p = 5 024, seulement 25 éclatements et une seule fusion. En moyenne, une page reçoit pratiquement autant d'insertions que de suppressions. La plupart d'entre elles conservent donc leur contenu sans éclatements.
  - À partir du **pas** 26, on observe une diminution presque constante jusqu'au **pas** 100, à partir duquel la courbe se redresse progressivement, suggérant une stabilisation du taux d'occupation.

Quelles seraient les courbes d'évolution de τ et de N_p pour les autres valeurs de **%ins** ? C'est ce que montrent les figures A4.14 et A4.15[24], pour les valeurs de N_{p0} = 5 000, M_{rpp} = 40, τ_0 = 0,8 et **%ins** de 50 à 100.

24. Ces courbes doivent être lues et utilisées dans la version électronique de cette annexe. Elles sont évidemment illisibles dans une version papier en N&B.

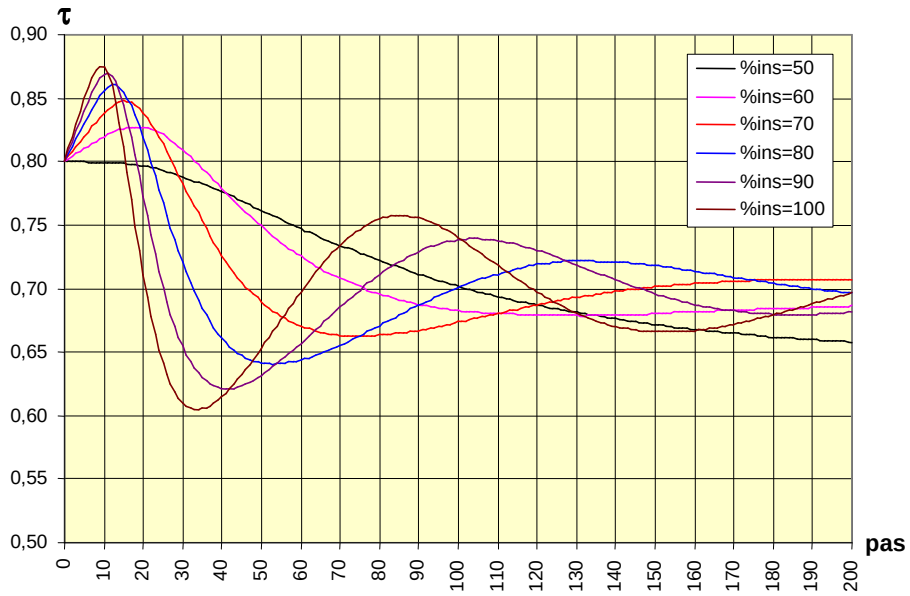


Figure A4.14 - Évolution du taux d'occupation τ d'un fichier [Np0 = 5.000; Mrpp = 40; τ0 = 0,8; %ins = 50 à 100; Nop = 400.000]

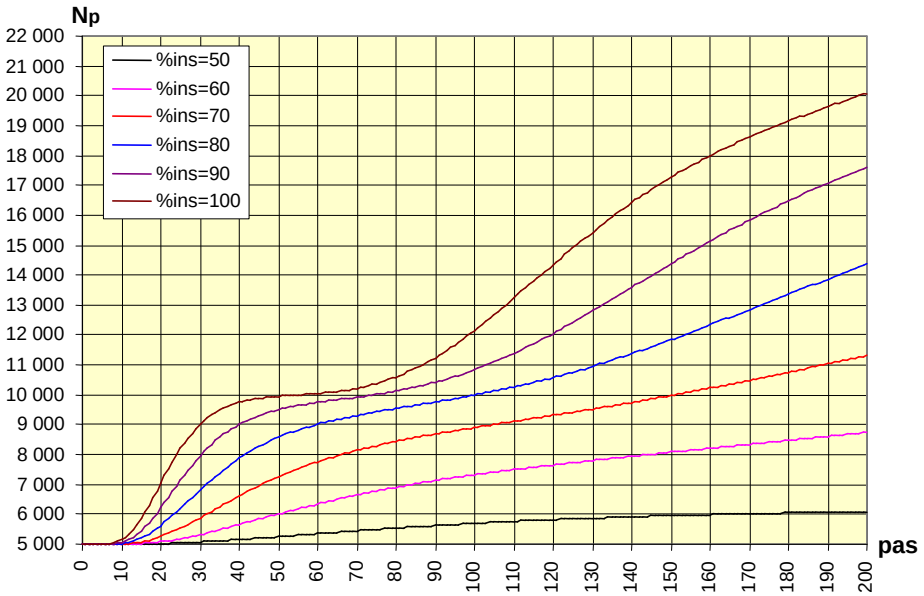


Figure A4.15 - Évolution du nombre de pages **Np** d'un fichier [Np0 = 5.000; Mrpp = 40; τ0 = 0,8; %ins = 50-100%; Nop = 400.000]

Les courbes sont différentes selon la valeur de **%ins** mais on observe qu'elle se déforment de manière continue d'une valeur à la suivante. Le comportement

oscillant s'atténue vers les faibles valeurs de **%ins**. En outre, elles semblent se stabiliser à des valeurs comprises entre 65 et 70%.

b) Les simulations sont-elles extrapolables ?

Nous étudions un phénomène via une simulation basée sur des valeurs relativement faibles et donc pas toujours réalistes, en tout cas pour ce qui concerne la taille du fichier N_{p0} (fichiers de 5 000 pages par exemple). Quelle est la généralité des résultats ainsi obtenus ? Peut-on les extrapoler à des problèmes de taille différente ? La réponse est heureusement positive.

Lorsqu'on effectue une simulation pour des valeurs déterminées de [N_{p0}, M_{rpp}, N_{rpp0}, **%ins**, N_{op}], on observe que les résultats concernant τ sont **absolument identiques** à ceux d'une simulation pour les valeurs [$2 \times N_{p0}$, M_{rpp}, N_{rpp0}, **%ins**, $2 \times N_{op}$]. Plus généralement, pour toutes les valeurs $k > 0$, les simulations [$k \times N_{p0}$, M_{rpp}, N_{rpp0}, **%ins**, $k \times N_{op}$] donnent les mêmes résultats pour τ.

Pour ce qui concerne les autres statistiques, dont N_p et n_{split}, leur extrapolation consiste simplement à les multiplier par **k**.

Ainsi, pour étudier le comportement d'un fichier de 1 000 000 pages pour 100 000 000 opérations, il suffit d'exécuter une simulation [N_{p0}=5 000; M_{rpp}; N_{rpp0}; **%ins**; N_{op}=500 000].

c) Comportement asymptotique de τ

Nous allons à présent nous intéresser au *comportement à la limite* de τ, ce qu'on appelle aussi son *comportement asymptotique*. Les premières observations issues des courbes de la figure A4.12 semblent montrer une stabilisation entre $\tau = 0{,}7$ et $\tau = 0{,}65$. Se confirme-t-elle si on augmente le nombre d'opérations ? Est-elle identique pour d'autres paramètres de la simulation ? C'est ce que nous allons examiner sous forme de quatre questions :

1. existe-t-il un comportement asymptotique ? Et comme tel sera le cas :
2. dépend-il du taux d'insertion **%ins** ?
3. dépend-il du taux d'occupation initial τ_0 ?
4. dépend-il de la taille des pages M_{rpp} ?

Nous établirons ensuite un tableau des valeurs asymptotiques des principales configurations.

• Existe-t-il un comportement asymptotique ?

Pour mieux visualiser ce qui se passe à droite des courbes, poussons la simulation en multipliant par 4 le nombre d'opérations. On obtient les courbes des figures A4.16 et A4.17, qui confirment bien l'intuition initiale[25]. Nous pouvons donc nous convaincre que tous les scénarios ont un comportement asymptotique de la forme

25. Les courbes correspondant aux hautes valeurs de **%ins** présentent encore des traces d'oscillation. De plus longues simulations montreront cependant que leur amortissement conduit à une valeur asymptotique proche de 0,70.

- $\tau = q$ pour le taux d'occupation
- $\mathbf{N_p} = a \times \mathbf{pas} + b$ pour le volume.

Il est clair (courbes A4.14) que la valeur asymptotique de τ dépend du profil des opérations **%ins**. Pour $\mathbf{M_{rpp}} = 40$, les valeurs limites de τ vont apparemment de 70% pour **%ins** = 100% à 64,7% pour **%ins** = 50%, soit une différence significative de plus de 5%. Nous avons aussi admis que les comportements asymptotiques ne dépendent pas de la taille initiale du fichier $\mathbf{N_{p0}}$, ou en tout cas pas du rapport $\mathbf{N_{p0}} / \mathbf{N_{op}}$.

Il nous reste à déterminer si la valeur asymptotique q dépend du taux d'occupation initial τ_0 et de la taille des pages $\mathbf{N_{p0}}$.

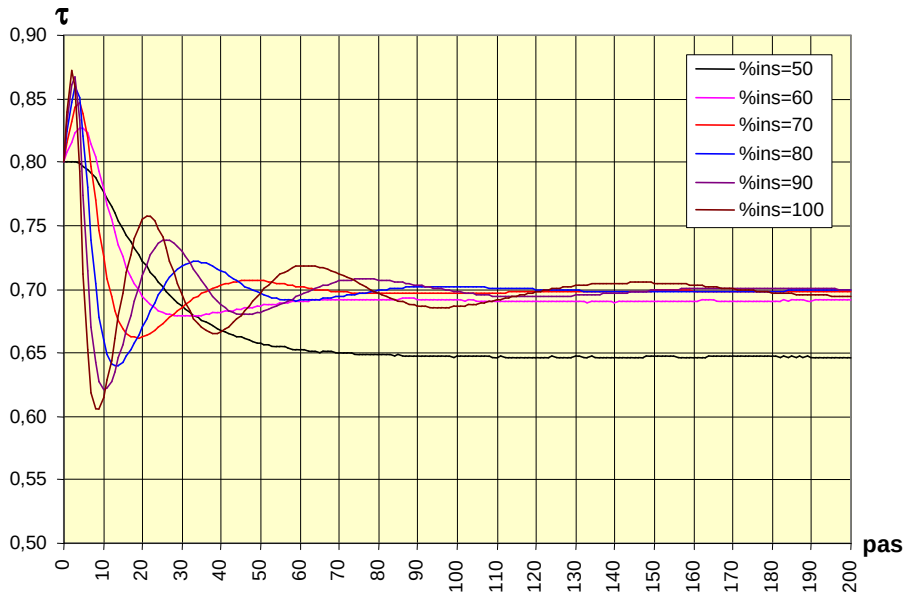


Figure A4.16 - Evaluation du comportement asymptotique de τ (%ins de 50 à 100%)
[Np0 = 5000; Mrpp = 40; τ_0 = 0,8; **%ins = 50-100%**; Nop = 1.600.000]

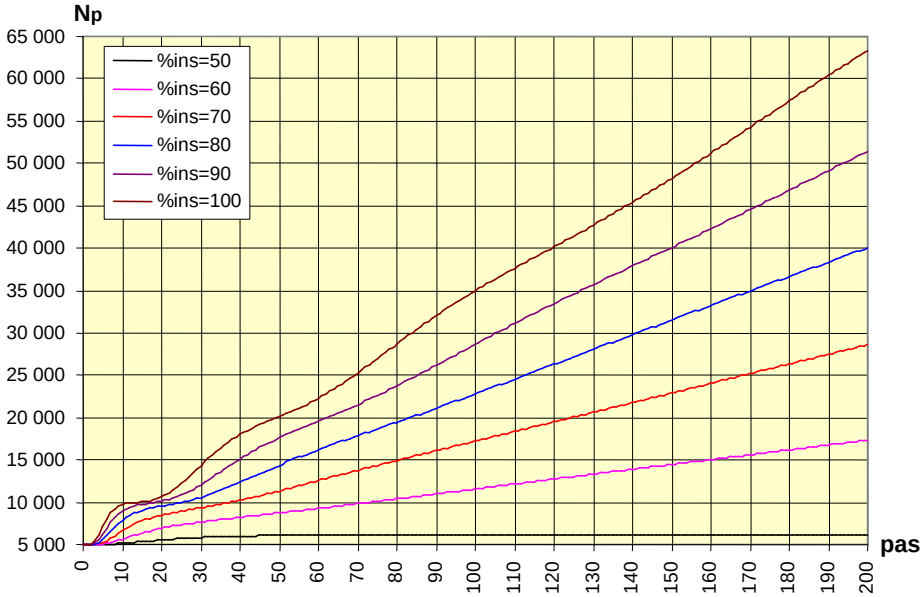


Figure A4.17 - Evaluation du comportement asymptotique de **Np** (%ins de 50 à 100%) [Np0 = 5000; Mrpp = 40; τ_0 = 0,8; **%ins = 50-100%**; Nop = 1.600.000]

• La valeur asymptotique de τ dépend-elle du taux d'occupation initial (τ_0) ?

Dans le comportement asymptotique $\tau = q$, la valeur de q dépend-elle de τ_0, les autres paramètres **Mrpp** et **%ins** restant égaux. Pour ne pas alourdir la discussion, nous ferons deux expériences seulement, à partir des valeurs extrêmes **%ins** = 50% (figure A4.18) et **%ins** = 100% (figure A4.19). Il est patent que, dans ces deux configurations, l'effet de τ_0, qui s'étend de 0,5 à 1, par pas de 0,1, s'estompe rapidement, laissant apparaître une valeur asymptotique commune. Par sécurité, on vérifiera qu'il en est bien ainsi pour les valeurs intermédiaires de **%ins**.

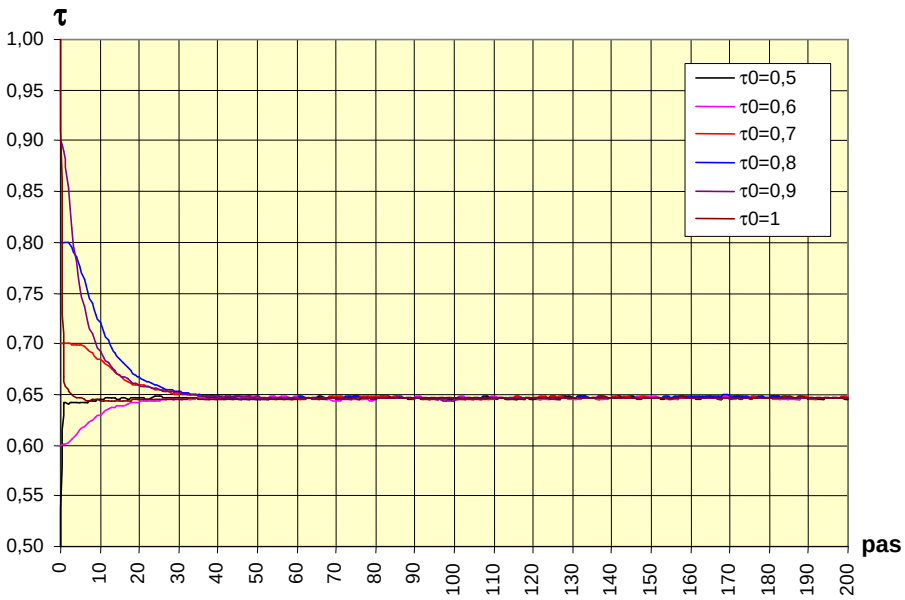


Figure A4.18 - La valeur asymptotique de τ dépend-elle du taux d'occupation initial ?
[Np0 = 2.500; Mrpp = 40; τ_0 = 0,5-1,0; **%ins = 50%**; Nop = 1.600.000]

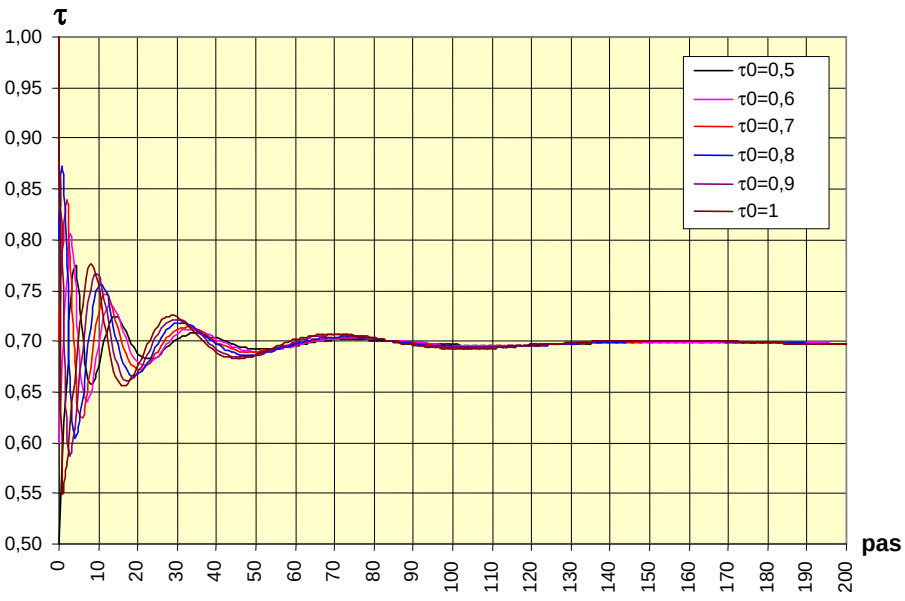


Figure A4.19 - La valeur asymptotique de τ dépend-elle du taux d'occupation initial ?
[Np0 = 2.500; Mrpp = 40; τ_0 = 0,5-1,0; **%ins = 100%**; Nop = 1.600.000]

• La valeur asymptotique τ dépend-elle de la taille des pages (Mrpp) ?

De la même manière, on se demande si la valeur asymptotique q de τ dépend de la taille des pages **Mrpp**, les autres paramètres (τ_0 et **%ins**) restant égaux. La figure A4.20 représente graphiquement les résultats de simulations pour **%ins** = 50% et des valeurs de **Mrpp** de 10 à 100. Bien que le comportement asymptotique ne soit pas encore atteint pour les hautes valeurs de **Mrpp** (la courbe **Mrpp** = 100 est encore en décroissance) on observe des différences significatives de plus de 2,5%, ce qui nous permet de conclure que **q** dépend bien de **Mrpp**.

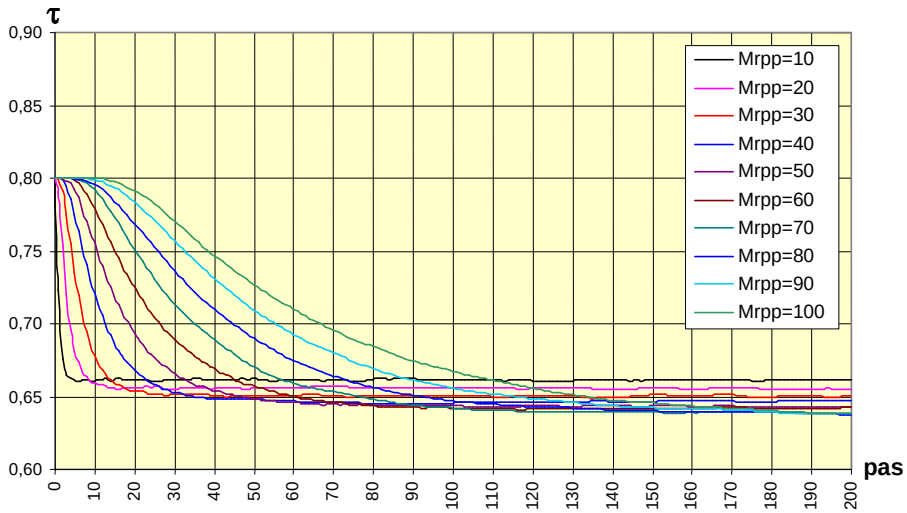


Figure A4.20 - La valeur asymptotique de τ dépend-elle de la taille des pages ? [Np0 = 2.500; Mrpp = 10-100; τ_0 = 0,8; **%ins = 50%**; Nop = 1.600.000]

• Synthèse des valeurs asymptotiques de τ

Il nous reste à mesurer les valeur asymptotique de τ en fonction des variables dont elles dépendent, soit **Mrpp** et **%ins** pour des configurations représentatives. Le tableau de la figure A4.21 reprend les résultats de 121 simulations réalisées pour des valeurs du taux d'insertion de 50% à 100% et pour des valeurs de taille de pages de 10 à 100 [26]. Les faibles valeurs de **%ins**, qui correspondent à des taux de croissance réalistes, ont été détaillées de 50 à 55 par pas de 1.

Ces valeurs ont été converties en deux jeux de courbes (abaques) équivalents. Le premier (figure A4.22) donne le taux en fonction de **Mrpp** pour les différentes valeurs de **%ins** et le second (figures A4.23) donne le taux en fonction de **%ins** pour les différentes valeurs de **Mrpp**. La présentation sous forme de courbes continues (par opposition à un simple *nuage de points*) permet une **interpolation** selon une des dimensions. Ainsi, si nous devons déterminer la valeur asymptotique d'un

26. La valeur asymptotique q est calculée comme la moyenne des taux sur un intervalle stable, par exemple, dans le cas de la figure A4.18, **Nop** = [100-200]. D'autre part, chaque courbe est obtenue comme la moyenne de plusieurs simulations (10 en général) de mêmes paramètres.

fichier dont **Mrpp** = 15 en régime **%ins** = 50, le premier abaque indique $\tau = 0{,}658$. Ou encore, à un fichier dont **Mrpp** = 10 et à un régime **%ins** = 62,5, le premier abaque répond $\tau = 0{,}68$.

		%ins										
		50%	**51%**	**52%**	**53%**	**54%**	**55%**	**60%**	**70%**	**80%**	**90%**	**100%**
Mrpp	**10**	0,6613	0,6628	0,6642	0,6657	0,6676	0,6690	0,6767	0,6891	0,6984	0,7051	0,7103
	20	0,6554	0,6586	0,6612	0,6645	0,6676	0,6700	0,6814	0,6935	0,6982	0,7008	0,7018
	30	0,6505	0,6554	0,6603	0,6645	0,6689	0,6729	0,6870	0,6968	0,6991	0,6997	0,6993
	40	0,6469	0,6539	0,6601	0,6665	0,6723	0,6770	0,6909	0,6983	0,6994	0,6981	0,6978
	50	0,6433	0,6524	0,6616	0,6694	0,6759	0,6810	0,6942	0,6995	0,6990	0,6986	0,6968
	60	0,6407	0,6526	0,6629	0,6723	0,6794	0,6842	0,6973	0,7011	0,6995	0,6983	0,6969
	70	0,6390	0,6522	0,6652	0,6754	0,6818	0,6880	0,6990	0,7018	0,6996	0,6987	0,6963
	80	0,6367	0,6534	0,6674	0,6782	0,6853	0,6906	0,7013	0,7021	0,6998	0,6987	0,6951
	90	0,6351	0,6544	0,6707	0,6810	0,6879	0,6930	0,7025	0,7028	0,6999	0,6979	0,6954
	100	0,6335	0,6557	0,6732	0,6839	0,6903	0,6948	0,7036	0,7032	0,6999	0,6977	0,6949

Figure A4.21 - Tableau des valeurs asymptotiques $\tau = \mathbf{q}$ pour les principaux jeux de paramètres

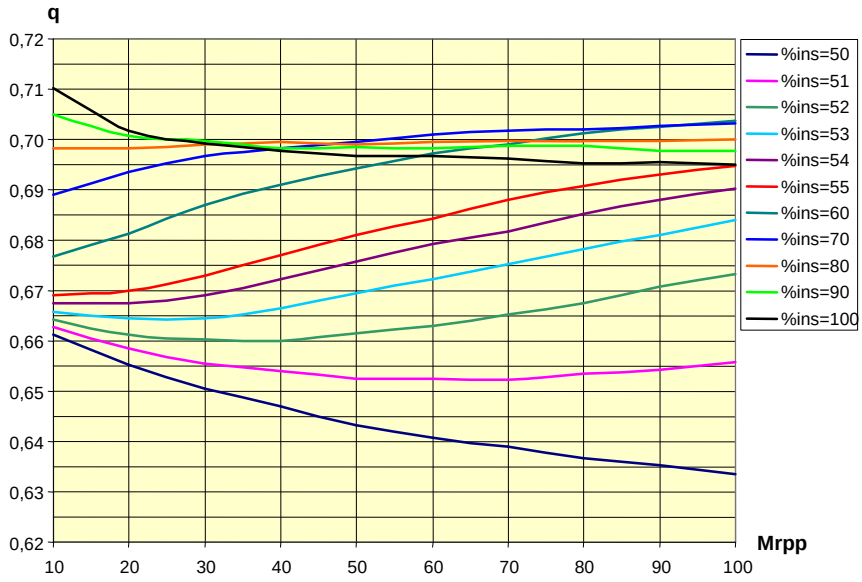


Figure A4.22 - Valeurs asymptotiques $\tau = \mathbf{q}$ en fonction de la taille des pages **Mrpp** pour différentes valeurs du taux d'insertion **%ins**

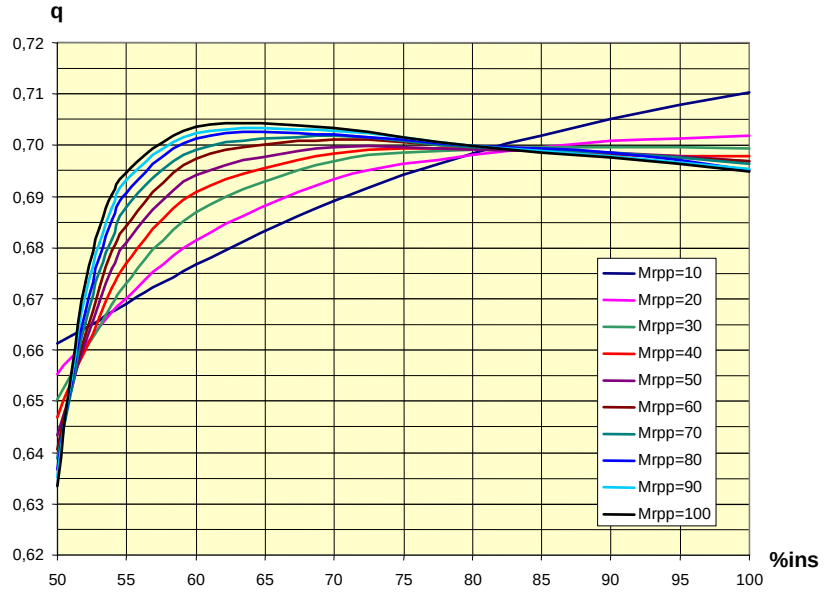


Figure A4.23 - Valeurs asymptotiques τ = **q** en fonction du taux d'insertion **%ins** pour différentes valeurs de la taille des pages **Mrpp**.

Ces résultats permettent d'évaluer le volume *à la limite* d'un fichier séquentiel indexé qui ne subit pas de réorganisation pendant une longue période. Ce comportement se rencontrera rarement en pratique dans une base de données bien gérée mais cette hypothèse permet d'obtenir rapidement une valeur *au pire* (*worst case*) pour une situation concrète.

Exemple

On considère un fichier séquentiel indexé nouvellement créé comprenant 16 000 000 enregistrements de 100 octets stockés dans des pages de 4 Ko. Lors du chargement initial, les pages ont été remplies chacune de 32 enregistrements. Durant la phase d'exploitation qui a succédé à ce chargement, on a effectué chaque jour 200 000 opérations de modification dont la moitié sont des insertions. On aimerait connaître le volume du fichier de base après plusieurs années de fonctionnement sans réorganisation.

On sait que **Mrpp** = 40. En outre, la taille initiale du fichier est de **Np0** = 16 000 000 / 32 = 500 000 pages. On peut admettre qu'après tout ce temps, le fichier a pratiquement atteint son taux d'occupation asymptotique[27]. La figure A4.22 (ou le tableau A4.21) nous donne, pour **Mrpp** = 40 et **%ins** = 50%, la valeur **q** = 0,6469. On en déduit **Nrpp** = 0,6469 x 40 = 25,9. Avec un taux d'insertion de 50%, le fichier est en régime stationnaire, le nombre d'enregistrements restant constant (en moyenne), soit 16 000 000. Le nombre de pages tend donc vers la

27. Après 2 ans, ou 730 jours, le fichier a subi 146 millions d'opérations, soit 9,125 x **Nr0**.

valeur limite 16 000 000 / 25,9 ≅ 640 000, soit une augmentation, à contenu constant, de près de 28%.

d) Évolution de τ - Abaques opérationnels

Lorsqu'on examine à nouveau les courbes de la figure A4.16, on observe que leur partie gauche (jusqu'au pas 60 environ) est sujette à d'importantes perturbations tandis que, dans leur partie droite, ces courbes se stabilisent vers leurs valeurs asymptotiques. Cette partie droite, que nous avons analysée dans la section précédente, correspond à des scénarios d'utilisation du fichier qui seront rares en pratique, notamment à cause d'un taux de rupture π_r excessif, comme nous le verrons plus loin.

Nous allons à présent nous intéresser à la partie gauche des courbes, qui correspond au régime normal de fonctionnement d'un fichier. Les simulations effectuées jusqu'ici sont trop peu détaillées pour que la partie gauche des courbes résultantes puisse être exploitée utilement. Nous allons donc les refaire en limitant les opérations selon la règle suivante : $\mathbf{N_{op}} = 2 \times \mathbf{N_{r0}}$. Par exemple, pour le scénario fréquent **%ins** = 100, on simule une séquence d'opérations conduisant au *triplement* du nombre d'enregistrements. Normalement (nous justifierons cette règle plus loin), un fichier devra être réorganisé bien avant ce moment.

Reportons-nous à la figure A4.12, qui comporte deux courbes dans leur partie *utile* (**N_{r0}** = 160 000 et **N_{op}** = 320 000), et examinons plus particulièrement la courbe **%ins** = 90, reprise à la figure A4.24. Le taux d'occupation évolue selon une loi complexe mais qui, à y regarder de plus près, s'apparente à une *sinusoïde amortie* autour de l'axe $\tau = 0{,}7$, dont la *phase*[28] dépend du taux d'occupation initial τ_0, dont l'*amplitude* est fonction directe de la taille des pages (les grandes pages provoquent de grands *débattements*) et la *période* est une fonction directe de la taille du fichier (la période croît selon son rang; la distance entre deux passages successifs par la valeur asymptotique croît en fonction du nombre de pages)[29].

28. Décalage par rapport à 0° sur le cercle trigonométrique. Ce décalage semble être de 0° pour $\tau_0 = 0{,}7$ et de l'ordre de 90° pour $\tau_0 = 0{,}98$.

29. Un défi pour le lecteur mathématicien : existe-t-il une expression analytique paramétrique de τ, et peut-on la définir par régression sur les résultats des simulations ? Des réponses positives à ces questions auraient peut-être permis d'éviter les 700 heures de calcul qui ont été nécessaires pour compléter le tableau A4.21. C'est que c'est loin l'infini !

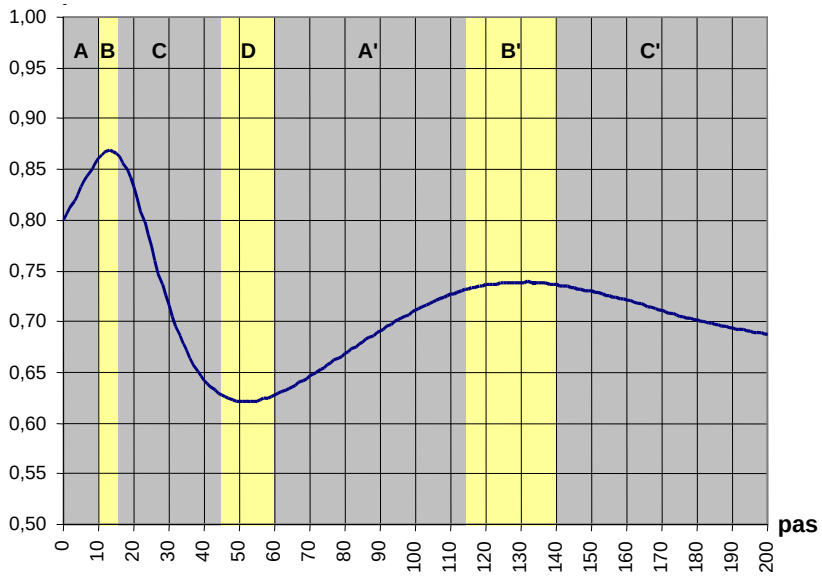


Figure A4.24 - Les phases d'une courbe typique de τ (%ins = 90%; τ0 = 0,8; M_{rpp}=40)

Sur plan du comportement du fichier, on peut identifier dans la courbe de la figure A4.24 des séquences récurrentes de quatre sections.

- Section **A** (intervalle [0-10]). Le taux d'occupation est croissant de 0,8 à 0,87, de manière pratiquement linéaire.
- Section **B** (intervalle [10-15]). La courbe fléchit rapidement pour atteindre son maximum (0,87 en 13), puis le taux commence à décroître.
- Section **C** (intervalle [15-45]). Le taux décroît rapidement, de manière quasiment linéaire.
- Section **D** (intervalle [45-60]). La courbe se redresse progressivement pour atteindre son minimum (0,62 en 52), puis commence à remonter.
- Sections ultérieures (**A'** et suivantes). La courbe remonte rapidement de 0,63 à 0,73, de manière pratiquement linéaire analogue à la section **A**. Les sections **A'** et les suivantes, clairement identifiables dans la figure A4.19 par exemple, reproduisent, avec une amplitude de plus en plus réduite, les quatre sections **A** à **D**, pour converger vers la valeur asymptotique.

Qu'en est-il des autres valeurs de **%ins** ? L'observation des courbes de la figure A4.14 montre que les hautes valeurs provoquent des oscillations initiales prononcées. Au fur et à mesure que la valeur de **%ins** diminue, ces oscillations s'affaiblissent pour disparaître pour **%ins** = 50.

La plage optimale de fonctionnement, du point de vue du taux d'occupation, et par conséquent du volume occupé, est dans cet exemple de 0 à 23, puisque le taux y est égal ou supérieur à 0,8. Ensuite, le taux s'effondre jusqu'à 0,62 et poursuit en

oscillant autour de 0,7. L'évolution des taux d'événements de réorganisation, tels que le taux de rupture π_r (section A4.4.8), confirmera cette hypothèse de *plage optimale*.
Comme nous cherchons à obtenir des informations utilisables (*opérationnelles*) pour évaluer avec une certaine précision les volumes et les temps d'accès de fichiers séquentiels indexés quelconques, nous réaliserons les simulations pour les 550 combinaisons obtenues à partir des valeurs suivantes :

- **%ins** = 50%, 51%, 52%, 53%, 54%, 55%, 60%, 70%, 80%, 90%, 100%
- τ_0 = 60%, 70%, 80%, 90%, 100%
- **M**$_{\mathbf{rpp}}$ = 10, 20, 30, 40, 50, 60, 70, 80, 90, 100

A partir de ces résultats, nous pouvons construire des familles d'abaques[30] qui nous serviront à choisir de manière rationnelle les paramètres des fichiers en projet. Ces simulations représentent de manière tabulaire la fonction

$$\tau = f(\mathbf{\%ins};\ \tau_0;\ \mathbf{M_{rpp}};\ \mathbf{pas})$$

Par exemple, la simulation à l'origine de la courbe de la figure A4.24 correspond à la fonction

$$\tau = f(\mathbf{\%ins}=0{,}9;\ \tau_0=0{,}8;\ \mathbf{M_{rpp}}=40;\ \mathbf{pas})$$

En faisant varier un des trois premiers paramètres de la fonction, les deux autres étant fixés, on obtient un abaque qui permet d'analyser l'influence de ce paramètre. C'est ce que montrent les trois exemples suivants, développés pour un nombre d'opérations égal au double du nombre initial d'enregistrements.

- Une première famille d'abaques est obtenue en fixant pour chacun d'eux les valeurs de **%ins** et de τ_0. Chaque abaque de ce type peut être défini comme suit :

  $$\tau = f(\mathbf{\%ins}=55;\ \tau_0=0{,}7;\ \mathbf{M_{rpp}}=(10\text{-}100);\ \mathbf{pas})$$

  Cet abaque est représenté à la figure A4.25. Il est caractérisé par un profil d'opérations (**%ins**) et un taux d'occupation initial (τ_0). Il comporte une courbe pour chaque valeur de **M**$_{\mathbf{rpp}}$.
- Une deuxième famille est obtenue en faisant varier **%ins** pour chaque abaque (figure A4.26) selon une définition telle que la suivante :

  $$\tau = f(\mathbf{\%ins}=(50\text{-}100);\ \tau_0=0{,}7;\ \mathbf{M_{rpp}}=20;\ \mathbf{pas})$$

- La figure A4.27 représente un membre d'une troisième famille, basée sur la variation de τ_0 :

  $$\tau = f(\mathbf{\%ins}=(55);\ \tau_0=(0{,}6\text{-}1);\ \mathbf{M_{rpp}}=20;\ \mathbf{pas})$$

Le document Abaques-ISAM-Tocc.pdf. présente une suite d'abaques correspondant au premier type. Chaque abaque est caractérisé par un profil d'opérations

30. Dans ce contexte, un abaque est un diagramme composé d'un réseau de courbes représentant des relations entre 3 variables ou plus.

(**%ins**) et un taux d'occupation initial (τ_0). Il comporte une courbe pour chaque taille de page (**Mrpp**). Le nombre d'opérations de chaque simulation est le double du nombre initial d'enregistrements (**Nop** = 2 x **Nr0**). Chacun des 200 pas représentera donc un nombre d'opérations égal à 1% de **Nr0**. Pour les autres combinaisons de paramètres, on procédera par interpolation[31] ou on effectuera une simulation ad hoc.

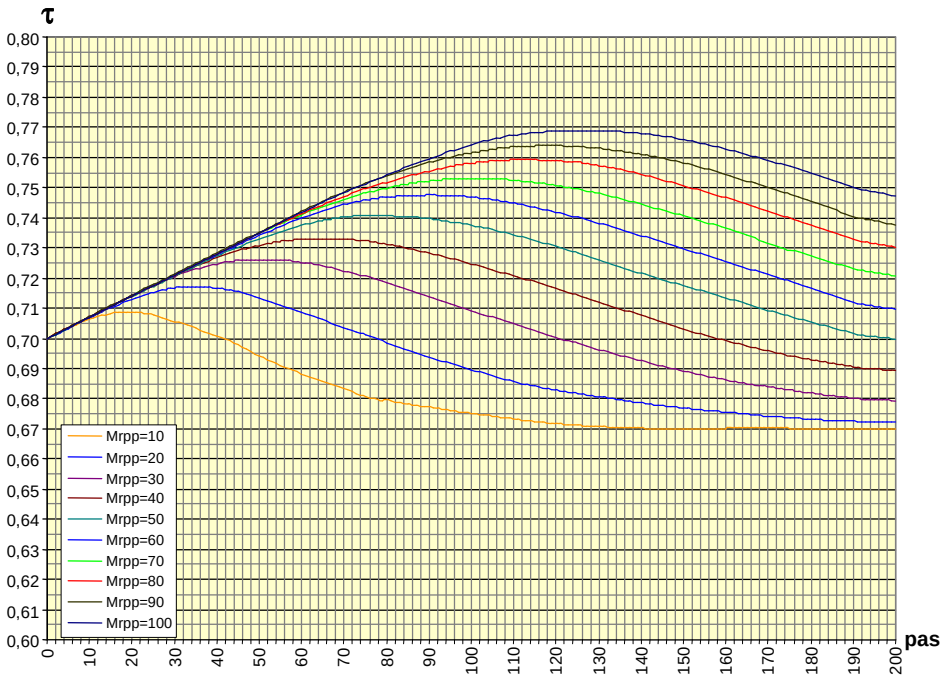


Figure A4.25 - Abaque τ = f(**%ins**=55; τ_0=0,7; **Mrpp**=(10-100); **pas**)

31. Contrairement aux valeurs asymptotiques, l'allure fortement non linéaires des courbes selon **Mrpp** et selon **%ins** rend l'interpolation assez délicate. Il sera préférable dès lors de lancer une simulation sur base des paramètres propres à chaque configuration spécifique.

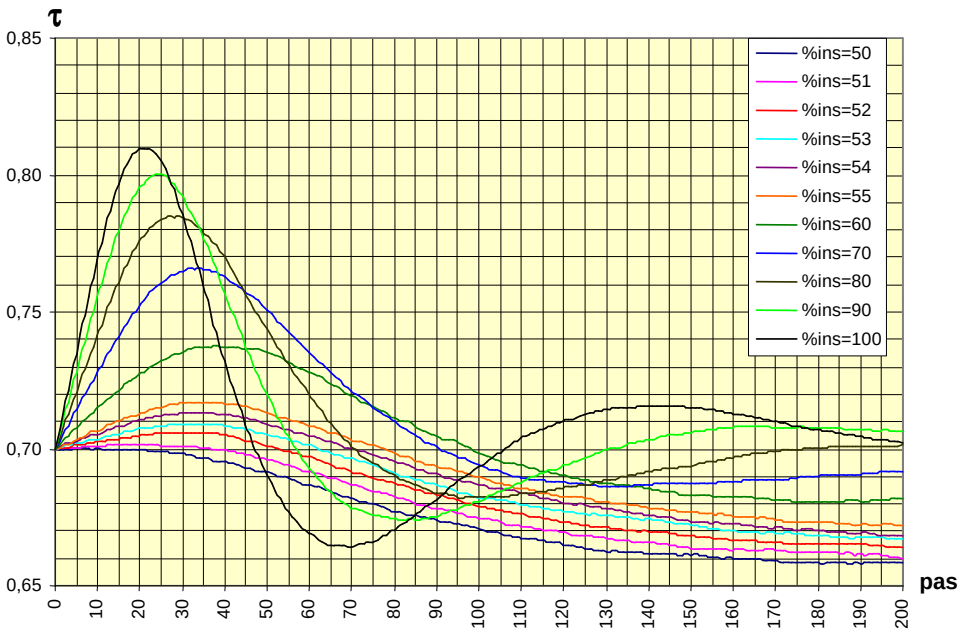


Figure A4.26 - Abaque τ = f(**%ins**=(50-100); τ_0=0,7; **Mrpp**=20; **pas**)

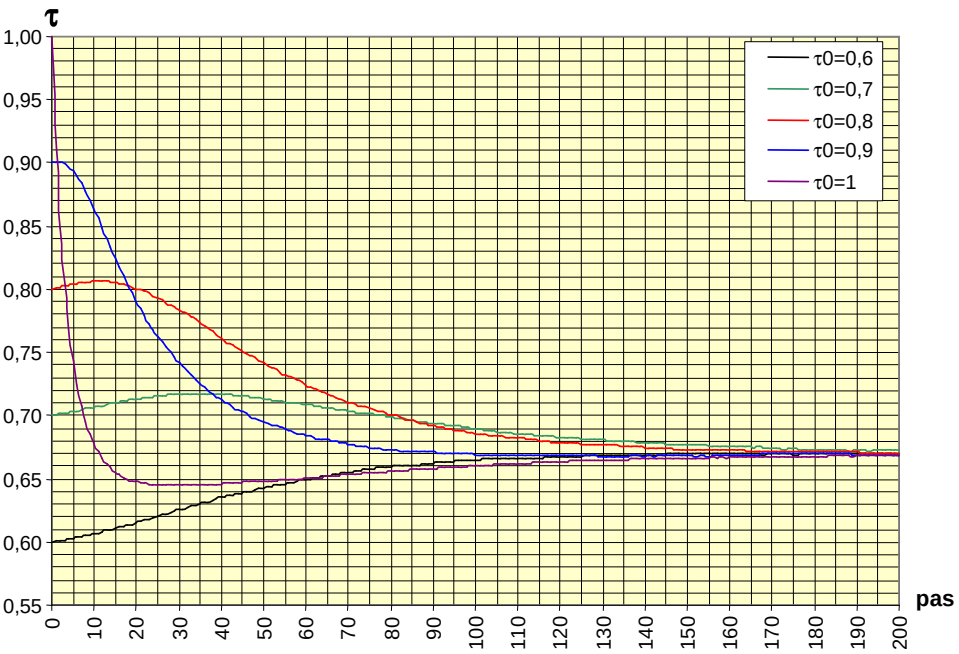


Figure A4.27 - Abaque τ = f(**%ins**=(55); τ_0=(0,6-1); **Mrpp**=20; **pas**)

A4.4.8 Evolution des taux de réorganisation τ_{split} τ_{bal} et τ_{merge}

Si les caractéristiques de performances d'un fichier séquentiel indexé sont dépendantes de son taux d'occupation, on a aussi montré que les événements à l'origine de l'évolution de ce taux (*éclatement*, *rééquilibrage* et *fusion* de pages) interviennent également de manière directe dans certains coûts :

- le temps de lecture séquentielle est dépendant du taux de rupture π_r, lui-même dérivé du nombre d'éclatements n_{split}
- le temps moyen d'une insertion est dépendant du taux de ruptures τ_{split}, obtenu à partir du nombre d'éclatements n_{split}
- le temps moyen d'une suppression est dépendant des taux de rééquilibrage τ_{bal} et de fusion τ_{merge}, obtenu à partir des nombres de rééquilibrages n_{bal} et de fusions n_{merge}.

On a en effet établi que :

$\pi_r = n_{split} / N_p$

$\tau_{split} = n_{split} / n_{ins}$

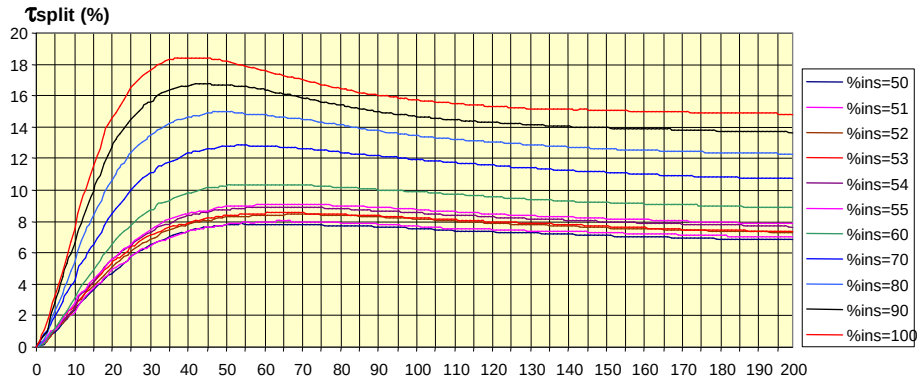
$\tau_{bal} = n_{bal} / n_{del}$

$\tau_{merge} = n_{merge} / n_{del}$

Il est donc important de comprendre et de mesurer ces événements.

a) Insertion d'un enregistrement

Examinons le phénomène d'éclatement dans le processus d'insertion. La figure A4.28 présente graphiquement les résultats de simulations pour des paramètres représentatifs [N_{p0} = 5.000; M_{rpp} = 10-100; τ_0 = 0,8; %ins = 50-100%; N_{op} = 2 x N_{r0}]



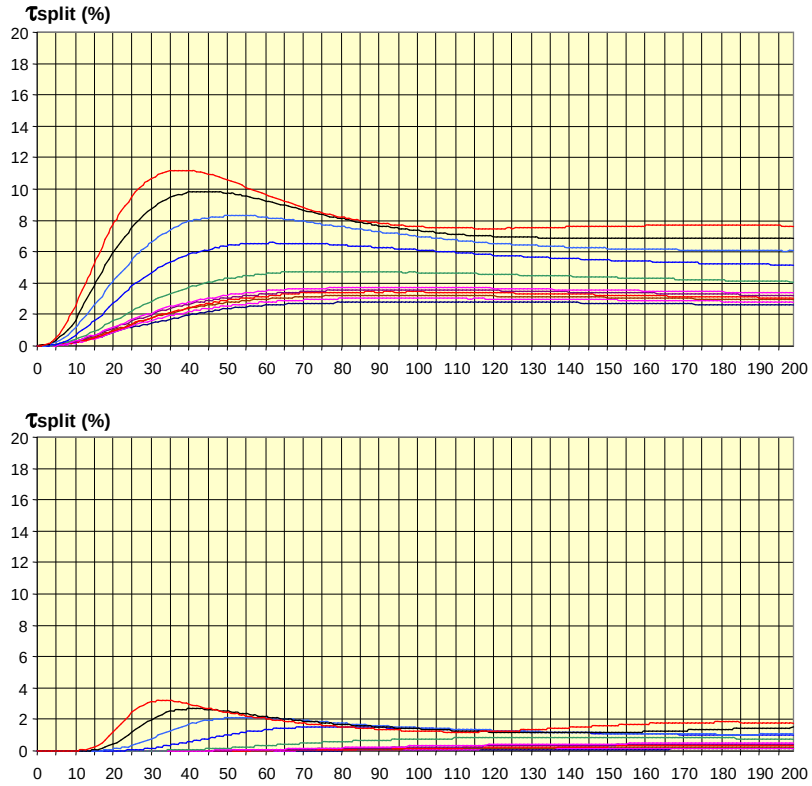


Figure A4.28 - Évolution du taux d'éclatements τ_{split} (en %) en fonction du taux d'insertion **%ins** pour des tailles de pages **Mrpp** = 10, 20 et 100 (de haut en bas) [τ_0 = 0,8]

On peut faire les observations suivantes.

– L'allure générale des courbes est sensiblement la même pour toutes les valeurs de **Mrpp** et de **%ins** : au démarrage, temps de latence fonction directe de la taille des pages, puis croissance pratiquement linéaire, maximum vers les pas 30-40, légère décroissance puis stabilisation progressive.

– A taille de page égale, le taux d'éclatement est d'autant plus important que le taux d'insertion **%ins** est élevé : 0,15 pour **%ins** = 100% à 0,07 pour **%ins** = 50% (valeurs de stabilisation pour **Mrpp** = 10). Ce résultat est conforme à l'intuition : un nombre élevé de suppressions fait baisser le taux d'occupation et raréfie les éclatements.

– A taux d'insertion égal, le taux d'éclatement diminue lorsque la taille des pages augmente : 0,07 pour **Mrpp** = 10 à 0,0015 pour **Mrpp** = 100 (valeurs de stabilisation pour **%ins** = 50%). Ce résultat est également conforme à l'intuition.

En combinant les observations précédentes, on conclut notamment que le phénomène d'éclatement décline pour des valeurs croissantes de **Mrpp** et de **%ins** jusqu'à devenir négligeable. Les tableaux A4.29, A4.29 et A4.29 indiquent les valeurs de

stabilisation[32] respectivement de τ_{split}, τ_{bal} et τ_{merge}, prises au pas 200, pour les combinaisons (**Mrpp**, **%ins**) habituelles.

Ce tableau peut être traduit en courbes tout comme nous l'avons fait des valeurs asymptotiques de τ (figures A4.22 et A4.23). Les abaques des figures A4.32 à A4.34 donnent chacun des taux en fonction de **Mrpp** pour les différentes valeurs de **%ins**, et les abaques des figures A4.35 à A4.37 donnent chacun des taux en fonction de **%ins** pour les différentes valeurs de **Mrpp**[33]. La première série permet une interpolation selon **Mrpp** et la seconde selon **%ins**. On observe que dans le deuxième jeu les courbes de τ_{bal} et τ_{merge} sont incomplètes puisqu'elles s'arrêtent à **%ins** = 90. Il n'y en effet pas de suppressions dans le régime **%ins** = 100. Cependant, ces deux événements se comportent de manière très différentes. Le taux de rééquilibrage τ_{bal} apparaît pratiquement constant pour toutes les tailles de pages et pour les valeurs de **%ins** (sauf les plus basses pour les grandes pages) de sorte qu'il n'existe pas de convergence clairement visible vers **%ins** = 100, où ce taux devrait être 0. Il est probable que la courbe, jouant sur de trop petit nombres de suppressions, devient chaotique à l'approche de cette valeur (à tester avec le simulateur). On pourrait sans doute appliquer une interpolation linéaire jusqu'à 95% mais elle serait moins fiable au-delà. En revanche, le taux de fusions τ_{merge} converge manifestement vers 0 pour **%ins** = 100. On serait donc autorisé à compléter linéairement les courbes entre 90 et 100.

		%ins										
		50%	**51%**	**52%**	**53%**	**54%**	**55%**	**60%**	**70%**	**80%**	**90%**	**100%**
Mrpp	**10**	6,85	7,02	7,29	7,35	7,65	7,84	8,86	10,74	12,30	13,68	14,86
	20	2,62	2,78	2,94	3,08	3,22	3,36	4,10	5,18	6,06	6,89	7,66
	30	1,43	1,58	1,69	1,82	1,97	2,09	2,62	3,38	3,98	4,65	5,27
	40	0,92	1,03	1,14	1,26	1,38	1,48	1,97	2,50	2,92	3,53	4,06
	50	0,62	0,72	0,82	0,92	1,04	1,14	1,55	1,99	2,30	2,86	3,33
	60	0,44	0,54	0,63	0,73	0,82	0,92	1,29	1,64	1,89	2,40	2,83
	70	0,33	0,41	0,50	0,59	0,68	0,76	1,11	1,40	1,59	2,08	2,46
	80	0,25	0,32	0,40	0,48	0,57	0,65	0,97	1,22	1,38	1,83	2,18
	90	0,19	0,26	0,33	0,41	0,49	0,56	0,86	1,09	1,21	1,64	1,96
	100	0,15	0,21	0,28	0,35	0,43	0,50	0,78	0,98	1,07	1,48	1,78

Figure A4.29 - Tableau des valeurs de stabilisation de τ_{split} pour les principaux jeux de paramètres [$\tau_0 = 0,8$]

32. Bien qu'il existe manifestement des valeurs asymptotiques pour ces trois taux, nous n'avons pas utilisé ce terme. En effet, les valeurs asymptotiques ne s'observeront avec une précision suffisante qu'au terme de simulations beaucoup plus longues, telles que $N_{op} = 50 \times N_{r0}$.

33. La seconde série présente d'ailleurs une apparence de quasi-linéarité. Il semble donc possible de remplacer l'usage du simulateur par une série d'équations obtenues par simple régression linéaire. Avis aux amateurs !

		%ins										
		50%	**51%**	**52%**	**53%**	**54%**	**55%**	**60%**	**70%**	**80%**	**90%**	**100%**
Mrpp	**10**	11,30	11,07	11,19	10,87	10,83	10,73	10,45	10,47	10,70	10,93	-
	20	5,77	5,69	5,66	5,59	5,68	5,65	5,88	6,09	6,26	6,51	-
	30	3,35	3,39	3,37	3,55	3,56	3,63	4,04	4,35	4,38	4,50	-
	40	2,10	2,14	2,27	2,34	2,46	2,59	3,05	3,37	3,26	3,48	-
	50	1,38	1,50	1,57	1,69	1,82	1,94	2,49	2,76	2,61	2,87	-
	60	0,94	1,06	1,15	1,30	1,42	1,55	2,10	2,28	2,20	2,49	-
	70	0,68	0,77	0,90	1,02	1,16	1,32	1,87	2,01	1,84	2,07	-
	80	0,49	0,58	0,71	0,82	0,97	1,09	1,61	1,75	1,60	1,86	-
	90	0,38	0,47	0,57	0,70	0,84	0,95	1,45	1,58	1,43	1,67	-
	100	0,27	0,36	0,47	0,58	0,71	0,84	1,33	1,39	1,28	1,50	-

Figure A4.30 - Tableau des valeurs de stabilisation de τ_{bal} pour les principaux jeux de paramètres [$\tau_0 = 0,8$]

		%ins										
		50%	**51%**	**52%**	**53%**	**54%**	**55%**	**60%**	**70%**	**80%**	**90%**	**100%**
Mrpp	**10**	4,15	4,01	3,93	3,71	3,67	3,53	3,06	2,34	1,63	0,99	-
	20	1,30	1,24	1,22	1,18	1,15	1,14	1,09	0,88	0,70	0,37	-
	30	0,56	0,58	0,55	0,56	0,58	0,58	0,60	0,55	0,41	0,21	-
	40	0,30	0,31	0,31	0,32	0,35	0,36	0,43	0,39	0,26	0,15	-
	50	0,16	0,18	0,20	0,21	0,23	0,25	0,31	0,30	0,20	0,12	-
	60	0,10	0,12	0,14	0,15	0,17	0,19	0,25	0,24	0,16	0,10	-
	70	0,07	0,08	0,10	0,11	0,13	0,14	0,22	0,20	0,14	0,08	-
	80	0,05	0,06	0,08	0,09	0,11	0,12	0,19	0,17	0,12	0,07	-
	90	0,04	0,05	0,06	0,07	0,08	0,10	0,16	0,15	0,10	0,07	-
	100	0,02	0,04	0,05	0,06	0,08	0,09	0,15	0,14	0,09	0,06	-

Figure A4.31 - Tableau des valeurs de stabilisation de τ_{merge} pour les principaux jeux de paramètres [$\tau_0 = 0,8$]

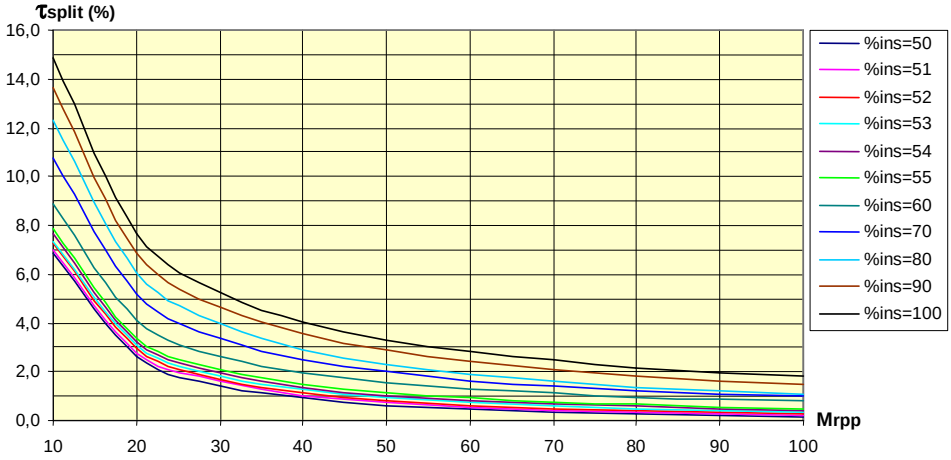


Figure A4.32 - Courbes des valeurs de stabilisation de τ_{split} en fonction de **Mrpp** pour différentes valeurs de **%ins** [τ_0 = 0,8]

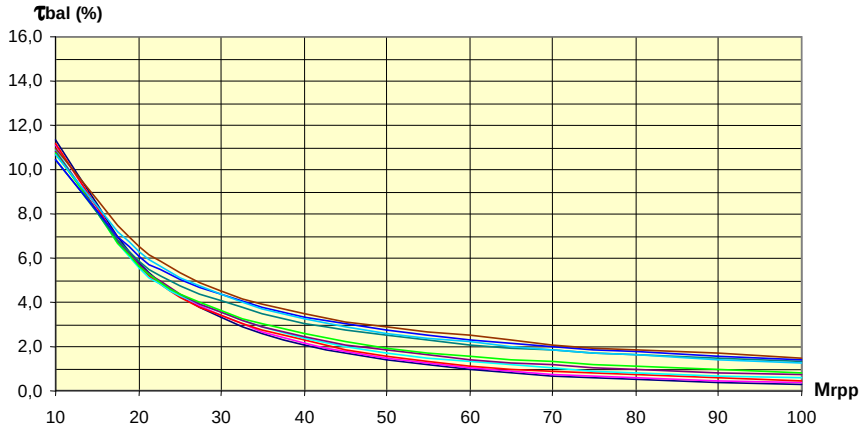


Figure A4.33 - Courbes des valeurs de stabilisation de τ_{bal} en fonction de **Mrpp** pour différentes valeurs de **%ins** [τ_0 = 0,8]

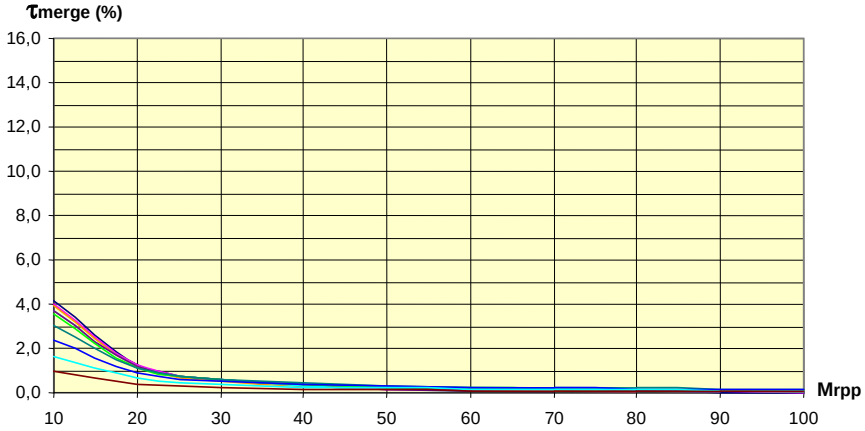


Figure A4.34 - Courbes des valeurs de stabilisation de τ_{merge} en fonction de **Mrpp** pour différentes valeurs de **%ins** [τ_0 = 0,8]

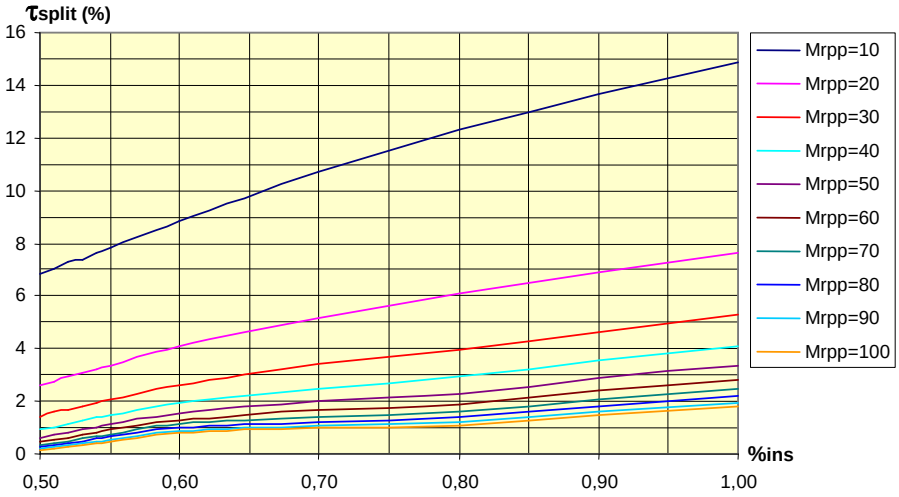


Figure A4.35 - Courbes des valeurs de stabilisation de τ_{split} en fonction de **%ins** pour différentes valeurs de **Mrpp** [τ_0 = 0,8]

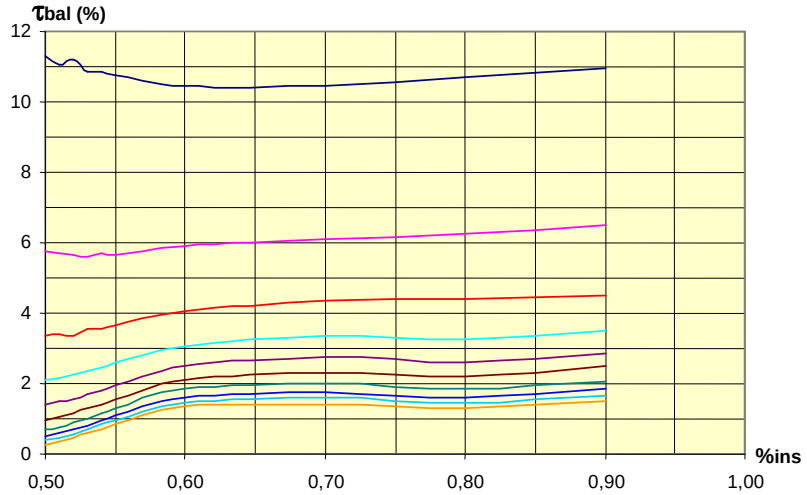


Figure A4.36 - Courbes des valeurs de stabilisation de τ_{bal} en fonction de **%ins** pour différentes valeurs de **Mrpp** [τ_0 = 0,8]

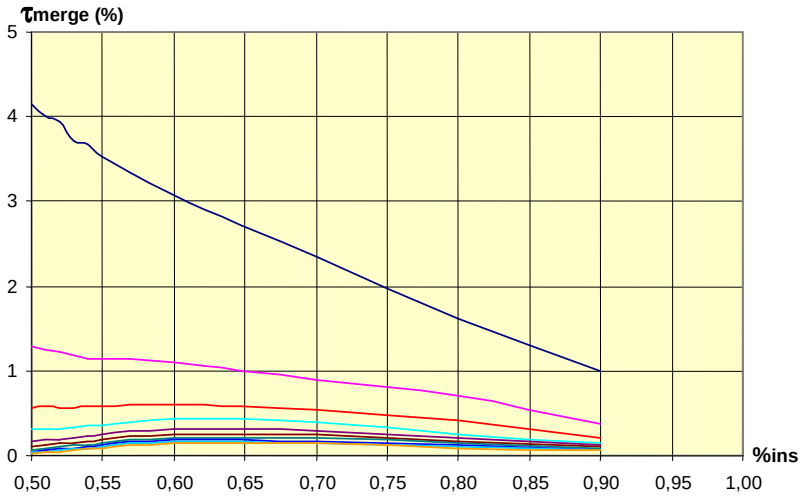


Figure A4.37 - Courbes des valeurs de stabilisation de τ_{merge} en fonction de **%ins** pour différentes valeurs de **Mrpp** [τ_0 = 0,8]

L'analyse que nous venons de développer est basée sur une expérimentation assez large mais toujours pour le taux de chargement initial τ_0 = 0,8. Qu'en serait-il des autres valeurs de τ_0 ?

<À faire : vérifier pour les autres valeurs de τ0 >

Analyse des résultats

On évalue de manière relative les taux de chargement initiaux vis à vis des 5 critères τ , π_r, τ_{split}, τ_{bal} et τ_{merge} . Les notes s'étendent de – – (très mauvais) à + + (excellent). Dans un cas, on utilisera la note de – – – pour exprimer le rejet absolu pour ce critère. Une note globale est proposée. Elle s'obtient en comptant 1 point négatif pour chaque "–" et un point positif pour chaque "+". ce tableau est suivit d'un autre, de même structure, qui commente le score du taux selon chaque critère.

	τ_0				
	60	**70**	**80**	**90**	**100**
τ	– –	0	+ +	–	– – –
π_r	+ +	+	0	–	– –
τ_{split}	+ +	+	0	–	– –
τ_{bal}	0	+ +	+	–	– –
τ_{merge}	0	+ +	+	–	– –
Global	2	6	4	– 5	– 11

	τ_0				
	60	**70**	**80**	**90**	**100**
τ	reste très faible dans la partie utile, en dessous de 69%	reste faible dans la partie utile, entre 70 et 74%	favorable : entre 82 et 70 dans la partie utile	dégradation rapide; atteint 70 deux fois plus tôt qu'à 80%	effondrement très rapide; atteint très vite moins de 64%
π_r	très bon; atteint 10% pour pas = 110	bon; atteint 10% pour pas = 70	moyen; atteint 10% pour pas = 35	mauvais; atteint 10% pour pas = 13	très mauvais; atteint 10% pour pas = 1
τ_{split}	négligeable; 1,25% au pas 100	très faible; 2,7% au pas 100	faible; croissance lente 4,75% au pas 100	important au début (atteint 10%); 6,67% au pas 100	très important au début; puis décroissance 8,45% au pas 100
τ_{bal}	pages faiblement chargées au départ, donc rééquilibrages assez nombreux au début	chargement moyen; donc peu d'éclatements et de rééquilibrages au début; profil idéal	bon; cf. 70% en moins favorable	mauvais; cf. 100% en moins prononcé;	très mauvais; éclatements très nombreux au début; donc beaucoup de pages à 50% et donc beaucoup de rééquilibrages;
τ_{merge}	cf. τ_{bal}	cf. τ_{bal}	cf. τ_{bal}	cf. τ_{bal}	cf. τ_{bal}

Conclusions relatives au choix de τ_0

Il apparaît que la valeur optimale du taux de chargement initial est 70%. Le taux de 80% est le meilleur en ce qui concerne les hautes valeurs de τ , ce qui est très favorables à la minimisation des volumes (y compris l'index) mais entraîne signi-

ficativement plus d'événements de restructuration. Le taux de 60% est le meilleur pour la minimisation de π_r, et optimise donc la lecture séquentielle. Il entraîne en revanche des coûts supérieurs pour les opérations de modification. Les taux de 90% est à éviter si le fichier fait l'objet d'insertions et de suppressions. Le taux de 100% n'est à envisager que si le fichier est en lecture seule et ne subit pas de modifications, ou lorsque les enregistrements sont insérés par valeurs croissantes de la clé. Ce taux de 100% est curieusement la valeur par défaut chez SQL Server de Microsoft. Simple distraction certainement.

A4.5 ORGANISATION CALCULÉE

A4.5.1 Principes de l'accès calculé

A4.5.2 Fonction de calcul d'adresse

A4.5.3 Gestion des débordements

A4.5.4 Caractéristiques et performances d'un fichier à accès calculé

Les paramètres de définition d'un fichier calculé sont le nombre de pages de base **Npb** et la taille de ces pages **Mrpp**. Son fonctionnement est décrit par une troisième grandeur : le nombre moyen d'enregistrements **Nrpp** que la fonction **f** a envoyés dans une page. Ce nombre peut dépasser **Mrpp**. On l'exprimera sous la forme du *taux de chargement* τ_{ch}, qui mesure la proportion de ce nombre par rapport à la taille des pages. Ce taux est compris typiquement entre 0,5 et 2. On a donc **Nrpp** = τ_{ch} x **Mrpp**.

Un petit exemple

Pour un fichier de **Npb** = 500 000 pages de **Mrpp** = 20 enregistrements, un taux τ_{ch} = 0,9 indique qu'on a essayé de stocker en moyenne **Nrpp** = 0,9 x 20 = 18 enregistrements par page, soit 18 x 500 000 = 9 000 000 enregistrements au total. Bien que, dans cet exemple, **Nrpp** < **Mrpp**, un nombre significatif de pages vont déborder (dans 134 000 pages de débordement, soit un accroissement de 27%, comme nous allons le voir), mais ces pages de débordement contiendront peu d'enregistrements (450 000 enregistrements, soit 5% du total). 9 000 000 - 450 000 = 8 550 000 enregistrements se trouvent dans leur page de base et leur accès nécessitera 1 lecture de page. Les autres en exigeront 2. En moyenne, l'accès par clé à un enregistrement réclamera 0,95 x 1 + 0.05 x 2 = 1.05 lectures de page ou 12,9 ms. Cette excellente performance est obtenue au prix d'un taux d'occupation effectif du fichier de 71%, ce qui est assez médiocre.

Un fichier à organisation calculée en cours de fonctionnement est donc décrit par trois grandeurs : **Npb**, **Mrpp** et τ_{ch}. Cependant, les règles qui nous permettront d'étudier le volume et les temps d'accès d'un tel fichier en toute généralité ne dépen-

dront que de deux paramètres, **Mrpp** et τ**ch**. Pour étudier la répartition des enregistrements dans les pages, nous supposerons que, lors de la création d'un enregistrement, toutes les pages de base ont la même probabilité d'être sélectionnées par la fonction **f**. La probabilité **p**(**k**) qu'une page de base reçoive **k** enregistrements obéit donc à une loi de distribution de Poisson[34] de moyenne **Nrpp**. Nous supposerons dans un premier temps (sections a, b et c) que le fichier ne subit que des insertions. Chaque page qui n'est pas la dernière d'une chaîne contient donc exactement **Mrpp** enregistrements.

a) Volume d'un fichier à organisation calculée

Le nombre total d'enregistrements est

$$\mathbf{Nr} = \tau\mathbf{ch} \times \mathbf{Mrpp} \times \mathbf{Npb}$$

La taille d'un fichier, y compris l'espace de débordement, se calcule comme suit :

$$\mathbf{Np} = \tau\mathbf{ap} \times \mathbf{Npb}$$

où τ**ap** exprime le taux d'accroissement du nombre initial de pages (**Npb**) dû au phénomène de débordement (il est de 1,27 dans l'exemple ci-dessus).

Une troisième grandeur nous sera très utile, le taux d'occupation global τ (72% ou 0,72 dans notre exemple ci-dessus). Ce taux mesure l'efficacité d'utilisation de l'espace du disque par le fichier. Il se calcule comme le rapport entre le volume strictement nécessaire (**Nr** / **Mrpp**) et le volume réellement occupé (**Np**) :

$$\tau = \mathbf{Nr} / (\mathbf{Mrpp} \times \mathbf{Np})$$

La seule grandeur inconnue est le taux d'accroissement du fichier τ**ap**. Examinons le taux d'accroissement d'une page qui a reçu **k** enregistrements. Pour $0 \leq \mathbf{k} \leq \mathbf{Mrpp}$, il n'y a pas de débordement de sorte que ce taux est de 1. Pour $\mathbf{Mrpp} < \mathbf{k} \leq 2\,\mathbf{Mrpp}$, il existe une page de débordement, soit un taux de 2. En toute généralité, le taux d'accroissement de cette page est de $\lceil \mathbf{k} / \mathbf{Mrpp} \rceil$. Tenant compte de toutes les valeurs de **k** avec leur probabilité, ce taux moyen est de :

$$\tau\mathbf{ap} = \sum_{\mathbf{k}=1..\mathbf{Nr}} \mathbf{p}(\mathbf{k}) \times \lceil \mathbf{k} / \mathbf{Mrpp} \rceil$$

Les courbes de la figure A4.38 donnent le taux d'accroissement τ**ap** pour les valeurs les plus courantes de taille des pages **Mrpp** et du taux de chargement τ**ch**. Le comportement oscillant des courbes à hautes valeurs de **Mrpp** ne doit pas surprendre. Les sections proches de l'horizontale (τ**ch** autour de 0,5 et de 1,5) correspondent à un nombre important de pages de fin de chaîne faiblement occupées; de nombreux enregistrements y sont accueillis sans provoquer d'augmentation significative de pages, rendant τ**ap** assez stable. Les sections proches de la verticale (τ**ch** autour de 1 et de 2) correspondent à un nombre important de pages de fin de chaîne pleines; un petit nombre d'enregistrements qui y sont envoyés provoquent l'allocation de nouvelles pages presque vides, entraînant une augmentation rapide de τ**ap**. Cet effet

34. En réalité il s'agit d'une loi binomiale, laquelle est approchée par la loi de Poisson pour autant que **Npb** soit suffisamment grand. Loi de Poisson : $\mathbf{p}(\mathbf{k}) = e^{-\mathbf{Nrpp}} \times \mathbf{Nrpp}^{\mathbf{k}} / \mathbf{k}!$

est d'autant plus important que la taille des pages est élevée : la perturbation d'une grande page additionnelle est plus importante que celle d'une petite page.

Soit un fichier de base de 500.000 pages de 40 enregistrements et un taux de chargement de 1,5. La courbe relative à **Mrpp** = 40 indique, pour τ_{ch} = 1,5 , la valeur τ_{ap} = 2. Ce fichier comporte **Np** = 1 000 000 pages dans lesquelles sont stockés **Nr** = 30.000.000 enregistrements. Le taux d'occupation global est τ = 0,75.

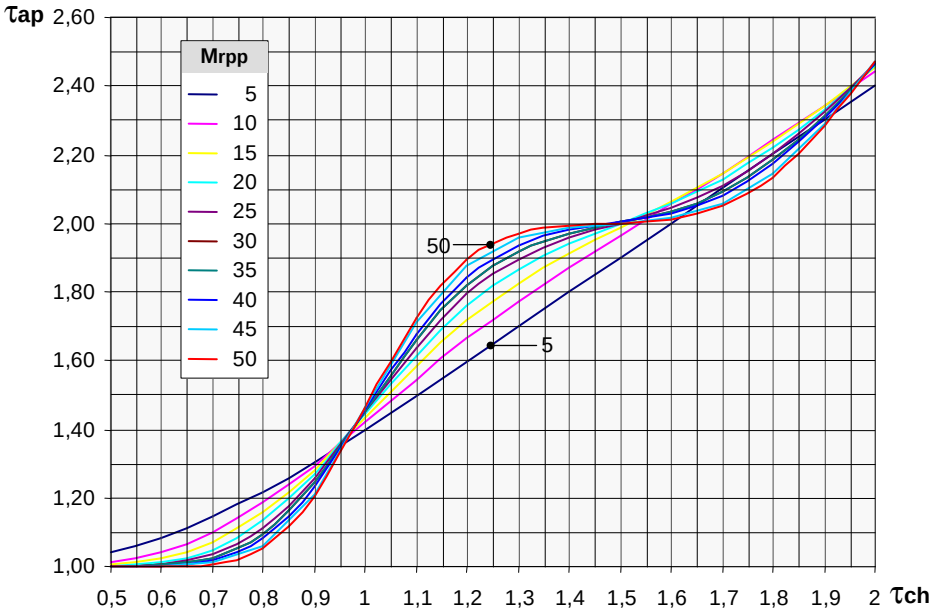


Figure A4.38 - Rapport du nombre total de pages au nombre de pages de base (τ_{ap}), en fonction du taux de chargement τ_{ch} pour les tailles de pages **Mrpp** de 5 à 50

Il est intéressant d'étudier l'évolution du taux d'occupation global τ. Les courbes de la figure A4.39 montre cette évolution en fonction du taux de chargement pour différentes valeurs de la taille de page. Contrairement à celle de τ_{ap} et de **Np**, l'évolution de τ n'est pas monotone pour les valeurs importantes de **Mrpp**. Par exemple, un taux global de 70% sera observé pour les taux de chargement de 70%, 96% et 138%.

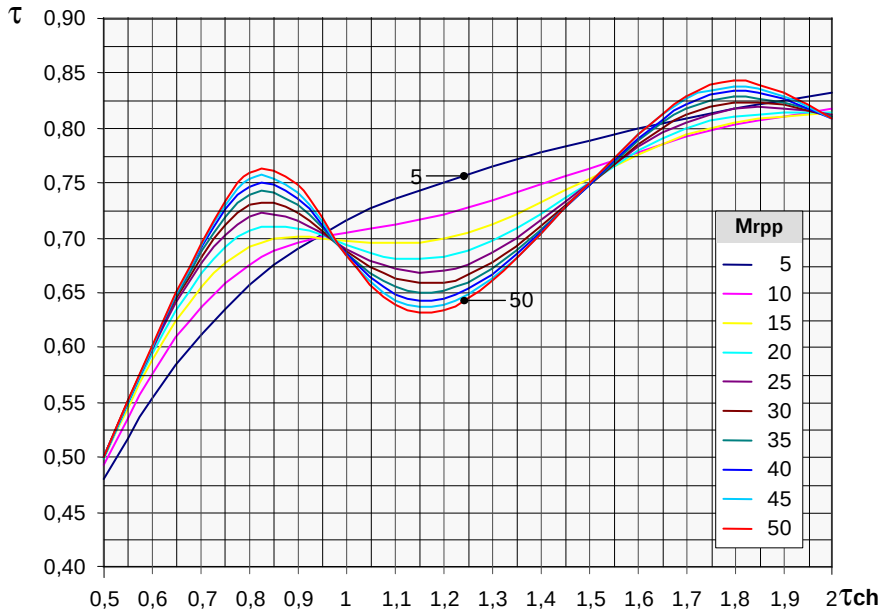


Figure A4.39 - Taux d'occupation global en fonction du taux de chargement de base, selon différentes les tailles de pages de 5 à 50

b) Temps de lecture séquentielle d'un fichier à organisation calculée

L'ordre des pages et des enregistrements dans celles-ci étant non significatif, on effectue une lecture séquentielle des pages de base suivie de celle des pages de débordement. Ces pages étant généralement consécutives sur le disque, on adoptera le temps de lecture séquentielle avec lecture anticipée d'une piste, soit t_{ls1}. Il vient donc :

$$t_{lsf} = N_p \times t_{ls1}$$

c) Temps de lecture par clé d'un fichier à organisation calculée

Le temps de lecture d'un enregistrement r d'adresse de base b dépend de sa position dans la chaîne des pages dont l'origine est la page b. Considérons une page de base qui a reçu **k** enregistrements. Elle est origine d'une chaîne de $n_{pk} = \lceil k / Mrpp \rceil$ pages. La dernière page de cette chaîne contient $k - (n_{pk} - 1) \times Mrpp$ enregistrements et toutes celles qui précèdent, y compris la page de base, en contiennent exactement **Mrpp**. Evaluons en toute généralité le nombre d'enregistrements qui tombent respectivement dans la première page, dans la deuxième page et plus généralement dans la page de rang **n** des chaînes du fichier.

- **n = 1**. Si **k** est compris entre 1 et **Mrpp**, la page de rang 1 contient **k** enregistrements, soit une contribution à la moyenne de

$$\sum_{k=1,Mrpp} k \times p(k)$$

Si **k** est supérieur à **Mrpp**, alors cette page contient exactement **Mrpp** enregistrements, soit une contribution de

$$\sum_{k=Mrpp+1,Nr} Mrpp \times p(k)$$

Le nombre d'enregistrements dans les pages de base (rang 1) est donc

$$n_{lp}(1) = Npb \times \left(\sum_{k=1,Mrpp} k \times p(k) + Mrpp \times \sum_{k=Mrpp+1,Nr} p(k) \right)$$

- **n = 2**. Si **k** est compris entre **Mrpp** + 1 et 2 x **Mrpp**, la page de rang 2 contient **k - Mrpp** enregistrements, soit une contribution de

$$\sum_{k=Mrpp+1,2xMrpp} (k - Mrpp) \times p(k)$$

Si **k** est supérieur à 2 x **Mrpp**, alors cette page contient exactement **Mrpp** enregistrements, soit une contribution de

$$\sum_{k=2xMrpp+1,Nr} Mrpp \times p(k)$$

Le nombre d'enregistrements dans les pages de rang 2 est donc

$$n_{lp}(2) = Npb \times \left(\sum_{k=Mrpp+1,2xMrpp} (k - Mrpp) \times p(k) + Mrpp \times \sum_{k=2xMrpp+1,Nr} p(k) \right)$$

- **n quelconque**. En toute généralité, si **k** est compris entre (**n**-1) x **Mrpp** + 1 et **n** x **Mrpp**, la page de rang **n** contient **k** - (**n**-1) x **Mrpp** enregistrements, soit une contribution de

$$\sum_{k=(n-1)xMrpp+1,nxMrpp} (k - (n-1) \times Mrpp) \times p(k)$$

Si **k** est supérieur à **n** x **Mrpp**, alors cette page contient exactement **Mrpp** enregistrements, soit une contribution de

$$\sum_{k=nxMrpp+1,Nr} Mrpp \times p(k)$$

Le nombre d'enregistrements dans les pages de rang **n** est égal à :

$$n_{lp}(n) = Npb \times \left(\sum_{k=(n-1)xMrpp+1,nxMrpp} (k - (n-1) \times Mrpp) \times p(k) + Mrpp \times \sum_{k=nxMrpp+1,Nr} p(k) \right)$$

L'accès à un enregistrement stocké dans une page de rang **n** réclame la lecture de **n** pages. Grâce à la ventilation des enregistrements selon le rang de leur page, nous pouvons désormais calculer n_{lp}, le nombre moyen de lectures de pages nécessaires pour accéder aux enregistrements :

$$n_{lp} = \sum_{n=1,Npb} n \times n_{lp}(n)$$

$$n_{lp} = \sum_{n=1,Npb} n \times \left(Npb \times \left(\sum_{k=(n-1)xMrpp+1,nxMrpp} (k - (n-1) \times Mrpp) \times p(k) + Mrpp \times \sum_{k=nxMrpp+1,Nr} p(k) \right) \right)$$

Les courbes de la figure A4.40 donnent le nombre moyen de lectures de pages n_{lp} pour les valeurs les plus courantes de taille des pages **Mrpp** et du taux de chargement τ_{ch}.

Reprenons notre fichier de 500.000 pages de 40 enregistrements au taux de chargement de 1,5. La courbe relative à **Mrpp** = 40 indique, pour τ_{ch} = 1,5, **nlp** = 1,33. L'accès à une page coûtant 12,3 ms, le temps moyen d'accès par clé à un enregistrement est de 16,4 ms. Le temps minimum est de 12,3 ms et le maximum[35] est de 4 lectures de pages, soit 49,2 ms.

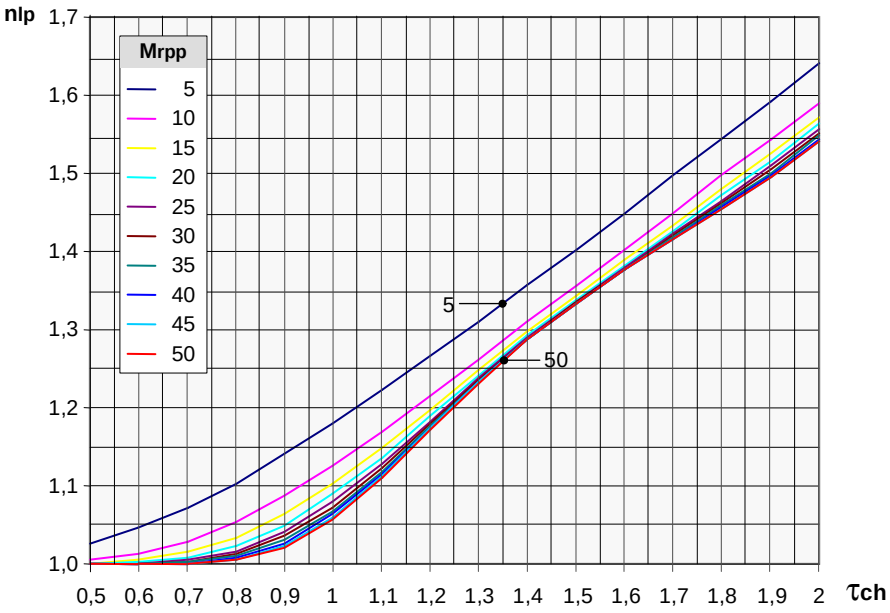


Figure A4.40 - Nombre moyen d'accès physiques (**nlp**) en fonction du taux de chargement de base τ_{ch}, pour les tailles de pages **Mrpp** de 5 à 50

On observe que le temps d'accès est d'autant plus petit que la taille des pages est importante. Cependant, cette amélioration est de plus en plus faible, au point de devenir imperceptible dès que le taux de chargement augmente au delà de 1,2. C'est pour un taux = 1 que l'écart est maximum : il est de 0,12 (soit 1,5 ms) entre **Mrpp** = 5 et **Mrpp** = 50. On en conclut que l'influence de la taille des pages est assez faible sur le temps d'accès par clé dans un fichier à organisation calculée.

d) Évaluation en cas d'insertion et de suppression d'enregistrement

Les études qui précèdent sont basées sur l'hypothèse que les seules opérations de modification admises sont des insertions. Examinons l'influence d'un comportement mixte, comportant des insertions et des suppressions d'enregistrements.

On admet que les suppressions sont aléatoires, et que tous les enregistrements ont la même probabilité d'être visés par une requête de suppression quelconque. L'examen de l'influence des suppressions sur le taux moyen d'occupation dépend avant tout de la proportion d'insertions parmi les opérations de modification (inser-

35. Ce maximum est observé dans la feuille de calcul détaillée qui a permis de construire les graphiques de cette section. Elle est disponible en ligne.

tions et suppressions). Cet examen dépend aussi de la manière dont la recherche d'une place libre est effectuée. Si la chaîne des enregistrements de même adresse de base est ordonnée, cette recherche obéit aux mêmes lois que l'organisation séquentielle indexée, la chaînes des pages de même origine jouant le rôle de (petit) fichier de base. Le taux d'occupation peut dès lors être déduit des courbes relatives à cette organisation, lesquelles sont indépendantes de la taille du fichier. Si, au contraire, la chaînes des enregistrements est laissée en vrac (ce qui accélère la gestion, puisqu'il suffit en cas d'insertion de trouver la première place libre dans la chaîne des pages), pour autant que le fichier soit en croissance (%ins > 50%) on observe que les premières pages de la chaîne seront en général pratiquement pleines (elle accueilleront en priorité les insertions) et que les pages en queue de chaîne pourront comporter de l'espace libre. Ces dernières étant très peu nombreuses, elles concernent un très petit nombre d'enregistrements, de sorte qu'il est raisonnable d'ignorer le phénomène des suppressions et de s'en tenir aux lois proposées dans les sections a, b et c ci-dessus[36].

e) Commentaires

Quelques observations et commentaires s'imposent.

- L'augmentation du taux d'occupation global est obtenu par l'allongement des chaînes de pages, au détriment du temps d'accès. Si ce dernier doit rester en deçà d'un certain seuil (**nlp** = 1,3), on réorganisera le fichier dès que la valeur correspondante de τ_{ch} est atteinte (ici 1,4) en recréant tous les enregistrements dans un espace **Np** plus grand (pour redémarrer à τ_{ch} = 0,8 par exemple, ce qui donne **nlp** = 1,02). Cette réorganisation est un processus très coûteux, puisqu'il exige la lecture et la réécriture d'une page aléatoire (2 x **tla1** ms) par enregistrement. Il existe cependant une technique permettant de réduire de manière significative le temps de réorganisation (voir l'exercice A4.4.6 résolu à l'annexe A).
- Sauf pour les très faibles valeurs (inférieures à 10), l'influence de la taille des pages **Mrpp** est très limitée, tant sur le taux d'occupation que sur le temps d'accès. Ce paramètre n'est donc pas critique dans la construction du fichier.
- Le temps d'accès aux enregistrements n'est pas le même pour tous. La plupart des enregistrements sont dans leur page de base (la situation illustrée par la figure A4.41 est anormalement défavorable) et sont lus en un seul accès au disque, soit 12,3 ms en moyenne. Les enregistrements en débordement sont en général peu nombreux pour autant qu'on veille à ce que le taux de chargement ne dépasse pas un seuil prédéfini.
- Il est possible de lire séquentiellement un fichier calculé, mais l'ordre des enregistrements lus est arbitraire.

36. Le concepteur consciencieux pourra, s'il le juge utile, envisager un taux d'occupation global des pages légèrement inférieur à 1 (0,95 par exemple) pour tenir compte de ce phénomène, qui restera marginal dans la plupart des cas.

- Il peut être utile de trier les enregistrements de même adresse de base. La recherche de l'enregistrement K = k ne nécessitera l'examen que de la *moitié* des enregistrements de la chaîne en moyenne y compris en cas d'absence de l'enregistrement recherché. En revanche, la gestion de ce tri lors des insertions et des suppressions sera plus lourde. Il faut en effet identifier le point d'insertion (page et position dans la page), y insérer l'enregistrement s'il existe de l'espace libre, sinon, insérer une page vide dans la chaîne, rééquilibrer le contenu des pages adjacentes et y insérer l'enregistrement. En cas de suppression, on procédera à une réorganisation locale des pages peu occupées. Néanmoins, dans un fichier bien géré (τ_{ch} limité), la longueur des chaînes de pages de débordement sera très faible en moyenne (typiquement de 0 à 1 pages), ce qui limite l'intérêt de cette optimisation.
- Il existe une grande variété d'organisations calculées. Certaines d'entre elles offrent un temps d'accès constant et même un accès séquentiel ordonné aux enregistrements, au prix de l'ajout d'un index auxiliaire, généralement de très petite taille. On en trouvera une description dans [Knuth, 1998], [Date, 2004][37] ou [Garcia-Molina, 2008] par exemple.

A4.6 LES INDEX SECONDAIRES

A4.6.1 Structure d'un index *bitmap*

Dans un index *bitmap*, la liste de pointeurs est remplacée par une chaîne de bits, le bit de rang **i** désignant le $\mathbf{i}^{e}$ enregistrement dans le fichier. Pour une entrée de valeur de clé k, tout bit de la chaîne correspondant à un enregistrement dont K = k est à 1, sinon il est à 0. Ces chaînes étant très longues (**Nr** bits exactement), une technique de compression sans perte, du type LZW par exemple, est appliquée au moment du stockage[38]. Cette représentation accélère la résolution des requêtes à conditions de sélection multiples : les opérateurs logiques and, or, not sont réalisés par les opérateurs booléens correspondants sur les chaînes de bits.

37. Annexe D, disponible sur http://www.aw-bc.com/cssupport.

38. Les techniques LZW, utilisées dans les archives *zip* ou *rar* par exemple, donnent le meilleur taux de compression. Il existe des techniques spécifiques aux bitmaps, plus rapides en compression/décompression et qui autorisent les opérations booléennes dans la version compressée. BBC (Byte-aligned Bitmap Code) est l'une d'entre elles. Elles donnent cependant des résultats plus volumineux de 50% en général. La littérature sur ces techniques est assez riche depuis une vingtaine d'années. On consultera notamment *Johnson, T., Performance Measurements of Compressed Bitmap Indices, Proc. 25th VLDB Conf., 1999* et *Wu, K., Otoo, E., Shoshani, A., A Performance Comparison of bitmap indexes, in Proc. of CIKM 2001*, disponibles sur le web.

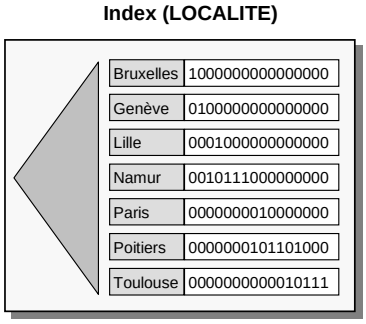


Figure A4.41 - Index *bitmap* sur la colonne LOCALITE (table en vrac de la figure 4.28)

Chaque entrée contient une chaîne de **Nr** bits. Par exemple, pour un fichier de 3 200 000 enregistrements, chaque chaîne occupera 400 000 octets. La comparaison de l'espace occupé par la liste de références d'une entrée est simple :

- par liste de pointeurs : **Lref/p** = **Nrpv** x **Lptr**
- par bitmap sans compression : **Lref/b** = **Nr**/8

En admettant des pointeurs de 6 octets, la technique *bitmap sans compression* est plus économique en volume si **Nr**/8 < **Nrpv** x 6, soit, si **Nrpv** ≥ 0,021 x **Nr**, ou encore si **Nv** < 50. Or, des index reprenant un très petit nombre de valeurs sont le plus souvent inutiles, un simple parcours séquentiel du fichier s'avérant plus rapide, comme nous le montrerons à la section A4.7. Un index *bitmap sans compression* ne présente donc pas d'intérêt.

L'évaluation de la technique par *bitmap avec compression* est moins immédiate. Nous allons procéder par simulation : des chaînes de bits sont générées pour différentes valeurs de **Nr** et de **Nrpv**. Les bits à 1 y sont distribués de manière aléatoire. Ensuite, les chaînes sont compressées par l'utilitaire Winrar[39] et le résultat est mesuré. Les résultats selon les deux techniques sont présentés dans les abaques A4.42 pour la technique *bitmap* compressée et A4.43 pour la technique par liste de pointeurs. On a indiqué en trait épais les zones favorables à chaque technique. Par exemple, pour un fichier de 6 200 000 enregistrements et un index de **Nv** = 12 500 valeurs (**Nrpv(%Nr)** = 100 / **Nv** = 0,008), une chaîne de bits compressée occupe 2 921 octets en moyenne et une liste de pointeurs 3 072 octets en moyenne.

Et qu'en disent les experts ?

Il est intéressant de lire les recommandations des experts Oracle certifiés ou autoproclamés. La plupart suggèrent d'utiliser un index bitmap pour les petites valeurs de **Nv** (de 100 à 1 000 au maximum selon les sources, sans justification), illustrant son usage pour une colonne GENDER (sexe). Certains proposent en revanche de les utiliser pour des index identifiants (**Nv** = **Nr**), tests de performances à l'appui. Ces

39. Le générateur ainsi que la feuille de calcul présentant les résultats précis sous forme numérique et graphique sont disponibles en ligne.

recommandations montrent surtout que les performances en temps d'accès ne constituent pas un critère déterminant distinguant les deux techniques. Deux autres recommandations intéressantes émergent :

- *la gestion d'un index bitmap peut être coûteuse* : lors d'une modification, il faut reconstruire et recopier toute la chaîne de bits des entrées modifiées de l'index. On réservera donc cette technique à des données stables, telles que celles des entrepôts de données. L'exemple évoqué ci-dessus suggère cependant que remplacer deux chaînes compressées de moins de 3 Ko ne sera guère coûteux.
- *les index bitmap se justifient surtout pour exécuter des requêtes à conditions de sélection complexes (and, or, not)*. L'argument avancé est qu'il très efficace d'exécuter les opérateurs booléens sur des chaînes de bits puisque, avance-t-on, les processeurs disposent d'instructions native réalisant ces opérations sur les chaînes de 32, 64 ou 128 bits. On pourrait arguer qu'il existe aussi des algorithmes performants (linéaires) réalisant ces opérateurs sur des listes ordonnées (de pointeurs par exemples).

Nous abandonnerons donc le lecteur à sa perplexité[40] !

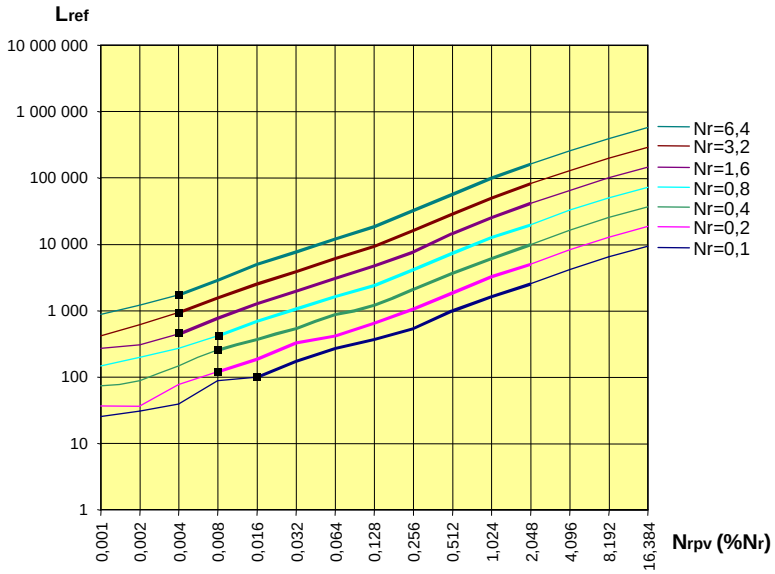


Figure A4.42 - Volume **Lref** des chaînes **bitmap compressées** selon le nombre **Nrpv(%Nr)** de valeurs de l'index, en pourcentage de **Nr**, et la taille du fichier **Nr**, exprimée en millions d'enregistrements

40. Que nous partageons, à notre grand regret.

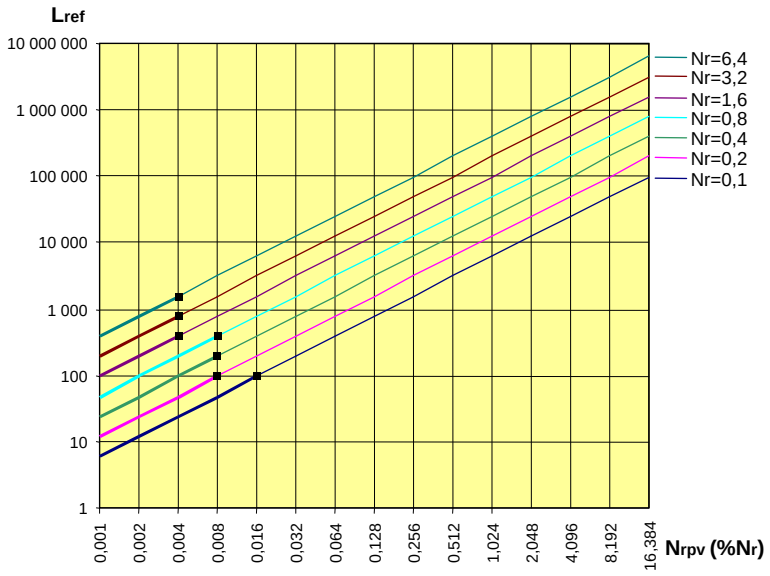


Figure A4.43 - Volume **Lref** des **liste de pointeurs** selon le nombre **Nrpv(%Nr)** de valeurs de l'index, en pourcentage de **Nr**, et la taille du fichier **Nr**, exprimée en millions d'enregistrements

A4.7 ACCÈS PAR INDEX OU RECHERCHE SÉQUENTIELLE ?

Un index sera utilisé pour exécuter une requête pour autant que le SGBD juge son usage plus efficace qu'une simple recherche séquentielle dans le fichier. Considérons la requête suivante :

```
select NCLI, NOM, ADRESSE from CLIENT
where  CAT = 'B1';
```

qui s'adresse à une table de 1 000 000 ennregistrements de 400 octets occupant 140 000 pages de 4 Ko. Supposons que CAT utilise 10 valeurs distinctes et fasse l'objet d'un index. En ignorant la consultation de l'index, l'accès par index à 10% des enregistrements, répartis aléatoirement dans le fichier, coûtera en moyenne (très approximativement)

$$t_{Ixk} = N_{rpv} \times t_{Ia1} = 100\,000 \times 0{,}0123 = 1\,230 \text{ s}$$

Une simple lecture séquentielle coûtera, quant à elle,

$$t_{Isf2} = N_p \times t_{Is1} = 140\,000 \times 0{,}00184 = 26 \text{ s}$$

Il est évident que le SGBD ignorera l'index au profit d'une recherche séquentielle **47 fois plus rapide** ! Cherchons à étudier le phénomène de manière plus précise.

Critères de choix d'un index

A priori, et sans autre analyse, on suggère la création d'un index sur le champ C lorsque la requête suivante, dénommée **Q**,

```
select * from T where C = k
```

est fréquente. On admet deux plans d'accès aux enregistrements du résultat : lecture séquentielle et lecture via un index. Si la lecture séquentielle s'avère plus rapide, le projet d'index est remis en question. En outre, nous devrions considérer quatre types d'index : *index primaire* en séquentiel indexé, *index primaire* en organisation calculée, *index secondaire* sur fichier en vrac et *index secondaire* sur fichier ordonné (*clustering index*). Nous étudierons sommairement, pour les comparer, les performances de la lecture séquentielle, de l'accès par *index secondaire* sur fichier en vrac et de l'accès par *index secondaire* sur fichier ordonné.

Soit une table T de N_r enregistrements, occupant N_p pages contenant chacune en moyenne N_{rpp} enregistrements. On admet que $N_r > N_p$ et que l'éventuelle fragmentation des enregistrements est négligeable. Soit un index associé à cette table et construit sur le champ C possédant N_v valeurs.

- **Accès séquentiel** (section 4.6.1). La requête **Q** est évaluée par le parcours de la totalité de la table. La condition de sélection where C = k est évaluée pour chaque enregistrement. L'accès aux pages successives de T coûte :

  $$t_{q1} = t_{lsf2} = N_p \times t_{ls1}$$

- **Index secondaire sur fichier en vrac** (section 4.10.2). Les enregistrements qui possèdent une valeur déterminée de C sont distribués aléatoirement dans l'espace de la table. Une entrée de l'index donne accès en moyenne à $N_{rpv} = N_r / N_v$ enregistrements de T (dits *enregistrements sélectionnés*) et chaque page de la table contient en moyenne m_v enregistrements sélectionnés :

  $$m_v = N_{rpv} / N_p$$

  Ces enregistrements occupent en moyenne q_v pages de la table[41] :

  $$q_v = N_p \times (1 - e^{-m_v})$$

  Ces pages étant distribuées aléatoirement dans l'espace, le temps d'exécution de la requête[42] **Q** est donc :

  $$t_{q2} = q_v \times t_{la1}$$

- **Index secondaire sur fichier ordonné** (*clustering index*, section 4.12.1). L'index spécifie une adresse d'origine, à partir de laquelle les pages successives sont lues

41. Selon la loi de Poisson, e^{-m_v} exprime la probabilité qu'une page ne contienne pas d'enregistrements sélectionnés.

42. Dans les deux scénarios d'index secondaires, on ignore le temps d'accès à l'index, considéré comme négligeable dans un raisonnement de comparaison. On admet en outre que le SGBD n'accède pas plus d'une fois à une page, quel que soit le nombre d'enregistrements sélectionnés qu'elle contient. Il s'agit d'une optimisation courante. Cette hypothèse est en outre raisonnablement vérifiée si N_{rpv} est assez petit, et en tout cas $<< N_p$.

tant qu'elles contiennent des enregistrements sélectionnés. Chaque page ne contenant que des enregistrements sélectionnés, sauf éventuellement la première et la dernière, on calcule n_{piv} le nombre de pages de la séquence et le temps de leur lecture (variante *sans ruptures, réaliste*) :

$$n_{piv} = \lfloor Np / N_v + 0{,}5 \rfloor + 1$$

$$t_{q3} = n_{piv} \times t_{ls1}$$

Faut-il créer un index sur le champ C ?

Comparons d'abord les deux premiers scénarios. Le temps d'exécution de la requête **Q** est t_{q1} par accès séquentiel et t_{q2} via un index secondaire sur fichier en vrac. Un index sur C est conseillé si $t_{q2} < t_{q1}$ *de manière significative*[43], soit, en considérant un seuil minimal Thr_1 (par exemple 0,85), si :

$$t_{q2} < Thr_1 \times t_{q1}$$

Par substitution, on obtient la règle suivante : un index secondaire sur fichier en vrac est utile si :

$$m_v < Z \text{ ou encore } N_{rpp} / N_v < Z$$

où Z est une constante dépendant de Thr_1 et de la technologie utilisée, définie comme suit :

$$Z = -\ln(1 - Thr_1/\alpha_{as})$$

Pour le modèle de disque du chapitre 4, et pour $Thr_1 = 0{,}85$, $Z = 0{,}012768$. Par exemple, en considérant des pages de 12 enregistrements en moyenne, un index secondaire est jugé utile ($Thr_1 = 0{,}85$) lorsque son dictionnaire contient au moins **934** valeurs. Il est à rejeter dès que les coûts estimés sont les mêmes, soit $N_v = 1\,106$.

La figure A4.44 montre l'utilisation d'une feuille de calcul pour déterminer l'utilité de créer un index selon ce modèle. On utilise deux valeurs Thr_1 et Thr_2 qui définissent trois régions du rapport $r = t_{q2} / t_{q1}$:

- l'index est *recommandé* avec certitude si $r < Thr_1$
- il se peut que l'index soit *intéressant* si $Thr_1 \leq r < Thr_2$
- l'index est *déconseillé* avec certitude si $Thr_2 \leq r$

Le raisonnement relatif au troisième scénario est plus facile. La règle est la suivante : un index secondaire sur fichier ordonné est utile si :

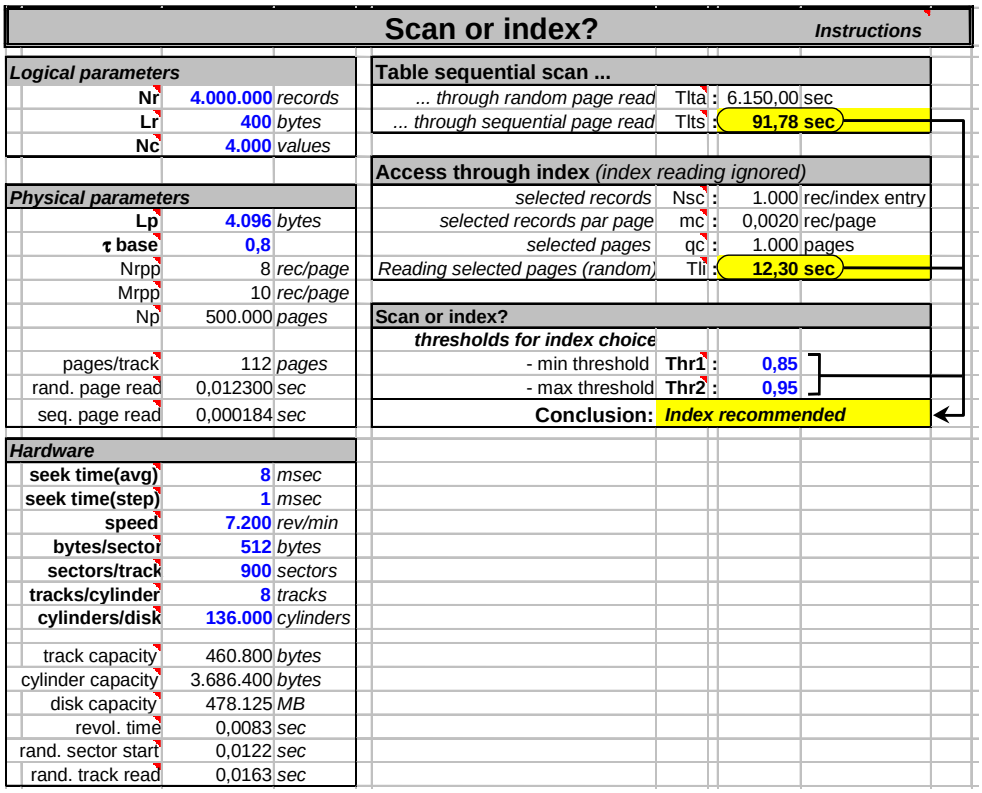
$$t_{q3} < Thr_1 \times t_{q1}$$

soit encore si :

$$N_v > \lceil 1 / Thr_1 \rceil$$

43. On tient compte du fait que la simple présence d'un index entraîne des coûts supplémentaires en espace occupé et en temps de gestion. Vu le degré d'imprécision des évaluations, un calcul rigoureux de ces coûts serait sans intérêt ici.

Pour **Thr1** ≥ 0,85, l'index est choisi dès que $N_V \geq 2$. Cette condition est évidemment toujours garantie, sauf cas pathologique. On en conclut qu'un tel index est toujours intéressant. Attention cependant aux contraintes qu'il implique : organisation triée des enregistrements de la table, dégradation progressive de l'ordre, coût de gestion et coût de réorganisation.



Scan or index? — *Instructions*

Logical parameters		
Nr	**4.000.000**	*records*
Lr	**400**	*bytes*
Nc	**4.000**	*values*

Physical parameters		
Lp	**4.096**	*bytes*
τ base	**0,8**	
Nrpp	8	*rec/page*
Mrpp	10	*rec/page*
Np	500.000	*pages*
pages/track	112	*pages*
rand. page read	0,012300	*sec*
seq. page read	0,000184	*sec*

Hardware		
seek time(avg)	**8**	*msec*
seek time(step)	**1**	*msec*
speed	**7.200**	*rev/min*
bytes/sector	**512**	*bytes*
sectors/track	**900**	*sectors*
tracks/cylinder	**8**	*tracks*
cylinders/disk	**136.000**	*cylinders*
track capacity	460.800	*bytes*
cylinder capacity	3.686.400	*bytes*
disk capacity	478.125	*MB*
revol. time	0,0083	*sec*
rand. sector start	0,0122	*sec*
rand. track read	0,0163	*sec*

Table sequential scan ...			
... through random page read	Tlta :	6.150,00	sec
... through sequential page read	Tlts :	**91,78 sec**	

Access through index *(index reading ignored)*			
selected records	Nsc :	1.000	rec/index entry
selected records par page	mc :	0,0020	rec/page
selected pages	qc :	1.000	pages
Reading selected pages (random)	Tli :	**12,30 sec**	

Scan or index?		
thresholds for index choice		
- min threshold	**Thr1** :	**0,85**
- max threshold	**Thr2** :	**0,95**
Conclusion:	***Index recommended***	

Figure A4.44 - Feuille de calcul évaluant l'opportunité d'utiliser un index

La figure 45 donne une représentation graphique de l'évolution des temps d'exécution de la requête type en fonction de N_V, le nombre de valeurs de la colonne C. La courbe bleue est relative à l'organisation en vrac avec index et la courbe horizontale verte représente le temps constant de l'accès séquentiel. Le fichier comporte 1 000 000 enregistrements rangés dans des pages de 10 enregistrements au taux d'occupation global de 80%, soit 8 enregistrements par page en moyenne. Le temps est calculé pour les valeurs de N_V comprises entre 20 et 1 100.

Les courbes montrent une intersection pour N_V = 531. Ceci signifie que, pour cette valeur, les deux techniques conduisent au même temps d'exécution de 23 secondes. Pour les valeurs inférieures à 531, l'accès séquentiel donne de meilleures performances, et pour les valeurs supérieures, l'accès par index s'avère meilleur.

Si on tient compte d'un seuil **Thr1** = 0,85, le temps d'exécution par index doit descendre à 23 x 0,85 = 19,55 pour égaler le temps d'accès séquentiel. Ceci est obtenu pour N_V = 625. En dessous de cette valeur, l'accès séquentiel est préférable.

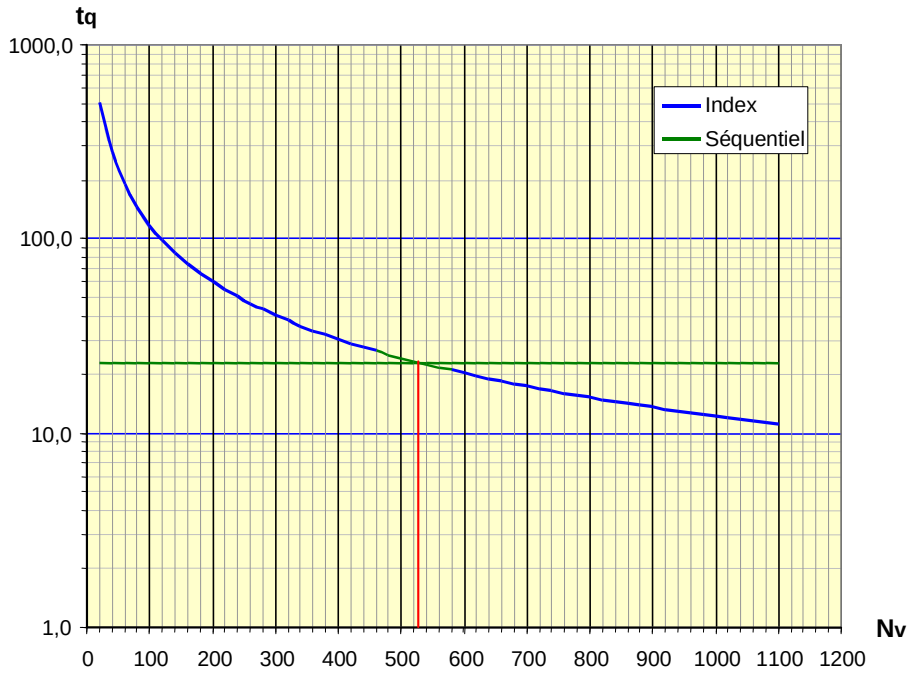


Figure A4.45 - Temp d'exécution en secondes (**tq**) de la requête pour les valeurs de N_V de 20 à 1100

Faut-il exclure les index de faible valeur de Nv ?

On observe qu'un index secondaire utilisé seul peut être moins intéressant pour les faibles valeurs de **Nv** que l'accès séquentiel. Il faut cependant nuancer. Reprenons l'exemple d'un index sur le champ CAT, dont on a montré l'inutilité pour l'évaluation du critère `where CAT = 'B1'`. Il existe cependant deux situations dans lesquelles un tel index peut être très utile.

1. En *conjonction* (**and**) avec d'autres conditions de sélection, cet index permettra de réduire à 10% en moyenne la liste de références produites par les autres index, comme dans la requête ci-dessous.

   ```
   select * from CLIENT
   where  LOCALITE = 'Paris' and CAT = 'B1'
   ```

   Si l'index sur LOCALITE fournit en moyenne 1.000 références à des enregistrements de la table CLIENT, l'utilisation combinée de l'index sur CAT, à supposer que les deux critères sont indépendants, réduira cette liste à 100 références, ce qui constitue un gain considérable.

2. L'autre situation est celle d'une *répartition non uniforme* des enregistrements

dans l'ensemble des valeurs de CAT. Si seules 0,001% des lignes possèdent la valeur `CAT = 'B1'`, alors l'utilisation de l'index s'avérera très utile alors que, pour d'autres valeurs de CAT, l'accès séquentiel sera plus favorable. Ceci suppose que le SGBD dispose de statistiques, non seulement sur le nombre de valeurs de l'index, mais aussi sur le nombre d'enregistrements pour chaque valeur de l'index, ce qui est en général possible, ne fût-ce que par la simple consultation de l'index.

A4.8 EXERCICES DU CHAPITRE 4

A4.46 On considère une table dont les lignes sont rangées dans un espace qui lui est exclusivement réservé et constitué de pages de 4 Ko. Chaque ligne occupe une seule page. Cet espace est implanté sur un disque de notre modèle de référence. La table contient 1 000 000 lignes d'une longueur de 200 octets. Le taux d'occupation moyen des pages est de 75%. Calculer le volume minimal de cet espace de stockage et le temps de lecture séquentielle de toutes les lignes de cette table.

Solution

Capacité d'une page : $\mathbf{Mrpp} = \lfloor \mathbf{Lp} / \mathbf{Lr} \rfloor = 20$ enregistrements par page.

Contenu effectif d'une page : $\mathbf{Nrpp} = \tau \times \mathbf{Mrpp} = 15$ enregistrements par page.

Volume minimal de l'espace : $\mathbf{Np} = \lceil \mathbf{Nr} / \mathbf{Nrpp} \rceil = \mathbf{66\,667}$ pages ou **261** Mo.

Temps de lecture séquentielle de la table (hypothèse de lecture anticipée d'une piste, soit $\mathbf{t_{ls1}} = 0{,}145$ ms par page) : $\mathbf{t_{ls}} = \mathbf{t_{ls1}} \times \mathbf{Np} = \mathbf{9{,}67}$ s.

A4.47 Une table APPEL, représentant des appels téléphoniques, comporte 6 000 000 lignes d'une longueur fixe de 200 octets. Elle dispose d'un identifiant constituée de la colonne ID de 40 caractères, sur laquelle on définit un index. La table est stockée dans un espace qui lui est réservé, implanté sur un disque du modèle de référence, décomposé en pages de 8 Ko. On envisage trois techniques d'implémentation de l'index sur ID : index primaire en séquentiel indexé, index primaire en calculé, index secondaire sur fichier en vrac. Calculer dans chaque cas le volume minimal de l'espace de stockage et le temps d'accès via ID.

Solution

On admet que dans les trois organisations le taux d'occupation moyen du fichier de base est $\tau_b = 0{,}8$. La capacité d'une page est $\mathbf{Mrpp} = \lfloor \mathbf{Lp} / \mathbf{Lr} \rfloor = 40$ enregistrements. Son contenu effectif moyen est $\mathbf{Nrpp} = \tau_b \times \mathbf{Mrpp} = 32$ enregistrements. La taille du fichier de base est donc $\mathbf{Npb} = \lceil \mathbf{Nr} / \mathbf{Nrpp} \rceil =$ **187 500** pages.

Index primaire en séquentiel indexé (SI).

Compte tenu d'un taux de remplissage des pages d'index de $\tau_i = 0{,}8$ et de pointeurs de page de 4 octets, on calcule **Li** = 44 octets, **Mipp** = $\lfloor$**Lp** / **Li**$\rfloor$ = 186 entrées par page et **Nipp** = τ_i x **Mipp** = 148,8 entrées par page. On en déduit **Npi** = **Npi(3)** + **Npi(2)** + **Npi(1)** = 1 261 + 9 + 1 = 1 271 pages et **n** = 3 (aussi calculable par **n** = $\lceil \log_{\mathbf{Nipp}} \mathbf{Npb} \rceil$ = $\lceil$ ln **Npb** / ln **Nipp** $\rceil$ = $\lceil 2{,}427 \rceil$ = 3). Taille du fichier **Np** = **Npb** + **Npi**. = 187 500 + 1 271 = **188 771** pages. Le calcul simplifié **Npi** ≅ **Npi(n)** conduit à une erreur totalement négligeable de 10 pages, soit 0,0053%.

Le temps d'accès via l'identifiant est celui de 4 lectures de pages, soit 49,2 ms. Compte tenu d'un verrouillage des deux premiers niveaux d'index (10 pages) dans le tampon, ce temps tombe à 2 lectures de pages, soit 24,6 ms. Si le tampon peut en outre accueillir 1 261 pages supplémentaires, le temps d'accès n'est plus que de **12,3** ms.

Index primaire en calculé.

L'espace occupé par le fichier calculé se limite à celui du fichier de base, soit **187 500** pages. L'abaque nous donne, pour la courbe **Mrpp** = 40 et un taux d'occupation de 80% un coût d'accès de 1,01 à 1,02 accès physique, soit 1,015 x 12,3 = **12,5** ms. L'accès ne nécessite pas plus d'une page dans le tampon.

Index secondaire sur fichier en vrac.

Le dictionnaire de valeurs de l'index contient **Nv** = 6 000 000 entrées de **Lv** = 46 octets (un pointeur d'enregistrement de 6 octets par entrée). On a donc **Mvpp** = $\lfloor$**Lp** / **Lv**$\rfloor$ = 178 entrées par page et **Nvpp** = τ_v x **Mvpp** = 142,4 entrées par page ($\tau_v = 0{,}8$). On en déduit la taille du dictionnaire : **Npv** = $\lceil$**Nv** / **Nvpp**$\rceil$ = 42 135 pages. On traite ensuite ce dictionnaire comme un fichier de base organisé en séquentiel indexé. Son index primaire se calcule de manière classique. On reprend les grandeurs dont nous disposons déjà : **Li** = 44; **Mipp** = 186; **Nipp** = 148,8. On a aussi **Npi** = **Npi(3)** + **Npi(2)** + **Npi(1)** = 284 + 2 + 1 = 287 pages et **n** = 3. Le volume de l'index secondaire est **Nps** = **Npv** + **Npi** = 42 442 pages. La taille totale du fichier est de **Np** = **Npb** + **Nps**. = 187 500 + 42 442 = **229 942** pages.

Le temps d'accès via l'identifiant est de 5 lectures de pages (3 pour l'index du dictionnaire + 1 pour le dictionnaire + 1 pour le fichier de base), soit 61,5 ms. Si on verrouille des deux premiers niveaux d'index du dictionnaire (3 pages) dans le tampon, ce temps est de 3 lectures de pages, soit 36,9 ms. Si le tampon contient la totalité de l'index du dictionnaire (287 pages), le temps d'accès est de 2 lectures de pages, soit de **24,6** ms. Il est difficile de diminuer ce temps.

En résumé

	volume index	**volume tampon**	**temps d'accès**
Primaire SI	10 Mo	0,08 ou 10 Mo	24,6 ou 12,3 ms
Primaire calculé	0	0,008 Mo	12,5 ms
Secondaire	332 Mo	0,024 ou 2,25 Mo	36,9 ou 24,6 ms

A4.48 Soit un fichier calculé dont **Lr** = 400 octets, **Lp** = 4 Ko et rempli à 90%. Combien de lectures par clé peut-on réaliser par seconde ?

Solution

La capacité des pages est de **Mrpp** = ⌊**Lp** / **Lr**⌋ = 5 enregistrements. L'abaque nous donne, à l'abscisse 90, le nombre d'accès **nk** = 1,29, soit un temps d'accès de **nk** x **tla1** = 15,9 ms. La vitesse de lecture est donc de 1/0,0159 = **62** enregistrements par seconde.

A4.49 Quel est le taux d'occupation à partir duquel il convient de réorganiser un fichier calculé dont **Mrpp** = 20 si le temps moyen d'accès par clé ne peut dépasser 15 ms ?

Solution

Un temps de 15 ms correspond à 0,015 / **tla1** = 1,22 accès physique aléatoires. Pour **Mrpp** = 20 et **nk** = 1,22 l'abaque nous donne un taux d'occupation de **96**%.

A4.50 Quelle taille de page minimum préconiser pour un fichier calculé dont le temps d'accès ne peut dépasser 1,15 accès physiques au taux de remplissage de 90% ?

Solution

Pour ces données, l'abaque nous indique une valeur **Mrpp** comprise entre 10 et 20 mais plus proche de 20. La fonction **nk** = f(**Mrpp**) présentant une allure logarithmique inverse, on estime la taille recherchée à **15** enregistrements par page.

A4.51 On considère un fichier séquentiel **FS** de 5 000 000 enregistrements de 200 octets. Ces enregistrements sont en désordre sur les valeurs de l'identifiant. On se propose de les ranger dans un fichier **FC** à organisation calculée sur l'identifiant avec gestion des débordements en zone indépendante. L'administrateur de la base de données hésite entre deux procédures :

1. lecture séquentielle de **FS** et insertion dans **FC** des enregistrements successifs selon la procédure standard,
2. constitution d'un 2e fichier séquentiel **FS'** constitué des enregistrements de **FS** auxquels on ajoute l'adresse de base **Ab** calculée par la fonction **f**; on trie **FS'** selon le champ **Ab**, ce qui produit le fichier séquentiel **FS"**; enfin, on poursuit selon la procédure 1) à partir du fichier **FS"**.

Quelle serait la procédure la plus rapide ?

Solution

Essayons d'abord de nous convaincre que la procédure 2, en apparence plus complexe, est néanmoins tout à fait pertinente. Les enregistrements de **FS"** se présentent par ordre croissant de leurs adresses de base. Lors de l'insertion

d'un enregistrement **r** dans **FC** par le SGBD, son adresse de base **p** est calculée[44]. Si cette adresse est identique à celle de l'enregistrement précédent, ce qui sera le plus souvent le cas, la page de base se trouve déjà dans le tampon. L'insertion de **r** dans cette page ne demandera donc pas d'accès au disque. Concrètement, le chargement du fichier calculé se comportera comme l'écriture des enregistrements successifs d'un fichier séquentiel à l'exception de la gestion des débordements, mais qui seront relativement rares[45].

Nous pouvons à présent calculer le temps d'exécution des deux procédures. On choisit les paramètres suivants pour le fichier calculé en création : L_p = 4 Ko (donc M_{rpp} = 20) et τ_{ch} = 0,9. On en déduit N_{rpp} = 0,9 x 20 = 18, le nombre de pages de base N_{pb} = 5 000 000 / 18 = 277 778, le taux d'augmentation des pages τ_{ap} = 1,27 (donné par le graphique de la figure A4.4.26 pour M_{rpp} = 20 et τ_{ch} = 0,9) et la taille totale du fichier après chargement N_p = 1,27 x 277 778 = 352 778 pages.

Procédure standard

Le fichier comporte N_r = 5 000 000 enregistrements dont chacun nécessite (1) la lecture de sa page de base, (2) la lecture des pages de débordement éventuelles, (3) la réécriture de la page de l'enregistrement et (4) lorsqu'une nouvelle page de débordement doit être créée, sa réservation et son lien avec la page précédente. Le graphique de la figure A4.4.27 nous donne, pour M_{rpp} = 20 et τ_{ch} = 0,9, la valeur n_{lp} = 1,05, qui indique que moins de 5% des enregistrements sont en débordement. Nous pouvons sans risque ignorer l'effet des débordements au moment du chargement en considérant que chaque enregistrement nécessite une lecture et une écriture de page. Le temps de chargement est donc :

$t_{ch1} = N_r \times 2 \times t_{la1} = 5\,000\,000 \times 2 \times 0{,}0123 = 123\,000$ s (> 34 heures)

Procédure à tri préalable

Avec les paramètres N_r = 5 000 000, L_r = 200 , L_p = 4 096 , v = 6 voies, un tampon de tri initial de 1 000 pages, la feuille de calcul sort.xls nous donne une estimation du temps de tri de 336 secondes.

Le chargement proprement dit consiste à lire séquentiellement le fichier **FS"** puis à écrire les enregistrements dans le fichier **FC**. Cette dernière opération correspond physiquement au garnissage des pages de base dans l'ordre de leurs adresses dans le fichier, c'est-à-dire l'écriture séquentielle du fichier (toujours en ignorant l'effet des débordements au moment du chargement, jugé négligeable). En résumé, le chargement comprend la lecture séquentielle

44. Même si elle est fournie par le champ **Ab**. Il serait imprudent (et d'ailleurs illégal) de court-circuiter la procédure interne de gestion des index du SGBD.

45. Petite inefficacité facile à résoudre par une lecture anticipée et une écriture retardée d'une piste (section A4.4.3.3) puisque tant l'espace de base que l'espace de débordement se comportent comme des (sous-)fichiers séquentiels. Ces paramètres de gestion des accès au disque sont généralement disponibles au niveau de l'espace de stockage.

de **FS"** et l'écriture séquentielle de **FC** (attention, les pages de base préexistant, l'écriture exige la lecture d'une page puis sa réécriture). On suppose une lecture anticipée d'une piste et la recopie des pages dès qu'elles atteignent le contenu d'une piste dans le tampon.

La taille de **FS"** est de 250 000 pages et celle de **FC** est de 352 778 pages. La lecture de **FS"** coûte 250 000 x 0,000184 = 46 secondes et l'écriture dans **FS"** coûte 2 x 352 778 x 0,000184 = 130 secondes. Au total, le chargement coûte **tch2** = 176 secondes soit près de 3 minutes.

La conclusion s'impose d'elle-même !

A4.52Lors de la sortie de l'UNIVAC I, en 1951, on annonçait les performances suivantes : mémoire centrale de 1 000 mots de 72 bits ou 12 caractères, vitesse des lecteurs de bande de 500 mots par seconde. On estimait que cette machine serait capable de trier 100 millions d'enregistrements de 10 mots en 9 000 heures[46]. Quel serait le temps d'un tel tri aujourd'hui, en supposant l'utilisation de 2 voies ?

Solution

Avec les paramètres suivants :

- **Nr** = 100 000 000
- **Lr** = 120 octets
- **Lp** = 4 096
- **v** = 6
- taille de l'espace de tri initial de 2 000 pages

la feuille de calcul donne un temps de **1 heure 30 minutes**.

A4.53On choisi d'implémenter les tables COMMANDE et DETAIL sous la forme d'un *clusters* (à la manière d'Oracle) articulé autour des colones NCOM. Les données sont stockées sur notre disque de référence organisé en pages de 4 Ko. On admet que la table COMMANDE contient 5 000 000 lignes et la table DETAIL 100 000 000 lignes. Les lignes de COMMANDE ont une longueur fixe de 400 octets et les lignes de DETAIL 250 octets. Les *clusters* sont organisés en fichier calculé. Calculer

Le volume occupé par les données de ces deux tables.

Le temps de lecture d'une ligne de COMMANDE de valeur de NCOM donnée.

Le temps de lecture d'une ligne de DETAIL de valeur de NCOM et NPRO données.

Le temps de lecture d'une ligne de COMMANDE et de toutes ses lignes de DETAIL

Le temps de lecture séquentielle des lignes de COMMANDE.

Le temps de lecture séquentielle des lignes de COMMANDE **triées**

46. Source [Knuth, 1998]. Une telle opération aurait nécessité de nombreuses manipulations, les bandes magnétiques de cette époque ayant une capacité de quelques méga-octets seulement.

A4.54L’image ci-dessous est celle de l’étiquette d’un composant d’un ordinateur de bureau. Quelles informations peut-on en tirer ? Il n’est pas interdit de faire appel à des sources additionnelles.