Les Fichiers structurés

I Généralités

Motivation

- Dans les applications du type “bases de données”, les fichiers sont utilisés pour conserver des données structurées en *enregistrements*.

- La manipulation des données se fait alors enregistrement par enregistrement.

- Il est essentiel de pouvoir très rapidement avoir accès à un enregistrement particulier parmi un très grand nombre. Ceci implique que simplement conserver les enregistrements dans un fichier séquentiel classique ne suffit pas.

- Il est donc nécessaire d’adopter des organisations spéciales pour les fichiers utilisés dans ce type d’applications.

Conserver de grandes quantités de données : Le support technologique

- Pour conserver de très grandes quantités de données, on utilise des ensembles de disques qui sont utilisés comme un seul disque virtuel.

- C'est ce qu'on appelle un RAID *Redundant Array of Inexpensive Disks*.

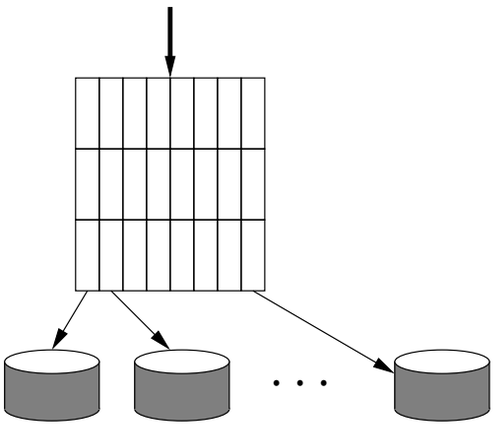
- Outre permettre une grande capacité, la technologie RAID est très flexible et permet des améliorations de performance et de fiabilité.

La technologie RAID : les niveaux

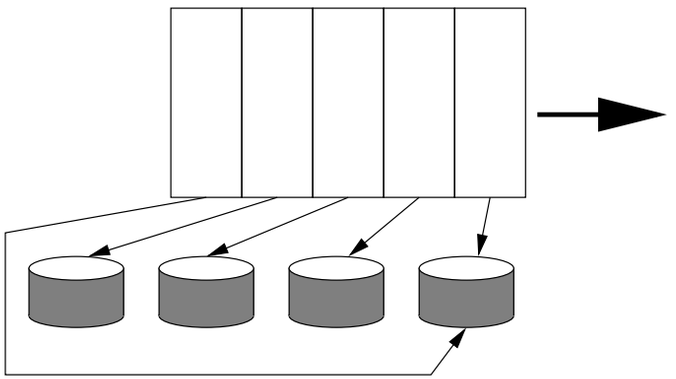
Plusieurs politiques de gestion des RAID ont été définies.
Traditionnellement on parle de *niveaux* RAID.

- Une technique de base qui permet une augmentation des performances est le *striping* (écriture par bandes). Cette technique consiste à distribuer les données entre les disques du RAID en vue de paralléliser les accès. Elle correspond au RAID *niveau 0*. On distingue deux types de “striping” .

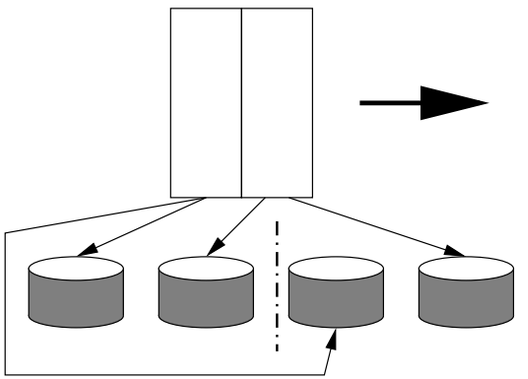
- Le “striping” au niveau du bit (*bit level*). Il correspond à créer des groupes de 8 disques et à répartir les bits de chaque octet entre ces 8 disques.



- Le “striping” au niveau du bloc (*bloc level*). Il correspond à utiliser n disques et à écrire le bloc i sur le disque $(i \bmod n) + 1$.



- Pour augmenter la fiabilité, une technique efficace est de partitionner l'ensemble des disques en 2 groupes et de maintenir sur chacun des deux groupes une copie complète des données. Chaque opération d'écriture est donc réalisée de façon simultanée et symétrique sur chacun des 2 groupes. C'est ce qu'on appelle le *mirroring* (écriture miroir) qui correspond au RAID *niveau 1*.



- Les autres niveaux RAID (2-6) prévoient une redondance par l'utilisation de codes de correction d'erreur.

- L'écriture par bandes (*striping*) peut être combinée avec l'écriture miroir (*mirroring*) ou avec les codes de correction d'erreur.

- Avec une redondance suffisante, on arrive à une excellente fiabilité et on peut même remplacer les disques défaillants sans arrêt du système. C'est ce qu'on appelle un *échange à chaud* (*hot swapping*).

Les performances des systèmes de fichiers

Quelles sont des performances acceptables lorsque l'on manipule de grosses quantités de données sur disque ?

- Ce qui compte de façon prépondérante est le nombre d'accès à des blocs du disque ; les opérations en mémoire nécessitent un temps comparativement très faible.

- Lorsque de nombreux blocs sont lus, il est essentiel qu'ils puissent être lus en séquence, sans déplacement des têtes ou temps de latence entre lectures successives.

- Pour caractériser les différentes méthodes d'organisation des fichiers, nous utiliseront la notation O représentant la complexité asymptotique.

- Une méthode d'accès aux données a une complexité $O(f(n))$ si, lorsqu'elle traite des données de taille n, le nombre $b(n)$ d'accès à des blocs nécessaire est tel que

$$\exists n_0 \exists k \; \forall n \geq n_0 \; (b(n) \leq kf(n))$$

Quelle complexités sont acceptables ?

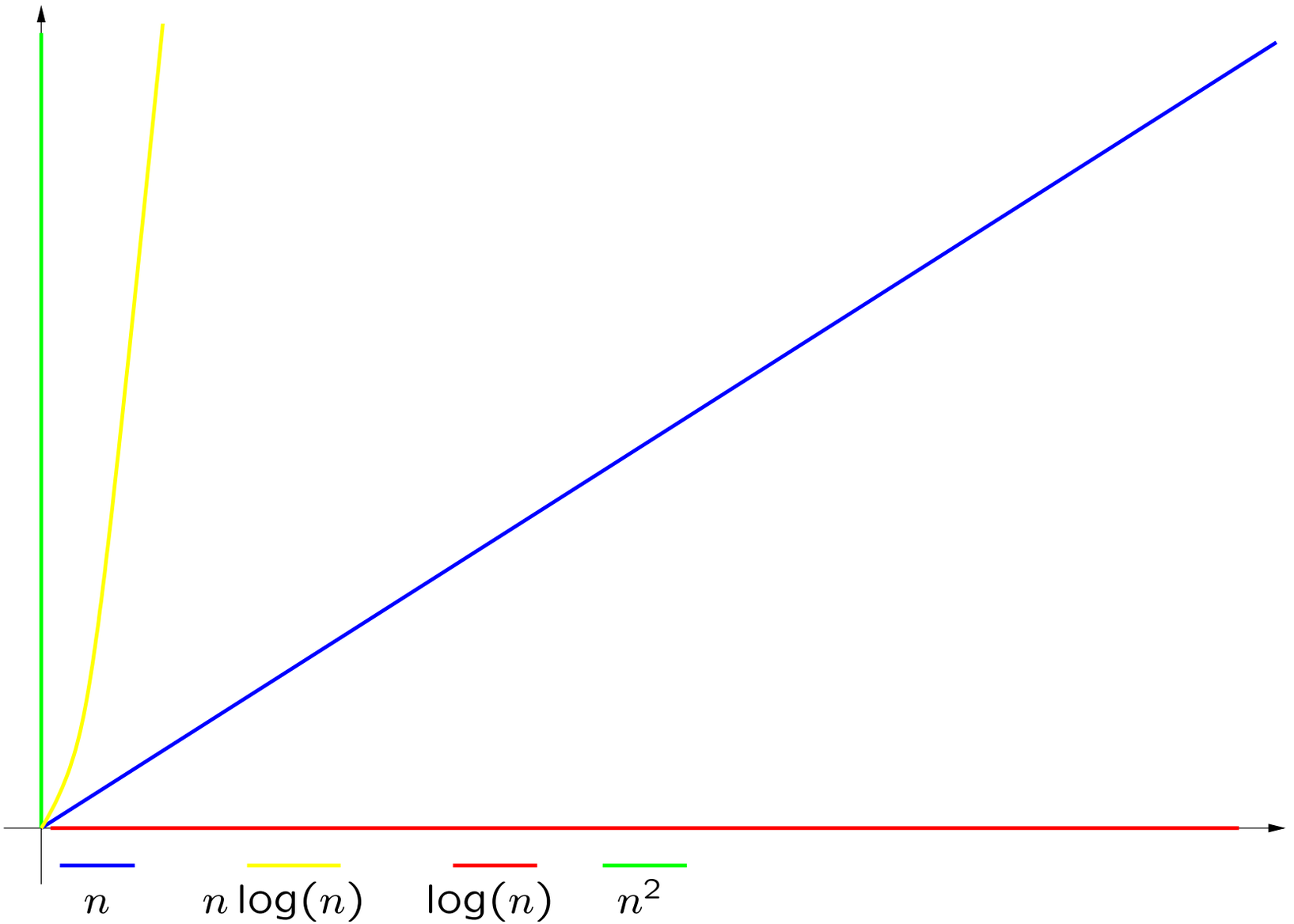
Supposons que nous avons à traiter une très grosse quantité de données, par exemple 10 Gbytes, soit 10^7 blocs de 1024 bytes. Supposons que le temps d'accès aux blocs soit de 10 ms, mais que le taux de transfert des données peut atteindre 10 Mbytes par seconde lors d'une lecture séquentielle.

- Une complexité linéaire ($O(n)$) implique un temps de traitement pouvant aller de 1000 secondes (16 minutes) dans le cas d'un transfert optimum à 100.000 secondes (1 jour).

- Une complexité $O(n \log(n))$ multiplie le temps de traitement correspondant à la complexité linéaire par un facteur de l'ordre de 10-20.

- Une complexité quadratique ($O(n^2)$) implique un temps de traitement largement prohibitif.

- Une complexité logarithmique ($O(\log(n))$ résulte en un temps de traitement qui sera presque toujours inférieur à une seconde.


n
$n \log(n)$
$\log(n)$
n^2

Conclusions pratiques

- Une complexité $O(\log(n))$ permet un traitement quasi instantané.
- Une complexité $O(n)$ (éventuellement aussi $O(n \log(n))$) est acceptable pour un traitement occasionnel, par exemple quotidien ou hebdomadaire, des données.
- Toute méthode de traitement de complexité supérieure est inexploitable pour de grandes quantités de données.

La notion d'enregistrement

Un enregistrement est un groupe de données de type fixé qui constitue l'unité de base pour la manipulation et la structuration d'un fichier.

- Dans un fichier générique (séquence d'octets) on peut considérer qu'un enregistrement est un octet.

- Les enregistrements sont des groupes de données de types éventuellement différents mais fixés. Les enregistrements utilisés dans les fichiers sont similaires à ceux employés dans les langages de programmation.

- On considérera non seulement des enregistrements de taille fixe, mais aussi des enregistrements de taille variable.

Enregistrement : Exemple

La définition ci-dessous décrit un type d'enregistrement de taille fixe.

```
type person_record = record
                       nom:  packed array[1..20] of char;
                       rue:  packed array[1..20] of char;
                       numero:  integer;
                       ville:  packed array[1..20] of char;
                       codepostal:  integer;
                       tel:  packed array[1..11] of char
                       end
```

La notion de clé

Une clé d'un type d'enregistrement donné est un sous-ensemble de ses champs qui permet d'identifier de façon unique un enregistrement de ce type.

- *Exemple* : Pour les enregistrement de type `person_record` une clé pourrait être le champ `nom`.

- Le fait qu'une collection de champs puisse ou non servir de clé dépend des enregistrements que l'on va effectivement introduire dans le fichier.

- Il est de pratique courante de prévoir un champ (numéro matricule, numéro de référence) explicitement destiné à servir de clé. Le fait qu'un tel champ est une clé n'est garanti que par l'usage que l'on en fait (attribution obligatoire de numéros de référence uniques).

La représentation des enregistrements

Nous considérons d'abord uniquement les enregistrements de taille fixe.

- Un enregistrement de taille fixe peut toujours se représenter par une séquence d'octets de taille fixée.

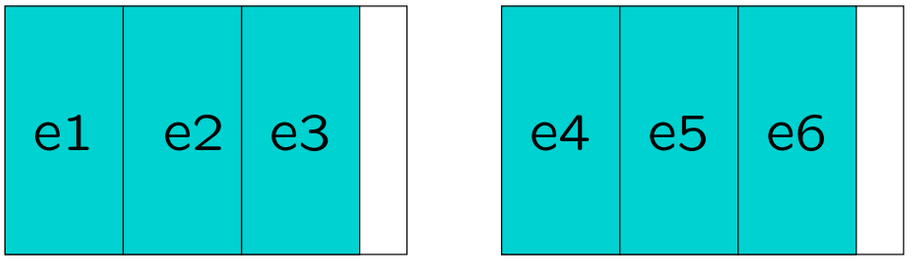
- Le problème qui se pose est de savoir comment organiser les enregistrements du fichier dans les blocs du disque qui servent à mémoriser le contenu du fichier.

- En général un enregistrement aura une taille qui est différente de celle des blocs et peut lui être inférieure ou supérieure.

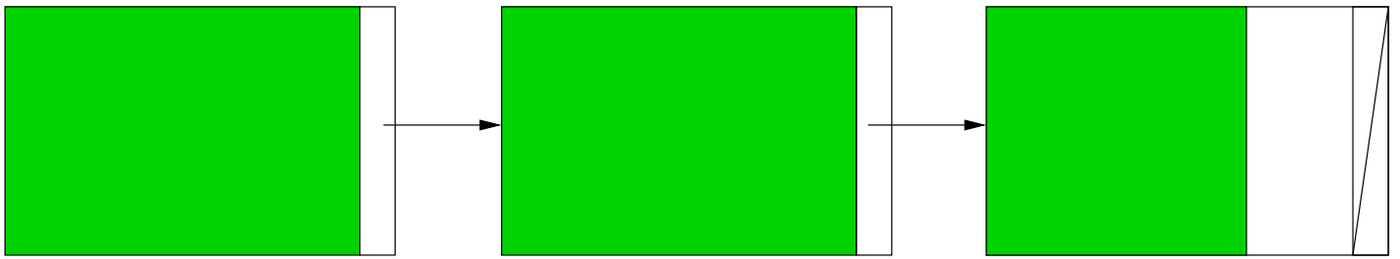
- Dans le cas d'enregistrement de taille inférieure à un bloc on placera dans chaque bloc le plus grand nombre d'enregistrements possible tout en évitant de séparer un enregistrement entre plusieurs blocs.

- Dans le cas d'enregistrements de taille supérieure à un bloc, on utilisera pour chaque enregistrement le nombre de blocs nécessaire chaînés à l'aide de pointeurs.

Enregistrements de taille inférieure à un bloc



Enregistrements de taille supérieure à un bloc



La représentation des enregistrements : enregistrements de taille variable

La taille variable des enregistrements provient de la répétition possible d'un champ.

Exemple :

```
type person_record = record
                       nom:  packed array[1..20] of char;
                       rue:  packed array[1..20] of char;
                       numero:  integer;
                       ville:  packed array[1..20] of char;
                       codepostal:  integer;
                       tel:  array[1..∞] of packed array[1..11] of char
                       end
```

Permet de mémoriser pour chaque personne un nombre quelconque de numéros de téléphone.

Il y a trois méthodes pour représenter des enregistrements de longueur variable.

- **La méthode de l'espace réservé**. On fixe un nombre maximum, de répétitions du champ variable et on réserve l'espace correspondant. On est donc ramené au cas des enregistrements de longueur fixe.

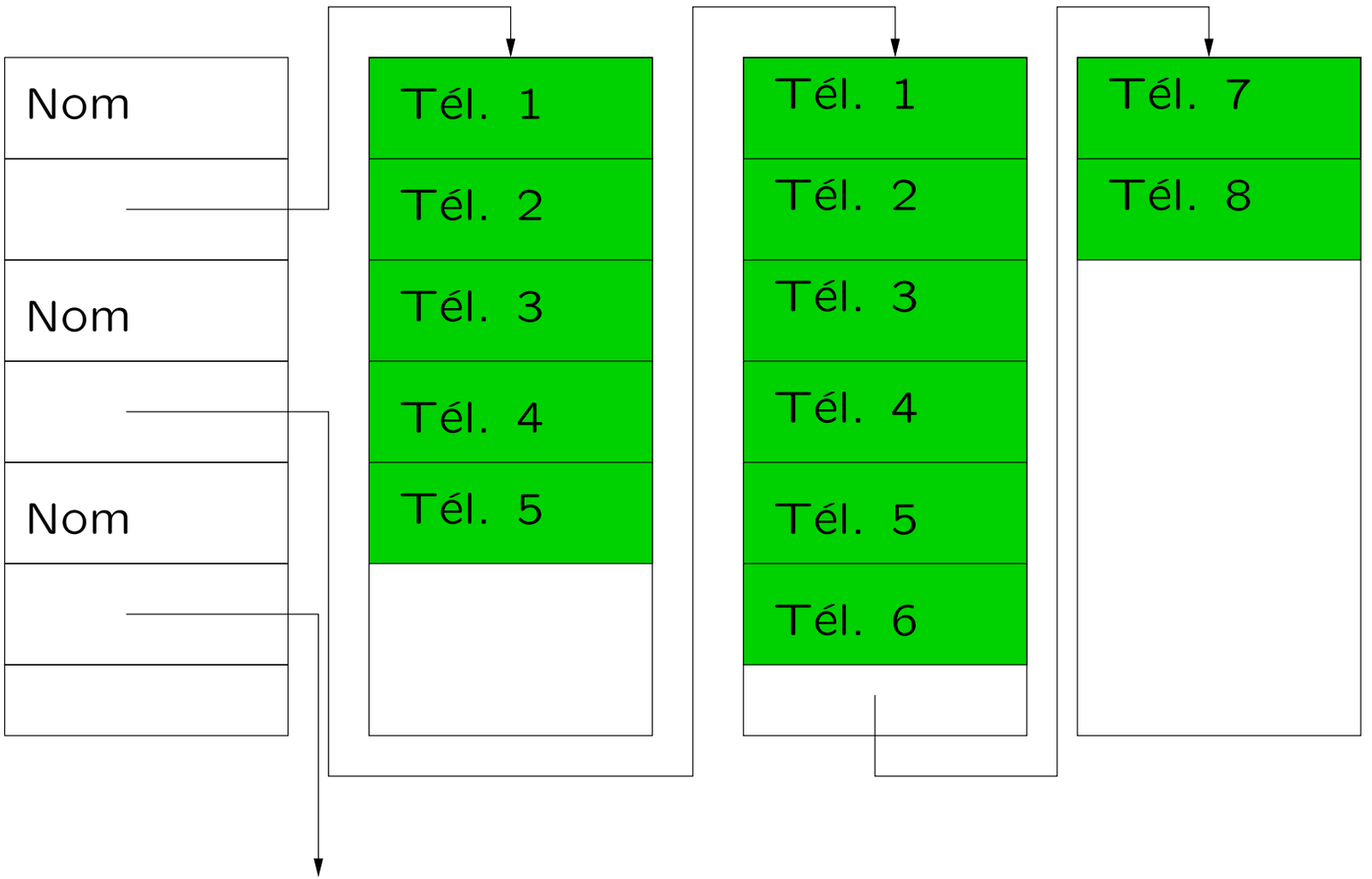
- **La méthode des pointeurs**. On mémorise la partie variable de l'enregistrement dans une liste chaînée de blocs indiquée par un pointeur.

- **La méthode combinée**. On réserve l'espace nécessaire au nombre le plus courant de répétitions ; les répétitions additionnelles sont gérées par la méthode des pointeurs.

Méthode de l'espace réservé

Nom	Nom	Nom	Nom	
Tél.1	Tél. 1	Tél. 1	Tél. 1	
Tél. 2	Tél. 2	Tél. 2	Tél. 2	

Méthode des pointeurs



Méthode combinée