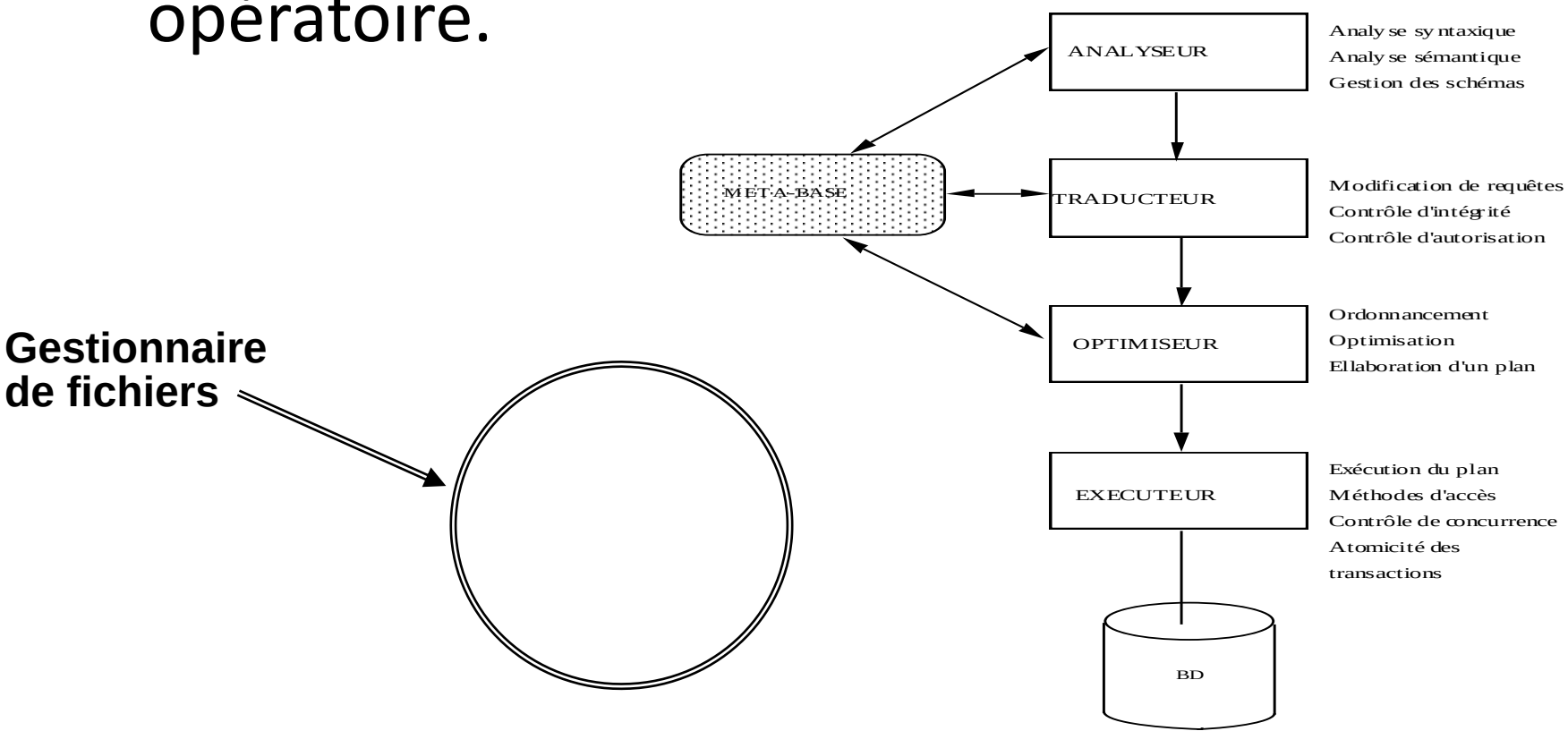
Hachage et Indexation

- 1. Concepts de base
- 2. Organisations par hachage
- 3. Organisations indexées

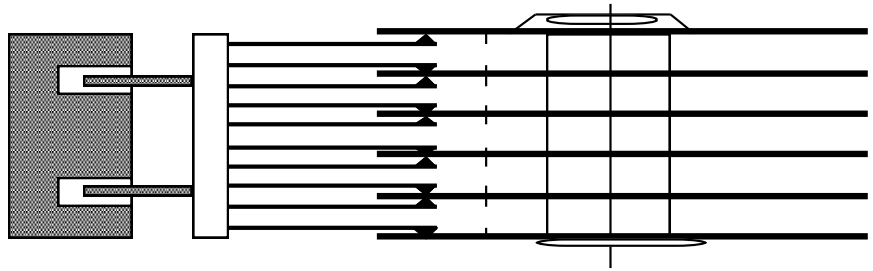
1. Concepts de Base

- Le gestionnaire de fichiers est la couche interne d'un SGBD, souvent intégrée au système opératoire.

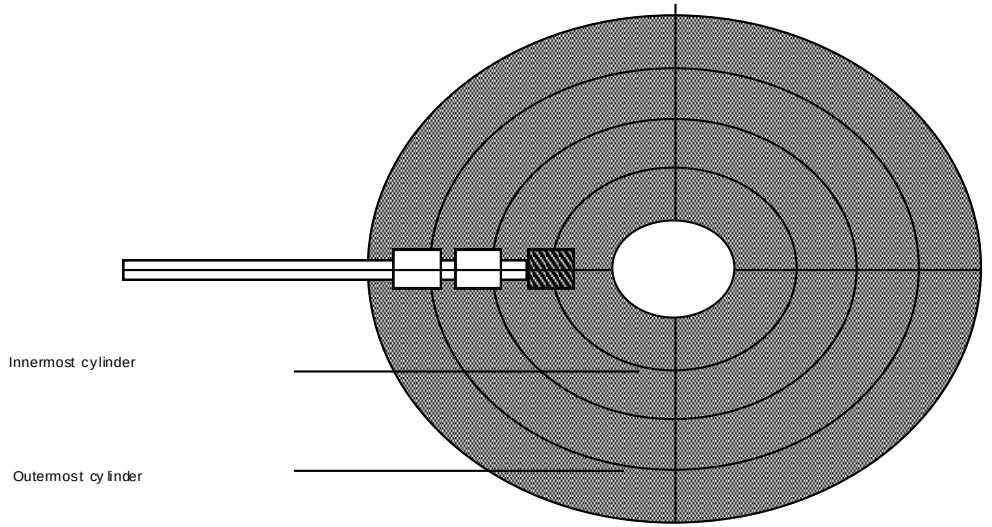


Structures des Disques

- Notion 1: Volume (Disk Pack)
  - Unité de mémoire secondaire amovible.



(a) Side view



(b) Top view

Notion de fichier

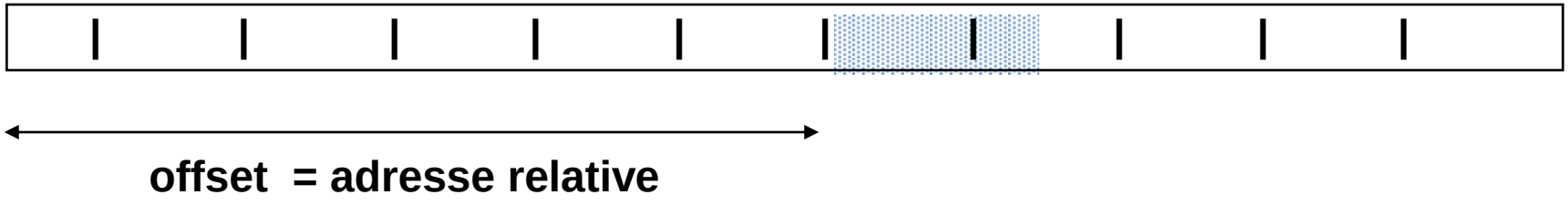
- Notion 2: Fichier (File)
  - Récipient d'information caractérisé par un nom, constituant une mémoire secondaire idéale, permettant d'écrire des programmes d'application indépendants des mémoires secondaires.
- Un fichier se caractérise plus particulièrement par :
  - UN NOM
  - UN CREATEUR
  - UNE DATE DE CREATION
  - UN OU PLUSIEURS TYPES D'ARTICLE
  - UN EMPLACEMENT EN MS
  - UNE ORGANISATION

Quelques notions de base

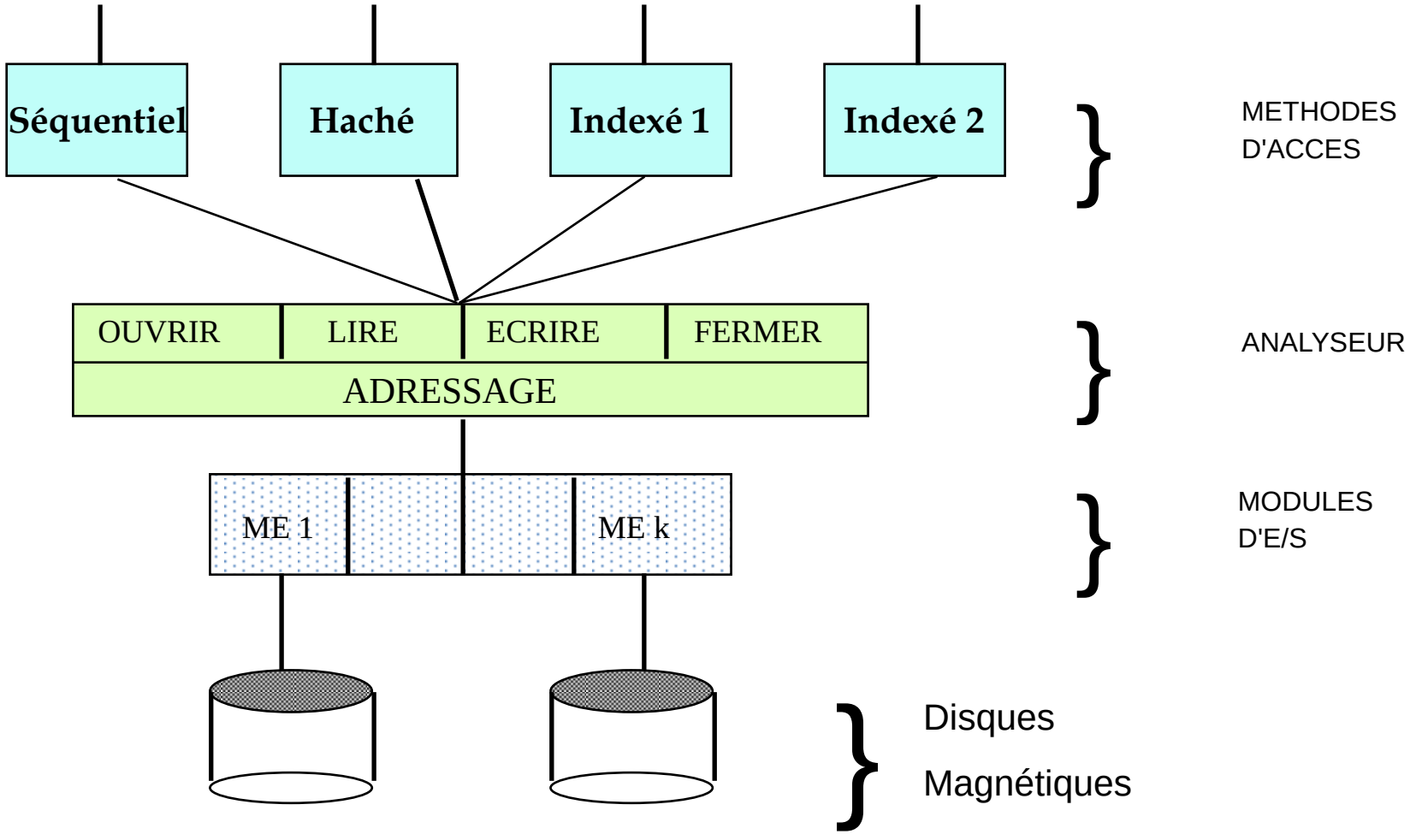
- Notion 3: Article (Record)
  - Elément composant d'un fichier correspondant à l'unité de traitement par les programmes d'application.
- Notion 4: Organisation de fichier (File organization)
  - Nature des liaisons entre les articles contenus dans un fichier.
- Notion 5: Méthode d'accès (Acces Method)
  - Méthode d'exploitation du fichier utilisée par les programmes d'application pour sélectionner des articles.
- Notion 6: Clé d'article (Record Key)
  - Identifiant d'un article permettant de sélectionner un article unique dans un fichier.

Adressage Relatif

- Notion 7: Adresse relative (Relative address)
  - Numéro d'unité d'adressage dans un fichier (autrement dit: déplacement par rapport au début du fichier).



Architecture d'un SGF



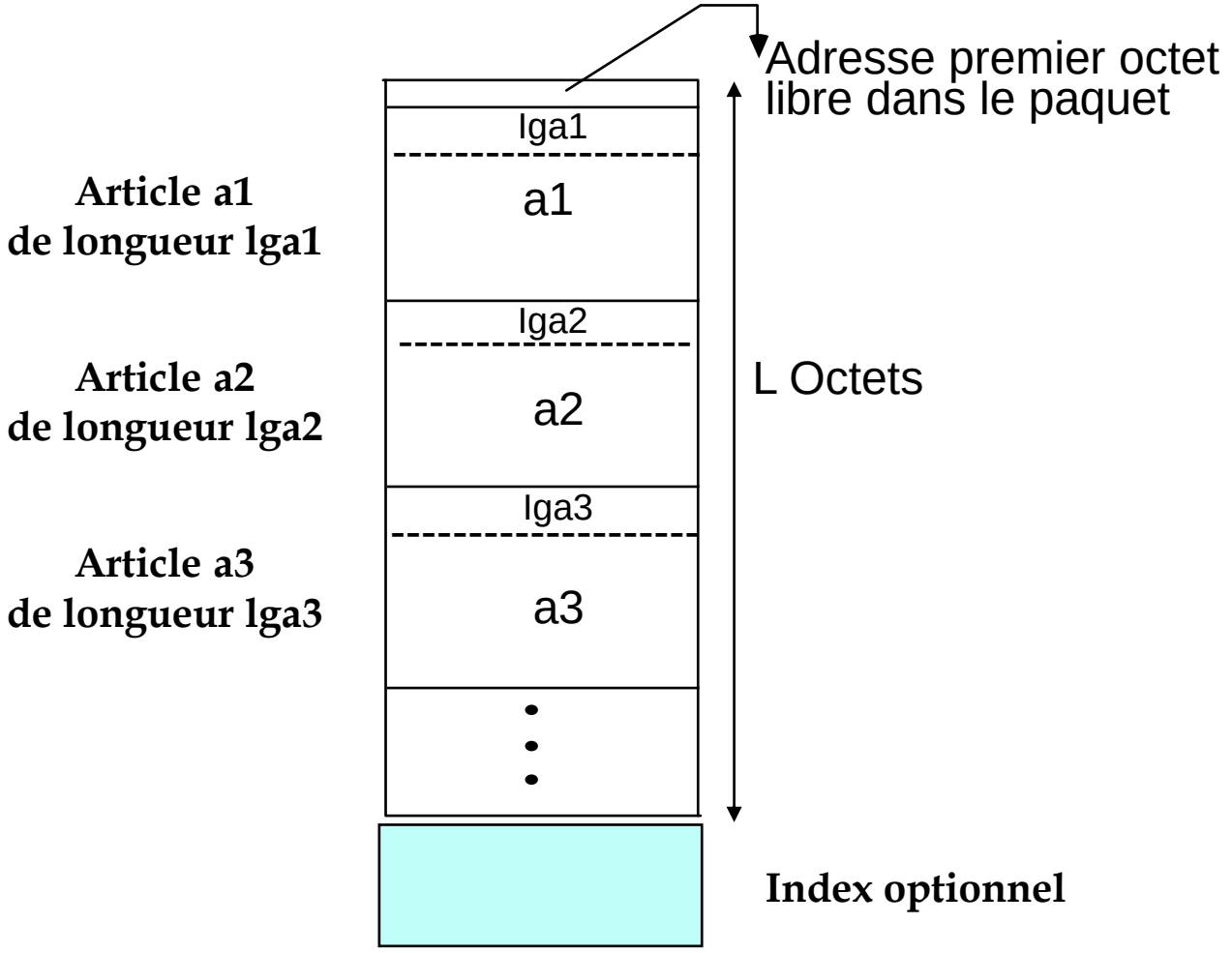
Commandes de base

- mount(), unmout()
  - monte et démonte un système
- mkdir(), chdir(), rmdir()
  - créer, changer de, détruire un répertoire
- open(nomf, file), close(nomf, file)
  - ouvrir et fermer un fichier
- lseek(file, offset)
  - se positionner dans un fichier
- read(file, buf, count, offset)
  - lecture d'octets sur un fichier
- write(file, buf, count, offset)
  - écriture d'octets dans un fichier

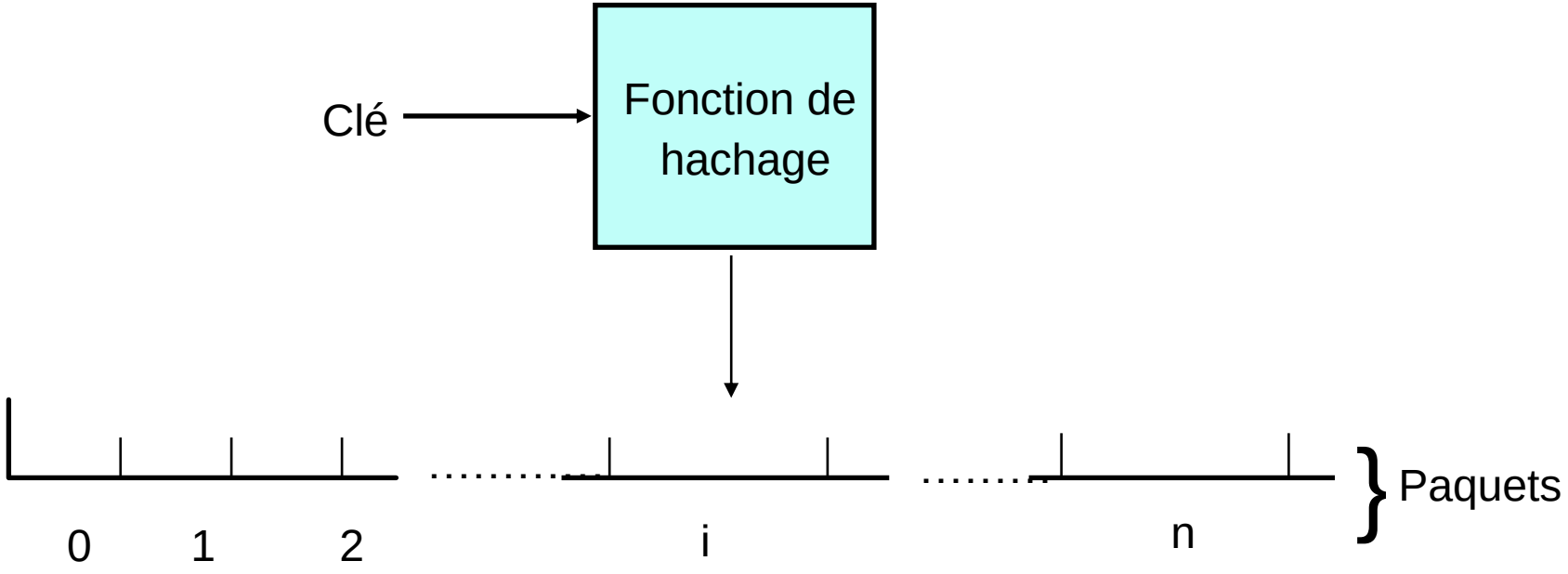
2. Organisations par Hachage

- Notion 8: Fichier haché statique (Static hashed file)
  - Fichier de taille fixe dans lequel les articles sont placés dans des paquets dont l'adresse est calculée à l'aide d'une fonction de hachage fixe appliquée à la clé.

Structure interne d'un paquet



Vue d'un fichier haché statique



Fonction de Hachage

- DIFFÉRENTS TYPES DE FONCTIONS :
  - PLIAGE DE LA CLE
  - CONVERSION
  - MODULO P
  - FONCTION PSEUDO-ALEATOIRE MIXTE
- BUT :
  - Obtenir une distribution uniforme pour éviter de saturer un paquet
  - Mauvaise fonction de hachage ==> Saturation locale et perte de place
- SOLUTION : AUTORISER LES DEBORDEMENTS

Techniques de débordement

- **l'adressage ouvert**
  - place l'article qui devrait aller dans un paquet plein dans le premier paquet suivant ayant de la place libre; il faut alors mémoriser tous les paquets dans lequel un paquet plein a débordé.
- **le chaînage**
  - constitue un paquet logique par chaînage d'un paquet de débordement à un paquet plein.
- **le re-hachage**
  - applique une deuxième fonction de hachage lorsqu'un paquet est plein pour placer en débordement.

Problème du hachage statique

- **Nécessité de réorganisation**
  - Un fichier ayant débordé ne garantie plus de bons temps d'accès (2 accès disque en écriture, 1 en lecture)
  - Le nombre de paquets primaires est fixe, ce qui peuT entrainer un mauvais taux de remplissage

- **Solution idéale: réorganisation progressive**
  - Un fichier ayant débordé devrait rester analogue à un fichier n'ayant pas débordé.
  - Il serait souhaitable de changer la fonction d'adressage.

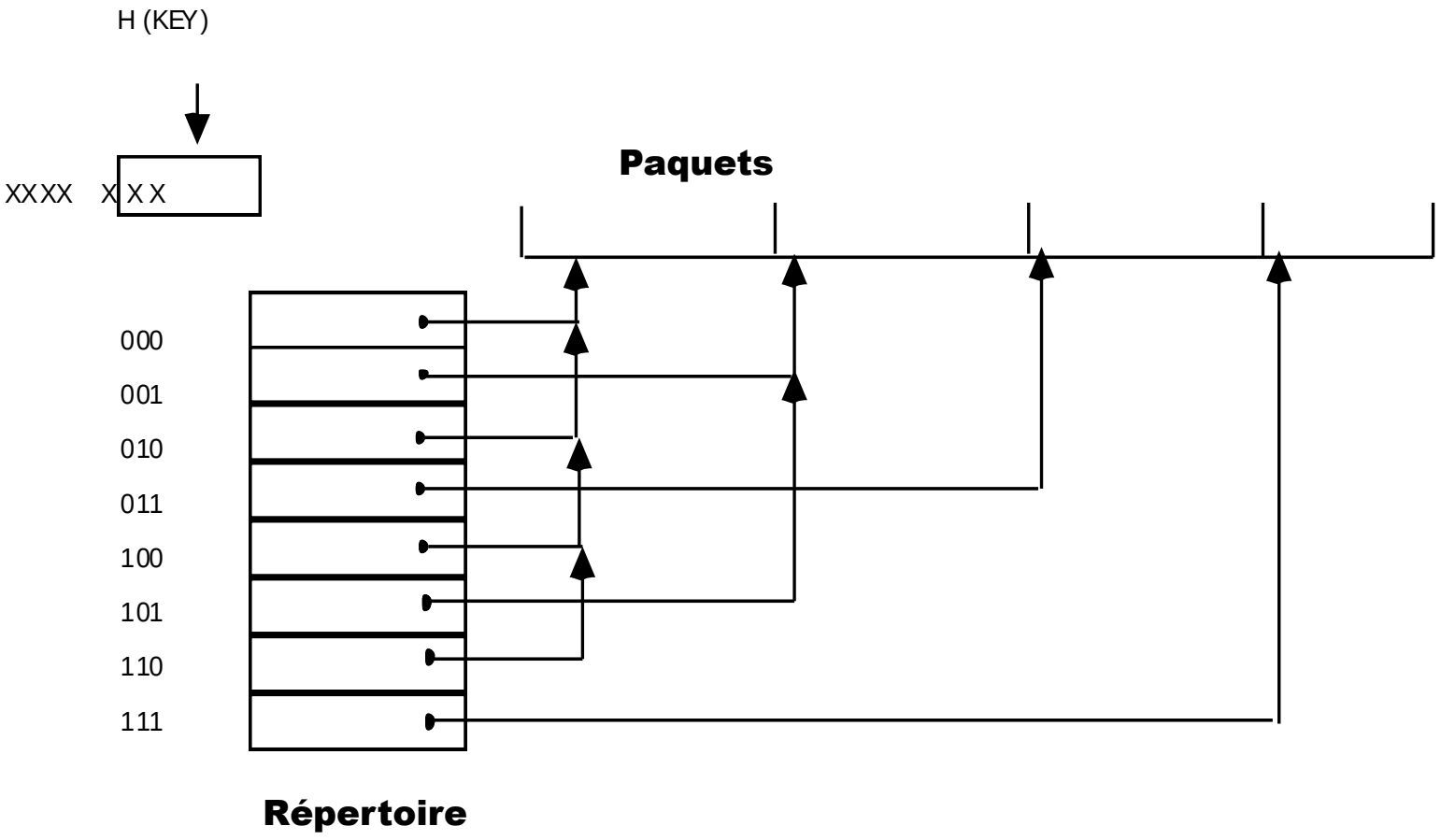
Techniques de hachage dynamique

- Techniques permettant de faire grandir progressivement un fichier haché saturé en distribuant les articles dans de nouvelles régions allouées au fichier.
- LES QUESTIONS CLÉS :
  - (Q1) Quel est le critère retenu pour décider qu'un fichier haché est saturé ?
  - (Q2) Quelle partie du fichier faut-il doubler quand un fichier est saturé?
  - (Q3) Comment retrouver les parties d'un fichier qui ont été doublées et combien de fois ont elles été doublées?
  - (Q4) Faut-il conserver une méthode de débordement et si oui quelle méthode?

Hachage extensible

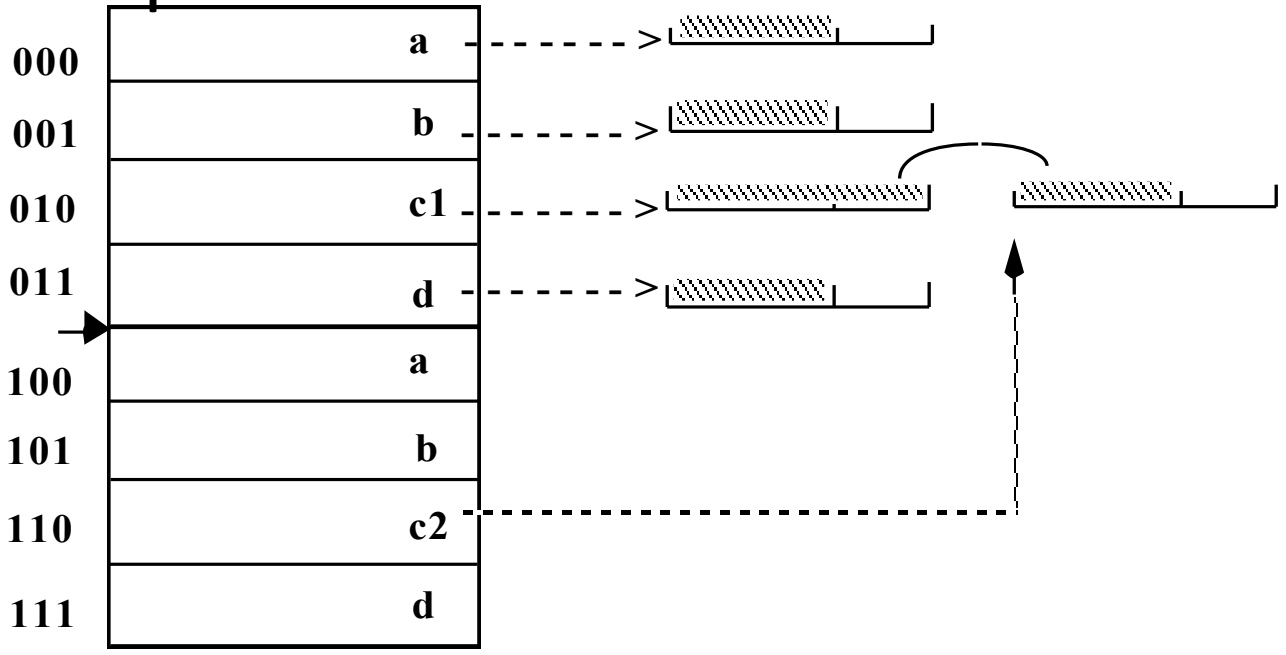
- (Q1) Le fichier est étendu dès qu'un paquet est plein; dans ce cas un nouveau paquet est ajouté au fichier.
- (Q2) Seul le paquet saturé est doublé lors d'une extension
  - Il éclate selon le bit suivant du résultat de la fonction de hachage appliquée à la clé h(K). Les articles ayant ce bit à 0 restent dans le paquet saturé, alors que ceux ayant ce bit à 1 partent dans le nouveau paquet.
- (Q3) Chaque entrée d'un répertoire donne l'adresse d'un paquet.
  - Les 2**(P-Q) adresses correspondant à un paquet qui a éclaté Q fois sont identiques et pointent sur ce paquet; ainsi, par l'indirection du répertoire, le système retrouve les paquets.
- (Q4) La gestion de débordement n'est pas nécessaire.

Fichier haché extensible



Eclatement d'un paquet

- L'entrée jumelle est forcée à l'adresse du nouveau paquet créé si elle pointe sur le paquet éclaté, sinon le répertoire est doublé.



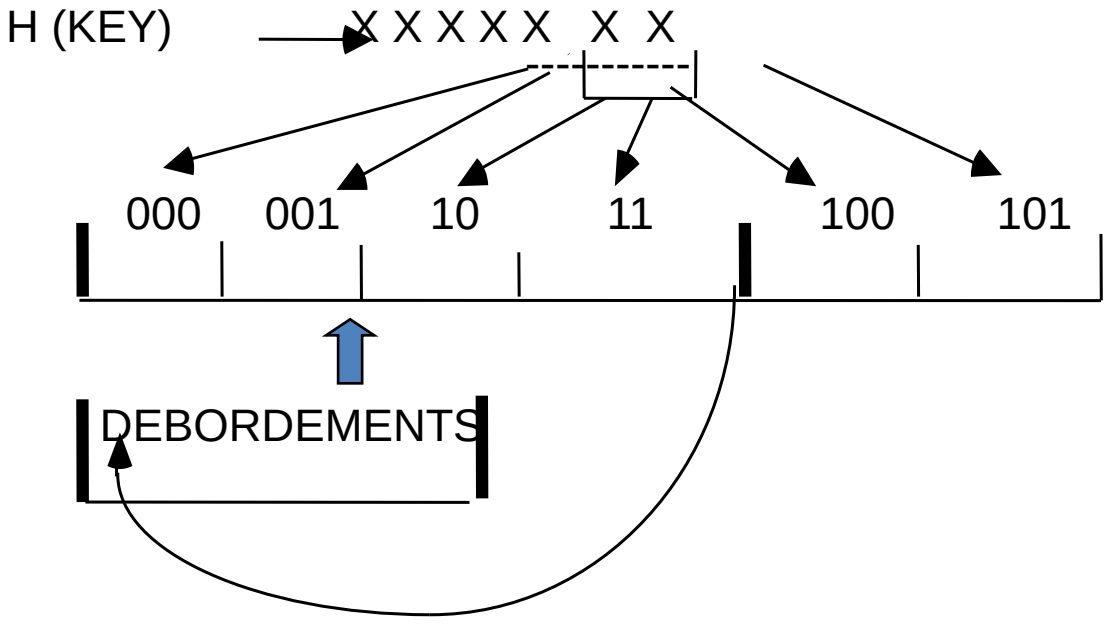
Définition du hachage extensible

- Notion 9: Hachage extensible (Extended hashing)
  - Méthode de hachage dynamique consistant à éclater un paquet plein et à mémoriser l'adresse des paquets dans un répertoire accédé directement par les (M+P) premiers bits de la fonction de hachage où P est le nombre d'éclatements maximum subi par les paquets.

Hachage linéaire

- (Q1) Le fichier est étendu par paquet dès qu'un paquet est plein.
- (Q2) Le paquet doublé n'est pas celui qui est saturé, mais un paquet pointé par un pointeur courant qui parcours le fichier circulairement.
- (Q3) Un niveau d'éclatement P du fichier est conservé dans le descripteur du fichier afin de préciser la fonction de hachage.
  - Pour un paquet situé avant le pointeur courant, (M+P+1) bits de la fonction de hachage doivent être utilisés alors que seulement (M+P) sont à utiliser pour adresser un paquet situé après le pointeur courant.
- (Q4) Une gestion de débordement est nécessaire puisqu'un paquet plein n'est en général pas éclaté.

Paquets d'un fichier haché linéaire



Définition du hachage linéaire

- Notion 10: Hachage linéaire (Linear hashing)
  - Méthode de hachage dynamique nécessitant la gestion de débordement et consistant à:
  - (1) éclater le paquet pointé par un pointeur courant quand un paquet est plein,
  - (2) mémoriser le niveau d'éclatement du fichier afin de déterminer le nombre de bits de la fonction de hachage à appliquer avant et après le pointeur courant.

Comparaison des hachages

	Ecriture	Lecture	Débordement	Répertoire
• Statique	2+d	1+d	oui	non
• Extensible	2+r+e	1+r	non	oui
• Linéaire	2+d+e	1+d	oui	non

Les taux d'occupation de place sont difficiles à comparer.

Le hachage linéaire peut être retardé (éclatement différé selon taux d'occupation).

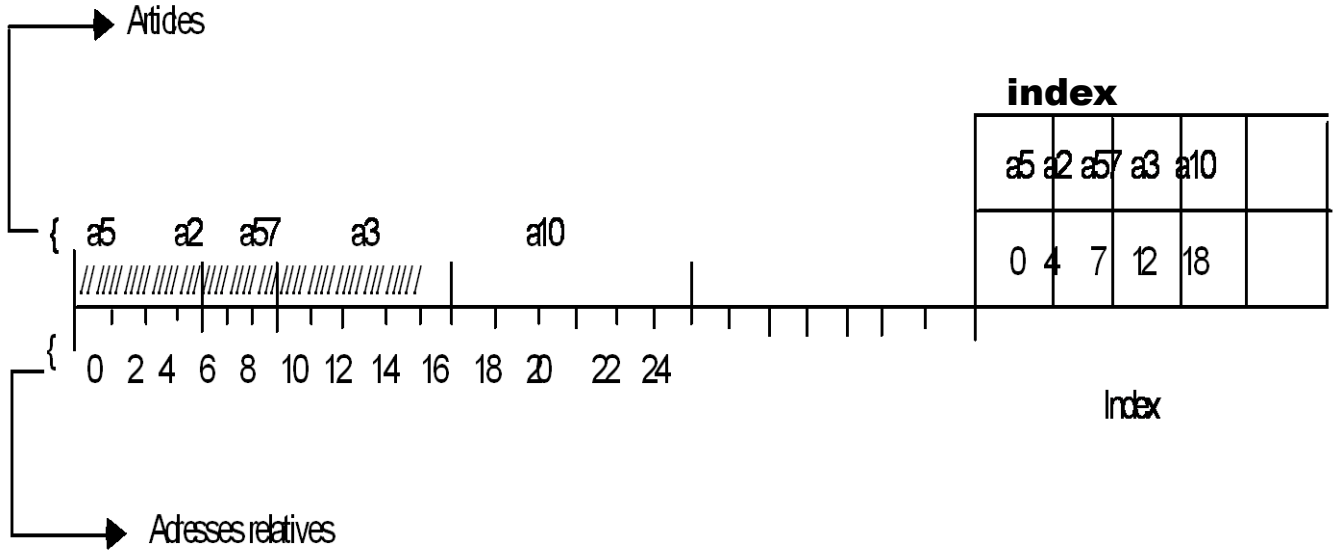
Exercice

- **Hachage multi-atributs**
  - Numéro paquet = h1(A1) || h2(A2)||… hi(Ai) || …
- **Calculer le nombre d'E/S nécessaires pour**
  - Ai = a
- **Choisir la fonction de hachage optimale** pour des fréquences d'interrogation respectives de
  - f1, f2, …fi,…

3. Organisations Indexées

- **OBJECTIFS :**
  - 1) Accès rapide a partir d'une clé
  - 2) Accès séquentiel trié ou non
- **MOYENS :**
  - Utilisation de tables permettant la recherche de l'adresse de l'article a partir de la CLE
- **Notion 11: Index (Index)**
  - Table (ou plusieurs tables) permettant d'associer à une clé d'article l'adresse relative de cet article.

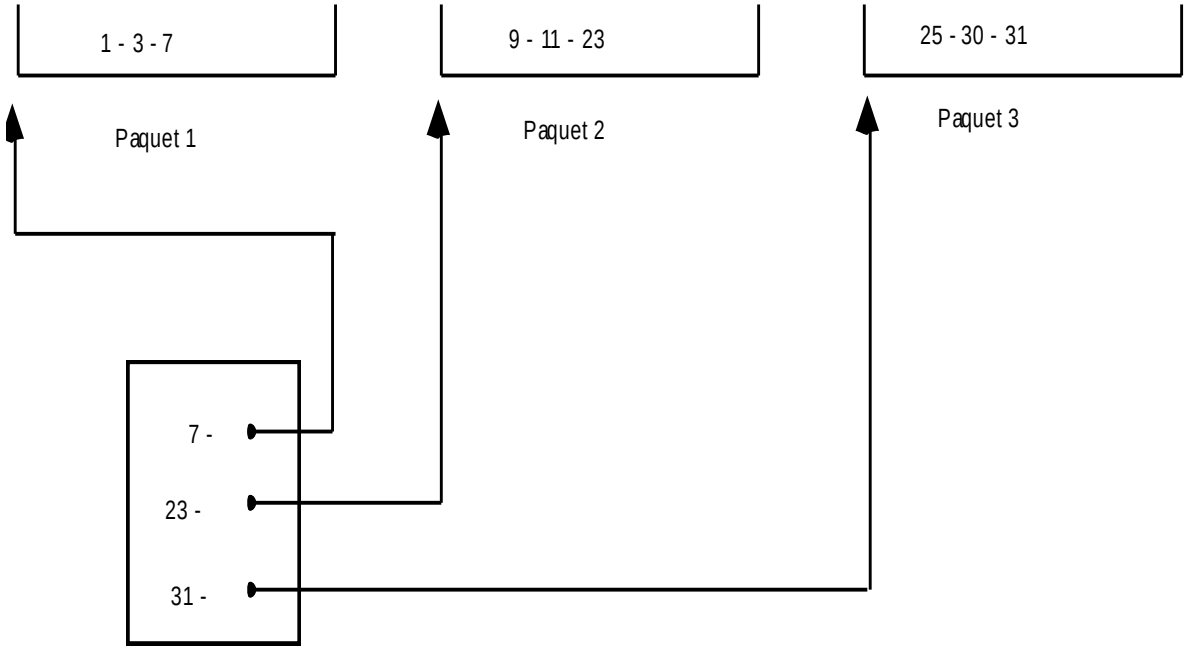
Exemple de fichier indexé



Différents Types d'Indexes

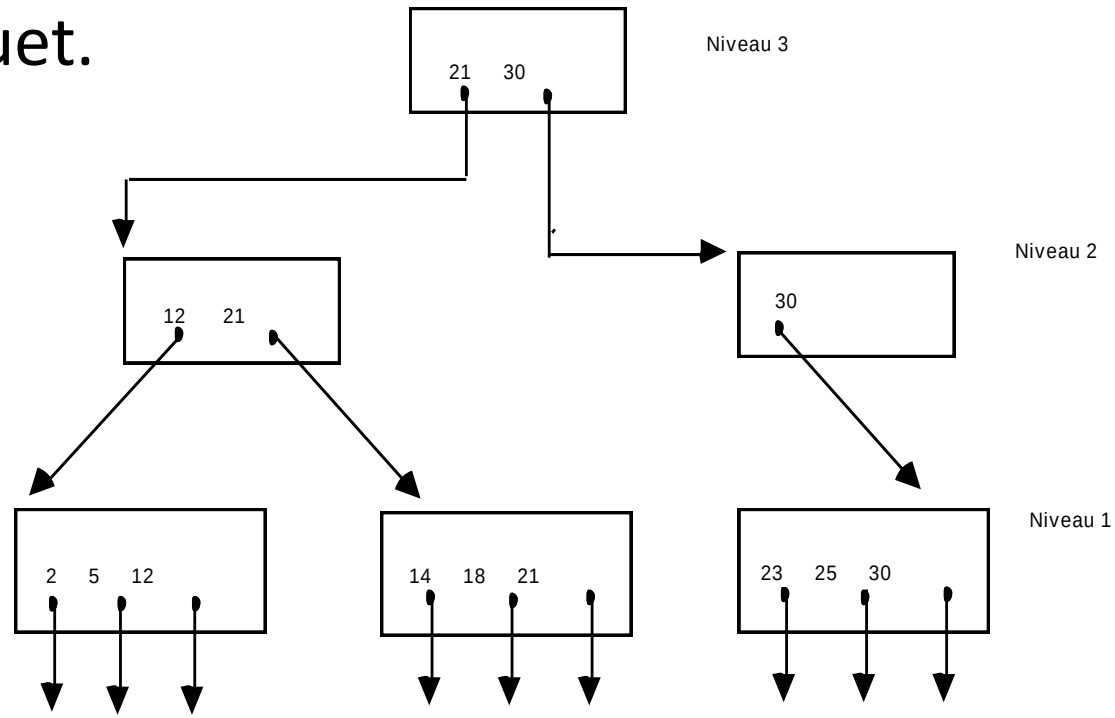
- Un index contenant toutes les cles est dense
- Notion 12: Densité d'un index (Index key selectivity)
  - Quotient du nombre de clés dans l'index sur le nombre d'articles du fichier.
- Un index non dense est possible si le fichier est trie
  - Il contient alors la plus grande clé de chaque bloc avec l'adresse relative du bloc.
- Il est possible de construire des indexes hiérarchisés
  - Chaque index possède alors un index qui permet d'accélérer la recherche.
  - Il est ainsi possible de gérer efficacement de gros fichiers.

Exemple d'index non dense

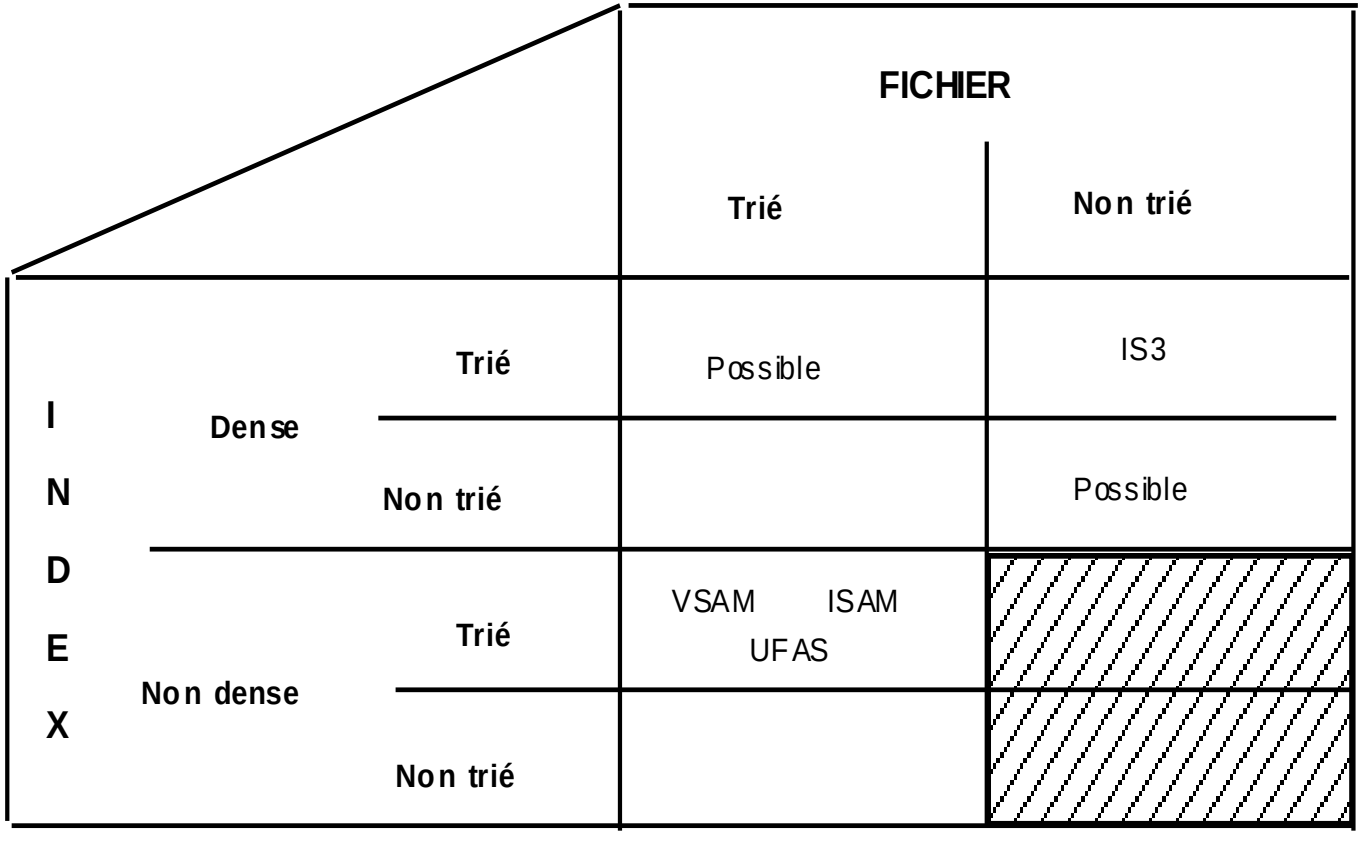


Exemple d'index hiérarchisé

- Notion 13: Index hiérarchisé (Multilevel index)
  - Index à n niveaux, le niveau k étant un index trié divisé en paquets, possédant lui-même un index de niveau k+1, la clé de chaque entrée de ce dernier étant la plus grande du paquet.



Variantes de méthodes indexées

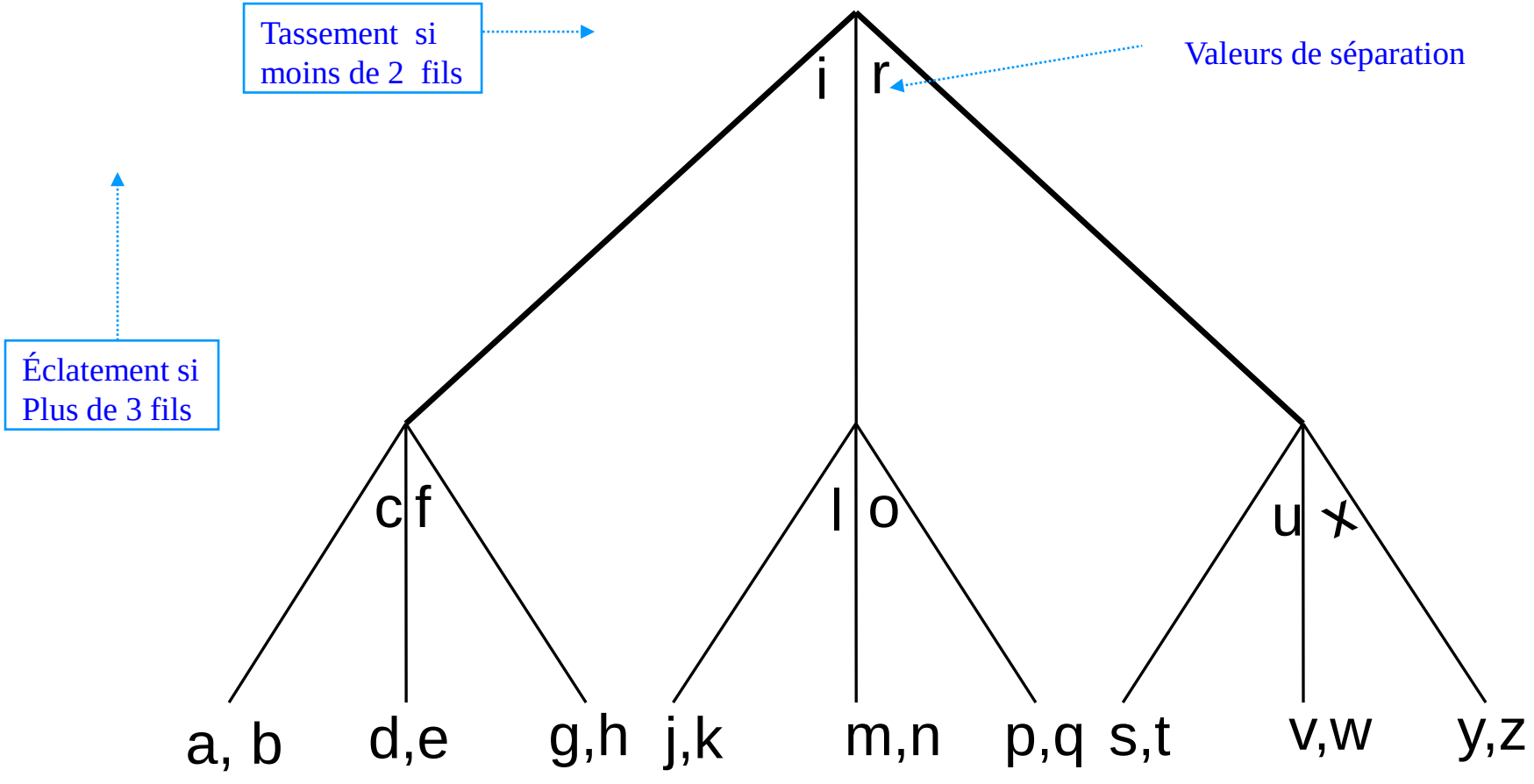


Arbre-B

- Les arbres-B (de Bayer) fournissent des outils de base pour construire des indexes équilibrés.

- Notion 14: Arbre-B (B-tree)
  - Un arbre-B d'ordre m est un arbre au sens de la théorie des graphes tel que:
    - 1) Toutes les feuilles sont au même niveau;
    - 2) Tout nœud non feuille à un nombre NF de fils tel que
      - m+1 <= NF < 2m+1 sauf la racine qui a un nombre NFR de fils tel que 0 <= NFR < 2m+1.

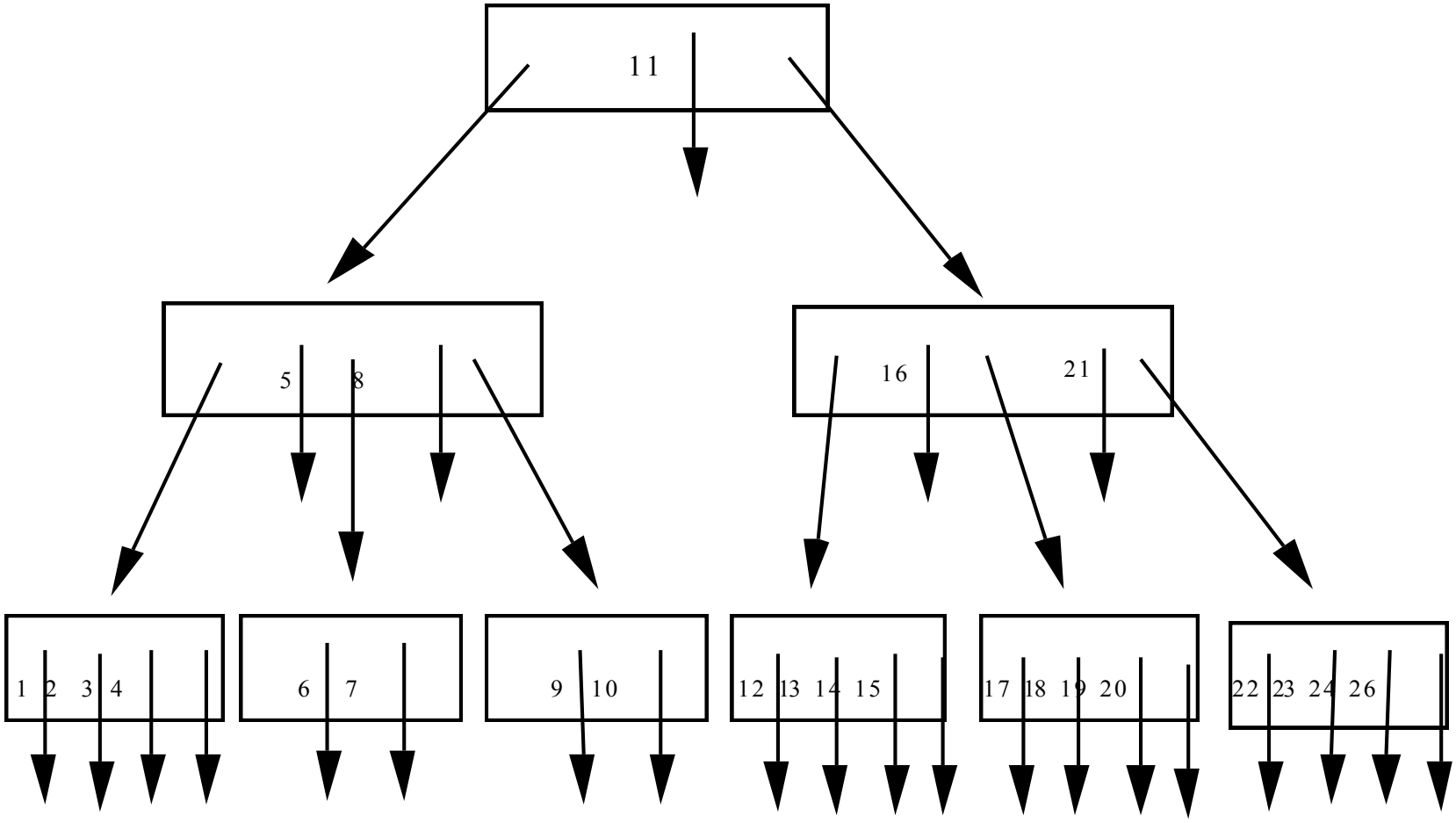
Arbre-B 2-3


Tassement si
moins de 2 fils
Valeurs de séparation
Éclatement si
Plus de 3 fils
i
r
c
f
l
o
u
x
a, b
d,e
g,h
j,k
m,n
p,q
s,t
v,w
y,z

Structure d'un nœud d'un arbre-B

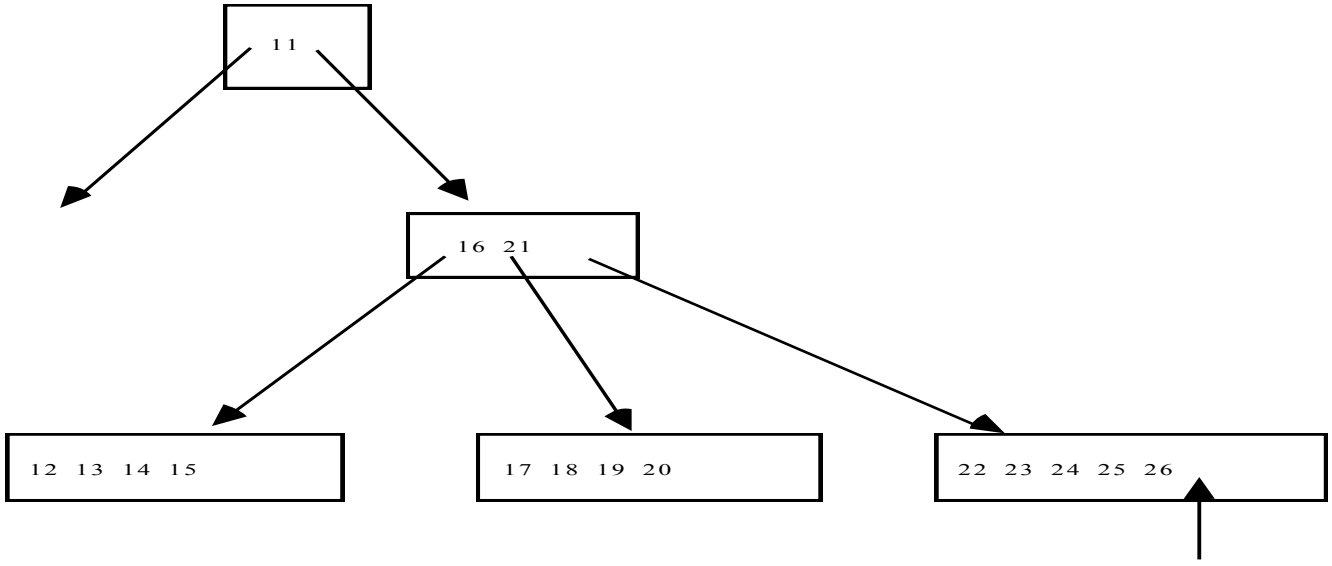
- Pi: Pointeur interne permettant de représenter l'arbre; les feuilles ne contiennent pas de pointeurs Pi;
- ai: Pointeur externe sur une page de données;
- xi: valeur de clé.
- (1) (x1, x2…xK) est une suite croissante de clés;
- (2) Toute clé y de K(P0) est inférieure à x1;
- (3) Toute clé y de K(P1) est comprise entre xi et xi+1;
- (4) Toute clé y de K(PK) est supérieure à xk.

Exemple d'index en arbre-B

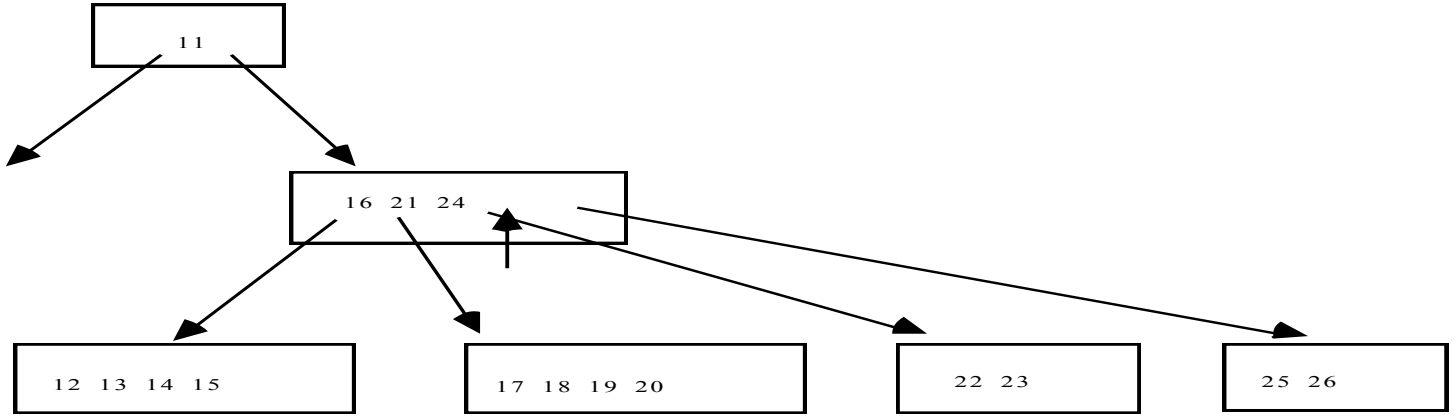


Insertion de la clé 25

(a)



(b)



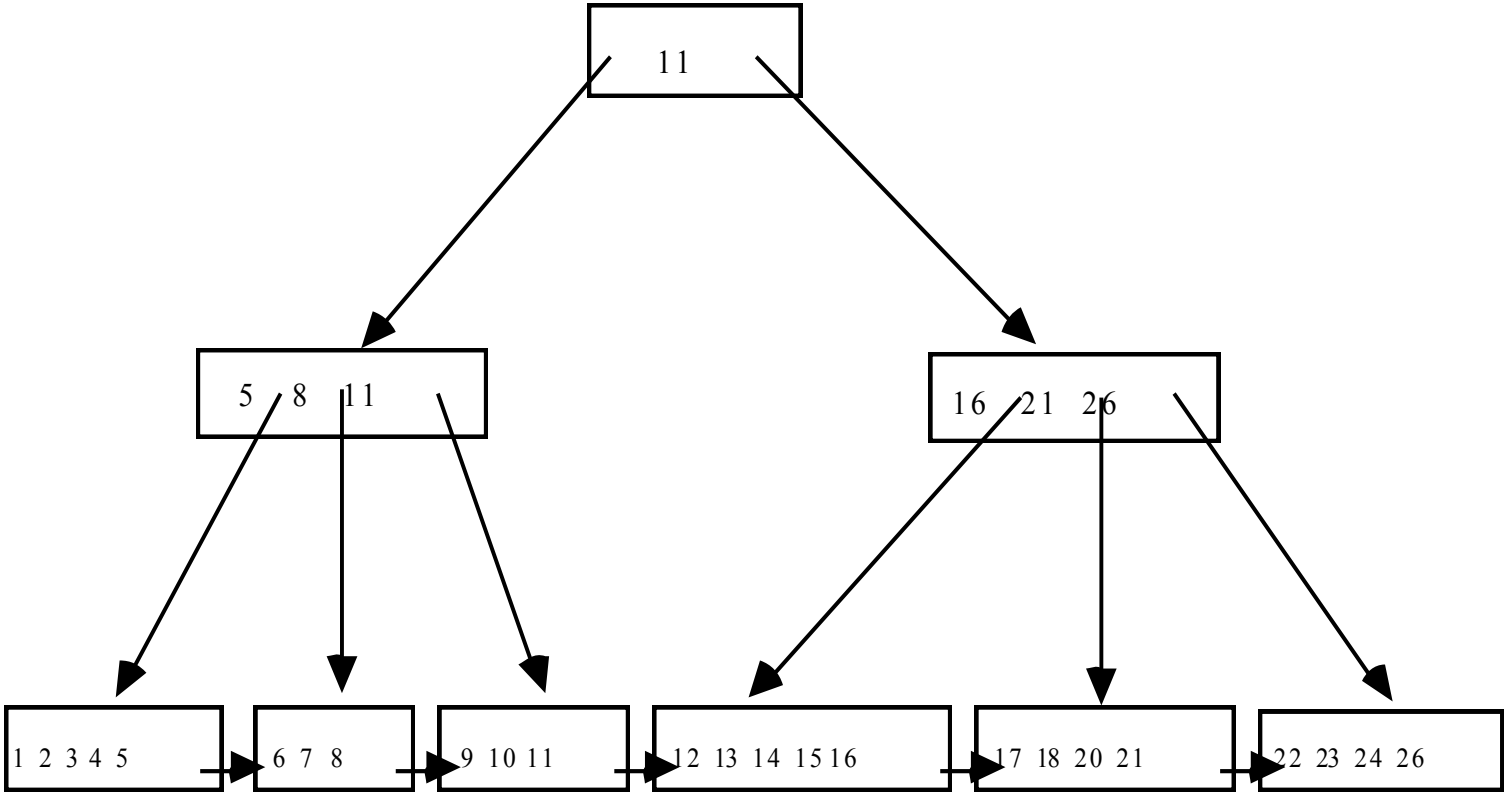
Hauteur d'un Arbre-B

- Le nombre de niveaux d'un arbre-B est déterminée par son degré et le nombre de clés contenues.
- Ainsi, dans le pire des cas, si l'arbre est rempli au minimum, il existe:
  - une clé à la racine,
  - deux branches en partent avec m clés,
  - (m+1) branches en partent avec m clés.
- Pour un arbre de niveaux h, le nombre de clés est donc:
  - $N = 1 + 2\ m\ (1+ (m+1) + (m+1)^2 + \ldots + (m+1)^{h-2})$
  - soit, par réduction du développement limité:
  - $N = 1 + 2\ ((m+1)^{h-1}-1)$
- D'où l'on déduit que pour stocker N clés, il faut:
  - $h = 1 + \log_{m+1}\ ((N+1)/2)$ niveaux.

Arbre-B+

- Notion 15: Arbre B+ (B+ tree)
  - Arbre-B dans lequel on répète les clés des nœuds
  - ascendants dans chaque nœud et on chaîne les nœuds
  - feuilles pour permettre un accès rapide en séquentiel trié.
- Les arbres-b+ sont utilises pour gérer des index hiérarchisés :
  - 1) en mettant toutes les clés des articles dans un arbre B+ et en pointant sur ces articles par des adresses relatives ==> INDEX NON PLACANT
  - 2) en rangeant les articles au plus bas niveau de l'arbre B+ ==> INDEX PLACANT

Exemple d'index en arbre-B+



Avantages et Inconvénients

- Avantages des organisations indexées par arbre-b (b+) :
  - Régularité = pas de réorganisation du fichier nécessaires après de multiples mises à jour.
  - Lecture séquentielle rapide: possibilité de séquentiel physique et logique (trié)
  - Accès rapide en 3 E/S au plus pour des fichiers de 1 M d'articles
- Inconvénients :
  - Les suppressions génèrent des trous difficiles à récupérer
  - Dans le cas d'index non plaçant, la localité est mauvaise pour des accès séquentiels ou sur clés secondaires, ce qui conduit à de nombreux déplacement de bras.
  - Taille de l'index pouvant être importante.

Exercice

- Discuter de la possibilité de mettre plusieurs indexes à un fichier
  - plaçant
  - non plaçant
- Avantages et inconvénient
  - coût de mise à jour
  - coût d’interrogation