Rappels sur les fichiers

Bernard ESPINASSE
Professeur à Aix-Marseille Université (AMU)
Ecole Polytechnique Universitaire de Marseille

Janvier 2018

1. Généralités sur les mémoires

2. Généralités sur les fichiers

3. Types d'organisations de fichiers

4. Exemple d'organisations indexées

Plan

1. Généralités sur les mémoires

Performance et coût des mémoires

- La **Mémoire cache** plus rapide et plus onéreuse que …
- La **RAM** est plus rapide et plus onéreuse que...
- Le **Disque Dur** est plus rapide et plus onéreux que...
- Le **CD** (ou DVD) est plus rapide et plus onéreux que...
- La **Bande magnétique** est plus rapide et plus onéreuse que ...
- Le **Papier** est plus rapide et plus onéreuse que ...

=> Tenir compte de ce qui est disponible et nécessaire en fonction de son usage …

Stockage sur disque dur

- En général, les **SGBD** sauvegardent les données sur les **Disques Durs (DD)**
- Les DD imposent certaines caractéristiques à la conception des SGBD :
  - ***Lecture*** : transfert les données depuis le DD vers la mémoire principale (RAM)
  - ***Ecriture*** : transfert les données de la RAM vers le DD
- Ces opérations ont un coût important

  => nécessitent une planification attentive

Stockage sur Disque Dur versus sur RAM

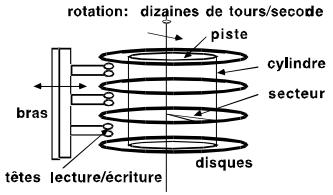
- Le coût du stockage sur RAM reste en général très **élevé** :
  - En 2003, pour **1000**$, DELL vend **3Go RAM** ou un **DD de 800 Go**,
  - En 2010, pour **150**$, DELL vend **8Go de RAM** ou bien un **DD de 1,5To** !
- La RAM est volatile : problème de persistance
- Hiérarchie typique des solutions de stockage :
  - **1. RAM** pour les données exploitées à l'instant t
  - **2. DD** pour le stockage des BD
  - **3. DD** voire **Bandes** pour l'archivage

Disques Durs (1)

- Mémoire **non adressable** directement par instructions du processeur central
- **Adressage** par des **instructions d'entrée-sortie (ES) spécialisée**
- Avantage principal sur les Bandes (offline : intervention humaine): **accès aléatoire** au lieu d'un *accès séquentiel*
- Les données sont stockées et récupérées dans une **unité de bloc disque** : contient plusieurs disques magnétiques fixés à un axe rotatif
- *1956 : premier DD (IBM):* ***1200 tours/minute***
- *2013 : vitesse maximale :* ***15000 tours/minute***

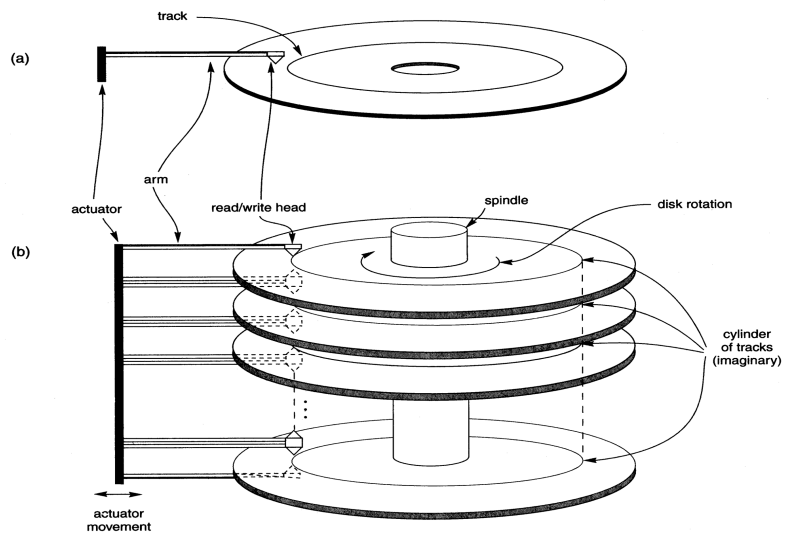
Disques Durs (2)

- Les disques sont divisés en **pistes** (tracks) **concentriques**, chacune d'une capacité de quelques dizaines K octets, définissant des **cylindres**
- Une piste est divisée en **secteurs**
- Les données sont lues/écrites par les têtes par **bloc** ou **page**, lue en **une entrée-sortie (ES)**
- Le temps de récupération de données sur un bloc disque **dépend de la localisation sur le disque** (contrairement à la RAM)



- Exemple :
  - capacité disque : 100 Tera octets ; piste (concentrique, ex. : n piste, n = 512)
  - temps d'accès : 10 milli-secondes, soit 20.000 fois temps accès RAM

Disques Durs (3)



Disques Durs (4)

Accès à un bloc (page ou **ES) :**

- Temps pour sélectionner le bon cylindre par un **mouvement de translation du bras** : *Seek time – qq milli à dizaine de milli sec.*
- Temps de rotation du disque pour le bon secteur (rotation de 4200 à 15000 tours/minute) : *Rotation time (temps de latence) – qq milli sec.*
- Transfert des données : *Transfert time*
- En général :

  Seek time > Rotation time > Transfert time

  le temps que prend un accès à un secteur d'un disque, la CPU peut effectuer des centaines de milliers d'instructions pour
- Facteur pour diminuer les coûts d'ES : réduire les temps de recherche (fichiers sur un même cylindre, …)

Exemple de DD

Description	**Cheetah X15 36LP**	**Cheetah 10K.6**
Model Number	ST336732L	ST3146807L
Form Factor (width)	C 3.5 inch	C 3.5 inch
Height	25.4 mm	25.4 mm
Width	101.6 mm	101.6 mm
Weight	0.68 Kg	0.73 Kg
Capacity/Interface		
Formatted Capacity	36.7 Gbytes	146.8 Gbytes
Interface Type	80-pin	80-pin
Configuration		
Number of disks (physical)	4	4
Number of heads (physical)	8	8
Number of Cylinders	18,479	49,854
Bytes per Sector	512	512
Areal Density	N/A	36,000 Mbits/sq.inch
Track Density	N/A	64,000 Tracks/inch
Recording Density	N/A	570,000 bits/inch

Exemple de DD (suite)

Description	**Cheetah X15 36LP**	**Cheetah 10K.6**
Performance *Transfer Rates*		
Internal Transfer Rate (min)	522 Mbits/sec	475 Mbits/sec
Internal Transfer Rate (max)	709 Mbits/sec	840 Mbits/sec
Formatted Int. Transfer Rate (min)	51 MBytes/sec	43 MBytes/sec
Formatted Int. Transfer Rate (max)	69 MBytes/sec	78 MBytes/sec
External I/O Transfer Rate (max)	320 MBytes/sec	320 MBytes/sec
Seek Times		
Avg. Seek Time (Read)	3.6 msec (typical)	4.7 msec (typical)
Avg. Seek Time (Write)	4.2 msec (typical)	5.2 msec (typical)
Track-to-track Seek,	0.5 msec (typical)	0.3 msec (typical)
Read Track-to-track	0.8 msec (typical)	0.5 msec (typical)
Seek, Write Average Latency	2 msec	2.99 msec
Other		
Default Buffer (cache) size	8,192 Kbytes	8,000 Kbytes
Spindle Speed	15K rpm	10K rpm

2. Généralités sur les fichiers

- Mémoire secondaire, indépendance programme - mémoire secondaire
- Notion de fichier, opérations sur les fichiers, utilisation et activité d'un fichier
- Exploitation d'un fichier : fonction de base et fonctions d'exploitation
- Méthodes d'accès : séquentielles et sélectives

Indépendance Programme/Mémoire secondaire

- Les mémoires secondaires doivent être utilisées par **plusieurs programmes** d'applications
- Pouvoir **partager l'espace mémoire secondaire** entre les données des différents programmes
- Les programmes **indépendants** de l'emplacement des données sur disque
- Possibilité de **changer les données de mémoires secondaires** sans changer les programmes.

=> Les programmes d'application ne peuvent assurer la gestion de cet espace et adresses associées

Pour réaliser cette indépendance, on introduit un objet intermédiaire :

Le Fichier

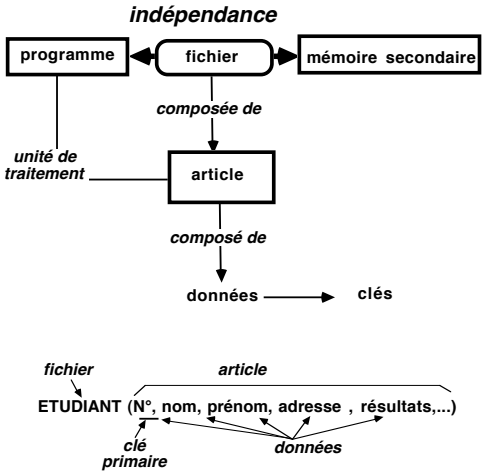
Programme / Fichier / Mémoire secondaire

Définitions

- **Fichiers :** récipient d'informations constituant une mémoire secondaire idéale, caractérisée par un nom.
- **Article/record/enregistrement :**
  - élément composant un fichier, est composé d'un ensemble de données.
  - unité de traitement pour les programmes : un programme traite, lit, écrit des portions successives de fichier (ex : un client, un compte,...),
- **Organisation de fichier :** ensemble des liaisons entre articles pour composer un fichier.
- **Méthodes d'accès :** méthode d'exploitation du fichier utilisée par les programmes pour sélectionner des articles.
- **Clé d'article (primaire, secondaire) :** donnée identificatrice d'un article, permet de sélectionner un article unique dans un fichier.
- **Langage hôte :** langage de programmation accueillant les verbes de manipulations de fichier et la définition des données des fichiers.

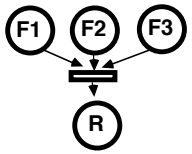
  Ex : Pascal, Open, Reset, Close, Write, ...

Fichier

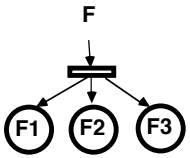


Opérations sur les fichiers

- **Sur tous les articles :**
  - création,
  - destruction,
  - tri (selon un ou plusieurs critères)
  - réunion :



  - éclatement :



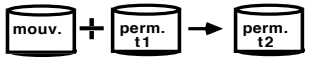
- **Sur 1 ou quelques articles :**
  - consultation d'un article,
  - ajout d'un article,
  - suppression d'un article,
  - mise à jour d'un article.

Opérations élémentaires sur un fichier

- **OPEN** : Prépare le fichier à des opérations de lecture ou écriture, fixe un pointeur sur le début du fichier, allocation de mémoires tampons pour stocker les pages
- **FIND** : Cherche le premier enregistrement qui satisfait la condition de recherche
- **FINDNEXT** : Cherche les enregistrements suivants qui satisfont la condition de recherche
- **READ** : Copier l'enregistrement depuis le buffer vers une variable du programme
- **INSERT** : Insérer un nouvel enregistrement dans le fichier à partir d'une localisation de page
- **DELETE** : Supprime l'enregistrement courant du fichier, souvent en marquant l'invalidité d'un enregistrement
- **MODIFY** : Changer la valeur d'un attribut de l'enregistrement courant
- **CLOSE** : Terminer l'accès à un fichier
- **REORGANIZE** : Réorganiser le fichier, par exemple en supprimant physiquement les enregistrements marqués
- **READ-ORDERED** : Lecture d'un fichier en respectant un ordre spécifique.

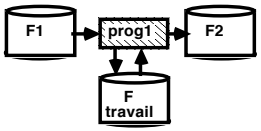
Utilisation d'un fichier

- **Fichier permanent :**
  - de situation
  - d'archivage,
  - d'historique.
- **Fichier de mouvement :** permet la mise à jour d'un fichier permanent :



- **Fichier de liaison :** permet la liaison entre 2 programmes :

- **Fichier de travail :** permet l'exécution d'un programme :



Activité d'un fichier

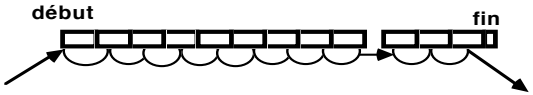
- **Taux de consultation** du fichier F pendant période T = $\tau_{c\,FT}$ =

  nb articles consultés de F pendant T / nb total d'articles de F

- **Taux de mouvement** du fichier F pendant période T = $\tau_{m\,FT}$ =

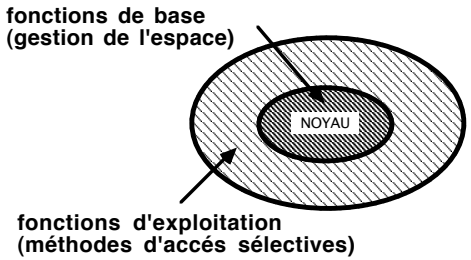
  nb articles modifiés de F pendant T / nb total d'articles de F

- **Soit T = temps d'exécution d'un programme P**
  - *Si* $\tau_{c\,PT} + \tau_{m\,PT}$ *-> 1:* ***méthodes d'accès séquentielles :***



  - *Si* $\tau_{c\,PT} + \tau_{m\,PT}$ *-> 0:* ***méthodes d'accès sélectives :***

Gestion et exploitation de fichiers



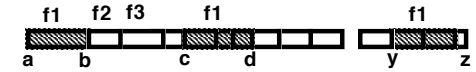
- **Fonctions de base :**
  - création / destruction des fichiers,
  - allocation de la mémoire secondaire,
  - localisation dans la mémoire secondaire,
  - contrôle.
- **Fonctions d'exploitation :**
  - organisation et méthodes d'accès sélectives (utilisent les fonctions de base).

Fonctions de base

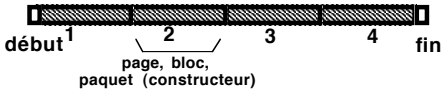
- **Manipulation des fichiers :** instructions accessibles le programmeur et les fonctions d'exploitation permettant la CREATION, DESTRUCTION, OUVERTURE, FERMETURE du fichier

- **Adressage relatif :**
  - fichier généralement discontinu sur mémoire secondaire
  - adresser son contenu à l'aide d'une adresse continue (o...n) = adresse relative :

adressage absolu



adressage relatif



- *taille page ou bloc : n Koctets*
- *adresse relative d'un enregistrement : 3°page, 331°octet*
- *si recopie, adresse relatives inchangées*

Fonctions permettant de :
- ACCEDER à une page de l'adressage relatif
- CONVERTION adresses relative absolue
- ALLOCATION de mémoire secondaire en pages.

Fonction de base : stockage d'enregistrements dans une page (bloc) de disque

- Les enregistrements d'un fichier sont stockés dans des **pages** ou **bloc** (en général 4Bo ou 8Ko) sur le disque dur
- **BFR** = « Blocking Factor » = nombre d'enregistrements par page/bloc :
  - Soit B la taille d'une page et R la taille fixe d'un enregistrement du fichier
  - Si B >= R alors BFR = B/R enregistrements par page/bloc
- Il peut y avoir des **espaces vides** dans un bloc si la totalité d'un enregistrement ne passe pas une page/bloc :
  - Espace non utilisé correspond à : B - (BFR x R) octets
  - Pour exploiter cet espace, possible de stocker une partie d'un enregistrement sur une page et le reste sur une autre page (un pointeur à la fin de la 1ère page pointe sur la page contenant le reste de l'enregistrement - page répartie ou « spanned record »)

Fonctions de base de localisation des fichiers sur les volumes

- **Mémoire secondaire = volume (**bande, disque dur, disque amovible, CD/DVD, clé USB, disquette, …)
- **Label du volume (label) :** premier secteur d'un volume permettant d'identifier le volume
- **Descripteur de fichier :** permet de retrouver les caractéristiques d'un fichier (nom, adresse, ...)
- **Table des matières d'un volume (directory) :** table contenant tous les descripteurs des fichiers du volume.

Fonctions de base de contrôle et de robustesse

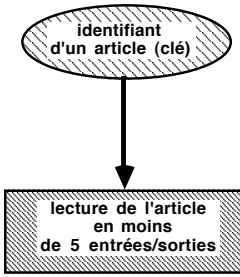
Fonctions permettant des **contrôles et de mécanismes de reprises** pour :
- partage des fichiers,
- résistance aux pannes,
- confidentialité, …

3. Types d'organisation de fichiers

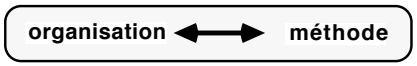
- Organisation et méthodes d'accès
- Organisation relative
- Organisation aléatoire
- Organisation indexée

Organisations et méthodes d'accès sélectives

- **problème :**



- **solution :**

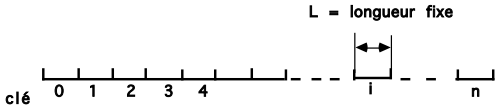


Organisation relative

clé = nombre entier = numéro de l'article (adresse relative de l'article dans le fichier)

Fichier relatif ou direct :

fichier d'articles de longueur fixe où chaque article possède une clé unique = son n° d'ordre dans le fichier.



adresse relative = L x i

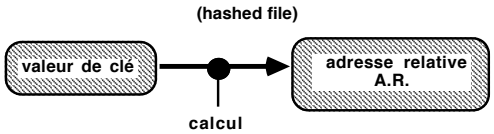
avantages :

- simple, performante (1 entrée/sortie)

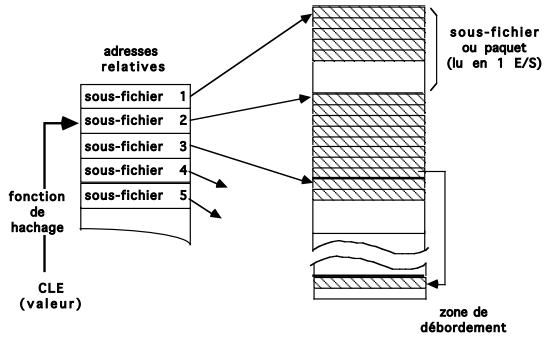
inconvénients :

- article de longueur fixe,
- association nombre entier <-> article
- gestion des trous -> perte place

Organisation aléatoire



A.R. = f (clé) *f = fonction de randomisation (ou fonction de hachage)*



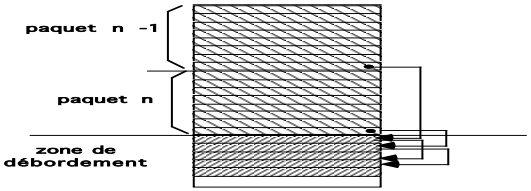
fonctions de hachage :

- pliage : choix et combinaison de bits de la clé,
- conversion de la clé en nombre entier,
- modulo : prendre pour numéro de paquet le reste de la division de la clé par le nombre de paquets,
- combinaisons de techniques ....

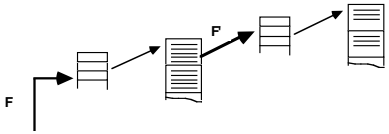
Organisation aléatoire : le problème du débordement

quelques techniques :

- **adressage ouvert :** placer l'article dans le premier paquet libre-> *mémorisation des paquets dans lesquels un paquet a débordé*
- **chaînage :** constituer un paquet logique pour chainage :



- **rehachage :** appliquer une deuxième fonction de hachage lorsqu'un paquet est plein :



Organisation aléatoire

Avantages :

- s'adapte à des fichiers de clés quelconques,
- simple,
- très bonnes performances si peu de débordement.

  -> choix d'une bonne fonction de hachage

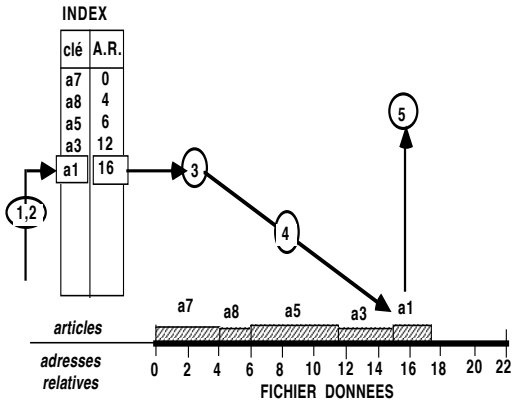
  -> voire ajustement de celle-ci

Inconvénients :

- coût de la gestion de débordement,
- taux d'occupation de la mémoire secondaire peut être faible,
- taille fixée a priori, (hachage virtuel Scholl 81)
- les articles sont sélectionnés sur une valeur exacte de la clé,
- peu de duplication de clé.

Organisations indexées (1)

Index : table(s) associant à une clé, l'adresse relative de l'article où elle se trouve.



1- accès à l'index,

2- recherche sur la clé de l'A.R.,

3- conversion A.R. adresse absolue,

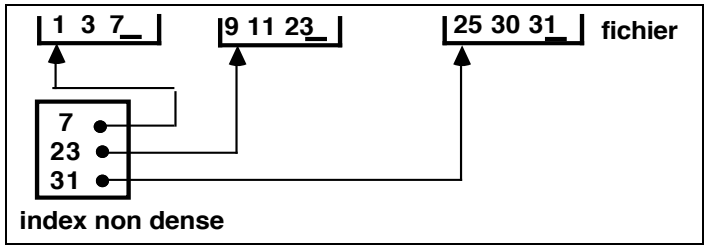
4- accès à l'article,

5- transfert de l'article dans zone programme.

Organisations indexées (2)

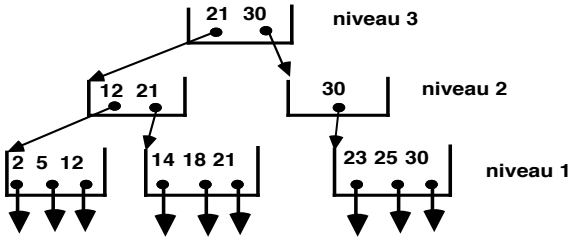
L'index peut être:

- **dans** le fichier des données ou **séparé** du fichier de données
- **trié** (permet recherche dichotomique) ou **non trié**
- **dense** ou **non dense** (densité = nb de clés dans l'index / nb d'articles dans fichier)



Index hiérarchisés

- **Index hiérarchisé à 2 niveaux** :
  - index trié divisé en paquets, possédant lui-même un index, dont chaque entrée est la clé la plus grande de chacun des paquets :



  - le nb de niveau peut grandir autant que nécessaire.

Arbres balancés (B-TREE) :

Arbre balancé d'ordre d :

- **Chaque noeud contient k clés** avec d ≤ k ≤ 2d (sauf la racine : l < k < 2d)
- Un noeud est :
  - soit **terminal**, soit possède (k = 1) fils,
  - le **ième fils possède des clés comprises** entre la (i - l) et la i clé du père,
- L'arbre est **équilibré** : **noeuds terminaux au même niveau.**

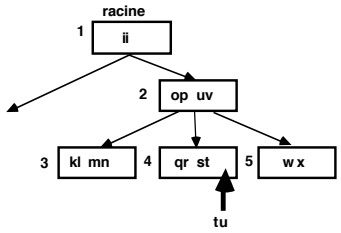
Un noeud à la forme :

P0 K1 P1 K2 P2 ....Pn-1 Kn Pn

où Ki représente les clés et Pi représente les pointeurs

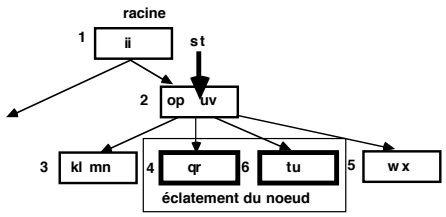
Recherche et insertion dans un B-TREE

1

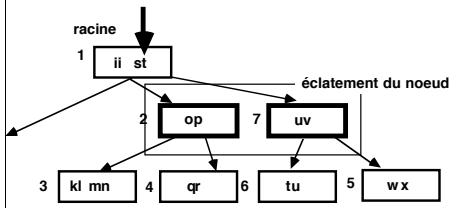


- **rechercher la feuille qui devrait contenir la clé tu à insérer**
- **si le nombre de clés après insertion > m-1 alors:**
  - **migration de la clé médiane st au niveau supérieur**
  - **éclatement du noeud saturé en 2 noeuds plus vides**

2



3



Example d'insertion dans un B-TREE :

Un exemple sur **YouTube** :

https://www.youtube.com/watch?v=coRJrcIYbF4

Suppression de clé dans un B-TREE

- Si **suppression de clé de feuille**, alors remontée de la clé suivante de la feuille
- Si **noeud a un nb de clés < (m/2 -1)** alors remontée sur noeud précédent

Nombre de niveaux d'un B-TREE :

nombre de niveaux L d'un B-tree d'ordre m permettant de stocker N clés :

$$L = 1+\log_{m+1}((N+1)/2)$$

Exemple :

N = 999 999 clès

m =99

alors $L = 1+ \log_{100} 10^6 = 4$ ***niveaux***

Utilisation de B-TREE

Index hierarchisés :

2 façons :

- **index organisé en B-TREE :**
  - clés organisées en B-TREE
  - articles stockés en fichier classique

    Ex: méthode IBM indexé série 3 (IS3)
  - toutes les clés des articles restent au niveau le plus bas de l'arbre B

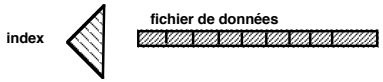
- **index et fichier organisé en B-TREE :**
  - articles stockés dans l'arbre, niveau inférieur,
  - les clés sont déplacées au niveau supérieur.

    Ex: méthode IBM séquentielle indexée régulière (VSAM)

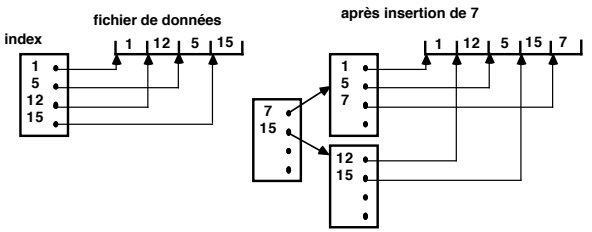
4. Exemples d'organisation indexées

- Organisation IS3
- Organisation ISAM
- Organisation VSAM

Organisation indexée IS3



- **Index :**
  - organisé en B-TREE,
  - dense (dernier niveau du B-TREE)
  - noeud = page = là p secteurs,
  - stocké indépendamment (historique d'index, régénération ...)
- **fichier:** insertion des articles en séquentiel dans le fichier (simple)



Organisation indexée IS3

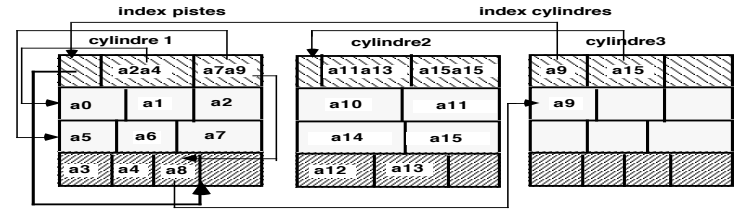
Avantages :

- historique, génération d'index,
- insertion simple des articles,
- performances bonnes : m = nombre clés/page index

Inconvénients :

- séparation fichier/index mouvement de bras,
- lecture du fichier par clé via index coûteuse,
- index dense grande taille.

Organisation séquentiel indexée ISAM (Indexed Sequential Acces Method IBM 1978)



(d'après G.Gardarin)

Zone primaire : articles enregistrés par ordre croissant des clés.

Zone de débordement : (pistes de débordement de cylindres) articles chainés entre eux

- **insertion :**
  - recherche de sa séquence
  - si placé en zone primaire articles suivants déplacés et dernier mis en zone de débordement, chainages mis à jour,
  - si placé en zone débordement inséré dans chaine.

Organisation ISAM

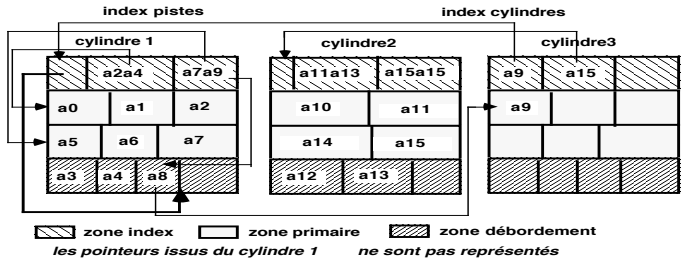
Zone index (non dense) : 2 niveaux d'index:

- index de pistes :

- 1 par cylindre (1°piste), pour chaque piste du cylindre contient:
  - la plus grande clé de la zone primaire
  - la plus grande clé de la zone débordement
  - adresse 1° article en zone débordement

- index de cylindre :

- 1 par fichier, pour chaque fichier contient la plus grande clé du cylindre



(d'après G.Gardarin)

Organisation ISAM

Avantages :

- le fichier est trié accés en séquentiel trié,
- temps d'accés bon si pas de débordement (3 E/S pour un article)

Inconvénients :

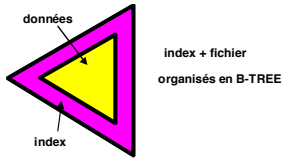
- gestion complexe des débordements
- réorganisation périodique des fichiers ayant débordés (utilitaires)
- méthode d'accés dépendante des caractéristiques physique du support (pistes, cylindres ...)
- performances dégradées si débordement :

  d = nb articles en débordement/piste

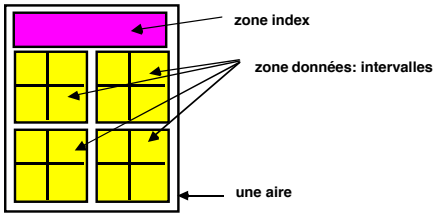
  1 lecture = (3 + d/2) E/S (équivalents E/S)

  1 écriture = (2 + d/2 + 4) E/S

Organisation séquentielle indexée VSAM (OS/VS) (Virtual Sequential Access Method IBM 1978)



- Fichier + index divisé en **aires :** *aire = {pistes d'un même cylindre ou cylindres contigus}*
- aire divisée en **intervalles** : *intervalle = {pistes ou partie de piste lue(s) en 1 E/S}*



avantage / ISAM : organisation indépendante de la mémoire secondaire (notion d'aire) et intervalle et non cylindre et piste.

Organisation VSAM

- **Zone des données :**
  - lors de la première écriture les articles sont enregistrés, triés,
  - de la place libre est laissée dans chaque intervalle et des intervalles libres laissés dans chaque aire pour insertions future d'articles,

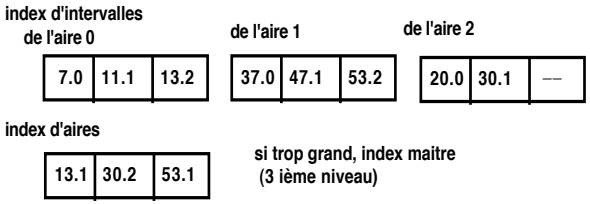
paramètres :

- % octets libres / intervalles = 25%
- % intervalles libres / aire = 25%

Organisation VSAM

- **Zone index :**
  - Index trié non dense
  - index d'intervalle et index d'aire

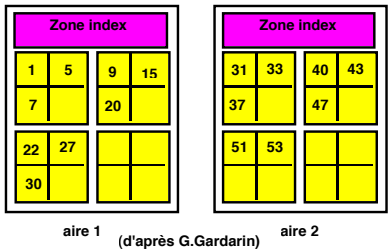
Organisation VSAM : détail zone index



- index d'intervalle (un par aire) : contient la plus gde clé de l'intervalle et son adresse
- index d'aires (un par fichier) : contient la plus grande clé de l'aire ainsi que son adresse. *(si index d'aires trop grand, possibilité 3 niveaux : index maitre)*

Organisation VSAM

- **Après première écriture :**

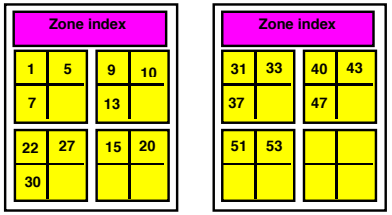


(d'après G.Gardarin)

- **Insertion sans restructuration :**

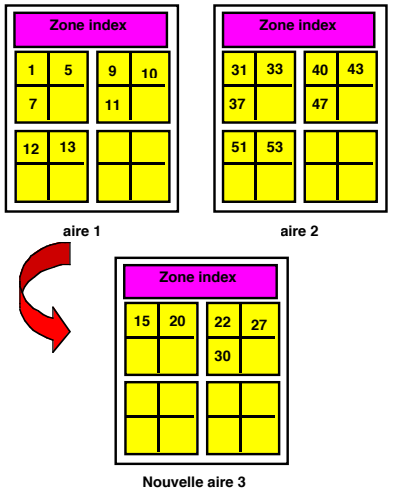
Insertion avec restructuration dans VSAM

1) restructuration d'intervalles



(d'après G.Gardarin)

2) restructuration d'aires



Organisation VSAM

- **Avantages :**
  - organisation indépendante de la mémoire secondaire (notions d'aire et intervalle)
  - fichier données trié (facilité accés séquentiel trié et temps d'accés à un article)
  - performance : lecture en moins de 3 E/S
  - ...
- **Inconvénients :**
  - peu nombreux,
  - écriture peut être longue si éclatement d'intervalles, ou pire d'aires,
  - non régénération facile des index
  - ...