

# STOCKAGE DES DONNÉES

---

**Nicole Bidoit** Université Paris XI, Centre d'Orsay [bidoit@lri.fr](mailto:bidoit@lri.fr)

## Stockage des données

- Comment stocker et gérer de grands volumes de données ?  
**gestion des mémoires** (centrale, secondaire, tertiaire)  
**organisation des données dans les fichiers**
- Quelles représentations et quelles structures de données pour des manipulations performantes ?  
**organisation pour un accès rapide aux fichiers**

## Capacité Mémoire – Approximation

---

(rappel)

- mesure capacité mémoire en  $2^n$  !

$$1 \text{ Kilo} = 10^3 \approx 2^{10} = 1024$$

$$1 \text{ Mega} = 10^6 \approx 2^{20}$$

$$1 \text{ Giga} = 10^9 \approx 2^{30} = 1074 \times 10^9$$

$$1 \text{ Tera} = 10^{12} \approx 2^{40}$$

$$1 \text{ Peta} = 10^{15} \approx 2^{50}$$

- adressage sur 32 octets i.e.  $2^{32}$  adresses  $\approx$  **1 billion d'adresses**

<b>Mémoire cache / vive</b>	
taille	1Mo / $10^2$ Mo
accès (sec.)	$\approx 10^{-8}$ / $\approx 10^{-8} - 10^{-7}$

RAM

<b>Mémoire secondaire</b>	
taille	$10^{12}$ Mo (Gigas)
accès (sec.)	$\approx 10^{-2}$

Disques

<b>Mémoire d'archivage</b>	
taille	$10^{15}$ Mo (Teras)
accès (sec.)	$\approx 1$ sec.

Jukeboxes

Bandes

## Hiérarchie de Mémoire

## Caractéristiques

- Capacité
- Temps d'accès
- Volatilité / fiabilité
- Coût

## Hiérarchie de Mémoire

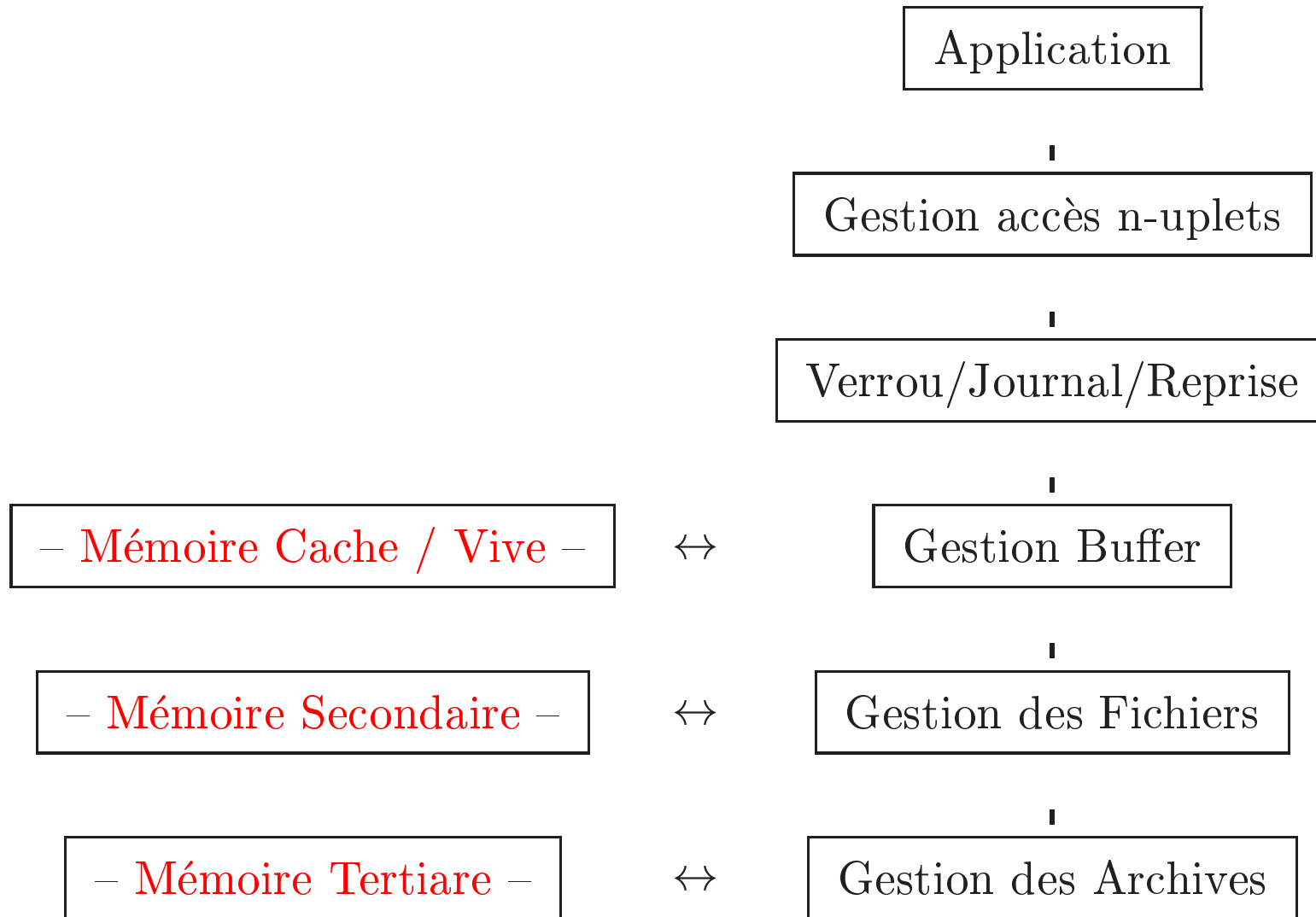
## Usages

- Fréquence d'accès
- Pertinence des données (age)

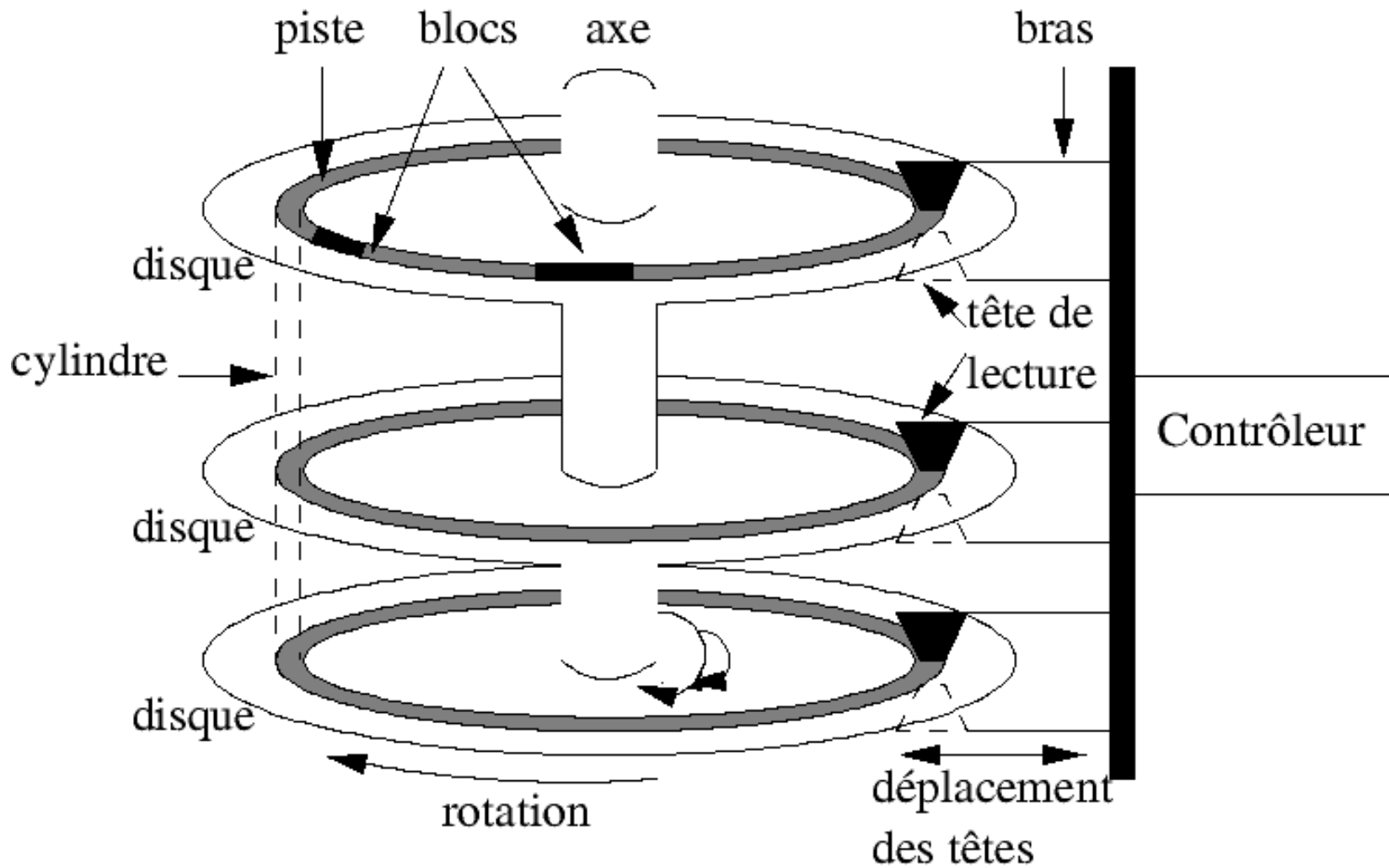
**1 accès disque  $\approx$  1 million d'accès mémoire principale**

## Hiérarchie de Mémoire

## Functionalités SGBD



## Le disque



## Le disque

## quelques caractéristiques

- Vitesse de rotation (1/millise.)
- Nombre de plateaux (5 plateaux i.e. 10 faces)
- Nombre de pistes/face ( $10^4$  pistes)
- Nombre d'octets/pistes ( $\geq 10^5$ )

## Temps d'accès disque

- processeur + contrôleur de disque négligé
- recherche = temps de positionnement sur piste
- rotation = temps de positionnement début bloc/secteur
- transfert = parcours du bloc/secteur

**temps de latence disque**

- **Traitement** données  $\Rightarrow$  **Transfert** des données
- Unité transfert = bloc ou page (Lire / Ecrire)  
**On lit toujours au moins 1 bloc même pour lire 1 seul octet!**
- **Temps/coût** transfert = **fonction**(position bloc à lire)
- Temps d'accès  $\gg$  Temps de traitement

---

**OPTIMISATION = STRATÉGIE DE PLACEMENT**

**SGBD  $\neq$  Système de fichiers du SE**

### REGROUPEMENT

- (exemple) organisation des données sur disque
  - stocker 5 strings de 1 koctets de façon dispersée
  - stocker 5 strings de 1 koctets sur 2 blocs consécutifs
  - gain = 1 à 5**
- regroupement resp. à **proximité maximale**
  1. même bloc
  2. blocs consécutifs
  3. blocs de la même piste
  4. blocs du même cylindre... distance en nombre de pistes à parcourir

## Optimisation et Placement sur Disque

---

### SEQUENCEMENT

- Adapter le regroupement aux **accès concurrents**  
demande de lecture du fichier F1  
demande de lecture du fichier F2  
⇒ alternance entre les lectures des blocs de F1 et de F2  
stockage “en blocs consécutifs” devient inefficace
- séquençement des demandes  
en les triant par piste, puis par bloc au sein d’une piste



## Optimisation et Placement sur Disque

---

### **SEQUENCEMENT** (suite)

- **technique de l'ascenseur**

déplacement régulier des têtes de lecture

du bord du disque vers l'axe, puis

de l'axe vers le bord

efficace : beaucoup de processus, peu de blocs

inefficace : un des processus lit beaucoup de blocs

### **MÉMOIRE TAMPON ou BUFFER**

- **gestion du (des) buffer(s)** en mémoire principale  
(voir suite du cours)

## Gestionnaire espace disque

---

- **Unité** de données = bloc / **page**
- **Allocation** de page
- **Désallocation** de page
- **Allocation** d'une **séquence** de pages

---

### 2 techniques de base

---

1. **Liste** des blocs libres + adresse 1er bloc libre  
**Inconvénient** : retrouver les blocs libres contigus
2. **Matrice** de booléen (1 cellule = 1 bloc)  
**Avantage** : Identification des zones contigues sur disque

## SE versus BD

- Le SE gère l'espace disque (gestion des fichiers).
- Pourquoi le SGBD n'utilise pas le gestionnaire de disque du SE ?

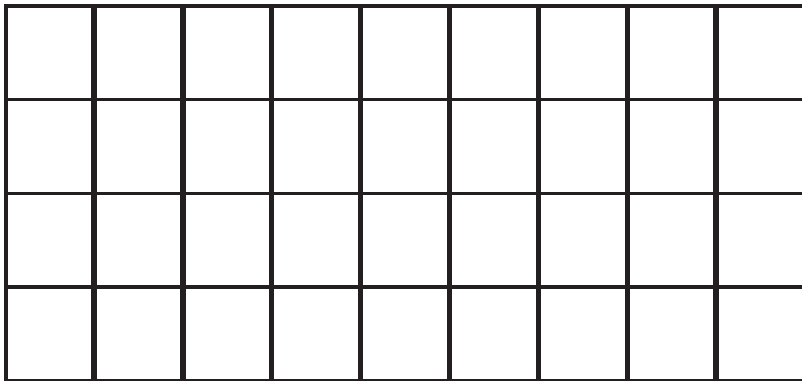
\_\_\_\_\_ **3 arguments forts** \_\_\_\_\_ attention : évolutions technologiques \_\_\_\_\_

1. **portabilité** installation sur différentes plateformes SE
2. **grande masse de données**  
adressage 32 octets  $\rightarrow |\text{fichier}| \leq 4\text{GB}$   
un fichier "SE" doit tenir sur un disque  
impossibilité d'emjamber plusieurs disques
3. **optimisation accès séquentiels et concurrents**  
pas la problématique d'un SE

## Buffer et Gestionnaire du Buffer

- **taille fichier**

modules de haut-niveau



Buffer en mémoire principale



DISQUE

- **accès concurrents**

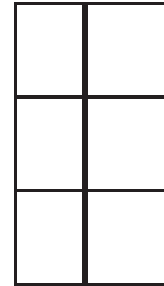
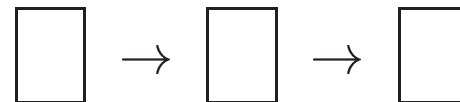


Table de hachage



Liste pages libres

## Gestionnaire de Buffer

- **Gestion transparente des transferts** pour les applications et les modules de haut niveau du SGBD.
- **Interface** module “haut-niveau” ↔ buffer
  - demander une page (ensemble de pages)
  - libérer une page
  - notifier la modification de la page
- le gestionnaire maintient 2 variables par cellule
  - compteur** = nbre de demandes page (pin-count)
  - état** = page modifiée (dirty bit)

## Gestionnaire de Buffer

- **Initialisation** : pin-count=0 ; dirty=0
- **Traitement d'une demande de page** :

Page dans Buffer ?

non (a) choix cellule – stratégie de remplacement –

(b) cellule occupée ?

si page cellule dirty alors écriture disque

(c) transfert page demandée vers cellule

Mise à jour pin-count et dirty

Notification module demandeur

- **stratégie LRU** – least recently used –  
↪ gérer une file de pointeurs sur les cellules “libérées”

- **stratégie de “CLOCK remplacement”**

5	6	7
4	1	8
3	2	9

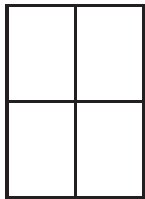
variable CEL (valeur 1 à 9)

variable REF = 1 quand pin-count s'annule

variable REF = 0 quand cellule testée

- **stratégies FIFO, MRU, random ...**

- **Parcours séquentiels itérés (sequential flooding)**



$|\text{Buffer}|=4$  cellules

Effectuer 3 parcours séquentiels  
du fichier F avec  $|\text{F}|=5$  pages  
(stratégie LRU)

---

**Les systèmes**

- Variantes LRU / Clock (variantes)
- Méthodes ad hoc pour résoudre le sequential flooding
- Techniques sophistiquées (DB2 d'IBM) :
  - buffer partitionné (plusieurs buffers)
  - 1 stratégie par partition (par buffer)

- Buffer  $\approx$  mémoire virtuelle
- Pourquoi le SGBD n'utilise pas le gestionnaire de MV du SE ?

\_\_\_\_\_ **3 arguments forts** \_\_\_\_\_ Attention : évolutions technologiques \_\_\_\_\_

1. **portabilité** :
2. **optimisation** : prévision des pages à charger  
préchargement (prefetching)
3. **persistance** : contrôle de l'écriture page en MS (journalisation)

- **Stockage sur disque** : fonctionnalité sensible pour le SGBD  
performance et **sécurité** en cas de panne
- **RAID : redundant Array of Independent Disk**      répartition  
(dupliquer) les données sur  $n$  disques  
défaillance 1 disque  $\rightarrow$  disponibilité maintenue

**RAID 1** : solution brutale

2 disques gérés en parallèle

2 fois + de disques

pas d'amélioration notable des performances

## Technologie RAID

---

**RAID 4** : combine 2 techniques

traiter  $n$  disques comme un seul

**1 seul disque pour la réplication** des données

copies des données : bit de parité

1 disque indisponible → rétablissement

gestion intelligente des redondances

**optimisation** : lectures en parallèle (blocs distribués)

**inconvenient** : écritures fréquentes sur le disque de réplication

+ panne de plusieurs disques

**RAID 5** : Répartition des bits de parité

**RAID 6** : Récupération après panne de plusieurs disques

## Codage BD - Codage Relation - Codage n-uple

- BD  $\approx$  collection de fichiers
- Relation  $\approx$  1 fichier (paginé)                      2 relations dans 1 fichier
- N-uplet  $\approx$  enregistrement

### \_\_\_\_\_opérations de référence \_\_\_\_\_

**recherche** d'un (champ) enregistrement

**insertion** d'un enregistrement

**suppression** d'un enregistrement

**modification** d'un (champ) enregistrement

## Enregistrement

## Longueur fixe

### consultation du dictionnaire des données

- nbre fixe d'attributs (toujours !)
- taille chaque champ (valeur d'attribut) fixe
- taille connue

### enregistrement (sans entête)

$C_1$	$C_2$	$C_3$	$C_4$
-------	-------	-------	-------

$C_i$  :  $i$ ème champ

$l_1$     $l_2$     $l_3$     $l_4$

$l_i$  = longueur  $i$ ème champ

adresse début enregistrement :  $B$

adresse début 3ème champ :  $B + l_1 + l_2$

## Enregistrement

## Longueur variable

- nbre fixe d'attributs (toujours !)
- taille variable de certain champ (chaine de caractères)

### Solution 1

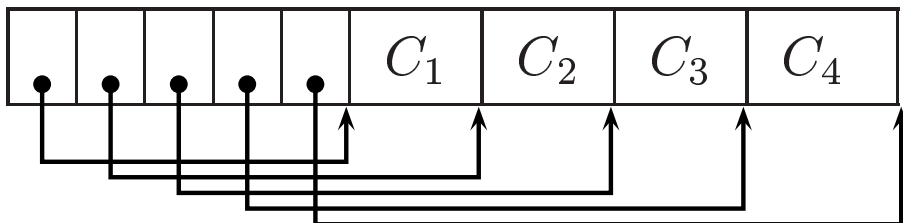
enregistrement (sans entête)



$C_i = i$ ème champ

### Solution 2

enregistrement (sans entête)



$C_i = i$ ème champ

- **Entête**

- taille de l'enregistrement

- pointeur vers schéma de la relation

- date de dernière mise à jour

- ...

- **gestion des valeurs nulles**

- ne rien stocker → identification champ

- stocker valeur spéciale → perte d'espace

- “masque” en entête : 1100 → les 2 derniers champs sont des valeurs nulles

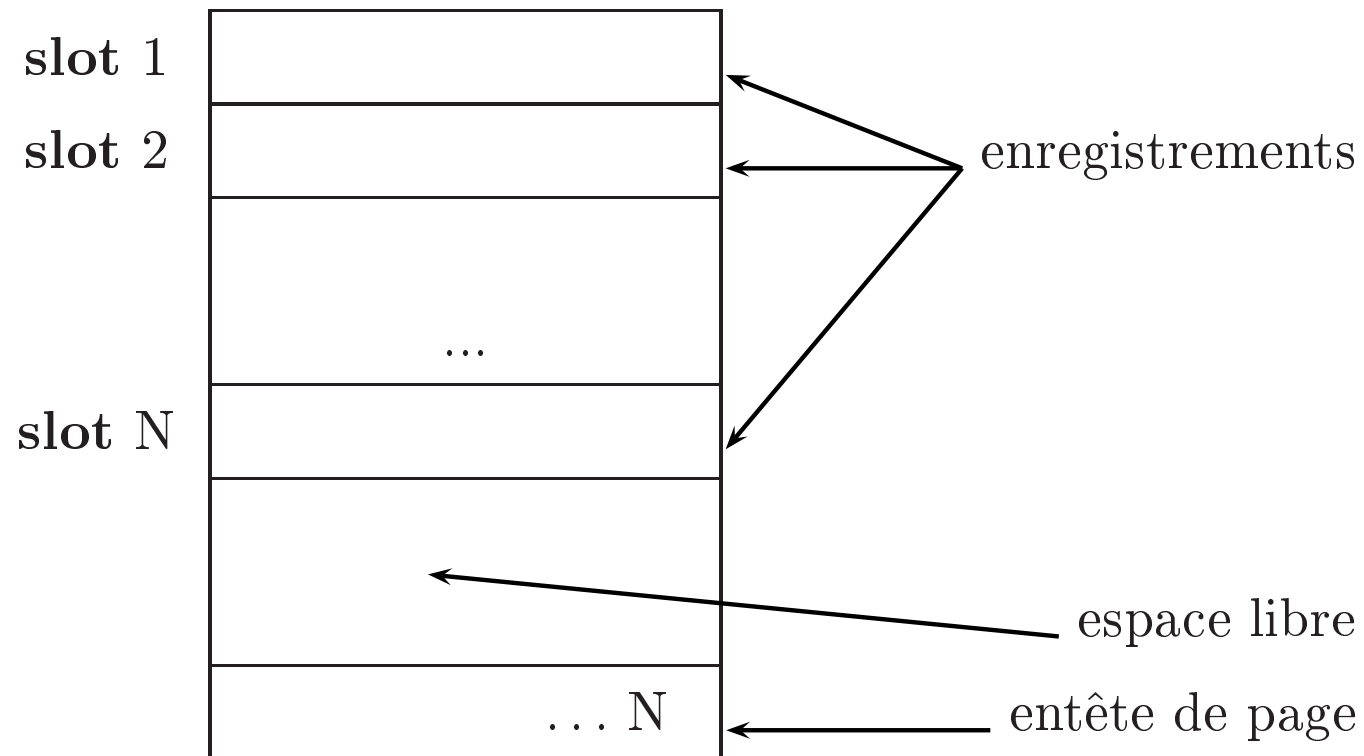
## Format page

---

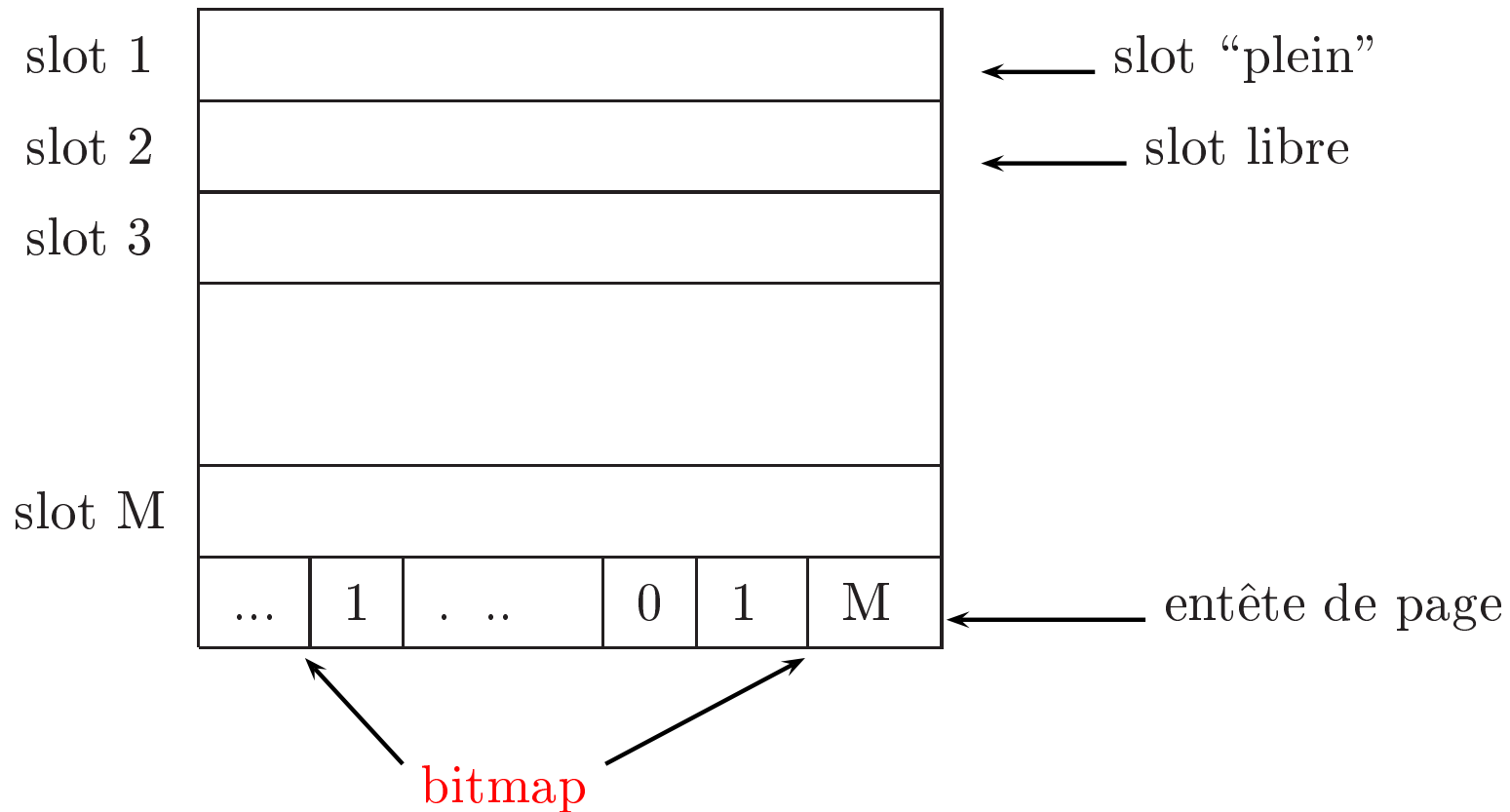
- relation = collection d'enregistrements
- page = collection d'emplacements
- adresse emplacement = (ident page , numéro emplacement)
- ident enregistrement (**rid**) = adresse emplacement stockage

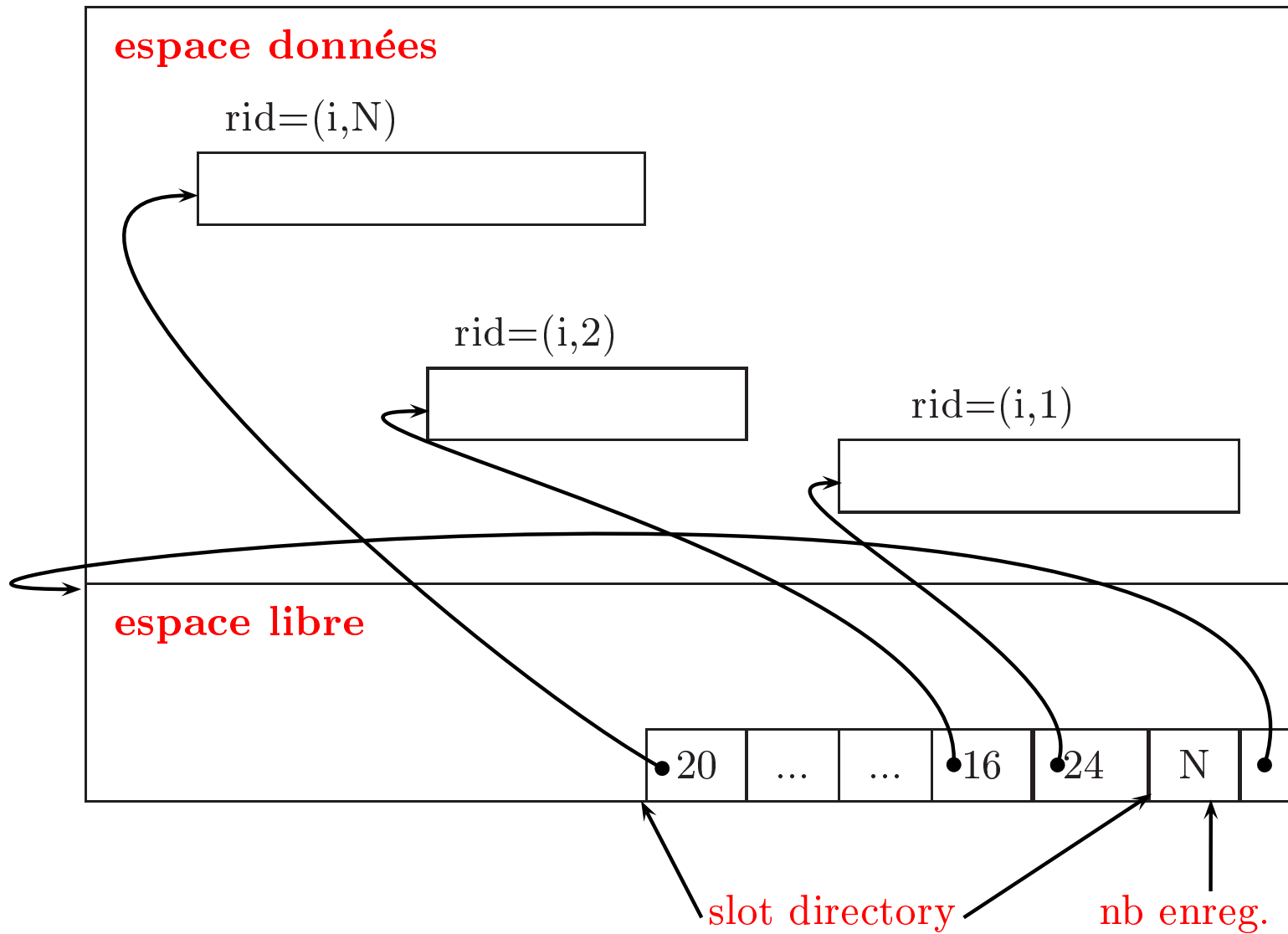
- 
- **insertion** : localisation emplacement libre
  - **suppression** : compactage

- Taille fixe → découpage en emplacements



- Taille fixe → découpage en emplacements





## Fichier

---

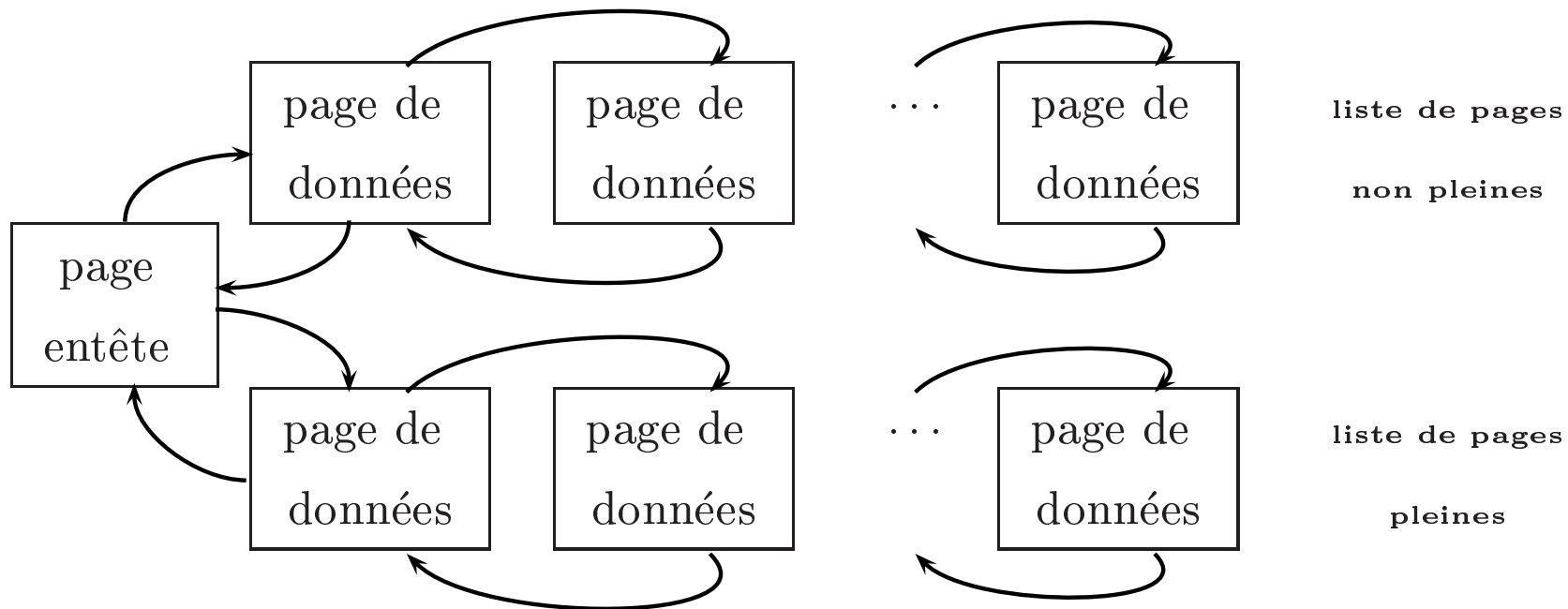
- **retrouver un enregistrement** =  
retrouver la page où est l'enregistrement  
retrouver l'enregistrement dans la page
- **retrouver un enregistrement à l'aide d'une condition**  
exemple :  $\text{age} \geq 35\text{ans}$
- **organisation des pages / structure du fichier**
  - tas (sans structure)
  - fichier trié sur une clé (de tri)
  - indexé sur une clé (d'index)

clé (tri-index) = liste d'attributs

clé (tri-index)  $\neq$  clé de relation

## Fichier

---



**Tas géré par une liste doublement chaînée**

## Fichier

---




répertoire  
pages-entête

Données Page 1
-------------------

Données Page 2
-------------------

Données Page N
-------------------

pages de  
données

Fichier géré par un répertoire de pages

## Catalogue / Dictionnaire

---

- Stockage - Interrogation - Modification des Méta-données

schéma logique : relations, attributs

niveau logique

schéma physique : index

niveau physique

statistique

optimisation

nbre de n-uplets d'une relation

nbre de valeurs distinctes

+ petite valeur, + grande valeur d'une clé d'index

- Catalogue = ensemble de RELATIONS

## Catalogue / Dictionnaire

---

- pour chaque relation :
  - nom de la relation
  - nom et type de chaque attribut
  - nom de chaque index
  - nom du fichier
  - structure du fichier (tas)
  - contraintes d'intégrité
- pour chaque index :
  - nom de l'index
  - structure de l'index
  - clé de recherche
- pour chaque vue :
  - nom de la vue
  - définition (SQL) de la vue

## Catalogue / Dictionnaire

---

nom-attribut	nom-relation	type	position
nom-attribut	cat-attribut	string	1
nom-relation	cat-attribut	string	2
type	cat-attribut	string	3
position	cat-attribut	integer	4
sid	Etudiant	string	1
nom	Etudiant	string	2
login	Etudiant	string	3
age	Etudiant	integer	4
fid	Professeur	string	1
fname	Professeur	string	2
salaire	Professeur	real	3
...	...	...	...

relation ATTRIBUT-CAT du Catalogue