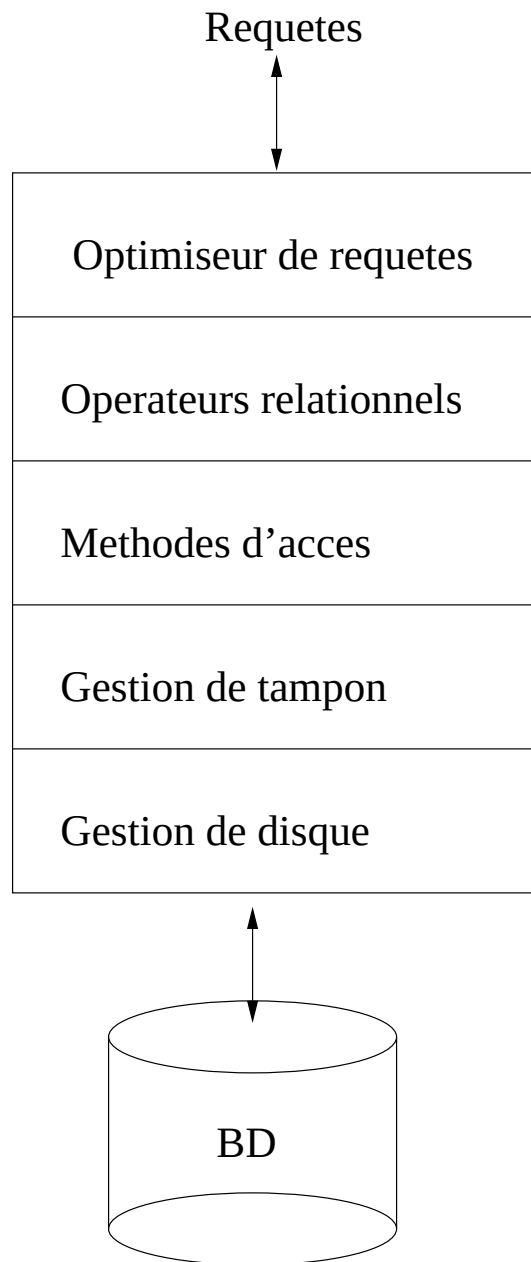


Structure fonctionnelle d'un SGBD



Fichiers et Disques

- **Lecture** : Transfert des données du disque vers la mémoire centrale.
- **Ecriture** : Transfert de la RAM vers le disque.
- Les 2 opérations sont relativement coûteuses par rapport à d'autres opérations qui sont effectuées en RAM.

Pourquoi ne pas tout garder en RAM ?

- La RAM est plus chère que les disques
- Volatilité de la RAM (Prix : 1998)

Hiérarchie de stockage

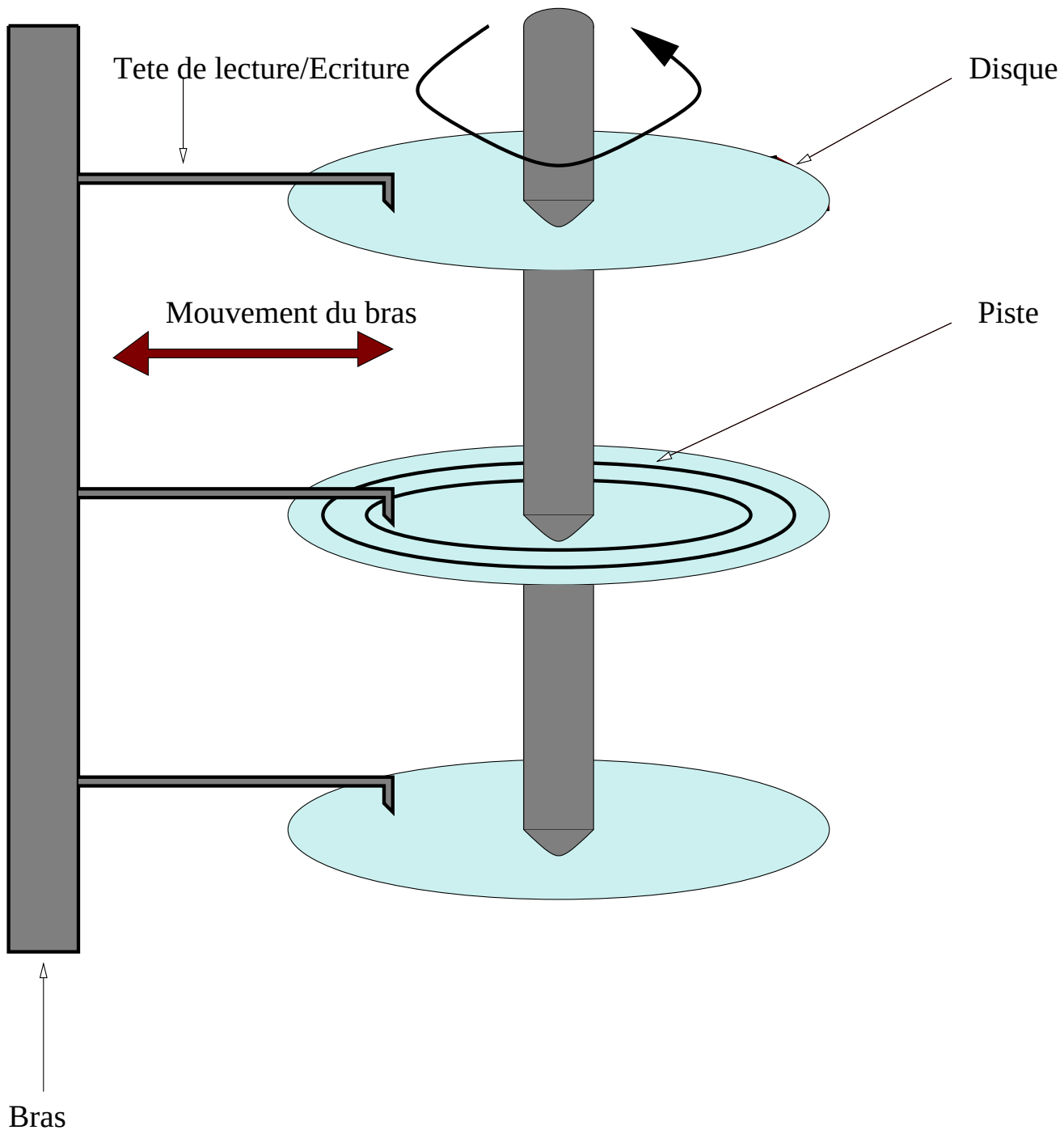
- RAM pour les données en cours d'utilisation,
- Disques pour toute la BD,
- Bandes magnétiques pour les archives.

Les disques

L'avantage par rapport aux bandes : Accès direct vs accès séquentiel.

Les données sont stockées et accédées par *blocs* ou *pages* entières.

Contrairement aux RAM, le temps d'accès aux données sur les disques dépend de l'emplacement $\Rightarrow$ bien placer les données pour optimiser les transferts.



Les disques ont une vitesse de rotation (ex : 90tr/s)

Les pistes de même diamètre forment **un cylindre**

Une **seule** tête peut lire ou écrire

La taille d'un **bloc** est un multiple de la taille d'un **secteur**

Paramètres d'un disque

Temps d'accès à un bloc (TAB) :

- *Temps de positionnement (TP)* : Le temps moyen qu'il faut pour positionner la tête de L/E sur la bonne piste.
- *Latence de rotation (LR)* : Le temps moyen qu'il faut pour positionner la tête de L/E au bon endroit sur la piste.
- *Taux de transfert (TT)* : La quantité d'info transférée par unité de temps (du disque à la RAM ou inversement)

TP : Ça varie entre 1 et 10 msec

LR : Ça varie entre 1 et 10 msec

TT : Le débit de transfert est de l'ordre de 4KO par msec.

Pour optimiser les transferts, il faut optimiser **TP** et **LR**.

Exemples (Infos datant de 2000)

Seagate Hawk 2XL :

2,15Go. **LR** = 5.55msec. **TP** = 9msec. Passer d'une piste à une piste voisine = 1msec. Le temps de positionnement maximum = 22msec. 512 octets par secteur. 4569 cylindres. 4 disques double face. Le débit du transfert **TT** = 5Mo/sec.

IBM Deskstar 14GPX :

14.4Go. **TP** = 9.1msec, **LR** = 4.17msec. Passer d'une piste à la suivante prend 2.2msec. **TT** = 13Mo/sec

Placement des pages sur disque

Le concept du bloc “suivant”

1. Mettre les blocs (successifs) sur la même piste,
2. blocs sur le même cylindre, ensuite
3. blocs sur le cylindre adjacent

Les blocs d'un fichier doivent être stockés d'une manière *séquentielle* sur le disque (“concept de suivant”) pour minimiser **TP** et **LR**.

Lorsque l'on veut faire **un balayage séquentiel** d'un fichier, le fait de pouvoir préparer (pre-fetch) plusieurs pages à la fois représente un gain considérable.

L'arrangement des fichiers sur les disques est une tâche supportée par une couche basse du SGBD (en collaboration avec le système d'exploitation).

Elle communique avec la couche juste au dessus par des appels de pages.

Bloc “suivant” : Exemple

On veut lire un fichier en supposant que

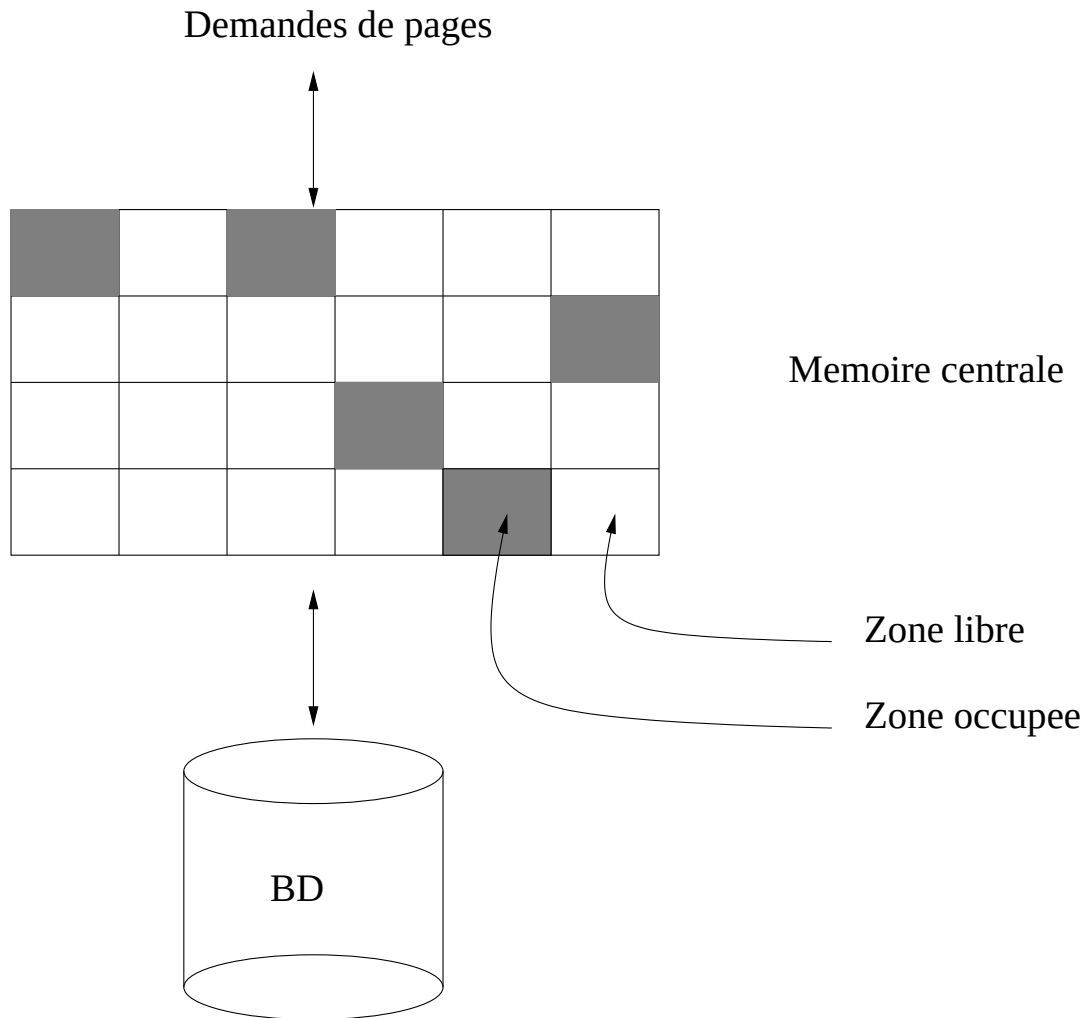
- la taille du fichier est de 1MO
- 1 bloc = 1 secteur = 512 octets
- une piste = 100 secteurs
- TP = 5ms , LR = 5ms et TP=4ko/ms

1. **Fichier contigu** : Positionner sur la bonne piste = TP, se positionner sur la bonne piste = LR puis lecture de 1MO.

$$\Rightarrow 5\text{ms} + 5\text{ms} + 256\text{ ms} = 266\text{ ms}$$

2. **Fichier dispersé** : Il y a 1MO $\Rightarrow$ 2048 blocs à lire. Pour chacun, il faut TP+LR+ TAB. $\text{TAB} = \frac{1}{8}\text{ms}$. $\Rightarrow 2048(5 + 5 + \frac{1}{8}) = 20736\text{ms}$

Gestionnaire de tampon



Pour que les données puissent être manipulées par le SGBD, il faut qu'elles soient en RAM

On a une table de $\langle zoneId, pageId \rangle$

A chaque page (en RAM) est associé un bit qui indique si elle a été modifiée

A chaque page (en RAM) est associé un compteur des transactions qui sont en train de l'utiliser

Quand une page est demandée ...

1. Si la page n'est pas disponible, le gestionnaire de tampon (GT) émet une demande au gestionnaire de disque :
 - Choisir une page (zone) du tampon à remplacer
 - Si la page à remplacer a été modifiée alors la réécrire sur le disque
 - Lire la page demandée, i.e la transférer du disque vers la zone libérée
2. Incrémenter son compteur et retourner son adresse

Dans tous les cas, la zone choisie pour être remplacée doit avoir son compteur égal à 0, i.e aucune transaction n'est en train de l'utiliser.

Choix de remplacement

LRU (Least Recently Used) : La moins récemment utilisée

MRU (Most Recently Used) : La plus récemment utilisée

FIFO (First In First Out), **LIFO** (Last In First Out)...

Chacune des stratégies peut être efficace dans une situation mais pas dans une autre. Exemple : Lire plusieurs fois un fichier de 10 pages avec un tampon de 9 zones. Dans ce cas, MRU est meilleur que LRU

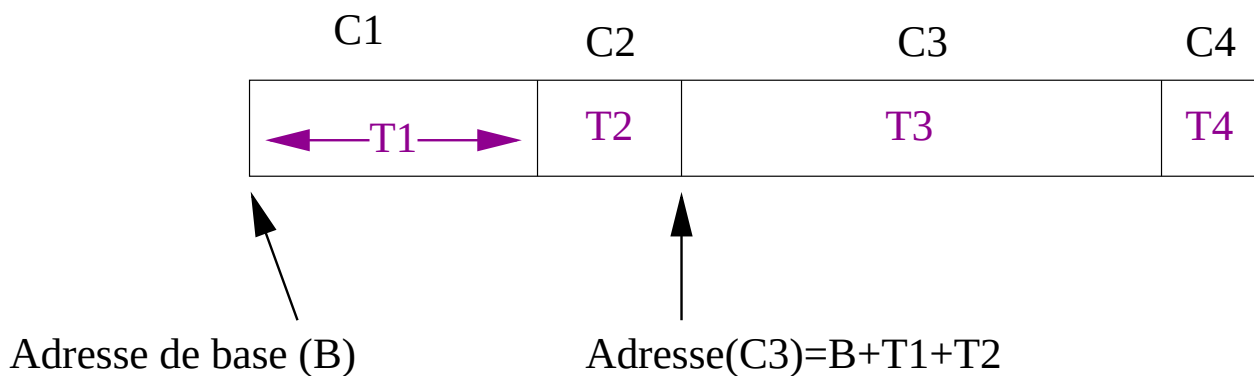
Formats d'enregistrements

Notons au préalable que le SGBD gère un **catalogue** qui permet de stocker des informations sur les fichiers (ex : le nom des champs, leurs tailles, ...)

Taille fixe versus taille variable

Taille fixe :

Dans ce cas, étant donnée l'adresse d'un enregistrement, on peut accéder directement à un champ particulier



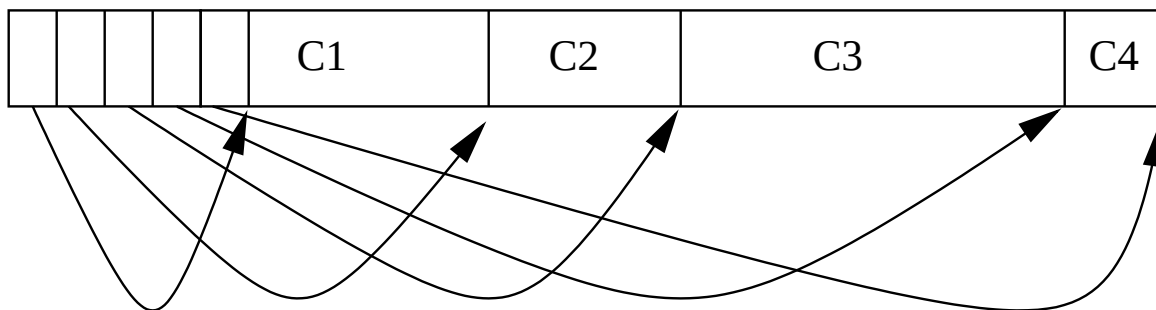
B est l'adresse de l'enregistrement

Formats d'enregistrements

Taille variable : (le nombre de champs est fixe)

Deux alternatives :

C1	\$	C2	\$	C3	\$	C4	\$
----	----	----	----	----	----	----	----

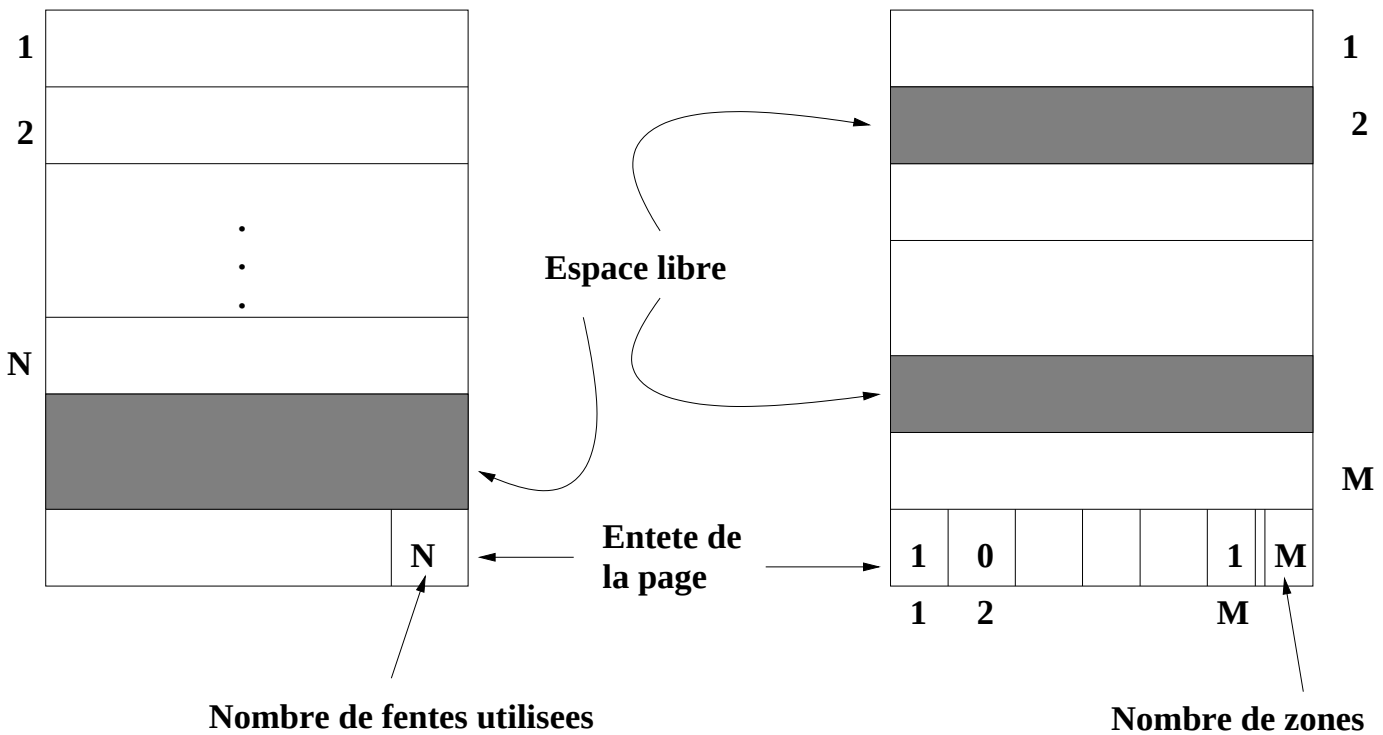


1. les champs sont séparés par un caractère spécial
2. l'entête de l'enregistrement indique le début de chaque champs

La 2ème alternative est plus intéressante surtout pour la gestion des valeurs **Nulls** : Si un champ est NULL alors les pointeurs de début et de fin associées à ce champ sont égaux.

Enregistrements de taille fixe

Deux variantes



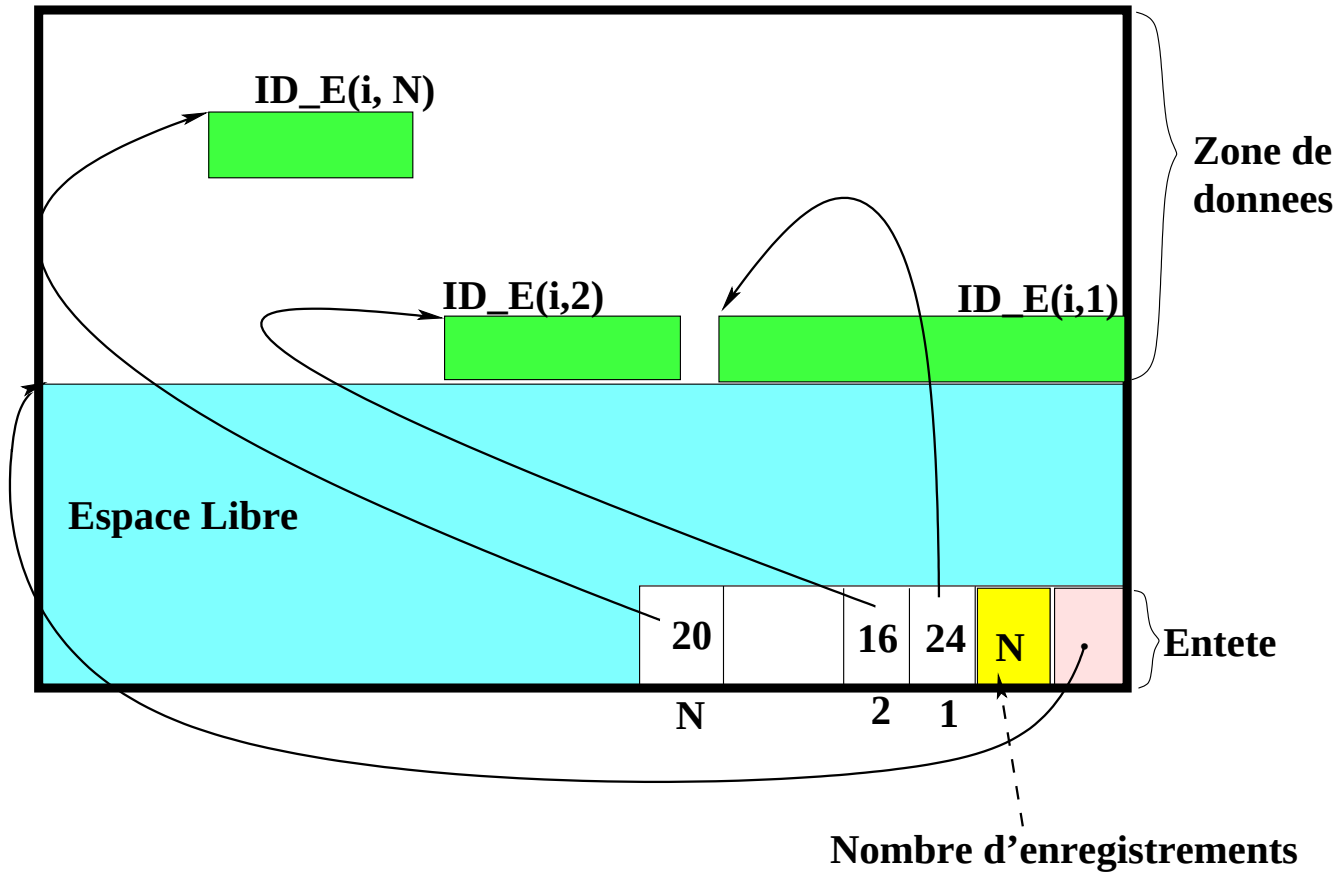
La page est composée de *fentes* (slots) chacune pouvant accueillir un enregistrement

$ID_Enregistrement = \langle Id_page, Num_fente \rangle$

Problème avec la 1ère alternative : l'IDE change suite aux suppressions.

IBM DB2, Informix, SQL Server, Oracle 8 utilisent la 2ème alternative

Enregistrements de taille variable



L'espace libre est contigu.

L'entête contient

- Un pointeur vers le début de la zone libre
- Le nombre d'enregistrements dans la page
- Une liste de pointeurs vers le début de chaque enregistrement ainsi que sa taille

Suite à une suppression, l'enregistrement peut bouger mais son ID_E n'est pas modifié.

Fichier d'enregistrements

- la notion de page est utilisée pour spécifier les E/S mais les utilisateurs ne manipulent pas directement ces pages
- **Fichier** : Collection de pages contenant chacune un ensemble d'enregistrements
- Possibilité de :
 - Insérer, Supprimer, modifier des enregistrements
 - Lire un enregistrement en spécifiant son ID_E
 - scanner tous les enregistrements

Fichiers non triés (tas)

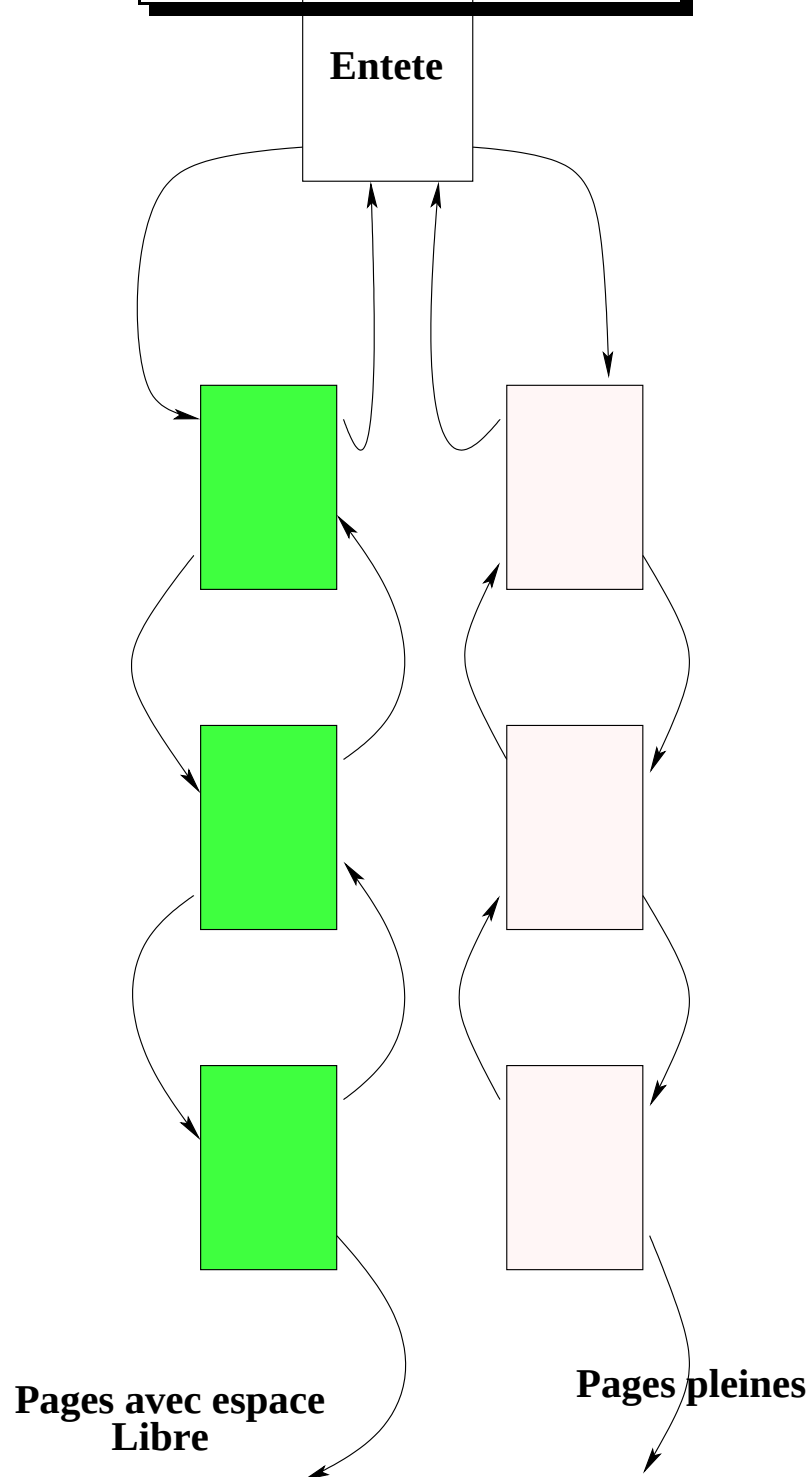
La structure de fichier la plus simple. La taille varie au gré des insertions et suppressions des enregistrements.

Les pages (blocs) sont allouées/désallouées en conséquence.

Pour pouvoir gérer les opérations sur les enregistrements, nous devons garder :

- les références des pages du fichier,
- l'espace libre dans chaque page,
- l'emplacement des enregistrements dans chaque page.

Fichiers tas : listes

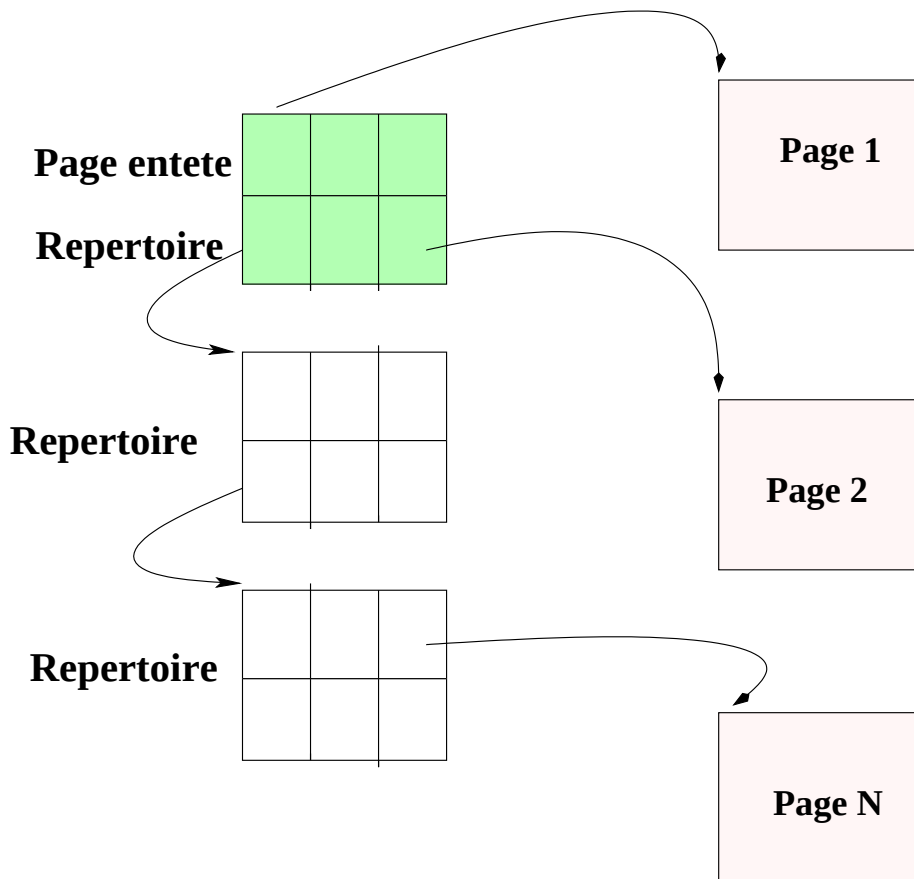


L'entête du fichier doit être stockée quelque part

Chaque page contient 2 pointeurs

Inconvénient : La liste des pages pleines risque d'être vide

Fichiers tas : Répertoire



Un répertoire est une page contenant des pointeurs vers des pages ainsi que l'espace libre qu'elles contiennent

Les index

Les fichiers tas permettent de retrouver un enregistrement en :

- spécifiant le *IdE*, ou
- balayant tout le fichier.

Souvent, on veut les enregistrements selon la valeur de certains champs. Pour ce type de recherche, les fichiers tas ne sont pas adéquats.

Les **index** sont des fichiers “annexes” permettant de répondre efficacement aux recherches basées sur des *valeurs*.

Le catalogue du système

Pour chaque index, sa structure (ex : arbre B+) ainsi que sa clé.

Pour chaque relation, (i) son nom, (ii) le fichier associé, (ii) la structure de fichier (indexé ou tas), (iii) le type et le nom des attributs, (iv) le nom de(s) l'index, (v) les contraintes d'intégrité.

Pour chaque vue, (i) son nom et (ii) sa définition.

Des statistiques, les droits d'accès, la taille de la zone tampon, ...

Att_Cat(att_name, rel_name, type, position)

att_name	rel_name	type	position
att_name	att_cat	char(20)	1
rel_name	att_cat	char(20)	2
type	att_cat	char(20)	3
position	att_cat	INTEGER	4
NumE	Etudiants	INTEGER	1
Nom	Etudiants	varchar(20)	2
...	dots	...	...

conclusion

Le gestionnaire de tampon *charge* les pages dans la RAM

- Les pages restent dans le tampon jusqu'à ce qu'il n'y ait aucun processus qui ne les utilise.
- Elles sont *réécrites* sur le disque après qu'elles soient libérées et quand la zone qu'elles occupent est choisie pour un remplacement.
- Le choix de la zone à libérer est dictée par différentes techniques possibles (LRU, ...)

Les enregistrements à taille variable permettent de bien gérer les valeurs nulles.

Le gestionnaire de fichier est chargé de la gestion des pages pour chaque fichier.

Les index permettent de faire des recherches basés sur des valeurs.

Le catalogue contient la description de la structure physique des relations, attributs, fichiers, index, ...