Bases de Données & Optimisation : Sotckage, pagination et fichiers

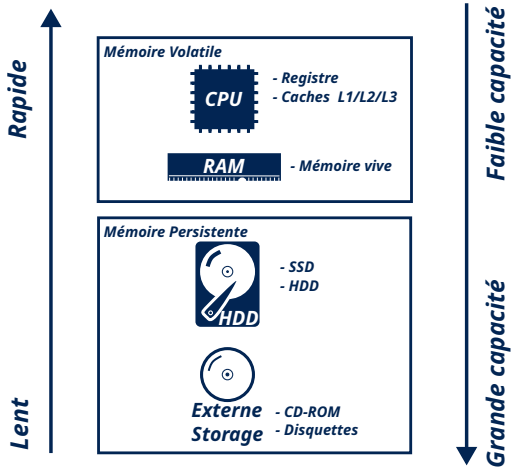
Thomas Gerald

September 23, 2025

Laboratoire Interdisciplinaire des Sciences du Numérique – LISN, CNRS
thomas.gerald@lisn.upsaclay.fr

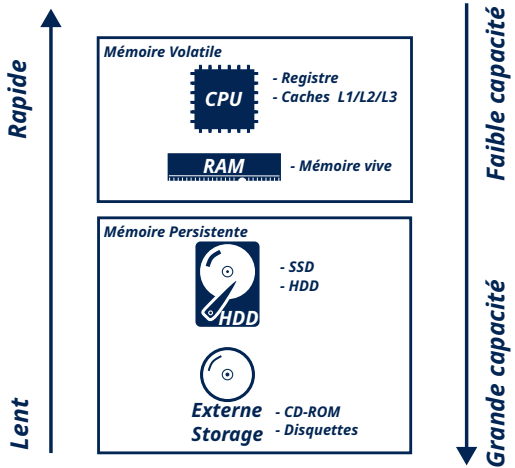
Les capacité de la mémoire

1. Mémoire du processeur (cache), quelques *Ko* à plusieurs dizaine de *Mo*
2. Mémoire vive, 4*Go* à 512*Go*
3. HDD/SSD, quelques 500*Go* à plusieurs *To*
4. CD-ROM/DVD/Blu-ray, 700*Mo* à 100*Go*



Les capacité de la mémoire

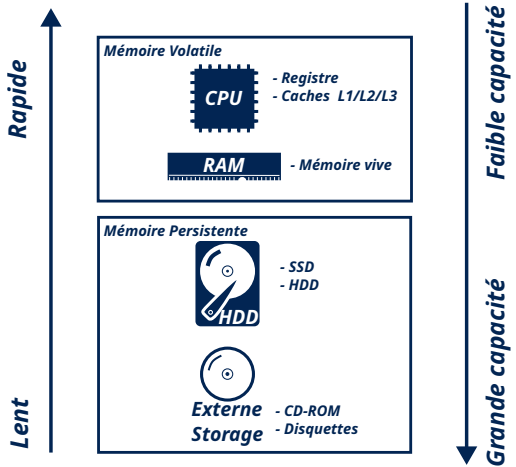
1. Mémoire du processeur (cache), quelques *Ko* à plusieurs dizaine de *Mo*
2. Mémoire vive, 4*Go* à 512*Go*
3. HDD/SSD, quelques 500*Go* à plusieurs *To*
4. CD-ROM/DVD/Blu-ray, 700*Mo* à 100*Go*



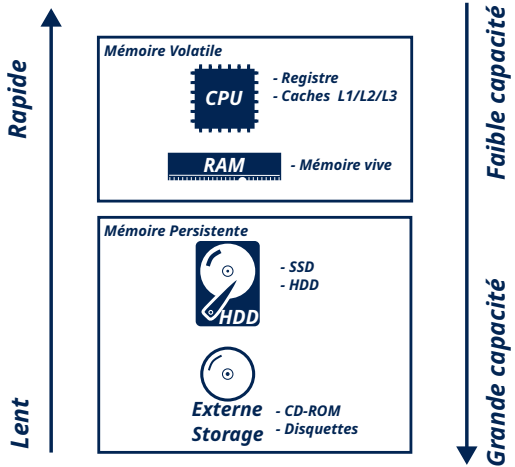
Besoins de grande capacité de stockage persistant → HDD/SSD

Hiérarchie des temps d'accès (approximatif)

1. Mémoire du processeur (cache), $\approx 370Go/s$ à $2300Go/s$
2. Mémoire vive, $20Go/s$ à $60Go/s$ (débit thèorique $largeur\_bus \times frequence$)
3. HDD/SSD, $100Mo/s$ à $8Go/s$ (Différence importante si données séquentielles ou non)
4. CD-ROM, quelques Mo/s

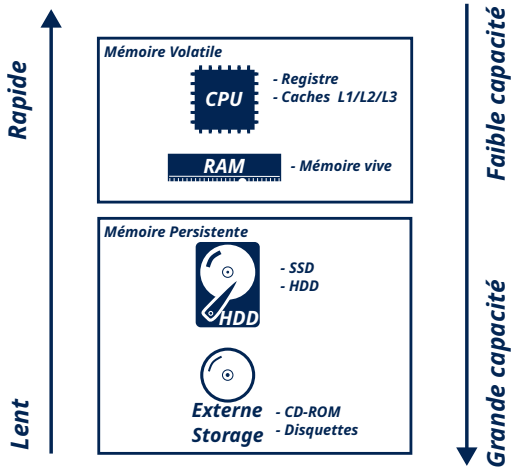


Optimisation: objectifs



Sur les supports persistants ?

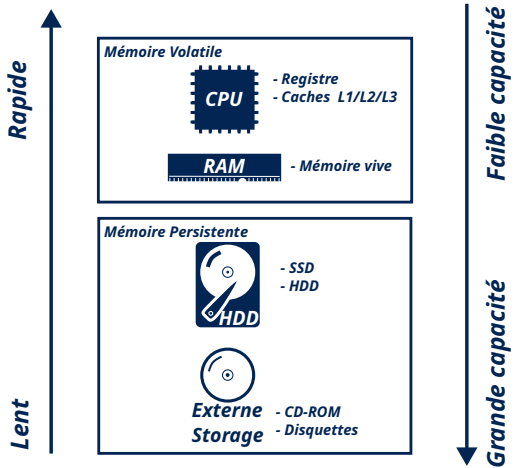
Optimisation: objectifs



Sur les supports persistants ?

- Minimiser le temps/nombre d'écritures
- Minimiser le temps/nombre de lectures

Optimisation: objectifs

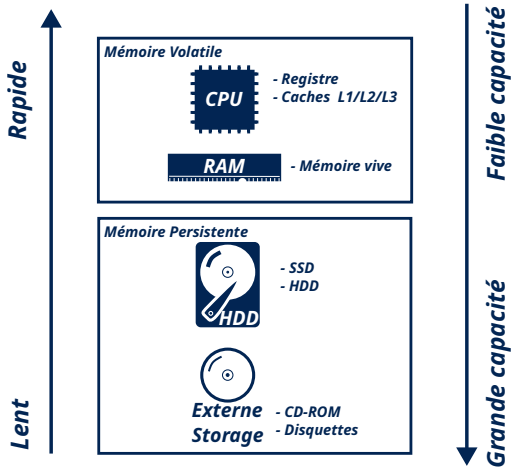


Sur les supports persistants ?

- Minimiser le temps/nombre d'écritures
- Minimiser le temps/nombre de lectures

Solutions ?

Optimisation: objectifs



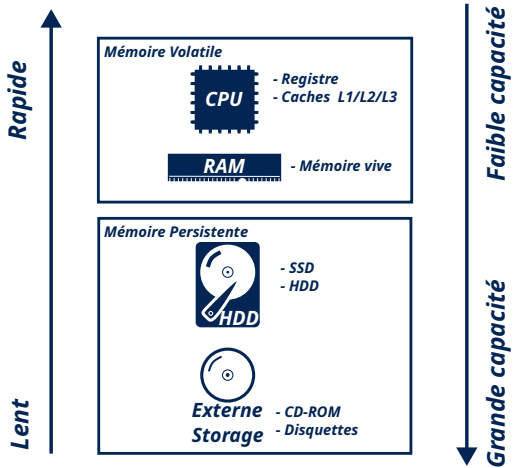
Sur les supports persistants ?

- Minimiser le temps/nombre d'écritures
- Minimiser le temps/nombre de lectures

Solutions ?

- Organisation de l'espace mémoire "fichiers" sur le disque

Optimisation: objectifs



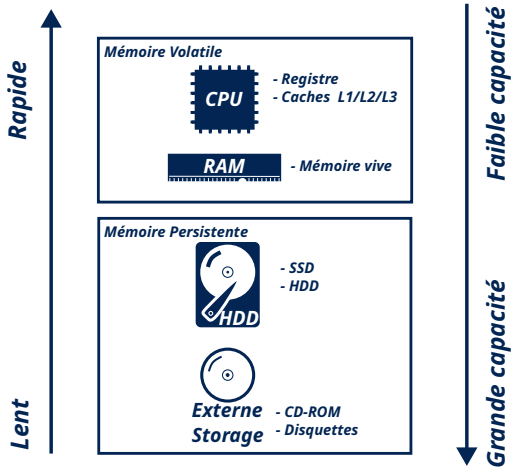
Sur les supports persistants ?

- Minimiser le temps/nombre d'écritures
- Minimiser le temps/nombre de lectures

Solutions ?

- Organisation de l'espace mémoire "fichiers" sur le disque
- Gestion de la mémoire vive (mettre en cache)

Optimisation: objectifs



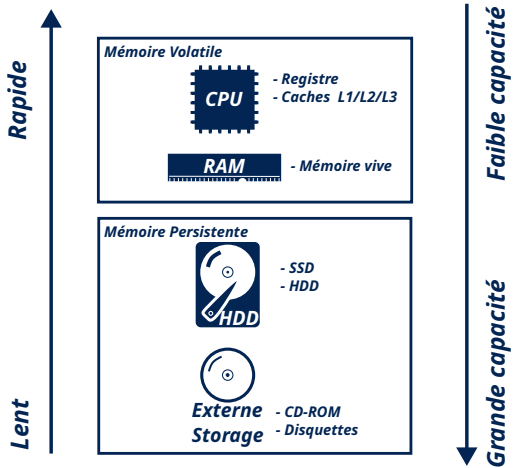
Sur les supports persistants ?

- Minimiser le temps/nombre d'écritures
- Minimiser le temps/nombre de lectures

Solutions ?

- Organisation de l'espace mémoire "fichiers" sur le disque
- Gestion de la mémoire vive (mettre en cache)
- Indexation des informations (retrouver rapidement des enregistrements)

Optimisation: objectifs



Sur les supports persistants ?

- Minimiser le temps/nombre d'écritures
- Minimiser le temps/nombre de lectures

Solutions ?

- Organisation de l'espace mémoire "fichiers" sur le disque
- Gestion de la mémoire vive (mettre en cache)
- Indexation des informations (retrouver rapidement des enregistrements)
- Optimisation des algorithmes

Gestionnaire de mémoire

Gestion de la mémoire

- Les relations (et d'autres informations) sont stockées sur un support persistant (mémoire secondaire)
- Pour exécuter une requête on travail sur les données en RAM (mémoire primaire)

Gestion de la mémoire

- Les relations (et d'autres informations) sont stockées sur un support persistant (mémoire secondaire)
- Pour exécuter une requête on travail sur les données en RAM (mémoire primaire)

Gestion de la mémoire

- Les relations (et d'autres informations) sont stockées sur un support persistant (mémoire secondaire)
- Pour exécuter une requête on travail sur les données en RAM (mémoire primaire)

Minimiser les transferts :

Gestion de la mémoire

- Les relations (et d'autres informations) sont stockées sur un support persistant (mémoire secondaire)
- Pour exécuter une requête on travail sur les données en RAM (mémoire primaire)

Minimiser les transferts :

- Optimiser les lecture/écriture sur le disque (organisation sur le disque)
  $\rightarrow$ Gestionnaire de disque/disc manager

Gestion de la mémoire

- Les relations (et d'autres informations) sont stockées sur un support persistant (mémoire secondaire)
- Pour exécuter une requête on travail sur les données en RAM (mémoire primaire)

Minimiser les transferts :

- Optimiser les lecture/écriture sur le disque (organisation sur le disque)
  → Gestionnaire de disque/disc manager
- Optimiser ce que l'on garde dans le cache
  → Gestionnaire de cache/buffer manager

Le gestionnaire de disque

Stockage des données : Les données persistentes

- Organisation de l'espace mémoire "fichiers" sur le disque ←
- Gestion de la mémoire vive (éviter de lire sur le disque)
- Indexation des informations (retrouver rapidement des enregistrements)
- Optimisation des plans d'executions

Gestionnaire de disque

→ Organiser les données sen mémoire persistantes
→ Lire les données sur le disque
→ Écrire les données sur le disque

Comprendre les accès aux disques : Hard Disc Drive (HDD)

Les disques à plateaux

- Plusieurs **plateaux** ou disques (double face) contenant les données
- Une **tête de lecture** par plateau (dupliquées sur les deux faces)
- Plusieurs **pistes** pour chaque disque
- Des **secteurs** de données de taille fixe

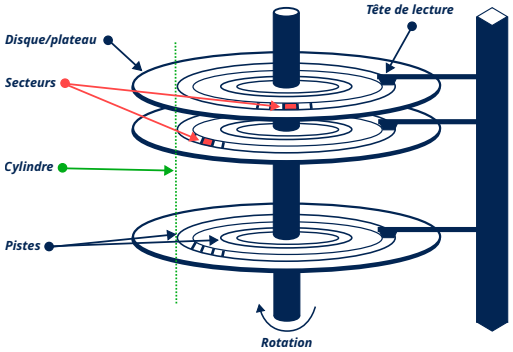


Figure 1: Schéma d'un disque à plateau

→ Données sont stockées sur des blocs/pages (groupement d'un ou plusieurs secteurs)

Comprendre les accès aux disques : Hard Disc Drive (HDD)

- **Données stockées** sur de blocs (ou des pages)
  - C'est l'unité de lecture et d'écriture
  - Même si on veut lire seulement 4 octets, tout le bloc est chargé
- Les pistes de même diamètre $\rightarrow$ **cylindre**
  - Lecture des données d'un cylindre sans déplacer les têtes !!!

Comprendre les accès aux disques : Hard Disc Drive (HDD)

Disques à plateaux : Le temps d'accès

1. Positionner la tête de lecture sur la piste ou le cylindre (**recherche**) (opération couteuse)
2. Effectuer une rotation pour accéder au début du secteur de la page (**rotation**)
3. Effectuer une rotation de lecture jusqu'à la fin de la page et renvoyer les données en mémoire (**transfert**)

→ Le **temps d'accès** est $T_{recherche} + T_{rotation} + T_{transfert}$

Temps de latence :

le temps de latence est le temps pour accéder au début du bloc ($T_{recherche} + T_{rotation}$)

- Un fois qu'on a payé le temps de latence → la lecture du bloc suivant est "rapide" (temps de transfer seulement)

Comprendre les accès aux disques : Hard Disc Drive (HDD)

Les disques à plateaux : Lire une page

Considérons 5 pages combien d'opérations pour les lire ?

1. Les pages sont écrites sur 5 blocs différents
2. Les pages sont écrites sur une même piste
3. Les pages sont écrites sur des blocs contigus
4. Les pages sont stockées sur un même cylindre sur des secteurs parallèles

On considère :

- T le temps de déplacement d'une tête
- R le temps pour une rotation
- L le temps d'une lecture (rotation aussi)

Comprendre les accès aux disques : Hard Disc Drive (HDD)

Les disques à plateaux : Lire une page

Considérons 5 pages combien d'opérations pour les lire ?

1. Les pages sont écrites sur 5 blocs différents
2. Les pages sont écrites sur une même piste
3. Les pages sont écrites sur des blocs contigus
4. Les pages sont stockées sur un même cylindre sur des secteurs parallèles

On considère :

- T le temps de déplacement d'une tête
- R le temps pour une rotation
- L le temps d'une lecture (rotation aussi)

Réponse :

1. Déplacement de la tête sur la piste de P_1
2. Rotation du disque jusqu'au bloc de P_1
3. Lire P_1
4. Déplacement de la tête sur la piste de P_2
5. Rotation du disque jusqu'au bloc de P_2
6. ...

$\rightarrow T \times 5 + R \times 5 + L \times 5$

Comprendre les accès aux disques : Hard Disc Drive (HDD)

Les disques à plateaux : Lire une page

Considérons 5 pages combien d'opérations pour les lire ?

1. Les pages sont écrites sur 5 blocs différents
2. Les pages sont écrites sur une même piste
3. Les pages sont écrites sur des blocs contigus
4. Les pages sont stockées sur un même cylindre sur des secteurs parallèles

On considère :

- T le déplacement d'une tête
- R le temps pour une rotation
- L le temps d'une lecture (rotation aussi)

Comprendre les accès aux disques : Hard Disc Drive (HDD)

Les disques à plateaux : Lire une page

Considérons 5 pages combien d'opérations pour les lire ?

1. Les pages sont écrites sur 5 blocs différents
2. Les pages sont écrites sur une même piste
3. Les pages sont écrites sur des blocs contigus
4. Les pages sont stockées sur un même cylindre sur des secteurs parallèles

On considère :

- T le déplacement d'une tête
- R le temps pour une rotation
- L le temps d'une lecture (rotation aussi)

Réponse :

1. Déplacement de la tête sur la piste de P_1
2. Rotation du disque jusqu'au bloc de P_1
3. Lire P_1
4. Rotation du disque jusqu'au bloc de P_2
5. Lire P_2
6. ...

$\rightarrow T + R \times 5 + L \times 5$

Comprendre les accès aux disques : Hard Disc Drive (HDD)

Les disques à plateaux : Lire une page

Considérons 5 pages combien d'opérations pour les lire ?

1. Les pages sont écrites sur 5 blocs différents
2. Les pages sont écrites sur une même piste
3. Les pages sont écrites sur des blocs contigus
4. Les pages sont stockées sur un même cylindre sur des secteurs parallèles

On considère :

- T le déplacement d'une tête
- R le temps pour une rotation
- L le temps d'une lecture (rotation aussi)

Comprendre les accès aux disques : Hard Disc Drive (HDD)

Les disques à plateaux : Lire une page

Considérons 5 pages combien d'opérations pour les lire ?

1. Les pages sont écrites sur 5 blocs différents
2. Les pages sont écrites sur une même piste
3. Les pages sont écrites sur des blocs contigus
4. Les pages sont stockées sur un même cylindre sur des secteurs parallèles

On considère :

- T le déplacement d'une tête
- R le temps pour une rotation
- L le temps d'une lecture (rotation aussi)

Réponse :

1. Déplacement de la tête sur la piste de P_1
2. Rotation du disque jusqu'au bloc de P_1
3. Lire P_1
4. Lire P_2
5. Lire P_3
6. ...

$\rightarrow T + R + 5 \times L$

Les disques à plateaux : Lire une page

Considérons 5 pages combien d'opérations pour les lire ?

1. Les pages sont écrites sur 5 blocs différents
2. Les pages sont écrites sur une même piste
3. Les pages sont écrites sur des blocs contigus
4. Les pages sont stockées sur un même cylindre sur des secteurs parallèles

- T le déplacement d'une tête
- R le temps pour une rotation
- L le temps d'une lecture (rotation aussi)

1. Déplacement de la tête sur la piste de P_1
2. Rotation du disque jusqu'au bloc de P_1
3. Lire P_1, P_2, P_3, P_4, P_5

$\rightarrow T + R + L$

Gestionnaire de cache

Lire les données

Accès disque

- Trop coûteux !!!
- Les mêmes pages/enregistrements sont demandés plusieurs fois !

- Organisation de l'espace mémoire "fichiers" sur le disque
- **Gestion de la mémoire vive (éviter de lire sur le disque)** ←
- Indexation des informations (retrouver rapidement des enregistrements)
- Optimisation des plans d'executions

Le cache (ou buffer)

Le cache, objectifs :

Limiter les accès aux disques, charger en RAM les données

Le cache (ou buffer)

Le cache, objectifs :

Limiter les accès aux disques, charger en RAM les données

- Transférer des pages en mémoire vive

Le cache, objectifs :

Limiter les accès aux disques, charger en RAM les données

- Transférer des pages en mémoire vive
- Garder les pages dont les accès seront ou sont requis par les processus
- Supprimer/remplacer les pages inutilisées

→ Gestionnaire de cache

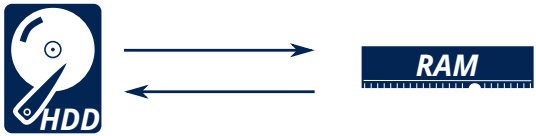


Figure 2: Stocker en mémoire vive les données d'accès fréquent

Le cache (ou buffer)

Le cache, objectifs :

Limiter les accès aux disques, charger en RAM les données

Le cache (ou buffer)

Le cache, objectifs :

Limiter les accès aux disques, charger en RAM les données

- Transférer des pages en mémoire vive

Le cache (ou buffer)

Le cache, objectifs :

Limiter les accès aux disques, charger en RAM les données

- Transférer des pages en mémoire vive
- Garder les pages dont les accès seront ou sont requis par les processus
- Supprimer/remplacer les pages inutilisées

→ **Gestionnaire de cache**

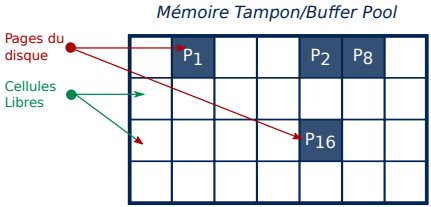


Figure 3: Le cache peut-être représenté comme un tableau de cellules, chaque cellule contient une page (ou vide)

Le cache (ou buffer)

Le cache, informations supplémentaires

- Stockage du nombre d'utilisateurs/processus pour une page (`pin_count`)
- Stockage d'un bit (`dirty bit`) informant de la modification de la page (l'écriture avant d'être fait sur le stockage dur est éfféctué dans le cache)

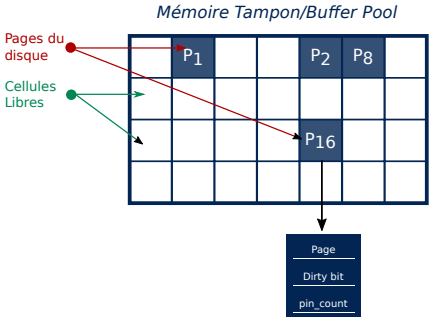
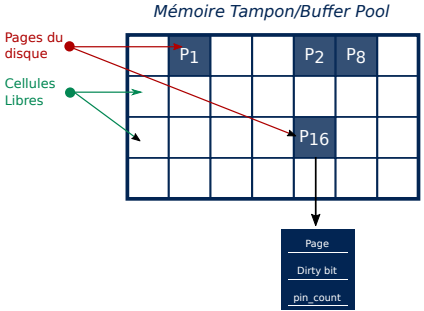


Figure 4: Le cache peut-être représenté comme un tableau de cellules, chaque cellule contient une page (ou vide)

Demande d'une page P au gestionnaire de mémoire

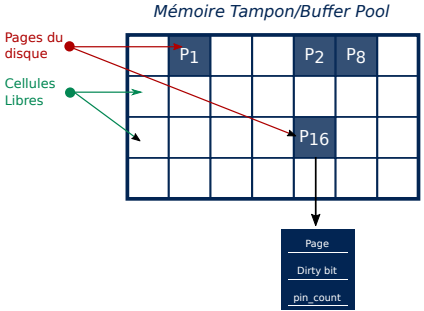
1. Vérification de la présence de la page dans le buffer
2. Si la page demandée n'est pas présente dans le cache
   - 2.1 **Sélection d'une cellule C_i (remplacement ou vide)**
   - 2.2 Si C_i n'est pas vide et `dirty_bit` vaut 1 → écriture de la page sur le disque
   - 2.3 Transfert de la page demandé depuis le disque vers le cache (remplacement)
   - 2.4 Initialiser `dirty_bit` et `pin_count` à 0 pour la page P
3. Incrémenter le compteur (`pin_count`)



Le cache (ou buffer)

Libérer une Page P

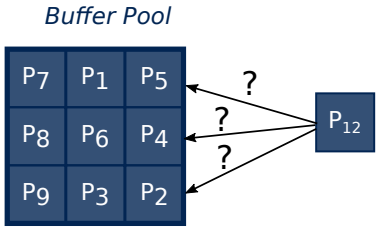
1. Si P à été modifié par le processus appelant la libération
   1.1 `dirty_bit` $\leftarrow 1$
2. `pin_count` $\leftarrow$ `pin_count` -1



Remplacer des pages du cache

Comment choisir la page qui sera remplacée ?

- **FIFO** First in First out
- **LRU** (least recently used) Les pages les moins récemment utilisées
- **MRU** (most recently used) Les pages utilisées récemment

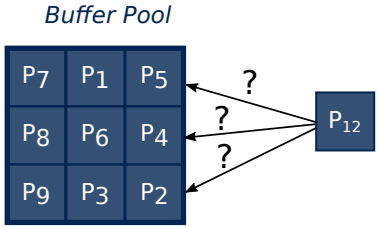


Maintient d'une liste des cellules libres ou de cellules contenant des pages dont le pin_count=0

Remplacer des pages du cache

Comment choisir la page qui sera remplacée ?

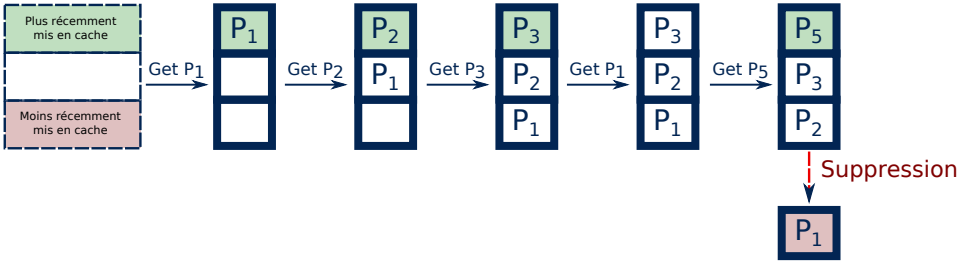
- **FIFO** First in First out
- **LRU** (least recently used) Les cellules les moins récemment utilisées
- **MRU** (most recently used) Les cellules utilisées récemment
- **Seconde chance**



Stratégie de remplacement : FIFO

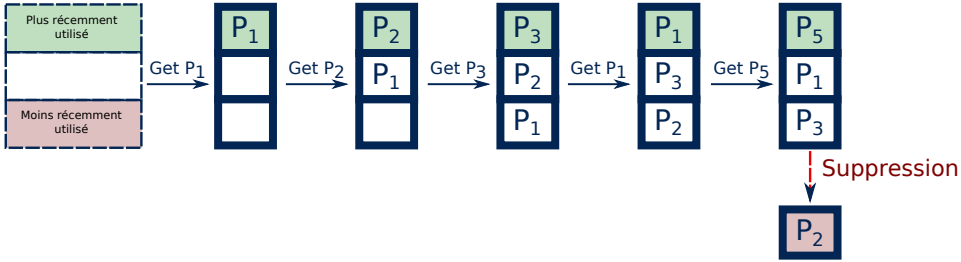
Premier arrivé, premier sorti :
Principe : Lorsque le cache est plein, on supprime l'élément le plus anciennement chargé.

- On maintiens une liste :
  - Les éléments de tête sont les plus anciens
  - Les éléments de queue sont les plus récents



Stratégie de remplacement : LRU

Least recently used : stockage
Principe : Supprimer les pages non utilisées récemments
Mettre à jour la liste + HashMap (implémentation différentes possible) ! Faire remonter en tête de liste à chaque fois qu'un élément est demandé



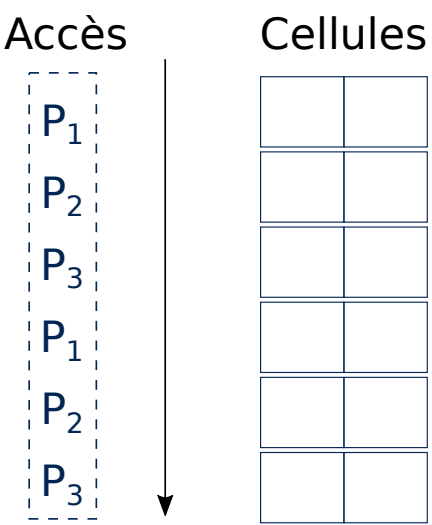
⚠ En pratique on va considèrer la page ne peut être enlevée du cache si le `pin_count` ≥ 1.

Stratégie de remplacement : Exemple LRU

Exercice
On dispose d'un cache d'une capacité de 4 pages vides (⚠ exemple jouet)

- Mesurer le nombre de transfert entre le disque et la RAM pour les opérations suivantes successives en utilisant la méthode LRU :
  1. Lecture de P_1, P_2, P_3 et P_4 (on libère dans le même ordre)
  2. Écriture sur P_1
  3. Lecture de P_5
  4. Lecture de P_2
  5. Lecture de P_1

  Pour ces opérations écrire la liste correspondante

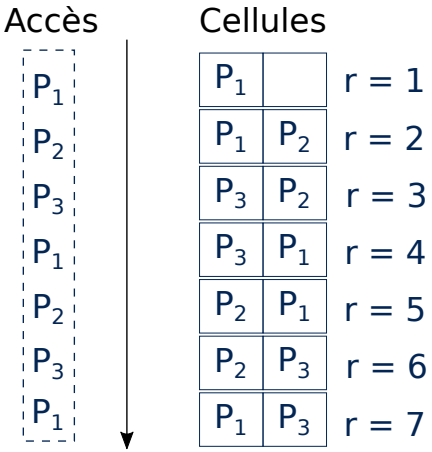


Sequential Flooding
Répéter une demande de suite de pages

```
Pour i dans A
    Pour j dans B
        lire la page j
```

- Quelles sont les conséquences pour les précédents algorithmes ?
- Quelle est le minimum d'accès possible ?

LRU



Sequential Flooding
Répeter une demande de suite de pages

```
Pour i dans A
    Pour j dans B
        lire la page j
```

- Quelles sont les conséquences pour les précédents algorithmes ?
- Quelle est le minimum d'accès possible ?

Stockage : Utilisation du Système d'exploitation ?

Le système d'exploitation

- Système de fichier, gestionnaire disque
- Gestionnaire de cache

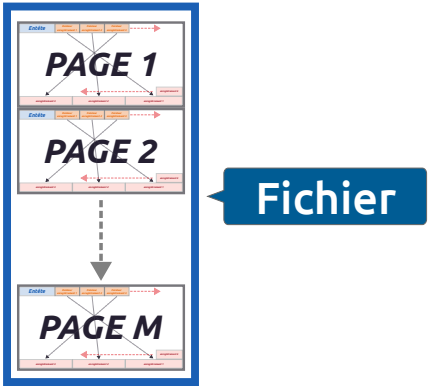
Pourquoi réimplémenter dans le SGBD le gestionnaire de mémoire ?

- Portabilité : fonctionnement sur différentes plateformes
- Très grands objets (par exemple sur plusieurs disques)
- Structurer les pages par rapport aux besoins de la base de données
- Gestion d'un buffer adapté aux données
- Gestion de la concurrence
- Gestion de la journalisation
- ...

Fichiers et pages

Organisation des relations

- Une relation est stockée dans un fichier
- Les fichiers contiennent des pages
- Il existe différentes organisations dans un fichier



Le record ID

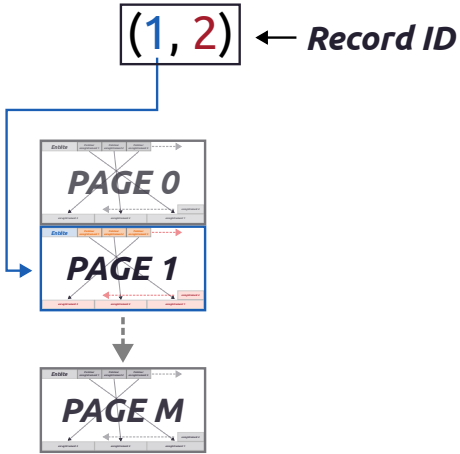
Identifiant pour les enregistrements permettant :

- Retrouver la page de l'enregistrement
- Retrouver l'enregistrement dans la page

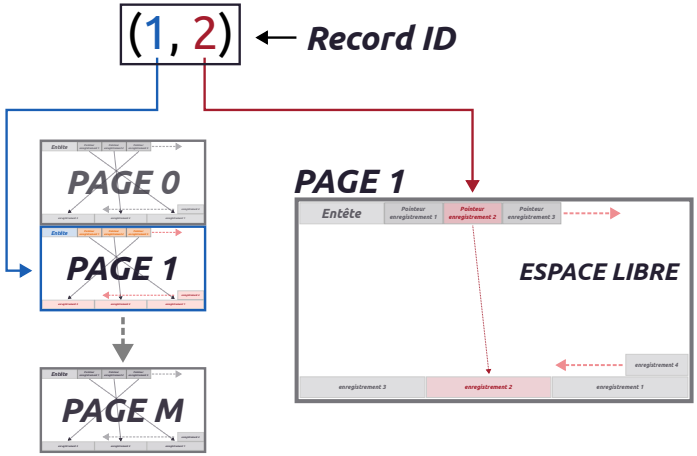
ctid	title
(0,1)	Toy Story
(0,2)	Jumanji
⋮	⋮
(0,22)	Hyvä poika
(1,1)	Copycat
⋮	⋮
(1,20)	Richard III
(2,1)	Dead Presidents
⋮	⋮
(2349,2)	Queerama

Table 1: ctid et titre de la table movie (`SELECT ctid,title FROM movies`)

Organisation en fichier(s) : accèder à une page/un enregistrement



Organisation en fichier(s) : accèder à une page/un enregistrement



Accéder à un enregistrement ?

- Depuis le numéro de page retrouver l'adresse de la page sur le disque
- Depuis le numéro de l'enregistrement, retrouver l'adresse de l'enregistrement

Accéder au **troisième enregistrement** de la cinquième page:

- l'adresse de la page est $0xAF00$ (données de l'énoncé)
- La taille d'un pointeur le disque est de 4 octets (taille fixe)
- La taille de l'entête est de 20 octets (taille fixe)

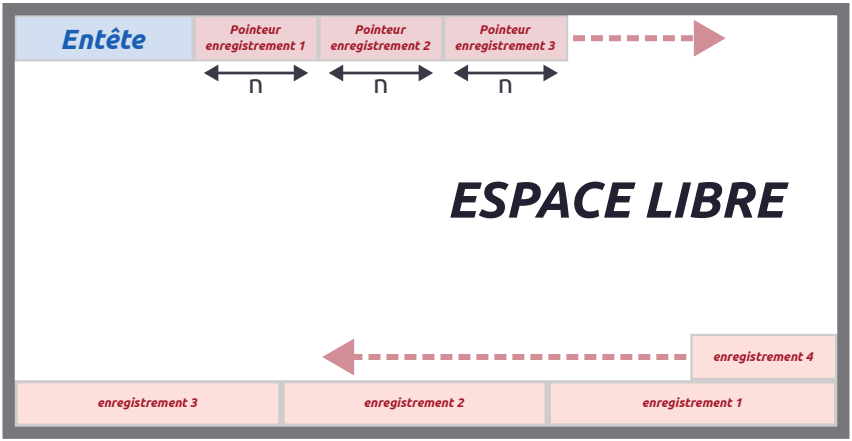
L'adresse du pointeur est donc (20 s'écrit 14 en hexadécimal):

-

$$e_3 = 0xAF00 + 4 \times 2 + 14 = 0xAF1C$$

e_3 **pointe sur le début de l'enregistrement !!!**

Organisation en fichier(s) : accèder à une page/un enregistrement



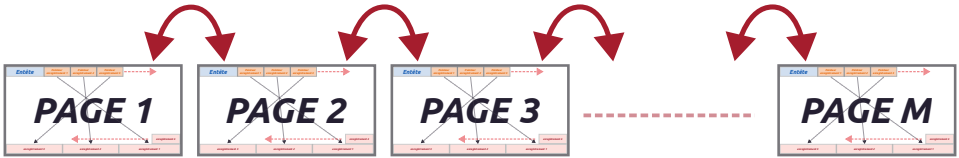
Comment accéder à la page n°k (adresse)

Organisation en liste chainées

- Chaque page contient l'adresse (sur le disque) de la page suivante
- Pour accéder à une page il faut lire la page précédente (ou du moins une partie)...

Organisation en répertoires

- Des pages spéciales existent contenant les adresses des pages disque
- Moins de lecture de pages pour accéder à n'importe quelle page



Le système de fichier (système)

En pratique on va s'abstraire de cette contrainte (dans ce cours)
$\rightarrow$ On considère les pages stockées dans un espace contiguë

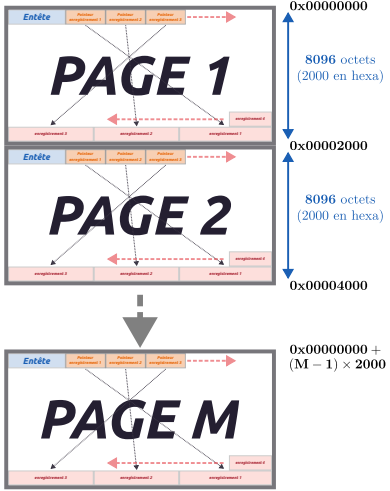
Organisation en fichier(s) : accèder à une page/un enregistrement

Le système de fichier (système)

En pratique on va s'abstraire de cette contrainte (dans ce cours)
$\rightarrow$ On considère les pages stockées dans un espace contiguë

Dans Postgres ?

$\rightarrow$ Utilisation du système de fichier du système d'exploitation (OS)



Des opérations

- **Parcours (SCAN)** : parcourir tous les enregistrements d'un fichier
- **Recherche sur égalité** : Retrouver les enregistrement avec un attributs ayant une valeur données
- **Recherche sur intervalle** : Retrouver les enregistrements avec un attributs dans un intervalle données
- **Insertion** : Ajouter un nouvel enregistrement
- **Suppression** : Supprimer un enregistrement (éventuellement par attribut ou rid)

Organisation en fichier(s) : Le tas

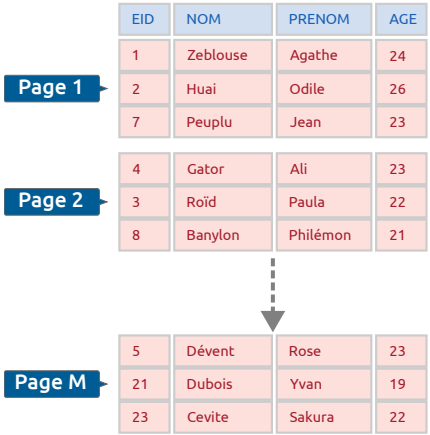
Le tas (Heap File)

- Structure simple
- Pas d'ordre dans les pages (enregistrements)

Ajouter un éléments ?

- Ajouter à la fin du fichier
  - dernière pages si assez d'espace
  - création d'une nouvelle page sinon

Un problème ?

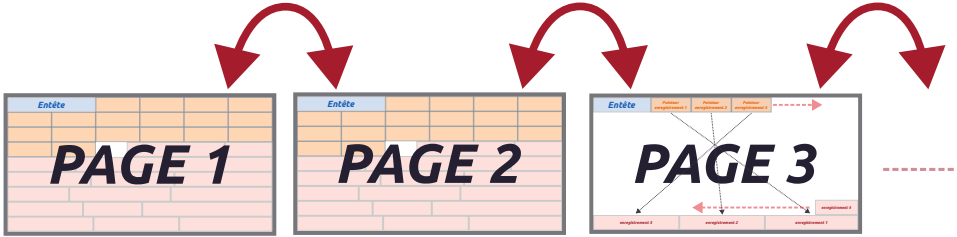


⚠ Des enregistrements peuvent êtres supprimés → dispersion des enregistrements (clairsemée)

→ Des pages sous-utilisés (mais prenant autant de place que les pages pleines)

Solutions ?

→ Parcourir les pages jusqu'a trouver une page avec un espace libre suffisant

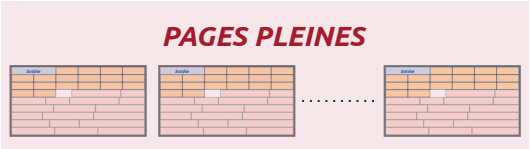


⚠ A chaque ajout d'enregistrement parcourir l'intégralité du fichier

Maintenir une liste des pages

Maintenir une liste avec les pages contenant un espace libre !!!

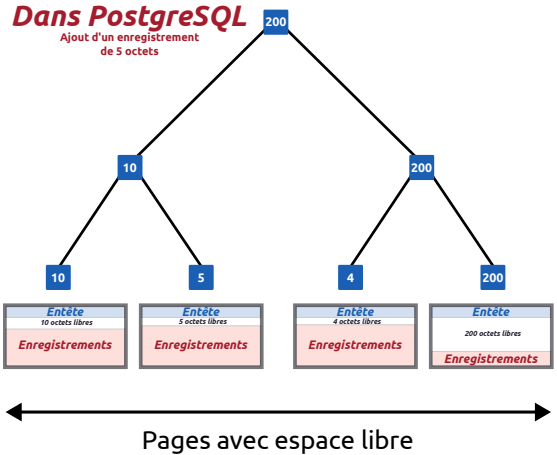
- À la création d'une page
- À la suppression d'une enregistrement



Dans postgres ?

Les pages libres sont stockées dans à l'aide d'une structure arborescente

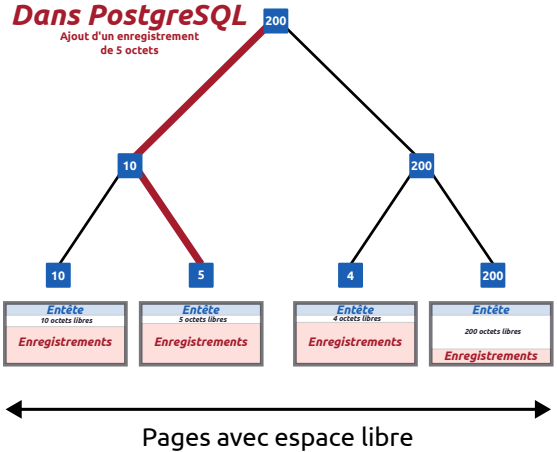
- Chaque nœud de l'arbre contient l'espace maximum des pages du sous arbre
- Chaque feuille pointe sur une page avec de l'espace libre



Dans postgres ?

Les pages libres sont stockées dans à l'aide d'une structure arborescente

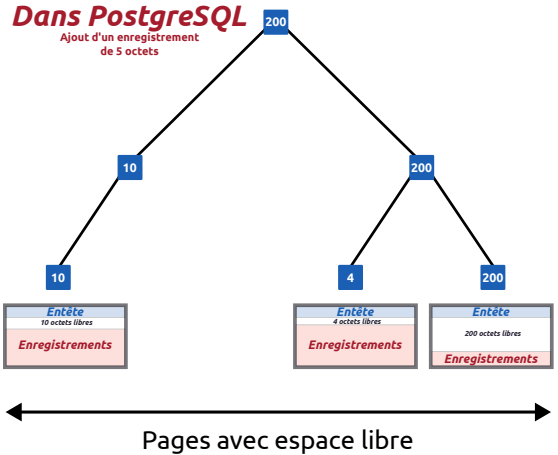
- Chaque nœud de l'arbre contient l'espace maximum des pages du sous arbre
- Chaque feuille pointe sur une page avec de l'espace libre



Dans postgres ?

Les pages libres sont stockées dans à l'aide d'une structure arborescente

- Chaque nœud de l'arbre contient l'espace maximum des pages du sous arbre
- Chaque feuille pointe sur une page avec de l'espace libre



Organisation en fichier(s) : Opérations sur un tas

$P \rightarrow$ Nombre de pages

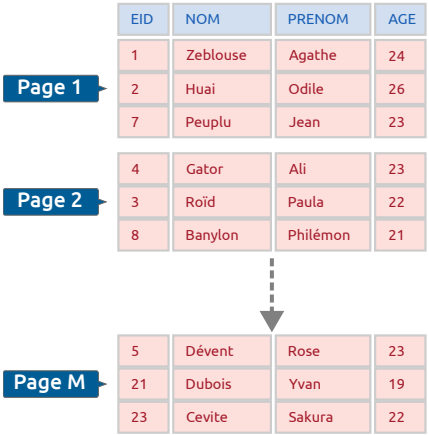
$T_P \rightarrow$ Temps d'ouverture d'une page

Le parcours

Ouverture de toutes les pages

$\rightarrow$ P pages sont ouvertes

$$\Rightarrow P \times T_P$$



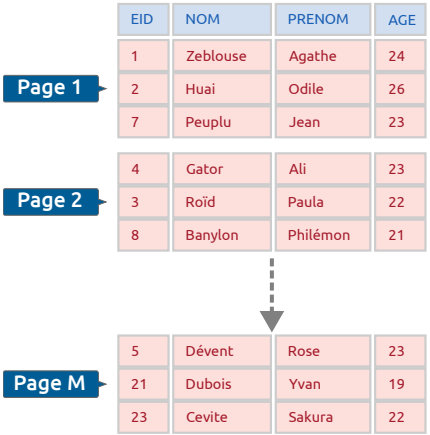
M pages dans le fichier organiser en tas (Heap-File)

Recherche sur égalité (unicité de l'attribut)

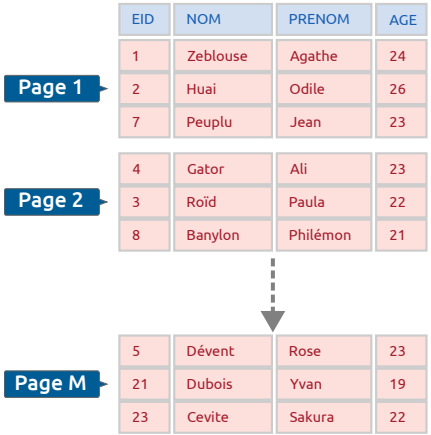
Pages non triés → ouverture de toutes les pages avant la bonne valeur

- **Au minimum** une page
- **Au maximum** P pages
- **En moyenne** $\frac{M}{2}$ pages

$$\Rightarrow \frac{P}{2} \times T_P$$



M pages dans le fichier organiser en tas (Heap-File)



Recherche sur égalité

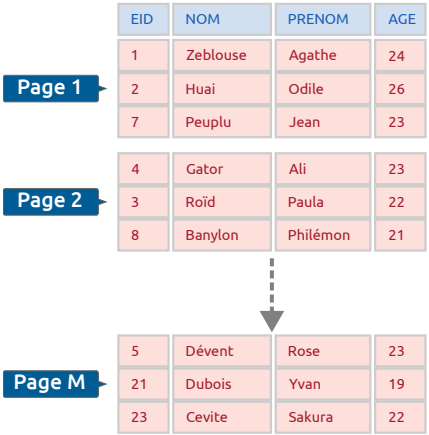
Pages non triés $\rightarrow$ ouverture de toutes les pages
$\rightarrow$ P pages sont ouvertes

$$\Rightarrow P \times T_P$$

Recherche d'un intervalle

Pages non triés $\rightarrow$ ouverture de toutes les pages
$\rightarrow$ P pages sont ouvertes

$$\Rightarrow P \times T_P$$

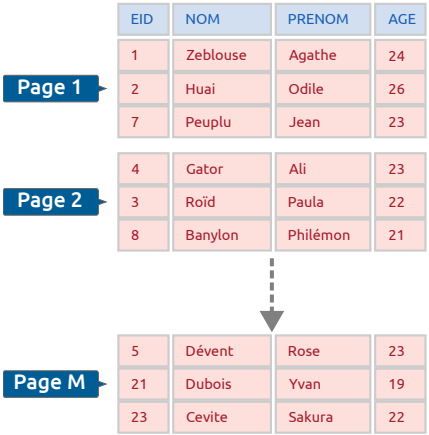


Insertion

Pages non triés $\rightarrow$ Placement libre
Si structure de l'espace libre sur $< N$ pages ($N << P$)

- Trouver une page libre $\rightarrow < N$ lectures
- Écriture des modifications (1 écriture)

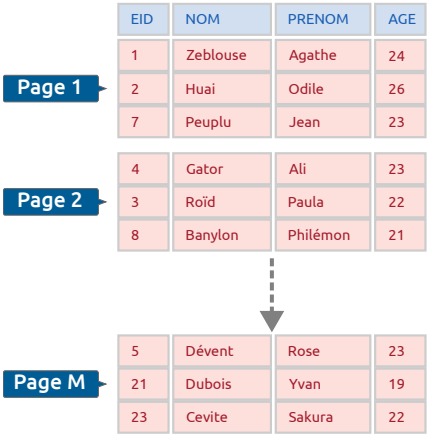
$$\Rightarrow N \times T_P + T_P = (N+2)T_P$$



La suppression

- Trouver la page contenant l'enregistrement → recherche
- Écriture des modifications (1 écriture)

$$\Rightarrow P \times T_P + T_P$$



Organisation en fichier(s) : Opérations sur un tas (recapitulatif)

	Scan	Rech (unicité)	Rech (intervalle)	Insert	Suppr
I/O (pages)	P	$\frac{P}{2}$	P	$N+2$	$P+1$
Complexité	P	P	P	1 (ne dépend pas de P)	P

→ ⚠ Peu efficace, excepté pour l'insertion

Des structures de fichier plus efficaces ?

Organisation en fichier(s) : Autre solution(s) ?

De meilleures Solutions

→ Si les enregistrements et les pages étaient triés sur un/des attributs

→ Si les enregistrements et les pages étaient regroupés sur un/des attributs

Organisation en fichier(s) : Un fichier trié

Le fichier trié

- Les enregistrements sont triés selon un attribut
- Les pages sont triés selon le même attribut

Données:

$P \rightarrow$ Nombre de pages
$T_P \rightarrow$ Temps d'ouverture d'une page

EID	NOM	PRENOM	AGE
21	Dubois	Yvan	19
8	Banylon	Philémon	21
3	Roïd	Paula	22

23	Cevite	Sakura	22
7	Peuplu	Jean	23
5	Dévent	Rose	23

4	Gator	Ali	23
1	Zeblouse	Agathe	24
2	Huai	Odile	26

Parcours

Ouverture de toutes les pages

$$\Rightarrow P \times T_P$$

EID	NOM	PRENOM	AGE
21	Dubois	Yvan	19
8	Banylon	Philémon	21
3	Roïd	Paula	22

23	Cevite	Sakura	22
7	Peuplu	Jean	23
5	Dévent	Rose	23

4	Gator	Ali	23
1	Zeblouse	Agathe	24
2	Huai	Odile	26

Organisation en fichier(s) : Opérations sur fichier trié

EID	NOM	PRENOM	AGE
1	Zeblouse	Agathe	24
2	Huai	Odile	26
3	Roïd	Paula	22
4	Gator	Ali	23
5	Dévent	Rose	23
7	Peuplu	Jean	23
8	Banylon	Philémon	21
21	Dubois	Yvan	19
23	Cevite	Sakura	22

Algorithm 1 Recherche par dichotomie

Require: P, a, v ▷ nb page, attribut, valeur

$s \leftarrow 0$

$e \leftarrow P$

EID	NOM	PRENOM	AGE
1	Zeblouse	Agathe	24
2	Huai	Odile	26
3	Roïd	Paula	22
4	Gator	Ali	23
5	Dévent	Rose	23
7	Peuplu	Jean	23
8	Banylon	Philémon	21
21	Dubois	Yvan	19
23	Cevite	Sakura	22

Algorithm 2 Recherche par dichotomie

Require: P, a, v ▷ nb page, attribut, valeur
  $s \leftarrow 0$
  $e \leftarrow P$
  while $e - s > 0$ **do**
    if $\exists e_a \in p_{\frac{e-s}{2}}$ tq $e_a = v$ **then**

EID	NOM	PRENOM	AGE
1	Zeblouse	Agathe	24
2	Huai	Odile	26
3	Roïd	Paula	22
4	Gator	Ali	23
5	Dévent	Rose	23
7	Peuplu	Jean	23
8	Banylon	Philémon	21
21	Dubois	Yvan	19
23	Cevite	Sakura	22

Algorithm 3 Recherche par dichotomie

Require: P, a, v ▷ nb page, attribut, valeur
  $s \leftarrow 0$
  $e \leftarrow P$
  while $e - s > 0$ **do**
    if $\exists e_a \in p_{\frac{e-s}{2}}$ tq $e_a = v$ **then**
      return $p_{\frac{e-s}{2}}$

EID	NOM	PRENOM	AGE
1	Zeblouse	Agathe	24
2	Huai	Odile	26
3	Roïd	Paula	22
4	Gator	Ali	23
5	Dévent	Rose	23
7	Peuplu	Jean	23
8	Banylon	Philémon	21
21	Dubois	Yvan	19
23	Cevite	Sakura	22

Algorithm 4 Recherche par dichotomie

Require: P, a, v ▷ nb page, attribut, valeur
  $s \leftarrow 0$
  $e \leftarrow P$
  while $e - s > 0$ **do**
    if $\exists e_a \in p_{\frac{e-s}{2}}$ tq $e_a = v$ **then**
      return $p_{\frac{e-s}{2}}$
    else if $v >$ **max** $p_{\frac{e-s}{2}}$ **then** ▷ Valeur max page $P_{\frac{e-s}{2}}$

EID	NOM	PRENOM	AGE
1	Zeblouse	Agathe	24
2	Huai	Odile	26
3	Roïd	Paula	22
4	Gator	Ali	23
5	Dévent	Rose	23
7	Peuplu	Jean	23
8	Banylon	Philémon	21
21	Dubois	Yvan	19
23	Cevite	Sakura	22

Algorithm 5 Recherche par dichotomie

Require: P, a, v ▷ nb page, attribut, valeur
  $s \leftarrow 0$
  $e \leftarrow P$
  while $e - s > 0$ **do**
    if $\exists e_a \in p_{\frac{e-s}{2}}$ tq $e_a = v$ **then**
      return $p_{\frac{e-s}{2}}$
    else if $v >$ **max** $p_{\frac{e-s}{2}}$ **then** ▷ Valeur max page $P_{\frac{e-s}{2}}$
      $s \leftarrow \frac{e-s}{2}$

EID	NOM	PRENOM	AGE
1	Zeblouse	Agathe	24
2	Huai	Odile	26
3	Roïd	Paula	22
4	Gator	Ali	23
5	Dévent	Rose	23
7	Peuplu	Jean	23
8	Banylon	Philémon	21
21	Dubois	Yvan	19
23	Cevite	Sakura	22

Algorithm 6 Recherche par dichotomie

Require: P, a, v ▷ nb page, attribut, valeur
  $s \leftarrow 0$
  $e \leftarrow P$
  while $e - s > 0$ **do**
    if $\exists e_a \in p_{\frac{e-s}{2}}$ tq $e_a = v$ **then**
      return $p_{\frac{e-s}{2}}$
    else if $v > \max p_{\frac{e-s}{2}}$ **then** ▷ Valeur max page $P_{\frac{e-s}{2}}$
      $s \leftarrow \frac{e-s}{2}$
    else if $v < \min p_{\frac{e-s}{2}}$ **then** ▷ Valeur min dans $P_{\frac{e-s}{2}}$

EID	NOM	PRENOM	AGE
1	Zeblouse	Agathe	24
2	Huai	Odile	26
3	Roïd	Paula	22
4	Gator	Ali	23
5	Dévent	Rose	23
7	Peuplu	Jean	23
8	Banylon	Philémon	21
21	Dubois	Yvan	19
23	Cevite	Sakura	22

Algorithm 7 Recherche par dichotomie

Require: P, a, v ▷ nb page, attribut, valeur
  $s \leftarrow 0$
  $e \leftarrow P$
  while $e - s > 0$ **do**
    if $\exists e_a \in p_{\frac{e-s}{2}}$ tq $e_a = v$ **then**
      return $p_{\frac{e-s}{2}}$
    else if $v >$ **max** $p_{\frac{e-s}{2}}$ **then** ▷ Valeur max page $P_{\frac{e-s}{2}}$
      $s \leftarrow \frac{e-s}{2}$
    else if $v <$ **min** $p_{\frac{e-s}{2}}$ **then** ▷ Valeur min dans $P_{\frac{e-s}{2}}$
      $e \leftarrow \frac{e-s}{2}$

EID	NOM	PRENOM	AGE
1	Zeblouse	Agathe	24
2	Huai	Odile	26
3	Roïd	Paula	22
4	Gator	Ali	23
5	Dévent	Rose	23
7	Peuplu	Jean	23
8	Banylon	Philémon	21
21	Dubois	Yvan	19
23	Cevite	Sakura	22

Algorithm 8 Recherche par dichotomie

Require: P, a, v ▷ nb page, attribut, valeur
  $s \leftarrow 0$
  $e \leftarrow P$
  while $e - s > 0$ **do**
    if $\exists e_a \in p_{\frac{e-s}{2}}$ tq $e_a = v$ **then**
      return $p_{\frac{e-s}{2}}$
    else if $v > \max p_{\frac{e-s}{2}}$ **then** ▷ Valeur max page $P_{\frac{e-s}{2}}$
      $s \leftarrow \frac{e-s}{2}$
    else if $v < \min p_{\frac{e-s}{2}}$ **then** ▷ Valeur min dans $P_{\frac{e-s}{2}}$
      $e \leftarrow \frac{e-s}{2}$
    else

EID	NOM	PRENOM	AGE
1	Zeblouse	Agathe	24
2	Huai	Odile	26
3	Roïd	Paula	22
4	Gator	Ali	23
5	Dévent	Rose	23
7	Peuplu	Jean	23
8	Banylon	Philémon	21
21	Dubois	Yvan	19
23	Cevite	Sakura	22

Algorithm 9 Recherche par dichotomie

Require: P, a, v ▷ nb page, attribut, valeur
  $s \leftarrow 0$
  $e \leftarrow P$
  while $e - s > 0$ **do**
    if $\exists e_a \in p_{\frac{e-s}{2}}$ tq $e_a = v$ **then**
      return $p_{\frac{e-s}{2}}$
    else if $v >$ **max** $p_{\frac{e-s}{2}}$ **then** ▷ Valeur max page $P_{\frac{e-s}{2}}$
      $s \leftarrow \frac{e-s}{2}$
    else if $v <$ **min** $p_{\frac{e-s}{2}}$ **then** ▷ Valeur min dans $P_{\frac{e-s}{2}}$
      $e \leftarrow \frac{e-s}{2}$
    else
      return NULL
    end if
  end while

Recherche (égalité)

Recherche par dichotomie → division par deux de l'espace de recherche à chaque étape

- Au maximum $\leq log_2(P)$ étapes
- Une page lue par étape
- Lecture de $\leq log_2(P)$ pages max

$$\Rightarrow log_2(P) \times T_P$$

EID	NOM	PRENOM	AGE
1	Zeblouse	Agathe	24
2	Huai	Odile	26
3	Roïd	Paula	22
4	Gator	Ali	23
5	Dévent	Rose	23
7	Peuplu	Jean	23
8	Banylon	Philémon	21
21	Dubois	Yvan	19
23	Cevite	Sakura	22

Recherche (intervalle)

Recherche par dichotomie puis parcours des pages ayant l'attribut la valeur dans le bon intervalle (N)

$$\Rightarrow < (log_2(P) + N) \times T_P$$

Exemple

Trouver les enregistrements avec l'attribut $age < 23$

- Recherche d'un élément $< sup(log_2(3)) = 2$ pages (arrondi supérieur)
- Parcourir les pages (ici 2 pages où $age < 23$)

EID	NOM	PRENOM	AGE
21	Dubois	Yvan	19
8	Banylon	Philémon	21
3	Roïd	Paula	22

23	Cevite	Sakura	22
7	Peuplu	Jean	23
5	Dévent	Rose	23

4	Gator	Ali	23
1	Zeblouse	Agathe	24
2	Huai	Odile	26

Insertion

1. Recherche (un élément) $\rightarrow \approx log_2(P)$
2. Insertion de l'enregistrement $\rightarrow 1$
3. Décalage dans toutes les pages suivantes...
   ⚠ $\rightarrow 2 \times P$ (lecture/écriture)

$$\Rightarrow < (log_2(P) + 2 \times P + 1) \times T_P$$

EID	NOM	PRENOM	AGE
21	Dubois	Yvan	19
8	Banylon	Philémon	21
3	Roïd	Paula	22
23	Cevite	Sakura	22
7	Peuplu	Jean	23
5	Dévent	Rose	23
4	Gator	Ali	23
1	Zeblouse	Agathe	24
2	Huai	Odile	26

Suppression

1. Recherche (un élément) $\rightarrow \approx log_2(P)$
2. Suppression de l'enregistrement $\rightarrow 1$
3. Décalage dans toutes les pages suivantes...
   ⚠ $\rightarrow 2 \times P$ (lecture/écriture)

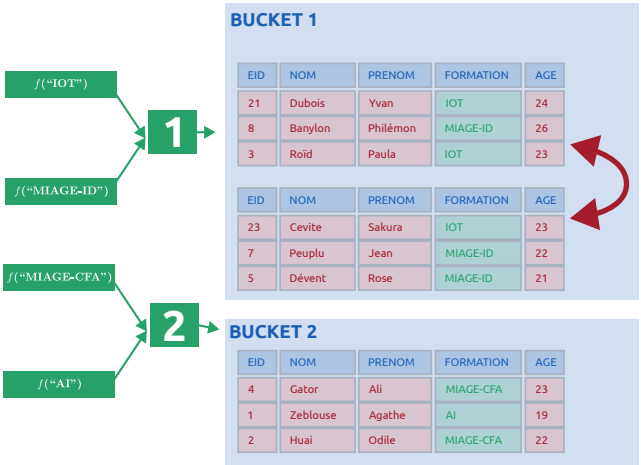
$$\Rightarrow < (log_2(P) + 2 \times P + 1) \times T_P$$

EID	NOM	PRENOM	AGE
21	Dubois	Yvan	19
8	Banylon	Philémon	21
3	Roïd	Paula	22
23	Cevite	Sakura	22
7	Peuplu	Jean	23
5	Dévent	Rose	23
4	Gator	Ali	23
1	Zeblouse	Agathe	24
2	Huai	Odile	26

Organisation en fichier(s) : Opérations sur fichier trié (recapitulatif)

	Scan	Rech (unicité)	Rech (intervalle)	Insert	Suppr
I/O (pages)	P	$log_2(P)$	$log_2(P) + N$	$log_2(P) + 2P$	$log_2(P) + 2P$
Complexité	P	$log(P)$	$log(P)$	P	P

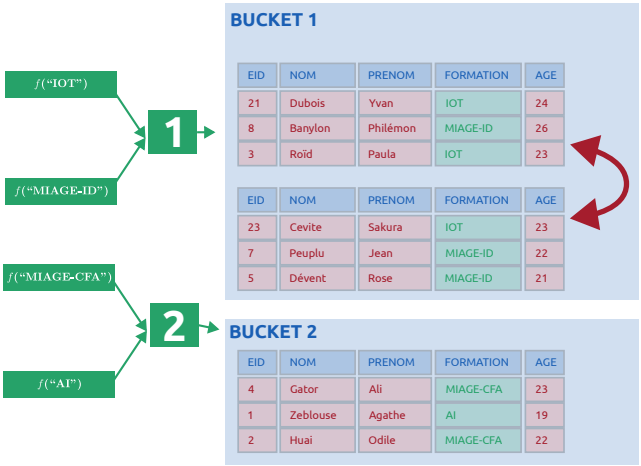
→ ⚠ Éfficace pour la recherche



Fichier Haché

- Les enregistrements sont regroupés par valeurs
- On supposera une page par valeur dans cet exemple
- Il existe une fonction prenant la valeur de l'attribut retournant la/les pages correspondantes

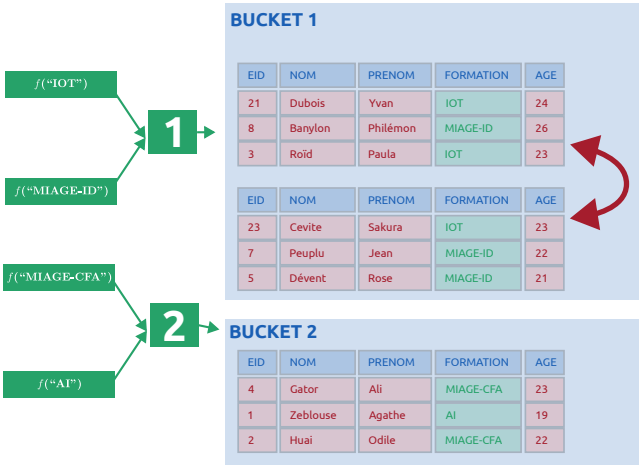
Organisation en fichier(s) : Fichier Haché



Parcours

Toutes les pages :

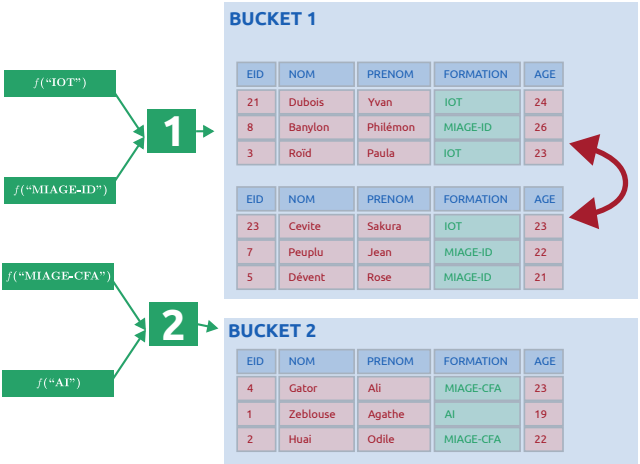
$$\Rightarrow P \times T_p$$



Recherche (égalité)
La fonction de hachage nous renvoi sur la bonne page donc une ouverture

$$T_p$$

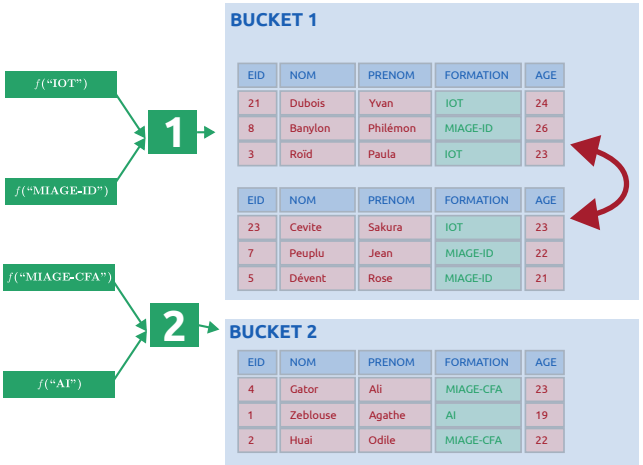
Organisation en fichier(s) : Fichier Haché



Recherche (intervalle)
Il faut parcourir toutes les valeurs :

$$\Rightarrow P \times T_p$$

Organisation en fichier(s) : Fichier Haché

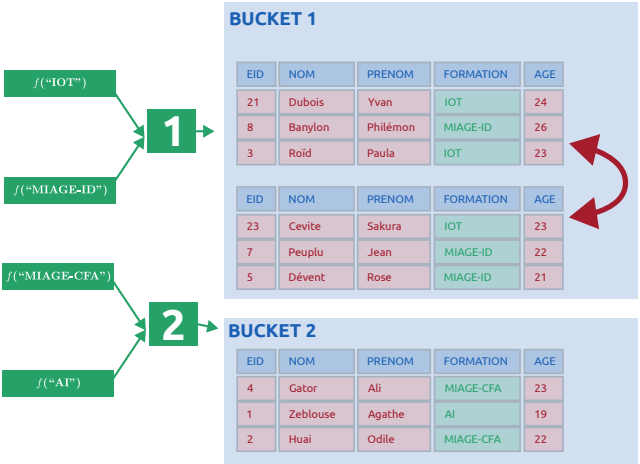


Insertion

- Rechercher la page (1 ouverture)
- Ajouter l'enregistrement (1 écriture)

$$\Rightarrow 2 \times T_p$$

Organisation en fichier(s) : Fichier Haché



Suppression

- Rechercher la page (1 ouverture)
- supprimer l'enregistrement (1 écriture)

$$\Rightarrow 2 \times T_p$$

Organisation en fichier(s) : Fichier Haché(recapitulatif)

	Scan	Rech (unicité)	Rech (intervalle)	Insert	Suppr
I/O (pages)	P	1	P	2	2
Complexité	P	1	P	1	1

→ Efficace pour la recherche sur égalité et la mise à jour

→ ⚠ Équilibrer les groupes (dépend de la fonction de hash)

Comparaison des organisations des fichiers :

	Scan	Rech (égalité)	Rech (intervalle)	Insert	Suppr
Tas (Heap File)	P	P	P	1	1
Trié (Sorted File)	P	$log(P)$	$log(P)$	P	P
Haché (Hashed File)	P	1	P	1	1

- Pour la recherche, préférence pour les fichiers triés
- Pour la recherche sur égalité et l'insertion/suppression, les fichiers hachés

Organisation en fichier(s) : Conclusion

Ce que nous avons vu :

- L'organisation en fichier des relations
- Les différents types de fichier

⚠ Les fichiers triés et hachés sont organisés selon un unique attribut !!!

Comment accélérer les opérations sur des attributs différents ?

Utilisation d'index !!!

Conclusion

Conclusion

- Organisation en page des données
- Les gestionnaires de mémoires

Dans la suite du cours **L'unité de temps pour les calculs du coût sera un transfert disque/mémoire**