SGBD-R : système

Stockage

Guillaume Raschia — Nantes Université

originaux de Philippe Rigaux, CNAM

Dernière mise-à-jour : 27 février 2023

Plan de la session

Support de stockage (S2.1)

Gestion des mémoires (S2.2)

Organisation des fichiers (S2.3)

Support de stockage (S2.1)

Techniques de stockage

Contenu de cette section :

- Où sont les données ? Mémoire RAM, mémoire persistante
- Les performances
- Disques, SSD [1]

Vocabulaire

Une « donnée » désigne une unité physique d'information : pour nous, une ligne d'une table, codée sous forme d'un enregistrement (*record*).

1. *Solid-State Drive*, disques à mémoire flash.

Problématique du stockage

Un SGBD gère deux types de mémoire :

- La mémoire centrale, volatile, rapide mais risque de perte.
- La mémoire secondaire, persistante, stable mais lente.

Les données sont toujours en mémoire secondaire dans des fichiers, et partiellement copiées en mémoire centrale.

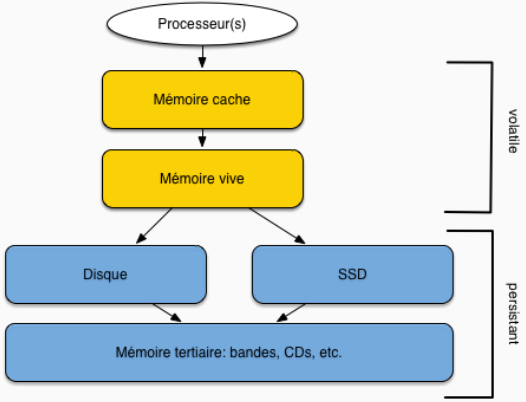
Une donnée, pour être traitée, doit impérativement être en mémoire centrale.

Comment gérer au mieux ces deux ressources ?

Les mémoires d'un ordinateur

Les mémoires dans un ordinateur forment une hiérarchie.

Constat : plus une mémoire est rapide, plus elle est chère , et moins elle est volumineuse.



Performances des mémoires

Critères essentiels :

- Temps d'accès : connaissant l'adresse d'une donnée, quel est le temps nécessaire pour aller la chercher à l'emplacement mémoire indiqué par cette adresse ?
  Critère essentiel pour les opérations dites d'accès direct.
- Débit : volume de données lues par unité de temps dans le meilleur des cas.
  Critère essentiel pour les opérations dites de lecture séquentielle.

Constat : tout accès aux données s'effectue avec l'une de ces deux opérations.

Quelques ordres de grandeur

Rappel : 1 Go = 1 000 Mo et 1 To = 1 000 Go.

Mémoire	Coût	Taille	Temps d'accès	Débit
cache	€€€	1Mo	$\approx 10^{-8}$ (10 nanosec.)	10+ Go/s
Principale (RAM)	€€	10Go	$\approx 10^{-8} - 10^{-7}$ (10-100 nanosec.)	Qq Go/s
SSD	€€	1To	$\approx 10^{-4}$ (0,1 millisec.)	1+ Go/s
Disque	€	10To	$\approx 10^{-2}$ (10 millisec.)	100 Mo/s

Un accès direct au disque est (environ) un million de fois plus coûteux qu'en mémoire centrale !

Le débit d'un disque est 20, 30, 40 fois plus lent que le débit RAM.

Un SSD est 10 à 100 fois plus performant qu'un disque magnétique.

Importance pour les bases de données

Un SGBD doit ranger sur disque les données ;

- parce qu'elle sont trop volumineuses (de moins en moins vrai) ;
- parce qu'elles sont **persistantes** (et doivent survivre à un arrêt du système).

Le SGBD doit toujours amener les données en mémoire pour les traiter.

Si possible, les données utiles devraient résider le plus possible en mémoire.

Principe général : la capacité à placer les données utiles au plus près du processeur (donc en RAM si possible) est un facteur essentiel de performance.

Organisation d'un disque

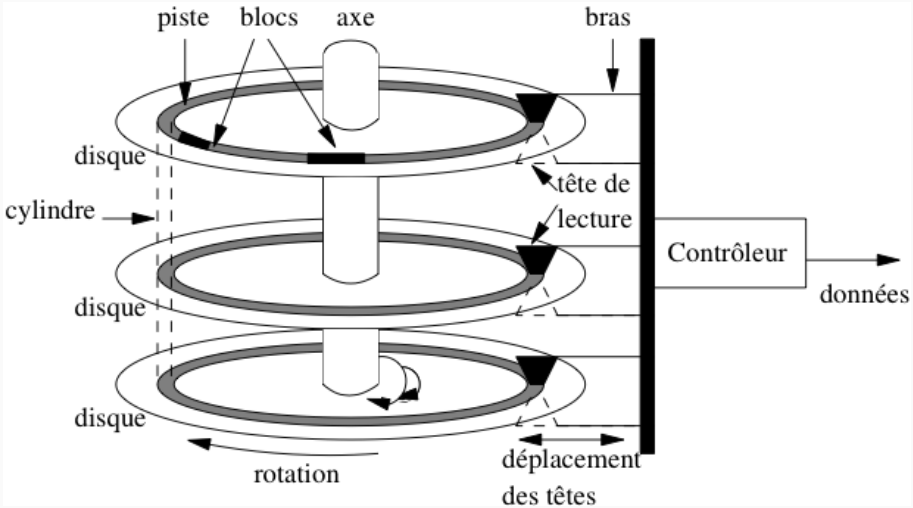
Disque : surface magnétique, stockant des 0 ou des 1.

Par extension : ensemble de surfaces (simple ou double-face) ayant mêmes caractéristiques.

Dispositif : les surfaces sont entraînées dans un mouvement de rotation ; les têtes de lecture se déplacent dans un plan fixe.

- Le secteur est la plus petite unité physique de stockage d'un disque, sa taille minimum est de 4 094 octets ;
- La piste est l'ensemble des secteurs d'une surface lus au cours d'une rotation ;
- le cylindre est l'empilement de pistes situées sous les têtes de lecture.

Structure d'un disque



Disque = accès semi-direct = latence

Adressage Cylindre/Tête/Secteur = numéro du disque (Tête) ; de la piste (Cylindre) où se trouve le bloc ; du numéro du bloc (Secteur) sur la piste.

- Délai de positionnement pour placer la tête sur la bonne piste :
- Délai de latence pour attendre que le bloc passe sous la tête de lecture ;
- Temps de transfert pour attendre que le (ou les) bloc(s) soient lus et transférés.

La latence d'un disque : on ne peut pas lire une donnée avant qu'elle passe sous une tête de lecture : la latence (qq ms) est un facteur pénalisant.

Granularité d'entrée/sortie : le bloc

Le bloc est une séquence de secteurs, défini par le système d'exploitation. Sa taille est donc un multiple de 4Ko.

Granularité : on lit toujours au moins un bloc même si on ne veut qu'un octet ; le bloc est l'unité d'entrée/sortie !

⇒ le remplissage des blocs, si possible avec des données « proches » de la donnée requise initialement, est essentiel.

Localité : la latence en pratique dépend de la proximité des adresses successives demandées par l'application.

Principe : si deux données sont proches du point de vue applicatif, alors elles doivent être proches sur le disque (idéalement, dans le même bloc).

Résumé

La gestion des disques n'est pas directement sous le contrôle de l'administrateur.

Retenir :

- Accès (aléatoire) au disque : très lent par rapport à un accès en RAM (qq. ms).
- Débit du disque : faible (100 Mo/s au mieux)
- SSD : 100× plus rapide qu'un disque (latence), 10× plus en débit

Règle générale : définir des données compactes (types) et stockées contiguement.

Organisation des données, algorithmes d'accès, concurrence : tout est conçu pour minimiser les accès aux disques, et surtout les accès aléatoires.

Gestion des mémoires (S2.2)

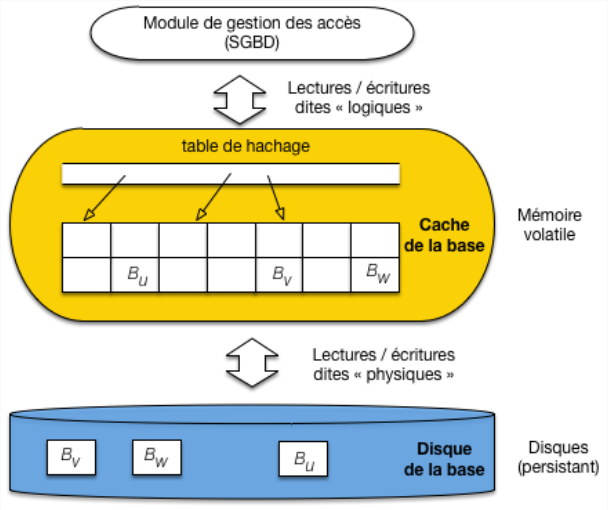
Gestion des mémoires

Contenu de cette section :

- Mémoires gérées par un SGBD : mémoire cache, disque
- La gestion des lectures
- La gestion des écritures
- Quelques applications du principe de localité

Le cache et le disque

Paramétrage des SGBD : partie de la mémoire centrale qui sert de *cache*.



Opération de lecture

Toute opération de lecture contient l'adresse du bloc à lire.

- Si la donnée fait partie d'un bloc dans le cache, le SGBD prend le bloc, accède à la donnée dans le bloc, et la retourne ;
- Sinon il faut d'abord lire un bloc du disque, et le placer dans le cache, pour se ramener au cas précédent.

Hypothèse de localité : le SGBD garde en mémoire les blocs après utilisation, afin d'exploiter à la fois

- la localité spatiale : les autres données du bloc ;
- la localité temporelle : il est probable qu'on va relire la donnée dans peu de temps.

Lectures logiques, lectures physiques : le *Hit ratio*

Le paramètre qui mesure l'efficacité d'une mémoire cache est le *hit ratio* :

$$hit\ ratio\ =\ \frac{nb\ de\ lectures\ logiques\ -\ nb\ de\ lectures\ physiques}{nb\ de\ lectures\ logiques}$$

- Si toutes les lectures logiques (demande de bloc) aboutissent à une lecture physique (accès au disque), le *hit ratio* est 0.
- Aucune lecture physique (tout est en mémoire) : le *hit ratio* est de 1.

En pratique. Un bon *Hit Ratio* est supérieur à 80%, voire 90% : presque toutes les lectures se font en mémoire !

Comment avoir un bon *Hit ratio* ?

Évident :

- il faut allouer le plus de mémoire possible au SGBD ;
- il faut limiter la taille de la base (attention aux longs textes, aux images, aux données binaires…).

Mais surtout : il faut que les données **utiles** soient en mémoire centrale

Intuition : certaines parties de la base sont beaucoup plus lues que d'autres : il faut qu'elles tiennent en mémoire.

Stratégie de remplacement

Que faire quand la mémoire est pleine ?

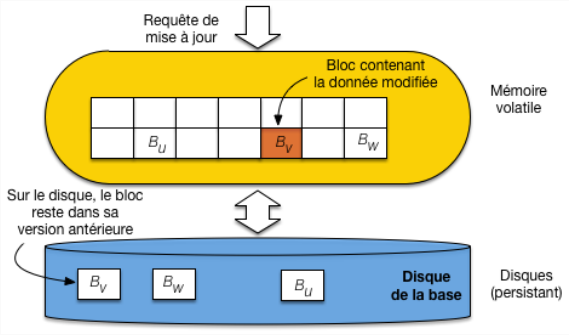
On applique le principe de localité : l'algorithme le plus courant est dit *Least Recently Used* (LRU).

- La « victime » est le bloc dont la dernière date de lecture logique est la plus ancienne.
- Ce bloc est alors soustrait de la mémoire centrale.
- Le nouveau bloc vient le remplacer.

Conséquence : le contenu du cache est une image fidèle de l'activité récente sur la base de données.

Opération de mise à jour

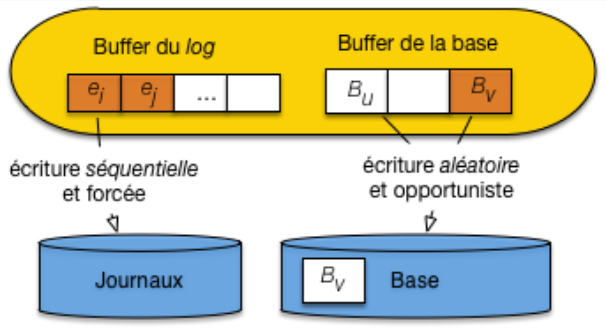
Approche naïve : on trouve le bloc (comme pour une lecture), on modifie la donnée.



Faut-il écrire le bloc ou non ? Deux mauvais choix (pourquoi ?)

La bonne méthode ; le fichier journal

On gère la base (accès aléatoire) et un fichier séquentiel, le journal (log).



Problème résolu : données sur disque (sûres) et écritures séquentielles.

Le principe de localité et ses conséquences

Principe de localité : l'ensemble des données utilisées par une application pendant une période donnée forme souvent un groupe bien identifié.

- Localité spatiale : si une donnée d est utilisée, les données proches de d ont de fortes chances de l'être également.
- Localité temporelle : quand une application accède à une donnée d, il y a de fortes chances qu'elle y accède à nouveau peu de temps après.
- Localité de référence : si une donnée d_1 référence une donnée d_2, l'accès à d_1 entraîne souvent l'accès à d_2.

Les systèmes exploitent ce principe en déplaçant dans la hiérarchie des mémoires des groupes de données.

Résumé

Un critère essentiel de la performance est la taille de la mémoire cache.

Pour vérifier que le système est bien paramétré : on calcule le *Hit Ratio*. Il doit être supérieur à 80%

Sinon, les pistes :

- Vérifier qu'on ne lit que des données utiles.
- Regarder si on peut limiter la taille des données (ex : partitionner la table en mettant les colonnes volumineuses à part).
- Ajouter de la mémoire...

Organisation des fichiers (S2.3)

Enregistrements, blocs et fichiers

Contenu de cette section : la représentation physique d'une base de données (par le SGBD).

- les enregistrements et leur adressage
- le stockage des enregistrements dans des blocs
- la séquence des blocs formant un fichier.

Une application n'accède jamais directement à la représentation physique.

Enregistrements

Un enregistrement = une suite de *champs* stockant les valeurs des attributs.

Type	Taille en octets
`INTEGER`	4
`FLOAT`	4
`DOUBLE PRECISION`	8
`DECIMAL`(M, D)	M, (D+2 si M $<$ D)
`CHAR`(M)	M
`VARCHAR`(M)	L+1 avec L $\leq$ M

Certains champs ont une taille variable ; d'autres peuvent être à `NULL` : pas de valeur.

Tailles variables et valeurs NULL

Si tous les champs étaient de taille fixe et avaient une valeur : pas de problème.

En pratique certains champs ont une taille variable ou sont à `NULL`.

- pour les champs de taille variable : on précède la valeur par la taille exacte ;
- pour les valeurs à `NULL` : on peut indiquer une taille 0 (Oracle) ; on peut créer un « masque » de bits.

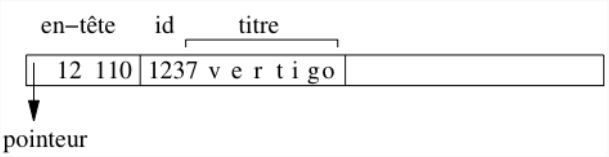
La norme n'impose pas de choix, mais dans tous les cas il faut gérer une information complémentaire sur les enregistrements, et la placer dans un entête.

En-tête d'enregistrement

Les informations complémentaires sont stockées dans l'en-tête d'un enregistrement.

Exemple :

- table Film (id INT, titre VARCHAR(50), année INT)
- Enregistrement (123, 'Vertigo', NULL)



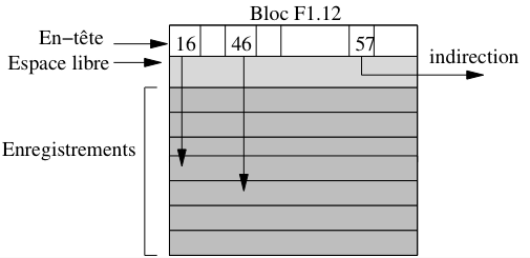
Le contenu de l'enregistrement est décodé au moment de la lecture.

Blocs

On veut qu'un enregistrement soit stocké de manière stable pour avoir une adresse fixe ; MAIS on doit envisager :

- le cas où la taille d'un enregistrement varie ;
- une méthode pour gérer un déplacement si nécessaire.

Une solution (Oracle) : chaque bloc gère les adresses internes de ses enregistrements.



Adressage des enregistrements

Avec le schéma précédent, l'adresse d'un enregistrement est constituée

- d'une adresse physique, celle du bloc. Ex : F1.12
- d'une adresse logique, interne au bloc. Ex : 16

L'adresse est fixe et utilisée pour les accès direct (indexation).

Avec cette solution :

- un enregistrement peut changer de taille, avec réorganisation interne ;
- plus de place dans le bloc ? On peut déplacer l'enregistrement.

Réorganisation du stockage

Un enregistrement s'agrandit, mais il reste de la place dans le bloc :

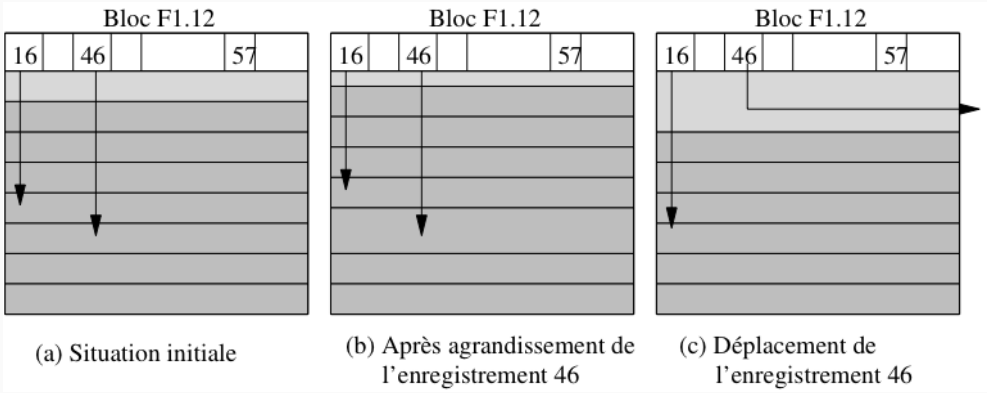
- on modifie l'organisation interne ; la table locale d'adressage est modifiée

Un enregistrement s'agrandit, plus de place dans le bloc :

- on déplace l'enregistrement et on crée un chaînage dans l'en-tête du bloc.

La création de chaînage pénalise les performances $\Rightarrow$ si possible laisser de l'espace libre dans un bloc.

Exemple d'évolution avec chaînage



(a) Situation initiale

(b) Après agrandissement de l'enregistrement 46

(c) Déplacement de l'enregistrement 46

On agrandit deux fois successivement l'enregistrement F1.12.46.

Fichier = séquence de blocs

Un fichier est simplement une séquence de blocs contenant les données d'un même objet (table, index).

La qualité d'un fichier est conditionnée par :

- la bonne utilisation de l'espace (idéalement tous les blocs sont pleins);
- le stockage le plus contigu possible (même piste, même cylindre, etc);
- peu ou pas de chaînage.

Si les blocs sont très dispersés sur le disque, on aboutit à une fragmentation très pénalisante.

Opérations sur un fichier

Essentiellement :

- En l'absence d'index, le seul moyen de rechercher un enregistrement est de parcourir séquentiellement le fichier.
- Si le fichier est trié il est possible d'effectuer des recherches par dichotomie.

En général, les fichiers gérés par les SGBD sont de simple fichiers séquentiels, sans ordre.

Résumé

La gestion des fichiers est entièrement à la charge du SGBD.

Retenir :

- Les enregistrements sont stockés dans des blocs.
- Le contenu des blocs s'adapte au changement de taille d'un enregistrement après insertion.
- Il faut un dispositif stable d'adressage de chaque enregistrement.
- Les fichiers sont des séquences de blocs, les moins fragmentés possibles.

L'adressage stable est essentiel pour permettre l'indexation et l'accès direct aux enregistrements.