Cours de bases de données, aspects systèmes, http://sys.bdpedia.fr

Gestion des mémoires

Philippe Rigaux

le cnam

Gestion des mémoires

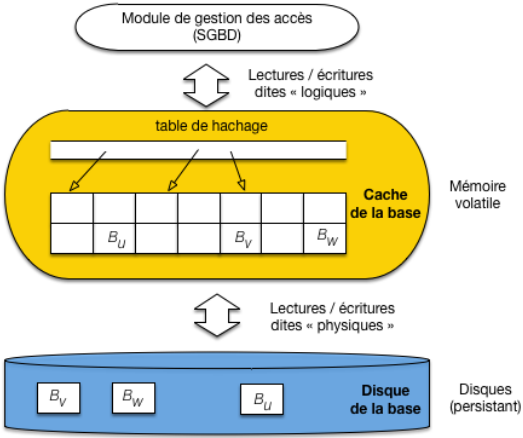
Contenu de ce cours :

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

Ces diapositives correspondent au support en ligne disponible sur le site `http://sys.bdpedia.fr/stock.html#s2-gestion-des-memoires`

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* :

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

- 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* ?

Evident :

- 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 ?

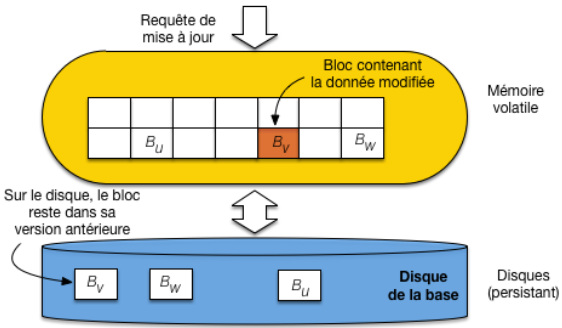
On applique le principe de localité : l'algorithme le plus courant est dit *Least Recently Used* (LRU).

- 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

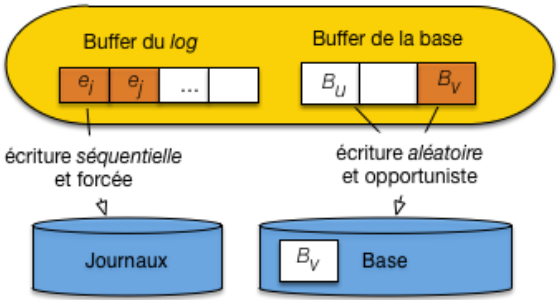
Approche naïve : on trouve le bloc (comme pour une lecture), on modifie la donnée.



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

La bonne méthode ; le fichier journal

On gère la base (accès alléatoire) **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...