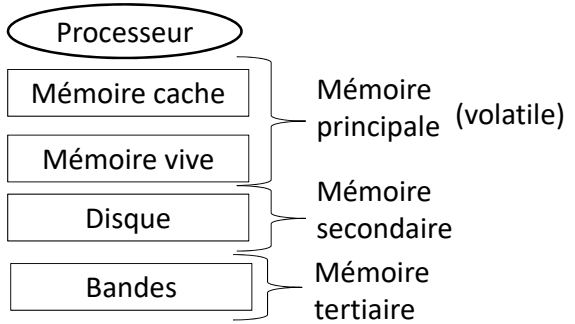
Bases de données avancées - Optimisation
Stockage des données

Sarah Cohen-Boulakia
Laboratoire de Recherche en Informatique
Université Paris Sud
http://www.lri.fr/~cohen/BD/BD.html

Les niveaux de mémoires

- **Mémoire non-volatile**
  - Les informations demeurent malgré l'arrêt de toute alimentation électrique
  - Exemple : le disque dur; contre-exemple : la RAM



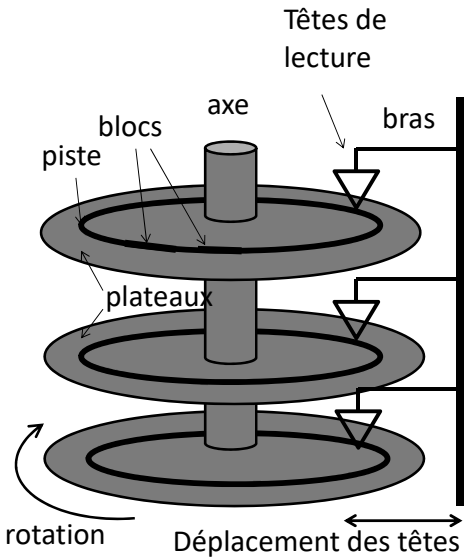
Mémoire principale et secondaire

- La mémoire vive et les disques : niveaux pour les BD
- >80 euros : 16 Go RAM ou 1 tera de disque
- Une BD est (presque) toujours placée sur le disque
- Mais les données doivent être chargées en mémoire pour être traitées

→Transferts fréquents entre mémoire secondaire (disque) et principale (RAM) pour satisfaire les requêtes

- Temps d'accès à la RAM 1 million de fois plus rapide que le temps d'accès au disque

Fonctionnement d'un disque



- Le disque a une taille fixe pour chaque secteur
- Données stockées sur des blocs (= *pages*)
- Un bloc a la taille d'un ou plusieurs secteurs
- Blocs regroupés en anneaux : les pistes

Fonctionnement d'un disque

- Données stockées sur des blocs (= pages)
  - C'est l'unité de lecture et écriture
  - →Même si seuls 4 octets doivent être lus c'est tout le bloc que l'on charge !
- Ensemble des pistes de même diamètre = cylindre
  - →Toutes les données accessibles sans déplacer les têtes
- Temps d'accès = temps de recherche (positionnement de la tête de lecture) + (temps rotation + temps de transfert)

Optimisation = stratégie de placement

Temps d'accès au disque

- Accès à un *secteur arbitraire*
  - Positionner la tête sur la bonne piste (recherche)
  - Attendre que le bon secteur soit sous la tête (rotation)
  - Parcourir le secteur et renvoyer les données en mémoire (transfert)

➔ Tout cela constitue la ***latence*** du disque

- Une fois qu'on a payé le temps de latence, lire le secteur suivant ne demande que le temps de transfert
- On va donc essayer d'organiser les données de manière éviter les déplacements arbitraires

Principe de Localité

- Localité spatiale
  - si une donnée *d* est utilisée, les données "proches" de *d* ont de fortes chances d'être utilisées
- Localité temporelle
  - si une application accède à la donnée *d* à un temps *t*, il y a de fortes chances qu'elle accède à nouveau à *d* à *t+i* (*i* petit)
- Localité de référence
  - si une donnée *d1* référence une donnée *d2*, l'accès à *d1* entraîne souvent l'accès à *d2*

Optimisation et placement

- Regroupement à proximité maximale
  - Même bloc (temps d'accès 2ème donnée = 0)
  - Blocs consécutifs (temps d'accès 2ème donnée = Temps Transfert)
  - Blocs de la même piste (temps d'accès 2ème donnée = Temps Rotation)
  - Blocs du même cylindre (pas de déplacement des têtes)
  - Blocs sur cylindres différents : distance en nombre de pistes à parcourir

Optimisation et placement

- Séquencement
  - Adapter le regroupement aux accès concurrents
  - Exemple: lecture fichier F1, lecture F2 → alternance lecture F1 et F2 stockage en bloc consécutif inefficace
  - →Séquencement des demandes en les triant par piste, par bloc au sein d'une piste…
  - Technique de l'ascenseur (horizontal): déplacement régulier des têtes de lecture du bord du disque vers l'axe puis de l'axe vers le bord
    - Favorise les demandes nombreuses (pistes adjacentes)
- Utilisation de buffers

Système d'exploitation et SGBD

- Pourquoi le SGBD n'utilise (ou n'utilisait) pas le gestionnaire de disque du SE ?
  - Portabilité
  - Grandes masses de données
    - Un fichier SE doit tenir sur un plateau, on n'enjambe pas plusieurs plateaux
  - Optimisation accès séquentiels et concurrents
    - Pas la problématique du SE
- →Les choses évoluent très vite ces dernières années au niveau des SE !
- Pour prédire le bon comportement d'un SGBD il faut connaître
  - non seulement ce dernier, mais aussi, de manière détaillée, les caractéristiques du SE et du système de fichiers.

Utilisation du disque par le SGBD

- n-uplet (ou ligne) : enregistrement, séquence contigüe d'octets
- table = ensemble de lignes = fichier
- base = ensemble de tables = ensemble de fichiers
- Opérations sur les enregistrements
  - recherche d'un enregistrement (SELECT)
  - ajout d'un enregistrement à une table (INSERT)
  - mise à jour d'un enregistrement (UPDATE)
  - suppression d'un enregistrement (DELETE)
- Les fichiers (i.e. les tables) sont paginées (découpées en pages)

Représentation des enregistrements fixes

- Nombre fixe d'attributs (schéma de la table)
  - Par exemple 5 attributs : C1 à C5
- Champs de taille fixe (INTEGER (32 bits), CHAR…)

  C1 C2 C3 C4 C5 (attributs)

  B L1 L2 L3 L4 L5 (longueurs des attributs)
- Pour accéder au ième champ d'un enregistrement en connaissant l'adresse de base B, on ajoute à B les longueurs des champs 0, 1, …, i-1.
- Adresse de C4 = B + L1 + L2 + L3

Représentation des enregistrements variables

- Nombre fixe d'attributs (schéma de la table)
- Champs de tailles variables (blobs de texte)
- C1 $ C2 $ C3 $ C4 $ C5
- On utilise un séparateur spécial entre les champs
- On doit faire un scan linéaire pour arriver au ième champ

Meta-données

- Chaque enregistrement possède une en-tête stockant des données auxiliaires
  - taille totale de l'enregistrement (avec en-tête)
  - pointeur vers le schéma de la table (pour déterminer les tailles des champs)
  - date de dernière mise à jour
  - information de gestion des valeurs nulles
    - Stockage d'une valeur spéciale (pas toujours possible)
    - Stockage d'un bitmap (masque)

Stockage des enregistrements dans une page

- Chaque enregistrement a une adresse (rid : record id) constituée de
  - l'adresse de la page
  - de la position de l'enregistrement au sein de la page
- Lors d'une insertion, on doit trouver un emplacement libre dans la page
- Lors d'une suppression, on doit « effacer » un emplacement occupé et éventuellement *compacter* la page

Mises à jour (taille variable)

- Stratégies différentes selon les SGDBs
- Exemple
  - Insert T(3, 12, 'cou', NULL) → bloc F1.12, emplacement 46
  - Insert T(4, 178, 'go', NULL) → bloc F1.12, emplacement 47
  - Update T(3, 12, 'coucou blablablablabla', NULL) → agrandissement de 46
  - Update T(3, 12, 'coucou blablablablabla', 'un super long texte') → plus de place en 46…
  - Solution 1 : déplacer l'enregistrement 46 entièrement (Oracle)
  - Solution 2 : fragmenter la ligne en stockant la nouvelle information dans un autre bloc, avec un chaînage au niveau de l'enregistrement (mySQL)
- Avec MySQL le coût d'une lecture d'un enregistrement en *n* parties = *n* fois le coût d'un enregistrement compact.
- Oracle réserve un espace disponible dans chaque bloc pour l'agrandissement des enregistrements afin d'éviter ces réorganisations.

Technologies RAID

(…pour en savoir plus !)

- RAID : Redundant Array of Independent Disk
  - Dupliquer les données sur n disques
  - Défaillance d'un disque → disponibilité maintenue
- RAID 1 (miroir)
  - Toutes les E/S s'effectuent en parallèle sur les deux disques
  - Ecritures non simultanées (pour prévenir le cas de panne)
  - Optimisation : demande de lecture transmise au disque dont la tête de lecture est la plus proche de la piste contenant le bloc (la page) à lire

Répartition et Parité (RAID 4)

- Combine 2 techniques
- Traiter n disques en parallèle
  - L'unité est le bloc
  - Une donnée de 5 blocs est placée : 1 bloc sur disque 1, bloc suivant sur disque 2…., bloc suivant sur disque 4, dernier bloc sur disque 5

  →Pour de très larges volumes en moyenne : temps de Lecture et débit divisé par n !
- Gestion intelligente de la redondance

Gestion redondance en RAID 4

- Pour n disques de données
  - Ajout d'un **disque de contrôle (DC)** en cas de pb sur 1 disque
  - n+1 disques même taille et même structure
  - A chaque bit du DC, on associe les n bits des disques de données
    - bit de parité = p mod 2 (p = le nb de 1)
  - Exemple : 3 disques de données, supposons que le 1er octet soit

D1	1 1 1 1 0 0 0 0
D2	1 0 1 0 1 0 1 0
D3	0 0 1 1 0 0 1 1
DC	0 1 1 0 1 0 0 1

Exemple : D2 en panne : il faut affecter des 0 et 1 pour rétablir la parité du nombre de 1 attendue

→ Technique très économique en espace

→ Goulot d'étranglement = DC → RAID 5

→ Cas où plusieurs disques sont défaillants → RAID 6