Bases de données

Polytech Paris-Sud

Apprentis 4ème année

Cours 2 : Stockage

kn@lri.fr
http://www.lri.fr/~kn

Plan

Où stocker les données ?

Hierarchie mémoire :

Type	Accès	Taille max	Coût	durée de vie
Registre	< 1ns	128 bits	Très cher	alimenté
Cache L1/2/3	~ 10 ns	1ko ~ 1Mo	Très cher	alimenté
RAM	~ 50 ns	10 Go	Cher	alimenté
Disque SSD	~ 0.1ms	1 To	- Cher	Écritures limitées
Disque dur	~ 5ms	10 To	Peu cher	Fragiles
Bandes	10 s	1 Po/Eo	Donné	Bonne

- RAM : mémoire primaire
- Disques : mémoire secondaire
- Bandes magnétiques/Disques optiques : mémoire tertiaire

Quels types de mémoire pour une BD ?

On attend en général d'une BD:

- Stockage d'un nombre important de données
- Interrogation rapide (et si possible, mise à jour rapide aussi)
- Resistance aux pannes, aux corruptions
- ...

Celà implique l'utilisation de mémoire primaire (comme toutes les applications) et secondaire

Pas d'adressage direct du disque par le processeur, nécessité de « monter » les données en RAM

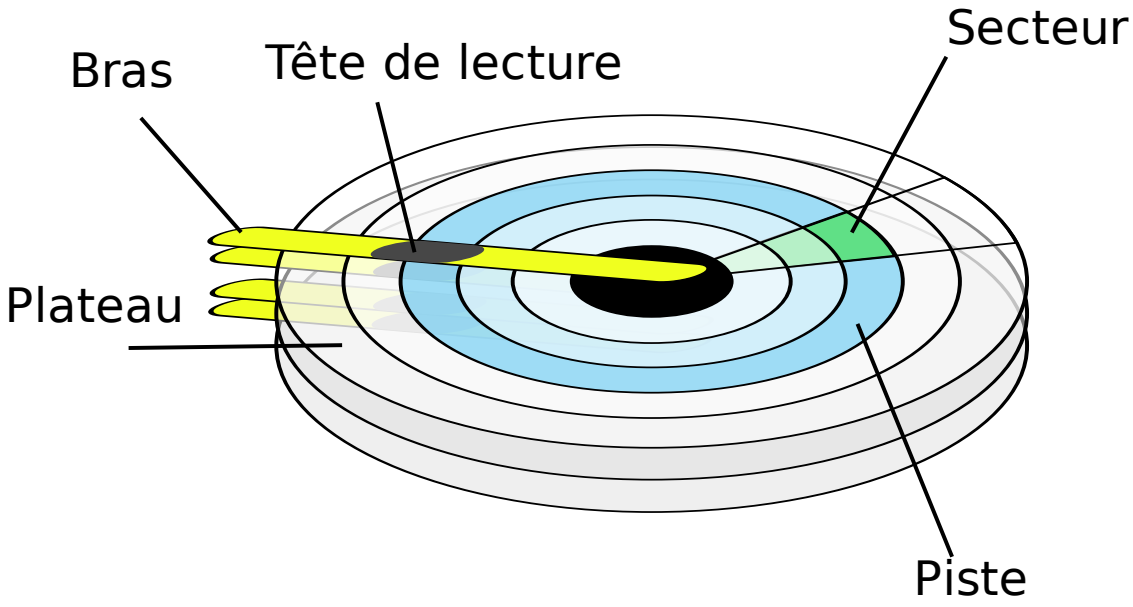
Goulet d'étranglement : facteur 10 000 (SSD) ~ 100 000 (HDD) entre mémoire et disques

Priorité des SGBD (ce qu'on présente dans ce cours) : limiter les accès disque

Plan

Caractéristiques physiques de disques

Disques rotatifs:



- **Chaque plateau a deux faces**
- **Un plateau est composé de pistes concentriques**
- **Les pistes sont décomposées en secteurs**
- **Une tête de lecture/écriture travaille secteur par secteur**
- **Un secteur fait plusieurs octets (typiquement 512)**
- **Un *cylindre* est l'ensemble des pistes situées à la même positions sur tous les plateaux**

Temps d'accès à un 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

Unité de transfert

On appelle page (ou bloc) la quantité de mémoire que le disque dur transfert de manière atomique en mémoire. En pratique 512o ~ 4ko (dépendant des disques). Une page fait au moins un secteur. De plus, une page ne peut pas déborder sur plus d'une piste.

On doit lire/écrire au moins une page même si on ne veut lire/écrire qu'un octet

Exemple : on souhaite stocker 5 chaines de caractères de 200o chacunes sur le disque

- **Si placés sur blocs différents (arbitraires) : 5 pages montées en mémoire,** `5 * temps de latence`
- **Si placés sur deux blocs consécutifs :** `temps de latence + temps de transfert` **(on gagne un facteur 5)**

Stratégies de placement

On peut placer en priorité les données « reliées » (qu'on veut utiliser en même temps):

1. Dans le même bloc (T.A. à la deuxième donnée: 0)
2. Sur deux blocs contigus (T.A. à la deuxième donnée: T. transfert)
3. Bloc à la même position sur un autre plateau (T.A. à la deuxième donnée: T. transfert)
4. Sur la même piste (T.A. à la deuxième donnée: T. rotation)
5. Sur des pistes proches

Mise en place de la stratégies de placement

Pour mettre en place les stratégies de placement il faut connaître:

- La liste des blocs libres et occupés
- La liste des blocs occupés par un « fichier » (i.e. qui stocke des informations reliées à la même donnée logique)

On ne travaille pas page à page : on alloue un buffer de pages

P1	P132	P10	P99
P2	P1000	P507	

On maintient pour chaque page : un compteur d'utilisation et un dirty-bit qui indique si la page a été modifiée (on doit donc l'écrire sur le disque à un moment donné).
Lorsque le buffer est plein, il faut supprimer des pages (et en monter d'autres en mémoire) suivant une stratégie: LRU, MRU, Random, FIFO, LIFO, ...

Qui gère les accès disques bas-niveau, le buffer, ... ?

- Dans le temps, le SGBD directement
- De nos jours, le système d'exploitation via ses systèmes de fichiers et gestion de la mémoire virtuelle (*swap* ou fichier d'echange)

Les systèmes d'exploitation ont *énormément* progressé (les SGBD ne pouvaient pas reposer sur quelque chose d'aussi primitif que FAT ou NTFS, mais les systèmes de fichier modernes sont plus performant que le traitement natif du disque fais par les SGBD, en particulier pour les SSD).
Pour prédire le bon comportement d'un SGBD il faut connaître non seulement ce dernier, **mais aussi**, de manière détaillée, les caractéristiques de l'OS et du système de fichier.

Un mot sur les SSD

Solid State Disk : « disque » dur composé entièrement de mémoire flash. Avantages:

- Pas de temps de recherche et de rotation, uniquement temps de transfert. Accès arbitraires aussi rapides que les séquentiels
- Consomation électrique moindre, poid moindre, pas de pièce mobile (resistance aux chocs)

- Couteux
- Écriture très complexe (écrire dans une cellule impose de l'effacer avant, nombre de cycle d'effaçage limité, etc)
- Nécessite une coopération à tous les niveaux, seul le système d'exploitation peut bien le gérer (bibliothèque système, gestion de la mémoire virtuelle, système de fichier, pilote du disque et *firmware*).

Plan

Utilisation du disque par les SGBD

- n-uplet (ou ligne) : enregistrement, séquence contiguë d'octets
- table = ensemble de lignes = fichier
- base = ensemble de tables = ensemble de fichiers

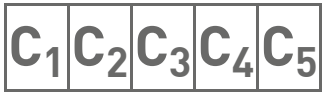
Opérations de base sur les enregistrements

- 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)**
- **Champs de taille fixe (`VARCHAR[50]`, `INTEGER` (32 bits), ...)**



B L_1 L_2 L_3 L_4 L_5

Pour accéder au $i^{ème}$ champ d'un enregistrement en connaissant l'addresse de base `B`, on ajout à `B` les longueurs des champs 0, 1, ..., i-1.
adresse de C_4 = B + L_1 + L_2 + L_3

Représentation des enregistrements variables

- Nombre fixe d'attributs (schéma de la table)
- Champs de taille variables (*blobs* de texte)

C_1	\$	C_2	\$	C_3	\$	C_4	\$	C_5

On utilise un séparateur spécial entre les champs (scan linéaire pour arriver au $i^{\text{ème}}$ champ.

L_1	L_2	L_3	L_4	L_5	C_1	C_2	C_3	C_4	C_5

On stocke les longueurs dans l'enregistrement

Méta-données

Chaque enregistrement possède un en-tête stockant des données auxiliaires

- taille totale de l'enregistrement (avec entête)
- 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) : $26_{10} = 11010_2$: valeur nulle dans les champ 0 et 2

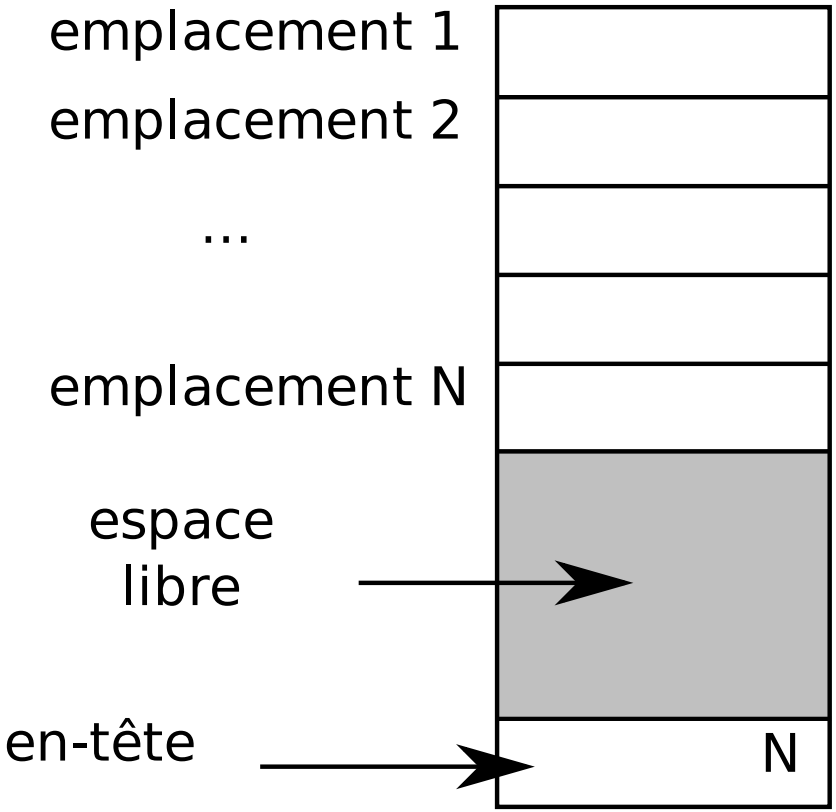
Stockage des enregistrements dans une page

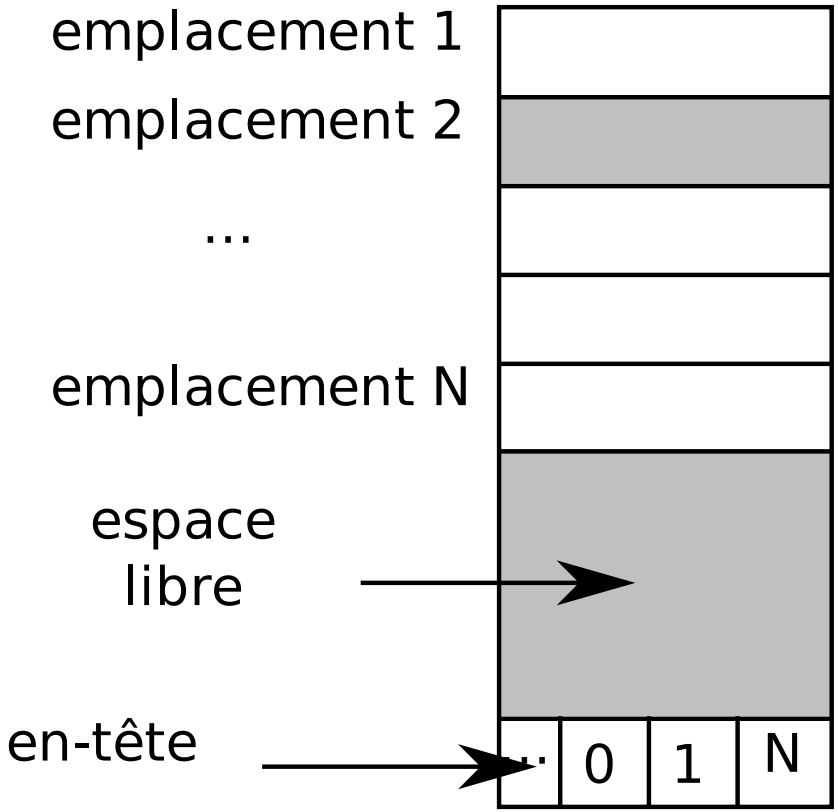
Chaque enregistrement à une adresse (*rid* : record id) constituée de l'adresse de la page et de la position de l'enregistrement au sein de la page

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

Stockage compact vs non compact (taille fixe)





Stockage compact (taille variable)