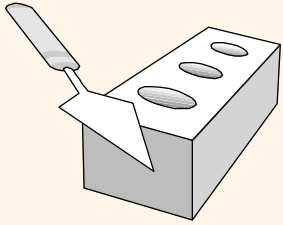


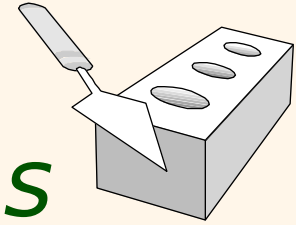
Bases de Données Avancées

Ioana Ileana

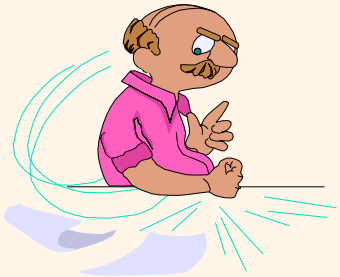
Université Paris Descartes

Cours basé sur le livre (et les diapositives de):
Database Management Systems 3ed, R. Ramakrishnan et J. Gehrke

LIPADE
Laboratoire d'Informatique PAris DEscartes

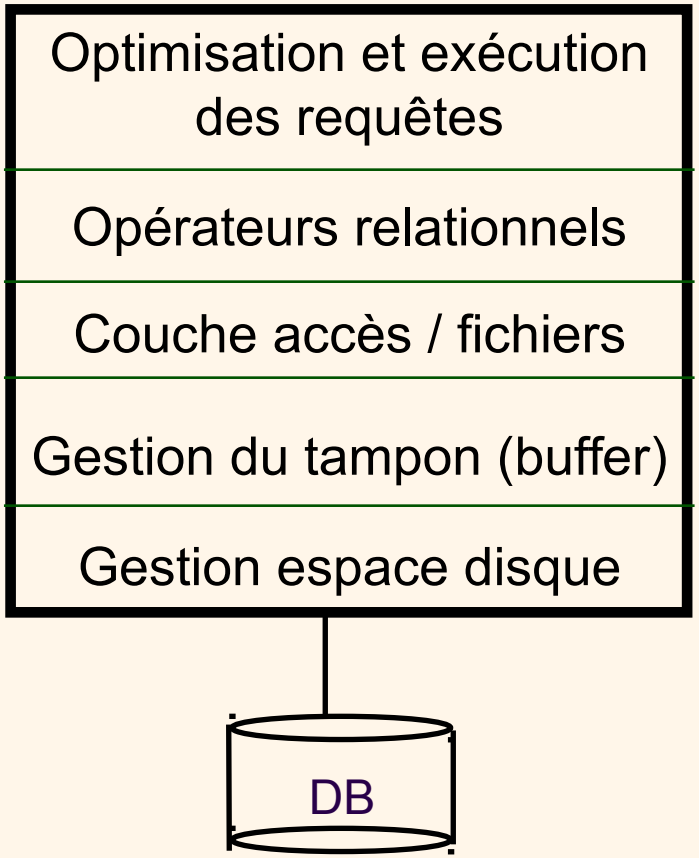


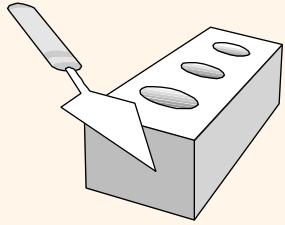
Cours 3: Stockage des données (2ème partie)



- Diapos traduites et adaptées du matériel fourni en complément du livre Database Management Systems 3ed, par Ramakrishnan et Gehrke ; un grand merci aux auteurs pour la réalisation et la disponibilité de ce matériel !
- Les diapos originales (en anglais) sont disponibles ici : http://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/slides/slides3ed.html
- Plus particulièrement, ce cours touche aux éléments dans le Chapitre 9 du livre ci-dessus
- Également, merci à Themis Palpanas pour ses adaptations de diapos en partie reprises dans ce cours !

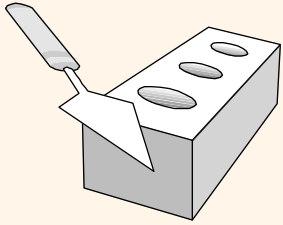
Rappels: structure d'un DBMS



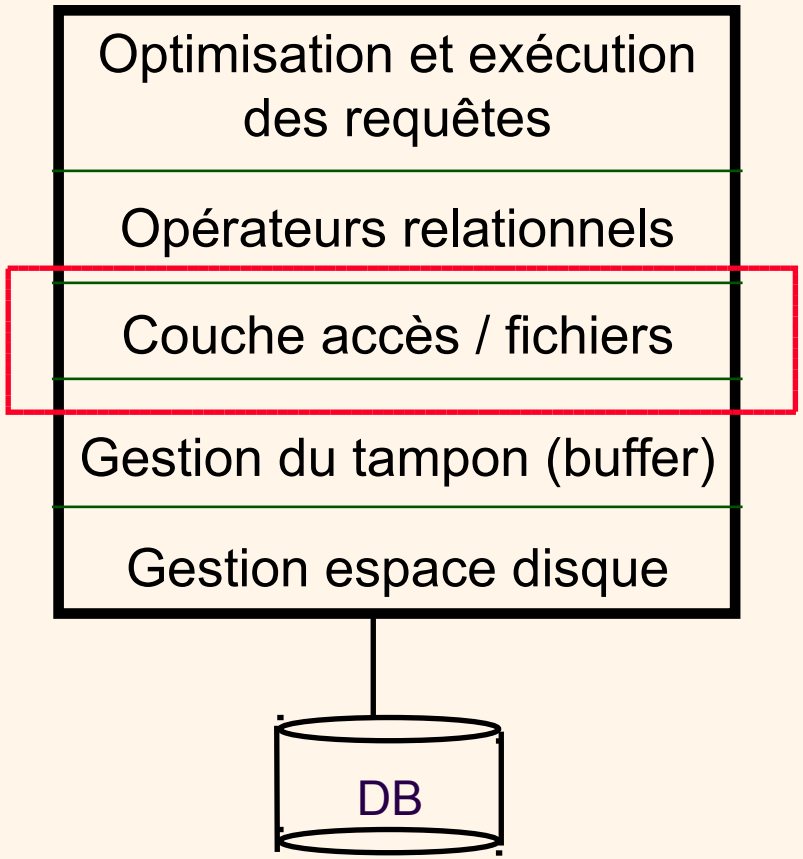


Rappels : le gestionnaire disque et le buffer manager

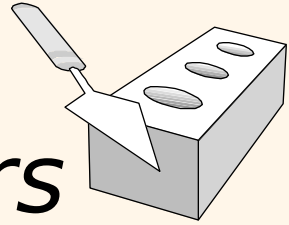
- Unité de transfert / stockage de données = une page (séquence d'octets dont la taille correspond à un bloc disque!)
- Le gestionnaire disque s'occupe de la gestion de l'espace disque ; les couches plus hautes font des appels à ce gestionnaire pour:
  - Allouer / désallouer une page
  - Lire / écrire une page
- Le buffer manager offre aux couches plus hautes une vue « tout en RAM » ; il fait appel au gestionnaire disque pour
  - Lire des pages depuis le disque vers la RAM
  - Ecrire des pages depuis la RAM vers le disque
- Important : au niveau du gestionnaire disque et du buffer manager, l'abstraction reste celle de la page ; aucun des deux ne s'intéresse « à ce qu'on trouve dans les pages », ni à comment les pages sont « regroupées ensemble » pour stocker une relation / table.



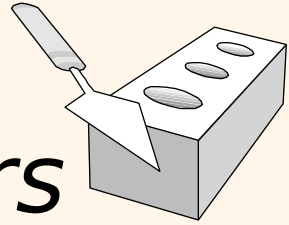
La couche accès / fichiers



Fichiers et couche accès / fichiers



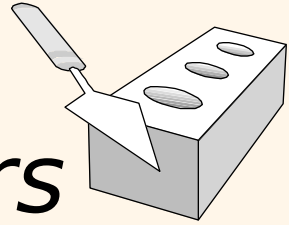
- L'abstraction de base pour les données dans un SGBD est le *fichier de records (ou fichier tout court)* = une collection de records (enregistrements, tuples)
  - Utilisé pour le contenu d'une relation
  - Attention : dans la suite de ce cours, nous allons utiliser le terme fichier dans son sens SGBD !
- Pour optimiser les accès disque, les records dans un fichier sont groupés sur des *pages*
  - *On peut donc voir un fichier de records comme une collection de pages, chaque page contenant une collection de records !*



Fichiers et couche accès / fichiers

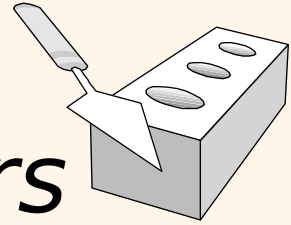
- La couche accès / fichiers assure « le suivi » des pages dans un fichier, ainsi que de l'espace disponible dans ces pages (et son évolution suite aux insertions / suppressions de records)
- L'accès au contenu persisté (= stocké sur le disque) d'une page, pour lecture ou modification, est fait via le Buffer Manager (qui « remonte en RAM » si besoin la page en question)
- La gestion de l'espace disque est faite par la couche la plus basse – le gestionnaire de l'espace disque
  - La couche accès / fichiers va demander à ce gestionnaire l'allocation d'une nouvelle page s'il n'y a plus d'espace disponible sur les pages courantes
  - Elle va également lui signaler quand « il n'y a plus besoin d'une page », pour que l'espace disque correspondant puisse être réutilisé.

Fichiers et couche accès / fichiers

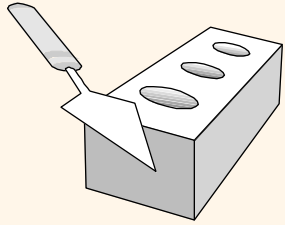


- Fichier = collection de pages, chaque page contenant une collection de records
- Chaque record dans un fichier a un identifiant unique – le *rid* (record id)
- Le *rid* doit permettre d'identifier également la page sur laquelle le record se trouve (de retrouver directement le PageId de cette page!)
- Au niveau d'un fichier de records, on doit pouvoir effectuer (au minimum):
  - l'insertion / l'effacement / la modification d'un record
  - l'accès « direct » à un record spécifié par son *rid*
  - le scan (parcours) de tous les records (potentiellement avec des conditions sur les records d'intérêt, par exemple *champ* = *valeur*)
- Ces opérations sont gérées différemment en fonction de la structure / organisation du fichier !

Fichiers et couche accès / fichiers

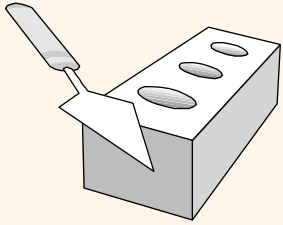


- L'*organisation / structure* d'un fichier de records (file organization) désigne tout simplement la manière d'organiser les records dans le fichier
- Il existe plein d'alternatives, chacune rendant certaines opérations très efficaces et d'autres plus (voire très) coûteuses :
  - Les Heap Files (fichiers non triés) :
    - Records stockées dans un ordre aléatoire dans les pages du fichier
    - Bien pour le scan ou l'accès à un record par son rid, mais moins pour des requêtes avec des conditions ou qui demandent des résultats triés
  - Les Sorted Files (fichiers triés) :
    - Quand un tri est demandé ou que la requête demande un « intervalle » de records (ex : étudiants dont la moyenne est comprise entre 8 et 10)
  - Les index : structures de données qui permettent d'organiser les records de manière à optimiser certaines opérations d'accès / requêtes
    - Nous les regarderons en détail dans la suite de ce cours !

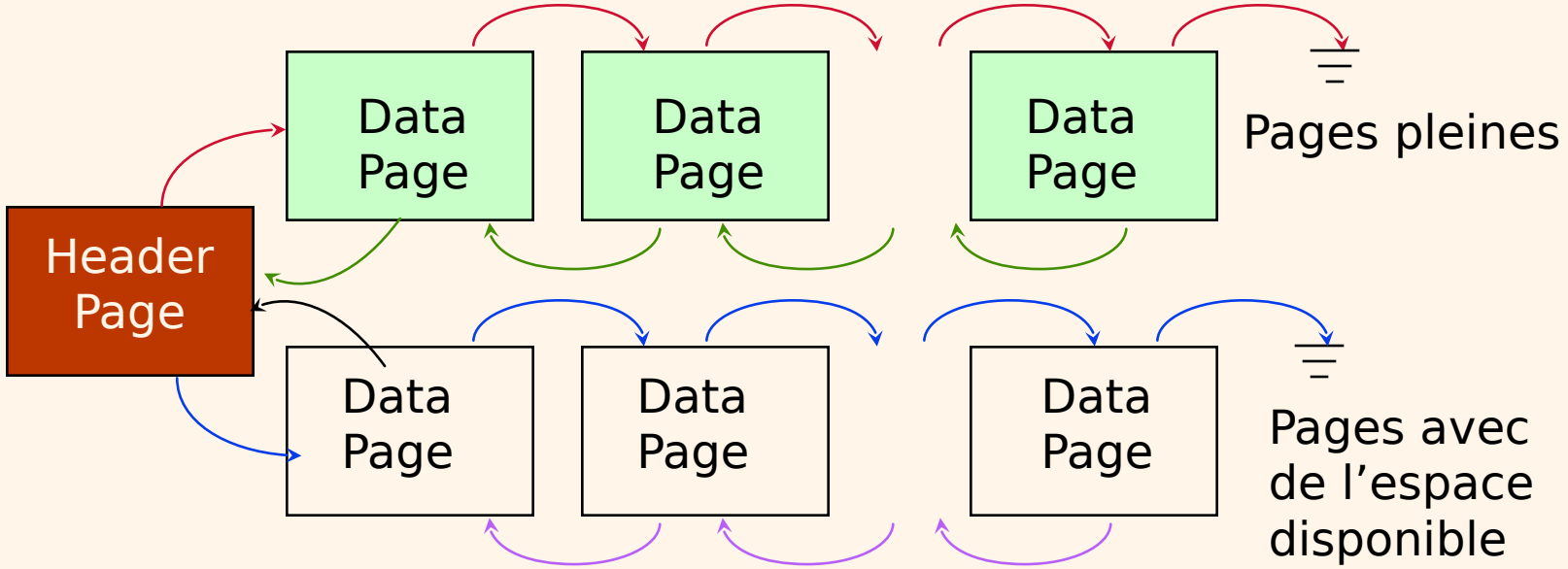


Heap Files

- La structure la plus simple ; un Heap File contient les records sans aucun ordre spécifique.
- Des pages disque sont allouées / désallouées au fur et à mesure que le fichier augmente ou diminue (suite à l'insertion ou suppression de records).
- Pour permettre des opérations sur les records, il faut gérer / suivre / garder la trace de:
  - Toutes les *pages* dans le fichier (autrement, comment peut-on parcourir tous les records?)
  - *L'espace libre* sur chaque page (autrement, comment sait-on où insérer un record?)
- Il existe de nombreuses approches pour ce type de gestion / suivi ; les plus standard sont par liste (doublement) chaînée et par Page Directory

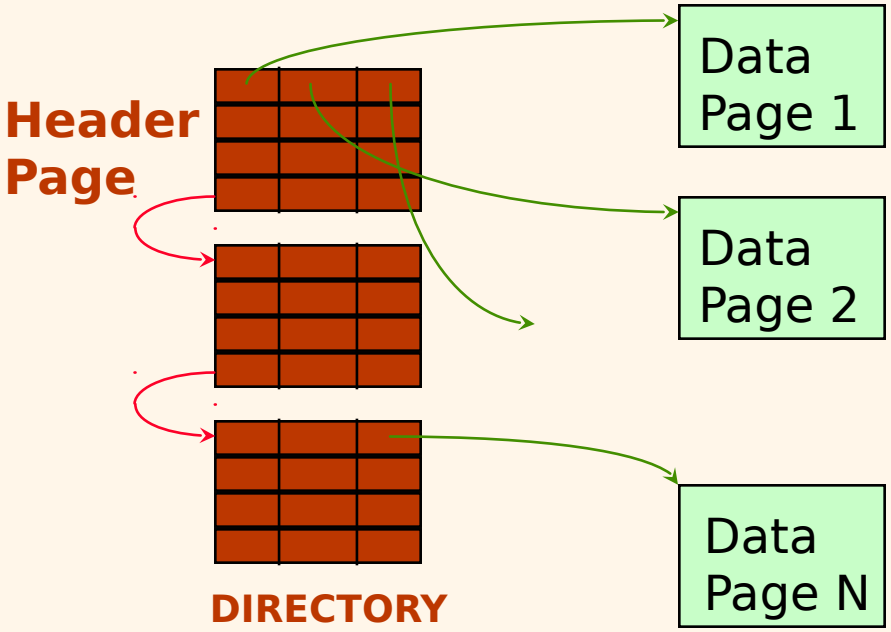


Heap File: suivi des pages par une liste (doublement) chaînée



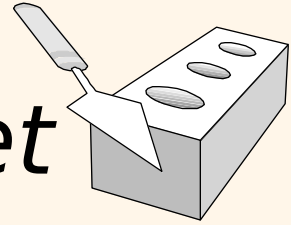
- Besoin de persister l'identifiant (PageId) de la page en-tête (Header Page)
- Chaque page de données (Data Page) contient deux pointeurs en plus des données de la page.

Heap File: suivi des pages par Page Directory



- L'entrée dans le directory qui correspond à une page peut inclure le nombre d'octets restant disponibles sur la page (ou simplement un flag indiquant « s'il reste de la place sur la page ») .
- Le directory est lui même une collection de pages, dont une des implémentations possibles peut être par une liste chaînée .
  - *Taille totale sensiblement inférieure par rapport à l'implémentation par liste chaînée vue précédemment !*

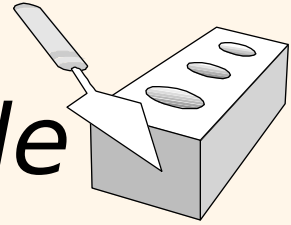
Structure des pages ; structure et placement des records



- Outre la gestion des pages, il est essentiel d'assurer *la gestion des records dans chaque page :* comment placer une liste de records sur une page *?*
- Le plus souvent, une page peut être vue comme une collection de slots (cases), chaque record étant placé dans un slot
- Cela nous fournit une manière très simple de construire les records ids (rids) :

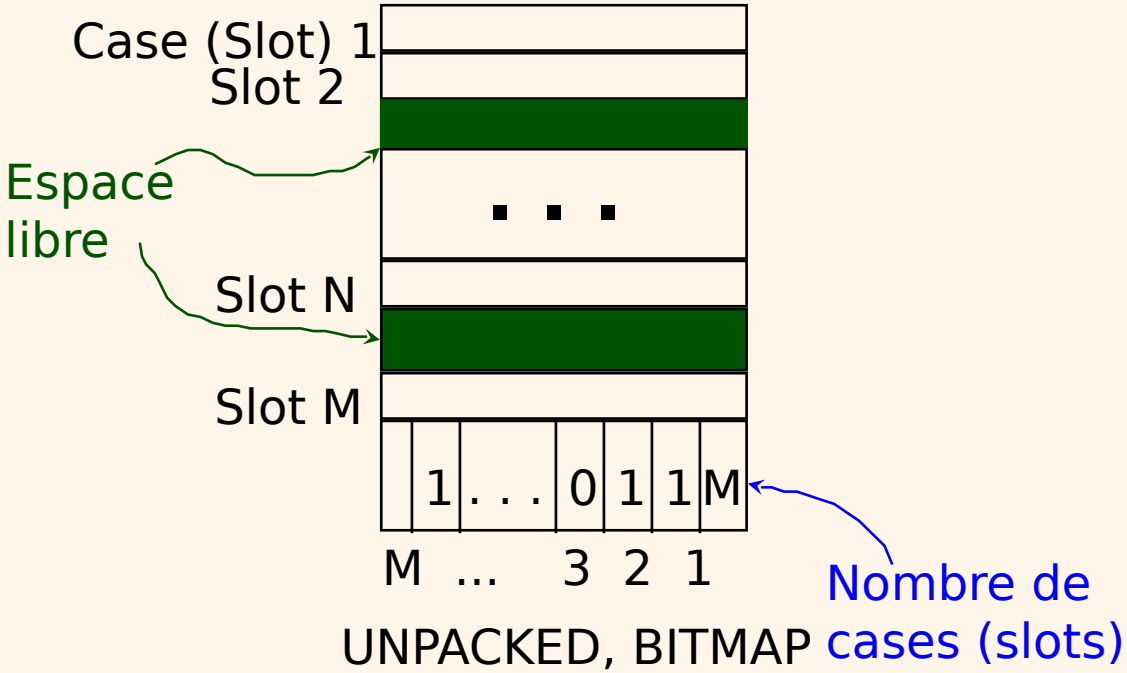
  Record id = <page id, slot #>
- La manière de placer les records sur la page va souvent fortement dépendre de la manière de représenter les records, plus particulièrement selon qu'on utilise *des records de longueur fixe ou des records de longueur variable*

Formats de page pour records de longueur fixe



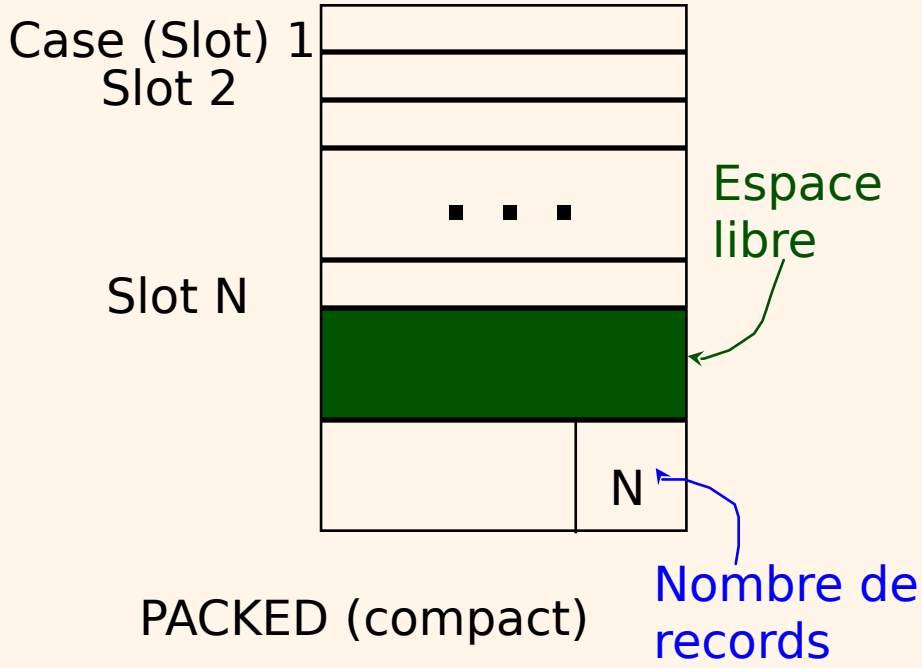
- Toutes les cases (slots) seront uniformes (même taille = taille d'un record) et pourront être disposées de manière consécutive sur la page
- Les deux façons les plus standard de stocker les records de taille fixe sur une page sont dans le format unpacked / bitmap et dans le format packed (compact)
  - l'insertion et la suppression des records seront gérées différemment en fonction de ces formats

Formats de page pour records de longueur fixe : unpacked / bitmap



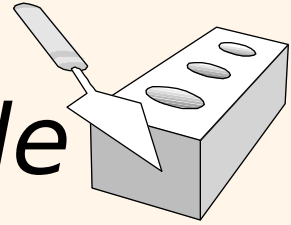
- *Insertion : chercher un slot libre (0 dans la bitmap)*
- *Suppression : remettre le bit correspondant à 0 !*

Formats de page pour records de longueur fixe : packed (compact)



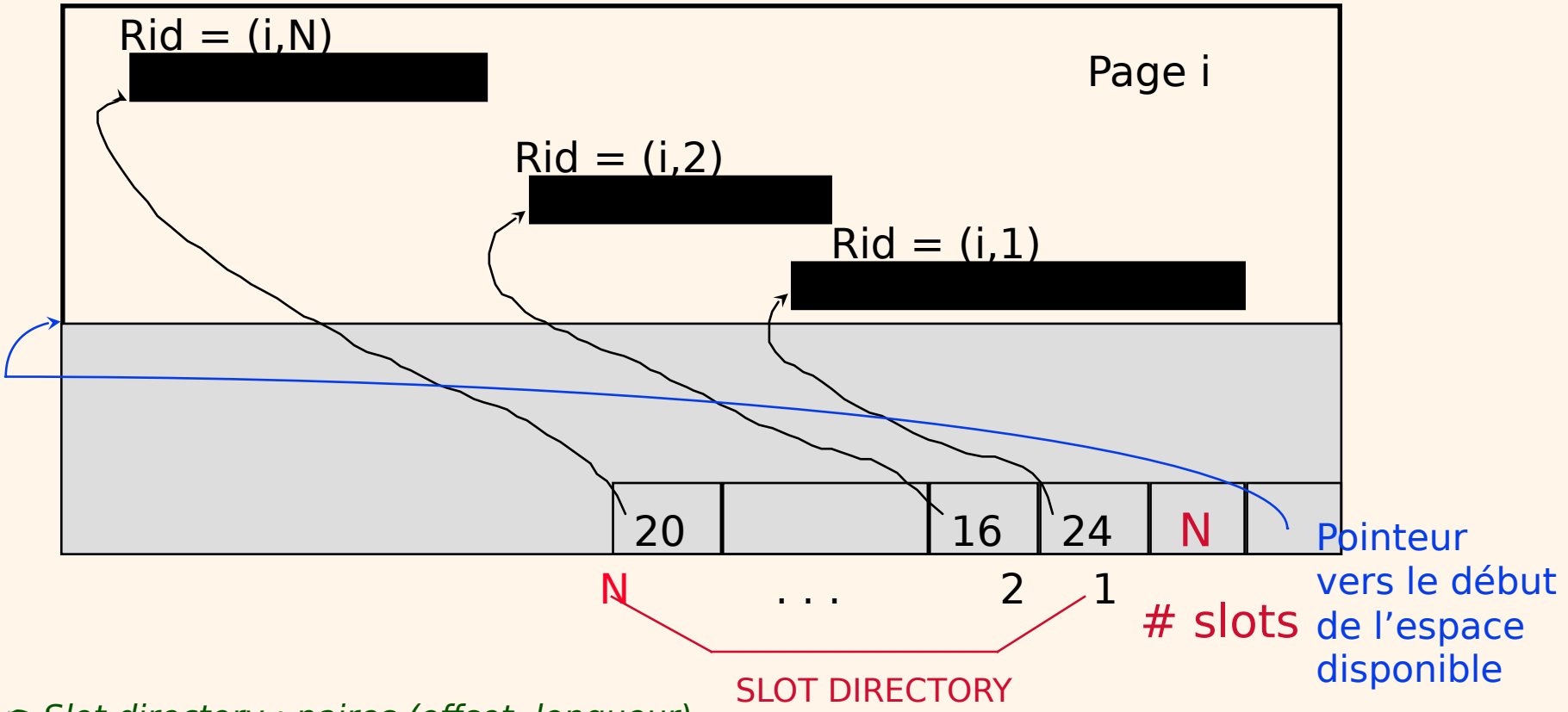
- *Insertion ? Suppression ? Avantages ?*
- *(Gros) désavantage : les records doivent être déplacés lors de la suppression, ce qui change le rid !*

Formats de page pour records de longueur variable



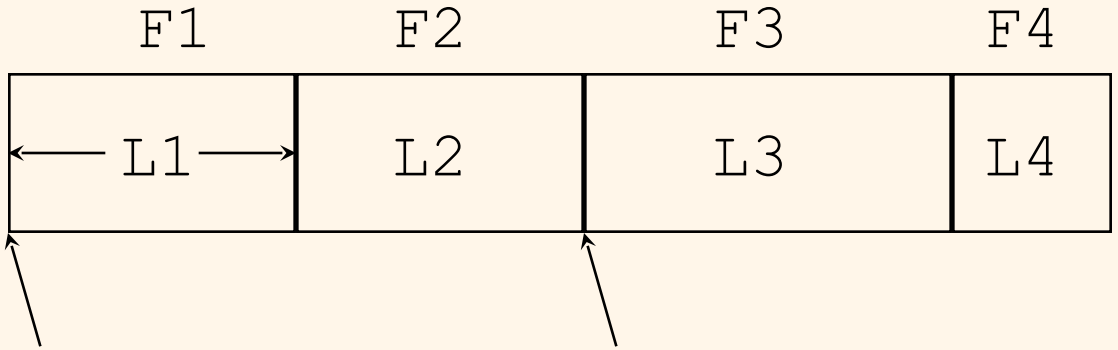
- Contrairement à la gestion des records de longueur fixe, nous ne pouvons pas diviser la page en cases de taille égale !
- → Quand on insère un record, il faut « trouver une case qui fait la bonne taille » - chose plutôt difficile, car souvent il y a ou bien « trop de place » ou bien « pas assez » !
- → Quand on supprime un record, on peut avoir besoin de « bouger tout le monde » pour garantir que l'espace libre qui reste sur la page est présent « en un seul bloc » !
  - Et ainsi faciliter les insertions !

Formats de page pour records de longueur variable: slot directory



- *Slot directory : paires (offset, longueur)*
- *Format qui permet de déplacer les records sur la page sans changer le rid; le slot directory est donc intéressant aussi pour les records de taille fixe !*
- *Q : peut-on « compacter » le slot directory après la suppression d'un record ?*

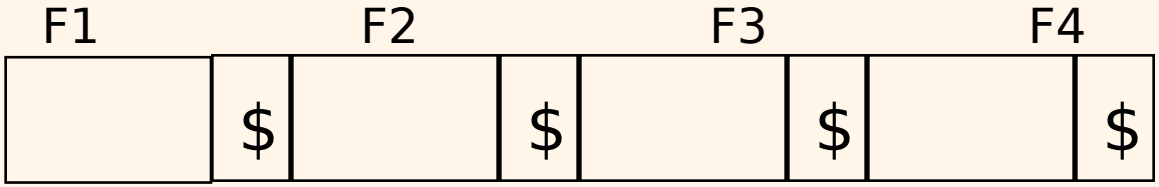
Formats de records: longueur fixe



- Information sur les types de champs: la même pour tous les records dans un fichier; stockée dans les *catalogues système (system catalogs).*
- Pour accéder au *ième* champ il n'y a pas besoin de parcourir tout le record.

Formats de records: longueur variable

- Deux formats alternatifs (#champs fixe):



Champs délimités par des symboles spéciaux

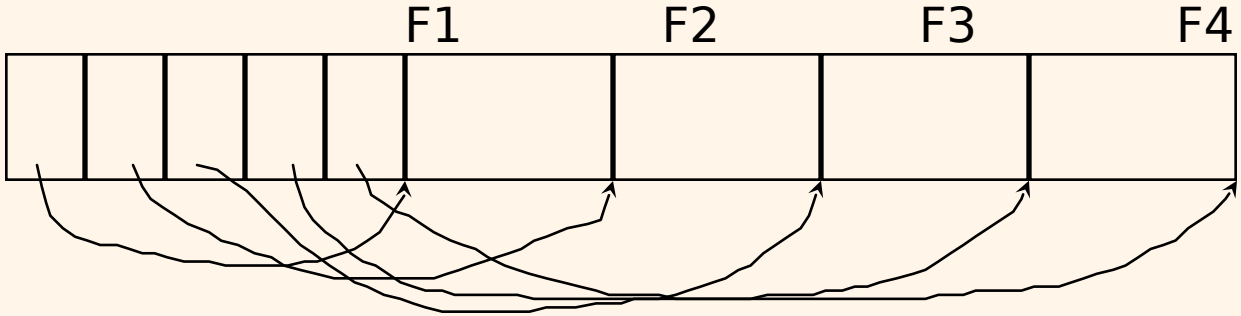
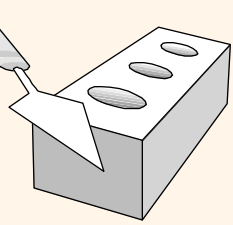


Tableau (directory) des offsets des champs

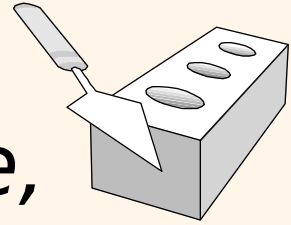
- Le deuxième offre un accès direct au ième champ ; le surcoût introduit par le tableau des offsets reste petit.

Les Catalogues Système (System Catalogs)



- Pour chaque relation:
  - Son nom, le nom du fichier qui y correspond, la structure du fichier (ex. Heap File)
  - Le nom et le type de chaque attribut (champ)
  - Le nom de chaque index associé
  - Contraintes d'intégrité
- + Infos spécifiques aux index et vues
- + Statistiques, autorisations, la taille du buffer pool, etc.

☛ *Les catalogues sont eux même stockées comme des relations.*



Attr_Cat(attr_name, rel_name, type, position)

attr_name	rel_name	type	position
attr_name	Attribute_Cat	string	1
rel_name	Attribute_Cat	string	2
type	Attribute_Cat	string	3
position	Attribute_Cat	integer	4
sid	Students	string	1
name	Students	string	2
login	Students	string	3
age	Students	integer	4
gpa	Students	real	5
fid	Faculty	string	1
fname	Faculty	string	2
sal	Faculty	real	3