Bases de Données Avancées : TP5

Pendant ce TP, vous allez travailler sur la couche Accès/Fichiers, notamment en enrichissant la classe **Relation**.

Nous allons nous concentrer sur les différentes briques nécessaires à la création d'une relation, à l'insertion de records, et à la sélection, pour pouvoir ainsi (très bientôt) traiter des commandes de type **CREATE TABLE, INSERT** et **SELECT**.

Prenez le temps de lire très attentivement la partie « Information » ci-dessous ; elle vous permettra de bien coder les différents éléments au niveau des Heap Files et de la gestion des records.

A. Information : Stockage des relations. A LIRE ATTENTIVEMENT !!!

Chaque relation dans votre base de données aura une structure de type Heap File associée (voir cours). Le suivi des pages de données sera par *Page Directory*, comme détaillé ci-dessous.

Dans les pages de données, nous utiliserons le format de page de données avec *Slot Directory*, comme décrit dans le cours et détaillé ci-dessous.

A1. La Header Page

Le contenu de la Header Page pour une relation sera le suivant :

- au tout début : un entier *N* (4 octets) correspondant au nombre de pages de données
- ensuite *N* « cases », décrivant chacune une page de données. Chaque « case » comprend :
  - un PageId (donc deux entiers, 4 octets chacun) correspondant à l'identifiant de la page de données
  - un entier (sur 4 octets) donnant le nombre d'octets disponible sur la page de données.

IMPORTANT : pour les besoins de ce projet, nous supposons qu'il n'y a pas besoin de chaîner plusieurs pages de directory. Toutefois, il y aura un scénario bonus / optionnel pour ceux qui souhaitent gérér ce chainage.

A2. Les pages de données (Data Pages)

Une page de données contient deux types d'information :

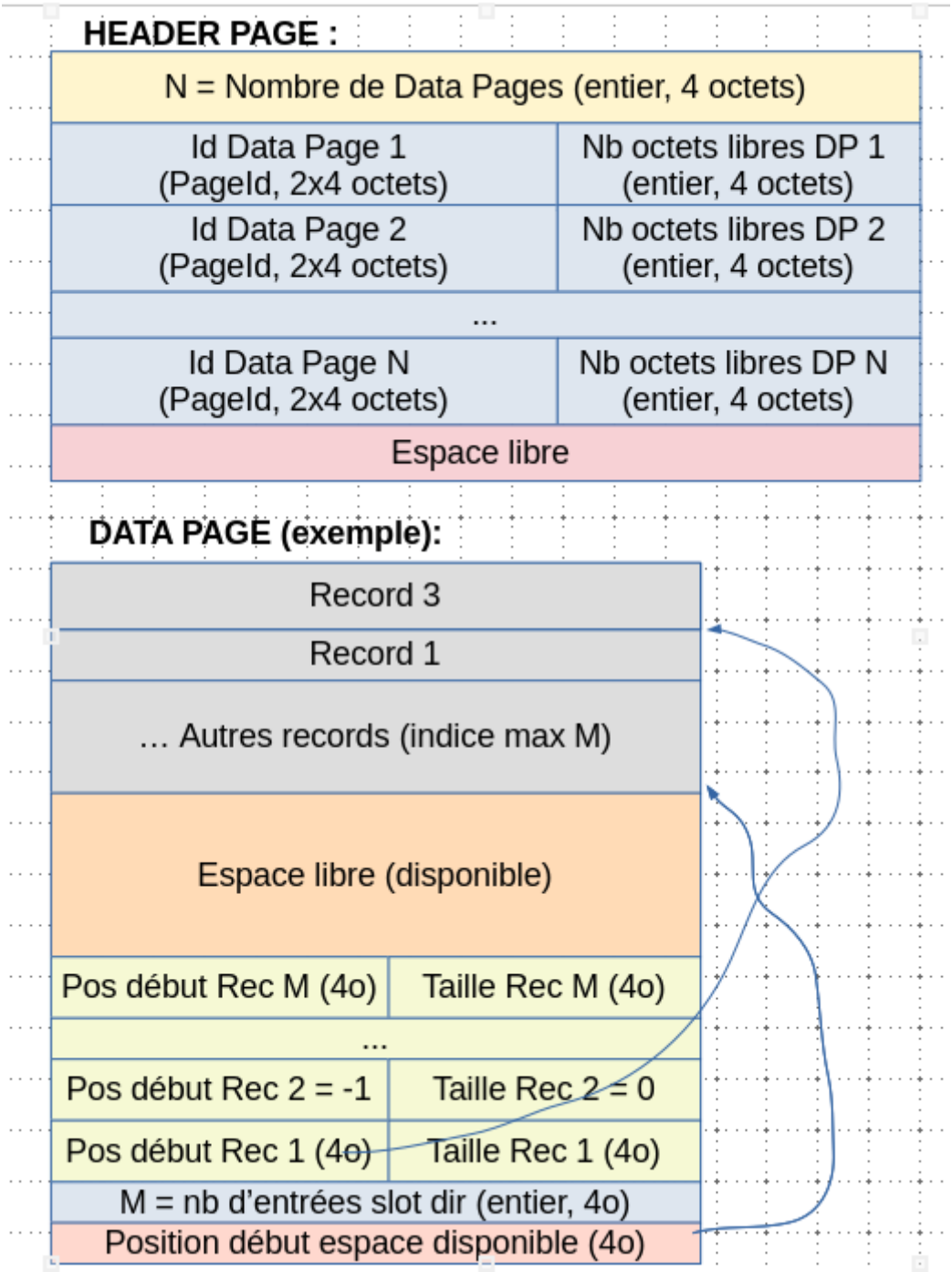
- Les records, écrits les uns à la suite des autres juste après le PageId
- Le slot directory, écrit « depuis la fin de la page » – voir aussi cours et schéma ci-dessous.

Le slot directory contient :

- un entier (4 octets) correspondant à la position (l'octet) à partir de laquelle commence l'espace libre sur la page (=là où on peut rajouter des records)
- un entier *M* (4 octets) correspondant au nombre de cases dans le directory
- *M* « cases », où la case i contient
  - un entier (4 octets) donnant la position (l'octet) de début dans la page du i-ème record
  - un entier (4 octets) donnant la taille (en octets) du i-ème record.

IMPORTANT : la taille marquée dans une « case » peut être 0 si le record a été supprimé !

Les schémas ci-dessous montrent le contenu de la Header Page et d'une Data Page



B. Code : La classe RecordId

Créez une classe **RecordId** qui correspond au Rid (Record Id, donc identifiant) d'un enregistrement.
Votre classe contiendra deux variables membres :

- *pageId*, un **PageId** qui indique la page à laquelle appartient le record
- *slotIdx*, un entier qui est l'indice de la case du slot directory qui pointe vers le record.

Libre à vous de gérer le constructeur etc.

C. Code : enrichissement de la classe Relation

C1. Relation : rajout de variables membres

Rajoutez, dans la classe **Relation :**

- une variable membre *headerPageId*, de type **PageId**. Cette variable représente l'identifiant de la Header Page de la relation ; nous avons besoin de maintenir cet identifiant et de le persister pour pouvoir accéder à tout moment au contenu d'une relation
- des variables membres type pointeur / référence vers des instances de **DiskManager** et **BufferManager**.

Modifiez le constructeur en conséquence.

C2. Relation: rajout d'une page de données

Rajoutez, dans la classe **Relation**, une méthode
void **addDataPage ()**

Cette méthode devra rajouter une page de données « vide » au Heap File correspondant à la relation.

Pour cela, elle devra :

- allouer une nouvelle page via **AllocPage** du **DiskManager**
- *actualiser le Page Directory en prenant en compte cette page*

C3. Relation: trouver une page « avec de l'espace disponible » pour une insertion

Rajoutez, dans la classe **Relation**, une méthode
pageId **getFreeDataPageId** (*sizeRecord*)**,** avec *pageId* un **PageId** et *sizeRecord* un entier correspondant à la taille du record à insérer.

Cette méthode doit retourner le **PageId** d'une page de données sur laquelle il reste assez de place pour insérer le record ; si une telle page n'existe pas, la méthode retournera *null*.

C4. Relation: écrire un Record dans une page de données

Rajoutez, dans la classe **Relation**, une méthode
rid **writeRecordToDataPage** (*record, pageId*), avec *record* un **Record,** *pageId* un **PageId** et *rid* un **RecordId**.

Cette méthode doit écrire l'enregistrement *record* dans la page de données identifiée par *pageId,* et renvoyer son **RecordId** ! Utilisez une des méthodes du TP4 pour écrire le **Record**.

IMPORTANT : Nous supposons que la page dispose d'assez d'espace disponible pour l'insertion (pas besoin de vérifier!)

C5. Relation: énumérer les Records d'une page de données

Rajoutez, dans la classe **Relation**, une méthode
listeDeRecords **getRecordsInDataPage** (*pageId*), avec *pageId* un **PageId** et *listeDeRecords* une liste de **Record**.

En guise de « liste », vous pouvez utiliser toute structure qui actualise sa taille automatiquement lors des rajouts d'éléments (par exemple ArrayList en Java).

Cette méthode doit renvoyer la liste des records stockés dans la page identifiée par *pageId*.
Utilisez une des méthodes de la classe **Record** du TP4.

Attention, n'oubliez pas de libérer la page après lecture !

C6. FileManager: énumérer toutes les pages de données

Rajoutez, dans la classe **Relation**, une méthode
listeDePageIds **getDataPages** (),

En guise de « liste », vous pouvez utiliser toute structure qui actualise sa taille automatiquement lors des rajouts d'éléments (par exemple ArrayList en Java).

Cette méthode doit renvoyer la liste des PageIds des pages de données, tels qu'ils figurent dans la Header Page.

Attention, n'oubliez pas de libérer la page après lecture !

C7. Code : « API » / méthodes publiques d'insertion et sélection de la classe Relation

Rajoutez dans la classe **Relation** les méthodes décrites ci-dessous :

- Insertion d'un Record :
  rid **InsertRecord (***record***)**, avec *record* un **Record,** et *rid* un **RecordId**.
- Lister tous les records :
  listeDeRecords **GetAllRecords ()**, avec *listeDeRecords* une liste ou tableau de **Record**.