

Théorie des Langages

TD

Bases de Données

Elana Courtines

courtines.e@gmail.com

<https://github.com/irinacake>

Séance 1 - 14 novembre 2022

- [name@domain](#)

Rappels

Opérations sur les fichiers (la syntaxe)

(1) Assignment : `assignation (<nom_variable_logique>, <fichier_physique>, organisation [, attribut])`
Si l'organisation est séquentielle indexée, on doit indiquer le nom d'attribut sur lequel le fichier est indexé.

(2) Ouverture : `ouverture (<nom_variable_logique>, <lecture/ecriture>, <mode_accès>)`
Mode_accès peut être séquentiel ou direct (utilisant l'index).

(3) Lecture : `lecture (<nom_variable_logique>, <variable_article> [, clé])`
Si l'accès est direct, on doit indiquer la valeur de la clé.

(4) Fin de fichier : `fdf (<nom_variable_logique>)`
La fonction retourne un booléen : vrai pour fin de fichier.

(5) Fermeture : `fermeture (<nom_variable_logique>)`

8

Énoncé 1

- Soit 2 fichiers `personne.txt` et `voiture.txt` qui contiennent les informations suivantes, respectivement :
 - `personne.txt` : nom, adresse
 - `voiture.txt` : num.im, marque, proprio (le nom de la personne)
- Hypothèse :
 - L'organisation du fichier est séquentielle

Note : parce que je trouve leur pseudo-code insupportable, j'ai opté pour un pseudo-code un peu plus proche du Python/C :)

Exercice 1

Écrivez un programme (pseudo-code) pour trouver la marque de la voiture 31xx31.

```
1 enregistrement voiture {
    num_im : number(4)
3    marque : varchar(20)
    proprietaire : varchar(20)
5 } v

7 assignation(fv, "C:\...\voiture.txt", sequentielle)

9 ouverture(fv, lecture , sequentiel)

11 while (!FDF(fv)):
    lecture(fv,v)
13     if (v.num_im == '31xx31'):
        afficher(v.marque)
15
fermeture(fv)
```

Cette opération est équivalente à un Select en SQL.

Exercice 2

Écrivez un programme (pseudo-code) pour trouver les adresses des propriétaires des voitures de marque "Renault".

```
enregistrement voiture {
2    num_im : varchar(4)
    marque : varchar(20)
4    proprietaire : varchar(20)
} v
6 enregistrement personne {
    nom : varchar(20)
8    adresse : varchar(20)
} p
10
assignation(fv, "C:\...\voiture.txt", sequentielle)
12 assignation(fp, "C:\...\personne.txt", sequentielle)

14 ouverture(fv, lecture , sequentiel)

16 while (!FDF(fv)):
    lecture(fv,v)
18     if (v.marque == "Renault"):
        ouverture(fp, lecture , sequentiel)
20         while (!FDF(fp)):
            lecture(fp,p)
22             if (v.proprietaire == p.nom):
                afficher(p.adresse)
24             fermeture(fp)

26 fermeture(fv)
```

Cette opération est équivalente à une Jointure en SQL.

Cf. fichiers source pour une version "avec moins de complexité".

Algorithme de sélection - Énoncé 2

Abonnement Téléphonique

- AB-T (nom, prenom, prof, adr, tel)
- Estimez le nombre de pages disques lues pour la question : numéros de téléphone de Monsieur Dupont :
 - 1) Sans Index
 - 2) Avec Index

Donnée	taille
Nom	20c
adresse	38c
prenom	18c
tel	12c
prof	12c
<i>Total</i>	100c

Adresse relative	5c
------------------	----

- Nombre de tuples : 200 000
- 1 page = 1000 caractères

1) Sans Index :

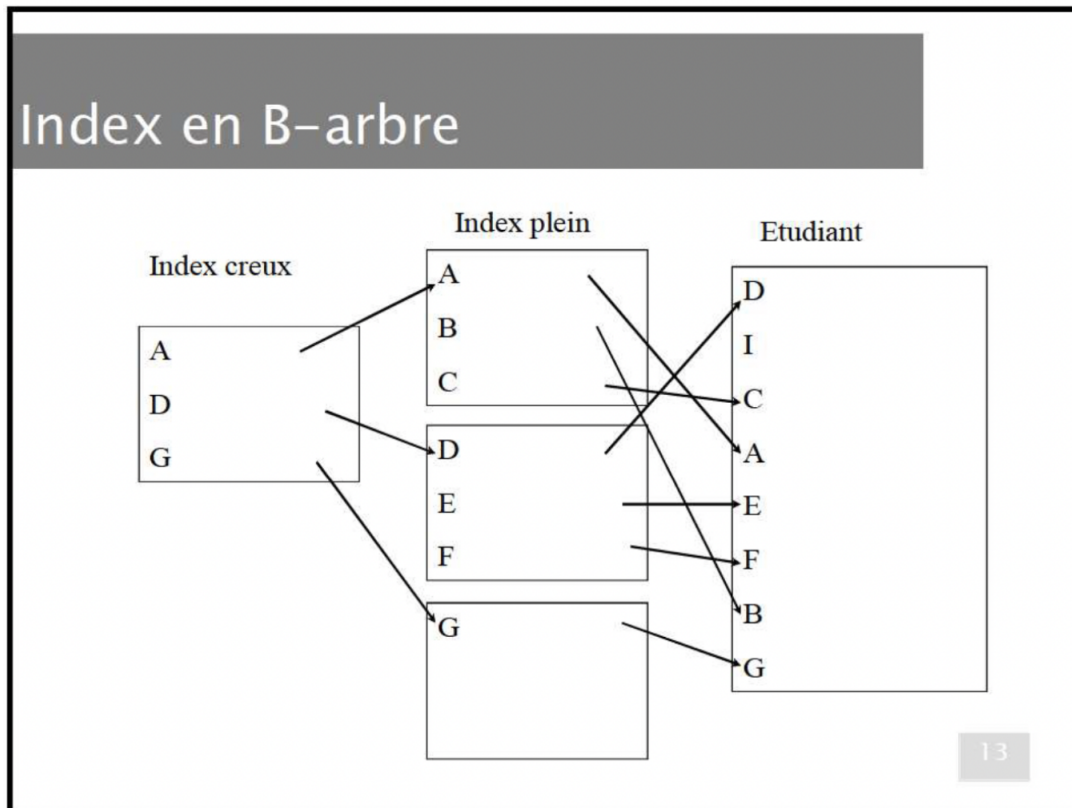
200 000 pages de 100 caractères, d'où :

- $200\,000 * 100c \rightarrow 20\,000\,000c$
- $1000c \rightarrow 1$ page
- nbpages à lire = $20\,000\,000/1000 = 20\,000$
- tps de lecture = $20000 * 15ms = 5mins$

Autrement dit, il faut 5 mins pour trouver le numéro de téléphone d'une personne dans un registre de 200 000 entrées.

2) Avec Index :

Rappel :



Dans notre cas, le disque contient des entrées de 100c, tandis que les index contiendront des entrées de 20c+5c (nom+référence).

Nombre de pages par index :

- Index plein : $200\,000 * (20c + 5c) / 1000 \rightarrow 5\,000$
- Index creux 1^{er} niveau : $5\,000 * (20c + 5c) / 1000 \rightarrow 125$
- Index creux 2^e niveau : $125 * (20c + 5c) / 1000 = 3.125 \rightarrow 4$ (on arrondit toujours au supérieur pour les données qui "débordent")
- Index creux 3^e niveau : $4 * (20c + 5c) / 1000 < 1 \rightarrow$ on s'arrête

Il faut donc lire 5 pages (dans cet ordre) :

- Index creux 3^e niveau : 1 lecture
- Index creux 2^e niveau : 1 lecture
- Index creux 1^{er} niveau : 1 lecture
- Index plein : 1 lecture
- lecture effective dans le disque : 1 lecture

D'où un temps d'exécution total de $5 * 15 = 75ms$

Afin de démontrer la puissance de l'indexage, on peut simuler ce qu'il se passerait en doublant la quantité d'entrées.

Nombre de pages par index avec 400 000 enregistrements :

- Index plein : $400\,000 * (20c + 5c)/1000 \rightarrow 10\,000$
- Index creux 1er niveau : $10\,000 * (20c + 5c)/1000 \rightarrow 250$
- Index creux 2e niveau : $250 * (20c + 5c)/1000 = 6.25 \rightarrow 7$
- Index creux 3e niveau : $7 * (20c + 5c)/1000 < 1 \rightarrow$ on s'arrête

En doublant la quantité d'entrées, le nombre de lecture de page par requête n'a pas augmenté.