TD Optimisation BD

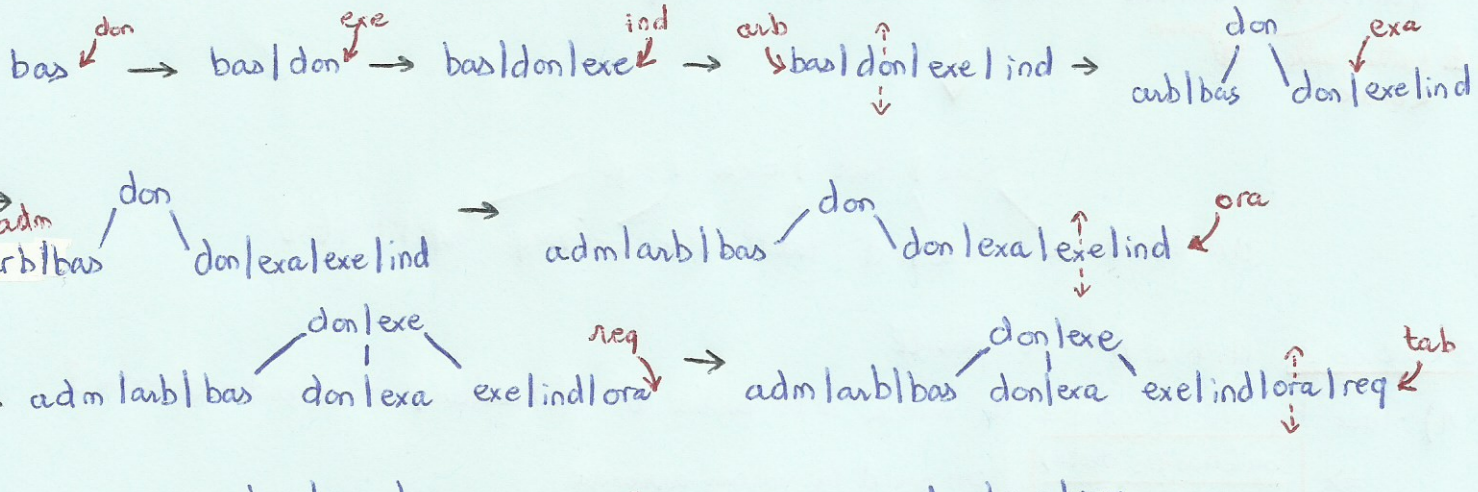
Exercice 1: Indexation par arbre B+

0) 1 opération minimum – 14 opérations maximum – 7 opérations en moyenne

1) On utilise un algorithme de tri lexicographique puis une recherche dichotomique → nombre d'opérations divisé par 2.

2) Arbre B+ d'ordre 2 : [2-4]

↳ bas, don, exe, ind, aub, exa, adm, ora, req, tab, sql



3) Il permet d'accélérer la recherche de mots (ex: donner tous les mots commençant par "a"). Au lieu de faire un full index scan, on fera un parcours par range

4) On ajoute "str", "liv", "lat" et "lon"

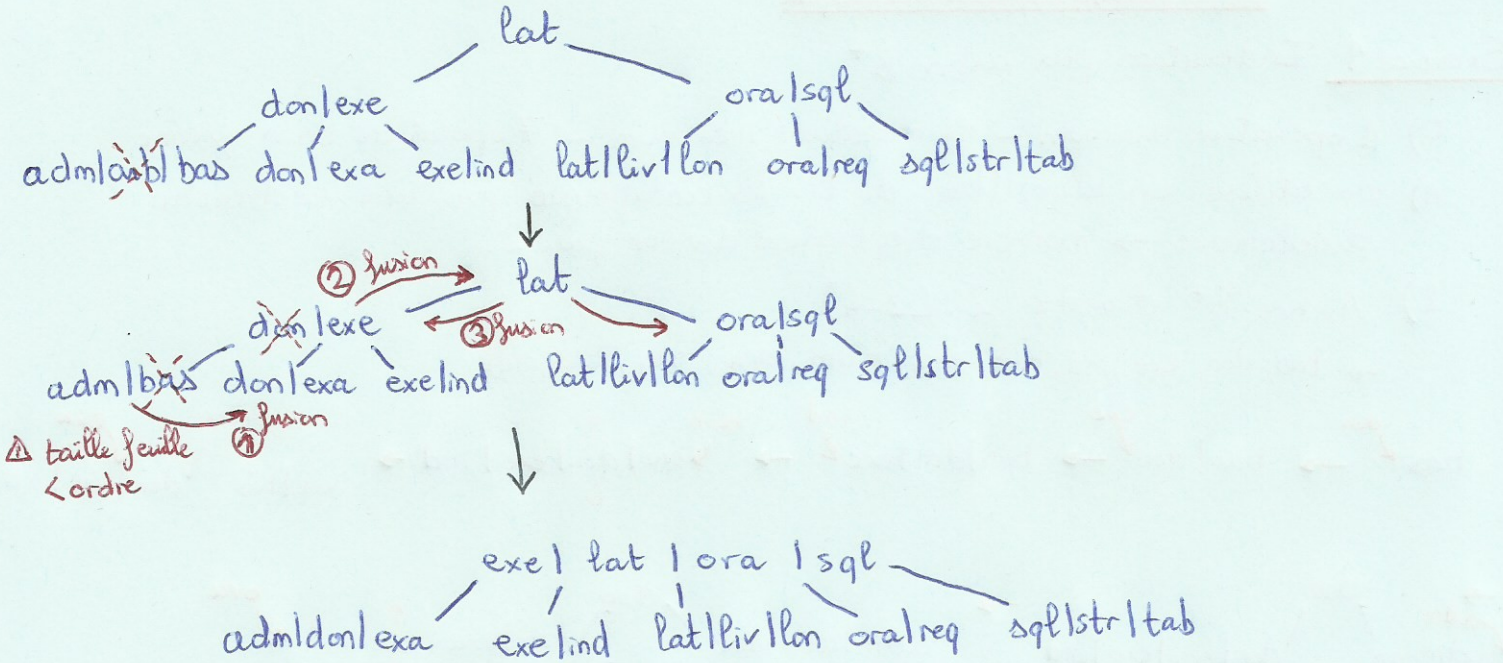
5) Pour trouver l'index d'un mot dans un arbre il faut parcourir h noeuds où h est la hauteur de l'arbre et n fois les valeurs du noeud où t est 2x l'ordre de l'arbre (2d).

↳ position dans la liste des valeurs de taille t

Au minimum on aura donc 3 opérations → h = 3 et n = 1 si l'arbre est complet

Au maximum on aura donc 3 x 4 opérations → h = 3 et n = 4 = 12 opérations

6) On veut supprimer les mots "arb" et "bas"



Exercice 2: Arbre B+ - complexité

1) Pour trouver le nombre de niveau à un arbre B+ (hauteur) d'un arbre d'ordre d

$$\Rightarrow \boxed{\frac{\log(\text{nombre data})}{\log(2d)}}$$ on prend la valeur arrondie au supérieure

ordre 3: Pour 1 million de data $\Rightarrow$ hauteur max $= \frac{\log(1\,000\,000)}{\log(2\times 3)} = 7,7 \Rightarrow H = 8$

nombre maximal d'opérations = 8 hauteurs avec 6 valeurs par noeud (6 comparaisons)
$= 8 \times 6 = 48$ opérations maximum $^{(2d)}$

2) ordre 100: Pour 1 million de data $\Rightarrow H = 3$ et $200 \times 3 =$ 600 opérations maximum

ordre 6 pour 1 billion de data $\Rightarrow H = 12$ et $12 \times 12 =$ 144 opérations max

ordre 100 pour 1 billion de data $\Rightarrow H = 6$ et $200 \times 6 =$ 1200 opérations max

Plus on augmente l'ordre de l'arbre plus on a d'opérations mais en cas de mise à jours de l'arbre (ajout/suppression de valeur) il y aura + de modifications dans un arbre d'ordre faible.

Exercice 3 : Table de hachage

1) Nombre de pages = nombre de lignes / Nombre elements par page = nombre de colonne

4 pages et 2 éléments par page ⇒ 

Fonction de hachage = Σ (chiffres) mod 4

On veut insérer les nombres : 16 → 1+6=7 → 7 mod 4 = 3
37 → 3+7=10 → 10 mod 4 = 2
8 → 8 mod 4 = 0
2 → 2 mod 4 = 2
53 → 5+3=8 → 8 mod 4 = 0
12 → 1+2=3 → 3 mod 4 = 3

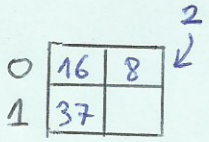
8 ③	53 ⑤	0
		1
37 ②	2 ④	2
16 ①	12 ⑥	3

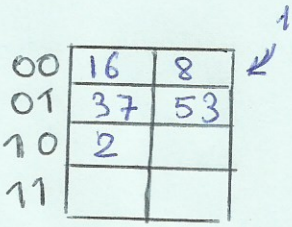
(n) : ordre insertion

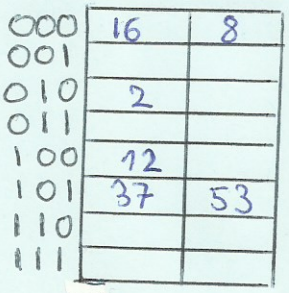
2) On s'intéresse maintenant a un hachage dynamique avec une indexation sur les bits de poids faibles des nombres.

① $16_{10} \rightarrow 1\,0000_2$ | ③ $8_{10} \rightarrow 1000_2$ | ⑤ $53_{10} \rightarrow 11\,0101_2$

② $37_{10} \rightarrow 10\,0101_2$ | ④ $2_{10} \rightarrow 10_2$ | ⑥ $12_{10} \rightarrow 1100_2$







Ces instructions vont entraîner un découpage potentiellement déséquilibré ⇒ certaines pages resteront vides.