

2. Soit la base de données ci-dessous, contenant 300.000 clients de 10 succursales.

Client (nro_client, nom_client, adresse, telephone, succursale)

La relation Client est organisée en arbre B+ sur l'attribut nro_client, clé de la relation.

On suppose que l'attribut succursale soit stocké sur 23 octets, l'attribut nom_client sur 50 octets, l'attribut adresse sur 50 octets, telephone sur 20 octets et nro_client sur 7 octets.

On suppose que :

- une adresse de page est stockée sur 8 octets
- les pages de la base ont 3000 octets pour les données + 8 octets d'adresse.

Complétez le tableau des statistiques contenues dans le catalogue

- Taille de nuplets =
- NTuples(Client) =
- BFactor(Client) =
- NBlocks((Client)) =

(clé primaire)	
nro_client	

→ pour stocker les enregistrements

→ par ex : permet de stocker l'adresse de la page suivante par ex.

→ Arbre B+ :

particularité, ses noeuds contiennent

explication :

Master 1 MIAGE
BDA -TD2 Indexation

- NLevel nro_client (l) =
- NLBlocks nro_client (l) =

Suppose que :

une adresse de page est stockée sur 8 octets

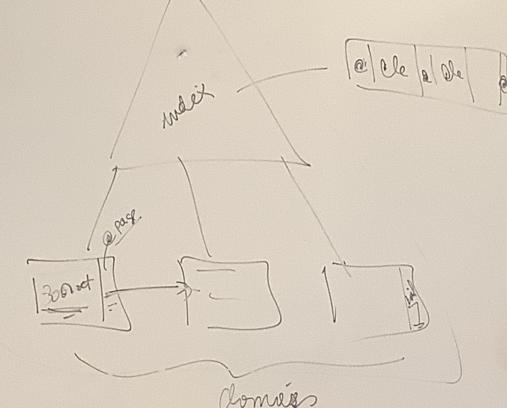
les pages de la base ont 3000 octets pour les données + 8 octets d'adresse. Taille d'une page = 3008.

Complétez le tableau des statistiques contenues dans le catalogue

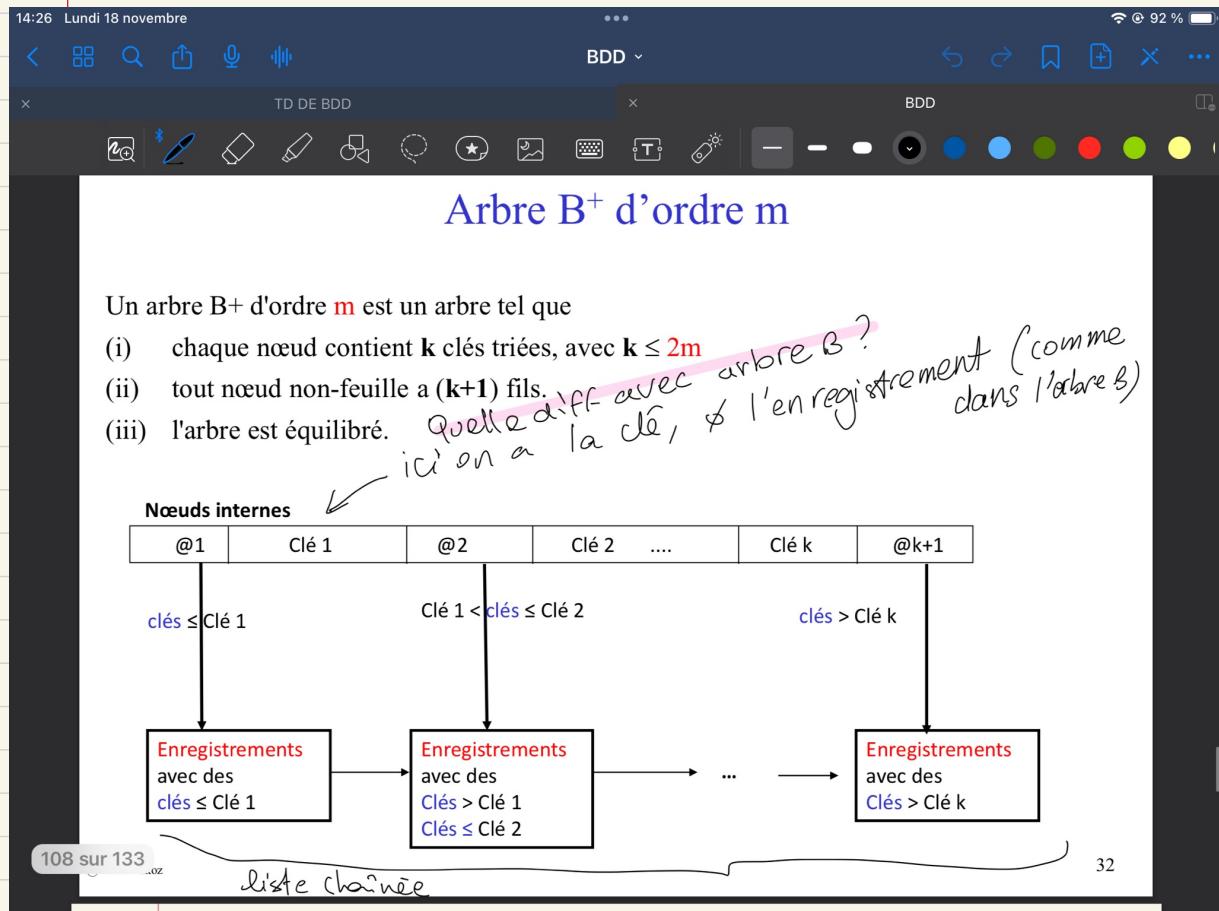
- Taille de nuplets = $23 + 50 + 50 + 20 + 7 = 150$ oct
- NTuples(Client) = 300.000
- BFactor(Client) = $3000 / 150 = 20$
- NBlocks((Client)) = $300.000 / 20 = 15.000$ blocks

Pourquoi 3000 ?

Voir ici

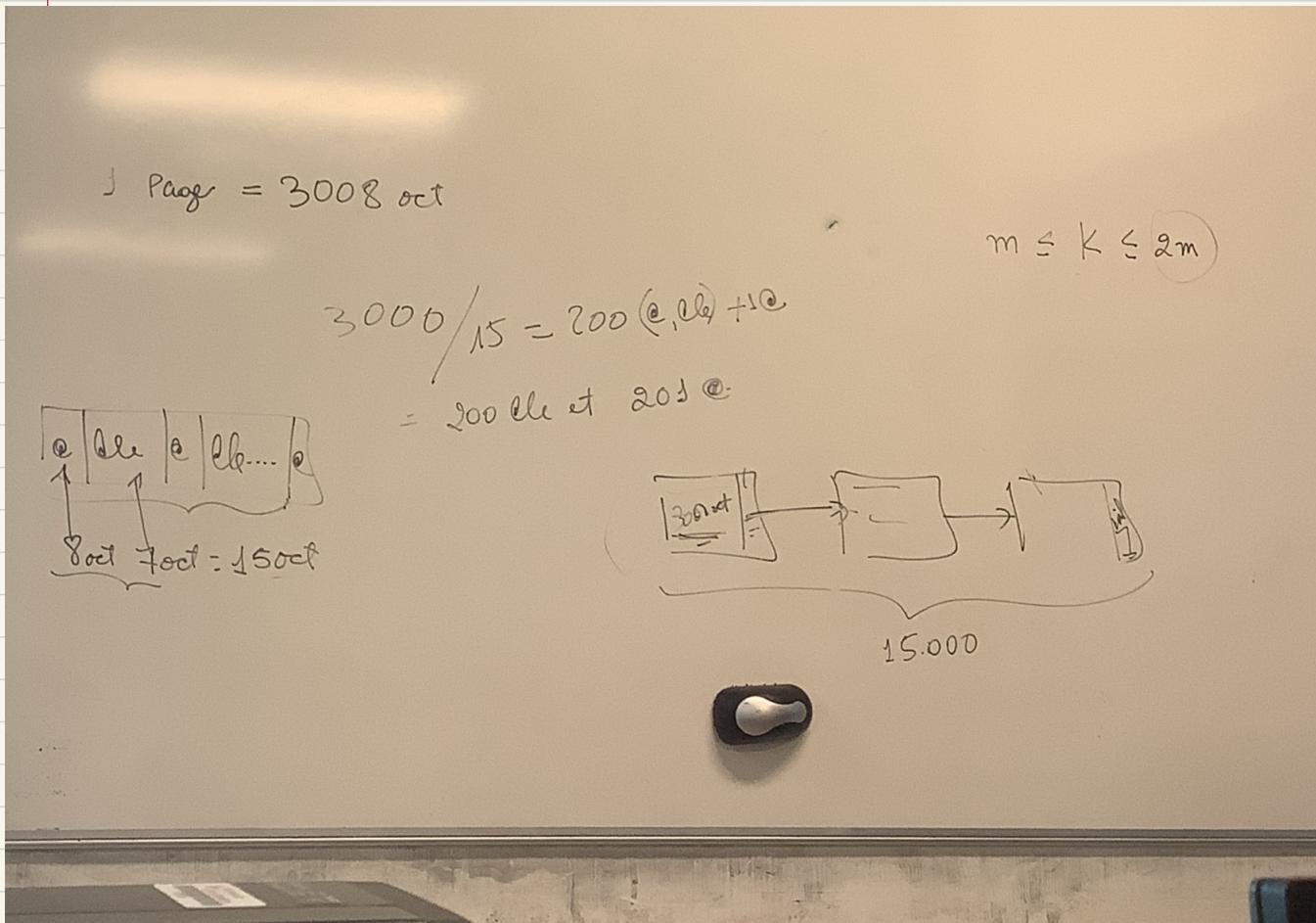


→ Les 8 octets sont un piège, pour expliquer qu'ici on est dans un arbre B+ et qu'on utilise une liste chaînée (au niveau des feuilles). Donc ça implique un emplacement mémoire en + de 8 octets.



→ Explications.

→ Dans un arbre B+, le nb de clés doit être pair, ici 200 est pair

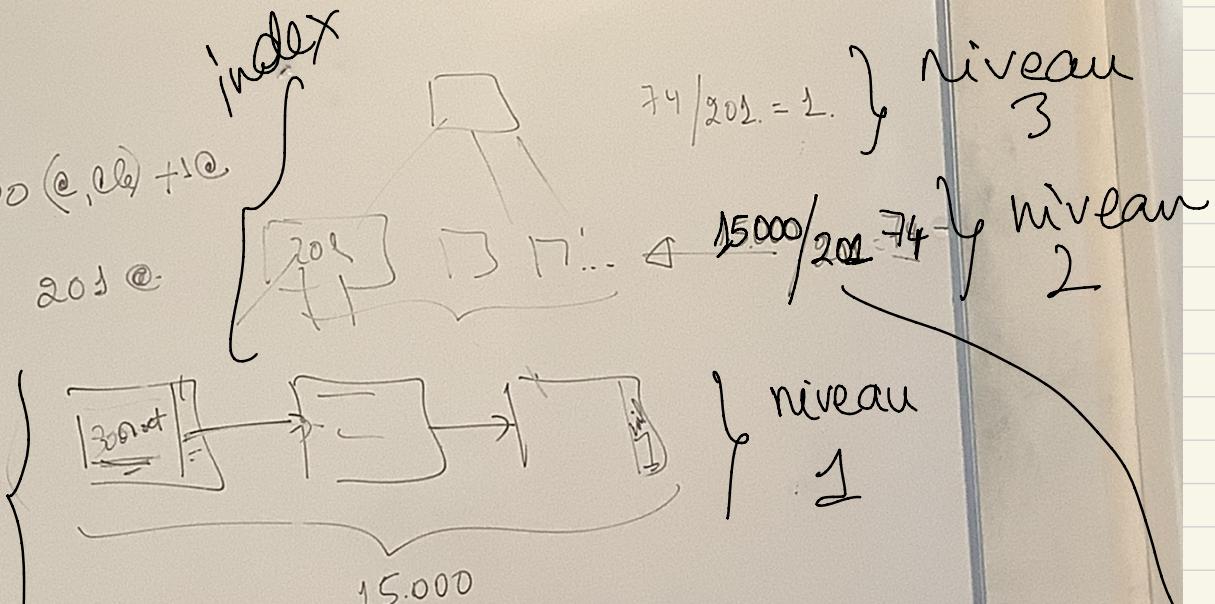


N level (nb de pages on a besoin pour stocker mes données)

Page = 3008 oct

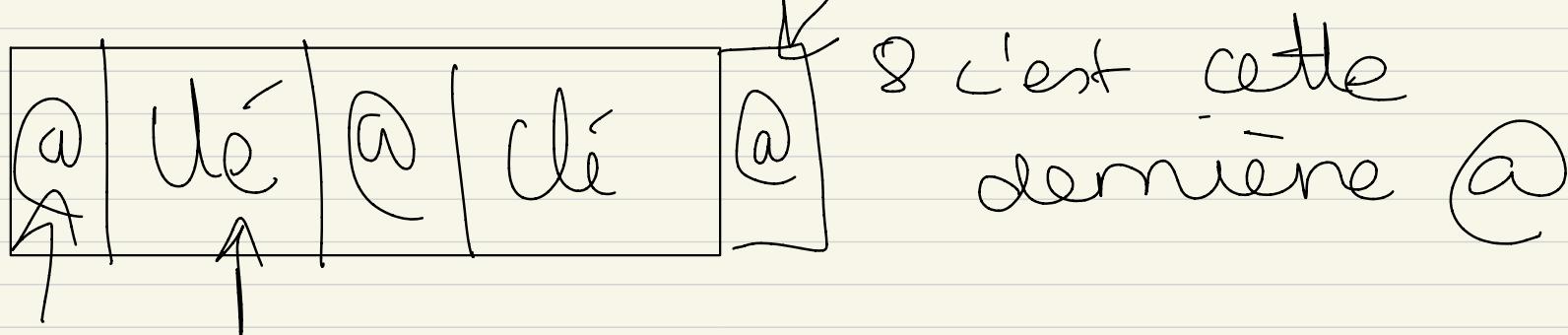
$$3000 / 15 = 200 \text{ (c. 16) } + 12 \\ = 200 \text{ feuilles et } 12 \text{ oct}$$

valeurs
en feuilles



Pourquoi 201 et pas 200 ?

Une page = 300 8 octets



8 octets 7 octets = 15 octets

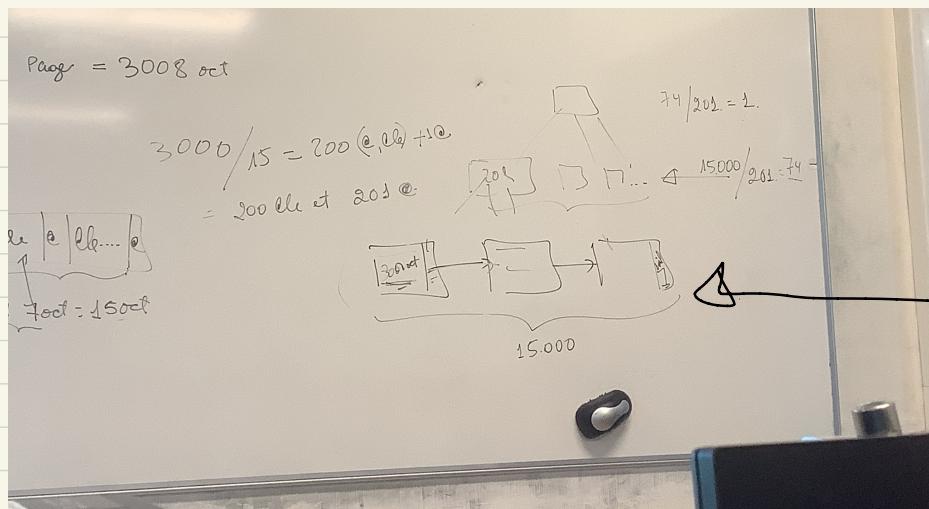
→ On peut aussi faire avec

log : $\log_{201} (15 \ 000)$;
nb de divisions (adresses)

C'est ϕ 300 000
car on regroupe
nos enregistrements
par paquets (en
pages)

$$\geq \text{NBlocks}(l) = \underbrace{74 + 1 = 75}_{\text{pour l'index}} + \underbrace{15\ 000}_{\text{les données}}$$
$$= 15\ 075$$

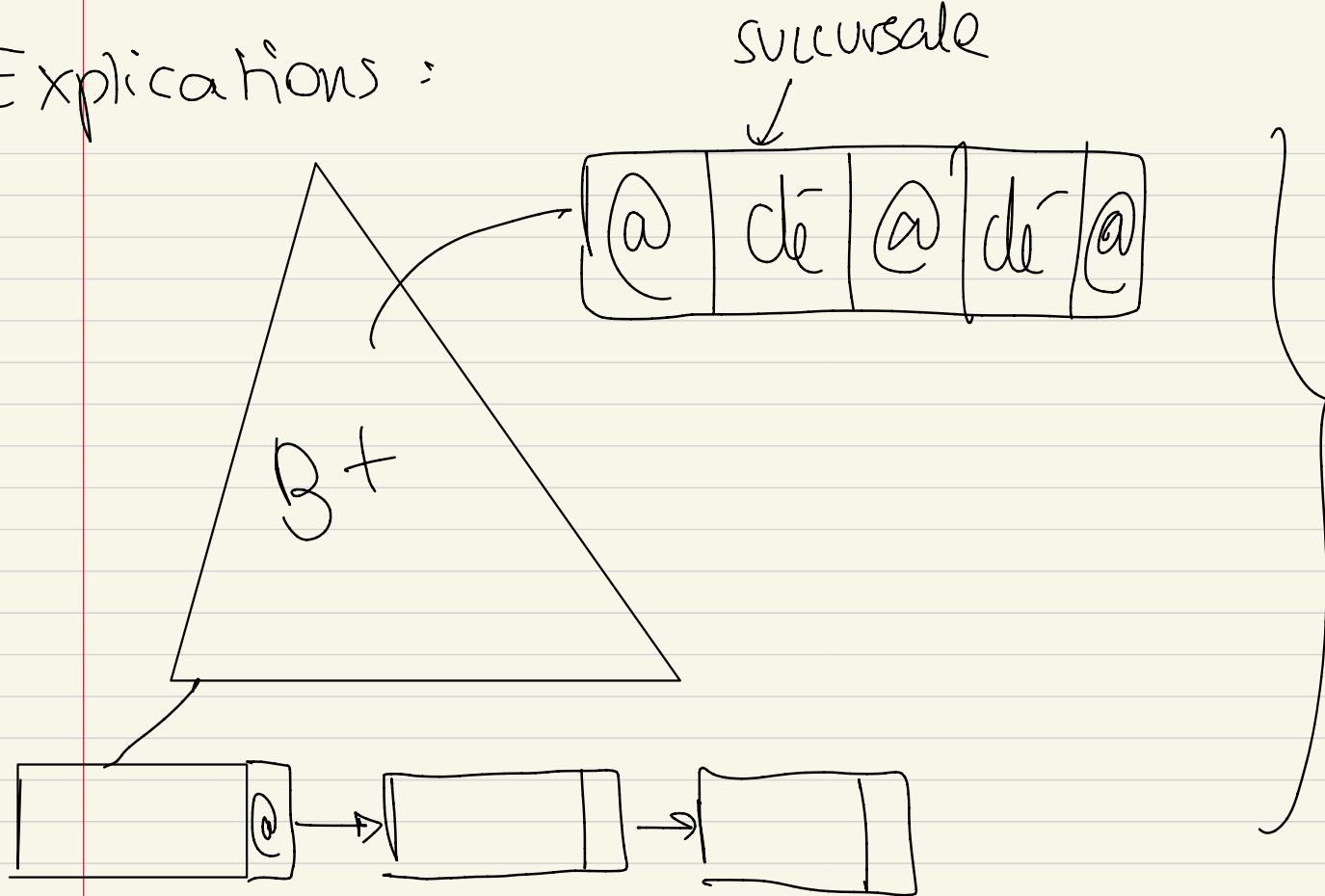
3. Index primaire = ici, les feuilles sont triées
selon la clé
→ donc index dense



→ cette liste
chaînée est triée

Pour l'I. secondaire = pq ça peut pas
être dense ? car non trié par
rapport à succursale
→ il est forcément non dense

Explications :



Index :

ça c'est les enregistrements de l'index

(SUCURSALE, nro-client)

$$\underbrace{23 \text{ oct}}_{23 \text{ oct}} \quad \underbrace{7 \text{ oct}}_{7 \text{ oct}} = 30 \text{ oct}$$

Combien d'enregistrements par page ?

$$R\text{Factor} = \frac{\text{Taille page}}{\text{Taille tuple}} = \frac{3000}{23 + 7} = 100$$

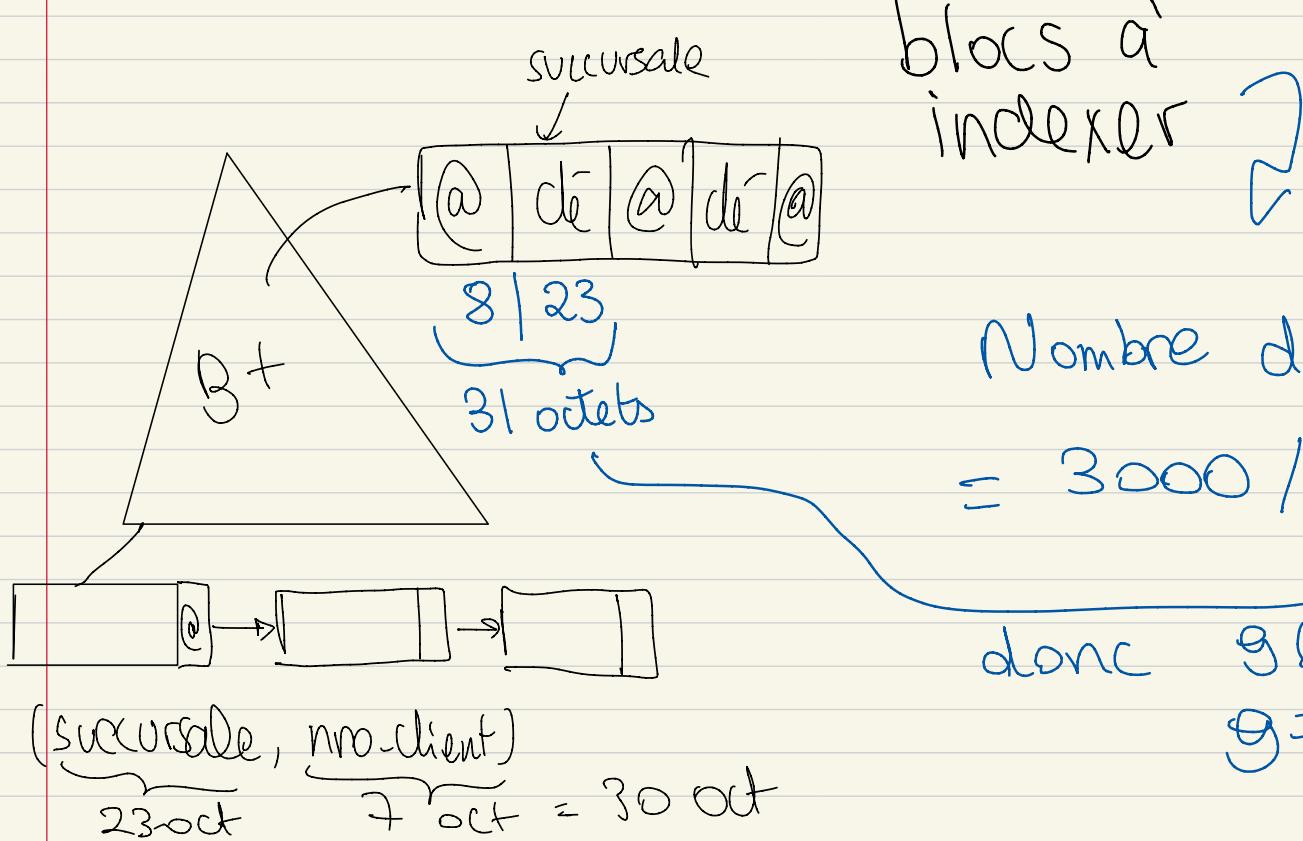
(SUCURSALE, nro-client)

100 enregistrements par page

$$\text{NBlocks (succursale)} = 300\ 000 / 100$$

nb de blocs
de l'index
succursale

BFactor

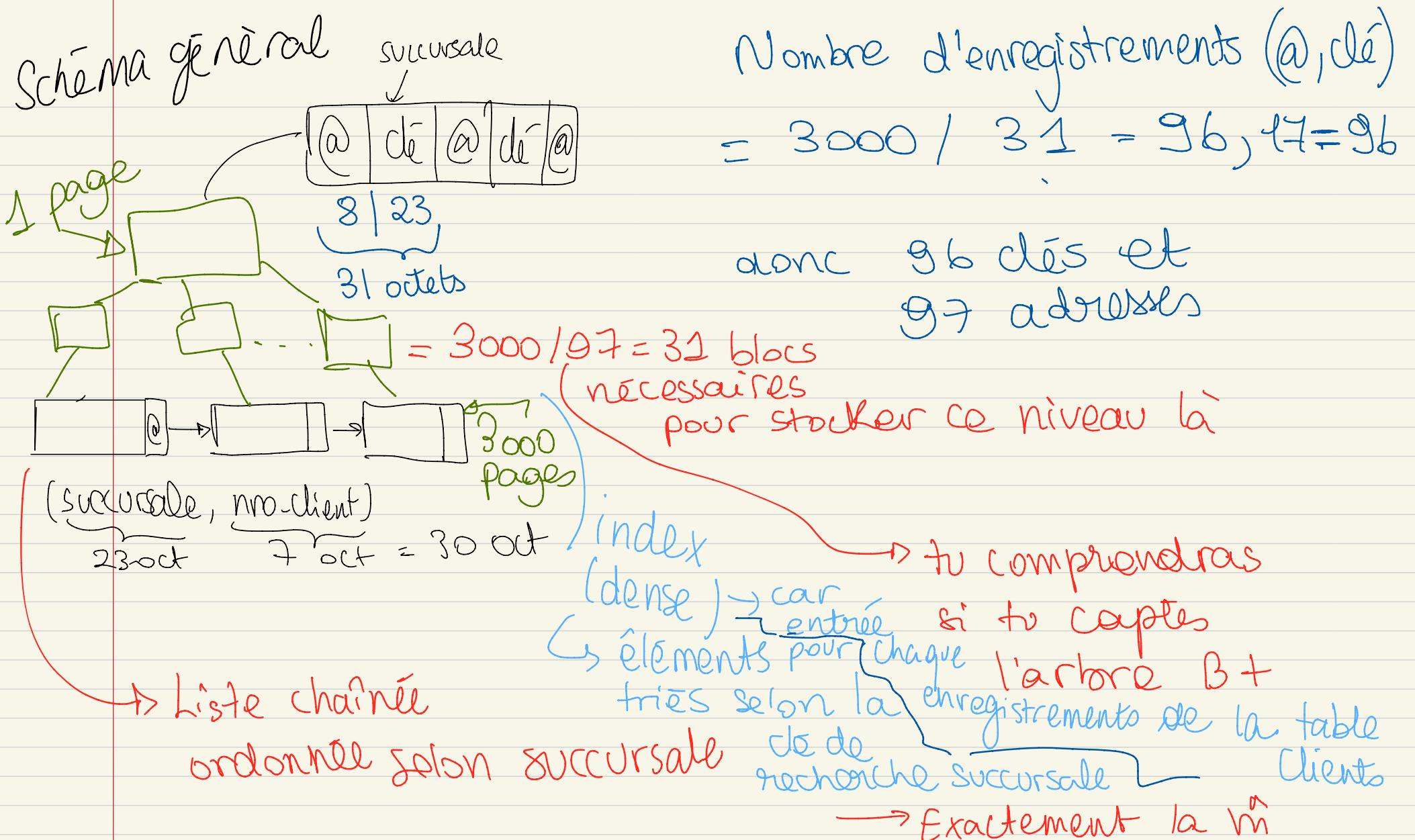


blocs à indexer

Nombre d'enregistrements (@, dé)

$$= 3000 / 31 = 96,47 = 96$$

donc 96 clés et 97 adresses



Cas Particulier : (Important, l'arbre n'a que 2 niv. dans ce cas)

\rightarrow on indexe pour chaque succursale et non sur chaque enregistrements (car y'a que 10 succursales)

Note : Savoir expliquer ce qu'est un index secondaire.

$$\triangleright SC_{succursale}(\text{client}) = \frac{300\ 000}{10} = 30\ 000$$

(

si je fais un

Select *

From Client

Where succursale = 'Valeur'

répartition
de manière
uniforme

10

→ 10 succursales
(# (énoncé))

→ je peux
donc en
indexer que

10 (pas
besoin d'indexer
en double)

→ Résultat : une table avec 30 000 nuplets

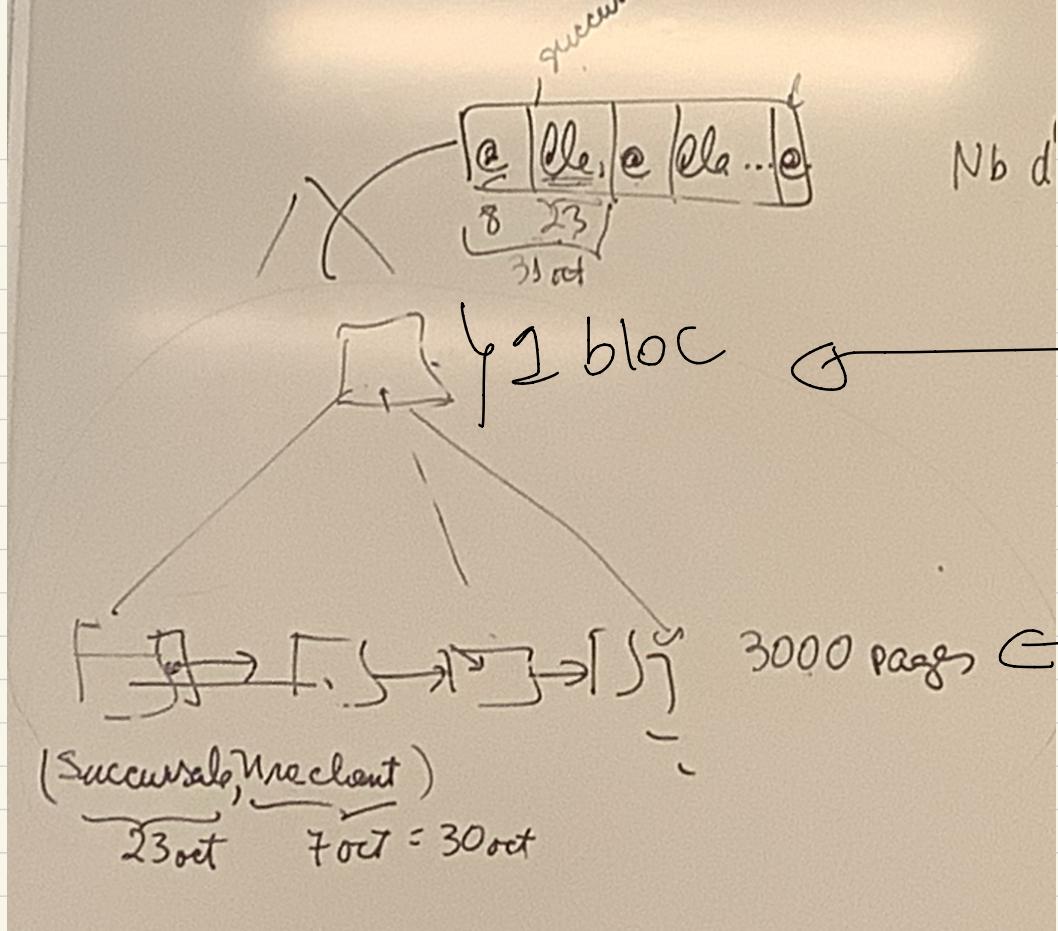
⚠

→ C'est à prendre en compte pour

la partie Evaluations de requêtes

$$\triangleright N_{Distinct}_{succursale}(\text{client}) = 10 (\text{énoncé})$$

Note: Un index n'est qu'une table
ordonnée



► N Level
succursale (1) = 2 (1 index + les données)

nb de niveaux

► NL Blocks
succursale (1) =
| nb blocs de l'index

1 (3000 pages +
1 page (racine))

↳ NLBlocks (succursale) = 3001

↳ nb blocs totaux

↳ enregistrements d'une table Article

4. Soit un fichier tel que chaque page peut contenir 10 articles. On indexe ce fichier avec un niveau d'index (un seul), et on suppose qu'une page d'index contient 100 paires (clé, pointeur). Si n est le nombre d'articles, donnez la fonction de n permettant d'obtenir le nombre minimum de pages pour les structures suivantes :
- Fichier séquentiel non ordonné avec un index dense.
 - Fichier trié sur la clé avec un index non-dense.
5. On reprend les hypothèses précédentes et on indexe le fichier avec un arbre-B+. Les feuilles de l'arbre contiennent donc des pointeurs vers le fichier, et les nœuds internes des pointeurs vers d'autres nœuds. On suppose qu'une page d'arbre B+ est pleine à 70 % (69 clés, 70 pointeurs).
- Le fichier est trié sur la clé et indexé par un arbre B+ non dense. Donnez (1) le nombre de niveaux de l'arbre pour un fichier de 1 000 000 d'articles, (2) le nombre de pages utilisées (y compris l'index), et (3) le nombre d'accès disque pour rechercher un article par sa clé.
 - On effectue une recherche par intervalle ramenant 1000 articles. Décrivez la recherche et donnez le nombre d'accès disque dans le pire des cas

Non dense
= indexé par page

Dense =
indexer par enregistrements

→ a) NPages (n) = $\frac{n}{10} + \underbrace{\frac{n}{100}}_{\substack{\text{nombre} \\ \text{de pages} \\ \text{pour les données}}} = \frac{10n + n}{100} = \frac{11n}{100}$

→ 100 paires

$$b) N\text{pages}(n) = \frac{n}{10} + \frac{\frac{n}{10}}{100}$$

=

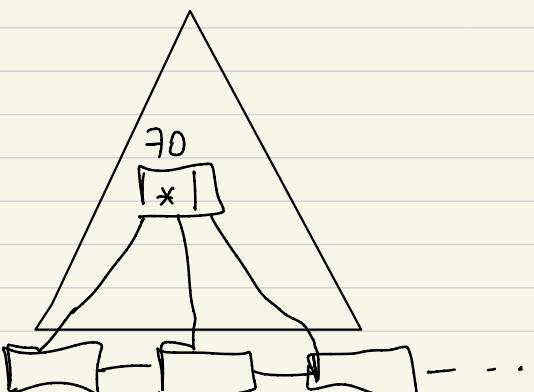
28/11 =
 ↳ 8h30
 ↳ Contrôle
 stockage /
 indexation

Calculatrice

5. On reprend les hypothèses précédentes et on indexe le fichier avec un arbre-B+. Les feuilles de l'arbre contiennent donc des pointeurs vers le fichier, et les nœuds internes des pointeurs vers d'autres nœuds. On suppose qu'une page d'arbre B+ est pleine à 70 % (69 clés, 70 pointeurs).

- a. Le fichier est trié sur la clé et indexé par un arbre B+ non dense. Donnez (1) le nombre de niveaux de l'arbre pour un fichier de 1 000 000 d'articles, (2) le nombre de pages utilisées (y compris l'index), et (3) le nombre d'accès disque pour rechercher un article par sa clé.
- b. On effectue une recherche par intervalle ramenant 1000 articles. Décrivez la recherche et donnez le nombre d'accès disque dans le pire des cas

a.



$n = 1\ 000\ 000$
 10 articles par page

$$B \text{ Factor} = \frac{x}{1000.000} = 10$$

$$\boxed{B} = 2$$

$$\rightarrow 1429 / 70 = 21.$$

$$\rightarrow \rightarrow \rightarrow 100.000 / 70 = 1429 \quad N = 1000000.$$

$$B \text{ Factor} = 10 \text{ art / Page}$$

$$\boxed{1} - \boxed{1} - \boxed{1} - \dots \overbrace{\quad}^{100.000}$$

$$N \text{ pages feuille} = \frac{1000000}{50} = 20000$$

$$Nb \text{ de pages} = 100.000 + 1429 + 21 + 1$$

$$N_{level}(I) = 3 + \text{les données}$$

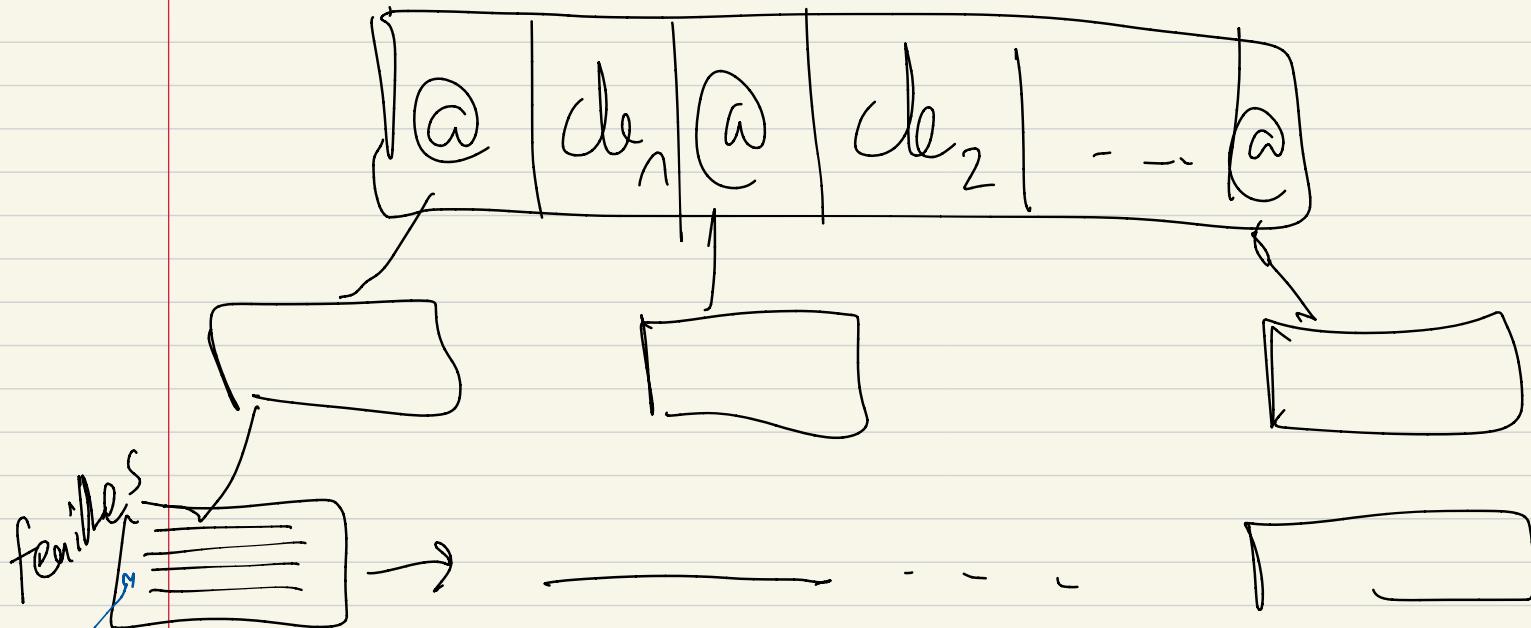
b) Meilleur des cas :

Pour y répondre

Note : 69 des
↳ c'est eux

qui vont faire l'indexat°

→ savoir : l'algo de recherche d'un enregistrement



10 liste chaînée ordonnée par rapp à la val de la clé
enregistrements par page de recherche

→ et nous on cherche 1000 articles

b. Meilleur des cas

↳ l'enregistrement est le 1er de la page

: 100 pages + 3^{nb de blocs pour les niveaux (1 bloc par niveau)}

↳ (1000 articles / BFactor = 10)

Pire cas : 101 pages + 3
 $(100 + 1)$

→ l'enregistrement
n'est pas le 1^{er}, ex: le 5^eme

pour récupérer

→ donc on prend

page
décodage
pr^{re}

