



## Exercice1 : Répartition des enregistrements dans les blocs

Soit la table Employé ci-dessous, contenant 600.000 Employés.

*Employé (num, nom, adresse, telephone, département)*

On suppose que la taille de l'attribut *département* soit 40 octets, de l'attribut *nom* 100 octets, de l'attribut *adresse* 50 octets, *telephone* 20 octets et *num* 10 octets.

On suppose que les pages de la base ont 3000 octets.

1. En considérant le stockage non-étendu :
  - a. Combien d'enregistrements peut-on stocker dans une page ?
  - b. Combien de pages sont-elles nécessaires pour stocker la table Employé ?
  - c. Supposons que la table est stockée dans le fichier F1, donnez l'adresse des enregistrements 100.000ème, 3ème, 25<sup>ème</sup> et 150.000ème.
  - d. Si on considère le stockage étendu, combien de pages seront-elles nécessaires pour stocker la table ?

## Exercice 2 : Organisation directe

1. Construisez une structure de **hachage statique** pour des enregistrements dont les clés sont : (2, 3, 5, 11, 19, 23, 29, 31, 7, 9, 20, 22). On suppose qu'on peut stocker jusqu'à 3 enregistrements par bloc et qu'on a 3 blocs primaires. Pour la gestion de collisions, utilisez la technique :
  - a) *Unchained overflow*
  - b) *Chained overflow*
  - c) *Multiple hashing*.
2. Construisez une structure de **hachage extensible** pour des enregistrements dont les clés sont : (2,3, 5, 11, 19, 23, 29, 31, 7, 9, 20, 22). On suppose qu'on peut stocker jusqu'à 3 enregistrements par bloc.
3. Décrivez l'évolution de la structure précédente (2) après les opérations suivantes :
  - a) Suppression de l'enregistrement 11.
  - b) Suppression de l'enregistrement 31.
4. Soit la table ElementGenerique ci-dessous :

```
Create table ElementGenerique (
    id integer not null,
    nom,
    constraint pk_id primary key (id))
```

et les données :

```
(1,a);
(2,b);
(3,c);
(4,d);
(5,e);
(6,f);
(7,g);
(8,h);
(9,i);
```

**Exercice 1 : Répartition des enregistrements dans les blocs**

Soit la table Employé ci-dessous, contenant 600.000 Employés.

*Employé (num, nom, adresse, telephone, département)*

On suppose que la taille de l'attribut *département* soit 40 octets, de l'attribut *nom* 100 octets, de l'attribut *adresse* 50 octets, *telephone* 20 octets et *num* 10 octets.

On suppose que les pages de la base ont 3000 octets.

1. En considérant le stockage non-étendu :

- Combien d'enregistrements peut-on stocker dans une page ?
- Combien de pages sont-elles nécessaires pour stocker la table Employé ?
- Supposons que la table est stockée dans le fichier F1, donnez l'adresse des enregistrements 100.000ème, 3ème, 25ème et 150.000ème.
- Si on considère le stockage étendu, combien de pages seront-elles nécessaires pour stocker la table ?

**Exercice 2 : Organisation directe**

$$\text{b)} \quad 600\ 000 \div \underbrace{13}_{\begin{array}{l} \text{nb enregistrements} \\ \text{stockables} \\ \text{dans une page} \end{array}} = 46\ 153 \text{ pages nécessaires}$$

c. La table est stockée dans le fichier F1

adresse de l'enregistrement 100 000<sup>ème</sup> :  $\frac{\text{N}^{\circ}\text{enregistrement}}{\text{B factor}}$

$$\text{Page} = \frac{100\ 000}{13} + 1 = 76\ 93 \hookrightarrow \text{il est dans cette page}$$

a) une page a 3000 octets  
un enregistrement

$$= 40 + 100 + 50 + 20 + 10 = 220$$

$$\text{Ensuite B factor} = 3000 / 220 = R / B \\ \approx 13, \text{ ت } \\ \approx 13 \leftarrow$$

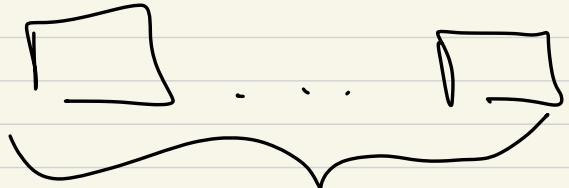
MyP :  
on est en stockage non étendu  
donc on prend la partie inférieure

→ 4<sup>eme</sup> position

Pages précédentes

(7692 x 13)

page précédente



7692 x 13

tous les enregistrements  
qui sont avant



99.997 ... 100 009

↓  
Plage de la page 7693



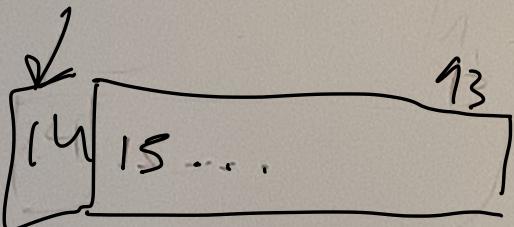
voir photos

3ème.

Page =  $\frac{3}{13} + 1$  = 1<sup>ère</sup> page (reste 3)  
85° : Page =  $\frac{25}{13} + 1$  = 2<sup>e</sup> page (reste 12)

=  $\leftarrow^{13^{\text{ème}}}$  enregistrement  
= 25 - minimum + 1

minimum =  $13 \times (\text{page précédente}) + 1$   
=  $13 \times 1 + 1 = 14$



13 MAX =  $13 \times (\text{page actuelle})$

$100000 - 99.997 = 3 + 1 = 4$

$7692 \times 13$        $7693$

(c)

R - employés (40, 20, 50, 60, 10) (Dept, Nom, Adresse, Tel, Num)

Nbre d'octets = 220

$\frac{R}{B} = \frac{3000}{220} \approx 13$

## ADDRESS 1

$\rightarrow 100000 : \frac{100000}{13} + 1 = 7693$

$\rightarrow 4^{\text{ème}}$  position.

$(7692 \times 13) + 1 = 99997$

$7693 \times 13 = 100000$  (13-1)=4  
 $99996 \quad 99997 \dots - 100000$

$(100000 - 99997) + 1 = 3 + 1 = 4$

Voir autre  
Calcul

d. Stockage étendu :

↳ Pas de gaspillage

$$\frac{600 \text{ 000}}{3000} \times (220) = 44 \text{ 000 pages}$$

taille  
d'un  
enregistrement

$$600 \text{ K} \times 220$$



organisat°

→ séquentielle

allocation  
chaînée

allocation  
contigüe

directe → ha

## Exercice 2 : Organisation directe

1. Construisez une structure de **hachage statique** pour des **enregistrements** dont les clés sont : (2, 3, 5, 11, 19, 23, 29, 31, 7, 9, 20, 22). On suppose qu'on peut stocker jusqu'à 3 enregistrements par bloc et qu'on a 3 blocs primaires. Pour la gestion de collisions, utilisez la technique :
  - a) *Unchained overflow*
  - b) *Chained overflow*
  - c) *Multiple hashing.*
2. Construisez une structure de **hachage extensible** pour des enregistrements dont les clés sont : (2, 3, 5, 11, 19, 23, 29, 31, 7, 9, 20, 22). On suppose qu'on peut stocker jusqu'à 3 enregistrements par bloc.
3. Décrivez l'évolution de la structure précédente (2) après les opérations suivantes :
  - a) Suppression de l'enregistrement 11.
  - b) Suppression de l'enregistrement 31.
4. Soit la table ElementGenerique ci-dessous :

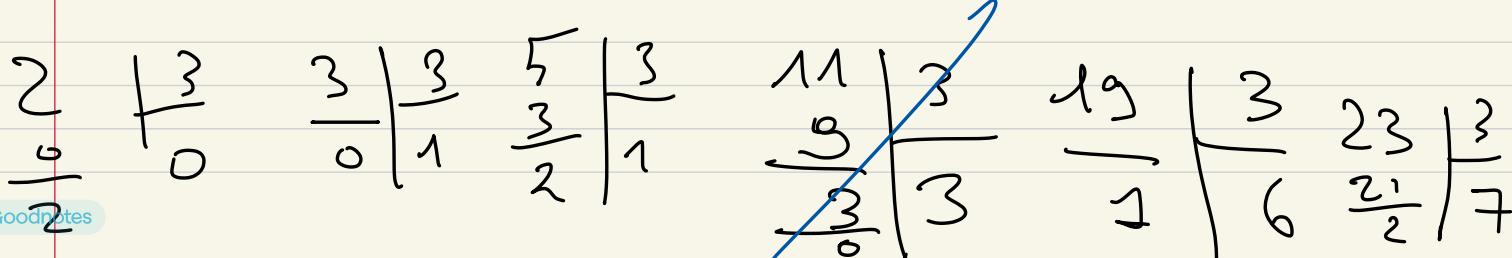
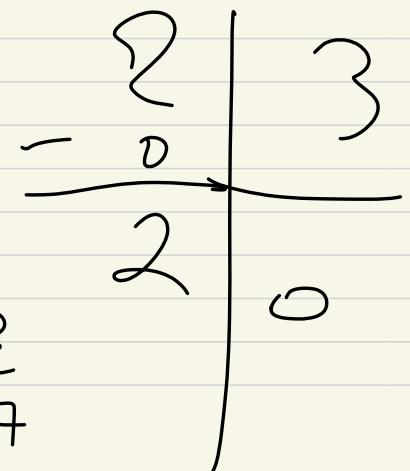
```
Create table ElementGenerique (
    id integer not null,
    nom,
    constraint pk_id primary key (id))
```

et les données :

```
(1,a);
(2,b);
(3,c);
(4,d);
(5,e);
(6,f);
(7,g);
(8,h);
(9,i);
```

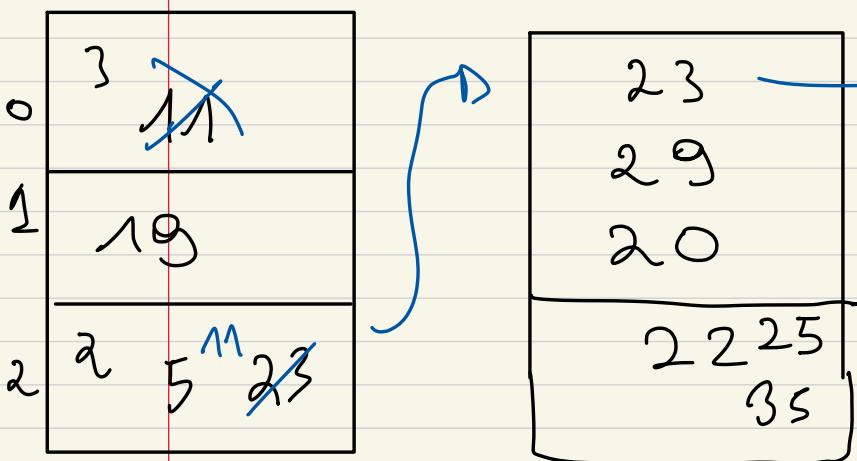
Fonct° de hachage:  
modulo 3

2 % 3

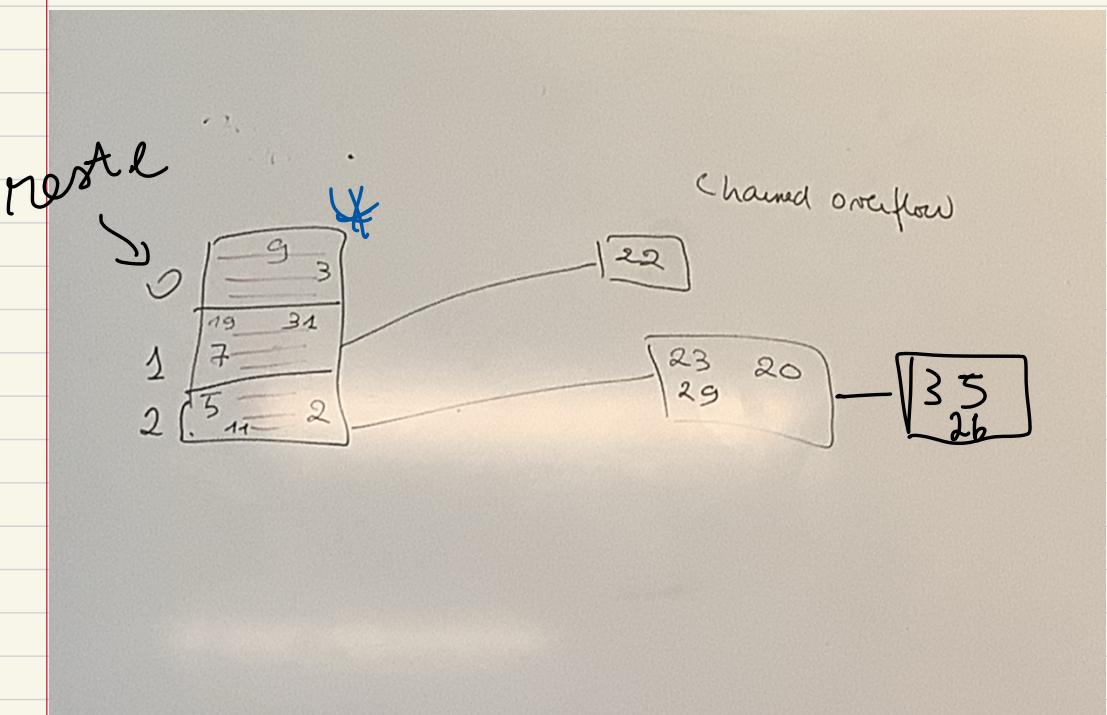


$$\begin{array}{r}
 & 9 & | & 3 \\
 - & 9 & | & 3 \\
 & 0 & &
 \end{array}$$

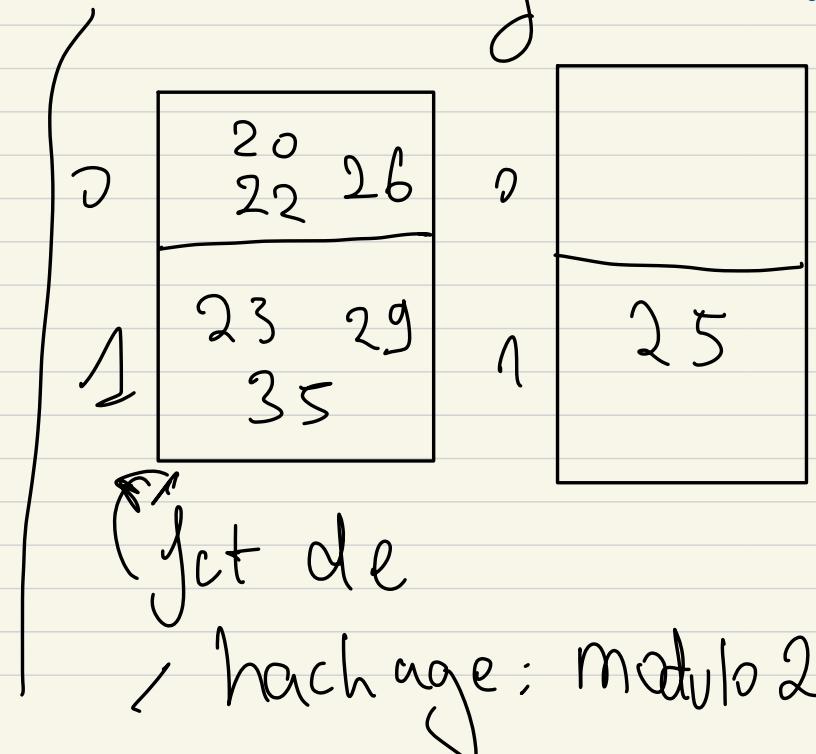
Unchained overflow



23 est mis car  
collision



Multi-tasking (répétition)



On doit choisir le niveau de blocs  
ici c'est : 2

il doit être inférieur à 3 ( bloc primaire)

CC :

→ Aspect théorique

↳ connaissances bases de données

↳ Notions :

→ Déf Index (index primaire)

→ clé primaire, clé étrangère

# → Jointure (savoir définir) 10/10

## Exercice 2 : Organisation directe

1. Construisez une structure de **hachage statique** pour des enregistrements dont les clés sont : (2, 3, 5, 11, 19, 23, 29, 31, 7, 9, 20, 22). On suppose qu'on peut stocker jusqu'à 3 enregistrements par bloc et qu'on a 3 blocs primaires. Pour la gestion de collisions, utilisez la technique :
  - a) Unchained overflow
  - b) Chained overflow
  - c) Multiple hashing.

→ déjà fait
2. Construisez une structure de **hachage extensible** pour des enregistrements dont les clés sont : (2,3, 5, 11, 19, 23, 29, 31, 7, 9, 20, 22). On suppose qu'on peut stocker jusqu'à 3 enregistrements par bloc.  
→ à faire RTN
3. Décrivez l'évolution de la structure précédente (2) après les opérations suivantes :
  - a) Suppression de l'enregistrement 11.
  - b) Suppression de l'enregistrement 31.
4. Soit la table ElementGenerique ci-dessous :  
→ home

```
Create table ElementGenerique (
    id integer not null,
    nom,
    constraint pk_id primary key (id))
```

et les données :

→ ici

```
(1,a);
(2,b);
(3,c);
(4,d);
(5,e);
(6,f);
(7,g);
(8,h);
(9,i);
```

(10,j);  
(11,k);  
(12,l);  
(13,m);  
(14,n);  
(15,o);  
(16,p);  
(17,q);  
(18,r);  
(19,s);  
(20,t);

Supposons que :

Un entier peut être stocké sur 8 octets, une chaîne de caractères sur 32 octets.

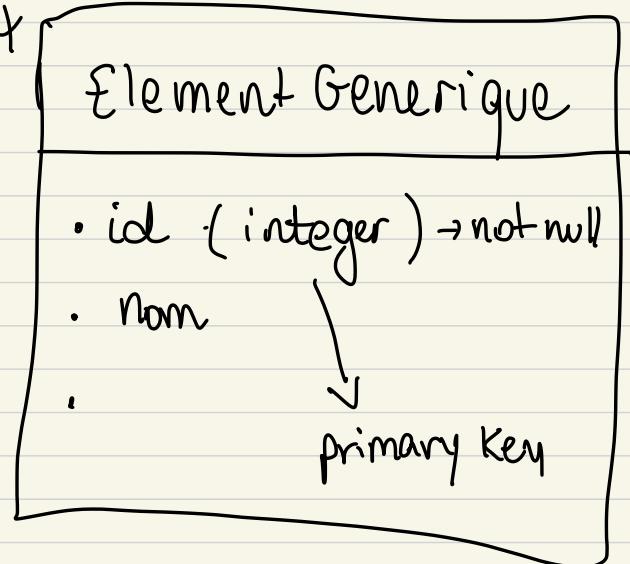
Une adresse de page est stockée sur 8 octets.

Les pages de la base ont 240 octets.

Stockez la table ElementGenerique en utilisant une organisation de **hachage extensible** sur la clé de la relation, et en ne considérant au départ que le dernier bit de la représentation binaire de la clé de la relation.

- a) Quelle est la taille d'un enregistrement ?
- b) Combien d'enregistrements peuvent être stockés par page ?
- c) Montrez l'organisation de hachage immédiatement après d'avoir inséré le 12ème élément.
- d) Montrez l'organisation de hachage immédiatement après d'avoir inséré le 13ème élément ?
- e) Montrez l'organisation de hachage après d'avoir inséré tous les éléments ?
- f) Combien de pages ont été nécessaires pour stocker la table ElementGenerique ?
- g) Combien de pages doit-on lire pour répondre les requêtes :
  - i. Select \* from ElementGenerique where id=7
  - ii. Select \* from ElementGenerique where nom ='r'
  - iii. INSERT INTO ElementGenerique (id, nom) VALUES (22, 'hf')
  - iv. DELETE FROM ElementGenerique where id=30

4



Données :

- (1, a) → ici 1 est l'id  
a est le nom
- (2, b)
- (3, c) ...

→ un entier est stocké sur 8 octets

- ↳ chaîne de caractères, sur 32 octets
- ↳ adresse de page, sur 8 octets
- ↳ pages de la base : 240 octets

$$\begin{array}{c|ccccc} z^4 & z^3 & z^2 & z^1 & z^0 \\ \hline 16 & 8 & 4 & 2 & 1 \end{array}$$

Hana Ouraghene

## Exercice 2, question 4

### Conversions

1 : 0001

6 : 0110

11 : 1011

16 : 10000

2 : 0010

7 : 0111

12 : 1100

17 : 10001

3 : 0011

8 : 1000

13 : 1101

18 : 10010

4 : 0100

9 : 1001

14 : 1110

19 : 10011

5 : 0101

10 : 1010

15 : 1111

20 : 10100

$z^7 z^2$

a) d'id, un entier = 8 octets

de nom, une chaîne de caractères = 32 octets

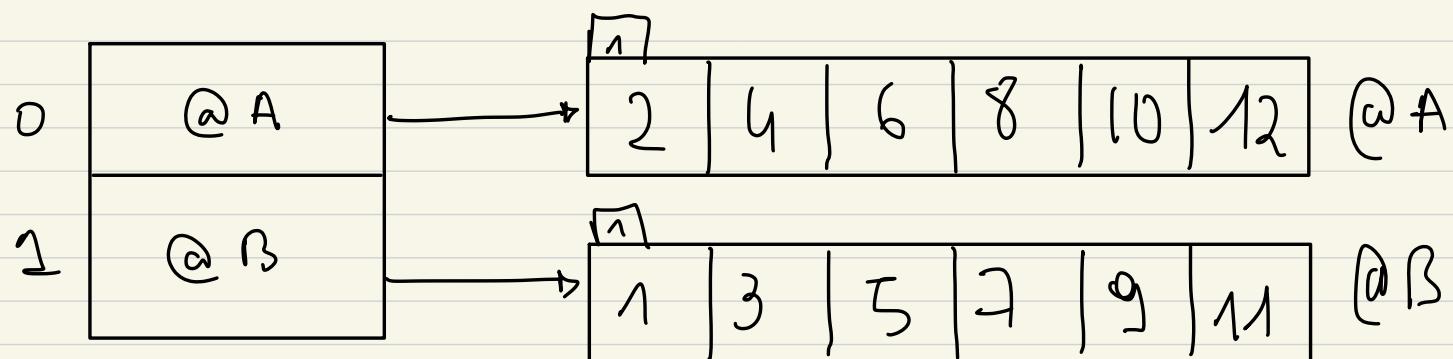
On additionne les champs, d'où 40 octets par enregistrement

b.) Taille d'une page : 240 octets

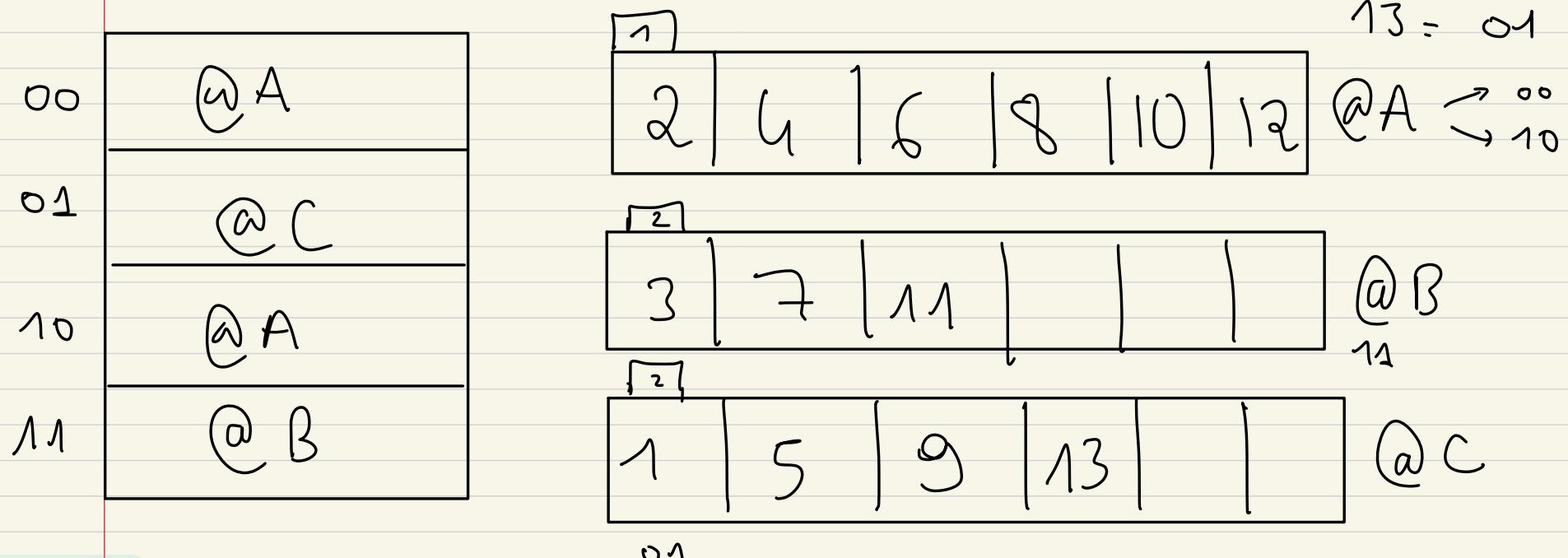
Un enregistrement fait 40 octets,

D'où : B factor =  $240 / 40 = 6$  enregistrements par page.

c.



d. Il faut dédoubler le répertoire car il n'y a plus de place pour insérer la clé: 13



Remarque:

Si, le bloc @A est plein aussi, mais j'ai interprété « immédiatement » (de la consigne) comme cela.

→ Insertion de 14 (11<sup>10</sup>)

→ Dédoublement de bits pour @A

00	@A
01	@C
10	@D
11	@B

2	4   8   12   16   20   .	13 = 01 @A $\rightarrow$ ..
---	--------------------------	--------------------------------

2	3   7   11   13   19	@B 11
---	----------------------	----------

2	1   5   9   13   17	@C 01
---	---------------------	----------

2	2   6   10   14   18	@D 10
---	----------------------	----------

f) Il aura fallu 4 pages au total, nous avons 20 enregistrements et chaque page peut en contenir 6.

g)

i. Nous effectuons la recherche à partir d'un index, donc il suffit de lire 1 seule page.

ii. L'attribut "nom" n'a pas d'index, on doit donc parcourir toutes les pages, soit 4.

iii. Pour insérer l'enregistrement d'id 22, il faut calculer son emplacement, on fonction des deux derniers bits. Donc, lecture de bloc@D. On ira lire 1 page.

iv. Nous devons calculer sa position, en fonction des deux derniers bits de son id, soit 1 page sera lue. (car lecture du bloc @D  
✓  
70 )

**Exercice 1 : Organisation séquentielle indexée**

1. Soit la table Film :

Titre	Année
Vertigo	1958
Brazil	1984
Twin peaks	1990
Underground	1995
Easy Rider	1969
Psychose	1960
Greystoke	1984
Shining	1980
Annie Hall	1977
Jurassic park	1992
Metropolis	1926
Manhattan	1979
Reservoir Dogs	1992
Impitoyable	1992
Casablanca	1942
Smoke	1995

- a) Combien de pages sont-elles nécessaires si on stocke les données suivant une organisation d'arbre B d'ordre 2 sur l'attribut Titre ?
- b) Combien de pages sont-elles nécessaires si on stocke les données suivant une organisation d'arbre B+ d'ordre 2 sur l'attribut Année ?

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)) =

## Exercice 1 : Organisation séquentielle indexée

1. Soit la table Film :

Titre	Année
Vertigo	1958
Brazil	1984
Twin peaks	1990
Underground	1995
Easy Rider	1969
Psychose	1960
Greystoke	1984
Shining	1980
Annie Hall	1977
Jurassic park	1992
Metropolis	1926
Manhattan	1979
Reservoir Dogs	1992
Impitoyable	1992
Casablanca	1942
Smoke	1995

- a) Combien de pages sont-elles nécessaires si on stocke les données suivant une organisation d'arbre B d'ordre 2 sur l'attribut Titre ?
- b) Combien de pages sont-elles nécessaires si on stocke les données suivant une organisation d'arbre B+ d'ordre 2 sur l'attribut Année ?

Si l'arbre est d'ordre 2, combien d'enregistrements peut-on avoir par page ?

alors on applique donc,

### Arbre B d'ordre m

Un arbre B d'ordre  $m$  est un arbre tel que

- (i) chaque nœud contient  $k$  clés triées, avec  $m \leq k \leq 2m$  sauf la racine pour laquelle  $k$  vérifie  $1 \leq k \leq 2m$ .
- (ii) tout nœud non-feuille a  $(k+1)$  fils. Le  $i^{\text{ème}}$  fils a des clés comprises les  $(i-1)^{\text{ème}}$  et  $i^{\text{ème}}$  clés du père.
- (iii) l'arbre est équilibré.

Androis disp

on peut avoir  $4 = (2 \times 2^m)$  enreg. / page

→ Pour récupérer la table films, le Sgbd doit récupérer la racine (c'est une page)

Métadonnées : toutes les infos de la base de données

↳ nom des tables

↳ de quels attributs

→ indique où est la racine de la table

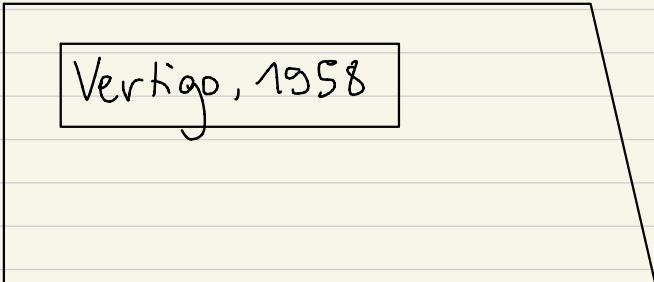
↳ au départ, cette racine est = nil

puis

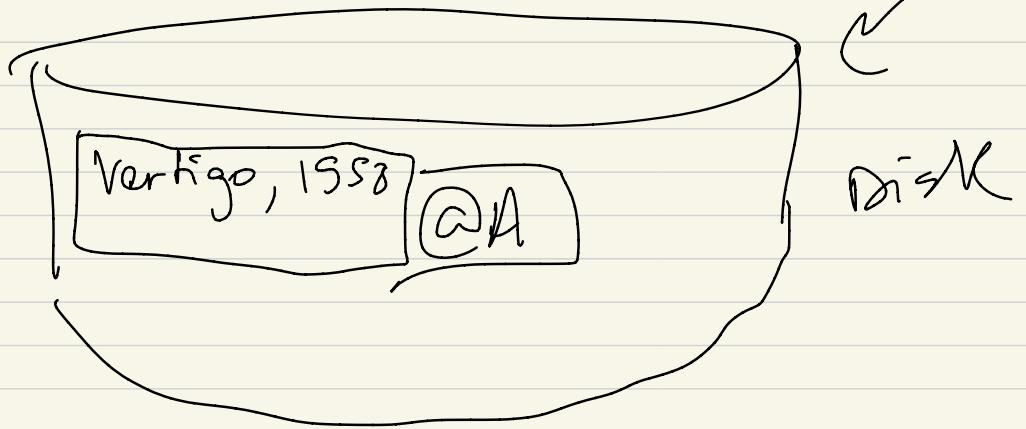
racine = @A

insertion de (Vertigo, 1958)

RAN



lui on stocke par rapport au titre

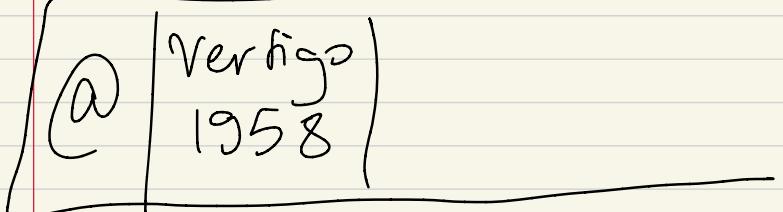


Insert - (Brazil, 1984)



On va insérer juste avant V  
Cap B < V

RAA

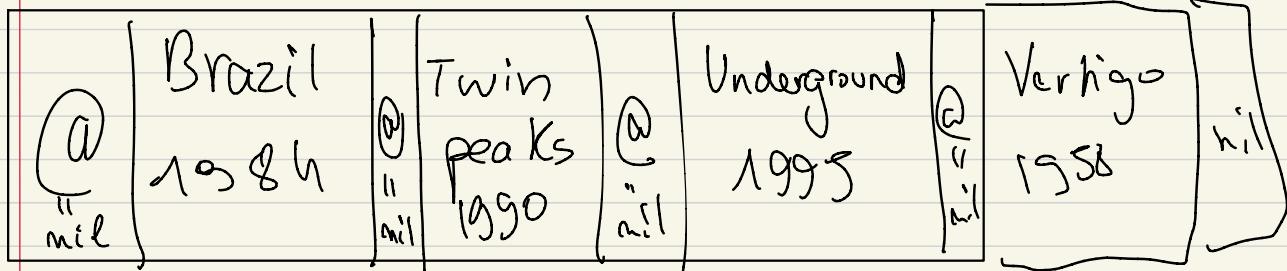


On lit -tjs la page disque,  
puis ajout dans la mémoire  
puis à nv écriture disque

Dick

Ensute, insert  $\Rightarrow$  sequential

- c'est



le noeud raline  
Une page

on veut insérer Easy Rider

mais  $K = 4$  donc impossible

max = 4  
min = 2 par page

→ que faire ? Allocation d'une new page

easy Rider

@A	
(@)	Brazil
"	1984
nil	
(@)	Twin Peaks
"	1990
nil	
(@)	Underground
"	1993
nil	
(@)	Vertigo
"	1958
nil	

@B


$$m = \text{ordre} = 2$$

→ on laisse dans la page @A, les  $m$  enreg.  
① avec les clés plus petites

et on déplace les  $m$  enreg. avec des clés plus grandes à la nouvelle page

② → l'enreg. du milieu au niveau supérieur

↳ ce qui signifie que l'on alloue une nouvelle page

# Etape 1

(@A)	Brazil 1986 "mil	(@B)	Twin peaks 1990 "mil	(@)	Underground 1995 "mil	(@)	Vertigo 1958 "mil	(@)	hill
------	------------------------	------	----------------------------	-----	-----------------------------	-----	-------------------------	-----	------

easy Rider

(@B)	Underground 1995		Vertigo 1958
------	---------------------	--	-----------------

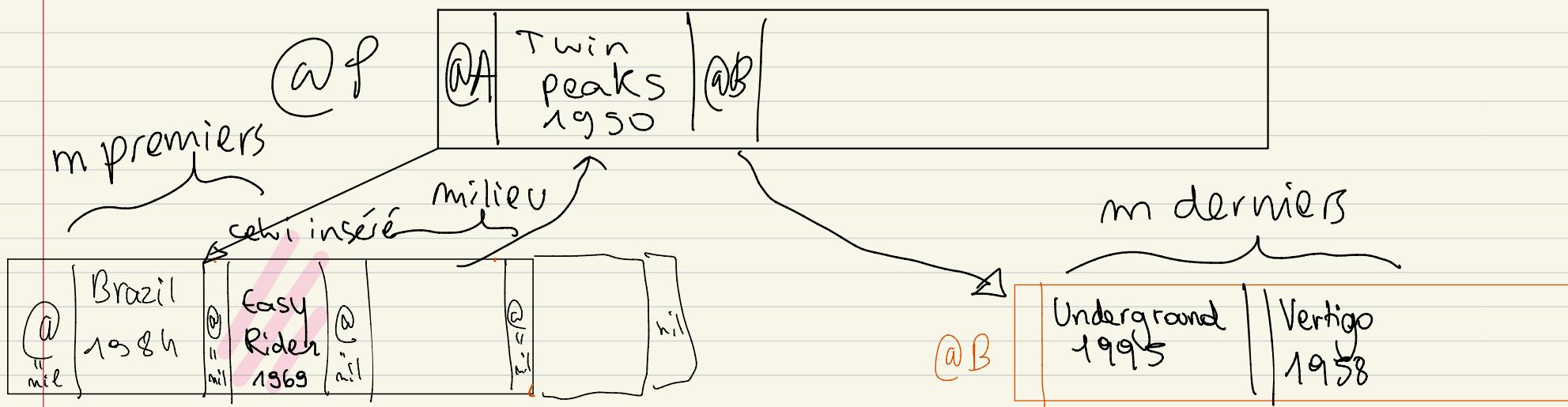
# Etape 2

(@A)	Brazil 1986 "mil	(@B)	Twin peaks 1990 "mil	(@)	Underground 1995 "mil	(@)	Vertigo 1958 "mil	(@)	hill
------	------------------------	------	----------------------------	-----	-----------------------------	-----	-------------------------	-----	------

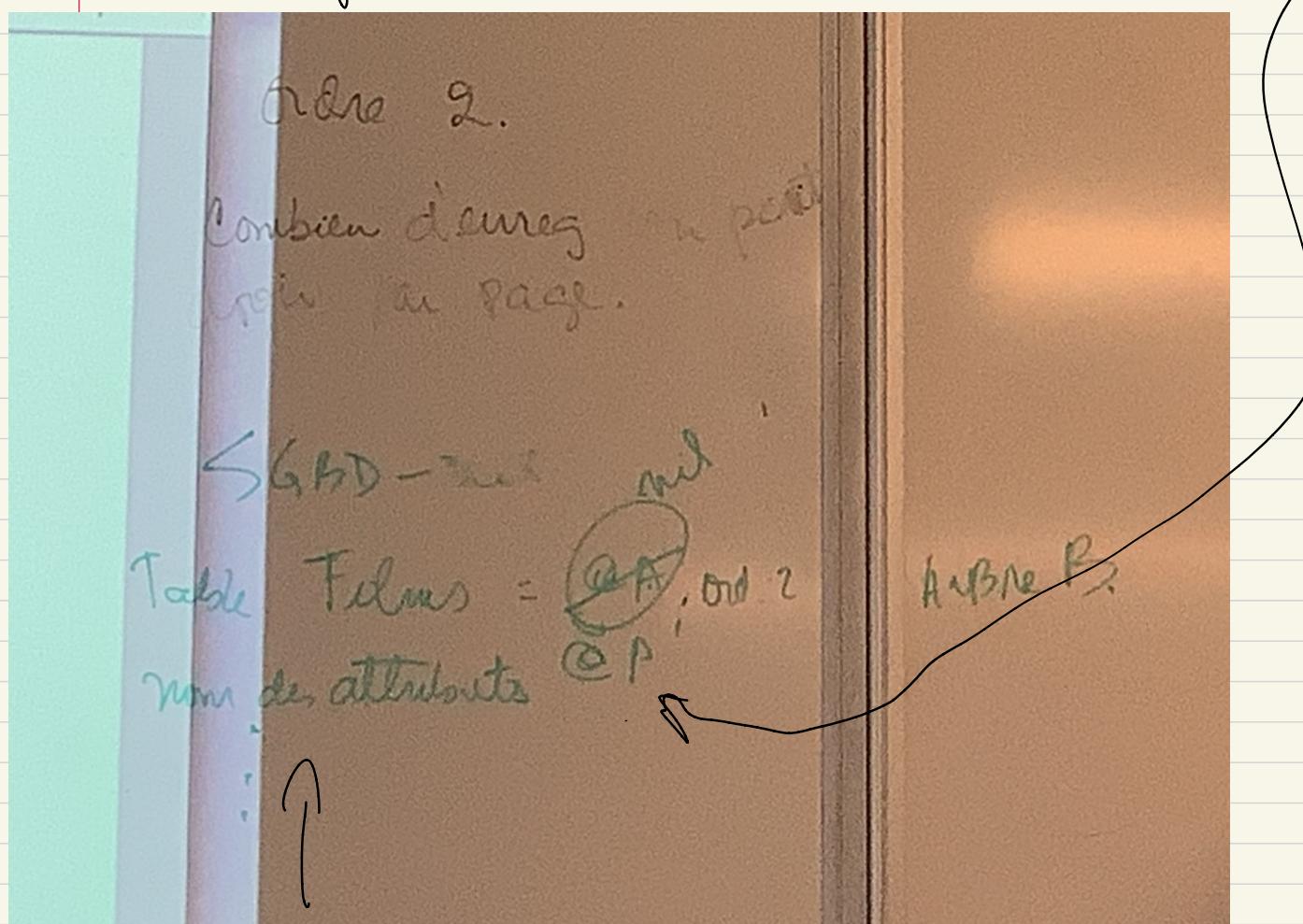
easy Rider

(@B)	Underground 1995		Vertigo 1958
------	---------------------	--	-----------------

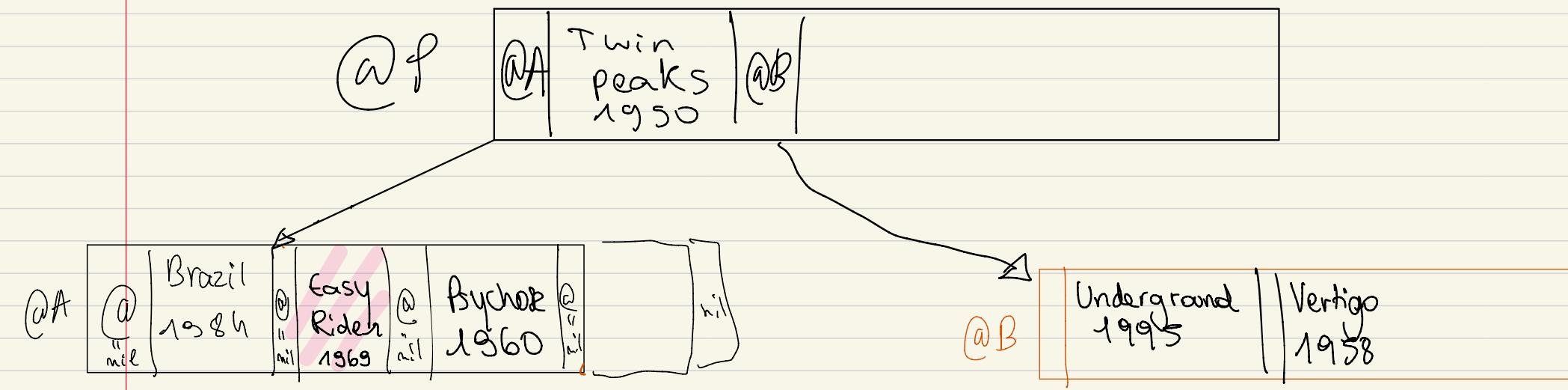
# Dcmc



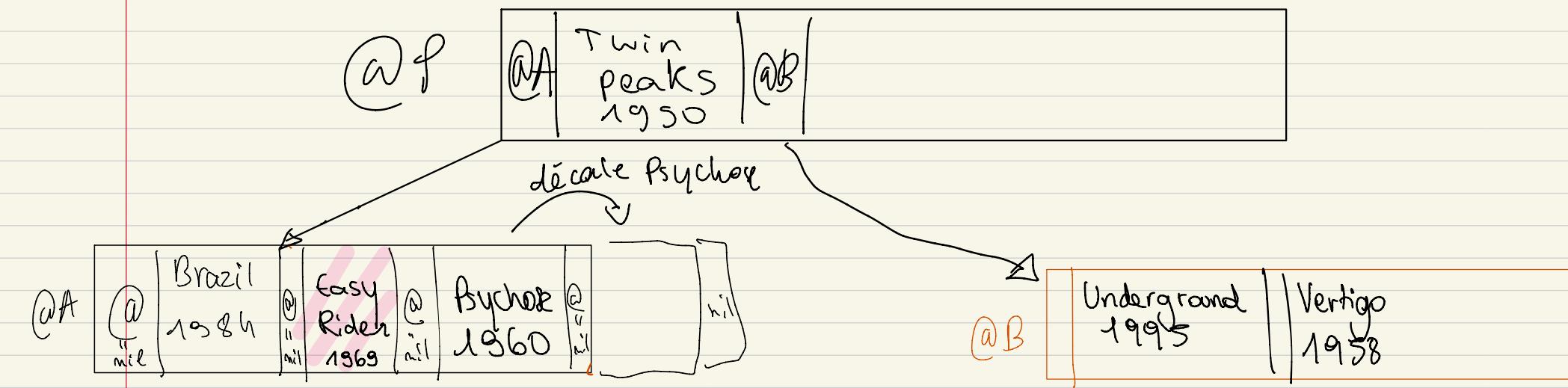
on change dans les métadonnées la racine ;



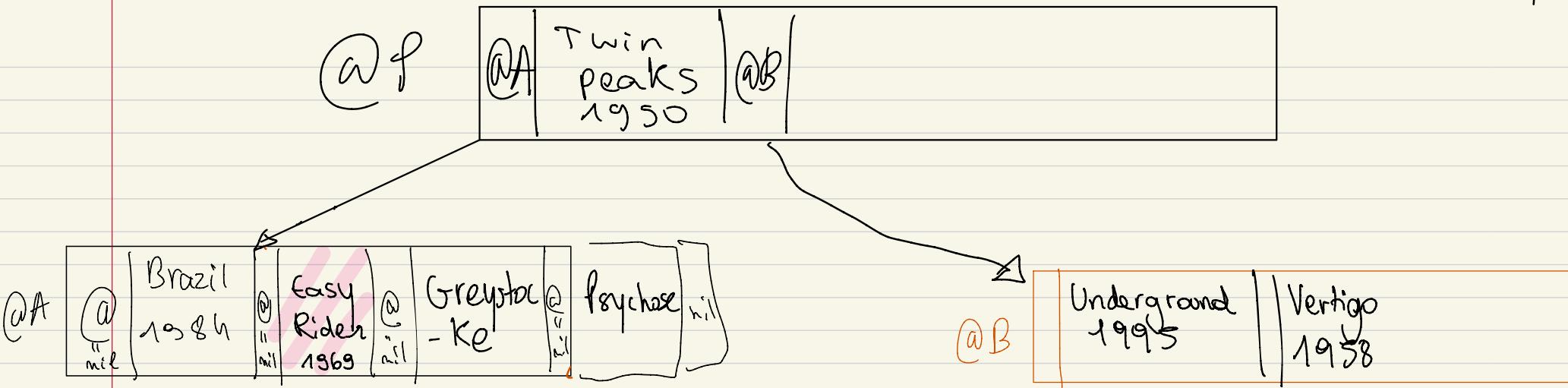
Ces données sont importantes, car le SGBD  
cherche l'@ de la racine afin d'insérer (Psychose)



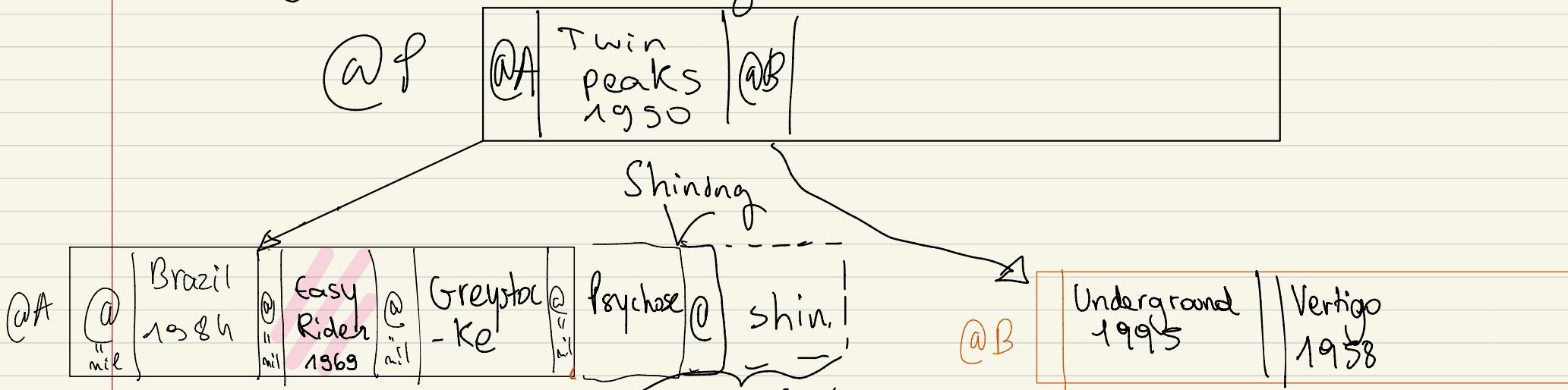
## Insertion de Grey Stocke



C'est bien entre 1 et  $2 \times 2 = 4$

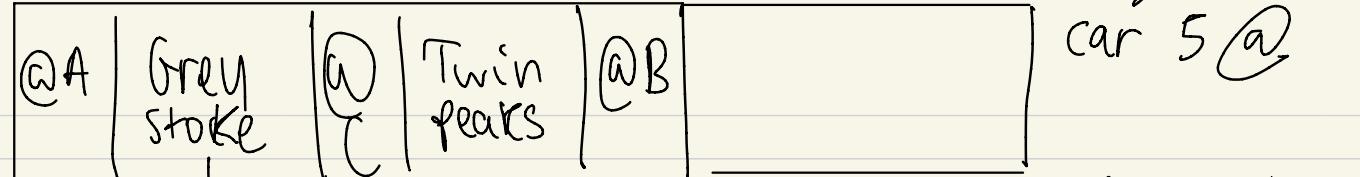


Ajout de Shining ( $S < T$ )

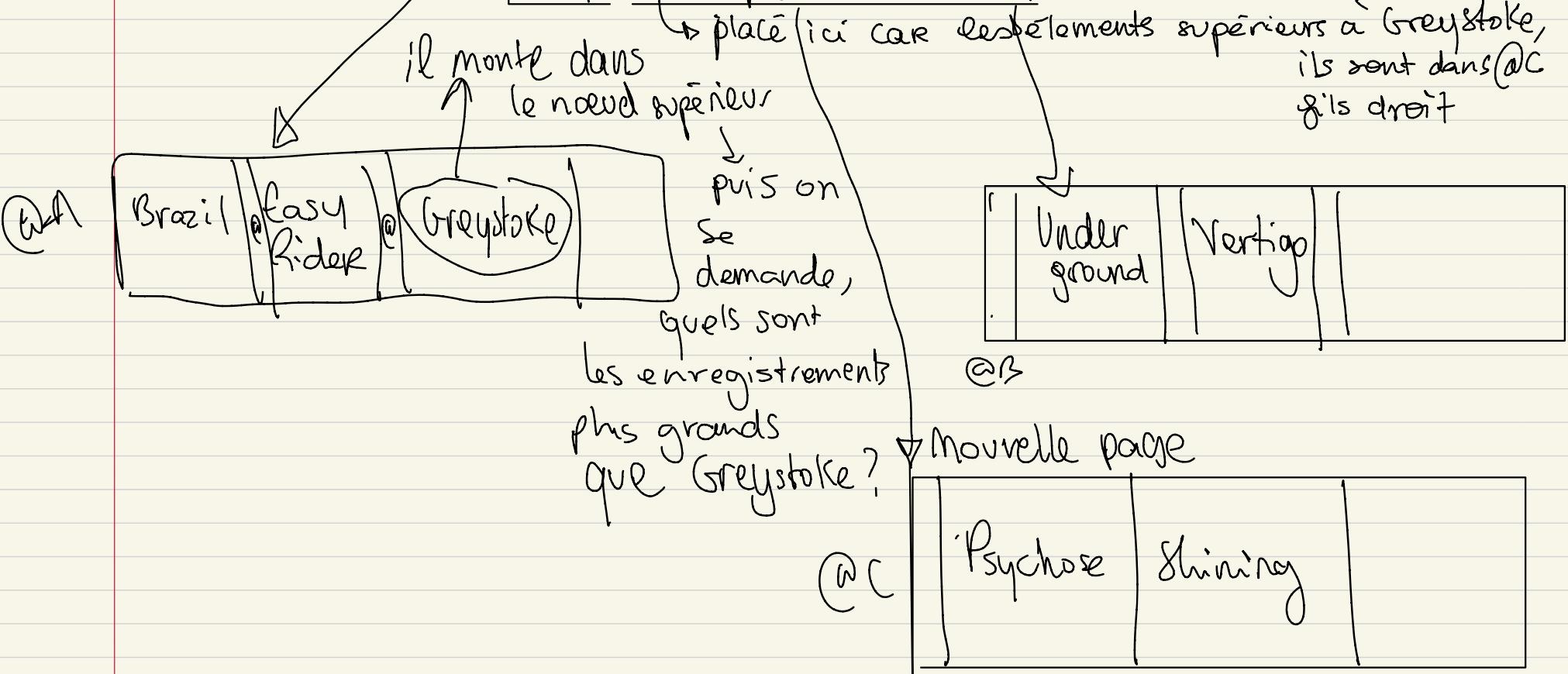


Création d'une nouvelle page

on divise les m premiers



► 5 fits  
car 5 @



Au propre :

1

@A	Grey stroke	@C	Twin peaks	@B	
----	----------------	----	---------------	----	--

(@A)

Brazil	Easy Rider	@	
--------	---------------	---	--

	Under ground	Vertigo	
--	-----------------	---------	--

@B

Mouvelle page

(@C)

	Psychose	Shining	
--	----------	---------	--

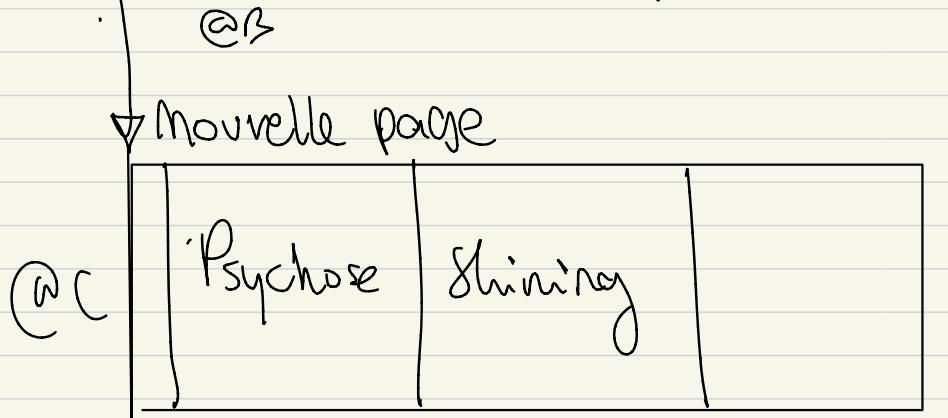
Ensuite, on insère  
Annie Hall

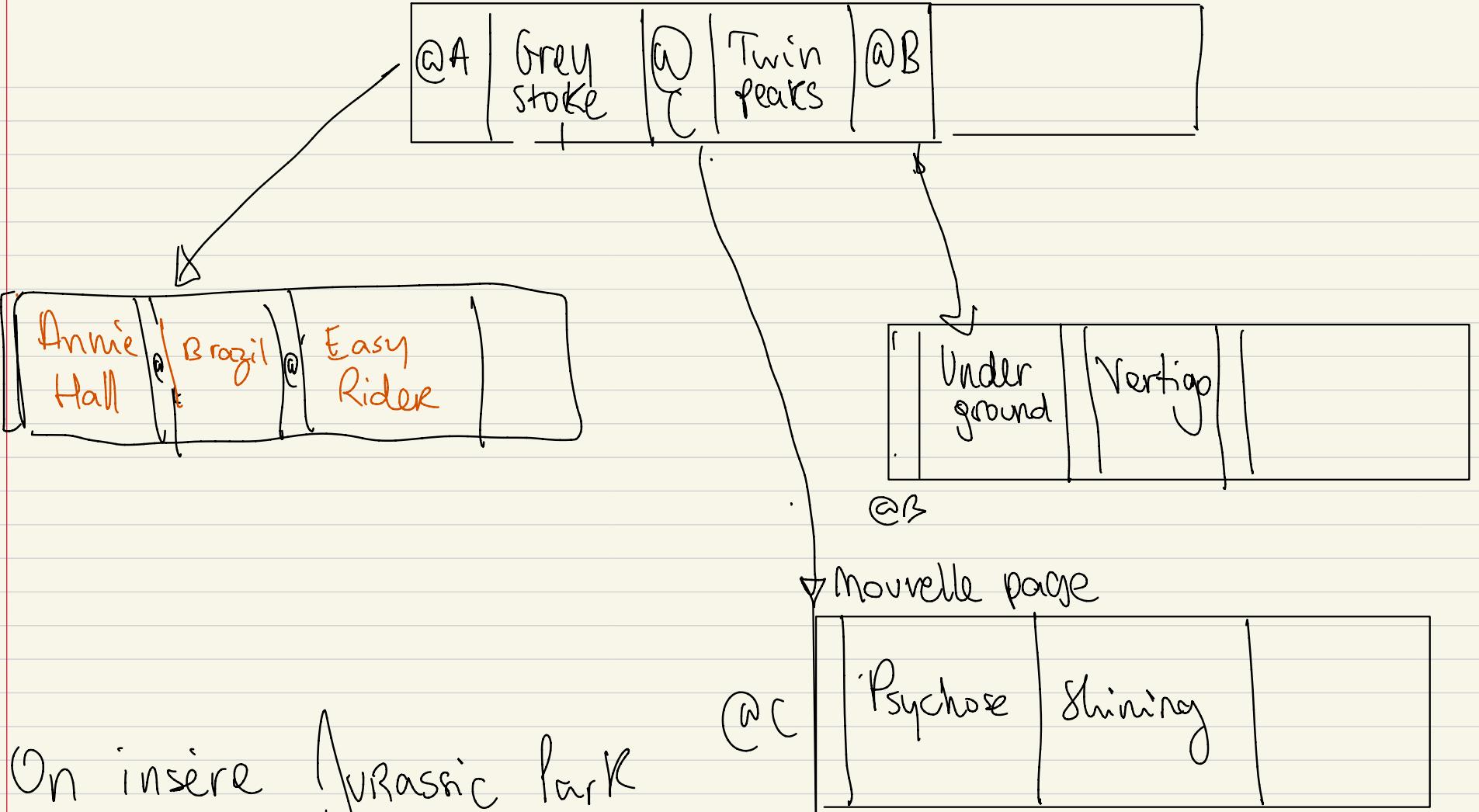
@A	Grey stroke	(C)	Twin peaks	@B	
----	----------------	-----	---------------	----	--

Annie Hall



Under ground | Vertigo





G ↗ f

→ on insère dans les feuilles  
donc fils droit (plus grand) de G, c'est @C

@A	Grey stroke	@C	Twin peaks	@B	
----	----------------	----	---------------	----	--

@A

Annie Hall	Brazil	Easy Rider
------------	--------	------------

Under ground	Vertigo
-----------------	---------

@B

→ nouvelle page

@C

Jurassic Park	Psychose	Shining
---------------	----------	---------

insertion de Metropolis

31 > 6

@A	Grey stroke	@C	Twin peaks	@B	
----	----------------	----	---------------	----	--



↓

	Under ground	Vertigo	
--	-----------------	---------	--

↓ Nouvelle page

↓

@B

@C	Jurassic Park	Metropolis	Psychose	Shining
----	------------------	------------	----------	---------

- \* Je cherche Casablanca, combien de pages j'en lis?
  - 1 page (racine)
  - 1 page (@A)

Insert 'Manhattan' :

G < M

Il s'agit de @C → où placer Manhattan?

@A	Grey stroke	@C	Twin peaks	@B	
----	----------------	----	---------------	----	--

@A

Annie Hall	Brazil	Easy Rider
------------	--------	------------

Under ground	Vertigo
-----------------	---------

@B

Jurassic Park	Metropolis	Psychose	Shining
---------------	------------	----------	---------

@C

Manhattan

m dommiers

@P

@A	Grey Stake	Metropolis	@D	Twin Peaks	@B
----	---------------	------------	----	---------------	----

@A

Annie Hall	Brazil	Easy Rider	
---------------	--------	---------------	--

@B

Under ground	Vertigo
-----------------	---------

@C

Jurassic Park	Manhattan	Metropolis
------------------	-----------	------------

on le monte  
au niveau sup (donc @P)

→ car ils sont  
tous des fils de m<sup>n</sup>  
niveau

↔

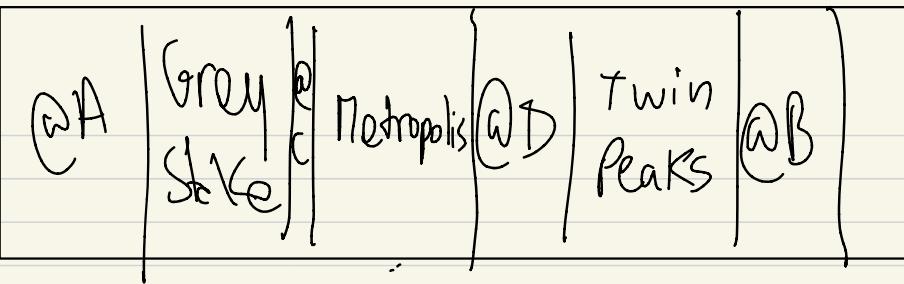
m premiers

@D

Psychose	Shining
----------	---------

↔  
m derniers

@P



@A

Annie Hall	Brazil	—	Easy Rider		
---------------	--------	---	---------------	--	--

@B

	Under ground	)	Vertigo	
--	-----------------	---	---------	--

@C

Jurassic Park	Manhattan	—		
------------------	-----------	---	--	--



m premiers

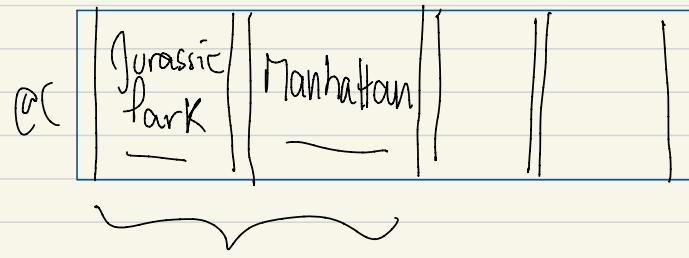
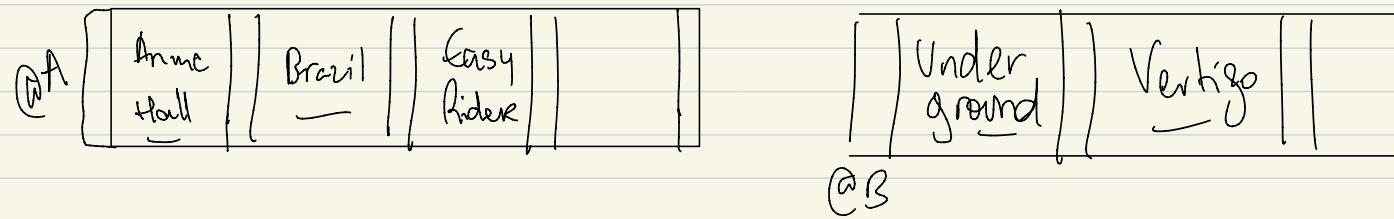
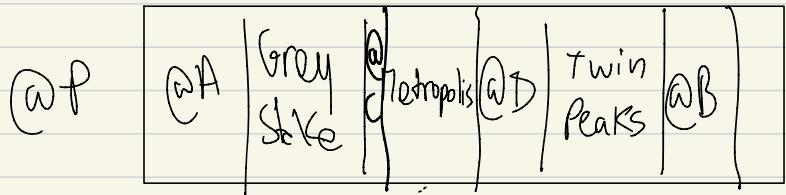
@D

Psychose	Shining	—	
----------	---------	---	--

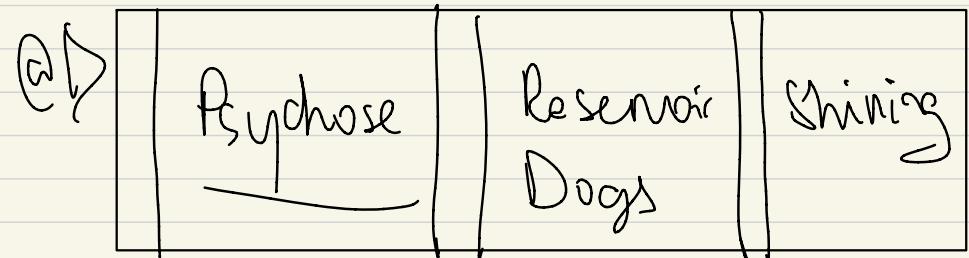
Quelle est la hauteur de l'arborescence ? 2 (avec la racine)  
Insertion de Reservoir Dogs

R < T

↳ donc insertion dans fils



mes premiers



Insert de Impitoyable GKI

droit fil, droit de G (@ C

@P	@A	Grey Stake	@C	Metropolis	@D	Twin Peaks	@B
----	----	---------------	----	------------	----	---------------	----

@A	Anne Hall	Brazil	Easy Rider			Under ground	Vertigo	
					@B			

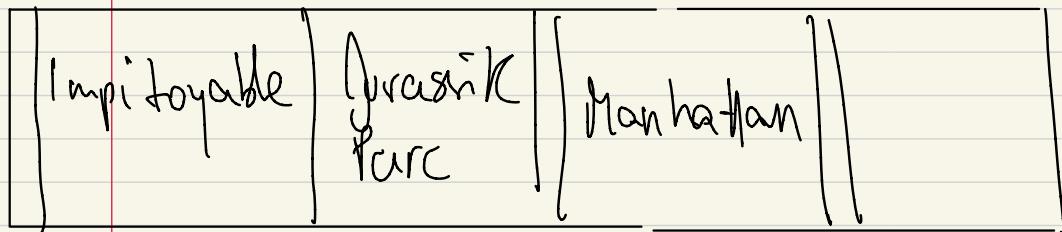
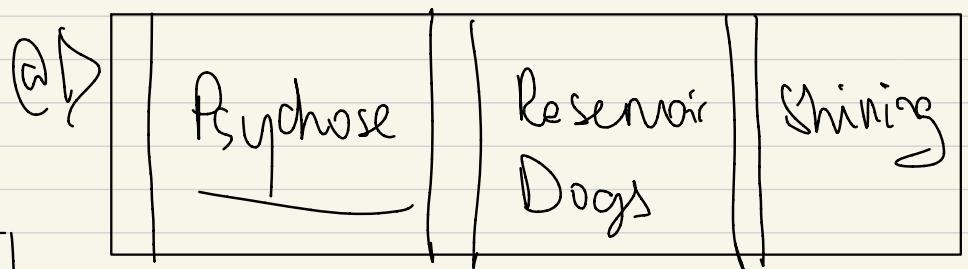
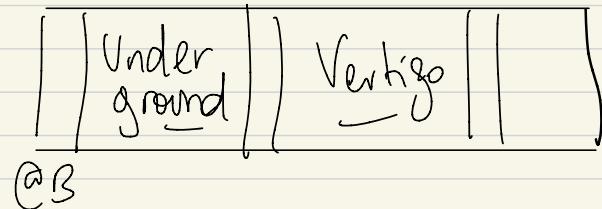
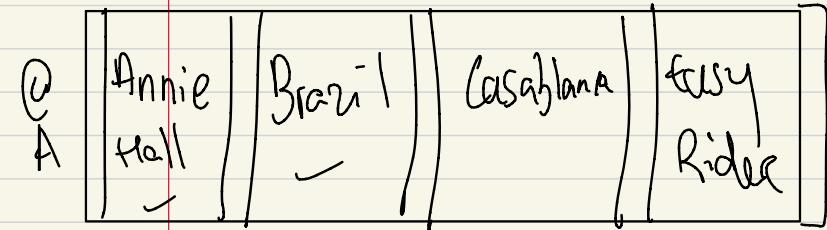
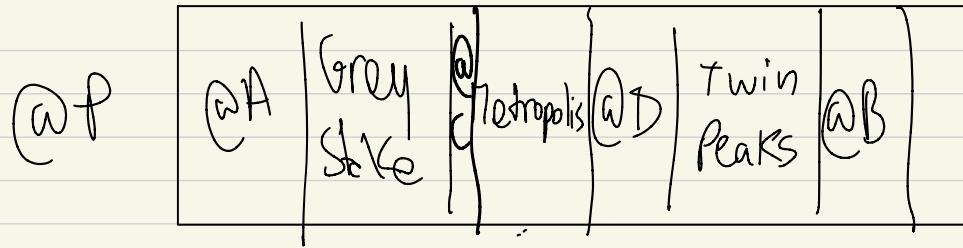
II < G <

Impitoyable	Jurassic Park	Manhattan
-------------	------------------	-----------

@D	Psychose	Reservoir Dogs	Shining
----	----------	-------------------	---------

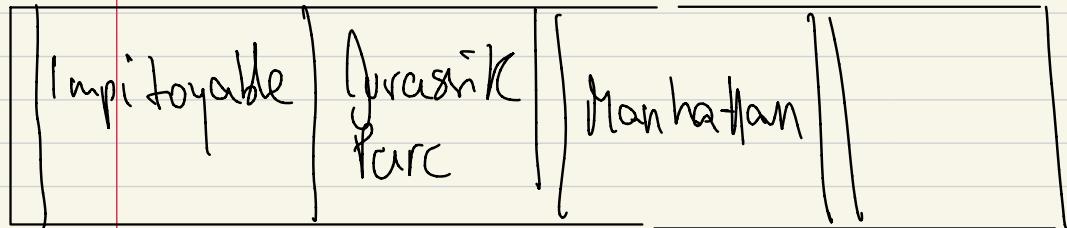
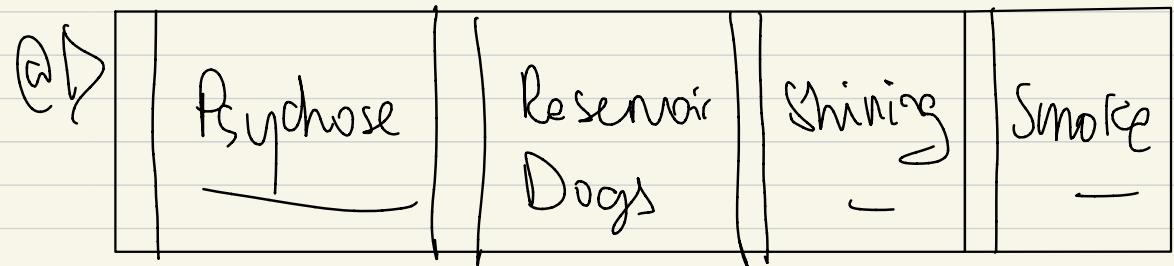
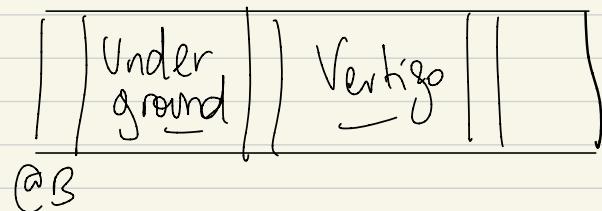
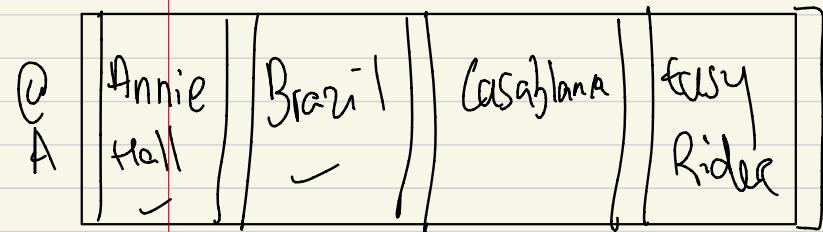
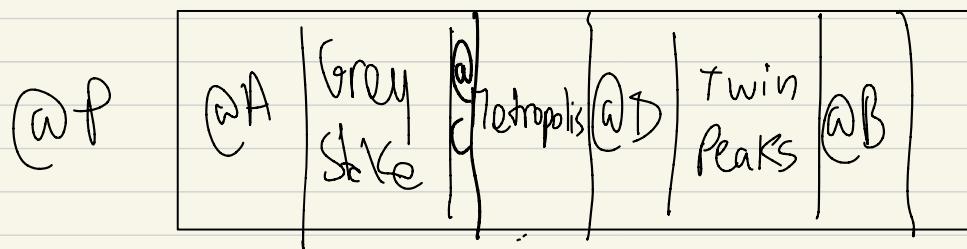
insertion de Casablanca

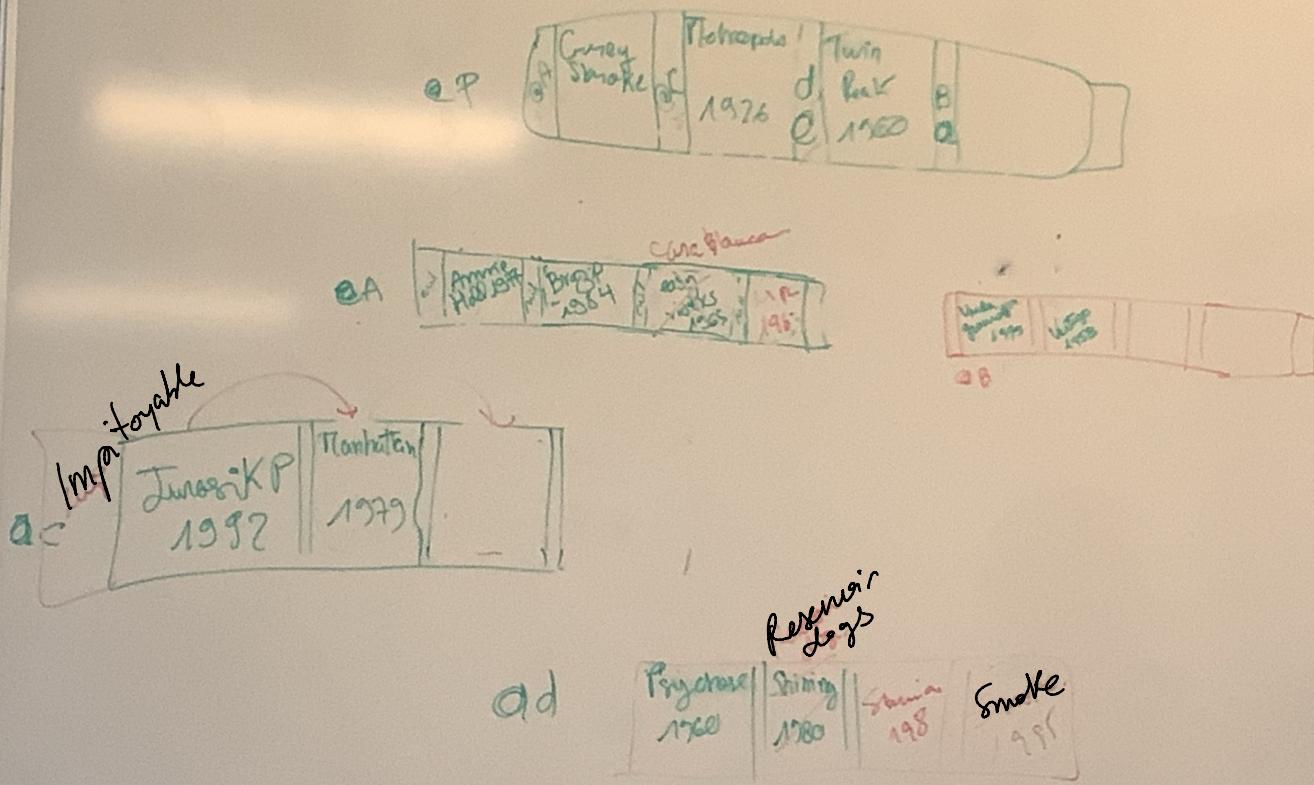
C < G → @ A



Inserción de Smoke

SCT  
→ @ D





Quelle est la structure du fichier )  
↳ arborescente

On a un  
arbre  
équilibré

ce qui est intéressant ici avec cette structure, c'est la recherche qui est rapide (parcours de l'arbre en profondeur)

## Exercice 1 : Organisation séquentielle indexée

1. Soit la table Film :

Titre	Année
Vertigo	1958
Brazil	1984
Twin peaks	1990
Underground	1995
Easy Rider	1969
Psychose	1960
Greystoke	1984
Shining	1980
Annie Hall	1977
Jurassic park	1992
Metropolis	1926
Manhattan	1979
Reservoir Dogs	1992
Impitoyable	1992
Casablanca	1942
Smoke	1995

b) 4 clés  
5 adresses

Dans une page, on peut stocker 3 enreg./  
par page

C'est arbitraire

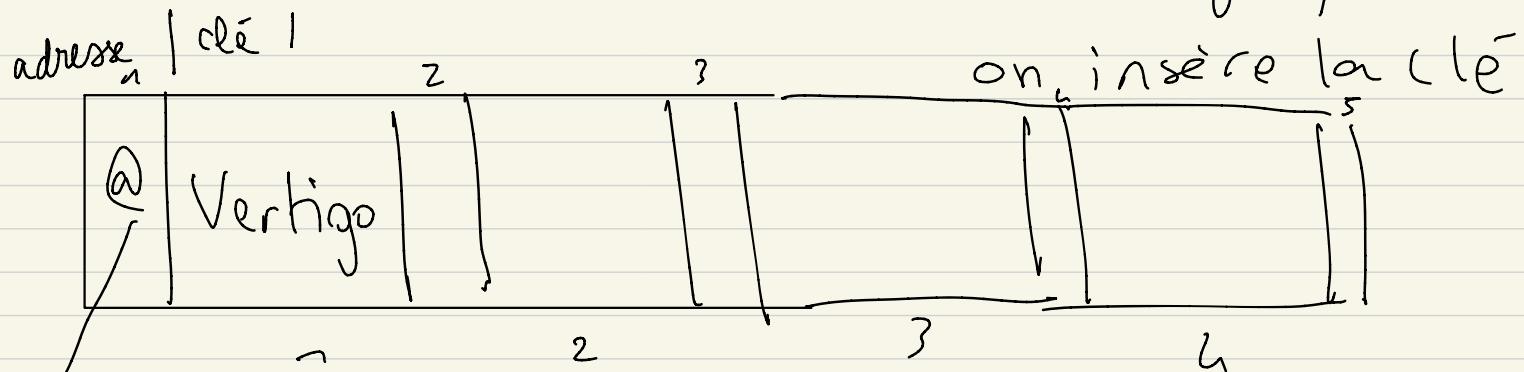
- stockage primaire
- II)
- a) Combien de pages sont-elles nécessaires si on stocke les données suivant une organisation d'arbre B d'ordre 2 sur l'attribut Titre ?
  - b) Combien de pages sont-elles nécessaires si on stocke les données suivant une organisation d'arbre B+ d'ordre 2 sur l'attribut Année ?

→ Arbre B+ afin de stocker les données

RAM :

On va faire le stockage sur le titre ( $\neq$  l'année)

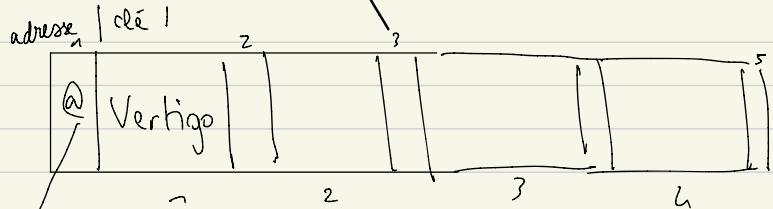
Insertion de (Vertigo, 1958)



CA

Vertigo  
1958

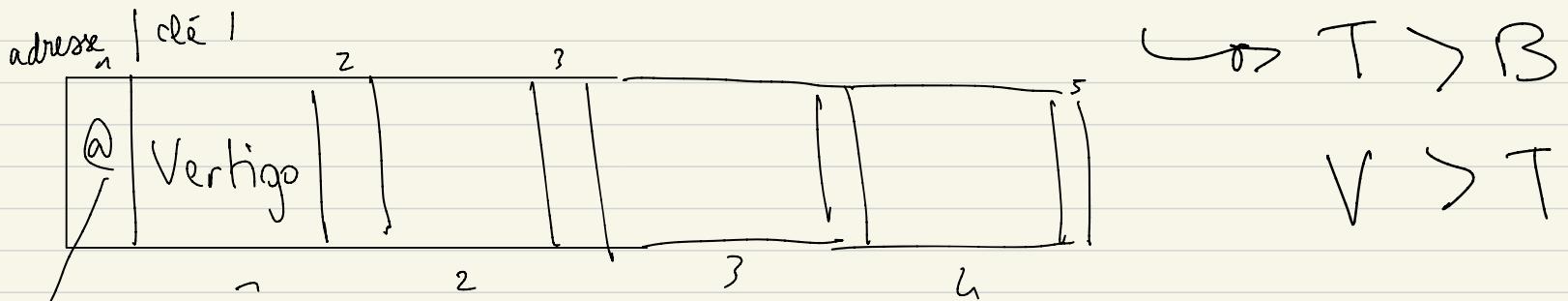
Insert° de (Brazil 1984)



CA

Brazil 1984  
vertigo 1958

# insert° (Twin peaks 1990)



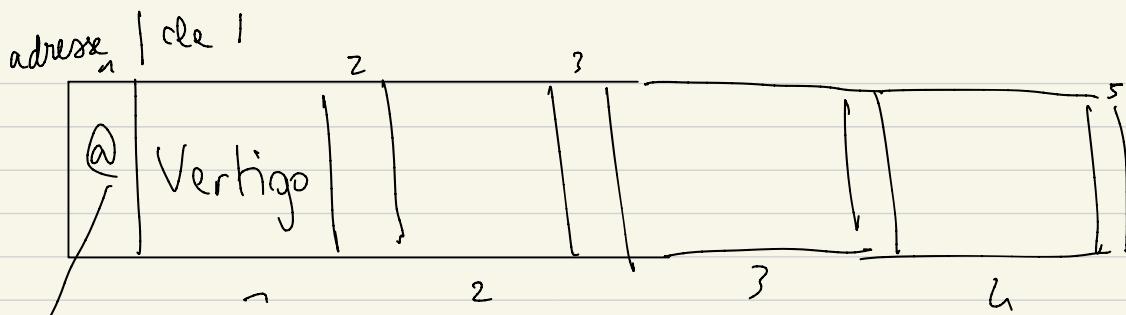
@A

Brazil 1984  
Twin Peaks 1990  
Vertigo 1958

# insertion (Under ground 1995) U < V

→ Plus de place

→ Trouver la place



@A

Brazil 1984

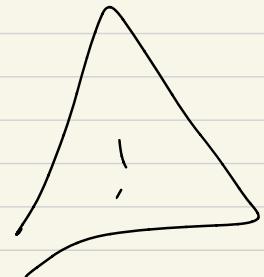
Twin Peaks 1990

Vertigo 1958

@P

Underground

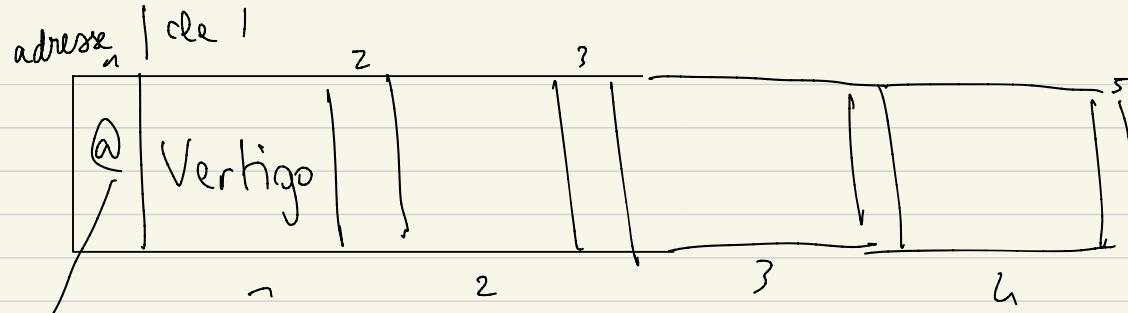
Underground



Obligatoirement 2<sup>ou</sup>4 enregistr. par page

→ Il doit être pair

a pas celui du milieu pour la découpe, sinon on

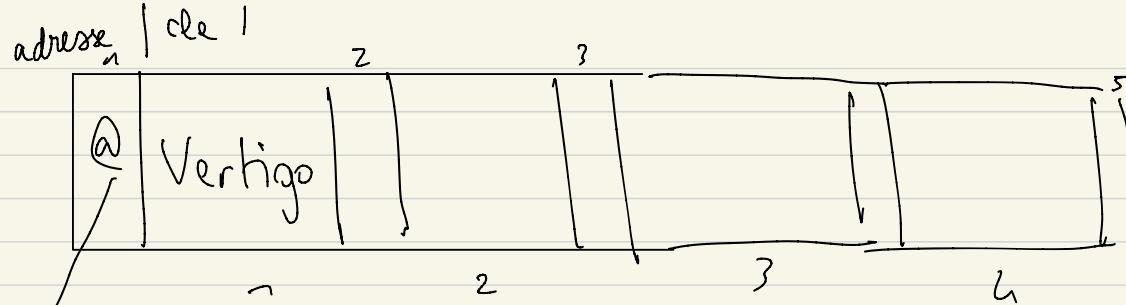


WA

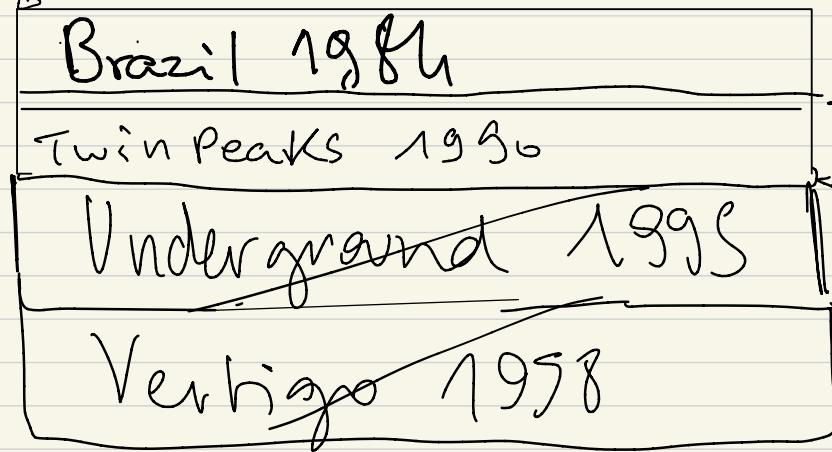
Brazil 1984  
Twin Peaks 1990  
Underground 1993  
Vertigo 1958

insert "easy rider"

→ Plus dc place

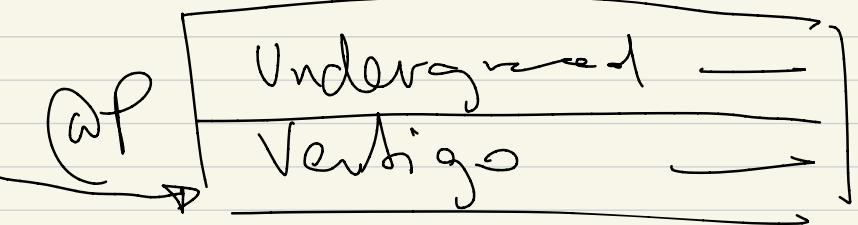


@A



Easy Rider

@P



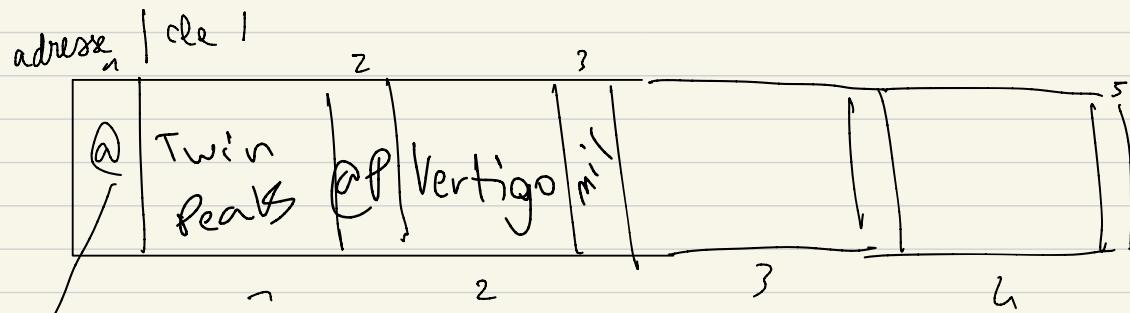
pointeur

car liste chaînée

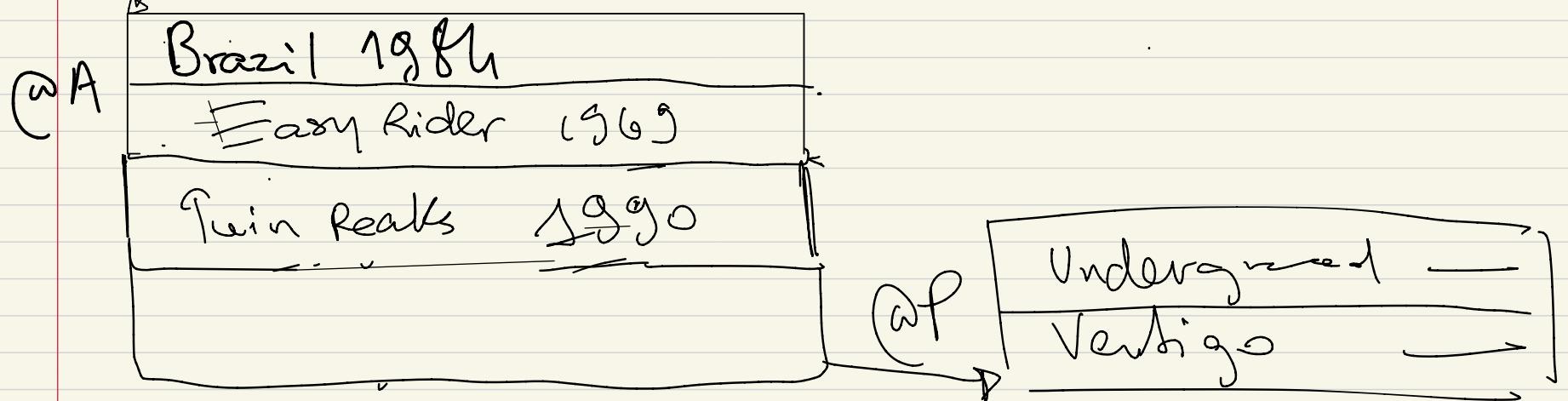
Tusi

Twin Peaks est le milieu de tout

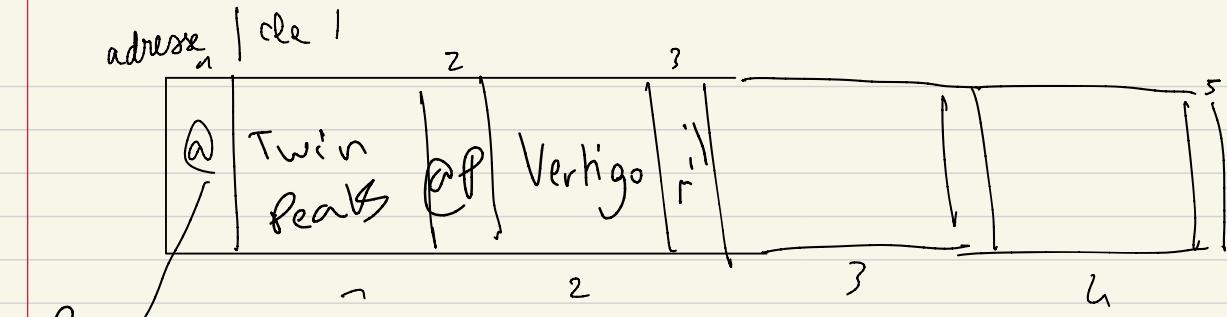
↳ on monte uniquement la clé



AP

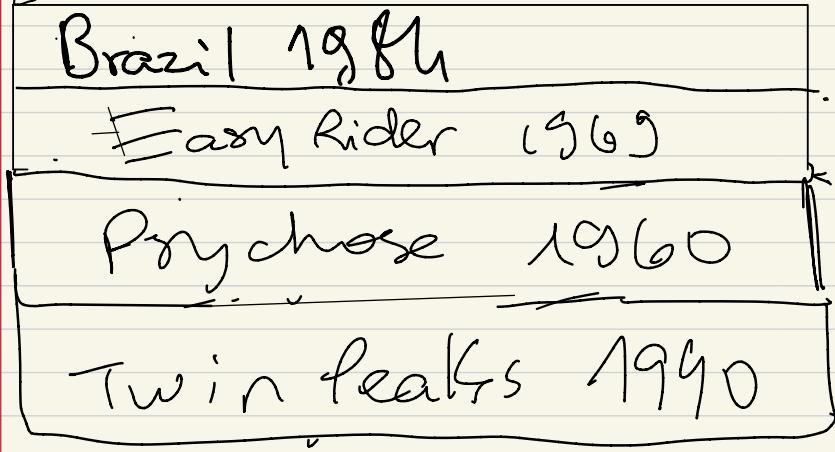


Insert Psychose (1960)

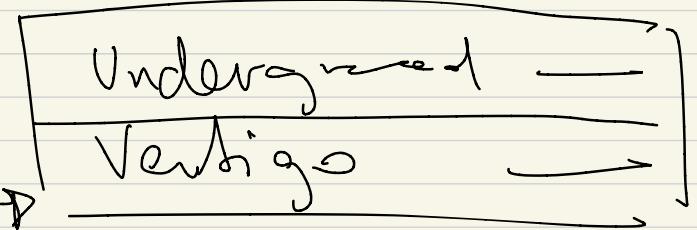


RP

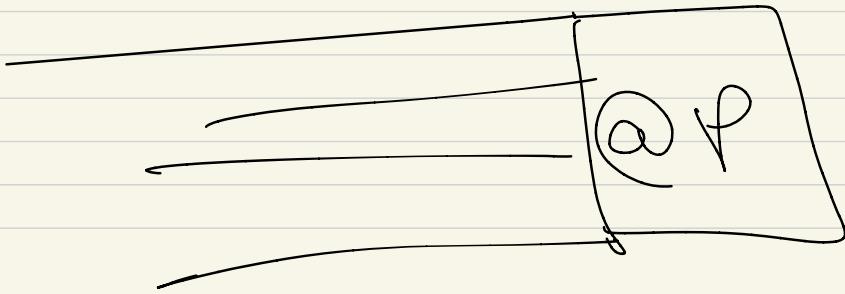
@A



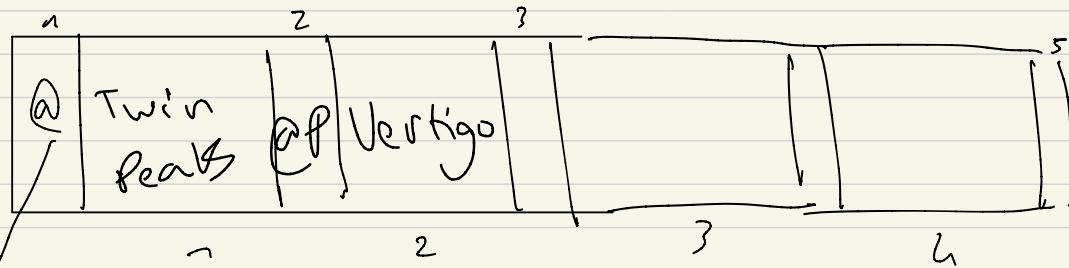
af



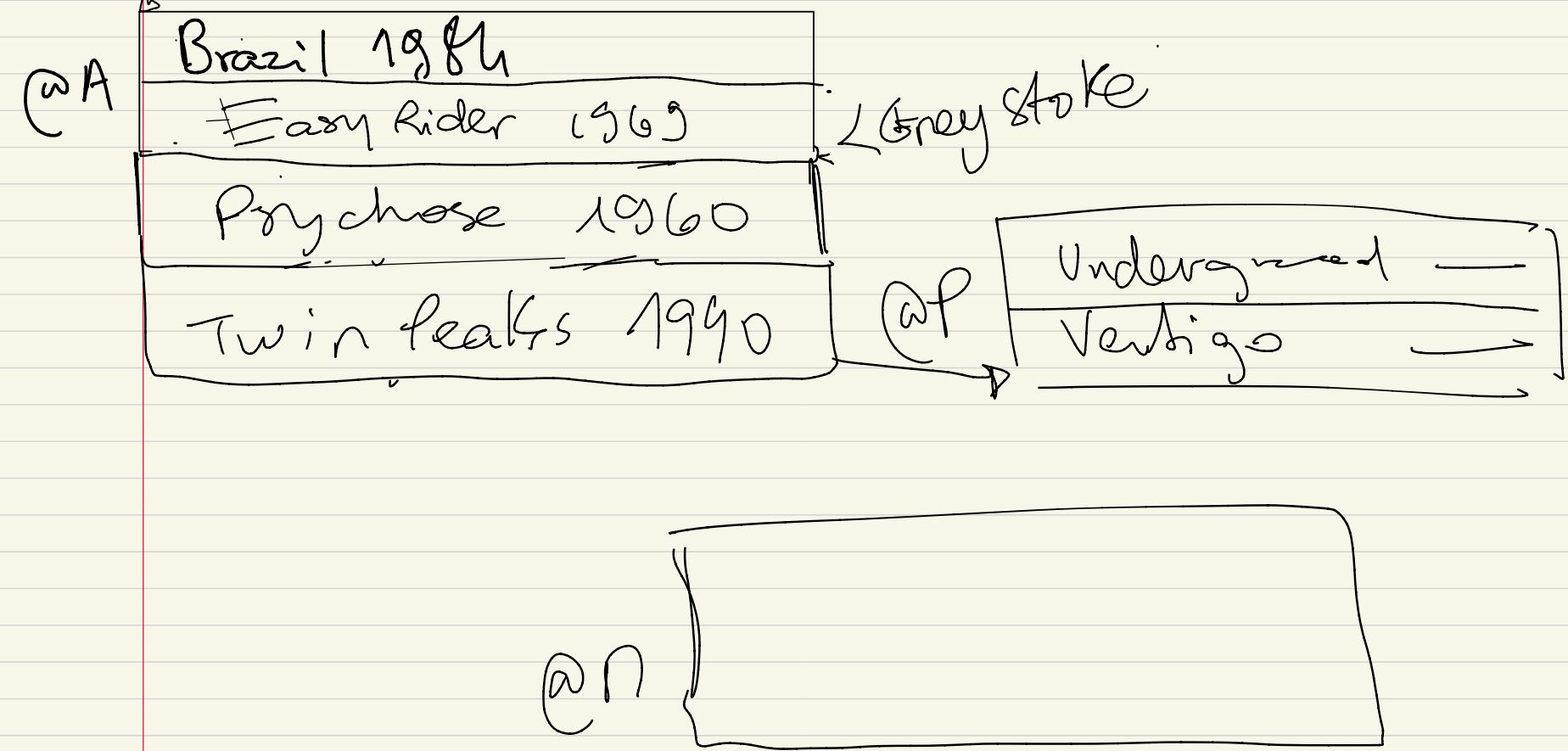
↓ autre manière de noter  
le pointeur



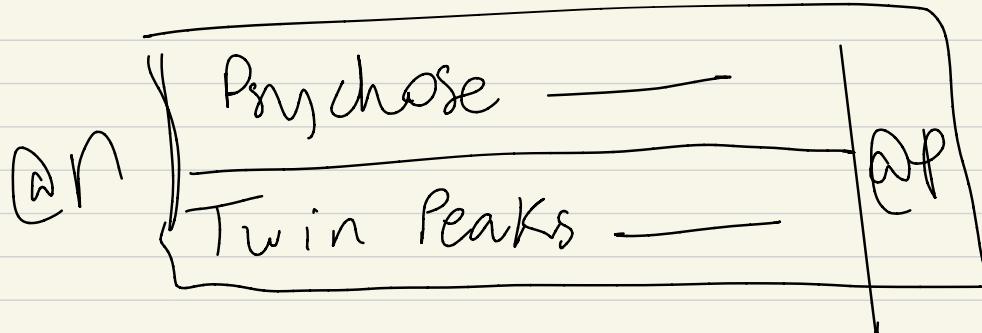
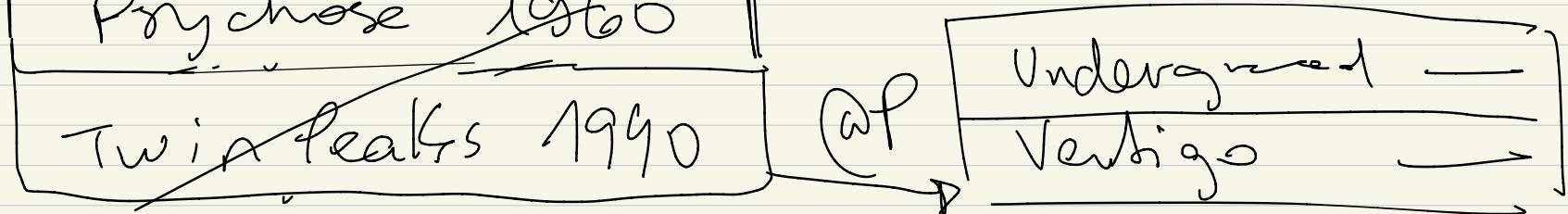
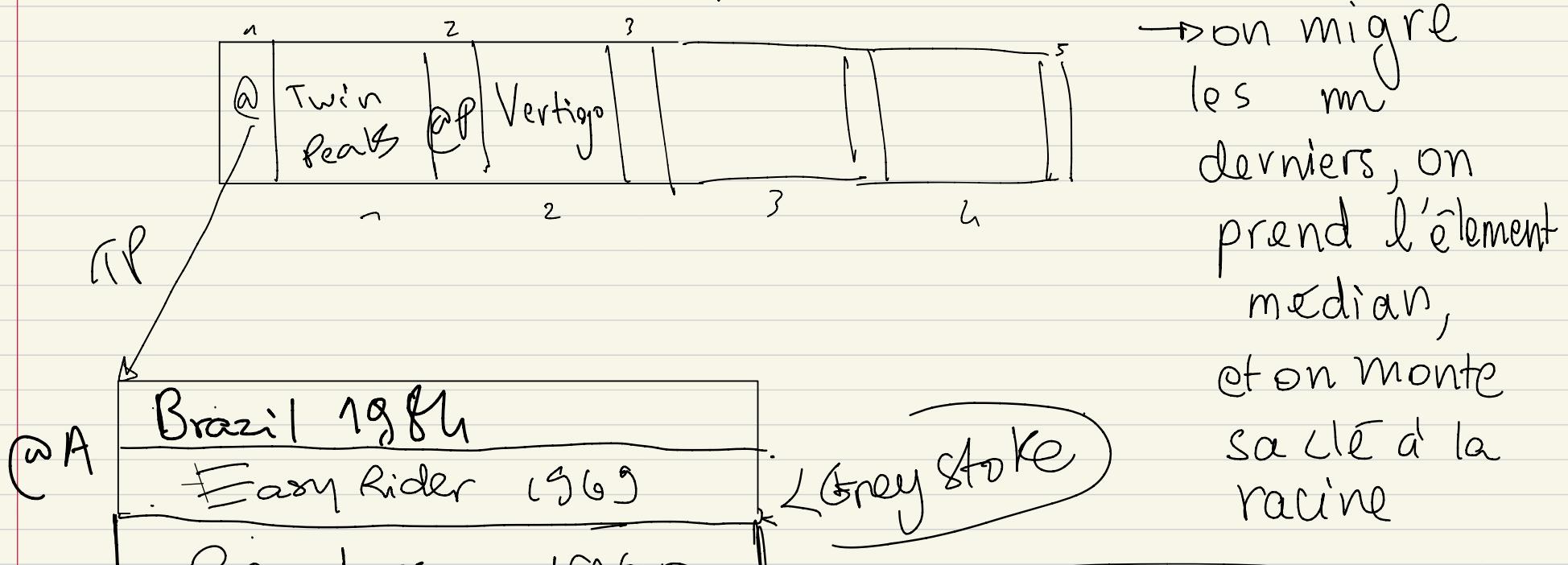
# Insertion de Grey Stocke (1984) : étape 1



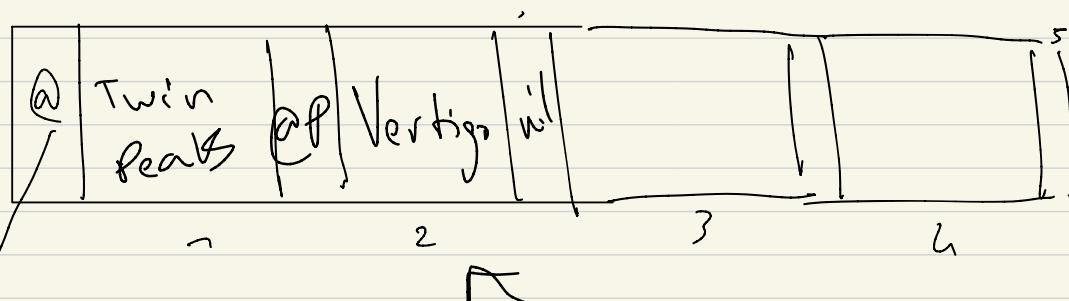
→ Allocation  
nouvelle feuille  
(page) d'@ M  
qui va se placer  
entre @ A et @ P



# Insertion de Grey Stocke (1984) : étape 2

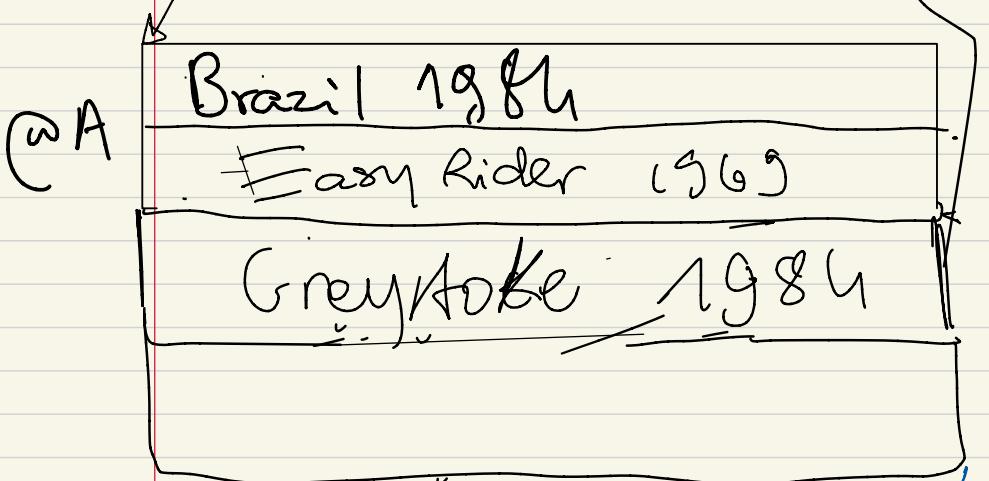


# Insertion de Greystoke (1984) : étape 2

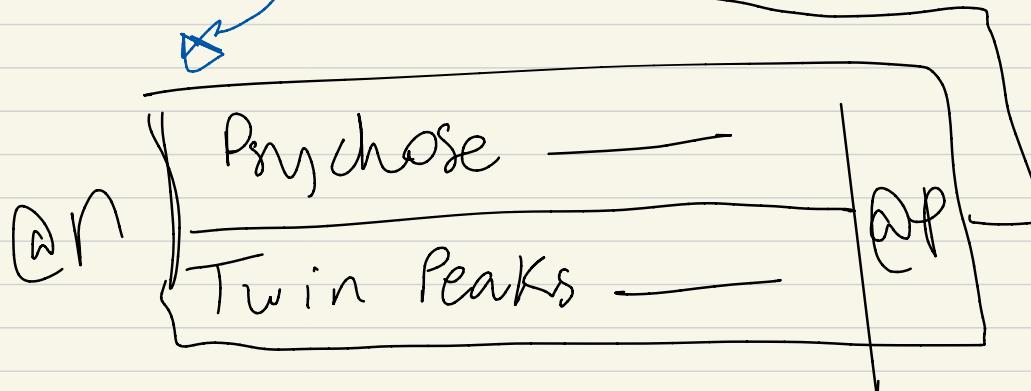
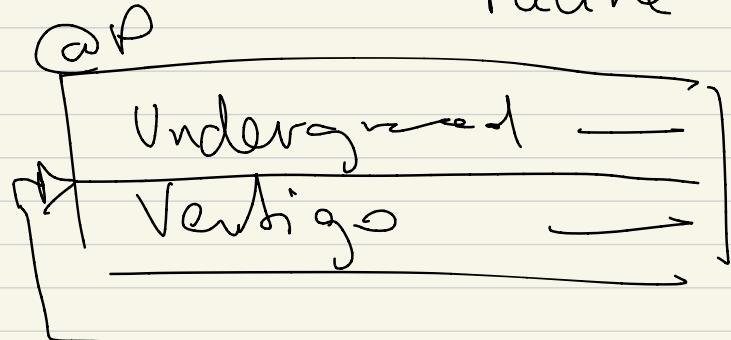


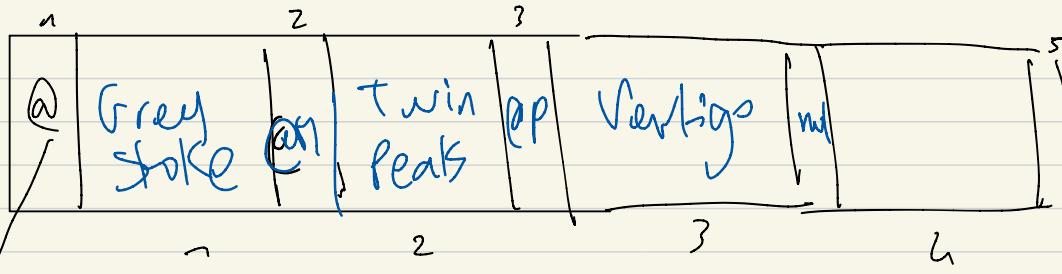
→ on migre  
les m  
derniers, on  
prend l'élément  
médian,

et on monte  
sa clé à la  
racine

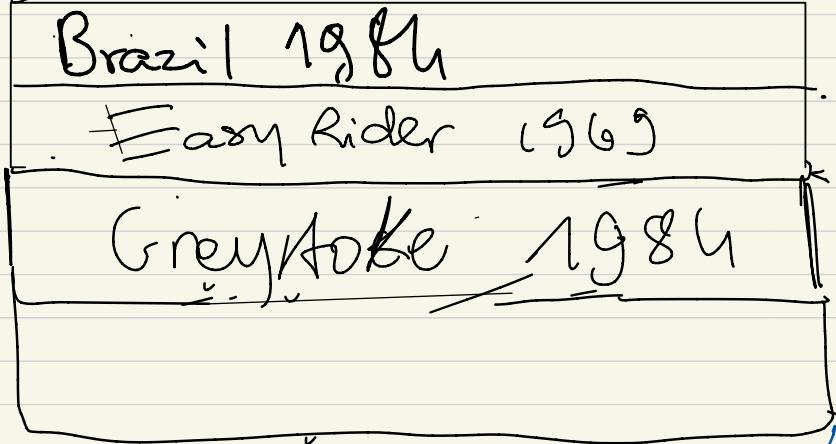


je monte

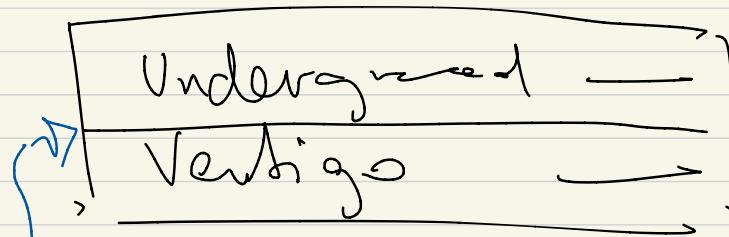




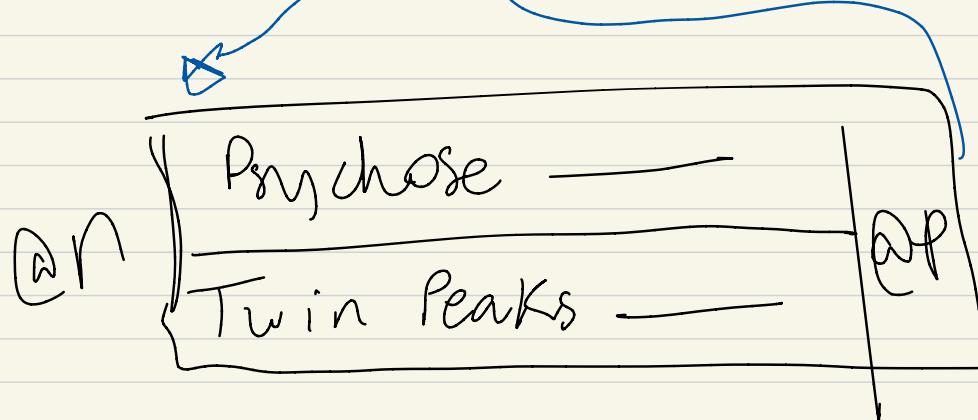
@A



de monte



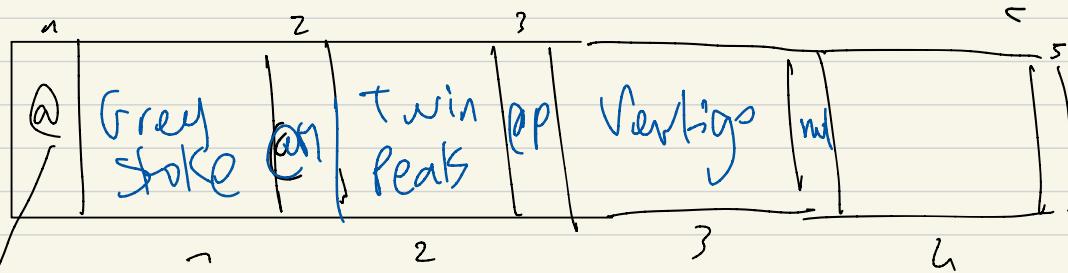
ap



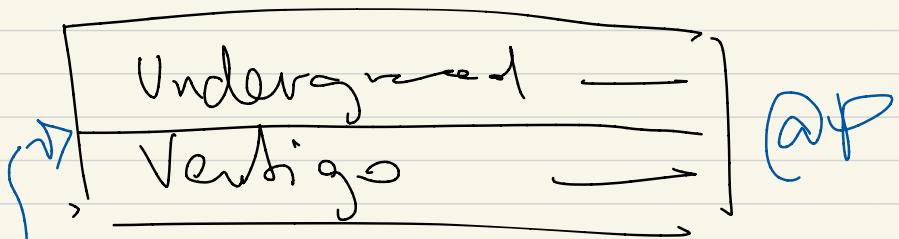
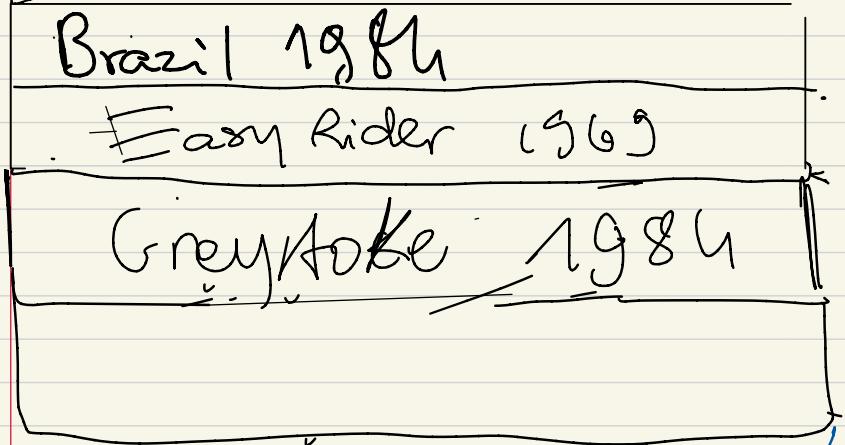
@R

ap

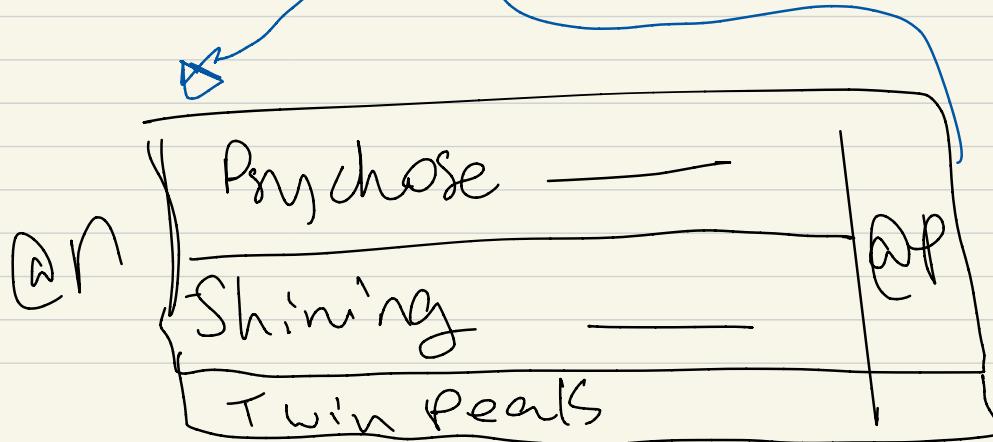
Insertion Shining = ~~S F V~~ S L T (



@A



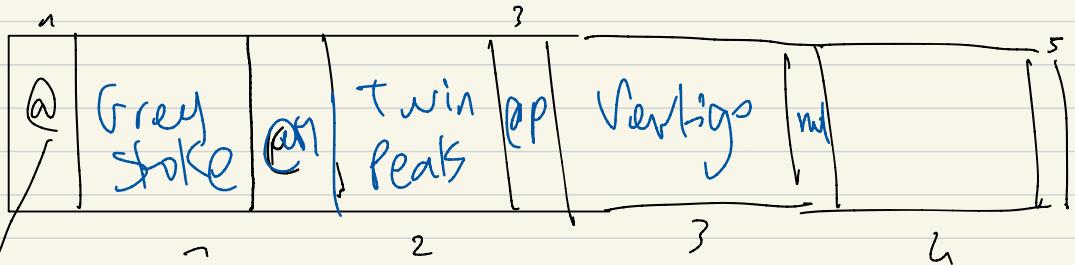
@P



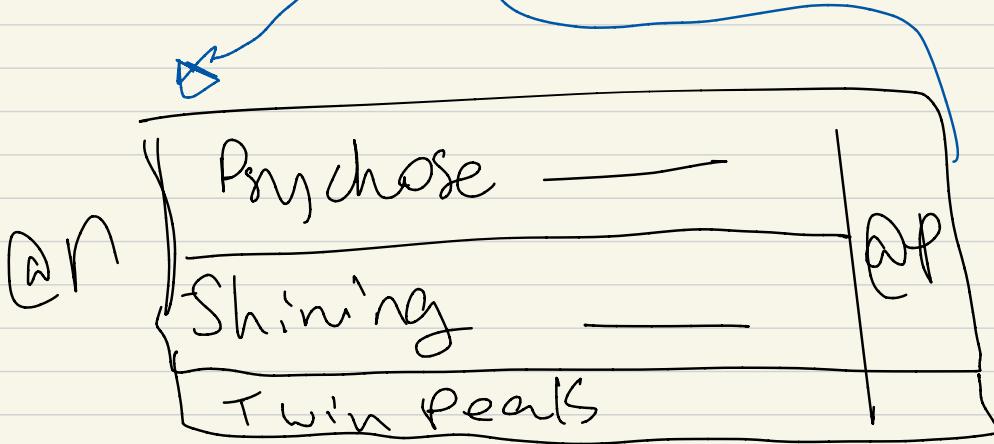
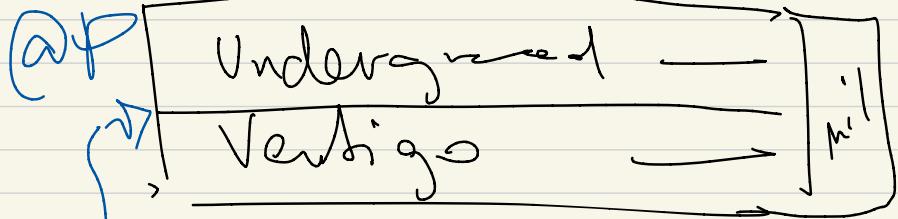
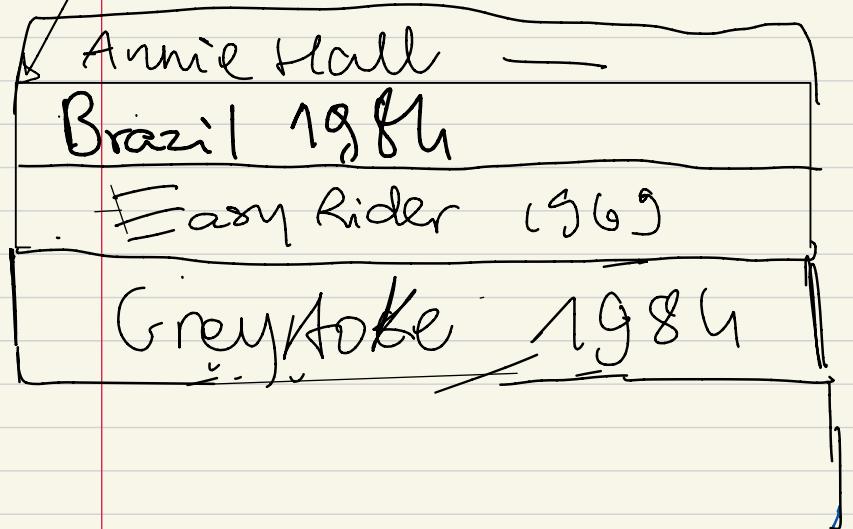
@R

@P

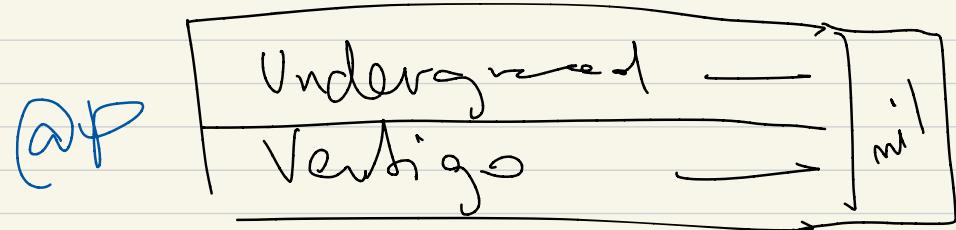
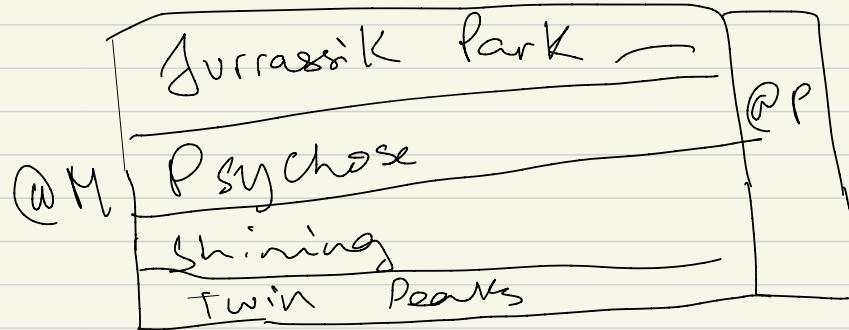
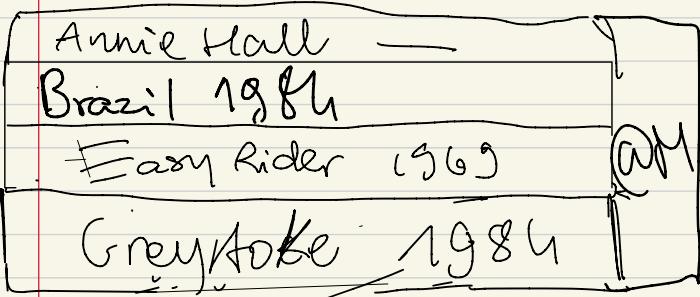
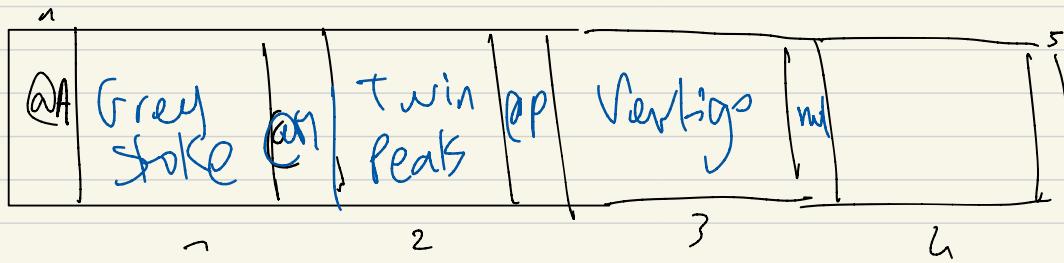
# Insert: Annie Hall



A



→ Jurassic Park → J, > G



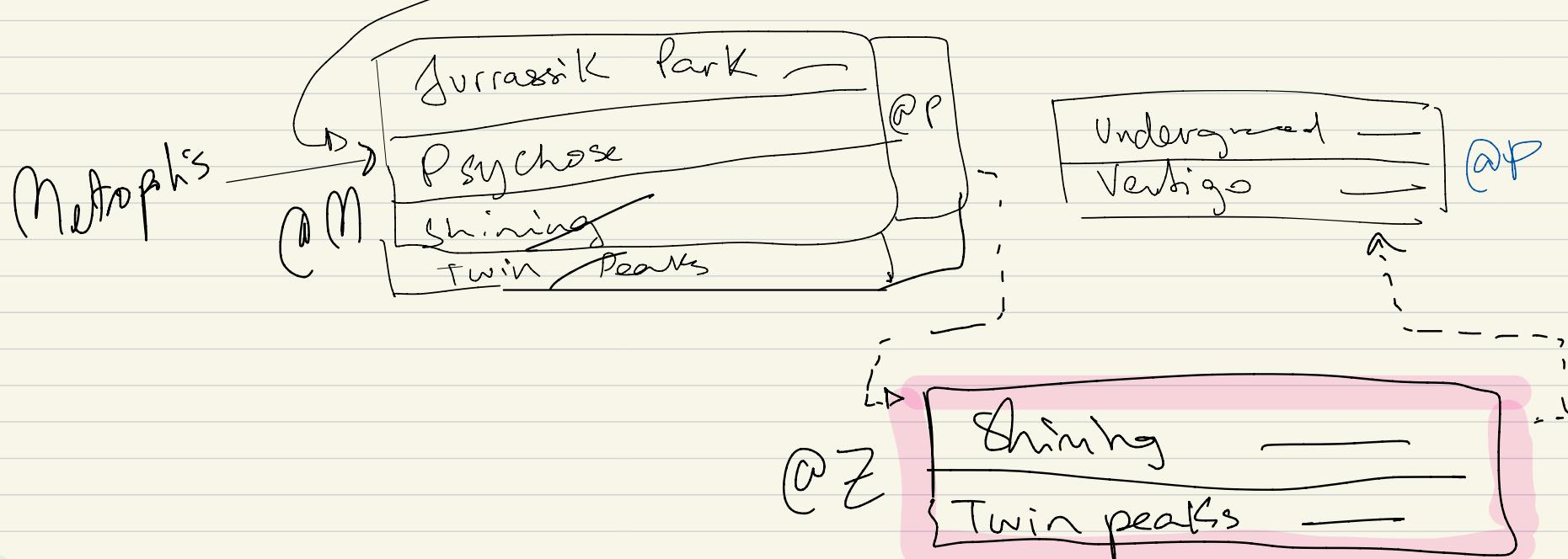
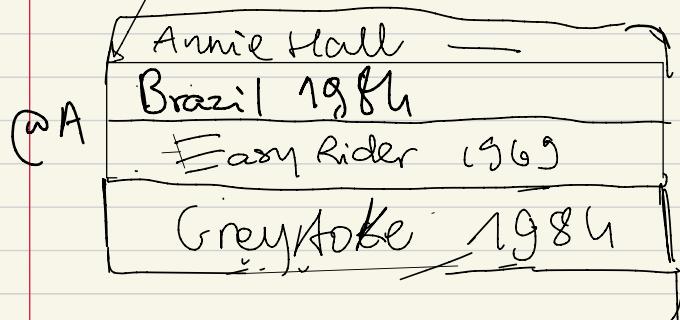
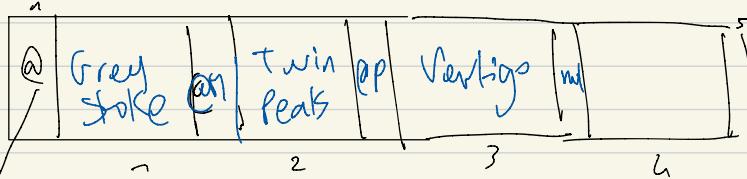
→ Metropolis → M > G

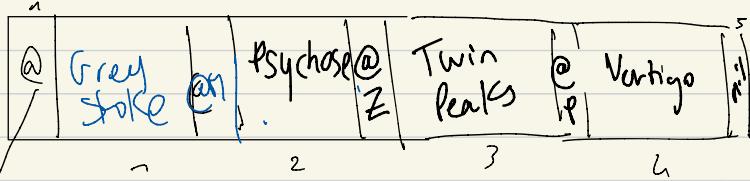
→ Allocation

d'une new

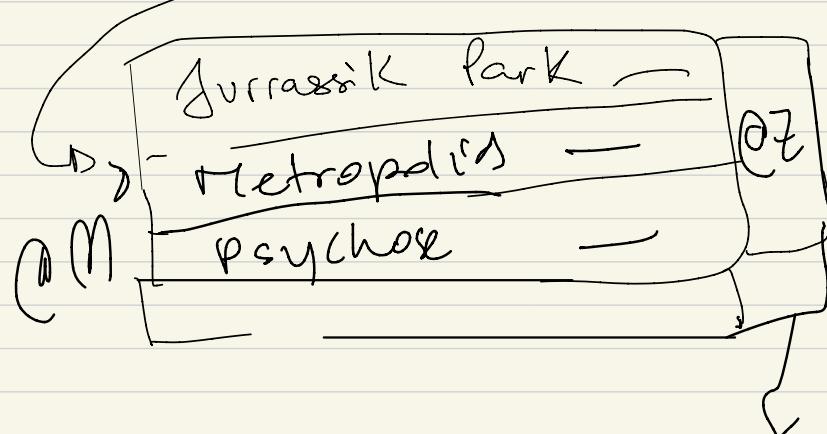
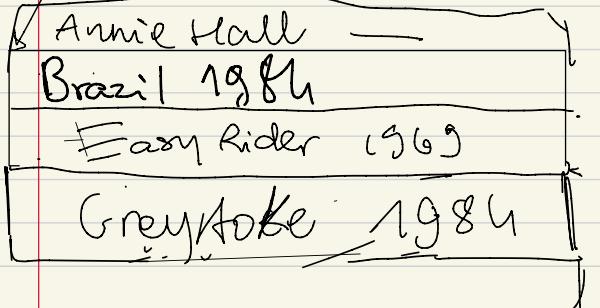
page @ Z,

on va la placer  
à la suite de @M

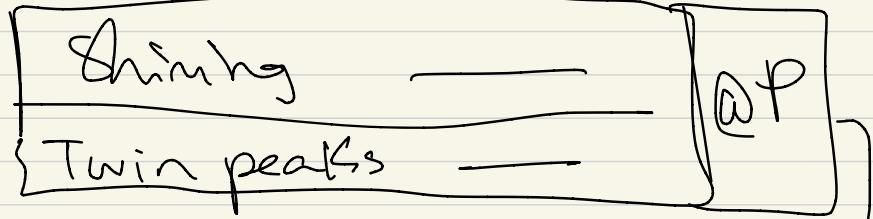




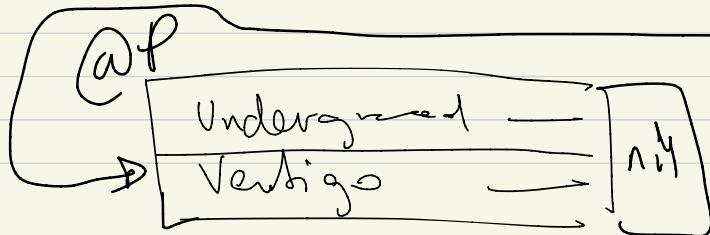
@ A

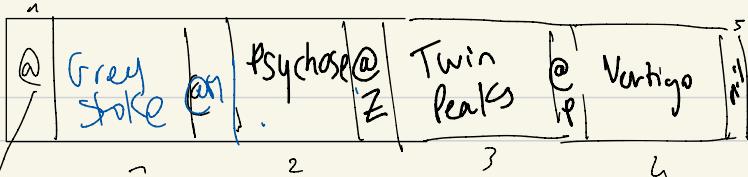


@ Z

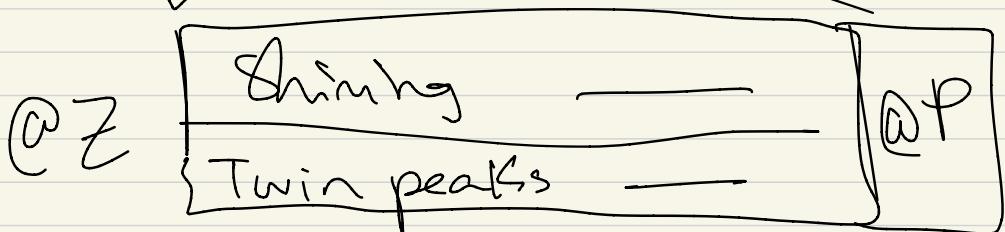
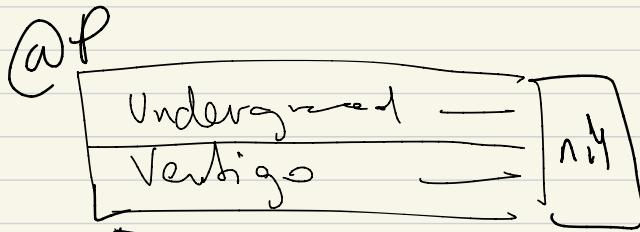
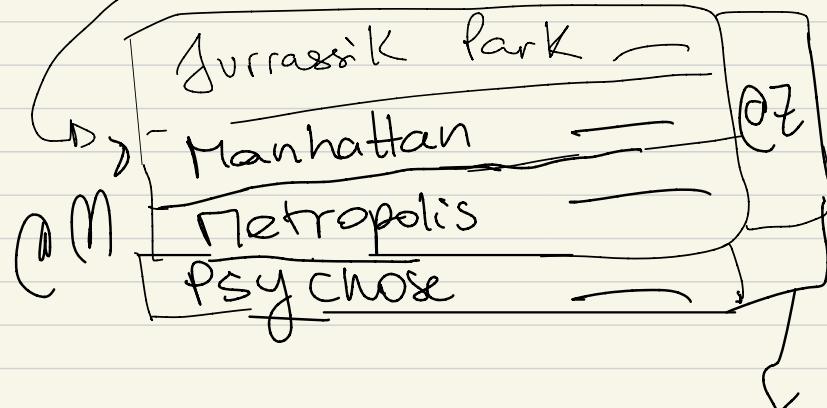
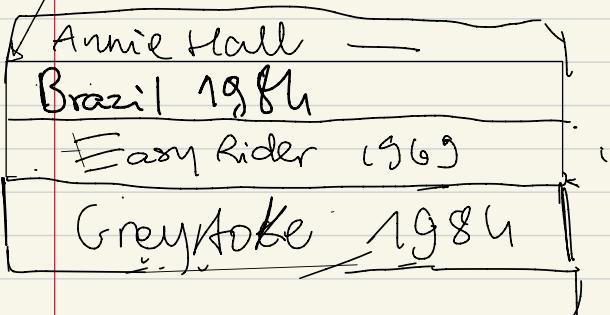


@ P

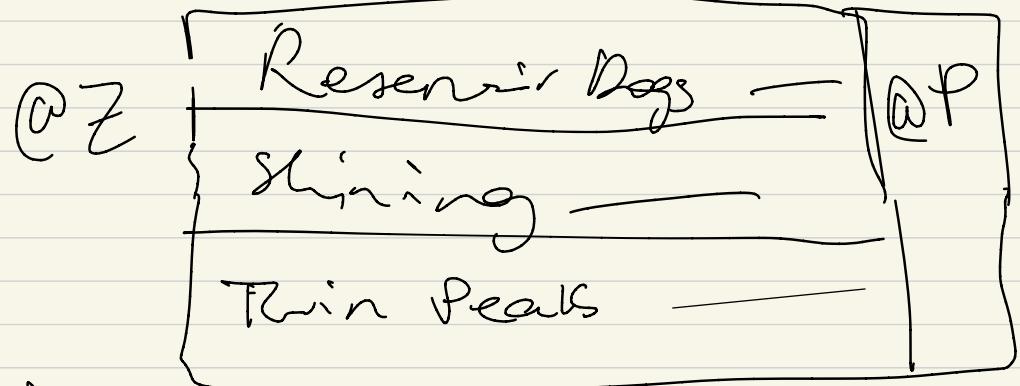
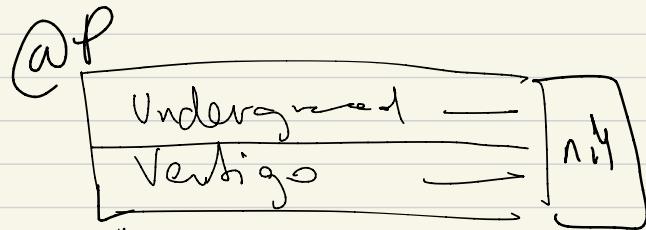
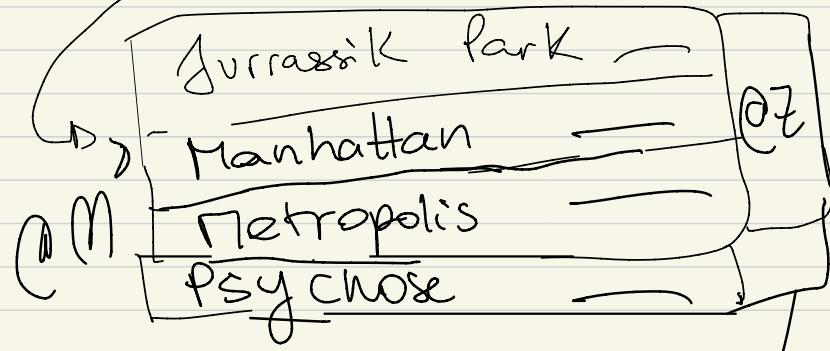
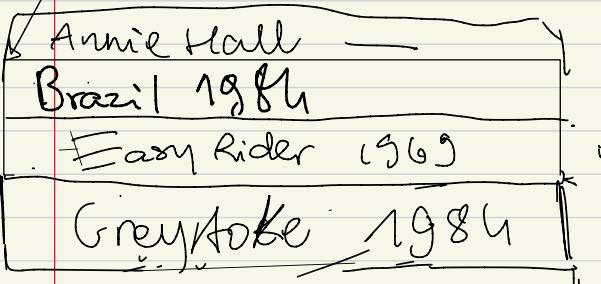
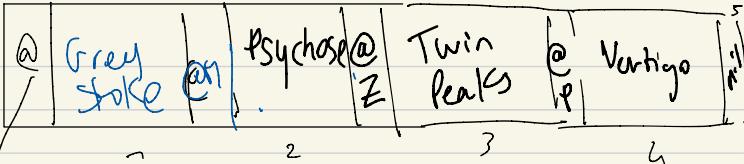




Insertion  
→ Manhattan

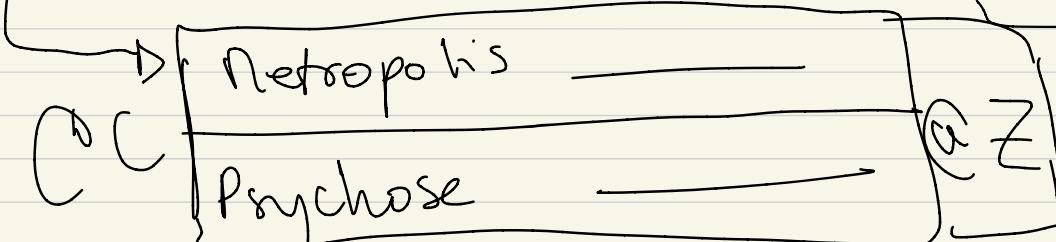
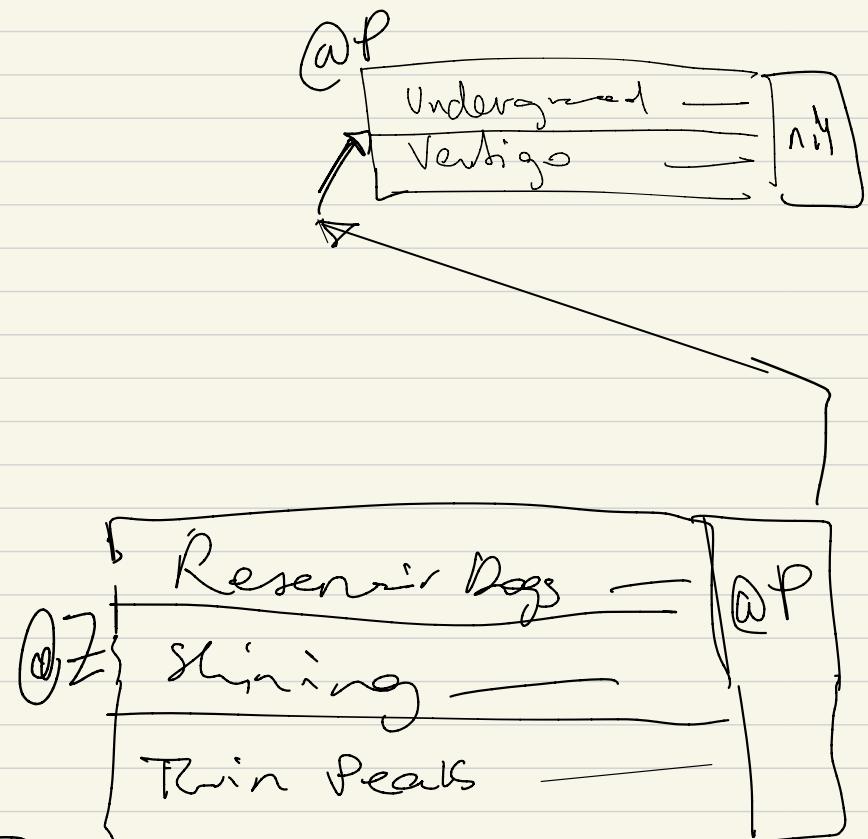
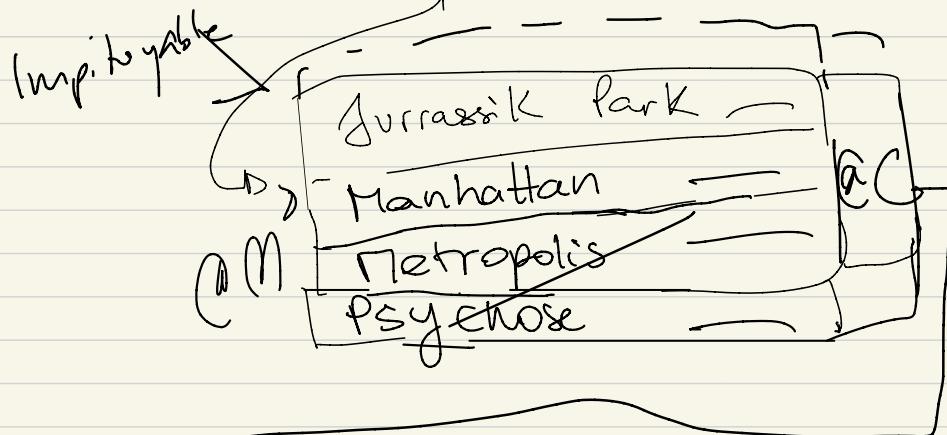
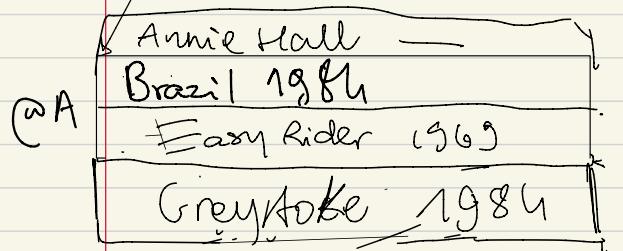
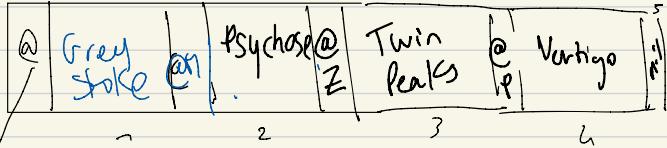


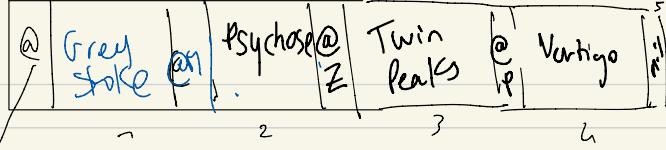
Insert → Reservoir dogs



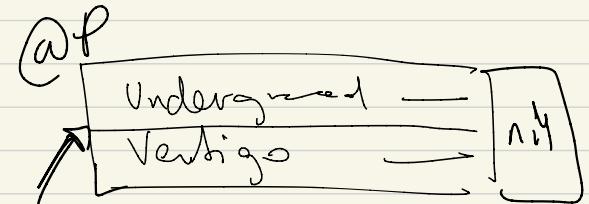
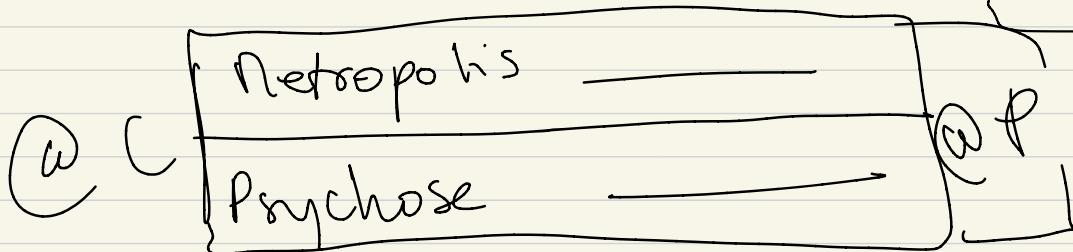
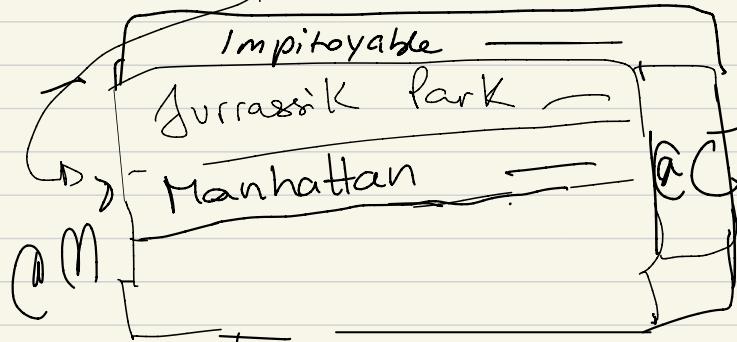
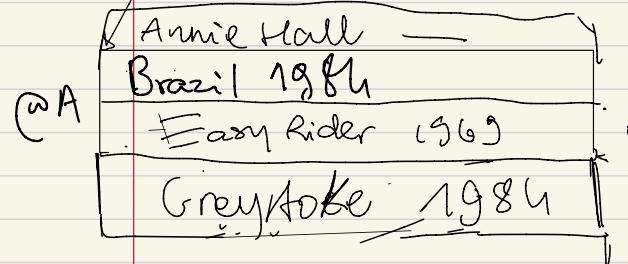
Insert°  
→ Impitoyable (1992)

I var en M → déjà Rempli  
→ Allocat - new page d'@ C

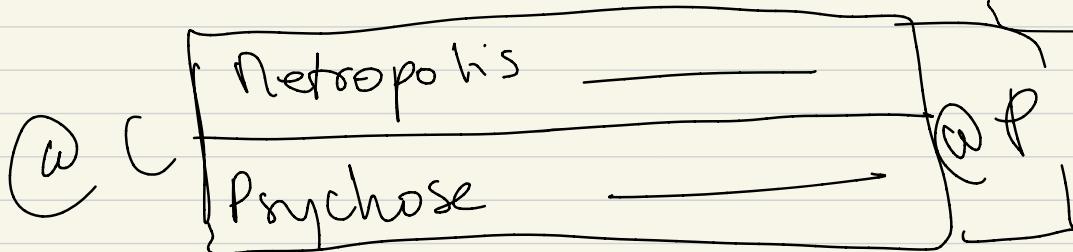
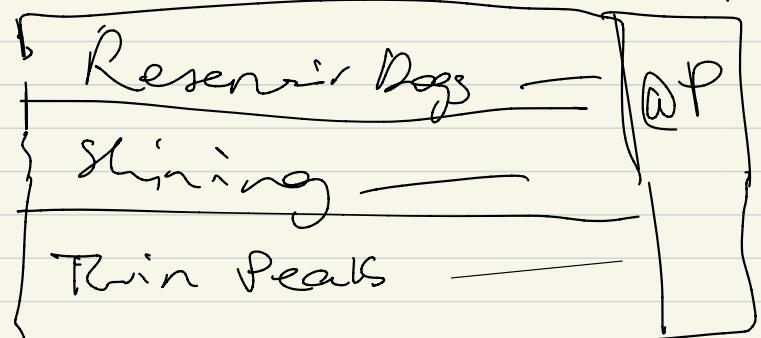




- Manhattan est l'élément médian

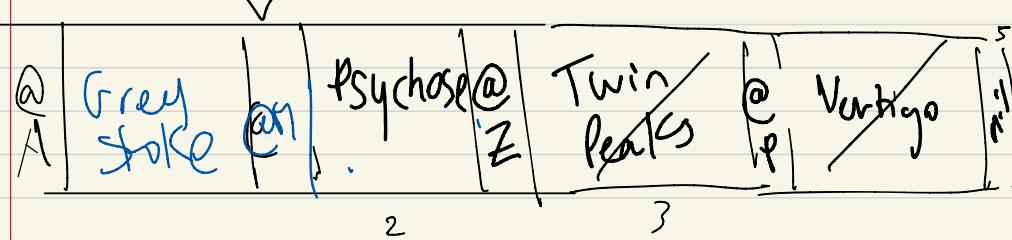


ON  
route la clé  
Manhattan

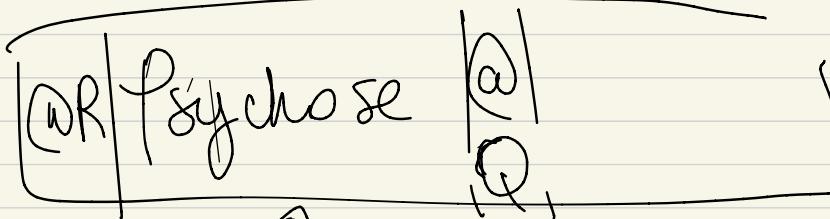


plus de place ds b' cache ?  
→ allocat° new page

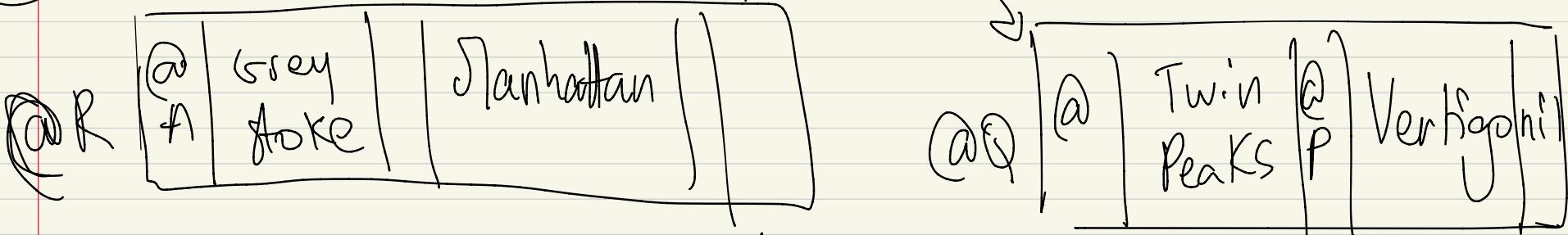
# Manhattan



①



②



Psychose est le m'lien  
donc on monte  
sa clé

→ cette étape  
est complexe  
(même pour la  
prof)  
→ s'appliquer

là on a  
montré  
psychose (sa dé)  
→ on garde  
psychose dans @R  
mais y'a un doute

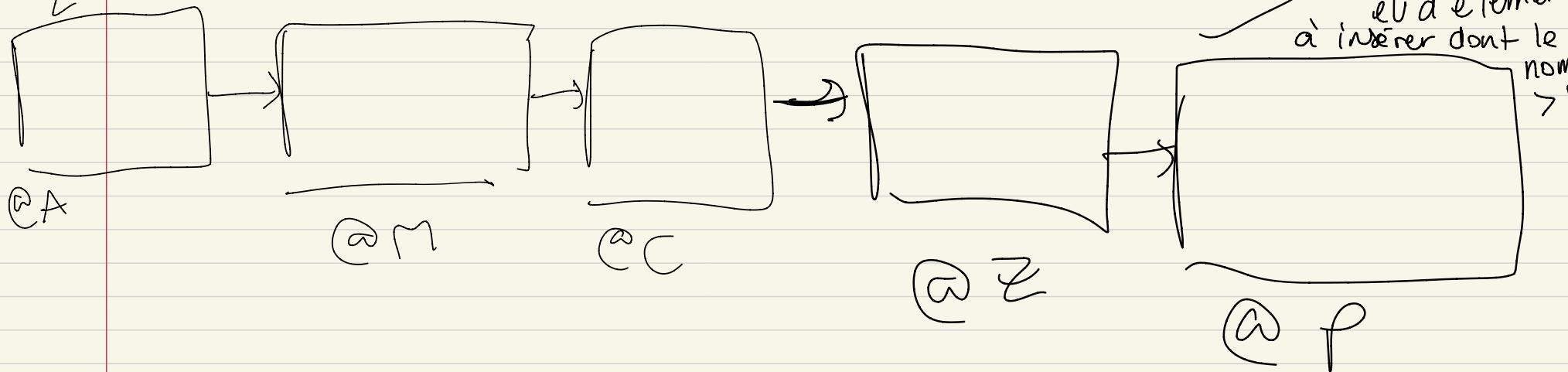
@R Psycho @Q

@R @A grey @M Stoke M @C Manhattan @C Psycho

@Q @Z Twin Peaks @P Verhoffil

ici c'est  
un  
stockage  
dé-adressé

toujours  
car on a  
pas encore  
eu d'élément  
à insérer dont le  
nom > V



+ Arborescence à un niveau de  $\oplus$

17/12 → Examen à 9h30

**Exercice 1 : Organisation séquentielle indexée**

1. Soit la table Film :

Titre	Année
Vertigo	1958
Brazil	1984
Twin peaks	1990
Underground	1995
Easy Rider	1969
Psychose	1960
Greystoke	1984
Shining	1980
Annie Hall	1977
Jurassic park	1992
Metropolis	1926
Manhattan	1979
Reservoir Dogs	1992
Impitoyable	1992
Casablanca	1942
Smoke	1995

- a) Combien de pages sont-elles nécessaires si on stocke les données suivant une organisation d'arbre B d'ordre 2 sur l'attribut Titre ?
- b) Combien de pages sont-elles nécessaires si on stocke les données suivant une organisation d'arbre B+ d'ordre 2 sur l'attribut Année ?

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	

- - $NLevel_{nro\_client}(l) =$
  - $NBlocks_{nro\_client}(l) =$
3. Supposons que pour la table client de l'exercice 2 on dispose d'un index dense organisé en arbre B+ associé à l'attribut succursale. Les pages de données de l'index sont composées de couples (succursale , nro\_client).
- Les clients sont distribués uniformément dans les succursales.
- Décrivez l'index secondaire en indiquant :
- $SC_{succursale}(\text{Client}) =$
  - $NDistinct_{succursale}(\text{Client})$
  - $NLevel_{succursale}(l)$
  - $NBlocks_{succursale}(l)$
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 noeuds internes des pointeurs vers d'autres noeuds. 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

À Savoir pour l'exercice :

- Concepts de base des arbres B<sup>+</sup>

- Indexes :

→ primaire : Les enregistrements sont stockés, ordonnés dès le départ, en suivant l'index

• et dans ce type d'index, la clé de recherche est la clé de la relation

→ secondaire : les enregistrements sont déjà stockés, ainsi, l'index second. est construit à partir des champs (=attributs), qui n'ordonnent pas le fichier

Exemple :

2 méthodes d'accès pour la requête : Select \*

/

From T  
Where age = 'Valeur';

soit effectuer  
une rech.  
sequentielle

utiliser la

clé de rech.  
correspondant à  
l'attribut age,  
index second.

CCL: L'index n'est pas

toujours la solution,

→ car:

- l'index est une table  
(à parcourir)
- à chaque modification

insertion / suppression,  
il faut changer l'index

→ Je comprends pas en quoi c'est mieux  
(B), alors qu'il faut aussi insérer  
dans l'index

↳ C'est coûteux aussi

↓  
Arbre B

Meilleur cas

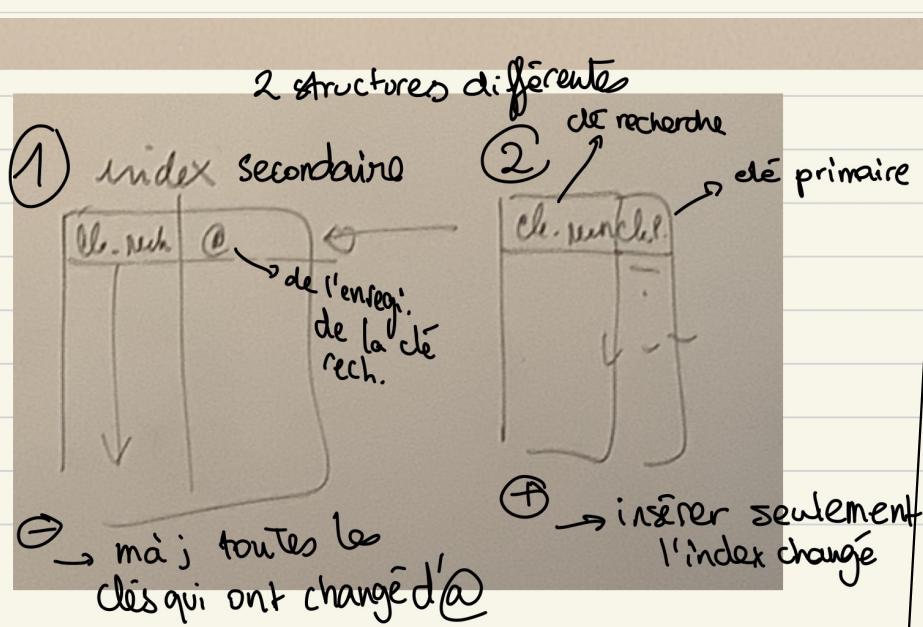
Lecture d'une seule page  
lorsque la clé est dans la racine

Pire cas

Lecture d'au max  
 $\log m(n)$

avec  $m$  l'ordre et  
 $n$  le nb d'éléments

# Comment se structure un index ?



questions  
omniprésentes :

- combien d'accès
- pour trouver l'enregistrement
- cb d'E/S ...

Réponse : hauteur  
de l'arbre + 1

## Expliqat°

→ une table basique  
ou  
→ une table d'index

.) Stockage d'une table de données de la BD

(on a vu → contigüe

→ ordonné  
→ ordonné hachage  
liste chaînée  
Arbre B / B<sup>+</sup>)

→ on utilise n'importe quelle structure  
vue en cours

→ une fois stockée, la table ne bouge plus  
(enfin sa structure)

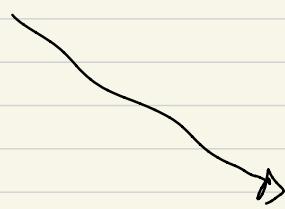
→ Ensuite, on peut créer un index secondaire  
(une fois la table stockée)

. index secondaire sera alors une table ordonnée  
par rapp. à la clé de recherche :

(sous) (clé de recherche, clé relation)

↳ ça c'est l'organisation stocke la table T par rapport à la clé de recherche

Types d'index :



(clé de recherche, @ de l'enregistrement avec attribut = clé de recherche)

↓  
(Rid)

Rq (en général, la table est stockée par rapport à la clé primaire) mais bon

Taille d'une page =  $P(\text{enregistrement}) + 8$

$$400 = 12 + 8 + 8$$

$$400 = P_z(20) + 8$$

$$400 - 8 = P(20)$$

$$\frac{392}{20} \approx P \approx 19,6 \text{ donc on peut stocker}$$

19 enreg / page

## Exercice 2 :

Q1) ~ Taille du tuple :  $7 + 50 + 50 + 20 + 23 = 150$  octets

On additionne chacune des tailles d'attribut, ce qui nous donne 150 octets

- NTuples (client) : la cardinalité est de 300 000, d'après l'énoncé.

$$\begin{aligned} - \beta\text{Factor} (\text{client}) &= \\ &\hookrightarrow \text{le nombre d'enregistrements par page} \\ &= \frac{\text{Taille d'une page}}{\text{Taille du tuple}} = \frac{3000}{150} = 20 \end{aligned}$$

une adresse de page est stockée sur 8 octets

les pages de la base ont 3000 octets pour les données + 8 octets d'adresse.

- NBlocks (client) = le s'agit du nombre total de blocs dont on a besoin pour stocker la table

$$N\text{Blocks(client)} = \frac{\text{Cardinalité}}{\text{BFactor}} = \frac{300\ 000}{20} = 15\ 000$$

-  $N_{\text{level nro-client}}$  (1) : Il s'agit du nombre de niveaux dans l'arbre  $B^+$ , construit pour l'index nro-client

Pour trouver, nous utiliserons la formule de la hauteur d'un arbre :  $\log_m(n)$

→ Dans chaque noeud interne, chaque couple (clé, pointeur) fait 15 octets (clé nro-client = 7 octets et pointeur d'adresse = 8 octets)

$$\text{donc } m = \frac{\text{Taille page}}{\text{couple}} = \frac{3000}{15} = 200$$

→ Nous avons une cardinalité = 300 000 =  $n$

Donc : hauteur =  $\log_{200}(300\ 000) \approx 3$  niveaux

N<sub>Level</sub> n<sub>raccont</sub>(1) = 3 niveaux

- NLBlocks<sub>nro-client</sub>(1) = Il s'agit de la taille totale de blocs nécessaires pour stocker l'index
- Nous allons calculer le nombre de blocs de feuilles :

Comme NTuples(client) = 300 000 =  $n$

On a également trouvé que l'ordre,  $m$ , est de 200.

Ainsi, on a =  $\frac{300\ 000}{200} = 1500$

- Ensuite, il faut calculer les niveaux internes de l'arbre :

On sait que l'arbre possède 3 niveaux, en incluant le niveau des feuilles.

Donc, il nous reste plus qu'à calculer les niveaux internes :

Le premier niveau interne, i.e., celui qui pointe vers ces 1500 pages de feuille, ses pages peuvent chacune pointer vers 200 fils.

Ainsi  $\frac{1500}{200} \approx 8$  pages sont nécessaires afin de gérer les 1500 adresses de pages

(feuilles).

→ Quant au niveau du dessus, la racine, le deuxième niveau interne doit contenir des pointeurs vers l'adresse des 8 pages du premier niveau, que nous venons de calculer. Chaque nœud peut pointer vers 200 fils différents, ici, seuls 8 pointeurs sont requis. Donc une page est suffisante.

Au total, il faut :

$$1500 + 8 + 1 \text{ pages} \cdot \text{NLBlocks}_{\text{nro.client}}(1) = 1509$$

Récapitulatif :

Taille du tuple = 150 octets

NTuples (client) : 300 000

Bfactor (client) : 20

NBlocks (client) = 15000

NLevel nro\_client (1) = 3

NLBlocks nro\_client (1) = 1509