Cours de bases de données, aspects systèmes, http://sys.bdpedia.fr

L'arbre B

Philippe Rigaux

le cnam

Arbre-B

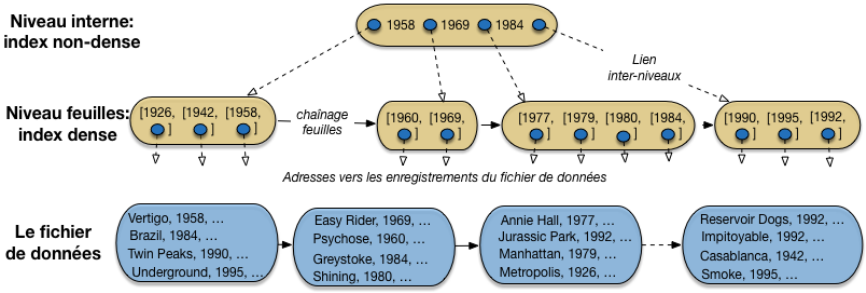
Aboutissement des structures d'index basées sur **l'ordre** des données

- c'est un arbre équilibré
- chaque nœud est un index local
- il se réorganise dynamiquement

Utilisé universellement !

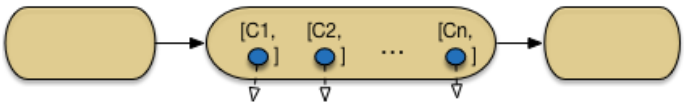
Comparable aux séquentiel indexé, mais évite d'avoir à maintenir des fichiers triés.

Exemple d'un arbre B



Nœud feuille d'un arbre B

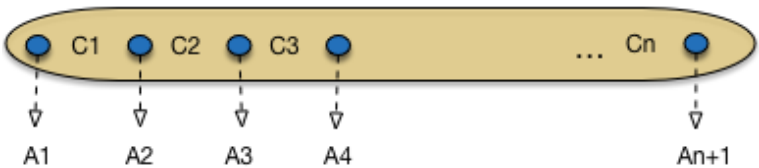
Un nœud feuille est un **index dense local**, contenant des entrées d'index.



Chaque entrée référence un (ou plusieurs) enregistrements du fichier de données : celui (ceux) ayant la même valeur de clé que l'entrée.

Nœud interne d'un arbre B

Un nœud interne est un **index non dense local**, les enregistrements servant de clé, intercalés avec des pointeurs.



Le sous-arbre référencé par A_2 contient tous les enregistrements dont la clé est comprise entre C_1 et C_2.

Avec notre fichier de 1M de films

Admettons qu'une entrée d'index occupe 12 octets, soit 8 octets pour l'adresse, et 4 pour la clé (l'année du film).

Chaque bloc contient 4 096 octets. On place donc $\lfloor \frac{4096}{12} \rfloor = 341$ entrées (au maximum) dans un bloc.

- Il faut $\lceil \frac{1000000}{341} \rceil = 2\,933$ blocs pour le niveau des feuilles.
- Le deuxième niveau est **non dense**. Il comprend autant d'entrées que de blocs à indexer, soit 2933. Il faut donc $\lceil \frac{2\,933}{341} \rceil = 9$ blocs (au mieux).
- Finalement, un troisième niveau, constitué d'un bloc avec 8 entrées suffit pour compléter l'index.

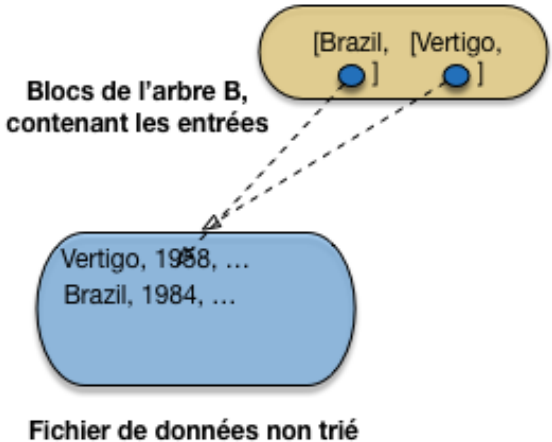
Capacité d'un arbre B

- avec un niveau d'index (la racine seulement) on peut donc référencer 341 films ;
- avec deux niveaux on indexe 341 blocs de 341 films chacun, soit $341^2 = 1\,162\,816$ films ;
- avec trois niveaux on indexe $341^3 = 39\,651\,821$ films ;
- enfin avec quatre niveaux on indexe plus de 1 milliard de films.

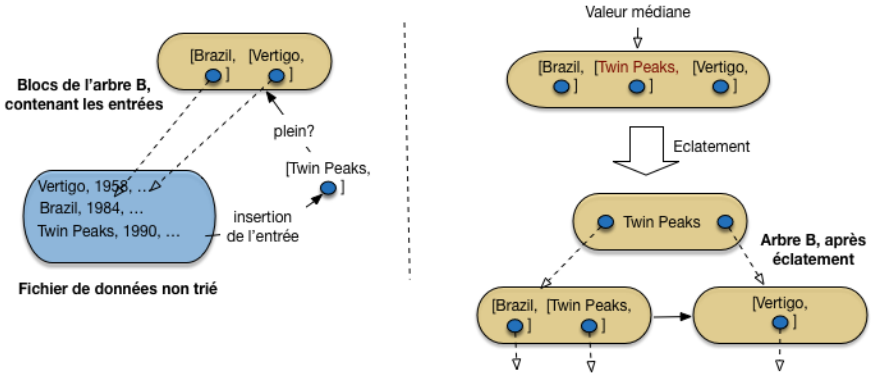
- La hauteur de l'arbre est **logarithmique** par rapport au nombre d'enregistrements.
- Inversement, avec un arbre de hauteur h, on indexe un collection de taille **exponentielle** en h.

Insertion dans un arbre B

On construit l'index sur le titre des films. Situation initiale.

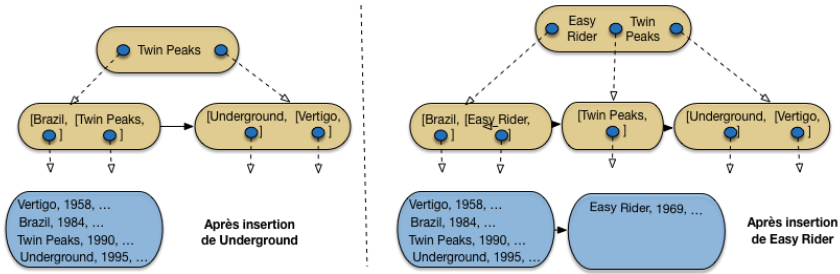


La procédure d'éclatement



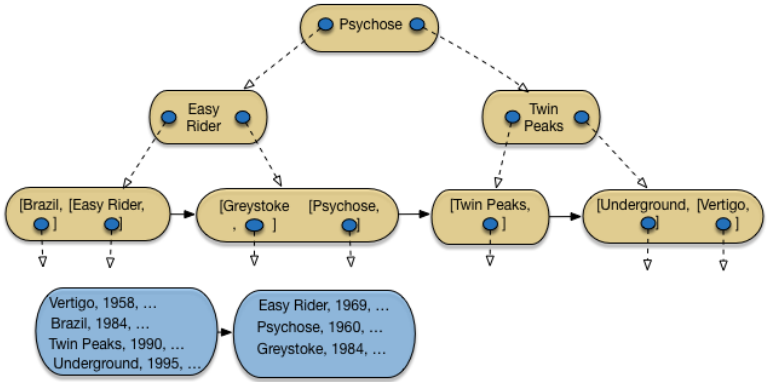
Quand un nœud est plein, il faut effectuer un **éclatement**.

Les insertions continuent



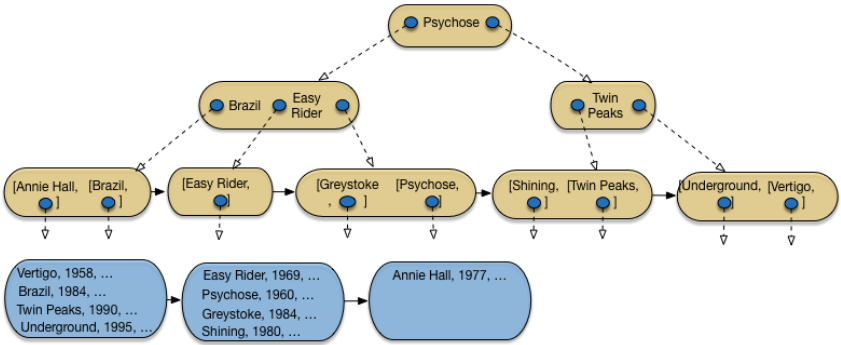
Underground, puis Easy Rider.

Ajout d'un niveau



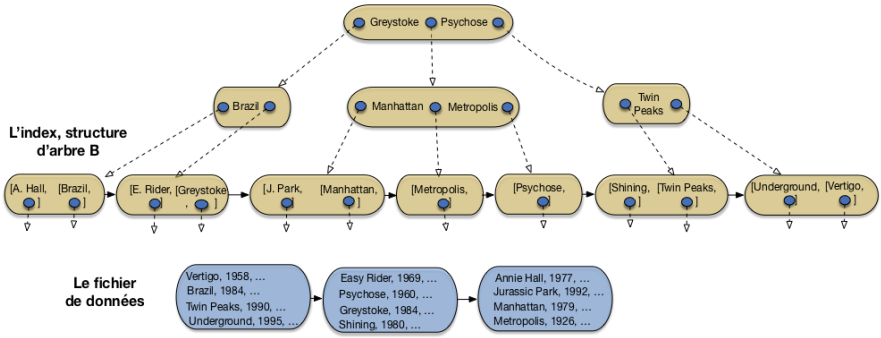
L'éclatement de la racine entraîne l'ajout d'un niveau

Les insertions continuent



Shining et Annie Hall

Les insertions continuent



Après insertion de Jurassic Park, Manhattan et Metropolis

Recherche par clé

```
SELECT *
FROM  Film
WHERE titre = 'Manhattan'
```

- on lit la racine de l'arbre : *Manhattan* étant situé dans l'ordre lexicographique entre *Easy Rider* et *Psychose*, on doit suivre le chaînage situé entre ces deux titres ;
- on lit le bloc intermédiaire, on descend à gauche ;
- dans la feuille, on trouve l'entrée correspondant à Manhattan ;
- il reste à lire l'enregistrement.

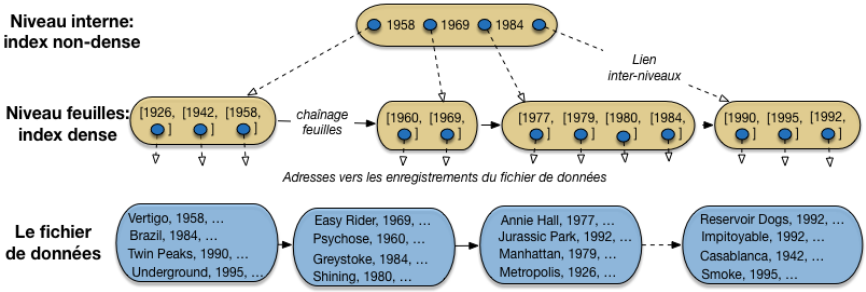
Recherche par intervalle

```
select *
from  Film
where annee between 1960 AND 1975
```

- On fait une recherche par clé pour l'année 60
- on parcourt les feuilles de l'arbre en suivant le chaînage, jusqu'à l'année 1975
- à chaque fois on lit l'enregistrement

Attention, les accès aux fichiers peuvent coûter très cher.

L'arbre sur les années



Recherche par préfixe

Exemple :

```
SELECT *
FROM  Film
WHERE titre LIKE 'M\%'
```

Revient à une recherche par intervalle.

```
SELECT *
FROM  Film
WHERE titre BETWEEN  'MAAAAAA...' AND 'MZZZZZZ...'
```

Contre-exemple :

```
SELECT *
FROM  Film
WHERE titre LIKE '\%e'
```

Ici index inutilisable

Création d'un arbre B

Sur la clé primaire.

```
create table Film (titre varchar(30) not null,
                   ...,
                   primary key (titre)
                   );
```

Sur n'importe quel attribut.

```
create index filmAnnee on Film (annee)
```

Le SGBD synchronise le contenu de la table et celui de l'index

Résumé : efficacité de l'arbre B

Il est (presque) parfait !

- On a très rarement besoin de plus de trois niveaux
- Le coût d'une recherche par clé est le nombre de niveaux, plus 1.
- Supporte les recherches par clé, par intervalle, par préfixe
- Dynamique

On peut juste lui reprocher d'occuper de la place.