

# Opérateur = itérateur

Tout opérateur est implanté sous forme d'un **itérateur**. Trois fonctions :

- `open` : initialise les ressources et positionne le curseur ;
- `next` : ramène l'enregistrement courant se place sur l'enregistrement suivant ;
- `close` : libère les ressources ;

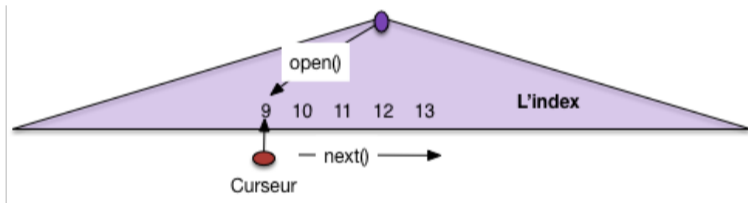
Échanges :

- Un itérateur **consomme** des nuplets d'autres itérateurs **source**.
- Un itérateur **produit** des nuplets pour un autre itérateur (ou pour l'application).

# Exemple : parcours d'index (IndexScan)

Rappel : index = arbre B.

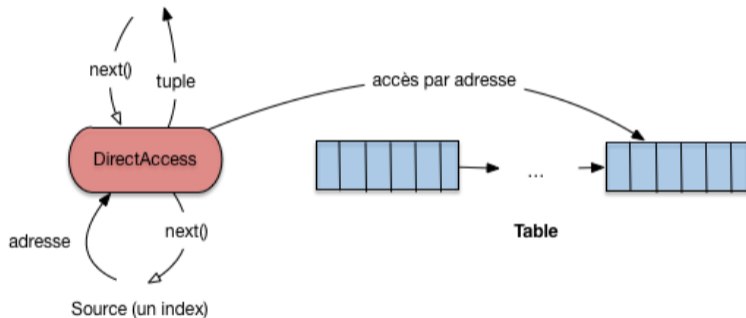
- Pendant le `open()` : parcours de la racine vers la feuille.
- À chaque appel à `next()` : parcours en séquence des feuilles.



**Efficacité** : très efficace, quelques lectures logiques  
(index en mémoire)

# Accès par adresse : DirectAccess

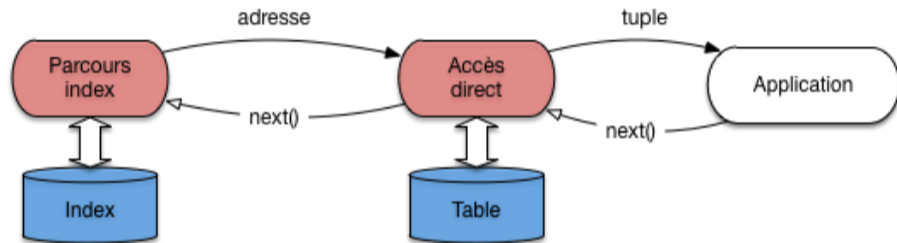
- Pendant le `open()` : rien à faire.
- À chaque appel à `next()` : on reçoit une adresse, on produit un nuplet.



Très efficace : un accès bloc, souvent en mémoire.

# Plan d'exécution

Un plan d'exécution connecte les opérateurs. Ici, recherche avec index.



Le pipelining reste complet.

# Un exemple de base

Nous allons étudier les plans permettant d'exécuter les requêtes **mono-table**.

```
select a1, a2, ..., an  
from T  
where condition
```

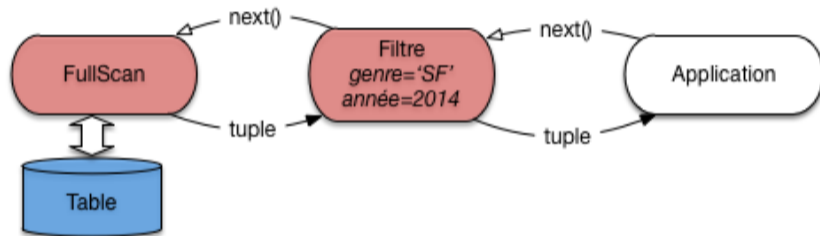
De quels opérateurs a-t-on besoin ?

- [FullScan] : parcours séquentiel de la table (déjà vu).
- [IndexScan] : parcours d'un index (si disponible).
- [DirectAccess] : accès **par adresse** à un nuplet.
- [Filter] : test de la condition.

Nous obtenons **deux** plans d'exécution possibles.

# Premier plan d'exécution : sans index

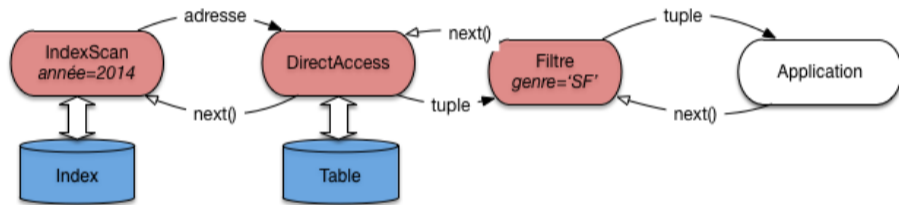
```
| select titre from Film where genre='SF' and annee = 2014
```



N'utilise pas d'index

# Second plan d'exécution : avec index

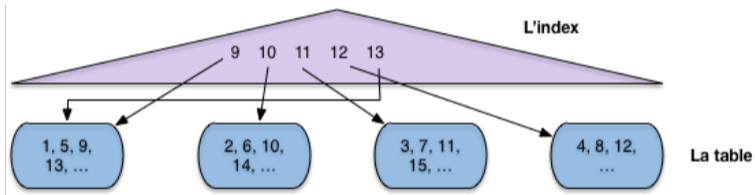
```
| select titre from Film where genre='SF' and annee = 2014
```



Utilise un index sur l'année.

# Index ou pas index ?

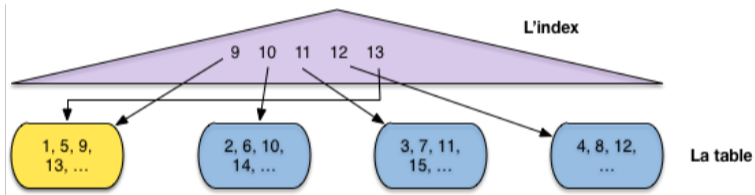
Recherche des nuplets entre 9 et 13, avec 3 blocs en mémoire.





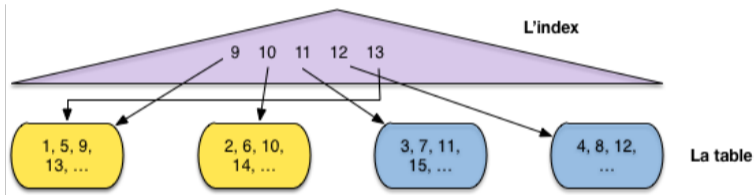
# Index ou pas index ?

Recherche des nuplets entre 9 et 13, avec 3 blocs en mémoire.



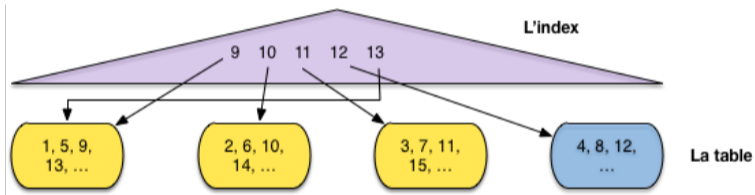
# Index ou pas index ?

Recherche des nuplets entre 9 et 13, avec 3 blocs en mémoire.



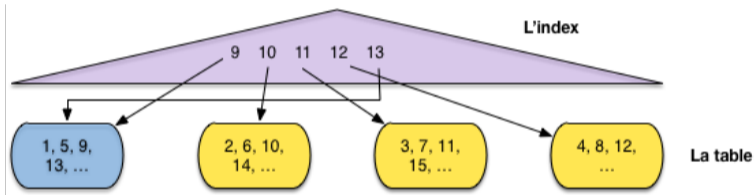
# Index ou pas index ?

Recherche des nuplets entre 9 et 13, avec 3 blocs en mémoire.



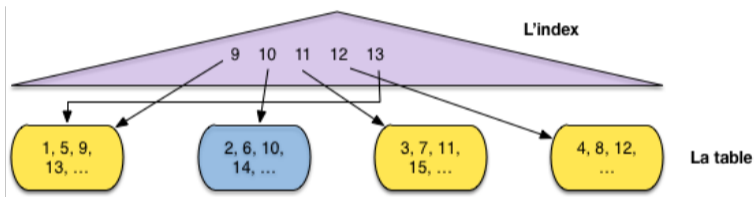
# Index ou pas index ?

Recherche des nuplets entre 9 et 13, avec 3 blocs en mémoire.



# Index ou pas index ?

Recherche des nuplets entre 9 et 13, avec 3 blocs en mémoire.



# Index ou pas index ?

Recherche des nuplets entre 9 et 13, avec 3 blocs en mémoire.

Avec l'index, il faut lire 5 blocs ! Parcours séquentiel bien préférable.

Un cas extrême ! **En pratique** : le SGBD décide en fonction des statistiques et des ressources disponibles.

# Résumé : plans pour requêtes mono-table

Premier aperçu de l'optimisation

- Le système a le choix entre plusieurs plans possibles.
- Distinguer le plus efficace n'est pas toujours trivial.
- Le choix peut changer selon le contexte.

# Résumé : plans pour requêtes mono-table

Premier aperçu de l'optimisation

- Le système a le choix entre plusieurs plans possibles.
- Distinguer le plus efficace n'est pas toujours trivial.
- Le choix peut changer selon le contexte.

**Merci !**