C018SA-W3-S4

Semaine 3 : Exécution et optimisation

1. Introduction
2. Réécriture algébrique
3. Opérateurs
4. **Plans d'exécution**
5. Tri et hachage
6. Algorithmes de jointure
7. Optimisation

Philippe Rigaux

Inria informatique mathématiques

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 ;

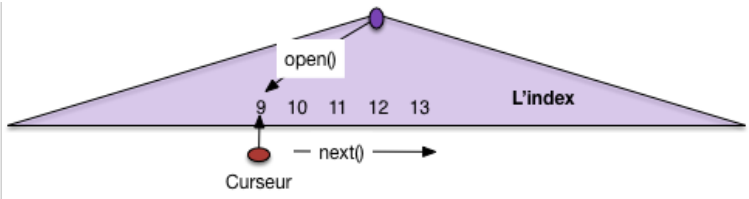
Échanges :

- 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.

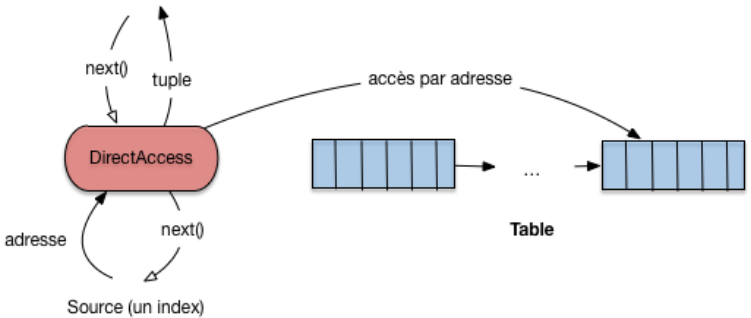
- Pendant le `open()` : parcours de la racine vers la feuille.
- À chaque appel à `next()` : parcours en séquence des feuilles.



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

Accès par adresse : `DirectAccess`

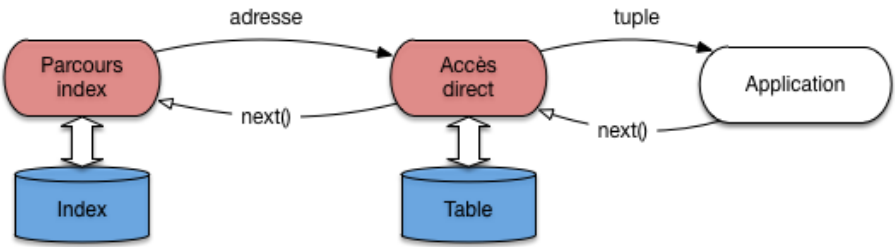
- Pendant le `open()` : rien à faire.
- À chaque appel à `next()` : on reçoit une adresse, on produit un nuplet.



Très efficace : un accès bloc, souvent en mémoire.

Plan d'exécution

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



Le pipelinage reste complet.