C018SA-W3-S6

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

Informatique mathématiques
Inria

L'opérateur qui nous manque

La jointure : opération très courante, potentiellement coûteuse.

L'opérateur de jointure complète notre petit catalogue pour (presque) toutes les requêtes SQL.

```
select a1, a2, .., an
from T1, T2, ..., Tm
where T1.x = T2.y and ...
order by ...
```

Plusieurs algorithmes possibles.

Principaux algorithmes

On ve se limiter à quelques exemples représentatifs :

Jointure avec index

- Algorithme de jointure par boucles imbriquées indexées.

Jointure sans index

- Le plus simple : **boucles imbriquées (non indexée)**.
- Plus sophistiqué : la **jointure par hachage**.

Jointure avec index

Très **courant** ; on effectue **naturellement** la jointure sur les clés primaires/étrangères.

- Les films et leur metteur en scène

```
select * from Film as f, Artiste as a
where f.id_realisateur = a.id
```

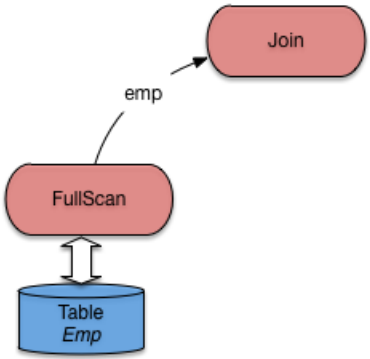
- Les employés et leur département

```
select * from emp e, dept d
where e.dnum = d.num
```

Garantit **au moins** un index sur la condition de jointure.

Jointure avec index : l'algorithme

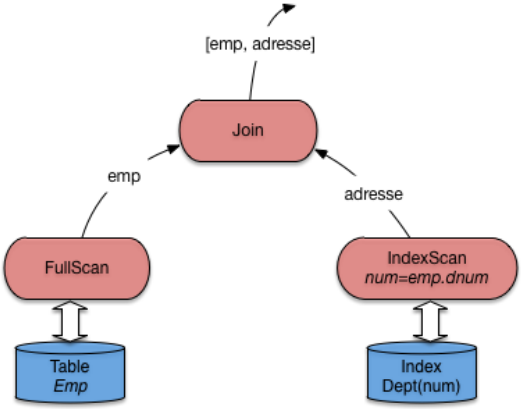
On parcourt séquentiellement la table contenant la clé étrangère.



On obtient des nuplets `employé`, avec leur no de département.

Jointure avec index : l'algorithme

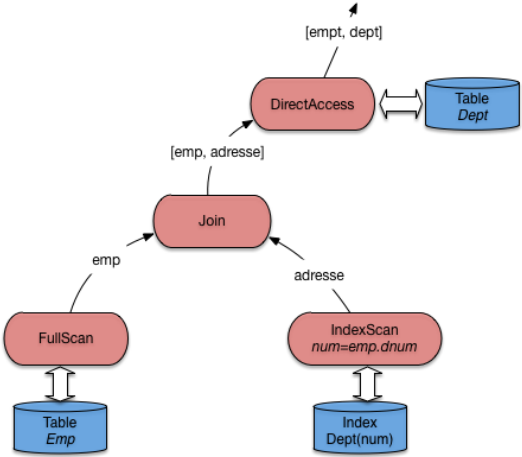
On utilise le no de département pour un accès à l'index `Dept`.



On obtient des paires `[employé, adrDept]`.

Jointure avec index : l'algorithme

Il reste à résoudre l'adresse du département avec un accès direct.



On obtient des paires `[employé, dept]`.

Jointure avec index : l'algorithme

Avantages :

- Efficace (un parcours, plus des recherches par adresse)
- Favorise le temps de réponse **et** le temps d'exécution