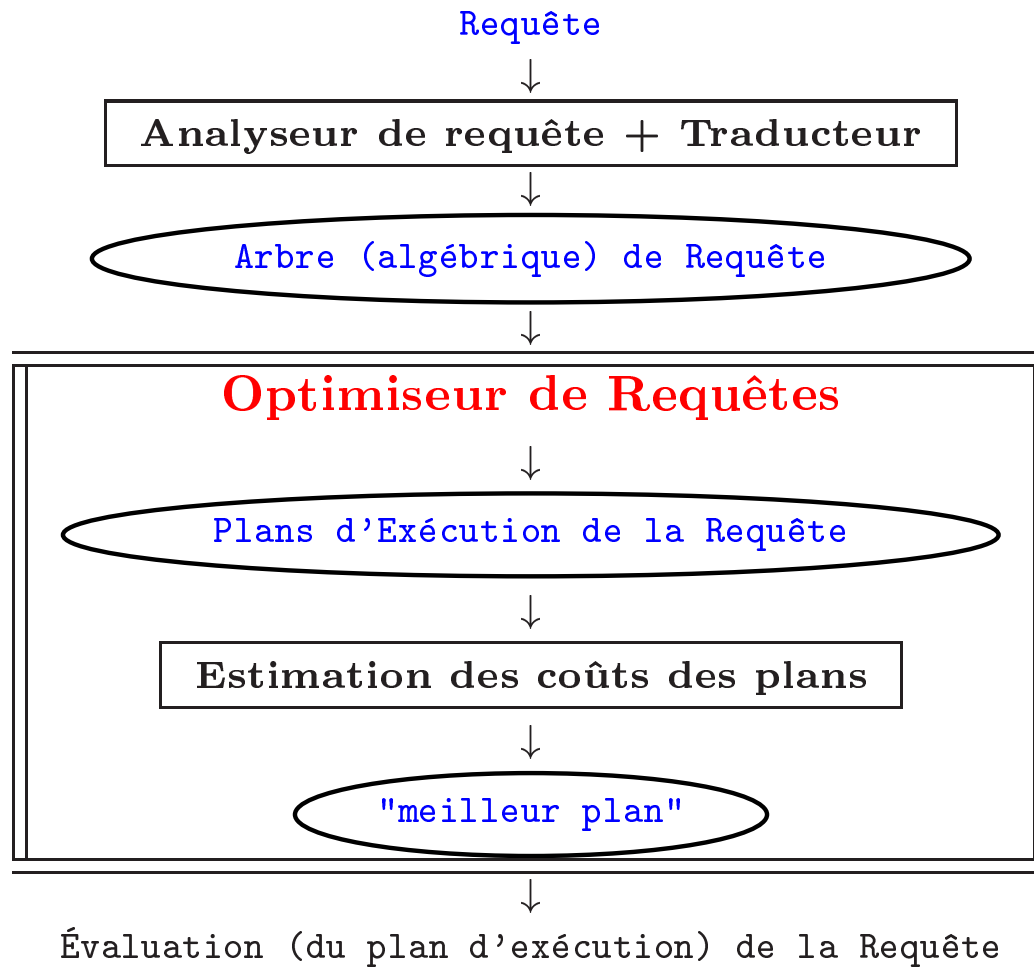


IMPLÉMENTATION DE L'ALGÈBRE OPTIMISATION DES REQUÊTES SQL

Nicole Bidoit Université Paris XI, Centre d'Orsay bidoit@lri.fr

Traitement des requêtes



Traitement des requêtes

comprendre, analyser, optimiser une requête SQL

- Traduction SQL → opérateurs algébriques
fin du chapitre
- Chemin d'accès (index)
fait
- Algorithmes (méthodes) implémentant les opérateurs algébriques
utilisation des index par certaines méthodes
ce chapitre
- Fonctionnement de l'optimiseur
choix des méthodes pour les opérateurs
utilisation d'information statistiques
ce chapitre

Opérateurs et Implémentation

- sélection : (parcours / index)
 - projection : (parcours - filtrage)
 - jointure : (nombreux algorithmes)
 - itération
 - partitionnement
 - utilisation des index
-
- composition des opérateurs

Exemple

- Coût = $|E/S|$ (sans compter l'écriture du résultat)
- Éléments pour évaluer les coûts :

relation	n-uplet	page	fichier
R	m oct.	p_r n-uplets	M pages
S	n oct.	p_s n-uplets	N pages

Schémas : R(ABC) S(ADE)

Valeurs numériques : $p_r = 100$; $M = 1000$; $p_s = 80$; $N = 500$

Sélectivité

- Sélectivité = pourcentage d'enregistrements pertinents respectivement au critère de recherche

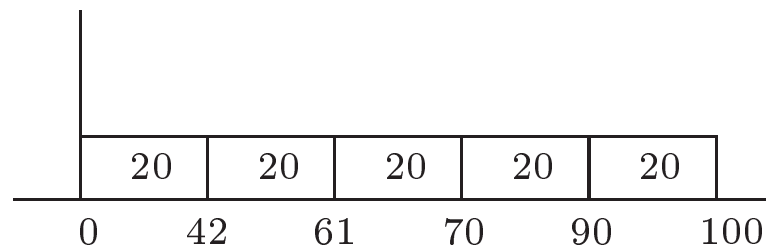
Select A from R where B='b'

Sélectivité de B = 80 % → parcours

Select A from R where B='b'

Sélectivité de B = 10 % → index (si existe)

- Histogrammes (en hauteur ou en largeur)



Histogramme en hauteur

Histogramme en largeur

Jointure par Double itération “brute”

pour chaque n-uplet r de R faire

 pour chaque n-uplet s de S faire

 Si $r[X]=s[X]$ Alors $res := + (r \bowtie s)$

• Coût : $M + p_r \cdot M \cdot N$

• Estimation chiffrée : $1000 + 100 \times 1000 \times 500$ E/S

Jointure par Double itération page à page

pour chaque page p_R de R faire
 pour chaque page p_S de S faire
 $res := + (p_R \bowtie p_S)$

• Coût : $M + M \cdot N$

• Estimation chiffrée : $1000 + 1000 \times 500 \text{ E/S} = 501\,000 \text{ E/S}$

• **Optimisation** : choix de la relation “externe”
 la + petite en nbre de pages!

• Coût : $N + N \cdot M$

• Estimation chiffrée : $500 + 500 \times 1000 \text{ E/S} = 500\,500 \text{ E/S}$

Jointure optimale

Hypothèse : le buffer peut contenir une des deux relations

pour chaque page p_R de R faire charger p_R ;

pour chaque page p_S de S faire

$$\text{res} := + (R \bowtie p_S)$$

• Coût : $M + N$

• Estimation chiffrée : $1000 + 500 E/S = 1\ 500E/S$

Hypothèse : le buffer peut contenir $B+2$ pages

Organisation : B cellules pour charger R + index de hachage ;
1 cellule pour S ; 1 cellule pour écrire le résultat.

pour chaque bloc B_R de B pages de R faire

pour chaque page p_S de S faire

$$\text{res} := + (B_R \bowtie p_S)$$

• Coût : $M + N (M / B)$

• Estimation chiffrée : $1000 + 500 (1000 / 50) E/S = 11\,000E/S$

• Variantes : par bloc pour R et pour S

Jointure par index

Hypothèse : Jointure "sur" X et Index sur S de clé de recherche X

Attention : la relation indexée est "interne" !

pour chaque n-uplet r de R faire

rechercher dans S par index les n-uplets s tels que $r[X]=s[X]$ et

$res := + (r \bowtie s)$

- Coût :

quelle structure d'index ? groupant / non groupant ?

clé de l'index = clé de S ?

combien de s de S sont "joignables" avec un r de R ?

- Optimisation :

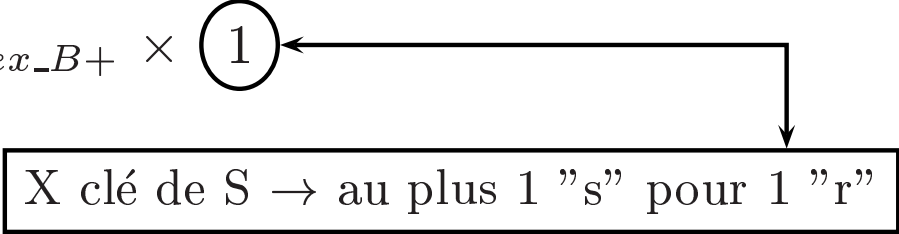
la plus petite relation "externe"

- Utiliser aussi 1 index sur R de clé de recherche X ?

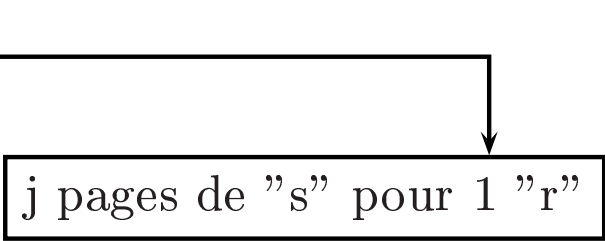
""ne sert à rien"" (accès non pertinents)

Jointure "sur" X par index de clé X sur S

Cas 1: index de type 2, groupant, dense, arbre B+, X clé de S

- coût : $M + M \times p_r \times h_{index_B+} \times 1$
- 
- X clé de S → au plus 1 "s" pour 1 "r"
-

Cas 2: index de type 2, groupant, dense, arbre B+

- coût : $M + M \times p_r \times h_{index_B+} \times j$
- 
- j pages de "s" pour 1 "r"
-

Cas 3: index de type 2, non groupant, dense, arbre B+

- coût : $M + M \times p_r \times h_{index_B+} \times m_{r_s}$.

Jointure par tri-fusion

trier R sur X;

trier S sur X;

fusionner R-triée et S-triée sur X : (rappel fusion)

Attention : R-triée est parcourue 1 fois

plusieurs parcours de parties de S-triée peuvent être nécessaires

- Coût :

le tri de R

$M \log M$

le tri de S

$N \log N$

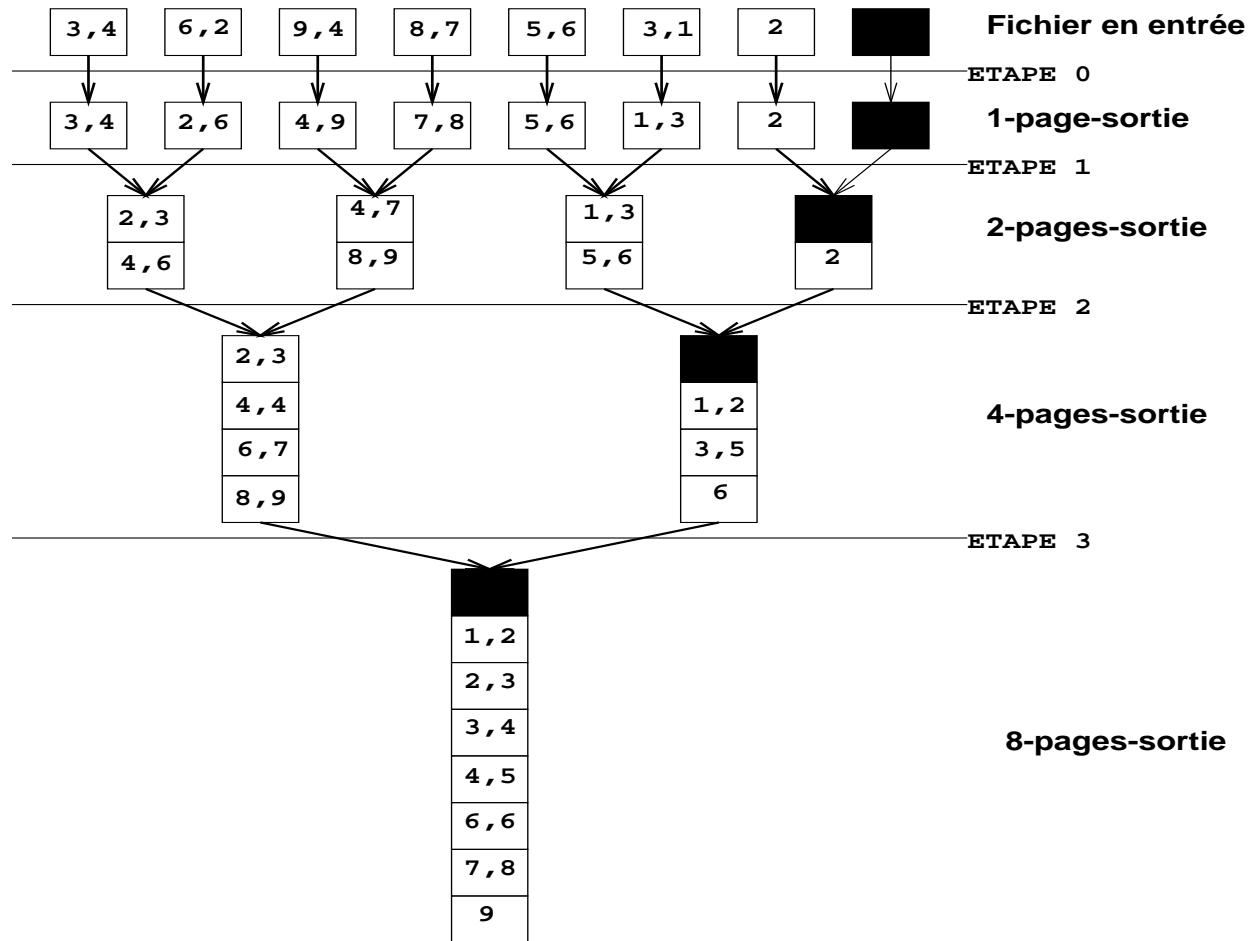
la fusion de R et S

$M + N$

Tri (une petite parenthèse)

- Le tri est une opération centrale dans un SGBD
 - ORDER BY
 - création d'index (arbre B+)
 - élimination de doublés
 - jointure
- Méthodes de tri externes
 - tri rapide adaptable
 - tri fusion

Tri-(bi)Fusion de S



Tri-(bi)Fusion de S

Etape 0 : tri interne de chaque page du fichier (1 à 1)

⇒ production de N pages triées

Etape 1 : fusion par 2 pages des N pages triées du fichier

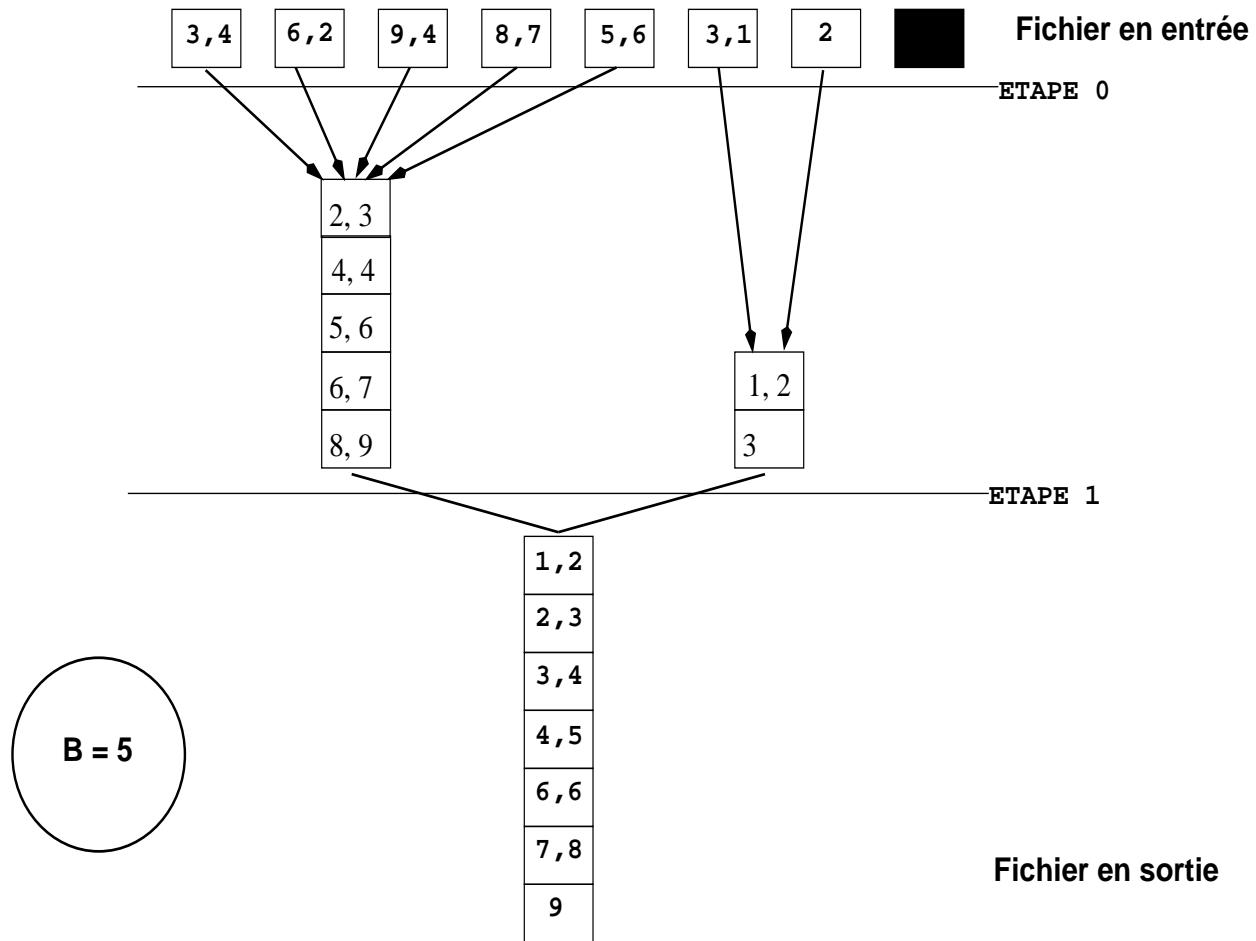
⇒ production de $N/2$ fragments triés de 2 pages

Etape i : fusion par 2 blocs des $N/(2^{i-1})$ blocs triés de 2^{i-1} pages

⇒ production de $N/2^i$ fragments triés de 2^i pages

• Coût : $2 N (\log_2 N + 1)$

Tri-(multi)Fusion



Tri-(multi)Fusion de S

capacité buffer = $B+1$ pages

Etape initiale : tri interne par B pages du fichier

⇒ production de N/B fragments triés de B pages

Etapas suivantes : fusion par B blocs des k blocs triés de p pages

⇒ production de k/B fragments triés de $p.B$ pages

-
- Coût d'une étape : $2 N$
 - Nombre d'étapes : $1 + \log_B (N/B)$

Tri-(multi)Fusion de S

- capacité du buffer $B = 5$
 - taille fichier $N=100$
-

Etape 0 : production de $500/5=100$ fragments triés de $B=5$ pages
par tri de pages 5

Etape 1 : production de $100/5=20$ fragments triés de $B.5=25$ pages
par fusion de 5 blocs de 5 pages

Etape 2 : production de $20/5=4$ blocs trié de $B.25=125$ pages
par fusion de 5 fragments de 25 pages

Etape 2 : production de $4/5=1$ bloc trié de 500 pages
par fusion de 4 fragments de 125 pages

Jointure par tri-fusion (Retour)

- tri-fusionner R, tri-fusionner S puis fusionner pour joindre
-

- tri-fusionner R et S ... mais ...

exploiter le tri-fusion pour joindre par fusion R et S

arrêt du tri de R et S dès que

$$| \text{fragments de R} | + | \text{fragments de S} | \leq B$$

- gain : R et S ne sont pas triées complètement !

Jointure par hachage

Etape 0 : Hachage de R sur X ; Hachage de S sur X (avec h)

Etape de jointure :

Pour chaque “paquet” p_R de R

Hachage de p_R (avec g plus fine que h)

Parcourir le paquet p_S de S (correspondant à p_R)

en utilisant g pour effectuer la jointure des 2 paquets !

nombre de partitions de R (par h) $< B-1$; idem pour S ;

taille d'une partition de R (par h) $< B-2$;

- Coût : $2(M+N) + M + N$

Jointure et Systèmes commerciaux

- Sybase ASE : par index ; fusion
- Sybase ASIQ : itération par page ; par index ; hachage simple ; tri-fusion ; index de jointure
- Oracle 8 : itération par page ; tri-fusion ; hachage hybride
- IBM DB2 : par bloc ; tri-fusion ; hachage hybride
- Microsoft SQL Server : par bloc ; par index ; tri-fusion ; hachage simple ; hash teams (!!)
- Informix : par bloc ; par index ; hachage hybride

Opérations d'aggrégat

SELECT AVG(S.A) FROM S

- parcours + mise à jour variables courantes

SUM → total des valeurs accédées

AVG → (total, nbre) des valeurs accédées

SELECT AVG(S.A) FROM S GROUP BY B

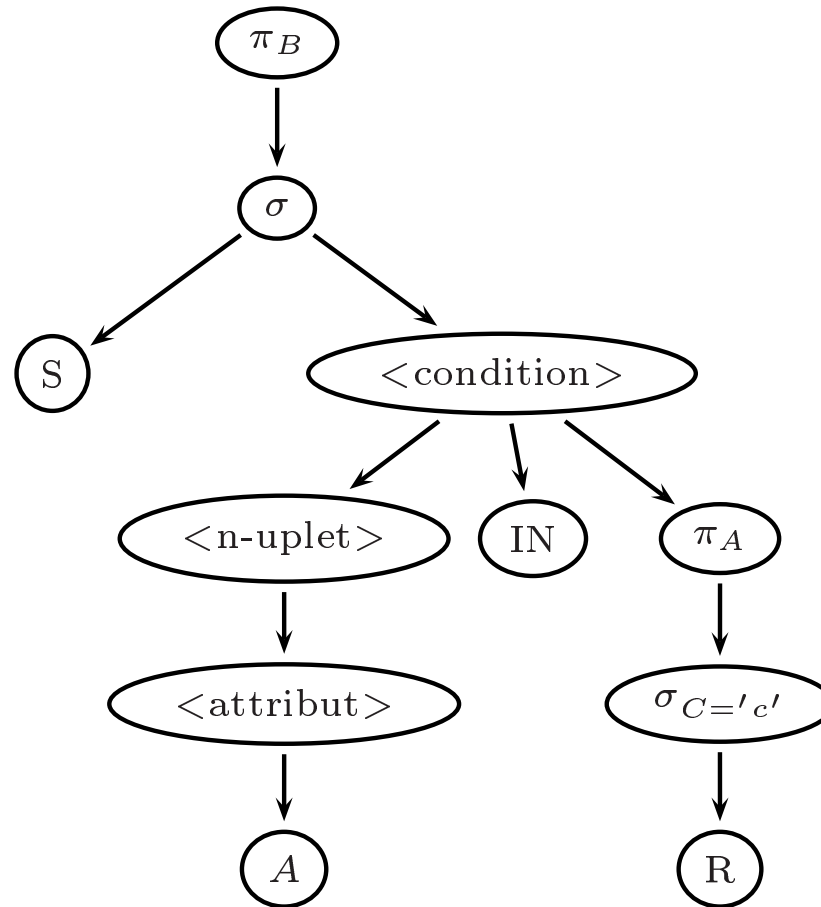
- trier sur les attributs de groupement
- création d'une table de hachage en mémoire centrale
entrée = (valeur de l'attribut de groupement, variable courante)

Optimisation des requêtes

- Analyseur (parseur) = syntaxe SQL + préprocesseur
 - production d'un arbre d'expression SQL
 - vérification des noms de table
 - vérification de l'usage des attributs
 - vérification des types
 - accès au dictionnaire / catalogue des données
- **problème** : les sous-requêtes SQL
 - isolation de "bloc" Select ... From ... Where
- Production d'un arbre de requête
 - arbre utilisant les opérateurs algébriques
 - traduction SQL → algèbre

Traitement/Élimination des sous-requêtes

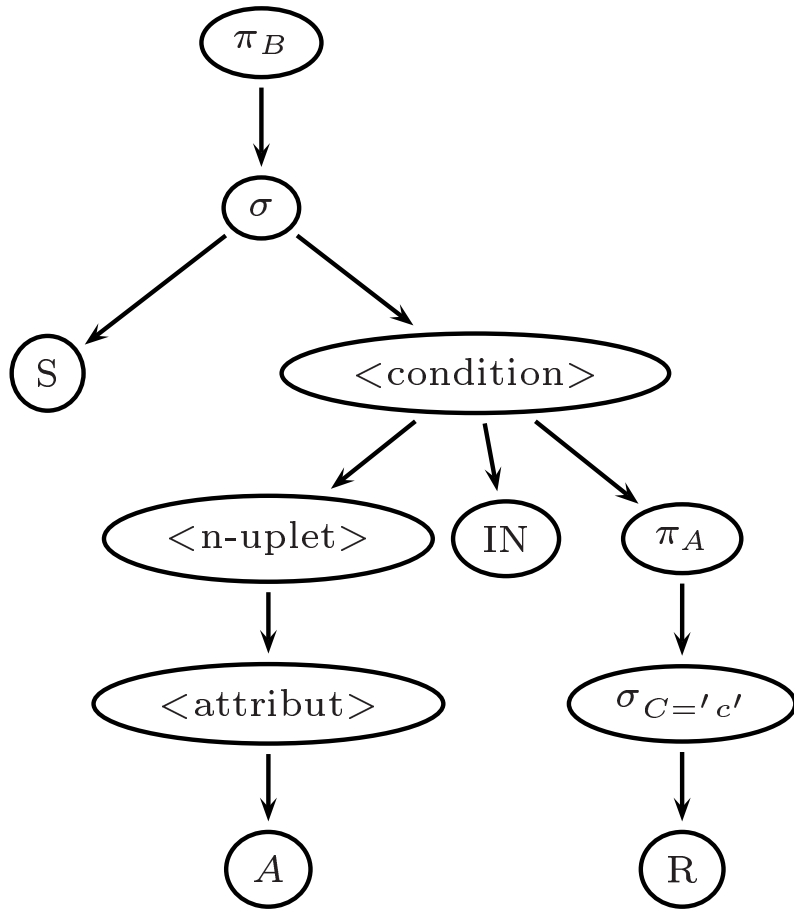
Select B From S
Where A IN
Select A from R
Where C='c'



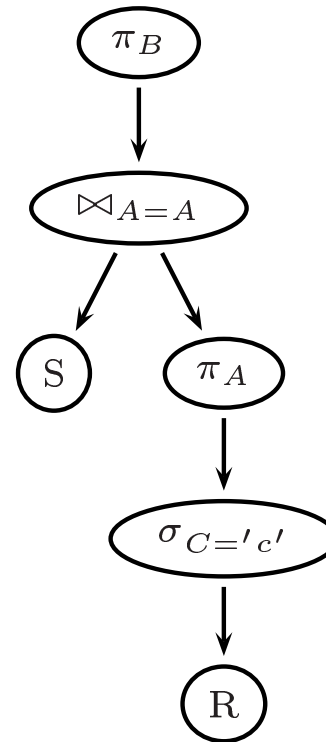
Requête SQL

Arbre de requête

Traitement/Élimination des sous-requêtes



Arbre de requête



Arbre de requête transformé

Plan d'exécution

- **Qu'est-ce qu'un plan d'exécution ?**

Programme combinant des méthodes d'évaluation des opérateurs

Indication de l'enchaînement des méthodes

- **Représentation d'un plan d'exécution = Arbre ordonné**

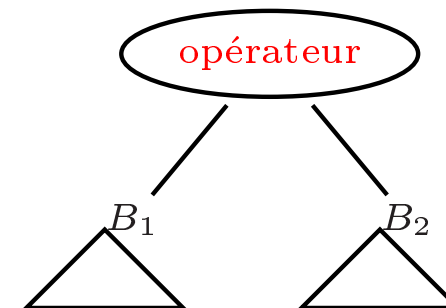
-

noeud interne

méthode pour un opérateur

entrée : résultats de B_1 et B_2

méthode



-

feuille = table

(R)

- **Exécution d'un plan \approx**

Parcours en profondeur à gauche de l'arbre

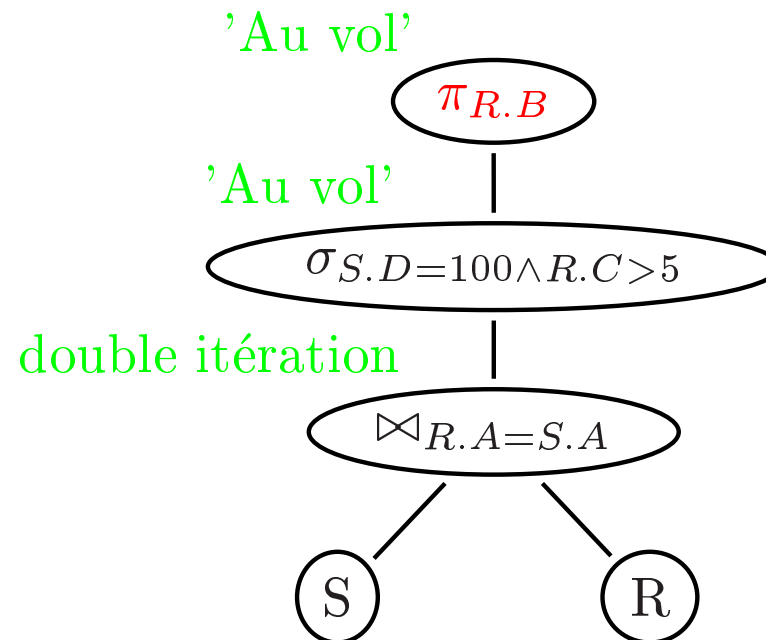
Exemple de Plan d'exécution

Select R.B From R, S
Where R.A=S.A And
S.D=100 And R.C>5

Coût = $500 + 500 \times 1000$

pas le meilleur plan !

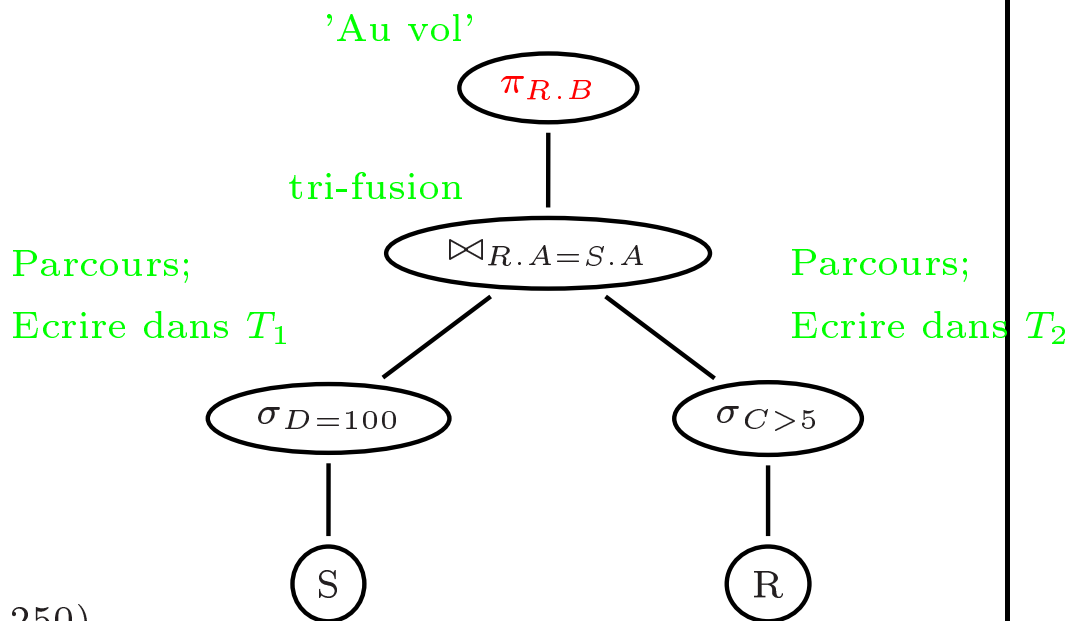
pas le plus mauvais !



Autre plan d'exécution

Select R.B From R, S
 Where R.A=S.A And
 S.D=100 And R.C>5

Coût : (5 pages buffer)
 (500+250)+ (1000+10) +
sélection sélection
 (2×3×250) + (2×2×10) + (10+250)
tri tri fusion
 ≈ 3569 E/S

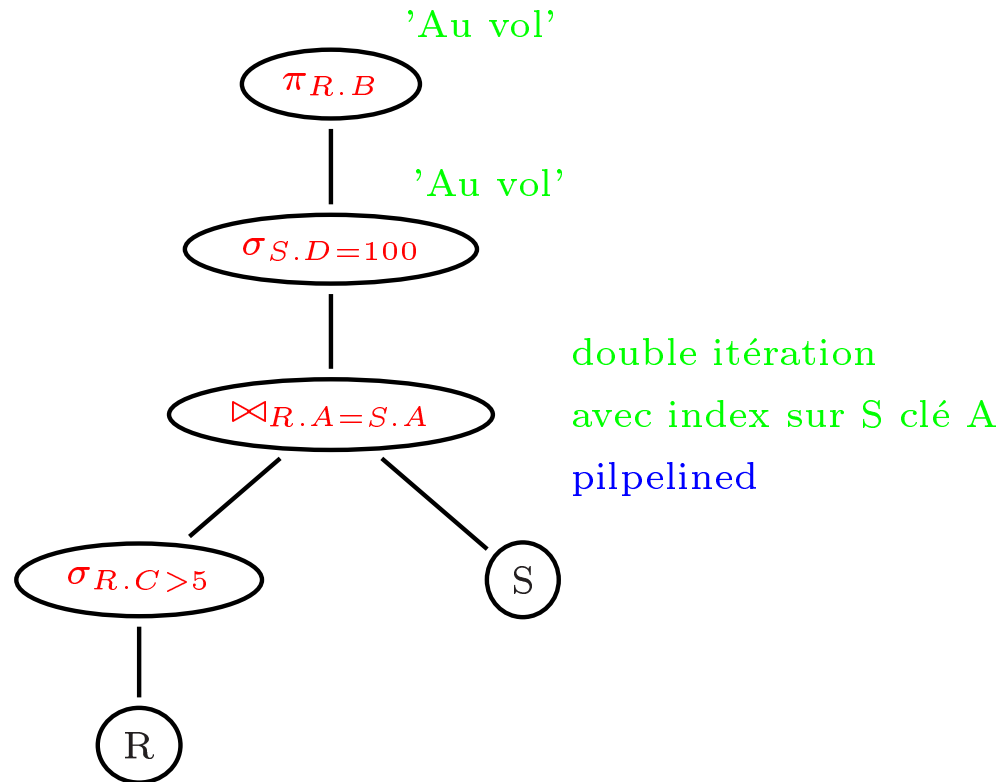


Autre plan d'exécution

Select R.B From R, S
Where R.A=S.A And
S.D=100 And R.C>5

index B+ sur R
clé C, groupant
pilpelled

Coût : exercice !



Plans d'exécution

- **Obtenir tous les plans d'exécutions !**
 - ordre des opérateurs
 - méthodes d'évaluation (en fonction des index)
 - ordre des opérandes (jointure)
- **Espace de recherche des plans**
 - limité aux "peignes gauches" (adaptés au pipelining)
- **Estimation** du coût d'un plan d'exécution
 - méthodes
 - taille des résultats intermédiaires

1. Recherche des "meilleurs" plans mono-relation
 2. Recherche des "meilleurs" plans 2-relations (jointure)
 - ...
 - n. Recherche des "meilleurs" plans n-relations
- "meilleurs" plans
 - coût minimal
 - coût minimal parmi les plans produisant des résultats ordonnés
-
- Autres stratégies (Heuristiques):
 - Hill Climbing, Dynamic Programming, Selinger-Style