



Fonctionnalités et Techniques des SGBDs Relationnels

Plan

- Pourquoi l'optimisation des requêtes est importante?
- Rappel du processus d'analyse et d'exécution de requêtes
- Optimisation algébrique
 - Règles de réécriture
 - Algorithme d'optimisation
- Limites de l'optimisation algébrique
- Optimisation interne
 - Principes
 - Algorithmes de sélection
 - Algorithmes de jointures
- Choix d'un plan d'exécution
 - Critère d'estimation des coûts
 - Le cas d'Oracle

Exemple

Table P

<u>NumP</u>	NomP	PrenomP	Ville
P1	Dupont	Pierre	Londres
P2	Durant	Thomas	Paris
P3	Haumont	Bastien	Paris
P4	Renard	Sylvain	Rome
P5	Thery	Mathieu	Paris

>< SP=Jointure de S et P

Table S

<u>NumS</u>	NomS	Type	Taille	Ville
S1	ActiveW	SARL	30	Paris
S2	Elec	EURL	5	Londres
S3	Factis	SA	125	Paris
S4	Lamini	SARL	22	Rome
S5	StatInter	SA	40	Londres
S6	Viabet	SARL	150	Paris

Table P x S

=> 30 tuples, 10 colonnes

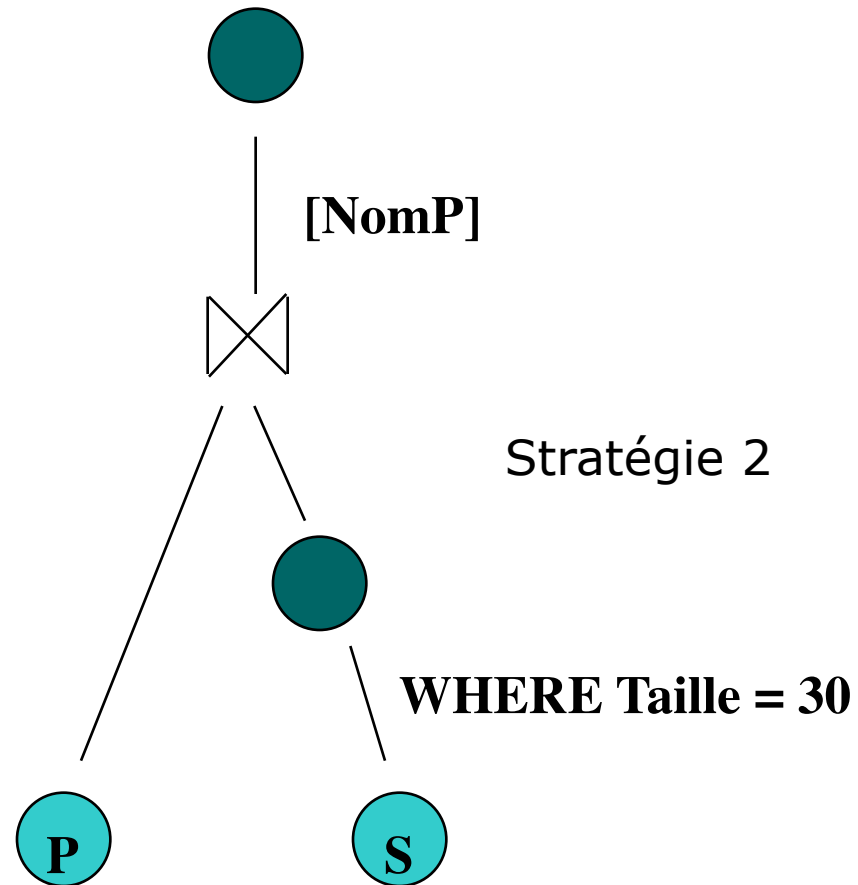
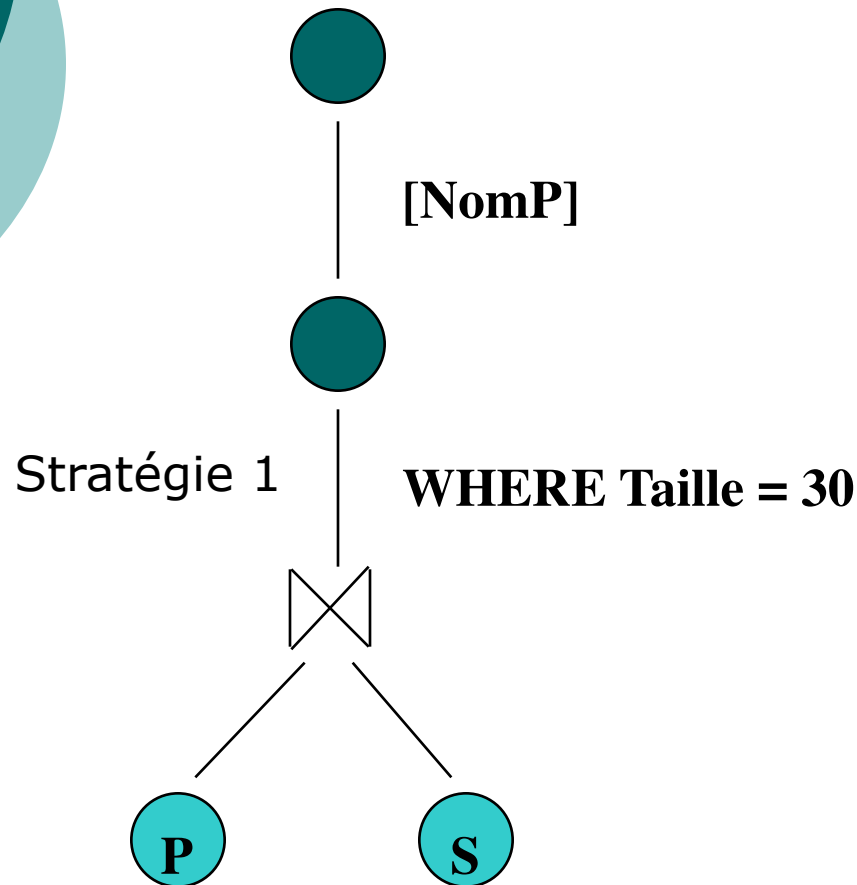
Jointure sur *Ville*

=> 12 tuples

Pourquoi optimiser l'évaluation d'une requête?

- Exemple de requête:
SELECT NomP FROM P , S
WHERE S.Ville = P.Ville
AND S.Taille = 30 ;
- Strategies possibles :
 1. ((P >< S) where S.Taille = 30) [NomP]
 2. (P >< (S where S.Taille = 30)) [NomP]

Graphes de la requête



Critères d' Evaluation des performances

- $|T|$: taille du résultat intermédiaire T
- $\{T\}$: nombre de tuples examinés pour produire T
- *Exemple:*
- $|P| = 100$
- $|S| = 500$

Note : dans le cas où il existe une intégrité référentielle entre P et S

Evaluation des performances

○ Stratégie 1:

| T = P > < S | ≤ |PS| = 500

| T' = T WHERE taille = 30 | = 5 (en moyenne)

| T'' = T' [NomP] | ≤ 5

{T} ≤ 100 * 500 = 50.000 (boucle imbriquée)

{T'} = 500

{T''} = 5 (en moyenne)

Stratégie 2

$| T = S \text{ WHERE taille} = 30 | = 5$ (en moyenne)

$| T' = P > < T | \leq | T | = 5$

$| T'' = T' [\text{NomP}] | \leq 5$

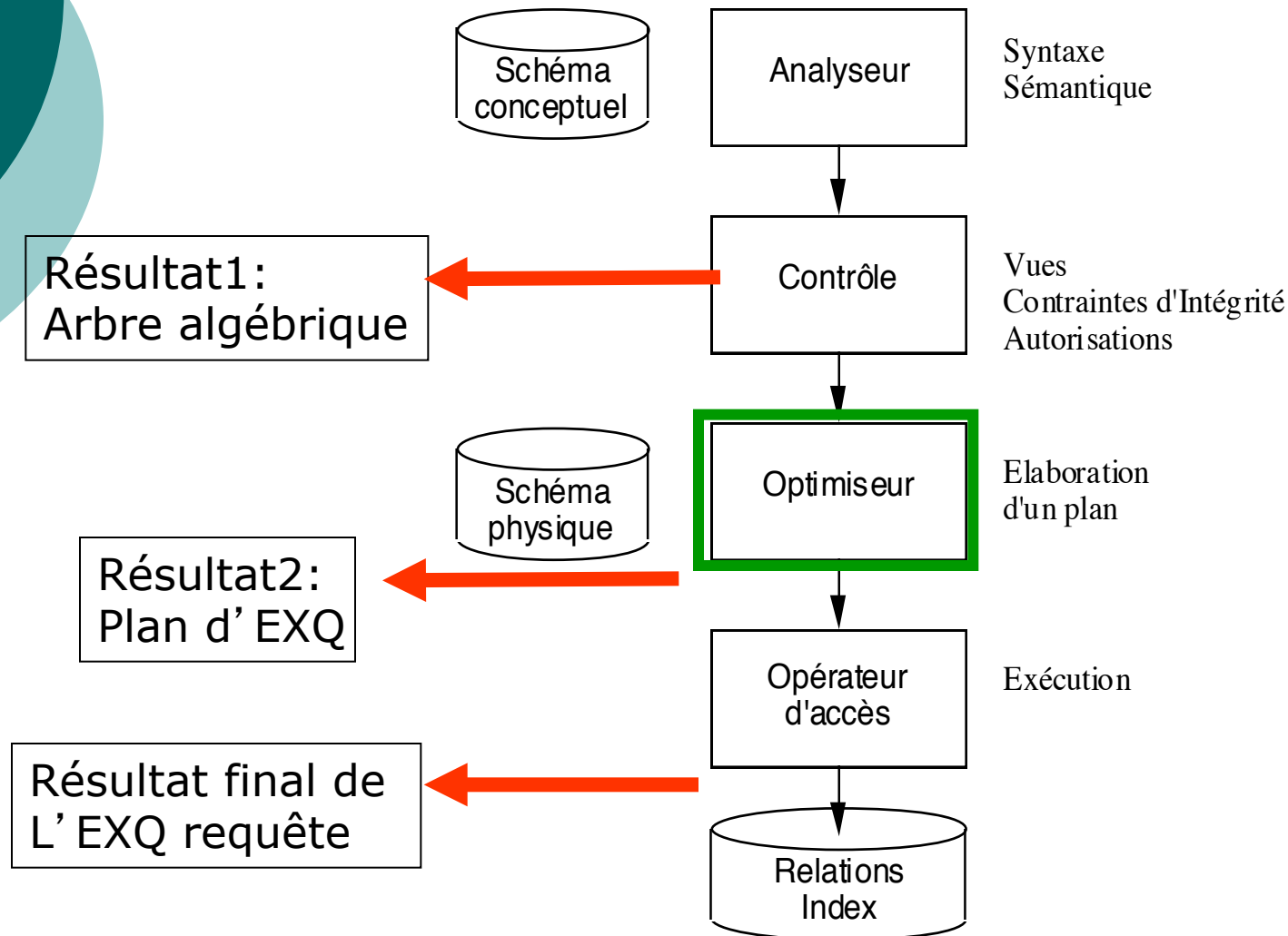
$\{T\} \leq 500$

$\{T'\} \leq 500$

$\{T_2\} \leq 5$

La stratégie 2 est plus intéressante

Traitement d'une Requête d'un SGBD Relationnel



Etapes de Traitement d'une Requête

- **Analyse**
 - Syntaxique
 - Syntaxe grammaticale
 - Sémantique
 - Existence des tables et des attributs => Métabase
- **Contrôle**
 - Contraintes d'Intégrité
 - Modification des Questions
 - Vues
 - réécriture du code de la requête (cf. SQL)
 - Déclencheurs
 - Autorisations
- **Optimisation**
 - Simplification – Normalisation + traduction algébrique
 - => arbre de requête
 - Restructuration
 - Choix d'un ordonnancement optimal des opérateurs
 - génération d'un plan d'exécution optimisé
 - choix entre différents algorithmes
- **Exécution**
 - Utilisation des méthodes d'accès
 - (B-Tree, Hachage, Index secondaires, ...)
 - Sélections - Projections - Elimination des doubles
 - Jointures
 - Agrégats



Vues

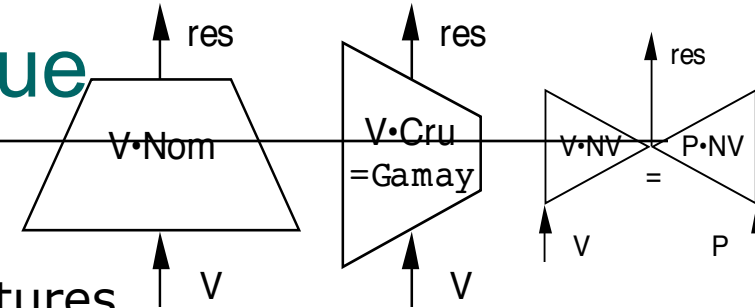
- Réécriture de la requête par rapport aux vues
- Vues matérialisées
 - Le résultat est pré-calculé et stocké dans une table
 - Les requêtes sont calculés au moyen de cette table
 - Propagation des mises à jour dans la vue



Phase d'optimisation

- Elaboration d'un plan d'exécution=
 - Ordonnancement des opérations d'un arbre algébrique en respectant :
 - Les dépendances sur les flux de données
 - En se basant sur l'associativité des opérations
- Pour chaque opération
 - Choix de l'algorithme le moins coûteux
 - Calcul des coûts
 - Réels
 - Statistiques
 - Sélection du plus petit coût

Règles de réécriture pour l'optimisation algébrique



○ Règles principales

(1) Commutativité des jointures

○ $(R1 \bowtie R2) \Leftrightarrow R2 \bowtie R1$

(2) Associativité des jointures

○ $(R1 \bowtie R2) \bowtie R3 \Leftrightarrow R1 \bowtie (R2 \bowtie R3)$

(3) Groupabilité des restrictions

● $\sigma(R, C1 \wedge C2) \Leftrightarrow \sigma(\sigma(R, C1), C2)$

(4) Semi-commutativité des projections et restrictions

● $\pi A1, A2, \dots, Ap(\sigma(R, Ai = val)) \Leftrightarrow \sigma(\pi A1, A2, \dots, Ap(R), Ai = val)$

● $\text{Ssi } i \in \{1, \dots, p\}$

(5) Semi-commutativité des restrictions et jointures

● $\sigma(R \bowtie S, A=val) \Leftrightarrow \sigma(R, A=val) \bowtie S$

● *Ssi A est un attribut de R*

Règles de réécriture pour l'optimisation algébrique(2)

○ Règles supplémentaires

(6) Semi-distributivité des projections / jointures

- $\pi_{A1,A2,\dots,Ap,B1,B2,\dots,Bq}(R \bowtie S) \Leftrightarrow$
- $\pi_{A1,A2,\dots,Ap}(R) \bowtie \pi_{B1,B2,\dots,Bq}(S)$

(7) Distributivité des restrictions / unions ou différences

- $\sigma(R \cup S, A=val) \Leftrightarrow \sigma(R, A=val) \cup \sigma(S, A=val)$

(8) Distributivité des projections / unions

- $\pi_{A1,A2,\dots,Ap}(R \cup S) \Leftrightarrow$
- $\pi_{A1,A2,\dots,Ap}(R) \cup \pi_{A1,A2,\dots,Ap}(S)$

Algorithme intuitif d'optimisation algébrique

- Toujours pousser vers le bas de l'arbre les restrictions (règles 3, 5 et 7)
 - Diminue le volume des résultats intermédiaires
- Commencer par les jointures les plus restrictives (règle 1 et 2)
 - Produisent moins de résultats
 - Allègent le reste du travail
- Pousser les projections sur l'attribut de jointure sous la jointure (règle 6)
- pousser au maxi les projections vers le bas (règles 4,7 et 8)

Exemple

- Application des règles de réécritures sur la requête :
 - Quels sont les noms et prénoms des buveurs parisiens qui ont commandé du Sauternes de 2012 en quantité > 8 ?

Limites de l'optimisation algébrique

- La restructuration algébrique est nécessaire mais non suffisante
- D'autres éléments doivent être pris en compte:
 - L'accès aux données (séquentiel, haché, indexé)
 - L'algorithme à retenir pour chaque opération
 - des caractéristiques des relations
 - taille des relations,
 - sélectivité des attributs, ...
 - Statistiques sur le peuplement de la base
 - de la configuration de la machine
 - mémoire primaire
 - accélérateur matériel, ...

Optimisation interne des requêtes

Algorithmes des Opérateurs

- Restriction
 - balayage séquentiel
 - fichier indexé (primaire, secondaire)
 - fichier haché
- Jointure
 - boucle imbriquée
 - tri-fusion
 - par hachage
 - avec index
- Projection
 - sans élimination des doublons
 - avec élimination des doublons
 - basé sur un tri de la relation à projeter
- Tri
- Agrégat

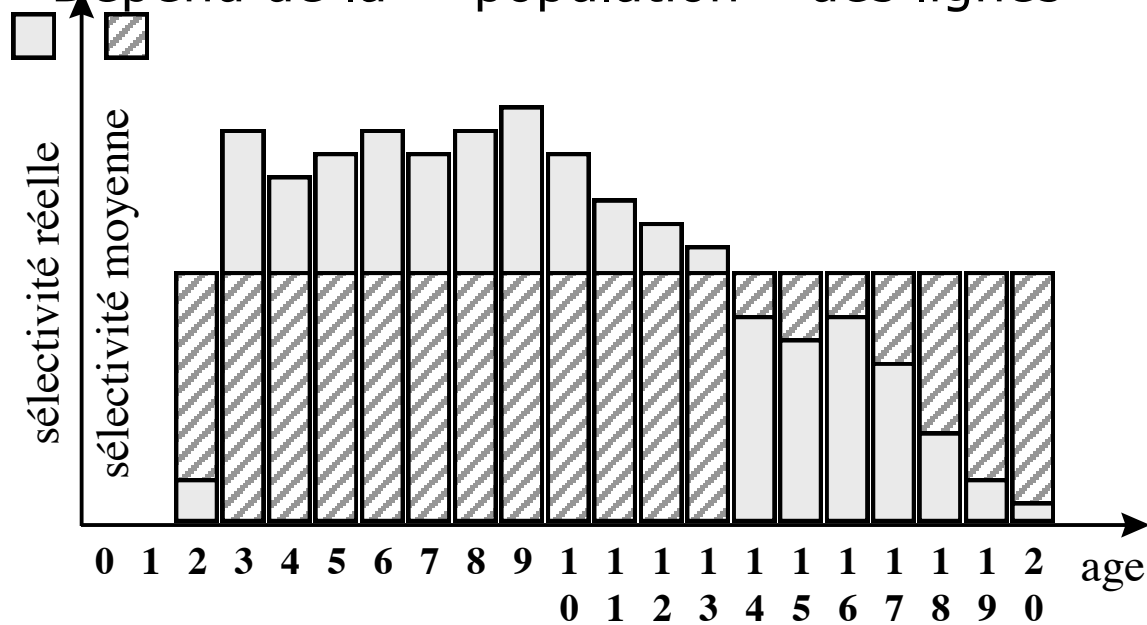
Restriction

Rappel: Supprime les lignes qui ne vérifient pas la condition

- Condition de Restriction
 - Comparaison entre une colonne et une valeur
 - Condition d'égalité à une valeur unique
 - `SELECT * FROM PERS WHERE SEX='MALE'`
 - Condition d'intervalle de valeurs
 - `SELECT * FROM PERS WHERE AGE>10 AND AGE<21`
 - Conjonction/Disjonction (AND,OR) sur plusieurs colonnes.
 - `SELECT * FROM PERS
WHERE SEX= 'MALE' AND AGE>10 AND AGE<21`

Sélectivité d' une Condition

- Taux de réduction de la relation source par restriction sur la condition.
- condition d'égalité
 - Restriction sur l'âge
 - Loi uniforme, Distribution de VA, Loi normal, ...
 - Dépend de la « population » des lignes



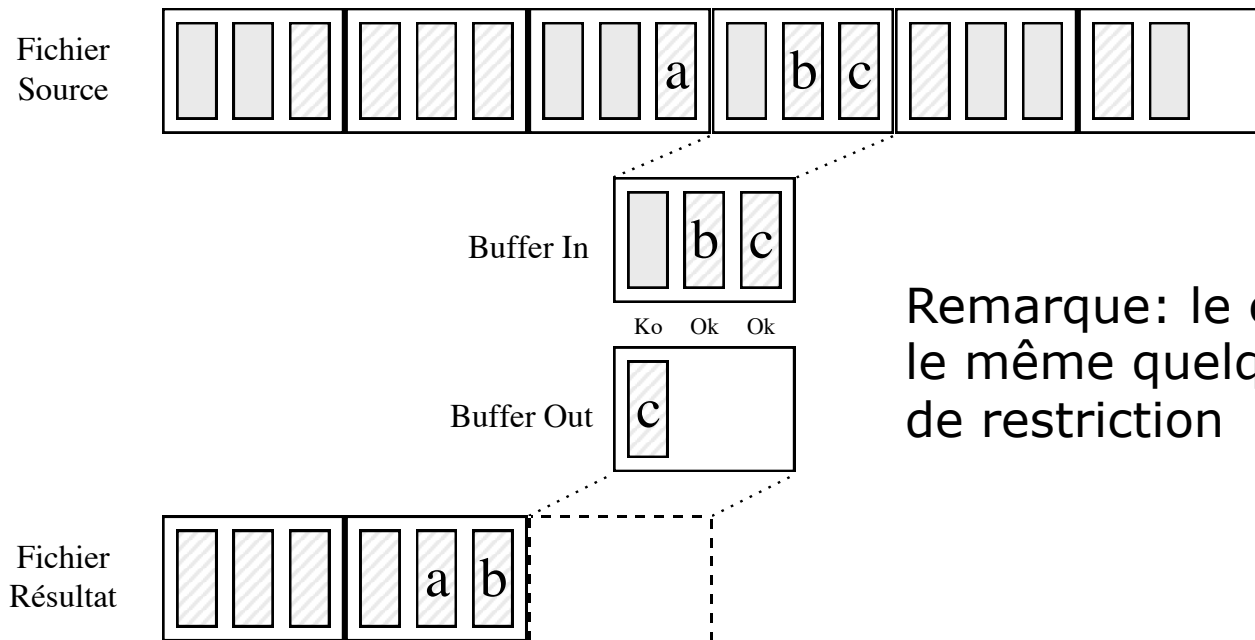
Propriétés de la sélectivité d'une question

- Condition $<$ (ex: AGE < 10)
 - $S_{\text{cond}} = \sum_{i \in [V_{\min}, V[} S_i$
- Condition $>$ (ex: AGE > 10)
 - $S_{\text{cond}} = \sum_{i \in]V, V_{\max}]} S_i$
- Condition d'intervalle (ex: AGE > 5 AND AGE < 10)
 - $S_{\text{cond}} = \sum_{i \in]V_1, V_2[} S_i$
- Condition C1 et C2 (ex: AGE > 5 AND SEX = 'M') *
 - $S_{C1 \text{ ET } C2} = S_{C1} * S_{C2}$
- Condition C1 OU C2 (ex: AGE > 5 OR SEX = 'M') *
 - $S_{C1 \text{ OU } C2} = S_{\neg C1} * S_{\neg C2}$

* Suppose que C1 et C2 portent sur des variables indépendantes

Restriction - Balayage Séquentiel

- Balayage séquentiel
 - Trivial ! : parcours des N blocs du fichier contenant la relation source et teste chaque tuple avec la condition



Remarque: le coût de l'écriture est le même quelque soit l'algorithme de restriction

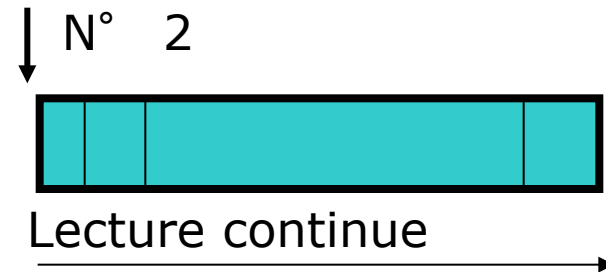
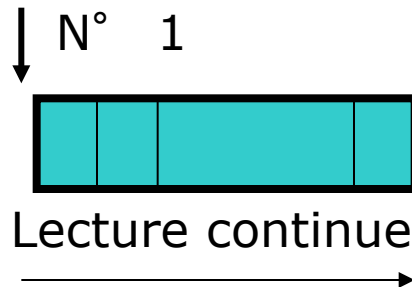
Restriction - Balayage Séquentiel (2)

TempsES (BAL) =

*BT * TempsTrans + NombrePos * TempsPosDébut*

*BT : nb de pages (blocs) qu'occupent sur disque la
table*

Positionnement tête de lecture



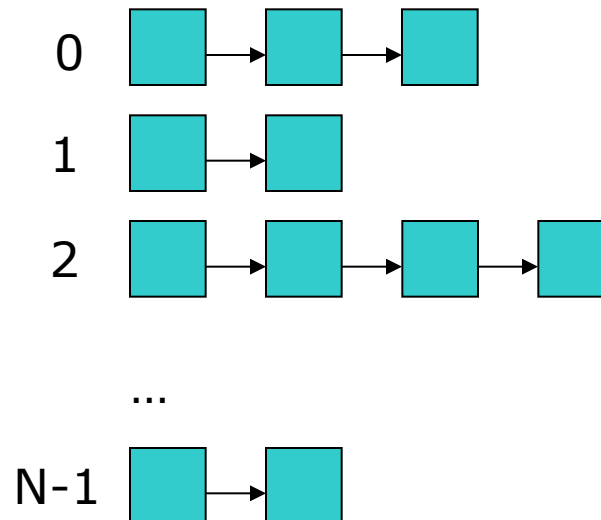
Restriction – Accès haché

- Fichiers Hachés

- SI attribut de restriction $\in \{ \text{attributs de placement} \}$
 $TempsES (S=H) = [N_T / (TH_T * FB_T)] * TempsESBloc$

Nt: nombre de tuples
THt: nombres de paquets
FBt: facteur de blocage

Ne convient pas aux
Traitements de requêtes
D'intervalles



Restriction - Accès par Index

- Rappel : Caractéristiques du Fichier Indexé
 - Attribut Clé / Non Clé
 - Trié / Non Trié (intervalle de valeurs)
- Condition d'égalité
 - Attribut Clé
 - l'index donne l'adresse du bloc où se trouve le tuple (au max 1 par valeur)
 - Attribut Non Clé
 - l'index donne l'adresse des blocs où se trouvent les tuples (si le fichier est trié sur l'attribut d'index, les tuples prenant la valeur de restriction sont groupés).
- Condition d'intervalle
 - Trié: l'index donne les adresses du premier bloc et du dernier bloc dans le fichier
- Cas d' Index Multiples
 - si plusieurs conditions "Attribut Opérateur Valeur"
 - On choisit t l' index de la condition la plus sélective

Restriction – Accès par index

- Coût Index Trié
 - lecture des blocs d'index dans le chemin vers le premier bloc contenant un enregistrement + lecture des blocs du fichier
- Coût Index Non Trié
 - lecture des blocs d'index + lecture des blocs du fichier (au max N)
- Remarque
 - Selon la sélectivité de la condition, un accès séquentiel peut être moins coûteux que l'accès indexé


Projection

- sans élimination des doublons
 - création d'un nouveau tuple en supprimant les attributs n'apparaissant pas dans la projection
 - cette réduction verticale est en général effectuée à la suite d'autres opérations (restriction, jointure, ...)
- avec élimination des doublons
 - Comme précédemment
 - Puis tri (sur disque) des tuples
 - Avec élimination les éléments en doubles en cours de tri
- Sélectivité
 - sans élimination des doublons
 - Même nombre de tuples que dans la rel source
 - Taille moyenne tuple projeté
= Taille moyenne du tuple source / $\sum \{\text{attributs projection}\}$ Taille moy. Attribut
 - avec élimination des doublons :
 - nombre de tuple inférieur à celui de la table source



Algorithmes de jointure

- Jointure sans index
 - Boucle imbriquée mais peu utilisé car coûteux si table de grandes tailles
 - Tri-Fusion
 - Jointure par hachage
- Jointure avec index
 - Boucle imbriquée
 - Sur la table externe
 - Sur la table interne



Equi-Jointure par Boucle Imbriquée (Nested Loop)

- Plusieurs variantes:

- par ligne
- par bloc
- multibloc
 - Raffinement
 - Arrêt de la boucle interne en cas de clé étrangère dès la première jointure
- avec index sur table interne
- Avec hachage sur table interne

Equi-Jointure par Boucle Imbriquée multibloc

- Hypothèse : $|R| < |S|$
 - R est dite table externe
- Algorithme
 - *Ressources : $N+2$ pages de mémoire primaire*
 - while(not eof(R)) {
 - lire N pages de R dans N pages de buffers
 - S.reset();
 - while(not eof(S)) {
 - lire 1 page de S dans 1 page de buffer
 - "marier" les tuples de R présents dans les N buffers
avec ceux de S présents dans le buffer
 - stocker dans le buffer de sortie

Equi-Jointure par Tri-Fusion (Sort-Merge)

- Algorithme

- *Ressources : $N+1$ pages de mémoire primaire*

- Trier S avec $N+1$ buffers

Trier R avec $N+1$ buffers

Lire les tuples de S et de R en parallèle

en "Mariant" les tuples de R et de S

- Remarque :

- le résultat est une relation triée sur l'attribut de jointure

- Algorithme souvent utilisé en cas d'absence d'index



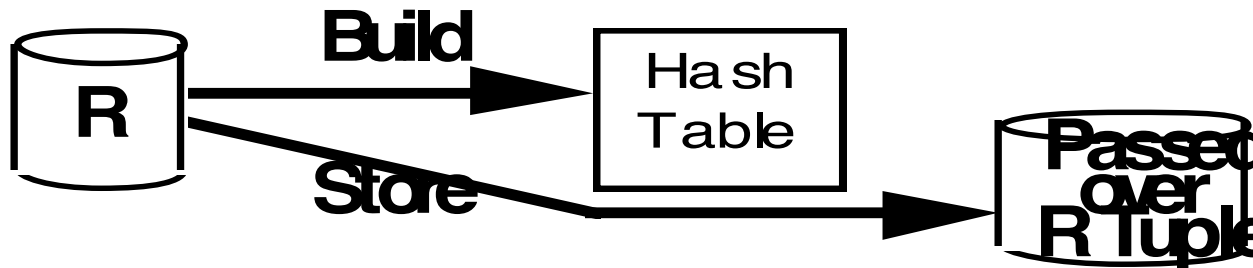
Equi-Jointure

Utilisation d' index

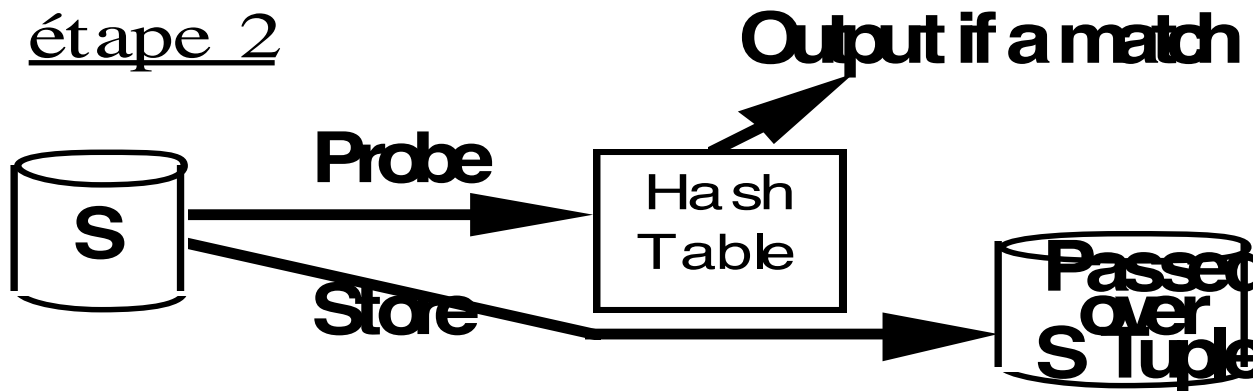
- Hypothèse
 - R est Indexé sur l'attribut de Jointure
- Algorithme
 - pour chaque page de S
 - pour chaque tuple dans cette page
 - chercher dans l'index de R le (ou les) tuples dont la valeur est celle de l'attribut de jointure de S
- Fusion d' index
 - Quand R et S tous les 2 sont indexés sur l'attribut de Jointure

Equi-Jointure

Hachage Simple
étape 1

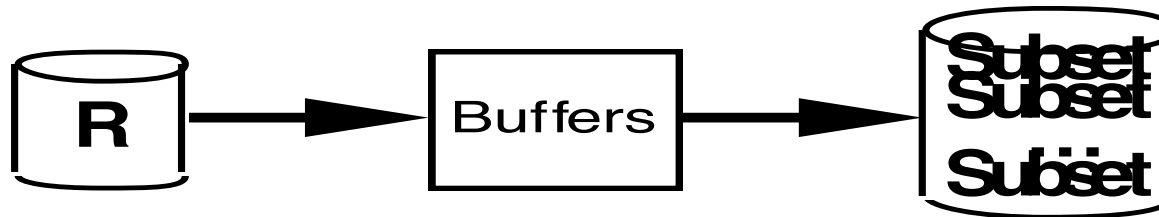


étape 2

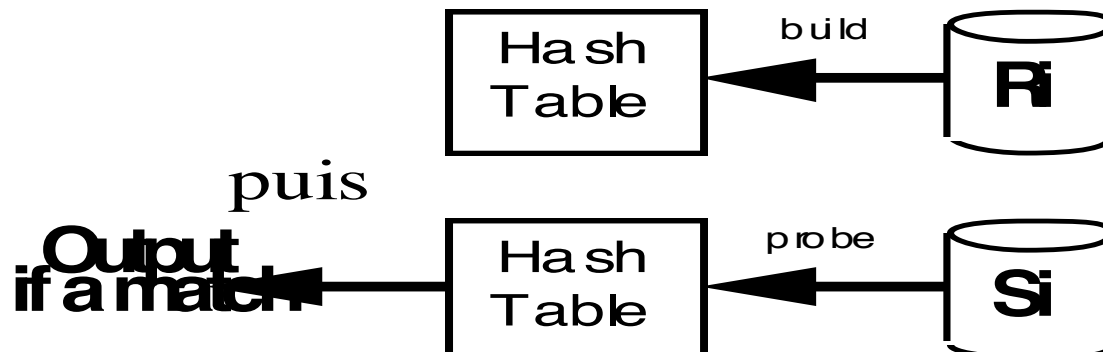


Equi-Jointure

- o Hachage "Grace" (Univ of Tokyo)
phase 1



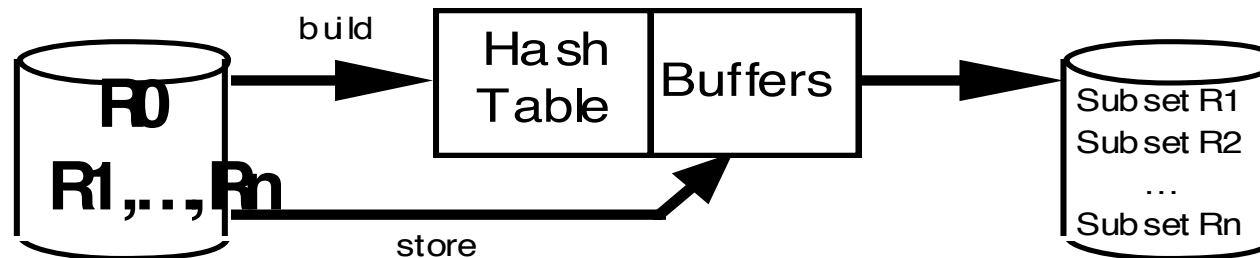
phase 2



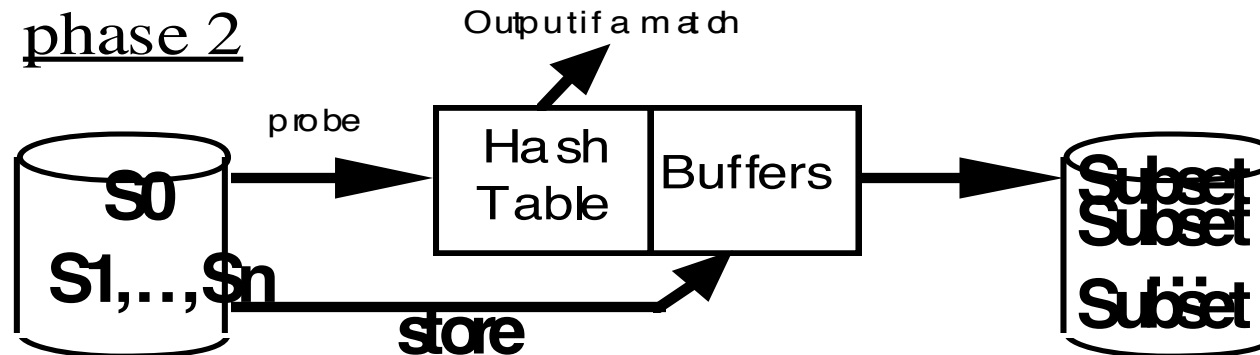
Equi-Jointure

- Hachage Hybride (Univ of Wisconsin-Madison)

phase 1



phase 2





L' effet des statistiques

- Le système choisira:
 - La boucle imbriquée simple si les tables sont de petites tailles
 - Le balayage séquentiel au lieu d'un parcours indexée si la sélectivité est trop grande
- Sources d'information
 - Echantillonnages
 - Calcul statistique périodique

Taille du Résultat d'une Jointure

- Evaluation de la Taille de l'article
 - Concaténation des 2 articles
- Evaluation du Nombre d'articles de jointure a_j
 - dépendant de la nature des attributs de jointure (S.A et R.B) pour lesquels il existe peut être des contraintes d'intégrité référentielle
 - dépendant du nombre de valeurs discrètes partagées entre l'ensemble des valeurs de S.A et l'ensemble des valeurs de R.B
- Contraintes d'intégrité référentielle
 - Exemple : S.A références R.B
 - $a_j = a_s$
 - Remarque :
 - Si une restriction de sélectivité SR porte sur R.B précède la jointure
 - $a_j = SR * a_s$

Sélectivité du critère de jointure

- Simplification

- pour simplifier, on suppose un coefficient de sélectivité SJ
- $SJ = a_j / (a_r * a_s)$
- $SJ = 0$ si aucun tuple ne joint
- $SJ = 1 / \text{MAX}(\text{NDIST}(A_r), \text{NDIST}(A_s))$
 - si distribution uniforme équiprobable des attributs A_r et A_s sur un même domaine
- $SJ=1$ pour un produit cartésien

Jointures Pré-Calculées

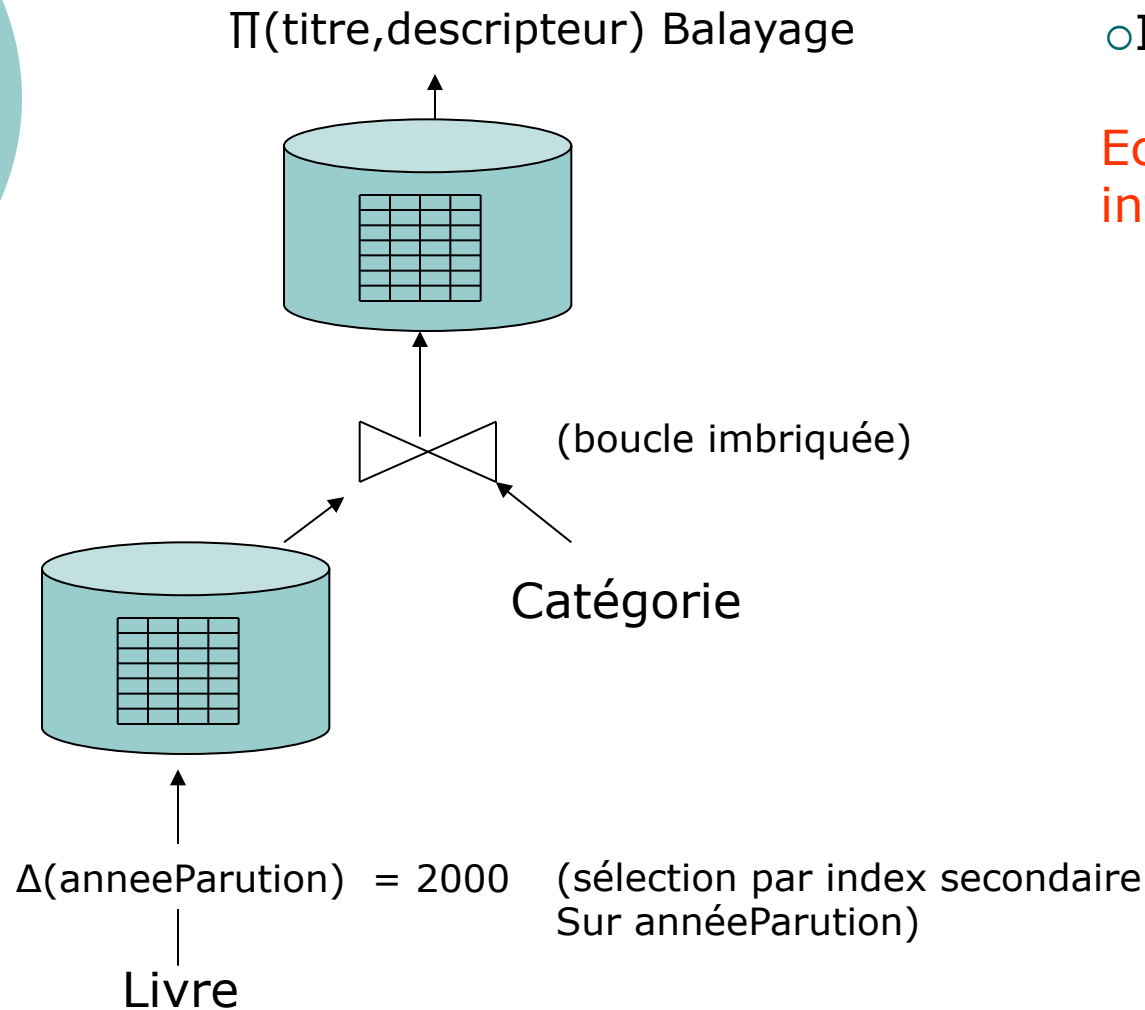
- Table contenant le résultat de la jointure
- Schéma
 - J(AS, ROWIDS, AR, ROWIDR)
 - ROWIDS est une adresse physique de ligne de S
 - ROWIDR est une adresse physique de ligne de R
- Mise à jour (delete, insert, update) de R et de S
 - Entraîne un Re-calcul partiel de J à partir de J, S, R



Mise en œuvre d'un plan d'exécution

- Objectif :
 - Réduire le coût de réalisation du plan d'exécution
- Plusieurs solutions
 - Mise en œuvre par matérialisation
 - Mise en œuvre par pipeline

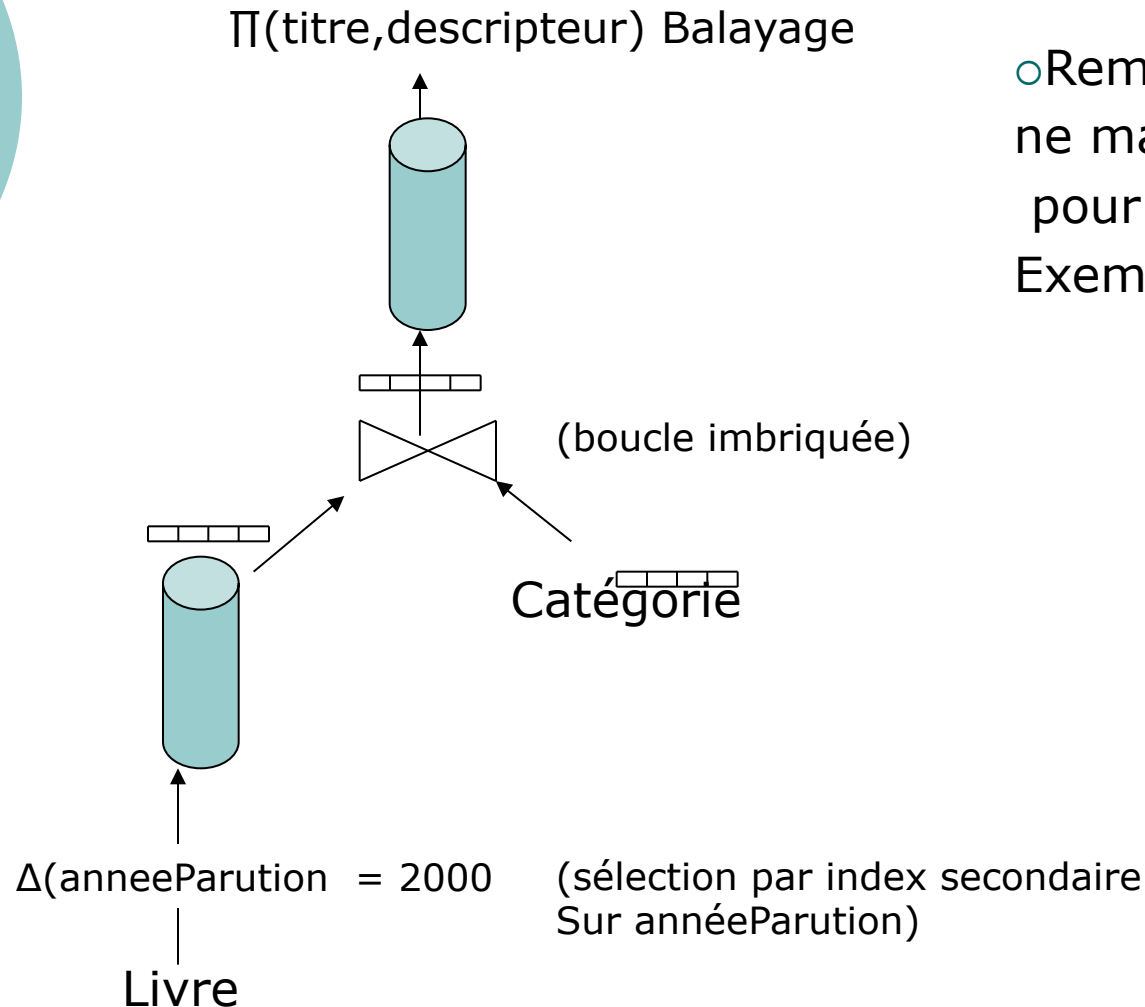
Mise en œuvre par matérialisation



○ Inconvénients

Ecriture des tables
intermédiaires sur disque!

Mise en œuvre par pipeline



○ Remarque:
ne marche pas
pour toutes les situations
Exemple: Tri-fusion



Quelques autres règles utiles

- Combinaison avec une opération binaire précédente
 - jointure et projection
 - jointure et restriction
- Combiner les opérations unaires
 - restriction et projection
- Commencer par les jointures qui impliquent des tables en cluster, avec index ou hachées.



Optimisation de requête dans Oracle

- Oracle permet:
 - Définition d'index : il s'agit d'arbre B+
 - Hachage
 - Clustering : regrouper physiquement des données susceptibles d'être concernées par une même jointure
 - Index sur la clé de jointure

Oracle : types d'accès

- accès séquentiel
 - Full TABLE SCAN <nom de table>
- accès par adresse
 - Accès à un tuple connaissant son adresse
 - ACCESS BY ROWID
- Parcours de données en « cluster »
 - CLUSTER SCAN
- Accès par hachage
 - HASH SCAN <nom de la table>
- Parcourir index
 - INDEX SCAN



Algorithmes de jointure utilisés

- Boucle imbriquée
 - Si un ou deux indexs car moins couteux
 - NESTED LOOP
- Tri fusion
 - Si pas d'index
 - SORT
 - MERGE
- CLUSTERING

L'outil EXPLAIN

- Permet de visualiser le plan d'exécution retenu
- Utilisation
 - EXPLAIN PLAN SET statement = « un_nom »
FOR <clause SFW>
 - Exemple
 - EXPLAIN PLAN SET statement = « un_nom »
FOR SELECT * FROM VINS WHERE COULEUR
=« Rouge »
 - Resultat
 - **SELECT STATEMENT**
 - **1 TABLE ACCESS FULL VINS**

Autre exemple

```
select desig, marque, prix
  from Produits, PrixFour, NoteMag
 where Produits.code=PrixFour.code
    and Produits.code=NoteMag.code
    and note > 8;
```

0 SELECT STATEMENT

1 MERGE JOIN

2 SORT JOIN

3 NESTED LOOPS

4 TABLE ACCESS FULL NOTEMAG

5 TABLE ACCESS BY INDEX ROWID PRODUITS

6 INDEX UNIQUE SCAN A34561

7 SORT JOIN

8 TABLE ACCESS FULL PRIXFOUR

« soulager » l'optimiseur

- Problématique

- Pour une même requête :
 - Plusieurs représentations internes équivalentes
 - Plusieurs choix d'implantation
 - Plusieurs plans d'exécutions possibles

- Objectif

- Réduire le nombre de plans d'EXQ

- Solutions

- Eliminer les plans inapplicables
- Heuristiques

NOM	PRENOM
dupont	david
achour	hakim
durant	laurent
loiseau	kevin
cruise	tom
jolie	angelina
pitt	brad
roselmack	harry
pajamas	david
delahousse	laurent
chazal	claire

NOM	PRENOM
lucet	elise
aliagas	nikos
castaldi	benjamin
dechavanne	christophe
leymergie	william
davant	sophie
durant	paul
thomas	estelle

19 ligne(s) s?lectionn?e(s).

Ecoule? : 00 :00 :00.01

Plan d'ex?cution

Plan hash value: 979701618

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		19	456	3 (0)	00:00:01
1	TABLE ACCESS FULL	ETUDIANT	19	456	3 (0)	00:00:01

Note

- dynamic sampling used for this statement

NOM	PRENOM
dupont	david
achour	hakim
durant	laurent
loiseau	kevin
cruise	tom
jolie	angelina
pitt	brad
roselmack	harry
pajamas	david

9 ligne(s) s?lectionn?e(s).

Ecoule? : 00 :00 :00.00

Plan d'ex?cution

Plan hash value: 979701618

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		9	324	3 (0)	00:00:01
* 1	TABLE ACCESS FULL	ETUDIANT	9	324	3 (0)	00:00:01

Predicate Information (identified by operation id):

1 - filter("VILLE"='valenciennes')

Note

- dynamic sampling used for this statement

NENT	NETUD	NENS	REMUN	ANNE
1	1	1	300	2012
3	3	3	0	2012
4	4	4	150	2012
1	5	5	600	2012
3	6	6	300	2012
4	7	2	200	2012
1	8	4	0	2012
3	9	4	100	2012
4	10	2	300	2012
1	11	1	200	2012
2	13	3	200	2012

NENT	NETUD	NENS	REMUN	ANNE
3	15	5	300	2012
4	17	6	500	2012

13 ligne(s) sélectionné(s).

Ecoulé : 00 :00 :00.00

Plan d'exécution

Plan hash value: 2239145125

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		13	728	3 (0)	00:00:01
* 1	TABLE ACCESS FULL	STAGE	13	728	3 (0)	00:00:01

Predicate Information (identified by operation id):

1 - filter("ANNEE"='2012')

NENS NOM	PRENOM	INSTITUT
1 niar	smaïl	ISTV
2 wilbaut	christophe	ISTV
3 desertot	mikael	ISTV
4 lecomte	sylvain	ISTV

Ecoulé : 00 :00 :00.00

Plan d'exécution

Plan hash value: 3986204123

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		4	196	3 (0)	00:00:01
* 1	TABLE ACCESS FULL	ENSEIGNANT	4	196	3 (0)	00:00:01

Predicate Information (identified by operation id):

1 - filter("INSTITUT"='ISTV')

NOM	PRENOM
-----	-----
dupont	david
achour	hakim
durant	laurent
loiseau	kevin
cruise	tom
jolie	angelina
pitt	brad
roselmack	harry
pajamas	david

9 ligne(s) s?lectionn?e(s).

Ecoule? : 00 :00 :00.02

Plan d'ex?cution

Plan hash value: 2351865347

Id	Operation	Name	Rows	Bytes	Cost (%CPU)
-----	-----	-----	-----	-----	-----
0	SELECT STATEMENT		9	324	2 (0)
00:00:01					
1	TABLE ACCESS BY INDEX ROWID	ETUDIANT	9	324	2 (0)
00:00:01					
* 2	INDEX RANGE SCAN	INDEXVILLE	9		1 (0)
00:00:01					

Predicate Information (identified by operation id):

2 - access("VILLE"='valenciennes')

NOM PRENOM

achour hakim

Ecoule : 00 :00 :00.01

Plan d'exécution

Plan hash value: 1760408377

Id	Operation	Name	Rows	Bytes	Cost (%CPU)
----	-----------	------	------	-------	-------------

0	SELECT STATEMENT		1	37	1 (0)
---	------------------	--	---	----	-------

1	TABLE ACCESS BY INDEX ROWID	ETUDIANT	1	37	1 (0)
---	-----------------------------	----------	---	----	-------

* 2	INDEX UNIQUE SCAN	SYS_C0053833	1		1 (0)
-----	-------------------	--------------	---	--	-------

Predicate Information (identified by operation id):

2 - access("NETUD"=3)

Plan d'exécution

Plan hash value: 3659458962

-

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
<hr/>						
-						
0	SELECT STATEMENT		15	1125	7 (15)	00:00:01
* 1	HASH JOIN		15	1125	7 (15)	00:00:01
2	TABLE ACCESS FULL	ENTREPRISE	4	140	3 (0)	00:00:01
3	TABLE ACCESS FULL	STAGE	15	600	3 (0)	00:00:01

-

Predicate Information (identified by operation id):

1 - access("ENTREPRISE"."NENT"="STAGE"."NENT")

Plan d'exécution

Plan hash value: 1465214984

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		42	3612	9 (12)	00:00:01
* 1	HASH JOIN		42	3612	9 (12)	00:00:01
2	NESTED LOOPS		42	2142	5 (0)	00:00:01
3	TABLE ACCESS FULL	ETUDIANT	19	475	3 (0)	00:00:01
* 4	INDEX RANGE SCAN	SYS_C0053837	2	52	1 (0)	00:00:01
5	TABLE ACCESS FULL	FORMATION	7	245	3 (0)	00:00:01

Predicate Information (identified by operation id):

- 1 - access("INSCRIPTIONS"."NFORM"="FORMATION"."NFORM")
- 4 - access("INSCRIPTIONS"."NETUD"="ETUDIANT"."NETUD")

NOM	SUM(STAGE.REMUN)
dupont	300
delahousse	200
loiseau	600
pitt	0

Ecoule : 00 :00 :00.01

Plan d'exécution

Plan hash value: 3921539622

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		4	396	11 (19)	00:00:01
1	HASH GROUP BY		4	396	11 (19)	00:00:01
* 2	HASH JOIN		4	396	10 (10)	00:00:01
3	MERGE JOIN CARTESIAN		19	1140	6 (0)	00:00:01
* 4	TABLE ACCESS FULL	ENTREPRISE	1	35	3 (0)	00:00:01
5	BUFFER SORT		19	475	3 (0)	00:00:01
6	TABLE ACCESS FULL	ETUDIANT	19	475	3 (0)	00:00:01
7	TABLE ACCESS FULL	STAGE	15	585	3 (0)	00:00:01

Predicate Information (identified by operation id):

- 2 - access("ENTREPRISE"."NENT"="STAGE"."NENT" AND "STAGE"."NETUD"="ETUDIANT"."NETUD")
- 4 - filter("ENTREPRISE"."NOM"='ATOS')

NETUD	NOM	PRENOM	VILLE	CP
12	chazal	claire	lille	59000
19	durant	paul	lille	59000
18	davant	sophie	lille	59000
14	aliagas	nikos	lille	59000
16	dechavanne	christophe	lille	59000
20	thomas	estelle	lille	59000

6 ligne(s) s?lectionn?e(s).

Ecoule? : 00 :00 :00.02

Plan d'ex?cution

Plan hash value: 363209155

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		6	396	6 (17)	00:00:01
* 1	HASH JOIN ANTI		6	396	6 (17)	00:00:01
2	TABLE ACCESS FULL	ETUDIANT	19	1007	3 (0)	00:00:01
3	INDEX FAST FULL SCAN	SYS_C0053840	15	195	2 (0)	00:00:01

Predicate Information (identified by operation id):

1 - access("NETUD"="NETUD")



Bibliographie

- Date
- Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom, "Database System Implementation", 2000, Ed Prentice Hall, ISBN 0-13-040264-8
- Steve Adams, "Oracle 8i Internal Services", O Reilly, 1999, ISBN1-56592-598-X
- Cours de G. Gardarin
- Cours de W. LITWIN, « Optimisation algébriques de requêtes relationnelles »
- Cours Rigaux