

Résolution de requêtes - Optimisation





Objectif

- L'accès aux données d'une BDR est en général faite à l'aide d'un langage de requêtes de haut niveau comme SQL
- SQL est déclaratif
 - L'utilisateur indique ce qu'il veut obtenir
 - Il n'indique pas comment l'obtenir
- Le système doit comprendre et exécuter la requête
 - Il peut y avoir plusieurs façons de répondre à cette requête
 - Le système doit choisir la stratégie la plus optimale

Rappel – Une requête

- Analyse syntaxique
- **Optimisation**
*Génération d'une requête/d'un programme optimisé(e)
à partir de la connaissance de la structure des données, de
l'existence d'index, de statistiques sur les données*
- Contrôles / Exécution
Sécurité, confidentialité, concurrence, intégrité
- Réponse

3

Résolution d'une requête : deux idées

- (1) Réécriture - transformation par :
 - simplification
 - ordonnancement des opérations élémentaires
- (2) Construction de plans d'exécution candidats
 - choix des algorithmes pour chaque opérateur,
 - calcul du coût de chaque plan,
 - choix du meilleur plan

Etape 1 : indépendante des données

Etape 2 : dépendante des données

Phases de résolution d'une requête

- Traduction de la requête à l'aide de l'algèbre relationnelle
 - Arbre de requête - arbre d'opérateurs (sélection, projection, jointure etc.)
- Optimisation
 1. transformation de l'arbre de requête en un arbre équivalent, basée sur les propriétés d'associativité et de commutativité de l'algèbre relationnelle
 2. déterminer un plan d'exécution de coût minimal - évaluation du coût de résolution de chaque opérateur

plan d'exécution logique – PEL
Plan d'exécution physique – PEP
- Évaluation du plan d'exécution produit

Exemple intuitif

CINEMA(Cinéma, Adresse, Gérant)
SALLE(Cinéma, NoSalle, Capacité)

Adresse des cinémas ayant des salles de plus de 150 places

```
SELECT Adresse
FROM  CINEMA, SALLE
WHERE capacité > 150
AND   CINEMA.cinéma = SALLE.cinéma
```

Hypothèses

- Il y a 300 n-uplets dans CINEMA, occupant 30 pages (10 cinémas/page)
- Il y a 1200 n-uplets dans SALLE, occupant 120 pages (10 salles/page)
- La mémoire centrale (tampon) ne contient qu'une seule page par relation

CINEMA(Cinéma, Adresse, Gérant)
SALLE(Cinéma, NoSalle, Capacité)

Exemple

```
SELECT Adresse
FROM  CINEMA, SALLE
WHERE capacité > 150
AND   CINEMA.cinéma = SALLE.cinéma
```

1. $\pi_{\text{Adresse}}(\sigma_{\text{Capacité} > 150}(\text{CINEMA} \bowtie \text{SALLE}))$
2. $\pi_{\text{Adresse}}(\text{CINEMA} \bowtie \sigma_{\text{Capacité} > 150}(\text{SALLE}))$

2 traductions en algèbre relationnelle

- Une jointure suivie d'une restriction/sélection puis projection
- Une sélection/restriction suivie d'une jointure puis projection

→ Le numéro 2 est meilleur

CINEMA(Cinéma, Adresse, Gérant)
SALLE(Cinéma, NoSalle, Capacité)

$$1. \pi_{\text{Adresse}} (\sigma_{\text{Capacité} > 150}(\text{CINEMA} \bowtie \text{SALLE}))$$

Evaluation des couts

- On suppose que
 - 5% de salles ont plus de 150 places
 - les résultats intermédiaires d'une opération et le résultat final sont écrits sur disque (10 n-uplets par page)
- Cas 1 : Jointure en premier
 - *Jointure*
 - on lit 3 600 pages (120x30)
 - on écrit le résultat intermédiaire (120 pages)
 - *Sélection*
 - on relit le résultat
 - on projette sur les attributs concernés de CINEMA, on obtient 5% de 120 pages, soit 6 pages

$$\rightarrow \text{Nombre d'E/S : } 3\,600E + 120 \times 2E/S + 6S = 3\,846$$

$$2. \pi_{\text{Adresse}} (\text{CINEMA} \bowtie \sigma_{\text{Capacité} > 150}(\text{SALLE}))$$

Evaluation des coûts

- On suppose que
 - 5% de salles ont plus de 150 places
 - les résultats intermédiaires d'une opération et le résultat final sont écrits sur disque (10 n-uplets par page)
- Cas 2 : Sélection en premier
 - *Sélection*
 - on lit 120 pages (salles)
 - on obtient (écrit) 5% des 120 pages soit 6 pages
 - *Jointure*
 - on lit 180 pages (6x30)
 - on obtient 6 pages

$$\rightarrow \text{Nombre d'E/S : } 120E + 6S + 180E + 6S = 312$$

Premier Bilan



- Traduction de la requête en un arbre de requête
- Optimisation
 1. Des règles de réécriture des expressions de l'algèbre : *des arbres équivalents*
 2. Des connaissances sur l'organisation physique de la base
 3. Des statistiques sur les caractéristiques de la base (taille des relations par exemple)
 - *coût des opérations*
- Un modèle de coût permet donc de classer les différentes stratégies envisagées

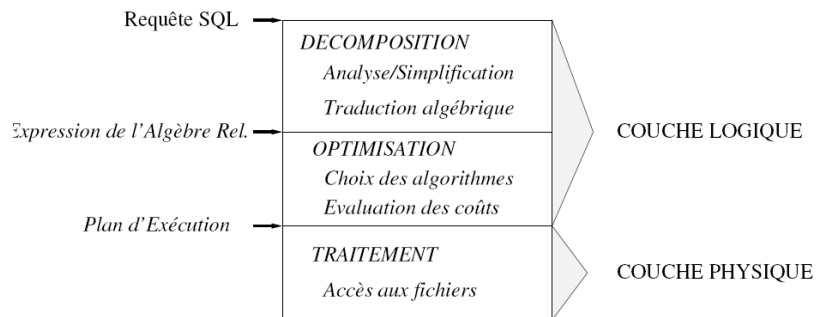
Evaluation efficace d'une requête



- Minimiser le temps
 - d'évaluation
 - Temps pour exécuter entièrement la requête
 - de réponse
 - Temps pour donner à l'utilisateur le premier résultat
- Le temps d'évaluation
 - nombre de pages accédées
- Pas de prise en compte de l'écriture du résultat
 - dépend de sa taille
 - ne dépend pas de l'algorithme choisi

Architecture du SGBD et résolution de requête

LES ETAPES DU TRAITEMENT D'UNE REQUÊTE



Analyse syntaxique

- Vérification de la validité syntaxique de la requête
 - Contrôle de la structure grammaticale
 - Vérification de l'existence des relations et des noms d'attributs

Analyse, simplification et normalisation

- Analyse sémantique pour la détection d'incohérences
 - $NoSalle = 11 \text{ AND } NoSalle = 12$
- Simplification de clauses inutilement complexes
 - $(A \text{ OR } NOT B) \text{ AND } B$ est équivalent à $A \text{ AND } B$
- Normalisation de la requête
 - transformation des conditions en forme normale conjonctive
 - décomposition en blocs SELECT-FROM-WHERE pour faciliter la traduction algébrique
 - Arguments du *select* : projections
 - Arguments du *where* : selon la forme, marquent une jointure ou une sélection

CINEMA(Nom, adresse, gérant)
 SALLE(#Nom,#noSalle, capacité)
 SEANCE(#noSalle,film, heure-début)

....

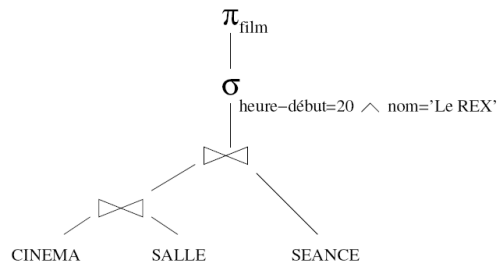
Traduction algébrique

- Déterminer l'expression algébrique équivalente à la requête et la représenter sous forme d'arbre de requête

```
SELECT  film
FROM    CINÉMA, SALLE, SÉANCE
WHERE   CINÉMA.nom = 'Le Rex'
AND     SÉANCE.heure-début = 20
AND     CINÉMA.nom = SALLE.nom
AND     SALLE.nosalle = SÉANCE.nosalle
```

Quels films passent au Rex à 20h ?

Expression algébrique – arbre de requête

$$\pi_{film}(\sigma_{Nom='Le\ Rex'\wedge heure-début=20}((CINEMA \bowtie SALLE) \bowtie SEANCE))$$


Quels films passent au Rex à 20h ?

Restructuration

- Règles de réécriture
 - Expressions équivalentes pour une même requête
 - Suivant l'ordre des opérateurs algébriques dans un arbre, le coût d'exécution est différent
- Pourquoi?
 - le coût des opérateurs varient en fonction du volume des données traitées
 - plus le nombre de n-uplets des relations traitées est petit, plus les coûts cpu et d'E/S sont minimisés
 - certains opérateurs diminuent le volume des données
 - Restriction/sélection et projection

Règles de réécriture

- Commutativité des jointures :

$$R \bowtie S \equiv S \bowtie R$$

- Associativité des jointures :

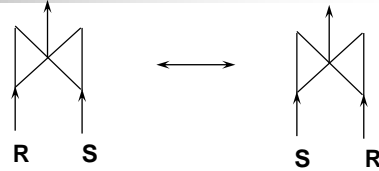
$$(R \bowtie S) \bowtie T \equiv R \bowtie (S \bowtie T)$$

- Regroupement des sélections :

$$\sigma_{A=a' \wedge B=b'}(R) \equiv \sigma_{A=a'}(\sigma_{B=b'}(R))$$

- Commutativité de la sélection et de la projection

$$\pi_{A_1, A_2, \dots, A_p}(\sigma_{A_i=a'}(R)) \equiv \sigma_{A_i=a'}(\pi_{A_1, A_2, \dots, A_p}(R)), i \in \{1, \dots, p\}$$



Règles de réécriture

- Commutativité des jointures :

$$R \bowtie S \equiv S \bowtie R$$

- Associativité des jointures :

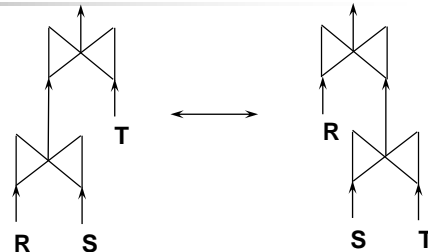
$$(R \bowtie S) \bowtie T \equiv R \bowtie (S \bowtie T)$$

- Regroupement des sélections :

$$\sigma_{A=a' \wedge B=b'}(R) \equiv \sigma_{A=a'}(\sigma_{B=b'}(R))$$

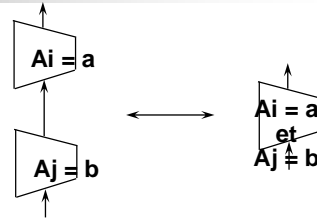
- Commutativité de la sélection et de la projection

$$\pi_{A_1, A_2, \dots, A_p}(\sigma_{A_i=a'}(R)) \equiv \sigma_{A_i=a'}(\pi_{A_1, A_2, \dots, A_p}(R)), i \in \{1, \dots, p\}$$



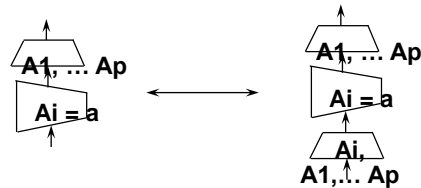
Règles de réécriture

- **Commutativité des jointures :**
 $R \bowtie S \equiv S \bowtie R$
- **Associativité des jointures :**
 $(R \bowtie S) \bowtie T \equiv R \bowtie (S \bowtie T)$
- **Regroupement des sélections :**
 $\sigma_{A=a' \wedge B=b'}(R) \equiv \sigma_{A=a'}(\sigma_{B=b'}(R))$
- **Commutativité de la sélection et de la projection**
 $\pi_{A_1, A_2, \dots, A_p}(\sigma_{A_i=a'}(R)) \equiv \sigma_{A_i=a'}(\pi_{A_1, A_2, \dots, A_p}(R)), i \in \{1, \dots, p\}$



Règles de réécriture

- **Commutativité des jointures :**
 $R \bowtie S \equiv S \bowtie R$
- **Associativité des jointures :**
 $(R \bowtie S) \bowtie T \equiv R \bowtie (S \bowtie T)$
- **Regroupement des sélections :**
 $\sigma_{A=a' \wedge B=b'}(R) \equiv \sigma_{A=a'}(\sigma_{B=b'}(R))$
- **Commutativité de la sélection et de la projection**
 $\pi_{A_1, A_2, \dots, A_p}(\sigma_{A_i=a'}(R)) \equiv \sigma_{A_i=a'}(\pi_{A_1, A_2, \dots, A_p}(R)), i \in \{1, \dots, p\}$



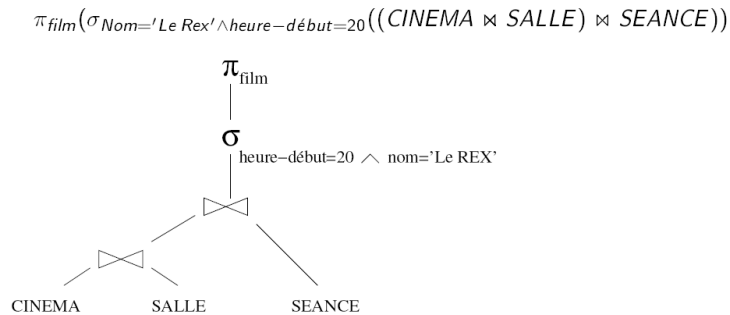
Règles de réécriture

- **Commutativité de la sélection et de la jointure.**
 $\sigma_{A=a'}(R(\dots A \dots) \bowtie S) \equiv \sigma_{A=a'}(R) \bowtie S$
- **Distributivité de la sélection sur l'union.**
 $\sigma_{A=a'}(R \cup S) \equiv \sigma_{A=a'}(R) \cup \sigma_{A=a'}(S)$
 NB : valable aussi pour la différence.
- **Commutativité de la projection et de la jointure**
 $\pi_{A_1 \dots A_p B_1 \dots B_q}(R \bowtie_{A_i=B_j} S) \equiv$
 $\pi_{A_1 \dots A_p}(R) \bowtie_{A_i=B_j} \pi_{B_1 \dots B_q}(S),$
 $(i \in \{1, \dots, p\}, j \in \{1, \dots, q\})$
- **Distributivité de la projection sur l'union**
 $\pi_{A_1 A_2 \dots A_p}(R \cup S) \equiv \pi_{A_1 A_2 \dots A_p}(R) \cup \pi_{A_1 A_2 \dots A_p}(S)$

Exemple d'algorithme de restructuration

- Séparer les sélections avec plusieurs prédicats en plusieurs sélections à un prédicat (règle 3)
- Descendre les sélections le plus bas possible dans l'arbre (règles 4, 5, 6)
- Regrouper les sélections sur une même relation (règle 3)
- Descendre les projections le plus bas possible (règles 7 et 8)
- Regrouper les projections sur une même relation

Arbre de requête avant restructuration

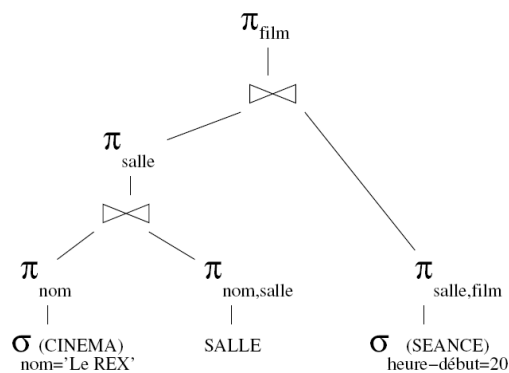


Arbre de requête après restructuration

L'idée : réduire le plus tôt possible (en bas de l'arbre) la taille des relations manipulées

→

1. On effectue les sélections, (opérateur le plus réducteur)
2. On élimine dès que possible les attributs inutiles par projection
3. On effectue les jointures




Le plan obtenu n'est pas TOUJOURS optimal
On pourrait trouver des contre exemples



Bilan – traduction algébrique

- La réécriture algébrique est nécessaire mais pas suffisante
 - L'optimiseur tient également compte
 - Des chemins d'accès aux données (dépendent des organisations de fichiers - index)
 - Des différents algorithmes implantant une même opération algébrique (sélection, projection, jointure)
 - De propriétés statistiques de la base
- Un balayage séquentiel peut être préférable à un parcours d'index



Evaluation des opérateurs relationnels – Optimisation physique

- On a le choix entre plusieurs algorithmes pour effectuer une opération
- Etude de quelques méthodes d'évaluation
 - De sélection
 - De jointure
 - De projection



Évaluation d'une sélection

- **Simple boucle – parcours séquentiel**
- **Indexée**
 - Utilisation d'un index primaire pour retrouver un *n-uplet dont la clé est donnée ou appartient à un intervalle donné*
 - Utilisation d'un index secondaire pour retrouver un ensemble de *n-uplets dont la clé est donnée ou appartient à un intervalle donné*




Évaluation d'une jointure

- **Equi-jointure**
 - **Pas d'index** sur les constituants de jointure
 - **Boucles imbriquées**
 - **Index** sur l'un des constituants de jointure
 - **Indexée**



Évaluation d'une projection

- **Simple boucle**



Pour l'étude de quelques unes de ces méthodes

On supposera que chaque relation est stockée dans un fichier qui contient les n -uplets de cette relation.

Le coût d'une opération sera mesuré en nombre de transferts de pages entre disque et mémoire centrale.

Ce coût peut se décomposer en deux parties :

- un **coût de production** de la relation résultat,
- un **coût d'écriture** de cette relation.

Le coût de production dépend de la méthode choisie pour réaliser l'opération alors que le coût d'écriture en est indépendant.

Dans le cas d'une évaluation pipeline d'une suite d'opérations :

op_1, \dots, op_n

les résultats des opérations op_1, \dots, op_{n-1} ne sont pas stockés sur disque car les n -uplets produits sont directement transmis à l'opérateur suivant :

- les coûts de lecture d'une relation opérande ou d'écriture des résultats d'un opérateur peuvent donc être nuls.

Coût de production d'une opération

- Il dépend de la méthode utilisée et des paramètres d'implantation des relations opérandes

$card(R)$	cardinalité de la relation R
$nb(R)$	nombre de blocs occupés par le fichier R
$nbatt(R)$	nombre d'attributs de la relation R
$nv(R, X)$	nombre de valeurs différentes du constituant X de la relation R

Coût d'écriture du résultat

- Indépendant de la méthode utilisée
- Ecriture réalisée au travers d'un tampon
 - Une case C du tampon est affectée à la relation résultat initialisée avec une page vide P
 - Pour chaque n -uplet t produit
 - Si la page P est pleine, l'écrire sur un bloc du disque
 - Placer une nouvelle page vide P dans la case C
 - Ecrire t dans P
- Le coût d'écriture du résultat est donc égal à $nb(\text{relation résultat})$

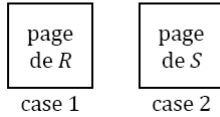
Sélection par simple boucle

$$S := \sigma_{cond}(R)$$

Coût de production

$$nb(R)$$

Tampon



Algorithme

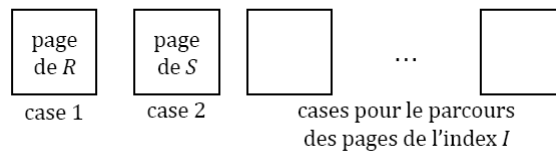
Pour chaque bloc du fichier *R* **faire**
 Transférer la page contenue dans ce bloc, dans la case 1.
Pour chaque *n*-uplet *t* de cette page **faire**
 si *cond(t)* **alors** écrire *t* dans la page de la case 2
 ou le transmettre à l'opérateur suivant.

Sélection indexée

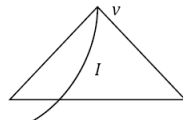
$$S := \sigma_{A=v}(R)$$

A est un attribut de *R*
 Il existe un index *I* sur l'attribut *A* de *R*

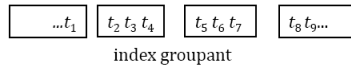
Tampon



Sélection indexée



I = index de l'attribut A de la relation R



Cas général - coût en moyenne – sélectivité haute
Valeurs uniformément réparties

$nb(R)$
$nv(R,A)$

Coût de production
(négligeant le parcours de l'index)

Comparaison

Soit une relation `livre` telle que :

- $card(livre) = 12000$
- $nb(livre) = 2400$
- $nv(livre, année) = 50$
- Il existe un index I sur l'attribut `année`

Le coût de production de l'opération :

- $\sigma_{année} = 2000(livre)$

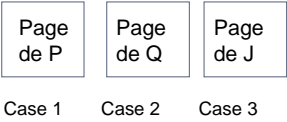
est égal à :

- sélection par boucle :
 - 2400
- sélection indexée :
 - $\lceil 2400/50 \rceil = 48,$



Jointure par boucles imbriquées

$J := P \text{ join}_{cond} Q$



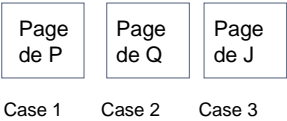
Pour chaque bloc de P
transférer la page contenue dans ce bloc dans la case 1
pour chaque bloc de Q
transférer la page contenue dans ce bloc dans la case 2
pour chaque nuplet t' de cette page faire
pour chaque nuplet t de la page de la case 1
si cond(t conc t') est vraie alors
 ecrire le nuplet t conc t' dans la page de la case 3
 ou le transmettre à l'opérateur suivant

$$nb(P) + nb(P)*nb(Q)$$



Jointure par boucles imbriquées

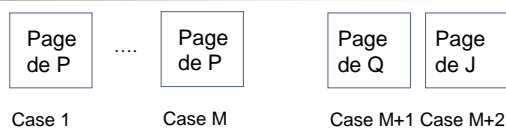
$J := P \text{ join}_{cond} Q$



$$nb(P) + nb(P)*nb(Q)$$

p ₁	nb(Q)
p ₂	nb(Q)
p ₃	nb(Q)
p ₄	nb(Q)
p ₅	nb(Q)
<hr/>	
nb(P)	nb(P)*nb(Q)

Jointure par boucles imbriquées : remarques

$$J := P \text{ join}_{cond} Q$$


Coût de production

$$nb(P) + \left\lceil \frac{nb(P)}{M} \right\rceil \times nb(Q)$$

Si $nb(P) \leq M$, c.-à-d. si la relation P tient en mémoire, le coût est égal à :

$$nb(P) + nb(Q)$$

Le coût est minimum si l'on choisit la relation de cardinalité minimum comme relation externe.

Jointure indexée

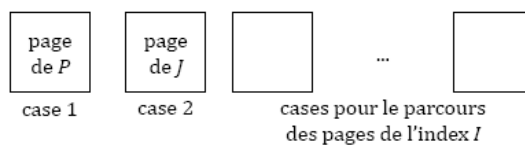
$$J := P \text{ join}_{A=B} Q$$

A est un attribut de P

B est un attribut de Q

Il existe un index I sur l'attribut B de Q

Tampon



Jointure indexée

Algorithme

```

pour chaque bloc de  $P$  faire
  Transférer dans la case 1 la page  $p$  contenue dans ce bloc.
  pour chaque  $n$ -uplet  $t$  de  $p$  faire
     $v = t.A$ 
    Effectuer la sélection indexée  $\sigma_{B=v}(Q)$  (index  $I$ )
    pour chaque  $n$ -uplet  $t'$  sélectionné par cette
    opération faire
      Ecrire le  $n$ -uplet  $t$  conc  $t'$  dans la page de la case 2
      ou le transmettre à l'opérateur suivant.
  
```

Coût de production

$$nb(P) + card(P) \times \left\lceil \frac{nb(Q)}{nv(Q, B)} \right\rceil \quad \text{si l'index } I \text{ est groupant}$$

Comparaison

Soit une relation livre (ISBN, nom_auteur) telle que :

- $card(livre) = 24000, nb(livre) = 2400,$
- $nv(livre, nom_auteur) = 6000,$
- Il existe un index $I \longrightarrow$ sur l'attribut nom_auteur.

et une relation auteur (nom, pays) telle que :

- $card(auteur) = 6000, nb(auteur) = 600$
- $nv(auteur, nom) = 6000$

Le coût de production de l'opération :

■ $auteur \text{ join}_{nom = nom_auteur} livre$
est égal à :

- jointure par boucles imbriquées :
 - $600 + 600 \times 2400$
- jointure indexée :
 - $600 + 6000 \times (2400/6000) = 3000$

Projection sans élimination des doublons

$$P := \pi_{A_1, \dots, A_k}(R)$$

Coût de production

$nb(R)$

Evaluation de la requête

- Le résultat de l'optimisation est un plan physique d'exécution autrement dit une séquence d'opérations à exécuter



Choix du plan d'exécution

- En théorie, le choix d'un plan d'exécution peut être effectué ainsi:
 - Traduire la requête sous forme d'un arbre de requête initial
 - Construire l'ensemble des arbres équivalents à l'arbre initial en appliquant les propriétés de commutativité et d'associativité des opérateurs
 - A partir de chacun de ces arbres, générer tous les plans d'exécution possibles en associant à chaque opérateur chacune des méthodes de résolution applicable
 - Évaluer le coût de chaque plan d'exécution
 - Choisir le plan de coût minimal



Choix du plan d'exécution

- Le nombre de plans d'exécution à examiner croît très vite avec le nombre de relations sur lesquelles porte la requête
- En pratique, on utilise des règles heuristiques pour diminuer le nombre de plans d'exécution à évaluer

Exemple

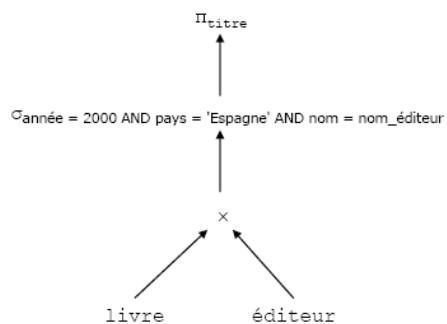
Schéma

- livre(isbn, titre, auteurs, nom_éditeur, année, prix)
- éditeur(nom, adresse, pays)

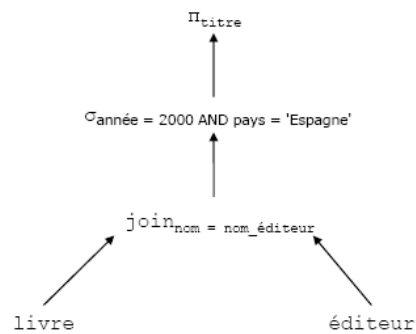
Requête

- *Titres des livres publiés en 2000 édités par un éditeur espagnol ?*
 - SELECT titre
 - FROM livre, éditeur
 - WHERE nom = nom_éditeur
 - AND pays = 'Espagne'
 - AND année = 2000;

Arbre de requête initial

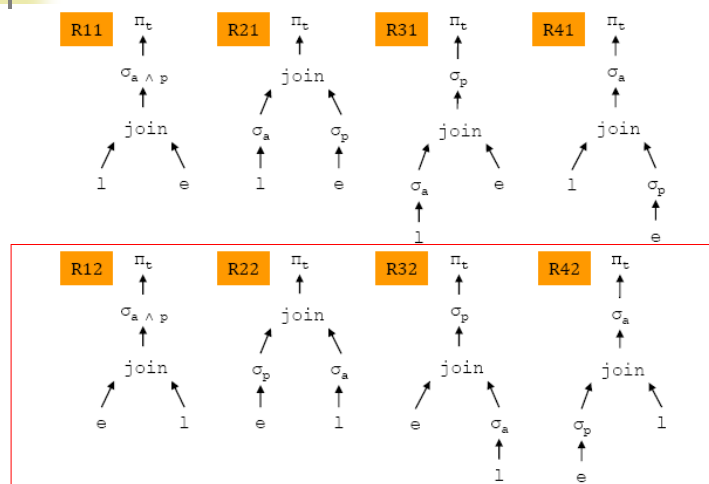


Avec jointure



l = livre
 e = éditeur
 a = année = 2000
 p = pays = 'Espagne'

Arbres de requêtes équivalents



Pourquoi ces plans ?

- Ils consistent à réaliser la jointure en choisissant comme relation externe la relation *éditeur* qui est beaucoup moins volumineuse que la relation *livre* et à traiter les relations en pipeline chaque fois que possible.

Tables

	<i>nbatt</i>	<i>card</i>	<i>nb</i>	<i>nv(titre)</i>	<i>nv(nom_éditeur)</i>	<i>nv(année)</i>
livre	6	12000	2400	12000	300	20

	<i>nbatt</i>	<i>card</i>	<i>nb</i>	<i>nv(nom)</i>	<i>nv(pays)</i>
éditeur	3	300	60	300	30

Autres paramètres

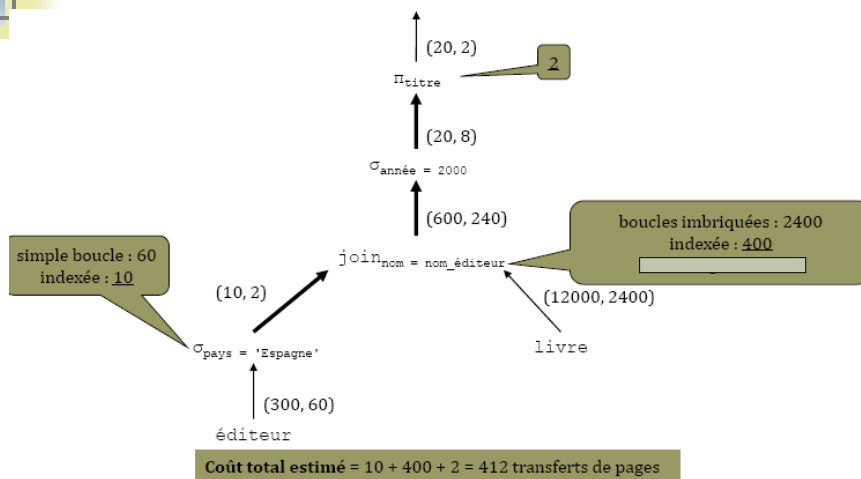
Index

livre(isbn)
livre(nom_éditeur)
livre(année)
éditeur(nom)
éditeur(pays)

Tampon
52 cases

L'étiquette (c, b) de chaque arête indique la cardinalité et le nombre de blocs pour la relation associée à cette arête

Le quatrième



1^{er} 4862 – 2nd 852 – 3^{ème} 1022 E/S