

# Cours

## “Bases de données”

### Optimisation

3° année (MIS)

Antoine Cornuéjols

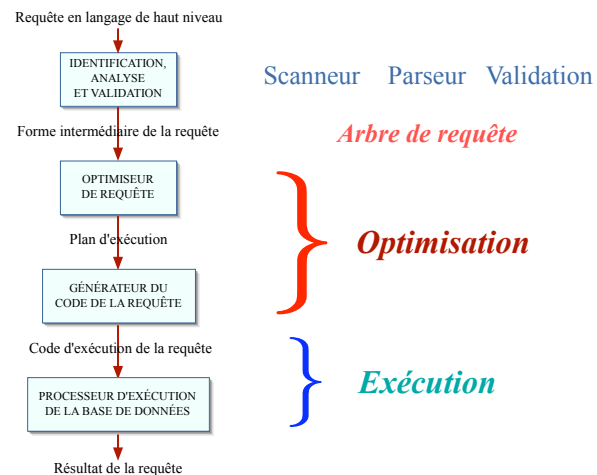
[www.lri.fr/~antoine](http://www.lri.fr/~antoine)  
[antoine.cornuejols@agroparistech.fr](mailto:antoine.cornuejols@agroparistech.fr)

## Méthodes d'optimisation des requêtes

1. **Introduction**
2. **Étude des coûts**
3. **Optimisation**
  - 3.1. Utilisation d'heuristiques
  - 3.2. Évaluation systématique
  - 3.3. Optimisation dans Oracle

## Traitement et optimisation des requêtes

### Étapes de traitement des requêtes



3

## Traitement et optimisation des requêtes

### Introduction

#### Langage de requête de bas niveau

- Le programmeur a la charge complète de l'optimisation des requêtes

#### Langage de requête de haut niveau

- Possède des systèmes dédiés à l'optimisation des requêtes
- Le programmeur peut souvent donner des "indications" ("hints")

4

## Traitement et optimisation des requêtes

### Introduction

#### Recherche du meilleur plan d'exécution

- Coût excessif

#### Recherche d'une solution approchée à un coût raisonnable

- Par heuristiques
- Par estimation approximative des coûts

*Un optimiseur de requête combine généralement ces deux techniques*

5

## Traitement et optimisation des requêtes

### Introduction

#### Deux grandes techniques d'optimisation

##### Utilisation de règles heuristiques

- Ré-ordonnent les opérations dans les arbres de requêtes

##### Évaluation systématique des coûts

- Choix du plan d'exécution le moins coûteux

*Un optimiseur de requête combine généralement ces deux techniques*

6

## Méthodes d'optimisation des requêtes

### 1. Introduction

### 2. Étude des coûts

### 3. Optimisation

- Utilisation d'heuristiques
- Évaluation systématique
- Optimisation dans Oracle

## Traitement et optimisation des requêtes

### Analyse des coûts

#### Opérations de tri

##### Algorithme le plus utilisé : le tri-fusion

- Phase de tri de petits sous-fichiers
- Fusion triée des sous-fichiers triés

##### Phase de tri

- Ex : Fichier de  $b=1024$  blocs ; espace tampon de  $n_B=5$  blocs  
->  $n_R=205$  runs initiaux (205 sous-fichiers triés)

##### Phase de fusion

- Ex : 4 sous-fichiers triés ->  $52 = 205/4$ , puis 13, puis 4, puis 1

Coût total :  $(2 \cdot b) + (2 \cdot (b \cdot (\log_{dM} n_R)))$

Nb d'accès de blocs (lect. et écriture)

Nb d'accès de blocs pour la phase de fusion

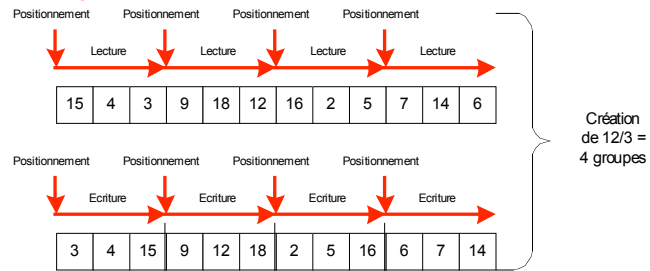
8

## Traitement et optimisation des requêtes

### Analyse des coûts : tri-fusion

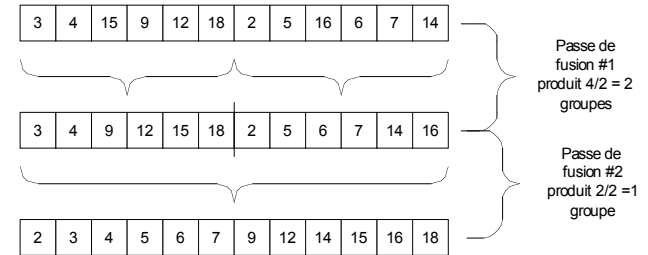
#### • Étape tri

- nombre de groupes =  $\lceil B_T / M \rceil = \lceil 12 / 3 \rceil = 4$
- Coût =  $2 * (\lceil B_T / M \rceil * TempsPosDébut + B_T * TempsTrans) = 104$  ms



## Traitement et optimisation des requêtes

### Analyse des coûts : tri-fusion



Coût des passes de fusion

$$= B_T * (2 * \log_{M-1} (B_T / M) - 1) * TempsESBloc$$

$$= 12 * (2 * \log_2 (12 / 3) - 1) * 11 \text{ ms} = 396 \text{ ms}$$

## Traitement et optimisation des requêtes

### Analyse des coûts : tri-fusion

$TempsES (TRI) =$

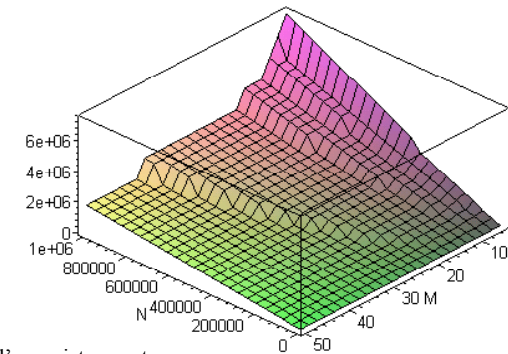
$$2 * (B_T / M * TempsPosDébut + B_T * TempsTrans) + B_T * (2 * \log_{M-1} (B_T / M) - 1) * TempsESBloc$$

$$= 104 \text{ ms} + 396 \text{ ms} = 500 \text{ ms}$$

## Traitement et optimisation des requêtes

### Analyse des coûts : tri-fusion

Estimation du coût  
d'un tri-fusion  
(FBM = 20)



N = nb d'enregistrements

M = Taille de la mémoire centrale  
disponible en nombre de blocs

## Traitement et optimisation des requêtes

### Analyse des coûts

#### Opérations de *SELECTION*

##### Responsabilité du *scanneur* de fichiers

- Scrute les enregistrements d'un fichier pour extraire ceux qui satisfont à une condition de sélection

##### Plusieurs méthodes de recherche

- *Test d'égalité* :
  - Si index : parcours d'index
  - Recherche linéaire (force brute)
  - Recherche binaire si attribut clé sur lequel le fichier est trié
  - Utilisation d'un index primaire ou d'une clé de hachage
  - Utilisation d'un index secondaire (arbre-B+)
  - ...
- *Sélections complexes* (conjonction, disjonction, ...)

## Traitement et optimisation des requêtes

### Analyse des coûts

#### Opérations de *JOINTURE*

- Opération très coûteuse

##### Plusieurs méthodes de jointure

- Utilisation de boucles imbriquées (force brute)
- Une seule boucle si index ou clé de hachage sur l'autre attribut
- Par tri-fusion préalable sur les deux attributs
- Par jointure-hachage et variantes

##### Grande influence de l'espace tampon disponible

## Traitement et optimisation des requêtes

### Analyse des coûts

#### Opérations de *PROJECTION* et opérations sur les ensembles

##### Projection

- Simple si la liste des attributs de projection inclut une clé.
- Sinon, problème des doublons

##### Produit cartésien

- Potentiellement très coûteux. À éviter.

##### Union / Intersection / Différence

- Même ensemble d'attributs sur chaque table
- Souvent des techniques par tri-fusion

## Traitement et optimisation des requêtes

### Analyse des coûts

#### Traitement par *flot* ou par *pipe-line*

##### Il faut éviter les lourds stockages intermédiaires

##### Traitement par *flot*

- On utilise le résultat partiel d'une opération immédiatement dans les opérations qui suivent

## Méthodes d'optimisation des requêtes

1. **Introduction**
2. **Étude des coûts**
3. **Optimisation**
  - 3.1. Utilisation d'heuristiques
  - 3.2. Évaluation systématique
  - 3.3. Optimisation dans Oracle

## Traitement et optimisation des requêtes

### Utilisation d'heuristiques

#### Arbre de requête ("query tree")

- structure de données arborescente
- visualise graphiquement une requête relationnelle traduite en une expression équivalente de l'algèbre relationnelle
  - **Feuilles** : tables utilisées dans la requête
  - **Noeuds intermédiaires** : opérations algébriques
  - **Noeud racine** : dernière opération algébrique avant le retour du résultat

L'opération d'un noeud intermédiaire est déclenchée quand ses opérandes sont disponibles.

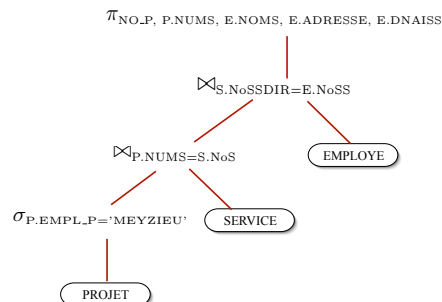
On remplace alors le noeud par la relation produite par l'opération.

## Traitement et optimisation des requêtes

### Utilisation d'heuristiques

#### Arbre de requête

$\pi_{NO.P, NUMS, NOMS, ADRESSE, DNAISS}(((\sigma_{EMPL.P='MEYZIEU'}(PROJET)) \bowtie_{NUMS=NoS} (SERVICE)) \bowtie_{NoSSDIR=NoSS} (EMPLOYE))$



## Traitement et optimisation des requêtes

### Utilisation d'heuristiques

#### Arbre de requête

#### Première étape du processus de traduction et d'exécution

- Vérification de l'existence des tables
- Vérification de l'existence des attributs
- Vérification des droits d'accès de l'utilisateur

On va ensuite optimiser l'arbre de requête

## Traitement et optimisation des requêtes

### Utilisation d'heuristiques

#### Optimisation de l'arbre de requête

Recherche d'expressions algébriques équivalentes

#### Optimisation par :

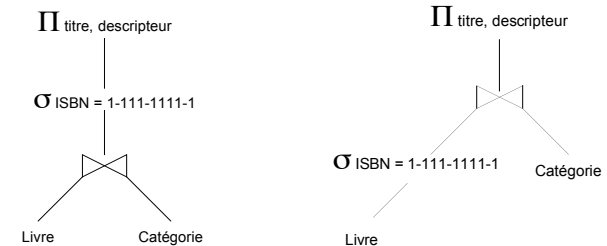
- Utilisation de la commutativité de certains opérateurs
- Conservation des seuls attributs nécessaires aux opérations ultérieures

#### Utilisation d'un ensemble de règles de transformation par l'optimiseur

## Traitement et optimisation des requêtes

### Plans d'exécution équivalents

#### Plusieurs arbres algébriques équivalents



## Traitement et optimisation des requêtes

### Plans d'exécution équivalents

#### Règles d'équivalence de l'algèbre relationnelle

- Eclatement d'une sélection conjonctive (SE)
  - $\sigma_{e_1 \wedge e_2}(T) = \sigma_{e_1}(\sigma_{e_2}(T))$
- Commutativité de la sélection (SC)
  - $\sigma_{e_1}(\sigma_{e_2}(T)) = \sigma_{e_2}(\sigma_{e_1}(T))$
- Elimination des projections en cascades (PE)
  - $\pi_{liste1}(\pi_{liste2}(T)) = \pi_{liste1}(T)$
- Commutativité de la jointure (JC)
  - $T_1 \mid T_2 = T_2 \mid T_1$
- Associativité de la jointure (JA)
  - $T_1 \mid (T_2 \mid T_3) = (T_1 \mid T_2) \mid T_3$

## Traitement et optimisation des requêtes

### Plans d'exécution équivalents

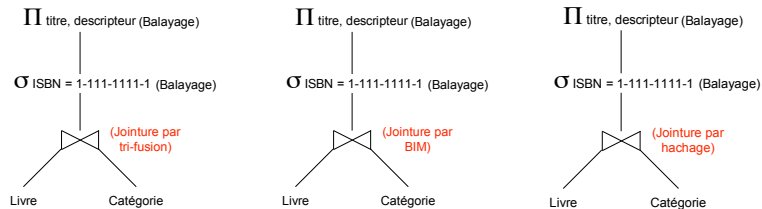
#### Règles d'équivalence de l'algèbre relationnelle (suite)

- Commutativité restreinte de la sélection et de la jointure (CSJ)
  - $\sigma_e(T_1 \mid T_2) = \sigma_e(T_1) \mid T_2$ 
    - si l'expression de sélection  $e$  ne contient que des colonnes de  $T_1$
- Commutativité restreinte de la projection et de la sélection (CPS)
  - $\pi_{listeColonnes}(\sigma_e(T)) = \pi_{listeColonnes}(\sigma_e(\pi_{listeColonnes \cup \text{colonnes de } e}(T)))$
- Commutativité restreinte de la projection et de la jointure (CPJ)
  - $\pi_{listeColonnes}(T_1 \mid T_2) =$ 
    - $\pi_{listeColonnes}(\pi_{listeColonnes \cap \text{colonnes de } T_1}(T_1) \mid \pi_{listeColonnes \cap \text{colonnes de } T_2}(T_2))$
- etc.

## Traitement et optimisation des requêtes

### Plans d'exécution équivalents

*Pour chaque opération logique : plusieurs choix d'opérations physiques*



## Traitement et optimisation des requêtes

### Utilisation d'heuristiques

#### Optimisation de l'arbre de requête

##### En règle générale :

- ☞ placer les opérateurs de projection et de sélection le plus près possible des "feuilles"
- ☞ **Fusionner plusieurs sélections sur une même table en une seule** afin que le prédicat de sélection ne soit testé qu'une seule fois
- ☞ **Effectuer les sélections le plus tôt possible** afin de réduire la taille des tables intermédiaires (le plus près des feuilles)
- ☞ **Effectuer les projections le plus tôt possible**, mais jamais avant les sélections (réduisent le nombre de colonnes et, souvent, le nombre de tuples)
- ☞ **Placer les opérateurs de jointure le plus près possible du noeud racine** à cause du coût de leur évaluation (il devrait y avoir moins de tuples et de colonnes près de la racine).

## Méthodes d'optimisation des requêtes

1. **Introduction**
2. **Étude des coûts**
3. **Optimisation**
  - 3.1. Utilisation d'heuristiques
  - 3.2. Évaluation systématique
  - 3.3. Optimisation dans Oracle

## Traitement et optimisation des requêtes

### Optimisation par évaluation des coûts

#### Complète l'utilisation d'heuristiques

**Il faut évaluer le plus précisément des stratégies alternatives tout en limitant le coût de l'optimisation elle-même.**

Surtout utilisée dans les requêtes compilées

##### Une évaluation des coûts est conservée dans le catalogue du SGBD

- ☞ Coût d'accès à un type de stockage secondaire (souvent le plus important)
- ☞ Coût de calcul
- ☞ Coût d'usage de la mémoire
- ☞ Coût des communications

## Traitement et optimisation des requêtes

### Optimisation par évaluation des coûts

Une évaluation des coûts est conservée dans le catalogue du SGBD

#### Exemple d'informations stockées :

- Nombre d'enregistrements (tuples) d'un fichier
- Taille moyenne des enregistrements
- Nombre de blocs
- Facteur de blocage du fichier (*nb max d'enregistrements par bloc*)
- Méthode d'accès primaire
- Attributs d'accès primaire de chaque fichier
- Informations sur tous les index secondaires et attributs d'indexation
- Nombre de blocs de l'index de plus haut niveau
- ...

## Traitement et optimisation des requêtes

### Optimisation par évaluation des coûts

Une évaluation des coûts est conservée dans le catalogue du SGBD

Certaines valeurs ont besoin d'être remises à jour fréquemment :

- E.g. nombre d'enregistrements d'un fichier

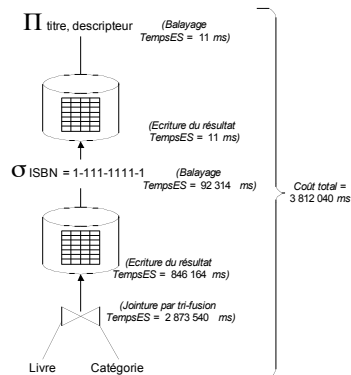
d'autres, non

- E.g. nombre de niveaux d'un index hiérarchique

## Traitement et optimisation des requêtes

### Optimisation par évaluation des coûts

#### Exemple d'utilisation des fonctions de coût

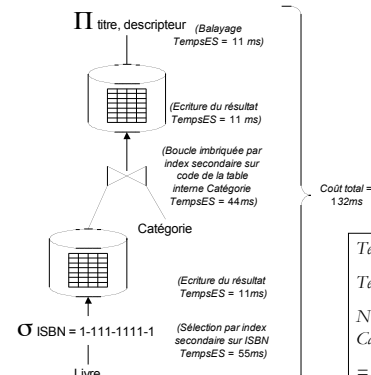


$$\begin{aligned} \text{TempsES}(\text{Plan avec pipeline}) &= \\ \text{TempsES}(JTF_{\text{Livres}} \text{ Catégorie}) &= 2 \\ 873\,540 \text{ ms} \end{aligned}$$

## Traitement et optimisation des requêtes

### Optimisation par évaluation des coûts

#### Exemple 2 d'utilisation des fonctions de coût



$$\begin{aligned} \text{TempsES}(\text{Plan avec pipeline}) &= \\ \text{TempsES}(S=IS \text{ pour index sur ISBN}) &+ \\ N_{\sigma \text{ ISBN}=1-11-111-1111} \cdot \text{TempsES}(S=IS \text{ sur code de } & \\ \text{Catégorie}) & \\ = 55 \text{ ms} + 33 \text{ ms} &= 88 \text{ ms} \end{aligned}$$

## Traitement et optimisation des requêtes

### Optimisation par évaluation des coûts

#### Estimation de la taille du résultat d'une opération

- Taille d'une sélection
  - $\lceil Sel_T(\text{Expression de sélection}) / FBM_T \rceil$  blocs
- Taille d'une projection
  - $N_{\pi \text{ liste}(T)} = N_T$  si contient une clé candidate
  - $N_{\pi C(T)} = Card_T(C)$  pour une colonne
  - $N_{\pi C_1, C_2, \dots, C_n(T)} = k(1 - (1/k)^{N_T})$ 
    - $k = \prod_{i=1, \dots, n} Card_T(C_i)$
- Taille d'une jointure naturelle
  - $N_{R_1 \Join S} = N_R * N_S / \text{maximum}(Card_R(\text{clé jointure}), Card_S(\text{clé jointure}))$

## Traitement et optimisation des requêtes

### Optimisation par évaluation des coûts

#### Exemple d'estimation de la taille d'une projection

```
SELECT DISTINCT code, annéeParution FROM Livre ;
```

$N_{Livre}$	1 000 000
$FBM_{Livre}$	20
$Card_{Livre}(code)$	4 000
$Card_{Livre}(annéeParution)$	50

- $k = \prod_{i=1, \dots, n} Card_T(C_i)$
- $= Card_{Livre}(code) * Card_{Livre}(annéeParution)$
- $= 4 000 * 50 = 200 000$
- $N_{\pi \text{ code, annéeParution}(T)} = k(1 - (1/k)^N)$
- $= 198 652$  lignes (en arrondissant)

## Méthodes d'optimisation des requêtes

1. Introduction
2. Étude des coûts
3. **Optimisation**
  - 3.1. Utilisation d'heuristiques
  - 3.2. Évaluation systématique
  - 3.3. Optimisation dans Oracle

## Traitement et optimisation des requêtes

### Optimisation dans Oracle

#### Oracle propose deux approches différentes

- L'une basée sur des règles
- L'autre sur les coûts

#### Approche basée sur des règles

- Opérations classées selon une heuristique
- Considère 15 types d'accès différents (depuis force brute à ROWID)

#### Approche basée sur les coûts

- Calcule le coût des stratégies en fonction des besoins en E/S, temps processeur, mémoire nécessaire

#### Possibilité d'intervention du développeur d'application

- Peut fournir des "hints" (si il a des informations que n'a pas le SGBD)

## Traitement et optimisation des requêtes

### Optimisation dans Oracle

**Oracle :**  
**mode RULE**

Rang	Chemin d'accès
1	Sélection par ROWID
2	Sélection d'une ligne par jointure dans une organisation par index groupant ou hachage hétérogène (CLUSTER)
3	Sélection d'une ligne par hachage sur clé candidate (PRIMARY ou UNIQUE)
4	Sélection d'une ligne par clé candidate
5	Jointure par une organisation par index groupant ou hachage hétérogène (CLUSTER)
6	Sélection par égalité sur clé de hachage (HASH CLUSTER)
7	Sélection par égalité sur clé d'index groupant (CLUSTER)
8	Sélection par égalité sur clé composée
9	Sélection par égalité sur clé simple d'index secondaire
10	Sélection par intervalle borné sur clé indexée
11	Sélection par intervalle non borné sur clé indexée
12	Tri-fusion
13	MAX ou MIN d'une colonne indexée
14	ORDER BY sur colonne indexée
15	Balayage

## Traitement et optimisation des requêtes

### Optimisation dans Oracle

#### ■ Cas Oracle

– outils

■ EXPLAIN PLAN

■ SQL Trace

■ SQL Analyse (*Enterprise Manager Tuning Pack*)

■ Oracle EXPERT

## Traitement et optimisation des requêtes

### Exercices