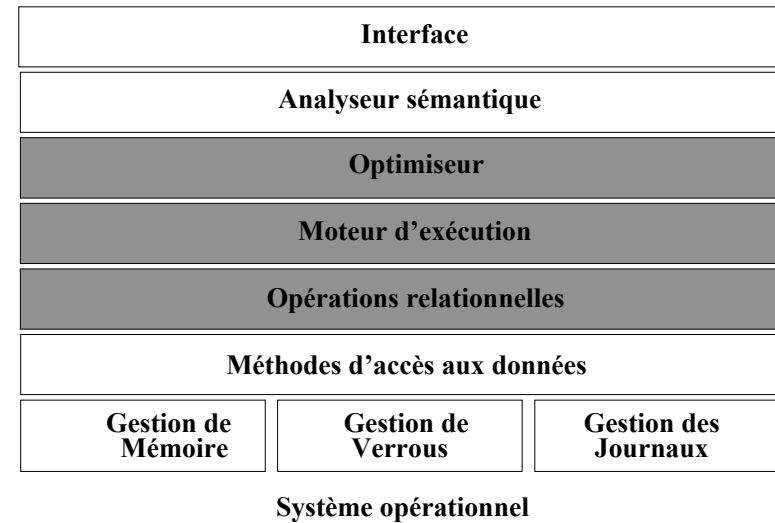


# OPTIMISATION DE QUESTIONS

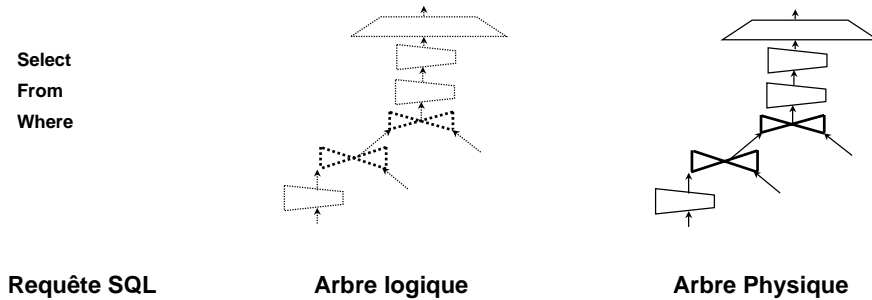
Luc Bouganim

Luc.Bouganim@prism.uvsq.fr

## Architecture en couche d'un SGBD



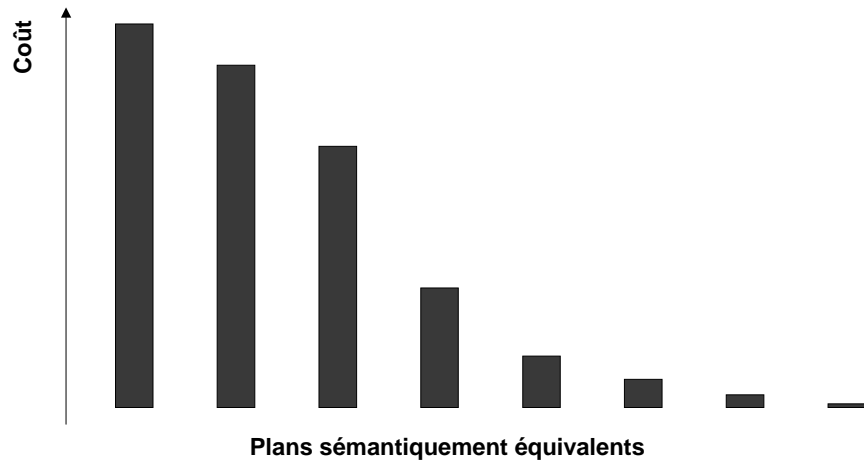
## Optimisation : ???



## Le problème : équivalence sémantique

- Une question → [Small grid]
- Plusieurs expressions équivalentes en SQL → [Medium grid]
- Plusieurs expressions équivalentes en algèbre → [Large grid]
- Plusieurs algorithmes algébriques équivalents → [Very large grid]

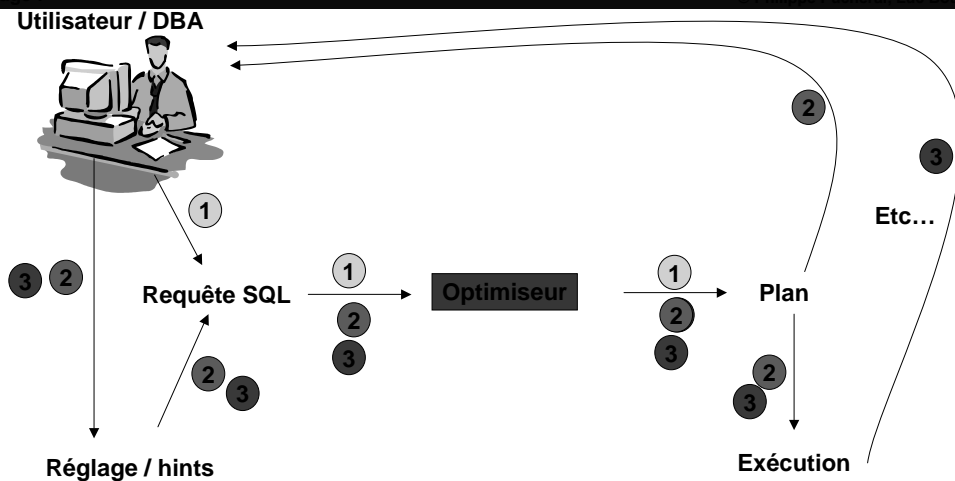
## Le Problème : différence de performances



## Les acteurs de l'optimisation

- **Idéalement :**
  - 2 requêtes équivalentes en SQL (langage déclaratif) doivent, après l'optimisation, produire le même arbre algébrique !
  - Qui plus est, cet arbre doit être le meilleur !
  - Seuls les concepteurs de SGBD (noyau) doivent comprendre l'optimisation et l'exécution
- **Dans la pratique :**
  - 2 requêtes équivalentes ne donnent pas toujours le même plan
  - Le plan n'est pas toujours le meilleur !
  - l'utilisateur (concepteur de l'appli) devra comprendre !!!

## Les acteurs de l'optimisation (suite)



## Objectifs de l'optimisation

**Trouver le meilleur plan d'exécution .... MEILLEUR ???**

- **Donnant les résultats le + vite ....**
  - Optimisation pour le temps de réponse (response time)
- **Minimisant la consommation de ressources**
  - Optimisation du travail total (Total work)
- **Minimisant le temps de délivrance des premiers tuples**
  - Optimisation de la latence (Latency / First tuples ...)

# ETAPES DE L'EVALUATION

1. CONTROLE SYNTAXIQUE ET SEMANTIQUE
2. SIMPLIFICATION DE LA REQUETE
3. DECOMPOSITION EN OPERATIONS ELEMENTAIRES ET CONSTRUCTION DES PLANS D'EXECUTION CANDIDATS
4. CALCUL DU COUT DE CHAQUE PLAN, CHOIX DU MEILLEUR ET EXECUTION

ETAPES 1,2 et 3 : INDEPENDANTES DES DONNEES

ETAPE 4: DEPENDANTE DES DONNEES

# Analyse de la question

## 1. ANALYSE SYNTAXIQUE

- VERIFIER LA SYNTAXE ET LA COHERENCE DE LA QUESTION PAR RAPPORT AU SCHEMA DE LA BD :

=> EXISTENCE DES RELATIONS ET DES ATTRIBUTS

## 2. ANALYSE SEMANTIQUE

- VERIFIER LA CORRECTION SEMANTIQUE DE LA QUESTION
- RECHERCHER DES QUESTIONS EQUIVALENTES PLUS SIMPLES

# Construction et choix du meilleur plan

LORSQUE LA REQUETE EST CORRECTE, LE BUT EST D'ASSOCIER UN PLAN D'EXECUTION A LA REQUETE, AFIN QUE LE COUT D'EXECUTION SOIT MINIMUM.

2 METHODES :

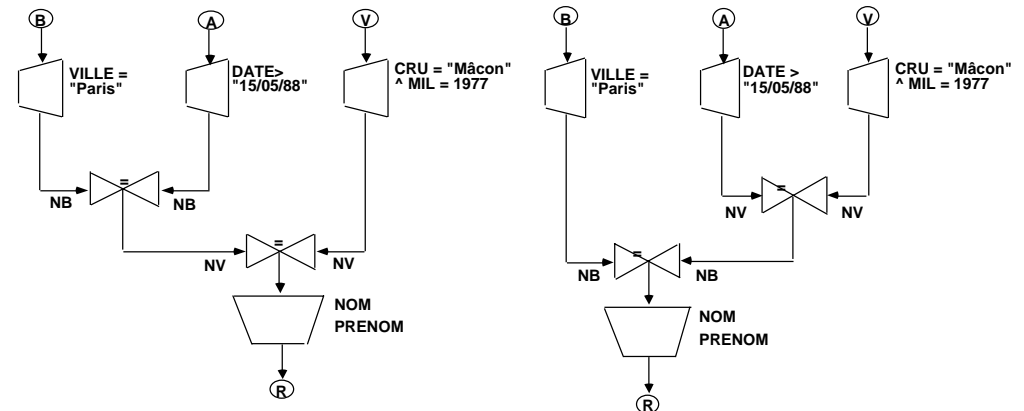
- LA RESTRUCTURATION ALGEBRIQUE
- L'EVALUATION DE PLANS

REMARQUE :

CES DEUX METHODES NE SONT PAS EXCLUSIVES

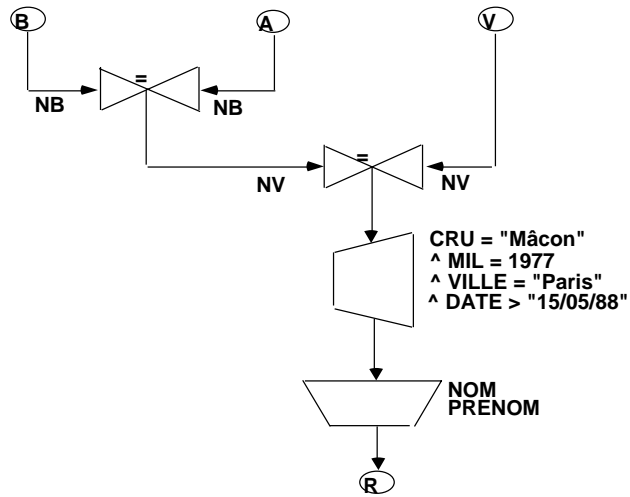
# Exemple (1)

"Donner les noms et les prénoms des buveurs habitant Paris qui ont bu du Mâcon 1977 après le 15/05/88"



Attention, les arbres se présentent généralement dans l'autre sens !

## Exemple (2)



De ces trois arbres, quel est le meilleur ??

## Restructuration algébrique

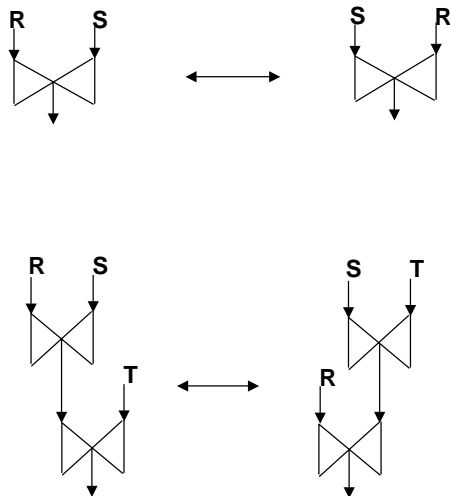
PROBLEME :

SUIVANT L'ORDRE DES OPERATEURS ALGEBRIQUES DANS UN ARBRE, LE COUT D'EXECUTION EST DIFFERENT

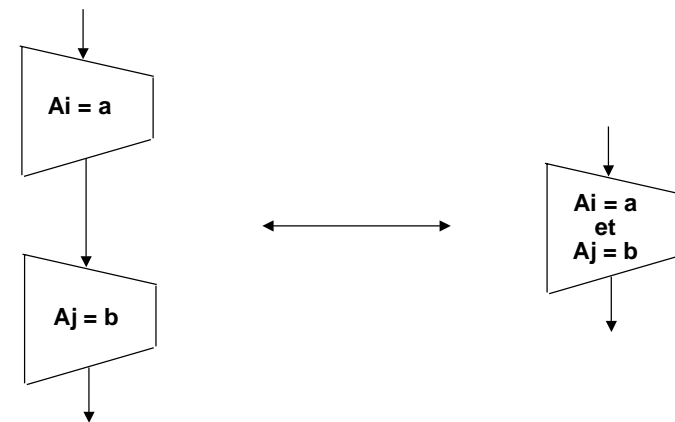
POURQUOI ?

1. LE COUT DES OPERATEURS VARIE EN FONCTION DU VOLUME DES DONNEES TRAITES i.e., PLUS LE NOMBRE DE TUPLE DES RELATIONS TRAITES EST PETIT, PLUS LES COUTS CPU ET D'E/S SONT MINIMISES
2. CERTAINS OPERATEURS DIMINUENT LE VOLUME DES DONNEES e.g., RESTRICTION ET PROJECTION

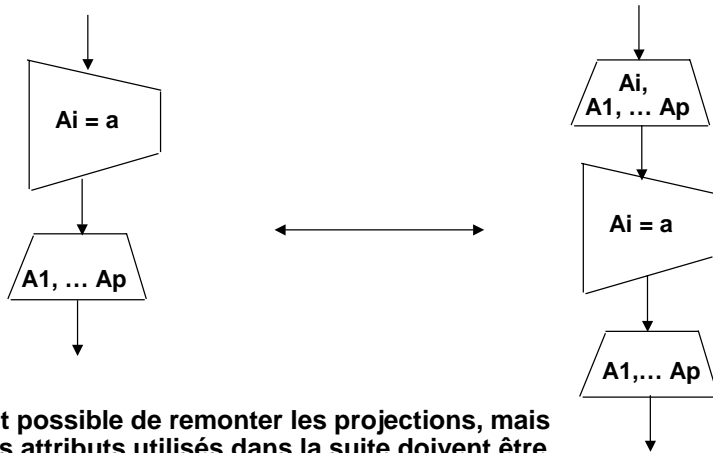
## Commutativité/Associativité des Jointures



## Groupage des Restrictions



## Semi-commutativité des Projections / ...



Il est possible de remonter les projections, mais les attributs utilisés dans la suite doivent être conservés !!!

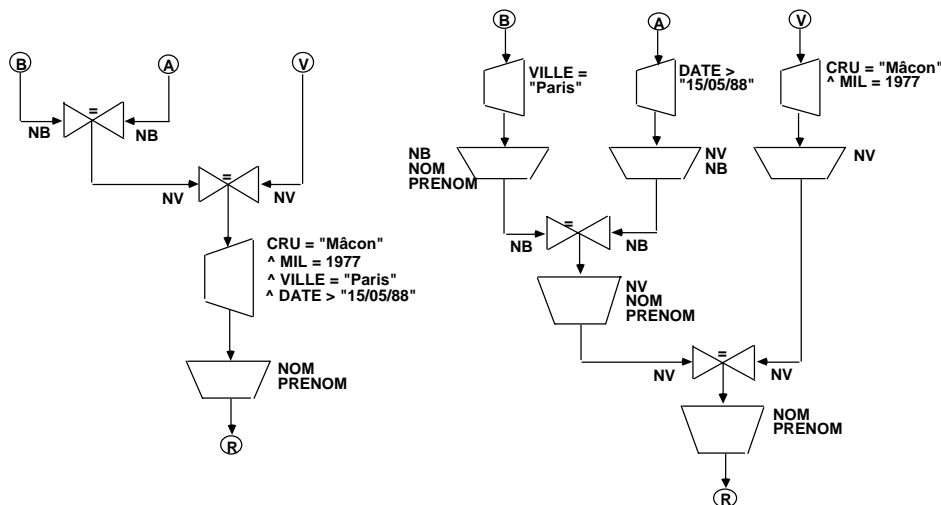
## Heuristique d'Optimisation

Appliquer d'abord les opérations réductrices (restrictions et projections) en les groupant sur chaque relation.

1. Dégrouper les restrictions
2. Rapprocher les restrictions des feuilles
3. Grouper les restrictions aux feuilles
4. Rapprocher les projections des feuilles

L'ordre des unions, différences et jointures reste inchangé !!!

## Exemple d'optimisation



## Problème de l'Ordonnancement

Il faut pouvoir ordonner jointures, union, différence, agrégat, ... en fonction des tailles des relations arguments

Il faut pouvoir prendre en compte les algorithmes par index afin de les favoriser (sélection, jointure sur index)

➔ Nécessité de développer un modèle de coût général permettant d'évaluer le coût d'un plan, c'est-à-dire d'un arbre annoté par des choix d'algorithmes.

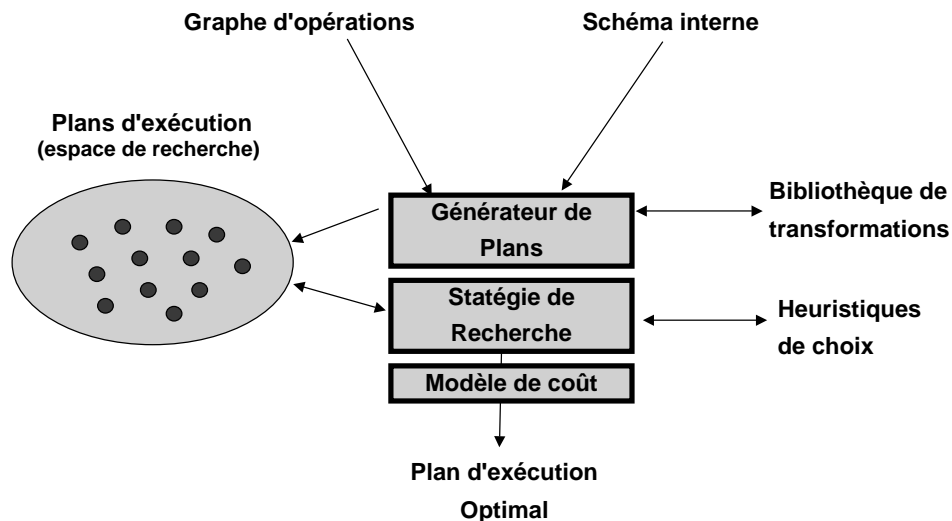
## Modèle de coût

- Paramètres d'entrée
  - machine (puissance, disques, mémoire, réseau, etc..)
  - Arbre d'exécution
  - Algorithmes relationnels
  - Schéma de la base
  - Statistiques sur les relations
  - Taille, Domaine, Nb valeurs distinctes, Répartition, Histogrammes
- Traitement
  - Evaluation de la taille des résultats intermédiaires
  - Evaluation du coût
- Sortie
  - Un coût en termes d'I/O, CPU, etc... ou un coût global

## Le calcul des tailles

- Taille des tables de base dans le catalogue
- Calcul des tailles à la compilation
  - application du coefficient de sélectivité
  - hypothèse d'uniformité
- Possibilité d'histogrammes
  - RunStat(<Table>, <attribut>)
  - Stockage dans le catalogue de l'histogramme de distribution de l'attribut
  - Utilisation par le modèle de coût

## CHOIX DU MEILLEUR PLAN



## Différentes Stratégies

