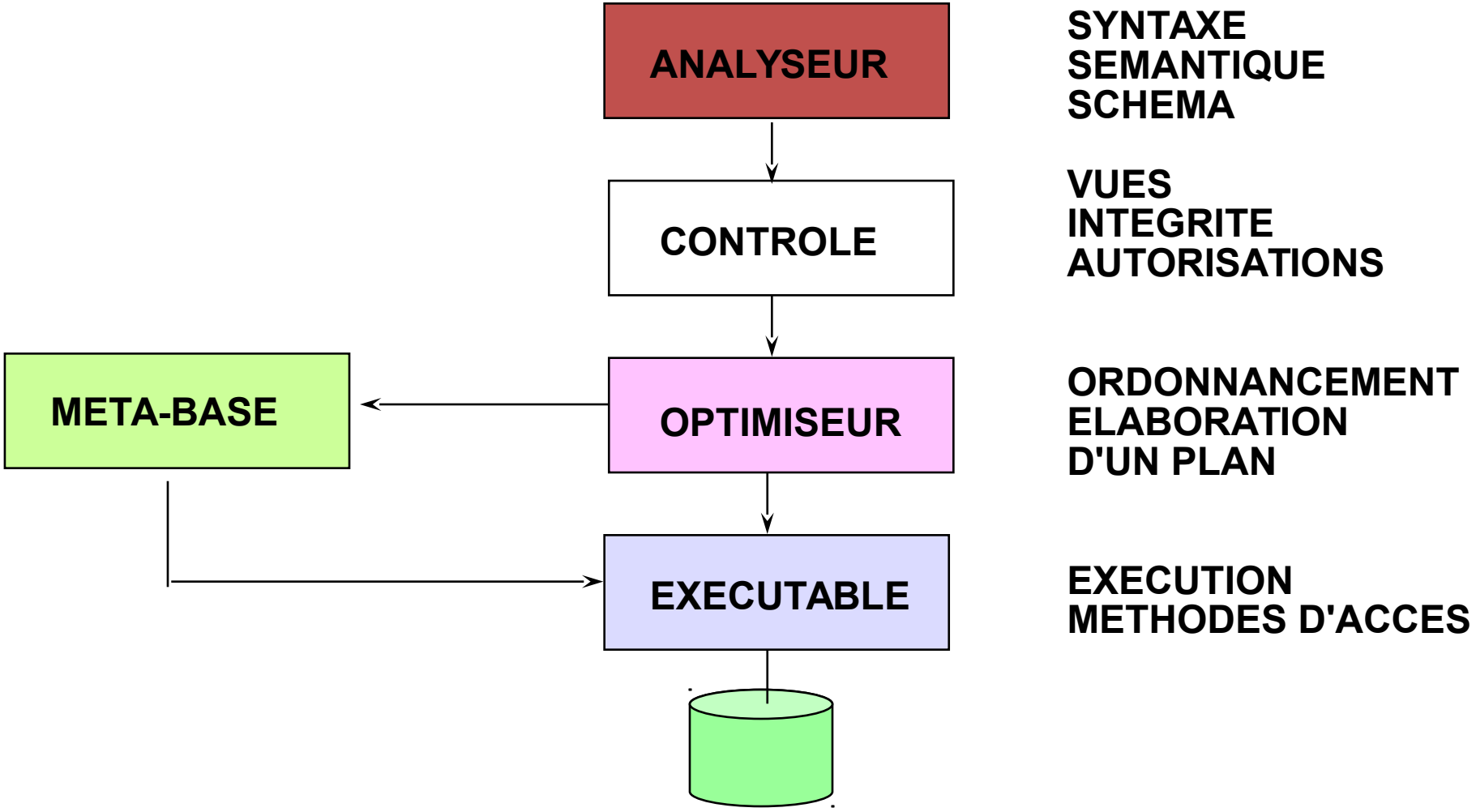
Optimisation de Requêtes

- **1. Introduction**
- **2. Arbres relationnels**
- **3. Restructuration algébrique**
- **4. Modèle de coût**
- **5. Choix du meilleur plan**
- **6. Conclusion**

1. ARCHITECTURE TYPE SGBD

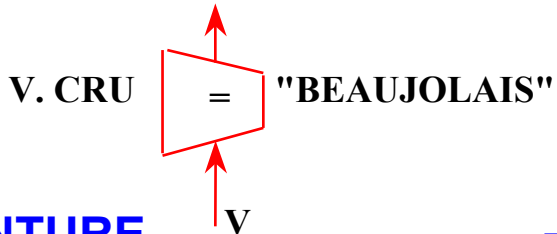


ETAPES DE L'OPTIMISATION

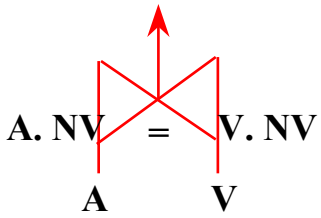
- **(1) Obtention d’une représentation canonique**
- **(2) Réécriture = transformation par :**
  - simplification
  - ordonnancement des opérations élémentaires
- **(3) Planning = construction des plans d'exécution candidats**
  - choix des algorithmes pour chaque opérateur,
  - calcul du coût de chaque plan,
  - choix du meilleur plan.
- **Etapes 1 et 2 : indépendantes des données**
- **Etape 3 : dépendante des données**

2. ARBRES RELATIONNELS

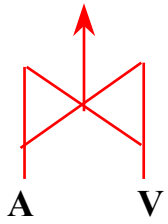
RESTRICTION



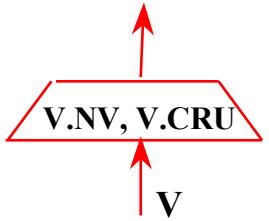
JOINTURE



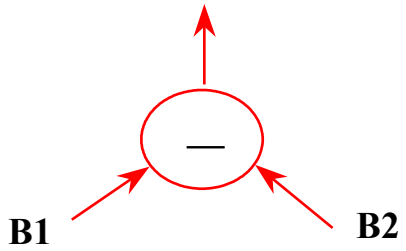
PRODUIT CARTESIEN



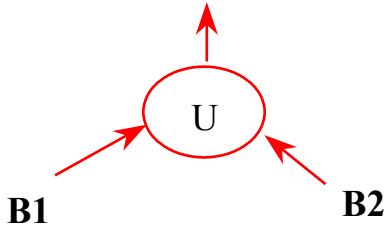
PROJECTION



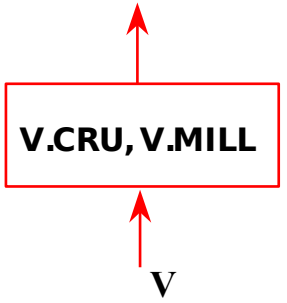
DIFFERENCE



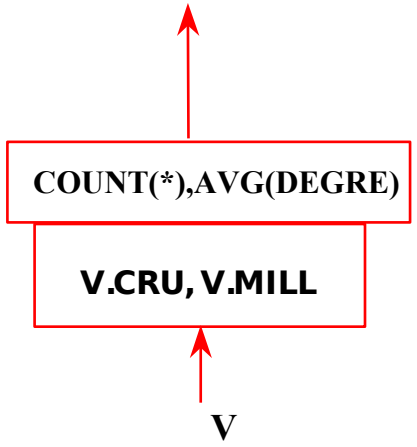
UNION



TRI

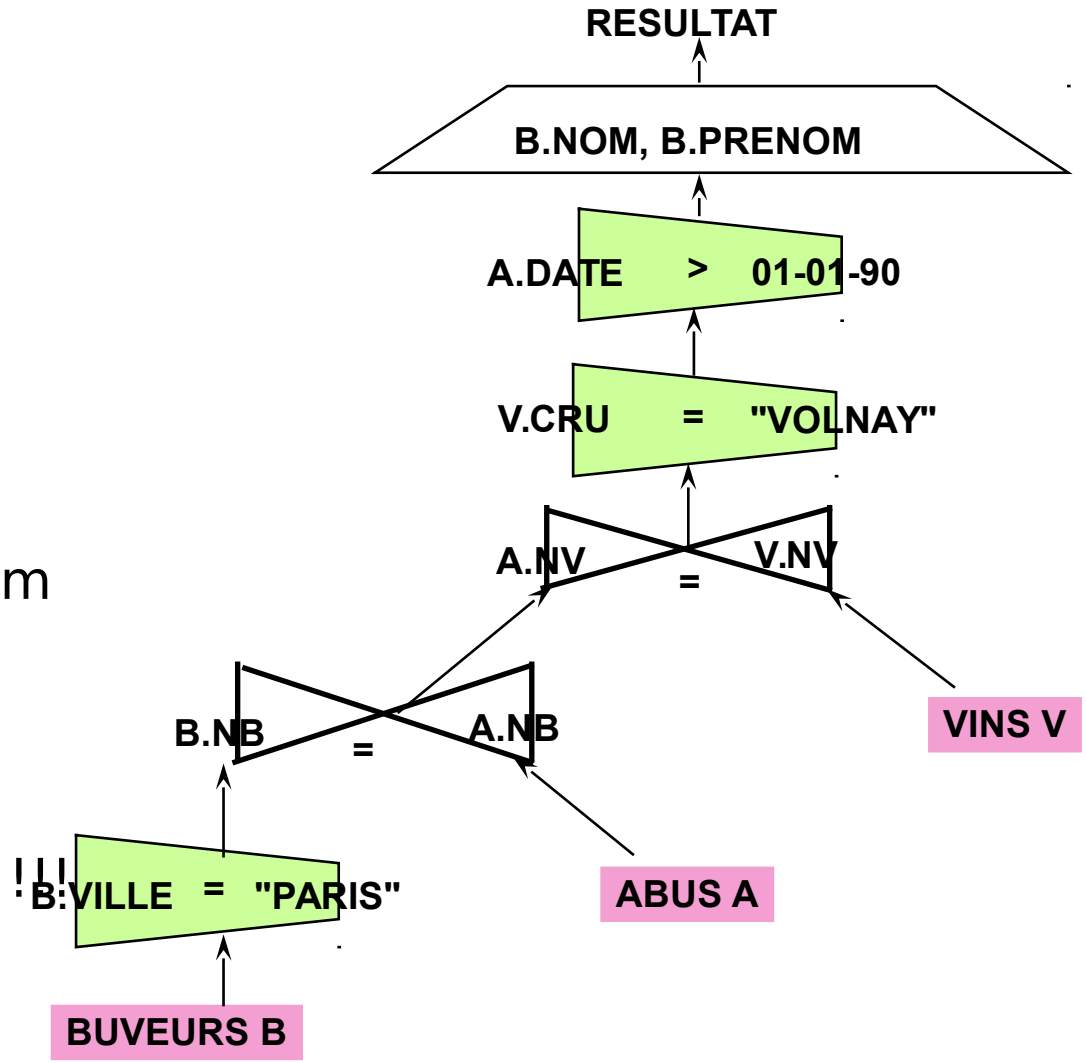


AGREGAT

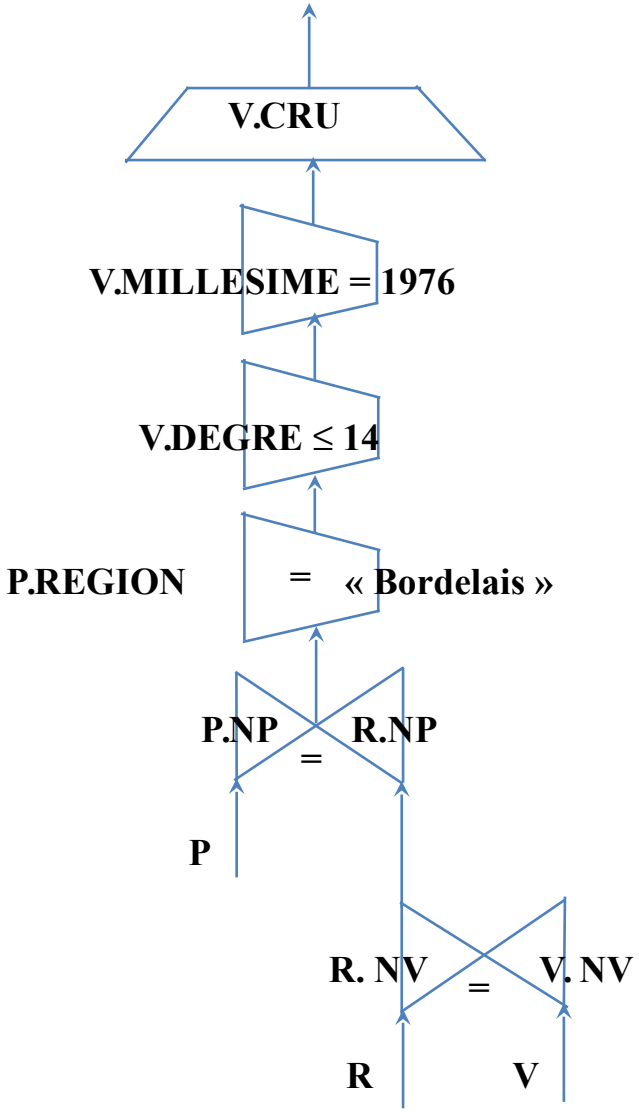


EXEMPLE D'ARBRE

- **Coût d'exécution:**
  - 10 millions de buveurs dont 1 m à Paris
  - 10 millions d'abus dont 10000 de Volnay
  - 1000 vins
  - 10 m + 10m * 1m + 10 m * 1000
  - + 10 m + 10000 + …
  - de l'ordre de 10 ** 13
  - comparaisons de tuples !!!

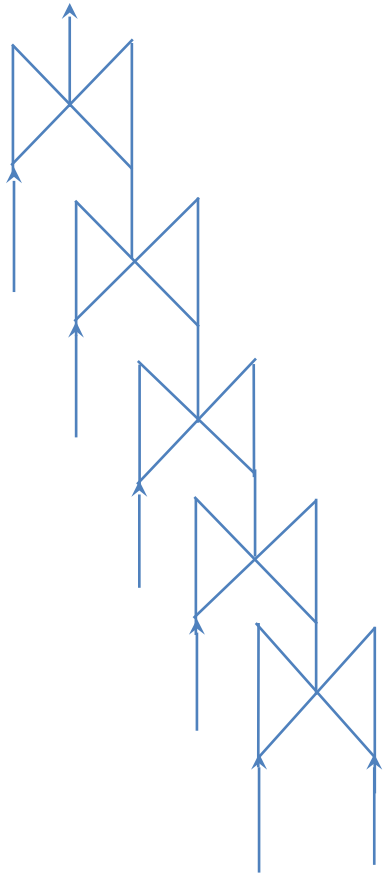


Arbre linéaire droit

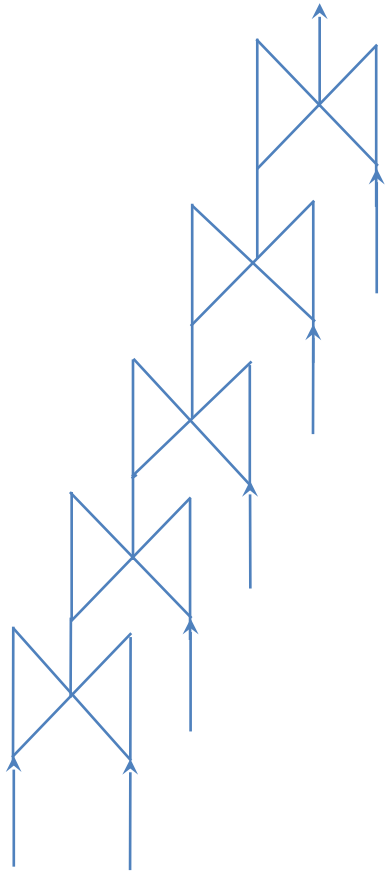



```
SELECT V.CRU
FROM PRODUCTEURS P, VINS V,
   PRODUIT R
WHERE V.MILLESIME = 1976 AND
   V.DEGRE ≤ 14
AND P.REGION = « BORDELAIS » AND
   P.NP = R.NP
AND R.NV = V.NV.
```

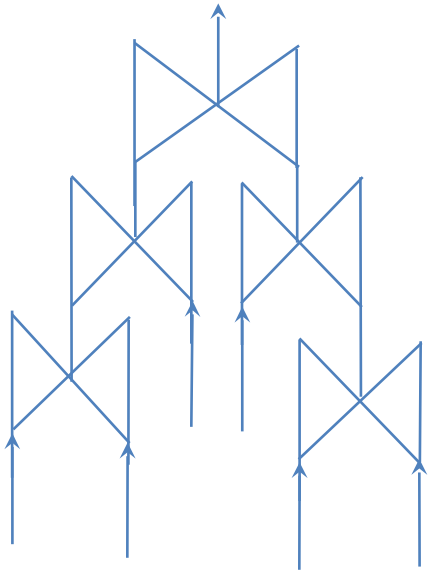
Typologie des arbres



Arbre linéaire droit

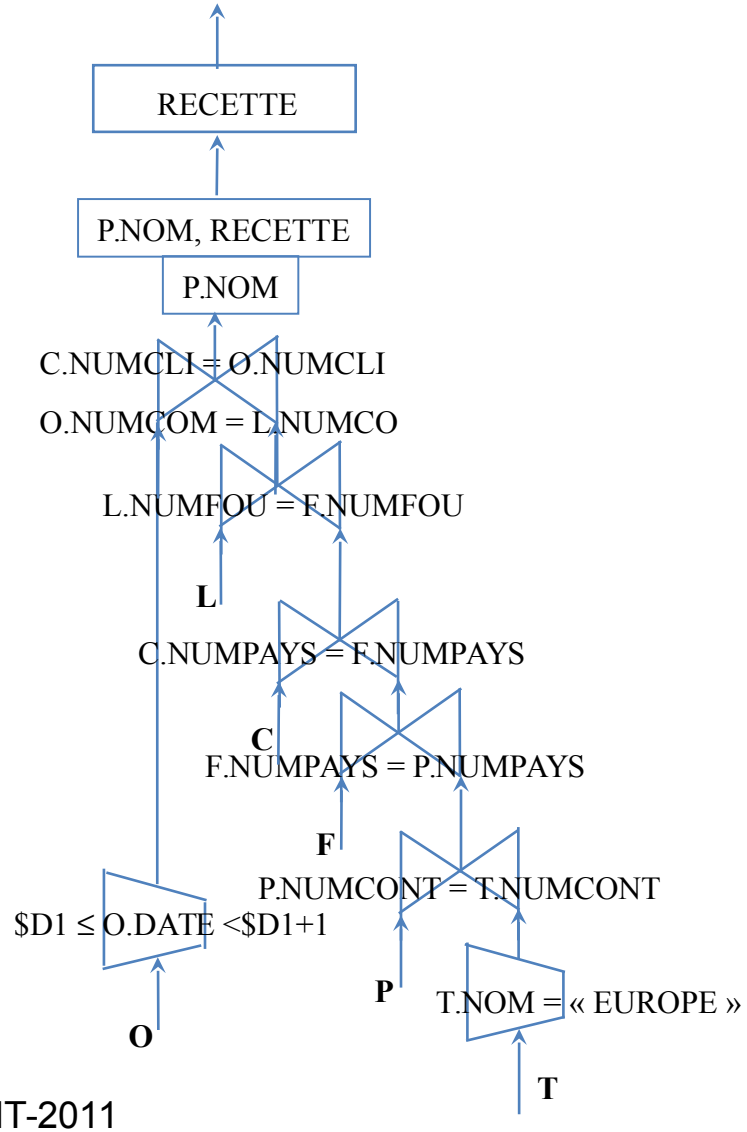


Arbre linéaire gauche



Arbre ramifié

Autre exemple




```
SELECT P.NOM, SUM(L.PRIX *
   (1-L.DISCOUNT))
FROM CLIENTS C, COMMANDES O, LIGNES L,
   FOURNISSEUR F, PAYS P, CONTINENTS T
WHERE    C.NUMCLI = O.NUMCLI
AND      O.NUMCOM = L.NUMCO
AND      L.NUMFOU = F.NUMFOU
AND      C.NUMPAYS = F.NUMPAYS
AND      F.NUMPAYS = P.NUMPAYS
AND      P.NUMCONT = T.NUMCONT
AND      T.NOM = « EUROPE »
AND      O.DATE ≥ $D1
AND      O.DATE < $D1 + INTERVAL 1 YEAR
GROUP BY P.NOM
ORDER BY RECETTE DESC ;
```

3. RESTRUCTURATION ALGEBRIQUE

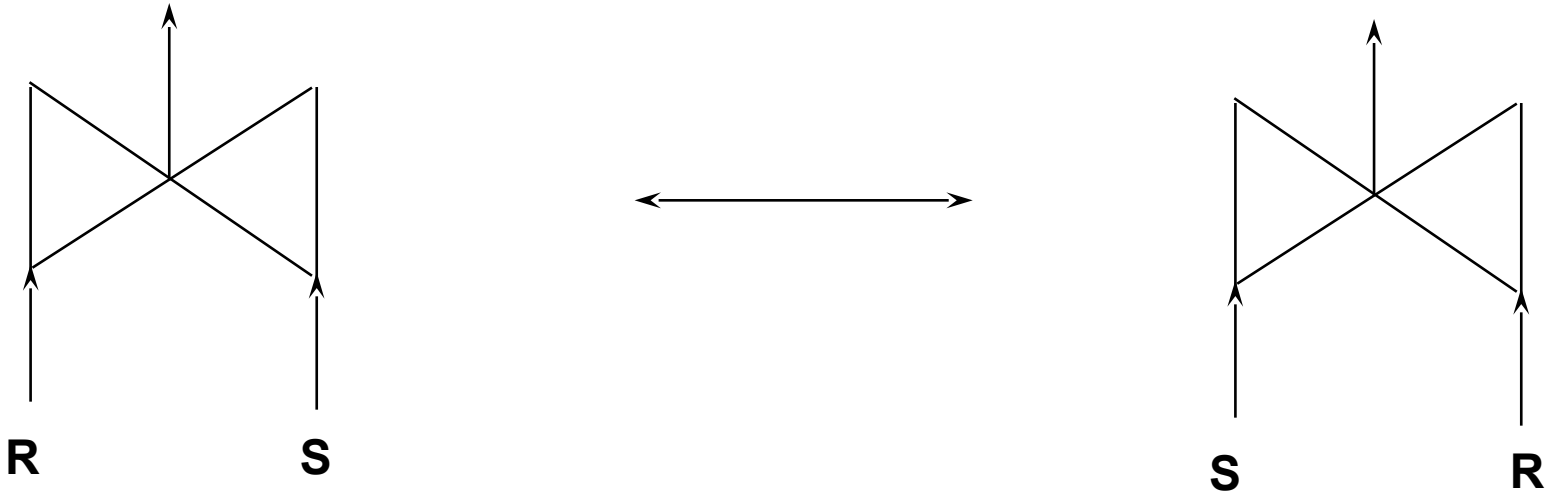
- **Problème :**
  - suivant l'ordre des opérateurs algébriques dans un arbre, le coût d'exécution est diffèrent

- **Pourquoi?**
  - 1. le coût des opérateurs varient en fonction du volume des données traitées

    i.e., plus le nombre de tuple des relations traitées est petit, plus les coûts cpu et d'E/S sont minimisés
  - 2. certains opérateurs diminuent le volume des données

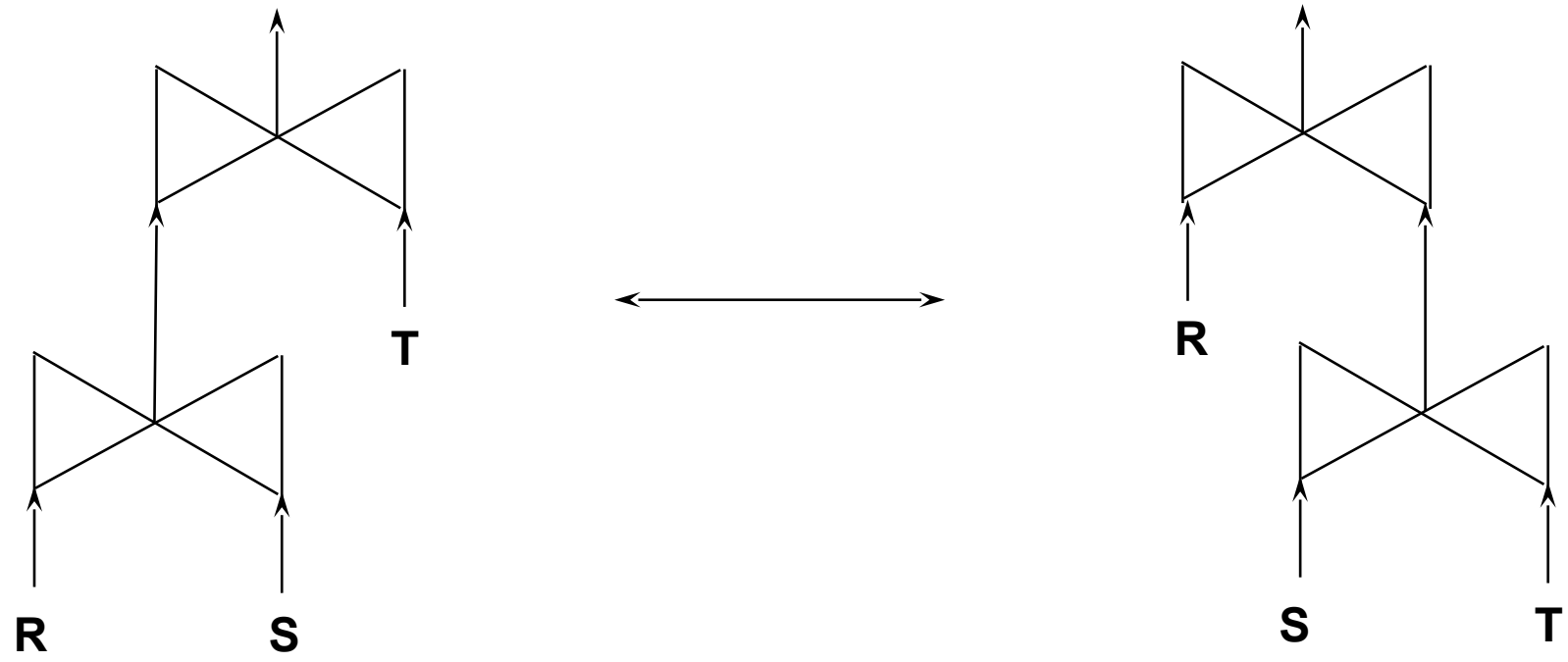
    e.g., restriction et projection

Commutativité des Jointures

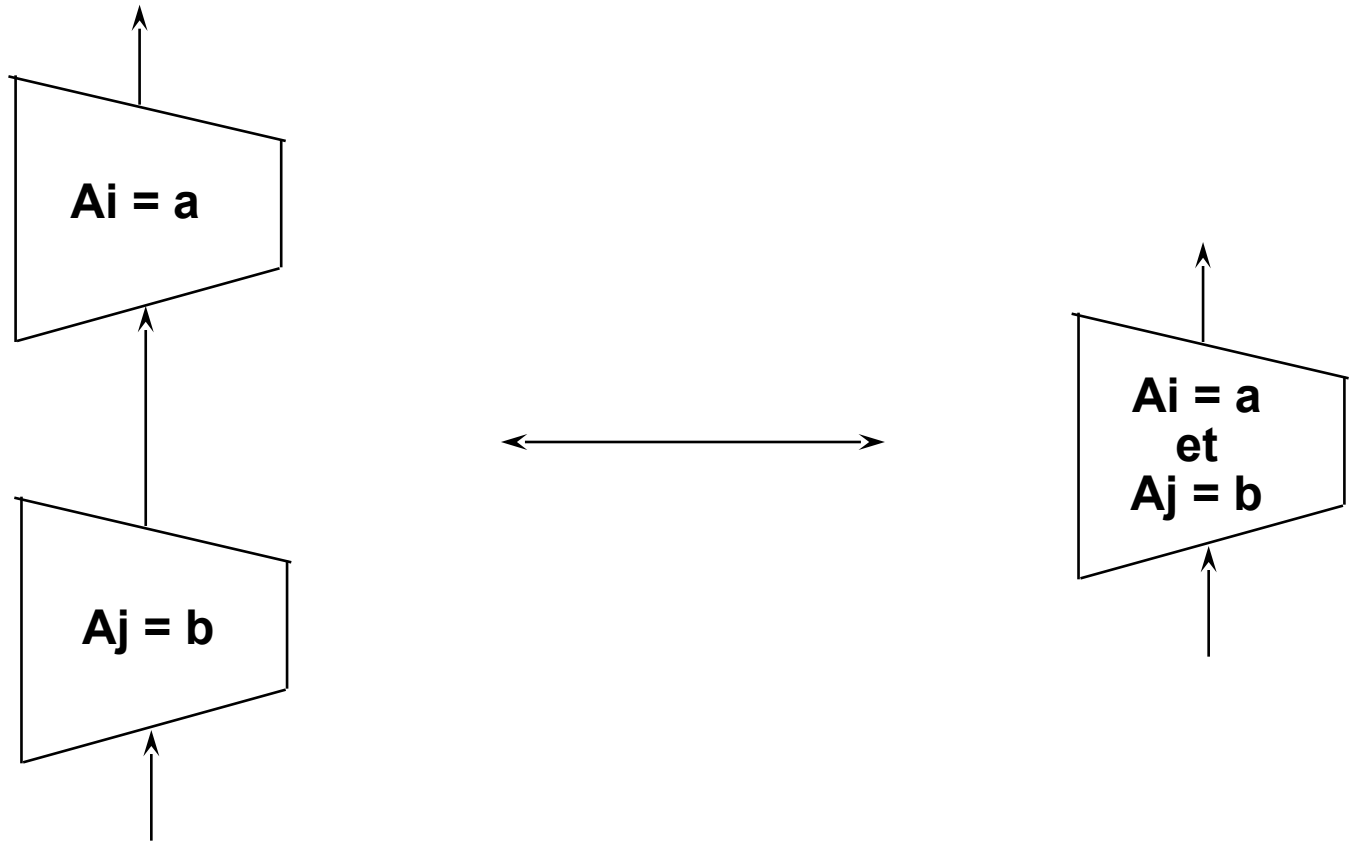


Associativité des jointures

- **Il existe N!/2 arbre de jointure de N relations.**
- **Parmi les jointures, certaines sont des produits cartésiens.**

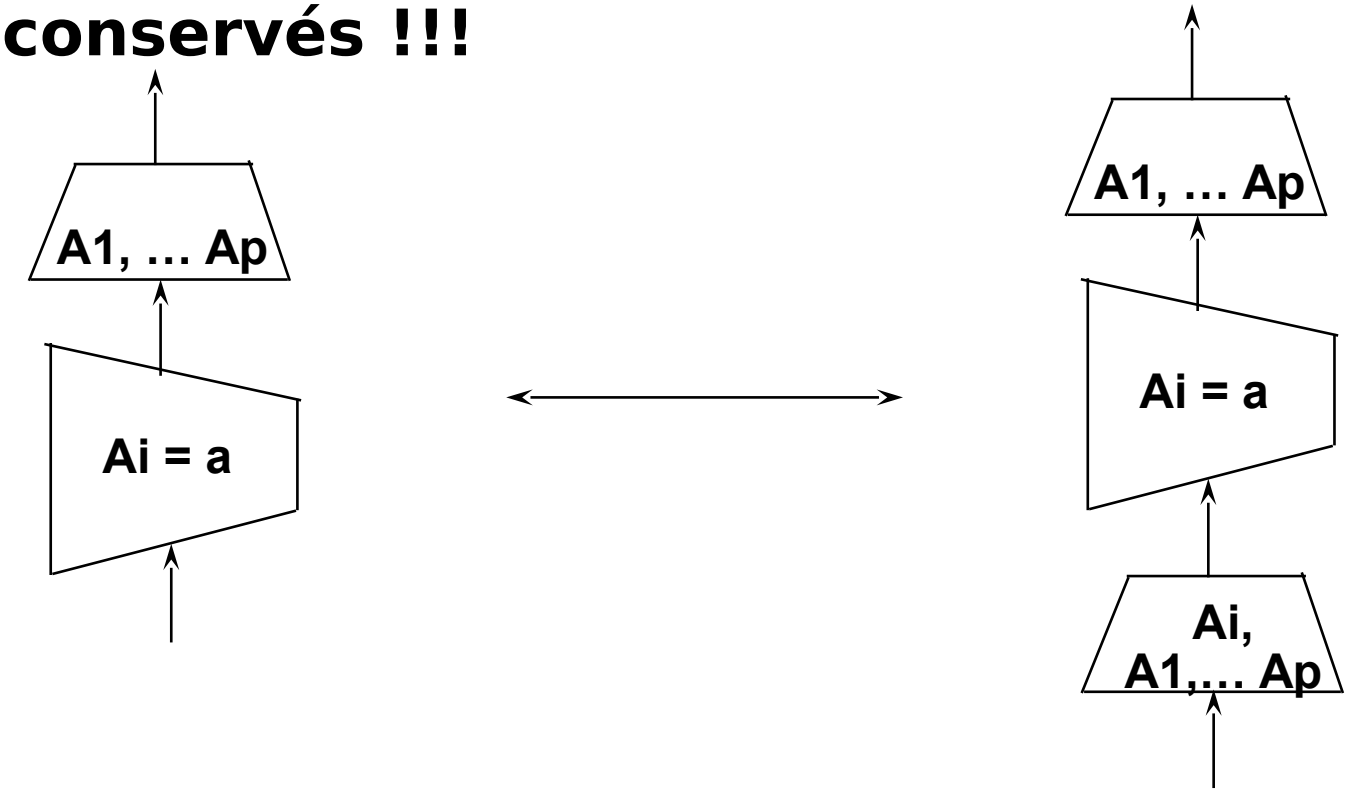


Groupage des Restrictions



Semi-commutativité des Projections

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



Règles de Restructuration

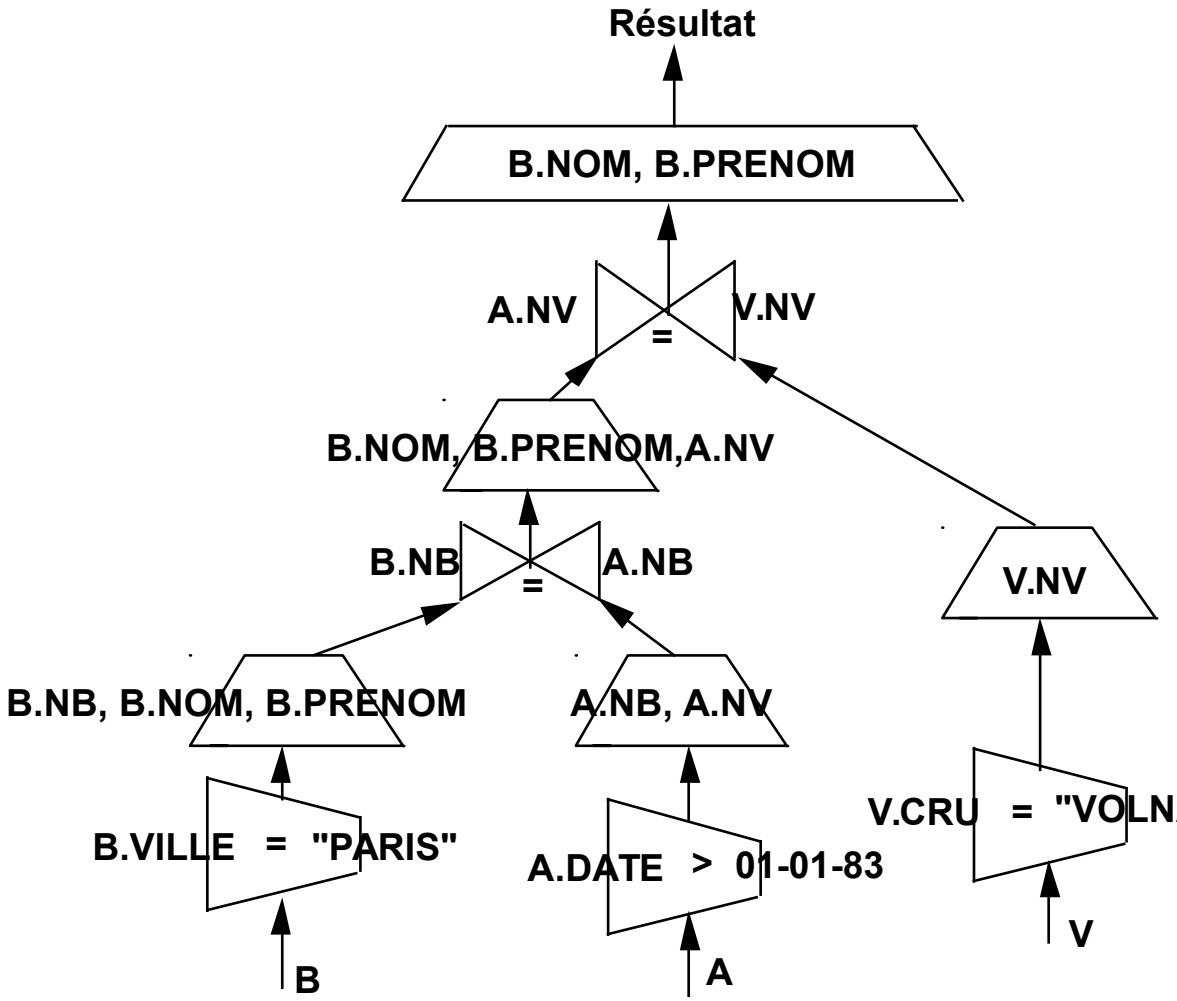
- **(1) Commutativité des jointures**
- **(2) Associativité des jointures**
- **(3) Groupabilité des restrictions**
- **(4) Semi-commutativité des projections et restrictions**
- **(5) Semi-commutativité des restrictions et jointures**
- **(6) Semi-distributivité des projections / jointures**
- **(7) Distributivité des restrictions / unions ou différences**
- **(8) Distributivité des projections / unions**

Heuristique d'Optimisation

- **Appliquer d'abord les opérations réductrices (restrictions et projections) en les groupant sur chaque relation.**
  - 1. Dégrouper les restrictions (Règle 3')
  - 2. Descendre les restrictions (Règles 4, 5 et 7)
  - 3. Grouper les restrictions aux feuilles (Règle 3)
  - 4. Descendre les projections (Règles 4, 6 et 8)

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

Exemple d'Arbre Optimisé



- **Coût d'exécution:**
- **10 m + 1m * 100000 + 1 m * 1000 + …**
- **de l'ordre de 10 ** 11 comparaisons de tuples !**

Ordonnancement des Jointures

- **HEURISTIQUES :**
  - Choix des relations de taille minimum
  - Jointures pré-calculés d ’abord (indexes)
  - Semi-jointures plus réductrices

- **ORDONNANCEMENT DES AGREGATS**
  - Permutations difficiles
  - Profiter des tris des jointures, dédoublement, etc..
  - Gains importants pour MIN et MAX

4. MODELE DE COUT

- **Facteur de sélectivité**
  - Proportion de tuples du produit cartésien des relations touchées qui satisfont une condition.
- **Exemple:**
  SELECT *
  FROM R1, R2
  ==> s =1
  SELECT *
  FROM R1
  WHERE A = valeur
  ==> s = 1/NDIST(A) avec un modèle uniforme

Sélectivité des Restrictions

- **TAILLE (σ(R)) = s * TAILLE(R) avec:**

  s (A = valeur) = 1 / NDIST(A)

  s(A > valeur) = (max(A) - valeur) / (max(A) - min(A))

  s(A < valeur) = (valeur - min(A)) / (max(A) - min(A))

  s (A IN liste valeurs) = (1/NDIST(A)) * CARD(liste valeurs)

  s(P et Q) = s(P) * s(Q)

  s(P ou Q) = s(P) + s(Q) - s(P) * s(Q)

  s(not P) = 1 - s(P)

- **Le coût dépend de l'algorithme (index, hachage ou balayage).**

Sélectivité des Projections

- **TAILLE(πx(R)) = p(x) * (1-d) * TAILLE(R)**
  - avec p(x) = Larg(x) / Larg(R)
  - d = probabilité de doubles
  - CARD(X) / CARD(DOM(X))**2

Sélectivité des Jointures

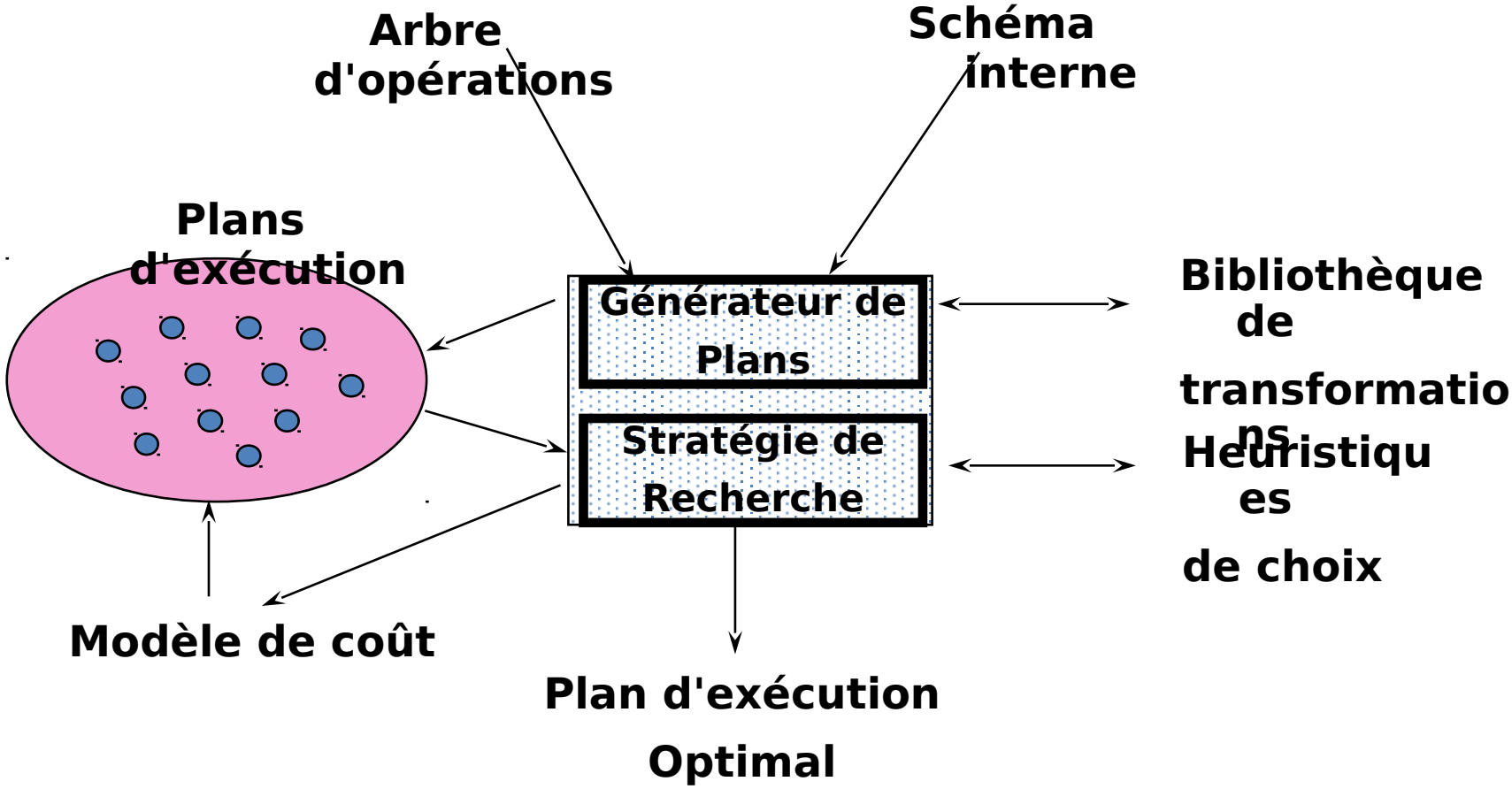
- **TAILE(R1 |><| R2) = p * TAILLE(R1) * TAILLE(R2)**
  - p dépend du type de jointure et de la corrélation des colonnes :
    - p = 0 si aucun tuple ne joint
    - p = 1 / MAX(NDIST(A),NDIST(B)) si distribution uniforme équiprobable des attributs A et B sur un même domaine
    - p = 1 si produit cartésien

- **L'algorithme change radicalement les coûts**
  - linéaire si index,
  - produit des tailles si boucles imbriquées.

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

5. CHOIX DU MEILLEUR PLAN



Sélectivité minimum

```
Rel = liste des relations à joindre ;
p = plus petite relation ;
Tant que Rel non vide {
      R = relation de selectivité minimum
  de Rel ;
      p = join(R,p) ;
      Relations = Relations - R ; } ;
Return(p) ;
```

Programmation Dynamique

PlanOuverts = liste de tous les plans mono-relation possible ;

Eliminer tous les plans équivalents excepté le moins coûteux ;

Pour chaque PlanOuverts p {

Pour chaque opérateur n'appartenant pas au plan p {

Etendre le plan en ajoutant cet opérateur ;

Calculer le coût du nouveau plan ;

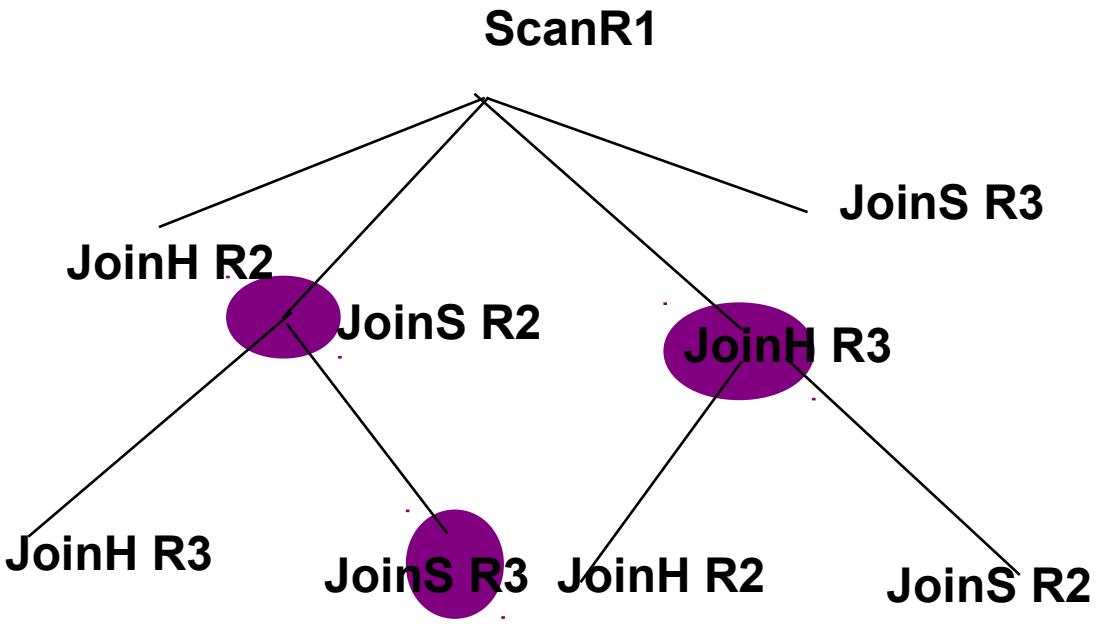
Insérer le nouveau plan dans la liste Nouveaux ; }

Eliminer tous les plans équivalents excepté le moins coûteux ;

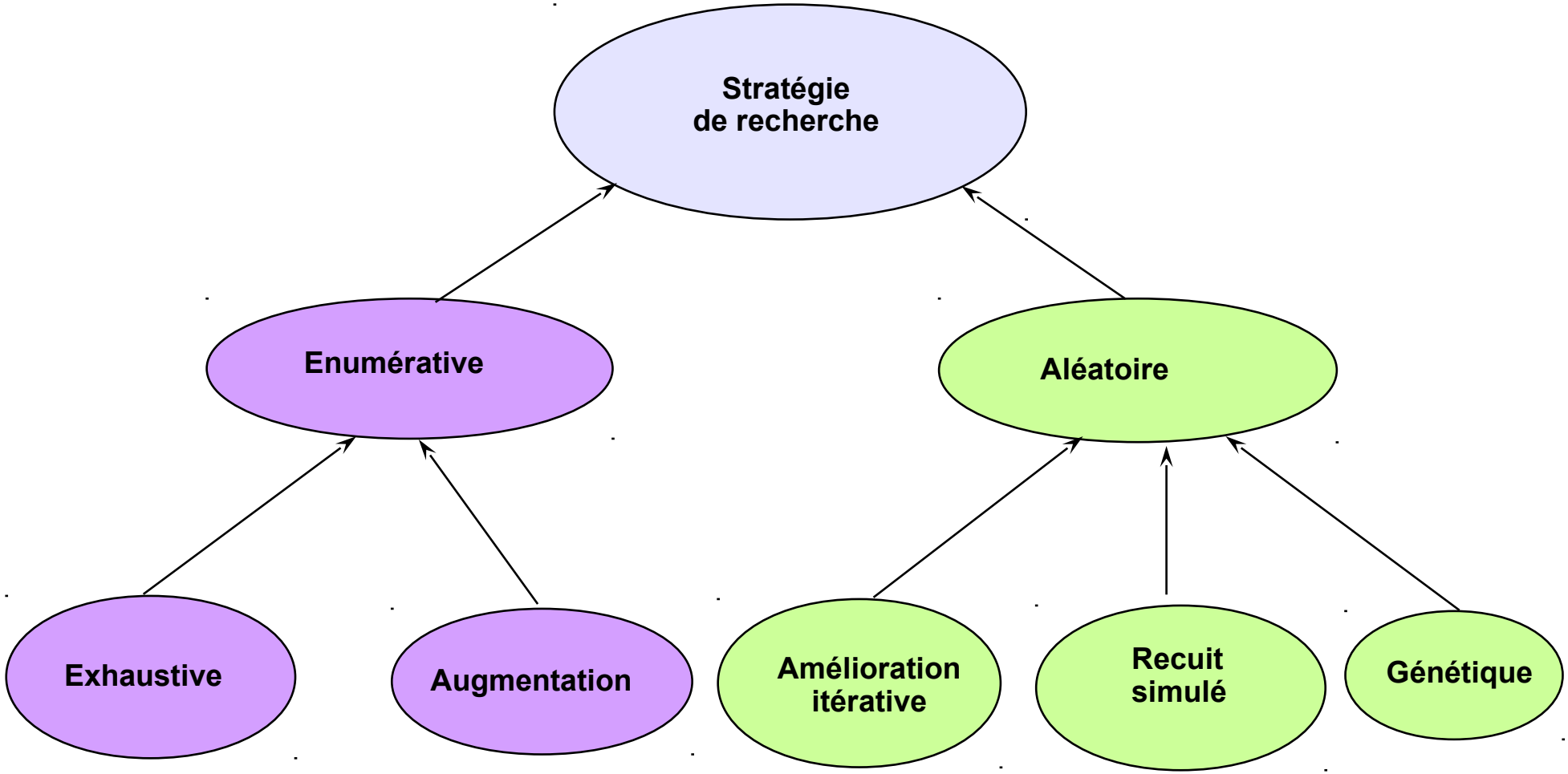
Transférer les plans Nouveaux dans PlanOuverts ; }

Retourner le plan optimal ;

Illustration DP



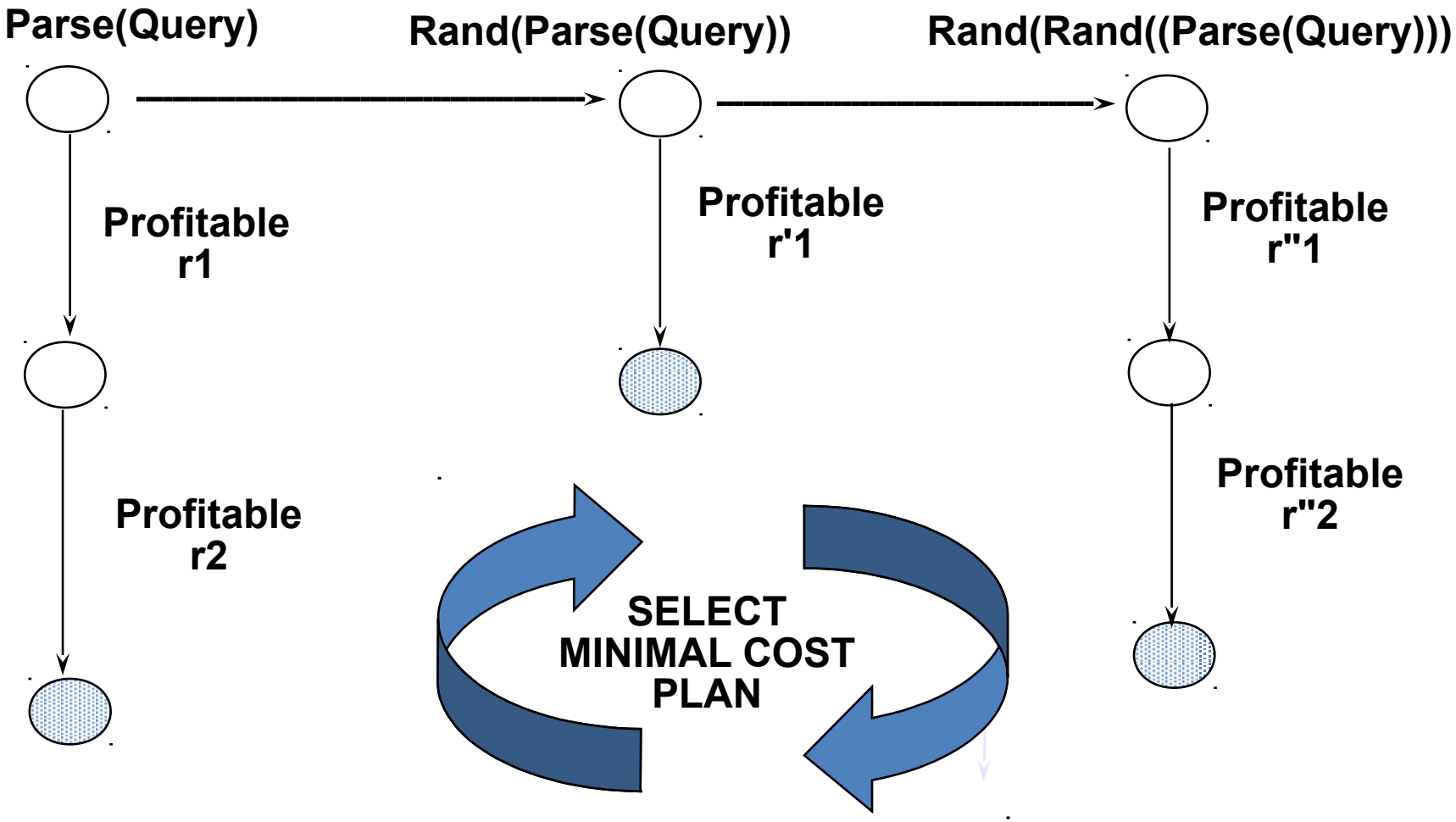
Différentes Stratégies



Amélioration itérative

```
Function Iterative(Query)
p:= Parse(Query) ; //  Set the initial plan
    S := {} ; // S is the set of locally optimum plans
    while not StopCond()
    {     nmoves := 0;
    while nmoves < MaxMoves(Query) and Transformable(p)
    {     p' := Transform (p) ; // Apply a transformation rule
    if  Cost(p') < Cost(p) then
    {     p ::= p';
    nmoves ::= nmoves + 1;
    }
    Insert (S, p') ; // Maintain the set of interesting plans
    p := Random(Parse(Query)); // Generate a new initial plan at
    random
    }
    }
    return Optimal(S) ;  // Select best plan among all generated ones }
```

Illustration II



6. CONCLUSION

- **Problème essentiel des SGBD**
- **Nécessité d'un modèle de coût**
- **Approches par compilation dans un langage d'accès (opérateurs avec annotations)**
- **Stratégies de choix aléatoires**