Base de données

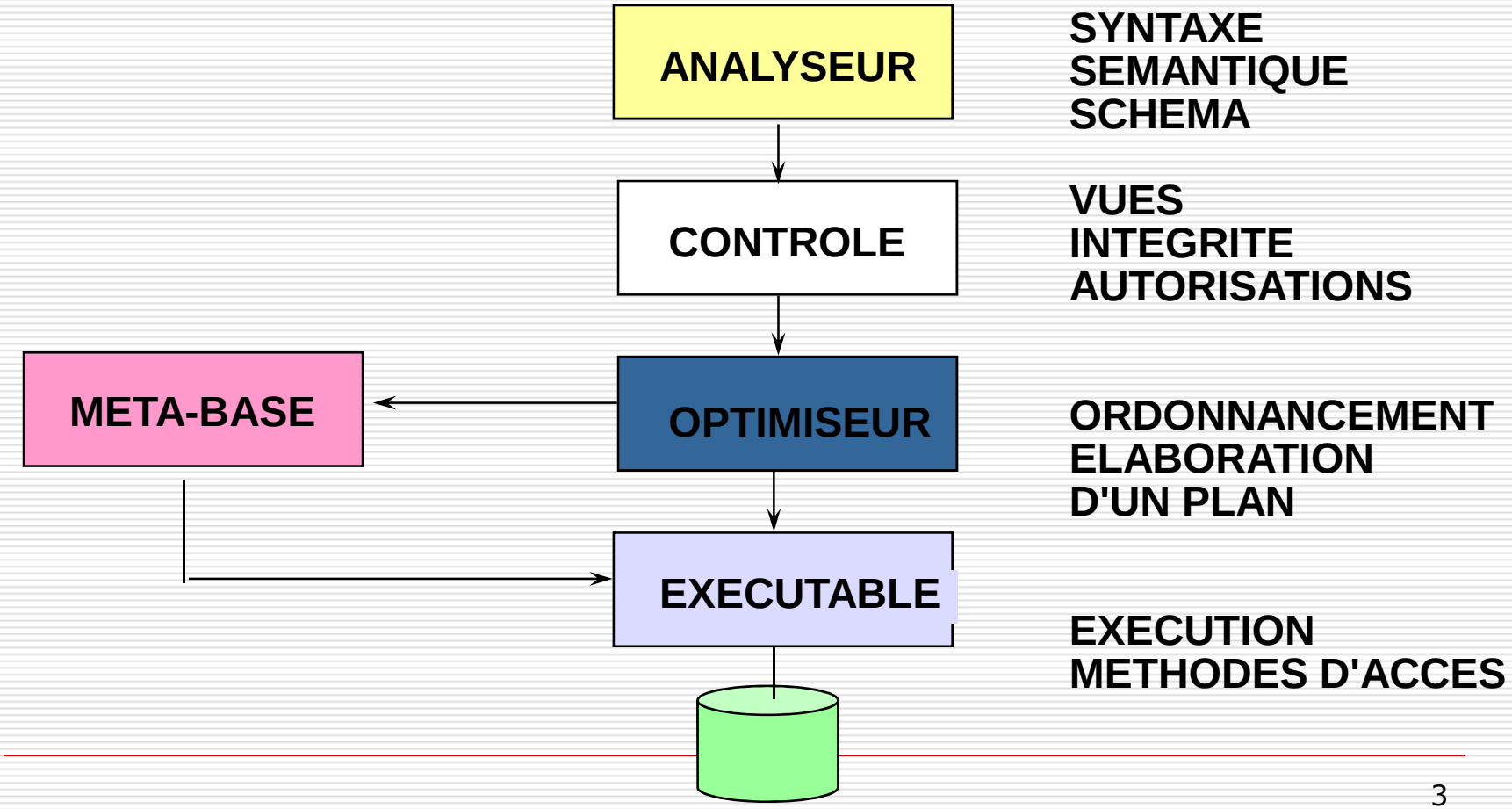
Séance 11

Optimisation des requêtes

Plan

- Objectifs
- Optimisation de requêtes
  - Réécriture (optimisation logique)
  - Planning (optimisation physique)
- Benchmarks

Architecture type SGBD



Optimisation de requêtes : phases

1. Réécriture
   - vérification syntaxique de la requête
   - obtention d'une représentation canonique
   - transformation par ordonnancement des opérations élémentaires
2. Planning = construction des plans d'exécution candidats
   - choix des algorithmes pour chaque opérateur
   - calcul du coût de chaque plan
   - choix du meilleur plan

Optimisation logique ou réécriture

- Problème :
  - suivant l'ordre des opérateurs algébriques dans un arbre, le coût d'exécution est différent
- Pourquoi?
  1. le coût des opérateurs varient en fonction du volume des données traitées, i.e., plus le nombre d'enregistrements des relations traitées est petit, plus les coûts CPU et d'E/S sont minimisés
  2. certains opérateurs diminuent le volume des données, i.e., restriction et projection
- But :
  - Réécrire la requête sous forme canonique simplifiée et logiquement optimisée.

Rappel : Arbre algébrique

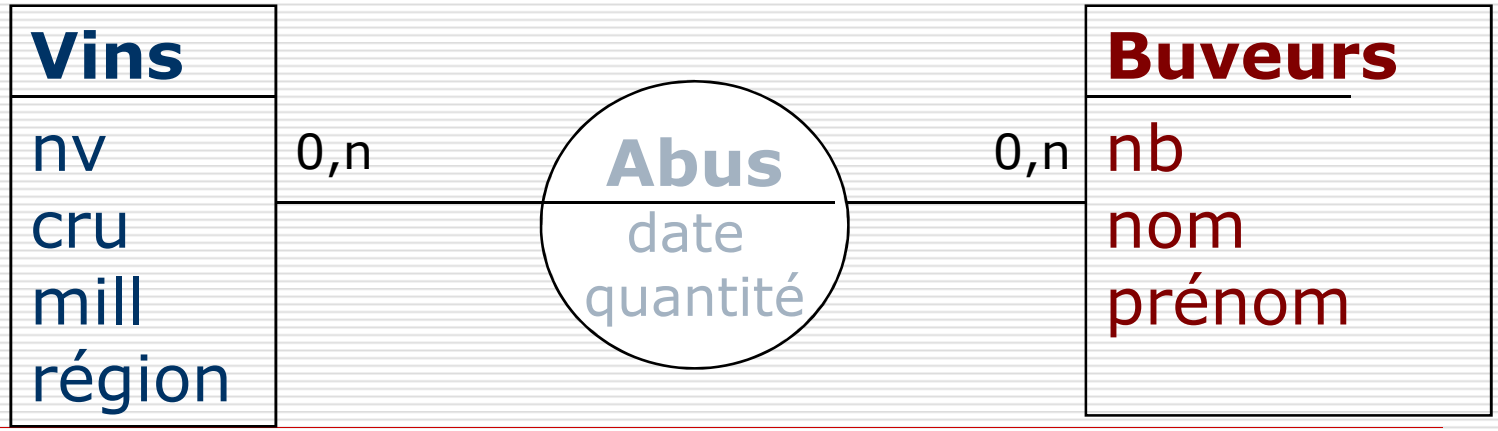
- Arbre représentant une question dont
  - les nœuds terminaux représentent les relations,
  - les nœuds intermédiaires des opérations de l'algèbre relationnelle,
  - le nœud racine le résultat d'une question,
  - et les arcs les flux de données entre les opérations.

Rappel : base des buveurs

Vins(nv, cru, mill, region)

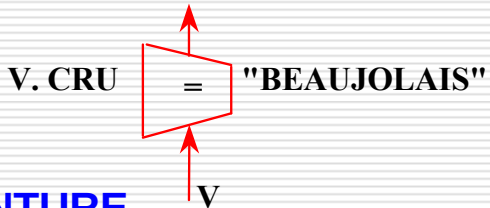
Buveurs(nb, nom, prenom, adresse)

Abus(*nb, nv, date*, quantite)

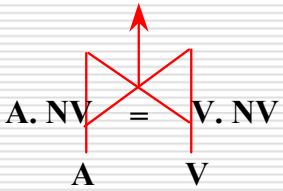


Rappel : opérateurs algébriques, tri et agrégat

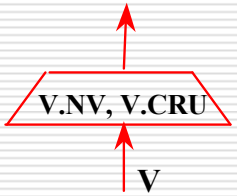
- **RESTRICTION**



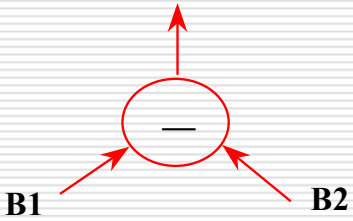
- **JOINTURE**



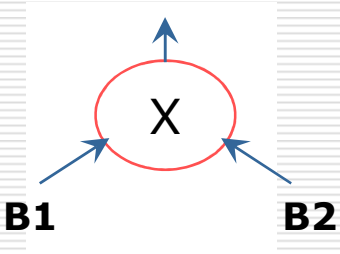
- **PROJECTION**



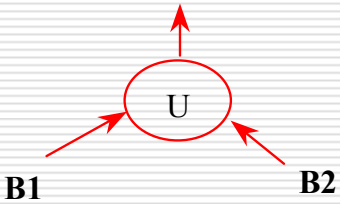
- **DIFFERENCE**



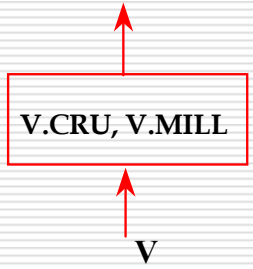
- **PRODUIT CARTESIEN**



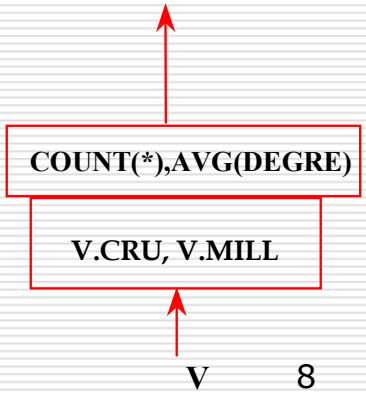
- **UNION**



- **TRI**



- **AGREGAT**



Exemple

- Rechercher tous les buveurs qui ont bu du vins "Volnay" après le 1er janvier 1990 et habitent à Paris.

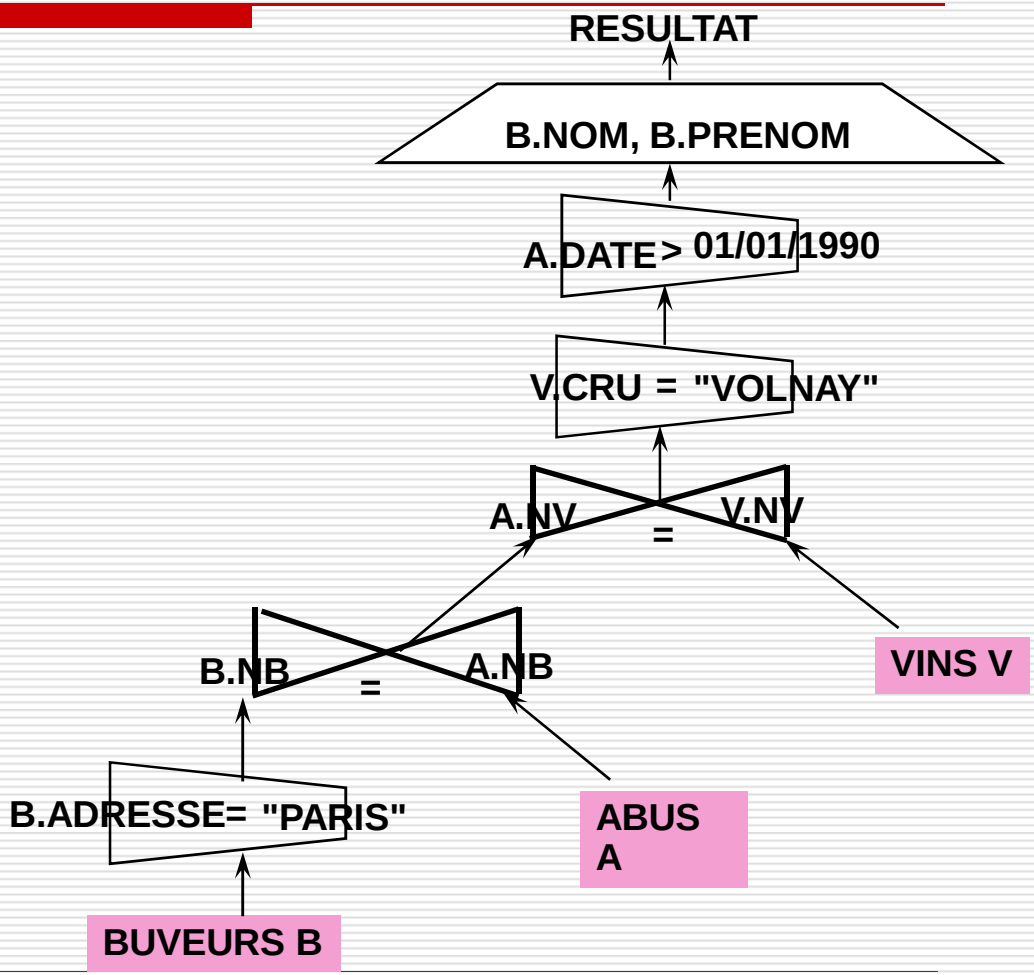
```
SELECT  B.nom, B.prenom
FROM    Buveurs B, Abus A, Vins V
WHERE   A.nv = V.nv
AND     B.nb = A.nb
AND     A.date > 01/01/1990
AND     V.cru = "Volnay"
AND     B.adresse = "Paris"
```

Un arbre algébrique pour cette requête

- 10 millions de buveurs dont 1 million à Paris
- 20 millions d'abus dont 10000 de Volnay, 100000 après le 01/01/1990
- 1000 vins
- Coût d'exécution:

  10m + 20m*1m + 20m*1000 + 20m + 10000 + …

  = de l'ordre de 10**13 comparaisons de tuples

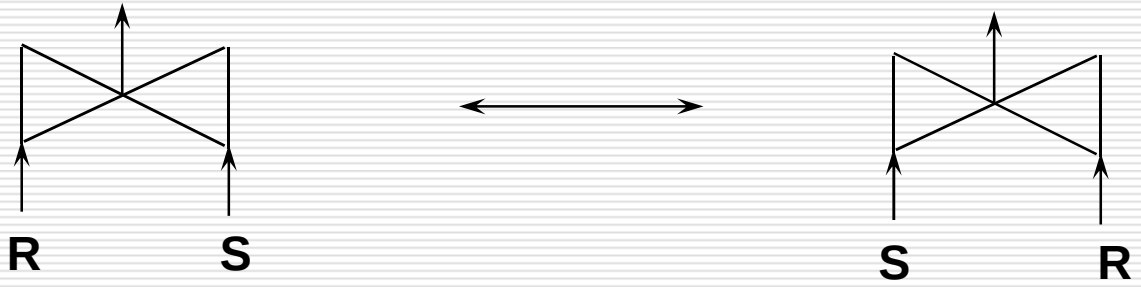


Règles de restructuration

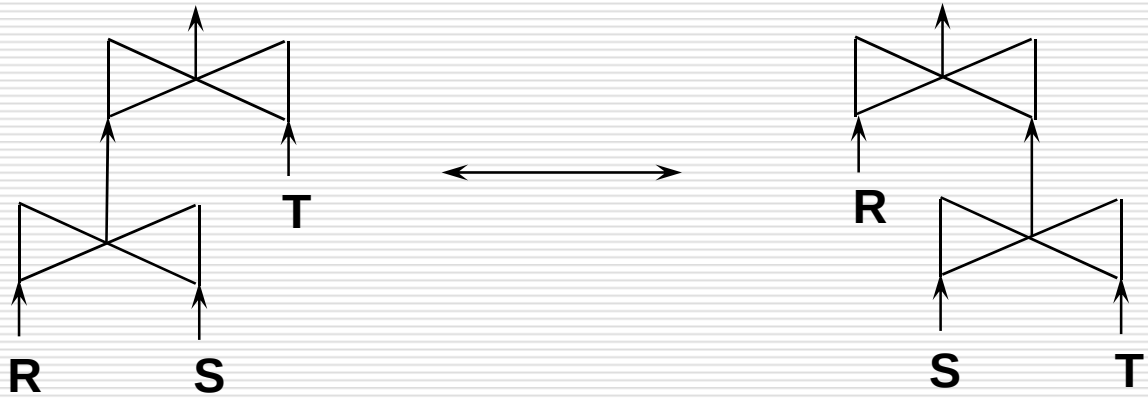
1. Commutativité des jointures
2. Associativité des jointures
3. Fusion des projections
4. Regroupement des restrictions
5. Quasi-commutativité des restrictions et des projections
6. Quasi-commutativité des restrictions et des jointures
7. Commutativité des restrictions avec les unions, intersections ou différences
8. Quasi-commutativité des projections et des jointures
9. Commutativité des projections avec les unions

Règles de restructuration

1. Commutativité des jointures

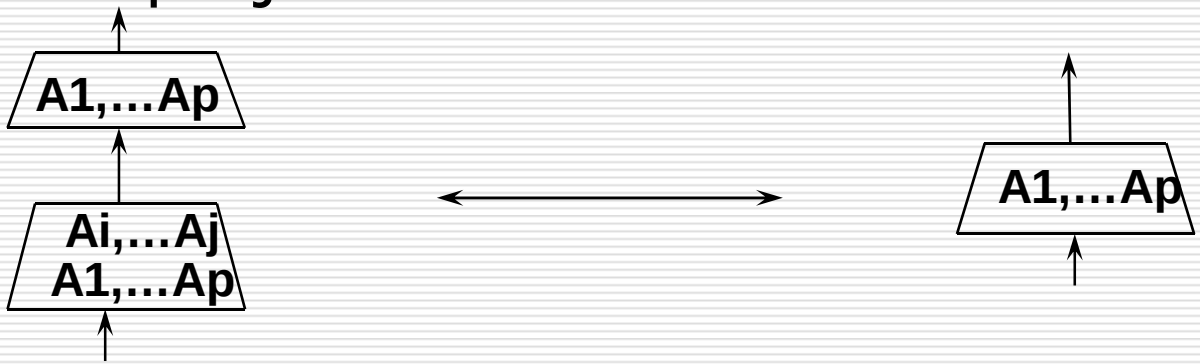


2. Associativité des jointures

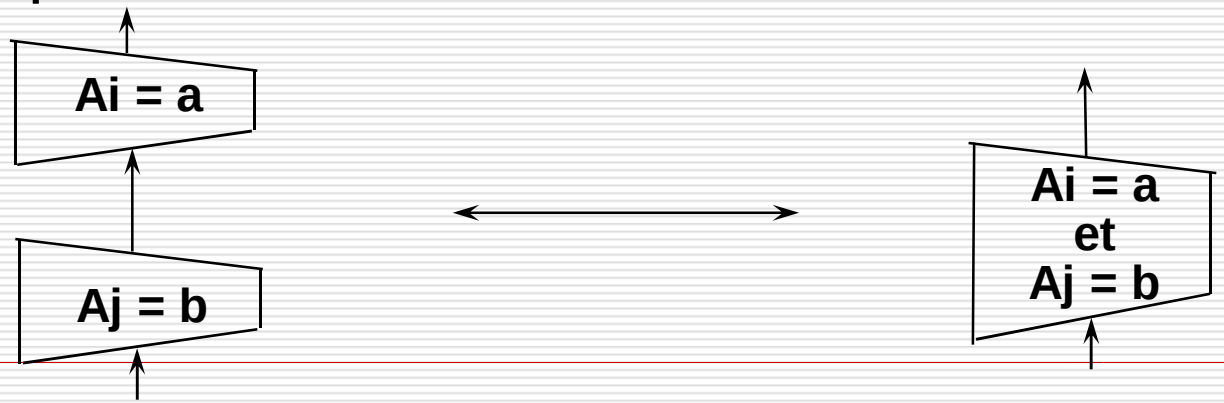


Règles de restructuration (cont.)

3. Fusion des projections

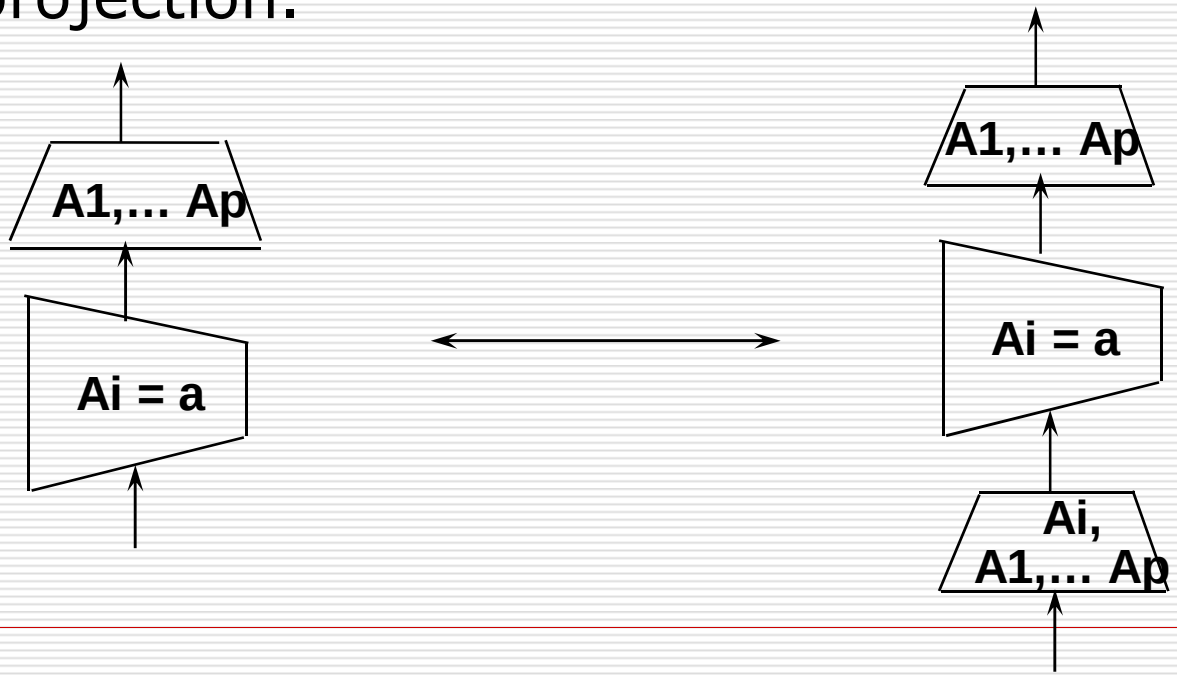


4. Regroupement des restrictions



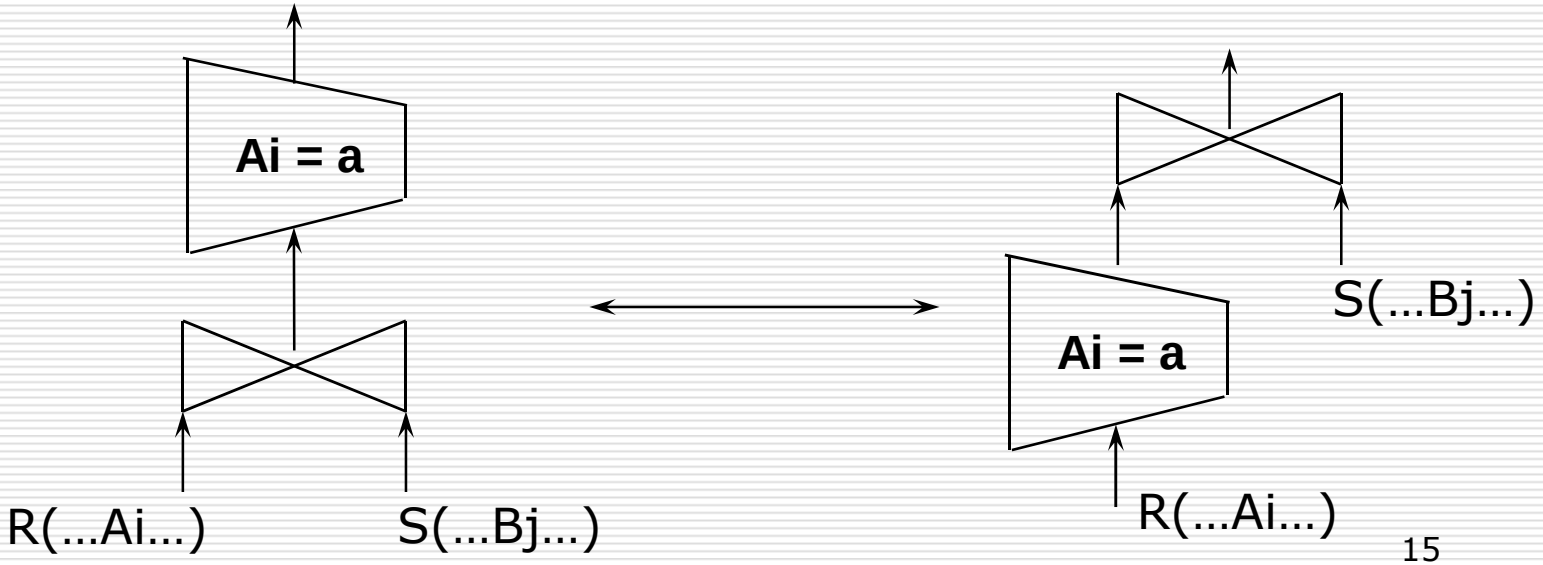
Règles de restructuration (cont.)

5. Quasi-commutativité des restrictions et projections
   - L'attribut de restriction doit être conservé par la projection.



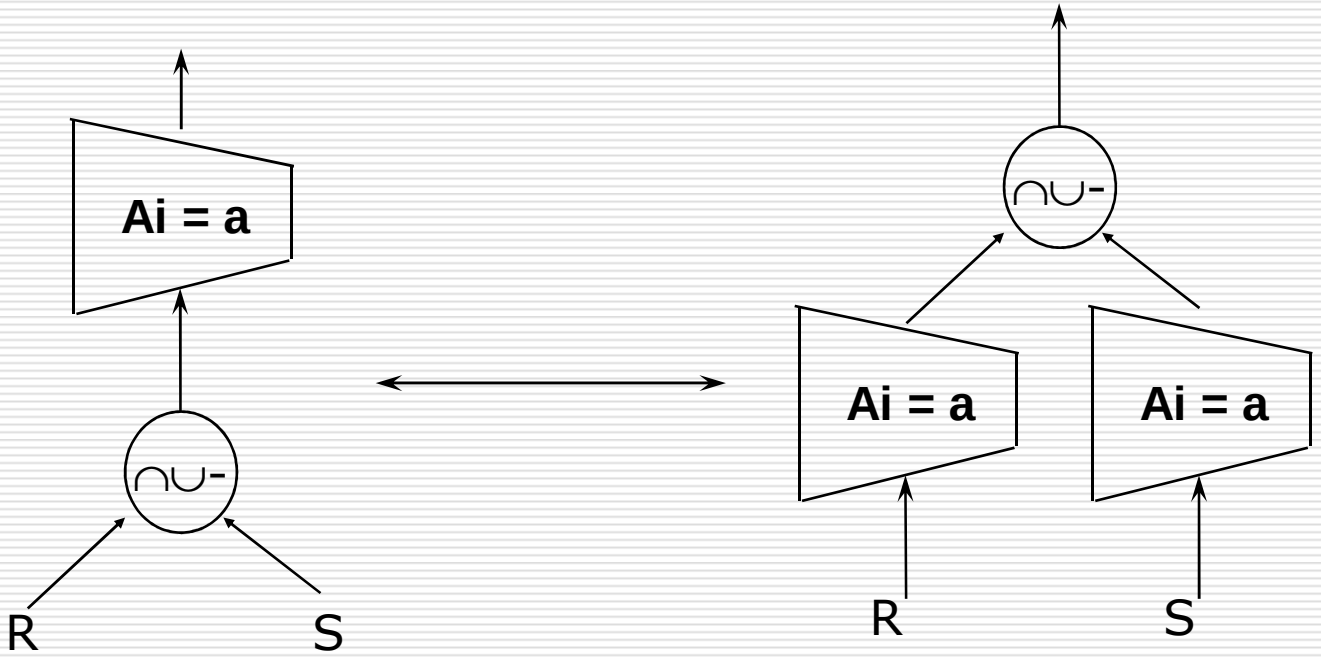
Règles de restructuration (cont.)

6. Quasi-commutativité des restrictions et des jointures
   - La restriction est appliquée sur la relation contenant l'attribut restreint



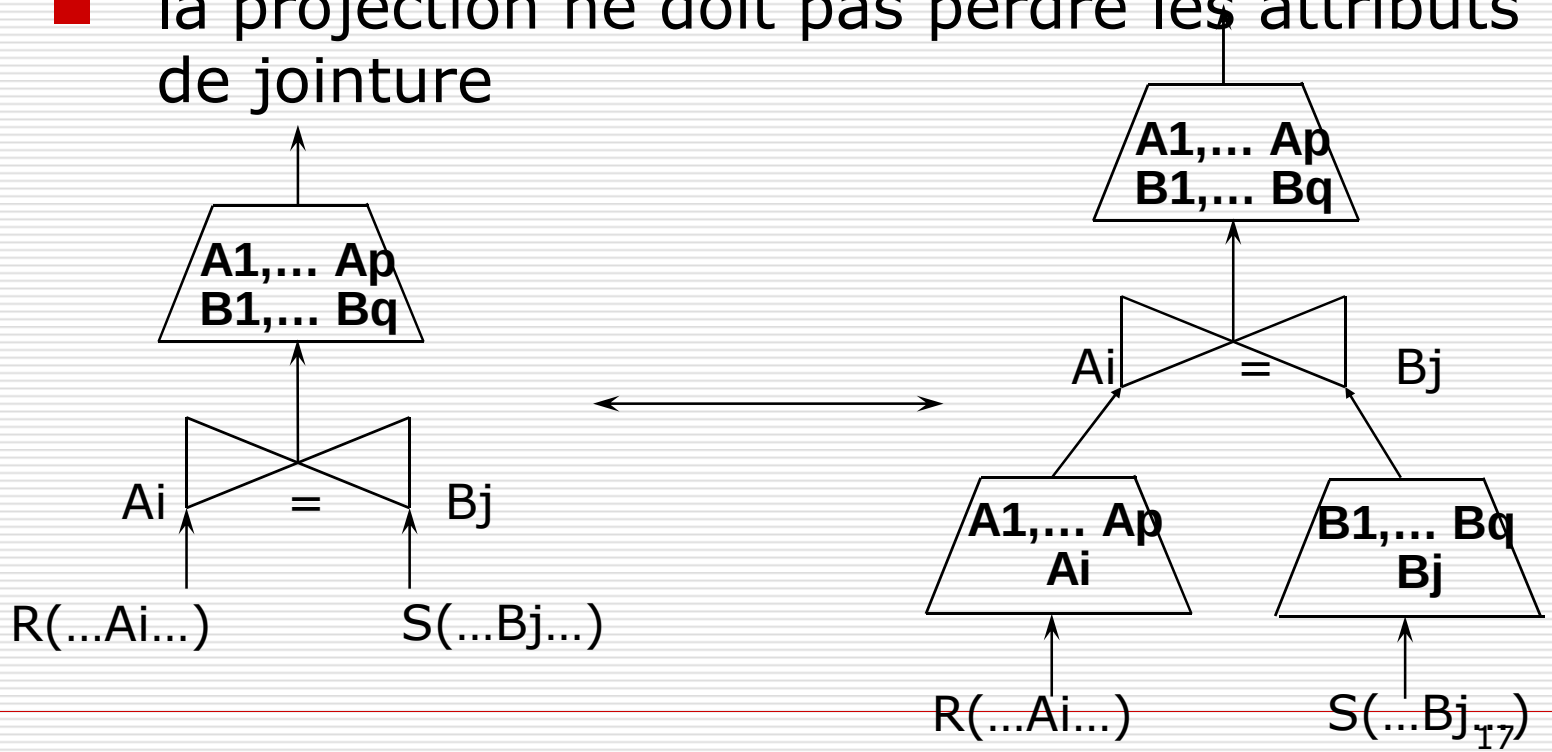
Règles de restructuration (cont.)

7. Commutativité des restrictions avec les unions, intersections ou différences



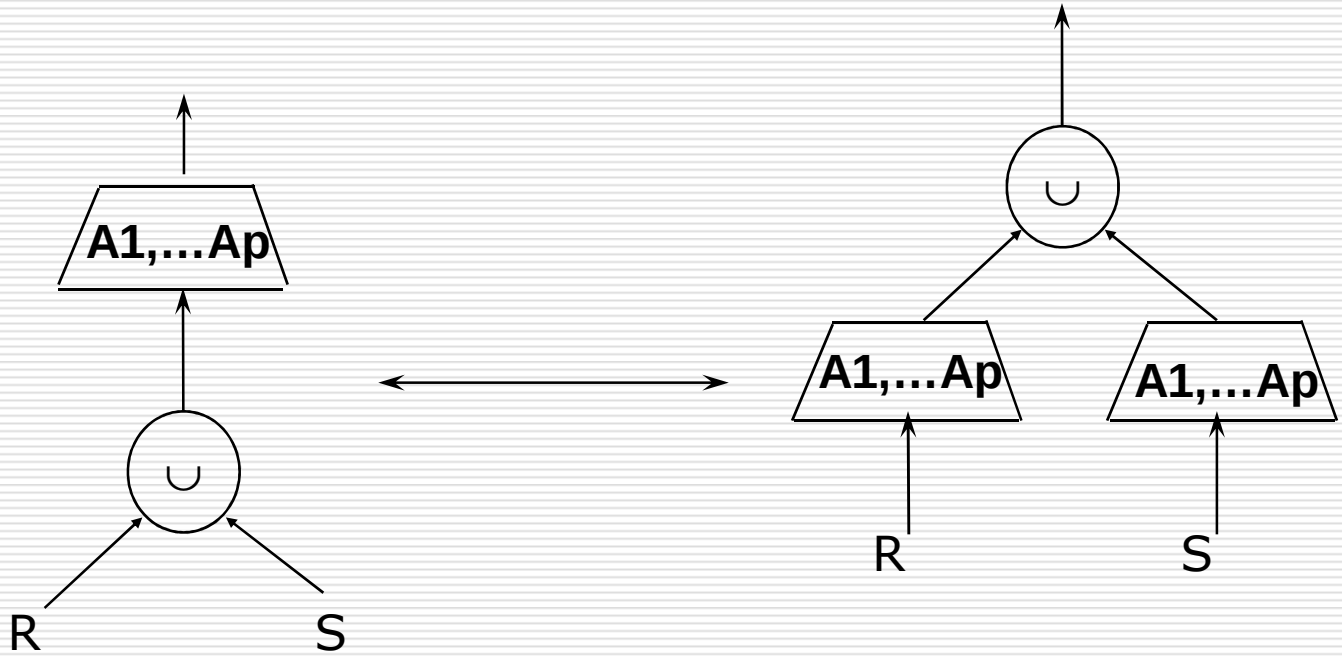
Règles de restructuration (cont.)

8. Quasi-commutativité des projections et des jointures
   - la projection ne doit pas perdre les attributs de jointure



Règles de restructuration (cont.)

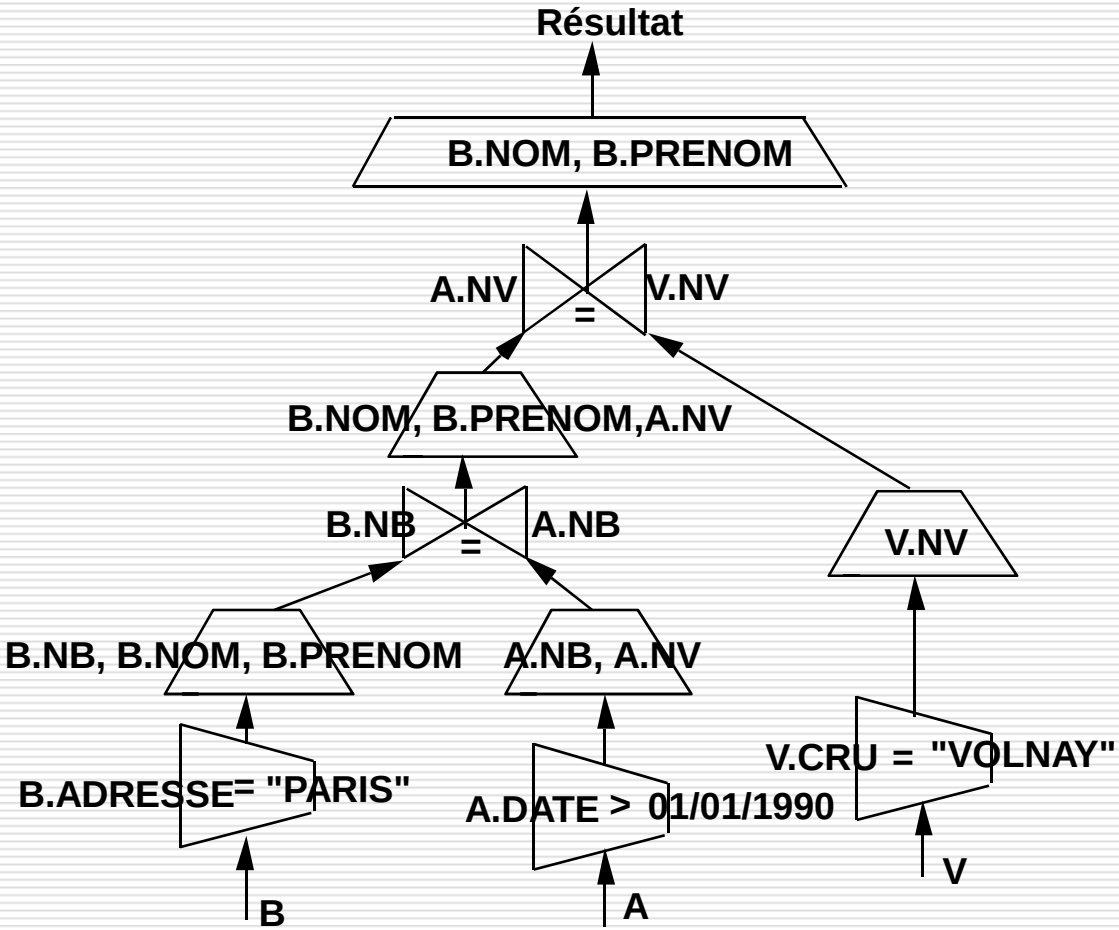
9. Commutativité des projections avec les unions



Heuristique d’optimisation

- Appliquer d'abord les opérations réductrices (restrictions et projections) en les groupant sur chaque relation.
  1. Dégrouper les restrictions (Règle 4')
  2. Descendre les restrictions (Règles 6 et 7)
  3. Grouper les restrictions aux feuilles (Règle 4)
  4. Descendre les projections (Règles 8 et 9)
  5. Regrouper les projections successives (Règle 3)
- L'ordre des unions, différences et jointures reste inchangé.

Exemple d'arbre optimisé

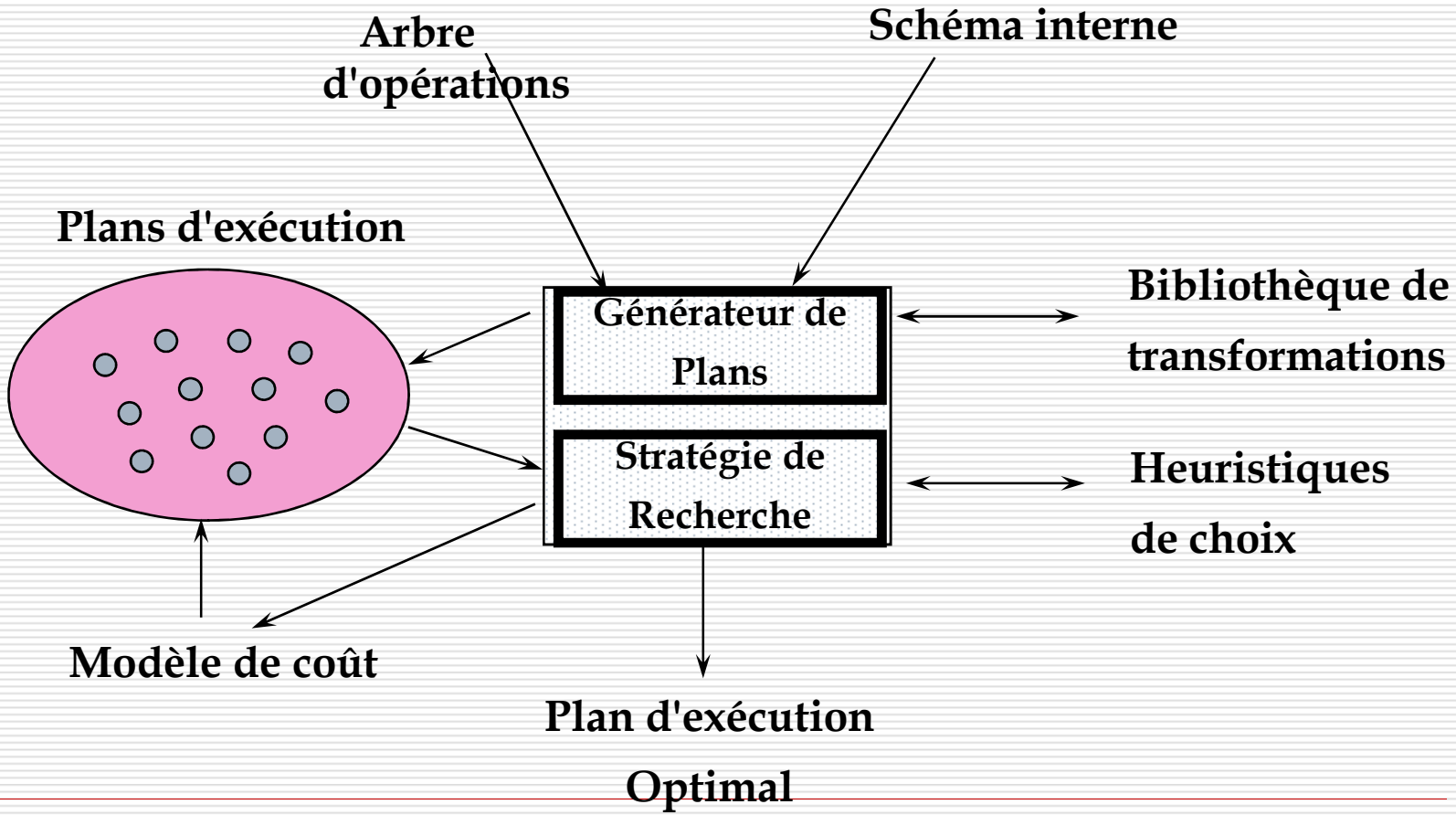


- Coût d'exécution:
  10m + 20m + 1m*100000 + 1000 + 100000*1000 + …
  de l'ordre de 10**11 comparaisons de tuples !

Optimisation physique

- But :
  - Effectuer le choix des meilleurs algorithmes pour les opérateurs de bas niveau compte tenu de la taille de données et des chemins d'accès disponibles.
- Problèmes :
  - Il faut pouvoir prendre en compte la taille des relations d'entrée, intermédiaires et de sortie
  - Il faut pouvoir prendre en compte des chemins d'accès aux relations (avec ou sans index, type d'index, hauteur d'index, hachage, mémoire disponible, …)
- Nécessité de développer un modèle de coût général permettant d'évaluer le coût d'un plan

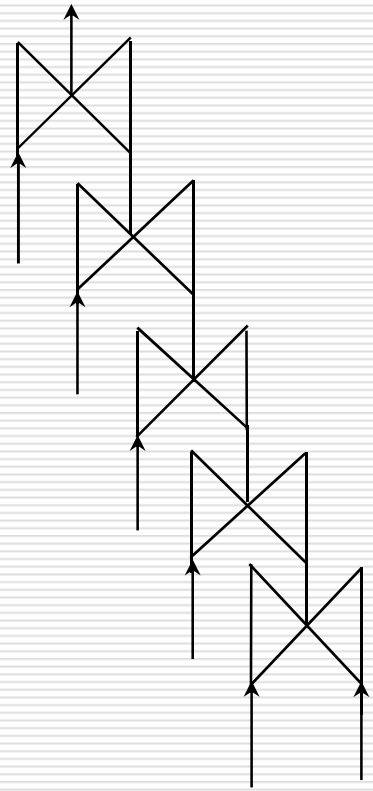
Choix du meilleur plan



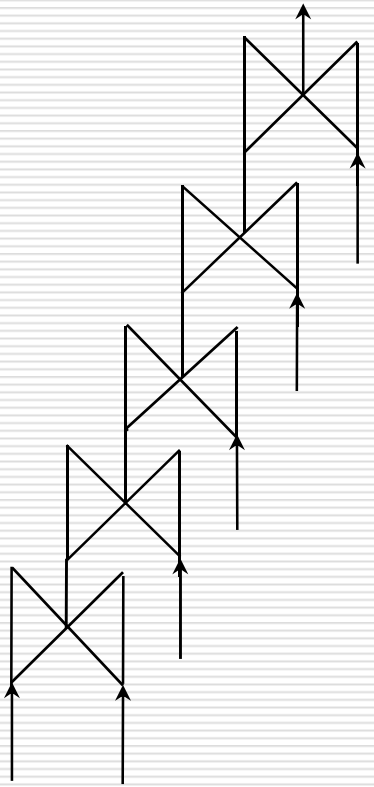
Plans d'exécution

- Nombre de plans d’exécution possible est très grand
- Heuristique :
  - Éviter des plans avec des produits cartésiens
  - Éviter des arbres ramifiés
  - Éliminer les plans qui n’effectuent pas les restrictions dès que possible.
  - …

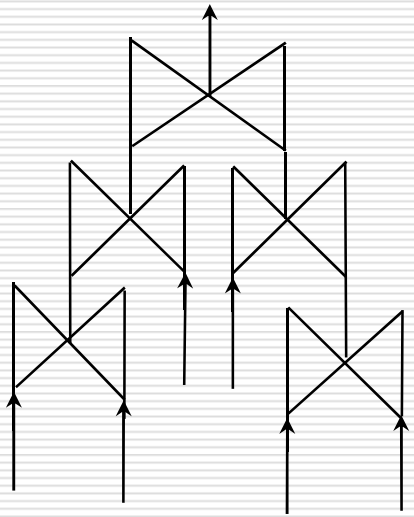
Typologie des arbres



Arbre linéaire droit



Arbre linéaire gauche



Arbre ramifié

Estimation du coût

- TUPLE(A) : nombre de valeurs d'un attribut A
- MIN(A) : valeur minimale d'un attribut A
- MAX(A) : valeur maximale d'un attribut A
- NDIST(A) : nombre de valeurs distinctes d'attribut A
- TAILLE(R) : nombre d'enregistrements de relation R
- s(critère) : facteur de sélectivité
  - Coefficient associé à une opération sur une table représentant la proportion de tuples de la table satisfaisant la condition de restriction.

Exemple de facteur de sélectivité

SELECT *
FROM R1, R2
➔ s = 1

SELECT *
FROM R1
WHERE A = valeur
➔ s = 1/NDIST(A), avec l'hypothèse de distribution uniforme des valeurs d'attributs

Sélectivité des restrictions

- TAILLE (RESTRICT(R,condition)) = 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) = (1/NDIST(A)) * CARD(liste)
  s(P et Q) = s(P) * s(Q) *// P et Q sont indépendantes*
  s(P ou Q) = s(P) + s(Q) - s(P) * s(Q)
  s(not P) = 1 - s(P)

Sélectivité des projections

- TAILLE(PROJECT(R,A1,A2,…An)) = (1-d) * TAILLE(R)

  - d : probabilité de doubles
  - A1, A2, … An : les attributs retenus dans la projection
  - d = (1 - NDIST(A1)/TAILLE(R)) *
    (1 - NDIST(A2)/TAILLE(R)) * …
    (1 - NDIST(An)/TAILLE(R))

Sélectivité des jointures

- TAILLE(JOIN(R1,R2,A op B)) = 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 / MIN(NDIST(A),NDIST(B)) si distribution uniforme équiprobable des attributs A et B sur un même domaine
  - p = 1 si produit cartésien

Algorithmes d'accès

- Indexé ?
- Trié ?
- Exemple :
  - restriction sans index :
    - pas trié : balayage séquentiel
    - trié : recherche dichotomique
  - restriction et jointure avec index :
    - accélérer la recherche (arbre B, arbre B+, …) mais ralentir les mises à jours => pénalisant si plus de 20% des enregistrements satisfont le critère de recherche

Exemple

VINS (NV, CRU, QUALITÉ, DEGRÉ, MILLÉSIME)

1, Beaujolais, Excellente, 11, 198
2, Chenas, Médiocre, 12, 1990
3, Julienas, Médiocre, 12, 1990
5, Beaujolais, Bonne, 10, 1985
6, Julienas, Médiocre, 12, 1987
7, Chenas, Excellent, 11, 1989
14, Chenas, Bonne, 10, 1994
15, Julienas, Excellente, 11, 1995
18, Beaujolais, Bonne, 10, 1988
25, Julienas, Médiocre, 12, 1990

INDEX SECONDAIRE SUR CRU

Beaujolais	1,5,18
Chenas	2, 7, 14
Juliennas	3, 6, 15, 25

INDEX SECONDAIRE SUR DEGRE

10	5, 14, 18
11	1, 7, 15
12	2, 3, 6, 25

Plan d’accès

(table VINS critère (CRU = « CHENAS ») AND (MILLESIME >= 1986) AND
(DEGRE = 12))

```
{     C = LIRE(index CRU entrée CHENAS)
      D = LIRE(index DEGRE entrée 12)
      L = INTERSECTION(C,D)
      Pour chaque l de L faire
      {     Tuple = LIRE(table VINS adresse l)
            if Tuple.MILLESIME >= 1986 alors
                  résultat = UNION(résultat, Tuple)
      }
}
```

DB Benchmarks

- Organisations : TPC, SPEC, AIM, SAP, BAPCo, ...
- TPC benchmarks (http://www.tpc.org)
  - TPC-A : mesure la performance dans les environnements typiques de type OLTP avec des maj. intensives
  - TPC-B : mesure le débit (nombre de transactions par secondes), pas OLTP
  - TPC-C : OLTP benchmark avec multiples types de transactions, BD plus complexe
  - TPC-W : benchmark pour des transactions Web e-commerce
  - …

TPC-C : OLTP benchmark

- Les transactions :
  - saisie de commande
  - livraison de commande
  - enregistrement de payement
  - vérification de l'état des commandes
  - contrôle de stock
- Les tables :
  - 9 tables
  - 100 000 produits, environ 10 produits par commande
  - 10 zones de vente (une zone correspond à environ 3 000 clients)
- Opérations : insert, update, delete, commit, abort
- 90% de transactions doit avoir un temps de réponse < 5s, le contrôle de stock < 20s

TPC-C

- Mesure de performance : nombre de transactions par minute (tpmC)
- Prix / performance : coût total du système
- Top performance :

Compagnie	**tpmC**	**prix/per**	**date**	**BDD**	**SE**
ORACLE	30 249 688	1.01 USD	6/1/11	Oracle Database 11g R2 EE	Oracle Solaris 10
IBM	10 366 254	1.38 USD	10/13/10	DB2 9.7	IBM AIX V6.1
ORACLE	7 646 486	2.36 USD	3/19/10	Oracle Database 11g	Sun Solaris 10

Conclusion

- Optimisation : problème essentiel des SGBD
- Nécessité d'un modèle de coût
- Stratégies de choix aléatoires
- Benchmarks