Université Abou Bakr Belkaid Tlemcen
Faculté des Sciences
Département d'informatique

Administration des Bases de Données

L3 2018-2019

Mr H.MATALLAH

Administration des Bases de Données

1. Notions fondamentales
2. SQL Avancé
3. Gestion d'intégrité et de cohérence
4. Vues et Index
5. Optimisation des requêtes
6. Gestion des transactions : Gestion des accès concurrents

CHAPITRE 5

OPTIMISATION DES REQUÊTES

Plan Chapitre 5

1. Introduction
2. Arbre algébrique
3. Exemple fixant la problématique
4. Traitement d'une requête
5. Stratégie générale d'optimisation
6. Optimisation basée sur les modèles de coût

Optimisation des Requêtes

Introduction

- **Optimisation :** Amélioration, maximalisation, rationalisation
- **Optimisation :** Donner le meilleur rendement possible
- **Optimisation en informatique :**
  - L'optimisation de code permet d'améliorer les performances d'un logiciel
  - L'optimisation pour les moteurs de recherche permet d'améliorer le classement d'un site web dans les résultats d'une requête sur les moteurs de recherche
  - Dans les bases de données, l'optimisation des bases de données et en particulier **l'optimisation de requête** définit la meilleure exécution par le SGBD d'un code donné

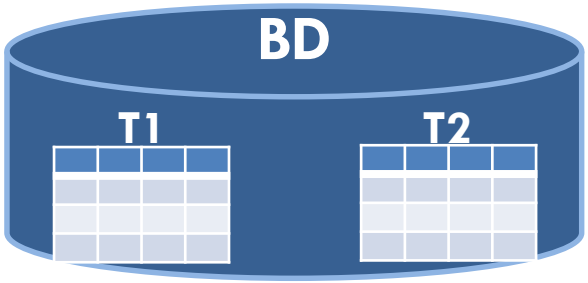
Optimisation des Requêtes

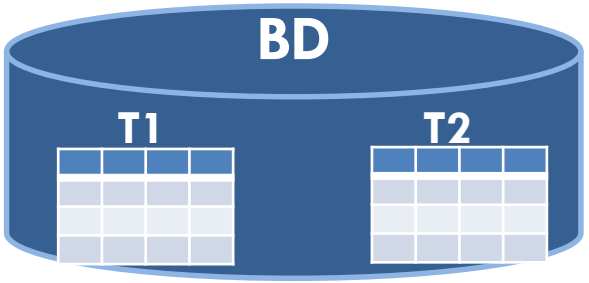
Introduction

- Parmi les objectifs des BD est de restituer l'information dans des **délais acceptables**
- **Optimisation** : Trouver la meilleure façon d'accéder et de traiter les données afin de répondre à la requête
- La plupart des SGBD adoptent des langages non procéduraux qui ne fournissent pas les chemins d'accès aux données, ainsi l'optimisation devient l'affaire du système
- Tout SGBD implante un ensemble de techniques de base d'optimisation et dispose d'un module d'évaluation de requêtes qui permet de choisir la meilleure stratégie possible d'exécution d'une expression relationnelle

Optimisation des Requêtes

- *Introduction*





Q

Tuples

Requête Q :

Select T1.A, T2.B

From T1,T2

Where T1.C=T2.C

Q

Tuples

Exécutée en 10ms !!

USER1

Une même requête peut s'exécuter de différentes manières

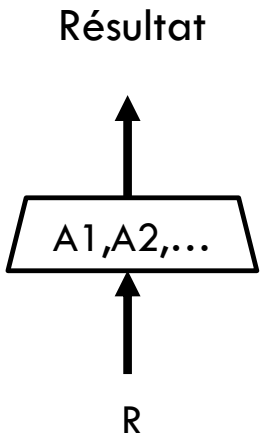
Exécutée en 110ms !!

USER2

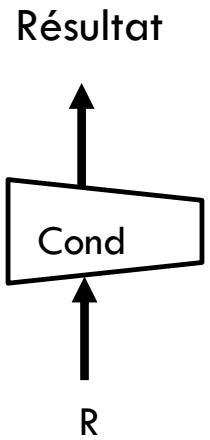
Optimisation des Requêtes

- ***Représentation graphique des opérateurs***

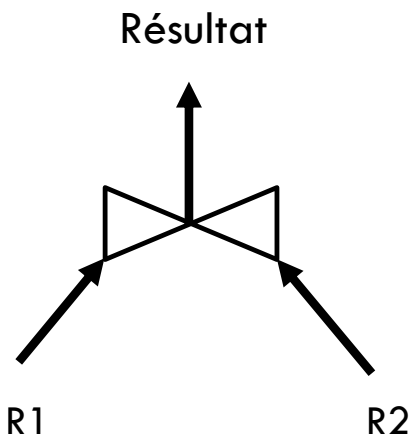
- **Projection**



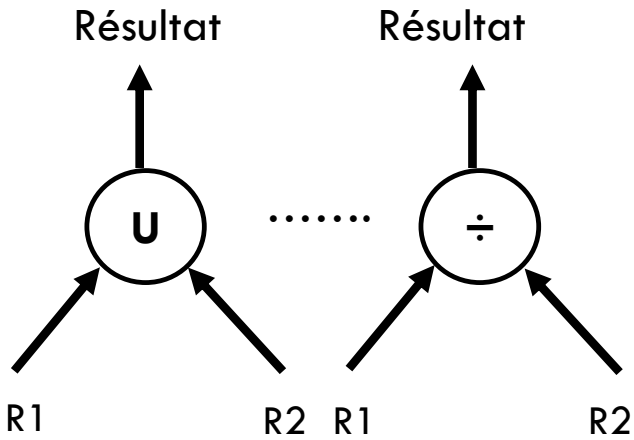
- **Sélection**



- **Jointure**



- **5 Autres opérateurs**



Optimisation des Requêtes

Arbre algébrique

- L'arbre algébrique ou l'arbre de requête est une description graphique plus lisible à traduire

- Un arbre de requête est une structure de données arborescente qui correspond à une expression de l'algèbre relationnelle :
  - Les relations fournies en entrée à la requête sont présentées sous forme de nœuds feuilles dans l'arbre
  - Les opérateurs de l'algèbre relationnelle sont présentés sous forme de nœuds internes

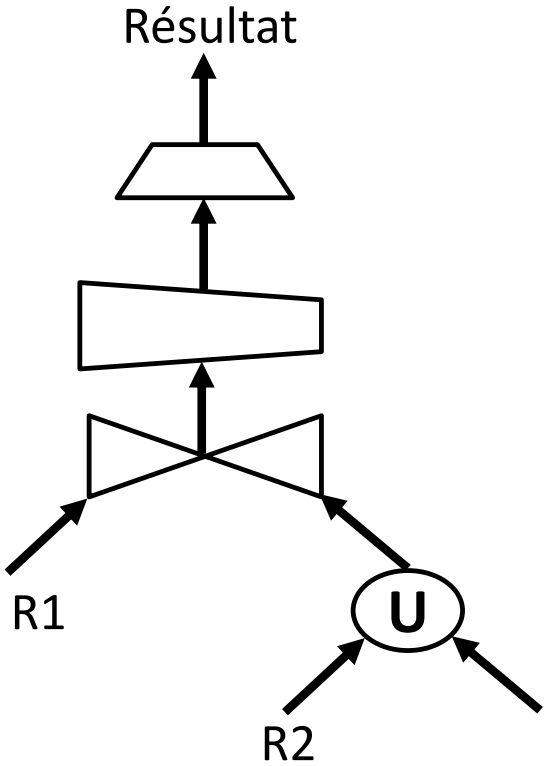
- L'arbre algébrique illustre **l'ordre d'exécution des opérateurs**

R3

Optimisation des Requêtes

- ***Arbre algébrique***
  - **Exemple :**

1. Union
2. Jointure
3. Sélection
4. Projection



Optimisation des Requêtes

Exemple fixant la problématique

- Soit la base de données du TD avec les hypothèses suivantes :
  - Cardinalité Fournisseur = 100
  - Cardinalité Commande = 10 000

    Cardinalité d'une table : Nombre de lignes
    Degré d'une table: Nombre de colonnes

- Soit la requête : « ***Donner les noms des fournisseurs qui ont fournit le produit P2999*** »
  - Requête SQL :

```
SELECT Nom
FROM Fournisseur, Commande
WHERE PNum = 'P2999' AND Fournisseur. FNum = Commande. FNum
```

Optimisation des Requêtes

Exemple fixant la problématique

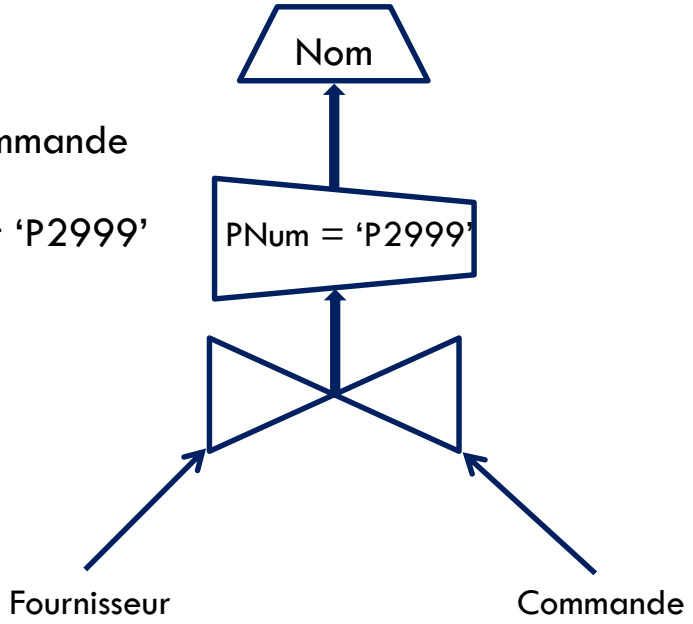
- Un arbre algébrique non optimisé de la requête :

Ordre d'exécution des opérateurs

$T_1 \leftarrow$ Joindre la relation Fournisseur avec Commande

$T_2 \leftarrow$ Sélectionner de T_1 les tuples du produit 'P2999'

$T_3 \leftarrow$ Projeter T_2 sur Nom



Optimisation des Requêtes

Exemple fixant la problématique

- **Evaluation de l'arbre non optimisé :**
  - 1ére étape : Jointure des relations Fournisseur et Commande
    - lecture des 10 000 tuples de commandes,
    - puis lecture de chacun des 100 fournisseurs 10 000 fois,
    - construction du résultat intermédiaire de jointure (10 000 tuples joints)

    (On n'est pas sûr que tous les résultats peuvent être gardés en mémoire)
  - 2éme étape : Restriction du résultat de l'étape précédente
    - les seuls tuples vérifiant PNum='P2999' seront gardés. Nous supposons qu'ils sont 50 et peuvent être gardés en mémoire
  - 3éme étape : Projection sur le Nom du Fournisseur
    - parcours des 50 tuples du résultat précédent afin de ne garder que Nom
- Perte de temps en effectuant la jointure en premier du fait des lectures répétées

Optimisation des Requêtes

Exemple fixant la problématique

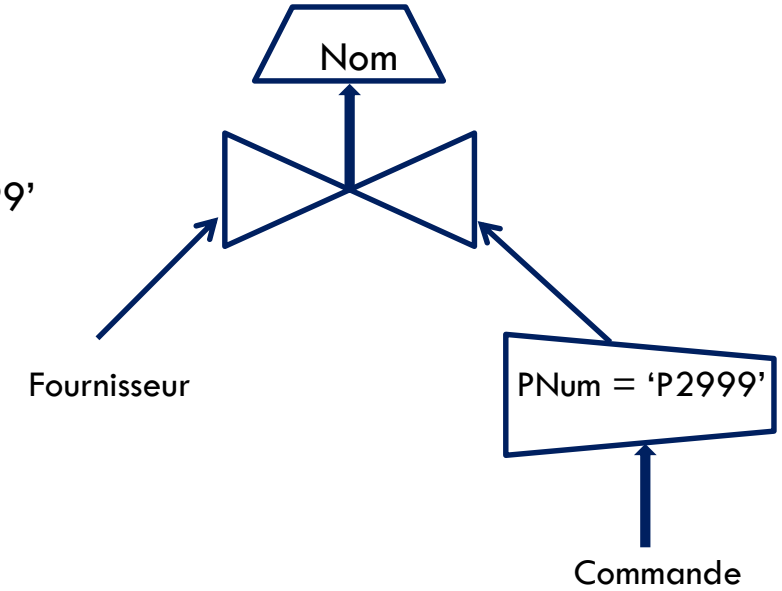
- Un arbre algébrique optimisé de la requête :

Ordre d'exécution des opérateurs

$T_1 \leftarrow$ Sélectionner les Commandes du produit 'P2999'

$T_2 \leftarrow$ Joindre T_1 avec la relation Fournisseur

$T_3 \leftarrow$ Projeter T_2 sur Nom



Optimisation des Requêtes

Exemple fixant la problématique

- **La deuxième stratégie est beaucoup plus efficace :**

  - 1ére étape : Effectuer la restriction de Commande à PNum= 'P2999'

    - lecture des 10 000 tuples de commande pour ne garder en mémoire qu'une table de 50 tuples

  - 2éme étape : Effectuer la jointure du résultat précédent avec Fournisseur

    - cette étape entraîne l'extraction des 100 fournisseurs. Le résultat contient 50 tuples

  - 3éme étape : Effectuer la projection

- La commutation de la jointure et la restriction réduit considérablement le nombre de transferts de la mémoire secondaire vers la mémoire centrale, donc implique un gain de temps et donc une réduction du coût

Optimisation des Requêtes

2 Questions

- Comment passe-t-on d'une requête SQL (définie dans un langage déclaratif) à un programme manipulant les données ?

- Comment optimise-t-on une requête afin de l'exécuter efficacement pour trouver un résultat correct ?

Optimisation des Requêtes

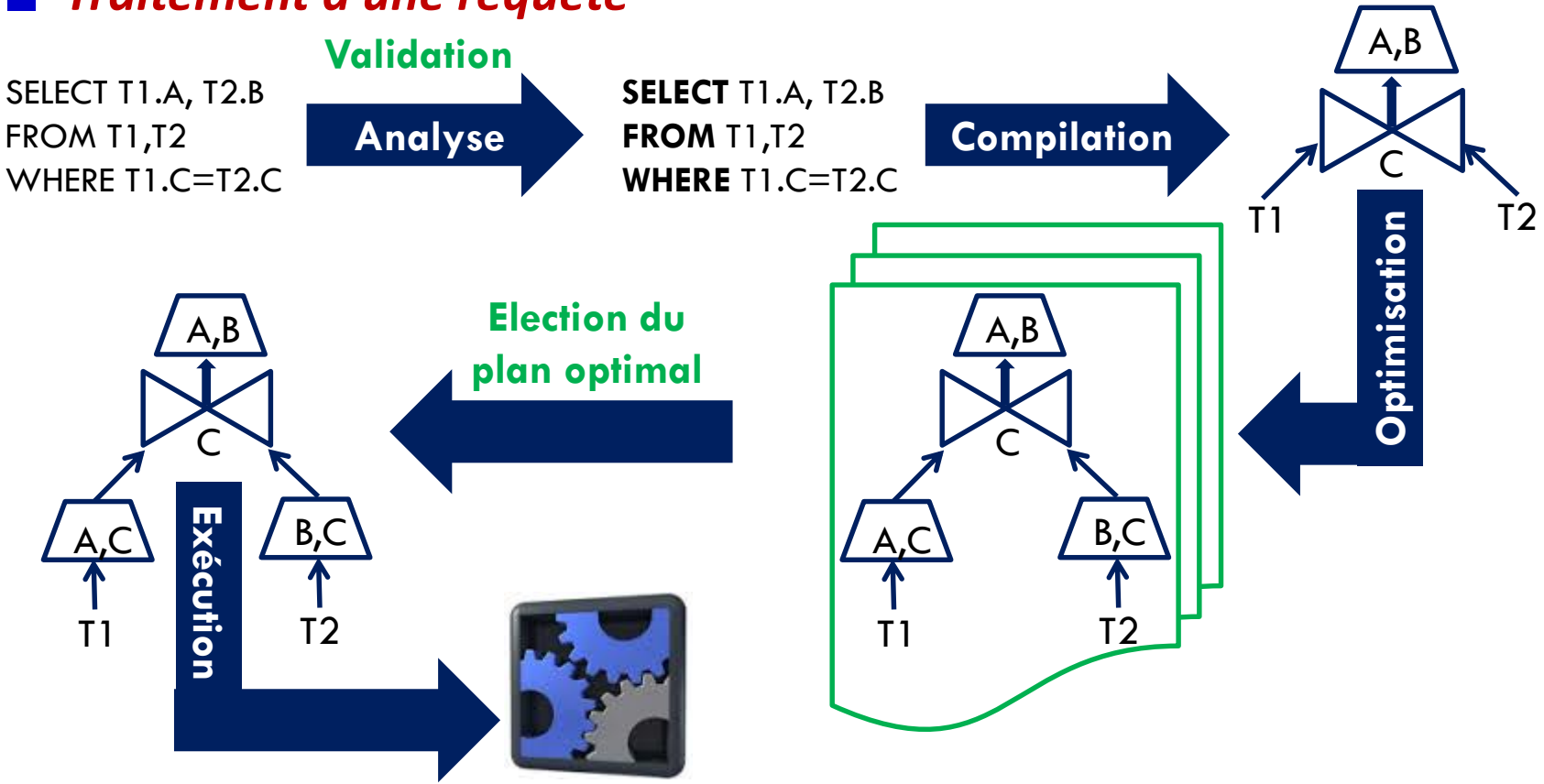
Traitement d'une requête

Toute requête SQL est traitée en quatre étapes :

- **Analyse**
  - Validation de la syntaxe de la requête
- **Compilation**
  - Introduction de la requête en arbre algébrique
- **Optimisation**
  - Recherche de l'arbre algébrique optimal
- **Exécution**
  - Exécution du programme correspondant à l'arbre algébrique optimal

Optimisation des Requêtes

- ***Traitement d'une requête***



Optimisation des Requêtes

Principe

- L'optimisation de requête est une opération dans laquelle plusieurs plans d'exécution d'une requête SQL sont examinés pour en sélectionner le meilleur
- L'estimation de leurs coûts dépend du temps d'exécution et du nombre de ressources utilisées pour y parvenir
- Les ressources coûteuses sont l'utilisation du processeur, la taille et la durée des tampons sur le disque dur, et les connexions entre les machines dans un système parallèle

Deux types d'optimisation :

- Plan d'exécution logique (PEL) qui dépend de l'algèbre relationnelle
- Plan d'exécution physique (PEP) qui tient compte des index et de la taille des données

Optimisation des Requêtes

Principe

- **2 problématiques importantes dans l'optimisation :**
  - Enumération des plans alternatifs pour une requête
  - Estimation des coûts de ces plans et choix de celui estimé être le moins cher

- **On doit se montrer pragmatique :**
  - Idéalement : Trouver le meilleur plan
  - Pratiquement : Éviter les pires plans !!

Optimisation des Requêtes

Stratégie générale d'optimisation

- Réécriture et transformation des requêtes en fonction des propriétés des opérateurs pour obtenir la meilleure séquence d'opérations

- Pour trouver les expressions équivalentes, on doit utiliser les équivalences algébriques

- Réduire le plus tôt la taille et le nombre de tuples manipulé
  - Tuples moins nombreux (réduction de cardinalité) : grâce à la sélection
  - Tuples plus petits (réduction de degré) : grâce à la projection

- Traiter en une seule fois les opérations de restriction et projection sur une même relation

Optimisation des Requêtes

Stratégie générale d'optimisation

- Combiner les sélections avec un produit cartésien pour aboutir à une jointure
- Pré-traiter les relations avant d'effectuer la jointure soit en créant des index soit en effectuant le tri (*CREATE INDEX i-Fourniss ON Fournisseur (Nom ASC)*)
- Mémoriser les sous-expressions communes
- Réordonner les jointures, en fonction de la taille des relations manipulées
- Choisir des algorithmes de jointures efficaces selon les relations manipulées

Optimisation des Requêtes

Règles de transformation des arbres

- **Règle 1 : Regroupement ou cascade des projections**

$\prod_{B1, B2, \ldots, Bp} (\prod_{A1, A2, \ldots, An} (R)) = \prod_{B1, B2, \ldots, Bp} (R)$
avec $\{Bj\} \subset \{Ai\}$

Exemple : $\prod_{Nom} (\prod_{[Nom, Prenom, Adresse, Anee\_Bac]} (Etudiant) = \prod_{Nom} (Etudiant)$

- **Règle 2 : Regroupement ou cascade des restrictions**

$\sigma_{P1(A1)} (\sigma_{P2(A2)} (R)) = \sigma_{[P1(A1) \wedge P2(A2)]} (R)$

Exemple : $\sigma_{age>25} (\sigma_{genre='F'} (Etudiant)) = \sigma_{[age>25 \wedge genre='F']} (Etudiant)$

Optimisation des Requêtes

Règles de transformation des arbres

- **Règle 5 : Commutation des restrictions et projections**

1ér Cas : l'argument de restriction fait partie des attributs de projection

$$\prod_{A1, \ldots, An} (\sigma_{P(Ai)} (R)) = \sigma_{P(Ai)} (\prod_{A1, \ldots..,Ai\ldots, An} (R))$$

2éme Cas : Sinon

$$\prod_{A1, \ldots, An} (\sigma_{P(Ai)} (R)) = \prod_{A1, \ldots, An} (\sigma_{P(Ai)} (\prod_{A1, \ldots, An,Ai} (R))$$

Optimisation des Requêtes

Règles de transformation des arbres

- **Règle 6 : Commutation des restrictions avec l'union**

$$\sigma_{P(Ai)} (R \cup T) = \sigma_{P(Ai)} (R) \cup \sigma_{P(Ai)} (T)$$

- **Règle 7 : Commutation des restrictions avec la différence**

$$\sigma_{P(Ai)} (\text{Minus } (R,T)) = \text{Minus } (\sigma_{P(Ai)} (R), \sigma_{P(Ai)} (T))$$

- **Règle 8 : Commutation de la jointure et l'union**

$$(R \cup T) \bowtie S = (R \bowtie S) \cup (T \bowtie S)$$

Optimisation des Requêtes

Règles de transformation des arbres

- **Règle 12 : Commutation des restrictions et produit cartésien**

1ér Cas : la restriction porte sur l'argument d'une seule relation : les deux relations n'ont pas d'attributs communs : R (A_1,….,A_i,….A_N) et T (B_1,….,B_j,…..B_N)

$\sigma_{P(Ai)} (R \otimes T) = \sigma_{P(Ai)} (R) \otimes T$

2éme Cas : la restriction porte sur deux arguments respectifs de R et T, les deux relations n'ont pas d'attributs communs : R (A_1,….,A_i,….A_N) et T (B_1,….,B_j,…..B_N)

$\sigma_{[P(Ai)\wedge P(Bj)]} (R \otimes T) = \sigma_{P(Ai)} (R) \otimes \sigma_{P(Bj)} (T)$

3éme Cas : la restriction porte sur deux arguments respectifs de R et T, les deux relations ont un attribut commun : R (A_1,….,A_i ,…,X,….A_N) et T (B_1,….,X,…..B_N)

$\sigma_{[P(Ai)\wedge P(X)]} (R \otimes T) = \sigma_{P(X)} (\sigma_{P(Ai)} (R) \otimes T)$

Optimisation des Requêtes

Règles de transformation des arbres

- **Règle 13 : Commutation de la restriction à la jointure**

  $R(A_1,\ldots,A_i,\ldots,X,\ldots A_N)$ et $T(B_1,\ldots,X,\ldots B_N)$

  $\sigma_{p(Ai)}(R \bowtie T) = \sigma_{p(Ai)}(R) \bowtie S$

  (1ér exemple fixant la problématique)

- **Règle 14 : Commutation de la projection à la jointure**

  Soient $R(A_1,\ldots\ldots,X,\ldots\ldots A_N)$ et $T(B_1,\ldots\ldots,X,\ldots\ldots B_N)$

  $\prod_{[A1,\ldots..,\ Ai,B1,\ldots..Bj]}(R \bowtie T) = \prod_{[A1,\ldots..,\ Ai,B1,\ldots..Bj]}(\prod_{[A1,\ \ldots,\ Ai,\ X]}(R) \bowtie \prod_{[B1,\ \ldots,\ Bj,\ X]}(T))$

  Application de la règle *: Réduire le plus tôt la taille et le nombre de tuples manipulés en exécutant les sélections et les projections aussitôt que possible*