Les langages de manipulation de données relationnelles

1. Introduction

Un langage de manipulation de données se compose d'un ensemble de commandes permettant d'interroger et de modifier une base de données. La modification inclut l'insertion, la mise à jour et la suppression.

Un langage de manipulation de données n'est pas suffisant à lui seul ; généralement incorporable dans un langage de programmation classique appelé langage hôte afin de réaliser des transactions programmées.
Ainsi, la plupart des SGBD disposent d'un langage de manipulation de données externe conversationnel et proposent une intégration de ce langage dans un langage hôte.

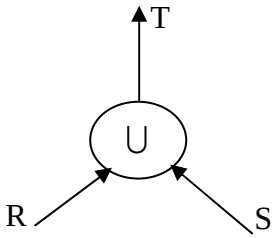
On peut distinguer trois grandes classes de langages : les langages basés sur l'algèbre relationnelle, le langage d'IBM SQL, les langages basés sur la logique des prédicats.

2. L'algèbre relationnelle

L'algèbre relationnelle a été inventée par CODD comme une collection d'opérations formelles sur les relations.

L'union

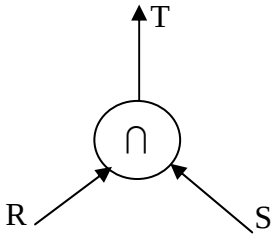
L'union de deux relations R et S de même schéma est une relation T de même schéma contenant l'ensemble des tuples appartenant à R ou S ou aux deux relations.
On notera cette opération : $T=R \cup S$ ou graphiquement :



📖 La réunion de deux relations R et S de même structure, est la relation T de même structure, contenant les tuples appartenant dans R ou dans S.

L'intersection

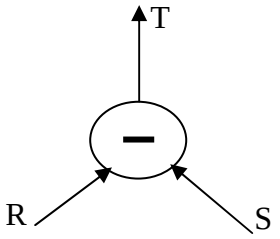
L'intersection de deux relations R et S de même schéma est une relation T de même schéma contenant les tuples appartenant à la fois à R et S.
On notera cette opération : $T=R \cap S$ ou graphiquement :



La différence

La différence de deux relations R et S de même schéma (dans l'ordre R-S) est une relation T de même schéma contenant les tuples appartenant à R et n'appartenant pas à S.

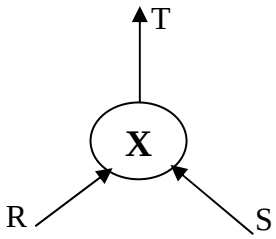
On notera cette opération : T=R-S ou graphiquement :



Le produit cartésien

Le produit cartésien de deux relations (de schéma quelconque)R et S, est une relation ayant pour attribut la concaténation des ceux de R et S et dont les tuples sont toutes les concaténations d'un tuple de R à un tuple de S.

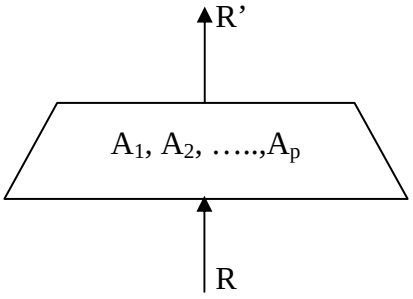
On notera cette opération : T=RxS ou graphiquement :



La projection

La projection d'une relation de schéma R(A_1, A_2, …..,A_n) sur les attributs A_1, A_2, …..,A_p est une relation R' de schéma R'(A_1, A_2, …..,A_p) dont les tuples sont obtenus par élimination des valeurs des attributs de R n'appartenant pas à R' et par suppression des tuples en double.

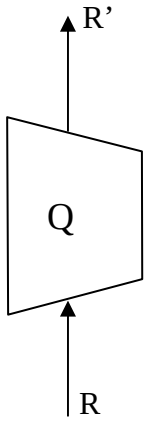
On notera cette opération : R'= Π (A_1, A_2, …..,A_p)[R] ou graphiquement :



La sélection (restriction)

L'opération de sélection consiste de sélectionner dans une relation, l'ensemble des tuples satisfaisant une condition donnée appelée 'qualification'.

On notera cette opération : R'=R[Q] ou graphiquement :



📖 Le résultat d'une sélection sur une relation est une relation de meme structure contenant seulement les tuples vérifiant certaines propriétés, dites 'critères de sélection'

Les critères de sélections sont : <attribut> <opérateur> <valeur>

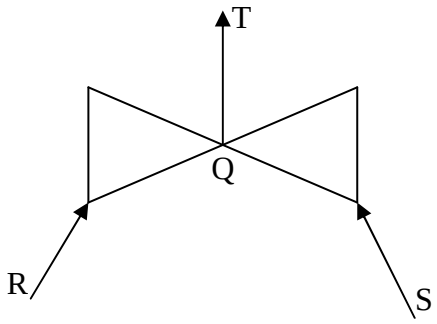
Les opérateurs sont : <, ≤ , >, ≥, <>, AND, OR, NOT

Exemple : ville = 'Béchar'

La jointure

La jointure de deux relations R et S selon une condition Q est l'ensemble des tuples du produit cartésien RxS satisfaisant la condition Q.

On notera cette opération : T=RxS[Q] ou T=R$\bowtie_Q$S ou graphiquement :



On appelle **équi-jointure** une jointure où la sélection se fait selon un critère d'égalité entre un attribut de la première relation et un attribut de la seconde relation.

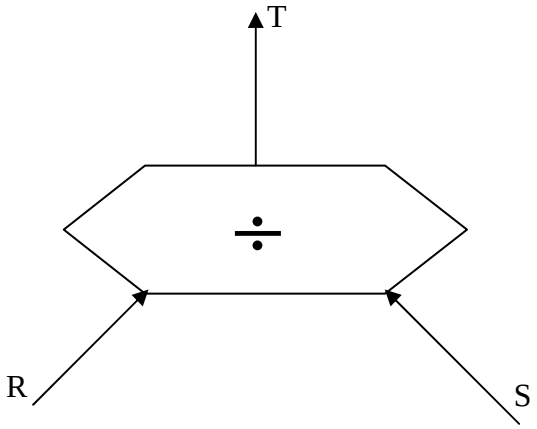
On appelle **Jointure Naturelle**, une équi-jointure sur des attributs portant le même nom dans la première relation et la deuxième relation. La jointure naturelle est l'opérateur de jointure le plus caractéristique de l'algèbre relationnelle.

.

La division

Le quotient de la relation R de schéma $R(A_1, A_2,\ldots,A_n)$ par la sous-relation S de schéma $S(A_{p+1},\ldots,A_n)$ est la relation T de schéma $T(A_1, A_2,\ldots,A_p)$ formée de tous les tuples qui concaténés à chacun des tuples de S donne toujours un tuple de R.

On notera cette opération : DIV(R,S) ou T=R ÷ S ou graphiquement :



📖 La division de A(a1, a2, a3, a4) par B(a1, a2) est la relation C(a3, a4), telleque le produit cartésien de B par C est contenu dans A

3. Composition d'opérations

Il est possible de composer la plupart des questions que l'on peut poser à une base de données relationnelle, avec les opérations de base successivement enchaînées sur des relations. En effet, les questions peuvent être exprimées à l'aide de succession des opérations : union, différence, jointure, restriction, projection, …etc. La représentation graphique de ces opérations permet de composer des arbres d'opérations relationnelles (voir figure.IV.1).

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.

Plusieurs arbres représentent une meme question, selon l'ordre choisi les opérations. Une méthode de génération simple d'un arbre consiste à prendre les prédicats de qualification dans l'ordre où ils apparaissent et à leur associer l'opération relationnelle correspondante, puis à terminer par une projection finale pour obtenir le résultat.

Par exemple, soit la base de données composées des relations suivantes :
etudiant (N°etud, nom, prénom)
module (N°mod, intitule)
examen (N°etud, N°mod, note)

Pour répondre à la question suivante ‘nom et prénom de tous les étudiants ayant obtenu une note supérieure à 10 dans le module de numéro N’, on peut concevoir l’arbre des opérations suivant :

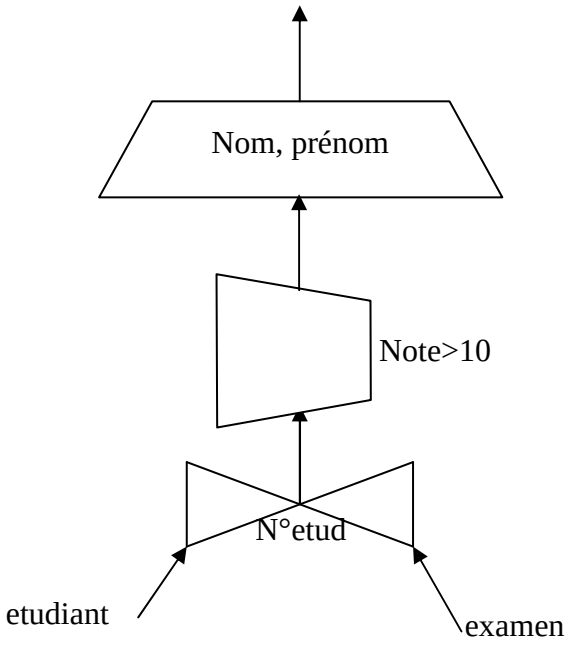


Fig.IV.1.exemple d’arbre d’opérations relationnelles

A partir d’un arbre algébrique, il est possible de générer un plan d’exécution en parcourant l’arbre, des feuilles vers la racine. Une opération peut être exécutée dès que ses opérandes sont disponibles ; ainsi, si l’opération a n’utilise pas les résultats de l’opération b, a et b peuvent être exécutées en parallèle.

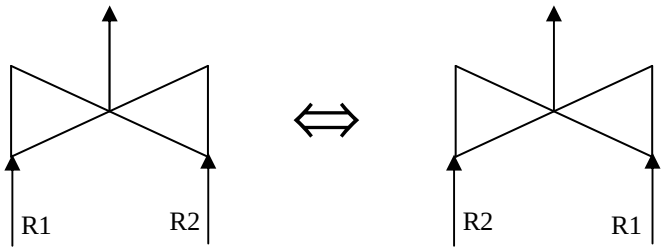
On cherche bien sur à optimiser les temps de réponse, donc à minimiser le temps nécessaire à l’exécution du plan. Le problème est donc de générer un plan optimal. Pour cela, il faut optimiser simultanément :

- Le nombre d’opérations d’entrées-sorties
- Le parallélisme entre les entrées-sorties
- La taille des tampons nécessaires à l’exécution
- Le temps unité central nécessaire

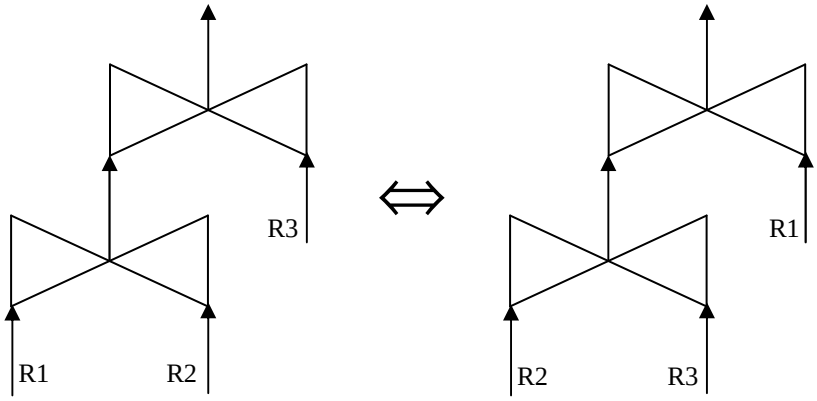
L’optimisation effectuée dépend essentiellement de l’ordre des opérations apparaissent dans l’arbre algébrique utilisé. Il est donc essentiel d’établir des règles permettant de générer, à partir d’un arbre initial, tous les arbres possibles afin de pouvoir ensuite choisir celui conduisant au meilleur plan. En fait, le nombre d’arbres étant très grand, on est amené à définir des heuristiques pour déterminer un arbre proche de l’optimum.

4. Règles de transformation des arbres

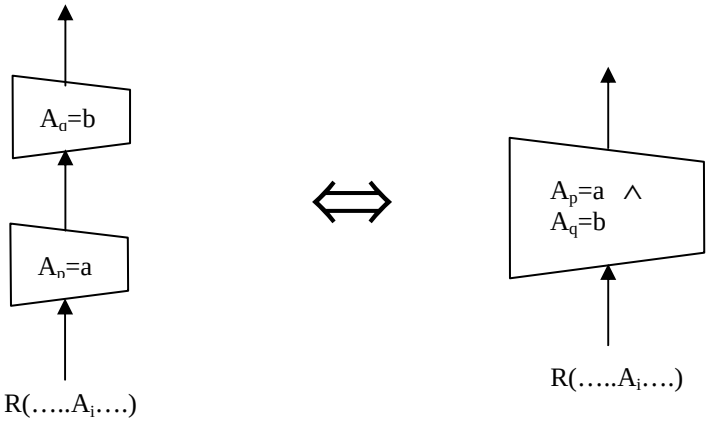
Règle1: Commutativité des jointures



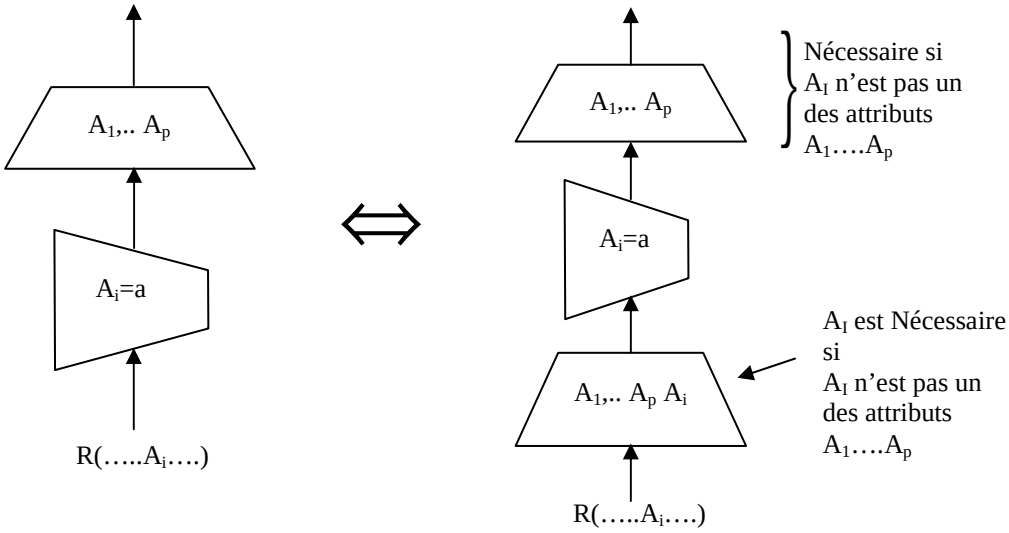
Règle2: Associativité des jointures:



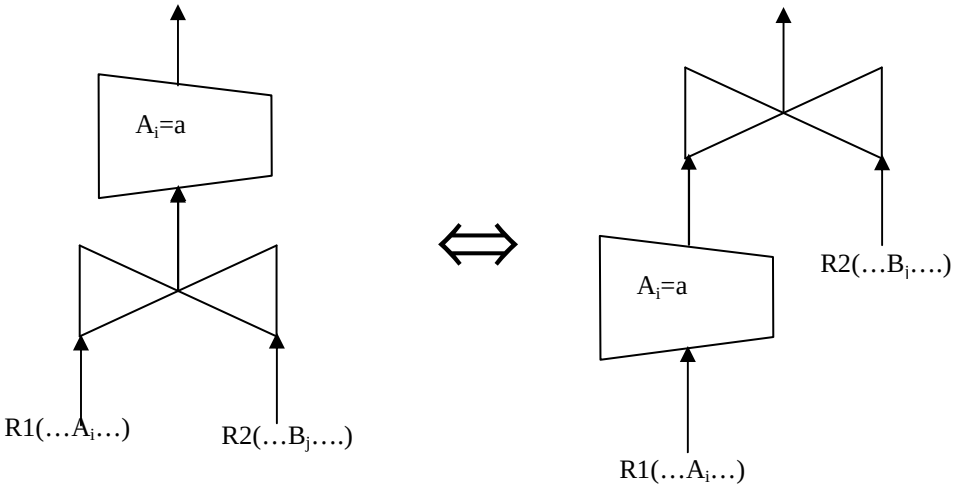
Règle3 : Regroupement des restrictions :



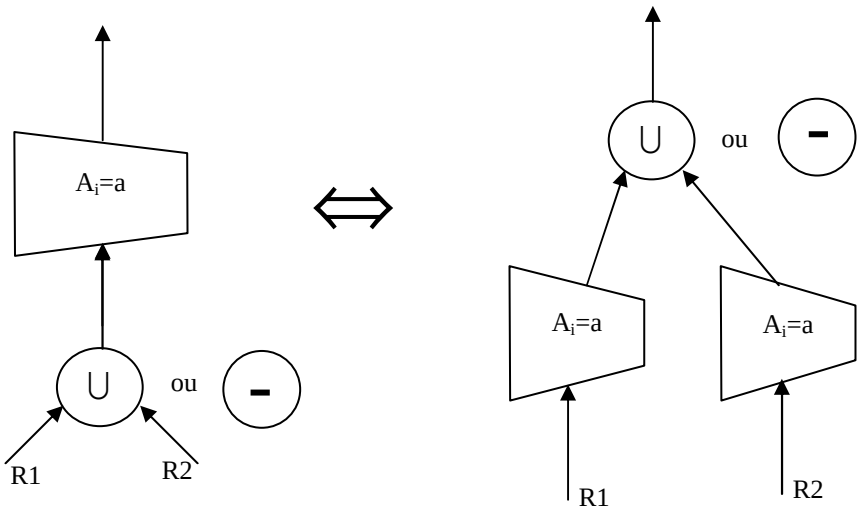
Règle4 : Commutation des restrictions et projections



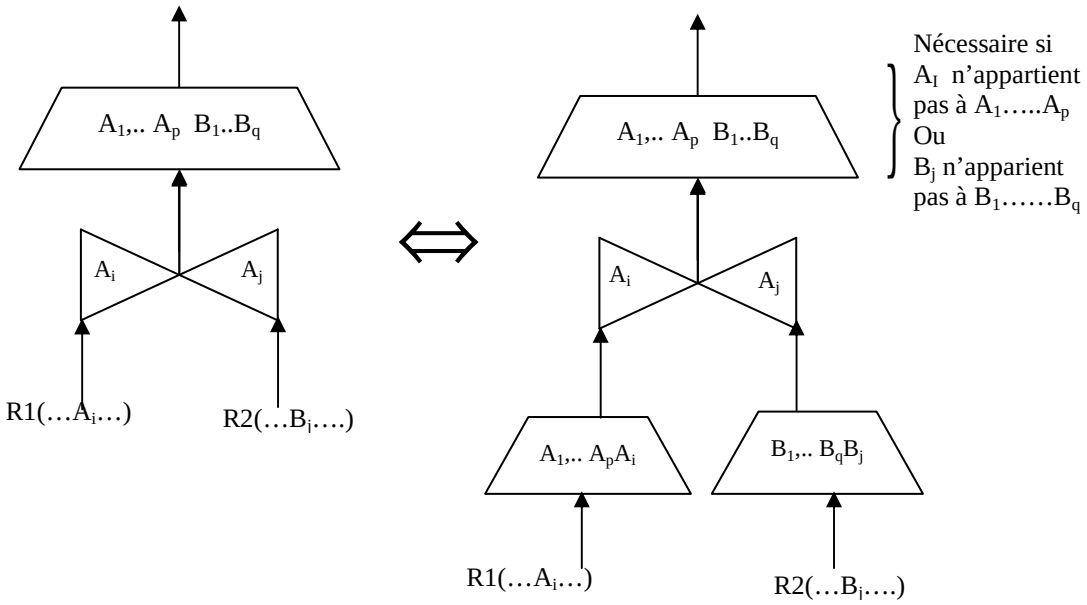
Règle5 : Commutation des restrictions et jointures :



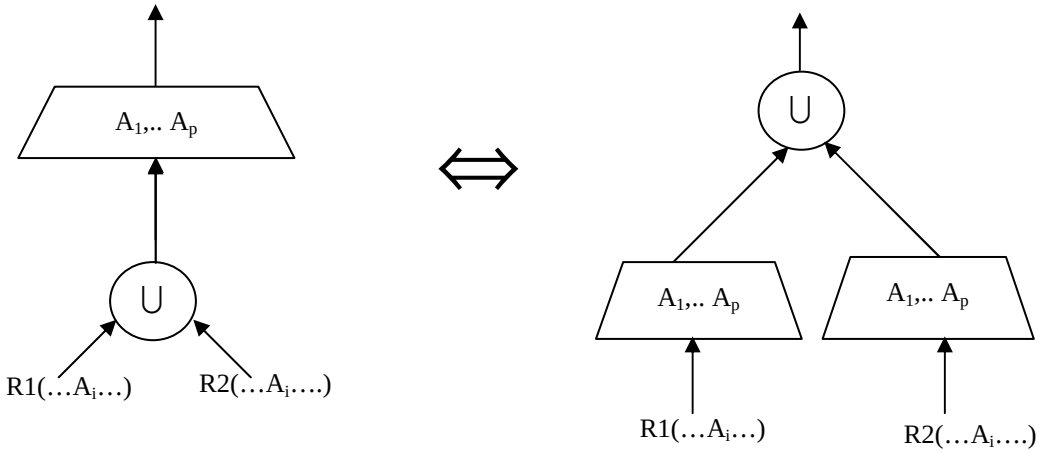
Règle6 : Commutation des restrictions et unions ou des restrictions et différences :



Règle7 : Commutation des projections et jointures :



Règle8 : Commutation des projections et unions :



5. Optimisation par descendante des opérations unaires

Une première optimisation simple consiste à exécuter tout d'abord les opérations unaires (restriction, projection) puis les opérations binaires. En effet, les opérations unaires sont des réducteurs de la taille des relations, alors qu'il n'en est pas ainsi de certaines opérations binaires qui ont tendance à accroître la taille des résultats par rapport aux relations arguments. Aussi, afin de ne considérer que les arbres à flux de données minimum et de réduire ainsi le nombre d'entrées-sorties à effectuer, on est conduit à descendre les restrictions et projections. De plus, quand deux projections successives portent sur une meme relation, il est nécessaire de les regrouper afin d'éviter un double accès à la relation.

Les principes précédents conduisent à l'algorithme d'optimisation suivant :

1. Séparer les restrictions comportant plusieurs prédicats à l'aide de la règle 3
2. Descendre les restrictions aussi bas que possible à l'aide des règles 4, 5, et 6
3. Regrouper les restrictions successives portant sur une meme relation.
4. Descendre les projections aussi bas que possible à l'aide des règles 7 et 8
5. Regrouper les projections successives en conservant les attributs restants et éliminer d'éventuelles projections inutiles qui auraient pu apparaître (projection sur tous les attributs d'une relation)

En règle générale, une restriction suivie par une projection sont exécutées simultanément par un même opérateur de sélection (restriction + projection) afin d'éviter plusieurs passes sur une même relation. Il en est de même pour une jointure suivie par une projection.