# INSIA
# Bases de données
# SIGL 3
# Optimisation – 1 : arbres algébriques

**Bertrand LIAUDET**

## SOMMAIRE

Première édition : février 2008
Deuxième édition : septembre 2009

# INTRODUCTION

## Bibliographie

Gardarin. Bases de données. Eyrolles 2005. §10 – pp. 301 à 350.

Darmaillac, Rigaux. Maîtriser MySQL 5. O'Reilly 2005. §11 – pp. 283 à 318.

Dubois, Hinz, Pedersen. MySQL-Guide officiel. Campus press 2004. §13 – pp. 433 à 475.

MySQL 5-Guide de l'administrateur. Campus press 2005. §6 – pp. 447 à 510.

Harrison. MySQL Stored Procedure. O'Reilly 2006. Part IV – pp. 421 à 582.

## Généralités sur l'optimisation des requêtes

### Objectif de l'optimisation

L'objectif de l'optimisation est d'accélérer de vitesse de traitement des requêtes.

### Public concerné par l'optimisation des requêtes

**L'utilisateur final :** il peut avoir des exigences de temps réponse tel que si elles ne sont pas prises en compte, l'application ne répond plus aux besoins de l'utilisateur et devient donc inutile.

**Le programmeur d'applications** : il intervient au niveau de l'application, pendant sa réalisation. Mieux connaître les possibilités du SGBD et de l'algèbre relationnelle permet de mieux concevoir la BD et de mieux donner satisfaction aux utilisateurs.

**L'administrateur BD :** il intervient au niveau du serveur, pendant la vie de l'application. Mieux connaître les possibilités du SGBD et de l'algèbre relationnelle permet de mieux administrer le serveur de la BD et de mieux donner satisfaction aux utilisateurs.

**Le programmeur de SGBD :** il développe l'application SGBD (le serveur). C'est lui qui réalise les outils d'optimisation du SGBD. Ces outils font partie de ce qui fait l'efficacité du SGBD.

### Typologie des requêtes d'un point de vue général d'optimisation

#### Informatique de gestion (au sens large) *vs.* informatique décisionnelle

**Les requêtes de l'informatique de gestion** (SI au sens large) concernent autant voir plus la création-modification-suppression (CMS : insert, update, delete) des données que leur interrogation. L'interrogation des données est souvent assez élémentaire.

**Les requêtes de l'informatique décisionnelle** concernent essentiellement l'interrogation des données. Ce sont des requêtes complexes, multi-tables, le plus souvent avec des statistiques et des agrégats, souvent écrites en PL-SQL.

**Requêtes statiques (compilées) *vs.* requêtes *ad hoc* (interprétées, dynamiques)**

**Les requêtes statiques** sont les requêtes programmées une fois pour toutes dans une application. Elles seront probablement utilisées plusieurs fois.

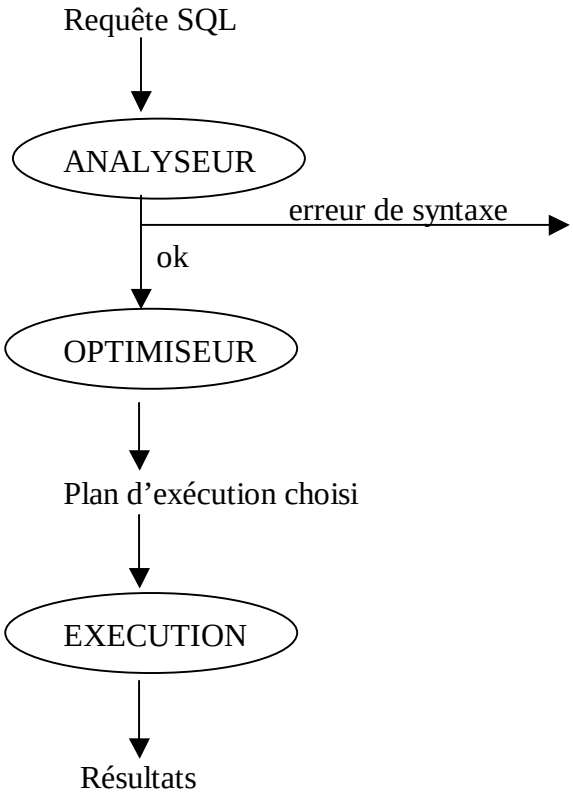
**Les requêtes dynamiques** sont les requêtes construites directement via une calculette SQL avec ou sans interface graphique : mysql sous MySQL, psql sous PostgreSQL, SQL-Plus sous Oracle, SQL Server Management Studio Express sous SQL-Server, etc. Elles sont probablement utilisées une seule fois.

L'optimisation concerne surtout les requêtes statiques. Le but est d'être le plus efficace possible, au moins pour les questions les plus fréquentes.

## L'optimiseur et optimisation

### L'optimiseur

L'optimiseur est un composant du SGBD. Son rôle est de transformer la requête SQL en opérations dites « de bas niveau » réalisant efficacement l'accès aux données.




En majuscule et dans un ovale : les traitements

En minuscules : les données

### Principe général de l'optimisation des requêtes par l'optimiseur

L'optimisation consiste à passer d'une requête exprimée dans un **langage source** (le **SQL** dans le modèle relationnel) à une requête exprimée par un ensemble d'**opérations plus élémentaires** exprimées dans un **langage cible** (opérations dites « de bas niveau »).

Ces opérations élémentaires dépendent particulièrement :

- de la requête SQL initiale
- des contraintes d'intégrité
- des index
- des statistiques de la BD : taille des tables et connaissances sur les résultats des restrictions.

**Les plans d'exécution**

Le plan d'exécution d'une requête est une suite possible d'opérations de bas niveau qui permettent de réaliser cette requête.

Il peut exister plusieurs plans d'exécution pour une requête donnée.

C'est le résultat de l'optimisation.

**Le plan d'exécution choisi et le calcul du coût**

L'optimisation consiste à choisir le meilleur plan d'exécution (le plus perfomant). Le choix se fait par un calcul du coût (du temps de traitement).

**Commande EXPLAIN**

La commande EXPLAIN permet de consulter le plan d'exécution mis en œuvre par le SGBD pour une requête donné.

En général sa syntaxe se résume à : EXPLAIN requête ;

## 3 approches de l'optimisation

Il y a **3 approches** de l'optimisation des requêtes :

**L'optimisation sémantique**

- L'optimisation sémantique consiste essentiellement à rechercher une contradiction dans les restrictions qui conduirait à obtenir 0 tuple en réponse à la question ou à une partie de la question.

  La contradiction peut apparaître directement dans la question. C'est le cas si deux restrictions sont contradictoires : (degré > 13 et degré <10) ou encore (région = R1 et région = R2).

  La contradiction peut apparaître quand on croise une restriction avec une contrainte d'intégrité de la BD. Par exemple, une contrainte de valeur peut préciser : (degré <15) et une question demander (degré >15).

  Si une telle contradiction est découverte alors la table concernée ainsi que toutes celles qui lui sont jointes n'auront pas de tuples.
- L'optimisation sémantique peut aussi traiter certains points particuliers : par exemple remplacer un like par un « = » si possible (pas de caractères spéciaux dans la restriction).
- L'optimisation sémantique consiste donc à remplacer une formule par une autre à l'intérieur du select (une requête par la valeur null, un like par un « = », etc.) sans changer la structure de la requête.
- Cette optimisation est totalement mécanique.

**L'optimisation syntaxique**

- Elle prend uniquement en compte les **propriétés de l'algèbre relationnelle**. Autrement dit, l'ordre des opérations sera changé pour obtenir le meilleur résultat en terme de performance. Par exemple, une table sera utilisée avant une autre, une restriction sera faite avant une autre, une jointure sera faite avant une autre, etc.
- L'optimisation syntaxique consiste donc à réorganiser les étapes des différentes opérations relationnelles élémentaires d'une requête. Elle part d'une requête en SQL pour arriver à une autre requête en SQL consituées de plusieurs étapes qu'on peut formaliser à travers des vues.
- Cette optimisation est totalement mécanique. Toutefois, plusieurs résultats sont possibles.

**L'optimisation physique**

- Elle prend en compte les index, les statistiques (taille des tables et sélectivité des restrictions), la gestion de la mémoire cache et les principes algorithmiques classiques de l'accès au données pour décomposer la requêtes en étapes dont les traitements relèvent de la programmation impérative classiquee (boucles et tests du langage C).
- L'organisation physique consiste donc à réorganiser les étapes des différentes opérations relationnelles en étapes de programmation impérative.
- Cette optimisation n'est pas totalement mécanique.

**Optimisation et heuristique**

- **Les optimisation sémantique et syntaxique** sont des étapes algébriques et logiques qui relèvent d'une méthode déductive.
- **L'optimisation physique** et de **choix du plan d'exécution** relèvent d'une méthode heuristique : méthode qui procède par hypothèses provisoires et par évaluations successives.

L'optimisation physique est liée à l'implémentation physique de la BD, mais aussi aux algorithmes de calcul qui sont utilisés. Certains algorithmes sont plus efficaces que d'autres. La qualité de ces algorithmes est un des atouts du SGBD (comme la qualité des algorithmes de recherche est un des atouts des moteurs de recherche).

**Optimisation physique et notion de METABASE**

L'optimisation physique prend en compte les statistiques de la BD, c'est-à-dire la taille des tables et la sélectivité des restrictions (le nombre de tuples pour une valeur donnée d'un attribut).

Ces informations sont des données sur les données : d'où la notion de **METABASE**.

Le SGBD gère automatiquement la mise à jour de la métabase à l'occasion des requêtes qui lui sont envoyées. Toutefois, la mesure des sélectivités n'est pas faisable à chaque requête sous peine de ralentir son exécution et donc de ne pas atteindre l'objectif d'optimisation. A noter toutefois que, statistiquement, l'ensemble des valeurs possibles pour un attribut est le plus souvent relativement fixe.

Cette mise à jour peut aussi être forcée par l'administrateur de la BD. Des scripts peuvent être écrits pour permettre une mise à jour automatique à heure fixe.

Les données de la métabase participent au choix du plan d'exécution en permettant de faire un calcul de coût le plus réaliste possible.

## Plan d'exécution

Un plan d'exécution d'une requête est une suite possible d'opérations de bas niveau (dans un langage cible) qui permettent de réaliser cette requête.

Une requête peut donner lieu à plusieurs plans d'exécution.

## Bilan de l'optimisation : le plan d'exécution choisi et calcul de coût

Le choix du plan d'exécution optimal se fera par un calcul de coût, c'est-à-dire de la performance.

Le calcul de coût consiste à calculer le temps de traitement maximum des opérations élémentaires et de leur succession.

Dans le calcul de coût (donc dans la performance de la requête), interviennent :

- Le nombre d'entrée-sortie (accès disque)
- Le temps de calcul des opérations élémentaires
- La taille des buffers requis

## Optimisation et méthodes (algorithmes) d'accès aux données

Le critère principal du calcul de coût concerne l'accès aux données.

Il y a deux grands types de méthodes d'accès aux données.

### les méthodes par indexation

On retrouve la méthode par indexation dans les SGBD-R. Elle utilile le principe alogorithique de la recherche dichotomique.

### les méthodes par hachage

Les méthodes par hachage consistent à utiliser une fonction de calcul qui, appliquée à la clé, détermine l'adresse relative d'un enregistrement. On ne les abordera pas dans ce cours.

## La question de la performance : l'évolution des SGBD en terme de rapidité

### Capacité de traitement

- **Années 70 :** quelques requêtes par seconde
- **Aujourd'hui :** plusieurs milliers de transactions par seconde

### 4 causes à l'amélioration des performances des SGBD

- Augmentation de la vitesse du processeur
- Amélioration de la production des plans d'exécution et le leur choix
- Optimisation des méthodes d'accès aux données (les algorithmes de calcul)
- Utilisation de mémoire cache dans les méthodes d'accès aux données

**Temps d'entrée-sortie : $10^{-2}$ seconde**

A noter que les temps d'entrée-sortie disque restent à peu près constant : de l'ordre de la **dizaine de millisecondes,** d'où l'intérêt de les limiter par l'optimisation.

# OPTIMISATION SYNTAXIQUE

L'optimisation logique met en oeuvre les **arbres algébriques** et les **règles de transformations** qui leur sont associées.

Une fois ces notions présentées, on pourra proposer un **algorithme d'optimisation**.

## 1. L'arbre algébrique

**Exemple traité**

On travaille sur l'exemple suivant : (Gardarin, p. 303)

**Schéma de la BD**

Buveurs (**NB**, Nom, Prénom, Type)

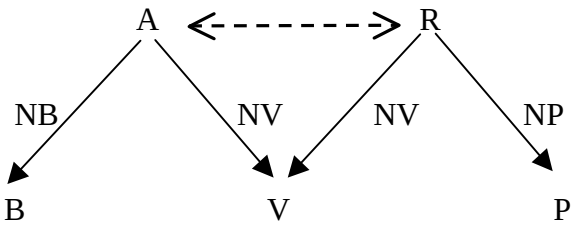
Vins (**NV**, Cru, Millésime, Degré)

Producteurs (**NP**, Nom, Région)

Abuser (**#NB, Date**, Quantité, #NV)

Produire (**#NP, #NV**)

**Graphe des tables**

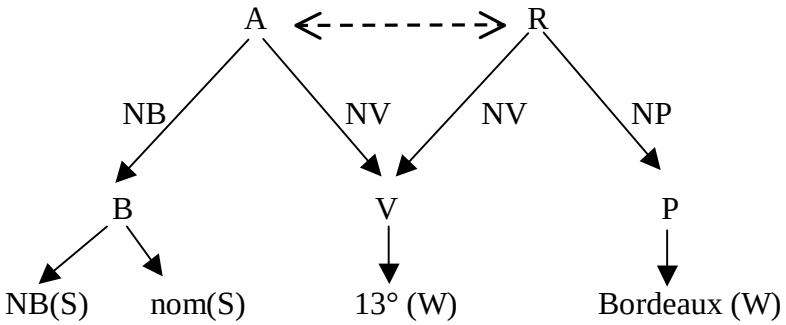



Rappelons que la flèche à double sens en pointillé symbolise une jointure artificielle.

**Question traitée**

On cherche tous les buveurs ayant bu un Bordeaux de degré supérieur ou égal à 13.

**Graphe de la question**




Les attributs « S » sont les attributs pour le Select, donc les attributs en sortie de la question.

Les attributs « W » sont les attributs pour le Where, donc les attributs en entrée de la question.

**Clé primaire du graphe de la question**

La clé primaire du graphe d'une question (hors distinct) est constituée de la concaténation des clés primaires des tables racines du graphe : #NB, Date pour A et #P, #NV pour P.

La clé primaire est donc : #NB, Date, #P, #NV

**Réponse SQL**

```
Select distinct B.NB, B.Nom
From Buveurs B, Abuser A, Vins V, Produire R, Producteurs P
Where B.NB = A.NB And A.NV =  V.NV
And P.NP = R.NP And R.NV = V.NV
And P.Region = "Bordelais"
And V.Degre >=13;
```

**Clé primaire de la question**

La question produit des buveurs : la clé primaire sera donc #NB

Etant donné que ce n'est pas la clé primaire du graphe de la question, il faut mettre un distinct pour éliminer les doublons.

## Rappel de vocabulaire

**Une restriction spécifique** est une restriction mono-table.

**Une restriction de jointure** est une restriction qui met deux tables en jeu.

**Une restriction d'agrégat** est une restriction qui se fait sur la table résultant de l'agrégat.

**Une jointure** c'est un produit cartésien associé à une restriction de jointure.

**Une jointure naturelle** c'est une jointure avec une restriction de jointure naturelle, c'est-à-dire une restriction mettant en jeu une clé étrangère et la clé primaire correspondante.

**Table maître et table jointe :** en cas de jointure naturelle, on distingue entre la table maître, celle dont la clé étrangère intervient dans la jointure et dont la clé primaire sera la clé primaire de la table résultat, et la table jointe, celle dont la clé primaire intervient dans la jointure.

**Une jointure artificielle** c'est une jointure avec une restriction de jointure artificielle, c'est-à-dire une restriction de jointure qui n'est pas une restriction de jointure naturelle.

## Représentation textuelle des opérateurs de l'algèbre relationnelle

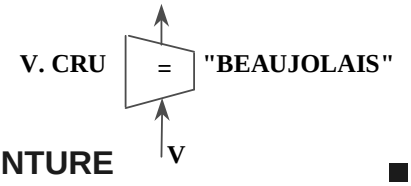
On reprend le formalisme textuel de l'algèbre relationnelle

| **Opération** | **Formalisme** |
|---|---|
| **Projection** | P (table ; liste d'attributs) |
| **Restriction** | R (table ; liste de restrictions) |
| **Produit cartésien** | PC (table1, table 2) |
| **Jointure naturelle** | JN (table maître, table jointe ; TM.NTJ=TJ.NTJ) |
| **Jointure artificielle** | = PC + Restriction |

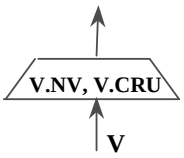
| | |
|---|---|
| **Union** | Union (table1, table2) |
| **Différence** | Diff (table1, table2) |
| **Intersection** | Inter (table1, table2) |
| **Tri** | Tri (table; liste d'attributs) |
| **Agrégat** | Agreg (table ; attributs de regroupement ; attributs de statistique) |
| **Fonction de groupe** | FG (table ; attributs de statistique) |

## Représentation graphique des opérateurs de l'algèbre relationnelle dans les arbres algébriques

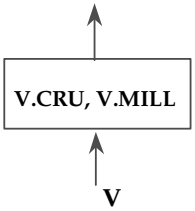
■ **RESTRICTION**



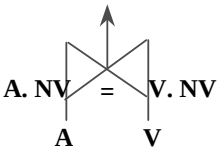

■ **PROJECTION**



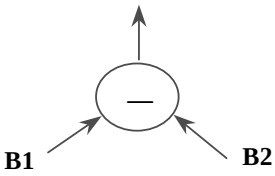

■ **TRI**



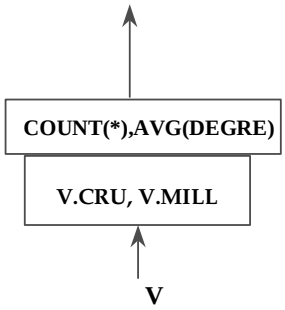

■ **JOINTURE**



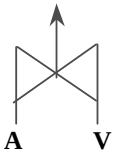

■ **DIFFERENCE**



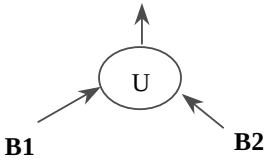

■ **AGREGAT**




■ **PRODUIT CARTESIEN**




■ **UNION**




**Cas du distinct**

Il n'y a pas de distinct en algèbre relationnelle car par définition une table ne contient pas de doublon et toute opération de l'algèbre relationnelle produit des tables sans doublons.

## Arbre algébrique

**Présentation**

Un arbre algébrique est la représentation d'une requête SQL sous la forme d'un arbre.

**Arbre algébrique = arbre relationnel = arbre de traitement**

**Les feuilles** de l'arbre représentent les tables de départ.

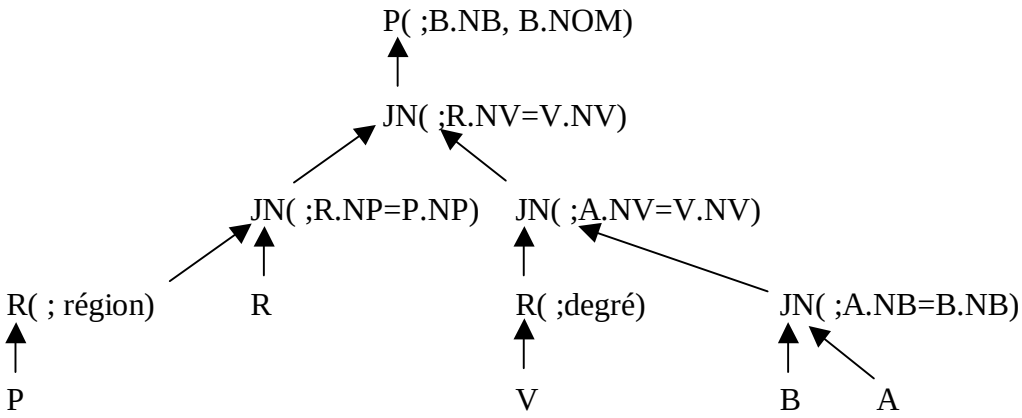
**La racine** de l'arbre représente la table résultat.

**Tous les autres nœuds** de l'arbre sont des opérateurs de l'algèbre relationnelle.

Un arbre algébrique correspond à une décomposition de la requête en opérateurs élémentaires avec introduction de tables intermédiaires : c'est donc l'équivalent d'une réécriture de la requête avec des vues.

**Arbre algébrique de l'exemple traité**

Tous les buveurs ayant bu un Bordeaux de degré supérieur ou égal à 13




**Notion d'arbres algébriques équivalent**

**Notion de requêtes équivalentes**

Pour une même question, il peut exister plusieurs requêtes SQL équivalentes à la question.

Deux requêtes sont équivalentes quand elles donnent toujours la même réponse quel que soit le jeu de données auquel elles s'appliquent.

- ***Exemple 1 de requêtes équivalentes :***

| | | |
|---|---|---|
| Select distinct NB, nom | ⇔ | Select NB, nom |
| from Buveurs | | from Buveurs |

Ces deux requêtes sont équivalentes car la clé primaire du graphe de la question et celle de la question sont identiques.

- ***Exemple 2 de requêtes équivalentes:***

| | | |
|---|---|---|
| Select distinct NB, nom | **NOT** ⇔ | Select NB, nom |
| from Buveurs, Abus | | from Buveurs, Abus |
| where B.NB = A.NB | | where B.NB = A.NB |

Les deux requêtes ne sont pas équivalentes car la clé primaire du graphe de la question n'est pas celle de la question.

**Notion d'arbres algébriques équivalent**

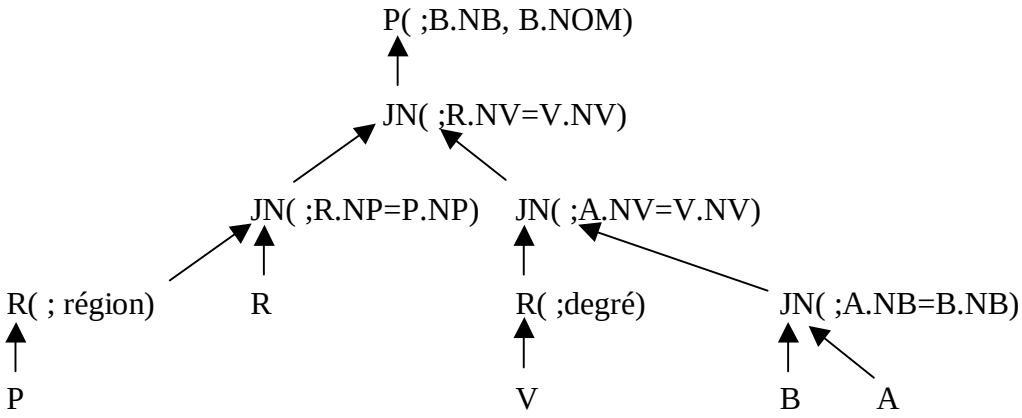
Pour une même requête SQL, on peut construire plusieurs arbres algébriques équivalents.

On distingue entre **arbre linéaire** et **arbre ramifié**.

On distingue aussi entre **arbre optimisé** et **arbre non optimisé**.

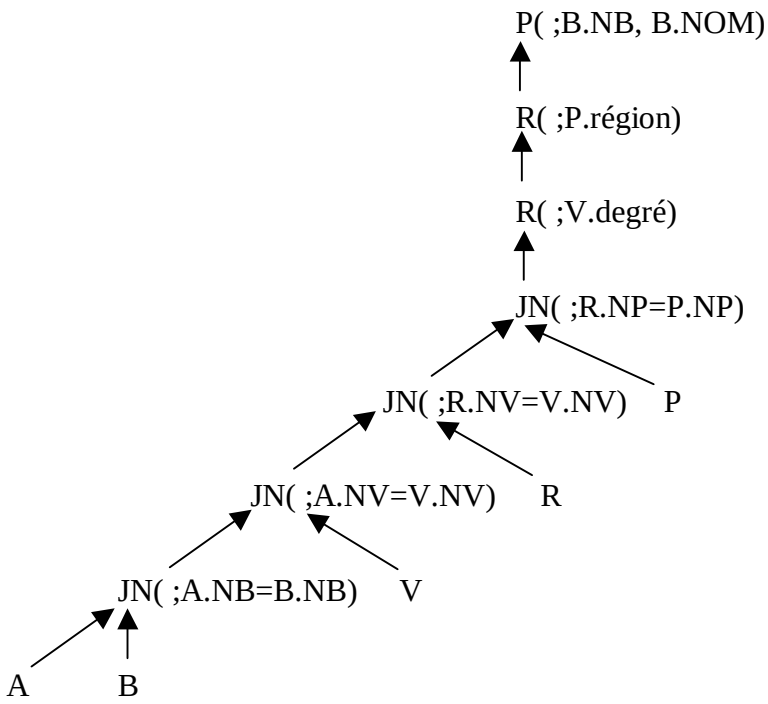
## Arbre ramifié

C'est un arbre dont au moins une jointure a deux jointures en entrée :




## Arbre linéaire

C'est un arbre dont chaque jointure a au moins une entrée sans jointure :




## Nombre théorique d'arbres équivalents pour une requête

En ne s'intéressant qu'aux jointures, on peut calculer le nombre théorique d'arbres équivalents pour une requête en fonction du nombre de tables en jeu.

Avec 2 tables, on peut fabriquer 1 arbre.

Avec 3 tables, on peut fabriquer 3 arbres.

Etc.

La formule générale est donnée par la définition d'une suite :

| |
|---|
| **NA(2) =1** |
| **NA(T) = NA(T-1) * (2T-3)** |

Ce qui nous donne : (2 ;1), (3 ;3), (4 ;15), (5 ; 105), (6 ; 945), (7 ; 10 395), etc.

On arrive donc très vite à un nombre d'arbres équivalents très important. Les algorithmes d'optimisation devront donc aussi être capables de choisir le meilleur.

## 2. Théorie sur les règles de transformation des arbres algébriques

Il existe plusieurs règles transformation des arbres qui vont permettent de théoriser l'optimisation de l'arbre algébrique d'une requête.

Ces règles proposent des équivalences entre les formulations.

### Règles de base sur les produits cartésiens et les jointures

**Règle 1 : Commutativité des produits cartésiens et des jointures**

JN(A,B) ⇔ JN(B,A)

PC(A,B) ⇔ PC(B,A)

**Règle 2 : Associativité des produits cartésiens et des jointures**

JN( JN(A,B), C) ⇔ JN (A, JN(B,C) )

PC( PC(A,B), C) ⇔ PC (A, PC(B,C) )

### Règles de base sur les restrictions

**Formalisme**

→ signifie : suivi de

A → B est équivalent à :

B
↑
A

**Règle 3 : Fusion et dissociation des restrictions**

R(table ; restiction1, restriction2) ⇔ R(table ; restiction1) → R(table ; restriction2)

**Règle 4 : Commutativité des restrictions**

R(table ; restiction1, restriction2) ⇔ R(table ; restiction2, restriction1)

**Règle 5 : Commutativité sous condition des restrictions et des projections**

On peut toujours faire la restriction avant la projection :

R(tab, Ai) ⇒ P(tab ; A1..An)

↑ ↑

P(tab ; A1.An) R(tab, Ai)

On peut faire la restriction après la projection sous certaines conditions :

⇐

Si Ai ∈ A1..An

**Règle 6 : Commutativité sous condition des restrictions et des produits cartésiens (et aussi des jointures)**

On peut toujours faire la restriction après le produit cartésien ou la jointure :

PC(table1(A1…An), table2) ⇒ R(tab, Ai)

R(tab, Ai) PC(table1(A1…An), table2)

On peut faire la restriction avant le produit cartésien ou la jointure sous certaines conditions :

⇐

Si Ai ∈ A1..An

**Règle 7 : Commutativité des restrictions avec les opérations ensemblistes**

R(tab1;Ai..Ak) et R(tab2;Ai..Ak) → Union (tab1, tab2) ⇔ Union (tab1, tab2) → R(;Ai..Ak)

R(tab1;Ai..Ak) et R(tab2;Ai..Ak) → Inter (tab1, tab2) ⇔ Inter (tab1, tab2) → R(;Ai..Ak)

R(tab1;Ai..Ak) et R(tab2;Ai..Ak) → Diff (tab1, tab2) ⇔ Diff (tab1, tab2) → R(;Ai..Ak)

**Règles de base sur les projections**

**Règle 8 : Fusion et dissociation des projections**

P(table ; Att1, Att2) ⇔ P(table ; Att1) → P(table ; Att2)

**Règle 9 : Commutativité des projections**

P(table ; Att1, Att2) ⇔ P(table ; Att2, Att1)

**Règle 10 : Commutativité sous condition des projections et des produits cartésiens**

On peut toujours faire la projection après le produit cartésien

P(tab, Ai) → PC(table1(A1…An), table2) ⇒ PC(table1(A1…An), table2) → P(tab, Ai)

On peut faire la projection avant le produit cartésien sous certaines conditions:

P(tab, Ai) → PC(table1(A1…An), table2) ⇐ PC(table1(A1…An), table2) → P(tab, Ai)

Si Ai ∈ A1..An

**Règle 11 : Commutativité sous condition des projections et des jointures**

On peut toujours faire la projection après la jointure

P(tab1, Ai..Ak) → PC(tab1(A1..An), tab2) ⇒ PC(tab1(A1..An), tab2) → P(;Ai..Ak)

On peut faire la projection avant la jointure sous certaines conditions:

P(tab1, Ai..Ak) → PC(tab1(A1..An), tab2) ⇐ PC(tab1(A1..An), table2) → P(; Ai..Ak)

Si Ai..Ak ∈ A1..An et si l'attribut de jointure ∈ Ai..Ak

**Règle 12 : Commutativité des projections avec les opérations ensemblistes**

P(tab1;Ai..Ak) et P(tab2;Ai..Ak) → Union (tab1, tab2) ⇔ Union (tab1, tab2) → P(;Ai..Ak)

P(tab1;Ai..Ak) et P(tab2;Ai..Ak) → Inter (tab1, tab2) ⇔ Inter (tab1, tab2) → P(;Ai..Ak)

P(tab1;Ai..Ak) et P(tab2;Ai..Ak) → Diff (tab1, tab2) ⇔ Diff (tab1, tab2) → P(;Ai..Ak)

## 3. Algorithme d'optimisation syntaxique et arbres algébriques optimisés

### Algorithme d'optimisation syntaxiquue

#### Principe intuitif de l'optimisation intuitive

Le principe général de l'optimisation repose sur le constat suivant :

- Les opérations unaires produisent des tables plus petites que la table d'origine.
- Les opérations binaires produisent des tables plus grandes que la table d'origine. Autrement dit, ce sont les produits cartésiens des jointures qui accroissent la taille des tables intermédiaires.

On va donc :

**Supprimer un maximum de lignes et de colonnes avant de faire les jointures.**

#### Etapes de l'optimisation intuitive

L'optimisation intuitive se résume à :

1. Faire toutes les restrictions spécifiques pour limiter le nombre de tuples.
2. Faire toutes les projections mono-tables possibles pour limiter le nombres d'attributs.
3. Faire les jointures et après chaque jointure, les projections possibles.
4. Finir la projection
5. Faire les distinct, les tris, les group by, les fonctions de groupe.

#### L'optimisation d'après les règles de transformation

En utilisant les règles de transformations précédentes, on arrive à l'algorithme suivant :

1. Règle 3 : on sépare les restrictions comportant plusieurs prédicats.
2. Règles 4, 5, 6 et 7 : on descend les restrictions le plus bas possible.
3. Règles 3 : on regroupe les restrictions successives
4. Règle 8 : on sépare les projections comportant plusieurs prédicats.
5. Règle 9, 10, 11 et 12 : on descend les projections le plus bas possibles
6. Règle 8 : on regroupe les projections successives

### Arbres algébriques optimisés

On peut maintenant optimiser les deux arbres algébriques précédemment présentés.

#### Rappel du modèle :

Buveurs (**NB**, Nom, Prénom, Type)

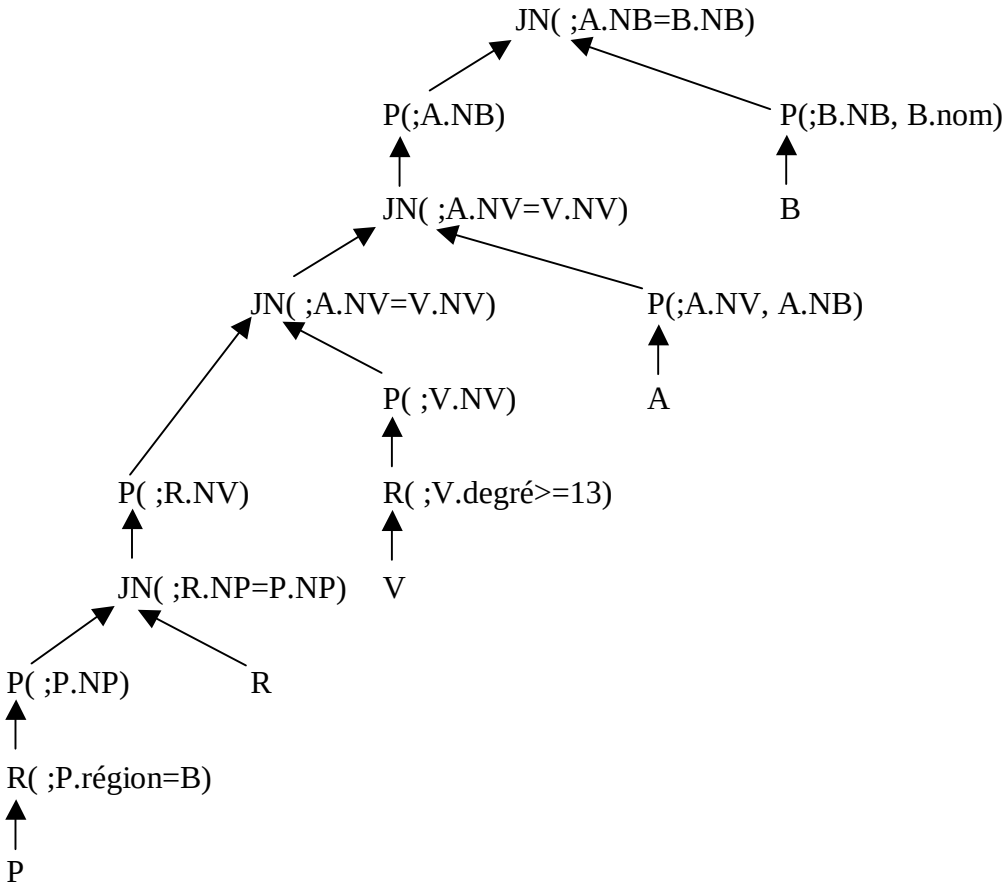
Vins (**NV**, Cru, Millésime, Degré)

Producteurs (**NP**, Nom, Région)

Abuser (**#NB, Date**, Quantité, #NV)

Produire (**#NP, #NV**)

**Arbre linéaire optimisé**

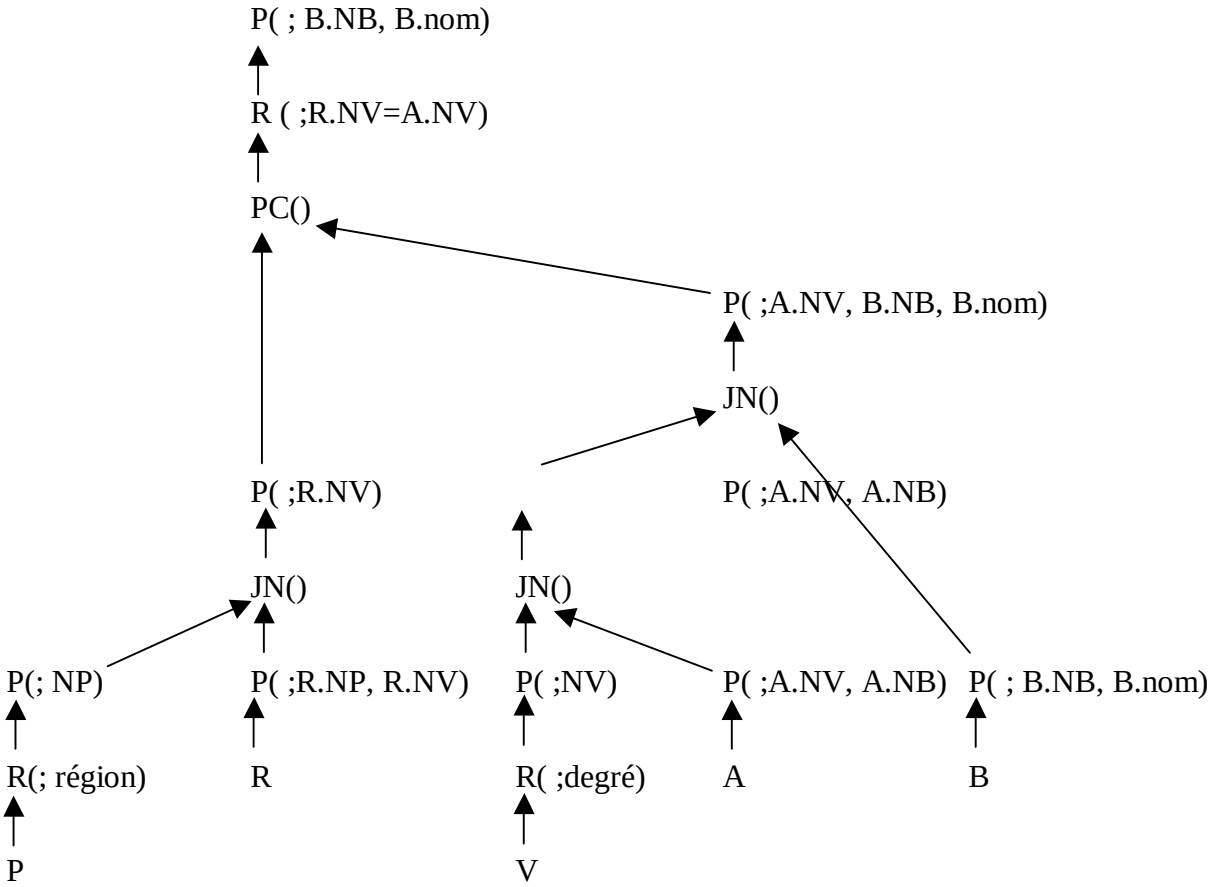



**Traduction SQL**

```
Create or replace view v1 as Select * from producteurs where region = “Bordelais”;
Create or replace view v2 as Select np from v1;
Create or replace view v3 as Select * from produire join v2 using (np);
Create or replace view v4 as Select nv from v3;
Create or replace view v5 as Select * from vins where degre >=13;
Create or replace view v6 as Select nv from v5;
Create or replace view v7 as Select * from v4 join v6 using (nv);
Create or replace view v8 as Select nv, nb from abuser;
Create or replace view v9 as Select * from v7 join v8 using (nv);
Create or replace view v10 as Select nb from v9;
Create or replace view v11 as Select nb, nom from buveurs;
Create or replace view v12 as Select * from v10 join v11 using (nb);
Select * from v12;
```

Ce dernier select produit le même résultat que le select de base.

**Arbre ramifié optimisé**



## 4. Première approche du calcul du coût

### Quantité de données

Rappel du modèle :

Buveurs (**NB**, Nom, Prénom, Type)
Vins (**NV**, Cru, Millésime, Degré)
Producteurs (**NP**, Nom, Région)
Abuser (**#NB, Date**, Quantité, #NV)
Produire (**#NP, #NV**)

Avec

- 100 buveurs : **B = (100 ; 4)**
- 50 vins : **V = (50 ; 4)**
- 20 producteurs : **P = (20 ; 3)**
- 250 abuser : **A = (250 ; 4)**
- 75 produire : **R = (75 ; 2)**

On ajoute les sélectivités suivantes :

- Sélectivité(Producteurs ; Région = “Bordeaux”) = 25%
- Sélectivité(Vins ; Degré >13) =20%
-

La sélectivité, c’est le pourcentage de tuples qui vérifient une condition.
La sélectivité est un élément de la métabase.

### Principe du calcul

On note le nombre de tuples et d’attributs dans chaques tables après chaque opération.

En cas de produit cartésien, le nombre de tuples de la table résultat est égal à la multiplication des nombres de tuples des 2 tables du produit cartésien

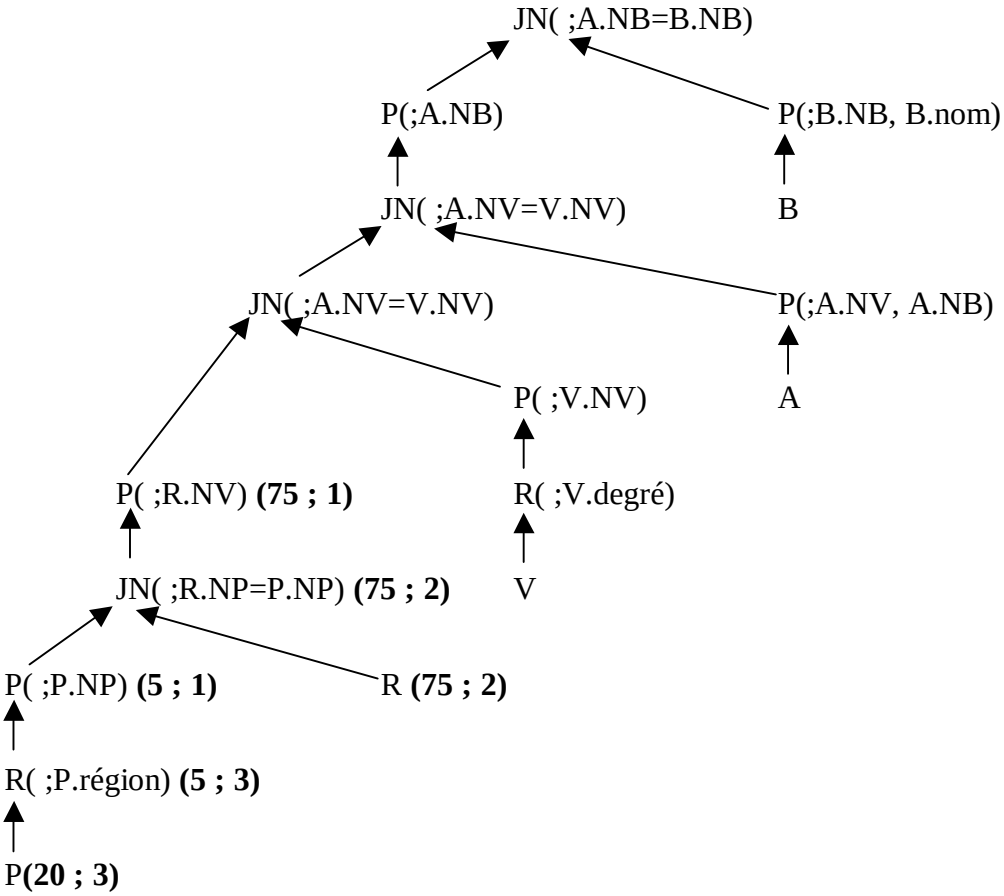
En cas de jointure artificielle, on ne peut rien dire et on en revient à un produit cartésien.

En cas de jointure naturelle, on retrouve exactement le nombre de tuples de la table maitre si la table jointe n’a pas subi de restriction, sinon on retrouve au maximum le nombre de tuples de la tables maître.

En cas de restriction, le coefficient de sélectivité d’applique directement si la restriction s’applique immédiatement à la table, sinon il est inutilisable.

**Exemple**

**Arbre linéaire optimisé avec calcul de coût**




- 100 buveurs : **B = (100 ; 4)**
- 50 vins : **V = (50 ; 4)**
- 20 producteurs : **P = (20 ; 3)**
- 250 abuser : **A = (250 ; 4)**
- 75 produire : **R = (75 ; 2)**

- Sélectivité(Producteurs ; Région = “Bordeaux”) = 25%
- Sélectivité(Vins ; Degré >13) =20%

# EXERCICES

## 0. Présentation générale des exercices

**Travail à effectuer pour chaque exercice**

1. Faire le graphe des tables
2. Faire le graphe de la question
3. Répondre à la question en SQL
4. Faire un arbre algébrique ramifié.
5. Faire un arbre linéaire et optimisé avec calcul de coût
6. Traduire cet arbre en SQL avec des vues.

**Rappel sur la création des vues**

CREATE OR REPLACE VIEW nom_vue AS SELECT …

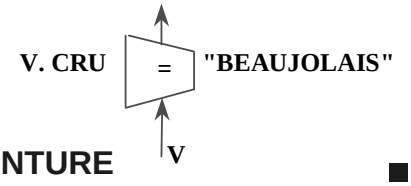
**Rappels sur l'écriture des opérateurs relationnels**

**Représentation textuelle des opérateurs de l'algèbre relationnelle**

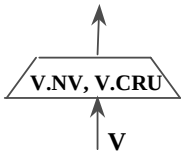
| **Opération** | **Formalisme** |
|---|---|
| **Projection** | P (table ; liste d'attributs) |
| **Restriction** | R (table ; liste de restrictions) |
| **Produit cartésien** | PC (table1, table 2) |
| **Jointure naturelle** | JN (table maître, table jointe)<br>Dans ce cas, par défaut, on a la restriction : T1.NT2=T2.NT2 |
| **Jointure artificielle** | = PC suivi de Restriction de jointure artificielle |
| **Union** | Union (table1, table2) |
| **Différence** | Diff (table1, table2) |
| **Intersection** | Inter (table1, table2) |
| **Tri** | Tri (table; liste d'attributs) |
| **Agrégat** | Agreg (table ; attributs de regroupement ; attributs de statistique) |
| **Fonction de groupe** | FG (table ; attributs de statistique) |

**Représentation graphique des opérateurs de l'algèbre relationnelle dans les arbres algébriques**

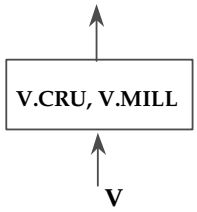
■ **RESTRICTION**



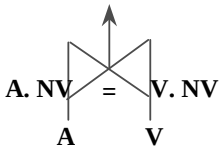

■ **PROJECTION**



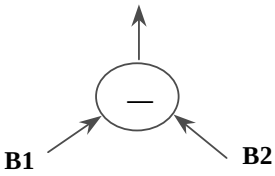

■ **TRI**



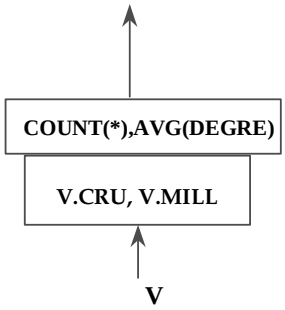

■ **JOINTURE**



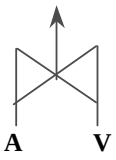

■ **DIFFERENCE**



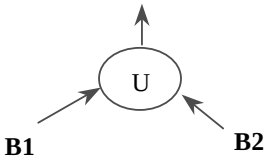

■ **AGREGAT**




■ **PRODUIT CARTESIEN**




■ **UNION**




*D'après Georges Gardarin*

## 1. Les employés

**Schéma de la BD Entreprise**

**EMPLOYES(<u>NE</u>**, nom, job, datemb, sal, comm., #ND, *NEchef)

**DEPARTEMENTS (<u>ND</u>,** nom, ville)

Employés(500 ; 8)

Départements (5 ; 3)

S (départements ; nom= « Comptabilité ») = 10%

**Ex. 1**

Liste des employés par ordre alphabétique du département comptabilité avec leurs jobs et le nom de leur supérieur hiérarchique

## 2. La bibliothèque

**Schéma de la BD Bibliothèque**

**ADHERENTS** (NA, nom, prenom, adr, CP, ville, tel)

**OEUVRES** (NO, titre, auteur)

**LIVRES** (NL, editeur, #NO)

**EMPRUNTER(**#NL, datemp, datretmax, datret, #NA)

Les livres sont les exemplaires physiques des livres.

Adhérents(10 000 ; 7)

Oeuvres (1 100 000 ; 3)

Livres (1 200 000 ; 3)

Emprunter (500 000 ; 5)

S (Adhérents ; CP = 75 019 ) = 10%

S (Oeuvres ; Auteur) = 1 / 300 000

**Ex. 2**

Tous les auteurs actuellement empruntés par des adhérents du 19ème arrondissement, trié par ordre alphabétique.

## 3. Les projets

**Schéma de la BD projets simplifié**

**EMPLOYES** (**NE,** nom)

**SERVICES** (**NS,** Sx, #NE) Un service a un chef de service.

**PROJETS** (**NP**, Px, #NS) Un projet est affecté à un service et un seul.

**PARTICIPER** (**#NE, #NP**) Les employés participent à des projets.

Les chefs de services ne participent pas forcément aux projets affectés à leur service.

Employes (1 000 ; 2)

Services (10 ; 3)

Projets (500 ; 3)

Participer (10 000 ; 2)

**Ex. 3**

Tous les projets sur lesquels Jean Dupont a travaillé et dont il a été le chef de service du service responsable du projet

## 4. La maison de disques

**Schéma de la BD Maison de disques**

**DISQUES** (**ND**, titre, année)

**CHANSONS** (**NC**, titre, durée, année)

**MUSICIENS** (**NM**, nom, prénom, nationalité)

**JOUER** ( **#NM, #NC, instrument** )

**REGROUPER ((#ND, #NC,** piste**)**

Disques (100 000 ; 3)

Chansons (1 000 000 ; 3)

Musiciens (10 000 ; 4)

Jouer (4 000 000 ; 3)

Regrouper (700 000 ; 3)

S (Disques ; année >= 1960) = 80%

**Ex. 4**

Quels sont les titre des disques auxquels a participé KUTI en tant que saxophoniste depuis 1960.

## 5. L'école

**Schéma de la BD Ecole**

**ETUDIANTS** (**NET**, nom, prenom, email)

**GROUPES** (**NGR**, nom, niveau, num, année ) : exemple : 37, SIGL, 3, 1, 2008 : SIGL3 groupe 1 de 2008-2009.

**EXAMENS** (**NEX**, matiere, prof, sujet, dateCréation, niveauPrévu, typePrévu, duréePrévue) : exemple de niveau préu : 3 pour troisième année.

**EPREUVES** (**NEP**, #NGR, dateHeureDébut, durée, type, salle, #NEX) : #NGR et dateheure forment une clé secondaire. A noter que #NGR et salle aussi.

**EVALUER** (**#NEP, #NET**, note) Les étudiants sont évalués dans une épreuve: il y a une note. En cas d'absence, ils ne sont pas dans la table.

**PARTICIPER (#NET, #NGR**) : les étudiants participent à un groupe. Ils peuvent participer à plusieurs groupes.

Fonctionnement du système : on crée un groupe, on enregistre les étudiants et on les affecte dans un groupe (table participer). On crée des examens, puis des épreuves qui font référence à

ces examens. L'épreuve est pour un groupe donné. Quand on récupère les copies, on enregistre des évaluations avec des notes.

Etudiants (1 000 ; 4)

Groupe (115 ; 5)

Examens (2 800 ; 8)

Epreuves (4 000 ; 7)

Evaluer (40 000 ; 3)

Participer (39 000 ; 2)

S (Groupe ; Niveau = 3) = 38 %

S (Groupe ; Année = 2007) = 10%

S (Examens ; Matière = BD) = 10%

S (Groupe ; Niveau = 2 et nom = SIGL) = 15%

S (Examens ; Matière = Math) = 10%

(Remarque : on part sur un modèle de 10 ans, 100 élèves par promo, 3 années, 3 groupes (1 classe) en 1ère année, 5 groupes (3 classes) en 2ème et 3ème année, 20 matières par an pour une classe, 2 examens par matières.

**Ex. 5**

Les notes des élèves de 3ème année pour les épreuves de BD pour l'année scolaire 2007-2008 ?

Le résultat devra être exploitable. Dans les résultats, on précisera la matière, son type et la date de l'examen.

**Ex. 6**

Quels sont les étudiants absents aux épreuves de math de l'année 2007 en SIGL 2.

Pour répondre à cette question, on a intérêt à se demander qui étaient les étudiants prévus et qui étaient les étudiants présents.