

## Chapitre III : Optimisation de requêtes en base de données relationnelle

### Introduction

La plupart des SGBD relationnels modernes offrent des langages de manipulation basés sur SQL, non procéduraux et utilisant des opérateurs ensemblistes. Avec de tels langages l'utilisateur définit les données qu'il veut visualiser sans fournir des algorithmes d'accès à ces données. L'objectif de l'optimisation est de déterminer ces algorithmes d'accès aussi appelés **plan d'exécution**. Il est essentiel pour un système d'utiliser des plan d'exécution optimisés pour les requêtes les plus fréquentes. Un plan d'exécution dépend du schéma interne de la base de données (particulièrement de l'existence d'index) et de la taille des tables.

Un optimiseur de requêtes transforme une requête exprimée dans un langage source (SQL par exemple) en un plan d'exécution composé d'une séquence d'opérations de bas niveau réalisant efficacement l'accès aux données.

L'exécution d'une requête SQL par un SGBD suit les étapes suivantes :

1. **Analyse syntaxique** : Vérification de la syntaxique et traduction en opérations algébriques ;
2. **Contrôle** : Vérification des droits d'accès ;
3. **Optimisation** : Génération des plans d'exécution et choix du meilleur ;
4. **Exécution** : Compilation et exécution du plan choisi.

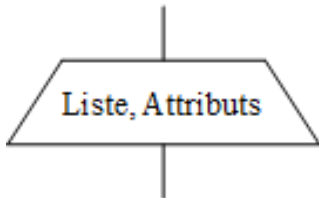
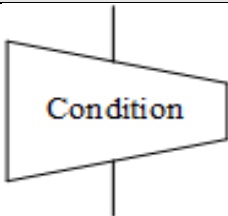
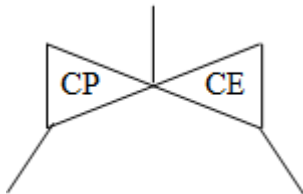
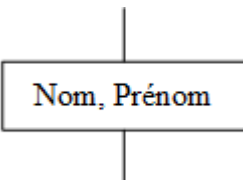
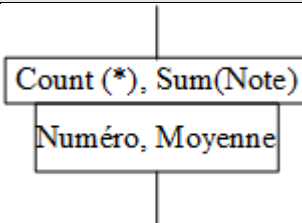
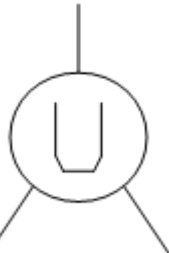
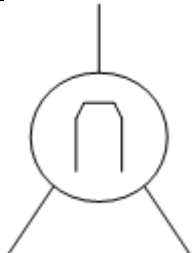
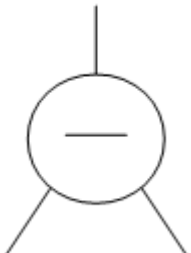
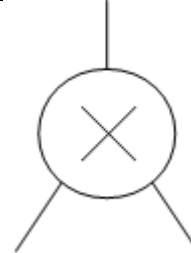
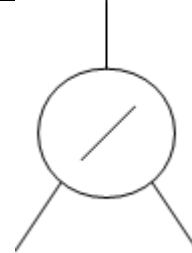
Optimiser une requête c'est chercher le plan d'exécution optimal pour minimiser le coût de son exécution. Pour ce faire, on construit tous les arbres algébriques possibles pour la requête en question, puis on évalue leur coût et enfin on choisit celui qui a le plus petit coût d'exécution.

### I. Arbre algébrique

Un arbre algébrique est la représentation graphique sous forme d'arbre d'une requête dans laquelle : les feuilles représentent les relations, les nœuds intermédiaires représentent les opérateurs algébriques, le nœud racine représente la relation résultat de la requête et les arcs représentent les flux de données entre les opérations. Dans l'arbre, les tris sont représentés par des rectangles contenant les attributs sur lesquels portent les tris, les agrégats sont représentés par un rectangle contenant les attributs de la clause Group By et d'un autre rectangle contenant les attributs résultats calculés.

## I. 1. Représentation des opérations de l'algèbre relationnelle

**Tableau 1 :** Représentation graphique des opérations de l'algèbre relationnelle

Projection		Sélection		Jointure naturelle	
					
Tri			Agrégat		
					
Union	Intersection	Différence	Produit	Division	
					

## I. 2. Restructuration algébrique

Pour optimiser une requête on la réécrit pour obtenir une requête équivalente en utilisant :

- la commutativité et l'associativité de la jointure naturelle : elles permettent de changer l'ordre des jointures pour minimiser la taille des données à parcourir et le nombre de comparaisons de valeurs à faire ;
- le groupage des sélections : il permet d'effectuer plusieurs sélections en un seul parcours de la table au lieu d'une sélection par parcours ;
- la commutativité des sélections et jointures : elle permet d'effectuer les sélections avant les jointures ;
- la descente des projections en veillant à conserver les attributs qui seront utilisés par la suite.

### I. 3. Heuristiques d'optimisation

L'ordonnancement des opérations se fait ensuite comme suit :

- on applique les opérations réductrices (projection, sélection) en les faisant descendre le maximum possible vers les feuilles ;
- on retarde le plus possible les jointures et les produits cartésien ;
- on exécute les jointures qui font appel aux relations les moins volumineuses.

#### Exemple

Soit le schéma de la base de données **Agence** suivante :

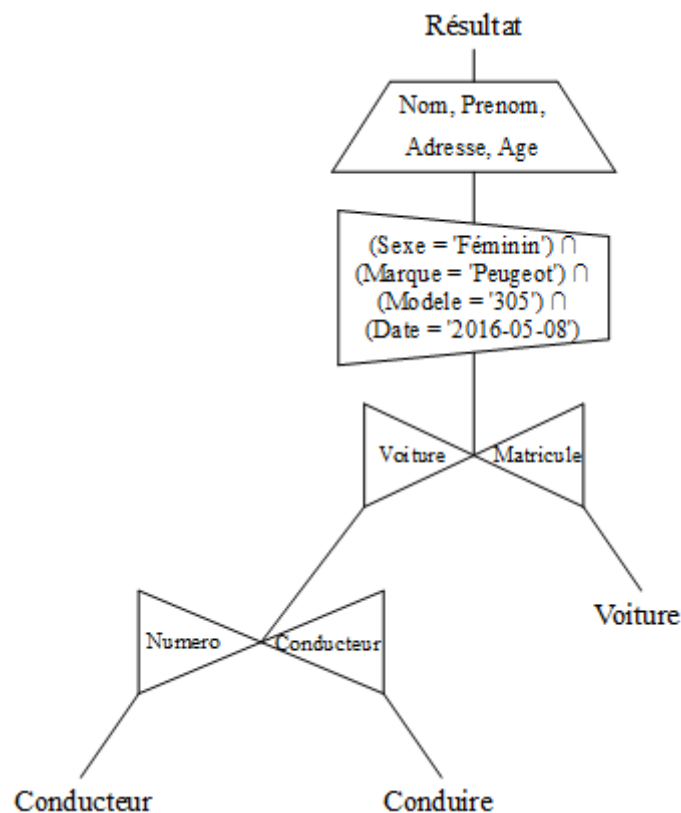
**Conducteur** (Numero, Nom, Prenom, Adresse, Age, Telephone, Sexe)

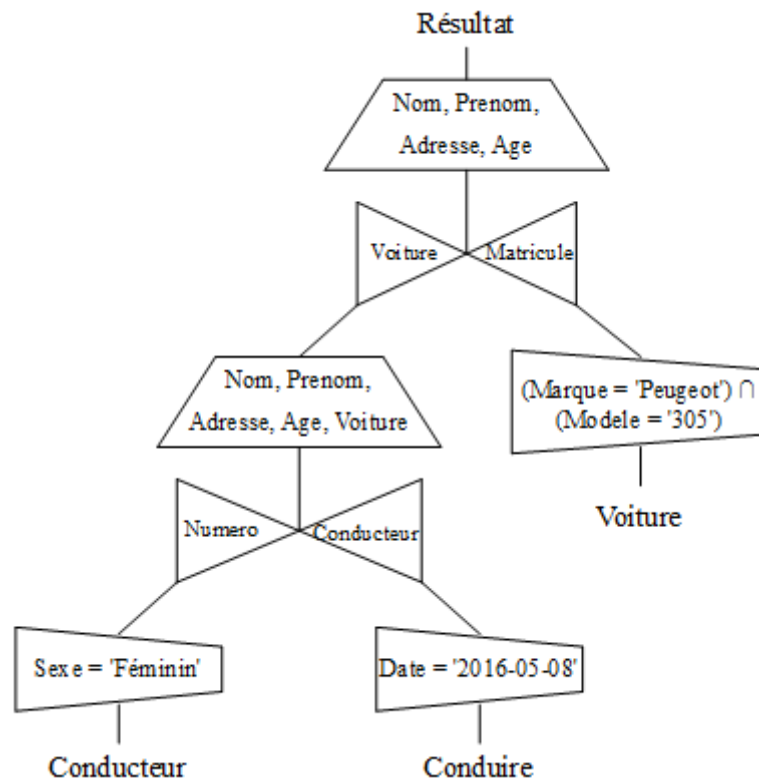
**Voiture** (Matricule, Marque, Modele, Type, Couleur, Annee)

**Conduire** (#Conducteur, #Voiture, Date, Nb\_Fois)

Soit la requête : Quelles sont les femmes (Nom, Prénom, Adresse, Age) qui ont conduit une Peugeot 305 le 08/05/2016 ?

Nous donnons ci-dessous deux arbres algébriques pour cette requête.





## II. L'indexation d'attributs

La taille des bases de données sont de plus en plus importantes. Si la recherche d'une information dans une table se fait de manière séquentielle, c'est-à-dire en parcourant toute la table ou tous les attributs concernés du premier au dernier, les temps de réponse peuvent être très élevés et les performances sont détériorées. L'indexation en base de données est un mécanisme permettant de résoudre ce problème.

### II. 1. Qu'est-ce qu'un index ?

Un index est une structure de données utilisée en base de données pour minimiser les temps de réponse pendant les opérations de **recherche**, de **tri**, de **jointure** et de **regroupement**. C'est une table à une seule colonne associant à une clé l'adresse de l'enregistrement considéré. Chaque index est placé sur une table et contient des entrées comportant chacune une valeur extraite des données et un pointeur sur son emplacement exacte. Pour retrouver un enregistrement, le système utilise l'index pour trouver sa localisation. Une table peut comporter plusieurs index : un index primaire et un/des index secondaire(s). Chaque index est créé sur un attribut ou une combinaison de plusieurs attributs. Pour chaque table, le système utilise d'abord l'index primaire lors de la recherche d'informations.

## II. 2. Index B-arbre

Il existe plusieurs types d'index dont le plus utilisé en base de données est le B-Tree (B-arbre). Cependant, l'index de hachage est aussi utilisé en base de données. En entrepôt de données c'est l'index Bitmap et l'index de jointure qui sont les plus utilisés.

Un B-arbre d'ordre  $k \geq 2$  est un arbre tel que :

- chaque sommet contient entre  $k - 1$  et  $2k - 1$  valeurs, sauf la racine qui peut contenir de 1 à  $2k - 1$  valeurs ;
- si un sommet contient  $m$  valeurs  $x_1, x_2, \dots, x_m$  alors  $i < j \Rightarrow x_i \leq x_j$  ;
- chaque sommet possède entre  $k$  et  $2k$  sous-arbres  $A_0, A_1, A_2, \dots, A_m$  ;
- toutes les feuilles sont à la même profondeur.

Un B-arbre est un arbre équilibré qui stocke les données sous une forme triée. Contrairement aux arbres binaires de recherche, les nœuds parents peuvent posséder plus de deux nœuds enfants. De plus un B-arbre grandit à partir de la racine alors que les arbres binaires de recherche grandissent à partir des feuilles.

Il est plus efficace si le nombre de valeurs est important avec plusieurs valeurs distinctes.

### Remarque

- La création d'index ralentit les opérations de mise à jour : insertion, modification ;
- L'index occupe de l'espace sur le disque dur et en mémoire ;
- La clé primaire est toujours indexée sans doublons ;
- Un index sur plusieurs attributs correspond à un index créé sur un seul attribut dans lequel on a concaténé tous les attributs.

### Conseils :

Il est conseillé de ne pas indexer un attribut :

- Si la table contient peu de données ;
- S'il y a très peu de recherche portant sur cet attribut ;
- S'il n'y a jamais de tri de la table sur cet attribut ;
- S'il est normal que l'attribut contient des doublons (champ de type booléen, etc.).

## II. 3. Création et suppression d'index

### II. 3. 1. Ajout d'index lors de la création de la table

Deux méthodes peuvent être utilisées pour ajouter un index pendant la création de la table :  
On peut créer l'Index lors de la description de l'attribut ou la mettre après celle-ci.

#### a. Lors de la description de l'attribut

**Create** TABLE Nom\_Table

```
(  
    Colonne1    Type        Index1,  
    Colonne2    Type        Index2  
);
```

#### Exemple

**Create** TABLE Bureau

```
(  
    Numero      Char(2)      Primary Key,  
    Telephone    Varchar(15) Unique  
);
```

**Remarque** : Avec cette méthode :

- Les index **Primary Key** et **Unique** ne sont pas précédés du mot Clé INDEX ;
- Il n'est pas possible de définir des index composites (composés de plusieurs attributs) ;
- Il n'est pas possible également de définir un index sur une partie d'un attribut.

#### b. Après la description de l'attribut

**Create** TABLE Nom\_Table

```
(  
    Colonne1    Type,  
    Colonne2    Type,  
    Colonne3    Type,  
    Colonne4    Type,  
    Constraint PK_Nom_Table    Primary Key (Colonne1),  
    INDEX      Nom_Index    (Colonne2 [, Colonne3]),  
    UNIQUE     [INDEX]      Nom_Index    (Colonne4 [, Colonne3])  
);
```

**Exemple****Create Table** Personne

```
(
    CNI          char(12),
    Nom          varchar(15),
    Prenom       varchar(25),
    DateNaiss    Date,
    Adresse      varchar(30),
    Sexe         varchar(8),
    Telephone    char(12),
    Constraint  PK_Personne Primary Key (CNI),
    Index       Ind_Pers_1  (Nom, Prenom, DateNaiss),
    Unique Index  UQ_Tel    (Telephone)
);
```

**II. 3. 2. Ajout d'index à une table déjà créée**

Il est possible d'ajouter un index à une table déjà créée. Deux syntaxes différentes peuvent être utilisées :

**a. Avec la syntaxe Alter Table**

```
Alter TABLE Nom_Table Add INDEX [Nom_Index] (Colonne1 [, Colonne2, ...]) ;
Alter TABLE Nom_Table Add UNIQUE [Nom_Index] (Colonne1 [, Colonne2, ...]) ;
Alter TABLE Nom_Table Add FULLTEXT [Nom_Index] (Colonne1 [, Colonne2, ...]) ;
```

**b. Avec la syntaxe Create Index**

```
Create INDEX Nom_Index On Nom_Table (Colonne1 [, Colonne2, ...]) ;
Create UNIQUE INDEX Nom_Index On Nom_Table (Colonne1 [, Colonne2, ...]) ;
Create FULLTEXT INDEX Nom_Index On Nom_Table (Colonne1 [, Colonne2, ...]) ;
```

**Exemple**

```
Alter Table Personne Add Index Prenom_DateNaiss (Prenom, DateNaiss) ;
Create Index PN_Pers On Personne (Prenom, Nom) ;
```

### II. 3. 3. Suppression d'un index

Un index créé ne peut pas être modifié. On peut par contre le supprimer et en créer un nouveau. La syntaxe de la suppression d'un index est :

**Alter TABLE** Nom\_Table **Drop INDEX** Nom\_Index ;

**Drop Index** Nom\_Index ;

#### Exemple

**Alter Table** Personne **Drop Index** Prenom\_DateNaiss ;

**Drop Index** Prenom\_Pers ;

### II. 4. Utilisation des Index par la gauche

Si nous avons un index créé sur plusieurs attributs (par exemple l'index Ind\_Pers sur le nom, le prénom et la date de naissance d'une personne), le système tri les enregistrements suivant les noms, puis les prénoms et enfin les dates de naissance.

Ainsi, si nous avons dans la table personne l'instance suivante :

CNI	Nom	Prenom	DateNaiss	Adresse	Sexe	Telephone
1 000 00 125	Dieme	Cheikh	28/05/2000	N° 014 Tilène	H	77 000 14 12
1 241 01 541	Ndoye	Cheikh	14/10/2001	N° 002 Santhiaba	H	77 000 14 00
2 010 89 014	Fall	Abibatou	30/02/1989	N° 142 Boucotte	F	76 000 84 14
2 200 05 124	Dieme	Aicha	18/02/2005	N° 014 Tilène	F	76 000 05 74
1 014 08 214	Ndoye	Cheikh	30/02/2008	N° 029 Boucotte	H	70 000 14 21
2 010 98 014	Seck	Aicha	28/05/1998	N° 102 Belfort	F	76 000 00 19

Avec cet index, les enregistrements sont triés comme suit :

CNI	Nom	Prenom	DateNaiss	Adresse	Sexe	Telephone
2 200 05 124	Dieme	Aicha	18/02/2005	N° 014 Tilène	F	76 000 05 74
1 000 00 125	Dieme	Cheikh	28/05/2000	N° 014 Tilène	H	77 000 14 12
2 010 89 014	Fall	Abibatou	30/02/1989	N° 142 Boucotte	F	76 000 84 14
1 241 01 541	Ndoye	Cheikh	14/10/2001	N° 002 Santhiaba	H	77 000 14 00
1 014 08 214	Ndoye	Cheikh	30/02/2008	N° 029 Boucotte	H	70 000 14 21
2 010 98 014	Seck	Aicha	28/05/1998	N° 102 Belfort	F	76 000 00 19

S'il y a fréquemment de recherches dont les conditions portent sur le nom uniquement ou sur le nom et le prénom, on peut être tenté de créer un index sur le nom ou un index sur le nom et le prénom. Par contre, ces index sont inutiles puisque le système est capable d'utiliser l'index



**Ind\_Pers** si la condition porte sur le nom ou sur le nom et le prénom. Cependant, s'il est fréquent que des conditions de recherche portent sur le prénom ou le prénom et la date de naissance, il faudra créer d'autres index.

## II. 5. Exercice d'application

Soit la relation **Equipe** (Nom, Categorie, Anciennete, AdresseSiege)

Les recherches qui sont le plus souvent faites sur la table implémentant cette relation portent sur les attributs :

- Nom ;
  - Nom et Categorie ;
  - Nom, Categorie et AdresseSiege ;
  - AdresseSiege.
1. Quels sont les index qu'il faut créer pour minimiser les temps de réponse ?
  2. Donnez les syntaxes de création de ces index sous oracle.