Chapitre I
ALGEBRE RELATIONNELLE

1 – HISTORIQUE

Dans les années 1970, les systèmes d'informations s'informatisent. Afin de les exploiter de manière optimum, il devient nécessaire de mettre en place un système capable d'organiser et de stocker un nombre conséquent d'informations. A l'époque, les systèmes d'informations à mettre en place dépassent déjà le Giga Octet. Les laboratoires d'IBM donnent alors naissance à un système de gestion de fichiers qui permet à un utilisateur d'accéder aisément aux informations via un langage d'interrogation simple, tout en occultant la complexité de l'architecture mise en œuvre. Les bases de données viennent de naître.

Une base de données est un ensemble d'informations structurées et stockées sur un support permettant une utilisation régulière. Les bandes des années 70 sont progressivement remplacées par les disques durs actuels. Les outils actuels permettent de gérer une ou plusieurs bases de données. Ces outils se nomment des SGBD : Système de Gestion de Bases de Données (DataBase Management System ou DMS en anglais).

Par la suite, diverses sociétés ont proposé leur solution de gestion des données et leur langage pour l'extraction des données. Actuellement, les principaux outils du marché sont les suivants (la liste n'est pas exhaustive) : MS Access et MS SqlServer de la société Microsoft, Oracle et MySQL (gratuit).

Tous ces outils intègrent le langage SQL (Structured Query Language, ou Langage de requêtes structuré). Ce langage permet :

- de définir les données, c'est un LDD ou langage de définition de données (DDL ou Data Definition Language en anglais). Il permet de créer, modifier et supprimer des tables au sein d'une base relationnelle.
- de manipuler les informations, c'est un LMD ou langage de manipulation de données (DML ou Data Manipulation Language en anglais). Il permet d'insérer, modifier et supprimer des informations dans les tables d'une base de données. Il permet également d'interroger le système afin d'obtenir des informations précises.
- de contrôler les données, c'est un LCD ou langage de contrôle de données (DCL ou Data Control Language en anglais). Il permet de définir des niveaux d'autorisation en fonction des différents types d'utilisateurs.

L'ancêtre du langage SQL date de 1979. Il sera normalisé en 1986 avec le SQL-ANSI (American National Standard Institute), en 1989 avec SQL-ISO (International Standard Application) et ANSI, en 1992 avec la seconde version du SQL-ISO et ANSI nommé SQL-2 et enfin en 1999, pour établir la troisième version, SQL 3. Toutefois, tous les outils n'intègrent pas encore cette dernière version.

2 – SYNTAXE GENERALE

L'algèbre relationnelle a été inventée en 1970 par Edgar Frank Codd, directeur de recherche du centre IBM de San José. Les opérations relationnelles permettent de créer une nouvelle relation (table) à partir d'opérations élémentaires sur d'autres tables (par exemple l'union, l'intersection, ou encore la différence). Les principes de l'algèbre relationnelle sont beaucoup utilisés de nos jours par les SGBD pour la gestion des bases de données informatiques comme le SQL.

Cependant, les bases de données relationnelles ne fonctionnent pas tout à fait selon les règles ensemblistes de l'**algèbre relationnelle**. En effet, si l'on ne définit pas de clé primaire, il est possible d'insérer plusieurs lignes identiques dans une table, ce qui d'un point de vue ensembliste n'a pas de sens : un élément fait partie ou ne fait pas partie d'un ensemble. Si l'on veut appliquer strictement les règles des ensembles, il faut vérifier à chaque ajout dans une table que les lignes ne sont pas déjà présentes.

Afin d'appréhender au mieux l'algèbre relationnelle, il faut connaitre un certain nombre de termes.

2.1. Définitions

- Soit Di avec i de 1 à n l'ensemble des domaines. Un domaine est lui-même un ensemble de données typées. Par exemple, D1 sera l'ensemble des réels, D2 sera l'ensemble des chaînes de caractères de 20 caractères maximum etc. On distinguera les domaines numériques, les domaines basés sur les chaînes de caractères, les domaines basés sur les dates ainsi que les domaines basés sur les types énumérés.
- Dans une base de données relationnelle, on n'a que des relations R_j , j allant de 1 à m. Une relation porte toujours un nom et est constituée d'attributs notés $D_{i,j}$ ou également A_k, k allant de 1 à x. Elle s'écrit ainsi : R ($D_{i,1}$, $D_{i,2}$, $D_{i,3}$ …).

> **Exemple :**
>
> FOURNISSEUR (NoFour, NomFour)

- Les noms R et $D_{i,j}$ constituent le schéma de la relation. Ce schéma, et l'ensemble des éléments possibles de R, constituent une intention de R. Les éléments de R présents à un moment donné constituent une extension de R.

Exemple :

Soit la relation FOURNISSEUR (NoFour, NomFour).
D1 désigne l'ensemble des chaînes de 2 caractères
D2 désigne l'ensemble des chaînes de 50 caractères maximum

NoFour appartient au domaine D1.
NomFour appartient au domaine D2.

Le schéma de la relation est S = { Fournisseur, NoFour, NomFour }
Le schéma en extension de Fournisseur peut se présenter ainsi :

FOURNISSEUR	
NoFour	NomFour
F1	BERTHOS
F2	BAURAND
F3	DUPUIS

- Dans toute relation R il existe une combinaison C d'attributs dite clé, telle que dans tout tuple t d'intention de R, la valeur C(t) identifie t et il n'y a pas de sous-combinaison de C avec cette propriété. Cette notion de clé est fondamentale et constitue le cœur même de l'algèbre relationnelle. Le choix de C est dicté par l'intention de R. R peut avoir plusieurs clés. Dans ce cas, une clé est arbitrairement choisie et est dite primaire et les autres deviennent clés candidates. La clé est, soit soulignée, soit clairement énoncée.

Exemple :

FOURNISSEUR (NoFour, NomFour)
ou
FOURNISSEUR (NoFour, NomFour)
Clé primaire : NoFour

- La clé C d'une relation R peut être des attributs d'une autre relation R' : ils deviennent une clé étrangère dans R'. La clé étrangère est précédée du signe # ou énoncée clairement.

Exemple :

FOURNISSEUR (NoFour, NomFour, #NumType)
TYPE (NumType, LibelléDeType)

ou

FOURNISSEUR (NoFour, NomFour, #NumType)
Clé primaire : NoFour
Clé étrangère : NumType, fait référence à l'attribut NumType de TYPE
TYPE (NumType, LibelléDeType)
Clé primaire : NumType

- Deux relations R et R' sont égales si elles sont constituées des mêmes attributs, nommés également colonnes, et des mêmes tuples, appelés également lignes. Toutefois, elles peuvent différer dans l'ordre des attributs et des tuples.

- Il n'y a pas de tuples égaux dans une relation, cela découle directement de la définition même de la clé primaire.
- Une relation est un fichier qui supporte les opérations relationnelles. Ce sont des opérations qui transforment des relations en une relation résultat.

2.2. Les opérations relationnelles

Afin de présenter les différentes opérations de l'algèbre relationnelle, on s'appuiera sur un schéma relationnel constitué de trois relations.

- Une entreprise a des fournisseurs F. Pour gérer ses fournisseurs, on utilise une relation dont le schéma en intention est fourni ci-dessous :

F			
FNo	**FNom**	**FStatut**	**FVille**
F1	Smith	Ok	Londres
F2	Jones	Ok	Paris
F3	Blake	A éviter	Paris
F4	Clark	A éviter	Londres
F5	Adams	Ok	Athènes

Un fournisseur a un identifiant FNo, un nom FNom, un statut FStatut et est localisé dans une ville FVille.

- Un fournisseur fournit des pièces P au travers de lignes de commandes présentées dans la relation L dont le schéma en extension est le suivant :

L		
FNo	**PNo**	**Qte**
F1	P1	300
F1	P2	200
F1	P3	400
F1	P4	200
F1	P5	100
F1	P6	150
F2	P1	300
F2	P2	400
F3	P2	200
F4	P2	200
F4	P4	300
F4	P5	400

Chaque ligne de commandes comporte un numéro de fournisseur FNo, un numéro de pièces PNo et une certaine quantité d'une pièce P.

- Les pièces fournies sont gérées par une relation dont le schéma en extension est le suivant :

P				
PNo	**PNom**	**PCouleur**	**PPoids**	**PVille**
P1	Ecrou	Rouge	12	Londres
P2	Verrou	Vert	17	Paris
P3	Vis	Bleu	14	Rome
P4	Vis	Rouge	12	Londres
P5	Boulon	Bleu	19	Londres
P6	Rouage	Rouge	19	Rome

Chaque pièce possède un numéro PNo, un nom PNom, une couleur PCouleur, un poids PPoids et est stockée en priorité dans une ville PVille.

a) Les opérateurs unaires

Les expressions algébriques transforment des tables en une table, ce qui explique que les opérateurs que l'on va présenter sont dits « unaires ».

✓ **Projection**

La relation résultante d'une projection est une relation ayant les mêmes tuples mais dont le schéma est réduit à des attributs précisés. Il faut noter que l'opération de projection élimine les doublons, c'est-à-dire les tuples projetés qui peuvent être identiques : cela est tout particulièrement vrai quand la projection se fait sur un seul attribut.

La syntaxe est la suivante :

R = **Projection** (Relation; A_1, A_2,….A_n)
où A_1, A_2….. sont les attributs de la relation R.

Exemple :

Formuler la requête permettant d'obtenir les différentes couleurs des pièces.

R = **Projection** (P; PCouleur)

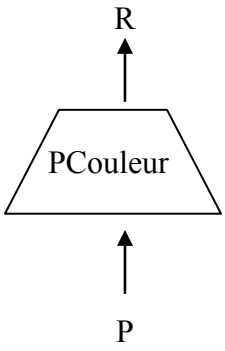
La projection élimine les doublons, ce qui explique que le schéma en extension se limite aux trois couleurs présentes dans la colonne PCouleur.

Exercice d'application :

Formuler la requête permettant d'obtenir les villes où sont situés les fournisseurs.

R1 = **Projection** (F; FVille)

Il existe également une autre syntaxe qui est une représentation graphique et qui se présente ainsi :



Si la représentation graphique est intéressante en termes de lisibilité, elle est malheureusement très lourde à mettre en œuvre dès que la relation à obtenir se complexifie. En conséquence, on l' utilise très peu souvent.

- ✓ **Sélection**

La relation résultante d'une sélection est une relation ayant le même schéma que la relation initiale mais dont les tuples sont ceux de la relation source qui satisfont une condition donnée sur les valeurs des attributs.

La syntaxe est la suivante :

R = **Sélection** (Relation; critère)

Afin d'exprimer les critères, on utilise tous les opérateurs mathématiques et logiques classiques c'est-à-dire >, <, >=, <=, = et l'opérateur logique Non.

Exemple :

Formuler la requête permettant d'obtenir le détail des pièces de couleur rouge.

R = **Sélection** (P; PCouleur = 'Rouge')

Le schéma en extension de R est alors :

R				
PNo	**PNom**	**PCouleur**	**PPoids**	**PVille**
P1	Ecrou	Rouge	12	Londres
P4	Vis	Rouge	12	Londres
P6	Rouage	Rouge	19	Rome

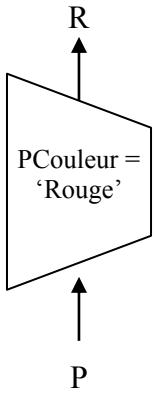
Il faut noter que tous les attributs sont conservés, y compris l'attribut ayant permis la sélection.

Exercice d'application :

Formuler la requête permettant d'obtenir le détail des fournisseurs de Rome.

R1 = **Sélection** (F; FVille = 'Rome')

Il existe également une autre syntaxe qui est une représentation graphique et qui se présente ainsi :



b) Les opérateurs ensemblistes

Les opérateurs ensemblistes sont basés sur le principe d'union-compatibilité. Par définition, deux relations sont union-compatibles si elles ont le même nombre d'attributs et que ceux-ci ont le même domaine.

Ainsi, il ne sera jamais possible de travailler avec des opérateurs ensemblistes sur des tables qui n'ont pas le même nombre d'attributs ou dont les attributs appartiennent à des domaines différents.

- ✓ **Différence**

La différence entre deux relations R2 et R1 ne peut se faire que si les deux relations sont union-compatibles. Cette opération produit une nouvelle relation de schéma identique à R2 possédant les enregistrements présents dans R2 mais pas dans R1.

La syntaxe est la suivante :

R3 = R2 – R1

Exemple :

Formuler la requête permettant d'obtenir le détail des pièces qui ne sont pas de couleur rouge.

R1 = **Sélection** (P; PCouleur='Rouge')
R2 = P – R1

Le schéma en extension de R2 est alors :

R2				
PNo	**PNom**	**PCouleur**	**PPoids**	**PVille**
P2	Verrou	Vert	17	Paris
P3	Vis	Bleu	14	Rome
P5	Boulon	Bleu	19	Londres

Dans cet exemple, on peut également utiliser l'approche classique avec l'opérateur <>. Toutefois, l'approche par différence est souvent nécessaire quand l'opérateur <> ne permet pas de répondre à la question posée.