Algèbre Relationnelle

1. Définition :

L'algèbre relationnelle est un ensemble d'opérateurs qui, à partir d'une ou deux relations existantes, créent en résultat une nouvelle relation temporaire (c'est-à-dire qui a une durée de vie limitée, généralement la relation est détruite à la fin du programme utilisateur ou de la transaction qui l'a créée). La relation résultat a exactement les mêmes caractéristiques qu'une relation de la base de données et peut donc être manipulée de nouveau par les opérateurs de l'algèbre.

Les opérateurs de l'algèbre peuvent être regroupés en deux classes :

-Les opérateurs provenant de la théorie mathématique sur les ensembles **: union, intersection, différence, produit ;**

-Les opérateurs définis spécialement pour les bases de données relationnelles :**sélection, projection, jointure, division et renommage**.

2. Les Opérateurs :

2.1 Union :

L'union de deux relations R1 et R2 de même schéma est une relation R3 de schéma identique qui a pour n-uplets les n-uplets de R1 et/ou R2. On notera : **R3 = R1 UR2**

R1

A	B
0	1
2	3

U

R2

A	B
0	1
4	5

R3= R1 U R2

R3

A	B
0	1
2	3
4	5

2.2 Intersection

L'intersection entre deux relations R1 et R2 de même schéma est une relation R3 de schéma identique ayant pour n-uplets les n-uplets communs à R1 et R2. On notera : **R3 = R1 ∩ R2**

R1

A	B
0	1
2	3

∩

R2

A	B
0	1
4	5

R3= R1 U R2

R3

A	B
0	1

2.3 Différence

La différence entre deux relations R1 et R2 de même schéma est une relation R3 de schéma identique ayant pour n-uplets les n-uplets de R1 n'appartenant pas à R2. On notera :**R3 = R1 − R2**

R1

A	B
0	1
2	3

-

R2

A	B
0	1
4	5

R3= R1 - R2

R3

A	B
2	3

2.4 Projection :

La projection d'une relation R1 est la relation R2 obtenue en supprimant les attributs de R1 non mentionnés puis en éliminant éventuellement les nuplets identiques On notera :

R2 = πR1 (Ai, Aj, ... , Am)la projection d'une relation R1 sur les attributs Ai, Aj, … , Am

 La projection permet d'éliminer des attributs d'une relation

• Elle correspond à un découpage vertical :

A1	A2	A3	A4

Requête 1 :

« Quels sont les références et les prix des produits ? » **PRODUIT (IdPro, Nom, Marque, Prix).**

IdPro	Nom	Marque	Prix
P	PS1	IBM	1000
Q	Mac	Apple	2000
R	PS2	IBM	3000
S	Word	Microsoft	4000

IdPro	Prix
P	1000
Q	2000
R	3000
S	4000

Requête 2 :

« Quelles sont les marques des produits ? » **PRODUIT (IdPro, Nom, Marque, Prix)**

(Notez l'élimination des doublons)

IdPro	Nom	Marque	Prix
P	PS1	IBM	1000
Q	Mac	Apple	2000
R	PS2	IBM	3000
S	Word	Microsoft	4000

ΠProduit (Marque)

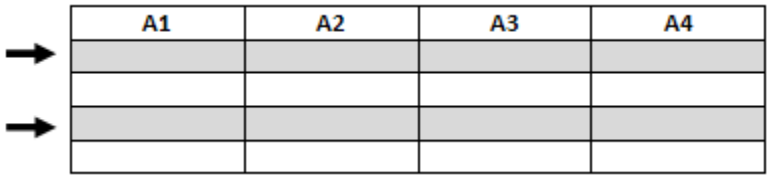
Marque
IBM
Apple
Microsoft

2.5 Restriction :

La restriction d'une relation R1 est une relation R2 de même schéma n'ayant que les n-uplets de R1 répondant à la condition énoncée. On notera : **R2 = σR1 (condition)** la restriction d'une relation R1 suivant le critère "condition" où "condition" est une relation d'égalité ou d'inégalité entre 2 attributs ou entre un attribut et une valeur

 La restriction permet d'extraire les n-uplets qui satisfont une condition

• Elle correspond à un découpage horizontal :



A1	A2	A3	A4

Requête 3 :

« Quelles sont les produits de marque 'IBM' ? » **PRODUIT (IdPro, Nom, Marque, Prix)**

IdPro	Nom	Marque	Prix
P	PS1	IBM	1000
Q	Mac	Apple	2000
R	PS2	IBM	3000
S	Word	Microsoft	4000

σ Produit (marque ='IBM')

IdPro	Nom	Marque	Prix
P	PS1	IBM	1000
R	PS2	IBM	3000

2.6 Jointure :

La jointure de deux relations R1 et R2 est une relation R3 dont les n-uplets sont obtenus en concaténant les nuplets de R1 avec ceux de R2 et en ne gardant que ceux qui vérifient la condition de liaison. On notera : **R3 = R1 × R2 (condition)** la jointure de R1 avec R2 suivant le critère condition

• Le schéma de la relation résultat de la jointure est la concaténation des schémas des opérandes (s'il y a des attributs de même nom, il faut les renommer)

• Les n-uplets de R1 × R2 (condition) sont tous les couples (u1, u2) d'un n-uplet de R1 avec un n-uplet de R2 qui satisfont "condition"

• La jointure de deux relations R1 et R2 est le produit cartésien des deux relations suivi d'une restriction

• La condition de liaison doit être du type : <attribut1>**::**<attribut2> où : attribut1 €1ère relation et attribut2 ∈2ème relation **::** est un opérateur de comparaison (égalité ou inégalité)

➢ La jointure permet de composer 2 relations à l'aide d'un critère de liaison

R1(A, B, C)

A	B	C
A1	B1	10
A2	B2	10
A3	B3	20
A4	B4	30

×

R2(U, V)

U	V
10	V1
20	V2
30	V3

R1 x R2 (R1.C = R2.U)

A	B	C	U	V
A1	B1	10	10	V1
A1	B2	10	10	V1
A3	B3	20	20	V2
A4	B4	30	30	V3

2.7 Jointure naturelle

Jointure où l'opérateur de comparaison est l'égalitédans le résultat on fusionne les 2 colonnes dont les valeurs sontégales

 La jointure permet d'enrichir une relation

Requête 5 :

« Donnez pour chaque vente la référence du produit, sadésignation, son prix, le numéro de client, la date et laquantité vendue »

VENTE As V

IdCli	IdPro	Date	Qte
X	P	1/1/98	1
Y	Q	2/1/98	1
Z	P	3/1/98	1

×

PRODUIT As P

IdPro	Désignation	Prix
P	PS	100
Q	Mac	100

VENTE x PRODUIT (V.idPro = P.idPro)

Idcli	IdPro	Date	Qte	Désignation	Prix
X	P	1/1/98	1	PS	100
Y	Q	2/1/98	1	Mac	100
Z	P	3/1/98	1	PS	100

2.8 Auto-jointure

C'est la jointure d'une relation par elle-même

CLIENT

IdCli	Nom	Ville
X	Smith	Nice
Y	Blake	Paris
Z	John	Nice

Requête 6 :

« Quels sont les noms des clients qui habitent la mêmeville que John ?»

CLIENT As C1

IdCli	Nom	Ville
X	Smith	Nice
Y	Blake	Paris
Z	John	Nice

×

CLIENT As C2

IdCli	Nom	Ville
X	Smith	Nice
Y	Blake	Paris
Z	John	Nice

R1 = CLIENT × CLIENT (C1.Ville = C2.Ville)

R1

C1.IdCli	C1.Nom	Ville	C2.IdCli	C2.Nom
X	Smith	Nice	X	Smith
X	Smith	Nice	Z	John
Y	Blake	Paris	Y	Blake
Z	John	Nice	X	Smith
Z	John	Nice	Z	John

R2 = σ R1 (C2.Nom = 'John')

R2

C1.IdCli	C1.Nom	Ville	C2.IdCli	C2.Nom
X	Smith	Nice	Z	John
Z	John	Nice	Z	John

R3 = π R2 (C1.Nom)

R3

C1.Nom
Smith
John

3. Langage Algébrique

Le langage algébrique permet de formuler une questionpar une suite d'opérations de l'algèbre relationnelle

Requêtes sur le schéma CLIENT, PRODUIT, VENTE

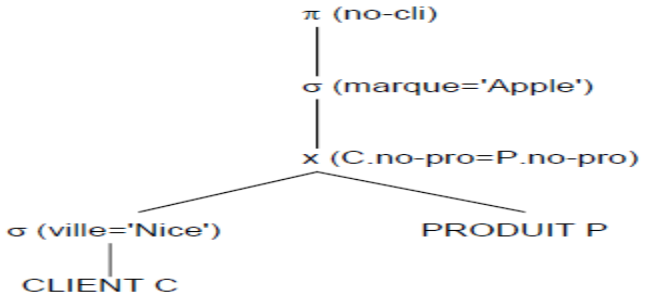
- **CLIENT** (**IdCli**, nom, ville)
- **PRODUIT** (**IdPro**, désignation, marque, prix)
- **VENTE** (**IdCli**, **IdPro**, **date**, qte)

Requête 8 :
« Donner les N° des produits de marque Apple et de prix <5000 »
R1 = σPRODUIT (marque = Apple')
R2 = σPRODUIT (prix < 5000)
R3 = R1∩R2
RESUL = πR3 (IdPro)

Requête 9 :
« Donner les N° des clients ayant acheté un produit de marqueApple »
R1 = σPRODUIT (marque = 'Apple'**)**
R2 = R1×VENTE (R1.IdPro = VENTE.IdPro**)**
RESUL = πR2 (IdCli**)**

Requête 10 :
« Donner les no des clients n'ayant acheté que des produits demarque Apple »
R1 = VENTE×PRODUIT (VENTE.IdPro = PRODUIT.IdPro**)**
R2 = σR1 (marque = 'Apple'**)**
R3 = πR2 (IdCli**)**
R4 = σR1 (marque ≠ 'Apple'**)**
R5 = πR4 (IdCli**)**
RESUL = R3 − R5

4. Arbre algébrique
Une question peut être représentée par un arbre
« Quels sont les clients de Nice ayant acheté un produit de marque 'Apple' ? »



5. Optimisation de requêtes
Plusieurs arbres équivalents peuvent être déduits d'un arbre donné à l'aide de règles de transformation simples, telles que permutation des jointures et restrictions, permutation des projections et des jointures, etc.
Ces transformations sont à la base des techniques **d'optimisation de requêtes**

Langage SQL

1. Définition :

SQL est un langage de manipulation de bases de données mis au point dans les années 70 par IBM, SQL est devenu le langage standard pour décrire etmanipuler les **BDR**. Il permet, trois types de manipulations sur les bases de données :

- **La maintenance des tables** : création, suppression, modification de la structure des tables.
- **Les manipulations des bases de données** : Sélection, modification, suppression d'enregistrements.
- **La gestion des droits d'accès aux tables** : Contrôle des données : droits d'accès, validation des modifications

2. Importance du langage SQL :

• Standard d'accès aux serveurs de donnéesrelationnels, norme **ISO**
• SQL est le langage commun de nombreux systèmescommercialisés
• SQL est l'interface logiciel/logiciel entre lesapplications et les BDR

3. Opérateurs de base

3.1 La projection

La projection permet d'extraire des données d'une table, en ne conservant que les colonnes souhaitées.

Notation :

R2 = **PROJECTION** R1 (Nom-Champ, Nom-Champ…)
R1 et R2 sont deux relations, entre parenthèses figurent les critères de projection.

Instructions SQL :

SELECT Nom-Table.Nom-Champ, Nom-Table.Nom-Champ…
FROM Nom-Table**;**

3.2 La Sélection

Permet d'extraire les lignes d'une table qui vérifient la réalisation d'une certaine condition (on parleparfois de critère).

Notation :

R2 = **SELECTION** R1 (Expression conditionnelle)
R1 et R2 sont deux relations, entre parenthèses figure le critère de sélection.

Instructions SQL :

SELECT Nom-Table.Nom-Champ, Nom-Table.Nom-Champ…
FROM Nom-Table
WHERE Critère de sélection**;**
Opérateurs de comparaison : = != <>>>= <<=

3.3 La jointure

La jointure consiste donc à combiner deux tables ligne à ligne en vérifiant la concordance entrecertaines colonnes des deux tables. Autrement dit, cela permet de relier deux tables ayant un champcommun et de faire correspondre les lignes qui ont une même valeur.

Notation :

R3 = R1 >< R2 (Expression conditionnelle)
Ou
R3 = JOINTURE R1.R2 (Expression conditionnelle)
R1 et R2 sont deux relations, entre parenthèses figure le critère de jointure.

Instructions SQL :

SELECT Nom-Table.Nom-Champ, Nom-table.Nom-Champ…
FROM Nom-Table, Nom-Table
WHERE Condition de jointure**;**
Opérateurs de comparaison : = != <>>>= <<=

3.4 Combinaison des opérateurs de base

Une requête d'extraction de données sur une base de données est la plupart du temps une combinaison astucieuse des trois opérations de base.
On peut donc selon les besoins combiner les instructions. L'opérateur AND permet d'avoir plusieursconditions de sélection. L'instruction ORDER BY permet de trier selon l'ordre d'un des champs(ASCendant ou DESCendant).

Exemple de séquence d'opérations algébriques :

R3 = R1 ><R2 (Condition de Jointure)
R4 = **SELECTION** R3 (Critère de sélection **ET** Critère de sélection)
R5 = **SELECTION** R4 (Critère de sélection)
R6 = **PROJECTION** R5 (Nom-Champ, Nom-Champ, …)

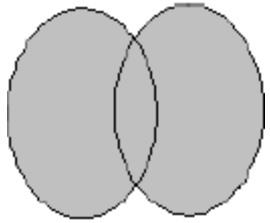
Exemple de séquence d'instructions SQL :

SELECT Nom-Table.Nom-Champ, Nom-table.Nom-Champ…
FROM Nom-Table, Nom-Table
WHERE Condition de jointure
AND Condition
ORDER BY Nom-Table.Nom-Champ **ASC;**

4. Les opérateurs ensemblistes

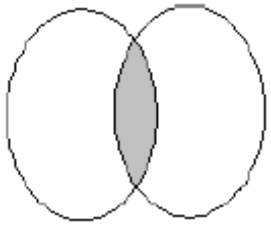
4.1 L'union (ou / or)

L'union consiste à combiner deux relations (compatibles) pour créer unetroisième relation qui contient toutes les occurrences appartenant à l'une ou àl'autre des relations de départ.



Notation :
R3 = R1UR2OuR3 = **UNION** (R1, R2)
Instructions SQL :
SELECT Nom-Champ, Nom-Champ, …
FROM Nom-Table1
UNION
SELECT Nom-Champ, Nom-Champ, …
FROM Nom-Table2**;**
On peut utiliser **UNION [ALL**] pour avoir toutes les lignes communes aux deux tables (y compris cellesen double), sans cela les doublons sont éliminés.

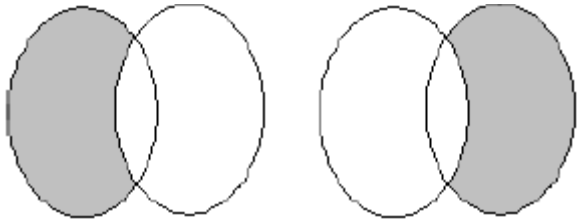
D.2. L'intersection (et / and)
L'intersection consiste à combiner deux relations (compatibles) pour créer unetroisième relation qui contient toutes les occurrences appartenant à l'une et àl'autre des relations de départ.
Notation :
R3 = R1 ∩ R2OuR3 = **INTERSECTION** (R1, R2)



Instructions SQL :
SELECT Nom-Champ, Nom-Champ, …
FROM Nom-Table1
INTERSECT
SELECT Nom-Champ, Nom-Champ, …
FROM Nom-Table2**;**

D.3. La différence (non / not)
La différence consiste à combiner deux relations(compatibles) pour créer une troisième relation quicontient toutes les occurrences appartenant à l'une desrelations et non contenues dans l'autre des relations dedépart. Deux différences sont possibles.
Notation :
R3 = R1 **-** R2 R3 = R2 **–** R1
Ou
R3 = **DIFFERENCE** (R1, R2) R3 = **DIFFERENCE** (R1, R2)
Instructions SQL :

SELECT Nom-Champ, Nom-Champ, …**SELECT** Nom-Champ, Nom-Champ, …
FROM Nom-Table2 **FROM** Nom-Table1
MINUS MINUS
SELECT Nom-Champ, Nom-Champ, … **SELECT** Nom-Champ, Nom-Champ, …
FROM Nom-Table1**; FROM** Nom-Table2**;**



Les Commandes :
SELECT
SELECT P.prix
FROM produit P
WHERE P.IdPro = 'p1'
La commande **SELECT** permet de rechercher des données à partir de plusieurs tables ; le résultat est présenté sous forme d'une table réponse

Expression des projections
Q1 Donner les noms, marques et prix des produits
SELECT P.nom, P.marque, P.prix
FROM produit P

Q2 Donner les différentes marques de produit
SELECT DISTINCT P.marque
FROM produit P
- ❖ Contrairement à l'algèbre relationnelle, SQL n'élimine pas les doublons
- ❖ Pour éliminer les doublons il faut spécifier **DISTINCT.**

Q3 Donner les références des produits et leurs prix majorés de 20%
SELECT P.IdPro, P.prix * 1.20
FROM produit P
- ❖ Il est possible d'effectuer des opérations arithmétiques (+, -, *,) sur les colonnes extraites

Q4 Donner tous les renseignements sur les clients
SELECT *
FROM client
Une étoile (*) permet de lister tous les attributs
• **Expression des restrictions**
Q5 Donner les noms des produits de marque IBM
SELECT P.nom
FROM produit P
WHERE P.marque = 'IBM'
- La condition de recherche (qualification) est spécifiéeaprès la clause **WHERE** par un prédicat

❖ Un prédicat simple peut-être :
- un prédicat d'égalité ou d'inégalité **(=, <>, <, >, <=, >=)**
- un prédicat **LIKE**
- un prédicat **BETWEEN**
- un prédicat **IN**
- un test de valeur **NULL**
- un prédicat **EXISTS**
- un prédicat **ALL** ou **ANY**

❖ Un prédicat composé est construit à l'aide des connecteurs **AND**, **OR** et **NOT**

❖ **Exemples**

Q6 Lister les clients dont le nom comporte la lettre A en 2ième position
```
SELECT *
FROM client C
WHERE C.nom LIKE '_A%'
```
Le prédicat **LIKE** compare une chaîne avec un modèle
(_) remplace n'importe quel caractère
(%) remplace n'importe quelle suite de caractères

Q7 Lister les produits dont le prix est compris entre 5000DAet 12000DA
```
SELECT *
FROM produit P
WHERE P.prix BETWEEN 5000 AND 12000
```
Le prédicat **BETWEEN** teste l'appartenance à un intervalle.

Q8 Lister les produits de marque IBM, Apple ou Dell
```
SELECT *
FROM produit P
WHERE P.marque IN ('IBM', 'Apple', 'Dell')
```
Le prédicat **IN** teste l'appartenance à une liste de valeurs

Q9 Lister les produits dont le prix est inconnu
```
SELECT *
FROM produit P
WHERE P.prix IS NULL
```
La valeur **NULL** signifie qu'une donnée est inconnue.

Q10 Lister les produits de marque IBM dont le prix est inférieur à 12000DA
```
SELECT *
FROM produit P
WHERE P.marque = 'IBM' AND P.prix < 12000
```
Le connecteur **AND**relie les 2 prédicats de comparaison

✓ **Tri du résultat d'un SELECT**

La clause **ORDER BY** permet de spécifier les colonnesdéfinissant les critères de tri
Le tri se fera d'abord selon la première colonnespécifiée, puis selon la deuxième colonne etc...

Exemple

Q11 Lister les produits en les triant par marques et à l'intérieur d'une marque par prix décroissants
```
SELECT *
FROM produit P
ORDER BY P.marque, P.prix DESC
```
❖ L'ordre de tri est précisé par ASC (croissant) ou DESC(décroissant) ; par défaut ASC

Résumé : Le Langage SQL (Structured Query Language)

SELECT	Précise les colonnes qui vont apparaître dans la réponse
FROM	Précise la (ou les) table intervenant dans l'interrogation
WHERE	Précise les conditions à appliquer sur les lignes. On peut trouver : - Des comparateurs : =, >, <, >=, <=,<> - Des opérateurs logiques : AND, OR, NOT - Les prédicats : IN, LIKE, NULL, EXISTS...
GROUP BY	Précise la (ou les) colonne de regroupement
HAVING	Précise la (ou les) conditions associées à un regroupement
ORDER BY	Précise l'ordre dans lequel vont apparaître les lignes de la réponse : - **ASC** : En ordre ascendant (par défaut) - **DESC**: En ordre descendant

Soit le schéma de la base de données **Ventes**

Client (**NumClient**, NomClient, AdrClient, TélClient, EmailClient,

NumRegClient, #CodeVille)

Ville (**CodeVille**, NomCodeVille)

Commande (**NumCom**, DateCom, #NumClient)

Produit (**RefProd**, DésignProd, DateProduct, DatePerempt)

Facture (**NumFact**, DateFact, #NumCom)

ComProd (**NumCom, RefProd**, QtéCom)

FactProd (**NumFact, RefProd** , PUProd, QtéFact)

Projection

- La liste des NomClients et NumRegClient de tous les Clients :

Select *NomClient, NumRegClient* ***From****Client*

Pour Lister tous les champs de la table Client, on écrit : Select * **FromClient**

Sélection : Where

- Liste de tous les clients dont le code CodeVille est 1 :
  Select * ***From*** *CLIENT* ***Where*** *CodeVille = 1*

Conditions composées : Les conditions composées permettent l'utilisation des opérateurs booléens **AND, OR, NOT**.

- Liste de tous les Clients ayant le code ville 1 ou 2 : ***Select*** * ***From*** *CLIENT* ***Where*** *CodeVille = 1* ***or*** *CodeVille = 2*

- Les Clients dont le CodeVille = 19 et le NumRegClient = 23005 :
  *Select ** ***From****Client Where CodeVille = 19 and NumRegClient = '23005'*

Utilisation d'autres opérateurs

- **BETWEEN** : Opérateur d'intervalle. Liste de tous les Clients ayant le code ville entre 19 et 41
  *Select ** ***From*** *Client Where CodeVille between 19 and 41*

- **IN** : Opérateur d'ensemble : Liste de tous les Clients habitant dans les CodeVilles 19, 23, 41, 80
  *Select ** ***From****Client Where CodeVille in (19, 23, 41, 80)*

- **IS NULL** : Liste de tous les Clients n'ayant pas le téléphone.
  *Select ** ***From****Client Where TélClient is null*

- **IS NOT NULL** : Liste de tous les Clients ayant le téléphone.
  *Select ** ***From****Client Where TélClient is not null*

Jointure :

❖ Afficher les noms des clients ainsi que leurs noms de villes ?**(Jointure entre 2 tables)**

Select *NomClient, NomVille****From*** *Client, Ville* ***Where*** *Client.CodeVille = Commande. CodeVille*

❖ Quel est le NomClient du Client ayant passé la commande numéro 3 ?**(Jointure entre 2 tables, avec sélection)**

Select *NomClient* ***From*** *Client, Commande****Where*** *Client.NumClient = Commande.NumClient* ***and*** *NumCom = 3*

❖ Afficher les noms des clients ainsi que leurs numéros de factures ?**(Jointure entre 3tables)**

Select *NomClient, NumFact* ***From*** *Client, Commande, Facture* ***Where*** *Client.NumClient = Commande.NumClient* ***and****Commande.NumCom = Facture.NumCom*

Organisation des résultats d'une requête : Order by

❖ Liste des Clients (NomClient, NumRegClient) par ordre croissant des NumRegClient

Select *NomClient, NumRegClient* ***From****Client* ***Order by*** *NumRegClient*

❖ Liste des Clients (NomClient, NumRegClient) par ordre décroissant des NumRegClientet croissant des NomClient :

Select *NomClient, NumRegClient* ***From*** *Client* ***Order by****NumRegClient* ***Desc****, NomClient* ***Asc***

Regroupement de valeurs :Group by/ Having

❖ **Fonction de comptage (simple) :** Combien de Clients sont référencés par l'entreprise ?

Select count*(*)* ***as*** *NbClient* ***From*** *Client*

❖ Afficher le nombre de commandes par nom de client.

Select *NomClient, count(*)* ***as*** *NbCom* ***From*** *Client, Commande* ***Where*** *Client.NumClient = Commande.NumClient* ***Group by****NomClient*

❖ Afficher le nombre de factures par nom de client.

Select *NomClient, count(*)* ***as*** *NbFact* ***From*** *Client, Commande, Facture* ***Where*** *Client.NumClient = Commande.NumClient* ***and*** *Commande.NumCom = Facture.NumCom* ***Group by****NomClient*

La clause **GROUP BY** va permettre le regroupement ou partitionnement du résultat d'une requête en fonction d'une ou plusieurs rubriques faisant parties du **SELECT**.
On peut ajouter à la clause de regroupement une condition de recherche par l'intermédiaire d'une clause dépendante **HAVING** qui va spécifier les conditions de sélection des regroupements. **HAVING** est toujours associé au **GROUP BY**

❖ Afficher les noms des clients qui ont plus de (>) 2 commandes.

Select *NomClient* ***From****Commande*
Group by *NomClient*
Having count*(*) > 2*

❖ Afficher le nombre de commandes par nom et numéro de téléphone des clients dont les numéros de téléphones commencent par '05'.

Select *NomClient, TélClient, count(*)* ***as*** *NbCom*
From *Client, Commande*
Where *Client.NumClient = Commande.NumClient*
Group by *NomClient, TélClient*
Having *TélClient* ***like*** *'05*'*

Fonctions de calculs (d'agrégation : Somme, moyenne, min, max)

❖ **Fonction Somme :** Calculez le **montant HT**par facture?

Select *NumFact****, sum****(Montant)* ***as****MontantHT* ***From*** *FactProd* *Group by NumFact*

❖ **Fonction moyenne : Avg :**Calculez le **montant moyen**par facture ?

Select *NumFact****, Avg****(Montant)* ***as****MoyMontant* ***From*** *FactProd* *Group by NumFact*

❖ **Fonction min/max** *:* Permet de rechercher le minimum/maximum d'une série de valeurs de type numérique.

Select *NumFact* ***, min****(Montant)****as*** *MinMont,* ***max****(Montant)* ***as*** *MaxMont*
From*FactProd Group by NumFact*

Suppression de données : Delete

❖ *Supprimez les données du client numéro 6*

Delete *** ***From*** *Client* ***Where****NumClient = 6*

Mise à jour (modification) des données : Update

❖ *Saisissez le nouveau numéro (0558121345) de téléphone du Client 1*

UPDATE*Client*
SET*TélClient =' 0558121345'*
WHERE*NumClient = 6*