

Institut Supérieur des Etudes Technologiques de Sfax

## Chapitre 5

# Langage SQL



© Borchani Anis  
AU: 2016-2017

- Contenu
  - LDD
  - LMD
  - LCD
  - LID



6 – Langage SQL

## Introduction

- SQL (Structured Query Language) est un langage informatique normalisé servant à exploiter des bases de données.
- SQL est basée sur l'algèbre relationnelle (opérations ensemblistes et relationnelles).
- SQL conçu par IBM dans les années 70, a été normalisé dès 1986 mais les premières normes, trop incomplètes, ont été ignorées par les éditeurs de SGBD.
- La norme actuelle SQL-2011 est acceptée par tous les SGBD relationnels.
- Ce langage permet l'accès aux données et se compose de quatre sous-ensembles :

Borchani Anis 83

6 – Langage SQL

## Introduction

**LDD** : « Langage de Définition de Données » : permet la définition et la mise à jour de la structure de la base de données (tables, attributs, vues, index, ...).

**LID** : « Langage d'Interrogation de Données » : permet de rechercher des informations utiles en interrogeant la base de données.

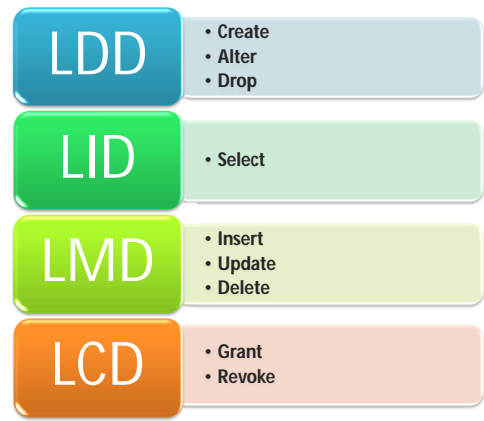
**LMD** : « Langage de Manipulation de Données » : permet de manipuler les données de la base et de les mettre à jour.

**LCD** : « Langage de Contrôle de Données » : permet de définir les droits d'accès pour les différents utilisateurs de la base de données, donc il permet de gérer la sécurité de la base et de confirmer et d'annuler les transactions.

Borchani Anis 84

6 – Langage SQL

## Introduction



Borchani Anis 85

6 – Langage SQL

### Exemple de base de données

**Produit** (NP, LibP, Coul, Poids, PU, Qtes) - Désigne l'ensemble des produits.

**Client** (NCI, NomCI, ADRCI) - Désigne l'ensemble des clients.

**Commande** (NCmd, DateCmd, #NCI) - Désigne l'ensemble des commandes.

**Ligne\_Cmd** (#NCmd, #NP, Qte) - Désigne l'ensemble des lignes de commandes.

Borchani Anis 86

6 – Langage SQL

### Exemple de base de données

Client		
NCI	NomCI	AdrCI
CL01	BATAM	Tunis
CL02	BATIMENT	Tunis
CL03	AMS	Sousse
CL04	GLOULOU	Sousse
CL05	PRODELEC	Tunis
CL06	ELECTRON	Sousse
CL07	SBATIM	Sousse
CL08	SANITAIRE	Tunis
CL09	SOUDURE	Tunis
CL10	MELEC	Monastr
CL11	MBATIM	Tunis
CL12	BATFER	Tunis

Commande		
NCmd	DateCmd	NCI
C001	10/12/2003	CL02
C002	13/02/2004	CL05
C003	15/01/2004	CL03
C004	03/09/2003	CL10
C005	11/03/2004	CL03

Ligne_Cmd			
NCmd	NP	Qte	
C001	P001	250	
C001	P004	300	
C001	P006	100	
C002	P002	200	
C002	P007	550	
C003	P001	50	
C004	P002	100	
C004	P004	150	
C004	P005	70	
C004	P008	90	
C005	P001	650	
C005	P002	100	

Produit					
NP	LibP	Coul	Poids	PU	Qtes
P001	Robinet	Gris	5	18.000	1200
P002	Prise	Blanc	1.2	1.500	1000
P003	Câble	Blanc	2	25.000	1500
P004	Peinture	Blanc	25	33.000	900
P005	Poignée	Gris	3	12.000	1300
P006	Serrure	Jaune	2	47.000	1250
P007	Verrou	Gris	1.7	5.500	2000
P008	Fer	Noir	50	90.000	800

Borchani Anis 87

6 – Langage SQL

### Types de données

TYPE	DESCRIPTION	Minimum	Maximum	Exemples
VARCHAR2 (taille) NVARCHAR2 (taille)	Chaîne de caractères de longueur variable	1 car.	2000 car.	'A' 'Bonjour DD'
NUMBER [(taille[,precision])]	Numérique. (prec<=38, exposant max -84 +127)	10 exp -84	10 exp 127	10.9999
LONG (taille)	Chaîne de caractères de longueur variable.	1 car.	2 giga car.	'AAAAHHHHH...HH H'
DATE	Date (du siècle à la seconde)	01/01/-4712 (avant J.C)	31/12/9999	'10-FEB-04' '10/02/04'
RAW (taille)	Données binaires devant être entrées en notation hexadécimale. Taille : 1 à 255 caractères	1 octet	2000 octets	
LONG RAW (taille)	Données binaires devant être entrées en notation hexadécimale.	1 octet	2 GO	
ROWID	Type réservé à la pseudo-colonne ROWID. Ne peut être utilisé.			
CHAR (taille) NCHAR (taille)	Chaîne de caractères de longueur fixe	1 car.	255 car.	'AZERTY' 'W'
BLOB	BLOB ! gros objet binaire	1 octet	4 giga car.	
TIMESTAMP	Date et heure			

qlqs types valides pour Oracle

Borchani Anis 88

6 – Langage SQL

### Contraintes d'intégrité

Une contrainte d'intégrité est une règle qui permet de contrôler la validité et la cohérence des valeurs entrées dans les différentes tables de la base.

Elle peut être définie sous deux formes :

- Dans les commandes de création des tables.
- Au moment de la modification de la structure de la table.

Borchani Anis 89

## Contraintes d'intégrité sur un attribut

La contrainte porte sur un seul attribut.

Ces contraintes sont :

**NOT NULL** : Spécifie que pour toute occurrence, l'attribut doit avoir une valeur (la saisie de ce champ est obligatoire).

**UNIQUE** : Toutes les valeurs de l'attribut sont distinctes.

**PRIMARY KEY** : L'attribut est une clé primaire pour la table.

**REFERENCES** table (attribut) : Il s'agit d'une contrainte d'intégrité fonctionnelle par rapport à une clé ; chaque valeur de l'attribut doit exister dans la table dont l'attribut est référencée. On utilise cette contrainte pour les clés étrangères.

**CHECK** : C'est une contrainte associée à une condition qui doit être vérifiée par toutes les valeurs de l'attribut (domaine des valeurs de l'attribut).

Borchani Anis 90

## Contraintes d'intégrité sur une table

La contrainte porte sur un ensemble d'attributs d'une même table.

Ces contraintes sont :

**UNIQUE (attri, attrj,...)** : L'unicité porte sur le n-uplet des valeurs.

**PRIMARY KEY (attri, attrj,...)** : Clé primaire de la table (clé composée).

**FOREIGN KEY (attri, attrj, ...) REFERENCES** table (attrm, attrn, ...) **[ON DELETE { CASCADE | SET NULL }]** : Désigne une clé étrangère sur plusieurs attributs.

L'option ON DELETE CASCADE indique que la suppression d'une ligne de la table de référence va entraîner automatiquement la suppression des lignes référencées.

Borchani Anis 91

## Création d'une table (1/3)

### Syntaxe

```
CREATE TABLE nom_table
(Attribut1 Type [Contrainte d'attribut],
Attribut2 Type [Contrainte d'attribut],
...
Attributn Type [Contrainte d'attribut],
[Contrainte de relation], ...);
```

[.....]: optionnel

Borchani Anis 92

## Création d'une table (2/3)

### Application

Création de la table Client en donnant à sa clé NCI les propriétés NOT NULL et UNIQUE.

```
CREATE TABLE Client
(NCI VARCHAR2(4) NOT NULL UNIQUE,
NomCI VARCHAR2(15),
AdrCI VARCHAR2(10));
```

Création de la table Produit en définissant la contrainte pk\_NP de clé primaire équivalente à l'attribution des propriétés NOT NULL et UNIQUE. Ne pas définir l'attribut Otes ceci sera défini ultérieurement par l'instruction ALTER TABLE.

```
CREATE TABLE Produit
(NP VARCHAR2(4) CONSTRAINT pk_NP PRIMARY KEY,
LibP VARCHAR2(15),
Coul VARCHAR2(10),
Poids NUMBER(6,3),
PU NUMBER(6,3));
```

Borchani Anis

6 – Langage SQL

## Création d'une table (3/3)

*Application*

Création de la table Commande sans définir de contrainte ni de propriétés sur la clé, ceci sera défini ultérieurement par l'instruction ALTER TABLE.

```
CREATE TABLE Commande
(NCmd VARCHAR2(4),
DateCmd DATE,
NCI VARCHAR2(4));
```

Création de la table Ligne\_Cmd en spécifiant la contrainte de clé primaire et l'une des contraintes de clé étrangère.

```
CREATE TABLE Ligne_Cmd
(NCmd VARCHAR2(4),
NP VARCHAR2(4),
Qte NUMBER(5),
CONSTRAINT pk_LCMD PRIMARY KEY (NCmd, NP),
CONSTRAINT fk_NP FOREIGN KEY (NP) REFERENCES Produit(NP));
```

Borchani Anis

6 – Langage SQL

## Modification de la structure d'une table (1/6)

**Ajout d'un attribut** : Permet d'ajouter un attribut à la structure initiale de la table.

*Syntaxe :*  
**ALTER TABLE** Nom\_Table  
**ADD** Attribut Type;

*Application :*  
Ajout de l'attribut Qtes à la table Produit.

```
ALTER TABLE Produit
ADD Qtes NUMBER(5);
```

Borchani Anis

95

6 – Langage SQL

## Modification de la structure d'une table (2/6)

**Modification d'un attribut** : Associée avec la clause **MODIFY**, la clause ALTER permet la modification du type de données d'un attribut. On ne peut qu'agrandir la taille d'un attribut.

*Syntaxe :*  
**ALTER TABLE** Nom\_Table  
**MODIFY** Attribut Nouveau\_Type;

*Application :*  
Modification de la taille de l'attribut LibP à VARCHAR(20).

```
ALTER TABLE Produit
MODIFY LibP VARCHAR(20);
```

Borchani Anis

96

6 – Langage SQL

## Modification de la structure d'une table (3/6)

**Suppression d'un attribut** : Permet de supprimer un attribut d'une table.

*Syntaxe :*  
**ALTER TABLE** Nom\_Table  
**DROP COLUMN** Attribut;

Il faut noter que la suppression d'attributs (colonnes) n'est possible que dans le cas où :

- L'attribut ne fait pas partie d'une vue,
- L'attribut ne fait pas partie d'un index,
- L'attribut n'est pas l'objet d'une contrainte d'intégrité.

Borchani Anis

97

## Modification de la structure d'une table (4/6)

**Ajout de contrainte:** Permet d'ajouter une contrainte au niveau d'une table.

*Syntaxe :*

```
ALTER TABLE Nom_Table
ADD CONSTRAINT Nom_Contrainte Définition_Contrainte;
```

*Applications :*

*Ajout de la contrainte de PRIMARY KEY sur l'attribut NCmd de la table Commande.*

```
ALTER TABLE Commande
ADD CONSTRAINT pk_NCMD PRIMARY KEY (NCmd);
```

*Ajout des contraintes de clés étrangères sur la table Commande.*

```
ALTER TABLE Commande
ADD CONSTRAINT fk_NCL FOREIGN KEY (NCI) REFERENCES Client(NCI);
```

## Modification de la structure d'une table (5/6)

**Ajout de contrainte:**

*Applications :*

*Ajout des contraintes de clés étrangères sur la table Ligne\_Cmd.*

```
ALTER TABLE Ligne_Cmd
ADD CONSTRAINT fk_NCMD FOREIGN KEY (NCmd) REFERENCES
Commande(NCmd);
```

*Ajout de la contrainte CHECK sur l'attribut Qte (Qte > 0) de la table Ligne\_Cmd.*

```
ALTER TABLE Ligne_Cmd
ADD CONSTRAINT ck_QTE CHECK (Qte > 0);
```

## Modification de la structure d'une table (6/6)

**Suppression de contrainte :** Permet de supprimer une contrainte.

*Syntaxe :*

```
ALTER TABLE Nom_Table
DROP CONSTRAINT Nom_Contrainte;
```

**Désactivation d'une contrainte :** Permet de désactiver une contrainte, elle est par défaut active (au moment de sa création).

*Syntaxe :*

```
ALTER TABLE Nom_Table
DISABLE CONSTRAINT Nom_Contrainte;
```

**Activation d'une contrainte :** Permet d'activer une contrainte désactivée.

*Syntaxe :*

```
ALTER TABLE Nom_Table
ENABLE CONSTRAINT Nom_Contrainte;
```

## Création d'index

- La création d'un index permet d'accélérer les recherches d'informations dans la base.
- La ligne est retrouvée instantanément si la recherche peut utiliser un index.
- Sinon la recherche se fait séquentiellement.
- Une autre utilité de la création d'index est de garantir l'unicité de la clé en utilisant l'option UNIQUE.

*Syntaxe :*

```
CREATE [UNIQUE] INDEX nom_index
ON nom_table (Attr1[ASC/DESC], Attr2[ASC/DESC], ...);
```

- L'option **UNIQUE** permet de définir la présence ou non de doublons pour les valeurs de l'attribut.
- Les options **ASC/DESC** permettent de définir un ordre de classement des valeurs présentes dans l'attribut.

## Suppression d'une table / un index

La clause **DROP** permet de supprimer des vues, des index et même des tables. Cette clause est toutefois à utiliser avec précaution dans la mesure où elle est irréversible.

Syntaxe :

Pour supprimer un index : **DROP INDEX** Nom\_Index [ON Nom\_Table];

Pour supprimer une table : **DROP TABLE** Nom\_table ;

NB :

- Le nom de la table est obligatoire si on veut supprimer un index d'une table d'un autre utilisateur.
- Un index est automatiquement supprimé dès qu'on supprime la table à laquelle il appartient.

## Langage de Manipulation de Données

Le LMD est un ensemble de commandes permettant de:

- l'insertion de nouvelles données. (**Insert**)
- la modification. (**Update**)
- la suppression de données existantes. (**Delete**)

## Insertion de données

Syntaxe :

```
INSERT INTO Nom_Table [(Attr1, Attr2,..., Attrn)]
```

```
VALUES (Val1, Val2,..., Valn);
```

ou

```
INSERT INTO Nom_Table (Attr1, Attr2, ..., Attrn)
```

```
SELECT...;
```

Permet d'insérer un tuple à la fois.

Permet d'insérer plusieurs tuple à partir d'une ou plusieurs autres tables.

L'ordre INSERT attend la clause INTO, suivie du nom de la table, ainsi que du nom de chacun des attributs entre parenthèses (les attributs omis prendront la valeur NULL par défaut).

Les données sont affectées aux attributs (colonnes) dans l'ordre dans lequel les attributs ont été déclarés dans la clause INTO.

## Insertion de données

Application :Remplissage de la table Client.

```
INSERT INTO Client VALUES('CL01','BATAM','Tunis');
```

```
INSERT INTO Client VALUES('CL02','BATIMENT','Tunis');
```

```
INSERT INTO Client VALUES('CL03','AMS','Sousse');
```

```
INSERT INTO Client VALUES('CL04','GLOULOU','Sousse');
```

```
INSERT INTO Client VALUES('CL05','PRODELEC','Tunis');
```

```
INSERT INTO Client VALUES('CL06','ELECTRON','Sousse');
```

```
INSERT INTO Client VALUES('CL07','SBATIM','Sousse');
```

```
INSERT INTO Client VALUES('CL08','SANITAIRE','Tunis');
```

```
INSERT INTO Client VALUES('CL09','SOUDURE','Tunis');
```

```
INSERT INTO Client VALUES('CL10','MELEC','Monastir');
```

```
INSERT INTO Client VALUES('CL11','MBATIM','');
```

```
INSERT INTO Client VALUES('CL12','BATFER','Tunis');
```

6 – Langage SQL

## Insertion de données

*Application :*

*Remplissage de la table Produit.*

```

INSERT INTO Produit VALUES('P001','Robinet','Gris',5,18,1200);
INSERT INTO Produit VALUES('P002','Prise','Blanc',1.2,1.5,1000);
INSERT INTO Produit VALUES('P003','Cable','Blanc',2,25,1500);
INSERT INTO Produit VALUES('P004','Peinture','Blanc',25,33,900);
INSERT INTO Produit VALUES('P005','Poignée','Gris',3,12,1300);
INSERT INTO Produit VALUES('P006','Serrure','Jaune',2,47,1250);
INSERT INTO Produit VALUES('P007','Verrou','Gris',1.7,5.5,2000);
INSERT INTO Produit VALUES('P008','Fer','Noir',50,90,800);

```

Borchani Anis 106

6 – Langage SQL

## Insertion de données

*Application :*

*Remplissage de la table Commande.*

```

INSERT INTO Commande VALUES('C001', '10/12/2003', 'CL02');
INSERT INTO Commande VALUES('C002', '13/02/2004', 'CL05');
INSERT INTO Commande VALUES('C003', '15/01/2004', 'CL03');
INSERT INTO Commande VALUES('C004', '03/09/2003', 'CL10');
INSERT INTO Commande VALUES('C005', '11/03/2004', 'CL03');

```

Borchani Anis 107

6 – Langage SQL

## Insertion de données

*Application :*

*Remplissage de la table Ligne\_Cmd.*

```

INSERT INTO Ligne_Cmd VALUES('C001','P001',250);
INSERT INTO Ligne_Cmd VALUES('C001','P004',300);
INSERT INTO Ligne_Cmd VALUES('C001','P006',100);
INSERT INTO Ligne_Cmd VALUES('C002','P002',200);
INSERT INTO Ligne_Cmd VALUES('C002','P007',550);
INSERT INTO Ligne_Cmd VALUES('C003','P001',50);
INSERT INTO Ligne_Cmd VALUES('C004','P002',100);
INSERT INTO Ligne_Cmd VALUES('C004','P004',150);
INSERT INTO Ligne_Cmd VALUES('C004','P005',70);
INSERT INTO Ligne_Cmd VALUES('C004','P008',90);
INSERT INTO Ligne_Cmd VALUES('C005','P001',650);
INSERT INTO Ligne_Cmd VALUES('C005','P002',100);

```

Borchani Anis 108

6 – Langage SQL

## Modification de données

*Syntaxe :*

```

UPDATE Nom_Table
SET Attr1 = Expr1, Attr2 = Expr2, ...
WHERE Condition;

```

**ou**

```

UPDATE Nom_Table
SET (Attr1, Attr2, ...) = (SELECT ...)
WHERE Condition;

```

La modification à effectuer est précisée après la clause SET. Il s'agit d'une affectation d'une valeur à un attribut grâce à l'opérateur = suivi d'une expression algébrique, d'une constante ou du résultat provenant d'une clause SELECT.

La clause WHERE permet de préciser les tuples sur lesquels la mise à jour aura lieu.

Borchani Anis 109

## Modification de données

### Application :

Modifier le Poids du produit numéro P002 à 1.

```
UPDATE Produit
SET Poids = 1
WHERE NP = 'P002';
```

Augmenter la quantité en stock des différents produits de 10%.

```
UPDATE Produit
SET Qtes = 1.1 * Qtes;
```

Borchani Anis 110

## Suppression de données

L'ordre DELETE permet de supprimer des lignes d'une table.

### Syntaxe :

```
DELETE FROM Nom_Table
WHERE Condition;
```

### Application :

Supprimer les clients de sousse.

```
DELETE FROM client
WHERE AdrCl ='sousse';
```

Borchani Anis 111

## Suppression de données

### Application :

Supprimer les produits blanc avec un poids <5.

```
DELETE FROM produit
WHERE coul='blanc' AND poids<5;
```

Supprimer tous les commandes des clients de sousse.

```
DELETE FROM commande
WHERE NCL in (select NCL from client where AdrCl='sousse');
```

Borchani Anis 112

## Langage d'interrogation de Données

La principale commande du langage d'interrogation de données est la commande SELECT.

La commande SELECT est basée sur l'algèbre relationnelle, en effectuant des opérations de sélection de données sur plusieurs tables relationnelles par projection. Sa syntaxe générale est la suivante :

Borchani Anis 113



6 – Langage SQL

## La commande Select

*Syntaxe :*

```

SELECT [ALL] | [DISTINCT] <liste des attributs> | *
FROM <liste des tables>
[WHERE <condition>]
[GROUP BY <liste des attributs> ]
[HAVING <condition de groupement >]
[ORDER BY <liste des attributs de tri> ] ;
    
```

[.....]: optionnel  
/: ou

Borchani Anis 114

6 – Langage SQL

## La commande Select : Options

- L'option **ALL** est, par opposition à l'option **DISTINCT**, l'option par défaut. Elle permet de sélectionner l'ensemble des lignes satisfaisant à la condition logique.
- L'option **DISTINCT** permet de ne conserver que des lignes distinctes, en éliminant les doublons.
- La liste des attributs indique la liste des attributs choisis, séparés par des virgules.
- L'option **\*** permet de sélectionner l'ensemble des colonnes d'une table.
- La liste des tables indique l'ensemble des tables (séparées par des virgules) sur lesquelles on opère.
- La **condition** permet d'exprimer des qualifications complexes à l'aide d'opérateurs logiques et de comparateurs arithmétiques.

Borchani Anis 115

6 – Langage SQL

## Projection

Une projection est une instruction permettant de sélectionner un ensemble d'attributs dans une table.

*Syntaxe :*

```

SELECT [ALL] | [DISTINCT] <liste des attributs> | *
FROM <la table> ;
    
```

*Application 1 :*

```

SELECT *
FROM Client ;
    
```

Client	NCI	NomCl	AdrCl
CL01	BATAM	Tunis	
CL02	BATIMENT	Tunis	
CL03	AMS	Sousse	
CL04	GLOULOU	Sousse	
CL05	PRODELEC	Tunis	
CL06	ELECTRON	Sousse	
CL07	SBATIM	Sousse	
CL08	SANITAIRE	Tunis	
CL09	SOUDURE	Tunis	
CL10	MELEC	Monastr	
CL11	MBATIM	Tunis	
CL12	BATFER	Tunis	

NCI	NomCl	AdrCl
CL01	BATAM	Tunis
CL02	BATIMENT	Tunis
CL03	AMS	Sousse
CL04	GLOULOU	Sousse
CL05	PRODELEC	Tunis
CL06	ELECTRON	Sousse
CL07	SBATIM	Sousse
CL08	SANITAIRE	Tunis
CL09	SOUDURE	Tunis
CL10	MELEC	Monastr
CL11	MBATIM	Tunis
CL12	BATFER	Tunis

Borchani Anis 116

6 – Langage SQL

## Projection

✓ Donner l'ensemble des produits qui ont été commandés (NP seulement).

*Application 2 :*

```

SELECT NP
FROM Ligne_Cmd;
    
```

```

SELECT DISTINCT NP
FROM Ligne_Cmd;
    
```

Ligne_Cmd		
NCmd	NP	Qte
C001	P001	250
C001	P004	300
C001	P006	100
C002	P002	200
C002	P007	550
C003	P001	50
C004	P002	100
C004	P004	150
C004	P005	70
C004	P008	90
C005	P001	650
C005	P002	100

NP
P001
P004
P006
P002
P007
P001
P002
P004
P005
P008
P001
P002

NP
P001
P004
P006
P002
P007
P005
P008

**DISTINCT** est utilisé pour éliminer les duplications, par défaut on obtient tous les tuples (**ALL**).

Borchani Anis

6 – Langage SQL

## Restrictions

Une restriction consiste à sélectionner les lignes satisfaisant à une condition logique effectuée sur leurs attributs. les restrictions s'expriment à l'aide de la clause WHERE suivie d'une condition logique exprimée à l'aide:

- Opérateurs logiques (AND, OR, NOT)
- Opérateurs arithmétiques (+, -, \*, /, %)
- Comparateurs arithmétiques (=, !=, >, <, >=, <=)
- Prédicats (NULL, IN, BETWEEN, LIKE, ALL, SOME, ANY, EXISTS).

Ces opérateurs s'appliquent aux valeurs numériques, aux chaînes de caractères et aux dates.

Syntaxe :

```
SELECT [ALL] | [DISTINCT] <liste des attributs> | *
FROM <la table>
WHERE <condition> ;
```

Borchani Anis 118

6 – Langage SQL

## La clause WHERE (1/2)

WHERE exp1 = exp2	Condition est vraie si les deux expressions exp1 et exp2 sont égales.
WHERE exp1 != exp2	Condition est vraie si les deux expressions exp1 et exp2 sont différentes.
WHERE exp1 < exp2	Condition est vraie si exp1 est inférieure à exp2.
WHERE exp1 > exp2	Condition est vraie si exp1 est supérieure à exp2.
WHERE exp1 <= exp2	Condition est vraie si exp1 est inférieure ou égale à exp2.
WHERE exp1 >= exp2	Condition est vraie si exp1 est supérieure ou égale à exp2.
WHERE exp1 BETWEEN exp2 AND exp3	Condition est vraie si exp1 est comprise entre exp2 et exp3, bornes incluses.
WHERE exp1 LIKE exp2	Condition est vraie si la sous-chaîne exp2 est présente dans exp1.
WHERE exp1 NOT LIKE exp2	Condition est vraie si la sous-chaîne exp2 n'est pas présente dans exp1.

Borchani Anis 119

6 – Langage SQL

## La clause WHERE (2/2)

WHERE exp1 IN (exp2, exp3,...)	Condition est vraie si exp1 appartient à l'ensemble (exp2, exp3, ...).
WHERE exp1 NOT IN (exp2, exp3,...)	Condition est vraie si exp1 n'appartient pas à l'ensemble (exp2, exp3, ...).
WHERE exp1 IS NULL	Condition est vraie si exp1 est <u>nulle</u> .
WHERE exp1 IS NOT NULL	Condition est vraie si exp1 n'est pas <u>nulle</u> .
WHERE exp1 Op ALL (exp2, exp3,...)	Op représente un opérateur de comparaison (<, >, =, !=, <=, >=) Condition est vraie si la comparaison de l'exp1 est vraie avec toutes les valeurs de la liste (exp2, exp3, ...). Si la liste est vide, le résultat est vrai.
WHERE exp1 Op ANY (exp2, exp3,...) WHERE exp1 Op SOME (exp2, exp3,...)	Op représente un opérateur de comparaison (<, >, =, !=, <=, >=) Condition est vraie si la comparaison de l'exp1 est vraie avec au moins une valeur de la liste (exp2, exp3, ...). Si la liste est vide, le résultat est faux.
WHERE EXISTS sous-requête	Condition est vraie si le résultat de la sous-requête n'est pas vide.

Borchani Anis 120

6 – Langage SQL

## Restrictions : Applications

Application 1 :

✓ Donner le numéro et le nom des clients de la ville de Sousse.

```
SELECT NCl, NomCl
FROM Client
WHERE AdrCl = 'Sousse';
```

Client		
NCl	NomCl	AdrCl
CL01	BATAM	Tunis
CL02	BATIMENT	Tunis
CL03	AMS	Sousse
CL04	GLOULOU	Sousse
CL05	PRODELEC	Tunis
CL06	ELECTRON	Sousse
CL07	SBATIM	Sousse
CL08	SANITAIRE	Tunis
CL09	SOUURE	Tunis
CL10	MELEC	Monastr
CL11	MBATIM	
CL12	BATFER	Tunis

NCl	NomCl
CL03	AMS
CL04	GLOULOU
CL06	ELECTRON
CL07	SBATIM

Borchani Anis 121

6 – Langage SQL

### Restrictions : Applications

Application 2:

✓ Donner la liste des commandes dont la date est supérieure à '01/01/2004'.

```
SELECT *
FROM Commandes
WHERE DateCmd > '01/01/2004';
```

Commande		
NCmd	DateCmd	NCI
C001	10/12/2003	CL02
C002	13/02/2004	CL05
C003	15/01/2004	CL03
C004	03/09/2003	CL10
C005	11/03/2004	CL03

NCmd	DateCmd	NCI
C002	13/02/2004	CL05
C003	15/01/2004	CL03
C005	11/03/2004	CL03

Borchani Anis 122

6 – Langage SQL

### Restrictions : Applications

Application 3:

✓ Donner la liste des produits dont le prix est compris entre 20 et 50.

```
SELECT *
FROM Produit
WHERE PU BETWEEN 20 AND 50;
```

Ou bien

```
SELECT *
FROM Produit
WHERE (PU >= 20) AND (PU <= 50);
```

Produit					
NP	LibP	Coul	Poids	PU	Qtes
P001	Robinet	Gris	5	18.000	1200
P002	Prise	Blanc	1.2	1.500	1000
P003	Câble	Blanc	2	25.000	1500
P004	Peinture	Blanc	25	33.000	900
P005	Poignée	Gris	3	12.000	1300
P006	Serrure	Jaune	2	47.000	1250
P007	Verrou	Gris	1.7	5.500	2000
P008	Fer	Noir	50	90.000	800

NP	LibP	Coul	Poids	PU	Qtes
P003	Câble	Blanc	2	25.000	1500
P004	Peinture	Blanc	25	33.000	900
P006	Serrure	Jaune	2	47.000	1250

Borchani Anis 123

6 – Langage SQL

### Restrictions : Applications

Application 4:

✓ Donner la liste des clients dont les noms commencent par 'B'.

```
SELECT *
FROM Client
WHERE NomCl LIKE 'B%';
```

Client		
NCI	NomCl	AdrCl
CL01	BATAM	Tunis
CL02	BATIMENT	Tunis
CL03	AMS	Sousse
CL04	GLOULOU	Sousse
CL05	PRODELEC	Tunis
CL06	ELECTRON	Sousse
CL07	SBATIM	Sousse
CL08	SANITAIRE	Tunis
CL09	SOUUDURE	Tunis
CL10	MELEC	Monastir
CL11	MBATIM	Tunis
CL12	BATFER	Tunis

NCI	NomCl	AdrCl
CL01	BATAM	Tunis
CL02	BATIMENT	Tunis
CL12	BATFER	Tunis

Borchani Anis 124

6 – Langage SQL

### Le prédicat Like

- Le prédicat **LIKE** permet de faire des comparaisons sur des chaînes grâce à des caractères, appelés caractères jokers :
- Le caractère **%** permet de remplacer une séquence de caractères (éventuellement nulle).
- Le caractère **\_** permet de remplacer un caractère.
- Les caractères **[-]** permettent de définir un intervalle de caractères (par exemple [J-M]).
- La sélection des clients dont les noms ont un E en deuxième position se fait par l'instruction : *WHERE NomCl LIKE "\_E%"*

Borchani Anis 125

6 – Langage SQL

### Restrictions : Applications

*Application 5:*

✓ Donner les numéros des clients dont les dates de leurs commandes se trouvent parmi les dates suivantes : ('10-12-03', '10-12-04', '13-02-04', '11-03-04').

```
SELECT DateCmd, NCl
FROM Commande
WHERE DateCmd IN ('10-12-2003', '10-12-2004', '13-02-2004', '11-03-2004');
```

Commande		
NCmd	DateCmd	NCl
C001	10/12/2003	CL02
C002	13/02/2004	CL05
C003	15/01/2004	CL03
C004	03/09/2003	CL10
C005	11/03/2004	CL03

DateCmd	NCl
10/12/2003	CL02
13/02/2004	CL05
11/03/2004	CL03

Borchani Anis 126

6 – Langage SQL

### Restrictions : Applications

*Application 6:*

✓ Donner les noms des clients qui n'ont pas d'adresse.

```
SELECT NomCl
FROM Client
WHERE AdrCl IS NULL;
```

Client		
NCI	NomCl	AdrCl
CL01	BATAM	Tunis
CL02	BATIMENT	Tunis
CL03	AMS	Sousse
CL04	GLOULOU	Sousse
CL05	PRODELEC	Tunis
CL06	ELECTRON	Sousse
CL07	SBATIM	Sousse
CL08	SANTAIRE	Tunis
CL09	SOUDURE	Tunis
CL10	MELEC	Monastir
CL11	MBATIM	
CL12	BATFER	Tunis

NomCl
MBATIM

*Lorsqu'un champ n'est pas renseigné, le SGBD lui attribue une valeur spéciale que l'on note NULL. La recherche de cette valeur ne peut pas se faire à l'aide des opérateurs standards, il faut utiliser les prédicats IS NULL ou bien IS NOT NULL.*

Borchani Anis 127

6 – Langage SQL

### Les Alias

Les alias permettent de renommer des attributs ou des tables.

```
SELECT attr1 AS aliasa1, attr2 AS aliasa2, ...
FROM table1 alias1, table2 alias2...;
```

Pour les attributs, l'alias correspond aux titres des colonnes affichées dans le résultat de la requête. Il est souvent utilisé lorsqu'il s'agit d'attributs calculés (expression).

NB : Le mot clé AS est optionnel.

Borchani Anis 128

6 – Langage SQL

### Les Alias

*Application :*

✓ Donner la valeur des produits en stock.

```
SELECT NP, (Qtes * PU) AS "Valeur Totale"
FROM Produit ;
```

Produit					
NP	LibP	Coul	Poids	PU	Qtes
P001	Robinet	Gris	5	18.000	1200
P002	Prise	Blanc	1,2	1.500	1000
P003	Câble	Blanc	2	25.000	1500
P004	Peinture	Blanc	25	33.000	900
P005	Poignée	Gris	3	12.000	1300
P006	Serrure	Jaune	2	47.000	1250
P007	Verrou	Gris	1,7	5.500	2000
P008	Fer	Noir	50	90.000	800

NP	Valeur Totale
P001	21600
P002	1500
P003	37500
P004	29700
P005	15600
P006	58750
P007	11000
P008	72000

On peut exprimer cette requête sans spécifier le mot clé AS :

```
SELECT NP, (Qtes * PU) "Valeur Totale"
FROM Produit ;
```

Borchani Anis 129

6 – Langage SQL

## Les fonctions

Le tableau suivant donne le nom des principales fonctions prédéfinies :

Nom de la fonction	Rôle de la fonction
AVG	Moyenne
SUM	Somme
MIN	Minimum
MAX	Maximum
COUNT (*)	Nombre de lignes
COUNT (Attr)	Nombre de valeurs non nulles de l'attribut
COUNT(DISTINCT) Attr)	Nombre de valeurs non nulles différentes de l'attribut

*Application :*

✓ Donner la moyenne des prix unitaires des produits.

```
SELECT AVG(PU)
FROM Produit;
```

Produit						Avg(PU)
NP	LibP	Coul	Poids	PU	Qtes	
P001	Robinet	Gris	5	18.000	1200	29
P002	Prise	Blanc	1.2	1.500	1000	
P003	Câble	Blanc	2	25.000	1500	
P004	Peinture	Blanc	25	33.000	900	
P005	Poignée	Gris	3	12.000	1300	
P006	Serrure	Jaune	2	47.000	1250	
P007	Verrou	Gris	1.7	5.500	2000	
P008	Fer	Noir	50	90.000	800	

Borchani Anis 130

6 – Langage SQL

## Sélection avec jointure

Il s'agit ici de sélectionner les données provenant de plusieurs tables ayant un ou plusieurs attributs communs. Cette jointure sera assurée grâce aux conditions spécifiées dans la clause WHERE.

*Syntaxe :*

```
SELECT [ALL] | [DISTINCT] <liste des attributs> | *
FROM <liste des tables>
WHERE Nom_Table1.Attrj = Nom_Table2.Attrj AND ...
AND <condition>;
```

Borchani Anis 131

6 – Langage SQL

## Sélection avec jointure

*Application 1:*

✓ Donner les libellés des produits de la commande numéro 'C002'.

```
SELECT DISTINCT LibP
FROM Produit, Ligne_Cmd
WHERE Produit.NP = Ligne_Cmd.NP
AND NCmd='C002';
```

*Ou bien*

```
SELECT DISTINCT LibP
FROM Produit P, Ligne_Cmd L
WHERE P.NP = L.NP
AND NCmd='C002';
```

Produit						Ligne_Cmd		
NP	LibP	Coul	Poids	PU	Qtes	NCmd	NP	Qte
P001	Robinet	Gris	5	18.000	1200	C001	P001	250
P002	Prise	Blanc	1.2	1.500	1000	C001	P004	300
P003	Câble	Blanc	2	25.000	1500	C001	P006	100
P004	Peinture	Blanc	25	33.000	900	C002	P002	200
P005	Poignée	Gris	3	12.000	1300	C002	P007	550
P006	Serrure	Jaune	2	47.000	1250	C003	P001	50
P007	Verrou	Gris	1.7	5.500	2000	C004	P002	100
P008	Fer	Noir	50	90.000	800	C004	P002	100

On peut exprimer cette jointure autrement ?

LibP
Prise
Verrou

Borchani Anis 132

6 – Langage SQL

## Sélection avec jointure

*Application 2:*

✓ Donner les produits (toutes les informations) commandés au cours de l'année 2003 et qui sont vendus aux clients de Tunis.

```
SELECT DISTINCT P.NP, LibP, Coul, Poids, PU, Qtes
FROM Produit P, Commande C, Client Cl, Ligne_Cmd L
WHERE P.NP = L.NP
AND C.NCmd = L.NCmd
AND Cl.NCl = C.NCl
AND Cl.AdrCl = 'Tunis'
AND DateCmd BETWEEN '01/01/2003' AND '31/12/2003';
```

NP	LibP	Coul	Poids	PU	Qtes
P001	Robinet	Gris	5	18	1200
P004	Peinture	Blanc	25	33	900
P006	Serrure	Jaune	2	47	1250

Borchani Anis 133

6 – Langage SQL

## Groupement

Il est possible de grouper des lignes de données ayant une valeur commune à l'aide de la clause GROUP BY et des fonctions de groupe.

*Syntaxe :*

```

SELECT Attr1, Attr2,..., Fonction_Groupe
FROM Nom_Table1, Nom_Table2,...
WHERE Liste_Condition
GROUP BY Liste_Groupe
HAVING Condition;
    
```

- La clause **GROUP BY**, suivie du nom de chaque attribut sur laquelle on veut effectuer des regroupements.
- La clause **HAVING** va de pair avec la clause GROUP BY, elle permet d'appliquer une restriction sur les groupes créés grâce à la clause GROUP BY.

Borchani Anis 134

6 – Langage SQL

## Groupement

*Application 1 :*

✓ Donner le nombre de commandes par client.

```

SELECT NCl, COUNT((NCmd) NbCmd
FROM Commande
GROUP BY NCl;
    
```

Commande			NCl	
NCmd	DateCmd	NCl	NCl	NbCmd
C001	10/12/2003	CL02	CL02	1
C002	13/02/2004	CL05	CL05	1
C003	15/01/2004	CL03	CL03	2
C004	03/09/2003	CL10	CL10	1
C005	11/03/2004	CL03		

**NB :** Les fonctions d'agrégat, utilisées seules dans un SELECT (sans la clause GROUP BY) fonctionnent sur la totalité des tuples sélectionnés comme s'il n'y avait qu'un groupe.

Borchani Anis 135

6 – Langage SQL

## Groupement

*Application 2 :*

✓ Donner la quantité totale commandée par produit.

```

SELECT NP, SUM(Qte) Som
FROM Ligne_Cmd
GROUP BY NP;
    
```

Ligne_Cmd			NP	
NCmd	NP	Qte	NP	Som
C001	P001	250	P001	950
C001	P004	300	P002	400
C001	P006	100	P004	450
C002	P002	200	P005	70
C002	P007	550	P006	100
C003	P001	50	P007	550
C004	P002	100	P008	90
C004	P004	150		
C004	P005	70		
C004	P008	90		
C005	P001	650		
C005	P002	100		

Borchani Anis 136

6 – Langage SQL

## Groupement

*Application 3 :*

✓ Donner le nombre de produits par commande.

```

SELECT NCmd, COUNT(NP) NbProd
FROM Ligne_Cmd
GROUP BY NCmd;
    
```

Ligne_Cmd			NCmd	
NCmd	NP	Qte	NCmd	NbProd
C001	P001	250	C001	3
C001	P004	300	C002	2
C001	P006	100	C003	1
C002	P002	200	C004	4
C002	P007	550	C005	2
C003	P001	50		
C004	P002	100		
C004	P004	150		
C004	P005	70		
C004	P008	90		
C005	P001	650		
C005	P002	100		

Borchani Anis 137

6 – Langage SQL

## Groupement

*Application 4 :*

✓ Donner les commandes dont le nombre de produits dépasse 2.

```

SELECT NCmd, COUNT(NP) NbProd
FROM Ligne_Cmd
GROUP BY NCmd
HAVING COUNT(NP) > 2;
    
```

NCmd	NP	Qte
C001	P001	250
C001	P004	300
C001	P006	100
C002	P002	200
C002	P007	550
C003	P001	50
C004	P002	100
C004	P004	150
C004	P005	70
C004	P008	90
C005	P001	650
C005	P002	100

NCmd	NbProd
C001	3
C004	4

Borchani Anis 138

6 – Langage SQL

## Groupement

*Application 5 :*

NP	LibP	Coul	Poids	PU	Qtes
P001	Robinet	Gris	5	18.000	1200
P002	Prise	Blanc	1,2	1.500	1000
P003	Câble	Blanc	2	25.000	1500
P004	Peinture	Blanc	25	33.000	900
P005	Poignée	Gris	3	12.000	1300
P006	Serrure	Jaune	2	47.000	1250
P007	Verrou	Gris	1,7	5.500	2000
P008	Fer	Noir	50	90.000	800

NCmd	NP	Qte
C001	P001	250
C001	P004	300
C001	P006	100
C002	P002	200
C002	P007	550
C003	P001	50
C004	P002	100
C004	P004	150
C004	P005	70
C004	P008	90
C005	P001	650
C005	P002	100

✓ Donner le total des montants par commande.

```

SELECT NCmd, SUM(PU*Qte) TotMontant
FROM Produit P, Ligne_Cmd L
WHERE L.NP = P.NP
GROUP BY NCmd;
    
```

NCmd	TotMontant
C001	19100
C002	3325
C003	900
C004	14040
C005	11850

Borchani Anis 139

6 – Langage SQL

## Tri

Les lignes constituant le résultat d'un SELECT sont obtenues dans un ordre quelconque. La clause **ORDER BY** précise l'ordre dans lequel la liste des lignes sélectionnées sera donnée.

*Syntaxe :*

```

SELECT Attr1, Attr2,..., Attrn
FROM Nom_Table1, Nom_Table2, ...
WHERE Liste_Condition
ORDER BY Attr1 [ASC | DESC], Attr2 [ASC | DESC], ...;
    
```

➤ **NB** : L'ordre de tri par défaut est croissant (ASC).

Borchani Anis 140

6 – Langage SQL

## Tri

*Application 1 :*

✓ Donner les noms des clients suivant l'ordre décroissant.

```

SELECT NomCl
FROM Client
ORDER BY NomCl DESC ;
    
```

NCl	NomCl	AdrCl
CL01	BATAM	Tunis
CL02	BATIMENT	Tunis
CL03	AMS	Sousse
CL04	GLOULOU	Sousse
CL05	PRODELEC	Tunis
CL06	ELECTRON	Sousse
CL07	SBATIM	Sousse
CL08	SANITAIRE	Tunis
CL09	SOUURE	Tunis
CL10	MELEC	Monastir
CL11	MBATIM	
CL12	BATFER	Tunis

NomCl
SOUURE
SBATIM
SANITAIRE
PRODELEC
MELEC
MBATIM
GLOULOU
ELECTRON
BATIMENT
BATFER
BATAM
AMS

Borchani Anis 141

6 – Langage SQL

### Tri

*Application 2 :*

✓ Donner le nombre de produits et la quantité totale par commande suivant l'ordre décroissant du nombre de produits et l'ordre croissant de la quantité totale.

```

SELECT NCmd,
COUNT DISTINCT(NP) NbProd,
SUM(Qte) SomQte
FROM Ligne_Cmd
GROUP BY NCmd
ORDER BY NbProd DESC,
        SomQte ASC ;
    
```

Ligne_Cmd		
NCmd	NP	Qte
C001	P001	250
C001	P004	300
C001	P006	100
C002	P002	200
C002	P007	550
C003	P001	50
C004	P002	100
C004	P004	150
C004	P005	70
C004	P008	90
C005	P001	650
C005	P002	100

NCmd	NbProd	SomQte
C004	4	410
C001	3	650
C005	2	750
C002	2	750
C003	1	50

Borchani Anis 142

6 – Langage SQL

### Sous requêtes

Consiste à effectuer une requête à l'intérieur d'une autre, ou en d'autres termes d'utiliser une requête afin d'en réaliser une autre (requêtes en cascade).

Une sous-requête doit être placée à la suite d'une clause WHERE ou HAVING, et doit remplacer une constante ou un groupe de constantes qui permettraient d'exprimer la qualification.

lorsque la sous-requête remplace une constante utilisée avec des opérateurs classiques, elle doit obligatoirement renvoyer une seule réponse (une table d'une ligne et une colonne). Par exemple :

```

SELECT ... FROM ...
WHERE ... < (SELECT ... FROM ...) ;
    
```

Borchani Anis 143

6 – Langage SQL

### Sous requêtes

lorsque la sous-requête remplace une constante utilisée dans une expression mettant en jeu les opérateurs IN, ALL ou ANY, elle doit obligatoirement renvoyer une seule colonne.

```

SELECT ... FROM ...
WHERE ... IN (SELECT ... FROM ...) ;
    
```

lorsque la sous-requête remplace une constante utilisée dans une expression mettant en jeu l'opérateur EXISTS, elle peut renvoyer une table de n colonnes et m lignes.

```

SELECT ... FROM ...
WHERE ... EXISTS (SELECT ... FROM ...) ;
    
```

Borchani Anis 144

6 – Langage SQL

### Sous requêtes

*Application 1 :*

✓ Donner les produits dont les prix unitaires dépassent la moyenne des prix.

```

SELECT *
FROM Produit
WHERE PU > (SELECT AVG(PU) FROM Produit);
    
```

Le résultat de la sous-requête :

AVG(PU)
29

Produit						
NP	LibP	Coul	Poids	PU	Qtes	
P001	Robinet	Gris	5	18.000	1200	
P002	Prise	Blanc	1.2	1.500	1000	
P003	Câble	Blanc	2	25.000	1500	
P004	Peinture	Blanc	25	33.000	900	→
P005	Poignée	Gris	3	12.000	1300	
P006	Serrure	Jaune	2	47.000	1250	→
P007	Verrou	Gris	1.7	5.500	2000	
P008	Fer	Noir	50	90.000	800	→

NP	LibP	Coul	Poids	PU	Qtes
P004	Peinture	Blanc	25	33.000	900
P006	Serrure	Jaune	2	47.000	1250
P008	Fer	Noir	50	90.000	800

Borchani Anis 145



6 – Langage SQL

### Sous requêtes

**Application 2 :**

✓ Donner les commandes qui ont une date inférieure à chacune des commandes du client 'CL03'.

```

SELECT *
FROM Commande
WHERE DateCmd < ALL
  (SELECT DateCmd
   FROM Commande
   WHERE NCl = 'CL03');
    
```

Commande		
NCmd	DateCmd	NCl
C001	10/12/2003	CL02
C002	13/02/2004	CL05
C003	15/01/2004	CL03
C004	03/09/2003	CL10
C005	11/03/2004	CL03

Le résultat de la sous-requête :
 

DateCmd
15/01/2004
11/03/2004

NCmd	DateCmd	NCl
C001	10/12/2003	CL02
C004	03/09/2003	CL10

Borchani Anis 146

6 – Langage SQL

### Sous requêtes

**Application 3 :**

✓ Donner les commandes qui ont une date inférieure à au moins une des commandes du client 'CL03'.

```

SELECT *
FROM Commande
WHERE DateCmd < ANY
  (SELECT DateCmd
   FROM Commande
   WHERE NCl = 'CL03');
    
```

Commande		
NCmd	DateCmd	NCl
C001	10/12/2003	CL02
C002	13/02/2004	CL05
C003	15/01/2004	CL03
C004	03/09/2003	CL10
C005	11/03/2004	CL03

Le résultat de la sous-requête :
 

DateCmd
15/01/2004
11/03/2004

NCmd	DateCmd	NCl
C001	10/12/2003	CL02
C002	13/02/2004	CL05
C003	15/01/2004	CL03
C004	03/09/2003	CL10

Borchani Anis 147

6 – Langage SQL

### Sous requêtes

**Application 4 :**

✓ Donner les clients qui ont passé au moins une commande.

```

SELECT NomCl
FROM Client Cl
WHERE EXIST (SELECT * FROM Commande C WHERE Cl.NCl=C.NCl);
    
```

Client		
NCl	NomCl	AdrCl
CL01	BATAM	Tunis
CL02	BATIMENT	Tunis
CL03	AMS	Sousse
CL04	GLOULOU	Sousse
CL05	PRODELEC	Tunis
CL06	ELECTRON	Sousse
CL07	SBATIM	Sousse
CL08	SANITAIRE	Tunis
CL09	SOUDURE	Tunis
CL10	MELEC	Monastir
CL11	MBATIM	
CL12	BATFER	Tunis

Commande		
NCmd	DateCmd	NCl
C001	10/12/2003	CL02
C002	13/02/2004	CL05
C003	15/01/2004	CL03
C004	03/09/2003	CL10
C005	11/03/2004	CL03

NomCl
BATIMENT
AMS
PRODELEC
MELEC

Borchani Anis 148

6 – Langage SQL

### Sous requêtes

**Application 5 :**

✓ Donner des clients qui n'ont passé aucune commande.

```

SELECT NomCl
FROM Client Cl
WHERE NOT EXIST (SELECT * FROM Commande C WHERE Cl.NCl=C.NCl);
    
```

Client		
NCl	NomCl	AdrCl
CL01	BATAM	Tunis
CL02	BATIMENT	Tunis
CL03	AMS	Sousse
CL04	GLOULOU	Sousse
CL05	PRODELEC	Tunis
CL06	ELECTRON	Sousse
CL07	SBATIM	Sousse
CL08	SANITAIRE	Tunis
CL09	SOUDURE	Tunis
CL10	MELEC	Monastir
CL11	MBATIM	
CL12	BATFER	Tunis

Commande		
NCmd	DateCmd	NCl
C001	10/12/2003	CL02
C002	13/02/2004	CL05
C003	15/01/2004	CL03
C004	03/09/2003	CL10
C005	11/03/2004	CL03

NomCl
BATAM
GLOULOU
ELECTRON
SBATIM
SANITAIRE
SOUDURE
MBATIM
BATFER

Borchani Anis 149

6 – Langage SQL

## Opérateurs ensemblistes : UNION

L'opérateur **UNION** permet de fusionner deux sélections de tables pour obtenir un ensemble de lignes égal à la réunion des lignes des deux sélections. Les lignes communes n'apparaîtront qu'une seule fois.

*Syntaxe :*

```
Requête1
UNION
Requête2 ;
```

- Requête1 et Requête2 doivent avoir la même structure.
- Par défaut les doublons sont automatiquement éliminés.
- Pour conserver les doublons, il est possible d'utiliser une clause UNION ALL.

Borchani Anis 150

6 – Langage SQL

## Opérateurs ensemblistes : UNION

*Application :*

✓ Donner l'ensemble des clients de Tunis et de Sousse qui ont des commandes.

```
SELECT Cl.NCI, NomCl, AdrCl
FROM Client Cl, Commande C
WHERE Cl.NCI=C.NCI
AND J.AdrCl='Tunis'
UNION
SELECT Cl.NCI, NomCl, AdrCl
FROM Client Cl, Commande C
WHERE Cl.NCI=C.NCI
AND J.AdrCl='Sousse';
```

Client			Commande		
NCI	NomCl	AdrCl	NCmd	DateCmd	NCI
CL01	BATAM	Tunis	C001	10/12/2003	CL02
CL02	BATMENT	Tunis	C002	13/02/2004	CL05
CL03	AMS	Sousse	C003	15/01/2004	CL03
CL04	GLOULOLOU	Sousse	C004	03/09/2003	CL10
CL05	PRODELEC	Tunis	C005	11/03/2004	CL03
CL06	ELECTRON	Sousse			
CL07	SBATIM	Sousse			
CL08	SANTAIRE	Tunis			
CL09	SOUDURE	Tunis			
CL10	MELEC	Monastir			
CL11	MBATIM				
CL12	BATFER	Tunis			

R1	NCI	NomCl	AdrCl	R2	NCI	NomCl	AdrCl
	CL02	BATMENT	Tunis		CL03	AMS	Sousse
	CL05	PRODELEC	Tunis				

NCI	NomCl	AdrCl
CL02	BATMENT	Tunis
CL05	PRODELEC	Tunis
CL03	AMS	Sousse

Borchani Anis 151

6 – Langage SQL

## Opérateurs ensemblistes : INTERSECTION

L'opérateur **INTERSECT** permet d'obtenir l'ensemble des lignes communes à deux requêtes.

*Syntaxe :*

```
Requête1
INTERSECT
Requête2 ;
```

- Requête1 et Requête2 doivent avoir la même structure.
- Il est possible de remplacer l'opérateur INTERSECT par des commandes usuelles :  
 SELECT a, b  
 FROM table1  
 WHERE EXISTS ( SELECT c, d FROM table2 WHERE a=c AND b=d );

Borchani Anis 152

6 – Langage SQL

## Opérateurs ensemblistes : INTERSECTION

*Application :*

✓ Donner l'ensemble des produits communs aux commandes C001 et C005.

```
SELECT P.NP, LibP
FROM Produit P, Ligne_Cmd L
WHERE P.NP=L.NP
AND NCmd='C001'
INTERSECT
SELECT P.NP, LibP
FROM Produit P, Ligne_Cmd L
WHERE P.NP=L.NP
AND NCmd='C005';
```

Produit						Ligne_Cmd		
NP	LibP	Coul	Poids	PU	Qtes	NCmd	NP	Qte
P001	Robinet	Gris	5	18.000	1200	C001	P001	250
P002	Prise	Blanc	1.2	1.500	1000	C001	P004	300
P003	Câble	Blanc	2	25.000	1500	C001	P006	100
P004	Peinture	Blanc	25	33.000	900	C002	P002	200
P005	Pozmée	Gris	3	12.000	1300	C002	P007	550
P006	Serrure	Jaune	2	47.000	1250	C003	P001	50
P007	Verrou	Gris	1.7	5.500	2000	C004	P002	100
P008	Fer	Noir	50	90.000	800	C004	P004	150
						C004	P005	70
						C004	P008	90
						C005	P001	650
						C005	P002	100

R1	NP	LibP	R2	NP	LibP
	P001	Robinet		P001	Robinet
	P004	Peinture		P002	Prise
	P006	Serrure			

NP	LibP
P001	Robinet

Borchani Anis 153

6 – Langage SQL

## Opérateurs ensemblistes : Différence

L'opérateur **MINUS** permet d'obtenir les lignes de la première requête et qui ne figurent pas dans la deuxième.

*Syntaxe :*

Requête1  
**MINUS (ou EXCEPT)**  
 Requête2 ;

➤ Requête1 et Requête2 doivent avoir la même structure.

Borchani Anis 154

6 – Langage SQL

## Opérateurs ensemblistes : DIFFERENCE

*Application :*

✓ Donner l'ensemble des produits qui n'ont pas été commandés.

```

SELECT NP, LibP
FROM Produit
MINUS
SELECT P.NP, LibP
FROM Produit P, Ligne_Cmd L
WHERE P.NP=L.NP;
                
```

Produit						Ligne_Cmd		
NP	LibP	Coul	Poids	PU	Qtes	NCmd	NP	Qte
P001	Robnet	Gris	5	18.000	1200	C001	P001	250
P002	Prise	Blanc	1,2	1.500	1000	C001	P004	300
P003	Câble	Blanc	2	25.000	1500	C001	P006	100
P004	Peinture	Blanc	25	33.000	900	C002	P002	200
P005	Poignée	Gris	3	12.000	1300	C002	P007	550
P006	Serrure	Jaune	2	47.000	1250	C003	P001	50
P007	Verrou	Gris	1,7	5.500	2000	C004	P002	100
P008	Fer	Noir	50	90.000	800	C004	P004	150
						C004	P005	70
						C004	P008	90
						C005	P001	650
						C005	P002	100

Tous les produits (R1) → Les produits commandés (R2) → Résultat (R3)

NP	LibP
P001	Robnet
P002	Prise
P003	Câble
P004	Peinture
P005	Poignée
P006	Serrure
P007	Verrou
P008	Fer

NP	LibP
P001	Robnet
P002	Prise
P004	Peinture
P005	Poignée
P006	Serrure
P007	Verrou
P008	Fer

NP	LibP
P003	Câble

Borchani Anis 155

6 – Langage SQL

## Langage de Contrôle de Données

Plusieurs personnes peuvent travailler simultanément sur une base de données, toutefois ces personnes n'ont pas forcément les mêmes besoins : certaines peuvent par exemple nécessiter de modifier des données dans la table, tandis que les autres ne l'utiliseront que pour la consulter. Ainsi, il est possible de définir des permissions pour chaque personne en leur octroyant un mot de passe.

Cette tâche incombe à l'administrateur de la base de données (en anglais DBA, DataBase Administrator). Il doit dans un premier temps définir les besoins de chacun, puis les appliquer à la base de donnée sous forme de permissions.

Borchani Anis 156

6 – Langage SQL

## Langage de Contrôle de Données

Le langage SQL permet d'effectuer ces opérations grâce à deux clauses :

**GRANT** permet d'accorder des droits à un (parfois plusieurs sur certains SGBD) utilisateur .

**REVOKE** permet de retirer des droits à un (ou plusieurs sur certains SGBD) utilisateur Les permissions (appelées aussi droits ou privilèges) peuvent être définies pour chaque (un grand nombre) clause.

D'autre part il est aussi possible de définir des rôles c'est-à-dire de permettre à d'autres utilisateurs d'accorder des permissions.

Borchani Anis 157

6 – Langage SQL

## Langage de Contrôle de Données

**GRANT** : Permet au propriétaire d'une table ou vue de donner à d'autres utilisateurs des droits d'accès à celles ci.

*Syntaxe :*  
**GRANT** Liste\_Privilège **ON** Table/ Vue **TO** Utilisateur [*WITH GRANT OPTION*];

Les privilèges sont :

SELECT	Droit de lecture
INSERT	Droit d'insertion de lignes
UPDATE	Droit de modification de lignes
UPDATE (Attr1, Attr2, ...)	Droit de modification de lignes à certains attributs
DELETE	Droit de suppression de lignes
ALTER	Droit de modification de la structure de la table
INDEX	Droit de création d'index
ALL	Tous les droits

*Application :*  
 GRANT SELECT ON Produit TO User1;

Borchani Anis 158

6 – Langage SQL

## Langage de Contrôle de Données

**REVOKE** : Un utilisateur ayant accordé un privilège peut l'annuler à l'aide de la commande REVOKE.

*Syntaxe :*  
**REVOKE** Liste\_Privilège **ON** Table/Vue **FROM** Utilisateur;

*Application :*  
 REVOKE SELECT ON Produit FROM User1;

**NB** : Si on enlève un privilège à un utilisateur, ce même privilège est automatiquement retiré à tout autre utilisateur à qui il l'aurait accordé.

Borchani Anis 159