

M E M E N T O S Q L

I- TYPES DE DONNEES

TYPE DE DONNEE	DEFINITION
VARCHAR2(N)	Comme Char, mais on ne stockera que le nombre de caractères nécessaires.
INTEGER	Valeurs entières. Les limites varient suivant le SGBD
SMALLINT	Comme integer, mais avec une plage de valeurs limitée par rapport à integer
NUMBER(w, d)	valeurs numériques de w chiffres dont d décimales
DECIMAL(w, d)	Valeurs numériques de w chiffres dont d décimales
DATE	Valeurs de type date
TIME	Valeur de type heure

II- LES TABLES

1- CREATION

```
CREATE TABLE nom-table  
(nom-colonne1 TYPEDONNEE [NOT NULL] [DEFAULT valeur_par_défaut]  
 [nom-colonne2 TYPEDONNEE [NOT NULL] [DEFAULT valeur_par_défaut]...]  
 [contrainte_table]  
);
```

2- MODIFICATION

```
ALTER TABLE nom-table  
ADD (nom-col1 TYPEDONNEE [NOT NULL] [DEFAULT valeur-par-défaut]  
 [nom-col2 ..... ]);  
  
ALTER TABLE nom-table  
MODIFY [COLUMN] (nom-col1 TYPEDONNEE [NOT NULL] [DEFAULT valeurpardéfaut]  
 [nom-col2 ..... ]);
```

3- SUPPRESSION

```
DROP TABLE nom-table
```

4- CREATION D'INDEX

```
CREATE [UNIQUE] INDEX nom-index ON nom-table  
(nom-col1 [ASC|DESC],[nom-col2...]);
```

5- SUPPRESSION D'INDEX

```
DROP INDEX nom-index ;
```

III- LES CONTRAINTES

1- LES CONTRAINTES COLONNE

```
{  
 [CONSTRAINT nom-contrainte]  
 {  
 | NOT NULL  
 | DEFAULT valeur  
 | UNIQUE | PRIMARY KEY }  
 | REFERENCES nom-table(nom-colonne)  
 }  
 | CHECK (condition)  
 }  
 [ON DELETE action_reference]  
 [ON UPDATE action_reference]
```

2- LES CONTRAINTES TABLE

```
{  
 [CONSTRAINT nom-contrainte]  
 {  
 | NOT NULL  
 | DEFAULT valeur  
 | UNIQUE | PRIMARY KEY (nom-colonne1 [,nom-colonne2]...)  
 | FOREIGN KEY (nom-colonne3 [,nom-colonne4]...)  
 | REFERENCES nom-table(nom-colonne5 [,nom-colonne6]...)  
 }  
 | CHECK (condition)  
 }  
 [ON DELETE action_reference]  
 [ON UPDATE action_reference]
```

3- AJOUT D'UNE CONTRAINTE A UNE TABLE

```
{  
 ALTER TABLE nom-table  
 ADD CONSTRAINT [nom-contrainte]  
 {  
 | NOT NULL  
 | DEFAULT valeur  
 | UNIQUE | PRIMARY KEY (nom-colonne1 [,nom-colonne2]...)  
 | FOREIGN KEY (nom-colonne3 [,nom-colonne4]...)  
 | REFERENCES nom-table(nom-colonne5 [,nom-colonne6]...)  
 }  
 | CHECK (condition)  
 };
```

IV- LES VUES

1- CREATION

```
CREATE VIEW nom-vue  
AS SELECT .....  
 [with check option | with read only] ;
```

2- SUPPRESSION

```
DROP VIEW nom-vue ;
```


V. LA CLAUSE SELECT

```
SELECT [DISTINCT | ALL] [nom-tab], nom-col1 [, [nom-tab], nom-col2] ...  
FROM nom-tab1 [, nom-tab2]  
[WHERE prédicat]  
[GROUP BY nom-col3 [, nom-col4...]] [HAVING prédicat]  
[ORDER BY nom-col5 [DESC] [, nom-col6 [DESC]...]] ;
```

Opérateurs de comparaison

```
égal (=)  
inférieur (<)  
supérieur (>)  
inférieur ou égal (<=)  
supérieur ou égal (>=)  
différent (<>)
```

BETWEEN valeur1 AND valeur2
IN (valeur1, valeur2...)
LIKE filtre

Filtre : désigne une chaîne de caractères comportant les symboles "%" et/ou "_".

Opérateurs logiques

AND OR NOT

Autres opérateurs

ALL ANY

- 1- LES QUALIFICATEURS COURTS OU ALIAS
FROM nom-table [AS] alias-table ...

- 2- EXPRESSION DE LA JOINTURE AVEC L'OPERATEUR JOIN :

- Jointure interne: SELECT ... FROM table1 [INNER] JOIN table2
ON nom-col1 = nom-col2 ...
- Jointure externe: SELECT ... FROM table-gauche LEFT | RIGHT [OUTER] JOIN table-droite
ON nom-col1 = nom-col2 ...

- 3- LES FONCTIONS SUR AGREGAT

SUM(nom, colonne) : somme arithmétique
AVG(nom, colonne) : moyenne arithmétique
MAX(nom, colonne) : valeur maximum
MIN(nom, colonne) : valeur minimum
COUNT(*) ou COUNT(DISTINCT) nom, colonne) : nombre d'occurrences

- 4- LES OPERATEURS ENSEMBLISTES

```
- UNION  
- INTERSECT  
- EXCEPT (parfois MINUS)  
  
SELECT ...  
UNION  
SELECT ... ;  
UNION  
SELECT ... ;  
UNION  
SELECT ... ;  
EXCEPT  
SELECT ... ;
```

- 5- TEST D'EXISTENCE

Ce sont les opérateurs EXISTS or NOT EXISTS.

VI- LE LANGAGE DE MANIPULATION

- 1- MISE A JOUR DE DONNÉES

```
UPDATE nom-table  
SET nom-col1 = Expression  
[nom-col2 = Expression] ...  
[WHERE condition];
```

- 2- SUPPRESSION DE LIGNES

```
DELETE FROM nom-table  
[WHERE prédicat];
```

- 3- SUPPRESSION DE TOUTES LES LIGNES

```
TRUNCATE TABLE nom-table
```

- 4- INSERTION DE LIGNES

```
INSERT INTO nom-table [(nom-col1, nom-col2, ...)]  
VALUES (val1, val2, ...);
```

```
INSERT INTO nom-table [(nom-col1, nom-col2, ...)]  
SELECT ... ;
```

VII- AUTORISATIONS ET CONTRÔLES D'ACCES

- 1- PRIVILEGES SYSTEM

a) Attribution

```
GRANT privilège-system [, privilège-system... ] TO |User1 [, User2, ...]  
|rôle [, rôle2, ...] |public [WITH ADMIN OPTION]
```

b) Révocation

```
REVOKE privilège-system [, privilège-system... ] FROM |User1 [, User2, ...]  
|rôle1 [, rôle2, ...] |public
```

c) Les principaux privilèges système

```
CREATE DATABASE, CREATE TABLE, CREATE VIEW, BACKUP DATABASE, CREATE  
DEFAULT, CREATE FUNCTION, CREATE PROCEDURE, CREATE RULE.
```

- 2- PRIVILEGES OBJETS (TABLES ET VUES)

a) Attribution

```
GRANT priv-objet [, priv-objet... ] ON objet |User1 [, User2, ...]  
|rôle [, rôle2, ...] |PUBLIC [WITH GRANT OPTION]
```


b) Revocation

```
REVOKE priv-objet [,priv-objet,...] ON objet  
FROM
```

```
|User1 [,User2,...]  
|rôle [,rôle2,...] [CASCADE ]  
|public
```

c) Les principaux privilèges objet

```
ALTER, DELETE, INDEX, INSERT, SELECT, UPDATE[(nom_col1[,nom_col2,...]), ALL
```

VIII- CURSEURS

1- DECLARATION D'UN CURSEUR

```
DECLARE nom_curs CURSOR FOR  
SELECT...  
[UPDATE OF nom_col1 [,nom_col2]]
```

2- UTILISATION D'UN CURSEUR

```
OPEN nom_curs  
CLOSE nom_curs
```

```
FETCH nom_curs INTO :var1 [, :var2]...
```

```
Mise à jour de colonne :  
UPDATE nom-table SET nom_col1 = Expression  
[nom_col2 = Expression]...  
WHERE CURRENT OF nom_curs ;
```

```
3- LECTURE D'UNE LIGNE UNIQUE  
SELECT att1 [, att2] Into :var1 [, :var2] FROM ... WHERE ...
```

IX- TRIGGER

```
CREATE TRIGGER NomDéclencheur  
(BEFORE | AFTER) {INSERT | DELETE | UPDATE (of (nom_col1[,nom_col2,...])}  
ON NomTable  
[REFERENCING NEW|OLD|NEW_TABLE|OLD_TABLE AS nom  
[FOR EACH {ROW | STATEMENT}]  
[WHEN condition]  
BEGIN  
Lots Instructions SQL  
END
```

```
Suppression d'un trigger :  
DROP TRIGGER nom_trigger,
```

X- PROCÉDURE STOCKÉE

```
CREATE [OR REPLACE] PROCEDURE nom_proc  
[nom_param [IN|OUT|IN OUT] type_param,...]  
AS BEGIN  
corps_procedure  
END ;
```

```
CREATE [OR REPLACE] FUNCTION nom_fonct  
[nom_param [IN|OUT|IN OUT] type_param,...]  
RETURN type_retourne  
AS BEGIN  
corps_fonction  
END ;
```

```
Suppression d'une fonction ou procédure stockée :  
DROP PROCEDURE nom_proc ;  
DROP FUNCTION nom_fonc ;
```

XI- TRANSACTIONS

```
BEGIN TRANSACTION [nom_transaction]  
/* lots d'instructions SQL qui appartiennent à la même transaction */  
COMMIT [transaction [nom_transaction]]  
  
ou  
  
BEGIN TRANSACTION [nom_transaction]  
/* lots d'instructions SQL qui appartiennent à la même transaction */  
ROLLBACK [transaction [nom_transaction]]
```