

Mémento

MySQL

Version 3.0 (créé le 15/04/2022, modifié le XX/XX/XXXX)



SQL (Structured Query Language) est un langage servant à exploiter des bases de données relationnelles.

MySQL est très utilisé pour les bases de données de sites web qui utilisent le langage PHP.

Toutes les options facultatives sont représentées par des crochets [].

Table des matières

1. Prise en main.....	7
1.1. Outils nécessaires	7
2. Bases	7
2.1. Syntaxe	7
2.2. Les variables	7
2.2.1. Types de variables	7
2.3. Commentaires	8
2.4. Opérations mathématiques	9
2.5. Opérateurs de conditions	9
3. Les fonctions	10
3.1. Les fonctions arithmétiques.....	10
3.2. Les fonctions pour les chaînes de caractères	11
3.3. Les fonctions pour les dates.....	11
3.3.1. Les masques pour les dates	11
3.3.2. Les fonctions de manipulation des dates	12
3.4. Autres fonctions	13
3.5. Les fonctions d'agrégat (pour des sélections uniquement, ne concernent pas les conditions)	14
4. Requêtes d'initialisation de la base de données	14
4.1. Créer et utiliser une base de données	14
4.2. Supprimer une base de données.....	14
4.3. Créer une table (tableau de valeurs)	14
4.4. Les contraintes	15
4.5. Copier une table.....	15
4.6. Renommer une table	15

4.7. Supprimer une table.....	16
5. Requêtes de mise à jour du contenu de la table.....	16
5.1. Insérer un ou plusieurs enregistrement (ligne)	16
5.2. Copier un ou plusieurs enregistrements	16
5.3. Mettre à jour un ou plusieurs enregistrement(s) respectant la condition	16
5.4. Supprimer un ou plusieurs enregistrement(s) respectant la condition	16
6. Requêtes de mise à jour de la structure de la table	17
6.1. Ajouter une colonne.....	17
6.2. Supprimer une colonne.....	17
6.3. Modifier le type d'une colonne	17
6.4. Modifier le nom d'une colonne.....	17
7. Requêtes de sélection de données.....	17
7.1. Syntaxe de base.....	17
7.2. La clause SELECT	18
7.2.1. Sélection de tous les enregistrements de la table.....	18
7.2.2. Sélection de certains attributs de la table.....	18
7.2.3. Sélection d'un attribut de la table sans répétitions	18
7.2.4. Utilisation d'une fonction d'agrégat (ne renvoie qu'une seule valeur sans regroupement)	18
7.2.5. Sélection des n premiers enregistrements de la table avec p décalages (facultatif).....	18
7.3. Les jointures (clause JOIN)	19
7.3.1. Syntaxe de base	19
7.3.2. Jointure interne simple	19
7.3.3. Jointure externe gauche (avec des enregistrements de la table de gauche non présents dans la table de droite)	19

7.3.4. Jointure externe droite (avec des enregistrements de la table de droite non présents dans la table de gauche).....	19
7.3.5. Jointure externe entière (avec des tous les enregistrements non présents dans une autre table)	20
7.3.6. Jointure naturelle (pour deux tables avec une colonne du même nom).....	20
7.3.7. Produit cartésien (retourne chaque ligne d'une table avec chaque ligne d'une autre table)	20
7.4. Les alias.....	20
7.4.1. Alias sur les tables (pour faciliter les jointures)	20
7.4.2. Alias sur une fonction d'agrégat (pour nommer une colonne sans nom ou renommer une colonne)	20
7.5. La clause WHERE	21
7.5.1. Sélection des enregistrements de la table respectant la condition	21
7.6. Les clauses GROUP BY et HAVING	21
7.6.1. Utilisation d'une fonction d'agrégat sur chaque groupe d'enregistrements de même(s) attribut(s)	21
7.6.2. Utilisation d'une fonction d'agrégat sur chaque groupe d'enregistrements de même(s) attribut(s) avec une condition avant et après regroupement	21
7.7. La clause ORDER BY	21
7.7.1. Sélection des enregistrements de la table dans l'ordre croissant ..	21
7.7.2. Sélection des enregistrements de la table dans l'ordre décroissant	22
7.8. Les requêtes multiples de sélection.....	22
7.8.1. Les sous-requêtes.....	22
7.8.2. Sélection des enregistrements d'un attribut présents dans la première table ou la deuxième ou les deux (union)	22

7.8.3. Sélection des enregistrements d'un attribut présents dans la première table, mais pas la deuxième (différence)	22
7.8.4. Sélection des enregistrements d'un attribut présents dans la première table et la deuxième (intersection)	22
8. Les index.....	23
8.1. Créer un index	23
8.2. Supprimer un index	23
9. Les vues	23
9.1. Créer une vue	23
9.2. Utiliser la vue	23
9.3. Supprimer la vue	23
10. Autres requêtes de données.....	24
10.1. Requête de description d'une table	24
10.2. Les transactions	24
10.2.1. Valider une transaction	24
10.2.2. Annuler une transaction	24
11. Programmation procédurale et structurée ()	24
11.1. Syntaxe du bloc principal.....	24
11.2. Déclarations (dans le bloc DECLARE).....	25
11.2.1. Variables de types simples	25
11.2.2. Variables constantes.....	25
11.2.3. Variables de types d'une colonne.....	25
11.2.4. Variables de types d'une ligne entière d'une table	25
11.2.5. Curseurs explicites	25
11.2.6. Exceptions.....	26
11.3. Opération d'affectation.....	26
11.3.1. Affectation simple.....	26
11.3.2. Affectation avec une requête de sélection (ne fonctionne que si la sélection ne renvoie qu'une seule valeur)	26

11.4. Conditions.....	27
11.5. Boucles.....	27
11.6. Les curseurs implicites et explicites.....	28
11.6.1. Les curseurs implicites.....	28
11.6.2. Les curseurs explicites	28
11.7. Les procédures.....	28
11.7.1. Créer une procédure.....	28
11.7.2. Retourner une valeur de la procédure	29
11.7.3. Faire un appel à la procédure	29
11.7.4. Supprimer une procédure.....	29
11.8. Les fonctions	29
11.8.1. Créer une fonction.....	29
11.8.2. Retourner une valeur de la fonction	29
11.8.3. Faire un appel à la fonction.....	29
11.8.4. Supprimer une fonction.....	29
11.9. Les exceptions (dans le bloc EXCEPTION)	30
11.9.1. Exceptions prédéfinies.....	30
11.9.2. Provoquer une erreur personnalisée	30
11.9.3. Gestion des erreurs (dans le bloc EXCEPTION).....	30
11.10. Les instructions de base.....	30

1. Prise en main

1.1. Outils nécessaires

- UwAmp ou Laragon (recommandé) ou WampServer (contenant MySQL version 8.0 et PhpMyAdmin version 8.1)
- Un logiciel de codage (ex : Notepad++, Visual Studio Code...)

2. Bases

2.1. Syntaxe

```
requête1;  
requête2;  
requête3;
```

2.2. Les variables

2.2.1. Types de variables

2.2.1.1. Numériques

Type	Minimum	Maximum	Description
INT	-2 147 483 648	2 147 483 647	Nombre entier
TINYINT	-128	127	Nombre entier
SMALLINT	-32 768	32 767	Nombre entier
MEDIUMINT	-8 388 608	8 388 607	Nombre entier
BIGINT	-9 223 372 036 854 775 808	9 223 372 036 854 775 807	Nombre entier
FLOAT	-3.4 _{x10} ^{38f}	3.4 _{x10} ^{38f}	Nombre décimal

2.2.1.2. Caractères

Type	Limites	Description
CHAR(X)	Longueur fixée à X caractères (maximum : 255)	Chaîne de caractères entre ''
VARCHAR(X)	Longueur inférieure ou égale à X caractères (maximum : 255)	Chaîne de caractères entre ''
TEXT	Infinité de caractères	Chaîne de caractères entre ''

2.2.1.3. Dates et heures

Type	Format	Description
DATE	YYYY-MM-DD	Date entre ''
DATETIME	YYYY-MM-DD hh:mm:ss	Date et heure entre ''
TIME	hh:mm:ss	Heure entre ''
YEAR	YYYY	Année entre 1901 et 2155

2.2.1.4. Autres types

Type	Description
BOOL	Valeur pouvant être TRUE ou FALSE (converti en TINYINT sous certaines versions)

2.3. Commentaires

--Commentaire tenant sur une ligne

#Commentaire tenant sur une ligne

/*

Commentaire pouvant être sur une ou plusieurs lignes

*/

2.4. Opérations mathématiques

Instruction	Description
$1 + 2$	Renvoie 3
$3 - 1$	Renvoie 2
$6 * 4$	Renvoie 24
$5.0 / 2.0$	Renvoie 2.5
$5 / 2$	Renvoie 2 (le quotient sans décimal)

2.5. Opérateurs de conditions

Condition	Description de ce que vérifie la condition
$a = b$	a égal à b (seulement en contenu : $1='1'$)
$a < b$	a strictement inférieur à b
$a > b$	a strictement supérieur à b
$a \leq b$	a supérieur ou égal à b
$a \neq b$	a n'est pas égal à b (seulement en contenu : $1 \neq '1'$)
$a \lt;> b$	a n'est pas égal à b (seulement en contenu : $1 \lt;> '1'$)
$a = \text{ALL } b$	a est égale à toutes les valeurs dans b (qui peut être une sous-requête ou une liste sous forme : $(val1, val2...)$)
$a = \text{ANY } b$	a est égale à au moins une valeur dans b (qui peut être une sous-requête ou une liste sous forme : $(val1, val2...)$)
$a \text{ IN } b$	a est présent dans b (qui peut être une sous-requête ou une liste sous forme : $(val1, val2...)$)
$a \text{ BETWEEN } b \text{ AND } c$	a est compris entre b et c

a LIKE ' <i>SousChaine</i> '	a contient la sous-chaîne (avec _ correspondant à 1 caractère et % correspondant à 0 ou plusieurs caractères)
a IS [NOT] NULL	Tester si une variable est nulle
a IS [NOT] UNKNOWN	Tester si une variable est définie
OR	À mettre entre deux conditions, permet d'avoir une des deux conditions qui doit être vraie
AND	À mettre entre deux conditions, permet d'avoir deux conditions qui doivent être vraies
NOT <i>condition</i>	Ne doit pas respecter la condition

3. Les fonctions

3.1. Les fonctions arithmétiques

Fonction	Description
ROUND(n , d)	Arrondit n au réel à d chiffres après la virgule ou à l'entier si d n'est pas renseigné
TRUNC(n , d)	Tronque n à d chiffres après la virgule ou à 0 si d n'est pas renseigné
POWER(n , m)	Renvoie n à la puissance m (si n est négatif, m doit être un entier)
CEIL(n)	Renvoie un entier directement supérieur ou égal à n
FLOOR(n)	Renvoie un entier directement inférieur ou égal à n (partie entière de n)
ABS(n)	Renvoie la valeur absolue de n
MOD(n , m)	Renvoie le reste de la division de n par m

SQRT(<i>n</i>)	Renvoie la racine carrée
SIGN(<i>n</i>)	Renvoie -1 si <i>n</i> est négatif, 1 si <i>n</i> est positif et 0 si <i>n</i> égal à 0

3.2. Les fonctions pour les chaînes de caractères

Fonction	Description
LENGTH(<i>chaîne</i>)	Renvoie la longueur de la chaîne
SUBSTR(<i>chaîne</i> , <i>debut</i> [, <i>Longueur</i>])	Renvoie la position (en commençant à une certaine position et allant jusqu'à la longueur fixée ou à la fin)
UPPER(<i>chaîne</i>)	Convertit en majuscule
LOWER(<i>chaîne</i>)	Convertit en minuscule
INITCAP(<i>chaîne</i>)	Met en majuscule la première lettre et en minuscule les autres
TRANSLATE(<i>chaîne</i> , <i>c1</i> , <i>c2</i>)	Remplace chaque caractère <i>c1</i> par <i>c2</i> dans la chaîne
REPLACE(<i>chaîne</i> , <i>ch1</i> , <i>ch2</i>)	Remplace chaque chaîne <i>ch1</i> par <i>ch2</i> dans la chaîne

3.3. Les fonctions pour les dates

3.3.1. Les masques pour les dates

Masque	Description
'%Y'	Année (ex : 2024)
'%u' ou '%v'	Numéro de la semaine dans l'année
'%c'	Numéro du mois (ex : 5)
'%m'	Numéro du mois (ex : 05)
'%e'	Numéro du jour dans le mois (ex : 8)
'%d'	Numéro du jour dans le mois (ex : 08)

'%w'	Numéro du jour dans la semaine (ex : 5) avec dimanche égal à 0
'%h'	Heure sur 12 heures (ex : 09)
'%H'	Heure sur 24 heures (ex : 21)
'%i'	Minutes (ex : 05)
'%s'	Secondes (ex : 30)
'%T'	Heure complète (ex : 12:34:56)
'%M'	Mois en lettres
'%b'	Mois abrégé (3 lettres)
'%W'	Jour en lettres
'%a'	Jour abrégé (3 lettres)

3.3.2. Les fonctions de manipulation des dates

Fonction	Description
NOW()	Renvoie la date et l'heure actuelle (heure à laquelle le script a démarré)
SYSDATE()	Renvoie la date et l'heure courante (heure à laquelle la fonction a été appelée)
CURDATE()	Renvoie la date actuelle
	Renvoie l'heure actuelle
DATE_ADD(<i>date</i> , <i>nombre</i> SECOND)	Ajoute un certain nombre de secondes
DATE_SUB(<i>date</i> , <i>nombre</i> SECOND)	Soustrait un certain nombre de secondes
DATE_ADD(<i>date</i> , <i>nombre</i> MINUTE)	Ajoute un certain nombre de minutes
DATE_SUB(<i>date</i> , <i>nombre</i> MINUTE)	Soustrait un certain nombre de minutes
DATE_ADD(<i>date</i> , <i>nombre</i> HOUR)	Ajoute un certain nombre d'heures
DATE_SUB(<i>date</i> , <i>nombre</i> HOUR)	Soustrait un certain nombre d'heures
DATE_ADD(<i>date</i> , <i>nombre</i> DAY)	Ajoute un certain nombre de jours

<code>DATE_SUB(date, nombre DAY)</code>	Soustrait un certain nombre de jours
<code>DATE_ADD(date, nombre WEEK)</code>	Ajoute un certain nombre de semaines
<code>DATE_SUB(date, nombre WEEK)</code>	Soustrait un certain nombre de semaines
<code>DATE_ADD(date, nombre MONTH)</code>	Ajoute un certain nombre de mois
<code>DATE_SUB(date, nombre MONTH)</code>	Soustrait un certain nombre de mois
<code>DATE_ADD(date, nombre YEAR)</code>	Ajoute un certain nombre d'années
<code>DATE_SUB(date, nombre YEAR)</code>	Soustrait un certain nombre d'années
<code>DATEDIFF(date1, date2)</code>	Renvoie le nombre de jours de différence
<code>STR_TO_DATE(chaine, masque)</code>	Convertit une chaîne de caractères en date (ex : <code>STR_TO_DATE('10/12/2024', '%d/%m/%Y')</code>)
<code>DATE_FORMAT(date, masque)</code>	Convertit une date en chaîne de caractères (ex : <code>DATE_FORMAT(NOW(), '%d/%m/%Y')</code>)

3.4. Autres fonctions

Fonction	Description
<code>CHR(n)</code>	Retourne le caractère dont le code (ASCII ou EBCDIC) est égal à l'expression numérique entré en paramètre
<code>GREATEST(exp1, exp2...)</code>	Retourne la plus grande des valeurs des expressions arguments
<code>LEAST(exp1, exp2...)</code>	Retourne la plus petite des valeurs des expressions arguments
<code>COALESCE(exp1, exp2...)</code>	Retourne la première valeur différente de NULL des expressions arguments, s'il y en a une, et la valeur NULL s'il n'y en a pas

3.5. Les fonctions d'agrégat (pour des sélections uniquement, ne concernent pas les conditions)

Fonction	Description
AVG(<i>expression</i>)	Moyenne des valeurs d'une colonne
SUM(<i>expression</i>)	Somme des valeurs d'une colonne
MIN(<i>expression</i>)	La plus petite des valeurs d'une colonne
VARIANCE(<i>expression</i>)	La plus grande des valeurs d'une colonne
STDDEV(<i>expression</i>)	Écart-type ou déviation standard
COUNT(*)	Nombre de lignes
COUNT(<i>expression</i>)	Nombre de lignes ayant pour valeur non nulle
COUNT(DISTINCT <i>expression</i>)	Nombre de lignes ayant des valeurs distinctes non nulles

4. Requêtes d'initialisation de la base de données

4.1. Créer et utiliser une base de données

```
CREATE DATABASE [IF NOT EXISTS] maBaseDeDonnees CHARACTER SET utf8;  
USE maBaseDeDonnees;
```

4.2. Supprimer une base de données

```
DROP DATABASE [IF EXISTS] maBaseDeDonnees;
```

4.3. Créer une table (tableau de valeurs)

```
CREATE TABLE [IF NOT EXISTS] table1 (  
    attribut1 type1 [contrainte1] [AUTO_INCREMENT],  
    attribut2 type2 [contrainte2],  
    attribut3 type3 [contrainte3],  
    PRIMARY KEY (attribut1),
```

```
[FOREIGN KEY (attribut3) REFERENCES table2(attribut1)]  
) [ENGINE = InnoDB];
```

AUTO_INCREMENT permet de créer automatiquement les valeurs des clés primaires si le type est un nombre entier.

ENGINE=InnoDB : facultatif, mais conseillé en cas d'utilisation de PhpMyAdmin pour faire des requêtes.

4.4. Les contraintes

Contrainte	Description
NOT NULL	Permet de rendre obligatoire l'entrée d'une valeur dans un attribut
CHECK(<i>condition</i>)	Permet de vérifier la condition avant d'insérer un enregistrement
UNIQUE	Permet d'éviter l'insertion de plusieurs valeurs identiques
DEFAULT <i>valeurParDefaut</i>	Permet de remplacer une valeur non renseignée par une autre
PRIMARY KEY (<i>attribut1</i>)	Permet de créer une clé primaire. Il s'agit généralement d'un identifiant, et sa valeur est unique
FOREIGN KEY (<i>attribut3</i>) REFERENCES <i>table2</i> (<i>attribut7</i>)	Permet de créer une clé étrangère en relation avec une autre table (déjà existante)

4.5. Copier une table

```
CREATE TABLE table2 AS (SELECT * FROM table1);
```

4.6. Renommer une table

```
RENAME ancienNom TO nouveauNom;
```

4.7. Supprimer une table

```
DROP TABLE [IF EXISTS] table1;
```

5. Requêtes de mise à jour du contenu de la table

5.1. Insérer un ou plusieurs enregistrement (ligne)

```
INSERT INTO table1 [(attribut1, attribut2...)] VALUES  
(valeur1, valeur2...)[,  
(valeur3, valeur4...)...];
```

5.2. Copier un ou plusieurs enregistrements

```
INSERT INTO table1 (attribut1, attribut2...)  
SELECT attribut1, attribut2... FROM table1  
WHERE condition;
```

5.3. Mettre à jour un ou plusieurs enregistrement(s) respectant la condition

```
UPDATE table1  
SET attribut1 = valeur1 [, attribut2 = valeur2...]  
WHERE condition;
```

5.4. Supprimer un ou plusieurs enregistrement(s) respectant la condition

```
DELETE FROM table1  
WHERE condition;
```


6. Requêtes de mise à jour de la structure de la table

6.1. Ajouter une colonne

```
ALTER TABLE table1  
ADD attribut type;
```

6.2. Supprimer une colonne

```
ALTER TABLE table1  
DROP [COLUMN] attribut;
```

6.3. Modifier le type d'une colonne

```
ALTER TABLE table1  
MODIFY attribut nouveauType;
```

6.4. Modifier le nom d'une colonne

```
ALTER TABLE table1  
RENAME ancienNom TO nouveauNom;
```

7. Requêtes de sélection de données

7.1. Syntaxe de base

```
SELECT attribut1, attribut2... FROM table1  
[WHERE condition1]  
[GROUP BY attribut1 [, attribut2...]  
[HAVING condition2]]  
[ORDER BY attribut1 [DESC], attribut2 [DESC]...];
```

7.2. La clause SELECT

7.2.1. Sélection de tous les enregistrements de la table

```
SELECT * FROM table1;
```

7.2.2. Sélection de certains attributs de la table

```
SELECT attribut1, attribut2 FROM table1;
```

7.2.3. Sélection d'un attribut de la table sans répétitions

```
SELECT DISTINCT attribut1 FROM table1;
```

7.2.4. Utilisation d'une fonction d'agrégat (ne renvoie qu'une seule valeur sans regroupement)

```
SELECT maFonction(attribut1) FROM table1;
```

7.2.5. Sélection des *n* premiers enregistrements de la table avec *p* décalages (facultatif)

```
SELECT * FROM table1  
[ORDER BY attribut1 [DESC], attribut2 [DESC]...]  
LIMIT n [, p];
```

ou

```
SELECT * FROM table1  
[ORDER BY attribut1 [DESC], attribut2 [DESC]...]  
LIMIT n [OFFSET p];
```

7.3. Les jointures (clause JOIN)

Les jointures permettent de sélectionner des attributs de plusieurs tables différentes (les jointures peuvent s'accumuler).

7.3.1. Syntaxe de base

```
SELECT * FROM table1
JOIN table2 ON table1.attribut = table2.attribut
[JOIN table3 ON table2.attribut = table3.attribut
...]
[WHERE condition];
```

7.3.2. Jointure interne simple

```
SELECT table1.attribut2, table2.attribut3 FROM table1
[INNER] JOIN table2 ON table1.attribut1 = table2.attribut4;
```

Autre solution (à utiliser uniquement pour deux attributs identiques de deux tables différentes) :

```
SELECT table1.attribut2, table2.attribut3 FROM table1
[INNER] JOIN table2 USING(attribut1);
```

7.3.3. Jointure externe gauche (avec des enregistrements de la table de gauche non présents dans la table de droite)

```
SELECT table1.attribut1, table2.attribut3 FROM table1
LEFT [OUTER] JOIN table2 ON table1.attribut1 = table2.attribut2;
```

7.3.4. Jointure externe droite (avec des enregistrements de la table de droite non présents dans la table de gauche)

```
SELECT table1.attribut1, table2.attribut3 FROM table1
RIGHT [OUTER] JOIN table2 ON table1.attribut1 = table2.attribut2;
```

7.3.5. Jointure externe entière (avec des tous les enregistrements non présents dans une autre table)

```
SELECT table1.attribut1, table2.attribut3 FROM table1  
FULL [OUTER] JOIN table2 ON table1.attribut1 = table2.attribut2;
```

7.3.6. Jointure naturelle (pour deux tables avec une colonne du même nom)

```
SELECT table1.attribut1, table2.attribut3 FROM table1  
NATURAL JOIN table2;
```

7.3.7. Produit cartésien (retourne chaque ligne d'une table avec chaque ligne d'une autre table)

```
SELECT table1.attribut1, table2.attribut3 FROM table1  
CROSS JOIN table2;
```

7.4. Les alias

7.4.1. Alias sur les tables (pour faciliter les jointures)

```
SELECT t1.attribut2, t2.attribut3 FROM table1 AS t1  
JOIN table2 AS t2 ON t1.attribut1 = t2.attribut4;
```

7.4.2. Alias sur une fonction d'agrégat (pour nommer une colonne sans nom ou renommer une colonne)

```
SELECT maFonction(attribut1) AS nomColonne FROM table1;
```

7.5. La clause WHERE

7.5.1. Sélection des enregistrements de la table respectant la condition

```
SELECT * FROM table1  
WHERE condition;
```

7.6. Les clauses GROUP BY et HAVING

7.6.1. Utilisation d'une fonction d'agrégat sur chaque groupe d'enregistrements de même(s) attribut(s)

```
SELECT maFonction(attribut1), attribut2, attribut3... FROM table1  
GROUP BY attribut2, attribut3...;
```

7.6.2. Utilisation d'une fonction d'agrégat sur chaque groupe d'enregistrements de même(s) attribut(s) avec une condition avant et après regroupement

```
SELECT maFonction(attribut1), attribut2, attribut3... FROM table1  
WHERE conditionAvantRegroupement  
GROUP BY attribut2, attribut3...  
HAVING conditionApresRegroupement;
```

7.7. La clause ORDER BY

7.7.1. Sélection des enregistrements de la table dans l'ordre croissant

```
SELECT * FROM table1  
[WHERE condition]  
ORDER BY attribut1 [ASC][, attribut2 [ASC]...];
```

7.7.2. Sélection des enregistrements de la table dans l'ordre décroissant

```
SELECT * FROM table1  
[WHERE condition]  
ORDER BY attribut1 DESC[, attribut2 DESC...];
```

7.8. Les requêtes multiples de sélection

7.8.1. Les sous-requêtes

Les sous-requêtes sont des requêtes effectuées dans d'autres requêtes. Elles remplacent souvent les tables, mais peuvent également remplacer les attributs uniquement si elles ne renvoient qu'une seule valeur.

7.8.2. Sélection des enregistrements d'un attribut présents dans la première table ou la deuxième ou les deux (union)

```
SELECT attribut1 FROM table1  
UNION  
SELECT attribut1 FROM table2;
```

7.8.3. Sélection des enregistrements d'un attribut présents dans la première table, mais pas la deuxième (différence)

```
SELECT attribut1 FROM table1  
MINUS  
SELECT attribut1 FROM table2;
```

7.8.4. Sélection des enregistrements d'un attribut présents dans la première table et la deuxième (intersection)

```
SELECT attribut1 FROM table1  
INTERSECT  
SELECT attribut1 FROM table2;
```

8. Les index

Les index permettent de gagner en temps d'exécution pour une sélection d'une colonne avec beaucoup de valeurs.

8.1. Créer un index

```
CREATE INDEX monIndex ON table1 [(attribut1[, attribut2...])];
```

8.2. Supprimer un index

```
DROP INDEX monIndex;
```

9. Les vues

Les vues permettent de réaliser des requêtes de sélections plus facilement. Elles se comportent comme des tables pour les requêtes de sélection.

9.1. Créer une vue

```
CREATE VIEW maVue AS SELECT * FROM table1;
```

9.2. Utiliser la vue

```
SELECT attribut1, attribut2... FROM maVue;
```

9.3. Supprimer la vue

```
DROP VIEW [IF EXISTS] maVue;
```

10. Autres requêtes de données

10.1. Requête de description d'une table

```
DESC table1;
```

ou

```
DESCRIBE table1;
```

10.2. Les transactions

10.2.1. Valider une transaction

```
COMMIT;
```

10.2.2. Annuler une transaction

```
ROLLBACK;
```

11. Programmation procédurale et structurée ()

11.1. Syntaxe du bloc principal

```
[DECLARE  
    declarations;  
BEGIN  
    instructions;  
[EXCEPTION  
    gestion_des_erreurs;  
END;
```

Note : Le bloc DECLARE peut être retiré si aucune déclaration n'a été faite. Le bloc EXCEPTION permet de gérer les erreurs du programme, il n'est donc pas obligatoire. Il est possible de faire plusieurs blocs BEGIN imbriqués.

11.2. Déclarations (dans le bloc DECLARE)

11.2.1. Variables de types simples

```
v_variable1 type1 = valeur;  
v_variable2 type2;  
v_variable3 type3;
```

11.2.2. Variables constantes

```
v_variable1 CONSTANT type1 = valeur;
```

11.2.3. Variables de types d'une colonne

```
v_variable1 table1.attribut1%TYPE;
```

11.2.4. Variables de types d'une ligne entière d'une table

```
v_variable1 table1%ROWTYPE;
```

11.2.5. Curseurs explicites

11.2.5.1. Curseur sans paramètres

```
CURSOR c_curseur IS (SELECT * FROM table1 WHERE condition [ORDER BY  
attribut1 [DESC]]);
```

11.2.5.2. Curseur avec paramètres

```
CURSOR c_curseur(v_attribut1 typeSansParentheses) IS (SELECT * FROM  
table1 WHERE attribut1 = v_attribut1 [ORDER BY attribut1 [DESC]]);
```

11.2.5.3. Curseurs avec verrouillage des données (pour une mise à jour de la table notamment)

```
CURSOR c_curseur IS (SELECT * FROM table1 WHERE condition [ORDER BY attribut1 [DESC]]) FOR UPDATE;
```

Note : Pour désigner l'élément sélectionné par le curseur dans une condition, entrez : `CURRENT OF c_curseur`

11.2.6. Exceptions

11.2.6.1. Exception simple

```
NOM_EXCEPTION_PERSONNALISEE EXCEPTION;
```

11.2.6.2. Exception créée à l'aide d'un numéro d'exception

```
NOM_EXCEPTION_PERSONNALISEE EXCEPTION;  
PRAGMA EXCEPTION_INIT(NOM_EXCEPTION_PERSONNALISEE,  
numero_exception);
```

11.3. Opération d'affectation

11.3.1. Affectation simple

```
v_variable = valeur;
```

11.3.2. Affectation avec une requête de sélection (ne fonctionne que si la sélection ne renvoie qu'une seule valeur)

```
SELECT attribut1 INTO v_variable FROM table1 WHERE condition;
```

11.4. Conditions

Instruction	Description
IF <i>condition1</i> THEN <i>instruction1</i> ; END IF;	Si <i>condition1</i> est vraie, alors on exécute <i>instruction1</i>
IF <i>condition1</i> THEN <i>instruction1</i> ; ELSE <i>instruction2</i> ; END IF;	Si <i>condition1</i> est vraie, alors on exécute <i>instruction1</i> , sinon, on exécute <i>instruction2</i>
IF <i>condition1</i> THEN <i>instruction1</i> ; } ELSIF <i>condition2</i> THEN <i>instruction2</i> ; ELSE <i>instruction3</i> ; END IF;	Si <i>condition1</i> est vraie, alors on exécute <i>instruction1</i> , sinon, si <i>condition2</i> est vraie, on exécute <i>instruction2</i> , sinon, on exécute <i>instruction3</i>

11.5. Boucles

Instruction	Description
FOR <i>i</i> IN 0.. <i>n</i> LOOP <i>instruction1</i> ; END LOOP;	On répète <i>n</i> fois <i>instruction1</i> pour <i>i</i> allant de 0 à <i>n</i> compris (pas besoin de déclarer la variable, elle est déclarée implicitement et est détruite à la fin de la boucle)
FOR <i>elt</i> IN (SELECT * FROM <i>table1</i>) LOOP <i>instruction1</i> ; END LOOP;	On parcourt la requête de sélection (ou une chaîne de caractère) pour <i>elt</i> prenant tous les enregistrements de la requête (pas besoin de déclarer la variable, elle est déclarée implicitement et est détruite à la fin de la boucle)
WHILE <i>condition</i> LOOP <i>instruction1</i> ; END LOOP;	On répète jusqu'à ce que <i>condition</i> soit fausse (peut ne pas être répété)

LOOP <i>instruction1</i> ; EXIT WHEN <i>condition</i> ; <i>instruction2</i> ; END LOOP;	On répète jusqu'à ce que condition soit fausse (<i>instruction1</i> est forcément répété une fois)
EXIT;	Permet de sortir d'une boucle sans la terminer

11.6. Les curseurs implicites et explicites

11.6.1. Les curseurs implicites

Curseur	Description
	Nombre de lignes traitées par la requête précédentes de mise à jour de la table
	Vérifier que le curseur est vide ou la dernière ligne est déjà atteinte
	Vérifier que le curseur n'est pas vide ou la dernière ligne n'est pas encore atteinte
	Vérifier que le curseur est ouvert

11.6.2. Les curseurs explicites

11.7. Les procédures

11.7.1. Créer une procédure

11.7.2. Retourner une valeur de la procédure

11.7.3. Faire un appel à la procédure

11.7.4. Supprimer une procédure

11.8. Les fonctions

11.8.1. Créer une fonction

11.8.2. Retourner une valeur de la fonction

11.8.3. Faire un appel à la fonction

11.8.4. Supprimer une fonction

11.9. Les exceptions (dans le bloc EXCEPTION)

11.9.1. Exceptions prédéfinies

Nom exception	Numéro exception	Description

11.9.2. Provoquer une erreur personnalisée

```
RAISE NOM_EXCEPTION_PERSONNALISEE;
```

11.9.3. Gestion des erreurs (dans le bloc EXCEPTION)

```
WHEN NOM_EXCEPTION1 THEN  
    instructions1;  
WHEN NOM_EXCEPTION2 THEN  
    instructions2;
```

11.10. Les instructions de base

Instruction	Description
	Affiche un texte dans la console avec un retour à la ligne