Loric Informatique

MySQL

Version 1.0 (créé le 15/04/2022, modifié le 30/01/2024)

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 seront représentées par des [].

Outils nécessaires :

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

Table des matières :

I. Bases

I.1. Syntaxe

```
requête1;
requête2;
requête3;
```

I.2. Types de variables

I.2.1. Chaînes de caractères

Type	Description
CHAR(X)	Longueur fixée à X caractères (maximum : 255)
VARCHAR(X)	Longueur inférieure ou égale à X caractères (maximum : 255)
TEXT	Aucune limite de caractères

I.2.2. Numériques

Type	Minimum	Maximum	Description
INT	-2147483648	2147483647	Nombre entier
FLOAT	$-3.402823466_{x10}{}^{38}$	$3.402823466_{x10}{}^{38}$	Nombre décimal
TINYINT	-128	127	Nombre entier
SMALLINT	-32768	32767	Nombre entier
MEDIUMINT	-8388608	8388607	Nombre entier
BIGINT	-9223372036854775808	9223372036854775807	Nombre entier

I.2.3. Autres types de variables

Type	Description
BOOL	0 = FALSE, 1 = TRUE (converti en TINYINT sous certaines versions)
DATE	Format : 'YYYY-MM-DD' (utiliser STR_TO_DATE() pour une meilleure compatibilité. Voir les fonctions pour les dates)
DATETIME	Format : 'YYYY-MM-DD hh:mm:ss'
TIME	Format : 'hh:mm:ss'
YEAR	Min : 1901, Max : 2155

I.3. Conditions

Opérateurs de comparaisons	= != (ou ‹›) ‹ ‹= › ›=
Connecteurs logiques	OR, AND
Opérateur de négation	NOT
Opérateurs mathématiques	+ - * /
Comparaison logique	IS [NOT] {TRUE|FALSE|UNKNOWN}
Comparaison avec valeur	IS [NOT] NULL
Intervalle	*valeur* BETWEEN *borne_basse* AND *borne_haute*
Comparaison à une liste de valeurs	*valeur* [NOT] IN (*liste*)
Comparaison à toutes les valeurs de la liste	ALL (*liste*)
Comparaison à une valeur contenue dans la liste	ANY (*liste*)
Comparaison avec une chaîne de caractères en MAJUSCULE	UPPER(*valeur*) = '*CHAINE*'
Comparaison avec une chaîne de caractères en minuscule	LOWER(*valeur*) = '*chaine*'
Comparaison avec une chaîne de caractères avec une majuscule au début et le reste en minuscules	INITCAP(*valeur*) = '*Chaine*'
Recherche d'une chaîne de caractères contenant une sous-chaîne de caractères avec : _ : 1 caractère % : 0 ou plusieurs caractères	LIKE '*SousChaine*'

I.4. Commentaires

```
-- Commentaire tenant sur une ligne

# Commentaire tenant sur une ligne (sur MySQL uniquement)

/* Commentaire pouvant être sur une ou plusieurs lignes */
```

II. Les fonctions

II.1. Les fonctions arithmétiques

Fonction	Utilité

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 supérieur 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(*n*)	Renvoie la racine carrée
SIGN(*n*)	Renvoie -1 si *n* est négatif, 1 si *n* est positif et 0 si *n* égal à 0

II.2. Les fonctions pour les chaînes de caractères

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

II.3. Les fonctions pour les dates

II.3.1. Les masques pour les dates

Masque	Description
`'%Y'`	Année (ex : 2024)
`'%u' ou '%v'`	Numéro de la semaine dans l'année
`'%c' ou '%m'`	Numéro du mois (ex : 5 ou 05)
`'%e' ou '%d'`	Numéro du jour dans le mois (ex : 7 ou 07)
`'%w'`	Numéro du jour dans la semaine (ex : 5) avec dimanche égal à 0
`'%h'`	Heure sur 12 heures (ex : 08)
`'%H'`	Heure sur 24 heures (ex : 20)
`'%i'`	Minutes (ex : 05)
`'%s'`	Secondes (ex : 30)
%T	Heure complète (ex : 12:34:56)
`'%M'`	Mois en lettres
`'%b'`	Mois abrégé
`'%W'`	Jour en lettres
`'%a'`	Jour abrégé

II.3.2. Les fonctions de manipulation des dates

Fonction	Utilité
`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

`NOW()`	Renvoie l'heure actuelle

`DATE_ADD(`*`date, nombre`* `SECOND)`
ou
`DATE_SUB(`*`date, nombre`* `SECOND)`
	Ajoute ou soustrait un certain nombre de secondes

`DATE_ADD(`*`date, nombre`* `MINUTE)`
ou
`DATE_SUB(`*`date, nombre`* `MINUTE)`
	Ajoute ou soustrait un certain nombre de minutes

`DATE_ADD(`*`date, nombre`* `HOUR)`
ou
`DATE_SUB(`*`date, nombre`* `HOUR)`
	Ajoute ou soustrait un certain nombre d'heures

`DATE_ADD(`*`date, nombre`* `DAY)`
ou
`DATE_SUB(`*`date, nombre`* `DAY)`
	Ajoute ou soustrait un certain nombre de jours

`DATE_ADD(`*`date, nombre`* `WEEK)`
ou
`DATE_SUB(`*`date, nombre`* `WEEK)`
	Ajoute ou soustrait un certain nombre de semaines

`DATE_ADD(`*`date, nombre`* `MONTH)`
ou
`DATE_SUB(`*`date, nombre`* `MONTH)`
	Ajoute ou soustrait un certain nombre de mois

`DATE_ADD(`*`date, nombre`* `YEAR)`
ou
`DATE_SUB(`*`date, nombre`* `YEAR)`
	Ajoute ou soustrait un certain nombre d'années

`DATEDIFF(`*`date1, date2`*`)`	Renvoie le nombre de jours de différence
`STR_TO_DATE(`*`chaine, masque`*`)`	Convertit une chaîne de caractères en dates (ex : `TO_DATE('10/12/2024', '%d/%m/%Y')`

II.4. Autres fonctions

Fonction	Utilité
`CHR(`*`n`*`)`	Retourne le caractère dont le code (ASCII ou EBCDIC) est égal à l'expression numérique entré en paramètre
`GREATEST(`*`exp1, exp2…`*`)`	Retourne la plus grande des valeurs des expressions arguments
`LEAST(`*`exp1, exp2…`*`)`	Retourne la plus petite des valeurs des expressions arguments

`COALESCE(exp1, exp2…)`	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

II.5. Fonctions d'agrégat (pour des sélections uniquement, ne concernent pas les conditions)

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

III. Requêtes d'initialisation de la base de données

III.1. Créer et utiliser la base de données

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

III.2. Supprimer la base de données

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

III.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.

III.4. Les contraintes

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

III.5. Renommer une table

```
RENAME ancienNom TO nouveauNom;
```

III.6. Supprimer une table

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

III.7. Copier une table

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

IV. Requêtes de mise à jour de la table

IV.1. Insérer un enregistrement (ligne)

```
INSERT INTO table1 VALUES
(valeur1, valeur2…)[,
(valeur3, valeur4…)];
```

ou

```
INSERT INTO table1 (attribut1, attribut2…) VALUES
(valeur1, valeur2…)[,
```

```
(valeur3, valeur4…)];
```

IV.2. Mettre à jour un ou plusieurs enregistrement(s)

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

IV.3. Supprimer un ou plusieurs enregistrement(s)

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

IV.4. Modifier la structure de la table

IV.4.1. Ajouter une colonne

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

IV.4.2. Supprimer une colonne

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

IV.4.3. Modifier le type d'une colonne

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

IV.4.4. Modifier le nom d'une colonne

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

V. Requêtes de sélection de données

V.1. Sélection de tous les enregistrements de la table

```
SELECT * FROM table1;
```

V.2. Sélection des enregistrements de la table respectant la condition

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

V.3. Sélection des enregistrements de la table dans un ordre spécifique

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

V.4. Sélection des *n* premiers enregistrements avec *p* décalages (facultatif) de la table

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

ou

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

V.5. Sélection des attributs de la table avec des enregistrements respectant la condition

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

V.6. Sélection d'un attribut de la table sans répétitions d'enregistrements respectant la condition

```
SELECT DISTINCT(attribut1) FROM table1
WHERE condition;
```

ou

```
SELECT DISTINCT attribut1 FROM table1
WHERE condition;
```

V.7. Les jointures (sélection des attributs de la table avec une ou plusieurs jointure(s) à une autre table)

V.7.1. Jointure interne simple

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

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)
WHERE condition;
```

V.7.2. 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
WHERE condition;
```

V.7.3. 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
WHERE condition;
```

V.7.4. 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
WHERE condition;
```

V.7.5. Jointure naturelle (pour deux tables avec une colonne du même nom)

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

V.7.6. 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
WHERE condition;
```

V.8. Utilisation d'une fonction d'agrégat

```
SELECT maFonction(attribut1, attribut2…) FROM table1
WHERE condition;
```

V.9. Les alias

V.9.1. Alias sur les tables (pour faciliter les jointures)

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

V.9.2. Alias sur une fonction d'agrégat (pour nommer une colonne sans nom)

```
SELECT COUNT(*) AS nomColonne FROM table1
WHERE condition;
```

V.10. Les groupes

Exemple : Calculer le nombre d'enregistrements de la table pour chaque enregistrement d'attribut du même nom respectant la condition 1 avant le groupement et la condition 2 après le groupement

```
SELECT attribut1 [, attribut2…], COUNT(*) AS variable FROM table1
WHERE condition1
GROUP BY attribut1 [, attribut2…]
[HAVIING condition2]
```

```
[ORDER BY attribut1 [DESC], attribut2 [DESC]…];
```

V.11. 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;
```

V.12. 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;
```

V.13. Sélection des enregistrements d'un attribut présents dans la première table et la deuxième (intersection)

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

VII. Autres requêtes de données

VII.1. Requête de description d'une table

```
DESC table1;
```

ou

```
DESCRIBE table1;
```

VII.2. Les transactions

Valider une transaction : `COMMIT;`

Annuler une transaction : `ROLLBACK;`