Mémento

MySQL

Version 2.0 (créé le 15/04/2022, modifié le 10/08/2024)

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 sont représentées par des crochets [ ].

Loric Informatique

Table des matières

1. Prise en main

1.1. Outils nécessaires

- UwAmp ou WampServer (contenant MySQL et PhpMyAdmin)
- 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	$-3.4_{x10}{}^{38}$f	$3.4_{x10}{}^{38}$f	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 `<=` *b*	*a* supérieur ou égal à *b*
a `!=` *b*	*a* n'est pas égal à *b* (seulement en contenu : 1='1')
a `<>` *b*	*a* n'est pas égal à *b* (seulement en contenu : 1='1')
a `= 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 `= 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 `IN` *b*	*a* est présent dans *b* (qui peut être une sous-requête ou une liste sous forme : `(val1, val2…)`)
a `BETWEEN` *b* `AND` *c*	*a* est compris entre *b* et *c*

`a LIKE 'SousChaine'`	*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 vraie
`NOT condition`	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 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

3.2. Les fonctions pour les chaînes de caractères

Fonction	Description
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

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(*date*, *nombre* SECOND)	Ajoute un certain nombre de secondes
DATE_SUB(*date*, *nombre* SECOND)	Soustrait un certain nombre de secondes
DATE_ADD(*date*, *nombre* MINUTE)	Ajoute un certain nombre de minutes
DATE_SUB(*date*, *nombre* MINUTE)	Soustrait un certain nombre de minutes
DATE_ADD(*date*, *nombre* HOUR)	Ajoute un certain nombre d'heures
DATE_SUB(*date*, *nombre* HOUR)	Soustrait un certain nombre d'heures
DATE_ADD(*date*, *nombre* DAY)	Ajoute un certain nombre de jours

`DATE_SUB(date, nombre DAY)`	Soustrait un certain nombre de jours
`DATE_ADD(date, nombre WEEK)`	Ajoute un certain nombre de semaines
`DATE_SUB(date, nombre WEEK)`	Soustrait un certain nombre de semaines
`DATE_ADD(date, nombre MONTH)`	Ajoute un certain nombre de mois
`DATE_SUB(date, nombre MONTH)`	Soustrait un certain nombre de mois
`DATE_ADD(date, nombre YEAR)`	Ajoute un certain nombre d'années
`DATE_SUB(date, nombre YEAR)`	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 date (ex : `STR_TO_DATE('10/12/2024', '%d/%m/%Y')`)
`DATE_FORMAT(date, masque)`	Convertit une date en chaîne de caractères (ex : `DATE_FORMAT(NOW(), '%d/%m/%Y')`)

3.4. Autres fonctions

Fonction	**Description**
`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

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

Fonction	Description
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
COUNT(DISTINCT *expression*)	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(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)

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;
```