Les types de données MySQL

Sommaire

Les repères de lecture


Retour au chapitre


Définition


Objectif(s)


Espace Élèves


Vidéo / Audio


Point important / À retenir


Remarque(s)


Pour aller plus loin


Normes et lois


Quiz

Introduction

Les types de données servent à décrire les données enregistrées dans une base de données MySQL qui peuvent être de différentes natures, par exemple des nombres ou des chaînes de caractères. Ces données possèdent donc un type qu'il est obligatoire de spécifier lorsque nous créons les champs d'une table, comme nous l'avons vu lors de l'utilisation de PhpMyAdmin.

MySQL fournit la liste de ces types qui sont classés en deux catégories : champs numériques et chaînes de caractères. Nous listons ci-dessous les champs les plus souvent utilisés.

I. Champs numériques souvent utilisés

A. TINYINT

Le type TINYINT décrit un entier (c'est-à-dire un nombre sans virgule) très petit compris entre 0 et 127. La valeur d'un champ TINYINT ne doit donc pas dépasser 127.

Si l'option UNSIGNED est spécifiée lors de la création du champ, la valeur pour ce nombre est alors comprise entre 0 et 254.

B. SMALLINT

Le type SMALLINT (lire : small int, int pour integer) décrit un entier compris entre -32 768 et 32 767. Si l'option UNSIGNED est spécifiée, cette valeur peut être comprise entre 0 et 65 535.

C. MEDIUMINT

Le type MEDIUMINT décrit un entier compris entre -8 388 608 et 838 860. Si l'option UNSIGNED est spécifiée cette valeur peut être comprise entre 0 et 16 777 215.

D. INT

Le type INT est le plus standard des types numériques. Il décrit un entier compris entre -2 147 483 648 et 2 147 483 647.

Si l'option UNSIGNED est spécifiée, cette valeur peut être comprise entre 0 et 4 294 967.

E. Chaînes de caractères souvent utilisées

1. TEXT

Contenu de type ASCII (casse insensible).

2. VARCHAR

Chaîne de caractères variable. M peut être compris entre 1 et 255.

F. Autres types de champs

Il existe d'autres types de champs qui sont moins utilisés et que nous ne détaillerons pas dans cette leçon. Vous les retrouverez si besoin dans la documentation MySQL. Les voici donc pour information : BIGINT, FLOAT, DOUBLE, REAL, DECIMAL, BLOB, CHAR (n), TINYTEXT, TINYBLOB, MEDIUMTEXT, MEDIUMBLOB, LONGBLOB, LONGTEXT, ENUM, SET.

Manuel MySQL : Liste des champs disponibles
http://dev.mysql.com/doc/refman/5.0/fr/column-types.html

II. Fonctions et opérateurs MySQL

À l'instar de PHP, MySQL possède ses propres fonctions et nous listons ici les plus fréquemment utilisées. La documentation de MySQL est disponible pour la liste complète de toutes les fonctions. Ces fonctions peuvent être utilisées dans les requêtes SELECT avec une clause WHERE.

A. Les opérateurs de calcul

- addition : + ;
- soustraction : - ;
- multiplication : * ;
- division : / ;
- modulo : %.

B. Les opérateurs logiques

- AND ;
- OR ;
- NOT ;
- BETWEEN ;
- IN.

C. Les opérateurs de comparaison

- inférieur strictement à : < ;
- inférieur ou égal à <= ;
- égal à = (notons que l'égalité dans une instruction MySQL repose sur le simple signe égal) ;
- supérieur strictement à : > ;
- supérieur ou égal à : >=.

Ajoutons également les parenthèses ( ) qui permettent de regrouper les instructions ou les opérations. **Tous ces opérateurs et ces fonctions peuvent êtres composés entre eux et fournissent ainsi à MySQL la possibilité de procéder aux calculs et opérations que nous n'aurons ainsi pas besoin d'effectuer en PHP**. MySQL étant généralement plus rapide que PHP, l'avantage en gain de temps est réel. Il est donc préférable pour tout ce qui concerne les informations contenues dans les bases de données que MySQL s'occupe des calculs que nous récupérons ensuite en PHP.

Quelques exemples :

```
SELECT ville_nom FROM villes WHERE ville_id <= 100

SELECT nom FROM salaries WHERE age BETWEEN 20 AND 30

SELECT model_id FROM motos WHERE couleur IN ('bleu', 'blanc', 'noir')

SELECT model_id FROM voitures WHERE couleur NOT IN ('vert', 'jaune')
```

Fig. 1

D. Fonction de comparaison de chaînes

La commande LIKE permet de comparer deux chaînes et est utilisée pour effectuer des recherches dans une table de la base de données.

Par exemple, si nous souhaitons rechercher la ville qui a pour nom Paris :

```
SELECT ville_nom FROM villes WHERE ville_nom LIKE 'paris'
```

Fig. 2

Nous pouvons rechercher aussi les villes dont les noms commencent par « pa » mais la requête LIKE suivante ne donnera aucun résultat.

```
SELECT ville_nom FROM villes WHERE ville_nom LIKE 'pa'
```

Fig. 3

Il faut, en effet, utiliser le caractère spécial % qui a ici une fonction de joker permettant de remplacer n'importe quel caractère. La requête suivante retourne « Paris ».

```
SELECT ville_nom FROM villes WHERE ville_nom LIKE '%pa%'
```

Fig. 4

La syntaxe de cette requête est celle qui est communément mise en oeuvre dans le cadre d'un formulaire de recherche d'un site web PHP.

E. Autres fonctions de MySQL

Il existe encore un grand nombre de fonctions MySQL : les fonctions mathématiques comme ROUND (x) qui arrondit une valeur numérique à l'entier le plus proche, CEILING (x) qui arrondit à l'entier supérieur ou FLOOR (x) qui arrondit à l'entier inférieur. Des fonctions de chaînes, des fonctions de dates et heures etc. Nous retrouverons ces fonctions qu'il est intéressant de connaître dans le manuel MySQL.

Manuel MySQL : Liste des fonctions MySQL
http://dev.mysql.com/doc/refman/5.0/fr/functions.html

En conclusion, nous vous invitons à réaliser l'exercice ci-dessous.

1. Cas pratique 5

Créez un moteur de recherche pour le site des villes qui offre la possibilité de rechercher le nom d'une ville. Ce moteur de recherche doit permettre via un champ de formulaire de restituer sur une page dédiée les résultats de la recherche.

Voici le résultat :

Le formulaire de recherche ne pose pas de problème particulier, il est intégré en HTML en méthode GET, il a pour cible la page resultat.php et contient le champ input de type texte « recherche ».

La structure de la page resultat.php est basée sur celle des autres pages de contenu du site, par exemple la page ville.php. La requête MySQL est un SELECT qui utilise le comparateur LIKE et intègre la saisie de l'utilisateur contenue dans la variable externe.

```
<?php
// récupération de la variable externe
$recherche = $_GET['recherche'];
// requête.
$result = $mysqli->query('SELECT ville_nom FROM villes
                          WHERE ville_nom LIKE "%' . $recherche .'%"');
while ($row = $result->fetch_array())
{
     echo $row['ville_nom'] .'<br>';
}
```

Fig. 5

Soyons ici attentif aux double quotes utilisés avec LIKE car la recherche s'effectue sur un champ VARCHAR.