Les requêtes 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

MySQL est un système complet de gestion de base de données permettant la récupération du contenu avec la requête SELECT mais aussi la modification, la suppression ou l'insertion de ce contenu.

Par ailleurs, les données enregistrées peuvent être de différents types, par exemple des nombres ou des chaînes de caractères, qui permettent de les décrire afin de les gérer au mieux.

Nous allons donc dans cette partie étudier les principales requêtes puis dans la partie suivante les types de données MySQL essentielles que nous serons amenés à utiliser dans nos scripts PHP et que nous pourrons tester dans l'interface de requête de PhpMyAdmin.

I. Tester les requêtes dans PhpMyAdmin

Les requêtes MySQL que nous allons apprendre peuvent être testées dans l'interface de saisie MySQL accessible via l'onglet SQL. Nous nous servirons de la base *projet_villes* pour effectuer ces requêtes. Il convient donc de commencer par se placer dans le contexte de la base de données *projet_ville*. Nous cliquons donc sur le lien *projets_villes* dans la barre de menu de la colonne de gauche de PhpMyAdmin puis vérifions que le fil d'Ariane en haut de page contient bien les liens « localhost > projet_villes »

Fig. 1 Fil d'Ariane indiquant que nous sommes sur la table villes

Nous pouvons ensuite accéder au menu contextuel relatif à la base de données *projet_villes*.

Nous cliquons sur l'élément « SQL » de manière à exécuter nos requêtes en langage MySQL.

L'interface de saisie nous permet par exemple d'écrire la requête dont nous nous sommes servis pour l'exemple des villes :

```
SELECT ville_id, ville_nom FROM villes
```

Fig. 2

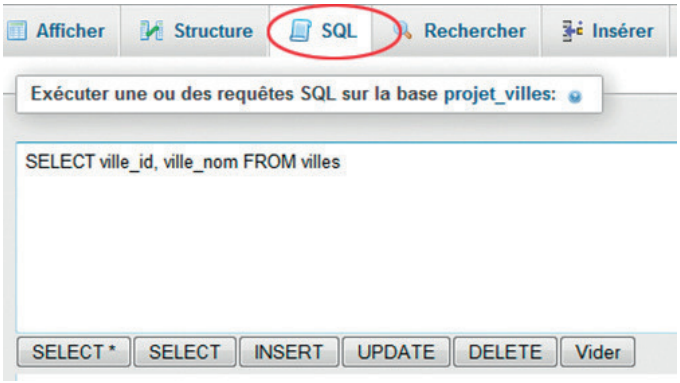

Fig. 3 Interface SQL de PhpMyAdmin permettant d'effectuer des requêtes en MySQL

PhpMyAdmin exécute la requête et retourne bien la liste des villes et des identifiants. Nous utiliserons cette interface pour tester les différentes requêtes que nous allons rencontrer.

II. Requête SELECT

SELECT est utilisé pour obtenir des enregistrements venant d'une ou plusieurs tables.

Nous avons déjà utilisé la requête SELECT dont la syntaxe de base est la suivante :

```
SELECT champ1, champ2 FROM nom_de_la_table
```

Fig. 4

A. Intégration d'un SELECT dans un script PHP

L'intégration d'une requête SELECT dans un script PHP s'effectue après avoir effectué une connexion à la base de données puis affecté la permission de connexion à une variable que nous nommons par convention $msqli. Nous associons ensuite à cette variable la requête souhaitée.

```
$mysqli->query('SELECT champ1, champ2 FROM nom_de_la_table');
```

Fig. 5

Rappelons-nous que les requêtes MySql sont des chaînes de caractères pour PHP.

La syntaxe « mysql› » utilisée dans la documentation est utilisée en mode console seulement, ce qui n'est pas notre cas car nous utilisons cette requête soit dans le code PHP soit avec PhpMyAdmin.

1. Page du manuel MySQL dédiée à SELECT

A l'instar du manuel PHP, le manuel MySQL propose une information détaillée sur les requêtes qu'il convient de consulter : https://dev.mysql.com/

III. La clause WHERE

La **clause** est synonyme de condition. En tant que système de gestion de bases de données, MySQL permet de gérer des requêtes précises. Il est par exemple possible d'obtenir les villes de plus de 3 millions d'habitants en le précisant dans la requête avec la clause (ou condition)

WHERE comme ceci :

```
SELECT ville_nom FROM villes WHERE population > 3000
```

Fig. 6

Avec le mot clef AND, nous pouvons préciser que nous voulons la liste des villes de plus de 3 millions d'habitants et ayant un identifiant supérieur à 3.

```
SELECT ville_nom FROM villes WHERE population > 3000 AND ville_id > 3
```

Fig. 7

Les mots clefs AND et OR ont le même usage en MySQL ou en PHP et indiquent soit une inclusion (ET inclusif) soit une exclusion (OU exclusif).

Comme nous le verrons par la suite, la clause WHERE est utilisable avec l'ensemble des requêtes de MySQL.

IV. Requête DELETE

DELETE est la requête de suppression. Elle s'emploie généralement avec la clause WHERE afin de supprimer un enregistrement précis de la table.

Si nous voulons supprimer la ville qui possède l'identifiant 5, la requête est :

```
DELETE FROM villes WHERE ville_id = 5
```

Fig. 8

Attention : si nous ne spécifions pas la clause WHERE, toutes les lignes de la table seront supprimées.

Remarquons ici que dans la syntaxe MySQL, l'égalité est signifiée avec le signe égal.

```
DELETE FROM villes
```

Fig. 9

Intégration d'un DELETE dans un script PHP

```
$mysqli->query('DELETE FROM villes WHERE ville_id = 5')
```

Fig. 10

Page du manuel MySQL dédiée à DELETE
http://dev.mysql.com/doc/refman/5.0/fr/delete.html

V. Requête INSERT INTO

INSERT INTO est la requête d'ajout de données. La syntaxe de la requête d'insertion repose sur l'utilisation de la commande INSERT INTO suivie du nom de la table et des noms de champs entre parenthèses puis du mot clef VALUES qui indique les valeurs qui seront respectivement ajoutées aux champs de la table. Remarquons que dans la syntaxe MySQL, les valeurs sont écrites entre *quotes*.

Ainsi la syntaxe de base est par exemple :

```
INSERT INTO nom_de_table (champ1, champ2) VALUES ('valeur1', 'valeur2');
```

Fig. 11

Si nous souhaitons par exemple ajouter la ville de Copenhague à la table villes, la requête serait :

```
INSERT INTO villes (ville_nom, population)
VALUES ('Copenhague', '500000');
```

Fig. 12

Si nous souhaitons ajouter une ville sans ajouter la population il suffit simplement d'indiquer le champ ville_nom et comme valeurs le nom de la ville. Seul ce champ sera mis à jour.

```
INSERT INTO villes (ville_nom) VALUES ('Stockholm');
```

Fig. 13

Intégration d'un INSERT INTO dans un script PHP

```
$mysqli->query('INSERT INTO villes (ville_nom, ville_texte)
VALUES ("'.$ville_nom.'", "'.$ville_texte.'")')
```

Fig. 14

Lorsque nous utilisons cette requête, soyons attentif à la syntaxe PHP et notamment aux concaténations et aux *quotes*.

Page du manuel MySQL dédiée à INSERT INTO : http://dev.mysql.com/doc/refman/5.0/fr/insert.html

1. PHP : La fonction insert_id

Utilisée généralement après une requête INSERT INTO, nous voyons ici la fonction PHP *insert_id()* qui est très pratique puisqu'elle permet de récupérer l'identifiant de la clé primaire AUTO_INCREMENT pour la dernière insertion.

Ainsi si nous insérons une nouvelle ville et que nous désirons connaître son identifiant *ville_id* qui est la clef primaire de la table villes, nous aurons le code suivant :

```
$mysqli->query('INSERT INTO villes (ville_nom, ville_texte) VALUES
("'.$ville_nom.'", "'.$ville_texte.'")');
echo $mysqli->insert_id ;// retourne l'identifiant clef primaire
```

Fig. 15

Ce résultat peut notamment être utilisé pour proposer directement un lien vers la nouvelle page de la ville que nous venons d'ajouter.

```
$ville_id_insert = .$mysqli->insert_id;
$message = '<p class="message">L\'ajout de la ville '. $ville_nom
.' est effectué.';
$message .= '<br>. <a href= "ville.php?id='. $ville_id_insert .
'" >Consulter la page de '. $ville_nom .' </a></p>';
echo $message;
```

Fig. 16

VI. Requête UPDATE

UPDATE est la requête permettant de modifier des données déjà enregistrées dans la table.

La syntaxe de cette requête repose sur la commande **UPDATE** et la clause **SET** ainsi que la clause **WHERE**.
La clause SET indique dans quel champ les enregistrements seront à modifier. Si elle est précisée, la clause WHERE spécifie les enregistrements à mettre à jour, sinon tous les enregistrements sont mis à jour.

Nous souhaitons, par exemple, pour la valeur « Stockholm » ajouter cette fois la population.
Nous aurons besoin de la clause WHERE pour spécifier l'identifiant de la ligne en question.

```
UPDATE villes SET population = '800' WHERE ville_id = 6;
```

Fig. 17

Si nous ne spécifions pas l'identifiant avec la clause WHERE, toutes les lignes de la table seront mises à jour avec la même nouvelle valeur.

```
UPDATE villes SET population = '800';
```

Fig. 18

Intégration d'un UPDATE dans un script PHP

```
$mysqli->query('UPDATE villes SET ville_nom = "'.$ville_nom.'",
pays_id= '.$pays_id.' WHERE ville_id = '. $ville_id);
```

Fig. 19

Page du manuel MySQL dédiée à UPDATE : http://dev.mysql.com/doc/refman/5.0/fr/update.html

VII. Autres requêtes

A. CREATE TABLE

La requête CREATE TABLE permet de créer une table.

Voyons la syntaxe avec un exemple théorique où nous créons une table *nouvelle_table* dans la base *projet_villes*.

```
CREATE TABLE projet_villes.nouvelle_table
(
     id int AUTO_INCREMENT,
     data varchar(20),
     PRIMARY KEY (id)
)
```

Fig. 20

La première ligne contient la commande de création de la table avec **CREATETABLE** suivi du nom de la base de données, d'un point et du nom de la nouvelle table. Le point sert à indiquer que la nouvelle table est une table de la base projet_villes.

Entre parenthèses, nous définissons ensuite les champs de la table :

– Un champ id qui est un entier INT et qui est AUTO_INCREMENT, ce qui, rappelons-le, signifie que MySQL mettra à jour automatiquement ce champ pour les nouveaux enregistrements de la table.

– Un champ data que nous spécifions comme étant de type VARCHAR et contenant au maximum 20 caractères.

– Et nous précisons que le champ id sera la clef primaire (PRIMARY KEY).

Bien que cela soit optionnel dans le cadre de ce cours, nous pourrions également préciser :

– le moteur de la base de données avec la commande ENGINE à laquelle nous affectons la valeur InnoDB. Il s'agit d'un moteur de base de données standard de MySQL ;

– l'encodage de la table si celui-ci ne devait pas être le même que celui de la base avec la commande DEFAULT CHARSET à laquelle nous affectons la valeur utf8 qui est également une donnée standard.

```
ENGINE = InnoDB DEFAULT CHARSET = utf8;
```

Fig. 21

B. TRUNCATE TABLE

Si nous voulons supprimer toutes les valeurs enregistrées dans une table, nous utilisons la commande TRUNCATE comme ceci :

```
TRUNCATE TABLE nom_de_la_table
```

Fig. 22

Notons que le résultat sera identique à celui fourni par une requête DELETE sans la clause WHERE.

C. DROP TABLE

Enfin, si nous voulons supprimer une table, nous utilisons la commande DROP comme ceci :

```
DROP TABLE nom_de_la_table
```

Fig. 23