Cour PHP Expert résumé

LES Sommaire De 01 a 05

01MySQL mode console et jointures

Sommaire

02Utiliser PDO et les Requêtes Préparées

Sommaire

E. Cas d'une requête avec deux variable ou plus

III. Gestion des erreurs

A. Utilisation try, throw et catch, les exceptions

B. Gestion des erreurs MySQL

Voire suite

Suite cour PHP expert résumé

03POO avancée : le système de brèves

Sommaire

I. Structure du contenu

II. Le besoin : création et affichage d'une brève

A. Démarrer en définissant les attributs de classe

B. Des méthodes pour accéder et modifier les attributs

C. Retour sur le constructeur

D. Une autre class spécifique pour la relation avec la base de données

04POO avancée : le CRUD

Sommaire

I. La méthode de lecture ou de récupération des brèves : *getBreve*

II. La méthode de suppression : *deleteBreve*

III. Testons la classe

05POO avancée : L'héritage : gérons un nouveau type de contenu

Sommaire

Cour 01 cour PHP expert

MySQL, mode console et jointures

IL est utile de savoir utiliser MySQL en mode console afin de pouvoir se passer des utilitaires en mode graphique comme : 1- PhpMyAdmin, 2- Atelier MySQL, 3- Le castor, 4- Atelier de l'apiculteur, 5- Administrateur, 6- Navicat pour MySQL, 7- OmniDB,8- Écureuil SQL. Ceci et une lites 8 outils d'interface graphique MySQL/MariaDB pour les administrateurs Linux.

Les modes graphiques peuvent ne pas être disponibles dans le cadre d'une intervention ou d'un projet.

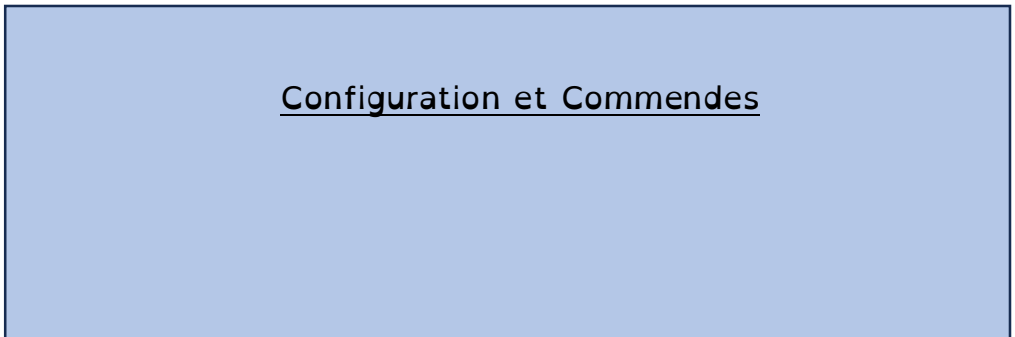
Les avantage du mode console :

Toujours disponible, l'usage du mode console permet de bien travailler ses requêtes et ainsi d'améliorer ses connaissances.

Les performances de MySQL sont meilleures en mode console car le traitement PHP de PhpMyAdmin n'est de fait pas utiliser.

I. Accéder au mode console avec wamp « Wanpserver64 »

Le mode console de MySQL s'active depuis le menu de wamp «Wanpserver64» PhpMyAdmin, via l'élément de liste intitulé « MySQL » qui ouvre le sous-menu contenant l'élément de liste « Console MySQL ». Il suffit de cliquer sur cet élément pour ouvrir la console.



- Il et possible de configure la fenêtre de console

- Enregistrer//récupérer ses commandes sur un fichier texte
  TEE chemin_ver_le_fichier ;
  mysql> TEE C:/Users/tripl/Desktop/PhpExpert/Recup-dbtest.txt;
  Repone:
  Logging to file 'C:/Users/tripl/Desktop/PhpExpert/Recup-dbtest.txt'

- Requête pour lister les basses de donner
  MySQL> SHOW DATABASES;
  Réponse MySQL affiche toutes les bases

Création ou Connexion

- Requête pour création de notre table:
  MySQL> CREATE DATABASE dbtest CHARATER SET utf8 COLLATE utf8_general_ci;

  Detaille des commandes :

  #Voire➔ II. Création ➔ A. création des tables

- **Connexion** : Pour travailler sur notre base la requête :
  MySQL> USE non_de_la base ;
  Pour se cour le nom de la base et : dbtest !
  MySQL> USE dbtest;
  MySQL répond : Database changed

- lister les tables de la base avec l'instruction :
  MySQL> SHOW TABLES
  Réponse MySQL affiche no tables

- Accéder à la structure des tables avec l'instruction :
  DESCRIBE nom_de_la_table
  MySQL> DESCRIBE auteur ;
  MySQL> DESCRIBE livre ;

- Afficher les données avec la commande suivante :
  MySQL> SELECT * FROM auteur ;
  MySQL> SELECT * FROM livre;

II. Création de la base de données

Voici le code MySQL :

```
CREATE DATABASE dbtest CHARATER SET utf8 COLLATE utf8_general_ci;
```

L'encodage utf8 permet d'inclure tout type de contenu accentué, c'est une norme qu'il est nécessaire d'utiliser pour tous nos développements. L'option d'encodage (collation) qui est choisie est utf8_general_ci, les lettres ci signifiant *case insensitive*, ce qui signifie donc insensible à la casse ce qui permet donc de récupérer des résultats pour une recherche par exemple de livre ou de LIVRE.

Pour travailler sur une base, il faut indiquer à MySQL quelle base nous voulons utiliser au moyen de USE: USE non_de_La_Base

Voici le code MySQL :

```
USE dbtest ;
```

A. création des table

Voici le code MySQL :

```
-- Structure de la table `auteur`

mysql> CREATE TABLE IF NOT EXISTS `auteur` (
   -> `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
   -> `nom` varchar(10) DEFAULT NULL,
   -> PRIMARY KEY (`id`)
   -> ) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=5 ;

-- Contenue de la table `auteur`

mysql> INSERT INTO `auteur`(`id`, `nom`) VALUES
   -> (1, 'Claude'),
   -> (2, 'Marie'),
   -> (3, 'Pierre'),
   -> (4, 'Jean');

-- Structure de la table `livre`

mysql> CREATE TABLE IF NOT EXISTS `livre`(
   -> `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
   -> `titre` varchar(20) DEFAULT NULL,
   -> `auteur_id` smallint(10) unsigned NOT NULL,
   -> PRIMARY KEY (`id`)
   -> ) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=7 ;
```

-- Contenue de la table `livre`

```
mysql> INSERT INTO `livre`(`id`, `titre`, `auteur_id`) VALUES
  -> (1, 'PHP et Mysql', 2),
  -> (2, 'Apprendre PHP', 2),
  -> (3, 'Cours de SQL', 3),
  -> (4, 'PHP par la pratique', 4),
  -> (5, 'Projet PHP', 2),
  -> (6, 'PHP et Ajax', 3);
```

l'instruction USE dbtest MySQL répond: Database changed. Ce qui signifie que nous travaillons bien désormais sur cette base de données .
lister les tables de la base avec l'instruction SHOW TABLES.

Récupérer les commandes et leur resulta avec l'intrusion TEE

TEE chemin_ver_le_fichioer ;

EX pour créer un fichier sur le bureau : TEE C:/Users/tripl/Desktop/superbDoc.txt;

MySQL répond alors Logging to file 'C:/Users/tripl/Desktop/superbDoc.txt' et l'ensemble des requêtes saisies et des résultats retournés seront donc écrits dans ce fichier.

Si nous souhaitons arrêter cette écriture dans le fichier il faut utiliser l'instruction NOTEE; ce à quoi MySQL répond Outfile disabled.

B. Les requêtes de jointures.

#Voire ➔C. Les alias ➔D. INNER JOIN, jointure inter

Les requêtes de jointure son utiliser dès que nous avons une relation entre les données de deux tables. Une jointure est en effet un rapprochement, une fusion et dans ce cas, joindre signifie alors relier.

Dans notre exemple, la table livre contient l'identifiant de l'auteur qui a écrit le livre : *auteur_id* et une première requête de jointure consistera alors à afficher les livres écrits par un auteur donné.

```
mysql> SELECT id FROM auteur WHERE nom LIKE '%marie%';
```

Resulta : id 2

Si nous cherchons les livres dont l'auteur est Marie, il nous faudrait effectuer deux requêtes:

```
mysql> SELECT titre FROM livre WHERE auteur_id = 2;
```

resulta : PHP et Mysql ; Apprendre PHP ; Projet PHP

- DEUX REQUETE POUR UN RESULTA

```
mysql> SELECT id FROM auteur WHERE nom LIKE '%marie%';
```
Resulta : id 2
```
mysql> SELECT titre FROM livre WHERE auteur_id = 2;
```
resulta : PHP et Mysql .Apprendre PHP. Projet PHP.

Avec les jointures, nous pourrons effectuer cette recherche en une seule requête.

C. Les alias

Un alias permet de renommer une table durant une requête.

Ce qui permet de distinguer les résultats, notamment pour deux tables ayant chacune une colonne id.

Soit, par exemple, la requête suivante destinée à afficher toutes les valeurs des colonnes id. des deux tables auteur et livre:

```
SELECT id , id FROM auteur , livre ;
```

MySQL ne sait alors pas à quelle colonne associer les résultats car elles ont le même nom et sa réponse est alors: UNE ERROR

```
ERROR 1052 (2300) : Column `ID' in fidel list is ambiguous
```

Les alias vont nous permettre de bien distinguer les colonnes en spécifiant que A est l'alias de la table auteur et que L est l'alias de la table livre:

```
mysql> SELECT A.id , L.id FROM auteur AS A , livre  AS L;
```

A.id. fait donc bien référence à la colonne de la table auteur dont l'alias est A tandis que L.id. fait bien référence à la colonne de la table livre dont l'alias est L.

Même si les noms des colonnes ne sont pas identiques, utiliser un alias est toujours une bonne idée afin de bien structurer nos requêtes.

À noter que le mot-clé AS est facultatif, il est cependant fortement recommandé de toujours l'employer.

D. INNER JOIN, jointure interne

Pour travailler sur ces deux tables, nous utilisons comme ceci la jointure interne INNER JOIN afin d'obtenir le nom de l'auteur et les titres de ses livres:

```
mysql> SELECT * FROM auteur AS A
```
Obtenir les valeurs depuis la table auteur (pour laquelle nous utiliserons l'alias A).

```
  -> INNER JOIN livre AS L
```
En la joignant (fusionnant) avec la table livre (pour laquelle nous utiliserons l'alias L) dont nous récupérerons aussi les valeurs.

```
  -> ON A.id = L.auteur_id
```
Et en effectuant cette jointure sur le fait que l'identifiant de la table auteur (A étant l'alias de la table auteur) est le même que auteur_id. de la table livre (L étant l'alias de la table livre).

```
  -> WHERE A.id = 2;
```
Lorsque l'identifiant de l'auteur a comme valeur 2.

Nous voyons que la jointure INNER JOIN s'exprime avec l'opérateur ON qui permet de spécifier les valeurs que nous allons utiliser: la colonne id. de la table auteur et la colonne auteur_id. de la table livre, chacune de ces colonnes devant contenir les mêmes valeurs.

```
mysql> SELECT * FROM auteur AS A
  -> INNER JOIN livre AS L
  -> ON A.id = L.auteur_id
  -> WHERE A.id = 2;
```

```
+----+-------+----+--------------------+-------------+
| id | nom | id | titre                | auteur_id |
+----+-------+----+--------------------+-------------+
| 2 | Marie | 1 | PHP et Mysql     |       2        |
| 2 | Marie | 2 | Apprendre PHP |       2        |
| 2 | Marie | 5 | Projet PHP         |       2        |
+----+-------+----+--------------------+-------------+
```

Nous constatons que ce résultat est bien constitué par la jointure:

– des deux colonnes de la table auteur;

– et des trois colonnes de la table livre.

Nous pouvons également effectuer une requête plus générale et obtenir la liste des auteurs de chaque livre, nous ne spécifierons donc pas de clause WHERE.

```
mysql> SELECT * FROM auteur AS A
   -> INNER JOIN livre AS L
   -> ON A.id = L.auteur_id ;
```

Ou

```
mysql> SELECT * FROM auteur AS A INNER JOIN livre AS L ON A.ID = L.auteur_id;

mysql> SELECT * FROM auteur AS A INNER JOIN livre AS L ON A.id = L.auteur_id;
```

RESULTA & REQUETTE

Resulta : mysql> SELECT * FROM auteur AS A INNER JOIN livre AS L ON A.ID = L. auteur_id;

id	nom	id	titre	auteur_id
2	Marie	1	PHP et Mysql	2
2	Marie	2	Apprendre PHP	2
3	Pierre	3	Cours de SQL	3
4	Jean	4	PHP par la pratique	4
2	Marie	5	Projet PHP	2
3	Pierre	6	PHP et Ajax	3

6 rows in set (0.00 sec)

La jointure INNER JOIN relie donc les deux tables tout en étant obligée d'obtenir des données pour chaque table.

Il n'y a donc pas de valeurs vides (NULL) dans le résultat.

L'obtention de valeurs vides sera possible uniquement avec les jointures externes.

E. LEFT OUTER JOIN et RIGHT OUTER JOIN, les jointures externes

Les jointures externes offrent donc la possibilité de récupérer des valeurs vides.

Une jointure externe est constituée par la commande **OUTER JOIN** que l'on préfixe avec **LEFT** ou **RIGHT** pour spécifier si nous souhaitons toutes

les lignes de la table qui est mentionnée à gauche ou à droite dans l'écriture de la requête.

LEFT OUTER JOIN peut s'écrire LEFT JOIN. RIGHT OUTER JOIN peut s'écrire RIGHT JOIN.

E suit. LEFT OUTER JOIN et RIGHT OUTER JOIN, les jointures externes

Commande les jointures externes	
SELECT * FROM auteur LEFT JOIN livre ON auteur.id = livre.auteur_id.;	La table auteur est à gauche de LEFT JOIN, et puisque nous utilisons LEFT alors nous récupérons toutes les lignes de la table auteur même si elles n'ont pas toutes des correspondances dans la table livre.
SELECT * FROM auteur RIGHT JOIN livre ON auteur.id = livre.auteur_id.;	La table livre est à droite de RIGHT JOIN, et puisque nous utilisons RIGHT alors nous récupérons toutes les lignes de la table livre même si elles n'ont pas toutes des correspondances dans la table auteur.

Résultat pour la première requête:

```
mysql> SELECT * FROM auteur AS A LEFT JOIN livre AS L ON A.id = L.auteur_id;
```

id	nom	id	titre	auteur_id
2	Marie	1	PHP et Mysql	2
2	Marie	2	Apprendre PHP	2
3	Pierre	3	Cours de SQL	3
4	Jean	4	PHP par la pratique	4
2	Marie	5	Projet PHP	2
3	Pierre	6	PHP et Ajax	3
1	Claude	NULL	NULL	NULL

Comme nous le voyons, nous récupérons l'auteur Claude qui n'a écrit aucun livre et le résultat correspondant dans la table livre. Ce résultat retourné est NULL, c'est-à-dire qu'il n'existe pas de valeur correspondante (comme nous l'avons déjà vu, NULL est l'absence de valeurs).

Le résultat serait identique pour la requête RIGHT JOIN qui intervertirait les noms des tables (la table auteur est cette fois à droite), même si dans ce cas les colonnes de la table livre sont retournées en premier:

```
mysql> SELECT * FROM livre AS L RIGHT JOIN auteur AS A ON A.id = L.auteur_id;
```

id	titre	auteur_id	id	nom
1	PHP et Mysql	2	2	Marie
2	Apprendre PHP	2	2	Marie
3	Cours de SQL	3	3	Pierre
4	PHP par la pratique	4	4	Jean
5	Projet PHP	2	2	Marie
6	PHP et Ajax	3	3	Pierre
NULL	NULL	NULL	1	Claude

F. Les jointures sur plusieur tables

Il est possible d'effectuer des jointures sur plusieurs tables pour établir des relations entre des données qui ne sont pas directement stipulées, comme, un auteur et un thème.

Ajouton a notre base donnée la tables thème. Thème d'un des ouvrages de la base contient quatre lignes 1,`PHP' 2, `Mysqul' 3, Àjax' 4, `HTML' :

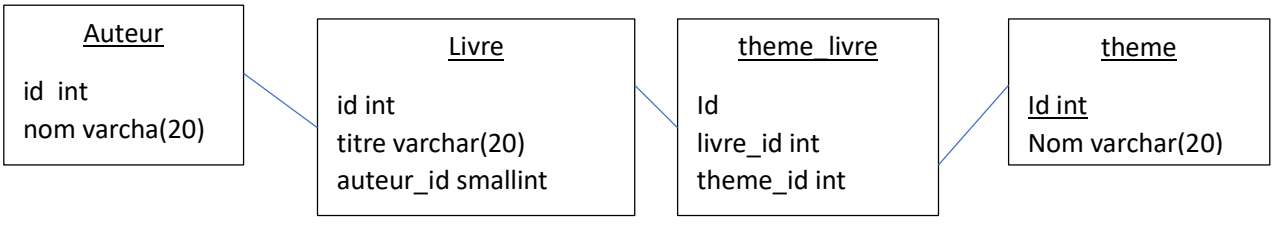
```
CREATE TABLE IF NOT EXISTS `theme` (
`id` int(11) NOT NULL AUTO_INCREMENT,
```

```
`nom` varchar (20) NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=5;

INSERT INTO `theme` (`id`,`nom`) VALUES
(1,'PHP'), (2,'MySQL'),(3,'Ajax'),(4,'HTML');
```

Cette fois, la règle que nous édictons est qu'un livre peut être associé à plusieurs thèmes, par exemple, le livre « PHP et Mysql » a deux thèmes: PHP et MySQL. Puisque nous ne pouvons donc pas utiliser de colonne livre_id. dans la table theme nous allons créer une table theme_livre qui nous servira de table associant les identifiants (on parle alors de table associative).

```
mysql> CREATE TABLE IF NOT EXISTS `theme_livre` (
   -> `id` int(11) NOT NULL AUTO_INCREMENT,
   -> `livre_id` int(11) NOT NULL,
   -> `theme_id` int(11) NOT NULL,
   -> PRIMARY KEY (`id`)
   -> ) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=9;

mysql> INSERT INTO `theme_livre` ( `id`, `livre_id`, `theme_id` ) VALUES
   -> (1, 1, 1), (2, 2, 1), (3, 4, 1), (4, 5, 1), (5, 6, 1), (6, 1, 2), (7, 3, 1), (8, 6, 3);
```

Cette nouvelle table permet d'associer un ou plusieurs thèmes à un livre, par exemple le livre « PHP et Ajax » dont l'identifiant id. est 6 est ainsi associé aux thèmes PHP (theme_id. = 1) et Ajax (theme_id. = 3).



Comme nous le voyons la table associative theme_livre permet de relier par leurs identifiants respectifs livre_id. et theme_id. les données de la table livre et celles de la table theme.

Pour récupérer les livres et les thèmes correspondants, nous procédons à une requête contenant une jointure externe:

```
mysql>SELECT L.titre, T.nom FROM livre AS L
        INNER JOIN theme_livre AS J
        ON L.id = J.livre_id
        INNER JOIN theme AS T
        ON T.id = J.theme_id;
```

titre	nom
PHP et Mysql	PHP
Apprendre PHP	PHP
PHP par la pratique	PHP
Projet PHP	PHP
PHP et Ajax	PHP
PHP et Mysql	MySQL
Cours de SQL	PHP
PHP et Mysql	MySQL

8 rows in set (0.00 sec)

Si nous souhaitons obtenir pour chaque auteur les thèmes abordés dans ses livres, nous pouvons effectuer une succession de trois jointures (en nous servant du schéma précédent):

```
mysql> SELECT A.nom, T.nom
   -> FROM auteur AS A
   -> INNER JOIN livre AS L
   -> ON A.id = L.auteur_id
   -> INNER JOIN theme_livre AS J
   -> ON L.id = J.livre_id
   -> INNER JOIN theme AS T
   -> ON T.id = J.theme_id;
```

Nous établissons une jointure entre la table auteur et la table livre, afin de définir la relation auteur/livre, puis entre la table livre et la table associative theme_livre puis entre la table associative theme_livre et la table theme, afin de définir la relation livre/theme. Nous obtenons ainsi la relation entre les auteurs et les thèmes

```
+--------+-------+
| nom    | nom   |
+--------+-------+
| Marie  | PHP   |
| Marie  | PHP   |
| Jean   | PHP   |
| Marie  | PHP   |
| Pierre | PHP   |
```

```
| Marie  | MySQL |
| Pierre | PHP   |
| Pierre | Ajax  |
+--------+-------+
8 rows in set (0.00 sec)
```

Nous obtenons huit résultats et constatons que des auteurs sont affichés plusieurs fois en étant associés aux mêmes thèmes.

Améliorons notre requête afin d'en supprimer les doublons de résultat avec le mot-clef **DISTINCT** et en ordonnant les résultats par ordres alphabétiques avec une clause **ORDER BY.** Nous spécifions également un alias mais cette fois pour les colonnes afin d'obtenir des intitulés plus clairs (nom auteur et nom theme).

Nous utilisons ici les quotes car ces alias contiennent des espaces.

```
mysql> SELECT DISTINCT A.nom AS 'nom auteur', T.nom AS 'nom theme'
    -> FROM auteur AS A
    -> INNER JOIN livre AS L
    -> ON A.id = L.auteur_id
    -> INNER JOIN theme_livre AS J
    -> ON L.id = J.livre_id
    -> INNER JOIN theme AS T
    -> ON T.id = J.theme_id
    -> ORDER BY A.nom, T.nom;
+------------+-----------+
| nom auteur | nom theme |
+------------+-----------+
| Jean       | PHP       |
| Marie      | MySQL     |
| Marie      | PHP       |
| Pierre     | Ajax      |
| Pierre     | PHP       |
+------------+-----------+
5 rows in set (0.00 sec)
```

Cette fois aucun doublon de résultat n'est retourné et les intitulés de

colonnes sont tout à fait explicites.

Les mots réservés de MySQL

Comme dans tout langage informatique et comme en PHP, il est à noter que des mots sont réservés pour MySQL et qu'il n'est donc pas permis de les utiliser.

Leur usage par méconnaissance de cette règle peut générer des erreurs et il est recommandé de vérifier si les noms des bases, des tables ou des colonnes ne font pas partie de la liste des mots réservés. La liste des mots réservés par MySQL est disponible ici:

https://dev.mysql.com/doc/refman/en/keywords.html